COMP5121
Data Mining and Data Warehousing Applications

Week 4: Data Warehousing, OLAP, and Data Cube Technology

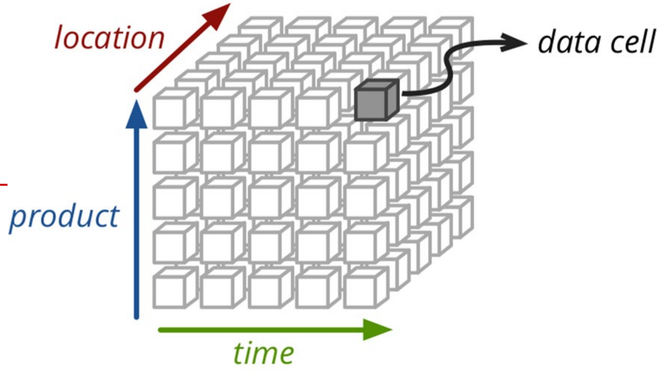
Dr. Fengmei Jin

- Email: fengmei.jin@polyu.edu.hk
- Office: PQ747 (+852 3400 3327)
- Consultation Hours: 2:30-4:30pm every Thursday

Outline

- Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Efficient Data Cube Computation Methods

What is a Data Warehouse?

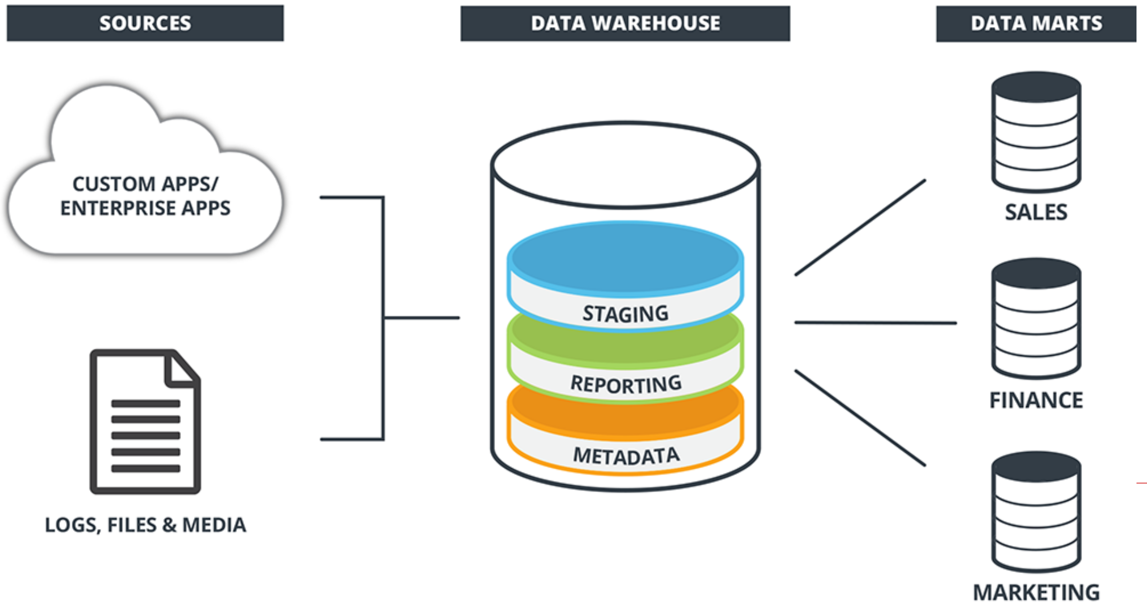


- Commonly defined as a:
  - **Subject-oriented system** that is organized for major entities such as customers, products, sales
  - **Integrated platform** that consolidates data from different sources
  - **Time-variant approach** that keeps historical snapshots of data over time.
  - **Non-volatile environment**: data is stable once loaded, primarily read-only.

"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 leveraging data warehouses for decision support
  - Maintains a specialized decision-support database separate from operational systems
  - Supports information processing via a solid platform for historical data analysis



Data Warehouse – *Subject-Oriented*

- Organized around major subjects
  - e.g., customer, supplier, product, and sales

- Focusing on modeling and analyzing data for decision makers
  - instead of daily operations or transaction processing

- Provide a simple and concise view of particular subject issues
  - by excluding data that are not useful in decision-making

Data Warehouse – *Integrated*

- Constructed by integrating heterogeneous data sources
  - e.g., relational DBs, flat files, on-line transaction records, …
- Data Cleaning and Data Integration techniques are applied.
  - To ensure consistency in naming conventions, encoding structures, attribute measures, and so on.
  - For example, hotel price may differ in currency, tax, breakfast inclusion, and parking fees.
- **When data is moved to the warehouse, it is converted!**

Data Warehouse – *Time-Variant*

- The time horizon for the data warehouse is much longer than that of operational systems
  - **Operational DB**: current value data
  - **Data warehouse**: from a historical view (e.g., past 5-10 years)

- Every key structure in the data warehouse includes a time element, either explicitly or implicitly.
  - For example, sales data might be aggregated by month, even if each individual record doesn't have a specific date attached.
  - But the key in operational DBs may or may not contain such "time element" – e.g., a customer table.

Data Warehouse – *Non-volatile*

- **Independence:** A physically separate store of data
  - Transformed and independent from the application data found in the operational environment

- **Static**: Operational updates do not occur in data warehouses
  - No transaction processing, recovery, and concurrency controls
  - Only two required operations in data accessing: initial loading of data and access to data

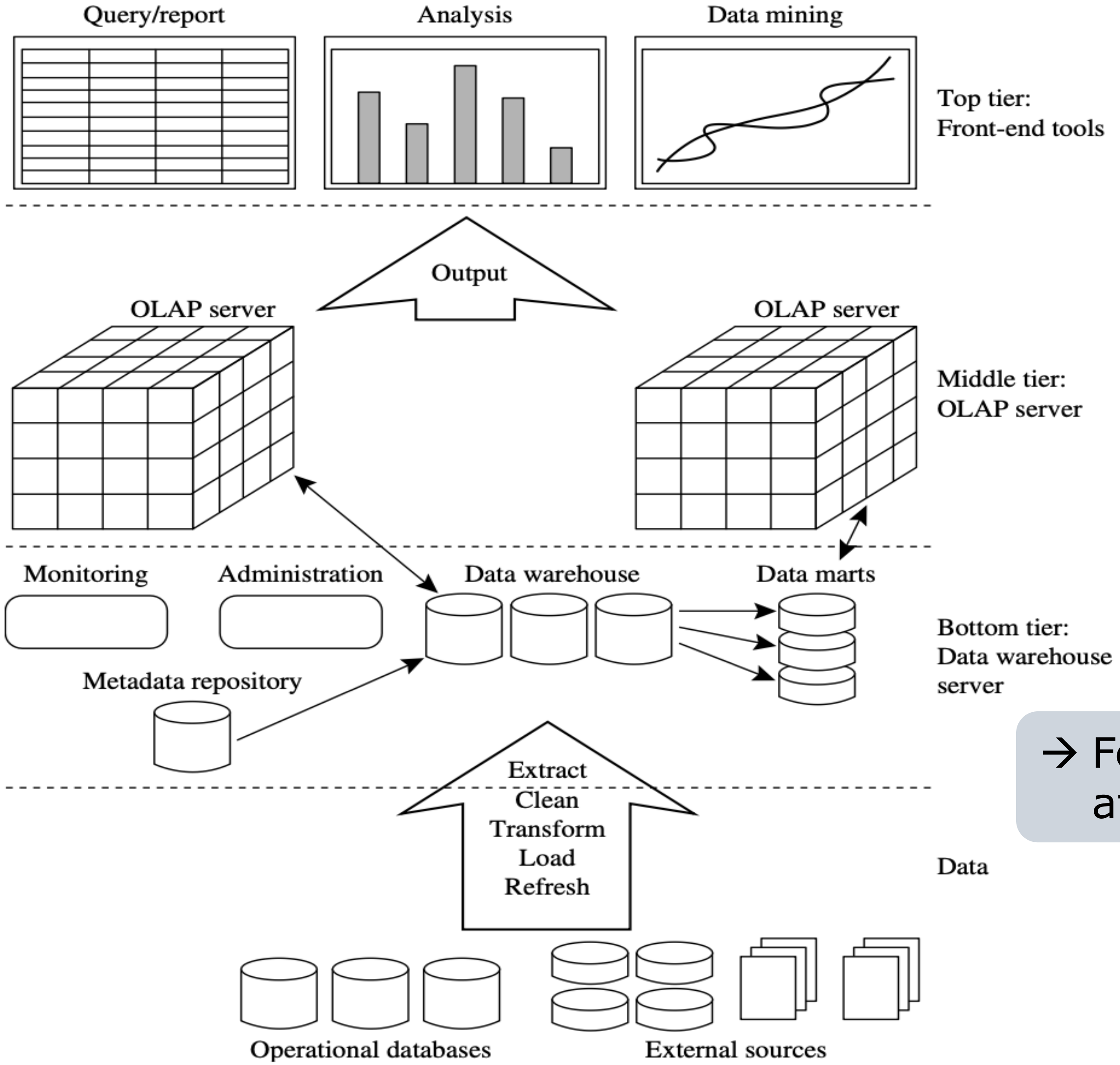
OLTP vs. OLAP

	OLTP in operational DB	OLAP in data warehouse
Full Name	On-Line Transactional Processing	On-Line Analytical Processing
Functions	customer-oriented: daily operations, transaction and query processing	market-oriented: complex query, data analysis, decision support
Users	clerks, clients, DBA, IT professionals	knowledge worker: managers, executives, analysts, etc.
Contents	current, up-to-date data – too detailed to be easily used for decision making	large amounts of historical data – summarized, integrated, information at different granularities
DB Design	application-oriented: entity-relationship (ER) data model	subject-oriented: a star or a snowflake model
View	within an enterprise or department	multiple versions of a DB schema; data from different organizations
Access	data in: read/write, index/hash on prime key	information out: lots of read-only scans
Size	tens records; GB to high-order GB; thousands users	millions records; ≥ TB; hundreds users

Why a Separate Data Warehouse?

- High performance for both systems:
  - DBMS – tuned for OLTP: access methods, indexing, hashing, concurrency control, recovery
  - Warehouse – tuned for OLAP: complex OLAP queries, consolidation, multi-dimensional view
- Different data and functions:
  - Data warehouses are structured for analysis, with standardized schemas and consolidated information from diverse sources.
  - Data warehouses support complex analytics on historical data. Operational databases handle frequent transactions and updates.
- *Some systems perform OLAP directly on DBs, but performance and scalability may be limited.*

Data Warehouse: A Multi-Tiered Architecture



→ **Top tier**: front-end tools provide query, reporting, analysis, and data mining tools (e.g., trend analysis, predictions)

→ Feed data into **bottom tier** after preprocessing

Three Types of Data Warehouse Models

- **Enterprise Warehouse** (full-size)
  - Collect all information about subjects spanning the entire organization
    - corporate-wide data integration, cross-functional in scope, both detailed and summarized data, size ranging from a few gigabytes to terabytes, or beyond.
    - extensive business modeling and take years to design / build using high-performance platforms like supercomputers
- **Data Mart** (partial)
  - A subset of corporate-wide summarized data – tailored to meet the needs of a specific group of users
    - low-cost departmental servers, simpler implementation cycle
- **Virtual Warehouse** (minimal)
  - A set of views over operational databases – only some of the possible summary views may be materialized.

Extraction, Transformation, and Loading (ETL)

- **Extraction:** gather data from multiple/external sources
- **Cleaning:** detect errors and rectify them when possible
- **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

- **Metadata** is the data defining warehouse objects. It stores:
  - Description of the data warehouse structure: schema, views, dimensions, hierarchies, derived data definition, data mart locations and contents
  - Operational metadata: data lineage, currency of data (active, archived, or purged), monitoring information (usage, statistics, error reports)
  - Algorithms used for summarization
  - The mapping from operational environment to the data warehouse
  - Data related to system performance: *indices and profiles* for data access and retrieval performance, *rules* for the timing and scheduling of refresh, update, and replication cycles
  - Business data: business terms/definitions, data ownership, charging policies

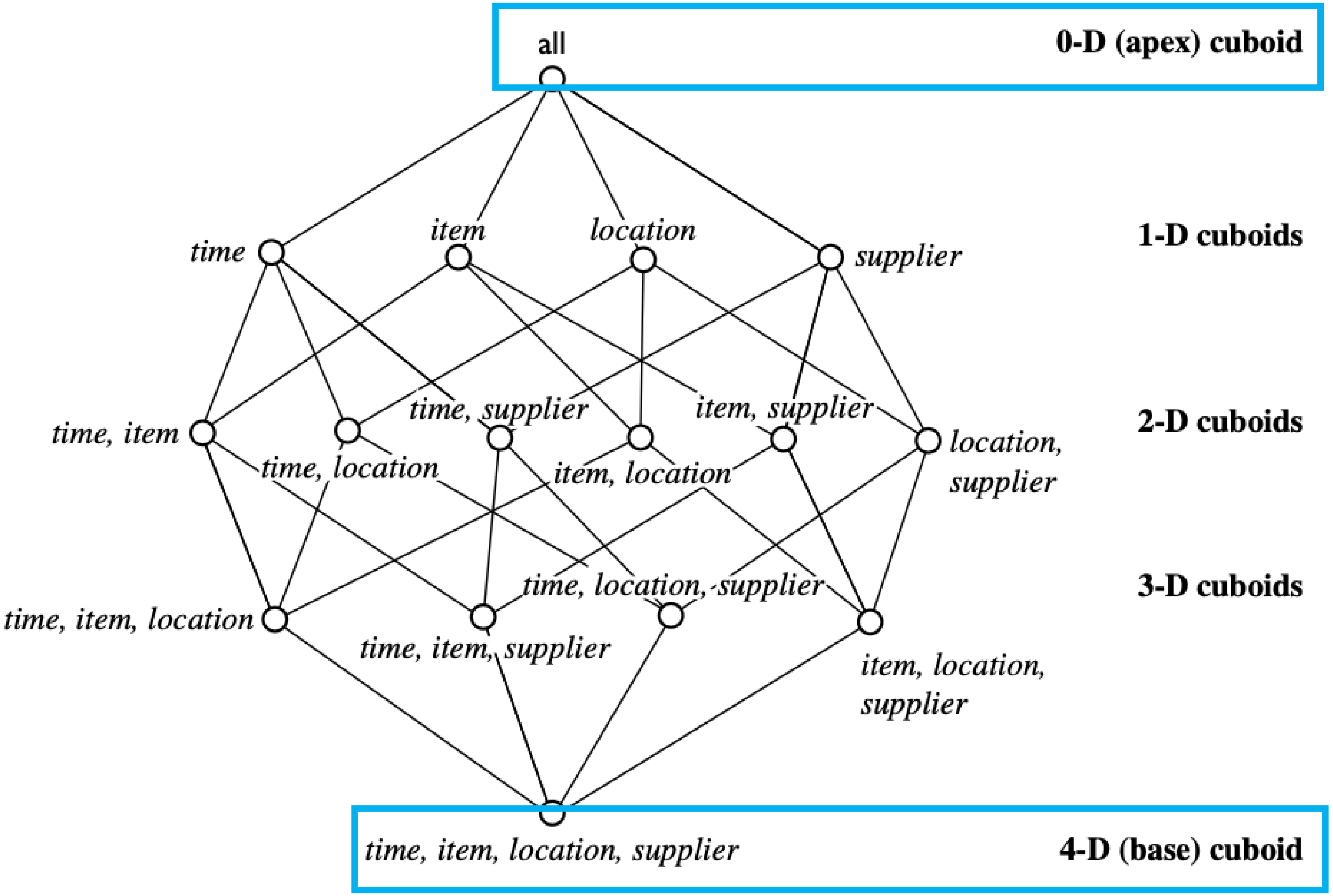
multidimensional data model, data cube, online analytical processing

DATA WAREHOUSE MODELING: DATA CUBE AND OLAP

From Tables and Spreadsheets to Data Cubes

- A data warehouse is based on a multi-dimensional data mode, which *views data* in the form of a data cube, defined by:
  - **Dimension tables**: to describe a dimension, e.g., item (*item_name*, *brand*, *type*), or time (*day*, *week*, *month*, *quarter*, *year*)
  - **Fact table**: to store numeric measures (e.g., *dollars_sold*) and keys linking to dimension tables – analyze relationships between dimensions
- Data cube is typically n-dimensional.
  - The n-dimensional base cube is called a **base cuboid**.
  - The topmost 0-dimensional cuboid, which provides the highest-level summarization, is called the **apex cuboid**.
  - All levels of cuboids form the entire data cube.

Example: Structure of Data Cube



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

Example: a 3-D Data Cube

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

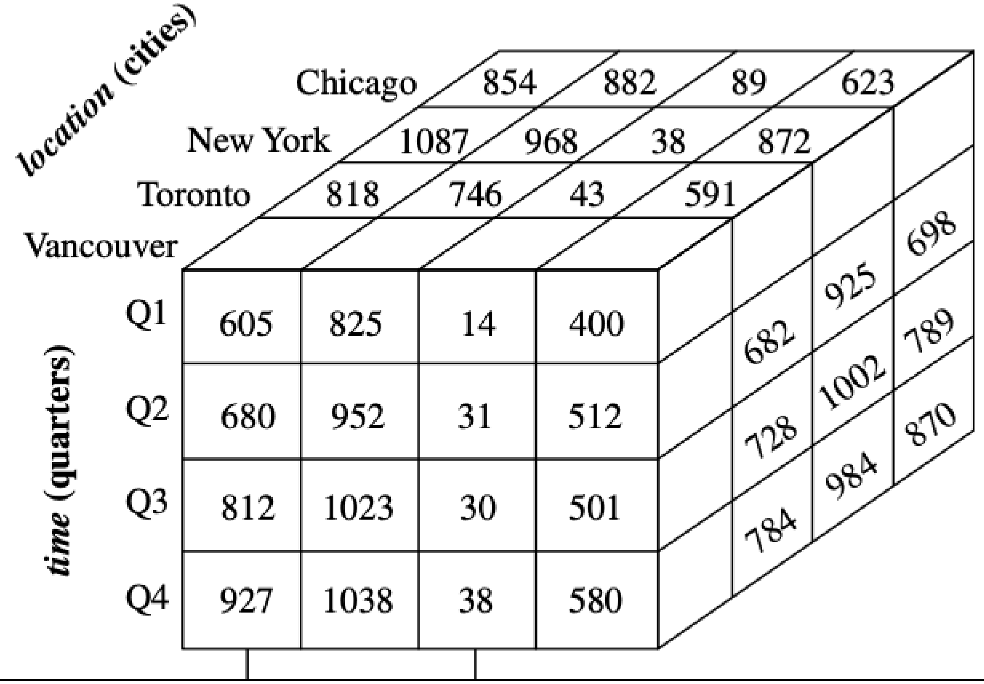
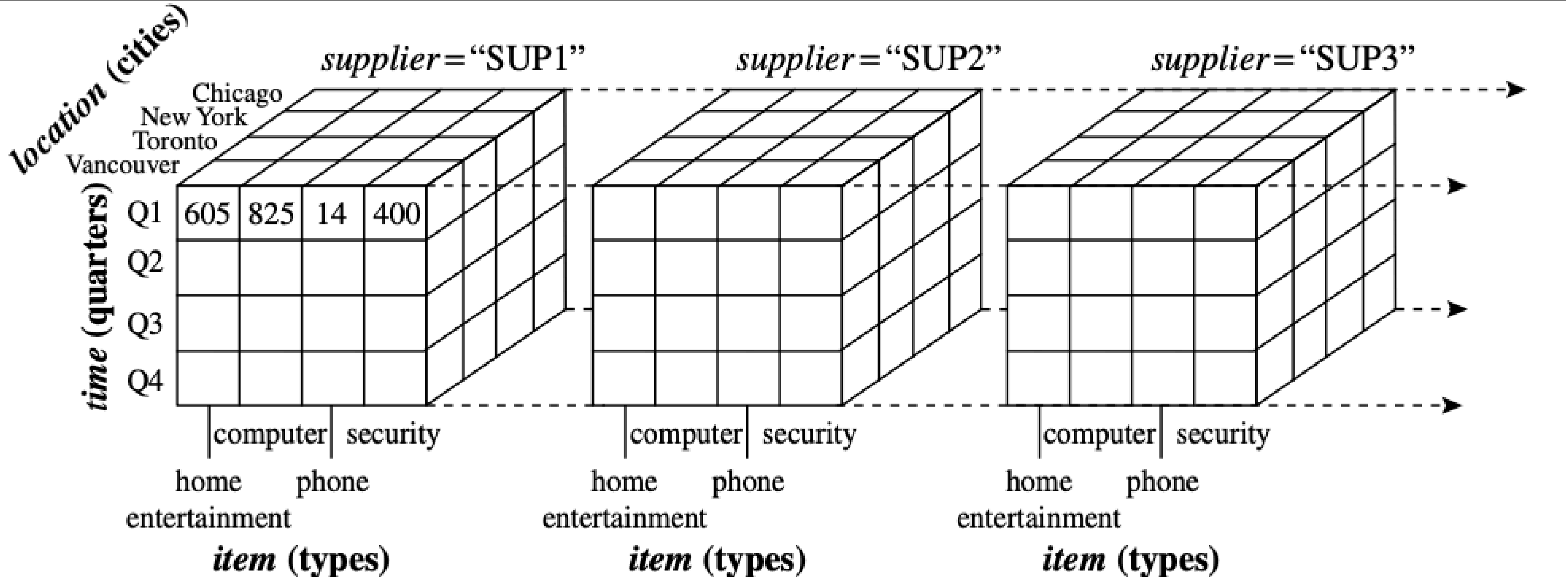


Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, an

	location = "Chicago"				**location** = "New York"				**location** = "Toronto"			
	item				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784

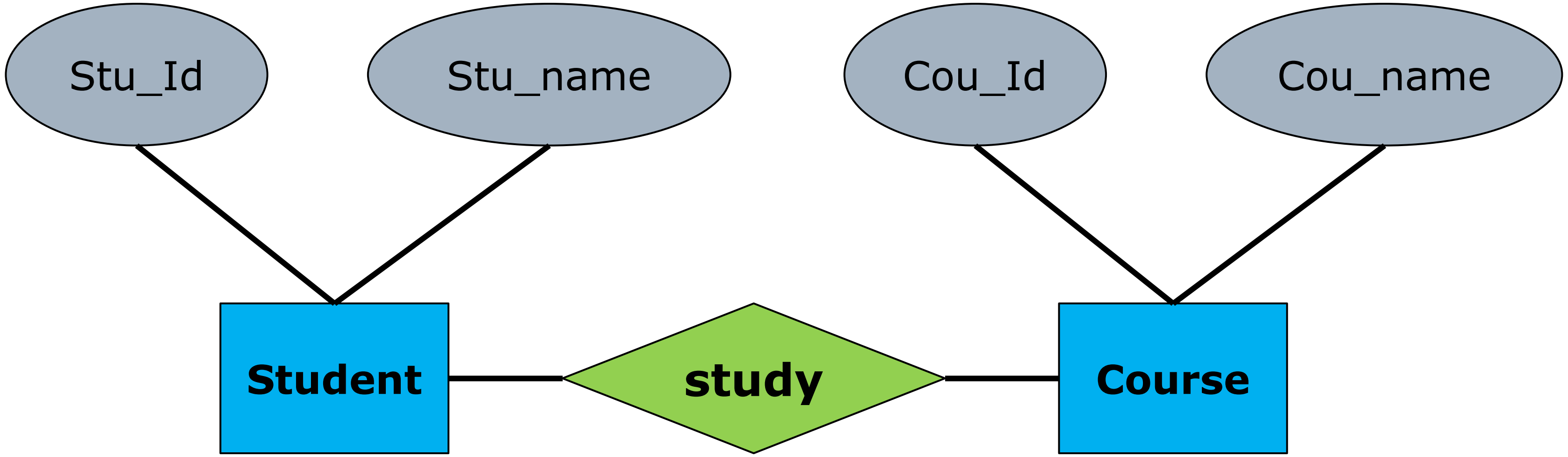
Note: The measure displayed is *dollars_sold* (in thousands).



A 4-D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*.

Schemas for Multi-dimensional Data Models

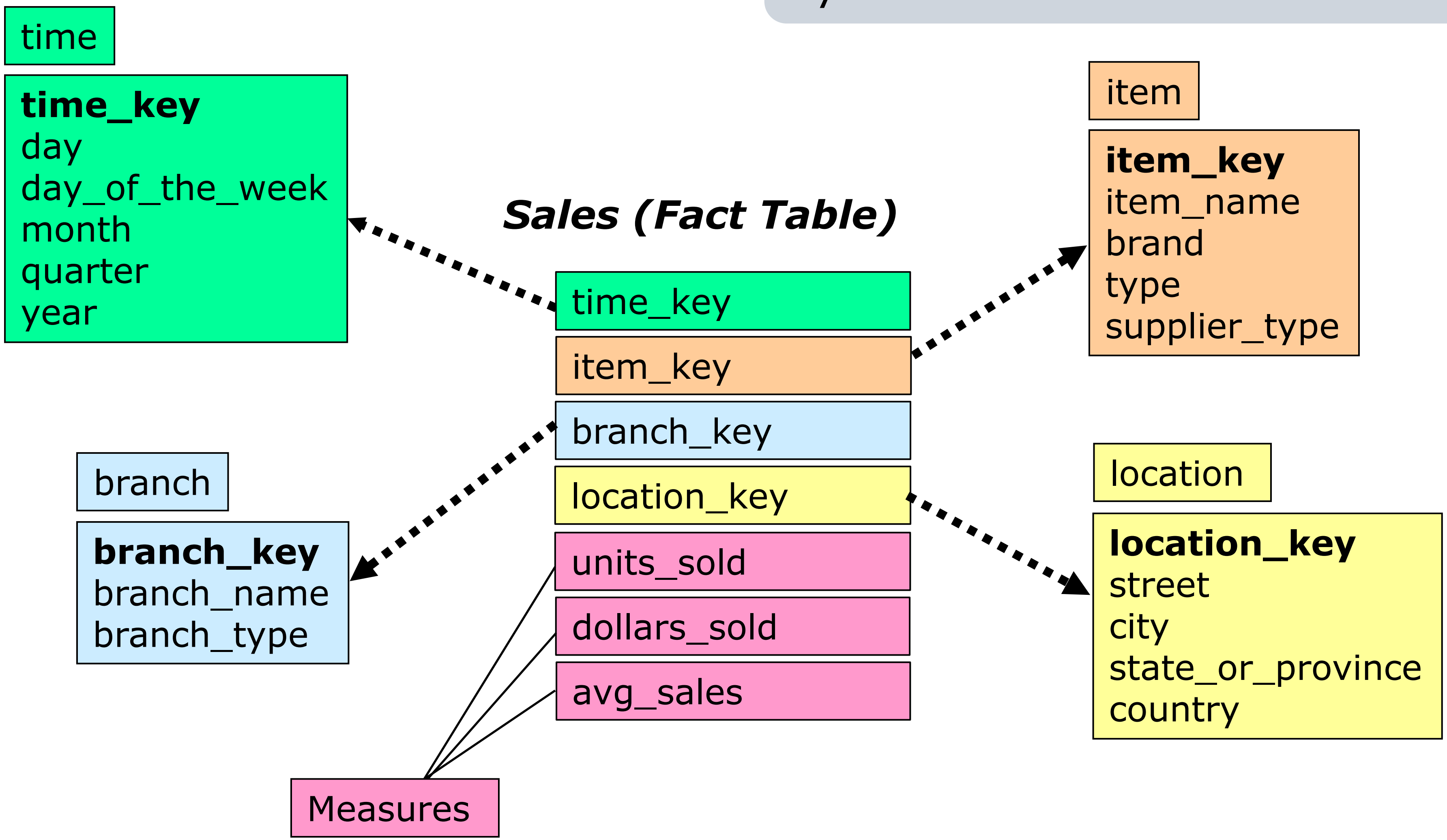
- Entity-Relationship (ER) model and the schema
  - a set of entities and their relationships – *appropriate for OLTP*



- A multi-dimensional model for data warehouses: focus on dimensions and measures, in the form of:
  - star schema, snowflake schema, fact constellation schema

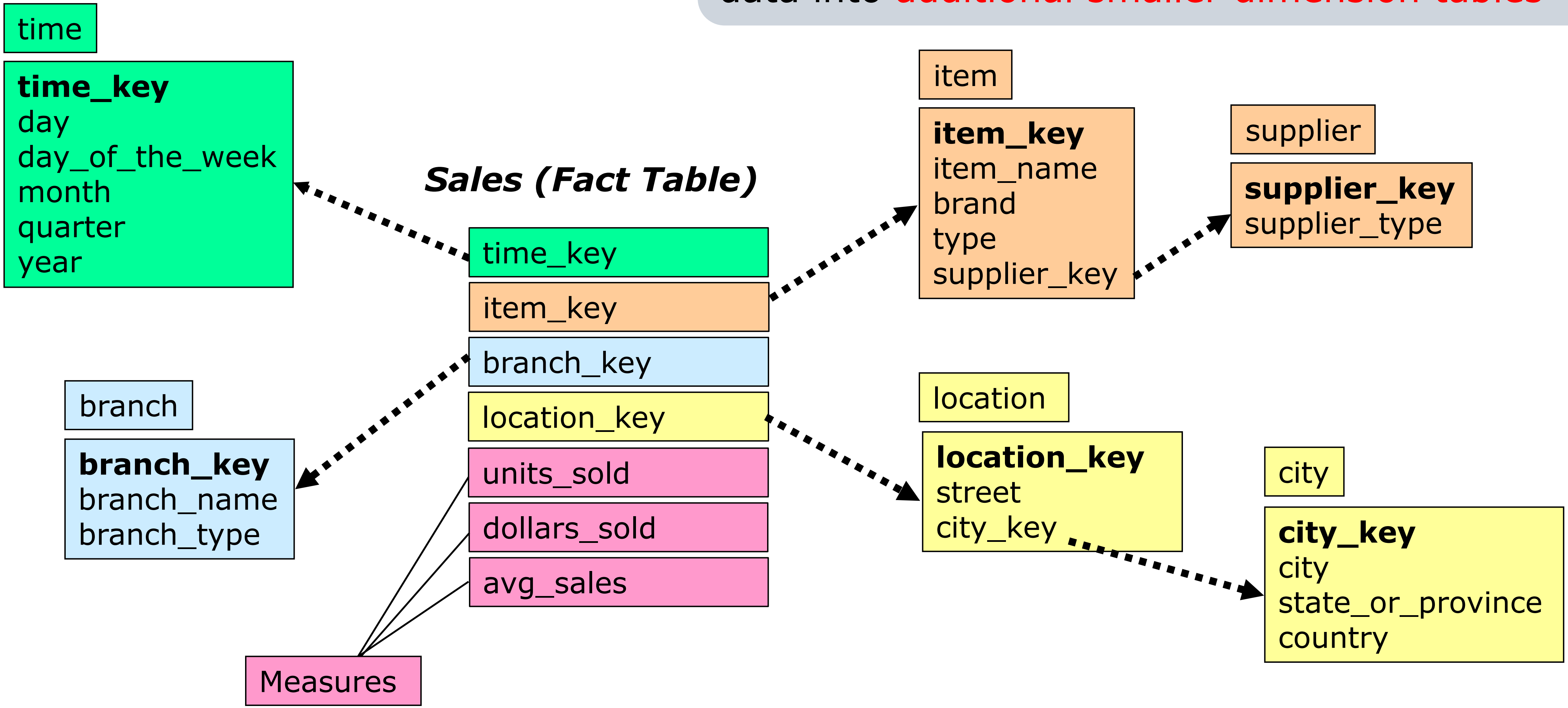
(1) Star Schema

A fact table in the center, surrounded by a set of dimension tables



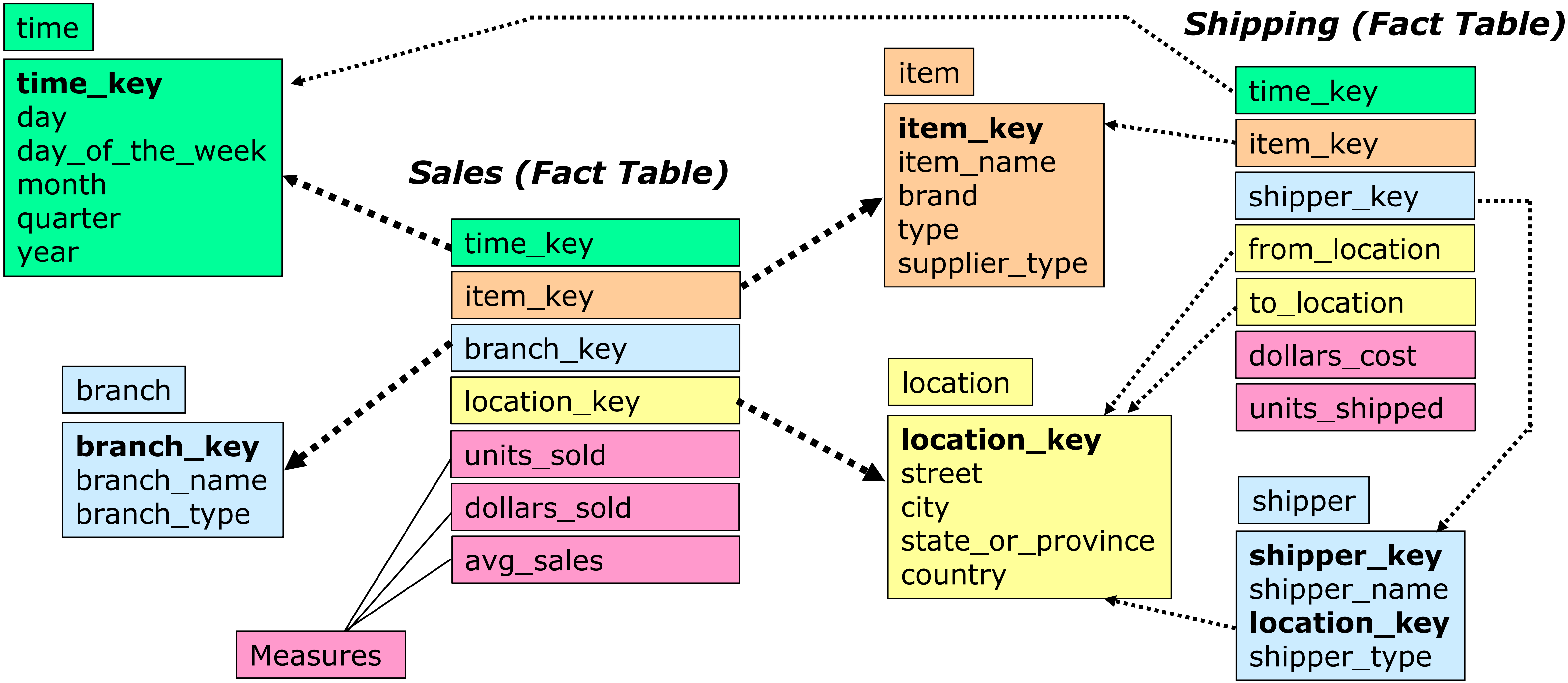
(2) Snowflake Schema

A variant of star schema: some dimensional hierarchy is normalized → thereby splitting the data into additional smaller dimension tables

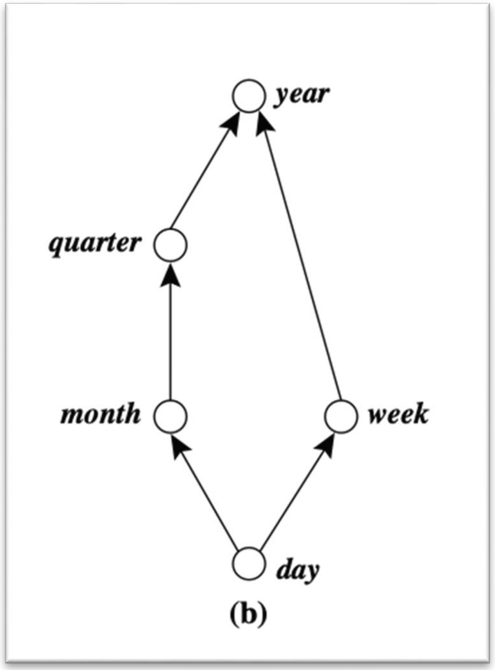


(3) Fact Constellation

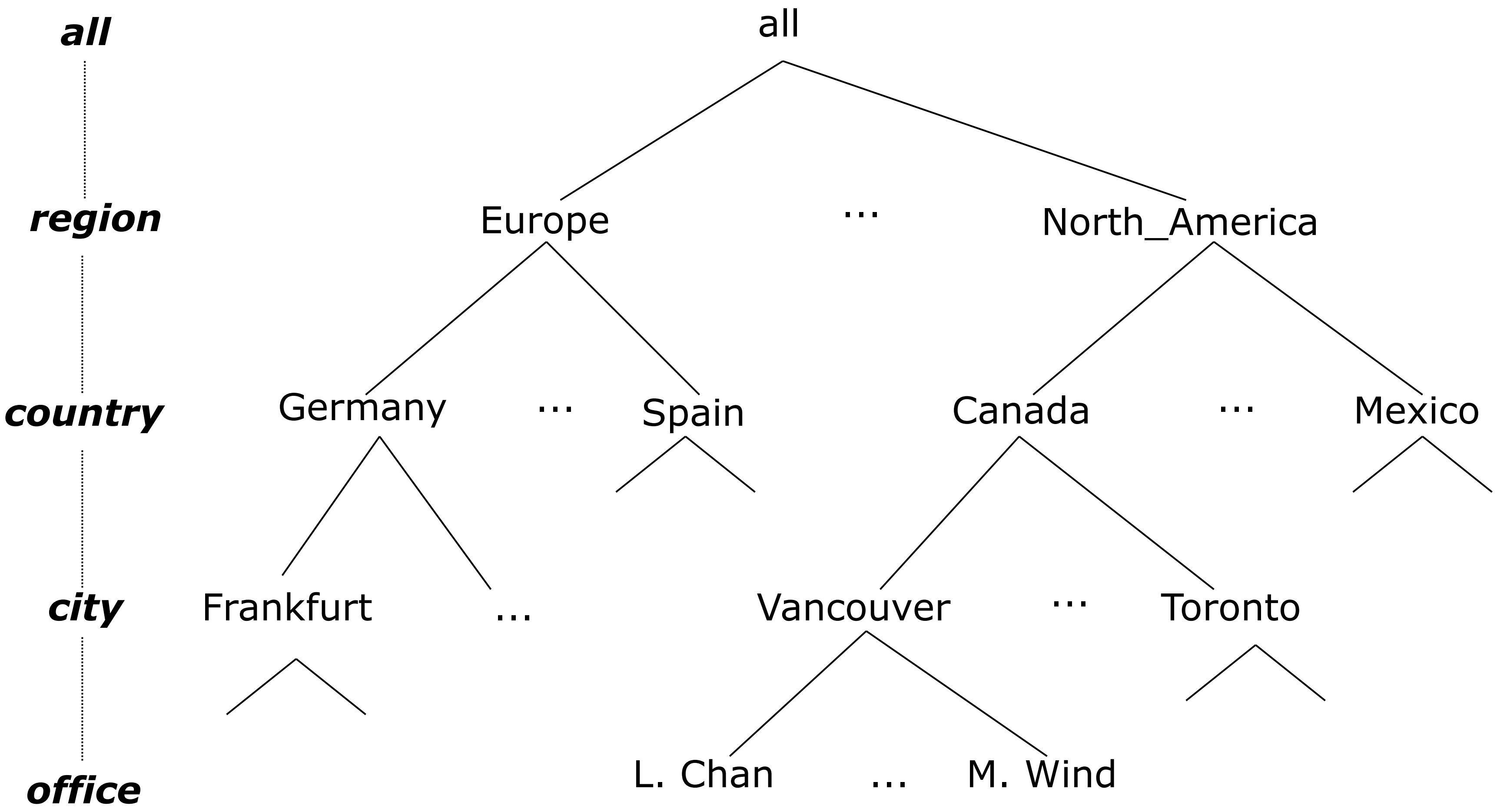
allow dimension tables to be shared between multiple fact tables → a collection of stars



The Role of Concept Hierarchies

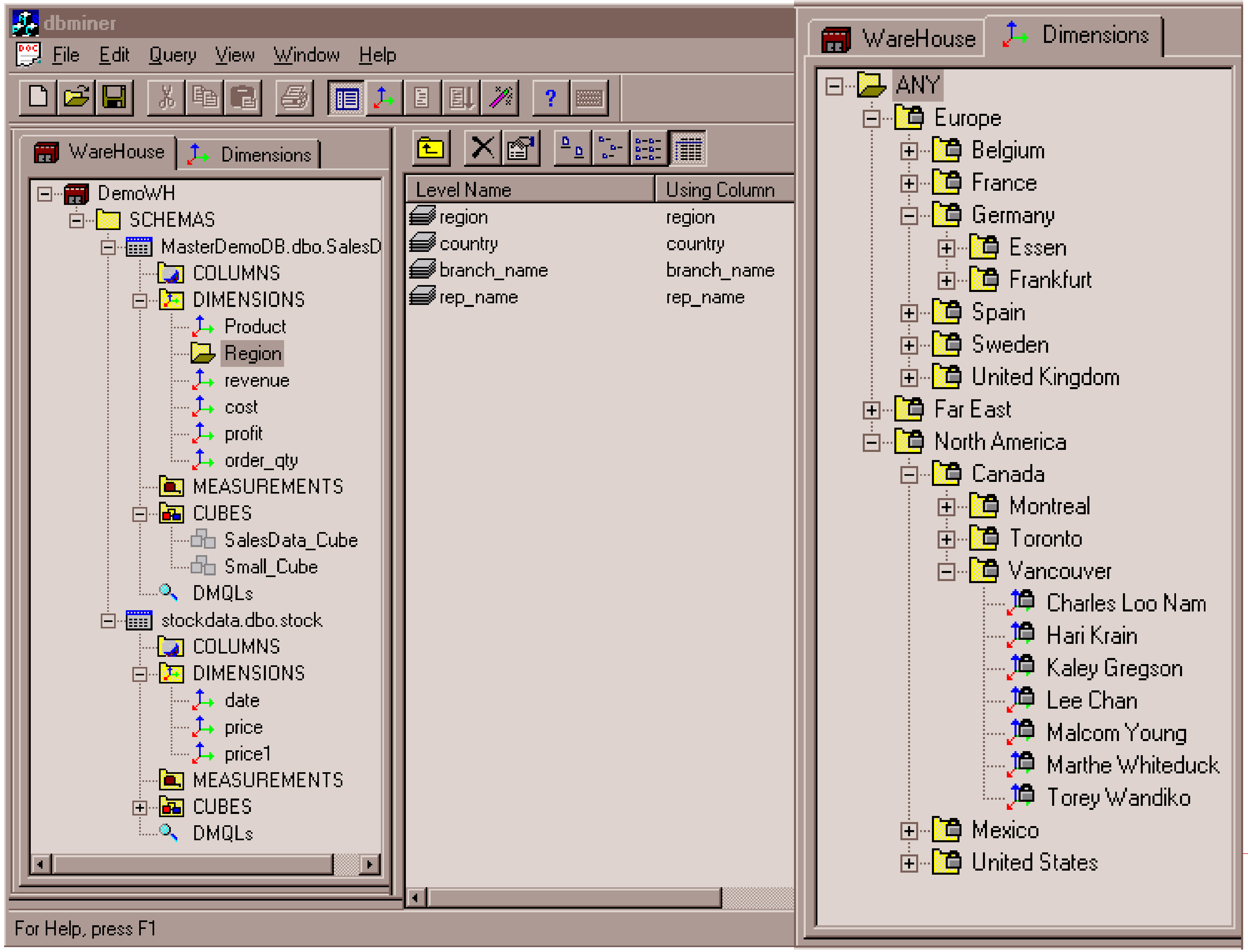


A lattice for *time*



A hierarchy for *location*

View of Warehouses and Hierarchies



Specification of hierarchies

1) Schema hierarchy:

- day < month < quarter;
- day < week < year

2) Set_grouping hierarchy

$\{1 \ldots 10\}$ < inexpensive

Data Cube Measures

- How are measures computed?
  - **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., $count()$, $sum()$, $min()$, $max()$

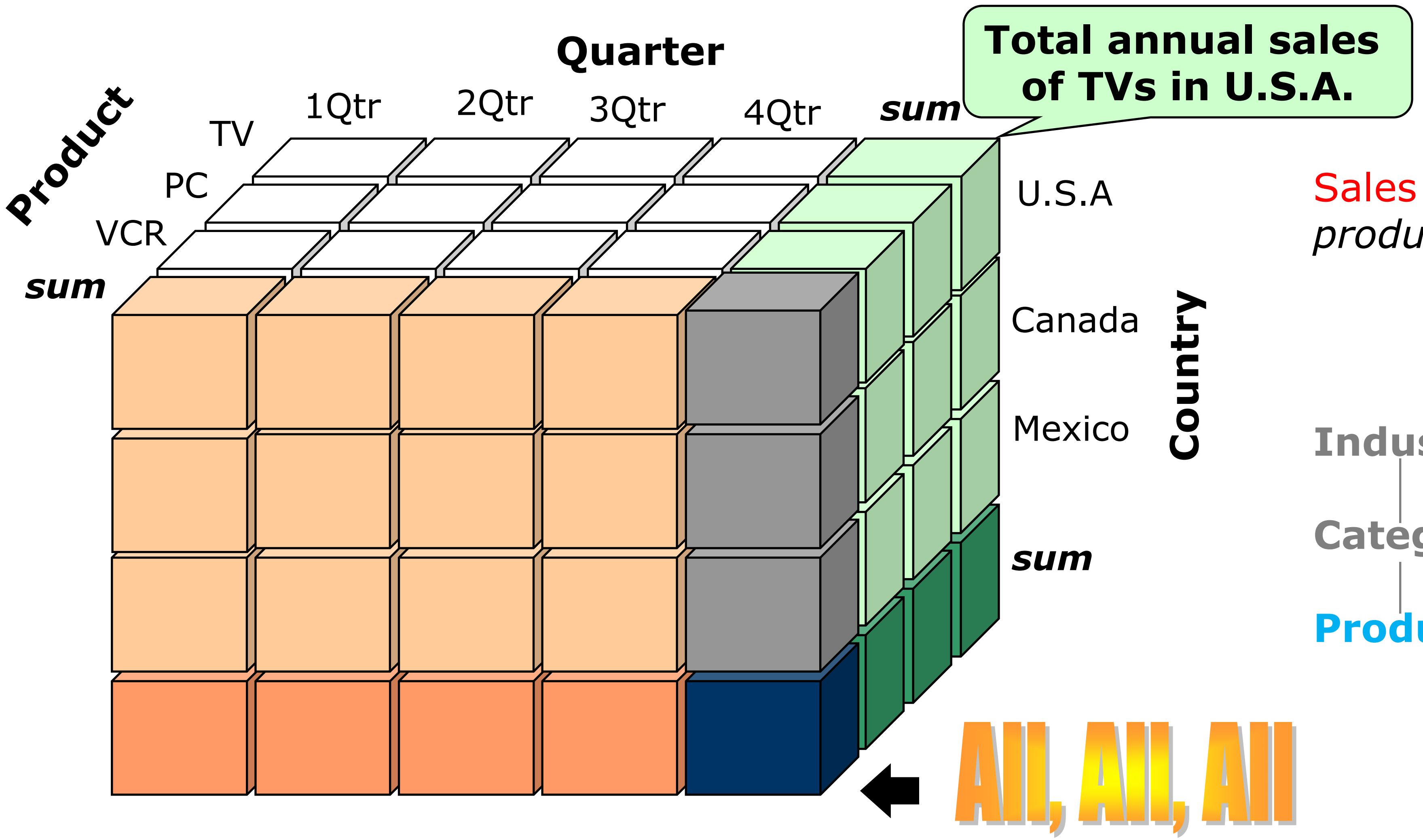
  - **Algebraic**: if it can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function

    e.g., $avg(x) = sum(x)/count(x)$

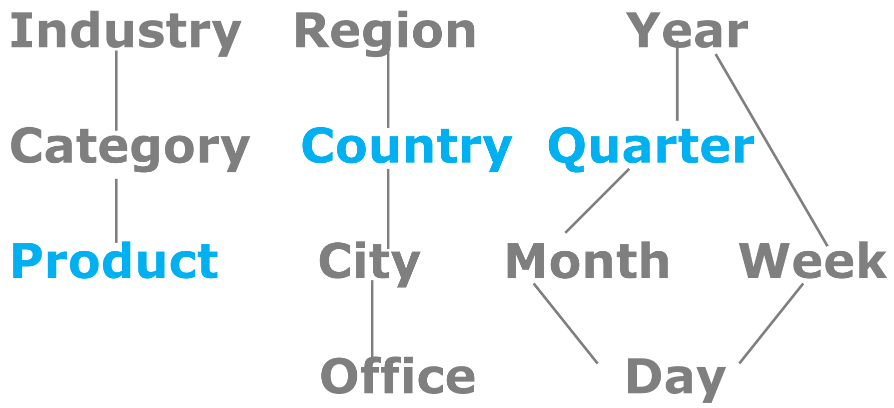
  - **Holistic**: if there does not exist an algebraic function that characterizes the computation

    e.g., $median()$, $mode()$, $rank()$

A Sample Data Cube



Sales volume as a function of *product*, *quarter*, and *country*



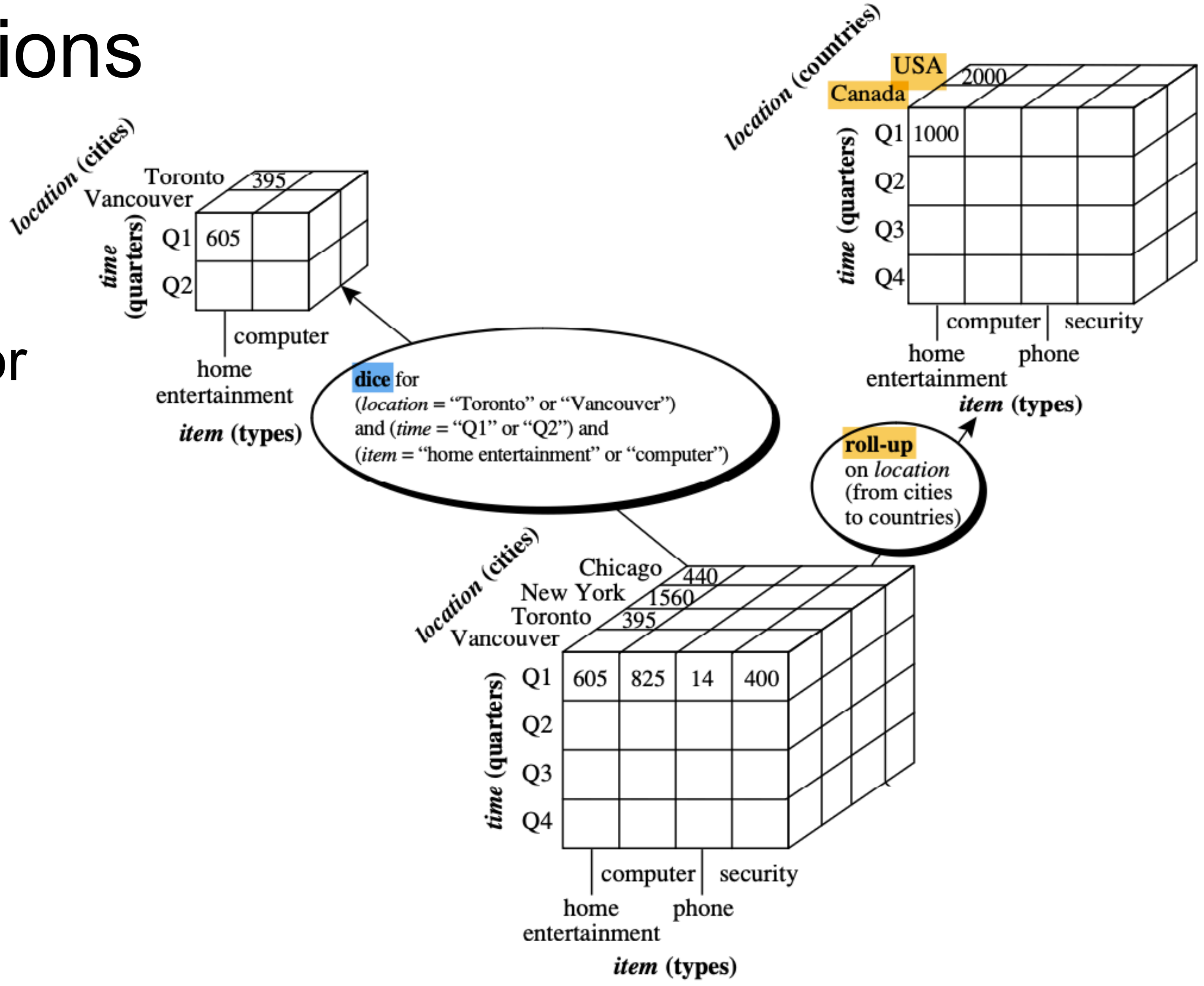
Typical OLAP Operations

- Roll up (drill-up): summarize data by climbing up hierarchy or by dimension reduction techniques
- 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)

Typical OLAP Operations

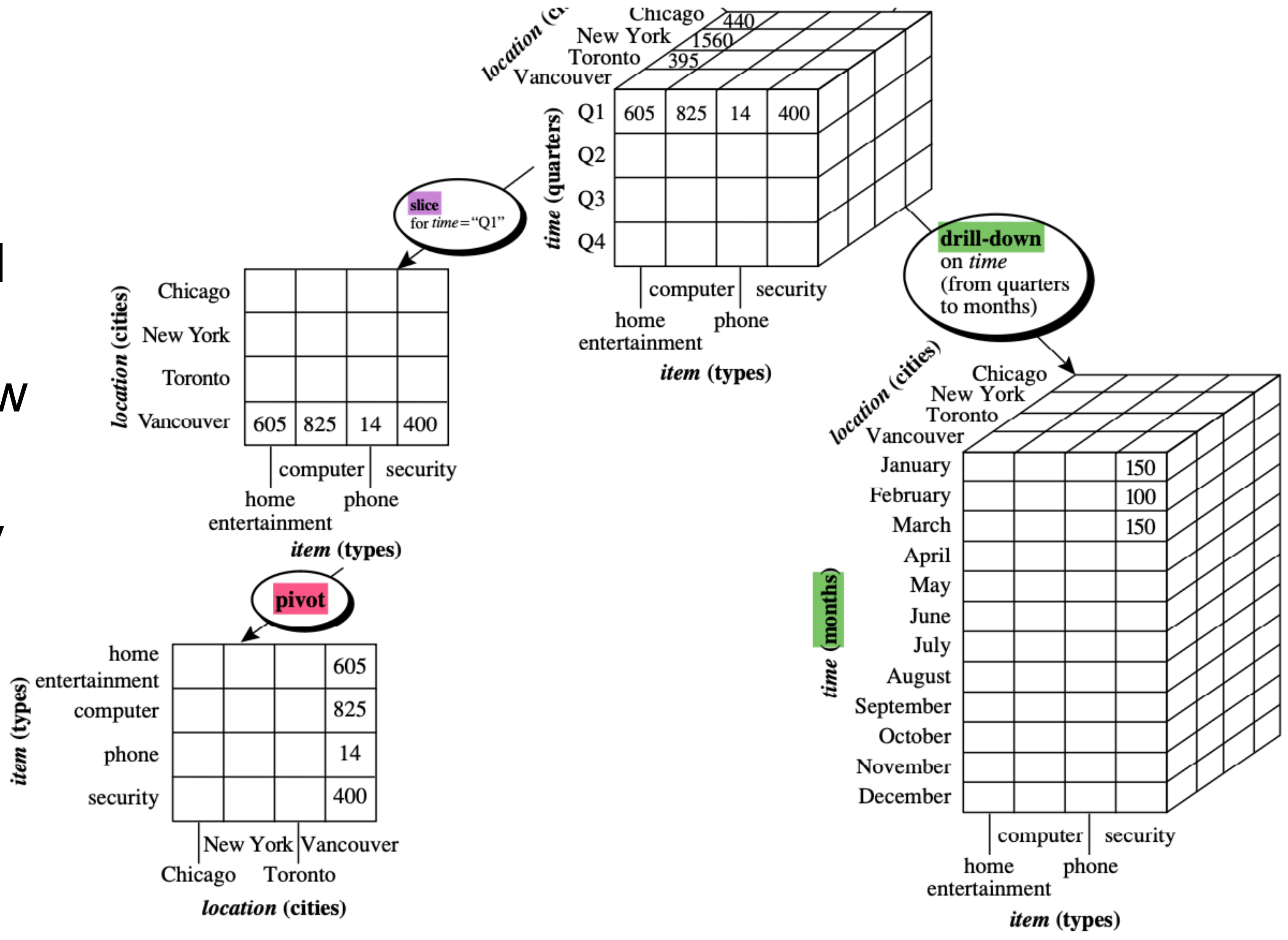
- Roll up (drill-up)
  - **summarize data** by climbing up hierarchy for a dimension or by dimension reduction

- Dice
  - define a subcube by performing a **selection** on two or more dimensions

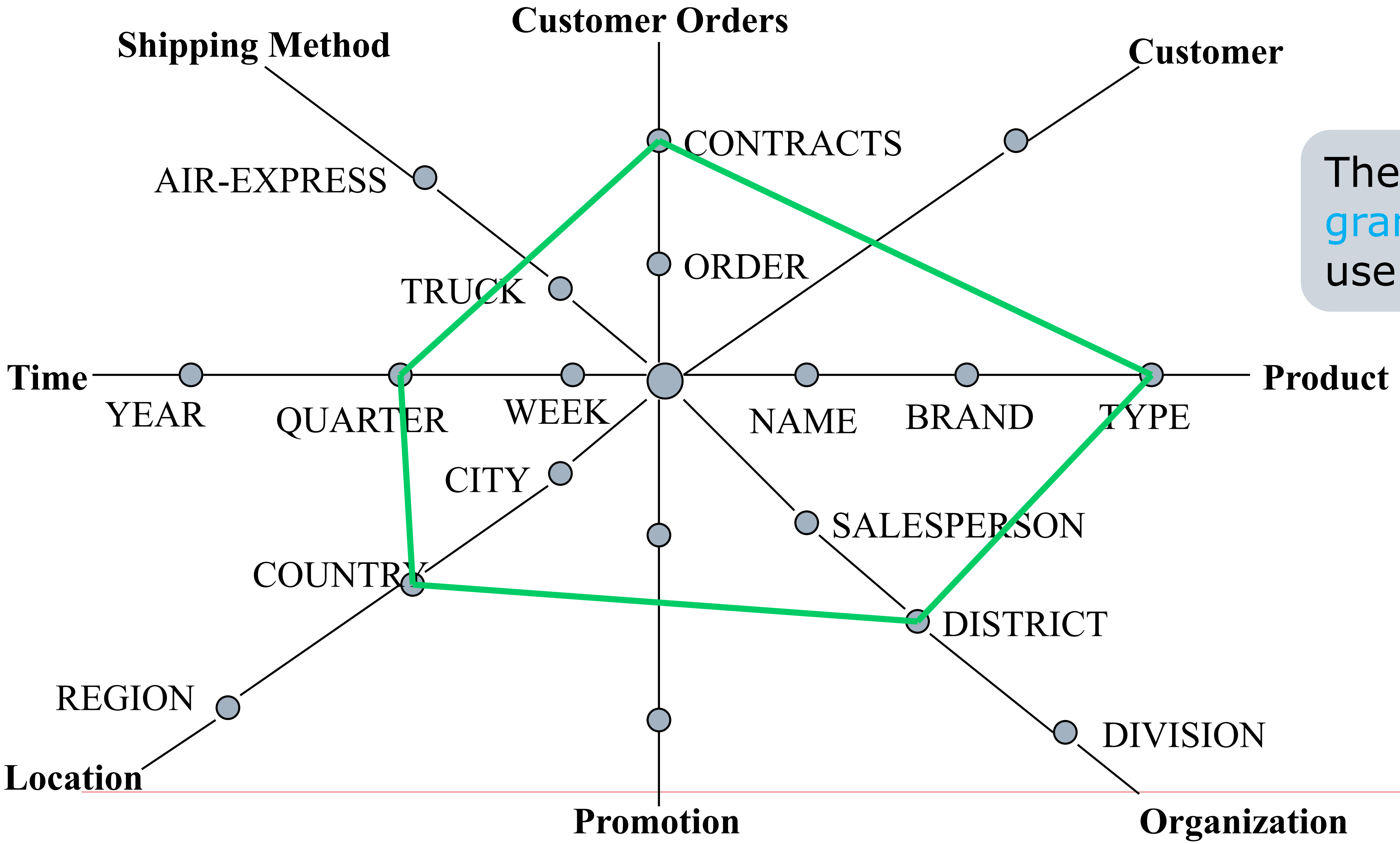


Typical OLAP Operations

- Roll-down (drill-down): reverse of roll-up
  - from higher-level summary to lower-level summary or detailed data, or introducing new dimensions
- Slice: define a subcube by performing a **selection** on one dimension
- Pivot (rotate): reorient the cube, visualization, 3D to series of 2D planes



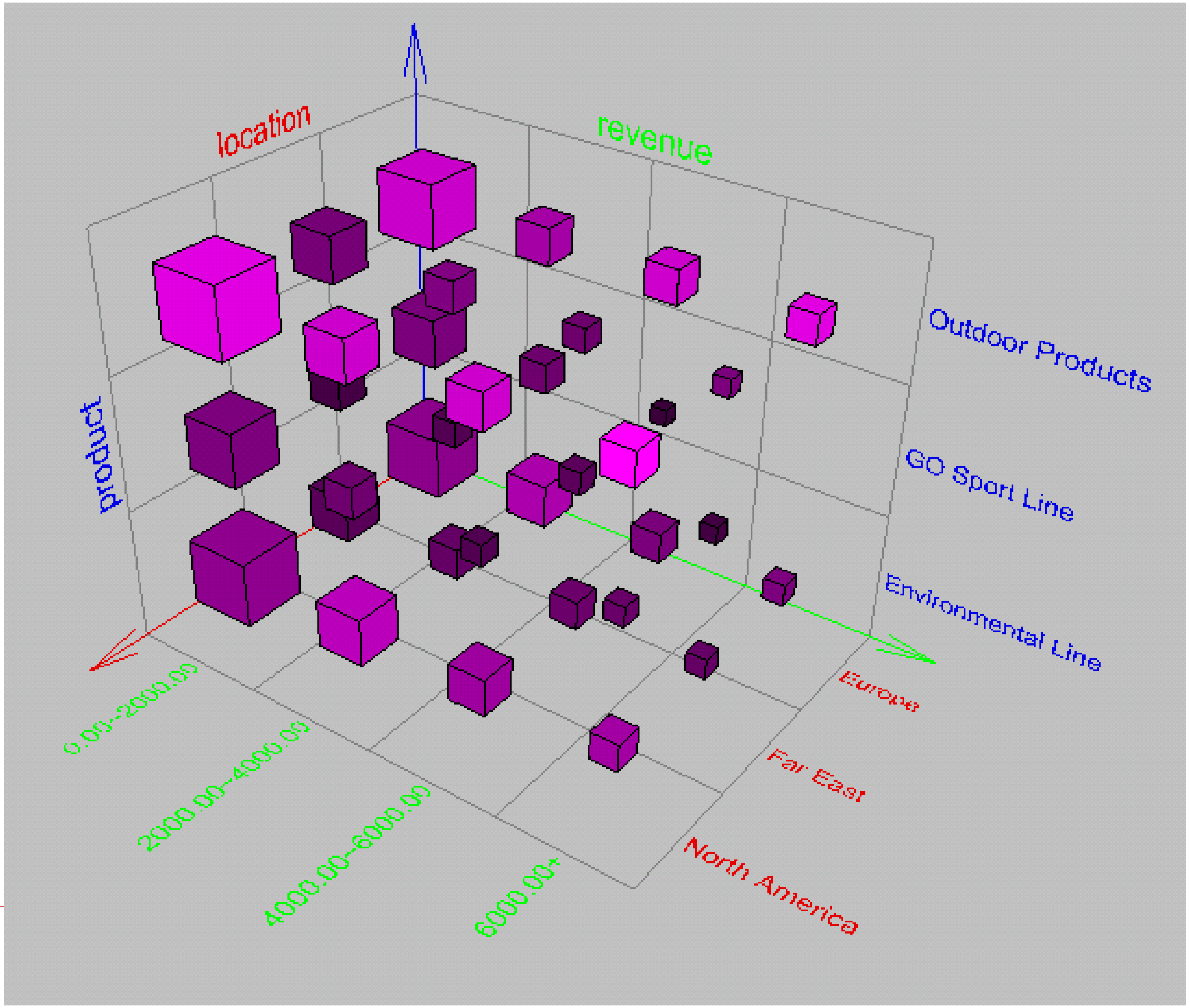
A Star-Net Query Model



These represent the granularities available for use by OLAP operations.

Browsing a Data Cube

- Visualization
- OLAP capabilities
- Interactive manipulation



business analysis, information processing, data mining, …

DATA WAREHOUSE DESIGN AND USAGE

A Business Analysis Framework for Data Warehouse Design

- Four views regarding the design of data warehouse
  - **Top-down View** – focus on selecting relevant information to align the data warehouse with current / future business needs
  - **Data Source View** – expose the information being captured, stored, and managed by operational systems
  - **Data Warehouse View** – represent the fact / dimension tables
  - **Business Query View** – see the perspectives of data in the warehouse from the end-user's viewpoint

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)

- Typical Design Process
  - Choose a business process to model, e.g., orders, invoices, shipments, etc.
  - Choose the business process grain (atomic level of data, e.g., transaction)
  - Choose the dimensions that will apply to each fact table record (time, item, …)
  - Choose the measure that will populate each fact table record (dollars_sold)

Data Warehouse Usage

- Three kinds of data warehouse applications:
  - **Information Processing:** supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts
    - low-cost web-based accessing tools integrated with web browsers
  - **Analytical Processing:** multi-dimensional data analysis
    - basic OLAP operations, slice-dice, drilling, pivoting
  - **Data Mining:** knowledge discovery
    - hidden patterns and associations, analytical models, classification and prediction, presenting the mining results using visualization

From OLAP to On-Line Analytical Mining (OLAM)

- Why online analytical mining?
  - High-quality data in data warehouses – integrated, consistent, cleaned data
  - Available information processing infrastructure surrounding data warehouses
    - accessing, integration, consolidation, and transformation of multiple DBs, Web accessing and service facilities, and reporting and OLAP tools
  - OLAP-based exploration of multi-dimensional data
    - Mining with drilling, dicing, slicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, tasks

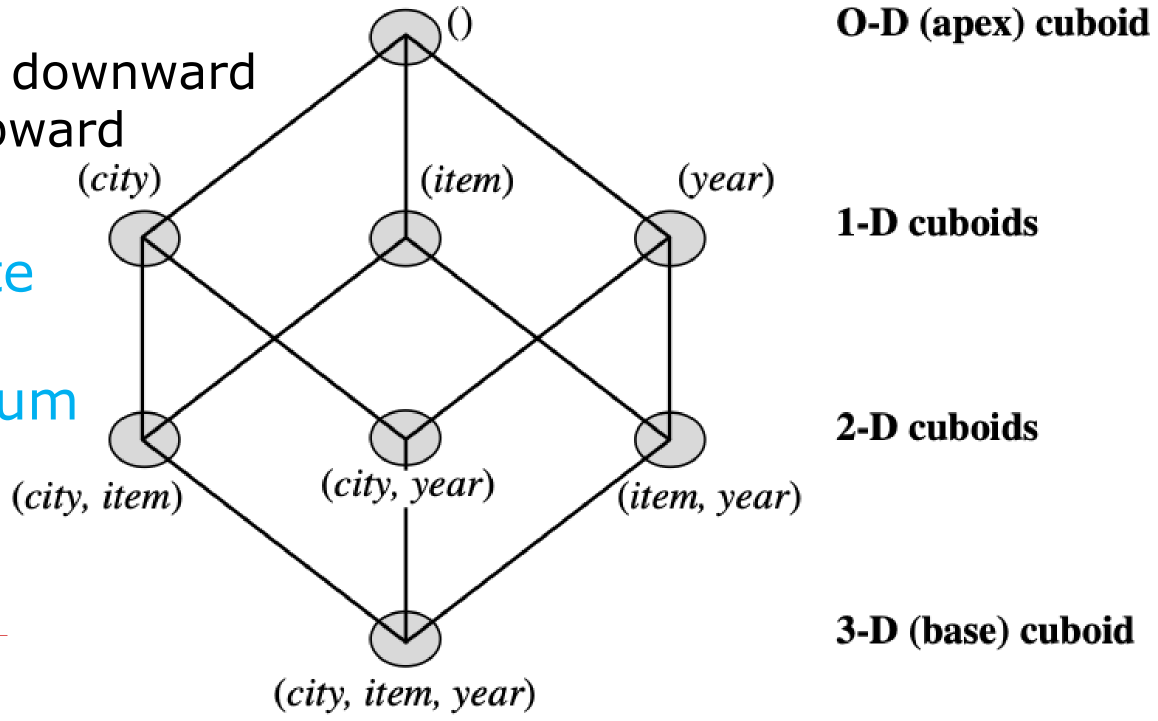
compute cube op, iceberg cube, bitmap, join index, ...

EFFICIENT DATA CUBE COMPUTATION

Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the **base** cuboid – the most specific
  - The top-most cuboid (**apex**) contains only one cell – the most generalized (all)

- **Drilling down**: start from apex cuboid and explore downward
- **Rolling up**: start at the base cuboid and explore upward



- **0-D op**: i.e., no group-by SQL, like "compute the sum of total sales"
- **1-D op**: one group-by, e.g., "compute the sum of sales, group-by city"
- …
- The cube operator is the n-dimensional generalization of the **group-by** operator.

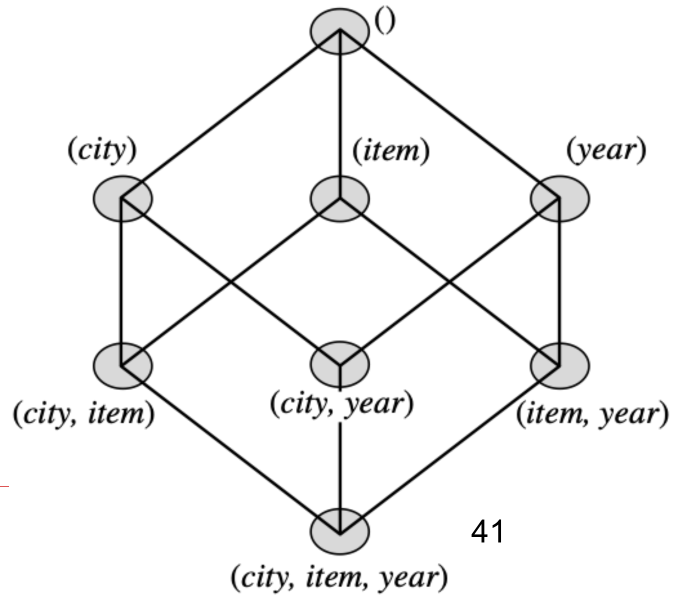
"Compute Cube" Operator

- Cube definition and computation in DMQL (Data Mining Query Language)

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

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

SELECT city, item, year, SUM (amount)
FROM SALES
CUBE BY city, item, year



Cube Materialization: Full Cube vs. Iceberg Cube

- Only a small portion of the cube's cells may have meaningful data (i.e., "above the water" in a sparse cube).
  - **Full Cube**: to compute all possible combinations of dimensions and measures in a data cube.
  - **Iceberg Cube**: to compute only the cells that meet a specified iceberg condition, such as a minimum threshold for a measure.

compute cube *sales_iceberg* **as**
select *month*, *city*, *customer_group*, count(*)
from *salesInfo*
cube by *month*, *city*, *customer_group*
having count(*) >= *min_support*

iceberg condition: Filters out low-value cells, retaining only significant data.

Efficient Processing of OLAP Queries

- Steps to Efficiently Process an OLAP Query
  - Identify the dimensions and filters
    - e.g., on {brand, province} with "year = 2004"
  - Convert OLAP operations into SQL-like operations:
    - e.g., dice = selection (year = 2004) + projection (brand, province)
  - Select the best materialized cuboids
    - Compare the query with available cuboids
    - Use cost-based estimation to choose the most efficient one

Efficient Processing of OLAP Queries

- Query on {brand, province} with "year = 2004"
  - Require dimensions of "brand" and "province", and a filter on "year"
  - 4 materialized cuboids are available.
    - Cuboid 1: {year, item_name, city}
    - Cuboid 2: {year, brand, country}
    - Cuboid 3: {year, brand, province}
    - Cuboid 4: {item_name, province} where year = 2004
  - **Which should be selected to process the query efficiently?**

Hierarchy
- item_name < brand < type
- city < province < country

- Not C2: Finer-granularity data CANNOT be generated from coarser-granularity data.
- Not C1: Cost the most as both item name and city are at a lower level
- C3 or C4? **Cost-based estimation** helps determine which is better.

Indexing OLAP Data: Bitmap Index

- Index on a particular column – bit-op is fast
  - n bits are needed for the attributes with n values.
    - If a record on the attribute has a value v in the base table, then the bit representing v is set to 1 in the corresponding row of the bitmap index.
    - All other bits for that row are set to 0.
  - *NOT suitable for high-cardinality domains*

Base table

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

Index on Region

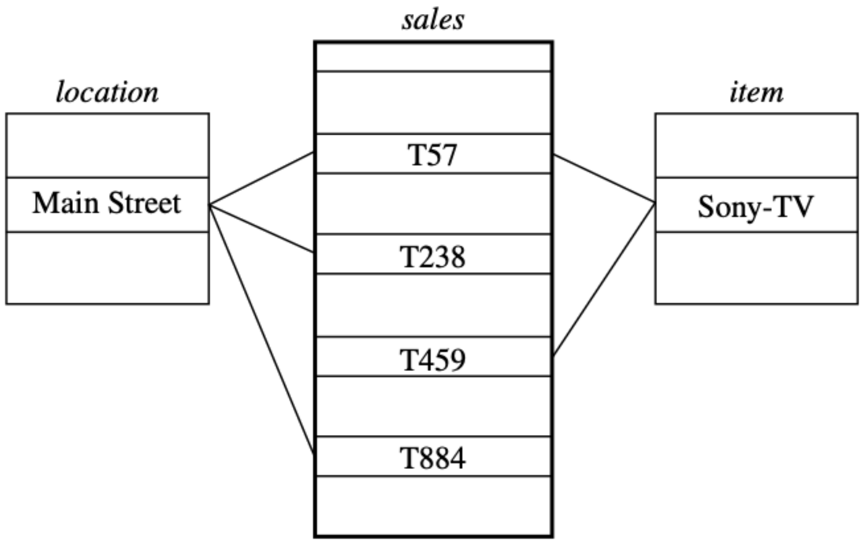
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

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 Index

- To register the joinable rows of relations from a relational DB
  - Maintain the joinable relationship
  - Speed up cross-table search in star schema
    - the fact table's foreign key & the dimension table's primary key
    - e.g., linking the **sales** fact table with two dimensions, **location** and **item**
  - Join index could span dimensions.
  - Identify sub-cubes that are of interest



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
. . .	. . .	. . .

Summary

- Data warehouse: subject-oriented, integrated, time-variant, and nonvolatile data collection organized for decision making
  - Distinguish and be maintained separately from operational databases
  - Architecture: data sources – bottom tier (RDBMS server) – middle tier (OLAP server) – top tier (client, end-user for query / reporting tools)
  - Multidimensional data model: star/snowflake/fact constellation schema
  - Data cube: fact (measures), dimensions; a lattice of cuboids (each with a different degree of summarization); concept hierarchies; indexing
  - OLAP operations: roll-up, drill-down, slice, dice, and pivot (rotate); bitmap index and join index for efficiency
  - Applications: information processing, analytical processing, data mining

Email: fengmei.jin@polyu.edu.hk

Office: PQ747

THANK YOU!