

SQL: add, modify, rename, drop.
↑

- 1) DDL - Create, alter, rename, drop.
- 2) DML - Insert, update, delete, Merge
- 3) DQL - select Stmt.
- 4) DCL - Grant, revoke.
- 5) TCL - Commit, rollback, Savepoint, Truncate.

DDL:

- 1) create table tab-name (parameter1, parameter2 ...);
- 2) Alter - add, modify, rename, drop.
 - (i) alter table tab-name add (column1, column2, ...);
 - (ii) alter table tab-name modify (column1, column2 ...);
Note: Modify only possible when the column is empty.
 - (iii) alter table tab-name rename column old-column-name to New-column-name;
 - (iv) alter table tab-name drop (col1, col2 ...); or drop column column-name;
↳ drop more than 1 column ↓ dropping only one column.
- 3) rename old-table-name to new-table-name;
- 4) drop table tab-name;

DML: 1) insert into tab-name values (1, 2, ...);

high level Insert: insert into tab-name (select col1, col2, from tab-name);

2) update tab-name set column1 = value1, column2 = value2, ...;

high level Update: update tab-name set column1 = value1; (with out where clause)

3) delete tab-name; (high level delete), with where clause deletes desired rows.

DQL: select Stmt (*) → projection operator.

4) DCL:

1) Grant } Permissions (insert, update, delete, select).
2) Revoke

* grant all on tab-name to Public; (will assign all permissions of table to all users in DB).

* revoke update, insert on tab-name from Public;

sharing selected columns to other users: (only insert, update supported).

* grant insert on tab-name (column1, column2, ...) to user1; [Select, delete] → Not supported.

> Grant insert(empno, ename), update(empno, ename, job) on emp to public;

Scott → user1
Privileges

Note: changes made in the user1 reflect in Scott only when commit in user1;

* grant all on emp to public;
↓
revolve all on emp from public;
doesn't work ← user1

grant → Public grant → user1
revolve → Public revolve → user1

(only Public will cancel the privileges where as specifying user1 doesn't work).

TCL: $\begin{array}{l} \uparrow \text{implicit (DDL)} \\ \downarrow \text{Explicit (DML)} \end{array}$

Commit: saving DML operations permanently.

rollback: get back the data of non committed DML operations.

Savepoint: go back to the un committed save point.

rollback to S1

Note: placing Commit, (or) rollback will clear save points

Truncate: Delete + Commit

structure of table is alive but data is lost permanently.

| DDL;
| DDL;
| S1; ←
| DDL;
| DDL;
| rollback to S1;
| ↘ Commit;
| saves first DDL stmts

Operators in SQL:

| Comparison | | | |
|------------|------------|-------------------------------|---------------------------|
| Arithmetic | Relational | Logical | Special operators |
| + | = | AND | IN, NOT IN |
| - | !=, <> | OR | BETWEEN, NOT BETWEEN (An) |
| * | <= | NOT | LIKE, NOT LIKE |
| / | >= | | |
| | > | order of exec: NOT AND OR. | IS NULL, IS NOT NULL |
| | < | | |
| | Some / Any | | |

Truth Tables:

AND:

| AND | T | F | N |
|-----|---|---|---|
| T | T | F | N |
| F | F | F | F |
| N | N | F | N |

$T \rightarrow 1$

$F \rightarrow 0$

NOT:

| NOT | T | F | N |
|-----|---|---|---|
| NOT | F | T | N |

OR:

| OR | T | F | N |
|----|---|---|---|
| T | T | T | T |
| F | T | F | N |
| N | T | N | N |

| |
|--------------------------|
| All comparison operators |
| NOT |
| AND |
| OR. |

$q_1 \text{ AND } q_2 \text{ OR } q_3 \text{ OR } q_4 \text{ AND } q_5 \text{ AND } q_6 \text{ AND } q_7 \rightarrow \text{AND will take the priority.}$

$R_1 \quad R_2$

$R_1 \text{ OR } q_3 \text{ OR } R_2 \text{ (either of the values)}$

Special operators: NOT IN with NULL value returns no rows.

IN, NOT IN (Searches particular value from list of values).

Select * from emp where hiredate NOT IN ('18-JAN-1987,

'20-FEB-2010').

↓
Recognize only oracle format

dd-Mon-yy (or)

dd - Month - yyyy .

Between, Not Between: is always used with 'AND' operator.

Select * from emp where sal Not Between 2000 AND 5000;

My

NOT SAL
Special operator
between 2000 AND 5000; result same
Logical operator.
* between lo-value and hi-value (between should operate b/w
Lo-val & hi-val but not hi-val & lo-val)
as above.

Like, Not Like: like operator used to search for particular

Pattern of characters.

Wild card characters: the char's which are used to search particular pattern of characters is called wild card char's.

2 char's: 1) % (represent any seq of zero or more char's)

2) _ (under score) { represent any single char, only at that position}.

1) Placing % before or after doesn't bother what ever or any no. of char's.

> select * from emp where ename like 'K%.' OR ename like '%T.%'

%ANA%, %KI%, %K , K%,
%T%

2) _ represents single char

> select * from emp where ename not like '___';
'__KI__', '_KI_';

IN, NOT IN:

IS NULL, IS NOT NULL:

> select * from emp where comm is not null;

* Any arithmetic operation with null is null - ~~null~~

Alias: column names in the select list can be aliased with as.

Note: Alias names not recognised in where, group by, having and in order by clause (works while displaying)

Clauses: select ^{distinct} Column1, Column2 from tab where = group by having order by

- 1) distinct
- 2) from
- 3) where
- 4) group by
- 5) having
- 6) order by.

operators of the same priority are evaluated from left to right.

- 7) join clause
- 8) on clause

1) Distinct: Will eliminate all duplicate rows mentioned in the distinct clause.

> Select distinct ename, Job from emp; (suppress the duplicate values in both columns and display that 2 columns).

2) from: placing one or more tables.

Writing a select stmt in from clause called In Line View.

3) where clause: filter the data from the table. supports ^{with} update, delete, ~~insert~~, select but not with ~~insert~~. * doesn't recognise Alias

Select * from tab-name where Arithmetic expression,

Function, column name, constant, list of values.

4) group by clause: used to calculate grouped values by grouping the rows.

> Select Column1, Column2 from tab_name group by Column1, Column2;

* Column Aliases cannot be used in group by clause.

* By default rows are sorted by Ascending order of the columns included in the group by List.

5) Having clause: Making a condition on grouped results. And group functions only exist in the having clause

Having clause doesn't recognise alias names of grouped results. only grouped fⁿs or grouped expressions.

first rows are grouped

Second Group fⁿ is applied to identified groups

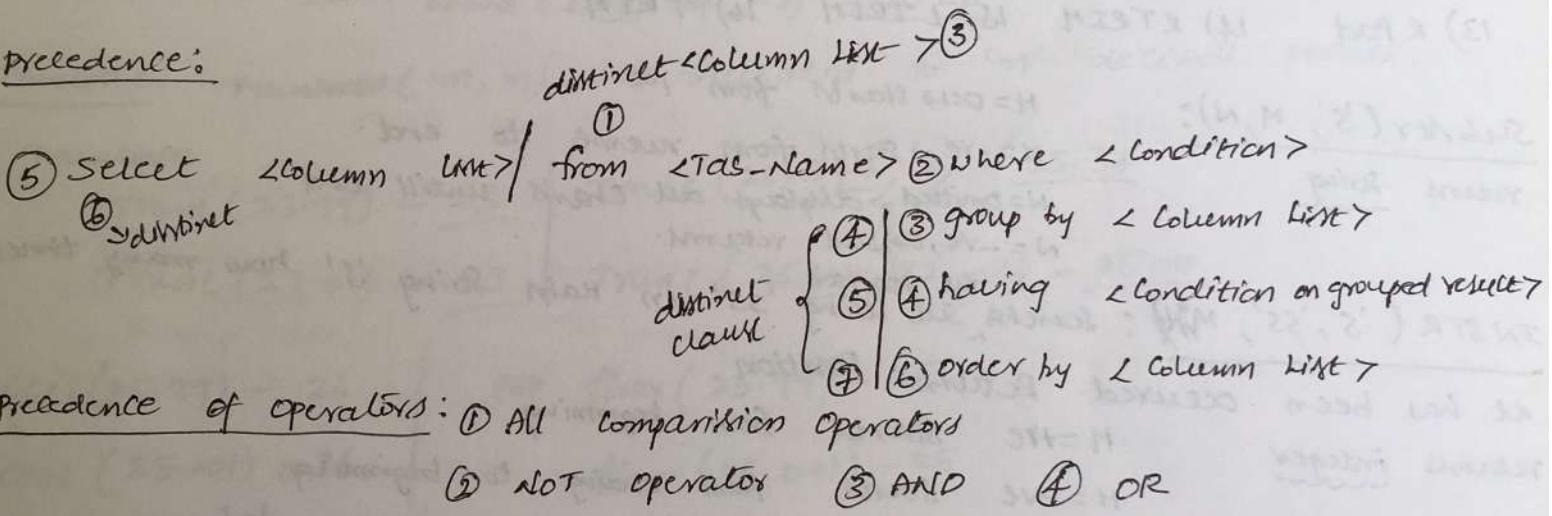
Third groups that match the criteria in the having clause are displayed.

* Existence of group by clause does not guarantee the existence of having clause. But existence of having clause demands the group by clause.

- 6) order by clause: used to sort the list of specified columns, ~~in~~ or grouped columns
 in order by clause either in ascending or descending. can sort by using alias names as well.
- > Select * from emp order by deptno, sal; (first 'deptno' is sorted in ASC then 'sal' will be sorted with it)
 * order by deptno, sal desc; (the dept)
- Select empno, ename, deptno from emp order by 3 (Select List deptno
 " " " order by 5 x (5 columns not there in select list))
- 7) Join:

- 8) ON:

precedence:



FUNCTIONS IN SQL

- 1) Single row fn: Returns single result for every row of a queried table.
 2) Multiple row fn: perform calculation on group of rows

and return one result per group of row.

3) General fn: NVL, DECODE

* Single Row fn: can appear in Select List,

1. char. fn

2. Num fn

3. Date fn

4. Conversion fn.

Where clause

Order by clause

Start with clause, connect by clause.

- ↳ NO of Arguments
- 1) char fn: 1) Lower(1); 2) Upper(1); 3) InitCap(1); 13) RPad
 4) Length(1); 5) Concat(s1, s2); 6) Ascii('char'); 14) RTrim.
 7) Chr(Num); 8) Substr(s, m, n); 9) Instr(s, 'ss', m); 15) LTrim.
 10) Replace('s', 'ss', 'vr'); 11) Translate('s', 'ss', 'rs'); 12) LPad
 16) Trim.

Substr(s, m, n):

returns string

M=0=> starts from 1st

M=-ve=> starts from reverse to end.

N=omitted=> displays all chars until end

N=-ve, 0=NULL. returns.

Instr('s', 'ss', m): searches ^{the position} sub string 'ss' in main string 's' how many times

it has been occurred. Returns Position

returns integer

M=+ve Starts from beginning

M=-ve Starts from ending to beginning.

'N' N of times occurred in main string 's'!

Default M=N=1

If M=N=0 Result is 0

S V1 V2

↓
error.

Replace('KIRAN', 'KI', '123'): 123RAN (Value by value) Replace.

Translate('KIRAN', 'ABC KDI', '1234567'): 46RAN (Char by char Replace)
 ↗ Specifying 'space' will be replaced space with every char.

Lpad: Lpad('Kiran', 15, '*\$'); pads '*' from left in total 15 chars.

My Rpad:

Trim: Trims the specified chars from the supplied string

TRIM(' Kiran '); empty spaces get trimmed.

TRIM('KK')

> Trim (Leading 'Ki' from 'KiKiKiran'); Trim leading 'Ki'!

> Trim (Trailing 'Ki' from 'KikiranKiKi'); ending 'Ki'.

> LTRIM (' ABABABCDABB', 'AB'); 'AB' is trimmed from left side.

> RTRIM ('ABAB.CDEFABAD', 'AB'); " " right side

Number fn': 1) Round 3) Floor 5) Power(m,n) 7) Sqrt(n) 9) Sign(n)

2) Truncate 4) Ceil 6) mod(m,n) 8) Abs(n) 10) Sin(30°)

Round(m,n) - 'm' truncated to 'n' decimal points. ($n+ve$ after decimal point
 $n-ve$ before decimal point).

Round(20.49) - 20 Round(20234.5643, 2) - 20234.56

Round(20.50) - 21 Round(2034.214, -2) - 20 00 → Rounding zeros.

Truncate: Truncate(m,n) - 'm' truncated to 'n' decimal points.

Truncate

Trunc(23.99) - 23

Trunc(25.345, 2) - 25.34

Trunc(23.01) - 23

Trunc(2564.342, -2) - 2500

Ceil(25.99) - 26 floor(25.99) - 25

Ceil(25.001) - 26 floor(25.001) - 25

Date functions: DD-MON-YY → standard format.

Date + number = Adding no of days to the date

Date - number = Subtract no days from date.

Date - date = Returns no of days between two dates.

Date + N/24 = Adding 'N' hours to that particular date

Days → Weeks days/7

Days → months days/30

months → years months/12

- 1) Add-months(D, ± N) - Add or sub 'N' no of months to that particular date.
- 2) Months-between(d1, d2) - Returns no of months between d1, d2.
- 3) * Next-day(date, 'char'/mon') - Returns date of coming day.
- 4) * Last-day(date) - Returns last date of that particular month.

Rounding, Truncating on Dates:

Round(date, 'dd') - check for nearest time of that day (Before PM OV) after PM
Round(date, 'day') - takes to the nearest Sunday

Round(date, 'month') - checks for $\frac{\text{days}}{<15(\text{or}) >15} \rightarrow$ $\begin{cases} <15 & \text{takes to the 1st of same month} \\ >15 & \text{takes to the 1st of next Month.} \end{cases}$

Round(date, 'year') - checks for months $\frac{<6(\text{or}) >6}{<6 & >6} \rightarrow$ $\begin{cases} <6 & \text{takes to the 1st day, 1st month of same year} \\ >6 & \text{" 1st day, 1st month of next year} \end{cases}$

Trunc(date, 'day'): takes back to the last Sunday.

Trunc(date, 'month'): takes back to the 1st of same Month.

Trunc(date, 'year'): " " " 1st day, 1st Month of same year.

Conversion Functions: 1) to_date(): used to convert character to date format
to-date(char, 'format') format is optional. However it will display as per the supplied format.

2) to-char(date*, format e'): converts date to character format.

to-char('13-may', 'dd-mon'), X
 ↓
 First convert using to-date

3) to-number('char'); converts number char's to number. Then to-char.

to-number('123') ✓

to-number('as12') X

| DD - MON - YY | | | | quarter | spellout | |
|---------------|-------|------|----------------|--------------|--------------|-------|
| DD | MON | YY | HH:MI:SS AM | CC | ↑ Q | WW SP |
| dy | MONTH | YYYY | hh24 | ↓ Century | ↓ Week of | Year |
| day | MM | Year | hh12 (default) | | | |

> display the employees who joined 2 years ago.

Spell numeric:

Select to-char(to-date('12345', 'JSP'), 'JSP') from dual;

Julian spell out.
 MAX = 14 million (101).

2) Multiple Row Fn's (or) Group Fn's: these fn's can appear in Select List, HAVING clause and order by clause only *

[group-function-name(distinct / All columns)]

* distinct makes the fn to consider along duplicate values.

* All " " " All values.

* All group fn except ~~Count~~ will ignore null values. except Count(*).

* When group fn is declared in select list remaining columns can also be declared but they should be placed in group by clause.

- | | | |
|------------------------------|-------------------------|-------------------------------|
| 1) sum (distinct / All) | 3) min (distinct / All) | 5) count (* / distinct / All) |
| 2) max (distinct / All) | 4) avg (distinct / All) | 6) stddev (distinct / All) |
| 7) variance (distinct / All) | | |

Pg: 77 (work out)

- 3) General Fn's:
- | | |
|---|--|
| 1) NVL | 4) least (m ₁ , m ₂ , ..., m _n) vid. |
| 2) Decode | 5) size (column) |
| 3) greatest (m ₁ , m ₂ , ...) 6) user | |

> select user, vid from dual;

1) NVL (column, \downarrow data-type matching value);

2) decode (column, decode): used to check for multiple condition (Like if else).

select deptno, decode (deptno, 10, sal*1.5, 20, sal*2.5, sal-1000) from emp;
If Then Then Else (last one).
↓

Specifying more than one value in else results null value.

* Data type should match in then, else.

Case Statement: case stmt supports comparison operators. (decode doesn't)

select sal, job, case when sal >= 800 AND job = 'CLERK' then 'Kiran'

when sal = 5000 then 'First'

when deptno = 10 then 'Ten'

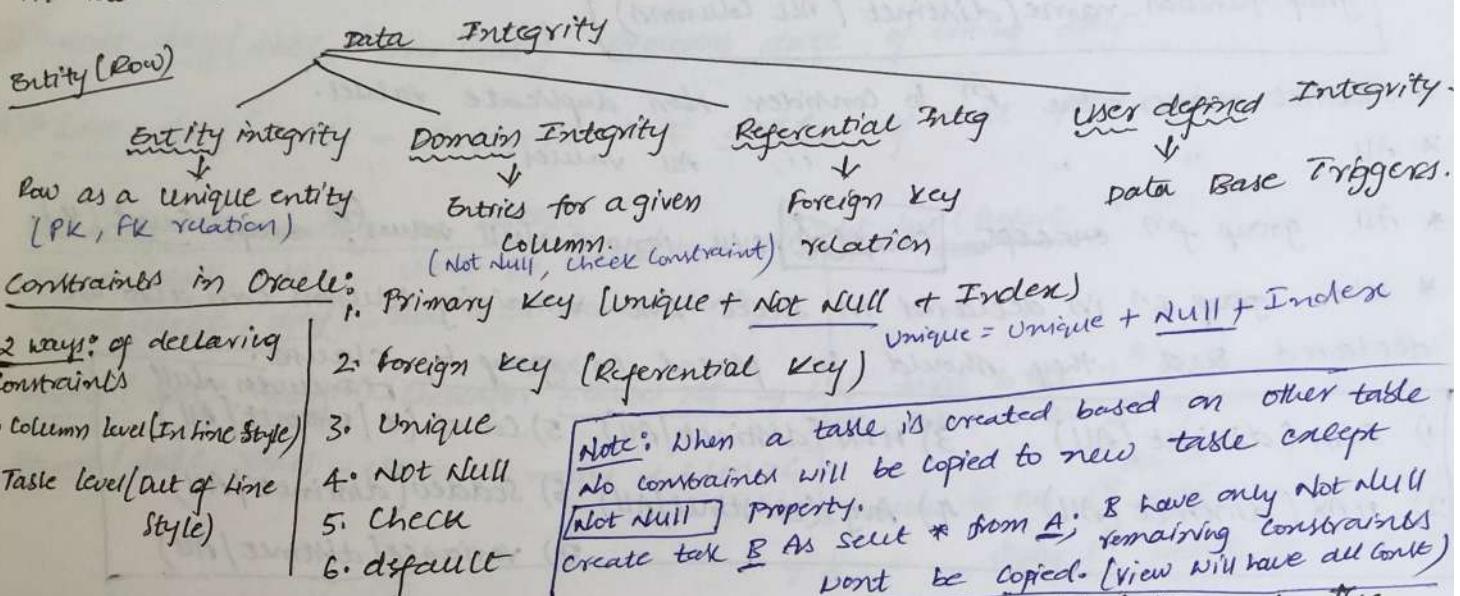
else 'Last'

end case from emp;

Then part can be either varchar (or) numeric but all then's should be in sync.

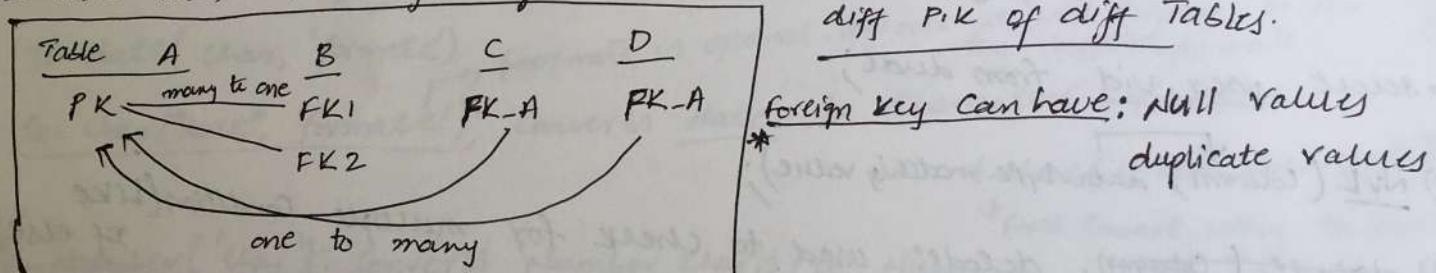
Constraints: (Conditions)

Data Integrity: It is a state in which all the data values stored in the database are correct.



Primary Key: The values of P.K columns should be unique, Not Null and the key column will be indexed.

foreign Key: forming a relation with P.K of other Table. one Table can have more than one foreign keys. And thus f.k can referenced to one P.K or diff P.K of diff Tables.



on delete cascade: when trying to remove records from 'parent table' automatically records of the 'dependent table' will be deleted.

> create table b (no number(10) references a (no) on delete cascade);

Removing rows of 'A' will automatically deletes 'B' rows.

delete (or) drop will throw error depending child records exists.

cascade Constraints: Allows to remove parent table even child records exists

drop table A cascade Constraints;

data is lost in depending table B, C. It will destroy relationship b/w child tables. But data is not removed from depending tables.

unique: columns specifies as unique, the values should be unique.

> create table A (no number(10) unique); \rightarrow nulls allowed

not null: null values not allowed for specified column with not null. Default will be null property.

> create table A (no number(10) not null);

default: whenever the value not specified or default passed as value then default value will be taken.

> create table A (no number(10) default 100); $\left| \begin{array}{l} \text{null value passed as value} \\ \text{then overrides the default value.} \end{array} \right.$
A (name varchar(10) default 'abc');

More than one constraint for single column: At a time declaring

> create table A (no number(10) referenced A on delete cascade unique not null);

check: checks the condition before taking the values of check column.

> create table A (no number(10) check (no > 10 and no < 100));

Composite Primary Keys, Composite foreign keys: (max 32 columns)

Defining more than one column in Table Level is called CPK (or) CFK.

> create table A (no number(10), no1 number(10), no2 number(10) unique not null,
constraint A-PK Primary key (no, no1));

> create table B (no number(20), no1 number(20), loc varchar(20) default 'NA',
constraint B-FK foreign key (no, no1) references A on delete cascade);

Note: A composite F.K can be declared only with the composite Primary Key. Ex: In above ex Table A C.P.K has 2 columns so C.F.K should also declare with 2 columns referenced to that table.

Self reference Key: A table contains foreign key referenced to the same table Primary Key.

> create table A (no number(10), no1 number(20),

constraint A-PK primary key (no), (make sure commas)*

constraint A-FK foreign key (no1) references A on delete cascade)

Table level declaration

Adding Constraints: with 'Alter' (PK, FK, Unique)

> Alter table tab-name add constraint const-name Primary key(list);
> " " unique (list)
> " " foreign key(list) references tab-name on delete cascade;

Ex: Alter table A add constraint A-FK foreign key(NO) references B on delete cascade;

Adding Properties (Not NULL, default): (for removing do reverse).

For existing Table: ① desc tab-name
② Take the size of the column to be modified.

③ Alter table tab-name modify colu-name datatype default 'value' Not Null;

Ex: desc A
no number(7, 2)

> Alter table A modify No(7,2) default 20 Not Null;*

Enabling, disabling, dropping Constraints:

> Alter table A disable/enable/drop constraint const-name;
a-PK, a-FK

System Tables (Constraints Existing):

- | |
|---|
| 1) <u>User_Constraints</u> . (constraint-name, constraint-type are columns of that table) |
| 2) <u>User_Const_Columns</u> . |

User_Constraints: holds the complete details of constraints defined on Tab names.

- > Select * from User_Constraints where table-name = 'TAB-Name';
> Select constraint-name, constraint-type from User_Constraints where Table-Name = 'EMP';

User_Const_Columns: only brief information about the constraints on table column.

- > Select * from User_Const_Columns where Table-name = 'EMP'; (Capital).

JOINS: Joining the rows of two (or) more tables, horizontally. Based on ~~in~~ the condition. Multiple table should be placed in 'from' clause.

views, materialized views
Note: Join b/w numer, varchar
will work. Join b/w Null values
will be ignored.

1) Inner Join (Natural Join, Equi Join).

2) Left Outer Join

3) Right Outer Join

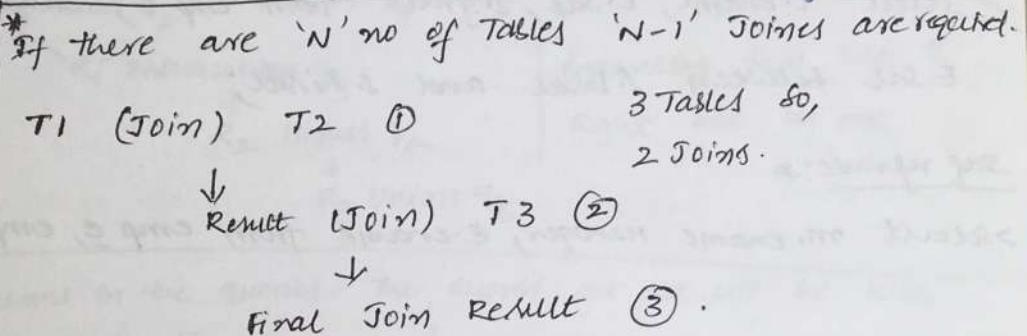
4) FULL Outer Join

5) Self Join.

6) Cartesian Join.

7) Non equi Join

| T1 | T2 |
|----|----|
| 10 | 10 |
| 20 | 20 |
| 30 | 40 |



$$\text{Inner Join: } (T1 \cap T2) = \{10, 20\}$$

$$\text{Left Outer Join: } T1 \cap (T1 \cup T2) = \begin{Bmatrix} 10 \\ 20 \\ 30 \end{Bmatrix} \cap \left[\begin{Bmatrix} 10 \\ 20 \\ 30 \end{Bmatrix} \cup \begin{Bmatrix} 10 \\ 20 \\ 40 \end{Bmatrix} \right] = \{10, 20, 30\}$$

$$\Leftrightarrow \{10, 20, 30, 40\}$$

Right Outer Join: $T2 \cap (T1 \cup T2)$: $\begin{Bmatrix} 10 \\ 20 \\ 40 \end{Bmatrix} \cap \left[\begin{Bmatrix} 10 \\ 20 \\ 30 \end{Bmatrix} \cup \begin{Bmatrix} 10 \\ 20 \\ 40 \end{Bmatrix} \right] = \{10, 20, 40\}$.

$$\Leftrightarrow \{10, 20, 30, 40\}$$

Full Outer Join: $(T1 \cup T2)$: $\begin{Bmatrix} 10 \\ 20 \\ 30 \end{Bmatrix} \cup \begin{Bmatrix} 10 \\ 20 \\ 40 \end{Bmatrix} = \{10, 20, 30, 40\}$.

Joining 3 Tables:

Select * from A Inner Join B ON (A.col = B.col) Inner Join C ON (A.col = C.col)

① Select * from emp Natural Join Dept Using (deptno);
 ② Select * from emp Natural Join Dept. Join will find the common column name.
Using clause: Using clause will take common column b/w the 2 tables

And the name should be same else through error.

③ Select * from A, B, C where (A.col = B.col) AND (A.col = C.col);

Cartesian Join: Rows of one table will be multiplied with all the rows of other table.

| T1 | T2 |
|----|----|
| 4 | 5 |

$$\frac{T1 \times T2}{20 \text{ rows.}}$$

2x5 = 8x5 ✓

Self Join: Same table appears twice in the from clause with corresponding Alice names

Non Equi Join: It is a join condition when no column in one table corresponds directly to a column in the other table.

> Select e.ename, e.sal, s.grade from emp e, salgrade s Where e.sal between s.lsal and s.hsal;

Say reference:

> Select m.ename Manager, e.ename from emp e, emp m where m.mgr = e.empno;

| | |
|-------|---|
| emp | e |
| ename | |

| | |
|-------|---|
| mgr | m |
| ename | |

SET OPERATORS: Used to combine the columns of select stmt's based up on the operator. Here the columns are joined vertically.

- 2) The select stmts that are involved in joining should have same no. of columns and corresponding data types must be matched.
- 3) Max 32 queries can be joined using set operators.

Compound Queries: SQL stmts containing set operators are called compound queries.

Component query: Each SQL stmt involved in Compound query.

- 4) The two select stmt cannot have the order by clause the final result of the entire set operation can be ordered.

$$A = \{1, 2, 2, 3, 4, 5\} \quad B = \{1, 2, 3, 3, 4, 5\}$$

- ① Union All: Combines elements of both sets including duplicates ($5+5=10$).

$$A \cup B = \{1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5\}_{10}$$

- ② Union: Combines elements of both records and removes duplicate values.

$$A \cup B = \{1, 2, 3, 4, 5\}$$

- ③ Intersect: Combines elements of both records which are common to both sets and duplicate common values will be removed. Ans $\{1, 2, 3, 4\}$

* no duplicates are allowed in the op set except union all.

④ Minus: Returns unique values of 'A' that are removed from B.
Duplicate values of set 'A' not allowed.

| A | B | $A - B = \{3\}$ (unique value w.r.t B') |
|----|---|--|
| 1x | 1 | |
| 1x | 2 | $q_1 \cup q_2 \text{ intersection } q_3 \text{ Minus } q_4 \cup q_5$ |
| 2x | 2 | $R_1 \text{ intersection } q_3$ |
| 2x | 5 | \downarrow |
| 3 | 4 | $R_2 \text{ Minus } q_4$ |
| 4x | 6 | \downarrow |
| 5x | | $R_3 \cup q_5$ |
| 3 | | |

Executing from Left to
Right one by one.

NOTE: When taking multiple columns in the queries, the query's output will be with distinct values of the set operator.

| q_1 | X | q_2 | Y | $q_1 \cup q_2$ | $AB \cup XY$ | $A \times X$ <small>wrong:</small> | Here: Actually 'A' has to display 1, 2, 3 |
|-------|---|-------|----|----------------|--------------|------------------------------------|--|
| 1 aa | | 1 | aa | | | 1 aa | But |
| 1 bb | | 2 | bb | | | 2 bb | |
| 2 cc | | 2 | bb | | | 2 cc | |
| 3 dd | | 3 | ee | | | 3 dd | \rightarrow Extra value taking corresponding value |
| | | | | | | 3 ee | |

$A \cup B$

$A \cup Y$

Makes the union operation on both columns with possible combinations and removes duplicates. (When taking more than one column in select line).

| A | B |
|---|----|
| N | C |
| 1 | aa |
| 1 | bb |
| 2 | cc |
| 3 | dd |

? select N, C from A
union
Select N, d from B;

O/p: $A \cup B$ first make union on N, M \rightarrow N
Then take possible combinations of C, d \rightarrow C

| N | C |
|---|--|
| 1 | aa (check aa with 1 or anything else) |
| 2 | bb (check bb with 2 and found bb with 1 as well) |
| 3 | dd similarly rest |
| 1 | bb |
| 2 | cc |
| 3 | dd |

VIEWS: A view is a data base object which is holding a select statement in it.

- View is stored in the logical memory. (Stored in 'User-views')
>> create view view-name AS select * from Table-name; (Same as copying a table).

* DML performed on view will be reflected on Table and vice versa.

* It is used for sharing selected rows & columns with other users & it will improve performance while manipulating or retrieving data through views.

Sharing Views:

grant all on VI to User1

Permissions:

| | |
|--------|--------|
| Insert | Select |
| Update | |
| Delete | |

DML, DQL

Droping a view:

drop view VI;

(only view is destroyed no reflection on Table).

To see Existing views:

select * from user-views;

select * from Tab;
Tables + views

'with check option' clause: Used to check for condition in the Where clause while inserting the values into the Table.

Ex: ① create or replace view VI AS select * from emp where deptno=10;
> insert into VI values(1100, ..., 20); (With out check option, it will be error - view with check option - where clause violation).
 ↑ Not allowed

② create or replace view VI AS select * from emp where sal>1500 with
> insert into VI values(1122, ..., _{sal}1000, 1000, 10);
 ↑ Check option;
 ↑ error (Where clause violation).

View Based on View:

> create view V2 AS select * from VI;

→ DML performed on V2 will be reflected in VI and VI holding Table.
DML " " Table " " VI and V2 views.

Read only option:

> create view VI AS select * from emp with Read only; (No DML's allowed on view).

NOTE: Altering the table will lead to db's corresponding views destroyed i.e view doesn't open saying error (view with errors). DDL on tables \rightarrow views Invalid.
* Alter add columns doesn't reflect but existing columns dropped throws error.

* Alter not allowed on views.

Read only views: views based on Arithmetic Expressions.

> Create or replace view v1 AS select sal S, sal*10 A, sal*20 B from emp;

* View will be created with Alias names.

* Without the alias names, a view cannot be created with expressions.

* Once a view created with alias names, we cannot perform 'insert', ^{update, delete} operation _{on alias column} in to view as the view contains virtual columns (select stat expression with alias name).
~~Insert, update, delete~~ cannot be performed on virtual columns (i.e. alias names because it contains expressions).

* We can make update, delete on Non expression Column of view w.r.t Table.

No Insert, update, delete performed on A (sal*10), B (sal*20) virtual columns with expressions. But update, delete performed on S (sal) it doesn't have expression.

View based on Aggregate functions:

> Create view v1 AS select ~~Max~~(sal), min(sal) from emp group by sal;

Adv: Used for reporting purpose improves performance while retrieving data through views.

View will support constraints automatically:

> Create or replace view v1 AS select ename, job from emp;

- error AS empno in emp when, trying to insert values in view v1.

- A view has been created based on a Table A' when altering the structure of Table A' will reflect on views. throws an error.

Force: It allows to create a view without a table that is not in the Data Base.

> Create or replace force view AS select * from XXYY;
 \hookrightarrow Table that doesn't exist in the DB.
 - view created with compilation errors.

> Select * from v1; - error (view scott.v1 has errors).

Advantage: Used to register the name in the Data Base.

JOIN VIEWS:

Key Preserved Table: The table whose key column is not duplicated in "view result" is known as K.P.T.

> Create view V1 AS select e.empno, e.name, d.deptno from EMP E, DEPT D;

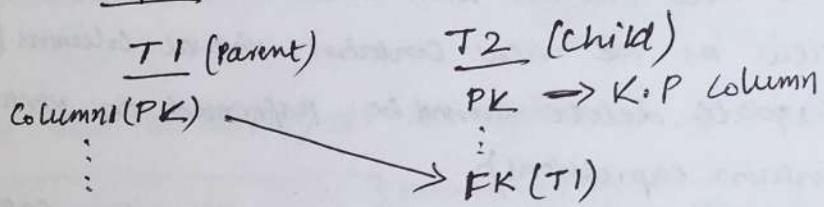
> Select * from V1;

| K.P.T (unique value) | ename | d.deptno |
|----------------------|-------|----------|
| 7900 | | 10 |
| | | 10 |
| | | 20 |
| | | 20 |
| | | 30 |
| | | 30 |

13 rows

Here in this example empno has unique values in the O/P so it will be Key Preserved Table.

But deptno has duplicated repeated.



The primary key, (or) unique key of child table will be key preserved Table. And the O/P of the view will have unique values of that column.

In Join view emp.deptno, dept.deptno not allowed only one column allowed
Because Join view doesn't allow duplicate columns.

1) In Join view only one table can be manipulated i.e. Key Preserved Table.

2) Both the tables of view allowed only through view using "Instead of Triggers" in PLSQL.

update operations performed on Key Preserved Table Columns of view will reflect in the K.P.T.

But operations performed on Non K.P.T Columns doesn't take an effect on its corresponding Table.

Insert supported with K.P.T

not supported with Non K.P.T.

Materialized views: It is a static view (i.e holding data).

- It will not support DML on it.
 - DML on Table not reflected on its view.
- * To create it "create materialized view" permission is required from Admin.
- used to maintain historic data for data analysis & reporting purpose.

System: [Grant create materialized view to slot];

Even the Table deleted the view exists with data.

> create materialized view v1 AS select * from emp;

> drop table emp;

> select * from v1; data exists as of emp (try to copy table).

Synonym: used to hide the original name and owner (user/obj) of the table. It resides in "User-Synonyms".
When

* Provides security by hiding Identity of the component.

* create or replace synonym s ~~AS select * from emp;~~ for x;

DML on Synonym reflected in Table [try view.]

" " Table reflected in synonym [Copied table doesn't reflect the DML on Main Table.]

[Dropping A Table no longer exists of synonym] * try view.

Dropping A Synonym doesn't reflect on Table

> create synonym s ~~FOR emp;~~ (we can ^{make} selected columns as view). created

* diff b/w view & synonym: A view can ^{share} selected rows & columns with other users.

Synonym makes the entire copy of the table & supports to share entire object with other user.

Sequence: used to generate sequence of numbers automatically.

It is a data base object stored in 'user_sequences'.

It uses 2 pseudo columns

(1) nextval

(2) currval

Syntax: create sequence seq_name

increment by $\pm N(-1)$

start with N (5)

MAXvalue M (5)

MINvalue P (-5)

cache integer/no cache (2)

cycle/no cycle (yes)

order/no order (yes)

default values

increment by will be +ve

start with 1

sequence is no cycle

cache will be 20

sequence is order sequence.

O/P: 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5
5, 4 ---- -5.

Alter Seq No:

> Alter sequence seq-name

increment by N

Maxvalue/no Maxvalue (no space)

Minvalue/no Minvalue (no space)

cycle/no cycle

cache N/no cache

Note: No possibility to change start with. only possible is drop.

Index: (User_Indexes):

It is a pointer locates the physical address of data memory location.

- It will improve the performance of oracle while retrieving or manipulating data from Table.

- It is automatically activated when Index column is used in 'where clause'. When Table dropped corresponding Indexes also be dropped.

Types of Indexes:

1) Normal Index (01) Simple Index.

4) Function Based Index.

2) Composite Index

5) Bit map Index.

3) Unique Index

- 1) Normal Index (or) Simple Index: created upon a Table by Considering only one Column.
- > create 'Index' Idx1 on emp(deptno); } When ever 'deptno' used in where clause retrieval will be very speed.
- 2) Composite Index: Index created on multiple columns: MAX 32 Columns Allowed.
- > create 'Index' Idx2 on emp(JOB, deptno, empno);
- 3) Unique Index: indicates the entry of the values to the Index Column. Should be unique. i.e no duplicates allowed.
- > create 'unique index' Idx3 on emp(empno); If the column is having unique property, it cannot be indexed. B'coz it is already unique itself is index.
- Note: Both unique, & Bit map cannot be indexed at a time.
- 4) Function Based Index: Fn based indexes are designed to improve query performance when the function is used in 'where clause'.
- > create Index Idx4 on emp(upper(JOB));
select * from emp where ~~upper~~ upper(JOB) = 'MANAGER';

- 5) Bit map Index: It specifies an Index has to be created with a Bit map for each 'distinct key' in the Table.
- Bit map Index stores the 'Rowids' associated with distinct key values of the column as a Bitmap (1's and 0's).

> select create bit map Index Idx5 on emp(deptno);

| Deptno | Start Row Id | End RowId | Bit pattern |
|--------|--------------|-----------|-------------------|
| 10 | - --- AAA | - --- AAN | 1010 01 |
| 20 | - --- AAA | - --- AAN | 10 - - - 01 |
| 30 | - --- AAA | - --- AAN | 1001 - - - - 10 |

> create bitmap Index Idx6 on emp(mgr);
↑
no space

clusters (user clusters): It holds the common column shared by 2 or more tables.

- It will improve the performance while retrieving or manipulating data from parent, child tables.

Note: It cannot be applied to the existing Table. It has to be created before creating Table.

dropping cluster: drop cluster c_1 ; (drops only cluster)

drop cluster c1 including tables; (cluster, parent, child will be dropped).

Roles: Used to share the ^{group of} 'privileges' of multiple objects with other users easily. Defined by "DBA" only.

- It holds collection of permissions to be shared and stored in "User_Roles"

create role hr; create role hr1; create role hr2;

> grant all on emp to hr;

>grant all on dept to hr; >grant select,insert on salgrade to hr;

> grant hr to user1; >grant hr, hr1 to hr2;

> revoke select on salgrade from hr;

dropping role: drop role by;

Joins:

| TABLE A | |
|---------|-------|
| 1 | { (3) |
| 2 | { (2) |
| 3 | → (1) |
| 4 | → (1) |
| 5 | → (1) |

TABLE B

| | |
|----|-----|
| 1 | (2) |
| 2 | (2) |
| 3 | (2) |
| 2 | |
| 3 | |
| 10 | (1) |
| 11 | (1) |
| 12 | (1) |

Inner Join: (Matching values of both tables)

1, 2, 3,

| A | B |
|-------|-------|
| 1 (3) | 1 (2) |
| 2 (2) | 2 (2) |
| 3 (1) | 3 (2) |

O/P values

$$3 \times 2 = 6$$

$$2 \times 2 = 4$$

$$1 \times 2 = 2$$

Total - 12 (Rows)

Left outer Join: A B (Matching & non matching of left table & matching of right table)

| | |
|-------|-------|
| 1 (3) | 1 (2) |
| 2 (2) | 2 (2) |
| 3 (1) | 3 (2) |
| 4 (1) | |
| 5 (1) | |

non matching

Total Rows =

$$(1) 3 \times 2 = 6$$

$$(2) 2 \times 2 = 4$$

$$(3) 1 \times 2 = 2$$

$$(4) 1 = 1$$

$$(5) 1 = 1$$

(14) Rows.

Right Join:

| | | |
|--------|-------|-----|
| 1 (3) | 1 (2) | = 6 |
| 2 (2) | 2 (2) | = 4 |
| 3 (1) | 3 (2) | = 2 |
| 10 (1) | | = 1 |
| 11 (1) | | = 1 |
| 12 (1) | | = 1 |

15 Rows.

| <u>Full Join:</u> | <u>A</u> | <u>B</u> | |
|-------------------|----------|----------|-----------------|
| | 1 (3) | 1 (2) | = 6 |
| | 2 (2) | 2 (2) | = 4 |
| | 3 (1) | 3 (2) | = 2 |
| --- | --- | --- | |
| | 4 (1) | 10 (1) | = 1 → (4) |
| | 5 (1) | 11 (1) | = 1 → (5) |
| | | 12 (1) | = 2 → (10) |
| | | | = 1 → (11) |
| | | | = 1 → (12) |
| | | | <u>18</u> rows. |

Sum of Common Rows + Non matching elements of both Tables.

3 Tables:

| <u>A</u> | <u>B</u> | <u>C</u> |
|----------|----------|----------|
| 1 (3) | 1 (2) | 1 (3) |
| 2 (2) | 2 (1) | 2 (2) |
| 3 (1) | 3 (2) | |
| 4 (1) | 10 (2) | |
| | 11 (1) | |
| | 12 (1) | |

Inner Join: matches of A, B, C (1, 2)
 $1's \rightarrow 3 \times 2 \times 3 = 18$
 $2's \rightarrow 2 \times 1 \times 2 = 4$
 Total rows = 22

Left Join: Left Join of A, B Result + C.

$$\begin{aligned} 1 \rightarrow 3 \times 2 &= 6 \\ 2 \rightarrow 2 \times 1 &= 2 \\ 3 \rightarrow 1 \times 2 &= 2 \\ 4 \rightarrow 1 \times 1 &= 1 \end{aligned}$$

Left result A,B 14

$$\begin{aligned} (1) \times 6 \times 3 &= 18 \\ (2) \times 2 \times 2 &= 4 \\ (3) \times 2 (3) &= 2 \\ (4) \times 1 (4) &= 1 \end{aligned}$$

25

Note: While joining 3 tables: A, B, C, ... 25
A Join B on $(A.C1 = B.C1)$ Join C on $(A.C1 = C.C1)$

While joining 3rd table columns of table A and B will select one row, and that row would be checked against 'C'.

System Tables

Dict: Data dictionary table. Holds the list of system tables in Data Base. (which holds the meta data of all tables). Dict acts as Repository.

- Includes all the tables of the data base. Like

dba-tables, all-tables, user-tables

dba-tab-columns, all-tab-^{columns}-tables, user-tab-^{columns}-tables.

> Select * from dict where table-name like '%.EMP%'; ^{like operator below lot of} Names on EMP in repository

1) TAB: It holds system default tables, + user created tables in the user login (Schema) (Tables, views, Synonyms) \rightarrow select * from tab;

2) CAT: synonym for "user-catalog" which will list that login (Tables, views, Synonyms and sequences) \rightarrow select * from cat;

Tab: It holds the list of tables available in user login.

User-Tables: holds the detailed information about the tables.

User-tab-columns: holds the brief information about columns defined in tables.

1) Select * from user-tables where table-name = 'EMP';

2) Select * from user-tab-columns where table-name = 'EMP';

All-tab-privs-mode: holds the list of permissions given to other users.

All-tab-privs-recd: holds the list of permissions received from other users.

Above two: Select * from All-tab-privs-mode;

All-users: system table which holds the list of users in the server.

Select * from All-users;

TABLE: (user-tables, user-tab-columns, TAB)

clusters: (user-clusters)

Views: (TAB, CAT, user-views)

Roles: (user-roles)

Synonyms: (user-synonyms, TAB, CAT)

Sequence: (user-sequences, CAT)

Index: (user-indexes)

Admin

- > create User user-name identified by password;
- > grant connect, resource to user1;
- > grant connect, resource to user1 identified by user1;
- > drop user user1 cascade; cascade cuts links with other users.

Locking/un Locking Alc:

Alter User user-name Account lock/unlock;

Alter ~~status~~ sys: Alter User Scott Account lock;

Constraints: (System Tables)

User_Constraints: holds the complete details of constraints defined on the table column.

User_Cons_Columns: holds brief info about the constraints applied on TDS Columns.

>desc user_constraints.

>select * from user_constraints where table_name = 'EMP';

P(PK), C(CHK/NN), U(unique), R(Referential)