UNIT 1
DATA WAREHOUSE INTRODUCTION

Introduction: Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions.

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

The four keywords—*subject-oriented, integrated, time-variant*, and *non-volatile*—distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

1. **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 modelling and analysis of data for decision makers. Hence, data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process.
2. **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.
3. **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.
4. **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*.

Differences between Operational Database Systems(OLTP) and Data Warehouses(OLAP)

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. Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. These systems are known as **online analytical processing (OLAP)** systems. The major distinguishing features of OLTP and OLAP are summarized as follows:

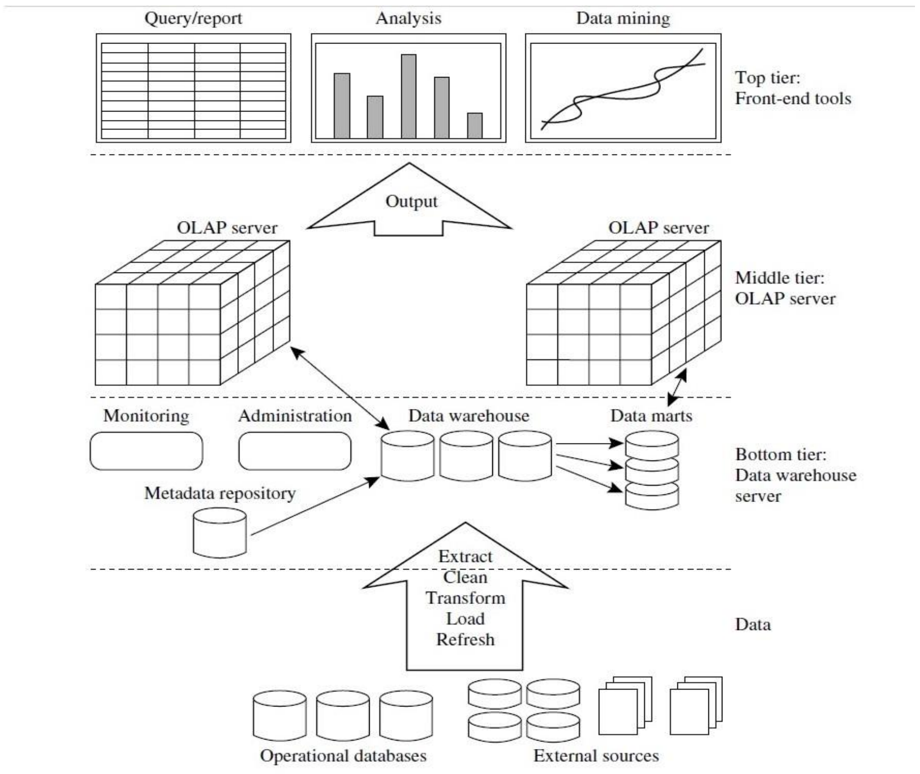
- **Users and system orientation**: An OLTP system is *customer-oriented* and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is *market-oriented* and is used for data analysis by knowledge workers, including managers, executives, and analysts.
- **Data contents**: An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. 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. These features make the data easier to use for informed decision making.
- **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 (see Section 4.2.2) 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.

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
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	>=TB

Data Warehousing: A Multitiered Architecture

Data warehouses often adopt a three-tier architecture



A three-tier data warehousing architecture.

1. The bottom tier is a **warehouse database server** that is almost always a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse.
2. The middle tier is an **OLAP server** that is typically implemented using either (1) a **Relational OLAP(ROLAP)** model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations); or (2) a **multidimensional OLAP (MOLAP)** model (i.e., a special-purpose server that directly implements multidimensional data and operations).
3. 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).

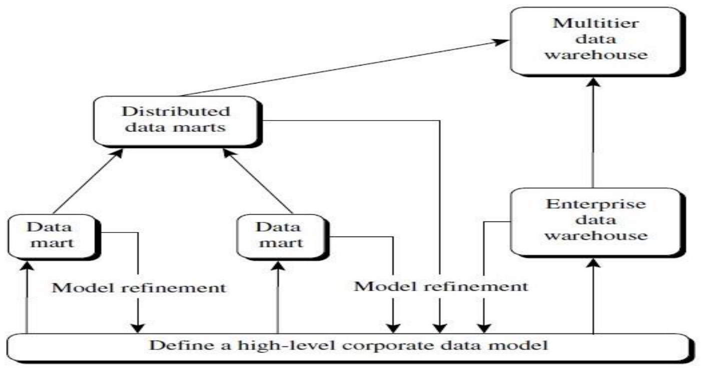
Data Warehouse Models: Enterprise Warehouse, Data Mart, and Virtual Warehouse

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.

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. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized. Data marts are usually implemented on low-cost departmental servers that are Unix/Linux or Windows based. The implementation cycle of a data mart is measured in weeks rather than months or years. *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.



A recommended approach for data warehouse development.

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.

Data Warehouse Modelling: Data Cube and OLAP

Data warehouses and OLAP tools are based on a **multidimensional data model**. This model views data in the form of a *data cube*.

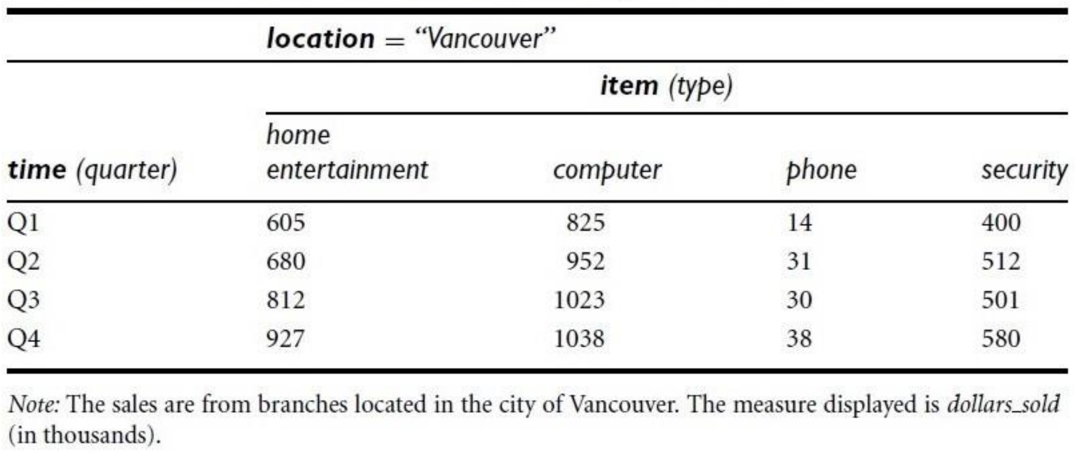
Data Cube: A Multidimensional Data Model

“What is a data cube?” A **data cube** allows data to be modelled and viewed in multiple dimensions. It is defined by dimensions and facts. In general terms, **dimensions** are the perspectives or entities with respect to which an organization wants to keep records. For example, *All Electronics* may create a *sales* data warehouse in order to keep records of the 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*.

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 analyse 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. 2-D representation, the sales for Vancouver are shown with respect to the *time* dimension (organized in quarters) and the *item* dimension (organized according to the types of items sold). The fact or measure displayed is *dollars sold* (in thousands). suppose that we would like to view the sales data with a third dimension. For instance, suppose we would like to view the data according to *time* and *item*, as well as *location*, for the cities Chicago, New York, Toronto, and Vancouver.

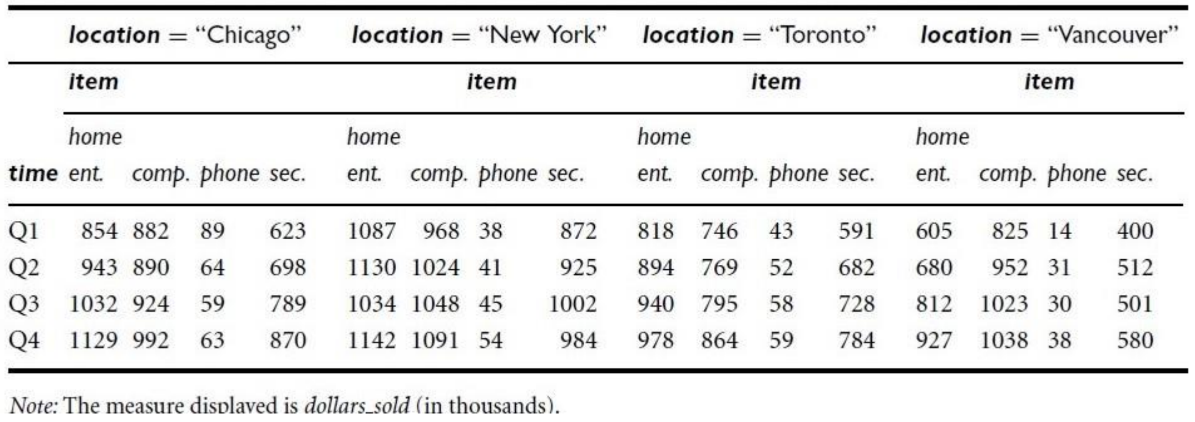
2-D View of Sales Data for *AllElectronics* According to *time* and *item*



	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 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

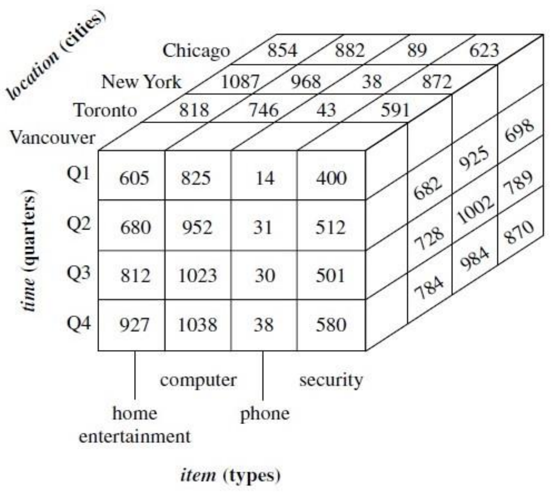


	location = "Chicago"				**location** = "New York"				**location** = "Toronto"				**location** = "Vancouver"			
	item				**item**				**item**				**item**			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*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

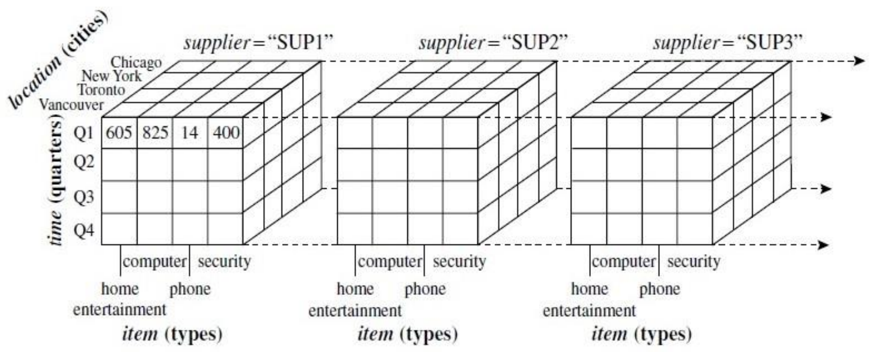
Note: The measure displayed is *dollars_sold* (in thousands).

Suppose that we would now like to view our sales data with an additional fourth dimension such as *supplier*. Viewing things in 4-D becomes tricky. The cuboid that holds the lowest level of summarization is called the **base cuboid**.

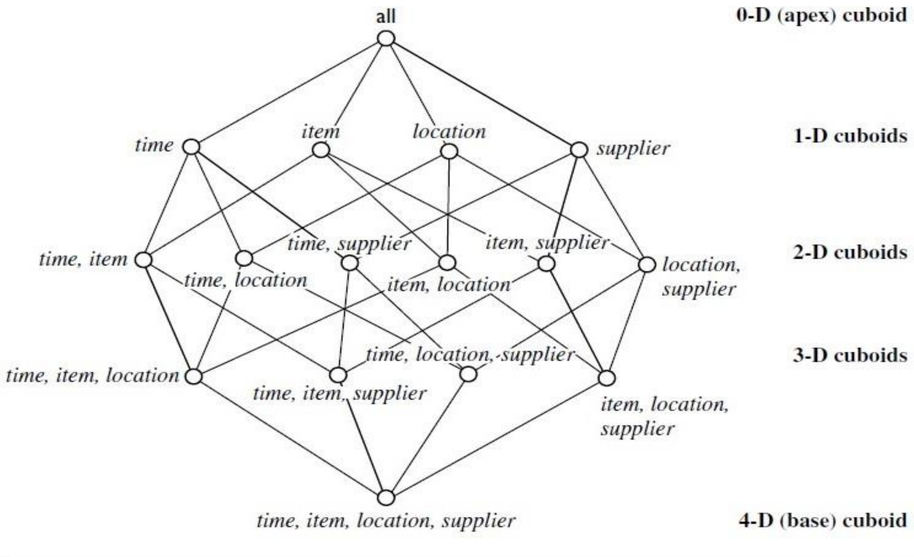
For example, the 4-D cuboid is the base cuboid for the given *time*, *item*, *location*, and *supplier* dimensions. 3-D (non-base) cuboid for *time*, *item*, and *location*, summarized for all suppliers.



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



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



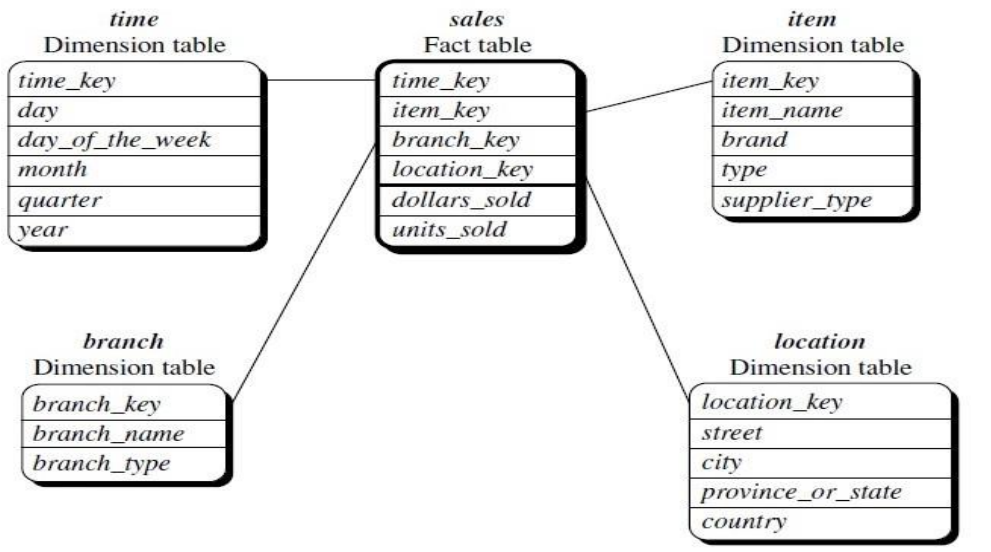
Lattice of cuboids, making up a 4-D data cube for *time*, *item*, *location*, and *supplier*. Each cuboid represents a different degree of summarization.

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

The entity-relationship data model is commonly used in the design of relational Databases. 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 or Galaxy schema.**

Star schema: The most common modelling 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.

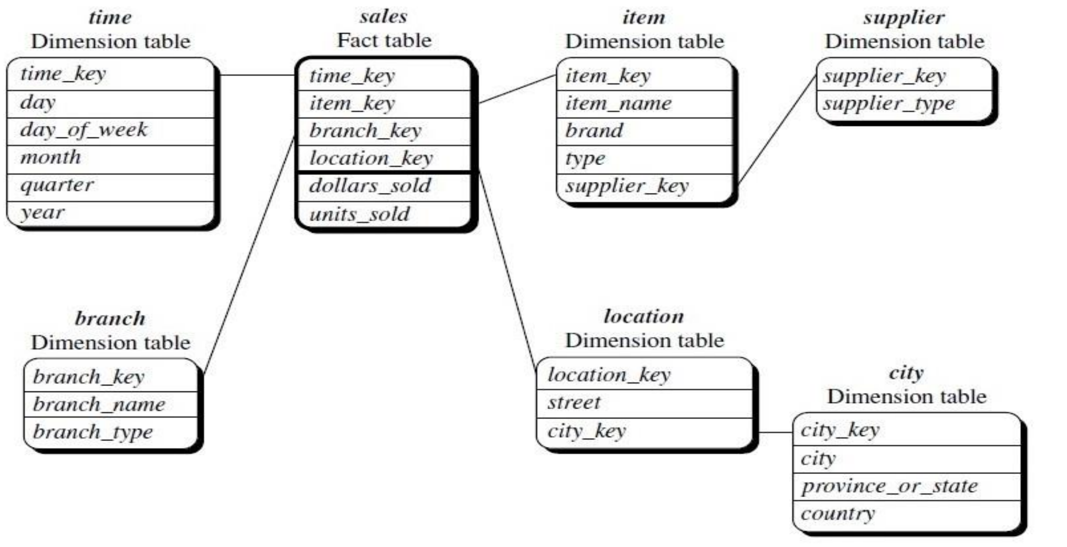
A star schema for *AllElectronics* sales is shown in Figure 4.6. Sales are considered along four dimensions: *time, item, branch*, and *location*. The schema contains a central fact table for *sales* that contains keys to each of the four dimensions, along with two measures: *dollars sold* and *units sold*. To minimize the size of the fact table, dimension identifiers (e.g., *time key* and *item key*) are system-generated identifiers.



Star schema of *sales* data warehouse.

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.

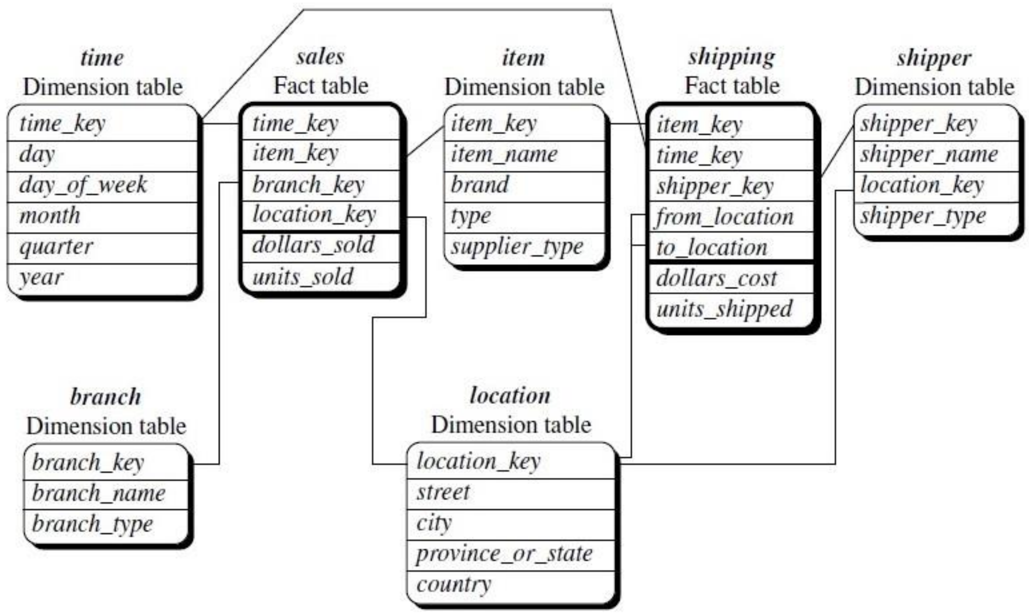
A snowflake schema for *AllElectronics* sales is given. The main difference between the two schemas is in the definition of dimension tables. The single dimension table for *item* in the star schema is normalized in the snowflake schema, resulting in new *item* and *supplier* tables. For example, the *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.



Snowflake schema of a *sales* data warehouse.

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**.

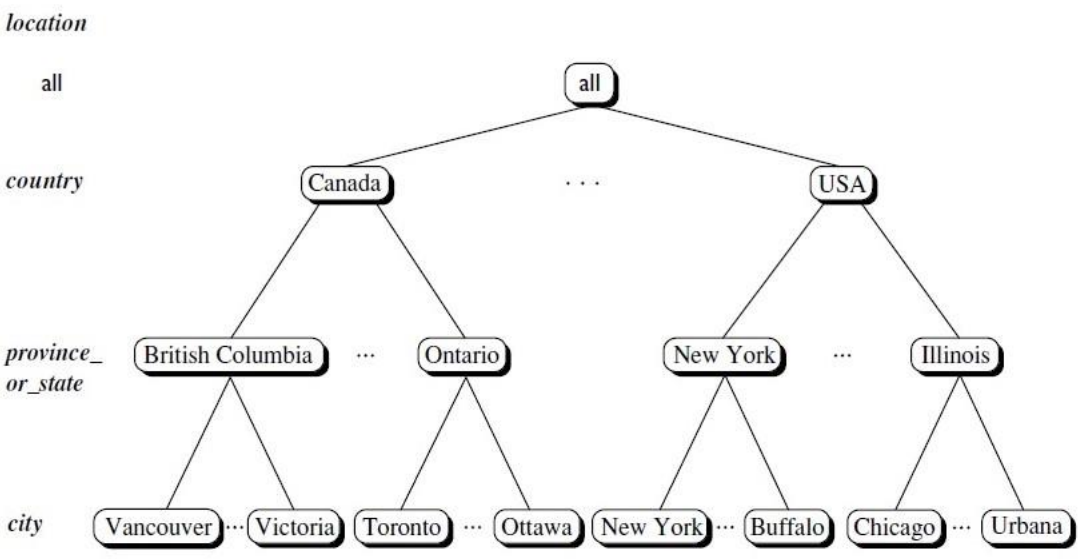
A fact constellation schema is shown in Figure. This schema specifies two fact tables, *sales* and *shipping*. The *sales* table definition is identical to that of the star schema. The *shipping* table has five dimensions, or keys—*item key, time key, shipper key, from location*, and *to location*—and 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.



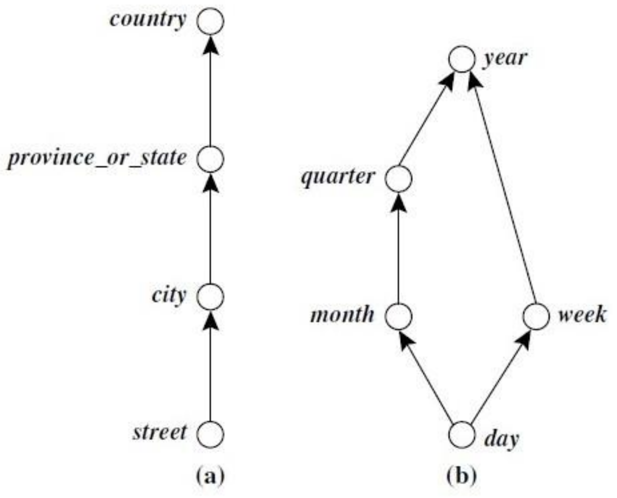
Fact constellation schema of a sales and shipping data warehouse.

Dimensions: The Role of Concept Hierarchies

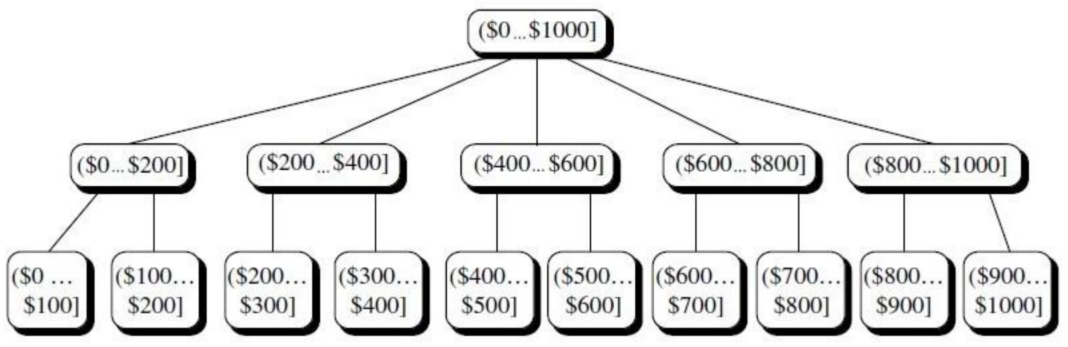
A **concept hierarchy** defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. Consider a concept hierarchy for the dimension *location*. City values for *location* include Vancouver, Toronto, New York, and Chicago.



A concept hierarchy for *location*. Due to space limitations, not all of the hierarchy nodes are shown, indicated by ellipses between nodes.



Hierarchical and lattice structures of attributes in warehouse dimensions: (a) a hierarchy for *location* and (b) a lattice for *time*.



A concept hierarchy for *price*.

A concept hierarchy that is a total or partial order among attributes in a database schema is called a **schema hierarchy**.

Typical OLAP Operations

In the multidimensional model, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. This organization provides users with the flexibility to view data from different perspectives. A number of OLAP data cube operations exist to materialize these different views, allowing interactive querying and analysis of the data at hand. Hence, OLAP provides a user-friendly environment for interactive data analysis.

Roll-up: The roll-up operation (also called the *drill-up* operation by some vendors) performs aggregation on a data cube, either by *climbing up a concept hierarchy* for a dimension or by *dimension reduction*. 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*. 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 on the central cube of Figure can occur by introducing an additional dimension, such as *customer group*.

Slice and dice: The *slice* operation perform a selection on one dimension of the given cube, resulting in a sub cube. Figure shows a slice operation where the sales data are selected from the central cube for the dimension *time* using the criterion *time* D "Q1." The *dice* operation defines a sub cube by performing a selection on two or more dimensions. Figure shows a dice operation on the central cube based on the following selection criteria that involve three dimensions: (*location* D "Toronto" or "Vancouver") and (*time* D "Q1" or "Q2") and (item D "home entertainment" or "computer").

Pivot (rotate): *Pivot* (also called *rotate*) is a visualization operation that rotates the data axes in view to provide an alternative data presentation. Figure shows a pivot operation where the *item* and *location* axes in a 2-D slice are rotated. Other examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

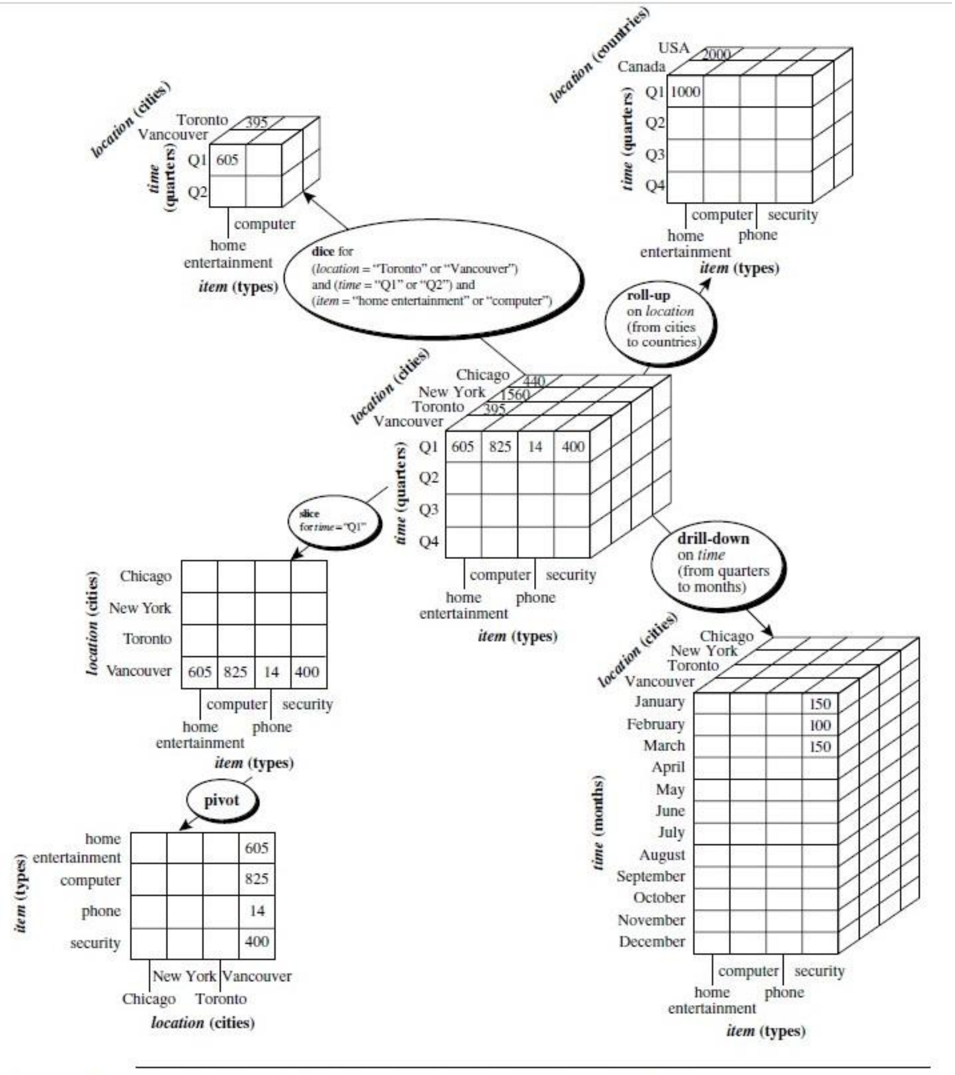


Figure 4.12 Examples of typical OLAP operations on multidimensional data.

Data Warehouse Design and Usage

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 modelled by traditional data modelling

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.
- Finally, the **business query view** is the data perspective in the data warehouse from the end-user's viewpoint.

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.

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.

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. One approach to cube computation extends SQL so as to include a compute cube operator. The compute cube operator computes aggregate over all subsets of the dimensions specified in the operation.

Example: A data cube is a lattice of cuboids. Suppose that you want to create a data cube for *AllElectronics* sales that contains the following: *city, item, year*, and *sales in dollars*. You want to be able to analyse the data, with queries such as the following:

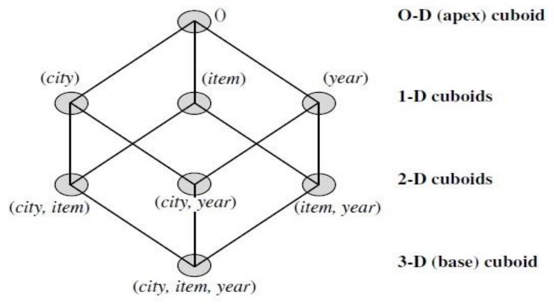
"*Compute the sum of sales, grouping by city and item.*"

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

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

Total number of cuboids, or group-by's, that can be computed for this data cube is 2^n ($2^3=8$). The possible group-by's are the following: {(*city, item, year*), (*city, item*), (*city, year*), (*item, year*), (*city*), (*item*), (*year*)}.

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.



Lattice of cuboids, making up a 3-D data cube. Each cuboid represents a different group-by. The base cuboid contains *city, item*, and *year* dimensions.

An SQL query containing no group-by (e.g., "*compute the sum of total sales*") is a *zero-dimensional operation*. An SQL query containing one group-by (e.g., *"compute the sum of sales, group-by city*") is a *one-dimensional operation*. the cube operator is the *n*-dimensional generalization of the group-by operator. Similar to the SQL syntax, the data cube could be defined as define cube sales cube [city, item, year]: sum (sales in dollars)

Total number of cuboids=$\prod_{i=0}^{n}(L_i+1)$ with dimension *i*. One is added to *Li.*