1 Introduction

2 Data warehousing concepts

3 Schemas for multidimensional data models

4 OLAP server architectures

5 Reading
Data warehouses generalize and consolidate data in multidimensional space.

Construction of data warehouses involves data cleaning, data integration, and data transformation.

Data warehouses provide online analytical processing (OLAP) tools for interactive analysis of multidimensional data of varied granualities, which facilates effective data mining.

Data mining functions such as clustering, classification, and associative rule mining can be integrated with OLAP functions to enhance interactive data mining.

As a conclusion, data warehousing form an essential step in knowledge discovery process.
What is a data warehouse?

A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management decision making process. (William H. Inmon)

The following keywords distinguish data warehouse from other data repository systems such as relational database systems.

- **Subject-oriented** A data warehouse is organized around major subjects such as customer, supplier, product, and sales.
- **Integrated** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- **Time-variant** Data are stored to provide information from an historic perspective (e.g., the past 510 years).
- **Nonvolatile** A data warehouse does not require transaction processing, recovery, and concurrency control mechanisms. It usually requires only two operations in data accessing: initial loading of data and access of data.
### Differences between operational databases and data warehouses

<table>
<thead>
<tr>
<th>Feature</th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristic</td>
<td>operational processing</td>
<td>informational processing</td>
</tr>
<tr>
<td>Orientation</td>
<td>transaction</td>
<td>analysis</td>
</tr>
<tr>
<td>User</td>
<td>clerk, DBA, database professional</td>
<td>knowledge worker (e.g., manager, executive, analyst)</td>
</tr>
<tr>
<td>Function</td>
<td>day-to-day operations</td>
<td>long-term informational requirements decision support</td>
</tr>
<tr>
<td>DB design</td>
<td>ER-based, application-oriented</td>
<td>star/snowflake, subject-oriented</td>
</tr>
<tr>
<td>Data</td>
<td>current, guaranteed up-to-date</td>
<td>historic, accuracy maintained over time</td>
</tr>
<tr>
<td>Summarization</td>
<td>primitive, highly detailed</td>
<td>summarized, consolidated</td>
</tr>
<tr>
<td>View</td>
<td>detailed, flat relational</td>
<td>summarized, multidimensional</td>
</tr>
<tr>
<td>Unit of work</td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td>Access</td>
<td>read/write</td>
<td>mostly read</td>
</tr>
<tr>
<td>Focus</td>
<td>data in</td>
<td>information out</td>
</tr>
<tr>
<td>Operations</td>
<td>index/hash on primary key</td>
<td>lots of scans</td>
</tr>
<tr>
<td>Number of records accessed</td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td>Number of users</td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td>DB size</td>
<td>GB to high-order GB</td>
<td>≥ TB</td>
</tr>
<tr>
<td>Priority</td>
<td>high performance, high availability</td>
<td>high flexibility, end-user autonomy</td>
</tr>
<tr>
<td>Metric</td>
<td>transaction throughput</td>
<td>query throughput, response time</td>
</tr>
</tbody>
</table>

Note: Table is partially based on Chaudhuri and Dayal [CD97].

Support requires historic data, whereas operational databases do not typically maintain historic data. In this context, the data in operational databases, though abundant, are usually far from complete for decision making. Decision support requires consolidation (e.g., aggregation and summarization) of data from heterogeneous sources, resulting in high-quality, clean, integrated data. In contrast, operational databases contain only detailed raw data, such as transactions, which need to be consolidated before analysis. Because the two systems provide quite different functionalities and require different kinds of data, it is presently necessary to maintain separate databases. However, many vendors of operational relational database management systems are beginning to optimize such systems to support OLAP queries. As this trend continues, the separation between OLTP and OLAP systems is expected to decrease.

#### 4.1.4 Data Warehousing: A Multitiered Architecture

Data warehouses often adopt a three-tier architecture, as presented in Figure 4.1.
Data warehouses often adopt a three-tier architecture, as presented below.

1. The bottom tier is a warehouse database server that is almost always a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse (see Section 4.1.6). The data are extracted using application program interfaces known as gateways. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object...
From the architecture point of view, there are three data warehouse models:

**Enterprise warehouse**  An enterprise warehouse collects all of the information about subjects spanning the entire organization.

**Data mart**  data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects.

**Virtual warehouse**  A virtual warehouse is a set of views over operational databases.

What are the pros and cons of the top-down and bottom-up approaches to data warehouse development?

- The top-down development of an enterprise warehouse serves as a systematic solution and minimizes integration problems. However, it is expensive, takes a long time to develop, and lacks flexibility due to the difficulty in achieving consistency and consensus for a common data model for the entire organization.

- The bottom-up approach to the design, development, and deployment of independent data marts provides flexibility, low cost, and rapid return of investment. It, however, can lead to problems when integrating various disparate data marts into a consistent enterprise data warehouse.
Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and utilities include the following functions:

- **Data extraction** This typically gathers data from multiple, heterogeneous, and external sources.
- **Data cleaning** This detects errors in the data and rectifies them when possible.
- **Data transformation** This converts data from legacy or host format to warehouse format.
- **Load** This sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
- **Refresh** This propagates the updates from the data sources to the warehouse.

Besides the above functions, data warehouse systems usually provide a good set of data warehouse management tools.
Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouse objects.

A metadata repository should contain the following:

- **A description of the data warehouse structure** including the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.

- **Operational metadata** such as history of migrated data and the sequence of transformations applied to it and monitoring information (warehouse usage statistics, error reports, and audit trails).

- **The algorithms used for summarization** including measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.

- **Mapping from the operational environment to the data warehouse** including source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and user authorization and access control.

- **Data related to system performance** including indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.

- **Business metadata** including business terms and definitions, data ownership information, and charging policies.
Data warehouse modeling

- Data warehouses and OLAP tools are based on a multidimensional data model.
- This model views data in the form of a data cube.
- What is a data cube?
  - A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.
  - Dimensions are the perspectives or entities with respect to which an organization wants to keep records.
  - Each dimension may have a table associated with it, called a dimension table, which further describes the dimension.
  - Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.
  - A multidimensional data model is typically organized around a central theme represented by a fact table. Facts are numeric measures.
  - Fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.
Data cube (example)

- Relational schema for a relational database
  
<table>
<thead>
<tr>
<th>table</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td>cust_ID, name, address, age, occupation, annual_income, credit_information,</td>
</tr>
<tr>
<td></td>
<td>category, ...</td>
</tr>
<tr>
<td>item</td>
<td>item_ID, brand, category, type, price, place_made, supplier, cost, ...</td>
</tr>
<tr>
<td>employee</td>
<td>empl_ID, name, category, group, salary, commission, ...</td>
</tr>
<tr>
<td>branch</td>
<td>branch_ID, name, address, ...</td>
</tr>
<tr>
<td>purchases</td>
<td>trans_ID, cust_ID, empl_ID, date, time, method_paid, amount</td>
</tr>
<tr>
<td>items_sold</td>
<td>trans_ID, item_ID, qty</td>
</tr>
<tr>
<td>works_at</td>
<td>empl_ID, branch_ID</td>
</tr>
</tbody>
</table>

- 2-D view of sales data

<table>
<thead>
<tr>
<th>time (quarter)</th>
<th>location = “Vancouver”</th>
<th>item (type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>home entertainment</td>
<td>computer</td>
</tr>
<tr>
<td>Q1</td>
<td>605</td>
<td>825</td>
</tr>
<tr>
<td>Q2</td>
<td>680</td>
<td>952</td>
</tr>
<tr>
<td>Q3</td>
<td>812</td>
<td>1023</td>
</tr>
<tr>
<td>Q4</td>
<td>927</td>
<td>1038</td>
</tr>
</tbody>
</table>
Data cube (example)

- 3-D view of sales data

<table>
<thead>
<tr>
<th>location = “Chicago”</th>
<th>location = “New York”</th>
<th>location = “Toronto”</th>
<th>location = “Vancouver”</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>home</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>ent. comp. phone sec.</td>
<td>ent. comp. phone sec.</td>
<td>ent. comp. phone sec.</td>
</tr>
<tr>
<td>Q1</td>
<td>854 882 89 623</td>
<td>1087 968 38 872</td>
<td>818 746 43 591</td>
</tr>
<tr>
<td>Q2</td>
<td>943 890 64 698</td>
<td>1130 1024 41 925</td>
<td>894 769 52 682</td>
</tr>
<tr>
<td>Q3</td>
<td>1032 924 59 789</td>
<td>1034 1048 45 1002</td>
<td>940 795 58 728</td>
</tr>
<tr>
<td>Q4</td>
<td>1129 992 63 870</td>
<td>1142 1091 54 984</td>
<td>978 864 59 784</td>
</tr>
</tbody>
</table>

Note: The sales are from branches located in the city of Vancouver. The measure displayed is dollars sold (in thousands).
A 3-D data cube representation of the data (previous slide)

![Diagram of a 3-D data cube with dimensions for time, location, and item types.](image)

- Time (quarters): Q1, Q2, Q3, Q4
- Location (cities): Chicago, New York, Toronto, Vancouver
- Item types: Home entertainment, Computer, Phone, Security
- Measures: Dollars sold (in thousands)

The measure displayed is dollars sold (in thousands).

Figure 4.3: A 3-D data cube representation of the data in Table 4.3, according to time, item, and location. The measure displayed is dollars sold (in thousands).

Figure 4.4: A 4-D data cube representation of sales data, according to time, item, location, and supplier. The measure displayed is dollars sold (in thousands). For improved readability, only some of the cube values are shown.

The 0-D cuboid, which holds the highest level of summarization, is called the apex cuboid. In our example, this is the total sales, or dollars sold, summarized over all four dimensions. The apex cuboid is typically denoted by all.
A 4-D data cube representation of the data
In the data warehousing, a data cube like those shown in (previous slides) is often referred to as a **cuboid**.

- Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions.
- The result would form a **lattice of cuboids**, each showing the data at a different level of summarization, or group-by.
- The **lattice of cuboids** is then referred to as a **data cube**.
- The cuboid that holds the lowest level of summarization is called the **base cuboid**.
- The 0-D cuboid, which holds the highest level of summarization, is called the **apex cuboid**.
4.2 Data Warehouse Modeling: Data Cube and OLAP

Lattice of cuboids

- Lattice of cuboids

![Diagram of Lattice of cuboids]

- 0-D (apex) cuboid
- 1-D cuboids
- 2-D cuboids
- 3-D cuboids
- 4-D (base) cuboid
The entity-relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities and the relationships between them.

The entity-relationship data model is appropriate for online transaction processing.

A data warehouse, however, requires a concise, subject-oriented schema that facilitates online data analysis.

The most popular data model for a data warehouse is a multidimensional model, which can exist in the form of a

- Star schema
- Snowflake schema
- Galaxy schema
The most common modeling paradigm is the star schema, in which the data warehouse contains:

1. A large central table (fact table) containing the bulk of the data, with no redundancy.
2. A set of smaller attendant tables (dimension tables), one for each dimension.

The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.
In star schema, each dimension is represented by only one table, and each table contains a set of attributes.

This constraint may introduce some redundancy.

The attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).

The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables.

The resulting schema graph forms a shape similar to a snowflake.

The major difference between snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form to reduce redundancies.

Tables in snowflake schema are easy to maintain and save storage space.

This space savings is negligible in comparison to the typical magnitude of the fact table.

The snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query.

The snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.
Snowflake schema (cont.)

An example of snowflake scheme

- **time**
  - Dimension table
  - `time_key`
  - `day`
  - `day_of_week`
  - `month`
  - `quarter`
  - `year`

- **sales**
  - Fact table
  - `time_key`
  - `item_key`
  - `branch_key`
  - `location_key`
  - `dollars_sold`
  - `units_sold`

- **item**
  - Dimension table
  - `item_key`
  - `item_name`
  - `brand`
  - `type`
  - `supplier_key`

- **supplier**
  - Dimension table
  - `supplier_key`
  - `supplier_type`

- **branch**
  - Dimension table
  - `branch_key`
  - `branch_name`
  - `branch_type`

- **location**
  - Dimension table
  - `location_key`
  - `street`
  - `city_key`

- **city**
  - Dimension table
  - `city_key`
  - `city`
  - `province_or_state`
  - `country`
Galaxy schema

- Sophisticated applications may require **multiple fact tables** to share dimension tables.
- This kind of schema can be viewed as a collection of stars, and hence is called a **galaxy schema** or a **fact constellation**.
A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.

- **country**
- **province_or_state**
- **city**
- **street**
- **year**
- **quarter**
- **month**
- **week**
- **day**
Concept hierarchies may also be defined by discretizing or grouping (following figure) values for a given dimension or attribute.

There may be more than one concept hierarchy for a given attribute or dimension, based on different user viewpoints.

Concept hierarchies may be provided manually by system users, domain experts, or knowledge engineers, or may be automatically generated based on statistical analysis of the data distribution.
A data cube measure is a numeric function that can be evaluated at each point in the data cube space.

A measure value is computed for a given point by aggregating the data corresponding to the respective dimension value pairs defining the given point.

Measures can be organized into three categories based on the kind of aggregate functions.

1. **Distributive**: An aggregate function is distributive if it can be computed in a distributed manner. For example, count(), min(), and max() are distributive aggregate functions. (appropriate for large data cube)

2. **Algebraic**: An aggregate function is algebraic if it can be computed by an algebraic function with M arguments (where M is a bounded positive integer), each of which is obtained by applying a distributive aggregate function. For example, avg() can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions. (appropriate for large data cube)

3. **Holistic**: An aggregate function is holistic if there is no constant bound on the storage size needed to describe a subaggregate. That is, there does not exist an algebraic function with M arguments (where M is a constant) that characterizes the computation. For example, median(), mode(), and rank(). A measure is holistic if it is obtained by applying a holistic aggregate function. (There are some techniques to approximate the computation of some holistic measures)
In the multidimensional model, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. This organization provides users with the flexibility to view data from different views. A number of OLAP operations exist to materialize these different views.

- **Roll-up** The roll-up operation performs aggregation on a data cube.
A number of OLAP operations exist to materialize these different views.

- **Drill-down**  
  Drill-down is the reverse of roll-up.
A number of **OLAP operations** exist to materialize these different views.

- **Slice** The slice operation performs a selection on one dimension of the given cube, resulting in a subcube.
A number of **OLAP operations** exist to materialize these different views.

- **Dice** The dice operation defines a subcube by performing a selection on two or more dimensions.
OLAP operations (cont.)

- A number of **OLAP operations** exist to materialize these different views.
  - **Pivot(rotate)** Pivot is a visualization operation that rotates the data axes in view to provide an alternative data presentation.
Implementations of a warehouse server for OLAP processing include the following:

1. **Relational OLAP (ROLAP) servers** These are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces.

2. **Multidimensional OLAP (MOLAP) servers** These servers support multidimensional data views through array-based multidimensional storage engines. They map multi-dimensional views directly to data cube array structures.

3. **Hybrid OLAP (HOLAP) servers** The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP.

4. **Hybrid OLAP (HOLAP) servers** Some database system vendors implement specialized SQL servers that provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.
Read chapter 4 of the following book
J. Han, M. Kamber, and Jian Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2012.