Data Mining:

Concepts and Techniques

(3rd ed.)

— Chapter 4 —

Jianjun Cheng

School of Information Science & Engineering

Lanzhou University

Chapter 4: Data Warehousing and On-line Analytical Processing

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**

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration: A query driven approach
  - Build wrappers/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, and the results are integrated into a global answer set
  - Complex information filtering, compete for resources
- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

Data Warehouse vs. Operational DBMS

- OLTP (on-line transaction processing)
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- OLAP (on-line analytical processing)
  - Major task of data warehouse system
  - Data analysis and decision making
- Distinct features (OLTP vs. OLAP):
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - Database design: ER + application vs. star + subject
  - View: current, local vs. evolutionary, integrated
  - Access patterns: update vs. read-only but complex queries

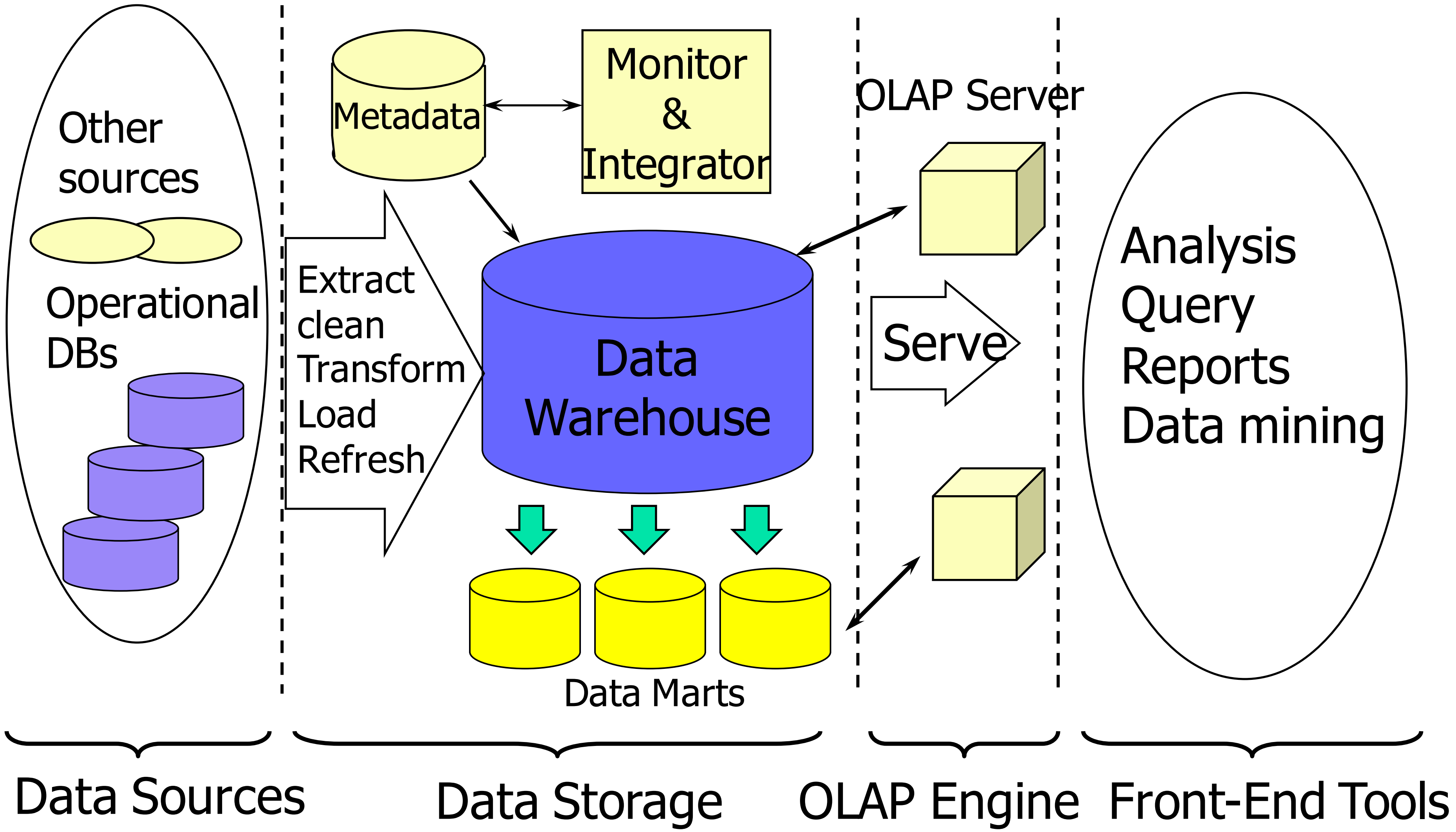
OLTP vs. OLAP

	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

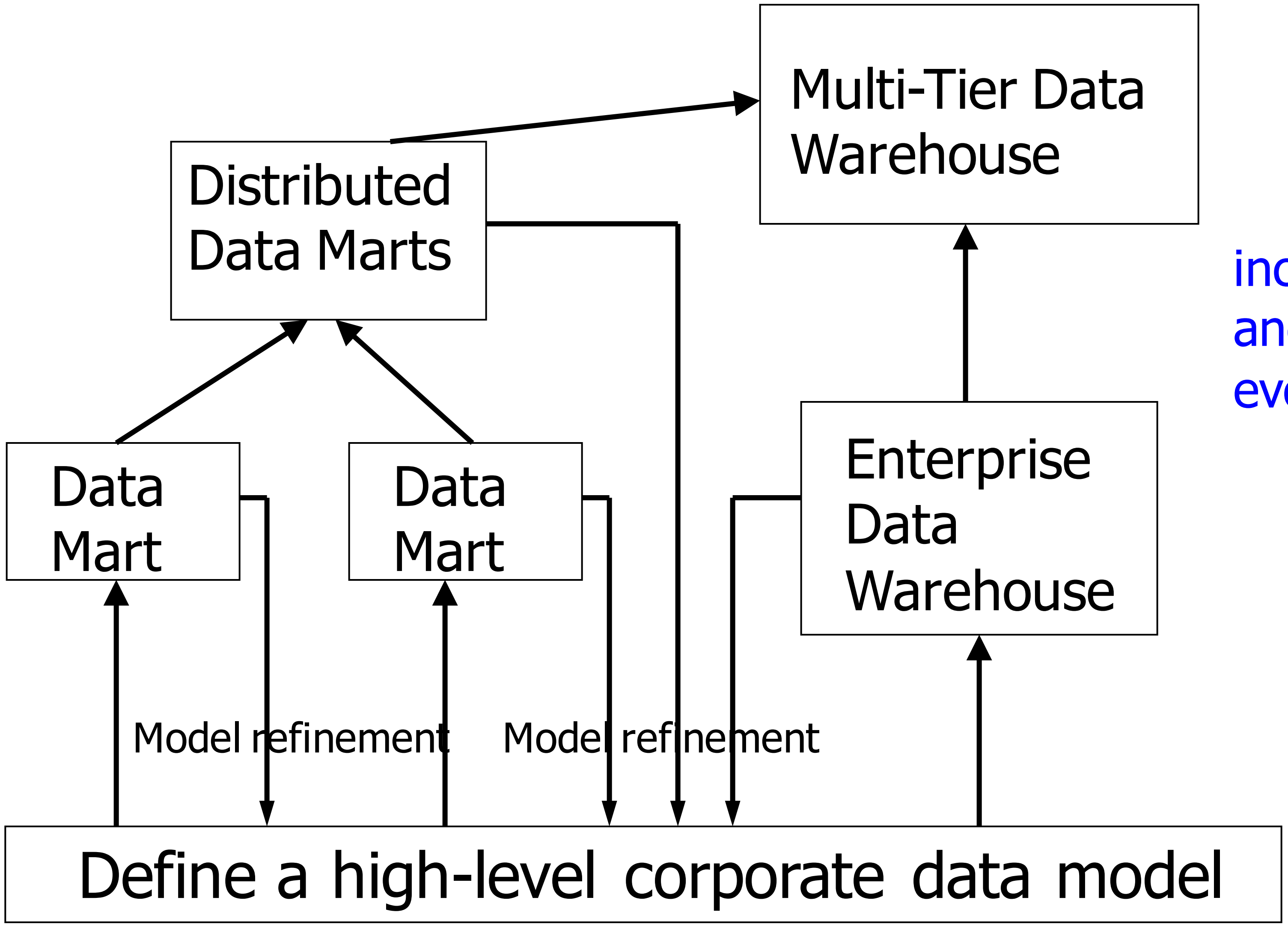
Data Warehouse: A Multi-Tiered Architecture



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 vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

Data Warehouse: A recommended approach



Multi-Tier Data Warehouse
Distributed Data Marts
Data Mart
Data Mart
Enterprise Data Warehouse
Model refinement
Model refinement
Define a high-level corporate data model

incremental and evolutionary

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 indicies and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

Metadata Repository

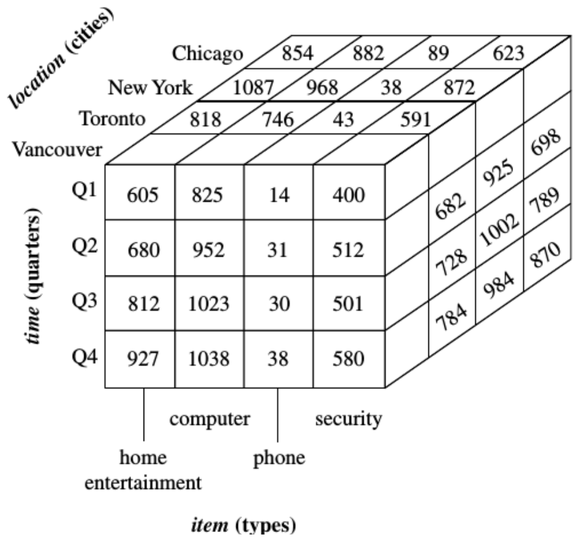
- **Meta data** is the data defining warehouse objects. It stores:
  - Description of the structure of the data warehouse
    - schema, view, dimensions, hierarchies, derived data defnition, data mart locations and contents
  - Operational meta-data
    - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
  - The algorithms used for summarization
    - measure, dimension definition algorithm, data on granularity, partitions, subject areas, aggregation, summary, predefined queries and reports
  - The mapping from operational environment to the data warehouse
    - src. db. and contents, gw. des., data parti., data extract., cleaning, transformation rules and defaults, data refresh and purging rules, security
  - Data related to system performance
    - warehouse schema, view and derived data definitions
  - Business data
    - business terms and definitions, ownership of data, charging policies

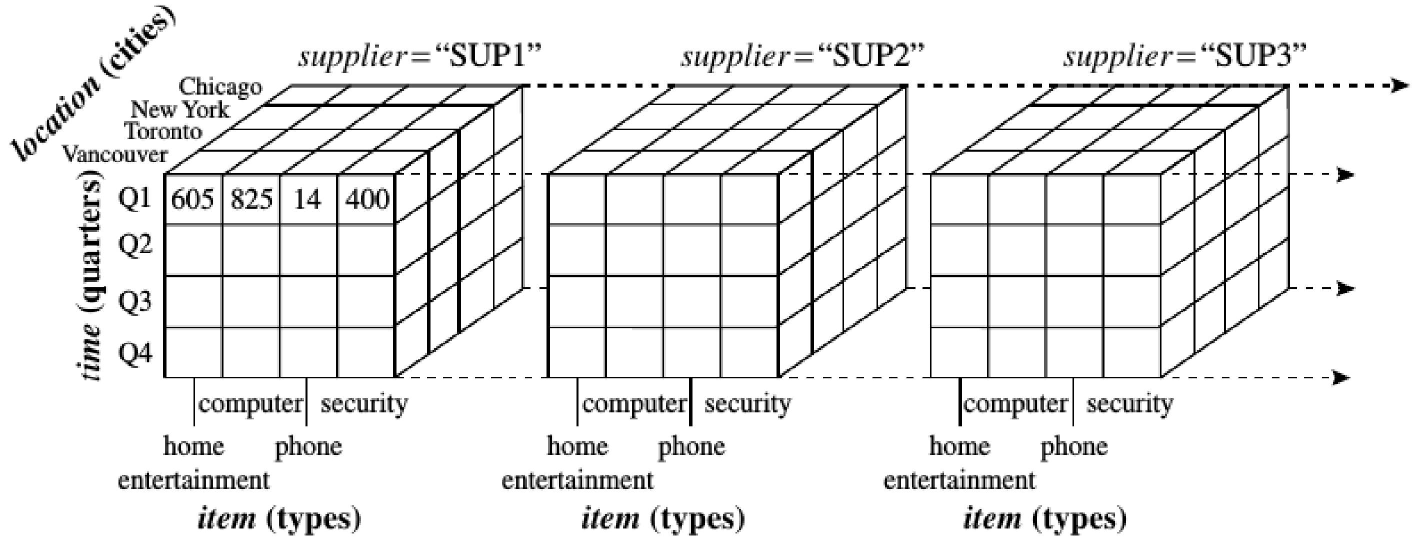
Chapter 4: Data Warehousing and On-line Analytical Processing

From Tables and Spreadsheets to Data Cubes

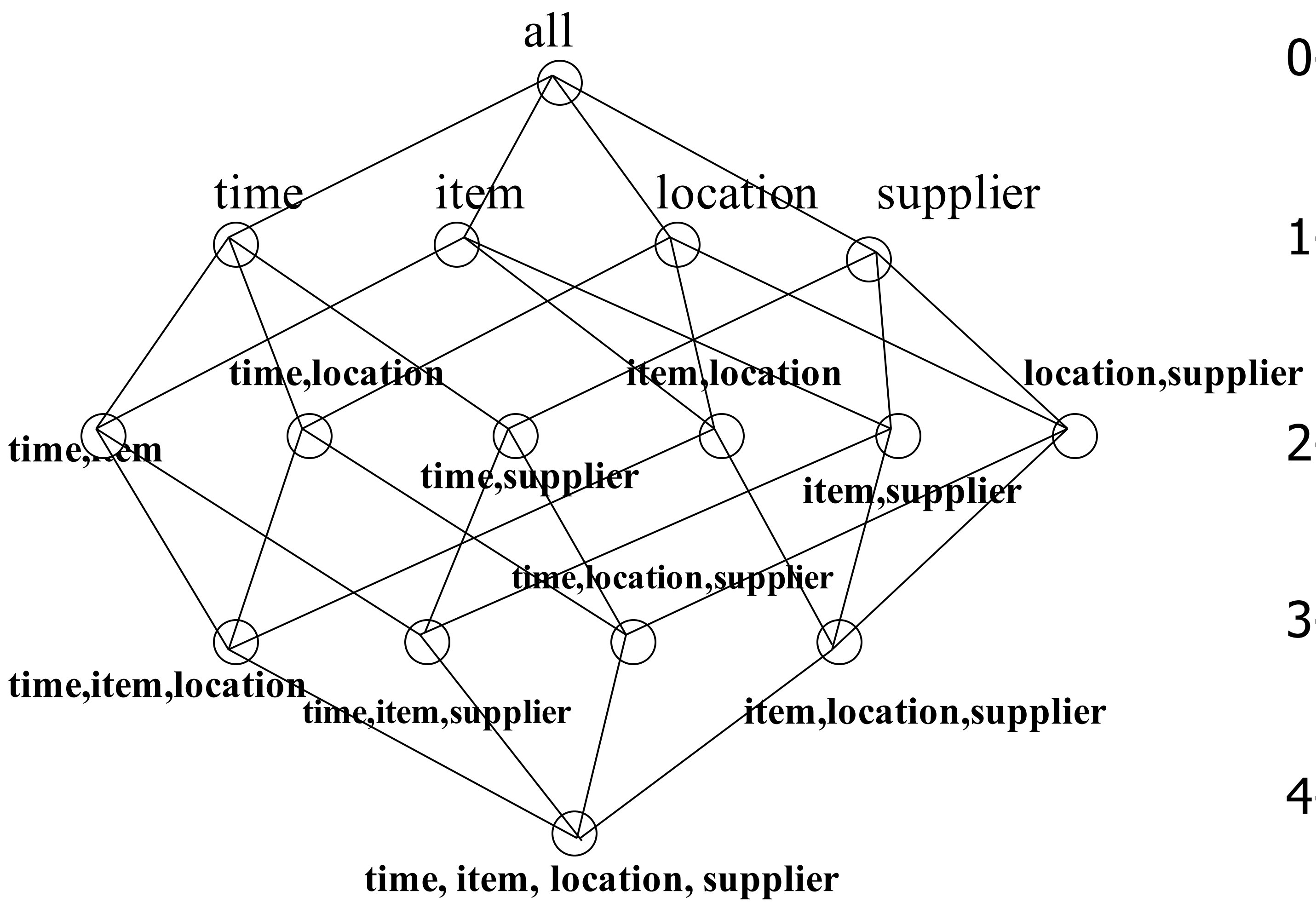
- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - **Dimension tables**, such as item(item_name, brand, type), or time(day, week, month, quarter, year)
  - **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: an example





Cube: A Lattice of Cuboids



0-D (*apex*) cuboid

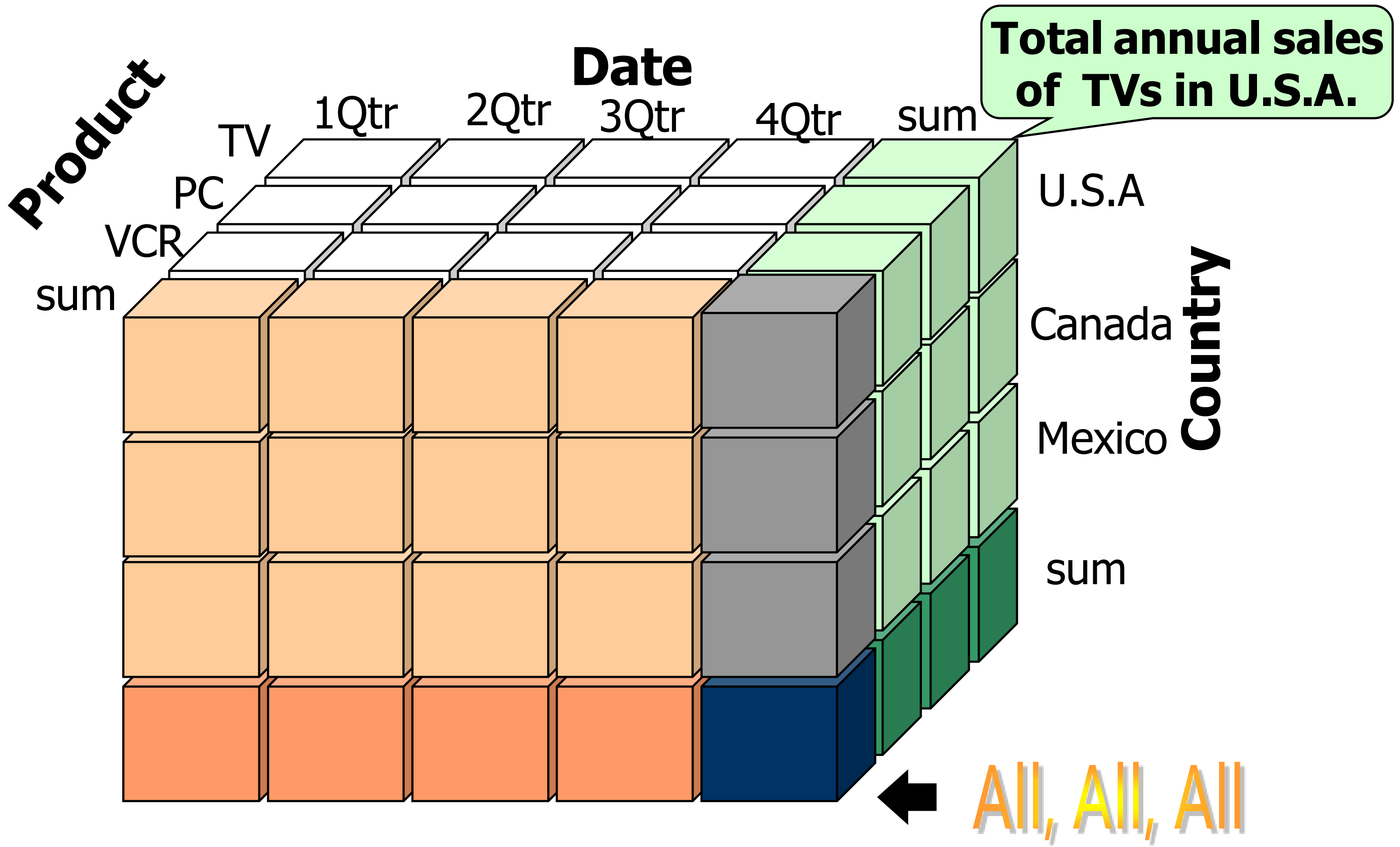
1-D cuboids

2-D cuboids

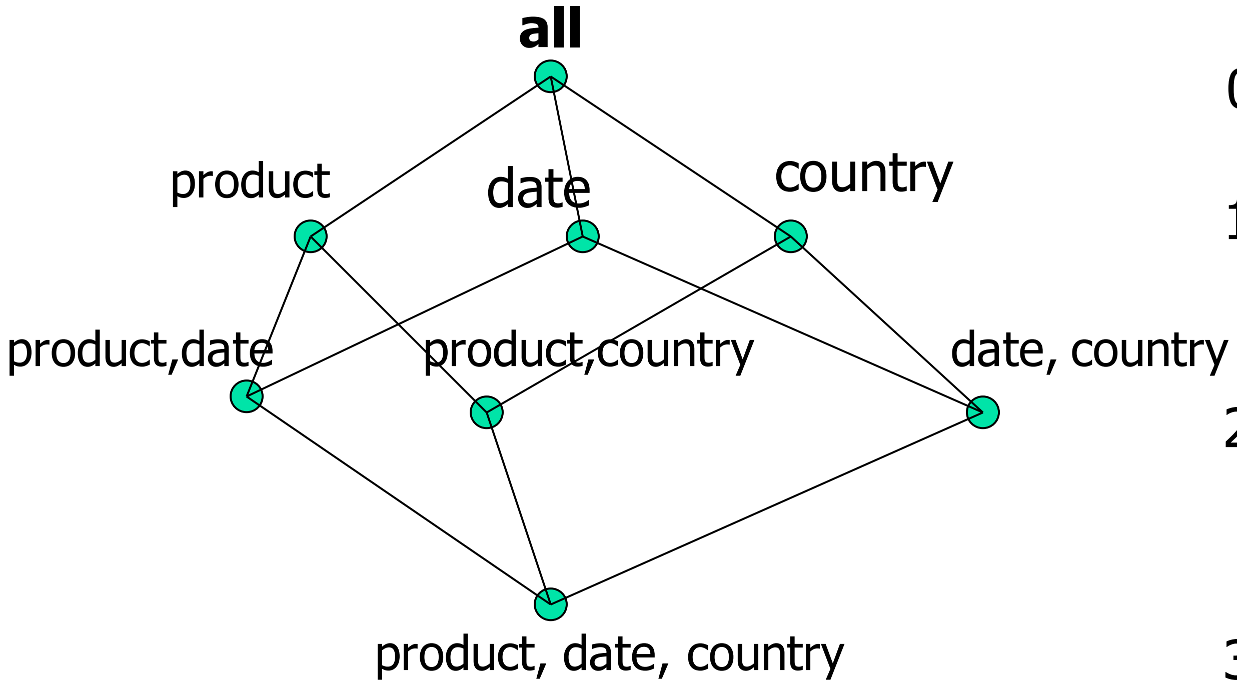
3-D cuboids

4-D (*base*) cuboid

A Sample Data Cube



Cuboids Corresponding to the Cube



0-D (*apex*) cuboid

1-D cuboids

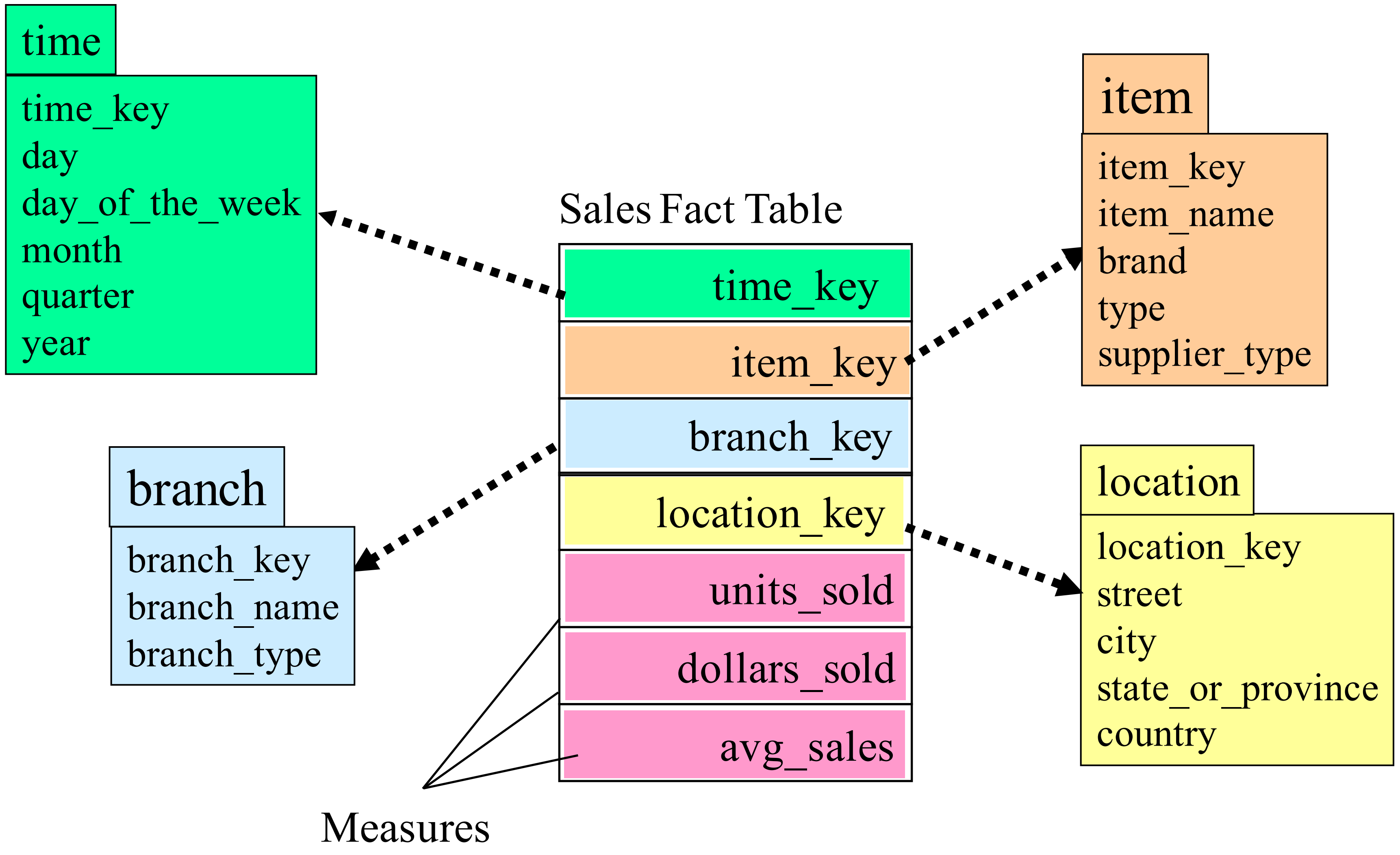
2-D cuboids

3-D (*base*) cuboid

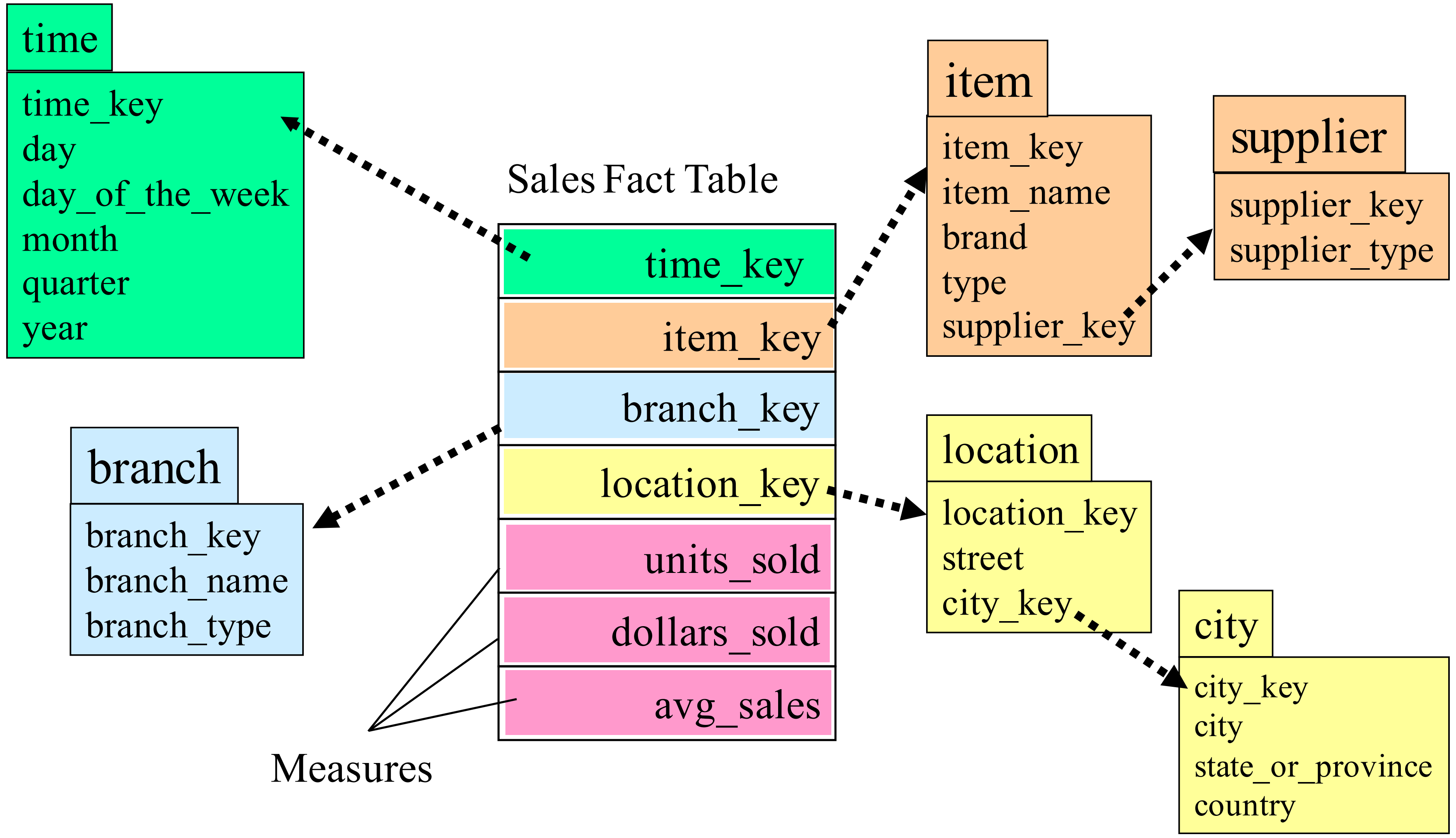
Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & 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

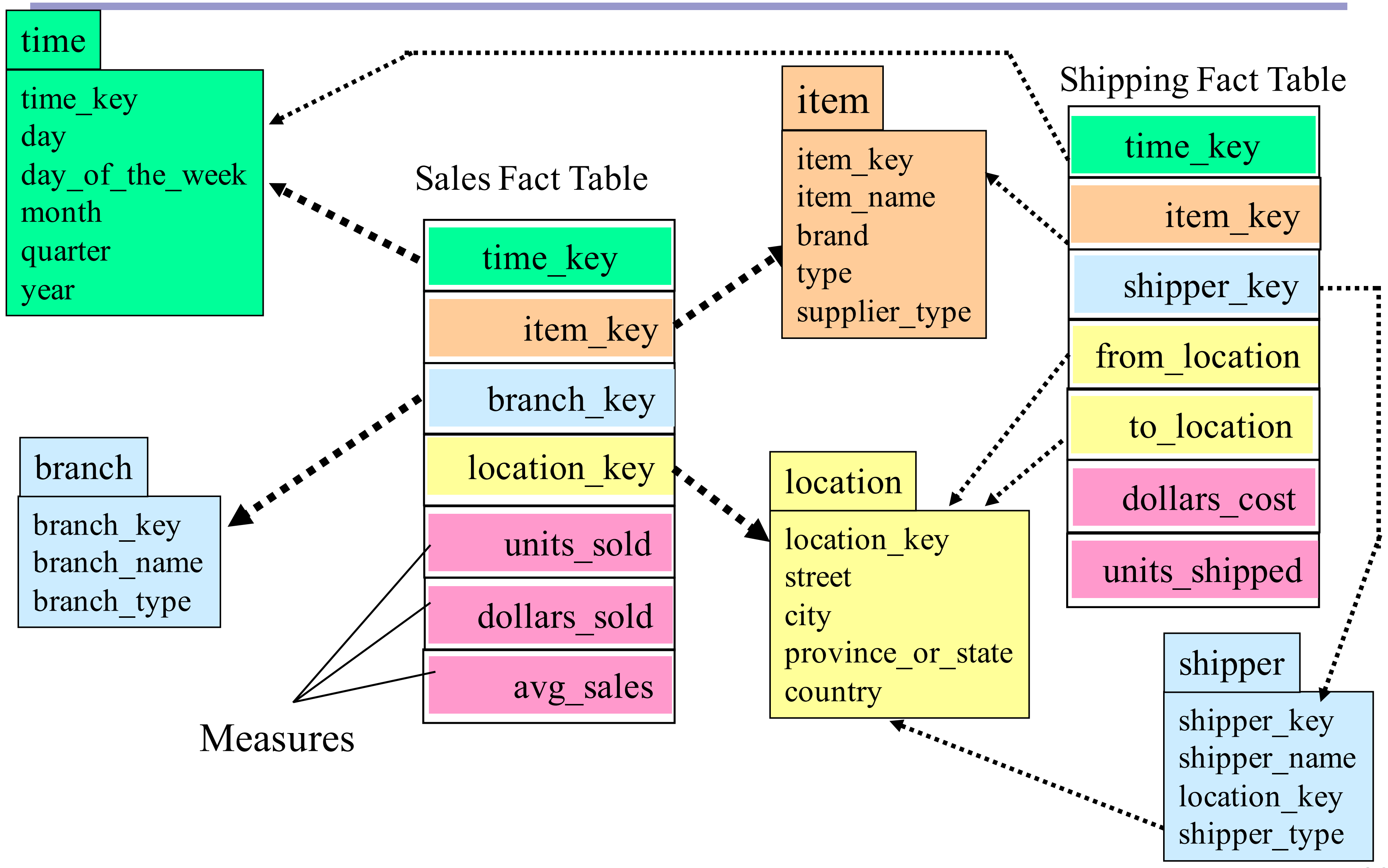
Example of **Star Schema**



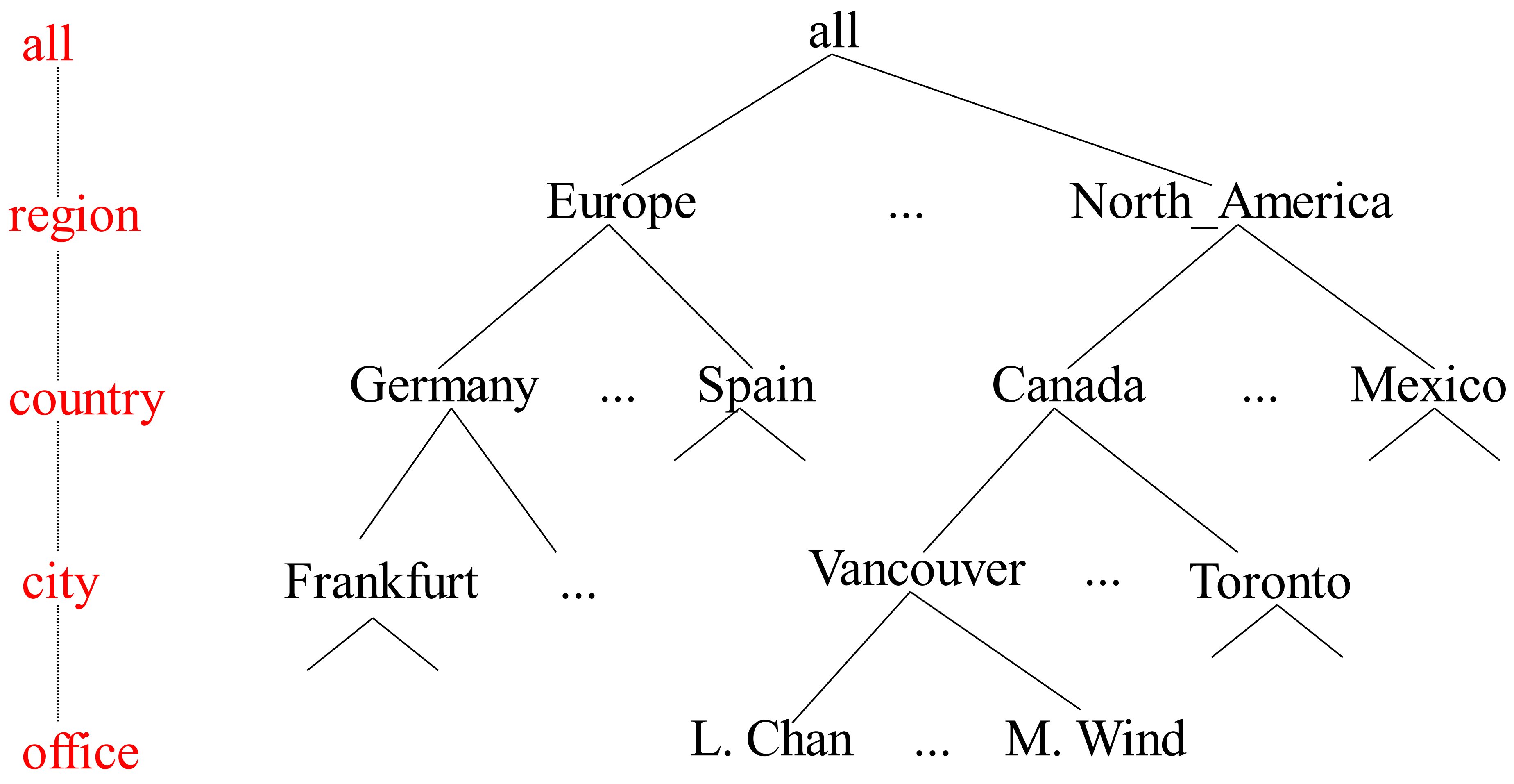
Example of **Snowflake Schema**



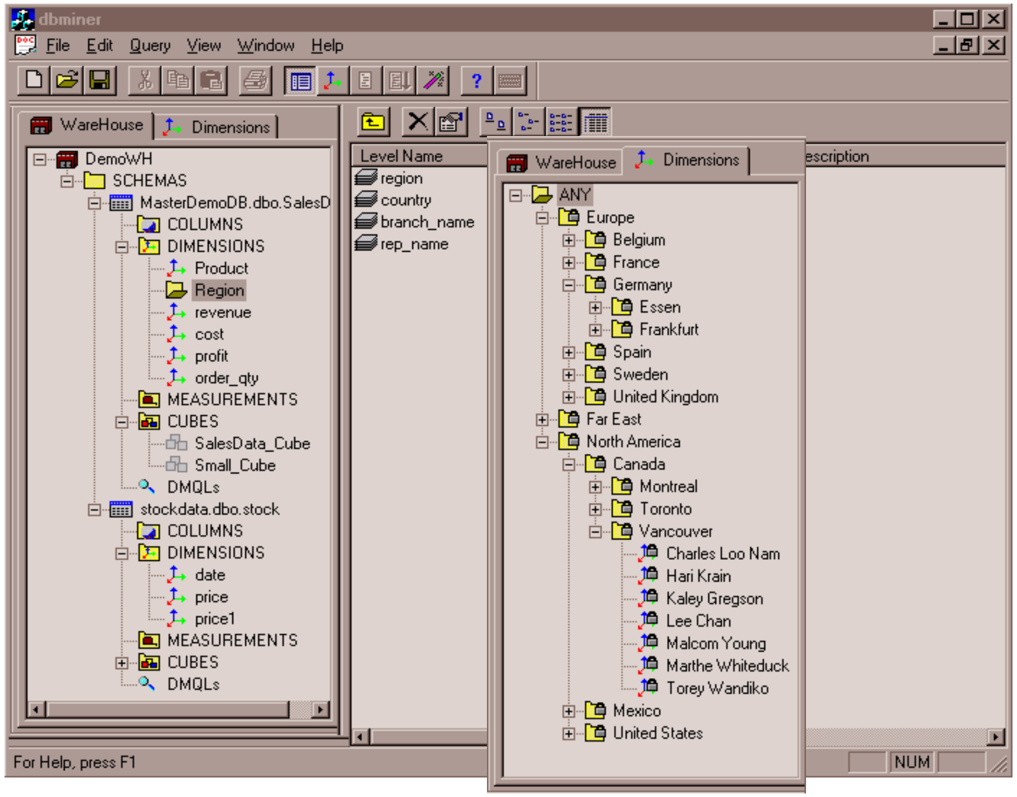
Example of **Fact Constellation**



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



View of Warehouses and Hierarchies



Specification of hierarchies

- Schema hierarchy

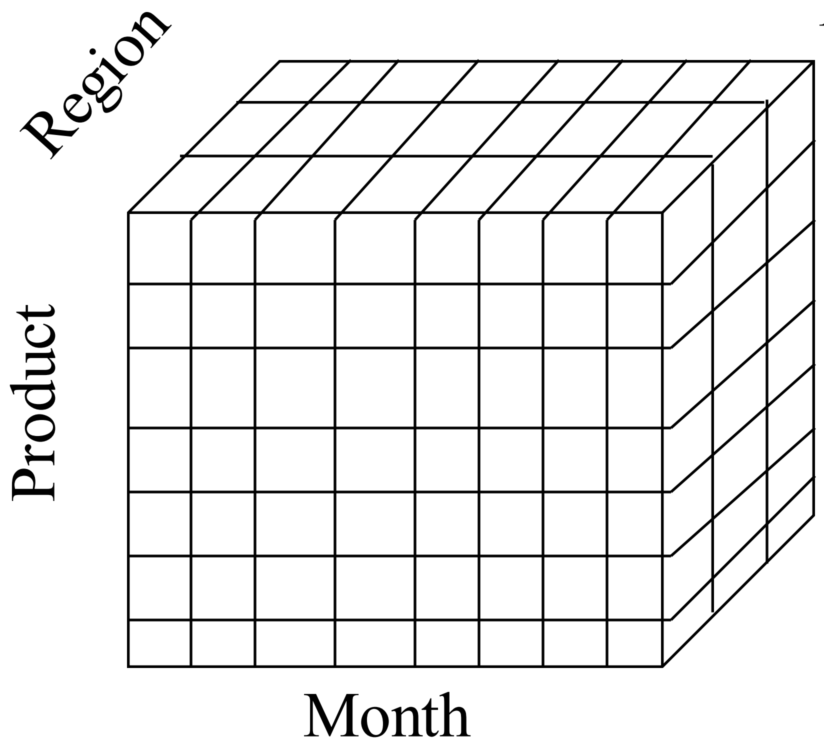
  day < {month < quarter; week} < year

- Set_grouping hierarchy

  {1..10} < inexpensive

Multidimensional Data

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



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

Hierarchical summarization paths

$$\begin{tikzcd}
\textbf{Industry} \arrow[d, dash] & \textbf{Region} \arrow[d, dash] & \textbf{Year} \arrow[d, dash] \arrow[ddr, dash] & \\
\textbf{Category} \arrow[d, dash] & \textbf{Country} \arrow[d, dash] & \textbf{Quarter} \arrow[d, dash] & \\
\textbf{Product} & \textbf{City} \arrow[d, dash] & \textbf{Month} \arrow[d, dash] & \textbf{Week} \arrow[dl, dash] \\
 & \textbf{Office} & \textbf{Day} &
\end{tikzcd}$$

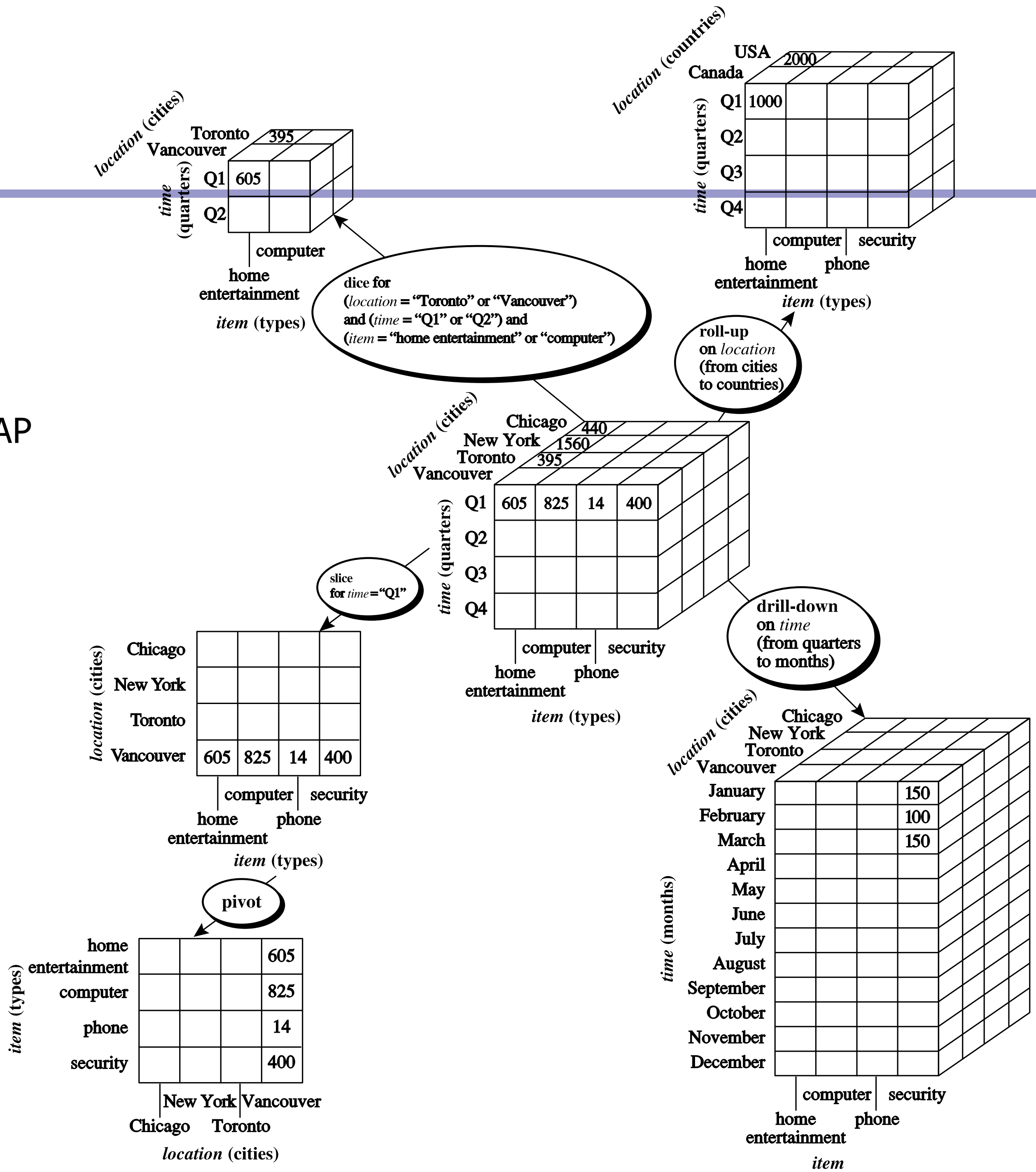
Data Cube Measures: Three Categories

- 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(), min_N(), standard_deviation()
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
    - E.g., median(), mode(), rank()

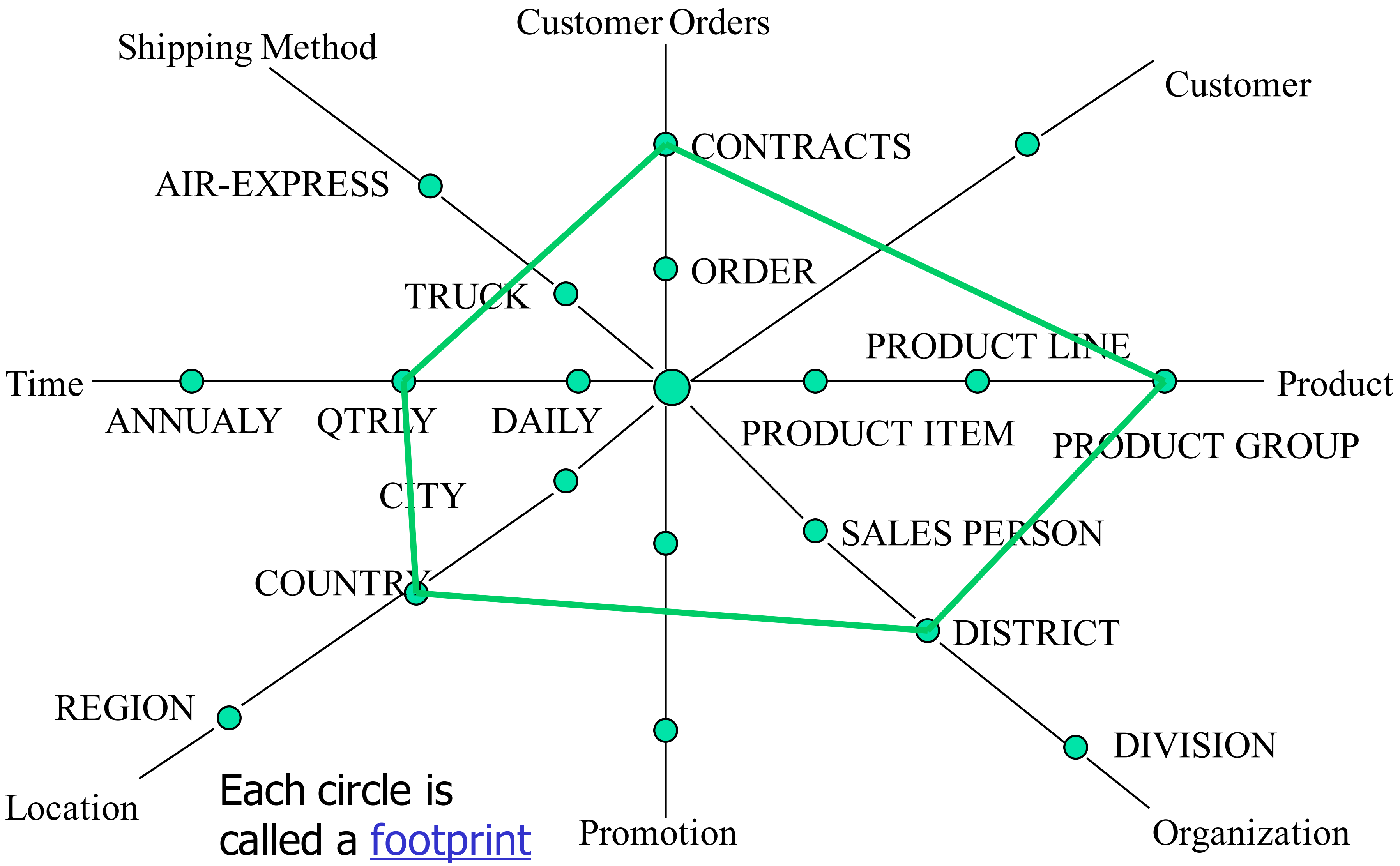
Typical OLAP Operations

- 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 to its back-end relational tables (using SQL)

Fig. 3.10 Typical OLAP Operations



A Star-Net Query Model



Each circle is called a footprint

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
- Data Generalization by Attribute-Oriented Induction
- Summary

Design of Data Warehouse: A Business Analysis Framework

- Business analysts gain much from having a data warehouse
  - competitive advantage
    - presenting relevant information from which to measure performance and make critical adjustment to help win over competitors
  - business productivity
    - abel to quickly and efficiently gather information that accurately describe the orgniazation
  - CRM
    - provides a consistent view of customer and items across all lines of business, all departments and all markets
  - 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 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

Design of Data Warehouse: A Business Analysis Framework

- Three skills needed
  - Business skills, involves understanding
    - how system store and manage their data
    - how to build extractors that transfer data from operational systems to data warehouse
    - how to build warehouse refresh software that keeps data warehouse up-to-date with the operational system's data

  Using a data warehouse involves understanding the significance of the data it contains, as well as understanding and translating the business requirements into queries that can be satisfied by the data warehouse

Design of Data Warehouse: A Business Analysis Framework

- Three skills needed
  - technology skills, data analysts are required to understand
    - how to make assessments from quantitative information and derive facts based on conclusions from historic information
    - include abilities to discover patters and trends, to extrapolate trends based on history and look for anomalies or paradigm shifts, to present coherent managerial recommendations based on such analysis
  - program management skills, involve
    - the need to interface with many technologies, vendors, and end-user in order to deliver results in a timely and cost-effective manner

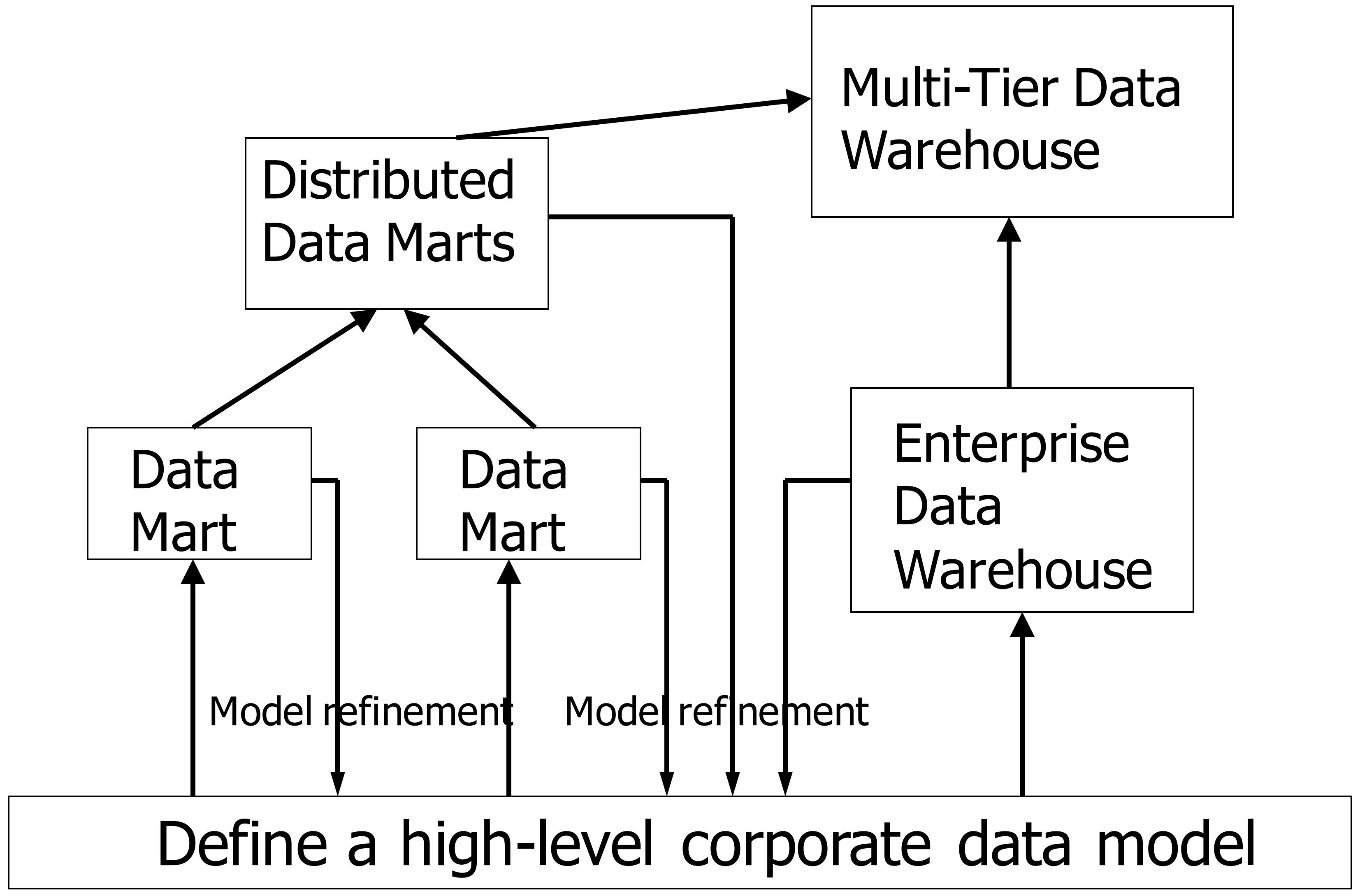
Data Warehouse Design Process

- **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**
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around

Data Warehouse Design Process

- **Typical data warehouse design process**
  - Choose a business process to model, e.g., orders, invoices, etc.
    - orgnizational, involves multiple complex object collections ---- data warehouse model
    - departmental, focuses on the analysis of one kind of business process ---- data mart
  - Choose the grain (atomic level of data) of the business process
    - individual transactions, individual daily snapshots, ...
  - Choose the dimensions that will apply to each fact table record
    - time, item, customer, supplier, warehouse, transaction type, status,...
  - Choose the measure that will populate each fact table record
    - numeric additive quantities: dollars_sold, unit_sold
- dw construction is a difficult and long-term task, its implementation scope should be clearly defined
- The goals of an initial dw implementation should be specific, achievable, and measurable, involve determining
  - time and budget allocations
  - the subset of the organization that is to be modeled
  - the number of data sources selected
  - the number and types of departments to be served

Data Warehouse Development: A Recommended Approach


Multi-Tier Data Warehouse
Distributed Data Marts
Data Mart
Data Mart
Enterprise Data Warehouse
Model refinement
Model refinement
Define a high-level corporate data model

Data Warehouse Usage

- data warehouse evolves along with its application
  - generate reports and answer predefined queries
  - anlyze summarized and detailed data, the results are represented in the form of reports and charts
  - is used for strategic purposes, performing multidimensional analysis and sophisticated slice-and-dice operations
  - is employed for knowledge discovery and strategic decision making using data mining tools

Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
    - construct low-cost web-based accessing tools that are then integrated with web browsers
  - 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

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

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - Mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, and tasks

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
- Data Generalization by Attribute-Oriented Induction
- Summary

Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels?

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

- Materialization of data cube
  - Materialize every (cuboid) (**full materialization**), none (**no materialization**), or some (**partial materialization**)
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.

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

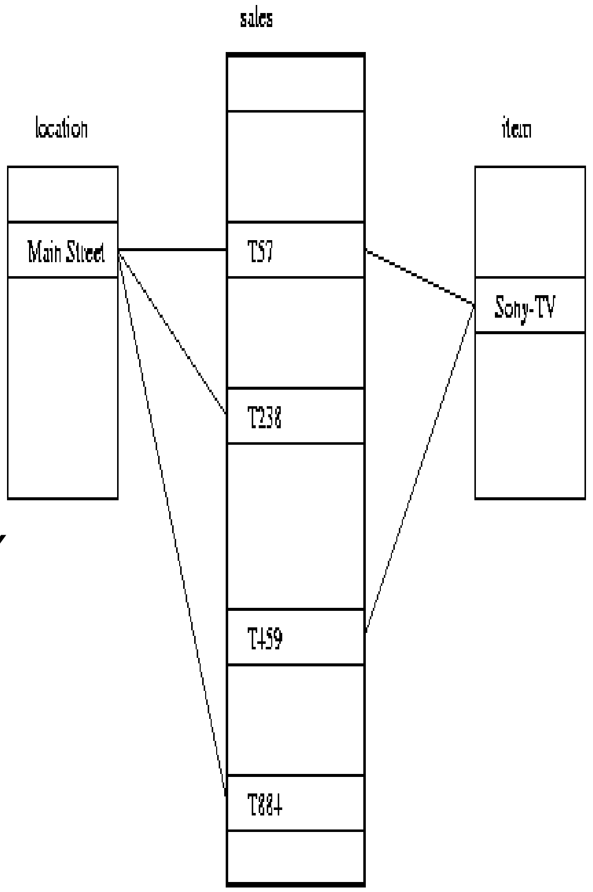
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

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: JI(R-id, S-id) where R (R-id, …) ⋈ S (S-id, …)
- 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
  - Join indices can span multiple dimensions



Efficient Processing OLAP Queries

- **Determine which operations** should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection
- **Determine which materialized cuboid(s)** should be selected for OLAP op.
  - Let the query to be processed be on {*brand, province_or_state*} with the condition "*year = 2004*", and there are 4 materialized cuboids available:

    1) {*year, item_name, city*}

    2) {*year, brand, country*}

    3) {*year, brand, province_or_state*}

    4) {*item_name, province_or_state*} where *year = 2004*

    Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability
- Multidimensional OLAP (MOLAP)
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas

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
- ☞ Data Generalization by Attribute-Oriented Induction
- Summary

Data Generalization

- Data generalization summarizes data by replacing relatively low-level values with high-level concepts, or by reducing the number of dimensions to summarize data in concept space involving fewer dimensions.
- Facilitating users to examine the general behavior of the data at multiple levels of abstraction
  - e.g., instead of examing individual custormer transactions, sales managers prefer to view data generalized to a higher level, such as summarized by customer groups according to geographic regions, frequency of purchases per group, customer income

Data Generalization: Concept Description

- Concept description is a form of data generalization
- A concept typically refers to a data collection, but is not a simple enumeration of data
- Concept description generates description for data generalization and comparsion
  - Characterization: provides a concise and succinct summarization of the given data collection
  - Discrimination: provides description comparing two or more data collection
- Complex data types, automation: AOI

Attribute-Oriented Induction

- Proposed in 1989 (KDD'89 workshop)
- Not confined to categorical data nor particular measures
- How it is done?
  - Collect the task-relevant data (*initial relation*) using a relational database query
  - Perform generalization by attribute removal or attribute generalization
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
  - Interaction with users for knowledge presentation

Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the *Big University* database

```
use Big_University_DB
mine characteristics as "Science Students"
in relevance to name, gender, major, birth_place,
                birth_date, residence, phone#, gpa
from student
where status in "graduate"
```

Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the *Big University* database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,

```
use Big_University_DB
Select * (i.e., name, gender, major, birth_place, birth_date,
   residence, phone#, gpa)
from student
where  student_status in {“Msc”, “MBA”, “PhD” }
```

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

Basic Principles of Attribute-Oriented Induction

- Data focusing: task-relevant data, including dimensions, and the result is the *initial working relation* -- Cor. Ana.
- Data Generalization
  - Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*'s higher level concepts are expressed in terms of other attributes
  - Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize A
- Attribute Generalization Control
  - Attribute-threshold control: typical 2-8, specified/default
  - Generalized relation threshold control: control the final relation/rule size

Attribute-Oriented Induction: Basic Algorithm

- InitialRel: Query processing of task-relevant data, deriving the *initial working relation*.
- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- PrimeGen: Based on the PreGen plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Attribute-Oriented Induction: Basic Algorithm

Algorithm: Attribute-oriented induction. Mining generalized characteristics in a relational database given a user's data mining request.

Input:

- *DB*, a relational database;
- *DMQuery*, a data mining query;
- *a_list*, a list of attributes (containing attributes, a_i);
- *Gen*(a_i), a set of concept hierarchies or generalization operators on attributes, a_i;
- *a_gen_thresh*(a_i), attribute generalization thresholds for each a_i.

Output: *P*, a *Prime_generalized_relation*.

Method:

```
1. $W \leftarrow$ get_task_relevant_data ($DMQuery$, $DB$); // Let $W$, the working relation, hold the
   task-relevant data.
```

2. prepare_for_generalization (W); // This is implemented as follows.

(a) Scan W and collect the distinct values for each attribute, a_i. (*Note:* If W is very large, this may be done by examining a sample of W.)

(b) For each attribute a_i, determine whether a_i should be removed. If not, compute its minimum desired level L_i based on its given or default attribute threshold, and determine the mapping pairs (v, v'), where v is a distinct value of a_i in W, and v' is its corresponding generalized value at level L_i.

3. $P \leftarrow$ generalization (W),

The *Prime_generalized_relation*, P, is derived by replacing each value v in W by its corresponding v' in the mapping while accumulating `count` and computing any other aggregate values.

This step can be implemented efficiently using either of the two following variations:

(a) For each generalized tuple, insert the tuple into a sorted prime relation P by a binary search: if the tuple is already in P, simply increase its `count` and other aggregate values accordingly; otherwise, insert it into P.

(b) Since in most cases the number of distinct values at the prime relation level is small, the prime relation can be coded as an m-dimensional array, where m is the number of attributes in P, and each dimension contains the corresponding generalized attribute values. Each array element holds the corresponding `count` and other aggregation values, if any. The insertion of a generalized tuple is performed by measure aggregation in the corresponding array element.

Class Characterization: An Example

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

Gender \ Birth_Region	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

Presentation of Generalized Results

- Generalized relation:
  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.
- Cross tabulation:
  - Mapping results into cross tabulation form (similar to contingency tables).
  - Visualization techniques:
  - Pie charts, bar charts, curves, cubes, and other visual forms.
- Quantitative characteristic rules:
  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

$$grad(x) \wedge male(x) \Rightarrow$$
$$birth\_region(x) = "Canada"[t:53\%] \vee birth\_region(x) = "foreign"[t:47\%].$$

Mining Class Comparisons

- Comparison: Comparing two or more classes
- Method:
  - Partition the set of relevant data into the target class and the contrasting class(es)
  - Generalize both classes to the same high level concepts
  - Compare tuples with the same high level descriptions
  - Present for every tuple its description and two measures
    - support - distribution within single class
    - comparison - distribution between classes
  - Highlight the tuples with strong discriminant features
- Relevance Analysis:
  - Find attributes (features) which best distinguish different classes

Mining Class Comparisons

- Example: comparison of the general properties of the graduate and undergraduate students at Big University

```
use Big University DB
mine comparison as "grad vs undergrad students"
in relevance to name, gender, major, birth place,
                birth date, residence, phone#, gpa
for "graduate students"
where status in "graduate"
versus "undergraduate students"
where status in "undergraduate"
analyze count%
from student
```

Class Comparison: An Example

Initial Target Relation

name	*gender*	*major*	*birth_place*	*birth_date*	*residence*	*phone#*	*gpa*
Jim Woodman	M	CS	Vancouver, BC, Canada	12-8-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	7-28-75	345 1st Ave., Vancouver	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	8-25-70	125 Austin Ave., Burnaby	420-5232	3.83
...	...	...	...	...	...	...	...

Initial Contrasting Relation

name	*gender*	*major*	*birth_place*	*birth_date*	*residence*	*phone#*	*gpa*
Bob Schumann	M	Chemistry	Calgary, Alt, Canada	1-10-78	2642 Halifax St., Burnaby	294-4291	2.96
Amy Eau	F	Biology	Golden, BC, Canada	3-30-76	463 Sunset Cres., Vancouver	681-5417	3.52
...	...	...	...	...	...	...	...

Class Comparison: An Example

Prime Generalized Relation

major	*age_range*	*gpa*	count%
Science	21...25	good	5.53
Science	26...30	good	5.02
Science	over_30	very good	5.86
...	...	...	...
Business	over_30	excellent	4.68

Prime Generalized Contrasting Relation

major	*age_range*	*gpa*	count%
Science	16...20	fair	5.53
Science	16...20	good	4.53
...	...	...	...
Science	26...30	good	2.32
...	...	...	...
Business	over_30	excellent	0.68

Concept Description vs. Cube-Based OLAP

- **Similarity**:
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing
- **Differences**:
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms

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
- Data Generalization by Attribute-Oriented Induction
- 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 (Online Analytical Mining)
- 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
- Data generalization: Attribute-oriented induction

References (I)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB’96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD’97
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE’97
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
- E. F. Codd, S. B. Codd, and C. T. Salley. Beyond decision support. Computer World, 27, July 1993.
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- A. Gupta and I. S. Mumick. Materialized Views: Techniques, Implementations, and Applications. MIT Press, 1999.
- J. Han. Towards on-line analytical mining in large databases. *ACM SIGMOD Record*, 27:97-107, 1998.
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD’96
- J. Hellerstein, P. Haas, and H. Wang. Online aggregation. SIGMOD'97

References (II)

- C. Imhoff, N. Galemmo, and J. G. Geiger. Mastering Data Warehouse Design: Relational and Dimensional Techniques. John Wiley, 2003
- W. H. Inmon. Building the Data Warehouse. John Wiley, 1996
- R. Kimball and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2ed. John Wiley, 2002
- P. O'Neil and G. Graefe. Multi-table joins through bitmapped join indices. *SIGMOD Record*, 24:8–11, Sept. 1995.
- P. O'Neil and D. Quass. Improved query performance with variant indexes. SIGMOD'97
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998
- S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. ICDE'94
- A. Shoshani. OLAP and statistical databases: Similarities and differences. PODS'00.
- D. Srivastava, S. Dar, H. V. Jagadish, and A. V. Levy. Answering queries with aggregation using views. *VLDB'96*
- P. Valduriez. Join indices. ACM Trans. Database Systems, 12:218-246, 1987.
- J. Widom. Research problems in data warehousing. CIKM'95
- K. Wu, E. Otoo, and A. Shoshani, Optimal Bitmap Indices with Efficient Compression, ACM Trans. on Database Systems (TODS), 31(1): 1-38, 2006

Surplus Slides

Compression of Bitmap Indices

- Bitmap indexes must be compressed to reduce I/O costs and minimize CPU usage—majority of the bits are 0's
- Two compression schemes:
  - Byte-aligned Bitmap Code (BBC)
  - Word-Aligned Hybrid (WAH) code
- Time and space required to operate on compressed bitmap is proportional to the total size of the bitmap
- Optimal on attributes of low cardinality as well as those of high cardinality.
- WAH out performs BBC by about a factor of two