Data Modeling Techniques — Conceptual, Logical, Physical, Star & Snowflake Models

Data modeling defines how data is structured, stored, and related across a system. It helps transform business requirements into technical designs. The modeling process generally passes through three key stages — Conceptual, Logical, and Physical — and extends into specialized warehouse models such as Star and Snowflake schemas.

■ Conceptual Data Model

A **Conceptual Data Model** defines the *high-level business view* of the data. It focuses on entities, their relationships, and key business rules, without considering database technologies or column types.

Example Entities and Relationships:

- Entity: **Customer** – attributes: Name, Email
- Entity: **Order** – attributes: OrderDate, Amount
- Relationship: **Customer places Order** (1-to-many)

Use: Used by business analysts and architects to capture business requirements.

■ Logical Data Model

A **Logical Data Model** expands the conceptual model with more technical detail, such as primary keys, foreign keys, and attribute data types (without vendor specifics). It ensures that data entities are normalized and relationships are clearly defined.

Example:

```
Entity: Customer (Customer_ID PK, Name, Email)
Entity: Order (Order_ID PK, Customer_ID FK, OrderDate, Amount)
Relationship: One Customer can have many Orders
```

Use: Used by data architects to design normalized schemas for system integration.

■ Physical Data Model

A **Physical Data Model** represents the actual implementation in a database. It includes exact column types, constraints, indexes, and partitioning strategies optimized for performance on a specific DBMS.

Example SQL:

```
CREATE TABLE Customer (
Customer_ID INT PRIMARY KEY,
Name VARCHAR(100),
Email VARCHAR(100)
);

CREATE TABLE Orders (
Order_ID INT PRIMARY KEY,
Customer_ID INT,
OrderDate DATE,
Amount DECIMAL(10,2),
```

```
FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);
```

Use: Used by DBAs and data engineers for physical deployment and optimization.

■ Star Schema

A Star Schema is a denormalized model where a central fact table connects directly to multiple dimension tables. It is optimized for fast query performance and simple analytics.

Structure:

```
DimCustomer --- FactSales --- DimProduct
        |
    DimDate    DimStore
```

Example SQL:

```
CREATE TABLE FactSales (
Sale_ID BIGINT PRIMARY KEY,
Date_Key INT,
Customer_Key INT,
Product_Key INT,
Store_Key INT,
Quantity_Sold INT,
Sales_Amount DECIMAL(10,2)
);

CREATE TABLE DimCustomer (
Customer_Key INT PRIMARY KEY,
Customer_Name VARCHAR(100),
Gender CHAR(1),
City VARCHAR(50),
State VARCHAR(50)
);
```

❄■ Snowflake Schema

A Snowflake Schema is a normalized version of the star schema. It removes redundancy by splitting dimensions into sub-dimensions (e.g., Category or Region). This increases joins but improves integrity.

Structure:

```
DimCustomer --- FactSales --- DimProduct --- DimCategory
        |
    DimStore --- DimRegion
```

Example SQL:

```
CREATE TABLE DimProduct (
Product_Key INT PRIMARY KEY,
Product_Name VARCHAR(100),
Category_Key INT,
Price DECIMAL(10,2)
);

CREATE TABLE DimCategory (
Category_Key INT PRIMARY KEY,
```

```
Category_Name VARCHAR(50),
Sub_Category VARCHAR(50)
);
```

■ Comparison Summary

Feature	Star Schema	Snowflake Schema
Structure	Denormalized	Normalized
Query Performance	Fast (fewer joins)	Slightly slower (more joins)
Data Redundancy	High	Low
Ease of Understanding	Simpler	More complex
Use Case	BI and OLAP tools	Enterprise-scale warehouses