Database Management System (DBMS)

Unit - I

Introduction

- Database Management System (DBMS) Concepts
- Relational Database Model

Unit - II

Structured Query Language

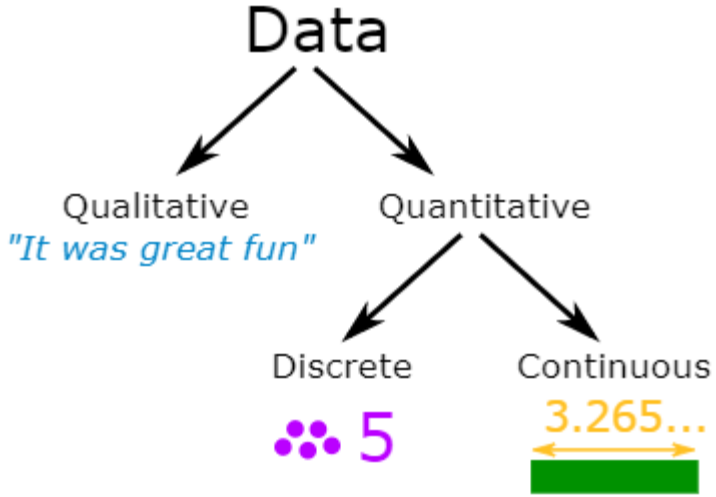
- Introduction to SQL
- SQL sublanguages – DDL, DML, DCL
- Basic data types
- SQL statements: Create, Select, Insert, Delete, Update etc.
- Database constraints

Terminologies

- **Data :** Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.
- **Data can be qualitative or quantitative.**
- **Qualitative:** Qualitative data is descriptive information (it describes something)
  - Your favorite holiday destination
  - Behaviour of a person
  - Describe the smell of a perfume
- **Quantitative:**

  Quantitative data is numerical information (numbers)

  Discrete data is counted, Continuous data is measured
  - Petals on a flower (Discrete)
  - Customers in a shop (Discrete)
  - Height (Continuous)
  - Weight (Continuous)



Terminologies

- **Entity :** Real world object that can be easily identified.
  - Student, Product, Book, Department etc.
- **Attribute :** Properties of an entity.
  - Student (Id, Name, Address)
  - Product (Product Id, Name, Price)
  - Book (Book Id, Author, Publisher, No. of pages)
  - Department (Dept Id, Name)
- **Information :** Data is processed for meaningful to the end users.
  - Attendance report, Product Stock, Subjectwise Book details, Department list etc.
- **Data Types :** Data having predefined characteristics.

E.g. Integer, Real (Float), Character, String, Date, etc.

  - Integer Values 1, 2, …….
  - Real (Float) Values 1.1, 2.9, 3.85, ……
  - Character Values : A … Z, a ….z , # , @ , ……
  - String Values Ambar, Harsh, Gujarat, India ……..
  - Date Values : 1/1/1980, 12/8/2000 , etc.

Fundamentals of RDBMS

- **Database**
  - It is a computerized record-keeping system.
  - It is a collection of coherent and meaningful data

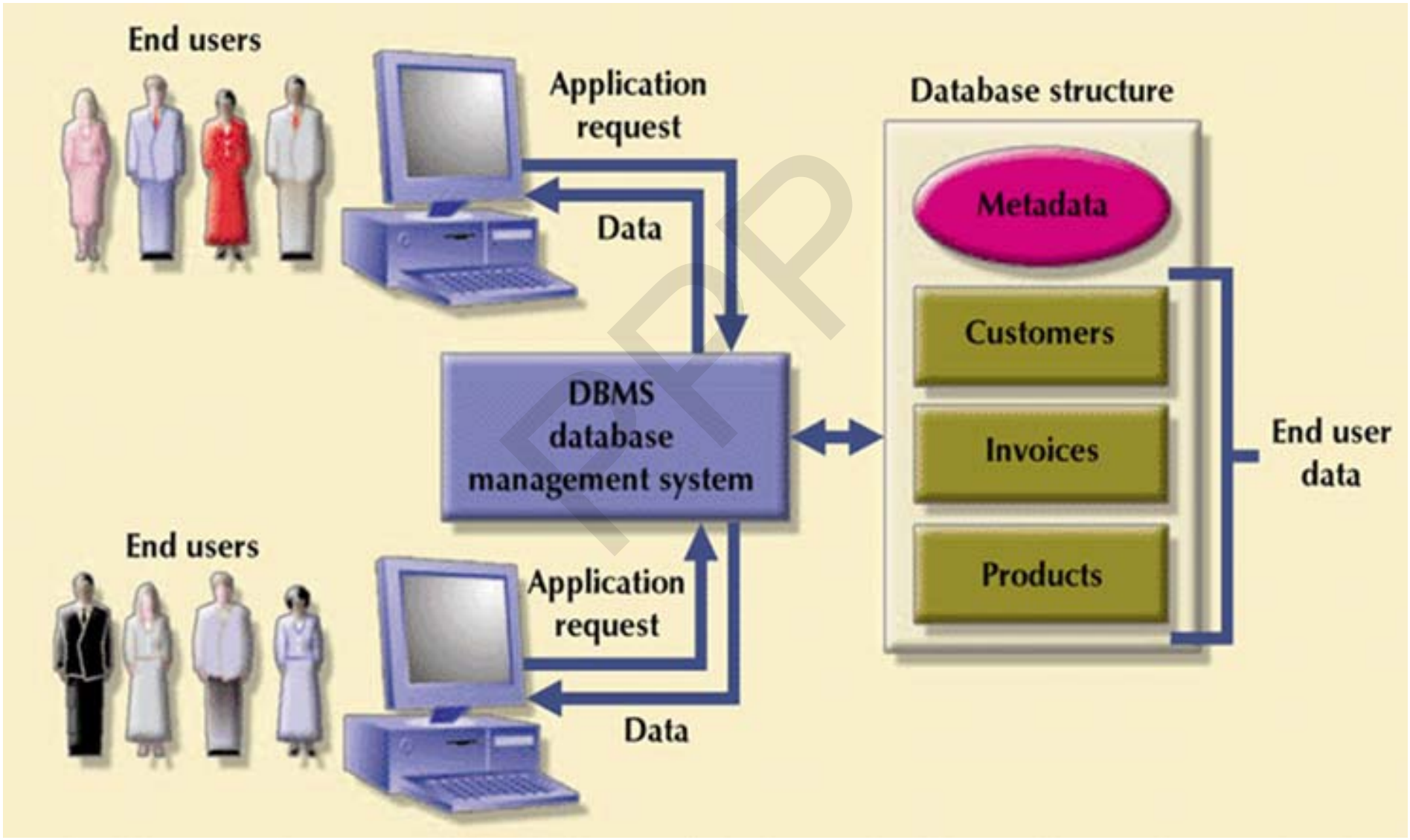
  E.g. A company database : Customers, Invoices and Products
- **DBMS**
  - It is a set of programs designed to create and maintain a database.

  OR
  - A system that allows creating, inserting, deleting, updating and processing of data.

  E.g. XML, Dbase, FoxPro etc.
- **RDBMS**
  - The Relational Database was invented by Edger. F. Codd at IBM in 1970.
  - A DBMS that is based on relational model and stores data in the form of related tables.

  E.g. Oracle, Microsoft SQL Server, Sybase, SQL Server, IBM's DB2

DBMS manages interaction between end users and database.



DBMS Benefits

- Controlling Redundancy : Redundancy can be reduced.
  - E.g. Students Data with Admin Level, Department Level, & Printing Dept.
- Restricting Unauthorized Access
  - Each user must follow authentication process
  - Each user have the specific rights (Authorization) to access the resources.
- Providing Persistent Storage for Program Objects
  - Objects like (Table, Procedure, Functions etc.) are stored in flash or non-volatile memory to protect against the power failure.
- Providing Storage Structures for Efficient Query Processing :
  - Indexes : It is used to quickly locate data without having to search every row in a database table. First column is index field & second column is pointer to the physical storage of the record.
  - Optimization : To determine the most efficient way to execute a given query by considering the possible query plans. The parameters like CPU Time, No. of Input / Output, No. of joins used for optimization.
- Providing Backup and Recovery
  - Export utility for full or partial back up of data stored in database.
  - Import utility recover the data in database.

DBMS Benefits

- Provides Multiple User Interfaces :
  - The data should be available to users as per their needs.
  - E.g. Employee (Id, Name, DoB, Email, Contact, Basic, HRA, DA, Salary, Designation, Department)
  - Finance Department Needs : Id, Name, Dept., Desi., Basic, HRA, DA, Salary
  - HR Department Needs : Id, Name, DoB, Email, Contact
  - Project Manager Needs Id, Name, Department & Designation
- Representing Complex Relationships among data
  - Data stored in a database is connected with each other and a relationship is made in between data. It provides efficient and accurate data.
- Enforce Integrity Constraints :
  - Data type : Number, Char, Date, Varchar2 etc.
  - Field Length : Length of field according to data placed in it.
  - Primary Key : To uniquely identify each row in a table.
  - Foreign Key : To reference the value from other table.
  - Not NULL : Value for the field is compulsory
  - Check Constraints : Restricted values are stored in field of a table.

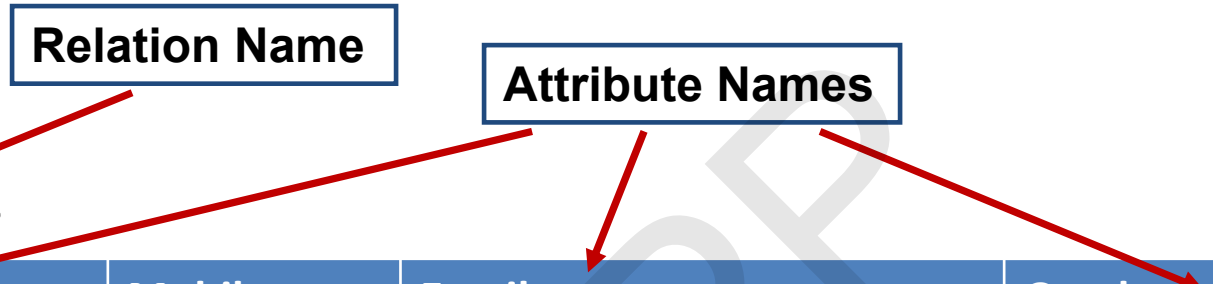
DBMS Benefits

- Permitting Interfacing And Actions Using Rules
  - Triggers, Stored Procedure & Functions
- Potential For Enforcing Standards
  - DBA Define & Enforce Standards Like
    - Names and Formats of data element
    - Report Structures
- Reduce Application Development Time
  - Once the database up & running, Development time is less.
- Flexibility
  - Change the structure of database.
  - Easily add a table and extend it in database.
- Availability of Up-to-Date Information
  - As soon as one users update is available to the database, all the other users of the database can immediately see the update.
  - E.g. Ticket Reservation, Hotel Booking, Bank Passbook etc.
- Economies of Scale :
  - Large scale business invest in more powerful processors, storage devices, communication lines.
  - E.g. Email, Social Media, Super Market, Flipkart, Amazon etc.
  - Reduce overall cost of operation and management.

Relation & Attribute

Relation : It is a table with name. It is set of tuples or rows.

- E.g.



Students

Id	Name	Mobile	Email	Gender
1	Shyam	9089787678	Shyam_parmar@yahoo.com	Male
2	Kalpna	9456453423	Kalpna_shah@gmail.com	Female
3	Abhay	8907898765	Abhay_rathod@yahoo.com	Male

Attribute

- Each attribute of a relation has a name
- Attribute values are **atomic**; that is, indivisible

- **Schema of the relation** is Student (Id, Name, Mobile, Email, Gender)

Domain of an Attribute

The domain of an attribute A, denoted by Dom(A)

It is the set of values that the attribute can take.

Examples of domains:

- Mobile_numbers: The set of ten-digit phone numbers.
- Aadhar Card Numbers: The set of valid twelve-digit numbers.
- Person Name: The set of characters known as string

Datatype of an Attribute

- Domain is represented by a data type.
- For example
  - Mobile_ number — Number(10)
  - Employee_age — Number(3)
  - Department_Name — Char(50)
  - Birth_Date — Date
  - Product_Id — Number(3)
  - Author_Name — Char(50)
  - Joining_Date — Date

Relation Schema & Relation Instance

- A **relation schema R**, denoted by **R(A1, A2, ... , An)** is made up of a relation name R and a list of attributes A1, A2, ..., An.

  E.g. Student (Id, Name, Mobile, Email, Gender)

- A **relation instance** (state) of a relation schema R(A1, …, An), denoted by r(R), is a set of tuples in the table of R at some instance of time.

E.g. r (Student)

Id	Name	Mobile	Email	Gender
1	Shyam	9089787678	Shyam_parmar@yahoo.com	Male
2	Kalpna	9456453423	Kalpna_shah@gmail.com	Female

Schema & Instance Update

- ☛A schema may have different states at different times.

- The **schema of a relation** may change (e.g., adding, deleting, renaming attributes and deleting a table schema) but it is infrequent

- The **state of a relation** may also change (e.g., inserting or deleting a tuple, and changing an attribute value in a tuple) & it is frequent

Relational Model

- It represents the database as a collection of relations.
- Each relation is a table of values **or**
- Each relation is a file of records

- **In the relational model,**
  - A table is called a relation.
  - A row is called a tuple
  - A column header is called an attribute
  - The data type in each column is represented by a domain of possible values.

Relational Model

- It is the primary data model used for data storage and processing.

- **History of Relational Model**

- 1970 E.F. Codd proposed relational model

- Early microcomputer based DBMSs - dBase, R:base, Paradox

- Most popular enterprise DBMSs, all relational
  - Oracle, DB2, Informix, Sybase
  - Microsoft's SQL Server
  - MySQL, PostgreSQL

Difference DBMS & RDBMS

DBMS	RDBMS
•DBMS stands for Database Management System.	RDBMS stands for Relational Database Management System.
Data is stored as file.	Data is stored as tables.
DBMS deals with small quantity of data.	RDBMS deals with large quantity of data.
DBMS supports single user at a time.	RDBMS supports multiple users at a time.
E.g. MSAccess.	E.g. Oracle , SQL Server etc.
•No much Secure	More Secure : Logging at O/S level Command Level & Object level
•Dose not Support Distributed Databases	Support Distributed Databases
• Relationship between two tables or files maintained by programmed.	Relationship between two tables or files can be specified at the time of table creation.

Unit – II

SQL (Structured Query Language)

- Introduction to SQL
- SQL sublanguages – DDL, DML, DCL
- Basic data types
- SQL statements: Create, Select, Insert, Delete, Update etc.
- Database constraints

SQL (Structured Query Langugar)

What is SQL?

- SQL is a language that provides an interface to RDBMS.
- SQL is an ANSI (American National Standards Institute) standard language.
- RDBMS like SQL Server , Oracle, MS Access use SQL as their standard database language.

Features of SQL

- SQL used to access and manipulate databases. It can
  - Creates a new objects in database
  - Insert records in a database
  - Update records in a database
  - Delete records from a database.
  - Retrieve records from a database.

SQL Engine : It is a server side process takes SQL queries from the application and execute.

Rules For SQL

- SQL starts with a verb (i.e. A SQL action word)
  - E.g. CREATE , INSERT , UPDATE , DELETE

- A comma (**,**) separates parameters.
  - E.g. INSERT INTO EMP VALUES (1 **,** 'HARSH');

- A ' **;** ' is used to end SQL statements.
  - E.g. INSERT INTO EMP VALUES (1 , 'HARSH') ;

- A space separate clauses. E.g. DROP TABLE EMP;
- Reserve words cannot be used as identifier unless enclosed with double quotes.
- Character and Data literals must be enclosed within single quotes.

DDL (Data Definition Language) commands

- It is set of SQL commands used to create, modify and delete database structures but not data.

- DDL commands **implicitly issues a commit** command to the database.

- **CREATE** – To create objects in the database.

- **ALTER** – Alters / Modifies the existing structure of the database.

- **DROP** – Delete an entire object from the database.

- **TRUNCATE** – Remove all records from a table, but the structure of table is present.

DML (Data Manipulation Language) commands

- Commands of SQL that allows inserting, changing or retrieving data within the database.

- **INSERT** – Insert data into a table.

- **UPDATE** – Updates existing data within a table.

- **DELETE** – Deletes records from a table.

- **SELECT** – Retrieve certain / all records from one or more tables.

DCL (Data Control Language) command

- Commands of SQL that control access to data and to the database.
- **COMMIT** – Save work done.
- **ROLLBACK** – Restore database to original since the last COMMIT.
- **GRANT –** Grant permissions to the oracle users.
- **REVOKE** – Revert permissions from the oracle users.

Data Types

Data Type	Description
CHAR(size)	Is used to store character strings values of fixed length. The size in brackets the number of characters the cell can hold. The maximum number of characters hold is 2000.
VARCHAR(size) / VARCHAR2(size)	Is used to store variable length alphanumeric data. It is a more flexible form of CHAR data type. Varchar can hold maximum 2000 characters. Varchar2 can hold maximum 4000 characters.
DATE	It is used to represent date and time. The standard format is DD-MON-YY as in 21-JUN-04. To enter dates other than standard format use the inbuilt functions.

Data Types

Data Type	Description
NUMBER(P, S)	It is used to store numbers (fixed or floating point). P : Maximum length of the data. It includes both decimal & fractional part digits. S : Number of places to the right of the decimal point. Precision is maximum 38 digits. If scale is not defined: Maximum no. of digits used in decimal part is P. If scale is defined: Maximum (P – S) no. of digits are used for Decimal part and maximum S number of digits are used for fractional part. Valid values : 0, positive nos. & negative nos. Numbers may be expressed in two ways
LONG	It is used to store variable length character strings containing up to 2 GB.
LONG RAW	These data types are used to store binary data, such as digitized picture or image. LONG RAW data type can contain up to 2 GB. Values stored in columns having LONG RAW data type cannot be indexed.

Data Types

Data Type	Description
Rowid	Every record in the database has a physical address or rowid. Fixed length binary data. The format of rowid is : BBBBBBB.RRRR.FFFFF Where **B :** Block in the database file **R :** row in the block
NCHAR(size)	Is used to store character strings values of fixed length. The characters may be of different language. The maximum number of characters hold is 2000.
NVARCHAR(size) / NVARCHAR2(size)	Is used to store variable length alphanumeric data. It is a more flexible form of NCHAR data type. NVarchar can hold maximum 2000 characters. NVarchar2 can hold maximum 4000 characters.
Blob	Maximum size is 4 GB. Large Binary Objects within the database.
Clob	Maximum size is 4 GB. Large Character Large Objects within the database.
Nclob	Maximum size is 4 GB. Large Character Large Objects within the database.

Primary Key

- A **Primary key** is a key (Attribute/Field) that uniquely identifies a row in each table.
- It is denoted with its first two letters, namely, **PK**.
- Employee table is uniquely identified by Employee_No field than it is called a primary key of a table.
- Project Table Project number is PK.

Project Master	
Project Number **(PK)**	Project Name
P001	HRMS
P002	Online Bidding
P003	Payroll

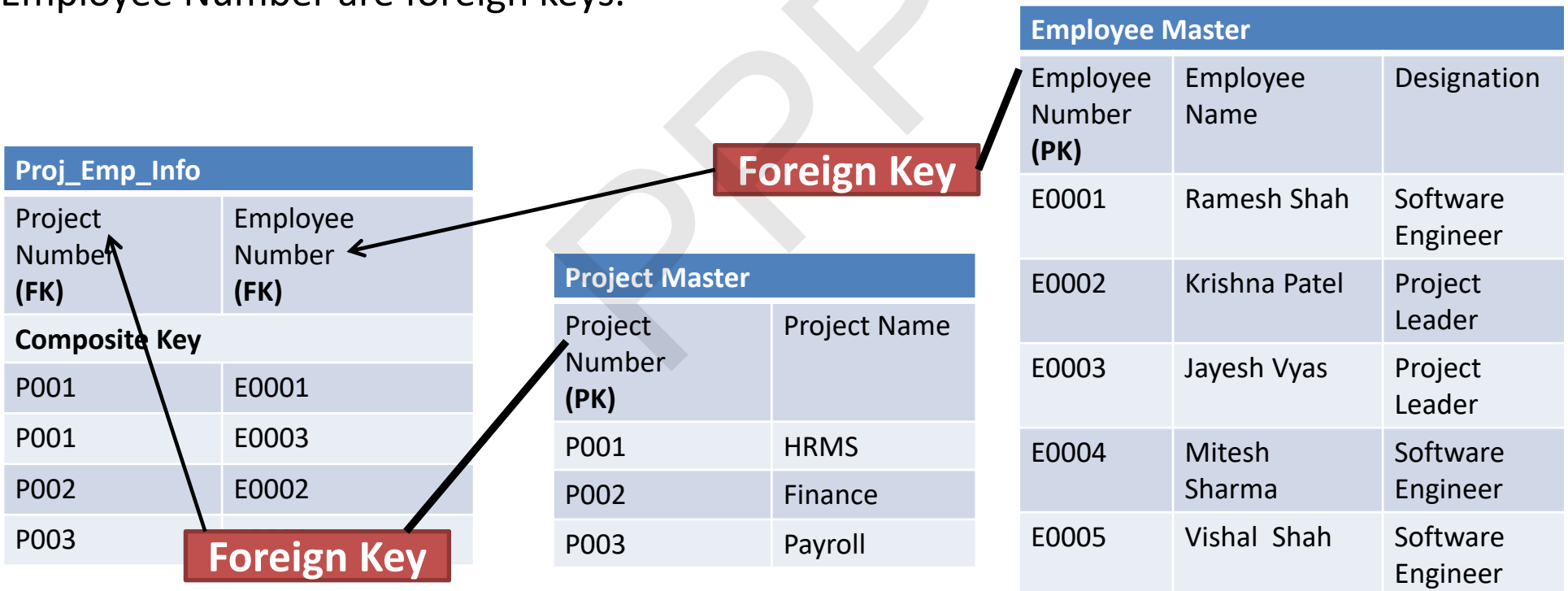
Employee Master		
Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer

Foreign Key

- A **Foreign key** is a key borrowed from another related table (that's why its foreign) in order to make the relationship between two tables.
- It is normally denoted with its first two letters, namely, **FK**.
- **Foreign key** must match actual values and data types with the related **Primary Key.**

E.g. Proj_Emp_Info table Project_Number &

Employee Number are foreign keys.



Proj_Emp_Info

Project Number **(FK)**	Employee Number **(FK)**
Composite Key	
P001	E0001
P001	E0003
P002	E0002
P003	

Project Master

Project Number **(PK)**	Project Name
P001	HRMS
P002	Finance
P003	Payroll

Employee Master

Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer

Alternate Key

- A **key** associated with one or more columns whose values uniquely identify every row in the table, but which is not the primary **key**.
- **E.g.** Employee Master having Employee Name as Alternate key but it is not primary key.

Employee Master		
Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer

Create Statement

- It is used to create a new **Table (Entity or Relation)** in database.
- Each table having multiple **Columns (Attributes or Fields)**.
- **Each Column of contains three things :**
  - Column-name,
  - Data-type &
  - Size
- **Rules for Creating Tables**
  - A name can have maximum upto 30 characters.
  - Alphabet from A-Z, a-z and number from 0-9 are allowed.
  - A name should begin with Alphabet.
  - The use of special characters like _ , $ & # signs are allowed.
  - SQL reserved words are not allowed.
- **Syntax**
- **CREATE TABLE** *table_name*
  (
  column_name1 data_type(*size*)
  ……. …….
  column_nameN data_type(*size*) *);*

Create Statement

```
CREATE TABLE Student_Master
   ( Stud_Id        NUMBER (2) PRIMARY KEY ,
     Stud_Name VARCHAR(20),
     Stud_Birth_Date DATE,
     Stud_Mobile   NUMBER(10)
   )
```

Constraints

- **Primary Key**
  - **At column level**
    - Column-name data-type(size) **PRIMARY KEY**
  - E.g.

```
CREATE TABLE Student_Master
( Stud_Id  NUMBER (2) PRIMARY KEY ,
  Stud_Name VARCHAR(20),
  Stud_Birth_Date DATE,
  Stud_Mobile   NUMBER(10) )          /
```

OR

  - **At table level**
    - **PRIMARY KEY** (col-name1, col-name2, ……. col-nameN)

```
CREATE TABLE Student_Master
  ( Stud_Id  NUMBER (2),
    Stud_Name VARCHAR(20),
    Stud_Birth_Date DATE,
    Stud_Mobile   NUMBER(10),
    PRIMARY KEY (Stud_Id))        /
```

To Display Table Structure

- DESCRIBE Table-name

OR

- DESC Table-name

E.g.

DESCRIBE Student_Master;

DESC Student_Master;

SQL Query Is Not a Case-Sensitive

- Query may be allowed in either
  - Upper case or
  - Lowercase or
  - Combination of uppercase and lowercase

E.g.

DESC Student_Master;

desc Student_Master;

DesC Student_Master;

DESC STUDENT_MasTer;

Insert Statement

- **Creates a new row in database tables and loads the values passed into the columns specified.**
- **Syntax**
- To insert values of all fields/attributes
  - **INSERT INTO** *table_name* **VALUES** (*value1*,*value2*, ...*valueN*)

INSERT INTO STUDENT_MASTER **VALUES** (1 , 'Amit' , '**05-DEC-90**', 9422290900)

OR

- To insert values of selected fields/attributes
  - **INSERT INTO** *table_name* (*column1*,*column2*, ..*columnN*) **VALUES** (*value1*,*value2*, ..*valueN*);
  - **INSERT INTO** STUDENT_MASTER (Stud_Id, Stud_Name) **VALUES** (2,'Kalpesh');

To Insert Records In A Table

- To insert multiple records of all fields/attributes
  - **INSERT INTO** *table_name*
    VALUES (&*column1*,&column2, ... &column*N*);
- To insert multiple records of selected fields/attributes
  - **INSERT INTO** *table_name* (*column1*, ... *columnX*)
    VALUES (&*column1,* ... &columnX);

  - **INSERT INTO** STUDENT_MASTER **VALUES**
    (&STUD_ID,'&STUD_NAME', '&STUD_BIRTH_DATE',
    &STUD_MOBILE);

    OR

  - **INSERT INTO** STUDENT_MASTER (STUD_ID) **VALUES** (&STUD_ID);

To Insert Records In A Table

- To insert multiple records of all fields/attributes
  - **INSERT INTO** *table_name* (*column1*,*column2*, ... *columnN*)
    VALUES (&*column1*,&column2, ... &column*N*);

E.g. For Oracle 11g Express Edition in Browser.

```
INSERT INTO STUDENT_MASTER
(STUD_ID,STUD_NAME,STUD_BIRTH_DATE,STUD_MOBILE)
 WITH names AS (
  SELECT 4, 'Ratna',    '10-20-98', '111111111'   FROM dual UNION ALL
  SELECT 6, 'Prakash',    '5-25-97', '3333333311'   FROM dual UNION ALL
  SELECT 5, 'Kartik', '09-23-86', '222222222'   FROM dual
 )
SELECT * FROM Names
```

Select Statement

- **It is used to retrieve rows select from one or more tables.**

- **Syntax**

SELECT [**DISTINCT**]

{ *|*column_name1, column_name2, .. column_nameN* **}**

FROM *table_name*

[**Where** conditions]

[**ORDER BY {** column_name1.., column_nameN [**ASC / DESC**] }] **;**

Select Statement

- **All rows and all columns**
  - SELECT * FROM table-name;

  E.g.
  - SELECT * FROM Student_Master;

- **Filtering Data**
  - **Select columns and all rows**
    - SELECT { **Selected COLUMN-LIST** } FROM table-name;

    E.g.
    - SELECT Stud_Id, Stud_Name FROM Student_Master;
    - SELECT Stud_Name FROM Student_Master;
    - SELECT Stud_Id FROM Student_Master;

Select Statement

- **Selected rows and all columns**
  - SELECT * FROM table-name [WHERE conditions]

E.g.

  - SELECT * FROM Student_Master **WHERE** Stud_Id > 2;
  - SELECT * FROM Student_Master **WHERE** Stud_Name = 'Amit';

- **Selected columns and selected rows**
  - SELECT {Selected COLUMN-LIST } FROM table-name [WHERE conditions]

E.g.

  - SELECT Stud_Name FROM Student_Master WHERE Stud_Id > 2;
  - SELECT Stud_iD FROM Student_Master WHERE Stud_Id > 2;

Select Statement

- **Eliminating Duplicate Rows**
  - SELECT DISTINCT * FROM table-name;
  - SELECT DISTINCT { Selected Column-list }

    FROM table-name;

  E.g.
  - SELECT DISTINCT * FROM Student_Master;
  - SELECT DISTINCT Stud_Name FROM Student_Master;

Select Statement :: Ordering

- **Sorting Data**
  - SELECT * FROM table-name **ORDER BY**

    { column_name1.., column_nameN [**ASC / DESC**] }

  E.g.
  - SELECT * FROM Student_Master ORDER BY Stud_Name
  - SELECT * FROM Student_Master ORDER BY Stud_Id
  - SELECT * FROM Student_Master ORDER BY Stud_Name DESC
  - SELECT * FROM Student_Master ORDER BY Stud_Name ASC
  - SELECT * FROM Student_Master ORDER BY Stud_Birth_Date DESC

Select Statement :: Grouping

- **Sorting Data**
  - SELECT * FROM table-name **GROUP BY**

    { column_name1.., column_nameN [**ASC / DESC**] }

  E.g.

  - CREATE TABLE Emp_Master (Emp_id Number(3), Emp_name Varchar(40), Desg Varchar(40));

  - SELECT Desg, Count(Emp_Id) FROM Emp_Master

    GROUP BY Desg

Update Statement

- **Syntax**

  UPDATE *table_name*
  SET
  { *column1=value1,column2=value2,...*
  columnN=valueN }
  [**WHERE** *Conditions*]

- ***Update single column of single row***

  UPDATE *table_name* **SET** *column1=value1*
  WHERE *condition*

  E.g. Change the name of student id = 2.

  – ***UPDATE*** *Student_Master* ***SET*** *Stud_Name='MAHESH'*
  WHERE *Stud_Id = 2*

Update Statement

- **Syntax**

  UPDATE *table_name*
  SET
  { *column1=value1,column2=value2,...*
  columnN=valueN }
  [**WHERE** *Conditions*]

- ***Update more than one column of single row.***

  UPDATE *table_name* **SET** *column1=value1, …….. columnN=valueN* ***WHERE*** *condition*

  E.g. Change the mobile & name of student with id = 2.

  UPDATE Student_Master
  SET Stud_Mobile = 8811223355 , Stud_name ='Maulik'
  WHERE Stud_Id = 2

Update Statement

- **Syntax**

  UPDATE *table_name*
  SET
  { *column1=value1,column2=value2,...*
  columnN=valueN}
  [**WHERE** *Conditions]*

- ***Update single column of selected rows.***

  UPDATE *table_name* **SET** *column1=value1 WHERE condition*

  E.g.

ALTER TABLE Student_Master **ADD** Graduation VARCHAR2(10);

  - **UPDATE** Student_Master
    SET Graduation = 'Science'
    WHERE Stud_id > 3

Update Statement

- **Syntax**

  UPDATE *table_name*
  SET
  { *column1=value1,column2=value2,...*
  columnN=valueN }
  [**WHERE** *Conditions*]

- ***Update single column of all rows.***

  UPDATE *table_name* **SET** *column1=value1*

  E.g. **UPDATE** Student_Master
  SET Graduation = 'Science'

Delete Statement

- **Syntax**

  DELETE FROM *table_name* [**WHERE** Condition];

- **Remove selected rows from table**

  DELETE FROM *table_name* **WHERE** Condition;

  E.g.

  DELETE FROM Student_Master **WHERE** Stud_Id = 5

  DELETE FROM Student_Master **WHERE** Stud_Id > 5

  DELETE FROM Student_Master **WHERE** Stud_Name = 'AMIT'

- **Remove all rows from table**

  DELETE FROM *table_name.*

  E.g. **DELETE FROM** Student_Master

Alter Table Syntax

- **ALTER TABLE** table_name **ADD** column_name datatype;

E.g. ALTER TABLE Student_Master **ADD** Email VARCHAR2(100);

- **ALTER TABLE** table_name **DROP COLUMN** column_name;

E.g. ALTER TABLE Student_Master **DROP COLUMN** Email ;

- **ALTER TABLE** table_name **MODIFY COLUMN** column_name datatype;

E.g. ALTER TABLE Student_Master **MODIFY** Email VARCHAR2(150);

- **Restriction on the Alter table**

  The following tasks cannot be performed.
  - Change the name of the table
  - Change the name of the column
  - Decrease the size of a column if any value with more than specified size.

- **Integrity constraints through Alter table**
  - **ALTER TABLE** table_name **ADD PRIMARY KEY** (Col_name1, …..Col_name N)
  - **ALTER TABLE** table_name **DROP PRIMARY KEY**

- **Objects created by user**
  - **SELECT * FROM TAB;**

- **RENAMING Table :** Change the name of a table
  - **RENAME** table-name **TO** new-table-name

- **TRUNCATING Table :** Remove only records, not a Structure of table.
  - **TRUNCATE TABLE** table-name

- **DESTROYING Table :** Remove the structure of the table also.
  - **DROP TABLE** table-name

- **SYNONYMS Table :** Alternative name of a table
  - **CREATE SYNONYM** synonym-name **FOR** table-name

- **DROPPING SYNONYM Table :** Remove alternative name of a table
  - **DROP SYNONYM** synonym-name

Computations on Data

- Arithmetic Operators
  - +, -, *, /, ( )
- Logical Operators
  - OR, AND, NOT
- Range Searching
  - BETWEEN
- Pattern Matching
  - LIKE
- Predicates
  - IN, NOT IN
- DUAL Table

Operators:

SQL supports various arithmetic, logical , character, and relational operators.

Arithmetic Operators:

Arithmetic operators in SQL will return numeric values.

Operators	Function	Example
+	Add	UPDATE Employee_Mst SET basic = basic + 500;
-	Subtract	UPDATE Employee_Mst SET basic = basic - 500;
*	Multiply	UPDATE Employee_Mst SET basic = basic + basic * 5/100;
/	Divide	UPDATE Employee_Mst SET basic = basic + basic * 5/100;

Logical Operators:

Logical operators in SQL will return either true or false value.

Operators	Function	Example
AND	Check two conditions are true	SELECT * FROM Employee_Mst WHERE basic >= 10000 **AND** basic <= 20000;
OR	Check any of the two conditions are true	SELECT * FROM Employee_Mst WHERE basic >= 10000 **OR** dept = 'Sales';
NOT	Reversed the result of logical expression	SELECT * FROM Employee_Mst WHERE **NOT**(basic >= 10000 OR dept = 'Sales');

Character Operators:

Character operators are used to manipulate character strings.

Operators	Function	Example
||	Concatenates character strings	SELECT 'Name is ' || ename FROM Employee_Mst;

Comparison Operators:

Comparison operators in SQL will compare two quantity

Operators	Meaning/Function
=	Equal to
!=, <>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Operators	Meaning/Function
IN	Equal to any member of a given set
NOT IN	Not equal to any member of a given set
BETWEEN	In between two values
NOT BETWEEN	Not in between two values
EXISTS	True if sub query returns at least one row
ANY	Compares a value to each list of value (must be prefaced with =,!=,<,<=,>,>=)
ALL	Compares a value to every list of value (must be prefaced with =,!=,<,<=,>,>=)

Example:

1) To find the employees who are either clerk or manager.

[I] SELECT * FROM Employee_Mst

WHERE post **IN** ('CLERK','MANAGER');

[ii] SELECT * FROM Employee_Mst

WHERE post = 'CLERK' **OR** post = 'MANAGER'

2) To find the employees who are neither clerks nor anager

SELECT * FROM Employee_Mst

WHERE post **NOT IN** ('CLERK','MANAGER');

Example:

3) To find the employees whose salary lies in between 10000 and 20000

[1] SELECT * FROM Employee_Mst

WHERE basic **BETWEEN** 10000 AND 20000;

[2] SELECT * FROM Employee_Mst

WHERE basic >= 10000 AND basic <= 20000;

4) To find the employees whose salary not lies in between 10000 and 20000

SELECT * FROM Employee_Mst

WHERE basic **NOT BETWEEN** 10000 AND 20000;

LIKE Operator: (Matching a character pattern)

When one does not know the exact value for the search conditions and wishes to use a replaceable parameter a LIKE operator can be used.

The replaceable parameters used are shown bellow.

Symbol	Represents
• **%**	Any sequence or more characters
• **_**	Any single character

Example:

1) To list employees whose names begin with 'P'

```
SELECT * FROM Employee_Mst
        WHERE ename LIKE 'P%' ;
```

Example:

2) To list employees whose name start with 'P', third character must be 'R', fourth character must be 'A' and length must be 5.

E.g. Values : ('PARAS',PARAM','PAVAN', 'PARAMVEER')

Selected Values : ('**P**A**RA**S',**P**A**RA**M')

```
SELECT * FROM Employee_Mst
WHERE ename LIKE 'P_RA_';
```

3) To list employees whose name start with 'P' , third character must be 'R', fourth character must be 'A' and length may be any number.

E.g. **Values :** ('PARAS',PARAM','PAVAN','PARAMVEER')

Selected Values : ('**P**A**RA**S',**P**A**RA**M','**P**A**RA**MVEER')

```
SELECT * FROM Employee_Mst
WHERE ename LIKE 'P_RA%';
```

Exists, Not Exists, All & Any Operators

Client_Master (Client_No, Client_Name, Client_*City*, Gender, BirthDate, Pincode)

Bill_Details (bill_no, client_no, bill_amt)

EXIST Operator

- The Oracle EXISTS operator is a Boolean operator that returns either true or false.
- The EXISTS operator is often used with a subquery to test for the existence of rows:
- SELECT * FROM table_name WHERE EXISTS (subquery);
- **E.g. To find all customers who have the order.**

```
SELECT client_no, client_name FROM client_master c
  WHERE EXISTS (
                SELECT 1 FROM bill_details
                WHERE client_no = c.client_no )
                ORDER BY client_name;
```

NOT EXIST Operator

- The Oracle **NOT EXISTS** operator is a Boolean operator that returns either true or false.
- The NOT EXISTS operator is often used with a subquery to test for not existence of rows:
- SELECT * FROM table_name WHERE EXISTS (subquery);
- **E.g. To find all customers who have not given an order.**

```
SELECT client_no, client_name FROM client_master c
  WHERE NOT EXISTS (
                SELECT 1 FROM bill_details
                WHERE client_no = c.client_no )
                ORDER BY client_name;
```

Constraints

Foreign Key

- It defines the relationship between the tables.
- It is a column / columns whose values are derived from the primary key or unique key of some other table.
- The table in which the foreign key is defined is called a Foreign Table or Detail Table.
- The table that defined s the primary key or unique key and is referenced by the foreign key is called Primary Table or Master Table.
- Features:
  - Child may have duplicates and nulls but unless it is specified.
  - Parent record can be delete provided no child record exist.
  - Master table can not be updated if child record exist.
  - Record can not be inserted in detail table if corresponding record does not exist in master table.

Constraints

- **Foreign Key**
  - **At column level**
    - Column-name data-type(size) **REFERENCES** table-name [column-name] [**ON DELETE CASCADE / ON DELETE SET NULL**]

```
CREATE TABLE bill_details
   (bill_no NUMBER (5) PRIMARY KEY ,
   client_no NUMBER(3) REFERENCES
   client_master(client_no),
   bill_amt number(6))
   /
```

Constraints

- **Foreign Key**

– **At table level**

- **FOREIGN KEY** (column-name1, column-name2, …….. column-nameN) **REFERENCES** table-name (column-name1, column-name2, …….. column-nameN) [**ON DELETE CASCADE / ON DELETE SET NULL**]

CREATE TABLE bill_details

(bill_no NUMBER (5) PRIMARY KEY ,

client_no NUMBER(3),

bill_amt number(6),

FOREIGN KEY (client_no) REFERENCES client_master(client_no))

/

Constraints

When user performs delete on master table and if detail table references that value, than it gives error message.

To solve such problem ON DELETE CASCADE is used.

- **ON DELETE CASCADE :** If master record is deleted the details record automatically deleted.
- **ON DELETE SET NULL :** If master record is deleted the details record store null value for it.

```
CREATE TABLE bill_details
   (bill_no NUMBER (5) PRIMARY KEY ,
   client_no NUMBER(3),
   bill_amt number(6),
   FOREIGN KEY (Client_No) REFERENCES
   client_master(client_no) ON DELETE CASCADE )
   /
```

Assigning User Defined Names To Constraints

- **Syntax:**
  - **CONSTRAINT** <Constraint Name> <Constraint Definition>
    - **At Column Level**

E.g.

```
CREATE TABLE bill_details
(bill_no NUMBER (5) CONSTRAINT p_Bill_No PRIMARY KEY,
client_no NUMBER(3),
bill_amt number(6),
FOREIGN KEY (client_no) REFERENCES client_master(client_no))
/
```

Assigning User Defined Names To Constraints

- **Syntax:**
  - **CONSTRAINT** <Constraint Name> <Constraint Definition>

    - **Table Level**
    - **E.g.**

```
CREATE TABLE bill_details
(bill_no NUMBER (5),
client_no NUMBER(3),
bill_amt number(6),
CONSTRAINT p_bill_no_client_no  PRIMARY KEY (bill_no,client_no),
CONSTRAINT f_Client_No FOREIGN KEY (Client_No) REFERENCES
Client_Master(Client_No) )
/
```

Constraints

- When constraints are defined, Oracle assigns a unique name to each constraint.
- The conversion is used by Oracle is SYS_C**n** : n is numeric value that makes the constraint name unique.
- **System table : User_Constraints**
  - It is used to store the constraints (like primary key , foreign key, check constraints, default constraints) of the tables
  - **DESC User_Constraints**
    - OWNER
    - CONSTRAINT_NAME
    - TABLE_NAME
    - CONSTRAINT_TYPE
    - R_OWNER
    - R_CONSTRAINT_NAME

E.g. SELECT Owner, Constraint_Name, Table_Name, Constraint_Type FROM user_constraints WHERE table_name = 'CLIENT_MASTER'

Constraints

- **Unique Key**
- It allows store unique values in column as primary key.
- But it allows multiple entries of NULL into the column.
- **Features**
  - It will not allow duplicate values.
  - Unique index is created automatically.
  - A table can have more than one Unique Key.
  - Unique key can combine upto 16 columns.
  - Unique key can not be LONG or LONG RAW Data type.
  - **Column-level**
    - Column-name data-type(size) [**UINIQUE KEY / UNIQUE]**
  - **Table – level**
    - **[UNIQUE KEY / UNIQUE] (** Column1,….. Column N)

Constraints

- **Unique Key**
  - **Column-level**
    - Column-name data-type(size) [**UINIQUE KEY / UNIQUE]**
    - **E.g.**
    - CREATE TABLE Client_Master (Client_No NUMBER(3) ,
      Client_Name VARCHAR(50), Client_City VARCHAR(50) ,
      Gender CHAR(1), BirthDate DATE, Pincode NUMBER(6),
      Mobile_No NUMBER(10) **UNIQUE**)
  - **Table – level**
    - **[UNIQUE KEY / UNIQUE] (** Column1,….. Column N)
    - **E.g.**
    - CREATE TABLE Client_Master (Client_No NUMBER(3) ,
      Client_Name VARCHAR(50), Client_City VARCHAR(50) ,
      Gender CHAR(1), BirthDate DATE, Pincode NUMBER(6),
      Mobile_No NUMBER(10), **UNIQUE (Mobile_No , BirthDate))**

Constraints

- **NOT NULL Constraint**
  - **It is a column level constraint.**
  - It ensure that a table column cannot be left empty.
  - it used for mandatory columns.
  - **Column-level**
    - Column-name data-type(size) **[NOT NULL]**

E.g.

- **CREATE TABLE** STUDENT_MASTER
  (s_id NUMBER(2),
  s_name VARCHAR(50) **NOT NULL,**
  PRIMARY KEY (s_id));

Constraints

- **DEFAULT Constraint**
  - **It is a column level constraint.**
  - It specify the default value of a table column.
  - **Column-level**
    - Column-name data-type(size) **DEFAULT (Value)**

E.g. Majority of students city = 'ANAND'.

```
CREATE TABLE STUDENT_MASTER
      (s_id NUMBER(3),
      s_name VARCHAR(50) NOT NULL,
        s_city VARCHAR(50) DEFAULT  'ANAND' ,
      PRIMARY KEY (s_id)  );
```

Constraints

- **CHECK Constraint**
  - Business rules are applied to data prior the data is being inserted into table columns.
  - This ensures that the data (records) in the table have integrity.
    - E.g.
      - Client Name should be inserted with Upper Case letters.
      - Employee Id should be start with 'E'.
      - Student ID between 1000 to 1100
      - No employee in company get salary less than Rs. 5000
  - It can be applied at a column level & table level by using CHECK clause.
  - It is a logical expression that evaluates either TRUE or FALSE.
  - It takes longer time to execute as compared to NOT NULL, Primary Key, Foreign Key & Unique Key.

Constraints

- **CHECK Constraint**
  - **Column-level**
    - Column-name data-type(size) **CHECK (** logical-expression**)**

E.g.

```
CREATE TABLE STUDENT_MASTER
(s_id NUMBER(3) CHECK (s_id BETWEEN 1000 and 1100),
 s_name VARCHAR(50) CHECK ( s_name = UPPER(s_name)),
 s_gender  CHAR(1) CHECK (s_gender IN ('M','F')),
 s_age NUMBER(2) CHECK ( s_age >= 18 ) ,
 s_city VARCHAR(50) DEFAULT  'ANAND' ,
 PRIMARY KEY (s_id)  )
```

Constraints

- **CHECK Constraint**
  - **Table-level**
    - **CHECK (** logical-expression1**), … CHECK (** logical-expressionN**)**

```
CREATE TABLE STUDENT
(s_id NUMBER(3),
 s_name VARCHAR(50) CHECK ( s_name = UPPER(s_name)),
 s_gender  CHAR(1),
 s_age NUMBER(2) CHECK ( s_age >= 18 ) ,
 s_city VARCHAR(50) DEFAULT  'ANAND' ,
CONSTRAINT c_sid CHECK (s_id BETWEEN 1000 and 1100),
CONSTRAINT c_sgender CHECK (s_gender IN ('M','F')),
 PRIMARY KEY (s_id)  )
```

Multiple Check Constraints

- **CHECK Constraint**
  - **E.g.**

```
CREATE TABLE STUDENT
(s_id NUMBER(3),
 s_name VARCHAR(50) CHECK ( s_name = UPPER(s_name)),
 s_gender  CHAR(1),
 s_age NUMBER(2) ,
 s_city VARCHAR(50) DEFAULT  'ANAND' ,
CONSTRAINT c_sga CHECK
        (s_id BETWEEN 1000 and 1100  AND
        s_gender IN ('M','F')  AND
        s_age >= 18 ),
 PRIMARY KEY (s_id)  )
```