Automating Database Schema Evolution in Information System Upgrades

Carlo A. Curino

In collaboration with:
Hyun J. Moon, Carlo Zaniolo

Oct 25, 2009
Motivations

*Panta Rhei*—*everything is in a state of flux.* (Heraclitus, 536-470 b.c.)

- **Information Systems (IS) are subject of continuous evolution:**
  - business reality forces IS to evolve frequently
  - requirements modify to adapt to a changing environment

- **Evolution is a difficult and expensive task:**
  - software evolution/maintenance represent 40% to 75% of the overall costs

- **The data management core:**
  - schema evolution: gracefully support changes to DB schemas
  - data evolution: historical archiving of a DB content
• **Starting point:**
  
  • a Schema $S_1$,
  
  • a database $DB_1$ under $S_1$, and
  
  • a set of queries $Q_1$ formulated over $S_1$
Evolution in the **real** world:

- The DBA defines an SQL DDL script modifying S1 into S2
- The DBA defines an SQL DML script migrating data from DB1 to DB2
- **Queries** in Q1 might fail, the DBA *adapts* them **manually**: $Q2 = Q1' + Q2_{\text{new}}$ (new queries added on S2)
• **Evolution in the real world:**
  
  - The DBA defines an SQL DDL script modifying S2 into S3
  - The DBA defines an SQL DML script migrating data from DB2 to DB3
  - Queries in Q2 might fail, the DBA adapts them manually as in $Q_3 = Q_2' + Q_{3\_new}$ (new queries added on S3)
DB Administrator (DBA) nightmares:

- **Data Migration**: Data loss, redundancy, efficiency of the migration, efficiency of the new design
- **Impact on Queries** and applications
- *What is the real impact of schema evolution? [iceis2008]*
Schema Evolution Case Study: Wikipedia DB

- **Analysis of Information Systems evolution:**
  - Wikipedia Backend: relational, open source, intense (170+ schema version in 4.5 years), need for archival (~30% of the schema)
  - Hundreds of other IS currently studied (NFS funding to create a benchmark for schema evolution)

- **Lessons learned:**
  - Schema Evolution is a real problem!
  - Distributed, collaborative environment increase need for SE
  - Open Source systems give us an unprecedented vantage point!
• Evolution in an *ideal* world:
  
  • Evolution design is **assisted** and **predictable**
  
  • Data migration scripts are **generated** automatically
  
  • Legacy Queries (and updates, views, integrity constraints,...) are **automatically adapted** to fit the new schema
# Prism: from 30,000 feet above

<table>
<thead>
<tr>
<th>Desiderata</th>
<th>PRISM proposed solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Evolution Design</td>
<td>Schema Modification Operators (SMO) (a language to express changes to the schema)</td>
</tr>
<tr>
<td>Increase Predictability of</td>
<td>SMO static analysis: foresee impact on schema, data and queries</td>
</tr>
<tr>
<td>evolution</td>
<td></td>
</tr>
<tr>
<td>Automate data migration</td>
<td>Generate SQL scripts from SMO sequences</td>
</tr>
<tr>
<td>Automate query support</td>
<td>Derive from SMOs logical mapping between schema versions</td>
</tr>
<tr>
<td></td>
<td>Automatic query rewriting (more recently work integrity constraints propagation and updates rewriting)</td>
</tr>
</tbody>
</table>
Schema Modification Operators

- **SMOs:**
  - Atomic changes
  - SQL-inspired syntax
  - Operates on both schema and data
  - Practical completeness (coverage of available evolution scenarios)

<table>
<thead>
<tr>
<th>SMO Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE a(a₁,a₂,a₃)</td>
</tr>
<tr>
<td>DROP TABLE a</td>
</tr>
<tr>
<td>RENAME TABLE a INTO b</td>
</tr>
<tr>
<td>COPY TABLE a INTO b</td>
</tr>
<tr>
<td>MERGE TABLE a, b INTO c</td>
</tr>
<tr>
<td>PARTITION TABLE a INTO b WITH cond₁, c</td>
</tr>
<tr>
<td>JOIN TABLE a, b INTO c WHERE cond</td>
</tr>
<tr>
<td>DECOMPOSE TABLE a INTO b(a₁,a₂), c(a₁, a₃)</td>
</tr>
<tr>
<td>ADD COLUMN a₄ AS [f(...)</td>
</tr>
<tr>
<td>DROP COLUMN a₄ FROM a</td>
</tr>
<tr>
<td>RENAME COLUMN a₃ IN a TO a₄</td>
</tr>
<tr>
<td>NOP</td>
</tr>
</tbody>
</table>
PRISM in a nutshell

- **DB Admin:**
  - expresses changes to the schema in terms of SMOs

- **PRISM:**
  - generates SQL scripts for migration
  - inverts SMO and generates Logical Schema Mapping
  - automatically rewrites queries (and updates) across schema versions
This is Prism, a tool supporting graceful schema evolution. The system provides a simple interface to support the definition of a set of schema changes, expressed in terms of (Schema and Integrity Constraint) Modification Operators. (see the project homepage for more details: http://schemaevolution.org)

1) Configuration
In the first phase, the system will ask configuration parameters to connect to the MySQL database, (leave default unchanged to test on a Demo database based on a version of the MediaWiki DB schema history).

2) Evolution-Design
The second phase, you can specify a series of atomic changes by issuing (Schema and Integrity Constraint) Modification Operators. The system will verify syntax for you and give you feedback about information preservation and redundancy. The “Show Resulting Schema” button will provide at any moment the snapshot of the resulting schema, highlighting the changes.

3) Inverse-Design
In this phase, the system tries to generate automatically an inverse sequence of changes, this is needed to allow query rewriting, and thus Legacy application support. The user is allowed to override the default system choices at any time. The user intervention is needed to disambiguate cases in which the inverse is not unique. The system tries to rewrite for a set of queries that have been logged during the system usage, and provides feedback on which percentage of queries is supported automatically, based on the chosen inverses.

4) Validation
In this phase, the system will test more queries and inspect rewritten queries and the logical mappings Disjunctive Embedded Dependencies (DEDs) derived from the schema evolution designed. This is an assessment phase, the user should verify to be satisfied with the results of this evolution.

5) Deployment
Experimental Results

- Based on the actual Wikipedia schema evolution [iceis2008]:
  - schemas (170+), queries (from on-line profiler) and data (released)
  - Completely automate 97.2% of evolution steps (78% automation in the rest of the cases)
  - High quality resulting queries (very close to manual rewriting performance)
Database Archival under Schema Evolution

Why we need Transaction Time DBs:
accountability obligations, flashback, auditing, fault recovery, legal compliance, backtracking in scientific workflows, trend analysis, information preservation/historical value, data provenance/lineage,...

Q: “What was the average salary in dept. 3 from 2003 to 2005?”

The Problem:
- “Archive and efficiently support temporal queries over the history of the content of a DB subject to schema evolution”

The Challenges:
1. Achieve Perfect Archival Quality (no information lost)
2. Make querying easy (complex temporal queries + multiple schemas)
3. Achieve Reasonable Performance (complex and expensive queries + huge data sizes)
## Prima: from 30,000 feet above

<table>
<thead>
<tr>
<th>Desiderata</th>
<th>PRIMA proposed solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect Archival Quality</td>
<td><strong>Original Schema Archiving</strong>: data are stored under the original schema</td>
</tr>
<tr>
<td>Intuitive Query Interface</td>
<td><strong>Support for schema evolution</strong>: automatic rewriting of temporal queries</td>
</tr>
<tr>
<td></td>
<td><strong>Decouple logic and physical layer</strong>: Expressive query interfaces (SQL 200n or Xquery), Optimized storage engine.</td>
</tr>
<tr>
<td>Performance</td>
<td><strong>Temporal Specific Query Optimizations</strong>: temporal join, temporal coalescing (novel algorithm) -&gt; <em>over 6 order of magnitude of performance improvement!</em></td>
</tr>
</tbody>
</table>
Conclusion

• **PRISM:**
  • Language of Schema Modification Operators
  • Generation of Data Migration scripts
  • Derive logical mapping (dealing with composition and inversion)
  • Rewrite legacy queries (and updates)

• **PRIMA:**
  • Perfect archival under schema evolution
  • Intuitive temporal queries (SQL 200n + Xquery)
  • Query answering performance (architecture + temporal optimizations)

• **HMM (not discussed here):**
  • recording and querying the metadata history!

*More if you are interested at: http://schemaevolution.org/*

“Mars: A system for publishing XML from mixed and redundant storage” Alin Deutsch and V. Tannen, VLDB, 2003


PRIMA: Archiving and Querying Historical Data with Evolving Schemas” Hyun J. Moon, Carlo A. Curino, MyungWon Ham, Carlo Zaniolo, demo SIGMOD 2009

“The PRISM Workwrench: Database Schema Evolution Without Tears” Carlo A. Curino, Hyun J. Moon, MyungWon Ham, Carlo Zaniolo, accepted as demo at ICDE 2009

“Managing the history of metadata in support for db archiving and schema evolution”, Carlo A. Curino, Hyun J. Moon, and Carlo Zaniolo, ECDM, 2008


“Supporting integrity constraints and update rewriting in Schema Evolution”, Carlo A. Curino, Alin Deutsch, Hyun J. Moon, Carlo Zaniolo, currently under submission

Panta Rhei  
Carlo A. Curino
[sub09b] “Architecture and Optimization for Transaction-time DBs with Evolving Schemas”, Hyun J. Moon, Carlo A. Curino, Carlo Zaniolo, currently under submission

[mdm06] “Context integration for mobile data tailoring” Cristiana Bolchini, Carlo A. Curino, Fabio A. Schreiber, Letizia Tanca, Mobile Data Management (MDM) 2006


[otm08] “Improving search and navigation by combining Ontologies and Social Tags”, Silvia Bindelli, Claudio Criscione, Carlo A. Curino, Mauro L. Drago, Davide Eynard, Giorgio Orsi, OTM Workshop: Ambient Data Integration (ADI) 2008