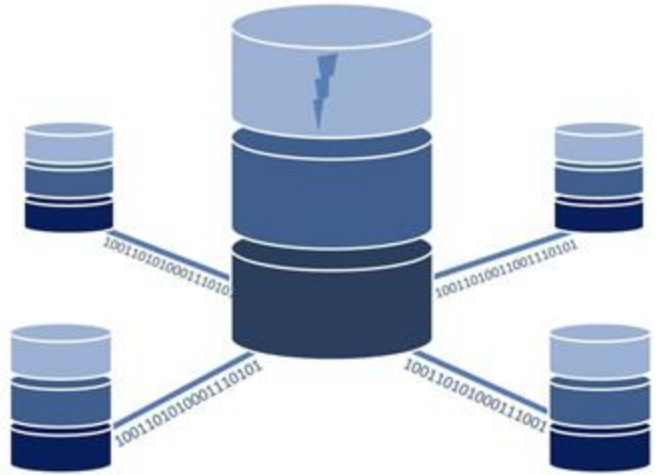
Databases

Relational databases

Week 4

Database: Definition

An organized collection of data in a manner that allows easy access, management, and updating



Database: Examples

- **Library System**:
  - **Purpose**: Manage books, users, borrowing history.
  - **Typical Data**: Titles, ISBNs, user names, due dates.
- **School/University Database**:
  - **Purpose**: Handle student enrollments, grades, courses, faculty.
  - **Typical Data:** Student & faculty names, courses, grades, schedules.
- **Hospital Management System**:
  - **Purpose**: Organize patient info, appointments, billing, inventory.
  - **Typical Data:** Patient names, medical history, treatment, billing details.

Purpose of a Databases

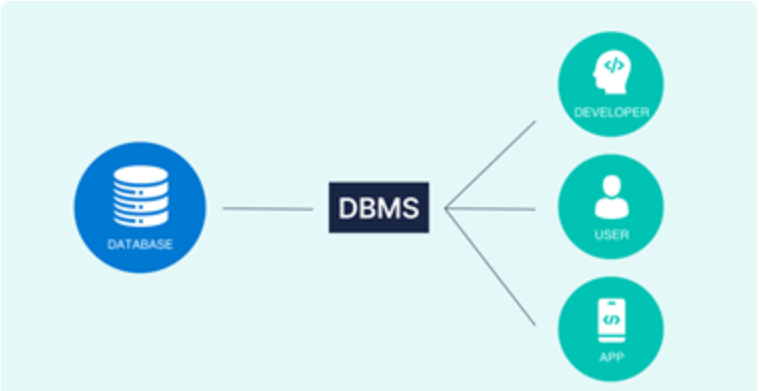
Centralized Storage	Consolidate vast data systematically. **Eliminate data silos** within organizations.
Data Integrity	Implement **validation rules** to maintain data quality.
Efficient Retrieval	**Indexing** and **optimized** queries for data retrieval.
Multiple User Access	Supports simultaneous operations by **multiple users**. Ensure data consistency with transactions.
Data Security	Granular user permissions to **control data access**. Defend against unauthorized access and breaches.

DBMS (Database Management System)

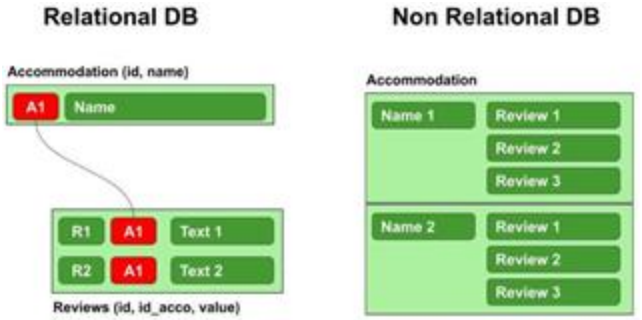
Database Management System (DBMS) is a software that allows users to create, manage and interact with databases.

- The role of a DBMS:
  - Acts as an **interface** between the database and its users or applications.
  - Enables users to store, retrieve, update and manage data.
  - DBMS support administrative tasks such as Performance monitoring, Tuning, Backup and recovery

Different DBMS cater to varied needs, from small-scale applications to enterprise-level data management.



Types of Databases: Relational & Non-Relational



- **Relational**
  - Structured in **tables, like spreadsheets**.
  - Tables have defined **"columns"**.
  - Allows **links** between tables.
  - Mostly use a **standard query language**.
  - Example: **address book** with **contacts** and their **details**.

- **Non-Relational**
  - **Flexible** data storage..
  - Not restricted to tables; **diverse** storage methods.
  - Tailored for **specific data needs**.
  - Uses **diverse query** techniques.
  - **Example**: A **box** storing notes, photos, books.

Relational Databases

Relational databases have **structured data**

- Use of **tables with rows and columns** to represent **data** and **relationships**.

Examples of Relational DBMS:

- **MySQL:** Popular for web applications.
- **PostgreSQL:** Known for extensibility and compliance.
- **Oracle Database:** Widely used in enterprise settings.
- **Microsoft SQL Server:** Ideal for business applications.

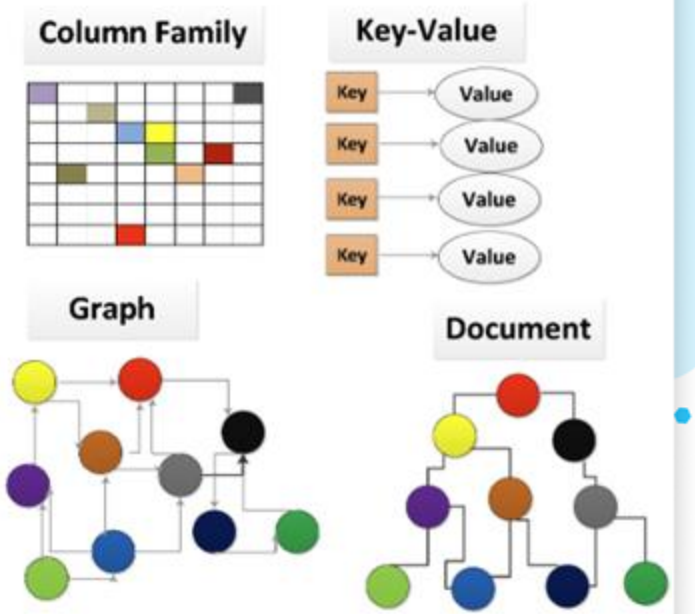
Non-Relational Databases: Detailed View

Non-Relational databases have **unstructured/semi-structured data**

- Designed for **specific data models**;
- Can handle data that **doesn't fit into tabular structures**.

Examples of Non-Relational DBMS:

- **MongoDB:** Uses JSON-like documents
- **Redis:** Stores data as key-value pairs in memory
- **Cassandra:** Stores data in columns rather than rows
- **Neo4j:** Designed for interconnected data



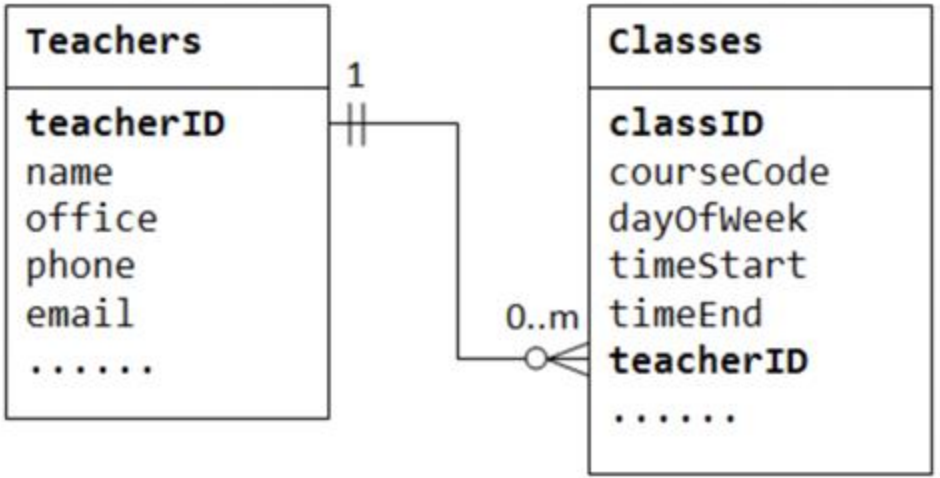
Key Takeaway

Relational databases are optimal for **structured data** with **clear relationships** and schemas.

Non-relational databases are versatile and can handle **unstructured** or differently structured data, catering to specific use-cases.

Relational Databases: Introduction

Data is organized and linked in tables for efficient query operations, there are *relationships* between tables



Relational Databases: Basic Components

Tables:

- Represents a set of data; e.g., "Customers" or "Orders".

Rows:

- Individual data entries in a table; e.g., a single customer's details.

Columns:

- Categories of data in a table; e.g., "First Name", "Last Name", "Email".

CustomerID	First Name	Last Name	Email
C001	John	Doe	johndoe@email.com
C002	Jane	Smith	janesmith@email.com
C003	Alice	Brown	alicebrown@email.com
C004	Robert	White	robertwhite@email.com

Relational Databases: Key Concepts

Primary key
Uniquely identifies each row in a table.
Ensures data integrity

Foreign key
Establishes a link between data in two tables.
Enforces referential integrity

Table 1: Customers

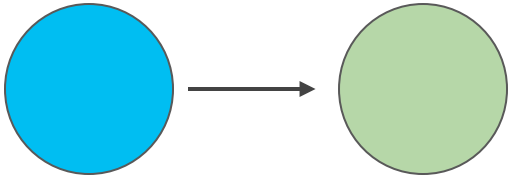
CustomerID (Primary Key)	First Name	Last Name	Email
C001	John	Doe	johndoe@email.com
C002	Jane	Smith	janesmith@email.com
C003	Alice	Brown	alicebrown@email.com

Table 2: Orders

OrderID	CustomerID (Foreign Key)	Product	Quantity
O001	C001	Book	2
O002	C003	Laptop	1
O003	C004	Chair	4

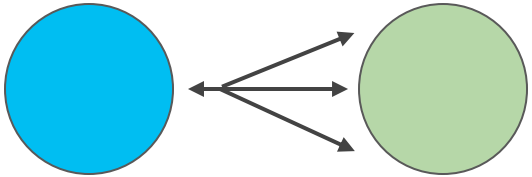
Relationships in Databases

One to one



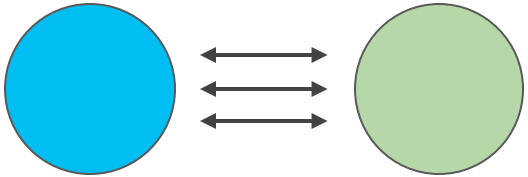
In this type of relationship, **one** record in the first table corresponds **to one**, and only one, record in the second table, and vice versa.

One to many



One record in the first table can relate **to multiple** records in the second table. However, a record from the second table can relate to only one record in the first table.

Many to many



Multiple records in the first table can relate **to multiple** records in the second table.

= a table **= a relationship**

Relationship Examples: One to one

- Each record in the Person table corresponds to one, and only one, record in the Passport table **through the PersonID field**, and vice versa.
- The PersonID in the Passport table is a **foreign key** that establishes a link to the PersonID in the Person table.

Table 1: Person

PersonID (Primary Key)	FirstName	LastName	Email
1	John	Doe	johndoe@email.com
2	Jane	Smith	janesmith@email.com
3	Alice	Brown	alicebrown@email.com

One to one

Table 2: Passport

PassportID (Primary Key)	PersonID (Foreign Key)	PassportNumber	Country
1	1	X1234567	USA
2	2	Y2345678	UK
3	3	Z3456789	Canada

Relationship Examples: One to many

- Each record in the Customers table can correspond to multiple records in the Orders table through the **CustomerID field**, but each order is associated with one and only one customer.
- The CustomerID in the Orders table is a **foreign key** that establishes a link to the CustomerID in the Customers table.

Table 1: Customers

CustomerID (Primary Key)	First Name	Last Name	Email
C001	John	Doe	johndoe@email.com
C002	Jane	Smith	janesmith@email.com
C003	Alice	Brown	alicebrown@email.com

One to many

Table 2: Orders

OrderID	CustomerID (Foreign Key)	Product	Quantity
O001	C001	Book	2
O002	C003	Laptop	1
O003	C004	Chair	4

Relationship Examples: Many to many

Table 1: Students

StudentID	FirstName	LastName
S001	John	Doe
S002	Jane	Smith
S003	Alice	Brown

Table 2: Courses

CourseID	CourseName
C001	Mathematics
C002	Physics
C003	Chemistry

Table 3: Enrollments (Junction Table)

EnrollmentID	StudentID	CourseID
E001	S001	C001
E002	S001	C002
E003	S002	C001
E004	S003	C003
E005	S003	C002

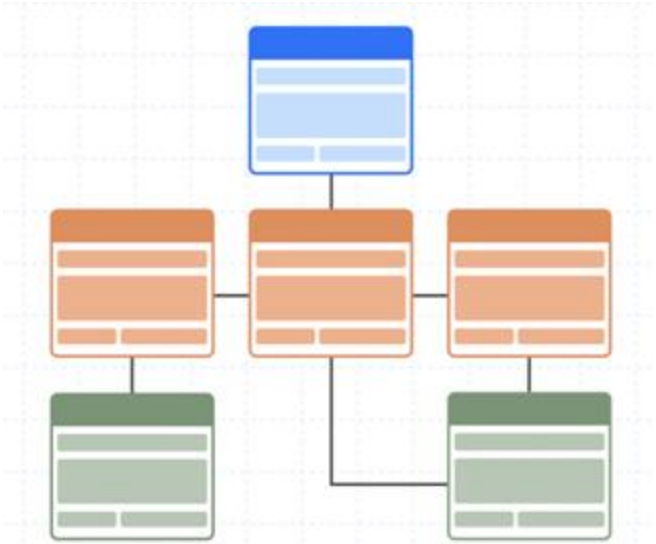
- Each student can enroll in multiple courses, and each course can have multiple students, creating a many-to-many relationship.
- The Enrollments table serves as a **junction table** that breaks down the many-to-many relationship into two one-to-many relationships: one between Students and Enrollments, and another between Courses and Enrollments.

Junction Tables -Why use them?

- **Avoid Data Redundancy:**
  - Junction tables prevent duplicate data, promoting efficient storage and management.
- **Maintain Data Integrity:**
  - Ensures consistency and accuracy, as each piece of data is stored in one place.
- **Simplify Data Retrieval:**
  - Facilitates easier and more efficient queries, without dealing with duplicates.
- **Enable Many-to-Many Relationships:**
  - Directly represents many-to-many relationships, which isn't feasible with just two tables.
- **Facilitate Relationship Management:**
  - Provides a structured way to manage and navigate complex relationships between data.
- **Ease of Update:**
  - Changes to data (updates or deletions) need to be made in only one location, reducing the risk of errors.
- **Enhance Scalability:**
  - Supports the addition of more records without requiring changes to existing table structures.
- **Improve Performance:**
  - Optimizes database performance, especially for large datasets, by streamlining queries and data retrieval processes.

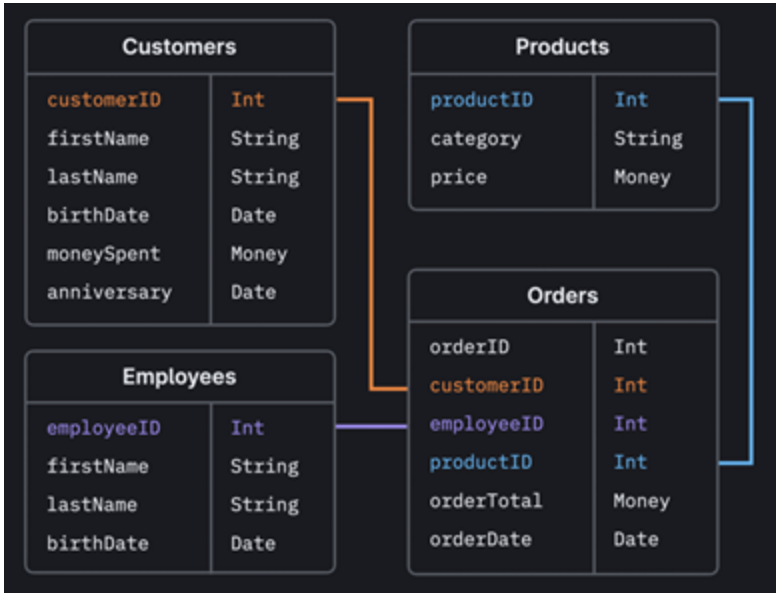
Database Schema

- **Relational Databases have Schemas.**
  - A blueprint or architecture of how a database is constructed and organized.
  - Defines the tables, the fields in each table, and the relationships between fields and tables, constraints and more.
- **Importance:**
  - Ensures data consistency and integrity.
  - Facilitates efficient data retrieval and insertion.
  - Provides a structured way to represent real-world entities and their interrelationships.



Database Schema: Components

- Tables: Store the actual data.
- Fields: Represent attributes or properties of entities.
- Constraints: Rules that determine what data can be stored.
- Relationships: Define how tables connect and interact with each other.



Transactions & ACID Properties

- When discussing database schemas and structures, it's not just about how data is stored.
- It's also about how data operations (like adding, updating, or deleting records) are handled.
- Transactions ensure that operations on a database are carried out correctly and safely.

ACID properties represent a set of four key principles that ensure reliable processing of transactions, guaranteeing data integrity and stability even in the face of system failures or errors.

Transactions & ACID Properties

Atomicity
Think of a transaction as a single unit of work. Atomicity ensures that either every operation within this unit succeeds, or none do. It's an all-or-nothing approach.

Consistency
After a transaction, the database should remain in a valid state. So, if we modify any data, it should adhere to the rules and constraints of the schema.

Isolation
Isolation ensures that each transaction is executed as if it's the only operation in the system, maintaining data integrity.

Durability
Once a transaction is acknowledged, the system ensures its effects remain, even if there's a system crash or power failure later on.

What is Database Indexing?

- A data structure that boosts operation speeds in a table.
- Similar to an index in a book, it leads to faster searches without scanning every row in a table.
- Facilitates swift random lookups and efficient access.

Database indexing is a technique that enhances data retrieval speeds, ensuring efficient and rapid access to information within large datasets.

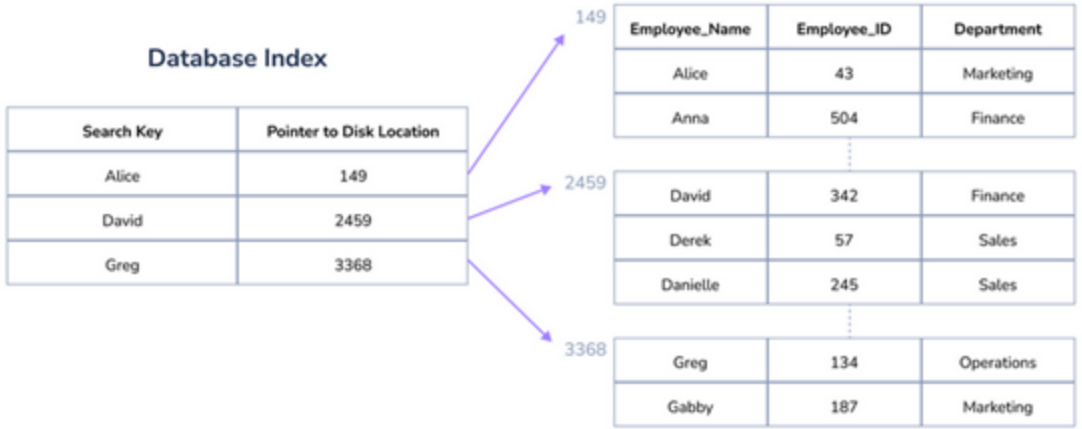
How Indexing Works

Basics of Indexing

- Indexes store a subset of data, typically column values, with a reference to the actual row.
- Checking the index first when querying reduces rows examined.

Composite Indexes

- Span multiple columns for combined column queries.
- Useful for frequent, specific column combinations but may have trade-offs in size and speed.



Data Modeling

What is data modeling?

- A visual representation of an information system.
- Illustrates data types, relationships, formats, and attributes.
- Acts like a roadmap, similar to an architect's blueprint.

Why model data?

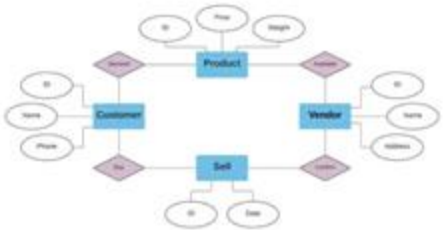
- Built around business needs.
- Ensures alignment with stakeholder feedback and requirements.
- Defines rules upfront for designing or iterating systems.

Data Modeling Techniques

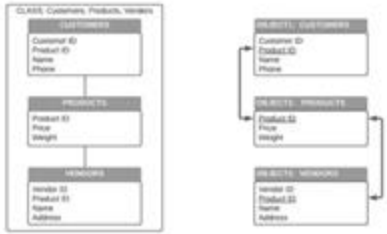
Hierarchical
Data modeling technique

Relational
Data modeling technique

Network
Data modeling technique



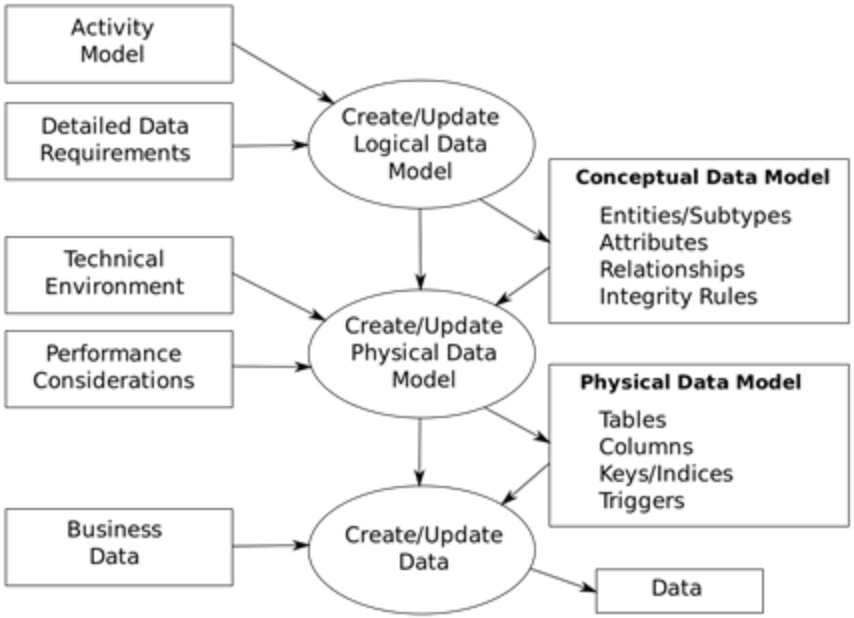
Entity-relationship
Data modeling technique



Object-oriented
Data modeling technique

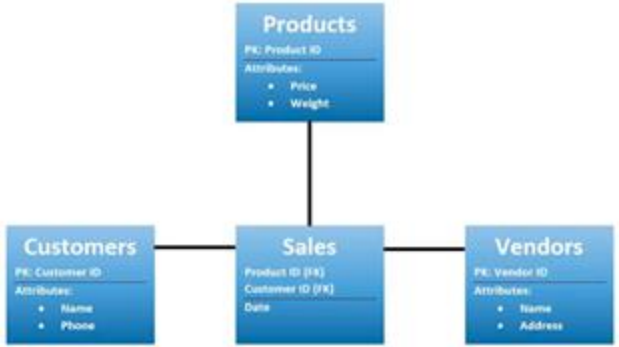
Different data modeling techniques offer diverse ways to organize, store, and interact with data. The choice depends on specific requirements and use-cases.

Data Modeling Process

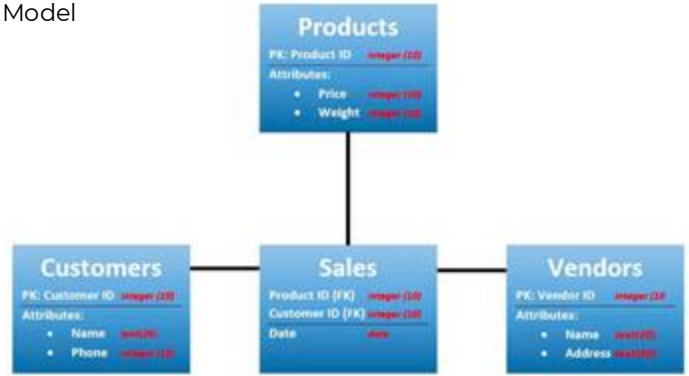


Data Modeling Process

Conceptual Model



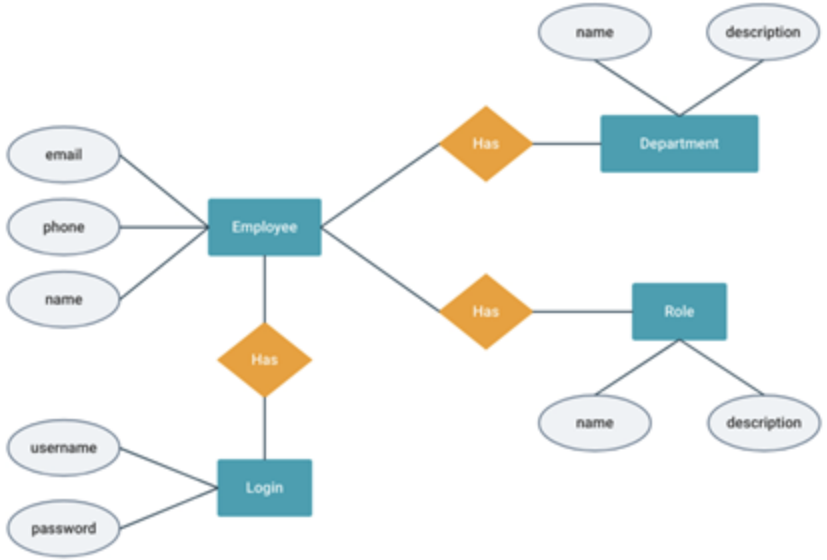
Logical Model



Physical Model

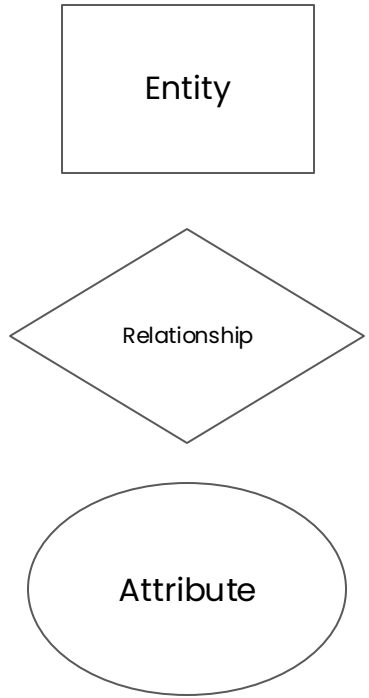
Entity Relationship Diagram (ERD)

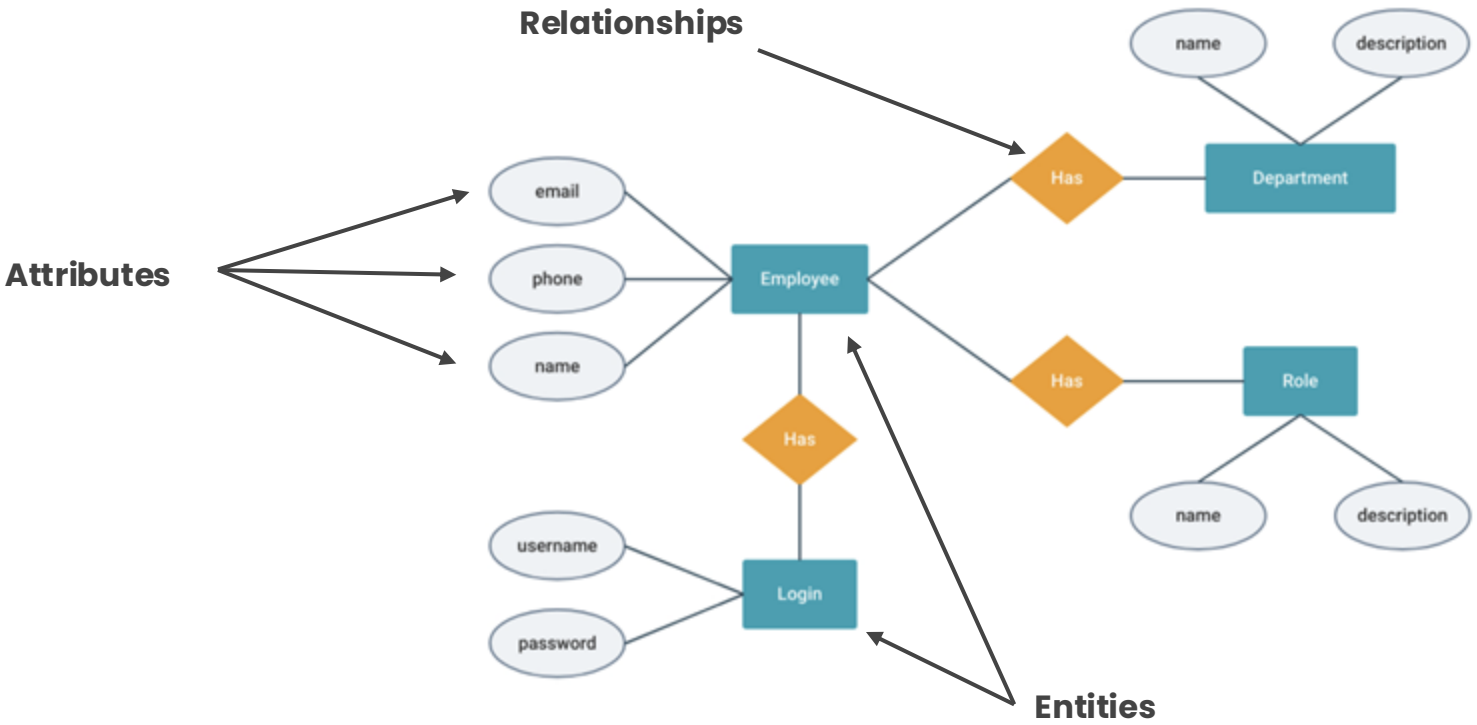
- **Definition:**
  - Visual representation illustrating how entities relate within a database or application.
- **Purpose:**
  - Guide the design of a new system's database or understand an existing system.
- **Setup:**
  - Specific shapes and arrows are used in ERD diagrams to depict the system components and relations.



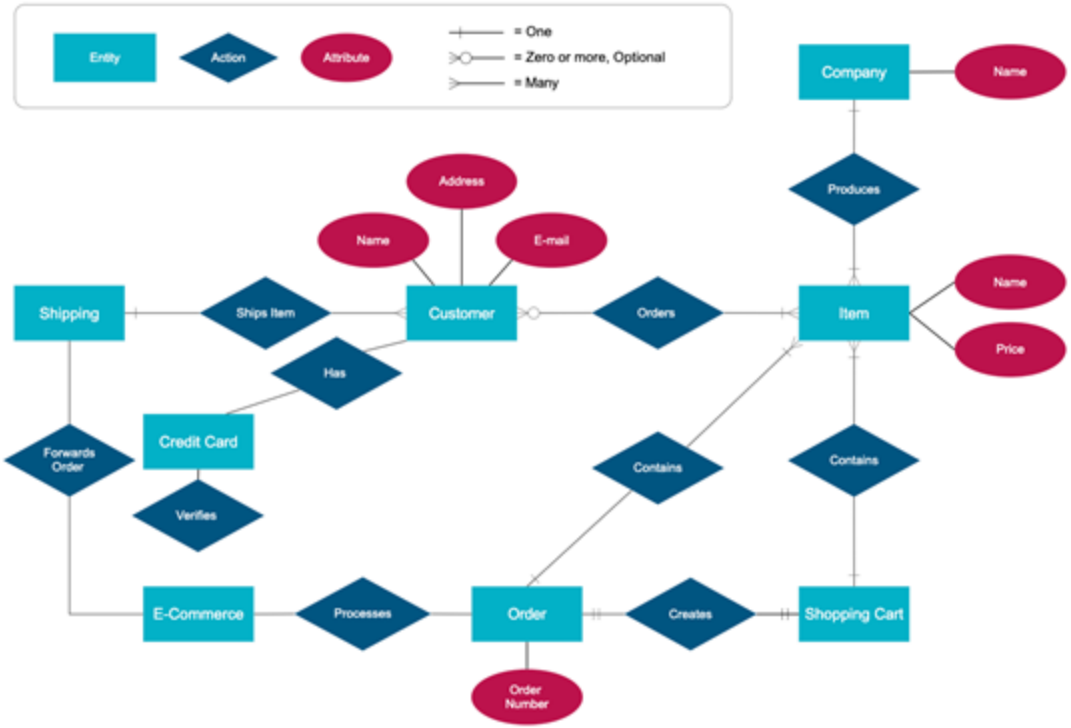
ERD Components

- **Entity:**
  - Represents a physical object, concept, or event. Exemplifies nouns in database conversations.
- **Relationships:**
  - Defines interconnections between entities.
  - Derived from verbs in database contexts.
    - One to one, one to many, many to many
- **Attributes**:
  - Describe properties of an entity.
  - Types:
    - **Simple**: Non-divisible attribute (e.g., first name).
    - **Composite**: Divisible attribute (e.g., full name).
    - **Derived**: Calculated attribute (e.g., age).
    - **Single/Multi-value:** Attributes with one or multiple entries.




Relationships
Attributes
Entities
email
phone
name
Employee
Has
Department
name
description
Has
Role
name
description
Has
username
password
Login

ERD Relationship Diagram: Internet Sales Model



Normalization

- **Purpose:**
  - Design technique to minimize data redundancy and avoid unwanted anomalies.
- **Process:**
  - Organizing columns and tables to decrease data redundancy.
- **Forms:**
  - From 1NF (First Normal Form) to 5NF (Fifth Normal Form) and beyond.

Data modeling lays the groundwork for structuring and defining relationships between data entities

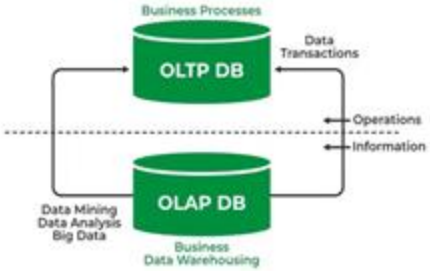
Normalization refines this structure, ensuring data is organized in the most efficient and consistent manner possible

In normalized form, data is stored in multiple tables, reducing data redundancy and inconsistency, thus achieving data integrity. In the denormalized form, data is stored in a limited number of tables (maybe a single table) to reduce querying time.

Data Warehouse

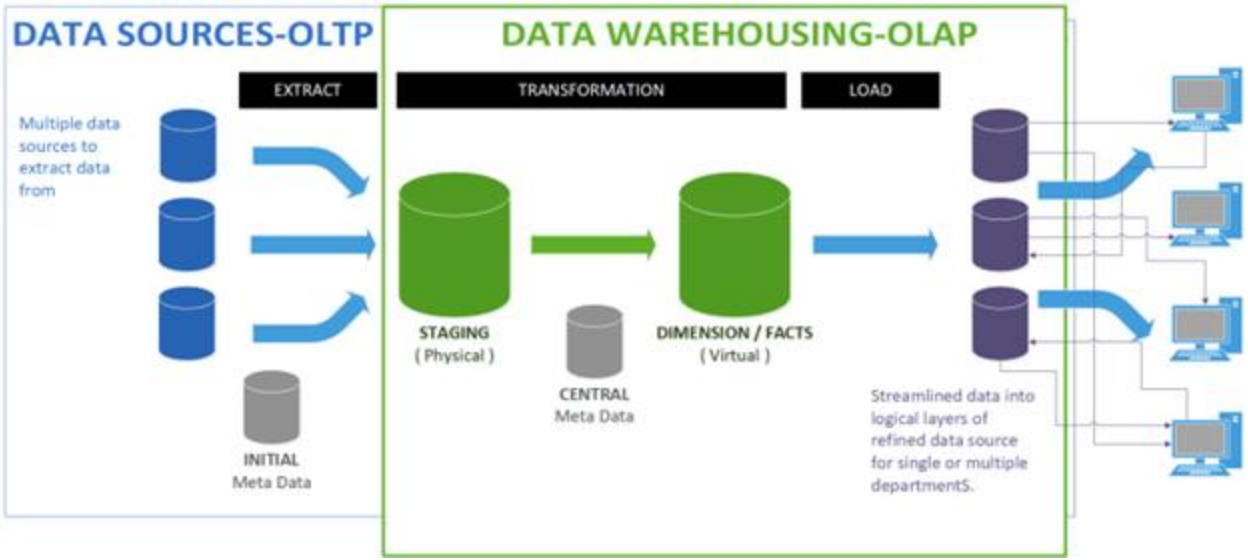
- A **central repository** of integrated data collected from disparate sources.
- Purpose**:** To support **business intelligence activities, particularly analytics and reporting.**
- Amazon Redshift, Google BigQuery, IBM Db2 Warehouse, Microsoft Azure Synapse, Oracle Autonomous Data Warehouse, Snowflake, Teradata Vantage etc.

Database vs Data Warehouse



Database	Data Warehouse
Designed to **record** data	Designed to **analyse** data
Stores **detailed** data	Stores **summarized** data
Used for **OLTP** (Online Transactional Processing)	Used for **OLAP** (Online Analytical Processing)
Performs fundamental business operations and transactions	Allows users to analyse business data
Data is available in real time	Data must be refreshed when needed
Application-oriented data collection	Subject-oriented data collection
Limited to a single application	Draws data from a range of other applications

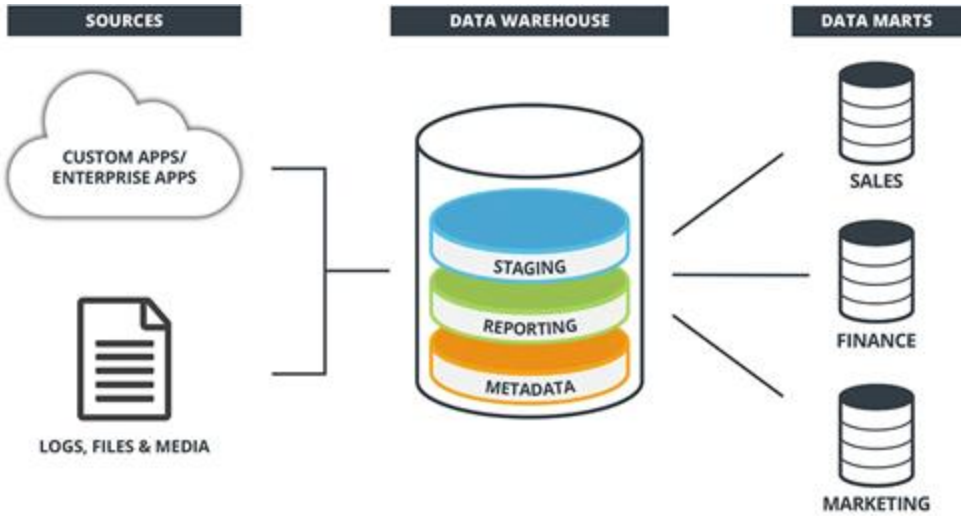
OLAP vs OLTP



Online analytical processing (OLAP) and online transaction processing (OLTP) are two different data processing systems designed for different purposes. OLAP is optimized for complex data analysis and reporting, while OLTP is optimized for transactional processing and real-time updates.

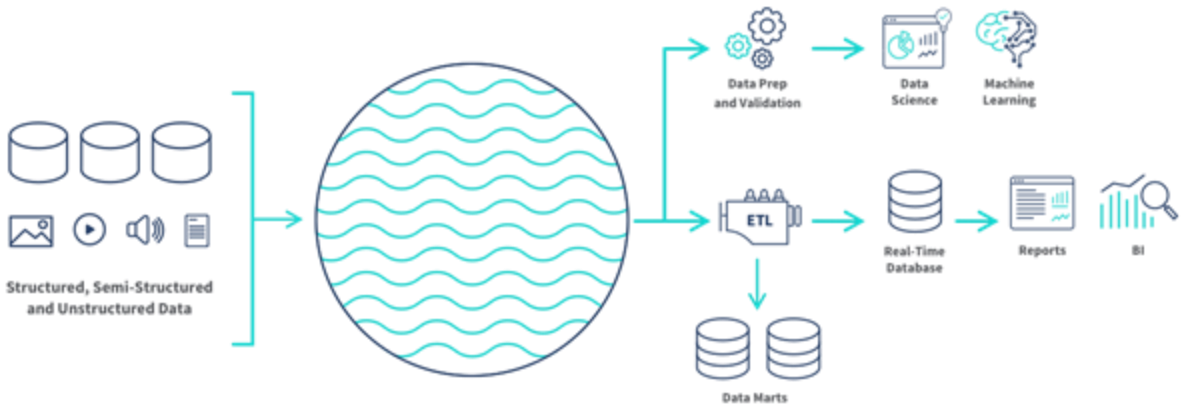
Data Marts

- A subset of a data warehouse designed to **serve a specific business unit.**
- Allows for quicker and more focused analysis.



Data Lakes

- A centralized repository that allows you to **store massive** amounts of **raw** data in various formats (structured and **unstructured** data) supporting **big data** and **real-time** analytics. Highly **scalable and flexible**.
- On premise: **HDFS.** Cloud: **Amazon S3, Azure Data Lake, Google Cloud Storage etc.**

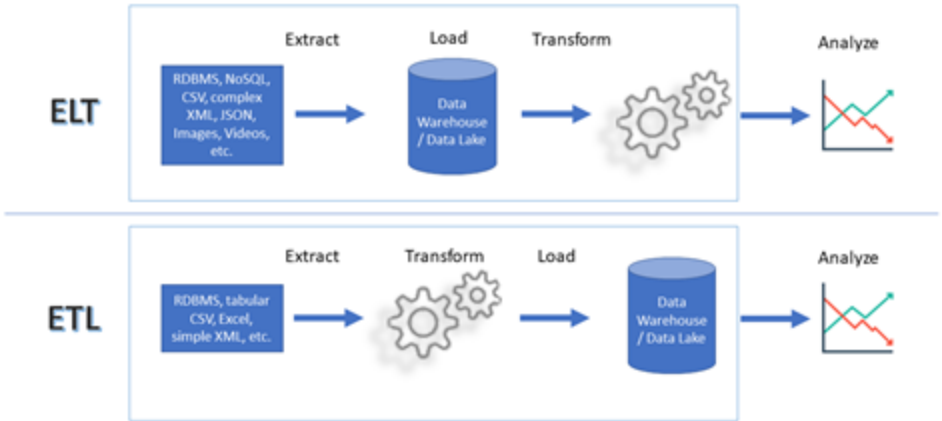


Data Warehouse vs Data Mart vs Data Lake

- **Data Warehouse:** a **centralized repository** for all data, optimized for **analytical** processing. Data is **structured**.
- **Data Mart:** a subset of a data warehouse, focused on **specific business** areas.
- **Data Lake:** a vast storage repository that holds **raw** data (structured and unstructured) in its native format.

ETL (Extract, Transform, Load) & ELT (Extract, Load, Transform)

- **ETL (Extract, Transform, Load): extracts** data from sources, **transforms** it into the desired format, and then **loads** it into the target database or data warehouse.
- **ELT (Extract, Load, Transform):** data is first **loaded** into the data warehouse and then **transformed**. This often leverages the power of modern data warehouses.

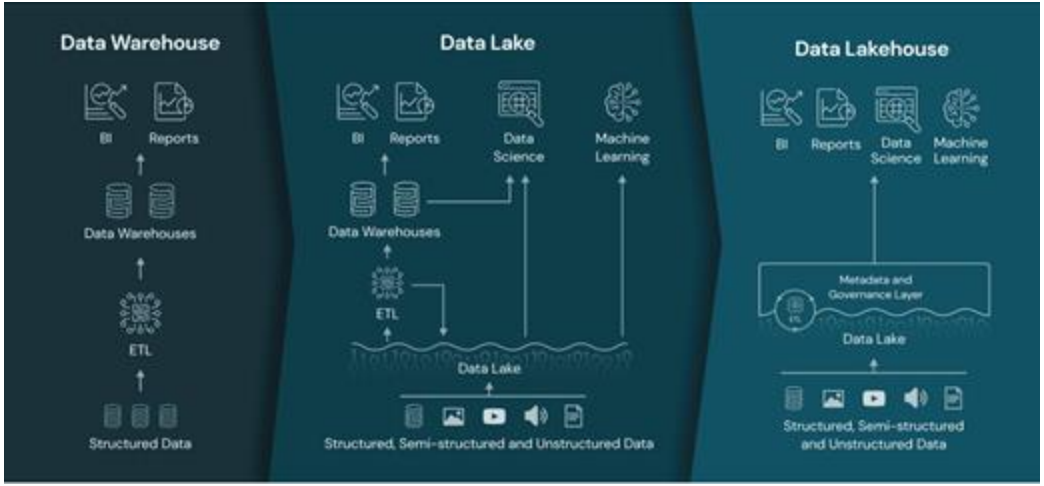


Both ETL & ELT aim to consolidate data from different sources, ensuring it's cleaned, transformed, and made ready for analysis

More Buzzwords: Delta Lake, Lakehouse

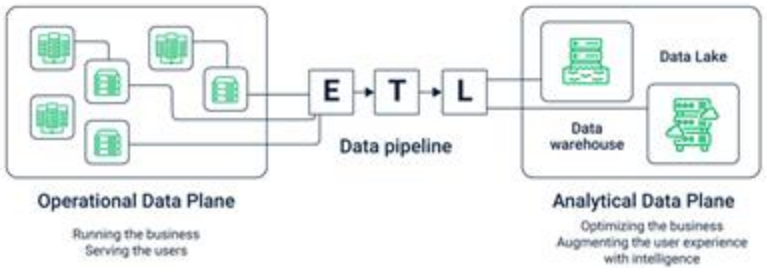
- **Delta Lake:** is an open-source storage layer that brings reliability and ACID transactions to Data Lakes, unifying batch and streaming data processing on top of existing data lakes, such as S3, ADLS, GCS, and HDFS.

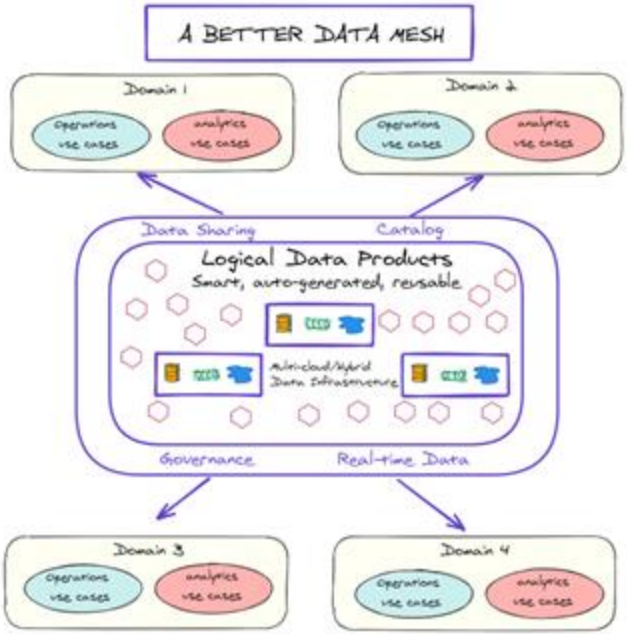
- **Lakehouse**: is a hybrid data architecture that combines the capabilities of a data lake and a data warehouse



More Buzzwords: Data Mesh...

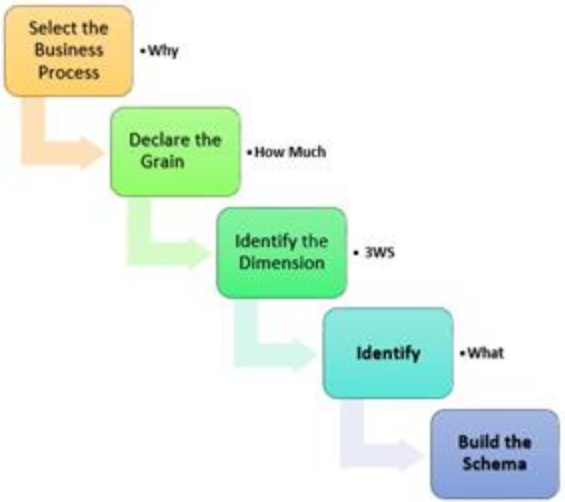
- **Data Mesh:** an architectural and organizational approach where data ownership and delivery are **decentralized** across domain-specific, cross-functional teams.
  Principles:
  - Data Ownership
  - Data as a Product
  - Self-service data information
  - Federal decision-making model





Dimensional Modeling

- **Definition:**
  - A design technique tailored for data warehousing processes
  - Focuses on optimizing the organization and presentation of data for querying and reporting purposes.
- **Core Concepts:**
  - **Facts**: Quantifiable data, often central to a business operation, such as sales figures, revenue, or the number of products sold.
  - **Dimensions:** store the description or textual information related to the business process, like who bought the products.



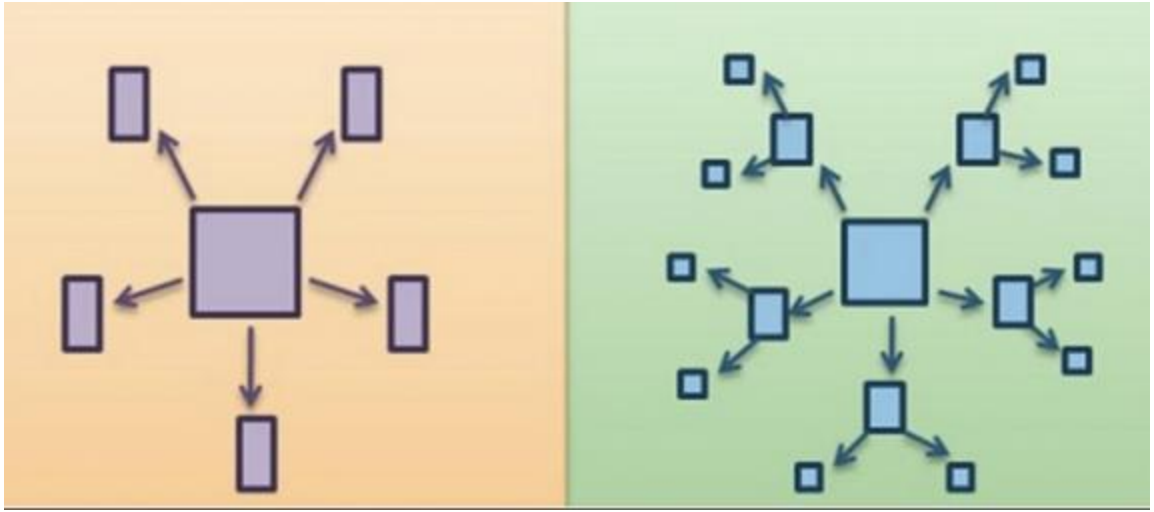
Star & Snowflake Schemas

- **Star Schema:**
  - A simplified, denormalized structure, featuring one large central fact table connected to surrounding dimension tables.
  - Example: Sales fact table linked to dimensions like time, customer, and product.

- **Snowflake Schema Definition:**
  - A more normalized structure, with the fact table connected to multiple related dimension tables, which are further split into related sub-dimension tables.
  - Example: Sales fact table linked to detailed customer, product, and time dimensions, further divided.

In data warehouses, star and snowflake schemas are popular architectures used to organize data for efficient querying and reporting

Star & Snowflake Schemas



Star schema is simpler and faster for queries but might use more storage

Snowflake schema is more complex but provides detailed analysis due to additional layers of related data.

Star & Snowflake Schemas Example