Sessions 5 - 6

Data Warehouse, Data Lake, and Data Lakehouse

Big Data Analytics Technology, MSc in Data Science,
Coventry University UK

Miyuru Dayarathna

Outline

- Data Warehouse
- Data Lake
- Data Lakehouse

Need for Data Warehouses : Some commonly found statements

- We collect tons of data, but we can’t access it.
- We need to slice and dice the data every which way.
- Business people need to get at the data easily.
- Just show me what is important.
- We spend entire meetings arguing about who has the right numbers rather than making decisions.
- We want people to use information to support more fact-based decision making.

Types of Analytics

- Analytics is divided into three major categories:
  - descriptive
  - predictive
  - prescriptive.
- Over the years, there has been a clear progression from relatively simple descriptive analytics to more advanced, forward looking and guiding forms of analytics.

Types of Analytics (Contd.)

- **Descriptive analytics** - is the oldest form of analytics
  - It primarily focuses on describing the past status of the domain of interest using a variety of tools through techniques such as reporting, data visualization, dashboards, and scorecards.
  - **Online analytical processing (OLAP)** is also part of descriptive analytics
    it allows users to get a multidimensional view of data and drill down deeper to the details when appropriate and useful

Types of Analytics (Contd.)

- **Predictive analytics** - systems apply statistical and computational methods and models to data regarding past and current events to predict what might happen in the future (potentially depending on a number of assumptions regarding various parameters)

Types of Analytics (Contd.)

- **Prescriptive analytics** focuses on the question “How can we make it happen?” or “What do we need to do to make it happen?”
- For prescriptive analysis we need optimization and simulation tools and advanced modeling to understand the dependencies between various actors within the domain of interest.

Types of Analytics (Contd.)

Type of Analytics	Key Questions
Descriptive Analytics	What happened yesterday/last week/last year?
Predictive Analytics	What might happen in the future? How does this change if we change assumptions?
Prescriptive Analytics	How can we make it happen? What needs to change to make it happen?

Need for Data Warehouses : Heterogeneous Information Sources

- Different interfaces to access data
- Different data representations
  - Free-form vs. structured fields
- Duplicate and inconsistent information
- Missing data
- Inconsistent key structures
- Synonyms

Organizational Trends Motivating Data Warehouses

- No single system of records
- Multiple systems not synchronized
- Organizational need to analyze activities in a balanced way
- Customer relationship management
- Supplier relationship management

Separating Operational and Informational Systems

- Operational system
  - A system that is used to run a business in real-time, based on current data (Also known as a system of record)
  - E.g., sales order processing, reservation systems, and patient registration systems
  - Operational systems must process large volumes of relatively simple read/write transactions and provide fast response.

Separating Operational and Informational Systems (Contd.)

- Informational System
  - A system designed to support decision making based on historical point-in-time and prediction data for complex queries or data-mining applications
  - E.g., systems for sales trend analysis, customer segmentation, and human resources planning

Comparison between Operational and Informational Systems

Characteristic	Operational Systems	Informational Systems
Primary purpose	Run the business on a current basis	Support managerial decision making
Type of data	Current representation of state of the business	Historical point-in-time (snapshots) and predictions
Primary users	Clerks, salespersons, administrators	Managers, business analysts, customers
Scope of usage	Narrow, planned, and simple updates and queries	Broad, ad hoc, complex queries and analysis
Design goal	Performance: throughput, availability	Ease of flexible access and use
Volume	Many constant updates and queries on one or a few table rows	Periodic batch updates and queries requiring many or all rows

Example of Heterogeneous Data

STUDENT DATA (from class registration system)

StudentNo	LastName	MI	FirstName	Telephone	Status	...
123-45-6789	Enright	T	Mark	588-1967	Soph	
389-21-4063	Smith	R	Elaine	283-4295	Jr	

Three tables from three separate systems of record, each containing similar student data.

STUDENT EMPLOYEE (from personal system)

StudentID	Address	Dept	Hours	...
123-45-6789	1218 Elk Drive, Phoenix, AZ91304	Soc	8	
389-21-4063	134, Messa Road, Tempe, AZ 90142	Math	10	

STUDENT HEALTH (from health center system)

StudentID	Telephone	Insurance	ID	...
Mark T. Enright	483-1967	Blue Cross	123-45-6789	
Elaine R. Smith	555-7828	?	389-21-4063	

Example of Heterogeneous Data (Contd.)

Suppose you want to develop a profile for each student, consolidating all data into a single file format. Some of the issues that you must resolve are as follows,

- **Inconsistent key structures** : The primary key of the first two tables is some version of the student Social Security number, whereas the primary key of STUDENT HEALTH is StudentName.
- **Synonym** : In STUDENT DATA, the primary key is named StudentNo, whereas in STUDENT EMPLOYEE it is named StudentID.

Example of Heterogeneous Data (Contd.)

- **Free-form fields versus structured fields** : In STUDENT HEALTH, StudentName is a single field. In STUDENT DATA, StudentName (a composite attribute) is broken into its component parts: LastName, MI, and FirstName.
- **Inconsistent data values** : Elaine Smith has one telephone number in STUDENT DATA but a different number in STUDENT HEALTH. Is this an error, or does this person have two telephone numbers?
- **Missing data** : The value for Insurance is missing (or null) for Elaine Smith in the STUDENT HEALTH table. How will this value be located?

Example of Heterogeneous Data (Contd.)

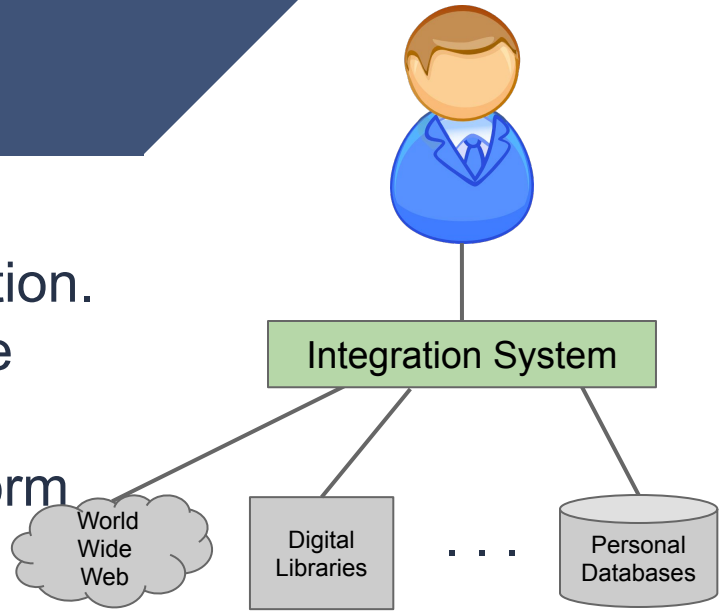
This simple example illustrates the nature of the problem of developing a single corporate view but fails to capture the complexity of that task. A real-life scenario would likely have dozens (if not hundreds) of tables and thousands (or millions) of records.

Need for Data Warehouses : Data Management in Large Enterprises

- Vertical fragmentation of informational systems (vertical stove pipes)
- Result of application (i.e., user) driven development of operational systems

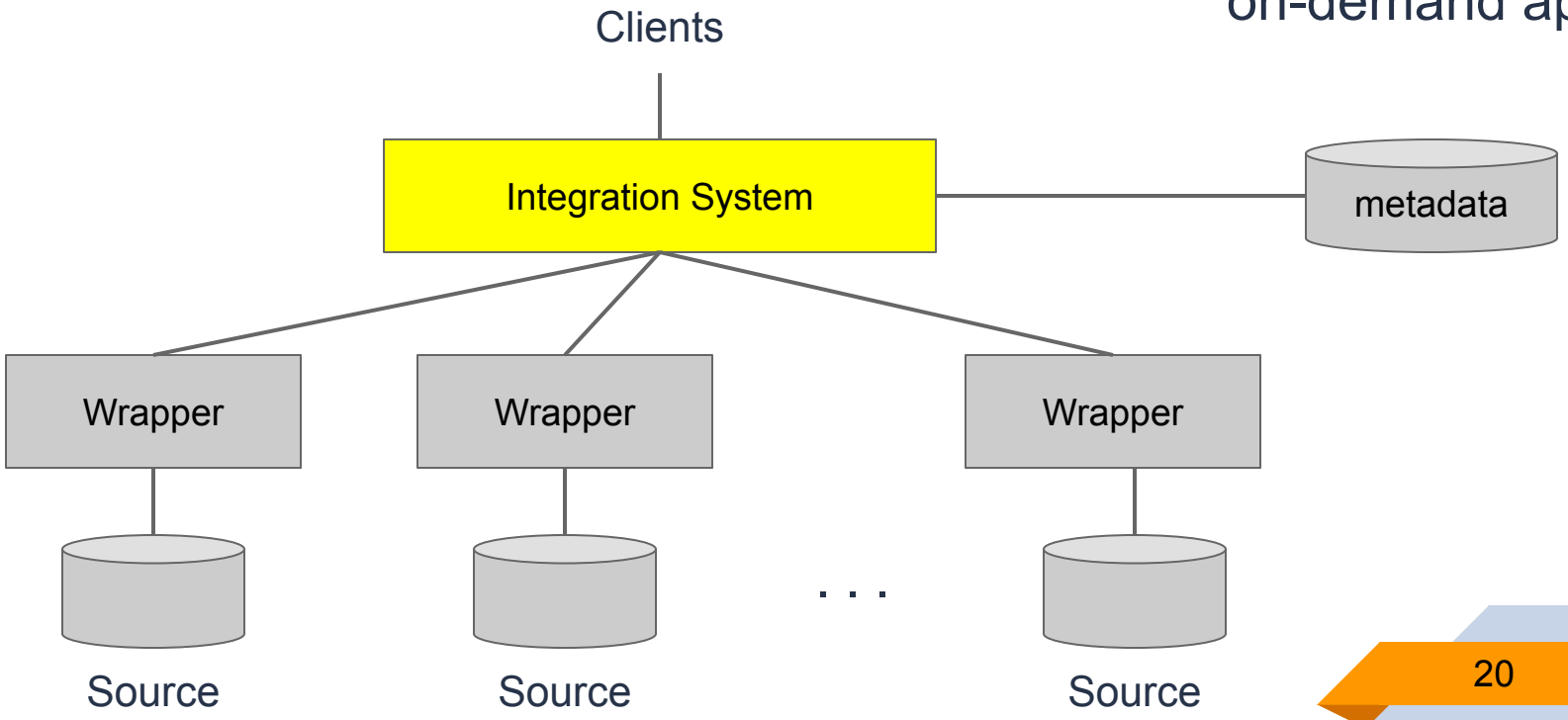
Need for Data Warehouses

- A business requires an integrated, company-wide view of high-quality information.
  - Integrated systems collect and combine information
  - They provide integrated view, and uniform user interface
  - They support information sharing
- The information systems department must separate informational from operational systems to improve performance dramatically in managing company data.



Traditional Solution : Query Driven Approach

This is a lazy, on-demand approach



Advantages of Query-driven approach

- Query-driven approach still better for
  - Rapidly changing information
  - Rapidly changing information sources
  - Truly vast amounts of data from large numbers of sources
  - Clients with unpredictable needs

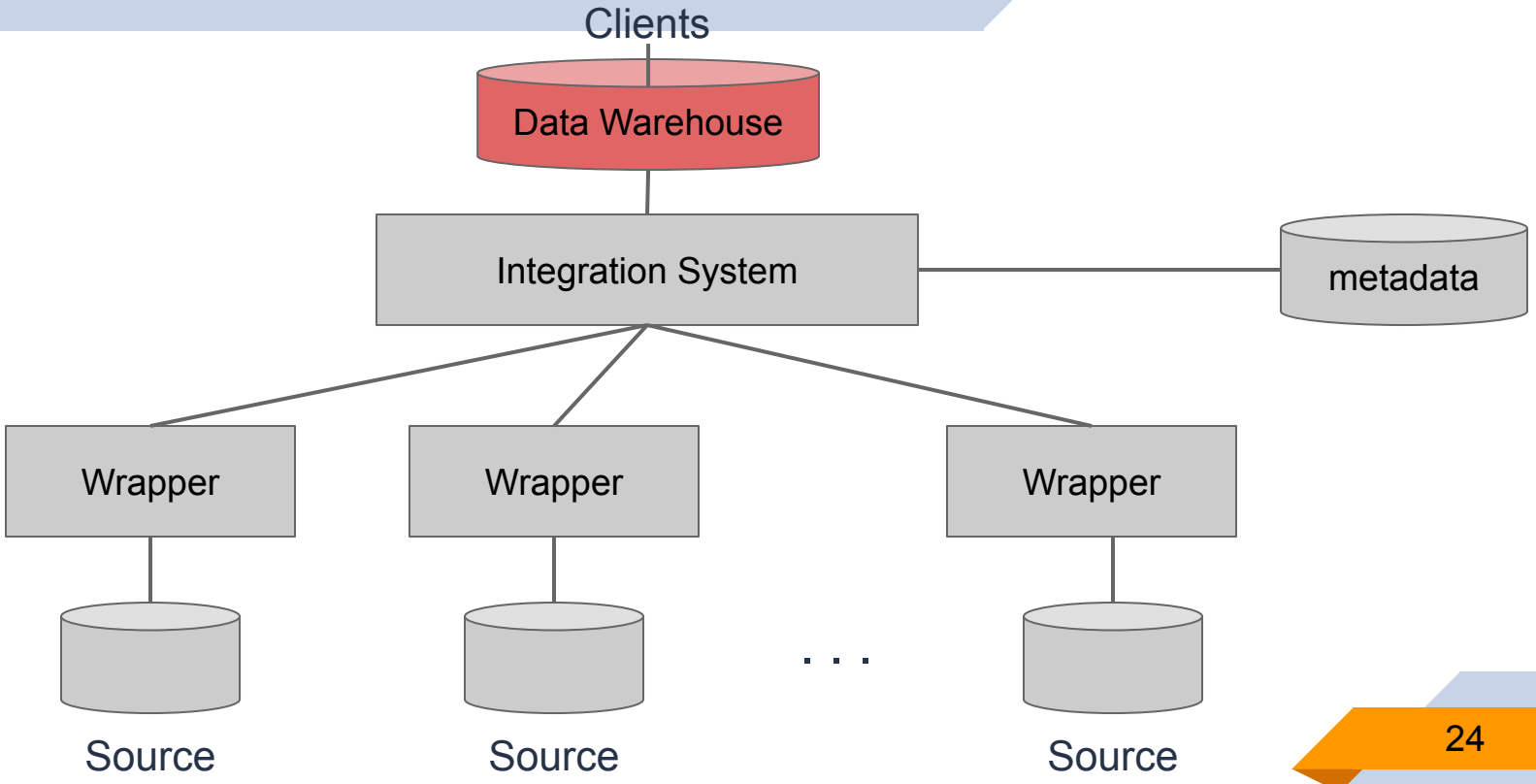
Query Driven Approach - Disadvantages

- Delay in query processing
  - Slow/unavailable information sources
  - Complex filtering and integration
- Competes with local processing at sources
- Potentially Expensive and inefficient for frequent queries

Warehousing Approach

- Information integrated in advance
  - Integrated, company-wide view of high-quality information (from disparate databases)
- Stored in warehouse for direct querying and analysis
- Separation of operational and informational systems and data (for improved performance)

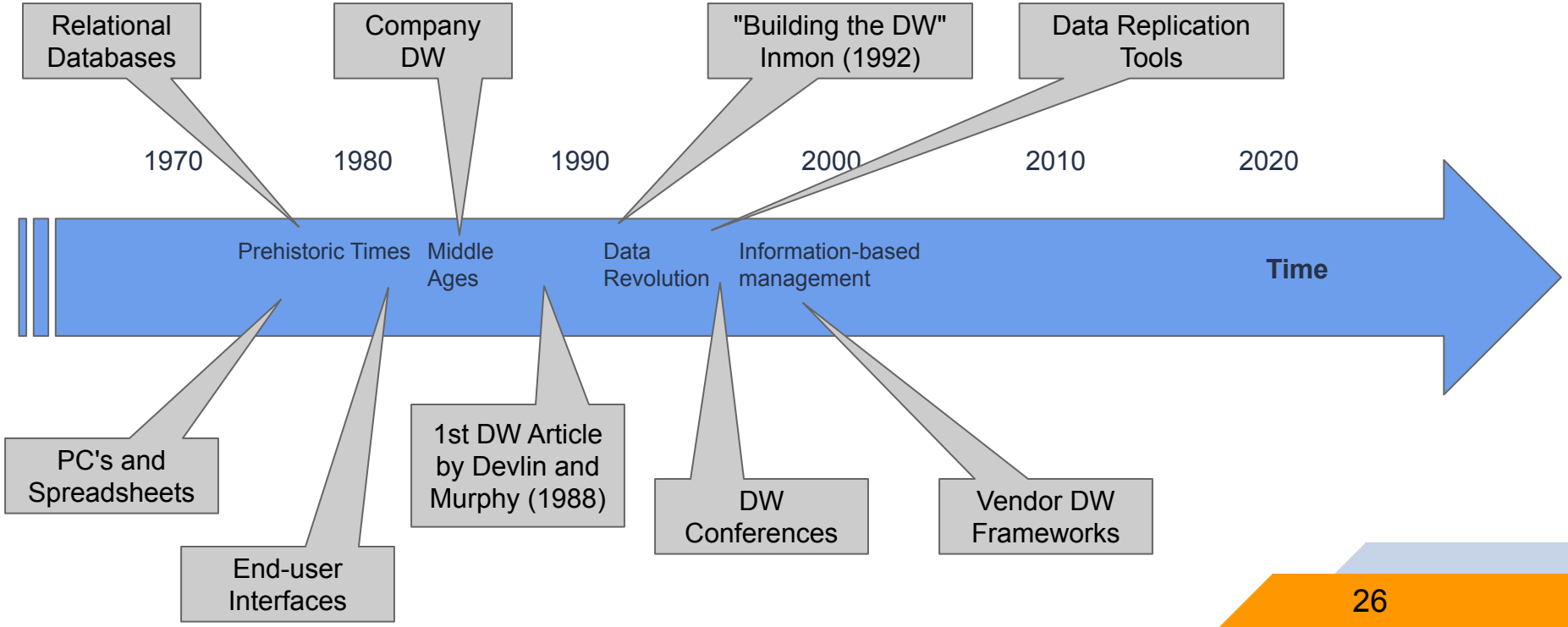
Warehousing Approach



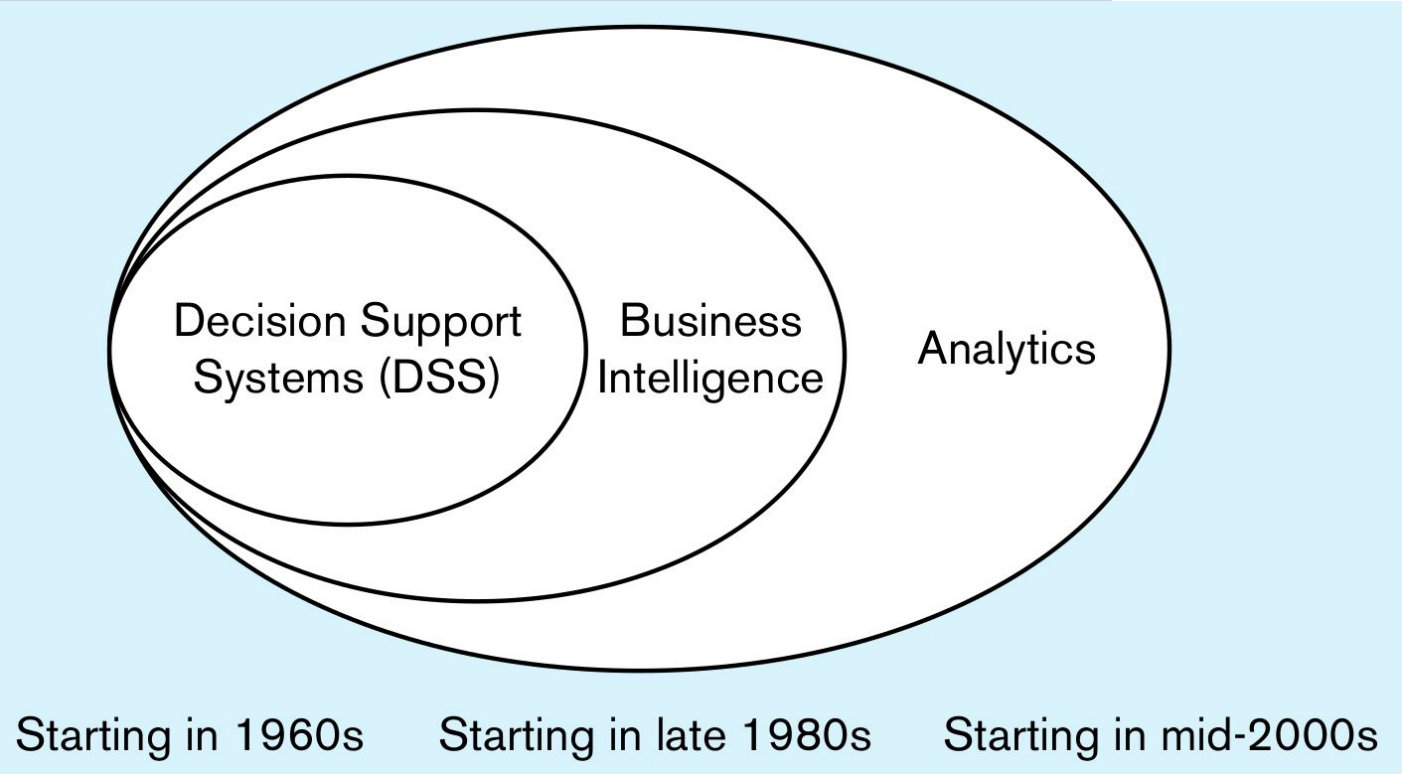
Advantages of Warehousing Approach

- Doesn't interfere with local processing at sources
  - Complex queries at warehouse
  - OLTP at information sources
- Information copied at Warehouse
  - Can modify, annotate, summarize, restructure, etc.
  - Can store historical information
  - Security, no auditing
- Has caught on in industry

Data Warehouse Evolution



Moving from decision support systems to analytics



Need for Data Warehouses - Characteristics of Traditional databases

- Traditional (i.e., Production) DBMSs are designed to manage operational data
  - Need to support everyday activities
  - Online transaction processing (OLTP)
- Traditional databases are not optimized for data access
  - Instead they are optimized for both transaction processing and integrity assurance
  - Hence need to balance the requirement of data access with the need to ensure data integrity

Need for Data Warehouses (Contd.)

- Data warehouses are designed to analyze and explore data
  - Need to summarize and discover trends to support decision making
  - Online analytical processing (OLAP)

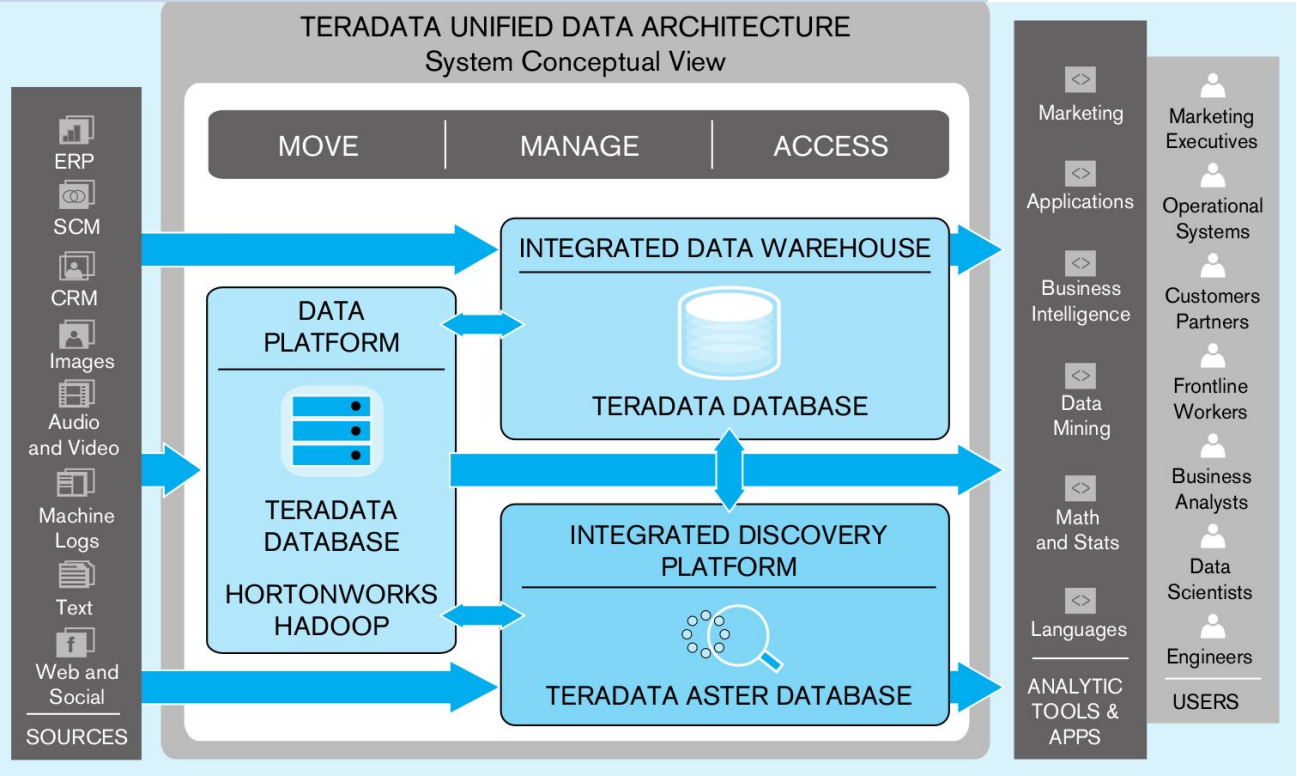
Features of Data Warehouses

- Data warehouses contain consolidated data from many sources augmented with summary information and covering a long time period.
  - May include databases from different data models
  - May include files acquired from independent systems and platforms
  - Need to create a single unified schema
  - The warehouse is like a materialized view
- Complex read-only queries
  - Most of the times the data warehouse users need only read access but, need the access to be fast over a large volume of data.
- Very large data size (Terabytes of data are common)
- Fast response time is important

Features of Data Warehouses

- Data Warehouse (DW) was introduced as a new type of database management system which would keep no transactional data but only maintain summarized historical information for decision making purposes
  - Are modeled and structured differently
  - Use different techniques for storage and retrieval
  - Cater to a different set of users

Teradata Unified Data Architecture – system conceptual view



Data Warehouse - A commonly used definition

- W. H. Inmon's definition of a data warehouse,

"A Data Warehouse is a
  - subject-oriented,
  - integrated,
  - time-variant,
  - non-volatile

collection of data used in support of management decision making processes."

W.H. Inmon (Known as the Father of the Data Warehouse)

-- Inmon & Hackathorn, 1994: viz. Hoffer, Chap 11

Data Warehouse - A commonly used definition (Contd.)

- **Subject-Oriented**
  - The data warehouse is organized around the key subjects (or high-level entities) of the enterprise. Major subjects include
    - Customers
    - Patients
    - Students
    - Products
    - Etc.

Data Warehouse - A commonly used definition (Contd.)

- **Integrated**
  - The data housed in the data warehouse are defined using consistent
    - Naming conventions
    - Formats
    - Encoding Structures
    - Related Characteristics
  - From multiple data sources

Data Warehouse - A commonly used definition (Contd.)

- Time-variant
  - The data in the warehouse contain a time dimension so that they may be used as a historical record of the business
  - Can study trends and changes

Data Warehouse - A commonly used definition (Contd.)

- Non-volatile (i.e., Non-updatable)
  - Data in the data warehouse are loaded and refreshed from operational systems, but cannot be updated by end-users
  - Read-only, periodically refreshed

Data warehouses - Another Definition

- Data warehouses are databases that store and maintain analytical data separately from transaction-oriented databases for the purpose of decision support
  - Traditional databases support online transaction processing - OLTP .
  - Data Warehouses are for analytical applications- largely OLAP.

Data warehouses - A practitioner's viewpoint

“A data warehouse is simply a single, complete, and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use it in a business context.”

-- Barry Devlin, IBM Consultant

Applications of Data Warehouses

- Online Analytical Processing (OLAP) - Corresponds to the analysis of complex data from the data warehouse
- Decision Support Systems (DSS)
  - Also known as Executive Information Systems (EIS)
  - Supports organization's leading decision makers for making complex and important decisions

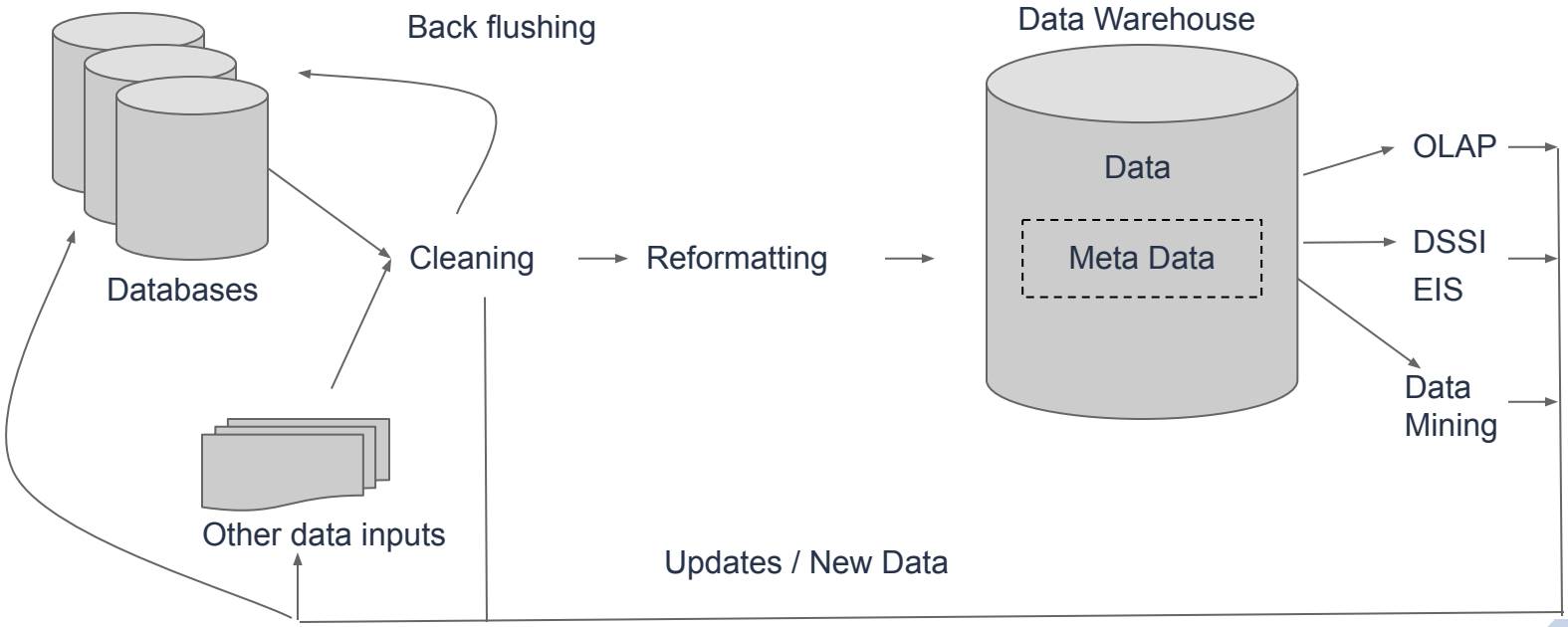
Structure of Data Warehouse

- Data Warehouse Processing includes,
  - Cleaning and formatting of data
  - ETL (Extract, Transform, Load)
  - OLAP - Data Analytics
  - Data Mining

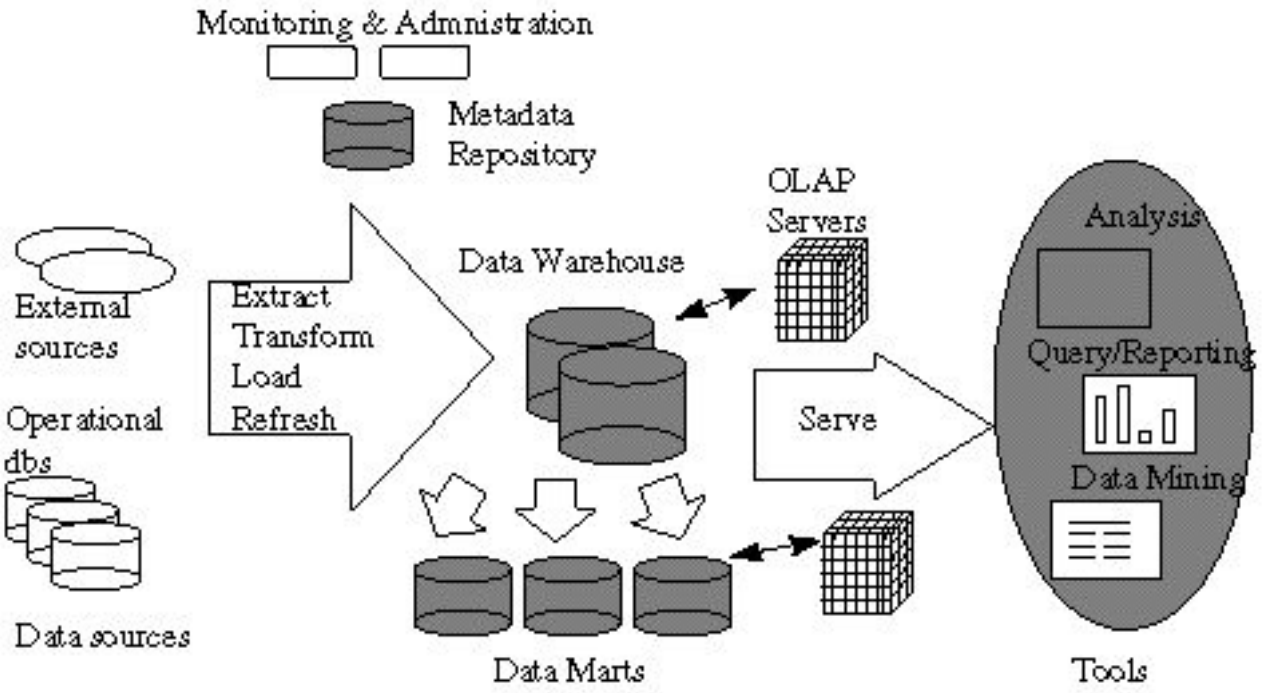
Types of Data

- Business Data - represents meaning
  - Real-time data (ultimate source of all business data)
  - Reconciled data
  - Derived data
- Metadata - describes meaning
  - Build-time metadata
  - Control metadata
  - Usage metadata
- Data as a product* - intrinsic meaning
  - Produced and stored for its own intrinsic value
    - e.g., the contents of a text-book

General Architecture of a Data Warehouse

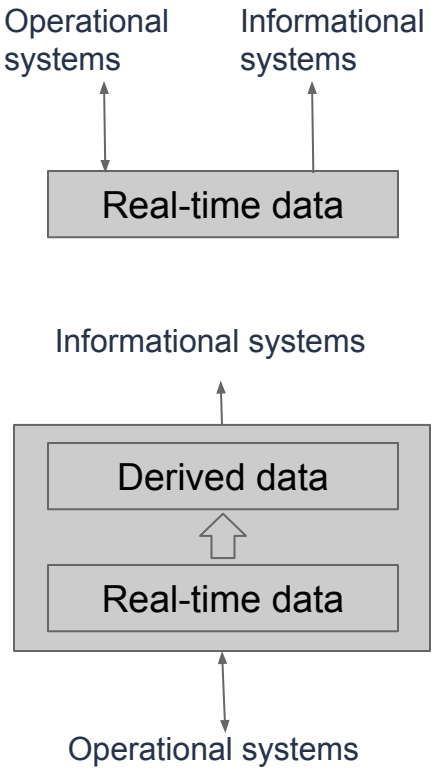


DW Architecture - Another View



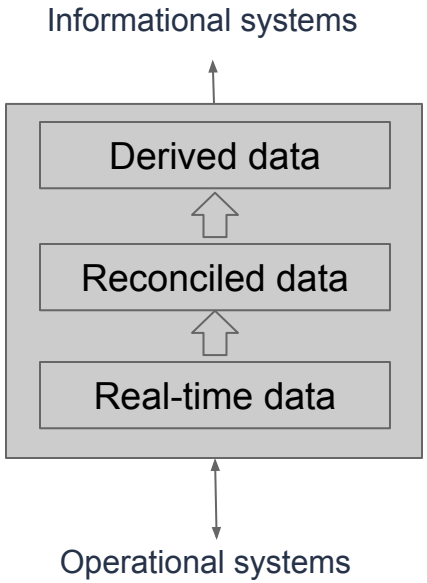
Data Warehouse Architectures: Conceptual View

- Single-layer
  - Every data element is stored only once
  - Virtual warehouse
- Two-layer
  - Real-time + derived data
  - Most commonly used approach in industry



Data Warehouse Architectures: Conceptual View

- Transformation of real-time data to derived data really requires two steps



View level
“Particular informational needs”

Physical Implementation of the Data Warehouse

DW compared to traditional databases

- Data Warehouses are mainly optimized for appropriate data access.
  - Traditional databases are transactional and are optimized for both transaction processing and integrity assurance.
- Data warehouses emphasize more on historical data as their main purpose is to support time-series and trend analysis.
- In transactional databases transaction is the mechanism of change to the database. By contrast, information in data warehouse is relatively coarse grained and DWs are regarded as non-real time.
- The periodic refresh policy is carefully chosen, usually incremental.
- Compared with transactional databases, data warehouses are nonvolatile.

DW compared to traditional databases (Contd.)

Traditional DB	DW
Mostly updates	Mostly reads
MB - GB size scale data	GB - TB of size cale data
Current snapshot	History
Many small transactions	Queries are long and complex
Index/hash on primary key	Lots of scans
Raw data	Summarized, reconciled data
Thousands of users (e.g., clerical users)	Hundreds of users (e.g., decision-makers, analysts)

Characteristics of Data Warehouses

- Multidimensional Conceptual View
- Unlimited dimensions and aggregation levels
- Unrestricted cross-dimensional operations
- Dynamic sparse matrix handling
- Client-server architecture

Characteristics of Data Warehouses (Contd.)

- Stored collection of diverse data
  - A solution to data integration problem
  - Single repository of information
- Large volume of data (GB, TB)
- Subject-oriented
  - Organized by subject, not by application
  - Used for analysis, data mining, etc.
- Optimized differently from transaction-oriented db

Characteristics of Data Warehouses (Contd.)

- Multi-user support
- Accessibility
- Transparency
- Intuitive Data Manipulation
  - User interface aimed at executive decision makers and analysts
- Inductive and deductive analysis
- Flexible distributed reporting

Characteristics of Data Warehouses (Contd.)

- Non-volatile
  - Historical
  - Time attributes are important
- Updates infrequent
- May be append-only
- Examples
  - All transactions ever at WalMart
  - Complete client histories at insurance firm
  - Stockbroker financial information and portfolios

Data Warehouse Classification

- DW can be classified based on the magnitude of the data they store.
  - **Enterprise-wide data warehouses**
  - **Virtual data warehouses**
  - **Logical data warehouses**
  - **Data marts**

Enterprise-wide data warehouses (EDW)

- These are gigantic projects requiring huge investment of resources and time
- Typically EDWs are relational data warehouses that contain a company's business data including information on its customers
- They act as a repository for most or all organizational data to facilitate broad access and analytics

https://www.snowflake.com/guides/what-enterprise-data-warehouse

Enterprise-wide data warehouses (EDW) - ETL vs ELT

- EDWs make data viewable and actionable in real-time by favoring an extract-load-transform (ELT) approach over the once common extract-transform-load (ETL) paradigm
  - ETL - data was cleansed, transformed, or enriched on an external server prior to being loaded into the data warehouse.
  - ELT - raw data is extracted from its source and loaded, relatively unchanged, into the data warehouse, making it much faster to access and analyze.

ETL Processing Happening in a DW

- Extraction and loading happen periodically - sometimes daily, weekly, or monthly.
- DW does not have current data
- DW do not support operational transaction processing, they may contain transactional data, but more often summaries of transactions and snapshots of status variables
- In most DW applications, users are looking for trends and patterns in the state of the organization across a large subset of the DW.

Enterprise-wide data warehouses (EDW)

- Business needs that require EDW
  - Real-time access to data for action
  - Holistic understanding of customer
  - Tracking and ensuring data compliance
    - EDWs enable data customers to audit and vet data sources directly and find errors quickly.
    - A modern EDW can also enable compliance with the EU's General Data Protection Regulation (GDPR) without implementing an involved process to check multiple data locations.
  - Empowering users with limited technical knowledge
    - An EDW benefits non-technical employees in job functions beyond marketing, finance, and the supply chain.
  - Consolidating data to a single, reliable repository
    - Modern data warehousing technology enables companies to store data across different regions and cloud providers.
    - Users can query an EDW as though it were a global unified data set (i.e., single version of the truth).

DW vs EDW

- An EDW is a data warehouse that encompasses and stores all of an organization’s data from sources across the entire business.
- A smaller data warehouse may be specific to a business department or line of business (like a data mart).
- In contrast, an EDW is intended to be a single repository for all of an organization’s data.

Key Components of an Enterprise Data Warehouse

- Data Sources
  - These encompass data collected from various operational and transactional systems within the organization
    - Enterprise Resource Planning (ERP) systems
    - Customer Relationship Management (CRM) platforms
    - Finance applications
    - Internet of Things (IoT) devices, and mobile and online systems.
- Staging Area
  - Serves as an intermediate space where data is aggregated, cleaned, and prepared before being loaded into the EDW.
  - The staging area ensures that data is in a consistent and usable format, ready for further processing and analysis.

Key Components of an Enterprise Data Warehouse

- Presentation or Access Space
  - This component provides an interface for users to access and interact with the data stored in the EDW.
  - It enables analytics, querying, reporting, and data sharing, facilitating efficient decision-making and strategic insights across the organization.
- Data Tool Integrations or APIs
  - An EDW often integrates with a variety of data tools and software to enhance its functionality.
  - These may include Business Intelligence (BI) software, data ingestion tools, ETL tools, and other APIs that enable seamless data integration and processing within the EDW.

Virtual data warehouses

- A virtual warehouse refers to a collection of practical database views. It is simple to set up, but it necessitates more capacity for operating database servers.
- Provide views of operational databases that are materialized for efficient access
- Another term for the compute clusters that power the modern data warehouse, acting as an on-demand resource
- It is an independent compute resource that can be leveraged at any time for,
  - SQL execution for short and long running queries
  - DML (Data Manipulation Language) operations
    - Updating rows in tables (INSERT, UPDATE, DELETE)
  - Loading data into tables
  - Unloading data from tables
- Finally, can be turned off when it isn't needed.

https://bytehouse.cloud/blog/what-is-virtual-warehouse

Virtual data warehouses (Contd.)

- For decades, traditional on-premise data warehouses have tightly coupled data storage and compute, making it a challenge to scale either on-demand
- However, today's businesses need to store and analyze vast quantities of both structured and unstructured data from disparate services, necessitating a service that can respond to large data volumes and variable compute needs for applications such as visualization.

Advantages of Virtual Data Warehouses

- Read-write separation and resource separation
  - Separate Virtual Warehouses can be created to perform data-loading tasks, or run queries in real-time.
  - Creating Virtual Warehouses also allows different departments or business lines to have clear separation of resources, avoiding potential impact on performances between different queries.
- One-click scaling
  - Virtual Warehouses can easily scale up or down according to business requirements for cost-effective resource utilisation.
- Access Control
  - Virtual warehouses can also be granted different permissions into various roles.
  - Thus, users can achieve fine-grained access control.

Logical Data Warehouses

- A logical data warehouse (LDW) is an architecture layer that rests on top of data warehouse persisted data and allows viewing data without transformation or movement.

- A logical data warehouse set-up allows analysts and other business users to access fresh data without formatting and eliminates the need to transform and consolidate data from disparate sources in order to view it.

https://www.tibco.com/reference-center/what-is-a-logical-data-warehouse

https://www.snowflake.com/trending/logical-data-warehouse/

Data marts

- Data marts are targeted for a subset of the organization, such as a department, and are more tightly focused.
  - Data marts make specific data available to a defined group of users, which allows those users to quickly access critical insights without wasting time searching through an entire data warehouse.
- Data marts are limited to a few chosen functions
  - E.g., Marketing, finance, sales data marts
- A data warehouse that is limited in scope
- Data mart's content are either obtained from independent ETL process (for an independent data mart) or are derived from the data warehouse

Data marts (Contd.)

- It is possible that each data mart is built using different tools
  - Financial data mart may be built using Hypersion's Essbase while sales data mart may be built on a more general-purpose data warehouse platform such as Teradata.
- They introduce complexity for end users when they need to access data in separate data marts
  - This complexity comes not only from having to access data from separate data mart databases but also from possibly a new generation of inconsistent data systems

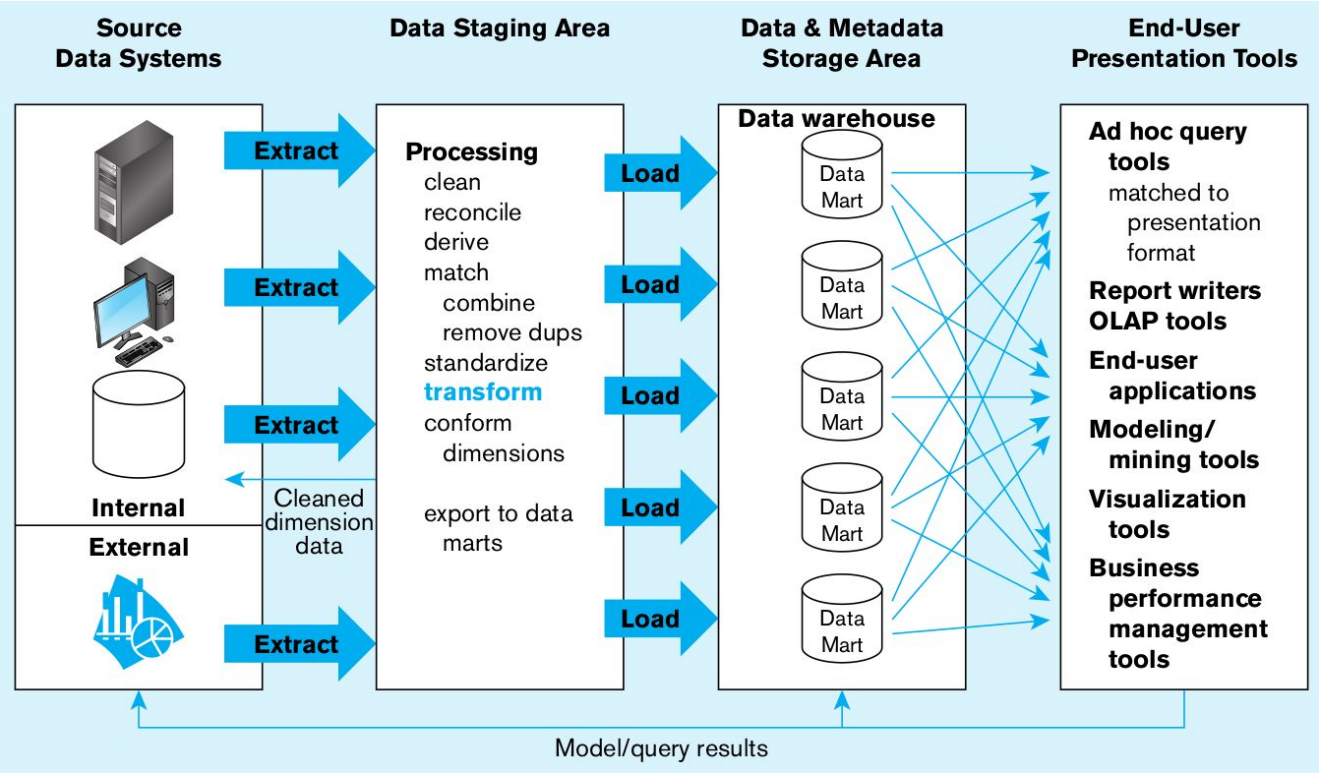
Data Mart - Types

- Independent data marts
- Dependent data marts
- Hybrid data marts

Data Mart - Types (Contd.)

- **Independent data marts**
  - These act as a standalone system that doesn't rely on a data warehouse.
  - Analysts can extract data on a particular subject or business process from internal or external data sources, process it, and then store it in a data mart repository until the team needs it.

Independent Data Mart Data Warehousing Architecture



Independent Data Mart Data Warehousing Environment

Building independent data mart data warehousing architecture requires four basic steps

1. Data are extracted from the various internal and external source system files and databases.
2. The data from the various source systems are transformed and integrated before being loaded into the data marts. Transactions may be sent to the source systems to correct errors discovered in data staging. The data warehouse is considered to be the collection of data marts.

Independent Data Mart Data Warehousing Environment (contd.)

3. The data warehouse is a set of physically distinct databases organized for decision support. It contains both detailed and summary data.
4. Users access the data warehouse by means of a variety of query languages and analytical tools. Results (e.g., predictions, forecasts) may be fed back to data warehouse and operational databases.

Why Independent Data Marts?

- Independent data marts are often created because an organization focuses on a series of short-term, expedient business objectives.
- The limited-short term objectives promote implementing comparably lower cost, independent data mart.
  - Keep in mind that you may lose flexibility for the long term and the ability to react to changing business conditions
- It can be organizationally and politically easier to have separate, small data warehouses than to get all organizational parties to agree to one view of the organization in a central data warehouse.
- Some data warehousing technologies have technical limitations for the size of the data warehouse they can support
- Hence the technology, rather than the business, may dictate a data warehousing architecture

Independent Data Mart Limitations

1. A separate ETL process is developed for each data mart, which can yield costly redundant data and processing efforts.
2. Data marts may not be consistent with one another, they may not provide a clear enterprise-wide view of data concerning important subjects such as customers, suppliers, and products.
3. There is no capability to drill down into greater detail or into related facts in other data marts or a shared data repository, so analysis is limited, or at best very difficult

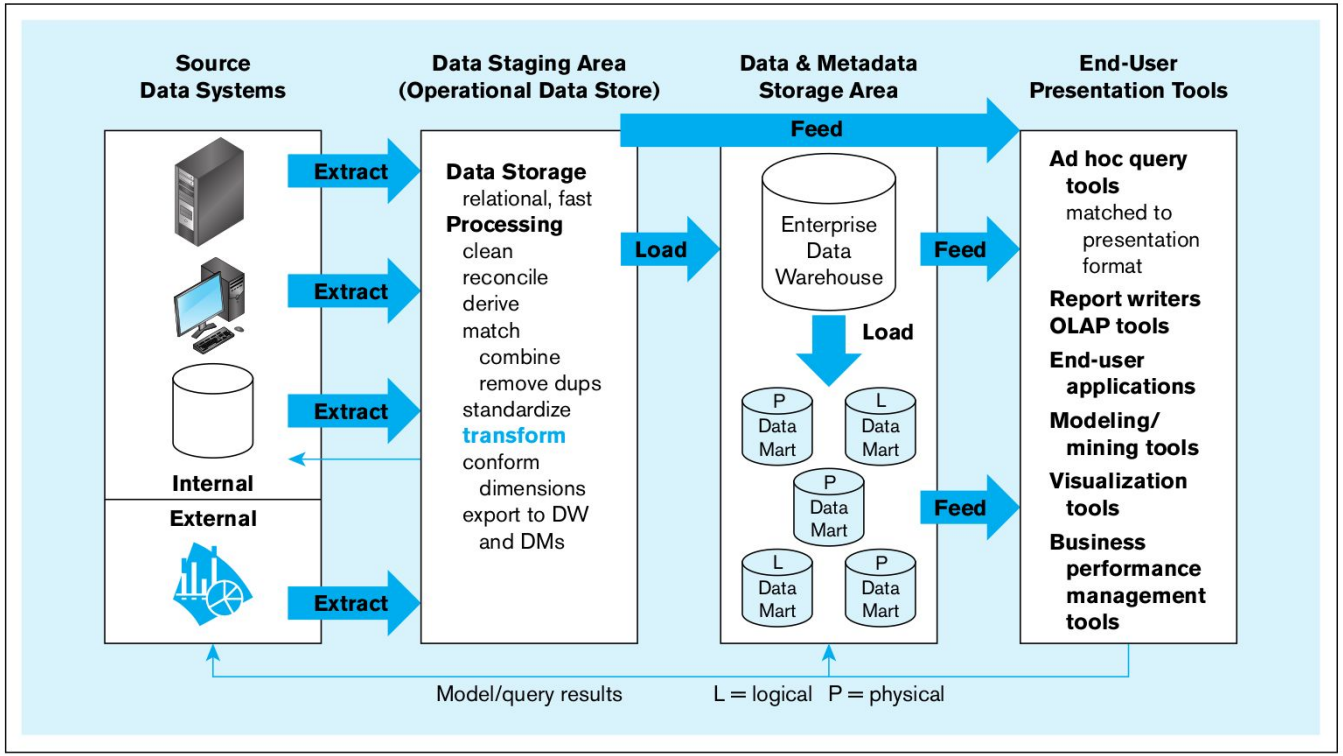
Independent Data Mart Limitations

4. Scaling costs are excessive because every new application that creates a separate data mart repeats all the extract and load steps.
    a. Usually, operational systems have limited time windows for batch data extracting, so at some point, the load on the operations system may require additional technologies to be used with added cost.
5. If there is an attempt to make the separate data marts consistent, the cost to do so is quite high.

Data Mart - Types

- **Dependent data marts**
  - They are partitioned segments within an enterprise data warehouse.
  - This top-down approach begins with the storage of all business data in one central location.
  - The newly created data marts extract a defined subset of the primary data whenever required for analysis.
  - Dependent data marts still have a purpose to provide a simplified and high-performance environment that is tuned to the decision-making needs of user groups.

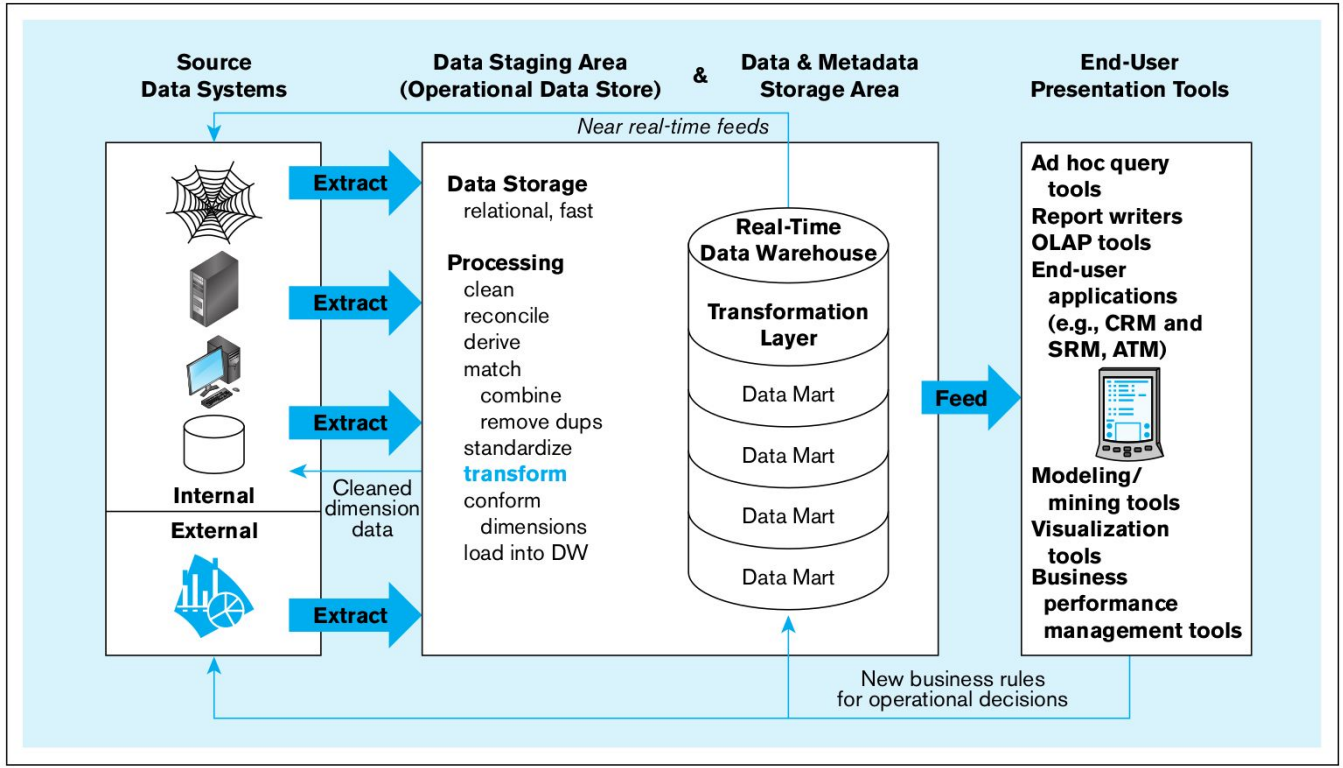
Dependent Data Mart and Operational Data Store : A three-level Architecture



Dependent Data Marts (Contd.)

- One of the most popular approaches to addressing the independent data mart limitations is to use the dependent data mart and operational data store architecture
- The first two limitations are addressed by loading the dependent data marts from an enterprise data warehouse (EDW)
- A user group can access its data mart, and then when other data are needed, users can access the EDW
- The dependent data mart and operational data store architecture is often called a “hub and spoke” approach, in which the EDW is the hub and the source data systems and the data marts are at the ends of input and output spokes.

Logical Data Mart and Real-Time Data Warehouse Architecture



Logical Data Mart and Real-Time Data Warehouse Architecture

- The logical data mart and real-time data warehouse architecture is practical for only moderate-sized data warehouses or when using high-performance data warehousing technology, such as the Teradata system.
    1. Logical data marts are not physically separate databases but rather different relational views of one physical, slightly denormalized relational data warehouse.
    2. Data are moved into the data warehouse rather than to a separate staging area to utilize the high-performance computing power of the warehouse technology to perform the cleansing and transformation steps.

Logical Data Mart and Real-Time Data Warehouse Architecture (Contd.)

3. New data marts can be created quickly because no physical database or database technology needs to be created or acquired and no loading routines need to be written.

4. Data marts are always up to date because data in a view are created when the view is referenced; views can be materialized if a user has a series of queries and analysis that need to work off the same instantiation of the data mart.

Real-time Data Warehouse

- The source data systems, decision support services, and the data warehouse exchange data and business rules at a near-real-time pace because there is a need for rapid response (i.e., action) to a current, comprehensive picture of the organization.
- The purpose of real-time data warehousing is to know what is happening, when it is happening, and to make desirable things happen through the operational systems.
  - For example, a help desk professional answering questions and logging problem tickets will have a total picture of the customer's most recent interactions.
- Real-time data warehousing moves data warehousing from the back office to the front office

Data Mart - Types (Contd.)

- **Hybrid data marts**
  - These combine data from existing data warehouses and other operational sources.
  - This unified approach leverages the speed and user-friendly interface of a top-down approach and also offers the enterprise-level integration of the independent method.

Advantages of Data Marts

- Cost-efficiency
  - Data marts typically costs only fraction of the cost of a data warehouse
- Simplified data access
  - Data marts only hold a small subset of data, so users can quickly retrieve the data they need with less work than they could when working with a broader data set from a data warehouse.
- Quicker access to insights
  - Intuition gained from a data warehouse supports strategic decision-making at the enterprise level, which impacts the entire business.
  - A data mart fuels business intelligence and analytics that guide decisions at the department level.
  - Teams can leverage focused data insights with their specific goals in mind. As teams identify and extract valuable data in a shorter space of time, the enterprise benefits from accelerated business processes and higher productivity.

https://www.ibm.com/topics/data-mart

Advantages of Data Marts (Contd.)

- Simpler data maintenance
  - A data warehouse holds a wealth of business information, with scope for multiple lines of business.
  - Data marts focus on a single line, housing under 100GB, which leads to less clutter and easier maintenance.
- Easier and faster implementation
  - A data warehouse involves significant implementation time, especially in a large enterprise, as it collects data from a host of internal and external sources.
  - On the other hand, you only need a small subset of data when setting up a data mart, so implementation tends to be more efficient and include less set-up time.
- Creating agile and scalable data management
  - Data marts provide an agile data management system that works in tandem with business needs, including being able to use information gathered in past projects to help with current tasks.
  - Teams can update and change their data mart based on new and evolving analytics project

Data Warehouse vs Data Mart

Data Warehouse	Data Mart
Scope	**Scope**
• Application independent • Centralized, possibly enterprise-wide • Planned	• Specific DSS application • Decentralized by user area • Organic, possibly not planned
Data	**Data**
• Historical, detailed, and summarized • Lightly denormalized	• Some history, detailed, and summarized • Highly denormalized
Subjects	**Subjects**
• Multiple subjects	• One central subject of concern to users
Sources	**Sources**
• Many internal and external sources	• Few internal and external sources
Other Characteristics	**Other Characteristics**
• Flexible • Data oriented • Long life • Large • Single complex structure	• Restrictive • Project oriented • Short life • Starts small, becomes large • Multi, semi-complex structures, together complex

Other Common Concepts with Data Warehouses

- Operational Data Store (ODS)
  - This is a commonly used term for intermediate form of databases before they are cleansed, aggregated, and transformed into a warehouse
- Analytical Data Store (ADS)
  - These are databases built for analysis
  - Usually ODS's are re-configured and repurposed into ADS's

Other Common Concepts with Data Warehouses (Contd.)

- With high-performance computers and data warehousing technologies, there may not be a need for a separate ODS from the enterprise data warehouse.
- When the ODS and EDW are one and the same, it is much easier for users to drill down and drill up when working through a series of ad hoc questions in which one question leads to another.
- It is also a simpler architecture, because one layer of the dependent data mart and operational data store architecture has been eliminated.

Issues in Data Warehousing

- Warehouse Design
- Extraction
  - Monitors (Change detectors), wrappers
- Integration
  - Cleansing and merging
- Warehousing specification and maintenance
- Optimizations
- Miscellaneous (e.g., Evolution)

Data Warehousing - Two Distinct Issues

1. How to get information into warehouse
   - Data Warehousing
2. What to do with data once it is in warehouse
   - Warehouse DBMS

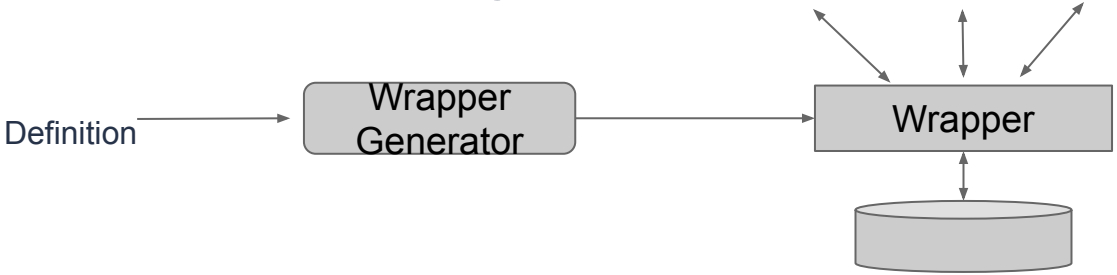
Data Extraction

- Source types
  - Relational, flat file, weather sensors, WWW, etc.
- How to get data out?
  - Replication tool
  - Dump file
  - Create report
  - ODBC or third-party "wrappers"

Wrapper

- Converts data and queries from one data model to another
- Extends query capabilities for sources with limited capabilities

- Wrapper Generation
  a. Hardcode for each source
  b. Automatic wrapper generation



Data Transformations

- Convert data to uniform format
  - Byte ordering, string termination
  - Internal layout
- Remove, add & reorder attributes
  - Add key
  - Add data to get history
- Sort tuples

Monitors

- Detects changes of interest and propagates to integrator
- How are the monitors implemented?
  - Triggers
  - Replication server
  - Log sniffer
  - Compare query results
  - Compare snapshots/dumps

Data Integration

- Receive data (changes) from multiple wrappers/monitors and integrate into warehouse
- Integrator is typically implemented as a rule-based engine
- Actions
  - Resolve inconsistencies
  - Eliminate duplicates
  - Integrate into warehouse (may not be empty)
  - Summarize data
  - Fetch more data from sources (wh updates)
  - etc.

Data Integration (Contd.)

- Each rule is responsible for handling one kind of change noti cation, and is implemented as an object-oriented method
- The method is called whenever a monitor generates a change notification of the appropriate type.
- The method body then performs the necessary processing to integrate the change into the warehouse.
- During this processing, the method may need to obtain additional data from the warehouse, or from the same or other sources.

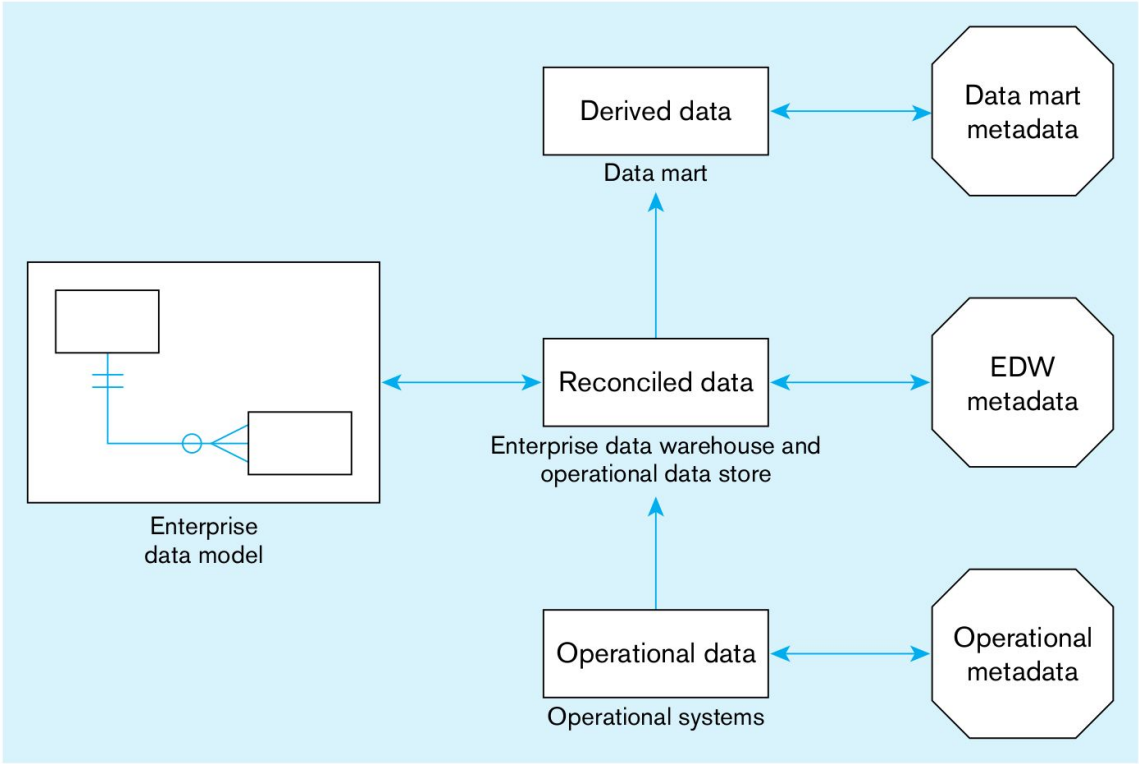
Data Integration (Contd.)

- Suppose the warehouse keeps the average salary of employees at a company.
- When an update to an employee salary is reported, the update rule will have to obtain the current average from the warehouse in order to compute the new value.

Data Cleansing

- Find (& remove) duplicate tuples
  - e.g., Jane Doe vs. Jane Q. Doe
- Detect inconsistent, wrong data
  - Attribute values that don’t match
- Patch missing, unreadable data
- Notify sources of errors found

Three-Layer Data Architecture



Three-Layer Data Architecture (Contd.)

Three-layer data architecture is characterized by the following,

1. **Operational data** are stored in the various operational systems of record throughout the organization (and sometimes in external systems).
2. **Reconciled data** are the type of data stored in the enterprise data warehouse and an operational data store. Reconciled data are detailed, current data intended to be the single, authoritative source for all decision support applications.
3. **Derived data** are the type of data stored in each of the data marts. Derived data are data that have been selected, formatted, and aggregated for end-user decision support applications.

Role of the Enterprise Data Model

- The enterprise data model presents a total picture explaining the data required by an organization. If the reconciled data layer is to be the single, authoritative source for all data required for decision support, it must conform to the design specified in the enterprise data model.

- Usually the enterprise data model evolves as new problems and decision applications are addressed. It takes too long to develop the enterprise data.

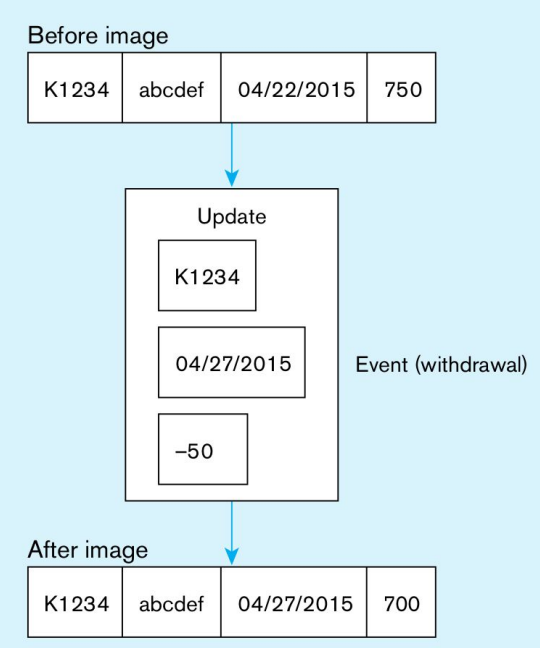
- model in one step, and the dynamic needs for decision making will change before the warehouse is built.

Role of MetaData

- Metadata are technical and business data that describe the properties or characteristics of other data,
  a. Operational metadata - describe the data in the various operational systems (as well as external data) that feed the enterprise data warehouse. Operational metadata typically exist in a number of different formats and unfortunately are often of poor quality.
  b. Enterprise data warehouse (EDW) metadata - derived from the enterprise data model. EDW metadata describe the reconciled data layer as well as the rules for extracting, transforming, and loading operational data into reconciled data.
  c. Data mart metadata - describe the derived data layer and the rules for transforming reconciled data to derived data.

Characteristics of Data Warehouse Data

Both status data and event data can be stored in a database. However, in practice, most of the data stored in databases (including data warehouses) are status data.



Transient Data

- Transient data are data in which changes to existing records are written over previous records, thus destroying the previous data content.

Table X (10/09)

Key	A	B
001	a	b
002	c	d
003	e	f
004	g	h

Table X (10/10)

Key	A	B
001	a	b
002	r	d
003	e	f
004	y	h
005	m	n

Table X (10/11)

Key	A	B
001	a	b
002	r	d
003	e	t
005	m	n

Periodic Data

- Periodic data are data that are never physically altered or deleted once added to the store

Storing periodic data can impose large storage requirements. Therefore, users must choose very carefully the key data that require this form of processing.

Table X (10/09)

Key	Date	A	B	Action
001	10/09	a	b	C
002	10/09	c	d	C
003	10/09	e	f	C
004	10/09	g	h	C

Table X (10/10)

Key	Date	A	B	Action
001	10/09	a	b	C
002	10/09	c	d	C
002	10/10	r	d	U
003	10/09	e	f	C
004	10/09	g	h	C
004	10/10	y	h	U
005	10/10	m	n	C

Table X (10/11)

Key	Date	A	B	Action
001	10/09	a	b	C
002	10/09	c	d	C
002	10/10	r	d	U
003	10/09	e	f	C
003	10/11	e	t	U
004	10/09	g	h	C
004	10/10	y	h	U
004	10/11	y	h	D
005	10/10	m	n	C

Other Data Warehouse Changes

Six other kinds of changes to a warehouse data model must be accommodated by data warehousing.

1. **New descriptive attributes** - new characteristics of products or customers that are important to store in the warehouse must be accommodated.
2. **New business activity attributes** - new characteristics of an event already stored in the warehouse
3. **New classes of descriptive attributes** - This is equivalent to adding new tables to the database.
4. **Descriptive attributes become more refined** - data about stores must be broken down by individual cash register to understand sales data

Other Data Warehouse Changes (Contd.)

5. **Descriptive data are related to one another** - For example, store data are related to geography data. This causes new relationships, often hierarchical, to be included in the data model.

6. **New source of data** - some new business need causes data feeds from an additional source system or some new operational system is installed that must feed the warehouse. This change can cause almost any of the previously mentioned changes, as well as the need for new extract, transform, and load processes.

Derived Data Layer

- This is the data layer associated with logical or physical data marts
- It is the layer with which users normally interact for their decision support applications.
- Ideally, the reconciled data level is designed first and is the basis for the derived layer, whether data marts are dependent, independent, or logical.
- In order to derive any data mart we might need, it is necessary that the EDW be a fully normalized relational database accommodating transient and periodic data; this gives us the greatest flexibility to combine data into the simplest form for all user needs, even those that are unanticipated when the EDW is designed.

Characteristics of Derived Data

- Derived data are data that have been selected, formatted, and aggregated for end-user decision support applications
  - I.e., derived data are information instead of raw data

- The source of the derived data is the reconciled data, created from what can be a rather complex data process to integrate and make consistent data from many systems of record inside and outside the organization.

- Derived data in a data mart are generally optimized for the needs of particular user groups, such as departments, workgroups, or even individuals, to measure and analyze business activities and trends.

Objectives that are sought with derived data

- Provide ease of use for decision support applications
- Provide fast response for predefined user queries or requests for information
- Customize data for particular target user groups
- Support ad hoc queries and data mining and other analytical applications

Data Modeling for Data Warehouses

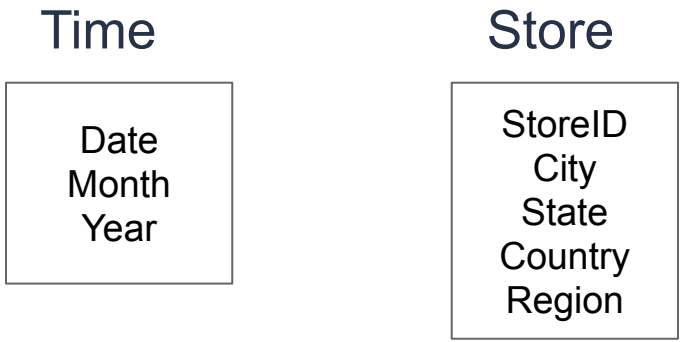
- Traditional Databases generally deal with two-dimensional data (similar to a spreadsheet).
  - However, querying performance in a multi-dimensional data storage model (matrices) is much more efficient.
- Data warehouses can take advantage of this feature as generally these are
  - Non volatile
  - The degree of predictability of the analysis that will be performed on them is high.
- Typical Dimensions used in corporate DWs:
  - Fiscal Periods, Product Categories, Geographic Regions

Multi-Dimensional Data

- **Measures** - numerical (and additive) data being tracked in business, can be analyzed and examined
- **Dimensions** - business parameters that define a transaction, relatively static data such as lookup or reference tables
- Example: Analyst may want to view sales data (measure) by geography, by time, and by product (dimensions)

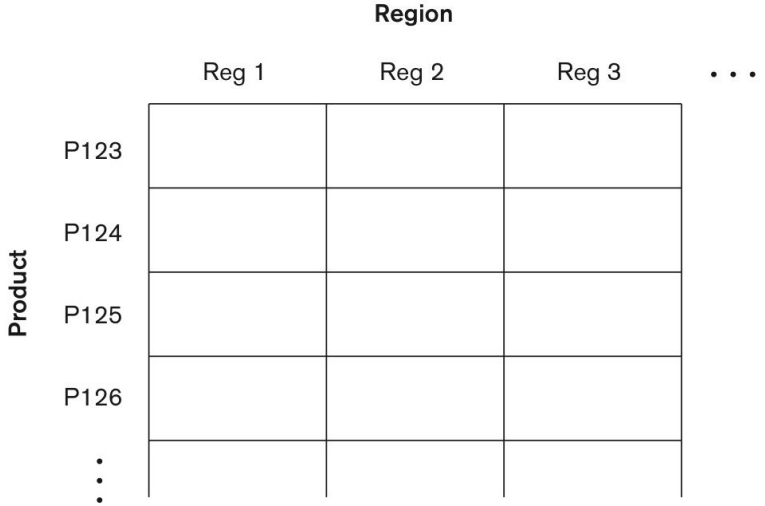
Multi-Dimensional Data (Contd.)

- Dimensions are organized into hierarchies
  - E.g., Time dimension: days, weeks, quarters, etc.
  - E.g., Product dimension: product, product line, brand, etc.
- Dimensions have attributes

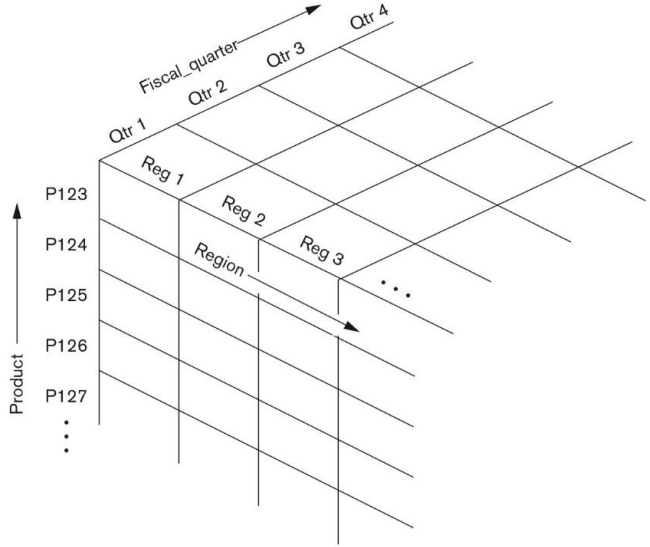


Data Modeling for Data Warehouses (Contd.)

- Two-dimensional vs Multi-dimensional data models



A two-dimensional data model



A three-dimensional (Cube) data model

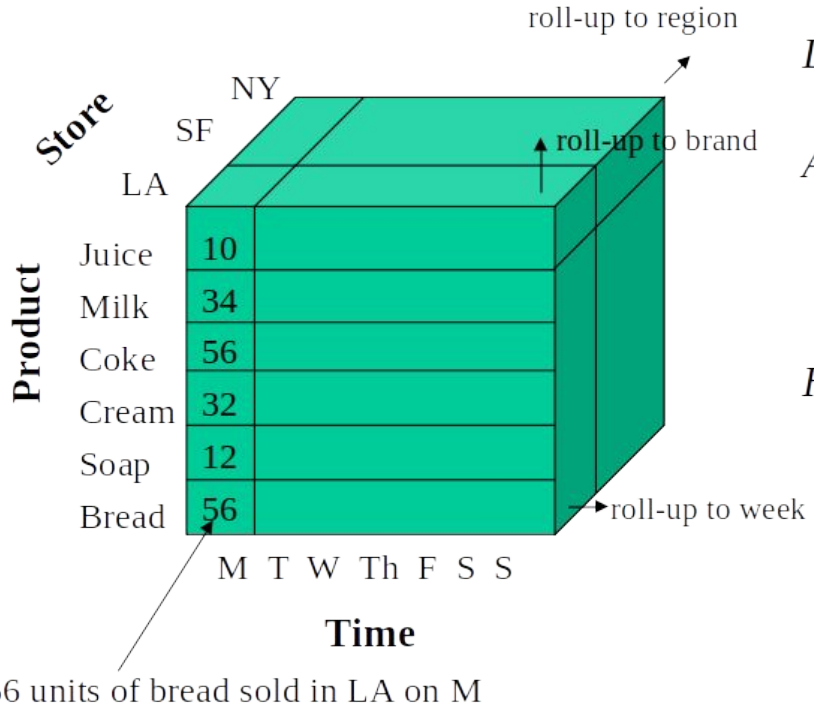
Functionality of a Data Warehouse

- **Pivot** : Cross tabulation (also referred to as rotation) is performed
- **Roll-up (also Drill-up)** : Data is summarized with increasing generalization (for example, weekly to quarterly to annually).
- **Drill-down** : Increasing levels of detail are revealed (the complement of roll-up).
- **Slice and dice** : Projection operations are performed on the dimensions.
- **Sorting** : Data is sorted by ordinal value.

Functionality of a Data Warehouse (Contd.)

- **Selection** : Data is filtered by value or range.
- **Derived (computed) attributes** : Attributes are computed by operations on stored and derived values.

Functionality of a Data Warehouse : Example



Dimensions:

Time, Product, Store

Attributes:

Product (upc, price, …)

Store …

…

Hierarchies:

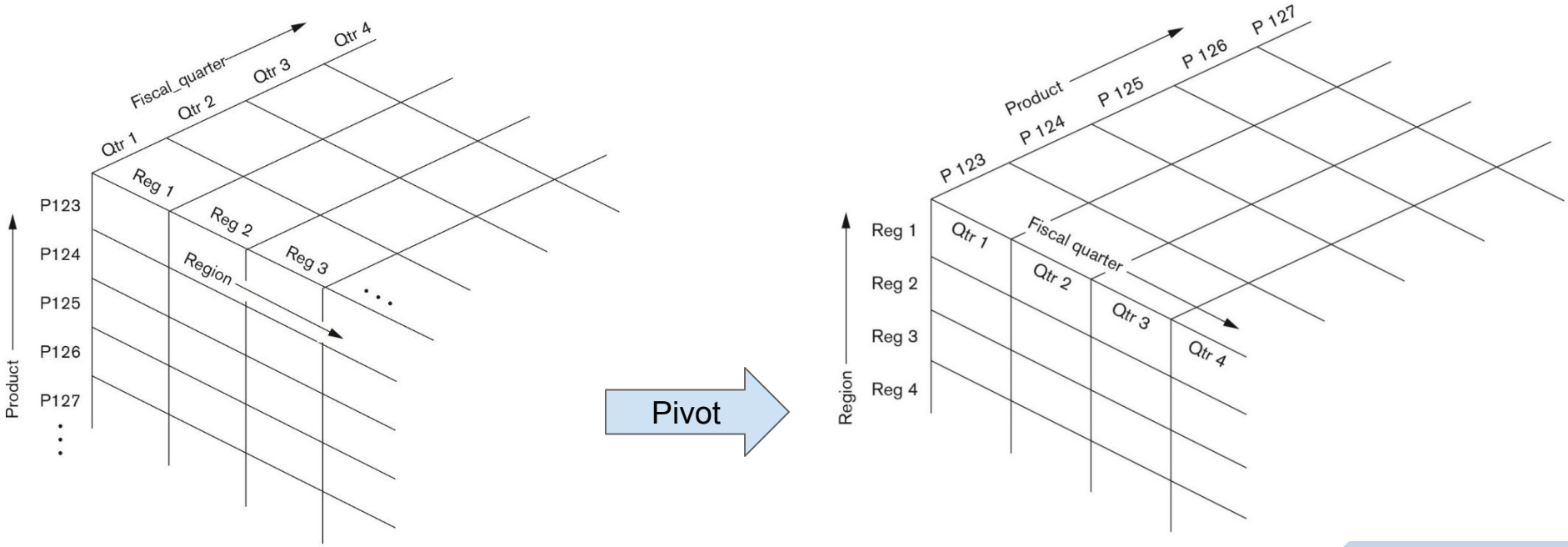
Product → Brand → …

Day → Week → Quarter

Store → Region → Country

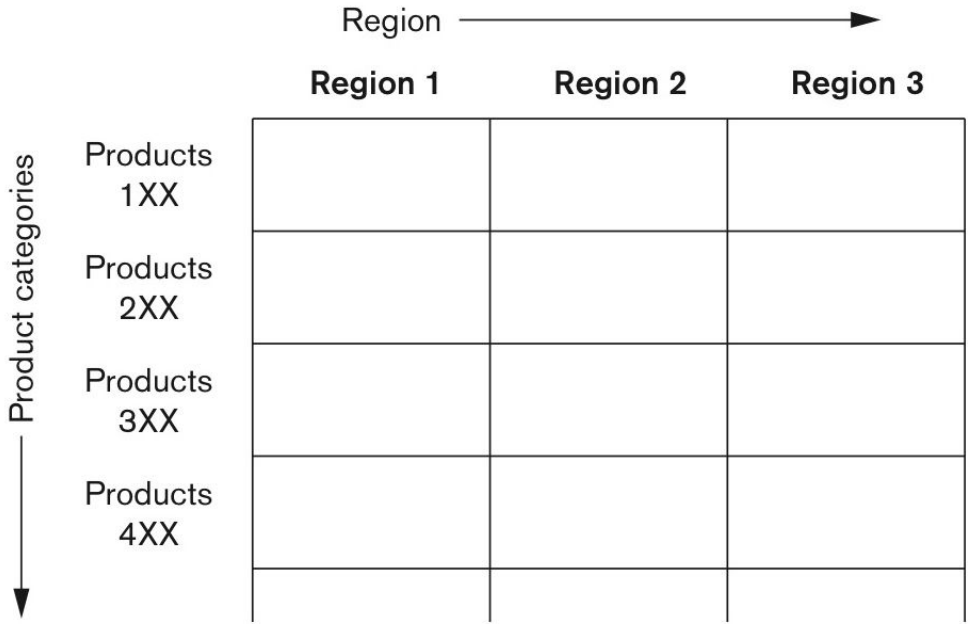
http://www.cise.ufl.edu/~jhammer/classes/6930/Fall98/lectures.html

Functionality of a Data Warehouse : Pivot operation



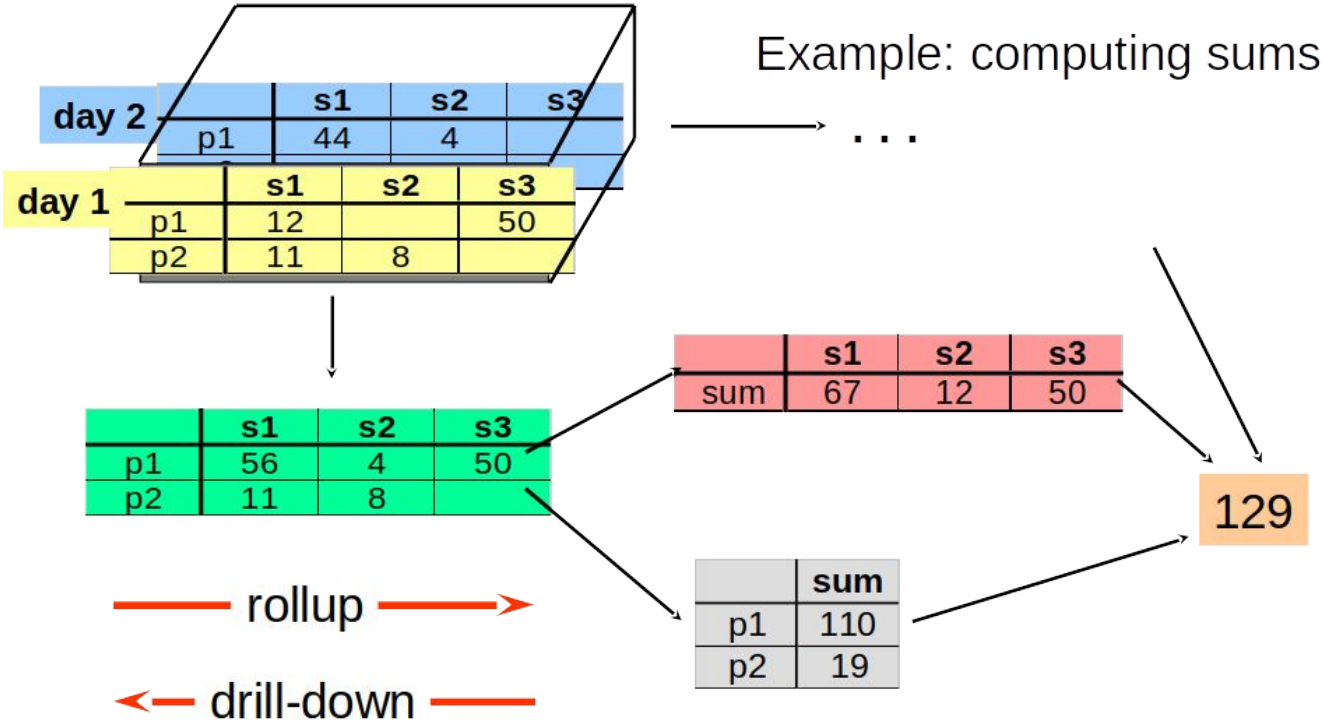
This presents data in terms of Region vs. Fiscal Quarter : product by product.
“Pivoting” is also called as “Rotation”

Functionality of a Data Warehouse : Roll-up operation



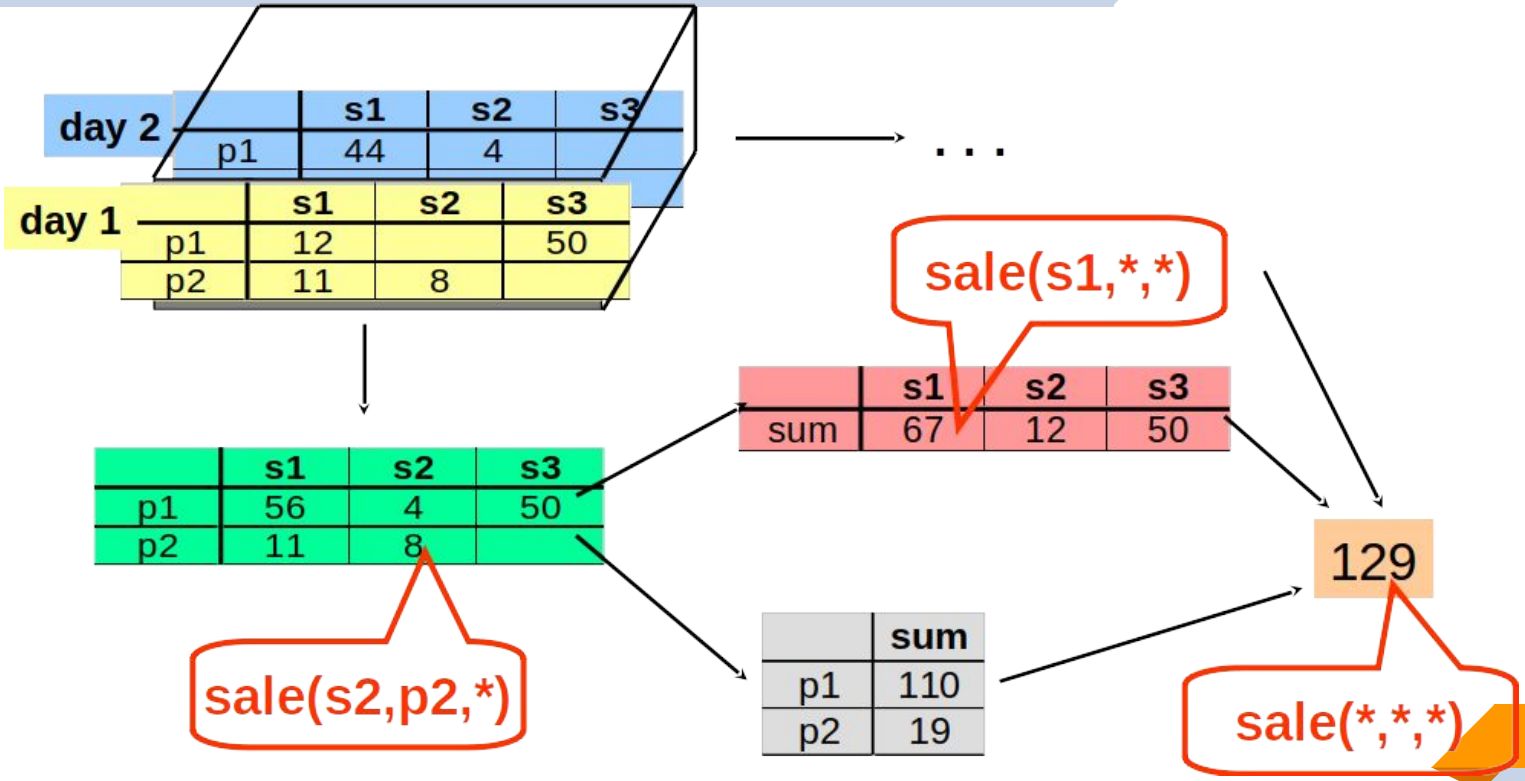
The roll-up in above figure aggregates data for all products numbered 123, 124, …… into 1XX, etc.

Functionality of a Data Warehouse : Cube Aggregation Roll-up operation (Contd.)

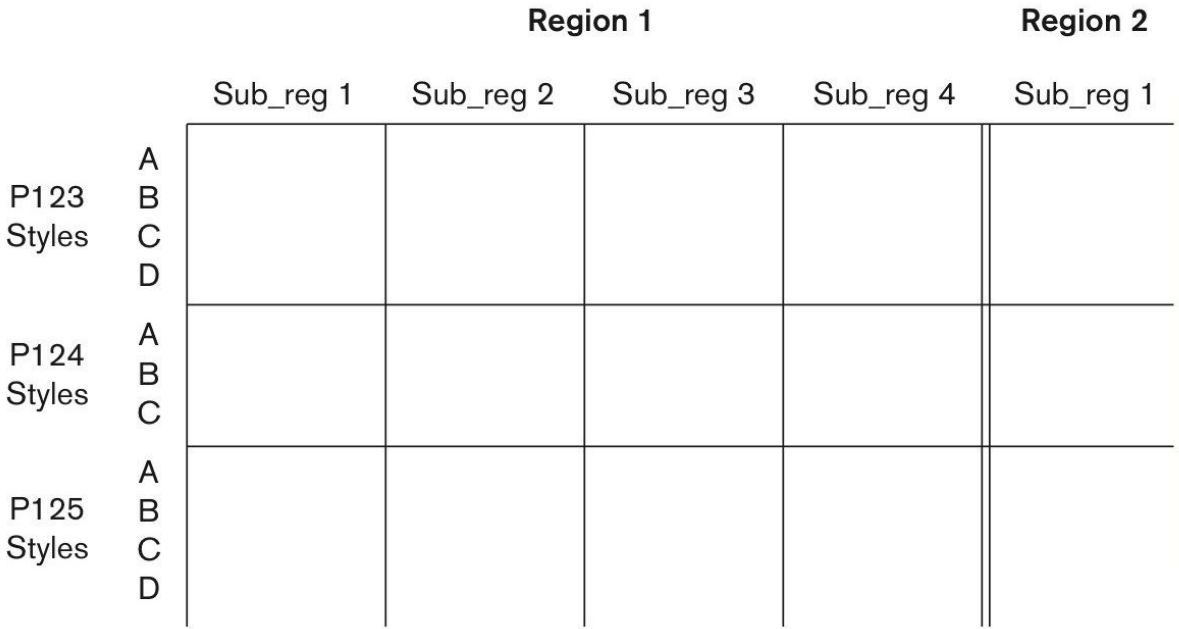


http://www.cise.ufl.edu/~jhammer/classes/6930/Fall98/lectures.html

Functionality of a Data Warehouse : Cube Operators for Roll-up (Contd.)



Functionality of a Data Warehouse : Drill-down operation



A drill-down expands aggregate data into a finer grained view. In this example , products are divided into styles and regions into sub-regions.

Functionality of a Data Warehouse : Drill-down operation - Another example

a) Summary Report

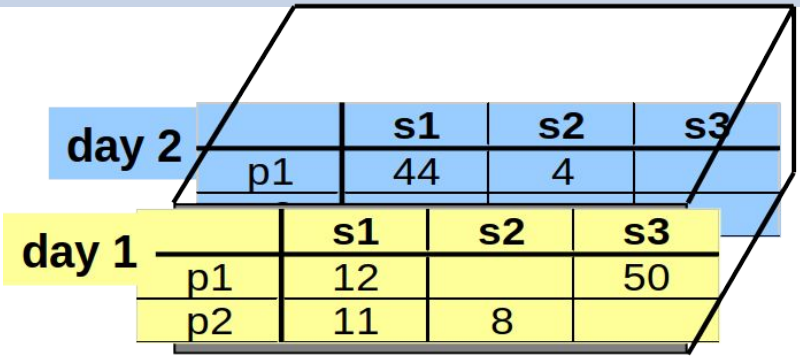
Brand	Package size	Sales
SofTowel	2-pack	$75
SofTowel	3-pack	$100
SofTowel	6-pack	$50

b) Drill-down with color attribute added

Brand	Package size	Color	Sales
SofTowel	2-pack	White	$30
SofTowel	2-pack	Yellow	$25
SofTowel	2-pack	Pink	$20
SofTowel	3-pack	White	$50
SofTowel	3-pack	Green	$25
SofTowel	3-pack	Yellow	$25
SofTowel	6-pack	White	$30
SofTowel	6-pack	Yellow	$20

Starting with summary data, users can obtain details for particular cells

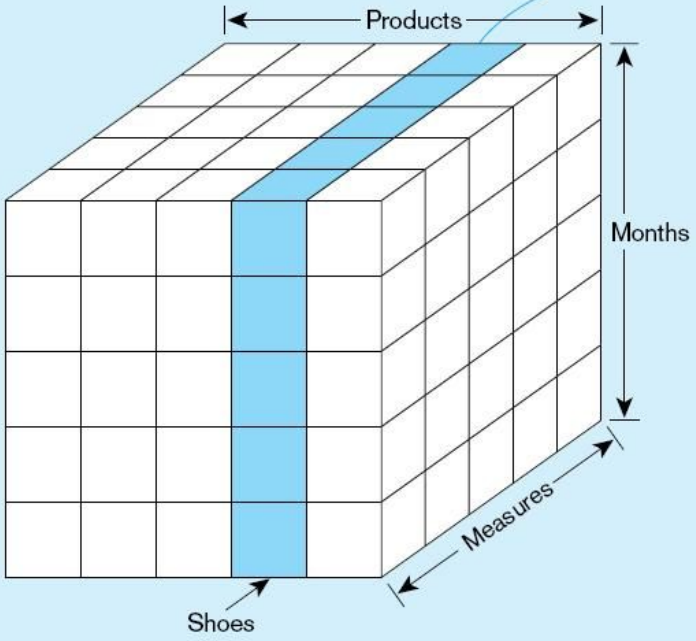
Functionality of a Data Warehouse : Slicing



TIME = day 1

	s1	s2	s3
p1	12		50
p2	11	8	

Functionality of a Data Warehouse : Slicing - another example



	Measure		
	Units	Revenue	Cost
January	250	1564	1020
February	200	1275	875
March	350	1800	1275
April	400	1935	1500
May	485	2000	1560

Product: Shoes

Functionality of a Data Warehouse : Slicing and Pivoting

	Sales ($ millions)			
	Products	Time		
		d1	d2	
Store s1	Electronics	$5.2		
	Toys	$1.9		
	Clothing	$2.3		
	Cosmetics	$1.1		
Store s2	Electronics	$8.9		
	Toys	$0.75		
	Clothing	$4.6		
	Cosmetics	$1.5		

	Sales ($ millions)			
	Products	d1		
		Store s1	Store s2	
Store s1	Electronics	$5.2	$8.9	
	Toys	$1.9	$0.75	
	Clothing	$2.3	$4.6	
	Cosmetics	$1.1	$1.5	
Store s2	Electronics			
	Toys			
	Clothing			

Multi-dimensional Schemas

- Multi-dimensional model (also called ”dimensional model”) includes two types of tables:
  - **Dimension table**
    - It consists of tuples of attributes of the dimension.
  - **Fact table**
    - Each tuple is a recorded fact. This fact contains some measured or observed variable (s) and identifies it with pointers to dimension tables. The fact table contains the data, and the dimensions to identify each tuple in the data.
    - A fact table is as an agglomerated view of transaction data whereas each dimension table represents “master data” that those transactions belonged to.

Multi-dimensional Schemas (Contd.)

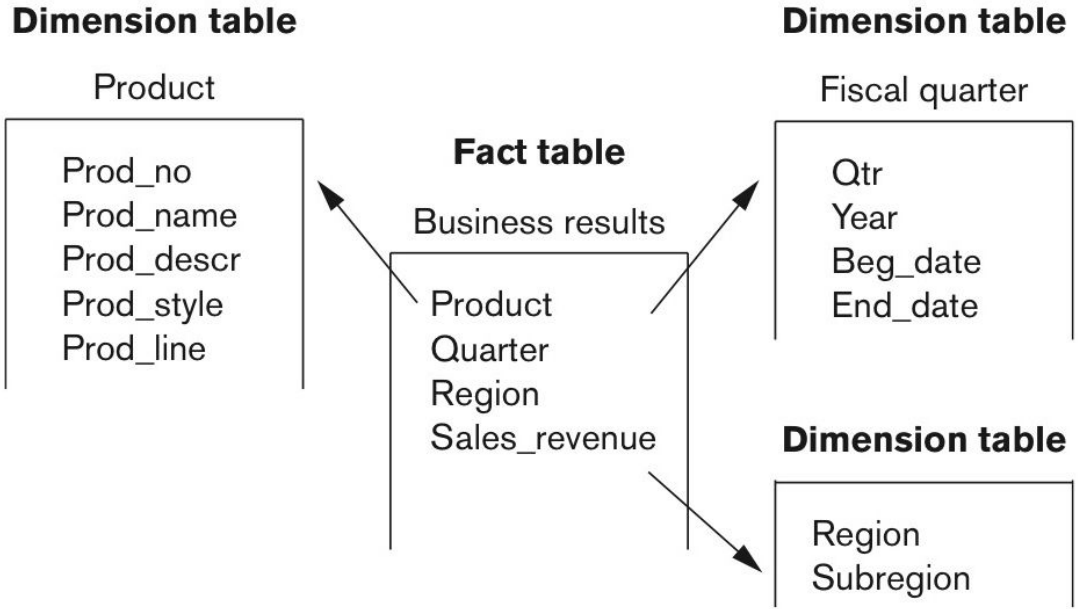
- Multidimensional DW systems implemented the multidimensional model as is.
- The more popular approach is to implement the multidimensional model on top of the relational model.
- Two common multi-dimensional schemas are
  - **Star schema**
    - Consists of a fact table with a single table for each dimension
  - **Snowflake Schema**
    - It is a variation of star schema, in which the dimensional tables from a star schema are organized into a hierarchy by normalizing them.

Star Schema

- A star schema is a simple database design (particularly suited to ad hoc queries) in which dimensional data are separated from fact or event data
- A star schema is one version of a dimensional model
- Although the star schema is suited to ad hoc queries, it is not suited to online transaction processing
  - Generally not used in operational systems, operational data stores, or an EDW

Star Schema

- Consists of a fact table with a single table for each dimension



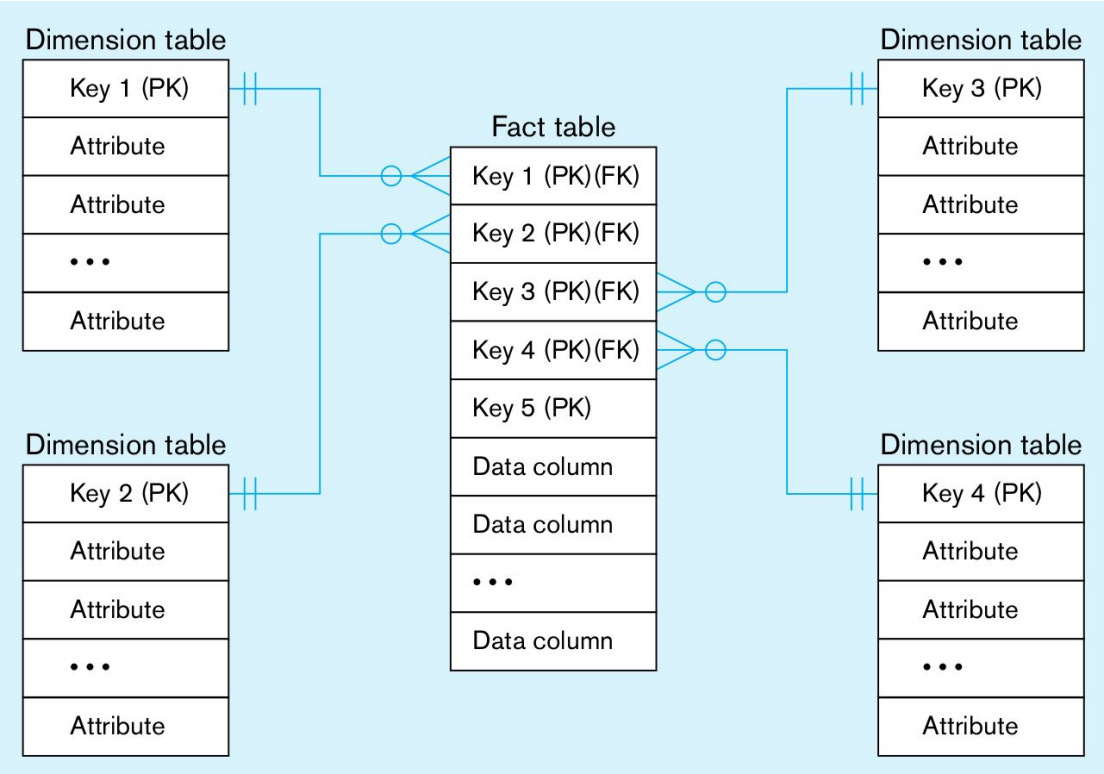
Star Schema (Contd.)

- Fact tables contain factual or quantitative data
  - Measurements that are numerical, continuously valued, and additive
  - About a business, such as units sold, orders booked, and so on
- Dimension tables hold descriptive data (context) about the subjects of the business.
  - The dimension tables are usually the source of attributes used to qualify, categorize, or summarize facts in queries, reports, or graphs
  - Dimension data are usually textual and discrete

Star Schema - Dimension tables

- A data mart might contain several star schemas with similar dimension tables but each with a different fact table.
- Typical business dimensions (subjects) are Product, Customer, and Period.
- Period, or time, is always one of the dimensions.
- There are variations of this basic star structure which provide further abilities to summarize and categorize the facts

Components of a Star Schema -Another View



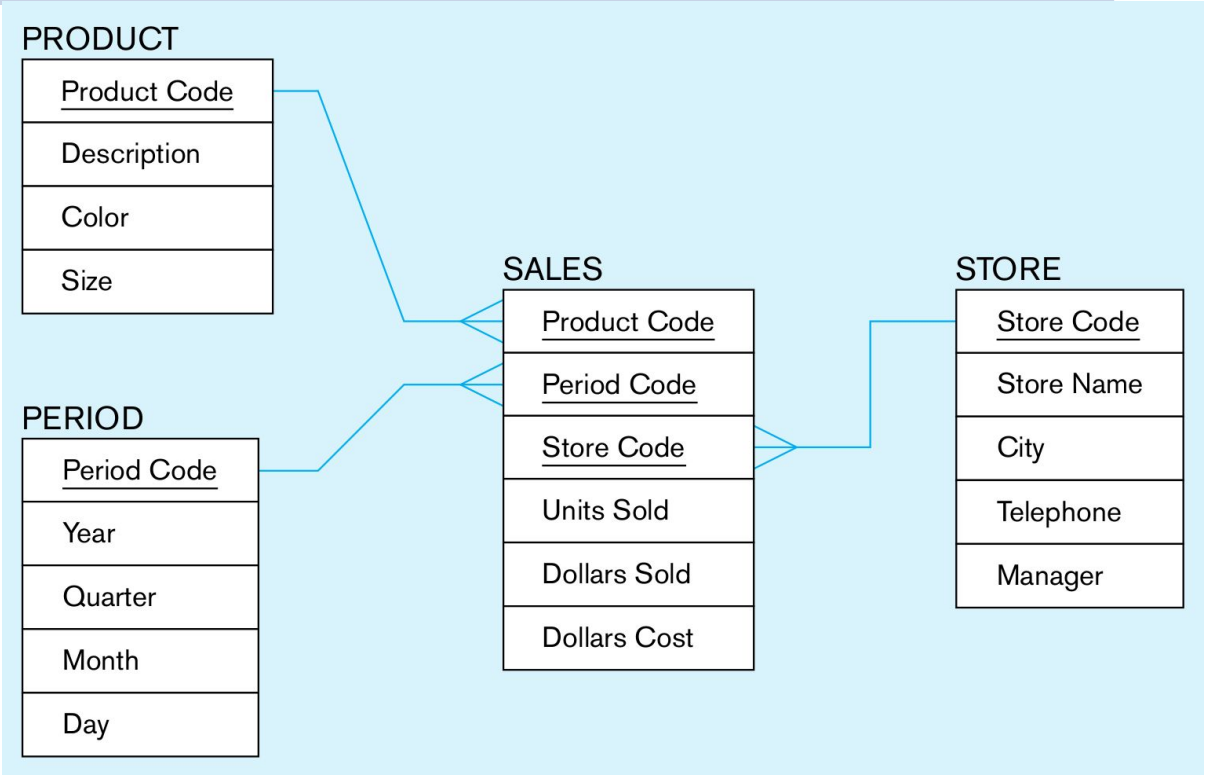
Components of a Star Schema -Another View (Contd.)

- Each dimension table has a one-to-many relationship to the central fact table.
- Each dimension table generally has a simple primary key, as well as several nonkey attributes.
- The primary key, in turn, is a foreign key in the fact table.
- The primary key of the fact table is a composite key that consists of the concatenation of all of the foreign keys, plus possibly other components that do not correspond to dimensions.
- The relationship between each dimension table and the fact table provides a join path that allows users to query the database easily, using SQL statements for either predefined or ad hoc queries.

Components of a Star Schema -Another View (Contd.)

- The star schema is not a new data model, but instead a denormalized implementation of the relational data model.
- The fact table plays the role of a normalized *n-ary* associative entity that links the instances of the various dimensions, which are in second, but possibly not third, normal form.
- The dimension tables are denormalized, and this denormalization is acceptable because dimensions are not updated and avoid costly joins
- The star is optimized around certain facts and business objects to respond to specific information needs
- Relationships between dimensions are not allowed, although such a relationship might exist in the organization

Example Star Schema



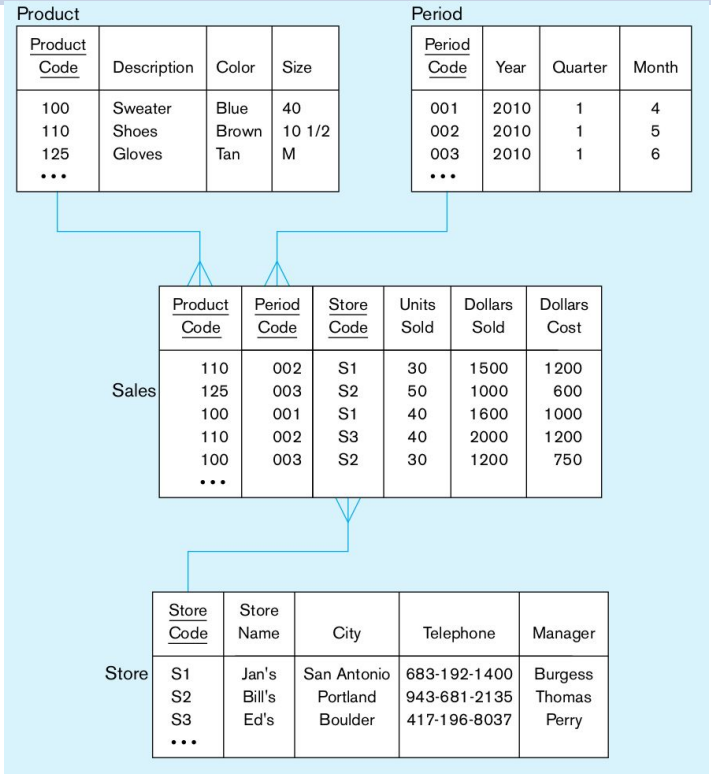
Example Star Schema

- A star schema provides answers to a domain of business questions.
  - What is the average monthly sales for each store manager?
  - Which cities have the highest sales of large products?
  - In which stores are we losing money on which products? Does this vary by quarter?

Example Star Schema (Contd.)

- These questions can be answered from a fully normalized data model of transaction data.
- A fully normalized and detailed database is the most flexible, able to support answering almost any question.
- However, more tables and joins would be involved, data need to be aggregated in standard ways, and data need to be sorted in an understandable sequence.
- These tasks might make it more difficult for the typical business manager to interrogate the data (especially using raw SQL), unless the business intelligence tool he or she uses can mask such complexity from them
- Sufficient sales history would have to be kept, more than would be needed for transaction processing applications

Example Star Schema



Product

Product Code	Description	Color	Size
100	Sweater	Blue	40
110	Shoes	Brown	10 1/2
125	Gloves	Tan	M
...			

Period

Period Code	Year	Quarter	Month
001	2010	1	4
002	2010	1	5
003	2010	1	6
...			

Sales

Product Code	Period Code	Store Code	Units Sold	Dollars Sold	Dollars Cost
110	002	S1	30	1500	1200
125	003	S2	50	1000	600
100	001	S1	40	1600	1000
110	002	S3	40	2000	1200
100	003	S2	30	1200	750
...					

Store

Store Code	Store Name	City	Telephone	Manager
S1	Jan's	San Antonio	683-192-1400	Burgess
S2	Bill's	Portland	943-681-2135	Thomas
S3	Ed's	Boulder	417-196-8037	Perry
...				

From the fact table, the following facts for product number 110 during period 002 can be extracted.

1. Thirty units were sold in store S1. The total dollar sale was 1500, and total dollar cost was 1200.

2. Forty units were sold in store S3. The total dollar sale was 2000, and total dollar cost was 1200.

Additional detail concerning the dimensions for this example can be obtained from the dimension tables. For example, in the PERIOD table, we find that period 002 corresponds to year 2010, quarter 1, month 5.

Surrogate Key

- Every key used to join the fact table with a dimension table should be a surrogate (non-intelligent, or system-assigned) key, not a key that uses a business value (sometimes called a natural, smart, or production key)
- In the above example Product Code, Store Code, and Period Code should all be surrogate keys in both the fact and dimension tables.
- The primary key of each dimension table is its surrogate key.
- The primary key of the fact table is the composite of all the surrogate keys for the related dimension tables, and each of the composite key attributes is obviously a foreign key to the associated dimension table.

Surrogate Key (Contd.)

- Business keys change, often slowly, over time, and we need to remember old and new business key values for the same business object.
- On slowly changing dimensions, a surrogate key allows us to handle changing and unknown keys with ease.
- Using a surrogate key also allows us to keep track of different non-key attribute values for the same production key over time. If a product package changes in size, we can associate the same product production key with several surrogate keys, each for the different package sizes.
- Surrogate keys are often simpler and shorter, especially when the production key is a composite key.
- Surrogate keys can be of the same length and format for all keys, no matter what business dimensions are involved in the database

Grain of the Fact Table

- The raw data of a star schema are kept in the fact table.
- All the data in a fact table are determined by the same combination of composite key elements
- If the most detailed data in a fact table are daily values, then all measurement data must be daily in that fact table, and the lowest level of characteristics for the period dimension must also be a day.
- Determining the lowest level of detailed fact data stored is arguably the most important and difficult data mart design step

Grain of the Fact Table (Contd.)

- The level of detail of this data is specified by the intersection of all of the components of the primary key of the fact table.
- This intersection of primary keys is called the grain of the fact table.
- Determining the grain is critical and must be determined from business decision-making needs
- There is always a way to summarize fact data by aggregating using dimension attributes, but there is no way in the data mart to understand business activity at a level of detail finer than the fact table grain.

Selection of the Grain

- Examples for commonly used grains would be each business transaction
  - An individual line item or an individual scanned item on a product sales receipt
  - a personnel change order
  - a line item on a material receipt
  - a claim against an insurance policy
  - a boarding pass
  - an individual ATM transaction

Selection of the Grain (Contd.)

- A transactional grain allows users to perform analytics such as a market basket analysis (the study of buying behavior of individual customers).
- A grain higher than the transaction level might be all sales of a product on a given day, all receipts of a raw material in a given month at a specific warehouse, or the net effect of all ATM transactions for one ATM session.
- The finer the grain of the fact table, the more dimensions exist, the more fact rows exist, and often the closer the data mart model is to a data model for the operational data store.

Selection of the Grain (Contd.)

- It is recommended using the smallest grain possible, given the limitations of the data mart technology.
- Even when data mart user information requirements imply a certain level of aggregated grain, often after some use, users ask more detailed questions (drill down) as a way to explain why certain aggregated patterns exist.
- You cannot “drill down” below the grain of the fact tables (without going to other data sources, such as the EDW, ODS, or the original source systems, which may add considerable effort to the analysis).

Duration of the Database

- Duration is the amount of history to be kept within a data mart
- The natural duration is about 13 months or 5 calendar quarters, which is sufficient to see annual cycles in the data
- Some businesses, such as financial institutions, have a need for longer durations.
- Older data may be difficult to source and cleanse if additional attributes are required from data sources.
- It may be most difficult to find old values of dimension data, which are less likely than fact data to have been retained.
- Old fact data without associated dimension data at the time of the fact may be worthless.

Size of the Fact Table

- The grain and the duration of the fact table have a direct impact on the size of that table
- Estimating the number of rows in the fact table can be done as follows,
  - Estimate the number of possible values for each dimension associated with the fact table (i.e., the number of possible values for each foreign key in the fact table)
  - Multiply the values obtained in the first step after making any necessary adjustments

Size of the Fact Table (Contd.)

Total number of stores = 1000
Total number of products = 10000
Total number of periods = 24 (2 years' worth of monthly data)

Although there are 10,000 total products, only a fraction of these products are likely to record sales during a given month. Let's assume that on average 50 percent (or 5000) items record sales during a given month.

Total rows = 1000 stores × 5000 active products × 24 months
= 120,000,000 rows

This example clearly illustrates that the size of the fact table is many times larger than the dimension tables

Size of the Fact Table (Contd.)

If we know the size of each field in the fact table, we can further estimate the size (in bytes) of that table. The fact table called SALES has six fields. If each of these fields averages four bytes in length, we can estimate the total size of the fact table as follows,

Total size = 120,000,000 rows × 6 fields × 4 bytes/field
= 2,880,000,000 bytes (or 2.88 gigabytes)

Size of the Fact Table (Contd.)

The size of the fact table depends on both the number of dimensions and the grain of the fact table. Suppose that after using the database discussed earlier for a short period of time, the marketing department requests that daily totals be accumulated in the fact table. (This is a typical evolution of a data mart.) With the grain of the table changed to daily item totals, the number of rows is computed as,

Total rows = 1000 stores × 2000 active products × 720 days (2 years)
= 1,440,000,000 rows

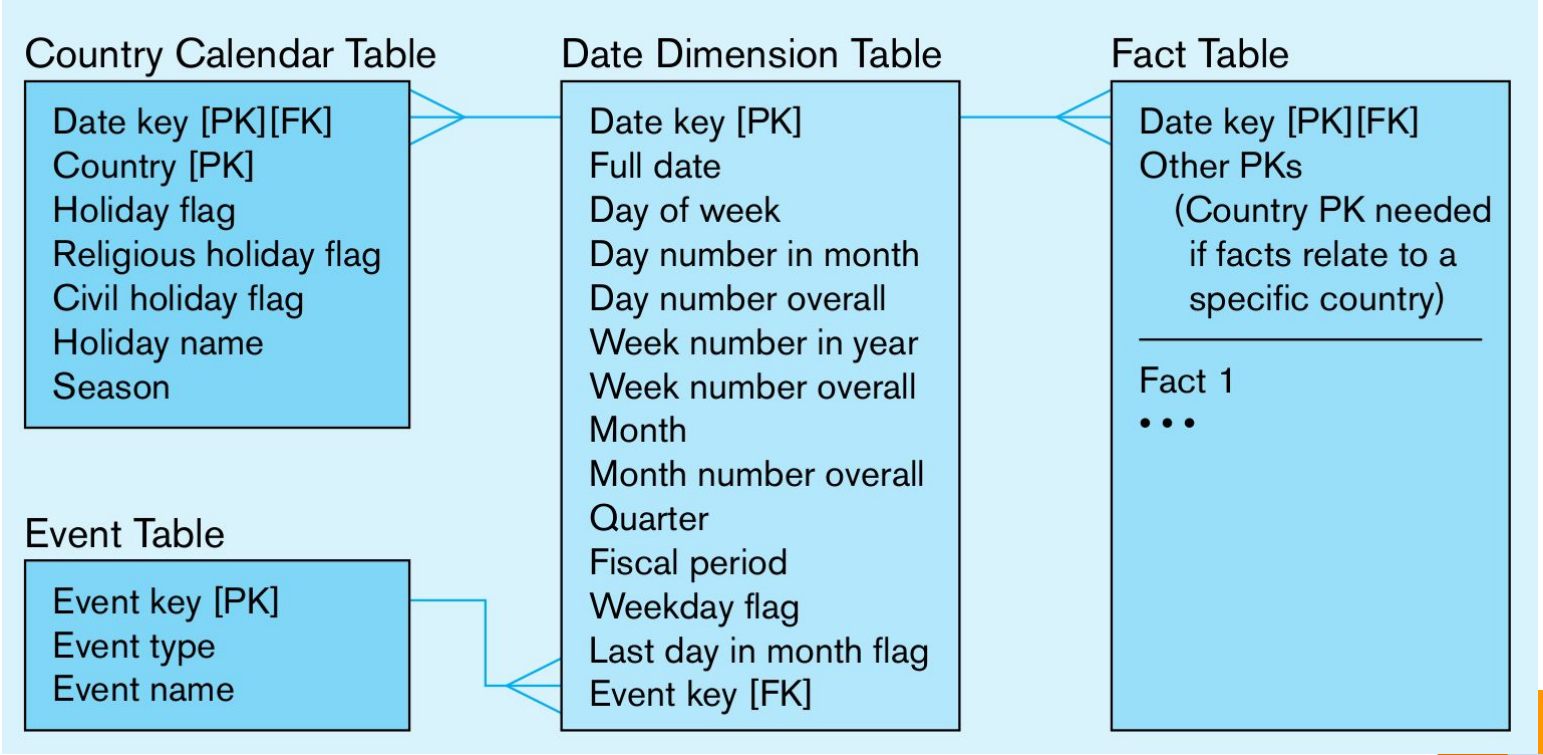
It is assumed that 20 percent of all products record sales on a given day. The database can now be expected to contain well over 1 billion rows.

Total rows = 1,440,000,000 rows × 6 fields × 4 bytes/field
= 34,560,000,000 bytes (or 34.56 gigabytes)

Modeling Date and Time

- Data warehouses and data marts record facts about dimensions over time, date and time (i.e., simply *date*)
- Date is always a dimension table, and a date surrogate key is always one of the components of the primary key of any fact table
- Since a user may want to aggregate facts on many different aspects of date or different kinds of dates, a date dimension may have many nonkey attributes
- Because some characteristics of dates are country or event specific, modeling the date dimension can be even more complex than we could imagine

Modeling Date and Time (Contd.)



Modeling Date and Time (Contd.)

- A date surrogate key appears as part of the primary key of the fact table and is the primary key of the date dimension table.
- The nonkey attributes of the date dimension table include all of the characteristics of dates that users use to categorize, summarize, and group facts that do not vary by country or event.
- For an organization doing business in several countries (or several geographical units in which dates have different characteristics), a Country Calendar table has been added to hold the characteristics of each date in each country.
- Therefore, the Date key is a foreign key in the Country Calendar table
- Each row of the Country Calendar table is unique by the combination of Date key and Country, which form the composite primary key for this table

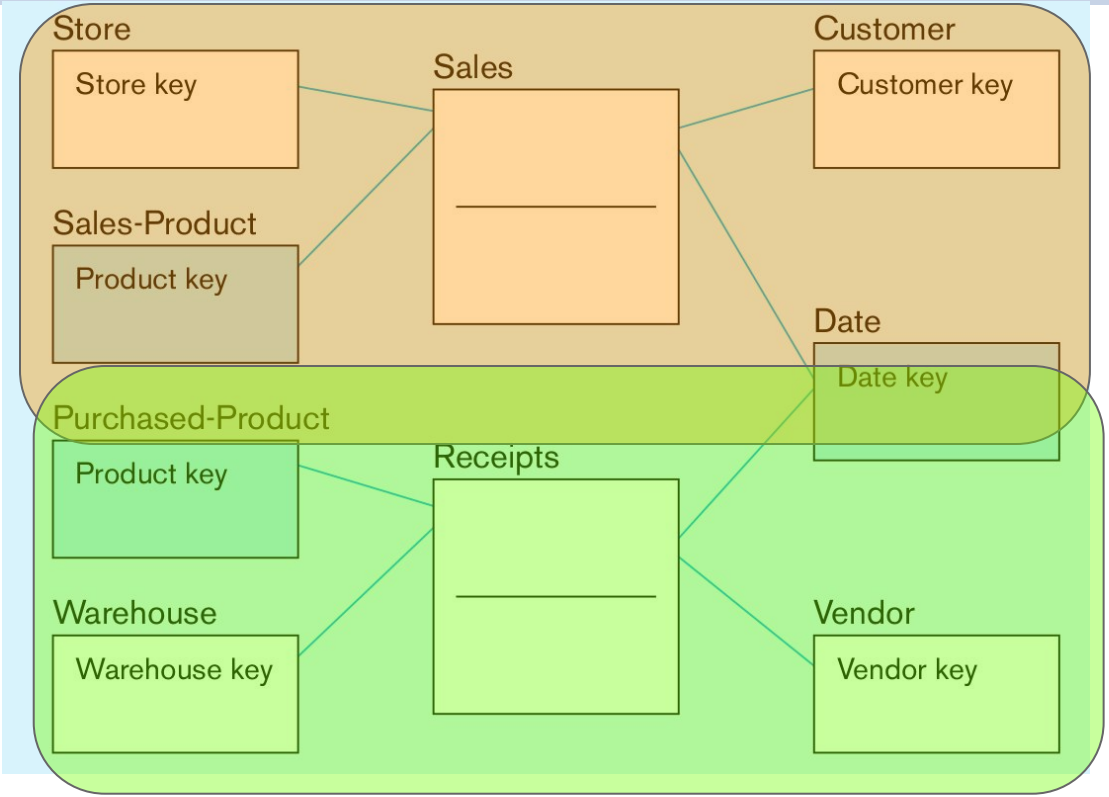
Variations of Star Schema

- The simple star schema is sufficient for many applications
- However,various extensions to this schema are often required to cope with more complex modeling problems.
  - Multiple Fact Tables
  - Factless Fact Tables

Variations of Star Schema - Multiple Fact Tables

- For a given Star schema more than one fact table may exist due to performance and other reasons
- Different users may require aggregation at different levels
- Performance can be improved by defining a different fact table for each level of aggregation
- However, storage requirements may increase dramatically with each new fact table
- Multiple fact tables are needed to store facts for different combinations of dimensions, possibly for different user group

Variations of Star Schema - Multiple Fact Tables



There are two fact tables, one at the center of each star

1. Sales—facts about the sale of a product to a customer in a store on a date
2. Receipts—facts about the receipt of a product from a vendor to a warehouse on a date

Variations of Star Schema - Multiple Fact Tables (Contd.)

- Data about one or more business subjects (in this case, Product and Date) need to be stored in dimension tables for each fact table, Sales and Receipts.
- Two approaches have been adopted in this design to handle shared dimension tables.
  - In one scenario because the description of product is quite different for sales and receipts, two separate product dimension tables have been created
  - In the second scenario because users want the same descriptions of dates, one date dimension table is used
- In each case, it has created a ***conformed dimension***, meaning that the dimension means the same thing with each fact table and, hence, uses the same surrogate primary keys

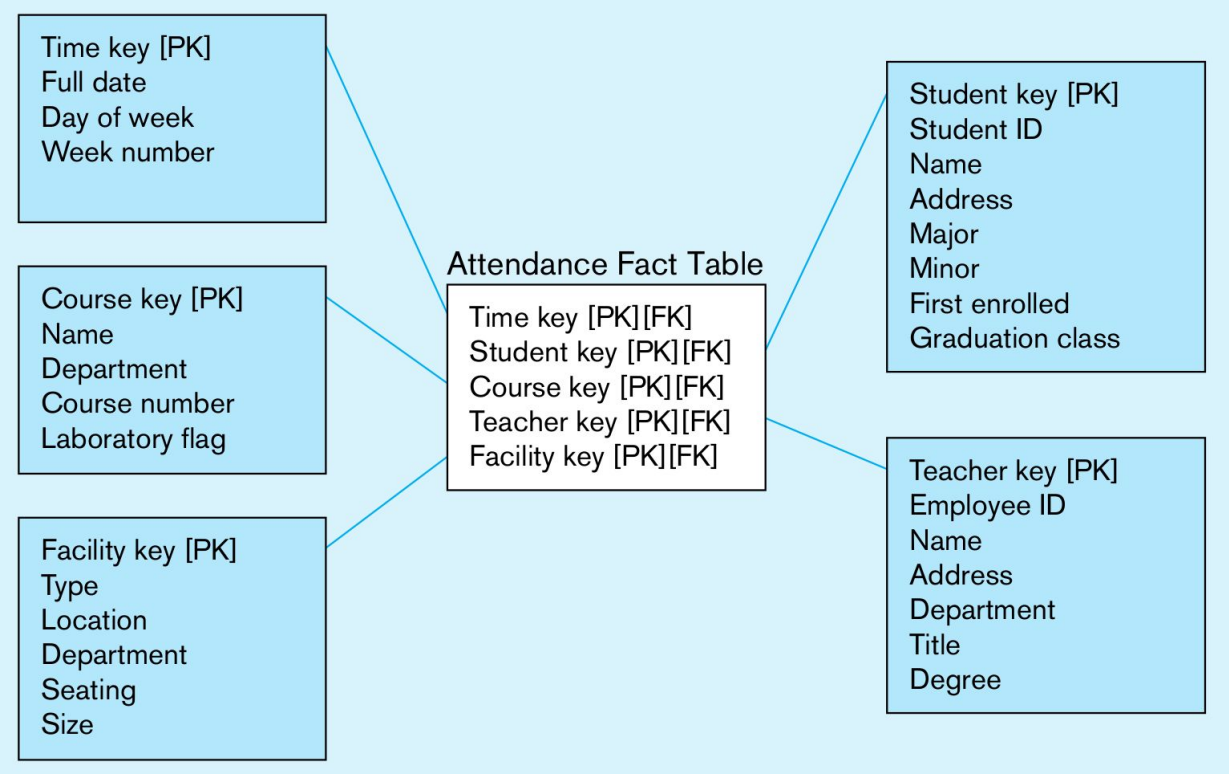
Variations of Star Schema - Multiple Fact Tables (Contd.)

- Even when the two star schemas are stored in separate physical data marts, if dimensions are conformed, there is a potential for asking questions across the data marts
- Conformed dimensions allow users to do the following
  - Share nonkey dimension data
  - Query across fact tables with consistency
  - Work on facts and business subjects for which all users have the same meaning

Factless Fact Tables

- There are applications for fact tables that do not have non-key (fact) data but do have foreign keys for the associated dimensions.
- The two general situations in which factless fact tables may apply
  - Tracking events
  - Taking inventory of the set of possible occurrences

Factless Fact Tables (Contd.)



Factless fact table showing occurrence of an event

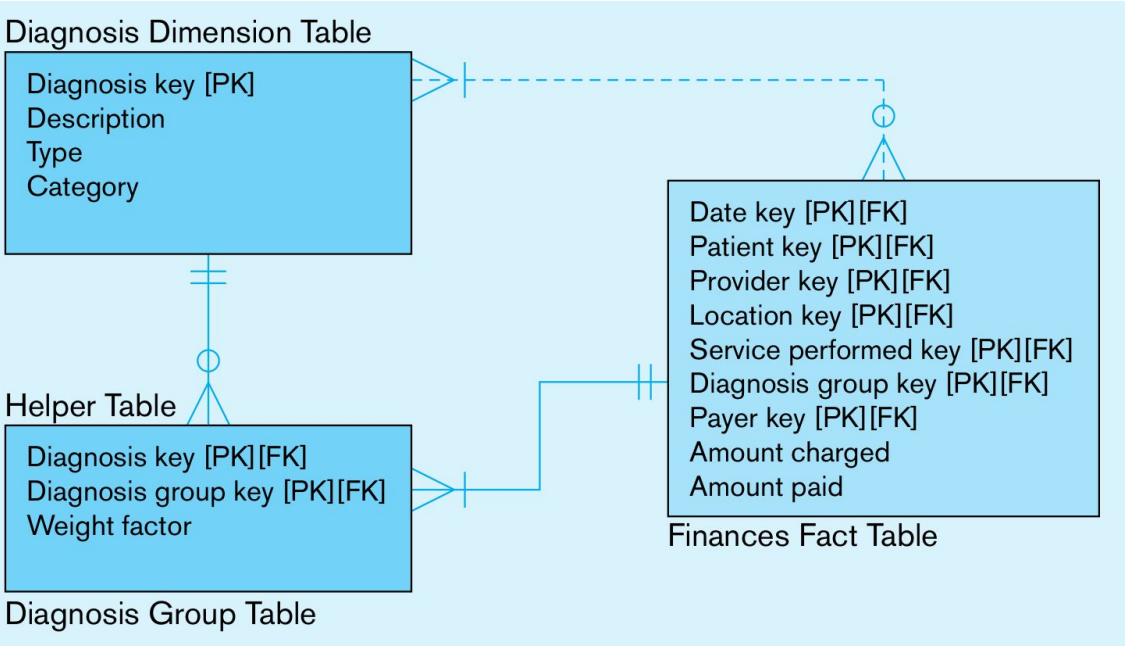
The star schema tracks which students attend which courses at which time in which facilities with which instructors

What is required is to know whether this event occurs, represented by the intersection of the five foreign keys.

Normalizing Dimension Tables

- Fact tables are fully normalized because each fact depends on the whole composite primary key and nothing but the composite key.
- However, dimension tables may not be normalized.
- Sometimes, as with any other relational database, the anomalies of a denormalized dimension table cause add, update, and delete problems
- Hence it requires further normalizing the dimension tables

Multi-valued Dimensions



- There may be a need for facts to be qualified by a set of values for the same business subject.
- E.g., a particular hospital charge and payment for a patient on a date is associated with one or more diagnoses
- We could pick the most important diagnosis as a component key for the Finances table, but that would mean we lose potentially important information about other diagnoses associated with a row.

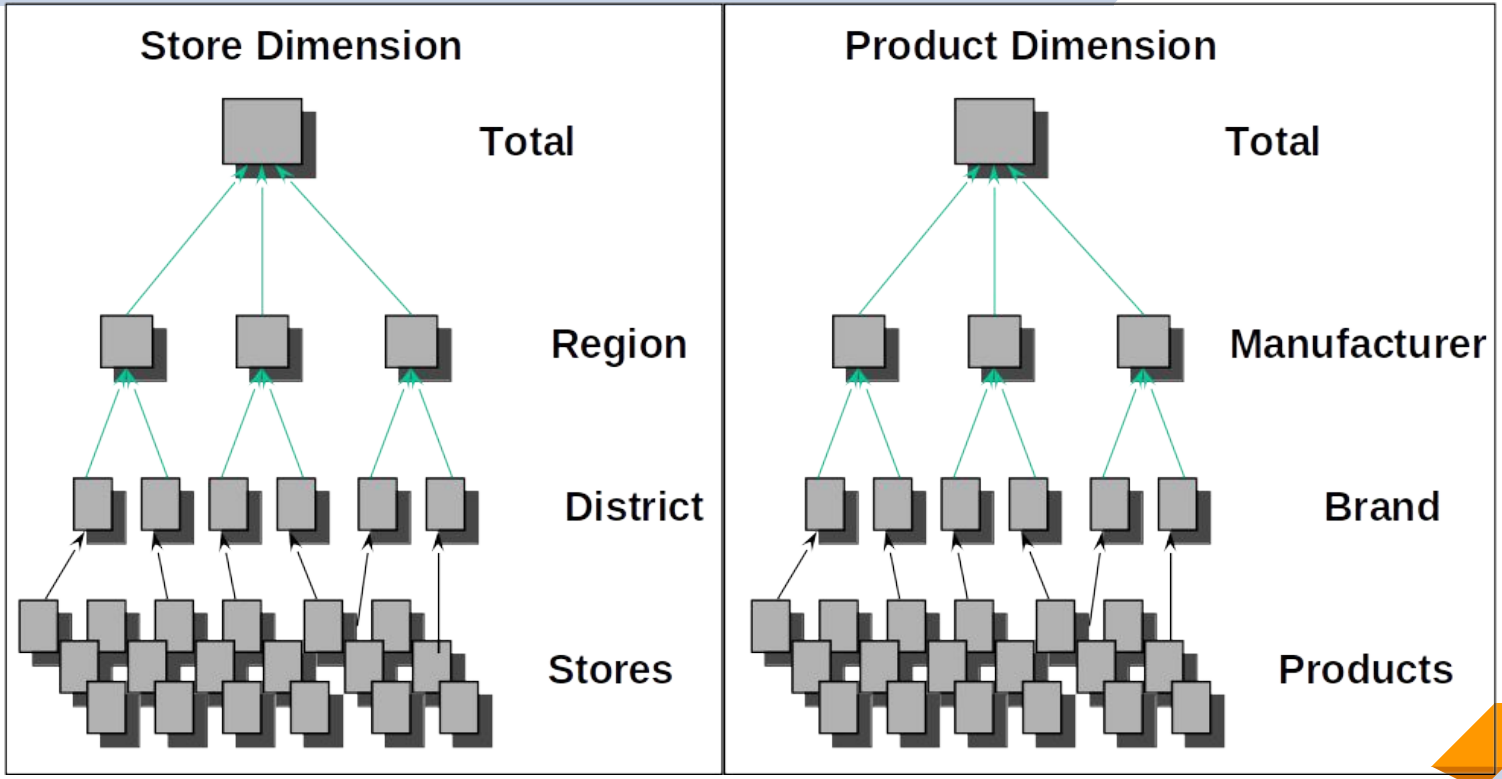
Multi-valued Dimensions (Contd.)

- The best approach (the normalization approach) is to create a table for an associative entity between Diagnosis and Finances, in this case the Diagnosis group table.
- Such an associative entity table is called a “helper table” (also known as *bridge tables*, or *reference tables*)
- A helper table may have nonkey attributes (as can any table for an associative entity)
  - E.g., the weight factor in the Diagnosis group table above indicates the relative role each diagnosis plays in each group, presumably normalized to a total of 100 percent for all the diagnoses in a group.

Hierarchies

- A dimension in a star schema forms a natural, fixed depth hierarchy
  - **Geographical Hierarchies** - markets within a state, states within a region, regions within a country
  - **Product hierarchies** - Packages or sizes within a product, products within bundles, and bundles within product groups

Hierarchies (Contd.)



Hierarchies (Contd.)

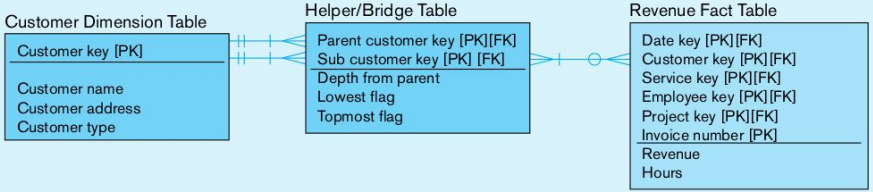
- When a dimension participates in a hierarchy, a database designer has two basic choices,
  1. Include all the information for each level of the hierarchy in a single denormalized dimension table for the most detailed level of the hierarchy, thus creating considerable redundancy and update anomalies.
     - Although it is simple, this is usually not the recommended approach.

Hierarchies (Contd.)

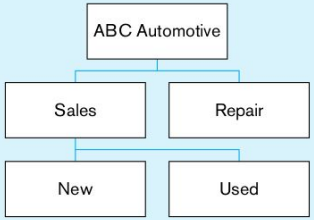
2. Normalize the dimension into a nested set of fixed number of tables with 1:M relationships between them.
   - Associate only the lowest level of the hierarchy with the fact table.
   - It will still be possible to aggregate the fact data at any level of the hierarchy, but now the user will have to perform nested joins along the hierarchy or be given a view of the hierarchy that is pre-joined.

Modelling a Hierarchy - Helper Tables

(a) Use of a helper table



(b) Sample hierarchy with customer and helper tables



Customer Table

Customer key	Name	Address	Type
C0000001	ABC Automotive	100 1st St.	Dealer
C0000002	ABC Auto Sales	110 1st St.	Sales
C0000003	ABC Repair	130 1st St.	Service
C0000004	ABC Auto New Sales	110 1st St.	Sales
C0000005	ABC Auto Used Sales	110 1st St.	Sales

Hierarchy Helper Table

Parent key	Sub key	Depth	Lowest	Topmost
C0000001	C0000001	0	N	Y
C0000001	C0000002	1	N	N
C0000001	C0000003	1	N	N
C0000001	C0000004	2	Y	N
C0000001	C0000005	2	Y	N
C0000002	C0000002	0	N	N
C0000002	C0000004	1	Y	N
C0000002	C0000005	1	Y	N
C0000003	C0000003	0	Y	N
C0000004	C0000004	0	Y	N
C0000005	C0000005	0	Y	N

A hierarchy is typically modeled in a data warehouse using a helper table

- Each customer organizational unit the consulting firm serves is assigned a different surrogate customer key and row in the Customer dimension table, and the customer surrogate key is used as a foreign key in the Revenue fact table.
- This foreign key relates to the Sub customer key in the Helper table because the revenue facts are associated at the lowest possible level of the organizational hierarchy.

Modelling a Hierarchy - Helper Tables (Contd.)

- The problem with joining in a recursive relationship of arbitrary depth is that the user has to write code to join an arbitrary number of times (once for each level of subordination) and these joins in a data warehouse, because of its massive size, can be very t ime-consuming.

- To avoid this problem, the helper table flattens out the hierarchy by recording a row for each organizational subunit and each of its parent organizational units (including itself) all the way up to the top unit of the customer organization

Modelling a Hierarchy - Helper Tables (Contd.)

- Each row of this helper table has three descriptors,
  - the number of levels the subunit is from its parent unit for that table row
  - a flag indicating whether this subunit is the lowest in the hierarchy
  - a flag indicating whether this subunit is the highest in the hierarchy.

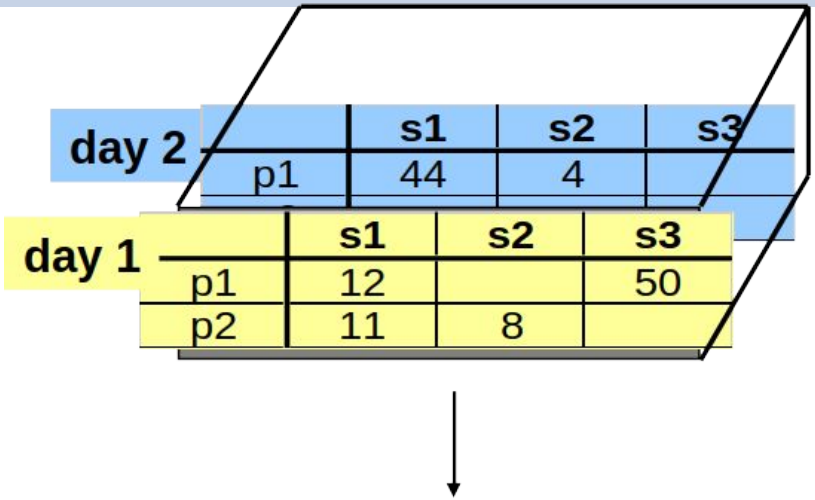
Modelling a Hierarchy (Contd.)

- The Revenue fact table in includes a primary key attribute of Invoice number.
- Invoice number is an example of a degenerative dimension, which has no interesting dimension attributes.
- Invoice number also is not a fact that will be used for aggregation because mathematics on this attribute has no meaning.
- This attribute may be helpful if there is a need to explore an ODS or source systems to find additional details about the invoice transaction or to group together related fact rows

Snowflake Schema

- When the dimension tables are further normalized by using helper tables, the simple star schema turns into a snowflake schema.
- A snowflake schema resembles a segment of an ODS or source database centered on the transaction tables summarized into the fact table and all of the tables directly and indirectly related to these transaction tables.

Aggregation Using Hierarchies

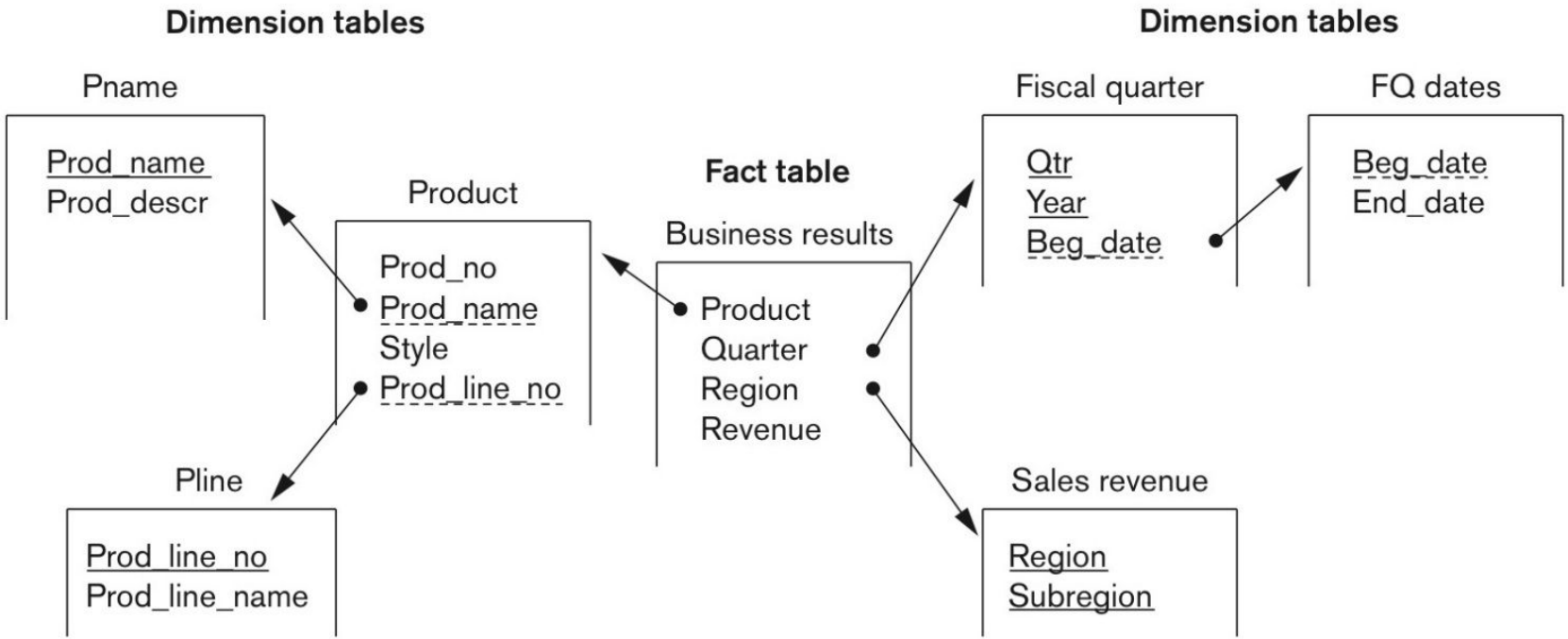


	region A	region B
p1	56	54
p2	11	8

store
region
country

(store s1 in Region A;
stores s2, s3 in Region B)

Snowflake Schema (Contd.)



It is a variation of star schema, in which the dimensional tables from a star schema are organized into a hierarchy by normalizing them.

Snowflake Schema (Contd.)

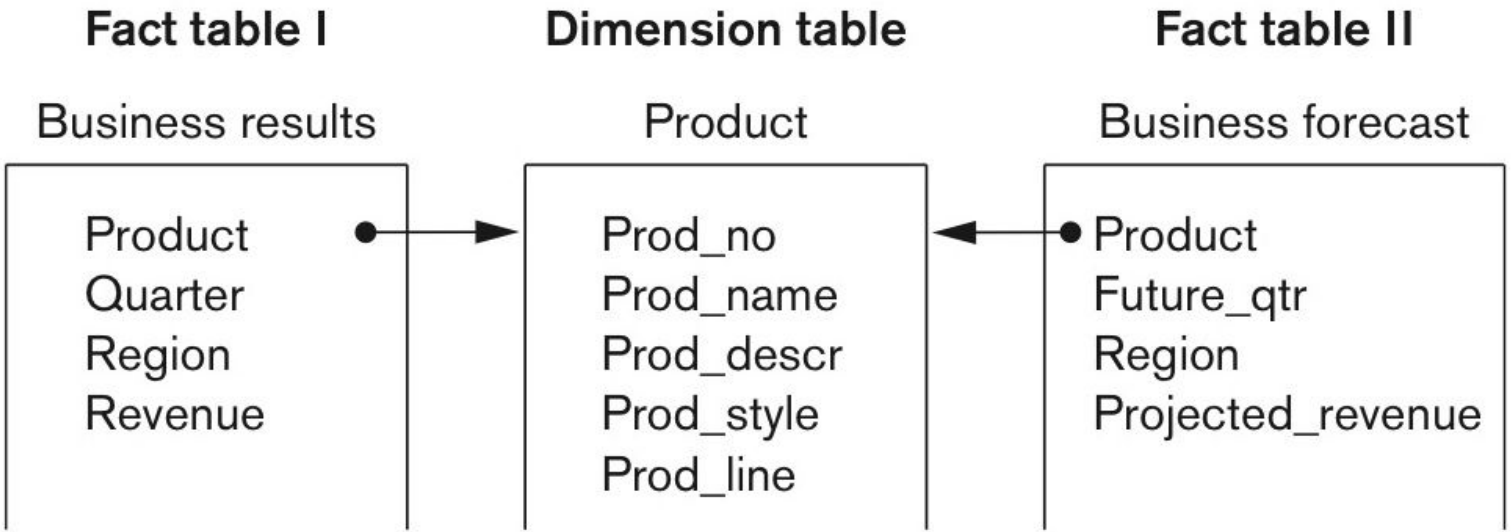
- Many data warehouse experts discourage the use of snowflake schemas because they are more complex for users and require more joins to bring the results together into one table.
- A snowflake may be desirable if the normalization saves significant redundant space (e.g., when there are many redundant, long textual attributes) or when users may find browsing through the normalized tables themselves useful.

Snowflake Schema (Contd.)

- Advantages:
  - Small saving in storage space
  - Normalized structures are easier to update and maintain
- Disadvantages:
  - Schema less intuitive and end-users are put off by the complexity
  - Ability to browse through the contents difficult
  - Degrade query performance because of additional joins

Fact Constellation

- Fact constellation is a set of tables that share some dimension tables. However, fact constellations limit the possible queries for the warehouse.
- Example shows the Product dimension table being shared by two Fact tables.



What is the Best Design?

- Performance benchmarking can be used to determine what is the best design.
- **Snowflake schema**: easier to maintain dimension tables when dimension tables are very large (reduce overall space). It is not generally recommended in a data warehouse environment.
- **Star schema:** more effective for data cube browsing (less joins): can affect performance.

Indexing

- Data warehouse also utilizes indexing to support high performance access.
- A technique called bitmap indexing constructs a bit vector for each value in the domain being indexed.
- Indexing works very well for domains of low cardinality.

Building a Data Warehouse

- The builders of Data warehouse should take a broad view of the anticipated use of the warehouse.
  - The design should harmonize dimensions across the whole enterprise and multiple data sources
  - The design should support ad-hoc querying
  - An appropriate schema should be chosen that reflects the anticipated usage and the business model of the organization.

Building a Data Warehouse (Contd.)

- Steps of designing a data warehouse includes,
  - Acquisition of data for the warehouse.
  - Ensuring that Data Storage meets the query requirements efficiently.
  - Giving full consideration to the environment in which the data warehouse resides.

Building a Data Warehouse - Data Acquisition

- Acquisition of data for the warehouse
  - The data must be extracted from multiple, heterogeneous sources.
  - Data must be formatted for consistency within the warehouse.
  - The data must be cleaned to ensure validity.
    - Difficult to automate cleaning process.
    - Back flushing: refers to upgrading the source data by returning cleaned data.

Building a Data Warehouse - Data Acquisition (Contd.)

- Acquisition of data for the warehouse (contd.)
  - The data must be fitted into the data model of the warehouse. Data may have to be converted from its source model into a multi-dimensional format.
  - The data must be loaded into the warehouse.
    - Proper design for refresh policy should be considered.
    - Data may come from different systems, language areas and time-zones.
    - Order of loading is critical and semantic constraints must be obeyed.

Storing Data in a Data Warehouse

- Storing the data according to the data model of the warehouse
- Creating and maintaining required data structures
- Creating and maintaining appropriate access paths
- Providing for time-variant data as new data are added
- Supporting the updating of warehouse data.
- Refreshing the data
- Purging data

Determining Dimensions and Facts

- Which dimensions and facts are required for a data mart is driven by the context for decision making
- Each decision is based on specific metrics to,
  - monitor the status of some important factor (e.g., inventory turns)
  - predict some critical event (e.g., customer churn)
- Many decisions are based on a mixture of metrics, balancing financial, process efficiency, customer, and business growth factors.

Determining Dimensions and Facts (Contd.)

- Decisions usually start with questions such as,
  - How much did we sell last month?
  - Why did we sell what we did?
  - How much do we think we will sell next month?
  - What can we do to sell the amount we want to sell?
- The answers to questions often cause us to ask new questions.
- Although for a given domain we can anticipate the initial questions someone might ask of a data mart, we cannot perfectly predict everything the users will want to know.

Determining Dimensions and Facts (Contd.)

- Independent data marts are discouraged due to this reason.
- With dependent data marts, it is much easier to expand an existing data mart or for the user to be given access to other data marts or to the EDW when their new questions require data in addition to what is in the current data mart
- The starting point for determining what data should be in a data mart is the initial questions the users want answered. Each question can be broken down into discrete items of business information the user wants to know (facts) and the criteria used to access, sort, group, summarize, and present the facts (dimension attributes)

Determining Dimensions and Facts (Contd.)

(a) Fact-qualifier matrix for sales and customer service tracking

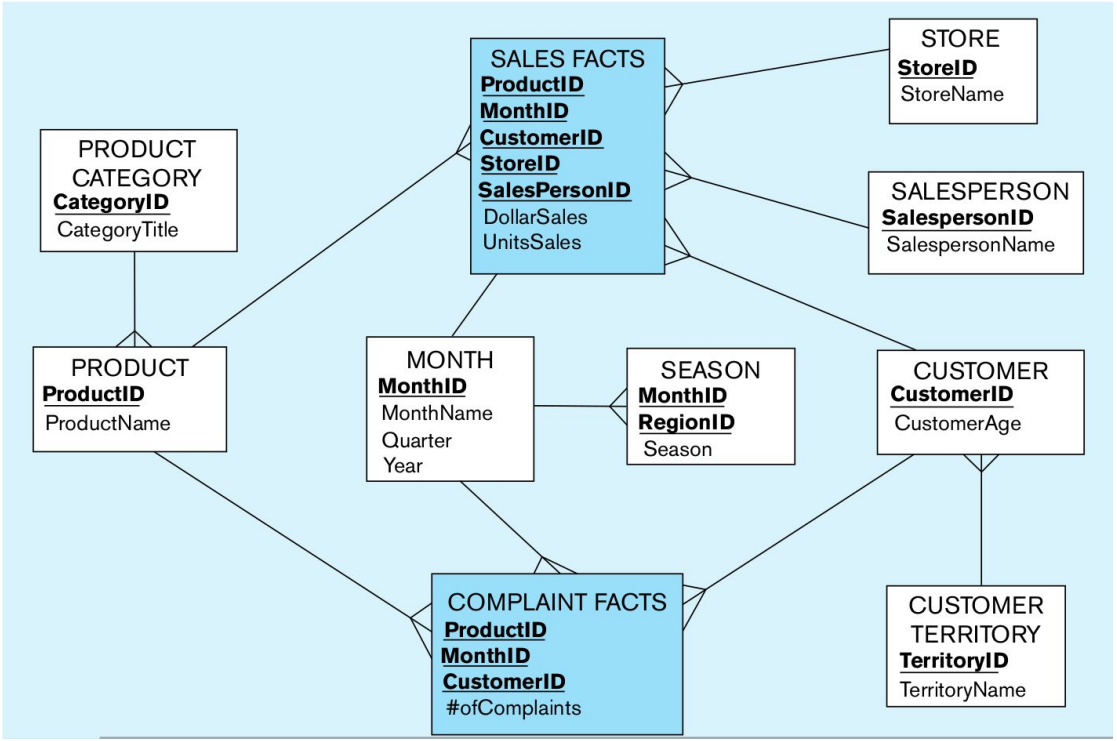
1. What was the dollar sales of health and beauty products in North America to customers over the age of 50 in each of the past three years? 2. What is the name of the salesperson who had the highest dollar sales of each product in the first quarter of this year? 3. How many European customer complaints did we receive on pet food products during the past year? How has it changed from month to month this year? 4. What is the name of the store(s) that had the highest average monthly quantity sales of casual clothing during the summer?	dollar sales	number of complaints	avg. qty. sales
product category	1	3	4
customer territory	1	3	
customer age	1		
year	1	3	
salesperson name	2		
product	2		
quarter	2		
month		3	
store			4
season			4

Determining Dimensions and Facts (Contd.)

- The rows are the qualifiers (dimension or dimension attributes) and the columns are the metrics (facts) referenced in the questions.
- The cells of the matrix contain codes to indicate which qualifiers and metrics are included in each question.
  - For example, question 3 uses the fact number of complaints and the dimension attributes of product category, customer territory, year, and month.

Determining Dimensions and Facts (Contd.)

(b) Star schema for sales and customer service tracking



Determining Dimensions and Facts (Contd.)

- The above diagram has designed two fact tables, because the grain of the facts are different
  - e.g., It was determined complaints have nothing to do with stores or salespersons.
- Also created hierarchical relationships between product and product category and between customer and customer territory
- Product, Customer, and Month are conformed dimensions because they are shared by two fact tables.

Ten Essential Rules of Dimensional Modeling

1. **Use atomic facts**: Eventually, users want detailed data, even if their initial requests are for summarized facts.

2. **Create single-process fact tables**: Each fact table should address the important measurements for one business process, such as taking a customer order or placing a material purchase order.

3. **Include a date dimension for every fact table**: A fact should be described by the characteristics of the associated day (or finer) date/time to which that fact is related.

4. **Enforce consistent grain**: Each measurement in a fact table must be atomic for the same combination of keys (the same grain).

5. **Disallow null keys in fact tables**: Facts apply to the combination of key values, and helper tables may be needed to represent some M:N relationships.

Ten Essential Rules of Dimensional Modeling

6. **Honor hierarchies**: Understand the hierarchies of dimensions and carefully choose to snowflake the hierarchy or denormalize into one dimension.

7. **Decode dimension tables**: Store descriptions of surrogate keys and codes used in fact tables in associated dimension tables, which can then be used to report labels and query filters.

8. **Use surrogate keys**: All dimension table rows should be identified by a surrogate key, with descriptive columns showing the associated production and source system keys.

9. **Conform dimensions**: Conformed dimensions should be used across multiple fact tables.

10. **Balance requirements with actual data**: Unfortunately, source data may not precisely support all business requirements, so you must balance what is technically possible with what users want and need.

Data Warehouse Design Considerations

- Usage projections
- The fit of the data model
- Characteristics of available resources
- Design of the metadata component
- Modular component design
- Design for manageability and change
- Considerations of distributed and parallel architecture
  - Distributed DWs: Replication, Partitioning, Communication, Consistency issues
  - Federated DWs : Decentralized federation of autonomous DWs.

Metadata Repositories

- The metadata repository is a key data warehouse component. It includes both technical and business metadata.
  - **Technical metadata** : covers details of acquisition, processing, storage structures, data descriptions, warehouse operations and maintenance, and access support .
  - **Business metadata** : includes the relevant business rules and organizational details supporting the warehouse.

Functionality of Data Warehouses

- Other functions include-
  - intersection and union of indexes
  - SQL extensions for aggregation
  - Advanced join techniques for star schemas
- Among OLAP functions, we have:
  - ROLAP : Relational OLAP – keeping DW as a relational DB
  - MOLAP: OLAP on multidimensional model
  - HOLAP: Hybrid OLAP – combines the above two. HOLAP can “drill-through” the data to underlying relations.

Data Warehouse vs. Data Views

- Views and data warehouses are alike in that they both have read-only extracts from the databases.
- However, data warehouses are different from views in the following ways:
  - Data Warehouses exist as persistent storage instead of being materialized on demand.
  - Data Warehouses are not just relational, but rather multi-dimensional with multiple levels of aggregation.
  - Data Warehouses can be indexed for optimal performance. Views cannot be indexed directly.
  - Data Warehouses provide specific support of functionality.
  - Data Warehouses deals with large volumes of integrated data that is contained generally in more than one database.
  - Data warehouses bring in data periodically from multiple sources via a complex ETL process. Views do not go through cleaning and pruning.

Warehouse Maintenance

- Warehouse data is analogous to materialized view. One can think of a data warehouse as designing and storing a materialized view (or views) over the information sources
  - Initial loading
  - View maintenance
- Warehouses are different from conventional view maintenance
  - Warehouses may be highly aggregated and summarized
  - Warehouse views may be over history of base data
  - Process large batch updates
  - Schema may evolve
  - Base data doesn’t participate in view maintenance
    - Simply reports changes
    - Loosely coupled
    - Absence of locking, global transactions
    - May not be queriable

Warehouse Maintenance Anomalies

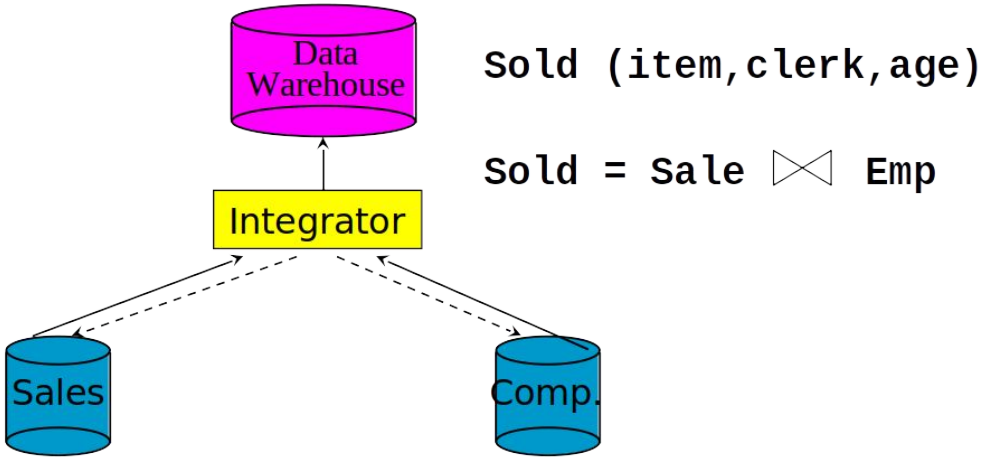
- In our warehousing environment, sources may be legacy or unsophisticated systems that do not understand views, and they are decoupled from the system where the view is stored.
- Sources may inform the integrator (through the monitor) when a change occurs, but they may not be able to determine or obtain additional data needed for incorporating the change into the warehouse.
- Hence, when a change notification arrives at the integrator, the integrator may discover that additional source data (from the same or different sources) is necessary to modify the view.

Warehouse Maintenance Anomalies

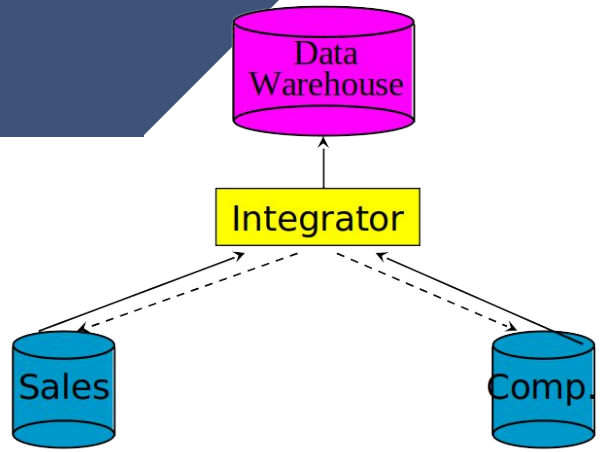
- When the integrator issues queries back to the sources, the queries are evaluated later than the corresponding changes, so the source states may have changed.
- This decoupling between the base data on the one hand (at the sources), and the view definition and view maintenance machinery on the other (at the integrator), can lead the to incorrect views at the warehouse.
- This problem is referred to as the warehouse update anomaly problem.

Warehouse Maintenance Anomalies

- Materialized view maintenance in loosely coupled, non-transactional environment
- Simple example,



Warehouse Maintenance Anomalies (Contd.)



`Sale(item,clerk)` `Emp(clerk,age)`

1. Insert into **`Emp(Mary,25)`**, notify integrator
2. Insert into **`Sale (Computer,Mary)`**, notify integrator
3. (1) → integrator adds **`Sale`** ⋈ **`(Mary,25)`**
4. (2) → integrator adds **`(Computer,Mary)`** ⋈ **`Emp`**
5. View incorrect (duplicate tuple)

Solution for Maintenance Anomalies

- There are a number of mechanisms for avoiding warehousing update anomalies.
- Some solutions assume the source performing view management
  - The source will only notify the integrator of relevant changes, and answer queries asked by the integrator.
- Some solutions require source to lock data while the warehouse view is modified
- Some solutions require source to maintain timestamps for its data

Solution for Maintenance Anomalies

In some solutions view maintenance is autonomous from source changes

- Recompute the view
  - The integrator can either recompute the view whenever a change occurs at a source, or it can recompute the view periodically.
  - Recomputing views is usually time and resource consuming, particularly in a distributed environment where a large amount of data might need to be transferred from the source to the warehouse.

Hammer, J., Garcia-Molina, H., Widom, J., Labio, W. and Zhuge, Y., 1995. The Stanford data warehousing project. Data Engineering, 18(2), pp.41-48.

Solution for Maintenance Anomalies (Contd.)

- Store at the warehouse copies of all data involved in views
  - By keeping up-to-date copies of all relevant source data at the warehouse, queries can be evaluated locally at the warehouse and no anomalies arise. The drawbacks are wasted storage and overhead for keeping the copies current.
- The Eager Compensating Algorithm (ECA)
  - The basic idea is to add to queries sent by the integrator to the sources compensating queries to offset the effect of concurrent updates.

Solution for Maintenance Anomalies

- Research issues: Self-maintainable views
  - What views are self-maintainable
  - Store auxiliary views so original + auxiliary views are self-maintainable

Slowly Changing Dimensions

- Data warehouses and data marts track business activities over time, often for many years
- The business does not remain static over time,
  - products change size and weight
  - customers relocate
  - stores change layouts
  - sales staff are assigned to different locations
  - etc.
- Most systems of record keep only the current values for business subjects (e.g., the current customer address)
- An operational data store keeps only a short history of changes to indicate that changes have occurred and to support business processes handling the immediate changes.

Slowly Changing Dimensions (Contd.)

- However, in a data warehouse or data mart, we need to know the history of values to match the history of facts with the correct dimensional descriptions at the time the facts happened
  - Associate a sales fact with the description of the associated customer during the time period of the sales fact, which may not be the description of that customer today
- Business subjects change slowly compared with most transactional data (e.g., inventory level)

Handling Slowly Changing Dimension (SCD) Attributes

1. Type 1 method - Overwrite the current value with the new value
   a. This is unacceptable because it eliminates the description of the past that we need to interpret historical facts.
2. Type 3 method - For each dimension attribute that changes, create a current value field and as many old value fields as we wish
   a. E.g., a multivalued attribute with a fixed number of occurrences for a limited historical view

Handling Slowly Changing Dimension (SCD) Attributes - Type 3 method

- Type 3 approach might work if there were a predictable number of changes over the length of history retained in the data warehouse
  - E.g., if we need to keep only 24 months of history and an attribute changes value monthly
- However, this works only under this kind of restrictive assumption and cannot be generalized to any slowly changing dimension attribute
- Queries can become quite complex because which column is needed may have to be determined within the query

Handling Slowly Changing Dimension (SCD) Attributes - Type 2 Method

- Type 2 method - Create a new dimension table row (with a new surrogate key) each time the dimension object changes.
- This new row contains all the dimension characteristics at the time of the change.
- The new surrogate key is the original surrogate key plus the start date for the period when these dimension values are in effect.
- A fact row is associated with the surrogate key whose attributes apply at the date/time of the fact.

Handling Slowly Changing Dimension (SCD) Attributes - Type 2 Method

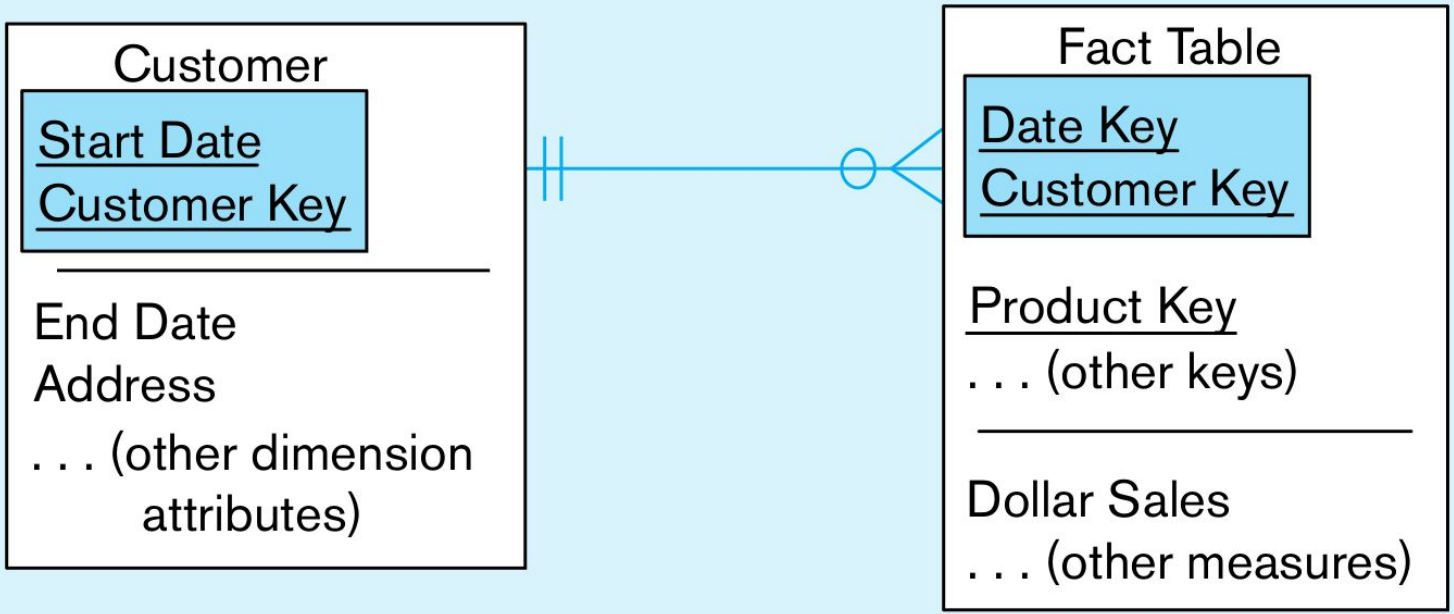
- We likely also want to store in a dimension row the date/time the change ceases being in effect and a reason code for the change.
- This approach allows us to create as many dimensional object changes as necessary.
- However, it becomes too heavy if rows frequently change or if the rows are very long.

Handling Slowly Changing Dimension (SCD) Attributes - Type 2 Method

- The Type 2 scheme is the most frequently used approach for handling slowly changing dimensions for which changes matter.
- Under this scheme, we likely also store in a dimension row the surrogate key value for the original object; this way, we can relate all changes to the same object.
- In fact, the primary key of the dimension table becomes a composite of the original surrogate key plus the date of the change

Handling Slowly Changing Dimension (SCD) Attributes - Type 2 Method

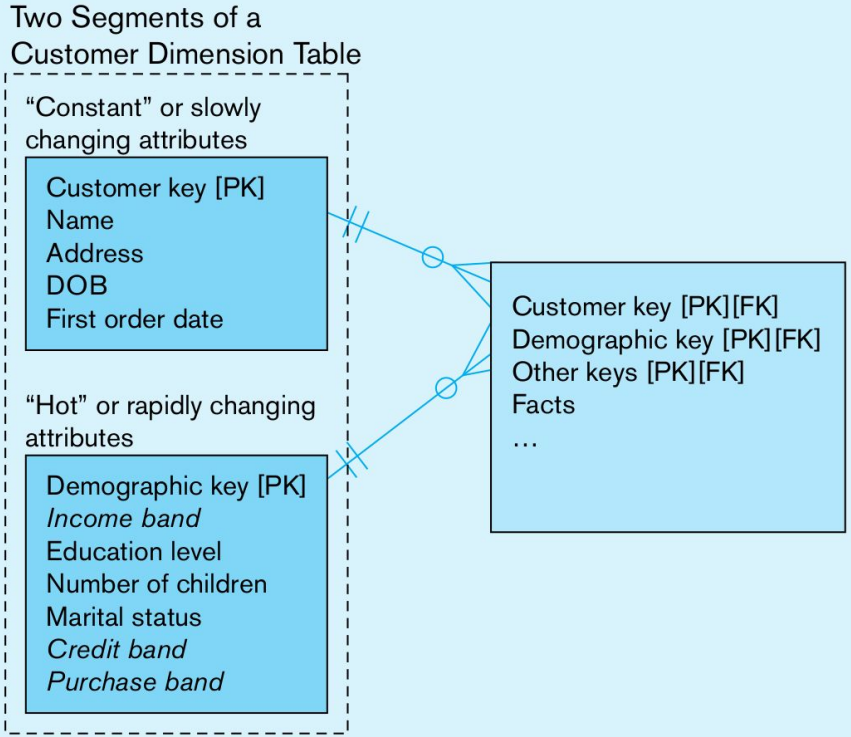
Example of Type 2
SCD Customer dimension table



Handling Slowly Changing Dimension (SCD) Attributes - Type 2 Method

- However, this schema can cause an excessive number of dimension table rows when dimension objects frequently change or when dimension rows are large “monster dimensions.”
- If only a small portion of the dimension row has changing values, there are excessive redundant data created
- ***Dimension Segmentation*** handles this situation

Dimension Segmentation



- Customer dimension is segmented into two dimension tables
  - One segment may hold nearly constant or very slowly changing dimensions
  - Other segments (only two shown in this example) hold clusters of attributes that change more rapidly and for attributes in the same cluster, often change at the same time
- These more rapidly changing attributes are often called "hot" attributes by data warehouse designers

Dimension Segmentation (Contd.)

- For hot attributes, we changed individual dimension attributes, such as customer income (e.g., $75,400/year), into an attribute for a band, or range, of income values (e.g., $60,000–$89,999/year).
- Bands are defined as required by users and are as narrow or wide as can be useful, but certainly some precision is lost.
- Bands make the hot attributes less hot, because a change within a band does not cause a new row to be written.
- This design is more complex for users because they now may have to join facts with multiple dimension segments, depending on the analysis.

Dimension Segmentation (Contd.) - Horizontal segmentation

One other common variation for handling slowly changing dimensions is to segment the dimension table horizontally into two tables,

- one to hold only the current values for the dimension entities
- the other table to hold all the history, possibly including the current row

Many queries need to access only the current values, which can be done quickly from a smaller table of only current rows

When a query needs to look at history, the full dimension history table is used

Difficulties of implementing Data Warehouses

- Lead time is huge in building a data warehouse
  - Potentially it takes years to build and efficiently maintain a data warehouse.
- Both quality and consistency of data as well as master data management are major concerns.
- Revising the usage projections regularly to meet the current requirements.
  - The data warehouse should be designed to accommodate addition and attrition of data sources without major redesign
- DW schema and acquisition component must handle the changes in data sources in terms of structure and content.
- Administration of data warehouse would require far broader skills than are needed for a traditional database.

Challenges in Data Warehousing

- Data cleaning, indexing, partitioning, and views could be given new attention with perspective to data warehousing.
- Automation of
  - data acquisition
  - data quality management
  - selection and construction of access paths and structures
  - self-maintainability
  - functionality and performance optimization
- Incorporating of domain and business rules appropriately into the warehouse creation and maintenance process more intelligently.
- Creating appropriate teams of technical experts that also understand the business.

The future of Data Warehousing

Three key business needs that have emerged,

1. speed of processing
2. cost of storage
3. variety of data

The future of Data Warehousing (Contd.)

1. speed of processing

- Organizations need to invest in upgrading their data warehouse infrastructure to handle the volume and variety of data.
- A key trend in this regard is that of engineered systems wherein the storage, database, and networking aspects of the warehouse are designed and purchased in tandem to provide better performance and flexibility.
- One example of such a platform is SAP HANA (www.saphana.com), a dedicated in-memory database that can meet the transactional, reporting, and analytical needs of an organization.
- To gain optimal performance, the software runs on Intel-based hardware (processor and memory) configurations specifically engineered to support the analytical processing needs of enterprises.

The future of Data Warehousing (Contd.)

2. Cost of Storing Data

- A very attractive option in this regard is to simply move the data warehouse into the cloud and thus enjoy the benefits of lower total cost of ownership (TCO).
- The cloud also allows organizations to use a pay-as-you-go model and grow the size of their data warehouses dynamically as demand arises. Almost all major vendors, such as IBM, Oracle, Microsoft, Teradata, and SAP (HANA), have a cloud-based data warehousing offering.
- In addition, Amazon Web Services recently entered this market with a product named Redshift. Behind the scenes, many of these cloud-based offerings use advanced techniques such as columnar databases, massively parallel processing, and in-memory databases to help achieve faster processing times.

The future of Data Warehousing (Contd.)

3. Dealing with Unstructured Data

- Unstructured data, for example, data from Twitter feeds, are inherently not in a form that can be stored in relational databases.
- This means that new approaches to data transformation and storage are needed to handle the variety of data that is being generated.
- Technologies such as Hadoop play a critical role in helping achieve this transformation and storage in a cost-efficient and timely fashion. Another key technology that is helping handle the variety of data is NoSQL (Not only SQL).

The future of Data Warehousing (Contd.) - Cloud Data Warehouse

- Organizations are actively considering cloud for functions like Hadoop, Spark, databases, data warehouses, and analytics applications
- This is a trend that is increasing due to the benefits of cloud computing such as massive economies of scale, reliability and redundancy, security best practices and easy to use managed services.
- Cloud data warehouses combine these benefits with traditional data warehouse functionality to deliver increased performance and capacity and reducing the administrative burden of maintenance.

Data Lake

- Definition: A data lake allows massive amounts of data to be stored in its native format. The structure and processing rules do not need to be defined until the data is needed.

https://www.matillion.com/uploads/pdf/essential-guide-to-data-lakes-designing-data-lakes-to-optimize-analytics.pdf

Characteristics of Data Lake

- Consolidation
  - The centralization of siloed data is one of the principles of a data lake architecture. This centralization brings a number of benefit's, including being easier to govern and manage, as well as making it easier to innovate non-disruptively around heterogeneous data sets.

- Collect and Store All Data at Any Scale
  - Data lakes allow the collection and storage of data at any scale. In terms of the 3 V’s of Big Data, cloud object storage services provide virtually unlimited space at very low costs.

Characteristics of Data Lake

- Locate, Curate, and Secure Data
  - Having a centralized data lake makes it easier to keep track of what data you have, who has access to it, what type of data you are storing, and what it's being used for. The cost of non-compliance with various regulations, such as GDPR and the Consumer Privacy Act, both in terms of fines and reputational damage is so great that organizations need a way to meet these requirements without stifling innovation.
- Increased Agility
  - Having a centrally curated data lake allows businesses to innovate in new ways of processing data. Users can introduce new use cases without having to re-engineer their architecture. For example, you may be dealing with batch data. However, extending the architecture to include real-time data streaming shouldn’t impact the use cases you already have in place. Similarly, you may be using a data warehouse as a computational resource. Adding Spark workloads to incorporate machine learning use cases can use the same underlying data, made possible by separating storage and compute resources, as well as allowing flexible access to different applications.

How Data Lake Add Value?

- A key value proposition of data lakes is the ability to store data of unknown value, importance or utility for almost negligible cost, data that would otherwise be discarded due to the unjustified costs associated with storage and security.
- A key value proposition of data lakes is the ability to store data of unknown value, importance or utility for almost negligible cost, data that would otherwise be discarded due to the unjustified costs associated with storage and security.
- The historical data can be used to train machine learning models and answer questions in the future

Data Lake Use Cases

- Augmented data warehouse
  - For data that is not queried frequently, or is expensive to store in a data warehouse, federated queries make the different storage types transparent to the end user.
- Support for IoT Data
  - Data lakes are excellent choices for storing the high volume high frequency data from streams. Easily adapted to a lambda architecture they can support near real-time analysis.

Data Lake Use Cases

- Advanced Analytics
  - Quicker access to untransformed data is useful for data scientists, particularly when feature engineering for machine learning models.
- Regulatory compliance
  - A centralized repository of an organization's data makes it easier to apply role-based security, catalog data sets, and track lineage. It allows simpler processes for subject access and removal requests.

Overcoming Data Lake Challenges

- Data Lake allows for the ingestion of large amounts of raw structured, semi-structured, and unstructured data that can be stored en masse and called upon for analysis as and when needed.
- However, it may sometimes lead to inherent risks and can lead to the dreaded Data Swamp which many organizations have fallen prey to

Overcoming Data Lake Challenges (Contd.)

“We see customers creating big data graveyards, dumping everything [into the data lake] and hoping to do something with it down the road. But then they just lose track of what’s there. The main challenge is not creating a data lake but taking advantage of the opportunities it presents.”

Sean Martin, CTO of Cambridge Semantics

Overcoming Data Lake Challenges (Contd.)

- 4 Pillars of Data Governance
  - What data do you have and where it is stored? (**Data Catalog**)
  - Where has data come from and what has happened to it? (**Data Lineage**)
  - Is data accurate and fit for purpose? (**Data Quality**)
  - Is data protected from unauthorized access? (**Data Security**)

This governance layer is a combination of process and tooling which generally increases the total cost of ownership (TCO) of a solution but makes a return on investment (ROI) more likely.

How Data Lakes get Implemented?

- Most data lake implementations use cloud object storage as the underlying storage technology. It is recommended for the following reasons:
  - **Durable** - You can typically expect to see eleven 9s availability
  - **Affordable** - You can store data for approximately $0.01 per GB/Month
  - **Scalable** - Object storage is particularly well suited to storing vast amounts of unstructured data, and storage capacity is virtually unlimited.
  - **Integrated** - Most processing technologies support object storage as both a source and a data sink.
  - **Secure** - Granular access down to the object level

Data Lake vs Data Warehouse

- A data lake is not a direct replacement for a data warehouse, they are supplemental technologies that serve different use cases with some overlap.
- Most organizations that have a data lake will also have a data warehouse

Data Lake vs Data Warehouse (Contd.)

- Data in Data Lakes is Stored in its Native Format
  - Data can be loaded faster and accessed quicker since it does not need to go through an initial transformation process.
  - For traditional relational databases, data would need to be processed and manipulated before being stored.
  - This requires the development of a transformation process, as well as the testing and execution of that process.
  - By loading data directly into the data lake in its source format, you can skip the transformation step for now.

Data Lake vs Data Warehouse (Contd.)

- Access Data Flexibility in a Data Lakes
  - Data scientists, engineers, and analysts can access data much quicker than would be possible in a traditional BI architecture.
  - This increases agility and provides greater opportunities for data exploration and proof of concept activities, as well as self-service business intelligence.

Data Lake vs Data Warehouse (Contd.)

- Data Lakes Provide **Schema-on-Read Access**
  - Traditional data warehouses employ **Schema-on-Write**. This requires an upfront data modeling exercise to define the schema for the data.
  - All data requirements, from all data users, need to be known upfront to ensure the models and schemas produce usable data for all parties. As new requirements are unearthed, those models may need to be redefined.
  - **Schema-on-Read**, conversely, allows the schema to be developed and tailored on a case-by-case basis.
  - The schema is developed and projected on the data sets required for a particular use case. This means that the data required is processed as needed. Once the schema has been developed it can be kept for future use or discarded when no longer required.

Data Lake vs Data Warehouse (Contd.)

- Data Lakes Provide Decoupled Storage and Compute
  - When you separate storage from compute you are fully able to optimize your costs by tailoring your storage requirements to the access frequency.
  - Traditional data warehouses and ETL servers have tightly coupled storage and compute, meaning if we need to increase storage capacity, we also need to increase compute and visa-versa.

Data Lake vs Data Warehouse - Summary

Data Lake	Traditional On-Premises Data Warehouse
• Data stored in native format	• Data requires transformation
• Can store unlimited data forever	• Expensive to store large volumes
• Schema-on-read	• Schema-on-write
• Decoupled storage & compute	• Tightly coupled storage & compute

Data Lake vs Data Warehouse - Summary (Contd.)

	Data Lake	Cloud Data Warehouse	On-Premises Databases
Unstructured data (schema-less data)	Yes	No	No
Semi-Structured data (Self-describing schema)	Yes	Yes	No
Structured Data (Relational)	Yes	Better	Better
Independently scale storage and compute	Yes	Yes	No
Schema-on-Read	Yes	Yes (semi-structured)	No
Schema-on-Write	No	Yes	Yes

Data Lakehouse

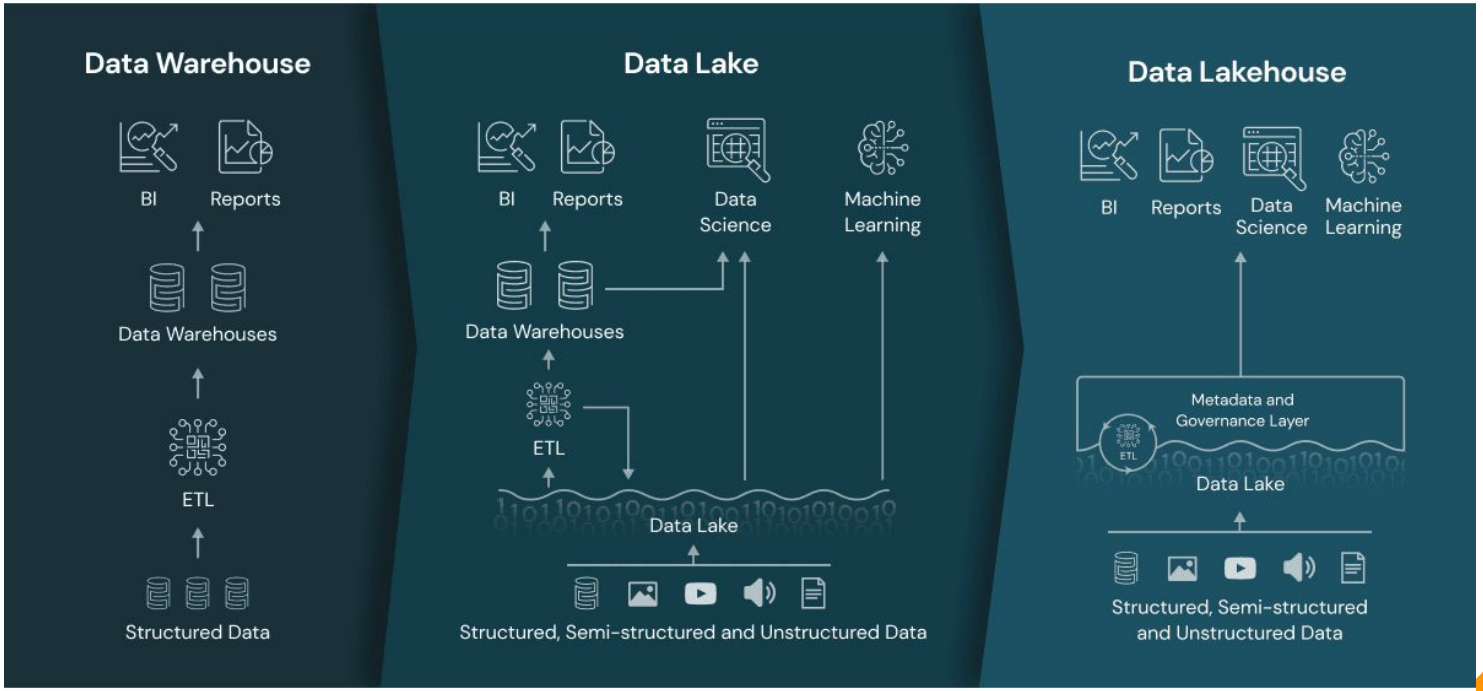
A data lakehouse is a new, open data management architecture that combines the flexibility, cost-efficiency, and scale of data lakes with the data management and ACID transactions of data warehouses, enabling business intelligence (BI) and machine learning (ML) on all data.

https://www.databricks.com/glossary/data-lakehouse

Data Lakehouse Benefits

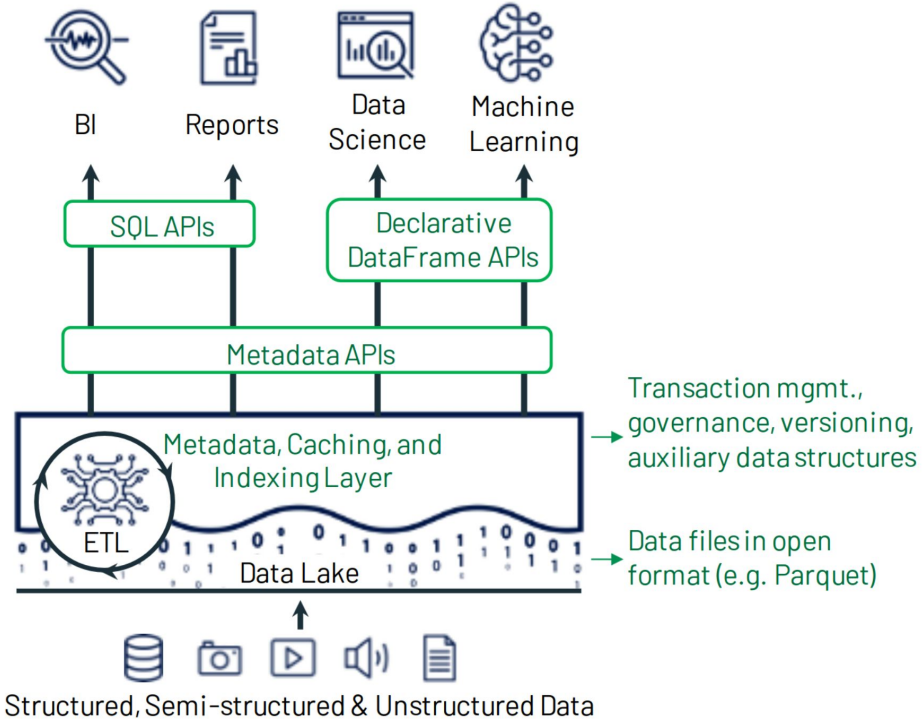
- Increases interoperability of data
- Schema-on-read of semi-structured data
- Increased query concurrency
- Join data lake files with data warehouse tables

Data Warehouse vs Data Lake vs Data Lakehouse



https://www.databricks.com/sites/default/files/2020/12/cidr_lakehouse.pdf

Example Lakehouse System Design



Key features of Data Lakehouse

- A lakehouse has the following key features:
  - **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
  - **Schema enforcement and governance**: The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
  - **BI support**: Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
  - **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.

Key features of Data Lakehouse

- **Openness**: The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data directly.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads**: including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming**: Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

Data Lakehouse Summary

The lakehouse is a new data management architecture that radically simplifies enterprise data infrastructure and accelerates innovation in an age when machine learning is poised to disrupt every industry.

- In the past most of the data that went into a company's products or decision making was structured data from operational systems, whereas today, many products incorporate AI in the form of computer vision and speech models, text mining, and others.

- Over time lakehouses will close these gaps while retaining the core properties of being simpler, more cost efficient, and more capable of serving diverse data applications.

Thank you!