


# Chapter 4: Data Warehousing and On-line Analytical Processing

---

- ❑ Data Warehouse: Basic Concepts 
- ❑ Data Warehouse Modeling: Data Cube and OLAP
- ❑ Data Warehouse Design and Usage
- ❑ Data Warehouse Implementation
- ❑ Summary

# What is a Data Warehouse?

---

- ❑ Defined in many different ways, but not rigorously
  - ❑ A decision support database that is maintained **separately** from the organization's operational database
  - ❑ Support **information processing** by providing a solid platform of consolidated, historical data for analysis
- ❑ “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
- ❑ Data warehousing:
  - ❑ The process of constructing and using data warehouses
- ❑ Data mining tools often access data warehouses rather than operational data.

# Data Warehouse—Subject-Oriented

---

- ❑ Organized around major subjects, such as **customer, product, sales**
- ❑ Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing of an organization
- ❑ Provide **a simple and concise** view around particular subject issues by **excluding data that are not useful in the decision support process**

# Data Warehouse—Integrated

---

- ❑ Constructed by integrating multiple, heterogeneous data sources
  - ❑ relational databases, flat files, on-line transaction records
- ❑ Data cleaning and data integration techniques are applied
  - ❑ ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
  - ❑ Ex. Hotel price: differences on currency, tax, breakfast covered, and parking
  - ❑ When data is moved to the warehouse, it is converted

# Data Warehouse—Time Variant

---

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational data: current (day-to-day) data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain “time element”

# Data Warehouse—Nonvolatile

---

- Independence
  - A **physically separate store** of data transformed from the application data found in the operational environment
- Static: Operational **update of data does not occur** in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*

# Data Warehouse vs. Heterogeneous DBMS

---

- ❑ Heterogeneous database: Query driven approach
  - ❑ Traditional database approach to heterogeneous DB integration is to build wrappers and integrators (mediators) on top of heterogeneous databases
  - ❑ When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, queries are then mapped and sent to local query processors and the results returned from the different sites are integrated into a global answer set.
  - ❑ Requires complex information filtering and integration process and compete with local sites for processing resources

# Data Warehouse vs. Heterogeneous DBMS

---

- Data warehouse: update-driven, high performance
  - Information from multiple, heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis
  - 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.
  - 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.



# Data Warehouse vs. Operational DBMS

---

- ❑ Operational Database System, OLTP (On-line transaction processing) system
  - ❑ The major task of online operational database systems is to perform online transaction and query processing
  - ❑ Covers most of the day-to-day operations of an organization such as purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- ❑ Data Warehouse, OLAP (On-line analytical processing) system
  - ❑ Serve users or knowledge workers ( e.g., managers, analysts, and executives) in the role of data analysis and decision making
  - ❑ Major task of data warehouse system (drilling, slicing, dicing, pivoting etc.)
  - ❑ Organize and present data in various formats in order to accommodate the diverse needs of different users

# Data Warehouse vs. Operational DBMS

---

- ❑ 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 and a *subject-oriented* database design.

# Data Warehouse vs. Operational DBMS

---

- ❑ **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.
- ❑ Other features that distinguish between OLTP and OLAP systems include database size, frequency of operations, and performance metrics.

# Comparison of OLTP and OLAP System

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	≥ TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

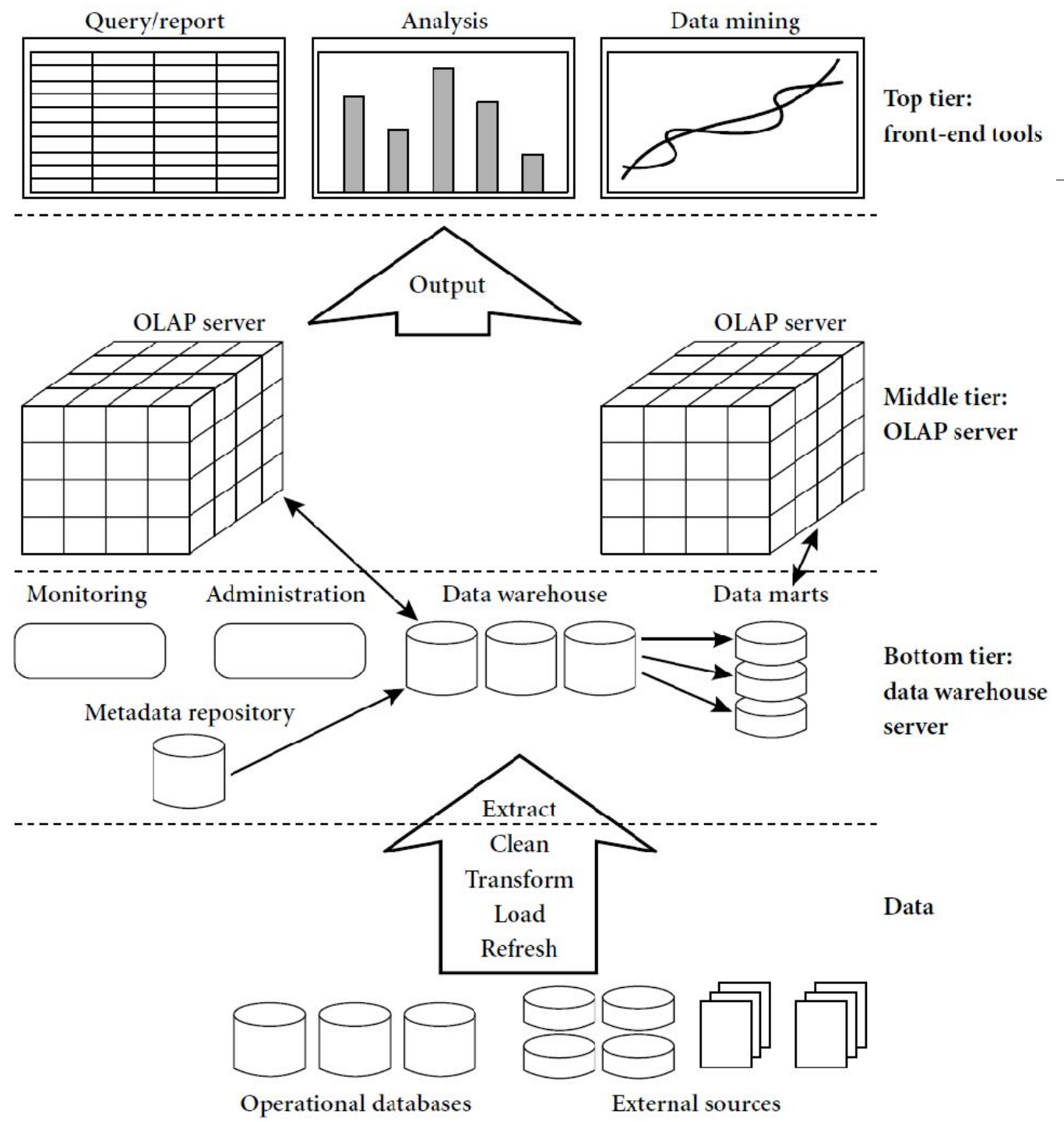
# Why a Separate Data Warehouse?

---

- ❑ High performance for both systems
  - ❑ DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - ❑ Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- ❑ Different functions and different data:
  - ❑ [missing data](#): Decision support requires historical data which operational DBs do not typically maintain
  - ❑ [data consolidation](#): DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - ❑ [data quality](#): different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- ❑ Note: There are more and more systems which perform OLAP analysis directly on relational databases

# Data Warehouse: A Multi-Tiered Architecture

- ❑ Top Tier: Front-End Tools
- ❑ Middle Tier: OLAP Server
- ❑ Bottom Tier: Data Warehouse Server
- ❑ Data





# Data Warehouse Models

---

- ❑ From the architecture point of view, there are three data warehouse models:
- ❑ **Enterprise warehouse**
  - ❑ An enterprise warehouse collects all of the information about subjects spanning the entire organization.
  - ❑ 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 superservers, or parallel architecture platforms.
  - ❑ It requires extensive business modeling and may take years to design and build.

# Data Warehouse Models

---

## ❑ Data Mart

- ❑ Data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. 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 more likely to be measured in weeks rather than months or years.
- ❑ However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.
- ❑ Depending on the source of data, data marts can be categorized as independent or dependent.



# Data Warehouse Models

---

- ❑ *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 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.

# Extraction, Transformation, and Loading (ETL)

---

- ❑ 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.
- ❑ Besides cleaning, loading, refreshing, and metadata definition tools, data warehouse systems usually provide a good set of data warehouse management tools.

# Metadata Repository

---

- ❑ **Metadata** are data about data. When used in a data warehouse, metadata are the data that define warehouse objects. A metadata repository should contain the following:
  - ❑ A description of the *data warehouse structure*, 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 (history of migrated data and the sequence of transformations applied to it), 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.

# Metadata Repository

---

- ❑ *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.

# Chapter 4: Data Warehousing and On-line Analytical Processing

---

- ❑ Data Warehouse: Basic Concepts
- ❑ Data Warehouse Modeling: Data Cube and OLAP
- ❑ Data Warehouse Design and Usage
- ❑ Data Warehouse Implementation
- ❑ Summary



# From Tables and Spreadsheets to Data Cubes

---

- ❑ A **data warehouse** is based on a multidimensional data model which views data in the form of a **data cube**. 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, An organization 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*.
- ❑ Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.

# From Tables and Spreadsheets to Data Cubes

---

- ❑ A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions.
  - ❑ **Dimension tables**, such as item (item\_name, brand, type), or time(day, week, month, quarter, year)
  - ❑ **Fact table** contains **measures** (such as dollars\_sold) and keys to each of the related dimension tables
- ❑ **Data cube**: A lattice of cuboids
  - ❑ In data warehousing literature, an n-D base cube is called a **base cuboid**
  - ❑ The top most 0-D cuboid, which holds the highest-level of summarization, is called the **apex cuboid**
  - ❑ The lattice of cuboids forms a **data cube**.

# From Tables and Spreadsheets to Data Cubes

**Table 4.2** 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

<i>location</i> = "Vancouver"				
<i>time</i> (quarter)	<i>item</i> (type)			
	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).



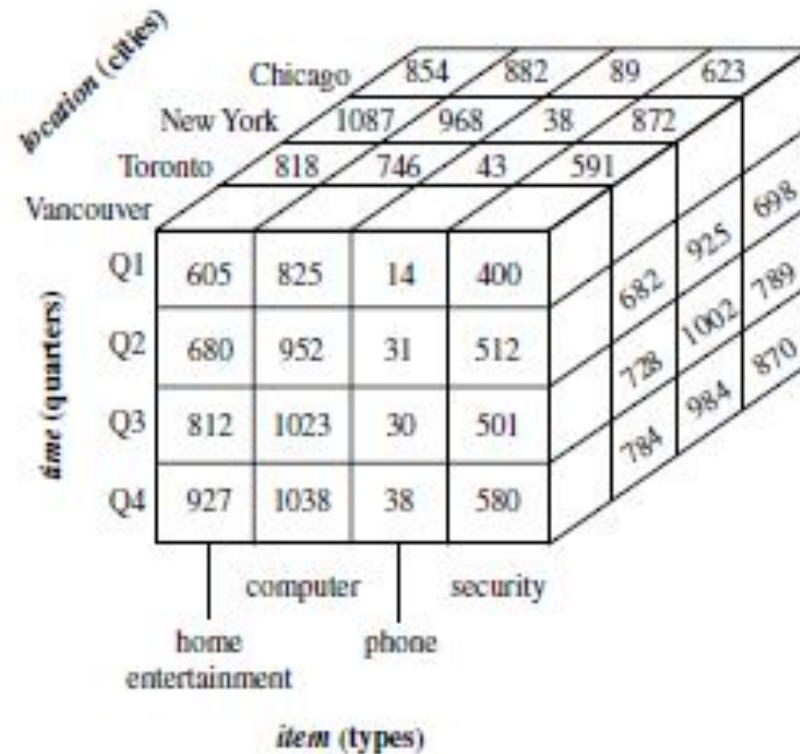
# From Tables and Spreadsheets to Data Cubes

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

<i>location</i> = "Chicago"					<i>location</i> = "New York"				<i>location</i> = "Toronto"				<i>location</i> = "Vancouver"			
<i>Item</i>					<i>Item</i>				<i>Item</i>				<i>Item</i>			
<i>time</i>	<i>home</i>				<i>home</i>				<i>home</i>				<i>home</i>			
	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

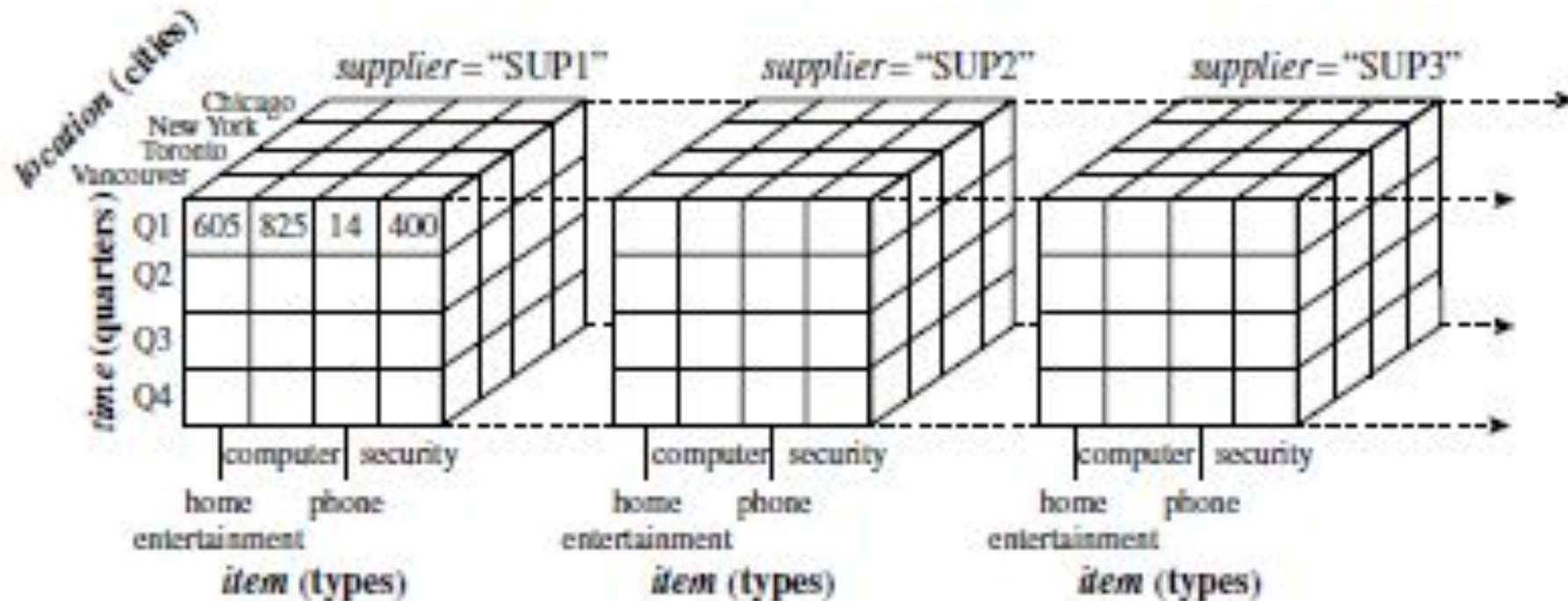
Note: The measure displayed is *dollars\_sold* (in thousands).

# From Tables and Spreadsheets to Data Cubes



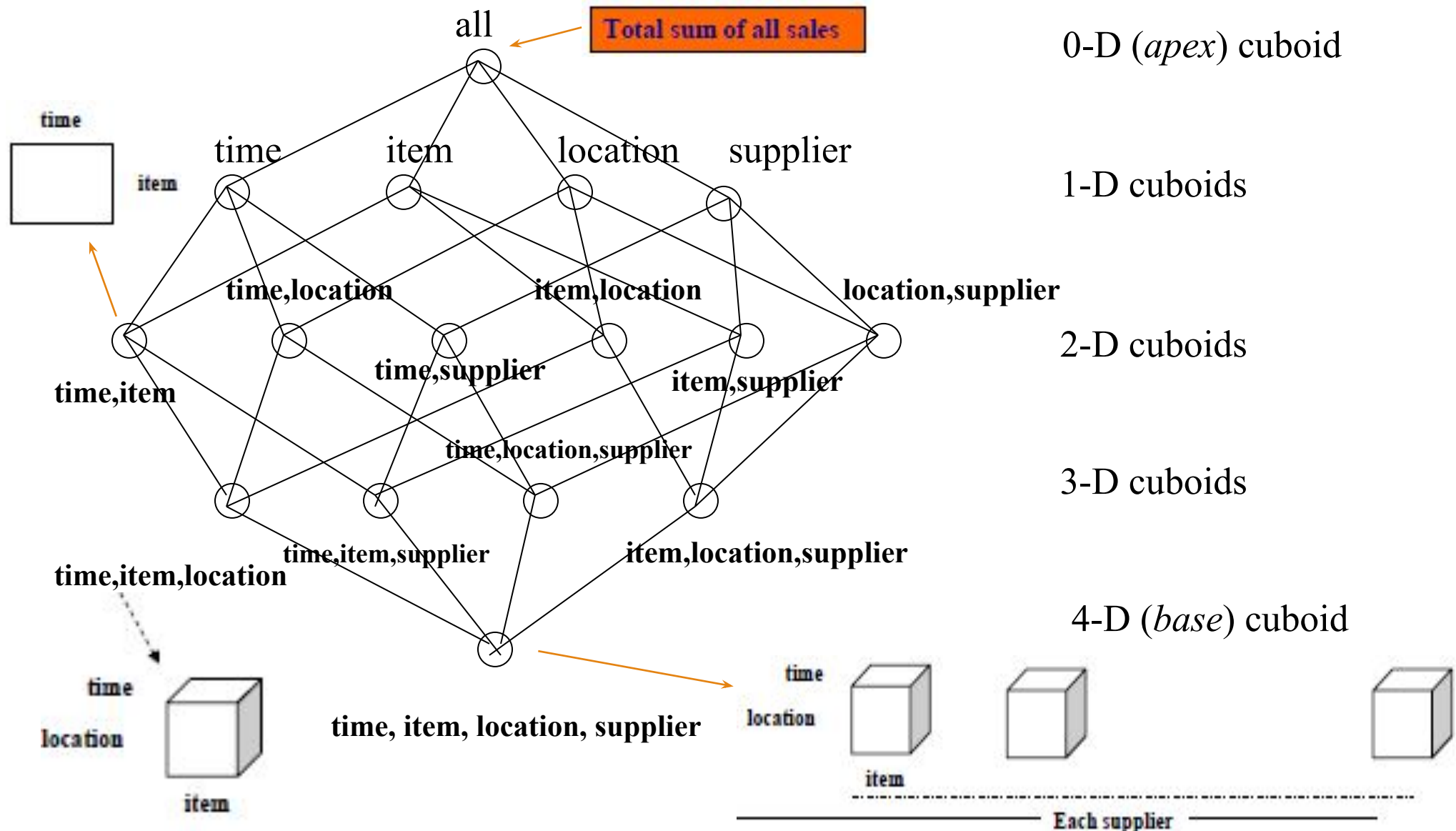
**Figure 4.3** A 3-D data cube representation of the data in Table 4.3, according to *time*, *item*, and *location*. The measure displayed is *dollars\_sold* (in thousands).

# From Tables and Spreadsheets to Data Cubes



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

# Data Cube: A Lattice of Cuboids

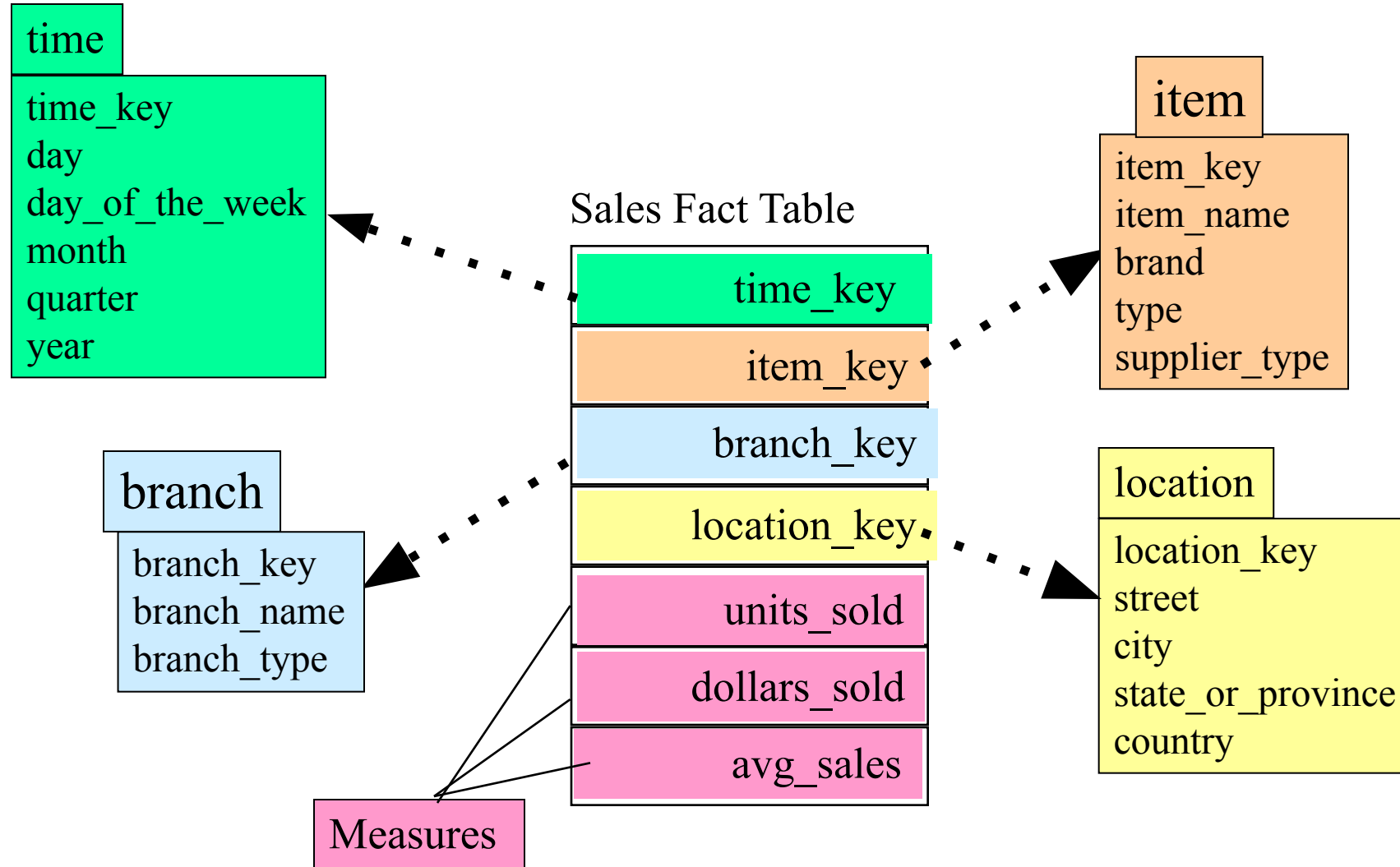


# Schemas for Multidimensional Data Models

---

- ❑ A data warehouse requires a concise, subject-oriented schema that facilitates online data analysis.
- ❑ The most popular data model for a data warehouse is a **multidimensional model**, which can exist in the form of a **star schema**, a **snowflake schema**, or a **fact constellation schema**.
- ❑ Star schema: In star schema data warehouse contains
  - ❑ (1) a large central table (**fact table**) containing the bulk of the data, with no redundancy, and
  - ❑ (2) a set of smaller attendant tables (**dimension tables**), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.
  - ❑ Moreover, the attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).

# Star Schema: An Example



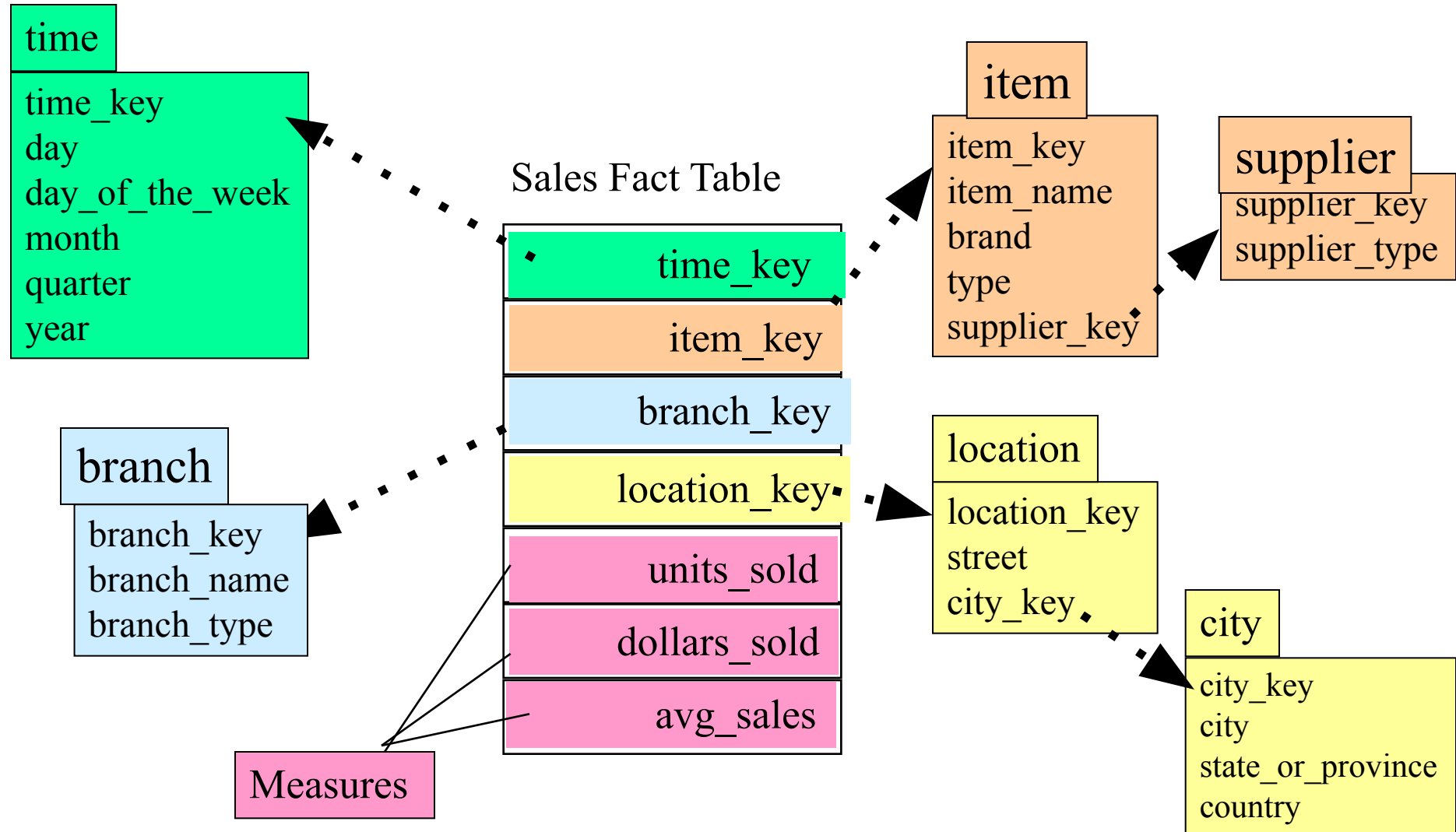


# Schemas for Multidimensional Data Models

---

- ❑ 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.

# Snowflake Schema: An Example



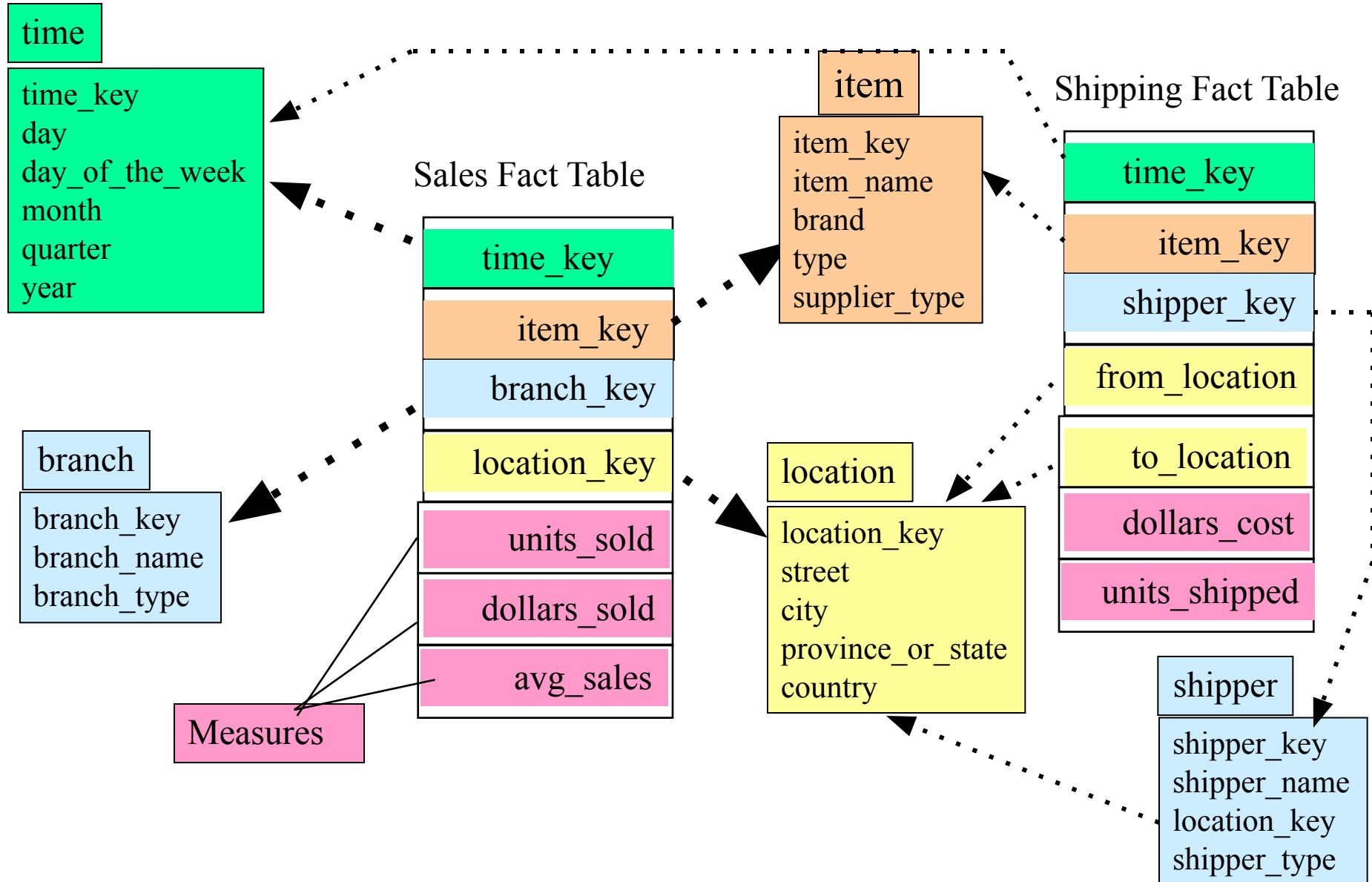


# Schemas for Multidimensional Data Models

---

- ❑ Fact constellation: Sophisticated applications may require multiple fact tables to *share* dimension tables.
- ❑ 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.
- ❑ This kind of schema can be viewed as a collection of stars, and hence is called a **galaxy schema** or a **fact constellation**.

# Fact Constellation: An Example



# Warehouse Model Vs. Schema

---

- ❑ 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**, on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is *department-wide*.
- ❑ 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.

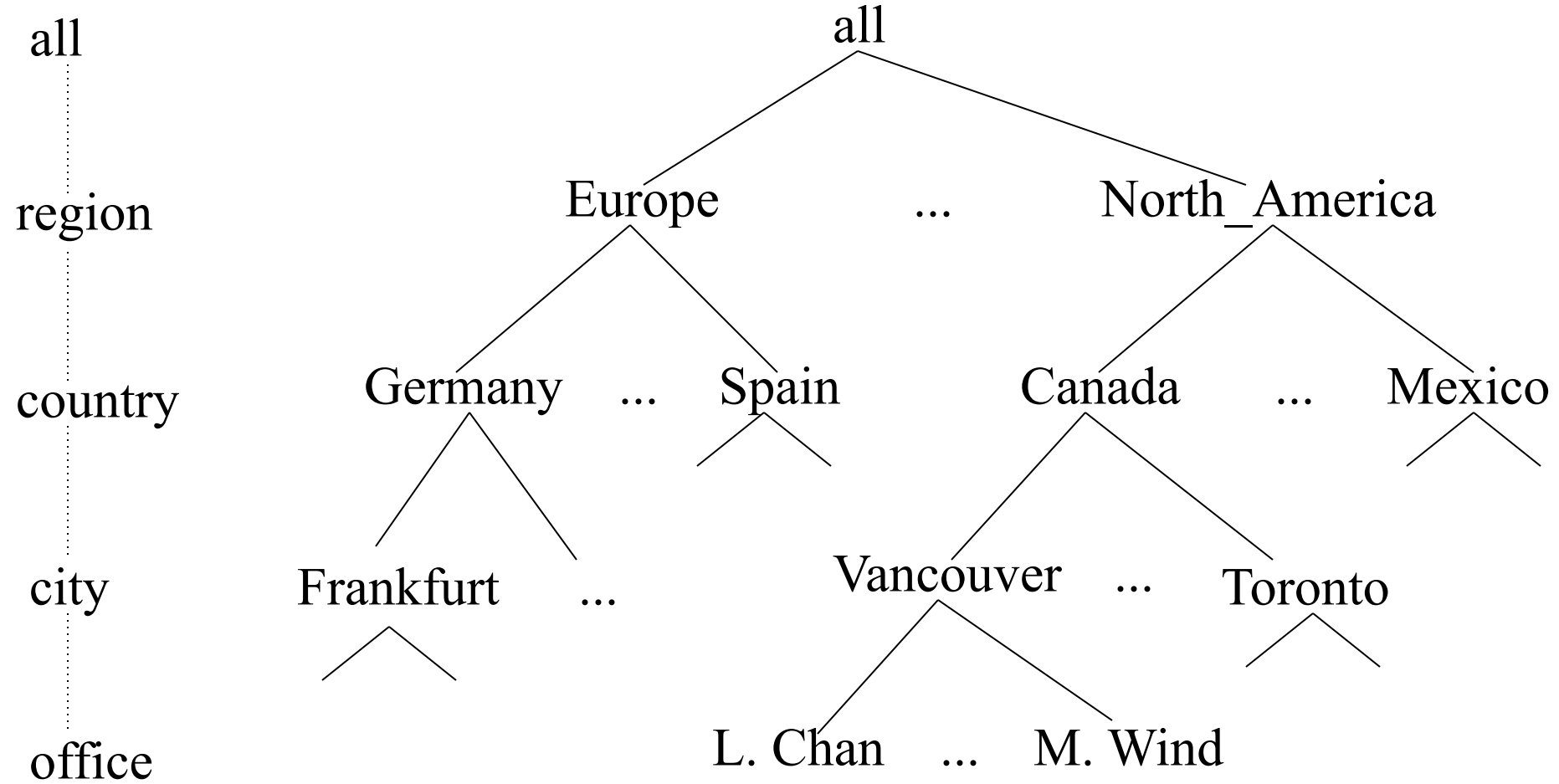
# Concept Hierarchies

---

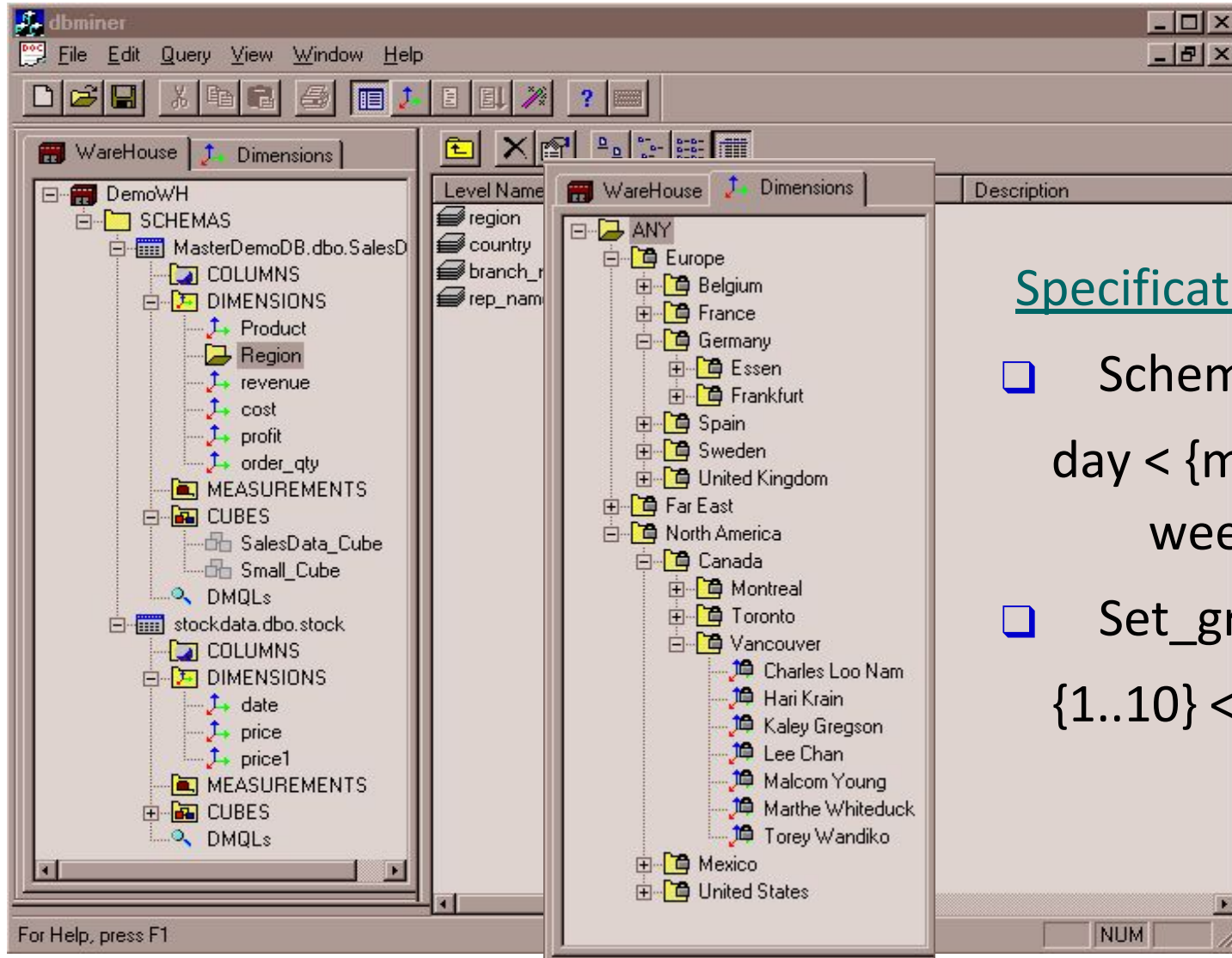
- ❑ A **concept hierarchy** defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. Concept hierarchies allow data to be handled at varying levels of abstraction.
- ❑ A concept hierarchy may have total or partial order among attributes:
  - ❑ Total: location, partial: time (forms lattice)
- ❑ Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a **set-grouping hierarchy**.
- ❑ There may be more than one concept hierarchy for a given attribute or dimension, based on different user viewpoints. For instance, a user may prefer to organize *price* by defining ranges for *inexpensive*, *moderately priced*, and *expensive*.
- ❑ Concept hierarchies may be provided manually by system users, domain experts, or knowledge engineers, or may be automatically generated based on statistical analysis of the data distribution.

# A Concept Hierarchy for a **Dimension** (location)

---



# View of Warehouses and Hierarchies



## Specification of hierarchies

- ❑ Schema hierarchy  
day < {month < quarter;  
week} < year
- ❑ Set\_grouping hierarchy  
{1..10} < inexpensive

# Data Cube Measures: Three Categories

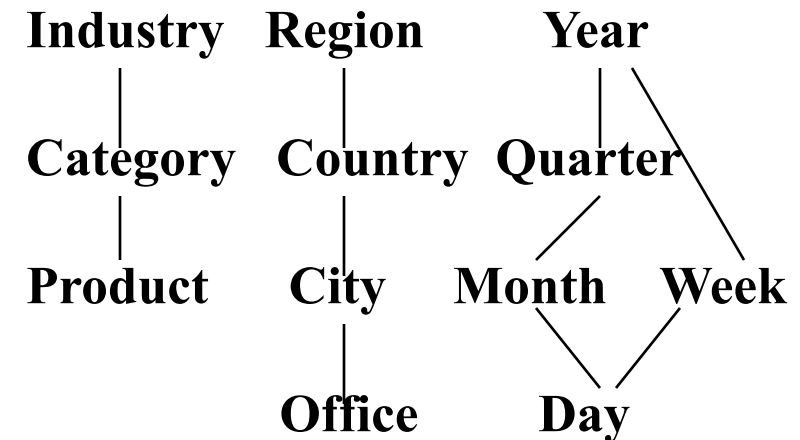
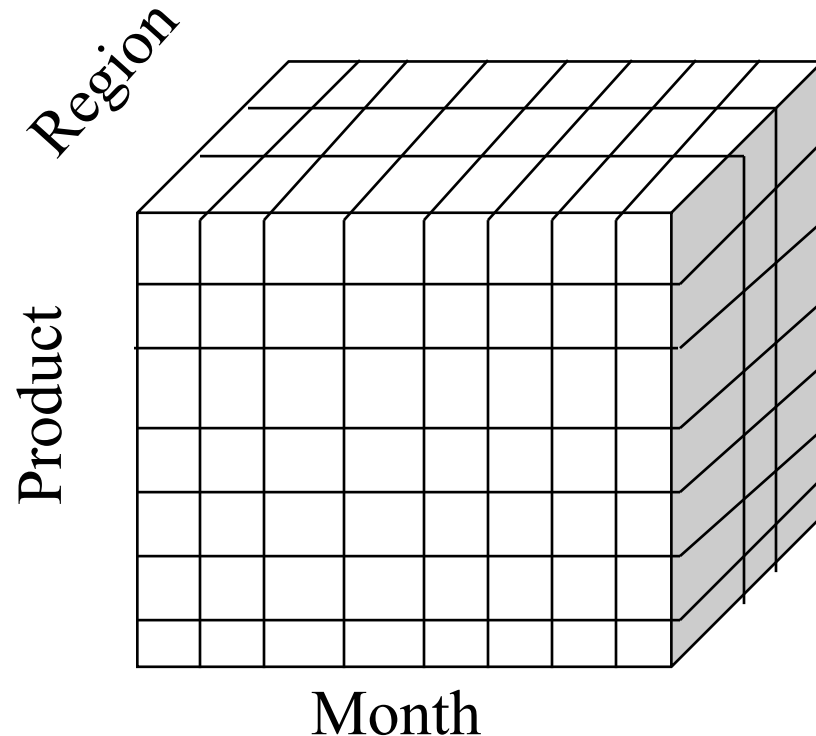
---

- ❑ Measures can be organized into three categories—distributive, algebraic, and holistic—based on the kind of aggregate functions used.
- ❑ Distributive: if the result derived by applying the function to  $n$  aggregate values is the same as that derived by applying the function on all the data without partitioning
  - ❑ E.g., `count()`, `sum()`, `min()`, `max()`
- ❑ Algebraic: if it can be computed by an algebraic function with  $M$  arguments (where  $M$  is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - ❑ `avg(x) = sum(x) / count(x)`, where both `sum()` and `count()` are distributive aggregate functions. Is `min_N()`, `max_N` are algebraic measures? How about `standard_deviation()`?
- ❑ Holistic: if there is no constant bound on the storage size needed to describe a sub-aggregate.
  - ❑ E.g., `median()`, `mode()`, `rank()`

# Multidimensional Data

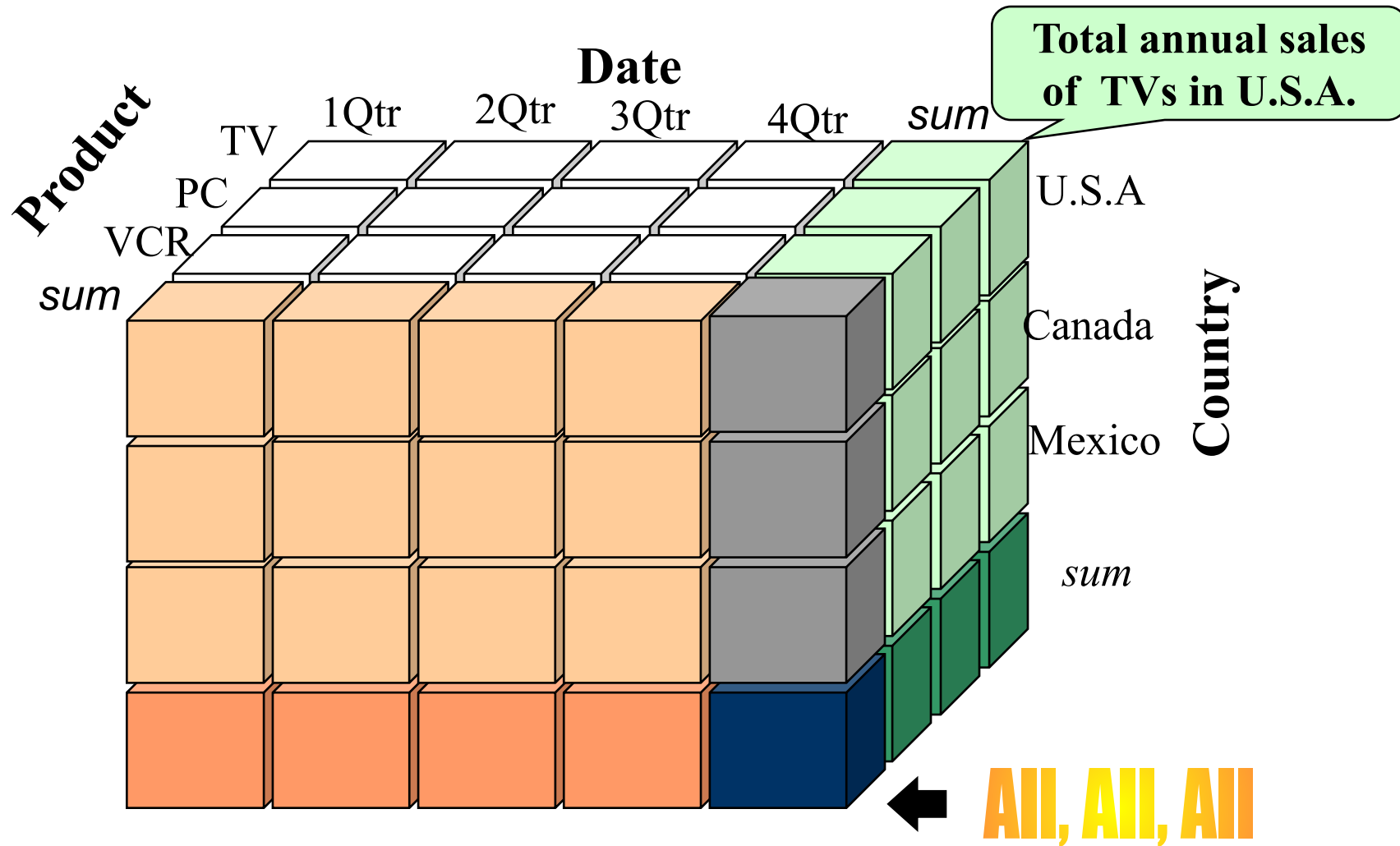
- ❑ Sales volume as a function of product, month, and region

**Dimensions:** *Product, Location, Time*  
**Hierarchical summarization paths**



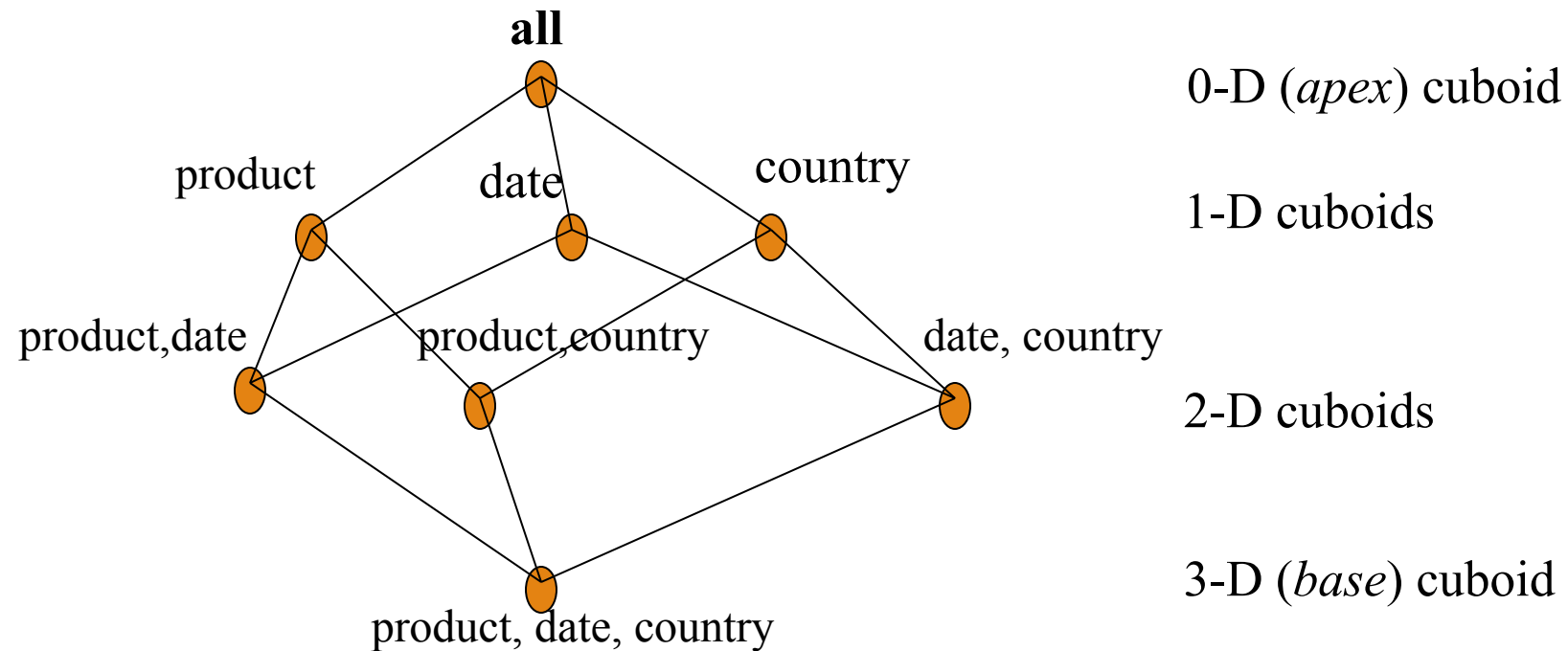


# A Sample Data Cube

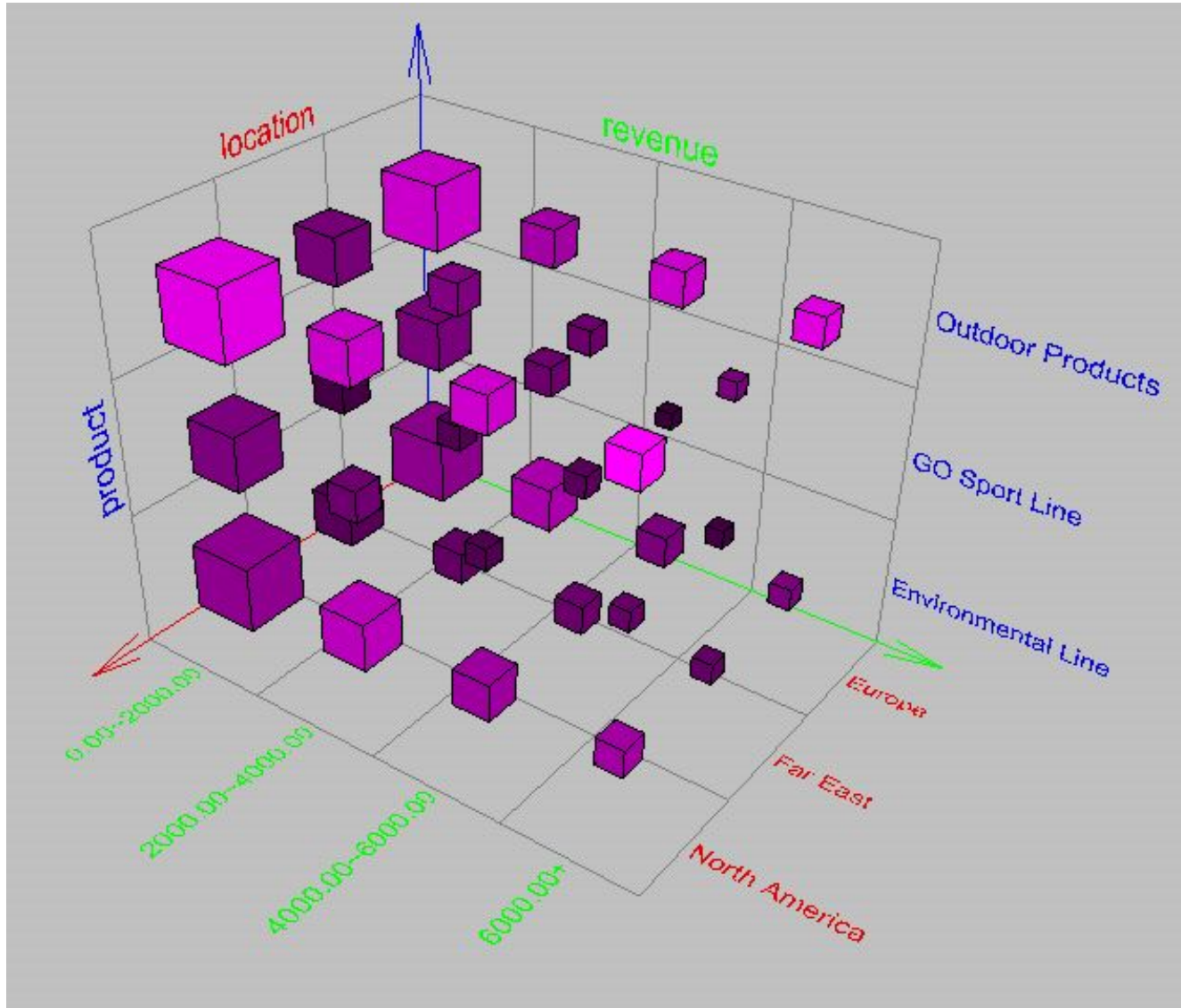


# Cuboids Corresponding to the Cube

---



# Browsing a Data Cube



- ☐ Visualization
- ☐ OLAP capabilities
- ☐ Interactive manipulation

# Typical OLAP Operations

---

## ❑ Roll-up (drill-up): summarize data

- ❑ The roll-up operation (also called the *drill-up* ) 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.

## ❑ Drill-down (roll-down): reverse of roll-up

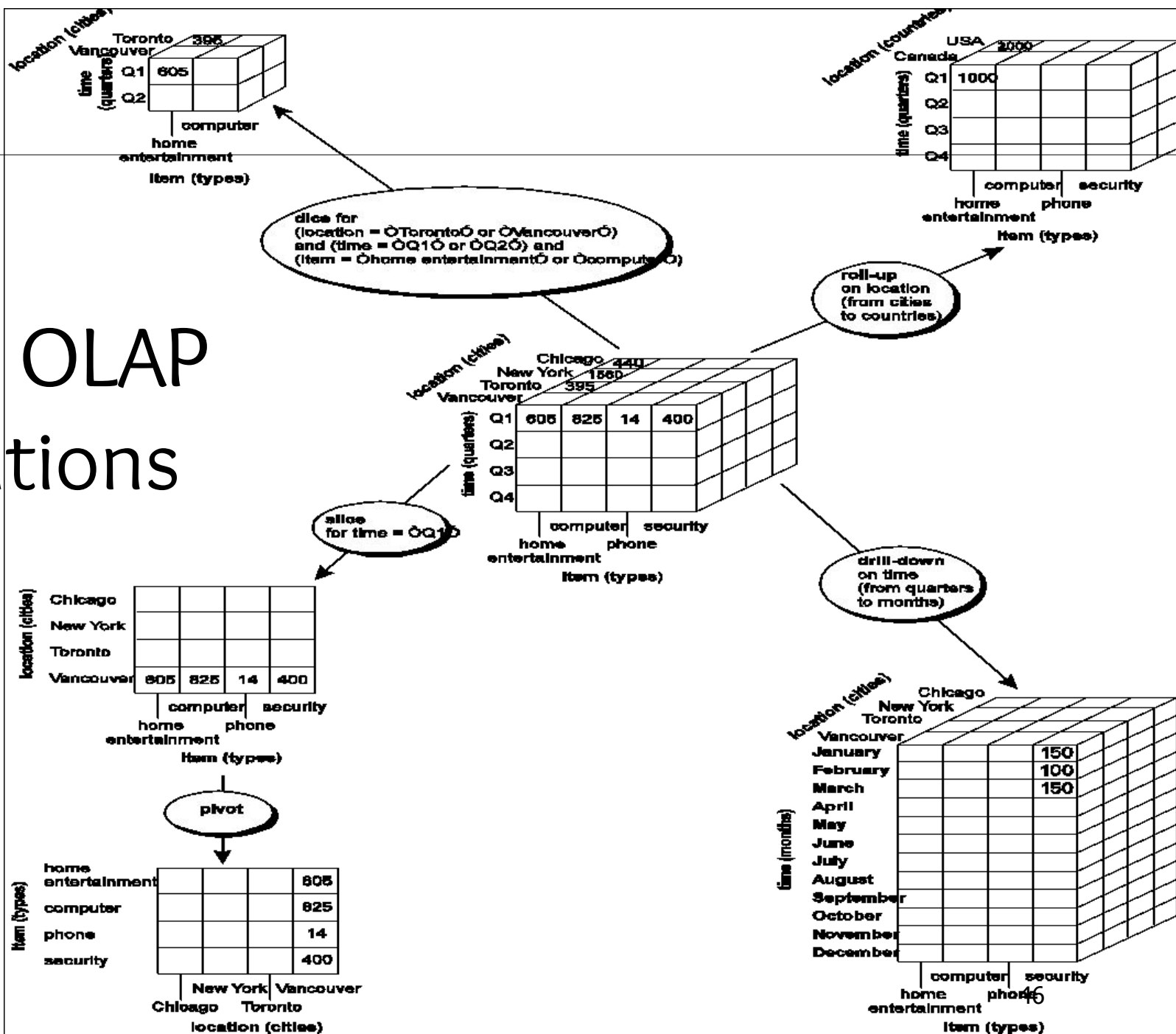
- ❑ It navigates from less detailed data to more detailed data. Drill-down can be realized by either *stepping down a concept hierarchy* for a dimension or *introducing additional dimensions*
- ❑ Because a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube. For example, a drill-down on the central cube 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.
  - ❑ Examples: rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.
- ❑ Other operations
  - ❑ **Drill across:** execute queries involving (across) more than one fact table
  - ❑ **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



# A Star-Net Query Model for Multidimensional Model

- The querying of multidimensional databases can be based on a **starnet model**
  - 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.
- Concept hierarchies can be used to **generalize** data by replacing low-level values (such as “day” for the *time* dimension) by higher-level abstractions (such as “year”), or to **specialize** data by replacing higher-level abstractions with lower-level values.

# A Star-Net Query Model

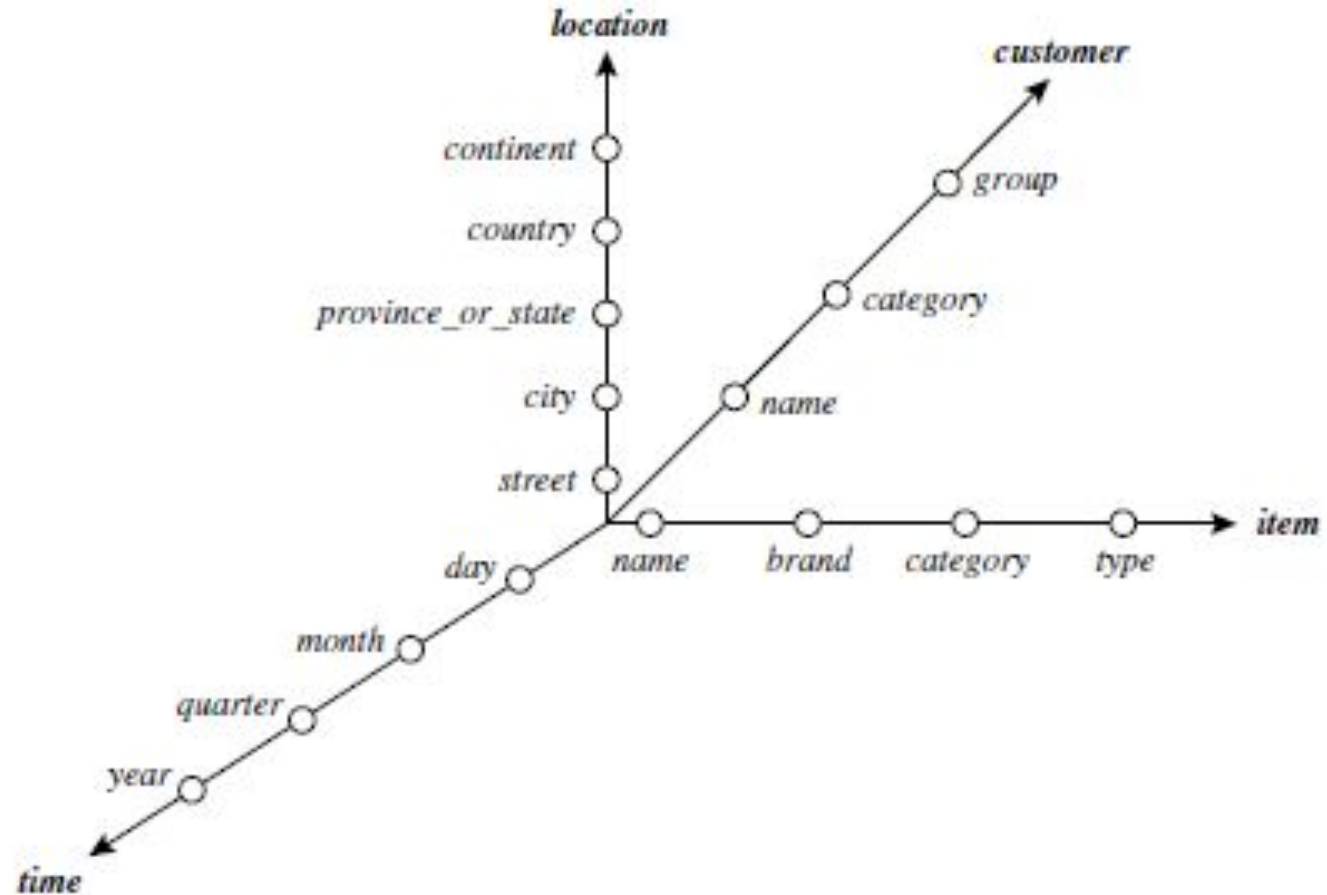
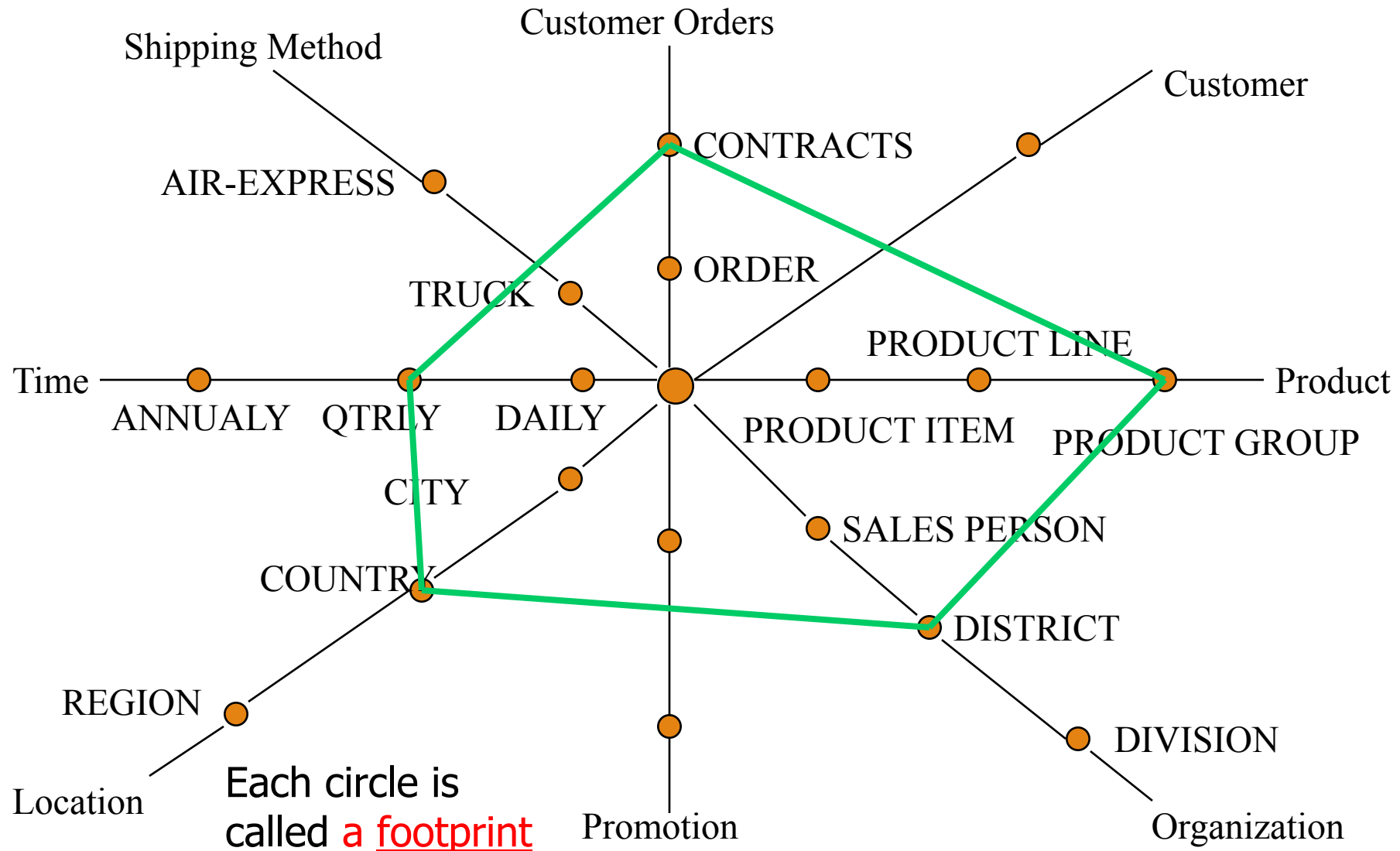


Figure 4.13 A starnet model of business queries.




# A Star-Net Query Model



# Chapter 4: Data Warehousing and On-line Analytical Processing

---

- ❑ Data Warehouse: Basic Concepts
- ❑ Data Warehouse Modeling: Data Cube and OLAP
- ❑ Data Warehouse Design and Usage 
- ❑ Data Warehouse Implementation
- ❑ Summary

# Design of Data Warehouse: A Business Analysis Framework

---

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

# Design of Data Warehouse: A Business Analysis Framework

---

- ❑ Four views regarding a data warehouse design must be considered:
  - ❑ Top-down view
  - ❑ Data source view
  - ❑ Data warehouse view
  - ❑ Business query view
  - ❑ **Top-down view**
    - ❑ allows selection of the relevant information necessary for the data warehouse. It matches the current and future business needs.
  - ❑ **Data source view**
    - ❑ This view exposes the information being captured, stored, and managed by operational systems.

# Design of Data Warehouse: A Business Analysis Framework

---

- ❑ Usually modeled by traditional data modeling techniques, such as ER model or CASE (computer-aided software engineering) tools.
- ❑ Data warehouse view
  - ❑ This view consists of fact tables and dimension tables.
  - ❑ It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- ❑ Business query view
  - ❑ It is the data perspective in the data warehouse from the end-user's viewpoint.

# Data Warehouse Design Process

---

- ❑ Built using *top-down approach, bottom-up approach or a combination of both.*
  - ❑ Top-down: Starts with overall design and planning (mature)
  - ❑ Bottom-up: Starts with experiments and prototypes (rapid)
  - ❑ Combined exploit the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach
- ❑ **From software engineering point of view**
  - ❑ Waterfall: structured and systematic analysis at each step before proceeding to the next
  - ❑ Spiral: 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

---

## □ Typical data warehouse design process

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

# Data Warehouse Usage

---

- Three kinds of data warehouse applications

- **Information processing**

- supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs.
    - current trend 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.
    - 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 Warehouse Usage

---

- ❑ Data mining
  - ❑ Supports knowledge discovery from hidden patterns
  - ❑ Also supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

# From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM)

---

- ❑ The data mining field has conducted substantial research regarding mining on various data types, including relational data, data from data warehouses, transaction data, time-series data, spatial data, text data, and flat files.
- ❑ **Multidimensional data mining** (also known as *exploratory multidimensional data mining*, **online analytical mining**, or **OLAM**) integrates OLAP with data mining to uncover knowledge in multidimensional databases.
- ❑ Multidimensional data mining is particularly important for the following reasons:
  - ❑ **High quality of data in data warehouses:**
    - ❑ Most data mining tools need to work on integrated, consistent, and cleaned data, which requires costly data cleaning, data integration, and data transformation as preprocessing steps.
    - ❑ A data warehouse constructed by such preprocessing serves as a valuable source of high-quality data for OLAP as well as for data mining.

# From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM)

---

- ❑ **Available information processing structure surrounding data warehouses**
  - ❑ Includes accessing, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web accessing and service facilities, and reporting and OLAP analysis tools.
  - ❑ It is prudent to make the best use of the available infrastructures rather than constructing everything from scratch.
- ❑ **OLAP-based exploration of multidimensional data**
  - ❑ It facilitates 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,


# From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM)

---

- ❑ 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.
- ❑ **On-line selection of data mining functions**
  - ❑ 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.

# Chapter 4: Data Warehousing and On-line Analytical Processing

---

- ❑ Data Warehouse: Basic Concepts
- ❑ Data Warehouse Modeling: Data Cube and OLAP
- ❑ Data Warehouse Design and Usage
- ❑ Data Warehouse Implementation 
- ❑ Summary

# 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. So, the knowledge of followings are important:
  - ❑ Overview of methods for the efficient implementation of data warehouse systems
  - ❑ Explore how to compute data cubes efficiently
  - ❑ How OLAP data can be indexed, using either bitmap or join indices
  - ❑ How OLAP queries are processed
  - ❑ Finally, various types of warehouse servers for OLAP processing

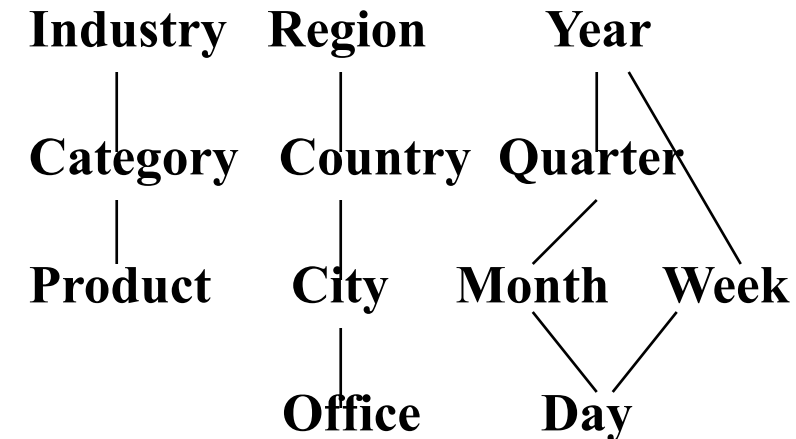
# Efficient Data Cube Computation: An Overview

- 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.
- No. of cuboids:

Why this formula?

$$T = \prod_{i=1}^n (L_i + 1)$$

$$\text{number of cuboids} = \begin{cases} 2^n & \text{If no hierarchy} \\ \prod_{i=1}^n (L_i + 1) & \text{if hierarchy and } L_i \text{ is number of levels associated with dimension } i \end{cases}$$



# The “Compute Cube” Operator

- ❑ Cube definition and computation in DMQL

```
define cube sales_cube [item, city, year]: sum (sales_in_dollars)
```

```
compute cube sales_cube
```

- ❑ Transform it into a SQL-like language (with a new operator **cube by**, introduced by Gray et al.'96)

```
SELECT item, city, year, SUM (amount)
```

```
FROM sales_cube
```

```
CUBE BY item, city, year
```

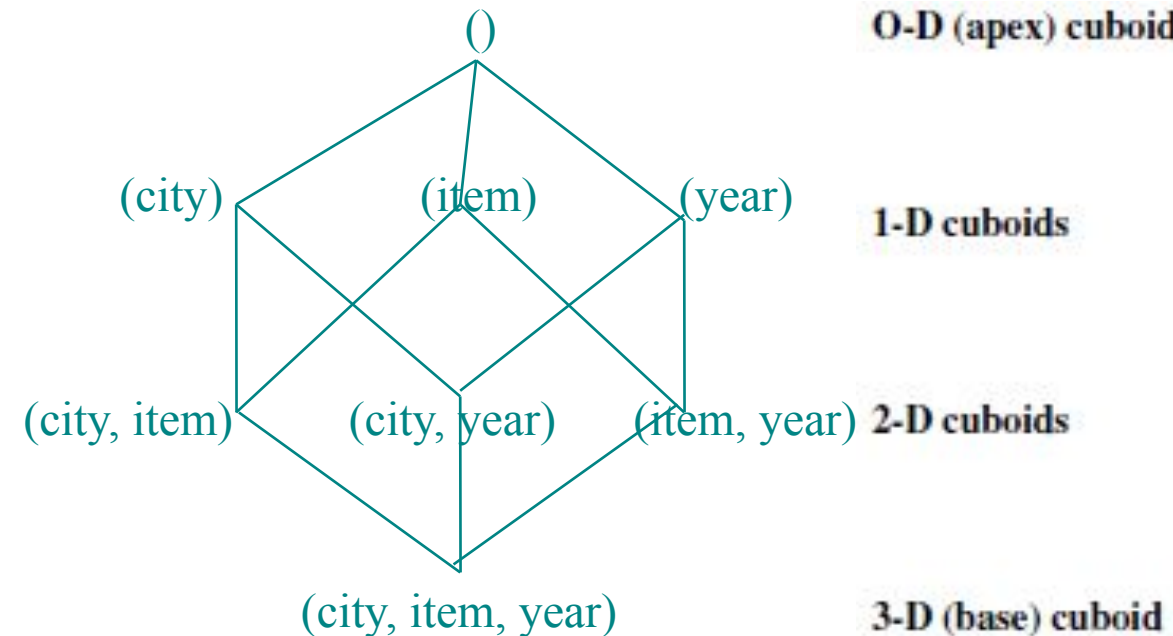
- ❑ Need compute the following Group-Bys

```
(item, city, year),
```

```
(item, city),(item, year), (city, year),
```

```
(item), (city), (year)
```

```
()
```





# Curse of Dimensionality

---

- ❑ 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.
- ❑ Pre-computation leads to fast response time and avoids some redundant computation. Most, if not all, OLAP products resort to some degree of pre-computation of multidimensional aggregates.
- ❑ A major challenge related to this pre-computation, however, is that the required storage space may explode if all the cuboids in a data cube are pre-computed, especially when the cube has many dimensions.
- ❑ The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels.
- ❑ This problem is referred to as the **curse of dimensionality**.

# Why Partial Materialization?

---

- ❑ At most, one abstraction level in each dimension will appear in a cuboid.
- ❑ For example, the time dimension (year < quarter < month < day ) 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  $5 \wedge 10 \approx 9.8 \times 10 \wedge 6$ .
- ❑ The size of each cuboid also depends on the *cardinality (i.e., number of distinct values)* of each dimension. As the number of dimensions, number of conceptual hierarchies, or cardinality increases, the storage space required for many of the group-by's will grossly exceed the (fixed) size of the input relation.
- ❑ So, it is unrealistic to pre-compute and materialize all of the cuboids that can possibly be generated for a data cube (i.e., from a base cuboid).
- ❑ If there are many large cuboids, a more reasonable option is *partial materialization*; that is, to materialize only some of the possible cuboids that can be generated.

# Partial Materialization: Selected Computation of Cuboids

---

- ❑ There are three choices for data cube materialization given a base cuboid:
- ❑ **1. No materialization:** Do not pre-compute any of the “non-base” cuboids. This leads to computing expensive multidimensional aggregates on-the-fly, which can be extremely slow.
- ❑ **2. Full materialization:** Pre-compute 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 pre-computed cuboids.
- ❑ **3. Partial materialization:** Selectively compute a proper subset of the whole set of possible cuboids. Alternatively, we may compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion, such as where the tuple count of each cell is above some threshold, called *subcube*, where only some of the cells may be pre-computed for various cuboids.
- ❑ Partial materialization represents an interesting trade-off between storage space and response time.

# Indexing OLAP Data: Bitmap Index

- Index on a particular column
  - Each value in the column has a bit vector: bit-op is fast
  - The length of the bit vector: # of records in the base table
  - The  $i$ -th bit is set if the  $i$ -th row of the base table has the value for the indexed column
  - not suitable for high cardinality domains
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

**Base table**

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

**Index on Region**

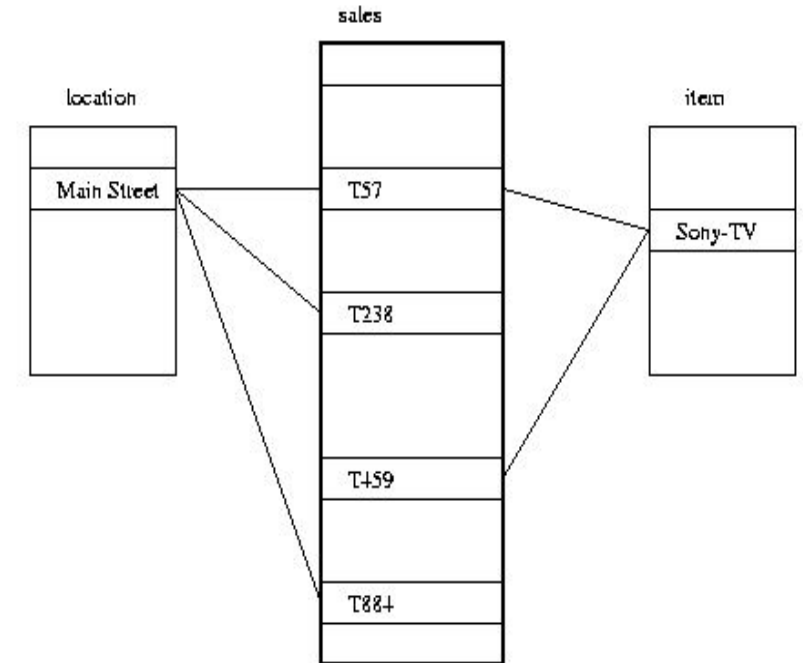
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

**Index on Type**

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

# Indexing OLAP Data: Join Indices

- ❑ Join index:  $Jl(R\text{-id}, S\text{-id})$  where  $R (R\text{-id}, \dots)$   $\square \square S (S\text{-id}, \dots)$
- ❑ Traditional indices map the values to a list of record ids
  - ❑ It materializes relational join in JI file and speeds up relational join
- ❑ In data warehouses, join index relates the values of the dimensions of a star schema to rows in the fact table.
  - ❑ E.g., fact table: *Sales* and two dimensions *city* and *product*
    - ❑ A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city



# Efficient Processing OLAP Queries

---

- ❑ The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes. Given materialized views, query processing should proceed as follows:
- ❑ **1. Determine which operations should be performed on the available cuboids:** This involves transforming any selection, projection, roll-up (group-by), and drill-down operations specified in the query into corresponding SQL and/or OLAP operations.
- ❑ For example, slicing and dicing a data cube may correspond to selection and/or projection operations on a materialized cuboid.
- ❑ **2. Determine to which materialized cuboid(s) the relevant operations should be applied:** This involves identifying all of the materialized cuboids that may potentially be used to answer the query, pruning the set using knowledge of “dominance” relationships among the cuboids, estimating the costs of using the remaining materialized cuboids, and selecting the cuboid with the least cost.

# Efficient Processing OLAP Queries

---

- ❑ **Example:** 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.
- ❑ Let the query to be processed be on {*brand, province\_or\_state*} with the condition “*year = 2004*”, and there are 4 materialized cuboids available:
  - ❑ 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 = 2004*
- ❑ Which cuboid should be selected to process the query? Cuboid 3 or 4?

# OLAP Server Architectures

---

## ❑ Relational OLAP (ROLAP)

- ❑ Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware to support missing pieces
- ❑ Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
- ❑ Greater scalability

## ❑ Multidimensional OLAP (MOLAP)

- ❑ Support multidimensional data views through array-based multidimensional storage engines.
- ❑ They map multidimensional views directly to data cube array structures
- ❑ Allows fast indexing to pre-computed summarized data



# OLAP Server Architectures

---

- ❑ **Hybrid OLAP (HOLAP)** (e.g., Microsoft SQL Server)
  - ❑ Combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP (low level: relational) and the faster computation of MOLAP (high-level: array)
  - ❑ 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.
- ❑ **Specialized SQL Servers** (e.g., Redbricks)
  - ❑ 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.

# Chapter 4: Data Warehousing and On-line Analytical Processing

---

- ❑ Data Warehouse: Basic Concepts
- ❑ Data Warehouse Modeling: Data Cube and OLAP
- ❑ Data Warehouse Design and Usage
- ❑ Data Warehouse Implementation
- ❑ Summary



# Summary

---

- ❑ Data warehousing: A multi-dimensional model of a data warehouse
  - ❑ A data cube consists of *dimensions & measures*
  - ❑ Star schema, snowflake schema, fact constellations
  - ❑ OLAP operations: drilling, rolling, slicing, dicing and pivoting
- ❑ Data Warehouse Architecture, Design, and Usage
  - ❑ Multi-tiered architecture
  - ❑ Business analysis design framework
  - ❑ Information processing, analytical processing, data mining, OLAM
- ❑ Implementation: Efficient computation of data cubes
  - ❑ Partial vs. full vs. no materialization
  - ❑ Indexing OALP data: Bitmap index and join index
  - ❑ OLAP query processing
  - ❑ OLAP servers: ROLAP, MOLAP, HOLAP