Maintenance Policy Selection in Heterogeneous Data Warehouse Environments: A Heuristics-based Approach

H. Engström
University of Skövde, Sweden
S. Chakravarthy
UTA, USA
B. Lings
University of Exeter, U.K.

Outline

- Introduction
- Problem Description
- Previous Work
- Method
- Results
- Conclusions
Introduction

- Maintenance of views over distributed, heterogeneous, and autonomous sources
  - **Note**: Not the typical DW assumptions
- Most previous research focus on immediate, incremental policies and consistency
- Our questions:
  - How important is consistency?
  - Are incremental policies the only choice?
  - What are the implications of autonomy?

Problem Description

- **Main problem**: how do we select a *good* maintenance policy for views based on distributed, heterogeneous, and autonomous sources?
- **Consider**: set of policies, evaluation criteria, source capabilities
- **Remember**: A maintenance policy has to be selected – explicitly or implicitly
Our Previous Work

• For a single source view we have:
  • Established a framework: characterized relevant policies, quality of service (QoS) criteria, and source capabilities
  • Developed a cost-model
  • Analysed policy selection based on the cost-model
  • Validated dependencies empirically using a test-bed system
• This has been done for heterogeneous sources

Autonomy

• A data source can be more or less autonomous:
  • Only queries are allowed
  • Schema changes possible: e.g. adding triggers
  • API available
  • Source code available
• We assume maximal source autonomy
• A wrapper may be used to extend the source and change its interface - but this may have implications
Policies

- Timings
  - Immediate (on commit)
  - Periodic
  - On-demand
- Strategies
  - Incremental
  - Recompute

Combined this gives six different policies

Evaluation Criteria

- Relevant evaluation criteria include QoS as well as system overhead aspects
- We consider three different quality of service properties:
  - Consistency
  - Staleness
  - Response time
- In addition we consider system overhead (processing, storage and communication) in sources and client
Source Capabilities

- A source may have different capabilities to support maintenance, for example:
  - It may notify (immediately) an external client whenever the source is changed
  - It may deliver changes (delta) that have been committed since last maintenance
  - It may provide the date (time-stamp) of the last change
  - It may be queryable and deliver the desired set of data
- We make no assumptions on available capabilities
  - A source can have any combination of the above capabilities

Results for a Single Source View

- Source capabilities impact on policy selection
  - Wrapping does not always come to the rescue
- Incremental policies are not always optimal
  - The source has to provide deltas
- Immediate maintenance is rarely possible to use
  - Periodic policies may be the best surrogate but setting of periodicity is difficult
- Staleness is an important QoS criteria
This Study

- We extend previous work and study a join view
- Sources are heterogeneous, distributed and autonomous

Example Application - Biological Data Integration

- Data is collected from several autonomous (and heterogeneous) sources

```
Query for sequences that match a particular non-PROSITE pattern

Internet
http://www.expasy.org

PROSITE

SWISS-PROT

Local DB with sequences of interest

Client application

http://www.ida.his.se/ida/mama

Patter DB (MAMA)
```
Extending the Framework

- **Policies:**
  - Each source contributes with a single source view (supporting view) which can be maintained with a policy
  - The integrator can do the joining with different policies
  - Auxiliary views may be used (store supporting views)
  - Combined it gives rise to a large number of policies
  - We have considered all principal types of policies (84 different)

- **Evaluation criteria:**
  - Policies may provide different degrees of consistency
  - Some policies are shown to provide strong consistency other require compensation

- **Source capabilities:**
  - A source may support “join queries”

- **Join technique may have an impact**
  - We consider nested loop and hash-based join
Analysing Policies

- As before, the aim is to support policy selection
- Method:
  - Extend the cost model
  - Develop a tool (PAM) based on the cost model
  - Explore the multidimensional search space using the tool
  - Identify general properties
  - Validate them empirically
- As the solution space is huge we focus on producing usable heuristics

Example – Analysing Policy Selection

Policy 1: Immediate incremental with auxiliary views
Policy 2: on-demand recompute without auxiliary views

All (630000 cases)
Policy 1: 96%
Policy 2: 4%

Hash-based join (315000 cases)
Policy 1: 92%
Policy 2: 8%

Nested loop join (315000 cases)
Policy 1: 100%
Policy 2: 0%

No source provide deltas (78750 cases)
Policy 1: 78%
Policy 2: 22%
Source 1 provides deltas (78750 cases)
Policy 1: 95%
Policy 2: 5%
Source 2 provides deltas (78750 cases)
Policy 1: 95%
Policy 2: 5%
Both sources provide deltas (78750 cases)
Policy 1: 100%
Policy 2: 0%
Results

- Many different policies can be optimal
- Based on analysis we propose a set of heuristics, for example:
  - Use auxiliary views unless storage is very critical
  - For most cases: use incremental maintenance
  - Make use of relaxed staleness requirements
- The heuristics have been captured in a selection process
Validation

- Empirical validation by comparing all types of policies in a testbed (TMID) with different source configurations
  - Relational and XML
  - Different source capabilities
  - On Linux and Solaris
- Quality of the selected policy:
  - Let $max$ and $min$ be the worst and best measured performance respectively (among the 84 policies)
  - Let $x$ be the measured performance of the selected policy
  - Then the quality is: $100 \times \frac{max - x}{max - min}$

Selection Quality

The quality of the selected policy in 48 different source and QoS scenarios
Result - Validation

- Heuristics give good policies in most cases
- Bad policies are always avoided
- Heuristics is significantly better than an ad hoc approach
- Analytical observations can be validated empirically

Conclusions

- Policy selection is a complex problem
- Heuristics are useful
  - Incremental policies are generally to prefer!
  - Immediate policies are rarely possible to use
  - Staleness is important for selection
  - Consistency is not a key factor for the problem
- Much remains to be studied
  - Real data
  - Real network environment (LAN, WAN)
  - Extend and refine heuristics