FedDW Global Schema Architect

UML-based Design Tool for the Integration of Data Mart Schemas

Dr. Stefan Berger

Department of Business Informatics – Data & Knowledge Engineering
Johannes Kepler University Linz

ACM 15th DOLAP ’12 — November 2, 2012
Outline

1. FedDW Approach
2. Tool Support: FedDW Tool Suite
FedDW Approach

1. General overview of FedDW
2. Integrating heterogeneous multidimensional schemata

Tool Support: FedDW Tool Suite
Problem definition; our contribution

**Problem**: similar autonomous data marts/DWs, but heterogeneous schemata and/or data

- Business collaboration
- Mergers and acquisitions

⇒ Preexisting DW data across autonomous organizations

**Contribution**: comprehensive tool suite for integration of autonomous data marts/DWs

- Visual integration of multidimensional schemas
- OLAP front-end prototype, based on SQL-MDi
  [Berger and Schrefl, 2006]
**Problem**: similar autonomous data marts/DWs, but heterogeneous schemata and/or data

- Business collaboration
- Mergers and acquisitions

⇒ Preexisting DW data across autonomous organizations

**Contribution**: comprehensive tool suite for integration of autonomous data marts/DWs

- Visual integration of multidimensional schemas
- OLAP front-end prototype, based on SQL-MDi [Berger and Schrefl, 2006]
General overview of FedDW

Motivating example

Telecommunications sector—sample, heterogeneous conceptual data mart schemas:

- Dimensionality (extra dimension `blue.promotion`)
- Hierarchy of `date` dimensions
- Decorations of `product` dimensions
- Measures of `connections` facts
Motivating example

Telecommunications sector—sample, heterogeneous conceptual data mart schemas:

- Dimensionality (extra dimension `blue.promotion`)
- Hierarchy of `date` dimensions
- Decorations of `product` dimensions
- Measures of `connections` facts
Integrating heterogeneous multidimensional schemata

Conflict classification I

- Dimension Instance ("Members") Conflicts
- Cube Instance ("Cells") Conflicts
- Schema-Instance Conflicts
- Dimension Schema Conflicts
- Cube Schema Conflicts

Model Entity

Modeling Scope

Instance

Scope

Dimension Cube

Stefan Berger (Univ. Linz)
## Conflict classification II

<table>
<thead>
<tr>
<th>Facts: conflicts</th>
<th>Relevant operator of FedDW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema-instance</td>
<td>Merge measures: PIVOT MEASURES (Fact)</td>
</tr>
<tr>
<td></td>
<td>Split measures: PIVOT SPLIT MEASURES (Fact)</td>
</tr>
<tr>
<td>Dimensionality</td>
<td>Choose attributes: add DIM reference (Cube)</td>
</tr>
<tr>
<td>Different measures</td>
<td>Choose measures: add MEASURE reference (Cube)</td>
</tr>
<tr>
<td>Domain (measures)</td>
<td>Convert domain: CONVERT MEASURES APPLY ... (Measure)</td>
</tr>
<tr>
<td>Naming of attributes</td>
<td>Rename attributes: operator “→ ...” (Measure, Dimension)</td>
</tr>
<tr>
<td>Base levels</td>
<td>Roll-up dimension attributes: ROLLUP TO LEVEL ... (Dimension)</td>
</tr>
<tr>
<td>Cube cells (fact extensions)</td>
<td>Join cubes: MERGE CUBES (<em>n-ary</em>)</td>
</tr>
<tr>
<td></td>
<td>Derive measure values: AGGREGATE MEASURE (<em>n-ary</em>)</td>
</tr>
</tbody>
</table>
## Conflict classification III

<table>
<thead>
<tr>
<th>Dimensions: conflicts</th>
<th>Relevant operator of FedDW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchies</td>
<td>Map corresponding levels: add level reference [ . . . ] (Dimension)</td>
</tr>
<tr>
<td>Domain (levels / decorations)</td>
<td>Convert domain: CONVERT ATTRIBUTES APPLY ... (Dimension)</td>
</tr>
<tr>
<td>Naming (levels)</td>
<td>Rename attributes: operator “→ ...” (Level)</td>
</tr>
<tr>
<td>Naming (decorations)</td>
<td>Map decorations: MATCH ATTRIBUTES (under Merge Dimensions—(n)-ary)</td>
</tr>
<tr>
<td>Members (dim. extensions)</td>
<td>Merge sets of members: MERGE DIMENSIONS ((n)-ary)</td>
</tr>
<tr>
<td>Roll-up functions</td>
<td>Overwrite hierarchies: RELATE Expression (under Merge Dimensions clause—(n)-ary)</td>
</tr>
<tr>
<td>Decoration values</td>
<td>Correct values: add RENAME function (under Merge Dimensions clause—(n)-ary)</td>
</tr>
</tbody>
</table>
Establish a *federation* of autonomous data marts:

1. Import data mart schemas (CWM supported)
   - (Optional: enrich roll-up hierarchies
     ⇒ *minimum match* integration strategy)
2. Design global multidimensional schema (canonical model)
3. Define semantic mappings – *both-as-view* paradigm [see McBrien and Poulavassilis, 2003]
   - (a) Resolve schema–instance conflicts
   - (b) Intensional integration – map conceptual schemata
     - Fact tables
     - Dimension tables + hierarchies
   - (c) Extensional integration – consolidate data
FedDW Approach

Tool Support: FedDW Tool Suite

FedDW Global Schema Architect
FedDW Query Tool
Overview of FedDW tool support I

Java- and Eclipse-based interactive tool suite (EMF, GMF, UML2)

- OLAP front-end prototype: FedDW Query Tool [Berger and Schrefl, 2009]
- Auxiliary components: Metadata Dictionary, Dimension Repository
Overview of FedDW tool support II

FedDW Global Schema Architect

Federated DW System

Global schema

Mappings (SQL-MDi)

Import Schemas

Meta-data dictionary

Dimension repository

Query Tool

SQL-MDi Parser

SQL-MDi Processor

DM 1

DM 2

...DM n

User Query (SQL)

OLAP Application

Stefan Berger (Univ. Linz)
FedDW Global Schema Architect

DOLAP – Nov. 2, 2012
Overview of FedDW GSA

Visual design environment for multidimensional schemas

- **Schema Editor** — nested UML diagrams
  - *Import schemas*
  - *Global schema*

- **Mapping Editor** — graphical, high-level code editor *(Master–Detail layout)*
  - *Import mappings*: unary operators (Fact, Dimension entities) — intensional
  - *Global mappings*: n-ary operators — extensional
Sample GSA Workflow

1. Import local, autonomous connections schemas
2. Design global connections schema
3. Create import mappings
4. Create one global mapping file
5. Export the mappings to metadata repository
6. Export fact and dimension metadata
FedDW Global Schema Architect

GSA: Step 1, Import Wizard I

- Enter username: blue_dw
- Password: **********
- Enter ip_Address: photos.dw.uni-linz.ac.at:1521
- Enter SID: dw4
- Enter Schema Name: blue_dw

New File Name: blue_dw.import.uml

- Enter username: red2_admin
- Password: **********
- Enter ip_Address: localhost
- Enter servername: DGCQL1
- Instance: red2

New File Name: red2_dw.import.uml
### GSA: Step 1, Import Wizard II

Wizard suggests appropriate UML stereotypes (based on PK/FK constraints):

<table>
<thead>
<tr>
<th>FACT</th>
<th>DIMENSION</th>
<th>NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERS</td>
<td>CUSTOMERS</td>
<td>CUSTOMERS</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>CATEGORY</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>PRODUCTS</td>
<td>PRODUCTS</td>
<td>PRODUCTS</td>
</tr>
<tr>
<td>DATES</td>
<td>DATES</td>
<td>DATES</td>
</tr>
<tr>
<td>PROMOTIONS</td>
<td>PROMOTIONS</td>
<td>PROMOTIONS</td>
</tr>
<tr>
<td>CONNECTIONS</td>
<td>CONNECTIONS</td>
<td>CONNECTIONS</td>
</tr>
</tbody>
</table>
GSA: Step 1, Import Wizard III

Initialized class diagram of `red.connections`:
Global Schema wizard:

- Comfortably create global schema as copy of one import schema
- Edit the schema later on
GSA Schema Editors: edit dimension

User-friendly editing of UML diagram possible (context menus, UML palette):
Recall the heterogeneities among red and global:

**red**
- customer
- prod_name
- prod_name
- regular_fee
- duration
- tn_tel
- tn_misc

**global**
- category
- customer
- p_name
- contract_type
- base_fee
- connections
- dur_min
- turnover
- customers
- contract_type
- base_fee
- date
- month
- year
- dates

**Abbildung 4: Necessary changes in conceptual schema of Data Mart “Red” (DOLAP 2012)**

Stefan Berger (Univ. Linz)
Repair `red.connections` import schema:

- Dimensionality: *add references* to all three dimensions
- Date hierarchy: *roll-up to LEVEL [date]*
- Product decorations: *delete* `regular_fee` from Red’s import schema
- Measures (schema–instance conflict): *PIVOT MEASURES* `tn_tel, tn_misc`
Import Mappings: dimensionality
Import Mappings: hierarchy

Prerequisite: delete level [date/hr] from red.date
(see Schema Editor, slide 21)
“Merge” measures `tn_tel, tn_misc` into turnover, extracting values “tn_tel”, “tn_misc” as members of the new `red.category` dimension:
Merge Dimensions:

```
MERGE DIMENSIONS USING c1.customers AS c1, c2.customers AS c2 INTO c1.customers AS custom
(REPLACE c1.customer_id, c2.customer_id WHERE c1.customer_id != c2.customer_id USING HIERARCHY OF 1)
(MATCH (attributes r.name IS bs1, bs1.name))
```

- Set operations
- Add dimension reference
- INTO (GLOBAL DIMENSION)
- Rename dimension
- Relate expression
- Rename content
- Prefering
- Match attributes
- Convert attributes apply
GSA Step 4, Global Mapping II

Merge Cubes:

```
MERGE CUBES c1, c2 INTO c0 ON date, customer, product, category
AGGREGATE MEASURE duration IS SUM OF duration,
AGGREGATE MEASURE turnover IS SUM OF turnover,
(AGGREGATE MEASURE *duration, AGGREGATE MEASURE *turnover, DIM category, DIM customer, DIM date, DIM pr
```

- Add Cube Reference
- Cum
- Date
- Customer
- Product
- Category

- Prefer Dimension
- Aggregate Measure
- Tracking Source As Dimension
GSA Step 5–6, export project metadata

Step 5: Export mapping file — export wizard
- Starts generation of SQL-MDi code
- Static syntax check
- Interface to FedDW Query Tool: file system

Step 6: populate Metadata Dictionary
- Facts + dimensions conceptual and physical metadata
- Later accessed by FedDW Query Tool
GSA Step 5–6, export project metadata

Step 5: Export mapping file — export wizard
- Starts generation of SQL-MDi code
- Static syntax check
- Interface to FedDW Query Tool: file system

Step 6: populate Metadata Dictionary
- Facts + dimensions conceptual and physical metadata
- Later accessed by FedDW Query Tool
Intelligent features:

- Import heuristics: analyzes PK/FK constraints in import schemas to suggest adequate UML stereotypes
- Create global schema as copy of one import schema
- User-friendly and intuitive UML notation
- Visual conversion modeling avoids “cheap” SQL-MDi syntax errors
- Automatically populates Dimension Repository from the exported metadata
- Supports the CWM standard [Poole, 2003]
FedDW Query Tool

Query Tool in a Nutshell

```sql
MERGE DIMENSIONS c1.date_hr AS d1, c2.date AS d2 INTO c3.date AS d3
MERGE DIMENSIONS c1.customer AS d3, c2.customer AS d4 INTO c4.customer AS d5
USING HIERARCHY OF c3
(WATCH ATTRIBUTES d3.name AS d4.customer_name)
(CONVERT ATTRIBUTES APPLY curr2Euro() FOR d4.base_fee DEFAULT)
MERGE DIMENSIONS c1.product AS d6, c2.product AS d7 INTO c0.product AS d8
(RENAME d8.product AS 'HandyTelCo' WHERE c1.product = 'HandyTelCo')
MERGE DIMENSIONS c1.category AS d9, c2.category AS d10 INTO c0.category AS d11
MERGE CUBES c1, c2 INTO c0 ON date, customer, product, category
#AGGREGATE MEASURE duration AS SUM OF duration, AGGREGATE MEASURE turnover AS SUM OF turnover
(MEASURE duration, MEASURE turnover, DIM date, DIM customer, DIM product, DIM category)

SELECT d.month AS Month, p.product AS Product, ROUND(SUM(s.turnover), 2) AS Turnover
FROM date d, products p, sales s
WHERE s.date = d.date AND p.product = s.product
GROUP BY d.month, p.product
ORDER BY Turnover DESC;
```

Evaluating OLAP query...
Finished evaluating OLAP query!
Done!
Thanks for your attention!

