**SYNONYM:** synonym is database object which can be created as an “alias”

for any object like table, view, procedure etc.

* If we apply any DML operations on synonym the same operations

automatically effected to corresponding base table and vice versa.

* If we create a synonym, the synonym will be created on entire table. It is not possible to create the synonym on partial table.
* When we create synonym based on another the new synonym does not allow us to perform any DML operations because Synonym chaining is not

allowed.

* Synonym will become invalid into two cases,

1. When we drop the base table

2. When we change the base table name

* On invalid synonym we cannot apply any DML operations and we cannot

create synonym based on more than one table at a time.

* When we change the structure of the base table the corresponding synonym automatically reflected with same changes.
* But, if we change the structure of the synonym that is not reflected to the

base table because we cannot change the structure of the synonym.

* **Syntax:** Create synonym <synonym name> for <object name>
* Ex: Create synonym synemp for employee

**Syntax to drop a synonym:** Drop synonym <synonym name>

* Ex: Drop synonym synemp

**TO SHOW ALL TABLES IN CURRENT DATABASE:**

select **TABLE\_NAME** FROM **sampledb** .INFORMATION\_SCHEMA.TABLES

OR

SELECT\*FROMsys.tables

**Syntax to Creating a table from an existing table:**

we can create a table from an existing table and maintain a copy of the actual table before manipulating the table.

* Syntax: Select \* into <New Table Name> from <Old Table Name>
* Ex1: Select \* into New\_Emp from Employee

In this case it creates a table New\_Emp by copying all the rows and columns of the

Employee table.

* Ex2: Select EID, ENAME into Test\_Emp from Employee

In this case it creates a table Test\_Emp with only the specified columns from the

employee table.

* Ex3: Select \* into Dummy\_Emp from employee where 1=2

In this case it creates the Dummy table without any data in it.

**Copying data from one existing table to another table:**

We can copy the data from one table to another table by using a combination of

insert and select statement as following

* Syntax: Insert into <Dummy Table name> select \* from <Table Name>
* Ex: Insert into Dummy\_Emp select \* from Employee

**Constraint in SQL:**

**Why Constraint in SQL:** Constraint is using to restrict the insertion of

unwanted data in any columns. We can create constraints on single or multiple

columns of any table. It maintains the data integrity i.e. accurate data or original

data of the table.

Data integrity rules fall into three categories:

* Entity integrity
* Referential integrity
* Domain integrity

**Entity Integrity:** Entity integrity ensures each row in a table is a uniquely

identifiable entity. You can apply entity integrity to a table by specifying a

PRIMARY KEY constraint.

* Ex: the Product ID column of the Products table is a primary key for the table.

**Referential Integrity:** Referential integrity ensures the relationships

between tables remain preserved as data is inserted, deleted, and modified. You

can apply referential integrity using a FOREIGN KEY constraint.

* Ex: The ProductID column of the Order Details table has a foreign key constraint applied referencing the Orders table. The constraint prevents an Order Detail record from using a ProductID that does not exist in the database.

Also, you cannot remove a row from the Products table if an order detail references the ProductID of the row.

**Domain Integrity:** Domain integrity ensures the data values inside a

database follow defined rules for values, range, and format. A database can enforce

these rules using a variety of techniques, including CHECK constraints, UNIQUE

constraints, and NOT NULL constraints.

**The following list gives a sampling of domain integrity constraints**.

* A product name cannot be NULL.
* A product name must be unique.

**There are 5 types of constraints in SQL Server:-**

1. Unique Key constraint.

2. Not Null constraint.

3. Check constraint

4. Primary key constraint.

5. Foreign Key constraint.

1. Unique Key:- Unique key constraint is use to make sure that there is no

duplicate value in that column. Both unique key and primary key both enforces the

uniqueness of column but there is one difference between them unique key

constraint allow null value but primary key does not allow null value.

In a table we create one primary key but we can create more than one unique key

in Sql Server.

Ex: create table EMP(EID int unique,ENAME

varchar(50) unique,SALARY money);

2. Not null constraint: - Not null constraint is used to restrict the insertion

of null value at that column but allow duplicate values. Not null constraint is using

for that column which is not ignorable.

Ex: create table EMP(EID int not null,ENAME

varchar(50) not null,SALARY money);

3. Check Constraint: - This constraint is using to check value at the time of

insertion like as salary of any employee is always greater than zero. So we can

create a check constraint on employee table which is greater than zero.

Ex: create table emp4(eno int,ename

varchar(50),age int check (age between 20 and

30))

4. Primary Key:- Primary key is a combination of unique and not null which

does not allow duplicate as well as null values into a column. In a table we create

one primary key only.

Ex:create table emp(EID int primary key,ENAME

varchar(50),SALARY money)

Adding primary key after creating of table: When we adding

primary key in any table before adding primary key we have to create that column

is not null if the column is not null then we have to create not null constraint on the

column after that we will create primary key constraint on that table.

Syntax:-Alter table <table name> alter column <column name> <data type> not null

Alter table <table name> add constraint <constraint name> primary key (column

name)

Eg:-Alter table EMP alter column EID int not null

Alter table EMP add constraint pk\_Emp primary key (EID)

5. Foreign Key: - One of the most important concepts in database is creating

relationships between database tables. These relationships provide a mechanism

for linking data stored in multiple tables and retrieving it in an efficient manner.

In order to create a link between two tables we must specify a foreign key in one

table that references a column in another table.

Foreign key constraint is used for relating or binding two tables with each other

and then verifies the existence of one table data in the other.

To impose a foreign key constraint we require the following things.

We require two tables for binding with each other and those two tables must have a

common column for linking the tables.

To create Department Table (PARENT TABLE):-create table Department(Deptno int primary key,DNAME

varchar(50),LOCATION varchar(max))

Insert Records Into Department Table:

insert into Department values(10,'Sales','Chennai')

insert into Department values(20,'Production','Mumbai')

insert into Department values(30,'Finance','Delhi')

insert into Department values(40,'Research','Hyderabad')

To create Employee Table(CHILD TABLE):-create table Employee(EID int,ENAME varchar(50),SALARY money,Deptno int

foreign key references Department(Deptno))

Insert Records Into Department Table:

insert into Employee values(101,'Sai',35000,10)

insert into Employee values(102,'Pavan',45000,20)

insert into Employee values(103,'Kamal',74000,30)

insert into Employee values(104,'Ravi',58000,40)

The below records are not allowed in to employee table:

insert into Employee values(105,'Kamal',74000,50)

insert into Employee values(106,'Ravi',58000,60)