An Extensible Rule Transformation Model for XQuery Optimization

Rules Pattern for XQuery Tree Graph View

Nicolas Travers\textsuperscript{1} Tuyêt Trâm Dang Ngoc\textsuperscript{2}

(1) PRiSM Laboratory-University of Versailles, France
(2) ETIS Laboratory - University of Cergy-Pontoise, France

14 June 2007

ICEIS 2007
Plan

1. Context
2. Extensible Optimization
3. Performances
4. Conclusion
1. **Context**
   - A Distributed System
   - Querying data sources
   - Motivations

2. **Extensible Optimization**

3. **Performances**

4. **Conclusion**
A Distributed System

XQuery Evaluation

Data Sources:
- Distributed;
- Heterogeneous.

Problems:
- XQuery;
- Data localization;
- Query optimization;
- Adaptability.
Tree Graph View (TGV) [DASFAA 2007]

**TGV** : a *query model* for XQuery.
- Based on Tree Pattern matching ;
- Describes selection and construction patterns ;
- Defines projections and associations between patterns ;
- Identifies groups of treatments ;

A TGV is a direct translation of an XQuery query.
TGV examples (Logical TGVs)

$\text{i} \quad \text{catalog}$

$\text{book}$

author = "Robin Hobb"

price = 

@isbn

$\text{j} \quad \text{book}$

@isbn

name

date 

>"2006-11-20"

"locations.xml"

@isbn

name

date 

>"2006-11-20"

"locations.xml"
Querying data sources

TGV examples (Logical TGVs)

N. Travers, T. T. Dang-Ngoc, T. Liu

Rules Pattern for XQuery Tree Graph View
Annotations on TGV:

- Localization.
- Cost model;
Annotations on TGV:
- Localization.
- Cost model.

(1) source 1

(2) source 2
Annotation Support (Physical TGVs)

Annotations on TGV:
- Localization.
- Cost model.

\begin{align*}
(1) \text{cost}_1 &= \text{card}_i \times IO \\
(2) \text{cost}_2 &= \text{cost}_1 \times \text{sel}(\text{author}) \\
(3) \text{cost}_3 &= \text{card}_i \times IO \\
(4) \text{cost}_4 &= \text{cost}_3 \times \text{sel}(\text{date}) \\
(5) \text{cost}_5 &= \text{CPU} \times \text{card} \left( (1) \times (2) \right) \\
(6) \text{cost}_6 &= \max(\text{cost}_2, \text{cost}_4) \\
(7) \text{cost}_7 &= \text{const} + \text{cost}_5 + \text{cost}_6
\end{align*}
Motivations of TGV optimization

- TGV simplifies XQuery conception, but needs to:
  - Define transformations;
  - Generate better evaluations;
- Distributed context:
  - Identify and optimize sub-queries;
  - Distributed optimization;
  - Adapt the optimizer (mediation, P2P, mobile agents...).
Motivations of TGV optimization

- TGV simplifies XQuery conception, but needs to:
  - Define transformations;
  - Generate better evaluations;
- Distributed context:
  - Identify and optimize sub-queries;
  - Distributed optimization;
  - Adapt the optimizer (mediation, P2P, mobile agents...).

An Extensible Optimizer corresponds to this needs.
1 Context

2 Extensible Optimization
   • Optimizer Conception
   • TGV Optimization

3 Performances

4 Conclusion
What is needed?

Search Strategy of the best plan, relies on:

- Equivalent representations;
- Costs types (execution time, communication, price);
- Transformation rules on TGVs;
- Distributed context.
What is needed?

Search Strategy of the best plan, relies on:

- Equivalent representations;
- Costs types (execution time, communication, price);
- Transformation rules on TGVs;
- Distributed context.
Optimizer Conception

What is needed?

Search Strategy of the best plan, relies on:

- Equivalent representations;
- Costs types (execution time, communication, price);
- Transformation rules on TGVs;
- Distributed context.
What is needed?

Search Strategy of the best plan, relies on:

- Equivalent representations;
- Costs types (execution time, communication, price);
- Transformation rules on TGVs;
- Distributed context.
What is needed?

Search Strategy of the best plan, relies on:

- Equivalent representations;
- Costs types (execution time, communication, price);
- Transformation rules on TGVs;
- Distributed context.
Static and Extensible Optimizers

- Standard optimizers: a set of rules and a strategy;
- Extensible optimizers: Rules and Strategies can be modified:
  - Emphasize Search Space;
  - Improve optimizer;
  - Adapt the system;

An extensible optimizer needs:
- Transformation rules;
- Search strategy(ies);
- Cost model(s);
Static and Extensible Optimizers

- Standard optimizers: a set of rules and a strategy;
- Extensible optimizers: Rules and Strategies can be modified:
  - Emphasize Search Space;
  - Improve optimizer;
  - Adapt the system;

An extensible optimizer needs:
- Transformation rules;
- Search strategy(ies);
- Cost model(s);
Static and Extensible Optimizers

- Standard optimizers: a set of rules and a strategy;
- Extensible optimizers: Rules and Strategies can be modified:
  - Emphasize Search Space;
  - Improve optimizer;
  - Adapt the system;

An extensible optimizer needs:
- Transformation rules;
- Search strategy(ies);
- Cost model(s);
Static and Extensible Optimizers

- Standard optimizers: a set of rules and a strategy;
- Extensible optimizers: Rules and Strategies can be modified:
  - Emphasize Search Space;
  - Improve optimizer;
  - Adapt the system;

An extensible optimizer needs:
- Transformation rules;
- Search strategy(ies);
- Cost model(s);
Static and Extensible Optimizers

- Standard optimizers: a set of rules and a strategy;
- Extensible optimizers: Rules and Strategies can be modified:
  - Emphasize Search Space;
  - Improve optimizer;
  - Adapt the system;

An extensible optimizer needs:
- Transformation rules;
- Search strategy(ies);
- Cost model(s);
Existing Extensible Optimizers

- Exodus [Carey et al. 90]:
  - Search strategy (improvement coefficient);

- Volcano [Graefe et McKenna 93]:
  - Defines search strategies;

- OPT++ [Kabra et DeWitt 99]:
  - Object Oriented;
  - Defines search strategies;
  - *logical* and *physical* optimization.
Existing Extensible Optimizers

- **Exodus** [Carey et al. 90]:
  - Search strategy (improvement coefficient);

- **Volcano** [Graefe et McKenna 93]:
  - Defines search strategies;

- **OPT++** [Kabra et DeWitt 99]:
  - Object Oriented;
  - Defines search strategies;
  - *logical and physical* optimization.
Existing Extensible Optimizers

- Exodus [Carey et al. 90]:
  - Search strategy (improvement coefficient);

- Volcano [Graefe et McKenna 93]:
  - Defines search strategies;

- OPT++ [Kabra et DeWitt 99]:
  - Object Oriented;
  - Defines search strategies;
  - *logical* and *physical* optimization.
We propose a framework for defining and modeling rules:

**Condition Rule Pattern** \( \Rightarrow \) **Conclusion Rule Pattern**

**Rule Patterns**: must be found into a TGV.

Two types of transformation:
- Logicals ;
- Physicals ;
Transformation Rules

Logical Optimization: predicate duplication

R1: $n_1 \overset{c_1}{=} n_2 \Rightarrow n_1 \overset{c_1}{=} n_2$
Transformation Rules

Logical Optimization : predicate duplication

R1: \[ \begin{align*}
  & n_1 \quad c_1 \\
  & = \quad H_1 \\
  & n_2 \\
  \Rightarrow \\
  & n_1 \quad c_1 \\
  & = \quad H_1 \\
  & n_2 \quad c_2
\end{align*} \]

N. Travers, T. T. Dang-Ngoc, T. Liu

Rules Pattern for XQuery Tree Graph View
Transformation Rules

Logical Optimization: predicate duplication

R1: \( \frac{\text{condition} \Rightarrow \text{place}}{\text{predicate} = \text{predicate}} \)

\[
R1: \frac{n_1 \text{ condition} \Rightarrow \text{place}}{n_1 \text{ predicate} = n_2 \text{ predicate}}
\]

N. Travers, T. T. Dang-Ngoc, T. Liu

Rules Pattern for XQuery Tree Graph View
Transformation Rules

Physical Optimization: Algorithm modification

R2: \( \frac{n_1}{H_1} = \frac{n_2}{H_1} \) => \( \frac{n_1}{H_1} = \frac{n_2}{H_1} \)

(1) Left Outer Join
(2) Card
(3) Card

(1) Bind Left Outer Join
(2) Card
(3) Card
Transformation Rules

Physical Optimization: Algorithm modification

R2: $n_1 = n_2$ => $H_1 = H_2$

(1) Left Outer Join
(2) Card
(3) Card

(1) Bind Left Outer Join
(2) Card
(3) Card

N. Travers, T. T. Dang-Ngoc, T. Liu
Rules Pattern for XQuery Tree Graph View
TGV Optimization

Transformation Rules

Physical Optimization : Algorithm modification

\[ R2 : \quad \frac{n_1}{H_1} = \frac{n_2}{H_2} \quad \Rightarrow \quad \frac{n_1}{H_1} = \frac{n_2}{H_2} \]

(1) Left Outer Join
(2) Card
(3) Card

(1) Bind Left Outer Join
(2) Card
(3) Card

\[ \text{(1)} \quad \text{Card : 10} \]
\[ \text{(2)} \quad \text{Algorithm : Bind Left Outer Join} \]
\[ \text{(3)} \quad \text{Card : 10000} \]
Search Strategy

Heuristic: **improvement coefficient** on each rule.
- Estimation on: before/after transformation;
- Determined by rules calibration;

Strategy to direct space generation:
- Incremental;
- Choose the best applicable coefficient;
Search Strategy

Heuristic: **improvement coefficient** on each rule.

- Estimation on: before/after transformation;
- Determined by rules calibration;

**Strategy to direct space generation:**

- Incremental;
- Choose the best applicable coefficient;
1. Context

2. Extensible Optimization

3. Performances

4. Conclusion
Rules Application
1 Context
2 Extensible Optimization
3 Performances
4 Conclusion
Conclusion

- Rule Patterns:
  - A framework to define rules;
  - Makes the optimizer extensible and adaptive;
  - Rely on TGV (intuitive);

- Search Strategy:
  - Improvement Coefficient;
  - Incremental.