inceptez
Web: Inceptez.com Mail: info@inceptez.com Call: 7871299810, 7871299817

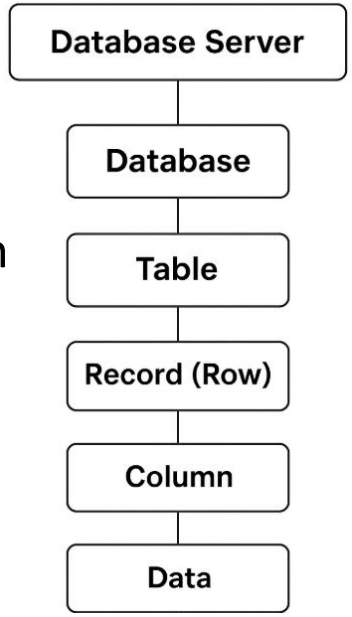
Data Warehouse

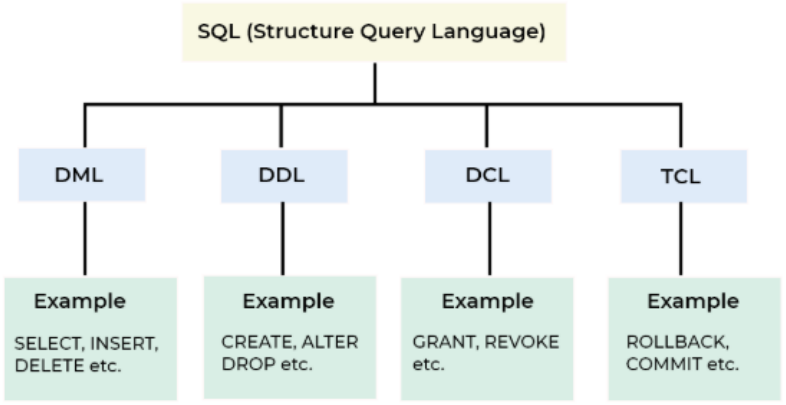
Topics

- Database
- Data Warehouse
- ETL
- Data Modeling
- Dimensional Modeling
- SCD
- Architectural Patterns

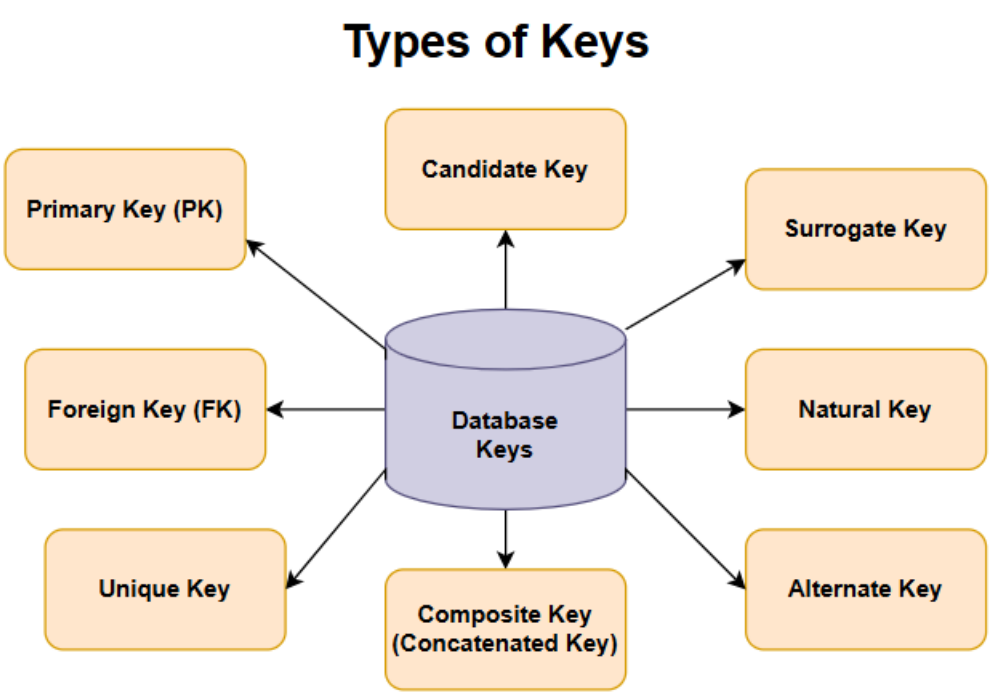
Database

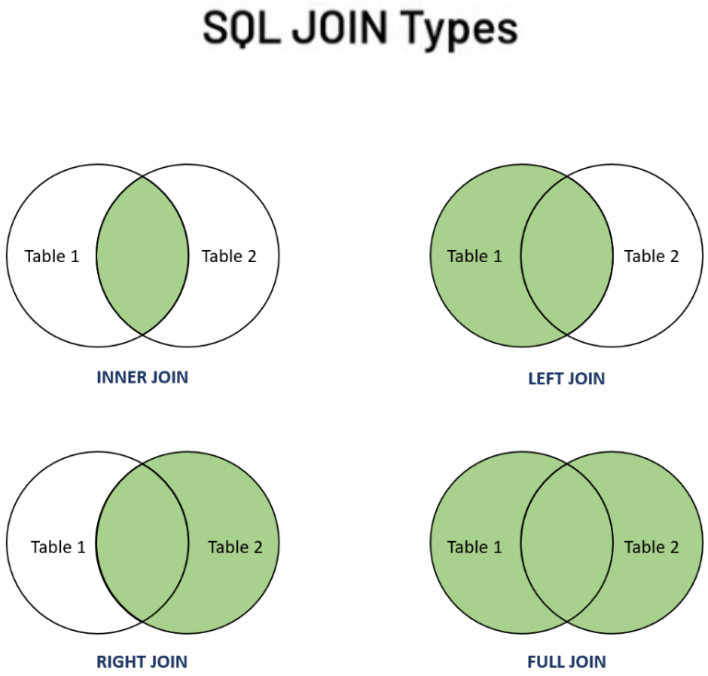
- A database is a place where data is stored and organized
- Enables efficient storage, retrieval, and management of information
- SQL is the language used to interact with and manage that data





Database Keys & Joins





Database Constraints

Primary Key

CREATE TABLE Employees
INT PRIMARY KEY;

Foreign Key

CREATE TABLE Orders
PRIMERY KEY, EmployeeID INT
REFERENCES REFERENCES

Unique

CREATE TABLE Users
VARCHAR(255) UNIQUE);

Not Null

CREATE TABLE Products
VARCHAR(255) NOT NULL);

Default

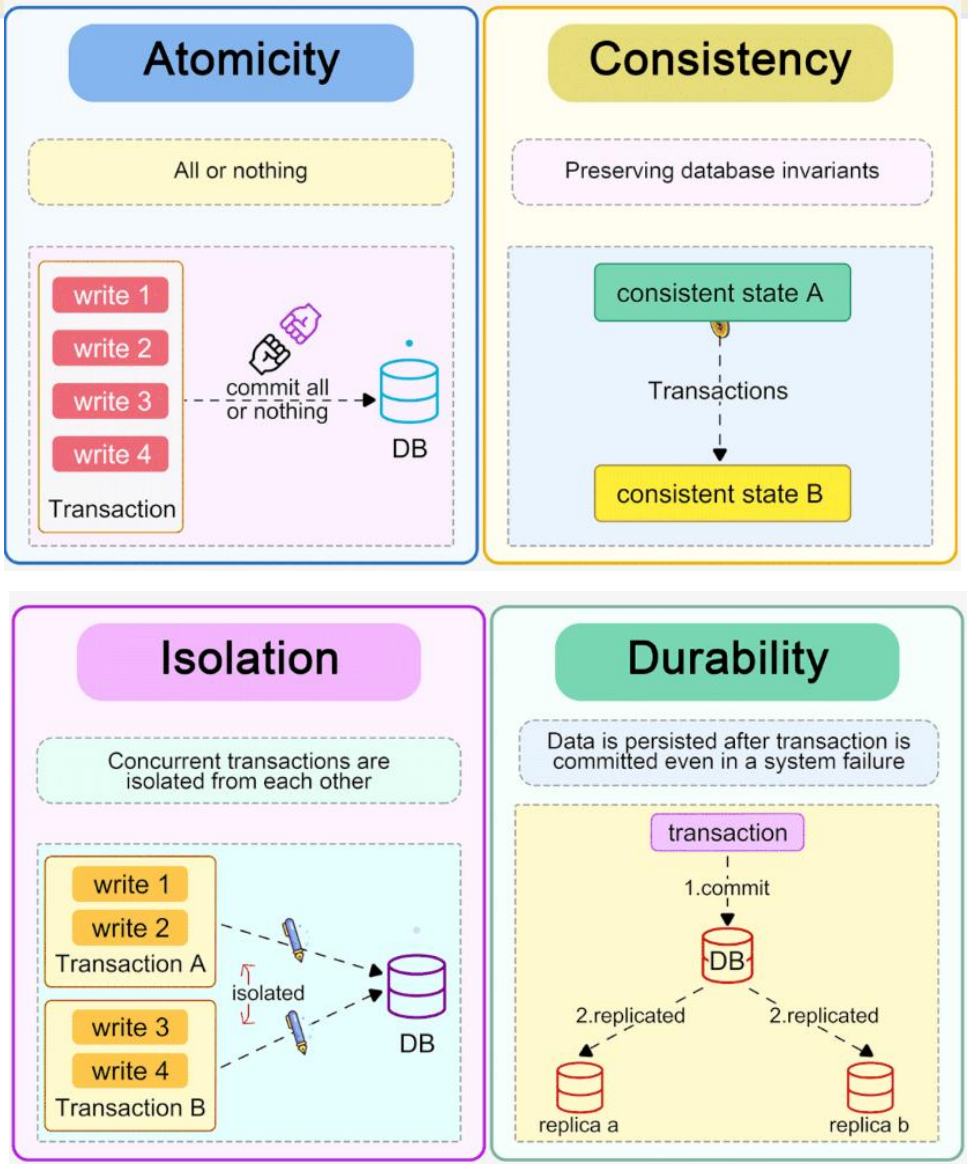
CREATE TABLE Products (Price
DECIMAL(10, 2) DEFAULT 0.00;

Check

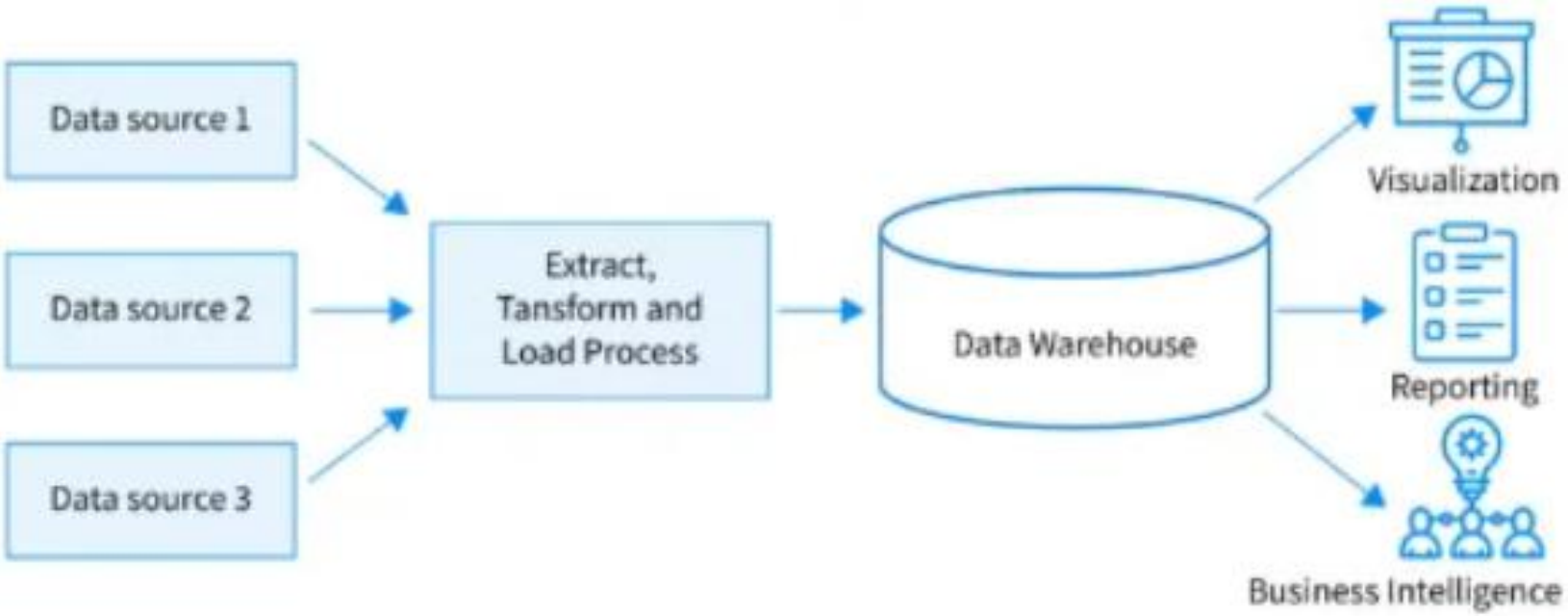
CREATE TABLE Orders INT
CHECK (Quantty > 0);

ACID Properties

ACID properties are the key principles that ensure safe and reliable database transactions.



Datawarehouse



Introduction | Data Warehouse

What is a Data Warehouse?

- A centralized repository of integrated data from one or more different sources.
- Designed for reporting and data analysis, and is a core component of business intelligence.
- It stores historical data and is optimized for analytical queries rather than transactional processing.

Purpose:

- Enable informed decision-making.
- Provide a single source of truth for business data.
- Support trend analysis, forecasting, and data mining.

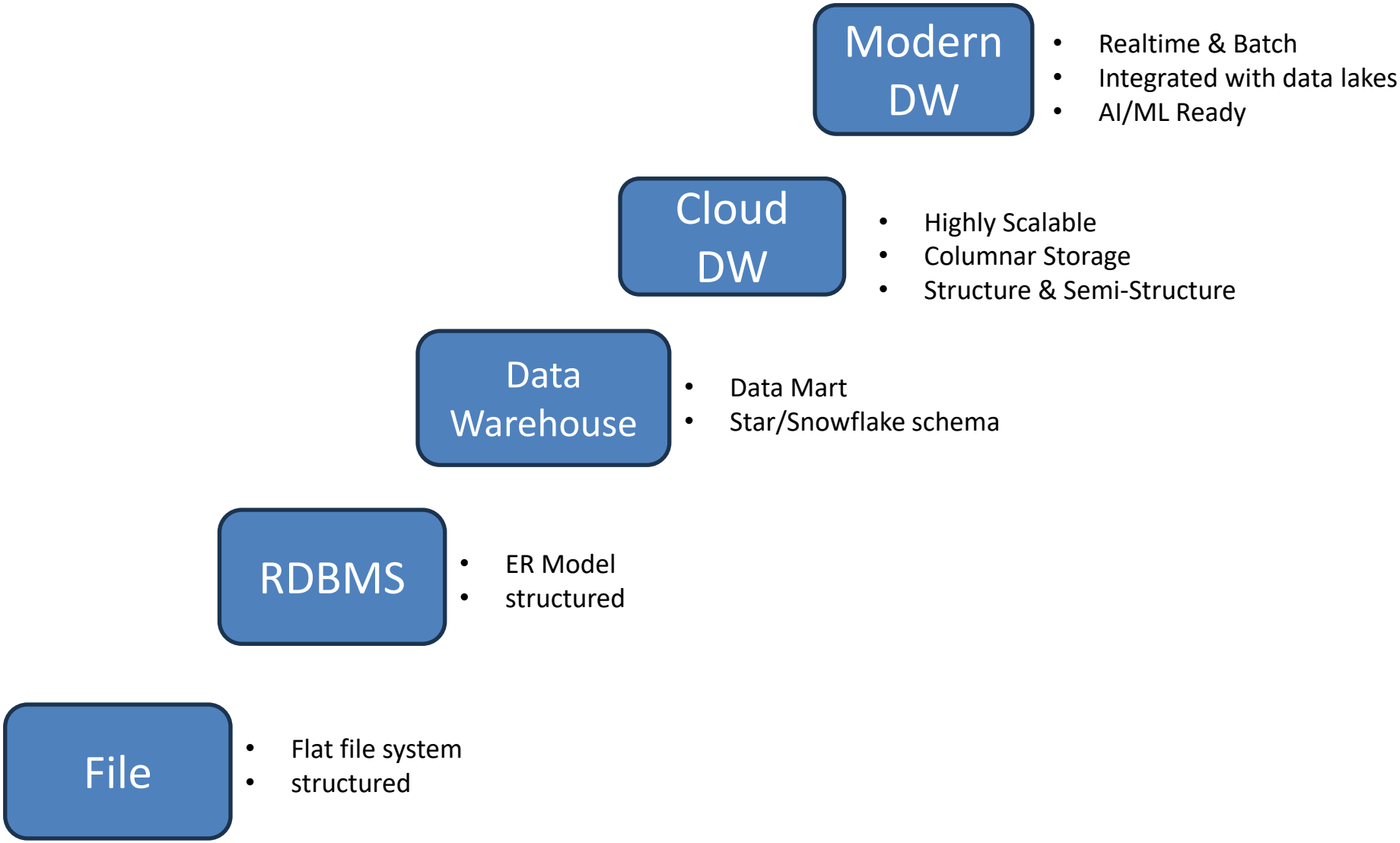
Key Characteristics:

- **Subject-Oriented:** Organized around major subjects of the enterprise (e.g., sales, customers, products).
- **Integrated:** Data is collected from various sources, cleaned, and integrated into a consistent format.
- **Time-Variant:** Data includes a time dimension, allowing for historical analysis.
- **Non-Volatile:** Once data is in the data warehouse, it typically does not change.

Database vs Data Warehouse

Feature	Database (OLTP)	Data Warehouse (OLAP)
Purpose	Daily operations	Analysis & reporting
Data Type	Real-time, current	Historical, summarized
Schema	Normalized	Denormalized
Query Type	Simple, frequent	Complex, aggregated
Users	Operational staff	Analysts & management
Operations	CRUD	Read-heavy (ETL/ELT)

Evolution of Datawarehouse

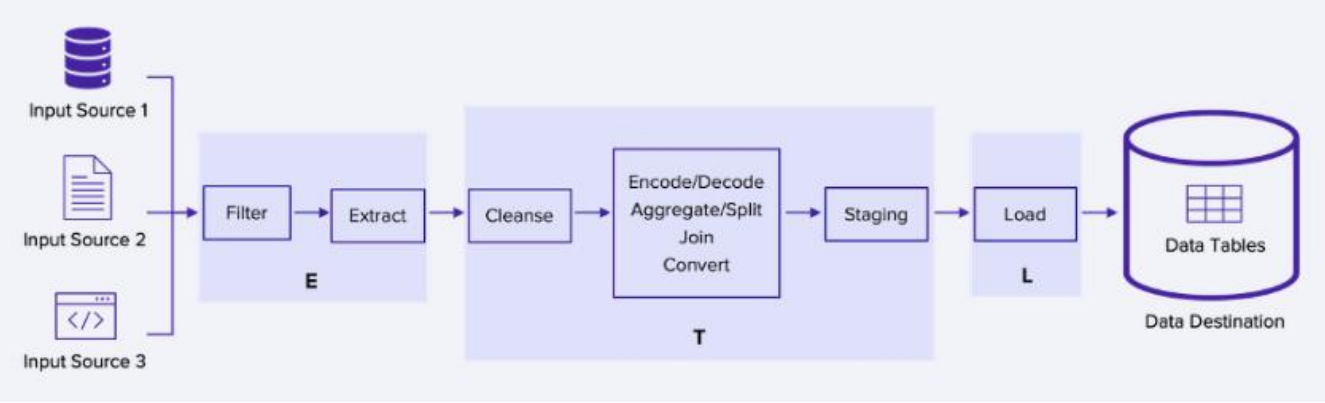


ETL

- Data is extracted from source systems, transformed into the desired format/structure outside the target system, and then loaded into the data warehouse.
- Transformations are done before loading the data.

When to Use:

- When the target system is not powerful enough to handle large-scale transformations.
- For traditional data warehouses with strict schema requirements (e.g., on-premise systems).



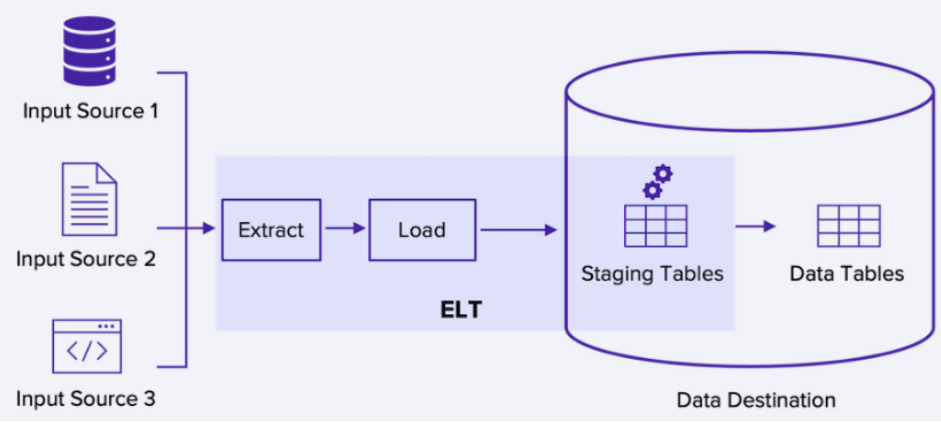
Fivetran
Informatica
IBM InfoSphere DataStage
SSIS
Talend
Oracle Data Integrator

ELT

- Data is extracted from source systems, loaded directly into the target (data warehouse or data lake), and then transformed inside the target system.
- Transformations are performed after loading using the processing power of the warehouse/lake.

When to Use:

- For cloud-based or modern data warehouses that can handle large-scale transformations (e.g., BigQuery, Snowflake, Redshift, Hadoop Hive).
- When raw data storage is needed for future transformations.



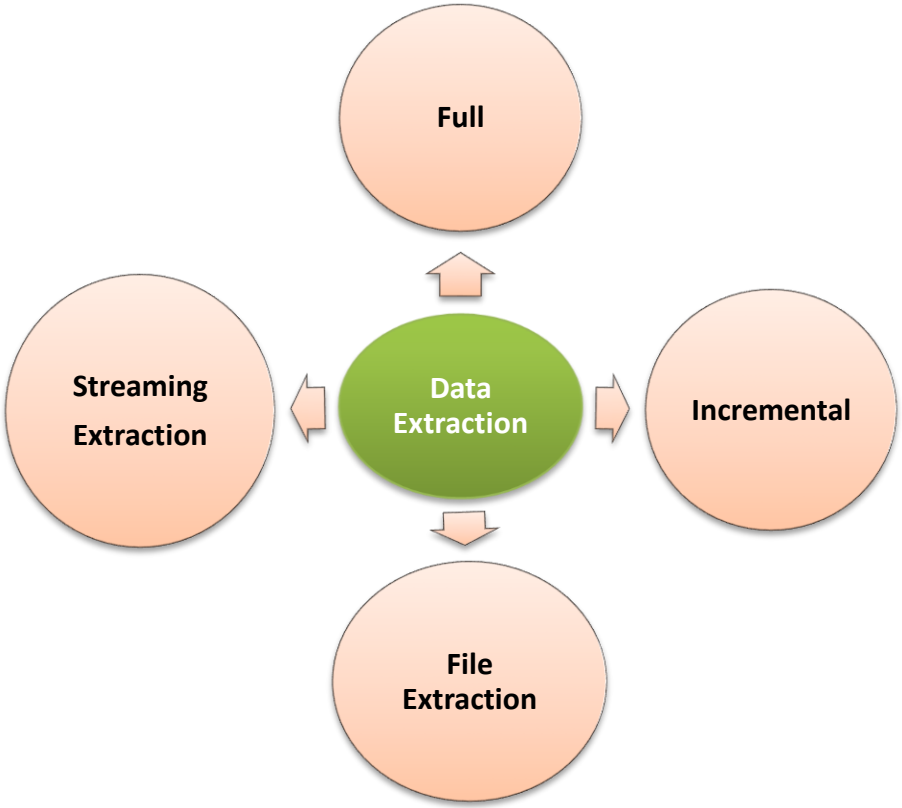
ETL and ELT

Feature	ETL (Schema-on-Write)	ELT(Schema-on-Read)
Transformation	Done before loading to the data warehouse	Done after loading into the data warehouse
Target System	Traditional data warehouses(on-prem)	Modern cloud-based data warehouse or data lake
Data Volume	Suited for small to medium dataset	Handle large-scale, high-volume dataset
Performance	Limited by ETL server or transformation engine	Leverage Datawarehouse or lake compute power
Latency	Typically, higher(Batch Oriented)	Supports near real-time or real-time
Tools	Talend, Informatica, SSIS, Pentaho, Datastage, Spark	Fivetran, Azure data factory, dbt, GCP Dataflow, Spark

Data Extraction

Data extraction is the process of retrieving data from various source systems (databases, websites, files, APIs, etc.) to use it for further processing like analysis, transformation, or loading into another system.

Method of Data Extraction



Data Transformation

Data Transformation: Converting data into a usable format for analysis, storage, or integration. Ensures data consistency, accuracy, and usability across systems.

Ex:

- Changing date format from MM/DD/YYYY to YYYY-MM-DD.
- Combining First Name and Last Name into Full Name.

Data Cleaning: Identifying and correcting errors, inconsistencies, and inaccuracies to improve data quality. Includes removing duplicates, fixing missing or incorrect values, and standardizing data.

Ex:

- Removing duplicate customer records.
- Correcting typos (e.g., 'Califronia' to 'California').
- Filling missing values.

Data Enrichment: Enhancing existing data by adding relevant information from internal or external sources. Provides deeper insights and more complete data profiles.

Ex:

- Adding geographic coordinates based on address.
- Appending demographic data like age or income.
- Including social media sentiment scores.

Data Load

The process of moving transformed data into a target system like a data warehouse, database, or data lake. Final step in ETL/ELT pipelines to make data available for analysis and reporting.

Data Load Strategies

Batch Load :

- Loads data in scheduled batches (e.g., nightly, hourly).

Real-time / Streaming Load:

- Continuous loading of data as it arrives, supporting near real-time analytics.

Upsert (Update + Insert):

- Updates existing records and inserts new ones in the target.

Overwrite:

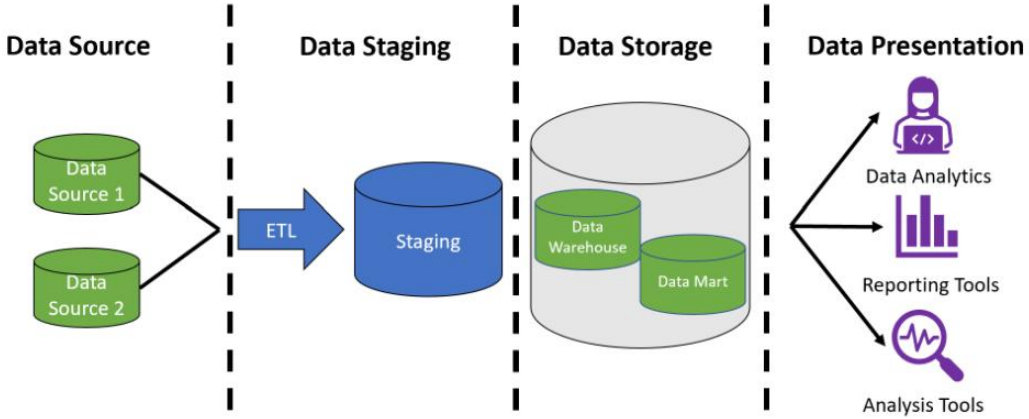
- Completely replaces data in the target table or partition.

Partition Load:

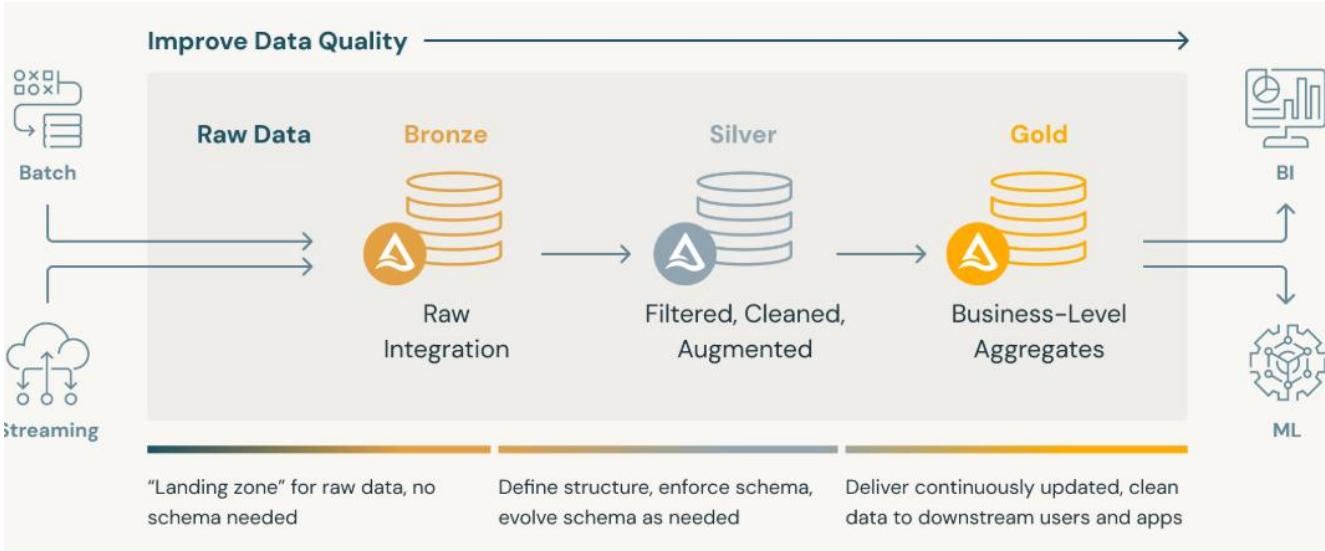
- Loads data into specific partitions (e.g., by date) to optimize performance.

Layers of DW Architecture

Traditional Architecture



Medallion Architecture(Modern DW)



Data Modeling

Definition

- Data modelling is the process of defining and structuring data to be stored, managed, and used efficiently within a database or data system
- It defines data elements, their relationships, and rules for how data flows between different parts of a system.

Purpose:

- Helps in understanding business requirements and translating them into database structures.
- Ensures consistency, accuracy, and efficiency in data management.
- Provides a clear framework for developers, analysts, and stakeholders.

Key Components:

- Entities (e.g., Customer, Order)
- Attributes (e.g., Name, Order Date)
- Relationships (e.g., One-to-Many, Many-to-Many)
- Constraints

Types of Data Modeling

Conceptual Data Model:

High-level view that defines business entities and relationships. Used for business stakeholders.

Logical Data Model:

More detailed, specifying attributes, keys, and relationships without focusing on physical storage.

Physical Data Model:

Defines how data is stored in a database, including tables, columns, indexes, partitions, and data types.

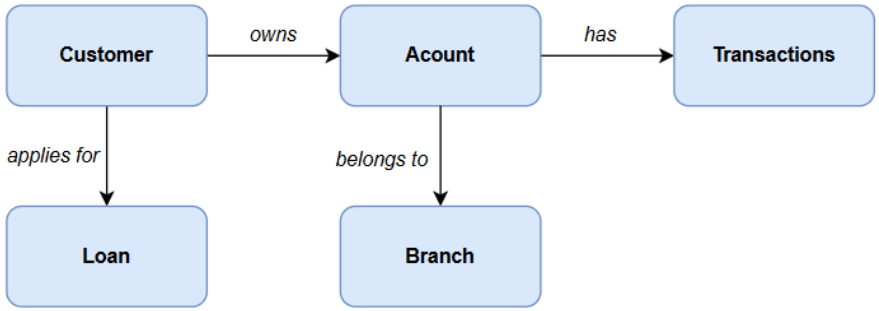
Conceptual Data Model

Definition:

- A high-level representation of organizational data.
- Focuses on what data is needed (entities, attributes, and relationships) rather than how it will be implemented.
- Abstract, business-oriented view = mainly for business stakeholders and analysts.

Characters:

- Define entities and relationships (how entities are linked).
- Contains minimal details (usually no primary keys, foreign keys, or data types).
- Independent of database technology.



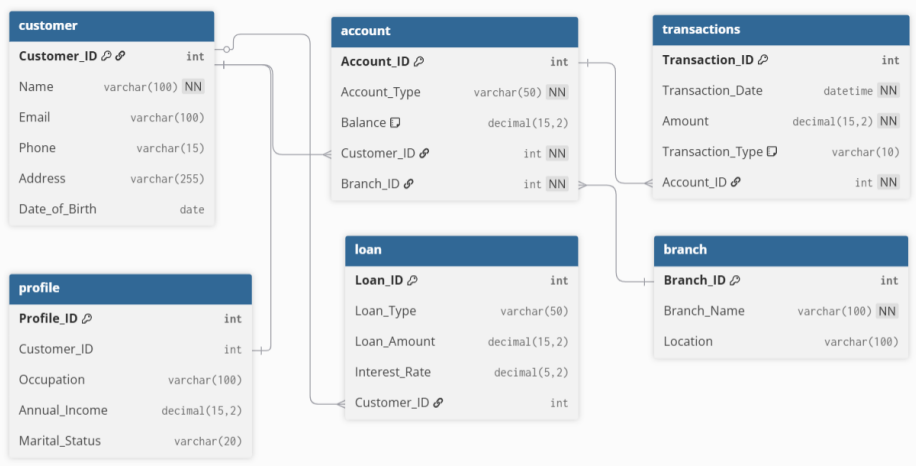
Logical Data Model

Definition:

- A detailed representation of data that describes the structure of data elements and the relationships between them
- Adds more detail compared to the Conceptual Data Model, but still independent of physical implementation (e.g., not tied to MySQL, Oracle, etc.).

Characters:

- Includes entities, attributes, and relationships.
- Specifies primary keys (PKs) and foreign keys (FKs).
- Defines cardinalities (1:1, 1:N, M:N).
- Independent of DB(index, partition , etc.)



Physical Data Model

Definition:

- The database-specific representation of the logical model.
- Includes tables, columns, data types, constraints, indexes, and relationships.
- Tied to a specific DBMS (MySQL, PostgreSQL, Oracle, etc.).
- Defines datatypes, constraints, index, partitions, etc

```
CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15)
);
```

Characters:

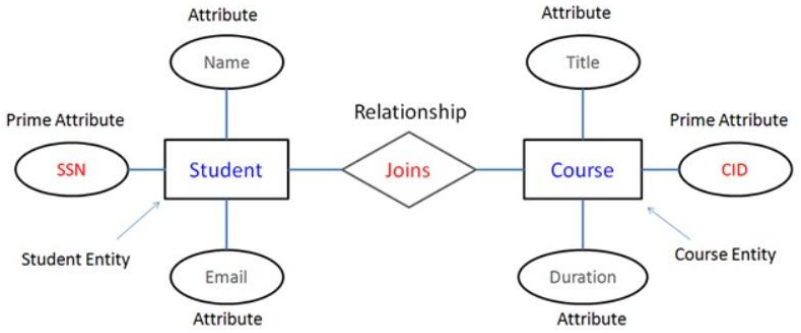
- Converts entities → tables.
- Converts attributes → columns with data types.
- Adds primary keys (PK), foreign keys (FK), unique constraints, indexes.
- Includes data types, field sizes, default values, NULL/NOT NULL.
- Represents physical storage & performance optimizations.

Entity Relationship Model

- ER model is graphical representation of database structure.
- Shows how entities (things) in a system are related.
- Widely used in database design.
- Helps in planning, communication, and implementation of data structures.

Entity:

- Represents a real-world object or concept.
- Each entity has attributes.
- Example: Customer, Branch, Account.



Attributes:

- Properties that describe an entity.
- Example: Customer → Name, Email, Phone.

Relationship:

- Define how entities are connected to each other.

Types of Relationships

One to One Relationship(1:1)

One instance of an entity is associated with exactly one instance of another entity.

Example: Each person has one passport, and each passport is assigned to one person.

Person -> Passport

One to Many Relationship(1:M)

One instance of an entity is associated with multiple instances of another entity.

Example: A teacher can teach many courses, but each course is taught by only one teacher.

Teacher -> Course

Many to Many Relationship (M:M)

Multiple instances of an entity are associated with multiple instances of another entity.

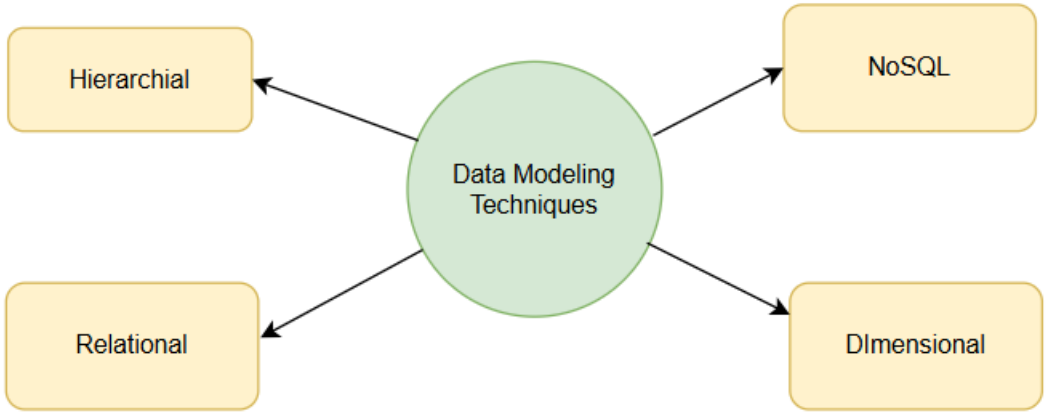
Example: Students enroll in many courses, and each course can have many students.

Student -> Course

Types of Relationship

Data Modeling Techniques

- **Hierarchical Modeling**
- **Relational Modeling**
- **NoSQL Data Modeling**
- **Dimensional Modeling**



Choosing the Right Technique

Transactional systems → Relational Modeling.

Analytics & Reporting → Dimensional Modeling.

High-speed, flexible data → NoSQL.

Legacy Systems → Hierarchical Modeling.

Dimensional Modeling

- Dimensional Modeling is a data design technique used to structure data in a way that's optimized for querying and reporting — especially in data warehouses.
- It focuses on making data easy to understand, fast to retrieve, and flexible for analysis by organizing it into facts and dimensions.

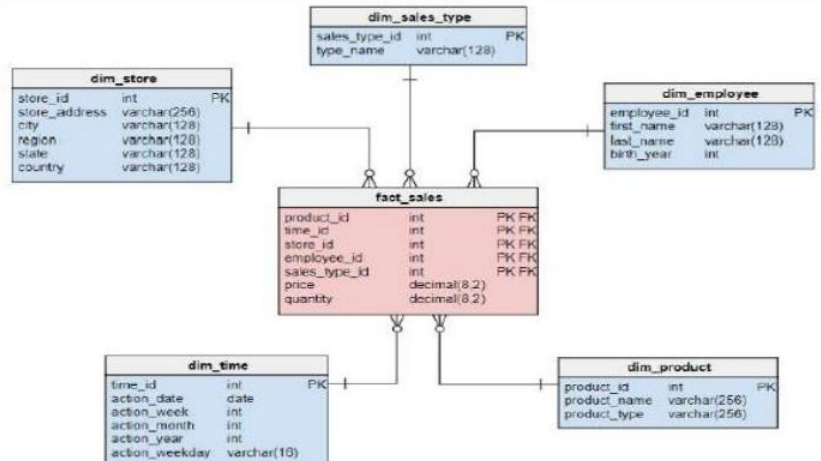
Key Components:

Fact Table

- Central table that contains measurable, quantitative data.
- Stores metrics or facts (e.g., sales amount, revenue, quantity).
- Has foreign keys to dimension tables.

Dimension Table

- Contain descriptive attributes related to facts.
- Answer "who, what, when, where, how".
- Examples: Customer, Product, Time, Region



Types of Fact Tables

Transactional/Granular:

- Stores data at the most granular (detailed) level for each event or transaction

Ex: Each online purchase is a row.

Granularity: One row per order line item.

Columns: Order_ID, Date_Key, Customer_Key, Product_Key, Quantity, Price, Discount.

Periodic Snapshot:

- Store aggregated data at predefined intervals (daily, weekly, monthly, etc.)
- Provides a "snapshot" of the state of the business at a specific point in time.

Ex: Daily sales performance by category. Captured every 24 hours.

Columns: Date_Key, Category_Key, Total_Sales, Total_Orders, Total_Returns.

Accumulating:

- Tracks the lifecycle of a process or workflow.
- Each row represents a single process instance, and it gets updated as the process progresses

Ex: Order delivery tracking. Shows how long each stage takes

Columns: Order_ID, Order_Date, Payment_Date, Shipped_Date, Delivered_Date, Returned_Date.

Types of Dim Tables

Conformed Dimension:

- Shared across multiple fact tables or multiple subject areas. Ensures consistency in reporting

Ex: Customer Dimension used by both sales' fact and Support ticket fact

Role Playing Dimension:

- One dimension used for multiple roles in the same fact table.

Ex: Date Dimension used as: Order_Date, Ship_Date, Delivery_Date

Junk Dimension:

- Combines miscellaneous low-cardinality attributes into one table to reduce clutter

Ex: Flag_ID, Gift_Wrap, Order_Channel,Payment_Status. Combine all columns into single Dim table

Degenerate Dimension:

- Dimension attribute stored inside the fact table itself, no separate dimension table.

Ex: Order Number in a sales fact table (not stored in a separate table)

Measures and Its Types

- Measures are numerical values (metrics) in a fact table that analyze or aggregate in reports.
- Usually quantitative in nature

  Ex: *Sales Amount, Quantity Sold, Profit, Discount, Order Count.*

Types of Measures in a Fact Table

- **Additive Measures** - Can be summed up across all dimensions (time, location, product, etc.)

  Ex: *Sales Revenue, Quantity Sold*

- **Semi-Additive Measures** - Can be summed across some dimeneions but not all.

  Ex: *accountbalance, stocklevel, loanoutstandingamt*

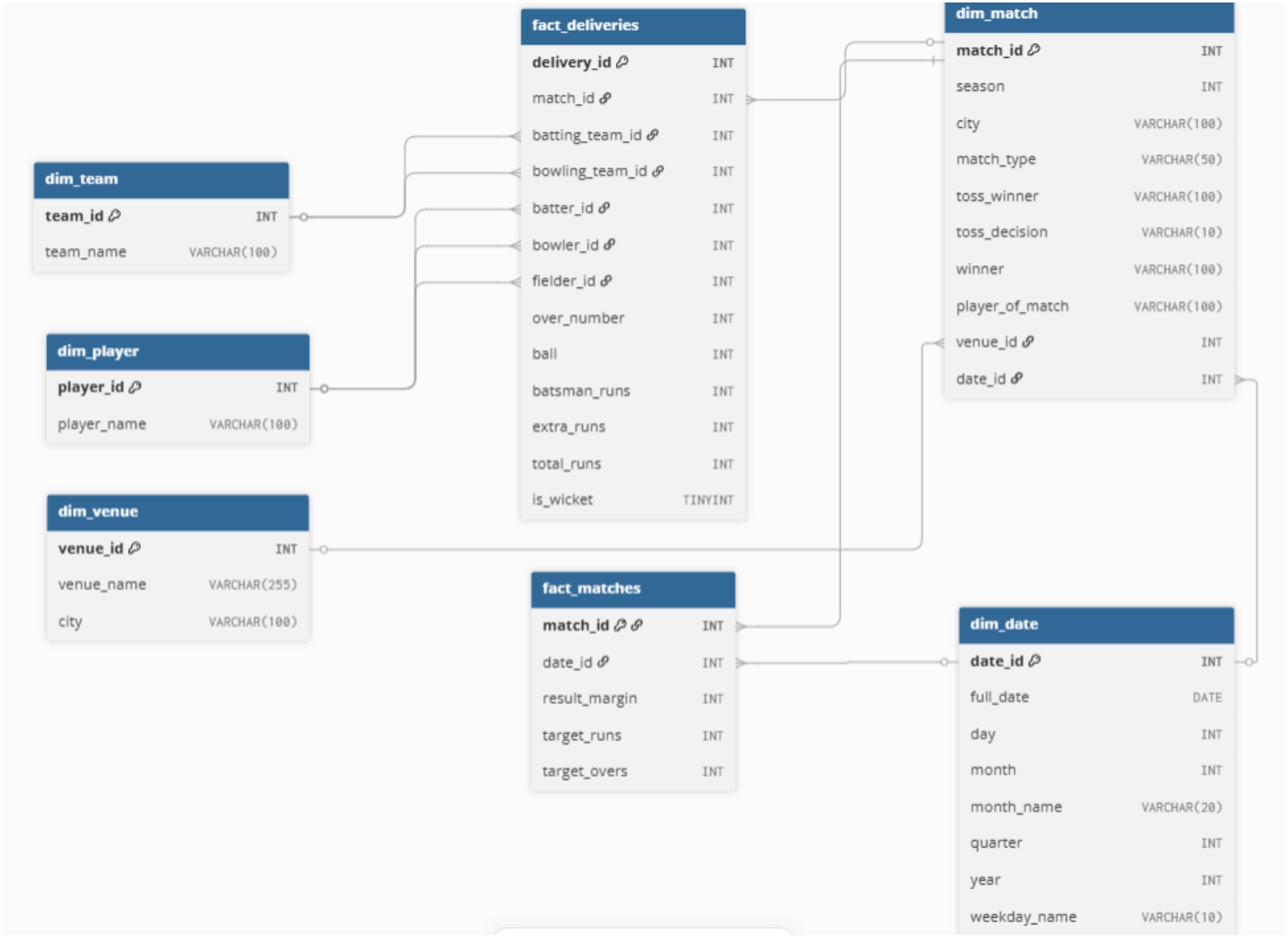
- **Non-Additive Measures** - Cannot be meaningfully summed across any dimension.

  Ex: *average, ratio, min, max, or weighted calculation, profitmarigin, returnrate*

- **Derived/Calculated Measures** – **Not stored directly in the fact table**; calculated from other measures.

  Ex: *Profit = Sales Revenue – Cost; Conversion Rate = Orders ÷ Visits.*

Fact & Dimensions



Dimensional Modeling Concepts

Star Schema:

Definition: A central fact table connected to multiple dimension tables in a star-like shape.

Fact table: Stores numeric measures (e.g., sales amount, quantity).

Dimension tables: Store descriptive attributes (e.g., product, customer, time, location).

Pros: Simple, easy to understand, and efficient for queries.

Cons: Can cause data redundancy.

Fact Table: Sales (SalesID, ProductID, CustomerID, DateID, Amount)

Dimension Tables: Product, Customer, Date, Store

Snowflake Schema:

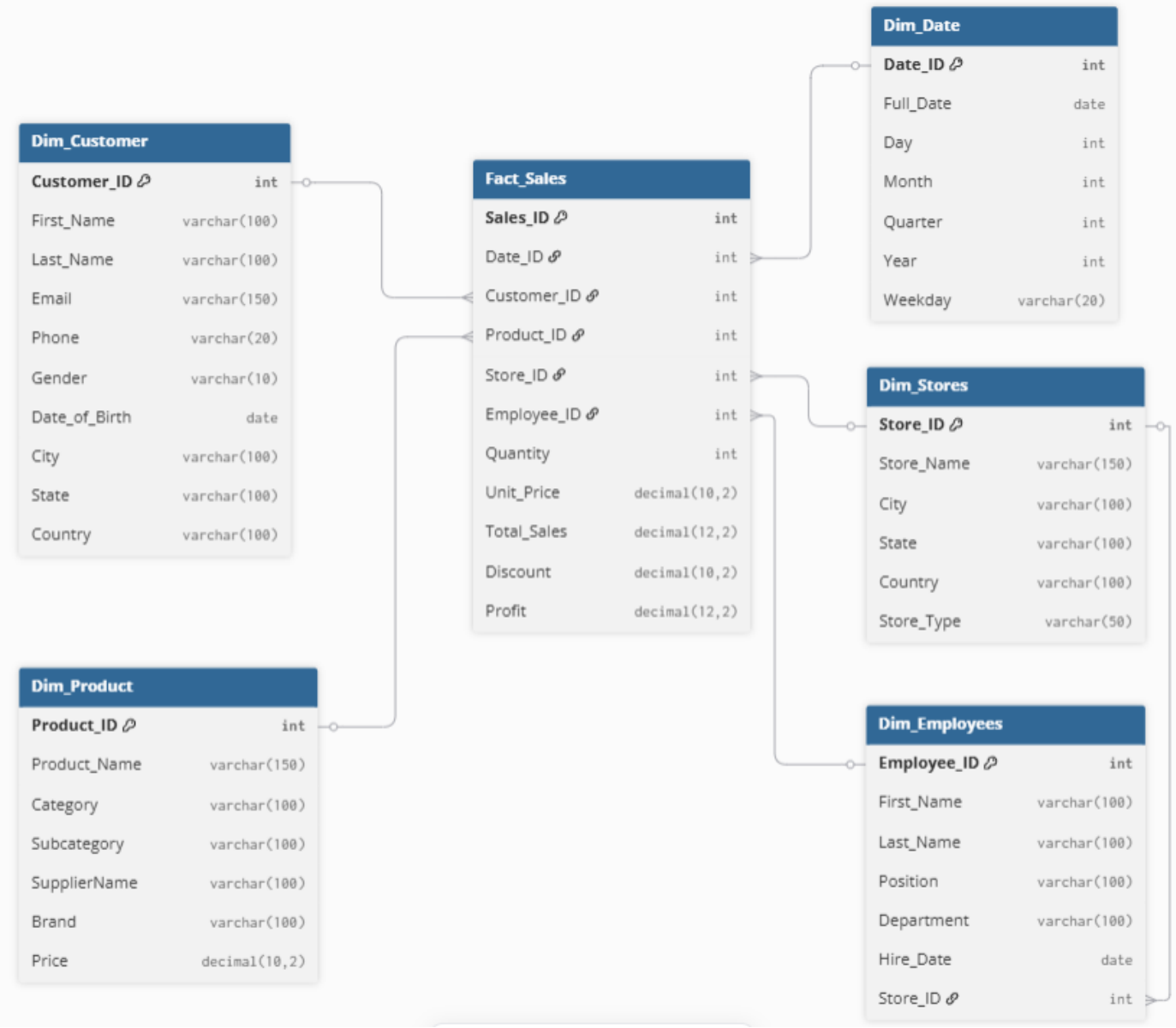
Definition: An extension of the star schema where dimension tables are normalized into multiple related tables.

Pros: Reduces redundancy, saves storage..

Cons: More complex joins, slightly slower queries.

Ex: Instead of storing all product details in one dimension, split into Product, Category, Supplier.

Star Schema


Dim_Customer
Customer_ID int
First_Name varchar(100)
Last_Name varchar(100)
Email varchar(150)
Phone varchar(20)
Gender varchar(10)
Date_of_Birth date
City varchar(100)
State varchar(100)
Country varchar(100)
Fact_Sales
Sales_ID int
Date_ID int
Customer_ID int
Product_ID int
Store_ID int
Employee_ID int
Quantity int
Unit_Price decimal(10,2)
Total_Sales decimal(12,2)
Discount decimal(10,2)
Profit decimal(12,2)
Dim_Date
Date_ID int
Full_Date date
Day int
Month int
Quarter int
Year int
Weekday varchar(20)
Dim_Stores
Store_ID int
Store_Name varchar(150)
City varchar(100)
State varchar(100)
Country varchar(100)
Store_Type varchar(50)
Dim_Product
Product_ID int
Product_Name varchar(150)
Category varchar(100)
Subcategory varchar(100)
SupplierName varchar(100)
Brand varchar(100)
Price decimal(10,2)
Dim_Employees
Employee_ID int
First_Name varchar(100)
Last_Name varchar(100)
Position varchar(100)
Department varchar(100)
Hire_Date date
Store_ID int

Snowflake Schema

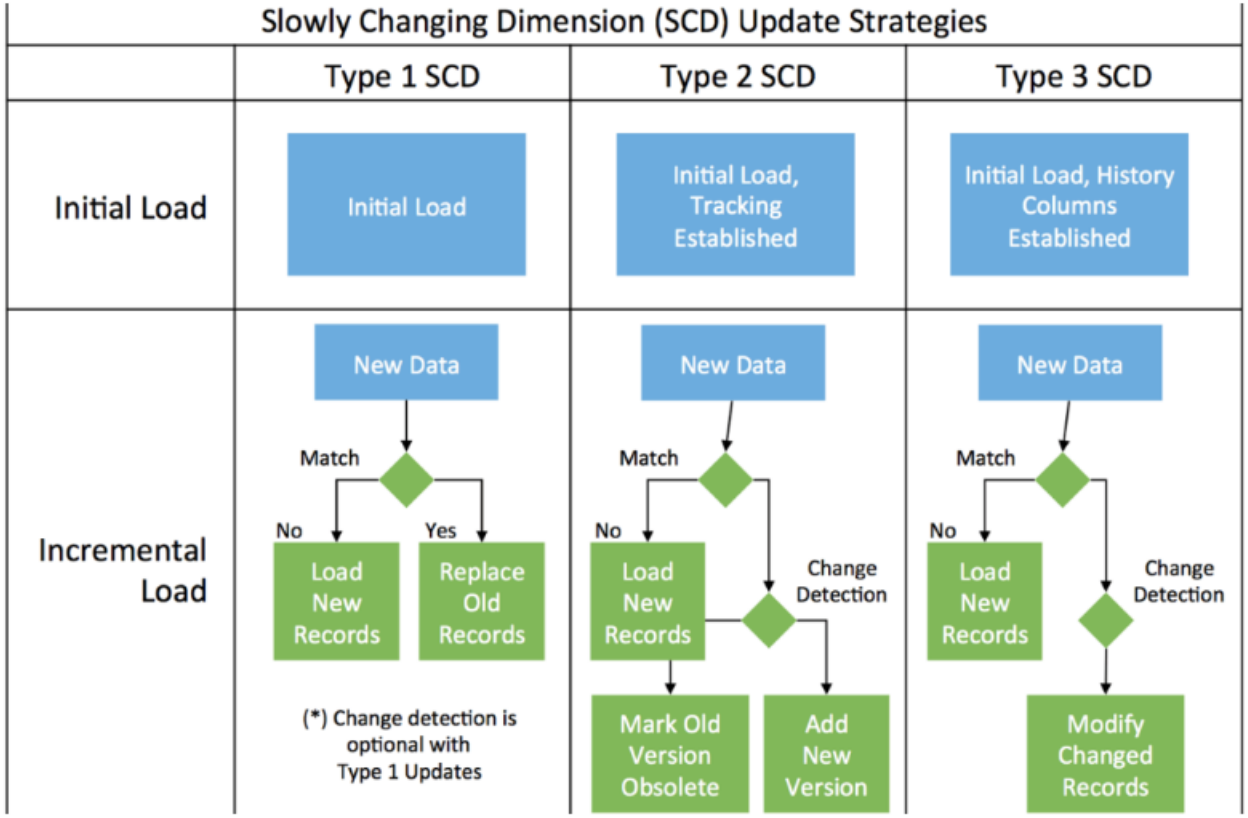
Star vs Snowflake Schema

Feature	**Star Schema**	**Snowflake Schema**
Dimension Tables	Denormalized	Normalized
Joins	Fewer (fast queries)	More (slower queries)
Storage	Higher (data repetition)	Lower (less redundancy)
Design	Simple	More complex
Best For	Best for quick analytics, dashboards, and BI tools.	Best for large-scale data warehouses needing storage optimization and high data integrity.

Key Difference

- Star schema **optimizes for speed and simplicity** (denormalized).
- Snowflake schema **optimizes for storage and consistency** (normalized).

Slowly Changing Dimensions(SCD)



SCD and Its Types

- Slowly Changing Dimensions (SCD) manage changes in dimension data over time in a data warehouse.
- SCD defines the strategy to decide whether to overwrite old data or preserve historical data.

SCD Types:

Type 0 – Retain Original: No changes are tracked. The initial data remains unchanged.

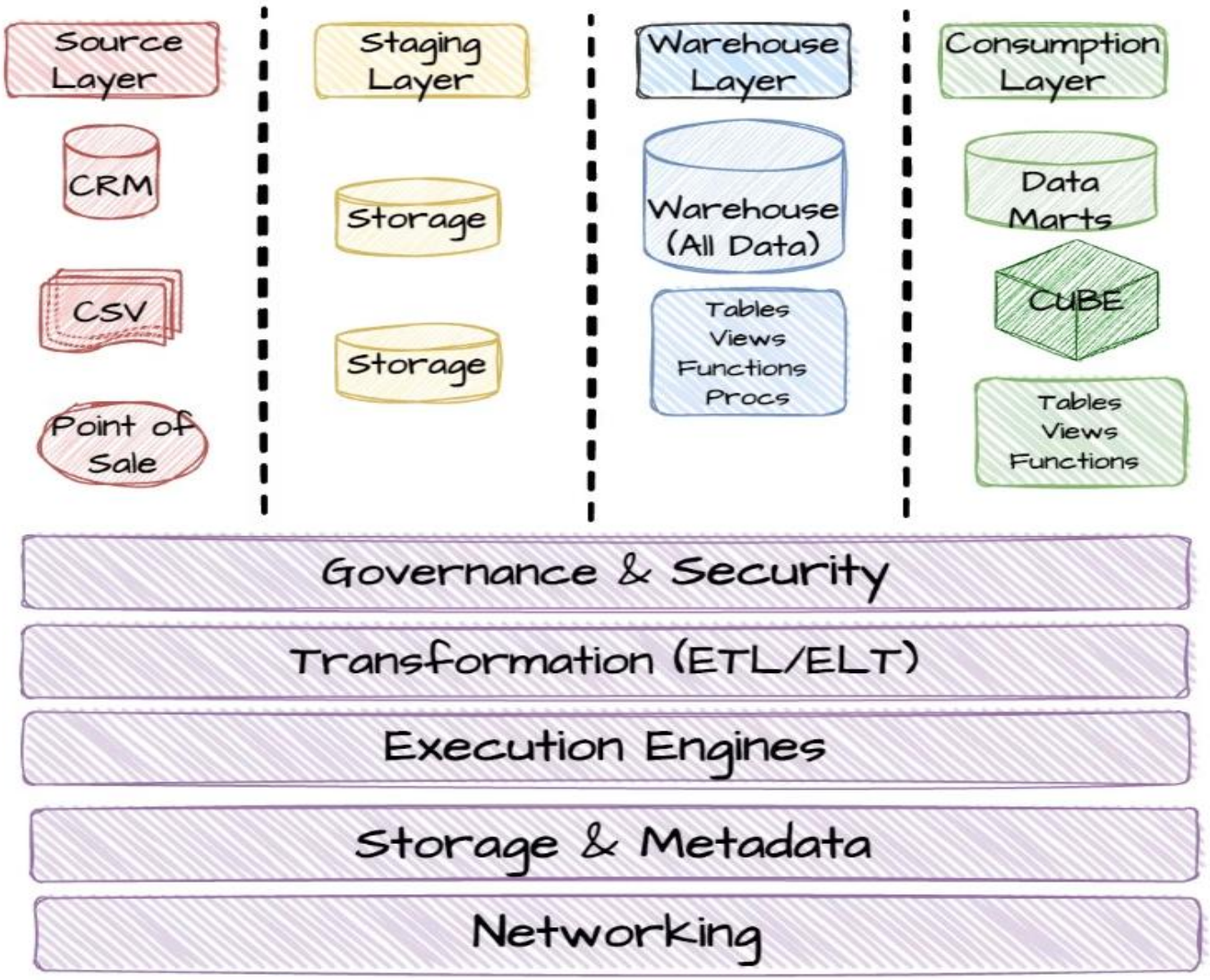
Type 1 – Overwrite: Updates the dimension with the new data, overwriting the old information. This approach is straightforward but loses historical data.

Type 2 – Add New Row: Adds a new record with a new version of the data. This type allows for a full historical trail of changes.

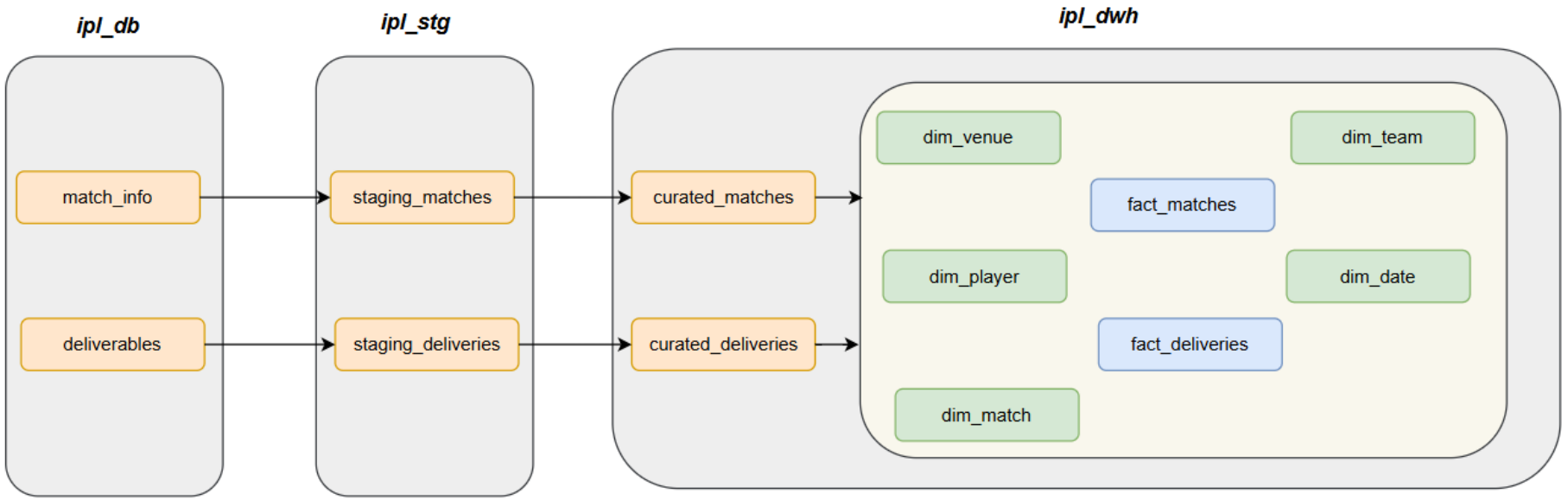
Type 3 – Add New Attribute: Adds new columns to the table to store the previous values. This method is suitable for tracking limited changes, typically where only the previous value is needed.

Type 4 – History Table: Maintains a separate historical table to keep track of changes. This approach is ideal for preserving a detailed history without cluttering the main dimension table.

3-Tier Architecture



Project - IPL Data Analysis



Thank You