B.S. Abdur Rahman
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956
GST Road, Vandalur, Chennai 600 048

COURSE CODE: CSD 3104

COURSE NAME: DATA MINING AND DATA WAREHOUSING

MODULE-1

TOPICS: Data Warehouse – Basic Concepts – Data Cube – Schemas – OLAP and OLTP operations – Design and Implementation

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."—W. H. Inmon
- Data warehousing:
  - The process of constructing and using data warehouses

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, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - 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*

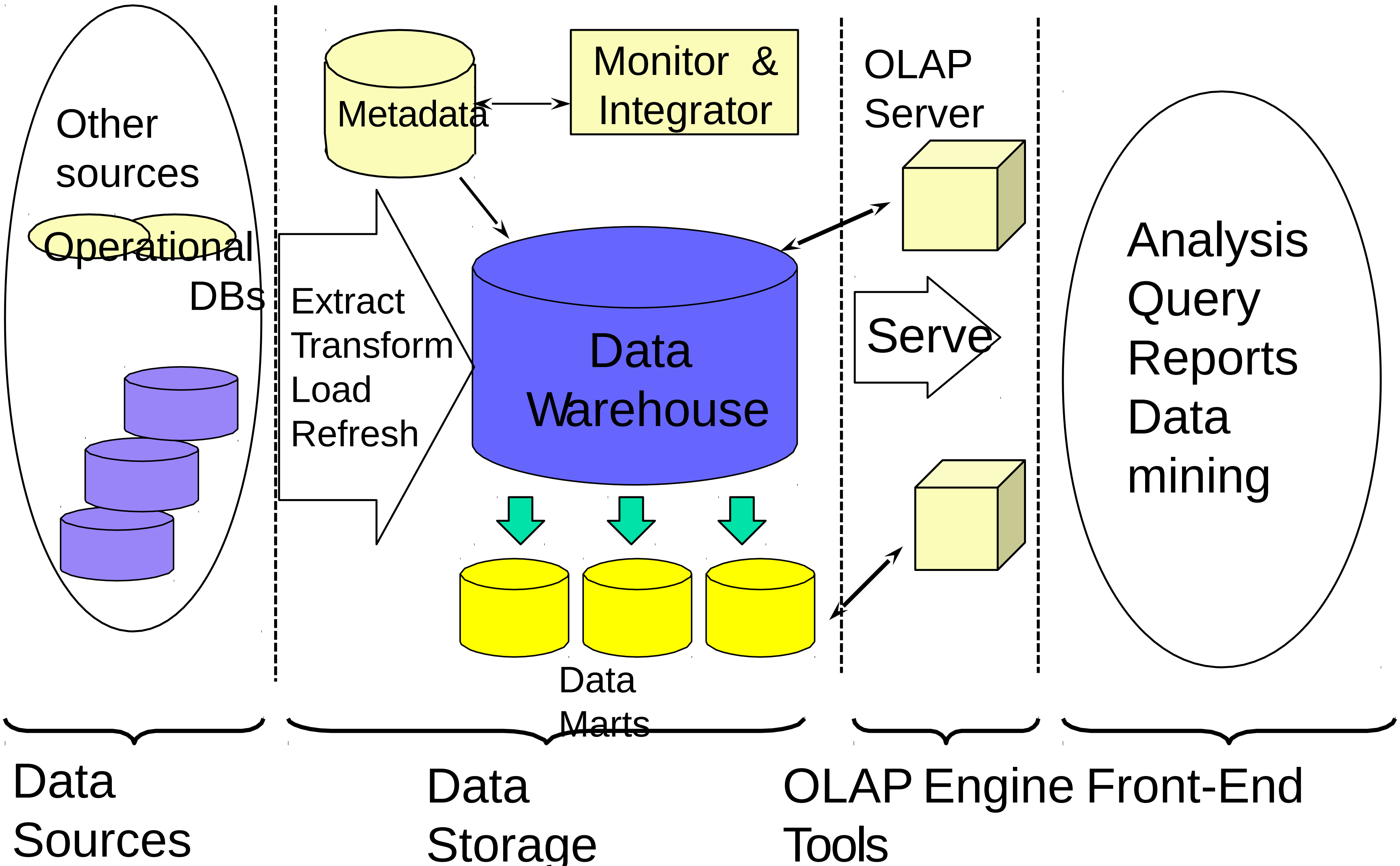
OLTP vs. OLAP

	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
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

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 (captured from external sources) Dependent (directly from company datawarehouse)
- 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, compute views, check integrity, and build indices and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

Metadata Repository

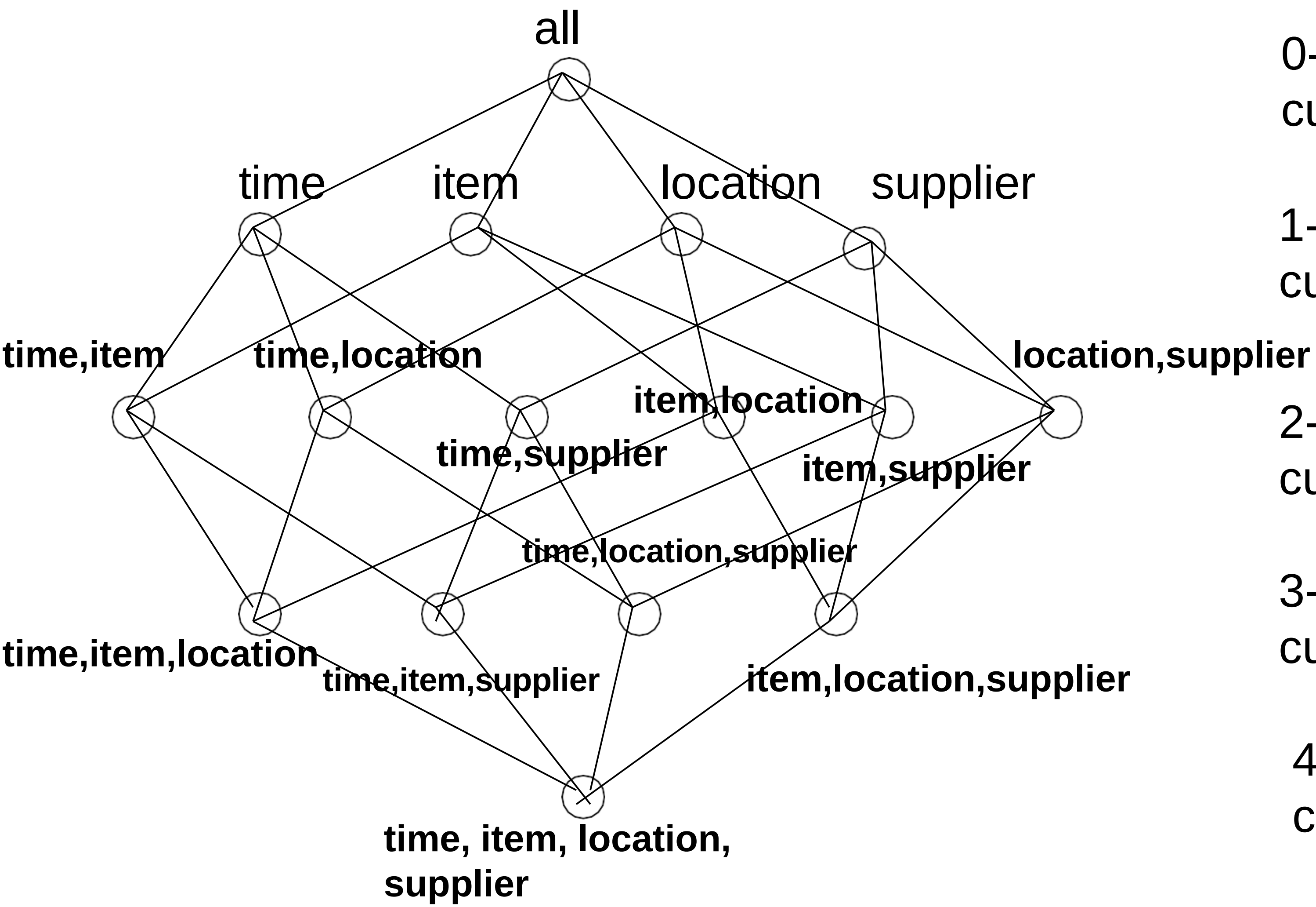
- **Meta data** is the data defining warehouse objects.
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
- Operational meta-data
  - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarization
- The mapping from operational environment to the DW Data
- related to system performance
  - warehouse schema, view and derived data definitions
- Business data
  - business terms and definitions, ownership of data ...

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 Lattice of Cuboids



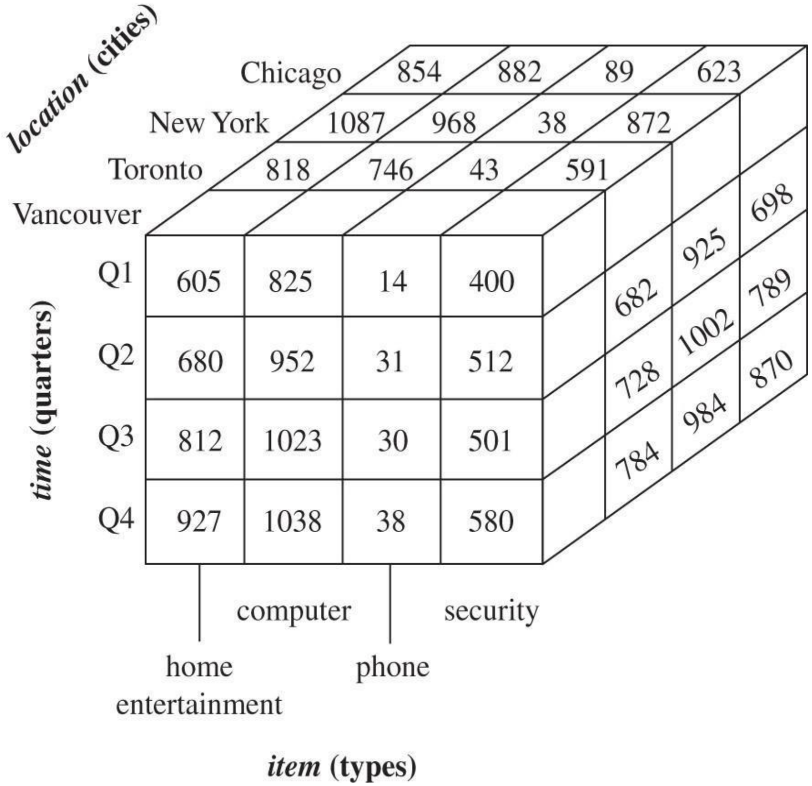
0-D (*apex*) cuboid

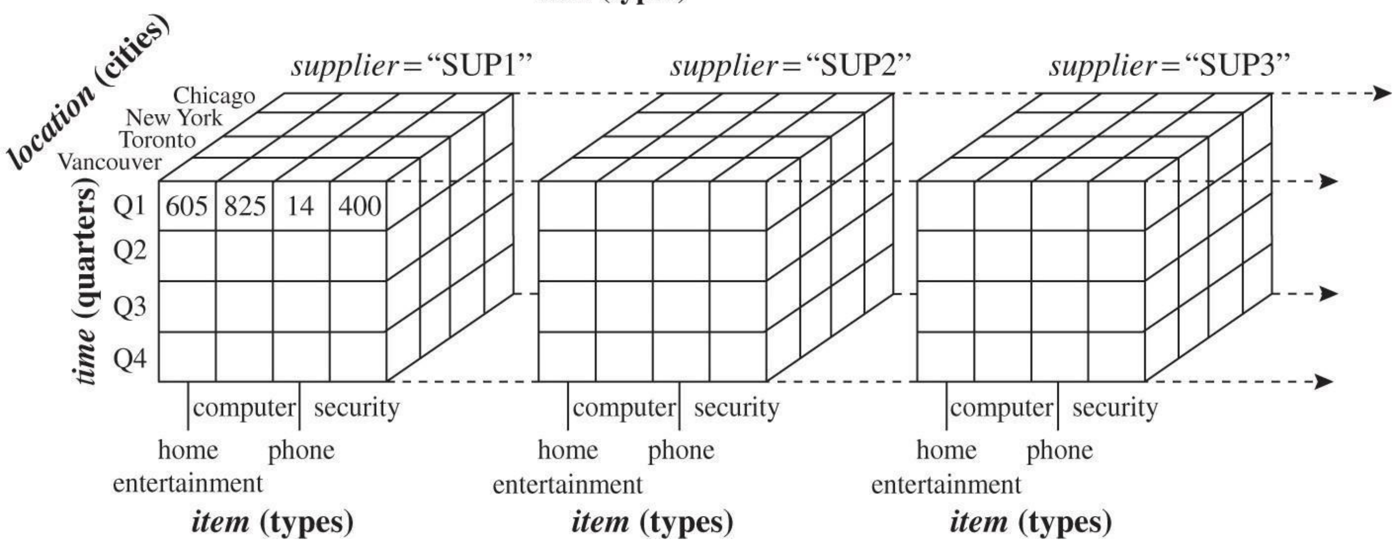
1-D cuboids

2-D cuboids

3-D cuboids

4-D (*base*) cuboid

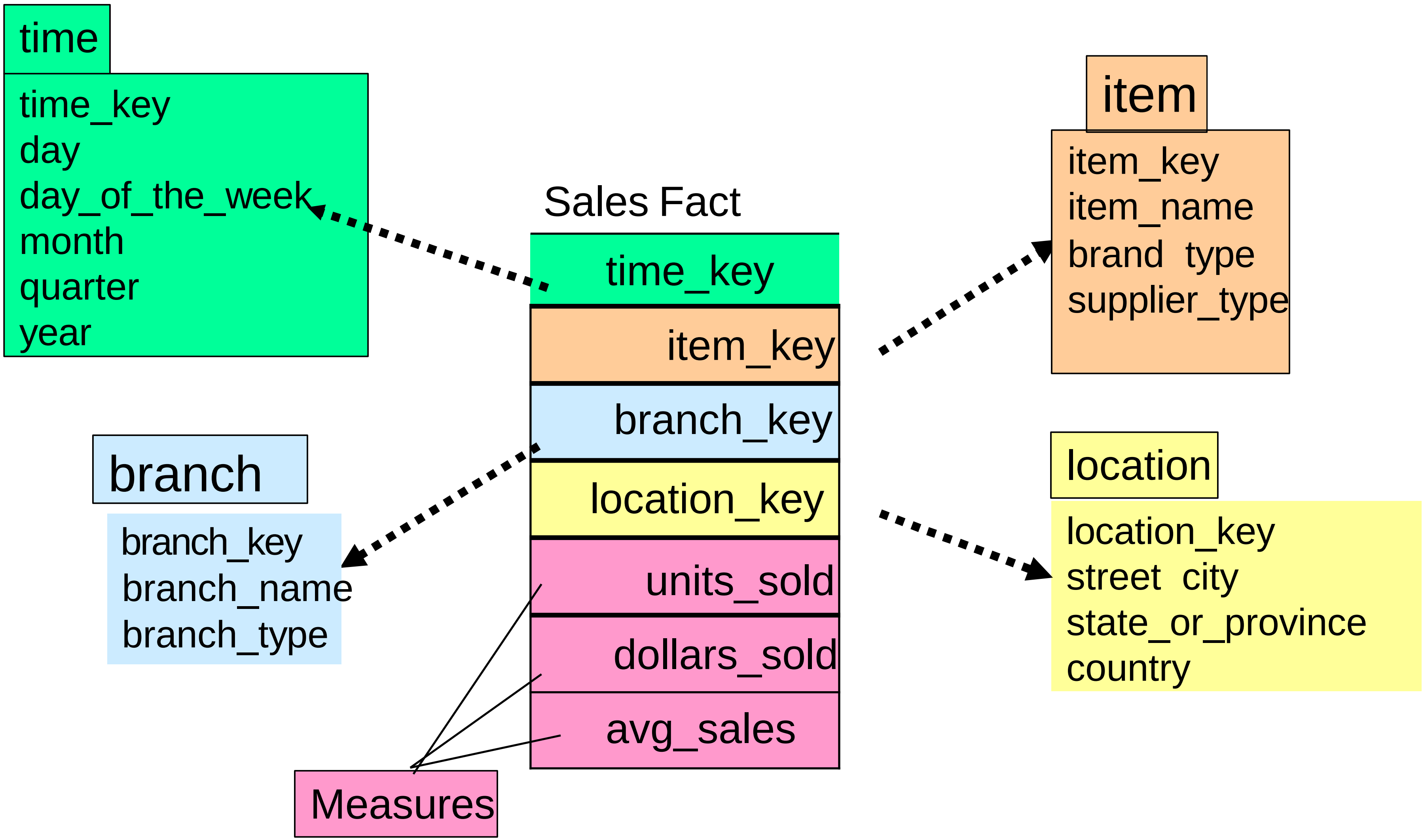

location (cities)
Chicago
854
882
89
623
New York
1087
968
38
872
Toronto
818
746
43
591
Vancouver
698
925
789
682
1002
870
728
984
784
Q1
605
825
14
400
Q2
680
952
31
512
Q3
812
1023
30
501
Q4
927
1038
38
580
time (quarters)
computer
security
home
entertainment
phone
item (types)


location (cities)
supplier = "SUP1"
supplier = "SUP2"
supplier = "SUP3"
Chicago
New York
Toronto
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
time (quarters)
computer
security
home
entertainment
phone
item (types)
computer
security
home
entertainment
phone
item (types)
computer
security
home
entertainment
phone
item (types)

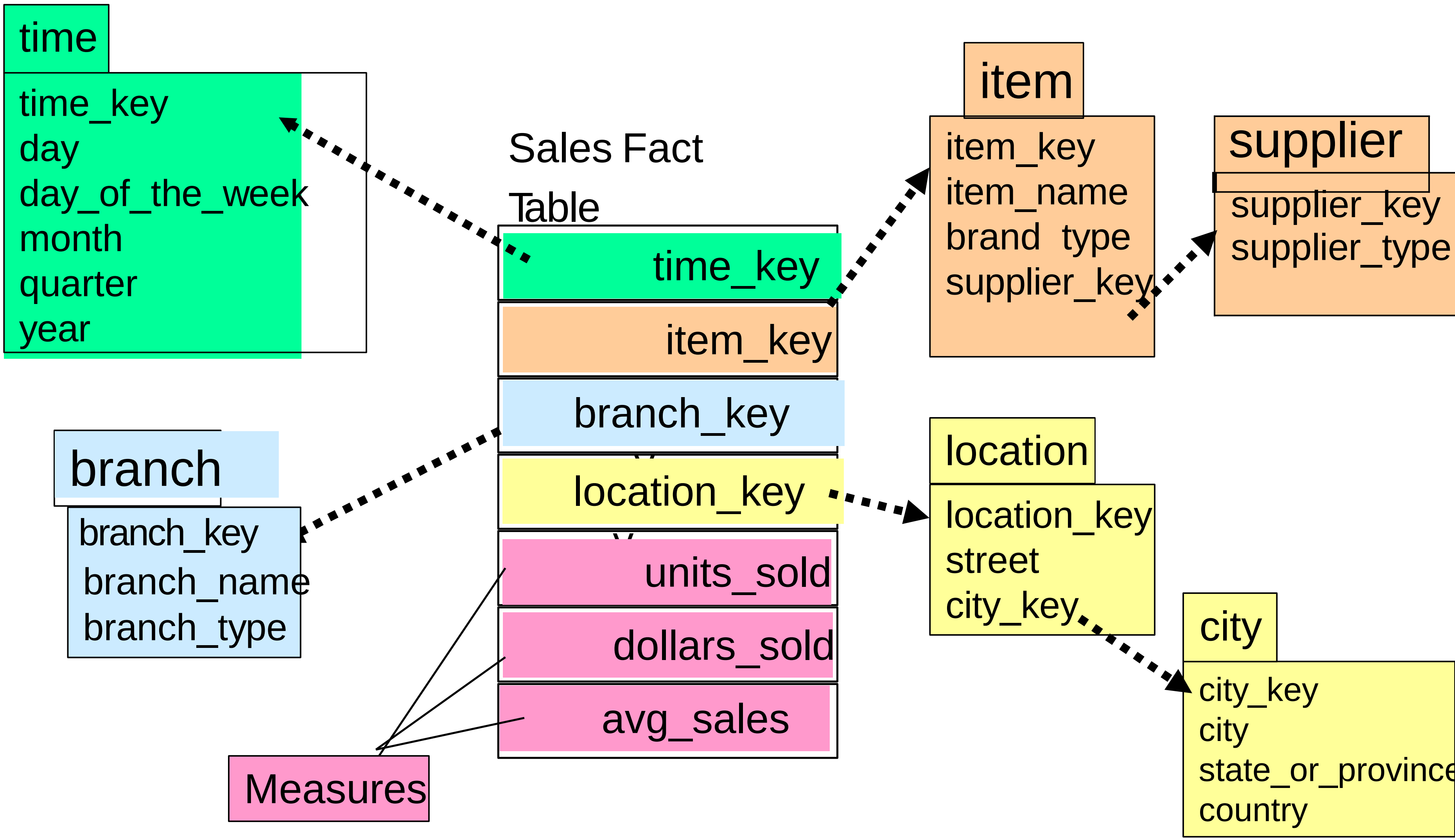
Conceptual Modeling of Datawarehouses

- Modeling datawarehouses: dimensions and measures
  measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

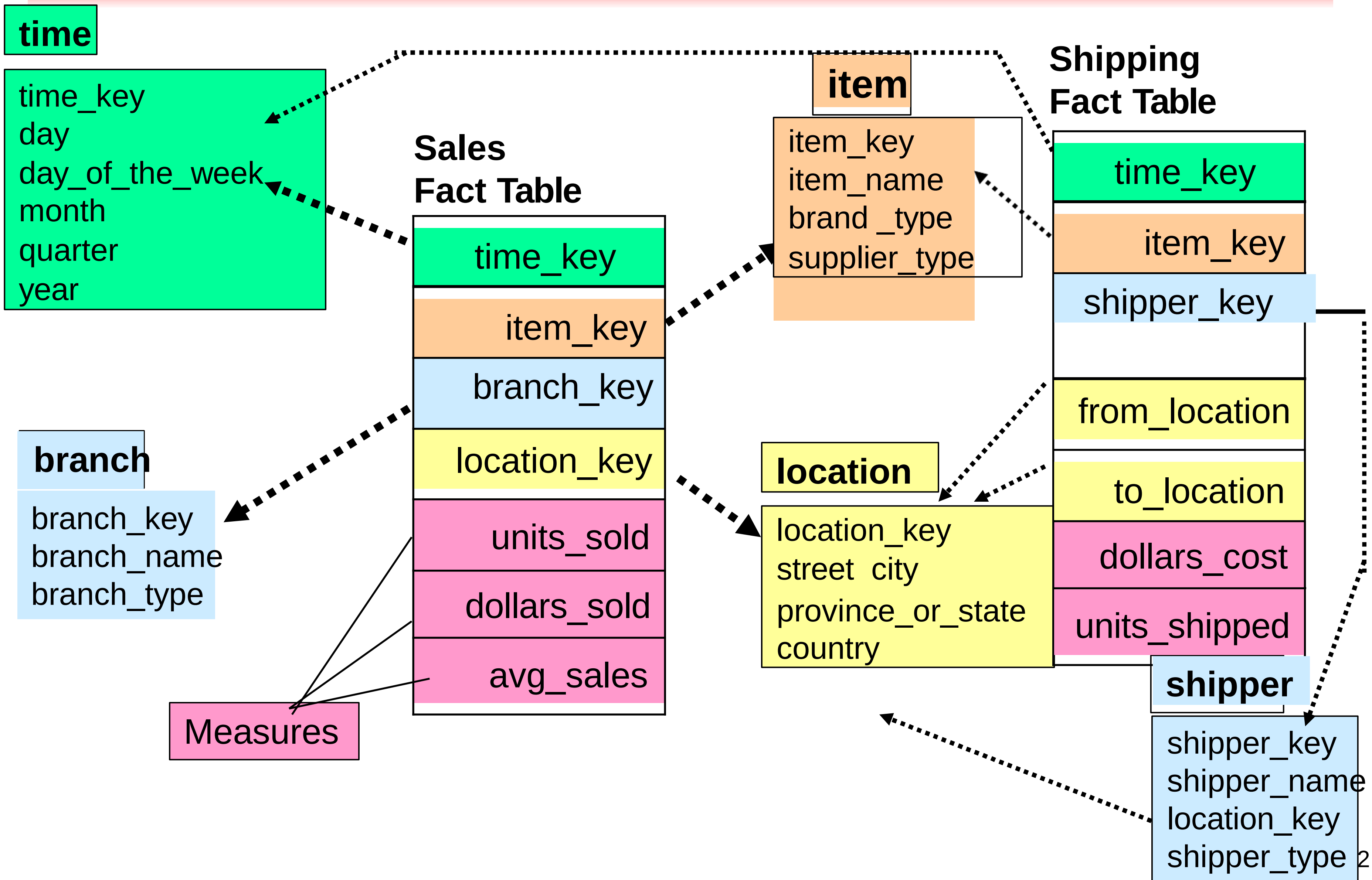
Example of **Star Schema**



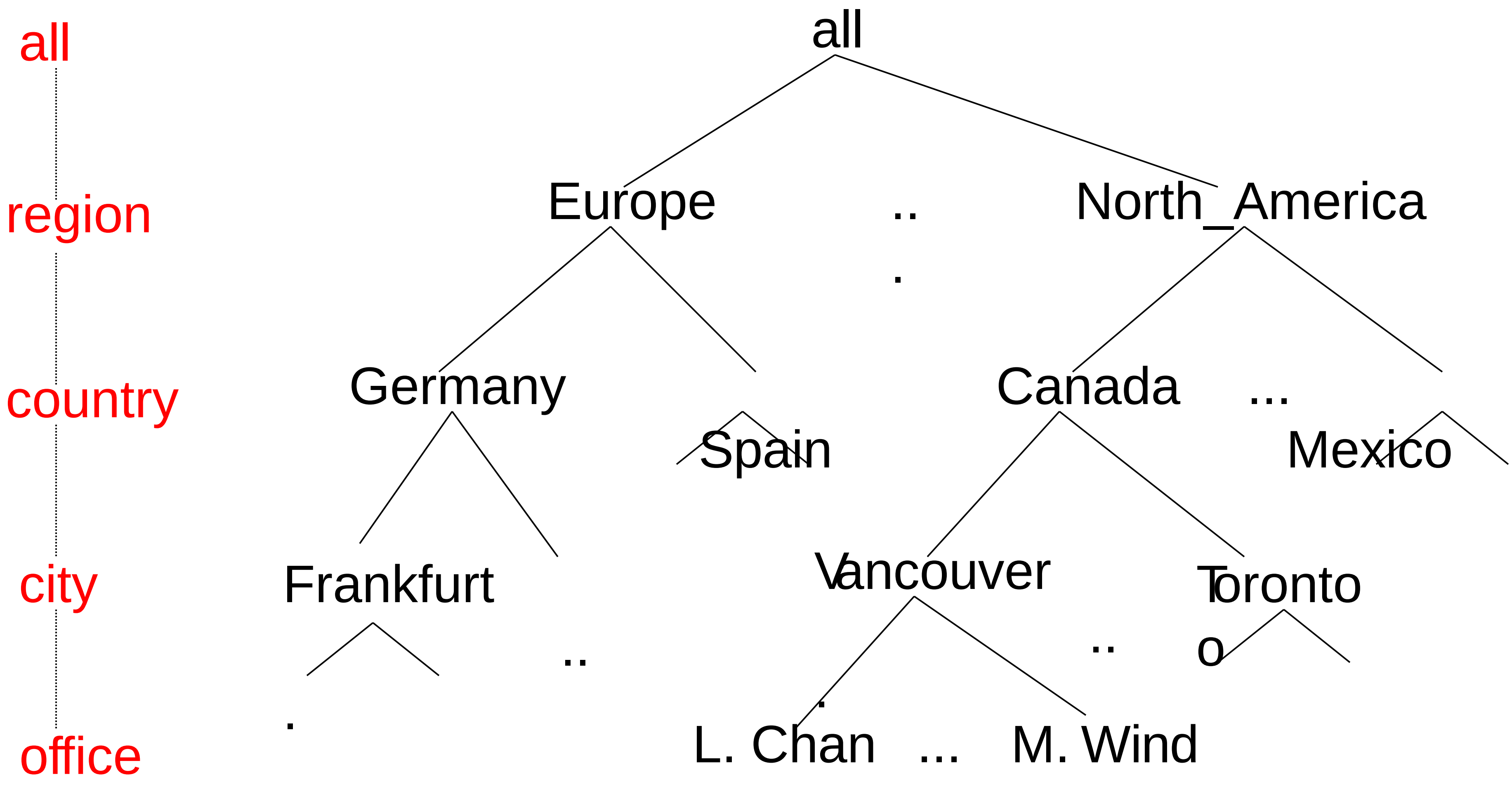
Example of **Snowflake Schema**



Example of **Fact Constellation**



A Concept Hierarchy: **Dimension** (location)



Data cube measures

- Measure: a numeric function that can be evaluated at each point in the data cube space:
  - Fact
  - Aggregation of facts

Data Cube Measures: Three Categories

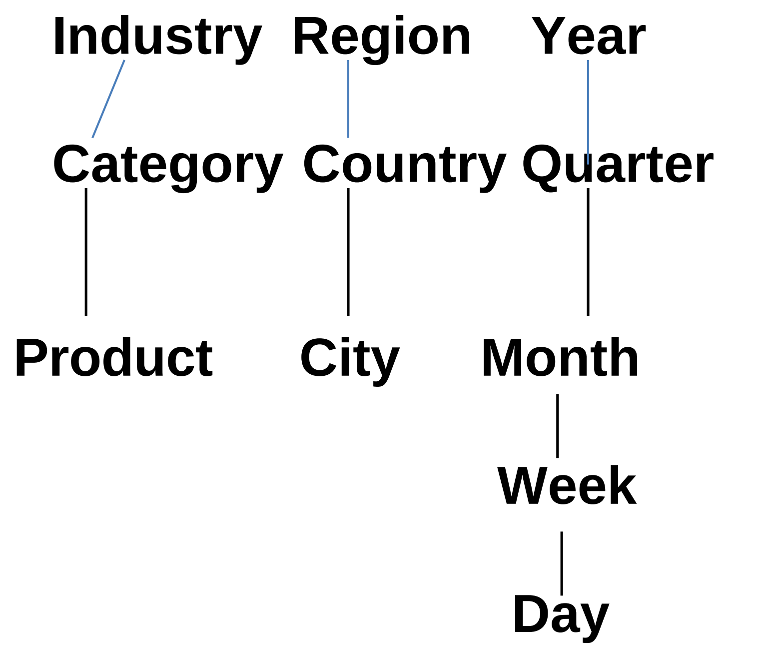
- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg() = sum() / count(), min_N() ...
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

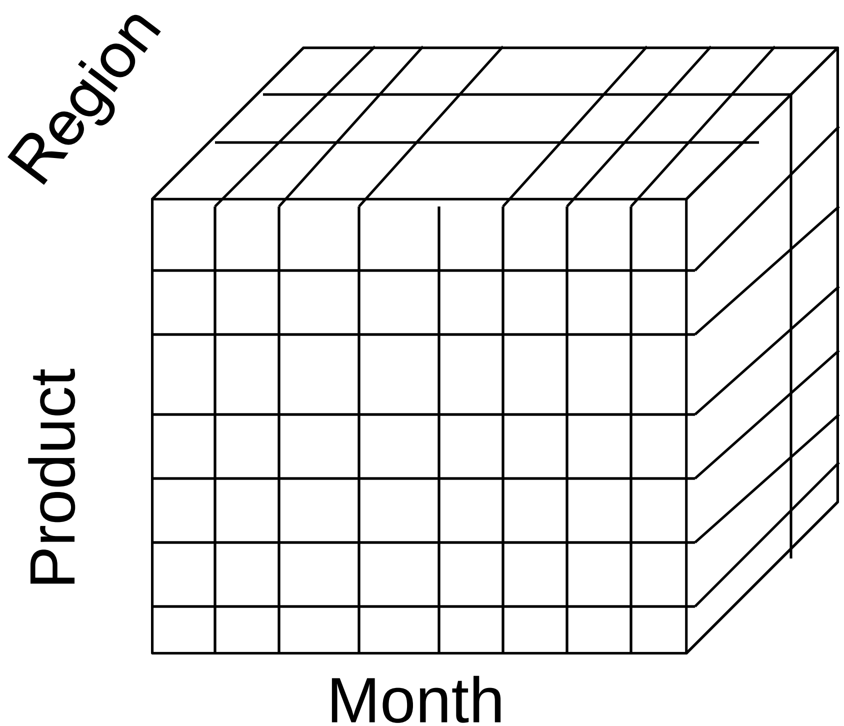
Multidimensional Data

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

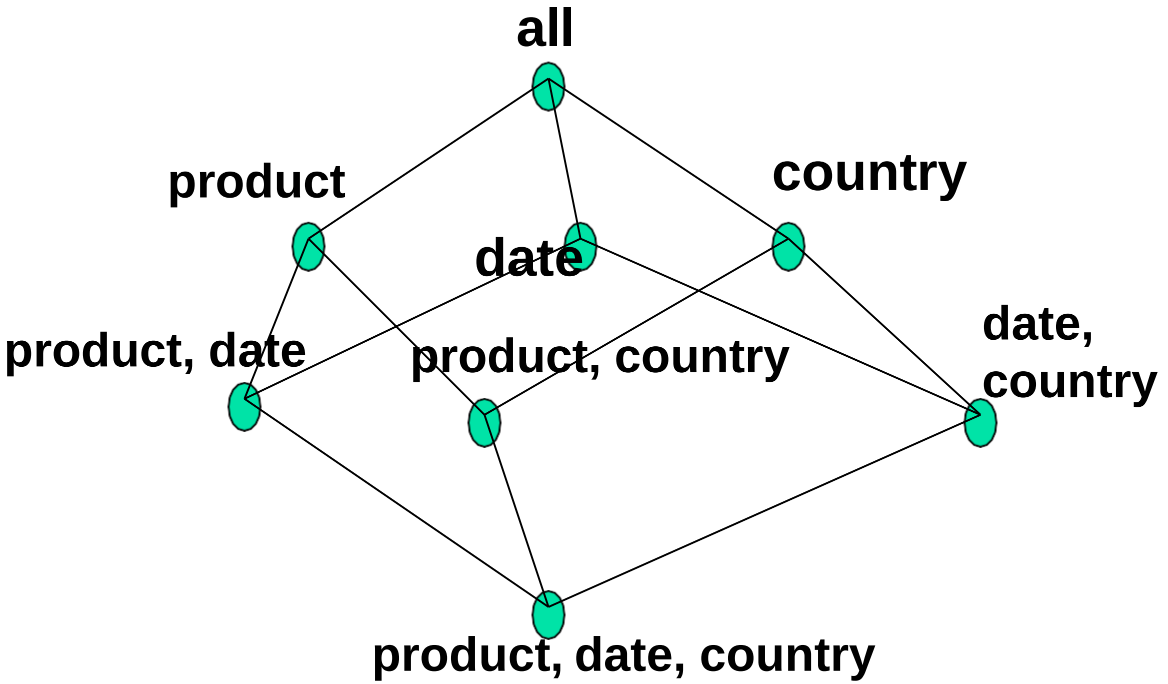
- **Dimensions: *Product, Location, Time***

Hierarchical summarization paths





Cuboids Corresponding to the Cube



0-D (*apex*) cuboid

1-D cuboids

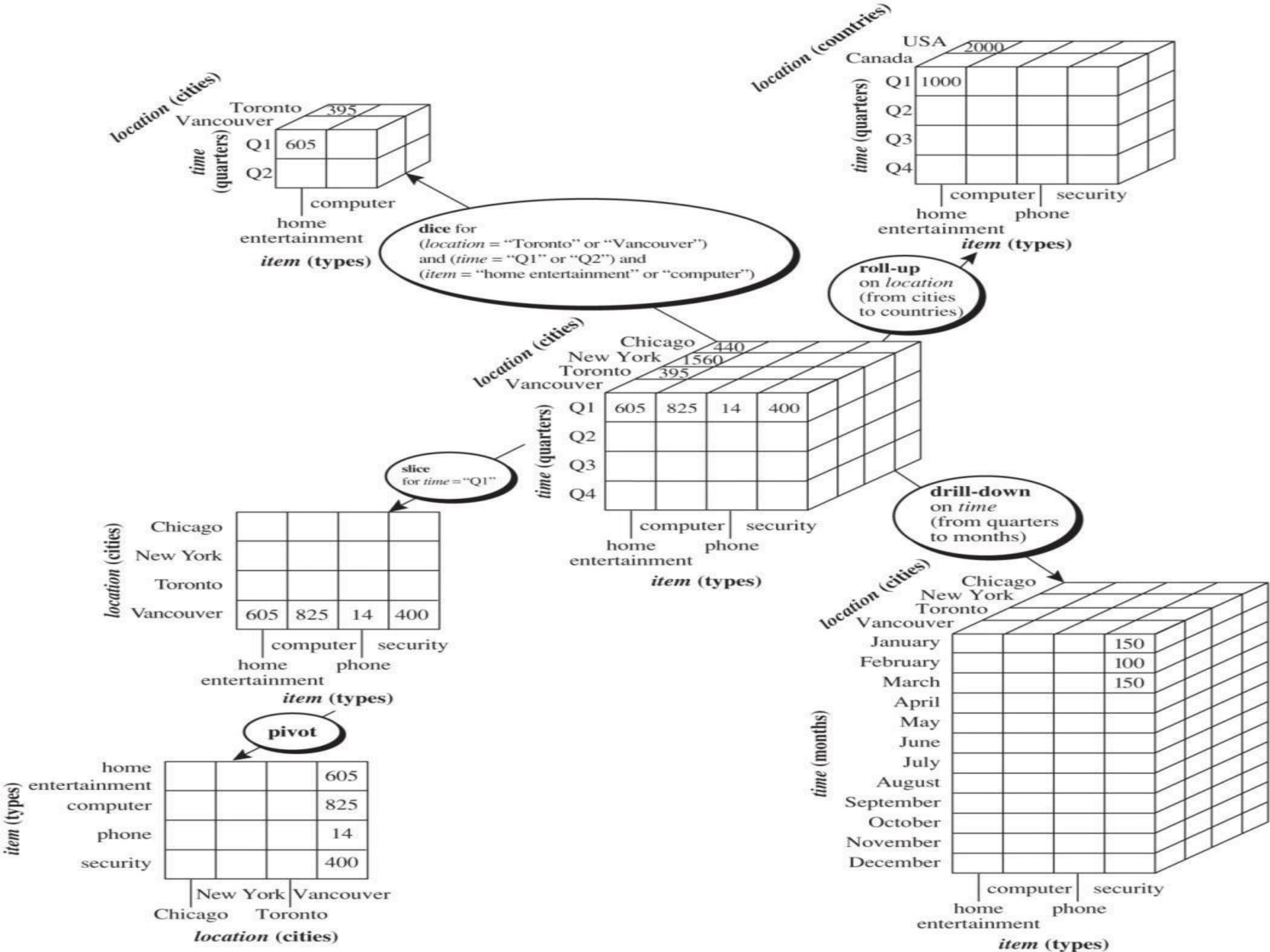
2-D cuboids

3-D (*base*) cuboid

Typical OLAP Operations

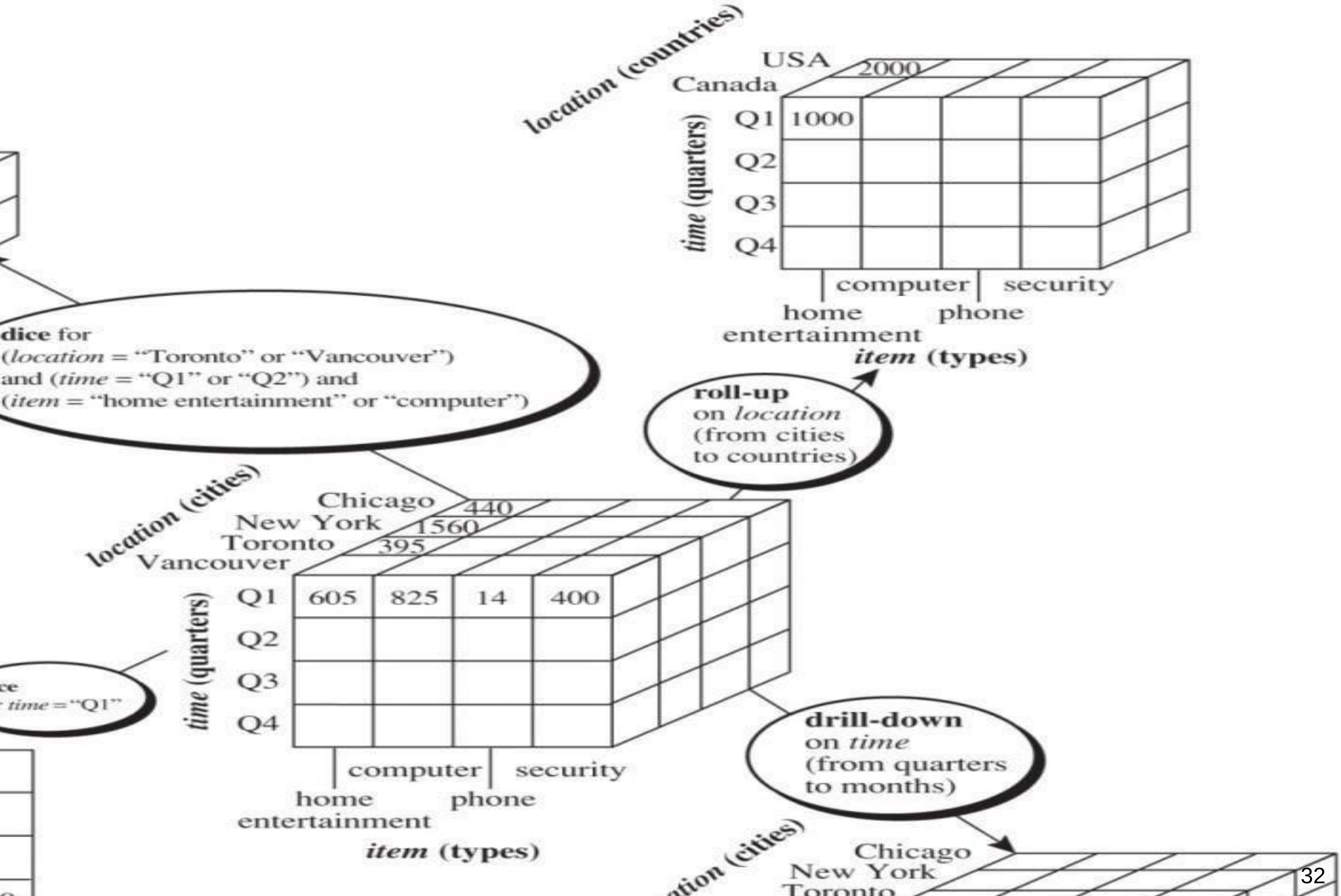
- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- Slice and dice: *project and select*
- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes*
- Other operations
  - *drill across: involving (across) more than one fact table*

    drill through: through the bottom level of the cube

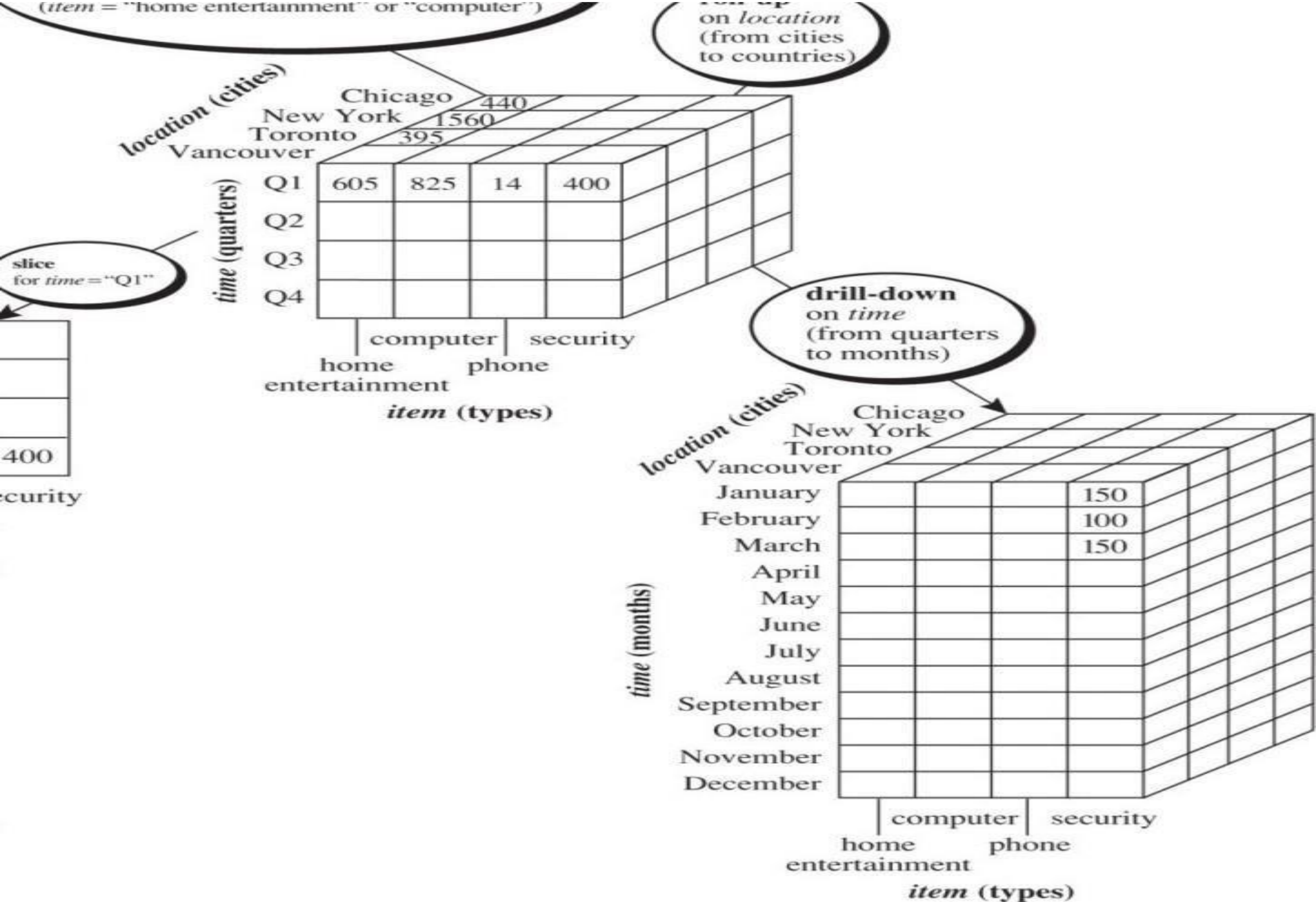


location (countries)
USA
Canada
2000
Q1
1000
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
roll-up on location (from cities to countries)
location (cities)
Toronto
Vancouver
395
Q1
605
Q2
time (quarters)
computer
home entertainment
item (types)
dice for (location = "Toronto" or "Vancouver") and (time = "Q1" or "Q2") and (item = "home entertainment" or "computer")
location (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Q1
605
825
14
400
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
slice for time = "Q1"
Chicago
New York
Toronto
Vancouver
location (cities)
605
825
14
400
computer
security
home entertainment
phone
item (types)
pivot
home entertainment
computer
phone
security
item (types)
605
825
14
400
New York
Vancouver
Chicago
Toronto
location (cities)
drill-down on time (from quarters to months)
location (cities)
Chicago
New York
Toronto
Vancouver
January
February
March
April
May
June
July
August
September
October
November
December
time (months)
150
100
150
computer
security
home entertainment
phone
item (types)

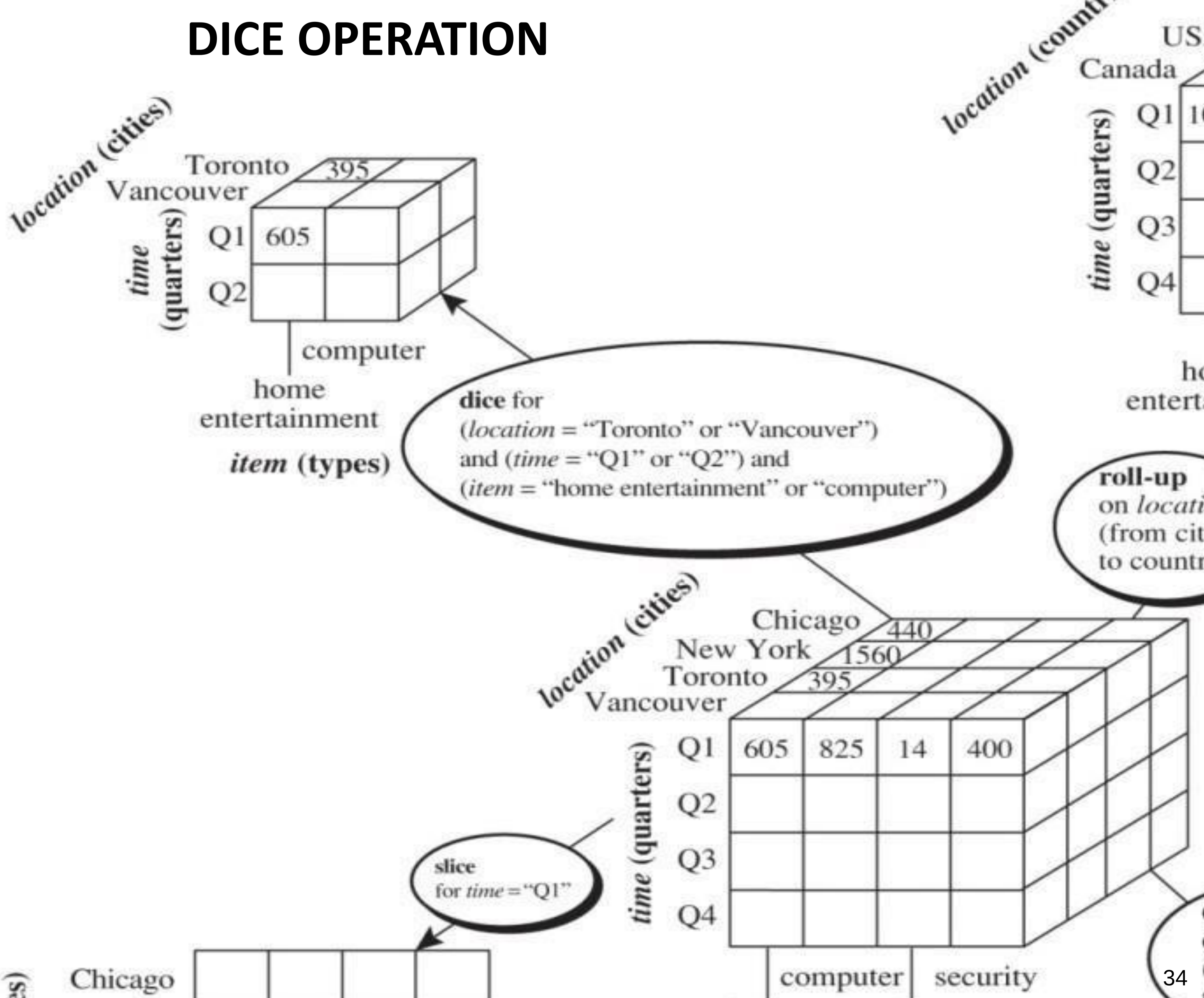
ROLL UP OPERATION



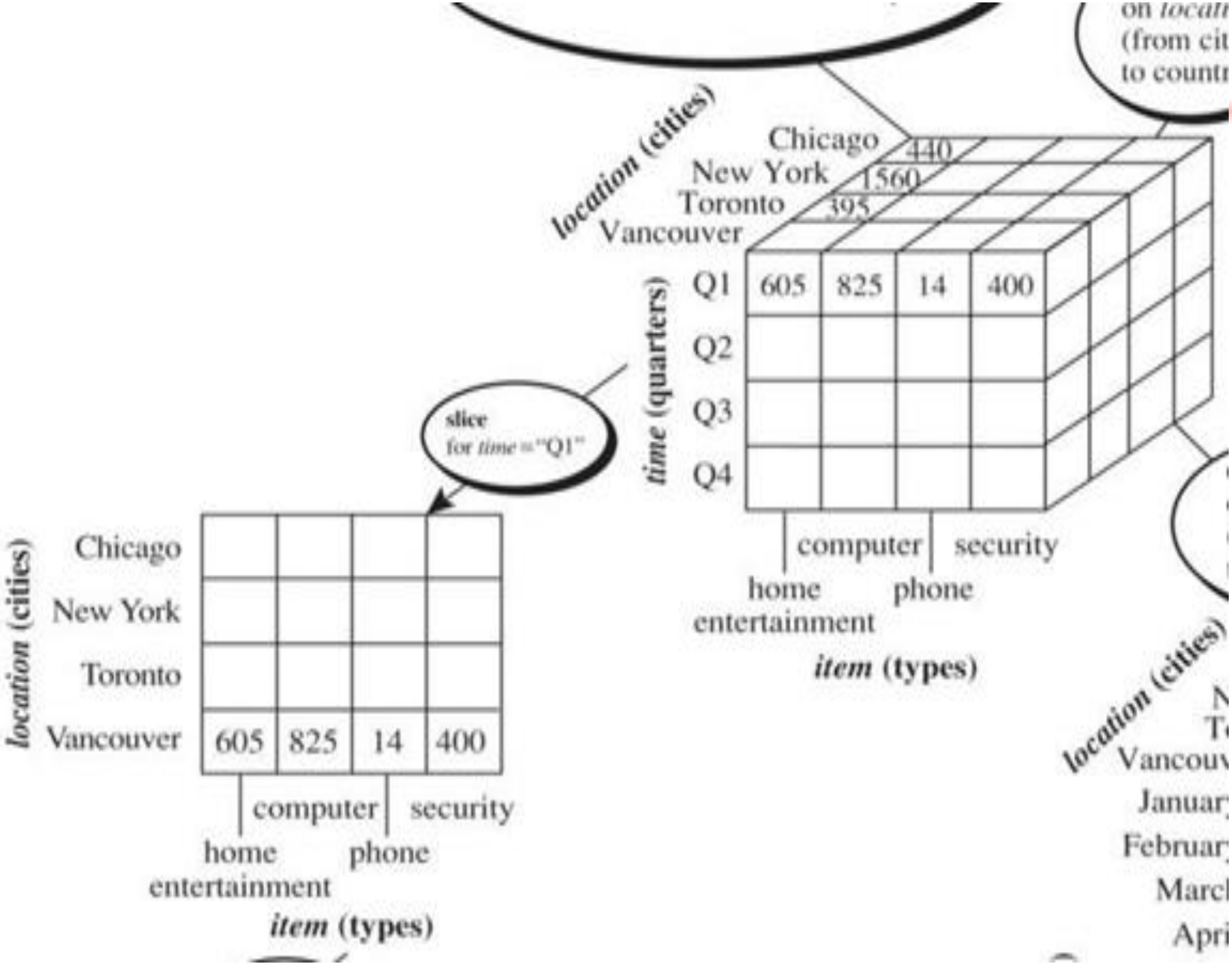
DRILL DOWN OPERATION



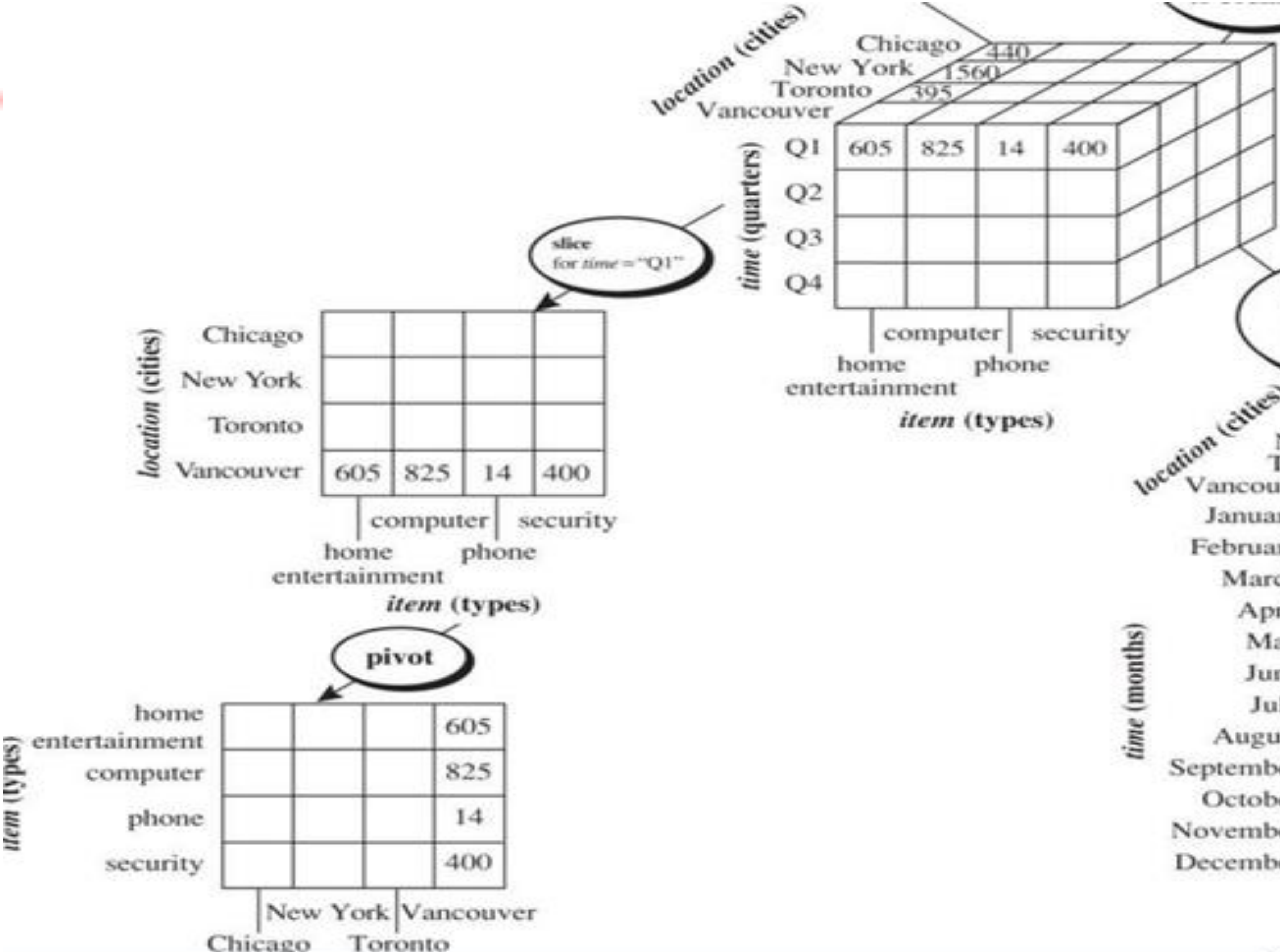
DICE OPERATION



SLICE OPERATION



PIVOT OPERATION



Design and Implementation of Datawarehouse

- Four views regarding the design of a data warehouse
  - Top-down view
    - allows selection of the relevant information necessary for the data warehouse
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user

Data Warehouse Design Process

- **Top-down, bottom-up approaches or a combination** of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- **From software engineering point of view**
  - (1) planning (2) requirements study (3) problem analysis (4) warehouse design (5) data integration and testing (6) deployment
  - Waterfall: structured and systematic analysis at each step before proceeding to the next (better for data warehouse)
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around (better for data marts)

Data Warehouse Design Process

Typical data warehouse design process

- Choose a business process to model, e.g., orders, invoices,etc.
- Choose the *grain* (*atomic level of data*) of the business process
- Choose the dimensions that will apply to each fact table record
- Choose the measure that will populate each fact table record

Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

Datawarehouse Implementation

- Efficient Computation of Data cubes
- Indexing of OLAP data - Bitmap and Join indices
- Processing of OLAP queries
- Warehouse servers for OLAP processing – ROLAP, MOLAP and HOLAP

Efficient Computation of Data cubes

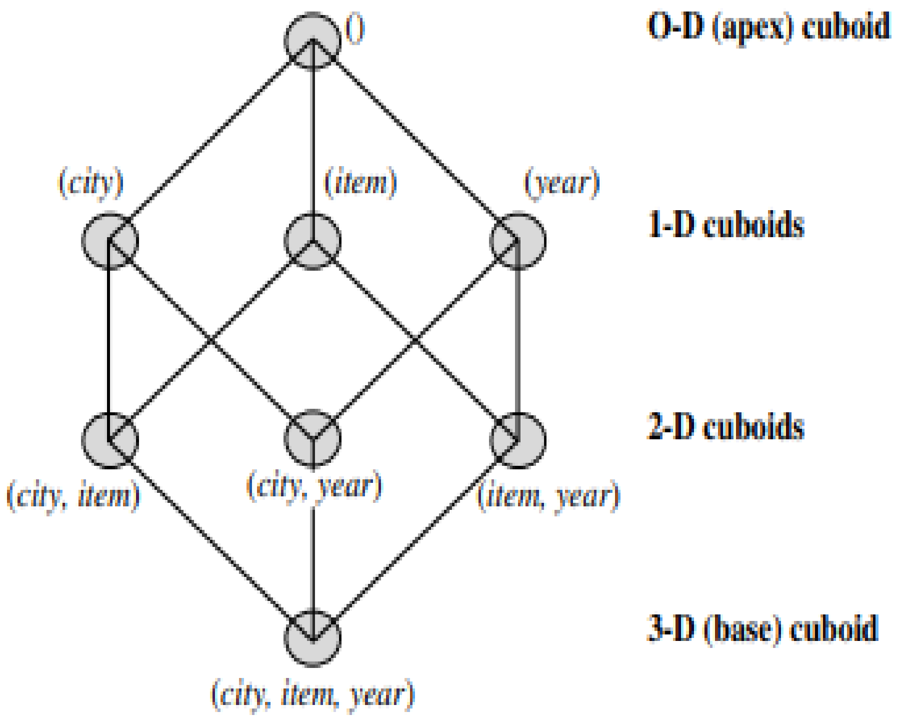
- **Cube computation** extends SQL by including compute cube operator.
- The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation.
- This can require excessive storage space, especially for large numbers of dimensions.
- ” For an n-dimensional data cube, the total number of cuboids that can be generated (including the cuboids generated by climbing up the hierarchies along each dimension) is

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

where Li is the number of levels associated with dimension i. One is added to Li in equation to include the virtual top level, all. (Note that generalizing to all is equivalent to the removal of the dimension.)

Efficient Computation of Data cubes:Example

- A data cube is a lattice of cuboids. Suppose that you want to create a data cube for AllElectronics sales that contains the following: city, item, year, and sales in dollars.
- You want to be able to 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.”



Efficient Computation of Data cubes

- Taking the three attributes, city, item, and year, as the dimensions for the data cube, and sales in dollars as the measure, the total number of cuboids, or groupby's, that can be computed for this data cube is $2^3 = 8$.
- The possible group-by's are the following: {(city, item, year), (city, item), (city, year), (item, year), (city), (item), (year), ()}, where () means that the group-by is empty

- Similar to the SQL syntax, the data cube could be defined as:
  - **define cube sales cube [city, item, year]: sum(sales in dollars)**
- For a cube with n dimensions, there are a total of 2^n cuboids, including the base cuboid.
- A statement such as : "**compute cube sales_cube**" would explicitly instruct the system to compute the sales aggregate cuboids for all eight subsets of the set {city, item, year}, including the empty subset.

Efficient Computation of Data cubes

Selected Computation of Cuboids

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

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

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

Indexing of OLAP data - Bitmap and Join indices

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

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.
- Suppose that the cube is stored as a relation table with 100,000 rows.
- Because the domain of item consists of four values, the bitmap index table requires four bit vectors (or lists), each with 100,000 bits.

Indexing of OLAP data - Bitmap indexing

- The below structure shows a base (data) table containing the dimensions item and city, and its mapping to bitmap index tables for each of the dimensions.

Base table

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

item bitmap index table

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

city bitmap index table

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

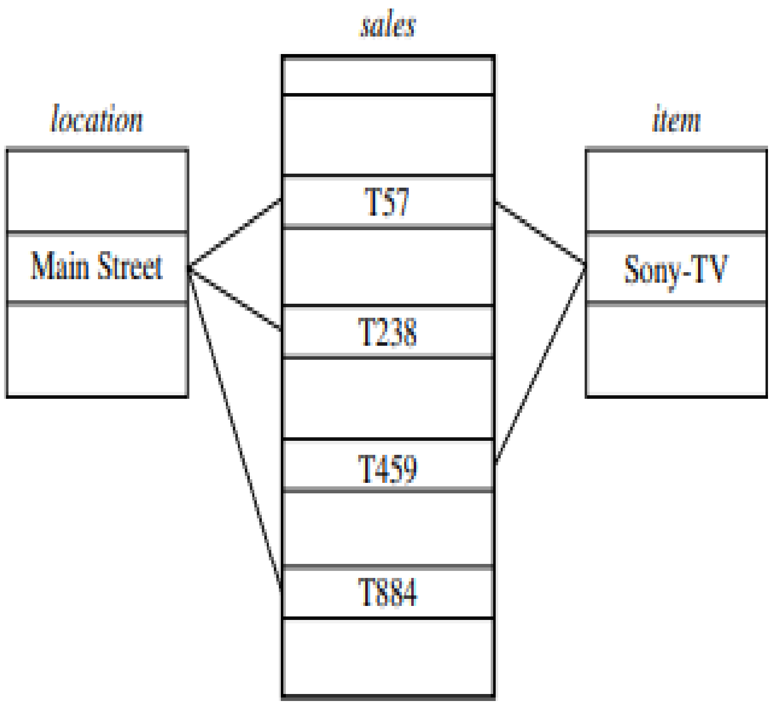
Note: H for "home entertainment," C for "computer," P for "phone," S for "security," V for "Vancouver," T for "Toronto."

Indexing OLAP data using bitmap indices.

Indexing of OLAP data - Join indexing

Join indexing

- A star schema for AllElectronics of the form "sales star [time, item, branch, location]: dollars sold = sum (sales in dollars)."



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
. . .	. . .	. . .

Fig 1: Linkages between a sales fact table and location and item dimension tables.

Fig 2: Join index tables based on the linkages between the sales fact table and the location and item dimension tables

Processing of OLAP queries

- The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes.
- Given materialized views, query processing should proceed as follows:
  1. **Determine which operations should be performed on the available cuboids:** This involves transforming any selection, projection, roll-up (group-by), and drill-down operations specified in the query into 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.

Processing of OLAP queries - Example

- Suppose that we define a data cube for AllElectronics of the form “sales cube [time, item, location]: sum(sales in dollars).”
- The dimension hierarchies used are “day < month < quarter < year” for time; “item name < brand < type” for item; and “street < city < province or state < country” for location.
- Suppose that the query to be processed is on {brand, province or state}, with the selection constant “year = 2010.”
- Also, suppose that there are four materialized cuboids available, as follows:
  - cuboid 1: {year, item name, city}
  - cuboid 2: {year, brand, country}
  - cuboid 3: {year, brand, province or state}
  - cuboid 4: {item name, province or state}, where year = 2010

“Which of these four cuboids should be selected to process the query?”

Processing of OLAP queries

- Suppose that the query to be processed is on {brand, province or state}, with the selection constant “year = 2010.”
- Also, suppose that there are four materialized cuboids available, as follows:
  - cuboid 1: {year, item name, city}
  - cuboid 2: {year, brand, country}
  - cuboid 3: {year, brand, province or state}
  - cuboid 4: {item name, province or state}, where year = 2010

“Which of these four cuboids should be selected to process the query?”

- ”Finer-granularity data cannot be generated from coarser-granularity data.
- Therefore, cuboid 2 cannot be used because country is a more general concept than province or state.
- Cuboids 1, 3, and 4 can be used to process the query.

Warehouse servers for OLAP processing – ROLAP, MOLAP and HOLAP

- OLAP servers present business users with multidimensional data from data warehouses or data marts, without concerns regarding how or where the data are stored.
- However, the physical architecture and implementation of OLAP servers must consider data storage issues. Implementations of a warehouse server for OLAP processing include the following:
  - **Relational OLAP (ROLAP) servers**
  - **Multidimensional OLAP (MOLAP) servers**
  - **Hybrid OLAP (HOLAP) servers**

Relational OLAP (ROLAP) servers

- These are the intermediate servers that stand in between a relational back-end server and client front-end tools.
- They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces.

Multidimensional OLAP (MOLAP) servers

- These servers support multidimensional data views through array-based multidimensional storage engines.
- They map multidimensional views directly to data cube array structures.

Hybrid OLAP (HOLAP) servers

- The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP.
- For example, a HOLAP server may allow large volumes of detailed data to be stored in a relational database, while aggregations are kept in a separate MOLAP store.
- The Microsoft SQL Server 2000 supports a hybrid OLAP server

Specialized SQL servers:

- To meet the growing demand of OLAP processing in relational databases, some database system vendors implement specialized SQL servers that provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.