DAY - 1

02/09/2024

Data (information)

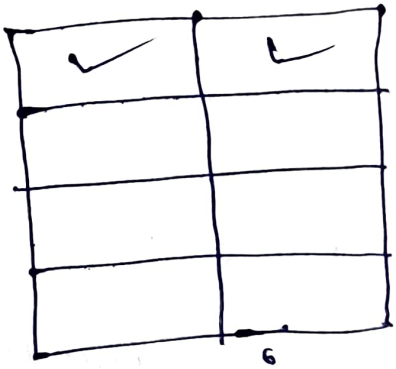
↓

Database (A container where the meaningful information/ data are containing)

↓

Database Management (The process by which the data are being collected & modified)

CHAR	VARCHAR
Name	XYZ
A/C No.	10000.....
Balance	5000/- ...
Date	02/09/2024



Oracle Environment

Oracle & its tools :-

1) server (backend)
2) client (GUI)
3) Network (communication b/w server & client)

File structure :-

1) logical structure → (How we are seeing the file logically (to view of the data in a two dimensional table)
2) Physical structure → (stored in a disk [one data files, two lock files, 3 control files])

Client side tools :-

1) SQL plus → 1) Interactive SQL / SQL (i)
→ 2) PL / SQL

SQL plus commands

1) DDL → Data definition language (lock file)
(create, alter, drop)

2) DML → Data manipulation language (data file)
(insert, delete, update)

3) DCL → Data control language (Commit, roll - back)
(control file)
like ⇒ (save, undo)

Definition :-

- Used to create objects (DDL)
- Used to manipulate objects within data (DML)
- Used to control the behaviour of the control (DCL)

Tools provided by Oracles :-

1) Declerative
2) Triggers
3) Alerts
4) Data base integrity
5) Procedures
6) security

Creating a table :-

```
create table college
                [table name]
(field name data type(),
 field name2 data type());
```

Student 5th

Name	Roll no.
ABC	123
XYZ	456
....	....

✓ Inserting data into a table :-
~ Insert into table name
values ('ABC', 123);

✓ viewing data in the table :-
~ all rows & all colounns for that —
select * from table name ;
[(student 5th) (e.g.)]

✓ select rows & all colounns :-
~ select colounn1 from student 5th

✓ condition search :-
~ select * from table name
where search condition
→ [(marks > 80) (e.g.)]

✓ Elimination of Doubliates from the select student :-
~ Select distinct colounn, colounne
from table name;

~ Select distinct *
from table name;

✓ Sorting data in the table :-
~ Select from table name
order by colounn name
with desc or asc;

✓ Deletion rooms from a table :-
~ Delete from table name;
~ Delete from table name
where search condition;

✓ update rows in the table

~ update table name
set field = valid;

✓ update table name

~ set field = value
where condition;

✓ Modifying the structure of table :-

~ alter table table name
add (new column data type ());

~ alter table table name
modify (column name ());

~ alter table name
drop (column name, data type ());

✓ Renaming table

~ rename old table name to new name
[e.g. → rename student 5th to student 6th]

✓ destroying table

~ drop table table name
~ tables → objects created by user ← delete the structure of table along with objects
~ select * from tab

DAY-2

09/09/24

LOGICAL OPERATOR

~ select * from table name
where column1 >= value AND column2 >= value;

[e.g. - sub1 >= 50 AND sub2 >= 90]

~ select * from table name
where column1 >= value OR column2 >= value;

[e.g. sub1 >= 50 OR sub2 >= 90]

~ select * from table name
where ~~so~~ ~~NOT~~ column1 >= value;

[e.g ~~NOT~~ sub1 >= 50]

~ select * from table name
where Roll no. BETWEEN 10 AND 20;

~ select * ~~from~~ ~~cal NO~~ table name
where cal NOT BETWEEN 10 AND 20

ADDITIONAL OPERATOR
OR
PATTERN MATCHING

'%' is used to denote any number of unknown characters.

'_' is used to denote only unknown characters

e.g. select * from table name
where name LIKE 'K%' ; [starting with K name &
'%K%' ; [containing K
[end with K] ⟶ '%K';

~ Select * from table name
where name LIKE 'H_ H';

~ Select * from table name
where name LIKE 'N__ S';

~ Select * from table name
where name LIKE '_ W_ ';

~ Select * from table name
where name IN '_ W_ ' ; PATTERN MATCHING
NOT IN '%A%' ;

ADDITION OPERATOR

~ Select * from table name
where st code NOT IN 'BWN, HWH'
IN 'HWH, NDLS';

COMMITE, ROLL-BACK
(Save) (Undo)

SAVEPOINT D1;
Delete from table name
where coloumn = value;

SAVEPOINT D2;
Delete from table name
wher colourn = value ;

ROLLBACK D2;

REFERENTIAL INTIGRITY → (clubing / United)

while considering REFERENTIAL INTIGRITY we must conside a master table & a transaction table / child table

create master table, table name
(Rollno, number 4 references
↳ primary key

~~only when~~ create transaction table, table name
roll no., number 4 references
master table on delete cascade.
↳ Foreign key.

? S2022067

→ create table student (roll no. varchar 2, name varchar 2 (20), add varchar 2 (30), DOB DATE, mobile varchar 2 (10);

create table student (rollno. varchar (10), Primary key, name varchar 2 (20), add varchar 2 (30), DOB DATE, mobile varchar 2 (10);

Create

S 2022067
DEPT 2022067
PROF 2022067

Join Operation

stud - org → roll no. name, add, ph no., pin ;

stud - marks → roll no name, roll. no, marks ;

join means to access tables.

The tables are — join with the where clause. in which the common key field must be specified.

(1) Equi Joint :-

when only the matching records are displayed from both the tables. it is called equi-joint.

(2) Self Joint —

Sometimes it is required to join a table to it self as through joining of two seperate tables. This is called as a self joint. For this joining two rows from the same table combine to form a resultant row.

(3) Outer Joint

If matching records are not present in second file many names from the 1st file are not listed at all to retrive this records also we have to perform an outer join (+). The data which are not available in the second file will be presented as Null values.

Example:- player → primary key

Roll no.	Name
10	Vijay
20	Liender
30	Anant
40	Mahesh

match

Match_no.	Roll_no.	Match_date	Oponent
1	20	10-JUL-24	
2	30	12-JAN-24	
3	20	12-AUG-24	
4	30	20-MAR-24	

```
select player.rollno, name, match_date, oponent
from player.match
where player.roll no = match.rollno;
```

Outer Join (+)

Select player. ball no., name, match date, opponent
from player, match
where player. ball no. = match no.

ball no.	match_date	Name	~~Opponent~~ match date
20	NULL	Landar	10th July
20	10-JUL-24	Landar	12th Aug
30		Anant	12th Jan
40		Anant	20" March

Select player. Ball, name, match date from player, match
where player. ball = match. ball (+);
10, 20, 20, 30, 30, 40

Table Alias :-

Select p. roll no, name, match_date, opponent.
from player p, match m
where p. roll = m. roll;

column Alias :-

Select employee_number as "ENO", ~~basic salary "BS",~~
~~Salary_date "SDATE"~~, employee_name as "NAME"
~~from salary~~ from employee.

Special Application of alter.

(i) Alter table table name (Attribute name)
(ii) add primary key
(iii) drop primary key cascade
(iv) add (foreign key (attribute) references master table on delete cascade);

Alter table table_name
modify attribute name not NULL

~~STUD2022107~~

Date : 21/10/24

Oracle functions

Oracle functions can return a result. Functions are also capable of accepting user supplied variables and operating on them. Such variables and constants are called as arguements. Any no. of arguments are passed through function.

1) Number Function
2) Character Function
3) Aggregate Function
4) Date Function
5) Conversion Function

Dual

Oracle provides a special table Dual which can be used to test any function. Eg: (1) Select sysdate
from DUAL
21-OCT-24

(2) Select user
from DUAL

1) Number Functions :-

i) ABS (Data):

Select ABS(-5000)
from DUAL
→ 5000

ii) CEIL(Data):

Select CEIL(125.55)
from DUAL
→ 126

iii) LOG(Data) :

Returns the natural logarithm of the data

iv) FLOOR(Data):

Select FLOOR(125.55)
from DUAL
→ 125

v) LOG(b, data) :

Returns the base logarithm of data

vi) MOD (Data, y):

MOD (10, 2)

vii) POWER (Data, y) :

POWER (2, 4)
↳ 16

viii) ROUND (Data, n)
↳ (223.55, 0) → 224

ix) SQRT (Data) :

x) TRUNC (Data, n) :
(123.55, 0) → 123
(123.55, 1) → 123.5

2) CHARACTER FUNCTIONS

i) INITCAP (Data) :

select INITCAP ('UIT')
from DUAL
↳ Uit

ii) LENGTH (Data) :

select LENGTH ('UIT')
from DUAL
↳ 3

iii) SUBSTR (Data, x, y) :

This will extract 'y' characters starting from x^{th} character of data

→ select SUBSTR ('MANIPAL', 5, 3)
from DUAL
↳ PAL

iv) INSTR (Data, x)

will find the location of string x in the string data a

Eg: ('MANIPAL UNIVERSITY', 'PAL')
→ 5

v) INSTR (Data, x, l, n) :

will return the location of the n^{th} occurrence string x in the string data starting from position 1

E.g. - ('MANIPAL UNIVESITY', 'I', 1, 3)
→ 6

vi) GREATEST (expr1, expr2, ... exprn) :

Return the greatest

vii) LEAST (expr1, expr2, ... exprn)

3) CONVERSION FUNCTIONS

TO_CHAR

The TO_CHAR function can be used to convert a date or number to a character string. With the TO_CHAR function a Date Item can be formatted in many ways. The format must be enclosed within single quotes.

e.g. – ① Select TO_CHAR(SYSDATE, 'MONTH')
from DUAL;
→ OCTOBER

② Select TO_CHAR('SYSDATE, 'DD/MM/YYYY')
from DUAL
→ 21/10/2024

③ select * from SALARY
where TO_CHAR(SALARY_DATE, 'MM')
EQUALS TO "03;
→ Returns the salary for March

④ select * from SALARY
where TO_CHAR(SALARRY_DATE, 'Q')
EQUALS TO 3;
→ Returns as the salary for july, augest, septembers months.

4) AGGREGATE FUNCTIONS

i) AVG(x)

Eg.- Select AVG(Basic)
from SALARY

EmpId	Basic	DA
1	3000	
2	5000	
3	2000	
4	3000	
5	1000	

x → the column name itself
→ Basic Table → 5

ii) COUNT(x)

e.g.- Select COUNT(Basic)
from SALARY;
↳ 5

iii) MAX(x)

Select MAX(Basic)
from SALARY

iv) MIN(x)

v) SUM(x)

vi) STD DEV(x)

vii) VARIANCE(x)

5) Date Functions

i) ADD_MONTHS (Date, Count)

Eg- select ADD_MONTHS ('15-JAN-24', 5)
from DUAL ;
↳ 15-JUN-24

ii) LAST_DAY (Date)

select LAST_DAY ('21-OCT-24')
from DUAL
↳ 31-OCT-24

iii) MONTHS_BETWEEN (Date2, Date1)

Eg- select MONTHS_BETWEEN
('15-JAN-24', '28-MARCH')
from DUAL;
↳ -2

iv) NEXT_DAY (Date, Day):

Select NEXT_DAY ('01-OCT-24', MON).
from DUAL;
→ 07-OCT-24

Date: 18/11/24

The check Constraints :-

To apply the validation to a table column, the check constraint is used. It is specified as a logical expression that evaluates either true or false.

Example: Create a table with following check constraints
i) Data value being inserted into the column comp_no must start with capital letter 'C' [C001, C002]
ii) Data value being inserted into the column Name should be in upper case only. [ROHIT]
iii) Allow only Delhi, Mumbai as value for the column city.

⇒ create table COMP
(COMP_NO varchar(6), check (COMP_NO like 'C%'), Name varchar(10) check (Name = UPPER (NAME)), CITY varchar(6), check (CITY IN ('Delhi', 'Mumbai'));

Naming of Constraints :-

create table tablename
(Name varchar(10) CONSTRAINT ABC CHECK (Name = upper(Name)));

Drop Constraints

Alter table tablename
DROP ABC;

NOTE : Two columns cannot be compared through check constraints in a single table.

Comp : Pdate | Sdate

check Pdate < Sdate ✗
alter table tablename
add constraint CHK Age check age>18

Extension of Character Function or String Function

- lower (char)

Ex: select lower ('name')

O/P - name
name

- upper (char)

select upper ('name') from dual

SYS_DATE
17-OCT2024

name
Name

- Ltrim (character, set)

select Ltrim ('Name', N) from dual;

name
ame

Rtrim (character, set)

select Rtrim ('Name', E)
from Dual

name
Nam

- LPad (char1, n, char2)
- RPad ('Page 1', 10, 'X') → it count the space and impose the characters (char2).

O/P Page 1XXXXX

groups By Clause

The groups by clause is used to produce summary information for a table. The group by clause will produce a single row for all the rows that ~~must~~ meet a particular condition.

Emp-no	Basic	Date
1001	2000	30 Jun 2024
1002	3000	30 " "
1003	4000	" " "
1001	2000	30 Jul 2024
1002	3000	30 Jul 2024
1003	4000	" " "

select Emp_no, Sum(Basic) from Emp
select Emp_no, Sum (Basic) from Salary
group by Emp-no

O/P

Emp_no	Sum (Basic)
1001	4000
1002	6000
1003	8000

* Select Emp_no, Sum(Basic) from salary
group by Emp_no.
order by Sum(Basic)

O/P

Emp_no	sum (Basic)
1001	4000
1002	6000
1003	8000

* select S.EMP_NO, INIT CAP (EMP_NAME), AVG (Basic) from SALARY S, EMP
where S.EMP_NO = EMP.EMP_No
Group By S.EMP_No, EMP_NAME
Order By AVG (Basic);

Emp_No	Name	Sum(Basic)
1001	ABC	4000
1002	XYZ	6000
1003	MNO	8000

NOTE : group by clause does not sort the data, it only groups together data which have in order to sort data the order by clause is used

O/P

Emp_No	Name	Avg (Basic)
1001	Abc	4000
1002	Xyz	6000
1003	Mno	8000

Example: select TO_CHAR (SALARY_DATE, 'MONTH') "MONTH", SUM (BASIC) "TOTAL BASIC" FROM SALARY GROUP BY TO_CHAR (SALARY_DATE, "MONTH");

O/P

MONTH	TOTAL BASIC
June	9000
July	9000

Having clause

The having clause can also be used with a join query which includes a group by clause. The having clause is useful for specifying a condition for group.

The where clause is used for specific condition to retrieve rows of a table and the having clause is used to specify a condition to retrieve grouped data.

Example

select S.Emp_No, Emp_Name, Avg (Basic) from
salary S, Emp
where S. Emp_No = Emp. Emp_No
group by S. Emp_No, Emp_Name
having avg (Basic) >= 2500
order by avg (Basic)

Emp_No	Emp_Name	Avg (Basic)
1001	XYZ	3000
1003	MNO	4000

28|11|24

Subquery

Subquery is a query which is within the query. i.e. a select statement is used as the part of another statement the outer statement is called the parent and the nested query executes first. the subquery passes a value to the outer query.

1. The nested query must return a single column,
2. The result can only contain columns from the tables referenced in the outer most query.
3. The nested query must return a single row when the standared operators such as = < > are used.
4. The between operators cannot be used whether the subquery.
5. Subquery is also used insert, update & delete statements.

example:- ~ select * from salary where basic < (select avg basic from salary);

~ Select * from salary where basic > (select avg basic from salary);

~ Select emp_no, emp_name from emp where emp_no not in (select emp_no from salary where dept = mkt)

This query can not be rewritten as a join as join should not be negative.

Join & Subquery

1. Joining is more expensive than subquery (space)
2. Joining is of same level execution.
3. Critical subqueries can't be replaced by joining
4. Join is faster, subquery is not that much faster.
5. Joining is same level & subquery is multilevel.
6. Critical subqueries cannot be replaced by Joining.

Using subquery insert data

~ Create Table saldup as (select * from salary);
[create a table which duplicate of previous table]

~ Create Table saldup as (select * from salary where 1>2);
[create a structure of table but don't copy the value of the table]

~ Insert into saldup (select * from salary where address = value);

~ Delete from emp where emp_no in (select emp_no from salary where deduction = 50);

Set Operators

Oracle supports ~~four~~ four set operation.
All set operators combines of two or more queries into a single result. Everyone is both ~~list~~ the list.

while avoiding sending two letters to someone who happens to be listed only for those ① who are in both the list, ② For those who are in only one list.

1. Union → returns all rows returned by either query eliminates duplicate rows.

2. Union All → returns all rows returned by either query doesn't eliminates duplicate.

3. Intersect → return all distinct rows returned by ~~queries~~ both queries. i.e. ~~both~~ ~~in~~ must exist in both query output to be return in the final result.

4. Minus → produces rows returned by 1st query but not the second