Comparison between Logical and Physical Data Models

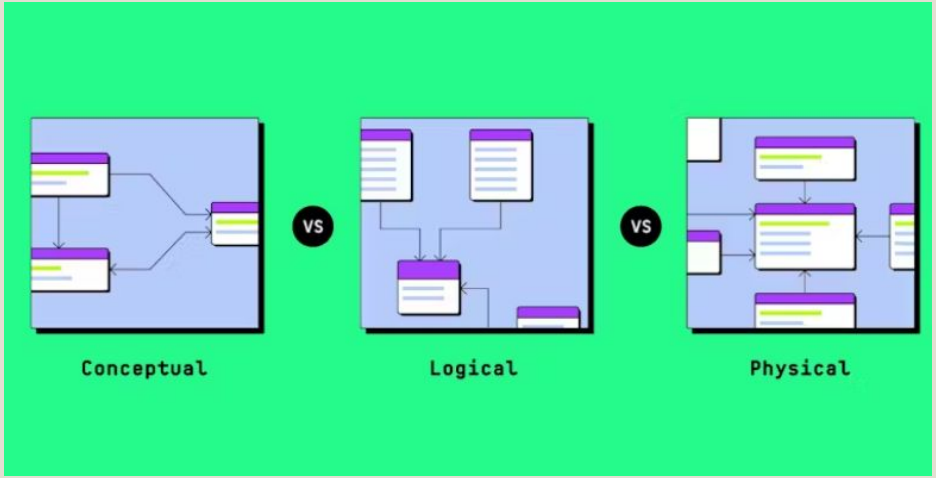

TABLE OF CONTENTS

Introduction

Overview of the Data

- Logical Data Model: Represents the structure of the data, defining relationships between entities without concern for how the data is physically stored.
- Physical Data Model: Focuses on the physical storage and implementation of the logical model.

Goals of the Presentation

- To explain the difference between logical and physical data models.
- To show how they work together to structure and store data efficiently.
- To highlight the role of these models in the design of databases.

Logical Data Model (LDM)

Concept of the Logical Data Model

- Entity-Relationship Diagram (ERD): Visualizes the entities and their relationships.
- Tables/Entities: Defines the main entities (e.g., Listings, Hosts, Reviews, Bookings).
- Attributes: Specifies the fields that belong to each entity (e.g., listing_id, host_id, review_date).
- Primary Keys: Unique identifiers for each entity.
- Foreign Keys: Define relationships between entities.

Benefits of the Logical Data Model

- Serves as a blueprint for database design.
- Helps in understanding the data structure before implementation.
- Facilitates communication among stakeholders (developers, business analysts).

Physical Data Model (PDM)

Concept of the Physical Data Model

- Tables: Specifies actual tables in the database.
- Data Types: Defines the data types for each column (e.g., VARCHAR, INT).
- Indexes: Optimizes query performance.
- Relationships: Defines how the data will be physically stored (e.g., through joins or denormalization).
- Constraints: Specifies constraints such as NOT NULL, UNIQUE, or CHECK.

Benefits of the Physical Data Model

- Optimizes performance for querying and retrieval of data.
- Aligns with hardware and software requirements.
- Helps in implementing the database with performance, scalability, and security in mind.

Key Differences between Logical and Physical Models

Aspect	Logical Data Model	Physical Data Model
Purpose	Focuses on what data is needed and how it is related.	Focuses on how data is stored and implemented.
Representation	Uses ERD (Entity-Relationship Diagram).	Uses actual database structure and schema.
Level of Detail	Higher-level, abstract.	Detailed, specific to database system.
Flexibility	More flexible, as it’s independent of DBMS.	Tied to a specific DBMS (e.g., MySQL, Oracle).
Optimization	Not concerned with performance.	Optimized for query performance and storage.