Name: Sadiq Ahmad

Roll No: 1540

Project Title: Object Oriented Data Model

Course: Data Base Management

Department: BCS 5th semester evening

The object-oriented data model (OODM) is a data management paradigm that applies principles from object-oriented programming (OOP) to structure and organize data. It represents data as objects, which encapsulate both data

(attributes) and behavior (methods), enabling a more intuitive representation of real-world entities compared to traditional relational models.

Key Concepts:

1. **Objects and Classes**:
   - **Objects**: Instances of classes that store data (attributes) and include methods to manipulate that data.
   - **Classes**: Blueprints defining the structure (attributes) and operations (methods) of objects.
2. **Inheritance**:
   - Classes can inherit attributes and methods from parent classes (superclasses), promoting code reuse and hierarchical organization.
3. **Encapsulation**:
   - Data and methods are bundled within objects, restricting direct access to internal data and enhancing security/modularity.
4. **Polymorphism**:
   - Methods can behave differently based on the object's class, enabling flexible interactions (e.g., a draw() method works for both Circle and Square objects).
5. **Object Identity**:
   - Each object has a unique identifier (OID), independent of its attribute values, ensuring distinctness even if attributes match.

How It Works:

- Data is stored as objects rather than in tables.
- Relationships between objects are managed via references (e.g., pointers), supporting complex associations like aggregation and composition.
- Persistent objects remain stored beyond program execution, managed by an **object-oriented database management system (OODBMS)**.

Contrast with Relational Models:

Feature	Object-Oriented Model	Relational Model
Structure	Objects with attributes/methods	Tables with rows and columns
Relationships	Direct references (OIDs)	Foreign keys and joins
Query Language	OQL (Object Query Language)	SQL
Complex Data	Handles multimedia, nested data	Limited to tabular, normalized forms
Impedance Mismatch	Minimal (aligns with OOP languages)	Common (requires ORM tools)

Advantages:

- **Natural Modeling**: Aligns with OOP, making it ideal for applications like CAD, multimedia, or complex systems.
- **Flexibility**: Supports inheritance and polymorphism for extensible designs.
- **Performance**: Avoids costly joins by storing related data within objects.
- **Reduced Mismatch**: Seamless integration with OOP languages (e.g., Java, Python).

Disadvantages:

- **Adoption**: Less mature and widely used than relational databases.
- **Learning Curve**: Requires familiarity with OOP concepts.
- **Query Complexity**: Less intuitive for users accustomed to SQL.
- **Standardization**: Fewer universal standards compared to SQL.

Use Cases:

- Applications requiring complex data structures (e.g., engineering systems, healthcare).
- Real-time systems where performance is critical.
- Projects using OOP languages that benefit from direct object persistence.

Examples of OODBMS:

- **db4o**: Embedded database for Java and .NET.
- **Object DB**: High-performance Java database.
- **Versant**: Enterprise-grade OODBMS.