UNIT-I

Relational Model

Relational Data Model:

* Relational Model can Represent as a Table with columns and rows.
* Each row is known as a Tuple.
* Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom (Ai)

Relational Instance: In the Relational database System, the relational Instance is represented by a finite set of Tuples.

* Relation Instances do not have duplicate Tuples.

Relational Schema: A Relational Schema contains the name of the relation and name of all columns or attributes.

Relational Key: In the Relational key, each row has one or more attributes. It can Identify the row in relation uniquely.

Example : STUDENT Relation.

NAME	ROLL NO	PHONE_NO	ADDRESS	AGE
Ram	14795	xx	Noida	24
Shyam	12839	yy	Delhi	35
Laxman	33289	zz	Gujarat	20
Mahesh	27857	17D	Punjab	27
Ganesh	17282	BB	Haryana	40

* In the given Table, NAME, ROLL_NO, PHONE_NO, ADDRESS and AGE are the attributes.
* The Instance of Schema STUDENT has 5 Tuples.
* t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

Properties of Relations:

* Name of the relation is distinct from all other relations.
* Each relation cell contains exactly one

atomic [single] value

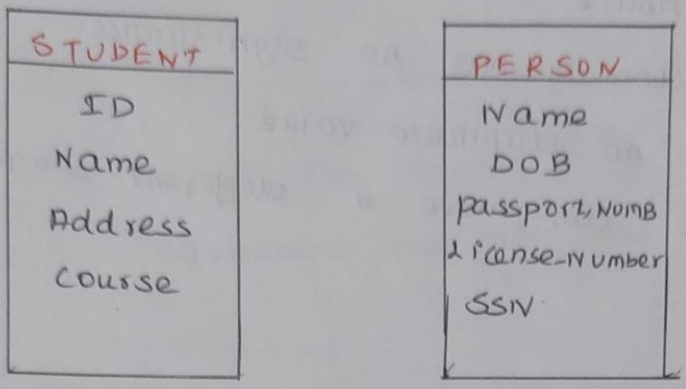
* Each attribute contains a distind name
* Attribute domain has no significance.
* Tuple has no duplicate value
* Order Tuple can have a different seaquence.

Keys:

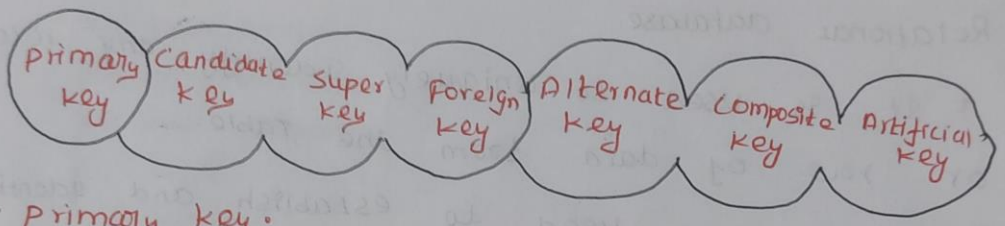
* keys play an Important role in the Relational database.
* It is used to uniquely Identify any record or row of data from the table.
* It is also used to establish and identify relationships between tables.

Example:

* ID is used as a key in the student table because It is unique for each student.
* In the PERSON table, Passport_number, license_number, SSN are keys. since they are unique for each person.



Types of keys:

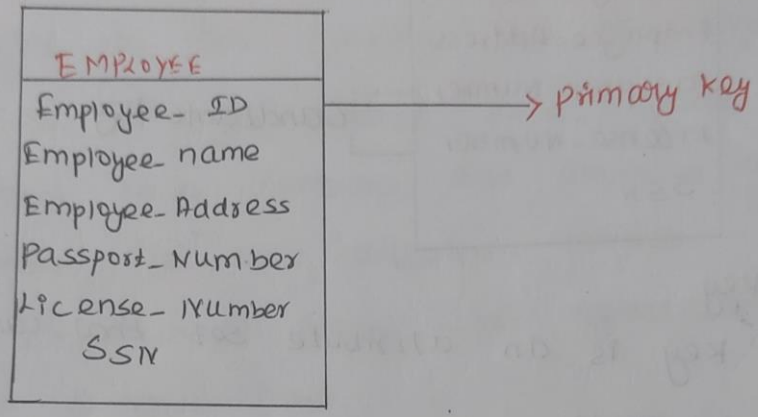


1. Primary key:

* It is the first key used to identify one and only one instance of an entity uniquely.
* An entity can contains multiple keys, as we saw in the PERSON Table.
* The key which is most suitable from those lists becomes a primary key.
* In the EMPLOYEE Table, ID can be the primary key since it is unique for each employee.

* In the EMPLOYEE Table, we can even select License_Number and Passport_Number as primary keys since they are also unique.

* For each entity, the primary key selection is Based on requirements and developers.



2. Candidate key:

* A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

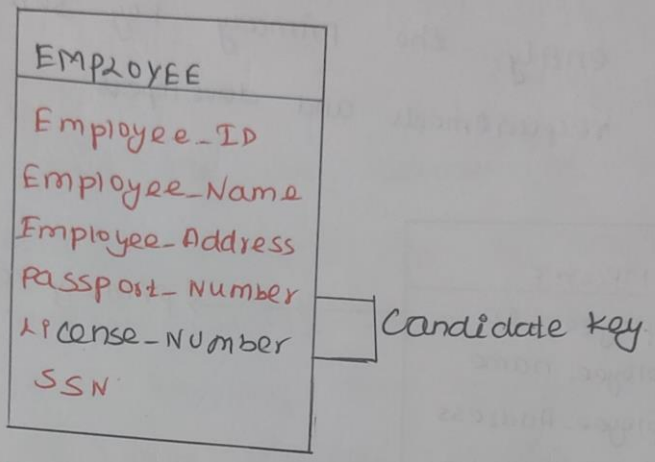
* Except for the primary key, the remaining attributes are considered a candidate key.

* The candidate keys are as strong as the primary key.

Example:

* In the EMPLOYEE Table, Id is best suited for the primary key.

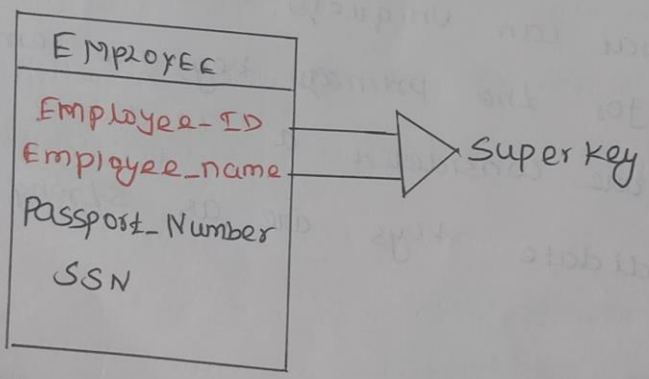
* The rest of the attributes, like SSN, Passport_Number, License_Number, etc.., are considered a candidate key.



3. Super key:

* Super key is an attribute set that can uniquely identify a tuple.

* A Super key is a superset of a candidate key.



Example:

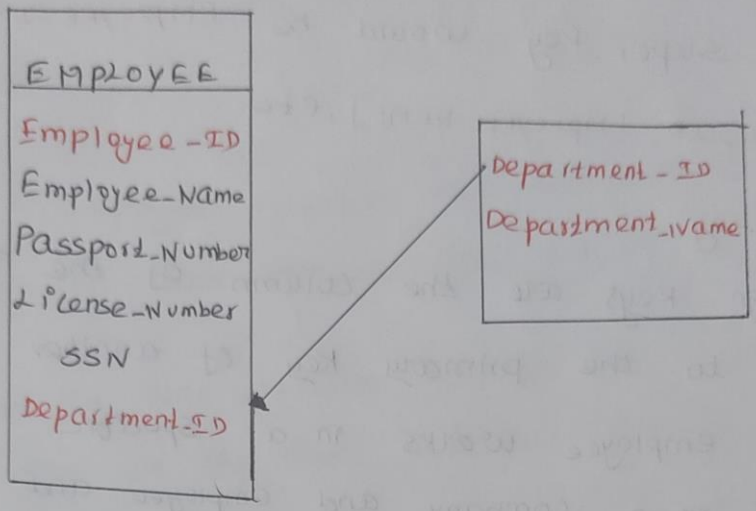
* In the above EMPLOYEE table, for [EMPLOYEE_ID, EMPLOYEE_NAME), the name of two

employees can be the same. Hence, this combination can also be a key.

* The super key would be EMPLOYEE-ID [EMPLOYEE_ID, EMPLOYEE_NAME], etc.

4. Foreign key:

* Foreign keys are the column of the Table used to point to the primary key of another table.
* Every employee works in a specific department in a company, and employee and department are two different entities.
* So we can't store the department's information in the employee Table.
* that's why we link these two Tables through the primary key of one table.
* we add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE Table.
* In the EMPLOYEE Table, Department_Id is the foreign key, Both the tables are related.



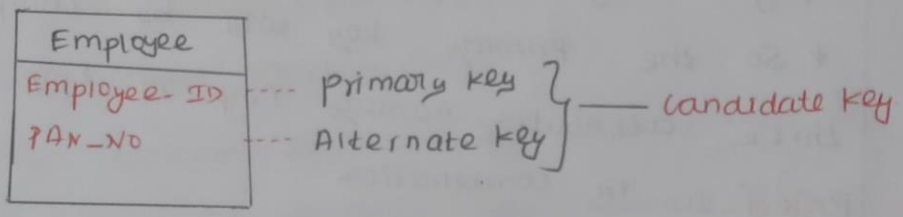
5. Alternate key:

* There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation.
* These attributes or combinations of the attributes are called the candidate keys.
* One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key.
* In other words, the total number of candidate keys minus the primary key.

* The alternate key may or may not exist.
* If there is only one candidate key in a relation it does not have an alternate key.

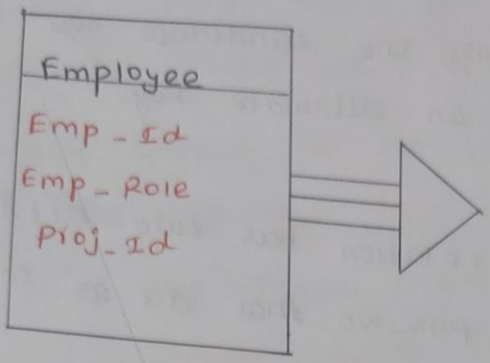
Example:

* Employee relation has two attributes Employee_Id and PAN_NO, that act as candidate key.
* In this relation, Employee_Id is chosen as the Primary key, so the other candidate key, PAN_NO, acts as the Alternate key.



6. Composite key:

* Whenever a primary key consists of more than one attribute, it is known as a composite key.
* This key is also known as concatenated key.
* Diagram; [Turn page]



Example:

* In employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously.
* So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Project_ID in combination.
* So these attributes act as a composite key since the primary key comprises more than one attribute.

7. Artificial Key:

* The key created using arbitrarily assigned data are known as artificial keys.
* These keys are created when a primary key is large and complex and has no relationship with many other relations.

* And the Table in which values are Inserted accordingly is known as child or Rejerencing Table.

* In other words, we can say that the table containing the foreign key is called the child Table, And the Table containing the Primary key/ candidate key is called the referenced or parent table.

The Syntax of the Master Table or Referenced Table is;

CREATE TABLE Student (Roll int PRIMARY KEY, Name Varchar (25), Course Varchar (10);

* Here column Roll is acting as primary key, which will help in deriving the value of foreign key in the child Table.

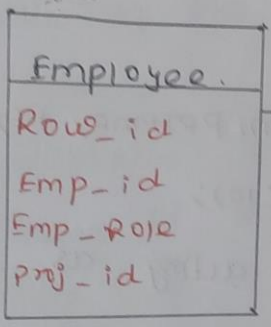
Student Table.

Roll	NAME	COURSE
1	John	MCA
2	smith	MTech
3	Shane	BTech
4	Ricky	MBA
5		

Example:

* In primary key, which is composed of Emp_ID, Emp-role, and Proj-ID, is large in employee relations.

* So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.



Referential integrity:

* A Referential Integrity constraint is also known as foreign key constraint.

* A foreign key is a key whose values are derived from the primary key of another Table.

* The Table from which the values are derived is known as Master or referenced Table.

using the Roll value of Primary key from Master table.

Foreign key constraint OR Referential Integrity Constraint

* There are Two referential integrity constraint.

Insert Constraint: value cannot be inserted in CHILD Table if the value is Not lying in MASTER Table.

Delete Constraint: value cannot be deleted from MASTER Table if the value is lying in CHILD Table.

• Suppose you wanted to Insert Roll=05, with other values of columns in SUBJECT Table, then you will immediately see an error "foreign key constraint violated". [On Running an Insertion command as

Insert into SUBJECT values (5,186, 05); will not be entertained by SQL due to Insertion constraint.

Note; Note; As you cannot Insert value in child Table If the value is not lying in the master

Table, Since Roll=5 is not present in the master Table, hence it will not be allowed to enter Roll = 5 in child Table).

* Similarly, if you want to delete Roll=4 from STUDENT Table, then you will immediately see an error "foreign key constraint violated" [on running a deletion command as:]

* Delete from STUDENT where Roll=4; will not be entertained by SQL due to Deletion Constraint.

Note; As you cannot delete the value from the Master Table if the value is lying in the child Table, Since Roll=5 is present in the child Table, hence it will not be allowed to delete Roll=5 from master Table, Lets, if somehow we managed to delete Roll=5, the Roll=5 will be available in child Table which will ultimately violate Insertion constraint.

ON DELETE CASCADE:- (2 marks) ⓐⓔ

Analyze question; As per deletion constraint:

Value cannot be deleted from the MASTER Table if the value is lying in CHILD

Table.

Ans;

We can delete the value from the Master Table If the value is lying in the child Table without violating the deletion constraint, we have to do slight modification while creating the child Table, (i.e) By adding on delete cascade.

Syntax;-

```
CREATE TABLE Subject (Roll int references Student on
delete cascade, Subcode int, SubName varchar (10));
```

* In above Syntax, Just after references keyword [used for creating foreign key], we have added on Delete cascade, by adding such now, we can delete the value from the Master table If the value is lying in the child Table without violating deletion constraint.

* Now If you wanted to delete Roll=4 from the Master Table, even though Roll=4 is lying in the child Table, It is possible because the moment you give the command to Delete Roll=4, from the master Table, the row having roll=4

from child Table will get replaced by a NULL value

ROLL	NAME	COURSE
1	John	MCA
2	Smith	MTECH
3	Shane	BTECH
4	Ricky	MBA.

SUBJECT TABLE;

Roll	Subcode	SubName.
1	001	DBMS
2	005	SQL
3	006	DS
4	010	OB

* The above two tables STUDENT and SUBJECT having four values each are shown. now suppose you are looking to delete Roll = 4 from (STUDENT master) Table by writing a SQL

Command: Delete from STUDENT where Roll =4;

STUDENT TABLE

ROLL	NAME	COURSE
1	John	MCA
2	Smith	M Tech
3	Shane	BTech
4	Ricky	MBA.

The Syntax of child Table or Referencing Table is

```
CREATE TABLE Subject (Roll int references Student,
Sub code int, SubName varchar (10));
```

SUBJECT TABLE;

ROLL	SubCode	SubName
1	001	DBMS
2	005	SQL
3	006	DS
4	070	OB.

* In the above Table, column Roll is acting as Foreign key, whose values are derived

* And the Table in which values core Inserted accordingly is known as child or Referencing Table In another other words, we can say that the Table containing the foreign key is called the child Table.

* And the Table containing the primary key / Candidate key is called the Referenced or Parent Table.

* The Candidate key can be defined as a Set of attribute which can have zero or more attributes.

The syntax of the Master Table or Referenced Table is:

```
CREATE TABLE Student( Roll int PRIMARY KEY, Name
    varchar (25), Course varchar (10));
```

* Here column Roll is acting as primary key, which will help in deriving the value of foreign key in the child Table.

19

* The moment SQL execute the above command the row having Roll = 4 from SUBJECT (child) Table will get replaced by a NULL value, The Resultant STUDENT and SUBJECT table will look like.

ROLL	NAME	COURSE
1	John	MCA
2	Smith	MTech
3	Shane	BTech

SUBJECT TABLE

ROLL	Subcode	SubName
1	001	DBMS
2	005	SQL
3	006	DS
NULL	070	OB

• From above TWO Tables STUDENT and SUBJECT you can see that in the Table STUDENT ROll = 4 get deleted while the value of Roll = 4 in the SUBJECT table is replaced by NULL.

* This proves that the foreign key can have null values.

* If in the case in SUBJECT Table, column Roll is primary key along with Foreign key then In that case we could not make a foreign key to have NULL values.

Relational Algebra:

* Relational Algebra is a procedural query language, which takes Relation as Input and generates relation as output.

* Relational algebra mainly provides a theoretical foundation for relational databases and SQL.

Operators in Relational Algebra:

Projection(π)

* projection is used to project required column data from a Relation.

Example;

* Suppose we want column A and B from Relation R.

R

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

$\pi_{B,C}(R)$ will show following columns.

B	C
2	4
2	3
3	4

Note; By Default, projection removes Duplicate data.

Selection (σ):

* Selection is used to select required tuples of the Relations.
* for the above relation.

$\sigma(C>3)R$

* will select the Tuples which have c more than 3.

Note: Selection operator only selects the required Tuples but does not display them.

* For display, the data projection operator is used.

* for the above - selected tuples, to display we need to use projection also.

$\pi(\sigma(C>3)R)$ will show following tuples

A	B	C
1	2	4
4	3	4

Union (U);

* Union operation in relational algebra is the same as union operation in set theory, the only constraint is for the union of two relations both relations must have the same set of Attributes.

Set Difference (-)

* Set Difference in relational algebra is the same set difference operation as in set theory with the constraint that both relations should have the same set of attributes.

Rename (ρ);

* Rename is a unary operation used for Renaming attributes of a relation. $\rho(a/b)R$ will Rename the attribute 'b' of the Relation by 'a'.

Cross product (x):

* Cross product between Two Relations let's say A and B, so cross product between A x B will Result in all attributes of A followed by each attribute of B.

* Each record of A will pair with every record of B.

Below is the Example:

A

(Name	Age	sex)
Ram	14	M
Sona	15	F
Kim	20	M

B

(Id,	Course)
1	DS
2	DBMS

A x B.

Name	Age	Sex	ID	Course.
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
kim	20	M	1	DS
Kim	20	M	2	DBMS

Note; If A has 'n' Tuples and B has 'm' tuples then A×B will have n * m tuples.

Natural Join (⋈)

* Natural Join is a binary operator.
* Natural Join between two or more relations will Result set of all combinations of Tuples where they have an equal common attribute.

Example;

Emp

(name	Id	Dept_name)
A	120	IT
B	125	HR
C	110	Sale
D	111	IT

Department

(Dept_name	manager)
Sale	Y
Prod	Z
IT	A

Emp ⋈ Dep

Name	ID	Dept_name	Manager
A	120	IT	A
C	110	Sale	Y
D	111	IT	A

Conditional Join:

* Conditional Join works similarly to Natural Join.
* In natural Join, By default condition is equal between common attributes while in conditional Join we can specify any condition such as greater than, less than or Not equal.

Example:

R

(ID	Sex	Marks)
1	F	45
2	F	55
3	F	60

S

(ID	Sex	Marks)
10	M	20
11	M	22
12	M	59

Join between R and S with condition R.marks >= S.marks

R.ID	R.Sex	R.Marks	S.ID	S.Sex	S.Marks
1	F	45	10	M	20
1	F	45	11	M	22
2	F	55	10	M	20
2	F	55	11	M	22
3	F	60	10	M	20
3	F	60	11	M	22
3	F	60	12	M	59

SQL fundamentals:-

* SQL stands for structured query language.
* It is used for storing and managing data in Relational database Management system (RDMS)
* It is a standard language for Relational Database system.
* It enables a user to create, read, update, and delete Relational databases and tables.
* All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL server use SQL as their standard database language.
* SQL allows users to query the database in a number of ways, using English like statements.

Rules:

SQL follows the following Rules:-

* Structure query language is not case sensitive.
* Generally, keywords of SQL are written in uppercase.

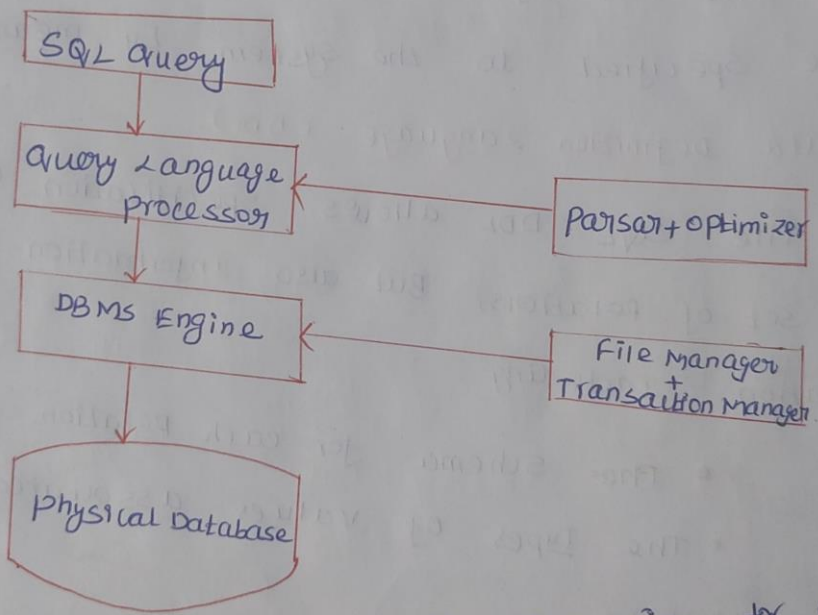
* In the process, various components are Included.
* These components can be optimization Engine, Query engine, query dispatcher, classic, etc.,
* All the non-SQL queries are handled by the classic query engine, But SQL query engine won't handle logical files.

SQL Data Definition:

* The set of relations in a database must be specified to the system by means of a Data Definition Language (DDL).
* The SQL DDL allows specification of not only a set of Relations, But also Information about each Relation, Including:
  - The schema for each Relation.
  - The Types of values associated with each attribute.
  - The Integrity constraints.
  - The set of Indices to be maintained for each Relation.

* Statements of SQL are dependent on text lines we can use a single SQL statement on one or multiple text line.
* using the SQL statements, you can perform most of the actions in a database.
* SQL depends on Tuple relational calculus and Relational algebra.

SQL process:



* When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.

are to the rights of decimal point.

* Real, Double precision; Floating point double precision floating point numbers. with machine Dependent Precision.

* float (n); A floating-point number, with precision of atleast n- digits.

* Each Type may Include a special value called the NULL value.

* The Char data Types store fixed length strings.

Tables;

CREATE TABLE:

* A Table Is combination of rows and columns.

* For creating Table we have to define the Structure of table by adding name to columns, providing data type and size of Data to be stored in columns.

* The security and authorization information for each relation.

* The physical storage structure of each Relation on disk.

Basic Types:

* char(n): A fixed-length character string with user-specified length n. The full form, character, can be used instead.

* varchar(n): A variable length character string with user specified maximum length n. The full form character varying, is equivalent.

* Int: An integer (A finite subset of Integers that is Machine Dependent) The full form integer is equivalent.

* Smallint: A small integer (Machine Dependent subset of the Integer Type).

* Numeric(p,d): A fixed point number with user-specified precision. The number consists of p digit (plus sign), and d of the p digits

Syntax:

```
CREATE table table_name
  (
  Column 1   DataType (size),
  Column 2   DataType (size),
    .
    .
  ColumN   Datatype (size)
  );
```

Here table_name is name of the Table, Column is the name of column.

Example;

Let us create a Table to store data of Subjects, So the table name is Subject, Columns are Sub_ID, Sub_Name.

```
CREATE TABLE Subject
  (
  Sub_ID INT,
  Sub_Name varchar(20)
  );
```

Here INT and varchar are dataType, DataType means Type of data we can store, we can store Integer Type data in the column.

Add data in TABLE

To add data in Table, we use INSERT INTO, the syntax is as shaon below.

Syntax:-

// Below query adds data in specific column, (like column 1 = value 1)//

```
Insert into Table_name (column 1, column 2, column 3)
Values (value 1, value 2, value 3);
```

// Below query adds data in Table in seaquence of column (value 1 will be added in column 1 and so on)//

```
Insert into Table_name
Values (value 1, value 2, value 3);
```

// Adding multiple data in the Table in one go//

```
Insert into Table_name
Values (value 01, value 02, value 03)
       (value 11, value 12, value 13),
       (value 21, value 22, value 23),
        :
       (valueN1, valueN2, valueN3)
```

Example query;

This query will add data in the Table named Subject.

```
// Adding data in first row //
Insert into subject
Values (1, 'English');

// Adding data in specific column //
Insert Into subject (sub_name)
Values ('Hindi');

// Adding multiple data in the Table in one go //
Insert into subject
values (1, 'English'),
(2, 'French');
(2, 'science');
(2, 'Maths');
```

Foreign key constraint in SQL;

* Foreign key is a column that refers to the primary key / unique key of other Table.

* So It demonstrates relationship between Tables and act as cross reference among them.

* Table In which foreign key is define is called foreign Table / Referencing Table.

* Table that defines primary / unique key and is referenced by foreign key is called primary Table / Master Table.

* It is Defined in create table / Alter Table command (or) statement.

Properties; (8) 2mark.

* Parent that is being referenced has to be unique / primary key.

* Child may have duplicates and nulls.

* Parent record can be deleted if no child exists.

* Master Table cannot be updated if child exists.

* Must reference PRIMARY KEY in primary Table.

34

* Foreign key column and constraint column should have Matching data Types.

* Records cannot be Inserted in child Table If corresponding record in Master Table do not exist.

* Records of Master Table cannot be deleted if corresponding records in child Table exists.

* SQL Foreign key At Column Level;

Syntax:

```
Create Table People (no int references person
                     fname varchar2(20));
```

OR

```
Create Table people (no int references person(id),
                     fname varchar2(20));
```

* Here person Table should have primary key with Type int.

To check the constraint;

- If parent Table doesn't have primary key.

output;

Error at Line 1: referenced table does not have a primary key.

- If parent Table has primary key of Different DataType.

OUTPUT:

Error at Line 1: Column Type Incompatible with referenced Column Type.

2. SQL Foreign key At table Level:

Syntax:

```
Create Table People (no varchar(10),
        Fname  varchar2(20), foreign key (no)
    reference person);
```

OR

```
(create Table people (no varchar2(10),
        fname varchar2(20), foreign key (no)
    references person (id));
```

3. Insert operation in Foreign key Table:

* If corresponding value in foreign Table doesn't exists, a record in child table cannot be inserted.

Output;

Error at line 1: Integrity constraint violated - Parent key not found.

5. Foreign key with ON DELETE CASCADE:

* The Default behavior of foreign key can be changed using ON DELETE CASCADE.
* When this option is specified in foreign key Definition.
* If a record is deleted in master table, all corresponding record in detail table will be deleted.

Syntax:

Create Table People (no Varchar2(10), fname Var char2(20), foreign key(no) references person on delete cascade);

* Now deleting records from person will delete all corresponding records from child Table.

Output;

Select * from person;

no rows selected.

Select * from People

no rows selected.

6. Foreign Key With ON DELETE SET NULL:

* A foreign key with SET NULL ON DELETE means If record in parent table is deleted, corresponding records in child table will have foreign key fields set to NULL.

* Records in child table will not be deleted.

Syntax:

Create Table People (no varchar2(10), fname varchar2(20), foreign key (no) references person on delete set null);

Output;

Select * from person;

no rows selected

Select * from People

NO fname.

Par.

38 38/39

* The field "NO" in People Table that was referencing primary key of person table.
* on Deleting Person data, it will set null in child Table People.
* But the record will not be deleted.

SQL Update Behaviors:

* The UPDATE statement in SQL is used to Update the data of an existing Table in DataBase.
* we can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Basic Syntax:

```
UPDATE table_name SET column1 = value1, column2 =
value2,... WHERE condition.
```

table_name : name of the Table.
column1 : name of first, second, third column...
value1 : new value for first, second, third column...
condition ; condition to seled the rows for which the values of columns need to be update.

NOTE: In the above query the SET statement is used to set new values to the particular column and the WHERE clause is used to select the rows for which the columns are needed to be updated.

* If we have not used the WHERE clause then the columns in all rows will be updated.

* So the WHERE clause is used to choose the particular rows.

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	Ram	Delhi	xxxxx	18
2	Ramesh	Gurgaon	xxxxx	18
3	sujit	Rohtak	xxxxx	20
4	Suresh	Delhi	xxxxx	18
3	Sujit	Rohtak	xxxxx	20
2	Ramesh	Gurgaon	xxxxx	18

41

Note: For updating multiple columns we have used comma (,) to separate the names and values of two columns.

Omitting WHERE clause:

* If we omit the WHERE clause from the update query then all of the rows will get updated.

```
UPDATE Student SET NAME = 'PRATIK';
```

Output:

The Table Student will now look like,

ROLL_NO	NAME	ADDRESS	PHONE
1	PRATIK	Delhi	xxxx
2	PRATIK	GURGAON	xxxx
3	PRATIK	ROHTAK	xxxx
4	PRATIK	Delhi	xxxx
3	PRATIK	ROHTAK	xxxx
2	PRATIK	GURGAON	xxxx

41

Example Queries:

Updating single column: Update the column NAME and set the value to PARTIK in all the rows where Age is 20.

UPDATE Student SET NAME = 'PRATIK' in all rows where age is 20.

UPDATE Student SET NAME

OUTPUT;

This query will update two rows (Third row and fifth row) and the Table Student NOW.

ROLL_NO	NAME	ADDRESS	PHONE
1	PRATIK	SIKKIM	x xxxx
2	RAMESH	GURGAON	x x xxx
3	PRATIK	ROHTAK	xxxxx
4	SURESH	Delhi	x xxx
3	PRATIK	ROHTAK	x xxx
2	RAMESH	GURGAON	xxx

42

Intermediate SQL:

Join expression:

* We Introduced the Natural Join Operation.

* SQL Provides other forms of the Join Operation, Including the ability to Specify an explict Join predicate, The ability to Include in the result tuples that are excluded by Natural Join.

Join conditions:

Consider the following query, which has a Join expression containing the on condition.

```
Select * from student join takes on
student.ID = takes.ID.
```

Outer Joins:

* The following SQL query may appes to retrieve the required Information.

```
Select * from student Natural Join
takes.
```

* The left outer join preserves tuples only in the relation named before (to the left of) the left outer join operation.

* The right outer join preserves tuples only in the relation named after (to the right of) of the right outer join operation.

* The full outer join preserves tuples in both relations.

Views:

* Views operated at the logical-model level.

query;

Create view v as <query expression>;