Automatic Example Queries for Ad Hoc Databases

Bill Howe[1], Garret Cole[2], Nodira Khoussainova[3], Leilani Battle[4]

{[1]billhowe, [2]gbc3, [3]nodira, [4]leibatt}@cs.washington.edu, University of Washington, Seattle, WA, USA

http://escience.washington.edu

"Here is my data. Where do I start?"

Tabular data extracted from files, spreadsheets, DBs, the web

- No schema available
- No query logs available
- No DBAs available
- Unknown relationships, semantics, utility
- Temporary, time-sensitive applications

} *"Ad Hoc Databases"*

*An **ad hoc database** is a collection of tables with unknown relationships gathered to serve a specific, often transient, often urgent, purpose.*

Examples of Ad Hoc Databases

- A researcher assembles an ad hoc database of recent experimental results to prepare a paper or proposal.
- Emergency workers responding to a natural disaster assemble an ad hoc data-base from lists of addresses of nearby schools, locations of resources (e.g., ambulances), and contact information for emergency workers.
- A consulting business analyst assembles an ad hoc database from a set of spreadsheets provided by management for a short term engagement
- A security analyst assembles an ad hoc database from a set of application trace logs after an attack

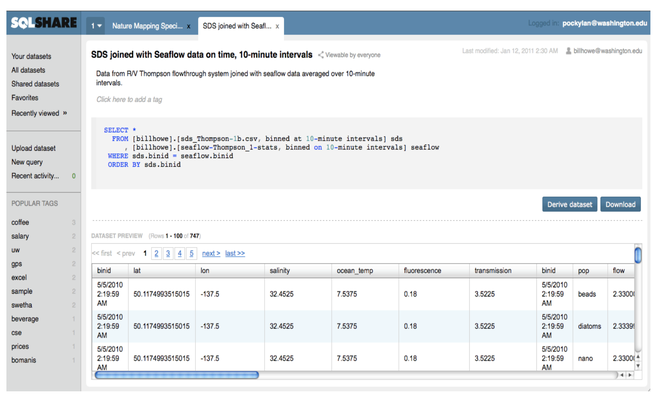
Approach

1. **Model** each operator independently using curated sts of example queries from the web
2. **Compose** operators to generate a search space of example queries
3. **Rank** each set using scores derived from configurable patterns called *idioms*

Q: Are users willing and able to write SQL?

A: Yes! But they need access to high-quality examples (c.f. Gray, Szalay et al. 2005; Howe 2010)

SQLShare: Database-as-a-Service for Ad Hoc Data



1. Streamlined for a single workflow:
2. No DDL; schema inferred from data
3. Views as first-class citizens
4. Unfettered sharing; cloud-hosted
5. Full SQL; no restrictions

http://sqlshare.escience.washington.edu

Upload → Query → Share

Modeling Each Operator

Join

Finding: "Good" joins characterized by linear relationships among a handful of set properties

Feature	Expression
max/min cardinality	$max/min(\|x\|, \|y\|)$
cardinality difference	$abs(\|x\| - \|y\|)$
intersection cardinality	$\|x \cap y\|$
union cardinality	$\|x \cup y\|$
Jaccard similarity	$\frac{\|x \cap y\|}{\|x \cup y\|}$

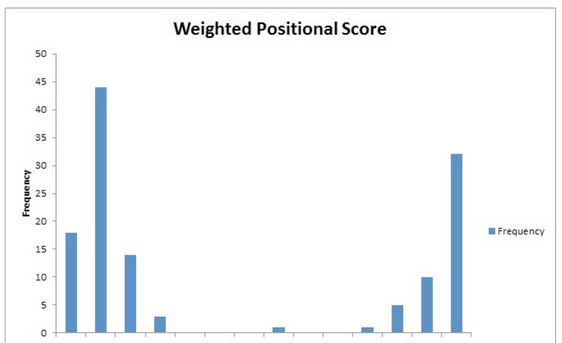
We train a decision tree over these features using existing sets of example queries (with > 80% precision and recall)

Buid a graph (V,E) where V is the set of tables and E is the set of "good" joins.

Each quey is a minimum spanning tree of a connected component of this graph.

Project

Finding: Important attributes appear near the far left or far right of the table.

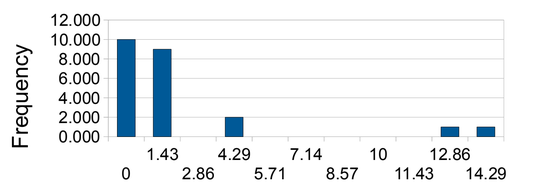


A histogram of the weighted positional scores of columns from the example queries of the SDSS database.

Select

Finding: Good queries return around 5-10% of the tuples in the table/join.

Histogram of Percentage of Tuples Returned by Example Queries for the SDSS Dataset



Histogram of Percentage of Tuples Returned by Example Queries for the GO Dataset

Group by

Grouping column: a column is selected if it has manageable distinct values.

Aggregates: In most cases, project count(*). If a separate numeric column is also discovered, demonstrate functions sum, min, max, and avg.

Union

Idea: two tables are good candidates for a union if they share sequence of columns with matching data types.

Finds more matches than just considering column name matches.

Evaluating "Good" Queries

What makes a good set of queries? It is application-dependent. e.g. for a DB class, the queries should demonstrate various SQL concepts vs. if user is familiar with SQL but not with the schema, better to have queries that refer to important tables or views.

What is an idiom? A function I : Q → [0, 1]. Takes a query and outputs score between 0 and 1. Examples:

- outputs 1 if query includes GROUP BY clause, 0 otherwise
- rewards queries with more joins
- outputs a score based on which important views the query references
- outputs 1 if query demonstrates a self-join, 0 otherwise

How to use idioms to select the set of starter queries?

Represent queries as vectors of idiom scores

	I_1	. . .	I_m
q_1	$I_1(q_1)$		$I_m(q_1)$
. . .		. . .	
q_n	$I_1(q_n)$		$I_m(q_n)$

Goals:

- maximize idiom scores
- select diverse set of queries

Score of a set of *k* queries:

$$w \cdot \sum_{i=1}^{k} \sum_{j=1}^{m} I_j(q_i) + (1-w) \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} d(q_i, q_j)$$

We use a greedy algorithm by selecting best query first, then iteratively add best additional query given the queries collected up to this point.