Object and Relational Models

by Sophia

WHAT'S COVERED

This lesson explores the use of object-oriented data models created in the mid-1980s through modern approaches to big data, in three parts. Specifically, this lesson will cover:

1. Object-Oriented Models
2. Extended Relational Models
3. Modern Big Data Challenges

1. Object-Oriented Models

As data became more complex, more frequent, and larger, the network and hierarchical database models simply could not grow to support business and data needs. Starting in the mid-1980s, the object-oriented and object-objected relational data models came into use. Unlike the entity-relationship model of network and hierarchical models, the **object-oriented data model (OODM)** stores both the data and the relationships in a single structure within an object. The OODM is the basis of the object-oriented database management system (OODBMS).

An OODM database supports the principles of object-oriented programming, such as **inheritance**, **encapsulation**, and **polymorphism**. OODM uses inheritance to organize and structure data models. For example, objects inherit attributes and methods from classes. The concept of encapsulation is used in object-oriented database modeling (OODM) to group data and operations that operate on it into a single entity. The encapsulation process hides an object's inner workings and details, exposing only the interfaces and methods necessary to access and manipulate its data. Data integrity depends on this concept. It prevents direct access and potential inconsistency by restricting how an object's internal state can be modified. Polymorphism enhances OODM's code reusability, modularity, and flexibility. Objects of different types can be accessed through a unified interface, simplifying the implementation of complex systems. "Programming to an interface" is enabled by polymorphism, where objects can be treated generically based on shared behavior rather than specific type. Maintainability and scalability are enhanced by loose coupling between objects.

Objects can be directly mapped to actual entities in the real world, allowing them to represent complex data structures more naturally. OO databases offer features such as object identity, complex data types, and

navigation access to relationships. The object-oriented design of these applications makes them particularly appropriate for applications with complex data models and interconnected data networks. Object-oriented databases share some features with document-based NoSQL ones. For example, both define relationships within an object's or document's **metadata**.

Objects contain facts, but they also contain metadata about themselves, which can include the relationships between the facts. This gives the facts within the objects greater meaning.

The OODM closely models how we process data in the real world. In many ways, the object in an OODM is equivalent to a fact, the data around that fact, and how that fact interacts with other facts. The attributes describe the properties of the object that contains them (similar to the columns within a table). Objects that have similar characteristics are grouped into **classes**, where they have a shared structure of attributes as well as shared behaviors or **methods**. These methods define an available real-world action related to the object. In many ways, they are similar to procedures or functions in other programming languages.

IN CONTEXT

An e-commerce system that stores information about products, customers, and orders is an example of an object-oriented database. An object containing attributes such as name, price, description, and quantity could be used to represent each product in such a system. A customer's data could be stored as objects with attributes such as name, address, email, and phone number. A purchase order could be represented as an object containing information about the purchased products, the customer placing the order, the date on which the order was placed, and the amount paid. An object-oriented database makes it easy to model and navigate relationships between objects, such as the association between customer orders and billing information. Data structures that are interconnected and complex can be represented in a natural and intuitive way using an object-oriented database.

THINK ABOUT IT

Why might the hierarchical data model have become useful again for this type of data? How might older approaches to organizing data become relevant again even in future models?

TERMS TO KNOW

Encapsulation

Groups data and operations that operate on it into a single entity.

Inheritance

OODM uses inheritance to organize and structure data models. For example, objects inherit attributes and methods from classes.

Metadata

Refers to descriptive and structural information providing context and data details. It describes various data characteristics, such as its format, source, quality, content, location, and relationships with other data.

Object-Oriented Data Model (OODM)

An object-oriented database model that organizes and represents data as objects, including both data and behavior.

Polymorphism

Enhances OODM's code reusability, modularity, and flexibility.

Object

An organizing unit in object-oriented databases that includes a fact, information about the fact, and the relationships between facts.

Class

A grouping of similar objects with shared attributes and behaviors.

Method

An action that can be performed on an object to which it is assigned.

2. Extended Relational Models

Starting in the mid-1990s, new models were developed to handle complex data representations. These new models borrowed features from both object-oriented and relational models and are considered **object/relational (O/R) models**. For example, the **extended relational data model (ERDM)** emerged as a way to add the object-oriented model features in a more straightforward relational database structure.

Extended relational database models enhance traditional relational models by adding additional features to extend their capabilities. Advanced ERDM features include array and multimedia support, encapsulation, inheritance, and additional integrity constraints based on assertions and triggers. The model supports complex scenarios and relationships in real-world scenarios, allowing for more flexible and expressive data modeling. An ERDM bridges the gap between relational databases and object-oriented databases, combining their benefits. The model provides support for sophisticated data manipulation and querying operations as well as supporting more diverse and dynamic data structures. Extending traditional relational databases allows for a richer and more comprehensive platform to manage and access complex data. Both O/R and NoSQL databases can use Extensible Markup Language (XML) to store and exchange data. O/R databases being able to add support for XML-based documents has made the use of these files more efficient.

Extended Relational Database Model (ERDM)

Extensions of traditional relational database models that add enhanced functionality. A relational model based on ERDM introduces object-oriented concepts, such as inheritance, encapsulation, and methods.

Object/Relational (O/R) Database Model

A type of database model that bridges the gap between object-oriented and relational databases by enabling objects to be directly stored and retrieved from the database.

3. Modern Big Data Challenges

The current generation of emerging data models from the early 2000s to the early 2020s focuses on NoSQL and big data. As you learned earlier in this unit, these include the key-value store, wide-column store, document-oriented, and graph stores.

In today's world, there are mountains of stored data, and it’s not always possible to fit some of the unstructured or social media data into the conventional structure of rows and columns. Many of these modern databases address the needs and flow of the data and are specific to solving a single problem. In that way, they are unique, and there isn't a single solution to meet all of the data storage needs of all organizations. In modern NoSQL databases, **JavaScript Object Notation (JSON)** has emerged as a replacement for XML as the primary means of storing structured and unstructured data.

In database design, JSON (JavaScript Object Notation) is widely used to represent structured data flexibly and without the use of schema. Database designers often use JSON to store complex and dynamic data, allowing nested and hierarchical structures that cannot be easily represented in traditional relational tables. JSON data can easily be stored as text-based values within database fields, making it easy to store and retrieve various data structures. JSON is a versatile and interoperable format for exchanging data between systems and platforms, which allows developers to accommodate evolving data requirements and support rapid application development. Despite JSON's flexibility, the data's quality, consistency, and indexing must be carefully managed to ensure efficient querying and data integrity.

Many modern databases are cloud-hosted. This enables them to take advantage of cloud-based features such as scaling automatically when the database is under load or when there is an increase in the speed of data coming into the database.

Federal, state and industry regulations also have a big impact on how databases secure and protect data. The entire organization must be involved in data governance to make sure that data is protected, stored, shared, and captured in such a way as to not break any laws or regulations.

The future of databases is going to be one of specialization around the data, and how to holistically approach design, implementation, management, oversight, and insights.

TERM TO KNOW

JavaScript Object Notation (JSON)

A lightweight, popular data interchange format commonly used in databases. Using it, one can represent data using key-value pairs or nested structures in a readable, easy-to-understand format.

SUMMARY

In this lesson, you learned that the object-relational and **object-oriented databases** reflected the shift to object-oriented computer programming languages in the 1980s. More recently, the need to manage, organize, and share large amounts of unstructured data has led to the development of **extended**

relational models like object/relational (O/R) and NoSQL. XML was initially used as a data language, but JSON is now the preferred format to address **modern big data challenges**.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

Class
A grouping of similar objects with shared attributes and behaviors.

Encapsulation
Groups data and operations that operate on it into a single entity.

Extended Relational Database Model (ERDM)
Extensions of traditional relational database models that add enhanced functionality. A relational model based on ERDM introduces object-oriented concepts, such as inheritance, encapsulation, and methods.

Inheritance
OODM uses inheritance to organize and structure data models. For example, objects inherit attributes and methods from classes.

JavaScript Object Notation (JSON)
A lightweight, popular data interchange format commonly used in databases. Using it, one can represent data using key-value pairs or nested structures in a readable, easy-to-understand format.

Metadata
Refers to descriptive and structural information providing context and data details. It describes various data characteristics, such as its format, source, quality, content, location, and relationships with other data.

Method
An action that can be performed on an object to which it is assigned.

Object
An organizing unit in object-oriented databases that includes a fact, information about the fact, and the relationships between facts.

Object-Oriented Database Model (OODM)
An object-oriented database model that organizes and represents data as objects, including both data and behavior.

Object/Relational (O/R) Database Model
A type of database model that bridges the gap between object-oriented and relational databases by enabling objects to be directly stored and retrieved from the database.

Polymorphism

Enhances OODM's code reusability, modularity, and flexibility.