Database Systems

A Collection of Exam Problems

University of Washington, 2000–2026

Dan Suciu

Fall 2026

Bug report: `https://forms.gle/Ddtr2oHSMTkEiBuA7`

Foreword

This is a collection of questions from exams that I gave in the database courses at the University of Washington between 2000 and 2026. It draws on every midterm and final I was able to recover from five courses: the undergraduate courses CSE 344 and CSE 414, the majors course CSE 444 that predated 344 until 2010, and the graduate courses CSE 544 and CSE 544p.

The problems are grouped by topic rather than by exam, so that all the questions on, say, functional dependencies or query optimization sit together in the same section, and they can be worked through in whatever order suits you. This partition is a best effort only: some topics are overlapping, and questions that could haven been placed in more than one section were placed in one arbitrarily.

Each problem is annotated with the course and quarter it came from; a problem that was given more than once lists every term in which it appeared. When questions were used on multiple exams I made an effort to remove duplicates, but some repetitions may still occur. I tried to keep the original wording in most cases, and only fixed grammar and a few typos. I have added explanations to solutions of more difficult questions. This is an evolving document and I plan to further fix errors or improve answers based on feedback.

Problems drawn from the graduate courses (CSE 544 and CSE 544p) tend to be generally harder than what is expected in CSE 344 or CSE 414. The material itself has moved a good deal over that span: the earliest exams ask about XML and DTDs, later the material covered JSON and parallel query processing, and these topics are no longer covered in more recent offerings of these courses.

This document is produced in two versions: one with solutions and one without, so that you can attempt a problem before reading its answer.

Dan Suciu
Seattle, 2026

Contents

1 SQL (1–26)

Almost every exam in these courses opened with SQL, and this is by some margin the largest section of the book. A typical problem gives you a small schema together with an English description of what to compute, and asks for a single SQL query.

The recurring difficulties are worth naming in advance. Aggregation with `group by` and `having` accounts for most of the questions, and many of them turn on the difference between counting rows and counting distinct values. Others require a subquery, and the choice between `in`, `exists` and a join is rarely arbitrary: duplicates and `NULL`s behave differently in each. Questions involving quantifiers ask for *all* or *no* elements of a group. These queries need either a double negation or a comparison of two counts. Finally, a good number ask for an outer join, usually because the expected answer must include entities with no matching rows at all.

Unless a problem says otherwise, assume the standard SQL semantics with duplicates, and remember that a query returning the right answer on the sample data may still be wrong in general.

1. *(from CSE 444, Autumn 2000, Midterm)*

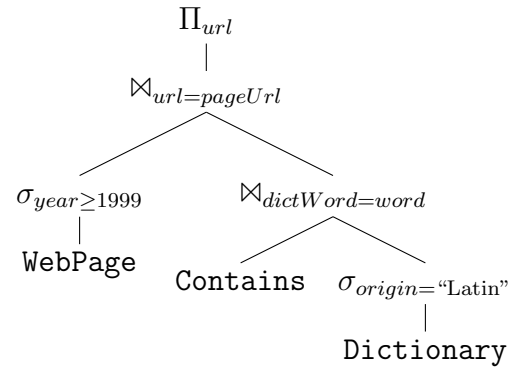
   Consider the following relational schema:

   ```
   WebPage(url, year, author)
   Contains(pageUrl, dictWord)
   Dictionary(word, origin)
   ```

   where `pageUrl` is a foreign key in `WebPage`, and `dictWord` is a foreign key in `Dictionary`.

   (a) Write a Relational Algebra expression tree that retrieves all urls of `WebPage`s created in 1999 or later and containing some word of "Latin" origin.

   Solution:

   (b) Retrieve the urls of all `WebPage`s that contain only words of "Old English" origin.

   Solution:

   ```
   select distinct url
   from WebPage
   where 'Old English' = ALL (select origin
                              from Contains, Dictionary
                              where url = pageUrl and dictWord = word)
   ```

   (c) An author's vocabulary is the set of all words she used in all pages she created. Retrieve all authors with vocabularies of at least 10,000 distinct words.

   Solution:

   ```
   select author
   from WebPage, Contains
   where url = pageUrl
   group by author
   having count(distinct dictWord) >= 10000
   ```

   (d) Retrieve all pairs of urls whose corresponding web pages have at least 50 distinct words in common.

Solution:

```
select x.url, y.url
from Contains x, Contains y
where x.dictWord = y.dictWord
group by x.url, y.url
having count(distinct x.dictWord) >= 50
```

2. *(from CSE 544, 2001, Final)*

We have a database on Web documents:

```
Doc(url, date)
Word(word, language)
Occurs(url, word, bytePos)
```

Write SQL statements that do the following:

(a) Find the urls of all documents that contain some Spanish word.

Solution:

```
select distinct Doc.url
from Doc, Occurs, Word
where Doc.url = Occurs.url and Occurs.word = Word.word and
      Word.language = "Spanish"
```

The join with `Doc` is necessary, since we have not said that `url` in `Occurs` is a foreign key in `Doc`.

(b) Find the urls of all documents that contain only Spanish words.

Solution:

```
select distinct Doc.url
from Doc
where Doc.url not in
        (select Occurs.url
         from Occurs, Word
         where Occurs.word = Word.word and Word.language != "Spanish")
```

Note that an empty document contains only Spanish words, hence should be included. There is no need to join with `Doc` again in the inner select, since $A - A * B = A - B$.

(c) Find the urls of all documents which do not have any Spanish word in the first 300 bytes.

Solution:

```
select distinct Doc.url
from Doc
where url not in (select Occurs.url
                  from Occurs, Word
                  where Occurs.word = Word.word and
                        Word.language = "Spanish" and
                        Occurs.bytePos <= 300)
```

(d) Find all pairs of distinct urls that contain at least 200 distinct words in common.

Solution:

```
select d1.url, d2.url
from Doc d1, Doc d2, Occurs o1, Occurs o2
where d1.url = o1.url and o1.word = o2.word and o2.url = d2.url
group by d1.url, d2.url
having count(distinct o1.word) >= 200
```

Using `count(*) >= 200` is wrong: it includes pairs of documents that have few words in common but occurring in many positions.

3. *(from CSE 444, Autumn 2001, Midterm)*

Consider the following relational schema:

```
User(login, name, office)
Relationship(uLogin, fName, type)
File(name, size)
```

The file size is in bytes. The `type` of a `Relationship` is one of `"owner"`, `"readPermission"`, `"writePermission"`; their meaning is obvious.

(a) Write a relational algebra expression equivalent to the query below. You may represent the expression as a tree.

```
SELECT name
FROM   File
WHERE  name not in (SELECT File.name
                    FROM User, Relationship, File
                    WHERE User.name="Smith" AND
                          User.login=Relationship.uLogin AND
                          Relationship.type="owner" AND
                          Relationship.fName=File.name)
```

Solution:

$\Pi_{name}(\texttt{File}) - \Pi_{name}\big(\sigma_{name=\text{“Smith”}}(\texttt{User}) \bowtie (\sigma_{type=\text{“owner”}}(\texttt{Relationship}) \bowtie \texttt{File})\big)$

(b) Write a SQL query that finds all user names that have write permissions to some file owned by "Mary" and share the same office with her.

Solution:

```
select x1.name
from User x1, Relationship y1, User x2, Relationship y2, File z
where x1.login = y1.uLogin and y1.type = "writePermission"
  and y1.fName = z.name
  and x2.login = y2.uLogin and x2.name = "Mary"
  and y2.type = "owner" and y2.fName = z.name
  and x1.office = x2.office
```

(c) For each user that owns files whose total size is more than 5MB, return their names, total number of files owned, and total size of those files.

Solution:

```
select User.name, count(File.name), sum(File.size)
from User, Relationship, File
where User.login = Relationship.uLogin
  and Relationship.type = "owner"
  and Relationship.fName = File.name
group by User.name
having sum(File.size) > 5000000
```

(d) We add the constraint that every user must have read permissions to all files to which she has write permissions. Write SQL statement(s) to insert all needed read permissions to satisfy this constraint.

Solution:

```
INSERT INTO Relationship(uLogin, fName, type)
   SELECT x.login AS uLogin, z.name AS fName,
          "readPermission" AS type
   FROM   User x, Relationship y, File z
   WHERE  x.login = y.uLogin AND y.fName = z.name
      AND y.type = "writePermission"
      AND not exists (SELECT *
                      FROM Relationship y2
                      WHERE x.login = y2.uLogin AND
                            y2.fName = z.name AND
                            y2.type = "readPermission")
```

(e) **Bonus question.** Retrieve all users who have read permissions only to files owned by "Smith".

Solution:

```
select x.name
from User x
where not exists (select y.name
                  from Relationship z, File y
                  where x.login = z.uLogin AND z.fName = y.name AND
                        z.type = "readPermission" AND
```

```
                    not exists (select *
                                from Relationship u, User v
                                where v.name = "Smith" AND
                                      u.uLogin = v.login AND
                                      u.type = "owner" AND
                                      u.fName = y.name))
```

4. *(from CSE 544, 2002, Final)*

   A document d is a set of words, $d = \{w_1, \ldots, w_n\}$. We ignore the word order in the document. Consider a database of French documents, English documents, and a French/English dictionary. The dictionary associates a confidence factor between 0 and 1 to a pair consisting of a French and an English word. The schema is:

   ```
   FrenchDocs(fID, fWord)
   EnglishDocs(eID, eWord)
   Dictionary(fWord, eWord, confidence)
   ```

   Each document has a unique identifier (either an `eID` or `fID`). Thus, given a document identifier, the corresponding document is the collection of words associated with that identifier.

   (a) For each French document, count how many words it contains.

   Solution:
   ```
   select x.fID, count(x.fWord)
   from   FrenchDocs x
   group by x.fID
   ```

   (b) Find all other English documents that have at least 100 words in common with the English document `eID = "e424"`.

   Solution:
   ```
   select distinct x.eID
   from EnglishDocs x, EnglishDocs y
   where y.eID = 'e424' and x.eWord = y.eWord
   group by x.eID, y.eID
   having count(distinct x.eWord) > 100
   ```

   (c) Given two words `fWord`, `eWord`, define their correlation $c(\texttt{fWord}, \texttt{eWord})$ to be the unique number c such that $(\texttt{fWord}, \texttt{eWord}, c)$ is in `Dictionary`, or 0 if they don't occur in the dictionary. Given two documents $d_1 = \{w_1, \ldots, w_m\}$ and $d_2 = \{v_1, \ldots, v_n\}$, define their correlation to be $\sum_{i,j} c(w_i, v_j)/\sqrt{m \cdot n}$. We say that `fID` is a *translation* of `eID` if their correlation is ≥ 0.9. Write a SQL query that retrieves all French documents that are translations of `"e424"`. Assume that the DBMS supports `sqrt`.

Solution:

```
select distinct x.fID
from FrenchDocs x, Dictionary y, EnglishDocs z
where x.fWord = y.fWord and y.eWord = z.eWord and z.eID = 'e424'
group by x.fID, z.eID
having sum(confidence) > 0.9 * sqrt(count(distinct x.fWord)
                                    * count(distinct z.eWord))
```

(d) Given a French word w, define its translation $w' = t(w)$ to be the English word such that $(w, w', 1.0)$ is in `Dictionary`; we assume that for each w there exists at most one such w'. Given a document $D = \{w_1, \ldots, w_n\}$, define its translation to be $t(D) = \{t(w_1), \ldots, t(w_n)\}$ (there may be fewer than n words, if some of the $t(w_i)$ values are undefined). Write a SQL query that inserts an English translation of the French document `"f822"` into the database. Call the new English document `"e916"`.

Solution:

```
insert into EnglishDocs(eID, eWord)
select 'e916' as eID, y.eWord
from FrenchDocs x, Dictionary y
where x.fID = 'f822' and x.fWord = y.fWord and y.confidence = 1.0
```

(e) The following conjecture has been suggested by an expert familiar with the application domain: "all English documents that contain the words 'vacation' and 'summer' also contain the words 'hiking' and 'biking'." Write a SQL query to prove or disprove this conjecture. Explain how the result of the SQL query can be used to prove or disprove the conjecture.

Solution:

```
select distinct x.eID
from   EnglishDocs x, EnglishDocs y
where  x.eID = y.eID and x.eWord = 'vacation'
       and y.eWord = 'summer'
       and not exists (select z.eID
                       from EnglishDocs z, EnglishDocs u
                       where x.eID = z.eID and z.eID = u.eID
                             and z.eWord = 'hiking'
                             and u.eWord = 'biking')
```

We run this query: if the answer is empty, then the conjecture is correct; otherwise, the query returns all documents that violate the conjecture.

5. *(from CSE 444, Autumn 2002, Final)*

   A Web search engine maintains a database with the following tables:

   ```
   Webpage(url, word)
   Dictionary(word, language)
   ```

   A webpage is uniquely identified by its URL. The relation `Webpage` represents a many-to-many relationship from the set of urls to the set of words: one url may contain multiple words, and conversely one word may occur in multiple urls. The table `Dictionary` consists of a many-to-many relationship between words and languages; here too, one word may belong to multiple languages. We say that a webpage is written in language L if all its words are in the language L (a page may happen to be in multiple languages).

   (a) Compute for each url the total number of words in that webpage.

   Solution:

   ```
   select x.url, count(*)
   from webpage x
   group by x.url
   ```

   (b) Retrieve all pairs of distinct urls which have more than 1000 words in common.

   Solution:

   ```
   select x.url, y.url
   from webpage x, webpage y
   where x.word = y.word
   group by x.url, y.url
   having count(*) > 1000
   ```

   (c) Retrieve the urls of all pages written in "French".

   Solution:

   ```
   select distinct x.url
   from webpage x
   where x.url not in (
           select y.url
           from webpage y
           where y.word not in (select z.word
                                from dictionary z
                                where z.language = "French"))
   ```

6. *(from CSE 444, Autumn 2002, Midterm)*

   Consider the following relational schema:

```
Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)
```

   (a) For all patrons of the bar 'Blue Moon', find the total number of beers that they like and are served by 'Blue Moon'.

Solution:

```
SELECT f.drinker, count(distinct l.beer)
FROM   Frequents f, Likes l, Serves s
WHERE  f.bar = 'Blue Moon' and
       f.drinker = l.drinker and l.beer = s.beer and s.bar = f.bar
GROUP BY f.drinker
```

   (b) Find all drinkers that frequent some bar that serves some beer that they don't like.

Solution:

```
SELECT DISTINCT f.drinker
FROM   Frequents f, Serves s
WHERE  f.bar = s.bar and (f.drinker, s.beer) not in Likes
```

   (c) Retrieve all drinkers that frequent only bars that serve only beers that they don't like.

Solution:

```
SELECT x.drinker
FROM   Likes x
WHERE  not exists (SELECT f.bar
                   FROM Frequents f, Likes l, Serves s
                   WHERE f.drinker = x.drinker and
                         f.drinker = l.drinker and
                         f.bar = s.bar and
                         s.beer = l.beer)
```

7. *(from CSE 444, Autumn 2003, Midterm)*

Consider the following relational schema:

```
Student(sid, sname, dept)
Takes(sid, cid, grade)
Course(cid, cname, dept)
```

Students have a name and a department to which they belong. Courses have a name, and a department that offers them. Write SQL queries to answer the following questions:

(a) For each student, find the average grade.

Solution:

```
select student.sname, avg(takes.grade)
from student, takes
where student.sid = takes.sid
group by student.sid, student.sname
```

(b) Find the names of all students that have only grades > 3.0. In other words print a student's name if all its grades are > 3.0.

Solution:

```
select student.sname
from student
where 3.0 < all (select takes.grade
                 from takes
                 where student.sid = takes.sid)
```

(c) Find all students that take only courses offered by the department to which they belong. In other words, print a student's name if all the courses he takes are offered by the department he belongs to.

Solution:

```
select sname
from student
where sid not in (select takes.sid
                  from takes, course
                  where takes.cid = course.cid
                    and course.dept <> student.dept)
```

8. *(from CSE 444, Autumn 2004, Midterm)*

A phone company has a database with the following schema:

```
Customer(cid, name, city, phone)

Bill(bid, cid, year, month, totalAmount)
     -- cid is a foreign key to Customer, not null
     -- totalAmount is updated once a month, during the billing cycle
        totalAmount is either NULL or is >= 0

DetailedCall(did, bid, date, time, phone, minutes, amount)
     -- bid is a foreign key to Bill, not null
     -- amount is updated once a month, during the billing cycle
```

(a) Write a sequence of SQL statements implementing the current billing cycle. Your statements should compute the 'October' 2004 bills of all customers. All necessary records have already been inserted; you only need to update the `amount` and `totalAmount` fields. More precisely, your statements should do the following:

- update the `amount` for each `DetailedCall`, based on the number of minutes of that call. The company charges all customers $0.07 per minute.
- compute the `totalAmount` for each `Bill` as follows: add up all the amounts of all detailed calls on that bill, apply a 7% tax, then add a $10 monthly fee.

Your answer should consist of two SQL `UPDATE` statements.

Solution:

```
update DetailedCall
  set amount = minutes * 0.07
  where bid in (select bid from Bill
                where year=2004 and month='October')

update Bill
  set totalAmount = 10 + 1.07 * (select sum(amount)
                                 from DetailedCall
                                 where DetailedCall.bid = Bill.bid)
  where year=2004 and month='October'
```

(b) Consider the following pairs of SQL queries `Q1` and `Q2`. For each pair, indicate whether the two queries are equivalent or not. Your answer only needs to indicate equivalent or not equivalent, without any justification.

i.

```
Q1: select distinct Customer.name
    from Customer x, Bill y
    where x.cid = y.cid and y.totalAmount = 150

Q2: select distinct Customer.name
    from Customer x, Bill y, Bill z
    where x.cid = y.cid and y.totalAmount = 150
      and x.cid = z.cid and z.totalAmount < 300
```

Solution: Equivalent.

ii.

```
Q1: select distinct Customer.name
    from Customer x, Bill y
    where x.cid = y.cid and y.totalAmount > 100
      and y.year=2004 and y.month='October'

Q2: select distinct Customer.name
    from Customer x, Bill y, Bill z
    where x.cid = y.cid and y.totalAmount > 100 and y.year=2004
      and x.cid = z.cid and z.month='October'
```

Solution: Not equivalent.

iii.

```
Q1: select distinct Customer.cid, Customer.name,
           (select sum(Bill.totalAmount)
            from Bill
            where Customer.cid = Bill.cid) as total
    from Customer

Q2: select Customer.cid, Customer.name,
           sum(Bill.totalAmount) as total
    from Customer, Bill
    where Customer.cid = Bill.cid
    group by Customer.cid, Customer.name
```

Solution: Not equivalent: `Q1` includes customers without bills, returning `total` $= 0$.

iv.

```
Q1: select Customer.cid, Customer.name
    from Customer, Bill
    where Customer.cid = Bill.cid
    group by Customer.cid, Customer.name
    having sum(Bill.totalAmount) > 0

Q2: select distinct Customer.cid, Customer.name
    from Customer, Bill
    where Customer.cid = Bill.cid
      and Bill.totalAmount > 0
```

Solution: Equivalent.

9. *(from CSE 444, Autumn 2005, Midterm)*

An e-commerce company takes orders daily, and bills its customers once per month. There is one bill per customer per month, including all purchases made by that customer that month. The company stores the data in the following schema:

```
Customer(cid, name, email)

Bill(cid, year, month, totalAmount)
     -- cid is a foreign key to Customer, not null
     -- totalAmount is updated once a month and is either NULL or is >= 0

Purchase(pid, cid, year, month, product, unitPrice, quantity, amount)
     -- cid is a foreign key to Customer, not null
     -- amount is updated once a month and is either NULL or is >= 0
```

(a) Write a sequence of SQL statements generating all orders for 'October' 2005 bills for all customers. You need to first update the `amount` field (= `unitPrice` × `quantity`) in all `Purchase` records during that month, then create a new `Bill` record for each customer that made any purchases that month. The `totalAmount` field in each `Bill` record must be the sum of the `amount` of all `Purchase` records plus 8% tax.

Your answer should consist of two SQL statements.

Solution:

```
update Purchase
   set amount = unitPrice * quantity;

insert into Bill
  (select x.cid, y.year, y.month, sum(amount)
   from Customer x, Purchase y
   where x.cid = y.cid and y.year = 2005 and y.month = 'October');
```

10. *(from CSE 444, Winter 2006, Final)*

An IT department stores information about users, systems, and files, in the following schema:

```
User(uid, name, group)
System(sid, hostname)
Access(uid, sid)
Directory(did, dirname)
File(fid, host_sid, owner_uid, directory_did)
```

Each `Directory` "is a" `File`.

(a) Describe the SQL schema for this data. You have to turn in five `CREATE TABLE` statements that contain the primary keys and the foreign keys. You may assume all attributes are `TEXT`.

Solution:

```
create table User(uid text primary key, name text, group text);
create table system(sid text primary key, hostname text);
create table access(uid text references User, sid text references system);
create table Directory(did text primary key, dirname text);
create table File(fid text primary key, host_sid text references System,
   owner_uid text references User, diretory_did text references Directory);
```

(b) Write a SQL query that returns for each system id the total number of files that are owned by users that do not have access to that system.

Solution:

```
select y.sid, y.hostname, count(*)
from User x, System y
where not exists (select * from Access z where x.uid=a.uid and y.sid = a.sid)
group by y.sid, y.hostname;
```

(c) Write a SQL query that removes from `User` all users that do not have access to any system that hosts a file that they own (i.e. they are "locked out" from all their files).

Solution:

```
delete from User
where not exists
   (select *
    from System y, Access z, File w
    where uid = z.uid and y.sid = w.host_sid and w.ownder_uid = uid)
```

11. *(from CSE 444, Winter 2006, Midterm)*

A human resource department maintains information about employees in the following table:

```
Employee(eid, name, dept, manager, gender)
     -- manager is a foreign key to Employee, may be null
```

(a) A *co-worker* of an employee is another employee who has the same manager. Write a SQL query that computes for each employee in the 'Sales' department the number of his/her coworkers (in all departments). Your query should order the answer in decreasing order of the number of coworkers.

Solution:

```
select x.eid, count(*)
from Employee x left outer join Employee y
  on x.manager = y.manager
where x.dept = 'Sales'
group by x.eid
```

(b) Write a SQL query that returns all managers that manage only employees that are in a different department from their own. For example, if 'Joe' is in the 'Sales' department and manages three employees in 'IT', 'HR', and 'HR' departments, then you should return 'Joe'; but if 'Joe' also managed a fourth employee who is in the 'Sales' department then you should not return 'Joe'.

Solution:

```
select x.eid
from Employee x
where x.eid not in (select y.manager
                    from Employee y
                    where x.dept = y.dept)
```

(c) Consider the following sets of SQL queries Q1, Q2, Q3. For each set, indicate which queries are equivalent and which not. You have to turn in an answer of the form Q1 = Q2 ≠ Q3, Q1 = Q2 = Q3, or, say, Q1 ≠ Q2 ≠ Q3 ≠ Q1.

i.

```
Q1: select dept
    from Employee
    where gender = 'F'
    group by dept
    having count(*) < 5
```

```
Q2: select distinct x.dept
    from Employee x
    where (select count(*)
           from Employee y
           where x.dept = y.dept
             and y.gender='F') < 5
```

```
Q3: select distinct x.dept
    from Employee x
    where x.gender = 'F' AND
          (select count(*)
           from Employee y
           where x.dept = y.dept and y.gender='F') < 5
```

Solution: Q1 = Q3 ≠ Q2

ii.

```
Q1: select distinct x.dept
    from Employee x, Employee y, Employee z
    where x.manager = y.eid AND z.manager = y.eid AND
          x.gender = 'F' AND y.gender = 'F' AND z.dept = 'Sales'
```

```
Q2: select distinct x.dept
    from Employee x, Employee y, Employee z
    where x.manager = y.eid AND z.manager = y.eid AND
          z.gender = 'F' AND y.gender = 'F' AND z.dept = 'Sales'
```

```
Q3: select distinct x.dept
    from Employee x, Employee y, Employee z, Employee u
    where x.manager = y.eid AND z.manager = y.eid AND
          u.gender = z.gender AND x.gender = 'F' AND
          y.gender = 'F' AND z.dept = 'Sales' AND u.dept = 'Sales'
```

Solution: Q1 = Q3 ≠ Q2

(d) The human-resource department decides to store separately for each employee the number of employees they manage. The new table has the schema `Supervise(eid, name, num)`, where `num` represents the number of employees supervised by the employee in that entry; note that `num` may be 0. Write a SQL statement that populates the table `Supervise` using the data in `Employee`.

Solution:
```
insert into Supervise(eid, name, num)
select x.eid, x.name, count(*) as num
from Employee x left outer join Employee y on y.manager = x.eid
group by x.eid, x.name
```

12. *(from CSE 444, Autumn 2006, Final)*

Consider the following relational schema:

`Employee(`<u>`eid`</u>`, name, dept, manager, gender)`

The attribute `eid` is a key, and `manager` is a foreign key to `Employee` (may be null); `gender` is `F` or `M`.

(a) Write a SQL query that returns all departments having only female employees (i.e. `gender` $=' F'$).

Solution:
```
select distinct e1.dept
from Employee e1
where e1.dept not in (select e2.dept
                      from Employee e2
                      where e2.gender = 'M')
```

(b) If an employee is managed by a manager in a different department, then we say that she/he is a *consultant*. Write a SQL query that computes for each manager the total number of consultants that they manage.

Solution:
```
select e1.eid, count(e2.eid) as CST_Num
from Employee e1 left outer join Employee e2
  on e2.manager = e1.eid and e2.dept != e1.dept
group by e1.eid
```

(c) The table `Employee` is altered to record for every employee his/her manager's man-

ager. The new table is:

```
Employee(eid, name, dept, manager, gender, managerManager)
```

Write a SQL statement that updates the new field `managerManager` in all records in the table.

Solution:

```
-- first let's see how the Employee table was created:
drop table if exists employee;
create table Employee(eid int primary key,
                      name text,
                      dept text,
                      manager int references employee,
                      gender text);
insert into employee values
  (1,'alice',10,null,'f'),
  (2,'bob',20,1,'m'),
  (3,'carol',30,1,'f'),
  (4,'david',40,2,'m');

-- now alter the table
alter table employee add column managermanager int references employee;
update employee u set managermanager =
    (select x.manager from employee x where x.eid=u.manager);

-- check
select * from employee;
```

13. *(from CSE 444, Autumn 2006, Midterm)*

A manufacturing company maintains a database of their technical manuals. Their schema is described as follows:

```
MANUAL(mid, mTitle, year)                /* key = mid */
CHAPTER(cid, mid, number, cTitle, text)  /* key = (cid,mid) */
CITES(mid, cid, position, midRef)        /* key = everything */
WORD(wid, word)                          /* key = wid */
OCCURS(wid, cid, mid, position)          /* key = everything */
```

Chapters have a foreign key to the manuals where they belong, and a chapter number (typically 1, 2, 3, ... for the chapters in a manual). The text in a chapter may have a reference to another manual, and this is recorded in `CITES`. For example a record in `CITES`

```
('m832', 'c432', 120, 'm7702')
```

means that the chapter with key `('m832','c432')` has a reference to the manual `'m7702'`, and this reference occurs at position 120 of the text. Finally `WORD` represents the collection of all the words in all the manuals, and `OCCURS` is a many-to-many relationship recording every occurrence of a word in a chapter.

(a) Typically each reference is to a manual written earlier. Write a query that returns all manuals that contain some reference to a manual written later (i.e. in a later year). Your query should return `mid`, `mTitle` pairs.

Solution:

```
select m1.mid, m1.mTitle
from MANUAL m1, MANUAL m2, CITES c
where c.mid = m1.mid and
      c.midRef = m2.mid and
      m1.year < m2.year
```

(b) Write a query that computes for each word the total number of occurrences in all manuals. That is, if the word `'bearing'` occurs in manual 1, chapter 1, positions 10 and 20, then in chapter 2, positions 5, 10 and 30, and then again in manual 2, chapter 1, position 20, then you return `('bearing', 6)`. If the word `'database'` doesn't occur anywhere then you return `('database', 0)`.

Solution:

```
select WORD.wid, count(OCCURS.wid)
from WORD left outer join OCCURS
  on WORD.wid = OCCURS.wid
group by WORD.wid
```

(c) Write a query that computes for every word in how many manuals it occurs. For the example above, you would return `('bearing', 2)` and `('database', 0)`.

Solution:

```
select WORD.wid, count(distinct OCCURS.mid)
from WORD left outer join OCCURS
  on WORD.wid = OCCURS.wid
group by WORD.wid
```

(d) For each pair of queries below say whether they are equivalent, i.e. return exactly the same answers. You have to answer 'yes' or 'no'.

i.

```
Q1: select distinct c1.mid
    from   CHAPTER c1, CHAPTER c2, CITES r1, CITES r2
    where  c1.cid = r1.cid and c1.mid = r1.mid
       and c2.cid = r2.cid and c2.mid = r2.mid
       and r1.position < 200 and r2.position < 300

Q2: select distinct c1.mid
    from   CHAPTER c1, CITES r1, CITES r2
    where  c1.cid = r1.cid and c1.mid = r1.mid
       and r1.position < 200 and r2.position < 300
```

Solution: Yes

ii.

```
Q1: select distinct m.mid, m.mTitle
    from   MANUAL m, CHAPTER c1, CITES r, MANUAL m2
    where  m.mid = c1.mid
       and c1.number > 5
       and c1.mid = r.mid and c1.cid = r.cid
       and r.midRef = m2.mid
       and m2.year = m.year

Q2: select distinct m.mid, m.mTitle
    from   MANUAL m, CHAPTER c1, CHAPTER c2, CITES r, MANUAL m2
    where  m.mid = c1.mid and m.mid = c2.mid
       and c1.number > 5 and c2.number > 3
       and c1.mid = r.mid and c1.cid = r.cid
       and r.midRef = m2.mid
       and m2.year = m.year
```

Solution: Yes

iii.

```
Q1: select c.cid, c.mid
    from   CHAPTER c
    where  c.cid not in (select r.cid
                         from CITES r
                         where r.mid = c.mid)

Q2: select c.cid, c.mid
    from   CHAPTER c, CITES r
    group by c.cid, c.mid
    having count(*) > 0
```

Solution: No

14. *(from CSE 444, Spring 2007, Midterm)*

Consider a database of social groups that allows people to become members of groups: a person can be a member of several groups and each group maintains a list of pictures that are accessible to all members. In addition to the groups, the database also maintains a list of friends. The schema is:

```
MEMBER(personName, groupName)
PICTURE(groupName, picture)   /* picture = primary key */
FRIEND(personName1, personName2)
```

`PICTURE` stores for each picture the name of the group that owns that picture. The

`FRIEND` table is symmetric, i.e. if X is friend with Y then Y is friend with X. Every person is a member of at least one group.

(a) Write a SQL query that computes for every person the total number of pictures they can access through their group memberships. That is, a person X can access a picture Y if X is a member of some group Z and Z owns the picture Y. You need to turn in a SQL query that returns a result like this:

```
'Fred', 12
'Joe', 7
'Sue', 0
'Rick', 9
 ...
```

Solution:

```
select x.personName, count(*)
from member x, picture y
where x.groupName = y.groupName
group by x.personName;
```

(b) A "cool person" is one that has at least 40 friends. Write a SQL query that returns all the cool persons in the database. You need to turn in a SQL query that computes a list of names.

Solution:

```
select x.personName
from member x, friend z
where x.personName = z.personName1
group by x.personName;
```

(c) The marketing department has decided to recommend people to subscribe to additional groups. However, they do not want to issue phony recommendations. They would like to recommend to a person X to subscribe to a group Y if all X's friends are members of the group Y, but X is not a member of Y. Write a SQL query that computes for each person X the set of groups to recommend that that person subscribes to. You need to turn in a SQL query that returns a list of (person, group) pairs.

Solution:

```
select x.personName, g.groupName
from member x, member g
where not exists
    (  -- check that x is not already in group g
     select *
     from member x1
     where x1.personName=x.personName
       and x1.groupName = g.groupName)
  and not exists
    ( -- check that none of his friends is a non-member of g
     select *
     from friend z
```

```
    where x.personName=z.personName
      and not exists
         (select * from member w
          where w.personName = z.personName
            and w.groupName = z.groupName))
```

(d) Create a new table `ACCESS(personName, picture)` that lists for each person the list of pictures that they can access. A person X can access a picture Y either if X belongs to a group that owns Y, or if X has a friend Z who belongs to a group that owns Y. Write SQL statements that insert the corresponding tuples in `ACCESS`. You need to turn in one or more `INSERT` queries.

Solution:

```
insert into Access
  (select x.personName, y.picture
   from member x, picture y
   where x.groupName = y.groupName or
         exists (select *
                 from friend z, member w
                 where x.personName = z.personName1
                  and z.personName2 = w.personName
                  and w.groupName = y.groupName));
```

15. *(from CSE 444, Autumn 2008, Final)*

The following schema contains chess players, and their games:

```
Player(PID, name, city)
Game(PID1, PID2)
```

An entry in `Game` means that `PID1` has won over `PID2`. There are no draws. There can be multiple games, i.e. both `PID1` won over `PID2` and vice versa.

(a) 'Joe Champion' is a famous player. (His name is unique in `Player`.) For each city count the number of players in that city that have won against 'Joe Champion' at least once.

Solution:

```
select x.city, count(*)
from player x, game y, player z
where x.pid = y.pid1 and y.pid2 = z.pid and z.name = 'JC'
```

Subqueries are not needed here.

(b) Find all cities where every player has won over 'Joe Champion'.

Solution:

```
select x.city
from player x
where x.city not in (select y.city
                     from player y
                     where y.pid not in
                          (select g.pid1
                           from game g, player z
                           where g.pid2=z.pid and z.pid = 'JC'))
```

Other variants are possible, e.g. `not exists` instead of `not in`. They all require three occurrences of the `Player` table.

(c) Find all cities where no player ever lost against 'Joe Champion'.

Solution:

```
select x.city
from player x
where x.city not in (select y.city
                     from player y, game g, player z
                     where y.pid=g.pid2 and g.pid1=z.pid and z.name='JC')
```

(d) Denote `Q1` and `Q2` the two queries in the previous two questions. Indicate which of the following are true:

$$Q1 \subseteq Q2$$
$$Q1 = Q2$$
$$Q1 \supseteq Q2$$

Solution: None of them.

16. *(from CSE 444, Spring 2010, Final)*

Consider the following Flickr-type database:

```
Users(uid, name)
Picture(pid, owner, size)
Comment(cid, auth, pict, text)
```

Where:

- `User.uid`, `Picture.pid`, `Comment.cid` are keys.
- `Picture.owner`, `Comment.auth` are foreign keys into `Users`.
- `Comment.pict` is a foreign key to `Picture`.

(a) Write a SQL query that counts, for every user, the number of other users who have commented on their pictures. That is, for each user you need to compute the total

number of distinct users (including herself) who have commented on any of her pictures. Your answer should return only those answers where the count is ≥ 1 (i.e. you don't need to return users that have no comments on any of their pictures).

Solution:

```
select x.owner, count(distinct y.auth)
from Picutre x, Comment y
where x.pid = y.pict and x.owner != y.auth
```

```
Users(uid, name)
Picture(pid, owner, size)
Comment(cid, auth, pict, text)
```

(b) A *spammer* is a user who comments on all pictures that are not owned by him (her). Write a SQL query that returns all spammers.

Solution:

```
select *
from Users x
where not exists
       select *
       from Picture y
       where y.owner != x.uid and
        not exists select *
                   from Comment z
                   where x.uid = z.auth and z.pic = y.pid
```

```
Users(uid, name)
Picture(pid, owner, size)
Comment(cid, auth, pict, text)
```

(c) Two users are *friends* if they comment on each others' pictures: that is, x, y are friends if x made a commented on some picture of y, and y made a comment on some picture of x. Write a SQL query that computes, for each user, the number of his/her friends. Your query should return only those answers where the count is ≥ 1.

Solution:

```
select x.uid, x.name, count(*)
from Users x, Users y
where exists (select *
              from Picture u, Comment v
              where u.owner = x.uid and v.auth=y.uid
               and v.pict = u.pid)
     and exists ... symmetrically
group by x.uid, x.name
```

17. *(from CSE 444, Spring 2010, Makeup Midterm; CSE 444, Spring 2010, Midterm)*

Consider the following social network database:

```
Person(pid, name)
Relationship(pid1, pid2, type)
```

Where:

- `Person.pid` is a key.
- `Relationship.pid1` and `Relationship.pid2` are foreign keys to `Person`.
- `Relationship.type` is either 'friend' or 'enemy'.

Keep in mind that `Relationship` is not symmetric: if `p1` is a friend of `p2`, that does not mean `p2` is a friend of `p1`. It is not transitive either: if `p1` is a fried of `p2` who is a friend of `p3`, it doesn't mean `p1` is a fried of `p3`.

(a) A *second degree friend* is the friend of a friend[1]. Write a SQL query that computes for each person the total number of their second degree friends. Your query should return answers of the form: `pid, name, count`. Cryptic hint: "not every person has friends, but you have to count *everyone*'s second degree friends".

Solution:

```
select x.pid, x.name, count(distinct z.pid2)
from Person x left outer join Relationship y
              left outer join Relationship z
     on x.pid = y.pid1 and y.pid2 = z.pid1
    and y.type = 'friend' and z.type = 'friend'
group by x.pid, x.name
```


```
Person(pid, name)
Relationship(pid1, pid2, type)
```


(b) Write a SQL query that returns all persons who have at least 12 common friends with "Mary". Your query should return answers of the form: `pid1, pid2, name`, where `pid1` is Mary's `pid` and `pid2` is that of a person who has 12 or more common friends (meaning there are at least 12 persons `p` such that `pid1` and `p` are friends, and `pid2` and `p` are friends). If there are multiple people called Mary, then you will report each of them.

Solution:

```
select x.pid, y.pid, y.name
from Person x, Person y, Relationship u, Relationship v
where x.pid = u.pid1 and y.pid = v.pid1 and u.pid2 = v.pid2
  and u.type = 'friend' and v.type = 'friend
  and x.name = 'Mary'
group by x.pid, y.pid, y.name
having count(*) >= 12
```

[1] A second degree friend can also be a first degree friend.

(c) Fred says: "my enemies' enemies are my friends". Prove that Fred is wrong: write a query that returns all Fred's enemies' enemies that are not his friends. Your query should return answers of the form: `pid1, pid2`, where `pid1` is Fred's `pid` and `pid2` represents an enemy's enemy that is not Fred's friend.

Solution:

```
select x.pid, z.pid2
from Person x, Relationship y, Relationship z
where x.name = 'Fred'
  and x.pid = y.pid1 and y.pid2 = z.pid1
  and y.type = 'enemy' and z.type = 'enemy'
  and not exists (select * from Relationship u
                  where u.type = 'friend'
                    and u.pid1 = y.pid1 and u.pid2 = z.pid2)
```

```
Person(pid, name)
Relationship(pid1, pid2, type)
```

(d) Your social network database has increased to more than 1M people. You are now running a business, and have two data-intensive applications accessing the database:

- Customer support: your customers call up, and ask questions about their friends and their friends' friends. (They don't ask about enemies).
- Mail advertising: once per week you send out mail to people who are part of groups of 5 or more mutual friends.

To assist in the development of these applications, and perhaps to speed them up, you decide to define two views:

```
CREATE VIEW  FriendsOfFriends(pid1, pid3)
             /* used by customer support */

CREATE VIEW  GroupsOfFive(pid1, pid2, pid3, pid4, pid5)
            /* used by mail advertising */
```

The applications will use these two views instead of, or in addition to the base tables of your database. For each view you have one or two choices: declare it a *virtual view* or a *materialized view*.

Indicate for each of the two views if you would declare it virtual, or materialized. Think about the requirements of the application when making your decision. You do not have to justify your answer.

Should `FriendsOfFriends` be materialized or virtual ?

Solution: virtual

Solution:

```
For customer support accuracy is most important.
```

Should `GroupsOfFive` be materialized or virtual ?

Solution: materialized

Solution:
```
For this large computation, speed is valued more than accuracy.
```

18. *(from CSE 544p, Autumn 2010, Final)*

Consider a photo sharing Website, where users can post pictures, as well as comment and rate other user's pictures. The schema is:

```
User(uid, name)
Comment(uid, pid, score, text)
Picture(pid, author, img)
```

The database has the following constraints:

- `Comment.uid` is a foreign key to `User`
- `Comment.pid` is a foreign key to `Picture`
- `Picture.author` is a foreign key to `User`
- `Comment.score` is an integer number between 1 and 10
- All attributes are `NOT NULL`

(a) A *fan* of a user X is another user (different from X) who has posted at least one comment with a score ≥ 8 for a picture authored by X. Write a SQL query that computes, for every user, the number of his/her fans. Your query should return only those users who have at least one fan, and it should return a list of `uid, name, count` triples.

Solution:
```
select u.uid, u.name, count(*)
from User u, User x, Comment y, Picture z
where u.uid = z.author and x.uid = y.uid and y.pid = z.pid
  and y.score >= 8
  and u.uid != x.uid
group by u.uid, u.name
```

(b) A *spoiler* is a user who gave only scores ≤ 3, except to their own pictures. (Users who did not post any comments are also considered spoilers.) An *accomplice* is a user who gave a score of 10 to some picture posted by a spoiler, other than himself. Write a SQL query that finds all accomplices; your query should return a list of `uid, name` pairs.

Solution:

```
select distinct x.uid, x.name
from User x, Comment y, Picture z
where x.uid = y.uid and y.pid = z.pid and x.uid != z.author
  and y.score = 10
  and z.author not in
        select u.uid
        from User u, Comment v, Picture w
        where u.uid = v.uid and v.pid = w.pid
          and v.score > 3 and v.uid != w.author
```

There must be a negated subquery (NOT IN, or NOT EXISTS, or EXCEPT); alternatively, there can be a HAVING clause to achieve that.

Another solution is with a HAVING query (I don't like that solution.)

Outer joins do not do the trick, they are still a monotone operation.

A major SQL error is to write "from blah x ... where y not in x"

(c) A hacker found a way to break into the system and ran the following commands:

```
insert into Comment(uid, pid, score, text)
  select x.uid, y.pid, 0 as score, 'worst picture i ever saw' as text
  from User x, Picture y
  where x.uid = y.author;

update Picture
set author = select x.uid
             from Comment x
             where x.pid = Picture.pid
             order by x.score desc
             limit 1
```

Luckily, no user activities were performed on the database during or after the breakin. As usual, the hacker was quickly caught, arrested, and sent to jail, but he refused to cooperate in repairing the database. Your job is to repair the database. Write SQL commands (one or more) that will restore the database to its original state.

Solution:

```
update Picture
set author =
  select min(x.uid)  -- we know x.uid is unique,
                     -- but SQL doesn't,
                     -- hence the use of min
  from Comment x
  where x.score = 0 and x.pid = Picture.pid;

delete Comment
where score = 0;
```

(d) We say that a query Q1 is "contained" in a query Q2, if, for every database instance that satisfies the constraints written above, every answer tuple returned by Q1 is also returned by Q2. For the pair of queries below, indicate whether one is contained

in the other. If the query containment holds, then indicate the homomorphism between the two queries.

1. The queries Q1 and Q2 are:

```
Q1:
  select distinct x.name
  from User x, Comment y1, Picture z1,
               Comment y2, Picture z2
  where x.uid = y1.uid and y1.pid = z1.pid
    and y1.score < 5
    and x.uid = y2.uid and y2.pid = z2.pid
    and y2.score > 9

Q2:
  select distinct x.name
  from user x, comment y1, picture z1,
               comment y2, picture z2
  where x.uid = y1.uid and y1.pid = z1.pid
    and y1.score < 6
    and x.uid = y2.uid and y2.pid = z2.pid
    and y2.score = 10
```

   (a) Check one of the following (multiple choices): (Q1 contained in Q2); (Q2 contained in Q1); (none)

   (b) Check the homomorphism mappings (matrix choice): $\{x, y1, y2, z1, z2\} \rightarrow \{x, y1, y2, z1, z2\}$

Solution: Q1 is contained in Q2. The homomorphism from Q2 to Q1 is:

$$\begin{aligned} x &\mapsto x \\ y_i &\mapsto y_i && i = 1, 2 \\ z_i &\mapsto z_i && i = 1, 2 \end{aligned}$$

2. The queries Q3 and Q4 are:

```
Q3:
  select distinct x.name, z.img
  from User x, Comment y1, Comment y2, Picture z
  where x.uid = y1.uid and y1.pid = z.pid
    and x.uid = y2.uid and y2.pid = z.pid
    and y1.score < 5 and y2.score > 9

Q4:
  select distinct x1.name, z3.img
  from User x1, Comment y1, Picture z1,
                Comment y2,
       User x3, Comment y3, Picture z3
  where x1.uid = y1.uid and y1.pid = z1.pid
```

```
and x3.uid = y2.uid and y2.pid = z1.pid
and x3.uid = y3.uid and y3.pid = z3.pid
and y1.score < 5 and y3.score > 9
```

(a) Check one of the following (multiple choices): (Q3 contained in Q4); (Q4 contained in Q3); (none)

(b) Check the homomorphism mappings (matrix choice):
$\{x, x1, x3, y1, y2, y3, z, z1, z3\} \rightarrow \{x, x1, x3, y1, y2, y3, z, z1, z3\}$

Solution: Q3 is contained in Q4. The homomorphism from Q4 to Q3 is:

$x_1 \mapsto x$
$x_3 \mapsto x$
$y_1 \mapsto y_1$
$y_2 \mapsto y_1$ also correct: $y_2 \mapsto y_2$
$y_3 \mapsto y_2$
$z_1 \mapsto z$
$z_2 \mapsto z$

(e) Consider the following virtual view:

```
create view Spammer
  select x.uid, x.name, count(z.pid) as spamCount
  from User x
         left outer join Comment y
         on x.uid = y.uid and y.score > 9
         left outer join Picture z on y.pid = z.pid
  group by x.uid, x.name
```

Consider the query below (it computes "suspiciously ranked pictures")

```
select distinct w.pid
from  Spammer u, Comment v, Picture w
where u.uid = v.uid and v.pid = w.pid
  and u.spamCount >= 2 and v.score > 5
```

Rewrite the query by inlining the view, and flatten the resulting query. You should turn in a single, non-nested SQL query over the base tables user, comment, picture, which is equivalent to the query above.

Solution:

```
select distinct w.pid
from User x, Comment y1, Picture z1, Comment y2, Picture z2,
     Comment v, Picture w
where x.uid = y1.uid and y1.pid = z1.pid and y1.score > 9
  and x.uid = y2.uid and y2.pid = z2.pid and y2.score > 9
  and x.uid = v.uid and v.pid = w.pid and v.score > 5
```

There is no correct solution with GROUP BY and HAVING.

19. *(from CSE 344, Spring 2011, Final)*

Consider a social network database, about people and their relationships. The database has two relations:

```
Person(pid, name)
Relationship(pid1, rel, pid2)
```

Here `Person.pid` is a key, and `Relationship.pid1` and `Relationship.pid2` are foreign keys; `rel` is a string representing the relation type, and can be `friend` or `enemy`. Note that the relationship is not necessarily symmetric: if `Alice` is friend with `Bob`, this does not imply that `Bob` is friend with `Alice`.

(a) Write the SQL statements that define the relational schema for this database. Assume that `pid`'s are integers, and `name` and `rel` are character strings.

Answer (write SQL statements):

Solution:

```
create table Person(
  pid int primary key,
  name varchar(20));
create table Relationship(
  pid1 int references Person,
  rel varchar(20),
  pid2 int references Person);
```

(b) Write a **SQL query** that computes, for each person, the total number of their friends. Your query should return results containing the `pid`, the `name`, and the `count`. Note that your query must return exactly one answer for every person in `Person`.

Answer (write a SQL query):

Solution:

```
select x.pid, x.name, count(*)
from Person x left outer join Relationship y
  on x.pid = y.pid1 and y.rel='friend'
group by x.pid, x.name;
```

20. *(from CSE 344, Spring 2011, Final)*

Your application needs to enforce the following constraint:

All my enemies' enemies are my friends

However, the database administrator has noticed that this constraint does not hold at all ! There are many people whose enemies' enemies are not listed as their friends. Help the database administrator enforce the constraint. You will do this in the following two ways. In each case, write a SQL query that fixes the database according to a certain rule:

(a) The rule is

My enemies' enemies **shall** *be my friends*

Answer (write a SQL INSERT query):

Solution:

```
insert into Relationship
  select x.pid1 as pid1, 'friend' as rel, y.pid2 as pid2
  from Relationship x, Relationship y
  where x.rel = 'enemy' and x.pid2=y.pid1 and y.rel='enemy'
   and not exists (select * from Relationship z
                   where x.pid1=z.pid1 and z.rel='friend'
                     and y.pid2=z.pid2);
```

(b) The rule is

I shall make peace with all my enemies whose enemies are not my friends

Answer (write a SQL DELETE query):

Solution:

```
delete from Relationship
where rel='enemy' and
   exists (select *
           from Relationship y
           where pid2=y.pid1 and y.rel='enemy'
            and not exists(select * from Relationship z
                           where pid1=z.pid1 and z.pid2=y.pid2
                             and z.rel = 'friend'));
```

21. *(from CSE 344, Spring 2011, Midterm)*

A database with the following schema stores a collection of Webpages and the words they contain, and a collection of dictionaries in several languages and the words in those languages:

```
Occurs(url, word)
Dictionary(language, word)
```

- `url` represents a Webpage.
- Every Webpage may contain several words, and every `word` may occur in several Webpages.
- Every `language` may contain several words, and every `word` may occur in several languages
- There are no nulls in the database.

(a) Write a **SQL query** that retrieves all languages that occur in more than 1000 Webpages. A language "occurs" in a Webpage if the Webpage contains a word in that language.

Answer (write a SQL query):

Solution:

```
select y.language
from Occurs x, Dictionary y
where x.word = y.word
group by y.language
having count(*) > 1000
```

```
Occurs(url, word)
Dictionary(language, word)
```

(b) Write a **SQL query** that computes, for each Webpage, the largest number of words on that page in any language.

For example, if a page has 100 words in French and 50 words in English (these two sets of words may be overlapping), then your query will return 100 for that page. If a Webpage has only words that do not occur in any language at all, then you do not need to return that Webpage.

Answer (write a SQL query):

Solution:

```
select url, max(c)
from (select x.url, y.language, count(*) as c
      from Occur x, Dictionary y
      where x.word = y.word
      group by x.url, y.language)
group by url
```

```
Occurs(url, word)
Dictionary(language, word)
```

(c) We say that a Webpage is "monolingual in X" if all words occurring on that Webpage are in the language X (and may be in other languages too). Write a query in the **Relational Calculus** that returns all monolingual Webpages together with the language(s) in which they are monolingual.

For example, if the Webpage is:

```
Introduction to Data Management
```

then your query should return English for that Webpage, because all four words are in English, hence the Webpage is monolingual in English. (The word Data occurs in other languages as well, but not he other three words.)

On the other hand, if the Webpage is:

```
NO SQL !
```

then it is monolingual in English, in French, in Italian, etc, because both NO and SQL are words in English, in French, in Italian. (But the Webpage is not monolingual in Dutch because NO is not a Dutch word; Dutch people say 'NEE').

Answer (write a Relational Calculus query):

Solution: Add $O(x, -)$ and $D(-, z)$: to make the query domain independent:

$$Q(x, z) = O(x, -) \wedge D(z, -) \wedge (\forall y.O(x, y) \Rightarrow D(z, y))$$

```
Occurs(url, word)
Dictionary(language, word)
```

(d) Now write the previous query **in SQL**. (Hint: you may want to first write it first in non-recursive datalog with negation, then in SQL.)

Answer (write a SQL query):

Solution: Starting from:

$$Q(x, z) = O(x, -) \wedge D(z, -) \wedge (\forall y.O(x, y) \Rightarrow D(z, y))$$

Convert it first to non-recursive datalog with negation:

$$T(x, z) = O(x, y), \neg D(z, y)$$
$$Q(x, z) = O(x, -), D(z, -), \neg T(x, z)$$

Note that T is not domain independent: we could have written it as $T(x, z) = O(x, y), D(z, -), \neg D(z, y)$, but since we are negating T anyway in the next rule, it's OK to keep it not domain independent. (Make sure you understand why.) Then SQL is:

```
select o1.url, d1.language
from Occurs o1, Dictionary d1
where not exists
   (select *
    from Occurs o2
    where o1.url = o2.url                      /* x */
      and not exists
            (select *
             from Dictionary d2
             where o2.word = d2.word           /* y */
               and d2.language = d1.language)) /* z */
```

22. *(from CSE 344, Winter 2012, Final)*

The following database contains information about email messages in a large organization:

`Person(pid,name)`
`Email(eid, pidFrom, tid, body, length)`
`EmailTo(eid,pidTo)`

Where:

- `Person.pid` and `Email.eid` are keys.
- `Email.pidFrom` and `EmailTo.pidTo` are foreign keys to `Person`.
- `EmaiTo.eid` is a foreign key to `Email`.
- Every email may be sent to several recipients, stored in the `EmailTo` table.
- `tid` represents the thread to which the email belongs.

(a) Write the SQL statements that define the relational schema for this database. Assume that `pid`'s and `eid` s are integers, and `name` and `body` are character strings (choose an appropriate length).

Answer (write a SQL statement):

Solution:

```
create table Person(
  pid int primary key,
  name varchar(20));
create table Email(
  eid int primary key,
  pidFrom int references Person,
  tid int,
  body varchar(1024),
  length int);
create table EmailTo(
  eid int references Email,
  pidTo int references Person);
```

Notes:

- 1 point off for each missing key or foreign key.

(b) For each email thread, compute the total number of distinct recipients of emails in that thread. For example, if the thread contains 100 emails, and all emails went to only one recipient, then your query should answer 1 for that thread id.

Answer (write a SQL statement):

Solution:

```
select e.tid, count(distinct e2.pidTo)
from Email e, EmailTo e2
where e.eid = e2.eid
group by e.tid
```

Some comments:

- Left outer join is OK (even preferable).
- Mistake: `count(*)` instead `count(distinct ...)`.
- Avoid nested query with `select distinct`
- Avoid joining unnecessarily with `Person`.

(c) A *circle* is a set of three users a, b, c such that a sent an email to b, b sent an email to c, c sent an email to a, and all these three emails are in the same thread. Write a SQL query that computes the total number of circles in the database.

Answer (write a SQL query):

Solution:

```
select count(*)
from email e1, emailTo t1, email e2, emailTo t2, email e3, emailTo t3
where e1.eid = t1.eid and t1.pidTo = e2.pidFrom
  and e2.eid = t2.eid and t2.pidTo = e3.pidFrom
  and e3.eid = t3.eid and t3.pidTo = e1.pidFrom
  and e1.tid = e2.tid and e2.tid = e3.tid;
```

Some comments:

- Avoid joining unnecessarily with `Person`.
- Mistake: missing join on `tid`, and similarly for `eid`.
- Mistake: nested queries (most of which don't work).

(d) A *spammer* is a person who has sent at least one email in every thread. Write a SQL query to find all spammers. Your query should return the spammer's pid and their name.

Answer (write a SQL query):

Solution: I first write it in the Relational Calculus, then remove universal quantifiers:

$$\begin{aligned} Q(pid, n) =& \texttt{Person}(pid, n) \\ & \wedge (\forall eid_1, pidf_1, tid, b_1, l_1.\texttt{Email}(eid_1, pidf_1, tid, b_1, l_1) \Rightarrow \exists eid_2, b_2, l_2.\texttt{Email}(eid_2, pid, tid, b_2, l_2)) \\ =& \texttt{Person}(pid, n) \\ & \wedge \not(\exists eid_1, pidf_1, tid, b_1, l_1.\texttt{Email}(eid_1, pidf_1, tid, b_1, l_1) \wedge \not\exists eid_2, b_2, l_2.\texttt{Email}(eid_2, pid, tid, b_2, l_2)) \end{aligned}$$

Next, datalog:

```
T(pid, tid) :- Email(eid2,pid,tid,b2,l2)
K(pid)      :- Person(pid,-), Email(eid1,pidf1,tid,b1,l1), not T(pid,tid)
Q(pid)      :- Person(pid,n), not K(pid)
```

The predicate `Person(pid,-)` in the definition of `K` is introduced in order to make the rule safe. We now translate to SQL, and may remove that predicate:

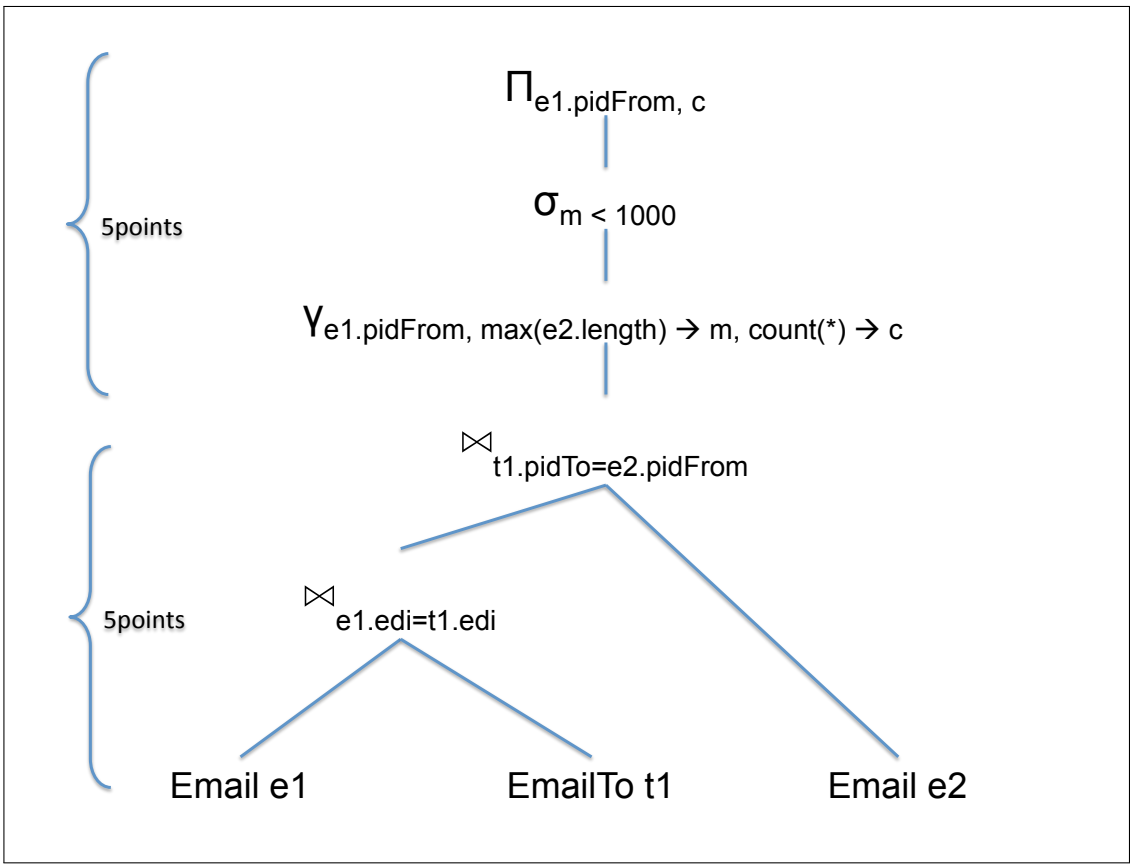
```
select p.pid
from Person p
where not exists (select *
                  from Email e1
                  where not exists (select *
                                    from Email e2
                                    where e2.pidFrom = p.pid and e2.tid=e1.tid));
```

(e) Write a query plan in the extended relational algebra that computes the following query:

```
select e1.pidFrom, count(*)
from Email e1, EmailTo t1, Email e2
where e1.eid = t1.eid and t1.pidTo = e2.pidFrom
group by e1.pidFrom
having max(e2.length) < 1000;
```

Answer (write a query plan; you may draw it as a tree):

Solution:

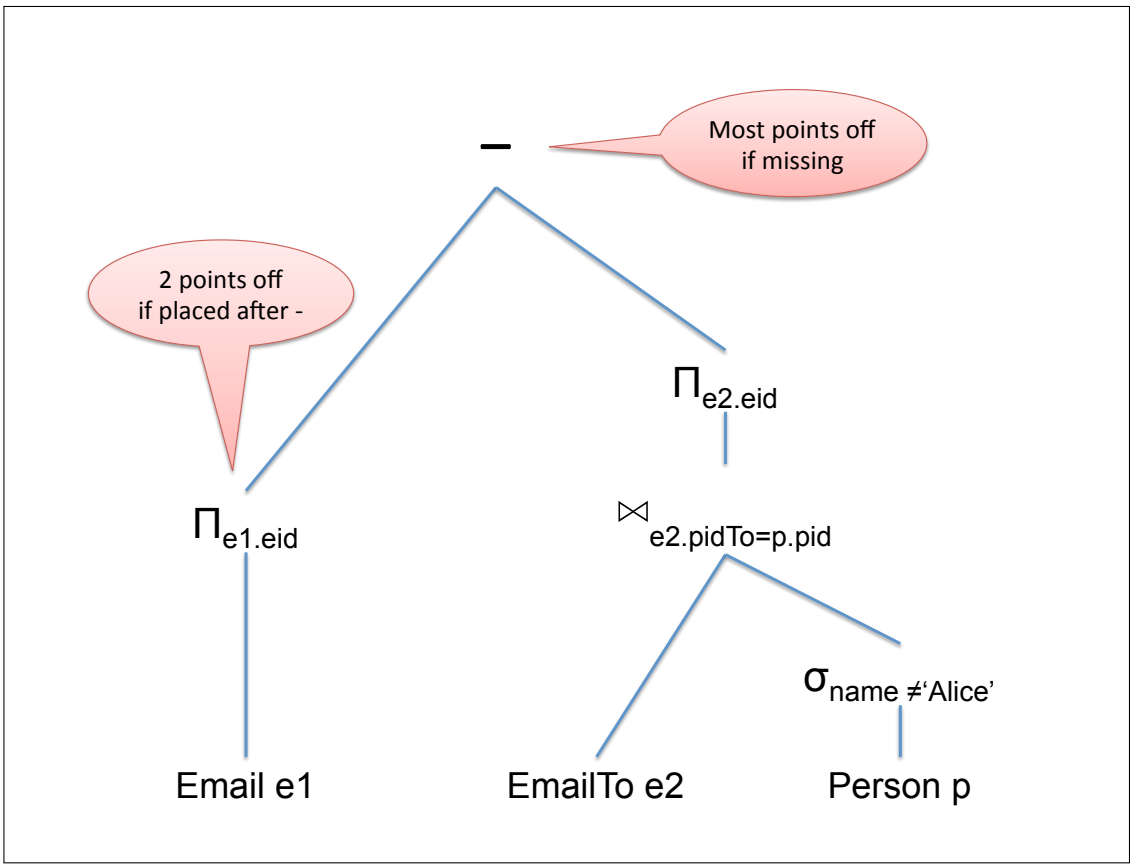


(f) The query below retrieves all emails where every recipient was named Alice. Write a query plan in the relational algebra for this query.

```
select e1.eid
from Email e1
where not exists (select *
                  from EmailTo e2, Person p
                  where e1.eid = e2.eid
                    and e2.pidTo = p.pid
                    and p.name != 'Alice');
```

Answer (write a query plan; you may draw it as a tree):

Solution:



23. *(from CSE 344, Winter 2012, Midterm)*

Consider a photo sharing Website, where users can post pictures, as well as comment and rate other user's pictures. The schema is:

```
Users(uid, name)
Comment(uid, pid, score, txt)
Picture(pid, author, img)
```

Solution: Test data for the solutions.

```
-- if needed:
drop table if exists comment;
drop table if exits picture;
drop table if exists users;

create table users(uid int primary key, name varchar(20));
create table picture(pid int primary key, author int references users, img varchar(10));
create table comment(uid int references users, pid int references picture, score int, txt varchar(30));

insert into users values(1,'Alice');
insert into users values(2,'Bob');
insert into users values(3,'Carol');
```

```
insert into picture values(100,1,'img1');
insert into picture values(200,1,'img2');
insert into picture values(300,1,'img3');
insert into picture values(400,2,'img4');

insert into comment values(1,400,8,'abc');
insert into comment values(3,100,8,'def');
insert into comment values(3,200,8,'hhk');
insert into comment values(3,400,8,'lmn');
```

The database has the following constraints:

- `Comment.uid` is a foreign key to `Users`
- `Comment.pid` is a foreign key to `Picture`
- `Picture.author` is a foreign key to `Users`
- `Comment.score` is an integer number between 1 and 10
- All attributes are `NOT NULL`

(a) Write a SQL query that returns all users who have given a score of 8 or higher to 50 pictures or more. For each user, your query should return the user ID and the name.

Solution:

```
select x.uid, x.name
from Users x, Comment y
where x.uid = y.uid and y.score >= 8
group by x.uid, x.name
having count(*) >= 50
```

(b) A picture is considered *highly rated* if it received at least one score of 10, from a user other than its author. A *cautious* user is a user who commented only on highly rated pictures. (A user who did not comment at all is also cautious.) Write a SQL query that finds all cautious users. Your query should return a list of `uid, name` pairs.

Solution:

```
select x.uid, x.name
from users x
where x.uid not in
   (select y.uid  -- these are the non-cautious users
    from comment y  -- check that y.pid is not highly rated
    where y.pid not in
       (select u.pid  -- the pictures that are highly rated
        from comment u, picture v
        where u.pid = v.pid and u.score = 10 and u.pid != v.author))
```

Two negations are necessary! 5 points off for a single negation. Almost 10 points off for no negation at all.

A negation can be avoided by using an aggregate, e.g. `having`(max(`score`) < 10).

(c) A hacker found a way to break into the system and ran the following commands:

```
insert into Comment(uid, pid, score, txt)
  select x.uid, y.pid, 0 as score, 'worst picture i ever saw' as txt
  from Users x, Picture y
  where x.uid = y.author;

update Picture
set author = (select x.uid
              from Comment x
              where x.pid = Picture.pid
              order by x.score desc
              limit 1);
```

That is, he inserted some fake comments, and messed up all the picture authors. Luckily, his two queries were logged by the system (that' why we know them) and no other user activities were performed on the database during or after the breakin. As usual, the hacker was quickly caught, arrested, and sent to jail, but he refused to cooperate in repairing the database. Your job is to repair the database. Assume that the database was in a consistent state right before the hacker's attack, and write SQL commands (one or more) that will restore the database to its original state. Your answer should consists of a sequence of INSERT/UPDATE/DELETE statements, and should not rely on any log or backup of the database. For example, you might turn in something like this (which are correct SQL statement, but are not the real answer):

```
delete from Comment
where exists (select * from Users x where x.uid = uid and x.name='Hacker Joe')

update Picture
set author = (select x.uid from Comment x
              where pid = x.pid and x.score = 10 limit 1)
```

Solution:

```
update Picture
set author =
 (select x.uid
  from Comment x
  where x.score = 0 and x.pid = Picture.pid
  limit 1);

delete from Comment
where score = 0;
```

Each update has 5 points.

Updated picture to some arbitrary uid (i.e. forgot to check `score` = 0) 4 points off.

`delete from comment where exists (...comment c ...c.score`= 0 3 points off.

Forgot `limit` ? No problem.
1-2 points taken off for what I thought was complex or too inefficient.

(d) For each statement below, indicate whether it is true of false.

i. An index may help a select-from-where SQL query run faster, or may not affect its running time, but it can never make a query run slower. True or false?

Solution: true

ii. An index may help an update (insert, delete, or update) SQL query run faster, or may not affect its running time, but it can never make a query run slower. True or false?

Solution: false

iii. Consider a selection operation $\sigma_{\texttt{price}>90 \wedge \texttt{price}<100}(\texttt{Product})$. Using an unclustered index on `price` will make the query at least as fast as scanning the entire table `Product`. True or false?

Solution: false

iv. Consider a selection operation $\sigma_{\texttt{price}>90 \wedge \texttt{price}<100}(\texttt{Product})$. Using a clustered index on `price` will make the query at least as fast as scanning the entire table `Product`. True or false?

Solution: true

v. A large table `Product(pid, name, price)` is queried intensively and is never updated. Then we should create three clustered indexes, on `Product(pid)`, `Product(name)`, and `Product(price)`. True or false?

Solution: false

24. *(from CSE 344, Winter 2013, Final)*

An online store uses a database with the following schema:

```
Product(pid, pname, price)
Customer(cid, cname, address)
Orders(oid, cid, date, total)        cid foreign key to Customer
LineItem(lid, oid, pid, quantity)    oid, pid foreign keys to Orders, Product
```

`Products` and `Customer` contains a set of products and of customers respectively. `Orders` represent orders placed by customers: each order has several lines, called LineItems, each line representing one product that the customer ordered. Each `Orders` record also stores the total value of the order, which is the sum of `price*quantity` for all line items in that order.

The following constraints must be enforced by the database system:

- All keys and foreign keys.
- All attributes are not null.
- `price` > 0

In addition, the business logic is such that the `total` in each order equals the sum of `price*quantity` for all line items in that order. This constraint is maintained by the application logic.

(a) Write the `CREATE TABLE` statements. Include all constraints that must be enforced by the database system. All identifiers (`pid`, `cid`, `oid`, `lid`) are integers; `quantity` is an integer; `price`, `total` are float; all other fields are `text`.

Answer (write SQL CREATE TABLE statements):

Solution: All queries were tested on postgres.

```
create table product(pid int primary key,
                     pname text not null,
                     price float not null
                         check (price > 0.0));
create table customer(cid int primary key,
                      cname text not null,
                      address text not null);
create table orders(oid int primary key,
                    cid int not null references customer,
                    date text not null,
                    total float not null);
create table lineitem(lid int primary key,
                      oid int not null references orders,
                      pid int not null references product,
                      quantity int);
```

To test answers for the remaining questions, run this:

```
insert into product values(1, 'gadget', 100.00);
insert into product values(2, 'gizmo', 1000.00);

insert into customer values(1, 'alice', 'nome');
```

```
insert into orders values(1, 1, 'today', 5300.00);
insert into lineitem values(1, 1, 1, 3);
insert into lineitem values(2, 1, 2, 5);

insert into orders values(2, 1, 'yesterday', 3500);
insert into lineitem values(3, 2, 1, 5);
insert into lineitem values(4, 2, 2, 3);
```

(b) In every order, the `total` must be equal to the sum of the line item quantity times the product price. If the total is incorrect, then that order must be reviewed by a human and the error must be corrected. Write a query that finds all inconsistent orders; your query should return a set of `oid`.

Answer (write a SQL query):

Solution:

```
select x.oid
from orders x, lineitem y, product z
where x.oid = y.oid and y.pid = z.pid
group by x.oid, x.total
having x.total != sum(y.quantity * z.price);
```

(c) The company's *revenue* is the sum of all `total`s, from all orders. In this question you will write a *what-if* query: without updating the database, your query will compute how much the revenue would change, had the company changed some prices:

Write a query that computes how much the company's revenue would increase, if they raised the price of the 'gizmo' product by 10% (`pname = 'gizmo'`).

Answer Write a SQL query that returns one number (the revenue increase):

Solution:

```
select sum(y.quantity * 0.1 * z.price)
from lineitem y, product z
where y.pid = z.pid and z.pname='gizmo';
```

(d) For the queries below we will consider a simplified schema:

```
Customer(cid, name)
Orders(oid, cid)
LineItem(lid, oid, quantity)
```

For each of the three Relational Calculus expressions below, give an equivalent datalog query. In addition, give a relational algebra plan for the first expression only.

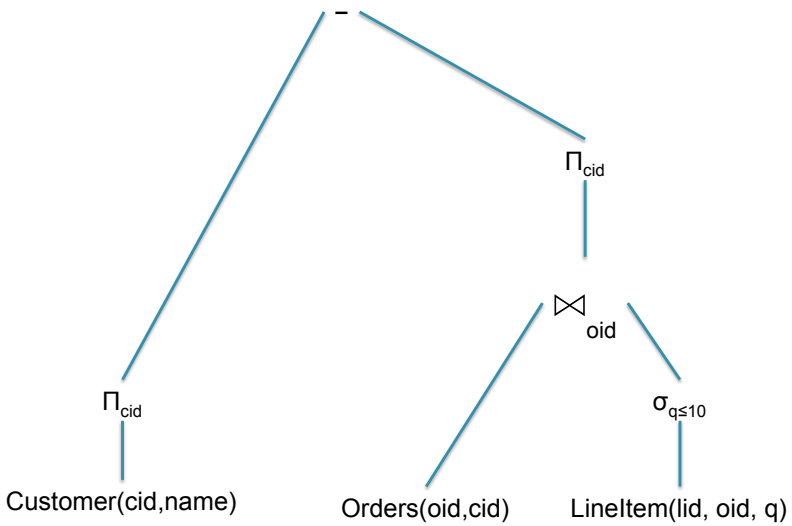
i. Find customers for which all their orders and all line items on those orders are for products with price > 10:

$q_1(x) = \exists y[\texttt{Customer}(x, y) \wedge \forall u \forall v \forall q(\texttt{Orders}(u, x) \wedge \texttt{LineItem}(v, u, q) \Rightarrow q > 10)]$

Answer write a datalog query, AND draw a query plan:

Solution:

```
v1(x) = Orders(u,x), LineItem(v,u,q), q <= 10
q1(x) = Customer(x,y), not v1(x)
```



ii. Find customers that placed some order where all line items are for products with price > 10:

$q_2(x) = \exists y \exists u[\texttt{Customer}(x, y) \wedge \texttt{Orders}(u, x) \wedge \forall v \forall q(\texttt{LineItem}(v, u, q) \Rightarrow q > 10)]$

Answer write a datalog query (do NOT draw the query plan):

Solution:

```
v2(u) = LineItem(v,u,q), q <= 10
q2(x) = Customer(x,y), Orders(u,x), not v2(u)
```

iii. Find customers for which all their orders have some line item for a product with price > 10:

$q_3(x) = \exists y[\texttt{Customer}(x, y) \wedge \forall u(\texttt{Orders}(u, x) \Rightarrow \exists v \exists q(\texttt{LineItem}(v, u, q) \wedge q > 10))]$

Answer write a datalog query (do NOT draw the query plan):

Solution:

```
v32(u) = LineItem(v,u,q), q > 10
v31(x) = Orders(u,x), not v32(u)
q3(x) = Customer(x,y), not v31(x)
```

25. *(from CSE 344, Winter 2013, Midterm)*

A search engine company maintains a database with the following schema:

```
QueryLog(uid,qid,word)
User(uid, name, language)
Dictionary(word, language)
```

`QueryLog` is the archive of all keyword queries issued by the users; `User` is the set of registered users; `Dictionary` stores some common words in several languages. Note that one word may occur in more than one language.

To illustrate, we consider the following search queries:

```
Joe:       valentine day
Jaque:     day trip
Joe:       sidereal day
Jim:       day
```

and the following small database:

`QueryLog:`

uid	qid	word
u1	q1	valentine
u1	q1	day
u2	q2	day
u2	q2	trip
u1	q3	sidereal
u1	q3	day
u3	q4	day

`User:`

uid	name	language
u1	Joe	English
u2	Jaque	French
u3	Jim	English

`Dictionary:`

word	language
valentine	Latin
valentine	English
day	English
trip	English
sidereal	Latin
cosmos	Greek

Solution: To test all queries in this question, first run the following in sqlite:

```
.header on
.mode columns
create table QueryLog(uid int, qid int, word text);
create table User(uid int, name text, language text);
create table Dictionary(word text, language text);

insert into QueryLog values(1,1,'valentine');
insert into QueryLog values(1,1,'day');
```

```
insert into QueryLog values(2,2,'day');
insert into QueryLog values(2,2,'trip');
insert into QueryLog values(1,3,'sidereal');
insert into QueryLog values(1,3,'day');
insert into QueryLog values(3,4,'day');

insert into User values(1,'Joe','English');
insert into User values(2,'Jaque','French');
insert into User values(3,'Jim','English');

insert into Dictionary values('valentine','Latin');
insert into Dictionary values('valentine','English');
insert into Dictionary values('day','English');
insert into Dictionary values('trip','English');
insert into Dictionary values('trip','English');
insert into Dictionary values('sidereal','Latin');
insert into Dictionary values('cosmos','Greek');
```

(a) Write a SQL query that returns the top 10 most common queried words, in decreasing order of their frequency. For our small example, the query would return:

day	4
sidereal	1
trip	1
valentine	1

Recall that you can limit the number of tuples returned using `limit`, for example:

```
select *
from T
limit 20
```

returns only 20 answers.

Solution:

```
select word, count(*)
from QueryLog
group by word
order by count(*) desc
limit 10;
```

(b) Write a SQL query that returns all languages that were never used in a query. In our small example, your query would return `Greek`, because no user asked for any word in `Greek`.

Solution:

```
select distinct x.language
from Dictionary x
where not exists
     (select *
      from QueryLog y, Dictionary z
      where y.word = z. word and z.language = x.language);
```

(c) You want to identify users who speak a foreign language. We assume that a user speaks a language L if she issues at least one query Q where all words are in L; if L

is different from the user's native language (stored in `Users`) then we assume that she speaks the foreign language L. Write a query that returns all users that speak a foreign language. Your query should return the `uid`, the user name, and the foreign language.

For example, in our toy database your query should return `Jaque`, because he issued the query `day trip` where all words are in `English`; you should not return `Joe`: for example in his query `sidereal day`, while the first word is in Latin, the second is only in his native `English`.

Solution:

```
select distinct x.uid, x.name, z.language
from User x, QueryLog y, Dictionary z
where x.uid = y.uid and y.word = z.word and x.language != z.language
  and not exists                  --  ALL words in Q must be in language L
       (select *
        from QueryLog y1          -- is there a word NOT in L?
        where y.uid = y1.uid and y.qid = y1.qid
                                  -- same user, same query
          and not exists
                    (select *
                     from Dictionary z1
                     where z1.word = y1.word
                       and z1.language = z.language));
```

One way to design such a query is to first write it in Relational Calculus:

$$q(uid, n, l) = \exists q \exists w \exists lu[\texttt{QueryLog}(uid, q, w) \wedge \texttt{User}(uid, n, lu) \wedge \texttt{Dictionary}(w, l) \\ \wedge l \neq lu \\ \wedge \forall w_1(\texttt{QueryLog}(uid, q, w_1) \Rightarrow \texttt{Dictionary}(w_1, l))]$$

The first two lines check if there is a query q that mentions a word w in some language l (line 1) other than the user's language (line 2). Line 3 checks that all other words w_1 in that query are in the language l.

Next compute the negation of line 3 first:

$$\texttt{queryHasSomeWordNotInL}(uid, q, l) = \texttt{QueryLog}(uid, q, w_1), \neg\texttt{Dictionary}(w_1, l)$$

This is the fist NOT EXISTS query in SQL. Notice that it still has a negation, hence the second NOT EXISTS query in SQL.

Finally, use this result to compute the answer that we want:

$$q(uid, n, l) = \texttt{QueryLog}(uid, q, w), \texttt{User}(uid, n, lu), \texttt{Dictionary}(w, l), \\ (l \neq lu), \neg\texttt{queryHasSomeWordNotInL}(uid, q, l)$$

(d) The company monitors some users, and some keywords on a regular basis. That is, the company frequently runs queries of the following kind (where the values "Joe" and "valentine" are replaced with different constants)

```
QueryType1:
  select y.qid, y.word
  from User x, QueryLog y
  where x.uid = y.uid and x.name = 'Joe'
  order by y.qid, y.word;

QueryType2:
  select x.uid, x.name
  from User x, QueryLog y
  where x.uid = y.uid and y.word = 'valentine';
```

To speedup queries like these (where `Joe` and `valentine` are replaced with various constants), the database administrator considers creating the following indexes:

```
create index QL1 on QueryLog(uid)
create index QL2 on QueryLog(word)
create index QL3 on QueryLog(uid,word)

create index U1 on User(uid)
create index U2 on User(name)
create index U3 on User(uid,name)
```

Help the database administrator choose which indexes to create, by filling in the table below. For each entry, indicate one of the following tree options:

OK: this index may speed up queries of this type.

Dominated by X: the index may speed up queries of this type, but the index X would be at least as good, so the DBA should always prefer X.

Bad: the index is never useful for queries of this type, the DBA should not even consider it.

Fill in with one of `OK`, `Dominated by X`, `Bad`:

	QueryType1
	`select y.qid, y.word`
	`from User x, QueryLog y`
	`where x.uid = y.uid`
	`and x.name = 'Joe'`
	`order by y.qid, y.word;`
		QueryType2
		`select x.uid, x.name`
		`from User x, QueryLog y`
		`where x.uid = y.uid`
		`qand y.word = 'valentine';`

QL1 `QueryLog(uid)`		
QL2 `QueryLog(word)`		
QL3 `QueryLog(uid,word)`		
U1 `User(uid)`		
U2 `User(name)`		
U3 `User(uid,name)`		

Solution:

	QueryType1
	`select y.qid, y.word`
	`from User x, QueryLog y`
	`where x.uid = y.uid`
	`and x.name = 'Joe'`
	`order by y.qid, y.word;`
		QueryType2
		`select x.uid, x.name`
		`from User x, QueryLog y`
		`where x.uid = y.uid`
		`qand y.word = 'valentine';`

QL1 `QueryLog(uid)`	OK	Bad
QL2 `QueryLog(word)`	Bad	OK
QL3 `QueryLog(uid,word)`	Dominated by QL1	Bad
U1 `User(uid)`	Bad	OK
U2 `User(name)`	OK	Bad
U3 `User(uid,name)`	Bad	Dominated by U1

(e) Answer the questions below:

i. Which of the following is *not* a database management system:

(a) Oracle

(b) SQL Server

(c) Excel

(d) SQLite

(e) Postgres

Answer (a), (b), (c), (d) or (e).

Solution: (c)

ii. Joe Pythonson never took a course in databases, but he is a proficient python programmer. For his job, he needs to develop an address book application, which stores names, phone numbers, emails, and addresses. Joe plans to develop it entirely in python, without using any database management system. You think that's a terrible idea, and try to convince Joe to use a database system like SQL Server or postgres. How would you answer to Joe's arguments?

(a) Joe: "I can implement the access procedure to the phone book very efficiently! I know splay-trees very well, they are efficient and I can implement them quickly". Your answer:

1. "Splay-trees do not support joins".
2. "You need to make splay trees persistent, and extend them with concurrent access: to implement them you will need much more time than you think".
3. "Splay trees are copyrighted, and you cannot use them in your commercial application"

1, 2, or 3?

Solution: 2

(b) Joe: "If I develop everything in python then the entire address book is in a single process. Thus, my application will run faster than, say, using SQL Server, which is a client server configuration and requires a network connection for every single user request". Your answer:

1. "True, but: the client-server configuration has the advantage of supporting simultaneous access by multiple users. If you program in python, then you have to implement multi users access yourself, and this requires a lot of work".
2. "False: the client-server configuration will run queries much more efficiently than the single process implementation because the database system optimizes the queries before running them."
3. "True, but: the client-server configuration can scale to much larger datasets, because the data is in a different process."

1, 2, or 3?

Solution: 1

(c) Joe: "I have just invented a new and amazing compression method for email addresses (for which I expect to receive the Turing Award). Using

this specialized email compression technique I can make the email lookup function run 10^6 times faster than any database system, crushing my competitors". Your answer:

1. "You don't need to give up on a database management system: get the source code of an open source database management system, like postgres, implement your amazing email compression technique, recompile it, and still use your specialized method to crush the competition".
2. "Users will be turned away from your address book application when they learn that it is not powered by a database management system, even with your ridiculously fast lookup".
3. "This only speeds up access to email addresses, while all other accesses will be have the same speed; users won't notice much difference, while you will put a lot of effort reimplementing much of the functionality offered by a database management system."

1, 2, or 3?

Solution: 3

26. *(from CSE 344, Autumn 2013, Final)*

An online picture sharing company uses a database with the following schema:

```
create table Usr (
    uid int primary key,
    uname text not null,
    city text not null);

create table Picture (
     pid int primary key,
     uid int not null references Usr(uid),
     size int not null,
     pdf text);
```

Every user has a key (`uid`), a name (`uname`) and a `city`.

Every picture has a key (`pid`), an author (`uid`) that is a foreign key to `Usr`, a `size`, and the `pdf` content (which is plain text).

Solution: Run this in postgres:

```
insert into Usr values(1,'Alice','Denver');
insert into Usr values(2,'Bob','Denver');
insert into Usr values(3,'Carol','Portland');
insert into Usr values(4,'David','Denver');
```

```
insert into Usr values(5,'Evan', 'Seattle');

insert into Picture values(10, 1, 1500000, 'abcd');
insert into Picture values(20, 1, 1500000, 'bcde');
insert into Picture values(30, 2, 1500000, 'cdef');
insert into Picture values(40, 1, 1500000, 'defg');
insert into Picture values(50, 2, 1500000, 'efgh');
insert into Picture values(60, 5,  500000, 'fghk');
insert into Picture values(70, 5, 4000000, 'ghkm');
```

(a) Write a SQL query that retrieves all users who have only pictures of less than 1MB (`size < 1000000`). Your query should return the users' `uid` and `uname`

Solution:

```
select x.uid, x.uname
from usr x
where not exists (select *
                  from picture y
                  where y.size >= 1000000 and x.uid = y.uid);
```

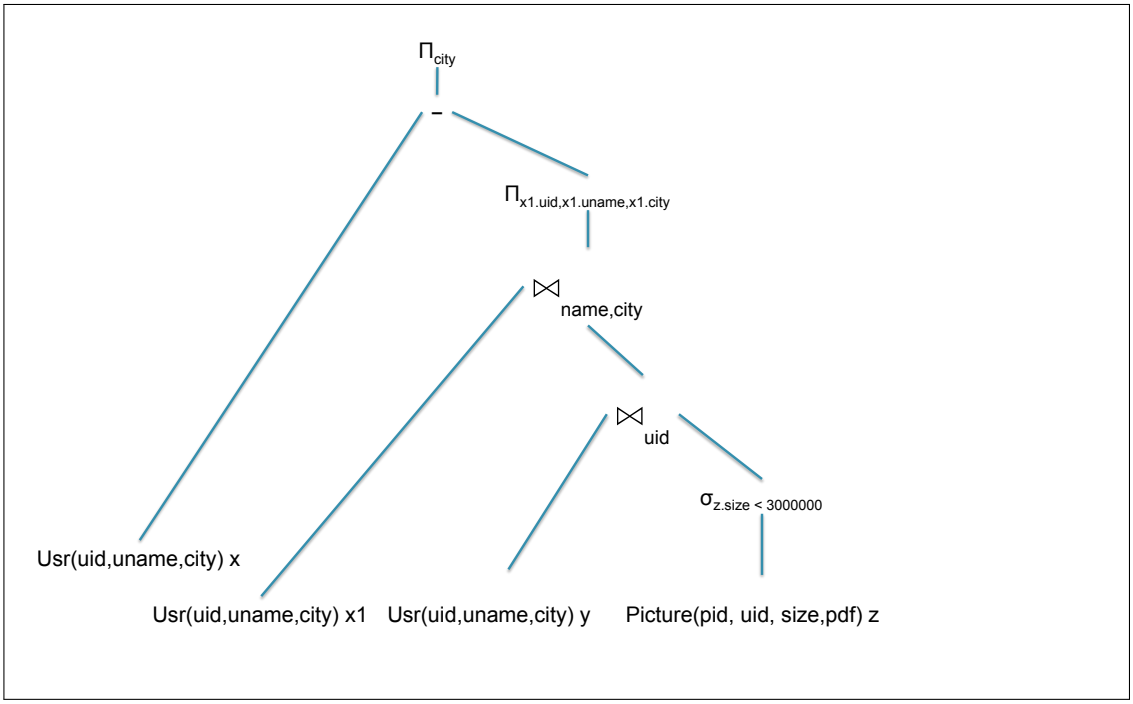
(b) Write a Relational Algebra expression that is equivalent to the following SQL query:

```
select distinct x.city
from usr x
where not exists (select *
                  from usr y, picture z
                  where x.uname = y.uname
                    and x.city = y.city
                    and y.uid = z.uid
                    and z.size < 3000000);
```

Solution: It is useful to write the datalog program first:

```
T(uid, city) = Usr(uid, -, city), Picture(pid, uid, s, -), s < 3000000
Q(city)      = Usr(uid, -, city), not T(uid, city)
```

Note that we cannot compute the difference `Usr`$-$`T` because the type is incorrect. Hence, we first semi-join `T` with `Usr`, i.e. project back on the `Usr` attributes, then take the difference:



(c) Consider the following query:

```
select x.uid, x.uname,
         (select count(*)
          from Picture y
          where x.uid = y.uid and y.size > 1000000)
from Usr x
where x.city = 'Denver';
```

For each query below indicate if it is correct AND equivalent to the given query. You should answer 'yes' only of the query returns exactly the same answers.

i. Is this query equivalent?

```
select x.uid, x.uname, count(*)
from Usr x, Picture y
where x.uid = y.uid  and y.size > 1000000 and x.city = 'Denver'
group by x.uid, x.uname;
```

Answer yes or no

Solution: No

ii. Is this query equivalent?

```
select x.uid, x.uname, count(*)
from Usr x, Picture y
where x.uid = y.uid  and x.city = 'Denver'
group by x.uid, x.uname
having y.size > 1000000;
```

Answer yes or no

Solution: No

iii. Is this query equivalent?

```
select x.uid, x.uname, count(y.pid)
from Usr x left outer join Picture y on x.uid = y.uid and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

Answer yes or no

Solution: Yes

iv. Is this query equivalent?

```
select x.uid, x.uname, count(*)
from Usr x left outer join Picture y on x.uid = y.uid and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

Answer yes or no

Solution: No

(d) Consider the following query:

```
select distinct x.uid, x.uname
from Usr x, Picture u, Picture v, Picture w
where x.uid = u.uid and x.uid = v.uid and x.uid = w.uid
  and u.size > 1000000 and v.size < 3000000 and w.size = u.size;
```

For each query below indicate if it is correct AND equivalent to the given query:

i. Is this query equivalent?

```
select distinct x.uid, x.uname
from Usr x, Picture u, Picture v, Picture w
where x.uid = u.uid and x.uid = v.uid and x.uid = w.uid
  and u.size > 1000000 and v.size < 3000000 and w.size > 1000000;
```

Answer yes or no

Solution: Yes

ii. Is this query equivalent?

```
select distinct x.uid, x.uname
from Usr x, Picture u, Picture v
where x.uid = u.uid and x.uid = v.uid
  and u.size > 1000000 and v.size < 3000000;
```

Answer yes or no

Solution: Yes

iii. Is this query equivalent?

```
select distinct x.uid, x.uname
from Usr x, Picture u, Picture w
where x.uid = u.uid and x.uid = w.uid
  and u.size > 1000000 and u.size < 3000000 and u.size = w.size;
```

Answer yes or no

Solution: No

iv. Is this query equivalent?

```
select distinct x.uid, x.uname
from Usr x, Picture u, Picture v, Picture w
```

```
where x.uid = u.uid and x.uid = v.uid and x.uid = w.uid
  and u.size > 1000000 and v.size < 3000000 and w.size = v.size;
```

Answer yes or no

Solution: Yes

(e) The company merges with a local advertising company in Denver, which has a database of artistic pictures. Each picture has a title, the pdf image, and may have several authors. All authors are from Denver.

```
create table Artwork (
   wid int primary key,
   title text,
   pdf text);

create table Author (
   aid int primary key,
   wid int not null references Artwork(wid),
   aname text not null);
```

i. Some users and pictures occur in both databases. Write a SQL query that finds all common user,picture pairs. Two users are considered to be the same if their names and cities are the same; two pictures are considered to be the same if their pdf's are the same. Your query should return the user name (which is the same in both databases), its two keys in `Usr` and `Author`, and the two keys of the identical picture in `Picture` and `Artwork`.

Solution:

```
insert into Artwork values(101, 'Artwork by Alice and Fred', 'abcd');
insert into Artwork values(102, 'Artwork by Fred',           'xyzu');

insert into Author values(201, 101, 'Alice');
insert into Author values(202, 101, 'Fred');
insert into Author values(203, 102, 'Fred');

select distinct x.uid, v.aid, x.uname, y.pid, v.wid
from Usr x, Picture y,
     Artwork u, Author v
where x.uid = y.uid and u.wid = v.wid and x.city = 'Denver'
  and x.uname = v.aname and y.pdf = u.pdf;
```

ii. The new company creates a new schema for the integrated data:

```
create table NewUsr (
    nuid int primary key,
    nuName text not null,
    city text not null);

create table NewPicture (
     npid int primary key,
     title text,
     size int,
     pdf text);
```

```
create table Authored (
     nuid int not null references newUsr(nuid),
     npid int not null references NewPicture(npid));
```

Write a sequence of SQL queries that insert all the data from the two old databases into the new database. You do not need to eliminate duplicates: that is, you will insert all the records from `Usr`, `Picture` into the new schema, then will insert all the records from `Artwork`, `Author` into the new schema, without worrying about duplicates. All keys in the `Usr`, `Picture` database are distinct from the keys in the `Artwork`, `Author` database.

Solution:

```
insert into NewUsr(nuid, nuName, city)
     select uid as nuid, uname as nuName, city from Usr;

insert into NewPicture(npid, title, size, pdf)
     select pid as npid, null as title, size, pdf from Picture;

insert into Authored(nuid, npid)
     select uid as nuid, pid as npid from Picture;

insert into NewUsr(nuid, nuName, city)
     select aid as nuid, aname as nuName, 'Denver' as city from Author;

insert into NewPicture(npid, title, size, pdf)
     select wid as npid, title, null as size, pdf from Artwork;

insert into Authored(nuid, npid)
     select aid as nuid, wid as npid from Author;
```

2 Database design (27–42)

These problems asks you to design a schema. An application is described in English, and you must produce the tables, declare their keys, and identify the foreign keys that connect them. Most of these problems then continue by asking for queries over the schema you just designed.

Two habits make these questions much easier. First, decide what a single row of each table *means* before writing any attributes; a table whose rows have no crisp meaning is usually the sign of a modelling mistake. Second, treat many-to-many relationships explicitly: they need a table of their own, and forgetting one is the most common error on this material.

Because the design is yours, sometimes several answers are defensible. The solutions give one reasonable design, not the only one; where your schema differs, check that it can still answer every query the problem goes on to ask.

27. *(from CSE 344, Autumn 2013, Midterm)*

A *sparse* matrix is a matrix $A = (a_{ij})$ where many elements a_{ij} are 0. A sparse matrix can be stored in a relation with three attributes: $A(i, j, v)$ where i, j are the row and column and v the value of the element in that row and column. For example the matrix:

$$A = \begin{pmatrix} 30 & 0 & 0 \\ -10 & 0 & 0 \\ 0 & 10 & 50 \end{pmatrix}$$

can be represented as:

i	j	v
1	1	30
2	1	-10
3	2	10
3	3	50

Notice that it's OK to keep 0 values, even though it may be less efficient. For example, the matrix above can also be represented as:

i	j	v
1	1	30
1	2	0
1	3	0
2	1	-10
3	2	10
3	3	50

(a) You are given two sparse matrices A, B with schemas:

```
A(i,j,v)
B(j,k,v)
```

Write an SQL query that computes the product matrix $A \cdot B$. Your query should return a set of triples (i, k, v) representing the product matrix; you do not need to remove the entries with value 0.

Recall that the product of an $m \times n$ matrix $A = (a_{ij})$ with an $n \times p$ matrix $B = (b_{jk})$ is the $m \times p$ matrix $C = (c_{ik})$ where $c_{ik} = \sum_{j=1,n} a_{ij} b_{jk}$. For example:

$$\begin{pmatrix} 30 & 0 & 0 \\ -10 & 0 & 0 \\ 0 & 10 & 50 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 4 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 20 & -10 \end{pmatrix}$$

Turn in a SQL query:

Solution:

```
select A.i,B.k, sum(A.v*B.v)
from A,B
where A.j=B.j
group by A.i,B.k
```

(b) You are given two sparse matrices A, B with schemas:

```
A(i,j,v)
B(i,j,v)
```

Write an SQL query that, given two sparse matrices A, B computes the sum matrix $A+B$. Your query should return a set of triples (i, j, v) representing the sum matrix; you do not need to remove the entries with value 0.

For example:

$$\begin{pmatrix} 30 & 0 & 0 \\ -10 & 0 & 0 \\ 0 & 10 & 50 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 4 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 30 & 0 & 0 \\ -10 & 2 & 4 \\ 0 & 10 & 49 \end{pmatrix}$$

Turn in a SQL query:

Solution:

```
with T as (select i,j from A union select i,j from B)
select T.i, T.j, (case when A.v is null then 0 else A.v end) +
                 (case when B.v is null then 0 else B.v end)
from T left outer join A on (T.i = A.i and T.j = A.j)
       left outer join B on (T.i = B.i and T.j = B.j)
```

An elegant solution, from a student:

```
select x.i, x.j, sum(x.v)
from (select * from A union select * from B) as x
group by x.i, x.j
```

Note that postgres doesn't accept `A union B`, instead we had to say `select * from A union ....`

A full outer join does not work here. This is *incorrect*:

```
select A.i, A.j, (case when A.v is null then 0 else A.v end) +
                 (case when B.v is null then 0 else B.v end)
from A full outer join B on A.i = B.i and A.j = B.j;
```

It doesn't work because, for the entries `(B.i,B.j)` that do not have a matching entry in `A`, the values of `A.i,A.j` are NULL.

28. *(from CSE 544p, Winter 2014, Final)*

Consider a photo sharing Website, where users can post pictures, as well as comment and rate other user's pictures. The schema is:

```
Usr(uid, name)
Comment(uid, pid, score, body)
Picture(pid, author, img)
```

The database has the following constraints:

- `Comment.uid` is a foreign key to `Usr`
- `Comment.pid` is a foreign key to `Picture`
- `Picture.author` is a foreign key to `Usr`
- `Comment.score` is an integer number between 1 and 10
- All attributes are `NOT NULL`

(a) A *fan* of a user X is another user (different from X) who has posted at least one comment with a score ≥ 8 for a picture authored by X. Write a SQL query that computes, for every user, the number of his/her fans. Your query should return only those users who have at least one fan, and it should return a list of `uid, name, count` triples.

Solution: Test data for the solution queries.

```
drop table if exists comment;
drop table if exists picture;
drop table if exists usr;

create table usr(uid int primary key, name text);
create table picture(pid int primary key, author int references usr, img text);
create table comment(uid int references usr, pid int references picture, score int, body text);

insert into usr values(1, 'Alice');
insert into usr values(2, 'Bob');
insert into usr values(3, 'Carol');

insert into picture values(10, 1, 'asdf');
insert into picture values(20, 1, 'asdf');
insert into picture values(30, 1, 'asdf');
insert into picture values(40, 2, 'asdf');
insert into picture values(50, 2, 'asdf');

insert into comment values(1, 40, 9, 'blah');
insert into comment values(3, 10, 2, 'blah');
```

```
select x.uid, x.name, count(distinct y.uid)
from Usr x, Comment y, Picture z
where x.uid = z.author and y.pid = z.pid
  and y.score >= 8
  and x.uid != y.uid
group by x.uid, x.name;
```

Common mistakes:

- Checking $count(*) > 0$ or $count(*) \geq 1$.

- Nested subqueries or other very complex solutions.
- Wrong or missed group by
- Missing `name` in group by
- Missed `distinct` in `count`

(b) A *spoiler* is a user who gave only scores ≤ 3, except to their own pictures. (Users who did not post any comments are also considered spoilers.) An *accomplice* is a user who gave a score of 10 to some picture posted by a spoiler, other than himself. Write a SQL query that finds all accomplices; your query should return a list of `uid, name` pairs.

Solution:

```
select distinct x.uid, x.name
from Usr x, Comment y, Picture z
where x.uid = y.uid and y.pid = z.pid and x.uid != z.author
  and y.score = 10
  and z.author not in
       (select u.uid
        from Usr u, Comment v, Picture w
        where u.uid = v.uid and v.pid = w.pid
          and v.score > 3 and v.uid != w.author);

-- same thing, more readable:
with spoiler as
  (select u.uid
   from usr u
   where u.uid not in
      (select v.uid
       from comment v, picture w
       where v.pid = w.pid and v.score>3 and v.uid!=w.author))
select distinct x.uid, x.name
from usr x, comment y, picture z, spoiler u
where x.uid = y.uid and y.pid = z.pid and x.uid != z.author
 and y.score = 10
 and z.author = u.uid;
```

Common mistakes:

- Monotone query doesn't work. A solution must included some non-monotone construction: `not in`, `not exists`, `except`, `having`. `outer join` is not enough.
- "FROM blah x ... WHERE y NOT IN x" is non-sense SQL.

(c) A hacker found a way to break into the system and ran the following commands:

```
insert into Comment(uid, pid, score, body)
  select x.uid, y.pid, 0 as score, 'worst picture i ever saw' as body
  from Usr x, Picture y
  where x.uid = y.author;
```

```
update Picture
set author = (select x.uid
              from Comment x
              where x.pid = Picture.pid
              order by x.score desc
              limit 1);
```

That is, he inserted some fake comments, and messed up all the picture authors (and his commands even violated some of the database constraints). Luckily, his two queries were logged by the system (that' why we know them) and no other user activities were performed on the database during or after the breakin. As usual, the hacker was quickly caught, arrested, and sent to jail, but he refused to cooperate in repairing the database. Your job is to repair the database. Assume that the database was in a consistent state right before the hacker's attack, and write SQL commands (one or more) that will restore the database to its original state. Your answer should consists of a sequence of INSERT/UPDATE/DELETE statements, and should not rely on any log or backup of the database.

Solution:

```
update Picture
set author =
 (select min(x.uid)  -- we know x.uid is unique,
                     -- but SQL doesn't,
                     -- hence the use of min
                     -- another possibility is to use 'limit 1'
  from Comment x
  where x.score = 0 and x.pid = Picture.pid);

delete from Comment
where score = 0;
```

(d) Consider the following virtual view:

```
create view Spammer as
  select x.uid, x.name, count(z.pid) as spamCount
  from Usr x
         left outer join Comment y
         on x.uid = y.uid and y.score > 9
         left outer join Picture z on y.pid = z.pid
  group by x.uid, x.name;
```

Consider the query below (it computes "suspiciously ranked pictures")

```
select distinct w.pid
from  Spammer u, Comment v, Picture w
where u.uid = v.uid and v.pid = w.pid
  and u.spamCount >= 2 and v.score > 5
```

Rewrite the query by inlining the view, and flatten the resulting query. You should turn in a single, non-nested SQL query over the base tables user, comment, picture, which is equivalent to the query above.

Solution: The query simply checks that `u.uid` posted two comments with a score > 9:

```
select distinct w.pid
from User x, Comment y1, Picture z1, Comment y2, Picture z2,
     Comment v, Picture w
where x.uid = y1.uid and y1.pid = z1.pid and y1.score > 9
  and x.uid = y2.uid and y2.pid = z2.pid and y2.score > 9
  and z1.pid != z2.pid
  and x.uid = v.uid and v.pid = w.pid and v.score > 5
```

There is no correct solution with `having`, because the query needs to group by `Picture.pid` while the `having` clause needs to group by the spammer's `User.uid` and these are irreconcilable.

(e) Below are several queries written in the *Tuple Relational Calculus*. For each of them, indicate which of the following English questions it answers.

1.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \forall p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \Rightarrow \\ & \quad (\forall c \in \textsc{Comment}(c.\textsc{pid} = p.\textsc{pid}) \Rightarrow (c.\textsc{score} \geq 9 \vee c.\textsc{score} \leq 2)) \end{aligned}$$

2.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \forall p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \Rightarrow \\ & \quad (\forall c \in \textsc{Comment}\ (c.\textsc{pid} = p.\textsc{pid}) \wedge (c.\textsc{score} < 9) \\ & \quad \Rightarrow (\forall d \in \textsc{Comment}\ (c.\textsc{uid} = d.\textsc{uid}) \Rightarrow d.\textsc{score} \leq 2)) \end{aligned}$$

3.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \forall p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \Rightarrow \\ & \quad \exists c \in \textsc{Comment} \wedge c.\textsc{pid} = p.\textsc{pid} \wedge c.\textsc{score} \geq 9 \wedge \\ & \quad (\forall d \in \textsc{Comment} \wedge (c.\textsc{uid} = d.\textsc{uid}) \Rightarrow d.\textsc{score} > 2)) \end{aligned}$$

4.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \forall p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \Rightarrow \\ & \quad (\forall c \in \textsc{Comment}(c.\textsc{pid} = p.\textsc{pid} \Rightarrow c.\textsc{score} \leq 2)) \vee \\ & \quad (\forall c \in \textsc{Comment}(c.\textsc{pid} = p.\textsc{pid} \Rightarrow c.\textsc{score} \geq 9)) \end{aligned}$$

5.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \forall p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \Rightarrow \\ & \quad \forall c \in \textsc{Comment} \wedge c.\textsc{pid} = p.\textsc{pid} \wedge c.\textsc{score} < 9 \Rightarrow \\ & \quad (\exists d \in \textsc{Comment} \wedge (c.\textsc{uid} = d.\textsc{uid}) \wedge d.\textsc{score} > 2) \end{aligned}$$

6.

$$\begin{aligned} Q(u) = & u \in \textsc{Usr} \\ & \wedge \exists p \in \textsc{Picture}\ (u.\textsc{uid} = p.\textsc{author}) \wedge \exists v \in \textsc{Usr} \\ & \quad (\forall c \in \textsc{Comment}\ (c.\textsc{pid} = p.\textsc{pid}) \wedge (c.\textsc{uid} = v.\textsc{uid}) \Rightarrow c.\textsc{score} \geq 9) \wedge \\ & \quad (\forall d \in \textsc{Comment} \wedge (v.\textsc{uid} = d.\textsc{uid}) \Rightarrow d.\textsc{score} > 2) \end{aligned}$$

i. Find all users whose pictures received only scores greater than or equal to 9, except from those users who gave only scores less than or equal to 2. 1-2-3-4-5-6?

Solution: 5

ii. Find all users whose pictures received only scores greater than or equal to 9 from users who gave at least one score greater than 2. 1-2-3-4-5-6?

Solution: 2

iii. Find all users that have a picture that received only scores greater than or equal to 9 from some user who gave only scores greater than 2. 1-2-3-4-5-6?

Solution: 6

iv. Find all users whose pictures received only scores that were either greater than or equal to 9 or less than or equal to 2. 1-2-3-4-5-6?

Solution: 1

v. Find all users for which every picture received either only scores greater than or equal to 9, or only scores less than or equal to 2. 1-2-3-4-5-6?

Solution: 4

vi. Find all users for which every picture received some scores greater than or equal to 9 from a user who gave only scores greater than 2. 1-2-3-4-5-6?

Solution: 3

Solution: 6

29. *(from CSE 544p, Autumn 2015, Final)*

A blog Website allows users to post blogs and to add discussion to other user's blogs. The schema is:

```
Blog(bid, author, body)
Discussion(bid, ts, uid, vote, content)
Usr(uid, name)
```

The database has the following constraints:

- `Discussion.bid` is a foreign key to `Blog`
- `Discussion.ts` is an integer timestamp
- `Discussion.uid` is a foreign key to `Usr`
- `Blog.author` is a foreign key to `Usr`
- `Discussion.vote` is number that is either $+1$ or -1 (up or down).
- All attributes are `NOT NULL`

(a) A user is a *follower* of some other user if she commented (i.e. added a discussion) on at least one blog posted by that the other user. Write a SQL query that retrieves all users with at least 50 followers. You should return the `uid`, `name`, and number of followers. Note that, if a user posts n blogs and another user comments on all of them, then the first user still has only one follower.

Solution: Test data for the solutions.

```
drop table if exists discussion;
drop table if exists blog;
drop table if exists usr;

create table usr(uid int primary key, name text);
create table blog(bid int primary key, author int references usr, body text);
create table discussion(uid int references usr, bid int references blog, vote int, body text, ts int);

insert into usr values(1, 'Alice');
insert into usr values(2, 'Bob');
insert into usr values(3, 'Carol');

insert into blog values(10, 1, 'asdf');
insert into blog values(20, 1, 'asdf');
insert into blog values(30, 1, 'asdf');
insert into blog values(40, 2, 'asdf');
insert into blog values(50, 2, 'asdf');

insert into discussion values(1, 40,  1, 'blah', 1);
```

```
insert into discussion values(2, 10, -1, 'blah', 1);
insert into discussion values(3, 20,  1, 'blah', 1);
insert into discussion values(3, 40, -1, 'blah', 2);
select u.uid, u.name, count(distinct y.uid)  -- count(y.uid) OK
from discussion y, blog z, usr u
where y.bid = z.bid and z.author = u.uid
  and y.uid != u.uid                         -- OK if missed
group by u.uid, u.name
having count(distinct y.uid) >= 50;
```

(b) A "disparager" is a user who gave only *down* votes (vote $= -1$), except to her own blogs. Write a SQL query that returns all disparager. Your query should return the uid and names.

Solution:

```
select x.uid, x.name
from usr x
where not exists    -- other non-monotone constructs:
                    -- not in, 1=ALL(...), except, max(vote)=-1,...
   (select *
    from discussion y, blog z
    where x.uid = y.uid
      and y.bid = z.bid
      and z.author != x.uid
      and y.vote = 1);
                    -- unnecessary, but OK to check if x gave at least one -1 vote
```

(c) Some users repost their discussion: in this problem you will write a SQL query to delete all duplicate discussions. A discussion is a duplicate if it refers to the same blog, is written by the same user, and has the same content and vote as a previous discussion. Write a SQL query to remove all duplicate discussions.

Solution:

```
delete from discussion y
   where exists
           (select *
            from discussion x
            where y.bid = x.bid
              and y.uid = x.uid
              and y.vote = x.vote
              and y.body = x.body
              and y.ts > x.ts);
```

(d) A *leader* is a user for which every blog has a discussion from every other user. Write a SQL query to find all leaders.

Solution:

```
select x.uid, x.name
from usr x
where not exists
```

```
(select *
 from blog y, usr z   -- find a blog y,
                      -- and user z
                      -- s.t. z did not discuss y
 where x.uid = y.author
  and x.uid != z.uid
  and not exists
        (select *
         from discussion w
         where w.bid = y.bid and w.uid = z.uid));
```

(e) Below are several queries written in the *Tuple Relational Calculus*. For each of them, indicate which of the following English questions it answers.

1.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \exists b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \wedge (\forall d \in \textsc{Discussion}(d.\textsc{bid} = b.\textsc{bid}) \Rightarrow (d.\textsc{vote} = 1)))$$

2.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \exists b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \wedge (\exists d_1 \in \textsc{Discussion} \exists d_2 \in \textsc{Discussion}(d_1.\textsc{bid} = b.\textsc{bid}) \wedge (d_2.\textsc{bid} = b.\textsc{bid}) \wedge (d_1.\textsc{ts} < d_2.\textsc{ts}) \wedge (d_1.\textsc{vote} > d_2.\textsc{vote})))$$

3.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \exists b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \wedge (\exists d \in \textsc{Discussion}(d.\textsc{bid} = b.\textsc{bid}) \wedge (d.\textsc{vote} = 1)))$$

4.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \forall b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \Rightarrow (\exists d_1 \in \textsc{Discussion} \exists d_2 \in \textsc{Discussion}(d_1.\textsc{bid} = b.\textsc{bid}) \wedge (d_2.\textsc{bid} = b.\textsc{bid}) \Rightarrow (d_1.\textsc{ts} < d_2.\textsc{ts}) \wedge (d_1.\textsc{vote} > d_2.\textsc{vote})))$$

5.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \forall b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \Rightarrow (\forall d \in \textsc{Discussion}(d.\textsc{bid} = b.\textsc{bid}) \Rightarrow (d.\textsc{vote} = 1)))$$

6.

$$Q(u) \equiv (u \in \textsc{Usr} \wedge \exists b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \wedge (\forall d_1 \in \textsc{Discussion} \forall d_2 \in \textsc{Discussion}(d_1.\textsc{bid} = b.\textsc{bid}) \wedge (d_2.\textsc{bid} = b.\textsc{bid}) \Rightarrow (d_1.\textsc{ts} \geq d_2.\textsc{ts}) \vee (d_1.\textsc{vote} \leq d_2.\textsc{vote})))$$

7.

$$\begin{aligned} Q(u) \equiv & (u \in \textsc{Usr} \wedge \forall b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \Rightarrow \\ & (\forall d_1 \in \textsc{Discussion} \forall d_2 \in \textsc{Discussion}(d_1.\textsc{bid} = b.\textsc{bid}) \wedge (d_2.\textsc{bid} = b.\textsc{bid}) \\ & \Rightarrow (d_1.\textsc{ts} \geq d_2.\textsc{ts}) \vee (d_1.\textsc{vote} \leq d_2.\textsc{vote}))) \end{aligned}$$

8.

$$\begin{aligned} Q(u) \equiv & u \in \textsc{Usr} \wedge \forall b \in \textsc{Blog}\ (u.\textsc{uid} = b.\textsc{author}) \Rightarrow \\ & (\exists d \in \textsc{Discussion}(d.\textsc{bid} = b.\textsc{bid}) \wedge (d.\textsc{vote} = 1)) \end{aligned}$$

i. Find all users that received only *up* votes on each of their blogs. Answer with a number 1-8

Solution: 5

ii. Find all users that received some *up* vote on each of their blogs. Answer with a number 1-8

Solution: 8

iii. Find all users that received only *up* votes on some of their blogs. Answer with a number 1-8

Solution: 1

iv. Find all users that received some *up* vote on some of their blogs. Answer with a number 1-8

Solution: 3

v. Find all users that have some blog whose votes never decreased from *up* to *down* over time. Answer with a number 1-8

Solution: 6

vi. Find all users that have only blogs whose votes never decreased from *up* to *down* over time. Answer with a number 1-8

Solution: 7

vii. Find all users that have some blog whose votes had some decrease from *up* to *down* over time. Answer with a number 1-8

Solution: 2

viii. Find all users that have only blogs whose votes had some decrease from *up* to *down* over time. Answer with a number 1-8

Solution: 4

Solution: 6

30. *(from CSE 344, Winter 2016, Final)*

(a) We represent sparse matrices as tables with three attributes: row, column, value. Write a SQL query that computes the product of two sparse matrices called A and B. Recall that the product $C = AB$ of two matrices is defined as:

$$C_{ik} = \sum_j A_{ij} B_{jk}$$

For example, consider the matrix product below:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 2 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad AB = \begin{bmatrix} 0 & 2 & 1 \\ 3 & 0 & 0 \\ 0 & 4 & 3 \end{bmatrix}$$

Then the inputs and output to your query would be:

A

row	col	val
1	1	1
2	2	3
3	1	2
3	3	1

B

row	col	val
1	2	2
1	3	1
2	1	1
3	3	1

Answer

row	col	val
1	2	2
1	3	1
2	1	3
3	2	4
3	3	3

Write a SQL query:

Solution:

```
drop table if exists A;
drop table if exists B;
drop table if exists C;

create table A(row int, col int, val int);
create table B(row int, col int, val int);
create table C(row int, col int, val int);

insert into A values(1, 1, 1);
insert into A values(2, 2, 3);
insert into A values(3, 1, 2);
```

```
insert into A values(3, 3, 1);

insert into B values(1, 2, 2);
insert into B values(1, 3, 1);
insert into B values(2, 1, 1);
insert into B values(3, 3, 1);

insert into C values(1, 1, 1);
insert into C values(2, 2, 1);
insert into C values(3, 3, 1);

  select A.row as row, B.col as col, sum(A.val * B.val) as val
  from A, B
  where A.col = B.row
  group by A.row, B.col;
```

(b) The traces of a matrix A is the sum of the elements on the diagonal: $Tr(A) = \sum_i A_{ii}$. Write a SQL query that computes $Tr(ABC)$, where A, B, C are three sparse matrices.

Write a SQL query:

Solution: Notice that $Tr(ABC) = \sum_{i,j,k} A_{ij}B_{jk}C_{ki}$. The SQL query is:

```
select sum(A.val * B.val * C.val)
from A,B,C
where A.col=B.row and B.col=C.row and C.col=A.row;
```

```
   select sum(A.val * B.val * C.val) from A,B,C where [wrong condition]
```

where the wrong condition was typically $A.row = A.col$ etc, or $A.row = B.row$ etc.

(c) Consider three relations with schemas:

$$R(A, B), S(C, D), T(E, F)$$

All attributes are NOT NULL, and no attribute is a key. For each query below, write an equivalent SQL query that is unnested. For example, if the given query were

```
select distinct x.A from (select y.A as A from R y where y.B = 2) x;
```

then your answer would be:

```
select x.A from R x where x.B = 2;
```

If query cannot be unnested, then indicate so.

i. Unnest this query:

```
select distinct x.A, (select sum(y.B) from R y where x.A = y.A) as S
from R x;
```

Write an unnested SQL query or say *impossible*:

Solution:

```
drop table if exists R;
```

```
drop table if exists S;
drop table if exists T;

create table R(A int, B int);
create table S(C int, D int);
create table T(E int, F int);

insert into R values(1,20);
insert into R values(2,30);
insert into S values(20,300);
insert into T values(300,4000);

select x.A, sum(x.B) as S
from R x
group by x.A;
```

ii. Unnest this query:

```
select x.A, x.B, (select sum(y.D) from S y where x.B = y.C) as S
from R x;
```

Write an unnested SQL query or say *impossible*:

Solution:

```
select x.A, x.B, sum(y.D) as S
from R x left outer join S y on x.B=y.C
group by x.A, x.B;
```

Note that the original query does not return any duplicates, because R does not contain duplicates (is a set), and A, B are all the attributes in R. THis means that it is possible to unnest.

iii. Unnest this query:

```
select distinct x.A as A, y.F as F
from R x, (select u.C as C, v.F as F
           from S u, T v
           where u.d = v.E) y
where x.B = y.C;
```

Write an unnested SQL query or say *impossible*:

Solution:

```
select distinct x.A as A, v.F as F
from R x, S u, T v
where x.B = u.C and u.D = v.E;
```

iv. Unnest this query:

```
select x.A as A, y.F as F
from R x, (select distinct u.C as C, v.F as F
           from S u, T v
           where u.D = v.E) y
where x.B = y.C;
```

Write an unnested SQL query or say *impossible*:

Solution: Not possible to unnest

(d) Consider four relations with the following schemas:

$$R(A, B), R2(A, B), S(C, D), V(G, H, K)$$

For each of the identities in the Relational Algebra below, indicate whether they hold. Assume set semantics for all operators:

i. Does this identity hold?

$$(R - R2) - R2 = R$$

Yes/No:

Solution: No

ii. Does this identity hold?

$$((R - R2) - R2) - R2 = R - R2$$

Yes/No:

Solution: Yes

iii. Does this identity hold?

$$\Pi_A(R \bowtie_{B=C} S) = \Pi_A(R)$$

Yes/No:

Solution: No

iv. Does this identity hold?

$$\Pi_{AB}(R \bowtie_{B=C} S) - R2 = \Pi_{AB}((R - R2) \bowtie_{B=C} S)$$

Yes/No:

Solution: Yes

v. Does this identity hold?

$$\gamma_{G,\text{sum}(K)\to L}(V) = \gamma_{G,\text{sum}(M)\to L}(\gamma_{G,H,\text{sum}(K)\to M}(V))$$

Yes/No:

Solution: Yes

31. *(from CSE 344, Winter 2016, Midterm)*

Consider the following database about picture tagging Website:

```
Member(mid, name, age)
Picture(pid, year)
Tagged(mid, pid)
```

Solution:

```
drop table if exists Tagged;
drop table if exists Member;
drop table if exists Picture;

create table Member(mid int, name text, age int);
create table Picture(pid int, year int);
create table Tagged(mid int, pid int);

insert into Member values(1, 'Alice', 10);
insert into Member values(2, 'Bob', 20);
insert into Member values(3, 'Carol', 30);

insert into Picture values(10, 2011);
insert into Picture values(20, 2012);
insert into Picture values(30, 2013);
insert into Picture values(40, 2014);

insert into Tagged values(1,10);
insert into Tagged values(1,20);
insert into Tagged values(2,20);
insert into Tagged values(3,10);
insert into Tagged values(3,20);
insert into Tagged values(3,30);
```

Relation `Member` contains a tuple for each registered member of the Website; `Picture` contains the metadata of the pictures (its key `pid`, and the year when the picture was taken), and `Tagged` says which user was tagged in what picture. Primary keys are underlined. Attributes values may be null.

(a) Write a SQL query that retrieves the names of all members that were tagged both in the year 2011 and in 2014. Return them sorted in lexicographic order.

Solution:

```
select x.name
from Member x, Tagged y1, Picture z1, Tagged y2, Picture z2
where x.mid = y1.mid and y1.pid = z1.pid and z1.year = 2011
  and x.mid = y2.mid and y2.pid = z2.pid and z2.year = 2014
order by x.name;
```

Two common mistakes:

1. Joining both Picture Tables to the same Tagged Table.

```
select M.name
from Member M, Tagged T, Picture P1, Picture P2
where M.mid = T.mid and P1.pid = T.pid and P2.pid = T.pid and
P1.year = 2011 and P2.year = 2014
order by M.name;
```

because:

```
T.pid = P1.pid and T.pid = P2.pid => P1.pid = P2.pid => P1.year = P2.year
```

Since pid is the primary key of Picture.

2. Selecting Members who were tagged in 2011 or 2014 instead of both.

```
select M.name
from Member M, Tagged T, Picture P,
where M.mid = T.mid and P.pid = T.pid and
(P.year = 2011 or P.year = 2014)
order by M.name;
```

(b) For each query below, indicate if the query correctly returns the `name` of all users who were never tagged in 2015. Anywhere between zero and all queries compute the correct answer. Each query is syntactically correct.

```
Q1 = select distinct x.name
     from Member x, Tagged y
     where x.mid = y.mid
       and not exists
                  (select *
                   from Picture z
                   where y.pid = z.pid
                     and z.year = 2015);

Q2 = select distinct x.name
     from Member x
     where not exists
                 (select *
                  from Tagged y, Picture z
                  where x.mid = y.mid
                    and y.pid = z.pid
                    and z.year = 2015);
```

```
Q3 = select distinct x.name
     from Member x
     where not exists
             (select *
              from Tagged y
              where x.mid = y.mid
               and not exists
                    (select *
                     from Picture z
                     where y.pid = z.pid
                       and z.year = 2015));

Q4 = select distinct x.name
     from Member x, Tagged y, Picture z
     where x.mid = y.mid and y.pid = z.pid and z.year = 2015
     group by x.name
     having count(z.pid) = 0;
```

Answer with any subset of Q1, Q2, Q3, Q4:

Solution: Q2

(c) Write a SQL query that returns for each member the total number of times that member was tagged in 2015. Order the results by the members' names. You should include all members, even those who were not tagged that year.

Solution:

```
select x.name, count(z.pid)
from Member x left outer join Tagged y on x.mid = y.mid
            left outer join Picture z on y.pid = z.pid and z.year = 2015
group by x.mid, x.name
order by x.name;
```

Common mistake:

```
SELECT m.name, COUNT(*)  -- incorrect count
FROM Member m LEFT OUTER JOIN Tagged t ON (m.mid = t.mid), Picture p
         -- incorrect join
WHERE p.pid = t.pid and p.year=2015
group by m.name
order by m.name
```

```
SELECT m.name, COUNT(*)
FROM Member m, Tagged t, Picture p -- need outer join!
WHERE m.mid = t.mid and p.pid = t.pid and p.year=2015
group by m.name
order by m.name
```

Another mistake is to use `count(*)` instead of `count(p.pid)`

(d) Write a SQL query that returns the mid's and names of all members that were tagged only in pictures were Alice was also tagged. Hint: you may want to first answer the corresponding Datalog question (same exam, in the Relational Algebra section).

Solution: Datalog:

```
aliceTagged(pid) :- Member(mid, 'Alice',-),Tagged(mid, pid)
nonAnswer(mid) :- Tagged(mid,pid) not aliceTagged(pid)
answer(mid,name) :- Member(mid,name,-), not nonAnswer(mid)
```

```
select x.mid, x.name
from Member x
where x.mid not in
           (select y.mid
            from Tagged y
            where y.pid not in
               (select v.pid from Member u,Tagged v
                where u.name='Alice' and u.mid=v.mid));
```

Common mistakes:

1. Have a single level of negation.
2. Have a single level of negation combined with `u.name!='Alice'`

(e) Consider a workload consisting of many queries of this kind:

```
select x.name
from Member x, Tagged y, Picture z
where x.mid = y.mid
  and y.pid = z.pid
  and z.year = ?;
```

For each index below, indicate if it can potentially speed up the workload, if it is the only index that exists. You will assume that the `Member` and `Tagged` are very large relations, and that only a very small number of pictures are in any given year.

i. Index on `Picture(year)`. Yes or no?

Solution: yes

ii. Index on `Picture(pid)`. Yes or no?

Solution: no

iii. Index on `Picture(pid,year)`. Yes or no?

Solution: no

iv. Index on `Picture(year,pid)`. Yes or no?

Solution: yes

v. Index on `Tagged(mid)`. Yes or no?

Solution: no

vi. Index on `Tagged(pid)`. Yes or no?

Solution: yes

vii. Index on `Tagged(pid,mid)`. Yes or no?

Solution: yes

viii. Index on `Member(mid)`. Yes or no?

Solution: yes

ix. Index on `Member(name)`. Yes or no?

Solution: no

x. Index on `Member(age)`. Yes or no?

Solution: no

32. *(from CSE 414, Spring 2016, Final)*

An online picture sharing company uses a database with the following schema:

```
Users(uid, uname, city)
Picture(pid, author, size, pdf)
```

- `Users` stores all users; `uid` is the key.
- `Picture` stores their pictures; `pid` is the key; `author` is the `uid` of the picture's author; `size` represents the size of the picture in bytes; `pdf` is the actual pdf content of the picture.
- `uid`, `pid`, `author`, `size` are integers; `uname`, `city`, `pdf` are text.

(a) Write the SQL statements to create the tables for this database.

Solution:

```
drop table if exists Picture;
drop table if exists Users;

create table Users (
```

```
    uid int primary key,
    uname text not null,
    city text not null);

create table Picture (
     pid int primary key,
     author int not null references Users(uid),
     size int not null,
     pdf text);

insert into Users values(1,'Alice','Denver');
insert into Users values(2,'Bob','Denver');
insert into Users values(3,'Carol','Portland');
insert into Users values(4,'David','Denver');
insert into Users values(5,'Evan', 'Seattle');

insert into Picture values(10, 1, 1500000, 'abcd');
insert into Picture values(20, 1, 1500000, 'bcde');
insert into Picture values(30, 2, 1500000, 'cdef');
insert into Picture values(40, 1, 1500000, 'defg');
insert into Picture values(50, 2, 1500000, 'efgh');
insert into Picture values(60, 5,  500000, 'fghk');
insert into Picture values(70, 5, 4000000, 'ghkm');
```

(b) Write a SQL query that returns all users that have posted both a picture larger than 1MB (`size > 1000000`) and a picture smaller than 1MB. Your query should return the users' `uid` and `uname`

Solution:

```
select x.uid, x.uname
from users x, picture y, picture z
where x.uid = y.author and x.uid = z.author
  and y.size > 1000000 and z.size <= 1000000;
```

(c) Write a SQL query that retrieves all users who do not have any picture greater than 1MB (`size > 1000000`). Your query should return the users' `uid` and `uname`

Solution:

```
select x.uid, x.uname
from Users x
where not exists (select *
                  from Picture y
                  where y.size > 1000000 and x.uid = y.author);
```

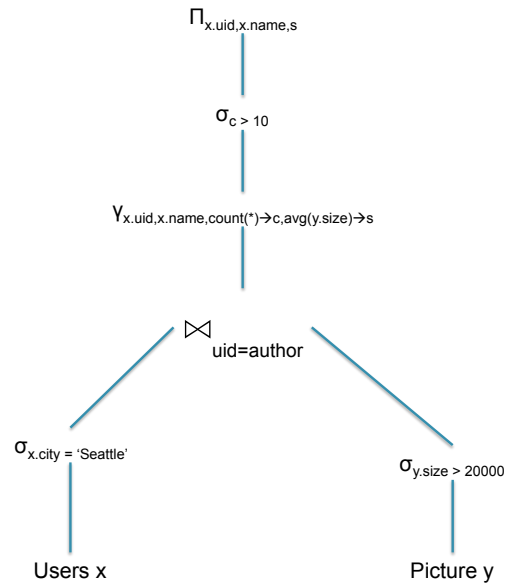
Other correct solutions use `not in` or $<$ `ALL`

(d) Write a Relational Algebra expression that is equivalent to the following SQL query:

```
select x.uid, x.uname, avg(y.size) as s
from Users x, Picture y
where x.uid = y.author
  and x.city = 'Seattle'
  and y.size > 20000
```

```
group by x.uid, x.uname
having count(*) > 10;
```

You may either write a Relational Algebra expression, or draw a query plan.



Solution:

(e) Consider the following query:

```
select x.uid, x.uname,
         (select count(*)
          from Picture y
          where x.uid = y.author and y.size > 1000000)
from Users x
where x.city = 'Denver';
```

For each query below indicate whether it is equivalent to the given query (meaning: it returns the same answers). All queries are syntactically correct.

i. Q1:

```
select x.uid, x.uname, count(*)
from Users x, Picture y
where x.uid = y.author  and x.city = 'Denver' and y.size > 1000000
group by x.uid, x.uname;
```

Solution: No

ii. Q2:

```
select x.uid, x.uname, count(y.pid)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
where x.city = 'Denver'
group by x.uid, x.uname, x.city;
```

Solution: Yes

iii. Q3:

```
select x.uid, x.uname, count(*)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

Solution: No

iv. Q4:

```
select x.uid, x.uname, count(y.pid)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

Solution: Yes

v. Q5:

```
select x.uid, x.uname, count(*)
from Users x, Picture y
where x.uid = y.author  and y.size > 1000000 and x.city = 'Denver'
group by x.uid, x.uname;
```

Solution: No

(f) The company merges with a local advertising company in Denver, which has a database of artistic pictures. Each artistic picture has a title, the pdf image, and may have several authors; all authors are from Denver. It was created using the following schema:

```
drop table if exists Artwork;
drop table if exists Author;

create table Artwork (
   wid int primary key,
   title text,
   pdf text);

create table Author (
   aid int primary key,
   wid int not null references Artwork(wid),
   aname text not null);
```

i. Some users and pictures occur in both databases. Write a SQL query that finds all common user,picture pairs. Two users are considered to be the same if their names and cities are the same; two pictures are considered to be the same if their pdf's are the same. Your query should return `uname, uid, aid, pid, wid`: the user name (which is the same in both databases), its two keys in `Users` and `Author`, and the two keys of the identical picture in `Picture` and `Artwork`.

Solution:

```
insert into Artwork values(101, 'Artwork by Alice and Fred', 'abcd');
insert into Artwork values(102, 'Artwork by Fred',          'xyzu');

insert into Author values(201, 101, 'Alice');
insert into Author values(202, 101, 'Fred');
insert into Author values(203, 102, 'Fred');

select distinct x.uid, v.aid, x.uname, y.pid, v.wid
from Users x, Picture y,
     Artwork u, Author v
where x.uid = y.author and u.wid = v.wid and x.city = 'Denver'
  and x.uname = v.aname and y.pdf = u.pdf;
```

ii. The new company creates a new schema for the integrated data:

```
create table NewUsers (
    nuid int primary key,
    nuName text not null,
    city text not null);

create table NewPicture (
     npid int primary key,
     title text,
     size int,
     pdf text);

create table Authored (
     nuid int not null references newUsers(nuid),
     npid int not null references NewPicture(npid));
```

Write a sequence of SQL queries that inserts all the data from the two old databases into the new database. You do not need to eliminate duplicates: that is, you will insert all the records from `Users`, `Picture` into the new schema, then will insert all the records from `Artwork`, `Author` into the new schema, without worrying about duplicates. All keys in the `Users`, `Picture` database are distinct from the keys in the `Artwork`, `Author` database.

Solution:

```
insert into NewUsers(nuid, nuName, city)
     select uid as nuid, uname as nuName, city from Users;

insert into NewPicture(npid, title, size, pdf)
     select pid as npid, null as title, size, pdf from Picture;

insert into Authored(nuid, npid)
     select author as nuid, pid as npid from Picture;

insert into NewUsers(nuid, nuName, city)
     select aid as nuid, aname as nuName, 'Denver' as city from Author;

insert into NewPicture(npid, title, size, pdf)
     select wid as npid, title, null as size, pdf from Artwork;

insert into Authored(nuid, npid)
```

```
select aid as nuid, wid as npid from Author;
```

33. *(from CSE 414, Spring 2016, Midterm)*

Consider the following database an online video rental service:

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

The `Customer` relation store all customers of the rental service, their names and the countries where they are located.

`Movie` stores all movies available for rent, their titles and the year of production.

When a customer rents a movie, a record is inserted in the relation `Rent`; a customer is allowed to rent a movie only once.

After watching the movie, a customer is allowed to rate the movie, with a score from 1 (worst) to 5 (best): this information is stored in the `Rate` table. Notice that a customer may rate a movie only if she/he has also rented the movie.

All primary keys are underlined. The attributes' types are as follows:

- `cid, mid, year, score` are integers.
- `name, country, title, date` are text.

CONTINUED ON NEXT PAGE

(a) Write SQL statements to create the tables for the rental service application. Make sure you choose the right types for the attributes, and define all key and foreign key constraints.

Solution: You can also use the data below to test the solutions.

```
drop table if exists Rent;
drop table if exists Rate;
drop table if exists Movie;
drop table if exists Customer;

create table Customer(cid int primary key, name text, country text);
create table Movie(mid int primary key, title text, year int);
create table Rent
  (cid int references Customer,
   mid int references Movie,
   date text,
   primary key (cid, mid));
create table Rate
  (cid int, mid int, score int,
   primary key (cid, mid),
   foreign key (cid, mid) references Rent);
```

(b) The *Canadian rating* of a movie is the average score received by that movie from customers in Canada. Write a SQL query that computes the average Canadian rating of each movie; order your answers by the Canadian rating, starting from the highest rating.

Solution:

```
select x.mid, x.title, avg(y.score) as s
from Movie x, Rate y, Customer z
where x.mid = y.mid and y.cid = z.cid
  and z.country = 'Canada'
group by x.mid, x.title
order by s desc;
```

(c) We want to return the `name` of all customers who were never rented a movie made before 2010; in other words, we want to find customers who rented only movies made in 2010 or later. Indicate which of the query or queries below answers this question correctly:

```
Q1 = select distinct x.name
     from Customer x, Rent y
     where x.cid = y.cid
       and not exists (select *
                       from Movie z
                       where y.mid = z.mid
                         and z.year < 2010);

Q2 = select distinct x.name
     from Customer x
     where not exists (select *
                       from Rent y, Movie z
                       where x.cid = y.cid
                         and y.mid = z.mid
                         and z.year < 2010);

Q3 = select distinct x.name
     from Customer x
     where not exists (select *
                       from Rent y
                       where x.cid = y.cid
                        and not exists
                            (select *
                             from Movie z
                             where y.mid = z.mid
                               and z.year >= 2010));

Q4 = select distinct x.name
     from Customer x left outer join Rent y on x.cid = y.cid
       left outer join Movie z on y.mid = z.mid and z.year < 2010
     group by x.name
     having count(z.mid) = 0;
```

Answer with any subset of Q1, Q2, Q3, Q4:

Solution: Q2, Q3, Q4

(d) Write a SQL query that returns the cid's and names of all customers who rented all the movies where "Alice" gave a score of 5. For example, if Alice gave a score of 5 to "Mad Max" and "Hunger Games" and gave no other score of 5, then your query should return the customers who rented both "Mad Max" and "Hunger Games".

Solution:

```
select *
from Customer x
where not exists
    (select *
     from Customer a, Rate b, Movie z
     where a.cid = b.cid and b.mid = z.mid
       and a.name = 'Alice' and b.score = 5
       and not exists (select * from Rent y
                       where x.cid = y.cid and y.mid = z.mid));
```

(e) Consider the following indexes:

```
create clustered index C_cid on Customer(cid);
create index C_name on Customer(name);
create index C_country on Customer(country);

create clustered index M_mid on Movie(mid);
create index M_year on Movie(year);

create index R_cid on Rate(cid);
create index R_mid on Rate(mid);
create index R_score on Rate(score);
```

For each query below, indicate which indexes might be used by the query optimizer in order to improve query performance. You may answer with more than one; if no index may be used, then write NONE. As usual, the optimizer makes uniformity assumptions when estimating if an index is useful.

i. Which indexes might be used by this query?

```
select distinct x.country
from Customer x
where x.name = 'Alice';
```

Solution: `C_name`

ii. Which indexes might be used by this query?

```
select distinct z.title
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and x.name = 'Alice' and y.score = 5;
```

Solution: `C_name,R_cid, M_mid`

iii. Which indexes might be used by this query?

```
select distinct x.name
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and y.score = 5 and z.year = '1910';
```

Solution: `M_year, R_mid, C_cid`

iv. Which indexes might be used by this query?

```
select distinct x.name
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and x.country = z.title and y.score = 5;
```

Solution: NONE

34. *(from CSE 344, Autumn 2017, Midterm)*

A company maintains a database about their employees and projects with the following schema:

```
Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)
```

`WorksOn` records which employee worked on which project; `salary` and `budget` represent yearly salary and budget respectively. An employee may work on multiple projects during the same year, and may also work on the same project during multiple years. All keys are underlined, and `WorksOn.eid`, `WorksOn.pid` are foreign keys to `Employee` and `Project` respectively.

(a) The *yearly salary expenses* of a project is the sum of all salaries of the employees who worked on that project during that year. Write a SQL query to find all the projects whose yearly salary expenses exceeded its budget. Return only the project title, and sort the projects in ascending order alphabetically by their title.

Solution:

```
select distinct z.title
from Employee x, WorksOn y, Project z
where x.eid = y.eid and y.pid = z.pid
group by z.pid, z.title, y.year, z.budget
having sum(salary) > z.budget
order by z.title;
```

(b) We say that an employee worked *intermittently* on a project p if she worked on p during one year, then did not work on p during a later year, then worked again on p during an even later year. For example if Alice worked on the project during 2012, 2013, and 2017 then we say that she worked intermittently on p; if Bob worked on that project during 2013, 2014, 2015 and no other years, then we say he worked continuously. Write a SQL query to retrieve all employees that worked intermittently on some project. Return the employee name, and the project title.

Solution:

```
select distinct u.name, v.title
from Employee u, WorksOn x, WorksOn y, Project v
where u.eid = x.eid and u.eid = y.eid
  and x.pid = v.pid and y.pid = v.pid
  and x.year + 1 < y.year
  and not exists (select *
                  from WorksOn z
                  where u.eid = z.eid and v.pid = z.pid
                    and x.year < z.year and z.year < y.year);
```

(c) For each question below indicate whether the two SQL queries are equivalent. Assume that the database does not contain any NULL values.

i. Are `Q1` and `Q2` equivalent?

```
Q1: select W.year, W.pid, count(*)
    from WorksOn W
    group by W.year, W.pid;

Q2: select distinct W.year, W.pid,
                (select count(*)
                 from WorksOn W2
                 where W.year=W2.year and W.pid = W2.pid)
    from WorksOn W;
```

Yes/No:

Solution: Yes

ii. Are `Q3` and `Q4` equivalent?

```
Q3: select W.year, W.pid, count(*)
    from WorksOn W, Employee E
    where W.eid = E.eid and E.salary > 100000
    group by W.year, W.pid;

Q4: select W.year, W.pid,
           (select count(*)
            from Employee E
            where W.eid = E.eid and E.salary > 100000)
    from WorksOn W;
```

Yes/No:

Solution: No

iii. Are `Q5` and `Q6` equivalent?

```
Q5: select distinct E.name
    from Employee E, WorksOn W, WorksOn W2
    where E.eid = W.eid and E.eid = W2.eid
      and W.pid = W2.pid
      and W.year > 2010 and W2.year < 2015;

Q6: select distinct E.name
    from Employee E, WorksOn W
    where E.eid = W.eid
      and W.year > 2010;
```

Yes/No:

Solution: No

iv. Are `Q7` and `Q8` equivalent?

```
Q7: select distinct E.name
    from Employee E, WorksOn W, WorksOn W2
    where E.eid = W.eid and E.eid = W2.eid
      and W.pid = W2.pid
      and W.year < 2010 and W2.year < 2015;

Q8: select distinct E.name
    from Employee E, WorksOn W
    where E.eid = W.eid
      and W.year < 2010;
```

Yes/No:

Solution: Yes

(d) Consider the following database instance:

`Employee`

eid	name	salary
1	Alice	1000
2	Bob	NULL

`WorksOn`

eid	pid	year
1	10	NULL
1	20	2015
2	20	NULL

`Project`

pid	title	budget
10	OS	NULL
20	ML	5000

Indicate for each query below what answers it returns; write "empty" if the answer is the empty set.

i. What does query `Q1` return?

```
Q1: select x.name, z.title
    from Employee x, WorksOn y, Project z
    where x.eid = y.eid and y.pid = z.pid
      and (y.year = 2015 or y.year != 2015);
```

Solution: `Alice ML`

ii. What does query `Q2` return?

```
Q2: select x.name, z.title
    from Employee x, WorksOn y, Project z
    where x.eid = y.eid and y.pid = z.pid
      and ( (salary = 1000 and year = 2015) or budget = 5000);
```

Solution:
```
Alice ML
Bob   ML
```

iii. What does query `Q3` return?

```
Q3: select x.name, z.title
    from Employee x, WorksOn y, Project z
    where x.eid = y.eid and y.pid = z.pid
      and (salary = 1000 or budget = 5000);
```

Solution:
```
Alice OS
Alice ML
Bob   ML
```

iv. What does query `Q4` return?

```
Q4: select x.name, z.title
    from Employee x, WorksOn y, Project z
    where x.eid = y.eid and y.pid = z.pid
      and ((salary = 1000 or year = 2015) and not(budget = 5000));
```

Solution: Empty.

35. *(from CSE 344, Winter 2019, Midterm)*

An online shopping company stores a database about its products, customers, and orders, with the following schema:

`Product(`<u>`pid`</u>`,pname,price)`
`Orders(`<u>`oid`</u>`,pid,cid,qnt,date)`
`Customer(`<u>`cid`</u>`,cname,city)`

Products have an ID, a name, and a sell price

Customers have an ID, a name, and the city where they live.

Each order is for one product and one customer. The attribute `qnt` is the quantity, i.e. the number of units of the product placed in that order; `qnt` > 0.

(a) Write the sequence of SQL statements necessary to create the tables above. The attributes pid, oid, cid, qnt are integers, price is a real number, date is of type DATE and all other attributes are text. Include all keys or foreign keys declarations.

Solution: You can also use this data to test the solutions to the other questions.

```
DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Product;
DROP TABLE IF EXISTS Customer;

CREATE TABLE Product(pid INT PRIMARY KEY, pname TEXT, price FLOAT);
CREATE TABLE Customer(cid INT PRIMARY KEY, cname TEXT, city TEXT);
CREATE TABLE Orders(oid INT PRIMARY KEY,
                    pid INT REFERENCES Product,
                    cid INT references Customer, qnt INT, date DATE);
```

(b) Write a SQL query that returns, for each product, the quantity of those products sold to customers in Seattle. Your query should return the pid, product name, and total quantity sold in Seattle, sorted in decreasing order of the total quantity. Your query should include products with a total = 0.

Solution:

```
select x.pid, x.pname, count(y.oid)
from Product x
     left outer join Orders y on x.pid=y.pid
     left outer join Customer z on y.cid=z.cid and z.city='Seattle'
group by x.pid, x.pname
order by count(y.oid);
```

(c) For each city, return the most expensive product sold in that city. Your query should return the city name, the `pid`, `pname`, and `price` of the most expensive product sold there; in case of a tie (two or more products had the same maximum price), then return all of those products. If nothing was ordered in a city, then your query does not need to return that city.

Solution: Solution 1:

```
with temp as
   (select z.city, max(x.price) as mp
    from Product x, Orders y, Customer z
    where x.pid = y.pid and y.cid = z.cid
    group by z.city)
select u.city, x.price
from Product x, Orders y, Customer z, temp u
where x.pid = y.pid and y.cid = z.cid
  and z.city=u.city and x.price=u.mp;
```

Solution 2:

```
select u.city, w.price
from Product x, Orders y, Customer z,
     Product w, Orders v, Customer u
where x.pid = y.pid and y.cid = z.cid
  and w.pid = v.pid and v.cid = u.cid
  and z.city = u.city
group by u.city, w.price
having w.price >= max(x.price);
```

(d) Consider the following query:

```
--  Q:
select distinct z.cid, z.cname
from Product x, Orders y, Customer z
where x.pid = y.pid and y.cid = z.cid
  and x.price < 200
  and y.qnt > 10
  and z.city = 'Seattle';
```

For each query below, indicate whether return the same answer or not. You only need to answer Y or N.

i. Is this query equivalent to Q?

```
--  Q1:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y2.qnt > 10
  and z.city = 'Seattle';
```

Yes or No?

Solution: N

ii. Is this query equivalent to Q?

```
--  Q2:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y1.qnt > 10
  and z.city = 'Seattle';
```

Yes or No?

Solution: Y

iii. Is this query equivalent to Q?

```
--  Q3:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200  and x2.price < 300
  and y1.qnt > 10 and y2.qnt > 5
  and z.city = 'Seattle';
```

Yes or No?

Solution: Y

iv. Is this query equivalent to Q?

```
--  Q4:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200  and x2.price < 100
  and y1.qnt > 10 and y2.qnt > 50
  and z.city = 'Seattle';
```

Yes or No?

Solution: N

v. Is this query equivalent to Q?

```
--  Q5:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z, Customer z2
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y1.qnt > 10
  and z.city = 'Seattle' and z2.city = 'Portland';
```

Yes or No?

Solution: N

36. *(from CSE 414, Spring 2019, Midterm)*

The city of Nullville maintains a database of all cars in the city, and records their license plates whenever they drive downtown:

```
Car(lp, make, owner)
Downtown(lp, day)
```

- The relation `Car` stores information about all the cars registered in the city. All its attributes are of type `text`; `lp` it is the primary key in `Car`, `make` represents the make of that car (Honda, Ford, etc) and `owner` is the name of the registered owner. Notice that an owner may have multiple cars.
- The relation `Downtown` represents days when a particular car drove downtown. The attribute `lp` is a foreign key to `Car`, and `day` is an integer that represents the day number, counting from the moment when the database became operational (assume this was about ten years ago). There is at most one record for each car and day: if a car drives downtown multiple times during one day, then we only record that once.

(a) Write the sequence of SQL statements necessary to create the tables above. Include all keys or foreign keys declarations.

Solution:

```
DROP TABLE IF EXISTS Downtown;
DROP TABLE IF EXISTS Car;

CREATE TABLE Car(lp text primary key, make text, owner text);
CREATE TABLE Downtown(lp text references Car, day int);

-- for instructor's testing purposes only
insert into car values ('XY52Z','Honda','Alice');
insert into car values ('ZY23X','Bentley','Bob');
insert into car values ('MWM92','Bentley','Alice');
insert into downtown values ('MWM92', '2000');
insert into downtown values ('MWM92', '3000');
insert into downtown values ('ZY23X', '4000');
insert into downtown values ('MWM92', '5000');
```

(b) Write a SQL query that computes, for each owner, how many cars they own. Your query should return a a set of `owner`, `count` pairs, sorted in decreasing order of the `count`.

Solution:

```
select x.owner, count(*) as Count
from Car x
group by x.owner
order by count(*) desc;
```

(c) Write a SQL query that returns, for each car `make` the total number of cars of that type that entered downtown on or after day 1000. Your query should count every day when a car passed through downtown, in other words, if the same car passes through downtown on days 2200, 2250, and 3100, then you count that as three. Your answer should consists of a set of `make`, `count` pairs, sorted in decreasing order of the `count`, like this:

Make	Count
Honda	4201
Fiat	3477
Ford	2010
...	
Bentley	0

Your query should include the makes of the cars that were never driven downtown, for example Bentley above.

Solution:

```
select x.make, count(y.lp) as Count
from Car x left outer join Downtown y
  on x.lp = y.lp and y.day >= 1000
```

```
group by x.make
order by count(*) desc;
```

(d) Find all owners who have not driven to downtown on or after day 1000. Notice that some owners may own multiple cars; you need to return them only if none of their cars drove downtown on or after day 1000. Your query should return a set of owners; include each owner only once.

Solution: Solution 1:

```
select distinct u.owner
from car u
where not exists (select *
                  from car x, downtown y
                  where x.lp = y.lp
                    and y.day >= 1000
                    and x.owner = u.owner);
```

Solution 2:

```
select distinct u.owner
from car u
where u.owner not in (select x.owner
                  from car x, downtown y
                  where x.lp = y.lp
                    and y.day >= 1000);
```

Solution 3 (not quite correct since `max(y.day)` only works for nonempty sets; in other words this solution fails to return owners who never drove downtown).

```
select distinct u.owner
from car u, car x, downtown y
where u.owner = x.owner
  and x.lp = y.lp
group by u.owner
having max(y.day) < 1000;
```

Solution 4

```
select distinct x.owner
from car x left outer join downtown y
  on x.lp = y.lp and y.day >= 1000
group by x.owner
having count(y.lp) = 0;
```

(e) Consider the following query:

```
--  Q:
select distinct x.owner
from car x, downtown y
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 1000;
```

For each query below, indicate whether return the same answer or not. You only need to answer Y or N.

i. Is this query equivalent to Q?

```
--  Q1:
select distinct x.owner
```

```
from car x, downtown y, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 5000
  and x.lp = v.lp
  and v.day >= 1000;
```

Yes or No?

Solution: N

ii. Is this query equivalent to Q?

```
--  Q2:
select distinct x.owner
from car x, downtown y, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 200
  and x.lp = v.lp
  and v.day >= 1000;
```

Yes or No?

Solution: Y

iii. Is this query equivalent to Q?

```
--  Q3:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 1000
  and x.owner = u.owner
  and u.lp = v.lp;
```

Yes or No?

Solution: Y

iv. Is this query equivalent to Q?

```
--  Q4:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and x.owner = u.owner
  and u.lp = v.lp
  and v.day >= 1000;
```

Yes or No?

Solution: N

v. Is this query equivalent to Q?

```
--  Q5:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.lp = y.lp
  and y.day >= 1000
  and x.owner = u.owner
  and u.make = 'Honda'
  and u.lp = v.lp
  and v.day = y.day;
```

Yes or No?

Solution: y

(f) Consider the following table:

R :

A	B
3	5
3	NULL
NULL	5

Answer the following questions:

i. What does the following query return?

```
select * from R where (A=3) and not(B=3);
```

Solution:

A	B
3	5

ii. What does the following query return?

```
select * from R where (A=3) or not(B=3);
```

Solution:

A	B
3	5
3	NULL
NULL	5

iii. What does the following query return?

```
select * from R where (A=3) or (A!=3);
```

Solution:

A	B
3	5
3	NULL

iv. What does the following query return?

```
select * from R where A+B != 8;
```

Solution: empty

v. What does the following query return?

```
select * from R where A is NULL or B is NULL;
```

Solution:

A	B
NULL	5
3	NULL

37. *(from CSE 544p, Spring 2021, Final)*

A blog Website allows users to post blogs and to add discussion to other user's blogs. The schema is:

```
Blog(bid, author, body)
Discussion(bid, ts, uid, vote, content)
Usr(uid, name)
```

The database has the following constraints:

- `Discussion.bid` is a foreign key to `Blog`
- `Discussion.ts` is an integer timestamp
- `Discussion.uid` is a foreign key to `Usr`
- `Blog.author` is a foreign key to `Usr`
- `Discussion.vote` is number that is either $+1$ or -1 (up or down).
- All attributes are `NOT NULL`

(a) A user is a *follower* of some other user if she commented (i.e. added a discussion) on at least one blog posted by that the other user. Write a SQL query that retrieves all users with at least 50 followers. You should return the `uid`, `name`, and number of followers. Note that, if a user posts n blogs and another user comments on all of them, then the first user still has only one follower.

Solution: Test data:

```
drop table if exists discussion;
drop table if exists blog;
drop table if exists usr;

create table usr(uid int primary key, name text);
create table blog(bid int primary key, author int references usr, body text);
create table discussion(uid int references usr, bid int references blog, vote int, body text, ts int);

insert into usr values(1, 'Alice');
insert into usr values(2, 'Bob');
insert into usr values(3, 'Carol');

insert into blog values(10, 1, 'asdf');
insert into blog values(20, 1, 'asdf');
insert into blog values(30, 1, 'asdf');
insert into blog values(40, 2, 'asdf');
insert into blog values(50, 2, 'asdf');

insert into discussion values(1, 40,  1, 'blah', 1);
insert into discussion values(2, 10, -1, 'blah', 1);
insert into discussion values(3, 20,  1, 'blah', 1);
insert into discussion values(3, 40, -1, 'blah', 2);
```

Solution to the first question:

```
select u.uid, u.name, count(distinct y.uid)
from discussion y, blog z, usr u
where y.bid = z.bid and z.author = u.uid
  and y.uid != u.uid
group by u.uid, u.name
having count(distinct y.uid) >= 50;
```

(b) A *leader* is a user for which every blog has a discussion from every other user. Write a SQL query to find all leaders.

Solution:

```
select x.uid, x.name
from usr x
where not exists
     (select *
      from blog y, usr z   -- find a blog y,
                           -- and user z
                           -- s.t. z did not discuss y
      where x.uid = y.author
       and x.uid != z.uid
       and not exists
             (select *
              from discussion w
              where w.bid = y.bid and w.uid = z.uid));
```

(c) Some users repost their discussion: in this problem you will write a SQL query to delete all duplicate discussions. A discussion is a duplicate if it refers to the same blog, is written by the same user, and has the same content and vote as a previous discussion. Write a SQL query to remove all duplicate discussions.

Solution:

```
delete from discussion y
   where exists
           (select *
            from discussion x
            where y.bid = x.bid
              and y.uid = x.uid
              and y.vote = x.vote
              and y.body = x.body
              and y.ts > x.ts);
```

38. *(from CSE 414, Spring 2024, Midterm Review)*

A doctor's office has several doctors, and several patients. They keep a database of appointments with the following schema:

`Doctor(did,dname,speciality)`
`Patient(pid,pname,dob)`
`Visit(did,pid,date)`

Doctors have an ID, a name, and a speciality (e.g. "orthopedics" or "dermatology" or "neurology" etc). The key is `Doctor.did`.

Patients have an ID, a name, and date of birth (dob). The key is `Patient.`*pid*.

`Visit.did` is a foreign key to `Doctor` and `Visid.pid` is a foreign key to `Patient`.

The attributes `did`, `pid` are integers. All other attributes are strings.

In some of the questions below we will illustrate with the following test database instance. Please ensure that your answers are correct on *any* database instance, not just this one.

Doctor

did	DName	Speciality
10	Alice	surgeon
20	Bob	ophthalmologist
30	Carol	surgeon
40	David	surgeon
50	Eve	pediatrics

Patient

pid	PName	Dob
100	Agnes	1990-10-10
200	Brian	1998-12-01
300	Cathy	2001-01-15
400	Dylan	2003-11-12

Visit

did	pid	date
20	100	2018-05-10
30	100	2019-05-10
40	200	2020-05-10
10	300	2021-05-10
40	200	2022-01-02

(a) Write the sequence of SQL statements necessary to create the tables above. Include all keys or foreign keys declarations.

Write your answer here:

Solution:

```
-- you only need to answer with the CREATE TABLE statement;
-- the other commands are for the instructor only

DROP TABLE IF EXISTS Visit;
DROP TABLE IF EXISTS Doctor;
```

```
DROP TABLE IF EXISTS Patient;

CREATE TABLE Doctor(did INT PRIMARY KEY, dname TEXT, speciality TEXT);
CREATE TABLE Patient(pid INT PRIMARY KEY, pname TEXT, dob TEXT);
CREATE TABLE Visit(did INT REFERENCES Doctor,
                   pid INT REFERENCES Patient,
                   date text);

insert into Doctor values
  (10, 'Alice', 'surgeon'),
  (20, 'Bob', 'ophthalmologist'),
  (30, 'Carol', 'surgeon'),
  (40, 'David', 'surgeon'),
  (50, 'Eve', 'pediatrics');

insert into Patient values
  (100, 'Agnes', '1990-10-10'),
  (200, 'Brian', '1998-12-01'),
  (300, 'Cathy', '2001-01-15'),
  (400, 'Dylan', '2003-11-12');

insert into Visit values
  (20, 100, '2018-05-10'),
  (30, 100, '2019-05-10'),
  (40, 200, '2020-05-10'),
  (10, 300, '2021-05-10'),
  (40, 200, '2022-01-02');
```

For this and the remaining questions you may assume that the dates are stored in a normalize form as `yyyy-mm-dd`, for example `2019-01-05` means January, 5th, 2019. This applies to both `Patient.dob` and `Visit.date`.

(b) Write a SQL query that computes, for each patient, the number of visits before 2021. Return the pid's, their names, and the number of times they visited before 2021.

On our test database instance your query should return the following result. Notice that the count for Cathy is 0, because her visit was during 2021.

Doctor

did	DName	Speciality
10	Alice	surgeon
20	Bob	ophthalmologist
30	Carol	surgeon
40	David	surgeon
50	Eve	pediatrics

Patient

pid	PName	Dob
100	Agnes	1990-10-10
200	Brian	1998-12-01
300	Cathy	2001-01-15
400	Dylan	2003-11-12

Visit

did	pid	date
20	100	2018-05-10
30	100	2019-05-10
40	200	2020-05-10
10	300	2021-05-10
40	200	2022-01-02

Query answer

pid	PName	count
100	Agnes	2
200	Brian	1
300	Cathy	0
400	Dylan	0

Write your answer here:

Solution: Solution 1:

```
select x.pid, x.pname, count(y.pid) as cnt
from Patient x left outer join Visit y
  on x.pid = y.pid and y.date < '2021-01-01'
group by x.pid, x.pname;
```

Solution 2:

```
select x.pid, x.pname,
       (select count(*)
        from Visit y
        where x.pid = y.pid
          and y.date < '2022-01-01') as cnt
from Patient x;
```

(c) Write a SQL query that returns all patients who have visited both a `surgeon` and an `ophthalmologist`. Return their PID, name, and date of birth.

On our test database instance your query should return:

Doctor

did	DName	Speciality
10	Alice	surgeon
20	Bob	ophthalmologist
30	Carol	surgeon
40	David	surgeon
50	Eve	pediatrics

Patient

pid	PName	Dob
100	Agnes	1990-10-10
200	Brian	1998-12-01
300	Cathy	2001-01-15
400	Dylan	2003-11-12

Visit

did	pid	date
20	100	2018-05-10
30	100	2019-05-10
40	200	2020-05-10
10	300	2021-05-10
40	200	2022-01-02

Query answer

pid	PName	dob
100	Agnes	1990-10-10

Write your answer here:

Solution:

```
select distinct x.pid, x.pname, x.dob
from Patient x, Visit y1, Visit y2, Doctor z1, Doctor z2
where x.pid = y1.pid and y1.did = z1.did and z1.speciality='surgeon'
  and x.pid = y2.pid and y2.did = z2.did and z2.speciality='ophthalmologist';
```

(d) Write a SQL query that retrieves, for each speciality, the name of the doctor with that speciality that had the first (earliest) appointment. Return the DID, name and speciality of the doctor, as well as his/her date of the earliest appointment. In case of ties (meaning: if there are two or more doctors in the same speciality that had the same earliest appointment), return all of them.

On our test database instance your query should look as follows. Notice that pediatrics is not included in the answer because there were no visits to pediatrics.

Doctor

did	DName	Speciality
10	Alice	surgeon
20	Bob	ophthalmologist
30	Carol	surgeon
40	David	surgeon
50	Eve	pediatrics

Patient

pid	PName	Dob
100	Agnes	1990-10-10
200	Brian	1998-12-01
300	Cathy	2001-01-15
400	Dylan	2003-11-12

Visit

did	pid	date
20	100	2018-05-10
30	100	2019-05-10
40	200	2020-05-10
10	300	2021-05-10
40	200	2022-01-02

Query answer

did	DName	speciality	date
20	Bob	ophthalmologist	2018-05-10
30	Carol	surgeon	2019-05-10

Write your answer here:

Solution: Solution 1:

```
select distinct x1.did, x1.dname, x1.speciality, y1.date
from Doctor x1, Visit y1, Doctor x2, Visit y2
where x1.did = y1.did and x2.did = y2.did
  and x1.speciality = x2.speciality
group by x1.did, x1.dname, x1.speciality, y1.did, y1.date
having y1.date = min(y2.date);
```

Solution 2:

```
with earlydate as
  (select x2.speciality, min(y2.date) as m
   from doctor x2, visit y2
   where x2.did = y2.did
   group by x2.speciality)
select distinct x1.did, x1.dname, x1.speciality, y1.date
from doctor x1, visit y1, earlydate z
where x1.did = y1.did
  and x1.speciality = z.speciality
  and y1.date = z.m;
```

39. *(from CSE 414, Spring 2024, Midterm Review)*

We will use the same database schema as in Part 38.

(a) Which of the following SQL queries are monotone?

i. Is this query monotone?

```
select distinct x.did, x.dname
from doctor x, visit y
where not (x.did = y.did and y.date < '2000');
```

Yes or No?

Solution: Y

ii. Is this query monotone?

```
select x.did, x.dname
from doctor x
where exists
        (select *
         from visit y
         where x.did = y.did
           and y.date < '2020');
```

Yes or No?

Solution: Y

iii. Is this query monotone?

```
select x.did, x.dname
from doctor x
where exists
        (select *
```

```
        from visit y
        where not (x.did = y.did
                   and y.date < '2020'));
```

Yes or No?

Solution: Y

iv. Is this query monotone?

```
select x.did, x.dname
from doctor x
where not exists
       (select *
        from visit y
        where x.did = y.did
          and y.date < '2020');
```

Yes or No?

Solution: N

v. Is this query monotone?

```
select x.did, x.dname
from doctor x
where not exists
       (select *
        from visit y
        where not (x.did = y.did
                   and y.date < '2020'));
```

Yes or No?

Solution: N

(b) We want to answer the following question:

> Retrieve all patients who did not visit the doctor's office before 2021, except for surgery appointments.

In other words, we want the patients who did not visit before 2021, but the surgery appointments don't count. On our test database instance, the query should return Brian, Cathy and Dylan: Brian did visit before 2021 (on 2020-05-10) but that was for surgery, Cathy did not visit before 2021 (she only visited on 2021-05-10), while Dylan never visited. On the other hand, Agnes should not be returned because she visited an ophthalmologist on 2018-05-10.

as follows:

Doctor

did	DName	Speciality
10	Alice	surgeon
20	Bob	ophthalmologist
30	Carol	surgeon
40	David	surgeon
50	Eve	pediatrics

Patient

pid	PName	Dob
100	Agnes	1990-10-10
200	Brian	1998-12-01
300	Cathy	2001-01-15
400	Dylan	2003-11-12

Visit

did	pid	date
20	100	2018-05-10
30	100	2019-05-10
40	200	2020-05-10
10	300	2021-05-10
40	200	2022-01-02

Query answer

pid	PName
200	Brian
300	Cathy
400	Dylan

For each of the SQL queries below indicate if it returns the correct answer to this question. In each case, answer *Yes* or *No*.

i. Does the query below return the correct answer?

```
select x.pid, x.pname
from patient x
where exists
        (select *
         from visit y, doctor z
         where x.pid = y.pid and y.did = z.did
          and z.speciality =  'surgeon'
          and y.date < '2021-01-01');
```

Yes or No?

Solution: N

Solution: This SQL query is monotone; our query is not

ii. Does the query below return the correct answer?

```
select x.pid, x.pname
from patient x
where exists
        (select *
         from visit y, doctor z
         where x.pid = y.pid and y.did = z.did
          and not(z.speciality =  'surgeon')
          and y.date < '2021-01-01');
```

Yes or No?

Solution: N

Solution: This SQL query is still monotone; our query is not

iii. Does the query below return the correct answer?

```
select x.pid, x.pname
from patient x
where not exists
        (select *
         from visit y, doctor z
         where x.pid = y.pid and y.did = z.did
```

```
          and z.speciality = 'surgeon'
          and y.date < '2021-01-01');
```

Yes or No?

Solution: N

Solution: This SQL query is not monotone, but returns the wrong answer: it returns patients who did not visit before 2021 for surgery. We need patients who did not visit before 2021 for other things than surgery.

iv. Does the query below return the correct answer?

```
select x.pid, x.pname
from patient x
where not exists
        (select *
         from visit y, doctor z
         where x.pid = y.pid and y.did = z.did
          and not(z.speciality = 'surgeon')
          and y.date < '2021-01-01');
```

Yes or No?

Solution: Y

v. Does the query below return the correct answer?

```
select x.pid, x.pname
from patient x, visit y
where x.pid = y.pid
  and y.date < '2021-01-01'
  and not exists
        (select *
         from doctor z
         where y.did = z.did
          and not(z.speciality = 'surgeon'));
```

Yes or No?

Solution: N

Solution: This query is non-sense. It returns all patients who visited before 2021, but not a surgeon. The not-exists subquery is equivalent to saying that the doctor `y.did` (which is unique, because `doctor.did` is a key) is not a surgeon.

40. *(from CSE 414, Spring 2024, Final; CSE 344, Spring 2026, Makeup Final)*

A large retailer stores the data about their local stores, their warehouses, and the shipments from warehouses to store, using the following schema:

```
Store(sid, city)
Shipment(sid, wid, product, date)
Warehouse(wid, city)
```

- `Store` is a table containing all local stores, and their addresses (the city where they are located); `sid` is the key of the store.
- `Warehouse` is a table containing all warehouses, and their addresses (the city where they are located); `sid` is the key of the warehouse.
- `Shipment` records each shipment from a warehouse to a store together with the product that was shipped and the date of the shipment.

The attributes `sid,wid` are integers, the rest are `TEXT`. The `date` attribute is stored in the format YYYY-MM-DD.

In some of the questions below we will illustrate with the following test database instance. Please ensure that your answers are correct on any database instance, not just this one.

Store

sid	city
123	Seattle
234	Boston
345	Boston
456	Dallas
567	Miami

Shipment

sid	wid	product	date
123	200	TV	2022-01-01
123	400	iPad	2020-05-31
234	100	iPad	2020-05-20
234	100	iPhone	2021-10-10
234	600	iPad	2020-05-20
345	400	iPad	2022-03-03
456	600	TV	2024-01-01
456	300	TV	2024-01-02

Warehouse

wid	city
100	Denver
200	Seattle
300	Denver
400	Boston
500	Denver
600	Seattle

Solution:

```
drop table if exists shipment;
drop table if exists store;
drop table if exists warehouse;
create table store(sid int primary key, city text);
create table warehouse(wid int primary key, city text);
create table shipment(sid int references store,
                      wid int references warehouse,
                      product text,
                      date text);

insert into store values
  (123, 'Seattle'),
  (234, 'Boston'),
  (345, 'Boston'),
  (456, 'Dallas'),
```

```
  (567, 'Miami');
insert into warehouse values
  (100, 'Denver'),
  (200, 'Seattle'),
  (300, 'Denver'),
  (400, 'Boston'),
  (500, 'Denver'),
  (600, 'Seattle');

insert into shipment values
  (123, 200, 'TV', '2022-01-01'),
  (123, 400, 'iPad', '2020-05-31'),
  (234, 100, 'iPad', '2020-05-20'),
  (234, 100, 'iPhone', '2021-10-10'),
  (234, 600, 'iPad', '2020-05-20'),
  (345, 400, 'iPad', '2022-03-03'),
  (456, 600, 'TV', '2024-01-01'),
  (456, 300, 'TV', '2024-01-02');
```

(a) Write a SQL query that computes for each city the number of TV shipments sent from warehouses in that city. Your query should return the `city` and the `count` of the number of TV shipments.

On our example database:

Store

sid	city
123	Seattle
234	Boston
345	Boston
456	Dallas
567	Miami

Shipment

sid	wid	product	date
123	200	TV	2022-01-01
123	400	iPad	2020-05-31
234	100	iPad	2020-05-20
234	100	iPhone	2021-10-10
234	600	iPad	2020-05-20
345	400	iPad	2022-03-03
456	600	TV	2024-01-01
456	300	TV	2024-01-02

Warehouse

wid	city
100	Denver
200	Seattle
300	Denver
400	Boston
500	Denver
600	Seattle

your query should return:

city	count
Seattle	2
Boston	0
Denver	1

For example, the two warehouses in Seattle 200 and 600 sent 3 shipments: a TV, an iPad, a TV; therefore its count is 2. There is only one warehouse in Boston, 400, and it only shipped an iPad; therefore its count is 0.

Write your answer here:

Solution:

```
select z.city, count(y.wid)
from Warehouse z left outer join Shipment y
     on y.wid = z.wid and y.product = 'TV'
group by z.city;
```

(b) Write a SQL query that finds for each store the earliest shipment to that store. Your query should return the `sid` and `city` of the store, the `product` and `date` of that earliest shipment, and the `wid` and `city` that sent that shipment.

On our example database:

Store

sid	city
123	Seattle
234	Boston
345	Boston
456	Dallas
567	Miami

Shipment

sid	wid	product	date
123	200	TV	2022-01-01
123	400	iPad	2020-05-31
234	100	iPad	2020-05-20
234	100	iPhone	2021-10-10
234	600	iPad	2020-05-20
345	400	iPad	2022-03-03
456	600	TV	2024-01-01
456	300	TV	2024-01-02

Warehouse

wid	city
100	Denver
200	Seattle
300	Denver
400	Boston
500	Denver
600	Seattle

your query should return:

sid	scity	product	date	wid	wcity
123	Seattle	iPad	2020-05-31	400	Boston
234	Boston	iPad	2020-05-20	100	Denver
234	Boston	iPad	2020-05-20	600	Seattle
345	Boston	iPad	2022-03-03	400	Boston
456	Dallas	TV	2024-01-01	600	Seattle

For example, the two shipments arriving at the store (123, Seattle) were on 2022-01-01 and 2020-05-31: the query returns the second one because it was earlier than the first. There were two shipments arriving at the store (234, Boston), both on the same date, and therefore the query returns both. There was no shipment to (567, Miami), and the query doesn't return it.

Write your answer here:

Solution:

```
with Early as
   (select y1.sid as sid, min(y1.date) as mindate
    from Shipment y1
    group by y1.sid)
select x.sid, x.city as scity, y.product, y.date, z.wid, z.city
from Store x, Shipment y, Warehouse z, Early u
where x.sid = y.sid and y.wid = z.wid and y.date = u.mindate
order by x.sid, z.wid;
```

or

```
select x.sid, x.city as scity, y.product, y.date, z.wid, z.city
from Store x, Shipment y, Warehouse z, Shipment y1
where x.sid = y.sid and y.wid = z.wid and x.sid = y1.sid
group by x.sid, x.city, y.product, y.date, z.wid, z.city
having y.date = min(y1.date)
order by x.sid, z.wid;
```

or

```
select x.sid, x.city as scity, y.product, y.date, z.wid, z.city
from Store x, Shipment y, Warehouse z
```

```
where x.sid = y.sid and y.wid = z.wid
  and y.date <= ALL (select y1.date from Shipment y1 where x.sid=y1.sid)
order by x.sid, z.wid;
```

(c) Find all stores that received shipments only from their own cities. Your query should return the `sid` and `city` of each such store.

On our example database:

Store

sid	city
123	Seattle
234	Boston
345	Boston
456	Dallas
567	Miami

Shipment

sid	wid	product	date
123	200	TV	2022-01-01
123	400	iPad	2020-05-31
234	100	iPad	2020-05-20
234	100	iPhone	2021-10-10
234	600	iPad	2020-05-20
345	400	iPad	2022-03-03
456	600	TV	2024-01-01
456	300	TV	2024-01-02

Warehouse

wid	city
100	Denver
200	Seattle
300	Denver
400	Boston
500	Denver
600	Seattle

your query should return:

sid	city
345	Boston
567	Miami

The store (345, Boston) received a single shipment, which was from the warehouse (400,Boston) in the same city. Store (567, Miami) didn't receive any shipment, in particular it didn't receive any shipment from a city other than Miami, hence it is also included in the answer.

Write your answer here:

Solution:

```
select x.sid, x.city
from Store x
where not exists
     (select * from Shipment y, Warehouse z
      where x.sid = y.sid and y.wid = z.wid
       and x.city != z.city);
```

41. *(from CSE 344, Autumn 2024, Final; CSE 344, Spring 2026, Final)*

The game of TRIXMATE has become very popular. Registered players may participate at the annual competition, and have an official `level`, which is determined based on their historical performance at past competitions. New players start from `level` 0.

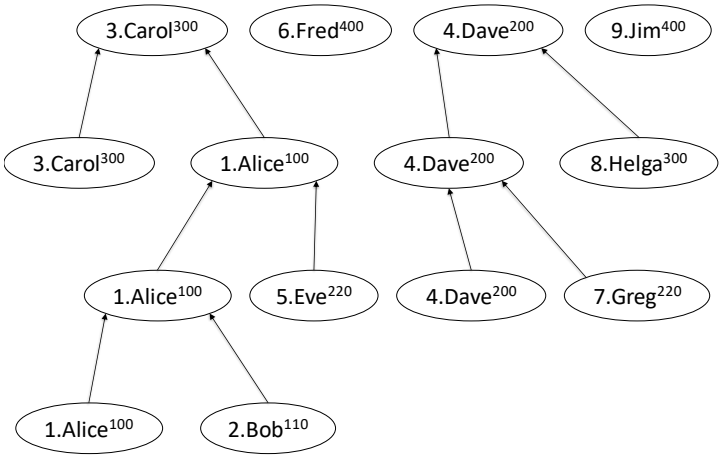
The annual competition is a single elimination tournament: after each game, the winner advances to the next round, while the loser is eliminated from the competition. The database for the competition has the following schema:

```
Player(pid, name, level)

Game(pidWin, pidLose)
```

`Player` stores all registered players. `Game` stores all games played so far during the current competition. `pidWin` and `pidLose` are the winning and losing player in the game: both attributes are foreign keys to `Player`. All attributes in the two tables are of type `int`, except for `Player.name` which is `text`.

(Continued on the next page)

For example, consider the current state of the competition:



Alice won against Bob and Eve, but then lost to Carol. Carol entered the competition later (perhaps because of a higher level). Fred and Jim have yet to enter the competition. The corresponding database instance is:

Player

pid	name	level
1	Alice	100
2	Bob	110
3	Carol	300
4	Dave	200
5	Eve	220
6	Fred	400
7	Greg	220
8	Helga	300
9	Jim	400

Game

pidWin	pidLose
1	2
4	7
1	5
3	1
4	8

(a) Write the `CREATE TABLE` statements for the competition database. Use the correct types for the attributes and create the right constraints.

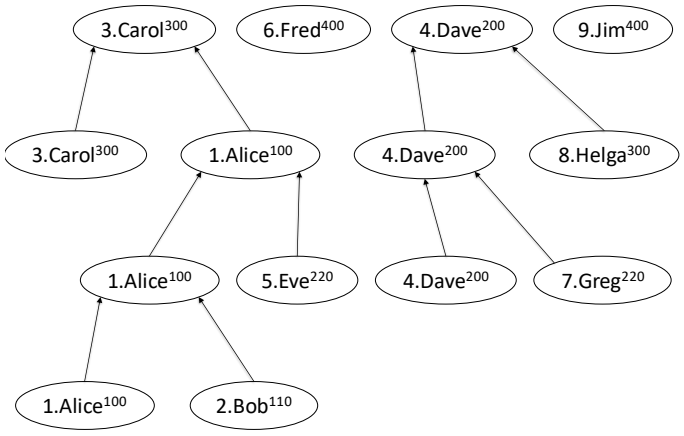
Write your answer here:

Solution:

```
drop table if exists Game;
drop table if exists Player;
create table Player(pid int primary key,name text, level int);
create table Game(pidWin int references Player,
                  pidLose int references Player);

insert into Player values
(1, 'Alice', 100),
(2, 'Bob', 110),
(3, 'Carol', 300),
(4, 'Dave', 200),
(5, 'Eve', 220),
(6, 'Fred', 400),
(7, 'Greg', 220),
(8, 'Helga', 300),
(9, 'Jim', 400);

insert into Game values
(1, 2),
(4, 7),
(1, 5),
(3, 1),
(4, 8);
```

(b) For each registered player compute the number of games that he/she won against a stronger adversary (meaning: a higher level). Your query should return the player ID, her name, and the number of games won.

For example, on our instance database your query should return the answer below:



Query answer:

pid	name	wins
1	Alice	2
2	Bob	0
3	Carol	0
4	Dave	2
5	Eve	0
6	Fred	0
7	Greg	0
8	Helga	0
9	Jim	0

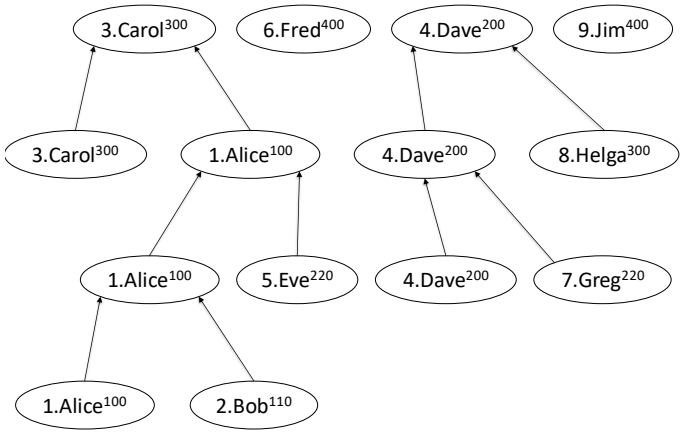
`Alice` has `level` 100 and she won against both `Bob` and `Eve`, who have higher levels 110 and 220 respectively. `Carol` won against `Alice`, but this doesn't count because `Carol`'s level is 300 and this is larger than `Alice`'s.

Write your answer here:

Solution:

```
select x.pid, x.name, count(y.pidWin)
from Player x left outer join (Game y join Player z on y.pidLose = z.pid)
     on x.pid = y.pidWin and x.level<z.level
group by x.pid, x.name
order by x.pid, x.name;
```

(c) Write a query that computes the set of registered players that have not yet been eliminated from the competition. For each such player return her ID, name, and level. In our example your query should return:



Query answer:

pid	name	level
3	Carol	300
4	Dave	300
6	Fred	400
9	Jim	400

Write your answer here:

Solution:

```
select x.pid, x.name, x.level
from Player x
where not exists
 (select *
  from Game y
  where x.pid = y.pidLose);
```

(d) At the beginning of the competition the organizers are trying to match players of similar levels. Write a SQL query that helps the organizers. Your query should return for each registered player the player or players whose score is closest to their score. You may use the SQL function `ABS`, e.g. `ABS`(-5) is 5. Your query should return the `pid`, `name`, and `level` of pairs of players `Player1`, `Player2`, where `Player2`'s level is closest to `Player1`. In our example, your answer would be the following:

Player:

pid	name	level
1	Alice	100
2	Bob	110
3	Carol	300
4	Dave	200
5	Eve	220
6	Fred	400
7	Greg	220
8	Helga	300
9	Jim	400

Query answer:

pid1	name1	level1	pid2	name2	level2
1	Alice	100	2	Bob	110
2	Bob	110	1	Alice	100
3	Carol	300	8	Helga	300
4	Dave	200	5	Eve	220
4	Dave	200	7	Greg	220
5	Eve	220	7	Greg	220
6	Fred	400	9	Jim	400
7	Greg	220	5	Eve	220
8	Helga	300	3	Carol	300
9	Jim	400	6	Fred	400

For `Eve`, the closest player is `Greg`, and vice versa, since both have level 220. For `Dave`, which has level 200, the closest players are both `Eve` and `Greg`.

Write your answer here:

Solution:

```
with Temp as
   (select x.pid, min(abs(x.level-y.level)) as m
    from Player x, Player y
    where x.pid != y.pid
    group by x.pid)
select u.pid, u.name, u.level, v.pid, v.name, v.level
from Player u, Player v, Temp w
where u.pid = w.pid
  and u.pid != v.pid
  and abs(u.level - v.level) = w.m;
```

42. *(from CSE 414, Spring 2024, Midterm; CSE 344, Autumn 2024, Midterm; CSE 344, Spring 2026, Midterm)*

We keep a database of ice cream customers, the flavors they like, and the stores where those flavors are available:

```
Cust(Cid,Name,Age)
Likes(Cid,Flavor)
Sells(Store,Flavor,Price)
```

`Cust` stores the customers: they have an ID, a name, and an age.

`Likes` stores the flavors that the customers like: `Cid` is a foreign key to `Cust`.

`Sells` stores what flavors are sold by which ice cream stores, and at what price.

In some of the questions below we will illustrate with the following test database instance. Please ensure that your answers are correct on any database instance, not just this one.

`Cust`

Cid	Name	Age
123	Alice	5
234	Bob	45
345	Carol	6
456	David	15
567	Eve	66
678	Frank	33

`Likes`

Cid	Flavor
123	Vanilla
123	Mint
345	Mint
456	Pecan
456	Maple
456	Pear
678	Pear

`Sells`

Store	Flavor	Price
Shug	Pecan	4
Shug	Maple	2
Molly	Maple	3
Molly	Choko	7
Molly	Mint	5
Frankie	Choko	5
Frankie	Mint	4

(a) Write a SQL query that computes for each customer the number of stores that sell some flavor that they like. Your query should return a table with attributes `Cid`, `Name`, `C`, where the last attribute represents the number of stores. On our example database your query should return as follows:

`Cust`

Cid	Name	Age
123	Alice	5
234	Bob	45
345	Carol	6
456	David	15
567	Eve	66
678	Frank	33

`Likes`

Cid	Flavor
123	Vanilla
123	Mint
345	Mint
456	Pecan
456	Maple
456	Pear
678	Pear

`Sells`

Store	Flavor	Price
Shug	Pecan	4
Shug	Maple	2
Molly	Maple	3
Molly	Choko	7
Molly	Mint	5
Frankie	Choko	5
Frankie	Mint	4

$\Longrightarrow$

Cid	Name	C
123	Alice	2
345	Carol	2
456	David	2

For example, consider `456, David`: he likes `Pecan, Maple, Pear`, which are sold by `Shug` and `Molly`, hence the answer is 2. You do not need to return customers who don't like any flavors sold by our stores, like `Bob`, `Eve` or `Frank`.

Write your answer here:

Solution:

Solution:

```
DROP TABLE IF EXISTS Likes;
DROP TABLE IF EXISTS Sells;
DROP TABLE IF EXISTS Cust;

CREATE TABLE Cust(cid int primary key, name text, age int);
CREATE TABLE Likes(cid int references Cust, flavor text);
CREATE TABLE Sells(store text, flavor text, price int);

INSERT INTO Cust VALUES
  (123, 'Alice', 5),
  (234, 'Bob', 45),
  (345, 'Carol', 6),
  (456, 'David', 15),
  (567, 'Eve', 66),
  (678, 'Frank', 33);

Insert into Likes values
  (123, 'Vanilla'),
  (123, 'Mint'),
  (345, 'Mint'),
  (456, 'Pecan'),
  (456, 'Maple'),
  (456, 'Pear'),
  (678, 'Pear');

Insert into sells values
  ('Shug', 'Pecan', 4),
  ('Shug', 'Maple', 2),
  ('Molly', 'Maple', 3),
  ('Molly', 'Choko', 7),
  ('Molly', 'Mint', 5),
  ('Frankie', 'Choko', 5),
  ('Frankie', 'Mint', 4);

SELECT x.Cid, x.Name, count(distinct z.Store) as c
FROM Cust x, Likes y, Sells z
WHERE x.Cid = y.Cid and y.Flavor = z.Flavor
GROUP BY x.Cid, x.Name;
```

(b) Write a query that returns for each store the cheapest flavor sold at that store. Your query should return a table with attributes `Store`, `Flavor`, `Price`. In our example:

Sells

Store	Flavor	Price
Shug	Pecan	4
Shug	Maple	2
Molly	Maple	3
Molly	Choko	7
Molly	Mint	5
Frankie	Choko	5
Frankie	Mint	4

⇒

Store	Flavor	Price
Shug	Maple	2
Molly	Maple	3
Frankie	Mint	4

Write your answer here:

Solution: Solution 1:

```
with Temp as
   (select Store, min(Price) as P
    from Sells
    group by Store)
select x.Store, x.Flavor, x.price
```

```
from Sells x, Temp y
where x.Store = y.Store and x.Price = y.P;
```

Solution 2:

```
select x.Store, x.Flavor, x.price
from Sells x, Sells y
where x.Store = y.Store
group by x.Store, x.Flavor, x.Price
having x.price = min(y.price);
```

Solution 3:

```
select x.Store, x.Flavor, x.Price
from Sells x
where x.Price <=
     ALL (select y.price
          from Sells y
          where x.Store=y.Store);
```

(c) A customer is happy if there exists a store that sells some of the flavors that they like. Write a SQL query to return all the unhappy customers. Your query should return a table with attributes `Name`, `Age`. In our running example:

Cust

Cid	Name	Age
123	Alice	5
234	Bob	45
345	Carol	6
456	David	15
567	Eve	66
678	Frank	33

Likes

Cid	Flavor
123	Vanilla
123	Mint
345	Mint
456	Pecan
456	Maple
456	Pear
678	Pear

Sells

Store	Flavor	Price
Shug	Pecan	4
Shug	Maple	2
Molly	Maple	3
Molly	Choko	7
Molly	Mint	5
Frankie	Choko	5
Frankie	Mint	4

⇒

Name	Age
Bob	45
Eve	66
Frank	33

For example `Alice` is happy, because she likes `Vanilla` and `Mint`; `Vanilla` is not sold anywhere, but `Mint` is sold at `Molly` and this makes `Alice` happy. But `Bob` and `Eve` are unhappy because they don't like anything, while `Frank` is unhappy, because he likes `Pear`, which is not sold anywhere.

Write your answer here:

Solution: Solution 1:

```
select x.Name, x.Age
from Cust x
where not exists
   (select *
    from Likes y, Sells z
    where x.Cid = y.Cid and y.Flavor = z.Flavor);
```

Solution 2:

```
with temp as
 (select x.cid, x.name, x.age, count(z.store) as c
  from Cust x left outer join Likes y on x.cid=y.cid
       left outer join Sells z on y.flavor=z.flavor
  group by x.cid, x.name)
select name, age
from temp
where c=0;
```

(d) A company A dominates a company B if every ice cream flavor sold by B is also

sold by A at a strictly lower price. Write a SQL query that returns all pairs of companies A,B where A dominates B. For example, if `Store` were the table on the left, then the answer should be the table on the right:

`Sells`

Store	Flavor	Price
Shug	Pecan	4
Shug	Mint	2
Molly	Pecan	3
Molly	Mint	1
Molly	Choko	4
Bee	Choko	5
Abe	Choko	3

$\Rightarrow$

CompanyA	CompanyB
Molly	Shug
Molly	Bee
Abe	Bee

Shug sells Pecan and Mint, and each of these flavors is also sold by Molly at a strictly cheaper price. Therefore Molly dominates Shug. On the other hand Molly does not dominate Abe because, although it does sell the sole ice cream flavor sold by Abe (Choko), it does not sell it at a strictly lower price.

Write your answer here:

Solution:

```
DROP TABLE IF EXISTS Sells;
CREATE TABLE Sells(store text, flavor text, price int);

INSERT INTO Sells VALUES
    ('Shug',  'Pecan',  4),
    ('Shug',  'Mint',  2),
    ('Molly',  'Pecan',  3),
    ('Molly',  'Mint',  1),
    ('Molly',  'Choko',  4),
    ('Bee',  'Choko',  5),
    ('Abe',  'Choko', 3);

select distinct a.store, b.store
from sells a, sells b
where not exists
        (select * from sells b1
         where b.store = b1.store
           and not exists
                (select *
                 from sells a1
                 where a.store = a1.store
                  and a1.flavor = b1.flavor and a1.price < b1.price));
```

(e) Consider the following table of companies:

`Company`

Cid	Name	Employees	Revenue	State
100	Acme	30	600000	WA
200	Bubble	NULL	200000	TX
300	Corp	50	NULL	OR
400	Duty	NULL	300000	MN
500	Easy	NULL	NULL	FL
600	Fit	20	800000	NULL

Solution:

```
drop table if exists company;

create table company
  (cid int, name text, employees int, revenue int, state text);

insert into company values
    (100, 'Acme',    30, 600000, 'WA'),
    (200, 'Bubble',  NULL, 200000, 'TX'),
    (300, 'Corp',    50,  NULL , 'OR'),
    (400, 'Duty',  NULL, 300000, 'MN'),
    (500, 'Easy',  NULL,  NULL , 'FL'),
    (600, 'Fit',    20, 800000, NULL);
```

For each query below indicate what it returns. For example if the query is:

```
SELECT Cid FROM Company
WHERE Name != 'Bubble';
```

then you should answer: 100,300,400,500,600

i.

```
SELECT Cid FROM Company
WHERE State != 'TX';
```

Solution: 100,300,400,500

ii.

```
SELECT Cid FROM Company
WHERE Employees > 25 and Revenue < 750000;
```

Solution: 100

iii.

```
SELECT Cid FROM Company
WHERE Employees > 25 or Revenue < 750000;
```

Solution: 100,200,300,400

iv.

```
SELECT Cid FROM Company
WHERE not(Employees > 25) or not(Revenue < 750000);
```

Solution: 600

v.

```
SELECT Cid FROM Company
WHERE not(not(Employees > 25) or not(Revenue < 750000));
```

Solution: 100

3 SQL: Deeper Query Understanding (43–48)

The questions here are about SQL rather than in SQL. Instead of (or in addition to) asking you to write a query, they may give you two queries and ask whether they always return the same answer, or they give a query and ask what it computes, or they ask you whether one result is guaranteed to be a subset of another.

Answering these questions requires reasoning about the semantics rather than guessing from examples, though a counterexample is the fastest way to prove that two queries *differ*: a single well-chosen instance, without duplicates and without `NULL`s, settles the question. Proving that two queries *agree* takes an argument covering every instance.

These problems come mainly from (the old version of) CSE 444 and CSE 544 and are noticeably harder than the query-writing exercises. They repay careful attention, because the equivalences at stake here are exactly the ones a query optimizer relies on later in the book.

43. *(from CSE 444, Autumn 2003, Final; CSE 544, 2004, Final)*

An insurance company maintains a database with the following schema:

```
Property(name, city)
Policy(pid, premium, propertyName, agent)
Claim(cid, pid, amount, year)
```

Here `premium` refers to the annual premium (what the client pays annually to the insurance company) and is the same every year; `amount` is a one-time payment (which the insurance company has to pay to the client), if and when the client submits a claim for that policy.

(a) An insurance agent processes the claim with `cid=03492` on behalf of his client. The claim is already in the `Claim` table. The agent would like to find out if there were other claims for the same property submitted during the same year. Write a SQL query to find this out: your query should return all `cid`'s of those other claims.

Solution:

```
select y.cid
from claim x, claim y, policy u, policy v
where x.cid = 03492 and x.pid = u.pid and y.pid = v.pid
  and u.propertyName = v.propertyName
  and x.year = y.year
```

(b) Write a SQL query that returns for each policy the years when the total amount on all claims in that year exceeded the annual premium for that policy. Your query should return a set of `pid`, `year` pairs.

Solution: The following does not quite work because `p.premium` is not a `group by` attribute:

```
select c.pid, c.year
from   claim c, policy p
where  c.pid = p.pid
group by c.pid, c.year
having sum(c.amount) > p.premium
```

This is ugly, but works:

```
select c.pid, c.year
from   claim c, policy p
where  c.pid = p.pid
group by c.pid, c.year, p.pid, p.premium
having sum(c.amount) > p.premium
```

Another way (one may take `min` here too):

```
select c.pid, c.year
from   claim c, policy p
where  c.pid = p.pid
group by c.pid, c.year
having sum(c.amount) > max(p.premium)
```

(c) Consider the pairs of queries `Q1`, `Q2` below. For each pair, indicate whether (A) the two queries return exactly the same answers, or (B) all the values returned by `Q1` are also returned by `Q2`, but `Q2` may return values not returned by `Q1`, or (C) all the values returned by `Q2` are also returned by `Q1`, but `Q1` may return values not returned by `Q2`, or (D) none of the above.

i.

```
Q1 = Select Distinct P.pid
     From Policy P, Claim C
     Where P.pid = C.pid and C.amount = 300
Q2 = Select Distinct P.pid
     From Policy P, Claim C1, Claim C2
     Where P.pid = C1.pid and P.pid = C2.pid and
           C1.amount > 100 and C2.amount < 2000
```

Solution: (B)

ii.

```
Q1 = Select Distinct A.name
     From Property A, Policy P, Claim C
     Where A.name=P.propertyName and
           P.pid=C.pid and C.amount > 5000
Q2 = Select Distinct A.name
     From Property A, Policy P, Claim C1, Claim C2
     Where A.name=P.propertyName and
           P.pid=C1.pid and C1.amount=10000 and
           P.pid=C2.pid and C2.amount < 300
```

Solution: (C). `Q2` returns properties A with a claim of \$10000; that claim is enough for `Q1` to return A too.

iii.

```
Q1 = Select Distinct A.name
     From Property A, Policy P, Claim C1, Claim C2
     Where A.name=P.propertyName and
           P.pid=C1.pid and C1.year=1980 and
           P.pid=C2.pid and C2.year=2002
Q2 = Select Distinct A.name
     From Property A, Policy P1, Policy P2, Claim C1, Claim C2
     Where A.name=P1.propertyName and
           P1.pid=C1.pid and C1.year=1980 and
           A.name=P2.propertyName and
           P2.pid=C2.pid and C2.year=2002
```

Solution: (B). If a property A is returned by `Q1`, then there is a policy P with two claims, one in 1980 the other in 2002; `Q2` will also return that property, by taking both P_1 and P_2 to be P. But `Q2` may return a property

having two *distinct* policies P_1, P_2, the first with a claim in 1980 and the other with a claim in 2002, and that property will not be returned by `Q1`.

iv.

```
Q1 = Select Distinct P2.pid
     From Property A1, Policy P1, Policy P2
     Where P1.agent='John Smith' and
           A1.city = 'Seattle' and
           A1.name=P1.propertyName and
           A1.name=P2.propertyName
Q2 = Select Distinct P2.pid
     From Property A1, Property A2, Policy P1, Policy P2, Policy P3
     Where P1.agent='John Smith' and
           A1.city = 'Seattle' and
           A1.name=P1.propertyName and
           A1.name=P2.propertyName and
           P2.agent=P3.agent and
           A2.name=P3.propertyName
```

Solution: (A). The two queries return the same answers (it takes some thinking to see that).

44. *(from CSE 444, Autumn 2004, Final)*

Consider the following relational schema:

`Employee(`<u>`eid`</u>`, name, dept, boss, gender)`

The attribute `eid` is a key, and `boss` is a foreign key to `Employee` (may be null); `gender` is `F` or `M`, and we know that every department has at least one female employee.

(a) Consider the four queries below:

```
Q1:
   select dept
   from Employee
   where gender = 'F'
   group by dept
   having count(*) > 5

Q2:
   select distinct x.dept
   from Employee x
   where (select count(*)
          from Employee y
          where x.dept = y.dept and y.gender='F') > 5

Q3:
```

```
   select distinct x.dept
   from Employee x
   where x.gender = 'F' and
         (select count(*)
          from Employee y
          where x.dept = y.dept) > 5

Q4:
   select distinct x.dept
   from Employee x
   where x.gender = 'F' and
         (select count(*)
          from Employee y
          where x.dept = y.dept and y.gender='F') > 5
```

For each pair of queries `Q`, `Q'` indicate which of the following relationships holds:

Equal `Q` $=$ `Q'` if they return precisely the same answer.

Contained `Q` $\subset$ `Q'` if `Q'` returns all answers returned by `Q`, and they are not equal. To indicate `Q'` $\subset$ `Q`, write `Q` $\supset$ `Q'`.

Different `Q` $\neq$ `Q'` if none of the above holds.

In each of the six entries below you have to write one of $=$, $\subset$, $\supset$, $\neq$.

	Q1	Q2	Q3	Q4
Q1	$=$			
Q2		$=$		
Q3			$=$	
Q4				$=$

Solution: Queries Q1, Q2, Q4 return the departments that have more than 5 female employees: they are equal.

Query Q3 returns departments that have a at least one female employee, and more than 5 employees in total. This set will always include all departments returned by Q1 or Q2 or Q4.

	Q1	Q2	Q3	Q4
Q1	$=$	$=$	$\subseteq$	$=$
Q2	$=$	$=$	$\subseteq$	$=$
Q3	$\supseteq$	$\supseteq$	$=$	$\supseteq$
Q4	$=$	$=$	$\subseteq$	$=$

(b) Consider the following four queries:

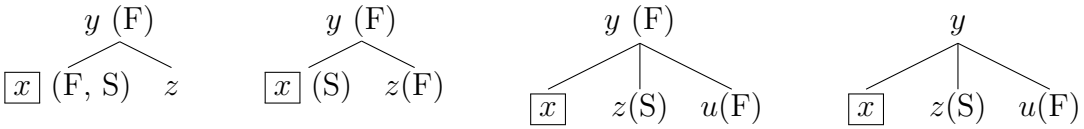
```
Q1: select distinct x.name
    from Employee x, Employee y, Employee z
    where x.boss = y.eid and z.boss = y.eid
      and x.dept = 'Sales' and x.gender = 'F' and y.gender = 'F'

Q2: select distinct x.name
```

```
    from Employee x, Employee y, Employee z
    where x.boss = y.eid and z.boss = y.eid
      and x.dept = 'Sales' and z.gender = 'F' and y.gender = 'F'

Q3: select distinct x.name
    from Employee x, Employee y, Employee z, Employee u
    where x.boss = y.eid and z.boss = y.eid and u.boss = y.eid
      and z.dept = 'Sales' and y.gender = 'F' and u.gender = 'F'

Q4: select distinct x.name
    from Employee x, Employee y, Employee z, Employee u
    where x.boss = y.eid and z.boss = y.eid and u.boss = y.eid
      and z.dept = 'Sales' and u.gender = 'F'
```

As for the previous question, fill in the six entries below with one of $=$, $\subset$, $\supset$, $\neq$.

	Q1	Q2	Q3	Q4
Q1	$=$			
Q2		$=$		
Q3			$=$	
Q4				$=$

Solution: To understand the four queries it helps to visualize them as small graphs:



Each query has 3 or 4 variables x, y, z, u, and each node points to its boss, for example the line from x to y corresponds to x.`boss` = y.`eid`. The letters in parentheses indicate which predicates we applied to the variable: S stands for `dept` = 'Sale' and F stands `dept` = 'Sale'.

Q1 returns all female employees x in the sales department whose boss is also a female y. The variable z is redundant, because it can be mapped to x: more precisely, if you think about the *for-each* semantics, then the loop for z will immediately find a match, namely the same record as x, since the condition on z (namely that his/her boss must be y) are already satisfied by x.

Q2 returns all employees x in the sales department (any gender) whose boss is also a female y, and y is also the boss of some female employee z. Here z is no longer redundant. If x happens to be a female then z can be mapped to x, but if x is a male employee, then Q2 may still return it if it finds any female employee z with the same boss y. Thus, Q2 returns a superset of what Q1 returns.

Q3 returns all employees x (any department, any gender) who have a boss who is also a female y, and y is also the boss of some employee y in the sales department (any gender) and also the boss of some female employee u (any department). Thus, Q3 returns a superset of what Q2 returns.

Q4 is like Q3 but doesn't require y to be a female. Thus, Q4 returns a superset of what Q3 returns.

The answer is:

	Q1	Q2	Q3	Q4
Q1	$=$	$\subseteq$	$\subseteq$	$\subseteq$
Q2	$\supseteq$	$=$	$\subseteq$	$\subseteq$
Q3	$\supseteq$	$\supseteq$	$=$	$\subseteq$
Q4	$\supseteq$	$\supseteq$	$\supseteq$	$=$

(c) The table `Employee` is altered to record for every employee his/her boss's boss. The new table is:

`Employee(`<u>`eid`</u>`, name, dept, boss, gender, bossesBoss)`

Write a SQL statement that updates the new field `bossesBoss` in all records in the table.

Solution:

```
-- first let's see how the Employee table was created:
drop table if exists employee;
create table Employee(eid int primary key,
                      name text,
                      dept text,
                      boss int references employee,
                      gender text);
insert into employee values
  (1,'alice',10,null,'f'),
  (2,'bob',20,1,'m'),
  (3,'carol',30,1,'f'),
  (4,'david',40,2,'m');

-- now alter the table
alter table employee add column bossesboss int references employee;
update employee u set bossesboss =
    (select x.boss from employee x where x.eid=u.boss);

-- check
select * from employee;
```

45. *(from CSE 444, Winter 2005, Final)*

Consider the following relational schema:

```
Person(name, age, city)
Trusts(name1, name2)
```

The `Trusts` relation tells us who trusts whom. This relation is not necessarily transitive: that is, if 'John' trusts 'Mary' and 'Mary' trusts 'Fred', it is not necessarily the case that 'John' trusts 'Fred'.

(a) For each pair of queries `Q1`, `Q2` below, indicate whether `Q1` is *contained* in `Q2`, i.e. if every answer returned by `Q1` is also returned by `Q2`.

i.

```
Q1: select distinct x.name1
    from Trusts x, Trusts y
    where x.name2 = y.name1 and y.name2='Betty'

Q2: select distinct x.name1
    from Trusts x
    where x.name2='Betty'
```

Solution: No: there is no connection between Q1 and Q2.
Q1 returns people who trust someone who trusts Betty.
Q2 returns people who trust Betty.
There is no relationships.
For example, suppose the relation Trusts is:

name1	name2
Alice	Bob
Bob	Betty

Then Q1 returns Alice, Q2 returns Bob, and there is no connection between the two sets.

ii.

```
Q1: select distinct x.name1
    from Trusts x
    where x.name2 = 'Fred'

Q2: select distinct x.name1
    from Trusts x, Trusts y, Trusts z
    where x.name2 = y.name2 and
          y.name1 = z.name1 and
          z.name2 = 'Fred'
```

Solution: $Q1 \subseteq Q2$.

To see this observe that, if Q1 finds a tuple x satisfying the condition $x.\texttt{name2}$ = 'Fred', then Q2 can set y and z to the same tuple and all conditions required by Q2 are still satisfies, because:

- $x.\texttt{name2} = y.\texttt{name2}$ holds because we have set $y = x$,
- $y.\texttt{name1} = z.\texttt{name1}$ holds because $y = z$ hence they must have the same `name1`,
- and finally $z.\texttt{name2}$ = 'Fred' because $z = x$.

On the other hand $Q2 \not\subseteq Q1$. For a simple example consider the following relation Trusts:

name1	name2
Alice	Bob
Carol	Bob
Carol	Fred

Then Q1 returns only Carol, while Q2 returns Alice and Carol.

iii.

```
Q1: select x.name, count(*)
    from Person x, Trusts y
    where x.city = 'Seattle' and
          x.age > 25 and
          x.name = y.name1
    group by x.name

Q2: select x.name, count(*)
    from Person x, Trusts y
    where x.city = 'Seattle' and x.name = y.name1
    group by x.name
```

Solution: $Q1 \subseteq Q2$ because Q2 simply dropped the condition $x.\texttt{age} > 25$.

iv.

```
Q1: select x.city, count(*)
    from Person x, Trusts y
    where x.city = 'Seattle' and
          x.age > 25 and
          x.name = y.name1
    group by x.city

Q2: select x.city, count(*)
    from Person x, Trusts y
    where x.city = 'Seattle' and x.name = y.name1
    group by x.city
```

Solution: Same as the previous question: $Q1 \subseteq Q2$ because Q2 simply dropped the condition $x.\texttt{age} > 25$.

46. *(from CSE 444, Winter 2005, Final; CSE 444, Autumn 2005, Final)*

Consider the following relational schema:

```
Trusts(name1, name2)
```

The `Trusts` relation tells us who trusts whom. This relation is not necessarily symmetric: i.e. if 'John' trusts 'Mary', then it is not necessarily the case that 'Mary' trusts 'John'. This relation is not necessarily transitive: i.e. if 'John' trusts 'Mary' and if 'Mary' trusts 'Fred', it is not necessarily the case that 'John' trusts 'Fred'. However, we will assume that the relation is reflexive: every person trusts themselves.

Consider the following types of people:

- A "loner" is a person who trusts no one but himself.
- A "loyal" is a person who trusts only those who trust him.
- A "ruler" is a person who trusts only those who trust only him.

Indicate for each of the queries below whether it computes the loners, or the loyals, or the rulers, or none of the above.

(a)

```
select distinct x.name1
from Trusts x
where x.name1 not in
               (select y.name1
                from Trusts y
                where y.name2 in
                             (select z.name1
                              from Trusts z
                              where z.name2 != y.name1))
```

What does this query computes?

Solution: The query returns the rulers.

For all these problems it is most helpful to express the queries in First Order Logic, also called *Relational Calculus*. Let's abbreviate $T(a, b)$ when a person with name a trusts a person with name b. Then the SQL query above computes the following:

$$Q(a) = \exists b\,(T(a,b) \wedge \neg\exists c\,(T(a,c) \wedge \exists d\,(T(c,d) \wedge d \neq a)))$$

The first predicate $T(a, b)$ represents the tuple x: we return a if the next condition is met.

The next condition is a negation, corresponding to `not in`. The predicate $T(a, c)$ represents the tuple y. Notice that the SQL query requires x.`name1` = y.`name1` (this implicit in the condition x.`name1 not in select` y.`name1`...) and for that reason we put a in the first position of $T(a, c)$.

Finally the last predicate $T(c, d)$ represents the tuple z, and the condition z.`name2` $\neq$ y.`name1` became $d \neq a$.

Now we use equivalence of logical formulas to transform Q into an expressions with universal quantifiers instead of negations using de Morgan's law: $\neg\exists(\cdots) = \forall\neg(\cdots)$:

$$\begin{aligned} Q(a) =& \exists b\,(T(a,b) \wedge \neg\exists c \exists d\,(T(a,c) \wedge T(c,d) \wedge d \neq a)) \\ Q(a) =& \exists b\,(T(a,b) \wedge \forall c \forall d \neg\,(T(a,c) \wedge T(c,d) \wedge d \neq a)) \\ Q(a) =& \exists b\,(T(a,b) \wedge \forall c \forall d\,(T(a,c) \wedge T(c,d) \Rightarrow d = a)) \end{aligned}$$

(In the last line we used the fact that $\neg(P \wedge \neg Q) = (P \Rightarrow Q)$). The last line is now easy to read in English: the query returns a if a trusts someone (namely b) and, moreover, everyone whom a trusts (namely c) trusts only a.

Thus, a is a ruler.

(b)

```
select distinct x.name1
from Trusts x
where x.name1 not in
                (select y.name1
                 from Trusts y
                 where y.name1 != y.name2)
```

What does this query computes?

Solution: This query computes the loners.

We apply the same method as before and translate the query into relational calculus:

$$\begin{aligned} Q(a) =& \exists b\,(T(a,b) \wedge \neg\exists c\,(T(a,c) \wedge a \neq c)) \\ Q(a) =& \exists b\,(T(a,b) \wedge \forall c \neg\,(T(a,c) \wedge a \neq c)) \\ Q(a) =& \exists b\,(T(a,b) \wedge \forall c\,(T(a,c) \Rightarrow a = c)) \end{aligned}$$

The last line reads as follows in English: a is return if everyone they trust (denoted c) is a itself. Same as saying that a only trusts himself/herself.

(c)

```
select distinct x.name1
from Trusts x
where x.name1 not in
                (select y.name1
                 from Trusts y
                 where y.name2 not in
                             (select z.name1
                              from Trusts z
                              where z.name2 = y.name1))
```

What does this query computes?

Solution: This query returns the loyals.
To see this, convert it into relational calculus:

$$\begin{aligned} Q(a) =& \exists b T(a,b) \wedge \neg \left(\exists c \left(T(a,c) \wedge \neg T(c,a)\right)\right) \\ Q(a) =& \exists b T(a,b) \wedge \forall c \neg \left(T(a,c) \wedge \neg T(c,a)\right) \\ Q(a) =& \exists b T(a,b) \wedge \forall c \left(T(a,c) \Rightarrow T(c,a)\right) \end{aligned}$$

In English: the query returns persons a such that every person c that they trust ($T(a,c)$) trusts them back ($T(c,a)$).

47. *(from CSE 544, 2006, Final)*

An insurance company maintains a database with the following schema:

```
Property(property, city)
Policy(pid, premium, property, agent)
Claim(cid, pid, amount, year)
```

`Property.property` is a key, `Policy.property` is a foreign key to `Property.property`, and `Claim.pid` is a foreign key to `Property.property`. Here `premium` refers to the annual premium (what the client pays annually to the insurance company) and is the same every year; `amount` represents a one time payment made by the insurance company to the client as a result of a claim; there can be zero or more claims during one year for the same policy; there can also be zero or more policies for the same property.

(a) For each `property` and for each `year` compute the total number of claims submitted for that property during that year. You have to write a SQL query that returns a set of triples of the form `property`, `year`, `number`, where `number` > 0.

Solution:

```
select x.property, z.year, count(*) as number
from Property x, Policy y, Claim z
where x.property=y.property and y.pid=z.pid
group by x.property, z.year;
```

(b) Find all policies for which the total amount paid by the insurance company during some year exceeds the annual premium on that property. You have to write a SQL query that returns a of pairs `pid`, `year`s.

Solution:

```
select distinct x.property, z.year
from Property x, Policy y, Claim z
where x.property=y.property and y.pid=z.pid
group by x.property, z.year, x.premium
having sum(z.amount) > x.premium
```

(c) Consider the pairs of queries `Q1`, `Q2` below. For each pair, indicate whether (A) the two queries are equivalent, $Q1 \equiv Q2$; or (B) `Q1` is contained in `Q2`, but `Q2` is not contained in `Q1`, $Q1 \subset Q2$; or (C) `Q2` is contained in `Q1`, but `Q1` is not contained in `Q2`, $Q1 \supset Q2$; or (D) none of the above. You only have to choose an answer, and are not required to provide any proof. For example, if

```
Q1 = Select distinct year        Q2 = Select distinct year
     From Claim                       From Claim
     Where amount > 100               Where amount > 50
```

then you would answer (B): $Q1 \subset Q2$.

i.

```
Q1 = Select Distinct P.pid
     From Policy P, Claim C
     Where P.pid = C.pid and C.amount = 300

Q2 = Select Distinct P.pid
     From Policy P, Claim C1, Claim C2
     Where P.pid = C1.pid and P.pid = C2.pid and
           C1.amount > 100 and C2.amount < 2000
```

Solution: $Q1 \subset Q2$ because if `C.amount` $= 300$ then, by setting $C1 := C$ and $C2 := C$ both conditions `C1.amount` > 100 and `C2.amount` < 2000 hold.

ii.

```
Q1 = Select Distinct A.name
     From Property A, Policy P, Claim C
     Where A.property=P.property and
           P.pid=C.pid and C.amount > 5000
```

```
Q2 = Select Distinct A.name
     From Property A, Policy P, Claim C1, Claim C2
     Where A.property=P.property and
           P.pid=C1.pid and C1.amount=10000 and
           P.pid=C2.pid and C2.amount < 300
```

Solution: No relationships.

iii.

```
Q1 = Select Distinct A.name
     From Property A, Policy P, Claim C1, Claim C2
     Where A.property=P.property and
           P.pid=C1.pid and C1.year=1980 and
           P.pid=C2.pid and C2.year=2002

Q2 = Select Distinct A.name
     From Property A, Policy P1, Policy P2, Claim C1, Claim C2
     Where A.property=P1.property and
           P1.pid=C1.pid and C1.year=1980 and
           A.property=P2.property and
           P2.pid=C2.pid and C2.year=2002
```

Solution: $Q1 \subset Q2$ because we can set the policies $P1, P2$ of the second query to be the same as the policy P of the first query, and all conditions in $Q2$ are satisfied.

iv.

```
Q1 = Select Distinct P2.pid
     From Property A1, Policy P1, Policy P2
     Where P1.agent='John Smith' and
           A1.city = 'Seattle' and
           A1.property=P1.property and
           A1.property=P2.property

Q2 = Select Distinct P2.pid
     From Property A1, Property A2,
          Policy P1, Policy P2, Policy P3
     Where P1.agent='John Smith' and
           A1.city = 'Seattle' and
           A1.property=P1.property and
           A1.property=P2.property and
           P2.agent=P3.agent and
           A2.property=P3.property
```

Solution: $Q1 \equiv Q2$. This can be seen by observing that $A2$ and $P3$ are redundant in $Q2$. Once SQL finds tuples $A1, P1, P2$ that satisfy the conditions:

```
Where P1.agent='John Smith' and
      A1.city = 'Seattle' and
      A1.property=P1.property and
      A1.property=P2.property
```

then setting $A2 := A1$ and $P3 := P2$ the other two conditions are immediately satisfied:

```
      P2.agent=P3.agent and
      A2.property=P3.property
```

48. *(from CSE 444, Autumn 2008, Midterm)*

We have a database of documents. Each document consists of several sections, each section contains several words:

`Doc(docID, docTitle)`	— documents
`Section(docID, secNumber, secTitle)`	— sections
`WordOcc(docID, secNumber, word)`	— word occurrences

`Section(docID)` is a foreign key to `Doc(docID)`; `WordOcc(docID, secNumber)` is a foreign key to `Section(docID, secNumber)`. Each document has at least one section; each section has at least one word.

(a) Write a SQL query that computes for each document the total number of distinct words used in that document. The answer should consist of triples: document id, document title, word-count.

Solution:

```
select x.docid, x.title, count(distinct z.word)
from doc x, section y, wordocc z
where x.docid=y.docid and y.docid=z.docid
  and y.secnumber=z.secnumber
group by x.docid, x.title
```

Note that the join through `Section` is not needed: instead of `x.docid = y.docid and y.docid = z.docid` it suffices to write `x.docid = z.docid`.

(b) Write a SQL query that finds all documents containing both keywords `'midterm'` and `'solution'` in the same section. For each such document you should return its document id and title.

Solution:

```
select distinct x.docid, x.title
from doc x, section y, wordocc z1, wordocc z2
where x.docid = y.docid
  and y.docid = z1.docid and y.secnumber = z1.secnumber
  and y.docid = z2.docid and y.secnumber = z2.secnumber
  and z1.word = 'midterm' and z2.word = 'solution'
```

(c) Next, you are given a table `QW(word)` of keywords. Write a SQL query that finds all documents that contain all the keywords listed in `QW`. For each such document you should return its document id and title.

Solution:

```
select distinct x.docid, x.title
from doc x
where not exists ( select *
                   from QW w
                   where not exists (select *
                                     from section y, wordocc z
                                     where x.docid = y.docid
                                       and z.docid = y.docid
                                       and z.secnumber = y.secnumber
                                       and z.word = w.word))
```

(d) Let $\texttt{Answers}_b$ be the set of answers returned by the query you wrote in part b. Assume `QW` contains exactly two keywords, `'midterm'` and `'solution'`, and let $\texttt{Answers}_c$ be the set of answers returned by the query you wrote in part c. Circle all the statements below that are guaranteed to be true:

$$\texttt{Answers}_b \subseteq \texttt{Answers}_c \quad \texttt{Answers}_b = \texttt{Answers}_c \quad \texttt{Answers}_b \supseteq \texttt{Answers}_c$$

Solution:

$$\texttt{Answers}_b \subseteq \texttt{Answers}_c$$

Explanation: if a document contains both words `'midterm'` and `'solution'` in the same section, then obviously that document contains both words. The converse is not always true: some document may contain both words but in different sections, in which case it will be in $\texttt{Answers}_c$ but not in $\texttt{Answers}_b$.

4 The Relational Data Model (49–54)

A short section on the foundations: what a relation is, what a key is, and what the model can and cannot express. The questions ask you to identify keys and foreign keys in a given schema, to decide whether a constraint can be enforced declaratively, and to reason about set versus bag semantics.

Several are true-or-false questions that look easier than they are. The distinction between a key and a superkey, the difference between integrity constraints the system enforces and assumptions the application merely hopes for, and the meaning of `NULL` all appear repeatedly, and all three are common sources of confusion.

49. *(from CSE 544p, Spring 2009, Final)*

A large IT corporation stores their access control policy to their objects in a database with the following schema:

```
User(uid, uname)          /* all the users */
Group(gid, gname)         /* all the groups */
Member(uid, gid)          /* users belong to groups (many-many) */
Object(oid, oname)        /* all the objects */
AccessGranted(gid, oid)   /* access to an object is granted on a per group basis */
AccessDenied(uid, oid)    /* access denied per user; overrides AccessGranted */
AccessLog(uid, oid, t)    /* logs whenever a user accesses an objectq; t=timestamp */
```

These policies are enforced in the applications, which have to read the database to determine if a request to an object should be granted or denied, then proceed accordingly. Grant policies are group-based: that is, an entire group is granted access to an object. Deny policies are user based: a particular user may be denied access to an object, even if the user belongs to a group that has access to the object. For auditing purposes, every time an object is accessed by a user, the system logs an entry into the `AccessLog` table.

(a) Describe the SQL schema and the constraints for this data. You have to turn in seven CREATE TABLE statements that contain the primary key and the foreign key constraints. All user ids, group ids, object ids, and the time stamps are integers; all other attributes are `VARCHAR(n)`, where you can choose a suitable value for n.

Solution:

```
create table Users(uid int primary key, uname varchar(50));
create table Groups(gid int primary key, gname varchar(50));
create table Member(uid int references Users, gid int references Groups);
create table Object(oid int primary key, oname varchar(50));
create table AccessGranted(gid int references Groups, oid int references Object);
create table AccessDenied(uid int references Users, oid int references Object);
create table AccessLog(uid int references Users, oid int references Object, t int);
```

(b) Write a SQL query that computes the total number of distinct objects accessed by the system, in a sliding window of width 10. That is, your query should report pairs `(t, count)` where `count` represents the number of distinct objects accessed during the timestamps `t...t+9`.

Solution:

```
select x.t, count(distinct y.oid)
from AccessLog x, AccessLog y
where x.t <= y.t and y.t <= x.t + 9
group by x.t
```

(c) Write a SQL query that checks if any of the accesses recorded in AccessLog were illegal. Your query should return all illegal accesses, i.e. accesses where the user

uid was not permitted to access the object `oid`.

Solution:

```
select x.*
from AccessLog x
where exists(select * from AccessDenied y where x.uid=y.uid and x.oid=y.oid)
   or not exists (select * from Member z, AccessGranted v
                  where x.uid=z.uid and z.gid=v.gid and v.oid=x.oid)
```

(d) A group is called *useless* if all the access grant permissions given through that group are overridden by a access deny entry. That is, the group is useless if for every member U of that group and for every object O to whom that group is being granted access, there is an entry (U,O) in `AccessDenied`: it means that the group is not useful in granting any access at all, since all the accesses are overridden in the `AccessDenied` table. Write a SQL query to find all useless groups.

Solution:

```
select *
from Group x
where not exists
       (select *
        from Member y, AccessGranted z
        where x.gid = y.gid and x.gid = z.gid
          and not exists (select *
                          from AccessDenied u
                          where y.uid=u.uid and z.oid=u.oid))
```

50. *(from CSE 344, Winter 2016, Midterm)*

Answer each question below.

(a) In the relational data model an attribute of a relation cannot be another relation. True or false?

Solution: True

(b) Consider the relation `Product(productName, manufacturer, price, weight)` where the key is `(productName, manufacturer)`. Can there be two different products with the same value of `manufacturer`? Yes or no?

Solution: Yes

(c) If an attribute in a table is a foreign key, then the table cannot contain two tuples with the same value of that attribute. True or false?

Solution: False

(d) In this and the following question we will assume that the relational algebra has set semantics. If the relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples, what is the largest possible size of the output of the natural join $R \bowtie S$? Choose one of the following:

(a) m
(b) n
(c) mn
(d) $m + n$
(e) $\max(m, n)$
(f) $(m + n)/2$
(g) None of the above.

Answer a-g:

Solution: c

(e) If the relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples, what is the largest possible size of the output of $\Pi_{AB}(R \bowtie S)$? Here $\bowtie$ represents the natural join. Choose from the same options as in the previous question. Answer a-g:

Solution: a

(f) Consider two relations $R(A, B), S(C, D)$, where all attributes are integers and cannot be NULL. For each identity below, indicate whether it is true or false.

i. $R \bowtie_{B=C} S = S \bowtie_{B=C} R$ True or false?

Solution: True

ii. $(\sigma_{B<1000}(R)) \bowtie_{B \neq C} S = (\sigma_{B<1000}(R)) \bowtie_{B \neq C} (\sigma_{C \geq 1000}(S))$ True or false?

Solution: False

iii. $R \times S - R \bowtie_{B=C} S = R \bowtie_{B \neq C} S$ True or false?

Solution: True

iv. $R - \Pi_{AB}(R \bowtie_{B=C} S) = \Pi_{AB}(R \bowtie_{B \neq C} S)$ True or false?

Solution: False

v. $\gamma_{A,\text{count}(*)\to F}(R \bowtie_{B=C} S) = \gamma_{A,\text{sum(E)}\to F}(R \bowtie_{B=C} (\gamma_{C,\texttt{count(*)}\to E}(S)))$ True or false?

Solution: True

(g) If a relation is stored in 1000 blocks on disc, then it cannot have more than 1000 records. True or false?

Solution: False

(h) Alice is a database administrator. She wants to add indexes to a table that is accessed frequently by her applications. She knows that a clustered index delivers better performance than an unclustered index, but most indexes she creates are unclustered. Why? Choose one of the following:

(a) Because clustered indexes take more space than unclustered indexes and Alice's database servers has little space.

(b) Because a relation can have only one clustered index, but many unclustered indexes.

(c) Because clustered indexes will slow down updates and Alice's application has many updates.

(d) None of the above.

Choose one of a-d:

Solution: b

(i) There is an important distinction between logical operators and physical operators. Indicate which of the operators below are physical operators:

(a) Theta join

(b) Hash join

(c) Natural join

(d) Eq join

(e) Index-based join

(f) Merge join

Answer a-f (choose all that apply):

Solution: b,e,f

(j) Did the speed of sequential reads from disk increase significantly over the years? Yes or no?

Solution: Yes

(k) Did the speed of random reads from disk increase significantly over the years? Yes or no?

Solution: No

51. *(from CSE 414, Spring 2016, Midterm)*

Answer each question below.

(a) True or false? If an attribute of a relation is a foreign key, then all records in the relation must have distinct values for that attribute.
True or false?

Solution: False

(b) In SQL, a relation can have at most one key.
True or false?

Solution: True

(c) In SQL, a relation can have at most one foreign key.
True or false?

Solution: False

(d) The goal of a foreign key is to help the query optimizer evaluate queries more efficiently.
True or false?

Solution: False

(e) If the relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples, what is the largest possible size of the output of the natural join $R \bowtie S$? Choose one of the following:

(a) m
(b) n
(c) mn
(d) $m + n$
(e) $\max(m, n)$
(f) $(m + n)/2$
(g) None of the above.

Answer a-g:

Solution: c

(f) Consider two relations $R(A, B), S(C, D)$.

i. Is the size of $R \bowtie_{A=C} S$ always larger than or equal to the size of $R \bowtie_{A=C \text{ and } B=D} S$? Yes or no?

Solution: Yes

ii. Is the size of $R \bowtie_{A=C \text{ and } B \neq D} S$ always larger than or equal to the size of $R \bowtie_{A=C \text{ and } B=D} S$? Yes or no?

Solution: No

iii. Are these two expressions equivalent $\sigma_{A=5}(R \bowtie_{B=C} S) = (\sigma_{A=5}(R)) \bowtie_{B=C} S$? Yes or no?

Solution: Yes

iv. Are these two expressions equivalent $(R \times S) - (R \bowtie_{B=C} S) = R \bowtie_{B \neq C} S$ Yes or no?

Solution: Yes

(g) True or false? A "disk block" is a mechanical device that moves over the disk platter in order to read from or write to disk.

True or false?

Solution: False

(h) Assuming $B(R) = 1000$, $T(R) = 200,000$ and $V(R, A) = 500$, estimate the number of disk I/O's for an index-based selection for $\sigma_{A=55}(R)$ in each of the following cases:

i. The system uses an unclustered index on $R.A$. Number of I/O's:

Solution: 400

ii. The system uses a clustered index on $R.A$. Number of I/O's:

Solution: 2

(i) The main advantage of a clustered index over an unclustered index is that the clustered index uses less space.

True or false?

Solution: False

(j) The main reason why we cannot create too many indexes is because they will slow down updates to the database.

True or false?

Solution: True

(k) There is an important distinction between logical operators and physical operators. Indicate which of the operators below are physical operators:

(a) Theta join
(b) Hash join
(c) Natural join
(d) Eq join
(e) Index-based join
(f) Merge join

Answer a-f (choose all that apply):

Solution: b,e,f

(l) Did the speed of sequential reads from disk increase significantly over the years? Yes or no?

Solution: Yes

(m) Did the speed of random reads from disk increase significantly over the years? Yes or no?

Solution: No

52. *(from CSE 344, Autumn 2017, Final)*

A university maintains a database of students, the courses they took, and their summer internships, with the following schema:

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
Intern(sid, companyName)
```

The keys are underlined; `Takes.sid` and `Intern.sid` are foreign keys to `Student`, while `Takes.cid` is a foreign key to `Course`.

(a) If two students took the same course in the same year then we call them *pals*. Write a SQL query that computes, for each student, the number of his/her pals. Your query should return all students indicating their student ID and name, and the number of their pals, ordered lexicographically by their names. (Hint: students who never took any course have zero pals; you need to return these too. Students who took at least one course have at least one pal, naturally (why?); you should not exclude himself/herself when you count their number of pals.)

Solution:

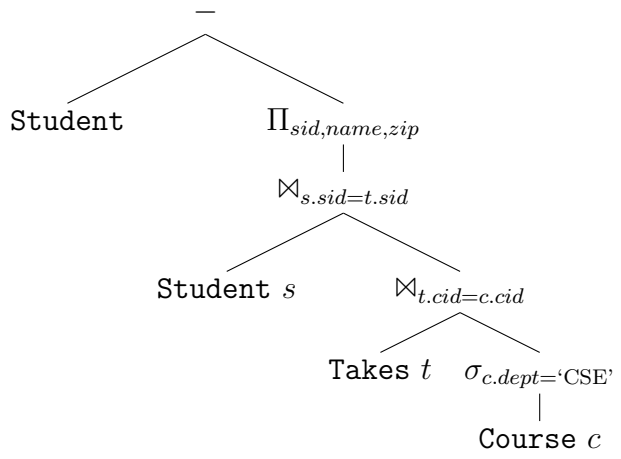
```
select s.sid, s.name, count(t2.sid)
from Student s
   left outer join Takes t on s.sid = t.sid
   left outer join Takes t2 on t.cid = t2.cid and t.year = t2.year
group by s.sid, s.name
order by s.name;
```

(b) Consider the following SQL query:

```
select *
from Student s
where not exists
      (select *
       from Takes t, Course c
       where s.sid = t.sid
         and t.cid = c.cid
         and c.dept = 'CSE');
```

i. Write this query in relational algebra.

Solution:



ii. Write this query in datalog.

Solution:

```
nonAnswer(sid) :- Takes(sid, cid, -), Course(cid, -, 'CSE')
Answer(sid,name,zip) :- Student(sid,name,zip), !nonAnswer(sid)
```

(c) We say that a student X has a *connection* to Y if either X, Y are pals, or they have a common connection. Alice wants to find a connection who was an intern at Microsoft. Write a datalog program to find all of Alice's connections who were Microsoft interns. In other words, if Alice is a pal with someone, who is a pal with someone, who is a pal ..., who is a pal with someone who interned at Microsoft, then you return the Microsoft intern; you should return the `sid` and `name`. Recall that *pal* means that the two students took a common course in a common year.

Solution:

```
ref(sid1,sid2) :- Takes(sid1,cid,y), Takes(sid2,cid,y)
ref(sid1,sid3) :- ref(sid1,sid2), ref(sid2,sid3)
answer(sid,name) :- Student(s,'Alice',-), ref(s,sid),
                    Intern(sid,'Microsoft')
```

53. *(from CSE 344, Winter 2019, Final)*

A Web browser stores the local data in a relational database[2] with the following schema:

```
History(url, ts)
Cache(url, content, size)
Bookmark(name, url)
```

- Every time the user visits a page, the browsers inserts a record in `History`; `url` is the URL of the Webpage and has type `Text`; `ts` is the time stamp when the user accesses the page and has type `Int`.
- Some Web pages are stored in the local cache of the browser; this is the table `Cache`; `content` is the HTML text of the page in the cache and has type `Text`, while `size` is its size in bytes and has type `Int`.
- The user may create bookmarks and give them unique names; all bookmarks are stored in `Bookmark`.

(a) Sometimes users created duplicate bookmarks; they give two or more names to the same URL. Write a SQL query that returns all duplicate bookmarks. Your query should return a list of names and URLs, sorted by the URLs.

Solution:

```
-- drop/create tables for testing only
drop table if exists history;
drop table if exists bookmarks;
drop table if exists cache;
create table history(url text, ts int);
create table cache(url text primary key, content text, size int);
create table bookmark(name text primary key, url text);

select distinct x.url, x.name
from Bookmark x, Bookmark y
where x.url = y.url and x.name != y.name
order by x.url;
```

(b) The browser wants to store a new page in the cache. There is no more space, and it decides to evict the oldest pages in order to make room. Write a SQL query that lists for each URL in the cache its size and the latest timestamp when that page was last accessed. Your query should return triples `url`, `size`, `ts`, ordered increasingly by `ts`.

Solution:

```
select y.url, max(x.ts) as tsmax, y.size
from History x, Cache y
where x.url = y.url
group by y.url, y.size
order by tsmax;
```

[2]Chrome uses SQlite to store and manage all its data.

No credit for returning all time stamps ts (i.e. must have some max or something like this)
no penalty for nested subquery (this was lenient!)
no penalty for missing $y.size$ in the group by

(c) The new page to be added to the cache has 1000 bytes, and the browser decides to evict from the cache the oldest pages in order to make room from the new page. For example, if there are four pages in the cache last accessed at time stamps 1,2,3,4 respectively, and their sizes are 500, 300, 300, 600, then the browser wants to delete the oldest three pages, since $500 + 300 + 300 \geq 1000$. Write a SQL query to return the URLs of all pages in the cache that the browser needs to delete to make room for 1000 bytes; your query should return a list of URL's (no need to actually delete them from `Cache`).

Note: only attempt to answer this question if you have answered question b. If you answered it, then you may refer to the query in b (no need to write it again).

Solution:

```
with tmp as
     (select y.url, max(x.ts) as tsmax, y.size
      from History x, Cache y
      where x.url = y.url
      group by y.url, y.size)
select x.url
from tmp x, tmp y
where x.url = y.url and y.tsmax < x.tsmax
group by x.url
having sum(y.size) < 1000;
```

(d) In this question we will represent sparse arrays and matrices as relations. For example, this shows how a matrix A and a vector X might be represented:

The matrix $A = \begin{bmatrix} 0 & 0 & 5 \\ 2 & -7 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ is represented as

A :

row	col	val
1	3	5
2	1	2
2	2	-7
3	3	-1

The array $X = \begin{bmatrix} 7 \\ 0 \\ -5 \\ 0 \end{bmatrix}$ is represented as

X :

pos	val
1	7
3	-5

Recall some standard definitions in linear algebra:

- Matrix/matrix product: $C = A \cdot B$, where $C_{ik} = \sum_j A_{ij} B_{jk}$.
- Matrix/vector product: $Y = A \cdot X$ where $Y_i = \sum_j A_{ij} X_j$.
- Vector/vector product: $X^t \cdot Y = \sum_i X_i Y_i$.
- The trace of a matrix is $\texttt{tr}(A) = \sum_i A_{ii}$.

A, B, C are matrices and X is a vector. For each of the SQL expressions below, write the corresponding formula in linear algebra.
For example, if the queries is:

```
select A.row, B.col, sum(A.val * B.val)
from A, B
where A.col = B.row
group by A.row,B.col;
```

then you answer $A \cdot B$.
If the query is:

```
select sum(A.val * B.val)
from A, B
where A.col = B.row and A.row = B.col;
```

then you answer $\texttt{tr}(A \cdot B)$, or $\texttt{tr}(B \cdot A)$ (both are correct answers).

Solution:

```
-- for testing only
drop table if exists A;
drop table if exists B;
drop table if exists C;
drop table if exists X;
create table A(row int, col int, val real);
create table B(row int, col int, val real);
create table C(row int, col int, val real);
create table X(pos int, val real);
```

i.

```
select sum(A.val) from A where A.row = A.col;
```

Linear algebra expression:

Solution: $\texttt{tr}(A)$

ii.

```
select sum(X1.val * A.val * X2.val)
from X X1, A, X X2
where X1.pos = A.col and A.row = X2.pos;
```

Linear algebra expression:

Solution: $X^t \cdot A \cdot X$

iii.

```
select sum(A.val) from A where A.row = A.col;
```

Linear algebra expression:

Solution: $\texttt{tr}(A)$

iv.

```
select A.row, A.col, A.val + sum(B.val*C.val)
from A, B, C
where A.row = B.row and B.col = C.row and C.col = A.col
group by A.row, A.col, A.val;
```

Linear algebra expression:

Solution: $A + B \cdot C$

v.

```
select sum(A.val*B.val*C.val)
from A, B, C
where A.col = B.row and B.col = C.row and C.col = A.row;
```

Linear algebra expression:

Solution: $\texttt{tr}(A \cdot B \cdot C)$

54. *(from CSE 414, Spring 2019, Final)*

A marine biologist collects samples of micro-organisms from the ocean. She stores her data in a relational database with the following schema:

```
Sample(sid, lat, long, depth, technician)
Analysis(sid, oid)
Organism(oid, name)
```

- `Sample` represents a small amount of water taken from the ocean. The data records the latitude, longitude, and depth where the sample was taken (real numbers), and the name of the technician (of type text) who collected and analyzed the sample.
- Each sample is analyzed for organisms. `Analysis` lists all organisms found in that sample. `sid` and `oid` are foreign keys to `Sample` and `Organism` respectively (integers).
- `Organism` is a collection of names of organisms (e.g. marine microbes). The name is a text.
- `sid`, `oid` are integers.

Solution:

```
-- FOR TESTING ONLY!!!
drop table if exists Analysis;
drop table if exists Sample;
drop table if exists Organism;
create table Sample (sid int primary key,
                     lat real, long real, depth int,
                     technician text);
create table Organism (oid int primary key, name text);
create table Analysis (sid int references Sample,
                       oid int references Organism);
```

(a) Technicians are rewarded by the total number of organisms that they identify in all samples. Write a SQL query that computes, for each technician, the total number of organisms that they found in all samples. (If Alice finds, say, Trichodesmium, in two different samples, then you count this as 2.) Your query should return the technician's name, the total number of organisms they identified, ordered decreasingly by this number.

Write your answer below:

Solution:

```
select x.technician, count(*)
from Sample x, Analysis y
where x.sid = y.sid
group by x.technician
order by count(*) desc;
```

(b) The biologists is particularly interested in the geographical area with latitude from 20.5 to 22.3 and longitude from -158.3 to -156.7. (Yes, this is close to Hawaii!) Write a SQL query that returns the `oid`s and names all organisms that were found in more than 100 samples from this area.

Write your answer below:

Solution:

```
select z.oid, z.name
from Sample x, Analysis y, Organism z
where x.sid = y.sid and y.oid = z.oid
  and 20.5 < x.lat and x.lat < 22.3
  and -158.3 < x.long and x.long < -156.7
group by z.oid, z.name
having count(*) > 100;
```

(c) A deep-see organism is one that lives only below a depth of 1000m. Write a SQL query that returns the names of all organisms that were found only at a depth greater than 1000m. (The `depth` is an integer and is represented in meters.)

Write your answer below:

Solution:

```
select x.oid, x.name
from Organism x
where not exists
  (select *
   from Analysis y, Sample z
   where x.oid = y.oid and y.sid = z.sid and z.depth < 1000);
```

5 Relational Algebra (55–66)

Relational algebra is the language query plans are written in, which is why it appears on nearly every final. The problems here ask you to translate between SQL and algebra in both directions, to read an expression drawn as a tree, and to decide whether two expressions are equivalent.

Watch the semantics carefully. Under set semantics projection removes duplicates, so $\Pi_A(R \bowtie S)$ can be a subset of $\Pi_A(R)$ but never a superset; under bag semantics the counts matter and several familiar equivalences fail. Selections push through joins and unions but not always through difference or aggregation, and the questions ask about exactly these boundaries.

A few problems supply the plan as a figure and ask you to annotate or rewrite it; those are the ones that lead directly into the optimization section later in the book.

55. *(from CSE 344, Spring 2011, Final)*

(a) Consider the following relational schema:

```
R(A)
S(A,B)
T(B,C)
U(A,C)
```

The following query is expressed in the relational calculus:

$$Q(x) = R(x) \wedge \forall y.(S(x,y) \Rightarrow \forall z.(T(y,z) \Rightarrow U(x,z)))$$

1. Write an equivalent query in non-recursive datalog with negation.

Solution: We need to be careful and make the query domain independent:

```
K(x,y) :- R(x), T(y,z), not U(x,z)
          /* R(x) added for domain independence */
L(x)   :- S(x,y), not K(x,y)
Q(x)   :- R(x), not L(x)
```

2. Write this query in the relational algebra.

Solution: All joins below are natural joins:

$$\begin{aligned} K &= \Pi_{AB}((R \bowtie T) - (T \bowtie U)) \\ L &= \Pi_A(S - K) \\ \texttt{Q} &= R - L \end{aligned}$$

(b) Consider a database consisting of the following two relations:

```
Person(pid, name)
Trusts(pid1, pid2)
```

Answer each question below by writing a query in non-recursive datalog with negation. You answer should return the person id and the name. For example, if the question were *find people who trust everyone except themselves* then your answer would be:

```
S(p)    :- Person(p,n), Person(q,m), not Trusts(p,q), p != q
A(p,n)  :- Person(p,n), not S(p)
```

1. A *loner* is a person who trusts no-one but himself. Return all loners.
Answer (write a datalog query):

Solution:

```
NA(p)  :- Trusts(p, x), p != x
A(p,n) :- Person(p,n), not NA(p)
```

2. A *loyal* is a person who trusts only those who trust him. Return all loyals.

Solution:

```
NA(p)   :- Trusts(p,x), not Trusts(x,p)
A(p,n)  :- Person(p,n), not NA(p)
```

3. A *ruler* is a person who trusts only those who trust only him. Return all rulers.

Solution:

```
NA(p)  :- Trusts(p,x), Trusts(x,y), p!=y
A(p,n) :- Person(p,n), not NA(p)
```

56. *(from CSE 344, Spring 2011, Midterm)*

Consider the following relational schema:

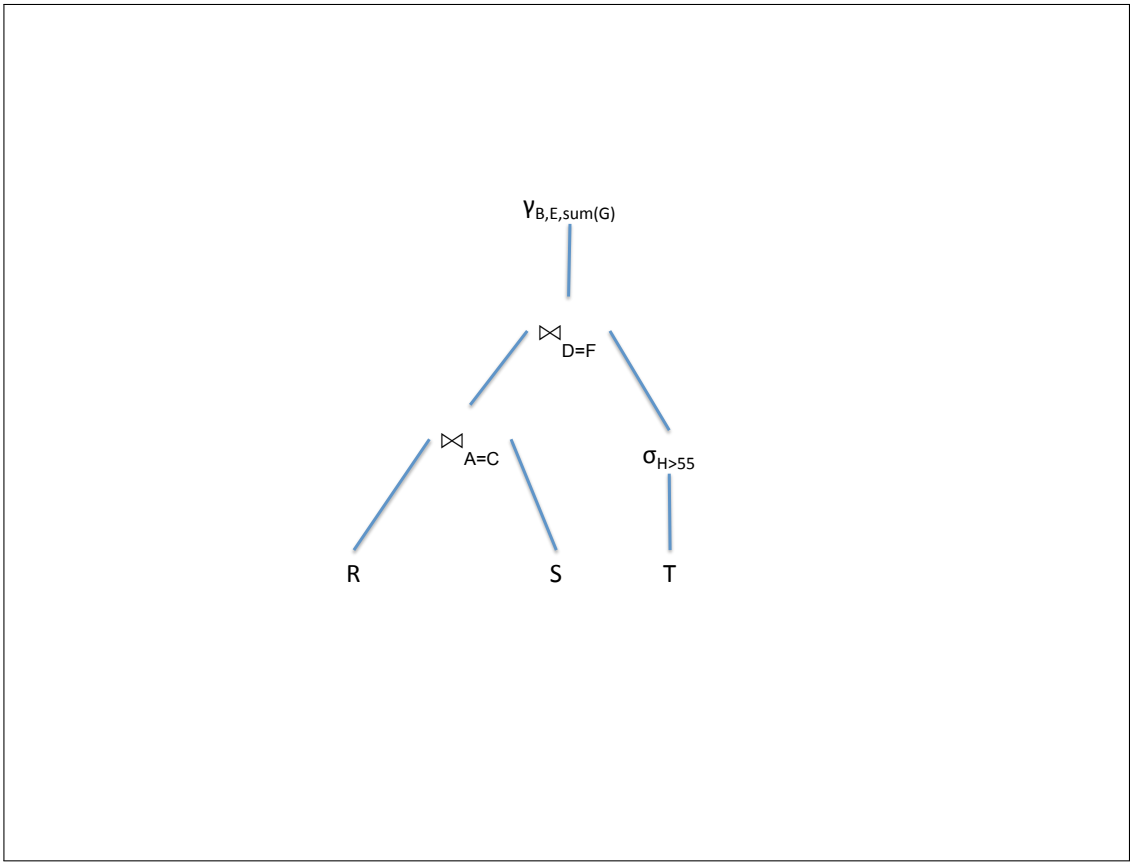
```
R(A,B)
S(C,D,E)
T(F,G)
```

(a) Write a **Relational Algebra Plan** for the SQL query below. Your answer should be a tree representing the relational algebra plan.

```
select R.B, S.E, sum(T.G)
form R, S, T
where R.A = S.C
  and S.D = T.F
  and T.H > 55
group by R.B, S.E
```

Answer (write a Relational Algebra Plan):

Solution:




```
R(A,B)
S(C,D,E)
T(F,G)
```

(b) Consider the pairs of relational algebra expressions below. For each pair indicate whether the two expressions are equivalent or not. In each row you should answer "yes" or "no". The first two rows are already answered for you, so you can see an example.

Expression 1	Expression 2	Equivalent ?
$R \bowtie_{A=C} S$	$S \bowtie_{C=A} R$	YES
$\sigma_{A>10 \vee B<50}(R)$	$\sigma_{A>10}(\sigma_{B<50}(R))$	NO
$\sigma_{B<50}(R \bowtie_{A=C} \sigma_{D>200}(S))$	$\sigma_{B<50}(R) \bowtie_{A=C} \sigma_{D>200}(S)$	
$\sigma_{B>D}(R \bowtie_{A=C} \sigma_{D>200}(S))$	$\sigma_{B>D}(\sigma_{B>200}(R) \bowtie_{A=C} \sigma_{D>200}(S))$	
$\sigma_{B<D}(R \bowtie_{A=C} \sigma_{D>200}(S))$	$\sigma_{B<D}(\sigma_{B<200}(R) \bowtie_{A=C} \sigma_{D>200}(S))$	
$\gamma_{B,sum(E)}(R \bowtie_{A=C} S)$	$\gamma_{B,sum(K)}(R \bowtie_{A=C} \gamma_{C,sum(E)\ as\ K}(S))$	
$\gamma_{B,sum(E)}((R \bowtie_{A=C} S) \bowtie_{D=F} T)$	$\gamma_{B,sum(E)}(R \bowtie_{A=C} S)$	
$\sigma_{B<G}((\sigma_{A<B}(R) \bowtie_{A=C} \sigma_{C>D}(S)) \bowtie_{D=F} \sigma_{F>G}(T))$	$(\sigma_{A<B}(R) \bowtie_{A=C} S) \bowtie_{D=F} T - (R \bowtie_{A=C} (S \bowtie_{D=F} T))$	

Solution: YES, YES, NO, YES, NO (the extra join removes tuples), YES (both are empty)

57. *(from CSE 344, Winter 2012, Midterm)*

```
Users(uid, name)
Comment(uid, pid, score, txt)
Picture(pid, author, img)
```

Consider the same database schema as in the first question:

```
Users(uid, name)
Comment(uid, pid, score, txt)
Picture(pid, author, img)
```

(a) Write a Relational Algebra expression that is equivalent to the SQL query below:

```
select distinct u.uid
from Users u, Picture x, Comment y
where u.uid = x.author and x.pid = y.pid and y.score > 8
group by u.uid, x.pid
having count(*) > 10
```

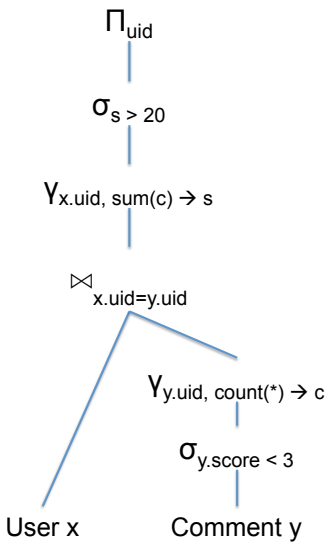
Solution: $\Pi_{u.uid}(\sigma_{cnt>10}(\gamma_{u.uid,count(*)\rightarrow cnt}(\texttt{Users u} \bowtie_{u.uid=x.author} (\texttt{Picture x} \bowtie \sigma_{score>8}(\texttt{Comment y})))))$

(b) Write a Relational Algebra expression that is equivalent to the SQL query below:

```
select x.pid
from picture x
where not exists
   (select *
    from comment y
    where x.pid = y.pid and y.score <5)
```

Solution: $\Pi_{pid}(\texttt{Picture}) - \Pi_{pid}(\sigma_{score<5}(\texttt{Comment y}))$

(c) Consider the Relational Algebra expression below:



Write an equivalent SQL query *without* using any subqueries.

Solution:

```
select x.uid
from Users x, Comment y
where x.uid = y.uid and y.score < 3
group by x.uid
having count(*) > 20
```

(d) Consider the following query in the relational calculus:

$$Q(u,n) = \texttt{Users}(u,n) \wedge \forall p.\forall i.\,(\texttt{Picture}(p,u,i) \Rightarrow \forall v.\forall s.\forall t.\,(\texttt{Comment}(v,p,s,t) \Rightarrow s > 5))$$

Write an equivalent expression that does not use a universal quantifier.

Solution:

$$\begin{aligned} Q(u,n) =& \texttt{Users}(u,n) \\ & \wedge\neg\exists p.\exists i.\,(\texttt{Picture}(p,u,i) \wedge \exists v.\exists s.\exists t.\,(\texttt{Comment}(v,p,s,t) \wedge s \leq 5)) \end{aligned}$$

58. *(from CSE 344, Winter 2013, Midterm)*

Consider the following database schema:

```
Person(name)
Friend(name1, name2)
```

`Person` contains all persons in the database, while `Friend` contains all friendship relationships.

Solution: Test the solutions using the following database:

```
create table Person(name text);
create table Friend(name1,name2);

insert into Person values('Alice');
insert into Person values('Bob');
insert into Person values('Carol');

insert into Friend values('Alice','Bob');
insert into Friend values('Alice','Carol');
insert into Friend values('Bob','Carol');
insert into Friend values('Carol','Alice');
```

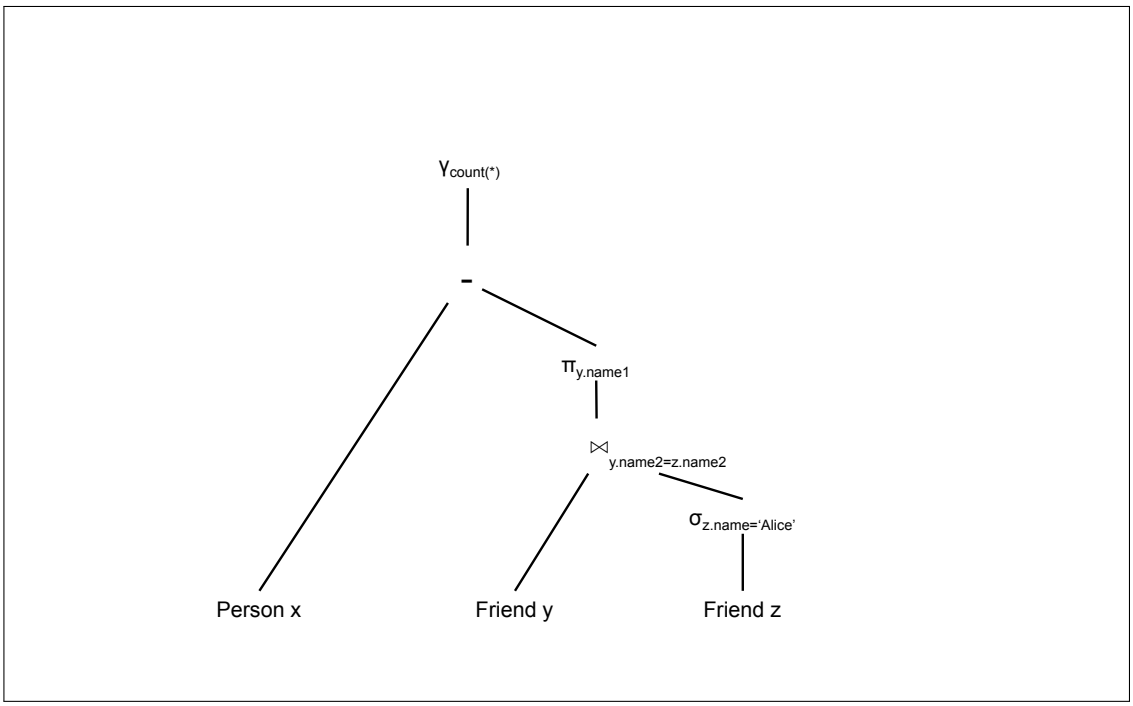
(a) The SQL query below counts how many persons have no friends in common with Alice:

```
select count(*)
from Person x
where not exists
       (select *
        from Friend y, Friend z
        where x.name =y.name1
          and z.name1='Alice'
          and z.name2=y.name2);
```

Write a relational algebra expression that is equivalent to this SQL query. Write your answer as query plan (i.e. a tree).

Solution:

$$\gamma_{\texttt{count(*)}}(\texttt{Person} - \pi_{\texttt{y.name1}}(\texttt{Friend y} \bowtie \sigma_{\texttt{z.name1='Alice'}}\texttt{,Friend z}))$$



(b) Write a query in the relational calculus that finds all persons who are friends with all of Alice's friends.

Solution:

$$q(x) = \texttt{Person}(x) \wedge \forall y.\texttt{Friend}(\texttt{'Alice'}, y) \Rightarrow \texttt{Friend}(x, y)$$

(c) Write a query in datalog with negation that finds all persons who are friends with all of Alice's friends.

Solution:

```
notAnswer(x) :- Person(x),Friend('Alice',y),not Friend(x,y)
Answer(x)    :- Person(x), not notAnswer(x)
```

59. *(from CSE 344, Winter 2016, Midterm)*

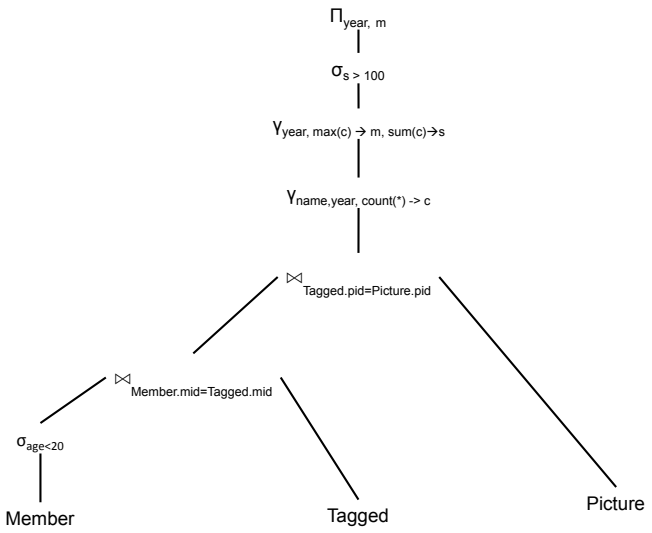
Consider the same relational schema as before:

```
Member(mid, name, age)
Picture(pid, year)
Tagged(mid, pid)
```

(a) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

```
select w.year, max(w.c) as m
from (select x.name, z.year, count(*) as c
      from Member x, Tagged y, Picture z
      where x.mid = y.mid
        and y.pid = z.pid
        and age < 20
      group by x.name, z.year) w
group by w.year
having sum(w.c) > 100;
```

Solution:



(b) Write a query in datalog with negation that returns the mid's and names of all members that were tagged only in pictures were Alice was also tagged.

Solution:

```
aliceTagged(pid) :- Member(mid, 'Alice',-),Tagged(mid, pid)
nonAnswer(mid) :- Tagged(mid,pid) not aliceTagged(pid)
answer(mid,name) :- Member(mid,name,-), not nonAnswer(mid)
```

(c) This question is about matching relational calculus queries to English statements.

Consider the relational calculus queries below:

(a)

$$P1(m,x) = \exists a.\texttt{Member}(m,x,a) \wedge \forall p(\texttt{Tagged}(m,p) \Rightarrow (\texttt{Picture}(p,2015) \vee \texttt{Picture}(p,2016)))$$

(b)

$$P2(m,x) = \exists a.\texttt{Member}(m,x,a) \wedge \forall p((\texttt{Picture}(p,2015) \vee \texttt{Picture}(p,2016)) \Rightarrow \texttt{Tagged}(m,p))$$

(c)

$$P3(m,x) = \exists a.\texttt{Member}(m,x,a) \wedge [\forall p(\texttt{Picture}(p,2015) \Rightarrow \texttt{Tagged}(m,p)) \vee \forall p(\texttt{Picture}(p,2016) \Rightarrow \texttt{Tagged}(m,p))]$$

(d)

$$P4(m,x) = \exists a.\texttt{Member}(m,x,a) \wedge \forall p1(\texttt{Tagged}(m,p1) \wedge \texttt{Picture}(p1,2015) \Rightarrow \exists p2(\texttt{Tagged}(m,p2) \wedge \texttt{Picture}(p2,2016)))$$

(e)

$$P5(m,x) = \exists a.\texttt{Member}(m,x,a) \wedge \forall p \forall n((\texttt{Picture}(p,2015) \vee \texttt{Picture}(p,2016)) \wedge \texttt{Tagged}(n,p) \Rightarrow m = n)$$

(continued on the next page)

For each English statement below, indicate which relational calculus query answers it:

(i) Find all members that were tagged in 2015 only if they were also tagged in 2016. Choose one of P1,P2,P3,P4,P5,none:

Solution: P4

(ii) Find all members that were tagged in all pictures in 2015 and in all pictures in 2016. Choose one of P1,P2,P3,P4,P5,none:

Solution: P2

(iii) Find the member that was the only person tagged in 2015 and 2016. Choose one of P1,P2,P3,P4,P5,none:

Solution: P5

(iv) Find all members that were tagged only in pictures in 2015 or 2016. Choose one of P1,P2,P3,P4,P5,none:

Solution: P1

(v) Find all members that were tagged in all pictures in 2015, or were tagged in all pictures in 2016. Choose one of P1,P2,P3,P4,P5,none:

Solution: P3

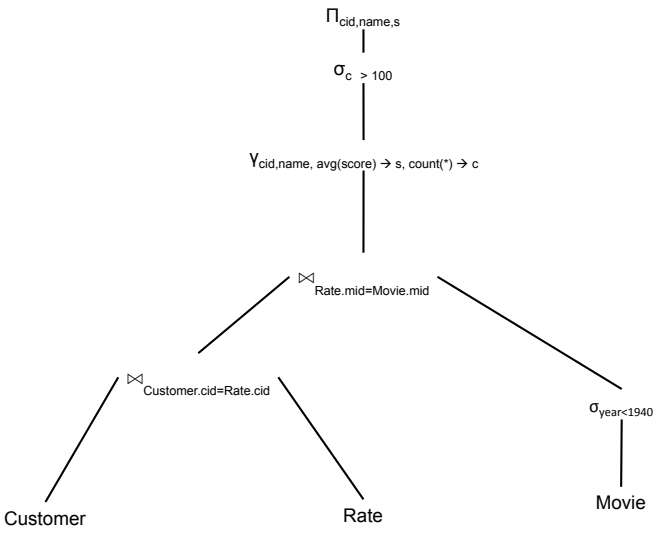
60. *(from CSE 414, Spring 2016, Midterm)*

Consider the same relational schema as before:

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(a) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

```
select x.cid, x.name, avg(y.score) as s
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid and z.year < 1940
group by x.cid, x.name
having count(*) > 100;
```



Solution:

(b) Write a query in datalog that returns all customers who rated the movie "Hunger Games" with a score of 3; return their cid's and their names.

Solution:

```
answer(cid, name) :- Customer(cid, name, country),
                     Rate(cid, mid, 3),
                     Movie(mid, 'Hunger Games',year)
```

(c) Write a query in datalog with negation that returns all movies who were never rented by any customer living in Canada; return their mid's and their titles.

Solution:

```
NonAnswer(mid) :- Customer(cid, name, 'Canada'),
                  Rent(cid, mid, date),
                  Movie(mid, title, year)
Answer(mid, title) :- Movie(mid, title, year), not nonAnswer(mid)
```

61. *(from CSE 344, Autumn 2017, Midterm)*

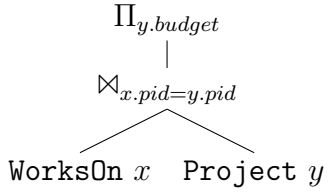
Consider the same relational schema as in the SQL question, including the key/foreign key constraints:

```
Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)
```

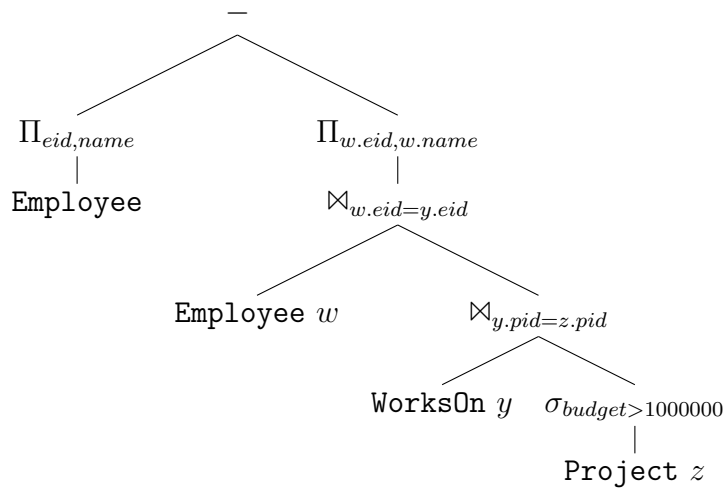
(a) Write a Relational Algebra expression in the form of a logical query plan (i.e. draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal"; however, points will be taken off for overly complex solutions.

```
select x.eid, x.name
from Employee x
where not exists (select *
                  from WorksOn y, Project z
                  where x.eid = y.eid and y.pid = z.pid
                    and z.budget > 1000000);
```

Hint: to avoid renaming, use aliases in the query plan, like this:



Solution:



(b) Assume that the database instance does not contain NULLs, but otherwise can be any valid instance (satisfying all key/foreign key constraints). Answer the questions below, where the relational algebra expressions have *bag* semantics.

i. Are these two expressions equivalent?

$$\Pi_{year}(\sigma_{salary>40000}(\texttt{Employee} \bowtie_{eid=eid} \texttt{WorksOn})) \quad = \quad \Pi_{year}(\texttt{WorksOn})$$

Yes/No:

Solution: No

ii. Are these two expressions equivalent?

$$\Pi_{year}(\sigma_{year>2015}(\texttt{Employee} \bowtie_{eid=eid} \texttt{WorksOn})) \quad = \quad \Pi_{year}(\sigma_{year>2015}(\texttt{WorksOn}))$$

Yes/No:

Solution: Yes (there are no NULLs).

iii. Are these two expressions equivalent?

$$\Pi_{name}(\sigma_{salary>40000}(\texttt{Employee} \bowtie_{eid=eid} \texttt{WorksOn})) \quad = \quad \Pi_{name}(\sigma_{salary>40000}(\texttt{Employee}))$$

Yes/No:

Solution: No (bag semantics).

iv. Are these two expressions equivalent?

$$\Pi_{name}(\sigma_{year>2015}(\texttt{Employee} \bowtie_{eid=eid} \texttt{WorksOn})) \quad = \quad \Pi_{name}(\texttt{Employee})$$

Yes/No:

Solution: No

v. The notation $|S|$ means the cardinality of a set S (number of tuples in S). Does the following always hold?

$$|\texttt{Employee} \bowtie_{eid=eid} \sigma_{year<2015}(\texttt{WorksOn})| \leq |\texttt{Employee}|$$

Yes/No:

Solution: No

vi. Does the following always hold?

$$|\sigma_{salary<5000}(\texttt{Employee}) \bowtie_{eid=eid} \texttt{WorksOn}| \leq |\texttt{WorksOn}|$$

Yes/No:

Solution: Yes

(c) For each statement below, indicate whether it is true or false.

i. If the attribute K of a relation is a key, then no two tuples in the relation can have the same value of K. True/False:

Solution: True

ii. If the attribute K of a relation is a foreign key, then no two tuples in the relation can have the same value of K. True/False:

Solution: False

iii. First normal form means that all relations in the database are flat. True/False:

Solution: True

iv. Physical data independence means that the relations in the database are physically independent of each other. True/False:

Solution: False

v. All queries expressible in Relational Algebra are monotone. True/False:

Solution: False

vi. All queries expressible in datalog (with recursion, but without negation and without aggregates) are monotone. True/False:

Solution: True

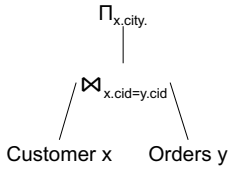
62. *(from CSE 344, Winter 2019, Midterm)*

Consider the same relational schema as before:

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(a) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

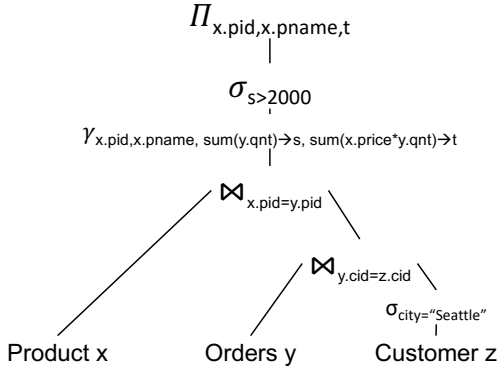
Hint: to avoid renaming, use aliases in the query plan, like this




```
select x.pid, x.pname, sum(x.price*y.qnt) as revenue
from Product x, Orders y, Customer z
where x.pid = y.pid and y.cid = z.cid
  and z.city = 'Seattle'
group by x.pid, x.pname
having sum(y.qnt) > 2000;
```

Write the Relational Query expression below:

Solution:



(b) i. Which of the following is the most accurate English interpretation of the SQL query below?

```
select distinct z.city
from Customer z
where not exists
       (select *
```

```
        from Orders y
        where y.cid = z.cid and y.qnt < 100);
```

Returns cities that ...

(A) had at least one order for < 100 units of a product.

(B) have a customer who placed at least one order for more < 100 units of a product.

(C) had only orders with ≥ 100 units.

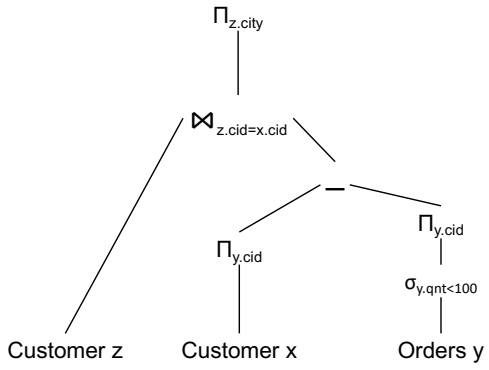
(D) have a customer that placed only orders with ≥ 100 units.

A/B/C/D:

Solution: D

ii. Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query above. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Solution:



(c) We continue to use the same schema as before; remember that some attributes are keys, and others are foreign keys. We further assume that the database has no NULLs. Answer the questions below, assuming all relational algebra expressions have set semantics. The notation $|S|$ means the cardinality of a set S (number of tuples in S).

i. Does the following always hold?

$$|(\texttt{Product x}) \bowtie_{\texttt{x.pid=y.pid}} \sigma_{\texttt{y.qnt}>20}(\texttt{Orders y})| = |\texttt{Orders}|$$

Yes/No:

Solution: No

ii. Does the following always hold?

$$|(\texttt{Product x}) \bowtie_{\texttt{x.pid=y.pid}} \sigma_{\texttt{y.qnt>20}}(\texttt{Orders y})| = |\sigma_{\texttt{qnt>20}}(\texttt{Orders})|$$

Yes/No:

Solution: Yes

iii. Does the following always hold?

$$\sigma_{\texttt{y.qnt>20}}(\texttt{Orders y}) \bowtie_{\texttt{x.pid=y.pid}} (\texttt{Product x}) =$$
$$= \sigma_{\texttt{y.qnt>20}}(\texttt{Product x} \bowtie_{\texttt{x.pid=y.pid}} (\texttt{Orders y}))$$

Yes/No:

Solution: Yes

iv. Does the following always hold?

$$\gamma_{\texttt{x.city,count}(*)\rightarrow\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} (\texttt{Orders y})) =$$
$$= \Pi_{\texttt{x.city,cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} \gamma_{\texttt{y.cid,count}(*)\rightarrow\texttt{cnt}}(\texttt{Orders y}))$$

Yes/No:

Solution: No

v. Does the following always hold?

$$\gamma_{\texttt{x.city,count}(*)\rightarrow\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} (\texttt{Orders y})) =$$
$$= \gamma_{\texttt{x.city,sum(cnt)}\rightarrow\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} \gamma_{\texttt{y.cid,count}(*)\rightarrow\texttt{cnt}}(\texttt{Orders y}))$$

Yes/No:

Solution: Yes

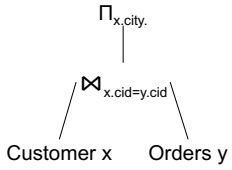
63. *(from CSE 414, Spring 2019, Midterm)*

Consider the same relational schema as before:

```
Car(lp, make, owner)
Downtown(lp, day)
```

(a) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

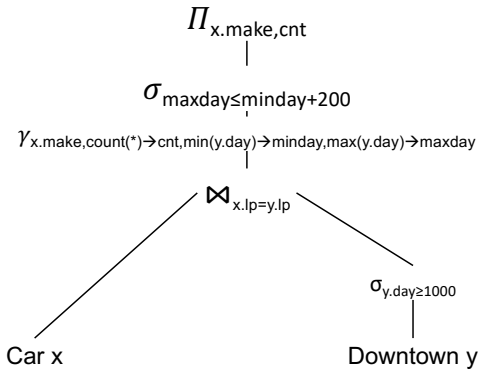
Hint: to avoid renaming, use aliases in the query plan, like this




```
select x.make, count(*) as cnt
from Car x, Downtown y
where x.lp = y.lp
  and y.day >= 1000
group by x.make
having max(y.day) <= min(y.day) + 200;
```

Write the Relational Query expression below:

Solution:



(b) i. Which of the following is the most accurate English interpretation of the SQL query below?

```
select distinct u.owner
from Car u
where not exists
       (select *
        from Car x, Downtown y
        where u.owner = x.owner
```

```
        and x.make = 'Honda'
        and x.lp = y.lp
        and y.day >= 1000);
```

Returns all owners that ...

(A) Don't own a Honda and never drove downtown after day 1000.

(B) Own a Honda but never drove downtown after day 1000.

(C) Never drove downtown with a Honda after day 1000.

(D) Never drove downtown with a Honda before day 1000.

(E) All their trips downtown before day 1000 were in a Honda.

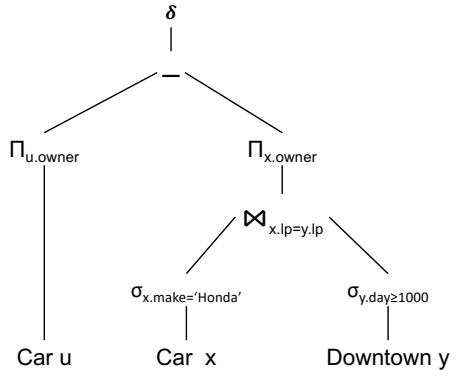
(F) All their trips downtown after day 1000 were in a Honda.

A/B/C/D/E/F:

Solution: C

ii. Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query above. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Solution:



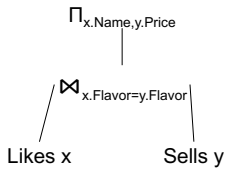
64. *(from CSE 414, Spring 2024, Midterm; CSE 344, Autumn 2024, Midterm)*

Consider the same relational schema as before:

```
Cust(Cid,Name,Age)
Likes(Cid,Flavor)
Sells(Store,Flavor,Price)
```

(a) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

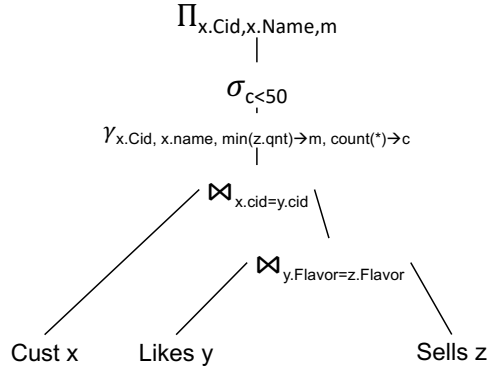
Hint: to avoid renaming, use aliases in the query plan, like this




```
SELECT x.Cid, x.Name, min(z.Price)
FROM Cust x, Likes y, Sells z
WHERE x.Cid = y.Cid and y.Flavor = z.Flavor
GROUP BY x.Cid, x.Name
HAVING count(*) < 50;
```

Write your answer here (you should turn in a Relational Algebra Plan):

Solution:



(b) Consider the following two relations:

R:

A	B
1	10
1	20
2	20

S:

B	C
10	100
20	200
20	300
30	400

For each relational algebra expression below, write its answer. The join operator $\bowtie$ is the natural join. We assume set semantics, so, for example, the answer to $\Pi_A(R)$ should be

A
1
2

because we remove duplicates: $\{1, 1, 2\}$ is the same as $\{1, 2\}$.

i. $R \bowtie S$. Write your answer below:

Solution:

A	B	C
1	10	100
1	20	200
1	20	300
2	20	200
2	20	300

ii. $R \bowtie \sigma_{C<250}(S)$. Write your answer below:

Solution:

A	B	C
1	10	100
1	20	200
2	20	200

iii. $R \bowtie \Pi_C(\sigma_{C<250}(S))$. Write your answer below:

Solution:

A	B	C
1	10	100
1	10	200
1	20	100
1	20	200
2	20	100
2	20	200

iv. $\Pi_B(R) \bowtie \Pi_B(S)$. Write your answer below:

Solution:

B
10
20

v. $\Pi_B(S) - \Pi_B(R)$. Write your answer below:

Solution:

B
30

(c) Answer the questions below. For the questions about relational algebra, assume set semantics.

i. In the relational model all tables need to be flat.
True or False?

Solution: True

ii. A nested SQL query returns a nested table.
True or False?

Solution: False

iii. If R, S, T are any three relations, then $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$, where $\bowtie$ is the natural join.
True or False?

Solution: True

iv. If R, S, T are three relations with the same schema (i.e. the same set of attributes), then the following two Relational Algebra expression are equal: $R - (S - T) = (R - S) \cup T$
True or False?

Solution: False

v. If R, S, T are three relations with the same schema, then the following two Relational Algebra expression are equal: $(R \cup S) - T = (R - T) \cup (S - T)$.
True or False?

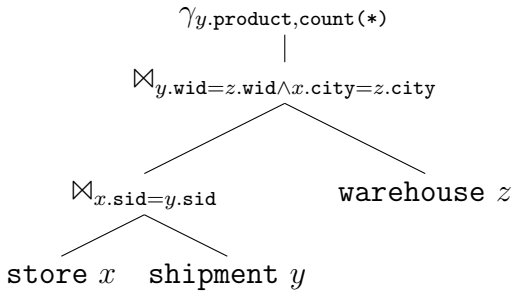
Solution: True

65. *(from CSE 414, Spring 2024, Final; CSE 344, Spring 2026, Makeup Final)*

(a) Write a relational algebra plan for the following SQL query:

```
select y.product, max(y.date)
from store x, shipment y, warehouse z
where x.sid = y.sid and y.wid = z.wid
  and x.city = z.city
group by y.product
having count(*) > 10;
```

Solution:



(b) Write a relational algebra plan for the following SQL query:

```
select z.wid
from warehouse z
where not exists
       (select *
        from shipment y
        where y.wid = z.wid and y.product = 'TV');
```

Solution: $\Pi_{\texttt{wid}}(\texttt{Warehouse}) - \Pi_{\texttt{wid}}(\sigma_{\texttt{product='TV'}}(\texttt{Shipment}))$

66. *(from CSE 344, Autumn 2024, Final; CSE 344, Spring 2026, Final)*

(a) Consider the following relational database schema:

`Customer(`<u>`cid`</u>`, name, zip)`
`Orders(`<u>`oid`</u>`, cid, product, price)`

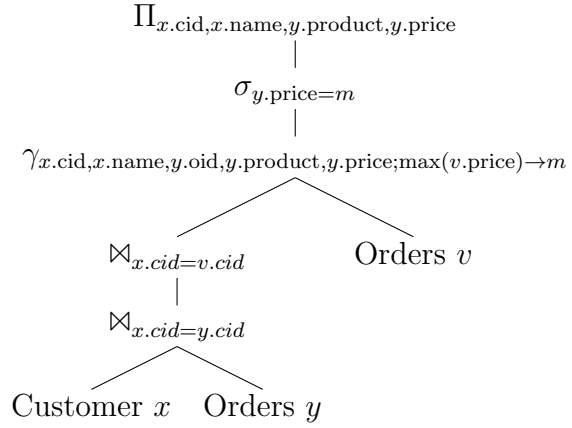
i. Write a relational algebra plan that computes the following SQL query:

```
select x.cid, x.name, y.product, y.price
from Customer x, Orders y, Orders v
where x.cid = y.cid and x.cid = v.cid
group by x.cid, x.name, y.oid, y.product, y.price
having y.price = max(v.price);
```

Write your answer here:

Solution:

```
drop table if exists Orders;
drop table if exists Customer;

create table Customer(cid int primary key, name text, zip int);

create table Orders(oid int primary key,
                    cid int references Customer,
                    product text,
                    price int);

insert into Customer values
  (100, 'Alice', 98195),
  (200, 'Bob', 98195),
  (300, 'Carol', 98190);

insert into Orders values
  (101, 100, 'iPad', 350),
  (102, 100, 'tent', 150),
  (103, 100, 'jacket', 450),
  (201, 200, 'iPad', 350),
  (202, 200, 'tent', 150);
```

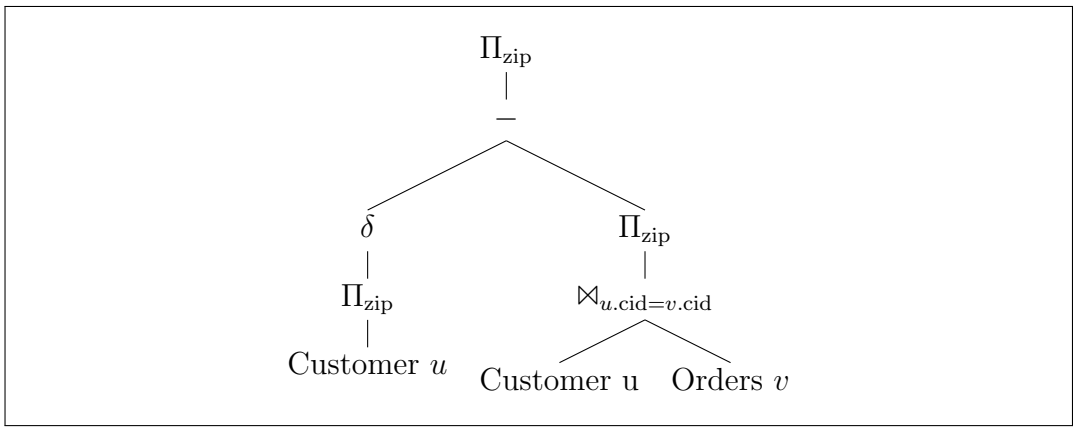


ii. Write a relational algebra plan that computes the following SQL query:

```
select distinct x.zip
from Customer x
where not exists
    (select *
     from Customer u, Orders v
     where x.zip = u.zip and u.cid = v.cid);
```

Write your answer here:

Solution:


Π_{zip}
$-$
δ
Π_{zip}
Π_{zip}
$\bowtie_{u.\text{cid}=v.\text{cid}}$
Customer u
Customer u
Orders v

6 Relational Calculus (67–70)

A small graduate-level section, drawn entirely from CSE 544. The questions ask you to express a query as a first-order formula, to translate a formula into algebra or SQL, and to recognize formulas that are unsafe, i.e. those whose answer depends on the underlying domain rather than on the database.

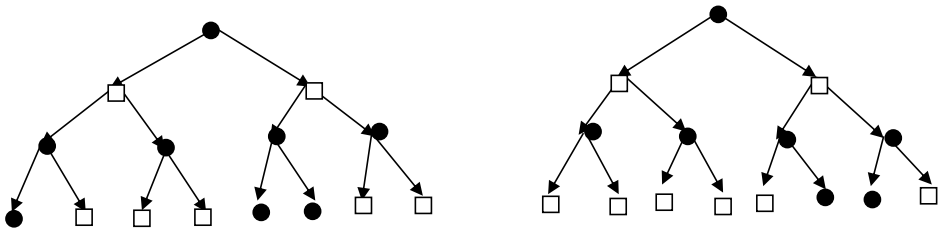
The technical core is quantifier handling. Queries phrased with *all* or *every* become universal quantifiers, which then have to be turned into a double negation before they can be written in SQL, and getting the scope of the quantifiers right is most of the work. Expect to spend more time per question here than anywhere else in the book.

67. *(from CSE 544, 2001, Final)*

(a) Evaluate the following query expressed in first order logic:

$$Red(x) \wedge \forall y.\big(R(x,y) \Rightarrow Blue(y) \wedge \exists z.(R(y,z) \wedge Red(z) \wedge \forall u.(R(z,u) \Rightarrow Blue(u)))\big)$$

Compute its answer on the following database (edges represent the relation $R(x,y)$, • represents $Red(x)$ and □ represents $Blue(x)$; please circle the nodes that satisfy the formula):



Solution: Left tree: the root and the three Red leaves.
Right tree: the two Red leaves (not the root!).

(b) Given a relation $R(x,y)$ describing a graph, determine whether the following queries are expressible in the languages indicated. Your answer should consist only of a "YES" or "NO".

i. $\{(x,y) \mid x$ and y are connected by a path of length $4\}$ in FO.

Solution: Yes

ii. $\{(x,y,z,u,v) \mid \{x,y,z,u,v\}$ is a clique of size $5\}$ in FO.

Solution: Yes

iii. $\{(x,y) \mid x$ and y are connected by a path of even length$\}$ in datalog.

Solution: Yes

iv. $\{x \mid x$ belongs to a clique whose size is at least half that of the number of nodes in the graph$\}$ in datalog$^\neg$.

Solution: No

(A *clique* in a graph is a set S of nodes such that any two nodes in S are connected by an edge.)

68. *(from CSE 544, 2002, Final)*

Consider a graph with coloured edges, $G(x, c, y)$, where x, y are nodes and c is a colour.

(a) We say that x is a "red" node if all its outgoing edges are "red". Write a FO query that returns all red nodes.

Solution:

$$\forall c.\forall y.\,(G(x, c, y) \Rightarrow c = \text{'red'})$$

(b) If (x, c, y) is in G, then we say that y is a *c-neighbor* of x. Write an FO query that returns all nodes where all "blue"-neighbors have at least one "green"-neighbor that is a "red" node.

Solution:

$$\forall y.\Big(G(x, \text{'blue'}, y) \Rightarrow \exists z.(G(y, \text{'green'}, z) \wedge \forall c.\forall u.\,(G(z, c, u) \Rightarrow c = \text{'red'}))\Big)$$

(c) Write a datalog program to compute the pairs of nodes (x, y) connected by a path in which all edges have the same colour. Your program should compute a binary predicate `Answer(x,y)` consisting of all pairs (x, y) in the answer.

Solution:

```
P(x,c,y)    :- G(x,c,y)
P(x,c,y)    :- P(x,c,z), G(z,c,y)
Answer(x,y) :- P(x,c,y)
```

69. *(from CSE 544, 2004, Final)*

Consider the following relational schema:

`Company(name, city)`	
`Product(pid, manufacturer)`	`manufacturer` is a foreign key in `Company`
`Order(oid, city)`	`city` is the shipping address for the order
`OrderItem(oid, pid)`	says which products have been placed on each order

Write First Order Logic queries to compute the following:

(a) All products manufactured in Seattle. This query returns a set of `pid`'s.

Solution:

$$\exists m.\, P(pid, m) \wedge C(m, \text{Seattle})$$

(b) All products that have been shipped at least once to the same city where they were manufactured.

Solution:

$$\exists m. \exists c.\, P(pid, m) \wedge C(m, c) \wedge \exists oid.\, OI(oid, pid) \wedge O(oid, c)$$

(c) All products that have been shipped only to the cities where they have been manufactured.

Solution:

$$\exists m. \exists c.\, P(pid, m) \wedge C(m, c) \wedge \forall oid.\, (OI(oid, pid) \Rightarrow O(oid, c))$$

(d) All companies that manufacture only products that have been shipped only to the city where they have been manufactured. This query returns a set of `name`'s.

Solution:

$$\exists c.\, C(m, c) \wedge \forall pid.\, \big(P(pid, m) \Rightarrow \forall oid.\, (OI(oid, pid) \Rightarrow O(oid, c))\big)$$

70. *(from CSE 544, 2006, Final)*

Consider the following relational schema:

```
Person(name)
Trusts(name1, name2)
```

The `Trusts` relation tells us who trusts whom. This relation is not necessarily symmetric: i.e. if 'John' trusts 'Mary', then it is not necessarily the case that 'Mary' trusts 'John'. This relation is not necessarily transitive: i.e. if 'John' trusts 'Mary' and if 'Mary' trusts 'Fred', it is not necessarily the case that 'John' trusts 'Fred'.

(a) Consider the following types of people:

- A "loner" is a person who trusts no one but himself.

- A "loyal" is a person who trusts only those who trust him.
- A "ruler" is a person who trusts only those who trust only him.

Write three queries in First Order Logic to compute the set of loners, the set of loyals, and the set of rulers:

i. The loners.

Solution:

$$Q(x) = \text{Person}(x) \wedge \forall y(\text{Trusts}(x, y) \Rightarrow x = y)$$

ii. The loyals.

Solution:

$$Q(x) = \text{Person}(x) \wedge \forall y(\text{Trusts}(x, y) \Rightarrow \text{Trusts}(y, x))$$

iii. The rulers.

Solution:

$$Q(x) = \text{Person}(x) \wedge \forall y \forall z(\text{Trusts}(x, y) \wedge \text{Trusts}(y, z) \Rightarrow z = x)$$

(b) Consider the following five boolean queries:

`Q1`	There are an even number of tuples in `Trusts`
`Q2`	`'Mary'`, `'John'` are in the transitive closure of `Trusts`
`Q3`	There exists a group of seven different people that all trust each other
`Q4`	There exists a group of people consisting of at least half of all persons such that no two people in the group trust each other
`Q5`	`'Sue'` trusts everyone else in the database

Consider the following query languages:

- Conjunctive queries (CQ).
- First Order Logic (FO).
- Datalog (recursive, without negation).

Indicate for each query if it can be expressed in the languages above. You have to turn in a `yes` or a `no` in each of the fifteen entries below, and do not have to justify your answers:

Query	CQ	FO	Datalog
`Q1`			
`Q2`			
`Q3`			
`Q4`			
`Q5`			

Solution:

Query	CQ	FO	Datalog
`Q1`	no	no	no
`Q2`	no	no	yes
`Q3`	no	yes	no
`Q4`	no	no	no
`Q5`	no	yes	no

7 Datalog (71–80)

We cover datalog in every graduate level class 544 and 544p. In the past, we covered datalog in some offerings of 344 and the early 444, but we not longer cover it in the undergraduate courses.

Datalog states a query as a set of rules, and its distinguishing feature is recursion: reachability in a graph, ancestry, and transitive closure in general can be written in a few lines that have no equivalent in non-recursive SQL. Many problems here are exactly that (a graph, a family tree, a call graph between library functions) and ask for the rules.

Non-recursive Datalog is expressively equivalent to relational algebra, and several questions ask you to exploit that correspondence in one direction or the other. Others introduce negation, where the order of evaluation starts to matter and the notion of stratification becomes necessary.

The most common mistakes are forgetting the base case of a recursion and writing a rule that is unsafe because a variable in the head never appears in a positive atom of the body.

71. *(from CSE 544, 2001, Final)*

Consider the following schema:

```
R(x,y)   Red(x)   Blue(x)   Leaf(x)
```

and assume we have a database that is a tree, in which every node is coloured either Red or Blue, and where `Leaf(x)` tells us whether x is a leaf. A Red node is called a "winner" if all its descendants are winners; otherwise it is a "loser". A Blue node is called a winner if at least one of its descendants is a winner; otherwise it is a "loser".

(a) Write a datalog program that computes all winner nodes in the tree.

Solution: Partial datalog with negation:

```
Win(x)  :- Red(x), Leaf(x)
Lose(x) :- Blue(x), Leaf(x)
Win(x)  :- Blue(x), R(x,y), Win(y)
Lose(x) :- Red(x), R(x,y), Lose(y)
Win(x)  :- Red(x), not Lose(x)
Lose(x) :- Blue(x), not Win(x)
```

Inflationary datalog with negation, assuming a perfectly balanced tree:

```
Win(x)   :- Red(x), Leaf(x)
Lose(x)  :- Blue(x), Leaf(x)
Win(x)   :- Blue(x), R(x,y), Win(y)
Lose(x)  :- Red(x), R(x,y), Lose(y)
Known(x) :- Leaf(x)
Known(x) :- R(x,y), Known(y)
Win(x)   :- Red(x), not Lose(x), Known(x)
Lose(x)  :- Blue(x), not Win(x), Known(x)
```

72. *(from CSE 344, Autumn 2013, Midterm)*

Consider the following two relations:

```
Person(id, name)
Trusts(id1, id2)
```

(a) Write a program in non-recursive datalog-with-negation that returns the id's and names of all persons who don't trust Alice. Turn in a datalog program:

Solution: Solution 1:

```
Ans(x,y) :- Person(x,y), Person(z,'Alice'), x!=z
```

Solution2:

```
NonAns(x) :- Trusts(x,z), Person(z,'Alice')
Ans(x,y) :- Person(x,y), not NonAns(x)
```

(b) Write a program in non-recursive datalog-with-negation that returns the id's and names of all persons who trust only Alice. (In particular, your query should return all persons who don't trust anyone, e.g. persons who do not appear in `Trust`.) Turn in a datalog program:

Solution: Solution 1:

```
NonAnsw(x) :- Trusts(x,y), not Person(y,'Alice')
Ans(x,y) :- Person(x,y), not NonAns(x)
```

Solution 2:

```
NonAnsw(x) :- Trusts(x,y), Person(z,'Alice'), y != z
Ans(x,y) :- Person(x,y), not NonAns(x)
```

Solution 3:

```
NonAnsw(x) :- Trusts(x,y), Person(y,n), n != 'Alice'
Ans(x,y) :- Person(x,y), not NonAns(x)
```

(c) Consider the following query in the relational calculus:

$$A(x,n) = \texttt{Person}(x,n) \wedge \forall y.(\texttt{Trusts}(x,y) \Rightarrow \forall z.(\texttt{Trusts}(y,z) \Rightarrow \neg\texttt{Person}(z,\texttt{'Alice'})))$$

i. Write this query in non-recursive datalog-with-negation. Turn in a datalog program:

Solution:

```
U(x) :- Trusts(x,y), Trusts(y,z), Person(z,'Alice')
A(x,n) :- Person(x,n), not U(x)
```

It's also OK to write two rules for `U`:

```
V(y) :- Trusts(y,z), Person(z,'Alice')
U(x) :- Trusts(x,y), V(y)
```

ii. Write this query in SQL. Turn in a SQL query:

Solution:

```
select p.id, p.name
from Person p
where not exists
          (select *
           from Trusts t1, Trusts t2, Person q
           where p.id = t1.id1 and t1.id2 = t2.id1
             and t2.id2 = q.id and q.name = 'Alice')
```

In many cases the datalog program was incorrect but the SQL query correctly implemented the datalog program; in that case I gave full credit.

(d) Consider the following SQL query:

```
select distinct t1.id1
from Trusts t1, Person p1
where t1.id2 = p1.id and p1.name = 'Alice' and
  not exists (select *
              from Trusts t2, Person p2
              where p1.id = t2.id1 and t2.id2 = p2.id and p2.name = 'Bob')
```

Write this query in the Relational Algebra. Turn in a Relational Algebra plan:

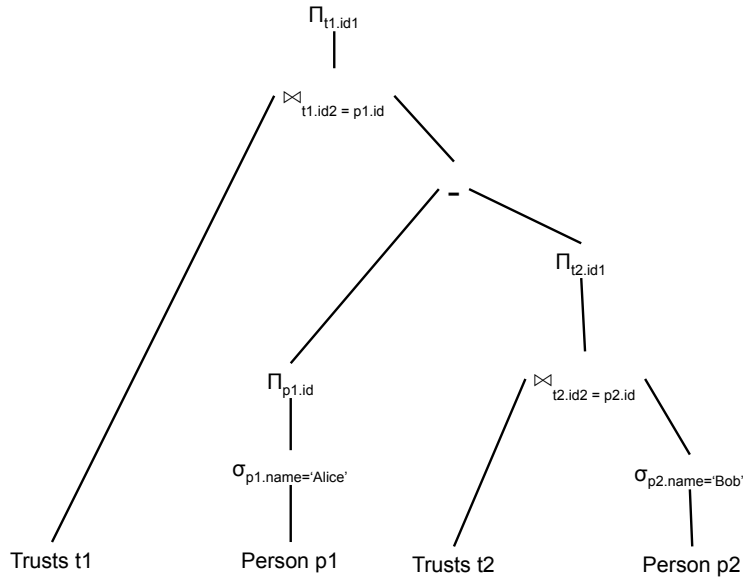
Solution: Write it first in datalog (it's much easier to read this way):

```
V(y) :- Trusts(y,z), Person(z,'Bob')
Q(x) :- Trusts(x,y), Person(y,'Alice'), not V(y)
```

Next, expose the difference clearly:

```
V(y) :- Trusts(y,z), Person(z,'Bob')
U(y) :- Person(y,'Alice'), not V(y) -- here's the set difference
Q(x) :- Trusts(x,y), U(y)
```

The query plan is:



73. *(from CSE 544p, Winter 2014, Final)*

A large software company stores information about its library functions in a database with the following schema:

```
Function(fid, fname, lang)
Call(fid1, fid2)
```

Both `Call.fid1` and `Call.fid2` are foreign keys to `Function`. The first table stores all function names and the language in which they were written (e.g. Java, or C#, or python etc). The second table stores which function calls what function.

(a) A software engineer needs to make some updates to the function `foo`. This will affect the behavior not only of the function `foo`, but of all functions that call `foo` directly or indirectly. Write a datalog program to compute all functions that might be affected by the change. Your program should return a set of `fid, fname` pairs.

Solution:

```
q(fid) :- function(fid, 'foo', -)
q(fid1) :- q(fid2), call(fid1,fid2)
answer(fid, fname) :- q(fid), function(fid, fname, -)
```

(b) The company considers Java a *supported* language and python a *non-supported* language; other languages do not have an assignation of supported or not supported. The company's policy is that no Java function may call, either directly or indirectly, a python function. Notice that a Java function may call functions in some language other than python, and that functions in other languages may call python functions. Write a datalog program to check for violations of the company's policy. Your program should return all pairs of functions `f1, f2` (id and name) that violate the policy.

Solution:

```
t(fid1,fid2) :- call(fid1,fid2)
t(fid, fid2) :- t(fid, fid1), call(fid1,fid2)
answer(fid1, fname1, fid2, fname2) :- t(fid1,fid2),
          function(fid1, fname1, 'Java');
          function(fid2, fname2, 'python');
```

(c) After a major revision, testers noticed the following unit test behavior:

```
function foo1  -- fail
function foo2  -- fail
function foo3  -- pass
```

The failures are likely to be caused by a bug introduced in a single function `f` in the revision. Write a stratified datalog$^{\neg}$ program that computes all candidate functions `f`: your program should return the ids and names of all functions `f` that are called directly or indirectly by `foo1` and `foo2`, but are not called by `foo3`.

Solution:

```
r1(f) :- function(f, 'foo1', -)
r1(f2) :- r1(f1),call(f1,f2)
r2(f) :- function(f, 'foo2', -)
r2(f2) :- r2(f1),call(f1,f2)
r3(f) :- function(f, 'foo2', -)
r3(f2) :- r3(f1),call(f1,f2)
answer(f, fname) :- r1(f), r2(f), not r3(f), function(f,fname,-)
```

Notice that `r1,r2` must be *intersected*, not *unioned*.

Intersection of `r1,r2` must be done *after* recursion.

74. *(from CSE 544p, Autumn 2015, Final)*

A social network has a collection users and of blog posts:

```
User(uid, name)
Blog(bid, author, text)
Follows(uid1, uid2)
```

Here `Blog.author`, `Follows.uid1` and `Follows.uid2` are foreign keys to `User`.

(a) The blog with `bid` = 359 contained incorrect information, and the site owners want to contact all direct and indirect followers of the blog's authors. Write a datalog program to find all followers of this blog's author. Your program should return a set of `uid, name` pairs.

Solution:

```
q(uid) :- Blog(359,uid, -)
q(p1) :- q(p2), Follows(p1, p2)
answer(p, name) :- q(p), User(p,name)
```

(b) One day an unusually large number of users have quit the social network: they simply canceled their accounts. You suspect that the reason they left is because they all have read the same offensive blog, posted by somebody whom they all follow. Given the relation `Left(uid)` of users who left that day, write a stratified datalog program that finds all blogs authored by somebody who is followed (directly or indirectly) by *all* users in `Left(uid)`. Your program should return a set of `bid`'s.

Solution:

```
f(u,v) :- Follows(u,v)
f(u,v) :- f(u,w), Follows(w,v)

nonAnswer(v) :- Left(u), User(v,-), not f(u,v)

Answer(bid) :- User(u), not nonAnswer(u), Blog(bid, u, -)
```

75. *(from CSE 344, Autumn 2017, Midterm)*

Consider again the schema:

```
Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)
```

Answer the questions below.

(a) Write a datalog program that returns all employees that worked *only* on the `'Compiler'` project. Your query should return the `eid` and `name` of each such employee.

Solution:

```
nonAnswer(eid) :- Employee(eid, -, -),
                  WorksOn(eid, pid, -),
                  Project(pid, t, -),
                  t != 'Compiler'
Answer(eid, name) :- Employee(eid, name, -), !nonAnswer(eid)
```

(b) A project p_1 *influences* a project p_2 if there exists an employee who worked on p_1 during some year, then worked on p_2 during some later year. After a major bug was discovered in several projects, the company traced it down to a design flaw in the `'Compiler'` project, and now wants to retain only the projects that were not influenced by `'Compiler'`. (Note: `'Compiler'` is influenced by `'Compiler'`.) Write a datalog program to find all projects that were not influenced by the `'Compiler'` project; return their `pid` and `title`.

Solution:

```
Infl(x) :- Project(x, 'Compiler', -)
Infl(z) :- Infl(x),
           WorksOn(eid, x, y1),
           WorksOn(eid, z, y2),
           y1 < y2
Answ(pid, title) :- Project(pid, title, -), !Infl(pid)
```

76. *(from CSE 344, Winter 2019, Final)*

In the kingdom of Datalandia there are nobles and commoners. When the kingdom was founded hundreds of years ago, the first king knighted some people who became the first noblemen, called *UrNobles*. The rule of the land is that a newborn becomes a noble if both his/her parents are nobles; otherwise he/she is a commoner. The kingdom meticulously maintains a database of all its subjects:

```
Person(pid,name); // every person in Datalandia who ever lived:
Father(fid,pid);  // fid is the father of pid
Mother(mid,pid);  // mid is the mother of pid
UrNoble(nid);     // the person ID's of the UrNobles
King(kid);        // the person ID's of all kings
```

(a) Write a datalog query that returns the `pid`'s and names of all nobles.

Solution:

```
Noble(pid) :- UrNoble(pid)
Noble(pid) :- Nobel(fid),Nobel(mid),Father(fid,pid),Mother(fid,pid)
```

(b) Unfortunately, something terribly wrong happened, and some of the kings of Datalandia were commoners. Write a datalog program to retrieve all the commoner

kings of Datalandia. You should return a set of person IDs (`pid`) and names. You may use the datalog query written for the previous question.

Solution:

Solution: CommonKing(kid :- King(kid), not Noble(pid)

77. *(from CSE 344, Winter 2019, Midterm)*

Consider a database storing the employees of a company:

`Employee(eid,name,office,mid)`

The attributes `eid`, `name`, and `office` are the Employee's ID, name, and office, while `mid` is the Manager's ID. The CEO of the company is called `''Smith''`. Every employee has a manager, except for the CEO, who is managing himself (`eid` = `mid`). An *indirect manager* of an employee is either her manager, or her manager's manager, etc, up to the CEO.

Answer the questions below.

(a) Write a datalog program that returns the Employee ID's and names of all employees managed directly or indirectly by `''Alice''`.

Solution:
Compute the ID and name at the same time:

```
Q(e,n) :- Employee(e,n,_,m), Employee(m,"Alice",_,_)
Q(e,n) :- Employee(e,n,_,m), Q(m,_)
```

Better: compute only the ID's, later fetch the names:

```
M(e) :- Employee(e,_,_,m), Employee(m,"Alice",_,_)
M(e) :- Employee(e,_,_,m), M(m)
Q(e,n) :- M(e), Employee(e,n,_,_)
```

(b) The company's requirement is that every employee should be managed directly or indirectly by the CEO, `''Smith''`. However, the database got corrupted and this requirement became violated. Write a datalog program to find all employees that are not managed directly or indirectly by `''Smith''`. Your program should return their Employee ID and name.

Solution: First compute `M(e)` above, then:

```
Q(e,n) :- Employee(e,n,_,_),!M(e)
```

most common mistakes:
missing recursion (-5)
using 'smith' as an id (-2)

(c) Two managers are called *relatives* if they manage (directly) two employees sharing the same office, or if they are relatives of some other common manager (i.e. relatives by transitivity). For example if `Alice` and `Bob` share the same office, then their direct managers `Carol` and `David` are relatives; furthermore, if `David` is also a relative of `Eve`, then `Carol` and `Eve` are relatives by transitivity. Write a datalog program to find all relatives of `Carol`. Your program should return their Employee ID's and names.

Solution:

```
R(e1,e2) :- Employee(m1,_,o,e1),Employee(m2,_,o,e2)
R(e1,e3) :- R(e1,e2),R(e2,e3)
Q(e,n) :- Employee(ec,"Carol",_,_), R(ec,e),Employee(e,n,_,_)
```

78. *(from CSE 414, Spring 2019, Final)*

At Gotham University[3] each course has two prerequisites: for some courses both prerequisites are required, for other courses only one of the two prerequisites is required. The only exceptions are introductory courses, which don't have prerequisites. Students can only take one course every quarter.

The schema is:

```
Course(cid, name, noQuarters)
NoPrereq(cid)
PrereqOneOfTwo(cid, cid1, cid2)
PrereqTwoOfTwo(cid, cid1, cid2)
```

(a) The university requires that every course appear under either `NoPrereq` or under `PrereqOneOfTwo` or under `PrereqTwoOfTwo`. Write a datalog program that checks this constraint. Your program should return the `cid`'s and `name`'s of courses that do not occur in `NoPrereq` or `PrereqOneOfTwo` or `PrereqTwoOfTwo`.

Write your answer below:

Solution:

```
Listed(cid) :- NoPrereq(cid)
Listed(cid) :- PrereqOneOfTwo(cid, -, -)
Listed(cid) :- PrereqTwoOfTwo(cid, -, -)
Answer(cid,name :- Course(cid, name, -), not Listed(cid)
```

(b) Write a datalog program that computes, for each course, the smallest number of quarters need for a student to take sufficient prerequisites for that course, and the course itself. Your query should return pairs `oid`, `number-of-quarters`. For example, if the database is the following:

[3]The *Wise Men of Gotham* are supposed to have feigned idiocy to avoid a Royal visit by King John. Gotham is also a fictional city in the comics of Batman.

Course:

cid	name	no Quarters
Math101	Math	3
Java102	Java	2
DB414	DB	3
ML446	ML	5

NoPrereq

cid
Math101
Java102

PrereqOneOfTwo

cid	cid1	cid2
DB414	Math101	Java102

PrereqTwoOfTwo

cid	cid1	cid2
ML446	Math101	Java102

then your answer should be:

cid	c	
Math101	3	– because has no prereq
Java102	2	– because has no prereq
DB414	5	– can take either Math101 (3qtr) or Java102 (2qtr) plus DB414 (3qtr)
ML446	10	– must take Math101 (3qtr) then Java102 (2qtr) plus ML446 (5qtr)

Hint if you were to compute the number of quarters that a student needs to take "Math101" and "Java102", which don't have any prerequisites, you could write:

```
Q(c1+c2) :- Course("Math101", -, c1), Course("Java102", -, c2)
```

Write your answer below:

Solution:

```
Q(cid, q) :- NoPrereq(cid), Course(cid, -, q)
Q(cid, q+q1) :- PrereqOneOfTwo(cid, cid1, cid2),
                Q(cid1,q1), Q(cid2,q2), q1<q2,
                Course(cid, -, q)
Q(cid, q+q2) :- PrereqOneOfTwo(cid, cid1, cid2),
                Q(cid1,q1), Q(cid2,q2), q1>=q2,
                Course(cid, -, q)
Q(cid, q+q1+q2) :- PrereqTwoOfTwo(cid, cid1, cid2),
                   Q(cid1,q1), Q(cid2,q2),
                   Course(cid, -, q)
```

79. *(from CSE 414, Spring 2019, Midterm)*

Consider the same schema as before:

```
Car(lp, make, owner)
Downtown(lp, day)
```

Answer the questions below.

(a) Write a datalog program that returns all owners who own both a Honda and a Ford.

Solution:

```
Q(x) :- Car(_, 'Honda',x), Car(_, 'Ford',x)
```

(b) Write a datalog program that returns all owners who own a Honda and do not own a Ford.

Solution:

```
HasFord(x) :- Car(_, 'Ford', x)
Q(x)       :- Car(_, 'Honda', x), !HasFord(x)
```

(c) A spy ring has infiltrated Nullville, and you are a counter intelligence officer charged with catching them. After lots of hard work you have found one suspected spy: Alice. To find more suspects, you make the following judgment. You know that some days the spies have secret meetings downtown, and when they meet downtown then they always drive Bentleys. You reason that, whenever you have found a suspect, if she/he drives a Bentley downtown one day, then every other person who drives a Bentley downtown the same day automatically becomes a suspect too. Write a datalog program to compute all suspects. (Hint: your first rule should be this single fact: `Suspect('Alice').`)

Solution:

```
Suspect('Alice').
Suspect(Y) :- Suspect(X),
              Car(lpx,'Bentley',X), Car(lpy,'Bentley',Y),
              Downtown(lpx,d), Downtown(lpy,d)
```

(d) Answer the questions below:

i. Is this datalog rule safe?
`Q(u) :- Car(x,'Honda',u), !Downtown(x,2000)`
Safe or Unsafe?

Solution: Safe

ii. Is this datalog rule safe?
`Q(u) :- Car(x,'Honda',u), !Downtown(x,y)`
Safe or Unsafe?

Solution: Unsafe

iii. Is this datalog rule safe?
`Q(u) :- Car(x,'Honda',u), Downtown(x,d), !Car(y,'Ford',u), Downtown(y,d)`
Safe or Unsafe?

Solution: Safe

iv. Is this datalog rule safe?
`Q(u) :- Car(x,'Honda',u), Downtown(x,d), Car(y,'Ford',u), !Downtown(y,d)`
Safe or Unsafe?

Solution: Safe

v. Is this datalog rule safe?
```
Q(u) :- Car(x,'Honda',u), !Downtown(x,d), Car(y,'Ford',u), Downtown(y,d)
```
Safe or Unsafe?

Solution: Safe

80. *(from CSE 544p, Spring 2021, Final)*

A social network has a collection users and of blog posts:

```
User(uid, name)
Blog(bid, author, text)
Follows(uid1, uid2)
```

Here `Blog.author`, `Follows.uid1` and `Follows.uid2` are foreign keys to `User`.

(a) The blog with `bid` = 359 contained incorrect information, and the site owners want to contact all direct and indirect followers of the blog's authors. Write a datalog program to find all users who the site owner needs to contact. Your program should return a set of `uid, name` pairs.

Solution:
```
q(uid) :- Blog(359,uid, -)
q(p1) :- q(p2), Follows(p1, p2)
answer(p, name) :- q(p), User(p,name)
```

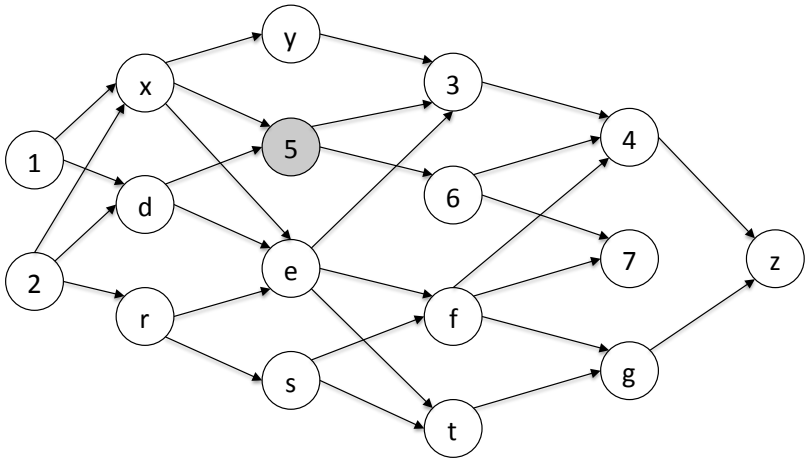
(b) One blogger has offended many of his followers, and the site owner was suddenly faced with a large number of requests for account cancellations, all coming from direct or indirect followers of the offensive blogger. The predicate `Cancel(uid)` contains all cancellation requests. Write a datalog program to find the offensive blogger(s). Your program should return the `uid` and `name` (zero or more) of the bloggers with the property that *all* users in `Cancel` are direct or indirect followers of that blogger.

Solution: Let $F(u, v)$ be the transitive closure of Follows:
```
F(u,v) :- Follows(u,v)
F(u,v) :- F(u,w), Follows(w,v)
nonAnswer(u) :- Cancel(v), User(u,-), not F(v,u)
Answer(u,n) :- User(u,n), not nonAnswer(u)
```

(c) Consider the datalog program below:
```
T(x,y) :- R(x,z),R(y,z)
T(x,y) :- T(x,z),T(z,y)
```

```
Q(w)   :- T(5,w)
```

Show the output of the IDB predicate Q when the program is run on the relation R given by the graph below; for example, the edge $2 \to d$ denotes the tuple $(2, d) \in R$. You do not need to justify your answer.



Solution: $5, y, e, s$

8 Conceptual Design and E/R Diagrams (81–100)

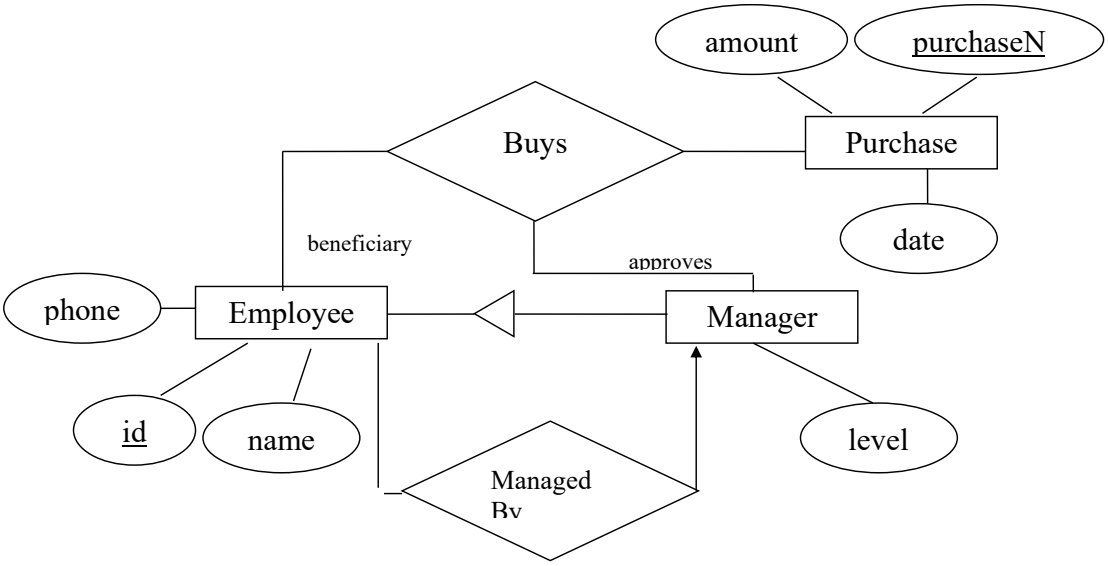
Conceptual design comes before schema design: an E/R diagram records what entities exist and how they relate, without committing to tables. The questions in this section run in both directions. Some give a description of an application and ask you to draw the diagram; others give a diagram and ask what it asserts, or ask you to translate it into relations.

The details that carry the most weight are the multiplicity of relationships (one-to-many or many-to-many), the treatment of weak entities (notice that we did not cover weak entities in every quarter), and the question of which constraints survive translation into a relational schema. A recurring exam question asks for a constraint the diagram *cannot* express, and the answer is usually a constraint that spans more than one relationship.

As with database design, more than one diagram can be correct; the solutions give one good answer.

81. *(from CSE 444, Autumn 2000, Final)*

Consider the E/R diagram below:



(a) Considering the `Buys` relationship, we know that each purchase is for the benefit of a single employee, and is approved by a single manager. Add arrows to the diagram to indicate this situation. Also, indicate whether the arrows capture this information faithfully.

Solution: We add two arrows, from `Buys` to `Employee` and from `Buys` to `Manager`. The representation is not faithful.

(b) Continue to assume the constraint at point (a). Create a relational schema to represent this data by giving the SQL statements for creating the tables. The statements should include all primary and foreign keys. The following information is known about the attribute domains: `id` consists of 10 characters, `phone` has 10 characters, `name` has up to 30 characters, `level` is an integer, `amount` is an integer, and `purchaseNo` has 20 characters.

Solution:

```
CREATE TABLE Employee (
       id CHAR(10) PRIMARY KEY,
       name VARCHAR(30),
       phone CHAR(10),
       ManagedBy CHAR(10) REFERENCES Manager(id) );
```

```
CREATE TABLE Manager (
       id CHAR(10) PRIMARY KEY REFERENCES Employee(id),
       level INT );

CREATE TABLE Purchase (
       purchaseNo CHAR(20) PRIMARY KEY,
       date DATE,
       amount INT );

/* Buys could be integrated into Purchase */
CREATE TABLE Buys (
       purchase CHAR(20) PRIMARY KEY REFERENCES Purchase(purchaseNo),
       beneficiary CHAR(10) REFERENCES Employee(id),
       approves CHAR(10) REFERENCES Manager(id) );
```

Note: the script above will not run on SQL Server because of the cyclic reference between `Employee` and `Manager`; there the foreign key `ManagedBy` must be added afterwards with `ALTER TABLE`. Also `level` is a keyword.

(c) Write SQL queries to compute the following:

i. Employee Smith is a high spender. For each manager, compute the total amount of all purchases approved by that manager for the benefit of Smith. Your query should print the manager id, name and the total amount of purchases approved.

Solution:

```
SELECT m.id, em.name, sum(p.amount)
FROM   Manager m, Employee em, Employee e, Buys b, Purchase p
WHERE  m.id = em.id AND e.name = 'Smith'
   AND b.purchase = p.purchaseNo
   AND b.beneficiary = e.id AND b.approves = m.id
GROUP BY m.id, em.name
```

ii. Find all purchases in which the approving manager and the beneficiary employee are the same person. List the purchase number, the purchase amount, and the name of the employee (manager).

Solution:

```
SELECT DISTINCT p.purchaseNo, p.amount, e.name
FROM   Purchase p, Buys b, Manager m, Employee e
WHERE  p.purchaseNo = b.purchase AND m.id = b.approves
   AND e.id = b.beneficiary AND m.id = e.id
```

(d) John, a new database administrator, had a bad day: he accidentally deleted all tuples in the `Manager` table. Help John recover the lost data. Specifically, write a sequence of SQL statements that will populate the `Manager` table by using other information in the database. The following information about the company helps you recover the data:

- Each manager manages at least one employee.
- The level of a manager is defined as follows. Managers that manage only regular

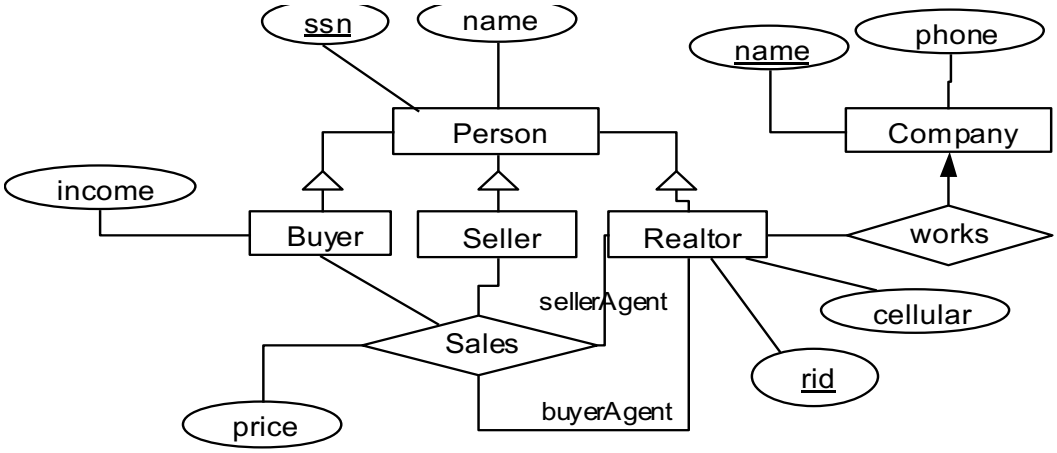
employees (i.e. that are not managers) have `level=1`. Managers that manage only regular employees and managers of level 1 have `level=2`, etc. John knows that the highest manager in the company is the CEO and his level is 4.

Solution: Run the following SQL statements in sequence:

```
INSERT INTO Manager(id, level)
      SELECT DISTINCT managedBy AS id, 1 AS level
      FROM   Employee
      WHERE  managedBy IS NOT NULL;

UPDATE Manager SET level = 2 WHERE id IN
      (SELECT m2.id
       FROM Employee e, Manager m1, Manager m2
       WHERE e.id = m1.id AND e.managedBy = m2.id);

UPDATE Manager SET level = 3 WHERE id IN
      (SELECT m3.id
       FROM Employee e, Manager m2, Manager m3
       WHERE e.id = m2.id AND e.managedBy = m3.id AND m2.level = 2);

UPDATE Manager SET level = 4 WHERE id IN
      (SELECT m4.id
       FROM Employee e, Manager m3, Manager m4
       WHERE e.id = m3.id AND e.managedBy = m4.id AND m3.level = 3);
```

82. *(from CSE 444, Autumn 2000, Midterm)*

(a) Design a relational schema for the E/R diagram below.

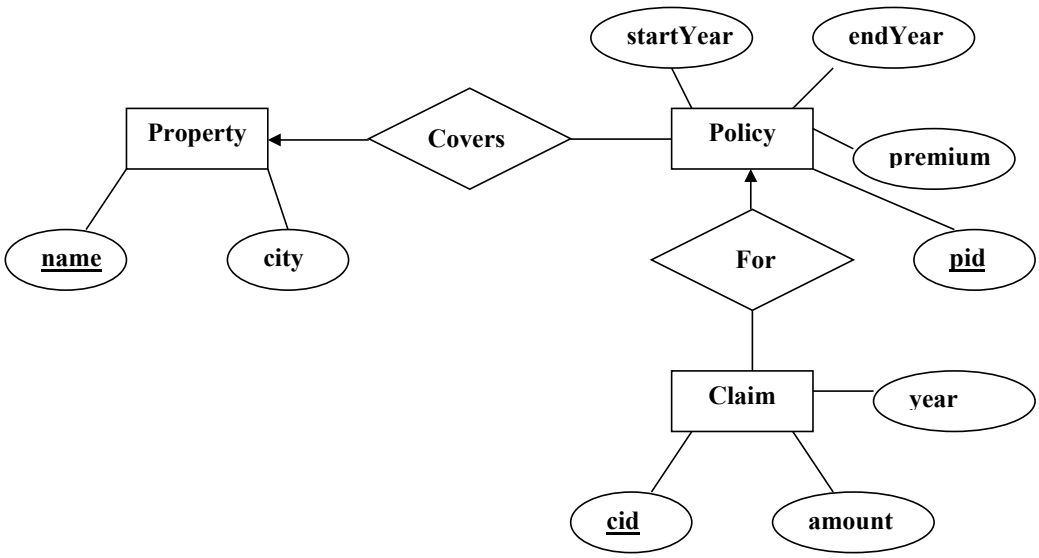


Solution:

```
Person(ssn, name)
Buyer(ssn, income)
Seller(ssn)
Realtor(ssn, rid, cellular, companyName)
Sales(buyerSsn, sellerSsn, buyerAgentRid, sellerAgentRid, price)
Company(name, phone)
```

83. *(from CSE 444, Autumn 2001, Final)*

An insurance company maintains a database whose schema is given by the E/R diagram below. Here `premium` refers to the annual premium (what the client pays annually to the insurance company) and is the same every year; `amount` is a one-time payment (which the insurance company has to pay to the client).



(a) Write SQL statements for creating the tables for the E/R diagram. Indicate keys and foreign keys. Assume the following domains: all keys (`name`, `pid`, `cid`) are fixed-length, 10 characters long; `city` is a variable length character field, max 30; all others (years, amounts) are integers.

Solution:

```
CREATE TABLE Property (
   name CHAR(10) PRIMARY KEY,
   city VARCHAR(30) );

CREATE TABLE Policy (
   pid CHAR(10) PRIMARY KEY,
   premium INT,
   startyear INT,
   endyear INT,
   covers CHAR(10) REFERENCES Property(name) );

CREATE TABLE Claim (
   cid CHAR(10) PRIMARY KEY,
   amount INT,
   year INT,
   for CHAR(10) REFERENCES Policy(pid) );
```

(b) An insurance agent processes the claim with `cid=03492` for a client. He would like to find out if there were any other claims for the same property submitted during

the same year. Write a SQL query to find this out: your query should return all `cid`'s of such claims.

Solution:

```
SELECT DISTINCT x1.cid
FROM Claim x1, Policy y1, Property p, Claim x2, Policy y2
WHERE x1.for = y1.pid AND y1.covers = p.name AND
      p.name = y2.covers AND y2.pid = x2.for AND
      x2.cid = "03492" AND x1.year = x2.year
```

(c) Write a SQL query that returns, for each year when some claim was filed, the number of claims and the total dollar amount for that year.

Solution:

```
SELECT year, count(*), sum(amount)
FROM   Claim
GROUP BY year
```

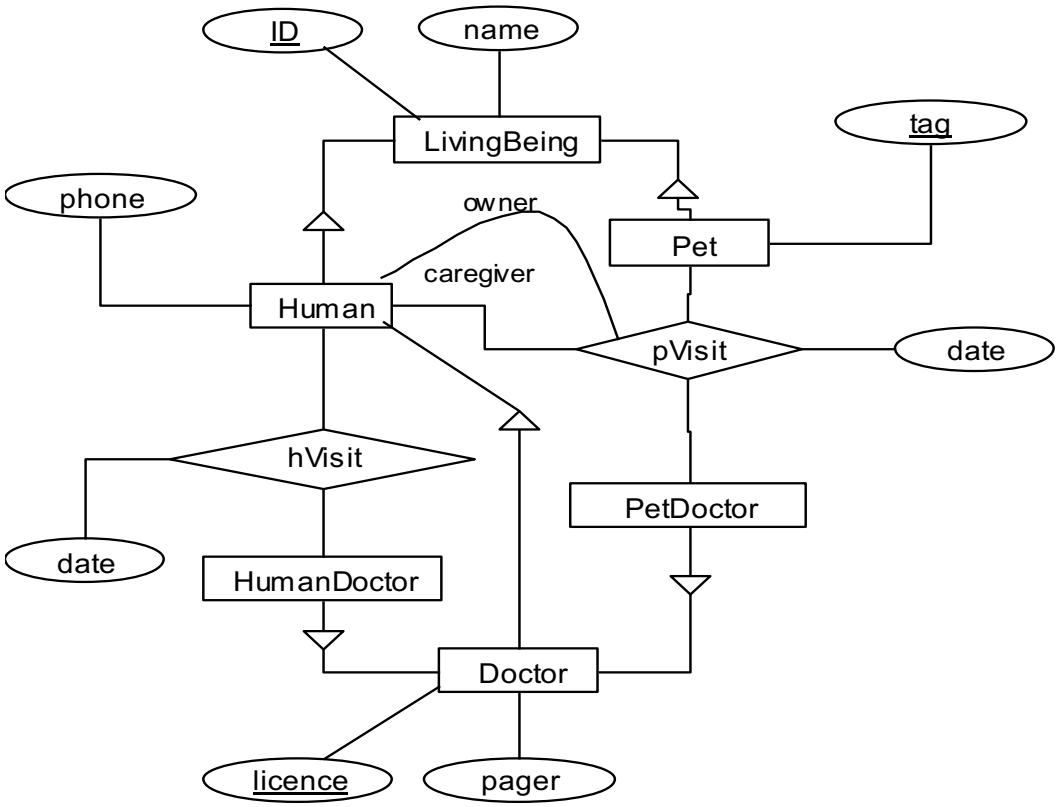
(d) A "bad" policy is defined to be a policy for which there exists a period of ten continuous years in which the total amount of claims exceeded the premium over those ten years. Write a SQL query to retrieve all "bad" policies.

Solution:

```
SELECT DISTINCT Policy.pid
FROM Policy, Claim x, Claim y
WHERE x.year <= y.year AND y.year - x.year < 10
  AND x.for = Policy.pid AND y.for = Policy.pid
GROUP BY Policy.pid, x.year
HAVING sum(y.amount) > 10*Policy.premium
```

84. *(from CSE 444, Autumn 2001, Midterm)*

Consider the E/R diagram below.



(a) Design a relational schema for it, and underline all keys in every table.

Solution:

```
LivingBeing(id, name)
Human(lbID, phone)
Pet(lbID, tag)
Doctor(lbID, licence, pager)
HumanDoctor(licence)
PetDoctor(licence)
hVisit(lbID, licence, date)
pVisit(tag, ownerLbID, caregiverLbID, licence, date)
```

(b) Write the SQL statements for creating these tables, and include statements for primary keys, foreign keys, and unique attributes. Choose any reasonable domains for the attributes.

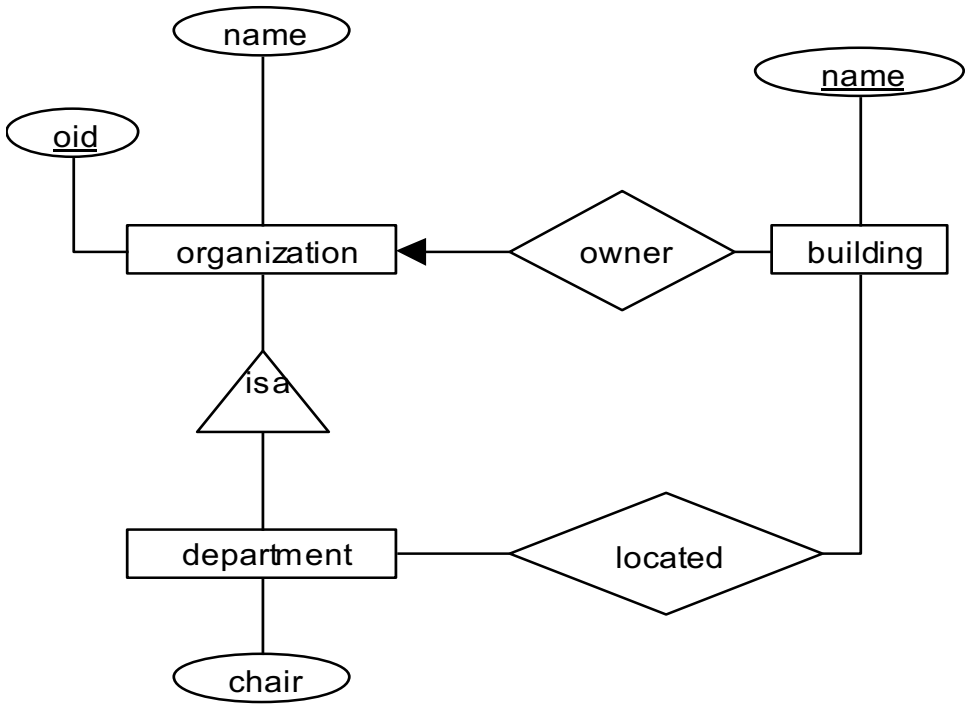
Solution:

```
CREATE TABLE LivingBeing (id CHAR(10) PRIMARY KEY,
                          name CHAR(50) );
CREATE TABLE Human (lbID CHAR(10) PRIMARY KEY,
                    phone CHAR(20),
                    FOREIGN KEY (lbID) REFERENCES LivingBeing(id) );
CREATE TABLE Pet (lbID CHAR(10) UNIQUE,
                  tag CHAR(30) PRIMARY KEY,
                  FOREIGN KEY (lbID) REFERENCES LivingBeing(id) );
```

```
CREATE TABLE Doctor (licence CHAR(20) PRIMARY KEY,
                     lbID CHAR(10) UNIQUE,
                     FOREIGN KEY (lbID) REFERENCES Human(lbID) );
CREATE TABLE HumanDoctor (licence CHAR(20) PRIMARY KEY,
                          FOREIGN KEY (licence) REFERENCES Doctor(licence) );
CREATE TABLE PetDoctor (licence CHAR(20) PRIMARY KEY,
                        FOREIGN KEY (licence) REFERENCES Doctor(licence) );
CREATE TABLE hVisit (lbID CHAR(10) REFERENCES Human(lbID),
                     licence CHAR(20) REFERENCES HumanDoctor(licence),
                     date DATE );
CREATE TABLE pVisit (tag CHAR(30) REFERENCES Pet(tag),
                     owner CHAR(10) REFERENCES Human(lbID),
                     caregiver CHAR(10) REFERENCES Human(lbID),
                     licence CHAR(20) REFERENCES PetDoctor(licence),
                     date DATE );
```

85. *(from CSE 444, Autumn 2002, Midterm)*

Consider the E/R diagram below:



(a) Design a relational schema for it, and underline all keys in every table.

Solution:

```
Organization(oid, name)
Building(name, oid)
Department(oid, chair)
Located(oid, bname)
```

(b) Write the SQL statements for creating these tables, and include statements for primary keys, foreign keys, and unique attributes. Choose any reasonable domains for the attributes.

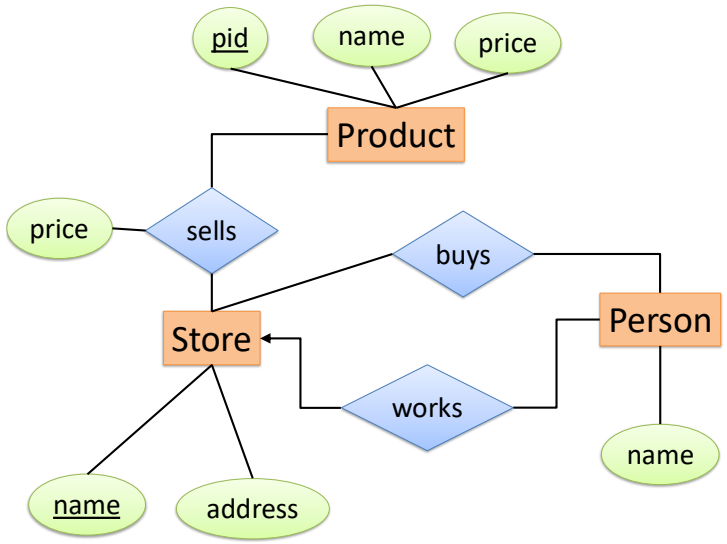
Solution:

```
CREATE TABLE Organization (oid varchar(20) PRIMARY KEY,
                           name varchar(60) NOT NULL);

CREATE TABLE Building (name varchar(25) PRIMARY KEY,
                       oid varchar(20) REFERENCES Organization(oid));

CREATE TABLE Department (oid varchar(10) PRIMARY KEY
                             REFERENCES Organization(oid),
                         chair varchar(30));

CREATE TABLE Located (oid varchar(20) REFERENCES Department(oid),
                      bname varchar(25) REFERENCES Building(name),
                      PRIMARY KEY(oid, bname));
```

86. *(from CSE 444, Autumn 2003, Final)*

(a) Design an E/R diagram to model stores, products, and people. Each store has a name and an address. Each product has a product identifier (`pid`), a name, and a suggested price. In addition model the following relationships. Stores sell products with a given price (that is, the same product may be sold at different stores with different prices). People work for stores, and people buy at stores. Finally, represent the fact that each person may work for at most one store.

Solution:



(b) We say a set of attributes X is *closed* (with respect to a given set of functional dependencies) if $X^+ = X$. Given the closed attribute sets, this gives us some

information on the underlying functional dependencies. Consider a relation with schema $R(A, B, C, D)$ and an unknown set of functional dependencies. For each of the statements below, give a set of functional dependencies for which that statement holds.

i. All sets of attributes are closed.

> **Solution:** The empty set of functional dependencies (i.e. only trivial ones).

ii. The only closed sets are $\emptyset$ and $\{A, B, C, D\}$.

> **Solution:** $A \to B$, $B \to C$, $C \to D$, $D \to A$.

iii. The only closed sets are $\emptyset$, $\{A, B\}$, and $\{A, B, C, D\}$.

> **Solution:** $A \to B$, $B \to A$, $C \to A$, $D \to A$.

87. *(from CSE 444, Autumn 2003, Midterm)*

(a) Design an E/R diagram for geography that contains the following kinds of objects together with the listed attributes:

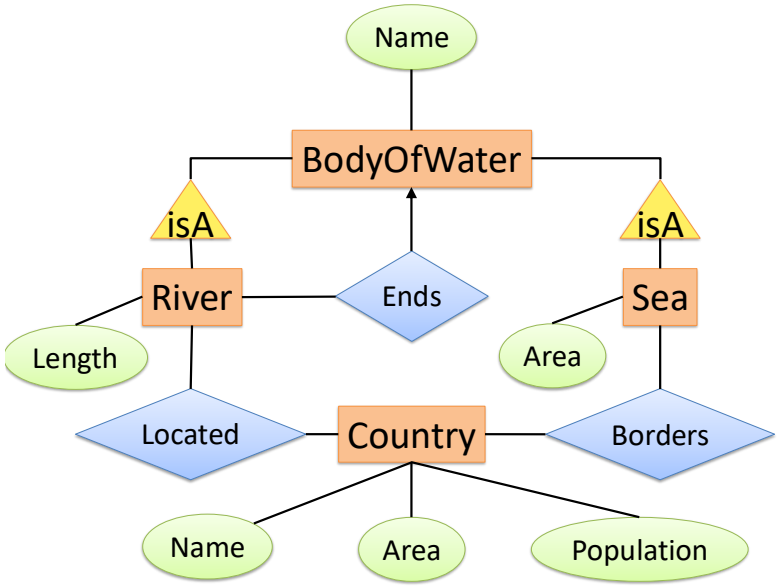
- **Countries**: name, area, population
- **Rivers**: name, length
- **Seas**: name, area

Model the following relationships between the geographical objects:

- Each river is located in one or several countries.
- Each sea borders one or several countries.
- Each river ends in either another river or in a sea.

You have to turn in only the E/R diagram.

Solution:



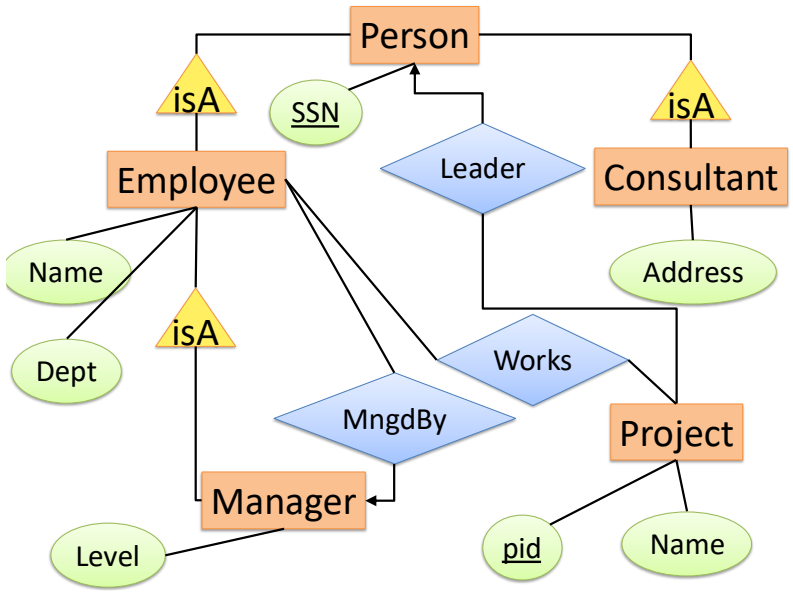
88. *(from CSE 444, Autumn 2004, Midterm; CSE 444, Winter 2006, Midterm; CSE 444, Autumn 2005, Midterm)*

(a) Draw an E/R diagram describing the following domain:

- **Employees.** Attributes: `ssn`, `name`, `department`
- **Managers** are Employees. Attributes: `level`
- **Consultants.** Attributes: `ssn`, `address`
- **Projects.** Attributes: `pid` (key), `name`
- Every Employee is managed-by (at most one) Manager.
- Employees work on Projects: an employee may work for an arbitrary number of projects, and an arbitrary number of employees may work for a project.
- Every project has exactly one leader, who is either an employee or a consultant.

Your answer should consist of an E/R diagram, which includes entity sets, attributes, relationships, and ISA relations. Indicate the type of each relationship with appropriate arrows (one-one, one-many, or many-many).

Solution:



89. *(from CSE 444, Autumn 2006, Midterm)*

(a) Design an E/R diagram for an application domain consisting of the following entity sets:

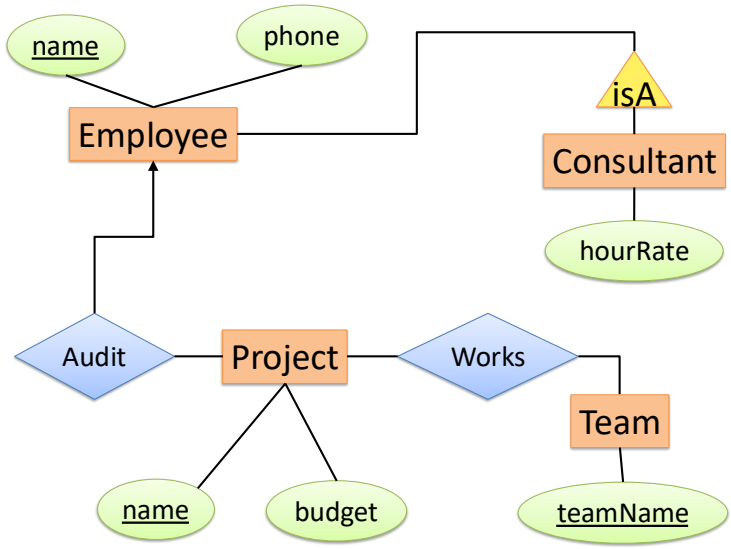
- **Projects.** Attributes: `name`, `budget`
- **Teams.** Attributes: `team_name`
- **Employees.** Attributes: `name`, `phone_number`
- **Consultants.** Attributes: `name`, `phone_number`, `hourly_rate`

and the following relationships:

- Each team works on one or more projects.
- Each project has an auditor, who is an employee.
- Consultants are employees.

Your answer should consist of an E/R diagram with entity sets, attributes (make sure you create appropriate keys; you may introduce new attributes if needed), relationships, and inheritance.

Solution:



(b) Create in SQL the tables for the E/R diagram in the previous point. All your attributes should be of type `text`, except for `budget` and `hourly_rate`, which are integers. (Pick appropriate types for the key attributes.) Indicate all keys and all foreign keys. You have to turn in several `CREATE TABLE` statements.

Solution:

```
drop table if exists works;
drop table if exists team;
drop table if exists project;
drop table if exists consultant;
drop table if exists employee;
create table employee(name text primary key, phone text);
create table consultant(name text primary key references employee);
create table project(name text primary key,
                     budget int,
                     audit text references employee);
create table team (teamName text primary key);
create table works (name text references project,
                    teamName text references team);
```

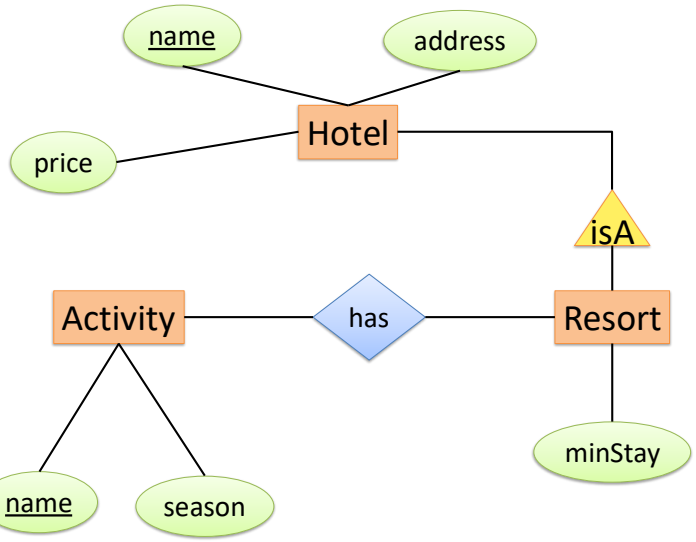
90. *(from CSE 444, Spring 2007, Midterm)*

Consider an application that needs to manage data for a travel agency. It needs to store the following entities and relationships:

- **Hotels:** have attributes name, address, price
- **Resorts:** are Hotels, that also have an attribute minimum-stay
- **Activities:** have attributes name, season
- **Has:** is a relationship between Resorts and Activities

(a) Design an E/R diagram for this application.

Solution:



(b) Write the `CREATE TABLE` statements for creating the SQL tables. You may choose very simple atomic datatypes for the attributes. Indicate all keys and foreign keys.

Solution:

```
drop table if exists has;
drop table if exists activity;
drop table if exists resort;
drop table if exists hotel;
create table hotel (name text primary key,
                    address text,
                    prince int);
create table resort (name text primary key references hotel,
                     minStay int);
create table activity (name text primary key,
                       season text);
create table has (nameActivity text references activity,
                  nameResort text references resort);
```

91. *(from CSE 444, Autumn 2008, Midterm)*

(a) Design an E/R diagram for a database of documents, authors, and readers. The database has the following entities and relationships:

- **Document**: has a **docID** (the key), **title**.
- **Section**: each document consists of several sections; each section has a **title**, and a **number** (1, 2, 3, ...). The section number is unique within each document.
- **Author**: each author is the creator of several documents. Each document was created by exactly one author.
- **Reader**: each reader reads several documents, and each document is read by several readers.
- **Person**: has **name**, **address**. Authors and readers are persons.

You have to turn in an E/R diagram.

Solution: Solution not provided. It requires a "weak entity set" which was not covered in every quarter.

(b) Database normalization. Consider a relation $R(A, B, C, D, E, F, G)$ with the following functional dependencies:

$$\begin{aligned} A &\rightarrow B \\ C &\rightarrow AD \\ CE &\rightarrow B \\ EF &\rightarrow C \end{aligned}$$

Compute the Boyce-Codd Normal Form (BCNF) decomposition of R. Indicate each step you make in your computation, by showing the relation to which you apply the step and the violation of BCNF that you use during that decomposition step. Indicate clearly your end result: the relations, their attributes, and their keys.

Solution: Decompose $R(A, B, C, D, E, F, G)$. Try $A^+ = AB$. Decompose into $R_1(A, B)$, $R_2(A, C, D, E, F, G)$.

Decompose $R_2(A, C, D, E, F, G)$. Try $C^+ = ACD$. Decompose into $R_3(A, C, D)$, $R_4(C, E, F, G)$.

Decompose $R_4(C, E, F, G)$. Try $EF^+ = CEF$. Decompose into $R_5(C, E, F)$ and $R_6(E, F, G)$.

End result: $R_1(A, B)$ with key A; $R_3(A, C, D)$ with key C; $R_5(C, E, F)$ with key EF; and $R_6(E, F, G)$ with key EFG.

(c) Consider a relation $R(A, B, C, D, E)$. A set of attributes X is called *closed* if $X^+ = X$. Give an example of functional dependencies on R such that the only closed sets are A, B, AC, BD, ABE.

Solution: Write $X \leftrightarrow Y$ to mean $X \to Y$ and $Y \to X$.

$$C \to A, \quad D \to B, \quad AB \leftrightarrow E$$

together with one of $CB \to D$, $CBA \to D$, $CBAE \to D$, and one of $AD \to C$, $ABD \to C$, $ABDE \to C$.

92. *(from CSE 444, Spring 2010, Final)*

(a) Decompose in BCNF the relation $R(A, B, C, D, E)$ that satisfies the following functional dependencies. Show your steps, and show the keys in the decomposed relations.

$$\begin{aligned} A &\to B \\ CD &\to E \end{aligned}$$

Solution:

$$A+ = AB$$

Decompose into $R1 = \underline{A}B$, $R2 = ACDE$ Continue with

$$CD+ = CDE$$

Decompose $R2$ into $R3 = \underline{CD}E$ and $R4 = \underline{CDA}$

(b) For each of the statements below, indicate whether they are true or false. You do not need to justify your answers.

- Every relation with only two attributes is in BCNF.

  Solution: true

- If X and Y are super-keys then $X \cup Y$ is also a super-key.

  Solution: true

- If X and Y are super-keys then $X \cap Y$ is also a super-key.

Solution: false. Example: $AB \to C, AC \to B$. Both AB, AC are keys (and also super-keys) but their intersection $AB \cap AC = A$ does not determine anything and is not a super-key.

- If $X \to A$ and $Y \to A$, then $(X \cap Y) \to A$.

Solution: false. Example: $BC \to A, BD \to A$ but the intersection $BC \cap BD = B$ and $B \to A$ does not hold.

- If $X^+ = X$ and $Y^+ = Y$ then $(X \cap Y)^+ = (X \cap Y)$.

Solution: true.

93. *(from CSE 444, Spring 2010, Makeup Midterm; CSE 444, Spring 2010, Midterm)*

Consider a database with the following entity sets:

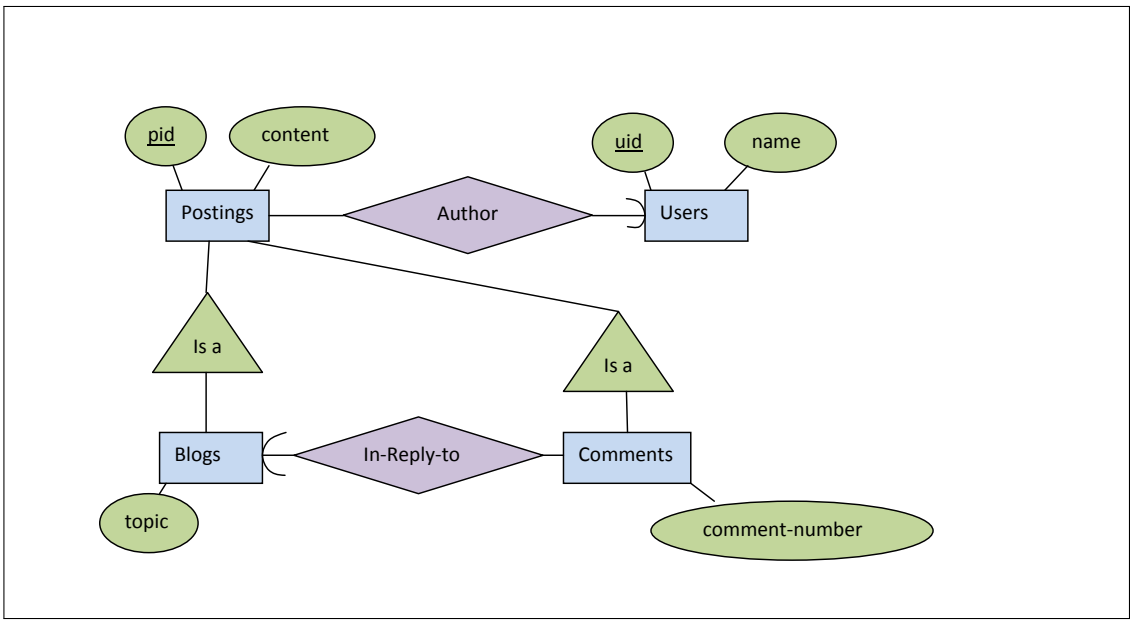
- `Users`: have an `uid` (10 characters) and `name` (up to 20 characters). `uid` is the key for `Users`.
- `Postings`: have an `pid` (10 characters) and `content` (text of 100 characters). `pid` is the key for `Postings`.
- `Blogs`: every blog is a `Posting` and has a `topic` attribute (up to 30 characters).
- `Comments`: every comment is a `Posting` and has a `comment-number` (an integer).

and the following relationships:

- There is an `Author` relationship connecting `Posting` to `User`.
- There is a `In-reply-to` relationship connecting `Comment` to `Blogs`.

(a) Draw the E/R diagram for the database schema.

Solution:



(b) Write the SQL statements for creating the tables that represent your E/R diagram. Indicate all keys and foreign keys.

Solution:

```
create table Users (
   uid char(10) primary key,
   name varchar(20)
)
create table Postings (
   pid char(10) primary key,
   uid char(10) not null references Users,
   content varchar(100)
)
create table Blogs (
    pid char(10) primary key references Postings,
    topic varchar(30)
)
create table Comments (
    pid char(10) primary key references Postings,
    comment-number int,
 bid char(10) not null references Blogs(pid)
}
```

94. *(from CSE 444, Spring 2010, Makeup Midterm; CSE 444, Spring 2010, Midterm)*

(a) Consider a relation with attributes $R(A, B, C, D, E)$ that satisfies the following functional dependencies:

$$\begin{aligned} AC &\rightarrow B \\ BD &\rightarrow C \\ CE &\rightarrow D \\ DA &\rightarrow E \\ EB &\rightarrow A \end{aligned}$$

Find all the keys that contain the attribute A. Your answer should include only the keys that contain A, for example $ABCD$ (not necessarily a real answer), but not $BCDE$.

Solution:

$$\begin{aligned} AB+ &= AB \\ AC+ &= ABC \\ AD+ &= ADE \\ AE+ &= AE \\ ABD+ &= ABCDE \\ ABE+ &= ABE \\ ACD+ &= ABCDE \\ ACE+ &= ABCDE \\ ADE+ &= ADE \end{aligned}$$

Keys that contain A are: ABD, ACD, ACE.

(b) Decompose in BCNF relation $R(A, B, C, D, E)$ that satisfies the following functional dependencies. Show your steps, and show the keys in the decomposed relations.

$$\begin{aligned} E &\rightarrow C \\ BD &\rightarrow E \end{aligned}$$

Solution: Step1: In $ABCDE$: $E+ = EC$. Decompose into $\underline{E}C$ and $EABD$
Step 2: In $EABD$: $BD+ = BDE$. Decompose into $\underline{BD}E$ and BDA
Final decomposition is: $\underline{E}C$, $\underline{BD}E$ and BDA.

(c) Consider two relations $R(A, B, C)$ and $S(D, E, F)$ satisfying the following functional dependencies:

$$\begin{aligned} A &\rightarrow B \\ E &\rightarrow F \end{aligned}$$

and consider the following views:

```
CREATE VIEW V1(A,B,C) AS
  SELECT *
  FROM R
  WHERE C = 29

CREATE VIEW V2(A,B,E,F) AS
  SELECT DISTINCT R.A, R.B, S.E, S.F
  FROM R, S
  WHERE R.A = S.E
```

Indicate all functional dependencies that hold in each view. It suffices if you show a minimal set of functional dependencies, i.e. one that implies all other functional dependencies that hold on that view.

Solution: In $V1$:

$$\begin{aligned} A &\rightarrow B \\ B &\rightarrow C \end{aligned}$$

In $V2$:

$$\begin{aligned} A &\rightarrow B \\ E &\rightarrow F \\ A &\rightarrow E \\ E &\rightarrow A \end{aligned}$$

95. *(from CSE 544p, Autumn 2010, Final)*

(a) Consider a relation $R(A, B, C, D, E)$ satisfying the following functional dependencies:

$$\begin{aligned} A &\rightarrow BC \\ D &\rightarrow BC \\ CD &\rightarrow A \end{aligned}$$

Decompose this relation into BCNF. You only need to show your end result: the relations with their attributes, and the keys in each relation. Please order your relations lexicographically, for example if your answers were $R1(B, E), R2(D, B, C), R3(D)$ then you would list them in the following order $R1(B, C, D), R1(B, E), R3(D)$.

1. Indicate the relation decomposition: matrix choice $\{R1, R2, R3, R4, R5\} \rightarrow \{A, B, C, D, E\}$.
2. Indicate the keys:
matrix choice $\{R1, R2, R3, R4, R5\} \rightarrow \{A, B, C, D, E\}$.

Solution:

$$\begin{aligned} &A+ = ABC \\ R1(\underline{A}, B, C), &R2(A, D, E) \\ &D+ = DBCA \\ R21(\underline{D}, A), &R22(D, E) \end{aligned}$$

or:

$$\begin{aligned} &D+ = DBCA \\ R1(\underline{D}, A, B, C), &R2(D, E) \\ &A+ = ABC \\ R11(\underline{A}, B, C), &R12(A, \underline{D}) \end{aligned}$$

In both cases the answer is (up to renaming of relations):

$$R1(\underline{A}, B, C), \qquad R2(A, \underline{D}), \qquad R3(D, E)$$

(b) Consider a relation $R(A_1, A_2, \ldots, A_n)$ satisfying the following functional dependen-

cies:

$$\begin{aligned} A_1 &\to A_2 \\ A_2 &\to A_3 \\ A_3 &\to A_4 \\ &\dots \\ A_{n-1} &\to A_n \end{aligned}$$

Decompose this relation into BCNF. You need to indicate only your answer, for example $R_1(A_2, A_3, \dots, A_n), R_2(A_1, A_3, \dots, A_n), \dots, R_n(A_1, A_2, \dots, A_{n-1})$ (this is not the real answer).

Solution:

$$R_1(A_1, A_2), R_2(A_2, A_3), \dots, R_{n-1}(A_{n-1}, A_n)$$

(c) The relation $R(A, B, C, D, E)$ satisfies the following functional dependencies:

$$\begin{aligned} AB &\to C \\ BC &\to D \end{aligned}$$

Consider the following view:

```
create view V(B,D,E,F) as
  select distinct R.B, R.D, R.E, 'foo' as F
  from R
  where R.A = 'bar'
```

Find the key(s) in V. Multiple choice: B, D, E, F.

Solution: $B+ = ABCDF$. Thus, the key is B. The attributes A and F are included because they are constants: $A =' bar'$ and $F =' foo'$.

96. *(from CSE 344, Spring 2011, Final)*

(a) Consider a relation $R(A, B, C, D, E)$ that satisfies the following functional dependencies:

$$\begin{aligned} ABC &\to D \\ E &\to B \\ AD &\to C \end{aligned}$$

Decompose the schema in BCNF. Show all your steps.

Answer (Show the steps leading to the BCNF decomposition):

Solution: There are two solutions:

Solution 1:

Table	$X^+ = ?$	New table 1	New table 2
$R(A, B, C, D, E)$	$ABC+ = ABCD$	$R_1(A, B, C, D)$	$R_2(A, B, C, E)$
$R_1(A, B, C, D)$	$AD+ = ACD$	$R_3(A, C, D)$	$R_4(A, B, D)$
$R_2(A, B, C, E)$	$E+ = BE$	$R_5(B, E)$	$R_6(A, C, E)$

Answer: $R_3(A, C, D), R_4(A, B, D), R_5(B, E), R_6(A, C, E)$.

Solution 2:

Table	$X^+ = ?$	New table 1	New table 2
$R(A, B, C, D, E)$	$E+ = BE$	$R_1(B, E)$	$R_2(A, C, D, E)$
$R_2(A, C, D, E)$	$AD+ = ACD$	$R_3(A, C, D)$	$R_4(A, D, E)$

Answer: $R_1(B, E), R_3(A, C, D), R_4(A, D, E)$.

(b) For each statement below, indicate whether it is true or false. You do not need to justify your answer.

1. A materialized view may contain data that is not up to date. True or false ?

   Solution: true

   .

2. A query that uses a virtual view always runs much faster than the same query using a materialized view. True or false ?

   Solution: false

   .

3. An index is special case of a virtual view. True or false ?

Solution: false

.

4. An index is a special case of a materialized view. True or false ?

Solution: true

.

5. It takes much longer to create a virtual view than a materialized view. True or false ?

Solution: false

.

(c) Consider the table below:

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_3	c_1
a_2	b_3	c_2

For each of the functional dependencies listed below, indicate whether it holds or not. If it holds, write OK. If it does not hold, indicate two tuples in the table above that violate the functional dependency. Refer to the tuples as 1,2,3,4; for example, you may say that $A \rightarrow C$ fails because of the tuples 3,4.

FD	Holds ?
$B \to A$	
$C \to A$	
$A \to B$	
$C \to B$	
$A \to C$	
$B \to C$	
$BC \to A$	
$AC \to B$	
$AB \to C$	

Solution:

FD	Holds ?
$B \to A$	holds
$C \to A$	fails: tuples 1,3
$A \to B$	fails: tuples 1,2
$C \to B$	fails: tuples 1,3
$A \to C$	fails: tuples 1,2
$B \to C$	fails: tuples 3,4
$BC \to A$	holds
$AC \to B$	holds
$AB \to C$	fails: tuples 3,4

(d) Consider two relations `R(A,B)`, `S(C,D,E)`, with the following functional dependencies. In R: $A \to B$. In S: $C \to D$. Consider the following view definition:

```
create view V as
  select distinct R.A, R.B, S.D, S.E
  from R, S
  where R.B=S.C and S.E=55
```

For each of the following functional dependencies below, indicate whether they hold in the view $V(A, B, D, E)$:

1. $A \to D$. True or false ?

   Solution: true

2. $E \to B$. True or false ?

   Solution: false

3. $A \to E$. True or false ?

   Solution: true

4. $D \to A$. True or false ?

   Solution: false

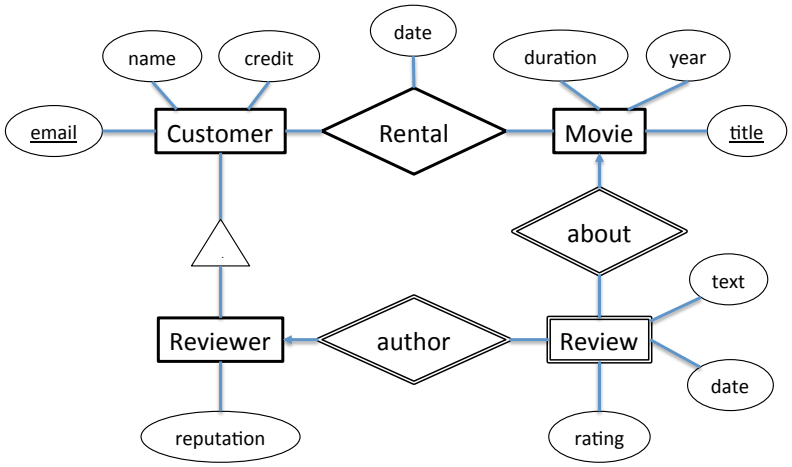
97. *(from CSE 344, Spring 2011, Midterm)*

(a) Design an E/R Diagram for an online video rental company:

- The company has data about movies, customers, rentals, reviewers, reviews.
- A Movie has a Title (key), Year, and Duration.
- A Customer has Name, Email (key), and Credit.
- Customers rent movies; customers may rent many movies, and a movie may be rented by many customers; each Rental has a Date.
- A Reviewer is a Customer, and has a Reputation attribute.
- A Review has a Rating, a Date, and Text (content). Each review is uniquely identified by the movie it is reviewing, and by the reviewer who wrote it.

Your answer should be an E/R Diagram.

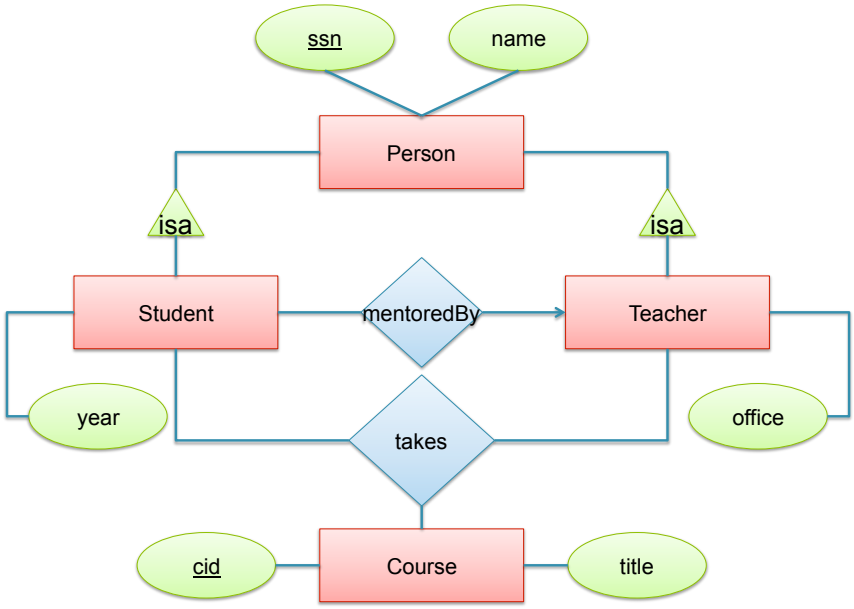
Solution:



Also OK: define RENTAL to be an entity set, with two many-one relationships, to CUSTOMER and MOVIE.

98. *(from CSE 344, Winter 2012, Final)*

(a) The following E/R diagram describes a database about students and teachers:



Every student and every teacher is a person. Some students have a teacher mentor. Every student is in some year (1, 2, 3, . . .) and every teacher has an office. Students take courses from teachers. Convert the E/R diagram to SQL. You should choose appropriate data types for all attributes, and should write all key and foreign key constraints.

Your Answer should consists of CREATE TABLE statements.

Solution:

```
create table Person(ssn int primary key, name varchar(30));
create table Teacher(
        ssn int primary key references Person,
        office varchar(20));
create table Student(
        ssn int primary key references Person,
        mentoredby int references Teacher,
        year int);
create table Course(cid int primary key, title varchar(30));
create table Takes(
        sid int references Student,
        tid int references Teacher,
        cid int references Course);
```

(b) Answer the following questions.

i. What is an update anomaly? Choose one of the following:

(a) One transaction reads an element that was updated by an earlier, uncommitted transaction.

(b) The application wants to update a foreign key to a new value that does not exist in the referenced relation.

(c) The same information is stored redundantly in the database, and only

some, but not all copies are updated.
Answer (a), (b), or (c).

Solution: (c)

ii. Every relational schema in SQL is in 1st normal form. True or false?

Solution: true

iii. Every XML data is in 1st normal form. True or false?

Solution: false

iv. Which of the following statements best describes the main reason for representing a relational database in 1st normal form?
(a) To achieve physical data independence.

(b) To remove data anomalies (insertion, update, deletion anomalies).

(c) To save space on disk.
Answer (a), (b), or (c).

Solution: (a)

v. Which of the following statements best describes the main reason for representing a relational database in BCNF?
(a) To achieve physical data independence.

(b) To remove data anomalies (insertion, update, deletion anomalies).

(c) To save space on disk.
Answer (a), (b), or (c).

Solution: (b)

(c) Consider the following instance of a relation $R(A, B, C, D)$:

A	B	C	D
a	b	c	d
a'	b	c'	d
a'	b'	c	d'

For each of the following statements indicate whether it is true or false:

i. A is a key. True or false?

Solution: false

ii. B is a key. True or false?

Solution: false

iii. AB is a key. True or false?

Solution: true

iv. BD is a key. True or false?

Solution: false

v. The functional dependency $A \to B$ holds. True or false?

Solution: false

vi. The functional dependency $B \to D$ holds. True or false?

Solution: true

vii. $D+ = BD$ holds. True or false?

Solution: true

(d) Consider two relations $R(A, B, C, D)$ and $S(A, B, C, D)$, with the following functional dependencies:

$$\begin{aligned} R: \quad & A \to BCD \\ & B \to ACD \end{aligned}$$

$$\begin{aligned} S: \quad & BC \to AD \\ & D \to B \end{aligned}$$

i. Find all keys in R. The keys are:

Solution: A and B

ii. Find all keys in S. The keys are:

Solution: BC and CD

iii. Find all keys in $R \cap S$. The keys are:

Solution: A, B, and CD

Solution: A set of attributes is a superkey in $R \cap S$ iff it is a superkey in R or in S. The superkeys in R are: A, AC, AD, ACD, B, BC, BD, BCD, ABC, ABD, $ABCD$. The superkeys in S are: BC, ABC, CD, ACD, BCD, $ABCD$. The superkeys in $R \cap S$ are all them, and therefore the keys are A, B, and CD.

iv. Assume that the relations $R(A, B, C, D)$ and $S(A, B, C, D)$ do not have any common values. That is, the values of the attribute $R.A$ are distinct from those of the attribute $S.A$, and the same for the attributes B, C, D. Find all keys in $R \cup S$ The keys are:

Solution: BC and ACD

Solution: A set of attributes is a superkey in $R \cup S$ iff it is a superkey in both R and S.
Thus, the superkeys in $R \cup S$ are $BC, ABC, ACD, BCD, ABCD$.
The keys are: BC and ACD.

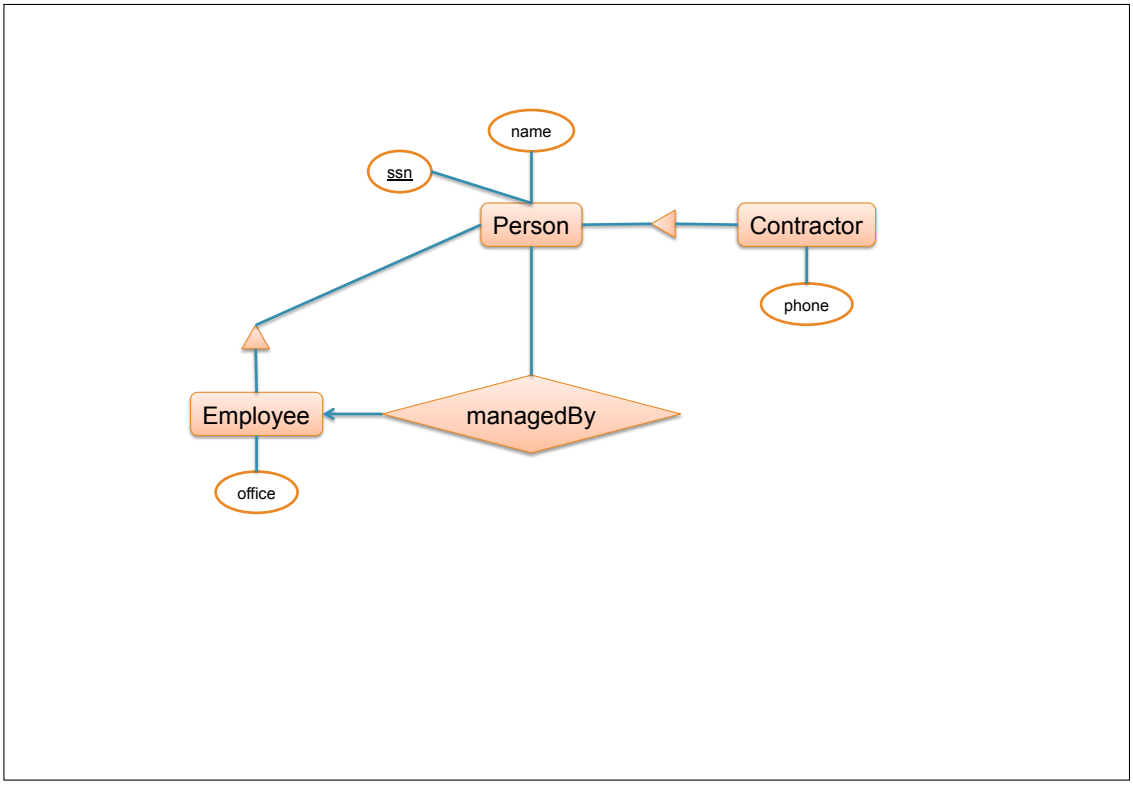
99. *(from CSE 344, Winter 2013, Final)*

(a) Design an E/R diagram for a database of employees about the following:

- Employee: they have ssn, name, office
- Contractor: they have ssn, name, phone
- ManagedBy: every Employee and every Contractor has a manager, who must be an Employee.

Answer Draw an E/R diagram; indicate all keys by underlining; mark all many-one relationships; show ISA relationships.

Solution:



(b) Consider the following relational schemas and sets of functional dependencies:

$$R(A,B,C,D): \qquad AB \to D, \quad AD \to C$$
$$S(A,B,C,D): \qquad C \to A, \quad D \to B$$

i. Show the keys for each relation. List all keys if there are more than one.

Answer Show the key(s) in R and the key(s) in S:

Solution:
The key in R is: AB.
The key in S is: CD.

ii. Consider the following two views:

$$V_1(A,B,C,D) = \sigma_{B='99'}(R(A,B,C,D))$$
$$V_2(A,B,C) = \Pi_{ABC}(R(A,B,C,D))$$

Show the keys in each view. List all keys if there are more than one.

Answer Show the key(s) in V_1 and the key(s) in V_2:

Solution:
The key in V_1 is: A.
The key in V_2 is: AB. (The FD in V_2 is $AB \to C$.)

iii. Consider the following view:

$$V_3(A, B, C, D) = R(A, B, C, D) \cap S(A, B, C, D)$$

Compute the closure of BC in V_3.

Answer Show $(BC)^+$.

Solution: $ABCD$

(c) Consider the following relational schema and set of functional dependencies.
R(A,B,C,D,E) with functional dependencies:

$$\begin{aligned} A &\to B \\ C &\to B \\ DE &\to C \end{aligned}$$

Decompose R into BCNF. Show your work for partial credit. Your answer should consist of a list of table names and attributes and an indication of the keys in each table (underlined attributes).

Answer (Decompose R into BCNF):

Solution:

Solution 1 Start with the set A: $A+ = AB$, decompose into $\underline{A}B$ and $ACDE$. In the latter use $DE+ = CDE$ and decompose into $C\underline{DE}$ and $\underline{ADE}$.

Final answer: $R_1(\underline{A}, B), R_2(C, \underline{D}, \underline{E}), R_3(\underline{A}, \underline{D}, \underline{E})$.

Solution 2 Start with the set C: $C+ = BC$, decompose into $B\underline{C}$ and $ACDE$. The latter decomposes as above.

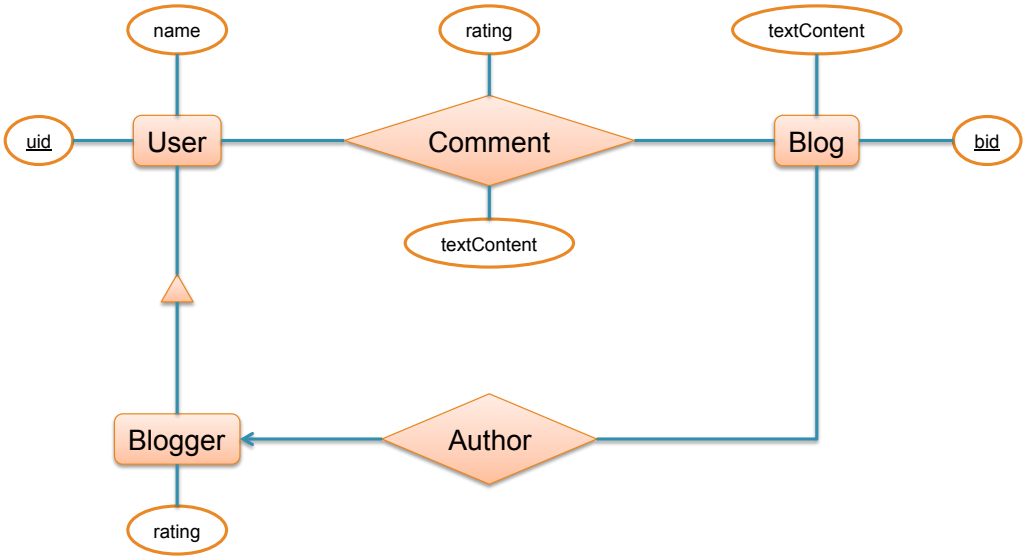
Final answer: $R_1(B, \underline{C}), R_2(C, \underline{D}, \underline{E}), R_3(\underline{A}, \underline{D}, \underline{E})$.

Solution 3 Start with the set DE: $DE+ = BCDE$, decompose into $BC\underline{DE}$ and ADE. In the former choose the set C: $C+ = BC$ hence it decomposes into $B\underline{C}$ and CDE.

Final answer: $R_1(B, \underline{C}), R_2(C, \underline{D}, \underline{E}), R_3(\underline{A}, \underline{D}, \underline{E})$ (same as Solution 2).

100. *(from CSE 344, Autumn 2013, Final)*

(a) Consider the following E/R diagram:



The schema describes a database of bloggers and blogs.

- Every blog is authored by exactly one blogger.
- Every blogger is a user.
- Users may comment on blogs.
- Each comment has an optional text content, and/or an optional rating.
- Each blogger also has a rating (which is not optional).

Write a sequence of SQL queries that create the tables for this E/R diagram; `uid`, `bid`, and `rating` are integers, all other attributes are `text`. You should turn in several CREATE TABLE statements, which include key, foreign key, and not null declarations.

Solution:

```
create table Usr(uid int primary key,
                 uname text not null);

create table Blgger(uid int primary key references Usr,
                    rating int not null);

create table Blog(bid int primary key,
```

```
                  textContent text,
                  blogger int references Usr not null);

create table Comment(uid int references Usr not null,
                     bid int references Blog not null,
                     rating int, textContent text);
```

(b) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	0	2	2	2
3	1	0	3	3
4	0	1	0	4
5	1	2	1	5
6	0	0	2	0
7	1	1	3	1
8	0	2	0	2

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others.

Solution: We notice:

$$\begin{aligned} B &= (A \mod 2) \\ C &= (A \mod 3) \\ D &= (A \mod 4) \\ E &= (A \mod 6) \end{aligned}$$

Therefore: $A \to BCDE$, $D \to B$, $E \to BC$.

(c) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. Show your work, show the final result, and show the keys in the final result.

Solution:

1. $D+ = BD$. Split $ABCDE$ into BD and $ACDE$
2. $E+ = (B)C$. Split $ACDE$ into EC and ADE

Final answer: $R1(\underline{A}, D, E)$, $R2(\underline{E}, C)$, $R3(\underline{D}, B)$.

9 Views (101–114)

A view is a query with a given name. That simple idea supports three distinct uses, and the questions here touch all of them: simplifying complex queries, providing logical data independence when the underlying schema changes, and restricting access so that a user sees only part of a table.

Materialized views raise the harder questions. Once a view is stored rather than recomputed, it must be maintained as the base tables change, and several problems ask what it costs to keep a given view up to date, or which updates can affect it at all. Others ask whether a query can be answered from the view alone (this is called the *query rewriting problem*), which is the same question a query optimizer asks when it considers using a materialized view.

101. *(from CSE 344, Winter 2013, Final)*

We have a table `Follows(person1, person2)` storing followers in a social network. A data analyst issues the following commands:

```
CREATE VIEW V1 AS
   SELECT x.person1 AS person1, y.person2 AS person2
   FROM Follows x, Follows y
   WHERE x.person2=y.person1;

CREATE VIEW V2 AS
   SELECT x.person1 AS person1, y.person2 AS person2
   FROM V1 x, V1 y
   WHERE x.person2=y.person1;

CREATE VIEW V3 AS
   SELECT x.person1 AS person1, y.person2 AS person2
   FROM V2 x, V2 y
   WHERE x.person2=y.person1;

CREATE VIEW V4 AS
   SELECT x.person1 AS person1, y.person2 AS person2
   FROM V3 x, V3 y
   WHERE x.person2=y.person1;

SELECT x.person1 AS person1, y.person2 AS person2
FROM V4 x, V4 y
WHERE x.person2=y.person1;
```

How many join operations will the SQL engine perform in response to these five commands (four view definitions and one query), in each of the two cases below:

(a) All views are virtual.

Write the total number of joins.

Solution: 31

(b) All views are materialized.

Write the total number of joins.

Solution: 5

Solution: 31 joins and 5 joins respectively. Database systems do not use DAGs in query plans. To check this in postgres, run the following:

```
create table follows(person1 int, person2 int);

insert into follows values (0,1);
insert into follows values (1,2);
```

```
insert into follows values (2,3);
insert into follows values (3,4);
insert into follows values (4,5);
insert into follows values (5,6);
insert into follows values (6,7);
insert into follows values (7,8);
insert into follows values (8,9);
insert into follows values (9,10);
insert into follows values (10,11);
insert into follows values (11,12);
insert into follows values (12,13);
insert into follows values (13,14);
insert into follows values (14,15);
insert into follows values (15,16);
insert into follows values (16,17);
insert into follows values (17,18);
insert into follows values (18,19);
insert into follows values (19,20);
insert into follows values (20,21);
insert into follows values (21,22);
insert into follows values (22,23);
insert into follows values (23,24);
insert into follows values (24,25);
insert into follows values (25,26);
insert into follows values (26,27);
insert into follows values (27,28);
insert into follows values (28,29);
insert into follows values (29,30);
insert into follows values (30,31);
insert into follows values (31,32);

-- create the four views above

\o outputfile.txt

explain SELECT x.person1 AS person1, y.person2 AS person2 FROM V4 x, V4 y WHERE x.person2=y.person1;
```

At command prompt:

```
more outputfile.txt | grep Join | wc
```

102. *(from CSE 344, Autumn 2013, Final)*

(a) For each of the following statements indicate whether it is true or false:

i. A *materialized* view is always up to date.
Answer Yes/No:

Solution: No

ii. A *virtual* view is always up to date.
Answer Yes/No:

Solution: Yes

iii. By default, a view defined in SQL using the CREATE VIEW command is a *materialized view.*

Answer Yes/No:

Solution: No

iv. An index is a *materialized* view.
Answer Yes/No:

Solution: Yes

(b) Consider the following schema:

```
Sales(sid, product, month, category)   /* sid = primary key */
```

The relation records individual sales. Note that the same product may be sold in different months and/or under different categories.

You want to execute the following query, which computes the total sales of `Toy` products in all months other than `December`:

```
select product, count(*)
from Sales
where  category = 'Toy' and month != 'December'
group by product;
```

Unfortunately, you do not have access to the `Sales` table. Instead, the database administrator gave you access only to the following three views:

```
create view V1 as
  select product, count(*) as c1
  from Sales
  where category != 'Toy'  and month != 'December'
  group by product;

create view V2 as
  select month, count(*) as c2
  from Sales
  group by month;

create view V3 as
  select product, month, count(*) as c3
  from Sales
  group by product, month;
```

Rewrite the query computing total sales of `Toy` products in all months other than `December`, by referring only to the views `V1`, `V2`, `V3`, and without referring to the table `Sales`. If needed, create new views or use the `with` syntax (see the reference sheet at the beginning of this paper), but make sure you only refer to `V1,V2,V3` and not to `Sales`.

Solution:

```
create table Sales(
    sid int primary key,
    product text,
    month text,
    category text);

insert into Sales values (1, 'aaa', 'June', 'Toy');
insert into Sales values (2, 'bbb', 'December', 'Toy');
insert into Sales values (3, 'ccc', 'June', 'Phone');
insert into Sales values (4, 'ddd', 'December', 'Phone');
insert into Sales values (5, 'ddd', 'December', 'Toy');
insert into Sales values (6, 'ddd', 'June', 'Toy');

create view  v4 as
  select product, sum(c3) as c4
  from v3
  where month != 'December'
  group by product;

select x.product, x.c4 - (case when y.c1 is null then 0 else y.c1 end)
from v4 x left outer join v1 y on x.product = y.product
where x.c4 - (case when y.c1 is null then 0 else y.c1 end) > 0;
```

103. *(from CSE 544p, Winter 2014, Final)*

(a) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	0	2	2	2
3	1	0	3	3
4	0	1	0	4
5	1	2	1	5
6	0	0	2	0
7	1	1	3	1
8	0	2	0	2
9	1	0	1	3
10	0	1	2	4
11	1	2	3	5
12	0	0	0	0

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others. Write the functional dependencies using `->`, for example `AB -> CD` (not a real answer).

Solution: $A \to BCDE$, $D \to B$, $E \to BC$, $BC \to E$, because:

$$B = (A \mod 2) \quad C = (A \mod 3) \quad D = (A \mod 4) \quad E = (A \mod 6)$$

(b) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. Show your work, show the final result, and show the keys in the final result (since you cannot underline, simply write what the keys are, e.g. "In `R2(B,D,E) the key is BE`").

Solution:

1. $D+ = BD$. Split $ABCDE$ into BD and $ACDE$
2. $E+ = (B)C$. Split $ACDE$ into EC and ADE

Final answer: $R1(\underline{A}, D, E)$, $R2(\underline{E}, C)$, $R3(\underline{D}, B)$.

(c) Consider a relation $R(A_1, A_2, \ldots, A_n)$ satisfying the following functional dependencies:

$$\begin{aligned} A_1 &\rightarrow A_2 \\ A_2 &\rightarrow A_3 \\ A_3 &\rightarrow A_4 \\ &\ldots \\ A_{n-1} &\rightarrow A_n \end{aligned}$$

Decompose this relation into BCNF. You need to indicate only your answer, for example $R_1(A_2, A_3, \ldots, A_n), R_2(A_1, A_3, \ldots, A_n), \ldots, R_n(A_1, A_2, \ldots, A_{n-1})$ (this is not the real answer). To represent subscripts, you can use the underscore key "_", for example R_{n-1} can be written as `R_`$(n-1)$.

Solution:

$$R_1(A_1, A_2), R_2(A_2, A_3), \ldots, R_{n-1}(A_{n-1}, A_n)$$

The following does not result from the BCNF decomposition algorithm and is considered a wrong solution:

$$R_1(A_1, A_2), R_2(A_1, A_3), \ldots, R_{n-1}(A_1, A_n)$$

(d) The relation $R(A, B, C, D, E)$ satisfies the following functional dependencies:

$$\begin{aligned} AB &\rightarrow C \\ BC &\rightarrow D \end{aligned}$$

Consider the following view:

```
create view V(B,D,E,F) as
  select distinct R.B, R.D, R.E, 'foo' as F
  from R
  where R.A = 'bar'
```

Find the key in V. Select all attributes that, together, form the key. Multiple choice: B, D, E, F.

Solution: $BE+ = ABCDEF$. Thus, the key is B, E. The attributes A and F are included because they are constants: $A =' bar'$ and $F =' foo'$.

104. *(from CSE 544p, Autumn 2015, Final)*

(a) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	2	2	2	0
3	3	3	0	1
4	4	0	1	0
5	5	1	2	1
6	0	2	0	0
7	1	3	1	1
8	2	0	2	0
9	3	1	0	1
10	4	2	1	0
11	5	3	2	1
12	0	0	0	0

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others. Write the functional dependencies using `->`, for example `AB -> CD` (not a real answer).

Solution: $A \to BCDE$, $C \to E$, $B \to DE$, $DE \to B$, because:

$$B =(A \mod 6) \quad C =(A \mod 4) \quad D =(A \mod 3) \quad E =(A \mod 2)$$

(b) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. Show your work, show the final result, and show the keys in the final result (since you cannot underline, simply write what the keys are, e.g. "In `R2(B,D,E) the key is BE`").

Solution:

1. $C+ = CE$. Split $ABCDE$ into CE and $ABCD$
2. $B+ = (E)D$. Split $ABCD$ into BD and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, D)$, $R3(\underline{C}, E)$.
Alternative:

1. $B+ = BDE$. Split $ABCDE$ into BDE and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, \underline{D, E})$.
Notice that $R1(A, B), R2(A, C), R3(A, D), R4(A, E)$ is, strictly speaking, a correct BCNF decomposition, since A is a key and every 2-attribute relation is in BCNF. But it is a poor decomposition because it fails to capture any other functional dependences besides A being a key. I gave only partial credit for such a decomposition.

(c) Consider a relation $R(A_1, A_2, \ldots, A_n)$ satisfying the following functional dependencies:

$$\begin{aligned} A_1 &\rightarrow A_2 \\ A_2 &\rightarrow A_3 \\ A_3 &\rightarrow A_4 \\ &\ldots \\ A_{n-1} &\rightarrow A_n \end{aligned}$$

Decompose this relation into BCNF. You need to indicate only your answer, for example $R_1(A_2, A_3, \ldots, A_n), R_2(A_1, A_3, \ldots, A_n), \ldots, R_n(A_1, A_2, \ldots, A_{n-1})$ (this is not the real answer). To represent subscripts, you can use the underscore key "_", for example R_{n-1} can be written as `R_`$(n-1)$.

Solution:

$$R_1(A_1, A_2), R_2(A_2, A_3), \ldots, R_{n-1}(A_{n-1}, A_n)$$

(d) The relation $R(A, B, C, D, E)$ satisfies the following functional dependencies:

$$\begin{aligned} AD &\rightarrow E \\ CD &\rightarrow A \end{aligned}$$

Consider the following view:

```
create view V(B,D,E,F) as
  select distinct R.B, R.D, R.E, 'foo' as F
  from R
  where R.C = 'bar'
```

Find the key in V. Select all attributes that, together, form the key. Multiple choice: B, D, E, F.

Solution: $BD+ = ABCDEF$. Thus, the key is BD. The attributes C and F are included because they are constants: $C =' bar'$ and $F =' foo'$.

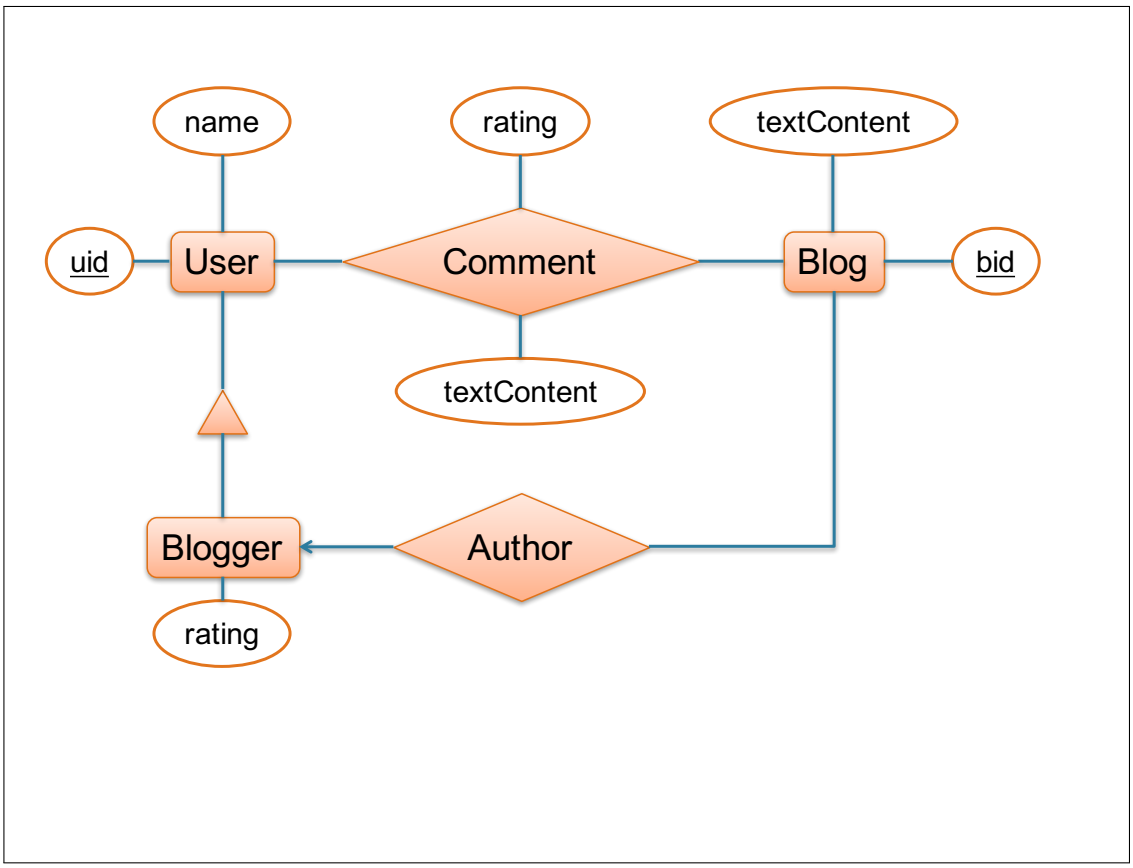
105. *(from CSE 414, Spring 2016, Final)*

(a) You are running a startup company allowing users to post blogs, and to comment on other users' blogs. Design the E/R diagram for your company's data. Your database should store information about users, bloggers, and blogs, and encode the following information:

- Every user has a `user ID` and a `name`.
- Every blog has a `blog ID`, a `text content` and one `author`, who is a blogger.
- Every blogger is a user.
- Every blogger also has a `rating` attribute.
- Users may comment on blogs.
- Every comment has an optional `text content`, and/or an optional `rating`.

Design the E/R diagram for this database.

Solution:



(b) Consider the following table:

First Name	First Initial	Last Name	Last Initial	FullName	FullAddress	Zip Code
"Alice"	"A"	"Levy"	"L"	"Alice Levy"	"45h St, Seattle, 98185",	"98185"
"Bob"	"B"	"Levy"	"L"	"Bob Levy"	"45h St, Seattle, 98185",	"98185"
"Alice"	"A"	"Davis"	"D"	"Alice Davis"	"Oak Ave, Seattle, 98185",	"98185"
"Carol"	"C"	"Davis"	"D"	"Carol Davis"	"Sunnyvale, 94085",	"94085"
"Carol"	"C"	"Louis"	"L"	"Carol Louis"	"Oak Ave, Seattle, 98185",	"98185"

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others.

Solution:

```
FirstName -> FirstInitial
LastName -> LastInitial
FirstName, LastName -> FullName
FullAddress -> ZipCode
FullName -> FirstName, LastName, FullAddress
```

(c) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. Show your work, and the final normalized schema including all keys. Then represent the relation instance below in your normalized

schema:

First Name	First Initial	Last Name	Last Initial	FullName	FullAddress	Zip Code
“Alice”	“A”	“Levy”	“L”	“Alice Levy”	“45h St, Seattle, 98185”,	“98195”
“Bob”	“B”	“Levy”	“L”	“Bob Levy”	“45h St, Seattle, 98185”,	“98195”
“Alice”	“A”	“Davis”	“D”	“Alice Davis”	“Oak Ave, Seattle, 98185”,	“98195”
“Carol”	“C”	“Davis”	“D”	“Carol Davis”	“Sunnyvale, 94085”,	“94085”
“Carol”	“C”	“Louis”	“L”	“Carol Louis”	“Oak Ave, Seattle, 98185”,	“98195”

Solution: The BCNF decomposition is:

R1(FirstName,FirstInitial)
R2(LastName,LastInitial)
R3(FullName,FirstName,LastName)
R4(FullName,FullAddress)
R5(FullAddress,ZipCode)

```
drop table if exists R;
create table R (
  FirstName text,
  FirstInitial text,
  LastName text,
  LastInitial text,
  FullName text,
  FullAddress text,
  ZipCode int
);

insert into R values('Alice', 'A', 'Levy', 'L', 'Alice Levy', '45h St, Seattle, 98185', 98195);
insert into R values('Bob', 'B', 'Levy', 'L', 'Bob Levy', '45h St, Seattle, 98185', 98195);
insert into R values('Alice', 'A', 'Davis', 'D', 'Alice Davis', 'Oak Ave, Seattle, 98185', 98195);
insert into R values('Carol', 'C', 'Davis', 'D', 'Carol Davis', 'Sunnyvale, 94085', 94085);
insert into R values('Carol', 'C', 'Louis', 'L', 'Carol Louis', 'Oak Ave, Seattle, 98185', 98195);

drop table if exists R1;
drop table if exists R2;
drop table if exists R3;
drop table if exists R4;
create table R1(FirstName text primary key, FirstInitial text);
create table R2(LastName text primary key, LastInitial text);
create table R3(FullName text primary key, FirstName text, LastName text);
create table R4(FullName text primary key, FullAddress text);
create table R5(FullAddress text primary key, ZipCode int);

insert into R1 (select distinct FirstName, FirstInitial from R);
insert into R2 (select distinct LastName, LastInitial from R);
insert into R3 (select distinct FullName, FirstName, LastName from R);
insert into R4 (select distinct FullName, FullAddress from R);
insert into R5 (select distinct FullAddress, ZipCode from R);

select * from R1;
--  firstname | firstinitial
-- -----------+--------------
--  Carol     | C
--  Alice     | A
--  Bob       | B

select * from R2;
--  lastname | lastinitial
-- ----------+-------------
--  Louis    | L
--  Davis    | D
```

```
-- Levy     | L

select * from R3;
--   fullname   | firstname | lastname
-- -------------+-----------+----------
--  Bob Levy    | Bob       | Levy
--  Alice Levy  | Alice     | Levy
--  Alice Davis | Alice     | Davis
--  Carol Louis | Carol     | Louis
--  Carol Davis | Carol     | Davis

select * from R4;
--   fullname   |       fulladdress
-- -------------+-------------------------
--  Bob Levy    | 45h St, Seattle, 98185
--  Alice Davis | Oak Ave, Seattle, 98185
--  Carol Davis | Sunnyvale, 94085
--  Alice Levy  | 45h St, Seattle, 98185
--  Carol Louis | Oak Ave, Seattle, 98185

select * from R5
--        fulladdress       | zipcode
-- -------------------------+---------
--  Sunnyvale, 94085        |   94085
--  45h St, Seattle, 98185  |   98195
--  Oak Ave, Seattle, 98185 |   98195

-- Final check that the decomposition is lossless:
select * from R1 natural join R2 natural join R3 natural join R4 natural join R5;
-- OK
```

(d) Consider the following database schema (the DROP TABLE statements are there for testing on a DBMS):

```
drop table if exists orders;
drop table if exists customer;
drop table if exists product;

create table customer (
    cid int primary key,
    cname text,
    city text);
create table product (
    pid int primary key,
    pname text,
    price int);
create table orders (
    cid int references customer,
    pid int references product);
```

Consider the following query:

```
select x.cid, x.cname, x.city, z.pid, z.pname, z.price
from customer x, orders y, product z
where x.city = 'Seattle'
  and x.cid = y.cid
  and y.pid = z.pid
```

```
  and z.price = 100;
```

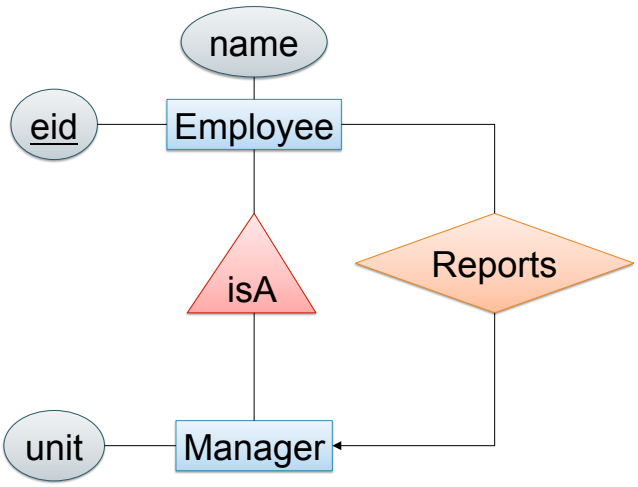
Write all functional dependencies that hold on the query answer.

Solution:

```
cid -> cname
pid -> pname
cname -> city
pname -> city
city -> price
price -> city
```

106. *(from CSE 344, Autumn 2017, Final)*

(a) Consider the following E/R diagram:



Write the CREATE TABLE statements to implement the E/R diagram in SQL. Your statements should contain all key and foreign key declarations. Assume that `eid` and `unit` are integers, and `name` is a text.

Solution:

```
create table Employee(eid int primary key, name text);
create table Manager(eid int primary key references Employee, unit int);
```

This SQL query would normally not execute on any engine, because of the cyclic reference between Employee and Manager, but it is an acceptable solution on an exam, and the students are told about this during the exam.

(b) The relation $R(A, B, C, D, E, F)$ satisfies the following functional dependencies:

$$AB \to C \qquad CD \to E$$

Find a BCNF decomposition for R. In each of the new relations indicate the key (by underlining the attribute, or set of attributes, that form a key).

Solution: $AB^+ = ABC \Rightarrow$ decompose R into $S(A, B, C)$, $T(A, B, D, E, F)$.
$CD^+ = CDE \Rightarrow$ decompose T into $K(C, D, E)$, $L(C, D, F)$.
Final answer: $S(\underline{A, B}, C)$, $K(\underline{C, D}, E)$, $L(\underline{C, D}, F)$.

(c) For each statement below indicate whether it is true or false.

i. The reason why we decompose a relation in Boyce-Codd Normal Form is to save disk space when the relation becomes too large. True or false?

Solution: False

ii. Suppose that the BCNF decomposition of R consists of three relations S, T, K: then S, T, K are a lossless decomposition of R. True or false?

Solution: True

iii. If $X \subseteq Y$ then $X^+ \subseteq Y^+$. True or false?

Solution: True

iv. $(X^+)^+ = X$. True or false?

Solution: False

v. $(X \cup Y)^+ = X^+ \cup Y^+$. True or false?

Solution: False

(d) Answer the questions below:

i. Consider the following three relations

$$R(A, B), \quad S(B, C), \quad T(C, A)$$

where $R.A$ is a key. Suppose they have the same cardinality N:

$$|R| = |S| = |T| = N$$

What is the largest possible size of the natural join $R \bowtie S \bowtie T$? Your answer should be a mathematical formula that depends on N, e.g. $N^2 \log(N)$ or $N + 5$ (not a real answer).

Solution: N

ii. Consider two relations $R(A, B, C)$ and $S(D, E, F)$ satisfying the following functional dependencies:

$$R : A \to B \qquad S : DE \to F$$

Let $T(A, B, C, D, E, F)$ be the following table:

```
create table T as
 (select R.A, R.B, R.C, S.D, S.E, S.F
  from R, S
  where R.C = S.D and S.E = 5)
```

Find the key or keys of T. If T has more than one key, list all keys.

Solution: Keys: AC and AD.

iii. Consider a relation with the schema $R(A, B, C)$. Suppose the relational instance satisfies the functional dependency

$$A \to B$$

and does not satisfy any other functional dependencies. We know the following statistics on R:

$$|R| = 512 \qquad V(R, B) = 510$$

Compute $V(R, A)$; your answer should consist of a number.

Solution: 511

iv. True or false? If R is a relation that satisfies a set of functional dependencies, then we can declare all of them in the `CREATE TABLE R` statement by using `PRIMARY KEY` and `UNIQUE` declarations. True or false?

Solution: False

v. True or false? If R is a relation that satisfies a set of functional dependencies and is in BCNF, then we can declare all of them in the `CREATE TABLE R` statement by using `PRIMARY KEY` and `UNIQUE` declarations. True or false?

Solution: True

107. *(from CSE 344, Winter 2019, Final)*

(a) Consider a relation $R(A, B, C, D, E)$ satisfying the following FD's:

$$AB \to CD$$
$$DE \to B$$

Decompose R into BCNF.

Solution:

- $\{DE\}^+ = BDE$. Decompose into $R_1(\underline{D, E}, B)$, $R_2(A, C, D, E)$.

Alternative:

- $\{AB\}^+ = ABCD$. Decompose into $R_1(\underline{A, B}, C, D)$, $R_2(A, B, E)$.

(b) Consider two relations $R(\underline{K}, A, B)$, $S(\underline{L}, C, D)$. The following query returns a relation with attributes K, A, B, L, C, D:

```
select *
from R, S
where R.B=S.L and R.A=S.D
```

Find all functional dependencies satisfied by the answer to the query above. You only need to indicate a minimal set of FDs; for example if you wrote $X \to Y$ and $YZ \to U$ then you don't need to write $XZ \to U$.

Solution:

$$K \to LABCD \qquad L \to BCD \qquad B \to L$$
$$A \to D \qquad D \to A$$

(c) Consider a relation with three attributes A, B, C. Answer the questions below.

i. Give an example of functional dependencies such that the relation has a single key consisting of two attributes.

Solution: $AB \to C$

ii. Give an example of functional dependencies such that the relation has two keys consisting of one attribute each.

Solution: $A \to BC$, $B \to AC$.

iii. Give an example of functional dependencies such that both AB and AC are keys, but A is not a key.

Solution: $AB \to C$, $AC \to B$.

iv. Give an example of functional dependencies such that the relation is *not* in BCNF.

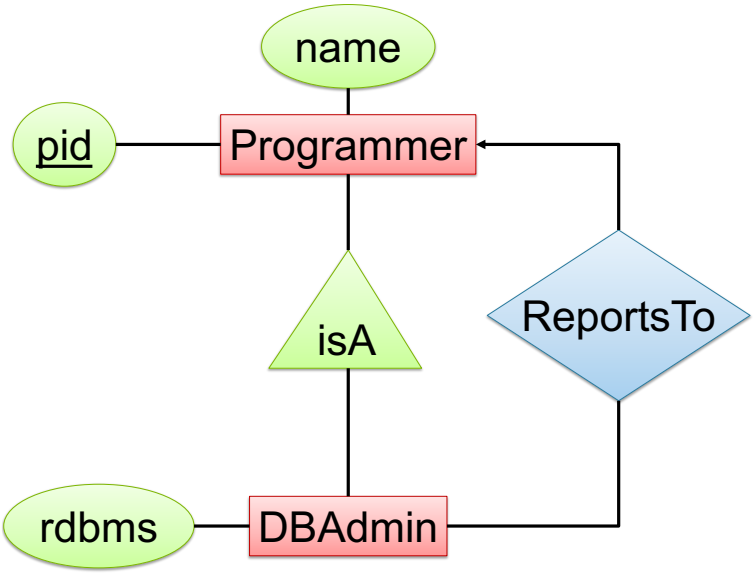
Solution: $A \to B$.

v. Give an example of functional dependencies such that the closures A^+, B^+, C^+ are three distinct sets, meaning no two sets can be equal.

Solution: $A \to B$, $B \to C$. Then $A^+ = ABC$, $B^+ = BC$, $C^+ = C$.

108. *(from CSE 414, Spring 2019, Final)*

(a) A large software company maintains a database of all its programmers. Some programmers are database administrators, and they have to report to some other programmer. The E/R diagram is given below. Write the CREATE TABLE statements for this E/R diagram, where `name` and `rdbms` are of type text, and `pid` is of type int. You answer should include keys and foreign keys where necessary.



Write your answer below:

Solution:

```
create table Programmer
    (pid int primary key,
     name text);
create table DBAdmin
    (pid int primary key references Programmer,
     rdbms text,
     reportsTo int references Programmer);
```

(b) Consider a relation $R(A, B, C, D, E)$ satisfying the following FD's:

$$A \to B \qquad C \to B \qquad BD \to E$$

Decompose R into BCNF. In your final answer indicate the key of each relation.

Solution: Solution 1:

- in $R(ABCDE)$: $A+ = AB$ split into $R_1(AB), R_2(ACDE)$.
- in $R_2(ACDE)$: $CD+ = CDE$ split into $R_3(CDE), R_4(ACD)$.

Final answer: $R_1(\underline{A}B), R_3(\underline{CD}E), R_4(ACD)$.
Solution 2:

- in $R(ABCDE)$: $C+ = BC$ split into $R_1(BC), R_2(ACDE)$.
- in $R_2(ACDE)$: $CD+ = CDE$ split into $R_3(CDE), R_4(ACD)$.

Final answer: $R_1(B\underline{C})$, $R_3(\underline{CD}E)$, $R_4(ACD)$.
Solution 3:

- in $R(ABCDE)$: $BD+ = BDE$ split into $R_1(BDE)$, $R_2(ABCD)$.
- in $R_2(ABCD)$: $A+ = AB$ split into $R_3(AB)$, $R_4(ACD)$.

Final answer: $R_1(\underline{BD}E)$, $R_3(\underline{A}B)$, $R_4(ACD)$.
Solution 4:

- in $R(ABCDE)$: $BD+ = BDE$ split into $R_1(BDE)$, $R_2(ABCD)$.
- in $R_2(ABCD)$: $C+ = BC$ split into $R_3(BC)$, $R_4(ACD)$.

Final answer: $R_1(\underline{BD}E)$, $R_3(B\underline{C})$, $R_4(ACD)$.

(c) Consider a relation $R(\underline{A}, B, C, D, E, F, G)$ where A is a key. We create a new table

S by running this query:

```
S:    select *  from R  where (B=C+2) and (D+E=F) and (G=7);
```

Indicate which of the following FDs are guaranteed to hold on S:

i. $B \rightarrow C$ Yes or no?

Solution: Yes

ii. $C \rightarrow B$ Yes or no?

Solution: Yes

iii. $D \rightarrow F$ Yes or no?

Solution: No

iv. $F \rightarrow D$ Yes or no?

Solution: No

v. $DE \rightarrow F$ Yes or no?

Solution: Yes

vi. $F \rightarrow DE$ Yes or no?

Solution: No

vii. $F \rightarrow G$ Yes or no?

Solution: Yes

viii. $G \rightarrow F$ Yes or no?

Solution: No

ix. $CDE \rightarrow A$ Yes or no?

Solution: No

x. $A \rightarrow CDE$ Yes or no?

Solution: Yes

109. *(from CSE 544p, Spring 2021, Final)*

(a) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	2	2	2	0
3	3	3	0	1
4	4	0	1	0
5	5	1	2	1
6	0	2	0	0
7	1	3	1	1
8	2	0	2	0
9	3	1	0	1
10	4	2	1	0
11	5	3	2	1
12	0	0	0	0

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others.

Solution: $A \to BCDE$, $C \to E$, $B \to DE$, $DE \to B$, because:

$B = (A \mod 6)$ $C = (A \mod 4)$ $D = (A \mod 3)$ $E = (A \mod 2)$

(b) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. You only need to show your final result: the relation names, their attributes, and their keys (underline the key attributes).

Solution:

1. $C+ = CE$. Split $ABCDE$ into CE and $ABCD$
2. $B+ = (E)D$. Split $ABCD$ into BD and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, D)$, $R3(\underline{C}, E)$.
Alternative:

1. $B+ = BDE$. Split $ABCDE$ into BDE and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, \underline{D}, \underline{E})$.
$R1(A, B), R2(A, C), R3(A, D), R4(A, E)$ is, strictly speaking, a correct BCNF decomposition, since A is a key and every 2-attribute relation is in BCNF. But it is a poor decomposition because it fails to capture any other functional dependences besides A being a key.

(c) The relation $R(A, B, C, D, E)$ satisfies the following functional dependencies:

$$AD \rightarrow E$$
$$CD \rightarrow A$$

Consider the following view:

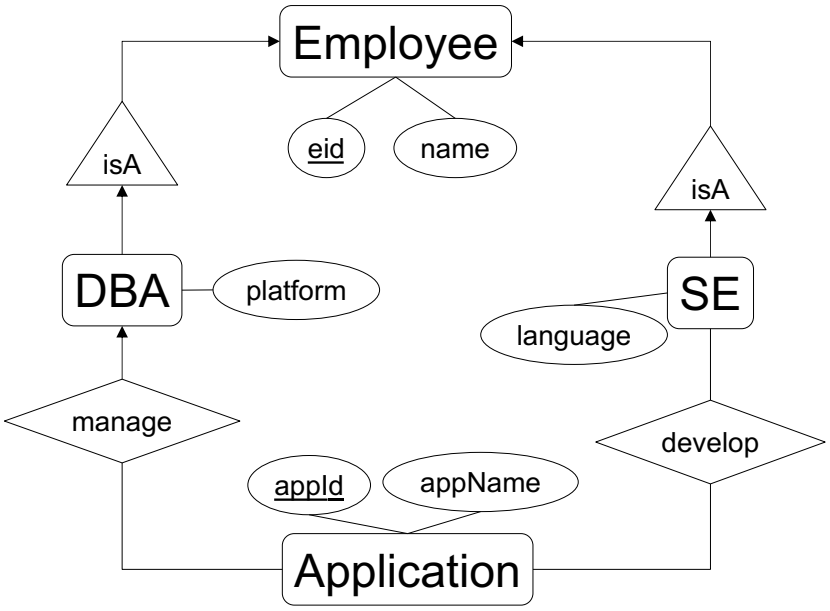
```
create view V(B,D,E,F) as
  select distinct R.B, R.D, R.E, 'foo' as F
  from R
  where R.C = 'bar'
```

Find the key in V.

Solution: $BD+ = ABCDEF$. Thus, the key is BD. The attributes C and F are determined because they are constants: $C =' bar'$ and $F =' foo'$.

110. *(from CSE 414, Spring 2024, Midterm Review)*

A company develops applications running on top of databases. Software Engineers (SE) develop the applications, and Database Administrators (DBA) manage the databases. This information is captured by the following Entity-Relationship diagram:



(a) For each of the statements below, indicate whether it is true or false:

i. A DBA can manage at most one application.
Answer TRUE or FALSE:

Solution: false

ii. An application can be managed by at most one DBA.
Answer TRUE or FALSE:

Solution: true

iii. Every application has a `platform` attribute.
Answer TRUE or FALSE:

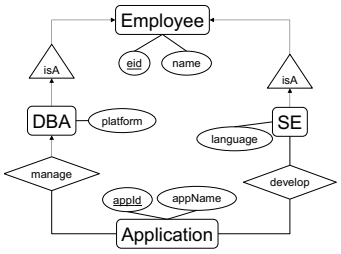
Solution: false

iv. Every DBA has a `platform` attribute.
Answer TRUE or FALSE:

Solution: true

v. Every DBA has a `name` attribute.
Answer TRUE or FALSE:

Solution: true



(b) Write the SQL statements to create the tables representing the E/R diagram above. The keys `eid, appId` are integers, `name, appName, platform, language` are strings. Include all key and foreign key statements.

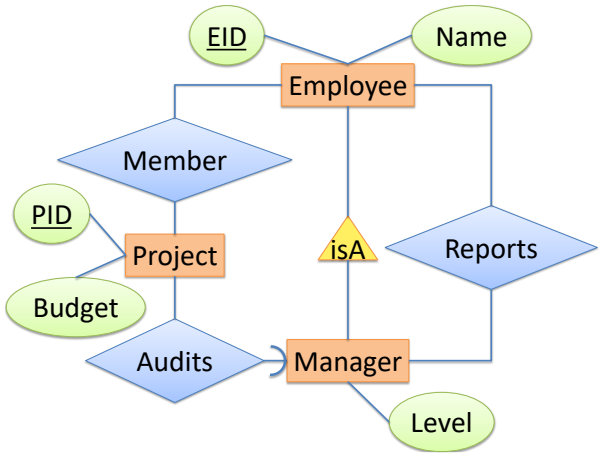
Write your answer here:

Solution:

```
drop table if exists develops;
drop table if exists application;
drop table if exists dba;
drop table if exists se;
drop table if exists employee;

create table Employee (eid int primary key, name text);

create table dba (eid int primary key references Employee,
                  platform text);

create table SE (eid int primary key references Employee,
                 language text);

create table Application (appId int primary key,
                          appName text,
                          manage int references se
                          );

create table develop (seId int references SE,
                      appId int references Application);
```

111. *(from CSE 414, Spring 2024, Midterm; CSE 344, Autumn 2024, Midterm)*

Consider the following E/R diagram:



(a) Write the CREATE TABLE statement needed to represent this in SQL The attributes `EID`, `PID`, `Budget`, `Level` are integers; `Name` is a text.

Write your answer here:

Solution:

```
create table employee
     (eid int primary key, name text);
create table manager
     (eid int primary key references employee, level int);
create table project
     (pid int primary key, budget int,
      auditor int not null references manager);
create table reports
     (eid int references employee, mid int references manager);
create table member
     (eid int references empolyee, pid int references project)
```

Solution: Yes

Solution: No

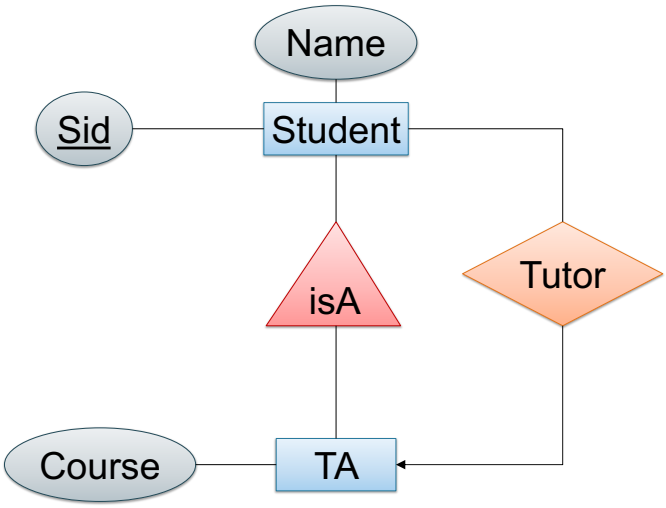
Solution: No

Solution: Yes

Solution: Yes

112. *(from CSE 414, Spring 2024, Final; CSE 344, Spring 2026, Makeup Final)*

(a) Consider the following E/R diagram:



Every student has an student ID and a `Name`. Some students are TA's for a course. `Sid` is an integer, `Name` and `Course` are text. Every student has at most one Tutor, who is one of the TA's. Write the CREATE TABLE statements to implement the E/R diagram in SQL. Your statements should contain all key and foreign key declarations.

Write your answer here:

Solution:

```
create table Student(sid int primary key, name text, tutor int references TA);
create table TA(sid int primary key references Student, course text);
```

This solution is acceptable for the purpose of the exam. In practice, it doesn't work because of the cyclic references, and one has to first create the Student table without the foreign key, then create the TA table, finally alter the Student table to add a foreign key. In class, we will simply pretend that cyclic references are OK.

(b) Consider a relation $R(A, B, C, D, E, F)$ satisfying the following FD's:

$$AB \rightarrow CD$$
$$BE \rightarrow F$$

Decompose R into BCNF. Show the normalized relations, and indicate the keys by underlining them. Write your answer here:

Solution:

- $AB^+ = ABCD$. Decompose into $R_1(\underline{A, B}, C, D)$, $R_2(A, B, E, F)$.
- $BE+ = BEF$ decompose $R_2(A, B, E, F)$ into $R_3(\underline{B, E}, F)$, $R_4(A, B, E)$.

Final and unique answer is: $R_1(\underline{A, B}, C, D)$, $R_3(\underline{B, E}, F)$, $R_4(A, B, E)$.

(c) We have a relation instance $R(A, B, C, D)$ that satisfies the functional dependency $AB \rightarrow C$. We run the following query and store its result in a new relation $S(A, B, C, D)$:

```
SELECT A, B, C, D
FROM R
WHERE (A = 5) and (B = D);
```

In each case below you are asked to indicate whether the relation S satisfies a given functional dependency. You only need to answer YES or NO.

i. Does S satisfy the FD $A \rightarrow B$? Yes or No?

Solution: NO

ii. Does S satisfy the FD $B \rightarrow A$? Yes or No?

Solution: YES

iii. Does S satisfy the FD $D \rightarrow A$? Yes or No?

Solution: YES

iv. Does S satisfy the FD $B \rightarrow C$? Yes or No?

Solution: YES

v. Does S satisfy the FD $C \rightarrow B$? Yes or No?

Solution: NO

(d) Answer the following questions:

i. What does Boyce Codd Normal Form mean? Choose one of:

1. All relations are flat (no nested sub-relations).
2. There are no data anomalies (redundancy/update/delete anomalies).
3. All functional dependencies are preserved.
4. It is equivalent to a serial schedule.

5. There are no foreign keys.

Solution: 2

ii. What does *First Normal Form* mean? Choose one of:
1. All relations are flat (no nested sub-relations).
2. There are no data anomalies (redundancy/update/delete anomalies).
3. All functional dependencies are preserved.
4. It is equivalent to a serial schedule.
5. There are no foreign keys.

Solution: 1

iii. What does *Physical Data Independence* mean? Choose one of:
1. The data is stored on a physical hard drive.
2. The user must unnest the SQL query.
3. The user writes queries over the logical schema, and the system optimizes the query for the physical execution.
4. All indices on the same table must be independent.
5. All functional dependencies are from a superkey.

Solution: 3

iv. Which relational algebra operators are monotone? Choose from the following list:
1. Union $\cup$
2. Selection σ
3. Difference $-$
4. Projection Π
5. Join $\bowtie$.

Your answer (e.g. 1,2,3):

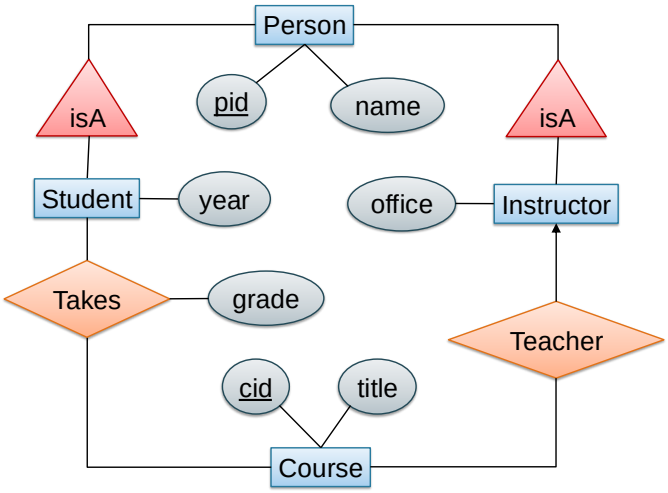
Solution: 1,2,4,5

v. If an attribute is key, then it is not allowed to also be a foreign key.
True or false?

Solution: False

113. *(from CSE 344, Autumn 2024, Final; CSE 344, Spring 2026, Final)*

(a) Consider the following E/R diagram:



The attributes have the following types:

- `pid, cid, year, grade` are integers.
- `name, office, title` are text.

Convert this E/R diagram to SQL. Your answer should consists of several CREATE TABLE statements.

Write your answer here:

Solution:

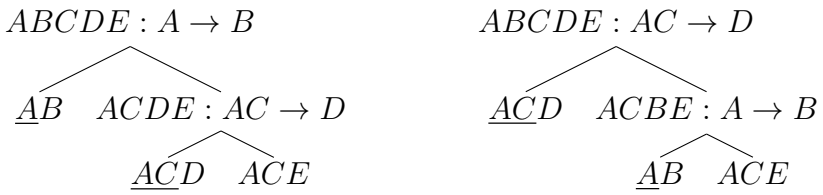
```
drop table if exists Takes;
drop table if exists Course;
drop table if exists Instructor;
drop table if exists Student;
drop table if exists Person;

create table Person(pid int primary key, name text);
create table Student(pid int primary key references Person,
                     year int);
create table Instructor(pid int primary key references Person,
                        office text);
create table Course(cid int primary key,
                    title text,
                    teacher int references Instructor);
create table Takes(pid int references Student,
                   cid int references Course,
                   grade int);
```

(b) Consider a relation $R(A, B, C, D, E)$ that satisfies the following Functional Dependencies:

$$A \to B$$
$$AC \to D$$

Decompose this relation in BCNF. Indicate the keys for each of your tables.
Write your answer here:

Solution:
Two decomposition steps, same result $R_1(\underline{A}, B)$, $R_2(\underline{A, C}, D)$, $R_3(A, C, E)$:



(c) Consider the following relational database schema:

```
Customer(cid, name, zip)
Orders(oid, cid, product, price)
```

Consider the following view $V(\texttt{zip}, \texttt{product}, \texttt{m})$:

Solution:

```
drop view if exists V;
```

```
create view V as
   select x.zip, y.product, count(*) as m
   from Customer x, Orders y
   where x.cid = y.cid and x.zip = 98195
   group by x.zip, y.product;
```

Compute the following closures of attributes in V:

i. $\texttt{zip}^+ =$

Solution: $\texttt{zip}$

ii. $\texttt{product}^+ =$

Solution: $\texttt{product}, \texttt{zip}, \texttt{m}$

iii. $\texttt{m}^+ =$

Solution: `m, zip`

iv. $(\texttt{zip}, \texttt{product})^+ =$

Solution: `zip, product, m`

v. $(\texttt{zip}, \texttt{m})^+ =$

Solution: `zip, m`

(d) Consider the following large relation instance:

A	B	C	D
0	0	0	0
1	0	0	1
2	1	0	2
3	1	1	0
4	2	1	1
5	2	1	2
6	3	2	0
7	3	2	1
8	4	2	2
9	4	3	0
...	...	...	...

The relation is large, only a fragment is shown. All values are positive integers. The attributes B, C, D are computed from A as follows:

$$\boxed{B := A/2, \;\; C := A/3, \;\; D := A\%3}$$

where the division is over integers, and $A\%3$ is modulo.

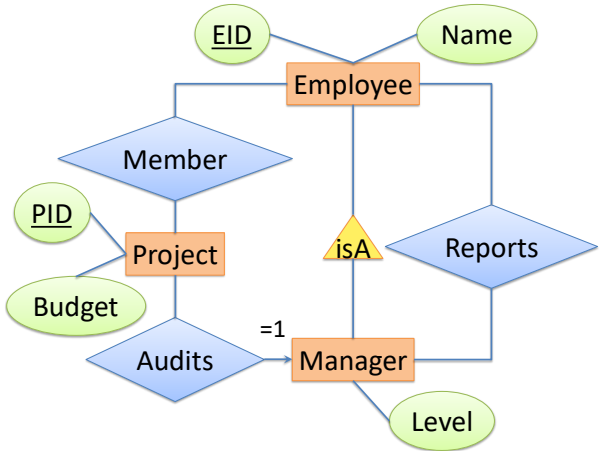
Write all Functional Dependencies that hold on the instance above. Your answer should be a list of FDs, for example $BC \to A$, $B \to C$ (not a real answer).

To write less, replace any two FDs $X \to Y$ and $X \to Z$ with a single FD $X \to YZ$, and omit redundant FDs, for example if you wrote both $X \to Y$ and $Y \to Z$ then $X \to Z$ is redundant, and if you wrote $X \to Y$ then $XZ \to Y$ is redundant.

Solution: $A \to BCD$
$CD \to A$
A subtle FD: $BD \to A$

114. *(from CSE 344, Spring 2026, Midterm)*

Consider the following E/R diagram:



(a) Write the CREATE TABLE statement needed to represent this in SQL The attributes `EID`, `PID`, `Budget`, `Level` are integers; `Name` is a text.

Write your answer here:

Solution:

```
drop table if exists member;
drop table if exists reports;
drop table if exists project;
drop table if exists manager;
drop table if exists employee;
create table employee
     (eid int primary key, name text);
create table manager
     (eid int primary key references employee, level int);
create table project
     (pid int primary key, budget int,
      auditor int not null references manager);
create table reports
     (eid int references employee, mid int references manager);
create table member
     (eid int references employee, pid int references project);
```

Solution: Yes

Solution: No

Solution: No

Solution: Yes

Solution: Yes

(b) Consider two tables that were created using the following SQL commands:

```
CREATE TABLE Country
  ( name text PRIMARY KEY,
    area int
  );
```

```
CREATE TABLE City
  ( name text PRIMARY KEY,
    located text REFERENCES Country,
    population int
  );
```

Notice that there is a primary key constraint in `Country`, and both a primary key and a foreign key constraint in `City`.

Indicate what does the database system need to check in each of the cases below:

i. `INSERT INTO Country VALUES ('France', 551500);`

Solution:
Check that no `Country` has `Name` = `'France'`

ii. `DELETE FROM Country WHERE name = 'Belgium';`

Solution:
Check that no `City` has `located` = `'Belgium'`

iii. `INSERT INTO City VALUES ('Rome', 'Italy', 2500000);`

Solution:
Check that there exists a `Country` with `name` = `'Italy'`
Check that no `City` has `Name` = `'Rome'`

iv. `DELETE FROM City WHERE name = 'Madrid';`

Solution:
Nothing

10 Functional Dependencies and Normal Forms (115–133)

This section is the most mechanical in the book, and also the most systematically tested. Given a relation and a set of functional dependencies, the standard tasks are to compute the closure of a set of attributes, to find the keys, to decide whether the schema is in BCNF, and if not, to decompose it.

The algorithms are worth executing carefully rather than by intuition. Attribute closure is a fixed-point computation; the keys are the minimal sets whose closure is everything: $X^+ =$ all attributes. A BCNF violation $X \to Y$ splits the relation into XY and X together with the remaining attributes. As discussed in class it's best to replace Y with the superset X^+, in other words "improve" the functional dependency from $X \to Y$ to $X \to X^+$ before splitting the relation. Decomposition is lossless exactly when the shared attributes form a key of one of the two fragments, and every BCNF decomposition is lossless. A BCNF decomposition not always preserve dependencies (which is the reason 3NF still exists).

Several questions run the process backwards, giving you the closed attribute sets or the decomposition and asking which dependencies could have produced it. Those reward genuine understanding rather than recall.

115. *(from CSE 444, Autumn 2000, Final)*

(a) Let X, Y be sets of attributes of a relational schema. Prove or disprove the following statements. In each case you have to either indicate "yes", or indicate "no" and provide a set of functional dependencies and sets of attributes X and Y that violate the statement.

i. $X \subseteq Y$ implies $X^+ \subseteq Y^+$.

Solution: Yes. If some attribute A is in X^+ then $X \to A$. Since $X \subseteq Y$ we also have $Y \to A$, hence $A \in Y^+$.

ii. $X^+ \cup Y^+ = (X \cup Y)^+$.

Solution: No. Consider $R(A, B, C)$ with the single FD $AB \to C$. Take $X = A$, $Y = B$. Then $X^+ = A$, $Y^+ = B$, so $X^+ \cup Y^+ = AB$, while $(X \cup Y)^+ = (AB)^+ = ABC$.

iii. $X^+ \cap Y^+ = (X \cap Y)^+$.

Solution: No. Consider $R(A, B, C, D)$ with the FDs $AB \to D$ and $AC \to D$. Take $X = AB$, $Y = AC$. Then $X^+ \cap Y^+ = ABD \cap ACD = AD$, while $(X \cap Y)^+ = A^+ = A$.

(b) Find all minimal keys in the relation below and find all non-trivial functional dependencies satisfied by this relation.

A	B	C	D
1	1	1	1
1	1	1	2
2	1	1	1
1	2	1	3
2	2	2	4

Solution: Computing X^+ for every subset X of $ABCD$:

$A^+ = A$	(rows 2 and 4 show $B \notin A^+$)
$B^+ = B$, $C^+ = C$	
$D^+ = BCD$	
$AB^+ = ABC$	(the first two rows show $D \notin AB^+$)
$AC^+ = AC$	(rows 1 and 4 show $B \notin AC^+$)
$AD^+ = ABCD$	this is a minimal key
$ABC^+ = ABC$	(the first two rows show $D \notin ABC^+$)
$ABD^+ = ACD^+ = ABCD$	(follows from AD being a key)
$BCD^+ = BCD$	(rows 1 and 3 show $A \notin BCD^+$)

There is only one minimal key: AD. A basis for all functional dependencies is $D \to BC$ and $AB \to C$.

116. *(from CSE 444, Autumn 2000, Midterm)*

Consider a relation $R(A, B, C, D, M, N, S)$ with functional dependencies

$$A \to M, \quad B \to M, \quad C \to N, \quad D \to N, \quad MN \to S, \quad CS \to A, \quad BS \to D$$

Answer the following questions:

(a) Compute AB^+.

Solution: $AB^+ = ABM$

(b) Compute AC^+.

Solution: $AC^+ = ACMNS$

(c) Compute BC^+.

Solution: $BC^+ = BCMNSAD$

(d) Give an example of a database instance in which all given functional dependencies hold but $AC \to B$ does not hold.

Solution:

A	B	C	D	M	N	S
a	b_1	c	d_1	m	n	s
a	b_2	c	d_2	m	n	s

117. *(from CSE 444, Autumn 2001, Final; CSE 444, Autumn 2002, Final)*

Answer the following questions. If you answer "yes" you don't have to justify your answer; if you answer "no" you need to give an example that justifies your answer.

(a) Suppose attribute A is a key in the relation R.

i. Is A a key in $\Pi_{ABC}(R)$?

Solution: Yes

ii. Is A a key in $\sigma_{C=55}(R)$?

Solution: Yes

iii. Is A a key in $R \cup S$?

Solution: No. Take

R

A	B	C
1	2	3

S

A	B	C
1	2	3
1	4	5

iv. Is A a key in $R - S$?

Solution: Yes

v. Is A a key in $R \bowtie T$?

Solution: No. Take R as above and

T

C	D
3	4
3	5

(b) Suppose R has attributes A and B and satisfies the functional dependency $A \to B$.

i. Does $\Pi_{ABC}(R)$ satisfy $A \to B$?

Solution: Yes

ii. Does $\sigma_{C=55}(R)$ satisfy $A \to B$?

Solution: Yes

iii. Does $R \cup S$ satisfy $A \to B$?

Solution: No — the same counterexample as in part (a) applies.

iv. Does $R - S$ satisfy $A \to B$?

Solution: Yes

v. Does $R \bowtie T$ satisfy $A \to B$?

Solution: Yes

118. *(from CSE 444, Autumn 2001, Midterm)*

Consider a relation $R(A, B, C, D)$ with functional dependencies

$$ABC \to D, \quad AD \to C, \quad BD \to A$$

Answer the following questions:

(a) Compute all keys.

Solution: The keys are ABC and BD.

(b) Decompose the relation in BCNF.

Solution: $AD \to C$ violates BCNF, so we decompose into $R_1(A, C, D)$ and $R_2(A, B, D)$. This is now in BCNF.

(c) Give an example of a database instance in which all given functional dependencies hold but $ACD \to B$ does not hold.

Solution:

A	B	C	D
a	b_1	c	d
a	b_2	c	d

119. *(from CSE 444, Autumn 2002, Midterm)*

Recall that S^+ denotes the closure of a set of attributes S under a given set of functional dependencies.

(a) Consider a relation $R(A, B, C, D)$ with functional dependencies

$$ABC \to D, \quad AD \to C, \quad BD \to A$$

i. Compute $(AD)^+$.

Solution: $(AD)^+ = ACD$

ii. Compute $(BD)^+$.

Solution: $(BD)^+ = ABCD$

iii. Find all keys in $R(A, B, C, D)$.

Solution: The minimal keys are BD and ABC; the keys are these plus all their supersets.

(b) A set S is called *closed* if $S^+ = S$. For each of the following statements, indicate whether it is TRUE or FALSE. If your answer is FALSE, then give an example database which shows that it is false.

i. If S_1 and S_2 are closed, then $S_1 \cup S_2$ is also closed.

Solution: FALSE. Take $R(A, B, C)$ with $AB \to C$. Then A is closed and B is closed, but AB is not closed.

ii. If S_1 and S_2 are closed, then $S_1 \cap S_2$ is also closed.

Solution: TRUE

(c) Does there exist a relation $R(A, B, C)$ where the closed sets are $\emptyset$, AB, AC, ABC? If YES, give an example of a set of functional dependencies for which the closed sets are precisely these; if NO, explain why.

Solution: NO. If AB and AC are closed then A must also be closed.

(d) Consider the relation `Course(name, instructor)`. Write an expression in the relational algebra that returns all instructors that teach only one course. You may

use only the five basic operators of the relational algebra, $\cup$, Π, σ, $\times$, $-$, as well as joins.

Solution:

$$\Pi_{instructor}(\texttt{Course}) \; - \; \Pi_{instructor}\big(\sigma_{name \neq n \wedge instructor = i}(\texttt{Course} \times \rho_{n,i}(\texttt{Course}))\big)$$

120. *(from CSE 444, Autumn 2003, Midterm)*

(a) Consider the following table:

Product	Department	Customer
P111	D222	C333
P111	D222	C555
P111	D444	C666
P222	D222	C555

Indicate which of the following functional dependencies hold:

Product $\rightarrow$ Department
Customer $\rightarrow$ Department
Department $\rightarrow$ Customer
Product, Department $\rightarrow$ Customer
Department, Customer $\rightarrow$ Product

Solution: Only Customer $\rightarrow$ Department holds.

(b) Consider the relational schema $ABCD$ and the following functional dependencies: $AB \rightarrow C$, $B \rightarrow D$, $C \rightarrow A$. Normalize the schema in BCNF.

Solution: In $ABCD$: $B^+ = BD$, so decompose into BD and BAC.
In BAC: $C^+ = AC$, so decompose into AC and BC.
The complete decomposition is BD, AC, BC.
An alternative route: in $ABCD$, $C^+ = CA$, decompose into CA and CBD; then in CBD, $B^+ = BD$, decompose into BD and BC — giving the same result CA, BD, BC.

121. *(from CSE 544, 2004, Final)*

(a) We say a set of attributes X is *closed* (with respect to a given set of functional dependencies) if $X^+ = X$. For each of the statements below, indicate whether they are true or false.

i. If X and Y are closed, then $X \cup Y$ is closed.

Solution: No

ii. If X and Y are closed, then $X \cap Y$ is closed.

Solution: Yes

(b) Consider a relation with schema $R(A, B, C, D)$ and an unknown set of functional dependencies. For each of the statements below, give a set of functional dependencies for which that statement holds, or say that no such set exists.

i. All sets of attributes are closed.

Solution: The empty set of functional dependencies.

ii. The only closed sets are $\emptyset$ and $\{A, B, C, D\}$.

Solution: $A \to B \to C \to D \to A$.

iii. The only closed sets are $\emptyset$, $\{A, B\}$, $\{B, C\}$, and $\{A, B, C, D\}$.

Solution: No such set exists, since $\{A, B\} \cap \{B, C\}$ must also be closed.

iv. The only closed sets are $\emptyset$, $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, and $\{A, B, C, D\}$.

Solution: $AB \to BC$, $BC \to AD$, $CD \to AB$, $AC \to BD$, $BD \to AC$, $AD \to BC$.

v. The only closed sets are $\emptyset$, $\{A, B\}$, and $\{A, B, C, D\}$.

Solution: $A \to B$, $B \to A$, $C \to A$, $D \to A$.

(c) Consider the relational schema $ABCD$ and the following functional dependencies: $AB \to C$, $B \to D$, $C \to A$. Normalize the schema in BCNF.

Solution: Use $B^+ = BD$ and split $ABCD \Rightarrow BD + BAC$; then use $C^+ = AC$ and split $BAC \Rightarrow AC + BC$. Final split: $ABCD \Rightarrow AC + BC + BD$.

(d) Consider the following table. Indicate which of the following functional dependencies hold. You only have to indicate 'yes' or 'no'.

Product	Department	Customer
P111	D222	C333
P111	D222	C555
P111	D444	C666
P222	D333	C555

i. `Department` → `Product`

Solution: Yes

ii. `Customer` → `Department`

Solution: No

iii. `Department` → `Customer`

Solution: No

iv. `Product`, `Department` → `Customer`

Solution: No

v. `Department`, `Customer` → `Product`

Solution: Yes

122. *(from CSE 444, Autumn 2004, Midterm; CSE 444, Autumn 2005, Midterm)*

(a) Consider the table $R(A, B, C, D, E)$ satisfying the following functional dependencies:

$$AB \to C, \quad BD \to A, \quad E \to BD$$

Decompose the table in BCNF. Your answer should consist of (1) a list of table names and attributes, and (2) an indication of the keys in each table.

Solution: In R: $AB^+ = ABC$, so split into $R_1(A, B, C)$ and $R_2(A, B, D, E)$.
In R_2: $BD^+ = ABD$, so split into $R_3(A, B, D)$ and $R_4(B, D, E)$.
Final answer:

$R_1(A, B, C)$ key $= AB$
$R_3(A, B, D)$ key $= BD$
$R_4(B, D, E)$ key $= E$

123. *(from CSE 444, Autumn 2005, Midterm)*

(a) For each of the statements below, indicate whether they are true or false. You do not need to justify your answer.

i. For any two sets X, Y: $(X \cup Y)^+ = X^+ \cup Y^+$.

> **Solution:** False. Example: $AB \to C$. Then $A^+ = A$, $B^+ = B$, but $AB^+ = ABC$.

ii. For any two sets X, Y: if $X \subseteq Y$ then $X^+ \subseteq Y^+$.

> **Solution:** True.

iii. For any two sets X, Y: if $X \subset Y$ then $X^+ \subset Y^+$.

> **Solution:** False. Example: $A \to B$. Then $A^+ = AB$ and $AB^+ = AB$.

iv. For any two sets X, Y: if $X^+ = X$ and $Y^+ = Y$ then $(X \cap Y)^+ = X \cap Y$.

> **Solution:** True.

124. *(from CSE 444, Winter 2006, Midterm)*

(a) Consider the table $R(A, B, C, D, E)$ satisfying the following functional dependencies:

$$AB \to C, \quad BE \to A, \quad D \to E$$

Decompose the table in BCNF. Your answer should consist of (1) a list of table names and attributes, and (2) an indication of the keys in each table.

> **Solution:** One decomposition is $R_1(B, E, A)$, $R_2(D, E)$, $R_3(D, B, C)$.
> Other valid answers are $R_1(A, B, C)$, $R_2(D, E)$, $R_3(D, B, A)$ — or $R_1(A, B, C)$, $R_2(B, E, A)$, $R_3(D, E)$, $R_4(D, B)$.

(b) Let `FD` be a set of functional dependencies on a table, and let X^+ denote the closure of a set of attributes X. For each of the statements below, indicate whether they are true or false.

i. For any two sets X, Y: $(X \cup Y)^+ = X^+ \cup Y^+$.

> **Solution:** False

ii. For any two sets X, Y: if $X \subseteq Y$ then $X^+ \subseteq Y^+$.

Solution: True

iii. For any two sets X, Y: if $X^+ = X$ and $Y^+ = Y$ then $(X \cap Y)^+ = X \cap Y$.

Solution: True

(c) Now suppose that all functional dependencies in `FD` are of the form $A \to B$, where A and B are single attributes. That is, the left hand side of each functional dependency consists of a single attribute. Which of the answers in (b) change?

Solution: All three statements now hold: True, True, True.

125. *(from CSE 444, Autumn 2006, Midterm)*

(a) X, Y are sets of attributes. Indicate for each statement below whether it is true or false.

i. If $X \to Y$ then the set X^+ contains Y^+.

Solution: True

ii. If $X, Y \to Z$ then $X \to Z$.

Solution: False. Example: $AB \to C$ does not imply $A \to C$.

iii. If $X \to Y$ and $Y, U \to V$ then $X, U \to V$.

Solution: True

iv. If the set X^+ is contained in Y^+ then X is contained in Y.

Solution: False. Example: $A \to B$. Take $X = B$, $Y = A$. Then $X^+ = B$ is contained in $Y^+ = AB$, yet X is not contained in Y.

v. Consider two relations $R(A, B, C, D)$ and $S(C, D, E, F)$, and suppose the following functional dependencies hold:

In R:	$A \to C$	$B \to D$	$D \to B$
In S:	$C, E \to F$	$G \to E$	

Now consider the following table $T(A, B, C, D, F, G)$ defined by a SQL view:

```
create  view T as
select  distinct R.A, R.B, R.C, R.D, S.E, S.F
from   R, S
where R.B = 'bbb' and R.C = S.C and R.D = S.D
```

Compute all the keys in T.

Solution: The FDs satisfied by T are all FDs satisfied by R and by S, plus $X \to B$ for every attribute X (because B has a single value in the view) and, because of $B \to D$, every attribute also implies D: $X \to D$ for all X. The key is AG because $A \to C$ (in R), $CG \to CGE \to CGEF$ (in S) hence $AG \to ABCDEFG$.

(b) Decompose the table $R(A, B, C, D, E)$ in BCNF, assuming the following functional dependencies hold on R:

$$A, B, C \to D \qquad B, D \to A \qquad B, E \to A$$

Show your steps.

Solution: One decomposition proceeds as follows: $R_1 = ABCD$, $R_2 = ABCE$ (ruling out $ABC \to D$); then $R_3 = ABD$, $R_4 = ABC$, $R_2 = ABCE$; then $R_3 = ABD$, $R_4 = ABC$, $R_5 = ABE$, $R_6 = BCE$.

Other valid decompositions are ABD, $BCDE$ — or ABE, $BCDE$.

126. *(from CSE 444, Spring 2007, Midterm)*

Consider a table $R(A, B, C, D)$. Recall that a set of attributes X is a superkey if $X^+ = ABCD$; a set X is a key if X is a superkey and no subset of X is a superkey; it is closed if $X^+ = X$.

(a) Give a set of functional dependencies that satisfies the following conditions: the closed sets are AB, CD, and the keys are AD and BC.

Solution:

$$\begin{aligned} A &\to B \\ B &\to A \\ C &\to D \\ D &\to C \\ AD &\to BC \\ BC &\to AD \end{aligned}$$

(b) For each of the statements below indicate if they are true or false. You need to answer only "true" or "false" and do not need to justify your answer. X and Y denote sets of attributes.

i. In a relation with attributes A, B, C, D: if AB is a key then ABC cannot be closed.

> **Solution:** True, because $AB \to D$ (it's a key) hence $ABC \to D$.

ii. In a relation with attributes A, B, C, D: if AB is closed then ABC cannot be a key.

> **Solution:** False. Example: $ABC \to D$.

iii. If X, Y are closed then $X \cup Y$ is closed.

> **Solution:** False. Example $AB \to C$. Then A is closed, B is closed, but AB is not closed.

iv. If X, Y are closed then $X \cap Y$ is closed.

> **Solution:** True.

(c) Consider the following FDs:

$$\begin{aligned} B &\to A \\ C &\to B \end{aligned}$$

Decompose the table in Boyce-Codd Normal Form (BCNF). Show your decomposition steps.

> **Solution:** Call the relation $R(A, B, C)$.
> $B^+ = AB$ violates BCNF. We decompose R into $R_1(A, \underline{B})$ and $R_2(B, \underline{C})$. This schema is in BCNF because very relation with 2 attributes is already in BCNF.

127. *(from CSE 544p, Spring 2009, Final)*

(a) Consider the following three relations, their attributes, and their keys:

$$R(\underline{A}, B, C)$$
$$S(\underline{D}, E)$$
$$T(\underline{F}, G, H)$$

For each of the views $V_1, \ldots, V_5$ below indicate their attributes, and list a set of functional dependencies (FDs) such that all the FDs that hold in the view can be inferred from these.

$$\begin{aligned} V_1 &= R \bowtie_{B=D} S \\ V_2 &= S \bowtie_{E=G} T \\ V_3 &= \sigma_{H=55}(T) \\ V_4 &= \Pi_{AB}(R) \\ V_5 &= \gamma_{H,sum(G) \text{ as } K}(T) \end{aligned}$$

For example, your answer could look as follows:

$$V_9(A, B, D, F, G) \qquad AB \to DFG; \quad F \to B$$

Solution:

$$\begin{aligned} V_1(A, B, C, D, E) &\quad A \to BCDE, B \to DE, D \to B \\ V_2(D, E, F, G, H) &\quad D \to E, F \to GH, E \to G, G \to E \\ V_3(F, G, H) &\quad F \to G, G \to H \\ V_4(A, B) &\quad A \to B \\ V_5(G, K) &\quad H \to K \end{aligned}$$

(b) Consider a table $R(A, B, C, D, E, F)$ and the following functional dependencies:

$$\begin{aligned} BC &\to A \\ BDE &\to F \\ EF &\to B \end{aligned}$$

Compute a BCNF representation of R. Explain your steps.

Solution:

1. In R: $BC+ = ABC$. Split R into $R_1(A, B, C)$, $R_2(B, C, D, E, F)$. R_1 is in BCNF.

2. In R_2: $BDE+ = BDEF$. Split R_2 into $R_3(B, D, E, F)$ and $R_4(B, D, E, C)$. R_4 is in BCNF.

3. In R_3: $EF+ = BEF$. Split R_3 into $R_5(E, F, B)$ and $R_6(E, F, D)$. Both are in BCNF.

Final answer: $R_1(\underline{A, B}, C)$, $R_4(\underline{B, D, E, C})$, $R_5(\underline{E, F}, B)$, $R_6(\underline{E, F, D})$.

(c) The MomPopDairy Company has a database with a single table:
`Orders(customerID, customerName, customerAddress, milkQuantity, date)`
In their current operation, every time a customer calls to place an order for milk delivery, the seller enters a new record in `Orders` with all the customer information. The company has operated for about ten years, and during this time there have been about 10,000 records inserted in `Orders`. Now, the owners want to create a Web interface to allow customers to place their own orders online: existing customers will provide their customer ID then can place their order, while new customers will provide their name and address and will be assigned a customer ID that they can use for future orders. The company owners would like to keep all the old data, but would also like to enable the new application. They hire you as a database consultant to advise them on how to manage their database so that it can support the new Web application. You observe quickly that their schema is not in normal form. Advise the MomPopDairy Company on their options. Give them two options on how to design the Web application and/or restructure the database, explaining the immediate costs versus long term benefits tradeoff. Keep your explanations short: for each of the two option, given your answer in 1-3 sentences, then briefly enumerate the pros and cons.

Solution:

1. Keep the database schema unchanged, catch and resolve update anomalies in the application: if a customer already exists, find their latest order to retrieve their name and address. Pros: old data remains intact. Cons: the application logic is complex and error prone.

2. Normalize the database schema, by splitting it into `Orders` and `Customers`. Cons: the old data needs to be restructured. Pros: the application is easier to write and maintain.

128. *(from CSE 414, Spring 2024, Midterm)*

(a) Consider the table below:

Meal	Taste	Temperature	Vegetarian
A	sweet	cold	veg
B	sweet	warm	nonVeg
C	savory	cold	nonVeg
D	savory	warm	veg

Indicate which of the following functional dependencies hold:

i. `Meal` $\rightarrow$ `Taste`, `Temperature`, `Vegetarian`.
Yes or no?

Solution: Yes

ii. `Vegetarian` $\rightarrow$ `Taste`
Yes or no?

Solution: No

iii. `Taste` $\rightarrow$ `Vegetarian`
Yes or no?

Solution: No

iv. `Taste`, `Temperature` $\rightarrow$ `Vegetarian`
Yes or no?

Solution: Yes

v. `Temperature`, `Vegetarian` $\rightarrow$ `Taste`
Yes or no?

Solution: Yes

(b) Consider a relation with four attributes $R(A, B, C, D)$ satisfying the following set of FDs:

$$AB \rightarrow C$$
$$BD \rightarrow A$$

Compute the closure of the sets below:

i. $A^+ =$

Solution: A

ii. $B^+ =$

Solution: B

iii. $AB^+ =$

Solution: ABC

iv. $AD^+ =$

Solution: AD

v. $BD^+ =$

Solution: $ABCD$

129. *(from CSE 414, Spring 2024, Midterm Review)*

Consider the following same relational schema, where all attributes are integers:

R(A,B,C)
S(A,B,D)

Solution:

```
-- for the instructor only

create table R(A int, B int, C int, primary key (A,B));
create table S(A int, B int, D int, primary key (A));
```

(a) Let `Q1(A,B,C)` be the relation returned by the following query:

```
select x.A, x.B, x.C
from R x
where x.B = 20;
```

i. Compute the closure $A+$ in the relation `Q1(A,B,C)`.

Solution: ABC

ii. Compute the closure $B+$ in the relation `Q1(A,B,C)`.

Solution: B

iii. Compute the closure $C+$ in the relation `Q1(A,B,C)`.

Solution: BC

(b) Let `Q2(A,B,D)` be the relation returned by the following query:

```
select y.A, y.B, y.D
from S y
where y.B = y.D;
```

i. Compute the closure $A+$ in the relation `Q2(A,B,D)`.

Solution: ABD

ii. Compute the closure $B+$ in the relation `Q2(A,B,D)`.

Solution: BD

iii. Compute the closure $D+$ in the relation `Q2(A,B,D)`.

Solution: BD

(c) Let `Q3(A,B,C,D)` be the relation returned by the following query:

```
select x.A, x.B, x.C, y.D
from R x, S y
where x.A = y.A and x.B = y.B;
```

i. Compute the closure $A+$ in the relation `Q3(A,B,C,D)`.

Solution: $ABCD$

ii. Compute the closure $B+$ in the relation `Q3(A,B,C,D)`.

Solution: B

iii. Compute the closure $C+$ in the relation `Q3(A,B,C,D)`.

Solution: C

iv. Compute the closure $D+$ in the relation `Q3(A,B,C,D)`.

Solution: D

130. *(from CSE 344, Autumn 2024, Midterm)*

(a) Consider the table below:

Meal	Taste	Temperature	Type
A	sweet	cold	vegan
B	sweet	warm	meat
C	savory	cold	veggie
D	savory	warm	veggie

Indicate which of the following functional dependencies hold:

i. `Meal` $\rightarrow$ `Taste`, `Temperature`, `Type`.
Yes or no?

Solution: Yes

ii. `Type` $\rightarrow$ `Taste`
Yes or no?

Solution: Yes

iii. `Taste` $\rightarrow$ `Type`
Yes or no?

Solution: No

iv. `Taste`, `Temperature` $\rightarrow$ `Type`
Yes or no?

Solution: Yes

v. `Type` $\rightarrow$ `Temperature`
Yes or no?

Solution: No

(b) Consider a relation with four attributes $R(A, B, C, D)$ satisfying the following set of FDs:

$$AB \rightarrow C$$
$$BD \rightarrow A$$

Compute the closure of the sets below:

i. $A^+ =$

Solution: A

ii. $B^+ =$

Solution: B

iii. $AB^+ =$

Solution: ABC

iv. $AD^+ =$

Solution: AD

v. $BD^+ =$

Solution: $ABCD$

131. *(from CSE 344, Spring 2026, Midterm)*

(a) Consider the table below:

Meal	Taste	Temperature	Type
A	sweet	cold	vegan
B	sweet	warm	meat
C	savory	cold	veggie
D	savory	warm	veggie

Indicate which of the following functional dependencies hold:

i. `Meal` $\rightarrow$ `Temperature`.
Yes or no?

Solution: Yes

ii. `Type` $\rightarrow$ `Taste`
Yes or no?

Solution: Yes

iii. `Taste` $\rightarrow$ `Type`
Yes or no?

Solution: No

iv. `Taste, Temperature` $\rightarrow$ `Type`
Yes or no?

Solution: Yes

v. `Type` $\rightarrow$ `Temperature`
Yes or no?

Solution: No

(b) Consider a relation with four attributes $R(A, B, C, D)$ satisfying the following set of FDs:

$$AB \rightarrow C$$
$$BD \rightarrow A$$

Compute the closure of the sets below:

i. $A^+ =$

Solution: A

ii. $B^+ =$

Solution: B

iii. $AB^+ =$

Solution: ABC

iv. $AD^+ =$

Solution: AD

v. $BD^+ =$

Solution: $ABCD$

(c) Consider a relation with three attributes: $R(A, B, C)$. All three attributes are integers. The relation R has no key, and no functional dependencies hold in R. We run the following SQL query:

```
CREATE TABLE S AS
   SELECT A,B,C, A+B+C as D
   FROM R;
```

This creates a new relation $S(A, B, C, D)$, where all four attributes are integers. List all the functional dependencies that hold in S.

Write your answer here:

Solution: $ABC \to D$, $ABD \to C$, $ACD \to B$, $BCD \to A$.

132. *(from CSE 344, Spring 2026, Midterm)*

(a) Consider a relation $R(A, B, C, D)$ that satisfies the following FDs:
$A \to B$, $AB \to C$, $C \to B$.

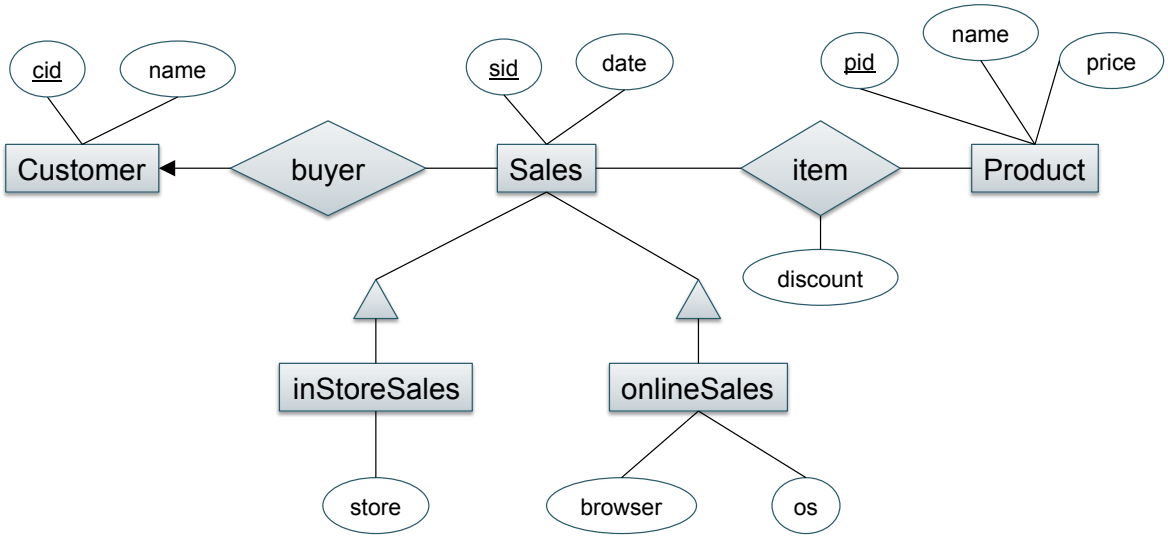
Decompose the relation in Boyce-Codd Normal Form. Show the keys of your decomposed relations by underlining them.

Write your answer here:

Solution: Use $A+ = ABC$ and decompose into $T_1(\underline{A}BC), T_2(AD)$.
In $T_1(\underline{A}BC)$: use $C+ = CB$ and decomposed into $T_3(\underline{C}B)$ and $T_4(C\underline{A})$.
Final answer: $T_2(AD), T_3(\underline{C}B), T_4(C\underline{A})$.

133. *(from CSE 344, Winter 2016, Final)*

A company selling products both online and in their own stores has a database having a schema described by the E/R diagram below:



- `Sales` represents individual sales. One sale contains several products bought by one single customer. The `discount` is a real number representing a percent, for example `discount` = 33.33 means a discount of 1/3 from the original product price.
- `inStoreSales` consists of the sales in brick-and-mortar stores; each such sale contains the store name.
- `onlineSales` consists of online sales; each such sale includes the name of the browser and the operating system used during the purchase.
- `cid, sid, pid` are `int`, `price` and `discount` are `float`, the rest are `text`.
- No attributes may be null, except for `inStoreSales.store` and `onlineSales.os`.

(a) Create a database schema for the E/R diagram in the figure. You should turn an a set of CREATE TABLE statements. Your schema should include all constraints captured in the E/R diagrams.

Solution:

```
drop table if exists Item;
drop table if exists onlineSales;
drop table if exists instoreSales;
drop table if exists Sales;
drop table if exists Product;
```

```
drop table if exists Customer;

create table Customer (cid int primary key, name text not null);
create table Product
   (pid int primary key,
    name text not null,
    price float not null);
create table Sales (
     sid int primary key,
     date text not null,
     cid int not null references customer);
create table inStoreSales (
     sid int primary key references Sales,
     store text);
create table onlineSales (
     sid int primary key references Sales,
     browser text not null,
     os text);
create table item (
     pid int references Product,
     sid int references Sales,
     discount float not null,
     primary key (pid, sid));
```

(b) Write a SQL query that computes, for each customer, the total amount that the customer spent online. Your query does not need to include customers who did not purchase anything online.

Solution:

```
select v.cid, v.name, sum(u.price * (100-z.discount)/100)
from onlineSales x, Sales y, Item z, Product u, Customer v
where x.sid = y.sid
  and y.sid = z.sid
  and z.pid = u.pid
  and y.cid = v.cid
group by v.cid, v.name;
```

(c) Consider three relations:

$$R(\underline{A, B}, C), S(C, \underline{D}), T(D, A)$$

AB is a key in R, and D is a key in S. For each of the queries below, show the key of the query's answer, and compute $D+$:

i. Query $Q1$:

```
select R.A, R.B, R.C, S.D
from R, S
where R.C = S.C and R.A = 20;
```

Key=? $D+ =?$

Solution: Key=BD, $D+ = ACD$

ii. Query $Q2$:

```
select T.A, S.C, S.D
from S, T
where S.D = T.D;
```

Key=? $D+$ =?

Solution: Key=AD, $D+ = CD$

iii. Query $Q3$:

```
select R.A, R.B, R.C, S.D
from R, S, T
where R.A=T.A and R.C = S.C and S.D = T.D;
```

Key=? $D+$ =?

Solution: Key=ABD, $D+ = CD$

(d) Consider a relation $R(A_1, A_2, \ldots, A_n)$ satisfying the following functional dependencies:

$$\begin{aligned} A_1 &\rightarrow A_2 \\ A_2 &\rightarrow A_3 \\ A_3 &\rightarrow A_4 \\ &\ldots \\ A_{n-1} &\rightarrow A_n \end{aligned}$$

Decompose this relation into BCNF. You need to indicate only your answer by showing the relation names, their attributes, and their key, for example you may write:

$$R_1(\underline{A_2}, A_3, A_4 \ldots, A_n), R_2(A_1, \underline{A_3}, A_4 \ldots, A_n), R_2(A_1, A_2, \underline{A_4} \ldots, A_n), \ldots, R_n(\underline{A_1}, A_2, \ldots, A_{n-1})$$

(not the real answer).

Solution:

$$R_1(\underline{A_1}, A_2), R_2(\underline{A_2}, A_3), \ldots, R_{n-1}(\underline{A_{n-1}}, A_n)$$

11 Indexes and Physical Design (134–141)

Indexes are where the logical schema meets the disk. The problems here ask you to trace insertions and deletions through a B+ tree, to compute the number of I/Os a lookup requires, and to choose the indexes that a given workload deserves.

The clustered versus unclustered distinction runs through nearly all of them, and it dominates the cost: a clustered index visits each data block at most once, while an unclustered one may visit a block per matching record. For a given query, a covering index is one that can be used to answer the query by avoiding the table entirely, and is often the cheapest option to answer a query.

Keep the height of the tree, the fanout, and the number of matching records separate in your arithmetic, and state the assumptions you make about each.

134. *(from CSE 544, 2001, Final)*

Consider a B+-tree of order 2 (i.e. each internal node has between 2 and 4 keys). We start with the "empty B+ tree", consisting of one leaf node with no keys, and one root with that leaf as the single child (i.e. two nodes in total). Consider inserting the keys $1, 2, 3, \ldots, n$ in some order.

(a) What is the maximum possible number of node splits that we have to do? Your answer should consist of a formula in n.

Solution: We have to count the largest number of nodes that are used by n keys. Writing $\lceil x \rceil$ for "round x up to the nearest integer": there are $\lceil n/2 \rceil$ leaves; there are x internal nodes, each with 3 outgoing edges except the root which has only two, giving $3x - 1$ outgoing edges. The edges' destinations are $x - 1$ internal nodes and $\lceil n/2 \rceil$ leaves, hence $x = \lceil \lceil n/2 \rceil / 2 \rceil$. The total number of splits is

$$\lceil n/2 \rceil - 1 + \lceil \lceil n/2 \rceil / 2 \rceil - 1 \approx n/2 + n/4 - 2 = 3n/4 - 2$$

(b) What is the minimum possible number of node splits?

Solution: We have to count the smallest number of nodes used by n keys. As above: $\lceil n/4 \rceil$ leaves and $\lceil (\lceil n/4 \rceil - 1)/4 \rceil$ internal nodes. Hence the total number of splits is

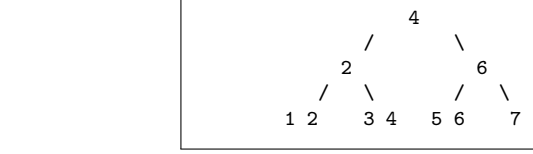
$$\lceil n/4 \rceil - 1 + \lceil (\lceil n/4 \rceil - 1)/4 \rceil - 1 \approx n/4 + n/16 - 1/4 - 2 = 5n/16 - 9/2$$

135. *(from CSE 544, 2002, Final)*

(a) Assume a B+ tree with $d = 1$, i.e. each node has between 1 and 2 keys. Starting from an empty tree, insert the keys $1, 2, 3, \ldots, 16$ in this order. Show at least the following four trees: after inserting 5, after inserting 7, after inserting 14, and after inserting 16.

Solution: After inserting 5:

After inserting 7:



(b) We sort a file using multi-way merge sort. The main memory can store $M = 101$ pages. A record is as large as one block (= page). There are 10000 records, with the following keys (in this order):

```
1 2 ... 99 100000 101 102 ... 199 200000 201 ... 299 300000
    ... 9901 9902 ... 9999 10000000
```

These are all keys $1, 2, \ldots, 10000$ in sequence, with every 100th key multiplied by 1000. The sorted file will be

```
1 2 ... 99 101 102 ... 199 201 ... 299 ... 9901 ... 9999
    100000 200000 300000 ... 10000000
```

How many times do we need to read and write each record from/to disk if (i) we use quicksort during run formation, and (ii) we use replacement selection during run formation?

Solution: (i) During the first pass, quicksort produces runs of length 100: there will be 100 such runs. A second pass is needed to merge the runs and output the sorted sequence. Hence each record is read twice and written twice.

(ii) During the first pass, replacement selection produces a single run of length 10000. There is no need for a second pass. Hence each record is read once and written once.

(c) Consider two tables $R(A, B)$ and $S(C, D)$, and the selection-join operation $\sigma_{A=v}(R) \bowtie_{B=C} S$. Assume that a record is as large as a block and that $T(R) = B(R) = B(S) = 10000$. There is a clustered index on $S.C$, and we need four disk accesses to look up a value in that index (including the disk access to get the record in S). We consider the following methods for the join: block nested loop join, partitioned hash join, index join. In each of the four cases below, indicate the best access method and its cost.

	$V(R, A) = 1$	$V(R, A) = 10000$
$M = 101$		
$M = 5002$		

Solution:

	$V(R, A) = 1$	$V(R, A) = 10000$
$M = 101$	index join $c = 10k + 4 \cdot 10k = 40000$	index join $c = 10k + 4 = 10004$
$M = 5002$	block nested loop $c = 10k + 2 \cdot 10k = 30000$	index join $c = 10k + 4 = 10004$

Full credit was also given for

	$V(R, A) = 1$	$V(R, A) = 10000$
$M = 101$	partitioned hash join $c = 3(10k + 10k) = 60000$	index join $c = 5k + 4 = 5004$
$M = 5002$	block nested loop $c = 10k + 2 \cdot 10k = 30000$	index join $c = 5k + 4 = 5004$

The index join here assumes more than typical optimizers do, namely that $R.A$ is a key.

136. *(from CSE 444, Autumn 2008, Final)*

Consider two indexes on the relation $R(A, B, C, D)$:

- clustered index on (A, B) called `CI`
- unclustered index on (B, C) called `UI`

For each query below indicate which of the following is the most efficient way to execute the query: (i) by using index `CI`, (ii) by using index `UI`, (iii) either by using `CI` or `UI` depending on the data, (iv) not using an index at all. If no index is usable for the query, indicate so.

(a)

```
SELECT * FROM R WHERE R.B = 55 and R.D = 99
```

Solution: Unclustered index `UI`.

(b)

```
SELECT * FROM R WHERE R.A = 22 and R.D = 99 and R.C = 33
```

Solution: Clustered index `CI`.

(c)

```
SELECT * FROM R WHERE R.A > 11 and R.A < 22 and R.B = 55 and R.C = 33
```

Solution: Either clustered or unclustered, depending on the data.

137. *(from CSE 544p, Spring 2009, Final)*

Consider a relation `R(A,B,C)`. Assume the following statistics:

$$\begin{aligned} B(R) &= 1000 \\ T(R) &= 10000 \\ V(R,A) &= 20 \end{aligned}$$

Thus, on average 10000/1000 = 10 records `(A,B,C)` fit in one block. There are two indexes on `R`:

- A $B+$-tree, clustered index on `A`
- A $B+$-tree, unclustered index on `B,A`. On average, a leaf nodes contains 250 `(B, A, TID)` triples, where TID is a tuple id.

Consider the query:

```
select B
from R
where A='abcd'
```

Indicate the I/O cost for each of the physical plans below. You may assume that the depth of each $B+$ tree is 3, that is, the query processor needs to read two blocks before reaching a leaf node of the $B+$ tree.

(a) Scan sequentially the table `R`, apply the predicate `A='abcd'` on-the-fly.

(b) Use the clustered index on `A` to retrieve the records `A='abcd'`.

(c) Use the unclustered index as a "covering index" (that is, the query is answered by reading *only* the unclustered index, and without reading the table `R`; see the lecture notes).

Solution:

1. 1000.
2. Access the clustered index on `A='abcd'`: $2 + 1000/20 = 52$.
3. Scan the entire unclustered index, apply the predicate `A='abcd'` on the fly: $2 + 10000/250 = 42$.

138. *(from CSE 444, Spring 2010, Final)*

Consider the following database about word occurrences in Webpages:

```
Webpage(url, author)
Occurs(url, wid)
Word(wid, text, language)
```

where:

- `Webpage.url` and `Word.wid` are keys.
- `Occurs.url` and `Occurs.wid` are foreign keys to `Webpage` and `Word` respectively.

Assume the following statistics

$$\begin{aligned} T(\texttt{Webpage}) &= V(\texttt{Occurs}, \texttt{url}) = 10^9 \\ T(\texttt{Occurs}) &= 10^{12} \\ T(\texttt{Word}) &= V(\texttt{Occurs}, \texttt{wid}) = 10^6 \\ V(\texttt{Webpage}, \texttt{author}) &= 10^7 \\ V(\texttt{Word}, \texttt{language}) &= 100 \end{aligned}$$

Assume ten records can be fit in one block, hence $B(\texttt{Webage}) = T(\texttt{Webpage})/10$ and similarly for all other tables.

(a) Consider the following plan:

$$(\sigma^{\texttt{index-lookup}}_{\texttt{author='John'}}(\texttt{Webpage}) \bowtie^{\texttt{index-join}}_{\texttt{url=url}} \texttt{Occurs})$$

$$\bowtie^{\texttt{main-memory-hash-join}}_{wid=wid} \sigma^{\texttt{index-lookup}}_{\texttt{language='French'}}(\texttt{Word})$$

Compute the cost of the plan in each of the following cases:

1. We have the following indexes:

$$\begin{aligned} \texttt{Webpage.url} &= \text{primary index} \\ \texttt{Webpage.author} &= \text{secondary index} \\ \texttt{Occurs.url} &= \text{secondary index} \\ \texttt{Occurs.wid} &= \text{primary index} \\ \texttt{Word.wid} &= \text{primary index} \\ \texttt{Word.language} &= \text{secondary index} \end{aligned}$$

Solution:

$$\begin{aligned} \text{Unclustered Index Lookup on Author} &= 10^2 \\ \text{Unclustered Index Join} &= 10^2 * 10^3 \\ \text{Unclustered Index Lookup on Language} &= 10^4 \\ \text{Main Memory Hash Join} &= 0 \\ \\ \text{Total} &= 10^2 + 10^5 + 10^4 \end{aligned}$$

2. We have the following indexes:

$$\begin{aligned} \texttt{Webpage.url} &= \text{secondary index} \\ \texttt{Webpage.author} &= \text{primary index} \\ \texttt{Occurs.url} &= \text{primary index} \\ \texttt{Occurs.wid} &= \text{secondary index} \\ \texttt{Word.wid} &= \text{secondary index} \\ \texttt{Word.language} &= \text{primary index} \end{aligned}$$

Solution:

$$\begin{aligned} \text{Clustered Index Lookup on Author} &= 10 \\ \text{Clustered Index Join} &= 10^2 * 10^2 \\ \text{Clustered Index Lookup on Language} &= 10^3 \\ \text{Main Memory Hash Join} &= 0 \\ \\ \text{Total} &= 10 + 10^4 + 10^3 \end{aligned}$$

(b) Consider the following plan:

$$(\texttt{Webpage} \bowtie^{\texttt{merge-join}}_{\texttt{url=url}} \texttt{Occurs}) \bowtie^{\texttt{merge-join}}_{wid=wid} \texttt{Word}$$

Choose a set of indexes that minimizes the total number of disk I/O's for the plan. Each index should be on a single attribute, and you can choose as many (or as few) indexes as you want. Indicate for each index if it is primary or secondary. For each merge-join we assume to have sufficient main memory to complete it in two pass. (There are no index-join operators in this plan: you need to figure out how indexes can help at all in this query plan.)

Solution: Primary Index on Webpage(url) Primary Index on Occurs(url) Primary Index on Word(wid) These clustered indexes allow the two-pas merge join to skip the initial sorting step for each of the tables.

139. *(from CSE 544p, Autumn 2010, Final)*

Consider the following database about word occurrences in Webpages:

```
Webpage(url, author, text)
Occurs(url, wid)
Word(wid, text, language)
```

where:`Occurs.url` and `Occurs.wid` are foreign keys to `Webpage` and `Word` respectively. This problem is about the execution of the following query, retrieving all Webpages written by 'John' that contain some French words:

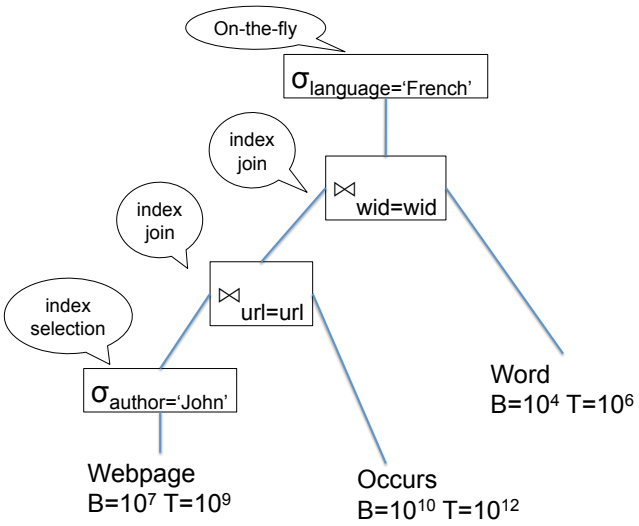
```
select *
from Webpage x, Occurs y, Word z
where x.url = y.url and y.wid = z.wid
  and x.author = 'John' and z.language = 'French'
```

Assume the following statistics

$$
\begin{aligned}
T(\texttt{Webpage}) = V(\texttt{Occurs}, \texttt{url}) &= 10^9 && \text{/* distinct URL's */}\\
V(\texttt{Webpage}, \texttt{author}) &= 10^7 && \text{/* about } 10^2 \text{ webpages/author */}\\
B(\texttt{Webpage}) &= 10^7 && \text{/* } 10^2 \text{ records/block */}\\
T(\texttt{Occurs}) &= 10^{12} && \text{/* about } 10^3 \text{ words/webpage */}\\
B(\texttt{Occurs}) &= 10^{10} && \text{/* } 10^2 \text{ records/block */}\\
T(\texttt{Word}) = V(\texttt{Occurs}, \texttt{wid}) &= 10^6 && \text{/* distinct words */}\\
V(\texttt{Word}, \texttt{language}) &= 100 && \text{/* about } 10^4 \text{ words in each language */}\\
B(\texttt{Word}) &= 10^4 && \text{/* } 10^2 \text{ records/block */}
\end{aligned}
$$

Assume that all indexes fit in main memory.

(a) Consider the following plan:



The plan uses three indexes, `Webpage.author`, `Occurs.url`, and `Word.wid`. Each index may be clustered, or unclustered: in each case, compute the cost of the plan.

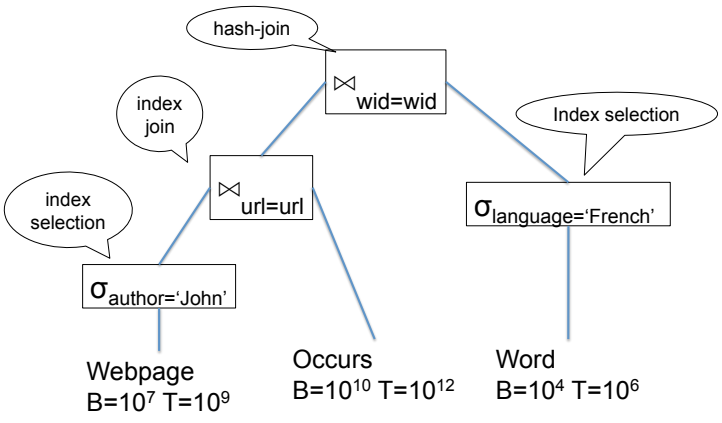
`Webpage.author`	`Occurs.url`	`Word.wid`	Plan Cost
Clustered	Clustered	Clustered	
Clustered	Clustered	Unclustered	
Clustered	Unclustered	Clustered	
Clustered	Unclustered	Unclustered	
Unclustered	Clustered	Clustered	
Unclustered	Clustered	Unclustered	
Unclustered	Unclustered	Clustered	
Unclustered	Unclustered	Unclustered	

Solution: There are $10^9/10^7 = 100$ `Webpage` tuples with author = 'John'. Thus, the cost of $\sigma_{\texttt{author='John'}}(\texttt{Webpage})$ is 1 (clustered) or 100 (unclustered)
For each of the 100 `Webpage` tuples, there are 10^3 matching `Occurs` tuples. Thus, the cost of $\bowtie_{\texttt{url=url}}$ per each `Webpage` tuple is 10 (clustered) or 10^3 (unclustered). Total cost of this join is 10^3 or 10^5.

For each of the 10^5 `Webpage-Occurs` tuples there is only one matching `Word` tuple. The lookup cost is 1, i.e. a total of 10^5.

`Webpage.author`	`Occurs.url`	`Word.wid`	Plan Cost
Clustered	Clustered	Clustered	$1 + 10^3 + 10^5 = 101001$
Clustered	Clustered	Unclustered	same
Clustered	Unclustered	Clustered	$1 + 10^5 + 10^5 = 200001$
Clustered	Unclustered	Unclustered	same
Unclustered	Clustered	Clustered	$100 + 10^3 + 10^5 = 101100$
Unclustered	Clustered	Unclustered	same
Unclustered	Unclustered	Clustered	$100 + 10^5 + 10^5 = 200100$
Unclustered	Unclustered	Unclustered	same

(b) Now consider the following plan:



The plan uses three indexes, `Webpage.author`, `Occurs.url`, and `Word.language`. Assuming all indexes are clustered and that the hash-join is a main memory join

operator, compute the cost of the plan.
Cost is:

Solution: $1 + 10^3 + 10^2 = 1101$.

140. *(from CSE 344, Autumn 2013, Midterm)*

Consider the following two relations:

```
Person(pid, name)
Order(oid, pid, product, quantity, date)
```

We create the following indexes:

```
create index Person_pid on Person(pid)
create index Person_name on Person(name)
create index Order_oid on Order(oid)
create index Order_pid on Order(pid)
create index Order_product on Order(product)
```

Consider the following two SQL queries:

```
-- Q1
select product, quantity, date
from Person, Order
where Person.name = 'Alice' and Person.pid = Order.pid

-- Q2
select name
from Person, Order
where Person.pid = Order.pid and Order.product = 'Tablet'
```

(a) Which indexes may be useful for query `Q1`?

	`Person_pid`	`Person_name`	`Order_oid`	`Order_pid`	`Order_product`
Answer Y/N:					

Solution:

	`Person_pid`	`Person_name`	`Order_oid`	`Order_pid`	`Order_product`
Answer:	N	Y	N	Y	N

(b) Which indexes may be useful for query `Q2`?

	`Person_pid`	`Person_name`	`Order_oid`	`Order_pid`	`Order_product`
Answer Y/N:					

Solution:

	`Person_pid`	`Person_name`	`Order_oid`	`Order_pid`	`Order_product`
Answer:	Y	N	N	N	Y

141. *(from CSE 544p, Winter 2014, Final; CSE 544p, Autumn 2015, Final)*

Consider the following database about word occurrences in Webpages:

```
Webpage(url, author, text)
Occurs(url, wid)
Word(wid, text, language)
```

where: `Occurs.url` and `Occurs.wid` are foreign keys to `Webpage` and `Word` respectively. This problem is about the execution of the following query, retrieving all Webpages written by 'John' that contain some French words (the projection on Webpages and the duplication elimination are omitted from both the query and the plan):

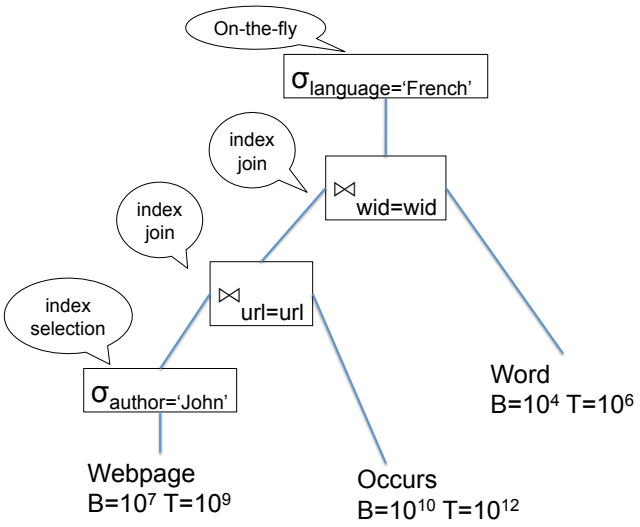
```
select *
from Webpage x, Occurs y, Word z
where x.url = y.url and y.wid = z.wid
  and x.author = 'John' and z.language = 'French'
```

Assume the following statistics

$$
\begin{aligned}
T(\texttt{Webpage}) = V(\texttt{Occurs}, \texttt{url}) &= 10^9 && \text{/* distinct URL's */}\\
V(\texttt{Webpage}, \texttt{author}) &= 10^7 && \text{/* about } 10^2 \text{ webpages/author */}\\
B(\texttt{Webpage}) &= 10^7 && \text{/* } 10^2 \text{ records/block */}\\
T(\texttt{Occurs}) &= 10^{12} && \text{/* about } 10^3 \text{ words/webpage */}\\
B(\texttt{Occurs}) &= 10^{10} && \text{/* } 10^2 \text{ records/block */}\\
T(\texttt{Word}) = V(\texttt{Occurs}, \texttt{wid}) &= 10^6 && \text{/* distinct words */}\\
V(\texttt{Word}, \texttt{language}) &= 100 && \text{/* about } 10^4 \text{ words in each language */}\\
B(\texttt{Word}) &= 10^4 && \text{/* } 10^2 \text{ records/block */}
\end{aligned}
$$

Assume that all indexes fit in main memory.

(a) Consider the following plan:



The plan uses three indexes, `Webpage.author`, `Occurs.url`, and `Word.wid`. Each of the first two indexes may be clustered, or unclustered; the last index is always clustered. In each case, compute the cost of the plan.

`Webpage.author`	`Occurs.url`	`Word.wid`	Plan Cost
Clustered	Clustered	Clustered	
Clustered	Unclustered	Clustered	
Unclustered	Clustered	Clustered	
Unclustered	Unclustered	Clustered	

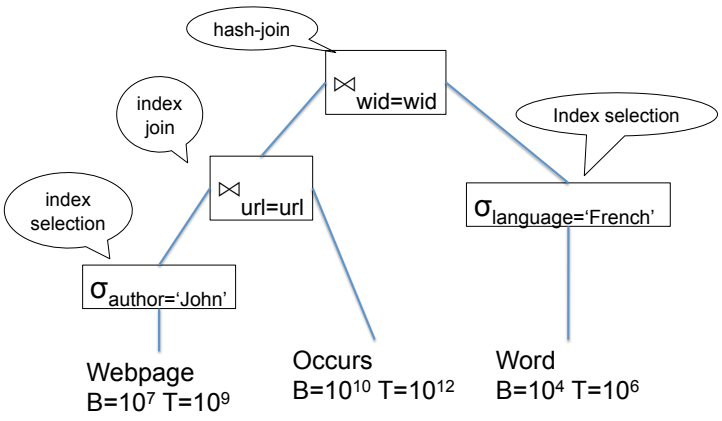
Solution: There are $10^9/10^7 = 100$ `Webpage` tuples with author = 'John'. Thus, the cost of $\sigma_{\texttt{author='John'}}(\texttt{Webpage})$ is 1 (clustered) or 100 (unclustered)

For each of the 100 `Webpage` tuples, there are 10^3 matching `Occurs` tuples. Thus, the cost of $\bowtie_{\texttt{url=url}}$ per each `Webpage` tuple is 10 (clustered) or 10^3 (unclustered). Total cost of this join is 10^3 or 10^5.

For each of the 10^5 `Webpage-Occurs` tuples there is only one matching `Word` tuple. The lookup cost is 1, i.e. a total of 10^5.

`Webpage.author`	`Occurs.url`	`Word.wid`	Plan Cost
Clustered	Clustered	Clustered	$1 + 10^3 + 10^5 = 101001$
Clustered	Unclustered	Clustered	$1 + 10^5 + 10^5 = 200001$
Unclustered	Clustered	Clustered	$100 + 10^3 + 10^5 = 101100$
Unclustered	Unclustered	Clustered	$100 + 10^5 + 10^5 = 200100$

(b) Now consider the following plan:



The plan uses three indexes, `Webpage.author`, `Occurs.url`, and `Word.language`. Assuming all indexes are clustered and that the hash-join is a main memory join operator, compute the cost of the plan.

Cost is:

Solution: $1 + 10^3 + 10^2 = 1101$.

12 Query Execution and Optimization (142–164)

Between a SQL query and its answer lies a physical plan, and this section is about choosing one. The problems ask for the I/O cost of nested-loop, sort-merge and hash joins, for the best order in which to join three or more relations, and for the plan that a cost-based optimizer would select, given a set of statistics on the database.

Cost here means blocks read and written, not tuples, and the size of main memory is what determines whether a join runs in one pass or two. Get the formulas for the one-pass and two-pass algorithms straight before starting, since most of the arithmetic in this section is an application of them.

A second group of questions is about plan enumeration rather than cost: which rewritings are legal, why pushing a selection below a join is always safe while pushing it below an aggregation is not, and how many plans a dynamic-programming optimizer must consider.

142. *(from CSE 444, Autumn 2000, Final)*

Consider the following schema:

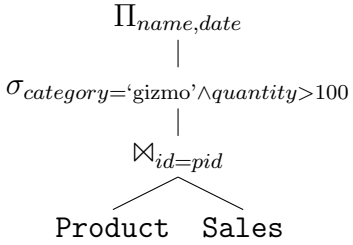
```
Product(id(4), name(16), manufacturer(20), category(10),
        color(10), webpage(40))
Sales(pid(4), quantity(4), shippingaddress(20), date(12),
      shippingmethod(10))
```

Each attribute has a fixed length, with the size (in bytes) indicated by the number in parentheses. The number of tuples in each relation is $T(\texttt{Product}) = 1{,}000{,}000$ and $T(\texttt{Sales}) = 2{,}000{,}000$. The size of one disk block is 1000 bytes, and there are 101 buffers available in main memory.

(a) Compute the number of blocks taken by each table: $B(\texttt{Product}) =?$, $B(\texttt{Sales}) =?$

Solution: One `Product` record takes 100 bytes so 10 fit in a block, hence $B(\texttt{Product}) = T(\texttt{Product})/10 = 100{,}000$. One `Sales` record takes 50 bytes so 20 fit in a block, hence $B(\texttt{Sales}) = T(\texttt{Sales})/20 = 100{,}000$.

(b) Consider the logical plan below.



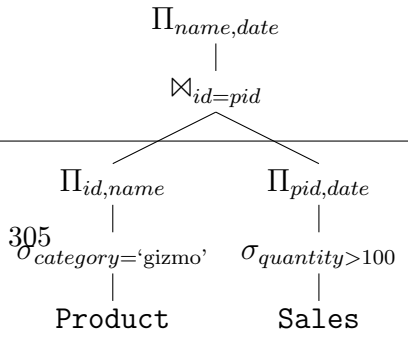
Assume the following physical plan: the join is implemented as a block-nested-loop join, its result is pipelined into the selection operator, and from there pipelined into the projection operator. Compute the total cost of this physical plan.

Solution: The cost is that of the join:

$$B(\texttt{Product}) + \frac{B(\texttt{Product})B(\texttt{Sales})}{M-1} = 100{,}000 + \frac{100{,}000 \times 100{,}000}{100} \approx 100M$$

(c) Derive a new logical plan by pushing selections and projections down as far as possible (you have to draw a plan).

Solution:



(d) Consider a physical plan for your new logical plan in which all selections and projections are pipelined and the join is a block-nested-loop join. Further assume that 1% of all `Product`s are in category "Gizmo" and that 20% of all `Sales` have a quantity over 100. Compute the cost of your plan.

Solution: Let $\texttt{Product}'$ and $\texttt{Sales}'$ be the left and right operands of the join. Then $T(\texttt{Product}') = 10{,}000$ with record size 20, so 50 records per block and $B(\texttt{Product}') = 200$; $T(\texttt{Sales}') = 400{,}000$ with record size 16, so $B(\texttt{Sales}') = 640$ approximately.

It is more advantageous to use $\texttt{Product}'$ as the outer relation: the block-nested loop reads `Product`, performs the projection and selection on the fly, and stores 100 blocks of $\texttt{Product}'$ records in main memory. For each such set it scans `Sales` fully, performs the selection and projection, then the join. Total cost:

$$B(\texttt{Product}) + \frac{B(\texttt{Product}')}{100} \times B(\texttt{Sales}) = 100{,}000 + \frac{200}{100} \times 100{,}000 = 300{,}000$$

143. *(from CSE 544, 2001, Final)*

Consider two tables $R(a, b)$ and $S(c, d)$. The number of tuples in S is $T(S) = 100M$. The amount of available main memory is $M = 100k$ (measured in blocks). We have an unclustered index on c in S, and we can fit 10 records in a block, both for R and for S (hence $B(S) = 10M$).

Consider the following alternative ways to compute $R \bowtie_{b=c} S$:

(a) Main memory hash join: R stays in memory.

(b) Main memory hash join: S stays in memory.

(c) Partition-based hash join: R's buckets stay in memory in step 2.

(d) Partition-based hash join: S's buckets stay in memory in step 2.

(e) Index join, using the index on c in S.

(f) Block nested loop join.

(a) For each of the four cases in the table below, indicate which join method you could choose. Recall that $V(S, c)$ indicates the number of distinct values of the attribute c in S. You have to answer with (a), (b), ..., or (f) in each of the four entries below.

	$T(R) = 1k$	$T(R) = 100M$, $V(R, b) = 10M$
$V(S, c) = 10$		
$V(S, c) = 50M$		

Solution:

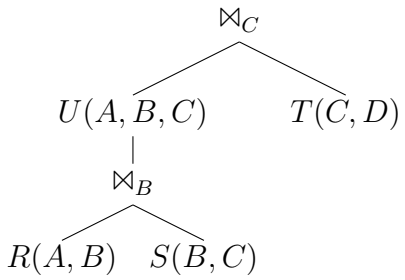
	$T(R) = 1k$	$T(R) = 100M$, $V(R, b) = 10M$
$V(S, c) = 10$	(a)	(c)
$V(S, c) = 50M$	(e)	(d)

144. *(from CSE 444, Autumn 2001, Final)*

Consider three relations $R(A, B)$, $S(B, C)$, $T(C, D)$ and assume the following statistics:

$$B(R) = 10000, \quad B(S) = 20000, \quad B(T) = 30000, \quad B(U) = 40000$$

where $U = R \bowtie S$. Consider the following plan:



We consider evaluating the two joins in a pipeline, as follows:

Step 1 Partition R, then S into buckets on B, and store the B-buckets on disk.

Step 2 Join each pair of B-buckets from R and S and partition the join tuples immediately on the C attribute (pipeline); store the C-buckets of U on disk.

Step 3 Partition T into buckets on C, and store its C-buckets on disk.

Step 4 Join each pair of C-buckets from U and T.

(a) Compute the minimum amount of memory M needed for this plan.

Solution: In step 1 we generate the maximum number of B-buckets, i.e. M (it never hurts to have more).

In step 2 we need to store in memory one bucket from R (we favour R over S since R is smaller) and use the remaining memory to partition U into buckets on C. The expected size of a bucket from R is $B(R)/M$, so the maximum number of C-buckets we can create in steps 2 and 3 is $M - B(R)/M$.

In step 4 we must hold a bucket from T in main memory (we favour T over U since T is smaller). The expected size of a bucket from T is $B(T)/(M -$

$B(R)/M$), and this has to be at most M:

$$\begin{aligned} B(T)/(M - B(R)/M) &\leq M \\ B(T) &\leq M^2 - B(R) \\ B(R) + B(T) &\leq M^2 \\ 10000 + 30000 &\leq M^2 \\ 200 &\leq M \end{aligned}$$

We need at least $M = 200$ for this plan to work.

(b) The number of B-buckets.

Solution: 200

(c) The average size of each B-bucket, for R and for S.

Solution: R: 50 and S: 100

(d) The number of C-buckets.

Solution: 150

(e) The average size of each C-bucket, for U and for T.

Solution: U: 266 and T: 200

145. *(from CSE 444, Autumn 2001, Final; CSE 444, Autumn 2002, Final)*

(a) Compute the optimal plan for $R \bowtie S \bowtie T \bowtie U$ using the dynamic programming algorithm, assuming the following:

$$B(R) = 500, \quad B(S) = 200, \quad B(T) = 600, \quad B(U) = 400$$

The size of a join is estimated as $B(A \bowtie B) = 0.01 \cdot B(A) \cdot B(B)$. The cost of a join is estimated to be the cost of the subplans plus the size of the intermediate result(s).

Solution:

Subquery	Size	Cost	Optimal plan
RS	1k	0	
RT	3k	0	
RU	2k	0	
ST	1.2k	0	
SU	0.8k	0	
TU	2.4k	0	
RST	6k	1k	$(RS)T$
RSU	4k	0.8k	$(SU)R$
RTU	12k	2k	$(RU)T$
STU	4.8k	0.8k	$(SU)T$
$RSTU$	24k	3.2k	$(RU)(ST)$

146. *(from CSE 444, Autumn 2002, Final)*

Consider two tables $R(A, B)$ and $S(B, C)$, where all attributes are integers. Let `Q1`, `Q2` be the following SQL queries:

```
Q1: select R.A, S.C
    from   R, S
    where  R.B = S.B
```

```
Q2: select R.A, sum(S.C)
    from   R, S
    where  R.B = S.B
    group by R.A
```

Assume $M = 102$, $B(R) = B(S) = 10000$, $V(R, B) = V(S, B) = 100$.

(a) Give two optimal physical query execution plans for `Q1` and `Q2`, and indicate their cost. Your plans should use the following physical operators:

- Nested loop join or block nested loop join
- Main memory hash join
- Partitioned hash join
- Main memory hash group-by
- Partitioned hash group-by

Solution: Q1: partitioned hash join. Cost $= 3B(R) + 3B(S) = 60000$.
Q2:

$t_1 = \gamma_{B,sum}(S)$	main memory hash; cost $= B(S)$
$t_2 = R \bowtie_B t_1$	main memory hash; cost $= B(R)$
$t_3 = \Pi_{A,sum}(t_2)$	pipelined from the previous operation; must materialise t_3; cost $= B(t_3) = B(R)$
answer $= \gamma_{A,sum}(t_3)$	main memory hash; cost $= B(t_3) = B(R)$

Total cost: $B(S) + 3B(R) = 40000$.

Notice that B is a key in $t_1(B, sum)$, hence $t_2 = R \bowtie_B t_1$ has as many tuples as R, and so has t_3. Since t_3's schema is essentially the same as that of R, we have $B(R) = B(t_2)$.

147. *(from CSE 444, Autumn 2003, Final)*

Consider two tables $R(A, B, C)$ and $S(D, E, F)$ and the following SQL query:

```
Select R.A, sum(S.F)
From   R, S
Where  R.B = '1234' and R.C = S.D and S.E = '5678'
Group by R.A
```

We have clustered indexes on $R.A$ and $S.D$, and unclustered indexes on $R.B$ and $S.E$. We assume that both tables R and S are very large and do not fit in main memory. Consider the following three logical plans:

$$
\begin{aligned}
P_1 &= \gamma_{A,sum(F)}\big(\sigma_{B=1234}(R) \bowtie_{C=D} \sigma_{E=5678}(S)\big) \\
P_2 &= \gamma_{A,sum(F)}\big(\sigma_{B=1234}(R \bowtie_{C=D} \sigma_{E=5678}(S))\big) \\
P_3 &= \gamma_{A,sum(F)}\big(\sigma_{E=5678}(\sigma_{B=1234}(R \bowtie_{C=D} S))\big)
\end{aligned}
$$

Notice that the order of the join arguments matters. We only consider plans in which R occurs on the left and S on the right.

(a) Indicate for which logical plans you can use an index join.

Solution: P_1 only. The others perform a selection on S and the index can no longer be used.

(b) Indicate for which logical plans you can use merge join without having to create the initial runs for the right argument.

Solution: P_1, P_2, P_3. In all cases the right operand returns tuples sorted by the primary index, i.e. by $S.D$.

(c) If we decide to implement the join operator as a partitioned hash join, indicate which of the three logical plans would always be more efficient than the other two.

Solution: P_1. Pushing selections down is always better here: we lose the indexes, but we do not need them for a hash join.

(d) Consider the logical plan P_1, and two associated physical plans, P_1' and P_1''. P_1' uses an index join, and P_1'' uses a hash join (a main memory hash join if possible, or a partitioned hash join otherwise). In which of the cases below do we know for sure that P_1' is definitely more efficient than P_1''? Answer YES only if you can guarantee it based only on the fact given, regardless of the other statistics on the data.

i. If all tuples in R satisfy $R.B =' 1234'$.

Solution: NO

ii. If there exists only one tuple in R satisfying $R.B =' 1234'$.

Solution: YES

iii. If all tuples in S satisfy $S.E =' 5678'$.

Solution: NO

iv. If there exists only one tuple in S satisfying $S.E =' 5678'$.

Solution: NO

(e) Consider the operator γ in any of the three plans (the answer is the same for all three). We consider a hash-based implementation: a main memory hash-based algorithm if possible, or a partition-based hash algorithm otherwise. We assume a relatively small amount of main memory, while the tables R and S are large. In which of the cases below are we certain that we can execute γ in main memory?

i. If all tuples in R satisfy $R.B =' 1234'$.

Solution: NO

ii. If there exists only one tuple in R satisfying $R.B =' 1234'$.

Solution: YES

iii. If all tuples in S satisfy $S.E =' 5678'$.

Solution: NO

iv. If there exists only one tuple in S satisfying $S.E =' 5678'$.

Solution: NO

148. *(from CSE 544, 2004, Final)*

Consider two tables $R(A, B, C, V)$ and $S(D, E, F)$ and the following SQL query:

```
Select S.F, sum(R.V)
From   R, S
Where  R.B = '1234' and R.C = S.D and S.E = '5678'
Group by S.F
```

We have clustered indexes on $R.A$ and $S.D$, and unclustered indexes on $R.B$ and $S.E$. We assume that both tables R and S are very large and do not fit in main memory. Consider the following three logical plans:

$$\begin{aligned} P_1 &= \gamma_{F,sum(V)}\big(\sigma_{B=1234}(R) \bowtie_{C=D} \sigma_{E=5678}(S)\big) \\ P_2 &= \gamma_{F,sum(V)}\big(\sigma_{E=5678}(\sigma_{B=1234}(R) \bowtie_{C=D} S)\big) \\ P_3 &= \gamma_{F,sum(V)}\big(\sigma_{E=5678}(\sigma_{B=1234}(R \bowtie_{C=D} S))\big) \end{aligned}$$

We only consider plans in which R occurs on the left and S on the right.

(a) Indicate for which logical plans you can use an index join. Recall that an index join uses the index on the right operand.

Solution: P_2, P_3. One cannot use an index join for P_1 because the index is lost after the selection.

(b) Assume that all selections applied directly to base tables are implemented as index lookups. Indicate for which logical plans you can use merge join without having to create the initial runs for the right argument.

Solution: P_3 only: use an index scan on $R.C$ and an index scan on $S.D$.

(c) If we decide to implement the join operator as a partitioned hash join, indicate in which of the three logical plans the join operator is guaranteed to perform at most as many page I/Os as any of the two other plans.

Solution: P_1, because it joins smaller tables.

(d) Consider the logical plan P_2. When is an index join guaranteed to perform fewer page I/O's than a hash join (assuming that a main memory hash join is possible)?

i. All tuples in R satisfy $R.B =' 1234'$.

Solution: No: the index join is terrible here.

ii. There exists only one tuple in R satisfying $R.B =' 1234'$.

Solution: Yes: the index join makes a single index lookup.

(e) Assume $B(R) = B(S) = 10000$, $M = 102$, all tuples in R satisfy $R.B =' 1234'$ and all tuples in S satisfy $S.E =' 5678'$ (hence $V(R, B) = V(S, E) = 1$), and $V(R, C) = V(S, F) = 100$. Notice that none of the logical plans can be implemented in main memory only, because the join is over two big tables. Find a new logical plan which can be executed entirely in main memory, except for reading from the tables R and S.

Solution:

$$\gamma_{F,sum(V)}\big(\gamma_{C,sum(V)}(\sigma(R)) \bowtie \sigma(S)\big)$$

The inner γ has only 100 buckets, hence can be done in main memory. The join has a left operand that fits in main memory, and can be done as a nested loop join, with the results piped into the outer γ operator. This can be implemented as a nested loop too, by reusing the buckets of the first γ operator.

149. *(from CSE 444, Winter 2005, Final)*

Consider two tables $R(A, B)$ and $S(C, D)$, and the following statistics:

$$\begin{array}{lll} B(R) = 5 & B(S) = 100 & M = 1000 \\ T(R) = 20 & T(S) = 400 & \\ & V(S, C) = 50 & \end{array}$$

There is a clustered index on $S.C$. Consider the logical operator $R \bowtie_{B=C} S$, and the following two physical operators:

$$\begin{aligned} P_1 &= \text{main memory hash join} \\ P_2 &= \text{index join} \end{aligned}$$

(a) Compute for each of them the number of disk I/O's required to execute the operator.

Solution: The cost of the main-memory hash-join is:

$$B(R) + B(S) = 5 + 100 = 105$$

The cost of a clustered index join is

$$B(R) + \frac{T(R)B(S)}{V(S,C)} = 5 + \frac{20 * 100}{50} = 5 + 40 = 45$$

150. *(from CSE 444, Winter 2006, Final)*

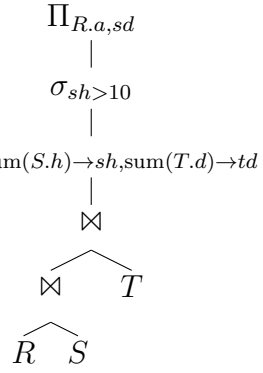
Consider three tables $R(a, b, f)$, $S(b, c, h)$, $T(c, d, g)$.

(a) Consider the following SQL query:

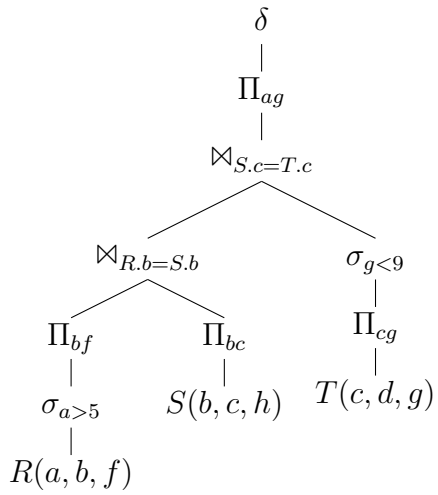
```
SELECT R.a, sum(T.d)
FROM   R, S, T
WHERE  R.b = S.b AND S.c = T.c AND R.f = 5 AND T.g = 9
GROUP BY R.a
HAVING sum(S.h) > 10
```

Draw a logical plan for the query. You may choose any plan as long as it is correct (i.e. no need to worry about efficiency).

Solution:



(b) Write a SQL query that is equivalent to the logical plan below:



Solution:

```
select distinct R.a, T.g
from R, S, T
where R.b=S.b and S.c=T.c
  and R.a>5 and T.g<9
```

151. *(from CSE 444, Autumn 2005, Final; CSE 444, Autumn 2006, Final; CSE 544, 2006, Final)*

Consider two tables $R(A, B)$ and $S(C, D)$ with the following statistics:

$$\begin{array}{lll} B(R) = 5 & B(S) = 100 & M = 1000 \\ T(R) = 200 & T(S) = 400 & \\ V(R, A) = 10 & V(S, C) = 50 & \end{array}$$

There is a clustered index on $S.C$ and an unclustered index on $R.A$. Consider the logical plan:

$$P = \sigma_{A=77}(R) \bowtie_{B=C} S$$

There are two logical operators, $S = \sigma_{A=77}$ and $J = \bowtie_{B=C}$, and for each we consider two physical operators:

$$\begin{array}{rcl} s1 & = & \text{one pass table scan} \\ s2 & = & \text{index-based selection} \\ j1 & = & \text{main memory hash join} \\ j2 & = & \text{index-based join} \end{array}$$

Both $s1$ and $s2$ are pipelined, i.e. the result of the select operator is not materialized.

(a) For each of the resulting four physical plans compute its cost in terms of number of disk I/O's, expressed as a function of the statistics above. Your answer should consist of four expressions, e.g. $\texttt{COST}(s1j1) = B(R)B(S)/M + V(R, A)$ (not the real answer).

i. $\texttt{COST}(s1j1) =$

Solution: $B(R) + B(S) = 5 + 100 = 105$

ii. $\texttt{COST}(s2j1) =$

Solution: $T(R)/V(R, A) + B(S) = 20 + 100 = 120$

iii. $\texttt{COST}(s1j2) =$

Solution: $B(R) + T(R)/V(R, A) = 5 + 20 = 25$

iv. $\texttt{COST}(s2j2) =$

Solution: $T(R)/V(R, A) + T(R)/V(R, A) = 20 + 20 = 40$

(b) Indicate the cheapest plan of the four, together with its cost expressed as a number.

Solution: $s1j2$, with cost 25.

152. *(from CSE 444, Autumn 2004, Final; CSE 444, Autumn 2005, Final; CSE 444, Autumn 2006, Final; CSE 544, 2006, Final)*

(a) Consider two tables $R(A, B, C)$ and $S(D, E, F)$, and the query plan P below. Indicate which of the four query plans P_1, P_2, P_3, P_4 are equivalent to P. The symbol $\rtimes$ in P_4 represents the right outer join, i.e. $R \rtimes S = S \ltimes R$.

$$\begin{aligned}
P &= \sigma_{A>9}(\gamma_{A,sum(F)}(R \bowtie_{C=D} S)) \\
P_1 &= \gamma_{A,sum(F)}(\sigma_{A>9}(R) \bowtie_{C=D} \gamma_{D,sum(F)}S) \\
P_2 &= \gamma_{A,sum(F)}(\sigma_{A>9}(R) \bowtie_{C=D} \gamma_{D,E,sum(F)}S) \\
P_3 &= \gamma_{A,sum(F)}(\sigma_{A>9}(R) \bowtie_{C=D} \gamma_{D,E,sum(F)}(\sigma_{A>9}(R) \bowtie_{C=D} S)) \\
P_4 &= \gamma_{A,sum(F)}(\sigma_{A>9}(R) \bowtie_{C=D} \gamma_{D,E,sum(F)}(\sigma_{A>9}(R) \rtimes_{C=D} S))
\end{aligned}$$

Solution: P_1 equivalent, P_2 equivalent, P_3 not equivalent, P_4 equivalent.

(b) Consider the following query, where $\bowtie$ denotes the natural join:

$$R(A, B) \bowtie S(B, C) \bowtie T(C, D) \bowtie U(D, E)$$

Here we only consider left linear plans, and do not distinguish between different join orders, i.e. $R(A,B) \bowtie S(B,C)$ is the same as $S(B,C) \bowtie R(A,B)$.

i. How many different left linear plans exist for this query?

Solution: 12

ii. Show two different left linear plans without cartesian products.

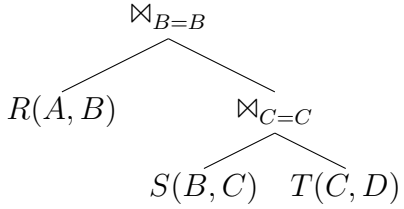
Solution: $(((R \bowtie S) \bowtie T) \bowtie U)$ and $(((U \bowtie T) \bowtie S) \bowtie R)$

iii. How many different plans without cartesian product exist for this query?

Solution: 4

153. *(from CSE 444, Autumn 2008, Final)*

Consider the following logical plan:



Consider the following parameters:

```
M = 100
B(R) = 8000000
B(S) = 7000000
B(T) = 5000
B(S ⋈ T) = 5000
```

(a) You will use partitioned hash join to process this query. You are allowed to interleave the two steps (partition and hash) for the two joins, as necessary. Show an optimal order of execution. Indicate the following:

- order of the two partition and two join steps
- the number of buckets used for each partition
- the size of the buckets (assume perfectly uniform distribution)
- which operations are pipelined
- indicate the total cost of this query plan

Solution:

$$3(B(R) + B(S) + B(T)) + 2B(S \bowtie T)$$

Pipeline the join phase of $S \bowtie T$ with the partition for the next join.

154. *(from CSE 544p, Spring 2009, Final)*

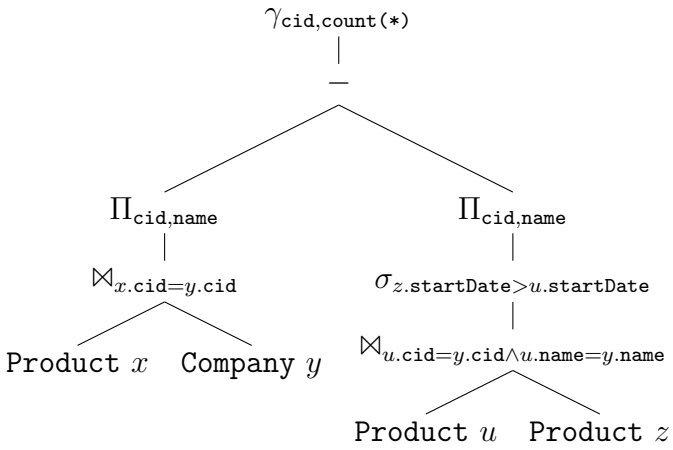
(a) Consider the following relational schema:

$$\texttt{Product}(\underline{\texttt{pid}}, \texttt{name}, \texttt{cid}, \texttt{startDate})$$
$$\texttt{Company}(\underline{\textit{cid}}, \texttt{address})$$

Show a relational algebra plan that is equivalent to the following SQL query:

```
select y.cid, count(*)
from Product x, Company y
where x.manufacturer = y.cid
  and y.address='Seattle'
  and not exists (select *
                  from Product z
                   where z.manufacturer = y.cid
                     and z.name = x.name
                     and z.startDate > x.startDate)
group by y.cid
```

Solution:



(b) Consider two tables $R(A, B)$ and $S(C, D)$ with the following statistics:

$$\begin{aligned} B(R) &= 5 \\ T(R) &= 200 \\ V(R,A) &= 10 \\ B(S) &= 100 \\ T(S) &= 400 \\ V(S,C) &= 50 \\ M &= 1000 \end{aligned}$$

There is an unclustered index on $R.A$ and a clustered index on $S.C$. Consider the logical plan:

$$P = \sigma_{A=77}(R) \bowtie_{B=C} S$$

There are two logical operators, $s = \sigma_{A=77}$ and $j = \bowtie_{B=C}$, and for each we consider two physical operators:

$$\begin{aligned} s_1 &= \text{sequential table scan} \\ s_2 &= \text{index-based selection using } R.A \\ j_1 &= \text{main memory hash join} \\ j_2 &= \text{index-based join using the index } S.C \end{aligned}$$

Both s_1 and s_2 are pipelined, i.e. the result of the select operator is not materialized. For each of the resulting four physical plans compute its cost in terms number of disc I/0s, expressed as a function of the statistics above, then indicate its numerical value. Your answer should consists of four expressions of the form $\texttt{COST}(s_1 j_1) = B(R)B(S)/M + V(R,A) = 544$ (not the real answer). You may ignore the cost of accessing an index, i.e. assume that all intermediate nodes of a $B+$ tree are in the buffer pool.

1. $\texttt{COST}(s_1 j_1) =$

Solution:

$$B(R) + B(S) = 105$$

2. $\texttt{COST}(s_2 j_1) =$

Solution:

$$T(R)/V(R,A) + B(S) = 20 + 100 = 120$$

3. $\texttt{COST}(s_1 j_2) =$

   Solution:

   $$B(R) + T(R)/V(R,A) * B(S)/V(S,C) = 5 + 200/10 * 100/50 = 45$$

4. $\texttt{COST}(s_2 j_2) =$

   Solution:

   $$T(R)/V(R,A) + T(R)/V(R,A) * B(S)/V(S,C) = 20 + 200/10 * 100/50 = 60$$

(c) Consider two tables $R(A,B)$ and $S(C,D)$, and the query plan P below:

$$P \quad = \quad \sigma_{A>9}(\gamma_{A,sum(D)} \text{ as } {}_D(R \bowtie_{B=C} S))$$

Indicate which of the following three query plans P_1, P_2, P_3 are equivalent to P. The symbol $\ltimes$ represents semijoin, $R \ltimes_{\texttt{condition}} S = \Pi_{\texttt{Attributes}(R)}(R \bowtie_{\texttt{condition}} S)$.

$$P_1 \quad = \quad \gamma_{A,sum(D)} \text{ as } {}_D(\sigma_{A>9}(R) \bowtie_{B=C} \gamma_{C,sum(D)} \text{ as } {}_D(S))$$

$$P_2 \quad = \quad \gamma_{A,sum(D)} \text{ as } {}_D(\gamma_{A,sum(B)} \text{ as } {}_B(\sigma_{A>9}(R)) \bowtie_{B=C} (S))$$

$$P_3 \quad = \quad \gamma_{A,sum(D)} \text{ as } {}_D(\sigma_{A>9}(R) \bowtie_{B=C} \gamma_{C,sum(D)} \text{ as } {}_D(S \ltimes_{B=C} \sigma_{A>9}(R)))$$

Solution: $P \equiv P_1 \equiv P_3 \not\equiv P_2$.

(d) Consider the following query, where $\bowtie$ denotes the natural join:

$$R(A,B) \bowtie S(B,C) \bowtie T(C,D) \bowtie U(D,E)$$

Here we only consider left linear plans, and do not distinguish between different join orders, i.e. $R(A,B) \bowtie S(B,C)$ is the same as $S(B,C) \bowtie R(A,B)$.

1. Show two different left linear plans without cartesian products.

Solution:

$$((R \bowtie S) \bowtie T) \bowtie U$$
$$((S \bowtie T) \bowtie R) \bowtie U$$
$$((S \bowtie T) \bowtie U) \bowtie R$$
$$((T \bowtie U) \bowtie S) \bowtie R$$

2. How many different plans without cartesian product exists for this query ?

Solution: Four (see above).

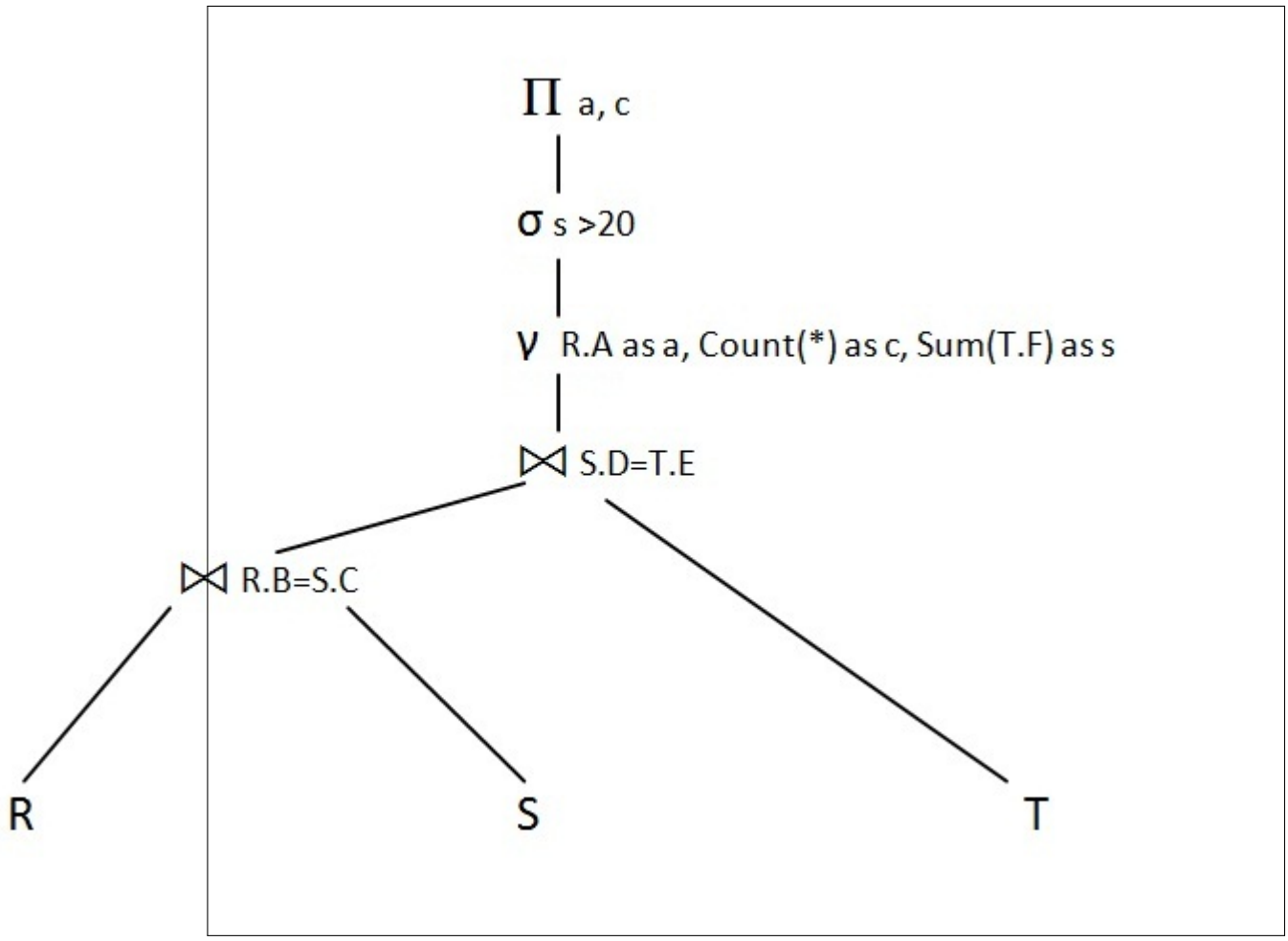
155. *(from CSE 444, Spring 2010, Final)*

For the following questions, we consider the schema $R(A, B)$, $S(C, D)$, $T(E, F)$, $U(G, H)$.

(a) Write a logical plan for the following query:

```
select R.A, sum(T.F)
from R, S, T
where R.B = S.C and S.D = T.E
group by R.A
having count(*) > 20
```

Solution:



(b) Consider the following query:

```
select *
from R, S, T, U
where R.B = S.C and S.D = T.E and S.D = U.G
```

- Write **all** left-deep, cartesian-free join trees for this query. Assume that the join operator is not commutative: that is, you should report both $R \bowtie S$ and $S \bowtie R$ as distinct plans. Note: you need to turn in several plans.

Solution:

$$R \bowtie S \bowtie T \bowtie U$$
$$R \bowtie S \bowtie U \bowtie T$$

$$S \bowtie R \bowtie T \bowtie U$$
$$S \bowtie R \bowtie U \bowtie T$$
$$S \bowtie T \bowtie R \bowtie U$$
$$S \bowtie T \bowtie U \bowtie R$$
$$S \bowtie U \bowtie R \bowtie T$$
$$S \bowtie U \bowtie T \bowtie R$$

$$T \bowtie S \bowtie R \bowtie U$$
$$T \bowtie S \bowtie U \bowtie R$$
$$T \bowtie U \bowtie S \bowtie R$$

$$U \bowtie S \bowtie R \bowtie T$$
$$U \bowtie S \bowtie T \bowtie R$$
$$U \bowtie T \bowtie S \bowtie R$$

- Write a full semijoin reducer for this query.

Solution:

$$\begin{aligned} S' &= S \ltimes T \\ S'' &= S' \ltimes U \\ S''' &= S'' \ltimes R \\ R' &= R \ltimes S''' \\ T' &= T \ltimes S''' \\ U' &= U \ltimes S''' \end{aligned}$$

$$Q = S''' \bowtie T' \bowtie U' \bowtie R'$$

156. *(from CSE 544p, Autumn 2010, Final)*

For the following questions, we consider the schema:

$R(A, B)$, $S(B, C)$, $T(C, D)$, $U(D, E)$, $V(C, F)$.

(a) Write a logical plan for the following query:

```
select R.A, sum(T.D)
from R, S, T
where R.B = S.B and S.C = T.C
group by R.A
having count(*) > 20
```

Solution:

```
T1 = R join S
T2 = T1 join T
T3 = gamma[A, sum(D) as S, count(*) as C](T2)
Answer = filter[C > 20](T3)
```

Notice that `count(*)` can only be used in the context of a `group by`. The following are incorrect:

```
T3 = filter T2 by count(*) > 20
```

Also `having count`$(*) > 20$ is incorrect.

(b) Consider the following query:

```
select *
from R, S, T, U
where R.B = S.B and S.C = T.C and T.D = U.D
```

Count the number of distinct left-deep, cartesian-free join trees for this query. Assume that the join operator is not commutative: that is, you should report both $R \bowtie S$ and $S \bowtie R$ as distinct plans. You need to turn in a number.

The number of plans is:

Solution: 8

(c) Generalize the question above: assuming there are n relations: $R_1(A_0, A_1), R_2(A_1, A_2), \ldots, R_n(A_{n-1}, A_n)$, count the number of distinct left-deep, cartesian-free join trees for the natural join query $R_1 \bowtie R_1 \bowtie \ldots \bowtie R_n$. You need to turn in a mathematical formula in n, for example $n^2 + n$ (not the real answer).

The number of plans is:

Solution: 2^{n-1}

(d) Write a full semi-join reducer for the query below

```
Q= select *
   from R, S, T, V
   where R.B = S.B and S.C = T.C and S.C = V.C
```

Your answer should contain a sequence of semi-join reductions, followed by the query expression, e.g.:

$$\begin{aligned} R1 =& R \ltimes V \\ R2 =& R1 \ltimes T \\ & \dots \\ Q =& R7 \bowtie S3 \bowtie T5 \bowtie V6 \end{aligned}$$

Solution:

$$Q(A,B,C,D,E,F) = R(A,B) \bowtie S(B,C) \bowtie T(C,D) \bowtie V(C,F)$$

$$\begin{aligned} S1(B,C) =& S(B,C) \ltimes R(A,B) \\ S2(B,C) =& S1(B,C) \ltimes T(C,D) \\ S3(B,C) =& S2(B,C) \ltimes V(C,F) \\ R1(A,B) =& R(A,B) \ltimes S3(B,C) \\ T1(C,D) =& T(C,D) \ltimes S3(B,C) \\ V1(C,F) =& V(C,F) \ltimes S3(B,C) \\ Q(A,B,C,D,E,F) =& R1(A,B) \bowtie S3(B,C) \bowtie T1(C,D) \bowtie V1(C,F) \end{aligned}$$

157. *(from CSE 544p, Winter 2014, Final)*

For the following questions, we consider the schema:

$R(A,B)$, $S(B,C)$, $T(C,D)$,$U(D,E)$, $V(C,F)$.

(a) Write a logical plan for the following query.

```
select R.A, sum(T.D)
from R, S, T
where R.B = S.B and S.C = T.C
group by R.A
having count(*) > 20
```

To input your plan, use a Pig-latin style notation, with ASCII characters instead of Greek letters. Your final answer should be a relation name called Answer. For example, the plan $\gamma_{A,sum(D)}(\sigma_{A>10}(R) \bowtie_{B=C} S)$ would be written as:

```
T1     =   filter[A>10](R)
T2     =   join T1 by B, S by C
ANSWER =   gamma[A,sum(D)](T2)
```

Solution:

```
T1 = R join S
T2 = T1 join T
T3 = gamma[A, sum(D) as S, count(*) as C](T2)
Answer = filter[C > 20](T3)
```

(b) Consider the following query:

```
select *
from R, S, T, U
where R.B = S.B and S.C = T.C and T.D = U.D
```

Count the number of distinct left-deep, cartesian-free join trees for this query. Assume that the join operator is not commutative: that is, you should report both $R \bowtie S$ and $S \bowtie R$ as distinct plans. You need to turn in a number.

The number of plans is:

Solution: 8

Solution: See the solution for the next question.

(c) Generalize the question above: assuming there are n relations: $R_1(A_0, A_1), R_2(A_1, A_2), \ldots, R_n(A_{n-1}, A_n)$, count the number of distinct left-deep, cartesian-free join trees for the natural join query $R_1 \bowtie R_1 \bowtie \ldots \bowtie R_n$. You need to turn in a mathematical formula in n, for example $n^2 + n$ (not the real answer).

The number of plans is:

Solution: 2^{n-1}

Solution: Consider the partial plan P_k that joins the first k relations. These relations must form a continuous sequence:

$$P_k = R_i \bowtie R_{i+1} \bowtie \ldots \bowtie R_{i+k-1}$$

At step $k+1$ we have two choices: we can either extended to the left, or we can extend to the right:

$$P_{k+1} = P_k \bowtie R_{i-1}$$
$$P_{k+1} = P_k \bowtie R_{i+k}$$

Thus, each complete plan can be represented by a sequence of $n-1$ 0's and 1's: a 0 for expanding to the right, and a 1 for expanding to the left. The

first relation, R_i, is uniquely defined by the number of 1's in the sequence. For example, if $n = 4$, and we call the relations R_1, R_2, R_3, R_4 as in the previous problem R, S, T, U then we have 8 plans:

000	$((R \bowtie_B S) \bowtie_C T) \bowtie_D U$
001	$((S \bowtie_C T) \bowtie_D U) \bowtie_B R$
010	$((S \bowtie_C T) \bowtie_B R) \bowtie_D U$
011	$((T \bowtie_D U) \bowtie_C S) \bowtie_B R$
100	$((S \bowtie_B R) \bowtie_C T) \bowtie_D U$
101	$((T \bowtie_C S) \bowtie_D U) \bowtie_B R$
110	$((T \bowtie_D U) \bowtie_C S) \bowtie_B R$
111	$((U \bowtie_D T) \bowtie_C S) \bowtie_B R$

158. *(from CSE 544p, Autumn 2015, Final)*

For the following questions, we consider the schema:

$R(A, B)$, $S(B, C)$, $T(C, D)$,$U(D, E)$, $V(C, F)$.

(a) Write a logical plan for the following query.

```
select R.A, sum(T.D)
from R, S, T
where R.B = S.B and S.C = T.C
group by R.A
having count(*) > 20
```

To input your plan, use a Pig-latin style notation, with ASCII characters instead of Greek letters. Your final answer should be a relation name called Answer. For example, the plan $\gamma_{A,sum(D)}(\sigma_{A>10}(R) \bowtie_{B=C} S)$ would be written as:

```
T1     =   filter[A>10](R)
T2     =   join T1 by B, S by C
ANSWER =   gamma[A,sum(D)](T2)
```

Solution:

```
T1 = R join S
T2 = T1 join T
T3 = gamma[A, sum(D) as S, count(*) as C](T2)
Answer = filter[C > 20](T3)
```

(b) Consider the following conjunctive query:

```
Q(x,y,z,u) = R(x,y,z),S(x,y,u),T(x,z,u),K(y,z,u)
```

Suppose $|R| = |S| = |T| = |K| = N$. What is the largest possible size of the query's output, $|Q|$? You will use here the AGM bound.

The size of Q's output is $\leq$:

Solution: $N^{4/3}$

Solution: An optimal fractional vertex cover is $w_R = w_S = w_T = w_K = 1/3$, because every variable is covered by 1. For example, x is covered because $w_R + w_S + w_T = 1$. Thus:

$$|Q| \leq N^{w_R} \cdot N^{w_S} \cdot N^{w_T} \cdot N^{w_K} = N^{4/3}$$

(c) For each of the queries below, indicate whether they are acyclic or cyclic.

i. $Q(x, y, z, u) = R(x, y), S(y, z), T(z, u), K(u, x)$

Solution: n

ii. $Q(x, y, z, u, v, w) = R(x, y), S(y, z), T(z, u), K(y, w), L(u, v)$

Solution: y

iii. $Q(x, y, z, u, v, w) = R(x, y, z), S(y, u), T(y, v), K(y, z, w)$

Solution: y

iv. $Q(x, y, z, u, v) = R(x, y, z), S(y, u), T(y, v), K(y, z, v)$

Solution: n

v. $Q(x, y, z, u, v, w) = R(x, y), S(y, z, v), T(y, v, w), K(y, z, u)$

Solution: y

(d) Write a full semi-join reducer for the query below:
`Q(x,y,z,u,w) = R(x,y),S(y,z),T(z,u),K(z,w)`
Your answer should contain a sequence of semi-join reductions, followed by the

query expression, e.g. (not the real answer):

$$\begin{aligned} R1 =& R \ltimes V \\ R2 =& R1 \ltimes T \\ & \dots \\ Q =& R7 \bowtie S3 \bowtie T5 \bowtie V6 \end{aligned}$$

Solution:

$$\begin{aligned} S1(y,z) =& S(y,z) \ltimes R(x,y) \\ S2(y,z) =& S1(y,z) \ltimes T(z,u) \\ S3(y,z) =& S2(y,z) \ltimes K(z,w) \\ R1(x,y) =& R(x,y) \ltimes S3(y,z) \\ T1(z,u) =& T(z,u) \ltimes S3(y,z) \\ K1(z,w) =& K(z,w) \ltimes S3(y,z) \\ Q(x,y,z,u,w) =& R1(x,y), S3(y,z), T1(z,u), K1(z,w) \end{aligned}$$

Why do we need semi-join reductions? To avoid computing large intermediate results. For example, assume we join in this order $((R \bowtie S) \bowtie T) \bowtie K$ and assume:

$$R = [N] \times [1] \qquad S = \{(1,1)\} \qquad T = [1] \times [N] \qquad K = \{(2,2)\}$$

Then the intermediate result $(R \bowtie S) \bowtie T$ has size N^2, but the final answer is empty (since K doesn't join with anything).

In the semi-join reduction above, we have $S3 = \emptyset$, and then $R1 = T1 = K1 = \emptyset$. Thus, all intermediate results have size $\leq N$, then the final plan joins three empty sets.

159. *(from CSE 344, Autumn 2017, Final)*

Consider the relations below:

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
```

(a) We run the following SQL command:

```
CREATE INDEX newI on Student(zip,name);
```

For each of the following SQL queries indicate whether the index `newI` will speed it up, or will not affect its performance, or will slow it down. Assume that the optimizer is super-smart, and will always choose the absolute best possible plan.

i. `select * from Student where name = 'Alice';`
Faster? The same? Or slower?

Solution: The same.

ii. `select * from Student where zip = 98195;`
Faster? The same? Or slower?

Solution: Faster.

iii.
```
select Student.name, Student.zip
from Student, Takes, Course
where Student.sid=Takes.sid
   and Takes.cid = Course.cid
   and Course.title = 'Databases';
```
Faster? The same? Or slower?

Solution: Faster.

iv. `insert into Student values(1234,'Alice',98195);`
Faster? The same? Or slower?

Solution: Slower.

v. `delete from Student where name = 'Alice';`
Faster? The same? Or slower?

Solution: Slower.

(b) Consider the following statistics on the relations and attributes:

$B(\texttt{Student}) = 2{,}000$	$B(\texttt{Takes}) = 1{,}000{,}000$	$B(\texttt{Course}) = 1{,}000$
$T(\texttt{Student}) = 200{,}000$	$T(\texttt{Takes}) = 10{,}000{,}000$	$T(\texttt{Course}) = 10{,}000$
$V(\texttt{Student}, zip) = 10{,}000$	$V(\texttt{Takes}, sid) = 200{,}000$	$V(\texttt{Course}, dept) = 100$
	$V(\texttt{Takes}, cid) = 5{,}000$	

Estimate the size of the answer of the following SQL query:

```
select *
from Student s, Takes t, Course c
where s.sid = t.sid and t.cid = c.cid
   and s.zip = 98195 and c.dept = 'CSE';
```

Solution:

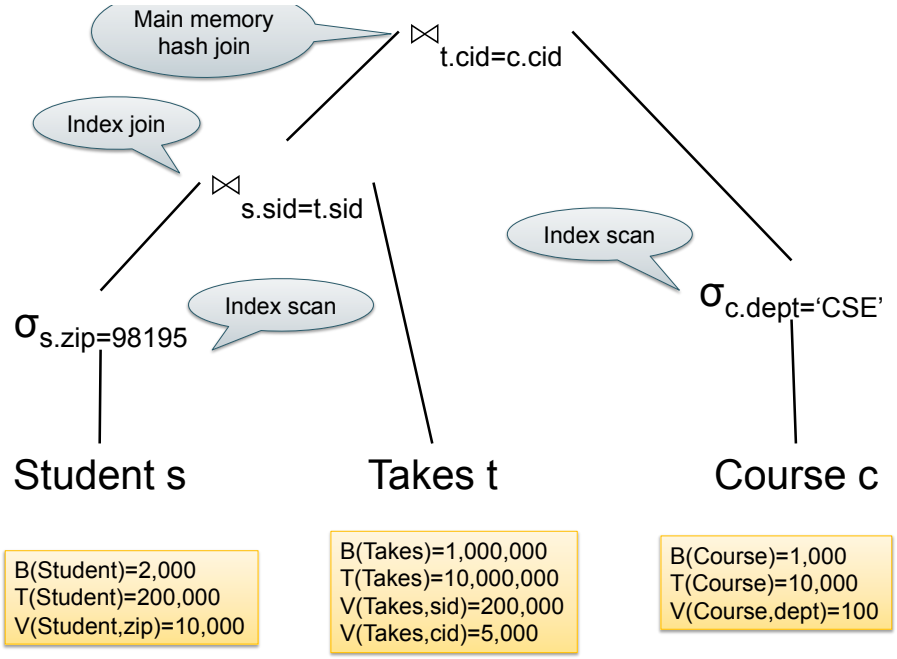
We have:

$$\max(V(\texttt{Student}, sid), V(\texttt{Takes}, sid)) = T(\texttt{Student}) = 200,000$$
$$\max(V(\texttt{Takes}, cid), V(\texttt{Course}, cid)) = V(\texttt{Course}, cid) = T(\texttt{Course}) = 10,000$$

$$= \frac{T(\texttt{Student}) \times T(\texttt{Takes}) \times T(\texttt{Course})}{V(\texttt{Student}, zip) \times V(\texttt{Course}, dept) \times T(\texttt{Student}) \times T(\texttt{Course})}$$
$$= \frac{T(\texttt{Takes})}{V(\texttt{Student}, zip) \times V(\texttt{Course}, dept)}$$
$$= \frac{10,000,000}{10,000 \cdot 100} = 10$$

(c) For each of the physical plans below, estimate its cost. Assume the size of the main memory is $M = 2000$ pages and that all indexes are unclustered, and are stored in main memory (hence accessing them requires zero disk I/O's).

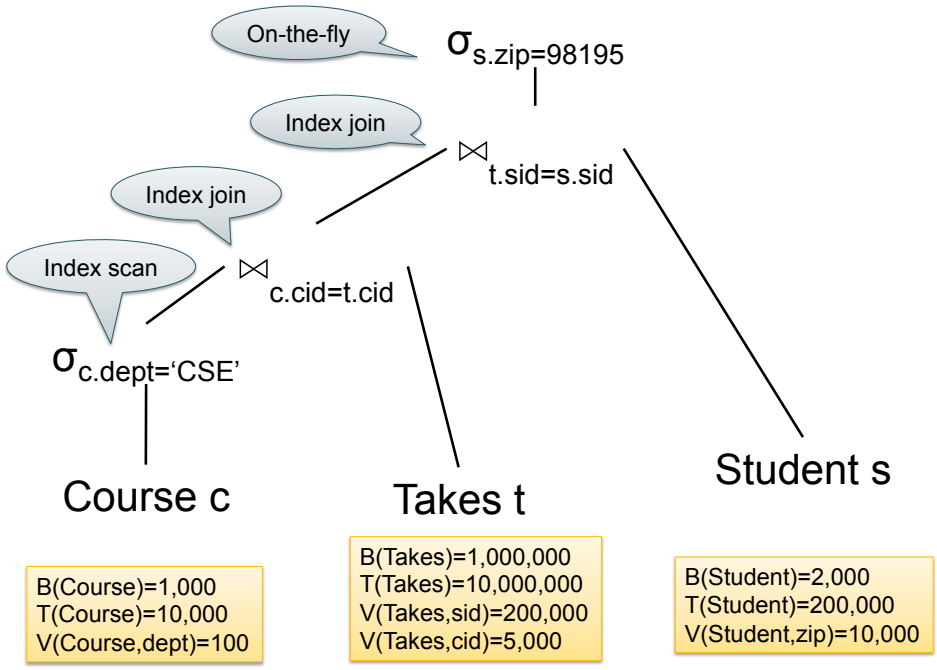
i.



Solution: $T = T(\texttt{Student})/V(zip) = 20$, total cost 20 for the index scan on `Student`; $T = T(\texttt{Takes})/V(sid) = 50$ per student, so the index join with `Takes` costs $20 \cdot 50 = 1000$; $T = T(\texttt{Course})/V(dept) = 100$, costing 100.

$$\text{Total cost} = 20 + 1000 + 100 = 1{,}120$$

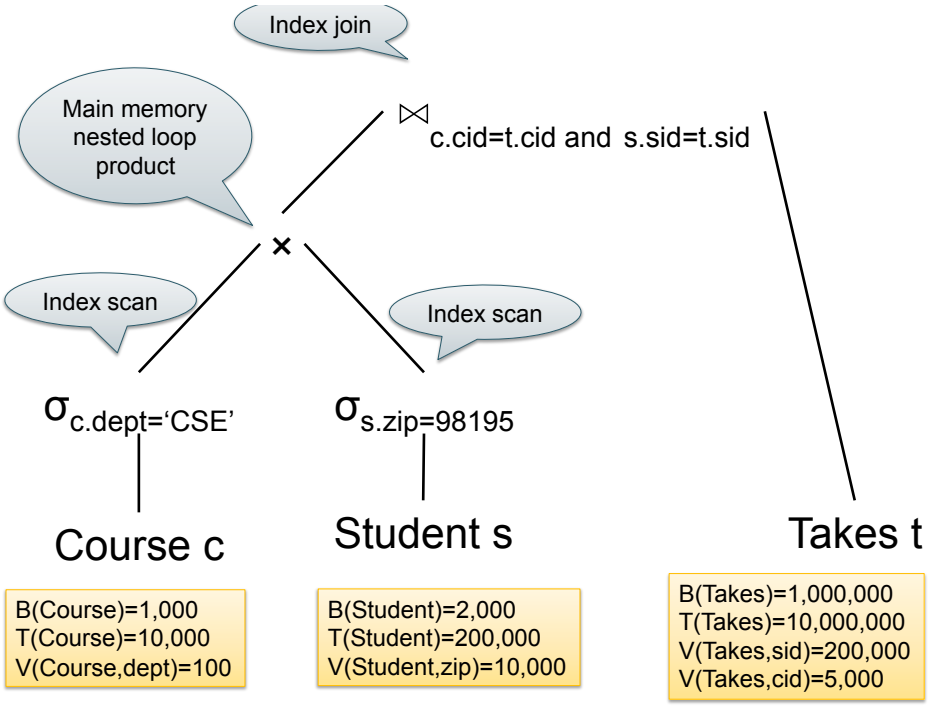
ii.



Solution: $T = T(\texttt{Course})/V(dept) = 100$, total cost 100; $T = T(\texttt{Takes})/V(cid) = 2000$ per course, so the index join costs $100 \cdot 2000 = 200{,}000$; the final index join costs 1 per `Takes` record, i.e. 200,000.

$$\text{Total cost} = 100 + 200{,}000 + 200{,}000 = 400{,}100$$

iii.



Solution: $T = T(\texttt{Course})/V(dept) = 100$, total cost 100; $T = T(\texttt{Student})/V(zip) = 20$ per course, total cost 20; the main memory nested loop product produces $100 \times 20 = 2000$ tuples at a cost of 2000; the index join then costs 1 per (course, student) pair, i.e. 2000.

$$\text{Total cost} = 100 + 2000 + 2000 = 4{,}100$$

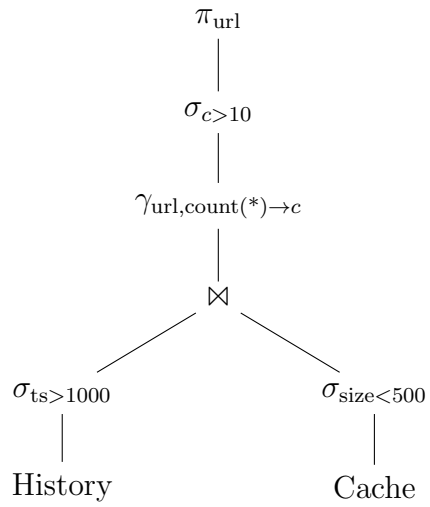
160. *(from CSE 344, Winter 2019, Final)*

(a) In this question we are using the browsing history schema again. Write a logical plan for the following query.

```
select x.url
from History x, Cache y
where x.url = y.url
  and x.ts > 1000 and y.size < 500
group by x.url
having count(*) > 10;
```

You should turn in a relational algebra tree.

Solution:



(b) In this question we consider three relations $R(A, B), S(B, C), T(C, D)$ and the following statistics:

$$\begin{aligned} T(R) &= 10^5 & B(R) &= 100 \\ T(S) &= 6 \cdot 10^6 & B(S) &= 3000 \\ T(T) &= 5 \cdot 10^4 & B(T) &= 40000 \\ V(R, A) &= 5 \cdot 10^4 \\ V(R, B) &= V(S, B) = 3 \cdot 10^3 \\ V(S, C) &= V(T, C) = 2 \cdot 10^4 \\ V(T, D) &= 10^4 \end{aligned}$$

i. Estimate the number of tuples returned by $\sigma_{A=2432}(R)$. You should turn in an integer number.

Solution:

$$\frac{T(R)}{V(R, A)} = \frac{10^5}{5 \cdot 10^4} = 2$$

ii. Estimate number of tuples returned by the following query:

```
SELECT *
FROM R, S, T
WHERE R.A = 2432 and R.B = S.B and S.C = T.C and T.D = 1234
```

You should turn in an integer number.

Solution:

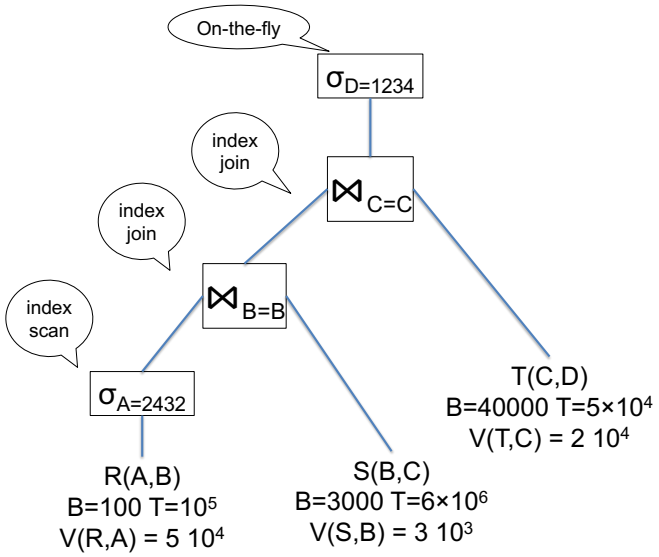
$$\frac{T(R)T(S)T(T)}{V(R,A)V(R,B)V(T,C)V(T,D)} = \frac{10^5 \cdot 6 \cdot 10^6 \cdot 5 \cdot 10^4}{5 \cdot 10^4 \cdot 3 \cdot 10^3 \cdot 2 \cdot 10^4 \cdot 10^4} = 1$$

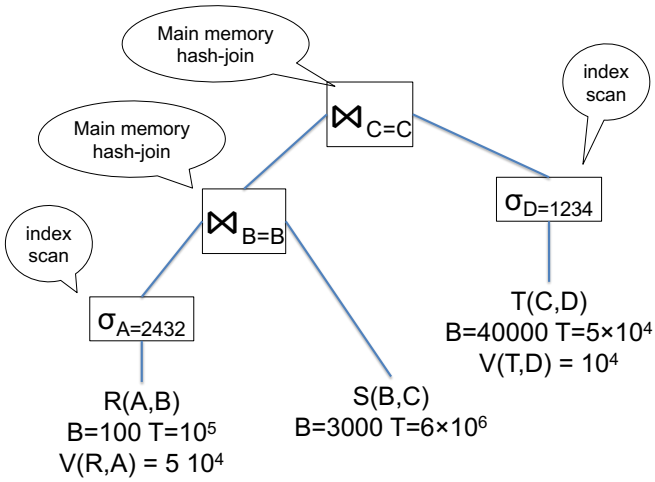
Here and in the previous question, significant partial credit was given if work shown (usually identifying correct variables) was close to what was correct.

iii. Assume the following indices:

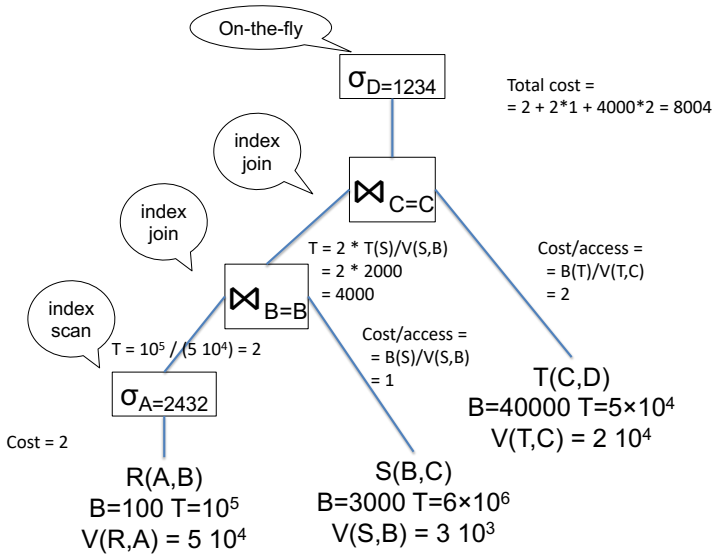
- Unclustered indexes on $R.A$ and $R.B$
- Clustered index in $S.B$, unclustered index on $S.C$.
- Clustered index on $T.C$, unclustered index on $T.D$.

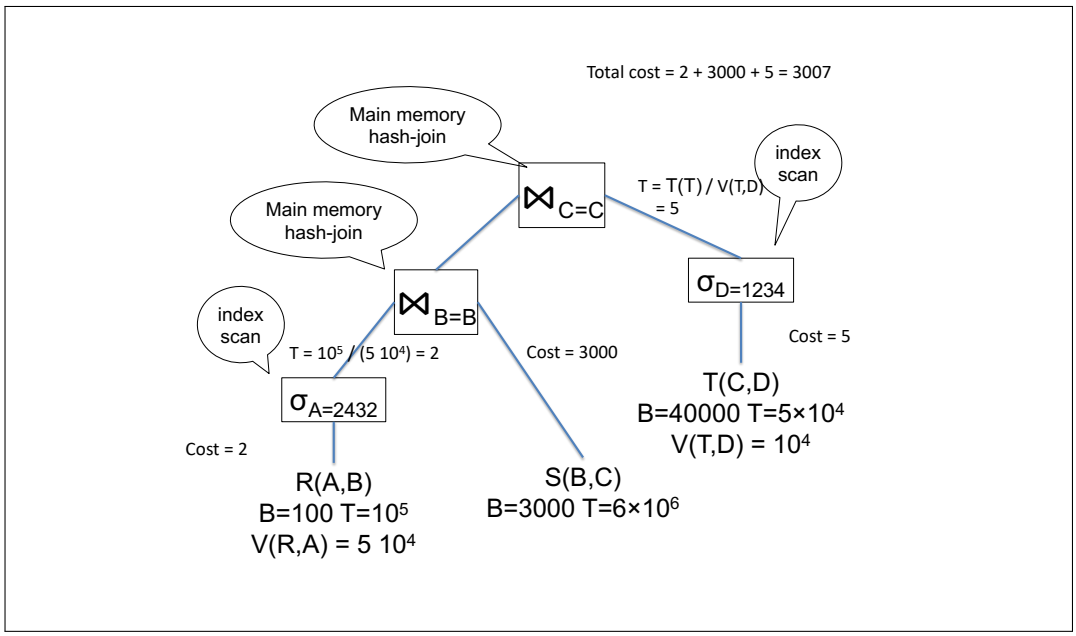
Estimate the I/O cost for two the physical plans below. Use the same statistics as in the previous question (they are shown on the plans, for your convenience).




Main memory hash-join
index scan
Main memory hash-join
⋈ C=C
index scan
⋈ B=B
σ D=1234
σ A=2432
T(C,D)
B=40000 T=5×10^4
V(T,D) = 10^4
R(A,B)
B=100 T=10^5
V(R,A) = 5 10^4
S(B,C)
B=3000 T=6×10^6

Solution:


On-the-fly
σ D=1234
Total cost =
= 2 + 2*1 + 4000*2 = 8004
index join
⋈ C=C
index join
T = 2 * T(S)/V(S,B)
= 2 * 2000
= 4000
Cost/access =
= B(T)/V(T,C)
= 2
index scan
⋈ B=B
T = 10^5 / (5 10^4) = 2
Cost/access =
= B(S)/V(S,B)
= 1
σ A=2432
T(C,D)
B=40000 T=5×10^4
V(T,C) = 2 10^4
Cost = 2
R(A,B)
B=100 T=10^5
V(R,A) = 5 10^4
S(B,C)
B=3000 T=6×10^6
V(S,B) = 3 10^3

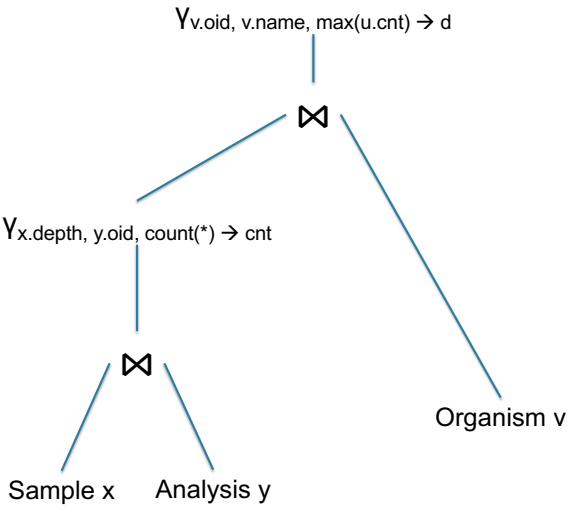


161. *(from CSE 414, Spring 2019, Final)*

(a) Write a logical plan for the following query

```
with  DepthCnt as
   (select x.depth, y.oid, count(*) as cnt
    from Sample x, Analysis y
    where x.sid = y.sid
    group by x.depth, y.oid)
select v.oid, v.name, max(u.cnt) as d
from DepthCnt u, Organism v
where u.oid = v.oid
group by v.oid, v.name;
```

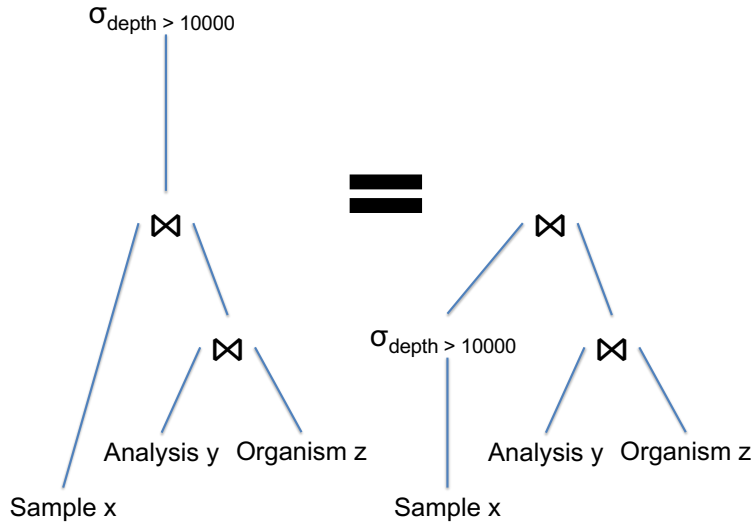
You should turn in a relational algebra tree.
Write your answer below:



Solution:

(b) Indicate which of the optimization rules below are correct. All joins are natural joins:

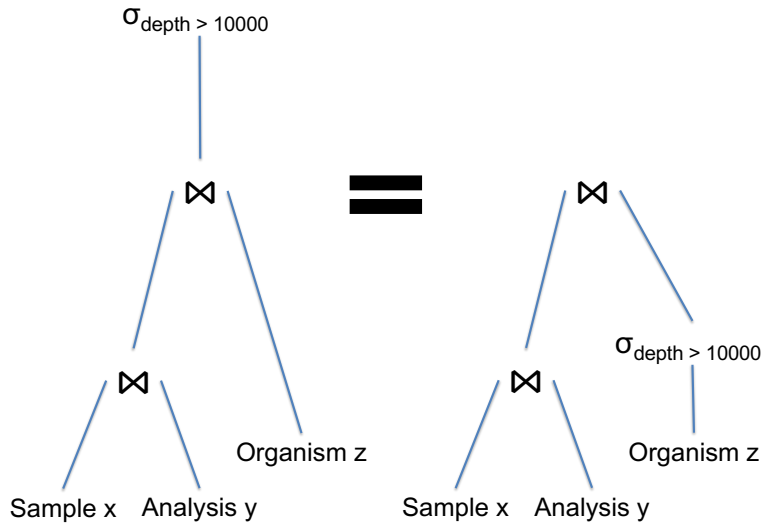
i.



Correct? [Yes/no]:

Solution: Yes

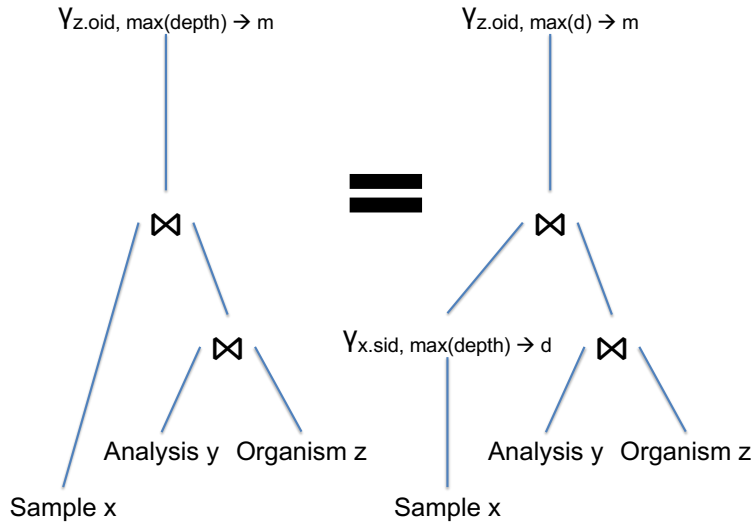
ii.



Correct? [Yes/no]:

Solution: No

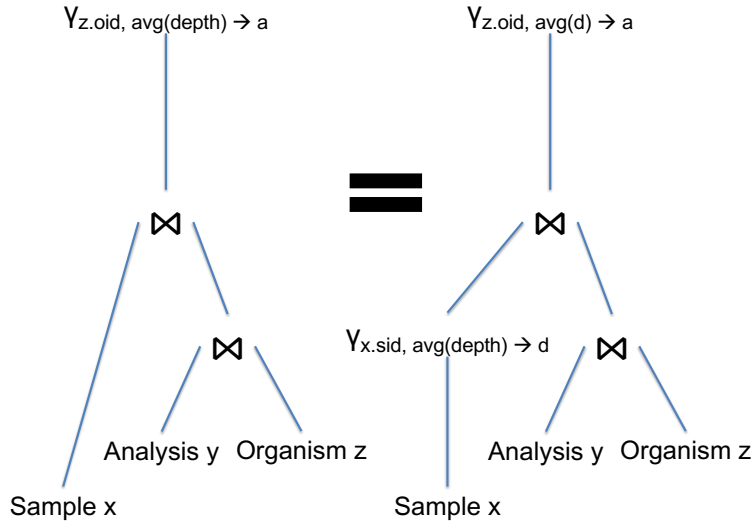
iii.



Correct? [Yes/no]:

Solution: yes

iv.



Correct? [Yes/no]:

Solution: YES(see note)

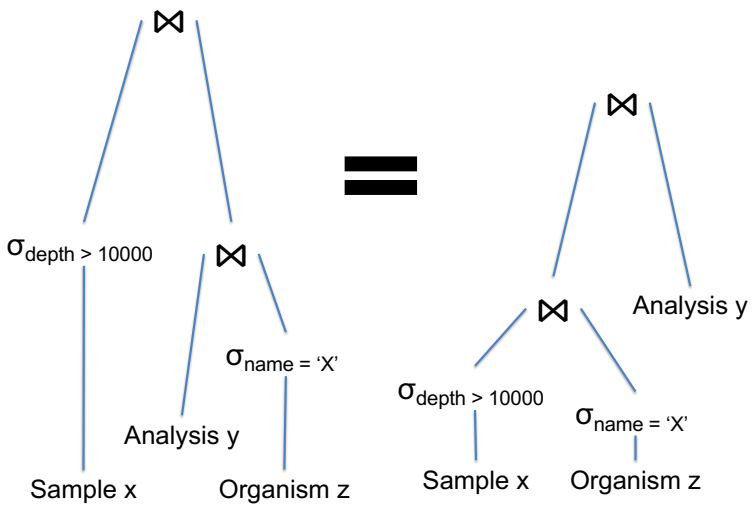
Solution: Note The previous two questions were designed wrongly. The lower group-by on the right is applied to a key, so it is trivially a no-op. As stated, both equalities hold trivially, because the extra group by does nothing, so the answer is "YES" for both.
The correct design should have applied the new group on a join:

$$\gamma_{z.oid,\max(d)\to m}\left(\gamma_{x.sid,\max(depth)\to d}((\texttt{Sample}\ x)\bowtie(\texttt{Analysis}\ y))\bowtie\texttt{Organism}\ z\right)$$
$$\gamma_{z.oid,avg(d)\to a}\left(\gamma_{x.sid,avg(depth)\to d}((\texttt{Sample}\ x)\bowtie(\texttt{Analysis}\ y))\bowtie\texttt{Organism}\ z\right)$$

The first answer is "yes", while the second is "no", because the average of averages is not always the average. For example:

$$avg(1,1,9,11,13)=7\neq avg(avg(1,1),avg(9,11,13))=avg(1,11)=6$$

v.



Correct? [Yes/no]:

Solution: yes

(c) Assume the following statistics on the database:

$T(\texttt{Sample}) = 100,000$	$T(\texttt{Analysis}) = 20,000,000$	$T(\texttt{Organism}) = 60,000$
$V(\texttt{Sample}, \texttt{technician}) = 100$	$V(\texttt{Analysis}, \texttt{sid}) = 10,000$	$V(\texttt{Organism}, \texttt{name}) = 15,000$
	$V(\texttt{Analysis}, \texttt{oid}) = 45,000$	

Estimate the size of the answer to the following SQL query. You should make the usual uniformity, independence, and preservation of values assumption that we used in class:

```
select *
from Sample x, Analysis y, Organism z
where x.sid = y.sid and y.oid = z.oid
  and x.technician = 'Alice'
  and z.name = 'Synechococcus';
```

Solution:

$$\frac{T(\texttt{Sample}) \cdot T(\texttt{Analysis}) \cdot T(\texttt{Organism})}{\underbrace{V(\texttt{Sample}, \texttt{sid})}_{>V(\texttt{Analysis},\texttt{sid})} \cdot \underbrace{V(\texttt{Organism}, \texttt{oid})}_{>V(\texttt{Analysis},\texttt{oid})} \cdot V(\texttt{Sample}, \texttt{technician}) \cdot V(\texttt{Organism}, \texttt{name})}$$

$$= \frac{T(\texttt{Analysis})}{V(\texttt{Sample}, \texttt{technician}) \cdot V(\texttt{Organism}, \texttt{name})}$$

$$= \frac{20,000,000}{100 \cdot 15,000} = 13.3$$

(d) The depths of the samples are highly skewed: for each additional 100m, the number of samples is reduced by half. (That is, half of the samples have depth < 100m; half of the rest have depth < 200m; half of the rest have depth < 300m, etc.). There is an unclustered, B^+ index on `Sample.depth`. Indicate the optimal physical plan that the optimizer should choose for each of the two queries below. Choose between *Table-scan and on-the-fly selection,* or *Index selection.*

i.

```
Q1: select * from Sample where depth < 100;
```

Table scan or index selection?

Solution: Table scan

ii.

```
Q1: select * from Sample where depth > 8000;
```

Table scan or index selection?

Solution: Index Selection

162. *(from CSE 544p, Spring 2021, Final)*

(a) Write a logical plan for the following query.

```
select R.A, sum(T.D)
from R, S, T
where R.B = S.B and S.C = T.C
group by R.A
having count(*) > 20
```

Solution:

```
T1 = R join S
T2 = T1 join T
T3 = gamma[A, sum(D) as S, count(*) as C](T2)
Answer = filter[C > 20](T3)
```

(b) Write a full semi-join reducer for the query below:
`Q(x,y,z,u,w) = R(x,y),S(y,z),T(z,u),K(z,w)`
Your answer should contain a sequence of semi-join reductions, followed by the query expression, e.g. (not the real answer):

$$R1 = R \ltimes V$$
$$R2 = R1 \ltimes T$$
$$\ldots$$
$$Q = R7 \bowtie S3 \bowtie T5 \bowtie V6$$

Solution:

$$\begin{aligned}
S1(y,z) &= S(y,z) \ltimes R(x,y) \\
S2(y,z) &= S1(y,z) \ltimes T(z,u) \\
S3(y,z) &= S2(y,z) \ltimes K(z,w) \\
R1(x,y) &= R(x,y) \ltimes S3(y,z) \\
T1(z,u) &= T(z,u) \ltimes S3(y,z) \\
K1(z,w) &= K(z,w) \ltimes S3(y,z) \\
Q(x,y,z,u,w) &= R1(x,y), S3(y,z), T1(z,u), K1(z,w)
\end{aligned}$$

(c) In this question we consider three relations $R(A,B), S(B,C), T(C,D)$ and the following statistics:

$$\begin{aligned}
T(R) &= 10^5 & B(R) &= 100 \\
T(S) &= 6 \cdot 10^6 & B(S) &= 1000 \\
T(T) &= 4 \cdot 10^4 & B(T) &= 100 \\
V(R,B) &= V(S,B) = 3 \cdot 10^3 & & \\
V(S,C) &= V(T,C) = 2 \cdot 10^4 & &
\end{aligned}$$

In addition, we have two histograms, on $R.A$ and $T.D$:

$R.A$	$0 \ldots 999$	$1000 \ldots 1999$	$2000 \ldots 2999$	$3000 \ldots 3999$	$4000 \ldots 4999$
	10^4	$2 \cdot 10^4$	$3 \cdot 10^4$	$2 \cdot 10^4$	$2 \cdot 10^4$

$T.D$	$0 \ldots 2499$	$2500 \ldots 2699$	$2700 \ldots 3999$	$4000 \ldots 7999$
	10^4	10^4	10^4	10^4

assuming the two histograms above, plus the following statistics:

i. For each of the histograms, indicate whether it is an eqdepth or an eqwidth histogram.

Solution: $R.A$ is eqwidth, $T.D$ is eqdepth.

ii. Estimate the number of tuples returned by $\sigma_{500 \le A \le 3499}(R)$. You should turn in an integer number.

Solution:
$10^4/2 + 2 \cdot 10^4 + 3 \cdot 10^4 + 2 \cdot 10^4/2 = 65000$

iii. Estimate number of tuples returned by the following query:

```
SELECT *
FROM R, S, T
WHERE R.A = 2432 and R.B = S.B and S.C = T.C and T.D = 1234
```

You should turn in an integer number.

Solution: Let $R' = \sigma_{A=2432}(R)$ and $T' = \sigma_{D=1234}(T)$. Their number of tuples are:

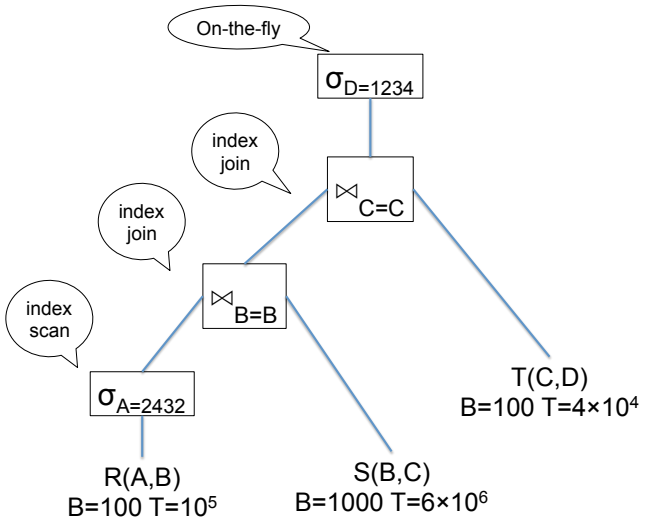
$$T(R') = \frac{3 \cdot 10^4}{1000}$$
$$T(T') = \frac{10^4}{2500}$$

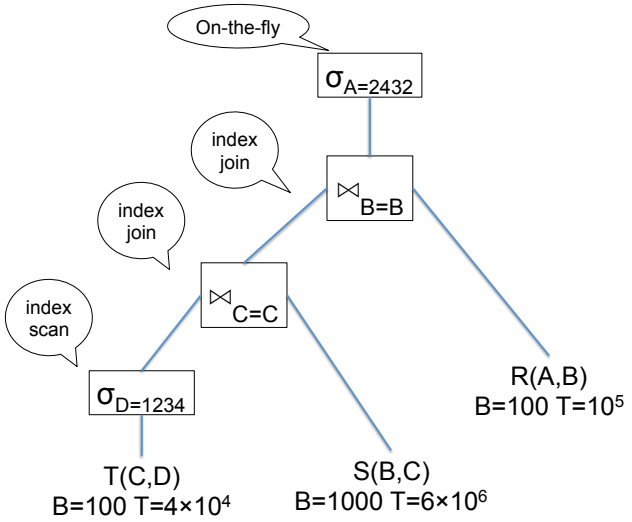
The estimated number of tuples is:

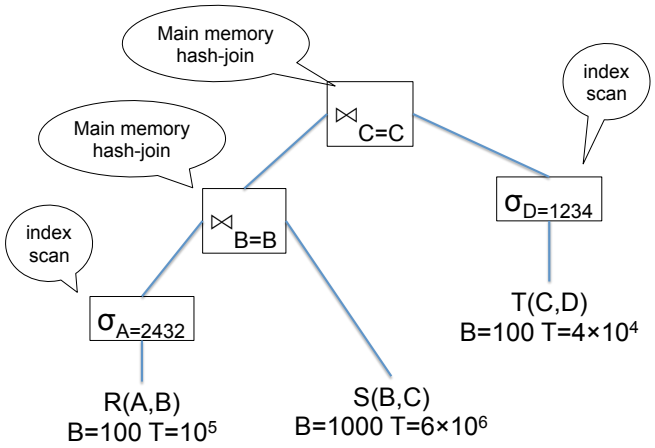
$$T(R') \times \frac{T(S)}{V(S,B) \cdot V(S,C)} \times T(T') = \frac{3 \cdot 10^4}{1000} \times \frac{6 \cdot 10^6}{3 \cdot 10^3 \cdot 2 \cdot 10^4} \times \frac{10^4}{2500} = 12$$

iv. Assume the following indices:
- Unclustered indexes on $R.A$ and $R.B$
- Clustered index in $S.B$, unclustered index on $S.C$.
- Unclustered indexes on $T.C$ and $T.D$.

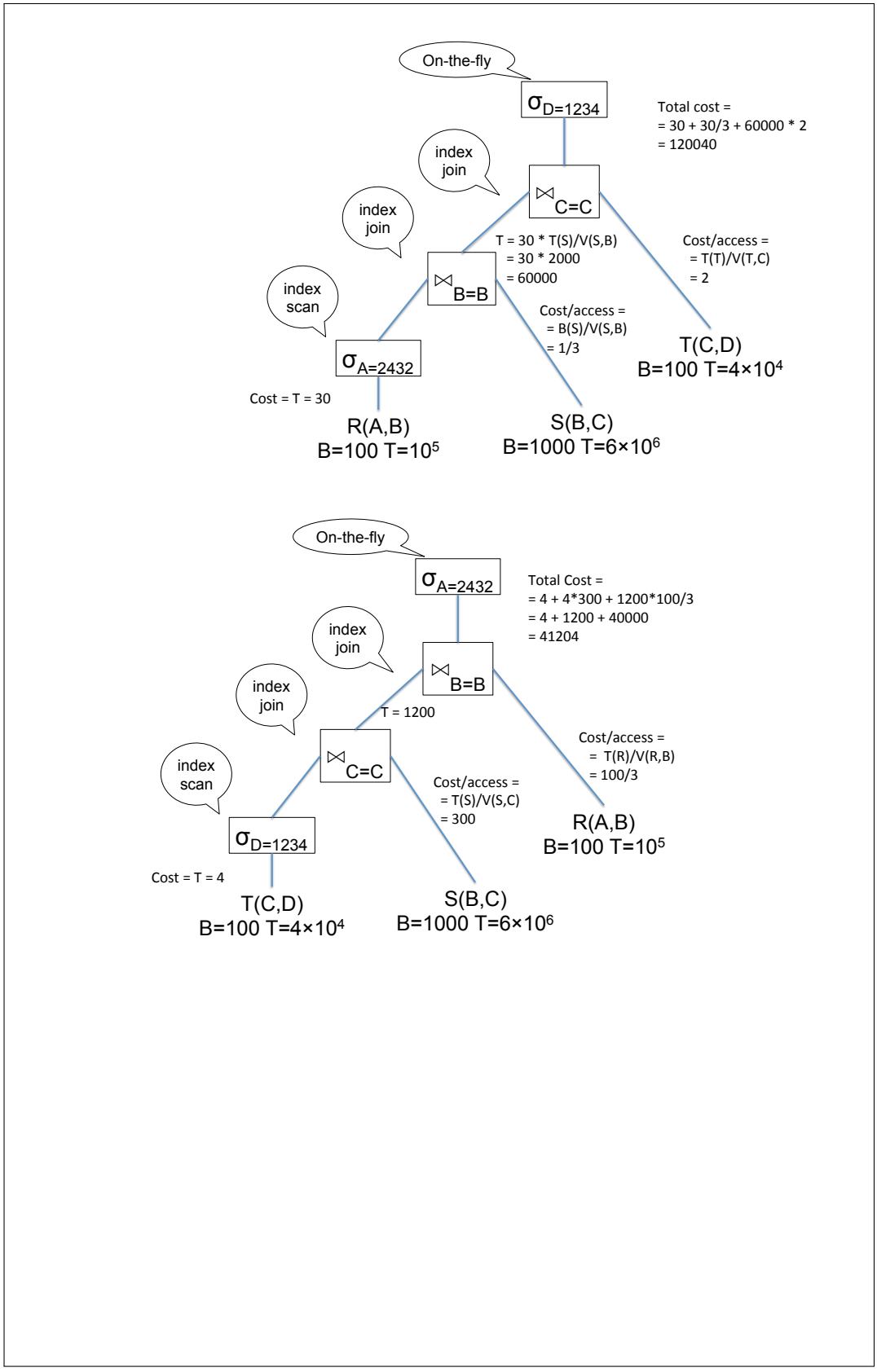
Estimate the I/O cost for each physical plan below:

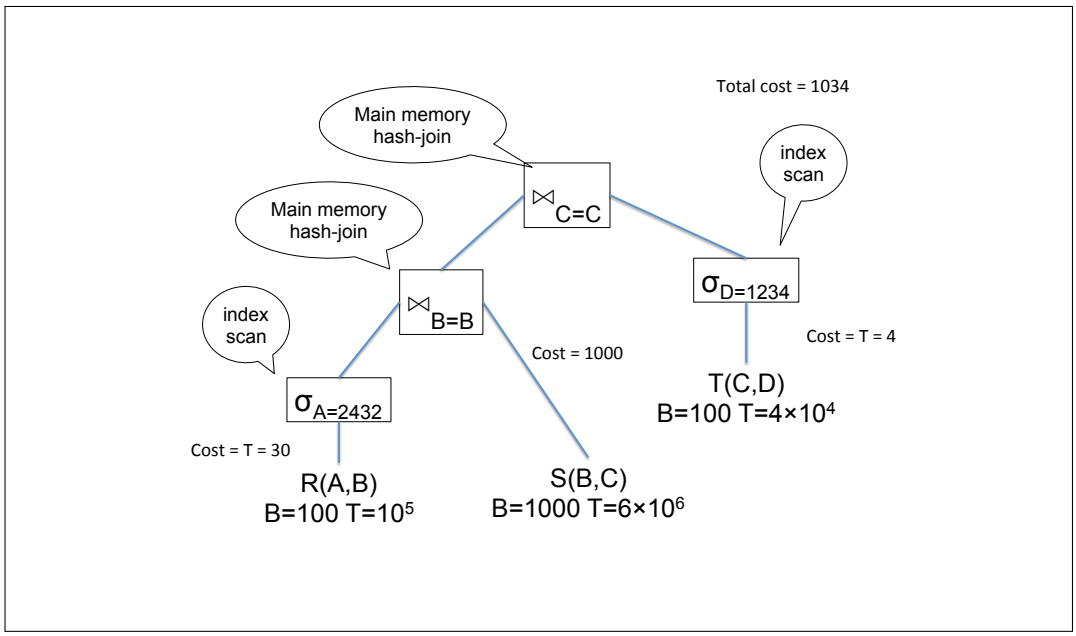



On-the-fly
σA=2432
index join
⋈B=B
index join
⋈C=C
index scan
σD=1234
R(A,B)
B=100 T=10^5
T(C,D)
B=100 T=4×10^4
S(B,C)
B=1000 T=6×10^6


Main memory hash-join
⋈C=C
index scan
Main memory hash-join
⋈B=B
σD=1234
index scan
σA=2432
T(C,D)
B=100 T=4×10^4
R(A,B)
B=100 T=10^5
S(B,C)
B=1000 T=6×10^6

Solution:


On-the-fly
σD=1234
Total cost =
= 30 + 30/3 + 60000 * 2
= 120040
index join
⋈C=C
index join
T = 30 * T(S)/V(S,B)
= 30 * 2000
= 60000
Cost/access =
= T(T)/V(T,C)
= 2
⋈B=B
index scan
Cost/access =
= B(S)/V(S,B)
= 1/3
σA=2432
T(C,D)
B=100 T=4×10^4
Cost = T = 30
R(A,B)
B=100 T=10^5
S(B,C)
B=1000 T=6×10^6
On-the-fly
σA=2432
Total Cost =
= 4 + 4*300 + 1200*100/3
= 4 + 1200 + 40000
= 41204
index join
⋈B=B
index join
T = 1200
⋈C=C
Cost/access =
= T(R)/V(R,B)
= 100/3
index scan
Cost/access =
= T(S)/V(S,C)
= 300
σD=1234
R(A,B)
B=100 T=10^5
Cost = T = 4
T(C,D)
B=100 T=4×10^4
S(B,C)
B=1000 T=6×10^6



163. *(from CSE 414, Spring 2024, Final; CSE 344, Spring 2026, Makeup Final)*

Consider the following schema about courses and the students' grades in these courses:

```
Student(sid, name)
Enroll(sid, cid, grade)
Course(cid, title)
```

(a) You need to run the following query template many times every day:

```
SELECT *
FROM Enroll
WHERE cid = ? and grade = ?;
```

You can create only one index to speed up these queries, and are considering one of the following five options. Indicate in each case whether that index can help speed up your queries:

i. The index `Enroll(sid, grade)` can help speedup the queries.
True or False?

Solution: False

ii. The index `Enroll(grade, sid)` can help speedup the queries.
True or False?

Solution: True

iii. The index `Enroll(sid)` can help speedup the queries.

True or False?

Solution: False

iv. The index `Enroll(cid, grade)` can help speedup the queries.
True or False?

Solution: True

v. The index `Enroll(grade, cid)` can help speedup the queries.
True or False?

Solution: True

(b) The database system has the following statistics on this database;

$T(\texttt{Student}) = 50000$	$T(\texttt{Enroll}) = 100000$	$T(\texttt{Course}) = 600$
$V(\texttt{Student}, \texttt{sid}) = 50000$	$V(\texttt{Enroll}, \texttt{sid}) = 40000$	$V(\texttt{Course}, \texttt{cid}) = 600$
$V(\texttt{Student}, \texttt{name}) = 40000$	$V(\texttt{Enroll}, \texttt{cid}) = 500$	
	$V(\texttt{Enroll}, \texttt{grade}) = 40$	

For each query below indicate the number of output tuples estimated by the cardinality estimator. Show your work, and round the answer to the nearest integer.

For example, if the query Q were:
```
SELECT * FROM Student
```
then your answer would be: $\texttt{Est}(Q) = 50000$.

i. The query Q is:
```
SELECT * FROM Student
WHERE name = 'Alice';
```
$\texttt{Est}(Q) =$

Solution:

$$\frac{T(\texttt{Student})}{V(\texttt{Student}, \texttt{name})} = \frac{50000}{40000} = 1.2 \approx 1$$

ii. The query Q is:
```
SELECT * FROM Enroll
WHERE cid='cse414'
  and grade = 3.0;
```
$\texttt{Est}(Q) =$

Solution:

$$\frac{T(\texttt{Enroll})}{V(\texttt{Enroll},\texttt{cid}) \cdot V(\texttt{Enroll},\texttt{grade})} = \frac{100000}{500 \cdot 40} = 5$$

iii. The query Q is:

```
SELECT *
FROM Student x, Enroll y
WHERE x.sid = y.sid;
```

$\texttt{Est}(Q) =$

Solution:

$$\frac{T(\texttt{Student}) \cdot T(\texttt{Enroll})}{\max(V(\texttt{Student},\texttt{sid}), V(\texttt{Enroll},\texttt{sid}))} = \frac{50000 \cdot 100000}{50000} = 100000$$

iv. The query Q is:

```
SELECT *
FROM Student x, Enroll y
9ERE x.sid = y.sid
  and x.name = 'Alice'
  and y.grade = 3.0;
```

$\texttt{Est}(Q) =$

Solution:

$$\frac{T(\texttt{Student}) \cdot T(\texttt{Enroll})}{\max(V(\texttt{Student},\texttt{sid}), V(\texttt{Enroll},\texttt{sid})) \cdot V(\texttt{Student},\texttt{name}) \cdot V(\texttt{Enroll},\texttt{grade})}$$
$$= \frac{50000 \cdot 100000}{50000 \cdot 40000 \cdot 40} \approx 1$$

v. The query Q is:

```
SELECT *
FROM Student x,
     Enroll y, Course z
WHERE x.sid = y.sid
  and y.cid = z.cid
```

$\texttt{Est}(Q) =$

Solution:

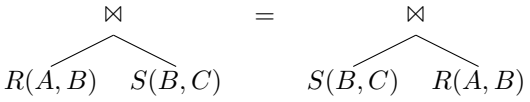
$$\frac{T(\texttt{Student}) \cdot T(\texttt{Enroll}) \cdot T(\texttt{Course})}{\max(V(\texttt{Student}, \texttt{sid}), V(\texttt{Enroll}, \texttt{sid})) \cdot \max(V(\texttt{Enroll}, \texttt{cid}), V(\texttt{Course}, \texttt{cid}))} =$$

$$= \frac{50000 \cdot 100000 \cdot 600}{50000 \cdot 600} = 100000$$

164. *(from CSE 344, Autumn 2024, Final; CSE 344, Spring 2026, Final)*

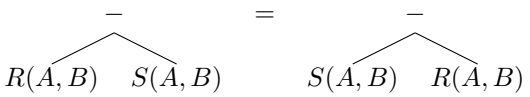
(a) You are implementing a database management system and consider which rewrite rules to include in your optimizer. You have several candidates: for each of the candidate rules below indicate whether they are correct or not. All expressions use set semantics (i.e. no duplicates), and joins are natural joins (including outer joins, e.g. $R(A,B) \leftouterjoin S(B,C)$ means $R(A,B) \leftouterjoin_{R.B=S.B} S(B,C)$).

For example, if the rule were:



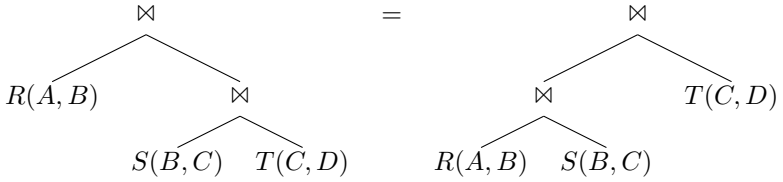
then your answer should be *Yes*.

But if the rule were:



then your answer should be *No*, because these two expressions are not equivalent.

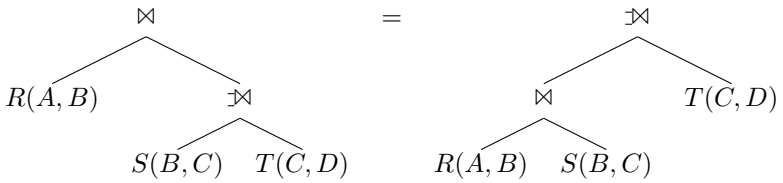
i. Is this rule correct?



Yes or no?

Solution: YES

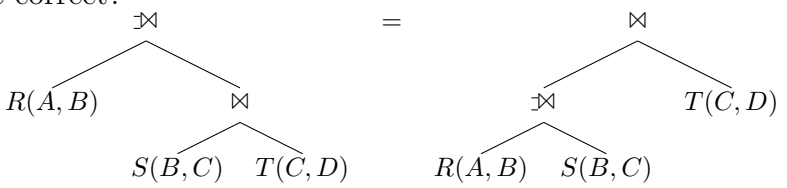
ii. Is this rule correct? ($\leftouterjoin$ is Left Outer Join)



Yes or no?

Solution: YES

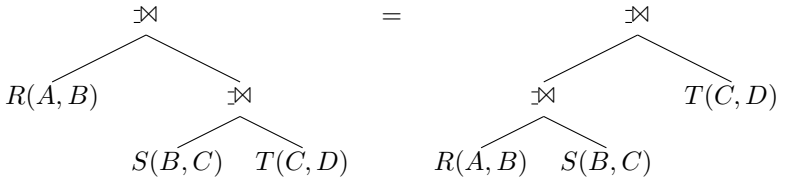
iii. Is this rule correct?



Yes or no?

Solution: NO

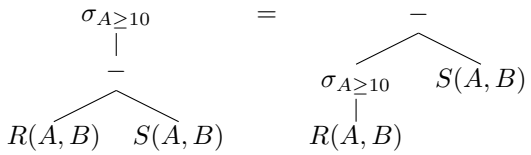
iv. Is this rule correct?



Yes or no?

Solution: YES

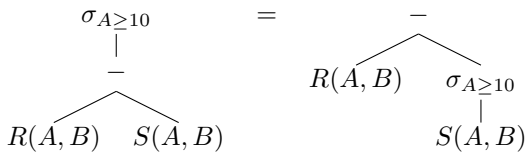
v. Is this rule correct?



Yes or no?

Solution: YES

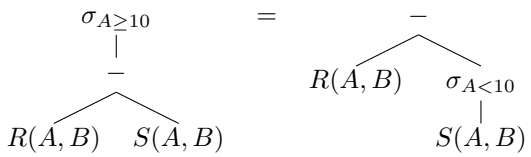
vi. Is this rule correct?



Yes or no?

Solution: NO

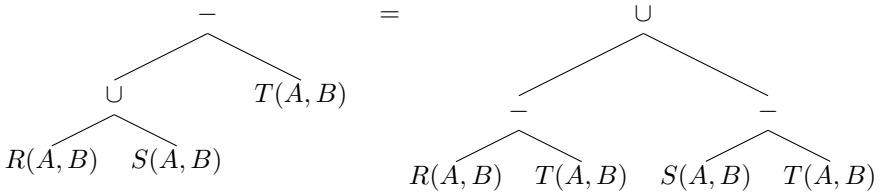
vii. Is this rule correct?



Yes or no?

Solution: NO

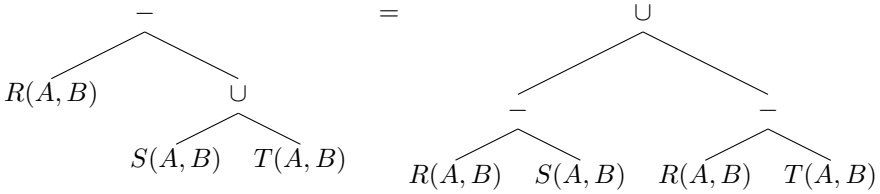
viii. Is this rule correct?



Yes or no?

Solution: YES

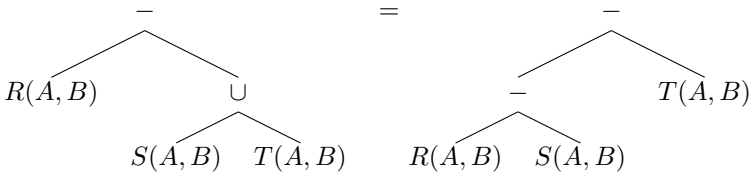
ix. Is this rule correct?



Yes or no?

Solution: NO

x. Is this rule correct?



Yes or no?

Solution: YES

(b) Consider the following relation and associated statistics:

`Product(`<u>`pid`</u>`,pname,price,color,size)`

$T(\texttt{Product}) = 6,000,000 \quad B(\texttt{Product}) = 5,000$

$V(\texttt{Product}, \texttt{price}) = 200 \quad V(\texttt{Product}, \texttt{color}) = 25 \quad V(\texttt{Product}, \texttt{size}) = 8$

The database administrator creates the following 5 unclustered indexes:

```
Product(color)
Product(size)
Product(price,color)
Product(color,size)
Product(size,price)
```

The first two indices are on a single attribute, the last three are on two attributes. The optimizer needs to execute the following query:

```
select *
from  Product
where color = 'red'
  and size = 'medium';
```

(continued on the next page)

The optimizer considers several options for computing the query. For each option, indicate whether that option is even possible, and, if *yes*, then compute the number of disk I/Os necessary to answer the query using that option. Recall that for an index lookup you only need to count the pages read from the data file, not the pages of the index.

i. The optimizer uses the index `Product(color)`.
Is this possible? What is the cost?

Solution: Yes, Cost= $\frac{T}{V} = \frac{6000000}{25} = 240000$

ii. The optimizer uses the index `Product(size)`.
Is this possible? What is the cost?

Solution: Yes, Cost= $\frac{T}{V} = \frac{6000000}{8} = 750000$

iii. The optimizer uses the index `Product(price,color)`.
Is this possible? What is the cost?

Solution: No

iv. The optimizer uses the index `Product(color,size)`.
Is this possible? What is the cost?

Solution: Yes, Cost= $\frac{T}{V(\texttt{color})V(\texttt{size})} = \frac{6000000}{25*8} = 30000$

v. The optimizer uses the index `Product(size,price)`.
Is this possible? What is the cost?

Solution: Yes, Cost= $\frac{T}{V} = \frac{6000000}{8} = 750000$

(c) We have a database with the following schema and statistics:

`Product(`<u>`pid`</u>`,pname,price,color,size)`	$T(\texttt{Product}) = 6,000,000$	$V(\texttt{Product}, \texttt{color}) = 25$, $V(\texttt{Product}, \texttt{size}) = 8$ $\texttt{MAX}(\texttt{price}) = 100$ $\texttt{MAX}(\texttt{price}) = 100$
`Purchase(pid, uid, year)`	$T(\texttt{Purchase}) = 75,000,000$	$V(\texttt{Purchase}, \texttt{pid}) = 4,700,000$, $V(\texttt{Purchase}, \texttt{uid}) = 760,000$, $V(\texttt{Purchase}, \texttt{year}) = 40$
`User(`<u>`uid`</u>`,uname,zip)`	$T(\texttt{User}) = 1,000,000$	$V(\texttt{User}, \texttt{zip}) = 20,000$

Since `pid` is a key, $V(\texttt{Product}, \texttt{pid}) = T(\texttt{Product})$, and similarly for `uid`.

Compute the estimated size of the answer of each of the queries below:

i.
```
SELECT * FROM Product
WHERE color='blue'
  and size='medium';
```

Solution: $\texttt{Est}(Q) = \frac{T(\texttt{Product})}{V(\texttt{Product},\texttt{color}) \cdot V(\texttt{Product},\texttt{size})} = \frac{6{,}000{,}000}{25 \cdot 8} = 30,000$

ii.
```
SELECT * FROM Product x, Purchase y
WHERE x.pid = y.pid
  and x.color='red' and y.year=2018;
```

Solution: We have $V(\texttt{Product}, \texttt{pid}) > V(\texttt{Purchase}, \texttt{pid})$ and we obtain:
$\texttt{Est}(Q) = \frac{T(\texttt{Product}) \cdot T(\texttt{Purchase})}{V(\texttt{Product},\texttt{pid}) \cdot V(\texttt{Product},\texttt{color}) \cdot V(\texttt{Purchase},\texttt{year})} = \frac{75{,}000{,}000}{25 \cdot 40} = 75,000$

iii.
```
SELECT * FROM Product x, Purchase y, User z
WHERE x.pid = y.pid and y.uid = z.uid
  and x.color='red' and z.zip = 98195;
```

Solution: Same reasoning as above gives us:
$\texttt{Est}(Q) = \frac{T(\texttt{Purchase})}{V(\texttt{Product},\texttt{color}) \cdot V(\texttt{User},\texttt{zip})} = \frac{75{,}000{,}000}{25 \cdot 20{,}000} = 150$

iv.
```
SELECT * FROM Product
WHERE 26 <= price and price <= 45;
```

Solution: Same reasoning as above gives us:
$\texttt{Est}(Q) = T(\texttt{Product}) \frac{45-26+1}{100-1+1} = 6,000,000 \cdot \frac{20}{100} = 1,200,000$

v. Revise item iv, given this histogram on `Product.price`:

`price:`	1 . . . 20	21 . . . 30	31 . . . 35	36 . . . 40	41 . . . 50
`#tuples`	500,000	1,000,000	2,000,000	2,000,000	500,000

Solution: $\texttt{Est}(Q) = 1,000,000\frac{1}{2} + 2,000,000 + 2,000,000 + 500,000\frac{1}{2} = 4,750,000$

(d) In this problem you will compute the largest possible output of some natural join expressions. Assume all relations have size N:

$$|R| = |S| = |T| = N$$

You will be asked to answer whether the maximum output size is N or N^2 or N^3 or N^4. For example if the expression were the following natural join:

$$R(A, B) \bowtie S(B, C)$$

the answer is N^2; answer N is wrong, because the result of the join can be larger than N, and N^3 is also wrong, because the size of the join can never exceed N^2.

i. What is the maximum size of $R(A, B) \bowtie S(C, D)$?
Circle one: N or N^2 or N^3 or N^4

Solution: N^2

ii. What is the maximum size of $R(A, B) \bowtie S(A, B)$?
Circle one: N or N^2 or N^3 or N^4

Solution: N

iii. What is the maximum size of $R(A, B) \bowtie S(A, C) \bowtie T(A, D)$?
Circle one: N or N^2 or N^3 or N^4

Solution: N^3

iv. What is the maximum size of $R(A, B) \bowtie S(B, C) \bowtie T(C, D)$?
Circle one: N or N^2 or N^3 or N^4

Solution: N^2

v. What is the maximum size of $R(A, B) \bowtie S(B, C) \bowtie T(B, C)$?
Circle one: N or N^2 or N^3 or N^4

Solution: N^2

13 Cost and Cardinality Estimation (165)

In order to compare two candidate plans an optimizer needs to estimate their costs, and, for that, it needs to estimate the number of tuples in each intermediate result, i.e. its cardinality. This short section is about that estimation: using histograms to predict the selectivity of a predicate, and combining those estimates across a join.

The standard assumptions are uniformity within a bucket, independence between predicates, and containment of value sets across a join. They are of course wrong in general, but we need them in order to do any cardinality estimation. The interesting exam questions are the ones that ask when they fail and in which direction the estimate errs. Related material appears in the preceding section, where the estimates feed directly into the choice of plan.

165. *(from CSE 444, Spring 2010, Final, CSE 544p, Autumn 2010, Final; CSE 544p, Winter 2014, Final; CSE 544p, Autumn 2015, Final)*

Consider the relations $R(A, B), S(B, C), T(C, D)$ and the following histograms on $R.A$ and $T.D$:

$R.A$	$0 \ldots 999$	$1000 \ldots 1999$	$2000 \ldots 2999$	$3000 \ldots 3999$	$4000 \ldots 4999$
	10^4	$2 \cdot 10^4$	$3 \cdot 10^4$	$2 \cdot 10^4$	$2 \cdot 10^4$

$T.D$	$0 \ldots 2499$	$2500 \ldots 2699$	$2700 \ldots 3999$	$4000 \ldots 7999$
	10^4	10^4	10^4	10^4

(a) Indicate the type of the kind of the histogram $R.A$:
Eqwidth or Eqdepth ?

Solution: Eqwidth

(b) Indicate the type of the histogram $T.D$:
Eqwidth or Eqdepth ?

Solution: Eqdepth

(c) Estimate number of tuples returned by $\sigma_{500 \leq A \leq 3499}(R)$. You should turn in an integer number.

Solution: $10^4/2 + 2 \cdot 10^4 + 3 \cdot 10^4 + 2 \cdot 10^4/2 = 65000$

(d) Estimate number of tuples returned by the following query:

```
SELECT *
FROM R, S, T
WHERE R.A = 2432 and R.B = S.B and S.C = T.C and T.D = 1234
```

assuming the two histograms above, plus the following statistics:

$$\begin{aligned} T(R) &= 10^5 \\ T(S) &= 6 \cdot 10^6 \\ T(T) &= 4 \cdot 10^4 \\ V(R, B) &= V(S, B) = 3 \cdot 10^3 \\ V(S, C) &= V(T, C) = 2 \cdot 10^4 \end{aligned}$$

You should turn in an integer number.

Solution: Let $R' = \sigma_{A=2432}(R)$ and $T' = \sigma_{D=1234}(T)$. Their number of tuples are:

$$T(R') = \frac{3 \cdot 10^4}{1000}$$
$$T(T') = \frac{10^4}{2500}$$

The estimated number of tuples is:

$$T(R') \times \frac{T(S)}{V(S,B) \cdot V(S,C)} \times T(T') = \frac{3 \cdot 10^4}{1000} \times \frac{6 \cdot 10^6}{3 \cdot 10^3 \cdot 2 \cdot 10^4} \times \frac{10^4}{2500} = 12$$

14 Transactions (166–181)

Transactions cover two largely independent concerns, and the questions here divide accordingly. Concurrency control asks whether a given schedule is serializable: draw the precedence graph, look for a cycle, and if there is none read off a serial order. Beyond that come the protocols: two-phase locking, strict 2PL, and timestamp-based control. Some questions concern deadlocks than can or cannot be produced.

Recovery is the other half. Given an undo, redo, or undo/redo log and a crash, the questions ask which transactions must be undone, which redone, and in what order, and what a non-quiescent checkpoint lets the system skip. The rules are short but unforgiving: the sequence in which log records reach disk relative to the data they describe is the whole subject.

A few problems ask about isolation levels and the anomalies each one permits, which is where the theory meets what a real system actually offers.

166. *(from CSE 444, Autumn 2000, Final)*

A database has five elements, X, Y, Z, U, V. The two questions below ask us to recover the database after a system crash.

(a) Assuming we maintain an *undo* log whose content, after the crash, is:

```
<start T1>
<T1,X,3>
<start T2>
<T2,Y,1>
<start ckpt(T1,T2)>
<T1,X,4>
<start T3>
<commit T1>
<T2,Y,2>
<T3,Z,3>
<commit T2>
<end ckpt>
<start T4>
<T4,U,8>
<start ckpt(T3,T4)>
<start T5>
<T4,U,9>
<T5,X,10>
<commit T5>
<start T6>
<T6,Y,12>
```

Recover the database, assuming that after the crash the five elements have the values $X = 1$, $Y = 2$, $Z = 3$, $U = 4$, $V = 5$. Indicate which portion of the log you needed to inspect, and which transactions have to be executed again.

Solution: Scanning the log from the end to the beginning, the uncommitted transactions are `T6`, `T4`, `T3`; these need to be run again by the application. The committed transactions are `T5`, `T1`, `T2`.

For recovery we scan the log from the bottom up to `<start ckpt(T1,T2)>` and undo the updates of `T6`, `T4`, `T3`:

$$\begin{aligned} X &= 1 \\ Y &= 1, 12 \\ Z &= 3, 5 \\ U &= 4, 9, 8 \\ V &= 5 \end{aligned}$$

(b) Assuming we maintain a *redo* log whose content, after the crash, is:

```
<start T1>
<T1,X,2>
<start T2>
<T2,Y,1>
<start T3>
<T3,Z,5>
<T2,Y,2>
<commit T2>
<start ckpt(T1,T3)>
<T1,X,4>
<start T4>
<end ckpt>
<commit T1>
<T3,U,2>
<T4,U,8>
<start ckpt(T3,T4)>
<start T5>
<T4,U,9>
<T5,X,10>
<commit T4>
<start T6>
<T6,Y,12>
```

Recover the database, assuming that after the crash the five elements have the values $X = 1$, $Y = 2$, $Z = 3$, $U = 4$, $V = 5$. Indicate which portion of the log you needed to inspect, and which transactions have to be executed again.

Solution: Scanning the log from the end to the beginning, the uncommitted transactions are `T6`, `T5`, `T3`; these need to be run again by the application. The committed transactions are `T4`, `T1`, `T2`.

The last successful checkpoint was `<start ckpt(T1,T3)>`, so we need to start at the earlier of `<start T1>` and `<start T3>` — hence from the beginning of the log. Redo the updates of `T4`, `T1`, `T2`:

$$\begin{aligned} X &= 1, 4 \\ Y &= 2 \\ Z &= 3 \\ U &= 4, 8, 9 \\ V &= 5 \end{aligned}$$

167. *(from CSE 444, Autumn 2001, Final)*

After a system crash, the undo log using non-quiescent checkpointing contains the following data:

```
<START T1>
<T1, X1, 1>
<START CKPT ????>
<START T2>
<T2, X2, 2>
<T1, X1, 3>
<START T3>
<END T1>
<END CKPT>
<START CKPT ????>
<T2, X2, 4>
<T3, X3, 5>
<START T4>
<END T2>
<T4, X4, 6>
<END T3>
<END CKPT>
<START T5>
<T5, X5, 7>
<START CKPT ????>
<T4, X4, 8>
CRASH !!!
```

(a) What are the correct values of the three `<START CKPT ????>` records? You have to provide three correct values for the three `????`s.

Solution:

```
<START CKPT T1>
<START CKPT T2,T3>
<START CKPT T4,T5>
```

(b) Assuming that the three `<START CKPT ???>` records are correctly stored in the log, according to your answer at (a), show which elements are recovered by the undo recovery manager and compute their values after recovery.

Solution:

$X4 =$ 8, 6 (final value: 6)
$X5 =$ 7

(c) Indicate what fragment of the log the recovery manager needs to read.

> **Solution:** Up to the second `START CKPT`, or up to `<START T4>` — both answers are correct.

168. *(from CSE 444, Autumn 2002, Final)*

The SuperSQL database system stores its undo log file in a table, with the following schema:

`Log(N, T, E, V)`

where `N` is the entry number $(0, 1, 2, 3, \ldots)$, `T` is the transaction id, `E` is the element id, and `V` is the old value. A log entry of the form `<T, E, V>` is represented by the tuple `(N, T, E, V)`; here `E` > 0. The log entries `<START T>`, `<COMMIT T>`, and `<ABORT T>` are represented by a tuple `(N, T, E, null)`, where `E` $= -1$ for `START`, `E` $= -2$ for `COMMIT`, and `E` $= -3$ for `ABORT`. For example, the log

```
<START T1>
<T1 X1 55>
<START T2>
<T2 X2 99>
<COMMIT T1>
. . . .
```

is represented by the table

N	T	E	V
0	T1	−1	
1	T1	X1	55
2	T2	−1	
3	T2	X2	99
4	T1	−2	
...			

Recall that each transaction starts and ends at most once, i.e. a sequence `<START T>` ... `<COMMIT T>` ... `<START T>` will not occur in the log. Moreover, any update by the transaction `T` will occur between its `<START T>` and `<COMMIT T>`, or between `<START T>` and `<ABORT T>` respectively.

(a) Write a SQL query that can be run during database recovery, after a system crash. The answer to your query should return a table with two attributes, `E` and `V`, indicating which elements have to be updated with what values. You should include each element `E` at most once in your answer; otherwise it would be ambiguous how to update it.

Solution: We need to return a pair (E, V) from an entry (N, T, E, V) if (a) there exists an START entry for T (b) there is no COMMIT or ABORT entry for T and (c) there is no entry (N', T, E, V') with $N' < N$ (because in that case we need to restore the value V' and not V).

```
select distinct x.e, x.v
from   Log x, Log y
where  x.t = y.t and x.e > 0
   and y.e = -1                  -- y is <START T>
   and not exists (select *
                   from Log z
                   where x.t = z.t   -- same transaction
                     and z.e < -1)   -- z is <COMMIT T> or <ABORT T>
   and not exists (select *
                   from Log u
                   where x.t = u.t   -- same transaction
                     and x.e = u.e   -- same element E
                     and u.n < x.n)  -- earlier time
```

169. *(from CSE 444, Winter 2005, Final; CSE 544, 2006, Final)*

(a) **Recovery.** Consider a transaction `T` consisting of the following actions:

`START, R(A), R(B), W(A), W(B), COMMIT`

This generates the following log entries:

```
<START T>
<T, A, 10>
<T, B, 20>
<COMMIT T>
```

We examine several sequences of write operations to disk. Each line represents one disk write, e.g. `<T, A, 10>` represents the operation that writes this log entry to disk, while `OUTPUT(A)` represents writing the `A` element to disk.

```
S1:  <START T> |S2:  <START T> |S3:  <START T> |S4:  <START T>
     OUTPUT(A) |     <T, A, 10>|     <T, A, 10>|     <T, A, 10>
     OUTPUT(B) |     <T, B, 20>|     <T, B, 20>|     OUTPUT(A)
     <T, A, 10>|     OUTPUT(B) |     <COMMIT T>|     <T, B, 20>
     <T, B, 20>|     OUTPUT(A) |     OUTPUT(A) |     OUTPUT(B)
     <COMMIT T>|     <COMMIT T>|     OUTPUT(B) |     <COMMIT T>
```

i. Assume that the transaction manager uses an *undo* log. Which of the above sequences is legal, according to the rules of the undo log? You have to turn in a list of sequence names, e.g. `S1, S3, S4`.

Solution: The UNDO log entry like `<T,A,10>` must always precede the OUTPUT entry AND must come before COMMIT. Therefore S2, S4 are legal, while S1, S3 are illegal.

ii. Assume that the transaction manager uses a *redo* log. Which of the above sequences is legal, according to the rules of the redo log?

Solution: The REDO log entry must always precede the OUTPUT, as before, but now it must come after COMMIT. Therefore S3 is legal, while S1, S2, S4 are illegal.

iii. Assume that the transaction manager uses an *undo/redo* log. (Assume that the entries `<T, A, 10>` and `<T, B, 20>` are replaced with `<T, A, 1, 10>` and `<T, B, 2, 20>`.) Which of the above sequences is legal, according to the rules of the undo/redo log?

Solution: Each log entry must still precede the output, hence S1 is illegal. Otherwise they may come before or after COMMIT, hence S2, S3, S4 are legal.

(b) Consider the following schedule, where `S1` means `START(T1)`, and `C2` means `COMMIT(T2)`:
`S1, S2, R1(A), R2(B), W1(B), W2(C), C1, C2`

i. Draw the precedence graph, and indicate whether the schedule is serializable. For this question you will ignore the `START` and `COMMIT` actions, and consider only the `R` and `W` actions.

Solution: TO DRAW A GRAPH

ii. Assume a lock-based concurrency control manager, which automatically introduces a lock action right before each read or write, and introduces all unlock actions before commit. Indicate up to which point the schedule can execute, by drawing a line after the last action that can execute. Then indicate in which order the remaining actions are scheduled by the lock-based concurrency control manager. You have to turn in a sequence of actions, e.g. `COMMIT(T2)`, `W2(C)`.

Solution:
`S1, S2, R1(A), R2(B), | W1(B), W2(C), C1, C2`
When T1 asks to write B it will be blocked.

iii. Assume a concurrency control manager based on timestamps. Indicate up to which point the schedule can execute, by drawing a line after the last action that can execute. Then indicate in which order the remaining actions are scheduled by the concurrency control manager based on timestamps.

Solution: All actions are possible: $W1(B)$ will be ignored by using Thomas' rule.
`S1, S2, R1(A), R2(B), [--W1(B),--] W2(C), C1, C2`

170. *(from CSE 444, Winter 2006, Final)*

Consider the two logs below, called `LogA` and `LogB`:

`LogA`

1	<START T1>
2	<T1, A, a>
3	<T1, B, b>
4	<START T2>
5	<T2, C, c>
6	<START T3>
7	<T3, D, d>
8	<COMMIT T1>
9	<START CKPT(T2,T3)>
10	<T2, E, e>
11	<START T4>
12	<T4, F, f>
13	<T3, G, g>
14	<COMMIT T3>
15	<END CKPT>
16	<COMMIT T2>
17	<COMMIT T4>

`LogB`

1	<START T1>
2	<T1, A, a>
3	<T1, B, b>
4	<START T2>
5	<T2, C, c>
6	<START T3>
7	<T3, D, d>
8	<COMMIT T1>
9	<START CKPT(T2,T3)>
10	<T2, E, e>
11	<START T4>
12	<T4, F, f>
13	<T3, G, g>
14	<COMMIT T3>
15	<COMMIT T2>
16	<END CKPT>
17	<COMMIT T4>

(a) One of the logs is an UNDO log, the other is a REDO log. Indicate which one is the UNDO log and which is the REDO log. (Note: you must answer this question correctly in order to be able to answer the other questions.)

Solution: `LogB` is the UNDO log, because `<END CHKPT>` is placed right after both transactions `T2,T3` have committed. In contrast, `LogA` has the `<END CHKPT>` entry before `T2` commits, which is inconsistent with a UNDO log.

(b) Consider the UNDO log. For each of the elements $A, B, \ldots, G$, indicate when the system might have issued the corresponding `OUTPUT` statement. For each you will answer with an interval, e.g. "`<OUTPUT B>` may be issued between time steps 7 and 14". If you want to indicate that the `OUTPUT` may happen after the end of the log, write either "7, 18" or "7, ∞".

Solution: This refers to `LogB`. The OUTPUT may be issued at any time AFTER the log entry and BEFORE `<COMMIT>`. For example, `<OUTPUT A>` is between 2-8 etc.

(c) Repeat the same question for the REDO log.

Solution: This refers to `LogA`. The output may be issued at any time AFTER `<COMMIT>`: for T1 both `<OUTPUT A>` and `<OUTPUT B>` are guaranteed to be issued before `<END CKPT>` (thus between $8 - 15$), while for T2, T3, T4 they may have not been issued at all.

(d) Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where st_i means that transaction T_i starts and co_i means T_i commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. In each case below tell what happens with the last write request. You have to choose one of: (1) the request is accepted, (2) the request is ignored, (3) the transaction is delayed, (4) the transaction is rolled back.

i. $st_1; st_2; st_3; r_1(A); r_2(B); r_2(C); r_3(B); co_3; w_2(B); w_1(C)$
What happens for $w_1(C)$?

Solution: Delayed: waiting for T2 to either commit (then abort T1) or abort (the proceed with $w_1(C)$).

ii. $st_1; st_2; r_1(A); r_2(B); w_2(A); co_2; w_1(B)$
What happens for $w_1(B)$?

Solution: Proceed

iii. $st_1; st_3; st_2; r_1(A); r_2(B); r_3(B); w_3(A); w_2(B); co_3; w_1(A)$
What happens for $w_1(A)$?

Solution: Abort.

iv. $st_1; r_1(A); st_2; w_2(B); r_2(A); w_1(B)$
What happens for $w_1(B)$?

Solution: Ignore

171. *(from CSE 444, Autumn 2004, Final; CSE 444, Autumn 2005, Final; CSE 444, Autumn 2006, Final)*

(a) The recovery manager uses *undo* logging. In each of the logs below there is a `<START CKPT(...)>` entry, but the matching `<END CKPT>` is missing. That is, the checkpoint ended, but the corresponding log entry was somehow omitted. Please indicate where the `<END CKPT>` entry should go in the log file. In the case where there may be several places, indicate the earliest possible place where the entry `<END CKPT>` can be. In addition, please indicate how far back the recovery manager has to read in the log, if the crash occurs after the last entry in the log.

```
<START S>
<S,A,60>
<START CKPT(S)>
<COMMIT S>
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>
```

```
<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START CKPT(T)>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>
```

Solution: Undo logging: the `<END CKPT>` may be placed immediately after `<COMMIT S>` in the first log, and immediately after `<START CKPT(T)>` . . . `<COMMIT T>` in the second, i.e. once every transaction active at the start of the checkpoint has committed.

(b) Repeat the same question above, assuming that the recovery manager uses *redo* logging.

```
<START S>
<S,A,60>
<START CKPT(S)>
<COMMIT S>
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>
```

```
<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START CKPT(T)>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>
```

Solution: Redo logging: the `<END CKPT>` may be placed immediately after `<START CKPT(S)>` in the first log, and immediately after `<START CKPT(T)>` in the second, i.e. once the dirty pages of the committed transactions have been flushed.

(c) Consider a concurrency control manager by timestamps. Below are several sequences of events, including *start* events, where st_i means that transaction T_i starts. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. In each case below tell what happens with the last *write* request. You have to choose between one of the following four possible answers: (1) the request is accepted, (2) is ignored, (3) the transaction is delayed, (4) the transaction is rolled back.

i. $st_1; st_2; r_1(A); r_2(B); w_2(A); w_1(B)$

Solution: (4) rolled back

ii. $st_1; r_1(A); st_2; w_2(B); r_2(A); w_1(B)$

Solution: (3) delayed

iii. $st_1; st_2; st_3; r_1(A); r_2(B); w_1(C); r_3(B); w_2(B); w_3(A)$

Solution: (1) accepted

iv. $st_1; st_3; st_2; r_1(A); r_2(B); r_3(B); w_2(A); w_2(B); w_3(A)$

Solution: (3) delayed

172. *(from CSE 444, Autumn 2008, Final)*

The scheduler uses shared locks for reads and exclusive locks for writes and follows the strict 2PL policy: locks are acquired on a per need basis, and released at commit time. There are three transactions wishing to execute the following:

```
T1:  ST1;  R1(A), R1(B), W1(A), W1(B), CO1.
T2:  ST2;  R2(B), R2(C), W2(B), W2(C), CO2.
T3:  ST3;  R3(C), R3(A), W3(C), W3(A), CO3.
```

(a) Consider the partial schedules below. For each schedule indicate whether (a) it will end in a deadlock no matter how the scheduler continues to schedule the transactions — in this case show a possible continuation of the schedule until it ends in a deadlock; (b) it may be completed successfully, but may also result in a deadlock — in this case show both a successful continuation of the schedule and one that ends in a deadlock; (c) it will complete successfully no matter what the scheduler does — in this case show a successful completion of the schedule.

```
Schedule 1
ST1, ST2, ST3, R1(A), R2(B), R3(C)

Schedule 2:
ST1, ST2, ST3, R2(B), R3(C), R3(A), W3(C)

Schedule 3:
ST1, ST2, ST3, R1(A), R2(B), R2(C), W2(B), W2(C)
```

Solution: Schedule 1: always deadlocks.
Schedule 2: may or may not deadlock.
Schedule 3: may or may not deadlock.

(b) Suppose now that the scheduler uses only exclusive locks, both for reads and writes. As an application writer you are asked to rewrite transaction `T3` to eliminate the possibility of deadlocks. Make a minimal change in the order of the operations in `T3` such that a deadlock can never occur.

Solution:

```
T3:  R3(A), R3(C), W3(C), W3(A)
```

173. *(from CSE 444, Autumn 2008, Midterm)*

(a) After a systems failure, the undo-redo recovery log has the following entries:

```
<START T1>
<T1 A 1 2>
<START T2>
<COMMIT T1>
<START T3>
<T3 A 2 3>
<START T4>
<CKPT(T2,T3,T4)>
<T2 B 10 20>
<COMMIT T2>
<START T5>
<T5 D 1000 2000>
<T4 C 100 200>
<COMMIT T5>
<START T6>
<END CKPT>
<T6 D 2000 3000>
```

An entry `<T, X, u, v>` means that transaction `T` has updated the value of `X` from `u` (the old value) to `v` (the new value). `<CKPT(...)>` denotes the beginning of a checkpoint and lists the currently active transactions. `<END CKPT>` is written to disk once all dirty pages of the active transactions have been flushed to disk. The redo phase precedes the undo phase during the recovery.

i. Which are the transactions whose actions the recovery manager needs to redo?

Solution: `T2`, `T5`

ii. Which are the transactions whose actions the recovery manager needs to undo?

Solution: `T3`, `T4`, `T6`

iii. Indicate the actions of the recovery manager on all the elements, separately during the Redo and the Undo phase.

Solution:

	Redo	Undo
$A =$		2
$B =$	20	
$C =$		100
$D =$	2000	2000
$E =$		

(b) Indicate for each statement below if it is true or false. You do not have to justify your answer.

i. Every schedule that is possible under a timestamp based concurrency control scheduler is also possible under a multiversion concurrency control scheduler. Yes or No?

Solution: Yes

ii. Every schedule that is possible under a multiversion concurrency control scheduler is also possible under a timestamp based concurrency control scheduler. Yes or No?

Solution: No

iii. Suppose that every transaction T_i has the form ST_i, $RD_i(X)$, $WT_i(Y)$, for some elements X and Y, followed by CO_i. Then a 2-phase locking scheduler that uses shared locks for read operations and exclusive locks for write operations will never result in a deadlock. Yes or No?

Solution: No

iv. Suppose that all transactions are either read-only (i.e. they have the form $ST_i, RD_i(X), RD_i(Y), \ldots, CO_i$), or they consist of a single write (i.e. they have the form $ST_i, WT_i(X), CO_i$). Then a 2-phase locking scheduler that uses shared locks for read operations and exclusive locks for write operations will never result in a deadlock. Yes or No?

Solution: Yes

174. *(from CSE 544p, Spring 2009, Final)*

(a) Consider the four schedules below, where s_i means *transaction i starts*, and c_i means *transaction i commits*.

$$s_1, s_2, s_3, w_1(A), w_1(C), r_2(A), w_2(B), c_2, r_3(B), r_3(C), c_1, c_3 \quad (1)$$
$$s_1, s_2, s_3, w_1(A), r_1(C), r_2(A), w_2(B), r_3(B), r_3(C), c_1, c_2, c_3 \quad (2)$$
$$s_1, s_2, s_3, w_1(A), r_2(A), r_3(C), r_3(D), r_2(B), w_3(B), r_2(C), w_1(D), c_1, c_2, c_3 \quad (3)$$
$$s_1, s_2, s_3, w_1(A), w_1(C), c_1, r_2(A), r_3(C), w_2(B), c_2, r_3(B), c_3 \quad (4)$$

For each of the schedule indicate which of the following applies:

1. The schedule is not conflict serializable: in this case indicate a cycle in the serialization graph.

2. The schedule is conflict serializable but non-recoverable: in this case indicate a violation of the recoverability condition. Your answer should be something like this: "if transaction 4 were to abort instead of committing, then we need to abort transaction 7 because [explain why] and we cannot do this because [explain why]" [you need to replace transactions 7, 4 with transactions in the schedules above].
3. The schedule is conflict serializable, recoverable, but does not avoid cascading aborts; in this case give an example of a cascading abort. Your answer should be something like this: "if transaction 4 where to abort instead of committing, then we need to abort transaction 7 because [explain why], and as a consequence we also need to abort transaction 5 because [explain why]".
4. The schedule is conflict serializable and avoids cascading aborts.

Solution: (3) not conflict serializable: $1 \stackrel{A}{\rightarrow} 2 \stackrel{B}{\rightarrow} 3 \stackrel{D}{\rightarrow} 1$.

(1) conflict serializable, non-recoverable: if transaction 1 aborts instead of committing (c_1 becomes a_1), then we need to undo transaction 2, because of $w_1(A), r_2(A)$, which is not possible because 2 has already committed.

(2) conflict serializable, but does not avoid cascading aborts. If 1 aborts (c_1 becomes a_1) then we need to abort 2, because of $w_1(A), r_2(A)$. But then we also need to abort 3, because of $w_2(B), r_3(B)$.

(4) conflict serializable, and avoids cascading aborts.

(b) Indicate for each of the concurrency control managers below whether it is guaranteed to produce a schedule that is (a) conflict serializable but not necessarily recoverable, (b) conflict serializable and recoverable but does not necessarily avoid cascading aborts, (c) conflict-serializable and avoids cascading aborts

1. 2PL.
2. Strict 2PL.
3. Timestamp based concurrency control (with the commit bit).
4. Validation-based concurrency control.

Solution:

1. 2PL does not guarantee a recoverable schedule.
2. Strict 2PL guarantees a schedule that avoids cascading aborts.
3. Timestamp based concurrency control guarantees a schedule that avoids cascading aborts.
4. Validation-based concurrency control guarantees a schedule that avoids cascading aborts.

(c) Consider the ARIES recovery algorithm. Answer the following questions:

1. Indicate if the following statement is true of false. "At the end of the analysis phase, the Dirty Page Table contains the exact list of all pages dirty at the moment of the crash."

   Solution: FALSE: some pages may have been modified in the buffer pool but the corresponding log entries were not flushed to disk yet. Conversely, some dirty pages (with log entries flushed to disk) may have been written to disk, hence they become clean, but this is not reflected in the log and at the end of the analysis phase they look dirty.

2. During the UNDO phase of the recovery, the ARIES system writes CLR records in the log. Suppose the system crashes during the recovery. At restart, how are the CLR records used ? Check one of the answers below:
   - They are used during the ANALYSIS phase.
   - They are used during the REDO phase.
   - They are used during the UNDO phase.
   - They are ignored.

Solution: They are used during the REDO phase.

175. *(from CSE 444, Spring 2010, Final)*

Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where sti means that transaction Ti starts and coi means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. In each case below, say what happens with the last request.

You have to choose between one of the following four possible answers:

1. the request is accepted,
2. the request is ignored,
3. the transaction is delayed,
4. the transaction is rolled back.

(a) st1; st2; st3; r1(A); w1(A); r2(A);
The system will perform the following action for r2(A): _____

Solution: The system will perform the following action for r2(A): delayed.

(b) st1; st2; r2(A); co2; r1(A); w1(A)
The system will perform the following action for w1(A): _____

Solution: The system will perform the following action for w1(A): rolled back.

(c) st1; st2; st3; r1(A); w2(A); w3(A); r2(A);
The system will perform the following action for r2(A): _____

Solution: The system will perform the following action for r2(A): rolled back.

(d) st1; st2; r1(A); r2(A); w1(B); w2(B);
The system will perform the following action for w2(B): _____

Solution: The system will perform the following action for w2(B): accepted.

(e) st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)
The system will perform the following action for w2(A): _____

Solution: The system will perform the following action for w2(A): ignored.

176. *(from CSE 444, Spring 2010, Makeup Midterm; CSE 444, Spring 2010, Midterm)*

(a) For each of the following statements indicate whether it is true or false:

- The main reason why transactions were invented was to improve the database performance. True or false ?

  Solution: FALSE

- A banking application that manages customer accounts should perform all updates to the accounts from within transactions with ACID properties. True or false ?

  Solution: TRUE

- A Facebook application that allows you to write on your Wall (so that other friends can see what you are up to) should update the wall content in the database from within a transaction with ACID properties. True or false ?

Solution: FALSE

(b) Consider the content of the following **undo log**:

LSN1	<START T1>
LSN2	<T1 X 5>
LSN3	<START T2>
LSN4	<T1 Y 7>
LSN5	<T2 X 9>
LSN6	<START T3>
LSN7	<T3 Z 11>
LSN8	<COMMIT T1>
LSN9	<START CKPT(T2,T3)>
LSN10	<T2 X 13>
LSN11	<T3 Y 15>
	*C*R*A*S*H*

- Show how far back in the recovery manager needs to read the log. Write below the earliest LSN that the recovery manager reads.

Solution: LSN3

- Show below the actions of the recovery manager during recovery:

Solution:

$$\begin{aligned} Y &= 15 \\ X &= 13 \\ Z &= 11 \\ X &= 9 \end{aligned}$$

- What is the value of X at the end of the recovery ?

Solution: 9

(c) Consider a **redo log**. For each of the questions below indicate whether it is true of false. You do not need to justify your answer.

- When a transaction wants to commit, it first needs to wait until all its pages have been written to disk. True or false ?

Solution: false

- If the last checkpoint in the log has not been completed, then it cannot be used during recovery. True or false ?

  Solution: true

- If the system crashes while a transaction T is active, then none of the records written by T have been written to disk. True or false ?

  Solution: true

- If transactions $T1, T2, T3$ are active when a checkpoint starts, then the checkpoint cannot end before all three transactions commit. True or false ?

  Solution: false

177. *(from CSE 544p, Autumn 2010, Final)*

(a) Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where sti means that transaction Ti starts and coi means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. In each case below, say what happens with the last request.

You have to choose between one of the following four possible answers:

1. the request is accepted,
2. the request is ignored,
3. the transaction is delayed,
4. the transaction is rolled back.

1. st1; st2; st3; r1(A); w1(A); r2(A);
   What will the system do in response to the request r2(A) ?

   Solution: r2(A): T2 is delayed.

2. st1; st2; r2(A); co2; r1(A); w1(A);
   What will the system do in response to the request w1(A) ?

   Solution: w1(A): T1 is rolled back.

3. st1; st2; st3; r1(A); w3(A); co3; r2(A);
   What will the system do in response to the request r2(A) ?

   Solution: r2(A): T2 is rolled back.

4. st1; st2; r1(A); r2(A); w1(B); w2(B);
   What will the system do in response to the request w2(B) ?

   Solution: w2(B): accepted.

5. st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)
   What will the system do in response to the request w2(A) ?

   Solution: w2(A): ignored.

(b) Consider a social network stored in a database with the following schema:

```
Person(pid, name, mood)
Friends(pid1, pid2)
```

The database system is running a concurrency control manager based on snapshot isolation (SI). One of the applications is called *Set my mood as that of my friends*, and is implemented by the following transaction:

```
Set_mood(%P)
  START TRANSACTION
    ch = select count(*)
         from Person x, Friends y, Person z
         where x.pid = %P
           and x.pid = y.pid1
           and y.pid2 = z.pid
           and z.mood = 'HAPPY'

    cs = select count(*)
         from Person x, Friends y, Person z
         where x.pid = %P
           and x.pid = y.pid1
           and y.pid2 = z.pid
           and z.mood = 'SAD'

    if ch > cs
      then update Person
           set mood = 'HAPPY'
           where pid = %P
      else update Person
           set mood = 'SAD'
           where pid = %P
  END TRANSACTION
```

Assume that the database is static, i.e. no records are inserted or deleted. If multiple instances of this transaction run simultaneously, then the SI scheduler will not guarantee serializability because of the write skew. Your task is to rewrite the application above, in order to ensure that the schedule is serializable. Your new rewritten application must satisfy the following:

- If a single instance is run on the database, then it has exactly the same semantics

as the application above.

- If multiple instances are run on the database, then, under SI, the schedule will be guaranteed serializable.

Solution:

```
Set_mood(%P)
  START TRANSACTION
    /* . . . same as above . . */
    /* then add the following dummy update */
    update Person
    set mood = mood
    where pid in
          select z.pid
          from Person x, Friends y, Person z
          where x.pid = %P
           and  x.pid = y.pid1
           and  y.pid2 = z.pid
   END TRANSACTION
```

The only way to create conflicts is to do an extra write. Credit was given only for the "set mood" statement.

(c) For each of the statements below indicate whether it is true of false about the ARIES recovery manager:

1. During the normal operation of a database (i.e. not during recovery) no update records in the log are undone.
   true or false ?

   Solution: false

2. During the normal operation of a database (i.e. not during recovery) no update records in the log are redone.
   true or false ?

   Solution: true

3. During the normal operation of a database (i.e. not during recovery) no CLR records are ever written to the log.
   true or false ?

   Solution: false

4. During recovery from a crash no update record is processed both during the redo and during the undo phase. (In other words, an update record is either redone or undone, but not both.)
   true or false ?

Solution: true

5. During recovery from a crash the CLR records in the log are neither undone nor redone. (In other words the CLR records server a different purpose, not to be redone or undone during recovery.)
   true or false ?

   Solution: false

6. Suppose that two update log entries were written in the log on behalf of a transaction, then the system crashed while the transaction was active. Even if the system crashes repeatedly during recovery, there will never be more than two CLR records written on behalf of this transaction.
   true or false ?

   Solution: true

7. All log entries preceding the last `begin_checkpoint` can be safely deleted from the log in order to reclaim their disc space.
   true or false ?

   Solution: false

178. *(from CSE 344, Spring 2011, Final)*

Consider a database consisting of a single relation R:

R:

A	B
1	10
2	20

Two transactions run concurrently on this database, resulting in the following schedule:

Line	T1	T2
1	begin;	
2		begin;
3	update R set B = (select sum(B) from R) where A=1;	
4		update R set B = (select sum(B) from R) where A=2;
5	select * from R;	
6		select * from R;
7	insert into r values (3,300);	
8		insert into r values (4,400);
9	select * from R;	
10		select * from R;
11	update R set B = (select sum(B) from R) where A=1;	
12		update R set B = (select sum(B) from R) where A=2;
13	select * from R;	
14		select * from R;
15	commit;	
16		commit;

(a) Is this schedule possible in SQL Lite ? If not, then indicate the first line where SQL Lite will change the schedule.

Yes ? Or No (and indicate line number) ?

Solution: No: line 4

(b) Is this schedule possible in Postgres ? If not, then indicate the first line where Postgres will change the schedule.

Yes ? Or No (and indicate line number) ?

Solution: Yes

(c) Consider running these two transactions in Postgres, using isolation level SERIALIZABLE. Assuming postgres executes exactly the schedule above, indicate the result of each of the six `select *` statements, as well as the content of the table

after both transactions commit.

Line Number	Result of `select * from r;`
5	
6	
9	
10	
13	
14	
after both commit	

Solution:

Line Number	Result of `select * from r;`
5	(1,30), (2,20)
6	(1,10), (2,30),
9	(1,30), (2,20), (3,300)
10	(1,30), (2,20), (4,400)
13	(1,350), (2,20), (3,300)
14	(1,10), (2,440), (4,400)
after both commits	(1,350), (2,440), (3,300), (4,400)

(d) Is the schedule above serializable ?
Yes or No ?

Solution: NO

179. *(from CSE 344, Winter 2012, Final)*

(a) Consider a database consisting of a single relation R:

R:

A	B
1	10
2	20

Three transactions run concurrently on this database, issuing commands at the following time stamps:

Time	T1	T2	T3
1	begin transaction;		
2	select * from R;		
3		begin transaction;	
4		select * from R where $A = 2$;	
5	update R set $B = 30$ where $A = 2$;		
6		select * from R where $A = 2$;	
7	commit;		
8			begin transaction;
9			select * from R where $A = 2$;
10		commit;	
11			
12			
13			commit;

Transaction T1 first reads the data, then updates the second record ($A = 2$). It attempts to commit at time stamp 7.

Transaction T2 attempts to read the second record ($A = 2$) at time stamp 4.

Transaction T3 attempts to read the second record ($A = 2$) at time stamp 9.

i. Find out what happened. You will consider two scenarios: when the database is managed with SQL Lite, and when the database is managed with SQL Server; in both cases the isolation level is set to SERIALIZABLE. For each command issued by a transaction, indicate one of the following outcome:

SUCCESS The request returns immediately, with success. In this case you should write SUCCESS, and also write the value that the transaction has read, if applicable.

ERROR The request returns immediately, with error. In this case you should write ERROR and indicate at which later time stamp the transaction needs to retry that command; if the command was a read, write down the value read by the transaction, when the command is reissued successfully.

WAIT The request causes the transaction to be placed on wait. In that case

you should write WAIT, and also write at which later time stamp the transaction will be allowed to resume; if the command was a read, write down the value read by the transaction, when the command is resumed.

For example, you may answer as follows (not a real answer):

Time	T1	T2	T3
1	begin transaction;		
2	select * from R; – WAIT UNTIL $T = 4$		
3		begin transaction; – SUCCESS	
4	 – RESUMED: values$10, 30$	select * from R where $A = 2$; – SUCCES: value= 30	
	...	...	...

This schedule is excecuted on **SQL Lite**			
	T1	T2	T3
1	begin transaction;		
2	select * from R;		
3		begin transaction;	
4		select * from R where $A = 2$;	
5	update R set $B = 30$ where $A = 2$;		
6		select * from R where $A = 2$;	
7	commit;		
8			begin transaction;
9			select * from R where $A = 2$;
10		commit;	
11			
12			
13			commit;

Solution:

Time	T1	T2	T3
1	begin transaction;		
2	select * from R; – SUCCESS: values 10,20		
3		begin transaction;	
4		select * from R where $A = 2$; – SUCCES: value 20	
5	update R set $B = 30$ where $A = 2$; – SUCCESS		
6		select * from R where $A = 2$; – SUCCES: value 20	
7	commit; ERROR: retry on 11		
8			begin transaction; SUCCESS
9			select * from R where $A = 2$; ERROR: retry on 12
10		commit; – SUCCES:	
11	– REISSUE: commit; – SUCCESS;		
12			– REISSUE: select * from R where $A = 2$; – SUCCESS: value 30
13			commit; – SUCCESS

This schedule is excecuted on **SQL Server**			
	T1	T2	T3
1	begin transaction;		
2	select * from R;		
3		begin transaction;	
4		select * from R where $A = 2$;	
5	update R set $B = 30$ where $A = 2$;		
6		select * from R where $A = 2$;	
7	commit;		
8			begin transaction;
9			select * from R where $A = 2$;
10		commit;	
11			
12			
13			commit;

Solution:

Time	T1	T2	T3
1	begin transaction;		
2	select * from R; – SUCCESS: values 10,20		
3		begin transaction;	
4		select * from R where $A = 2$; – SUCCES: value 20	
5	update R set $B = 30$ where $A = 2$; – SUCCESS		
6		select * from R where $A = 2$; – WAIT: until 7	
7	commit; – SUCCESS	 – SUCCCESS: value 30	
8			begin transaction; – SUCCESS
9			select * from R where $A = 2$; – SUCCESS: value 30
10		commit; – SUCCES:	
11			
12			
13			commit; – SUCCESS

ii. What is the serialization order of the three transactions on SQL Lite? Give an order or the transactions T1, T2, T3:

Solution: T2, T1, T3

iii. What is the serialization order of the three transactions on SQL Server? Give an order or the transactions T1, T2, T3:

Solution: T1, T2, T3

(b) Recall that a *shared lock* (or *read lock*) may be held by several transactions, while an *exclusive lock* (or *write lock*) can be held by only one transaction.

- System A uses shared and exclusive locks.
- System B uses only exclusive locks.

Indicate which statements below are true.

i. System A may be able to execute more transactions per second than system B. True or false?

Solution: true

ii. If all transactions are read-only, then on system A no transaction ever waits. True or false?

Solution: true

iii. If all transactions are write-only, then on system B no transaction ever waits. True or false?

Solution: false

iv. There exists schedules that result in a deadlock on system A but not in a deadlock on system B. True or false?

Solution: true

Solution: True. For example the following schedule: Start1, Start2, Read1(X), Read2(X), Write1(X), Write2(X), Commit1, Commit2. On system A, both transactions start with a read lock on X, then attempt to escalate it to a write lock, and result in deadlock. On system B, the second transaction is placed on wait during the operation Read2(X).

(c) A *static database* is a database where no insertions or deletions are performed. A *dynamic database* is a database that allows insertions and deletions. We consider a scheduler that has one lock for each record in the database (like SQL Server). Answer the questions below:

i. In a static database, strict two phase locking guarantees that the schedule is serializable while two phase locking does not. True or false

Solution: false

ii. Strict two phase locking guarantees that the schedule is recoverable, while two phase locking does not. True or false

Solution: true

iii. In a dynamic database, strict two phase locking can prevent phantoms, while two phase locking cannot. True or false

Solution: false: needs table lock

iv. Strict two phase locking is more difficult to implement and most database system do not support it. True or false

Solution: false

v. Strict two phase locking holds all the locks until the end of a transaction, while two phase locking may release the locks earlier. True or false

Solution: true

Solution: false

vi. In both two phase locking and strict two phase locking all locks must precede all unlocks. True or false

Solution: true

vii. In strict two phase locking deadlocks are not possible, while in two phase locking deadlocks are possible. True or false

Solution: false

viii. If the database uses shared locks for read operations, then if all transactions are read-only then no deadlocks are possible. True or false

Solution: true

ix. SQL Server checks for deadlocks at regular intervals, and if it detects a deadlock then aborts a transaction True or false

Solution: true

x. Suppose that the table R has 1000 records. Then the transaction below needs to acquire 1000 locks:

```
begin transaction;
select * from R;
commit;
```

True or false

Solution: true

180. *(from CSE 344, Winter 2013, Final)*

(a) Consider the following schedule:

T1	T2
BEGIN	
READ(A)	
$A = A + 50$	
WRITE(A)	
	BEGIN
	READ(A)
	$A = A + 200$
	WRITE(A)
	READ(B)
	$B = 3 * B$
	WRITE(B)
	COMMIT
READ(B)	
$B = 12 * B$	
WRITE(B)	
COMMIT	

Which of the following statements are true? Check all that apply.

i. The schedule is serial. Yes or no:

Solution: No

ii. The schedule is serializable. Yes or no:

Solution: Yes

iii. The schedule is conflict serializable. Yes or no:

Solution: No

iv. This schedule is possible under SQL Lite. (Recall that SQL Lite uses a lock-based concurrency control manager, with a single lock for the entire database.) Yes or no:

Solution: No

v. This schedule is possible under SQL Server. (Recall that SQL Server uses a lock-based concurrency control manager, with one lock for each database element.) Yes or no:

Solution: No

(b) The initial value of the database element A is $A = 0$. The following four transactions

are started at approximately the same time:

T0
BEGIN
READ(A)
$X = A$
READ(A)
$Y = A$
$B = X + Y$
WRITE(B)
COMMIT

T1
BEGIN
A=1
WRITE(A)
ABORT

T2
BEGIN
A=10
WRITE(A)
COMMIT

T3
BEGIN
A=100
WRITE(A)
COMMIT

Both X and Y could be $0, 1, 10$ or 100, hence B has 10 ($= \frac{5 \cdot 4}{2}$) possible values. Indicate what values are possible for B at the end of transaction T0. *Circle* the B values that are possible, and/or *cross out* those that are impossible. (The tables show only $B = X + Y$ where $X \geq Y$, since $X + Y = Y + X$.)

i. The isolation level is set to `READ UNCOMMITTED`

$X = 0$	$X = 1$	$X = 10$	$X = 100$	$B = X + Y$
$B = 0$	$B = 1$	$B = 10$	$B = 100$	$Y = 0$
	$B = 2$	$B = 11$	$B = 101$	$Y = 1$
		$B = 20$	$B = 110$	$Y = 10$
			$B = 200$	$Y = 100$

Solution: All values are possible.

ii. The isolation level is set to `READ COMMITTED`

$X = 0$	$X = 1$	$X = 10$	$X = 100$	$B = X + Y$
$B = 0$	$B = 1$	$B = 10$	$B = 100$	$Y = 0$
	$B = 2$	$B = 11$	$B = 101$	$Y = 1$
		$B = 20$	$B = 110$	$Y = 10$
			$B = 200$	$Y = 100$

Solution: The value $A = 1$ is never read. Hence, the possible outputs are $0, 10, 100, 20, 110, 200$

iii. The isolation level is set to `REPEATABLE READ`

$X = 0$	$X = 1$	$X = 10$	$X = 100$	$B = X + Y$
$B = 0$	$B = 1$	$B = 10$	$B = 100$	$Y = 0$
	$B = 2$	$B = 11$	$B = 101$	$Y = 1$
		$B = 20$	$B = 110$	$Y = 10$
			$B = 200$	$Y = 100$

Solution: Possible values are $0, 20, 200$.

(c)

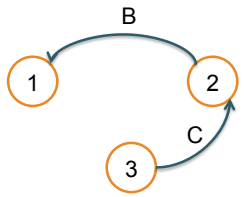
Consider the following transaction schedules. For each schedule, indicate if it is **conflict-serializable** or not. In each case show: the precedence graph; a cycle (if one exists) or an equivalent serial schedule (if one exists).

i.

```
r1(A), w1(A), r2(B), w2(B), r3(C), w3(C), r2(C), w2(C), r1(B), w1(B), c1, c2, c3
```

Answer (Draw the precedence graph and determine if it is serializable):

Solution:



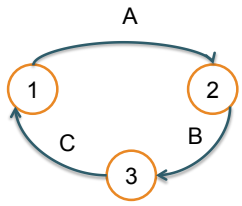
Serializable. The serialization order is: T3, T2, T1.

ii.

```
r1(A),r2(B),r3(C),w1(C),w2(A),w3(B),c1,c2,c3
```

Answer (Draw the precedence graph and determine if it is serializable):

Solution:



Not serializable, because of the cycle $T1 \to T2 \to T3 \to T1$.

iii. Give an example of a schedule with two or more transaction with the following three properties:

- T1 commits before T2 starts.
- The schedule is conflict serializable.
- In any equivalent serial schedule, T2 must come before T1 (there may be other transactions between T2 and T1).

Solution:
`s3,w3(A),s1,w1(A),c1,s2,w2(B),c2,w3(B),c3`
T1 commits before T2 starts, yet the only serialization order is T2,T3,T1.

(d) A database has 1000 elements, denote them $A_1, \ldots, A_{1000}$, and needs to run a workload of transactions of the following kind:

```
BEGIN
READ(A_i)
run a non-parallelizable, CPU-intensive computation for 0.1seconds
WRITE(A_i)
COMMIT
```

Each transaction reads one element, performs some intensive computation, then

writes the same element back. Different transactions may update the same or different elements, for example five transactions may update elements A_3, A_1, A_3, A_3, A_2.

The system is shared-memory, has 100 CPUs, and uses inter-query parallelism (multiple transactions are run in parallel, but each transaction runs on a single CPU). For serializability, the system uses one lock per element (like SQL Server).

How many transactions per second can the system execute in each case below?

i. All transactions update element A_1; no other elements are read or updated. That is, the transactions update elements in this sequence: $A_1, A_1, A_1, \ldots$
Answer Write the number of transactions per second (TPS):

> **Solution:** 10 TPS. The transactions are executed serially.

ii. The transactions update only elements from the first 50 elements. More precisely, each transaction reads/writes an element A_i chosen uniformly at random from among $A_1, \ldots, A_{50}$. For example, the transactions may update the elements in this sequence: $A_{44}, A_2, A_{13}, A_2, A_{36}, A_{11}, \ldots$
Answer Write the number of transactions per second (TPS):

> **Solution:** 500 TPS. For example, suppose the elements are updated round-robin: $A_1, A_2, \ldots, A_{50}, A_1, A_2, \ldots$ T51 can start only after T1 terminates, which is 0.1s after T1 starts. We run 50 transactions every 0.1s, or 500 TPS. We can sustain this rate because there are 100 CPUs and each can sustain 10 TPS max.

iii. The transactions update all elements. More precisely, each transaction reads/writes an element A_i, chosen uniformly at random from among $A_1, \ldots, A_{1000}$.
Answer Write the number of transactions per second (TPS):

> **Solution:** 1000 TPS, because the system becomes CPU bound. The max rate per CPU is 10 TPS and there are 100 CPUs.

181. *(from CSE 344, Autumn 2013, Final)*

(a) Answer the following questions:

i. Every serializable schedule is also conflict-serializable.
Answer Yes/No:

Solution: No

ii. Every conflict-serializable schedule is also serializable.
Answer Yes/No:

Solution: Yes

iii. SQL Lite uses optimistic concurrency control.
Answer Yes/No:

Solution: No

iv. Strict Two-Phase-Locking is guaranteed to produce a serializable schedule.
Answer Yes/No:

Solution: Yes

v. Strict Two-Phase-Locking is guaranteed to produce a conflict-serializable schedule.
Answer Yes/No:

Solution: Yes

vi. Strict Two-Phase-Locking is guaranteed to avoid deadlocks.
Answer Yes/No:

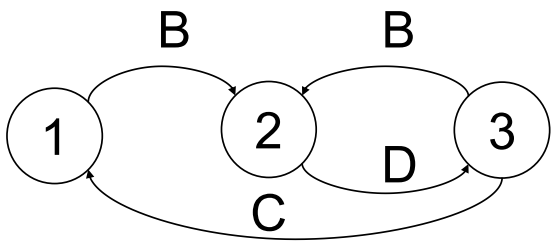
Solution: No

(b) For each of the schedules below, indicate whether they are conflict-serializable. If you answer *yes*, then give the equivalent serial order of the transactions. Show your work.

i. Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(D), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution:

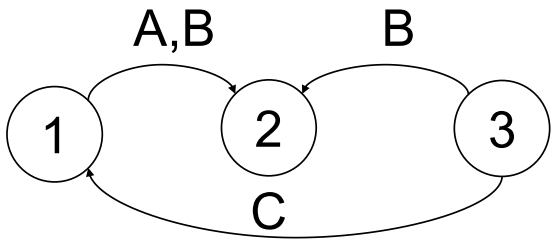


Not conflict serializable.

ii. Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(A), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution:



Serialization order: T3, T1, T2.

(c) A scheduler uses the strict two-phase locking protocol. In each of the cases below, indicate whether the scheduler may result in a deadlock. If you answer *yes*, then give an example of a schedule that results in deadlock.

i. Can these transactions result in deadlock?

```
T1: W1(A), W1(C), CO1

T2: W2(B), W2(D), CO2

T3: W3(A), W3(B), CO3
```

```
T4: W4(D), W4(A), CO4
```

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution:
```
W2(B),W3(A),W4(D),W2(D) -- now T2, T3, T4 are deadlocked.
```
Transaction T1 is not needed.

ii. Can these transactions result in deadlock?

```
T1: W1(A), W1(C), CO1

T2: W2(B), W2(D), CO2

T3: W3(A), W3(B), CO3

T4: W4(B), W4(C), CO4

T5: W5(C), W5(D), CO5
```

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution: No: all transactions acquire the elements in the order `A,B,C,D` and therefore they avoid deadlock.

15 Parallel Data Processing (182–193)

Once data no longer fits on one machine, the cost model changes: what matters is how much data crosses the network and how evenly the work divides. The questions here ask how a relation should be partitioned (by hash, by range, or round-robin), and what a join costs under each choice.

Skew is the recurring theme. Under a uniform distribution every server receives its fair share and the speedup is linear, so several problems first ask for the uniform case and then ask for the worst case, where a single value can send every matching tuple to one server. The gap between those two answers is the point of the exercise.

182. *(from CSE 544p, Autumn 2010, Final)*

Consider the following relations $R(A, B), S(B, C)$ and the following query:

```
select R.A, sum(S.C)
from R, S
where R.B = S.B
group by R.A
```

Assume that a block is about 10^4 bytes, and assume the following statistics:

$$
\begin{aligned}
B(R) &= 10^9 && \text{/* about 10 Terabytes */}\\
B(S) &= 10^9 && \text{/* about 10 Terabytes */}\\
B(R \bowtie S) &= B(R) + B(S) = 2 \cdot 10^9 &&\\
P &= 10^5 && \text{/* number of servers */}\\
M &= 10^5 && \text{/* about 1GB local memory at each server */}
\end{aligned}
$$

At the beginning of the execution, both relations R and S are distributed in a round-robin fashion among the P servers and are stored on their local disks.

(a) The query is evaluated in parallel using two map/reduce steps. We will call this PLAN 1.

Map/reduce 1. The *map* phase redistributes both R and S by their B attribute. The *reduce* phase performs a main-memory join of their R and S chunks.

Map/reduce 2. The *map* phase redistributes the join $R \bowtie S$ by the A attribute. The *reduce* phase performs a main-memory group-by A, computing the sum of the C values on the local chunk, then writes the result to disk.

Compute the total parallel I/O cost for PLAN 1. You have to turn in an integer number.

> **Solution:** Map 1: reads sends writes. Reduce 1: reads joins writes. Map 2: reads sends writes. Reduce 2: reads. $(B(R) + B(S))/P * [(1 + 1) + (1 + 1) + (1 + 1) + 1] = 140000$
>
> This question was ambiguous, because it said that the last reduce phase writes back to disk, which contradicts the convention we made in class. I accepted a wide range of answers.

(b) Now assume that the query is evaluated as follows (not longer using the map/reduce framework). We call this PLAN 2.

Step 1: Redistribute both R and S by their B attribute. The receiving severs will hold both table chunks in main memory.

Step 2: Perform a main-memory join the local R and S, and keep the result in main memory.

Step 3: Redistribute the join $R \bowtie S$ by the A attribute. The receiving servers keep their chunks in main memory.

Step 4: Each server performs a main-memory group-by A, computing the sum of the C values on the local chunk.

Compute the total parallel I/O cost for PLAN 2. You have to turn in an integer number.

Solution: $(B(R) + B(S))/P * 1 = 20000$

(c) For each of the statement below indicate whether it is true or false:

1. PLAN 1 is always more efficient than PLAN 2 because it uses the map/reduce infrastructure. True or False ?

   Solution: false

2. If any server fails during the execution of PLAN 1, then we do not need to restart the entire query execution, but instead can re-execute only a fragment of the computation. True or False ?

   Solution: true

3. If any server fails during the execution of PLAN 2, then we do not need to restart the entire query execution, but instead can re-execute only a fragment of the computation. True or False ?

   Solution: false

4. When R and S are distributed by their B attribute, it is best to use a different hash function for $R.B$ from $S.B$, to better achieve load balance. True or False ?

   Solution: false

5. Even if the hash function $h(A)$ used to distribute the join result $R \bowtie S$ (in preparation of the group-by) is perfectly uniform, the distribution may be heavily skewed if the data is skewed. True or False ?

   Solution: true

183. *(from CSE 344, Spring 2011, Final)*

(a) You work for an Information company, and your job is to manage a 1 TB dataset. You have a large database with 100 servers, and there is a job that runs every day and takes 10 hours to complete. You try to explain to your boss the concepts of speedup and scaleup. Help your boss understand these concepts by illustrating on your databases how and ideal speedup, and an ideal scaleup would work:

	Number of Processors	Database Size	Running time
Current	100	1 TB	10 hours
Ideal speedup:			
Ideal scaleup:			

Solution:

	Number of Processors	Database Size	Running time
Current	100	1 TB	10 hours
Ideal speedup:	1000	1 TB	1 hour
Ideal scaleup:	1000	10 TB	10 hours

(b) You are the salesperson of a database server company, and you are offering three kinds of parallel databases: (A) shared memory, (B) shared disk, and (C) shared nothing. Your customer is interested in the following criteria: speedup, scaleup, cost per processing unit, and ease of programming. Help your customer understand the pros and cons of these architecture by filling out the table below, with signs − (poor), = (neutral), or + (good).

	(A) Shared memory	(B) Shared disk	(C) Shared nothing
Speedup			
Scaleup			
Cost / processing unit			
Ease of programming			

Solution: The solution is somewhat debatable, but the idea was to answer in a way that contrasts the three options:

	(A) Shared memory	(B) Shared disk	(C) Shared nothing
Speedup	=	=	+
Scaleup	=	+	+
Cost / processing unit	−	=	+
Ease of programming	+	=	−

(c) We have a large table $R(A, B)$ where A is an integer attribute. The table has 1 Billion ($= 10^9$) records, and the values of A are exactly the first 1 Billion perfect

squares. That is, $R.A$ is $1, 4, 9, 16, \ldots, 10^{18}$. We have a parallel database with only two servers, $P = 2$, and wish to partition the table R among these servers. We consider two partition strategies: hash based partition, where the hash function is $h(A) = 1 + (A \texttt{ mod } 2)$, and range partition, where the two ranges are $[0, 0.5 \cdot 10^{18}]$ and $(0.5 \cdot 10^{18}, 10^{18}]$. Compute in each case the number of tuples stored on each server:

	# of tuples on Server 1	# of tuples on Server 2
Hash-partition		
Range partition		

Solution:

	# of tuples on Server 1	# of tuples on Server 2
Hash-partition	$0.5 \cdot 10^9$	$0.5 \cdot 10^9$
Range partition	$0.7 \cdot 10^9$	$0.3 \cdot 10^9$

184. *(from CSE 344, Winter 2012, Final)*

(a) We have a large database, on which we need to run repeatedly SQL queries. Each SQL query has up to 5 joins, a group-by, and some selections. We use a parallel database system, and we consider the following alternative evaluation strategies: (a) inter-query parallelism, (b) inter-operator parallelism, (c) intra-operator parallelism. In each case we deploy only one strategy, i.e. we do not combine them. Consider a job J consisting of several SQL queries, and assume it has the following running time:

- Job J runs in $T = 100$ minutes on 10 nodes.

Estimate the running time of that job if we increase the number of nodes from 10 to 100, in each of the six cases below. In each case, assume that the database is capable of delivering linear speedup, when the execution strategy is parallelizable.

Write the running time T on 100 nodes			
Job J consists of:	Type of Parallelism		
	Inter-query	Inter-operator	Intra-operator
1 SQL query			
1000 SQL queries			

Solution:

Job J consists of:	Type of Parallelism		
	Inter-query	Inter-operator	Intra-operator
1 SQL query	100	100	10
1000 SQL queries	10	100	10

(b) The query below computes the total number of customers with any given date of birth:

```
select birthdate, count(*)
from Customer x
group by x.birthdate
```

The attributes `birthdate` represents the date of birth of the customer: it contains the day and month only (not the year!). We evaluate the query using Map-Reduce. Assume:

- The relation `Customer` has 16MB

We consider choosing the block size to be one of the following three values: 128KB, 64KB, 32KB. (Recall that $1MB = 1024KB$; thus, if the block size is 128KB, then the `Customer` file has $16 \cdot 1024/128 = 128$ blocks.) Indicate the maximum number of instances that you can use and still achieve linear speedup, if the data is uniformly distributed. Assume that the number of map tasks is equal to the number of blocks, and that the number of reduce tasks is set to the number of instances.

i. The block size is 128KB. Maximum number of instances:

Solution: $16MB/128KB = 128$

ii. The block size is 64KB. Maximum number of instances:

Solution: $16MB/64KB = 256$

iii. The block size is 32KB. Maximum number of instances:

Solution: 365: # days/year

(c) A Map/Reduce Job runs on 10 instances, and uses 100 Map Tasks and 50 Reduce Tasks. The input file has 5GB, and the block size is 50K. We assume that the map function produces an output whose size is approximately equal to that of the input: in other words, the size of the intermediate result is also 5GB.

i. What is the total number of intermediate files to which the mappers write their outputs? Write the number of files:

Solution: $50 \cdot 100 = 5000$

ii. After a reducer copies, it needs to sort. How large is the file that needs to be sorted? Answer with the expected value. Write the size of the file:

Solution: $5GB/50 = 100MB$

iii. At any time, one instance runs only one map task, or only one reduce task. Suppose that a map task takes 1 minute to finish, and a reduce task also takes 1 minute to finish. What is the total running time for the Map/Reduce job? Write the time in minutes:

Solution: $100\frac{1}{10} + 50\frac{1}{10} = 15$minutes

iv. Continue to assume that each map task and each reduce task takes 1 minute. However, one single map task exhibits a skew, and take 10 minutes to complete instead of 1 minute; all other 99 map tasks still take 1 minute. What is the total running time for the Map/Reduce job? Write the time in minutes:

Solution: $15 + 9 = 24$ minutes

185. *(from CSE 344, Winter 2013, Final)*

(a) There is a new search engine in town! To answer keyword queries, they compute an *inverted index* , with the following schema:

```
Inv(word, url, occ)
```

Each record consists of a word, a URL where that word occurs, and the number of occurrences of that word at that URL. When a user issues a keyword search, say she searches for `xyz`, then the search engine runs a query like this:

```
SELECT url FROM Inv WHERE word = 'xyz'
ORDER BY occ DESC
```

`Inv` has **1 Terabyte** ($= 10^{12}$ bytes).

The company needs to support **10,000 queries per second**.

They plan to use a cluster of servers, each server running an independent SQL Lite database, on a fragment of `Inv`. You are charged with managing their server cluster and databases.

You measure the performance of SQL Lite on a single server: on a subset of `Inv` and with an index on `Inv.word` SQL Lite answers about **10 queries per second**. This performance is largely independent on the size of the subset of `Inv` (this, of course, is due to the index); it is also independent on whether the queries return an empty or a nonempty answer.

i. How many servers P do you purchase in order to answer 10,000 queries per second on the 1 Terabyte database?
$P =$

Solution: 1000

ii. Next, you need to decide how to partition the table `Inv`. Assuming the value P you determined at the previous question, indicate how many queries per second your system can support if:

- `Inv` is block partition. The number of queries per second is:

  Solution: 10

- `Inv` is hash partition on `word`. The number of queries per second is:

  Solution: 10,000

- `Inv` is hash partition on `url`. The number of queries per second is:

  Solution: 10

(b) Every week or so the company needs to recompute the entire inverted index, from a fresh crawl of the Web. The crawl is stored in a table with the following schema:

```
WebCrawl(url, fullText)
```

To generate the inverted index, you use a MapReduce job. Fill in the details of the Map and the Reduce functions below. Your answer may consists of pseudocode (they don't need to be perfect Java). Write your answers in the boxes provided:

```
function map(String url, String fullText):
  for each word w in fullText:
```

```
/* EmitIntermediate(...)  */
```

```
function reduce(String [          ], Iterator [          ]):
  /* fill in the parameters */

    /* compute word, url, occ */




    /* Emit(word, url, occ) */
```

Solution:

Solution 1:

The intermediate key is a word, and its values are all the URLs where it occurs.

```
function map(String url, String fullText)
  for each word w in fullText
     EmitIntermediary(w, url)

function reduce(String w, Iterator urls)
  hashTable = empty
  /* group by url and count */
  for each url in urls
      if hashTable[url] == NULL
           hashTable[url] = 1
      else hashTable[url] = hashTable[url]+1
  for each url in hashTable
      Emit(w, url, hashTable[url])
```

Solution 2:

The intermediate key is a (word, url) pair, and the values are $1, 1, 1, \ldots$

```
function map(String url, String fullText)
  for each word w in fullText
     EmitIntermediary(concat(w,',',url), 1)

function reduce(String wu, Iterator values)
  c = 0
  for each v in values
      c = c + ParseInt(v)
  Emit(wu[0], wu[1], c)
```

(c) Assume the following about your MapReduce job:

- The input (`WebCrawl`) has 1TB ($= 10^{12}$).

- The chunk size is 100MB ($= 10^8$).
- The number of servers is 1000.
- The number of reduce tasks is 10,000 ($= 10^4$).

Answer the following questions:

i. How many map tasks are in your job?
Number of map tasks:

> **Solution:** $10^4 = 10,000$

ii. How many intermediate files are created?
Number of intermediate files:

> **Solution:** $10^8 = 100,000,000$

iii. Both the map tasks and the reduce tasks are scheduled on the same 1000 servers. Each server is single-core, and runs only one map task or only one reduce task at any time. All map tasks and all reduce tasks are perfectly uniform, i.e. tasks that start at the same time will finish at the same time.
After the first 2,000 reduce tasks have completed, server number 333 crashes and never recovers. Which of the following happens? Circle exactly one:

1. The MR system continues running on 999 servers, without taking any other actions.
2. The MR system continues with 999 servers, as follows. All 999 reducers currently running will continue to run, and the system will reschedule reduce task number 2333 on a different server at a later time.
3. The MR system aborts all active reduce tasks currently running *that have not finished the copy phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.
4. The MR system aborts all active reduce tasks currently running *that have not finished the sort phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.
5. The MR system aborts *all* 999 reduce tasks currently running; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining 8000 reduce tasks on the 999 servers.
6. The MR system aborts the entire job, because it cannot recover in this case.
7. None of the above.

Solution: item 3

186. *(from CSE 344, Autumn 2013, Final)*

(a) A social network company stores the "likes" data in a very large file with a simple schema:

```
Likes(person1, person2)
```

The following statistics are known about `Likes`:

Number of people in the social network	10^8
Number of records in `Likes`	10^{10}
Everybody likes somebody!	true
Size of the table `Likes`	1TB (10^{12})
HDFS chunk size	100MB (10^8)
Number of workers for the MapReduce jobs:	100

We run a MapReduce program with the following functions:

```
function map(person1, person2)
   EmitIntermediary(person1, person2)

function reduce(person, list)
   Emit(person, length(list))
```


Number of people in the social network	10^8
Number of records in Likes	10^{10}
Everybody likes somebody!	true
Size of the table Likes	1TB (10^{12})
HDFS chunk size	100MB (10^8)
Number of workers for the MapReduce jobs:	100

```
function map(person1, person2)
   EmitIntermediary(person1, person2)

function reduce(person, list)
   Emit(person, length(list))
```


Answer the following questions approximately. If the answer is totally dependent on the configuration then write 'System dependent'.

i. How many map functions are there approximately? Write a number or say "system dependent":

Solution: 10^{10}

ii. How many map tasks are there approximately? Write a number or say "system dependent":

Solution: 10^4

iii. How many reduce functions are there approximately? Write a number or say "system dependent":

Solution: 10^8

iv. How many reduce tasks are there approximately? Write a number or say "system dependent":

Solution: System dependent

v. How large is the size of the intermediate result approximately? Write a number or say "system dependent":

Solution: 1TB (10^{12})

vi. How many records are in the final result approximately? Write a number or say "system dependent":

Solution: 10^8

(b) We modify the MapReduce program by adding a `combine` function:

```
function map(person1, person2)
   EmitIntermediary(person1, person2)

function combine(person, list)
   EmitIntermediary(person, length(list))

function reduce(person, list)
   Emit(person, sum(list))
```

For each of the following questions indicate whether they are true or false. The statements claim that certain parameters "decrease", without saying how much: you should respond "true" even if the decrease is tiny, but should respond "false" if there is no change.

i. The addition of the `combine` function will cause the number of reduce functions to decrease.
True or false?

Solution: False

ii. The addition of the `combine` function will cause the number of reduce tasks to decrease.
True or false?

Solution: False

iii. The addition of the `combine` function will cause the size of the intermediate result to decrease.
True or false?

Solution: True

iv. The addition of the `combine` function will cause the size of the final result to decrease.
True or false?

Solution: False

v. The `combine` function is executed *before* the data is reshuffled.
True or false?

Solution: True

187. *(from CSE 544p, Winter 2014, Final; CSE 544p, Autumn 2015, Final)*

You need to implement an inverted-index for a document management company. The *inverted index* is a table with the following schema:

```
Invndx(word, did, occ)
```

Each record consists of a word, a document ID where that word occurs, and the number of occurrences of that word in that document. Users search the documents by keywords; for example a user may search for `xyz`, then your system runs a query like this:

```
SELECT did, occ FROM Invndx WHERE word = 'xyz'
ORDER BY occ DESC
```

`Invndx` has **1 Terabyte** ($= 10^{12}$ bytes).

The company needs to support **10,000 queries per second**.

They plan to use a cluster of servers, each server running an independent postgres database, on a horizontal fragment of `Invndx`. You are charged with managing their server cluster and databases.

You measure the performance of a single-server postgres installation and notice that, given an index on `Invndx.word`, postgres can answers about **10 queries per second**,

and that this performance is largely independent on the size of `Invndx`. (this, of course, is due to the index); it is also independent on whether the queries return an empty or a nonempty answer.

Solution:

```
create table invndx(word text, did int, occ int);
insert into invndx values ('abc', 3, 1);
insert into invndx values ('abc', 4, 2);
insert into invndx values ('xyz', 2, 1);
insert into invndx values ('xyz', 3, 1);
select did, occ from Invndx where word = 'xyz' order by occ desc;
```

(a) i. How many servers P do you purchase in order to answer 10,000 queries per second on the 1 Terabyte database?
$P =$

Solution: 1000

Solution: Almost everyone answered correctly.

ii. Next, you need to decide how to partition the table `Invndx`. Assuming the value P you determined at the previous question, indicate how many queries per second your system can support if:

- `Invndx` is block partition. The number of queries per second is:

  Solution: 10

- `Invnx` is hash partition on `word`. The number of queries per second is:

  Solution: 10,000

- `Invndx` is hash partition on `docid`. The number of queries per second is:

  Solution: 10

(b) Since the document collection is updated continuously, you must recompute the entire inverted index on a weekly bases. The company provides you with a crawl of the entire document collection stored in a large HDFS text file where each line has the following schema:

```
DocCrawl(did, fullText)
```

You need to generate the inverted index using a MapReduce job. Fill in the details of the Map and the Reduce functions below. Your answer may consists of pseudocode (they don't need to be perfect Java).

```
function map(String did, String fullText):
  for each word w in fullText:
    [                                        ]
    [                                        ]
    [                                        ]
    [                                        ]
    [ /* EmitIntermediate(...)  */           ]

function reduce(String [          ], Iterator [          ]):
  /* fill in the parameters */

    [ /* compute word, did, occ */           ]
    [                                        ]
    [                                        ]
    [                                        ]
    [                                        ]
    [ /* Emit(word, did, occ) */             ]
```

Solution:

Solution 1:

The intermediate key is a word, and its values are all the did's where it occurs.

Map function:

```
function map(String did, String fullText)
  for each word w in fullText
     EmitIntermediary(w, did)
```

Reduce function:

```
function reduce(String w, Iterator dids)
  hashTable = empty
  /* group by did and count */
  for each did in dids
      if hashTable[did] == NULL
           hashTable[did] = 1
      else hashTable[did] = hashTable[did]+1
  for each did in hashTable
      Emit(w, did, hashTable[did])
```

Solution 2:

The intermediate key is a (word, did) pair, and the values are $1, 1, 1, \ldots$

```
function map(String did, String fullText)
  for each word w in fullText
     EmitIntermediary(concat(w,',',did), 1)

function reduce(String wd, Iterator values)
  c = 0
  for each v in values
      c = c + ParseInt(v)
  Emit(wd[0], wd[1], c)
```

(c) Assume the following about your MapReduce job:

- The input (`DocCrawl`) has 1TB ($= 10^{12}$).
- The chunk size is 100MB ($= 10^8$).
- The number of servers is 1000.
- The number of reduce tasks is 10,000 ($= 10^4$).

Answer the following questions:

i. How many map tasks are in your job?
Number of map tasks:

Solution: $10^4 = 10,000$

Solution: Almost everyone answered correctly.

ii. How many intermediate files are created?
Number of intermediate files:

Solution: $10^8 = 100,000,000$

Solution: Most answered correctly.

iii. Both the map tasks and the reduce tasks are scheduled on the same 1000 servers. Each server is single-core, and runs only one map task or only one reduce task at any time. All map tasks and all reduce tasks are perfectly uniform, i.e. tasks that start at the same time will finish at the same time.
After the first 2,000 reduce tasks have completed, server number 333 crashes and never recovers. Which of the following happens? Select only one:

1. **Continue:** The MR system continues running on 999 servers, without taking any other actions.
2. **Restart one reducer:** The MR system continues with 999 servers, as

follows. All 999 reducers currently running will continue to run, and the system will reschedule reduce task number 2333 on a different server at a later time.

3. **Abort/restart reducers before copy:** The MR system aborts all active reduce tasks currently running *that have not finished the copy phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.
4. **Abort/restart reducers before sort:** The MR system aborts all active reduce tasks currently running *that have not finished the sort phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.
5. **Abort/restart all reducers:** The MR system aborts *all* 999 reduce tasks currently running; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining 8000 reduce tasks on the 999 servers.
6. **Abort/restart job:** The MR system aborts the entire job, because it cannot recover in this case.
7. **None of the above.**

Solution: item 3

188. *(from CSE 344, Winter 2016, Final)*

(a) Consider a social network database with two relations shown below:

`User(uid,name)`	`1 Million tuples`
`Follows(uid1,uid2)`	`10 Million tuples`

The table `User` contains user information, while `Follows` tells us that user1 follows user2. Suppose we are computing the following query:

```
select x.uid1,x.uid2,y.name
from Follows x, User y
where x.uid2 = y.uid
```

We use a distributed system with p servers, and compute the join using partitioned hash-join. In other words:

- The system partitions `User(uid,name)` by applying a hash function to `uid`,
- partitions `Follows` by applying a hash function to `uid2`,

- then each server computes a join of its local data.

On $p = 10$ servers, the query runs in 1000 seconds. Estimate the runtime of the system in each of the cases below, assuming the number of servers is increased as shown. Your numbers are only *estimates*: try to estimate within a factor of 2. For example, if the question were *what is the runtime on one server?* then you would answer 10,000 seconds, since one server must do the work of all the 10 servers that took 1000 seconds, although one server could run much faster than 10×1000 second.

i. Assume that every user follows at most 5 users, and is followed by at most 5 users:

$p =$	10	100	1000	100000
Time=	1000s			

Solution:

$p =$	10	100	1000	100000
hline Time=	1000s	100s	10s	0.1s

ii. As in item i, every user follows and is followed by at most 5 users, except for user `'JB'` who is followed by 10,000 users.

$p =$	10	100	1000	100000
Time=	1000s			

Solution: All the 10,000 `Follows` records with followers of `'JB'` will be sent to the same server. Notice that 10,000 is the average load per server when $p = 1000$, hence we do not have any speedup after that point:

$p =$	10	100	1000	100000
hline Time=	1000s	100s	20s	10.1s

Also OK: 1000,100,10,10, or numbers that are close enough.

iii. As in item i, every user follows and is followed by at most 5 users, except for user `'JB'` who is followed by 100,000 users.

$p =$	10	100	1000	100000
Time=	1000s			

Solution: Now 100,000 records will be sent to one single server; this is the average load for $p = 100$ serers, hence there is more speedup beyond that point:

$p =$	10	100	1000	100000
hline Time=	1000s	200s	110s	100.1s

Also OK: 1000, 100, 100, 100

(b) We are running a MapReduce job over HDFS. Our input file has 10^{10} records, its size is 1TB= 10^{12}B. Hadoop's block size is configured at 100KB. Answer each of the questions below.

i. How many map tasks will the MapReduce system create by default? If there

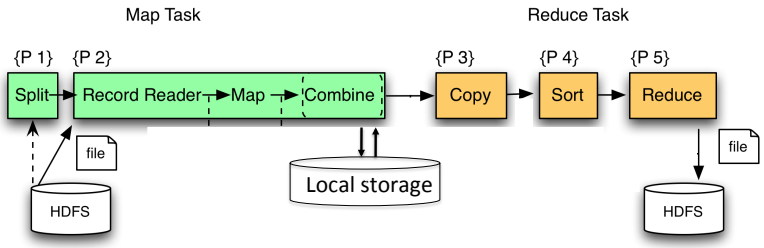
is no default, then indicate so.
Number of map tasks:

Solution: 10^7

ii. How many reduce tasks will the MapReduce system create by default? If there is no default, then indicate so.
Number of reduce tasks:

Solution: No default

For the next few questions, recall the steps of a MapReduce job from the lecture notes:



iii. The *Copy* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.
Yes or no?

Solution: Yes

iv. The *Sort* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.
Yes or no?

Solution: No

v. The *Reduce* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.

Yes or no?

Solution: No

(c) A MapReduce job runs on 100 workers and has 500 reduce tasks. At some point in time, all map tasks have finished, and 150 reduce tasks have finished too: the system is executing 100 reduce tasks, while another 250 reduce tasks are still waiting to be scheduled. At this point worker number 44 fails: the worker and its local disk are lost and not recoverable. Indicate which of the following will happen:

1. The system continues executing the 99 active tasks, then will schedule the remaining 250 tasks on the 99 remaining workers.
2. The system continues executing the 99 active tasks, then will schedule the remaining 251 tasks on the 99 remaining workers (including the reduce task that was running on worker 44).
3. The system reruns all 500 reduce tasks.
4. The system continues executing the 99 active tasks, then reruns the map tasks that had been ran on worker 44, and after that continues executing the remaining reduce tasks.
5. The system needs to restart the entire job (all map tasks and all reduce tasks) on 99 workers.

Solution: 4

Solution:

(d) The following questions compare MapReduce to Spark. For each statement indicate whether it is true or false.

i. A program that involves iteration (such as page rank) requires the execution of several separate MapReduce jobs.
True or False?

Solution: True

ii. A program that involves iteration (such as page rank) requires the execution of several separate Spark program.
True or False?

Solution: False

iii. In a MapReduce program, all intermediate results are stored on disk.
True or False?

Solution: True

iv. In a Spark program, all intermediate results are stored on disk.
True or False?

Solution: False

v. If a worker fails during the execution of a MapReduce program, then the entire program needs to be restarted.
True or False?

Solution: False

vi. If a worker fails during the execution of a Spark program, then the entire program needs to be restarted.
True or False?

Solution: False

vii. MapReduce is ideally suited for OLTP applications.
True or False?

Solution: False

viii. Spark is ideally suited for OLTP applications.
True or False?

Solution: False

189. *(from CSE 414, Spring 2016, Final)*

(a) We are running a MapReduce job over HDFS with a block size of 100KB. The input file has 1TB= 10^{12} Bytes. Answer each of the questions below.

i. How many map tasks will the MapReduce system create by default? If there is no default, then indicate so.
Number of map tasks:

Solution: 10^7

ii. How many reduce tasks will the MapReduce system create by default? If there is no default, then indicate so.
Number of reduce tasks:

Solution: No default

iii. We are running a MapReduce job over HDFS with a block size of 100KB. Most of the computation time is spend in the Map phase; the Reduce phase is very fast. Answer each question below:

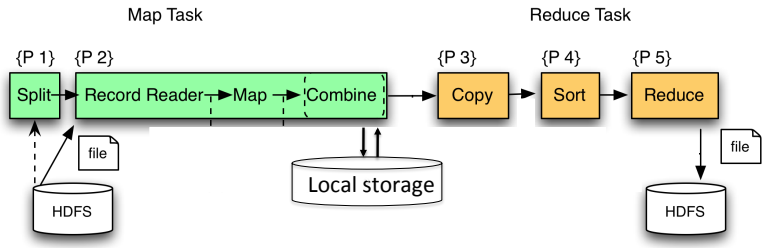
α) Our input file has size 1TB= 10^{12} Bytes. When we run the MapReduce job on 10 workers, the job takes 500 minutes. How long will the job take if we use 100 workers?
Write the number of minutes:

Solution: 50 minutes

β) Our input file has size 50KB= $5 \cdot 10^4$ Bytes. When we run the MapReduce job on 10 workers, the job takes 10 minutes. How long will the job take if we use 100 workers?
Write the number of minutes:

Solution: 10 minutes

For the next few questions, recall the steps of a MapReduce job from the lecture notes:



iv. The *Copy* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish. (Recall that MapReduce implements the shuffle phase by having each reducer copy its values from the mappers.)
Yes or no?

Solution: Yes

v. The *Sort* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.
Yes or no?

Solution: No

vi. The *Reduce* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.
Yes or no?

Solution: No

(b) The following questions compare MapReduce to Spark. For each statement indicate whether it is true or false.

i. In order to cope with worker failure, MapReduce stores all intermediate results to disk.
True or False?

Solution: True

ii. In order to cope with worker failure, Spark stores all intermediate results to disk.
True or False?

Solution: False

iii. If a worker fails during the execution of a MapReduce program, then the entire program needs to be restarted.
True or False?

Solution: False

iv. If a worker fails during the execution of a Spark program, then the entire program needs to be restarted.
True or False?

Solution: False

v. Which of the following three Spark transformations corresponds to the `map` phase of a MapReduce program?

1. `map`(f) : `RDD`$[T] \Rightarrow$ `RDD`$[U]$, where $f : T \Rightarrow U$.
2. `flatMap`(f) : `RDD`$[T] \Rightarrow$ `RDD`$[U]$, where $f : T \Rightarrow$ `Seq`$[U]$.
3. `map`(f) : `RDD`$[T] \Rightarrow$ `RDD`$[T]$, where $f : T \Rightarrow$ `Bool`.

Answer 1,2, or 3:

Solution: 2

Solution: Reshuffle

vi. What is the difference between `RDD`$[T]$ and `Seq`$[T]$? Select all answers that apply.

α) An RDD can be longer than a Sequence. True or false?

Solution: False

β) An RDD is distributed while a Sequence is stored on a single node. True or false?

Solution: True

γ) An RDD can be nested, e.g. we can have a data type `RDD`$[(K, $`RDD`$[T])]$. True or false?

Solution: False

δ) A Sequence can be nested, e.g. we can have a data type `Seq`$[(K, $`Seq`$[T])]$. True or false?

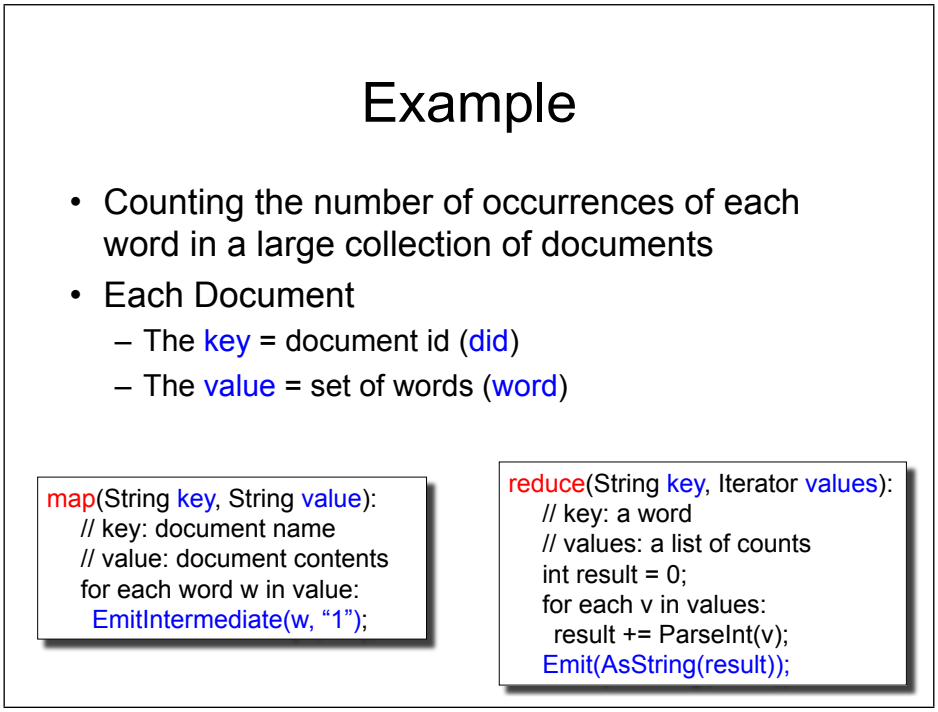
Solution: False

190. *(from CSE 344, Autumn 2017, Final)*

(a) Write a Map/Reduce program to compute the following SQL query:

```
select p.category, count(*)
from Product p
where p.price > 100
group by p.category
having sum(p.quantity) > 200
```

You need to write two functions: Map and Reduce. Use pseudo-code to write both functions, as we did in class; the relevant slide from the lectures is shown below to refresh your memory. You may assume that the Map function has a single input, namely the `Product` record.



Solution:

```
function map(String p):
    if p.price > 100
        then EmitIntermediate(p.category, p)

function reduce(String c, iterator ps):
    s = 0;
    for each p in ps:
        s = s + p.quantity
    if s > 200 then
        r = 0
        for each p in ps:
            r = r + 1
        Emit(c, AsString(r))
```

(b) Consider the following SQL query, computing for each person p the number of its followers under 20 years old:

```
select p.pid, p.name, count(*)
from Person p, Follower f
where p.pid = f.follows and f.age <= 20
group by p.pid, p.name
```

We have $T(\texttt{Person}) = 10^7$, $T(\texttt{Follower}) = 10^9$, $\min(age) = 1$, $\max(age) = 99$, and $V(\texttt{Follower}, follows) = 10^7$. Assume the logical query plan is:

$$\gamma_{pid,name,count(*)}(\texttt{Person} \bowtie \sigma_{age<20}(\texttt{Follower}))$$

The system computes the query on 200 servers as follows.

Step 1 Initially both relations are distributed randomly and uniformly on all servers. The selection is computed on the fly.

Communication Next, there is a communication step where `Person` is hash-partitioned on `pid`, and `Follower` is hash-partitioned on `follows`.

Step 2 Each server computes the join and group by locally.

Assume the values of `age` are uniformly distributed among `Follower` and that the hash function is uniform. For anything else, assume the worst case distribution.

i. How many `Person` records does each server hold during each step of the computation? Report the maximum number of records during steps one and two, i.e. do not add up the number of records held during steps one and two.

> **Solution:** $10^7/200 = 5000$

ii. How many `Follower` records does each server hold during each step of the computation? Like in the previous question, report the maximum number, not their sum.

> **Solution:** $T(\texttt{Follower})/5 = 2 \cdot 10^8$. There are $T(\texttt{Follower})/5$ followers with $age \leq 20$. Since `follows` may be skewed, all of them may be following the same person.

(c) Answer the following questions:

i. If you can compute a query in parallel on 1000 servers in 15 minutes, then you can always compute it in parallel on 500 servers in at most 30 minutes. True or false?

> **Solution:** True

ii. If you can compute a query in parallel on 500 servers in 30 minutes, then you can always compute it in parallel on 1000 servers in at most 15 minutes. True or false?

> **Solution:** False

iii. If one server of a parallel database system fails, then the entire job that the system was computing is restarted; if one server of a Map/Reduce job fails then the M/R system can recover and continue the job. Why the difference? Choose one answer below:

1. Parallel databases can answer complex SQL queries, while an M/R job is restricted to two functions (map and reduce).
2. Parallel databases were developed in the 80's when fault tolerance tech-

niques were not yet developed, while M/R was developed in the early 2000's, when the technology exists.

3. Parallel database systems run on a small number of servers (10–20) while M/R jobs can run on a very large number of servers (10000) making failure much more likely.

4. Parallel database systems must maintain statistics on the database to be used by the query optimizer, while M/R systems do not maintain statistics on the data.

Choose one answer:

Solution: 3

(d) Answer the following questions about Map/Reduce:

i. The number of map tasks is usually equal to the number of blocks of the input file. True or false?

Solution: True

ii. The number of reduce tasks is usually equal to the number of blocks of the output file. True or false?

Solution: False

iii. The first reduce task can start as early as the first map task finishes. True or false?

Solution: False

iv. The number of reduce tasks must be less than or equal to the number of servers. True or false?

Solution: False

v. The combine function is executed by the map task. True or false?

Solution: True

vi. If a server running a reduce task fails during the computation, then the master will start that reduce task on another server (or the same server, if it recovers). True or false?

Solution: True

vii. In a Map/Reduce job, *shuffle* refers to the process of distributing the input data to the map tasks. True or false?

Solution: False

(e) Answer the following questions about Spark:

i. A Resilient Distributed Dataset (RDD) is another term for a distributed file on disks. True or false?

Solution: False

ii. Consider the function $map(f) : RDD\langle T\rangle \rightarrow RDD\langle U\rangle$, where $f : T \rightarrow U$. The output RDD can never be strictly smaller than the input RDD. (In other words, the number of U elements in the output can never be strictly smaller than the number of T elements in the input.) True or false?

Solution: True

iii. Consider the function $flatMap(f) : RDD\langle T\rangle \rightarrow RDD\langle U\rangle$, where $f : T \rightarrow Seq(U)$. The output RDD can never be strictly smaller than the input RDD. (Recall: $Seq(T)$ means sequence of T, or array of T.) True or false?

Solution: False

191. *(from CSE 344, Winter 2019, Final)*

Consider two relations with the following schema and statistics:

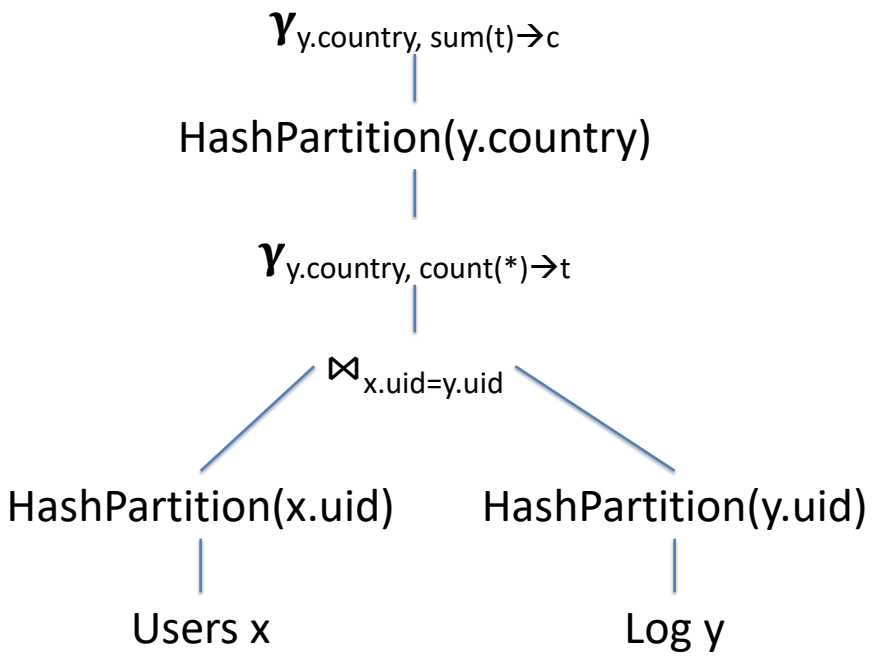
`Users(uid, name, country)`
`Log(uid, url)`
$T(\texttt{Users}) = 10,000,000$
$T(\texttt{Log}) = 20,000,000,000$
$V(\texttt{Users}, \texttt{country}) = 100$
$V(\texttt{Log}, \texttt{uid}) = 5,000,000$

The data is initially block partitioned on 1000 servers, so that each server holds 10000 records; we assume that, initially, the records of both `Users` and `Log` are randomly distributed to the 1000 servers,

The query below counts the number log entries from each country:

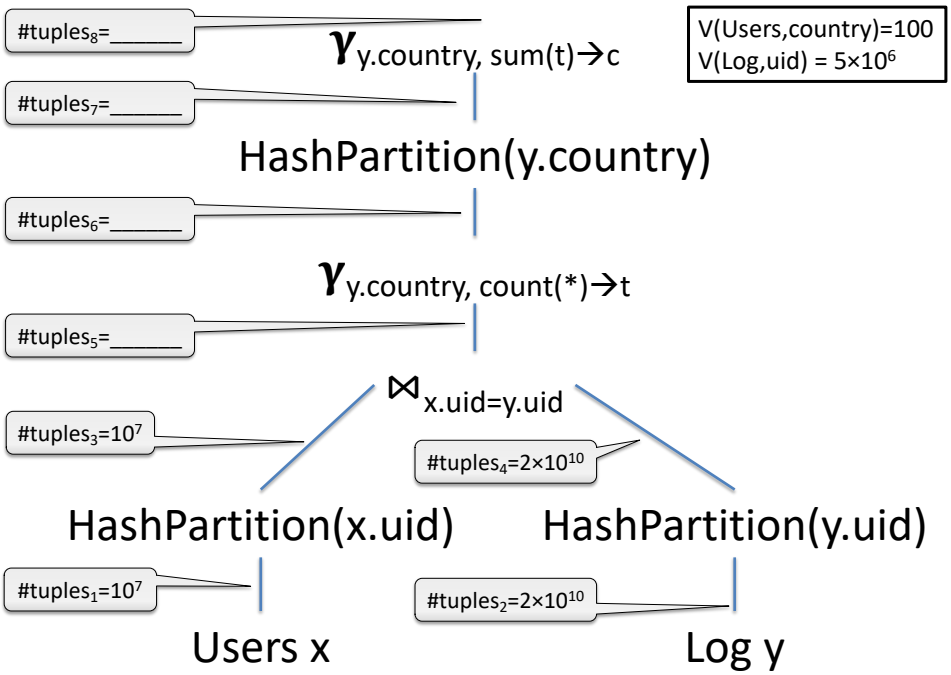
```
select x.country, count(*) as c
from Users x, Log y
where x.uid = y.uid
group by x.country
```

(a) The query optimizer chooses to compute the query using the following plan:

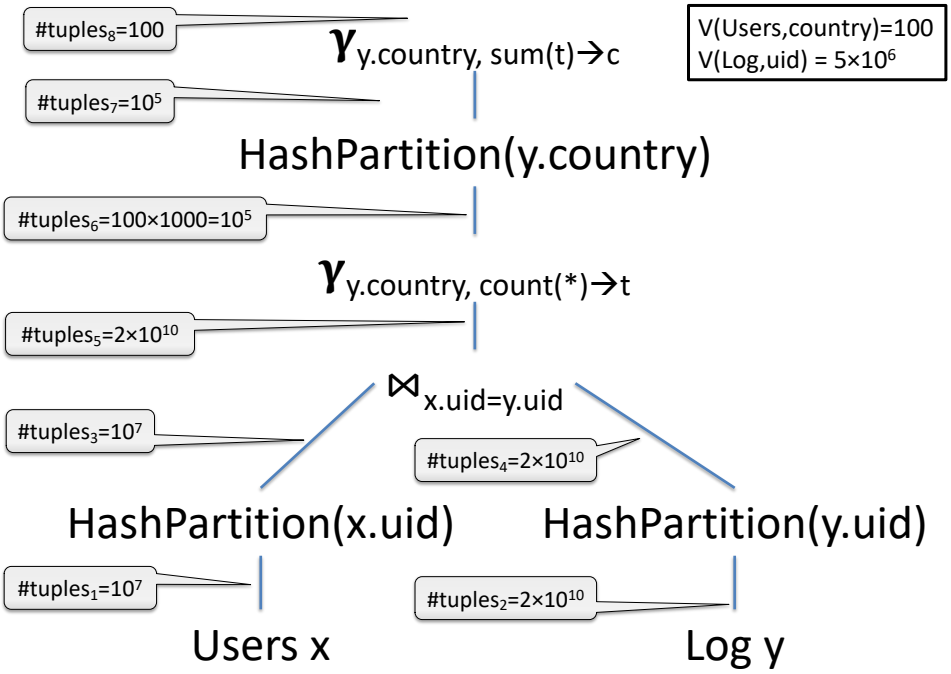


In words, the data is first hash-partitioned on `uid`, followed by a local join and a local group-by, followed by a repartition on `country`, and a final group-by.

i. Estimate the size (number of tuples) of each intermediate relation in the plan. You may assume that the data is uniformly distributed and that the attributes are independent. Write your answer in the figure below, by filling out each missing `#tuples`. Notice that `#tuples` represents the total number of tuples, from all servers.

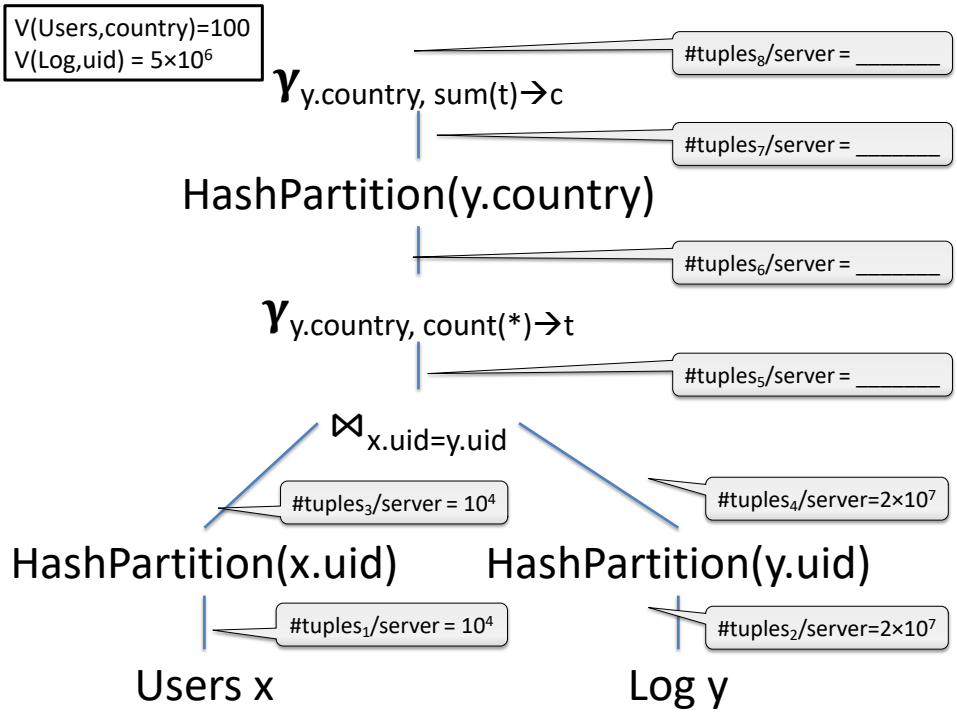


Solution:

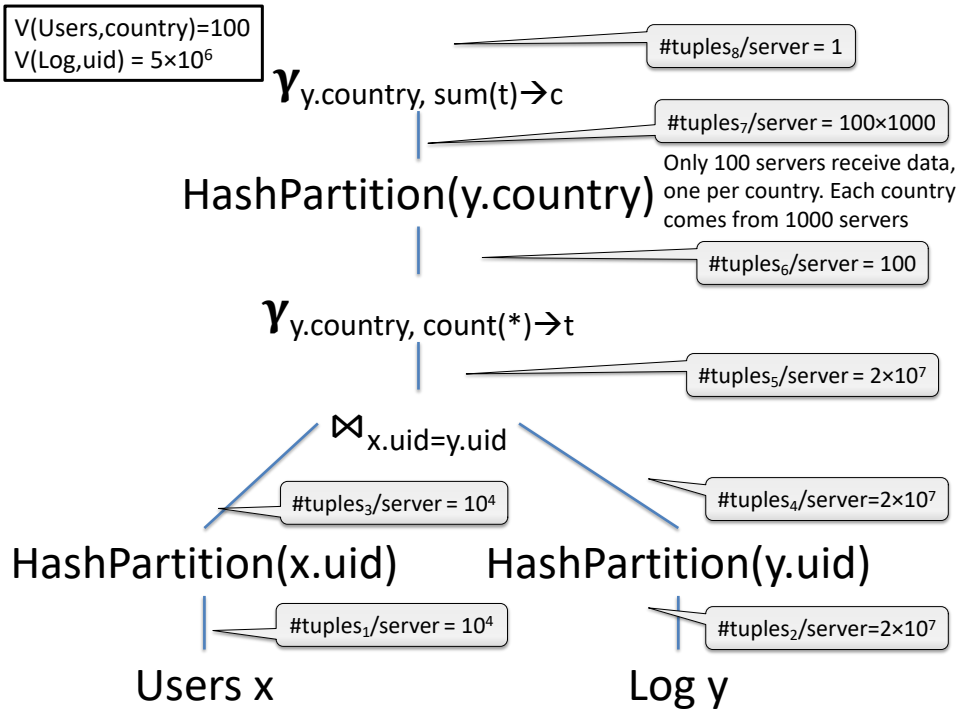


ii. Assuming the system uses 1000 servers to compute the query, indicate the number of tuples per server at each step (i.e. the load per server). Assume the data is uniformly distributed, in the best possible way. In the case when not

all servers receive the same number of tuples, then indicate the largest number. (This question should be easy to answer if you answered the previous one.)



Solution:

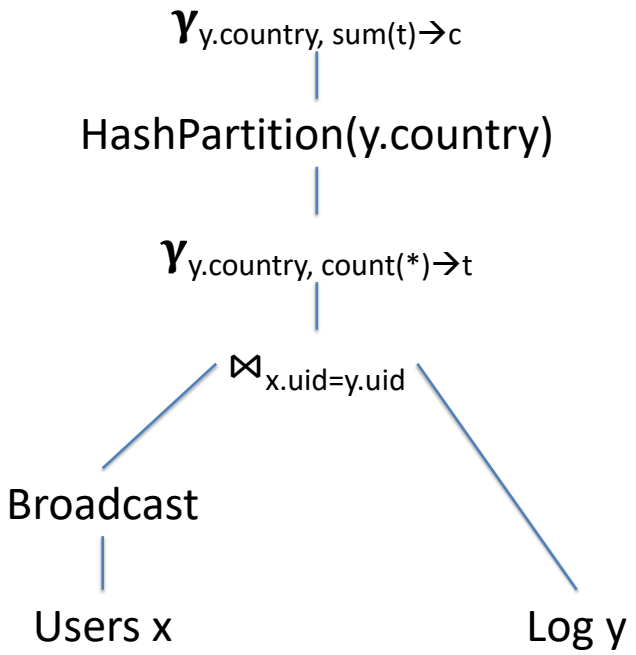


iii. Now assume that the data is not uniform. What is the largest possible number

of tuples received by any server, and which intermediate result creates this largest number of tuples? Write your answers by referring to the figure above, for example you may write $\texttt{tuples}_4/\texttt{server} = 7 \cdot 10^{19}$ (not a real answer).

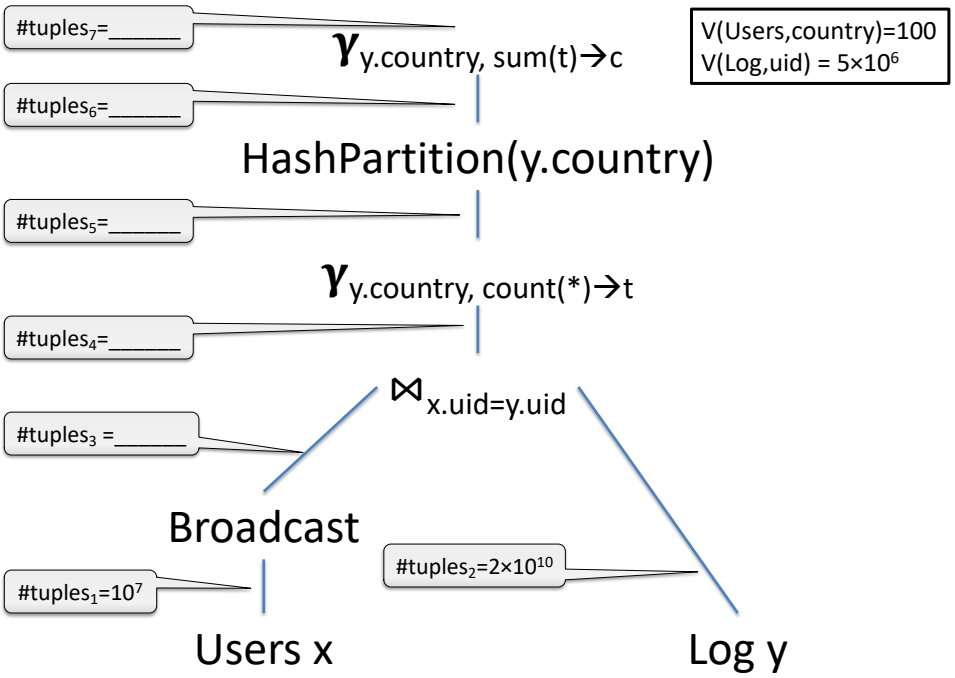
> **Solution:** The maximum values are as follows, and they happen when a single user issues all the log entries. The `Log` is still distributed uniformly, but all `Users` are sent to the same server.

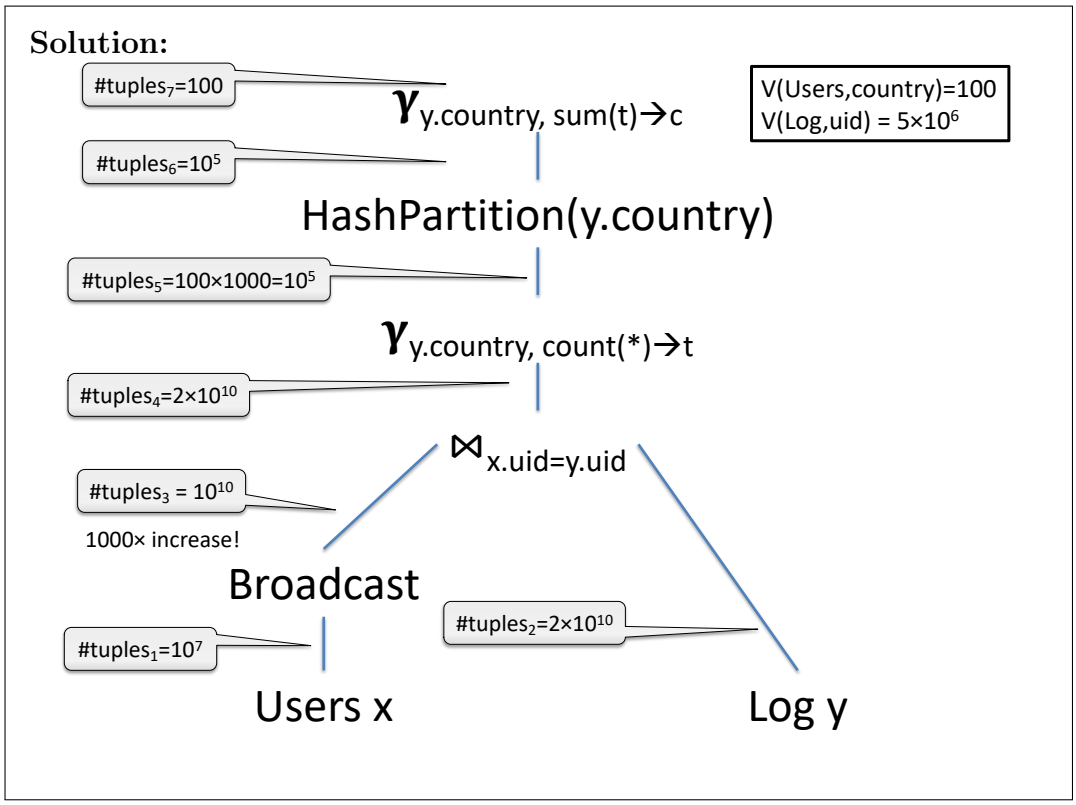
(b) Now the query optimizer chooses to compute the query using the following plan:



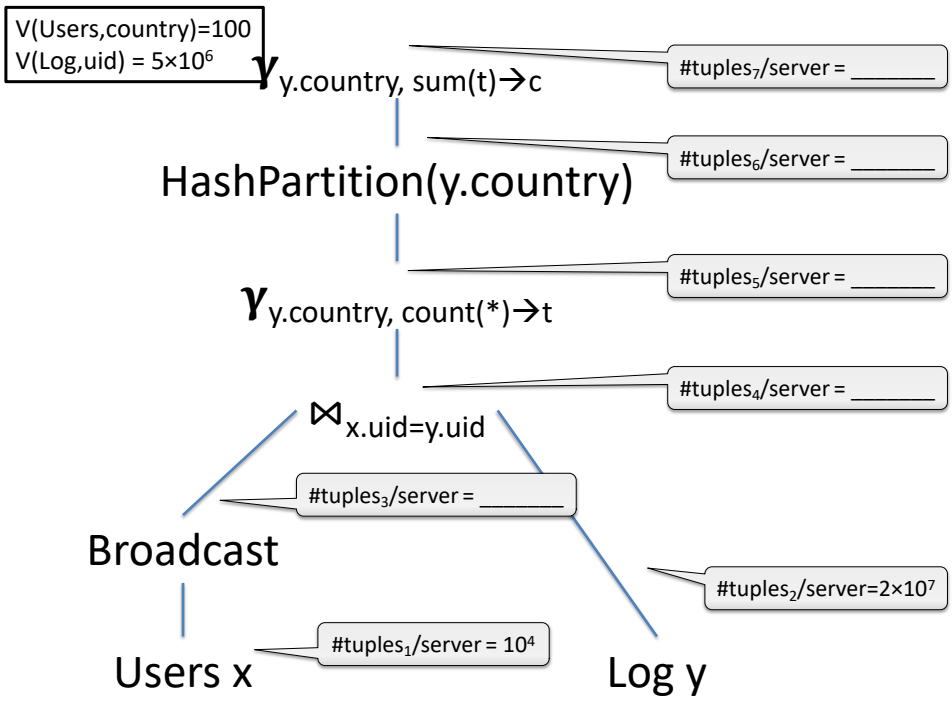
In other words, it first broadcasts `Users`, then computes the local join followed by a local group-by, then reshuffles the data based on `country`, followed by a final local group-by. Answer the same questions as before:

i. Estimate the size (number of tuples) of each intermediate relation in the plan, as before.

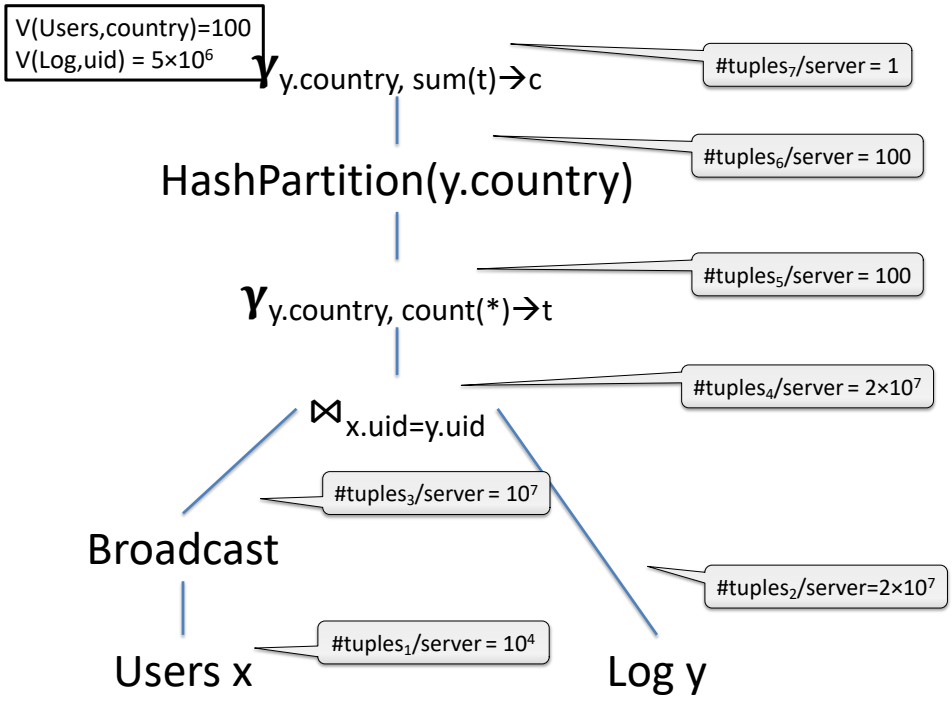




ii. Assuming the system uses 1000 servers to compute the query, indicate the number of tuples per server at each step (i.e. the load per server), as before.



Solution:



iii. Now assume that the data is not uniform. What is the largest possible number of tuples received by any server, and which intermediate result creates this largest number of tuples?

Solution: Skew does not affect `Users` because they are broadcast, and does not affect `Log`, because we hash partition on the key.
Skew only affects the partition on `country`. Skew actually helps here, because after the last hash-partition only one server receives the 1000 records (one record from each server).

192. *(from CSE 414, Spring 2019, Final)*

Consider the same relational database, and the same statistics:

$$T(\texttt{Sample}) = 100,000 \qquad T(\texttt{Analysis}) = 20,000,000$$

We are storing and processing the data on $P = 100$ servers. The data is initially block partitioned, and we compute the following query:

```
select *
from Sample x, Analysis y
where x.sid = y.sid;
```

The query is computed distributively, on the 100 servers. You will be asked to estimate the number of answer tuples returned by each server. You only need to estimate the number of tuples in the **final answer**, not in any intermediate results. In case this estimate differs among servers, indicate the maximum number.

(a) **Plan 1:** `Sample` is broadcast to all 100 servers, then each server joins it with its local fragment of `Analysis`.

i. Estimate the number of tuples/server assuming the data is uniform.

Solution: 200,000

ii. Estimate the number of tuples/server assuming the data is skewed.

Solution: 200,000

(b) **Plan 2:** `Sample` and `Analysis` are hash-partitioned on the `sid` attribute on the 100 servers, then each server joins its local fragments.

i. Estimate the number of tuples/server assuming the data is uniform.

Solution: 200,000

ii. Estimate the number of tuples/server assuming the data is skewed.

Solution: 20,000,000

193. *(from CSE 544p, Winter 2014, Final; CSE 544p, Autumn 2015, Final; CSE 544p, Spring 2021, Final)*

A database consists of the following elements:

$$A_1, \ldots, A_{1000}$$

The system runs a workload of transactions of the following kind:

```
BEGIN
READ(A_i)
/* ...  compute...  compute...  compute ...  for 0.1 seconds */
WRITE(A_i)
COMMIT
```

Each transaction reads one element A_i, performs some intensive computation, then writes the same element back. Different transactions may access the same element or different elements.

The system is shared-memory, has 100 CPUs, and uses inter-query parallelism (multiple transactions are run in parallel, but each transaction runs on a single CPU). For serializability, the system uses one lock per element (like SQL Server).[4]

How many transactions per second can the system execute in each case below?

(a) All transactions access the same element A_1. That is, the transactions update elements in this sequence: $A_1, A_1, A_1, \ldots$

Answer Write the number of transactions per second (TPS):

Solution: 10 TPS. The transactions are executed serially.
Almost everyone answered correctly.

(b) The transactions update only elements from the first 50 elements. More precisely, each transaction reads/writes an element A_i chosen uniformly at random from among $A_1, \ldots, A_{50}$. For example, the transactions may update the elements in this sequence: $A_{44}, A_2, A_{13}, A_2, A_{36}, A_{11}, \ldots$

Answer Write the number of transactions per second (TPS):

[4]While in a real multicore system speedup is affected dramatically by latch contention, our multicore system has magic latches that do not cause any slowdown.

Solution: 500 TPS. For example, suppose the elements are updated round-robin: $A_1, A_2, \ldots, A_{50}, A_1, A_2, \ldots$ T51 can start only after T1 terminates, which is 0.1s after T1 starts. We run 50 transactions every 0.1s, or 500 TPS. We can sustain this rate because there are 100 CPUs and each can sustain 10 TPS max.

Almost everyone answered correctly.

(c) The transactions update all elements. More precisely, each transaction reads/writes an element A_i, chosen uniformly at random from among $A_1, \ldots, A_{1000}$.

Answer Write the number of transactions per second (TPS):

Solution: 1000 TPS, because the system becomes CPU bound. The max rate per CPU is 10 TPS and there are 100 CPUs.

Almost everyone answered correctly.

16 Parallel Databases and MapReduce (194–203)

MapReduce was the novel massively parallel, distributed data processing paradigm of the early 2000s. It is no longer in use today, since SQL engines have been extended to also do massively parallel data processing.

This section covers the programming models built on top of parallel storage: MapReduce and, in the later exams, Spark. The questions ask you to express a query as one or more MapReduce jobs, to say what the map and reduce functions emit, and to count the rounds of communication a computation requires.

The mechanical part is choosing the key emitted by the map phase, since that key alone determines what ends up together at a reducer; most solutions here are short once that decision is made. Joins need the join attribute as the key, grouping needs the grouping attribute, and a computation needing two different keys needs two jobs.

Later problems compare this model with the parallel database of the preceding section, which is the comparison the courses were making at the time these exams were given.

194. *(from CSE 544p, Winter 2014, Final)*

(a) Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where sti means that transaction Ti starts and coi means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starting time. In each case below, say what happens with the last request.

You have to choose between one of the following four possible answers:

1. the request is accepted,
2. the request is ignored,
3. the transaction is delayed (i.e. transaction is put on wait),
4. the transaction is rolled back.

i. st1; st2; st3; r1(A); w1(A); r2(A);
What will the system do in response to the request r2(A) ?

Solution: r2(A): T2 is delayed.

ii. st1; st2; r2(A); co2; r1(A); w1(A);
What will the system do in response to the request w1(A) ?

Solution: w1(A): T1 is rolled back.

iii. st1; st2; st3; r1(A); w3(A); co3; r2(A);
What will the system do in response to the request r2(A) ?

Solution: r2(A): T2 is rolled back.

iv. st1; st2; r1(A); r2(A); w1(B); w2(B);
What will the system do in response to the request w2(B) ?

Solution: w2(B): accepted.

v. st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)
What will the system do in response to the request w2(A) ?

Solution: w2(A): ignored.

(b) Consider a social network stored in a database with the following schema:

```
Person(pid, name, mood)
Friends(pid1, pid2)
```

Both `Friends.pid1` and `Friends.pid2` are foreign keys to `Person`. Consider the following procedure which *sets my mood to be the same as that of my friends*:

```
Set_mood(%P)
  START TRANSACTION
    ch = select count(*)
         from Friends y, Person z
         where y.pid1 = %P
           and y.pid2 = z.pid
           and z.mood = 'HAPPY'

    cs = select count(*)
         from Friends y, Person z
         where y.pid1 = %P
           and y.pid2 = z.pid
           and z.mood = 'SAD'

    if ch > cs
      then update Person
           set mood = 'HAPPY'
           where pid = %P
      else update Person
           set mood = 'SAD'
           where pid = %P
   END TRANSACTION
```

Assume that the database is static, i.e. no records are inserted or deleted.

i. Suppose there are only two users in the system, call them Alice and Bob, and they are friends with each other: Alice is Bob's only friend, and Bob is Alice's only friend. Initially their moods are as follows:

Alice.mood = HAPPY Bob.mood = SAD

Both Alice and Bob run the Set_mood procedure at about the same time. Assuming that the scheduler allows only serializable schedules, which of the following four outcomes are possible? Check all that apply:

Alice	Bob	Possible? Y/N
HAPPY	HAPPY	
HAPPY	SAD	
SAD	HAPPY	
SAD	SAD	

Solution: Y/N/N/Y

ii. Assume now that the database system is running a concurrency control manager based on snapshot isolation (SI). Assuming that both Alice and Bob run the Set_mood procedure at about the same time, which of the following four outcomes are possible? Check all that apply:

Alice	Bob	Possible? Y/N
HAPPY	HAPPY	
HAPPY	SAD	
SAD	HAPPY	
SAD	SAD	

Solution: Y/N/Y/Y

iii. Your database system is running SI, but you must write an application that ensures full serializability. Your task is to rewrite the application above, in order to ensure that the schedule is serializable even on a database running SI. Your new rewritten application must satisfy the following:

- You will not use locks, or other explicit synchronization mechanisms (which may not be available in any RDBMS). Instead, you will only use regular read and/or write operations from/to the database.
- If a single instance of the procedure is run on the database, then it has exactly the same semantics described above: set my mood as that of the majority of my friends. In other words, you are not allowed to change the semantics of the procedure.
- If multiple instances are run on the database, then, under SI, the schedule is guaranteed serializable.

Solution:

```
Set_mood(%P)
  START TRANSACTION
    /* . . . same as above . . */
    /* then add the following dummy update */
    update Person
    set mood = mood
    where pid in
          select z.pid
          from Person x, Friends y, Person z
          where x.pid = %P
           and  x.pid = y.pid1
           and  y.pid2 = z.pid
   END TRANSACTION
```

The only way to create conflicts is to do an extra write. Credit was given only for the `update Person set mood ...` statement.

(c) For each of the statements below indicate whether it is true of false about the ARIES recovery manager:

i. During the normal operation of a database (i.e. not during recovery) no update records in the log are undone.
true or false ?

Solution: false

ii. During the normal operation of a database (i.e. not during recovery) no update records in the log are redone.
true or false ?

Solution: true

iii. During the normal operation of a database (i.e. not during recovery) no CLR records are ever written to the log.
true or false ?

Solution: false

iv. During recovery from a crash no update record is processed both during the redo and during the undo phase. (In other words, an update record is either redone or undone, but not both.)
true or false ?

Solution: true

v. During recovery from a crash the CLR records in the log are neither undone nor redone. (In other words the CLR records server a different purpose, not to be redone or undone during recovery.)
true or false ?

Solution: false

vi. Suppose that two update log entries were written in the log on behalf of a transaction, then the system crashed while the transaction was active. Even if the system crashes repeatedly during recovery, there will never be more than two CLR records written on behalf of this transaction.
true or false ?

Solution: true

vii. All log entries preceding the last `begin_checkpoint` can be safely deleted from the log in order to reclaim their disc space.
true or false ?

Solution: false

195. *(from CSE 544p, Autumn 2015, Final)*

(a) Consider a concurrency control manager by timestamps. Below are several sequences of events; sti means that transaction Ti starts and coi means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starting time. In each case below, say what happens with the last request.

You have to choose between one of the following four possible answers:

1. the request is accepted,
2. the request is ignored,
3. the transaction is delayed (i.e. transaction is put on wait),
4. the transaction is rolled back.

i. st1; st2; st3; r1(A); w3(A); co3; r2(A);
What will the system do in response to the request r2(A) ?

Solution: r2(A): T2 is rolled back.

ii. st1; st2; r2(A); co2; r1(A); w1(A);
What will the system do in response to the request w1(A) ?

Solution: w1(A): T1 is rolled back.

iii. st1; st2; r1(A); r2(A); w1(B); w2(B);
What will the system do in response to the request w2(B) ?

Solution: w2(B): accepted.

iv. st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)
What will the system do in response to the request w2(A) ?

Solution: w2(A): ignored.

v. st1; st2; st3; r1(A); w1(A); r2(A);
What will the system do in response to the request r2(A) ?

Solution: r2(A): T2 is delayed.

(b) We have a database of airports and flights between them. Each airport decides whether they are going to support the Daylight Savings Time policy. The schema is:

```
Airport(code, dst)
Flight(c1,c2)
```

Both `Flight.c1` and `Flight.c2` are foreign keys to `Airport`. Some airports decide to set their DST according to the majority of their connecting airports. They run the procedure below, which sets the DST policy to 'Y' iff the majority of their neighbors have the DST policy set to 'Y':

```
Set_DST(%C)
  START TRANSACTION
    cy = select count(*)
         from Flight x, Airport y
         where x.c1 = %C
           and x.c2 = y.code
           and y.dst = 'Y'

    cn = select count(*)
         from Flight x, Airport y
         where x.c1 = %C
           and x.c2 = y.code
           and y.dst = 'N'

    if cy > cn
      then update Airport
           set dst = 'Y'
           where code = %C
      else update Airport
           set dst = 'N'
           where code = %C
   END TRANSACTION
```

Assume that the database is static, i.e. no records are inserted or deleted.

i. Suppose there are only two airports in the system, call them ABC and DEF, and there are two flights, one from ABC to DEF and one from DEF to ABC. Initially their DST settings are the following:

ABC.DST = 'Y' DEF.DST = 'N'

Both airports run the SET_DST procedure at about the same time. Assuming that the scheduler allows only serializable schedules, which of the following four outcomes are possible? Check all that apply:

ABC.DST	DEF.DST	Possible? Y/N
'Y'	'Y'	
'Y'	'N'	
'N'	'Y'	
'N'	'N'	

Solution: Y/N/N/Y

ii. Assume now that the database system is running a concurrency control manager based on snapshot isolation (SI). Assuming that both airports run the

SET_DST procedure at about the same time, which of the following four outcomes are possible? Check all that apply:

ABC	DEF	Possible? Y/N
'Y'	'Y'	
'Y'	'N'	
'N'	'Y'	
'N'	'N'	

Solution: Y/N/Y/Y

iii. Your database system is running SI, but you must write an application that ensures full serializability. Your task is to rewrite the application above, in order to ensure that the schedule is serializable even on a database running SI. Your new rewritten application must satisfy the following:

- You will not use locks, or other explicit synchronization mechanisms (which may not be available in some RDBMS). Instead, you will only use regular read and/or write operations from/to the database.
- If a single instance of the procedure is run on the database, then it has exactly the same semantics described above: set my the DST policy the same as that of the majority of the connected airports. In other words, you are not allowed to change the semantics of the procedure.
- If multiple instances are run on the database, then, under SI, the schedule is guaranteed serializable.

Solution:

```
Set_DST(%C)
  START TRANSACTION
    /* . . . same as above . . */
    /* then add the following dummy update */
    update Airport x
    set dst = dst
    where code in
          select y.c2 as code
          from Flight y
          where %C = y.c1
   END TRANSACTION
```

(c) For each of the statements below indicate whether it is true of false about the ARIES recovery manager:

i. During the normal operation of a database (i.e. not during recovery) no updates in the log are undone.
true or false ?

Solution: false

ii. During the normal operation of a database (i.e. not during recovery) no updates

in the log are redone.
true or false ?

Solution: true

iii. During the normal operation of a database (i.e. not during recovery) no CLR records are ever written to the log.
true or false ?

Solution: false

iv. During recovery from a crash no log record is used both during the redo and during the undo phase. (In other words, a log record is either redone or undone, but not both.)
true or false ?

Solution: true

v. CLR log records are never used during recovery (in other words they serve a different purpose, not for recovery).
true or false ?

Solution: false

vi. Suppose that transaction performed two updates to the database, resulting in two log entries, then the system crashed, and crashed repeatedly during recovery. No matter how often the system crashes again, there are never more than two CLR records belonging to this transaction.
true or false ?

Solution: true

vii. All log entries preceding the last `begin_checkpoint` can be safely deleted from the log in order to reclaim their disc space.
true or false ?

Solution: false

viii. During recovery, the first undo log entry always precedes the first redo log entry.
true or false ?

Solution: false

ix. There is a separate ARIES log file for every user of the database.
true or false ?

Solution: false

x. The log may contain more than one CHECKPOINT entry
true or false ?

Solution: true

196. *(from CSE 344, Winter 2016, Final)*

(a) Consider a concurrency control manager that uses strict two phase locking that schedules three transactions:

- $T1 : R_1(A), R_1(B), W_1(A), W_1(B), Co_1$
- $T2 : R_2(B), W_2(B), R_2(C), W_2(C), Co_2$
- $T3 : R_3(C), W_3(C), R_3(A), W_3(A), Co_3$

Each transaction begins with its first read operation, and commits with the Co statement. Answer the following questions for each of the schedules below:

- Is the schedule conflict-serializable? If yes, indicate a serialization order.
- Is this schedule possible under a strict 2PL protocol?
- If strict 2PL does not allow this schedule because it denies a read or a write request, is the system in a deadlock at the time when the request is denied?

i. Schedule 1:

$R_2(B), W_2(B), R_3(C), W_3(C), R_3(A), W_3(A), Co_3, R_2(C), W_2(C), Co_2, R_1(A), R_1(B), W_1(A), W_1(B), Co_1$

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: yes: 3,2,1

β) Is it possible under strict 2PL?

Solution: yes

γ) Does strict 2PL lead to a deadlock?

Solution: no

ii. Schedule 2:

$R_2(B), W_2(B), R_3(C), W_3(C), R_1(A), R_1(B), W_1(A), W_1(B), Co_1, R_2(C), W_2(C), Co_2, R_3(A), W_3(A), Co_3$

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: no $L(C)$ and none can make progress.

β) Is it possible under strict 2PL?

Solution: no $L(C)$ and none can make progress.

γ) Does strict 2PL lead to a deadlock?

Solution: yes: T_1 holds $L(A)$, T_2 holds $L(B)$, T_3 holds $L(C)$ and none can make progress.

iii. Schedule 3:

$R_1(A), R_1(B), R_2(B), W_2(B), R_2(C), W_2(C), Co_2, R_3(C), W_3(C), R_3(A), W_3(A), Co_3, W_1(A), W_1(B), Co_1$

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: no:$R_2(B)$ is denied (or $W_2(B)$ is denied if shared locks are used)

β) Is it possible under strict 2PL?

Solution: no

γ) Does strict 2PL lead to a deadlock?

Solution: no

iv. Schedule 4:

$R_1(A), R_1(B), W_1(A), R_3(C), W_3(C), R_3(A), W_3(A), Co_3, W_1(B), R_2(B), W_2(B), Co_1, R_2(C), W_2(C), Co_2$

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: yes: 1,3,4;

β) Is it possible under strict 2PL?

Solution: no: $W_2(B)$ is impossible since T_1 holds the lock on B

γ) Does strict 2PL lead to a deadlock?

Solution: no

(b) Consider the following three transactions:

- $T1 : R_1(A), W_1(B), Co_1$
- $T2 : R_2(B), W_2(C), Co_2$
- $T3 : R_3(C), W_3(D), Co_3$

Given an example of a conflict-serializable schedule that has the following properties: transaction $T1$ commits before transaction $T3$ starts, and the equivalent serial order is $T3, T2, T1$.

Solution: $R_1(A), R_2(B), W_1(B), Co_1, R_3(C), W_2(C), Co_2, W_3(D), Co_3$
Variations include: swap the first two reads (of A and B), and the last two writes (of C and D, together with the commit order)

(c) A read-only transaction is a transaction that only reads from the database, without writing/inserting deleting. Answer the questions below.

i. If all transactions are read-only, then every schedule is serializable.
True or False?

Solution: True

ii. If no transaction reads the same element twice, then the serialization level READ COMMITTED is equivalent to REPEATABLE READS.
True or False?

Solution: False

Solution: A counterexample is: $R_1(A), W_2(B), W_2(A), Co_2, R_1(B)$. This schedule is possible under READ COMMITTED, but not under REPEATABLE READS (since the latter uses strict 2PL, which on a static database ensures conflict serializability, while this schedule is not conflict serializable).

iii. If no transaction inserts or deletes records to/from the database, then the serialization level REPEATABLE READS is equivalent to SERIALIZABLE.
True or False?

Solution: True

iv. The reason why some applications use serialization levels other than SERIALIZABLE is because they would not be correct under the SERIALIZABLE isolation level.
True or False?

Solution: False

v. In Sqlite phantoms are not possible.
True or False?

Solution: True

vi. The difference between Two Phase Locking and Strict Two Phase Locking is that the latter avoids deadlocks, while the former may allow deadlocks.
True or False?

Solution: False

vii. Only one transaction can hold a *shared lock* at any time.
True or False?

Solution: False

viii. Only one transaction can hold an *exclusive lock* at any time.
True or False?

Solution: True

197. *(from CSE 414, Spring 2016, Final)*

(a) For each statement below, indicate whether it is true or false:

i. In a static database, every serializable schedule is also conflict-serializable.
Answer Yes/No:

Solution: No

ii. In a static database, every conflict-serializable schedule is also serializable.
Answer Yes/No:

Solution: Yes

iii. SQL Lite uses optimistic concurrency control.
Answer Yes/No:

Solution: No

iv. In a static database, strict Two-Phase-Locking is guaranteed to produce a serializable schedule.
Answer Yes/No:

Solution: Yes

v. In a static database, strict Two-Phase-Locking is guaranteed to produce a conflict-serializable schedule.
Answer Yes/No:

Solution: Yes

vi. In a static database, strict Two-Phase-Locking is guaranteed to avoid deadlocks.
Answer Yes/No:

Solution: No

(b) For each of the schedules below, indicate whether they are conflict-serializable. If you answer *yes*, then give the equivalent serial order of the transactions. Show your work.

i. Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(D), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution: No: there is the cycle $3 \xrightarrow{B} 2 \xrightarrow{D} 3$ in the dependency graph.

ii. Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(A), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution: Yes: the dependency graph is acyclic and a serialization order is $T3, T1, T2$.

(c) A scheduler uses the strict two-phase locking protocol. In each of the cases below, indicate whether the scheduler may result in a deadlock. If you answer *yes*, then give an example of a schedule that results in deadlock.

i. Can these transactions result in deadlock?

T1:	W1(A), W1(C), CO1
T2:	W2(B), W2(D), CO2
T3:	W3(A), W3(B), CO3
T4:	W4(D), W4(A), CO4

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution:
`W2(B),W3(A),W4(D),W2(D) -- now T2, T3, T4 are deadlocked.`
Transaction T1 is not needed.

ii. Can these transactions result in deadlock?

T1:	W1(A), W1(C), CO1
T2:	W2(B), W2(D), CO2
T3:	W3(A), W3(B), CO3
T4:	W4(B), W4(C), CO4
T5:	W5(C), W5(D), CO5

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution: No: all transactions acquire the elements in the order `A,B,C,D` and therefore they avoid deadlock.

(d) We run the four transactions below concurrently on sqlite. Consider the following schedule:

Time	T1	T2	T3	T4
1	begin transaction			
2	select * from R where A = 1			
3		begin transaction		
4		select * from R where A = 2		
5	update R set B=10 where A=1			
6			begin transaction	
7			select * from R where A = 3	
8	commit			
9				begin transaction
10				select * from R where A = 4
11		commit		
12			commit	
13				commit

Circle the first action that will not be permitted by sqlite. Then modify the schedule,

by filling out the table below, to reflect the actual schedule on sqlite. You can only delay actions, not move them earlier in time.

Time	T1	T2	T3	T4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

Solution:
T4 is not allowed to begin until T1 commits:

Time	T1	T2	T3	T4
1	begin transaction			
2	select * from R where A = 1			
3		begin transaction		
4		select * from R where A = 2		
5	update R set B=10 where A=1			
6			begin transaction	
7			select * from R where A = 3	
8				
9				
10				
11		commit		
12			commit	
13	commit			
14				begin transaction
15				select * from R where A = 4
16				commit

198. *(from CSE 344, Autumn 2017, Final)*

(a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule.

i.

$$R_1(A),\ R_4(D),\ W_3(C),\ R_2(B),\ W_2(A),\ R_1(C),\ W_3(B)$$

Solution: Not conflict serializable. The conflicts are $R_1(A) \to W_2(A)$ giving $T_1 \to T_2$, $R_2(B) \to W_3(B)$ giving $T_2 \to T_3$, and $W_3(C) \to R_1(C)$ giving $T_3 \to T_1$. The precedence graph therefore contains the cycle $T_1 \to T_2 \to T_3 \to T_1$.

ii.

$$R_1(A),\ W_2(B),\ R_3(C),\ W_3(A),\ R_3(D),\ R_4(B),\ W_4(D),\ W_2(C)$$

Solution: Conflict serializable. The conflicts are $R_1(A) \to W_3(A)$ giving $T_1 \to T_3$, $W_2(B) \to R_4(B)$ giving $T_2 \to T_4$, $R_3(C) \to W_2(C)$ giving $T_3 \to T_2$, and $R_3(D) \to W_4(D)$ giving $T_3 \to T_4$. The precedence graph is acyclic,

so an equivalent serial schedule is

$$T_1,\ T_3,\ T_2,\ T_4$$

(b) An application uses a database of customers and their accounts:

```
Customer(cid,name,acc)
Account(acc, balance)
```

The application runs concurrently many instances of the following transaction (shown here in pseudocode):

```
function updateAccount(c, delta)
  BEGIN TRANSACTION
  myCustomer = "SELECT * FROM Customer WHERE cid = [c]";
  myAccount  = "SELECT * FROM Account WHERE acc = [myCustomer.acc]";
  if myAccount.balance + delta < 0 then ROLLBACK
  "UPDATE Account
   SET balance a = [a.balance + delta]
   WHERE acc = [myCustomer.acc]"
  COMMIT
```

Thus, the transaction first looks up the customer with `cid = c`, call him/her `myCustomer`, then looks up the account `myCustomer.acc`, and adds `delta` to the account balance, checking first that it doesn't become negative.

The concurrency control manager of the database uses a locking based concurrency control mechanism. It uses the strict 2PL mechanism for write locks, the mechanism decided by the isolation level for read locks, and table-level locking for phantoms. The only transactions on the system are instances of the `updateAccount` shown above.

In each case below you are asked to indicate (a) whether the schedule is guaranteed to be serializable, and (b) whether a deadlock may occur.

i. Suppose we run all transactions with
`SET ISOLATION LEVEL READ UNCOMMITTED`
Is it always serializable? Can deadlock occur?

Solution: Serializable: No. Deadlock: No.

ii. Suppose we run all transactions with
`SET ISOLATION LEVEL READ COMMITTED`
Is it always serializable? Can deadlock occur?

Solution: Serializable: No. Deadlock: No.

iii. Suppose we run all transactions with
`SET ISOLATION LEVEL REPEATABLE READ`
Is it always serializable? Can deadlock occur?

Solution: Serializable: Yes. Deadlock: Yes.

iv. Suppose we run all transactions with
`SET ISOLATION LEVEL SERIALIZABLE`
Is it always serializable? Can deadlock occur?

Solution: Serializable: Yes. Deadlock: Yes.

v. Assume that the concurrency control mechanism uses strict 2PL with exclusive locks only; in other words, it no longer uses separate read and write locks.
Is it always serializable? Can deadlock occur?

Solution: Serializable: Yes. Deadlock: No.

(c) For each of the following statements indicate whether it is true or false:

i. In a static database, every serializable schedule is conflict serializable. True or false?

Solution: False

ii. In a dynamic database, every serializable schedule is conflict serializable. True or false?

Solution: False

iii. In a static database, every conflict serializable schedule is serializable. True or false?

Solution: True

iv. In a dynamic database, every conflict serializable schedule is serializable. True or false?

Solution: False

v. In SQLite all schedules are guaranteed serializable. True or false?

Solution: True

199. *(from CSE 344, Winter 2019, Final)*

(a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule.

i.

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), R_1(C), W_4(C)$$

Solution: Conflict serializable, in the unique order $1, 3, 2, 4$
TO DRAW THE GRAPH.

ii.

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), W_4(C), R_1(C)$$

Solution: Not conflict serializable.
TO DRAW THE GRAPH

(b) A concurrency manager uses strict 2PL. The system runs only three transactions concurrently, denoted T_1, T_2, T_3. In each case below indicate whether there exists a schedule that leads to a deadlock. If you answer yes, then write a schedule that leads to a deadlock (up to the deadlock).

i. The transactions are:

$$T_1 : R_1(A), W_1(B)$$
$$T_2 : R_2(B), W_2(C)$$
$$T_3 : R_3(C), W_3(A)$$

Can this lead to a deadlock?

Solution: Yes

Solution:

$$R_1(A), R_2(B), R_3(C), \underbrace{W_1(B)}_{\texttt{deadlock}}$$

ii. The transactions are:

$$T_1 : R_1(A), W_1(B), W_1(C)$$
$$T_2 : W_2(A), R_2(B), W_2(C)$$
$$T_3 : W_3(A), W_3(B), R_3(C)$$

Can this lead to a deadlock?

Solution: No

iii. The transactions are:

$$T_1 : W_1(D), R_1(A), W_1(B)$$
$$T_2 : W_2(D), R_2(B), W_2(C)$$
$$T_3 : W_3(D), R_3(C), W_3(A)$$

Can this lead to a deadlock?

Solution: No

(c) Consider the following three transactions, where St_i indicates the start of T_i and Co_i indicates the commit of T_i:

$$T_1 : St_1, R_1(A), W_1(B), Co_1$$
$$T_2 : St_2, R_2(B), W_2(C), Co_2$$
$$T_3 : St_3, R_3(C), Co_3$$

Give an example of a schedule with the following properties: (1) the schedule is serializable; (2) transaction T_1 ends before transaction T_3 begins; (3) the only serialization order is T_3, T_2, T_1. In your schedule include the St_i and Co_i actions.

Solution:

$$St_1, R_1(A), St_2, R_2(B), W_1(B), Co_1, St_3, R_3(C), W_2(C), Co_2, Co_3$$

(d) The SQL standard defines three *weak isolation levels*: dirty reads, read committed, and repeatable reads. As you know:

- *Dirty reads* means no read locks.
- *Read committed* means short-duration read locks.
- *Repeatable reads* means long-duration read locks (full 2PL).

Consider two transactions T_1, T_2. For each schedule below, indicate under which isolation level that schedule is possible:

	Time →							
Schedule$_1$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	$R_1(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_2$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	Co_2	$R_1(B)$,	$W_1(A)$, Co_1
Schedule$_3$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_2(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_4$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_1(A)$,	Co_1	$W_2(B)$, Co_2

Write yes/no answers below:

	Dirty reads	Read committed	Repeatable Reads
Schedule$_1$			
Schedule$_2$			
Schedule$_3$			
Schedule$_4$			

Solution:

	Dirty reads	Read committed	Repeatable Reads
$\texttt{Schedule}_1$	yes	no	no
$\texttt{Schedule}_2$	yes	yes	no
$\texttt{Schedule}_3$	yes	no	no
$\texttt{Schedule}_4$	no	no	no

(e) For each of the following statements indicate whether it is true or false:

i. If there are an odd number of transactions, then deadlock can never occur. True or false?

Solution: False

ii. In a static database, every serializable schedule is conflict serializable. True or false?

Solution: False

iii. In a dynamic database, every serializable schedule is conflict serializable. True or false?

Solution: False

iv. In a static database, every conflict serializable schedule is serializable. True or false?

Solution: True

v. In a dynamic database, every conflict serializable schedule is serializable. True or false?

Solution: False

vi. T_1 holds a shared lock on A. When T_2 requests a shared lock on A, the scheduler

will grant it
True or false?

Solution: True

vii. T_1 holds a shared lock on A. When T_2 requests an exclusive lock on A, the scheduler will grant it
True or false?

Solution: False

viii. A concurrency management system uses strict 2PL, with shared locks for reads and exclusive locks for writes. If all transactions are read-only, then deadlock is not possible.
True or false?

Solution: True

ix. An OLAP workload ("Online analytical processing") means a workload consisting of simple queries and many updates.
True or false?

Solution: False

x. An OLTP workload ("Online transaction processing") means a workload consisting of simple queries and may updates.
True or false?

Solution: True

200. *(from CSE 414, Spring 2019, Final)*

(a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule. Show your work by drawing the precedence graph.

i.

$$R_2(A), R_4(C), W_2(B), W_1(D), W_3(A), R_4(D), R_1(B), W_3(C)$$

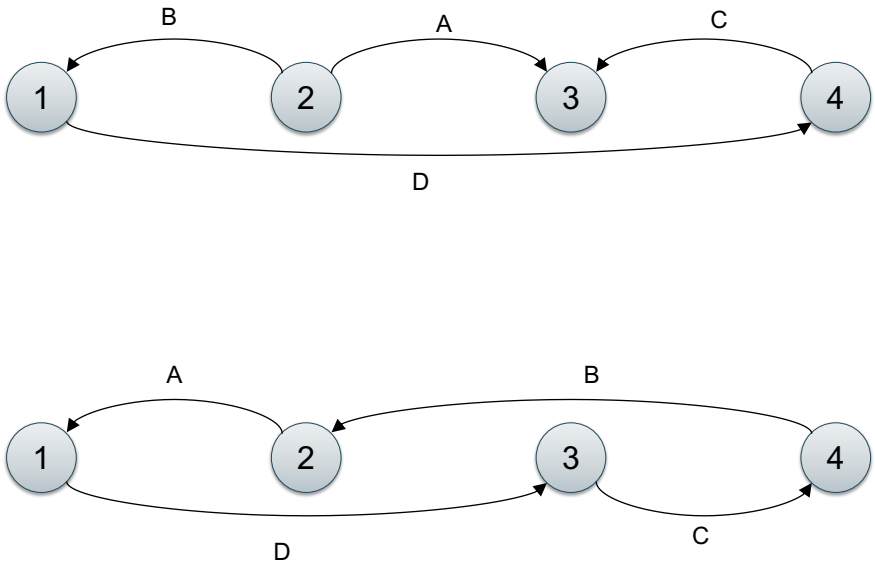
Solution: Conflict serializable, in the unique order $2, 1, 4, 3$.

ii.

$$R_2(A), R_1(D), W_4(B), R_3(C), W_3(D), R_2(B), W_4(C), W_1(A)$$

Solution: Not conflict serializable.

Solution: Both graphs:



(b) Consider a database about the happiness status of Alice and Bob. Each morning Alice and Bob wake up happy or sad. Today their morning status is:

```
A = happy          B = sad
```

Every day at noon we run these two transactions, at about the same time:

```
T1: START
    READ(A,x) - read Alice
    WRITE(B,x) - update Bob
    COMMIT
```

```
T2: START
    READ(B,y)
    if y='sad' then WRITE(A,'sad')
    COMMIT
```

Indicate which of the following outcomes are possible if we run the transactions (1) under the REPEATABLE READ isolation level[5], or (2) under the READ UNCOMMITTED isolation level. Write **yes** or **no** in each entry below.

[5]This is the same as SERIALIZABLE on our static database.

Alice	Bob	REPEATABLE READS	READ UNCOMMITTED
		Possible?	Possible?
sad	sad	**Solution:** yes	**Solution:** yes
sad	happy	**Solution:** no	**Solution:** yes
happy	sad	**Solution:** no	**Solution:** no
happy	happy	**Solution:** yes	**Solution:** yes
DEADLOCK		**Solution:** yes	**Solution:** no

(c) For each of the following statements indicate whether it is true or false:

i. In a static database, every serializable schedule is conflict serializable.
True or false?

Solution: False

ii. In a dynamic database, every serializable schedule is conflict serializable.
True or false?

Solution: False

iii. In a static database, every conflict serializable schedule is serializable.
True or false?

Solution: True

iv. In a dynamic database, every conflict serializable schedule is serializable.
True or false?

Solution: False

v. The two-phase locking protocol guarantees conflict serializability.
True or false?

Solution: True

vi. The strict two-phase locking protocol guarantees conflict serializability.
True or false?

Solution: True

vii. Deadlocks can occur under the READ UNCOMMITTED isolation level.
True or false?

Solution: True

viii. Deadlocks can occur under the REPEATABLE READ isolation level.
True or false?

Solution: True

ix. Two transactions can hold the same SHARED LOCK at the the same time.
True or false?

Solution: True

x. Two transactions can hold the same EXCLUSIVE LOCK at the the same time.
True or false?

Solution: False

201. *(from CSE 544p, Spring 2021, Final)*

(a) Consider a concurrency control manager by timestamps (without multiversion). Below are several schedules: $\mathtt{st}_i$ means that transaction T_i starts, and $\mathtt{co}_i$ means T_i commits. In each case below, say what happens with the last request. You have to choose between one of the following four possible answers:

1. the request is accepted,
2. the request is ignored,
3. the transaction is delayed (i.e. transaction is put on wait),
4. the transaction is rolled back.

i. $\mathtt{st}_1; \mathtt{st}_2; \mathtt{st}_3; \mathtt{r}_1(A); \mathtt{w}_3(A); \mathtt{co}_3; \mathtt{r}_2(A);$
What will the system do in response to the request $\mathtt{r}_2(A)$?

Solution: r2(A): T2 is rolled back.

ii. $\mathtt{st}_1; \mathtt{st}_2; \mathtt{r}_2(A); \mathtt{co}_2; \mathtt{r}_1(A); \mathtt{w}_1(A);$
What will the system do in response to the request $\mathtt{w}_1(A)$?

Solution: w1(A): T1 is rolled back.

iii. $\mathtt{st}_1; \mathtt{st}_2; \mathtt{r}_1(A); \mathtt{r}_2(A); \mathtt{w}_1(B); \mathtt{w}_2(B);$
What will the system do in response to the request $\mathtt{w}_2(B)$?

Solution: w2(B): accepted.

iv. $\mathtt{st}_1; \mathtt{st}_2; \mathtt{st}_3; \mathtt{r}_1(A); \mathtt{w}_3(A); \mathtt{co}_3; \mathtt{r}_2(B); \mathtt{w}_2(A);$
What will the system do in response to the request $\mathtt{w}_2(A)$?

Solution: w2(A): ignored.

v. $\mathtt{st}_1; \mathtt{st}_2; \mathtt{st}_3; \mathtt{r}_1(A); \mathtt{w}_1(A); \mathtt{r}_2(A);$
What will the system do in response to the request $\mathtt{r}_2(A)$?

Solution: r2(A): T2 is delayed.

(b) We have a database of airports and flights between them. Each airport decides whether they are going to support the Daylight Savings Time policy. The schema is:

```
Airport(code, dst)
Flight(c1,c2)
```

Both `Flight.c1` and `Flight.c2` are foreign keys to `Airport`. Some airports decide

to set their DST according to the majority of their connecting airports. They run the procedure below, which sets the DST policy to 'Y' iff the majority of their neighbors have the DST policy set to 'Y':

```
Set_DST(%C)
  START TRANSACTION
    cy = select count(*) from Flight x, Airport y
         where x.c1 = %C and x.c2 = y.code and y.dst = 'Y'

    cn = select count(*) from Flight x, Airport y
         where x.c1 = %C and x.c2 = y.code and y.dst = 'N'

    if cy > cn   // majority have DST = YES?
      then // set own DST = YES
           update Airport set dst = 'Y' where code = %C
      else // set own DST = NO
           update Airport set dst = 'N' where code = %C
   END TRANSACTION
```

Assume that the database is static, i.e. no records are inserted or deleted.

i. Suppose there are only two airports in the system, call them ABC and DEF, and there are two flights, one from ABC to DEF and one from DEF to ABC. Initially their DST settings are the following:

ABC.DST = 'Y' DEF.DST = 'N'

Both airports run the SET_DST procedure at about the same time. Assuming that the scheduler allows only serializable schedules, which of the following four outcomes are possible? Check all that apply:

ABC.DST	DEF.DST	Possible? Y/N
'Y'	'Y'	
'Y'	'N'	
'N'	'Y'	
'N'	'N'	

Solution: Y/N/N/Y

ii. Assume now that the database system is running a concurrency control manager based on snapshot isolation (SI). Assuming that both airports run the SET_DST procedure at about the same time, which of the following four outcomes are possible? Check all that apply:

ABC	DEF	Possible? Y/N
'Y'	'Y'	
'Y'	'N'	
'N'	'Y'	
'N'	'N'	

Solution: Y/N/Y/Y

(c) For each of the statements below indicate whether it is true of false about the ARIES recovery manager:

i. During the normal operation of a database (i.e. not during recovery) no updates in the log are undone.
true or false ?

Solution: false

ii. During the normal operation of a database (i.e. not during recovery) no updates in the log are redone.
true or false ?

Solution: true

iii. During the normal operation of a database (i.e. not during recovery) no CLR records are ever written to the log.
true or false ?

Solution: false

iv. During recovery from a crash no log record is used both during the redo and during the undo phase. (In other words, a log record is either redone or undone, but not both.)
true or false ?

Solution: true

v. CLR log records are never used during recovery (in other words they serve a different purpose, not for recovery).
true or false ?

Solution: false

vi. Suppose that a transaction performed two updates to the database, resulting in two log entries, then the system crashed, and crashed repeatedly during recovery. No matter how often the system crashes again, the log never contains more than two CLR records for this transaction.
true or false ?

Solution: true

vii. All log entries preceding the last `checkpoint` can be safely deleted from the log in order to reclaim their disc space.
true or false ?

Solution: false

viii. During recovery, the first undo log entry always precedes the first redo log entry.
true or false ?

Solution: false

ix. There is a separate ARIES log file for every user of the database.
true or false ?

Solution: false

x. The log may contain more than one `checkpoint`.
true or false ?

Solution: true

202. *(from CSE 414, Spring 2024, Final; CSE 344, Spring 2026, Makeup Final)*

(a) For each schedule below draw the conflict graph and indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule. You have to turn in a graph and to indicate whether the schedule is conflict serializable.

i.

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), R_1(C), W_4(C)$$

Write your answer here:

Solution: Conflict serializable, in the unique order $1, 3, 2, 4$
TO DRAW THE GRAPH.

ii.

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), W_4(C), R_1(C)$$

Write your answer here:

Solution: Not conflict serializable.
TO DRAW THE GRAPH

Solution: Yes

Solution: No

Solution: No

(b) Consider the following three transactions:

$$T_1 : W_1(A)$$
$$T_2 : R_2(A), W_2(B)$$
$$T_3 : R_3(B)$$

Give an example of a schedule with the following properties:

1. The schedule is serializable;
2. $W_1(A)$ occurs before $R_3(B)$ in the schedule;
3. The only serialization order is T_3, T_2, T_1.

For example, if your answer is:

$$W_1(A), R_2(A), W_2(B), R_3(B)$$

then it satisfies conditions 1 and 2, but it does not satisfy condition 3, because its serialization order is T_1, T_2, T_3 (in fact, it is already serial). Your answer should satisfy all three conditions.

Write your answer here:

Solution:

$$R_2(A), W_1(A), R_3(B), W_2(B)$$

(c) The SQL standard defines three *weak isolation levels*: dirty reads, read committed, and repeatable reads. They are defined as follows (this is the same definition as in the lectures):

- *Dirty reads* means no read locks, strict 2PL write locks.
- *Read committed* means short-duration read locks, strict 2PL write locks..

- *Repeatable reads* means strict 2PL read and write locks.

Consider two transactions T_1, T_2. For each schedule below, indicate under which isolation level that schedule is possible:

	Time →							
Schedule$_1$:	St_1,		$R_1(A)$,			$R_1(B)$,		$W_1(A)$, Co_1
		St_2,		$W_2(A)$,	$W_2(B)$,		Co_2	
Schedule$_2$:	St_1,		$R_1(A)$,				$R_1(B)$,	$W_1(A)$, Co_1
		St_2,		$W_2(A)$,	$W_2(B)$,	Co_2		
Schedule$_3$:	St_1,			$R_1(A)$,	$R_1(B)$,			$W_1(A)$, Co_1
		St_2,	$W_2(A)$,			$W_2(B)$,	Co_2	
Schedule$_4$:	St_1,			$R_1(A)$,	$R_1(B)$,	$W_1(A)$,	Co_1	
		St_2,	$W_2(A)$,					$W_2(B)$, Co_2

Indicate for each schedule whether it is possible under each isolation level, by writing *yes* or *no* in the table below:

	Dirty reads	Read committed	Repeatable Reads
Schedule$_1$			
Schedule$_2$			
Schedule$_3$			
Schedule$_4$			

Solution:

	Dirty reads	Read committed	Repeatable Reads
$\texttt{Schedule}_1$	yes	no	no
$\texttt{Schedule}_2$	yes	yes	no
$\texttt{Schedule}_3$	yes	no	no
$\texttt{Schedule}_4$	no	no	no

(d) For each of the following statements indicate whether it is true or false:

i. If there are an odd number of transactions, then deadlock can never occur. True or false?

Solution: False

ii. In a static database, every serializable schedule is conflict serializable. True or false?

Solution: False

iii. In a dynamic database, every serializable schedule is conflict serializable. True or false?

Solution: False

iv. In a static database, every conflict serializable schedule is serializable. True or false?

Solution: True

v. In a dynamic database, every conflict serializable schedule is serializable. True or false?

Solution: False

vi. Transactions were introduced in order ensure database consistency.

Yes or No?

Solution: Yes

vii. Transactions were introduced in order to improve performance.
Yes or No?

Solution: No

viii. A concurrency management system uses strict 2PL, with shared locks for reads and exclusive locks for writes. If all transactions are read-only, then deadlock is not possible.
True or false?

Solution: True

ix. An OLAP workload ("Online analytical processing") means a workload consisting of simple queries and many updates.
True or false?

Solution: False

x. An OLTP workload ("Online transaction processing") means a workload consisting of simple queries and many updates.
True or false?

Solution: True

203. *(from CSE 344, Autumn 2024, Final; CSE 344, Spring 2026, Final)*

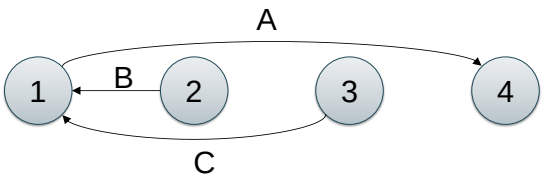
(a) For each of the schedules below construct the precedence graph and indicate whether the schedule is conflict serializable or not.

We underlined the difference from the previous schedule, to help you read them.

i.

$R_1(A), R_2(D), W_2(B), R_4(D), R_3(D), W_1(A), R_1(B), W_3(C), R_4(A), R_1(C)$

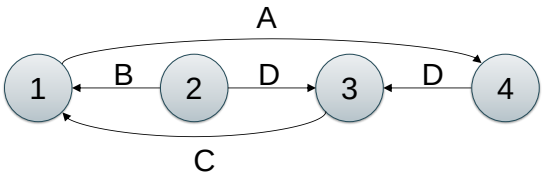
Solution:



ii.

$R_1(A), R_2(D), W_2(B), R_4(D), \underline{W_3(D)}, W_1(A), R_1(B), W_3(C), R_4(A), R_1(C)$

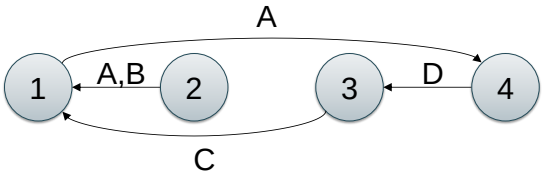
Solution:



iii.

$R_1(A), \underline{R_2(A)}, W_2(B), R_4(D), W_3(D), W_1(A), R_1(B), W_3(C), R_4(A), R_1(C)$

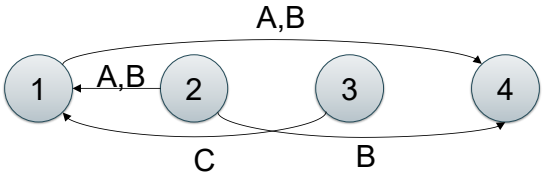
Solution:



iv.

$R_1(A), R_2(A), W_2(B), \underline{R_4(B)}, W_3(D), W_1(A), R_1(B), W_3(C), R_4(A), R_1(C)$

Solution:



(b) The initial values of the elements A, B are: $A = 100, B = 1$. We run transactions T_1, T_2 below:

T_1	T_2
BEGIN	BEGIN
READ(A, x)	READ(A, u)
READ(B, y)	READ(B, v)
$x = x + 100$	$w = u + v$
$y = y + 1$	WRITE(C, w)
WRITE(A, x)	COMMIT
WRITE(B, y)	
COMMIT	

Obviously, the final values of A, B are $A = 200, B = 2$, but the final value of the element C depends on how these two transactions are scheduled. For each of the isolation levels below indicate the possible values of C. For example, you may say that C can take the values 205 or 301 (not a real answer).

Hint: there are only a few possible schedules. Write them down, then check for each if it is possible under each isolation level.

i. ISOLATION LEVEL UNCOMMITTED.
Possible values of C?

Solution: $C = 101, 102, 201, 202$

ii. ISOLATION LEVEL COMMITTED.
Possible values of C?

Solution: $C = 101, 102, 202$

iii. ISOLATION LEVEL REPEATABLE READS.
Possible values of C?

Solution: $C = 101, 202$

Solution:
We only need to consider these schedules:

T_1	T_2
	$R_2(A)$
	$R_2(B)$
$W_1(A)$	
$W_1(B)$	

T_1	T_2
	$R_2(A)$
$W_1(A)$	
$W_1(B)$	
	$R_2(B)$

T_1	T_2
$W_1(A)$	
	$R_2(A)$
	$R_2(B)$
$W_1(B)$	

T_1	T_2
$W_1(A)$	
$W_1(B)$	
	$R_2(A)$
	$R_2(B)$

The outcomes of the four schedules above are:
$C = 101$, $C = 102$, $C = 201$, $C = 202$
ISOLATION LEVEL UNCOMMITTED. possible schedules: 1, 2, 3, 4
ISOLATION LEVEL COMMITTED. Possible schedules 1, 2, 4
ISOLATION LEVEL REPEATABLE READS. Possible schedules 1, 4

17 Semistructured Data: XML, JSON, SQL++ (204–239)

The second largest section of the book, and the one that shows the passage of time most clearly. The early exams ask about XML, DTDs, XPath and XQuery; the later ones ask the same questions about JSON and SQL++. The underlying subject does not change: data that is nested and irregular, without a schema fixed in advance.

The XPath questions ask what a path expression returns, and most of the difficulty lies in the difference between `/` and `//` and in what a predicate is evaluated against. The XQuery questions ask for a transformation, usually restructuring one document into another or turning relational data into nested form; there the FLWOR expression is the tool, and grouping is done by iterating over `distinct-values` of a key. Several problems ask you to design a DTD, or to compare relational encodings of XML data.

Read the sample document carefully before starting: several questions turn on an element that happens to be optional or repeated.

204. *(from CSE 444, Autumn 2000, Midterm)*

(a) Consider the following DTD:

```
<!ELEMENT memories  (party*)>
<!ELEMENT party     (date, (person* | everybody))>
<!ELEMENT person    (name, (phone|email)?)>
<!ELEMENT date      (#PCDATA)>
<!ELEMENT name      (#PCDATA)>
<!ELEMENT phone     (#PCDATA)>
<!ELEMENT email     (#PCDATA)>
<!ELEMENT everybody (#PCDATA)>
```

The XML document below is not valid for this DTD. Add new elements and/or delete existing elements to make it valid for this DTD.

```
<memories>
  <party> <date>1999</date>
    <person> <name>John</name>
             <email>j@partygoers.org</email>
    </person>
    <person> <name>Jim</name>
    </person>
    <person> <name>Mary</name>
             <phone>555-1234</phone>
             <email>m@partygoers.org</email>
    </person>
  </party>
  <party> <date>2000</date>
          <everybody> </everybody>
  </party>
  <party> <person><name>John</name>
                  <phone>555-2345</phone>
          </person>
          <everybody> </everybody>
  </party>
</memories>
```

Solution: `Jim` needs a `phone` or `email` child removed from the requirement — as written `person` allows an optional `(phone|email)`, so `Jim` is already valid; `Mary` has *both* a `phone` and an `email`, so one must be deleted. The third `party` is missing its `date`, so a `<date>2001</date>` must be added; and a `party` may contain either `person*` or `everybody` but not both, so the `<everybody>` element must be deleted from it.

205. *(from CSE 544, 2001, Final)*

(a) Consider the following XML document:

```
<db>
   <a> <b> 1 </b>
       <c> 2 </c>
   </a>
   <a> <b> 3 </b>
       <c> <b> 4 </b> </c>
       <c> <b> 5 </b> </c>
   </a>
</db>
```

What do the following XPath expressions return?

i. `/db//b`

Solution:
`<b> 1 </b>` `<b> 3 </b>` `<b> 4 </b>` `<b> 5 </b>`

ii. `/db//c/b`

Solution:
`<b> 4 </b>` `<b> 5 </b>`

iii. `/db//c[b]`

Solution:
`<c> <b> 4 </b> </c>` `<c> <b> 5 </b> </c>`

(b) We run a few XPath queries on an XML document, and obtain the following results:

`/db/a`	returns 10 elements
`/db/a/*/*/b`	returns 20 elements
`/db/a//*/b`	returns 100 elements
`/db/a/*`	returns 55 elements
`/db/a//*`	returns 344 elements

For each of the expressions below indicate your best guess of the number of elements returned. For example you may answer "at least 37" or "at most 37" or "between 37 and 73"; if there is really nothing useful you can say, then answer "at least 0". Recall that `a//*` means all strict descendants of `a`, i.e. not `a` itself.

i. `/db/a/c`

Solution: ≤ 55

ii. `/db/a[//b]`

Solution: ≥ 1 and ≤ 10

iii. `/db/a//*/*/b`

Solution: ≥ 20 and ≤ 100. (The intent was to ask about `/db/a//*/*/*/b`, but one star got lost during typing; that answer is more interesting: $= 80$.)

206. *(from CSE 444, Autumn 2001, Midterm)*

Consider the following relational schema about courses, homeworks for courses, students, and a relationship between students and homeworks:

```
Course(id, name)
HomeWorks(courseId, hwNumber, points)
Turnins(courseId, hwNumber, studentId, grade)
Students(id, name)
```

In each of the cases below, design a DTD for exporting the information to an XML format that includes information about course names, homework numbers and points, student names, and the grade each student got for every homework. It should not include information about course id's and student id's.

(a) The hierarchy is: courses (top level), homeworks, turnins, students.

Solution:

```
<!ELEMENT course   (name, homework*)>
<!ELEMENT homework (number, points, turnin*)>
<!ELEMENT turnin   (student, grade)>
<!ELEMENT student  (name)>
<!ELEMENT name     (#PCDATA)>
<!ELEMENT number   (#PCDATA)>
<!ELEMENT grade    (#PCDATA)>
<!ELEMENT points   (#PCDATA)>
```

(b) The hierarchy is: students (top level), turnins, homeworks, courses.

Solution:

```
<!ELEMENT student  (name, turnin*)>
<!ELEMENT turnin   (grade, homework)>
<!ELEMENT homework (course, number, points)>
<!ELEMENT course   (name)>
```

(with the same `#PCDATA` declarations as above).

207. *(from CSE 544, 2002, Final)*

Consider the following relational schema:

```
Product(name, manufacturer, city)
```

The following XQuery specification defines an XML view over the relational schema. Here `$db` denotes a canonical relational view of the relational schema:

```
<sales-by-city>
  let $c-all := distinct-values($db/Product/tuple/city/text())
  for $c in $c-all
  return
    <city>
       <name> { $c } </name>
       { let $m-all := distinct-values(
             $db/Product/tuple[city/text()=$c]/manufacturer/text())
         for $m in $m-all
         return
           <manufacturer>
              <name> { $m } </name>
              { for $p in $db/Product/tuple[city/text()=$c]
                                          [manufacturer/text()=$m]
                return <product> { $p/name/text() } </product>
              }
           </manufacturer>
       }
    </city>
</sales-by-city>
```

(a) Write the DTD of the XML view.

Solution:

```
<!ELEMENT sales-by-city (city*)>
<!ELEMENT city          (name, manufacturer*)>
<!ELEMENT manufacturer  (name, product*)>
<!ELEMENT product       (#PCDATA)>
```

(b) Translate the following XPath queries into SQL. Your SQL queries have to return the same number of occurrences, i.e. if "Gizmo Plus" occurs three times in the result of the second XPath expression, then it must also occur three times in the SQL's answer. The order in your answer is irrelevant.

i. `/sales-by-city/city`

Solution:

```
select distinct p.city
from Product p
```

ii. `/sales-by-city/city[name/text()="Seattle"]//product/text()`

Solution:
```
select p.name
from Product p
where p.city = 'Seattle'
```

iii. `/sales-by-city/city/manufacturer/name/text()`

Solution:
```
select manufacturer
from (select distinct manufacturer, city from Product) as t
```

208. *(from CSE 444, Autumn 2002, Midterm; CSE 544, 2002, Final)*

We have the following DTD:

```
<!ELEMENT vacation (region*)>
<!ELEMENT region   (name, package*)>
<!ELEMENT package  (hotel, airline, month, price)>
```

All other elements are assumed to be `#PCDATA`.

(a) Write XPath expressions to return the following. It is OK to include duplicates in the answers (XPath does not eliminate duplicates).

i. Return all `package` elements that use "Northwest Airlines" as airline and whose price is under $2000. Assume that prices are expressed as integers, i.e. a test like `<= 2000` should work.

Solution:
```
/vacation/region/package[airline/text()="Northwest Airlines"]
                        [price/text() <= 2000]
```

ii. Return all `hotel` elements in the "North West" region that are available in "December".

Solution:
```
/vacation/region[name/text()="North West"]
        /package[month/text()="December"]/hotel
```

(b) Write an XQuery expression that translates the document into the following structure:

```
<!ELEMENT promotion (hotel*)>
<!ELEMENT hotel     (name, package*)>
<!ELEMENT package   (region, airline, month, price)>
```

Your result should have a single entry for each hotel, even if that hotel occurs under several packages, or under several regions in the input.

Solution:

```
<promotion>
{ FOR $h in distinct(//hotel/text())
  RETURN
    <hotel>
      <name> { $h } </name>
      { FOR $r in /vacation/region,
            $p in $r/package[hotel=$h]
        RETURN
          <package>
            <region>  { $r/name/text() }  </region>
            <airline> { $p/airline/text() } </airline>
            <month>   { $p/month/text() }   </month>
            <price>   { $p/price/text() }   </price>
          </package>
      }
    </hotel>
}
</promotion>
```

209. *(from CSE 444, Autumn 2003, Final; CSE 544, 2004, Final)*

Consider an XML document describing colleges and departments at a university. Each college has a name, a dean, and a set of departments. Each department has a name, a chair, and a list of students that are registered in that department. An example is:

```
<university>
  <college>
    <name> Engineering </name>
    <dean> Denton </dean>
    <department>
      <name> Computer Science </name>
      <chair> Notkin </chair>
      <student> John Reader </student>
      <student> Joe Writer </student>
      . . . . . .
    </department>
    <department>
      <name> Electrical Engineering </name>
      . . . .
    </department>
    . . . . . .
  </college>
  <college>  (another college) . . . . . . </college>
  . . . . .
</university>
```

(a) Write an XPath expression that returns all students in the Computer Science Department.

Solution:

```
/university/college/department[name/text()="Computer Science"]/student
```

(b) Write an XQuery expression that returns for each student the number of colleges in which they appear.

Solution:

```
let $ss := distinct-values(/university/college/department/student/text())
for $s in $ss
return
  <student> <name> { $s } </name>
            <no_depts> {
              count(/university/college[department/student/text()=$s]) }
            </no_depts>
  </student>
```

(c) Design a relational schema that can represent the same data. Indicate the keys and the foreign keys in that schema.

Solution:

```
college(cid, name, dean)              key: cid
department(did, cid, name, chair)     key: did
                                      foreign key: cid to college.cid
student(sid, did, name)               key: sid
                                      foreign key: did to department.did
```

210. *(from CSE 444, Autumn 2003, Midterm)*

Consider the following DTD:

```
<!ELEMENT db       (website*)>
<!ELEMENT website  (url, document*)>
<!ELEMENT document (author, (href | word)*)>
```

All unspecified elements are assumed to be `#PCDATA`. The data describes several websites, where each website consists of a URL and a set of documents. The documents have an author, and a list of words and/or links to other urls (`href`'s).

(a) Write an XPath or XQuery expression that returns all URLs that contain documents authored by "John Smith".

Solution:

```
/db/website[document/author/text()="John Smith"]/url
```

(b) Write an XPath or XQuery expression that returns all authors who used more than 5000 distinct words in all the documents they created on some website. Notice that the 5000 words have to occur all in the documents of the same website, not in all documents of that author.

Solution:

```
for $r in /db
let $a = distinct-values(
     for $w in $r/website,
         $x in $w/document/author/text()
     where count(distinct-values(
                 $w/document[author/text()=$x]/word/text())) > 5000
     return $x
     )
for $z in $a
return <author> { $z } </author>
```

(c) Design a relational schema that can store the same information and write the corresponding `CREATE TABLE` statements. Indicate the keys, foreign keys, and whether the attributes are nullable or not. Choose reasonable types for the attributes. You may assume that URLs are unique (i.e. they are keys for websites).

Solution:

```
CREATE TABLE WEBSITE (
     URL varchar(50) PRIMARY KEY );

CREATE TABLE DOCUMENT (
     DOCID varchar(50) PRIMARY KEY,
     URL varchar(50) REFERENCES WEBSITE(URL),
     AUTHOR varchar(50) );

CREATE TABLE WORD (
     DOCID varchar(50) REFERENCES DOCUMENT(DOCID),
     WORD varchar(100) );

CREATE TABLE HREF (
     DOCID varchar(50) REFERENCES DOCUMENT(DOCID),
     URL varchar(100) );
```

211. *(from CSE 444, Autumn 2004, Midterm; CSE 444, Autumn 2005, Midterm)*

Consider the following XML document:

```
<a>
   <a>
      <a> 1 </a>
      <a> <a> 2 </a>
          <a> 3 </a>
      </a>
   </a>
   <a>
      <a> 4 </a>
      <a> <a> 5 </a>
          <a> 6 </a>
      </a>
   </a>
</a>
```

For each of the XPath expressions below indicate what they return. For example, if the expression is `/a/a/a/a/text()` then you would answer `2 3 5 6`; and if the expression is `//a[a/text() = 8]/a/text()` you would answer EMPTY, since the value 8 is not in the document.

(a) `//a/text()`

Solution: `1 2 3 4 5 6`

(b) `//a[a/text()='2']/a/text()`

Solution: `2 3`

(c) `/a/a[.//text()='1']//a/text()`

Solution: `1 2 3`

(d) `//a[a/text()="2"][a/text()="3"]/a/text()`

Solution:
`\texttt{2 3}`

(e) `//a[.//a/text()="3"][.//a/text()="4"]//a/text()`

Solution: EMPTY

(f)
`//a[.//a/text()="1"]//a[.//a/text()="4"]//a[.//a/text()="5"]//a/text()`

Solution: `5 6`

212. *(from CSE 444, Winter 2005, Final)*

Consider an XML document containing information about job postings and job applications. The postings are grouped by company, and applications are listed under postings. An application contains only the applicant's name. For example:

```
<jobs>
    <company>
        <name> MicroScience Corp. </name>
        <posting>
            <jobtitle> sales rep </jobtitle>
            <salary> 30000 </salary>
            <application> Mark </application>
            <application> Lucy </application>
            <application> Frank </application>
        </posting>
        <posting>
            <jobtitle> technology consultant </jobtitle>
            <salary> 80000 </salary>
            <application> Lucy </application>
            <application> Fred </application>
            <application> Mark </application>
            <application> Betty </application>
        </posting>
    </company>
    <company>
        <name> MacroConsulting Inc. </name>
        <posting>
            <jobtitle> technology consultant </jobtitle>
            <salary> 40000 </salary>
            <application> Frank </application>
        </posting>
        <posting>
            <jobtitle> debugger </jobtitle>
            <salary> 20000 </salary>
        </posting>
        <posting>
            <jobtitle> programmer analyst </jobtitle>
            <salary> 35000 </salary>
            <application> Lucy </application>
            <application> Mark </application>
        </posting>
    </company>
</jobs>
```

(a) For each of the XPath expressions below, indicate how many answers it will return. For example, if the XPath expression is:

```
/jobs/company[name/text()='MacroConsulting Inc.']/posting
```

then you would answer 3.

i. `//application`

> **Solution:** 10

ii. `/jobs/company/posting[salary/text()>60000]/application`

> **Solution:** 4 (only applications for the 2nd posting of the 1st company)

iii. `/jobs/company[posting/salary/text()>60000]//application`

> **Solution:** 7 (all applications of the 1st company)

(b) Write an XQuery expression that returns the names of all applicants who submitted applications to at least two companies. Your query should return a list of `<name>` elements, and each element should be included only once. For example, on the XML document shown above it would return:

```
<name> Mark  </name>
<name> Lucy  </name>
<name> Frank </name>
```

> **Solution:** Group the applications by applicant name, and keep the names for which the applications come from at least two distinct companies:
>
> ```
> for $n in distinct-values(//application/text())
> let $c := /jobs/company[posting/application/text() = $n]
> where count($c) >= 2
> return <name> { $n } </name>
> ```
>
> The comparison `posting/application/text() = $n` is an existential comparison: it holds for a company as soon as *some* application in *some* of its postings carries the name `$n`. Since each company element is counted at most once, `count($c)` is exactly the number of distinct companies the applicant applied to. On the document above this returns Mark, Lucy and Frank; Fred and Betty applied only to MicroScience Corp.

213. *(from CSE 444, Autumn 2005, Final)*

Consider an XML document with the following DTD:

```
<!ELEMENT root     (category*)>
<!ELEMENT category (name, prod*)>
<!ELEMENT prod     (name, price, store*)>
<!ELEMENT store    (name, city)>
```

All elements that are not indicated above have content `#PCDATA`.

(a) Write an XQuery query that computes for each city the total number of product categories that are sold at stores in that city.

Solution:

```
for $c in distinct-values(//store/city/text())
let $cats := /root/category[prod/store/city/text() = $c]
return
  <city>
    <name> { $c } </name>
    <count> { count($cats) } </count>
  </city>
```

A category is counted once for a given city no matter how many of its products are sold there, or in how many stores, because the predicate filters `category` elements rather than `prod` or `store` elements.

214. *(from CSE 544, 2006, Final)*

Consider the following XML document:

```
<db>
   <a> <b> 1 </b>
       <c> 2 </c>
   </a>
   <a> <b> 3 </b>
       <c> <b> 4 </b> </c>
       <c> <b> 5 </b> </c>
   </a>
</db>
```

(a) What do the following XPath expressions return? In each case you have to turn in an XML fragment.

i. `/db//b`

Solution:

```
<b> 1 </b> <b> 3 </b> <b> 4 </b> <b> 5 </b>
```

ii. `/db//c/b`

Solution:
 4   5

iii. `/db//c[b]`

Solution:
<c> 4 </c> <c> 5 </c>

(b) On some XML document, the expression `/db/a` returns 10 elements, `/db/a/*/*/b` returns 20 elements, and the expression `/db/a//*/b` returns 100 elements. For each of the expressions below indicate a lower and/or an upper bound on the number of elements returned. For example you may answer "between 10 and 15", or "at least 37", or "at most 37" or, if there is really nothing useful you can say, then answer "at least 0".

i. `/db/a/c`

Solution: at least 0

ii. `/db/a[//b]`

Solution: at least 1

iii. `/db/a//*/*/b`

Solution: at least 80

215. *(from CSE 444, Winter 2006, Final)*

Consider the following XML data about Webpages:

```
<root>
  <webpage>
     <url> .... </url>
     <title> ... </title>
     <author> .... </author>
     <author> .... </author>
     <author> .... </author>
     ...
     <words>
        <word> ... </word>
        <word> ... </word>
        . . .
     </words>
  </webpage>
```

```
  <webpage> . . . </webpage>
  . . .
</root>
```

The document consists of several (0 or more) webpages; each webpage has one url, an optional title, several (0 or more) authors, and one `words` element: the latter lists all distinct words in the document (in some arbitrary order).

(a) Write the DTD for this type of XML data.

Solution:

```
<!ELEMENT root  (webpage*)>
<!ELEMENT webpage  (url, title, author*, words)>
<!ELEMENT words  (word*)>
```

(b) Design a relational database that stores the same data as the XML document. You have to turn in some relation names and attributes, together with an indication of their keys and foreign keys. For example:

```
PRODUCT(pid, name)
ORDER(oid, pid, date)     pid is foreign key in PRODUCT
```

Solution:

```
webpage(url, title)
author(url, name)  -- each author authored a single page
word(url, wordOccurrence) -- each word occurrence is a string
```

(c) Write an XQuery expression that transforms the relational data to the XML document above. You will assume that the XQuery sees the relational data as a canonical XML representation, e.g. like below:

```
<data>
  <product> <row> <pid>...</pid><name>..</name></row>
            <row> <pid>...</pid><name>..</name></row>
            ...
  </product>
  <order>....</order>
</data>
```

Solution: The XML representation of the relational data is:

```
<db>
  <webpage> <url>...</url> <title>...</title> </webpage>
  ...
  <webpage> <url>...</url> <title>...</title> </webpage>
  <author> <url> ... </url> <name>...</name> </author>
  ...
  <author> <url> ... </url> <name>...</name> </author>
  <word> <url> ... </url> <wordOccurrence> ... </wordOccurrence>
  ...
  <word> <url> ... </url> <wordOccurrence> ... </wordOccurrence>
</db>
```

The XQuery that converts the relational data into the original XML form is:

```
<root>
{
  for $p in /db/webpage
  let $u  := $p/url/text()
  let $ws := /db/word[url/text() = $u]/wordOccurrence/text()
  return
    <webpage>
      { $p/url }
      { $p/title }
      { for $a in /db/author[url/text() = $u]
        return <author> { $a/name/text() } </author> }
      <words>
        { for $w in distinct-values($ws)
          return <word> { $w } </word> }
      </words>
    </webpage>
}
</root>
```

The two inner loops perform the joins that the relational design replaced the nesting with: `author` and `word` rows are matched back to their page on `url`. Copying `$p/title` directly is what makes the title optional — if the row has no `title` the expression returns the empty sequence and nothing is inserted. The `distinct-values` restores the requirement that `words` list each word only once.

(d) Write an XQuery expression that takes the initial XML document and computes for each word the number of authors that use that word in their vocabulary. Your XQuery should return an answer like this:

```
<answer>
  <word> <text>...</text> <count>...</count> </word>
  <word> <text>...</text> <count>...</count> </word>
  ....
</answer>
```

Solution:

```
<answer>
{
  for $w in distinct-values(//words/word/text())
  let $a := distinct-values(
        /root/webpage[words/word/text() = $w]/author/text())
  return
    <word>
      <text> { $w } </text>
      <count> { count($a) } </count>
    </word>
}
</answer>
```

An author who uses the same word on several of their pages must be counted once, which is what the inner `distinct-values` ensures; without it the query would count author *occurrences* rather than authors. Note that pages with no authors contribute nothing to any count.

216. *(from CSE 444, Winter 2006, Midterm)*

Consider the following XML document:

```
<a>
   <a>
      <a> 1 </a>
      <a> <a> 2 </a>
          <a> 3 </a>
      </a>
   </a>
   <a>
      <a> 4 </a>
      <a> <a> 5 </a>
          <a> 6 </a>
      </a>
   </a>
</a>
```

For each of the XPath expressions below indicate what they return. Answer EMPTY if the expression returns nothing.

(a) `//a/text()`

Solution: `1 2 3 4 5 6`

(b) `/a/a/a/a/text()`

Solution: `2 3 5 6`

(c) `/a[.//a/text()="2"]/a/a/text()`

Solution: `1 4`

(d) `//a[a/text()="2"][a/text()="3"]/a/text()`

Solution: `2 3`

(e) `//a[.//a/text()="3"][.//a/text()="4"]/a/a/text()`

Solution: `1 4`

(f) `/a/a[2]/a[2]/a[1]`

Solution: `5` (or `<a> 5 </a>`)

217. *(from CSE 444, Autumn 2004, Final; CSE 444, Autumn 2005, Final; CSE 444, Autumn 2006, Final)*

Consider an XML document with the following DTD:

```
<!ELEMENT root     (category*)>
<!ELEMENT category (name, prod*)>
<!ELEMENT prod     (name, price, store*)>
<!ELEMENT store    (name, city)>
```

All elements that are not indicated above have content `#PCDATA`.

(a) Write an XPath expression that returns all products under the "toy" category, whose price is < 200 and which are sold at some store in "New York City".

Solution:

```
/root/category[name='toy']/prod[price < 200][store/city = 'NYC']
```

(b) Write an XQuery that restructures the data into the following DTD:

```
<!ELEMENT root (store*)>
<!ELEMENT store (name, city, product*)>
<!ELEMENT product (name, price, category)>
```

All elements that are not indicated above have content `#PCDATA`. Each store in the input XML document is uniquely identified by its name, and should occur exactly once in the output document.

Solution:

```
<output>
{
  for $r in /root
  let $s := distinct-values($r//store/name/text())
  for $x in $s
  return <store>
           <name> { $x } </name>
           <city> { distinct-values($r//store[name = $x]/city/text()) } </city>
           {
             for $c in $r/category,
                 $p in $c/prod[store/name = $x]
             return <product>
                      { $p/name, $p/price,
                        <category> { $c/name/text() } </category> }
                    </product>
           }
         </store>
}
</output>
```

(c) Consider the three XQuery expressions below:

```
Q1:
for $x in document()/root/category/prod[price/text()>5]
                                       [store/city/text()="NYC"]
return $x/name

Q2:
for $c in document()/root/category,
    $x in $c/prod[price/text() > 5][store/city/text()="NYC"]
return $x/name

Q3:
for $x in document()/root/category/prod,
    $c in $x/store/city
where $x/price/text() > 5 and $c/text()="NYC"
return $x/name
```

Indicate which queries are equivalent and which are different.

Solution: `Q1` and `Q2` are equivalent.
`Q2` and `Q3` are not equivalent.
`Q1` and `Q3` are not equivalent.

218. *(from CSE 444, Autumn 2008, Final)*

Consider the XML document below:

```
<aa>
  <bb>
    <cc> 1 </cc>
    <dd> 2 </dd>
    <bb> 3 </bb>
  </bb>
  <cc>
    <dd> 4 </dd>
    <cc> 5 </cc>
    <bb> 6 </bb>
  </cc>
  <cc>
    <dd> 7 </dd>
    <cc> 8 </cc>
  </cc>
</aa>
```

For each question below, write an XPath expression that returns precisely the values indicated. Your XPath expression should be restricted to navigation and qualifiers only: do not use numerical predicates. For example `//dd/text()` and `/*/*[bb]/*/text()` are acceptable answers but `//*[text()=4]/text()` is not.

(a) Return 3, 6

Solution: `//bb/text()`

(b) Return 4, 5, 6, 7, 8

Solution: `/aa/cc/*/text()`

(c) Return 4, 5, 6

Solution: `/aa/cc[bb]/*/text()`

219. *(from CSE 544p, Spring 2009, Final)*

Consider an XML instance having the following DTD:

```
<!ELEMENT doc (movie)*>
<!ELEMENT movie (title, year, actor*)>
<!ELEMENT actor (name, gender?)>
```

Movie titles are unique in the data. All `<gender>` elements are either `<gender> male </gender>` or `<gender> female </gender>`. Furthermore the actor's names are unique. That is, if `<name> Bacon </name>` occurs under different movies then it is the same actor, and all occurrences will have the same `<gender>`, or will miss the `<gender>` element. That is, we may find occurrences of the form:

```
<actor> <name> Bacon </name> <gender> male </gender> </actor>
```

or of the form:

```
<actor> <name> Bacon </name> </actor>
```

but the data will not contain

```
<actor> <name> Bacon </name> <gender> female </gender> </actor>
```

because in that case there would be two genders for the same actor.

(a) Write an XPath expression that computes all the movies where "Bacon" acted. Your expression should return an answer of the following form:

```
<title> a movie title here... </title>
<title> another title here... </title>
. . .
```

Note that you have to write an XPath expression, not an XQuery expression.

Solution:

```
/doc/movie[actor/name/text()='Bacon']/title
```

(b) Write an XQuery (or XPath) expression that computes the gender of the actor named "Subrahmanian". Your XQuery expression should return either the single element `<gender> male </gender>` or the single element `<gender> female </gender>` or nothing at all (if no gender information is found for Subrahmanian).

Solution:

```
for $d in document("file.xml")/doc
let $g = distinct-values($d/movie/actor[name/text()='Subrahmanian']/gender/text())
for $x in $g
return <gender> { $x } </gender>
```

(c) Write an XQuery expression that transforms the data into another data having the following DTD:

```
<!ELEMENT doc (actor*)>
<!ELEMENT actor (name, gender?, movie*)>
<!ELEMENT movie (title, year)>
```

Actors should occur uniquely, their gender should be listed only once, if it is available, and the actor element should include all movies that they acted in.

Solution:

```
<doc> {
   for $d in document("file.xml")/doc
   let $a = distinct-values($d/movie/actor/name/text())
   for $b in $a
   return <actor>
            <name> { $b } </name>
            { let $g = distinct-values($d/movie/actor[name/text()=$b]/gender/text())
              for $x in $g
              return <gender> { $x } </gender>
              for $m in $d/movie[actor/name/text()=$b]
              return <movie> { $m/title, $m/year } </movie>
            }
          </actor>
}
</doc>
```

220. *(from CSE 544p, Autumn 2010, Final)*

Consider an XML document with the following DTD:

```
<!ELEMENT root     (category*)>
<!ELEMENT category (name, prod*)>
<!ELEMENT prod     (name, price, store*)>
<!ELEMENT store    (name, city)>
```

All elements that are not indicated above have content `#PCDATA`.

(a) Write an XPath expression that returns all the names of all products under the category with name "toy", whose price is < 200 and that are sold at some store in "NYC". Your expression should return an answer of the following form:

```
<name> a product name here... </name>
<name> another name here...   </name>
. . .
```

Note that you have to write an XPath expression, not an XQuery expression.

Solution:

```
/root/category[name="toy"]/prod[price/text()<200][store/city/text()="NYC"]/name
```

(b) Write an XQuery query that computes for each city the total number of product categories that are sold at stores in that city. Your query should return an answer of the following form:

```
<answer> <city> city 1 here </city> <cnt> number here </cnt> </answer>
<answer> <city> city 2 here </city> <cnt> number here </cnt> </answer>
...
```

Solution:

```
let $r := doc()/root
let $cc := distinct-values($r/category/prod/store/city/text())
for $c in $cc
return
   <answer> <city> { $c } </city>
            <cnt> { count($r/category[prod/store/city/text()=$c]) } </cnt>
   </answer>
```

(c) If Q, Q' are two XPath queries, then we say that Q is *contained* in Q' if every answer returned by Q is also returned by Q'. For the four XPath queries below, determine which pairs of queries are contained.

```
Q1:
/root/category[prod[price/text()>10]]
              [prod[store[city/text()="NYC"]]
                   [store[city/text()="LA"]]]/name

Q2:
/root/category[prod[price/text()>10]
```

```
                    [store[city/text()="NYC"]]]
               [prod[store[city/text()="LA"]]]/name

Q3:
/root/category[prod[price/text()>10]]
               [prod[store[city/text()="NYC"]]]
               [prod[store[city/text()="LA"]]]/name

Q4:
/root/category[prod[price/text()>10]
                    [store[city/text()="NYC"]]
                    [store[city/text()="LA"]]]/name
```

(Answer: matrix).

Solution:

	Q1	Q2	Q3	Q4
Q1	y	-	y	-
Q2	-	y	y	-
Q3	-	-	y	-
Q4	y	y	y	y

221. *(from CSE 344, Spring 2011, Final)*

Consider an XML document that contains data about books. Consider the following two XPath queries on this XML data:

```
P1 = /bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()
P2 = /bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
```

In this problem you are asked to design a relational schema for this data, then translate the two XPath queries into SQL over that schema.

(a) Assume that the XML data conforms to the following DTD:

```
<!DOCTYPE bib [
 <!ELEMENT bib  (book* )>
 <!ELEMENT book  (title,  (author+ | editor+ ), publisher?, price )>
 <!ATTLIST book  year CDATA  #REQUIRED >
 <!ELEMENT author  (last, first )>
 <!ELEMENT editor  (last, first, affiliation )>
 <!ELEMENT title  (#PCDATA )>
 <!ELEMENT last  (#PCDATA )>
 <!ELEMENT first  (#PCDATA )>
 <!ELEMENT affiliation  (#PCDATA )>
 <!ELEMENT publisher  (#PCDATA )>
 <!ELEMENT price  (#PCDATA )>
]>
```

1. Design a relational schema for the XML data. Your schema should have relations corresponding to entity sets such as `Book`, `Author`, etc., as well as relationships between these entity sets. Write only the relation names and their columns; for example, `Author(aid, last, first)`; do not write the field types nor the key/foreign key constraints.
   Answer (write a relational schema):

   Solution:
   ```
   Book(bid, title, publisher, price)
   Author(aid, last, first)
   Editor(eid, last, first, affiliation)
   Writes(aid, bid)
   Edits(edi, bid)
   ```

2. Translate the two XPath queries `P1` and `P2` into SQL queries over your relational schema.

   ```
   P1 =
   /bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()

   P2 =
   /bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
   ```

   Answer (write two SQL queries):

   Solution:
   ```
   select Book.title
   from book, edits, editor
   where book.bid = edits.bid and edits.bid = editor.bid
    and book.year < 2005 and book.price < 99
    and editor.last = 'Samet'

   select x.title
   from book x, writes y, author z, writes u, author v
   where x.bid = y.bid and y.aid = z.aid
     and x.bid = y.bid and y.aid = z.aid
   ```

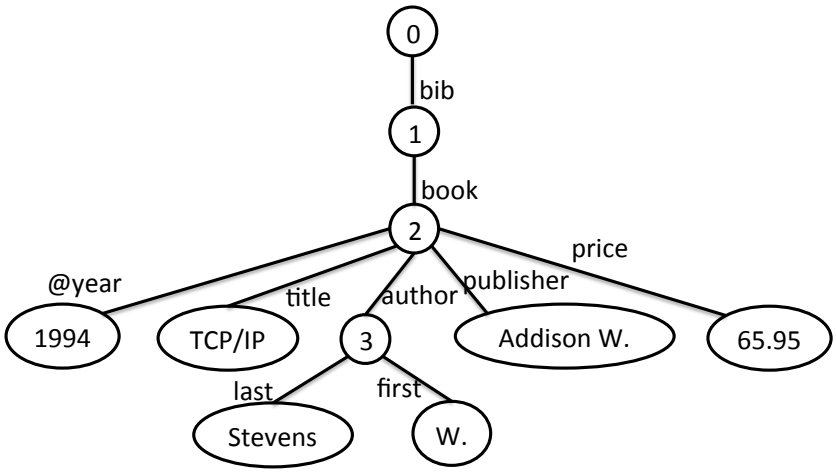
(b) Next, assume that no DTD is given, and that we do not know anything about the structure of the XML document. In this case we store it in generic table:

```
Element(id, tag, child)
```

`id` is a node identifier, `tag` is a tag (or attributes) in that node, `child` is the identifier, or the content of the child. For example, if the document were:

```
<bib> <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
      </book>
</bib>
```

Then its tree representation, and tabular representation are:



Element:

id	tag	child
0	bib	1
1	book	2
2	@year	1994
2	title	TCP/IP
2	author	3
2	publisher	Addison Wesley
2	price	65.95
3	last	Stevens
3	first	W.

The root node has always identifier 0, but all others can be arbitrary.

Translate the two XPath queries P1 and P2 into SQL over the table `Element`.

For example, if the XPath query were:

```
P0 = /bib/*/title/text()
```

then your SQL query would be:

```
select z.child
from Element x, Element y, Element z
where x.id = 0 and x.tag = 'bib'
  and x.child = y.id
  and y.child = z.id and z.tag = 'title'
```

Answer write two SQL queries for P1, P2:

```
P1 =
/bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()

P2 =
/bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
```

Solution: Let E stand for Element

```
select distinct xtitle.child
from E xbib, E xbook, E xeditor, E xlast, E xprice, E xtitle
where xbib.id = 0 and xbib.tag='bib'
  and xbook.id=xbib.child and xbook.tag='book'
  and xeditor.id=xbook.child and xeditor.tag='editor' and xeditor.child='Samet'
  and xprice.id=xbook.child and xprice.tag='price' and xprice.child <'99'
  and xtitle.id=xbook.child and xtitle.tag='title'

select distinct xtitle.child
from E xbib, E xbook, E xa1, E xl1, E xa2, E xl2, E xtitle
where xbib.id = 0 and xbib.tag='bib'
  and xbook.id=xbib.child and xbook.tag='book'
  and xa1.id = xbook.child and xa1.tag = 'author'
  and xl1.id = xa1.child and xl1.tag = 'last' and xl1.child = 'Hull'
  and xa2.id = xbook.child and xa2.tag = 'author'
  and xl2.id = xa2.child and xl2.tag = 'last' and xl2.child = 'Vianu'
  and xtitle.id = xbook.id and xtitle.tag = 'title'
```

(c) Consider the advantages and disadvantages of using the two relational representations of the XML data. For each of the criteria below, write a + if one representation is better, write a − if it is worse, and write an = is there is no clear advantage. For illustration, the first two answers are already filled out.

Criteria	Representation	
	DTD-based	DTD-free
Uses fewer tables	−	+
Uses less space	=	=
Easy to make changes to the structure (DTD)		
The XPath query `//*` translates to a simple SQL query		
Typical XPath queries translate to simple SQL queries		
Easy to check that the data is a tree		

Solution:

Criteria	Representation	
	DTD-based	DTD-free
Uses fewer tables	−	+
Uses less space	=	=
Easy to make changes to the structure (DTD)	−	+
The XPath query `//*` translates to a simple SQL query	−	+
Typical XPath queries translate to simple SQL queries	+	−
Easy to check that the data is a tree	+	−

222. *(from CSE 344, Spring 2011, Midterm)*

Consider the XML document on the next page. For each of the XPath expressions given below it, indicate what data values it returns. You may ignore any formatting. For example, if the XPath expression where `/a/b/c/text()` then you would respond $1, 2, 2, 3, 4, 5$.

```
<a>
  <b>
     <c>
        1
     </c>
     <c>
        2
     </c>
  </b>
  <b>
     <c>
        2
     </c>
     <c>
```

```
        3
      </c>
    </b>
    <b>
      <c>
        4
      </c>
      <c>
        5
      </c>
    </b>
  </a>
```

(a) `/a/b[c/text()=2]/c/text()`
The expression returns:

Solution: $1, 2, 2, 3$

(b) `/a/b/[c/text()=2][c/text()=3]/c/text()`
The expression returns:

Solution: $2, 3$

(c) `/a[b[c/text()=3][c/text()=4]]/b/c/text()`
The expression returns:

Solution: nothing

(d) `/a/[b/c/text()=3][b/c/text()=4]/c/text()`
The expression returns:

Solution: $1, 2, 2, 3, 4, 5$

(e) Consider the following DTD, representing the same data as in Question 21:

```
<!DOCTYPE db [
 <!ELEMENT db  (webpage* )>
 <!ELEMENT webpage  (url,word* )>
 <!ELEMENT word  (content, language*)>
 <!ELEMENT url      (#PCDATA )>
 <!ELEMENT content  (#PCDATA )>
 <!ELEMENT langauge (#PCDATA )>
]>
```

Write **an XQuery** expression that transforms this document into another document that lists, for each language, the urls of all webpages that have a word in that

language. Your XQuery should return an XML document with the conforming to the following DTD:

```
<!DOCTYPE answer [
 <!ELEMENT answer  (language* )>
 <!ELEMENT language (name, url*)>
 <!ELEMENT name     (#PCDATA )>
 <!ELEMENT url      (#PCDATA )>
]>
```

Answer (write an XQuery):

Solution:

```
<answer>
 { for $x in distinct-values(doc("db.xml")/db/webpage/language/text())
   return
    <language>
       <name> { $x } </name>
       { for $y in doc("db.xml")/db/webpage[word/language/text()=$x]/url/text()
         return <url> { $y } </url>
       }
    </language>
 }
</answer>
```

223. *(from CSE 344, Winter 2012, Final)*

The following DTD describes an XML document about students and the courses they take:

```
<!DOCTYPE Enrollment [
 <!ELEMENT Enrollment  (student* )>
 <!ELEMENT student  (name,  address, course*)>
 <!ELEMENT course  (title, instructor, grade?)>
 <!ELEMENT name  (#PCDATA )>
 <!ELEMENT address (#PCDATA )>
 <!ELEMENT title  (#PCDATA )>
 <!ELEMENT instructor  (#PCDATA )>
 <!ELEMENT grade  (#PCDATA )>
]>
```

(a) Write an XPath expression that retrieves the names of all students whose address is 'Seattle' and who received a grade < 3.0. You should assume that the query processor performs the correct type conversions: that is, is suffices for you to write `grade` < 3.0.

Answer (write a XPath expression):

Solution:

```
doc("problem2.xml")/Enrollment/student[address='Seattle'][course/grade<3.0]/name
```

(b) The data is not normalized: the same course is listed multiple times, once for each student who took that course. You normalize the data and represent it as follows:

```
<!DOCTYPE Normalized [
 <!ELEMENT Normalized  (students,courses,takes)>
 <!ELEMENT students  (student* )>
 <!ELEMENT courses   (course* )>
 <!ELEMENT takes   (take* )>
 <!ELEMENT student   (name, address)>
        <!ELEMENT course    (title, instructor)>
        <!ELEMENT take    (name, title, grade?)>
]>
```

Your Boss does not understand data anomalies and normalization theory, so he asks you to convert it back to the orignal format. Write an XQuery that transforms the `Normalized` data to the `Enrollment` data.

Answer (write a XPath expression):

Solution:

```
<Enrollment>
  { for $n in doc("problem2b.xml")/Normalized,
    $s in $n/students/student
    return <student>
             { $s/name,
               $s/address,
               for $e in $n/takes/take[name=$s/name],
                   $c in $n/courses/course[title=$e/title]
               return <course>
                        { $c/title,
                          $c/instructor,
                          $e/grade
                        }
                      </course>
             }
           </student>
   }
</Enrollment>
```

224. *(from CSE 344, Winter 2012, Midterm)*

(a) Consider the XML document below:

```
<a>
  <a>
    <a> 1 </a>
    <a> 2 </a>
  </a>
  <a>
    <a> 3 </a>
    <a> 4 </a>
  </a>
</a>
```

For each of the XPath queries below, indicate how many elements the expression returns:

i. `//a`

Solution: 7

ii. `//a[a/text()=3]/a`

Solution: 2

iii. `//a[a/a/text()=3]/a/a`

Solution: 4

iv. `//a[a/text()=2][a/text()=3]`

Solution: 0

v. `//a[a/a/text()=2][a/a/text()=3]`

Solution: 1

(b) For each statement below indicate whether it is true of false.

i. XML databases are replacing relational databases in today's enterprise applications. True or false?

Solution: False

ii. Compared to relational data, XML data is easier to exchange between applications or between users. True or false?

Solution: True

iii. The XPath expressions `/a/b[@c='5'][@d='2']` and `/a/b[@d='2'][@c='5']` are equivalent. True or false?

Solution: True

iv. The XPath expressions `/a/b[@c='5'][2]` and `/a/b[2][@c='5']` are equivalent. True or false?

Solution: False

v. The XPath expressions `/a/*//b` and `/a//*/b` are equivalent. True or false?

Solution: True

225. *(from CSE 344, Winter 2013, Final)*

Consider the following XML document:

```
<node>
   <node>
       <node> 1 </node>
       <node> 2 </node>
   </node>
   <node>
       <node> 3 </node>
       <node> 4 </node>
   </node>
   <node>
       <node> 2 </node>
       <node> 3 </node>
   </node>
   <node>
       <node> 4 </node>
       <node> 5 </node>
   </node>
</node>
```

Give the answer to each of the XPath expressions below.

(a) `/node/node[3]/node/text()`

Solution: 2 3

(b) `/node/node[node/text()=3]/node/text()`

Solution: 3 4 2 3

(c) `/node/node[node/text()=3][node/text()=4]/node/text()`

Solution: 3 4

(d) `/node[node/node/text()=3][node/node/text()=4]/node/node/text()`

Solution: 1 2 3 4 2 3 4 5

226. *(from CSE 344, Winter 2013, Midterm)*

Consider the following XML document:

```
<persons>
 <person>
    <name> Alice </name>
    <friend>Bob</friend>
    <friend>Carol</friend>
 </person>
 <person>
    <name> Bob </name>
    <friend>Carol</friend>
 </person>
 <person>
    <name>Carol</name>
    <enemy>Alice</enemy>
 </person>
<persons>
```

(a) Draw the tree representation of the XML document.

Solution: Straightforward.

(b) Represent the same data in relational format. Show the tables, including the table names, their attributes, and their values (tuples).

Solution:

Person:

Name
Alice
Bob
Carol

Friend:

Name1	Name2
Alice	Bob
Alice	Carol
Bob	Carol

Enemy:

Name1	Name2
Carol	Alice

227. *(from CSE 344, Autumn 2013, Midterm)*

For each statement below indicate whether it is true or false.

(a) XML is better suited than the relational data model for data exchange between applications.

Solution: Yes

(b) The data model for XML is a tree.

Solution: Yes

(c) B^+ trees are stored in XML

Solution: No

(d) An XML tree can be at most 6 levels deep.

Solution: No

(e) The tuples in a relation are unordered.

Solution: Yes

(f) The elements in an XML document are unordered

Solution: No

(g) The *Data Independence* principle states that the tuples in a relation must be independent of each other.

Solution: No

(h) The *Data Independence* principle states that applications should be written independently of the way the data is stored and organized.

Solution: Yes

(i) All elements returned by the XPath expression `/a//b[c=4]/d` are of type `<d>...</d>`.

Solution: Yes

(j) If the XPath expression `/a//b[c=4]` returns 10 elements, then the XPath expression `/a//b/c` must return at least 10 elements.

Solution: Yes

228. *(from CSE 544p, Winter 2014, Final; CSE 544p, Autumn 2015, Final)*

(a) Consider the following data items with associated vector clocks. Indicate whether they are in conflict. Check *only* if they are in conflict:

i. $([SX, 3], [SY, 7])$, $([SX, 5], [SY, 7])$
Conflict?

Solution: no

ii. $([SX, 9], [SZ, 2])$ $([SX, 9], [SY, 6])$
Conflict?

Solution: yes

iii. $([SX, 9], [SY, 2], [SZ, 1])$, $([SX, 9], [SY, 2])$
Conflict?

Solution: no

iv. $([SX, 8], [SY, 1])$, $([SX, 8], [SY, 2], [SZ, 2])$
Conflict?

Solution: no

v. $([SX, 3], [SY, 10])$, $([SX, 3], [SY, 6], [SZ, 2])$
Conflict?

Solution: yes

(b) For each of the following statements indicate whether it is true or false.

i. Column-oriented databases are more efficient than row-oriented databases on

OLTP query workloads.
True or false?

Solution: false

ii. Column-oriented databases can compress the data much more efficiently than row-oriented databases.
True or false?

Solution: true

iii. A *latch* is just another name for a lock in a database.
True or false?

Solution: false

iv. Many NoSQL databases have support only for single record transactions.
True or false?

Solution: true

v. All column-oriented databases are NoSQL databases.
True or false?

Solution: false

229. *(from CSE 344, Winter 2016, Final)*

(a) Answer the multi-choice questions below:

i. What do we mean when we say that the data is *not in First Normal Form*?
Check all that apply.

1. There exists a non-trivial functional dependency $X \rightarrow Y$ where X is not a superkey.
2. The data is represented in a human-readable form, like JSon.
3. The value of an attribute of a table is a collection, such as a table or an array.
4. The table has no clustered index.

Select from 1,2,3, and/or 4:

Solution: 3

ii. Check which of the following statements are true about the semistructured data model:

1. JSon is semistructured data.
2. In semistructured data the value of an attribute can be another collection.
3. Semistructured data means that the data is compressed.
4. There are no query languages for semistructured data.

Select from 1,2,3, and/or 4:

Solution: 1,2

(b) Consider the following database, given in JSon:

```
{"Course":
  [ {"title": "Math101",
     "room": "F777",
     "instructor": {"Name": "Bob", "Office": "E999"};
     "enrollment": [{"name": "David", "year": 2},
                    {"name": "Erol", "year": 3}
                   ]
    },
    {"title": "Phys202",
     "room": "H909",
     "instructor": {"name": "Alice", "office": "C222"};
     "enrollment": [{"name": "Carol", "year": 2},
                    {"name": "Erol", "year": 3},
                    {"name": "Fred", "year": 1}
                   ]
    },
    {"title": "CSE703",
     "room": "G080",
     "instructor": {"name": "Bob", "office": "E999"}
    }
  ]
}
```

Your task is to convert this data into a relational database.

i. Design a schema for the a relational database capable of storing the database above. Turn in relation names, their attributes, and underline the key. For example, you may write:

`Room(`<u>`roomnumber`</u>`, instructorname, floorid),Floor(floorid, level)`

(not a real answer). Write your answer below:

Solution:

```
Course(title,room,instructor)
Instructor(name,office)
Student(name,year)
Enrollment(courseTitle,studentName)
```

JSon file repeated from previous page:

```
{"Course":
  [ {"title": "Math101",
     "room": "F777",
     "instructor": {"Name": "Bob", "Office": "E999"};
     "enrollment": [{"name": "David", "year": 2},
                    {"name": "Erol", "year": 3}
                   ]
    },
    {"title": "Phys202",
     "room": "H909",
     "instructor": {"name": "Alice", "office": "C222"};
     "enrollment": [{"name": "Carol", "year": 2},
                    {"name": "Erol", "year": 3},
                    {"name": "Fred", "year": 1}
                   ]
    },
    {"title": "CSE703",
     "room": "G080",
     "instructor": {"name": "Bob", "office": "E999"}
    }
  ]
}
```

ii. Show the content of your tables, representing the same data as the JSon file. For example, you may write:

Room

roomnumber	instructorname	floorid
E999	Carol	flr0521
C222	Alice	flr0521

Floor

floorid	level
flr0521	2

(not a real answer) Turn in several table instances:

Solution:

Professor

Name	Office
Alice	C222
Bob	E999

Student

Name	Year
Carol	2
David	2
Erol	3
Fred	1

Course

Title	Room	Instructor
Math101	F777	Bob
Phys202	H909	Alice
CS303	G080	Bob

Enrollment

courseTitle	studentName
Math101	David
Math101	Erol
Phyis202	Carol
Phyis202	Erol
Phyis202	Fred

230. *(from CSE 414, Spring 2016, Final)*

(a) For each small data instance below indicate whether we should call it relational data, or semi-structured data:

i. Is this data:

```
--------------------------
| name  | email    | age  |
--------------------------
| Alice | a@li.ce  | 31   |
| Bob   | b@ob.com | 24   |
| Carol | NULL     | 37   |
| David | da@vid.el| NULL |
| Erol  | e@rol    | 22   |
--------------------------
```

relational or semi-structured?

Solution: Relational

ii. Is this data:

```
name,email,age
Alice,a@li.ce,31
Bob,b@ob.com,24
Carol,,37
David,da@vid.el,
Erol,e@rol,22
```

relational or semi-structured?

Solution: Relational

iii. Is this data:

```
{"Person"
   [{"name":"Alice","email":"a@li.ce","age":31},
    {"name":"Bob","email":"b@ob.com","age":24},
    {"name":"Carol","age":37},
    {"name":"David","email":"da@vid.el"},
    {"name":"Erol","email":"e@rol","age":22}
   ]
```

```
    }
```

relational or semi-structured?

Solution: Relational

iv. Is this data:

```
{"Person"
   [{"name":"Alice","email":["a@li.ce","alice@gmail.com"],"age":31},
    {"name":"Bob","email":"b@ob.com","age":24},
    {"name":"Carol","age":37},
    {"name":{"first":"David","last":"Brown"}, "email":"da@vid.el"},
    {"name":"Erol","email":"e@rol","age":22}
   ]
}
```

Relational or semi-structured?

Solution: Semi-structured

(b) Consider the following applications. For each, indicate whether you would use a relational data model, or a semistructured data model:

i. You work for a large, national financial institution. Every night, all databases from the local branches of your institution sent over the internet all their updates for the day to a central server. The updates are complex in structure and may involve accounts, users, financial transactions, etc. What data model would you use to model the data exchange?
Semi-structured or relational?

Solution: Semi-structured

ii. You work for a large online shopping company, and maintain their Orders database. They have tens of millions of orders, and need to access about 2-300 orders per second; orders are retrieved either by the order ID, or by the date, or by the customer. What data model would you use for the orders database?
Semi-structured or relational?

Solution: Relational

231. *(from CSE 344, Autumn 2017, Final)*

(a) A NoSQL database stores one billion (10^9) key-value pairs. The system runs on 100 servers, and replicates every key-value pair on three randomly chosen servers, to increase reliability. Replicas may be out of sync, but they will eventually be reconciled by the system, i.e. if a value is updated on one server, eventually the other two replicas of that value will also be updated. Answer the following questions. In all cases, you may round your answer to the closest integer.

i. How many key-value pairs are stored on one server?

Solution: 30 million ($30 \cdot 10^6$).

ii. A transaction needs to access 5 different key-value pairs, and does not necessarily need to get the most up to date values. From how many servers does it need to read (in expectation)?

Solution: 5

iii. A transaction needs to access 5 different key-value pairs, and must get the most up to date values. From how many servers does it need to read (in expectation)?

Solution: 15

iv. One day three servers fail catastrophically, simultaneously. In expectation, how many key-value pairs are lost after this failure? Your answer may be approximate. Note that the NoSQL database can recover a key-value pair if at least one replica survives; you need to compute the (approximate) number of key-value pairs that cannot be recovered after three servers fail.

Solution: 6000: each key is lost with probability $1/\binom{100}{3}$.

(b) For each statement below indicate if it is true or false.

i. The main reason why NoSQL databases were introduced was because relational databases did not scale up to a very large number of servers. True or false?

Solution: True

ii. In JSon one can represent nested collections. True or false?

Solution: True

iii. The JSon data model is in First Normal Form. True or false?

Solution: False

iv. It is easy to represent a many-to-one relationship in JSon, but it is difficult to represent a many-to-many relationship. True or false?

Solution: True

v. SQL++ can express recursive queries, such as transitive closure. True or false?

Solution: False

232. *(from CSE 344, Autumn 2017, Midterm)*

(a) Answer the following questions. In your answer you may omit apostrophes, i.e. you may write {A:a1} instead of {'A':'a1'}. All four questions refer to the same collection t:

```
with t as
[{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
 {'A':'a2', 'F':[{'B':'3'}, {'B':'4'},{'B':'5'}], 'G':[ ]},
 {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
```

i. What does the following SQL++ query return?

```
Select x.A
From t x, x.G y
where y.C='1';
```

Solution:

```
{ "A": "a1" }
{ "A": "a3" }
```

ii. What does the following SQL++ query return?

```
Select x.A, y.B
From t x, x.F y;
```

Solution:

```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
{ "A": "a2", "B": "3" }
{ "A": "a2", "B": "4" }
{ "A": "a2", "B": "5" }
{ "A": "a3", "B": "2" }
```

iii. What does the following SQL++ query return?

```
Select x.A, y.B
From t x, x.F y, x.G z
where y.B=z.C ;
```

Solution:
```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
```

iv. What does the following SQL++ query return?
```
Select x1.A as A1, x2.A as A2, y.B
From t x1, t x2, x1.F y, x2.G z
where y.B=z.C ;
```

Solution:
```
{ "A1": "a1", "A2": "a1", "B": "1" }
{ "A1": "a1", "A2": "a3", "B": "1" }
{ "A1": "a1", "A2": "a1", "B": "2" }
{ "A1": "a2", "A2": "a3", "B": "3" }
{ "A1": "a3", "A2": "a1", "B": "2" }
```

(b) For each statement below, indicate whether it is true or false.

i. An OLTP workload means a workload of queries that have many joins, aggregates, and very few or no updates. True/False:

Solution: False

ii. An OLAP workload means a workload of queries that have many joins, aggregates, and very few or no updates. True/False:

Solution: True

iii. NoSQL systems are primarily intended for OLTP workloads, and not for OLAP workloads. True/False:

Solution: True

iv. NoSQL systems emerged because SQL is a very old language and needs to be replaced by something more modern. True/False:

Solution: False

v. NoSQL systems typically run on a large, distributed cluster (meaning: many servers). True/False:

Solution: True

vi. NoSQL systems support physical data independence better than relational database systems. True/False:

Solution: False

233. *(from CSE 344, Winter 2019, Final)*

(a) We are given a JSON file with noble prize laureates, with the following structure:

```
{"prizes": [
    { "year": "2018",
      "category": "physics",
      "overallMotivation": "\For groundbreaking inventions in the field of laser physics"",
      "laureates": [
        { "id": "960",
          "name": "Arthur Ashkin",
          "motivation": "\"for the optical tweezers and their application to biological systems\"",
          "share": "2"
        },
        { "id": "961",
          "name": "Gérard Mourou",
          "motivation": "\"for their method of generating high-intensity, ultra-short optical pulses\"",
          "share": "4"
        },
        { "id": "962",
          "name": "Donna Strickland",
          "motivation": "\"for their method of generating high-intensity, ultra-short optical pulses\"",
          "share": "4"
        }
      ]
    },
    { "year": "2018",
      "category": "chemistry",
      ...
    },
    { "year": "2018",
      "category": "medicine",
      ...
    }
  ]
```

Write a SQL++ query that returns each noble prize laureate who has received more than one award, along with a list of the years and categories that each such laureate has received. Your query should return a JSON file with a structure like the following:

```
{ "name": "Frederick Sanger",
  "awards": [ { "year": "1958", "category": "chemistry" },
              { "year": "1980", "category": "chemistry" } ]
}
{ "name": "Marie Curie, née Sklodowska",
  "awards": [ { "year": "1903", "category": "physics" },
              { "year": "1911", "category": "chemistry" } ]
}
...
```

[this page is intentionally left blank]

Solution:

```
SELECT DISTINCT l.name, awards
FROM prizes p, p.laureates l
LET awards=(                          --Use a subquery to nest
  SELECT p2.year, p2.category         --DISTINCT is optional
  FROM prizes p2, p2.laureates l2
  WHERE l2.name = l.name
)
WHERE array_count(awards) > 1;        --coll_count() also works

Note: data adapted from: http://api.nobelprize.org/v1/prize.JSON

Using the real data, one solution is:

CREATE DATAVERSE noble;
CREATE TYPE noble.prizeType AS {auto_id:uuid};
CREATE DATASET noble.prize(noble.prizeType)
  PRIMARY KEY auto_id AUTOGENERATED;
LOAD DATASET noble.prize USING localfs
  (("path"="localhost://<path_to>/prize.json"),
   ("format"="json"));

WITH reduced AS (
  SELECT p.year, p.category,
         l.firstname || " " || l.surname AS name
  FROM noble.prize np, np.prizes p, p.laureates l
)
SELECT DISTINCT r1.name, awards
FROM reduced r1
LET awards=(
  SELECT year, category
  FROM reduced r2
  WHERE r2.name = r1.name
)
WHERE array_count(awards) > 1;
```

234. *(from CSE 344, Winter 2019, Midterm)*

(a) Answer the following questions. In your answer you may omit apostrophes, i.e. you may write {A:a1} instead of {'A':'a1'} .

i. What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'},{'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A
From  t  x, x.G y
where y.C='1';
```

Solution:

```
{ "A": "a1" }
{ "A": "a3" }
```

ii. What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
```

```
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A, y.B
From  t  x, x.F y;
```

Solution:

```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
{ "A": "a2", "B": "3" }
{ "A": "a2", "B": "4" }
{ "A": "a2", "B": "5" }
{ "A": "a3", "B": "2" }
```

iii. What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A, y.B
From  t  x, x.F y, x.G z
where y.B=z.C ;
```

Solution:

```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
```

iv. What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x1.A as A1, x2.A as A2, y.B
From  t x1, t x2, x1.F y, x2.G z
where y.B=z.C ;
```

Solution:

```
{ "A1": "a1", "A2": "a1", "B": "1" }
{ "A1": "a1", "A2": "a3", "B": "1" }
{ "A1": "a1", "A2": "a1", "B": "2" }
{ "A1": "a2", "A2": "a3", "B": "3" }
{ "A1": "a3", "A2": "a1", "B": "2" }
```

(b) For each statement below, indicate whether it is true or false.

i. An *OLTP workload* means a workload of queries that have many joins, aggregates, and very few or no updates.
True/False:

Solution: false

ii. An *OLAP workload* means a workload of queries that have many joins, aggre-

gates, and very few or no updates.
True/False:

Solution: true

iii. *Physical data independence* means that the data is compressed.
True/False:

Solution: false

iv. The key-value pair data model is better suited for complex queries (with aggregates) than for simple queries (single lookups).
True/False:

Solution: false

v. NoSQL systems typically run on a large, distributed cluster (meaning: many servers).
True/False:

Solution: true

vi. NoSQL systems support physical data independence better than relational database systems.
True/False:

Solution: false

235. *(from CSE 414, Spring 2019, Final)*

The raw data that biologist used to populate her database came in JSon format directly from the ship where the samples were collected and analyzed. It had the following structure:

```
{"samples": [
    { "sid": "s001",
      "lat": "21.2",
      "long": "-157.9",
      "depth": "300",
      "analysis":
        { "technician": "Alice",
          "organisms": [ {"oid": "o252", "name": "Trichodesmium"},
                         {"oid": "o301", "name": "Crocosphaera"},
                         ...
                       ]
        }
    },
    { "sid": "s002",
```

```
      "lat": "25.0",
      "long": "-150.0",
      "depth": "400",
      "analysis":
        { "technician": "Bob",
          "organisms": [ {"oid": "o301", "name": "Crocosphaera"},
                         {"oid": "o999", "name": "Prochlorococcus"},
                         {"oid": "o777", "name": "Synechococcus"},
                         ...
                       ]
        }
    },
    ...
    ]
}
```

Write tree SQL++ queries to convert the JSon data above into the relational schema.

(a) Write the SQL++ query to construct the `Sample` table. Your query should return an output like:

```
[ { "sid": "s001", "lat": "21.2", "long": "-157.9", "depth": "300", "technician": "Alice"},
  { "sid": "s002", "lat": "25.0", "long": "-150.0", "depth": "400", "technician": "Bob" },
  ...
]
```

Solution:

```
select x.sid, x.lat, x.long, x.depth, x.analysis.technician
from samples x;
```

(b) Write the SQL++ query to construct the `Analysis` table. Your query should return an output like this:

```
[ { "sid": "s001", "oid": "o252"},
  { "sid": "s001", "oid": "o301"},
  ...
  { "sid": "s002", "oid": "o301"},
  ...
]
```

Solution:

```
select x.sid, y.oid
from samples x, x.analysis.organisms y;
```

(c) Write the SQL query to construct the `Organisms` table. Your query should return an output like this:

```
[ { "oid": "o252", "name": "Trichodesmium"},
  { "oid": "o301", "name": "Crocosphaera},
  { "oid": "o999", "name": "Prochlorococcus},
  ...
]
```

Solution:

```
select distinct y.oid, y.name
from samples x, x.analysis.organisms y;
```

236. *(from CSE 414, Spring 2019, Midterm)*

(a) Consider the relational database instance below:

Car:

lp	make	owner
XY52Z	Honda	Alice
ZY23X	Bentley	Bob
MWM92	Bentley	Alice

Downtown:

lp	day
MWM92	2000
MWM92	3000
ZY23X	4000
MWM92	5000

Write a Json file that represents the same data.

Solution:

```
{"Database":
   [{"lp": "XY52Z",
     "make": "Honda",
     "owner": "Alice",
     "downtown":[]},
    {"lp": "ZY23X",
     "make": "Bentley",
     "owner": "Bob",
     "downtown":[4000]},
    {"lp": "MWM92",
     "make": "Bentley",
     "owner": "Alice",
     "downtown": [2000,3000,5000]}]}
```

We also accepted solutions grouped by owner:

```
{"Database":
   [{"owner": "Alice",
     "cars": [ {"lp": "XY52Z",
                "make": "Honda",
                "downtown":[]},
               {"lp": "MWM92",
                "make": "Bentley",
                "downtown": [2000,3000,5000]}]},
    {"owner": "Bob",
     "cars": [ {"lp": "ZY23X",
                "make": "Bentley",
                "downtown":[4000]} ] } ] }
```

237. *(from CSE 414, Spring 2024, Final)*

(a) We are given a JSON dataset called `courses` which contains an array of courses, where each course contains the following fields:

- `code` is the code of the course.
- `title` is the title of the course.
- `students` is an array of students.

Each student contains the following fields: `name` and `grade` (the grade in that class). Write a SQL++ query that re-groups the data by students. Your query should return an array of students, each consisting of the student's name and an array of classes taken by that student.
For example, if the JSON data were:

```
[ { "code" : "cse414",
    "title": "Introduction to Data Management",
    "students":
       [ { "name": "Alice", "grade": 3.5},
         { "name": "Bob",   "grade": 3.0}
       ]
   },
   { "code" : "cse373",
     "title" : "Introduction to Data Structures",
     "students":
        [ { "name": "Alice", "grade": 3.9},
          { "name": "Eve",   "grade": 2.5}
        ]
   }
]
```

then your query should return:

```
{ "name": "Eve",
  "courses": [ { "code": "cse373", "title": "Introduction to Data Structures", "grade": 2.5 } ]
}
{ "name": "Bob",
  "courses": [ { "code": "cse414", "title": "Introduction to Data Management", "grade": 3.0 } ]
}
{ "name": "Alice",
  "courses": [ { "code": "cse414", "title": "Introduction to Data Management", "grade": 3.5 },
               { "code": "cse373", "title": "Introduction to Data Structures", "grade": 3.9 }
             ]
}
```

Write your answer here:

Solution:

```
DROP DATAVERSE course IF EXISTS;
CREATE DATAVERSE course;

CREATE TYPE course.studentType AS
  { name: string, grade: float };

CREATE TYPE course.courseType AS
  { code: string, title: string, students: [course.studentType] };

CREATE DATASET course.course(course.courseType) PRIMARY KEY code;
```

```
LOAD DATASET course.course USING localfs
  (("path"="localhost:///data/course.json"),
   ("format"="json"));

USE course;

let names = (from course.course x, x.students y select distinct y.name)
from names s
select s.name,
       (from course.course x, x.students y
        where s.name = y.name
 select x.code, x.title, y.grade) as courses;
```

238. *(from CSE 344, Autumn 2024, Final)*

(a) Consider a dataset called `NestedAB` with consists of a set of objects with two attributes, A and B. The value of A is an integer, and the value of B is an array of integers. For example, `NestedAB` might look like this:

```
[ { "A" : 1,
    "B": [ 10, 20, 30 ]
   },
   { "A" : 2,
    "B": [ 20, 50 ]
   },
   { "A" : 3
   },
   { "A" : 2,
    "B": [ 30, 10 ]
   }
]
```

Write a SQL++ query to unnest this dataset. Your query should return an array of objects, where each of the attributes A, B is an integer.

For example, given the instance above, your query should return:

```
[ { "A" : 1,  "B": 10 },
  { "A" : 1,  "B": 20 },
  { "A" : 1,  "B": 30 },
  { "A" : 2,  "B": 20 },
  { "A" : 2,  "B": 50 },
  { "A" : 2,  "B": 30 },
  { "A" : 2,  "B": 10 }
]
```

Notice that the value $A = 3$ is lost, because it is not paired with any B values.

Write your answer here:

Solution:

```
DROP DATAVERSE AB IF EXISTS;
CREATE DATAVERSE AB;

CREATE TYPE AB.NestedABType AS
  { K: uuid, A: int, B: [int]? };

CREATE TYPE AB.UnnestedABType AS
  { K: uuid, A: int, B: int };
```

```
CREATE DATASET AB.NestedAB(AB.NestedABType) PRIMARY KEY K AUTOGENERATED;

CREATE DATASET AB.UnnestedAB(AB.UnnestedABType) PRIMARY KEY K AUTOGENERATED;

LOAD DATASET AB.NestedAB USING localfs
  (("path"="localhost:///data/nestedAB.json"),
   ("format"="json"));

LOAD DATASET AB.UnnestedAB USING localfs
  (("path"="localhost:///data/unnestedAB.json"),
   ("format"="json"));

from AB.NestedAB as x, x.B as y
select x.A, y as B;
```

(b) Write a SQL++ query to go in the opposite direction. Given the unnested dataset `UnnestedAB`, compute a nested output. For example, if the unnested data is the one above then your query should return:

```
[ { "A" : 1,
    "B": [ 10, 20, 30 ]
   },
   { "A" : 2,
    "B": [ 20, 50, 30, 10]
   },
]
```

Write your answer here:

Solution:

```
let temp = (from  AB.UnnestedAB as x select distinct x.A)
from temp z
select z.A;

let temp = (from  AB.UnnestedAB as x select distinct x.A)
from AB.UnnestedAB as x, temp as z
where x.A = z.A
select x;

let temp = (from  AB.UnnestedAB as x select distinct x.A)
from temp as z
select z.A, (from AB.UnnestedAB as x where x.A = z.A select x);

let temp = (from  AB.UnnestedAB as x select distinct x.A)
from temp as z
select z.A, (from AB.UnnestedAB as x where x.A = z.A select x.B);

let temp = (from  AB.UnnestedAB as x select distinct x.A)
from temp as z
select z.A, (from AB.UnnestedAB as x where x.A = z.A select x.B) as B;

let AllA = (from  AB.UnnestedAB as x select distinct x.A)
from AllA a
select a.A, (from AB.UnnestedAB x where a.A = x.a select x.B);
```

239. *(from CSE 344, Autumn 2013, Final)*

Consider a relational database about publications, with the following schema:

```
Author(aid, name)       /* aid = key */
Authored(aid, pid)      /* aid = foreign key to Authored;
                           pid = foreign key to Paper */
Paper(pid, title, year) /* pid = key */
```

Solution:

```
create table author(aid int primary key, name text);
create table paper(pid int primary key, title text, year int);
create table authored(aid int references author, pid int references paper);

insert into author values(1, 'Alice');
insert into author values(2, 'Bob');
insert into author values(3, 'Carol');

insert into paper values(10, 'abc', 1995);
insert into paper values(20, 'def', 1995);
insert into paper values(30, 'ghk', 2010);

insert into authored values(1, 10);
insert into authored values(2, 10);
insert into authored values(2, 20);
insert into authored values(1, 30);
```

We export the entire data in XML, using the following DTD:

```
<!DOCTYPE bib [
        <!ELEMENT bib  (paper* )>
        <!ELEMENT paper   (pid, title,  year, author*)>
        <!ELEMENT author  (aid, name)>
        <!ELEMENT pid   (#PCDATA )>
        <!ELEMENT title (#PCDATA )>
        <!ELEMENT year  (#PCDATA )>
        <!ELEMENT aid   (#PCDATA )>
        <!ELEMENT name  (#PCDATA )>
]>
```

Solution:

```
<bib>
  <paper>
        <pid>10</pid>
        <title>abc</title>
        <year>1995</year>
        <author>  <aid>1</aid> <name>Alice</name> </author>
        <author>  <aid>2</aid> <name>Bob</name> </author>
```

```
  </paper>
  <paper>
        <pid>20</pid>
        <title>def</title>
        <year>1995</year>
        <author>  <aid>2</aid> <name>Bob</name> </author>
  </paper>
  <paper>
        <pid>30</pid>
        <title>ghk</title>
        <year>2010</year>
        <author>  <aid>1</aid>  <name>Alice</name> </author>
  </paper>
</bib>
```

(a) Answer the following questions:

i. Are the `pid` elements unique in the XML document?
Answer yes or no

Solution: yes

ii. Are the `aid` elements unique in the XML document?
Answer yes or no

Solution: no

iii. Could there be any `Paper`s lost during export?
Answer yes or no

Solution: no

iv. Could there be any `Author`s lost during export?
Answer yes or no

Solution: yes

(b) The following SQL query returns all papers written by Alice and Bob in 1995:

```
select x.title
from Paper x, Authored y, Author z,
              Authored u, Author v
where x.pid = y.pid and y.aid = z.aid
  and x.pid = u.pid and u.aid = z.aid
  and x.year = 1995
  and z.name = 'Alice'
  and v.name = 'Bob';
```

Write an equivalent query in XPath over the exported data.

Solution:

```
doc("file.xml")/bib/paper[year/text()=1995]
                         [author/name/text()='Alice']
                         [author/name/text()='Bob']/title
```

(c) The following SQL query returns all authors who published both in 1995 and in 2010:

```
select distinct w.name
from Paper x, Authored y, Paper u, Authored v, Author w
where x.pid = y.pid and y.aid = w.aid
  and u.pid = v.pid and v.aid = w.aid
  and x.year = 1995 and u.year = 2010;
```

Write an equivalent query in XQuery over the exported data.

Solution:

```
let $a :=
   for $x in doc("file.xml")/bib/paper[year/text()=1995]/author,
        $u in doc("file.xml")/bib/paper[year/text()=2010]/author
   where $x/aid/text() = $u/aid/text()
   return $x/name/text()
for $x in distinct-values($a)
return <name> { $x } </name>
```

18 Miscellaneous (240–246)

What remains does not fit under any single heading. The largest group concerns similarity search: Jaccard similarity of q-gram sets, edit distance between strings, minhash signatures, and locality-sensitive hashing. This is the machinery for finding near-duplicate documents without comparing every pair. Bloom filters make a brief appearance.

The rest are short review questions, mostly true-or-false, on the ideas that span the whole course: physical and logical data independence, what the normal forms are for, and where the boundary lies between what the database system guarantees and what the application must arrange for itself. They make a reasonable last check before an exam.

240. *(from CSE 444, Autumn 2008, Final)*

Data supplier `S1` has $n = 1M$ $(= 10^6)$ documents. Data supplier `S2` has also $n = 1M$ documents. Each document has $1k$ bytes. They have 50 documents in common and they want to compute these. They will proceed as follows:

- `S1` computes a hash map M with cn bits, where $c = 8$, and sends it to `S2`
- `S2` checks its items in M and sends all matches to `S1`
- `S1` computes the result and sends the matching 50 documents to `S2`

(a) Indicate the total number of bytes transferred over the network in each step assuming (a) the hash map is a standard hash table, (b) the hash map is a Bloom filter. (You may use the formulas in the lecture notes.)

Solution: Step 1: compute a hash map of $8 \cdot 10^6$ bits = 1MB. Send 1MB.

Step 2: compute matches. Expected number of matches:

$1M \times 11\% = 100k$ documents for the hash map
$1M \times 2\% = 20k$ documents for the Bloom filter

(one may want to add the 50 documents that are true positives). Send 100MB, or 20MB respectively.

Step 3: compute the intersection, find the 50 documents, send them back. Send 50k.

241. *(from CSE 444, Spring 2010, Final)*

(a) Is the following statement true or false ? Bloom filters are used to improve cardinality estimation. True or false ?

Solution: FALSE. Bloom filter is used for reducing communication costs between two nodes in a distributed system.

(b) A Bloom filter using a hash map of 1k Bytes has a false positive error rate of 19%. In order to improve the error rate, the systems administrator decides to double the size of the hash map to 2k Bytes, and, at the same time, to double the number of hash functions used by the Bloom filter. What is the new false positives error rate ?

The new false positive error rate is:

Solution: We are given that $[1 - \texttt{exp}(-kn/m)]^k = .19$. If m and k are now doubled, we have:

$[1-\texttt{exp}(-2kn/2m)]^{2k} = [1-\texttt{exp}(-kn/m)]^{2k} = [[1-\texttt{exp}(-kn/m)]^k]^2 = .19^2 = .0361$

Thus the new false positive rate is 3.61%.

(c) A regular hash map of 1k Bytes has a false positive error rate of 19%. In order to improve this rate, the systems administrator decides to double the size of the hash map to 2k Bytes. What is the new false positive error rate ?

The new false positive error rate is:

Solution: We are given that $1 - \texttt{exp}(-n/m) = .19$. So $\texttt{exp}(-n/m) = .81 \Rightarrow \texttt{exp}(-n/2m) = \sqrt{.81} = .9$. Thus the new false positive rate is $1-\texttt{exp}(-n/2m) = 1 - .9 = .1$.

(d) Data supplier S1 has $n = 1M$ $(= 10^6)$ documents. Data supplier S2 has also $n = 1M$ documents. Each document has $1k$ bytes. They have 50 documents in common and they want to compute these. They will proceed as follows:

- S1 computes a hash map M with cn bits, where c=8 and sends it to S2
- S2 checks its items in M and sends all matches to S1
- S1 computes the result and sends the matching 50 documents to S2

Indicate the total number of bytes transferred over the network in each step assuming (a) the hash map is a standard hash table, (b) the hash map is a bloom filter.

Solution: There are three parts to the cost:

1. S1 computes a hashtable / bloom filter and sends it to S2. This takes cn bits = 8M bytes = 1MB.
2. S2 sends back documents that is positive when matched against the hashtable / bloom filter. The number of documents is 50 + (number of false positives).
   - For hash table, the false positive rate is $1 - \texttt{exp}(-n/m) = 1 - \texttt{exp}(-n/8n) = 1 - \texttt{exp}(-1/8)$, or 11%. So there are 110,000 false positive documents, making the total number of documents 110,050 and size 110,050KB.
   - For bloom filter, the false positive rate is $(1 - \texttt{exp}(-k/8))^k$.

Assuming optimal number of hash functions, the false positive rate is $2^{(-8ln2)}$, or 2%. So there are 20,000 false positive documents, making the total 20,050 and size 20,050KB.

3. S1 computes and sends the 50 common documents back to S2. This takes 50KB.

So for hash table, the total cost is $1MB + 110.05MB + 0.05MB = 111.10MB$. For bloom filter, the total cost is $1MB + 20.05MB + 0.05MB = 21.20MB$.
(Full credit was given if second step was left in terms of exponentials or in terms of k if optimal bloom filter was not assumed.)

242. *(from CSE 544p, Autumn 2010, Final)*

(a) Is the following statement true or false ? Bloom filters are used to improve cardinality estimation.
True or false ?

Solution: FALSE. Bloom filter is used for reducing communication costs between two nodes in a distributed system.

(b) A Bloom filter using a hash map of 1k Bytes has a false positive error rate of 19%. In order to improve the error rate, the systems administrator decides to double the size of the hash map to 2k Bytes, and, at the same time, to double the number of hash functions used by the Bloom filter. What is the new false positives error rate ?
The new false positive error rate is (express it in percentages):

Solution: We are given that $[1 - \texttt{exp}(-kn/m)]^k = .19$. If m and k are now doubled, we have:

$$[1-\texttt{exp}(-2kn/2m)]^{2k} = [1-\texttt{exp}(-kn/m)]^{2k} = [[1-\texttt{exp}(-kn/m)]^k]^2 = .19^2 = .0361$$

Thus the new false positive rate is 3.61%.
I also accepted answers close to the correct value 3.61, like 3.6 or 3.75.

(c) A regular hash map of 1k Bytes has a false positive error rate of 19%. In order to improve this rate, the systems administrator decides to double the size of the hash map to 2k Bytes. What is the new false positive error rate ?

The new false positive error rate is:

Solution: We are given that $1 - \texttt{exp}(-n/m) = .19$. So $\texttt{exp}(-n/m) = .81 \Rightarrow \texttt{exp}(-n/2m) = \sqrt{.81} = .9$. Thus the new false positive rate is $1-\texttt{exp}(-n/2m) = 1 - .9 = .1$. The answer is 10%. I accepted both 10 an 0.1 and values close to these numbers.

(d) Alice and Bob are two secret agents in a dangerous country, with computers but without internet. Alice has a set of $n = 1M$ $(= 10^6)$ secret documents. Bob also has a collection of $n = 1M$ secret documents. Alice and Bob know that they have exactly 50 documents in common and they would like to find out which ones. But the only way they can exchange secret documents securely is to print them and send them by mail (!). Since mailing a document is expensive, they decide to use a hash map M to filter out some documents. They need to print M too (hence it cannot be too big), so they decide to use a Bloom filter with $m = 16n$ bits $(= 16 \cdot 10^6$ bits, or 2M Bytes) to filter out some documents. Specifically, they proceed as follows:

- Alice computes a hash map M consisting of an optimal Bloom filter with $m = 16n$ bits, prints it, and mails it to Bob.
- Bob checks each of its documents if it is in M. For each document that is in M, it prints it, and mails it to Alice.
- Alice opens the letters, and checks each document it receives whether it is in her collection. She then notifies Bob of all the common documents she found.

Compute the expected number of documents that Bob needs to send by mail; round to the next largest integer. You have to turn in an integer number.

Solution: The false positive rate for an optimal Bloom filter is:

$$f = \left(\frac{1}{2}\right)^{\log 2 \frac{m}{n}} = 0.000458$$

Thus, the number of documents sent is:

$$999950 * 0.000458 + 50 = 509$$

243. *(from CSE 344, Spring 2011, Final)*

(a) Consider the following two words:

```
s1 = tamper
s2 = teleporter
```

1. Compute the edit distance between the two strings. Assume that we allow three edit operations: insert, replace, delete, and each has cost 1. You may use the table below to compute the answer, but you do not need to fill in the entire table: you only need to write the final answer.

		t	a	m	p	e	r
t							
e							
l							
e							
p							
o							
r							
t							
e							
r							

   The edit distance is:

2. Represent both s1 and s2 as sets of 2-grams (`tamper` has 5 2-grams, and `teleporter` has 9 2-grams). Compute their Jaccard similarity.
   When $q = 2$, the Jaccard similarity is:

3. Represent both s1 and s2 as sets of 3-grams. Compute their Jaccard similarity.
   When $q = 3$, the Jaccard similarity is:

Solution: 1. The edit distance is 6. Filling in the dynamic programming table (rows: `teleporter`, columns: `tamper`; each entry is the edit distance between the two prefixes) gives:

		t	a	m	p	e	r
	0	1	2	3	4	5	6
t	1	0	1	2	3	4	5
e	2	1	1	2	3	3	4
l	3	2	2	2	3	4	4
e	4	3	3	3	3	3	4
p	5	4	4	4	3	4	4
o	6	5	5	5	4	4	5
r	7	6	6	6	5	5	4
t	8	7	7	7	6	6	5
e	9	8	8	8	7	6	6
r	10	9	9	9	8	7	6

The answer is the bottom right entry, 6. One alignment achieving it keeps `t`, `p`, `e`, `r` and costs four insertions (`ele` and `o`) plus two replacements (`a` → `r`, `m` → `t`).

2. When $q = 2$, the Jaccard similarity is $1/12 \approx 0.083$.

$$G_2(\texttt{tamper}) = \{\texttt{ta}, \texttt{am}, \texttt{mp}, \texttt{pe}, \texttt{er}\}$$
$$G_2(\texttt{teleporter}) = \{\texttt{te}, \texttt{el}, \texttt{le}, \texttt{ep}, \texttt{po}, \texttt{or}, \texttt{rt}, \texttt{er}\}$$

Only `er` is common, and the union has $5+8-1 = 12$ elements, so $J = 1/12$. Note that `teleporter` has 9 2-grams by position but only 8 *distinct* ones, because `te` occurs twice; the Jaccard similarity is computed on sets, so the repetition is counted once.

3. When $q = 3$, the Jaccard similarity is 0.

$$G_3(\texttt{tamper}) = \{\texttt{tam}, \texttt{amp}, \texttt{mpe}, \texttt{per}\}$$
$$G_3(\texttt{teleporter}) = \{\texttt{tel}, \texttt{ele}, \texttt{lep}, \texttt{epo}, \texttt{por}, \texttt{ort}, \texttt{rte}, \texttt{ter}\}$$

The two sets are disjoint, so $J = 0/12 = 0$. The two words are already dissimilar at $q = 2$, and increasing q makes matching strictly harder.

(b) You have a private collection of 1M documents $d_1, d_2, \ldots, d_n$, $n = 10^6$. You also crawled the Web and collected 100B documents $w_1, w_2, \ldots, w_N$, $N = 10^{11}$. You would like to find potential copyright violations (suspected copies of your private documents). Concretely, you want to find all pairs of documents (d_i, w_j) that have a Jaccard similarity ≥ 0.8:

$$J(d_i, w_j) \geq 0.8$$

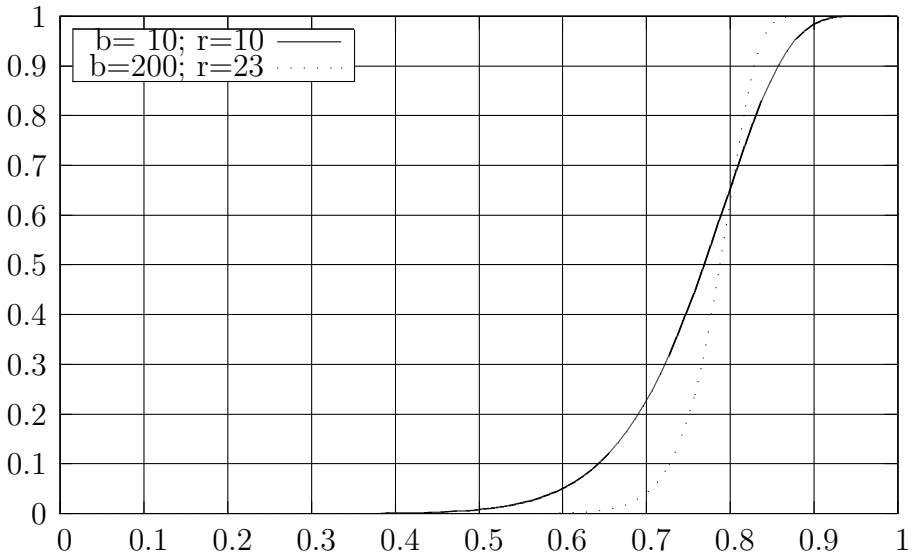
Brute-force would require that you compare $n \cdot N = 10^{17}$ pairs of documents. Instead of brute-force, you are using minhashes and LSH. The hash function that you use for LSH returns 10 bytes.

You consider two options:

Option 1 Use $m = 100$ minhashes. Split them into $b = 10$ bands of $r = 10$ minhashes each.

Option 2 Use $m = 4300$ minhashes. Split them into $b = 200$ bands of $r = 23$ minhashes each.

Recall that each band results in one hash value, which has 10 bytes in our case. For both options the inflexion point is at $\left(\frac{1}{b}\right)^{\frac{1}{r}} \approx 0.8$, see the graph below:



As usual, your algorithm based on LSH has four steps:

Step 1 Compute and store the LSH of all $n = 10^6$ private documents. (Recall that you need to store b hash values of 10 bytes each, per document.)

Step 2 Compute and store the LSH for all $N = 10^{11}$ Web documents.

Step 3 Find all pairs of documents (d_i, w_j) that have a common hash value.

Step 4 For each pair (d_i, w_j) of documents returned during step 3, compute the Jaccard similarity; if it is ≥ 0.8, then return the pair (d_i, w_j).

You expect the following:

- Approximately 10,000 pairs (d_i, w_j) have $J(d_i, w_j) = 0.85$. (If Step 3 fails to return such a pair, then it is called a *false negative*.)
- Approximately 10^{10} pairs (d_i, w_j) have $J(d_i, w_j) = 0.6$. (If Step 3 returns such a pair, then it is called a *false positive*.)
- The rest of the 10^{17} pairs (d_i, w_j) have a very small Jaccard similarity; we will assume that $J(d_i, w_j) = 0$.

Compute the cost of each option below. Use the graph to estimate the cost; you only need to give approximate answers. (If the graph gives a number that is to small to read, write *negligible*.)

Cost	Option 1	Option 2
Storage used by Step 1		
Storage used by Step 2		
# Total pairs returned by Step 3		
# False negatives		

Solution:

Cost	Option 1	Option 2
Storage used by Step 1	$100 \cdot 10^6 = 10^8$	$2000 \cdot 10^6 = 2 \cdot 10^9$
Storage used by Step 2	$100 \cdot 10^{11} = 10^{13}$	$2000 \cdot 10^{11} = 2 \cdot 10^{14}$
# Total pairs returned by Step 3	$5 \cdot 10^8$	negligible
# False negatives	1000	100 (or negligible)

244. *(from CSE 344, Winter 2019, Midterm)*

Answer each question below.

(a) For each statement below, indicate whether it is true or false:

i. The First Normal Form means that an attribute of a relation cannot be another relation.
True or false?

Solution: True

ii. In the relational data model the data is in First Normal Form.
True or false?

Solution: True

iii. In JSON the data is in First Normal Form.
True or false?

Solution: False

iv. Consider the relation `Product(productName, manufacturer, price, weight)` where the key is `(productName, manufacturer)`. Can there be two different products with the same value of `manufacturer`?
Yes or no?

Solution: Yes

v. If an attribute in a table is a foreign key, then the table cannot contain two tuples with the same value of that attribute.
True or false?

Solution: False

(b) In this and the following question we will assume that the relational algebra has set semantics. The relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples. You are asked to indicate the largest possible size of natural join $R \bowtie S$, in each of the cases below. Choose one of the following:

(a) m
(b) n
(c) mn
(d) $m + n$
(e) $\max(m, n)$
(f) $(m + n)/2$
(g) None of the above.

i. Assume that B is a key in S, and a foreign key in R. What is the largest possible size of the output of the natural join $R \bowtie S$?
Answer a-g:

Solution: (a)

ii. Assume that B is a key in R, and a foreign key in S. What is the largest possible size of the output of the natural join $R \bowtie S$?
Answer a-g:

Solution: (b)

iii. No keys or foreign keys were declared in R or S. What is the largest possible size of the output of the natural join $R \bowtie S$?

Answer a-g:

Solution: (c)

(c) For each question below, indicate whether it is true or false.

i. Consider a datalog program consisting of a single, non-recursive rule. Then the program can be rewritten in relational algebra.
True or false?

Solution: true

ii. Consider a datalog program consisting of multiple rules, including recursion. Then the program can be rewritten in relational algebra.
True or false?

Solution: false

245. *(from CSE 414, Spring 2019, Midterm)*

For each statement below, indicate whether it is true or false:

(a) *Physical data independence* means that the data is compressed.
True/False:

Solution: False

(b) *First Normal Form* means that an attribute of a relation cannot be another relation.
True or false?

Solution: True

(c) A relational database is always in First Normal Form.
True or false?

Solution: True

(d) JSON data is always in First Normal Form.
True or false?

Solution: False

(e) If an attribute is a foreign key, then no two tuples may have the same value of that attribute.
True or false?

Solution: False

(f) It is possible for a relation to have three different primary keys, for example each of A, B, and C is a primary key in $R(A, B, C, D)$.
True or false?

Solution: False

(g) It is possible for a relation to have three different foreign keys, for example each of A, B, and C in $R(A, B, C, D)$ is a foreign key.
True or false?

Solution: True

(h) Alice writes a SQL query that ends in . . . `group by A`.
but her query returns 1000 answers, too many to see on the screen. She makes a single change, replace the last line by . . . `group by A,B`.
She hopes to get fewer than 1000 answers. Will the new query return at least 1000 answer, or at most 1000 answers?
Answer "≥ 1000" or "≤ 1000" or "unknown":

Solution: ≥ 1000

(i) Suppose the attribute K is a key in R, and suppose the attribute FK in S is foreign key to R. Then the size of the join $R \bowtie_{R.K=S.FK} S$ is always less than or equal to the size of R.
True or false?

Solution: False

(j) Suppose the attribute K is a key in R, and suppose the attribute FK in S is foreign key to R. Then the size of the join $R \bowtie_{R.K=S.FK} S$ is always less than or equal to the size of S.

True or false?

Solution: True

246. *(from CSE 344, Spring 2026, Final)*

(a) What does *data independence mean*? Circle one answer:

1. The schema design should be done independently of previous applications.
2. SQL is designed independently of other languages.
3. The system can choose to represent the data and optimize the query independently of the user's logical view.
4. Transactions should run independently of each other, without interference.

Solution: Item 3.

(b) What does *schema normalization* (e.g. BCNF or 3NF) mean? Circle one answer:

1. The schema is modified to remove data anomalies.
2. This is the process by which we add indices to speedup queries.
3. This is a constraint that we write in SQL when we create tables.
4. It means that the query optimizer normalizes the query plan before execution.

Solution: Item 1.

(c) What is the *prepareStatement* command? Circle one answer:

1. It refers to the preparation we need to do before designing a new schema.
2. This is a statement that is placed at the beginning of each transaction.
3. This is the first phase of the hash-join operator.
4. It pre-compiles a SQL query template and then execute it multiple times with different data values of its parameters.

Solution: Item 4.

(d) Consider a hash-join $R \bowtie S$. What do the *probe-phase* and the *build-phase* refer to? Circle one answer:

1. The build-phase creates the hash table on R, then the probe phase iterates over S and finds matching tuples.

2. The probe-phase creates the hash table on R, then the build phase iterates over S and finds matching tuples.
3. The build-phase creates the hash table on S, then the probe phase iterates over R and finds matching tuples.
4. The probe-phase creates the hash table on S, then the build phase iterates over R and finds matching tuples.

Solution: Item 1.

(e) What does *block-nested loop join* mean? Circle one answer:

1. It is just another name for nested loop join.
2. One block may contain several nested blocks, and this can continue recursively, forming a tree.
3. It refers to a nested loop join where both tables are on disk, and the outer loop reads a large number of blocks at each iteration.
4. It is a particular variant of a join that uses a hash table.

Solution: Item 3.