

# DBMS Lab 1.0

# RDBMS

- Relational database is a collection of related information that has been organized into tables. Each table contains rows and columns.
- The tables are stored in the database in structures known as schemas.
- Each row is called an entity, thus table as entity set.
- Row is known as Tuple. Columns are the properties called Attributes.
- The facts describing an entity are known as data.
- For an attribute, the set of permitted values is called domain of that attribute.

| 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-NOV-81 | 3000 |      | 20     |
| 7839  | KING   | PRESIDENT |      | 17-NOV-81 | 5000 |      | 10     |
| 7844  | TURNER | SALESMAN  | 7698 | 08-SEP-81 | 1500 | 0    | 30     |
| 7876  | ADAMS  | CLERK     | 7788 | 23-SEP-81 | 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     |

**Employee Table**

## Examples of RDBMS

- Oracle
- DB2
- Microsoft SQL-Server
- MySQL
- Microsoft Access
- Apache Derby
- Visual FoxPro
- OpenBase
- PostgreSQL
- SQLite
- Vertica
- IBM Informix
- Ingres
- IBM Lotus
- SQL Anywhere

# Client/Server Database

- Client/Server databases run the DBMS as a process on the server and run a client database application on each client.
- The client application sends a request for data over the network to the server.
- When the server receives the client request, the DBMS retrieves data from the database, performs the required processing on the data, and sends only the requested data back to the client over the network.

# SQL

- ❖ SQL stands for Structured Query Language
- ❖ SQL lets you access and manipulate databases
- ❖ SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

## What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

# Naming Conventions

- ☐ A table is an object that can store data in a database
- ☐ When you create a table, you must specify the table name, name of each column, data type of each column, and size of each column
- ☐ The table and column names can be up to 30 characters long
- ☐ Table or column name must begin with a letter
- ☐ The names are not case sensitive
- ☐ Spaces and hyphens are not allowed in a table or a column name; but \$, \_ and # are allowed



# Data Types



*Data type specifies the type of data that will be stored in the column. Data types also help to optimize storage space.*

## CHAR(n)

- Stores fixed-length alphanumeric data in a column
- Default and minimum size is one character
- Maximum allowable size is 2000 characters (previously 255)
- If a string of a smaller length is stored, it is padded with spaces at the end

## VARCHAR(n)/VARCHAR2(n)

- Stores variable-length alphanumeric data in a column
- Default and minimum size is one character
- Maximum allowable size is 4000 characters (previously 2000)
- If the data are smaller than the specified size, only the data value is stored; no padding is done

## DATE

- Stores date and time values
- The range of allowable dates is between January 1, 4712B.C. and December 31, 9999A.D.
- The default date format is DD-MON-YY. The DD-MON-YYYY format also works

## NUMBER(precision, scale)

- Stores floating point numbers as well as integer numbers. Precision is the total number of significant digits in the number; scale is the total number of digits to the right of the decimal point(if used). The precision can range from 1 to 38
- If neither precision nor scale is specified, any number may be stored up to a precision of 38 digits

## INTEGER

- Stores integer number

## NUMERIC(p,d)

- Stores fixed-point number with user specified precision
- Similar to NUMBER data type

## LONG

- Stores variable length character strings containing up to 2GB
- Similar to VARCHAR
- There can be one LONG data type per table

## RAW

- Stores binary data such as digitized picture or image
- Maximum allowable size is 2000 Bytes (previously 255 Bytes)

## LONG RAW

- It is the higher range of RAW
- There can be one LONG data type per table
- Maximum allowable size is 2GB

## LOB(Large Object)

Stores large volume of data

### BLOB

Used for binary data such as graphics, video clips and audio files up to 4GB

### CLOB

Used for character data up to 4GB

### BFILE

Stores references to a binary file that is external to the database and is maintained by the operating system's file system



# SQL Statements

## Data Definition Language(DDL)

Defines the data structure that make up a database

- **CREATE** statement
- **ALTER** statement
- **DROP** statement
- **RENAME** statement
- **TRUNCATE** statement

## Data Manipulation Language(DML)

Modifies the contents of tables

- **INSERT** statement
- **UPDATE** statement
- **DELETE** statement

## Query Statement

Retrieves data from the database

- **SELECT** statement

## Transaction Control Language(TCL)

Permanently records the changes made to the rows stored in a table or undoes those changes affected by DML statements

- **COMMIT** statement
- **ROLLBACK** statement
- **SAVEPOINT** statement

## Data Control Language(DCL)

Gives and removes permissions on database structure

- **GRANT** statement
- **REVOKE** statement

# Table Creation

## CREATE TABLE statement

CREATE statement is used for table creation. The syntax is:

**CREATE TABLE table\_name( column datatype,  
column datatype, ... column datatype);**

For example, create a table for STUDENT ( Roll, Name, Gender, Age, CGPA)

Solution: *CREATE TABLE STUDENT(Roll NUMBER(6), Name VARCHAR2(20), Gender CHAR(1), Age NUMBER(3), CGPA NUMBER(4,2));*

# Viewing Table Structure

## DESCRIBE statement

DESCRIBE statement is used for viewing table structure. The syntax is:

**DESCRIBE table\_name;** or  
**DESC table\_name;**

For example: *DESCRIBE STUDENT;*  
Solution:

| Name   | Null? | Type         |
|--------|-------|--------------|
| Roll   |       | NUMBER(6)    |
| Name   |       | VARCHAR2(20) |
| Gender |       | CHAR(1)      |
| Age    |       | NUMBER(3)    |
| CGPA   |       | NUMBER(4,2)  |



# Record Insertion

## INSERT statement

INSERT statement is used to insert a new row/record into a table. The syntax is:

**INSERT INTO table\_name (column1, column2,..) VALUES (value1, value2,..);**

- Column names are optional
- Numeric data is not enclosed within quotes; while character and date values are enclosed within single quotes

*INSERT INTO STUDENT(Roll, Name, Gender, Age,  
CGPA) VALUES (705129, 'Uday', 'M', 19, 9.2);*

*INSERT INTO STUDENT VALUES (705129, 'Uday', 'M',  
19, 9.2);*

*INSERT INTO STUDENT(Roll, Name, CGPA) VALUES  
(705129, 'Uday', 9.2);*

*INSERT INTO STUDENT VALUES (&Roll, '&Name',  
'&Gender', &Age, &CGPA);*

*INSERT INTO STUDENT (Roll, Name, Gender, Age)  
VALUES(&Roll, '&Name', '&Gender', &Age);*

## Querying Data

### SELECT statement

SELECT statement is used to retrieve data from the underlying table. The syntax is:

**SELECT column1,column2 FROM table\_name;**

If the user wants to see all the columns in a table, \* can be used in place of columns

*SELECT \* FROM STUDENT;*

| Roll   | Name | Gender | Age | CGPA |
|--------|------|--------|-----|------|
| 705129 | Uday | M      | 19  | 9.2  |
| 705170 | Ram  | M      | 20  |      |
| 705171 | Kim  | F      | 19  | 8.6  |
| 705172 | Raji |        | 20  | 7.5  |

**NULL value** means the value is unknown or doesn't exist

*SELECT Roll, Name, CGPA FROM STUDENT;*

| Roll   | Name | CGPA |
|--------|------|------|
| 705129 | Uday | 9.2  |
| 705170 | Ram  |      |
| 705171 | Kim  | 8.6  |
| 705172 | Raji | 7.5  |



## Displaying Distinct Rows

The DISTINCT keyword is used to suppress duplicate values.  
The syntax is:

**SELECT DISTINCT column FROM tablename;**

*SELECT DISTINCT City FROM Student;*

| City          |
|---------------|
| Bhubaneswar   |
| Jharkhand     |
| Uttar Pradesh |
| Ranchi        |
| Rajasthan     |
| Delhi         |
| Cuttack       |
| Kolkata       |

## Use of Arithmetic Expressions

The arithmetic expressions are used to display mathematically calculated data. The syntax is:

**SELECT column, expression FROM tablename;**

*SELECT Name, Age, Age+3 FROM Student;*

| Name   | Age | Age+3 |
|--------|-----|-------|
| Ram    | 19  | 22    |
| Uday   | 20  | 23    |
| Vikas  | 19  | 22    |
| Sweta  | 19  | 22    |
| Yogesh | 18  | 21    |
| Smriti | 20  | 23    |
| Sudam  | 21  | 24    |
| Vikas  | 23  | 26    |
| Manish | 19  | 22    |

*SELECT Name, Age, Age+3 "Passing Age" FROM Student;*

| Name   | Age | Passing Age |
|--------|-----|-------------|
| Ram    | 19  | 22          |
| Uday   | 20  | 23          |
| Vikas  | 19  | 22          |
| Sweta  | 19  | 22          |
| Yogesh | 18  | 21          |
| Smriti | 20  | 23          |
| Sudam  | 21  | 24          |
| Vikas  | 23  | 26          |
| Manish | 19  | 22          |

## Concatenation

Concatenation joins a column or a character string to another column. The syntax is:

**SELECT column1||' '||column2 [AS] ALIAS FROM  
tablename;**

*SELECT Name||' '||City FROM Student;*

*SELECT Name||' '||City AS "Address"FROM Student;*

| Name  ' '  City     | Address             |
|---------------------|---------------------|
| Ram Bhubaneswar     | Ram Bhubaneswar     |
| Hari Bhubaneswar    | Hari Bhubaneswar    |
| Uday Jharkhand      | Uday Jharkhand      |
| Vikas Uttar Pradesh | Vikas Uttar Pradesh |
| Sweta Ranchi        | Sweta Ranchi        |
| Yogesh Rajasthan    | Yogesh Rajasthan    |
| Smriti Delhi        | Smriti Delhi        |
| Sudam Cuttack       | Sudam Cuttack       |
| Vikas Kolkota       | Vikas Kolkota       |
| Manish              | Manish              |

## Selecting Specific Records

Specific records can be selected by using a WHERE clause with the SELECT statement. The syntax is:

**SELECT columns FROM tablename WHERE *cond*<sup>n</sup>;**

*SELECT \* FROM Student WHERE city= 'Bhubaneswar';*

| Roll | Name | City        | Age | CGPA |
|------|------|-------------|-----|------|
| 101  | Ram  | Bhubaneswar | 19  | 9.0  |
| 102  | Hari | Bhubaneswar |     | 6.7  |



## Operators used in WHERE condition

### Relational Operators

= ex: CGPA=9.0  
> ex: Age>20  
< ex: Age<20  
>= ex: Age>=20  
<= ex: Age<=20  
<> or != ex: Name !='Hari'  
ANY ex: Age > ANY(20,23,19)  
ALL ex: Age > ALL(20,18)

### Logical Operators

AND ex: City='Bhubaneswar' AND Age=20  
OR ex: City ='Bhubaneswar' OR Age=20  
NOT ex: NOT(Age=20 OR Age=21)

### LIKE Operator

LIKE operator uses wild cards for matching as:

?: represents zero or more characters

\_: represents any one character

ex: Name LIKE 'S%'

ex: Name LIKE 'S\_\_'

ex: Name LIKE '%i%'

ex: Name LIKE '\_i%'

### Special Operators

IN ex: City IN('Delhi','Cuttack','Ranchi')

BETWEEN ex: Age BETWEEN 20 AND 22

IS NULL ex: SELECT Name FROM Student WHERE Age is NULL;

### ORDER BY clause using column name

ORDER BY clause is used to sort records in a table

**SELECT columns FROM tablename [WHERE *cond<sup>n</sup>*]  
ORDER BY column [ASC/DESC];**

*SELECT \* FROM Student ORDER BY Age;*

*SELECT \* FROM Student ORDER By CGPA, Age DESC;*

NULL values come at the end of the table in case of ORDER BY clause

### ORDER BY clause using column number

Records can be sorted by using the column number

**SELECT columns FROM tablename [WHERE *cond<sup>n</sup>*]  
ORDER BY columnno [ASC/DESC];**

*SELECT \* FROM Student ORDER BY 3;*



## ALTER Statement

Stud (roll, name, age)

| Column | NULL? | Datatype     |
|--------|-------|--------------|
| ROLL   |       | NUMBER(6)    |
| NAME   |       | VARCHAR2(20) |
| AGE    |       | NUMBER(2)    |

### Adding a New Column

**ALTER TABLE tablename ADD(column definition);**

*ALTER TABLE Stud ADD (address number(20));*

| Column  | NULL? | Datatype     |
|---------|-------|--------------|
| ROLL    |       | NUMBER(6)    |
| NAME    |       | VARCHAR2(20) |
| AGE     |       | NUMBER(2)    |
| ADDRESS |       | NUMBER(20)   |

### Modifying an Existing Column

**ALTER TABLE tablename MODIFY(column definition);**

*ALTER TABLE Stud MODIFY(address varchar2(20));*

| Column  | NULL? | Datatype     |
|---------|-------|--------------|
| ROLL    |       | NUMBER(6)    |
| NAME    |       | VARCHAR2(20) |
| AGE     |       | NUMBER(2)    |
| ADDRESS |       | VARCHAR2(20) |

### Dropping a Column

**ALTER TABLE tablename DROP COLUMN columnname;**

*ALTER TABLE Stud DROP COLUMN address;*

| Column | NULL? | Datatype     |
|--------|-------|--------------|
| ROLL   |       | NUMBER(6)    |
| NAME   |       | VARCHAR2(20) |
| AGE    |       | NUMBER(2)    |

**END**