Optimizing Database-Backed Applications Using Query Synthesis

Alvin Cheung
Armando Solar-Lezama  Samuel Madden
MIT
Developing Database Applications

Application -> SQL Queries -> Database

Relations
Developing Database Applications

Application

ORM libraries

Database

Methods

SQL Queries

Objects

Relations
Developing Database Applications

Application

ORM libraries

Methods

Objects

SQL Queries

Relations

Lacks info about app

Inefficient queries

Database

Application

ORM libraries

Methods

Objects

SQL Queries

Relations

Lacks info about app

Inefficient queries

Database

Application

ORM libraries

Methods

Objects

SQL Queries

Relations

Lacks info about app

Inefficient queries

Database
Relational Operations in Imperative Code

List getUsersWithRoles () {
    List users = User.getAllUsers();
    List roles = Role.getAllRoles();
    List results = new ArrayList();
    for (User u : users) {
        for (Role r : roles) {
            if (u.roleId == r.id)
                results.add(u); }
    }
    return results; }

List getUsersWithRoles () {
    return executeQuery(
        "SELECT   u FROM user u, role r
        WHERE     u.roleId == r.id
        ORDER BY u.roleId, r.id"; }

convert to

SELECT * FROM user
SELECT * FROM role
Relational Operations in Imperative Code

List getUsersWithRoles () {
    List users = User.getAllUsers();
    List roles = Role.getAllRoles();
    List results = new ArrayList();
    for (User u : users) {
        for (Role r : roles) {
            if (u.roleId == r.id) {
                results.add(u); }
        }
    }
    return results; }

Goal
Find a post-condition that we can rewrite into a SQL expression

List getUsersWithRoles () {
    return executeQuery(
        "SELECT   u FROM user u, role r
        WHERE     u.roleId == r.id
        ORDER BY u.roleId, r.id"; }

close output variable
Query By Synthesis (QBS)

• Identify potential code fragments
  – i.e., regions of code that fetches persistent data and return values

• Find SQL exprs for output variables

• Try to prove that those expressions preserve program semantics
  – if so, convert the code!
Query By Synthesis (QBS)

• Identify potential code fragments
  – i.e., regions of code that fetches persistent data and return values

  **Code pre-processing**

• Find SQL exprs for output variables
  **Language for verification conditions**

• Try to prove that those expressions preserve program semantics
  – if so, convert the code

  **Compute verification conditions**
Language for Verification Conditions
Language Requirements

• Compute verification conditions of imperative code fragment

• Handle relational operations as well
  – Order of records matter!

• Output variable expressions must be translatable to SQL

• Expressions should be easily synthesized
Language Design

- Relational Algebra
  - models database operations
  - translates to SQL
  - preserves record order
  - does not express VCs of imperative code

- First-Order Logic
  - models all operations
  - preserves record order
  - easy to synthesize
  - question mark for translates to SQL
Theory of Ordered Relations (TOR)

• Similar to relational algebra
• Model relations as ordered lists

L := program var | [] | e := L[i]
| L : L | L : e | e op e
| top_e(L) | max(L) | min(L)
| L \times_f L | \sigma_f(L) | sum(L) | avg(L)
| \pi_f(L) | order_e(L) | size(L)
• Semantics defined using axioms, e.g.:
  \[ \text{top}_i([],) = [] \]
  \[ \text{top}_i(L) = [] \quad \text{if } i = 0 \]
  \[ \text{top}_i(h : L) = h : \text{top}_{i-1}(L) \quad \text{if } i > 0 \]
Computing Verification Conditions

- Standard Hoare logic rules

- Treat loop invariants and post-conditions for output variables as function calls
  - Leave function bodies to be synthesized
Example

List getUsersWithRoles () {
    List users = query(select * from users);
    List roles = query(select * from roles);
    List results = [];
    for (User u : users) {
        for (Role r : roles) {
            if (u.roleId == r.id)
                results = results : []
        }
    }
    return results; }

Verification conditions

assume(preCondition = true)
preCondition ⇒
    outerInvariant(users/query(...), results/[], ...) 
outerInvariant(...) ∧ outer loop terminates ⇒
    results = postCondition(users, roles) ...
Synthesizing Expressions
Synthesis Templates for Invariants and Post-conditions

• Template for invariants:
  \[
  \Lambda (\text{<variable in scope>} = \text{<TOR expr>})
  \]
  – Only consider expressions that type check

• Template for post-conditions:
  \[
  \text{<output variable>} = \text{<TOR expr>}
  \]
  – Limit to TOR expressions that are translatable to SQL
Speeding up Synthesis

• Solve incrementally
  – Increase complexity of expression templates iteratively

• Break symmetries
  – Use relational equivalences, e.g.:
    • $\sigma_f(\sigma_g(L)) = \sigma_g(\sigma_f(L))$
      template only include one of the expressions
Initial Code Fragments
Identification

• Find program points that retrieve persistent data
• Run pointer analysis
• Determine where persistent data flow to
• Delimit start and end of code fragment to analyze
• Convert to kernel language
QBS Toolchain

Application source + config files

Entry Point Identifier

Identified persistent data methods

Code Fragment Identifier

Kernel Language Compiler

Identified persistent data methods

Code fragment to be analyzed

Code fragment

Inferred SQL

VC Computation

Invariant + Post-condition Synthesizer

Transformed method body
Experiments
Experiment Setup

• No standard benchmarks available

• Experimented on two large scale open source web applications
  – How many code fragments can be converted
  – Difference in page load times while scaling to different database sizes
# Real-world Evaluation

## Wilos (project management application) – 62k LOC

<table>
<thead>
<tr>
<th>Operation type</th>
<th># Fragments found</th>
<th># Fragments converted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Selection</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>Join</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Aggregation</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>33</strong></td>
<td><strong>28</strong></td>
</tr>
</tbody>
</table>
# Real-world Evaluation

**iTracker** (bug tracking system) – 61k LOC

<table>
<thead>
<tr>
<th>Operation type</th>
<th># Fragments found</th>
<th># Fragments converted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Selection</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Join</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Aggregation</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>16</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>
Performance Evaluation: Selection Query

50% selectivity

10% selectivity

Page load time (ms)

Total number of users in DB

Original

Inferred

0 20K 40K 60K 80K 100K

0 20K 40K 60K 80K 100K

100 1K 10K
Performance Evaluation: Join Query

- Nested-loop join $\Rightarrow$ Hash join!
  - $O(n^2)$
  - $O(n)$
Failed Code Fragments

- Custom comparators
- Use database schema information

```java
List records = Query("SELECT id FROM t");
List results = new ArrayList();
Collections.sort(records); // sort by id
int i = 0;
while (records.get(i).id < 10) {
    results.add(records.get(i)); ++i;
}

→ SELECT id FROM t ORDER BY id LIMIT 10
```
Query By Synthesis

Convert imperative program statements into declarative SQL

Shows substantial improvement in real-world applications

Illustrates power of synthesis in enabling complex optimizations