Data Model

Bachelor of Technology
Computer Science and Engineering (AIML)

Submitted By

Nishant Raj

University Roll No. - 13030823129

Registration No. – 231300121019

Subject- Database Management System

Subject Code - OECAIML601A

Dept.- CSE AIML (B)

YEAR : 3rd

Semester : 6th

Techno Main
EM-4/1, Sector-V, Salt Lake
Kolkata- 700091, West Bengal, India

Abstract

Data models form the cornerstone of Database Management Systems (DBMS), offering structured frameworks to represent, store, and manage data efficiently in a digital ecosystem. This report provides an in-depth exploration of data models, focusing on their evolution, types, and applications within DBMS. Beginning with early hierarchical and network models, the discussion traces the paradigm shift to the relational model, which introduced a tabular structure that remains prevalent today due to its simplicity and scalability. The emergence of object-oriented and NoSQL models reflects the growing complexity of data needs, from multimedia systems to big data analytics. Each model addresses specific requirements: hierarchical for rigid parent-child relationships, network for complex interconnections, relational for structured data, and object-oriented for integrating programming paradigms. NoSQL models, including document, key-value, and graph structures, cater to unstructured data and high scalability demands.

The significance of data models lies in their ability to ensure data integrity, accessibility, and performance optimization. For instance, the relational model's use of keys and normalization minimizes redundancy, while graph models excel in relationship-centric applications like social networks. This report examines the strengths and limitations of these models, highlighting their practical implications in real-world scenarios such as e-commerce, scientific research, and cloud computing.

Introduction

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

In the Database Management Systems (DBMS), data models serve as the architectural backbone, defining how data is structured, stored, and retrieved to meet diverse application needs. As organizations increasingly rely on data-driven decision-making, the importance of robust data management cannot be overstated. Data models bridge the gap between raw data and meaningful information, abstracting real-world entities into manageable formats that facilitate efficient storage and querying. From the earliest file-based systems to today's sophisticated databases, the evolution of data models mirrors advancements in computing technology and the growing complexity of data requirements.

The history of data models begins with rudimentary approaches, where data was stored in flat files with little structure, leading to issues like redundancy and inconsistency. The introduction of hierarchical and network models in the mid-20th century marked a significant leap, offering structured ways to represent relationships among data entities. However, their navigational complexity spurred the development of the relational model by E.F. Codd in 1970, which revolutionized DBMS with its table-based approach and standardized query language, SQL.

Context

The concept of data models in Database Management Systems (DBMS) has undergone a remarkable transformation, reflecting the changing landscape of data management. In the pre-database era, data was stored in flat files—simple, unstructured collections prone to duplication and inefficiency. The advent of DBMS introduced formal data models to address these shortcomings, beginning with the hierarchical model in the 1960s. Used in systems like IBM's IMS, this model organizes data in a tree-like structure, with a single parent for each child record. For instance, in a university database, departments might be parents to courses. While effective for one-to-many relationships, it struggled with many-to-many scenarios, requiring redundant data or complex workarounds.

The network model, standardized by the CODASYL committee, improved upon this by allowing multiple parent-child relationships, forming a graph-like structure. It suited applications like manufacturing systems, where parts and suppliers shared intricate links. However, its reliance on predefined paths made querying cumbersome, requiring programmers to navigate the structure manually. These early models laid the groundwork for data organization but were overshadowed by the relational model's introduction in 1970 by E.F. Codd. Published in his seminal paper, the relational model uses tables (relations) connected via keys, such as primary and foreign keys, to represent data. Its simplicity and mathematical foundation—rooted in set theory—enabled the development of SQL, a declarative language that abstracts navigation complexity. In an e-commerce system, for example, customer and order tables linked by a customer ID allow efficient retrieval without redundant storage. Normalization, a key principle, reduces anomalies by organizing data into related tables, though excessive normalization can impact performance.

The relational model's dominance stems from its data independence, separating logical design from physical storage, and its widespread adoption in systems like MySQL, Oracle, and PostgreSQL. It excels in structured data environments, such as financial systems, where consistency and integrity are paramount. However, as applications diversified, limitations emerged. Multimedia and CAD systems demanded richer representations, leading to the object-oriented data model in the 1980s. This model encapsulates data as objects with attributes and methods, mirroring object-oriented programming principles. A video streaming platform might store video objects with metadata (e.g., title, duration) and behaviors (e.g., play, pause), enhancing integration with application code. Despite its power, its complexity and lack of a standard query language hinder broad adoption compared to relational systems.

Types of Data Models

1. Conceptual Data Model
2. Representational Data Model
3. Physical Data Model

1. Conceptual Data Model

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database.

Components of ER Model:

1. Entity: An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.
2. Attributes: An attribute can be defined as the description of the entity.
3. Relationship: Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.

2. Representational Data Model

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database.

3. Physical Data Model

The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other.

Advantages of Data Models

1. Data Models help us in representing data accurately.
2. It helps us in finding the missing data and also in minimizing Data Redundancy.
3. Data Model provides data security in a better way.
4. The data model should be detailed enough to be used for building the physical database.

Disadvantages of Data Models

1. In the case of a vast database, sometimes it becomes difficult to understand the data model.
2. You must have the proper knowledge of SQL to use physical models.
3. Even smaller change made in structure require modification in the entire application.
4. There is no set data manipulation language in DBMS.

Conclusion

Data models are the linchpin of Database Management Systems, evolving from hierarchical and network structures to relational, object-oriented, and NoSQL paradigms. Each model addresses specific needs: relational for structured consistency, object-oriented for complex entities, and NoSQL for scalability and flexibility. This report highlights their historical development and practical significance, from enabling transactional integrity in banking to powering relationship analysis in social networks. The relational model's simplicity and standardization have ensured its enduring dominance, while NoSQL's adaptability meets modern big data demands. Challenges like performance trade-offs and legacy integration persist, yet data models remain indispensable for data integrity, accessibility, and efficiency. As technology advances, hybrid and AI-driven models may emerge, further enhancing DBMS capabilities. By providing structured frameworks for an increasingly data-centric world, data models underscore their critical role in shaping the future of computing, balancing tradition with innovation.

References

[1] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377-387, Jun. 1970.

[2] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston, MA, USA: Addison-Wesley, 2003.

[3] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2015.

[4] P. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley, 2012.

[5] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, 2nd ed. O'Reilly Media, 2015.