Extracting Queries by Static Analysis of Transparent Persistence

Ben Wiedermann and William R. Cook
The University of Texas at Austin
Databases

Programming Languages
Databases

Programming Languages
Databases

Print name and manager’s name of every employee whose salary > $65,000

Programming Languages
Database APIs (JDBC, etc)

Print name and manager’s name of every employee whose salary > $65,000

Transparent Persistence
String query = "from Employee e 
  left join fetch e.manager 
  where e.salary > 65000";
List result = session.createQuery(query);
for (Employee e : result.list()) {
    print(e.name + e.manager.name);
}
String query = "from Employee e 
    left join fetch e.manager 
    where e.salary > 65000";
List result = session.createQuery(query);
for (Employee e : result.list()) {
    print(e.name + e.manager.name);
}
for (Employee e : root.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
for (Employee e : root.employees) {
   if (e.salary > 65000) {
      print (e.name + e.manager.name);
   }
}
Database APIs

Transparent Persistence
- Not true integration
- Not type-safe
- Burdens programmer
- Good performance

- Better integration
- Type safe
- Relieves programmer burden
- Poor performance
Query Extraction

- Good performance
- Better integration
- Type safe
- Relieves programmer burden
String query = "from Employee e 
                left join fetch e.manager
                where e.salary > 65000";
List result = session.createQuery(query);
for (Employee e : result.list()) {
    print(e.name + e.manager.name);
}

for (Employee e : root.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
Approach

• Identify subset of the database used by program
  • Traversals from root define shape of query
  • Identify conditions under which data is used
• Current Assumptions
  • 1 transaction per program
  • Query result has same structure as database
Object View of Database
Object View of Database

Retrieve subgraph program requires
Simple Study Language

- Transparent persistence
- Access through variable `root`
- Imperative
  
  ```
  x := y + z;
  ```
- No database updates
- Iteration over persistent collections
  
  ```
  for e in root.employees {...}
  ```
- No procedures
for (Employee e : root.employees) {
   if (e.salary > 65000) {
      print (e.name + e.manager.name);
   }
}

employees
employees.t
employees.t.salary
employees.t.manager
employees.t.manager.name
Abstract Interpretation

- **Paths** describe **data**
- Concretization = query execution
- Computes **sound** over-approximation
- Field traversal generates new path(s)
- Merge conditional branches
- Merge assignments
How Precise Are Paths?

Need more precise approximation
for (Employee e : root.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
Query Condition Restrictions

- Executable by database
- Independent of collection order
- Require no sub-select queries
- Query results reflect database structure
Order Independence

for $l_1$ in $p$

\[ x := E[l_1] + x; \]

if $C[l_1, x]$ then $S$;
No Sub-select Queries

for $l_1$ in $p$
    if $C_1[l_1]$ then
        $x := E[l_1]$;
    for $l_2$ in $p$
        if $C_2[l_2, x]$ then $S$;
Query Results Reflect Structure

for $l_1$ in $p_1$
  for $l_2$ in $p_2$
    if $C[l_1, l_2]$ then $S$;
Abstract Interpretation

- Domain: Path $\times$ Condition
- Field traversal generates new path(s)
- Merge conditional branches
- Merge assignments
- Attach query conditions to paths
Example

for (Employee e : root.employees) {
   if (e.salary > 65000) {
      print (e.name + e.manager.name);
   }
}

employees
employees.t
employees.t.salary
employees.t.manager
employees.t.manager.name

C = employees.t.salary > 65000
Query Creation

```
struct (employees = (
    select struct (salary = e.salary,
        name = e.name,
        manager = struct(name = e.manager.name))
    from employees as e
    where e.salary > 65000)

employees        employees.t.name [C]
employees.t       employees.t.manager [C]
employees.t.salary employees.t.manager.name [C]

C = employees.t.salary > 65000
```
Program Creation

qs = "struct (employees = (select struct (salary = e.salary,
                               name = e.name,
                               manager = struct(name =
                                                e.manager.name))
                               from employees as e
                               where e.salary > 65000)";

result = session.executeQuery(qs);

for (Employee e : root.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
Program Creation

qs = "struct (employees = (  
    select struct (salary = e.salary,  
        name = e.name,  
        manager = struct(name =  
            e.manager.name))  
    from employees as e  
    where e.salary > 65000)";

result = session.executeQuery(qs);

for (Employee e : result.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
Condition Removal

qs = "struct (employees = ( 
    select struct (salary = e.salary, 
        name = e.name, 
        manager = struct(name = 
            e.manager.name)) 
    from employees as e 
    where e.salary > 65000)"

result = session.executeQuery(qs);

for (Employee e : result.employees) {
    if (e.salary > 65000) {
        print (e.name + e.manager.name);
    }
}
Condition Removal

qs = "struct (employees = (  
select struct (salary = e.salary,  
    name = e.name,  
    manager = struct(name =  
       e.manager.name))  
from employees as e  
where e.salary > 65000)";  
result = session.executeQuery(qs);  
for (Employee e : result.employees) {  
   print (e.name + e.manager.name);  
}
Query Simplification

qs = "struct (employees = (
  select struct (salary = e.salary,
  name = e.name,
  manager = struct(name = e.manager.name))
  from employees as e
  where e.salary > 65000)"

result = session.executeQuery(qs);

for (Employee e : result.employees) {
  print (e.name + e.manager.name);
}
Query Simplification

qs = "struct (employees = ( 
  select struct ( 
    name = e.name, 
    manager = struct(name = 
      e.manager.name)) 
  from employees as e 
  where e.salary > 65000)";
result = session.executeQuery(qs);

for (Employee e : result.employees) {
    print (e.name + e.manager.name);
}
Related Work

• Shape Analysis for Data Access
  • Vitenberg, Kvilekval, and Singh [ECOOP04]
  • Kvilekval and Singh [DOA04]

• Queries as First-Class Program Values
  • Bierman, Meijer, and Schulte [ECOOP05]
  • Cooper, Lindley, Wadler, and Yallop (Links)
  • Cook and Rai [ICSE05]
  • Willis, Pearce, and Noble [ECOOP06]
Future Work

- Inter-procedural analysis
- Multiple queries
- Implementation / evaluation
- Persistent update
- More expressive queries
Databases

Programming Languages
Query Extraction