ADBMS UNIT 1 OBJECT ORIENTED DATABASE

OOP + Relational Model (ie. db concepts) = OODBMS (Object Oriented database management systems)

→ Instead of storing db data in rows & columns (traditional), OODBMS stored data in the form of objects.
→ This makes managing complex information easier.

OBJECT ?

① Object is anything in world which consist of states and behaviour
② For states we use variables for behaviour we use functions..
③ The current state of object is described by one/more attributes (instance varibles)
④ These objects are stored directly in db.

BASIC STRUCTURE OF OBJECT

Attributes → Data fields that represent the state of object
Methods → function that define behaviour of object

Eg. 'Person' object has attributes 'name', 'address', 'dob' and methods 'updateAddress', 'walk & talk'

BASIC PROPERTIES OF OODBMS

① Persistence - Objects are stored in db (ie saved) & can be retrived or manipulated later.
② Inheritance - OODBMS allows object to inherit properties & methods from parent objects, promoting data & code reuse.
③ Encapsulation - Data & methods are bundled together within objects, ensuring that data can only be accessed or modified through the objects' method.
④ Identity - Each object in OODBMS /OODB has unique identity, which distinguishes it from other objects, regardless of objects state.
⑤ Complex Objects - OODB's can store complex objects that might contain other objects as attributes, enabling the modeling of real-world relationships more naturally than relational db.

ADVANTAGES OF OODB's

① Support for Complex Datatypes - OODB can handle complex dt including multimedia, spatial data, & other UDtypes, more naturally than relational db.

② Inheritance & Reusability - Inheritance in OODB allows for reuse of common data structures & behaviours, reducing redundancy & promoting maintainability.

③ Rich Query Capabilities - OODB offers rich query capabilities, often supporting Object Query Language (OQL), that allows complex query on objects & their relationships.

④ Polymorphism - Different objects can be used interchangeably

⑤ Enhanced flexibility - modify.

⑥ Support for Distributed Systems

Eg. Class - Employee (Inherits from Person)
Attributes - empId, dept
Methods - promote()

Attribute ??
⇒ characteristic / property of an object that stores data / info.
class - car includes attribute - color, model, year
⇒ func^c is block of code

ADT (ABSTRACT DATA TYPES)

- It is data type where the type of data & the operations that can be performed on the data are defined by the programmer, rather than being restricted to predefined data types like int, string.
- Role ? allow for creating complex data structures that reflects real world entities more accurately.

Characteristics
① Encapsulation ② Data abstraction ③ Custom data Structures.

Types
① simple ② Composite ③ Parameterised ④ User Define

Advantages
① Better data modeling ② Encapsulation & Integrity ③ Reusability
④ Enhanced Query Capabilities

ENCAPSULATION

The internal details of an object are hidden from outside world, and interaction with the object is only possible through a well-defined interface.

Role - Ensures that data within an object is protected from unintended ~~interface~~ interference and misuse, promoting data integrity & security

CLASS HIERARCHY (INHERITANCE)

It is the organization of classes in a parent-child relationship. In this structure, a child class inherits attributes & methods from its parent class, allowing for code reuse & logical organization.

Advantages :

① Code Reusability ② Logical Organization
classes are organized in hierarchy that reflects real world relationships ③ Extensibility.
new subclasses can be added with minimal changes to existing code.

POLYMORPHISM

- Objects of different classes are treated as objects of a common super class. It allows same method to behave differently based on the object that it is acting upon.
- Role? enables flexibility in db

Eg. To design db for university, you need both "student" & "faculty" They have some same attributes like name, address & some ~~comm~~ unique student-id, major, faculty-id, department

Traditional RDB approach	Object ODB approach
2 sperate tables	Base Class
student	Derived Class
faculty	Directly store instances

Eg. A car, Truck & bike all have breaks, but the mechanism is different

RELATIONAL MODEL CONCEPTS

- first proposed by E.F. Codd hence he is known as father of relational database.
- RDB was an attempt to simplify database structure by making use of tables and columns.
- Tables → relations
  Columns → attributes, fields
  Rows → Records, tuples

RELATIONAL MODEL CONCEPTS

① Relation - In RDB, a relation is essentially a table. It consist of rows & columns. Each row repreents unique instance of data, and each column represent particular attribute of the data.

② Tuple - It is single row within a table. It represent a single record in relation, containing a specific set of values.

③ Attribute - It is a column in a table. Each attribute has a name and a domain.

④ Domain - Domain of attribute is the set of all possible values that attribute can have. for eg attribute age can have domain '0 to 120'

⑤ Primary key

⑥ Foreign key

⑦ Relation schema - Schema describes the structure of a relation, including attributes, their domains & any constraints.

⑧ RDB schema - This is a collection of relation schemas. It defines the structure of entire database.

RELATIONAL MODEL CONSTRAINTS

① Domain constraints :-
These ensures that the values of an attribute must lie within a specified domain.
For eg. 'dob' attribute must lie/contain only valid dates.

DOMAIN CONSTRAINT = NOT NULL + CHECK

② Entity Integrity Constraint

This ensures that the primary key of a relation cannot have null values because pk is used to identify a tuple uniquely.

ENTITY INTEGRITY = PRIMARY KEY + UNIQUE

allows 1 null value which is unique by itself.

③ Referential Integrity Constraint.

This ensures that FK value in one relation must either be null or match a PK value in another relation.
This maintains links between tables.

FOREIGN KEY

④ Default constraint

Any violations by FK ?? → YES 3 violations.

① NO ACTION / RESTRICT

db engine will not allow user to delete / add the row if it is not present in other table

② CASCADE (Delete / update)

deletes / updates values from referencing table
ie does not maintains links

③ SET NULL & SET DEFAULT

all values in FK are set NULL, if corresponding row in parent table is deleted.

all values that make up FK are set to their default values of corresponding row in parent table is deleted.

DEALING WITH CONSTRAINT VIOLATIONS

① PK VIOLATION → Attempt to insert duplicate value or null value

② FK (done prior)

③ UNIQUE CONSTRAINT VIOLATION → attempt to insert / update a row with value that is not unique in column defined as unique.

④ CHECK CONSTRAINT VIOLATION → inserting / updating a row with value that does not meet specified condition

⑤ ~~NOT~~ NULL CONSTRAINT VIOLATION → inserting/updating with null

ANOMALIES

Anomalies are problems that can arise when inserting, updating or deleting data in a db that has not been properly normalized.

① Insertion Anomalies

- Occurs when you cannot add data into db due to absence of other data
- Eg. If a table stores both student & course information, you might not be able to add a new course if no students have enrolled in it yet.

② Update Anomalies

- Occurs when changes to data require multiple updates in different places, leading to inconsistencies.
- Eg. If a students address is stored in multiple rows, updating their address requirs changes in all these rows, increasing risk of errors.

③ Deletion Anomalies

- Occurs when deleting data inadvertently removes additional important data.
- Eg. If deleting a course from a table also removes student information because they are stored together, this is deleting anomaly.

RELATIONAL DB SCHEMAS

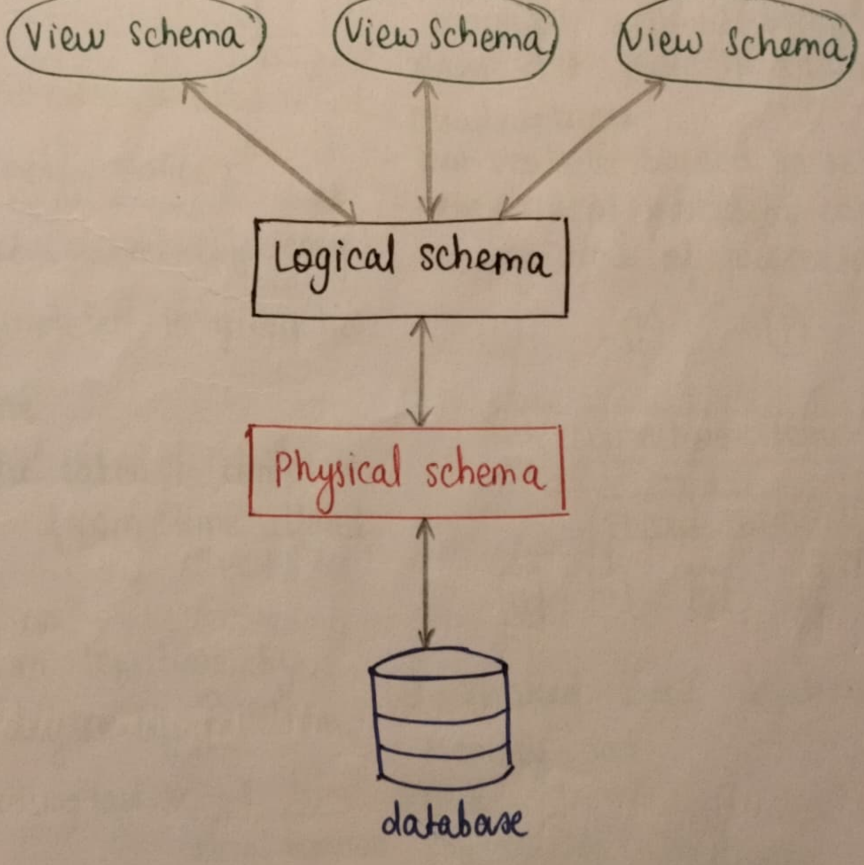
- Description / design of db is called db schema.
- It does not expect to change frequently.
- It defines how the data is arranged & how the relations among them are associated.
- It also represents all the constraints applied on data

Types :

① Physical / Internal
defines how data is stored physically in storage sys. in the form of files & indices.
This is actual code / syntax needed to create db.

② Logical
It defines all logical constraints that need to be applied to stored data.
It describes how the data is stored in the form of tables & how attributes of table are connected

③ External / View
It defines the interaction betn end-user & db
User is able to interact with db with the help of interface without knowing much about stored mechanism of data in db.



OID (Object Identifiers)

- OIDs are unique identifiers assigned to every object in db.
- They are used to identify objects such as tables, indexes, and columns, as well as other db components (UDF or types)
- OIDs typically represented as numeric values, although they can also be represented as string or other dt.
- Every object when created will get one OID.
- Assⁿ of OID is automatically handled by DBMS, but some DBMSs allow users to assign OIDs manually.
- Each OID is unique to a specific object & is used to identify that object within the db.
- OID is imp because:-
  1. Helps to ensure that data is stored & retrived accurately
  2. To optimize db performance.