Extensible Query Processing in Starburst

Presentation: Charles
Discussion: Cong

Outline

1 Motivation
2 Overview of Starburst
3 Language Processing
4 Query Graph Model
5 Query Rewrite
6 Summary

Motivation

To bridge the gap between relational DBMSs and Apps

What Starburst Provides

- **Extensibility**
  - Language extensions
    (e.g., datatypes and operations)
  - Data management extensions
    (e.g., access and storage methods)
  - Internal processing extensions
    (e.g, join methods and query transformations)

Two Major Components

- **Corona: query language processor**
  Roughly like RDS in System R
  (parsing, semantic analysis …)
- **Core: data manager**
  Roughly like RSS in System R
  (record management, buffer …)

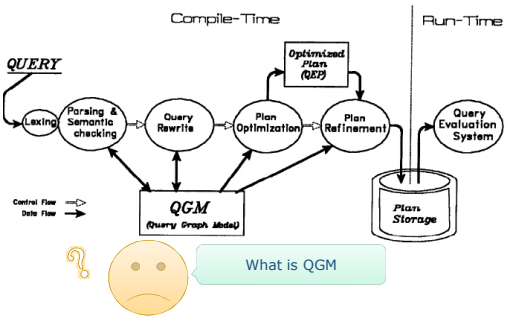
Starburst's Language: Hydrogen

- **Based on SQL**
- **Aspect1: Orthogonal**
  Objective: any operation on table produces a table, which can be used anywhere
- **Aspect2: Extensibility**
  Objective: Allow new functionality added in columns, tables & data types

Pros & Cons of Hydrogen

- **Pros: Expressiveness!**
- **Cons:**
  1. **Complex, hard to read & write**
  2. **Hard to identify equivalent expressions**

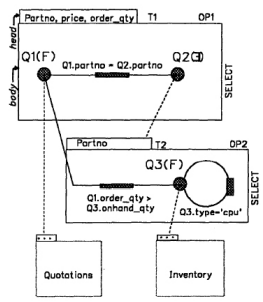
Overview of Language Processing



Query Graph Model

- **Objective: Providing an *understandable, detailed* and *unambiguous* semantic representation of a query**
- **Modifiable by DBCs to add extensions!**

Query Graph Model Example



- Vertices
- Edges
- boxes

```
SELECT partno, price, order_qty
 FROM quotations Q1
 WHERE Q1.partno IN
   (SELECT partno
     FROM inventory Q3
     WHERE Q3.onhand_qty < Q1.order_qty
           AND Q3.type = 'cpu')
```


Discussion Question 1 – DBC & Extensibility

The paper claimed that most extensions in Starburst would be made by a DataBase Customizer or (DBC). DBCs are knowledgeable database implementers; they are neither end-users nor skilled application programmers.

(1) In terms of extensibility

Do you think it is a good idea to target extensibility features only at a specialized class of "database expert" developers (and not ordinary developers)?

(2) Other possible problems

Would the introduction of a knowledge DBC cause some other problems?

-Contradicting Codd's goal of data management in the abstracted high level?
-Bugs in implementation of new types/functions?
-Potential security problems?
-…

Query Rewrite

- **Nonprocedural Languages should perform equivalently**
- **To reduce complexity of queries**
- **Rule-based approach to make Hydrogen extensive**

Rule-based Approach

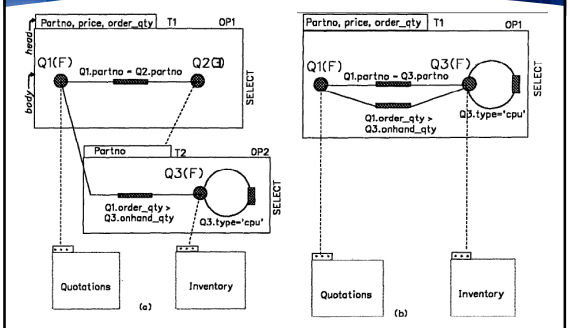
- **Rule language is C**
- **Two parts: condition and action, each written a C function**

Rule 1 (Subquery to Join):[3]
IF OP1.type = Select ∧ Q2.type = '∃' ∧
(at each evaluation of the existential predicate
at most one tuple of T2 satisfies the predicate)
THEN
Q2.type = 'F'; /*convert to join*/

Rules- three classes

- **Predicate migration**
- **Projection push-down**
- **Operation merging**
  - **Merge QGM "boxes"**

Query Rewrite Example



Rule Engine

- **Meet query rewrite rules in general**
- **Forward chaining**
- **Several control strategies:**
  - Sequential
  - Priority
  - Statistical
- **budget**

Query Rewrite Characteristics

- **Interact with plan optimization to estimate cost of alternative QGM representations**
- **Cannot prune alternatives until optimization**

Plan Optimization

- **Estimate cost of query evaluation plan**
- **3 major aspects**
  plan generation

  plan costing

  search strategy
- **Designed Orthogonal**

STARS – a plan generation

- Strategy alternative rules (STARs)
- A general-purpose STAR evaluator
- A search strategy that chooses the next STAR to evaluate
- An array of STARs
- Express all strategies plus new ones

Summary

- Starburst: extensible DBMS
- Extensions to the language, the language processing and the data manager
- Orthogonality + Table Abstraction (QGM) = Success

Summary

- QGM simplifies the DBC's task, give him a great deal of flexibility and power
- Rule-based query rewrite
- Grammar-like rules to generate plans

Discussion Question 2 – Completeness

The authors state (at the beginning of the third paragraph of section 5) that "*It is unlikely that we will ever have a complete set of relational transformations, even for a standard relational language*". Please give your comments on:

(1) Why do you think they say this?

(2) Do you believe it?

YES: What prevents us from having the complete set of transformations?

NO: What would a complete set even look like?

Thank you!

Extensible Query Processing in Starburst