UNIT III

INTRODUCTION TO DATA WAREHOUSING AND MULTI-DIMENSIONAL MODELING

3.1 Operational Vs Decisional Support System ,The Need for Data Warehousing

3.2 Data Warehouse Definition, Features , The Information Flow Mechanism, Architecture , Role of Metadata, Classification of Metadata

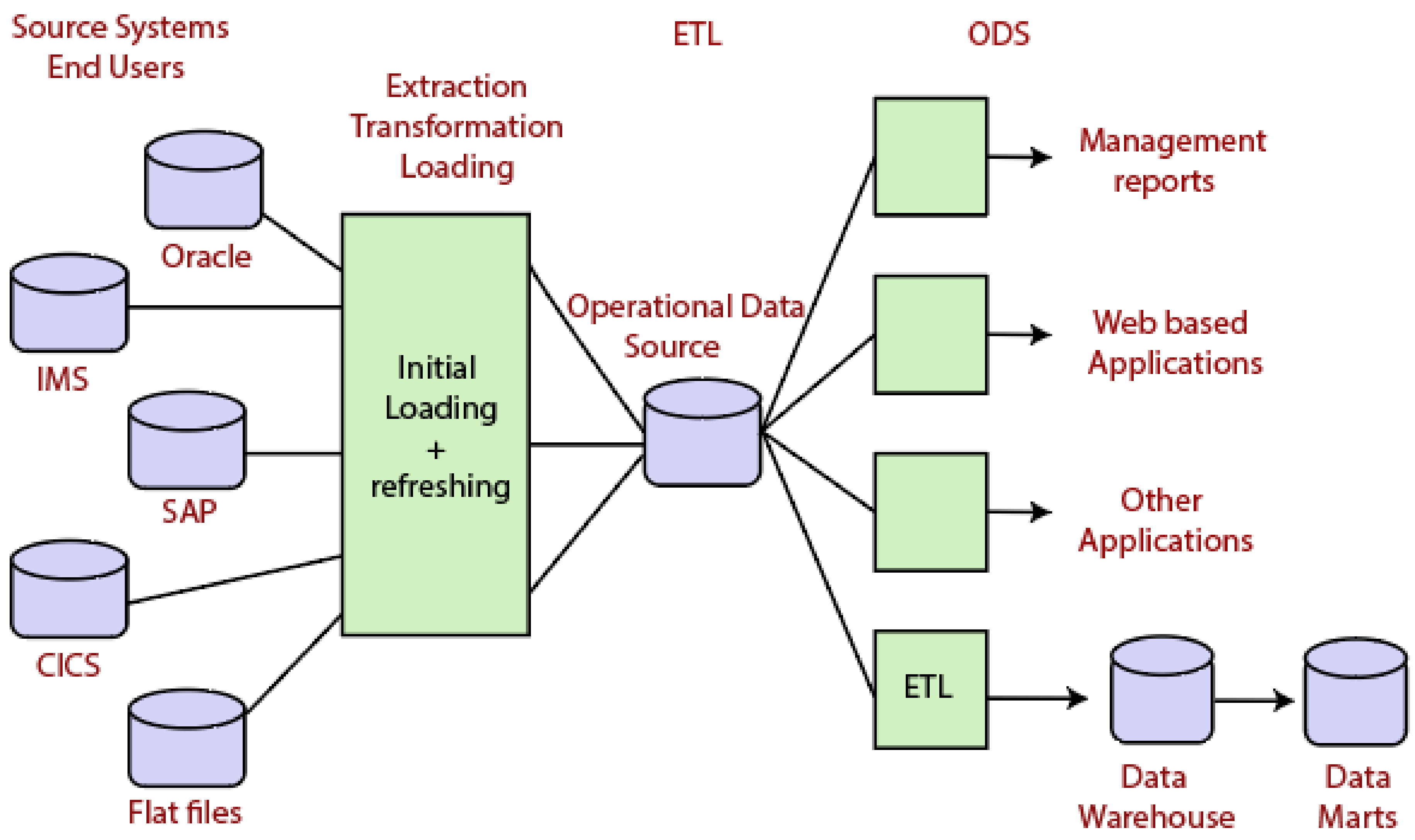
3.3 Data Warehouse Vs Data Marts, Data Warehousing Design Strategies, Data Warehouse Modeling Vs O perational Database Modeling

3.5 Updates To Dimension Tables - Slowly Changing Dimensions, Type 1, Type 2 and Type 3 Changes, Large Dimensions , Rapidly Changing Dimensions, Junk Dimensions, Aggregate Fact Tables

3.6 Data lake, Architecture of Data lake, Data Warehouse Vs Data lake

Operational Vs Decisional Support System

Operational Support systems	**Decisional Support Systems**
Data represented by operational support systems are transactions that happen in real time.	Decisional support systems use the operational data at a particular point in time.
Operational data is regarded as update transactions.	Decisional support data is regarded as query transaction. Which is read-only.
The concurrent transaction volume in operational data is very high.	The concurrent transaction volume is found at low or medium levels.
Operational data usually consists of information about transactions and is stored in numerous tables.	Decision support data is usually stored in less number of tables that store data derived from the operational data.



Operational Data Store Structure

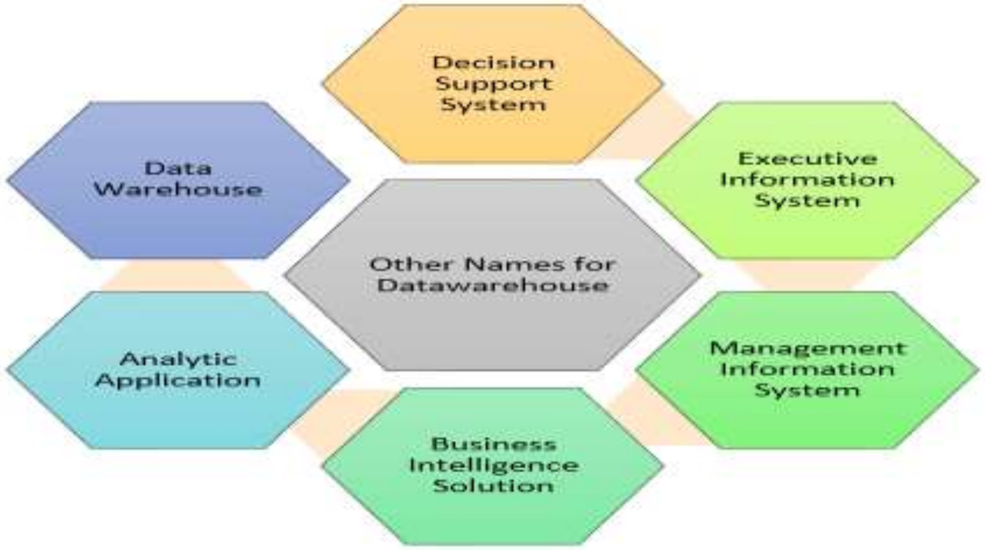
What is a Data Warehouse?

- 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 Definition

A Data Warehousing (DW) is process for collecting and managing data from varied sources to provide meaningful business insights.

A Data warehouse is typically used to connect and analyze business data from heterogeneous sources. The data warehouse is the core of the BI system which is built for data analysis and reporting.


Decision Support System
Data Warehouse
Executive Information System
Other Names for Datawarehouse
Analytic Application
Management Information System
Business Intelligence Solution

The Need for Data Warehousing

- It is difficult to retrieve data from tables for analytical purposes.
- Data warehousing is the best way to integrate valuable data from different sources into the database of a particular application.
- Data warehouses make it easy to develop and store metadata
- Business experts or users become habitual to see many customized data on display screens fields such as rolled-up general ledger balances. These fields do not exist in the database.
- When we perform reporting and analysis functions on the hardware that handles transactions, the performance is often poor. Therefore, data warehouse should be used for reporting and analysis.

Features of Data Warehouse

A data warehouse is a combination of data from enterprise-wide sources. A data warehouse consists of four main features. These features are:

- Subject Oriented
- Integrated
- Non-volatile
- Time variant

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*

Information Flow Mechanism

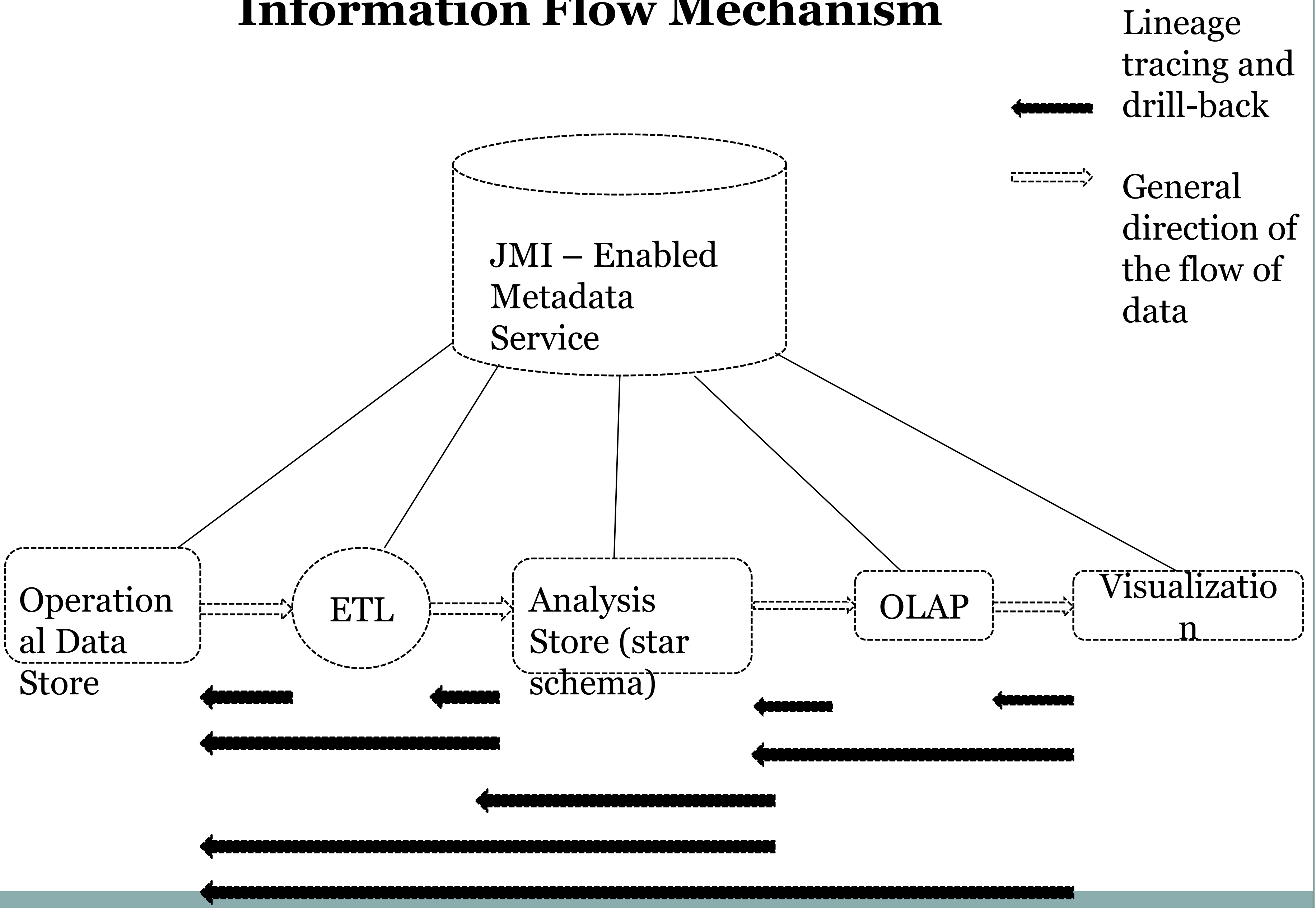

Lineage tracing and drill-back
General direction of the flow of data
JMI – Enabled Metadata Service
Operational Data Store
ETL
Analysis Store (star schema)
OLAP
Visualization

Fig. Data Warehouse Information Flow

- Information supply chain is fully-integrated
- Integration via centralized shared metadata and metadata – driven tools
- Metadata communication via JMI programmatic API and bulk interchange (XML)
- MOF/JMI reflection used to reconcile metadata integration between tools based on dissimilar metamodels.

OLTP vs. OLAP

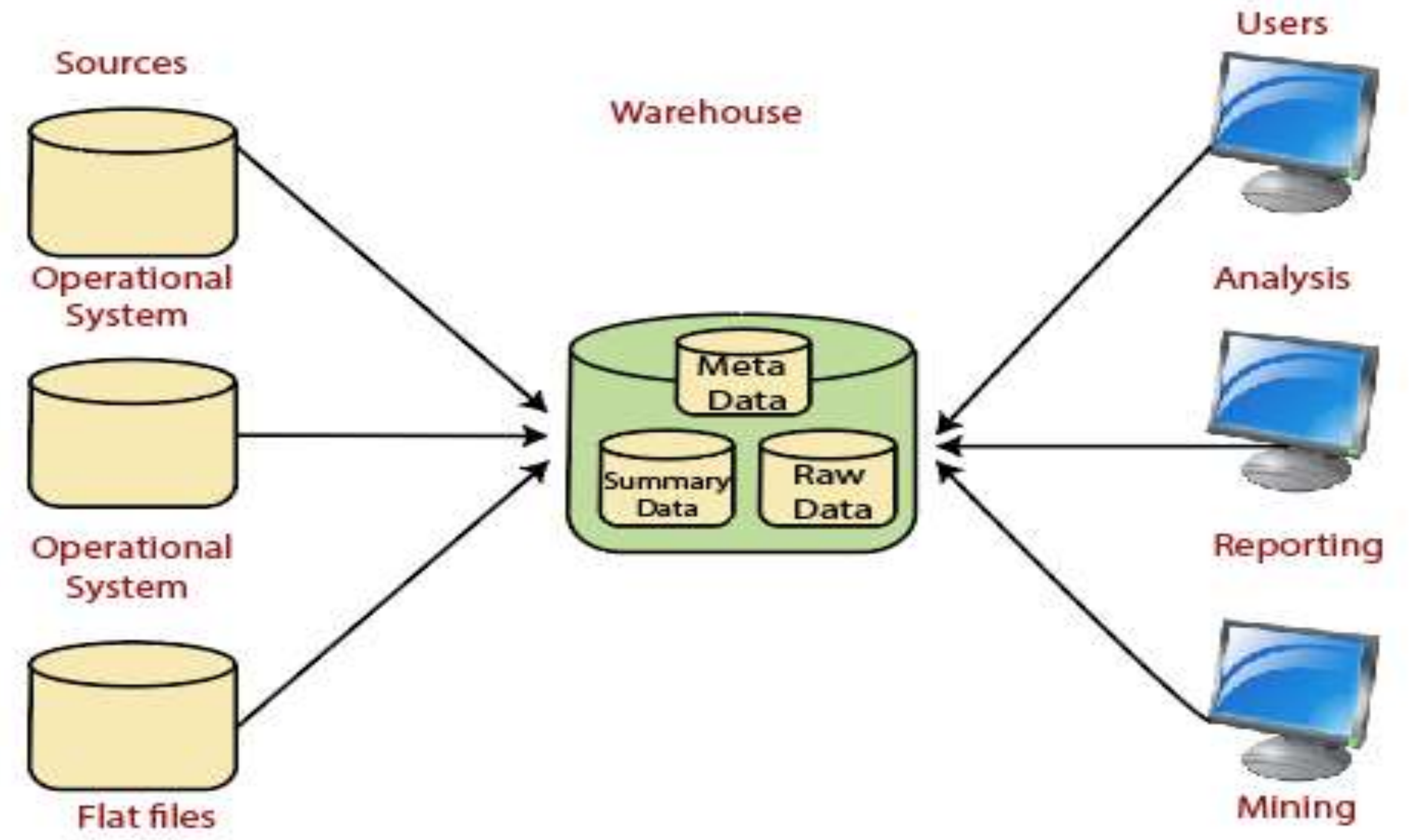
	OLTP	**OLAP**
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
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

Data warehouses and their architectures very depending upon the elements of an organization's situation.

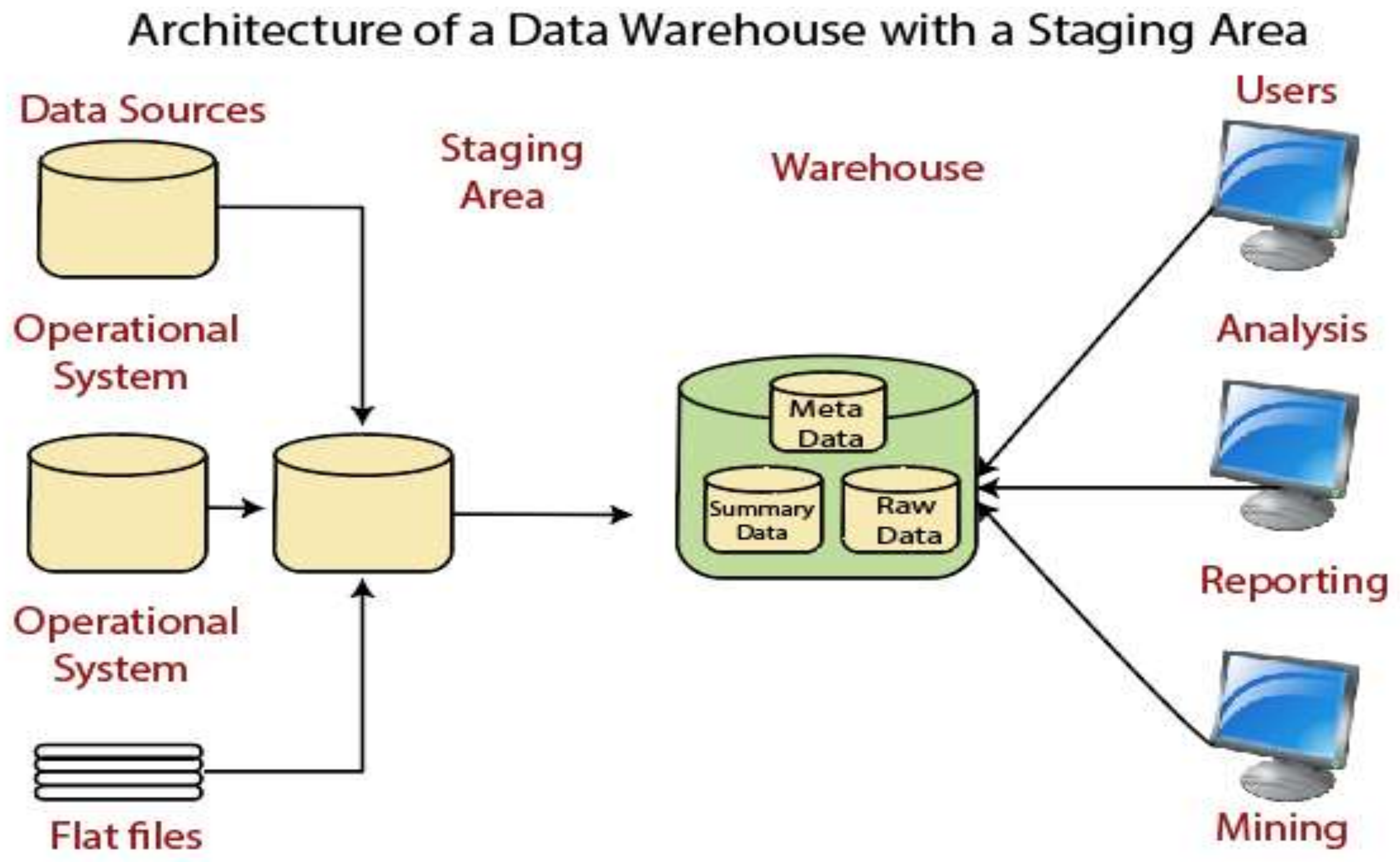
Three common architectures are:

- Data Warehouse Architecture: Basic
- Data Warehouse Architecture: With Staging Area
- Data Warehouse Architecture: With Staging Area and Data Marts

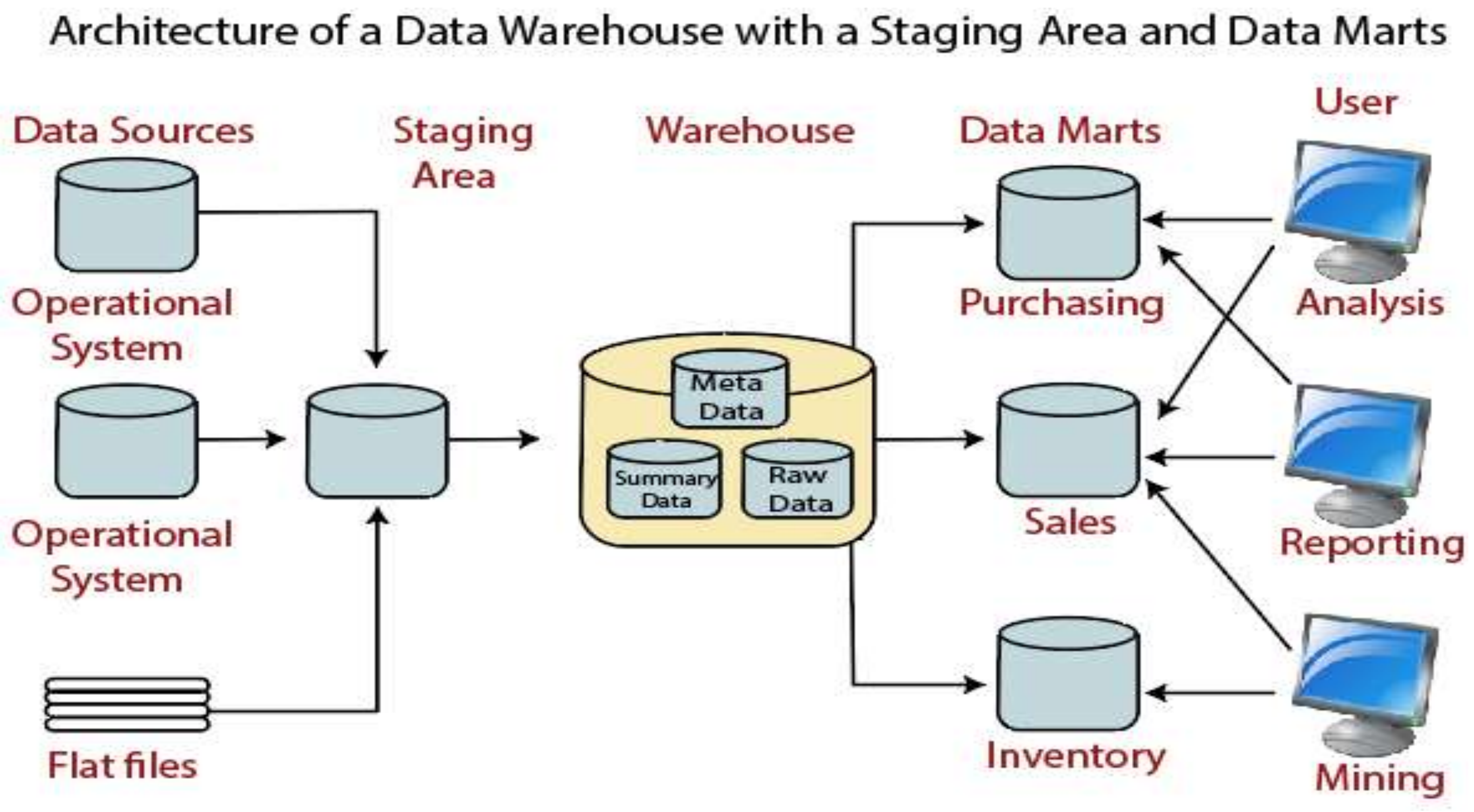
Architecture of a Data Warehouse



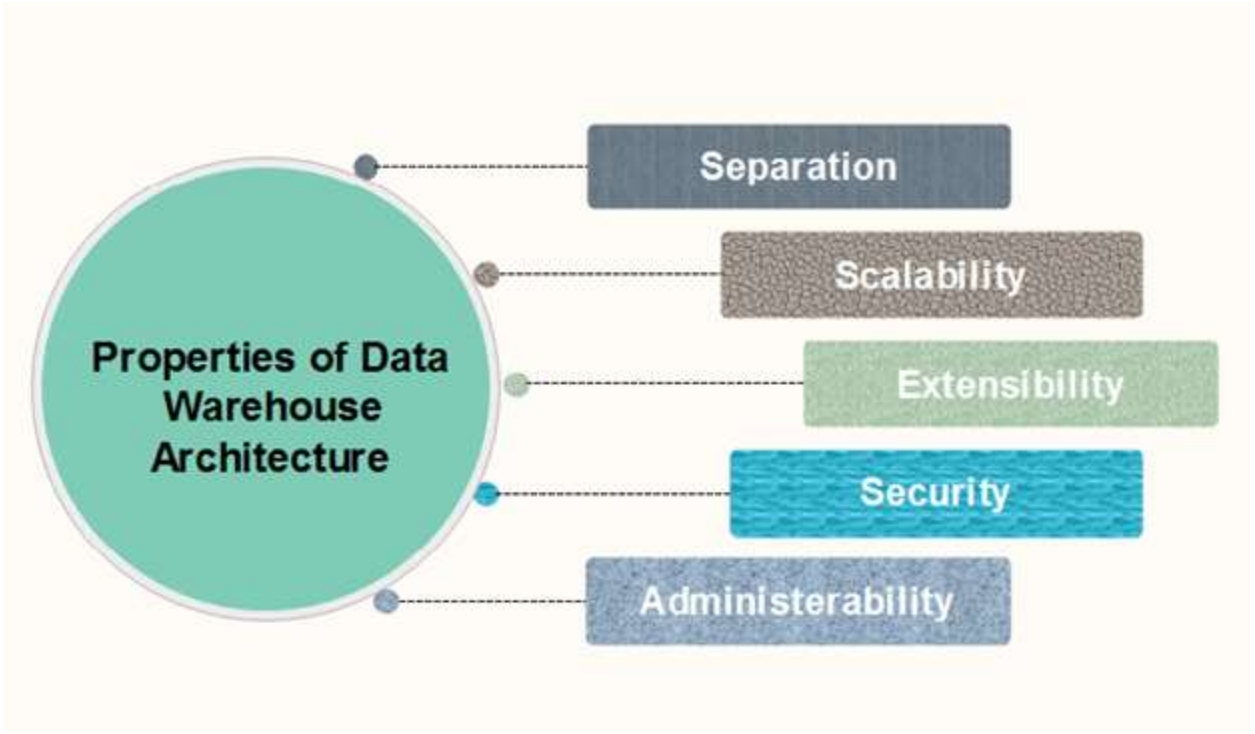
Data Warehouse Architecture: With Staging Area



Data Warehouse Architecture: With Staging Area and Data Marts



Properties of Data Warehouse Architectures



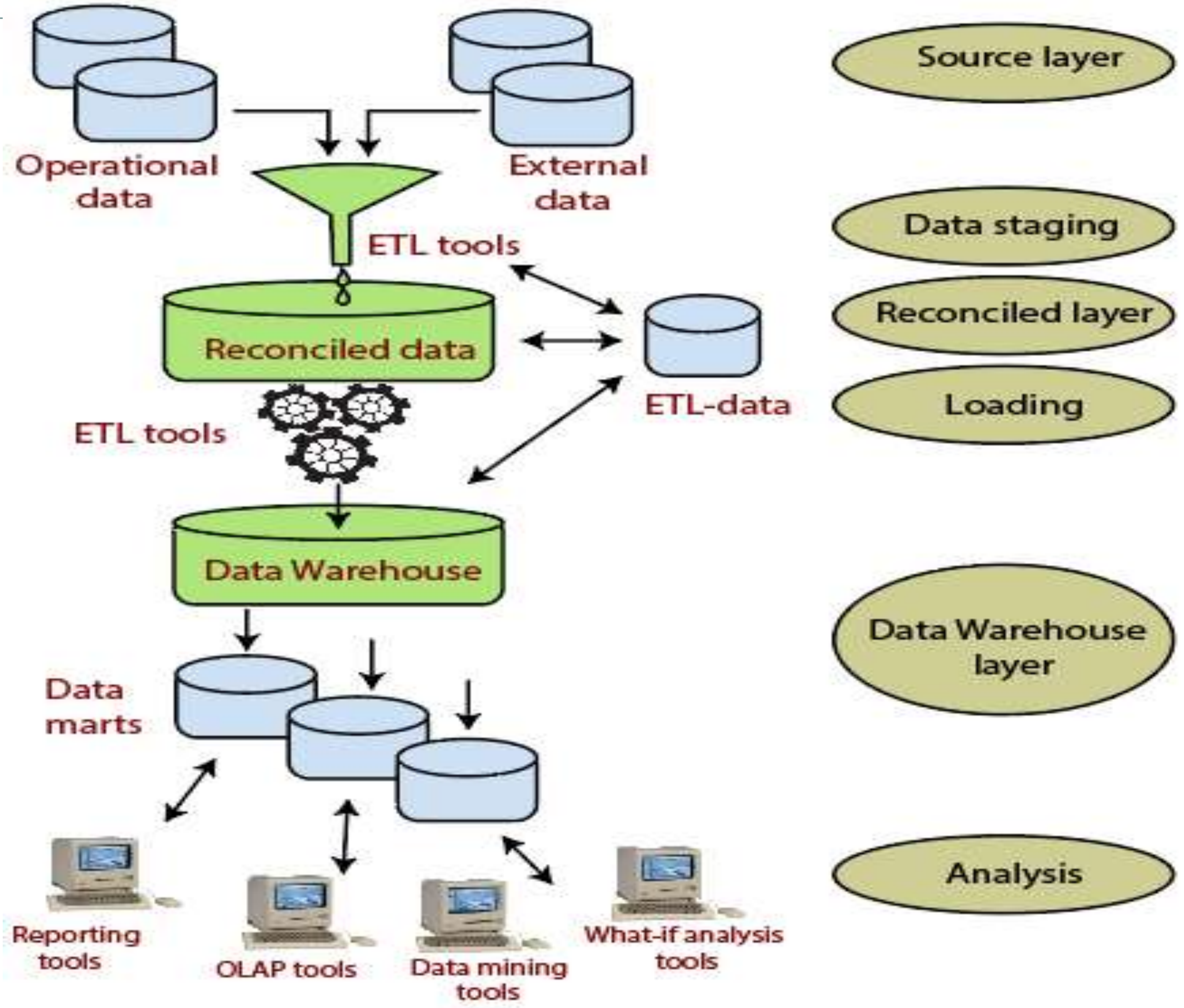
1. Separation: Analytical and transactional processing should be keep apart as much as possible.

2. Scalability: Hardware and software architectures should be simple to upgrade the data volume, which has to be managed and processed, and the number of user's requirements, which have to be met, progressively increase.

3. Extensibility: The architecture should be able to perform new operations and technologies without redesigning the whole system.

4. Security: Monitoring accesses are necessary because of the strategic data stored in the data warehouses.

5. Administer ability: Data Warehouse management should not be complicated.



Three-Tier Architecture for a data warehouse system

Data extraction commonly happens in one of the three following ways.

Update notification

In update notification, the source system notifies you when a data record changes. You can then run the extraction process for that change. Most databases and web applications provide update mechanisms to support this data integration method.

Incremental extraction

Some data sources can't provide update notifications but can identify and extract data that has been modified over a given time period. In this case, the system checks for changes at periodic intervals, such as once a week, once a month, or at the end of a campaign. You only need to extract data that has changed.

Full extraction

Some systems can't identify data changes or give notifications, so reloading all data is the only option. This extraction method requires you to keep a copy of the last extract to check which records are new. Because this approach involves high data transfer volumes, we recommend you use it only for small tables.

Basic data transformation
Basic transformations improve data quality by removing errors, emptying data fields, or simplifying data.
Examples of these transformations follow.

- Data cleansing
- Data deduplication
- Data format revision
- Advanced data transformation
- Derivation
- Joining
- Splitting
- Summarization
- Encryption

Full load
In full load, the entire data from the source is transformed and moved to the data warehouse. The full load usually takes place the first time you load data from a source system into the data warehouse.

Incremental load
In incremental load, the ETL tool loads the delta (or difference) between target and source systems at regular intervals. It stores the last extract date so that only records added after this date are loaded. There are two ways to implement incremental load.

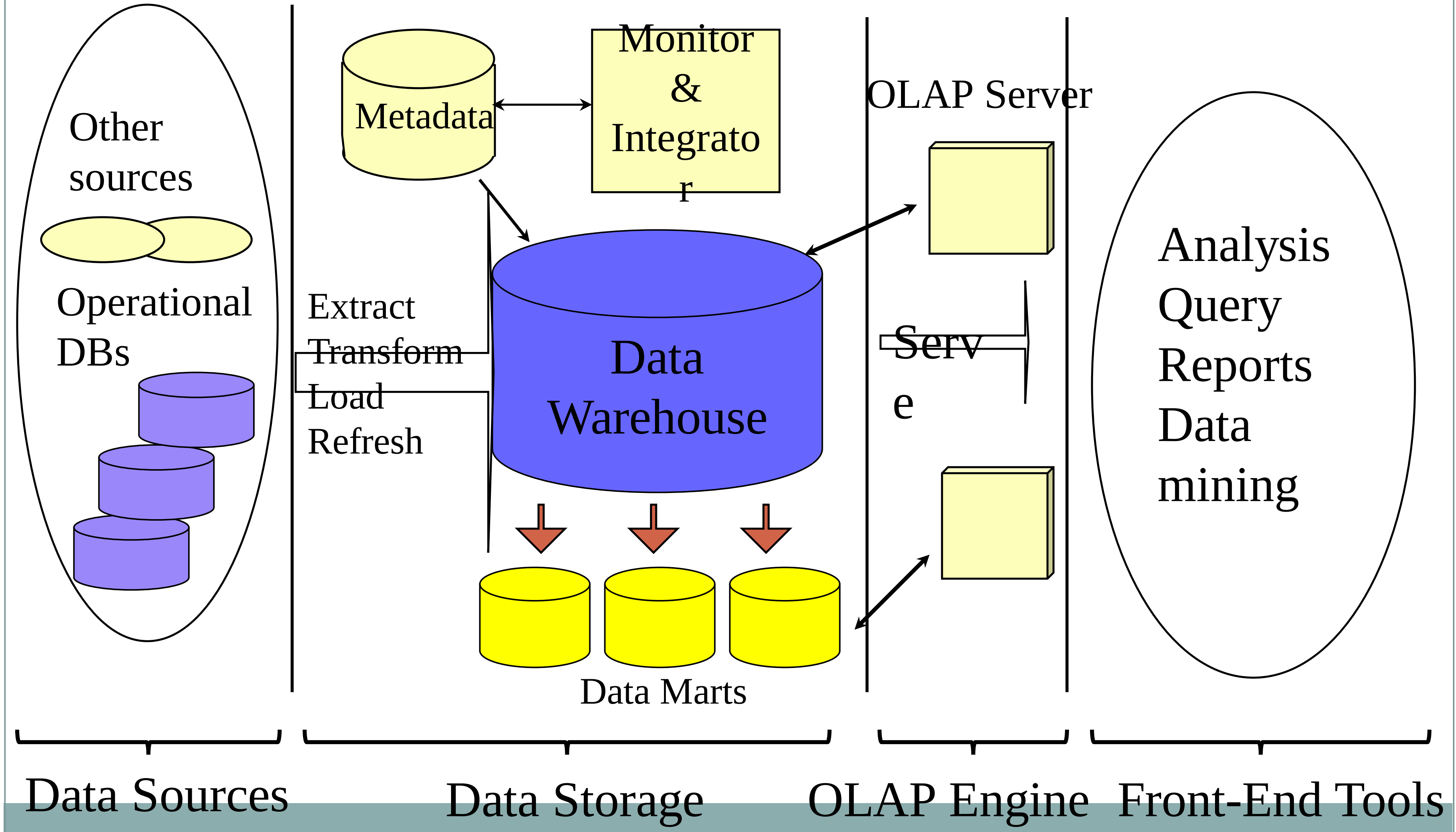
Streaming incremental load
If you have small data volumes, you can stream continual changes over data pipelines to the target data warehouse. When the speed of data increases to millions of events per second, you can use event stream processing to monitor and process the data streams to make more-timely decisions.

Batch incremental load
If you have large data volumes, you can collect load data changes into batches periodically. During this set period of time, no actions can happen to either the source or target system as data is synchronized.

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

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 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

Classification of Metadata

- Determine if dissimilar kinds are grouped together. While grouping, subject or subgroup does not inherit all the features of the superset. Which signifies that the knowledge and requirements about the superset are not relevant for the members of the subset.

- Determine whether the classes have overlaps.

- Whether subordinates (may) have several superordinate's or not. Multiple supertypes for one subtype signify that the subordinate contains the features of all its superordinate's.

- Evaluate whether the standards for belonging to a class or group are well defined.

- Whether or not the types of relations between the concepts are made clear and well defined.

- Whether or not the subtype-supertype relations are differentiated from composition relations and from object-role relations.

Data warehouse and Data Marts

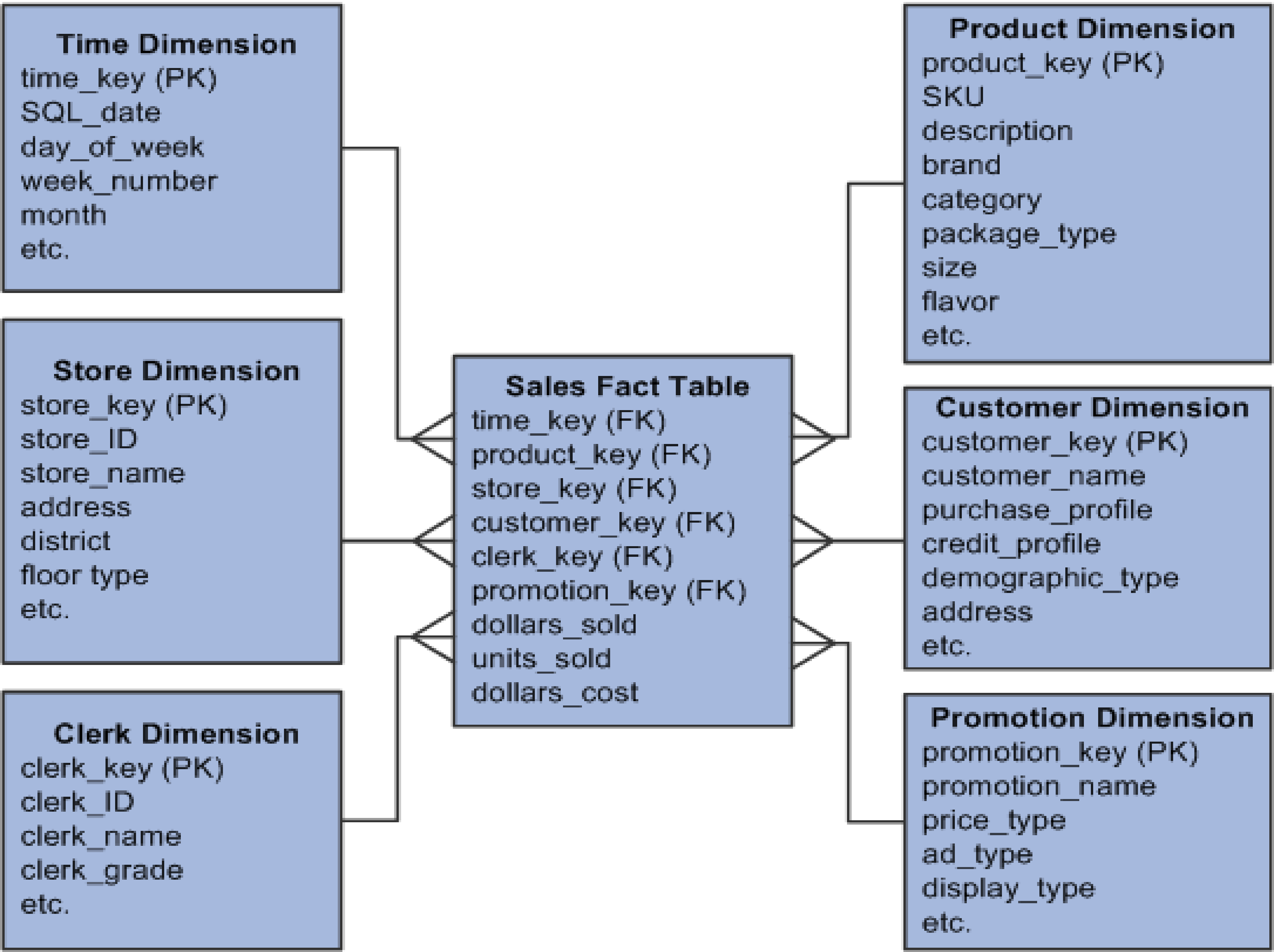
Data warehouse	Data Marts
It has a corporate/ enterprise-wide scope.	Its scope is departmental that is specific to one department.
It is a union of all data marts	It is a single business process.
Data is received from the staging area.	Data is received from star-join (facts and dimensions).
It queries on the presentation resources.	It is technology optimal for data access and analysis.
It has structure for corporate view of data.	It has structure to suit the departmental view of data.

Chapter 4: Data Warehousing and On-line Analytical Processing

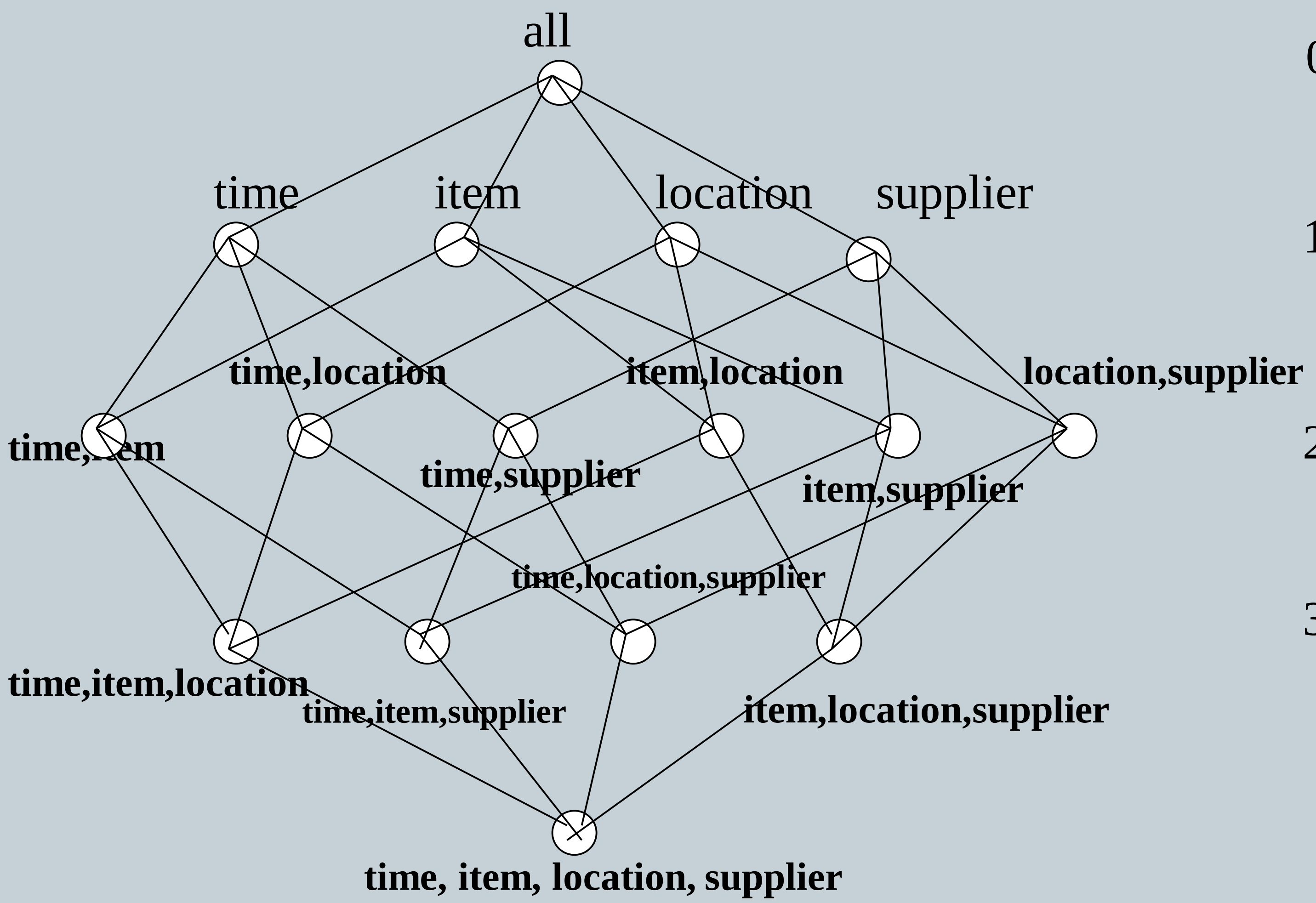
3.4 The Star Schema - Fact Tables and Dimension Tables, The Factless Fact Table, Keys in the Data Warehouse Schema- Primary Keys, Surrogate Keys & Foreign Keys, The Snowflake Schema, Fact Constellation Schema(Family of Stars)

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.


Time Dimension
time_key (PK)
SQL_date
day_of_week
week_number
month
etc.
Store Dimension
store_key (PK)
store_ID
store_name
address
district
floor type
etc.
Clerk Dimension
clerk_key (PK)
clerk_ID
clerk_name
clerk_grade
etc.
Sales Fact Table
time_key (FK)
product_key (FK)
store_key (FK)
customer_key (FK)
clerk_key (FK)
promotion_key (FK)
dollars_sold
units_sold
dollars_cost
Product Dimension
product_key (PK)
SKU
description
brand
category
package_type
size
flavor
etc.
Customer Dimension
customer_key (PK)
customer_name
purchase_profile
credit_profile
demographic_type
address
etc.
Promotion Dimension
promotion_key (PK)
promotion_name
price_type
ad_type
display_type
etc.

Cube: A Lattice of Cuboids



0-D (*apex*) cuboid

1-D cuboids

2-D cuboids

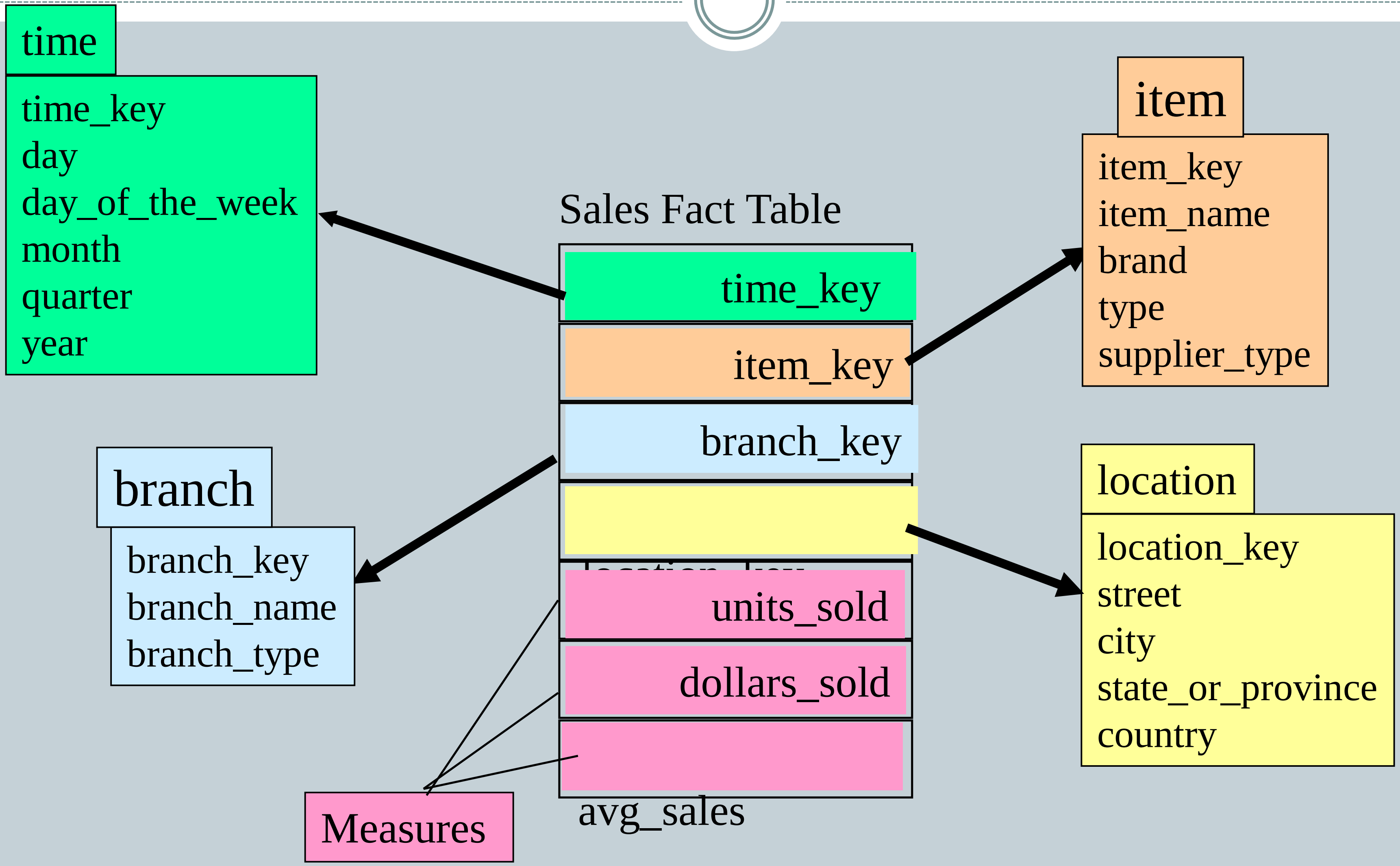
3-D cuboids

4-D (*base*) cuboid

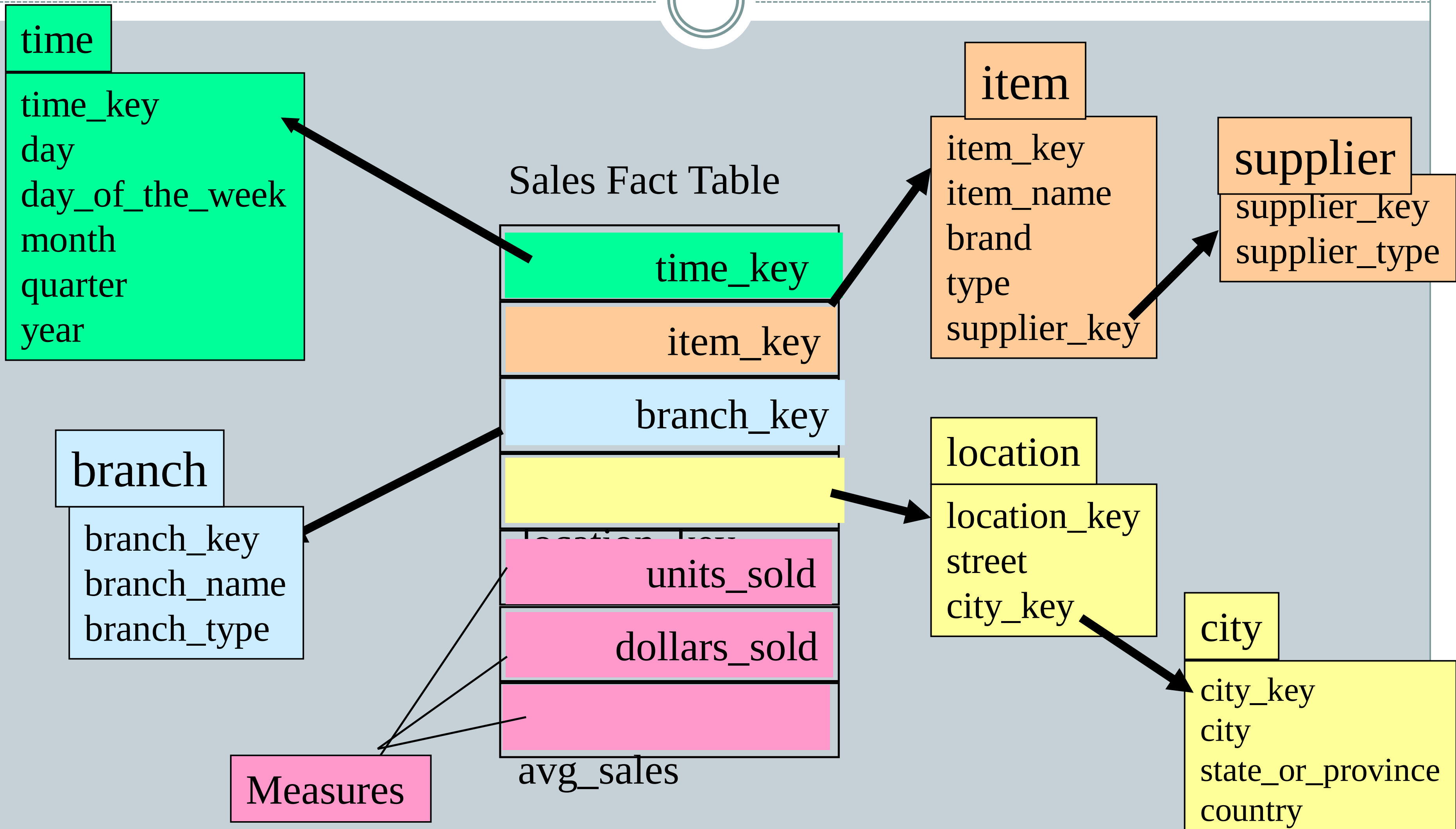
Conceptual Modeling of Data Warehouses

- 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

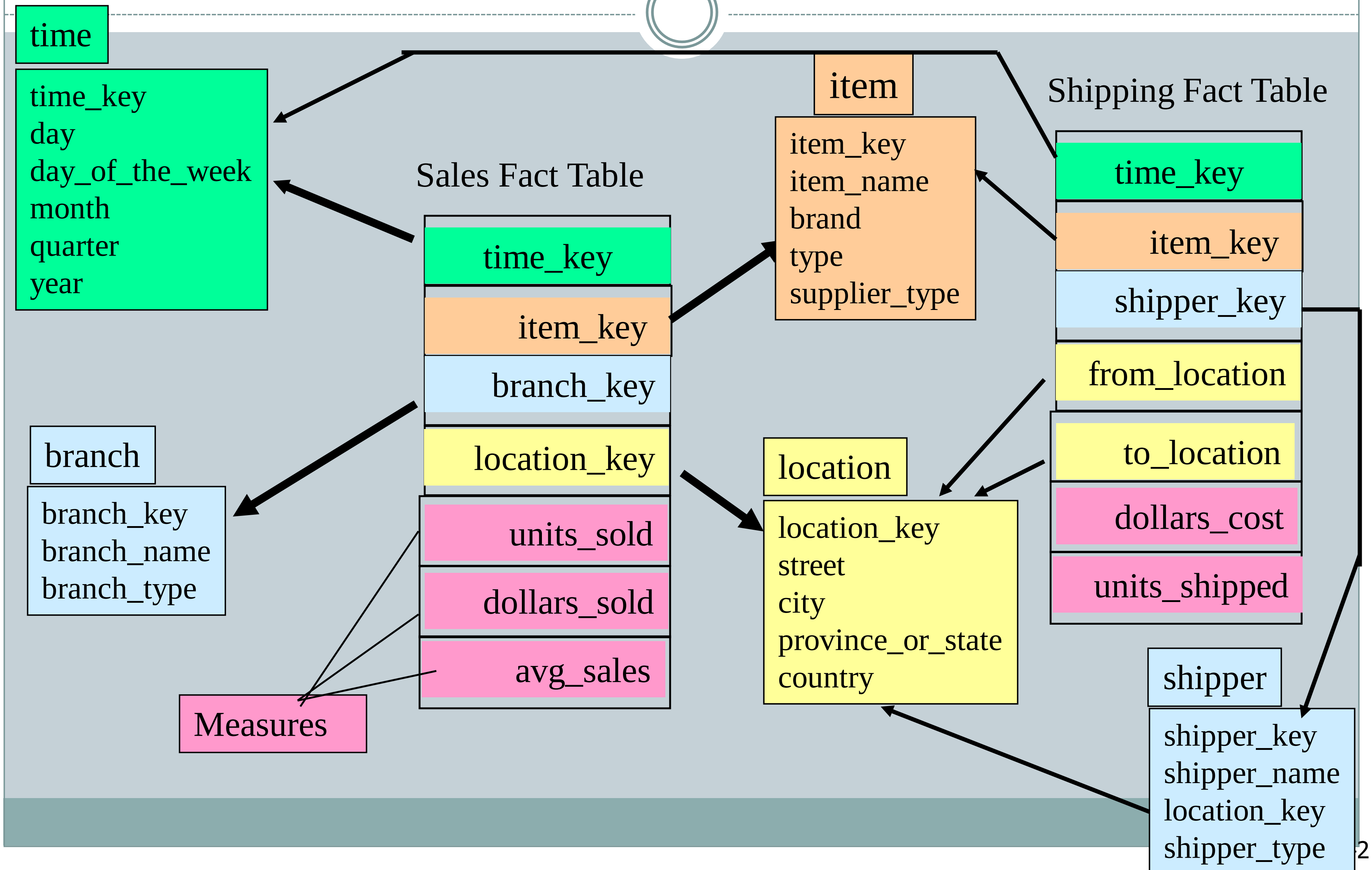
Example of **Star Schema**



Example of **Snowflake Schema**



Example of **Fact Constellation**



Slowly Changing Dimensions

What is a Slowly Changing Dimension?

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered and implemented as one of the most critical ETL tasks in tracking the history of dimension records.

There are three types of SCDs and you can use Warehouse Builder to define, deploy, and load all three types of SCDs.

What are the three types of SCDs?

The three types of SCDs are:

Type 1 SCDs - Overwriting

In a Type 1 SCD the new data overwrites the existing data. Thus the existing data is lost as it is not stored anywhere else. This is the default type of dimension you create. You do not need to specify any additional information to create a Type 1 SCD.

Type 2 SCDs - Creating another dimension record

A Type 2 SCD retains the full history of values. When the value of a chosen attribute changes, the current record is closed. A new record is created with the changed data values and this new record becomes the current record. Each record contains the effective time and expiration time to identify the time period between which the record was active.

Type 3 SCDs - Creating a current value field

A Type 3 SCD stores two versions of values for certain selected level attributes. Each record stores the previous value and the current value of the selected attribute. When the value of any of the selected attributes changes, the current value is stored as the old value and the new value becomes the current value

Type 1 : Overwrite the dimension records

Type 2 : Add a new dimension records

Type 3 : Create new fields in the dimension record

Type 1 changes: Correction of Errors

- These changes relate to correction of errors in source systems.
- These changes do not have any significance in the source systems.
- The old value needs to be discarded.
- New value overwrites the old value in the source system.
- Such changes in the source system need not be preserved in the data warehouse.

Type 1 changes are applied to the data stored in the data warehouse.

- Overwrite the value of the attribute with the new value in the dimension table now.
- The old value of the attribute is discarded , that is, not persevered.
- No other changes are made in the dimension table row.
- The key of this dimension table row is not affected.

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	July Michael	Single	AAAAAA A

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	July Michel	Single	AAAAAA A

Method for applying Type 1 change

Type 2 Changes: Preservation of History

- These changes relate to true changes in source system.
- The history must be preserved in the data warehouse.
- These changes cause the history to be partitioned in the data warehouse.
- Every change that occurs in the attribute value must be preserved.

How Type 2 changes applied to the data store in datawarehouse

- A new dimension table row with the new value of the changed attribute is added.
- A new column called the effective date is added in the dimension table.
- The original row in the dimension table is not changed.
- The key of the original row remains the same.
- The new row is inserted with anew surrogate key in the dimension table

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	Jenny david	Single	AAAAAAA
-	-	-	-	-

Customer key	Customer ID	Customer Name	Marital status	Address	Effective Date
-	-	-	-	-	-
22522134	C12345	Jenny david	Single	AAAAAAA	1/Mar/03
-	-	-	-	-	-
22522134	C12345	Jenny david	Married	AAAAAAA	16/Jan/06

Method for applying Type 2 change

Type 3 changes: Tentative Soft Revisions

How Type 3 changes applied to the data stored in the datawarehouse

- There is a need to track history with both old and new value of the same attribute.
- Type 3 changes are used to compare performance across the transition.
- They enable the users to track data in both forward and backward directions.
- An “old” field is added in the dimension table for the affected attribute.
- The existing value of the attribute is pushed down from the “current” field to the “old” field.
- The new value of the attribute is kept in the “current” field.

- A “current” effective data field is also added for the changed attribute.
- The key of the row is not affected.
- No new dimension row is added in the dimension table.
- The existing queries will automatically switch to the “current” value.
- Revision must be done to queries that need to use the “old” value.
- This technique works well with one soft change at a time and if there are a succession of changes, more sophisticated techniques must be devised.

Salesperson key	Salesperson ID	Salesperson Name	Old Location	Current Location	Effective date
-	-	-	-	-	
22522134	C12345	Jenny david		Delhi	1/Mar/03
-	-	-	-	-	

Salesperson key	Salesperson ID	Salesperson Name	Old Location	Current Location	Effective date
-	-	-	-	-	
22522134	C12345	Jenny david	Delhi	Mumbai	1/Mar/03
-	-	-	-	-	

Method for applying type 3 changes

A Product dimension table contains four attributes namely, Product ID which is the Primary key, launch year in which the product was brought into the market, name which specifies the name of the product and finally product price which tells the price of the product. In the year 2005 , the price of Product 1 was Rs. 350 which later on changed to Rs. 450. With this dimension construct slowly changing Dimensions.

Types of Dimensions

Dimension: A dimension table has two types of columns, primary keys and descriptive data. For example, **Time** and **Customer.**

Types of Dimensions

- Slowly Changing Dimensions
- Rapidly Changing Dimensions
- Junk Dimensions

Slowly Changing Dimensions

It depends on the business requirement, where any particular feature history of changes in the data warehouse is preserved. It is called a slowly changing feature, and a quality dimension is called a slowly changing dimension.

Rapidly Changing Dimensions

A dimension attribute change is a rapidly changing feature. If we do not need to track changes, rapid quality is not a problem. If you need to follow the changes, then using the standard slowly changing amplitude technique can cause massive amplitude size inflation. The solution moves the attribute to its dimension, with a different foreign key. The new dimension is called a rapidly changing size.

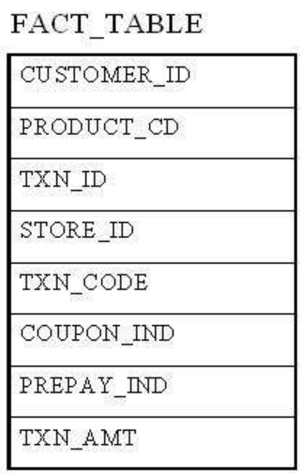
Junk Dimensions

A junk dimension fact table is a single table with the combination of **attributes** to avo

multiple foreign keys. Junk dimensions are created to manage **foreign dimensions**,

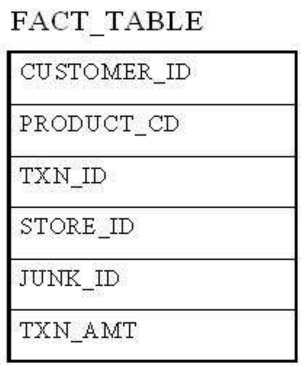
that are created by **rapidly changing dimensions**.

Assuming that we have the following fact table



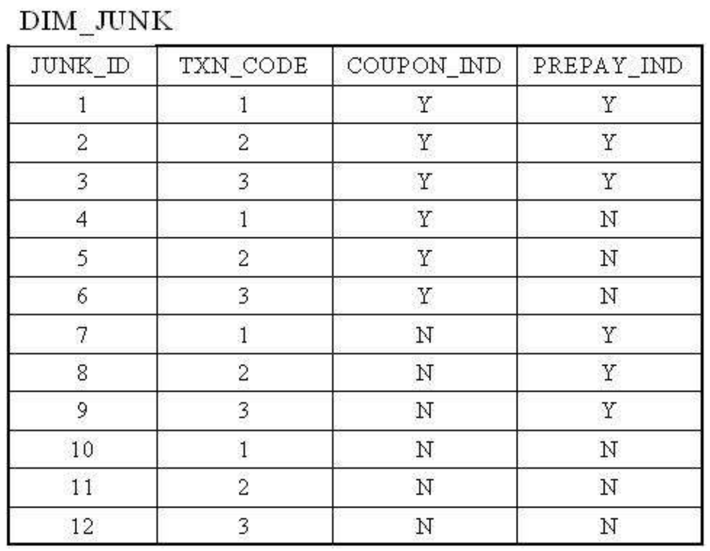
FACT_TABLE

CUSTOMER_ID
PRODUCT_CD
TXN_ID
STORE_ID
TXN_CODE
COUPON_IND
PREPAY_IND
TXN_AMT



FACT_TABLE

CUSTOMER_ID
PRODUCT_CD
TXN_ID
STORE_ID
JUNK_ID
TXN_AMT



DIM_JUNK

JUNK_ID	TXN_CODE	COUPON_IND	PREPAY_IND
1	1	Y	Y
2	2	Y	Y
3	3	Y	Y
4	1	Y	N
5	2	Y	N
6	3	Y	N
7	1	N	Y
8	2	N	Y
9	3	N	Y
10	1	N	N
11	2	N	N
12	3	N	N

we have 3 possible values for the TXN_CODE field, 2 possible values for the COUPON_IND field, and 2 possible values for the PREPAY_IND field. This results in a total of 3 x 2 x 2 = 12 rows for the junk dimension table.

By using a junk dimension to replace the 3 indicator fields, we have decreased the number of dimensions by 2 and also decreased the number of fields in the fact table by 2. This will result in a data warehousing environment that offer better performance as well as being easier to manage.

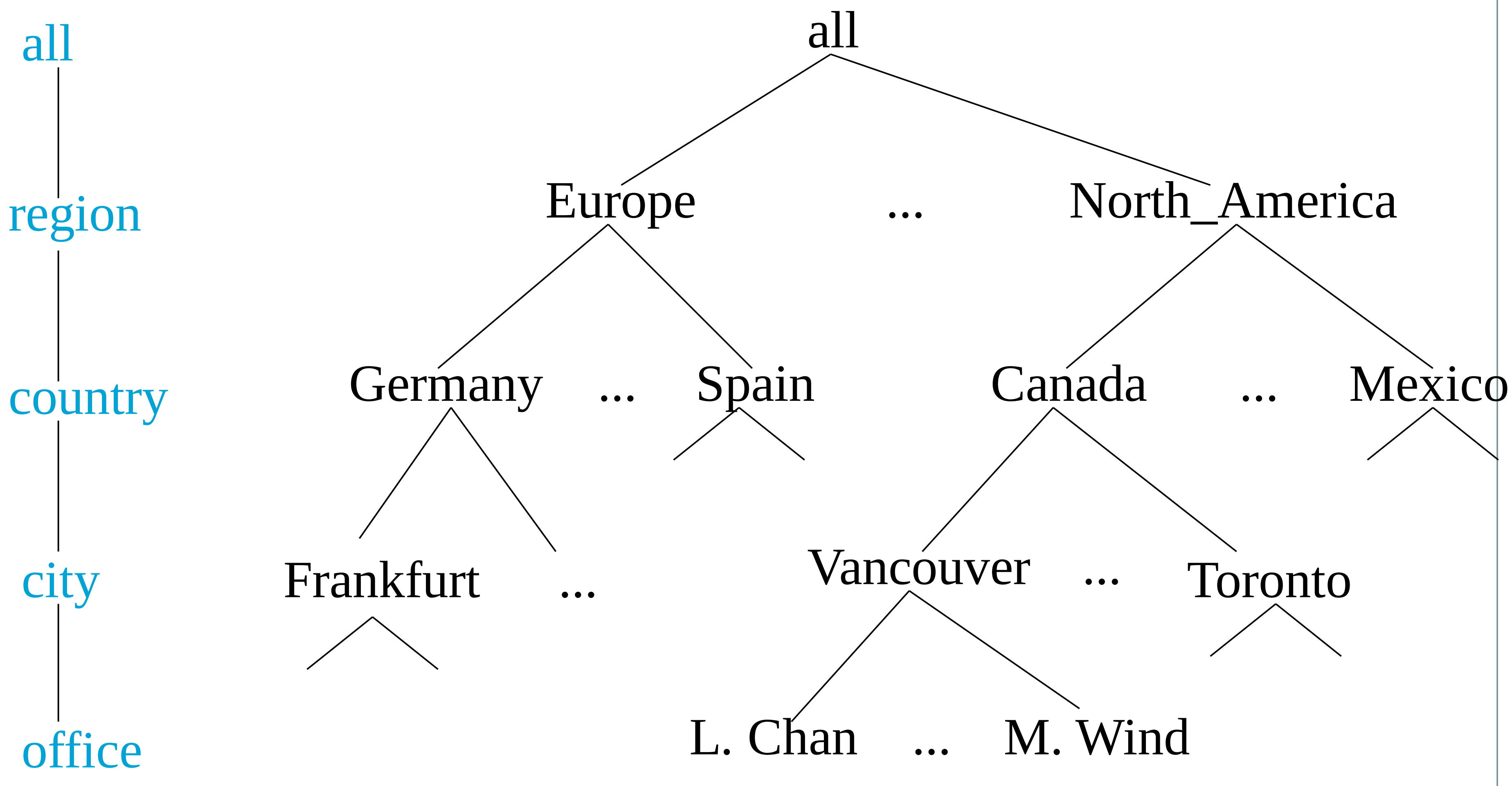
Data lake, Architecture of Data lake, Data Warehouse vs Data lake

Data lake data often comes from disparate sources and can include a mix of structured, semi-structured , and unstructured data formats. Data is stored with a flat architecture and can be queried as needed. For companies that need to collect and store a lot of data but do not necessarily need to process and analyze all of it right away, a data lake offers an effective solution that can load and store large amounts of data very rapidly without transformation.

Why do you need a data lake?

Organizations that successfully generate business value from their data, will outperform their peers. An Aberdeen survey saw organizations who implemented a Data Lake outperforming similar companies by 9% in organic revenue growth. These leaders were able to do new types of analytics like machine learning over new sources like log files, data from click-streams, social media, and internet connected devices stored in the data lake. This helped them to identify, and act upon opportunities for business growth faster by attracting and retaining customers, boosting productivity, proactively maintaining devices, and making informed decisions.

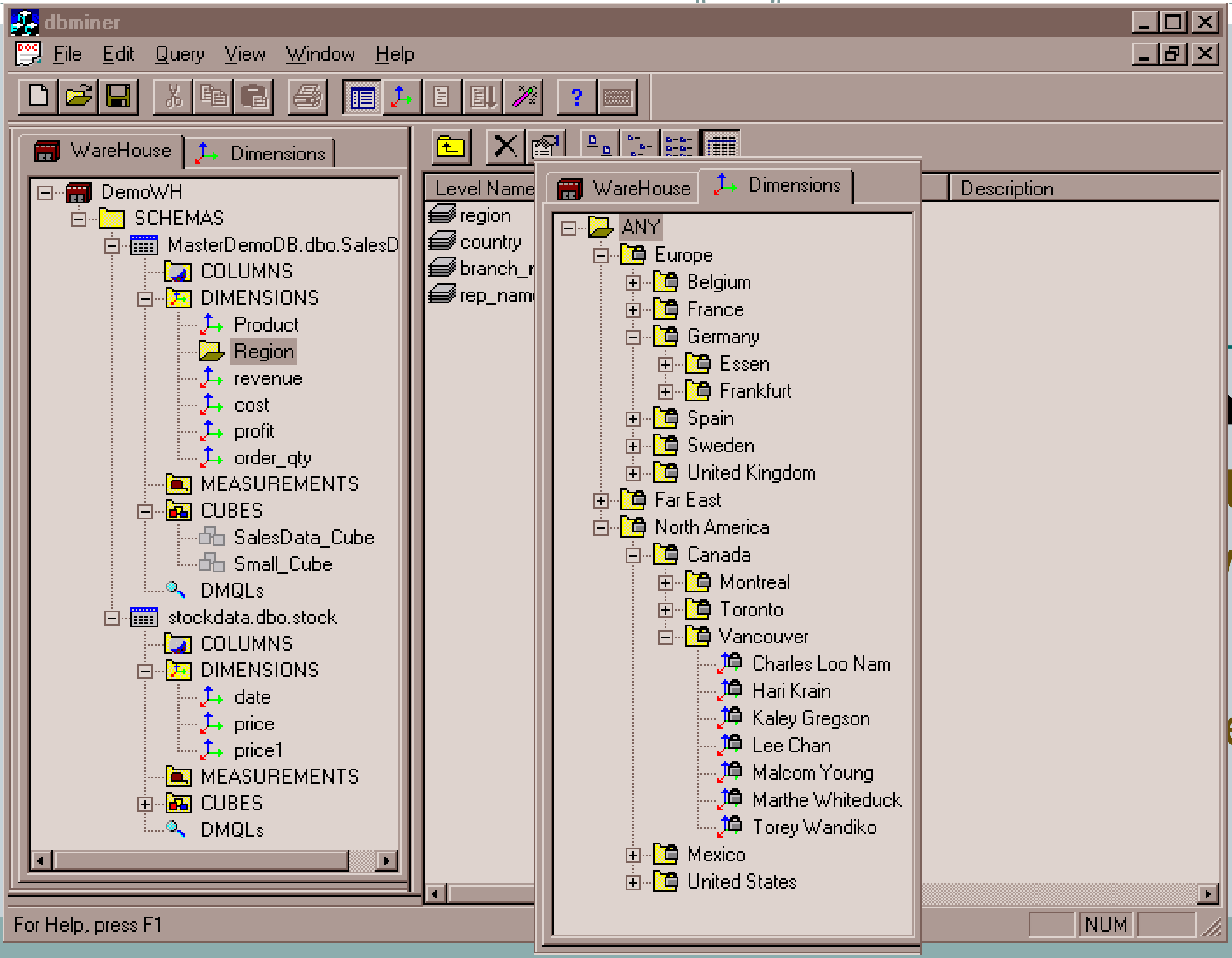
A Concept Hierarchy: **Dimension** (location)



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()

View of Warehouses and Hierarchies



dbminer

File Edit Query View Window Help

WareHouse | Dimensions

- DemoWH
  - SCHEMAS
    - MasterDemoDB.dbo.SalesD
      - COLUMNS
      - DIMENSIONS
        - Product
        - Region
        - revenue
        - cost
        - profit
        - order_qty
      - MEASUREMENTS
      - CUBES
        - SalesData_Cube
        - Small_Cube
      - DMQLs
    - stockdata.dbo.stock
      - COLUMNS
      - DIMENSIONS
        - date
        - price
        - price1
      - MEASUREMENTS
      - CUBES
      - DMQLs

Level Name	Description
region	
country	
branch_r	
rep_nam	

WareHouse | Dimensions

- ANY
  - Europe
    - Belgium
    - France
    - Germany
      - Essen
      - Frankfurt
    - Spain
    - Sweden
    - United Kingdom
  - Far East
  - North America
    - Canada
      - Montreal
      - Toronto
      - Vancouver
        - Charles Loo Nam
        - Hari Krain
        - Kaley Gregson
        - Lee Chan
        - Malcom Young
        - Marthe Whiteduck
        - Torey Wandiko
    - Mexico
    - United States

For Help, press F1 | NUM

hierarchies

rchy

th <

eek} < year

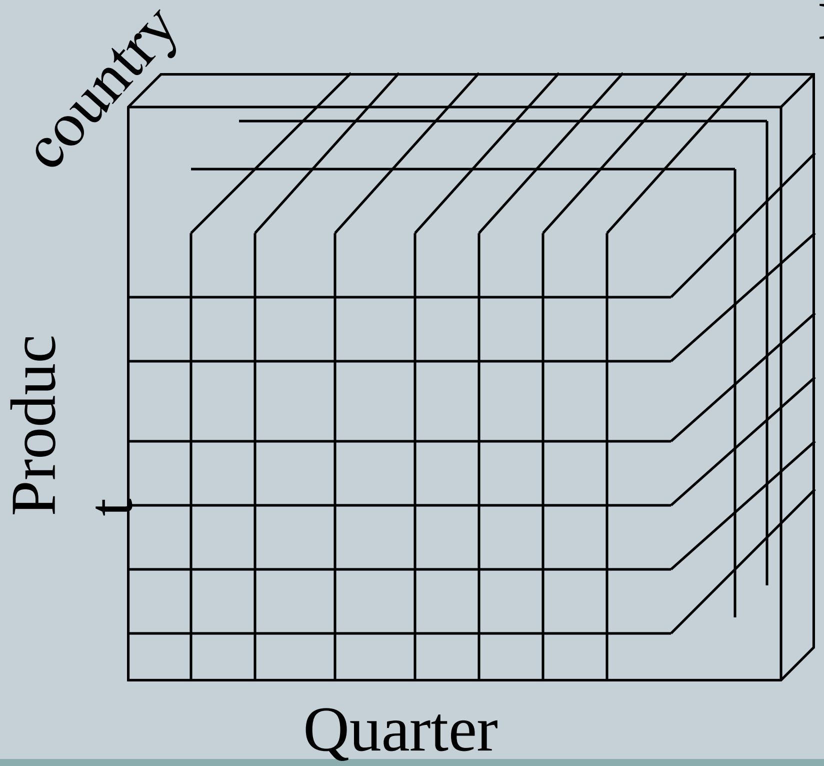
hierarchy

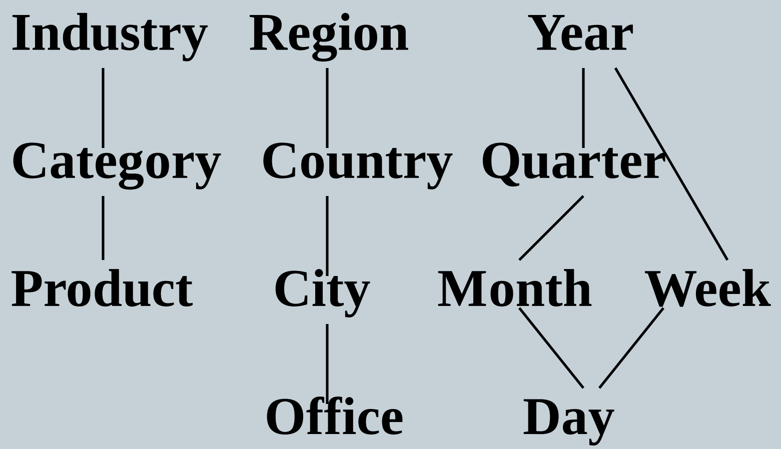
expensive

Multidimensional Data

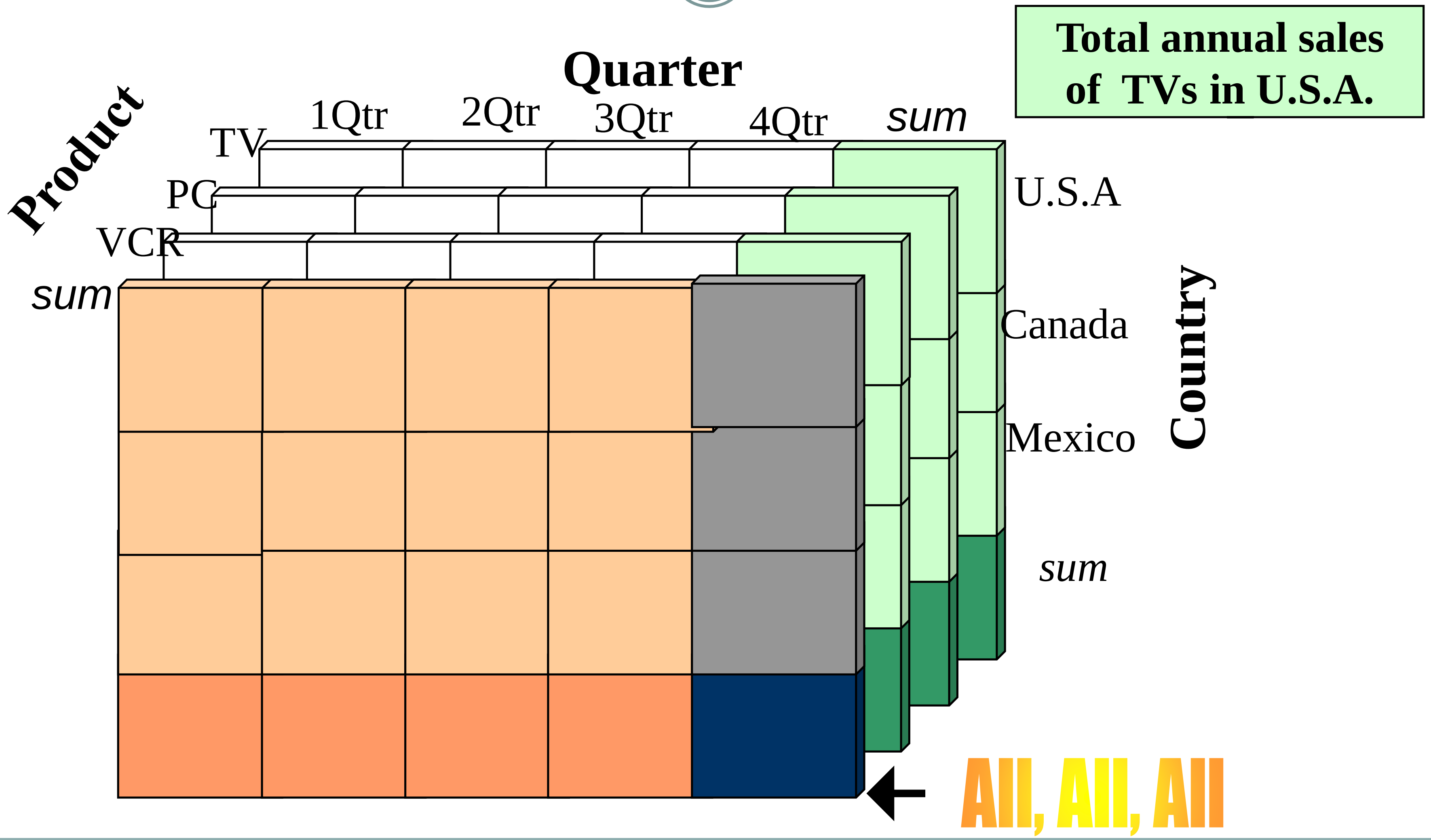
- Sales volume as a function of product, quarter, and country

Dimensions: *Product, Location, Time*
Hierarchical summarization paths

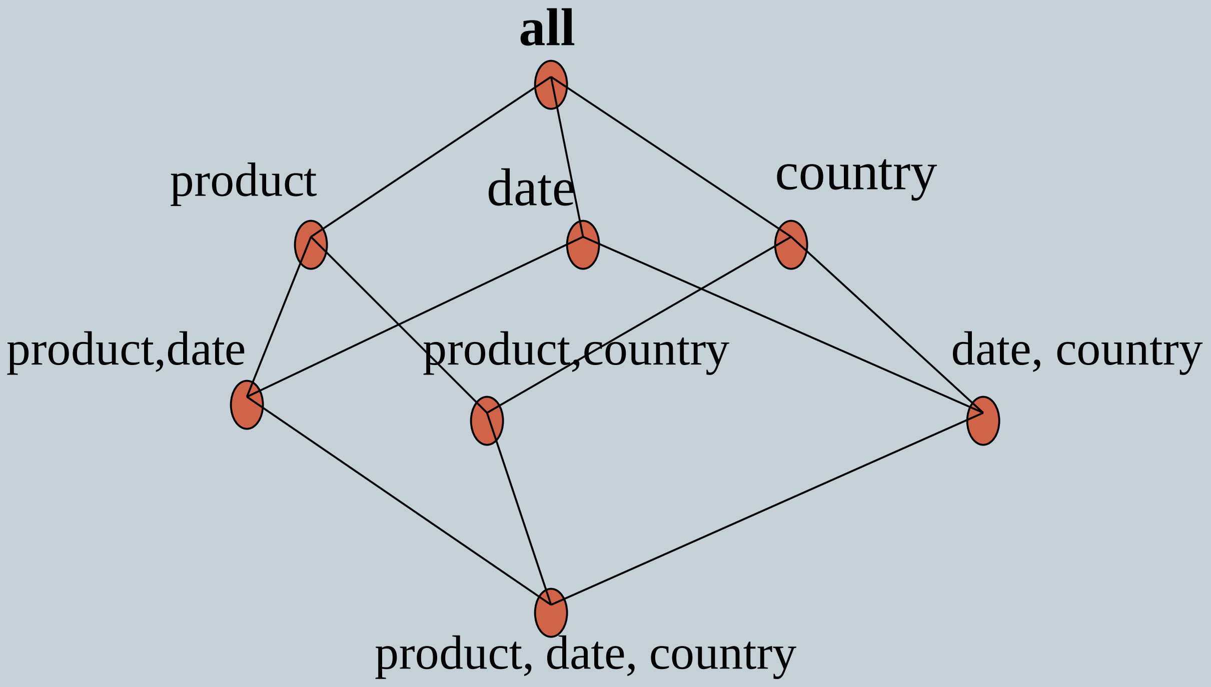




A Sample Data Cube



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

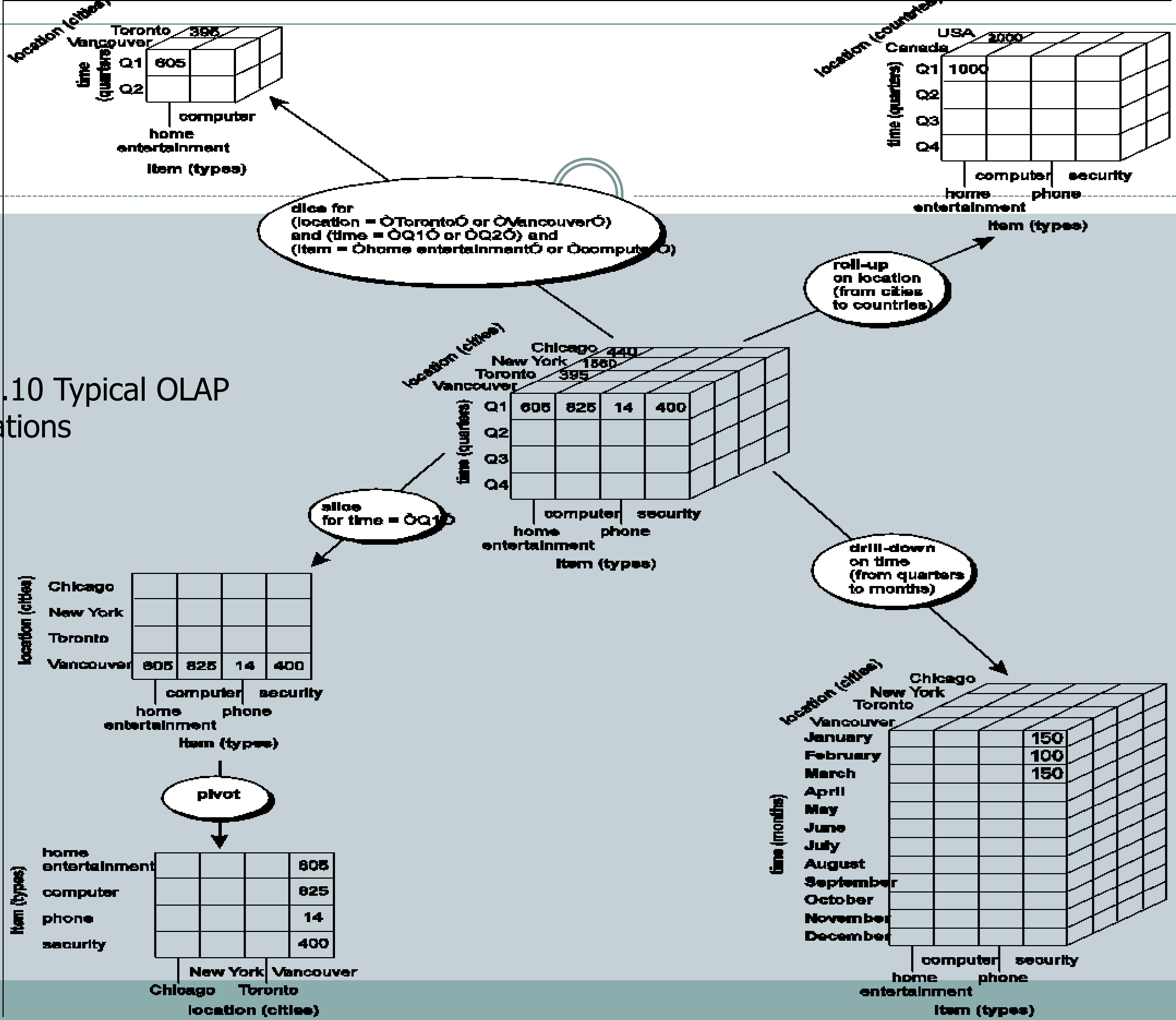
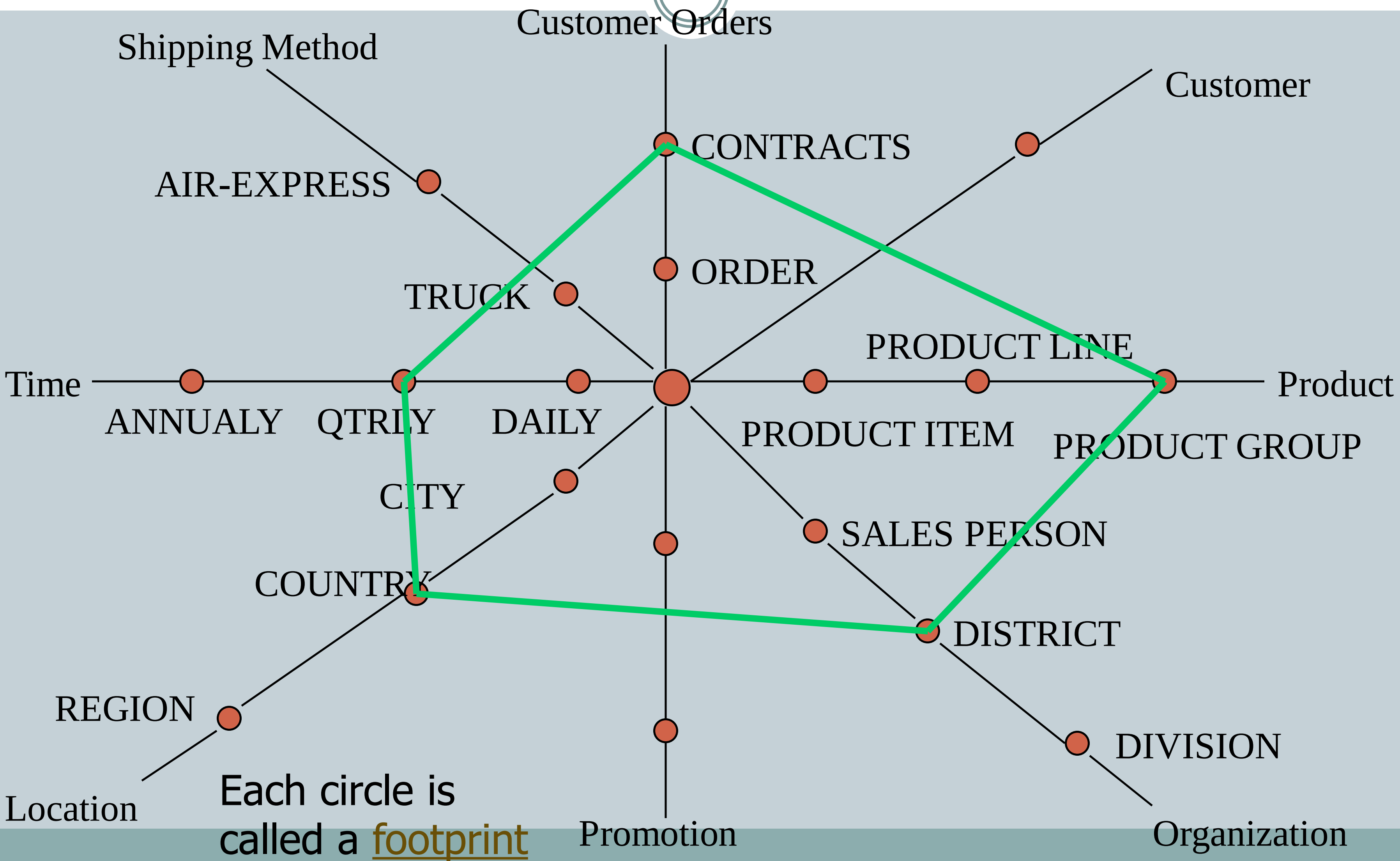


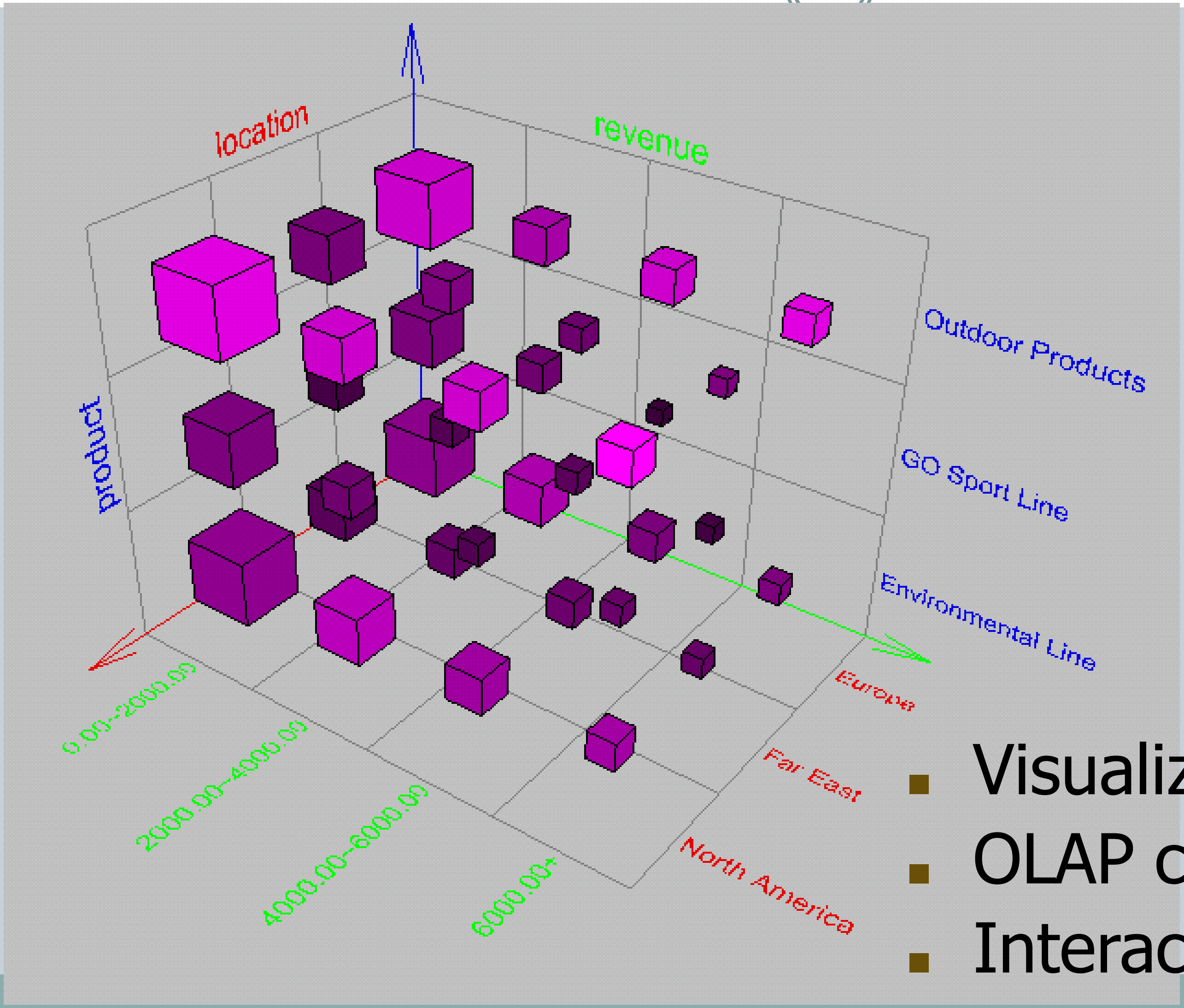
Fig. 3.10 Typical OLAP Operations

A Star-Net Query Model



Each circle is called a footprint

Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

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

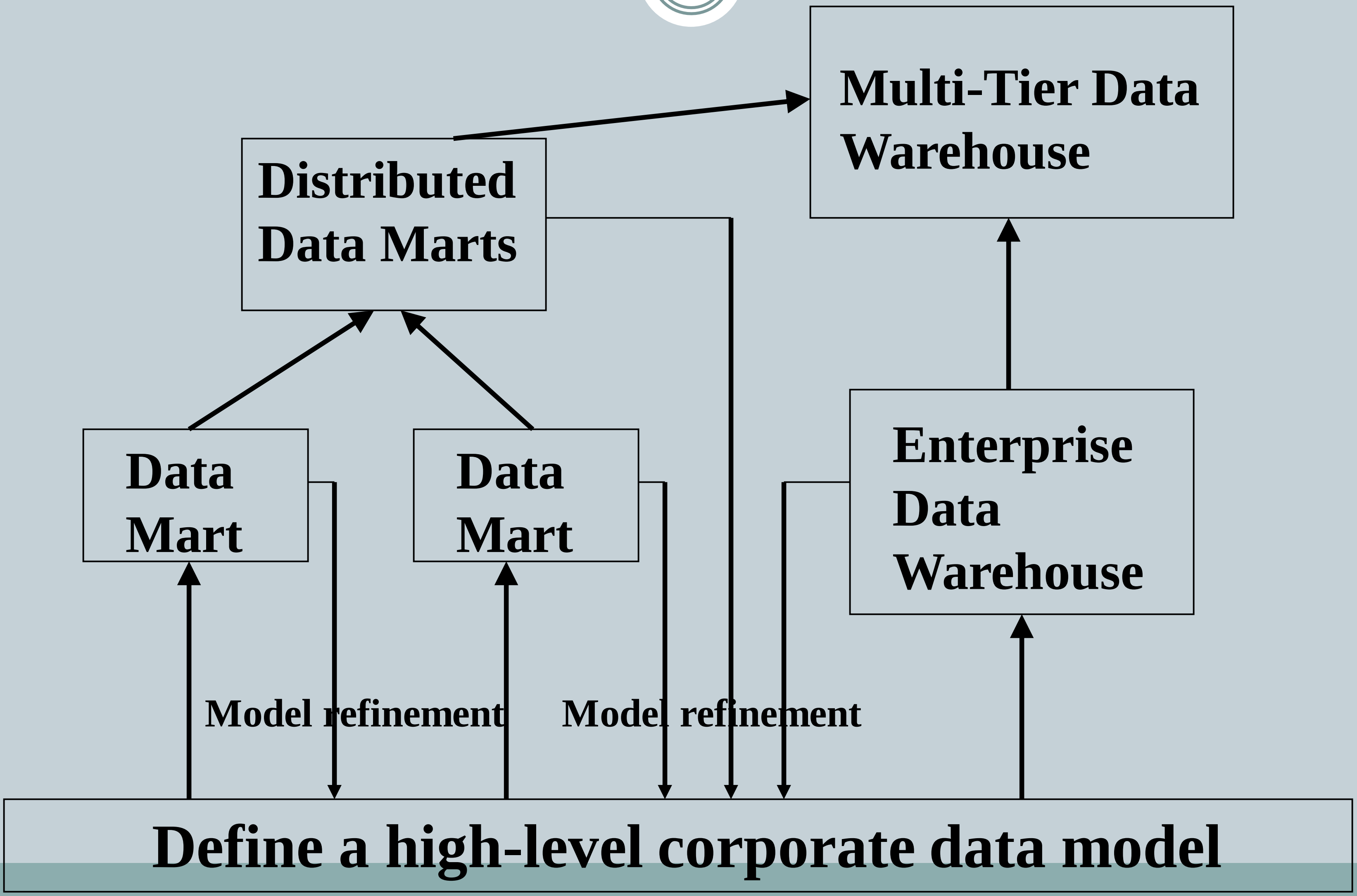
Design of Data Warehouse: A Business Analysis Framework

- 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**
  - 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

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 <u>every</u> (cuboid) (**full materialization**), <u>none</u> (**no materialization**), or <u>some (**partial materialization**)</u>
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.

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

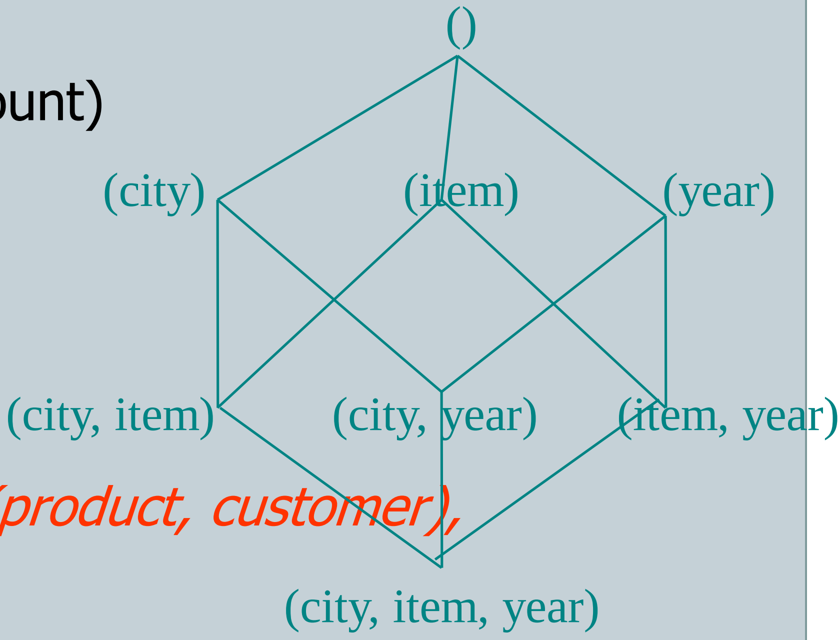
- Need compute the following Group-Bys

  (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]

Base table

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

Index on Region

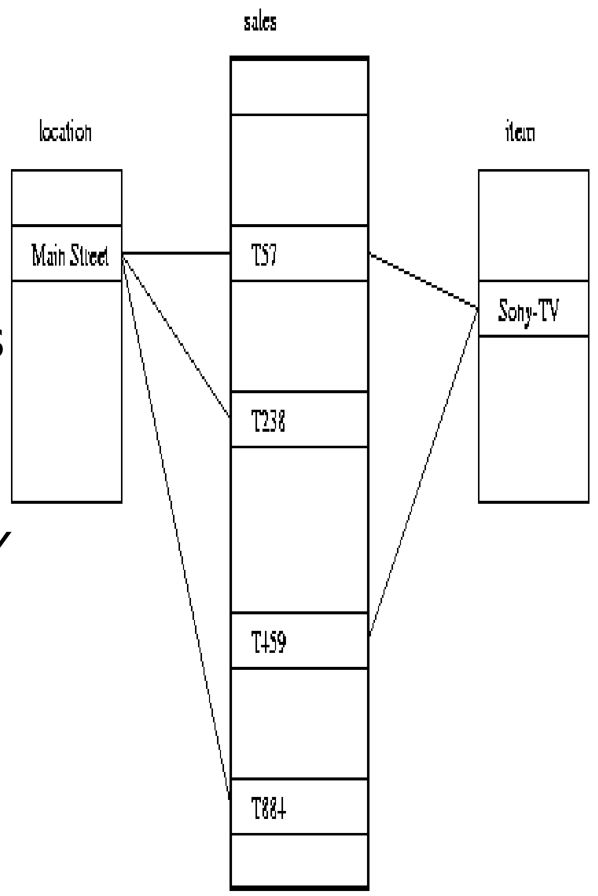
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

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 start 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

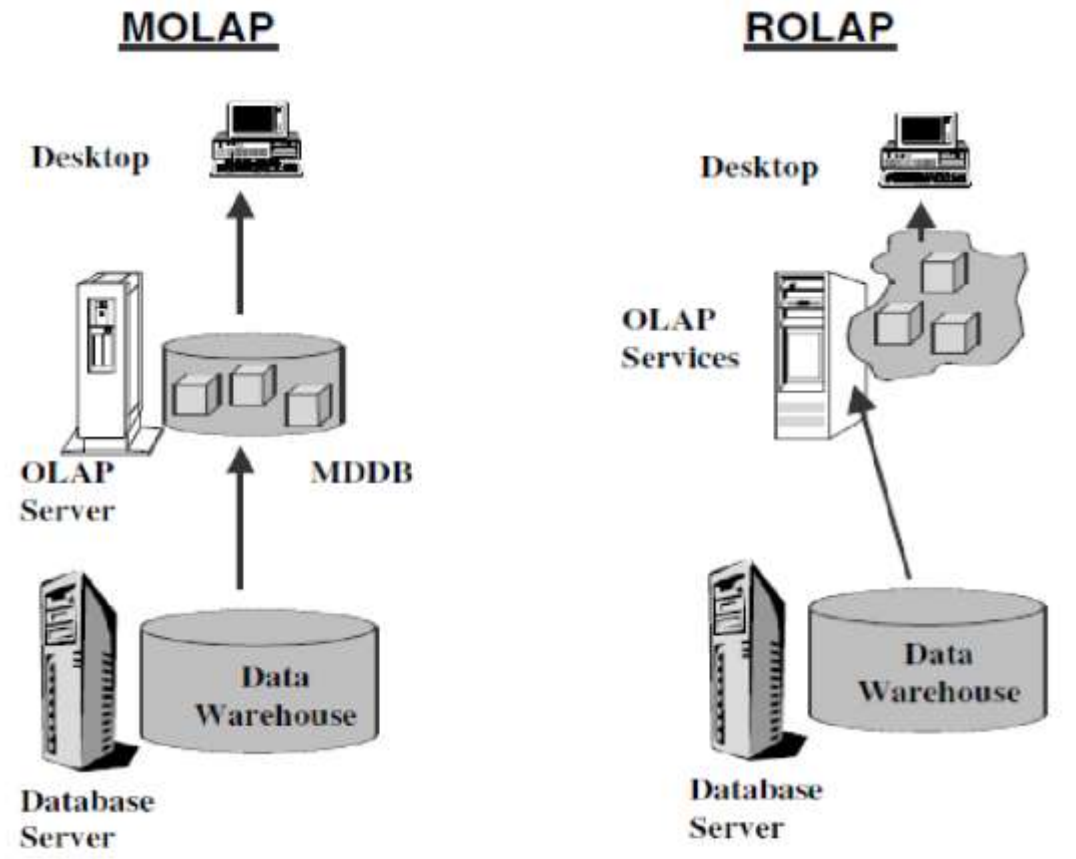


Figure 15-15 OLAP models.

References (II)

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

- 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.