Introduction to advance data models such as object relational model, object-oriented model

Advanced data models like the Object-Relational Model and Object-Oriented Model enhance traditional database systems by supporting complex data types, relationships, and programming paradigms.

1. Object-Oriented Data Model (OODM)

◆ Overview

The Object-Oriented Data Model integrates database capabilities with object-oriented programming principles. It treats data as objects, similar to how they are handled in object-oriented languages like Java or C++.

◆ Key Concepts

- **Objects**: The basic unit of data, encapsulating both state (attributes) and behavior (methods).
- **Classes**: Blueprints for objects. A class defines the structure and behavior of its objects.
- **Inheritance**: Classes can inherit properties and methods from other classes, promoting reuse.
- **Encapsulation**: Data and methods are bundled together, hiding internal details.
- **Polymorphism**: The ability to redefine methods in derived classes.

◆ Advantages

- Natural mapping to real-world entities.
- Reusability through inheritance.
- Better support for complex data types (e.g., multimedia, CAD).
- Seamless integration with object-oriented programming.

◆ Limitations

- Less mature query languages compared to SQL.
- More complex to implement and manage.

- Limited support in commercial DBMSs.

◆ Use Cases

- Engineering design systems (CAD/CAM)
- Multimedia databases
- Real-time systems
- Scientific research databases

2. Object-Relational Data Model (ORDM)

◆ Overview

The Object-Relational Data Model extends the traditional relational model by incorporating object-oriented features. It allows complex data types and relationships while retaining the tabular structure of relational databases.

◆ Key Concepts

- **User-Defined Types (UDTs)**: Custom data types that can include attributes and methods.
- **Inheritance**: Tables can inherit structure and behavior from other tables.
- **Complex Objects**: Support for nested tables, arrays, and composite types.
- **Extended SQL**: SQL is enhanced to support object features (e.g., CREATE TYPE, METHOD, INHERITS).

◆ Advantages

- Combines the robustness of relational databases with the flexibility of object-oriented models.
- Easier for organizations to adopt incrementally.
- Supports complex applications without abandoning SQL.

◆ Limitations

- More complex schema design.

- Performance overhead for managing object features.
- Not all RDBMSs support full ORDM capabilities.

◆ Use Cases

- Enterprise applications (e.g., CRM, ERP)
- Financial systems
- Geographic Information Systems (GIS)
- Hybrid systems requiring both structured and complex data

Comparison Table

Feature	**Object-Oriented Model**	**Object-Relational Model**
Data Structure	Objects and Classes	Tables with Object Extensions
Query Language	OQL (Object Query Language)	Extended SQL
Inheritance	Fully Supported	Partially Supported
Encapsulation	Yes	Limited
Integration	Tight with OO languages	Moderate
Schema Flexibility	High	Moderate
Commercial Adoption	Limited	Widely Supported (e.g., PostgreSQL, Oracle)