Date: 19-6-18
MON TUE WED THU FRI SAT

Chapter - 6

- DATA : Data is a raw fact or material that doesn't provide any meanings to the user.
For Ex : Class, Students, teachers, benches blackboard, etc.
It Conveys inappropriate message to the user. After processing of the data it is Converted into Information.

- INFORMATION : The process data that provides the meaningful reason or message to the user is called as an information.
For Ex : The Students of B.Tech Comp. engineering studying in a class room

The Datas that are used to display and to perform the processing are stored in the form of tables. The tables is devided into the two dimens. that is either rows and columns. The rows & columns are used to store the multiple amount of data for which the datas are simplified.

Date:
MON TUE WED THU FRI SAT

- INCOMPLETE DATA: The Datas that doesn't provides complete meaning or facts to the user or improove is called as incomplete data.

Ques: Which tool is used to store the large amount of data?

1. Apache Hadoop
2. Microsoft HD Insight
3. No SQL
4. Hive
5. Sqoop
6. PolyBase
7. Bigdata in Excel
8. Presto

Oracle: Larry Ellison
Bob Miner
Ed Oates

C: Dennis Ritchie
Ken Thompson

- SQL: Structed Query Language is the standard command set used to communicate with the relational database management System. All tasks related to database management such as creating tables, modifying the data in the database

deleting them, Querying the database for information giving access to the users and etc. can be done using SQL

- Characteristic of SQL
→ It is flexible language.
→ It uses a Syntax that gives the user the ability to structure SQL statements in the way which is best suited to that.
→ Each SQL request or Query is check by relational database management System before execution for proper syntax and to optimize the request.

- Advantages of SQL
→ SQL is the higher level language that provides a greater degree of abstraction.
→ SQL enables the end-users and systems personnes to deal with a number of database management System that is available.
→ Applications written in SQL can be easily ported across the system.
→ It Specifies what is require and how it should be done.
→ It is very simple as well as easy to learn and it can handle the complex situation.

Date:
MON TUE WED THU FRI SAT

All the SQL Operations can be performed at a set level.

- SQL Datatypes
  - → Character
  - → Varchar
  - → time-stamp
  - → integer
  - → Small integer
  - → float
  - → Date (date)
  - → time
  - → Real and double decision

- Character: This datatype represent the fixed length of a string with exactly "n" character where n ≥ 0 should be any integer value.

- Syntax name character(10)
  n = 10

- Varchar: This datatype represents a varying length of string whose maximum length is "n" Character
- Syntax name varchar(10)

- integer: integer represent assigned integer value decimal or binary
  Syntax enroll_no Integer(15)

Float : A floating point number with precision of atleast 'n' digits
Syntax Valuefloat(5)

Real and double decision : floating point value and double decision floating point number with machine dependent precision.

date : A character date containing the four digit number year, month & day.
Ex → date_of_birthdate

time : The time of a day in hours, minutes & seconds
Ex → arrival_time time

time-stamp : The combination of date and time is represented by time-stamp.
exc → date '2005-04-25'
time '09:10:25'
time-stamp '2005-04-25 09:10:25'

The SQL Statement is divided into the four categories
Data definition language
Data Manipulation language
Data Control language
Data Query language.

Date:
MON TUE WED THU FRI SAT

Data Definition Language: It is used to create alter and delete database and objects. The Commands that are used are listed below

1. Create - to Create a table
2. alter - to perform changes
3. droop - to delete the table.

Data Manipulation language: It allows the user to insert, modify and delete the data in the database. SQL provides three data manipulation statements.

1. insert - to insert the row into the table
2. update - to perform change into the table
3. delete - to delete any row from the table.

Data Query language: The SQL statement enables the user to query one or more tables to get the information they needed. For this purpose SQL has only one data query statement that is Select

Data Control language: It Consist of commands that controls the users to access the database objects. Various data control commands are used to update as well as to modify the database.

Date:
MON TUE WED THU FRI SAT

For ex: grant, reboke, abort, etc.

Datatypes: The datatypes are used to store format specific value.

Small integer: A small integer is a machine independent subset of the integer datatype that are used to store two digits only.

SQL Command:

SYNTAX Create table <tablename>
(<columnName1> <datatype> <size>,
<column Name 2> <datatype> <size>);
This command is used to create a new table the column names should be unique while creating a table.
For ex: Create table Student
(Stud_id Varchar(15), Stud_name
Vcarchar(25), Stud_add Varchar(50));

Inserting the Values into the table

SYNTAX insert into <tablename> values
(<expression1>, <expression 2>, <exp..n>);

The insert statement is used to enter the rows into the table. The rows which are entered into the table should be uniquely defined by making use of any

identity.
For ex: insert into Student values ('01', 'xyz', '1217');

The rows are always mapped according to the sequence of the column name.

Viewing of the data

Once the rows are inserted into the table and for viewing the rows the select command is used to find the rows from the table.

Select * from <tablename>

i) → select stud_id, stud_name from student
ii) → select * from student where stud_id = '2'

For selecting column (i).
For selecting rows (ii)

NOTE) To View the data in the database we have three categories
i) For all rows & all columns.
ii) For selected rows & all columns.
iii) For all rows & selected columns.

Date.:
MON TUE WED THU FRI SAT

Eliminating duplicate rows by selecting Select Statement.

The distinct claws allows to remove the duplicate value from the result set. It scans through the value of columns specified and display only the unique values among them.

SELECT DISTINCT <columnname1><ColumnName2>
FROM <Tablename>;

The distinct claws will scan the entire rows & eliminate the rows that have exactly the same contents in each column

SELECT DISTINCT * FROM <Tablename>;

It will display the unique rows of the table.
For ex: Select distinct * from student

How to Short the data in the table

The rows that are retrieve from the table will be sorted either in ascending order or in descending order.

Date:
MON TUE WED THU FRI SAT

SYNTAX

Select * from <Tablename> order by <columnname1> <ColumnName2> <[sort order]>;

The Order by claws short sort the result based on the column specified. The order by claws is used in the select statements.

Show the details of student according to the students enroll no.

Ascen→ Select * from student order by eno [AESC]

Descen→ Select * from student order by eno [DESC]

Creating a Table from another table.

To Create the Subtable from the source table we are making use of As Select Statement

SYNTAX.

Create table <Tablename> (<column Name1>

<ColumnName2>)
As Select <ColumnName1>, <ColumnName2> FROM <Tablename>

Ex: Create table Student mstr from a table Student

Create table Student mstr(Student_eroll , Student_name).
As select student_eroll , Student_name FROM Student_mstr;

* Inserting a data into a table from another table

insert into <tablename> SELECT <columnName1>, <columnName2>. FROM <Tablename>

Delete Operation

To delete a specific row from the table.

Delete from <tablename> where condition

Ex: Delete 0 Student for where enrollno = 1

Date:
MON TUE WED THU FRI SAT

For the removal of all the rows
Delete from <Tablename>;

To delete the table from the database
drop <tablename>;

Updating the contents of the table

Update <Tablename> set <column Name1>
= <Expression1>, <column Name2> =
<Expression2>;

This statement is used for updating all the rows in the table. The update statement updates the column in the existing rows of table with the new headings. The set clause indicates which column data should be modified and the new values are inserted in that column.

Ex: Update the address details (Modify, Change) of the Student by changing its city name to Bombay.

Update Student set address =
'Bombay';

Updating records Conditional where clause is used.

update <Tablename> set <ColumnName1> = <Expression1>, <ColumnName2> = <Expression2> where <condition>;

Ex: Update the student details by changing Bombay to Surat.

update student set address = Surat;

Modifying the structures of table.

The structure of the table can be modify by using 'alter' command. 'ALTER' Table allows changing the structure of an existing table. With 'alter' table it is possible to add or to delete the columns. To create or to destroy the indexes, to change the datatype of existing columns or to rename the columns or the table itself.

SYNTAX FOR ADDING THE NEW COLUMNS.

ALTER Table <Table name> ADD (<New Column Name> <Datatype> (<size>), <New Column Name> <Datatype> (<size>), ...);

Date:
MON TUE WED THU FRI SAT

Eg: Modify or Enter a new field (column) called city in the table Student_mstr.

ALTER Table Student_mstr ADD (city varchar(100));

Droping a Column from the Table

SYNTAX

ALTER Table <Table name> DROP Column <ColumnName>;

Eg: Drop the column city from Student_mstr table

ALTER Table Student_mstr DROP column city;

Modifing the existing columns

SYNTAX

ALTER Table <Table name> MODIFY (<ColumnName> <New Datatype> (<New size>));

Eg:

ALTER Student_mstr table to allow the name field to hold maximum of 30 char.

Date:
MON TUE WED THU FRI SAT

ALTER
MODIFY Table Student mstr MODIFY
Student name Varchar (30);

Rules / Restriction for using ALTER Table

The foll. task cannot be preform by using ALTER Table claws.

1. To change the name of a table.
2. To change the name of a column.
3. To decrease the size of the column if the table data exist.

Creating / Renaming

Renaming the name of the table.

SYNTAX
RENAME <Table name> to <New-Table name>

Eg: change the name of Student mstr table to Student table

Rename student_mstr to student

Truncating the table

Truncate Table is used to empty a table logically. This is equivalent to a delete statement that deletes all the rows.

By using the truncate keyword. table the number of deleted row are not returned as it use make use of primary key constrain

Truncate table <Tablename>

Synonyms

A Synonyms is an alternative name for the objects such as tables, Views, Sequences Stored procedure and other database object.

SYNTAX

Create [OR Replace] [Public] SYNONYM [SCHEMA] SYNONYM_NAME FOR [SCHEMA] Object-Name [@DBLINK]

Examine to objects created by a user

Select * from TAB;

To display the table structure

Describe <Tablename>

DESC

Date:
MON TUE WED THU FRI SAT

Show the table structure of student_mstr

Describe student_mstr

Data Constraint

Data Constraints are the set of rules that must be applied to the data which is gathered, stored or analized in the database or table to ensure its intigrity. The rules appling to the data will be different for the different applications. There are two types of data constraints that are applied to the data being inserted into the oracle table. One type of constraints is i/o constraint, and the other type of constraint is business rule constraint.

SYNTAX :

PRIMARY KEY

<columnName> <Data type> (<size>)
primary key

A primary key is one or more column's use to uniquely identify each row in the table. None of the fields that are the part of primary key can contain a null value.

A table can have only one primary key.
A single column primary key is called as a simple key.

A multicolumn primary key is called as a composite primary key.
The only fun of a primary key in a table is to uniquely identify a row

Features of Primary key

→ The main purpose of the primary key is to record uniqueness
→ It will not allow duplicate values
→ It will not allow the null values
It is not compulsory but it is recommanded.
→ Only one primary key is allowed per table.

Primary Key Constraint defined at table level.

SYNTAX

PRIMARY KEY (<columnName>
,<columnName>)

Eg: Insert the primary key for the whole table Student mstr with the fields

Date.:
MON TUE WED THU FRI SAT

Student_e no, Student name, cloud Student_add

Create table Student mstr (Student en_no Varchar (20), Student_name (20 Varchar (25), Student_add Vcachar (50), primary key (Student_en_no, Student_name, student_add));

Foreign Key Constraint

The foreign key represents relationship between the two tables. A foreign key is a column or the group of columns whose values are derived from the primary key or unique key of some other table. The table in which foreign key is defined is called as foreign table or detail table. The table that defines the primary or unique key & it is referenced by the foreign key is called primary table or master table.

SYNTAX : <ColumnName> <Datatype> (<size>) REFERENCES <Tablename> → Primary Key value [(<columnName>)] [ON DELETE CASCADE]

Eg: Create a Table Student_mstr with its primary key as Student_en_no referencing the foreign key course no in Student table.

Date: 13-7-18
MON TUE WED THU FRI SAT
☐ ☐ ☐ ☐ ☑ ☐

Create table Student

Course_no varchar(20) Reference Student [student_en_no]

Foreign Key Constraint at table level.

Foreign Key (<columnName>, <columnName> REFERENCES <TableName>[(<columnName>) (<column Name>)]

Eg: Create table student details with Courseno as foreign key referer in the column of course_no in the table student_mstr

Create table Student_details (Student_id varchar(10); Course_no varchar(10), Foreign Key (Course_no) REFERENCES Student_mstr (course_no));

Foreign Key Constraint define with on delete cascade.

Create a table Student details with its foreign key as Student Serial no with On Delete Cascade option

Create table Student

Date:
MON TUE WED THU FRI SAT

Unique Key

The unique Column Constraint permits multiple entries of null into the column. The null values are joint together or collected at the top of the Column in the Order in which they are enter into the Sql Table. This is the primary difference between the primary key & the unique Constraint applied to the Table Columns.

<ColumnName> <Datatype>(<size>)
UNIQUE

Unique Key Constraint Defined at table level

Create table tablename(<ColumnName1>
<Datatype>(<size>) UNIQUE
(<ColumnName1>, <ColumnName2>));

Create table Student mstr such that the Contains of the Column, student id are unique across the entire Column.

Create table S

Date:
MON TUE WED THU FRI SAT

Null Value Concepts

- A Null Value is used when the actual value is unknown or when the value is not meaningful.
- A Null Value is not equivalent to value of zero if the datatype is number & it is not equivalent to space if the datatype is char.
- A Null Value with evaluate to null in any expression.
- A Null value can be inserted into columns of any datatype.

Difference between an empty string and a null value.

```
Select * from Student_mstr where
    Name = '' '';
```

→ While this statement is executed it is expected to retoy the row that is inserted about but this statement will not retrive any record.

```
select * from Student mstr
where Name = Null;
```

→ When this Statement is executed the rows are retrive from the table.

NotNull Constraint Defined at the Table level.

Oracle has the Notnull Constraint defined at its Column level Constraint. The notnull Constraint ensures that a table Column cannot be left empty when a Column is defined as notnull then that Column becomes the mandatory column.

SYNTAX

<ColumnName> <Datatype> (<size>)
NOTNULL

CHECK CONSTRAINT

Business rule Validation | Can be applied to a table Column by using the check Constraint. Check Constraints must be specified as the logical expression that evaluates either True or False.

SYNTAX

At Column Level

<columnName> <Datatype> (<size>)
CHECK (<logical expression>)

At table level

CHECK (<logical expression>

Date:
MON TUE WED THU FRI SAT

Create a student mstr with the foll Check Constraint

1) Data values inserted into the colum Course no must start with the Capital letter C.
2) Data value being inserted into the column Fname, Mname, Lname should be in the upper case only.

SYNTAX:

CHECK (course_no LIKE 'C%')
CHECK (Fname = Upper(Ename))

Computation done on TABLE DATA

The Computations done on table data is done by using arithmetic operation & logical operato

Arithmetic Operators
Data manipulation asuch as insert, update & delete can also be perfo rm by using arithmetic operators Such as

+	Addition
-	Subtraction
*	Multiplication
/	Division

Date: 18/7/18
MON TUE WED THU FRI SAT

CHAPTER-

Computation Done On Table Data

The Computations done on table data is done by using arithmetic operation & logical operator

Arithmetic Operators

Data manipulation Such as insert, update & delete can also be perform by using arithmetic operator such as

+ Addition
- Subtraction
* Multiplication
/ Division
** Exponent
( ) Enclosed bracket

Logical Operators:

AND: The AND Operator allows for creating the SQL Statements based on two or more conditions. It can be used in any SQL Statements such as Select, Insert, Update or Delete.

Display all those transection peroformed today for the amount ranging betn 500 - 5000; Amount, Period, Interest Rate

```
Select * from Trans_mstr where
amt >= 500 and amt <= 5000
    and To_char (DT, DD/MM/YYYY)
```

Date:
MON TUE WED THU FRI

OR: The OR condition requires that any of the condition must be satisfied for the record to be included in the result set.

Display the customer that belongs to I.T or are self employed
Customer, cust_no, F_name, L_name, O

Select * from customer mstr where OCC = 'I.T' or OCC = 'Self employed'.

Combining AND and OR Operators
The Oracle engine will process all the rows in a table & display the result only when all of the conditions specified using the AND Operator are satisfied & when any of the conditions specified using the OR operator is satisfied. O

NOT Operator
The Oracle engine with process all the rows in a table & display only those records that do not satisfy the conditions specified.

list the accounts details of those accounts which are niether single nor joint accounts
Account mstr

Date:
MON TUE WED THU FRI SAT

Acc no, Type, OPR_Mode, Status

select * from Account_msts where NOT (OPR_Mode = 'Singly' OR OPR_Mode = 'Joint');

Range Searching

To select the data that is specified within the range of values the Between Operators is used.

List the transections performed in the month of Jan. to March

Trans_msts
Amount, Period, Interest_Rate

Select * from Trans_msts where To_char(DT, 'MM') Between 01 AND 03

Select * from Trans_msts where To_char(DT, 'MM') >= 01 AND To_char(DT, 'MM') <= 03

NOT Betn Operators

The Oracle engine will display the result according to the conditions not specified.

Pattern Matching

The Like predicate allows the comparision of one string value with another string value which is not

identical or similar. This can be done by using different characters

1. % allows to match any string of any length (including 0 length)
2. _ allows to match on a single character

List the customers name that begin with the letters 'Ch'

Cust_mstr
F_name, L_name, DOB

```
Select * from cust_mstr where
(F_name LIKE 'Ch%');
```

List the customers names that have the second character as 'a' or 's'

```
Select * from cust_mstr where
(F_name LIKE 'a%' or 's%');
```

List the customers names that begins with the letters Iv & it is a fourth letter word.

```
Select * from cust_mstr where
(F_name Like 'Iv%_');
```

Date:
MON TUE WED THU FRI SAT

IN, NOT And NOT IN OPERATOR

The arithmetic operator equal to '=' compares a single value to the another single values in case a value needs to be compare to the list of values then inpredicate is used. The inpredicate helps to reduce the need of multiple or condition.

List the customer details of the customer name Hasel, Mamta, Namita, ARUNA.

Cust mstr (Fname, Lname, occup)

Select * from custimstr where Fname IN ('Hasel', 'Mamta', 'Namita', 'Aruna')

NOT-INPREDICATE

The not-Inpredicate is the opposite of inpredicate. This will select all the rows where the values do not match with the values in the list.

List the customer details of the customer than Hasel, Mamta, Namita, Aruna

Select * from cust mstr where Fname NOTIN (

Date:
MON TUE WED THU FRI SAT

ORACLE Table - DUAL

Select * from DUAL
Columnname NULL Datatype

SYSTEM : DATE
System date is a column that contains the current date & time with no arguments.

Select SYSDATE from DUAL
OUTPUT : 20-July-18

ORACLE FUNCTIONS
1. Aggrigate Functions
2. String Functions
3. Date Functions

1⇒ Aggrigate functions or the numeric functions are used for number datatypes.
'Returns the average value of 'n' By ignoring the null values in a column.

* AVG :- 'n'
AVG ([<DISTINCT>|<ALL>] <n>)

Select AVG(Price) from Cust_mstr;

* MINIMUM

Date:
MON TUE WED THU FRI SAT

Select MIN(Price) from cust_mstr;

It will return the minimum values.

* MAXIMUM

Select MAX(price) from cust_mstr;

It will return the maximum values. for all column

* COUNT

Select Count(Acc_no) from cust_mstr;

* SUM

Select SUM(Price) from cust_mstr;

a) Numeric Functions

1) ABS(n) :- Absolute

Select ABS(-15) from DUAL

OUTPUT :→ 15

2) Power(m,n) :- Power

Select Power(3,2) from DUAL

OUTPUT :→ 9

3) Round(n,[p,m])

Select ROUND(15.19,1) from DUAL

OUTPUT :→ 15.2

4) SQRT(n) :- Square root

Select SQRT(25) from DUAL

OUTPUT :→ 5

5) GREATEST(expr1, expr2, expr3, ...)

Select GREATEST(1,5,17) from DUAL

OUTPUT :→ 17

6) LEAST(exp1, exp2, exp3, ...)
Select LEAST(1, 5, 17) from DUAL

7) MOD(m,n) :- modulus (%)
Select MOD(15,7) from DUAL
OUTPUT: 1

8) TRUNC(number, [decimal_places]) :-
Truncate
Select TRUNC(125.815, 1) from DUAL
OUTPUT: 125.8

9) FLOOR(n)
Select FLOOR(24.8) from DUAL
OUTPUT: 24
It returns the largest integer value that is equal to or less than a no.

10) CEIL(n)
It returns the smallest integer value that is greater than or equal to a no.
Select CEIL(24.8) from DUAL
OUTPUT: 8

2 ⇒

1) LOWER.
It returns the character with all the letter in lowercase.

Date:
MON TUE WED THU FRI SAT

String Function

a) lower(char)
Select lower('Ivan Baymass') From dual;

b) INITCAP First letter should be capital.
It return a string with the first letter of each word in upper case.
select INITCAP ('ivan baymass') From dual;

c) Upper(char)
It returns characters with all the letter force to uppercase.
Select UPPER('Ms. Carol') From dual;
Output: MS. CAROL

d) SUBSTR
It returns the position or the portion of the character beginning at the character 'm' & going upto the character 'n'.
Select SUBSTR(<String>,<start position>, [<length>])
Select SUBSTR('SECURE', 3, 4) from dual;

e) ASCII
It returns the number code that represent the specified character. If more than one character is entered the func will return the value for the first character &

ignore all the characters after the first.

ASCII (single character)

Select ASCII ('A'), ASCII ('a') From dual;

Output: A → 65
a → 97

f) Compose

It returns the unicode string. It can be either a character or varchar, etc.

COMPOSE (single)

Select Compose ('a' || UNISTR ('\0301')) from dual;

g) INSTR

It returns the location of a substring in a string.

INSTR ('string', <start position>, <m^th appearance>)

Select INSTR ('SCT on the net', 't'); // 8
INSTR ('SCT on the net', 't', 1, 2)
from dual; // 14

h) TRANSLATE

TRANSLATE ('String1', 'String2', 'Replacement');

It replaces a sequence of characters in a string with another set of characters.

Date:
MON TUE WED THU FRI SAT

Select Translate ('1sct523', '123', 'Ta9')
from dual;
Output: 7sct5a9

i) Length
Length (word)

j) LTRIM
It removes the characters from the left of characters with initial characters removed upto the first character not in the set.
LTRIM (char, [, set])
Select LTRIM ('xyz', 'x') from dual;
output: yz

k) RTRIM
It returns the character after removes from the right of characters with initial characters removed upto the first character not in the set
RTRIM (char, [, set])
Select RTRIM ('xyz', 'z') from dual;
output: xy

l) TRIM
It removes all the specified characters either from the beginning or ending.

Date:
MON TUE WED THU FRI SAT

SYNTAX
TRIM ([leading|trailing|both [<trim char> FROM]] <string1>)

eg: Select TRIM (' Hansel ') from Dual;
o/p :- Hansel
Select TRIM (leading X from 'XXXHanselXXX') from Dual;
o/p :- HanselXXX
Select TRIM (Both x from 'xxx HanselXXx') from dual;
o/p :- Hansel
Select TRIM (Both '1' from '123 Hansel 12111') from dual

m) Left Padding [LPAD]

→ It returns character 1 left Padded to the length 'n' with sequence of character specified with character.

SYNTAX
Select LPAD (char1, n [char2])

eg:
Select LPAD ('Page1', 10 '*') from Dual;
o/p :
**** Page1

n) RPAD (Right Padding)

select RPAD('Ivan', 10, 'x') from Dual;

output: Ivanxxxxxx

Vsize

It returns the no. of bytes for the internal representation of the expression.

Select Vsize ('TELNET') from dual;

Output: 6

o) Conversion Function

It converts a character or the character value expressing a number to a number datatype

1) To_number (char)

SYNTAX

Eg: Update Student_mstr Set {DOB = + To_number ('Substr' ('$100'), 2, 3)};

Output: 10 - $100 Jan-2011

2) To_char (n, fmt)

Select To_char (17145, '$099,999') from Dual

Output: $017145

P) Date Conversion Function

1) To_Date

It Converts the character field to date field.

SYNTA To_Date (char, fmt)

Date:
MON TUE WED THU FRI SAT

Insert into stud_details (Std_no, Fname, M_name, L_name, DOB) values (Values ('S1', 'Ivan', 'Nelson', 'Bayross', To_Date ('25-JUN-1952 10:55 AM', 'DD-MM-YYYY HH:MI AM'));

2) ADD_Months
It returns the date after adding the no. of months specified in the fun^n.
SYNTAX: Add_months (system_date, 4)
from dual;
Output: 27-Nov-2018

3) LAST_DAY
It returns the last date of the month specified within the fun^n.
SYNTAX: LAST_DAY (DAY)
Select SYSDATE, LAST_DAY (SYSDATE)
from Dual;
Output: 31st-July-2018

Months bet^n, Next day, Round

Date:
MON TUE WED THU FRI SAT

Grouping of Data in SQL

Group By - The Group By clause creates a data set containing several sets of record grouped together on the basis of some specified condition

SYNTAX

Select <ColumnName 1>, <ColumnName 2>, <ColumnNameN> Aggregate function (<expression>) from tablename where <condition>
Group by <Columnname1><ColumnName2> <ColumnName N>

Find out how many employees are there in each branch.

emp_mstr (branch_no, emp_no)

B1	1
B2	2
B3	3
B4	4
B5	5

Select <branch_no>, Count(<emp_no>) from emp_mstr Group by <branch_no>;

Find Out the total no. of current & savings bank account Verified by each employee.

ACCT_MSTR (Verify_Acct, Acc_no, emp_no)

Date:
MON TUE WED THU FRI SAT

Select Verify_Act_emp, Count(Acc_no)
from Acct_Mstr Group by
Verify_Act_emp

Find out Total no. of Accts seperated on the basis of acc. type per branch
Acct_mstr (Br. no, Acct no, type)
, Br_no
Select type, Count (Acct_no) from
Acct mstr. Group by type, Br no

Having clause
The having clause is used with group by clause. Having clause imposes condition on the group by clause which further filters the groups created by the group by clause.

Find out the customers having more than one acct in a bank.
Acct_Fd_Cust_Dtls (Cust_no, Acc Fd_no)

Select cust_no, Count (Acc_Fd_no) from
Acct_Fd_cust Dtls where Group by
cust_no Having Count (Acc_Fd_no) > 1.

Find out the no. of accounts opened at a branch after 3rd-Jan-2003 only if the no. of acct open after 3-Jan-2003 exceed 4.

Date:
MON TUE WED THU FRU SAT

Acct_mstr (Br_no, Acct_no, OPNDT)

Select Br_no
Select Br_no, Count (Acct_no) from Acct_mstr where (OPNDT) to_char (OPNDT, 'DD-MM-YYYY') > '03-Jan-2003' group by branch_no having count (Act_no) >1;

List out the Cust no which are associated with only one acct in the bank
Acct_Fd_cust_DTLS (cust_no, Acc_Fd_no)

Group By using Rollup operator
The rollup operator is used to calculate aggregates and the super aggregates for the expression with in a group by statement.

Create a report on the fix deposit account available in the bank providing the amount & the due amount per fix deposit and per slot of the fix deposit held by the customer.

Date:
MON TUE WED THU FRI SAT

FDMSTR (FD Sr No, FD Name,

Select FD_SER_NO, FD_NO, SUM(AMT), SUM(DUEAMT) FROM FDMSTR Group by Roll up (FD_SER_NO, FD_NO)

FD SR NO	FD NO	Amt	Due Amt
FS1	F1	10,000	11,000
	F2	20,000	22,000
	F3	10,000	11,000
FS1	3FD	40,000	44,000

Group by using Cube operator

The Cube operator can be applied to all the aggregate fun^n like avg AVG(), count(), Sum(), max(), min() with in a group by statement. The Cube operator produces sub totals for all possible combination of the grouping specified in the group by clause with a grant total as opposite of roll up operator which produces only a fraction of possible sub total combinations.

Find out the balance of the account holders on per account & per branch basis along with the grant total.

Date.:
MON TUE WED THU FRI SAT

Acc mstr (Br_no, Acc_no, Sum of curbal)

Select Br no, Acc no, Sum(Sum of curbal) from Acc_mstr Group by Cube (branch_no, Acc no);

Brane	Acc_no	Sum(Sum_of_curbal)
B1		23000
B1	CA7	12000
B1	SB1	500
B1	SB11	500

SubQueries

A Subquery is a form of SQL Statement that appears inside another SQL statement. It is also being termed as nested query. The Statement Containing a Subquery is called as the parent statement. The parent Statement uses the rows written by the Sub queries.

- purpose of the Subqueries

1. To insert the records in the target table.
2. To create the tables and to insert the q record in the table created
3. To update the records in the table created or the target table.
4. To create the views.
5. To provide the values for

Date:
MON TUE WED THU FRI SAT

Conditions in where clause, heavy clause, in predicate and it is used with Select, Update and Delete statements

6. It is used in reference with parent table.

Retrive the address of a customer named 'Ivan Bayross'

cust_mstr, Add_details
(Cust no, F_name, L name), (Add1, Add2, city, State, pincode)

~~Select Add1, Add2,~~

Select Custno, Add1, Add2, State, City, Pincode from Address_details where cust no IN (Select cust_no from cust mstr where F.Name = 'Ivan' and Lname = 'Bayross');

Find the customers who don't have Bank branches in their Vicinity

TN: cust_mstr, add_dtls
cust mstr (cust no, fname, l name)
Add_dtls (code no, pincode)

eg

Date:
MON TUE WED THU FRI SAT

Select fname, lname from cust_mstr where cust_no IN (Select code_no from add_dtls where code_no like 'C%' and pincode NOT IN (Select pincode from add_dtls where code_no like 'B%'));

The 1st step is to identify the vicinities in which branches are located. This is done by extracting the pincode from add_dtls table for all the entries belonging to the branches.

The 2nd step is to identify the customer who are not resident to a nearby branch, to perform this the customer number i.e. code_no has to be retrived from add_dtls table.

The 3rd step the outer sub query will find out the customer name from the custo_mstr table by using cust_no from the cust_mstr table.

output: 'c' name.

Eg: List the customer holding fix-deposit in the bank account of the amount more than 5000.
TN: cust_mstr, acc_Fd, cust_dtls, fd_dtls.

Cust_mstr CN: Cust_no, f_name, l_name
acc_fd_cust CN: cust_no, acc_fd_no
Fd_dtls CN: fd_sr_no, amt

Select f_name, l_name from cust_mstr where cust_no IN (select cust_no, * acc_fd_no from acc_fd_cust_dtls where acc_fd_no IN (Select fd_sr_no from fd_dtls where amt > 5000));

List the acc_dtls with the current balance, the branch to which it belongs and the avg balance of the branch having a balance more than the avg. balance of the branch to which the acc. belongs.
Acct_mstr (Acc no, Cur_bal, Br_no)

Select A.Acc no, A.Cur bal, A.Br no, B.Avg_bal from Acct_mstr A (Select branch_no, Avg(curbal) Avg bal FROM Acct_mstr Group by branch_no) B where A.Br_no = B.Br_no And A.Cur_bal > B.Avg_bal

⇒ The first step is to identify the Branch number & their avg balance.
⇒ The second step is to associate the data returned by the inner que

with the outer query.

* Multi Columns Sub Queries

Find out all the cust having the same name as the employee.

Cust_mstr, Emp_mstr

cust_mstr(F_name, L_name)

emp_mstr(F_name, L_name, city, Add, Pincode)

⇒ Operators: IN

Select (F_name, L_name from cust_mstr) where (F_name, L_name) IN (Select F_name, L_name, city, Add, Pincode from Emp_mstr);

* Joins

Sometimes it is necessary to work with the multiple tables at that time joint statement is used to work on multiple tables. Tables in the Database can be related with each other by using different keys. There are three types of joints 1st inner join, outer join, cross join.

Inner join: Inner join is also known as equi join. They are known as equijoin because when the statement compares the two columns from the two table equivalents operation (=) is used. The inner join returns all the rows from both the tables where there is a match.

Outer Join: Outer join are similar to inner join but this kind of join can be used in a situation to select all the rows from the table on the left or right or both from the other table which has common values in it.

Cross Join: Cross join is more known as Cartesian product which means that join combines every row from the left table with every row in the right table.

SYNTAX:

Select <ColumnName 1>, <ColumnName 2> <ColumnName N> from <Table Name 1> Inner Join <TableName 2> On <TableName 1>.<ColumnName 1> = <TableName 2>.<ColumnName 2> where <condition> ORDER BY <columnName 1>

Date:
MON TUE WED THU FRI SAT

<ColumnName2>, <ColumnNameN)

List the employee details along with branch name to which they belongs

empl_mstr(em_no, F_name, M_name, L_name, Dept, Designation, br_no);

br_mstr(br_no, br_name);

Select * from empl_mstr Inner Join br_mstr on empl_mstr.br_no = br_mstr.br_no

Select * from empl_mstr E, br_mstr B where B.br_no = E.br_no;

List the customers along with their multiple address details.

Cu_mstr(cu_no, F_name, M_name, L_name)

Add_dtls(Code_no, Add1, Add2, City, State, Pincode)

Select * from Cus_mstr C Inner Join Add_dtls A on C.cu_no = A.code_no where C.cu_no like 'C%' ORDER BY C.cu_no;

Select * from Cu_mstr C, Add_dtls A where C.cu_no = A.code_no And C.cu_no LIKE 'C%' ORDER By C.cu_no;

Date:
MON TUE WED THU FRI SAT

Outer Join

List the employee details along with Contact dtls (IF ANY) By using left Outer Join

em_mstr (em_no, E_name, M_name, Dept);
Contact_dtls (Code_no, Con_type, Con_data);

Select * from em_mstr E Outer Join Contact_dtls C on E.em_no = C.code_no ORDER BY E.em_no;

Select * from em_mstr E, Contact_Dtls C where E.em_no = C.code_no (+);

Select * from em_mstr E, contact_dtls C where C.code_no(+) = E.em_no;

Cross Join

Sometimes it is desired to combine the various entities or columns with the another entities to calculate their minimum & maximum amount. This can be done by using a cross join.

Create a report using cross join that will display the amount for the predefined deposit based on the min. & max. period of time for that deposit.

Temp_Fd_amt (Fd_amt);
Fd_mstr (min_p, max_p, int_rate);

Select T.Fd_amt from Temp_Fd_amt T
Cross Join Fd_mstr F on

Select T.Fd_amt, S.min_p, S.int_rate,
S.max_p Round (T.Fd_amt + (T.Fd_amt * (S.int_rate
/100) * S.min_p/365) from Fd_mstr S Cross
Join Temp_Fd_amt T;

Self Join

In some situation it is necessary to join the table to itself as joining the two separate table this is reffered as self joint. In a self joint the two rows from the same table are combined to form the resultant row. To join the table to itself two copies of very same table are opened in the memory. Hence by using the 'from' clause the table name records needs to be mentioned twice since the table name are same the second table will overwrite the first table & finally the resultant table is only one table that is saved in specific memory location. To avoid this each table is opened using an alias.

Date:
MON TUE WED THU FR SAT

Retrive the name of employees & the names of the respective managers from the employee table

Emp_mngr (Emp_no, Fname, Lname, mng_no);

Select * from Emp_mngr E where E.Emp_no = E.mng_no;

Union, Intersection, Minus clause

→ The Union clause merges the outp of two or more query into the sing set of rows & columns.

Rules

→ Union Operator of all the columns sho ld be selected.

→ The IN Operator has the higher prece dence than the union operator

→ Null values are not ignored duri the duplication of operator.

Retrive the names of customers & employees resided in the city mumb

Cust_mstr (cust_no, F_name, L_name);
Add_dtls (code_no, city);
em_mstr (em_no, code_no);

Date:
MON TUE WED THU FRI SAT

*

Select E.Ename, C.t_name from Emp_mstr E, Add_dtls where E.emp_no = A.Mbls_code_no And A.city = 'Mumbai' And A. Code_no LIKE '6%'

Intersection

Multiple queries can put together & their output are combined using the intersect clause. The intersect clause outputs only rows produce by both the queries intersected that means the output in an intersection clause will include only those class that are required common to both the query.

Retrive the customer holding the accts as well as a fixed deposit in the bank

cust_dtls (Cust_no, acc_Fd_no)

Select DISTINCT cust_no from cust_dtls where acc_Fd_NO LIKE 'CA%' or acc_Fd_no LIKE 'SB%' INTERSECT Select Distinct cust_no from cust_dtls where acc_Fd_no LIKE 'FS%'

Date:
MON TUE WED THU FRI SAT

MINUS Clause

Multiple queries can be put together and their output is combined using the minus clause. The minus clause output the rows produce by the first query after filtering the rows retrieve by the second query.

Retrive the customers holding the account but not holding any fixed deposit in the bank.

```
Select DISTINCT cust_no from cust_dtls
where acc_Fd_no LIKE 'CA%' or Acc_Fd_
LIKE 'SB%' MINUS Select DISTINCT
cust_no from Cust_dtls where acc_Fd_no
LIKE 'FS%'
```

Date: 7/9/18
MON TUE WED THU FRI SAT

Introduction To PL/SQL Language

Disadvantage of Conventional Programming Language.

1. SQL doesn't have any procedure capabilities such as condition checking, looping & branching for data testing before it's permenant storage.
2. SQL statements are passed to the oracle engine one at a time each time an SQL statement is executed. which increases the traffic on the network & decreases the speed of data processing.
3. By processing an SQL statements, if any error occurs the oracle engine display it's own error message. SQL doesn't have the facility for programm handling errors that arise during the manipulation of data

Advantage of PL/SQL

1. It provides facilities of conditional checking, branching & looping.
2. PL/SQL sends an entire block of SQL statements to the oracle engine at one time. All the changes made to the data in the table are done or undone at one time.
3. PL/SQL also permites detete with the errors as required & facilates the displaying user freindly messages when

errors are encountered.
4. It allows the declaration & use of variables in the block of quote.
5. All sort of calculation can be done quickly & efficiently without the use of oracle engine which improves the transaction performance.
6. Application return in PL/SQL block are portable to any computer hardware & operating system.

Control Structure
1. Conditional Control
2. Iterative Control
3. Sequential Control.

Syntax
If <Condition> Then <Action>
Else if <Condition> Then <Action>
Else <Action>
End if;

Variables :-
→ Variables in PL/SQL block are named. Var. A Variable name must begin with character & it can be followed by maximum of 29 character.
→ Reserved words cannot be used as a variable unless it is enclosed with double quote.

Date:
MON TUE WED THU FRI SAT

A space cannot be used in a variable name.
The datatype of a variable can be number, char, date & native such as boolean expression can be declared.

Assigning Values to Variable:
It can be done in a two ways
1) Using assignment operator (:=)
2) By selecting or fetching table data values into variables.

ROWID
This datatype is same as the database rowid, column type.
→ It can holds a ROWID which can be considered as a unique key for every row in the database.
→ ROWID's are stored internally as a fix length binary quantity whose actual fixed length varies depending on operating system

Displaying the user message on the screen.
DBMS_OUTPUT.PUTLINE
→ DBMS_OUTPUT is a package that includes the number of procedures & function.
→ PUT_LINE puts the piece of information in the package buffer followed by

the end of line.

Write a PL/SQL Block that will accept acc-no from user.

```
Declare
    mAcct_no Varchar(50)
    Acct_no Varchar(7);
Begin
    dbms_output_put_line('Enter
    Your account number')
    mAcct_no := &mAcct_no;
    dbms_output_put_line('The a/c
    no is' mAcct_no).
End
```

Find Out the area of triangle by PL/SQL.

```
Declare
    base number(5)
    height number(5)
    area number(10)
Begin
    dbms_output_put_line('Enter
    Value for base')
    base := & base;
    dbms_Output_put_line('Enter
    Value for height')
    height := & height;
    area = 0.5 * base * height;
    dbms_output_put_line('The area
    of triangle is' area)
End
```

Write a PL/SQL block to check whether the person can vote or not

```
Declare
    age number(3);
Begin
    dbms_output.put_line('Enter the age of person');
    age := &age;
    If (age >= 18) Then
    dbms_output.put_line('Person can Vote');
    Else
    dbms_output.put_line('Person can't Vote')
    Endif
End.
```

Write a program to find maximum numbers among three numbers

```
Declare
    a number(5)
    b number(5)
    c number(5)
```

```
Begin
dbms_output_put_line ('Enter
    number a')
    a := &a;
dbms_output_put_line ('Enter
    number b')
    b := &b;
dbms_output_put_line ('Enter
    number c')
    c := &c;
If (a>c && a>b) Then
dbms_output_put_line ('a is
    greater')
Elseif (b>a && b>c) Then
dbms_output_put_line ('b is
    greater')
Else
dbms_output_put_line ('c is
    greater')
Endif
End
```

Iterative Control

Iterative control statements indicates the ability to repeat or to skip the section of block of code.

A loop marks the sequence of statement to be repeated for the n no. of times.

Date.:
MON TUE WED THU FRI SAT

The keyword 'END LOOP' indicates the sequence of the statements to be terminated. Hence, a conditional statement that controls the number of times a loop is executed is always accompanied by loop statement.

SYNTAX : Loop
< Sequence of Statements >
End Loop;

Create a simple loop such that a message is displayed when the loop exceeds a value of 10.

Declare i number := 0;

Begin
Loop
i := i+1;
Exit when i>10;
END Loop;
dbms_output.put_line('Loop exited as the value of i is' || to_char(i));

End

Date:
MON TUE WED THU FRI SAT

While loop

Syntax:

```
while <condition>
  loop
    <Action>
  End loop;
```

Write a PL/SQL block to calculate the area of circle for the value of radius varying from 3 to 7.

```
Declare radius number(3);
        area number(3);
        pi constant number(4,2) := 3.14;
Begin
  dbms_output.put_line('Enter the radius')
    radius := &radius;
  area := 3.14 * radius * radius;
  while (radius >= 3 And radius <
    radius := 3
  while radius <= 7
  loop
    area := pi * power(radius,2);
    radius := radius + 1;
  End loop;
End
```

Date:
MON TUE WED THU FRI SAT

FOR loop

Syntax:

For Variable IN [Reverse] start .. end
LOOP
<Action>
END LOOP;

Write a PL/SQL Block for inverting a no. 5639 to 9365

```
Declare a number(5);
Begin a := 5639
  For a IN [Reverse]

Declare given_number Varchar(5) := '5639';
        Str_length number(2);
        inverted_number Varchar(5);

Begin
  Str_length := length (given_number);
  For Cnta IN Reverse 1 ... Str_length
  Loop
  inverted_number := inverted_number
      || Substr (given_number, Cnta, 1)
  End loop;
dbms_output.put_line ('|| given_num. ||');
  dbms_ || inverted_number

End.
```

Date:
MON TUE WED THU FRI SAT

Sequential Control

The go to statement changes the flow of control within a PL/SQL Block. This statement allows the execution of the section of a code which is not in the normal flow of control. The entry point in such block of code is marked using tag << user-defined name >>

PL/SQL block using CURSOR.

The Oracle engine uses the work area for its internal processing in order to execute the SQL statement. This work area is private to the SQL operation and it is called as a CURSOR.

The data that is stored in a CURSOR is called as active data set. The size of the CURSOR in the memory is the size required to hold the number of rows in the active data set.

Types of CURSOR

There are two types of CURSOR

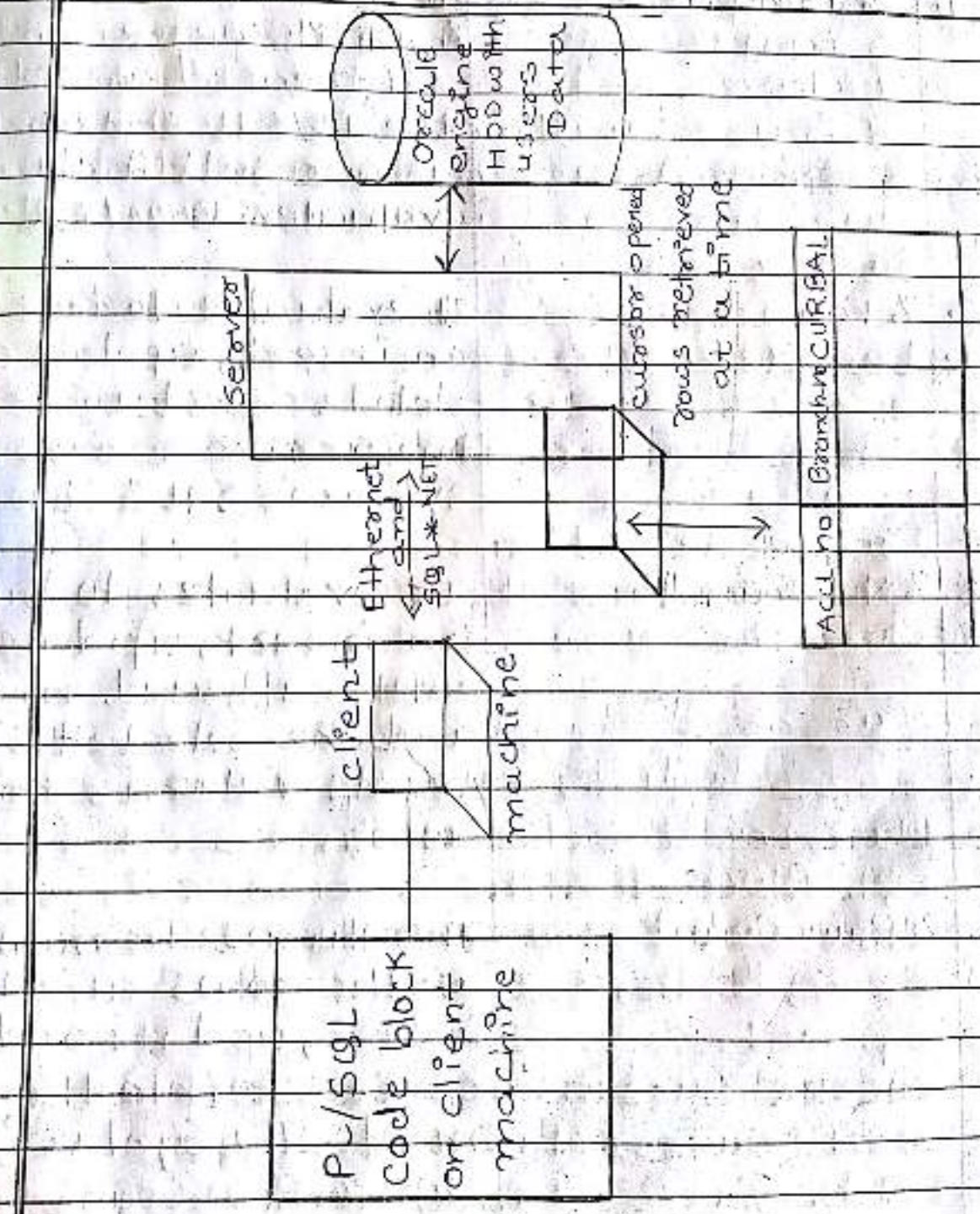
1. Implicit CURSOR
2. Explicit CURSOR

1. ⇒ If the Oracle engine open a CURSOR for its internal processing. It is known as implicit CURSOR.

Implicit CURSOR has the different

Data:
MON TUE WED THU FRI SAT

attributes that can be used to access the information about the status of last insert, update, delete or single row select statement.



Date: MON TUE WED THU FRI SAT

Implicit cursor attributes can be defined as follows (inserts

	Attribute Name	Description
1.	% ISOPEN	This attribute of an implicit CURSOR can't referenced outside the PL/SQL statement as a result it always evaluates True/False
2.	% Found	It evaluates to true if an insert, update or delete Select to return one or more rows. The syntax is SQL % Found
3.	% Not Found	It evaluates to true if an insert, update or delete affected no rows and the Selected Statements returns no rows.
4.	% Row Count	It returns the number of the rows affected by insert, update or delete or Select into the statements. The Syntax is SQL % Row Count

Scanned by CamScanner

Data:
MON TUE WED THU FRI SAT

There are major four operations that can be apply on the implicit cursor that are inserts, update, delete or select.

Ex: The bank manager has decided to transfer the employees across the branches.

Write a PL/SQL block to accept the em_no and the Br_no followed by updating the Br_no of that employee to which he/she belongs. Display an appropriate message by using the record from emp_mstr table.

emp_mstr(em_no, Br_no, em_add, em dtls)

```
Begin
  Update Emp MSTR Set branch_no =
&branch_no where Emp_No = &Emp_no;
  IF SQL%Found Then
    dbms_output.put_line('Employee
Successfully transferred')
  End if;
  If SQL%NOT FOund then
    dbms_output.put_line('Employee
  Number does not exist');
  End if;
END;
```

Date:
MON TUE WED THU FRI SAT

Explicit Cursor

When in individ. records in the table have to be processed inside the PL/SQL Block. The cursor is use This type of cursor will be declared & mapped to an SQL query in th declare section of the PL/SQL Block it is used within it's executable section. A Cursor thus created and which is used is known as explici Cursor.

1. Declare a Cursor
2. open a Cursor
3. Fetch data from the cursor one row at a time.
4. Process the data
5. Exit from the loop
6. Close the Cursor.

Pg

Cursor declaration

Initialization of a cursor takes place in the declarative part of PL/SQL Block. The three commands which are used to control the cursor are open, fetch, & close.

Syntax:

```
CURSOR CursorName IS Select Statement;
OPEN CursorName;
Fetch CursorName INTO Variable1,
        Variable2....,
```

Date.:
MON TUE WED THU FRI SAT

Close CursorName;

* The explicit cursor is open for its working area by using the open command.
* Fetching record from the CURSOR
  The Fetch statement retrives the rows from the active set open in the circle into the memory variables declared in the PL/SQL Block of code
* Closing a cursor
  The close statement disables the cursor and the active set becomes undefined. This will release the memory occupied by the cursor & its data set both on the client site & as well as server site.

Pg→ 36 Write a PL/SQL block that will display customer name, fixed deposit number & fixed deposit amount of the first five customers holding the highest amount in the fix deposit.

⇒ Create table cust_mstr (cu_name Varchar (20), Fd_no Varchar(10), FD_Amt Varchar (5))

Declare
  Cursor C1 IS select * from cust_mstr C, Acc_dtls A where C.cust_no = A.cust_no
  ORDER BY AMT Desc;

Begin

Stored procedures

A procedure or a function is a group set of SQL and PL/SQL statemen that perform a specific task. A stored procedure or a fun^n is PL/SQL code of block that has been compiled & stored in one of the Oracle engine System tables. To make a procedure or function either of them can be pa ed parameters before the execution procedures & functions are made up of

1) A Declarative Part
2) An Executable Part
3) Exception handling

1.) The declarative part may contains the declaration of cursor, constans,

Date:
MON TUE WED THU FRI SAT

Variables & Subprograms. This object are local to the function or to the procedure.

2-> The executable part contains the assigned values control the execution & manipulate the data.

3-> This part contains the code that deals with the exception that may be praised during the execution of the code in the executable part.

How does the Oracle engine executes the procedure or function.

1) Verifies the user access
2) Verifies procedure or function validity
3) Executes the program that contains procedure or function.
4) The Status of the procedure or function is shown by using the Select Statement as follows

Select <object Name>, <object Type> <STATUS> FROM <user objects> where <object Type> = <'procedure'>;

Date:
MON TUE WED THU FRI

Difference between Procedure & function

1. A fun^n must return on a value back to the caller.	By defining multiple out parameter
2. A fun^n can return only one value to the calling block.	
3. By defining multiple out parameter in a procedure multiple values can be passed to a calling fun^n	
4. The out variable can be a global variable & its value is accessible by any PL/SQL block of code including the PL/SQL block which it was called.	

Date:
MON TUE WED THU FRI SAT

Syntax (Procedure)

Create or replace procedure [schema.]
<procedure Name> (<Argument>
{IN, OUT, IN, OUT} <Datatype>, ...)
{IS, AS} <variable> declarations;
<constant> declaration is
Begin
<PL/SQL subprogram body>
Exception
<Exception PL/SQL block>;
End;

Keyword	Description
IN	Indicates that parameter will accept the value from the user.
OUT	Indicates that parameter will return a value to the user.
Datatype	It is the datatype of an argument it supports all the datatypes that are supported by PL/SQL block
Schema	It is the schema that contains the procedure

Date:
MON TUE WED THU FRI

Syntax for Creating a fun^{n}

Create or Replace function [schema]
<Function Name> (<Argument> IN
<Datatype>,...)
Return <Datatype> {IS, AS} <Variable
declarations> <constant declarations>

Begin
<PL/SQL subprogram body>;
Exception
<Exception PL/SQL block>;
END;

	Keyword	Description
→	Function	It is the name of a fun to be created.
→	IN	It indicates that the parameter will accept value from the user.
→	Return <Datatype>	It is the datatype of the fun^{n} returning a value.

How to delete a stored procedure or function

Drop procedure <procedure fun^{n} name>
Drop function <fun^{n} name>

Date:
MON TUE WED THU FRI SAT

Views

- To reduce the redundant data to the minimum possible amount oracle allows the creation of an object called as Views.
- Views is stored only as a definition in oracle System Catalog. The reasons for creating a view.
- For Data Security
- When Data Redundancy is to be kept to the mining of level it is used.

Syntax for Creating a View

Create View <viewname> As Select <Columnname1> <Columnname2> from <Tablename> where <ColumnName> = <Expression list>; Group By <grouping Criteria> having <predicate>

NOTE: ORDER BY Clause can't be used while creating a View.

Create a View called Emp on Empmst table

Create View Emp As Select * from Empmstr;

Select * from <viewname>; → To display a View

Date:
MON TUE WED THU FRI SAT SUN

To insert values in a View

insert into Viewname values ('', '', '')

For performing modification

Update View Viewname set where <cond:

To drop/delete a View

Drop view <Viewname>

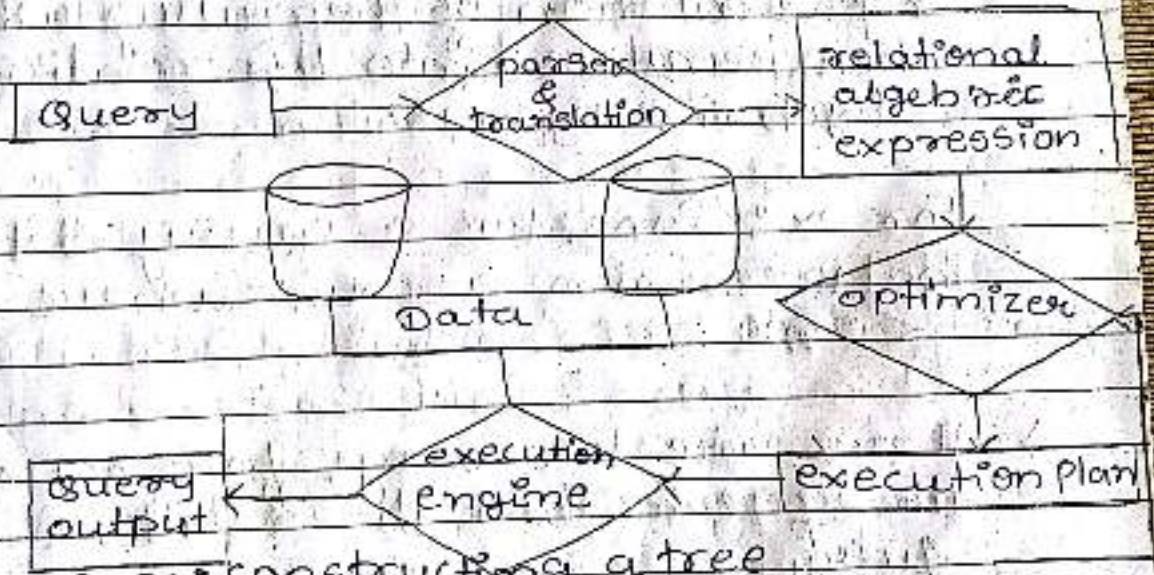
NOTE When a View is created by using the reference clause an insert, delete or update can't be perform on the Vie

Date: 26
MON TUE WED THU FRI SAT

Query Processor

→ Query processing refers to the range of activities involve in extracting the data from the database.
The activities include translation of the queries in high level database languages into the expressions that can be used at the physical level of the file system. for p. query optimization and actual evaluation of the queries

Steps in Query Processing



parser: constructing a tree
translator: convert low level lang.
optimizer: Select best way in giv. Soln

→ Working :-
1. → Parsing & translation
2. → optimization
3. → Evaluation

Date:
MON TUE WED THU FRI SAT

1. → The system must take in query processing to translate a given query into its internal form. This translation process is similar to the work performed by the parser of a compiler.

2. ⇒ To generate the internal form of a query the parser checks the syntax of the user query that verifies that the relation name appearing in the query are the names of the relations stored in the database

3. ⇒ The system constructs a parse tree representation of the query which is then translated into the relational algebraic expression.

For ex: Consider a query Select balance from account where balance < 2500

→ There are two ways for representing the queries into the relational algebraic query.

i) $\sigma_{balance < 2500}$

ii) $\Pi_{balance}(\sigma_{balance < 2500}(account))$

Date:
MON TUE WED THU FRI SAT

A Relation operati

→ A Relational algebric expression operation denoted with instruction or how to evaluate a query is called as evaluation primitive.

→ A sequence of relative primitive operation that is ran be used to evaluate a query is called as query execution plan. or query evaluation plan.

$\Pi_{balance}$
|
$\sigma_{balance < 2500}$
|
account

$\sigma_{balance < 2500}$
|
$\Pi_{balance}$
|
account

query evaluation plan

query evaluation plans

The query execution engine takes a query evaluation plan executes that plan & return the answer to the queries.

The different evaluation plans for a given query can have the different cause, so the query evaluation plan will be the less cause will be prefer.

Date:
MON TUE WED THU FR SA

Write down a program to create a Stored procedure insert, update and delete into ACCT MSTR table field - ACCT NO, ACCT Type, Open Date and Balance

⇒ Create table ACCT MSTR (ACCT No number(10) NOT NULL, Acct. Type varchar (10) NOT NULL, OPEN-DATE Date, Balance number (10, 2) NOT NULL);

1. Create procedure for insert

Create or replace procedure insert ACCT MSTR (Pacct no IN ACCT MSTR. ACCT NO %. Type Pacct type IN ACCT MSTR. ACCT Type %. Type Paper date IN ACCT MSTR. OPEN Date %. Type pbalance IN ACCT MSTR. Balance %. Type);

Measures of Query Cost

The cost for query evaluation can be measured in terms of number of different resources such as disk access, Cpu time to execute a query and in a distributed or parallel database system.

The cost of communication.

The number of block transfers from a disk is measure as a actual cost. More &

i)

Date:
MON TUE WED THU FRI SAT

For more accurate measure we can evaluate.

1. Number of seek operations performe.
2. The number of blocks read.
3. The number of blocks written and then add up these numbers after multiplying them by the average seek time. Average transfer time for reading a block & average transfer time for writting a block.

Operation

The Selection operation search through the data file for the records meeting selection criteria. Foll are the ex. of it:

i) Binary Search (Linear search) (Ctrl+F)

ii) Index Searching.

i) ⇒ Every record from the file is read & Compared to the Selection Criteria the execution cost for searching a null key attribute is b_R (block read) where b_R is the no. of blocks in the file representing relation R. on a key attribute the average cost is $b_R/2$ and for the worst case it is b_R.

Date:
MON TUE WED THU FRI SAT

Advantages
1. A linear search can apply to any file.

Disadvantages
1. It is slower.
2. It can be applied to only one file at a time.

Binary Search by using a primary key
In a binary search on equality perform on a primary key attribute has a worst case cost of $\log(b/2)$. This can be applied more efficiently than the linear search for the large no. of records & equality comparison on a null key attribute will be same except the multiple records that meets the condition, for which we can add no. of blocks containing the records to the box cost.

Complex Selection Operations
1. Conjuction
A Conjuctive selection is a selection of the form which can be represented as $\sigma_{\theta_1} \wedge \theta_2 \wedge \cdots \wedge \theta_n(r)$

2. Disjunction
A disjunction selection is a selection of the form which can be

represented as $\sigma_{\theta_1 \vee \theta_2 \vee \cdots \theta_n}(r)$.

4. Negation:

The result of a expression selection operator Such as $\sigma_\theta(r)$ is set of tupples of r for which the condition θ evaluates to false. We can implement a selection operation by using either a conjuction or disjunction for the simple conditions of the algorithm.

Algorithm

1) Conjuctive Selection algorithm using one index

We first determine wether an access path is available for the attribute in one of the simple conditions if one path is available then the selection algorithm can retrive the record by satisfing the condition.

2) Conjuction Selection by intersection of the identifiers.

This Algorithm requires indices with record pointers on the fields involved in the indivision conditions.

This Algorithm scans each index for the pointer to tupples that satisfies an individual condition.

The intersection of all the retrived pointe is the set of pointers to the tupples that satisfies the conjuctive condition. This algorithm that uses the pointe to retrive the actual records, the co can be calculated by sorting the list pointers and retriving the records in the sorted order by the foll. conditi

(i) All pointers to records in a block Con together hence all the selected record in the block can be retrived using single input output i/o operations

ii) Blocks are read in a sorted order it minimises the movement of the disk

3) Disjunctive Selection by union of identifiers :-

If the access path are availabl on all the conditions of a disjunctive Selection each index is scanned for the pointer to the tupples that satisfies the individual conditions.

The union of all the retrived pointer gives the set of pointer that satisfies the disjunctive conditions

Date:
MON TUE WED THU FRI SAT

* Join Operations
Like Selection operation the join operation can be implemented in a different ways that are basically 4 types of join algorithms.

(i) Nested loop join :-
This algorithm consist of inner for loop nested within an outer for loop it use the foll. notations
r, s Relations r and s
tr tuple (records) in relation r
ts tuple (records) in relation s.
nr Tuple Number of records in relation r.
ns Number of records in relation s
br Number of blocks with records in relation r
bs Number of blocks with records in relation s

J1 : Nested-Loop Join
for each tuple tr in r
for each tuple ts in s
if join Condition is true for (tr,ts)
add tr+ts to the result.
end
end
each record in the outer relation are scanned once & each record in the

inner relation s is scanned nr * bs times which results into nr * bs total scans.

→ If only one block of each relation can fit into the memory then the cost will be nr * bs + br.

→ if all the blocks in both the relations can fit into the memory then the cost will be bs + br.

→ if all the blocks in a relation S can fit into the memory then the cost will be identical to both relations fit into the memory i.e br + bs.

(ii) Index Nested-Loop Join

This algorithm is same as nested-loop join except an index file on the inner relation. Join attribute is use to scan the data find on each index look up in the inner loop is essential for the equality selection of details. The cause for the look up is denote by c. and the worst case for joining r & s relation is br + nr * c

(iii) Block Nested-Loop Join

for each block Br of r do begin
for each block Bs of S do begin
for each tuple tr in Br do begin
for each tuple ts in Bs do begin

Date:
MON TUE WED THU FRI SAT

test pair (tr, ts) to see if they satisfy if they do, add tr to ts to the result of join condition

end

end

end

end.

In block nested loop join every block of the inner relation is paired with every block of the inner outer relation. Every tuple in one block is paired with every tuple in the other block to generate all pairs of the tuples. The primary diff in the cost between block nested loop join & basic nested loop join is that in the worst case each block in the inner relation s is read only once for each block in the outer relation instead of once for each tuple in the outer relation. Thus in the worst case there will be total br * bs + br block access where br & bs denotes the no. of blocks in relation r & s

Thus in the worst case total

worst case → 100 * 400 + 100

→ 40100

best case → 400 + 100

→ 500

Sort merge join

```
pr := address of first tuple of r;
ps := address of first tuple of s;
while (ps ≠ null & pr ≠ null) do
begin
    ts := tuple to which ps points;
    Ss := {ts};
    Set ps to point to next tuple of s;
    done := false;
while (not done and ps ≠ null)
    do
    begin
    ts := tuple to which ps points
    if (ts [join Attrs] = ts (join Attrs))
    then begin
    Ss := Ss ∪ {ts}
    set ps to point to next tuple of s;
    end
    else
        done := true;
end
    tr := tuple to which pr points;
which (pr ≠ null & tr [join Attrs] <
    ts [join Attrs]) do
begin
Set pr to point to next tuple of r;
tr := tuple to which pr points;
end
while (pr ≠ null & tr [join Attrs] = ts
    [join Attrs]
```

Date:
MON TUE WED THU FRI SAT

do
begin
for each ts in Ss do
begin
add ts and to to result;
end
Set pr to point to rj;
tr = tuple to which pr points to;
end
end

This algorithm can be used to perform natural join & equijoin & requires that each relation R & S be sorted by the common attributes betn them i.e (R ⋈ S) It is notable to point out that each record in R & S is scanned only once by producing the best & the worst case of both sort merge join operations or algorithms are used when the data files are in the unsorted order but had exist secondary indices for the two relations.

Hash join
Like the sort merge join the hash join algorithm can be used to perform natural join & equijoin. The hash join utilizes to hash table file structure to partition each relations

Date:
MON TUE WED THU FRI SAT

record into the sets containing identical hash values on the join attributes. Each relation is scanned & its corresponding hash table on the join attribute values is built.

Collision may occur which results in some of the partition containing diff. sets of records with matching partition in the hash table or the attributes value.

After the two hash table are built hash index of the smaller relation records is built & a nested loop join is performed against the corresponding records in the outer relation.

The hash join algorithm works only if the required amount of the memory is available to hold the hash index & the no. of records in any partition of built relation if not then the process is known as recursive partitioning.

The cost for computing the hash join without recursive portion is $3(b_r + b_s) + 4n_h$ that n_h is the member of partion in the hash table & with recursive it is (M).

M = no. of memory block is used.

for each tuple ts in S do begin
i := h(ts[joinAttrs]);
Hsi := Hsi ∪ {ts};
end
for each tuple tr in R do begin
i := h(tr[joinAttrs]);
Hri = Hri ∪ {tr};
for i = 0 to n do begin
read Hsi and build an in-memory hash index
for each tuple ts in Hsi do begin
probe hash index on Hsi
for each matching tuple ts in Hsi do begin
add tr and ts to the result
end
end
end

Sorting

g	24
a	19
d	31
c	33
b	14
e	16
r	16
d	21
m	3
p	2

d	7
a	14

a	19
d	31
g	24

b	14
c	33
e	16

d	21
m	3
r	16

a	14
d	7
p	2

a	19
b	14
c	33
d	31
e	16
g	24
a	14
d	7
d	21
m	3
p	2
r	16

a	14
a	19
b	14
c	33
d	7
d	21
d	31
e	16
g	24
m	3
p	2
r	16

Sorting in dbms is important for the two reasons

i) Output can be display in a sorted order
ii) Several relational operations can be implemented efficiently if input relations are first sorted.

The external-merge sort algorithm is explain as above let M denotes the memory size.

i) Create the sorted runs
- Let i be 0 initially. Repeat the procedure till the end of the relation
  a) Read in blocks of relation into the memory
  b) Sort the in memory blocks.
  c) Write the sorted data to run R_i, increment the value of i. Let the final value of I be N.

ii) Merge the runs
- We assume that N is less than M (N<M)
  a) Use in blocks of memory to buffer input runs and one block to buffer the output runs
  b) Repeat
     1. Select the first record in the sorted order among all the buffer pages
     2. Write the record to the output buffer
     3. Delete the record from it's input buffer page.

Date:
MON TUE WED THU FRI SAT

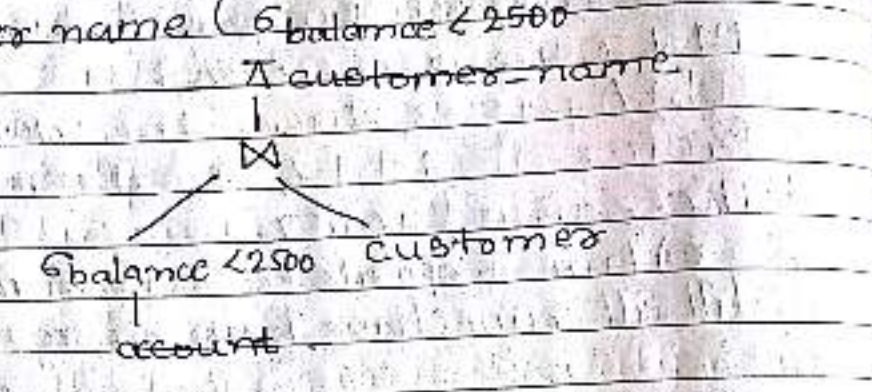
c) Perform the process until on input buffer pages are empty.
iii) If $i \geq M$ several merge passes are required
a) In each pass continuous loops of M-1 runs are merge
b) A pass reduces the no of runs by a factor of M-1 & create runs longer by the same factor.
c) Now Repeated passes are performed till all runs have been merged into one.

- The total number of the disc access is $br\ (2\lceil \log_{M-1}(br/M) \rceil + 1)$

evaluation of expression
one way to evaluate the expression is simply to evaluate an expression at a time in an appropriate manner. The result of each evaluation is materialized in a temporary relation for the subsequence use.
The disadvantage of this approach is that it is needed to construct the temporary relation which must be return to the disk an alternative approach is to evaluate several operations one by one in a pipeline where the result of one operation is passed to the next operation without the need to store a

temporary relation -

Materialization



The pictorial representation of given query is shown here.

If we apply the materialization approach we start from the lowest level of the operation in the given expression we can use the temporary files all the relation to execute the operation at the next level up in the database. In our example the input to the join are the customer relation and the temporary relation created by the selection on account. This join can now be evaluated by creating another temporary relation.

The cost of evaluating an expression is the addition of all the operation with their cost as well as the cost of writing intermediate result to the disk.

Date:
MON TUE WED THU FRI SAT

Pipelining

We can improve query evaluation efficiency by reducing the number of temporary files that are produced. We can achieve this reduction by combining several relational expression into the pipeline of operations in which the result of one operation is passed along to the next operation in the pipeline. This type of evaluation is called as pipeline evaluation.

For ex: Consider the expression

$(\Pi_{a_4, a_2}(r \bowtie s))$

when the join operation generate its result it passes that tuple immediately to the project operation for processing by combining the join & the projection we can avoid to create the intermediate result.

one by one execution
↳ sub routine
↳ f

Execution of Pipeline

⇒ To execute a pipeline

i) Demand driven

ii) Procedure driven

i) ⇒ In demand driven pipeline the system makes the repeated request for the tuples from the operations at the top of the pipeline.

ii) ⇒ In procedure driven pipeline operation don't wait for the request to produce the tuple but instead generate the

tuples egarely.
- Each operation at the bottom of a pipeline continuously generates the output tuples & put them into the output buffer.

Data Security
As Managing the computer & the cyber / network security programs has become a difficult & one of the challenging jobs. The information security manager must establish and maintain a security program which ensures the three requirements
i) Confedentiality
ii) Integerity
iii) Availaibility of the information resources of the Company. hence, it is necessary to provide the authencity as well as the authority to the user.
There are different types of threats that can affect the confedentiality

Confedentiality : It is the ~~Accessed~~ protection of information on the System so that the unauthorized person can't access the message.
Confedentiality must be well defined which can be use for a single

Date.:
MON TUE WED THU FRI SAT

computer.
A coucial/critical aspects of the confedentiality is user identification and authentication. Positive identification of each system users is essential to ensure the effectiveness of the policies that specifies who is allowed to access which data item.

Database Security.
Database security is concerned with the privacy of data. Privacy of the database protecting the database from unauthorized access by the users and software applications. that are generally three categories of secrecy related problem in database system.
i) The improper release of information from reading the data that was intentionally or accidently access by unauthorized user
ii) The improper modification of data
iii) Denial of service attack / DOS attack
- This threat has been demonstrated to a significant degree with the flooding attacks against network service provide
- Database provide many layers and the types of information security that is specified in the data dictionary which includes 1st access control.
2. Auditing floodi
3. Authentication.

4. Encryption
5. Integrity controls

1. An access control system is a system which enables an authority to control the access to the areas and resources in a given computer base information system.
Ex: A lock on a car door
A Pin in an atm system
A Password in a person computer
access control system provides essential services of identification & authentication authorization & accountability.

Types of Access Control techniques
i) Discretionary Access control (DAC)
ii) Mandatory Access Control (MAC)

i) → It is a means of restricting access to objects base on the identity of the subject & the groups to which they belong. The controls are discretory which means that a subject with the certain access permission is capable of passing or transfering the permission passing that to any other
There are two important concept in DAC mechanism
a) File and Data Ownership
→ Every object in the system has an

Date:
MON TUE WED THU FRI SAT

Owner in most of the discretory access control system each object initial Owner is the subject that called cause it to be created. The access policy for an object is determine by it's owner.

b) Access Rights and Permission

⇒ These are the controls that an owner can assign to the other subjects for the specific resources.

ii) ⇒ MAC is the access policy determine by the system but not by owner. MAC is used in multilevel system that process highely sensitive data such as military information & classified government work. This multilevel system is a single computer system that handles multiple classification levels between the subjects & the objects. The two important terms which are related to MAC are

a) Sensivity labels
b) Data Import & export.

a) ⇒ All the objects & subject in a MAC base system must have the labels assigned to them.

b) ⇒ Controlling the import of information from the other system & export to the other system is a critical function of MAC based system.

Date:
MON TUE WED THU FRI SAT

Encryption: In Cryptography encryption is a process of transferring the information called as plain text using an algorithm to make it unreadable to anyone except those processing or possessing special knowledge by using a key the result of this process is the encrypted information called as cypher text.

Decryption: The reverse process of encryption is known as Decryption.

Symmetric Encryption

→ It uses a single key to encrypt and decrypt the message

Asymmetric Encryption

→ It is also known as Public key encryption which uses two different keys that is a public key to encrypt the message & a private key to decrypt the message.