Data warehouse design

Elena Baralis
Politecnico di Torino
Risk factors

- High user expectation
  - the data warehouse is *the* solution of the company’s problems
- Data and OLTP process quality
  - incomplete or unreliable data
  - non integrated or non optimized business processes
- “Political” management of the project
  - cooperation with “information owners”
  - system acceptance by end users
  - deployment
    - appropriate training
Data warehouse design

• Top-down approach
  – the data warehouse provides a global and complete representation of business data
  – significant cost and time consuming implementation
  – complex analysis and design tasks

• Bottom-up approach
  – incremental growth of the data warehouse, by adding data marts on specific business areas
  – separately focused on specific business areas
  – limited cost and delivery time
  – easy to perform intermediate checks
Business Dimensional Lifecycle (Kimball)

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Data mart design

CONCEPTUAL DESIGN

user requirements

workload
data volume
logical model

logical schema

LOGICAL DESIGN

workload
data volume
DBMS

physical schema

PHYSICAL DESIGN

feeding schema

FEEDING DESIGN

reconciled schema

reconciled schema

RECONCILIATION

operational
source
schemas

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Requirement analysis

Elena Baralis
Politecnico di Torino
Requirement analysis

• It collects
  – data analysis requirements to be supported by the data mart
  – implementation constraints due to existing information systems

• Requirement sources
  – business users
  – operational system administrators

• The first selected data mart is
  – crucial for the company
  – fed by (few) reliable sources
Application requirements

• Description of relevant events (facts)
  – each fact represents a category of events which are relevant for the company
    • examples: (in the CRM domain) complaints, services
  – characterized by descriptive dimensions (setting the granularity), history span, relevant measures
  – informations are gathered in a glossary

• Workload description
  – periodical business reports
  – queries expressed in natural language
    • example: number of complaints for each product in the last month
Structural requirements

• Feeding periodicity
• Available space for
  – data
  – derived data (indices, materialized views)
• System architecture
  – level number
  – dependent or independent data marts
• Deployment planning
  – start up
  – training
Conceptual design

Elena Baralis
Politecnico di Torino
Conceptual design

- No currently adopted modeling formalism
  - ER model not adequate
- **Dimensional Fact Model** (Golfarelli, Rizzi)
  - graphical model supporting conceptual design
  - for a given fact, it defines a *fact schema* modelling
    - dimensions
    - hierarchies
    - measures
  - it provides design documentation both for requirement review with users, and after deployment
Dimensional Fact Model

- **Fact**
  - it models a set of relevant events (sales, shippings, complaints)
  - it evolves with time

- **Dimension**
  - it describes the analysis coordinates of a fact (e.g., each sale is described by the sale date, the shop and the sold product)
  - it is characterized by many, typically categorical, attributes

- **Measure**
  - it describes a numerical property of a fact (e.g., each sale is characterized by a sold quantity)
  - aggregates are frequently performed on measures

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
DFM: Hierarchy

- Each dimension can have a set of associated attributes
- The attributes describe the dimension at different abstraction levels and can be structured as a hierarchy
- The hierarchy represents a generalization relationship among a subset of attributes in a dimension (e.g., geographic hierarchy for the shop dimension)
- The hierarchy represents a functional dependency (1:n relationship)

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Comparison with ER

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Advanced DFM

Aggregation

• Aggregation computes measures with a coarser granularity than those in the original fact schema
  – detail reduction is usually obtained by climbing a hierarchy
  – standard aggregate operators: SUM, MIN, MAX, AVG, COUNT

• Measure characteristics
  – additive
  – not additive: cannot be aggregated along a given hierarchy by means of the SUM operator
  – not aggregable
Measure classification

• Stream measures
  – can be evaluated cumulatively at the end of a time period
  – can be aggregated by means of all standard operators
  – examples: sold quantity, sale amount

• Level measures
  – evaluated at a given time (snapshot)
  – not additive along the time dimension
  – examples: inventory level, account balance

• Unit measures
  – evaluated at a given time and expressed in relative terms
  – not additive along any dimension
  – examples: unit price of a product
Aggregate operators

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Aggregate operators

- Distributive
  - can always compute higher level aggregations from more detailed data
  - examples: sum, min, max
### Non distributive operators

**Measure:** unit price

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>Product</th>
<th>Year</th>
<th>I’99</th>
<th>II’99</th>
<th>III’99</th>
<th>IV’99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home cleaning</td>
<td>Washing powder</td>
<td>Brillo</td>
<td>1999</td>
<td>2</td>
<td>2</td>
<td>2,2</td>
<td>2,5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sbianco</td>
<td></td>
<td>1,5</td>
<td>2</td>
<td>2</td>
<td>2,5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lucido</td>
<td></td>
<td>–</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Soap</td>
<td>Manipulite</td>
<td></td>
<td>1</td>
<td>1,2</td>
<td>1,5</td>
<td>1,5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scent</td>
<td></td>
<td>1,5</td>
<td>1,5</td>
<td>2</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>Year</th>
<th>I’99</th>
<th>II’99</th>
<th>III’99</th>
<th>IV’99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home cleaning</td>
<td>Wash. p.</td>
<td></td>
<td>1,75</td>
<td>2,17</td>
<td>2,40</td>
<td>2,67</td>
</tr>
<tr>
<td></td>
<td>Soap</td>
<td></td>
<td>1,25</td>
<td>1,35</td>
<td>1,75</td>
<td>1,50</td>
</tr>
<tr>
<td></td>
<td>avg:</td>
<td></td>
<td>1,50</td>
<td>1,76</td>
<td>2,08</td>
<td>2,09</td>
</tr>
</tbody>
</table>

**From Golfarelli, Rizzi,”Data warehouse, teoria e pratica della progettazione”, McGraw Hill 2006**
Aggregate operators

- Distributive
  - can always compute higher level aggregations from more detailed data
  - examples: sum, min, max

- Algebraic
  - can compute higher level aggregations from more detailed data only when supplementary support measures are available
  - examples: avg (it requires count)

- Olistic
  - *can not* compute higher level aggregations from more detailed data
  - examples: mode, median
Advanced DFM

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Advanced DFM

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Factless fact schema

- Some events are not characterized by measures
  - empty (i.e., factless) fact schema
  - it records occurrence of an event
- Used for
  - counting occurred events (e.g., course attendance)
  - representing events not occurred (coverage set)

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Representing time

• Data modification over time is explicitly represented by event occurrences
  – time dimension
  – events stored as facts

• Also dimensions may change over time
  – modifications are typically slower
    • slowly changing dimension [Kimball]
  – examples: client demographic data, product description
  – if required, dimension evolution should be explicitly modeled
How to represent time (type I)

• Snapshot of the current value
  – data is overwritten with the current value
  – it overrides the past with the current situation
  – used when an explicit representation of the data change is not needed
  – example
    • customer Mario Rossi changes marital status after marriage
    • all his purchases correspond to the “married” customer
How to represent time (type II)

• Events are related to the temporally corresponding dimension value
  – after each state change in a dimension
    • a new dimension instance is created
    • new events are related to the new dimension instance
  – events are partitioned after the changes in dimensional attributes
  – example
    • customer Mario Rossi changes marital status after marriage
    • his purchases are partitioned in purchases performed by “unmarried” Mario Rossi and purchases performed by “married” Mario Rossi (a new instance of Mario Rossi)
How to represent time (type III)

- All events are mapped to a dimension value sampled at a given time
  - it requires the explicit management of dimension changes during time
    - the dimension schema is modified by introducing
      - two timestamps: validity start and validity end
      - a new attribute which allows identifying the sequence of
        modifications on a given instance (e.g., a “master” attribute
        pointing to the root instance)
    - each state change in the dimension requires the
      creation of a new instance
How to represent time (type III)

- Example
  - customer Mario Rossi changes marital status after marriage
  - validity end timestamp of first Mario Rossi instance is given by the marriage date
  - validity start timestamp of the new instance is the same day
  - purchases are partitioned as in type II
  - a new attribute allows tracking all changes of Mario Rossi instance
Workload

• Workload defined by
  – standard reports
  – approximate estimates discussed with users

• Actual workload difficult to evaluate at design time
  – if the data warehouse succeeds, user and query number may grow
  – query type may vary over time

• Data warehouse tuning
  – performed after system deployment
  – requires monitoring the actual system workload
Data volume

• Estimation of the space required by the data mart
  – for data
  – for derived data (indices, materialized views)

• To be considered
  – event cardinality for each fact
  – domain cardinality (number of distinct values) for hierarchy attributes
  – attribute length

• It depends on the temporal span of data storage

• Sparsity
  – occurred events are not all combinations of the dimension elements
  – example: the percentage of products actually sold in each shop and day is roughly 10% of all combinations
Sparsity

- It decreases with increasing data aggregation level
- May significantly affect the accuracy in estimating aggregated data cardinality

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Logical design

Elena Baralis

Politecnico di Torino
Logical design

• We address the relational model (ROLAP)
  – inputs
    • conceptual fact schema
    • workload
    • data volume
    • system constraints
  – output
    • relational logical schema

• Based on different principles with respect to traditional logical design
  – data redundancy
  – table denormalization
Star schema

- **Dimensions**
  - one table for each dimension
  - surrogate (generated) primary key
  - it contains all dimension attributes
  - hierarchies are not explicitly represented
    - all attributes in a table are at the same level
  - totally denormalized representation
    - it causes data redundancy

- **Facts**
  - one fact table for each fact schema
  - primary key composed by foreign keys of all dimensions
  - measures are attributes of the fact table
Star schema

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Snowflake schema

• Some functional dependencies are separated, by partitioning dimension data in several tables
  – a new table separates two branches of a dimensional hierarchy (hierarchy is cut on a given attribute)
  – a new foreign key correlates the dimension with the new table

• Decrease in space required for storing the dimension
  – decrease is frequently not significant

• Increase in cost for reading entire dimension
  – one or more joins are needed
From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Star or snowflake?

• The snowflake schema is usually not recommended
  – storage space decrease is rarely beneficial
    • most storage space is consumed by the fact table (difference with dimensions is several orders of magnitude)
  – cost of join execution may be significant

• The snowflake schema may be useful
  – when part of a hierarchy is shared among dimensions (e.g., geographic hierarchy)
  – for materialized views, which require an aggregate representation of the corresponding dimensions
Multiple edges

- Implementation techniques
  - bridge table
    - new table which models many to many relationship
    - new attribute weighting the contribution of tuples in the relationship
  - push down
    - multiple edge integrated in the fact table
    - new corresponding dimension in the fact table
Multiple edges


Copyright – All rights reserved
Multiple edges

• Queries
  – Weighted query: consider the weight of the multiple edge
    • example: author income
    • by using bridge table:
      
      \[
      \text{SELECT Author\_ID, SUM(Income*Weight)}
      \]
      
      \[
      \ldots
      \]
      
      \[
      \text{group by Author\_ID}
      \]
  – Impact query: do not consider the weight of the multiple edge
    • example: book copies sold for each author
    • by using bridge table:
      
      \[
      \text{SELECT Author\_ID, SUM(Quantity)}
      \]
      
      \[
      \ldots
      \]
      
      \[
      \text{group by Author\_ID}
      \]
Multiple edges

• Comparison
  – weight is explicit in the bridge table, but wired in the fact table for push down
    • (push down) hard to perform impact queries
    • (push down) weight is computed when feeding the DW
    • (push down) weight modifications are hard
  – push down causes significant redundancy in the fact table
  – query execution cost is lower for push down
    • less joins
Degenerate dimensions

- Dimensions with a single attribute
Degenerate dimensions

• Implementations
  – (usually) directly integrated into the fact table
    • only for attributes with a (very) small size
  – junk dimension
    • single dimension containing several degenerate dimensions
    • no functional dependencies among attributes in the junk dimension
      – all attribute value combinations are allowed
      – feasible only for attribute domains with small cardinality
Junk dimension

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006