DATA WAREHOUSE AND OLAP TECHNOLOGY:

Data Warehouse:

A data warehouse is like a big library where an organization stores important information from different sources over time. It's separate from the database the organization uses to run its daily operations. This helps the organization to analyze and make decisions based on the stored data.

Data warehousing provides a way for business leaders to organize and use their data to make important decisions. It helps them to understand the past trends and patterns in their data which can be used to make future predictions and develop strategic plans.

Data warehouses are very useful for companies today as they provide a competitive advantage in a fast-changing world. Many organizations invest a lot of money in building enterprise-wide data warehouses to ensure they have easy access to their historical data for decision making.

Data Warehouse subject-oriented:

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
  - When data is moved to the warehouse, it is converted.

<u>Data Warehouse time-variant:</u>

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".

<u>Data Warehouse non-volatile:</u>

- 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.

<u>Why a Separate Data Warehouse?</u>

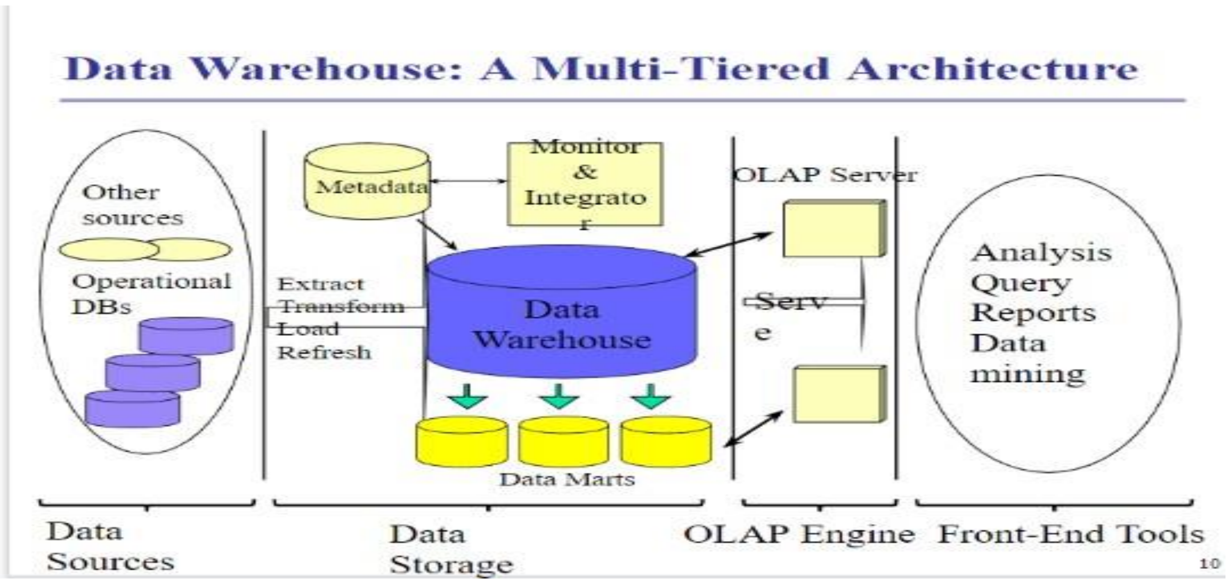
- High performance for both systems
    - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
    - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
    - <u>missing data</u>: Decision support requires historical data which operational DBs do not typically maintain
    - <u>data consolidation</u>: 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

Table 3.1 Comparison between OLTP and OLAP systems.

Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements, decision support
DB design	ER based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

NOTE: Table is partially based on [CD97].



<u>Three Data Warehouse Models:</u>

- 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

<u>Extraction, Transformation, and Loading (ETL):</u>

- **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

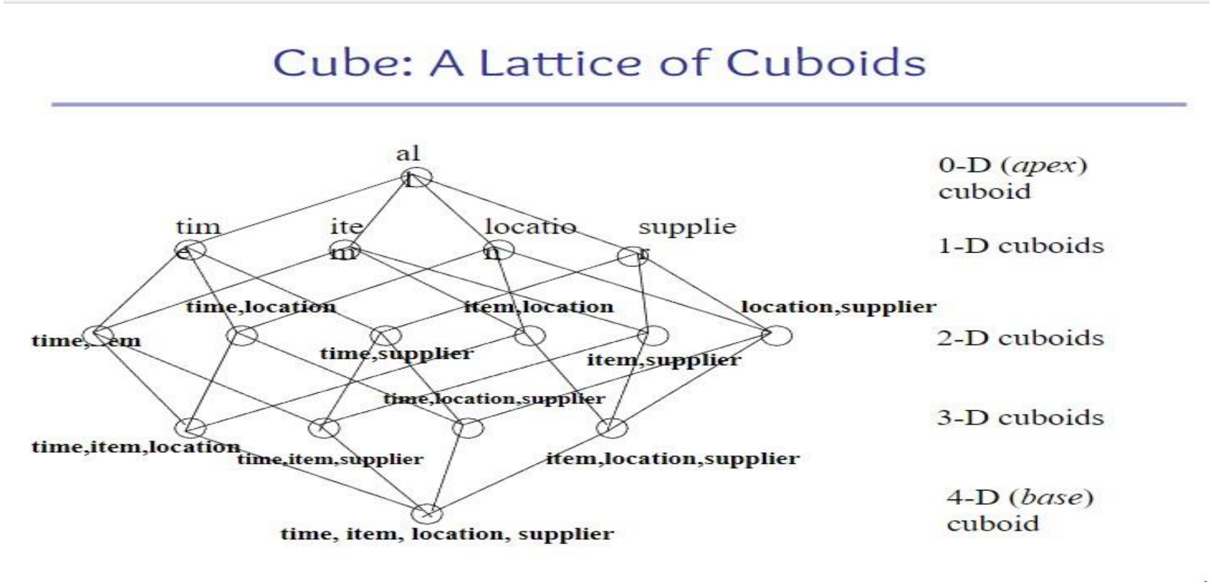
Metadata Repository:

- **Meta data** is the data defining warehouse objects. It stores:
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, derived data 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 data warehouse
- Data related to system performance
  - warehouse schema, view and derived data definitions
- Business data
  - business terms and definitions, ownership of data, charging policies

From Tables and Spreadsheets to Data Cubes:

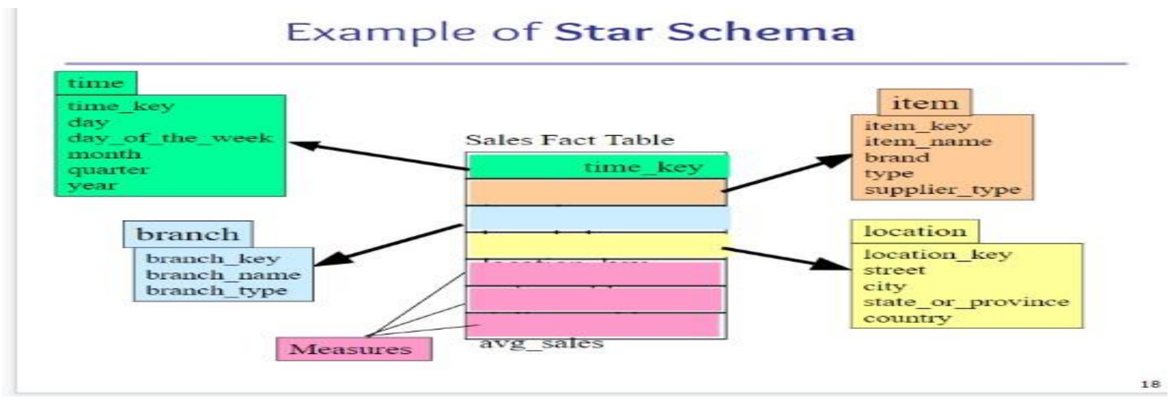
- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - **Dimension tables**, such as item (item_name, brand, type), or time(day, week, month, quarter, year)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

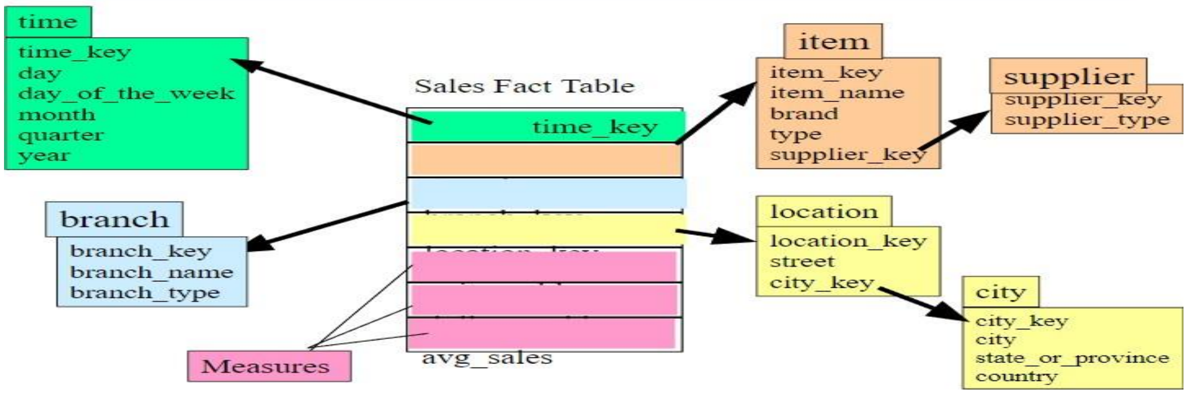


Conceptual Modeling of Data Warehouses:

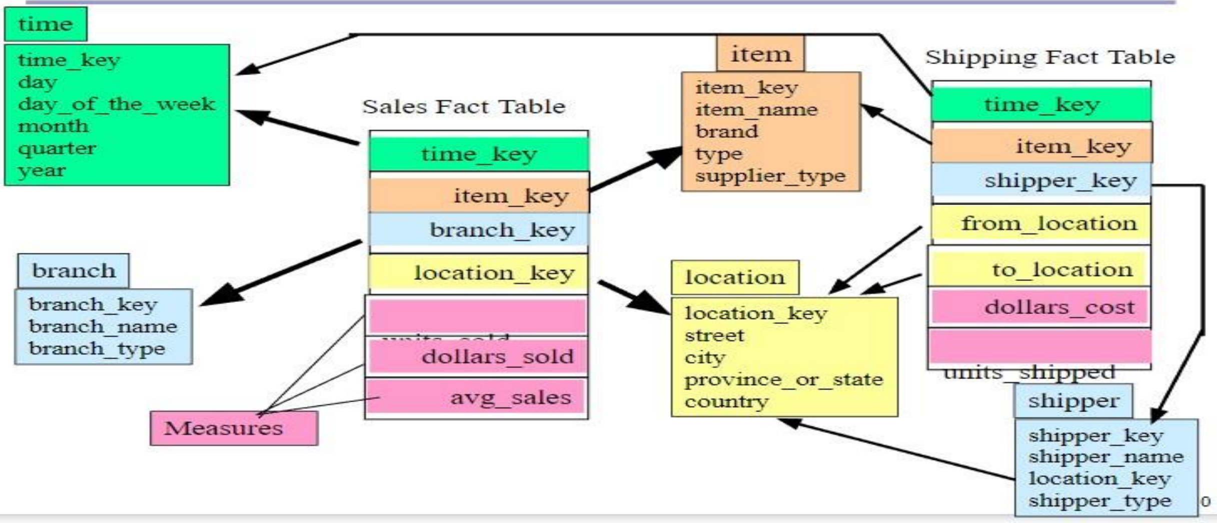
- Modeling data warehouses: dimensions & measures
    - <u>Star schema</u>: A fact table in the middle connected to a set of dimension tables
    - <u>Snowflake schema</u>: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape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



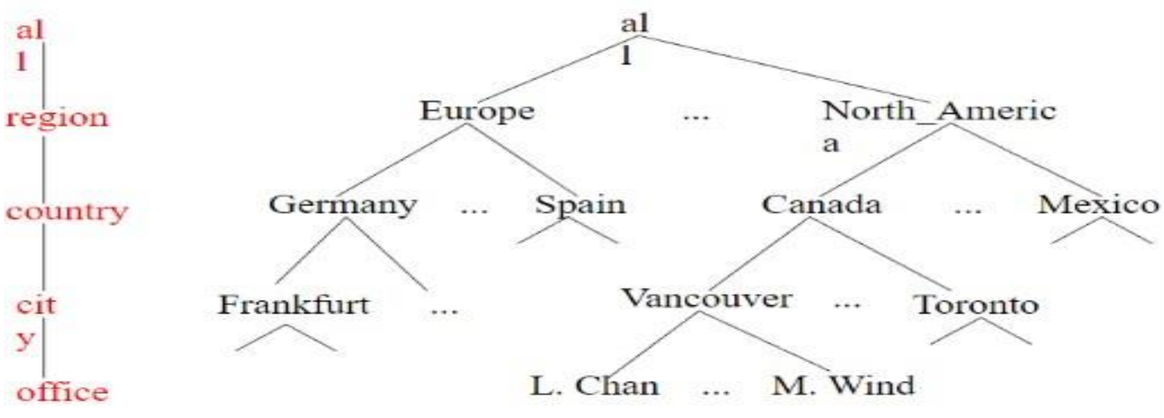
Example of **Snowflake Schema**



Example of **Fact Constellation**



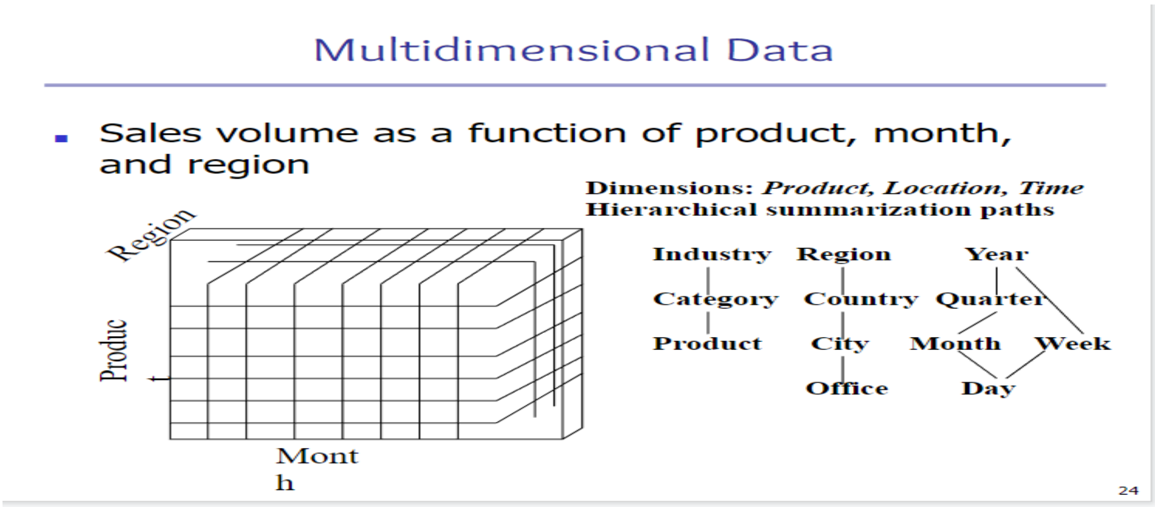
A Concept Hierarchy: **Dimension** (location)



<u>Data Cube Measures: Three Categories:</u>

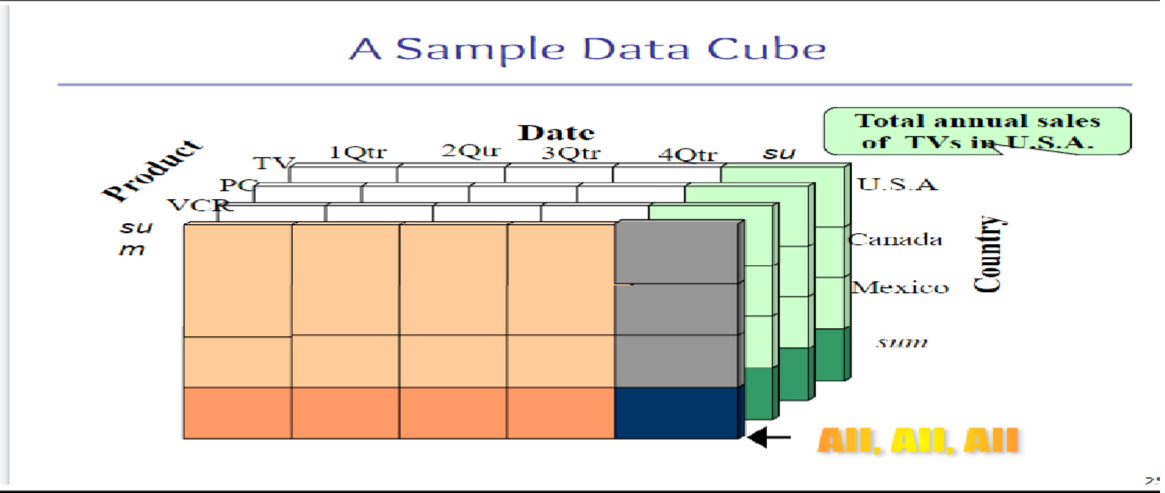
- <u>Distributive</u>: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
    - E.g., count(), sum(), min(), max()
- <u>Algebraic</u>: if it can be computed by an algebraic function with *M* 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()
- <u>Holistic</u>: if there is no constant bound on the storage size needed to describe a subaggregate.
    - E.g., median(), mode(), rank()

<u>View of Warehouses and Hierarchies:</u>



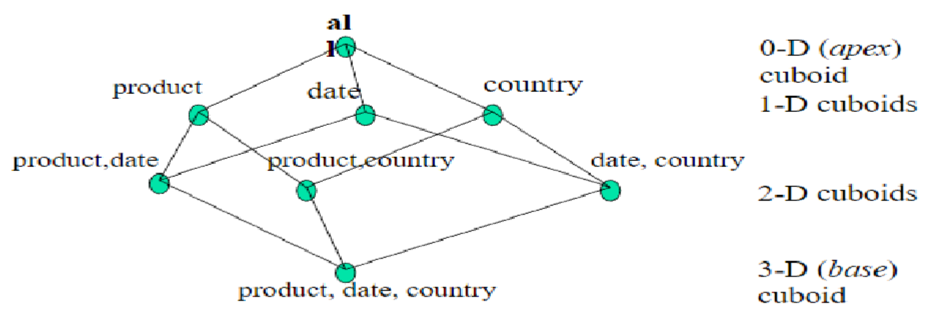
Multidimensional Data

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



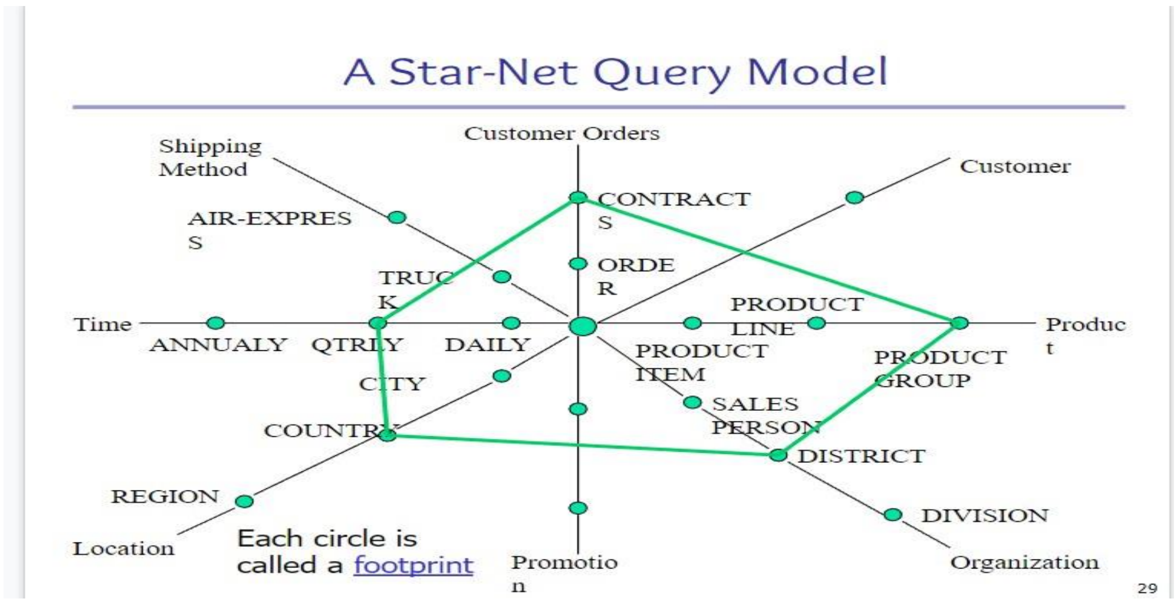
A Sample Data Cube

Cuboids Corresponding to the Cube



Typical OLAP Operations:

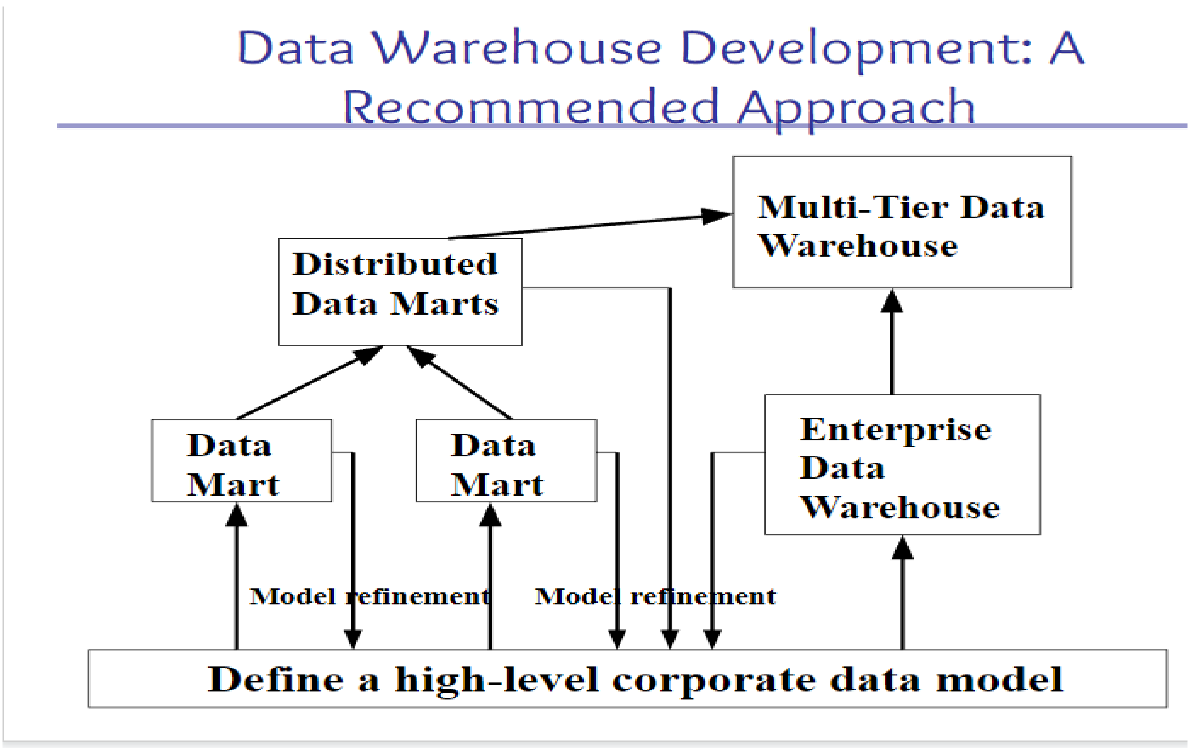
- Roll up (drill-up): summarize data
  - by climbing up hierarchy or by dimension reduction
- Drill down (roll down): reverse of roll-up
  - from higher level summary to lower level summary or detailed data, or introducing new dimensions
- Slice and dice: project and select
- Pivot (rotate):
  - reorient 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)



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
- **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
- **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 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:

- 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

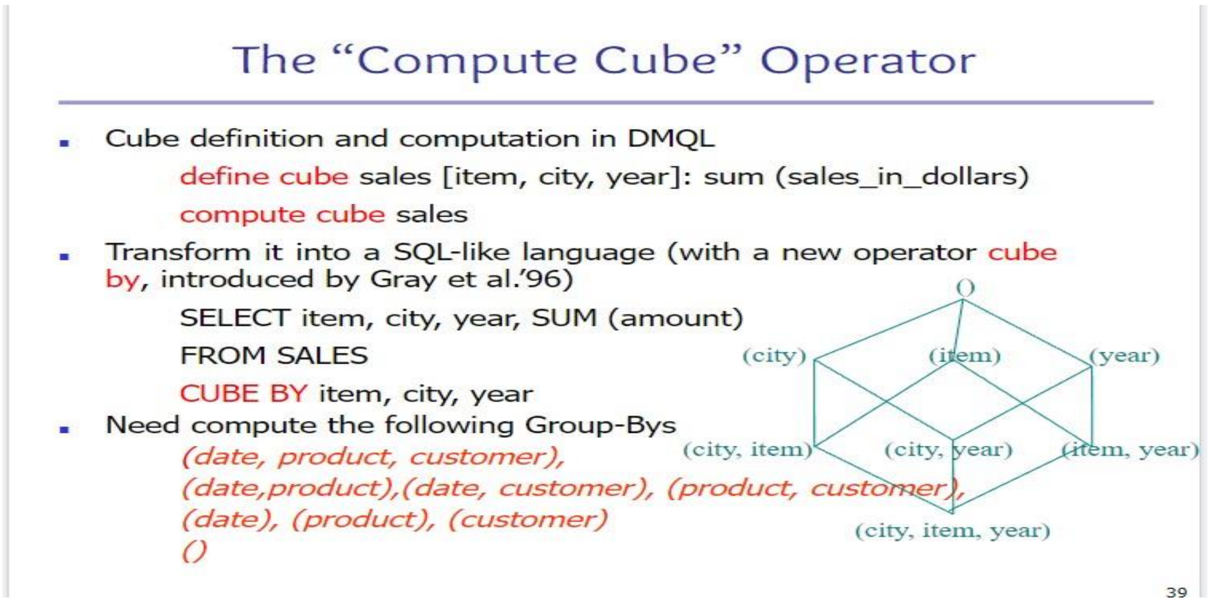
From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM):

- 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

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?
- 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.
- The “Compute Cube” Operator



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

- Need compute the following Group-Bys

  (date, product, customer),

  (date,product),(date, customer), (product, customer),

  (date), (product), (customer)

  ()

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]

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 <u>dimensions</u> of a start schema to <u>rows</u> 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

<u>**Efficient Processing OLAP Queries:**</u>

- 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

<u>**OLAP Server Architectures:**</u>

- <u>Relational OLAP (ROLAP)</u>
    - 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

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 University database

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

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