UNIT – I

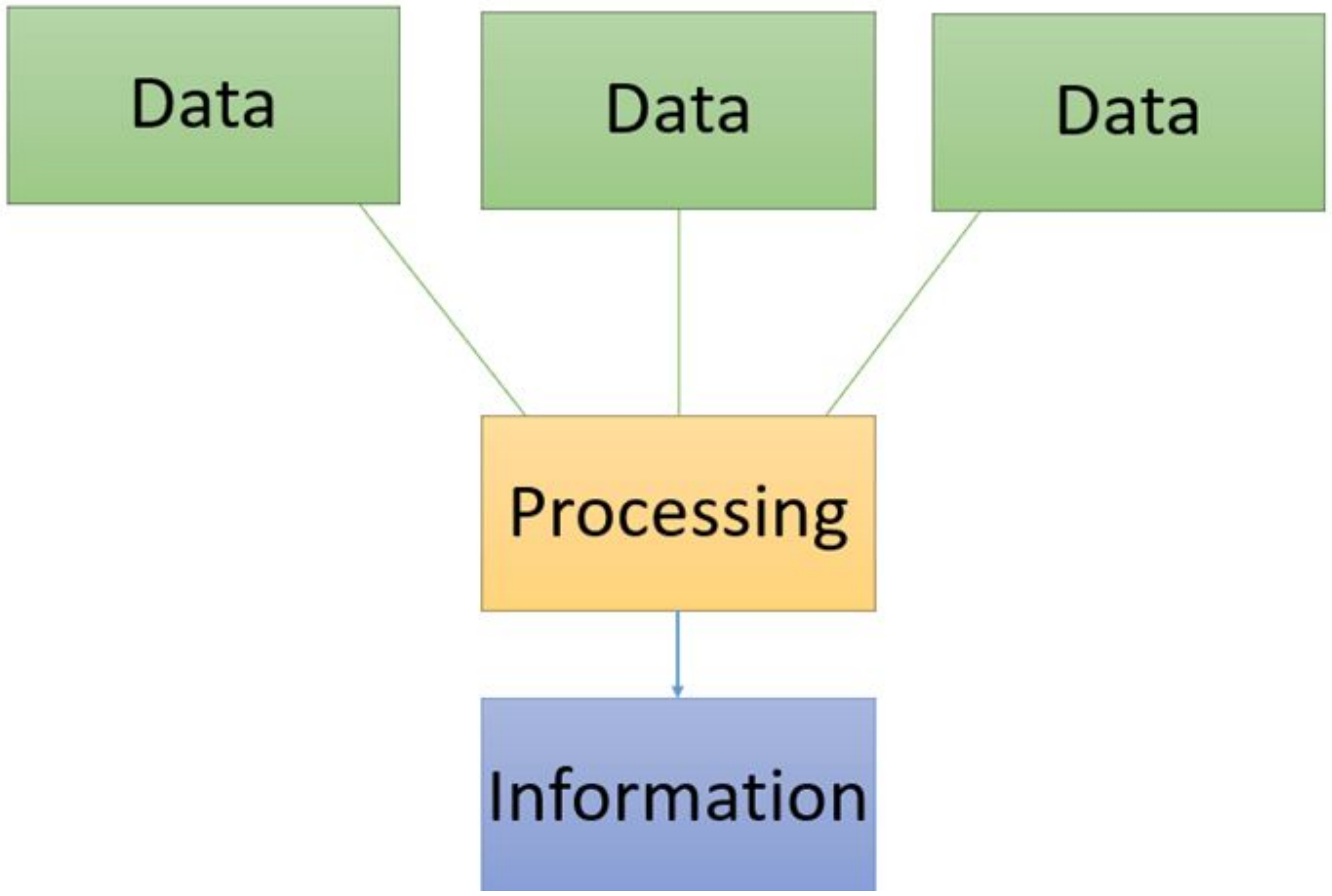
18CSC303J - DATABASE MANAGEMENT SYSTEMS

What is Data?

- Data is a raw and unorganized fact that required to be processed to make it meaningful.
- Data can be simple at the same time unorganized unless it is organized.
- Generally, data comprises facts, observations, perceptions numbers, characters, symbols, image, etc.
- Data is always interpreted, by a human or machine, to derive meaning.
- So, data is meaningless. Data contains numbers, statements, and characters in a raw form.
- Data : It is a RAW FACT (It won't give any meaning)

  Ex: 10, RAM, etc.

Information



What is Information?

- Information is a set of data which is processed in a meaningful way according to the given requirement.
- Information is processed, structured, or presented in a given context to make it meaningful and useful.
- It also involves manipulation of raw data.
- Information assigns meaning and improves the reliability of the data.
- So, when the data is transformed into information, it never has any useless details.

Information : Which gives meaning for Data

Ex: id = 10 , name = ‘RAM’ Distance in miles = 200, etc.

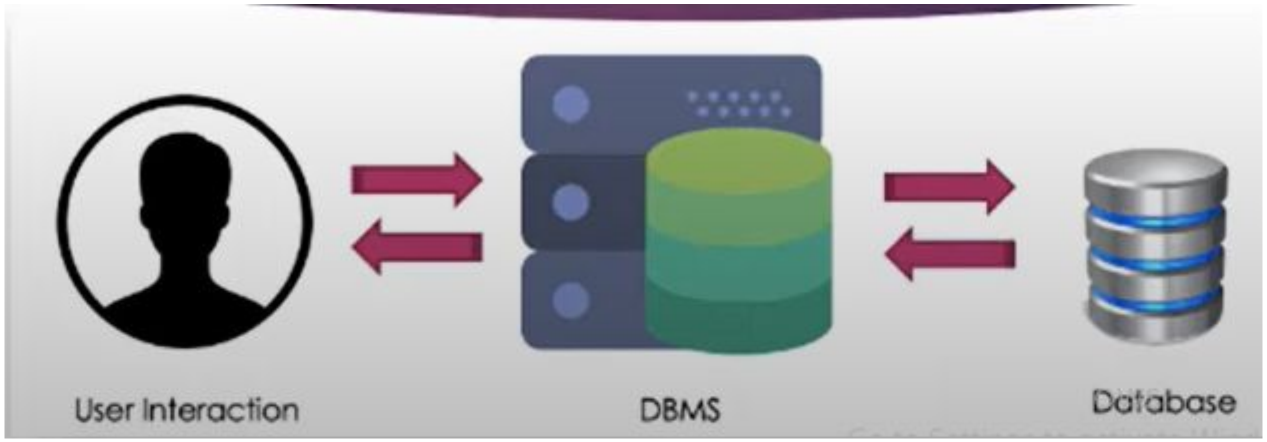
Information system

- A system is a set of interrelated components, with a clearly defined boundary, working together to achieve a common set of objectives.

- Information system, **an integrated set of components for collecting, storing, and processing data and for providing information, knowledge, and digital products**.

DBMS

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise.
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.



DBMS Tools

- MySQL. One of the most useful database management tools is MySQL
- SQL Server Management Studio
- Oracle RDBMS
- Salesforce
- DevOps
- Visual Studio Code
- ESM (Enterprise Service Management (ESM) Tools
- PhpMyAdmin

Basic Concepts

Basic Concepts

Software : Collections of programs. Example: Apps, Software's

Hardware : Collection of mechanical, electrical and electronics parts.

System Software : Run machine hardware & provide platform to the application. Example. OS

Application software : Developed as per user requirements. Example. Word, Browsers & Apps etc.

Program: To instruct machine to perform a particular task.

Data : Collections of facts, figures, numbers etc.

Database : Collections of interrelated data.

Manipulation: Operations perform on database.

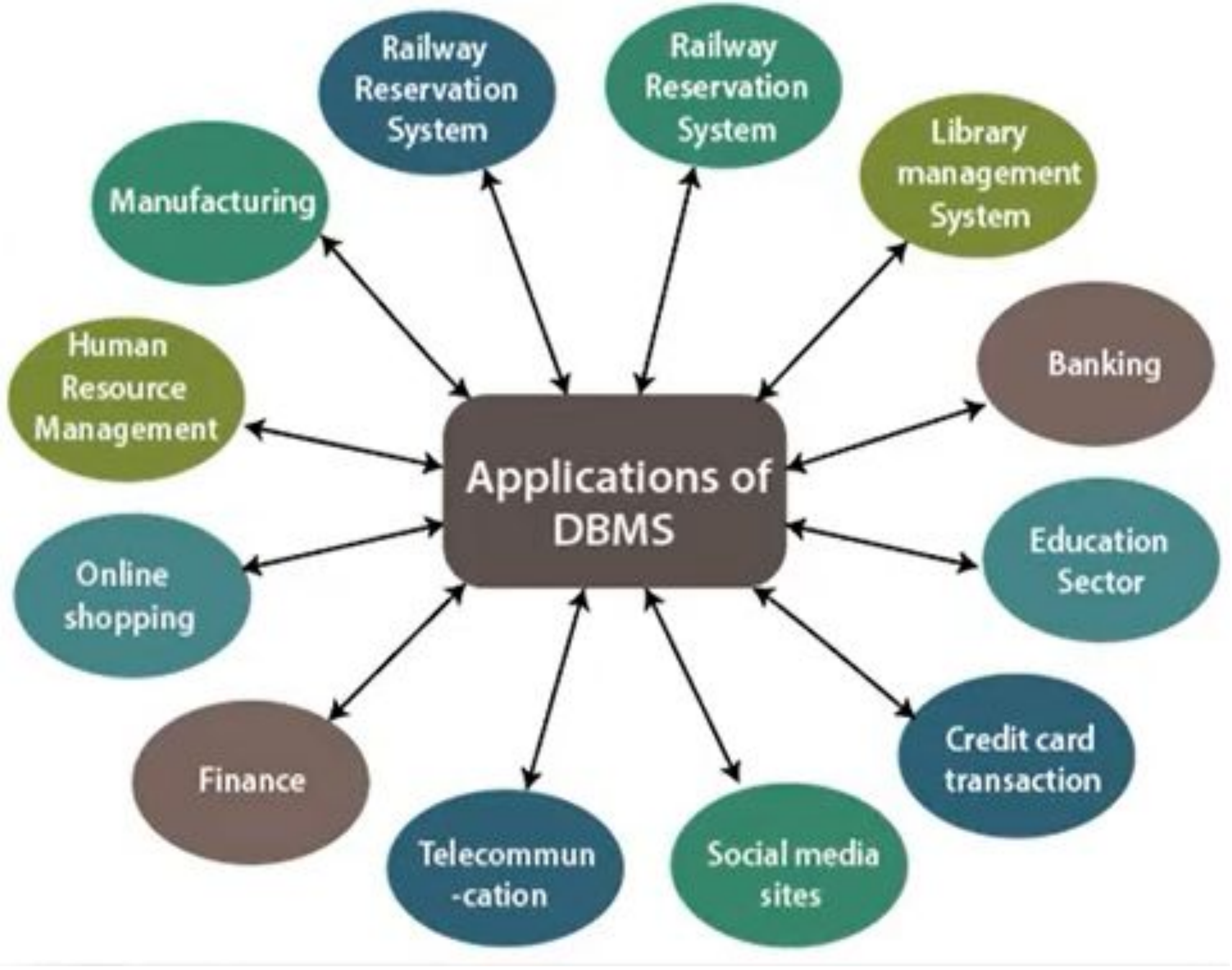
Database-System Applications

Databases are widely used. Here are some representative applications:

Enterprise Information

- **Sales:** For customer, product, and purchase information.
- **Accounting**: For payments, receipts, account balances, assets and other accounting information.
- **Human resources**: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- **Manufacturing:** For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- **Online retailers:** For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Database-System Applications



Database-System Applications

Banking and Finance

Banking: For customer information, accounts, loans, and banking transactions.

Credit card transactions: For purchases on credit cards and generation of monthly statements.

Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.

Database-System Applications

Universities: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).

Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.

Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Why use DBMS

- ✔ To develop software applications In less time.
- ✔ Data independence and efficient use of data.
- ✔ For uniform data administration.
- ✔ For data integrity and security.
- ✔ For concurrent access to data, and data recovery from crashes.
- ✔ To use user-friendly declarative query language

Purpose of DBMS

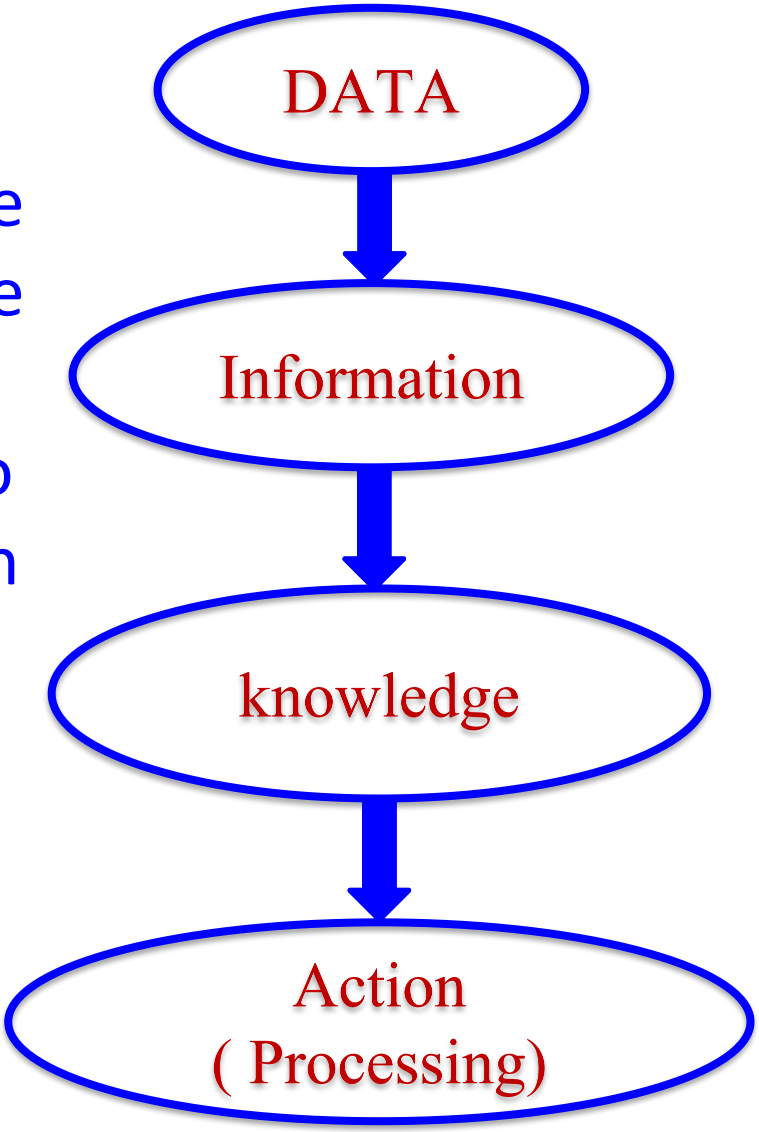
The purpose of DBMS is to transform the following –

- ✔ Data into information.
  - Raw fact is converted into Meaningful information
  - For Example:
    - The data '1000' is converted into INR = '1000'
- ✔ Information into knowledge.
  - Using the information or comparing the information , can easily develop knowledge
  - For Example:
    - INR 75 = 1 USD
    - INR100 = 1.17 EURO
- ✔ Knowledge to the action.
  - Can predict the USD , EURO value in the mere future from history of information
  - In 1900 what is the equivalent of INR for USD
  - In 1950, 2000, etc.,
    - What will be the equivalent value in 2030?

Purpose of DBMS

Purpose of DBMS:

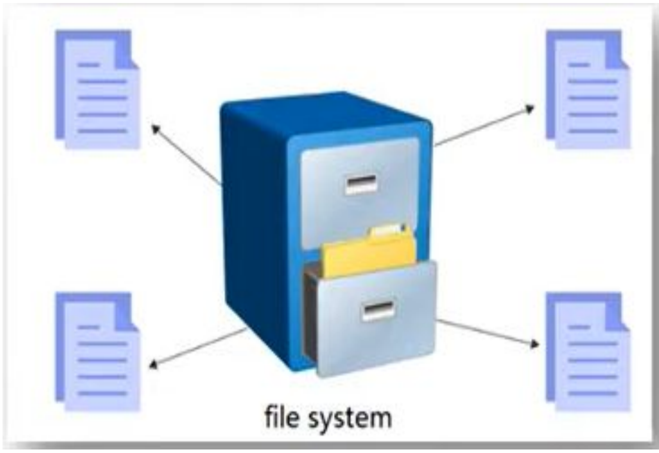
The diagram given explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS



About the File System vs DBMS

What is file System?

- A file Management system is a DBMS that allows access to single files or tables at a time.
- In a file System, data is directly stored in set of files.
- It contains flat files that have no relation to other files



About the File System vs DBMS

What is DBMS?

- A database Management System is application software that allows to efficiently define, create, maintain and share databases.
- Defining a database involves specifying the data types, structures and constraints of the data to be stored in the database.

Advantages of DBMS over File System

Data redundancy and inconsistency

Redundancy is the concept of repetition of data i.e. each data may have more than a single copy.

- The file system cannot control redundancy of data as each user defines and maintains the needed files for a specific application to run.
- There may be a possibility that two users are maintaining same files data for different applications.
- Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users.
- Example :Address and phone number of particular customer may appear in a file that consists of personal information and savings account also.

Advantages of DBMS over File System

Data Sharing / Difficulty in accessing data:

- File System does not allow sharing of data or sharing is too complex.
- Whereas in DBMS, data can be shared easily due to centralized system.

Examples : Google drive, Google forms etc.

Data isolation :

- Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

Advantages of DBMS over File System

Security problems:

- Not every user of the database system should be able to access all the data.
- For example, in a university, payroll personnel need to see only that part of the database that has financial information.
- They do not need access to information about academic records.
- But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

Enforcement of security :

- Security enforced by DBA.
- DBA sets the security constraints to users for the common operations like adding, deleting, retrieving & updating of records.
- DBA assign different privileges to the users.

Advantages of DBMS over File System

Integrity problems:

- The data values stored in the database must satisfy certain types of consistency constraints.
- Suppose the university maintains an account for each department, and records the balance amount in each account.
- Suppose also that the university requires that the account balance of a department may never fall below zero.
- Developers enforce these constraints in the system by adding appropriate code in the various application programs.
- However, when new constraints are added, it is difficult to change the programs to enforce them.
- The problem is compounded when constraints involve several data items from different files.

Advantages of DBMS over File System

Atomicity problems:

- A computer system, like any other device, is subject to failure.
- In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.
- Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.
- That is, the funds transfer must be atomic—it must happen in its entirety or not at all.
- It is difficult to ensure atomicity in a conventional file-processing system.

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data.
- That is, the system hides certain details of how the data are stored and maintained.

Data Abstraction

- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user.
- Data abstraction has different views and support in attaining data independence which is used to enhance the security of data.
- The database systems consist of complicated data structures and relations.
  - To make the easy access of data by the users the complications are kept hidden and the remaining part of the database is accessible to the them through data abstraction
  - developers hide the complexity from users through several levels of abstraction, to simplify users’ interactions with the system:

Levels of Abstraction

Physical level: describes how the data are (e.g., student) actually was stored.

Logical level: describes data stored in database, and the relationships among the data.

```
type student = record
    student_reg_number : integer;
    student_name : string;
    student_degree : string;
    customer_mobile : integer;
    student_email : string
end;
```

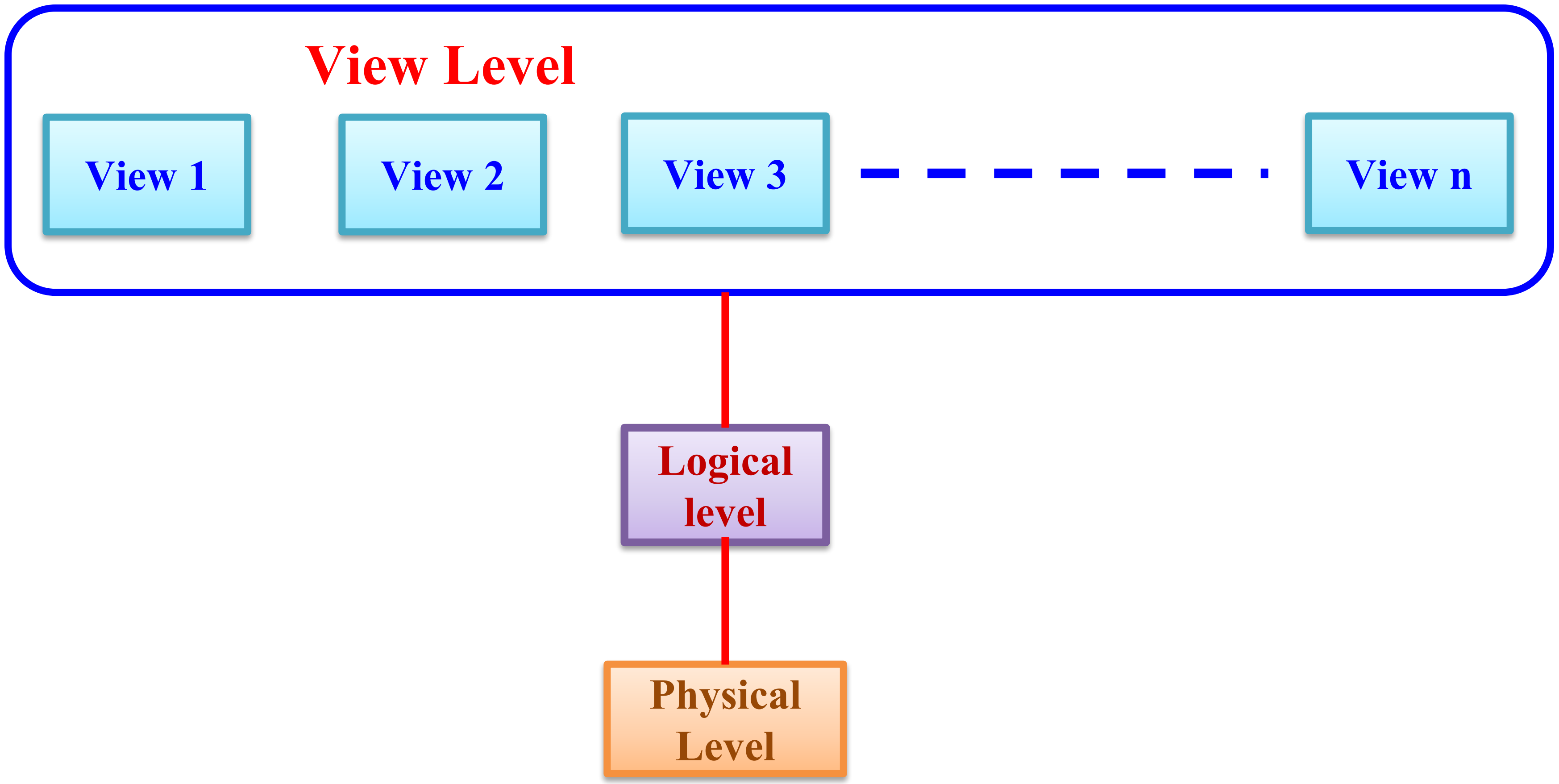
- This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction

Levels of Abstraction

View level:

- The highest level of abstraction describes only part of the entire database.
- Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.
- Many users of the database system do not need all this information; instead, they need to access only a part of the database.
- The view level of abstraction exists to simplify their interaction with the system.
- The system may provide many views for the same database.

View of Data


View Level
View 1
View 2
View 3
View n
Logical level
Physical Level

Instances and Schemas

- Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database.
- The overall design of the database is called the database schema.
- Schemas are changed infrequently, if at all.
- The concept of database schemas and instances can be understood by analogy to a program written in a programming language.

Instances and Schemas

Instances

- A database schema corresponds to the variable declarations (along with associated type definitions) in a program.
- Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema.

Schema :

- Schema is the overall description of the database. The basic structure of how the data will be stored in the database is called schema.

Instances and Schemas

- Database systems have several schemas, partitioned according to the levels of abstraction.
- The physical schema describes the database design at the physical level, while the logical schema describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called subschemas, that describe different views of the database.

Instances and Schemas

- Of these, the logical schema is by far the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema.
- The physical schema is hidden beneath the logical schema, and can usually be changed easily without affecting application programs.

Instances and Schemas

Schema is of three types: Logical Schema, Physical Schema and view Schema.

- Logical Schema – It describes the database designed at logical level.
- Physical Schema – It describes the database designed at physical level.
- View Schema – It defines the design of the database at the view level.

Difference between Schema and Instance

Schema	Instance
It is the overall description of the database.	It is the collection of information stored in a database at a particular moment.
Schema is same for whole database.	Data in instances can be changed using addition, deletion, updation.
Does not change Frequently.	Changes Frequently.
Defines the basic structure of the database i.e how the data will be stored in the database.	It is the set of Information stored at a particular time.

Data Models

- Data models is , a collection of tools for describing Data and its relationships, Semantics and consistency constraints.
- A data model provides a way to describe the design of a database at the physical, logical, and view levels.
- There are a number of different data are available in DBMS.
- The data models can be classified into four different categories:

Data Models

Relational Model :

- The relational model uses a collection of tables to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has a unique name. Tables are also known as relations.
- The relational model is an example of a record-based model.
- Record-based models are so named because the database is structured in fixed-format records of several types.

Data Models

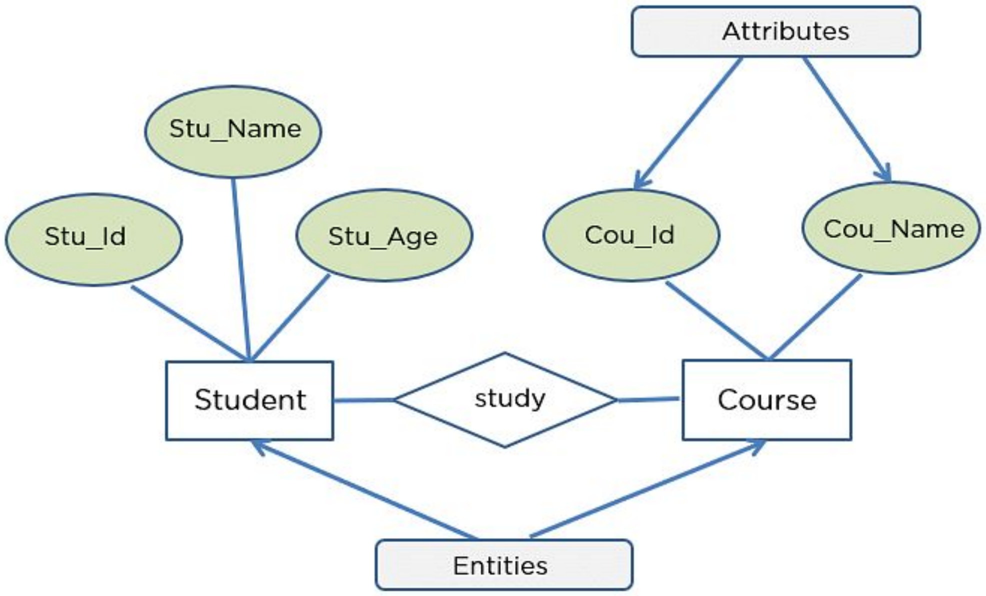
Relational Model :

- Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type.
- The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

Data Models

Entity-Relationship Model:

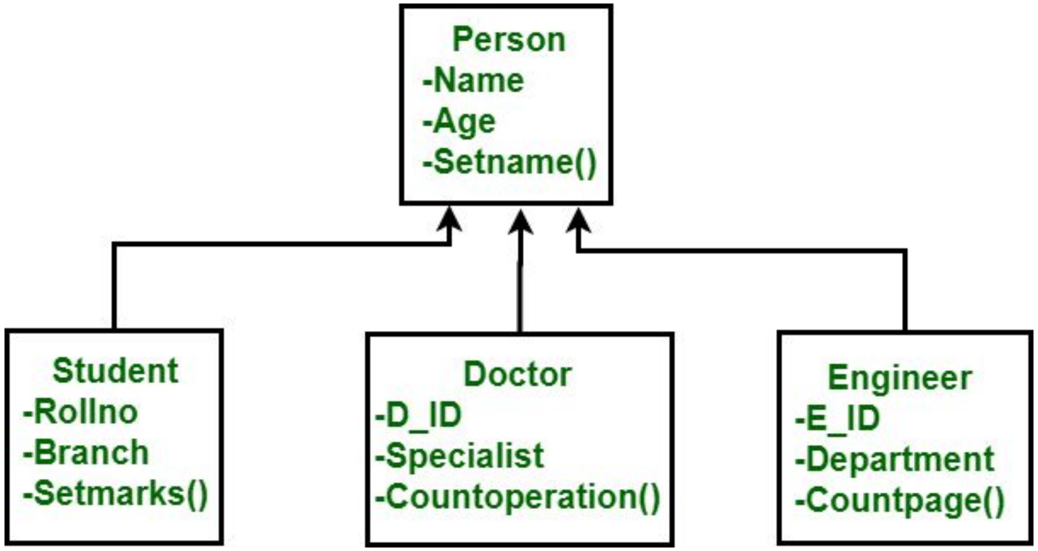
- The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects.
- An entity is a "thing" or "object" in the real world that is distinguishable from other objects.



Data Models

Object-Based Data Model :

- Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology.
- This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.
- The object-relational data model combines features of the object-oriented data model and relational data model.



Data Models

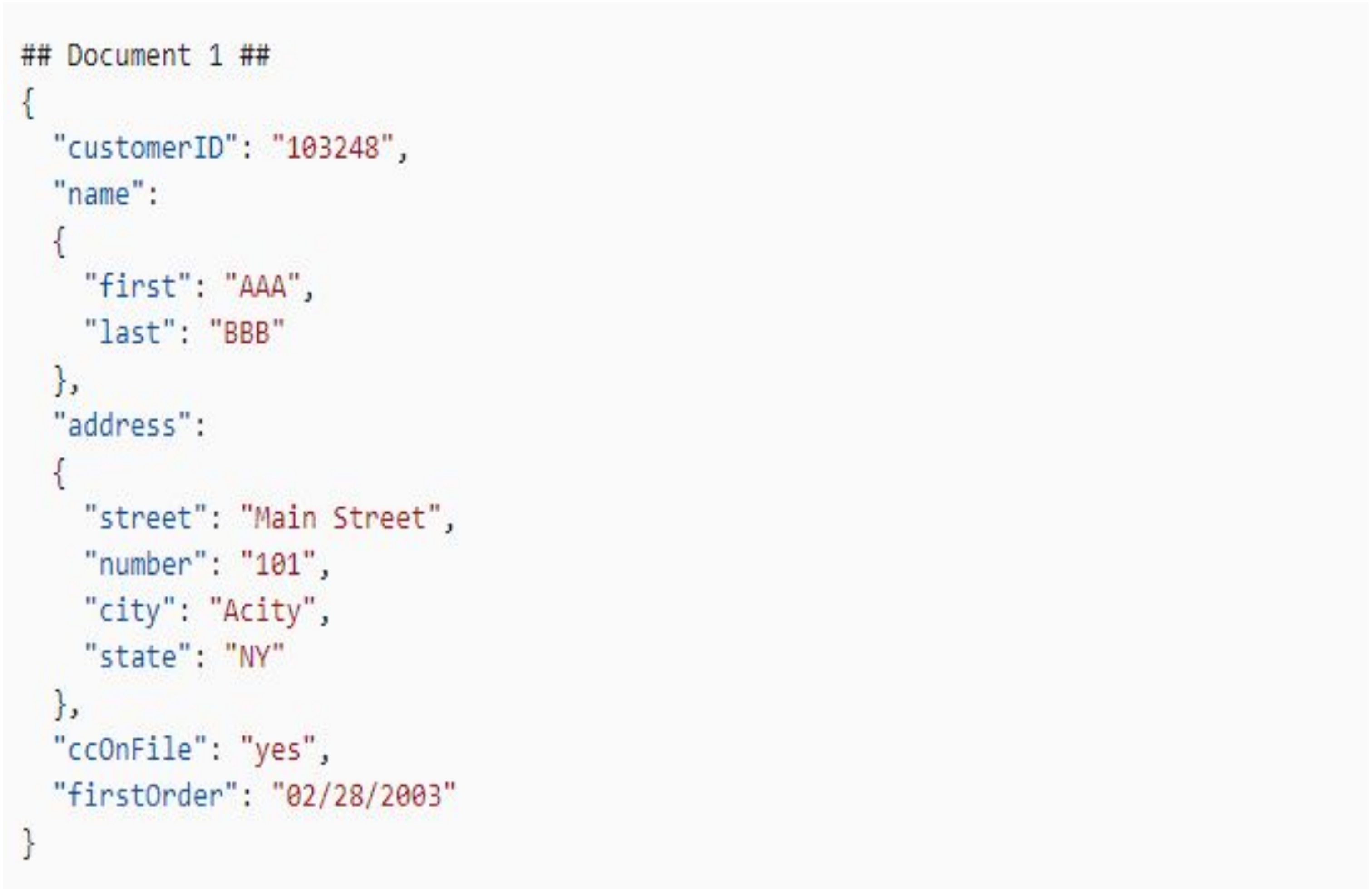
Semi structured Data Model:

- The semi structured data model permits the specification of data where individual data items of the same type may have different sets of attributes.
- The Extensible Markup Language (XML) is widely used to represent semistructured data.
  - Other older models:
    - Network model
    - Hierarchical model

Data Models

Example : Semi structured Data Model:

- Semi-structured data is information that doesn’t consist of Structured data (relational database) but still has some structure to it.

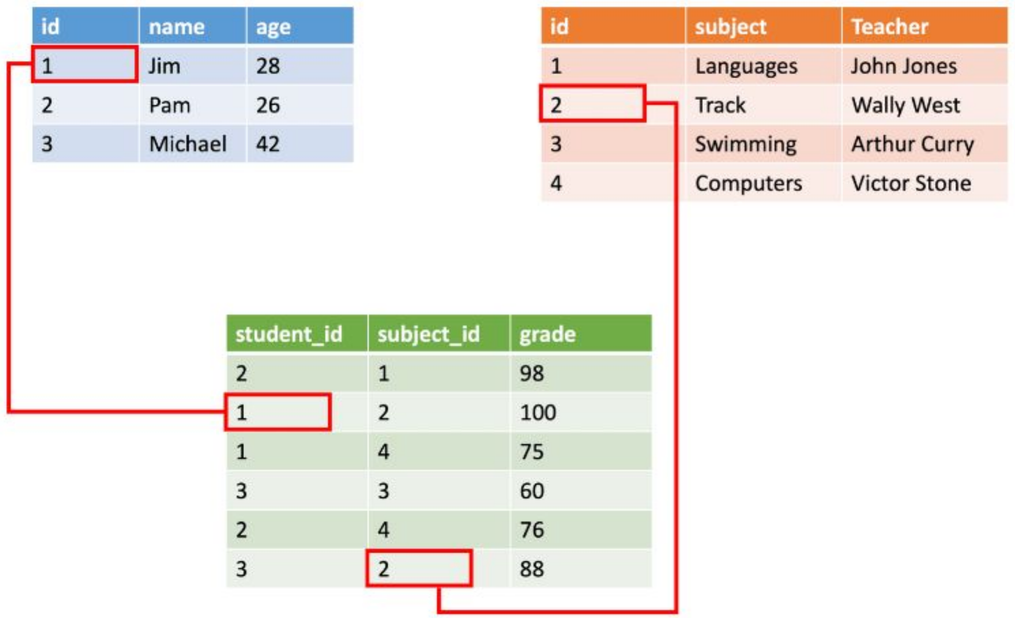


```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

Data Models

Example : structured data

- Structured data is generally tabular data that is represented by columns and rows in a database.
- Databases that hold tables in this form are called *relational databases.*
- The mathematical term "*relation*" specify to a formed set of data held as a table.
- In structured data, all row in a table has the same set of columns.
- SQL (Structured Query Language) programming language used for structured data.



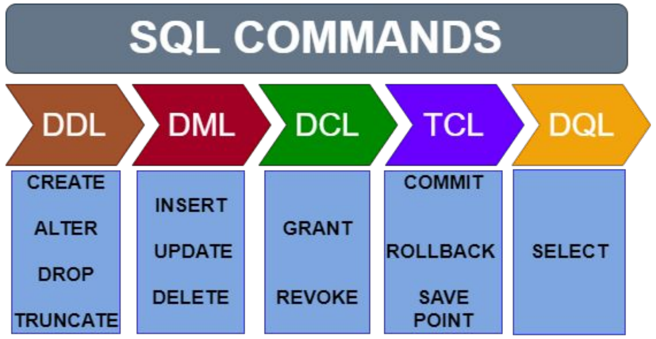
id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject	Teacher
1	Languages	John Jones
2	Track	Wally West
3	Swimming	Arthur Curry
4	Computers	Victor Stone

student_id	subject_id	grade
2	1	98
1	2	100
1	4	75
3	3	60
2	4	76
3	2	88

Structured Query Language (SQL)

- ✔ SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987
- ✔ Common language for all Databases
- ✔ Fourth generation Language
- ✔ Non procedural Language
- ✔ Commands like an normal English statements
- ✔ **SQL is not a case sensitive language**
- ✔ All SQL statements should ended with terminator , the default terminator is **semi-colon (;)**
- ✔ Based on the operation SQL divided into three categories
  - – DDL (Data Definition Language)
  - – DML (Data Manipulation Language)
  - – DCL (Data Control Language)



Database Languages

- A database system provides a Data-definition Language to specify the database schema and a data-manipulation language to express database queries and up- dates.
- In practice, the data-definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language

Data types in SQL (ORACLE)

- ✓ CHAR
- ✓ VARCHAR2
- ✓ NUMBER
- ✓ DATE
- ✓ RAW
- ✓ LONG
- ✓ CLOB
- ✓ BLOB , etc.,

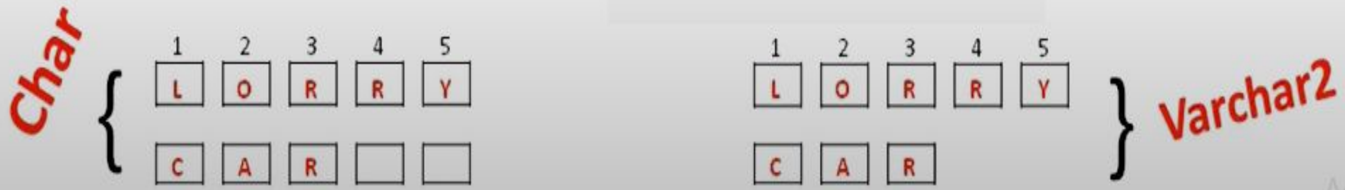
Data types in SQL (ORACLE)

1. Data types are used to specify the basic behavior of a column in the table

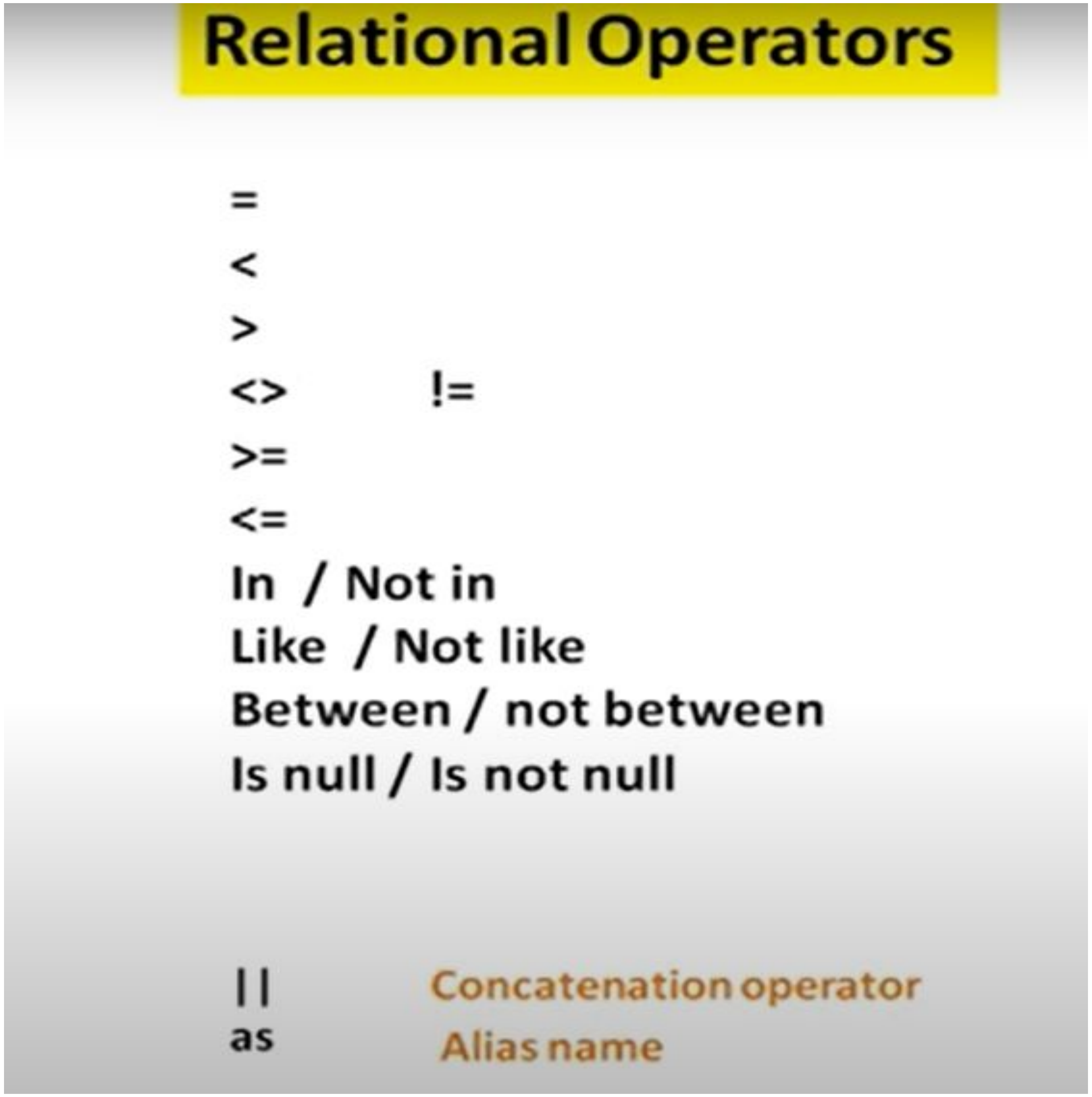
2. Datatype (length)

ID	NAME	DOB	PINCODE
10	Mani	11/12/1990	600001
11	Raja	10/1/2005	600020
12	Anbu	7/7/1997	600050

NUMBER		Numbers/Integers	
CHAR	2000	ALPHA + NUMBER + SPL CHAR	
VARCHAR2	4000	ALPHA + NUMBER + SPL CHAR	VARIABLE CHARACTER
LONG	2GB	ALPHA + NUMBER + SPL CHAR	only once/table
CLOB	4GB	ALPHA + NUMBER + SPL CHAR	CHARACTER LARGE OBJECT
BLOB	8GB	image	BINARY LARGE OBJECT
DATE		DATE (12-11-2015)	
TIMESTAMP		DATE + TIME (12-11-2015 01:30:15:28)	
BFILE		Path/links/url	
XMLTYPE			



RELATIONAL OPERATORS



Relational Operators

=

<

>

<>   !=

>=

<=

In / Not in

Like / Not like

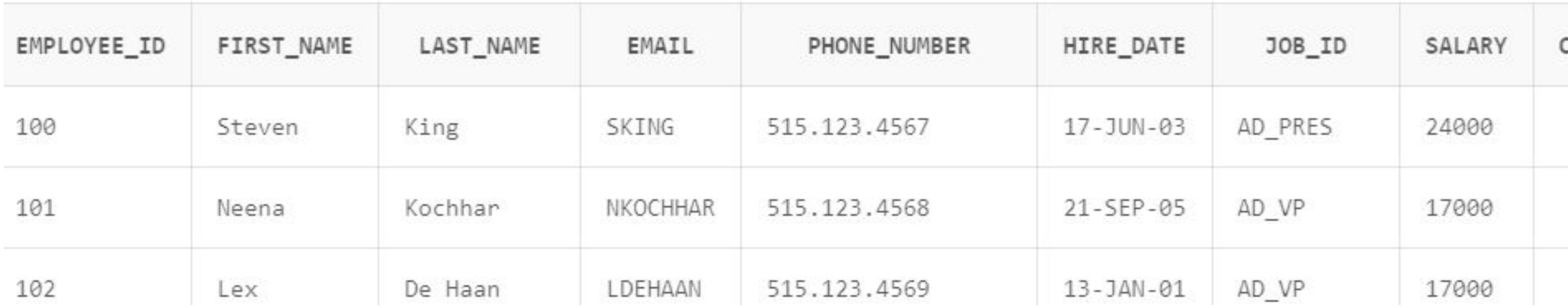
Between / not between

Is null / Is not null

||	Concatenation operator
as	Alias name

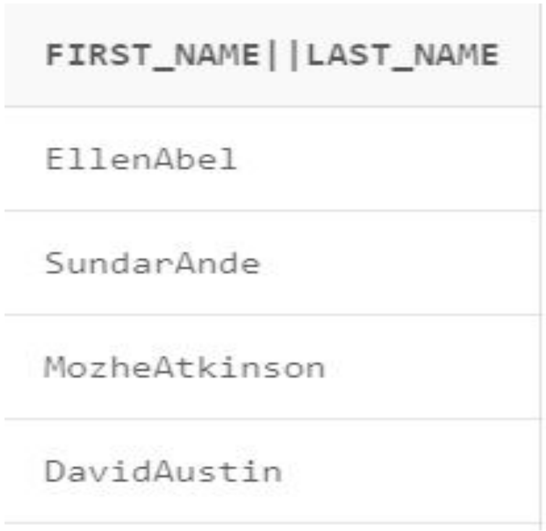
RELATIONAL OPERATORS – ||

select * from hr.employees; https://livesql.oracle.com



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000

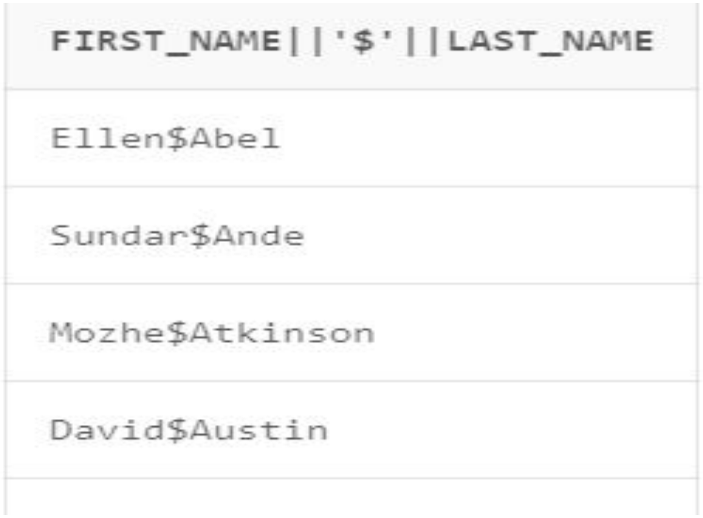
select first_name || last_name from hr.employees;



FIRST_NAME||LAST_NAME
EllenAbel
SundarAnde
MozheAtkinson
DavidAustin

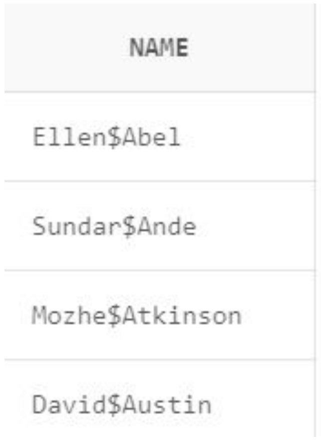
RELATIONAL OPERATORS - AS

```
select first_name ||'$' || last_name from hr.employees;
```



FIRST_NAME||'$'||LAST_NAME
Ellen$Abel
Sundar$Ande
Mozhe$Atkinson
David$Austin

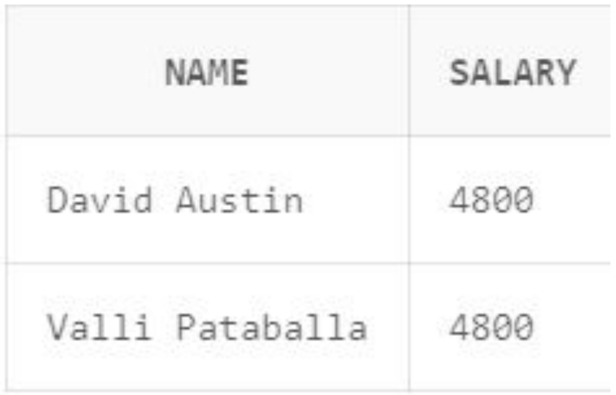
```
select first_name ||'$' || last_name AS NAME from hr.employees;
```



NAME
Ellen$Abel
Sundar$Ande
Mozhe$Atkinson
David$Austin

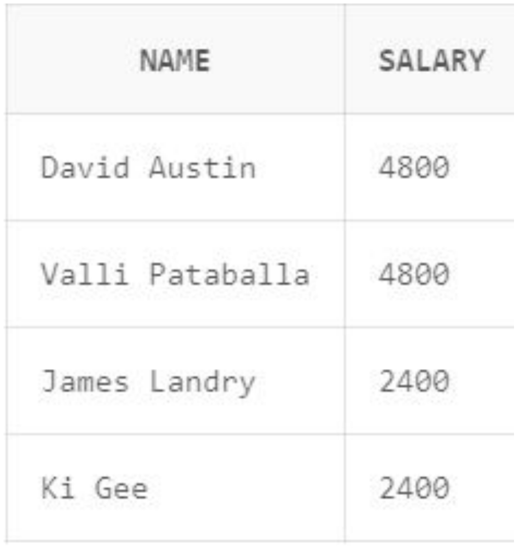
RELATIONAL OPERATORS - IN

```
select first_name ||' ' || last_name  AS NAME, SALARY from
hr.employees where salary = 4800;
```



NAME	SALARY
David Austin	4800
Valli Pataballa	4800

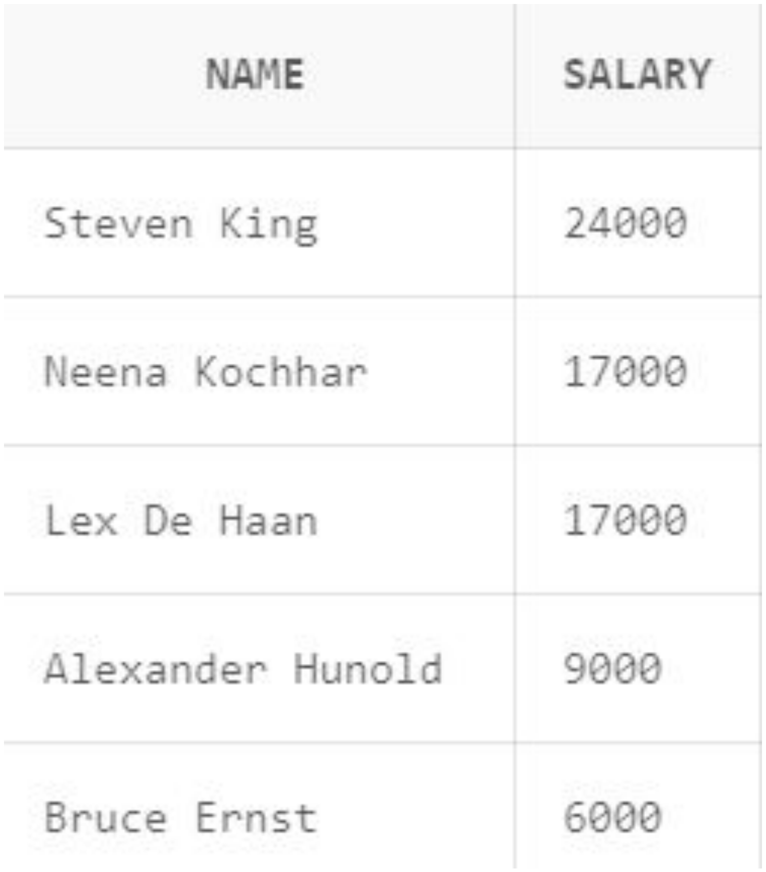
```
select first_name ||' ' || last_name  AS NAME, SALARY from hr.employees
where salary IN (4800,2400);
```



NAME	SALARY
David Austin	4800
Valli Pataballa	4800
James Landry	2400
Ki Gee	2400

RELATIONAL OPERATORS – NOT IN

```
select first_name ||' ' || last_name  AS NAME, SALARY from
hr.employees where salary NOT IN (4800,2400);
```

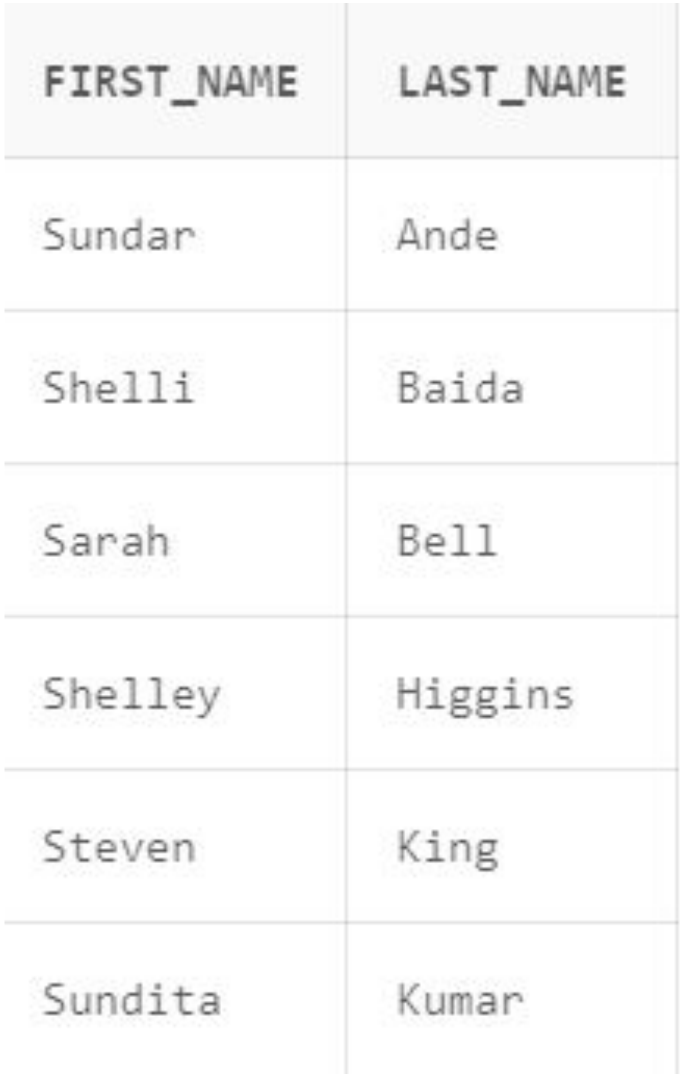


NAME	SALARY
Steven King	24000
Neena Kochhar	17000
Lex De Haan	17000
Alexander Hunold	9000
Bruce Ernst	6000

RELATIONAL OPERATORS – LIKE

First letter

select first_name,last_name from hr.employees where first_name LIKE ('S%');

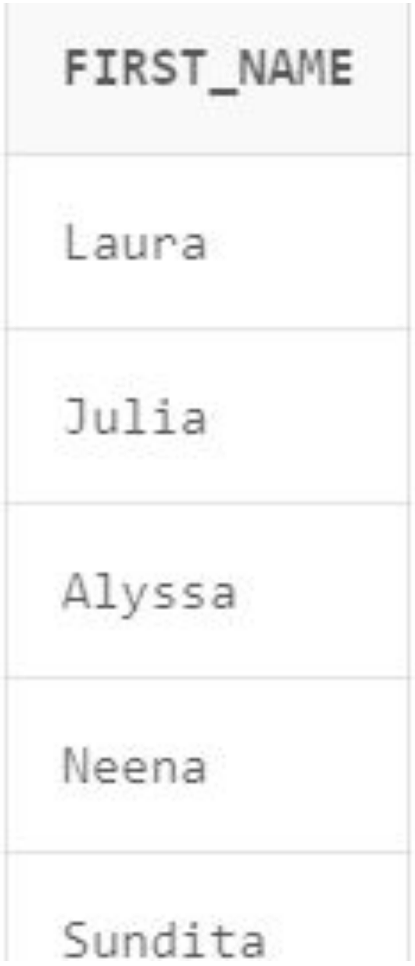


FIRST_NAME	LAST_NAME
Sundar	Ande
Shelli	Baida
Sarah	Bell
Shelley	Higgins
Steven	King
Sundita	Kumar

RELATIONAL OPERATORS – LIKE

Last letter

select first_name from hr.employees where first_name LIKE ('%a');

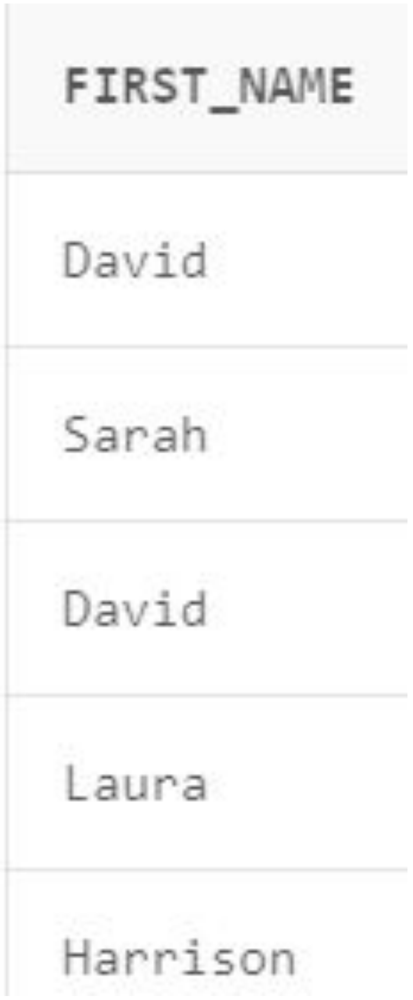


FIRST_NAME
Laura
Julia
Alyssa
Neena
Sundita

RELATIONAL OPERATORS – LIKE

Second letter

select first_name from hr.employees where first_name LIKE ('_a%');

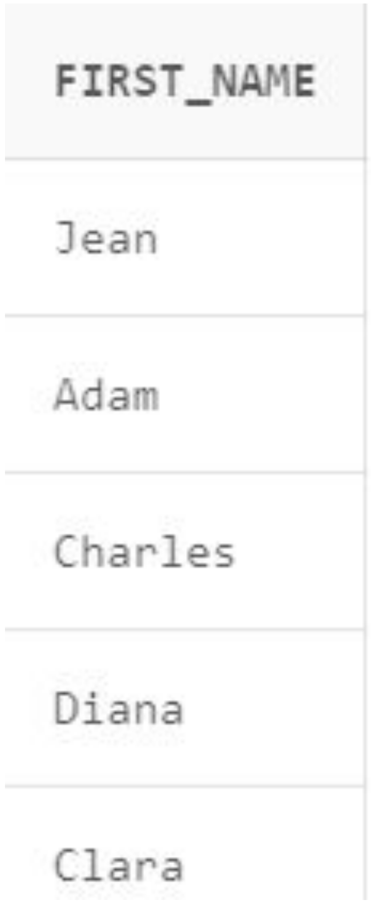


```
FIRST_NAME
David
Sarah
David
Laura
Harrison
```

RELATIONAL OPERATORS – LIKE

3rd letter

```
select first_name from hr.employees where first_name LIKE ('__a%');
```



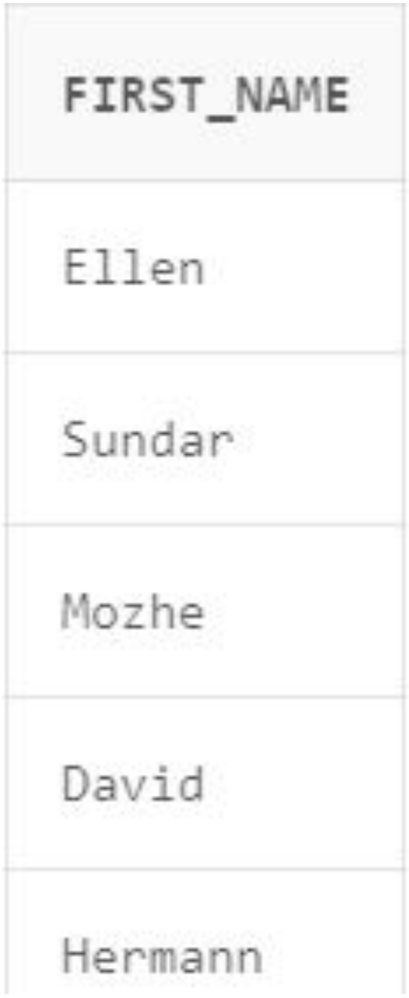
FIRST_NAME
Jean
Adam
Charles
Diana
Clara

RELATIONAL OPERATORS – NOT LIKE

Second letter

select first_name from hr.employees where first_name NOT LIKE ('__a%');

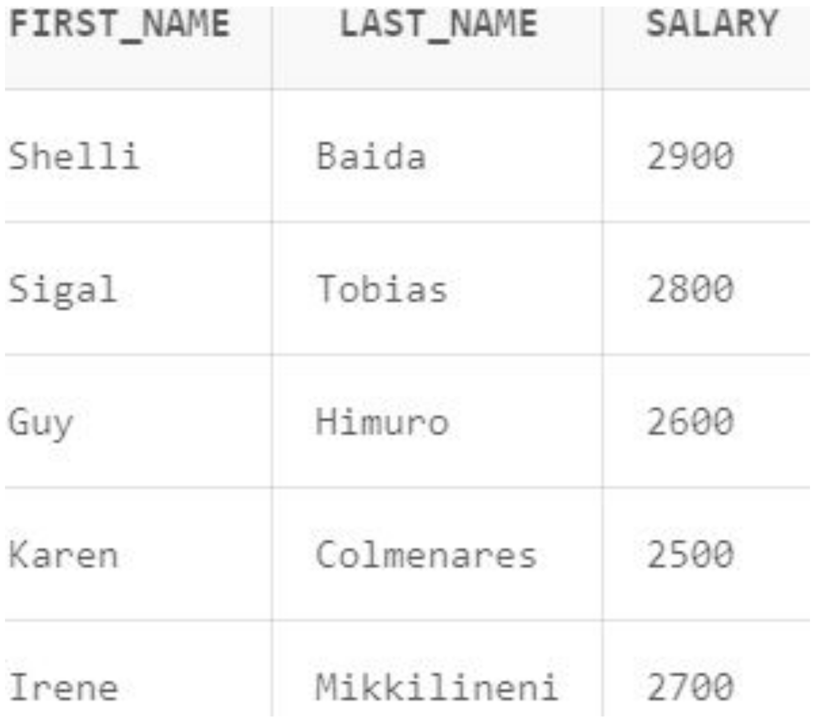
select first_name from hr.employees where first_name NOT LIKE UPPER ('a%');



FIRST_NAME
Ellen
Sundar
Mozhe
David
Hermann

RELATIONAL OPERATORS – BETWEEN

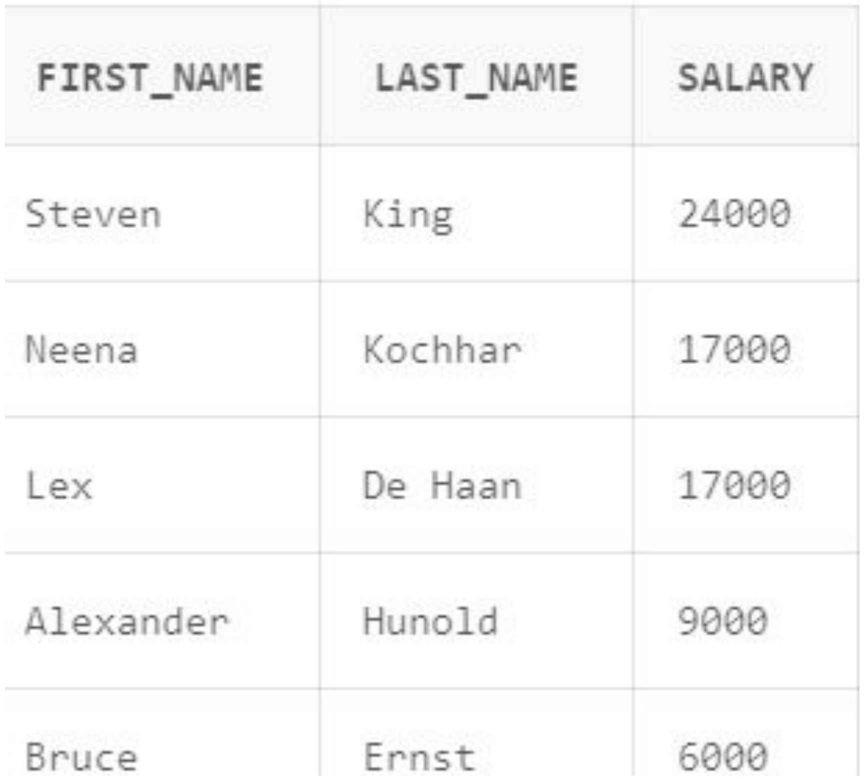
select first_name, last_name, salary from hr.employees where salary BETWEEN 2400 AND 3000;



FIRST_NAME	LAST_NAME	SALARY
Shelli	Baida	2900
Sigal	Tobias	2800
Guy	Himuro	2600
Karen	Colmenares	2500
Irene	Mikkilineni	2700

RELATIONAL OPERATORS – BETWEEN

select first_name, last_name, salary from hr.employees where salary NOT BETWEEN 2400 AND 3000;



FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Neena	Kochhar	17000
Lex	De Haan	17000
Alexander	Hunold	9000
Bruce	Ernst	6000

RELATIONAL OPERATORS – <>

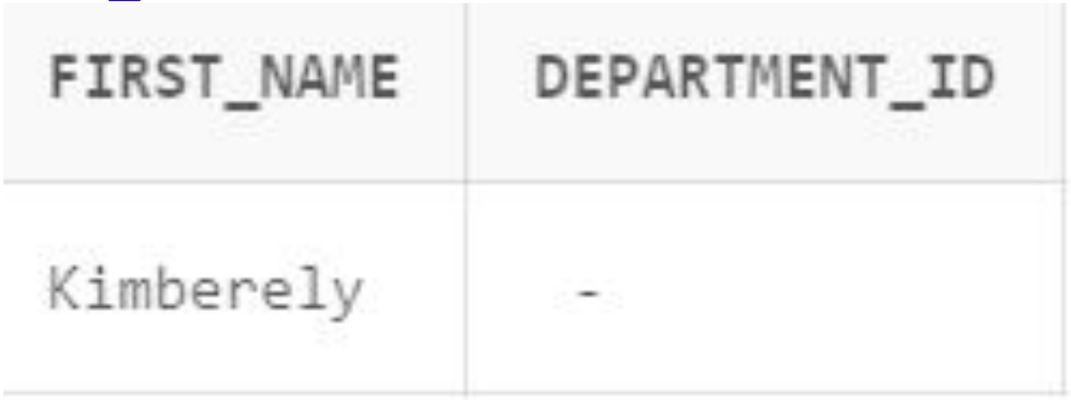
```
select first_name, last_name, salary from hr.employees where
salary  > 4800 order by asc;
```

```
select first_name, last_name, salary from hr.employees
where salary  < > 4800 order by asc;
```

```
select first_name, last_name, salary from hr.employees
where salary  != 4800 order by asc;
```

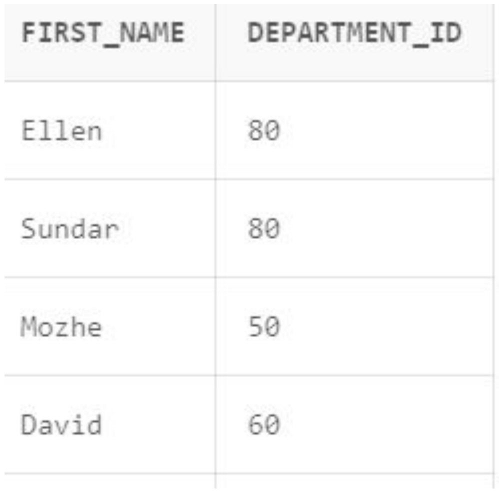
RELATIONAL OPERATORS – null / not null

select first_name, department_id from hr.employees where department_id is null;



FIRST_NAME	DEPARTMENT_ID
Kimberely	-

select first_name, department_id from hr.employees where department_id is not null;



FIRST_NAME	DEPARTMENT_ID
Ellen	80
Sundar	80
Mozhe	50
David	60

DDL COMMANDS

Data Definition Language (DDL)

- ✓ DDL is the subset of SQL and part of DBMS
- ✓ DDL relates only with base tables structure and it is no where relates with the information stored in the table.
- ✓ **Note : All the DDL command statements are AUTO COMMIT Statements**
- ✓ DDL consists of the following commands
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

Data Definition Language (DDL)

CREATE COMMAND

Used to create a new object / schema with a defined structure

Syntax :

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

```
CREATE TABLE EMP
      (EMPNO NUMBER(4) NOT NULL,  ENAME VARCHAR2(10),  JOB VARCHAR2(9),
MGR NUMBER(4), HIREDATE DATE, SAL NUMBER(7, 2), COMM NUMBER(7, 2),
DEPTNO NUMBER(2));
```

Data Definition Language (DDL)

ALTER COMMAND

- ✓ Alter command used to modify the base table structure

- ✓ Using this command
  - a new column can be added with restrictions
  - column data width can be increased / decreased with restrictions
  - a column can be dropped

- ✓ Two key words are using in this command
  - ADD
  - MODIFY

Data Definition Language (DDL)

ALTER COMMAND

SYNTAX

```
ALTER TABLE table_name ADD / MODIFY column_name
datatype;
```

EXAMPLE 1: To add a new column in a table

```
ALTER TABLE emp ADD phone_no number(10);
```

EXAMPLE 2 : TO modify the existing column data width

```
ALTER TABLE emp MODIFY phone_no number(13);
```

To create a DDL to perform creation of table, alter, modify and drop column.

DDL COMMANDS

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

Syntax:

```
Create table <table name> (<col1>
<datatype>(<size>),<col2> <datatype><size>));
```

Ex: create table emp(empno number(4) primary key, ename char(10));

To create a DDL to perform creation of table, alter, modify and drop column.

2. Modifying the structure of tables.

Add new columns

Syntax:

Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size));

Ex: alter table emp add(sal number(7,2));

To create a DDL to perform creation of table, alter, modify and drop column.

3. Dropping a column from a table.

Syntax:

Alter table <tablename> drop column <col>;

Ex: alter table emp drop column sal;

To create a DDL to perform creation of table, alter, modify and drop column.

4. Modifying existing columns.

Syntax:

```
Alter table <tablename>
modify(<col><newdatatype>(<newsize>));
```

Ex: `alter table emp modify(ename varchar2(15));`

To create a DDL to perform creation of table, alter, modify and drop column.

5. Renaming the tables

Syntax:

Rename <oldtable> to <new table>;

Ex: rename emp to emp1;

To create a DDL to perform creation of table, alter, modify and drop column.

6. truncating the tables.

Syntax:

Truncate table <tablename>;

Ex: trunc table emp1;

- Whereas the TRUNCATE command is used to delete all the rows from the table.

To create a DDL to perform creation of table, alter, modify and drop column.

7. Destroying tables.

Syntax:

Drop table <tablename>;

Ex: drop table emp;

- DROP is a DDL(Data Definition Language) command and is used to remove table definition and indexes, data, constraints, triggers etc for that table.
- Performance-wise the DROP command is quick to perform but slower than TRUNCATE because it gives rise to complications.
- Unlike DELETE we can't rollback the data after using the DROP command.
- In the DROP command, table space is freed from memory because it permanently delete table as well as all its contents

Database System Architecture

The architecture of a database system is greatly influenced by the underlying computer system on which the database is running:

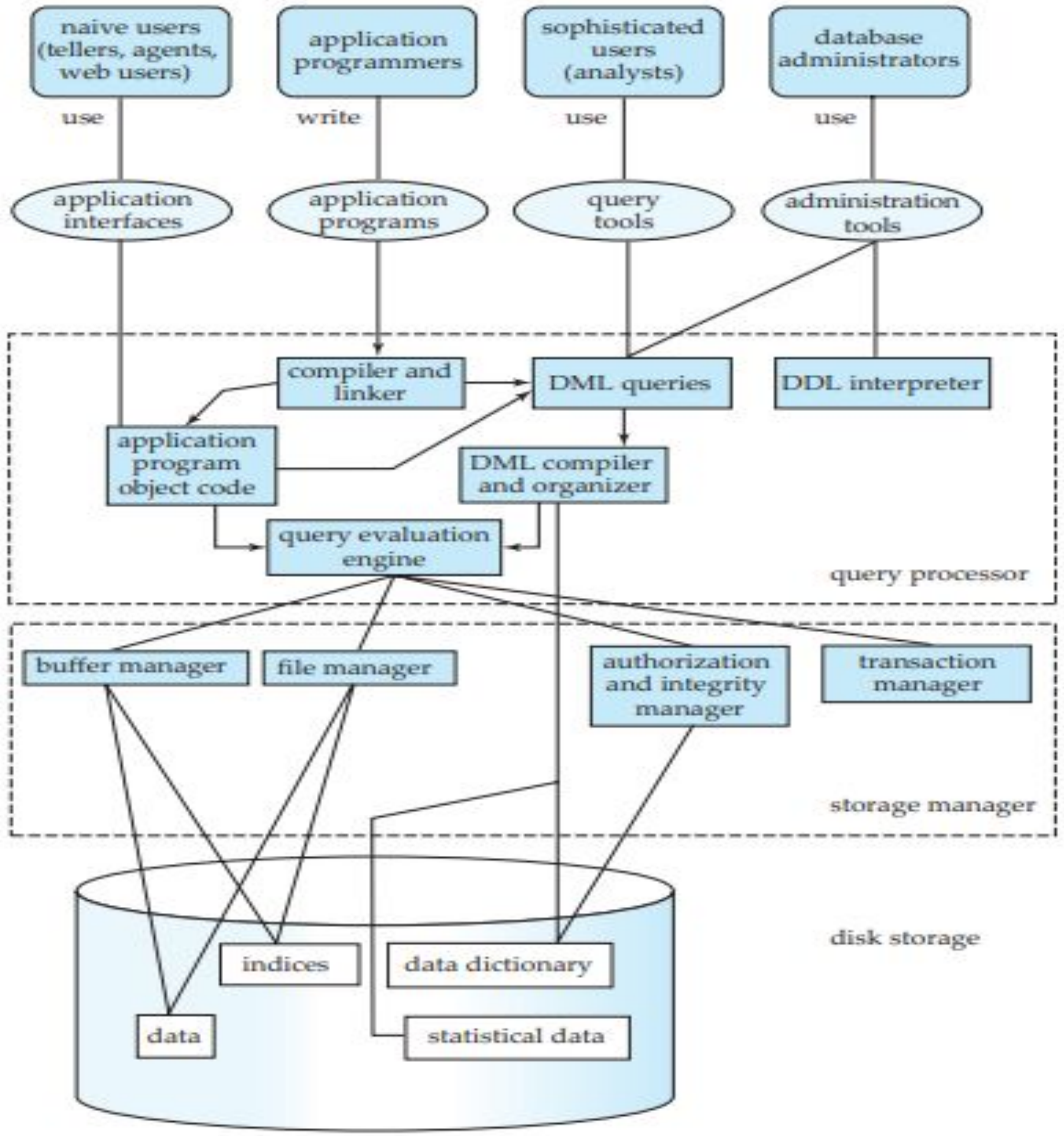
i. Centralized

ii. Client-server

iii. Parallel (multi-processor)

iv. Distributed

Database System Architecture



Database System Architecture

Database Users:

Users are differentiated by the way they expect to interact with the system:

Application programmers:

- Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
- Rapid Application Development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated users:

- Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.
- They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.

Database System Architecture

Specialized users :

- Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.
- Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Naïve users :

- Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

Database System Architecture

Database Administrator:

Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise’s information resources and needs.

Database administrator's duties include:

- **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access method definition.**
- **Schema and physical organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
- **Granting user authority to access the database:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
- **Specifying integrity constraints.**
- **Monitoring performance and responding to changes in requirements.**

Database System Architecture

Query Processor:

The query processor will accept query from user and solves it by accessing the database.

Parts of Query processor:

DDL interpreter

This will interprets DDL statements and fetch the definitions in the data dictionary.

DML compiler

a. This will translates DML statements in a query language into low level instructions that the query evaluation engine understands.

b. A query can usually be translated into any of a number of alternative evaluation plans for same query result DML compiler will select best plan for query optimization.

Query evaluation engine

This engine will execute low-level instructions generated by the DML compiler on DBMS.

Database System Architecture

Storage Manager/Storage Management:

A storage manager is a program module which acts like interface between the data stored in a database and the application programs and queries submitted to the system.

Thus, the storage manager is responsible for storing, retrieving and updating data in the database.

The storage manager components include:

Authorization and integrity manager: Checks for integrity constraints and authority of users to access data.

Transaction manager: Ensures that the database remains in a consistent state although there are system failures.

File manager: Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager: It is responsible for retrieving data from disk storage into main memory. It enables the database to handle data sizes that are much larger than the size of main memory.

Data structures implemented by storage manager.

Data files: Stored in the database itself.

Data dictionary: Stores metadata about the structure of the database.

Indices: Provide fast access to data items.

Statistical Data :Contains the statistics of all information

DML COMMANDS

Data-Manipulation Language

- A Data-Manipulation Language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:
- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

DATA MANIPULATION LANGUAGE

- **Structured Query Language (SQL)** commands are used to create and maintain Databases.
- **Data Manipulation Language(DML)** is a type of SQL command used to manipulate the data present in the database.
- DML commands in SQL are used to change the data present in the database tables, views, etc.
- These commands deal with inserting data into the tables, updating the data according to the conditions, and removing the data from the tables

- INSERT
- UPDATE
- DELETE
- MERGE

DDL - CREATE

```
CREATE TABLE student (Name varchar2(20),
                      Regno number,
                      Sub_Marks1 number,
                      Sub_Marks2 number,
                      Sub_Marks3 number);
DESC student;
```

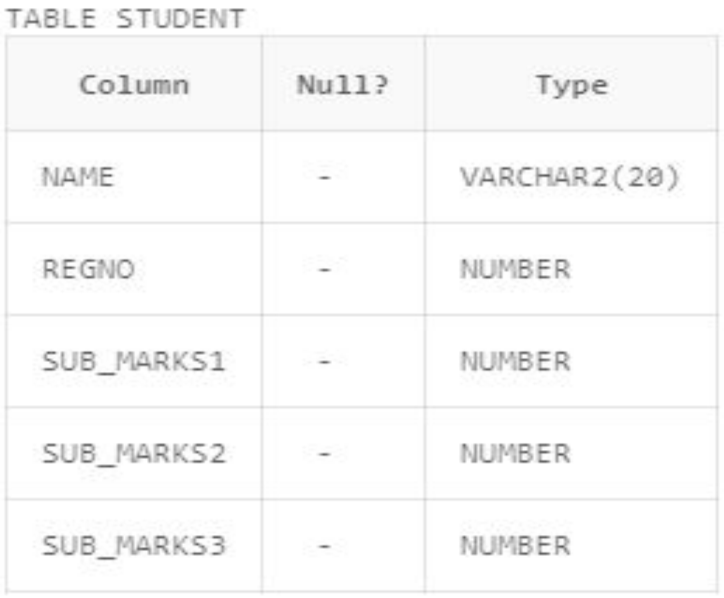


TABLE STUDENT

Column	Null?	Type
NAME	-	VARCHAR2(20)
REGNO	-	NUMBER
SUB_MARKS1	-	NUMBER
SUB_MARKS2	-	NUMBER
SUB_MARKS3	-	NUMBER

DML - INSERT

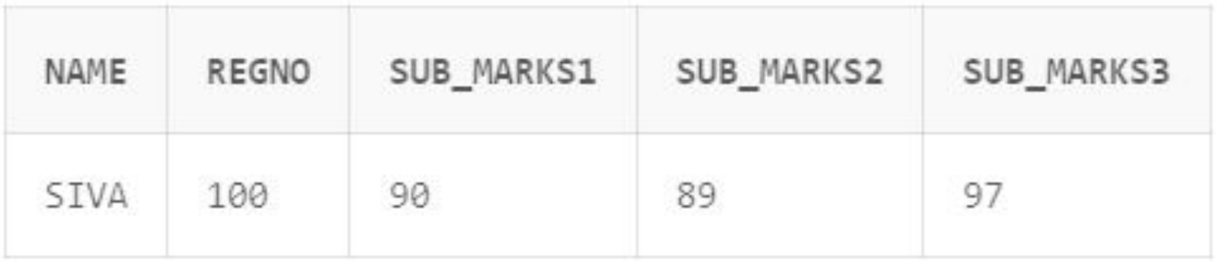
- This command is used to INSERT data into the table

Syntax :

```
INSERT INTO table_name ( column_name) VALUES ( new value);
```

Example - 1 :

```
INSERT INTO student (Name,Regno,Sub_Marks1,
Sub_Marks2, Sub_Marks3) VALUES ( 'SIVA', 100,90,89,97);
```

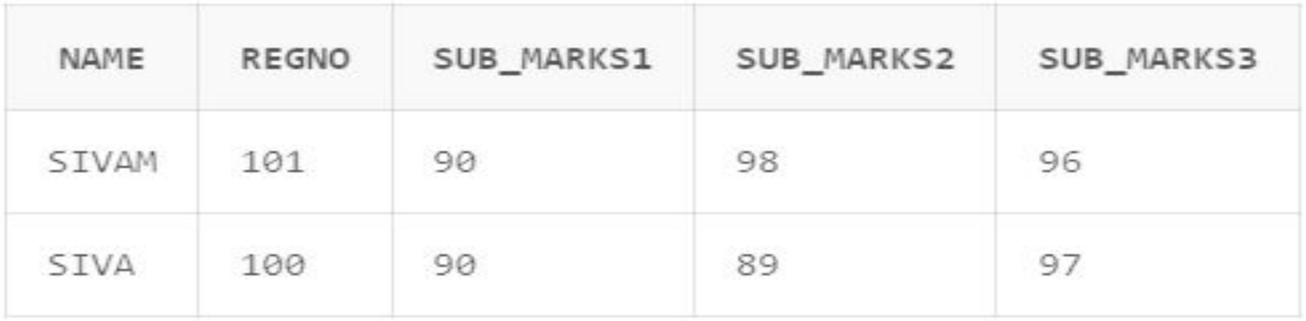


NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVA	100	90	89	97

DML - INSERT

Example -2 :

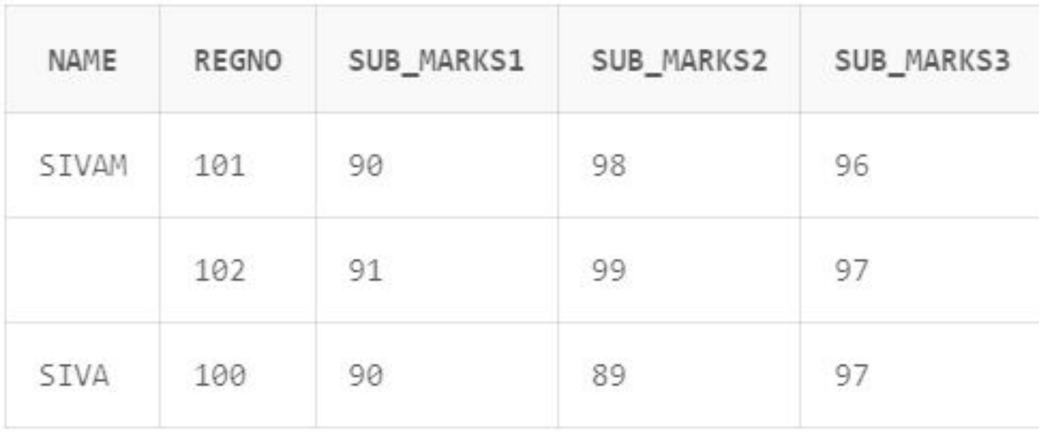
INSERT INTO student VALUES ('SIVAM', 101,90,98,96);



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVAM	101	90	98	96
SIVA	100	90	89	97

Example - 3 :

INSERT INTO student VALUES (' ', 102,91,99,97);

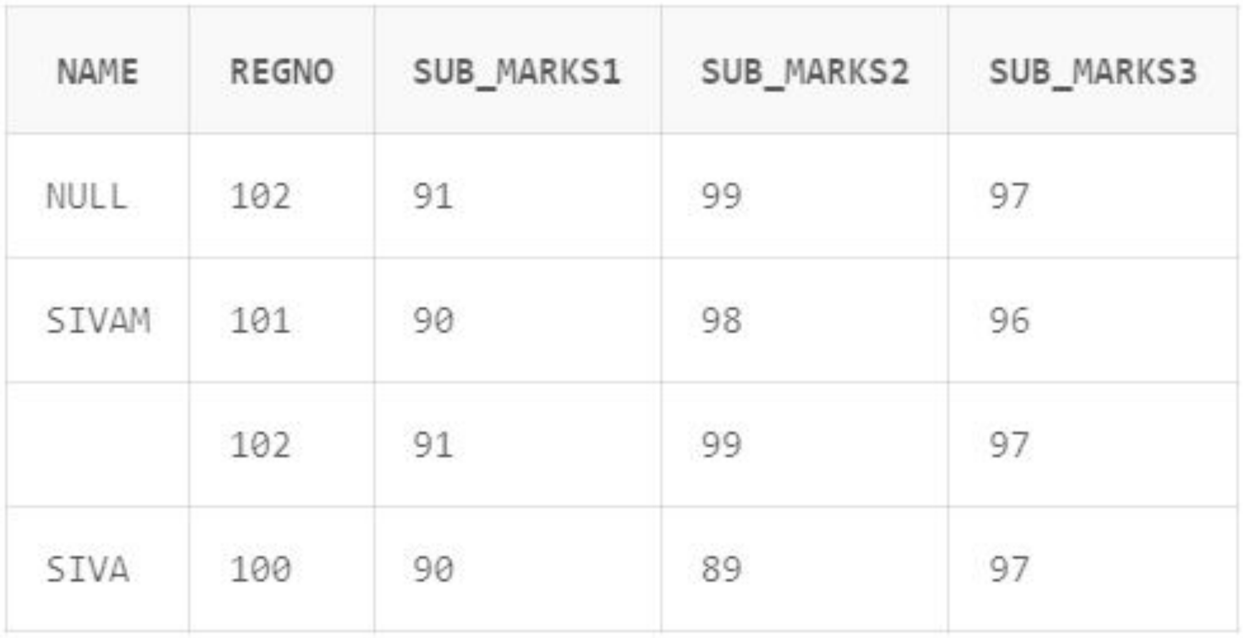


NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVAM	101	90	98	96
	102	91	99	97
SIVA	100	90	89	97

DML - INSERT

Example -4 :

INSERT INTO student VALUES ('NULL', 102,91,99,97);

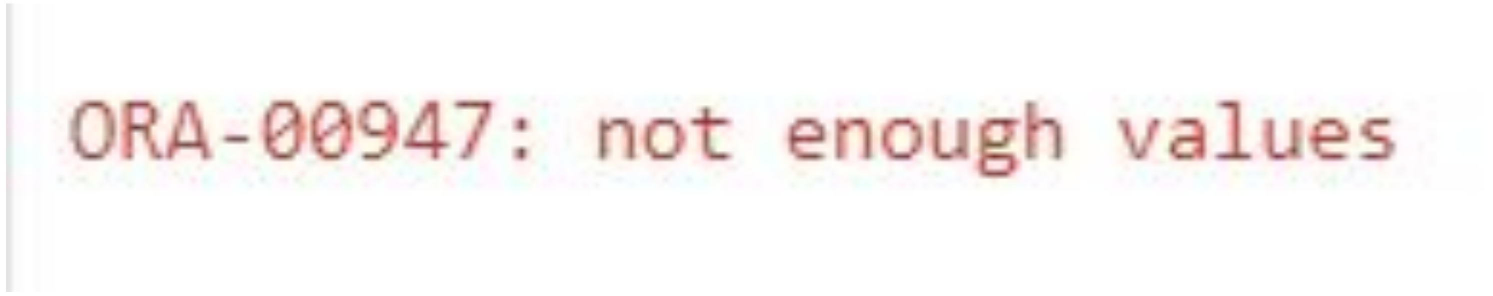


NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
NULL	102	91	99	97
SIVAM	101	90	98	96
	102	91	99	97
SIVA	100	90	89	97

DML - INSERT

Example - 5:

```
INSERT INTO student VALUES ( ' ', 100,99);
```



```
ORA-00947: not enough values
```

DML – INSERT ALL

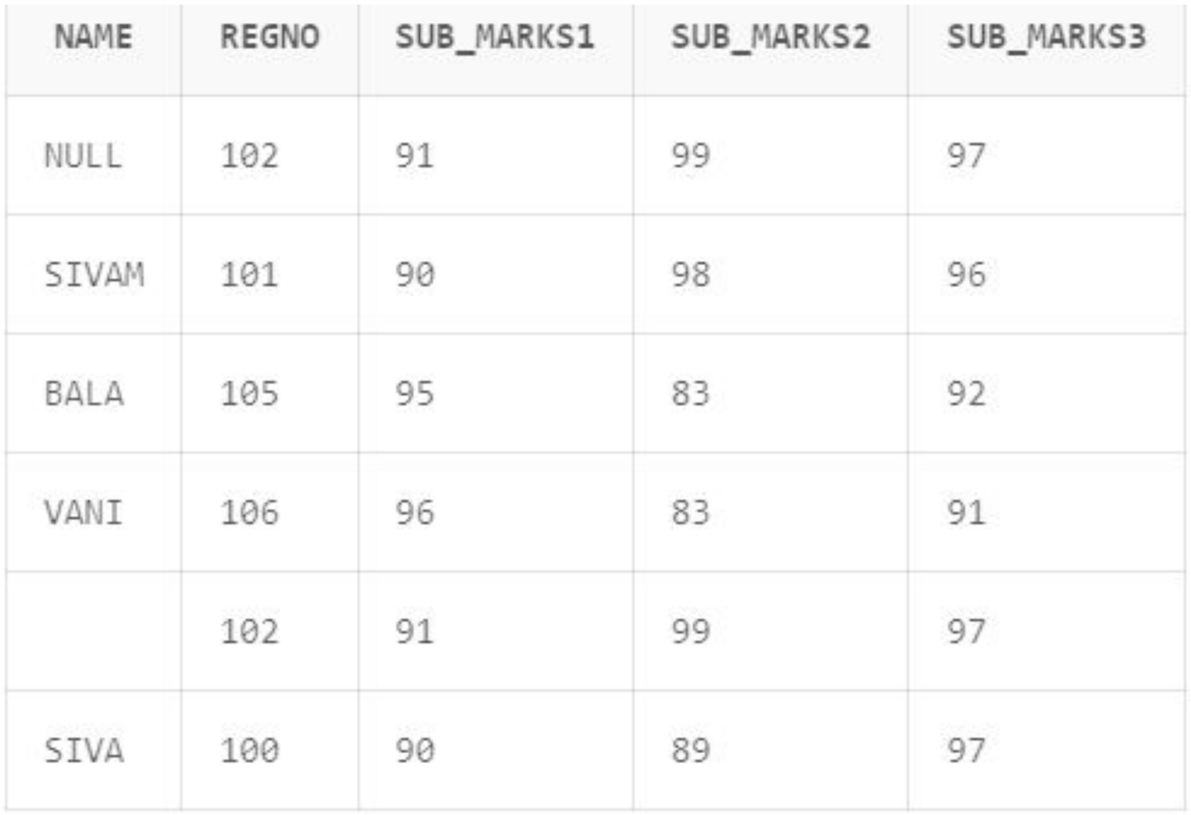
SYNTAX :

```
INSERT ALL
INTO table_name (column_name) VALUES ( new values)
INTO table_name (column_name) VALUES ( new values)
SELECT * FROM dual;
```

DML – INSERT ALL

Example - 6:

```
INSERT ALL
INTO student (Name,Regno,Sub_Marks1, Sub_Marks2, Sub_Marks3) VALUES ( 'BALA', 105,95,83,92)
INTO student (Name,Regno,Sub_Marks1, Sub_Marks2, Sub_Marks3) VALUES ( 'VANI', 106,96,83,91)
SELECT * FROM dual;
```

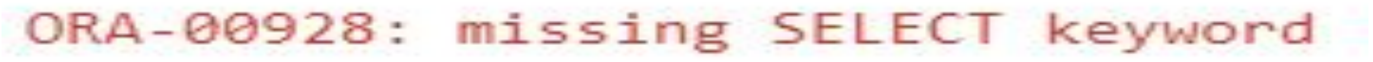


NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
NULL	102	91	99	97
SIVAM	101	90	98	96
BALA	105	95	83	92
VANI	106	96	83	91
	102	91	99	97
SIVA	100	90	89	97

DML – INSERT ALL

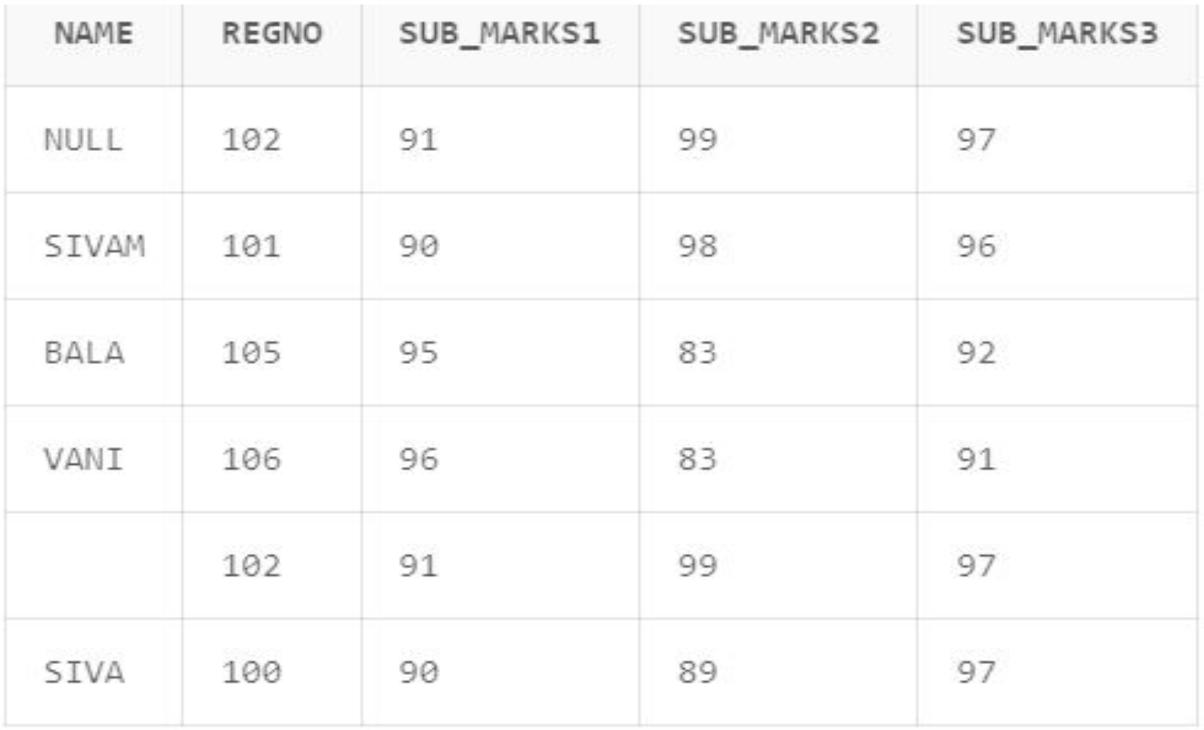
Example - 7:

```
INSERT ALL
INTO student (Name,Regno,Sub_Marks1, Sub_Marks2, Sub_Marks3) VALUES ( 'RAM', 105,95,83,92)
INTO student (Name,Regno,Sub_Marks1, Sub_Marks2, Sub_Marks3) VALUE ( 'SASI', 106,96,83,91)
SELECT * FROM dual;
```



```
ORA-00928: missing SELECT keyword
```

SELECT * FROM student;



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
NULL	102	91	99	97
SIVAM	101	90	98	96
BALA	105	95	83	92
VANI	106	96	83	91
	102	91	99	97
SIVA	100	90	89	97

DML – UPDATE

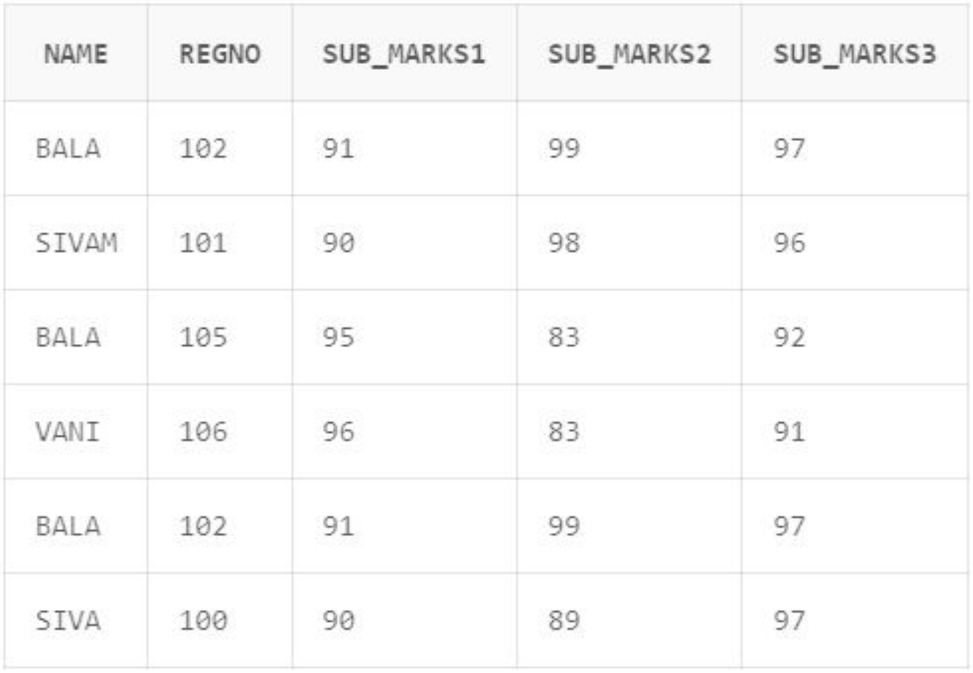
This command is used to update existing records in a table / database.

Syntax :

```
UPDATE table_name
SET <column_name>=<new_values>
WHERE conditions;
```

Example - 1 :

```
UPDATE student
SET name = 'BALA'
WHERE regno=102;
```



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
BALA	102	91	99	97
SIVAM	101	90	98	96
BALA	105	95	83	92
VANI	106	96	83	91
BALA	102	91	99	97
SIVA	100	90	89	97

DML – UPDATE

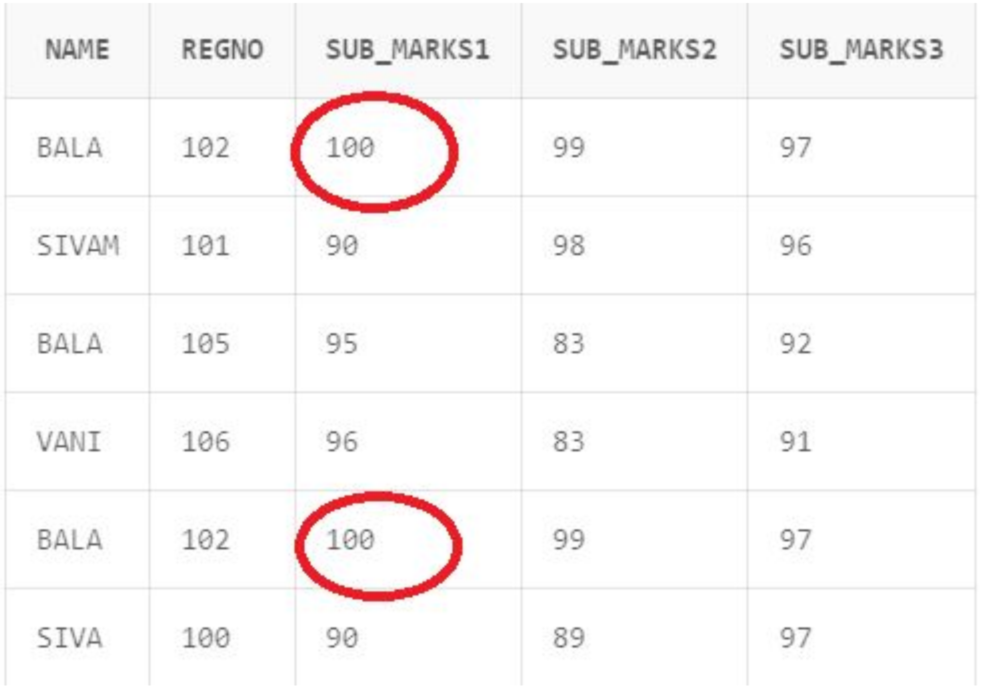
This command is used to update existing records in a table / database.

Syntax :

```
UPDATE table_name
SET <column_name>=<new_values>
WHERE conditions;
```

Example - 2 :

```
UPDATE student
SET sub_marks1 = 100
WHERE name = 'BALA' a
regno=102;
```



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
BALA	102	100	99	97
SIVAM	101	90	98	96
BALA	105	95	83	92
VANI	106	96	83	91
BALA	102	100	99	97
SIVA	100	90	89	97

DML - MERGE

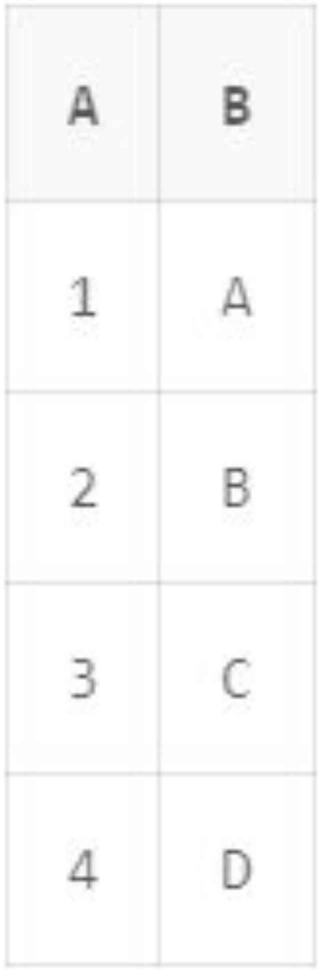
- This command is used to perform two actions (insert & update) into a target table
- It’s often called an “upsert (Update+Insert)

SYNTAX :

```
MERGE INTO target_table
USING source_table
ON (search_condition)

WHEN MATCHED THEN
     UPDATE SET col1 = value1;
                 col2 = value2,…
WHEN NOT MATCHED THEN
     INSERT (col1,col2,…)  VALUES (value1,value2,…);
```

DML - MERGE

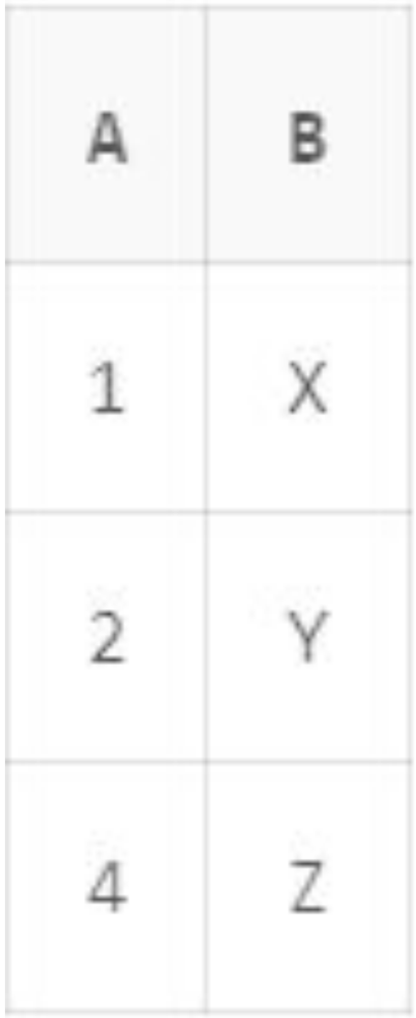
```
CREATE TABLE Test_1
(A number,
B varchar2(3)
);

INSERT INTO Test_1 VALUES(1,'A');
INSERT INTO Test_1 VALUES(2,'B');
INSERT INTO Test_1 VALUES(3,'C');
INSERT INTO Test_1 VALUES(4,'D');

SELECT * FROM TEST_1;
```



A	B
1	A
2	B
3	C
4	D

DML - MERGE

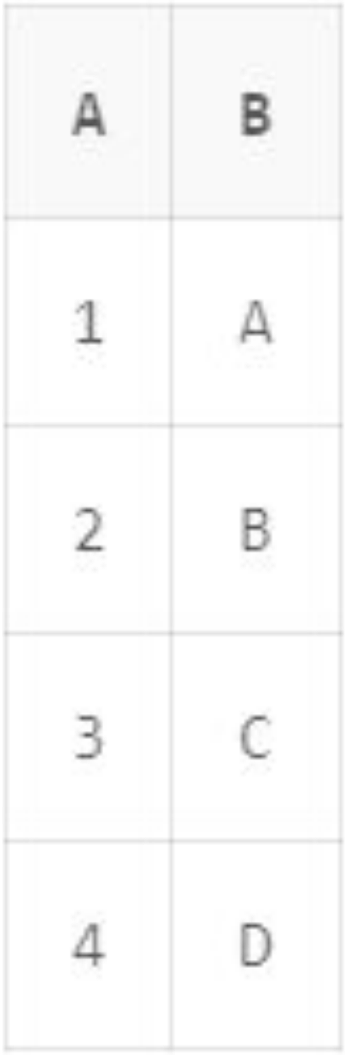
```
CREATE TABLE Test_2
(A number,
B varchar2(3)
);

INSERT INTO Test_2 VALUES(1,'X');
INSERT INTO Test_2 VALUES(2,'Y');
INSERT INTO Test_2 VALUES(4,'Z');

SELECT * FROM TEST_2;
```



A	B
1	X
2	Y
4	Z

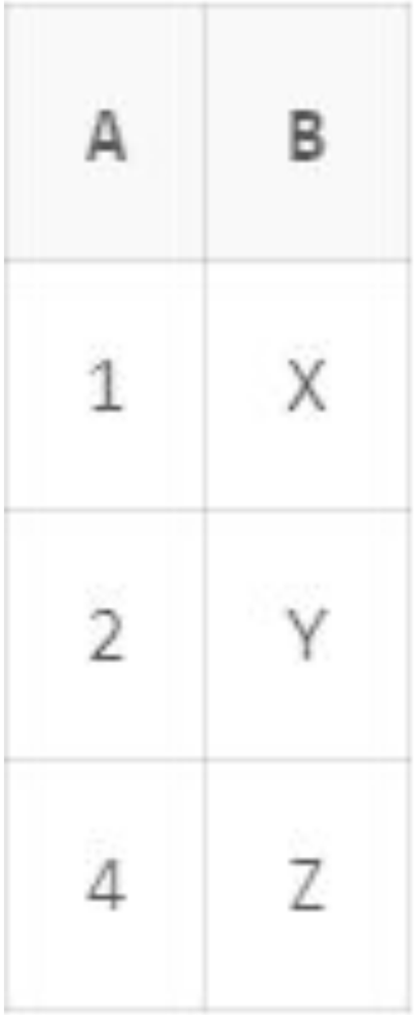
DML - MERGE

Test_1 (Source Table)



A	B
1	A
2	B
3	C
4	D

Test_2 (Target Table)

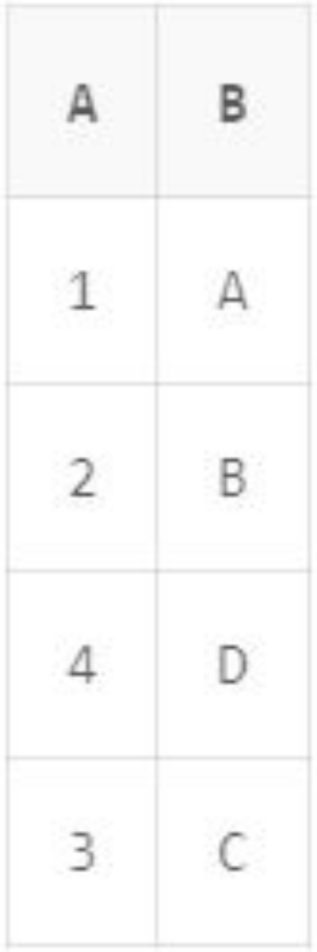


A	B
1	X
2	Y
4	Z

DML - MERGE

```
MERGE INTO Test_2 T2
USING Test_1 T1
ON ( T1.A = T2.A)

WHEN MATCHED THEN
UPDATE SET T2.B = T1.B

WHEN NOT MATCHED THEN
INSERT(T2.A,T2.B) VALUES (T1.A,T1.B);
```

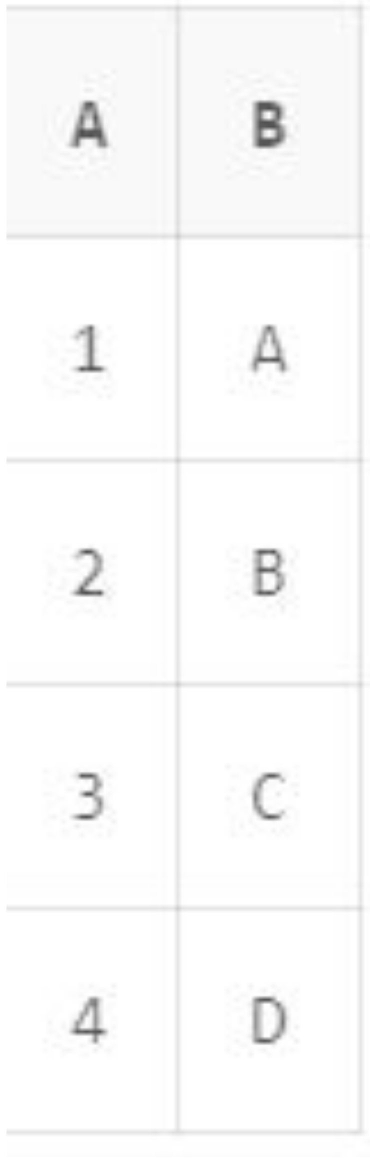
```
SELECT * FROM TEST_2;
```



A	B
1	A
2	B
4	D
3	C

ORDER BY - ASC

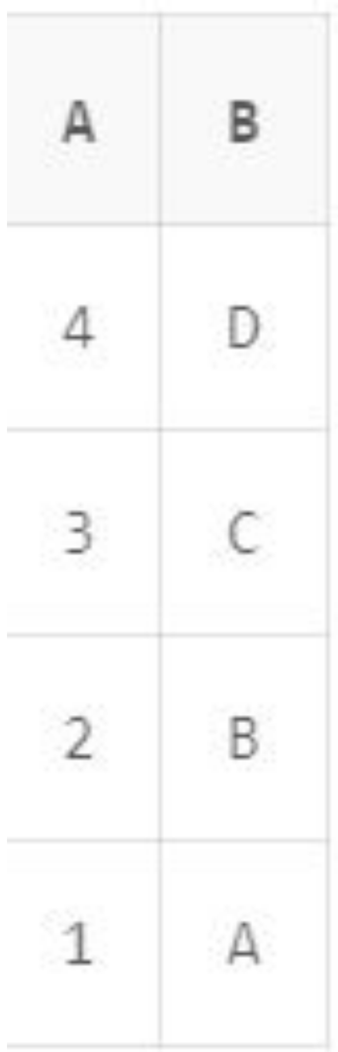
```
SELECT * FROM TEST_2 ORDER BY ASC;
```



A	B
1	A
2	B
3	C
4	D

ORDER BY - DESC

SELECT * FROM TEST_2 ORDER BY A DESC;



A	B
4	D
3	C
2	B
1	A

DML - DELETE

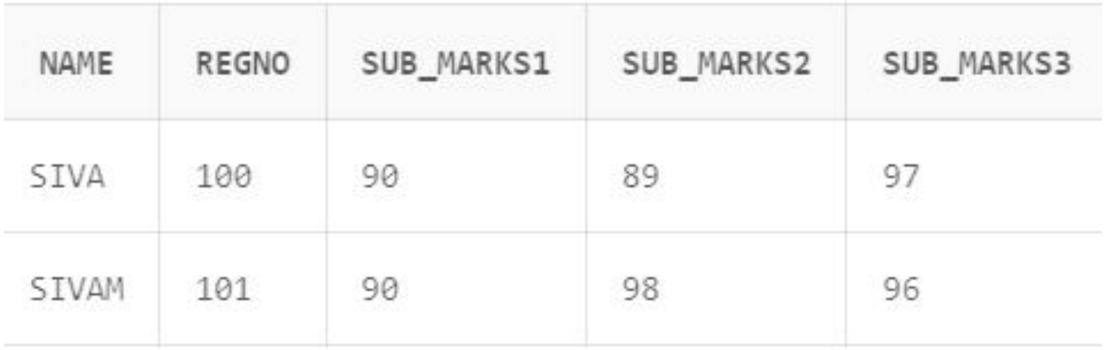
This command is used to DELETE existing records in a table/database.

Syntax :

```
DELETE FROM table_name
WHERE conditions;
```

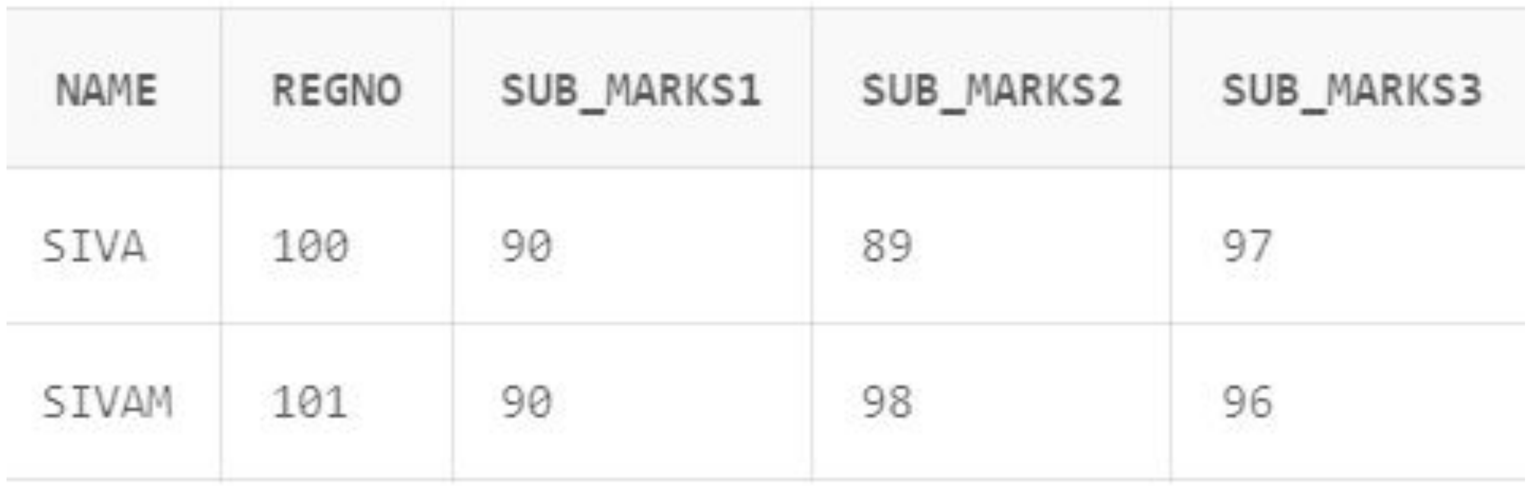
EXAMPLE :

```
INSERT INTO student (Name,Regno,Sub_Marks1, Sub_Marks2, Sub_Marks3)
VALUES ( 'SIVA', 100,90,89,97);
INSERT INTO student VALUES ( 'SIVAM', 101,90,98,96);
```



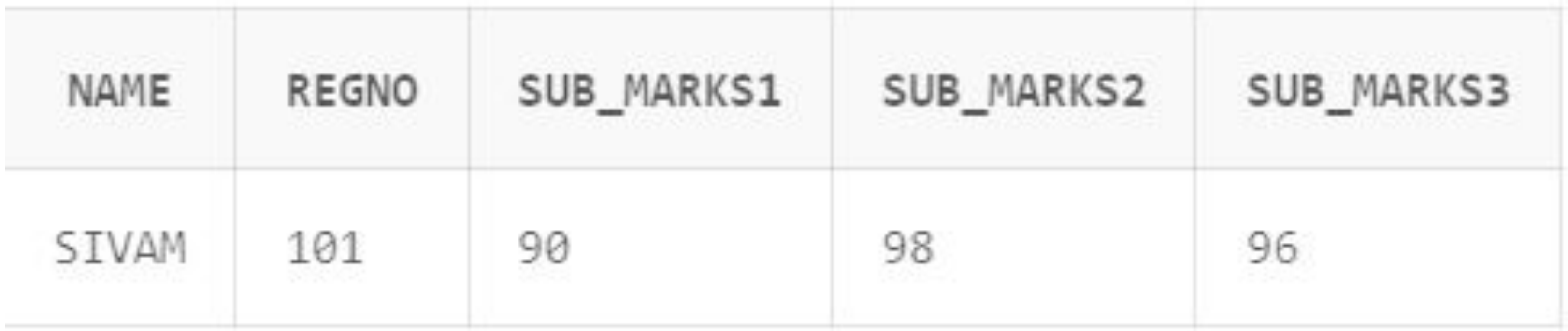
NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVA	100	90	89	97
SIVAM	101	90	98	96

DML - DELETE



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVA	100	90	89	97
SIVAM	101	90	98	96

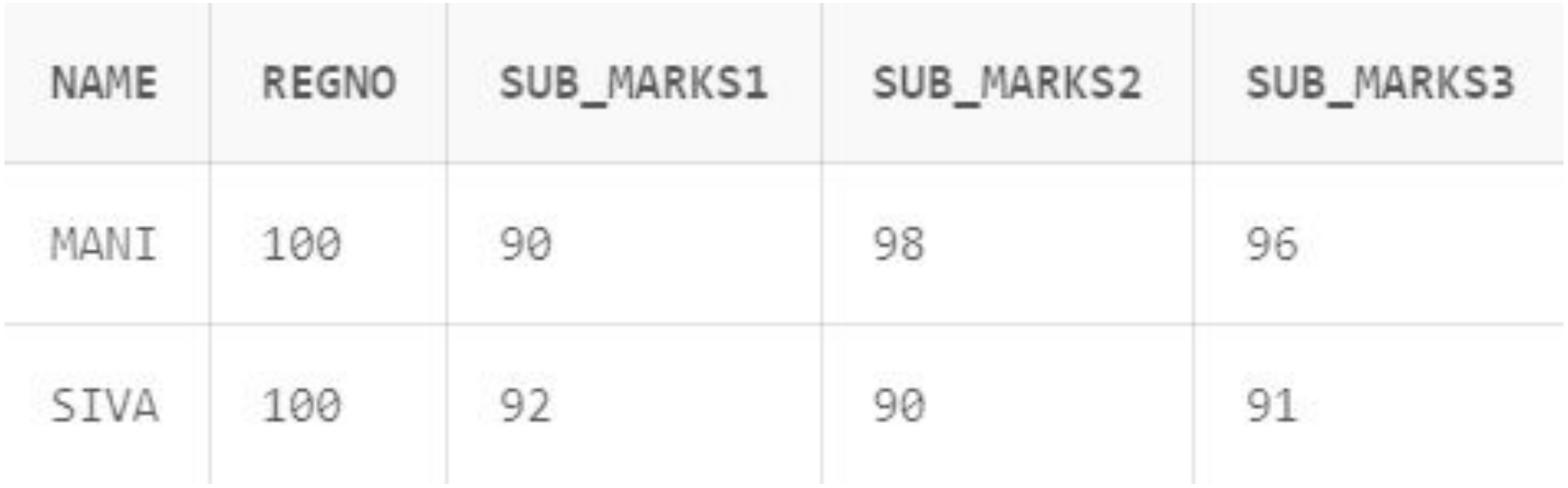
DELETE FROM student WHERE REGNO=100;



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
SIVAM	101	90	98	96

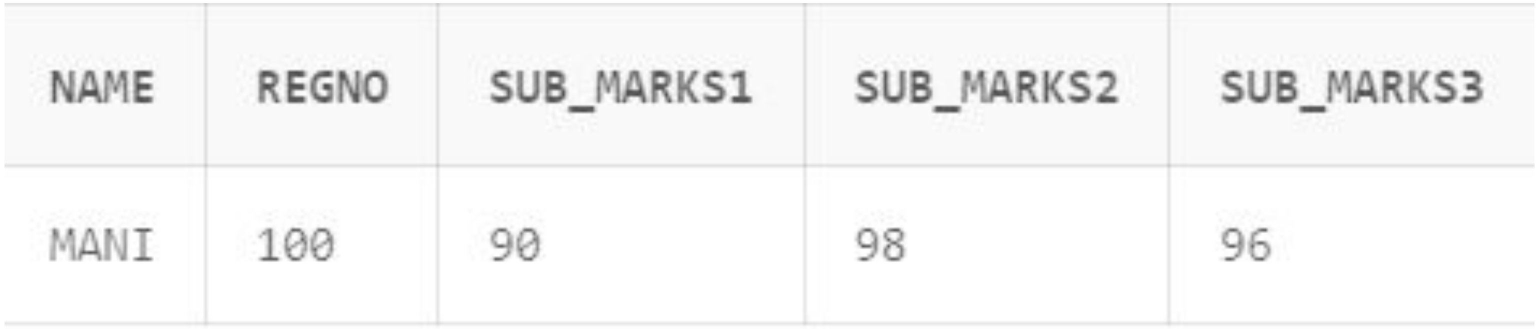
DML - DELETE

```
INSERT INTO student VALUES ( 'SIVA', 100,92,90,91);
INSERT INTO student VALUES ( 'MANI', 100,90,98,96);
```



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
MANI	100	90	98	96
SIVA	100	92	90	91

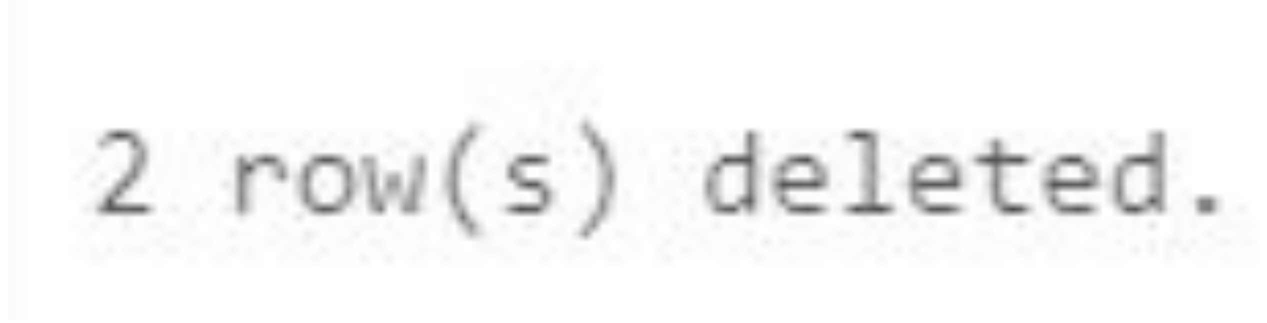
```
DELETE FROM student WHERE regno=100 AND name ='SIVA';
```



NAME	REGNO	SUB_MARKS1	SUB_MARKS2	SUB_MARKS3
MANI	100	90	98	96

DML - DELETE

```
DELETE FROM student;
```



```
2 row(s) deleted.
```

DDL – AUTO SAVE

DML – NO AUTO SAVE

COMMIT; (ie. SAVE)

CONSTRAINTS

CONSTRAINTS

CONSTRAINTS :

1. Constraints are used to restrict column data or to set rules defined in Oracle tables to ensure data integrity
2. These rules are enforced placed for each column or set of columns.
    1. Primary Key
    2. Unique Key
    3. NOT NULL
    4. Check
    5. Foreign Key

CONSTRAINTS

PRIMARY KEY

1. It accepts no duplicate and no null values
2. A table can have only one primary key
3. Oracle internally creates unique index to prevent duplication in the column values
4. Multiple columns can be clubbed under a composite primary key

Column level Constraint

```
CREATE TABLE table_name (
                              COLUMN[data type] CONSTRAINT <constraint_name> PRIMARY KEY,
                              COLUMN[data type];
CREATE TABLE table_name (
                              COLUMN[data type]  PRIMARY KEY,
                              COLUMN[data type];
```

Table level Constraint

```
CREATE TABLE table_name (
                              COLUMN [data type],
                              COLUMN [data type],
                               CONSTRAINT <constraint_name> PRIMARY KEY(ID);
```

CONSTRAINTS

Unique Key

It accepts no duplicate and accepts null values

Multiple Unique key can be defined per table

Composite columns key should be defined at the table level constraint

Column level Constraint

```
CREATE TABLE table_name (
                    COLUMN[data type] CONSTRAINT <constraint_name> UNIQUE;,
                    COLUMN[data type];
CREATE TABLE table_name (
                    COLUMN[data type] UNIQUE,
                    COLUMN[data type];
```

Table level Constraint

```
CREATE TABLE table_name (
                    COLUMN [data type],
                    COLUMN [data type],
                    COLUMN[data type] CONSTRAINT <constraint_name> UNIQUE;
```

CONSTRAINTS

```
create table student1 ( regno number constraint PK_1
primary key,
name varchar2(20));
insert into student1 values ('','asdfs');
select * from student;
select * from user_constraints;
```

WORKING WITH DATA MANIPULATION COMMANDS

Q1: Insert a single record into dept table.

Ans: SQL> insert into dept values (1,'IT','Tholudur');

WORKING WITH DATA MANIPULATION COMMANDS

Q2: Insert more than a record into emp table using a single insert command.

Ans: SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);

WORKING WITH DATA MANIPULATION COMMANDS

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as ASP

```
SQL> update emp set sal=15000 where job='ASP';
```

WORKING WITH DATA MANIPULATION COMMANDS

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into the table using select clauses

Ans: SQL> create table employee as select * from emp;

WORKING WITH DATA MANIPULATION COMMANDS

Q5: select employee name, job from the emp table

Ans: SQL> select ename, job from emp;

Data Independence

Data Independence

- Data independence is the ability to modify the scheme without affecting the programs and the application to be rewritten.
- Data is separated from the programs, so that the changes made to the data will not affect the program execution and the application.

Data Independence

- ✔ Data Independence is an important property of DBMS and also an advantage.
- ✔ There are three levels in database:
  1. Physical level / Low level (Disk storage)
  2. Conceptual level (query / procedure / logics / etc.,)
  3. Logical level / View level (User Interface)
- ✔ Data Independence is used to achieve the changes in physical level without affecting logical level and vice versa.
- ✔ There are two types of Data Independence in DBMS:
  1. Physical Data Independence
  2. Logical Data Independence

Data Independence

Need of Data Independence

- ✓ To improve the quality of data
- ✓ Easy maintenance of DBMS
- ✓ To achieve database security
- ✓ Developer need not be worry about internal structure
- ✓ Easily making the changes in physical level to improve the performance

Data Independence

Physical Data Independence

It is defined as to make the changes in the structure of the physical level /low level of DBMS without affecting the logical level / view level.

Some of the changes in Physical level

- ✔ Changing the storage devices
- ✔ Changing the file organization techniques
- ✔ Changing the data structures
- ✔ Changing the data access method
- ✔ Modifying indexes
- ✔ Migrating the Database from one drive to another

Data Independence

Logical Data Independence

It is defined as to make the changes in the structure of the logical level / view level of DBMS without affecting the physical / low level.

Some of the changes in Logical level

- ✓ Add a new attribute in an entity set
- ✓ Modify / Delete an attribute
- ✓ Merging records
- ✓ Splitting records

Data Independence

Difference between physical data independence and logical data independence

physical data independence	logical data independence
Concerned with the storage of the data.	Concerned with the structure of data definition.
Easy to retrieve	Difficult to retrieve because of dependent on logical structure
Easy to achieve, compare with logical data independence	Difficult to achieve, compare with physical data independence
Concerned with physical schema	Concerned with logical schema

Difference between Logical Data Independence and Physical Data Independence

Physical Data Independence	Logical Data Independence
Physical data independence is used to change the internal schema without requiring a change in the logical schema.	Logical data independence is making sure that if you add any new field or delete any existing field we do not need to change the application program.
Physical data independence is easy to attain in comparison to logical data independence.	It is difficult to attain logical data independence compared to physical data independence.
Physical data independence provides feasibility if we want to shift the database or want to change the file structure.	Logical data independence helps us to change the data definition and the structure of the data without having changes in the physical schema.
Physical data independence deals with the internal structure of the schema.	Logical data independence deals with conceptual schema.
Examples of changes in Physical independence are Changing the compression techniques, hashing algorithms, SSD, location of the database, etc.	Examples of changes in logical independence are Adding, deleting, or modifying the entity or relationship.

STRING FUNCTIONS

Function name	Description	Syntax	Example
LENGTH	Used to find the length of the arguments	LENGTH(argument)	SELECT first_name, LENGTH(first_name) FROM hr.employees;
REVERSE	Used to reverse the arguments	REVERSE(argument)	SELECT first_name, REVERSE(first_name) FROM hr.employees;

SUBSTR	Used to get the specific portion of the arguments	SURSTR(argument,start_position, no.of kength)	SELECT first_name, SUBSTR(first_name,1,4) FROM hr.employees;
			
			SELECT first_name, SUBSTR(first_name,-1,4) FROM hr.employees;

STRING FUNCTIONS

Function name	Description	Syntax	Example

INSTR	• Used to get the exact position of the arguments
	• Always returns a numeric value
	• If searching is not found then it will return 0
	• If string is NULL or searching string is NULL, then it will return NULL
		INSTR(argument, searching_string,start_position, nth_appearance)
		
		optional
			select first_name, INSTR(first_name,'e') from hr.employees;
			
			select first_name, INSTR(first_name,'e',1,1) from hr.employees;
			
			select first_name, INSTR(first_name,null,1,1) from hr.employees;

REPLACE	• Used to replace the characters with another set of characters	REPLACE(argument,replace_what,replace_with)	select first_name, REPLACE(first_name,'e','S') from hr.employees;
			
			select first_name, REPLACE(first_name,'en','SSS') from hr.employees;

STRING FUNCTIONS

Function name	Description	Syntax	Example
TRANSLATE	• Used to translate the characters with another set of characters	TRANSLATE (argument, translate_what, translate_with)	select translate ('SIVASI','SI','12') from dual;
LPAD	• Used to pads the left-side of a string with a specific set of characters	LPAD(argument,padded_length,padding_string)	select first_name, LPAD(first_name,15,'*') from hr.employees;
RPAD	• Used to pads the right-side of a string with a specific set of characters	RPAD(argument,padded_length,padding_string)	select first_name, RPAD(first_name,15,'*') from hr.employees;
LTRIM	• Used to trim a character from the left side of a string	LTRIM(argument,trimming_character)	select first_name, LTRIM(first_name,'S') from hr.employees;

STRING FUNCTIONS

Function name	Description	Syntax	Example
RTRIM	• Used to trim a character from the right side of a string	RTRIM(argument,trimming_character)	select first_name, RTRIM(first_name,'d') from hr.employees;

TRIM	• Used to trim a character from the string	TRIM((LEADING/TRAILING/BOTH) trimming_character FROM argument)	select first_name, TRIM(LEADING 'E‘ from first_name) from hr.employees;
			
			select first_name, TRIM(TRAILING 'E‘ from first_name) from hr.employees;
			
			select TRIM(BOTH 'M‘ from 'MADAM') from dual;

STRING FUNCTIONS

Function name	Description	Syntax	Example
UPPER	• Used to convert the set of characters into a upper case	UPPER(argument)	select first_name, upper(first_name) from hr.employees;
LOWER	• Used to convert the set of characters into a lower case	LOWER(argument)	select first_name, lower(first_name) from hr.employees;
INITCAP	• Used to convert the initial character to upper in the set of characters	INITCP(argument)	select INITCAP('madam') from dual;

DATE FUNCTIONS

Function name	Description	Syntax	Example
SYSDATE	• Used to display current system date	SYSDATE	SELECT SYSDATE from dual;

ADD_MONTHS	• Used to add number of month(n)	ADD_MONTHS(data_expression,Month)	SELECT ADD_MONTHS(SYSDATE, 2) from dual;
			
			select add_months(sysdate,-2) from dual;
			
			SELECT ADD_MONTHS(DATE '2023-2-14',2) from dual;

DATE FUNCTIONS

Function name	Description	Syntax	Example

MONTHS_BETWEEN	• Used to get the number of months between two dates	MONTHS_BETWEEN(start_date, End_date)	select MONTHS_BETWEEN(DATE '2020-2-14', DATE '2023-2-14') from dual;
			
			select ABS (MONTHS_BETWEEN(DATE '2020-2-14', DATE '2023-2-14')) from dual;

NEXT_DAY	• Used to get the date of the weekday specified by the day name	NEXT_DAY(date,weekday)	select NEXT_DAY(SYSDATE,'SUNDAY') from dual;
			
			select NEXT_DAY(DATE '2023-2-5','SUNDAY') from dual;

DATE FUNCTIONS

Function name	Description	Syntax	Example
LAST_DAY	• Used to get the last day of the month	LAST_DAY(date)	select LAST_DAY(DATE '2023-3-5') from dual;

ROUND	• Used to round the date to a specific unit	ROUND(date,format)	select ROUND(DATE '2023-1-17',‘DAY') from dual;
			
			select ROUND(DATE '2023-2-17','MONTH') from dual;

TRUNC	• Used to truncate the date to a specific unit	TRUNC(date,format)	select trunc(DATE '2022-1-10','DAY') from dual;
			
			select trunc(DATE '2022-2-10','MONTH') from dual;
			
			select trunc(DATE

STRING FUNCTIONS

- select * from hr.employees;
- select first_name, upper(first_name) from hr.employees;
- select first_name, reverse(first_name) from hr.employees;
- select first_name, substr(first_name,1,3) from hr.employees;
- select first_name, substr(first_name,-1,1) from hr.employees;
- select first_name, replace(first_name,1,3) from hr.employees;
- select first_name, INSTR(first_name,'e') from hr.employees;
- select first_name, INSTR(first_name,'e',1,2) from hr.employees;
- select first_name, INSTR(first_name,null,1,1) from hr.employees;
- select first_name, REPLACE(first_name,'en','SSS') from hr.employees;
- select translate ('SIVASI','SI','12') from dual;
- select first_name, LPAD(first_name,15,'*') from hr.employees;
- select first_name, RTRIM(first_name,'d') from hr.employees;
- select first_name, TRIM(LEADING'E'from first_name) from hr.employees;
- select TRIM(BOTH 'M'from 'MADAM') from dual;
- select first_name, upper(first_name) from hr.employees;
- select INITCAP('madam') from dual;

STRING FUNCTIONS

- select sysdate from dual;
- select add_months(sysdate,2) from dual;
- select add_months(sysdate,-2) from dual;
- select add_months(DATE '2023-2-4',2) from dual;
- select MONTHS_BETWEEN(DATE '2020-2-14', DATE '2023-2-14') from dual;
- select ABS (MONTHS_BETWEEN(DATE '2020-2-14', DATE '2023-2-14')) from dual;
- select NEXT_DAY(SYSDATE,'SUNDAY') from dual;
- select NEXT_DAY(DATE '2023-2-5','SUNDAY') from dual;
- select LAST_DAY(DATE '2023-3-5') from dual;
- select ROUND(DATE '2023-2-17','MONTH') from dual;
- select ROUND(DATE '2023-1-19','DAY') from dual;
- select trunc(DATE '2022-1-10','DAY') from dual;
- select trunc(DATE '2022-2-10','MONTH') from dual;
- select trunc(DATE '2022-2-1','YEAR') from dual;

DCL COMMANDS

GRANT

REVOKE

How to create the user in database?

CREATE USER<user_name> IDENTIFIED BY<password>; To create a user

GRANT CONNECT, RESOURCE TO<user_name>; - To Grant privilege to the user

GRANT SELECTION ON<table_name> TO <user_name>; - To Grant select privilege to the user

REVOKE SELECT ON<table_name> FROM (user_name); - To revoke select privilege from the user

REVOKE ALL ON<table_name> FROM (user_name); - To revoke all privilege from the user

DCL COMMANDS

PUBLIC

GRANT SELECTION ON<table_name> TO PUBLIC; - To Grant select privilege to all user

REVOKE ALL ON<table_name> FROM PUBLIC; - To revoke all privilege from all user

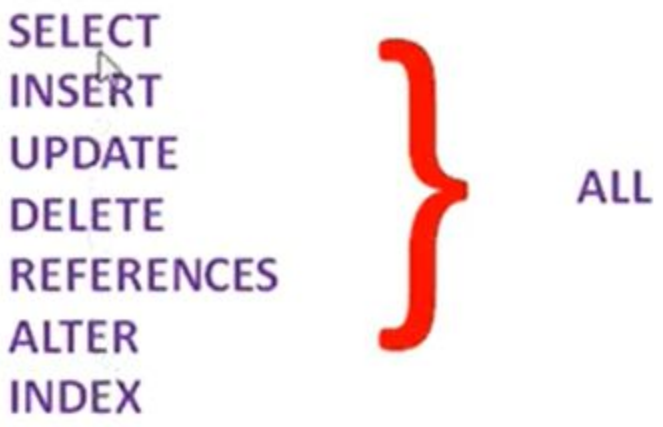
DCL COMMANDS

ROLE (GROUP)

CREATE ROLE <role_name>; To create a role

GRANT <role_name> TO <user_1>,<user2>….; - To add users to role

REVOKE <role_name> FROM <user_1>; - To remove users from the role



$$\left.\begin{array}{l}\text{SELECT}\\\text{INSERT}\\\text{UPDATE}\\\text{DELETE}\\\text{REFERENCES}\\\text{ALTER}\\\text{INDEX}\end{array}\right\}\quad\text{ALL}$$

DCL COMMANDS

```
SELECT USER FROM DUAL;
SELECT * FROM ALL_USERS;
CREATE USER SIVA IDENTIFIED BY SIVAS;
GRANT CONNECT, RESOURCE TO AB;
SELECT * FROM PRODUCTS;
GRANT SELECT ON PRODUCTS TO AB;
GRANT INSERT ON PRODUCTS TO AB;
GRANT SELECT,INSERT,UPDATE ON PRODUCTS TO AB;
REVOKE SELECT ON PRODUCTS FROM AB;
REVOKE ALL ON PRODUCTS FROM AB;
```

```
SELECT * FROM ALL_TABLES WHERE OWNER = ‘AB’;
SELECT * FROM HR.PRODUCTS;
INSERT INTO HR.PRODUCTS VALUES(23,23,45);
```

THE EVOLUTION OF DATA MODELS

THE EVOLUTION OF DATA MODELS

To come across the limitations of file systems, there are lot of researchers and software developers designed and developed various data models.

The important and widely accepted models are:

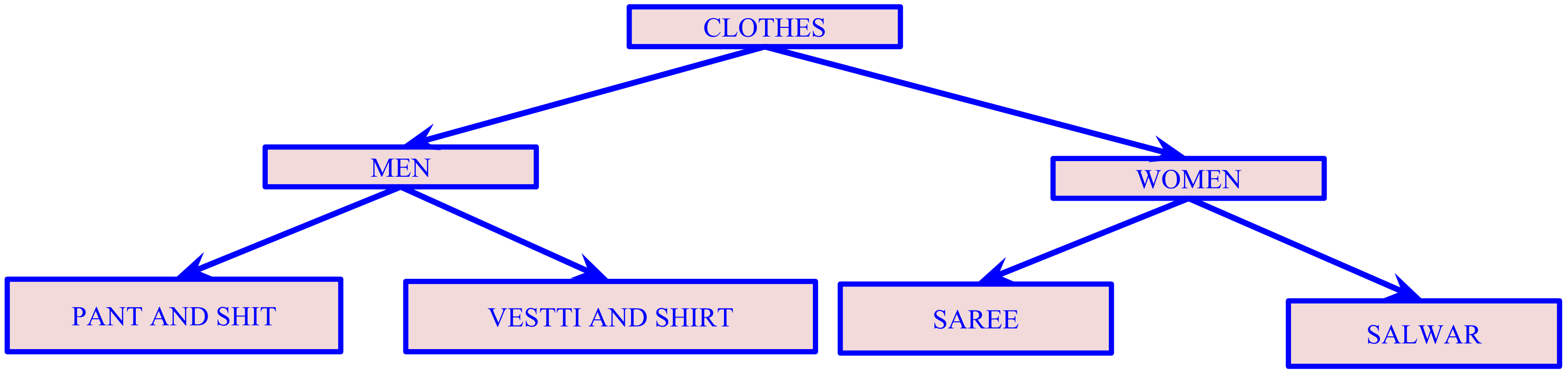
- ✔ Hierarchical
- ✔ Network
- ✔ Entity relationship
- ✔ Relational
- ✔ Object oriented

THE EVOLUTION OF DATA MODELS

Hierarchical Model

- ✔ The first and fore most model of the DBMS.
- ✔ This model organizes the data in the hierarchical tree structure.
- ✔ This model is easy to understand with real time examples site map of a website

Example : For example the following is the representation of relationships present on online clothes shopping



THE EVOLUTION OF DATA MODELS

Features of a Hierarchical Model

- ✓ One-to-many relationship:
- ✓ Parent-Child Relationship
- ✓ Deletion Problem:
- ✓ Pointers

Advantages of Hierarchical Model

- ✓ Simple and fast traversal because of using tree structure
- ✓ Changes in parent node automatically reflected in child node

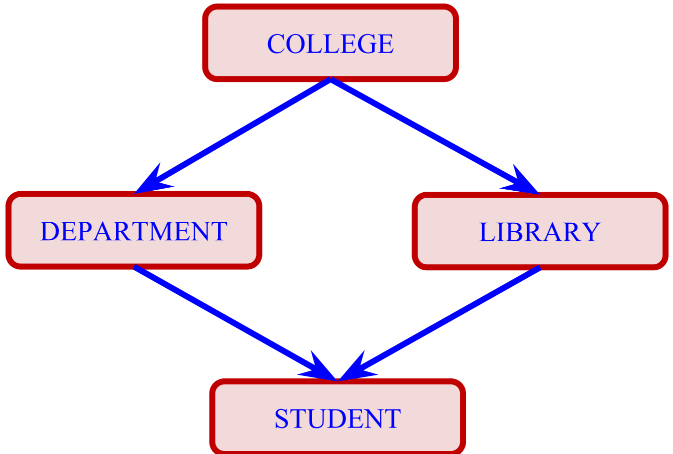
Disadvantages of Hierarchical Model

- ✓ Complexity
- ✓ Parent mode deleted automatically child node will be deleted

THE EVOLUTION OF DATA MODELS

Network Model

- ✔ Network model is an extension of hierarchical model.
- ✔ This model was recommended as the best before relationship model.
- ✔ Same like hierarchical model, the only difference between these two models are a record can have more than one parent
- ✔ For Example consider the following diagram a student entity has more than one parent



THE EVOLUTION OF DATA MODELS

Features of a Network Model

- ✔ Manage to Merge more Relationships
- ✔ More paths
- ✔ Circular Linked List

Advantages of Network Model

- ✔ Data access is faster
- ✔ Because of parent child relationship , the changes in parent reflect in child

Disadvantages of Network Model

- ✔ More complex because of more and more relations

THE EVOLUTION OF DATA MODELS

Entity-Relationship Model (ER Model)

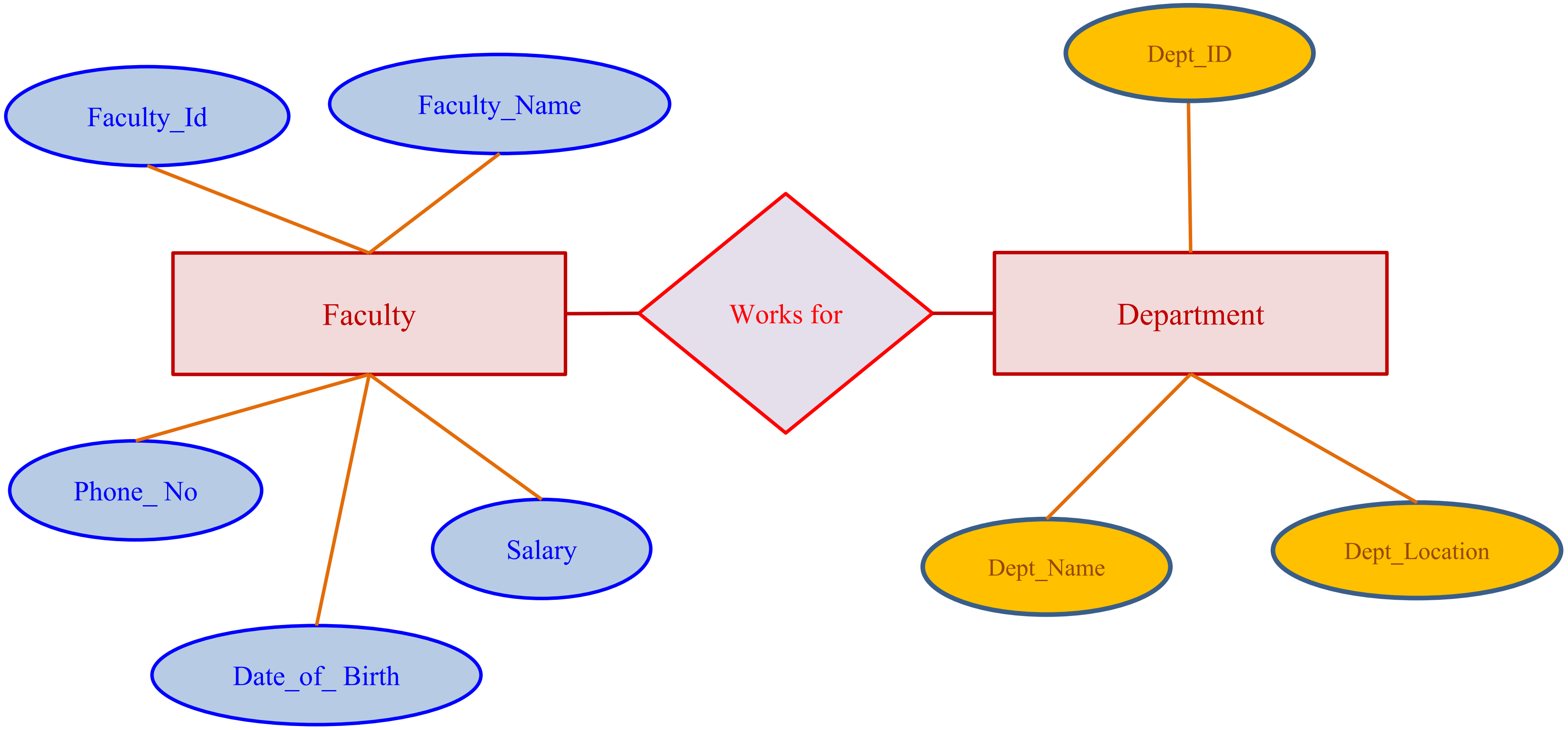
- ✓ This model is a high level data model
- ✓ Represents the real – world problem as a pictorial representation
- ✓ Easy to understand by the developers about the specification
- ✓ It is like a visualization tool to represent a specific database

It contains three components

1. Entities
2. Attributes
3. Relationships

THE EVOLUTION OF DATA MODELS

Example for ER Diagram (Faculty and Department entity set)



THE EVOLUTION OF DATA MODELS

In the above example:

- ✔ There are two entities , Faculty and Department
- ✔ The attributes of Faculty entities are
  - Faculty_Id
  - Faculty_Name
  - Phone_No
  - Date_of_birth
  - Salary
- ✔ The attributes of Department entities are
  - Dept_ID
  - Dept_Name
  - Dept_Location
- ✔ Relationship : Faculty works for a department

THE EVOLUTION OF DATA MODELS

Features of ER Model

- ✔ Graphical representation
- ✔ Visualization
- ✔ Good Database design (Widely used)

Advantages of ER Model

- ✔ Very Simple
- ✔ Better communication
- ✔ Easy to convert to any model

Disadvantage of ER Model

- ✔ No industry standard
- ✔ Hidden information

THE EVOLUTION OF DATA MODELS

Relational Model

- ✔ Widely used model
- ✔ Data are represented as row-wise and column-wise (2 Dimensional Array)

Example : EMP (Employee) Table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	-	20
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	-	10

THE EVOLUTION OF DATA MODELS

Relational Model

- ✔ Each row is known as RECORD or TUPLE
- ✔ Each Column is known as ATTRIBUTE or FILED
- ✔ The collection of attributes are called as record – An Entity
- ✔ The collection of records are called as Table – Entity Set
- ✔ In the above example:

  Table – EMP

  Attributes – Empno, Ename, Sal,….

THE EVOLUTION OF DATA MODELS

Features of Relational Model

- ✓ Records
- ✓ Attributes

Advantages of Relational Model

- ✓ Simple
- ✓ Scalable
- ✓ Structured format
- ✓ Isolation

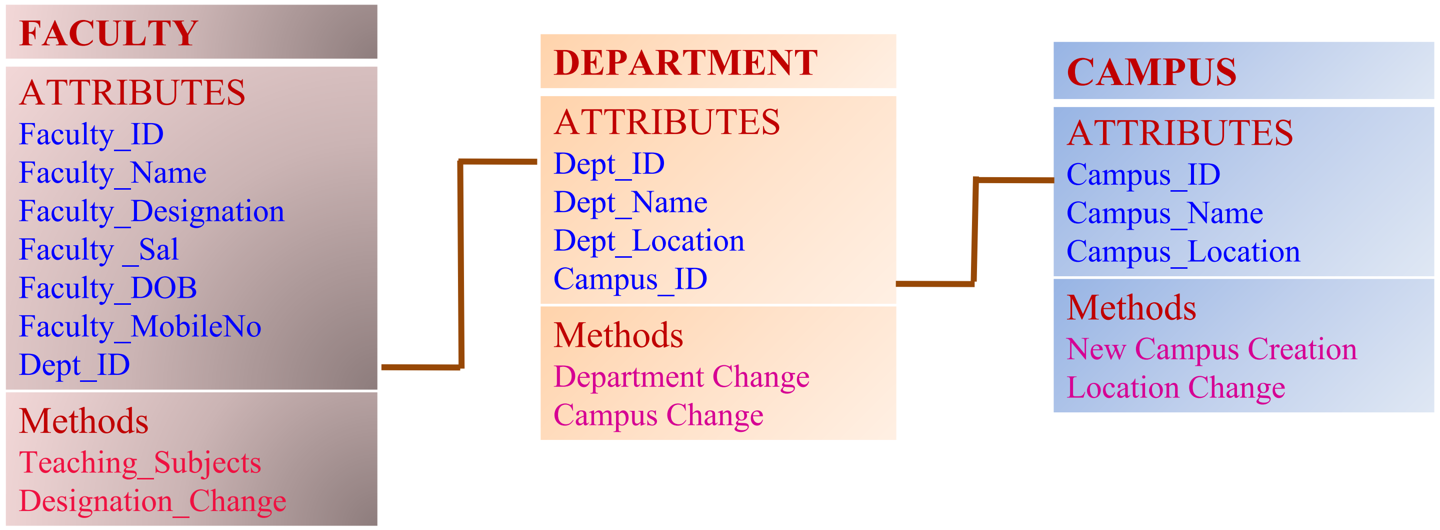
Disadvantages of Relational Model

- ✓ Hardware overheads

THE EVOLUTION OF DATA MODELS

Object Oriented Model

- ✔ The real- time problems are easily represented through object-oriented data model which is an OBJECT.
- ✔ In this Model, the data and its relationship present in the single structure
- ✔ Complex data like images, audio, videos can be stored easily
- ✔ Objects connected through links using common attribute(s)
- ✔ Example : Three Objects Faculty, Department and Campus linked using common attribute



Lab 2: SQL Data Manipulation Language Commands

Consider EMP table for DML operations

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	-	20
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	-	10

Lab 2: SQL Data Manipulation Language Commands

DML Commands are relates only with base table information (value in an attribute)

There are four commands in DML:

1. **INSERT**
2. **UPDATE**
3. **DELETE**
4. **SELECT**
   - ✔ WHERE clause (Conditional retrieval)
   - ✔ ORDER BY clause (Retrieval in Ascending or Descending Order)
   - ✔ GROUP BY clause (Retrieval of distinct values by considering groups)
   - ✔ HAVING clause (Followed by Group by clause with COUNT function)

Lab 2: SQL Data Manipulation Language Commands

INSERT COMMAND

- ✓ It relates only with new records.
- ✓ Only one row can be inserted at a time
- ✓ Multiple rows can be inserted using “&” symbol one by one
- ✓ Can insert to entire table
- ✓ Can insert in selected columns with some restrictions
- ✓ Must follow the order of the column specified in the query statement

Lab 2: SQL Data Manipulation Language Commands

INSERT COMMAND

Syntax:

```
INSERT INTO <table_name> (column_name1 <datatype>,
                          column_name2 <datatype>,
                          . . . ,

                          column_name_n <datatype>)
                    VALUES
                          (value1,
                           value2,
                           . . . ,
                          value n);
```

Note :

- **Number values can be inserted as integer or float**
- **Char and Date values must be in single quote**

Lab 2: SQL Data Manipulation Language Commands

INSERT COMMAND

Example 1: To insert a record using all fields in EMP table

```
INSERT INTO EMP VALUES (7369, 'SMITH',  'CLERK', 7902, '17-12-1980‘, 800,
    NULL, 20);
```

(OR)

```
INSERT INTO EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
VALUES (7369, 'SMITH',  'CLERK', 7902, '17-JUNE-1980',  800, NULL, 20);
```

Example 2: To insert a record using selected fields in EMP table

```
INSERT INTO EMP (EMPNO, ENAME) VALUES (7499, 'ALLEN);
```

Note : When a record is inserted using selected fields, it must include NOT NULL and Primary key fields.

Lab 2: SQL Data Manipulation Language Commands

Example 3: To insert multiple records using all fields in EMP table

```
INSERT INTO EMP values
(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&
    DEPTNO) ;
```

NOTE : '&' (Ampersand) symbol used to ask

Enter value for followed by the string during runtime.

The input value will be store in the appropriate field using

bind variable (:OLD and :NEW)

Lab 2: SQL Data Manipulation Language Commands

UPDATE COMMAND

- ✔ It works with only existing records
- ✔ It works only column wise
- ✔ It is used to modify the column values (increase / decrease / change)

Syntax

UPDATE <table_name> set <field_name> = value [where <condition>];

Note : Update command without where condition will update all the records.

Update command with where condition will update the records which are satisfy the condition

Example 1:

UPDATE emp set comm = 2000 ; (Update all the records in EMP table)

Example 2 :

Update emp set comm = 1000 where empno = 7369;

(Update the records having the empno as 7369)

Lab 2: SQL Data Manipulation Language Commands

Delete command

- ✔ It works only with existing records
- ✔ It works only with row wise
- ✔ It not possible to delete a single column in a row

Syntax

DELETE from <table_name> [where <condition>];

Note :

- Delete command with out where condition will delete all the records in the table.
- Delete command with where condition will delete the selected records which are satisfy the condition.

Example 1: DELETE from emp; (All records will be deleted from emp)

Example 2: DELETE from emp where empno = 7369;

(Those records holding the value in the field empno as 7369 will be deleted)

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

- ✔ Works with existing records
- ✔ Works with row wise and column wise
- ✔ Works with multiple tables
- ✔ Never affect / change / update / modification in the data base
- ✔ Using this command , we can select a column , multiple columns, all columns, single row, multiple row, all rows
- ✔ Specially called as “QUERY STATEMENT”

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

Syntax

```
SELECT column_list FROM table-name
                [WHERE Clause]
                [GROUP BY clause]
                [HAVING clause]
                [ORDER BY clause];
```

NOTE : To retrieve all the column from the table ‘ * ’ symbol can be used instead of specifying the column_list.

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

Example 1: To retrieve all the columns and rows from emp table

SELECT * from emp; (‘*’ stands from all columns and rows)

Example 2: To select retrieve the specific columns from all rows

SELECT empno,ename from emp;

Select command with where clause

Example 3: To retrieve the records from emp table which record holds the salary value greater than 1000;

SELECT * from emp WHERE sal> 1000;

Example 4: To retrieve the columns empno and ename from emp table which records holds the value as CLERK in job column.

SELECT empno, ename from emp WHERE job = ‘CLERK’

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

Select command with order by clause

Example 5 : To retrieve the records from emp table in ascending order using empno

```
SELECT * from emp order by empno asc;
```

(OR)

```
SELECT * from emp order by empno;
```

Example 6: To retrieve the records from emp table in ascending order using job and empno

```
SELECT * from emp order by job,empno asc;
```

(OR)

```
SELECT * from emp order by job,empno;
```

NOTE : Ascending order is default condition, no need to specify

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

Select command with order by clause

Example 7 : To retrieve the records from emp table in descending order using empno.

```
SELECT * from emp order by empno desc;
```

Example 8: To retrieve the records from emp table in descending order using job and empno

```
SELECT * from emp order by job desc,empno desc;
```

Example 8: To retrieve the records from emp table in ascending order using job and descending order empno

```
SELECT * from emp order by job asc,empno desc;
```

Lab 2: SQL Data Manipulation Language Commands

SELECT COMMAND

Select command with group by clause

Example 9: To retrieve the different jobs from emp table

```
SELECT job from emp group by job;
```

Example 10: To retrieve the different jobs and its average salary from emp table

```
SELECT job, avg(sal) from emp group by job;
```

Select command with group by and having clause

Example 11: To retrieve the different jobs from emp table where the total numbers in a group is greater than 2;

```
SELECT job from emp group by job having count(job) >2;
```

NOTE : Count is built-in group function

DEGREES OF DATA ABSTRACTION

- ✔ Data abstraction is the idea that a database design begins with a high level view and as it approaches implementation level, the level of detail increases.

- ✔ In 1970, the American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) established a framework for database design based on the degrees of abstraction.

- ✔ The ANSI/SPARC architecture is composed of four levels of data abstraction; these levels are external, conceptual, internal, and physical

Reference : https://databasemanagement.fandom.com/wiki/Degrees_of_Abstraction

DEGREES OF DATA ABSTRACTION

- ✔ The External Model is the end users' view of the data. The end users view of data usually applies to their specific business needs and those of their organizational unit.

- ✔ The Conceptual Model is the database as seen by the specific DBMS. What sets the internal model apart from the external and conceptual is its reliance on its software platform.

- ✔ The goal in designing the internal model is to achieve logical independence, where the internal model can be changed without affecting conceptual model.

DEGREES OF DATA ABSTRACTION

- ✔ The Physical Model is the final and lowest level of abstraction. This is the model which describe such implementation level design as how the data is stored on media and what media to use. This level of abstraction is reliant on software and hardware.

  Note:

  - If the rules established by the ANSI/SPARC are followed, the database is easily scalable and upgradeable.
  - A common need is for the ease of upgradability in the physical model.
  - As technology improves and as the database grows and needs more processing power and space it is important to be able to upgrade the hardware without worrying about needing to redesign parts or the entire database.

DATABASE USERS AND DBA

Database Users

- ✔ Naive Users
- ✔ Application Programmers
- ✔ Sophisticated Users
- ✔ Native Users
- ✔ Specialized Users
- ✔ Stand-alone Users

DATABASE USERS AND DBA

Naive Users

- ✔ Those who don’t have any knowledge about DBMS
- ✔ Use DBMS applications frequently
- ✔ Mostly using the internet browser as an interface to access the database
- ✔ They don’t have any privileges to modify the database, simply use the application
- ✔ **Example :** Railway booking users, Clerks in bank accessing database

Application Programmers

- ✔ Users who develop DBMS applications.
- ✔ They are backend programmers
- ✔ Programs can be written in any programming languages like C++, JAVA, Python, PHP

DATABASE USERS AND DBA

Sophisticated Users

- ✓ Having knowledge about database and DBMS
- ✓ They can create their own applications based on requirements
- ✓ They don't write codes in any programming languages, but able to manage using queries
- ✓ **Example :** Business Analyst, Researchers

Native Users

- ✓ These are the users, who use the existing database applications
- ✓ They don't write any codes or queries
- ✓ **Example:** Library Management Systems, Inventory Control Systems

DATABASE USERS AND DBA

Specialized Users

- ✔ These are also sophisticated users, but they write special database application programs.
- ✔ They are the developers who develop the complex programs to the requirement.

Stand-alone Users

- ✔ These users will have a stand-alone database for their personal use.
- ✔ These kinds of the database will have readymade database packages which will have menus and graphical interfaces.

DATABASE USERS AND DBA

Database Administrator (DBA)

- ✓ DBA is a person or a group who define and manage the database in all three levels.
- ✓ DBA can create / modify /remove the users based on the requirements.
- ✓ DBA is the super user having all the privileges of DBMS

Responsibilities of DBA

- ✓ Install the Database
- ✓ Upgrade the Database
- ✓ Design and Implementation
- ✓ Database tuning
- ✓ Migrating the Database
- ✓ User Management
- ✓ Backup and Recovery
- ✓ Security of the Database in all access points
- ✓ Documentation

DATABASE LANGUAGES

The common language is Structured Query Language

It is categorized into three types based on operations

- ✔ Data Definition Language (DDL) – To specify the database schema
- ✔ Data Manipulation Language (DML) – To express the database queries and updates.
- ✔ Data Control Language (DCL) - To manage the database operations

DATABASE LANGUAGES

Data Definition Language (DDL)

- ✔ Can specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language.
- ✔ The values stored in a database must satisfy certain constraints to maintain consistency and reliability

DATABASE LANGUAGES

Data Definition Language (DDL)

Different types of constraints

- ✔ Domain Constraints
  - Data types , Not Null, Check ,Unique, Primary key
- ✔ Referential Integrity constraints
  - Foreign key
- ✔ Assertions
  - Any condition that the database must satisfy.
  - Domain and Referential Integrity constraints are special forms of assertion.
- ✔ Authorization
  - User Authorization
  - Read Authorization
  - Insert Authorization
  - Update Authorization
  - Delete Authorization

Note : The output of the DDL will be stored in Data Dictionary which contains all the details about the data, like meta-data

DATABASE LANGUAGES

Data Manipulation Language

- ✔ Enables users to access or manipulate data as organized by theappro priate data model.
- ✔ The followings are the different types of access
  - – Retrieval of information stored in the database (SELECT)
  - – Insertion of new information into the database (INSERT)
  - – Deletion of information from the database (DELETE)
  - – Modification of information stored in the database (UPDATE)

✔There are basically two types:

- Procedural DMLs require a user to specify what data are needed and how to get those data.
- Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.

S-14-15SLO-1 & SLO-2 : Lab 3: SQL Data Control Language commands and Transaction control commands to the sample exercises

Data Control Languages

✔Used to give / get back / control the privileges of an object by the owner

GRANT : To give access privileges of an object to other user by the owner

Syntax : GRANT [ALL / INSERT /UPDATE /DELETE /SELECT]

on <OBJECT_NAME> to <USER_NAME>;

Example: GRANT all on emp to scott;

REVOKE : To get back all the privileges from the user who has been granted

Syntax : REVOKE [ALL / INSERT /UPDATE /DELETE /SELECT]

on <OBJECT_NAME> from <USER_NAME>;

Example: REVOKE all on emp from scott;

S-14-15SLO-1 & SLO-2 : Lab 3: SQL Data Control Language commands and Transaction control commands to the sample exercises

Transaction Control Language

✔ To control the database operation

- COMMIT: Commits a Transaction. Save the changes permanently , can't rollback
- ROLLBACK: Rollbacks a transaction in case of any error occurs.
- SAVEPOINT: Sets a savepoint within a transaction. Rolled back from the specified savepoint