Data Warehousing and Online Analytical Processing

Content to be covered

- Data Warehouse basic concepts,
- Data Warehouse Modelling - Data Cube and OLAP,
- Data Warehouse Design and Usage,
- Data Warehouse Implementation

What Is a Data Warehouse?

- A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process. [William H. Inmon]
- Subject-oriented:
  - A data warehouse is organized around major subjects such as customer, supplier, product, and sales.
  - Rather than concentrating on the day-to-day operations and transaction processing of an organization, a data warehouse focuses on the modeling and analysis of data for decision makers.
  - Data warehouses provide a simple and concise 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 are stored to provide information from an historic perspective (e.g., the past 5–10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, a time element.
- Nonvolatile:
  - A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment.
  - 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.

Data Warehouses

- In sum, a data warehouse is a semantically consistent data store that serves as a physical implementation of a decision support data model.
- It stores the information an enterprise needs to make strategic decisions.
- A data warehouse is also often viewed as an architecture, constructed by integrating data from multiple heterogeneous sources to support structured and/or ad hoc queries, analytical reporting, and decision making.
- The construction of a data warehouse requires data cleaning, data integration, and data consolidation. The utilization of a data warehouse often necessitates a collection of decision support technologies. This allows “knowledge workers” (e.g., managers, analysts, and executives) to use the warehouse to quickly and conveniently obtain an overview of the data, and to make sound decisions based on information in the warehouse.

Data Warehouses

- “How are organizations using the information from data warehouses?”
  - increasing customer focus, which includes the analysis of customer buying patterns;
  - repositioning products and managing product portfolios by comparing the performance of sales by quarter, by year, and by geographic regions in order to fine-tune production strategies;
  - analyzing operations and looking for sources of profit; and
  - managing customer relationships, making environmental corrections, and managing the cost of corporate assets.
- Data warehousing is also very useful from the point of view of heterogeneous database integration.
- Data warehousing employs an update driven approach in which information from multiple, heterogeneous sources is integrated in advance and stored in a warehouse for direct querying and analysis. Unlike online transaction processing databases, data warehouses do not contain the most current information. However, a data warehouse brings high performance to the integrated heterogeneous database system because data are copied, preprocessed, integrated, annotated, summarized, and restructured into one semantic data store. Furthermore, query processing in data warehouses does not interfere with the processing at local sources. Moreover, data warehouses can store and integrate historic information and support complex multidimensional queries. As a result, data warehousing has become popular in industry.

Differences between Operational Database Systems and Data Warehouses

- 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, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as online analytical processing (OLAP) systems.

Differences between Operational Database Systems and Data Warehouses

- Users and system orientation: An OLTP system is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.
- Data contents: An OLTP system manages current data and too detailed. An OLAP system manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.
- Database design: An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design. An OLAP system typically adopts either a star or a snowflake model and a subject-oriented database design.
- View: An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historic 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. Because of their huge volume, OLAP data are stored on multiple storage media.
- Access patterns: The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations (because most data warehouses store historic rather than up-to-date information), although many could be complex queries.

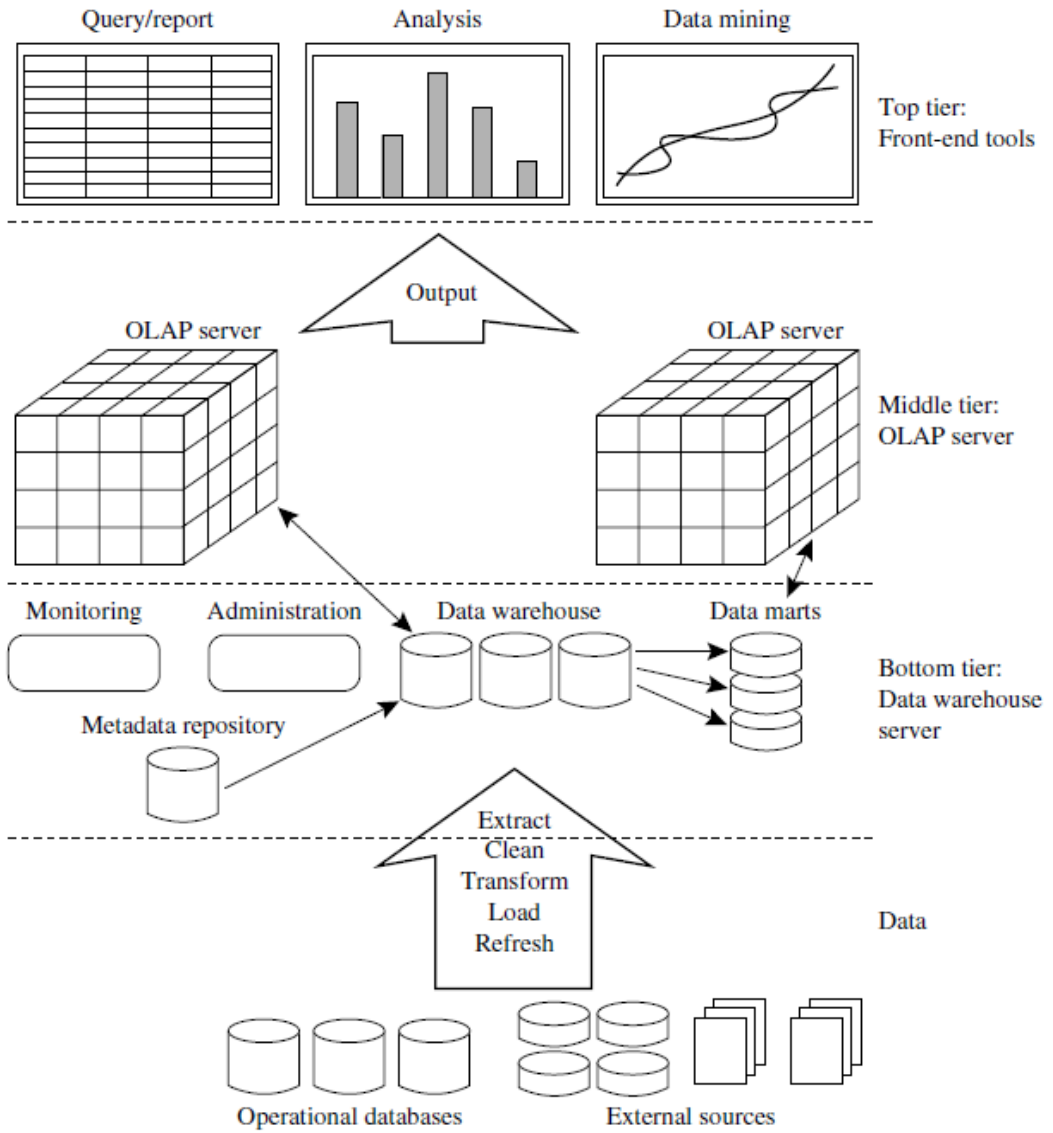
Why Have a Separate Data Warehouse?

- A major reason for having separate data warehouses 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.
  - But, 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.
- 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 of 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.

Why Have a Separate Data Warehouse?

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

Data Warehousing: A Multitiered Architecture



Data Warehousing: A Multitiered Architecture

- 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. These tools and utilities perform data extraction, cleaning, and transformation, as well as load and refresh functions to update the data warehouse. 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 Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents.
- The middle tier is an OLAP server that is typically implemented using either
  - a relational OLAP(ROLAP) model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations); or
  - a multidimensional OLAP (MOLAP) model (i.e., a special-purpose server that directly implements multidimensional data and operations).
- The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

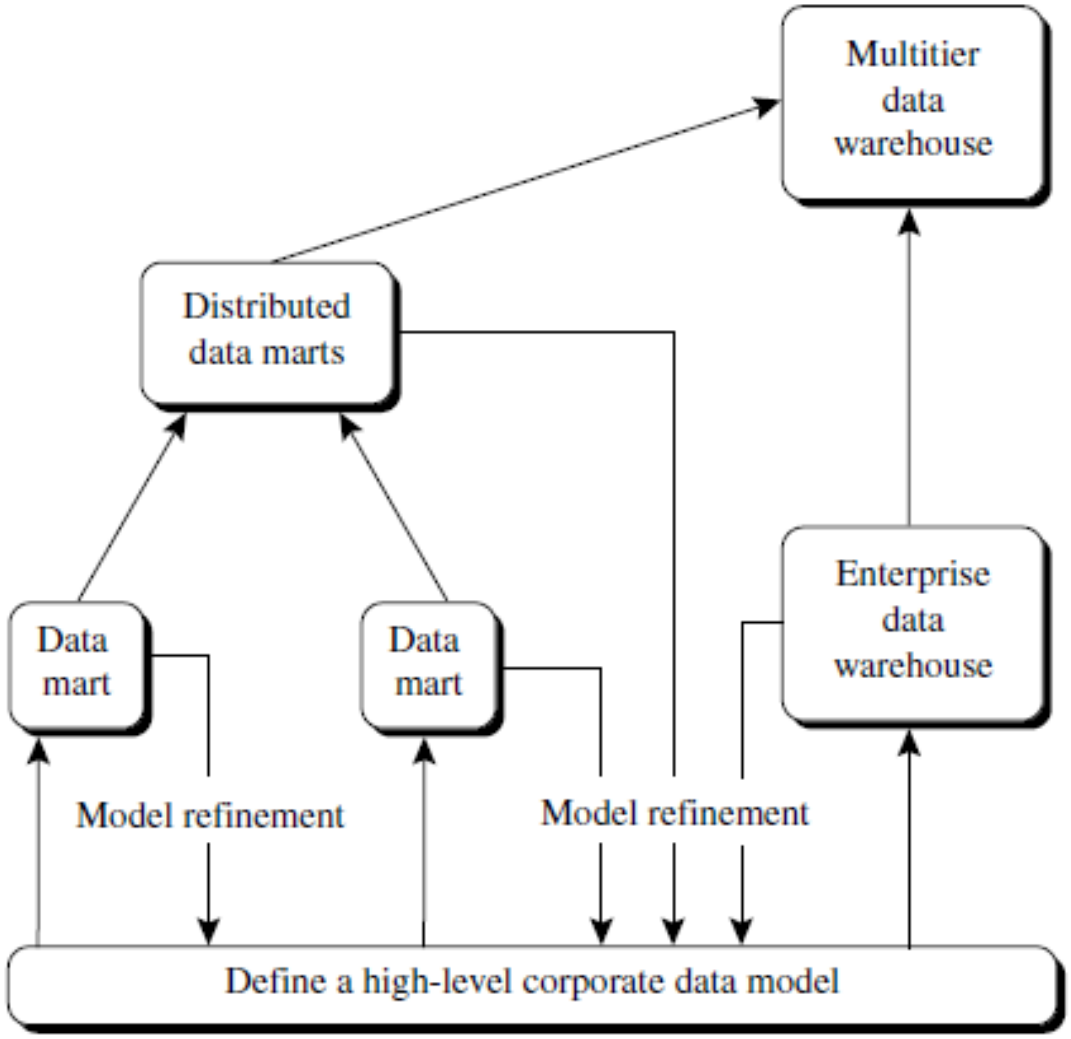
Data Warehouse Models: Enterprise Warehouse, Data Mart, and Virtual Warehouse

- Enterprise warehouse:
  - An enterprise warehouse collects all of the information about subjects spanning the entire organization.
  - It provides corporate-wide data integration, from one or more operational systems or external information providers, and is cross-functional in scope.
  - It contains detailed data and summarized data, and can range in size.
  - 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.
- Data mart:
  - 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. The data is summarized.
  - Data marts are usually implemented on low-cost departmental servers requires weeks to build.
  - It may involve complex integration in the long run if its design and planning were not enterprise-wide.
  - Data marts can be categorized as independent or dependent.
    - Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area.
    - 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.

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

- 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.
- A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner.
  - 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.

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



Extraction, Transformation, and Loading

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

Metadata Repository

- Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouse objects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for timestamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.
- A metadata repository should contain the following:
  - A description of the data warehouse structure, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
  - Operational metadata, which include data lineage, currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
  - The algorithms used for summarization, which include 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, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).
  - Data related to system performance, which include 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, which include business terms and definitions, data ownership information, and charging policies.

Data Warehouse Modeling: Data Cube

- Data warehouses and OLAP tools are based on a multidimensional data model
- 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. For example, store's sales with respect to the dimensions time, item, branch, and location. These dimensions allow the store to keep track of things like monthly sales of items and the branches and locations at which the items were sold.
- Each dimension may have a table associated with it, called a dimension table, which further describes the dimension. For example, a dimension table for 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.

Data Warehouse Modeling: Data Cube

- 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 amount budgeted. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.
- Although we usually think of cubes as 3-D geometric structures, in data warehousing the data cube is n-dimensional.

Data Warehouse Modeling: Data Cube

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

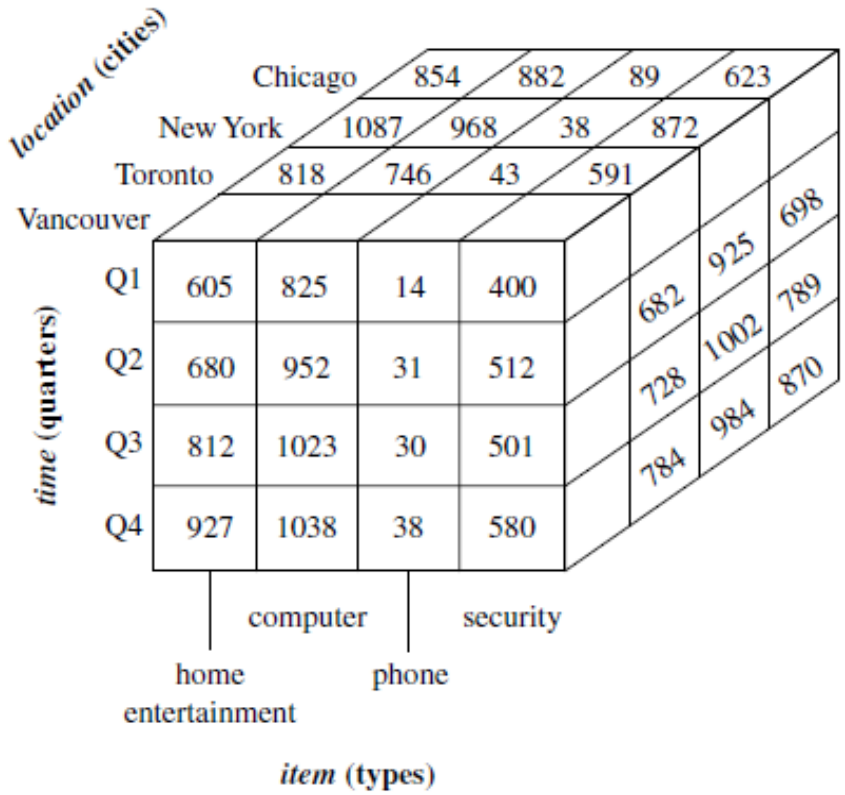
	location = "Vancouver"			
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

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

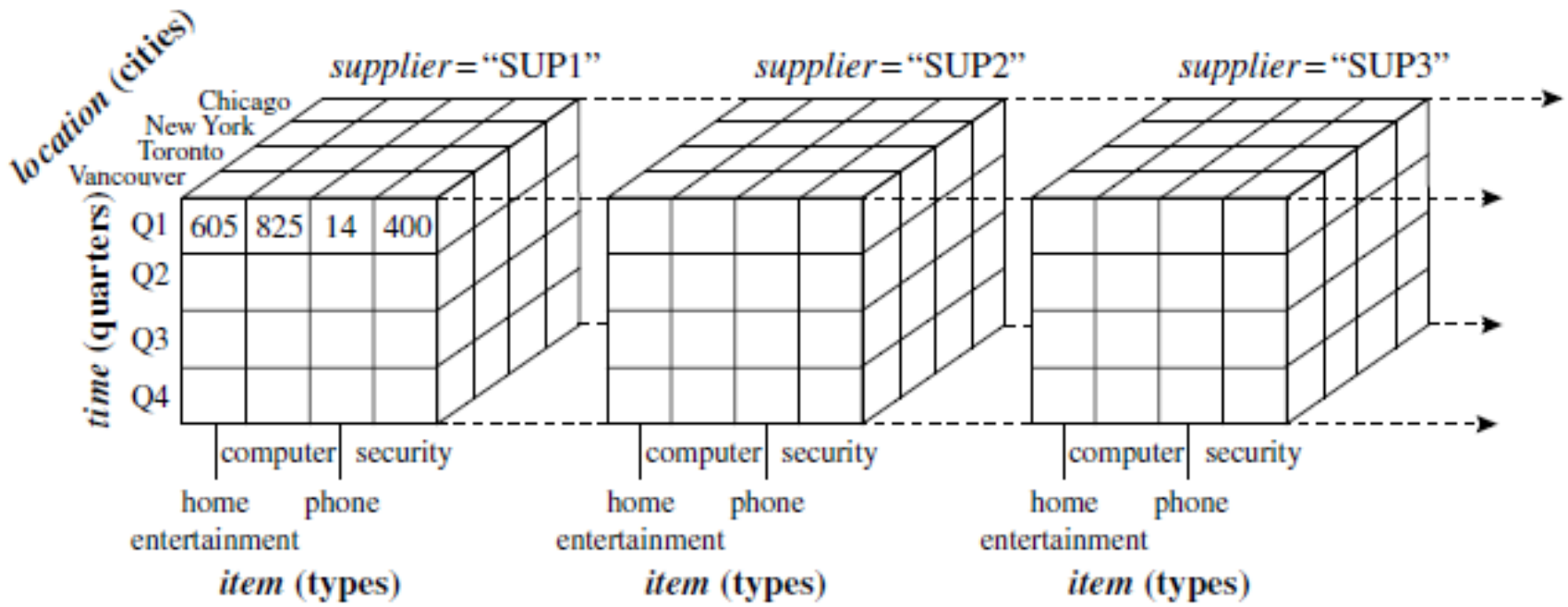
Data Warehouse Modeling: Data Cube

	location = "Chicago"				***location*** = "New York"				***location*** = "Toronto"				***location*** = "Vancouver"			
	item				***item***				***item***				***item***			
	home				*home*				*home*				*home*			
time	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*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

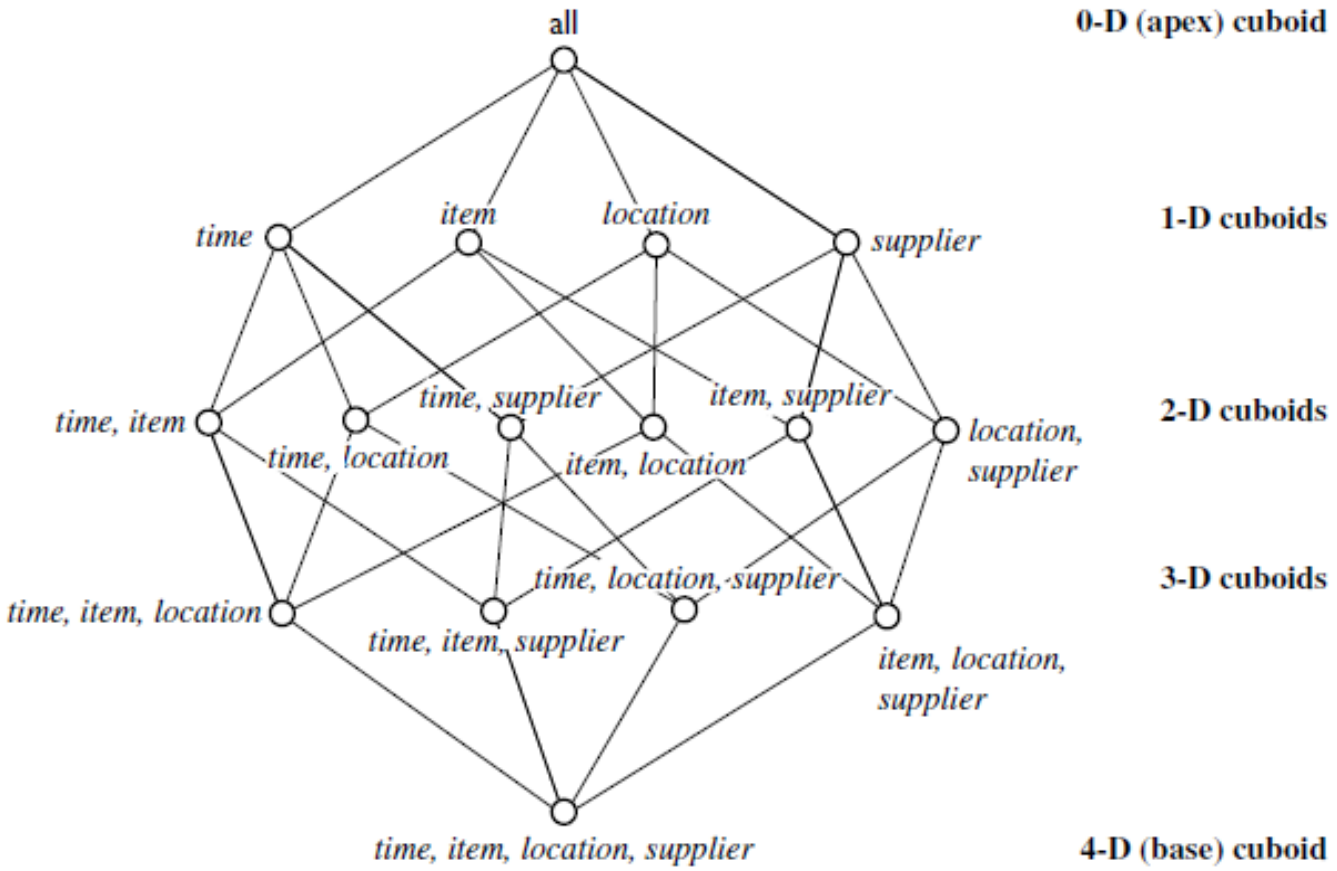
Data Warehouse Modeling: Data Cube



Data Warehouse Modeling: Data Cube

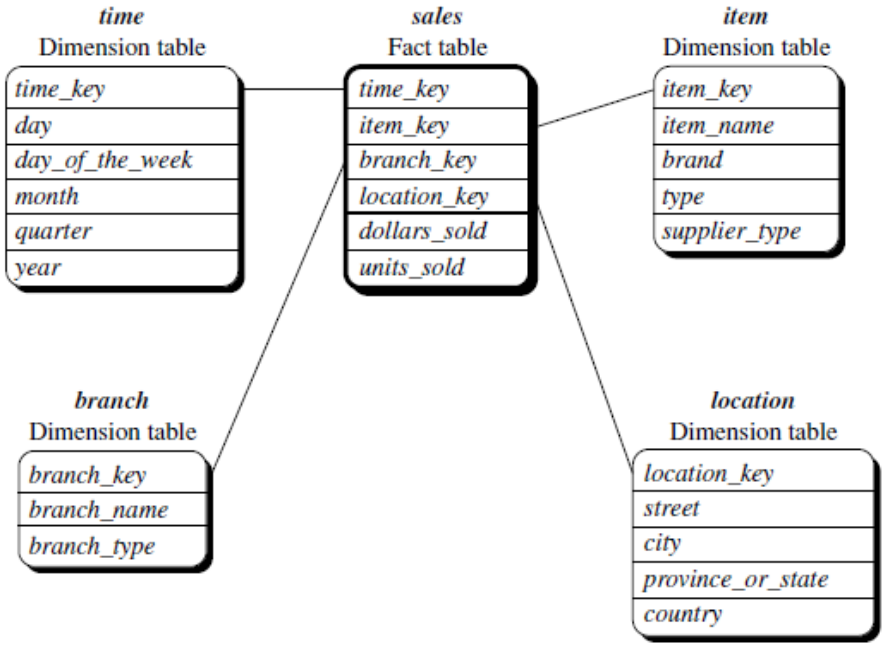


Data Warehouse Modeling: Data Cube



Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models

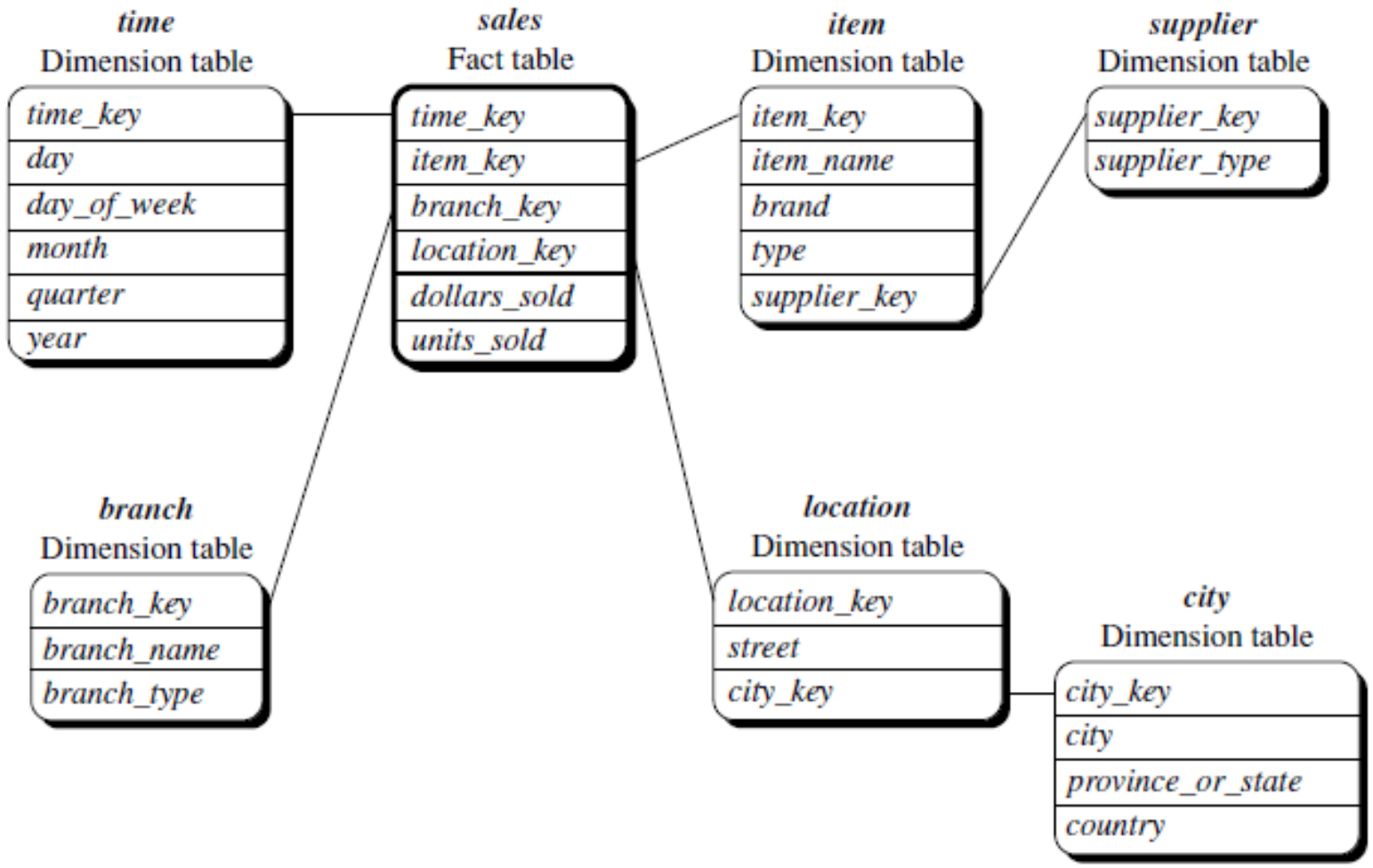
- Star schema: The most common modeling paradigm is the star schema, in which the data warehouse contains
  - a large central table (fact table) containing the bulk of the data, with no redundancy, and
  - a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.



Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models

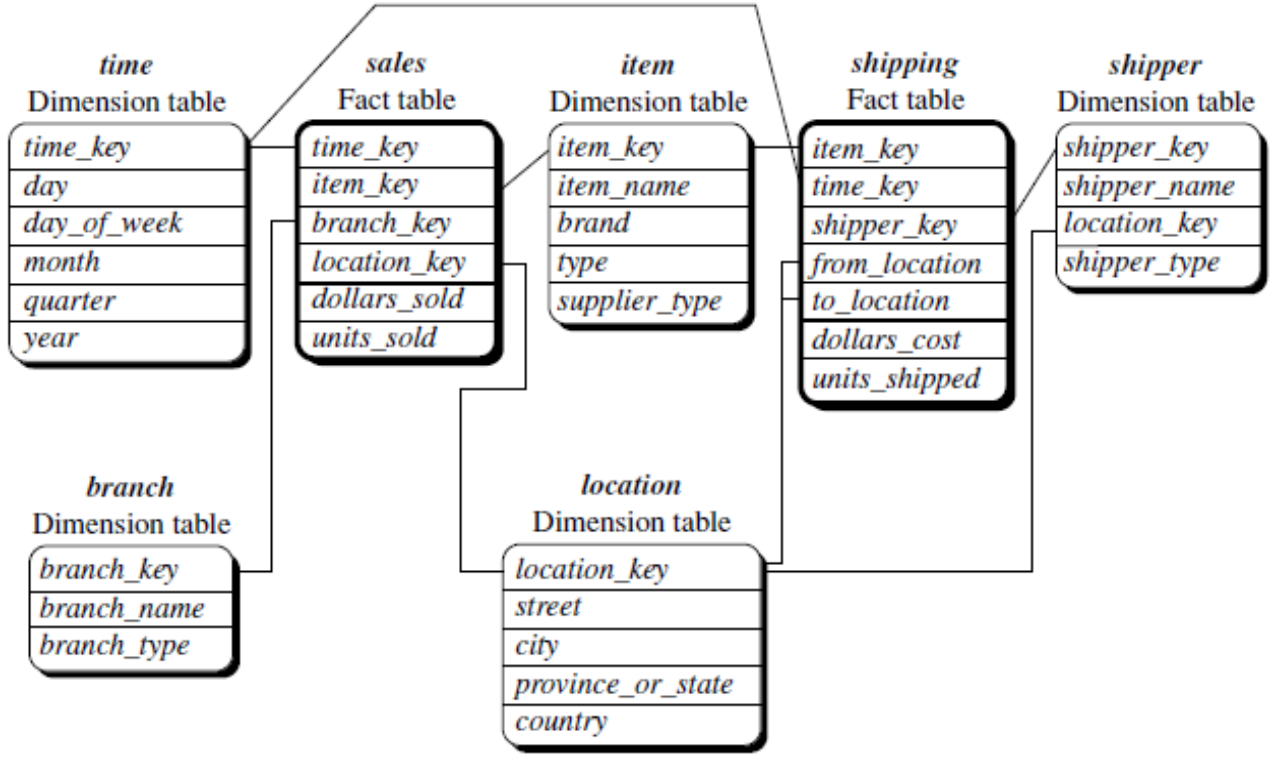
- Snowflake schema: The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.
- The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in 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.

Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models



Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models

- 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.



Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models

- In data warehousing, there is a distinction between a data warehouse and a data mart.
- A data warehouse
  - collects information about subjects that span the entire organization, such as customers, items, sales, assets, and personnel, and thus its scope is enterprise-wide.
  - For data warehouses, the fact constellation schema is commonly used, since it can model multiple, interrelated subjects.
- A data mart,
  - is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is departmentwide.
  - For data marts, the star or snowflake schema is commonly used, since both are geared toward modeling single subjects, although the star schema is more popular and efficient.

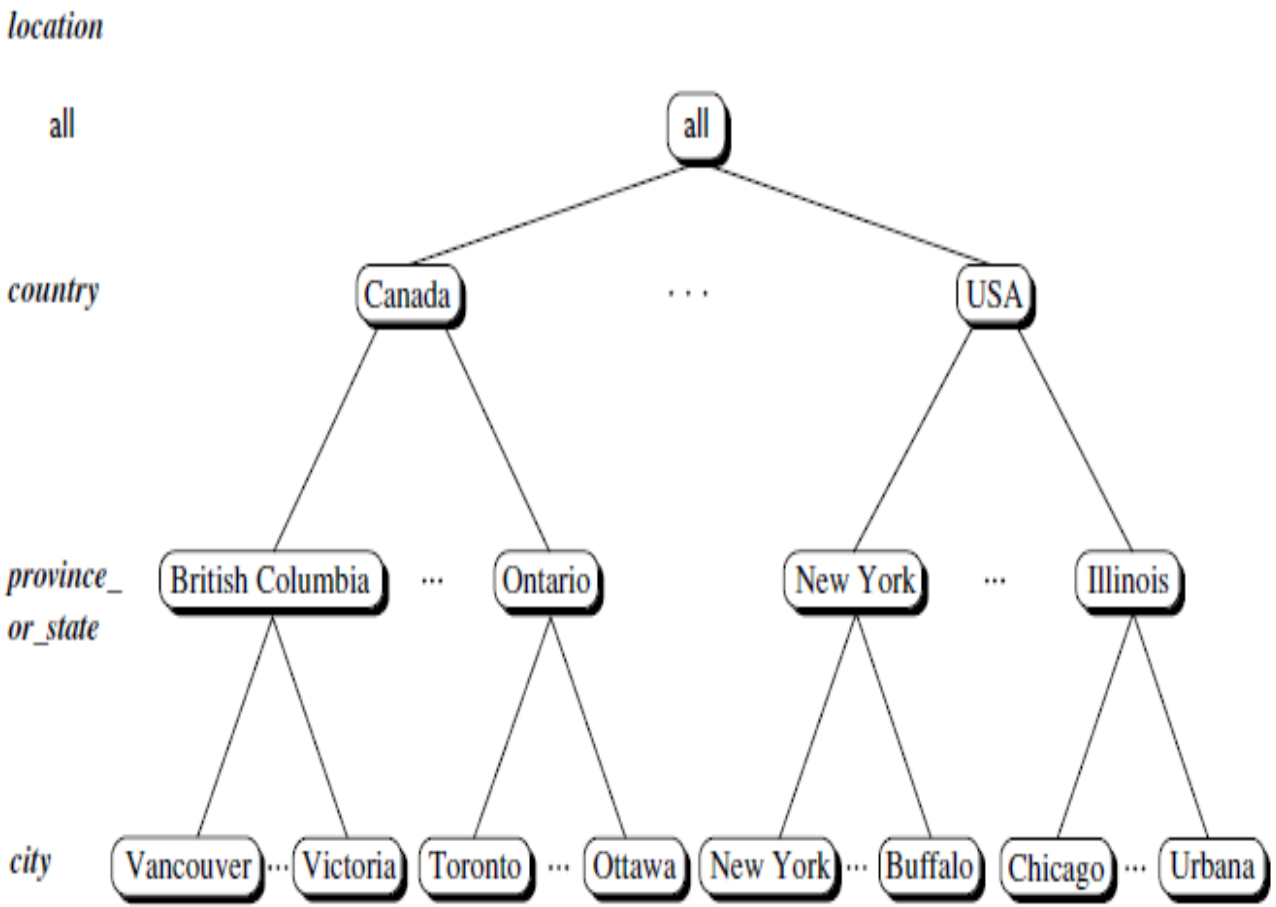
Dimensions: The Role of Concept Hierarchies

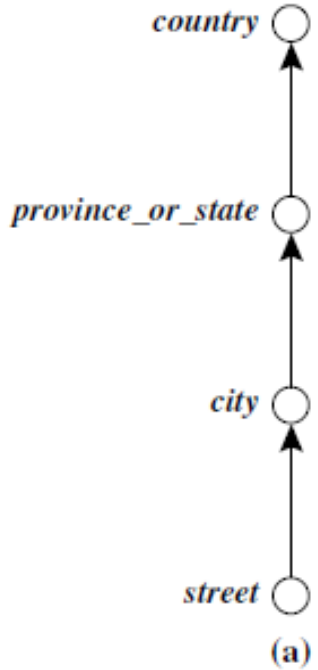
- A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.
- Consider a concept hierarchy for the dimension location. City values for location include Mangaluru, Bengaluru, Mumbai, and Chennai. Each city, however, can be mapped to the province or state to which it belongs. For example, Mangaluru can be mapped to Karnataka, and Mumbai to Maharashtra. The provinces and states can in turn be mapped to the country (e.g., India 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).
- Many concept hierarchies are implicit within the database schema. For example, suppose that the dimension location is described by the attributes number, street, city, province or state, zip code, and country. These attributes are related by a total order, forming a concept hierarchy such as “street < city < province or state < country.” Alternatively, the attributes of a dimension may be organized in a partial order, forming a lattice.

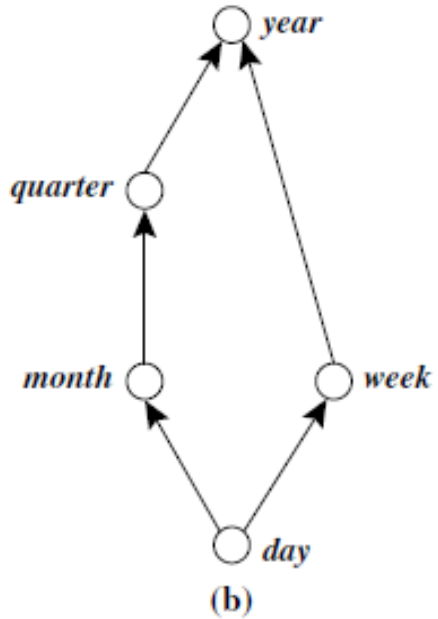
Dimensions: The Role of Concept Hierarchies

- Concept hierarchies that are common to many applications (e.g., for time) may be predefined in the data mining system. Data mining systems should provide users with the flexibility to tailor predefined hierarchies according to their particular needs. For example, users may want to define a fiscal year starting on April 1 or an academic year starting on September 1.
- Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a set-grouping hierarchy. A total or partial order can be defined among groups of values.
- 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.

Dimensions: The Role of Concept Hierarchies







Measures: Their Categorization and Computation

- 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—distributive, algebraic, and holistic— based on the kind of aggregate functions used.
- 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.
  - A measure is distributive if it is obtained by applying a distributive aggregate function. Distributive measures can be computed efficiently because of the way the computation can be partitioned.

Measures: Their Categorization and Computation

- 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.
  - 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 subaggregate. 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.

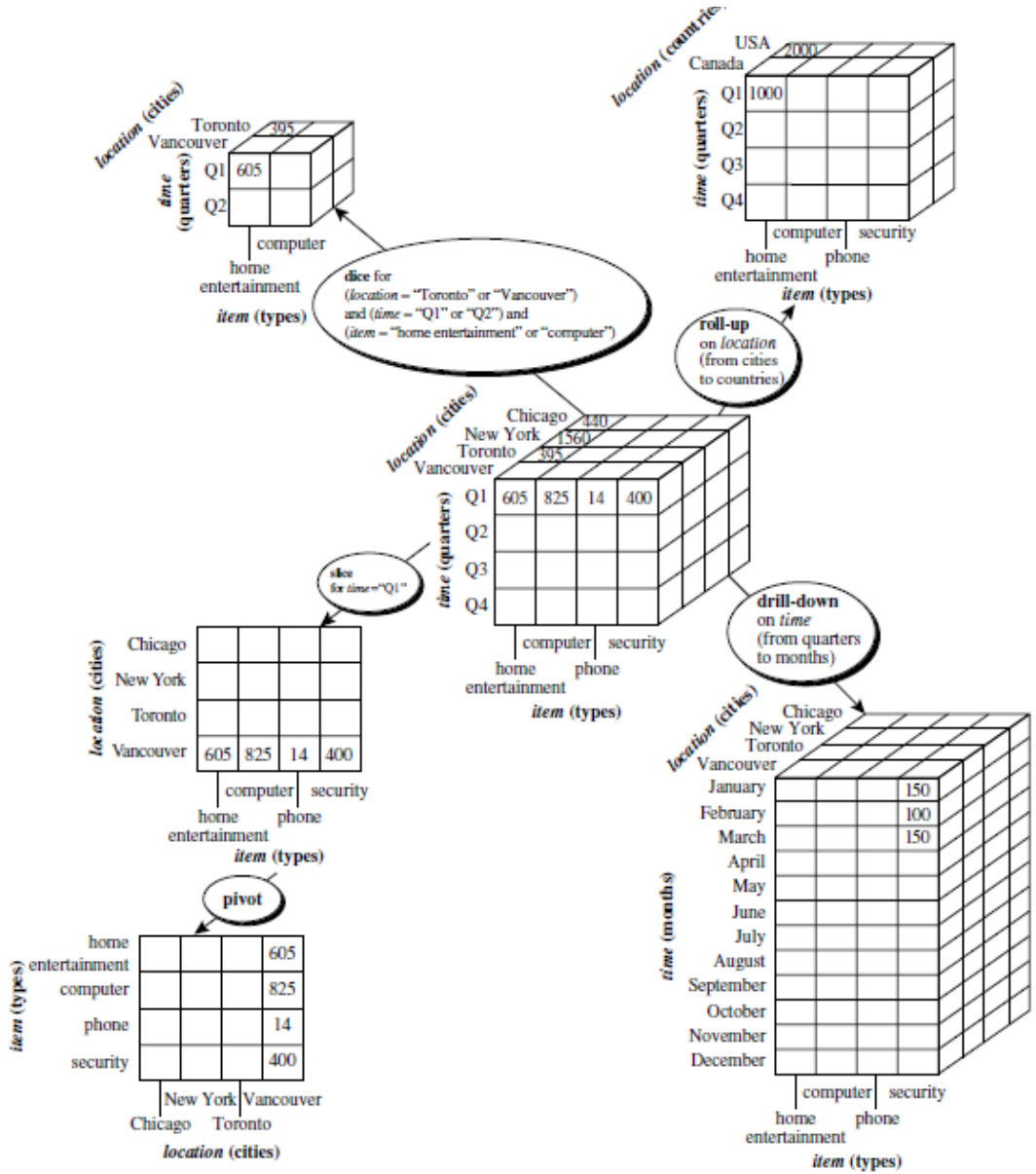
Typical OLAP Operations

- Roll-up:
  - The roll-up operation (or drill-up) performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction. The roll-up operation aggregates the data by ascending the location hierarchy from the level of city to the level of country. In other words, rather than grouping the data by city, the resulting cube groups the data by country.
  - When roll-up is performed by dimension reduction, one or more dimensions are removed from the given cube. For example, consider a sales data cube containing only the location and time dimensions. Roll-up may be performed by removing, say, the time dimension, resulting in an aggregation of the total sales by location, rather than by location and by time.
- Drill-down:
  - Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.
  - Drill-down occurs by descending the time hierarchy from the level of quarter to the more detailed level of month. The resulting data cube details the total sales per month rather than summarizing them by quarter.
  - Because a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube. For example, a drill-down can occur by introducing an additional dimension, such as customer group.

Typical OLAP Operations

- Slice and dice:
  - The slice operation performs a selection on one dimension of the given cube, resulting in a subcube.
  - The dice operation defines a subcube by performing a selection on two or more dimensions.
- Pivot (rotate): Pivot (also called rotate) is a visualization operation that rotates the data axes in view to provide an alternative data presentation. Other examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.
- Other OLAP operations: Some OLAP systems offer additional drilling operations. For example, drill-across executes queries involving (i.e., across) more than one fact table. The drill-through operation uses relational SQL facilities to drill through the bottom level of a data cube down to its back-end relational tables. Other OLAP operations may include ranking the top N or bottom N items in lists, as well as computing moving averages, growth rates, interests, internal return rates, depreciation, currency conversions, and statistical functions.

Typical OLAP Operations

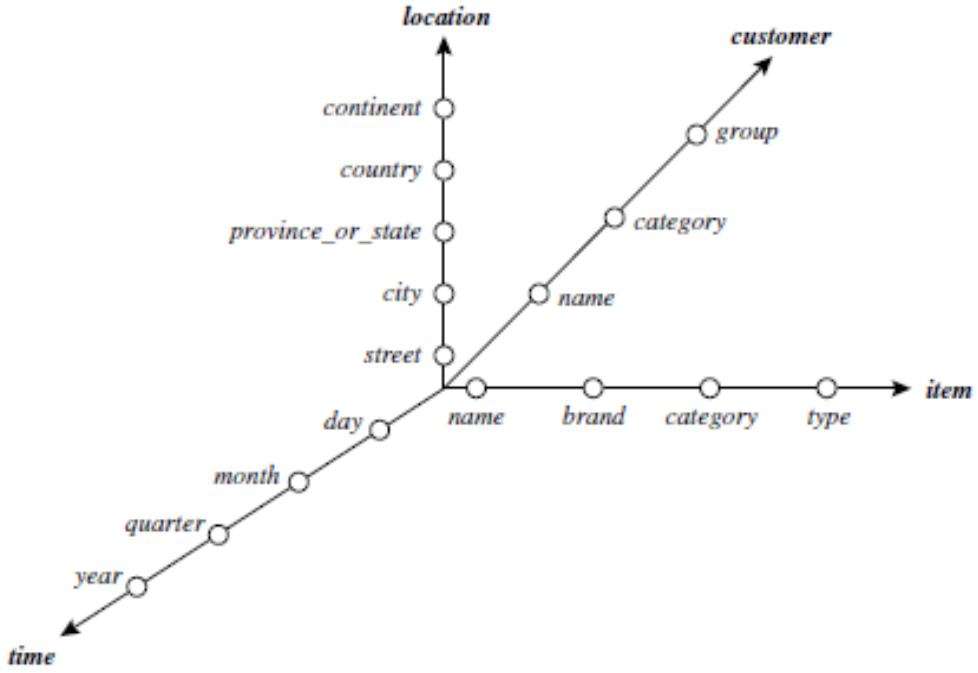


OLAP Systems versus Statistical Databases

- Many OLAP systems' characteristics (e.g., the use of a multidimensional data model and concept hierarchies, the association of measures with dimensions, and the notions of roll-up and drill-down) also exist in earlier work on statistical databases (SDBs).
- A statistical database is a database system that is designed to support statistical applications. Similarities between the two types of systems are rarely discussed, mainly due to differences in terminology and application domains.
- OLAP and SDB systems, however, have distinguishing differences. While SDBs tend to focus on socioeconomic applications, OLAP has been targeted for business applications.
- Privacy issues regarding concept hierarchies are a major concern for SDBs. For example, given summarized socioeconomic data, it is controversial to allow users to view the corresponding low-level data. Finally, unlike SDBs, OLAP systems are designed for efficiently handling huge amounts of data.

A Starnet Query Model for Querying Multidimensional Databases

- The querying of multidimensional databases can be based on a starnet model, which consists of radial lines emanating from a central point, where each line represents a concept hierarchy for a dimension. Each abstraction level in the hierarchy is called a footprint. These represent the granularities available for use by OLAP operations such as drill-down and roll-up.



Data Warehouse Design and Usage:
A Business Analysis Framework for Data Warehouse Design

- "What can business analysts gain from having a data warehouse?"
  - 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.
  - a data warehouse can enhance business productivity because it is able to quickly and efficiently gather information that accurately describes the organization.
  - 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.
  - a data warehouse may bring about cost reduction by tracking trends, patterns, and exceptions over long periods in a consistent and reliable manner.
- Four different views regarding a data warehouse design must be considered: the topdown view, the data source view, the data warehouse view, and the business query view.

Data Warehouse Design and Usage:
A Business Analysis Framework for Data Warehouse Design

- 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 precalculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- The business query view is the data perspective in the data warehouse from the end-user's viewpoint.

Data Warehouse Design and Usage:
A Business Analysis Framework for Data Warehouse Design

- 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 operational 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, to extrapolate trends based on history and look for anomalies or paradigm shifts, and to present coherent managerial recommendations based on such analysis.
- Program management skills involve the need to interface with many technologies, vendors, and end-users in order to deliver results in a timely and cost effective manner.

DataWarehouse Design Process

- 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 bottomup 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.

Data Warehouse Design Process

- From the software engineering point of view, the design and construction of a data warehouse may consist of the following steps:
  - planning, requirements study, problem analysis, warehouse design, data integration and testing, and finally deployment of the data warehouse.
- Large software systems can be developed using one of two methodologies: the waterfall method or the spiral method.
  - The waterfall method performs a structured and systematic analysis at each step before proceeding to the next. The spiral method involves the rapid generation of increasingly functional systems, with short intervals between successive releases. This is considered a good choice for data warehouse development, especially for data marts, because the turnaround time is short, modifications can be done quickly, and new designs and technologies can be adapted in a timely manner.

Data Warehouse Design Process

- In general, the warehouse design process consists of the following steps:
  - 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.
  - 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).
  - Choose the dimensions that will apply to each fact table record. Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
  - Choose the measures that will populate each fact table record. Typical measures are numeric additive quantities like dollars sold and units sold.

Data Warehouse Design Process

- 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.

Data Warehouse Usage for Information Processing

- Business executives use the data in data warehouses and data marts to perform data analysis and make strategic decisions.
- Data warehouses are used as an integral part of a plan-execute-assess “closed-loop” feedback system for enterprise management.
- Data warehouses are used in banking and financial services, consumer goods and retail distribution sectors, and controlled manufacturing such as demand-based production.
- Initially, the data warehouse is mainly used for generating reports and answering predefined queries. Progressively, it is used to analyze summarized and detailed data, where the results are presented in the form of reports and charts. Later, the data warehouse is used for strategic purposes, performing multidimensional analysis and sophisticated slice-and-dice operations.
- Finally, the data warehouse may be employed for knowledge discovery and strategic decision making using data mining tools.
- Business users need to have the means to know what exists in the data warehouse (through metadata), how to access the contents of the data warehouse, how to examine the contents using analysis tools, and how to present the results of such analysis.

Data Warehouse Usage for Information Processing

- 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.

From Online Analytical Processing to Multidimensional Data Mining

- 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. Multidimensional data mining is 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 accessing, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web accessing and service facilities, and reporting and OLAP analysis tools.
- 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, 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.

Data Warehouse 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.

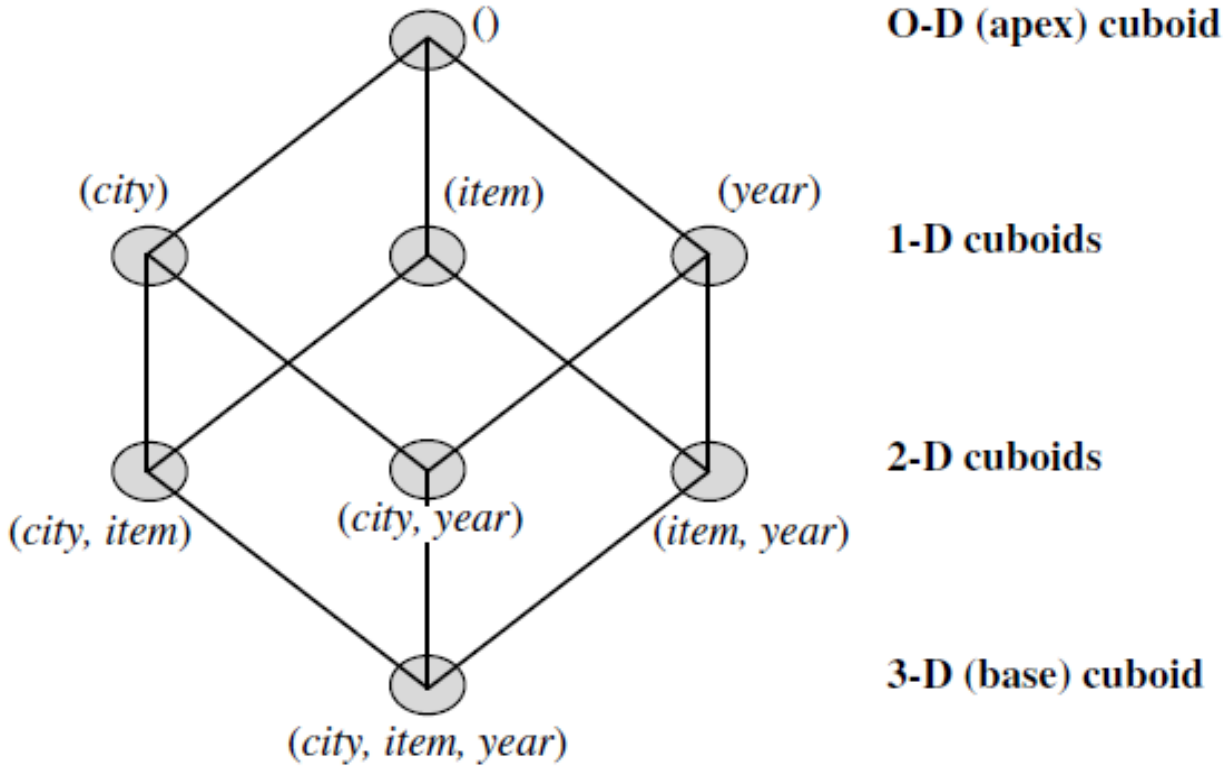
Efficient Data Cube Computation: An Overview

- 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.
- The compute cube Operator and the Curse of Dimensionality
  - One approach to cube computation extends SQL so as 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.

Efficient Data Cube Computation: An Overview

- A data cube is a lattice of cuboids. Let us create a data cube for sales that contains the following: city, item, year, and sales in dollars 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 groupby’s, that can be computed for this data cube is 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

Efficient Data Cube Computation: An Overview



Efficient Data Cube Computation: An Overview

- 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 () 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.

Efficient Data Cube Computation: An Overview

- 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.
- Precomputation leads to fast response time and avoids some redundant computation. Most, if not all, OLAP products resort to some degree of precomputation of multidimensional aggregates.
- A major challenge related to this precomputation, however, is that the required storage space may explode if all the cuboids in a data cube are precomputed, 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.

Efficient Data Cube Computation: An Overview

- “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 2^n.
- 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 that can be generated is $$\prod_{i=1}^{n}(L_i + 1)$$
- Where Li is the number of levels associated with dimension i. 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. it is unrealistic to precompute and materialize all of the cuboids that can possibly be generated for a data cube.

Partial Materialization: Selected Computation of Cuboids

- There are three choices for data cube materialization given a base cuboid:
  - No materialization: Do not precompute any of the “nonbase” cuboids. This leads to computing expensive multidimensional aggregates on-the-fly, which can be extremely slow.
  - 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 in order to store all of the precomputed cuboids.
  - 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.

Partial Materialization: Selected Computation of Cuboids

- The partial materialization of cuboids or subcubes should consider three factors:
  - identify the subset of cuboids or subcubes to materialize;
  - exploit the materialized cuboids or subcubes during query processing; and
  - 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. In addition, it should consider workload characteristics, the cost for incremental updates, and the total storage requirements. The selection must also consider the broad context of physical database design such as the generation and selection of indices. A popular approach is to materialize the cuboids set on which other frequently referenced cuboids are based. 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. Queries on additional combinations of the dimensions can be computed as needed.
- Once the selected cuboids have been materialized, it is important to take advantage of them during query processing. This involves several issues, such as how to determine the relevant cuboid(s) from among the candidate materialized cuboids, how to use available index structures on the materialized cuboids, and how to transform the OLAP operations onto the selected cuboid(s).
- Finally, during load and refresh, the materialized cuboids should be updated efficiently. Parallelism and incremental update techniques for this operation should be explored.

Indexing OLAP Data: Bitmap Index and Join Index

- 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, Bv, 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.
- 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.
- 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.

Indexing OLAP Data: Bitmap Index and Join Index

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

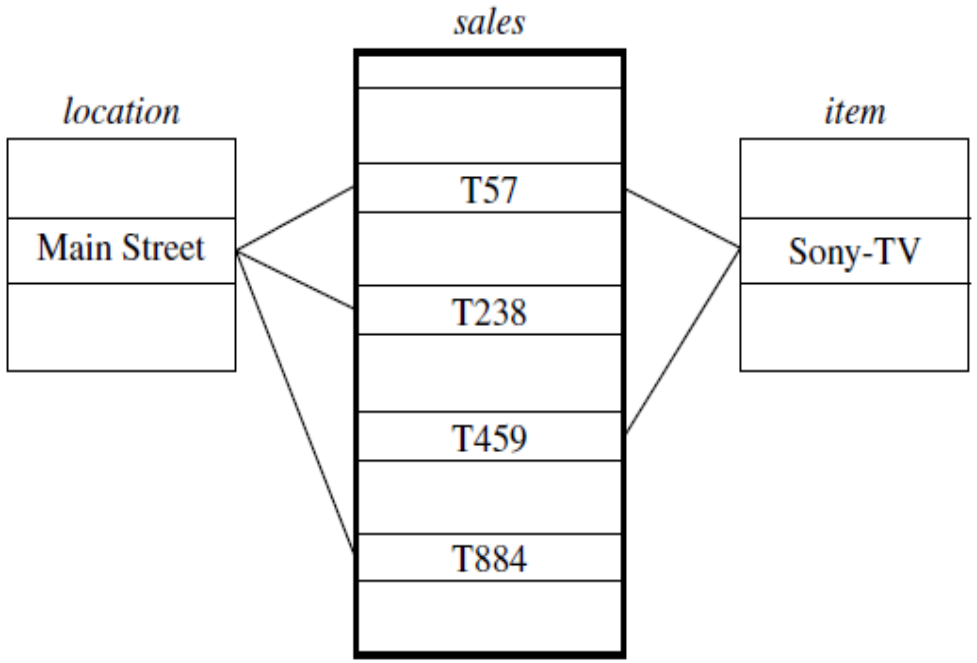
RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Note: H for "home entertainment," C for "computer," P for "phone," S for "security," V for "Vancouver," T for "Toronto."

Indexing OLAP Data: Bitmap Index and Join Index

- 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.
- The star schema model of data warehouses makes join indexing attractive for crosstable search, because the linkage between a fact table and its corresponding dimension tables comprises the fact table's foreign key and the dimension table's primary key. Join indexing maintains relationships between attribute values of a dimension (e.g., within a dimension table) and the corresponding rows in the fact table.
- Join indices may span multiple dimensions to form composite join indices. We can use join indices to identify subcubes that are of interest.

Indexing OLAP Data: Bitmap Index and Join Index



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
. . .	. . .	. . .

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:
  - 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 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.
  - 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.

Efficient Processing of OLAP Queries

- OLAP query processing. Suppose that we define a data cube 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 country is a more general concept than province or state. Cuboids 1, 3, and 4 can be used to process the query because
  - they have the same set or a superset of the dimensions in the query,
  - the selection clause in the query can imply the selection in the cuboid, and
  - the abstraction levels for the item and location dimensions in these cuboids are at a finer level than brand and province or state, respectively.

Efficient Processing of OLAP Queries

- “How would the costs of each cuboid compare if used to process the query?”
- It is likely that using cuboid 1 would 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.

OLAP Server Architectures: ROLAP versus MOLAP versus HOLAP

- 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. ROLAP servers include optimization for each DBMS back end, implementation of aggregation navigation logic, and additional tools and services. ROLAP technology tends to have greater scalability than MOLAP technology. The DSS server of Microstrategy, for example, adopts the ROLAP approach.
- Multidimensional OLAP (MOLAP) servers:
  - These servers support multidimensional data views through array-based multidimensional storage engines. They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to precomputed summarized data. Notice that with multidimensional data stores, the storage utilization may be low if the data set is sparse. In such cases, sparse matrix compression techniques should be explored.
  - Many MOLAP servers adopt a two-level storage representation to handle dense and sparse data sets: Denser subcubes are identified and stored as array structures, whereas sparse subcubes employ compression technology for efficient storage utilization.

OLAP Server Architectures: ROLAP versus MOLAP versus HOLAP

- 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. For example, a HOLAP server may allow large volumes of detailed data to be stored in a relational database, while aggregations are kept in a separate MOLAP store. The Microsoft SQL Server 2000 supports a hybrid OLAP server.
- Specialized SQL servers:
  - To meet the growing demand of OLAP processing in relational databases, 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.
- ROLAP uses relational tables to store data for online analytical processing. The base fact table stores data at the abstraction level indicated by the join keys in the schema for the given data cube. Aggregated data can also be stored in fact tables, referred to as summary fact tables.
- MOLAP uses multidimensional array structures to store data for online analytical processing.
- Most data warehouse systems adopt a client-server architecture. A relational data store always resides at the data warehouse/data mart server site. A multidimensional data store can reside at either the database server site or the client site.

OLAP Server Architectures: ROLAP versus MOLAP versus HOLAP

Single Table for Base and Summary Facts

RID	*item*	…	*day*	*month*	*quarter*	*year*	*dollars_sold*
1001	TV	…	15	10	Q4	2010	250.60
1002	TV	…	23	10	Q4	2010	175.00
…	…	…	…	…	…	…	…
5001	TV	…	all	10	Q4	2010	45,786.08
…	…	…	…	…	…	…	…