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

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
  – feded 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

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

<table>
<thead>
<tr>
<th>category</th>
<th>type</th>
<th>product</th>
<th>1999</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>year</td>
<td>quart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>home cleaning</td>
<td>washing powder</td>
<td>Brillo</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sbianco</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lucido</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Manipulite</td>
<td>15</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scent</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>soap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>milk</td>
<td></td>
<td>Latte F Slurp</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Latte U Slurp</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yogurt Slurp</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>soda</td>
<td></td>
<td>Bevimi</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Colissima</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

Aggregate operators

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>home cleaning</td>
<td>washing</td>
<td>Brillo</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2,2</td>
<td>2,5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>powder</td>
<td>Sbianco</td>
<td></td>
<td></td>
<td>1,5</td>
<td>1,5</td>
<td>2</td>
<td>2,5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lucido</td>
<td></td>
<td></td>
<td>–</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>soap</td>
<td>Manipulite</td>
<td></td>
<td></td>
<td>1</td>
<td>1,2</td>
<td>1,5</td>
<td>1,5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scent</td>
<td></td>
<td></td>
<td>1,5</td>
<td>1,5</td>
<td>2</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

Measure: unit price

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>home cleaning</td>
<td>wash. p.</td>
<td>wash. p.</td>
<td></td>
<td></td>
<td>1,75</td>
<td>2,17</td>
<td>2,40</td>
<td>2,67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>soap</td>
<td>soap</td>
<td></td>
<td></td>
<td>1,25</td>
<td>1,35</td>
<td>1,75</td>
<td>1,50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>avg:</td>
<td></td>
<td></td>
<td>1,50</td>
<td>1,76</td>
<td>2,08</td>
<td>2,09</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>home clean.</td>
<td></td>
<td></td>
<td></td>
<td>1,50</td>
<td>1,84</td>
<td>2,14</td>
<td>2,38</td>
<td></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

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

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
Snowflake schema


Copyright – All rights reserved
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

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

Copyright – All rights reserved
Materialized views

Elena Baralis
Politecnico di Torino
Materialized views

- Precomputed summaries for the fact table
  - explicitly stored in the data warehouse
  - provide a performance increase for aggregate queries

\[ v_1 = \{\text{product, date, shop}\} \]
\[ v_2 = \{\text{type, date, city}\} \]
\[ v_3 = \{\text{category, month, city}\} \]
\[ v_4 = \{\text{type, month, region}\} \]
\[ v_5 = \{\text{quarter, region}\} \]

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

• Defined by SQL statements
• Example: definition of $v_3$
  – Starting from base tables or views with higher granularity
    
    \[
    \text{group by City, Category, Month}
    \]
  – Aggregation (SUM) on Quantity, Income measures
  – Reduction of detail in dimensions
Materialized views

- Materialized views may be exploited for answering several different queries
  - not for all aggregation operators

Materialized view selection

• Huge number of allowed aggregations
  – most attribute combinations are eligible
• Selection of the “best” materialized view set
• Cost function minimization
  – query execution cost
  – view maintainance (update) cost
• Constraints
  – available space
  – time window for update
  – response time
  – data freshness
Materialized view selection

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

Copyright – All rights reserved

DATA WAREHOUSE: DESIGN - 52

Elena Baralis
Politecnico di Torino
Materialized view selection

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

Copyright – All rights reserved
Materialized view selection

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

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

Copyright – All rights reserved
Physical design

Elena Baralis
Politecnico di Torino
Physical design

• Workload characteristics
  – aggregate queries which require accessing a large fraction of each table
  – read-only access
  – periodic data refresh, possibly rebuilding physical access structures (indices, views)

• Physical structures
  – index types different from OLTP
    • bitmap index, join index, bitmapped join index, ...
    • B⁺-tree index not appropriate for
      – attributes with low cardinality domains
      – queries with low selectivity
  – materialized views
    • query optimizer should be able to exploit them
Physical design

• Optimizer characteristics
  – should consider statistics when defining the access plan (cost based)
  – aggregate navigation

• Physical design procedure
  – selection of physical structures supporting most frequent (or most relevant) queries
  – selection of structures improving performance of more than one query
  – constraints
    • disk space
    • available time window for data update
Physical design

• Tuning
  – a posteriori change of physical access structures
  – workload monitoring tools are needed
  – frequently required for OLAP applications

• Parallelism
  – data fragmentation
  – query parallelization
    • inter-query
    • intra-query
  – join and group by lend themselves well to parallel execution
Index selection

• Indexing dimensions
  – attributes frequently involved in selection predicates
  – if domain cardinality is high, then B-tree index
  – if domain cardinality is low, then bitmap index

• Indices for join
  – indexing only foreign keys in the fact table is rarely appropriate
  – bitmapped join index is suggested (if available)

• Indices for group by
  – use materialized views
ETL Process

Elena Baralis
Politecnico di Torino
Extraction, Transformation and Loading (ETL)

• Prepares data to be loaded into the data warehouse
  – data extraction from (OLTP and external) sources
  – data cleaning
  – data transformation
  – data loading

• Eased by exploiting the staging area

• Performed
  – when the DW is first loaded
  – during periodical DW refresh
Extraction

• Data acquisition from sources
• Extraction methods
  – static: snapshot of operational data
    • performed during the first DW population
  – incremental: selection of updates that took place after last extraction
    • exploited for periodical DW refresh
    • immediate or deferred
• The selection of which data to extract is based on their quality
Extraction

• It depends on how operational data is collected
  – historical: all modifications are stored for a given time in the OLTP system
    • bank transactions, insurance data
    • operationally simple
  – partly historical: only a limited number of states is stored in the OLTP system
    • operationally complex
  – transient: the OLTP system only keeps the current data state
    • example: stock inventory
    • operationally complex
Incremental extraction

• **Application assisted**
  – data modifications are captured by ad hoc application functions
  – requires changing OLTP applications (or APIs for database access)
  – increases application load
  – hardly avoidable in legacy systems

• **Log based**
  – log data is accessed by means of appropriate APIs
  – log data format is usually proprietary
  – efficient, no interference with application load
Incremental extraction

• Trigger based
  – triggers capture interesting data modifications
  – does not require changing OLTP applications
  – increases application load

• Timestamp based
  – modified records are marked by the (last) modification timestamp
  – requires modifying the OLTP database schema (and applications)
  – deferred extraction, may lose intermediate states if data is transient
## Comparison of extraction techniques

<table>
<thead>
<tr>
<th>Management of transient or semi-periodic data</th>
<th>Static</th>
<th>Timestamps</th>
<th>Application assisted</th>
<th>Trigger</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Incomplete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support to file-based systems</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>Rare</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Implementation technique</th>
<th>Tools</th>
<th>Tools or internal developments</th>
<th>Internal developments</th>
<th>Tools</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs of enterprise specific development</td>
<td>None</td>
<td>Medium</td>
<td>High</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Use with legacy systems</td>
<td>Yes</td>
<td>Difficult</td>
<td>Difficult</td>
<td>Difficult</td>
<td>Yes</td>
</tr>
<tr>
<td>Changes to applications</td>
<td>None</td>
<td>Likely</td>
<td>Likely</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>DBMS-dependent procedures</td>
<td>Limited</td>
<td>Limited</td>
<td>Variabile</td>
<td>High</td>
<td>Limited</td>
</tr>
<tr>
<td>Impact on operational system performance</td>
<td>None</td>
<td>None</td>
<td>Medium</td>
<td>Medium</td>
<td>None</td>
</tr>
<tr>
<td>Complexity of extraction procedures</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

From Devlin, Data warehouse: from architecture to implementation, Addisono-Wesley, 1997
Incremental extraction

<table>
<thead>
<tr>
<th>Cod</th>
<th>Product</th>
<th>Customer</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Greco di tufo</td>
<td>Malavasi</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Barolo</td>
<td>Maio</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>Barbera</td>
<td>Lumini</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>Sangiovese</td>
<td>Cappelli</td>
<td>45</td>
</tr>
</tbody>
</table>

4/4/2010

<table>
<thead>
<tr>
<th>Cod</th>
<th>Product</th>
<th>Customer</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Greco di tufo</td>
<td>Malavasi</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Barolo</td>
<td>Maio</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>Trebbiano</td>
<td>Maltoni</td>
<td>150</td>
</tr>
</tbody>
</table>

6/4/2010

<table>
<thead>
<tr>
<th>Cod</th>
<th>Product</th>
<th>Customer</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Barbera</td>
<td>Lumini</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>Sangiovese</td>
<td>Cappelli</td>
<td>145</td>
</tr>
<tr>
<td>5</td>
<td>Vermentino</td>
<td>Maltoni</td>
<td>25</td>
</tr>
</tbody>
</table>

Incremental difference

<table>
<thead>
<tr>
<th>Cod</th>
<th>Product</th>
<th>Customer</th>
<th>Qty</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Barbera</td>
<td>Lumini</td>
<td>75</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>Sangiovese</td>
<td>Cappelli</td>
<td>145</td>
<td>U</td>
</tr>
<tr>
<td>5</td>
<td>Vermentino</td>
<td>Maltoni</td>
<td>25</td>
<td>I</td>
</tr>
<tr>
<td>6</td>
<td>Trebbiano</td>
<td>Maltoni</td>
<td>150</td>
<td>I</td>
</tr>
</tbody>
</table>

Data cleaning

• Techniques for improving data quality (correctness and consistency)
  – duplicate data
  – missing data
  – unexpected use of a field
  – impossible or wrong data values
  – inconsistency between logically connected data

• Problems due to
  – data entry errors
  – different field formats
  – evolving business practices
Data cleaning

- Each problem is solved by an ad hoc technique
  - data dictionary
    - appropriate for data entry errors or format errors
    - can be exploited only for data domains with limited cardinality
  - approximate fusion
    - appropriate for detecting duplicates/similar data correlations
      - approximate join
      - purge/merge problem
  - outlier identification, deviations from business rules
- Prevention is the best strategy
  - reliable and rigorous OLTP data entry procedures
Approximate join

- The join operation should be executed based on common fields, not representing the customer identifier

From Golfarelli, Rizzi, "Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006
Purge/Merge problem

• Duplicate tuples should be identified and removed
• A criterion is needed to evaluate record similarity

Data cleaning and transformation example

Normalization

Elena Baralis
C.so Duca degli Abruzzi 24
20129 Torino (I)

name: Elena
surname: Baralis
address: C.so Duca degli Abruzzi 24
ZIP: 20129
city: Torino
country: Italia

Standardization

name: Elena
surname: Baralis
address: Corso Duca degli Abruzzi 24
ZIP: 20129
city: Torino
country: Italia

Correction

name: Elena
surname: Baralis
address: Corso Duca degli Abruzzi 24
ZIP: 10129
city: Torino
country: Italia

Adapted from Golfarelli, Rizzi, “Data warehouse, teoria e pratica della progettazione”, McGraw Hill 2006
Transformation

- Data conversion from operational format to data warehouse format
  - requires data integration
- A uniform operational data representation (reconciled schema) is needed
- Two steps
  - from operational sources to reconciled data in the staging area
    - conversion and normalization
    - matching
    - (possibly) significant data selection
  - from reconciled data to the data warehouse
    - surrogate keys generation
    - aggregation computation
Data warehouse loading

• Update propagation to the data warehouse
• Update order that preserves data integrity
  1. dimensions
  2. fact tables
  3. materialized views and indices
• Limited time window to perform updates
• Transactional properties are needed
  – reliability
  – atomicity
Dimension table loading

Staging area

ODS

ID1
attr 1
attr 2
......

ID2
attr 3
attr 4
......

ID3
attr 5
attr 6
......

ID2
attr 1
attr 3
attr 5
attr 6

ODS

Data mart

Dimension Table

Identify updates

Map identifiers and sur. keys

New/updated tuples for DT

Look-up table

New/updated tuples for DT

Sur. Key S
attr 1
attr 3
attr 5
attr 6

Load new/updated tuples in DT

Fact table loading

Materialized view loading

Materialized view loading