Database - 2K22/IT/72

From Sir's slides and lab tasks.

File Based Systems

A collection of data, it is the traditional way of storing data. Each department stored and managed its own data.

Limitations

- Separation and Isolation of Data
- Duplication and Redundancy of Data
- Data Dependence
- Incompatible File Formats
- Fixed Queries

Reason for Limitations → No access/manipulation control because Data Definition was embedded in the programs instead of being stored separately. This resulted in the creation of DBMS.

Database Management System

A collection of data, Each department can access and manage the same collection. This enables users to define, create, maintain and control access to the database.

- Oracle
- MySQL
- SQLserver
- Postgres
- Sybase
- MS Access
- Informix

Procedures → Instructions/Rules that should be applied to design/use DBMS.

Schema → Description of the data. It is the "blueprint" which the database follows.

Database Application Program → A program that interacts with the database by giving the appropriate SQL request to the DBMS.

Roles in a Database Environment

- Data Admin (DA)
- Database Admin (DBA)
- Database Designers
- Programmers
- End Users

Database

Shared collection of logically related data (includes entities, attributes and relationships).

Data Models

It tells how the system will look after implementation.

- Object-Based Data Models
    - **Entity-Relationship Models**
    - **Semantic Models**
    - **Functional Models**
    - **Object-Oriented Data Model**
- Record-Based Data Models
    - **Hierarchical Data Model** → Uses a tree structure to store data.
    - **Network Data Model** → Uses a graph structure to store data.
    - **Relational Data Model** → Uses tables to store data.
- Physical Data Models

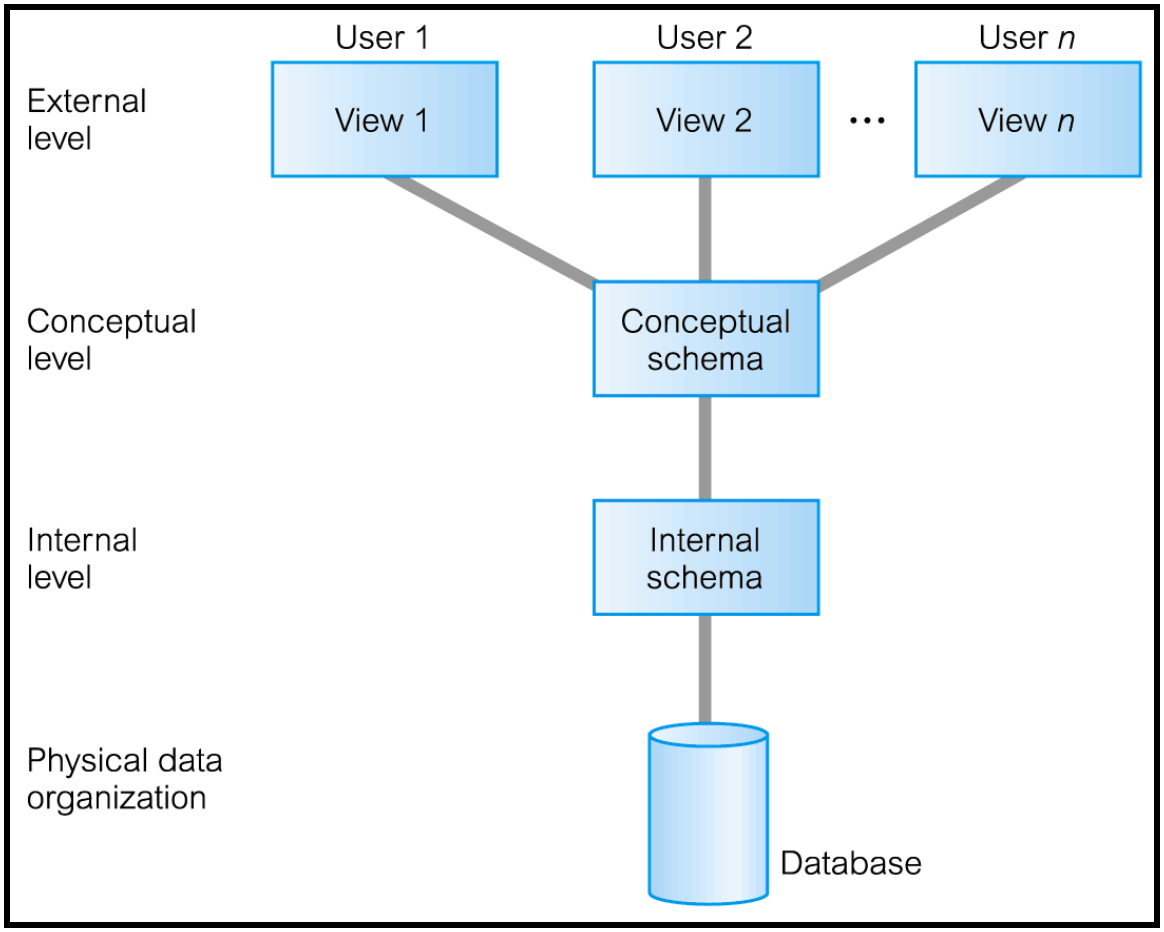
1st Gen → Hierarchical, Network
2nd Gen → Relational
3rd Gen → Object-Relational, Object-Oriented

Components of Database → Hardware, Software, Data, Machine, Human

ANSI-SPARC Three-Level Architecture

1. **External Level** → Shows the data relevant to the user using Views and hides rest.
2. **Conceptual Level** → Tells what data is stored and the relationships between tables.
3. **Internal Level** → Tells how the data is stored inside the storage in form of bits.

- **Physical Data** → Database itself



Multi-User Database Architectures

- **Teleprocessing** → Terminals are connected to a single mainframe.
- **File-Server** → Connected to several workstations across a network.
- **Client-Server**
    - **Two-Tier Client-Server** → Client can directly communicate with the Server.
    - **Three-Tier Client-Server** → Client and Server have a middle Application layer.

Architecture	**Interface**	**Applications**	**DBMS**	**Database**
Teleprocessing	Terminals	Single Mainframe		
File-Server	All Workstations			Files Server
2-Tier Client-Server	Client		Database Server	
3-Tier Client-Server	Client	Application Server	Database Server	

Data Independence

The ability to modify one of the three schema without needing to change others.

Logical Data Independence → Ability to change Conceptual Schema without needing to change External Schema.
Physical Data Independence → Ability to change Physical Schema without needing to change Conceptual Schema.

System Catalog → Metadata to describe data in the database. (names, types, sizes, constraints etc.)

Languages

- **Data Definition Language** (DDL) → Defines structure/schema of the Database.
- **Data Manipulation Language** (DML) → Helps managing/manipulating the data in the Database.
    - **Procedural DML** → Tells how to manipulate data.
    - **Non-Procedural DML** → Tells what data is needed.
- **Data Control Language** (DCL)

Views

A subset of the database. It is a virtual table that can be created using SQL statements.

Purpose of Views

Provides Security.
Shows Relevant data to Users.

Change in View should be reflected by a **Change in Relation**.
Change in Relation should be reflected by **Changes in all Views**.

Relational Model

- **Relation** → **Table**, **File**. It is about the Logical Structure NOT the Physical Structure.
    - **Relational Database** → Collection of relations.
- **Attribute** → **Column**, **Field**. A named column of the relation.
    - **Degree** → Number of attributes in a relation.
- **Tuple** → **Row**, **Record**. A row of the relation.
    - **Cardinality** → Number of tuples in a relation.
- **Domain** → Set of allowed values for one or more attributes.
- **Relation Schema** → Set of table definitions, constraints, and rules.
    - **Relational Database Schema** → Set of Relation Schemas.

Properties

1. **Uniqueness**
    - Relation name must be different from other relations.
    - Attribute name must be different from other attributes.
    - Tuple must be different from other tuples.
2. **Bounds**
    - Cells of relation can contain exactly one value.
    - Attributes of relation can only contain values from the same domain.
3. **Order**
    - Attribute order doesn't matter.
    - Tuple order doesn't matter.

Relational Algebra and Relational Calculus → Formal Languages in the relational model.

Relational Algebra

Closure Property → Expressions can be nested.

Operations

1. **Projection** → Shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table. It is denoted by pi (Π).
2. **Selection** → Selects tuples(rows) that satisfy a given predicate. It is denoted by sigma (σ).

R and S are relations/list of tuples.

3. **Cartesian Product** → Combines each row in R with each row in S. It is also known as a cross product. It is denoted by X.
4. **Union** → Contains all the tuples that are either in R or S or both in R & S. It eliminates the duplicate tuples. It is denoted by ∪.
5. **Set Difference** → Contains all tuples that are in R but not in S. It is denoted by minus (-).

6. **Intersection** → Contains all tuples that are in both R & S. It is denoted by ∩.
7. **Join** → To combine R and S into a single relation.
8. **Division** → Tuples of R that match every combination Tuples of S.

Aggregate Functions

- **SUM** → Returns the Sum of values. Only works on numbers.
- **COUNT** → Returns the number of values.
- **MIN** → Returns the smallest value. Only works on numbers.
- **MAX** → Returns the largest value. Only works on numbers.
- **AVG** → Returns the average of values. Only works on numbers.

Grouping Functions → Using multiple Aggregate Functions.

Relational Calculus

Tells what to get instead of how to get it.

Predicate → A truth-valued function with arguments.
Proposition → An expression made from the function by substituting values.

Tuple Relational Calculus

To find tuples where the predicate is true.

- **Bound/Free Variables** → Variables that fulfill one of the Quantifiers.
    - **Existential Quantifier** (There exists)
    - **Universal Quantifier** (For all)

Domain Relational Calculus

Uses variables that take values from domains **NOT** tuples.

Structured Query Language (SQL)

Keywords(Statements) are not case-sensitive.

Text Fields → Enclosed in single quotations.
Numeric Fields → Not Enclosed.

Relational Keys

- **Super Key** → Set of Attributes/Columns that can uniquely identify Tuples/Rows.
    - **Candidate Keys** → A super key from which removing even a single Atrribute/Column breaks this property.
        - **Primary Key** → A candidate key which can uniquely identify Tuples/Rows alone.
        - **Alternate Keys** → A candidate key not selected to be primary key.
        - **Foreign Key** → A candidate key that matches primary key in another table.

Constraints

They are rules at the table level.

- **Null** → Represents the absence of a value and not the same as Zero or Space.
- **Entity Integrity** → No attribute of a primary key can be null.

Constraints in Oracle SQL	Specifies that
NOT NULL	Column can’t contain a NULL value.
UNIQUE	Column can’t have duplicate values.
PRIMARY KEY	Column is the primary key.
FOREIGN KEY	Column is the foreign key to some other table.
CHECK	Certain condition must be true.

Syntax →

Referential Integrity constraint

Also called Foreign Key constraint. It ensures that there is a valid reference between tables and no inconsistencies.

The referenced table/relation is called the **Master**.
The table/relation that is referencing is called the **Child**.

- **INSERT Constraint** → Value can not be inserted in Child, if it's not found in Master.
- **DELETE Constraint** → Value can not be deleted from Master, if it's found in Child.
    - **ON DELETE CASCADE** → Deleting from Master, deletes the same value in Child.
    - **ON DELETE NULL** → Deleting from Master, nulls the same value in Child.

Comments

- **Single Line**
  By giving -- before text.
- **Multi Line**
  By giving /* before and */ after the text.

Operators

- **Arithmetic Operators** → +, -, *, /, %
- **Comparison Operators** → <, >, =, <=, >=, <>
- **Logical Operators** → AND, OR, NOT, BETWEEN, IN, LIKE, ANY, ALL

Alias

Nickname of a column or table.

- Columns can have alias using the AS statement in the SELECT clause.

```
SELECT Salary AS Wage
FROM Employees
```

- Tables can have alias using the AS statement

```
SELECT Salary AS Wage
FROM Employees
```

- OR by specifying it after table name and space in the FROM clause.

```
SELECT e.Salary
FROM Employees e
```

Concatenation

SELECT firstName + ', ' + lastName AS Name FROM Employees

Like

The LIKE operator is used in WHERE clause to find specific string patterns.

- **%** is used to represent **multiple** wildcard characters.
    - **'A_'** → Starting with A. Single character after it.
    - **'_A'** → Ending with A. Single character before it.
    - **'_A_'** → Contains A. Single character before and after it.
- **_** is used to represent a **single** wildcard character.
    - **'A%'** → Starting with A. Any amount of characters after it.
    - **'%A'** → Ending with A. Any amount of characters before it.
    - **'%A%'** → Contains A. Any amount of characters before/after it.

Statement	Property	Syntax	Function
SELECT	To select columns	SELECT * FROM Employees	Shows all columns of Employees table.
SELECT DISTINCT	To select unique values in the column.	SELECT Salary FROM Employees	Shows Salary column of Employees table.
FROM	To specify the table.	SELECT DISTINCT Salary FROM Employees	Shows Salary column of Employees and only values that are unique.
WHERE	To filter out rows.	SELECT Salary FROM Employees WHERE Salary>1000	Show Salary column of Employees table. Only the rows with 1000+ Salary.
BETWEEN	To search between a range.	SELECT Salary FROM Employees WHERE Salary BETWEEN 100 AND 500	Shows Salary column of Employees table. Only the rows with values between 100 and 500.
LIKE	To search for a pattern (used for strings).	SELECT Name FROM Employees WHERE Name LIKE 'a%'	Shows Name column of Employees table. Only the rows with values starting from a.
IN	To search for specific values.	SELECT Salary FROM Employees WHERE Salary IN (100,200)	Shows Salary column of Employees table. Only rows with the values 100 or 200.
AND	Shows if fulfills both conditions.	SELECT Salary FROM Employees WHERE Salary>100 AND Salary<800	Shows Salary column of Employees table. Only rows with the values greater than 100 and less than 800.
OR	Shows if fulfills either condition.	SELECT Salary FROM Employees WHERE Salary<100 OR Salary>800	Shows Salary column of Employees table. Only rows with either the values greater than 800 or less than 100.
NOT	Shows if the condition if not fulfilled.	SELECT Salary FROM Employees WHERE Salary NOT IN (100,200)	Shows Salary column of Employees table. All rows except ones with values 100 or 200.
IS NULL	To select null values.	SELECT Salary FROM Employees WHERE Salary IS NULL	Shows Salary column of Employees table. All rows that are NULL.
IS NOT NULL	To select not null values.	SELECT Salary FROM Employees WHERE Salary IS NOT NULL	Shows Salary column of Employees table. All rows that are not NULL.

AS	For alias making.	SELECT Salary AS Wage FROM Employees	Shows Salary column of Employees table and shows the name as Wage.
ORDER BY	Shows in ascending order.	SELECT Salary FROM Employees ORDER BY Salary	Shows Salary column of Employees table in ascending order.
DESC	Shows in descending order.	SELECT Salary FROM Employees ORDER BY Salary DESC	Shows Salary column of Employees table in descending order.
DELETE	To delete rows.	DELETE FROM Employees WHERE Salary = 0	Deletes rows of Salary column of Employees table which have value of 0
INSERT INTO	To add new rows. If you want to add values to all columns	INSERT INTO Employees (Name, Age, Salary) VALUES ('Jaish', 20, 10000) INSERT INTO Employees VALUES ('Jaish', 20, 10000)	Inserts 'Jaish' in the Name column, 20 in the Age column and 10000 in the Salary column; as a row. Used if you want to add values to all columns. Order of columns matters.
CREATE TABLE	To create a new table.		
ALTER TABLE	To change an already existing table.		
DROP TABLE	To delete a table.		