CHAR stores fixed length characters

CHAR (5) if no of characters is less than 5 then appropriate space padding will be used, length will be 5

VARCHAR stores variable length character

VARCHAR(5) can have length from 1-5

DESC used to query data dictionary

Null value are treated as high values in order by clause

NULLS LAST is default for ASC and

NULLS FIRST is default for DESC

1=1 OR 1=0 AND 0=1 returns TRUE

Escape character in LIKE \

**Simple CASE**

CASE when <value> then <result>

else <result>

END

If else if not specified and none of the when are executed then NULL is returned

**Searched CASE**

CASE when <condition> then <result>

else <result>

END

When using operators like =, <,>,<=,>=,!= with sub-query the sub-query should always return single row

Same is not applicable for IN

=, <,>,<=,>=,!= do not consider NULL values as the result itself is NULL, it shows result which are either TRUE or FALSE

Same is applicable to IN as well if NULL is present in list of values then no rows are returned

**TRUE OR NULL** is TRUE

**NULL OR TRUE** is TRUE

**FALSE AND NULL** is FALSE

**NULL AND FALSE** is FALSE

Functions

**Single row** takes 1 row as input and 1 value as output

UPPER, LOWER, TRUNC, TO\_CHAR

**Multi row** (Group) takes multiple rows/group as input and returns single value for each group

COUNT, MIN, MAX ,SUM, AVG

**Arithmetic Functions**

ROUND(345.68,-1) = 350

ROUND(345.68,-2) = 300

ROUND(345.68,-3) = 0

TRUNC(345.68,-1) = 340

TRUNC(345.68,-2) = 300

TRUNC(345.68,-3) = 0

MOD(12,0) = 12

**Text Functions**

ASCII('a') = ASCII('abc') = 65

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | C | C | O | U | N | T |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

SUBSTR('ACCOUNT',4) = OUNT

SUBSTR('ACCOUNT',4,3) = OUN

INSTR('ACCOUNT','C') = 2 //first occurrence of C

INSTR('ACCOUNT','C',1) = 2 // first occurrence of C from 1 index

INSTR('ACCOUNT','C',1,2) = 4 // second occurrence of C from 1 index

TRANSLATE('TJENA SUJIT','TJENA','HELLO') = HELLO SUJIT //Replace every character in TJENA with HELLO

REPLACE('TJENA SUJIT','TJENA','HELLO') = HELLO SUJIT //Replace TJENA with HELLO

REPLACE('TJENA SUJIT','TJENA') = SUJIT // Removes TJENA

Functions supporting regex are also supported, they can be used for normal strings as well

GREATEST and LEAST works at row level

MIN and MAX works at column level

SAMPLE function

Used to do sampling in in case of large no of records

select \* from orders sample (1);

Will return 1% of total records

select \* from orders sample (1) seed (1234);

Will return same samples every time

Oracle does implicit conversion

WITH function - inline functions not stored in data dictionary

Transaction is a logical unit of work, ROLLBACK and COMMIT are used to end current transaction and start a new one

Autonomous transaction - sub transaction that can be COMMITED or ROLLBACKED irrespective of the status of main transaction

Eg Logging info while doing transaction, the logged data can be visible even the transaction gets rolled back

MERGE used to combine INSERT,UPDATE and DELETE statements together

Types of constraints

1. CHECK - check for particular condition
2. PRIMARY KEY
3. FOREIGN KEY
4. UNIQUE - check for uniqueness
5. NOT NULL

In case of foreign key constraint if on cascade is not specified then deleting the master records results in integrity constraint if child record exists

On cascade can be of 2 types

1. DELETE in which child records are also deleted
2. SET NULL in which child records are not deleted but set to NULL

Synonyms

1. Private available to user that created it, default
2. Public available to all DB users, requires DBA privileges

Tables dropped are present in RECYCLEBIN view, tables are restored using FLASHBACK but it may or may not succeed because the recycle bin can be purged by oracle or DBA

If drop table is used with PURGE, it is not moved to recycle bin

Indexes are database objects with reference to block where the actual record is present

By default indexes created as B Tree indexes (the cost of accessing any row is same)

1..1000

|  |  |
| --- | --- |
| 1..500 | 500-1000 |
| 1-250 | 250-500 |

1 rowid

……

….

250 rowid

The columns in the index are sorted asc by default

When we create a table with primary key or unique constraint Oracle creates a unique index containing the primary key column

No index is created for foreign key by default

Insert, update and deletes are costly as they require the same operations to be performed in indexes as well, as the contain redundant data so they consume extra space

Unusable indexes - data related to index is dropped from memory, query optimizer does not use the index (insert/update/delete do not affect the index)

Invisible index - data related to index still exits but query optimizer will not use the index (insert/update/delete still affects the index) - to check the usefulness of particular index before dropping it

Indexes are ideal when querying large table for less amount of data

Composite index - index containing multiple columns

Order of columns decide if index will used by query optimizer, if composite index contains col1,col2,col3 then index will be used in following cases

1. col1,col2,col3
2. col1,col2
3. col1

**Function indexes**

When functions are applied on the indexed column the query optimizer will not use the index for searching, so in such cases functional indexes are required (trunc(date)=sysdate)

Indexes containing the result of function of 1 or more columns

**Bitmap indexes**

To be used for columns have low cardinality i.e. having less distinct values

Not suitable where concurrent insert/update/delete of table containing bitmap columns takes place because of deadlock issue

|  |  |
| --- | --- |
| Id | Gender |
| 1 | Male |
| 2 | Female |

|  |  |
| --- | --- |
| Male | Female |
| 1 | 0 |
| 0 | 1 |

**Partitioning**

Divide the tables, indexes in to smaller manageable pieces enabling to be accessed at finer level of granularity

1. Performance
   1. Retrieving queries is faster
2. Availability
   1. Even if 1 partition is not available other partitions can be still accessed
3. Manageability
   1. Faster backup and restore

Oracle stores data in

1. Blocks (8KB) - basic unit , can have multiple rows in single block
2. Extent - specific no of continuous blocks that represent some data type
3. Segment - set of extends, each table is stored as separate segment, each partition has its own segment

In case of table the records are stored in blocks in random manner

In case of index it is organized as B Tree with the starting block as root block, intermediate blocks are branch block

and leaf blocks are the blocks containing the row id which points to the block where the table row exists

Accessing the table block is random I/O which is slowest so index is useful for small amount of data (Single block I/O)

In case of full table scan multiple blocks are read at the same time (128) so the number of I/O is less when accessing full table

Partition key - used to specify in which partition the data needs to be stored

If during insert particular value is not part of the partition key then ORA-1440 error is raised, in such cases use default partition to store this data

List partition to be used when the values for partition key are known before

Local index - if index is created on or part of partition key

Global index - if index is created on other than partition key

Range partition use maxvalue partition for ORA-14400

**Analytical functions**

Prevents self-join

Function(column\_name) over ([partition clause][order by clause][windowing clause])

lead() used to access the next row value based on order by clause

|  |  |  |
| --- | --- | --- |
| **Date** | **Dept** | **Name** |
| 02-02-1990 | 1 | Sam |
| 01-01-1991 | 1 | Tam |

select a.\*, lead(date) next\_date over (partition by dept order by date)

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Dept** | **Name** | **Next\_date** |
| 02-02-1990 | 1 | Sam | 01-01-1991 |
| 01-01-1991 | 1 | Tam | null |

sum(sal) over (partition by dept\_id order by emp\_id) will return cumulative salary dept wise

sum(sal) over (order by emp\_id) will return cumulative salary dept wise will return cumulative salary for the table

sum(sal) over (partition by dept\_id) will return dept wise total salary for each record

If you don’t specify windowing clause with order by then the default is range between unbounded preceding and current row

There are 2 types of windowing clause

1. Range between <start point> and <end point>(no of rows are not known)
2. Rows between <start point> and <end point>(no of rows are fixed)

Values for start\_point and end\_point are

1. UNBOUNDED PRECEDING
2. UNBOUNDED FOLLOWING
3. CURRENT ROW
4. <number> PRECEDING
5. <number> FOLLOWING

Rows between 2 preceding and current row - Previous 2 rows and current row

Range between 2 preceding and current row - Current rows and rows having values lesser than current row by 2

Range between considers the all duplicate values as current row whereas rows between considers them distinct

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dept\_id | | Salary | | Avg using rows between | | | Avg using range between | |
| 1 | 100 | | 100 | | 100 | 100 | |
| 2 | 200 | | 200 | | 150 | 250 | |
| 3 | 200 | | 200 | | 250 | 250 | |
| 4 | 300 | | 300 | | 200 | 200 | |

**Views**

Views are virtual table in which the script for creating the view is run in the background every time you query a view, used to provide restrictive access to underlying tables

Views support DML operations

Provides abstraction by hiding complex logic, data security (by reovking or granting access to view rather than table)

CREATE or REPLACE view preserves the granted privileges

FORCE options does not heck if underlying table exists or if access to table exists

If the view script contains select \* it does not mean all new columns added in underlying table will be accessible from view

You cannot change view definition, only compiled using ALTER VIEW

WITH READ ONLY option prevents DML of views

Instead of triggers are run when DML operations are performed on view, they can be used to update non updateable views

When you insert or update rows in view then those updates/inserts are invisible to the view

To prevent this behaviour use WITH CHECK OPTION while creating view which will result insert/update failure

**Materialized view**

Objects that store the output of query

Benefits

1. Performance improvement using query rewrites
2. Less n/w bandwidth when accessing data from remote DB

Query rewrite

When the optimizer uses the materialized view instead of the underlying table while fetching query result

**Global Temporary Table**

Data exists till end of transaction (commit/rollback) or session but table definition still exists

Several concurrent session can uses the same GTT as GTT data session in exclusive

Higher performance due to less locking

**JOINS**

Inner join - The value should exists in both the tables

Left join/Left outer join - All the values from the driving table should be present

Right outer join

Full outer join

Cartesian product - for generating test data

Theta / no equijoin - join condition contains other than == operator

Natural join - join columns are columns common to the tables, no ON condition specified

Cross join - join without ON or USING conditions but instead using WHERE clause

Group By considers NULL as separate group if it is applied on column containing NULL values

Group By also considers duplicate values

AVG(1,2,3,NULL) = 2

AVG(1,2,3,0) = 1.5

If a column exists in select clause other than group function argument it should be part of group by clause

Select A, B ,count© from T group by A,B

Select count(A) from T group by C

Are all valid

UNION ALL - has duplicate values

UNION/MINUS/INTERSECT - has unique values

Q1 UNION Q2 - column names and alias are inherited from Q1, order by can be present in Q2 and is applicable to the result

**rowid** is pseudo column which returns physical address of the row, always remains the same

**rownum** is pseudo column which returns the order in which data was retrieved in select, may vary each time

**Invisible columns -** columns are not visible in "select \* " clause but are when explicitly written in select clause, columns in view can also be made invisible

**SQL normal forms**

1NF

A table is said in 1NF if there exists no multivalued attribute

EMP\_ID         EMP\_NAME         EMP\_PHONE

1                 Sujit                1234,2345

----------------------------------------------------------

EMP\_ID         EMP\_NAME         EMP\_PHONE        EMP\_DEPT         EMP\_DEPT\_NAME

1                 Sujit                1234                1                        COLLATERAL

1                 Sujit                2345                1                        COLLATERAL

2                Ra

2NF

Relation should be in INF

No partial dependency should exists

If non-prime atrribute is dependant on subset of candidate key then it is partial dependency

MEM\_TYPE         MEM\_CD         MEM\_NAME         AMOUNT

M                         08081        KOTAK SEC        100

C                         HDFC        HDFC SEC        200

------------------------------------------

MEM\_TYPE        MEM\_CD        AMOUNT

M                         08081        100

C                 HDFC        200

MEM\_CD                MEM\_NAME

08081                 KOTAK SEC

HDFC                HDFC SEC

Here non-prime attribute is Member name, Member code->Member Name and candidate key is {MEM\_TYPE,MEM\_CD}

3NF

Relation should be in 2NF

No transitive dependency should exists

EMP\_ID         EMP\_NAME        EMP\_DEPT         EMP\_DEPT\_NAME

1                 Sujit                1                        COLLATERAL

2                Rajat                1                        COLLATERAL

-------------------------------------------------

EMP\_ID         EMP\_NAME        EMP\_DEPT

1                 Sujit                1

2                 Rajat                1

DEP\_ID        DEPT\_NAME

1                COLATERAL

Here Employee id -> Dept id and Dept id->Dept name so Emp id->Dept name is transitive dependency

BCNF

For every FD x->y x should be super key of table

If a relation is in BCNF, then 3NF is also also satisfied.

If all attributes of relation are prime attribute, then the relation is always in 3NF.

A relation in a Relational Database is always and at least in 1NF form.

Every Binary Relation ( a Relation with only 2 attributes ) is always in BCNF.

If a Relation has only singleton candidate keys( i.e. every candidate key consists of only 1 attribute), then the Relation is always in 2NF( because no Partial functional dependency possible).

ora\_max\_name\_len constant specifying the max name length for objects

Oracle object names are 128 bytes instead 30bytes

Two tables can have same rowid for particular row