Methodological framework
Logical Modeling: The Multidimensional Model
Logical Design: From Fact schema to Logical schema in ROLAP context

Plan
1. Methodological Framework
   - Conceptual Design & Logical Design
   - Design Phases and schemata derivations
2. Logical Modelling: The Multidimensional 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 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

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
  - …
**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) which is then **translated** into a **logical model**: Logical Design

---

**Schemata derivations**

**Design Phases**

---

**2. Logical Modeling: The Multidimensional Model**

- Problematic of the Logical Design
- The Multidimensional Model: fact, measures, dimensions
- Star and Snowflake schemata
- Aggregates and views
**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 schemata
- Optimization (view materialization, fragmentation)

**Conceptual Schema**
- Workload
- Data Volume
- Constraints

**Logical Design**
- Logical Schema

Different principles from the one used in operational databases:
- Data redundancy
- Denormalization of tables

*We adopt the logical « Multidimensional Model »*

---

**The Multidimensional Model**

**Multidimensional Model**:
- Is a **logical model** and has one purpose: Data analysis
- Is better at that purpose than E/R model:
  - Less flexible
  - Not suited for OLTP systems
- Is more built in “meaning”:
  - What is important
  - What describes the important
  - What we want to optimize
  - Automatic aggregations means easy querying
- Is the most popular data model for DW
- Is recognized by OLAP/BI tools: Tools offer powerful query facilities based on MD design

---

**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 »**

---

**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 (sale)
- **« Fact-less » facts**:
  - A fact per event (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 (inventory)
- **Cumulative snapshot facts**:
  - A fact for every dimension combination at given time interval
  - Captures cumulative status up to now, e.g., 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:
  - 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:

```
- country
  - Province_or_state
    - city
      - street
```
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
- Generally, transaction detail can be handled: Except perhaps huge clickstreams, etc.

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 meaningful for all aggregation levels.

Types of Measures

3 types of measures are distinguished:

- Additive measures:
  - Can be aggregated over all dimensions using SUM
  - 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, customer_count: 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

### Star Schema

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

  ![Star Schema Diagram]

### 3. Implementing a Dimensional Model in ROLAP

- **Star schema**
- **Snowflake schema**
- **Aggregates and views**

### Dimension Schema and Instance in Star Schema

- **Schema** of dimension Date
- **Instance** of dimension Date

<table>
<thead>
<tr>
<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>January 1, 2002</td>
<td>Tuesday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
</tr>
<tr>
<td>2</td>
<td>01/02/02</td>
<td>January 2, 2002</td>
<td>Wednesday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
</tr>
<tr>
<td>3</td>
<td>01/03/02</td>
<td>January 3, 2002</td>
<td>Thursday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
</tr>
<tr>
<td>4</td>
<td>01/04/02</td>
<td>January 4, 2002</td>
<td>Friday</td>
<td>January</td>
<td>2002</td>
<td>Weekday</td>
</tr>
<tr>
<td>5</td>
<td>01/05/02</td>
<td>January 5, 2002</td>
<td>Saturday</td>
<td>January</td>
<td>2002</td>
<td>Weekend</td>
</tr>
</tbody>
</table>
Relational “Star Schema” Evaluation

Forces:
• Simple -> ease-of-use
• Relatively flexible
• Fact table is normalized
• Dimension tables often relatively small
• “Recognized” by many RDBMSes -> good performance

Weakness:
• Hierarchies are "hidden" in the columns
• Dimension tables are de-normalized

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

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 aggregate(s) (secondary group-by sets), populated by others views (not by operational data)

Relational Schemas 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:

Snowflake Schema

• Is obtained from a star schema by breaking down one or more dimension tables into smaller tables to remove transitive functional dependencies
• Dimension table referenced in the fact table are "primary dimension tables" and other are "secondary dimension tables"

• v1 = primary view and v2 … v5 = secondary views
• if vi -> vj then vj can be calculated by aggregating vi data

Bernard ESPINASSE - Data Warehouse Logical Modelling and Design

25

26

27

28
**Relational Schemas with Aggregate Data (2)**

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

- Here a fact table concerns primary view V1 and another V5 secondary view.
- Dimension 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 performance depends on the fact table optimisation.

**Relational Schemas 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:

- Here are 2 different views V1 and V5 with their specific dimension tables.
- Note that the key of the fact table for secondary view V5 has no attribute related to store hierarchy, which is completely aggregated.
- This solution is the best in performance because the table access are optimized, however, replicate dimension tables need disk space.

**Relational Schemas 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.

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

From « Fact schema » to « Tables » :

• The fact box of the fact schema leads to create a fact table (FT)
• The dimension leads to create a dimension table (DT)

Dimension table (DT) attributes :

• dimension attribute become FD attribute
• the first dimension attribute (first level) become FD key attribute

Fact table (FT) attributes :

• FD key attribute of each dimension become FT foreign key
• measure attributes become FT attribute

Relational Star-Schema “Sale” example (1)

Fact schema :

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)

Relational Star-Schema “Sale” example (3)

Instances :

<table>
<thead>
<tr>
<th>ProductDT</th>
<th>DateDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductId</td>
<td>DateId</td>
</tr>
<tr>
<td>Product</td>
<td>Day</td>
</tr>
<tr>
<td>Type</td>
<td>Month</td>
</tr>
<tr>
<td>Category</td>
<td>Year</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SalesFT</th>
<th>StoreDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductId</td>
<td>StoreId</td>
</tr>
<tr>
<td>Product</td>
<td>DateId</td>
</tr>
<tr>
<td>Type</td>
<td>Quantity</td>
</tr>
<tr>
<td>Category</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StoreId</th>
<th>Store</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sika</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;
```

Relational Snowflake Schema “Sale” example (1)

Relational schema:
- SaleFT (ProductID, StoreID, DateID, Sale)
- ProductDT (ProductID, Product, TypeID)
- ProductType (TypeID, Type, CategoryID)
- StoreDT (StoreID, Store, City, State, Country)
- DateDT (DateID, Day, MonthID)
- MonthYearDescription (MonthID, Month, YearID)

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;
```
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

---

Snowflake Schema “Rent a Car” example (1)

« RENTAL » Fact schema obtained by the Conceptual Design:

- Area
- Country
- State
- City
- idDate
- dtDate
- dtOffice
- pickupOffice
- dropoffOffice
- DT_DATE
- pickupDate
- dropoffDate
- Model
- Category
- RegistrationDate
- idCar
- idOffice
- Office
- State
- Month
- Year
- Duration
- Miles
- PaymentMode
- Amount
- Discount
- idCar
- Car
- Fuel
- Model
- Brand
- Category
- RegistrationDate

Snowflake Schema “Rent a Car” example (2)

Logical schema obtained from previous rental fact schema is the following snowflake schema:

- DT_CAR (idCar, Car, Fuel, Model, Brand, Category, registrationDate:DT_DATE)
- DT_OFFICE (idOffice, Office, City, State, Country, Area)
- DT_DATE (idDate, Date, Month, Year)
- FT_RENTAL (pickupOffice:DT_OFFICE, dropoffOffice:DT_OFFICE, idCar:DT_CAR, pickupDate:DT_DATE, PaymentMode, Amount, Discount, Duration, Miles)

---

Market Basket Analysis

**Benefits:**
- Better understanding of customer behavior
- Improved decision-making
- Enhanced product recommendations

**Challenges:**
- High data volume
- Complex data relationships
- Resource-intensive processing

---

Data Warehouse Logical Modelling and Design

**STAR Schema:**
- Day facts
- Sales fact
- Transaction fact
- Customer fact
- Product fact
- Time fact

**Snowflake Schema:**
- Fact schema
- Dimension schema
- Relationship schema

**Worst Case Scenario:**
- Sales fact
- Customer fact
- Product fact
- Time fact

**Benefits:**
- Improved query performance
- Easier data manipulation

**Challenges:**
- Increased complexity
- Resource-intensive processing

---

Snowflake Schema “Rent a Car” example (3)

**Snowflake schema:**

*Available as a Diagram*
Advanced Logical Modeling: Descriptive Attributes (1)

Conceptual Modelling:
- A **descriptive attribute** contains additional information about an attribute of the hierarchy
- It cannot be used for aggregation

Logical Modelling:
- If a **descriptive attribute** is linked to a **dimensional attribute**, it have to be included in the dimension table for the hierarchy that contains it
  - Ex: the **size** of the product have to be included in PRODUCT table
- If a **descriptive attribute** is linked to a **fact attribute**, it have to be included in the fact table with measures

---

Advanced Logical Modeling: Descriptive Attributes (2)

Conceptual Modelling: A **cross-dimensional attribute** b is a dimensionnal or **descriptive attribute** whose value is defined by the combination of 2 or more dimensional attributes a1, ...am, possibly belonging to different hierarchies.

Logical Modelling: translation leads to a new table that includes the b **cross-dimensional attribute** and has the a1, ...am attributes as primary key

Ex.: a product Value Added Tax (VAT) depends both on the product category and on the country where the product is sold

---

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:

---

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:
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