Unit - 1: Data Warehousing and Online Analytical Processing

CONTENTS

Objectives

After studying this unit, you will be able to:

- Know data warehouse concept.
- Outlines the difference between the operational database system and data warehouse.
- Acquire how data cubes model n-dimensional data
- Examine the way to index OLAP information by bitmap indexing and join indexing.
- Explores different methods to compute data cubes efficiently.
- Study the design and usage of data warehousing for information processing, analytical processing, and data mining.

Introduction

Data warehouses simplify and combine data in multidimensional space. The building of data warehouses includes data cleaning, data integration, and data transformation, and can be seen as an significant preprocessing step for data mining. Furthermore, data warehouses offer online analytical processing (OLAP) tools for the collaborative analysis of multidimensional data of diverse granularities, which simplifies effective data generalization and data mining. Numerous other data mining tasks, such as association, classification, prediction, and clustering, can be combined with OLAP operations to improve interactive mining of knowledge at several levels of abstraction. Henceforth, the data warehouse has convert an progressively important stage for data analysis and OLAP and will deliver an effective platform for data mining. So, data warehousing and OLAP form an important step in the knowledge discovery process. This chapter focus on the overview of data warehouse and OLAP technology.

1.1 What Is a Data Warehouse?

Data warehouses have been well-defined in many ways,making it hard to articulate a demanding definition. Lightly speaking, a data warehouse refers to a data repository that is maintained separately from an organization's operational databases. Data warehouse systems allow for the integration of a variety of application systems. They support information processing by providing a solid platform of consolidated historic data for analysis.

According to William H. Inmon, a leading architect in the construction of data warehouse systems, "A data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process". This short but comprehensive definition presents the major features of a data warehouse. The four keywords—subject-oriented, integrated, time-variant, and non-volatile—differentiate data warehouses from other data source systems, such as relational database systems, transaction processing systems, and file systems.

Let's take a closer look at each of these key features.

- **Subject-oriented**: A data warehouse is systematized around major subjects such as customer, supplier, product, and sales. Rather than focussed on the day-to-day operations and transaction processing of an organization, a data warehouse emphases on the modeling and analysis of data for decision-makers. Henceforth, data warehouses usually provide a simple and succinct view of particular subject issues by excluding data that are not useful in the decision support process.
- **Integrated**: A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records. Data cleaning and data integration techniques are applied to ensure consistency in naming conventions, encoding structures, attribute measures, and so on.
- **Time-variant**: Data is stored to provide information from a historic perspective (e.g., the past 5–10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, a time element.
- **Non-volatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. Due to this separation, 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.*

Example: A typical data warehouse is organized around major subjects, such as customer, vendor, product, and sales rather than concentrating on the day-to-day operations and transaction processing of an organization.

Features of a Data Warehouse

- ***It is separate from the Operational Database.***
- ***Integrates data from heterogeneous systems.***
- ***Stores HUGE amount of data, more historical than current data.***
- ***Does not require data to be highly accurate.***
- ***Queries are generally complex.***

- ***The goal is to execute statistical queries and provide results that can influence decision-making in favor of the Enterprise.***
- ***These systems are thus called Online Analytical Processing Systems (OLAP).***

1.2 The need for a Separate Data Warehouse

Because operational databases store huge amounts of data, you may wonder, "Why not perform online analytical processing directly on such databases instead of spending additional time and resources to construct a separate data warehouse?" A major reason for such a separation is to help promote the high performance of both systems. An operational database is designed and tuned from known tasks and workloads like indexing and hashing using primary keys, searching for particular records, and optimizing "canned" queries. On the other hand, data warehouse queries are often complex. They involve the computation of large data groups at summarized levels and may require the use of special data organization, access, and implementation methods based on multidimensional views. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.

Moreover, an operational database supports the concurrent processing of multiple transactions. Concurrency control and recovery mechanisms (e.g., locking and logging) are required to ensure the consistency and robustness of transactions. An OLAP query often needs read-only access to data records for summarization and aggregation. Concurrency control and recovery mechanisms, if applied for such OLAP operations, may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

Finally, the separation of operational databases from data warehouses is based on the different structures, contents, and uses of the data in these two systems. Decision 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.

1.3 Data Warehouse Models: Enterprise Warehouse, Data Mart, and Virtual Warehouse

From the architecture point of view, there are three data warehouse models: *the enterprise warehouse, the data mart,* and the *virtual warehouse*.

Enterprise warehouse: An enterprise warehouse collects all of the information about subjects spanning the entire organization. It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope. It typically contains detailed data as well as summarized data and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond. An enterprise data warehouse may be implemented on traditional mainframes, computer super servers, or parallel architecture platforms. It requires extensive business modeling and may take years to design and build.

Datamart: A 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. For example, a marketing data mart may confine its subjects to a customer, item, and sales. The data contained in data marts tend to be summarized.

Depending on the source of data, data marts can be categorized into the following two classes:

1. Independent data marts are sourced from data captured from one or more operational systems or external information providers, or data generated locally within a particular department or geographic area.

2. Dependent data marts are sourced directly from enterprise data warehouses.

Virtual warehouse: A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers.

A recommended method for the development of data warehouse systems is to implement the warehouse incrementally and evolutionarily, as shown in Figure 1.

First, a high-level corporate data model is defined within a reasonably short period (such as one or two months) that provides a corporate-wide, consistent, integrated view of data among different subjects and potential usages. This high-level model, although it will need to be refined in the further development of enterprise data warehouses and departmental data marts, will greatly reduce future integration problems. Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set noted before. Third, distributed data marts can be constructed to integrate different data marts via hub servers. Finally, a multitier data warehouse is constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.

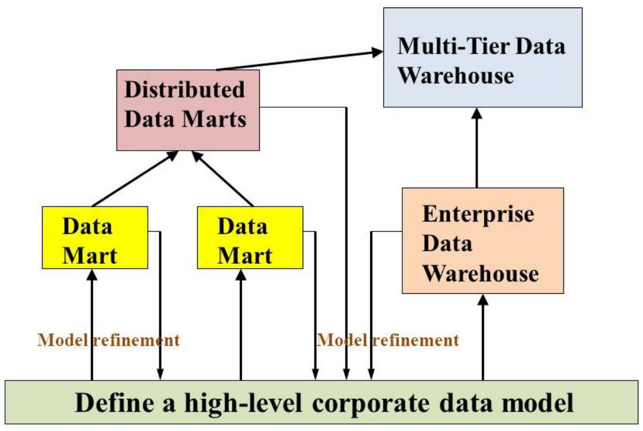

Multi-Tier Data Warehouse
Distributed Data Marts
Data Mart
Data Mart
Enterprise Data Warehouse
Model refinement
Model refinement
Define a high-level corporate data model

Figure 1 A recommended approach for data warehouse development

1.4 Differences between Operational Database Systems and Data Warehouses.

Because most people are familiar with commercial relational database systems, it is easy to understand what a data warehouse is by comparing these two kinds of systems.

The major task of online operational database systems is to perform online transaction and query processing. These systems are called **online transaction processing (OLTP) systems**. They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting. Data warehouse systems, on the other hand, serv ice users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats to accommodate the diverse needs of different users. These systems are known as **online analytical processing (OLAP) systems.**

The major distinguishing features between OLTP and OLAP are summarised in Table 1.

Table 1 Difference between OLTP and OLAP System

Feature	OLTP System	OLAP System
Characteristic	Operational Processing	Informational Processing
Users	Clerks, clients, and information technology professionals.	Knowledge workers, including managers, executives, and analysts.
System orientation	Customer-oriented and used for transaction and query processing Day to day operations	Market-oriented and used for data analysis long-term informational requirements, decision support.
Data contents	Manages current data that typically, are too detailed to be easily used for decision making.	Manages large amounts of historical data, provides facilities for summarization and

		aggregation, and stores and manages information at different levels of granularity.
Database design	Adopts an entity-relationship (ER) data model and an application-oriented database design	Adopts either a star or snowflake model and a subject-oriented database design.
View	Focuses mainly on the current data within an enterprise or department, without referring to historical data or data in different organizations.	In contrast, an OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization. OLAP systems also deal with information that originates from different organizations, integrating information from many data stores.
Volume of data	Not very large	Because of their huge volume, OLAP data are stored on multiple storage media.
Access patterns	Consists mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms.	Accesses to OLAP systems are mostly read-only operations (since most data warehouses store historical rather than up-to-date information), although many could be complex queries.
Access mode	Read/write	Mostly write
Focus	Data in	Information out
Operations	Index/hash on a primary key	Lots of scans
Number of records accessed	Tens	Continue…. Millions
Number of users	Thousands	Hundreds
DB size	100 MB to GB	100 GB to TB
Priority	High performance, high availability	High flexibility, end-user autonomy
Metric	Transaction throughput	Query response time

Exactly what the difference between operational database and data warehouse? Explain with the suitable of suitable example.

1.5 Data Warehouse Modeling: Data Cube

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. Such a data model is appropriate for on-line transaction processing. The data warehouse requires a concise, subject-oriented schema that facilitates OLAP. Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a data cube. A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts. In general terms, dimensions are the perspectives or entities concerning 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. For example, a dimension table for an item may contain the attributes item name, brand, and type. 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, such as sales. This theme is represented by a fact table. Facts are numeric measures. Think of them as the quantities by which we want to analyze relationships between dimensions. Examples of facts for a sales data warehouse include dollars sold (sales amount in dollars), units sold (number of units sold), and the amount budgeted. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables. You will soon get a clearer picture of how this works when we look at multidimensional schemas.

Table 2 2-D View of Sales Data for AllElectronics According to time and item

	location = "Vancouver"			
	item *(type)*			
time *(quarter)*	*home entertainment*	*computer*	*phone*	*security*
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Note: The sales are from branches located in the city of Vancouver. The measure displayed is dollars sold (in thousands).

Table 3 3-D View of Sales Data for AllElectronics According to time, item, and location

	location = "Chicago"				*location* = "New York"				*location* = "Toronto"				*location* = "Vancouver"			
	item				**item**				**item**				**item**			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Table 2 and Table 3 show the data at different degrees of summarization. In the data warehousing research literature, a data cube like those shown in Figure 2 and Figure 3 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**. Figure 4 shows a lattice of cuboids forming a data cube for the dimensions time, item, location, and supplier.

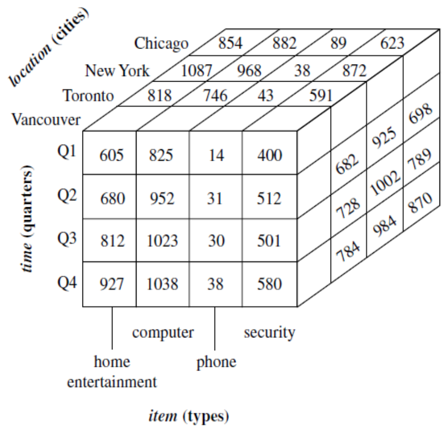


Figure 2 A 3-D data cube representation of the data in Table 3 according to time, item, and location.

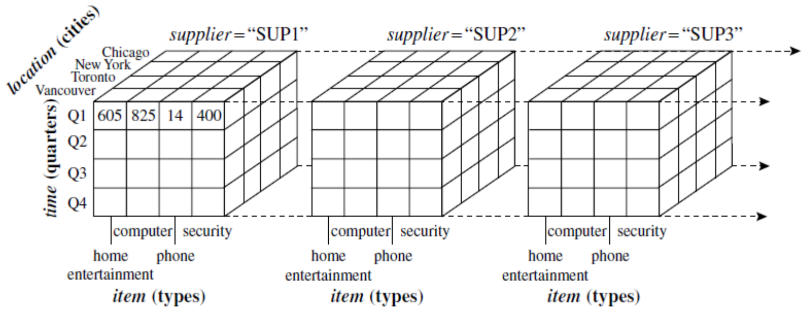


Figure 3 A 4-D data cube representation of sales data, according to time, item, location, and supplier.

The cuboid that holds the lowest level of summarization is called the base cuboid. For example, the 4-D cuboid in Figure 3 is the base cuboid for the given time, item, location, and supplier dimensions. Figure 2 is a 3-D (non-base) cuboid for time, item, and location summarized for all suppliers. 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.

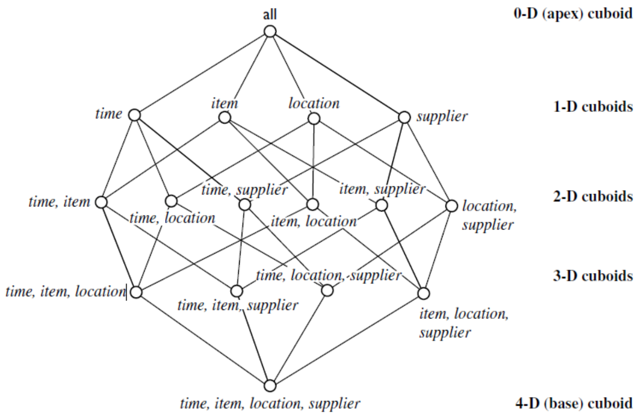


Figure 4 Lattice of cuboids, making up a 4-D data cube for time, item, location, and supplier.

Example: "Date" can be grouped into "day", "month", "quarter", "year" or "week", which forms a lattice structure.

1.6 Conceptual Modeling of Data Warehouse

The most popular data model for a data warehouse is a multidimensional model, which can exist in the form of a **star schema, a snowflake schema**, or a **fact constellation schema**. Let's look at each of these.

Star 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, and (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.

A star schema for AllElectronics sales is shown in Figure 5. Sales are considered along four dimensions: time, item, branch, and location. The schema contains a central fact table for sales that contain keys to each of the four dimensions, along with two measures: dollars sold and units sold. To minimize the size of the fact table, dimension identifiers (e.g., time key and item key) are system-generated identifiers. Notice that in the star schema, each dimension is represented by only one table, and each table contains a set of attributes.

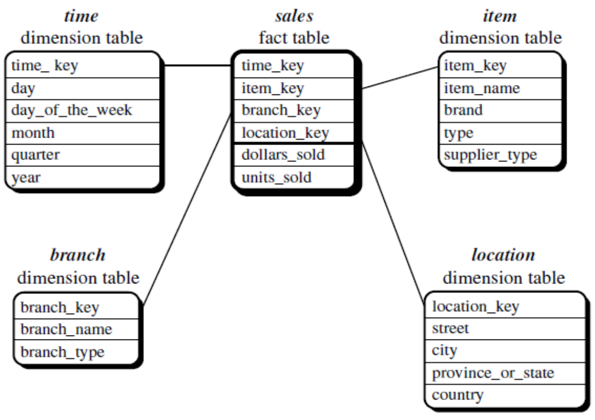


Figure 5 Star schema of a sales data warehouse.

Example: Let, an organization sells products throughout the world. The main four major dimensions are time, location, time, and branch.

Snowflake schema: 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 the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in the normalized form to reduce redundancies. Such a table is easy to maintain and saves storage space. However, this space savings is negligible in comparison to the typical magnitude of the fact table. Furthermore, the snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

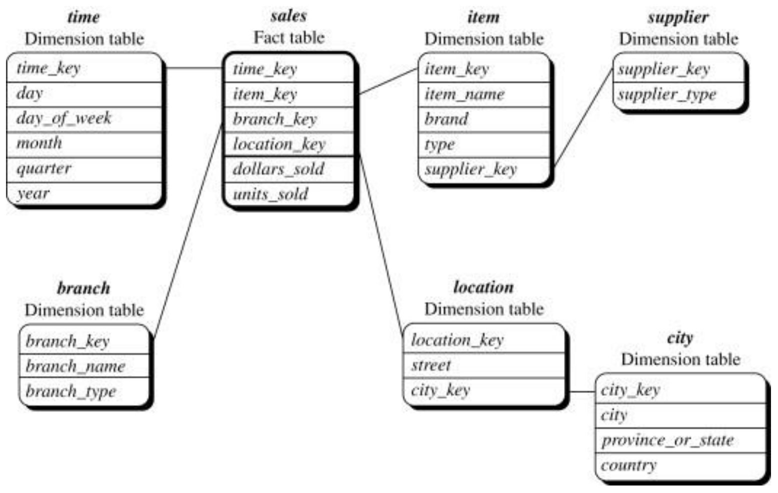


Figure 6 Snowflake schema of a sales data warehouse.

A snowflake schema for AllElectronics sales is given in Figure 6. Here, the sales fact table is identical to that of the star schema in Figure 5. The main difference between the two schemas is in the definition of dimension tables. The single dimension table for an item in the star schema is normalized in the snowflake schema, resulting in new item and supplier tables. For example, the item dimension table now contains the attributes *item key, item name, brand, type, and supplier key,* where supplier key is linked to the supplier dimension table, containing supplier key and supplier type information. Similarly, the single dimension table for location in the star schema can be normalized into two new tables: location and city. The city key in the new location table links to the city dimension. Notice that, when desirable, further normalization can be performed on province or state and country in the snowflake schema shown in Figure 6.

Fact constellation: 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.

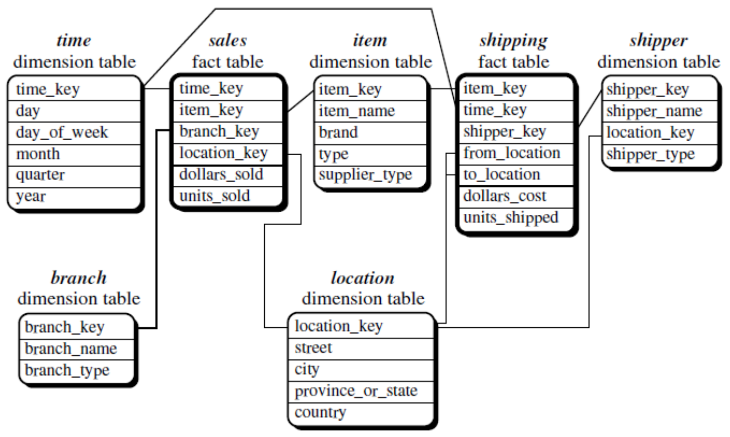


Figure 7 Fact constellation schema of a sales and shipping data warehouse.

A fact constellation schema is shown in Figure 7. This schema specifies two fact tables, sales, and shipping. The sales table definition is identical to that of the star schema (Figure 5). The shipping table has five dimensions, or keys—item key, time key, shipper key, from location, and to location—and two measures—*dollars cost and units shipped*. A fact constellation schema allows dimension tables to be shared between fact tables. For example, the dimensions tables for time, item, and location are shared between the sales and shipping fact tables.

1.7 Concept Hierarchies

*A **concept hierarchy** defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. Consider a concept hierarchy for the dimension location. City values for location include*

Vancouver, Toronto, New York, and Chicago. Each city, however, can be mapped to the province or state to which it belongs. For example, Vancouver can be mapped to British Columbia, and Chicago to Illinois. The provinces and states can in turn be mapped to the country (e.g., Canada or the United States) to which they belong. These mappings form a concept hierarchy for the dimension location, mapping a set of low-level concepts (i.e., cities) to higher-level, more general concepts (i.e., countries). This concept hierarchy is illustrated in

Figure 8.

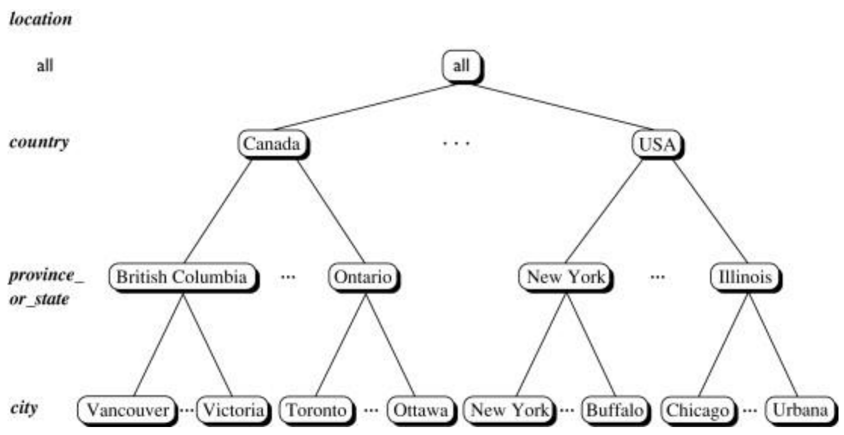


Figure 8 A concept hierarchy for location.

1.8 Measures: Their Categorization and Computation

"How are measures computed?" To answer this question, we first study how measures can be categorized. Note that a multidimensional point in the data cube space can be defined by a set dimension–value pairs; for example,(time ="Q1", location = "Vancouver", item ="computer"). A data cube measure is a numeric function that can be evaluated at each point in the data cube space. A measured value is computed for a given point by aggregating the data corresponding to the respective dimension-value pairs defining the given point. We will look at concrete examples of this shortly.

Distributive: An aggregate function is distributive if it can be computed in a distributed manner as follows. Suppose the data are partitioned into n sets. We apply the function to each partition, resulting in n aggregate values. If the result derived by applying the function to the n aggregate values is the same as that derived by applying the function to the entire data set (without partitioning), the function can be computed in a distributed manner. For example, sum() can be computed for a data cube by first partitioning the cube into a set of sub-cubes, computing sum() for each sub-cube, and then summing up the counts obtained for each sub-cube. Hence, sum() is a distributive aggregate function. For the same reason, count(), min(), and max() are distributive aggregate functions.

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() (average) can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions. Similarly, it can be shown that min N() and max N() (which find the N minimum and N maximum values, respectively, in a given set) and standard deviation() are algebraic aggregate functions. A measure is algebraic if it is obtained by applying an algebraic aggregate function.

Holistic: An aggregate function is holistic if there is no constant bound on the storage size needed to describe a sub-aggregate. That is, there does not exist an algebraic function with M arguments (where M is a constant) that characterizes the computation. Common examples of holistic functions include median(), mode(), and rank(). A measure is holistic if it is obtained by applying a holistic aggregate function.

1.9 OLAP Operations

"A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.

Dimensions are the entities concerning which an organization wants to keep records.

Facts are numerical measures. It is the quantities by which we want to analyze relationships between dimensions.

The data cube is used by the users of the decision support system to see their data. 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**.

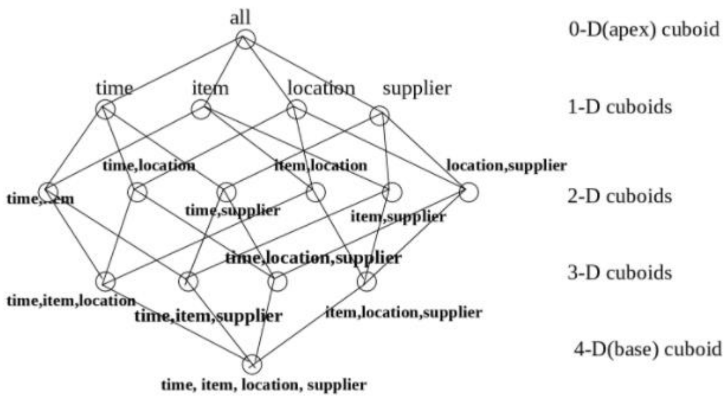


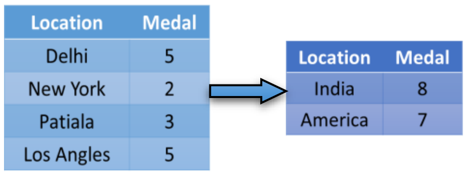
Figure 9 A Lattice of Cuboids

1.10 Operations in the Multidimensional Data Model (OLEP)

In the multidimensional model, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. The organization provides users with the flexibility to view data from different perspectives. Some OLAP data cube operations exist to materialize these different views, allowing interactive querying and analysis of the data at hand.

Five basic OLAP commands are used to perform data retrieval from a Data warehouse.

1. Roll-up (drill-up):- The roll-up operation performs aggregation on a data cube either by climbing up the hierarchy or by dimension reduction.



Location	Medal
Delhi	5
New York	2
Patiala	3
Los Angles	5

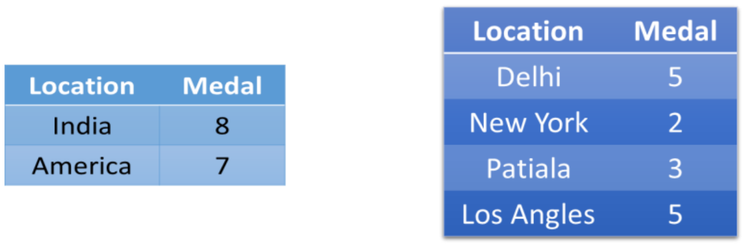
Location	Medal
India	8
America	7

Delhi, New York, Patiala, and Los Angeles wins 5, 2, 3, and 5 medals respectively. So in this example, we roll upon Location from cities to countries.

2. Drill-down:- Drill-down is the reverse of roll-up. That means lower-level summary to higher-level summary.

Drill-down can be performed either by:-

- Stepping down a concept hierarchy for a dimension.
- By introducing a new dimension.



Location	Medal
India	8
America	7

Location	Medal
Delhi	5
New York	2
Patiala	3
Los Angles	5

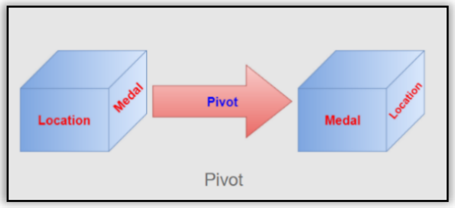
Drill-down on Location from countries to cities.

3. Slice and dice:- The slice operation perform a selection on one dimension of the given cube, resulting in a subcube. Reduces the dimensionality of the cubes. For example, if we want to make a selection where Medal = 5

Location	Medal
Delhi	5
Los Angles	5

The dice operation defines a sub-cube by performing a selection on two or more dimensions. For example, if we want to make a selection where Medal = 3 or Location = New York.

4. Pivot:- Pivot is also known as rotate. It rotates the data axis to view the data from different perspectives.



Discuss OLAP application identify its characteristics.

1.11 DataWarehouse Design and Usage

"What can business analysts gain from having a data warehouse?" First, having a data warehouse may provide a competitive advantage by presenting relevant information from which to measure performance and make critical adjustments to help win over competitors. Second, a data warehouse can enhance business productivity because it can quickly and efficiently gather the information that accurately describes the organization. Third, a data warehouse facilitates customer relationship management because it provides a consistent view of customers and items across all lines of business, all departments, and all markets. Finally, a data warehouse may bring about cost reduction by tracking trends, patterns, and exceptions over long periods consistently and reliably.

To design an effective data warehouse we need to understand and analyze business needs and construct a business analysis framework. The construction of a large and complex information system can be viewed as the construction of a large and complex building, for which the owner, architect, and builder have different views.

Four different views regarding a data warehouse design must be considered: *the top-down view, the data source view, the data warehouse view,* and *the business query view*.

- The **top-down view** allows the selection of the relevant information necessary for the data warehouse. This information matches current and future business needs.
- The **data source view** exposes the information being captured, stored, and managed by operational systems. This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables. Data sources are often modeled by traditional data modeling techniques, such as the entity-relationship model or CASE (computer-aided software engineering) tools.

- The **data warehouse view** includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- Finally, the **business query view** is the data perspective in the data warehouse from the end-user's viewpoint.

Building and using a data warehouse is a complex task because it requires *business skills, technology skills*, and *program management skills*. Regarding *business skills*, building a data warehouse involves understanding how systems store and manage their data, how to build **extractors** that transfer data from the operational system to the data warehouse, and how to build **warehouse refresh software** that keeps the data warehouse reasonably up-to-date with the operating system's data. Using a data warehouse involves understanding the significance of the data it contains, as well as understanding and translating the business requirements into queries that can be satisfied by the data warehouse.

Regarding *technology skills*, data analysts are required to understand how to make assessments from quantitative information and derive facts based on conclusions from historic information in the data warehouse. These skills include the ability to discover patterns and trends, extrapolate trends based on history and look for anomalies or paradigm shifts, and to present coherent managerial recommendations based on such analysis. Finally, *program management skills* involve the need to interface with many technologies, vendors, and end-users to deliver results in a timely and cost-effective manner.

Data Warehouse Design Process

Let's look at various approaches to the data warehouse design process and the steps involved. A data warehouse can be built using a **top-down approach, a bottom-up approach**, or a **combination of both.** The top-down approach starts with overall design and planning. It is useful in cases where the technology is mature and well known, and where the business problems that must be solved are clear and well understood. The *bottom-up approach* starts with experiments and prototypes. This is useful in the early stage of business modeling and technology development. It allows an organization to move forward at considerably less expense and to evaluate the technological benefits before making significant commitments. In the combined approach, an organization can exploit the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach.

In general, the warehouse design process consists of the following steps:

1. Choose a business process to model (e.g., orders, invoices, shipments, inventory, account administration, sales, or the general ledger). If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed. However, if the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.
2. Choose the business process grain, which is the fundamental, atomic level of data to be represented in the fact table for this process (e.g., individual transactions, individual daily snapshots, and so on).
3. Choose the dimensions that will apply to each fact table record. Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
4. Choose the measures that will populate each fact table record. Typical measures are numeric additive quantities like dollars sold and units sold.

Because data warehouse construction is a difficult and long-term task, its implementation scope should be clearly defined. The goals of an initial data warehouse implementation should be *specific, achievable, and measurable*. This involves determining the time and budget allocations, the subset of the organization that is to be modeled, the number of data sources selected, and the number and types of departments to be served.

Once a data warehouse is designed and constructed, the initial deployment of the warehouse includes initial installation, roll-out planning, training, and orientation. Platform upgrades and maintenance must also be considered. Data warehouse administration includes data refreshment, data source synchronization, planning for disaster recovery, managing access control and security, managing data growth, managing database performance, and data warehouse enhancement and extension. Scope management includes controlling the number and range of queries, dimensions, and reports; limiting the data warehouse's size; or limiting the schedule, budget, or resources. Various kinds of data warehouse design tools are available. **Data warehouse development tools** provide functions to define and edit metadata repository contents (e.g., schemas, scripts, or rules), answer queries, output reports, and ship metadata to and from relational database system catalogs. **Planning and analysis tools** study the impact of schema changes and refresh performance when changing refresh rates or time windows.

Data Warehouse Usage for Information Processing

Data warehouses and data marts are used in a wide range of applications. There are three kinds of data warehouse applications*: information processing, analytical processing, and data mining.*

Information processing supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts, or graphs. A current trend in data warehouse information processing is to construct low-cost web-based accessing tools that are then integrated with web browsers.

Analytical processing supports basic OLAP operations, including slice-and-dice, drill-down, roll-up, and pivoting. It generally operates on historic data in both summarized and detailed forms. The major strength of online analytical processing over information processing is the multidimensional data analysis of data warehouse data.

Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

"*How does data mining relate to information processing and online analytical processing*?" Information processing, based on queries, can find useful information. However, answers to such queries reflect the information directly stored in databases or computable by aggregate functions. They do not reflect sophisticated patterns or regularities buried in the database. Therefore, information processing is not data mining. Because data mining systems can also mine generalized class/concept descriptions, this raises some interesting questions: "*Do OLAP systems perform data mining? Are OLAP systems are data mining systems?*" The functionalities of OLAP and data mining can be viewed as disjoint: OLAP is a data summarization/aggregation tool that helps simplify data analysis, while data mining allows the automated discovery of implicit patterns and interesting knowledge hidden in large amounts of data. OLAP tools are targeted toward simplifying and supporting interactive data analysis, whereas the goal of data mining tools is to automate as much of the process as possible, while still allowing users to guide the process. In this sense, data mining goes one step beyond traditional online analytical processing. Data mining is not confined to the analysis of data stored in data warehouses. It may analyze data existing at more detailed granularities than the summarized data provided in a data warehouse. It may also analyze transactional, spatial, textual, and multimedia data that are difficult to model with current multidimensional database technology. In this context, data mining covers a broader spectrum than OLAP for data mining functionality and the complexity of the data handled.

1.12 From Online Analytical Processing to Multidimensional Data Mining

The data mining field has conducted substantial research regarding mining on various data types, including relational data, data from data warehouses, transaction data, time-series data, spatial data, text data, and flat files. **Multidimensional data mining** (also known as *exploratory multidimensional data mining*, **online analytical mining, or OLAM**) integrates OLAP with data mining to uncover knowledge in multidimensional databases. Among the many different paradigms and architectures of data mining systems, multidimensional data mining is particularly important for the following reasons:

High quality of data in data warehouses: Most data mining tools need to work on integrated, consistent, and cleaned data, which requires costly data cleaning, data integration, and data transformation as preprocessing steps. A data warehouse constructed by such preprocessing serves as a valuable source of high-quality data for OLAP as well as for data mining. Notice that data mining may serve as a valuable tool for data cleaning and data integration as well. Available information processing infrastructure surrounding data warehouses: Comprehensive information processing and data analysis infrastructures have been or will be systematically constructed surrounding data warehouses, which include access, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web access, and service facilities, and reporting and OLAP analysis tools. It is prudent to make the best use of the available infrastructures rather than constructing everything from scratch.

OLAP-based exploration of multidimensional data: Effective data mining needs exploratory data analysis. A user will often want to traverse through a database, select portions of relevant data, and analyze them at different granularities, and present knowledge/results in different forms. Multidimensional data mining provides facilities for mining on different subsets of data and at varying levels of abstraction—by drilling, pivoting, filtering, dicing, and slicing on a data cube and/or intermediate data mining results. This, together with data/knowledge visualization tools, greatly enhances the power and flexibility of data mining.

Online selection of data mining functions: Users may not always know the specific kinds of knowledge they want to mine. By integrating OLAP with various data mining functions, multidimensional data mining provides users with the flexibility to select desired data mining functions and swap data mining tasks dynamically.

1.13 DataWarehouse Implementation

Data warehouses contain huge volumes of data. OLAP servers demand that decision support queries be answered in the order of seconds. Therefore, it is crucial for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques.

Efficient Data Cube Computation: An Overview

At the core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions. In SQL terms, these aggregations are referred to as group-by's. Each group-by can be represented by a cuboid, where the set of group-by's forms a lattice of cuboids defining a data cube. In this subsection, we explore issues relating to the efficient computation of data cubes.

The compute cube Operator and the Curse of Dimensionality

One approach to cube computation extends SQL to include a compute cube operator. The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation. This can require excessive storage space, especially for large numbers of dimensions. We start with an intuitive look at what is involved in the efficient computation of data cubes. Suppose that you want to create a data cube for all Electronics sales contain the following: city, item, year, and sales in dollars. You want to be able to analyze the data, with queries such as the following:

"Compute the sum of sales, grouping by city and item."

"Compute the sum of sales, grouping by city."

"Compute the sum of sales, grouping by item."

What is the total number of cuboids, or group-by's, that can be computed for this data cube? Taking the three attributes, city, item, and year, as the dimensions for the data cube, and sales in dollars as the measure, the total number of cuboids, or group by's, that can be computed for this data cube is 2^3=8. The possible group-by's are the following: {(city, item, year), (city, item), (city, year), (item, year), (city), (item), (year),( ),} where ( )means that the group-by is empty (i.e., the dimensions are not grouped). These group-by's form a lattice of cuboids for the data cube, as shown in Figure 10.

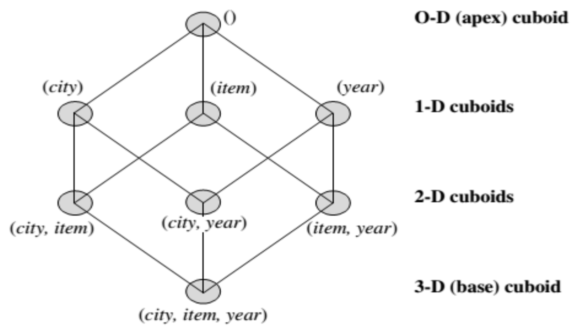


Figure 10 Lattice of cuboids, making up a 3-D data cube.

The **base cuboid** contains all three dimensions, city, item, and year. It can return the total sales for any combination of the three dimensions. The **apex cuboid**, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales. The base cuboid is the least generalized (most specific) of the cuboids. The apex cuboid is the most generalized (least specific) of the cuboids and is often denoted as all. If we start at the apex cuboid and explore downward in the lattice, this is equivalent to drilling down within the data cube. If we start at the base cuboid and explore upward, this is akin to rolling up.

Online analytical processing may need to access different cuboids for different queries. Therefore, it may seem like a good idea to compute in advance all or at least some of the cuboids in a data cube. A major challenge related to this pre-computation, however, is that the required storage space may explode if all the cuboids in a data cube are pre-computed, especially when the cube has many dimensions. The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels. This problem is referred to as the **curse of dimensionality**.

"How many cuboids are there in an n-dimensional data cube?" If there were no hierarchies associated with each dimension, then the total number of cuboids for an n-dimensional data cube, as we have seen, is 2n. However, in practice, many dimensions do have hierarchies. For example, time is usually explored not at only one conceptual level (e.g., year), but rather at multiple conceptual levels such as in the hierarchy "day < month < quarter < year." For an n-dimensional data cube, the total number of cuboids can be generated.

Total number of cuboids D, $= \prod_{i=1}^{n} (L_i + 1),$

.

where Li is the number of levels associated with dimension i. One is added to Li to include the virtual top level, all. This formula is based on the fact that, at most, one abstraction level in each dimension will appear in a cuboid. For example, the time dimension as specified before has four conceptual levels, or five if we include the virtual level all. If the cube has 10 dimensions and each dimension has five levels (including all), the total number of cuboids that can be generated is $5^{10} \approx 9.8 \times 10^6$. The size of each cuboid also depends on the cardinality (i.e., number of distinct values) of each dimension.

Partial Materialization: Selected Computation of Cuboids

There are three choices for data cube materialization given a base cuboid:

1. **No materialization:** Do not pre-compute any of the "nonbase" cuboids. This leads to computing expensive multidimensional aggregates on-the-fly, which can be extremely slow.

2. **Full materialization:** Precompute all of the cuboids. The resulting lattice of computed cuboids is referred to as the full cube. This choice typically requires huge amounts of memory space to store all of the precomputed cuboids.

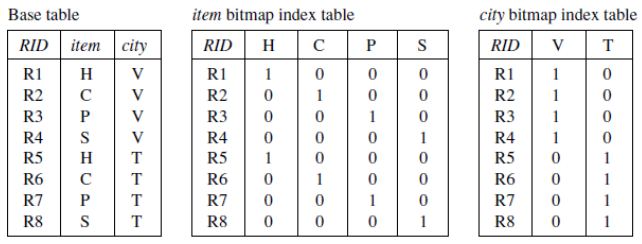
3. **Partial materialization**: Selectively compute a proper subset of the whole set of possible cuboids. Alternatively, we may compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion, such as where the tuple count of each cell is above some threshold. We will use the term subcube to refer to the latter case, where only some of the cells may be precomputed for various cuboids. Partial materialization represents an interesting trade-off between storage space and response time.

The partial materialization of cuboids or subcubes should consider three factors: (1) identify the subset of cuboids or subcubes to materialize; (2) exploit the materialized cuboids or subcubes during query processing; and (3) efficiently update the materialized cuboids or subcubes during load and refresh. The selection of the subset of cuboids or subcubes to materialize should take into account the queries in the workload, their frequencies, and their accessing costs. Alternatively, we can compute an **iceberg cube**, which is a data cube that stores only those cube cells with an aggregate value (e.g., count) that is above some minimum support threshold. Another common strategy is to materialize a **shell cube**. This involves precomputing the cuboids for only a small number of dimensions (e.g., three to five) of a data cube.

1.14 Indexing OLAP Data: Bitmap Index and Join Index

To facilitate efficient data accessing, most data warehouse systems support index structures and materialized views. The **bitmap indexing** method is popular in OLAP products because it allows quick searching in data cubes. The bitmap index is an alternative representation of the record_ID (RID) list. In the bitmap index for a given attribute, there is a distinct bit vector, B*v*, for each value v in the attribute's domain. If a given attribute's domain consists of n values, then n bits are needed for each entry in the bitmap index (i.e., there are n bit vectors). If the attribute has the value v for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0.

In the AllElectronics data warehouse, suppose the dimension item at the top level has four values (representing item types): "home entertainment," "computer," "phone," and "security." Each value (e.g., "computer") is represented by a bit vector in the item bitmap index table. Suppose that the cube is stored as a relation table with 100,000 rows. Because the domain of the item consists of four values, the bitmap index table requires four-bit vectors (or lists), each with 100,000 bits. Figure 11 shows a base (data) table containing the dimensions item and city, and its mapping to bitmap index tables for each of the dimensions.


Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

city bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Figure 11 Indexing OLAP data using bitmap indices.

Bitmap indexing is advantageous compared to hash and Tree indices. It is especially useful for low-cardinality domains because comparison, join, and aggregation operations are then reduced to bit arithmetic, which substantially reduces the processing time. Bitmap indexing leads to significant reductions in space and input/output (I/O) since a string of characters can be represented by a single bit. For higher-cardinality domains, the method can be adapted using compression techniques.

The **join indexing** method gained popularity from its use in relational database query processing. Traditional indexing maps the value in a given column to a list of rows having that value. In contrast, join indexing registers the joinable rows of two relations from a relational database. For example, if two relations R.RID, A/ and S.B, SID/ join on the attributes A and B, then the join index record contains the pair.RID, SID/, where RID and SID are record identifiers from the R and S relations, respectively. Hence, the join index records can identify joinable tuples without performing costly join operations. Join indexing is especially useful for maintaining the relationship between a foreign key and its matching primary keys, from the joinable relation.

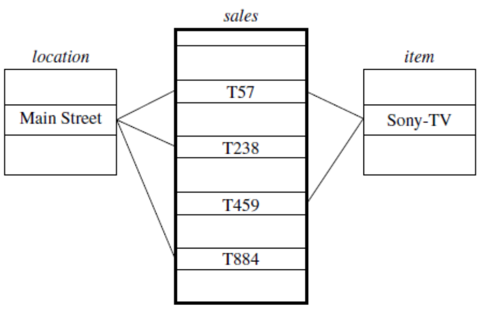
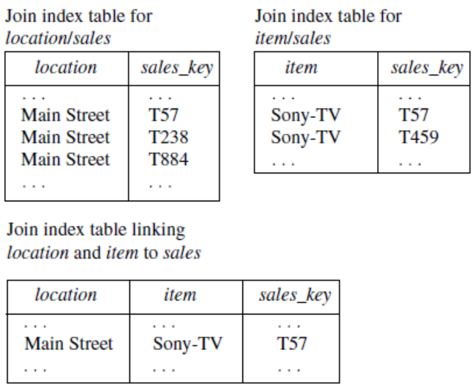


Figure 12 Linkages between a sales fact table and location and item dimension tables.

We defined a star schema for AllElectronics of the form "sales star [time, item, branch, location]: dollars_sold = sum (sales_in_dollars)." An example of a join index relationship between the sales fact table and the location and item dimension tables is shown in Figure 12. For example, the "Main Street" value in the location dimension table joins with tuples T57, T238, and T884 of the sales fact table. Similarly, the "Sony-TV" value in the item dimension table joins with tuples T57 and T459 of the sales fact table. The corresponding join index tables are shown in Figure 13.



Join index table for *location/sales*

location	*sales_key*
. . .	. . .
Main Street	T57
Main Street	T238
Main Street	T884
. . .	. . .

Join index table for *item/sales*

item	*sales_key*
. . .	. . .
Sony-TV	T57
Sony-TV	T459
. . .	. . .

Join index table linking *location* and *item* to *sales*

location	*item*	*sales_key*
. . .	. . .	. . .
Main Street	Sony-TV	T57
. . .	. . .	. . .

Figure 13 Join index tables based on the linkages between the sales fact table and the location and item.

Efficient Processing of OLAP Queries

The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes. Given materialized views, query processing should proceed as follows:

1. **Determine which operations should be performed on the available cuboids:** This involves transforming any selection, projection, roll-up (group-by), and drill-down operations specified in the query into the corresponding SQL and/or OLAP operations. For example, slicing and dicing a data cube may correspond to selection and/or projection operations on a materialized cuboid.

2. **Determine to which materialized cuboid(s) the relevant operations should be applied:** This involves identifying all of the materialized cuboids that may potentially be used to answer the query, pruning the set using knowledge of "dominance" relationships among the cuboids, estimating the costs of using the remaining materialized cuboids and selecting the cuboid with the least cost.

Suppose that we define a data cube for AllElectronics of the form"sales cube [time, item, location]: sum(sales_in_dollars)." The dimension hierarchies used are"day < month < quarter < year" for time; "item name < brand < type" for item; and "street < city < province or state < country" for location. Suppose that the query to be processed is on {brand, province, or state}, with the selection constant "year = 2010." Also, suppose that there are four materialized cuboids available, as follows:

cuboid 1: {year, item name, city}

cuboid 2: {year, brand, country}

cuboid 3: {year, brand, province or state}

cuboid 4: {item name, province or state}, where year = 2010

"Which of these four cuboids should be selected to process the query?" Finer-granularity data cannot be generated from coarser-granularity data. Therefore, cuboid 2 cannot be used because the country is a more general concept than province or state. Cuboids 1, 3, and 4 can be used to process the query because (1) they have the same set or a superset of the dimensions in the query, (2) the selection clause in the query can imply the selection in the cuboid, and (3) the abstraction levels for the item and location dimensions in these cuboids are at a finer level than brand and province or state, respectively.

"How would the costs of each cuboid compare if used to process the query?" Using cuboid 1 would likely cost the most because both item name and city are at a lower level than the brand and province or state concepts specified in the query. If there are not many year values associated with items in the cube, but there are several item names for each brand, then cuboid 3 will be smaller than cuboid 4, and thus cuboid 3 should be chosen to process the query. However, if efficient indices are available for cuboid 4, then cuboid 4 may be a better choice. Therefore, some cost-based estimation is required to decide which set of cuboids should be selected for query processing.

Summary

- A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile data collection organized in support of management decision-making.
- A data warehouse contains back-end tools and utilities for populating and refreshing the warehouse. These cover data extraction, data cleaning, data transformation, loading, refreshing, and warehouse management.
- A multidimensional data model is typically used for the design of corporate data warehouses and departmental data marts.
- A data cube consists of a lattice of cuboids, each corresponding to a different degree of summarization of the given multidimensional data.
- Concept hierarchies organize the values of attributes or dimensions into gradual abstraction levels. They are useful in mining at multiple abstraction levels.
- Full materialization refers to the computation of all of the cuboids in the lattice defining a data cube.
- OLAP query processing can be made more efficient with the use of indexing techniques.

Keywords

Data Sources: Data sources refer to any electronic repository of information that contains data of interest for management use or analytics.

Data Warehouse: It is a relational database that is designed for query and analysis rather than for transaction processing.

Data Mart: Data marts contain a subset of organization-wide data that is valuable to specific groups of people in an organization.

Dimensions: Dimensions contain a set of unique values that identify and categories data.

Hierarchy: A hierarchy is a way to organize data at different levels of aggregation.

Star Schema: A star schema is a convention for organizing the data into dimension tables, fact tables, and materialized views.

Self Assessment

1) OLTP stands for

(a) On-Line Transactional Processing

(b) On Link Transactional Processing

(c) On-Line Transnational Process

(d) On-Line Transactional Program

2) Data warehouse is

(a) The actual discovery phase of a knowledge discovery process

(b) The stage of selecting the right data for a KDD process

(c) A subject-oriented integrated time-variant non-volatile collection of data in support of management

(d) None of these

3) A data warehouse is which of the following?

(a) Can be updated by end-users.

(b) Contains numerous naming conventions and formats.

(c) Organized around important subject areas.

(d) Contains only current data. "

4) The data warehouse is normally a ....

5) A .... is a physical database that receives all its information from the data warehouse.

Review Questions

1. Describe materialized views with the help of a suitable example.
2. What are the differences between the three main types of data warehouse usage: information processing, analytical processing, and data mining? Discuss the motivation behind OLAP mining (OLAM).
3. Describe OLAP operations in the multi-dimensional data model.
4. "Concept hierarchies that are common to many applications may be predefined in the data mining system". Explain.
5. "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process." Discuss.
6. Differences between operational database systems and data warehouses.

Answers: Self-Assessment

1. a	2. c
3. c	4. relational database
5. dependent data mart	

Further Readings

Jiawei Han, Micheline Kamber, Data Mining – Concepts and Techniques, Morgan Kaufmann Publishers, First Edition, 2003.

Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, Panos Vassiliadis, *Fundamentals of Data Warehouses*, Publisher: Springer.

The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd Edition.

Data Warehousing Fundamentals for IT Professionals.

Sam Anohory, Dennis Murray, *Data Warehousing in the Real World*, Addison Wesley, First Edition, 2000.

https://www.javatpoint.com/data-mining-cluster-vs-data-warehousing.

https://www.classcentral.com/subject/data-warehousing

https://www.tutorialspoint.com/dwh/dwh_data_warehousing.htm

https://www.oracle.com/in/database/what-is-a-data-warehouse