Methodological framework

Logical Modeling: The Multidimensional Model
Logical Design: From Fact schema to ROLAP Logical schema
ROLAP schema in MDX for Mondrian

Bibliographie

- Livres

- Cours
  - Course of M. Golfarelli M. and S. Rizzi, University of Bologna
  - Course of M. Böhlen and J. Gamper J., Free University of Bolzano
  - …

Outline

1. Methodological Framework
   - Conceptual Design & Logical Design
   - Design Phases and schemata derivations

2. Logical Modelling: The Multidimensionnal Model
   - Problematic of the Logical Design
   - The Multidimensional Model: fact, measures, dimensions

3. Implementing a Dimensional Model in ROLAP
   - Star schema
   - Snowflake schema
   - Aggregates and views

4. Logical Design: From Fact schema to ROLAP Logical schema
   - From fact schema to relational star-schema: basic rules
   - Examples towards Relational Star Schema
   - Examples towards Relational Snowflake Schema
   - Advanced logical modelling

5. ROLAP schema in MDX for Mondrian

1. Methodological framework
   - Conceptual Design & Logical Design
   - Life-Cycle
   - Design Phases
   - Schemata derivations
**Conceptual Design & Logical Design**

- **Entity-Relation models** are not very useful in modeling DWs
- Is now universally recognized that a DW is based on a **multidimensional view of data**:
  - But there is still no agreement on **HOW** to implement its conceptual design!
- **Most of the time, DW design is at the logical level**: a multidimensional model (star/snowflake schema) is directly designed:
  - But a *star schema* is nothing but a **relational schema**: it contains only the definition of a set of relations and integrity constraints!
- **A better approach**:  
  1) **Design** first a conceptual model: Conceptual Design  
  2) Then **translate this conceptual model** into a logical model: Logical Design

---

**Problematic of the Logical Design**

The **Logical Design** transforms the Conceptual Schema for a DM into a **Logical Schema**:

- Choice of the **type of logical schema**
- Translation of conceptual schema to logical schema
- Optimization (view materialization, fragmentation)

---

**Schemata derivations**

![Schemata derivations diagram](image)

**Design Phases**

- Source analysis and integration
- Conceptual design
- Workload and data volume
- Logical design
- ETL design
- Physical design

---

**Bernard ESPINASSE - Data Warehouse Logical Modelling and Design**
2. Logical Modeling: The Multidimensional Model

- The Multidimensional Model:
  - Fact,
  - Measures,
  - Dimensions
- Star and Snowflake Schemata
- Aggregates and Views

The Multidimensional Model: Fact and Dimensions

- Data is divided into **facts** (with **measures**) and **dimensions**
- **Facts**
  - are the important entity, e.g., a sale
  - have measures that can be aggregated, e.g., sales price
- **Dimensions**:
  - describe facts
  - Ex: a sale has the dimensions Product, Store and Time
- **Goal for dimensional modeling:**
  - Surround facts with as much context/dimensions as possible (redundancy may be ok in well-chosen places)
  - But you should not try to model all relationships in the data (unlike E/R and OO modeling!)

Facts (data) "live" in a multidimensional «cube»

The Multidimensional Model

- **Multidimensional Model:**
  - is a logical model
  - has one purpose: **Data analysis**
  - is the most popular data model for DW
  - is more **built in "meaning"**:
    - What is important
    - What describes the important
    - What we want to optimize
    - Automatic aggregations means **easy querying**
  - is **recognized by OLAP/BI tools**: Tools offer powerful query facilities based on MD design

Facts and subject

**Facts** represent the **subject** of the desired analysis: the "important" in the business that should be analyzed:

- A fact is most often identified via its **dimension values**:
  - A fact is a **non-empty cell**
  - Some models give facts an **explicit identity**
- Generally, a fact should:
  - be attached to **exactly one** dimension value in each dimension;
  - only be attached to dimension values in the **bottom levels**
  - **Ex**: if the lowest time granularity is day, for each fact the exact day should be specified
  - some models do not require this.
**Types of Facts**

Different *types of facts* are distinguished:

- **Event facts (transaction)**: a fact for every business event *(Ex: sale)*
- **« Fact-less » facts**:
  - A fact per event *(Ex: customer contact)*
  - No numerical measures
  - *An event happened for a dimension value combination*
- **Snapshot fact**:
  - A fact for every dimension combination at given time interval
  - *Captures current status* *(Ex: inventory)*
- **Cumulative snapshot facts**:
  - A fact for every dimension combination at given time interval
  - *Captures cumulative status up to now* *(Ex: sales to date)*

Every type of facts answers different questions: often event facts and snapshot facts exist.

---

**Dimension**

- **Dimensions** are the core of multidimensional databases
- Other types of databases do not support dimensions
- Dimensions are used for:
  - **Selection** of data
  - **Grouping** of data at the right level of detail
- Dimensions consist of **dimension values**
  
  *Ex*:
  - Product dimension has values "milk", "cream", ...
  - Time dimension has values "1/1/2001", "2/1/2001", ...
- Dimension values may have an **ordering**:
  - Used for comparing cube data across values
  - Especially used for Time dimension
  
  *Ex*: percentage of sales increase compared with last month

---

**Dimension and Hierarchies**

- Dimensions encode **hierarchies** with **levels**: Typically 3-5 levels (of detail)
- Dimension values are organized in a **tree structure** or **lattice**:
  
  *Ex*:
  - **Product**: Product -> Type -> Category
  - **Store**: Store -> Area -> City -> County
  - **Time**: Day -> Month -> Quarter -> Year
- Dimensions have a **bottom level** and a **top level** (ALL)
- **Levels** may have **attributes** simple, non-hierarchical information
  
  *Ex*: Day has Workday as attribute
- **General rule**: dimensions should contain much information:
  
  *Ex*:
  - *Time dimensions may contain holiday, season, events, ...*
  - Good dimensions have 50-100 or more attributes/levels

---

**Hierarchy example**

- A location dimension with attributes *street, city, province_or_state, and country* encodes implicitly the following hierarchy:

![Hierarchy Example Diagram](image-url)
The Multidimensional Model: Cube

• A cube may have many dimensions:
  - More than 3 – the term "hypercube" is sometimes used
  - Theoretically no limit for the number of dimensions
  - Typical cubes have 4-12 dimensions
• But only 2-3 dimensions can be viewed at a time:
  - Dimensionality reduced by queries via projection/aggregation
• A cube consists of cells:
  - A given combination of dimension values
  - A cell can be empty (no data for this combination)
  - A sparse cube has many empty cells
  - A dense cube has few empty cells
  - Cubes become sparser for many/large dimensions.

Granularity of facts

• Granularity of facts is important:
  - What does a single fact mean?
  - Determines the level of detail
  - Given by the combination of bottom levels
  - Ex: "total sales per store per day per product"
• Important for number of facts: Scalability
• Often the granularity is a single business transaction:
  - Ex: sale
  - Sometimes the data is aggregated (total sales per store per day per product)
  - Aggregation might be necessary for scalability

Measures

• Measures represent the fact property that users want to study and analyze:
  - Ex: the total sales price
• A measure has 2 components:
  - Numerical value (ex: sales price)
  - Aggregation formula (ex: SUM): used for aggregating / combining a number of measure values into one
• Additivity is an important property for measures:
  - Single fact table rows are (almost) never retrieved, but aggregations over millions of fact rows
• Measure value determined by the combination of dimension values:
  - Measure value is important for all aggregation levels.

Types of Measures

3 types of measures are distinguished:

• Additive measures:
  - Can be aggregated over all dimensions using SUM operator
  - Ex: sales price, gross profit computed from sales and cost
  - Often occur in event facts
• Semi-additive measures:
  - Cannot be aggregated over some dimensions – typically time
  - Ex: inventory: additive across time or store, non-additive across product
  - Often occur in snapshot facts
• Non-additive measures:
  - Cannot be aggregated over any dimensions
  - Ex: unit cost
  - Occur in all types of facts
General DW Design Steps

1. Choose the business process(es) to model:
   - *Ex: Sales*

2. Choose the granularity of the business process:
   - *Ex: Items by Store by Promotion by Day*
   - Low granularity is needed?
   - Are individual transactions necessary/feasible?

3. Choose the dimensions:
   - *Ex: Time, Store, *

4. Choose the measures:
   - *Ex: Dollar_sales, unit_sales, dollar_cost, customer_count*

3. Implementing a Dimensional Model in ROLAP

- Star schema
- Snowflake schema
- Aggregates and views

Star Schema

- Is a **common approach** to draw a dimensional model
- Consists of: **one fact table** and **many dimension tables**:

![Star Schema Diagram]

Dimension Schema and Instance in Star Schema

<table>
<thead>
<tr>
<th>Date</th>
<th>DateKey</th>
<th>Date</th>
<th>DateFull</th>
<th>DayOfWeek</th>
<th>CalMonth</th>
<th>CalYear</th>
<th>Weekday</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-01-02</td>
<td>Jan 1, 2002</td>
<td>Tuesday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>01-02-02</td>
<td>Jan 2, 2002</td>
<td>Wednesday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>01-03-02</td>
<td>Jan 3, 2002</td>
<td>Thursday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>01-04-02</td>
<td>Jan 4, 2002</td>
<td>Friday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>01-05-02</td>
<td>Jan 5, 2002</td>
<td>Saturday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
<td></td>
</tr>
</tbody>
</table>
Relational “Star Schema” Evaluation

Forces:
- Simple -> ease-of-use
- Relatively flexible
- Fact table is normalized (dimensions tables are not necessary normalized)
- Dimension tables often relatively small
- “Recognized” by many RDBMS -> good performance

Weakness:
- Hierarchies are “hidden” in the columns
- Dimension tables are de-normalized

=> From relational Star schema to relational « Snowflake » schema

Constellation Schema
- Merge several star schemata, which use common dimensions
- Has consequently several facts and dimensions common or not
  Ex : Drugs sales in pharmacies

- Make up of 2 star-schemas:
  - first concerning sales in pharmacies (SALE)
  - second concerning medecin prescription (PRESCRIPTION)
- DATE and STORE dimensions are shared by PRESCRIPTION and SALE facts.

Snowflake Schema Evaluation

Forces:
- Hierarchies are made explicit/visible
- Very flexible
- Dimension tables use less space:
  - However this is a minor saving
  - Disk space of dimensions is typically less than 5 percent of disk for DW

Weakness:
- Harder to use due to many joins
- Worse performance:
  Ex : efficient bitmap indexes are not applicable
**Aggregates and Views**

*View = Fact table that include Aggregate Data*
- **Primary view** : defined in the fact schema dimensions (primary group-by sets), populated by operational data
- **Secondary views** : new fact tables including aggregates) (secondary group-by sets), populated by others views (not by operational data)

**Star schema**

- primary view = V1
- secondary views: V2, V3, V4, V5

**Fact table and dimension tables**

- Primary view (V1):
  - Product, Date, Store
  - Attributes: SalesQuantity, Receipts, UnitPrice

- Secondary views (V2 to V5):
  - Additional attributes: StoreCity, Type, Category

- Dimension tables:
  - Store: keyStore, store, storeCity, state, country
  - Product: keyProduct, type, category

**Relational Schema with Aggregate Data (1)**

**Solution 1**: the easier solution in a star schema, is to store both primary view data and secondary view data in the same fact table:

<table>
<thead>
<tr>
<th></th>
<th>keyStore</th>
<th>keyDate</th>
<th>keyProduct</th>
<th>quantity</th>
<th>receipts</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1 view</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>170</td>
<td>85</td>
</tr>
<tr>
<td>V2 view</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>300</td>
<td>150</td>
</tr>
<tr>
<td>V3 view</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1700</td>
<td>850</td>
</tr>
<tr>
<td>V4 view</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4500</td>
<td>2200</td>
</tr>
<tr>
<td>V5 view</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>9000</td>
<td>4500</td>
</tr>
</tbody>
</table>

**Aggregation level of fact table tuples specified by corresponding tuples in dimension table**

- The dimension table STORE related to aggregate data will have NULL values in all attributes whose aggregation level is finer than the current one
- In the fact table SALES:
  - first tuple is related to one sale,
  - second tuple store aggregate data for every sale in Austin,
  - third tuple sums up all the sales in Texas.

*Only fact table is used for request, but performance are bad because table is too big*

**Relational Schema with Aggregate Data (2)**

**Solution 2**: a common solution is to store different group-by sets into separate fact tables in a constellation schema:

**Solution 3**: dimension tables are replicated for every view including only the set of attributes that are valid for the aggregation level at which that dimension table is created:

*here a fact table concerns primary view V1 and an other V5 secondary view

- Dimensions tables can be merged as in solution 1, or replicate for each aggregate view
- The fact tables that correspond to the group-by sets where one or more dimensions are completely aggregated do not have foreign key referencing these dimensions.

The size of fact table size is far larger than size of the dimension tables, then the performance depends on fact table optimisation

**Relational Schema with Aggregate Data (3)**

**Solution 3**: dimension tables are replicated for every view including only the set of attributes that are valid for the aggregation level at which that dimension table is created:

- There are V1 primary view and V5 secondary view with their specific dimension tables
- Note that the key of the fact table for secondary view V5 has not attribute related to store hierarchy, which is completely aggregated

This solution is the best in performance because the table access is optimized, however replicate dimension tables need disk space
**Relational Schema with Aggregate Data (4)**

Solution 4: a compromise, where aggregate views are materialized in a snowflake schema:

- take advantage of the optimization achieved with aggregate data by aggregation level without replicating dimension table
- the sales fact is modeled by a snowflake schema

**Problematic of the Logical Design**

The Logical Design transforms the Conceptual Schema for a DM into a Logical Schema:

- Choice of the type of logical schema
- Translation of conceptual schema to logical schema
- Optimization (view materialization, fragmentation)

Different principles from the one used in operational databases:

- data redundancy
- denormalization of tables

**4. Logical Design: From Fact Schema to Logical Schema in ROLAP context**

- From fact schema to relational star-schema: basic rules
- Examples towards Relational Star Schema
- Examples towards Relational Snowflake Schema
- Advanced logical modelling

**Schemata derivations (1)**
**Logical Design:**

- transforms the Fact Schemata into a Logical Schema, a Relational Schema (in ROLAP strategy)

**Conceptual Design:**

Step to derive Dimensional Fact (DF) schemata from Relational schema of operational sources (DB):

1. Finding and defining FACT from Relational/E-R schema
2. Building the Attribute Tree from Relational/E-R schema
3. Building the Fact Schemata from Attribute Tree

**From Fact Schema (DFM) to Relational Schema: Basic Rules**

R1. From «Fact schema» to «Tables»:

- Fact box of the fact schema leads to create a Fact Table (FT)
- dimension leads to create a Dimension Table (DT)

R2. Dimension table (DT) attributes:

- first dimension attribute of fact schema become DT attribute
- the first dimension attribute (first level) become DT key attribute

R3. Fact table (FT) attributes:

- DT key attribute of each dimension become FT foreign key
- measure attributes of fact schema become FT measure

**Relational Star-Schema “Sale” example (1)**

Fact schema:

- category
- type
- product
- fact
- dimensions
- measures

- year
- month
- week
- store
- city
- country
- quantity
Relational Star-Schema “Sale” example (2)

Relational schema:
- SaleFT (ProductId, StoreId, DateId, Quantity)
- WeekDT (ProductId, Product, Type, Category)
- StoreDT (StoreId, Store, City, State, Country)
- DateDT (DateId, Day, Month, Year)

Instances:

<table>
<thead>
<tr>
<th>ProductDT</th>
<th>DateDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductId</td>
<td>DateId</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>May</td>
</tr>
<tr>
<td>1997</td>
<td></td>
</tr>
</tbody>
</table>

SalesFT

<table>
<thead>
<tr>
<th>ProductId</th>
<th>StoreId</th>
<th>DateId</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>575</td>
</tr>
</tbody>
</table>

StoreDT

<table>
<thead>
<tr>
<th>StoreId</th>
<th>Store</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bilka</td>
<td>Aalborg</td>
<td>Denmark</td>
</tr>
<tr>
<td>2</td>
<td>Spar</td>
<td>Bolzano</td>
<td>Italy</td>
</tr>
</tbody>
</table>

OLAP Query on Star schema:

Total quantity sold for each product type, week, and city, only for food products:

```
SELECT City, Week, Type, SUM(Quantity)
FROM WeekDT, StoreDT, ProductDT, SaleFT
WHERE WeekDT.WeekID = SaleFT.WeekID AND
  StoreDT.StoreID = SaleFT.StoreID AND
  ProductDT.ProductID = SaleFT.ProductID AND
  ProductDT.Category = 'Food'
GROUP BY City, Week, Type;
```
OLAP Query on Snowflake schema:

Total quantity sold for each product type, week, and city, only for food products:

```
SELECT City, Week, Type, SUM(Quantity)
FROM WeekDT, StoreDT, ProductDT, CityDT, TypeDT, SaleFT
WHERE WeekDT.WeekID = SaleFT.WeekID AND
    StoreDT.StoreID = SaleFT.StoreID AND
    ProductDT.ProductID = SaleFT.ProductID AND
    StoreDT.CityID = CityDT.CityID AND
    ProductDT.TypeID = TypeDT.TypeID AND
    ProductDT.Category = 'Food'
GROUP BY City, Week, Type;
```
Advanced Logical Modeling: Shared Hierarchies

Conceptual Modelling: Entire portion of hierarchies are frequently replicated 2 or more time in fact schemata. Ex: calling and called phone numbers …

Logical Modelling: we have to avoid multiples dimension tables, which contain all or part of the same data. Ex: the hierarchies contain all the same data, we choose to insert 2 foreign keys referencing the one table that models the telephone number in the fact table:

<table>
<thead>
<tr>
<th>CALLS</th>
<th>NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyCalling</td>
<td>telNumber</td>
</tr>
<tr>
<td>KeyCalled</td>
<td>type</td>
</tr>
<tr>
<td>number</td>
<td>district</td>
</tr>
<tr>
<td>duration</td>
<td></td>
</tr>
</tbody>
</table>

Advanced Logical Modeling: Multiple Arcs

Conceptual Modelling: A multiple arc models a many-to-many association between the 2 dimensional attributes it connects. Ex: in a fact schema modeling the sales of books, book is a dimension, but it would not be relevant to model author as a dimensional child attribute of book because many different authors can write many books ⇒ the relationship between books and authors is modeled as a multiple arc:

Logical Modelling: solution is to insert a bridge-table to model multiple arcs:

<table>
<thead>
<tr>
<th>CALLS</th>
<th>NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyCalling</td>
<td>telNumber</td>
</tr>
<tr>
<td>KeyCalled</td>
<td>type</td>
</tr>
<tr>
<td>number</td>
<td>district</td>
</tr>
<tr>
<td>duration</td>
<td></td>
</tr>
</tbody>
</table>

Advanced Logical Modeling: Additivity Issues

Conceptual Modelling: Non-additive measure can be explicitly specify with its operator(s) used for aggregation – other that SUM

Ex: AVG and MIN for inventory level:

Logical Modelling: set a new measure for each aggregation operator.

Count is a support measure necessary to calculate average level

minLevel is required to calculate the minimum level for each month and for each product type.

4. ROLAP schema in XML for Mondrian

- Example
- Examples towards Relational Star Schema
- Examples towards Relational Snowflake Schema
- Advanced logical modelling
The Mondrian ROLAP server

- **Mondrian** is an open-source relational online analytical processing (ROLAP) server
- It is also known as Pentaho Analysis Services and is a component of the Pentaho Business Analytics suite
- In Mondrian, a **cube schema**, written in an XML syntax, defines a mapping between the physical structure of the relational data warehouse and the multidimensional cube
- A **cube schema** contains the declaration of cubes, dimensions, hierarchies, levels, measures, and calculated members
- A **cube schema** does **not define** the data source, this is done using a JDBC connection string.

A cube schema definition in Mondrian in XML (1)

1. `<Schema name="MySocietyDW"`
2. `description="Sales cube of MySociety company">
3. `</PhysicalSchema>
4. ...
5. `</PhysicalSchema>
6. `</Dimension name="Time" table="Time" ... >`
7. ...
8. `</Dimension>
9. `</Cube>
10. `</Dimensions>
11. ...
12. `</MeasureGroups>
13. `</MeasureGroup name="Sales" table="Sales">
14. `</Measures>
15. `</MeasureGroups>
16. `</Cube>
17. `</Schema>`

The Mondrian ROLAP server (2)

- The Schema element (starting in line 1) defines all other elements in the schema
- The **PhysicalSchema** element (lines 3–5) defines the **tables that are the source data** for the dimensions and cubes, and the foreign key links between these tables
- The **Dimension element** (lines 6–8) defines the **shared dimension Time**, which is used several times in the MySociety cube for the role-playing dimensions OrderDate, DueDate, and ShippingDate. Shared dimensions can also be used in several cubes
- The **Cube element** (lines 9–23) define the **Sales cube**. A cube schema contains dimensions and measures, the latter organized in measure groups
- The **Dimensions element** (lines 10–12) defines the **dimensions of the cube**
- The **measure groups** are defined using the element **MeasureGroups** (lines 13–22)
- The **measure group Sales** (lines 14–21) defines the **measures** using the **Measures element** (lines 15–17)
- The **DimensionLinks element** (lines 18–20) defines how measures relate to dimensions.
**Browsing the cube with Saiku**

Saiku is an open-source analytics client server. In the figure:

- the countries of customers and the categories of products are displayed on rows, the years are displayed on columns, and
- the sales amount measure is displayed on cells.

Saiku also allows the user to write OLAP queries in MDX directly on an editor.