UNIT-1
Data Warehouse

What is a Data Warehouse?

- Data warehouse is
  - 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."—W. H. Inmon

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
- 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, attribute measures, etc. among different data sources
  - 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 database: current value 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

- A physically separate store of data transformed from the operational environment
- 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*

OLTP vs. OLAP

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

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

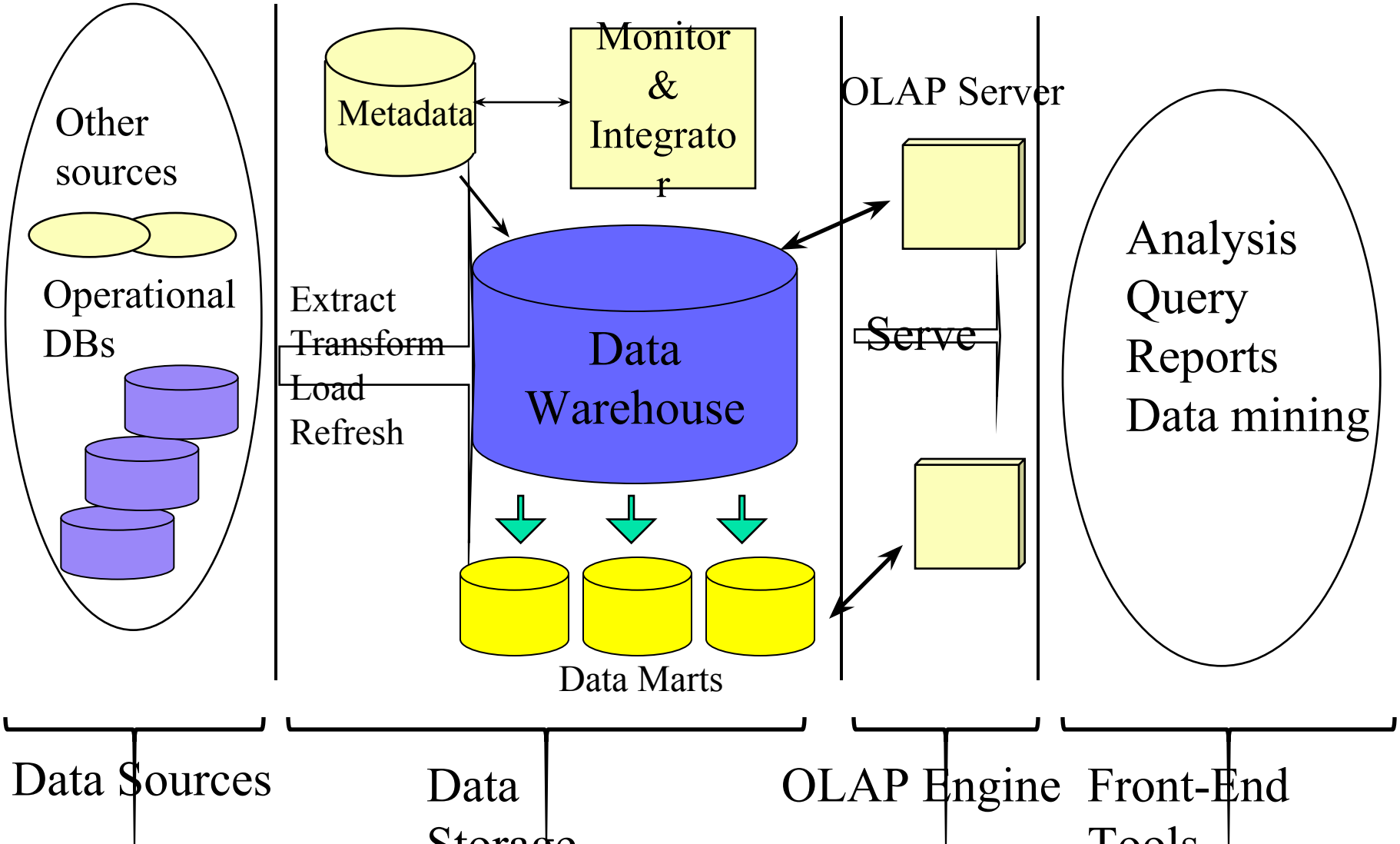
Types of Data Warehouse

- **Information Processing** –
  - A data warehouse allows to process the data stored in it.
  - The data can be processed by means of querying, reporting using tables, charts, or graphs.
- **Analytical Processing** –
  - A data warehouse supports analytical processing of the information stored in it.
  - The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.

- **Data Mining** –
  - Data mining supports knowledge discovery by finding
    - hidden patterns
    - Associations
    - constructing analytical models
    - performing classification
    - and prediction.

    These mining results can be presented using the visualization tools.

Data Warehouse: A Multi-Tiered Architecture



Three Data Warehouse Models

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse)
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

Extraction, Transformation, and Loading (ETL)

- **Data extraction**
  - get data from multiple, heterogeneous, and external sources
- **Data cleaning**
  - detect errors in the data and rectify them when possible
- **Data transformation**
  - convert data from legacy or host format to warehouse format
- **Load**
  - sort, summarize, consolidate
- **Refresh**
  - propagate the updates from the data sources to the warehouse

Metadata Repository

- **Meta data** is the data defining warehouse objects. It stores:
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, data mart locations and contents
- Operational meta-data
  - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarization
  - Measure and dimension definition

Metadata Repository

- The mapping from operational environment to the data warehouse
  - Source database and its content, gateway description, cleaning and transformation rule
- Data related to system performance
  - Which include indices and profile that improve data access and retrieval information
  - Rule for timing and scheduling of refresh, update
- Business data
  - business terms and definitions, ownership of data, charging policies

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
- 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
- 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.

Data Cube; A Multidimensional Data Model

- Data warehouse and OLAP tools are bases on multidimensional data model
- This model view data in the form of data cube
- A data cube allows data to be modeled and viewed in multiple dimensions.
- It is defined by dimensions and facts.

Dimension:-

- Dimensions are the perspectives or entities with respect to which an organization wants to keep records
- For example, AllElectronics 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.

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

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

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

Table 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**			
	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

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

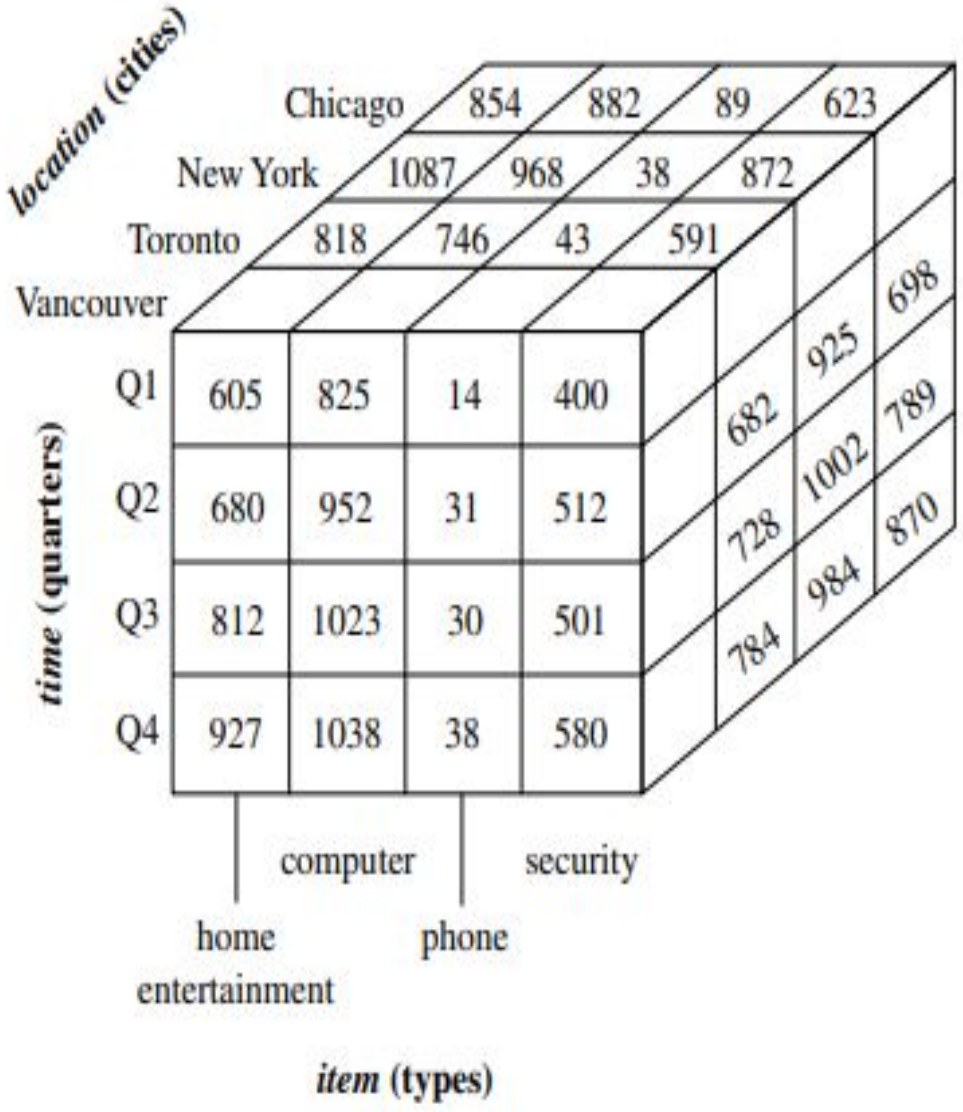


Figure 4.3 A 3-D data cube representation of the data in Table 4.3, according to *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

Fact:-

- 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.

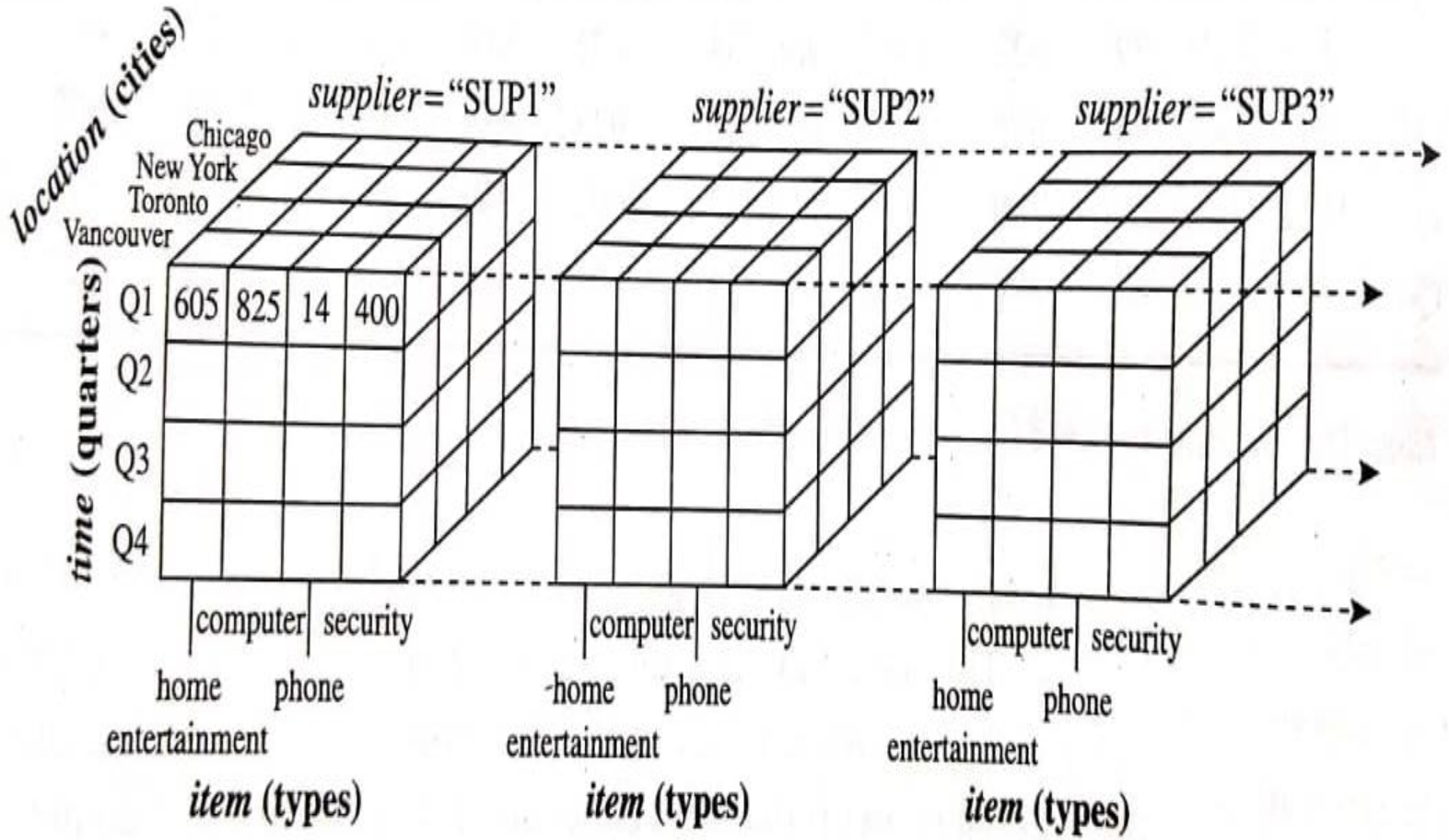


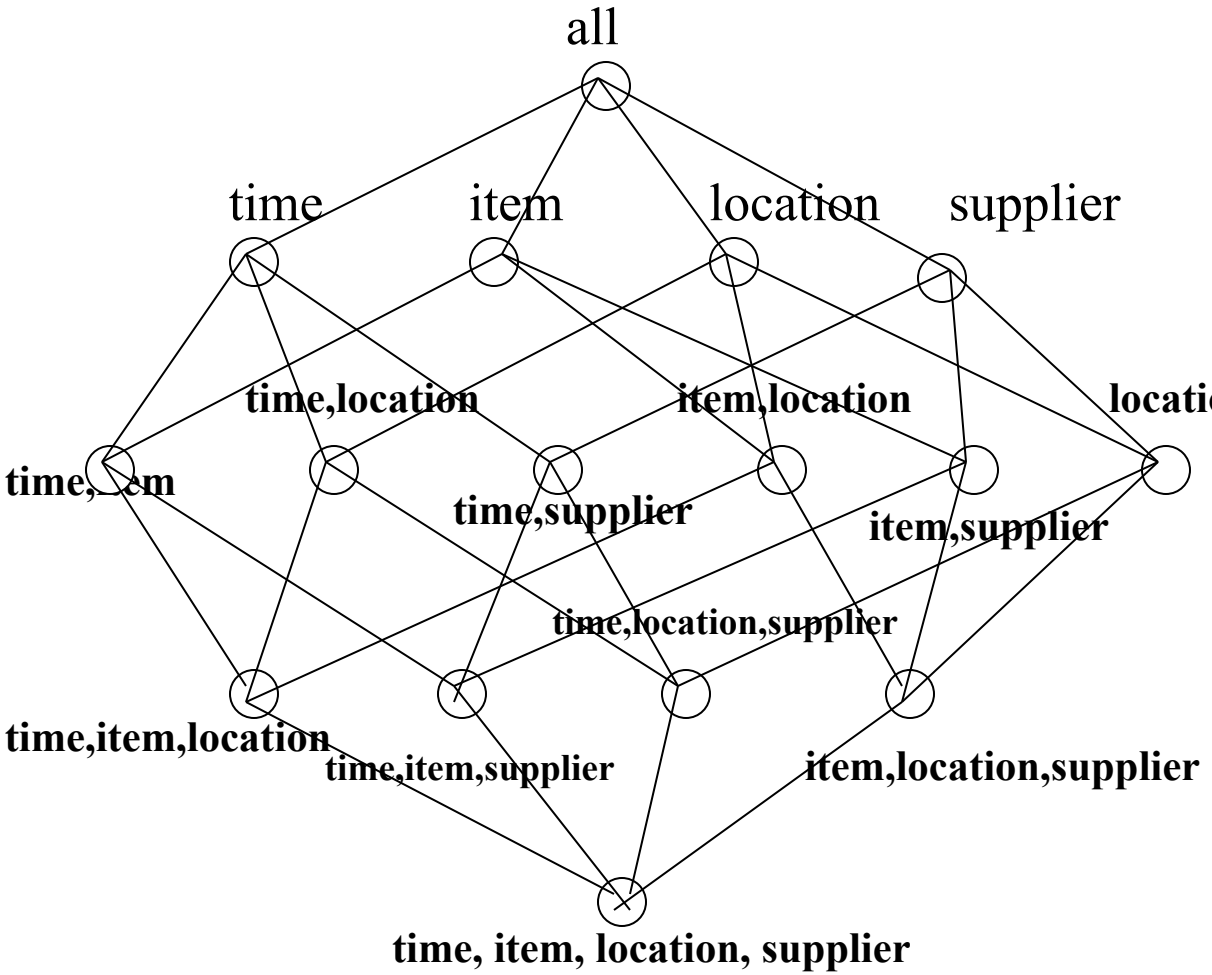
Figure 4.4 A 4-D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of the cube values are shown.

Cube: A Lattice of Cuboids

Cuboids

- A data cube at different degree of summarization referred as cuboids
- We can generate cuboids for each possible subset of the given summarization
- The result would form lattice of cuboids: Each showing data at different level of summarization Or group by
- The lattice of cuboids is then referred as a data cube
- The cuboids that holds the lowest level of summarization is called the base cuboids.
- The 0-D cuboids, which holds the highest level of summarization, is called the apex cuboids.

Cube: A Lattice of Cuboids



0-D (*apex*) cuboid

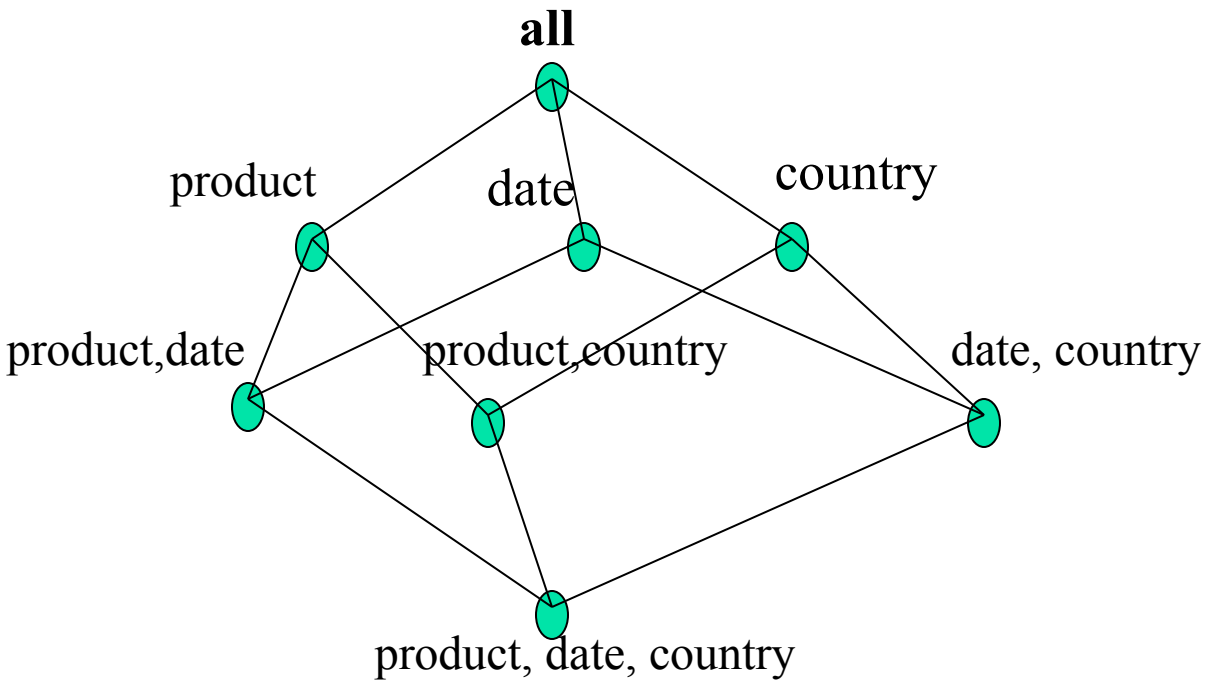
1-D cuboids

2-D cuboids

3-D cuboids

4-D (*base*) cuboid

Cuboids Corresponding to the Cube



0-D (*apex*) cuboid

1-D cuboids

2-D cuboids

3-D (*base*) cuboid

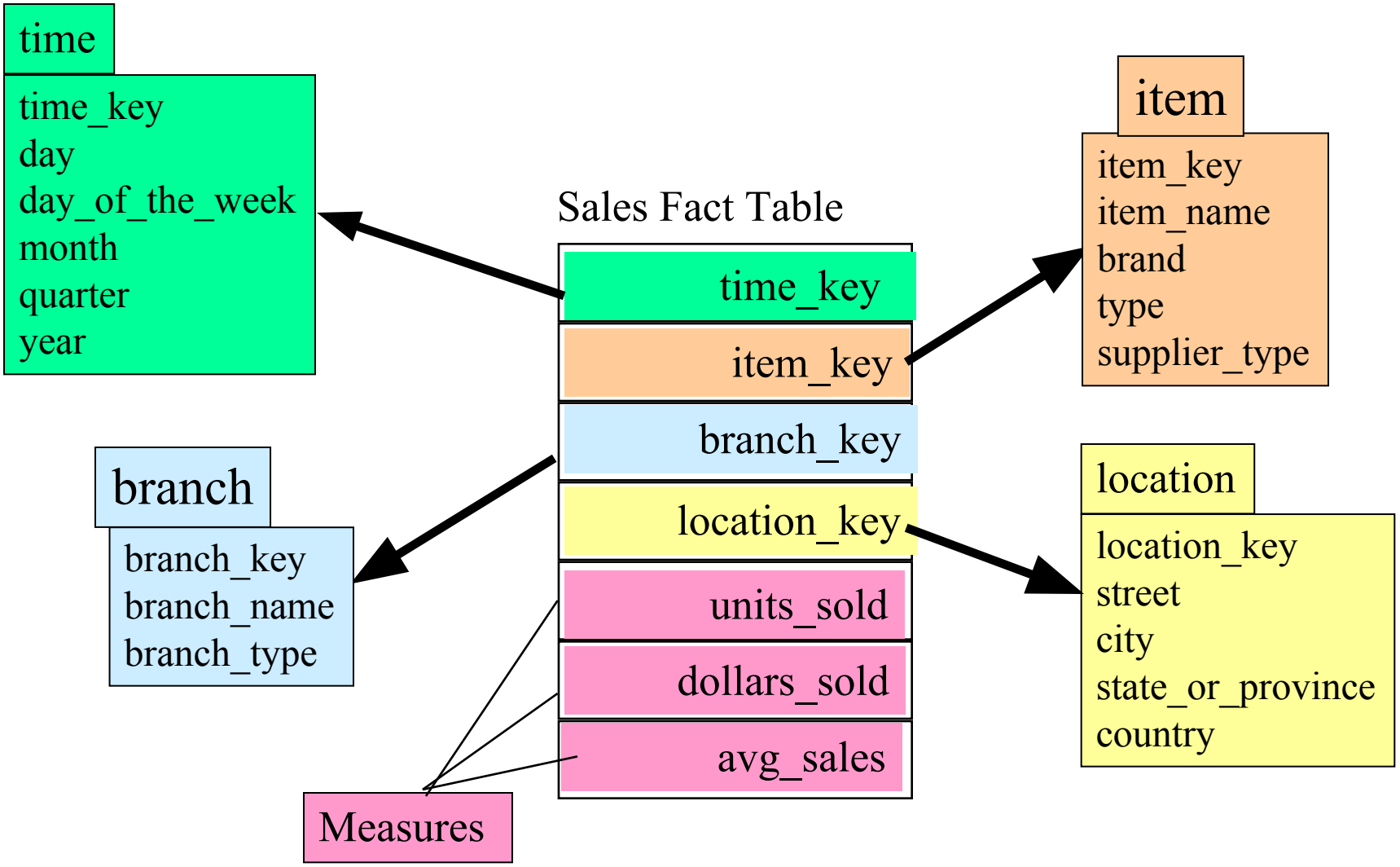
Schemas for Multidimensional Data Model

- The entity-relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities and the relationships between them.
- Such a data model is appropriate for online transaction processing
- A data warehouse, however, requires a concise, subject-oriented schema that facilitates online data analysis.
- The most popular data model for a data warehouse is a multidimensional model, which can exist in the form of a star schema, a snowflake schema, or a fact constellation schema.

Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  - Star schema:
  - The most common modeling paradigm is the star schema, in which the data warehouse contains
  - (1) a large central table (fact table) containing the bulk of the data, with no redundancy,
  - (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.

Example of **Star Schema**



- Disadvantage
- For example, the location dimension table contains the attribute set {location_key, street, city, province_or_state, country }.
- **This constraint may introduce some redundancy**.
- For example, “Urbana" and “Chicago" are both cities in the state of Illinois, USA. Entries for such cities in the location dimension table will create redundancy among the attributes province_or_state and country; that is, (..., Urbana, IL, USA) and (..., Chicago, IL, USA). Moreover, the attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).

Snowflake schema:

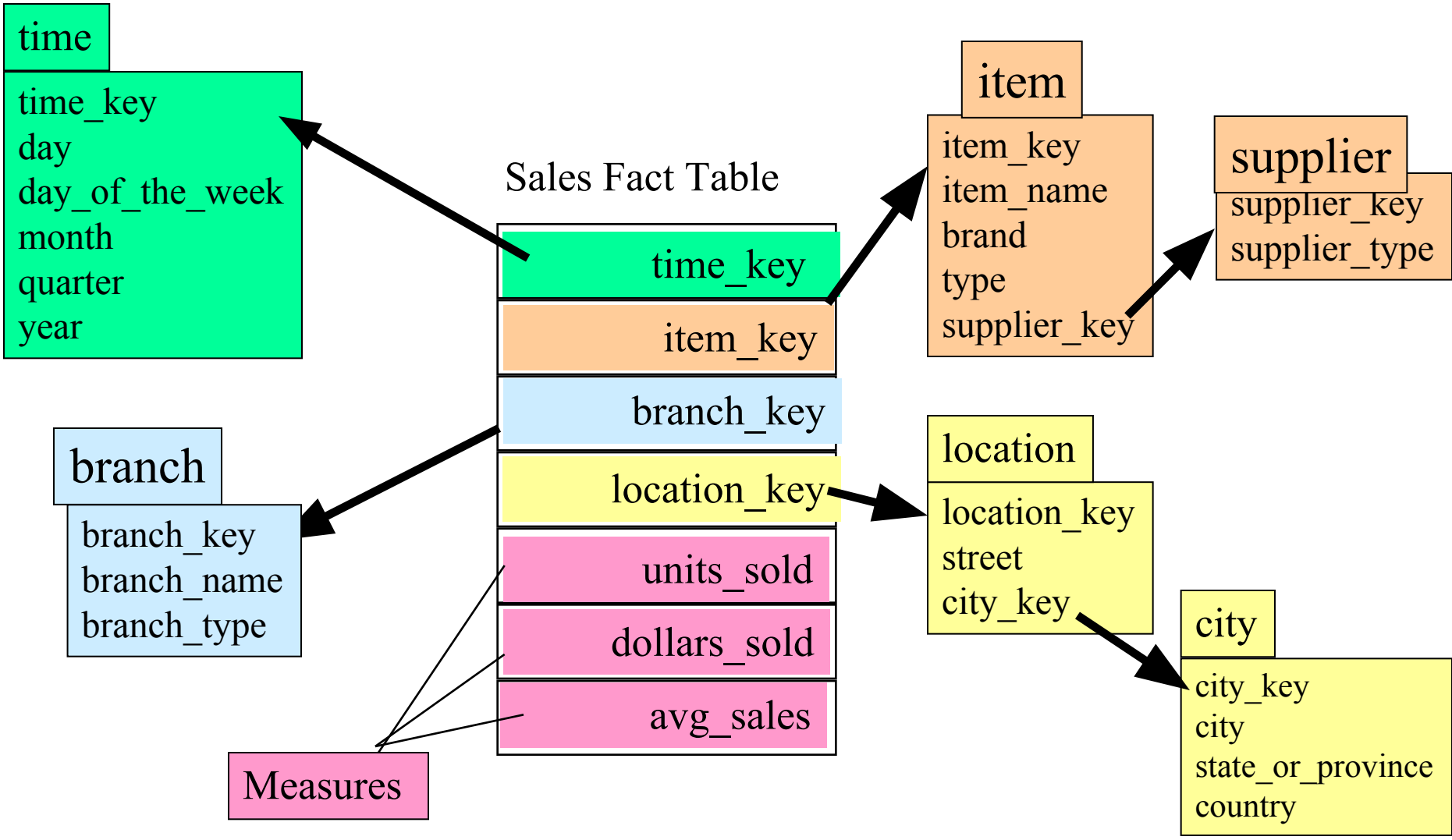
- The snowflake schema is a variant of the star schema model, where some **dimension tables are normalized**, thereby further splitting the data into additional tables.
- The resulting schema graph forms a shape similar to a snowflake.

Advantage of Snowflake Schema

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.**

Example of **Snowflake Schema**

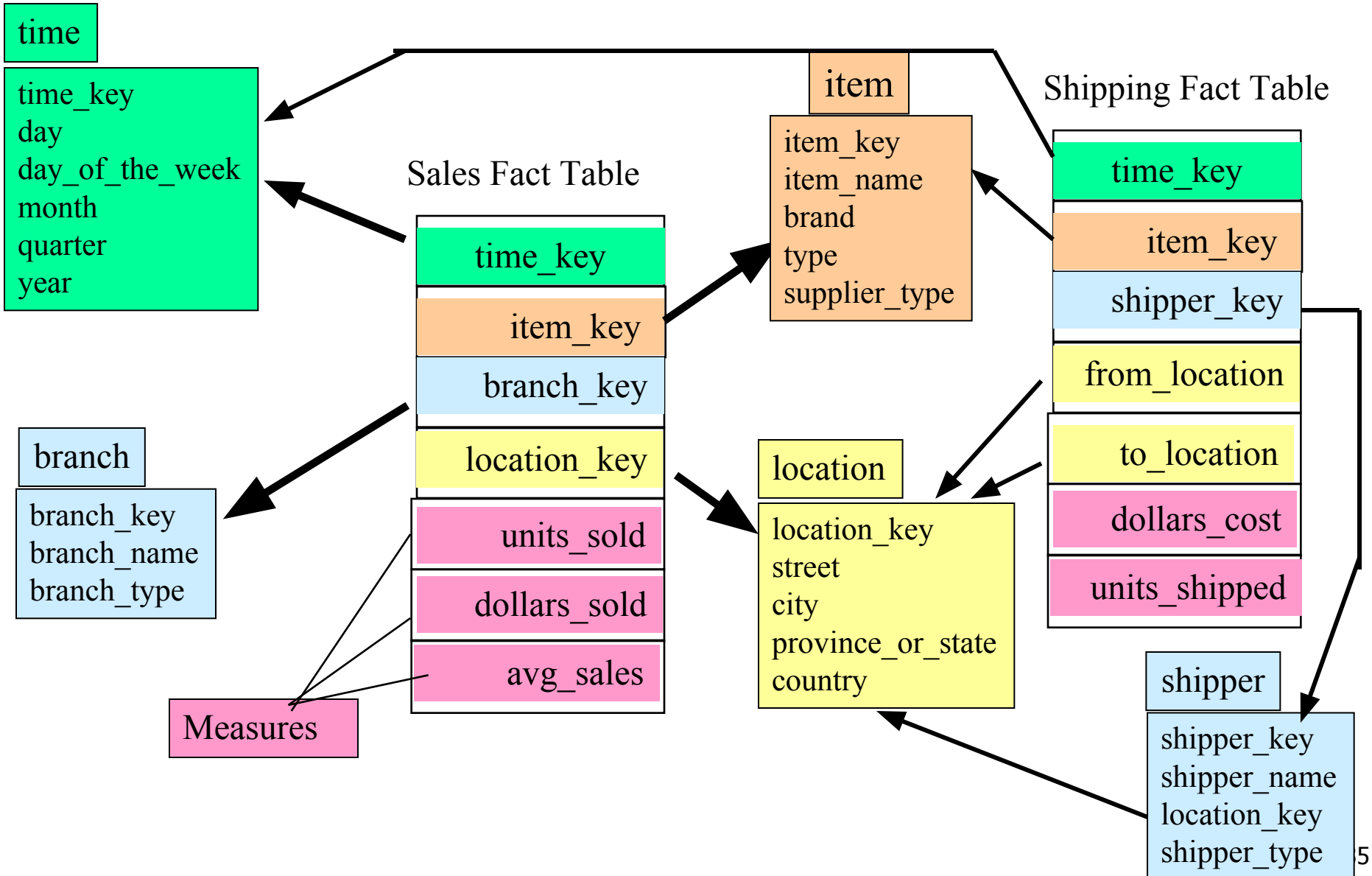


S.NO	Star Schema	Snowflake Schema
1.	In star schema, The fact tables and the dimension tables are contained.	While in snowflake schema, The fact tables, dimension tables as well as sub dimension tables are contained.
2.	Star schema is a top-down model.	While it is a bottom-up model.
3.	Star schema uses more space.	While it uses less space.
4.	It takes less time for the execution of queries.	While it takes more time than star schema for the execution of queries.
5.	In star schema, Normalization is not used.	While in this, Both normalization and denormalization are used.
6.	It's design is very simple.	While it's design is complex.
7.	The query complexity of star schema is low.	While the query complexity of snowflake schema is higher than star schema.
8.	It's understanding is very simple.	While it's understanding is difficult.
9.	It has less number of foreign keys.	While it has more number of foreign keys.
10. *	It has high data redundancy.	While it has low data redundancy.

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.

Example of **Fact Constellation**



Dimension: 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

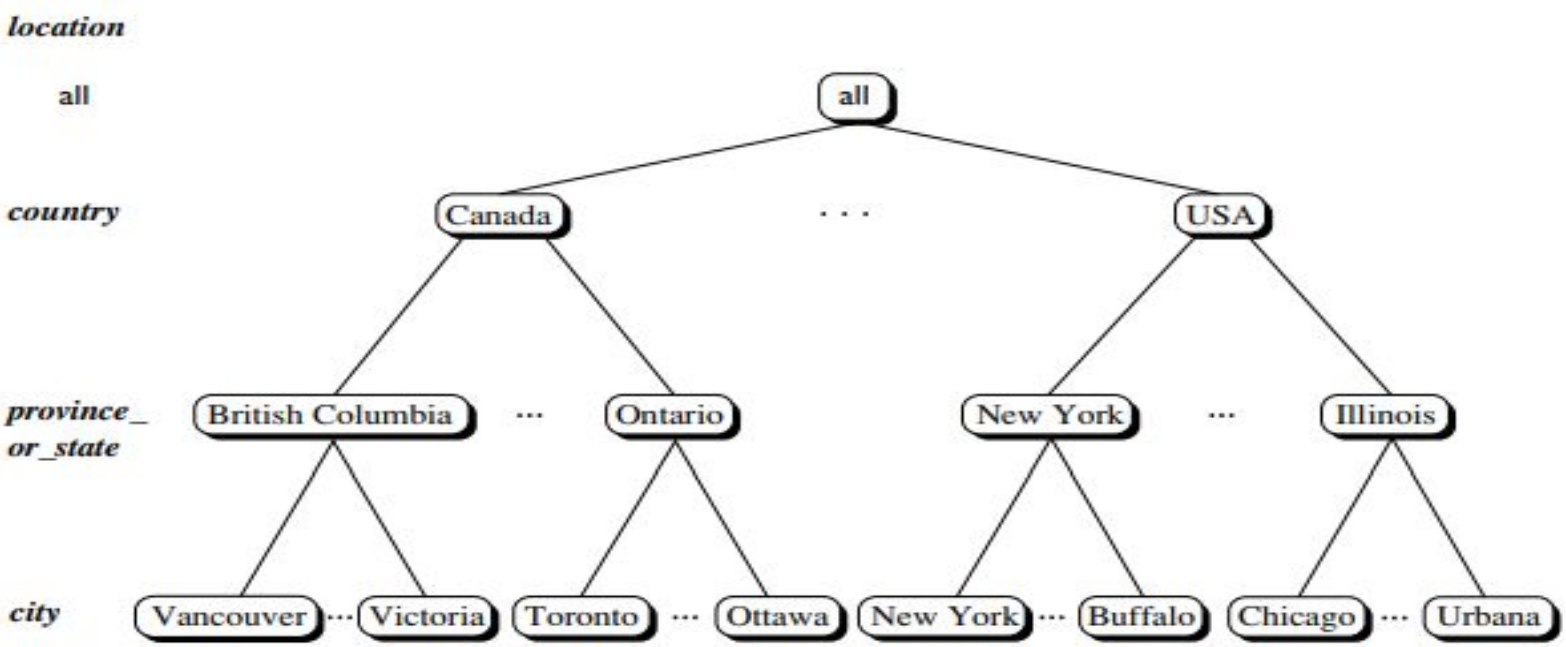


Figure 4.9 A concept hierarchy for *location*. Due to space limitations, not all of the hierarchy nodes are shown, indicated by ellipses between nodes.

- 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.”

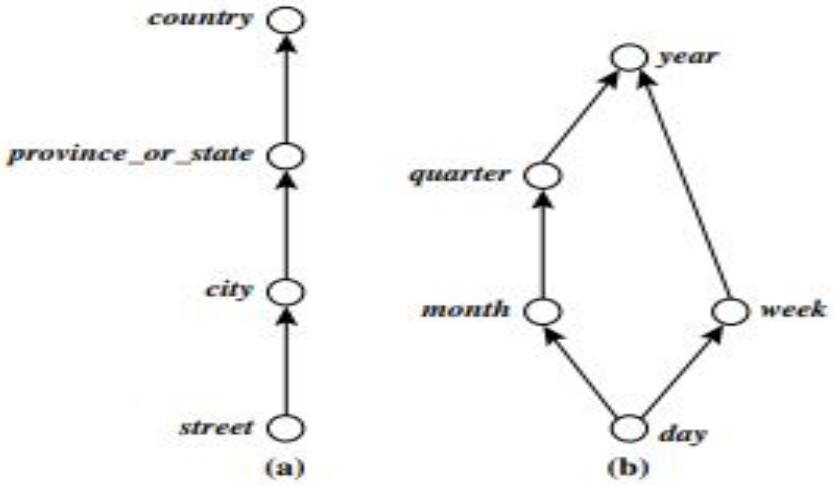


Figure 4.10 Hierarchical and lattice structures of attributes in warehouse dimensions: (a) a hierarchy for *location* and (b) a lattice for *time*.

- The attributes of a dimension may be organized in **a partial order**, forming a lattice.
- An example of a partial order for the time dimension based on the attributes day, week, month, quarter, and year is “day < {month < quarter; week} < year.”
- A concept hierarchy that is a total or partial order among attributes in a database schema is called a **schema hierarchy**.
- 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.

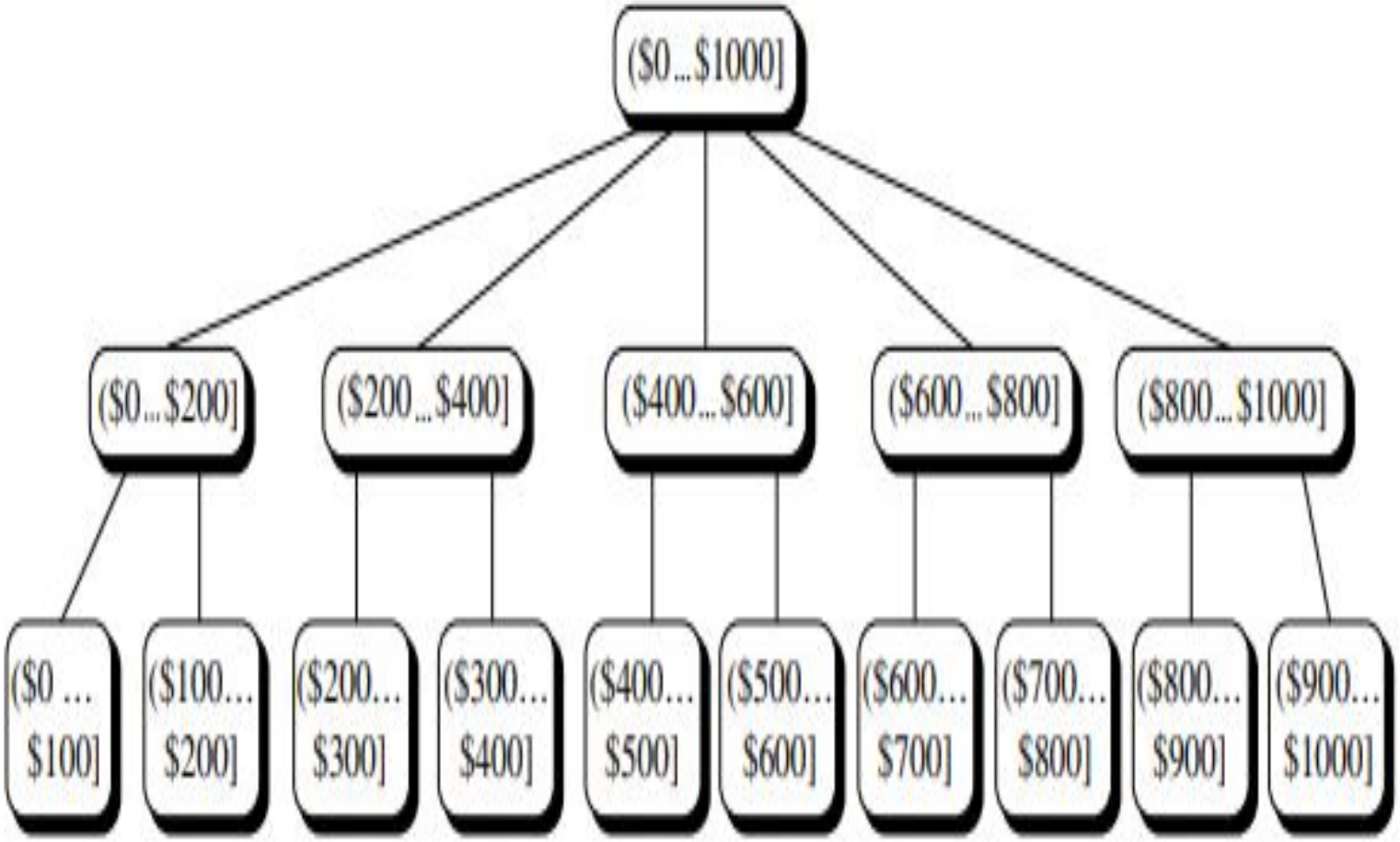


Figure 4.11 A concept hierarchy for *price*.

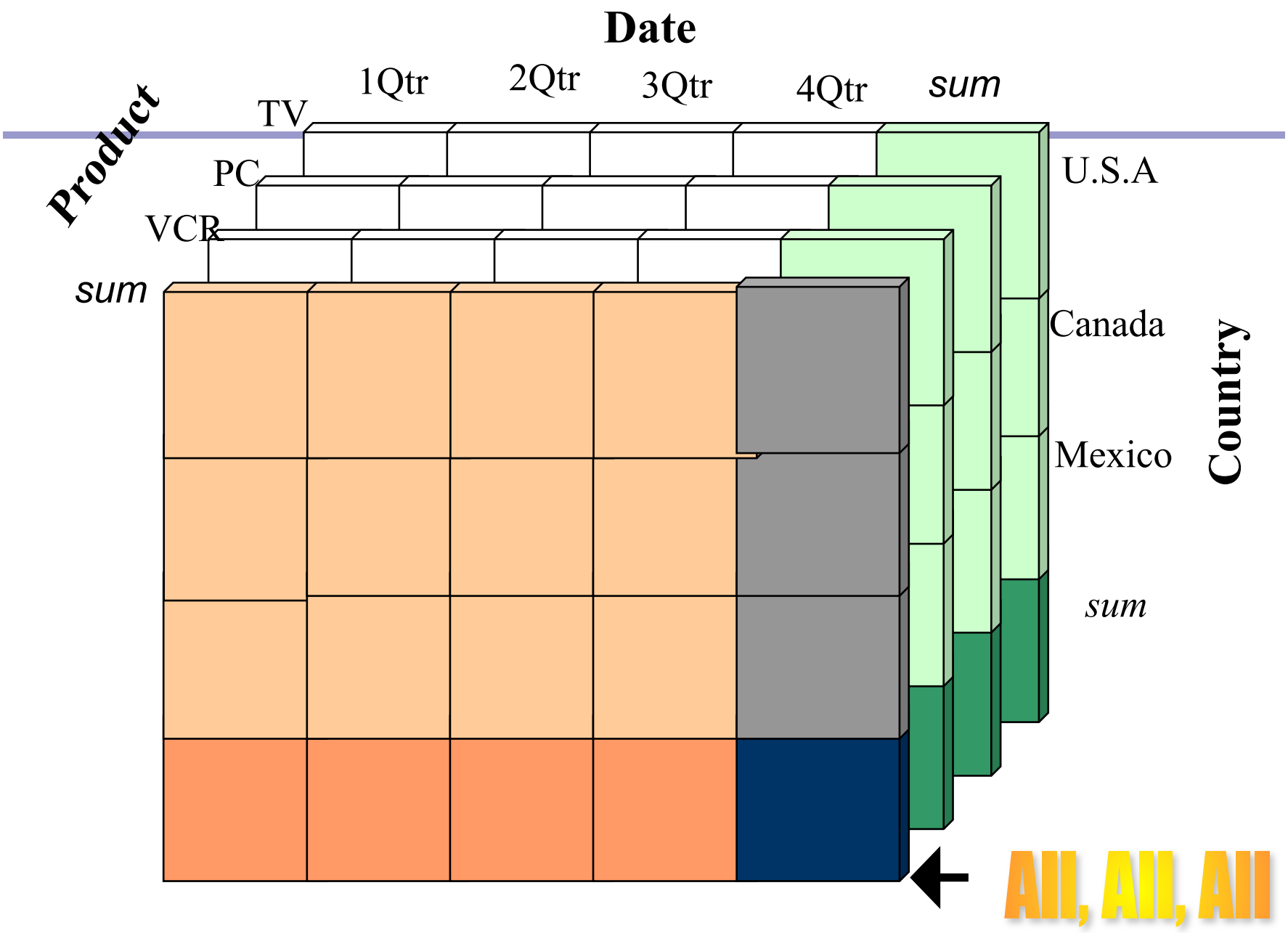
Measures: Their Categorization and Computation

- A data cube space can be defined by a set of dimension–value pairs; for example, time = “Q1”, location = “Vancouver”, item = “computer”)
- A data cube measure is a **numeric function** that can be evaluated at each point in the data cube space
- A 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

  based on the kind of aggregate functions used.

Data Cube Measures: Three Categories

- Distributive:
- An aggregate function is distributive if it can be computed in a distributed manner
- 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()


Date
1Qtr
2Qtr
3Qtr
4Qtr
sum
Product
TV
PC
VCR
sum
U.S.A
Canada
Mexico
sum
Country
All, All, All

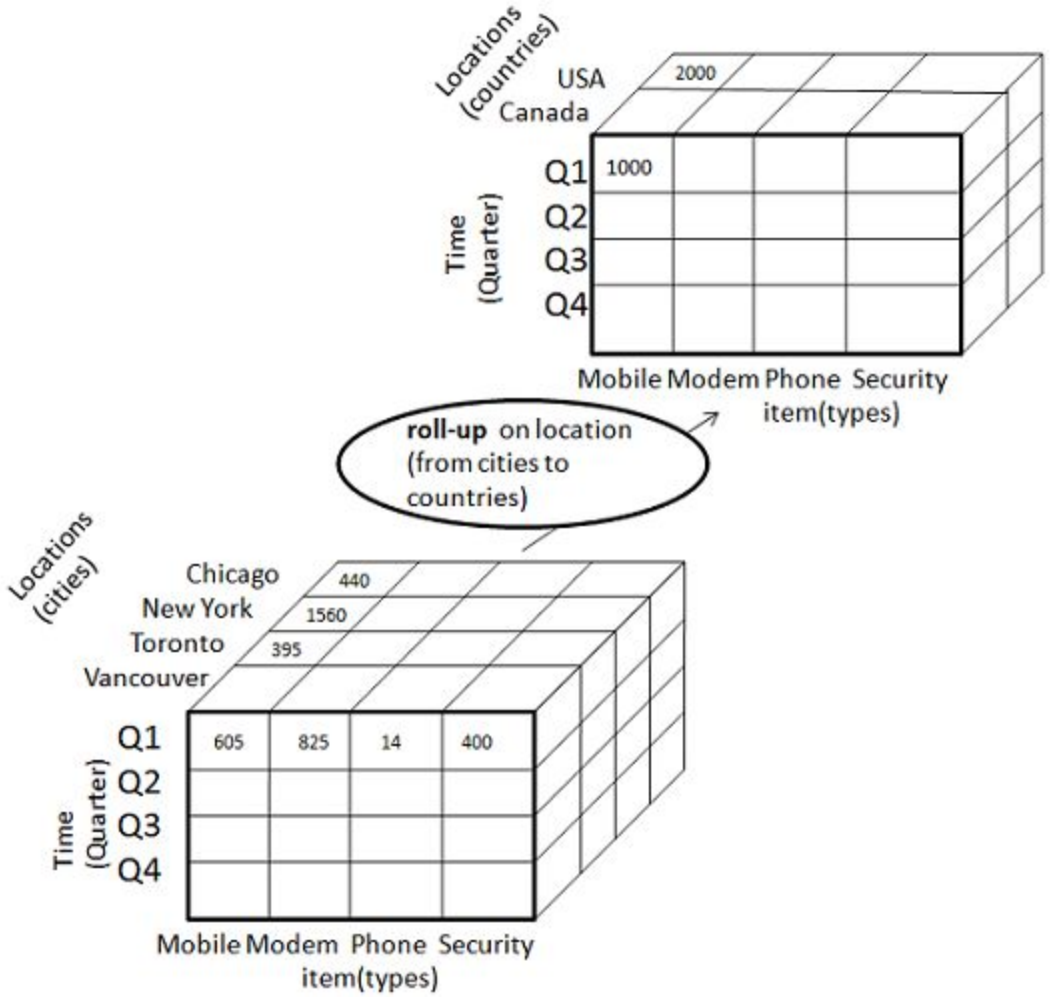
- Algebraic:
- If it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer)
- Each of which is obtained by applying a distributive aggregate function
- For example, avg() (average) can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions
    - E.g., avg(), min_N(), standard_deviation()

Typical OLAP Operations

- In the multidimensional model, data are organized into multiple dimensions
- 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

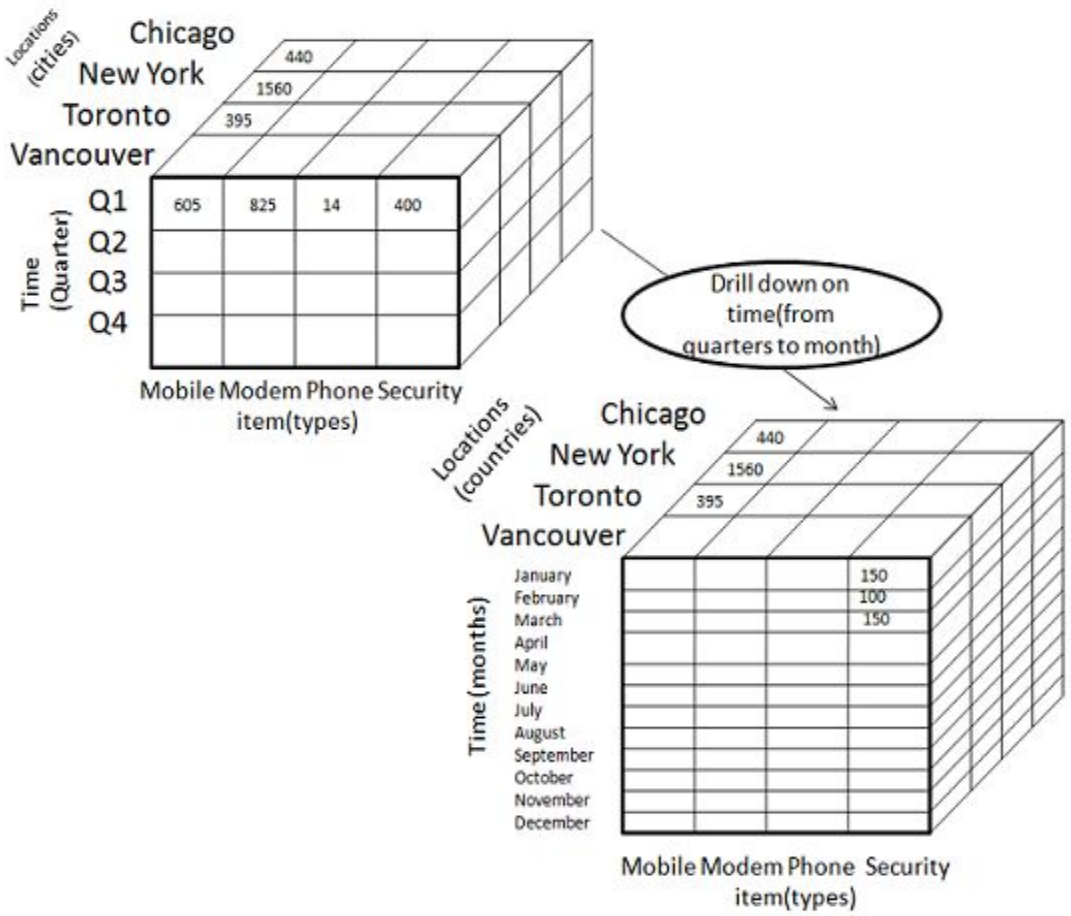
Typical OLAP Operations

- Roll up (drill-up):
- Also called the drill-up operation
- Performs aggregation on a data cube, either by climbi up a concept hierarchy for a dimension or by dimensio reduction

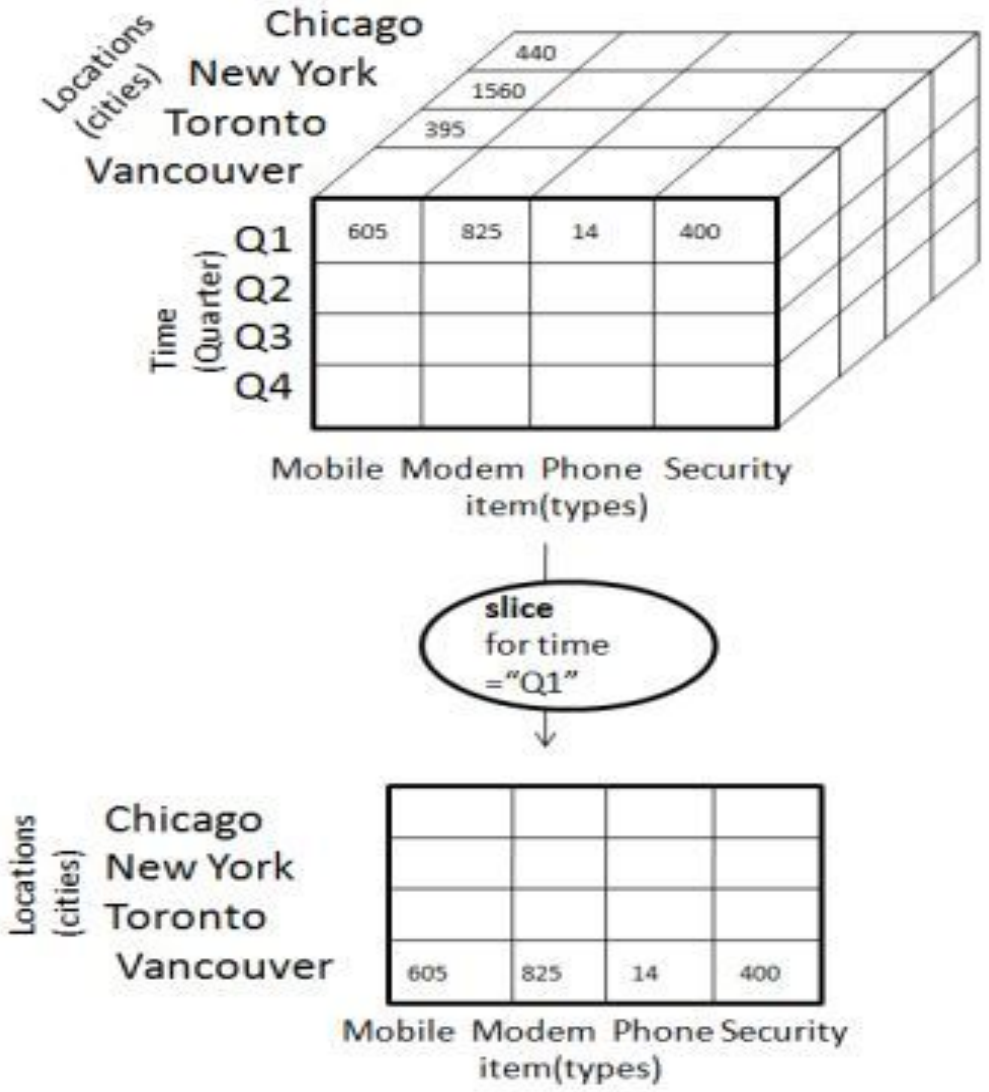


Drill down (roll down):

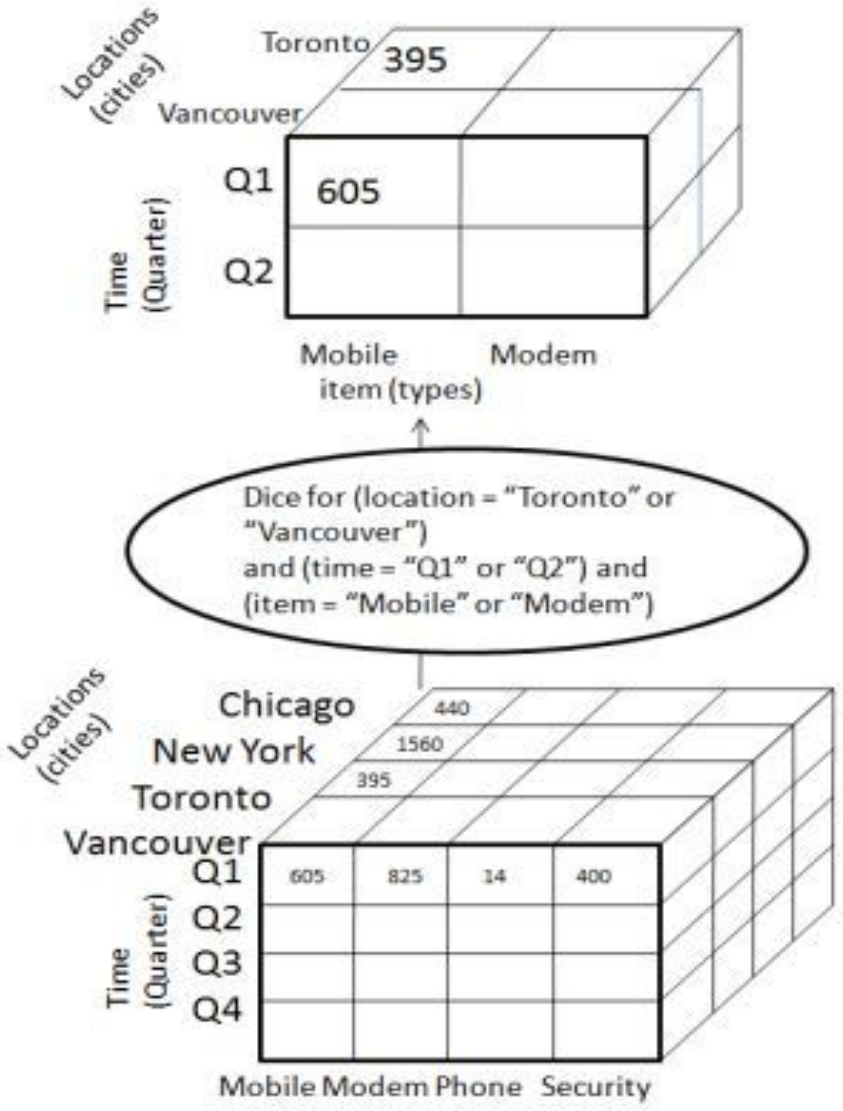
- Drill-down is the reverse operation of roll-up. It is performed by either of the following ways −
  - By stepping down a concept hierarchy for a dimension
  - By introducing a new dimension.



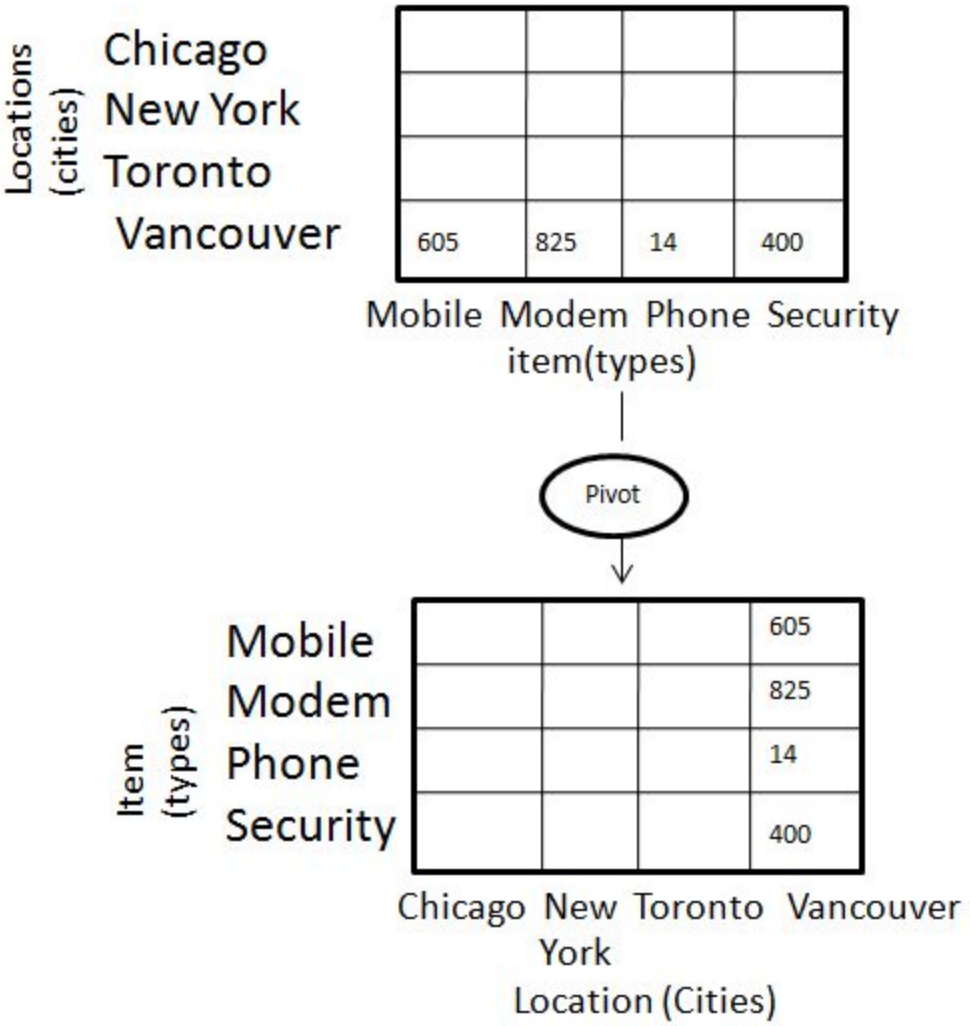
- Slice
- The slice operation selects **one particular dimension** from a given cube and provides a new sub-cube



- Dice:
- Dice selects **two or more dimensions** from a given cube and provides a new sub-cube.



- Pivot (rotate):
  - The pivot operation is also known as rotation.
  - It rotates the data axes in view in order to provide an alternative presentation of data.



- 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.

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.

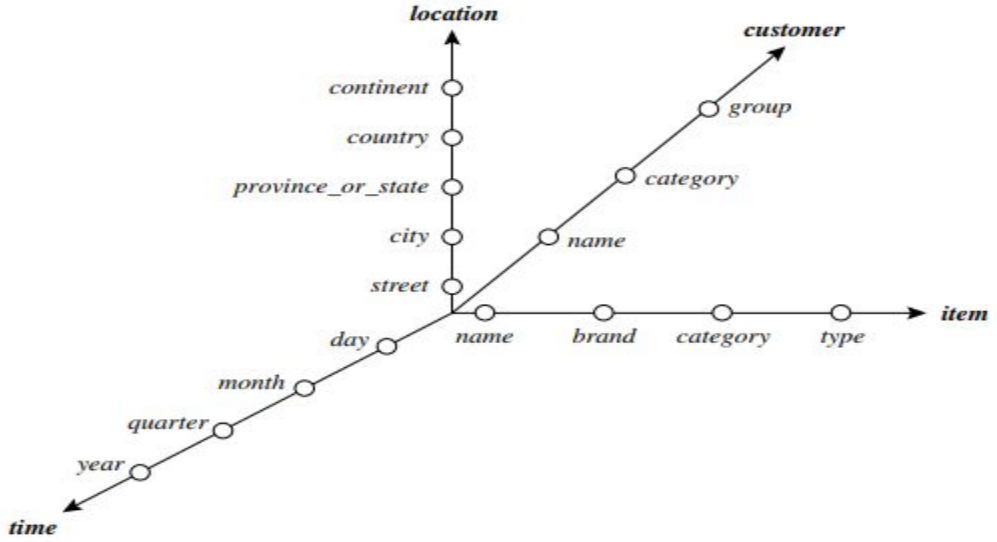
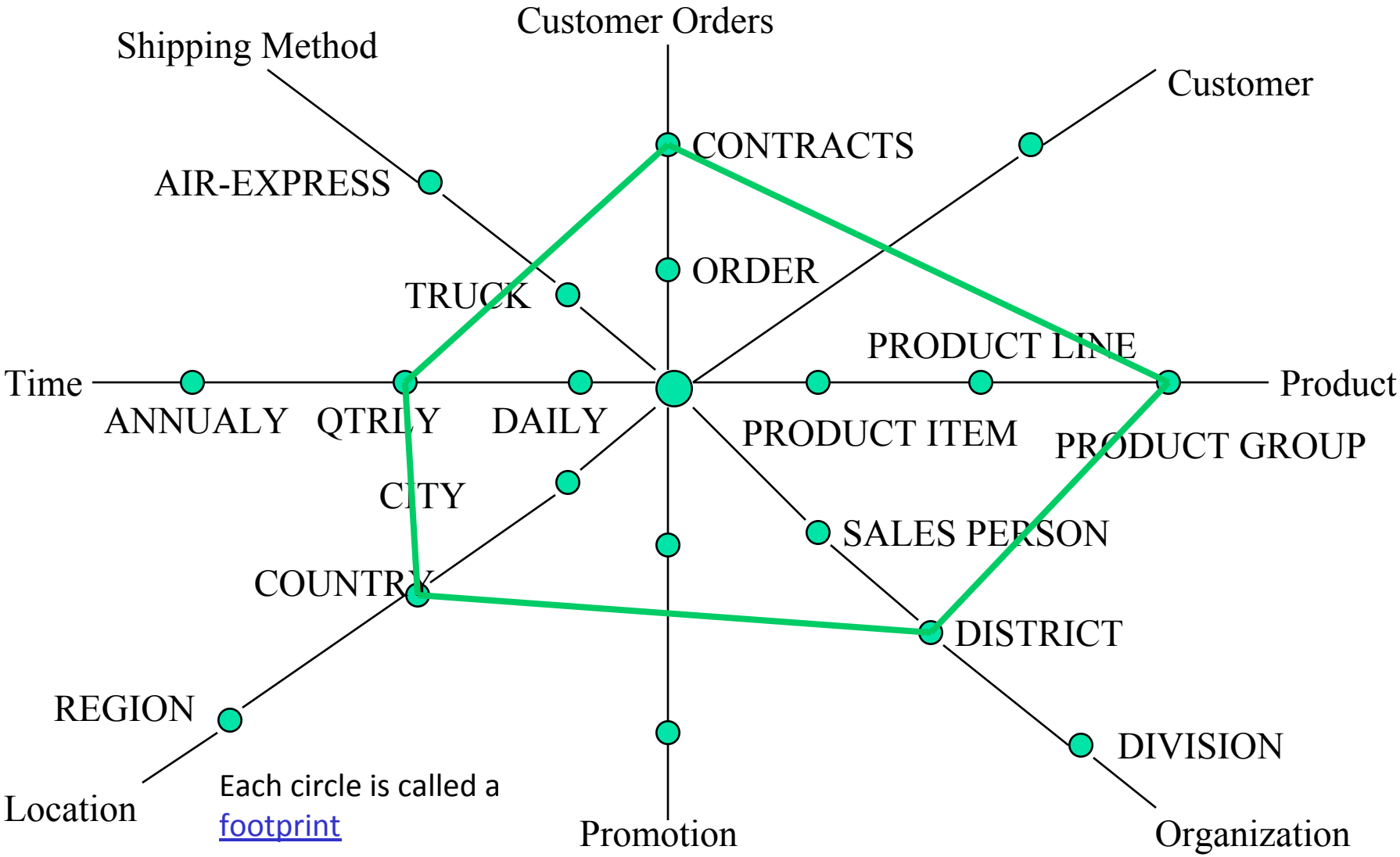


Figure 4.13 A starnet model of business queries.

A Star-Net Query Model



Each circle is called a [footprint]

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,
  - Query processing techniques

- 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.
- Each group-by can be represented by a cuboid, where the set of group-by forms a lattice of cuboids defining a data cube

The compute Cube Operator and 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

- **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 analyze the data, with queries such as the following:

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

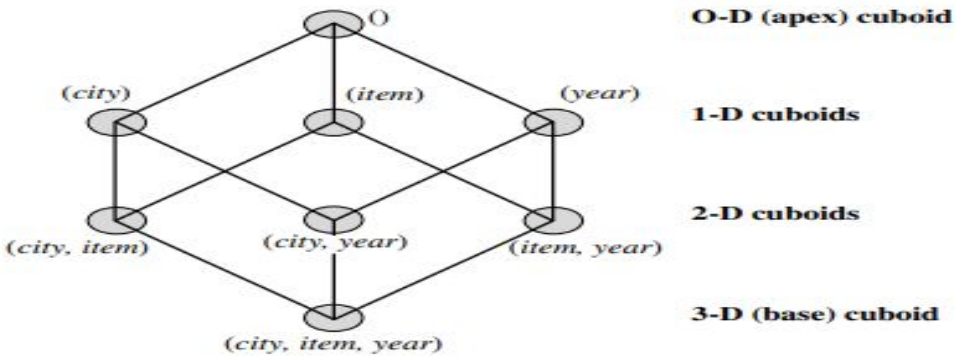


Figure 4.14 Lattice of cuboids, making up a 3-D data cube. Each cuboid represents a different group-by. The base cuboid contains *city*, *item*, and *year* dimensions.

*

- the total number of cuboids, or groupby, that can be computed for this data cube is 2^3 = 8.
- Similar to the SQL syntax, the data cube in Example could be defined as in DMQL

define cube sales_cube [city, item, year]: sum(sales in dollars)

compute cube sales_cube

- Online analytical processing may need to access different cuboids for different queries.
- Therefore, it may seem like a good idea to compute in advance all or at least some of the cuboids in a data cube.
- A major challenge related to this pre-computation, however, is that the required storage space ,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.

The "Compute Cube" Operator

- Cube definition and computation in DMQL

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

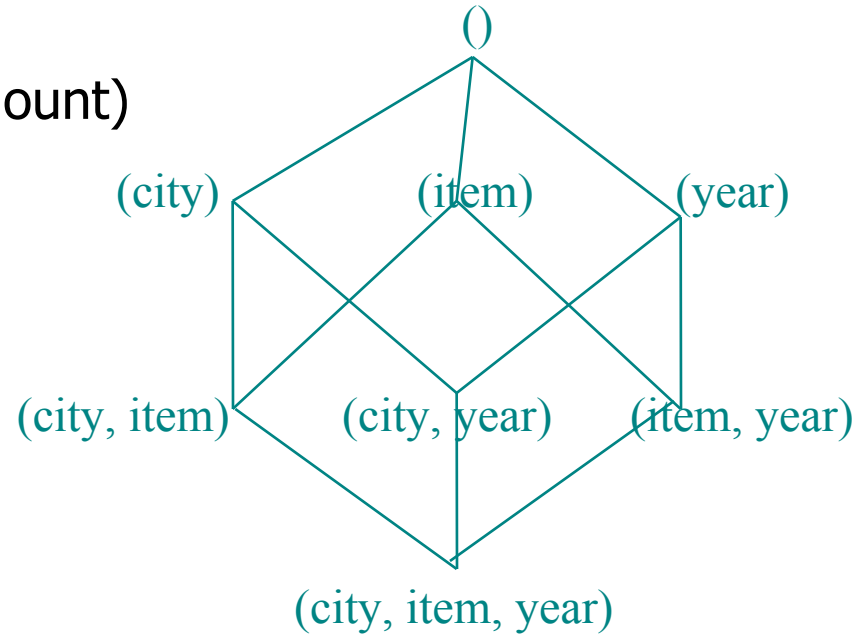
  compute cube sales

- 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 BY item, city, year



Sales table

Continent	Country	City	Unit_sold
Asia	China	Hong Kong	7000
Asia	China	Shanghai	300
Asia	Japan	Tokyo	5000
Europe	France	Paris	5000
Europe	UK	London	6000
Europe	UK	Manchester	12000
North America	Canada	Toranto	5000
North America	Canada	Toranto	10000
North America	Canada	Vancouuver	15000

Partial Materialization: Selected Computation of Cuboids

- There are three choices for data cube materialization
- **No materialization**:
  - Do not pre-compute any of the "nonbase" cuboids.
  - This leads to computing expensive multidimensional aggregates on-the-fly, which can be extremely slow.
- **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

- **Partial materialization**:
  - Selectively compute a proper subset of the whole set of possible cuboids.
  - compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion
  - Partial materialization represents an interesting trade-off between storage space and response time

- The partial materialization of cuboids or subcubes should consider three factors:

(1) Identify the subset of cuboids or subcubes to materialize;

(2) Exploit the materialized cuboids or subcubes during query processing

(3) Efficiently update the materialized cuboids or subcubes during load and refresh.

1.Identify the subset of cuboids or subcubes to materialize;

- The selection of the subset of cuboids or subcubes to materialize should take into account
  - The queries in the workload
  - Their frequencies
  - Their accessing costs
  - Workload characteristics
  - The cost for incremental updates,
  - The total storage requirements.

- Several OLAP products have adopted **heuristic approaches** for cuboid and subcube selection
  - 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 pre computing the cuboids for only a small number of dimensions (e.g., three to five) of a data cube

compute cube sales iceberg as

select month, city, customer_group, count(*)

from sales

cube by month, city, customer_group

having count(*) >= min_sup

Indexing OLAP Data : Bitmap Index and join Index

- To facilitate efficient data accessing, most data warehouse systems support index structures and materialized views (using cuboids).
- we examine how to index OLAP data by bitmap indexing and join indexing

Bitmap-Indexing

- 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 (i.e., there are n bit vectors).
- If the attribute has the value v for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0

- **Bitmap indexing.** In the AllElectronics data warehouse, suppose the dimension item at the top level has four values (representing item types): “home entertainment,” “computer,” “phone,” and “security.” Each value (e.g., “computer”) is represented by a bit vector in the item bitmap index table.

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.”

Figure 4.15 Indexing OLAP data using bitmap indices.

Advantage of Bitmap Index

- 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

Join Indexing

- The join indexing method gained popularity from its use in relational database query processing.
- Traditional indexing maps the value in a given column to a list of rows having that value.
- In contrast, join indexing registers the joinable rows of two relations from a relational database.
- For example, if two relations R(RID, A) and S(B, SID) join on the attributes A and B, then the join index record contains the pair (RID, SID), where RID and SID are record identifiers from the R and S relations, respectively.

- Hence, the join index records can identify joinable tuples without performing costly join operations.
- Join indexing is especially useful for maintaining the relationship between a foreign key2 and its matching primary keys, from the joinable relation
- Defined a star schema for AllElectronics of the form “sales star [time, item, branch, location]: dollars sold = sum (sales in dollars).”
- An example of a join index relationship between the sales fact table and the location and item dimension tables is shown in Figure 4.16. For example, the “Main Street” value in the location dimension table joins with tuples T57, T238, and T884 of the sales fact table. Similarly, the “Sony-TV” value in the item dimension table joins with tuples T57 and T459 of the sales fact table. The corresponding join index tables

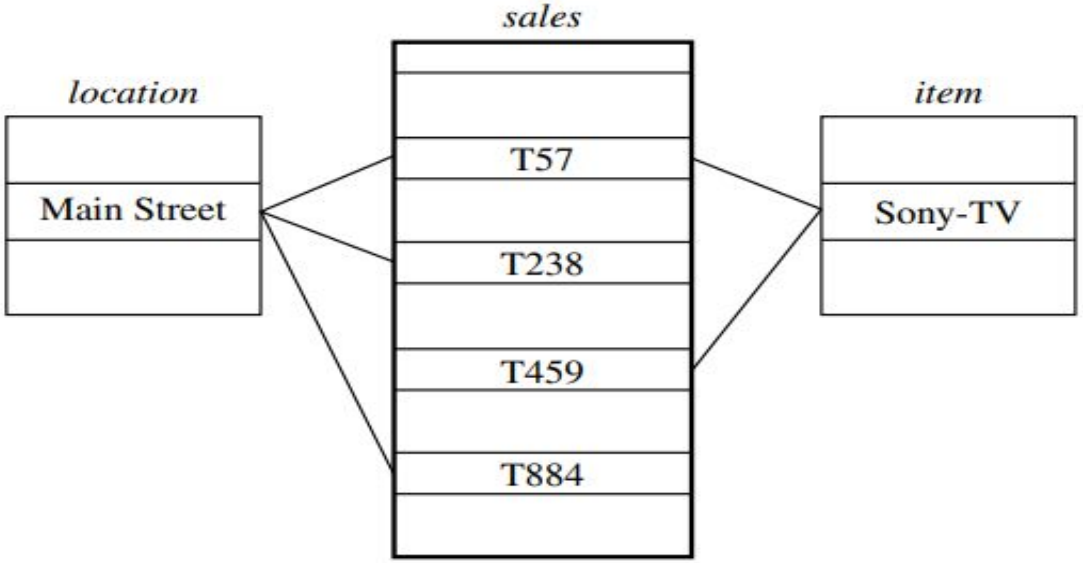


Figure 4.16 Linkages between a *sales* fact table and *location* and *item* dimension tables.

Join index table for *location/sales*

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

Join index table for *item/sales*

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

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

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

Figure 4.17 Join index tables based on the linkages between the *sales* fact table and the *location* and *item* dimension tables shown in Figure 4.16.

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.
- .

- **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,
- Selecting the cuboid with the least cost

Example 4.9 OLAP query processing.

Suppose that we define a data cube for *AllElectronics of the* form"*sales cube [time, item, location]: sum(sales in dollars)." The dimension hierarchies*used are

"*day < month < quarter < year" for time;*

"item name < brand < type" for item; and

"street < city < province or state < country" for location.

Suppose that the query to be processed is on *brand, province or state, with the* selection constant "*year = 2010."* *Also, suppose that there are four materialized cuboids* available, as follows:

cuboid 1: *year, item name, city*

cuboid 2: *year, brand, country*

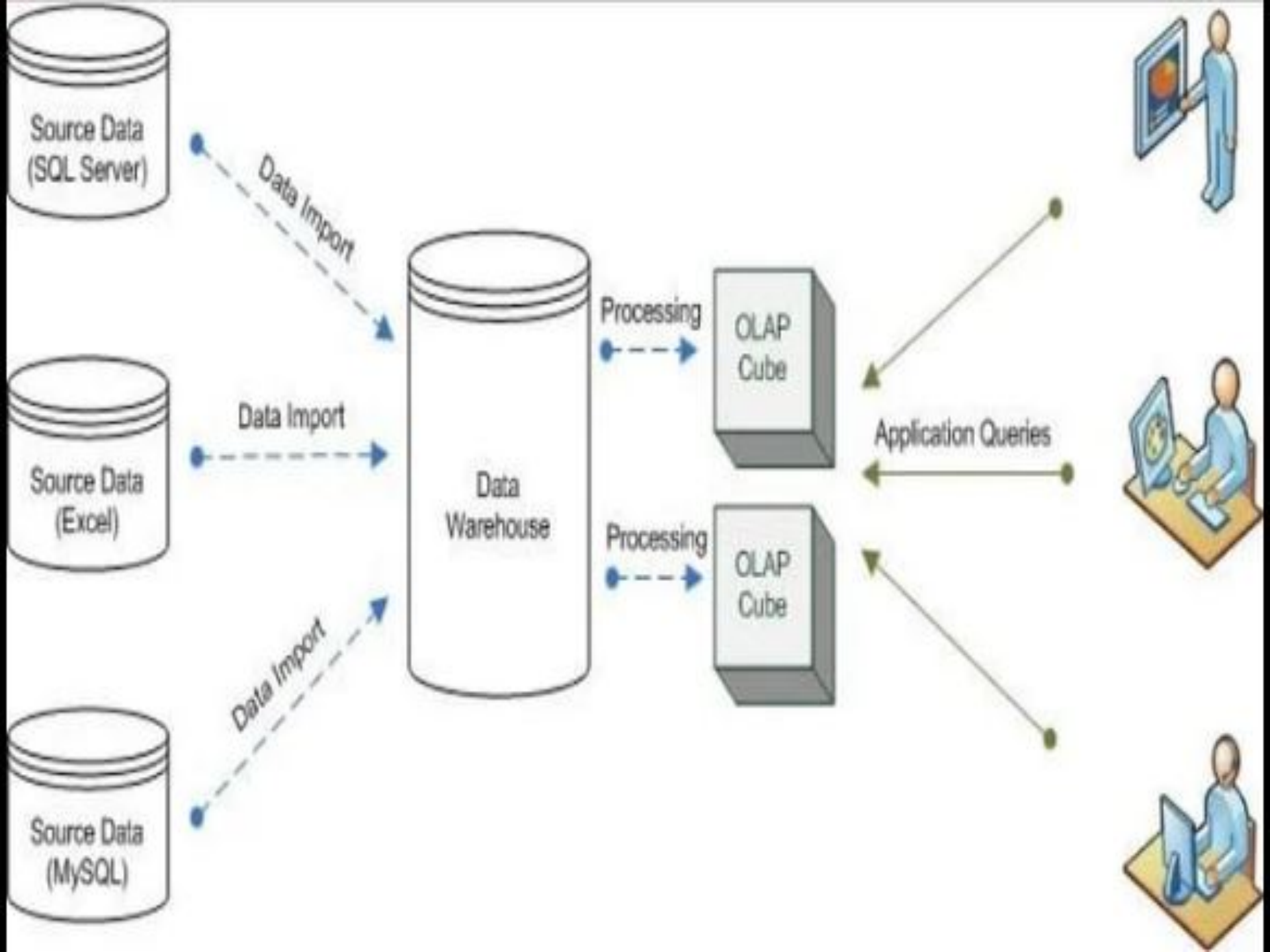
cuboid 3: *year, brand, province or state*

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

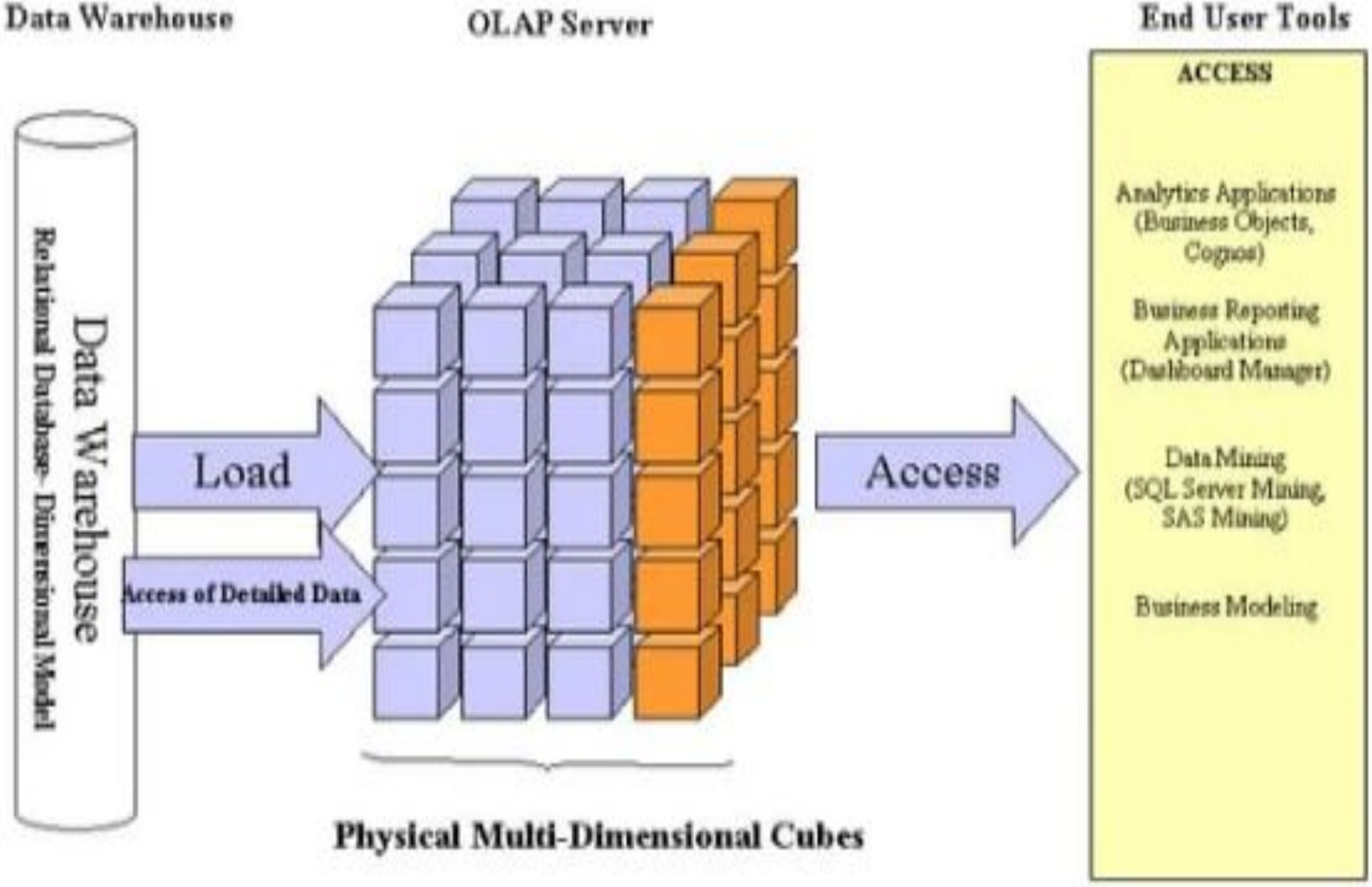
- OLAP (Online Analytical Processing) was introduced into the business intelligence (BI) space over 20 years ago, in a time where computer hardware and software technology weren't nearly as powerful as they are today.
- OLAP introduced a ground breaking way for business users (typically analysts) to easily perform multidimensional analysis of large volumes of business data.
- Aggregating, grouping, and joining data are the most difficult types of queries for a relational database to process.
- The magic behind OLAP derives from its ability to pre-calculate and pre-aggregate data. Otherwise, end users would be spending most of their time waiting for query results to be returned by the database

- **Limitations of OLAP cubes**
- OLAP requires restructuring of data into a star/snowflake schema
- There is a limited number of dimensions (fields) a single OLAP cube
- It is nearly impossible to access transactional data in the OLAP cube
- Changes to an OLAP cube requires a full update of the cube – a lengthy process

- OLAP is computer processing that enable user to easily and selectively extract and view data from different pint of view
- OLAP aw user to analyze database information from multiple database system at the same time
- First product that perform OLAP query as express which was release in 1970 and acquired by oracle in 1995
- Some popular OLAP server software include
- Oracle express server
- Hyperion solution Essbase


Source Data
(SQL Server)
Data Import
Source Data
(Excel)
Data Import
Source Data
(MySQL)
Data Import
Data
Warehouse
Processing
OLAP
Cube
Processing
OLAP
Cube
Application Queries

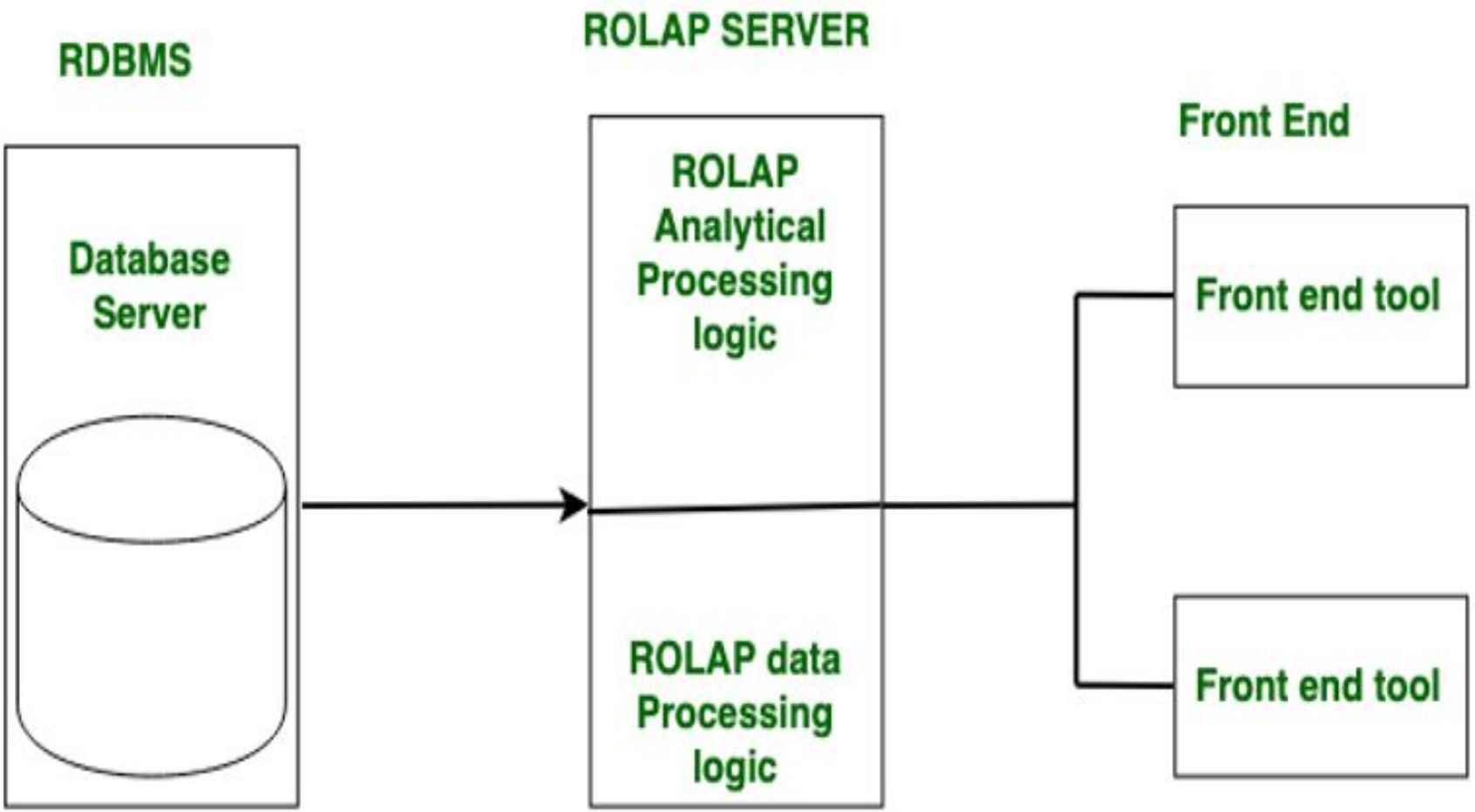
- OLAP cube is a data structure that allow fast analysis of data
- Arrangement of data into cube overcome the imitation f RDBMS
- OLAP contain numeric fact called measure which are categories by dimension
- Purpose of OLAP cube is to derived summarizes information from large volume f data
- OLAP is significant improvement on query system


Data Warehouse
OLAP Server
End User Tools
ACCESS
Analytics Applications (Business Objects, Cognos)
Business Reporting Applications (Dashboard Manager)
Data Mining (SQL Server Mining, SAS Mining)
Business Modeling
Data Warehouse
Relational Database- Dimensional Model
Load
Access of Detailed Data
Access
Physical Multi-Dimensional Cubes

- Vendors offer a variety of OLAP products that can be grouped into three categories
- ROLAP(Relational)
- MOLAP(Multi-dimensional)
- HOLAP(Hybrid)

ROLAP

- ROLAP stands for Relational Online Analytical Processing.
- ROLAP stores data in columns and rows (also known as relational tables) and retrieves the information on demand through user submitted queries.
- A ROLAP database can be accessed through complex SQL queries to calculate information.
- ROLAP can handle large data volumes, but the larger the data, the slower the processing times.
- ROLAP servers are used to bridge the gap between the relational back-end server and the client's front-end tools.
- Because queries are made on-demand, ROLAP does not require the storage and pre-computation of information.


RDBMS
ROLAP SERVER
Front End
Database Server
ROLAP Analytical Processing logic
ROLAP data Processing logic
Front end tool
Front end tool

Advantage

- It is compatible with data warehouses and OLTP systems.
- The data size limitation of ROLAP technology is determined by the underlying RDBMS. As a result, ROLAP does not limit the amount of data that can be stored.

Limitations:

- SQL functionality is constrained.
- It's difficult to keep aggregate tables up to date.

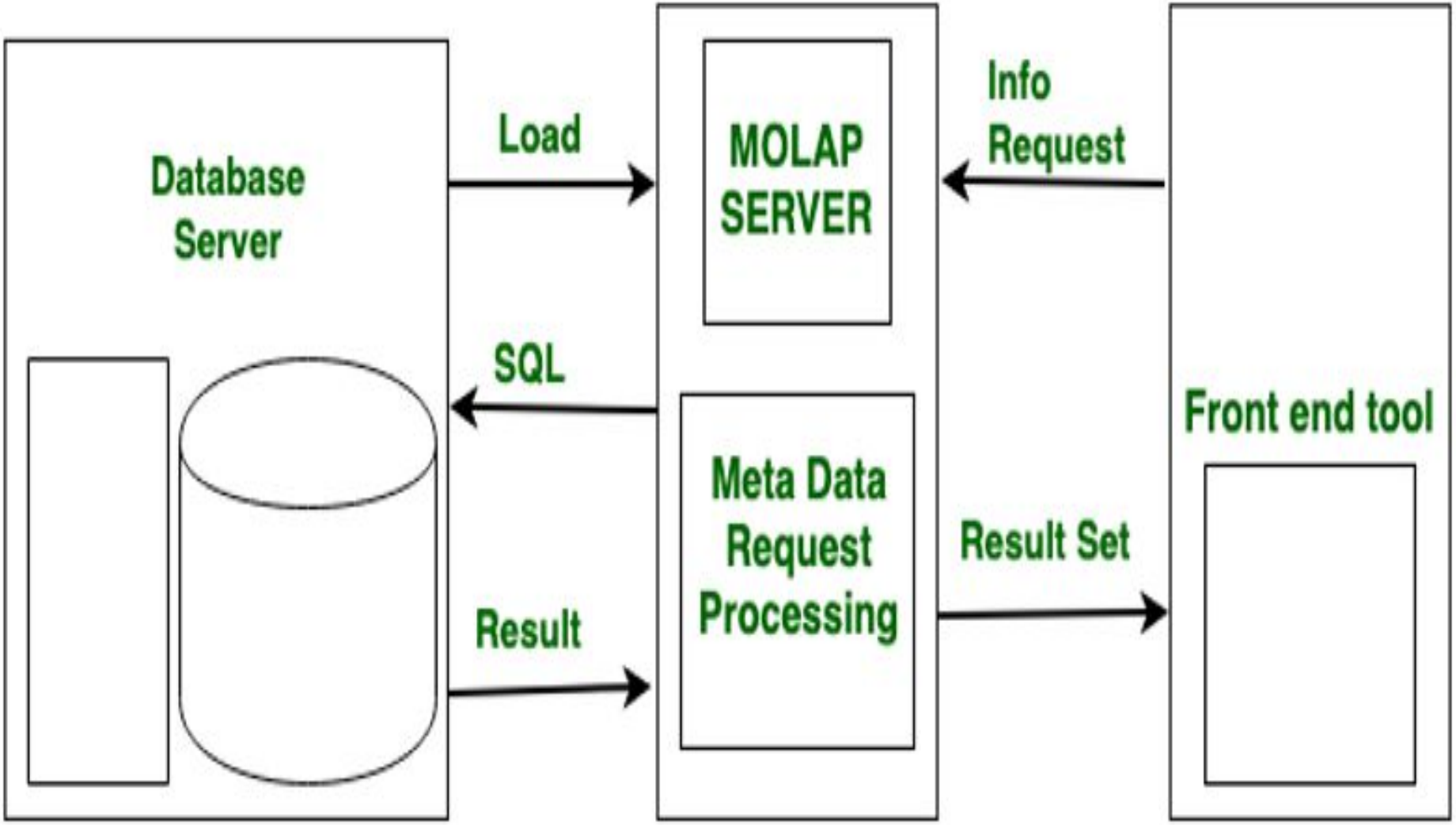
Examples of popular ROLAP products include

- Metacube by Stanford Technology Group
- Red Brick Warehouse by Red Brick Systems

MOLAP

- MOLAP stands for Multidimensional Online Analytical Processing.
- Array-based multidimensional storage engines
- MOLAP uses a multidimensional cube that accesses stored data through various combinations.
- Data is pre-computed, pre-summarized, and stored (a difference from ROLAP, where queries are served on-demand).
- In this approach, a series of dense, small, pre-calculated cubes make up a hypercube.
- Tools that incorporate MOLAP include
  - Oracle Essbase
  - IBM Cognos
  - Apache Kylin.

- Its simple interface makes MOLAP easy to use, even for in experienced users.
- Its speedy data retrieval makes it the best for “slicing and dicing” operations.
- One major disadvantage of MOLAP is that it is less scalable than ROLAP, as it can handle a limited amount of data.


Database
Server
Load
MOLAP
SERVER
Info
Request
SQL
Meta Data
Request
Processing
Front end tool
Result Set
Result

Benefits:

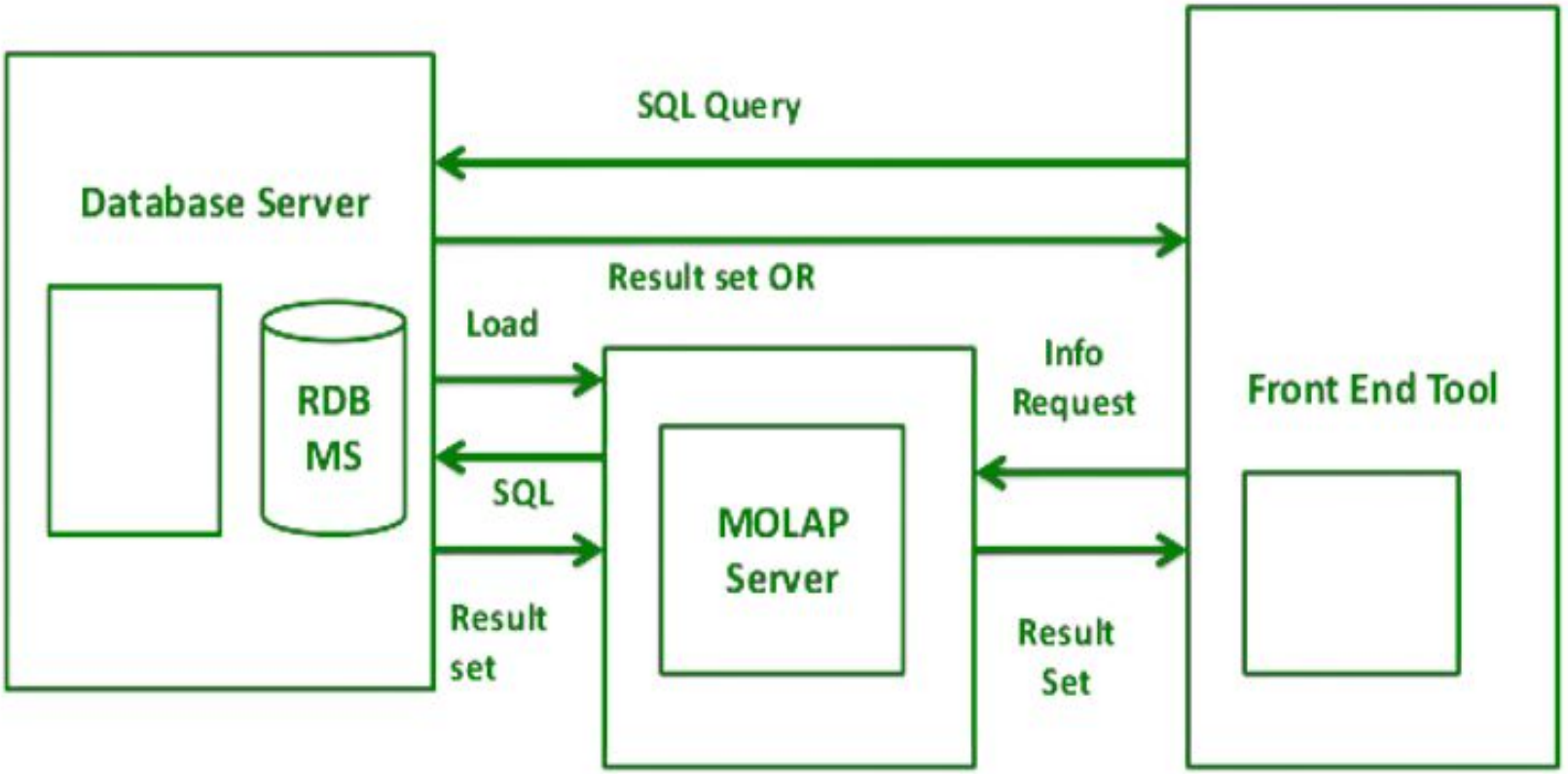
- Suitable for slicing and dicing operations.
- Outperforms ROLAP when data is dense.
- Capable of performing complex calculations.

Limitation

- It is difficult to change the dimensions without re-aggregating.
- Since all calculations are performed when the cube is built, a large amount of data cannot be stored in the cube itself.

HOLAP

- HOLAP stands for Hybrid Online Analytical Processing.
- As the name suggests, the HOLAP storage mode connects attributes of both MOLAP and ROLAP.
- Since HOLAP involves storing part of your data in a ROLAP store and another part in a MOLAP store, developers get the benefits of both.
- With this use of the two OLAPs, the data is stored in both multidimensional databases and relational databases.
- The decision to access one of the databases depends on which is most appropriate for the requested processing application or type.
- This setup allows much more flexibility for handling data.


SQL Query
Database Server
Result set OR
Load
RDB
MS
Info
Request
Front End Tool
SQL
MOLAP
Server
Result
set
Result
Set

- Microsoft Analysis Services and SAP AG BI Accelerator are products that run off HOLAP

Benefits:

- HOLAP combines the benefits of MOLAP and ROLAP.
- Provide quick access at all aggregation levels.

Limitations

- Because it supports both MOLAP and ROLAP servers, HOLAP architecture is extremely complex.
- There is a greater likelihood of overlap, particularly in their functionalities.

Specialized SQL Servers

- Specialized SQL servers provides advanced Query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.