Model Management 2.0: Manipulating Richer Mappings

Phil Bernstein & Sergey Melnik
Microsoft Research

Sept 30, 2007
Data Programmability

- Make it easier to write programs that access databases
- Traditionally, for large IT departments
- Much progress, but it’s still ~40% of the work
- Core problem is developing and using complex mappings between schemas
Mapping Problems are Pervasive
And it's a Growth Industry

- Data translation
- XML message mapping
- Data warehouse loading
- Query mediators
- Forms managers
- Report writers
- Query designers
- Object-relational wrappers
- Portal generation from DB
- OLAP databases
- Application integration
- Composing web services
Most packaged business apps need to access an OO view of relational data

Requires an OR mapping
An Example Mapping

- Person = HR \cup \pi_{\text{Id,Name}}(\text{Client})
- Employee = HR \bowtie \text{Empl}
- Customers = \text{Client}

Target: EER

Source: SQL
SELECT VALUE

CASE
    WHEN (T5._from2 AND NOT(T5._from1)) THEN Person(T5.Person_Id, T5.Person_Name)
    WHEN (T5._from1 AND T5._from2) THEN Employee(T5.Person_Id, T5.Person_Name, T5.Employee_Dept)
END
FROM (   (SELECT T1.Person_Id, T1.Person_Name, T2.Employee_Dept,
                CAST(NULL AS SqlServer.int) AS Customer_CreditScore,
                CAST(NULL AS SqlServer.nvarchar) AS Customer_BillingAddr,
                False AS _from0,
                (T2._from1 AND T2._from1 IS NOT NULL) AS _from1, T1._from2
        FROM HR AS T)
    AS T1
    LEFT OUTER JOIN (SELECT T.Id AS Person_Id, T.Name AS Person_Name, True AS _from2
        FROM dbo.Empl AS T)
    AS T2
    ON T1.Person_Id = T2.Person_Id )
UNION ALL (SELECT T.Id AS Person_Id, T.Name AS Person_Name,
                CAST(NULL AS SqlServer.nvarchar) AS Employee_Dept,
                T.Score AS Customer_CreditScore, T.Addr AS Customer_BillingAddr,
                True AS _from0, False AS _from1, False AS _from2
        FROM Client AS T)

[ Melnik, Adya, Bernstein, SIGMOD 07]
Why is mapping hard?

- Heterogeneity
- Impedance mismatch
- Insufficient abstraction
- Potpourri of tools

[Haas, ICDT 07]
And It's Getting Harder

- More data models
  - Java, ODMG, XSD, .NET
  - RDF, OWL, EDM, SML
- More programming languages
- More types of tools
- More schema sources
  - Web site wrappers
  - Google Base
  - Generic info extractors [Gubanov, Bernstein WebDB 06]
Mapping Space

Info Integration Workshop
http://db.cis.upenn.edu/iiworkshop/

Precise

Approx

Process

Fully Automatic

Time Boxed

Incrementally Developed

Carefully Engineered

Precision

Formatted

Text

Type

GIS
Model Management 1.0

Manipulate models & mappings as bulk objects

Meta-model independent
- relational, ER, OO, XML, RDF, OWL, SML, ...

Operations
- Match
- Compose
- Merge
- Diff/Extract
- ModelGen
- Inverse

Tools
- Wrapper Generator
- Query Mediator
- ETL

Model Management Engine

Metadata Repository

[ Bernstein, Halevy, Pottinger
SIGMOD Record 00]
More research focus on primary operations
- More powerful operations
- Hence better tools

More leverage from tool investments

More uniform behavior across tools
Good News

- Lots of progress on operations
- Some practical applications
- A lot has been learned

Bad News

- Still waiting for the first reasonably-complete practical implementation

Good news

- A lot of research left to do
Outline

- What has changed: Use richer mappings
- What has changed: Include the runtime
- Model Management 2.0 in detail
What Has Changed?

Use Richer Mappings

2000
Structural mappings
- Mappings are structural
  - Relate schemas, *not* data
- Operations oblivious to mapping semantics
- Semantics is a plug-in

2007
Semantic mappings
- Mappings are semantic
  - Relate schemas *and* data
- Operations sensitive to mapping semantics
- Semantics is built-in
I(S_1) are the instances of schema S_1
- Each d in I(S_1) is a database (e.g., a set of relations)

I(S_2) are the instances of schema S_2

map_{12} \subseteq I(S_1) \times I(S_2)

Usually, we represent a mapping by an expression

V= R \bowtie S

R \bowtie S = T \bowtie U
Example
In 2000, mapping is a structure

Plug in semantic expression

\[ \pi_{EID}(Empl) = \pi_{SID}(Staff) \land \]
\[ \pi_{EID,Name}(Empl) = \pi_{SID,Name}(Staff) \land \]
\[ \pi_{EID,AID,Concat(Street,City,Zip)}(Empl \bowtie Addr) = \pi_{SID,Address}(Staff) \]
Example
In 2007, just use the expression

\[ \pi_{EID}(Empl) = \pi_{SID}(Staff) \land \]
\[ \pi_{EID,Name}(Empl) = \pi_{SID,Name}(Staff) \land \]
\[ \pi_{EID,AID,Concat(Street,City,Zip)}(Empl \bowtie Addr) = \]
\[ \pi_{SID,Address}(Staff) \]
Element correspondences
- First step in aligning schemas
- For lineage & impact analysis
- Weak or no formal semantics

Mapping constraints relate instances of schemas
- E.g., equality of relational expressions

```
SELECT Id, Name, Dept = SELECT Id, Name, Dept
FROM Employee
FROM HR JOIN Empl ON Id
```

Transformation is an executable mapping constraint
- Constructs target instances from source instances
- E.g., SQL query, XSLT, C# program

[Casanova, Vidal. PODS 83]
[Biskup, Convent. SIGMOD 86]
[Catarci, Lenzerini. J. CoopIS 93]
[Miller, Haas, Hernandez. VLDB 00]
**Mapping Expressiveness**

What we want: first-order logic with
- aggregation
- set and bag semantics
- regular expressions
- nested collections and lists
- rich type constructors (e.g., to construct XML fragments),
- user-defined functions
- deduplication and other heuristic functions

What can we handle? ... Much less.
Parallel Evolution

Clio Project
• IBM, Univ. of Toronto, U.C. Santa Cruz
• Miller, Haas, Hernandez, Fagin, Ho, Popa, Tan, ...

Build a design tool for semantic mappings

Model Management
• Microsoft, Univ. of Washington, Univ. of Leipzig
• Bernstein, Halevy, Pottinger, Rahm, Madhavan, Melnik, ...

Build model management operations with plug-in semantics

Study model management operations with semantics
What Has Changed?

Include the Runtime

2000

Design-time

- Model management is independent of run-time
- No special run-time functionality

2007

Run-time

- Model management is tied to a run-time
- Run-time functions are sensitive to mapping expressiveness and model mgt capabilities
Mapping Runtime

**Target**
- Actions & Constraints

**Interpret or Compile**
- Source
- Actions & Constraints

**Source**
- Interpret or Compile
- Actions & Constraints

- Queries
- Updates
- Peer-to-peer
- Provenance
- Access Control

- Integrity constraints
- Synch logic
- Business logic
- Debugging

- Errors
- Indexing
- Notifications
- Batch loading
- Data exchange
Integrity constraints on target $T$ are enforced by a combination of constraints enforced by the source and by the target runtime.

Feasibility - some constraints on $T$ may not be expressible in source $S$. 

$$(\text{Class}_1 \text{ as Class}_0) \cap (\text{Class}_2 \text{ as Class}_0) = \emptyset$$
Provenance

- User moves data from source $S$ to target $T$
- Which source data contributed to a particular target data item?

Errors

- A data access via $T$ is translated into an access on $S$ that generates an error
- The error needs to be passed back through the mapping in a form that is understandable in the context of $T$. 

[Cui, Widom, Weiner. TODS 00]
[Baghwat, Chiticariu, Tan, Vijayvargia. VLDB J. 05]
[Buneman, Chapman, Cheney. SIGMOD 06]
Scenarios

1. Create mappings
   - Match
   - ConstraintGen
   - TransGen
   - ModelGen

2. Evolve mappings
   - Compose
   - Diff
   - Merge
   - Inverse
Exploit lexical analysis of element names, schema structure, data types, thesauri, value distributions, ontologies, instances, and previous matches.

Past Goal - improved precision & recall
- Big productivity gains are unlikely

Better goals
- Return top-k, not best overall match
- Avoid the tedium. Manage work.
- HCI – handle large schemas.
- User studies – what would improve productivity?
Cast of Thousands

- AnHai Doan
- Alon Halevy
- Pedro Domingos
- Phil Bernstein
- Erhard Rahm
- Sergey Melnik
- Jayant Madhavan
- Jeffrey Naughton
- Jaewoo Kang
- Tova Milo
- Pavel Shvaiko
- Fausto Giunchiglia
- Sonia Bergamaschi
- Silvana Castano
- Bin He
- Kevin Chang
- Namyoun Choi
- Il-Yeol Song
- Hyoil Han
- Domenico Ursino
- Luigi Palopoli
- Dominico Sacca
- Georgio Terracina
- David Embley
- David Jackman
- Li Xu
- Yihong Ding
- Jacob Berlin
- Amihai Motro
- Hong Hai Do
- Fabien Duchateau
- Zohra Mellahsene
- Ela Hunt
- Toralf Kirsten
- Andreas Thor
- Alexander Bilke
- Avigdor Gal
- Michalis Petropoulos
- Christoph Quix
- Chris Clifton
- Arnie Rosenthal
- Wen-Syan Li
- Hector Garcia-Molina
- Sagit Zohar
- Gio Wiederhold
- Anna Zhdanova
- Jerome Euzenat
- Prasenjit Mitra
- Natasha Noy
- Anuj Jaiswal
- Mikalai Yatskevich
- Nuno Silva
- Joao Rocha
- David Aumueller
- Sabine Massmann
- Felix Naumann
Code Generation Scenarios [Miller, Haas, & Hernandez, VLDB 00]

Models

Correspondences

Match

Customer
Order
Scheduled
Delivery
Product
Salesperson

Foo (a,b,c)
Bar (d,e,f)

Match

Correspondences

ConstraintGen

Select ord#, prod#, cust# From Shipped
= Select ord#, prod#, cust# From Order Join Item on ord#

Constraints

Constraints

TransGen

Transformation

Select ord#, prod#, cust# From Shipped
≤
Select ord#, prod#, cust# From Order Join Item on ord#
For a given target element

- Find all source elements linked by correspondences
- Find all ways that source elements are related
- Choose one of them and generate the transformation

\[
\pi_{\text{SID}, \text{Name}, \text{City}}(\text{Staff}) = \pi_{\text{EID}, \text{Name}, \text{City}}(\text{Empl} \bowtie \text{Addr})
\]
Directly interpret correspondences as mapping constraints

If it’s a tree schema and keys correspond

1. $\pi_{EID}(Empl) = \pi_{SID}(Staff)$
2. $\pi_{EID,Name}(Empl) = \pi_{SID,Name}(Staff)$
3. $\pi_{EID,City}(Empl \bowtie Addr) = \pi_{SID,City}(Staff)$
SELECT p.Id, p.Name
FROM Persons AS p
WHERE p IS OF (ONLY Person)
  OR p IS OF (ONLY Employee)

SELECT e.Id, e.Dept
FROM Persons AS e
WHERE e IS OF Employee

SELECT c.Id, c.Name,
    c.CreditScore, c.BillingAddr
FROM Persons AS c
WHERE c IS OF Customer
A Relationship Constraint

**Target:** EER

- **Person**
  - Id
  - Name
- **Employee**
  - Dept
- **Supports**
  - 0..1
- **Customer**
  - CreditScore
  - BillingAddr

**Source:** SQL

- **HR**
  - Id
  - Name
- **Empl**
  - Id
  - Dept
- **Client**
  - Id
  - Eid
  - Name
  - Score
  - ...

```
SELECT Key(s.Customer).Id, Key(s.Employee).Id
FROM Supports s
```

```
SELECT Cid, Eid
FROM Client
WHERE Eid IS NOT NULL
```
```
SELECT p.Id, p.Name
FROM Persons AS p
WHERE p IS OF (ONLY Person) OR p IS OF (ONLY Employee)

SELECT e.Id, e.Dept
FROM Persons AS e
WHERE e IS OF Employee

SELECT c.Id, c.Name, c.CreditScore, c.BillingAddr
FROM Persons AS c
WHERE c IS OF Customer
```

```
SELECT Id, Name
FROM dbo.HR

SELECT Id, Dept
FROM dbo.Empl

SELECT Id, Name, Score, Addr
FROM dbo.Client

SELECT VALUE CASE
WHEN (T5._from2 AND NOT(T5._from1)) THEN Person(T5.Person_Id, T5.Person_Name)
WHEN (T5._from1 AND T5._from2) THEN Employee(T5.Person_Id, T5.Person_Name, T5.Employee_Dept)
END
FROM (SELECT T1.Person_Id, T1.Person_Name, T2.Employee_Dept, CAST(NULL AS SqlServer.int) AS Customer_CreditScore,
       CAST(NULL AS SqlServer.nvarchar) AS Customer_BillingAddr, False AS _from0,
       (T2._from1 AND T2._from1 IS NOT NULL) AS _from1, T1._from2
       FROM HR AS T1
       LEFT OUTER JOIN (SELECT T.Id AS Person_Id, T.Name AS Person_Name, True AS _from2
                       FROM dbo.Empl AS T)
       ON T1.Person_Id = T2.Person_Id )
UNION ALL
(SELECT T.Id AS Person_Id, T.Name AS Person_Name, CAST(NULL AS SqlServer.nvarchar) AS Employee_Dept,
       T.Score AS Customer_CreditScore, T.Addr AS Customer_BillingAddr,
       True AS _from0, False AS _from1, False AS _from2
       FROM Client AS T)
```
Difficulty depends on:

- Whether the constraints are functions
- The transformation language (e.g., SQL, XSLT)
- Expressiveness of constraints
- Optimization required
**Declarative mapping language**
- Allows non-expert users to specify complex mappings
- Formal semantics

**Bidirectional views**
- Uniform, efficient runtime
- Simplifies dev & test

**Updates via view maintenance**
- Arbitrary updates
- Uses view maintenance technology

---

Select ord#, prod#, cust# From Shipped

Select ord#, prod#, cust# From Order Join Item on ord#
Compiling Constraints

Mapping: \{Q_{C1} = Q_{S1}, \ldots, Q_{Cn} = Q_{Sn}\}

E.g., \( f: \)

\[
\begin{align*}
&\text{SELECT p.Id, p.Name} \\
&\text{FROM Persons p}
\end{align*}
\]

= \( g: \)

\[
\begin{align*}
&\text{SELECT Id, Name} \\
&\text{FROM ClientInfo}
\end{align*}
\]

\( f: V_1 = Q_{C1} \cup \ldots \cup V_n = Q_{Cn} \)

\( g: V_1 = Q_{S1} \cup \ldots \cup V_n = Q_{Sn} \)

query view
Composition

$I(S_1)$ are the instances of schema $S_1$

$\text{map}_{12} \subseteq I(S_1) \times I(S_2)$ \quad \text{map}_{13} \subseteq I(S_2) \times I(S_3)$

$\text{map}_{13} = \{ <d_1 \in I(S_1), d_3 \in I(S_3)> |$

$\exists d_2 \in I(S_2) ( <d_1, d_2> \in \text{map}_{12} )$

$\land ( <d_2, d_3> \in \text{map}_{23} ) \}$

Well known examples

- View unfolding $S_1 \xrightarrow{v} S_2 \xrightarrow{q} S_3$
- Answering queries using views $\overleftarrow{S_1} \xrightarrow{v} S_2 \xrightarrow{q} S_3$
Use query rewriting to compute $f'$ from $f$.
This is mapping composition.
1. Create mappings
   - Match
   - ConstraintGen
   - TransGen
   - ModelGen

2. Evolve mappings
   - Compose
   - Diff
   - Merge
   - Inverse
There are several credible prototypes
Don't know of products, yet

[Atzeni, Torlone. EDBT 96]
[Bernstein, Melnik, Mork. VLDB 05]
[Atzeni, Cappellari, Bernstein. EDBT 06]
Data is transferred to super-metamodelling DB

Data is transformed within super-metamodelling DB

Data is transferred to output schema’s database
Obtaining Mappings From ModelGen

[Bernstein, Melnik, Mork VLDB’05, ER’07]

- Super meta-model
- Native meta-model

\( S_0 \) \( \xrightleftharpoons{\text{map}_1} \) \( S_1 \) \( \cdots \) \( \xrightleftharpoons{\text{map}_n} \) \( S_n \)

Eliminate \( m:n \) associations
Eliminate inheritance

Input OO schema \( \xleftarrow{\text{map}_1 \circ \text{map}_2 \circ \ldots \circ \text{map}_n} \) Output SQL schema

Leverages Compose operator
Each \( \text{map}_i \) roundtrips data
1. Create mappings
- Match
- ConstraintGen
- TransGen
- ModelGen

2. Evolve mappings
- Compose
- Diff
- Merge
- Inverse
Schema Evolution

[ Rahm, Bernstein. SIGMOD Rec. Dec 06]

- Schema evolves
- What about database & view?
1. Create mapping: \textit{schema} \Leftrightarrow \textit{evolved schema}

2. Generate a transformation
Compose $Map_{SV}$ and $Map_{ES}$ to connect view to evolved schema.
Some natural 1st-order mapping languages are not closed under composition
- Sometimes, it's undecidable whether the composition is expressible in the input language
- Can settle for a partial solution over 1st-order mappings

Or you can use a 2nd-order mapping language that's closed under composition
- There's a composition algorithm to compute it

Some prototype implementations reported
- Practical applications needed

[Fagin, Kolaitis, Popa, Tan. TODS 05]
[Nash, Bernstein, Melnik. TODS 07]
[Yu, Popa. VLDB 05]
[Bernstein, Green, Melnik, Nash. VLDB 06]
Augment View with $S_E'$'s new data

$V' = \text{Merge}(V, S_E', p\cdot q\cdot r)$

$p = \text{Map}_{SV}$

$q = \text{Map}_{ES}$

$r = \text{Map}_{S_E S_E'}$

$S_E' = \text{Diff}(S_E, \text{Map}_{ES})$
[\(S'', \text{map}_{S''-S'}\] = Extract\((S', \text{map}_{S'-S})\)

- \(S''\) is a maximal sub-schema of \(S'\) that can be populated with data from \(S\) via \(\text{map}_{S'-S}\)
- Related to the materialized view selection problem: \(S''\) is the minimal view needed to populate \(S\)

\(\text{Diff}(S', \text{map}_{S'-S})\) is the complement of Extract

- It's the view complement problem [Bancillon &Spyratos, TODS 81]
- An algorithm for select-project-join-union views is in [Lechtenbörger, Vossen. TODS 03]
Take disjoint union of schemas and constraints and then optimize

Merge algorithms for structural mappings

Extension: input map is a first-class model

Nothing known for semantic mappings

[Barroso, Gallo. PODS 00]
[Spaccapietra, Parent. TKDE 94]
[Biskup, Convent. SIGMOD 86]
[Pottinger, Bernstein. VLDB 03]
[Buneman, Davidson, Kosky. EDBT 92]
More surveys
- Solutions to data programability problems
- Products that address these problems (e.g. runtimes)

More case studies
- Using published solutions and products to solve mapping problems
Other Challenges

- Semantics and algorithms of operators with more expressive mappings
- Translating behavior on target via mapping to behavior on source
Is it still a goal to build a MMS?

Or is it just a set of techniques to be applied?
Summary

There’s a big market looking for solutions

Limited known about run-time scenarios
  • Mostly just for queries
  • Some updates, provenance, integrity constraints
  • Much work needed for synch logic, errors, indexing, notifications, batch loading, ....

There’s progress on many operators
  • But it's incomplete
  • For mappings with limited expressiveness
  • Little known about merge, diff, extract
References

- P. Bernstein, S. Melnik, “Model Management 2.0: Manipulating Richer Mappings,” *SIGMOD 2007*

- S. Melnik, A. Adya, P. Bernstein, “Compiling Mappings to Bridge Applications and Databases,” *SIGMOD 2007*