Data Warehouse and Data Mining

Lecture No. 04

Data Marts & Storage Structure

Naeem Ahmed
Email: naeemmahoto@gmail.com

Department of Software Engineering
Mehran University of Engineering and Technology Jamshoro
Architecture of DW

Data Sources
- Operational System
- Flat files

Staging Area

Warehouse
- Metadata
- Summary Data
- Raw Data

Data Marts
- Purchasing
- Sales
- Inventory

Users
- Analysis
- Reporting
- Mining
A **data mart** is a special purpose subset of enterprise data for a particular function or application (It may contain detail or summary data or both).

**Data Mart types:**
- **Independent** — created directly from operational systems to a separate physical data store
- **Logical** — exists as a subset of existing data warehouse.
- **Dependent** — created from data warehouse to a separate physical data store
A dimensional model for a large data warehouse consists of between 10 and 25 similar-looking data marts. Each data mart will have 5 to 15 dimensional tables.
Dependent vs. Independent Data Marts
Data warehouse and Data mart

- Bill Inmon in 1998: “The single most important issue facing the IT manager this year is whether to build the data warehouse or the data mart first.”

- Fundamental issues:
  - Top-down or bottom-up approach?
  - Enterprise-wide or departmental?
  - Which first – data warehouse or data mart?
  - Dependent or independent data marts?
  - Build pilot or with a full-fledged implementation?
Top-down Approach

• The breaking down of a system to gain insight into its compositional sub-systems.

• The advantages of this approach are:
  – An enterprise view of data
  – Inherently architectured—not a union of disparate data marts
  – Single, central storage of data about the content
  – Centralized rules and control

• The disadvantages are:
  – Takes longer to build
  – High risks of failure
Bottom-up Approach

• The piecing together of systems (elements) to give rise to more complex systems.

• The advantages of this approach are:
  – Faster and easier implementation of manageable pieces
  – Favorable ROI (return on investment)
  – Less risk of failure
  – Inherently incremental

• The disadvantages are:
  – Spread redundant data in every data mart
  – Makes inconsistent data
Reasons behind Data marts

- Provide access to data most often for analysis
- Improve end-user response time due to reduction in volume of data to be accessed
- Simpler to build compared with establishing a corporate data warehouse
- Cost of implementation is normally less than that required to establish a data warehouse
Storage Structure

• Storage structure
  – After extraction from the operational data, in DW information is stored in databases
  – The databases are operated by a DBMS
  – Different database structures can be used for a DW:
    • Relational model (RDB) operated by a RDBMS
    • MultiDimensional model (MDB) operated by a MDBMS
Storage Structure

• RDB and MDB are complementary and do not have to exclude each other
  – In the **staging area** some RDBMS can be used, however it must be off-limits to user queries because of performance reasons
  – By default, normalized databases are excluded from the **presentation area**, which should be strictly multi-dimensionally (MDBMS)
Relational DB

- DB in relational model
  - A database is seen as a collection of **predicates** over a finite set of variables
  - The content of the DB is modeled as a set of **relations** in which all predicates are satisfied
Relational DB

- A **relation** is defined as a set of **tuples** that have the same attributes
  - It is usually described as a table
Multidimensional DB

- **Multidimensional DB (MDB)** are optimized for DW and OLAP applications
  - They are created using input from the **staging area**
  - Designed for efficient and convenient storage and retrieval of large volumes of data
  - Stored, viewed and analyzed from different perspectives called **dimensions**
Multidimensional DB

- Example: an automobile manufacturer wants to increase sale volumes
  - Evaluation requires to view historical sale volume figures from multiple dimensions
  - Sales volume by model, by color, by dealer, over time
  - A relational structure of the given evaluation would be:

<table>
<thead>
<tr>
<th>Model</th>
<th>Color</th>
<th>Sales volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini VAN</td>
<td>Blue</td>
<td>324</td>
</tr>
<tr>
<td>Mini VAN</td>
<td>Black</td>
<td>113</td>
</tr>
<tr>
<td>Mini VAN</td>
<td>Red</td>
<td>18</td>
</tr>
<tr>
<td>Sedan</td>
<td>Black</td>
<td>160</td>
</tr>
<tr>
<td>Sedan</td>
<td>Blue</td>
<td>115</td>
</tr>
<tr>
<td>Sedan</td>
<td>Red</td>
<td>6</td>
</tr>
<tr>
<td>Sports coupe</td>
<td>Red</td>
<td>16</td>
</tr>
<tr>
<td>Sports coupe</td>
<td>Black</td>
<td>16</td>
</tr>
<tr>
<td>Sports coupe</td>
<td>Blue</td>
<td>12</td>
</tr>
</tbody>
</table>
Multidimensional DB

- **Model**
  - Mini VAN: 113, 324, 18, 455
  - Sedan: 160, 115, 6, 281
  - Coupe: 16, 12, 16, 44

- **Color**
  - Black, Blue, Red, *

- Table with dimensions:
  - Rows: Mini VAN, Sedan, Coupe
  - Columns: Black, Blue, Red, *
  - Values: 289, 451, 40, 1560
Multidimensional DB

- The **complexity** grows quickly with the number of dimensions and the number of positions
  - Example: 3 dimensions with 10 values each and no indexes
  - If it is considered to view information in a RDB, it would result in a worst case of $10^3 = 1000$ records view
Multidimensional DB

• Now, if **performance** is considered
  – For responding to a query when car type = Sedan, color = Blue, and dealer = Berg
    • RDBMS has to search through 1000 records to find the right record
    • MDB has more knowledge about where data lies
    • The maximum of searches in the case of MDB is of 30 positions
    • Average case 18 vs. 501 positions
Multidimensional DB

• If the query is more relaxed
  – Total sales across all dealers for all colors when car type = sedan
    • RDBMS still has to go through the 1000 records
    • MDB, however, goes only through a slice of 10x10
Multidimensional DB

• Performance advantages
  – MDBs are an order of magnitude faster than RDBMSs
  – Performance benefits are more for queries that generate cross-tab views of data (the case of DW)

• Conclusion
  – The performance advantages offered by MDBs facilitates the development of interactive decision support applications like OLAP that can be impractical in a relational environment
Cube: A group of data cells arranged by the dimensions of the data.

Data Dimensionality: Cube

1st Qtr Sales of TV in Pakistan

Total annual sales of TV in Pakistan

Total annual sales of TV, PC & VCR in India
Data Dimensionality

Possible Views of Sale

• How many Products sold at Time to specific Customer(s)?

• How many Customers bought at specific Time the Product(s)?

• At which Time(s) the Customer(s) bought the specific Product(s)?
Multi-dimensional Data

- **Measures** - numerical data being tracked
- **Dimensions** - business parameters that define a transaction
- **Example**: Analyst may want to view *sales* data (measure) by *geography*, by *time*, and by *product* (dimensions)
- **Dimensional modeling** is a technique for structuring data around the business concepts
- **ER models** describe “entities” and “relationships”
- **Dimensional models** describe “measures” and “dimensions”
Multi-dimensional Model

“Sales by product line over the past six months”
“Sales by store between 1990 and 1995”
Multi-dimensional Model

- Every dimensional model (DM) is composed of one table with a composite primary key, called the **fact table**, and a set of smaller tables called **dimension tables**
- Forms ‘star-like’ structure, which is called a **star schema** or **star join**
- Dimensions are organized into hierarchies
  - E.g., Time dimension: days → weeks → quarters
  - E.g., Product dimension: product → product line → brand
- Dimensions have attributes
  - e.g., owner city and county of store
RDB vs. MDB

• Any database manipulation is possible with both technologies

• MDBs however offer some advantages in the context of DW:
  – Ease of data presentation
  – Ease of maintenance
  – Performance
RDB vs. MDB

• Ease of data presentation
  – Data views are natural output of the MDBs
  – Obtaining the same views in RDB requires a complex query
    • Example with Walmart and Sybase:
      
      ```sql
      select sum(sales.quantity_sold) from sales, products, product_categories, manufacturers, stores, cities where manufacturer_name = 'Colgate' and product_category_name = 'toothpaste' and cities.population < 40 000 and trunc(sales.date_time_of_sale) = trunc(sysdate-1) and sales.product_id = products.product_id and sales.store_id = stores.store_id and products.product_category_id = product_categories.product_category_id and products.manufacturer_id = manufacturers.manufacturer_id and stores.city_id = cities.city_id
      ```
RDB vs. MDB

• Ease of data presentation
  – Top k queries cannot be expressed well in SQL
    • Find the five cheapest hotels in Frankfurt
      – `SELECT * FROM hotels h WHERE h.city = Frankfurt AND 5 > (SELECT count(*) FROM hotels h1 WHERE h1.city = Frankfurt AND h1.price < h.price);`
    • Some RDBMS extended the functionality of SQL with STOP AFTER functionality
      – `SELECT * FROM hotels WHERE city = Frankfurt Order By price STOP AFTER 5;`
RDB vs. MDB

• Ease of **maintenance**
  – No additional overhead to translate user queries into requests for data
    • Data is stored as it is viewed
  – RDBs use **indexes** and **sophisticated joins** which require **significant maintenance** and storage to provide same intuitiveness
RDB vs. MDB

• Performance
  – Performance of MDBs can be matched by RDBs through database tuning
  – Not possible to tune the database for all possible ad-hoc queries
  – Aggregate navigators are helping RDBs to catch up with MDBs as far as aggregation queries are concerned
Multidimensional DB

• When MDBs are in-appropriate?
  – If the dataset types are not highly related, using a MDB results in a sparse representation
Multidimensional DB

- When MDBs are appropriate?
  - In the case of **highly interrelated** dataset types MDBs are recommended for greatest ease of access and analysis
  - Examples of applications
    - Financial Analysis and Reporting
    - Budgeting
    - Promotion Tracking
    - Quality Assurance and Quality Control
    - Product Profitability
DW Architectures

• Popular DW architectures
  – Generic Two-Tier Architecture
  – Independent Data Mart
  – Dependent Data Mart and Operational Data Store
  – Logical Data Mart and Active Warehouse
  – Three-Tier Architecture

• Other
  – One-Tier Architecture
  – N-Tier Architecture
  – Web-based Architecture
Layered Architectures

- Generic Two-Tier Architecture
  - Data is not completely current in the DW
  - Periodic extraction
Layered Architectures

• Data **analysis** comes in two flavors
  – Depending on the execution place of the analysis
  • Thin Client
    – Analytics are executed on the server
    – Client just displays
    – This architecture fits well for Internet/Intranet DW access
Layered Architectures

• Fat Client
  – The server just delivers the data
  – Analytics are executed on the client
  – Communication between client and server must be able to sustain large data transfers
Layered Architectures

• **Independent Data Mart**
  - Mini warehouses – limited in scope
  - Separate ETL for each independent Data Mart
  - High Data Marts access complexity
Layered Architectures

• **Dependent Data Mart** and Operational Data Store
  – Single ETL for the DW
  – Data Marts are loaded from the DW
  – More simple data access than in the previous case
Layered Architectures

- **Logical Data Mart** and Active Warehouse
  - The ETL is near real-time
  - Data Marts are *not* separate databases, but logical views of the DW
# DW vs. Data Marts

<table>
<thead>
<tr>
<th>Subjects</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DW</strong></td>
<td><strong>Data Marts</strong></td>
</tr>
<tr>
<td>Multiple subjects</td>
<td>One central subject</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sources</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DW</strong></td>
<td><strong>Data Marts</strong></td>
</tr>
<tr>
<td>Many internal and external sources</td>
<td>Few internal and external sources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DW</strong></td>
<td><strong>Data Marts</strong></td>
</tr>
<tr>
<td>Flexible</td>
<td>Restrictive</td>
</tr>
<tr>
<td>Data-oriented</td>
<td>Project oriented</td>
</tr>
<tr>
<td>Long life</td>
<td>Short life</td>
</tr>
<tr>
<td>Large</td>
<td>Start small, becomes large</td>
</tr>
<tr>
<td>Single complex structure</td>
<td>Multiple, semi-complex structure, together complex</td>
</tr>
</tbody>
</table>

## Scope

<table>
<thead>
<tr>
<th>DW</th>
<th>Data Marts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application independent</td>
<td>Specific DSS application</td>
</tr>
<tr>
<td>Centralized,</td>
<td>Decentralized by user area</td>
</tr>
<tr>
<td>Planned</td>
<td>Organic, possibly not planned</td>
</tr>
</tbody>
</table>

## Data

<table>
<thead>
<tr>
<th>DW</th>
<th>Data Marts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historical, detailed, summarized</td>
<td>Some history, detailed, summarized</td>
</tr>
<tr>
<td>Lightly denormalized</td>
<td>Highly denormalized</td>
</tr>
</tbody>
</table>
Layered Architectures

• **Generic Three-Tier Architecture**
  - Derived data
    • Data that had been selected, formatted, and aggregated for DSS support
  - Reconciled data
    • Detailed, current data intended to be the single, authoritative source for all decision support
Layered Architectures

• One-Tier Architecture
  – Theoretically possible
  – Might be interesting for mobile applications

• N-Tier Architecture
  – Higher tier architecture is also possible
    • But the complexity grows with the number of tier-interfaces

• Web-based Architecture
  – Advantages:
    • Usage of existing software, reduction of costs, platform independence
  – Disadvantages:
    • Security issues: data encryption/user access and identification
Distributed DW

• In most cases the economics and technology greatly favor a single **centralized DW**
• But in some cases, distributed DW make sense
• Types of **distributed DW**
  – **Geographically** distributed
    • Local DW/global DW
  – **Technologically** distributed DW
    • Logically one DW, physically more DW
  – **Independently evolving** distributed DW
    • Uncontrolled growth
Distributed DW

- **Geographically** distributed
  - In the case of corporations spread around the world
    - Information is needed both *locally* and *globally*
  - A distributed DW makes sense
    - When much processing occurs at the local level
    - Even though local branches report to the same balance sheet, the local organizations are their own companies
Distributed DW

[Diagram showing a distributed data warehouse (DW) across Europe, Asia, and the USA, with local DW sites and operational processing.]
Distributed DW

• Technologically distributed DW
  – Placing the DW on the distributed technology of a vendor
  – Advantages
    • The entry cost is cheap – large centralized hardware is expensive
    • No theoretical limit to how much data can be placed in the DW – one can add new servers to the network
  – As the DW starts to expand network data communication starts playing an important role
    • Example: Let’s simplify and consider one has 4 nodes holding each data regarding the last 4 years
    • Now let’s consider one has a query which needs to access the data from the last 4 years: such a query arises the issue of transporting large amount of data between processors
Distributed DW

• **Independently evolving distributed DW**
  
  – In practice there are many cases in which independent DW are developed concurrently and uncontrolled in the same organization
    
    • The first step many corporations make is to build a DW for financial or marketing
    
    • Once it is successfully set up, other parts of the organization follow independently the process resulting in the coexistence of more independent DW in the same organization
    
    • This problem will be addressed later
Summary

• Storage structures:
  – MDB are more suitable for DW

• Architectures:
  – One-Tier Architecture: interesting for mobile applications
  – N-Tier Architecture: complexity grows N
  – Web-based Architecture: reduction of costs
    • Security issues: data encryption/user access and identification

• DW are usually distributed geographically and technologically