Oracle

9i — 'i' nteractive

10g — 'g' rid

11g — 'g' rid

12c = 'c' loud computing

GATE

DE, C, DBMS, COD, OS, Computer Networks,

Automata, Aptitude, Maths.

14/6/2016

INT 306

Data Base Management System

DDL - Data definition Language

* DDL Commands:
  - ⇒ * create
  - ⇒ Alter
  - ⇒ desc/describe
  - ⇒ rename
  - ⇒ drop
  - ⇒ truncate

* TO create table:-

```
create table tablename
(
  colname1  datatype(size),
  colname2  datatype(size),
  .
  .
  .
);
```

Data types:
- ⇒ char - only alphabets
- ⇒ varchar - @, K1624 ... etc
- ⇒ int (: NOTE: no mention of size)
- ⇒ float
- ⇒ Number ⇒ It takes both int and float

∴ for float Number(p, s)

p - total length

s - no. of digits after decimal

Ex:- Number(3,2) = "7.63"

* TO view the created table:-

```
desc tablename;
(or)
describe tablename;
```

* To alter the table in following ways:-

1) TO change datatype of a column:-

After table tablename
modify colname Newdatatype(size);

2) TO add new column:-

After table tablename
add colname datatype(size);

3) TO drop a column:-

After table tablename
drop column columname;

4) TO rename a columnname:-

Rename oldname column name
oldcolumname to new colname;

* TO change the name of the table:-

* Rename old tablename to
new tablename;

TO drop the table:-

* Drop table tablename;

TO delete the structure & data of the table:-

* truncate table tablename

16/8/2017 INT 306

DML:- Data manipulation language.

DML commands:-

- Insert
- select
- update

* TO insert values into the table:-

insert into tablename values
(value1, value2, value3, - - -);
⇓
as per the colname and its datatype

* TO insert the values which we want:-

insert into tablename (colname1, colname2)
values (value1, value2);
⇓
as per the colname written before
values.

Select:-

* TO view the inserted values of a table:-

select * * from tablename;

* TO view only particular column we want:-

select colnname from tablename;

* To view all details of particular rows:-

select * from tablename where condition;

condition = any values of a column of that particular row.

update:-

* To update the values of a row:-

update tablename set colname = value where condition...;

16/8/2017 INT-306

* 3 types of levels in Data abstraction :- (hiding)

→ physical level
→ logical level
→ view level

There are different views for different users

Data dictionary is called "Metadata".

Meta ⇒ data about data

Architecture of DBMS:-

External level (view level)
Conceptual level (logical level)
Internal level (physical level)

Data Independence - Achievement of layered Architecture of DBMS:-

Two kinds — logical data independence
— physical data independence

21/8/2016 INT-306

SQL - Constraints:-

Constraints:- restrictions

- NOT NULL
- primary Key → V.IMP keys (unique + NOT NULL)
- Foreign Key
- unique → also accepts not null
- default
- check

Features of primary Key:-

* primary key is column (or set of columns that uniquely identifies
* primary key is cannot be have duplicate and Null values
* There is only one primary Key in a table
* But One can combine upto 16 columns to make Composite primary Key
* primary key should have min no. of attributes

To add primary Key Constraint:-

Table level:-

```
Create table tablename
    (
     colname1   datatype(size),
     colname2   datatype(size),
       !
       !
     Constraint primary key(colname1,colname2)
    );
```

Column level:-

```
Create table tablename
    (colname1 datatype(size) primary key,
       !
       !
    );
```

* To Know what constraints are added:-

```
Select * from user_constraints where
    table_name = "tablename";
```

To name the existed Constraint:-

```
create table tablename
    (
     colname datatype(size),
     constraint any_name primary key
          (primary key (colname)
    );
```

* To remove the primary key:-

```
After table tablename drop primary key;
    (or)
After table tablename drop constraint
                Constraint_name;
```

To add NOT NULL Constraint:-

It is only columnlevel constraint.

```
Create table table_name
    (Key colname1 datatype(size) NOT NULL,
     colname2 datatype(size),
       .
       .
    );
    (or)
After table table_name
    modify colname datatype(size) NOT NULL;
```

23/8/2017 INT-306

Foreign Key:

Foreign key is a column or set of columns that refers to primary key of same table or different table

* Foreign key can have duplicate values and Null values

① you cannot delete any value from primary key table if the same value exist in the forcign key table

② you cannot update any value from parent table if the same value exist in child table also.

3) you can insert only those in child table which exist in parent table.

They are referential Integrity rules.

Syntax of foreign key:-

Column level:-

```
Create table tablename
(
colname1 datatype references parent table_name
(colname of primary key),
colname2 datatype(size) );
```

Table level

```
Create table tablename
(
colname1 datatype(size),
colname2 datatype(size),
;
foreignkey (primary key colname) child table
references tablename(primary key colname)
);
```

24/8/2017 INT-306

Foreign Key:-

using constraints method:-

```
Create table tablename
(
colname datatype(size),
;
colname datatype(size),
Constraint userconstraintname foreignkey
(child table colname) references
parent table_name (primary key colname)
);
```

Using alter:-

```
Alter table tablename
    add foreign key (colname)
references parenttable name(primary key colname);
        (or)
Alter table tablename add constraint
userconstraint name foreign key (colname)
references parent table_name(primary key (colname));
```

Unique:-

Column level:-

```
Create table tablename
(
  colname1 datatype(size) unique,
    :
  columnname 2 . . datatype(size)
);
```

table level:-

```
Create table tablename
( colname1 datatype(size),
  colname n datatype(size),
unique(colname1, . . . . n)
);
```

Using Constraint:-

```
Create table tablename
( colname datatype(size),
    :
  colname datatype(size),
Constraint userconstraint name unique(colname)
);
```

By alter keyword:-

```
Alter table tablename
    add unique(colname);
        (or)
Alter table tablename
    add constraint user constraint name
        unique (colname);
```

Check:-

Column level

```
Create table tablename
( colname datatype(size) check(condition),
    :
  colname datatype(size)
);
```

Table level

```
Create table tablename
  (
   Columnname datatype(size),
      :
   Colname datatype(size),
    check (condition)
  );
```

Alter method:-

```
alter table tablename
    add check(condition);
```

(or)

```
alter table tablename add
  constraint userconstraint_name
  check(condition);
```

Default:-

Default & NOT NULL are Coulumn level Constraints only

```
Create table tablename
  ( colname datatype(size) default value=,
  );
```

using alter:-

```
After table tablename
modify colname datatype(size)
  default 'value';
```

* If we want to delete from parent as well as child table
  - on delete cascade

* on delete set NULL

30/8/2017 INT-306

SQL operators:-

① Arithmetic operators:-

+, -, *, /, ( )

```
Select colname * 12 from tablename;
Select colname1, colname2+colname3 as "user view" from tablename;
Select colname1, colname2*12. as "user view"
  from tablename;
```

② logical operators:

AND, OR, NOT

IN & NOT IN operators:

Select name from tablename
where dept in ('IT', 'CSE', 'MEC')

Between

Select colname from tablename
where colname between low limit
and upperlimit

1/9/2017

INT-306

Select name from tablename
where columnname between

~~Like~~: Like:

It is used for pattern matching

% → zero match or n no. of characters

_ → single character

Syntax:

Select e_name from tablename
where ~~like~~ e_name like 'a%';

To fetch if the req'd letter is at last

Select e_name from tablename
where e_name like '%a';

To fetch data if req'd letter is in anywhere:

Select e_name from table name
where ename like '%a%';

① To fetch the data while required is at second position of a string :-

select ename from employee where ename like '_aman';

Same as ① but whatever be the string after 'a' "required char" :-

Select ename from employee where ename like '_a%';

Concatenation operator :-

To add the both columns :-

Select colname1 || Colname2 from tablename;

To show both the colums to different :-

Select colname1 || Colname2 as NewColname from tablename;

We can add a string to both columns.

Syntax:

Select F_name || 'lives in' || Address as information from tablename;

Quotation operator :- (q operator)

To remove the confusion in compiler about quotes.

Select colname1 || q'[user's information]' || colname2 as newcolname from tablename;

order by

select * from tablename order by colname/position of colname Asc/DESC;

By default it works as ascending order

distinct Keyword:

Select distinct colname from tablename;

Q1)

EID	Ename	Salary
101	Anuj	10,000
102	yaman	40000
103	Raman	8000
104	Raman	35000

4/9/2017

INT - 306

Creating table from another table:

Create table newtablename
as select * from old tablename;

If we want to copy only two columns to new table:

Create table newtablename as select colname1, colname2 from oldtablename;

If we want to copy from another table on the basis of condition:-

create table newtablename as
extra colname1, colname2 from oldtablename
where condition;

If we want to copy the structure of another table:

Create table Newtablename as
Select * from oldtablename where 1=2;

If we want to insert the values from 1 table to another table:

Insert into New tablename select
* from old tablename;

* TCL-commands:

TCL - Transaction Control language

	Syntax
- Commit	⇒ Commit;
- Rollback	⇒ Roll back;
- Save point	

Save point a;
Insert into tablename values(- - - - -);
Save point b;
~~Insert~~ int
delete from T2 where condition;
Save point c;

If we want to ~~restore~~ the data that we deleted;

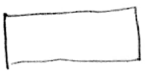
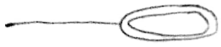
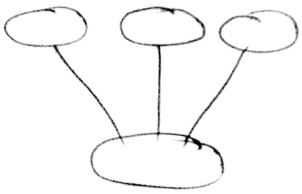
~~Roll ba~~ Rollback to b; → savepoint that we created

6/9/2017 INT-306 most imp to learn

ER-model:-

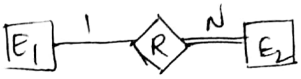
E R - Entities Relationship.

ER-model: Symbols and notations

Symbol	meaning
	Entity type
	weak entity type
	Relationship type
	Identifying relationship type
	Attribute
	Key attribute
	Multivalued attribute
	Composite attribute

derived attribute
age is derived from DOB

total participation of E_2 in R



cardinality relation
1:N for E1 : E2 in R

Key:-

Super Key ⟹ set of all keys. (unique & type)

Candidate Key → unique
→ irreducible

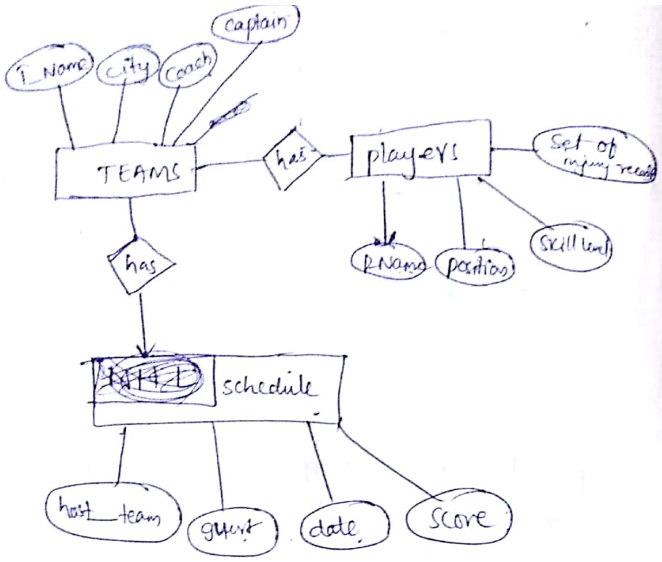
primary Key ⇒ The column with min of attributes.

alternate Key → col's which cannot be make a primary key, they are known to be alternate key

foreign Key

 ⇒ discriminator

8/9/2017 INT-306



We have to create two table when we have multivalued attributes in ER diagrams..

1: one table is having multivalued attribute

2. other table consists of rest of attributes

Derived attributes:-

(1) Conversion of many to many relationships into table from ER diagrams :-

- For many to many we have to make two tables of given entities.
- And also make one more table which consists of primary key attributes of first two entities / tables by using foreign key

(2) Conversion of one to many relationship into table from ER diagrams:-

Copy the primary key column of at "One" table to "many" table

Design issues:

① used of entity sets vs attributes

② use of entity sets vs relationship sets

③

SQL functions:

① Group functions/Aggregate function:-

- Sum
- Avg
- MAX
- Min
- Count → count only not null
- Count(*) → Count all values.
- distinct Count → Count different and donot counts the duplicate

To find the sum of column in table:

```
Select Sum(price) as "Total price"
from tablename;
```

② Single row functions

- Numeric
- String
- Conversion
- Date
- Special

i) Numeric (or) Number function:

a) ABS (absolute function):-

Syntax: ABS(n)

NOTE:-

To perform all single row functions a inbuilt table is used.

Syntax:

```
Select * from dual;
```

(a) use of dual for Abs:

Select abs(n), from dual;

(b) power function:

Syntax: Select power(3,2) from dual.

If from table:-

Select power(colname, n) from tablename;

(c) SQRT (function):-

Syntax:- Select SQRT(n) from dual

Table:-

Select SQRT(colname) from tablename;

(d) MOD(m,n):-

Syntax: Select MOD(m,n) from dual;

Table: Select mod(colname, n) from tablename;

(e) Round (m, [n]):- optional

Syntax: Select Round(m,n) from dual;

Ex:- Select Round(15.23456, 1) from dual;

output = 15.2

(f) Truncate (m, [n]):- optional

Syntax: Select trunc(m,n) from dual;

Ex:- select trunc(15.26, 1) from dual;

Ex:- 15.2

(2) character (or) string functions:-

(a) upper:-

Syntax:- select upper('database') from dual;

Table:-

Select upper (colname) from tablename;

(b) lower:-

Syntax: Select lower('Name') from dual;

Table: Select lower (colname) from tablename;

(c) Initcap:-

Syntax: Select Initcap('Name') from dual;

Table:

(d) Length:-

Syntax:- Select length("name") from dual;

Table:-

Select length"

(e) Substring:-

Syntax:-

Select SUBSTR('SECURE', startposition, [length])
from dual;

Ex:- Secure ⟹ cure

Select SUBSTR('secure', 3, 4) from dual.

* If length is not mentioned, by default it will go up to last of the given string

(f) INSTR:-

Syntax

Select INSTR(string1, string2, [start pos], [nth appearance])
from dual;

Ex:- Select INSTR('database', 'a'). from dual;

Out loc = '2'.

Select INSTR('database', 'a', 1, 3) from dual;

output loc = '6'

(g) RTRIM:- TO trim the string from Rhs.
RTRIM('string', [char set])

Syntax:-

Select RTRIM('string', [char set])
from dual

(h) LTRIM:- TO trim the string from Lhs
LTRIM('string', [char set])

Syntax:-

Select LTRIM('string', [char set])
from dual;

(i) TRIM:-

~~TRIM~~ ('string',

TRIM is used as left trim (or) right trim (or) both.

TRIM(leading/trailing/both [trim char from] 'string')

∴ Syntax:-

Select Trim('char' from 'string') from dual;

(j) LPAD :-

LPAD(string, length, [char])

Syntax:-

Select LPAD('Name', length, [char]) from dual;

LPAD('salary', 8, '*'); ***24000

Table:-

Select LPAD('colname', length, [char]) from table name;

(K) RPAD :-

RPAD (string, length, [char]) from dual;

Table:-

Select RPAD ('colname', length, [char]) from tablename;

(L) Replace:-

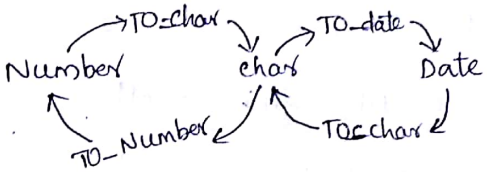
Replace(string, search char, Replace char).

Syntax:-

Select Replace('string', 'search char, replace char) from dual

(3) Conversion functions

Conversion can be done among these 3 datatypes.



∴ Ex:-

Name	Salary
A	1000
B	2000
C	3000

char (Name) char (Salary)

∴ Select to_number (Salary) from tablename;

X : ~~Select~~ sum(salary)

Ex:- If we want to put $ infront of a Salary :-

∴ Select tochar (10,000, $99999) from dual/tablename;

NOTE:- '9' - represents one digit

13/9/2017 FN-306

(4) DATE functions:-

These are applied on only date data type.

a) Sysdate:-

Select sysdate from dual;

Ext 13-Sep-2017

b) ADD_MONTHS (d,n):-

Select ADD_months ('DD-MM-YY", n)
from dual;

Ext

Select ADD_MONTHS ('13-sep-17', 4);
from dual;

Output:- 13-jan-2018

c) MONTHS_BETWEEN (d1,d2):-

It gives no of months between two dates.

* Select MONTHS_BETWEEN (d,d2)
from dual;

d) Last_day(d):-

It gives last date of the given month

* Select last_day(d) from dual;

e) Next_day(d, char):-

It gives next day after the particular date, it gives only day that we provided.

* Select next_day ("dd-mm-yy", 'day-name')
from dual;

Ext Select Next-day('13-sep-2017', 'Thursday');
from dual;

Output 14-sep-2017,

if we write 'wed' output:- 20-sep-2017

(5) General functions:-

- NVL
- NVL2
- NULLIF
- COALESCE

(a) NVL

Select ~~NVL(cola~~

Select NVL(colname, 'Unavailable') from tablename;

(b) NVL2

NULL

NVL2(exp1, exp2, exp3)

NOT NULL

here exp1 – any colname of the table

Select colnames NVL2(expr1, expr2, expr3) as user view name from tablename where condition;

NOTE:- we cannot select all columns by putting "*"

(c) NULL if:-

equal it gives null

NULLif(exp1, expr2)

Select colname1 length(colname1) "expr1" colname2 length(colname2) "expr2" NULLIF(length(colname1), length(colname2)) User view name from tablenames;

(d) COALESCE:-

It gives first not null expression from table of a particular row.

∴ * Select COALESCE(colname1, colname2, colname3) userview name from tablename;

(e) Conditional expressions:-

- CASE
- DECODE } used perform IF-THEN-ELSE condition.

* CASE colnames when comparison_expr1 Then return_expr1 [when comparison_expr2 then return_expr2 when comparison_expr3 then return_expr3 ELSE END] userview name from tablename;

* DECODE (colname1, 'expr1', 'returnexpr1' expr2, return_expr2 expr3, return_expr3

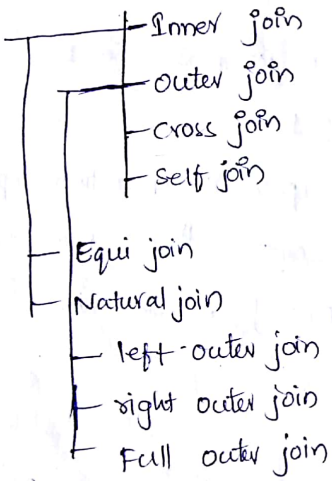
Set operators:

- Union
- Union all
- Intersect
- Minus.

15/9/2017 INT-306

"See tables in SQL Saved as "SQL joins".. All the below examples are written to the tables".

SQL joins:



* In cross join, if table1 having 'm' rows, table 2 having 'n' columns, then after cross join the resultant table will have mxn

To cross join the two tables:

* Select * from tablename1 cross join tablename2;
* select * from tablename1, tablename2;

To view only particular columns:-

* Select tablename1.(desired colname), tablename2.(desired colname) from tablename1, tablename2;

Inner join

Equi join:-
To view only the same particulars details from two tables

- Ex:-
① Select * from student12, info where student12.rno = info.rno;

② Select * from student12 join info on student12.rno = info.rno;

③ Select * from student12 inner join info on student12.rno = info.rno

NOTE: If 'join' then 'on'
No 'join', then 'where'.

* If we want give alloy names to the tables:-

select * student456. a join info456 b

on a.rno = b.rno;

* If we want view desired colums after joining the tables:-

Select name, cgpa, a.rno from

Student a join info456 b on

a.rno = b.rno;

Natural join:-

It is as same as Equi join

Select * from Student456 natural join

info456;

NOTE:-

Duplicate columns are not visible and no need of condition to compare

Outer join:-

left outer join:-

① Select * from Student456 left join info456

on student456.rno = info456.rno;

② Select * from student456 leftouter join info456

on student456.rno = info456.rno;

By using only '9':-

③ Select * from student456, info456

where student456.rno = info456.rno (+);

↓

written right Side of condition to get left outer join

With alloys:-

⇓

view left table

④ Select * from student456 a left join info b

on, a.rno = b.rno;

TO view specific columns:-

⑤ Select student456.rno, cgpa from

Student456. left join info456 on

Student456.rno = info456.rno;

NOTE:- "For right outer join replace 'left' with 'right'.

NOTE:
When written with (+).

Select * from student456, info456 on

student456.rno(+) = info456.rno;

↓

written left side of the condition to view right table

Eg:
on right outer join.

Full outer join:-

All the commands of right and left outer join will work for full outer join except (+) method.

i.e

① Select * from student456 full join info456 on student456.rno = info456.rno;

② select * from student456 full outer join info456 on student456.rno = info456.rno;

③ Select student456.rno, cgpa from student456 full join info456 info456 on student456.rno = info456.rno;

Self join:-

It is used to join the table itself

Emp 1

EID	EName	DOJ	Sup-ID
101	Anuj	12-9-17	Null
102	Tanuj	12-9-17	101
103	Manuj	12-8-17	101
104	Anu	9-9-17	102
105	Anjali	20-2-17	104

Emp 2

EID	EName	DOJ	Sup-ID
101	Anuj		
102	Tanuj		
103	Manuj		
104	Anu		
105	Anjali		

Same

EID, EName → E_ID

Select * from

Select a.ename as "Employe name", b.ename as "Supervisor name" from empl a, empl b where a.supV_ID = b.EID;

Roll → Name

A = {1, 2}

B = {

Questions:-

add Salary, dep to above tables.

insert 10000, 20000, 60000, as salary

insert HR, MRKT, HR, MRKT as dept

18/9/2017 INT-306

EMPLOYEE:-

EID	ENAME	SALARY	DEPARTMENT
101	Anuj	10,000	HR
102	Aman	40,000	MRKT
103	Bharat	20,000	HR
104	Suvathi	50,000	ACC
105	Suman	10,000	MRKT

① TO count no.of employees:-

Select count(EID) from EMPLOYEE;

② TO find max(salary):-

Select max(salary) from EMPLOYEE;

③ TO count no.of employees for particular dept:-

select count(EID) from employee group by department;

④ Conditions of group by:-

Select department, count(EID) from Employee group department;

⑤ TO find max(salary) in each department:-

Select department, max(salary) from EmployEE group by department;

⑥ TO find sum of salaries in each dept:-

Select department, sum(salary) from employee group by department;

⑦ TO display departments which are giving total salary more than 30,000:-

Select department from employee group by department having sum(salary)>30,000;

⑧ TO find second largest salary:-

Select max(salary) from employee where Salary != (select max(salary) from employee);

V.V.Imp:-

Functional dependency & Normalization:-

Functional dependency:-

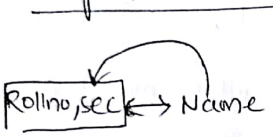
Ex:- $X \rightarrow Y$: read as X functionally determines Y

: Y depends on X

X :→ determinant

Y :→ determined

Fully functional dependency:-



Roll no	sec	Name
1	K1624	Anuj
2	K1650	Bhasti
3	K1624	Anuj
4	K1650	Swasti

The given table is fully functional depended if ~~data~~ Name depends on both of the attributes.

NOTE:- If name depends on any one of the attribute individually then it is not fully functional dependent

partial functional dependency:-

In partial functional dependency determined depends on subset of any attributes of the composite key attributes, but not the entire primary key

Transitive functional dependency:-

~~An~~ A, B, C are attributes., said to have transitive dependency if $A \rightarrow B$ and $B \rightarrow C$ then if $A \rightarrow C$, then it is called transitive functional dependency

Trivial functional & Non-Trivial functional:-

Trivial :-

If $AB \rightarrow B$; $B \subseteq AB$ then it $AB \rightarrow B$ is always true.

⇒ It is called "Trivial" functional dependency".

Non-Trivial:-

If $AB \rightarrow B$, then $B \not\subseteq AB$. But dependency exists.

⇒ It is called "Non-trivial functional dependency".

20/9/2017 INT-306

closure:

Attribute or set of attributes through which we can determine all the attributes

Axioms:

Reflexitivity:

If x is set of attributes and y is subset of x then $x \rightarrow y$ holds

Augmentation Rule:

If $x \rightarrow y$ holds and w is set of attributes, then $wx \rightarrow wy$ holds

Transitivity Rule:

Union rule:

Decomposition rule:

pseudotransitivity rule:

Normalization

First normal form

It defines all the attributes in a relation must have atomic domains.

⇒ Each attribute must contain only a single value from its predefined domain.

∴ P → NP.

Second Normal form

⇒ Every non-prime attribute should be fully functionally dependent on prime key attribute

∴ if partial dependency exists.

$P_{12} \rightarrow NP$

if $P_1 \rightarrow NP$

$P_2 \rightarrow NP$

result:

remove partial dependency

Third normal form

- No non-prime attribute is transitively dependent on prime attribute

- For any non-trivial functional dependency $A \rightarrow A$, then
  - X is a super key
  - A is a prime attribute

∴ NP → NP

∴ NP → P

BCNF

Every attribute should be a candidate key

25/9/2017 INT-306

TEMPORARY TABLES:-

Eg: TO Create temporary table:-

```
Create global temporary table tablename
(
    Colname1  datatype(size),
    Colname2  datatype(size),
      :
    Colnamen  datatype(size).
);
```

TO delete the rows:-

```
On commit delete rows;
```

TO preserve the rows:-

```
On commit preserve rows;
```

External tables:-

These tables are made to import the data from outside file and to export from SQL data to outside file

[∵ outside file = Notepad, file (or) any other file]

In notepad, the attributes are divided by "," and rows are divided by next line

Syntax:-

```
Create or replace directory my_file as 'G:';
Create table tablename
(   Colname1  datatype(size),
    Colname2  datatype(size).
);
ORGANISATION EXTERNAL (-1
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY "my_file"
    ACCESS PARAMETERS (-2
        RECORS DELIMITED BY Newline.
        FIELDS TERMINATED by ','
        MISSING FIELD values are NULL
        (-3
            id char(50), name char(50)
        )-3
    )-2
    location ('filename.txt') )-1
```

REJECT LIMIT UNLIMITED;
Select * from Tablename;

ORACLE_LOADER is used import the values from external file.

DATA-PUMP :- is used for exporting the values ~~from~~ to external file

27/9/2017 INT-306

Few operators :-

All, Any, exists
↓ ↓ ↓
and or In

Syntax:

Select * from Tablename
where Colname (condition) any (expr1, expr2--)

Ex:

Select * from T1 where
Sal > Any (10000, 20000, 30000);

8/10/2017 INT-306

UNIT-4 : VIEWS

VIEW:-

A view is a virtual table it logically represents subsets of data from one or more data

Advantages:-

1) To restrict data access
2) To make data independence
3) To present different views
4) To make complex queries easy

Syntax:-

Create or replace view viewname as select colname1 .... from tablename where condition;

Types of views:-

1) Simple views
2) Complex views

Complex views:-

Complex views are formed by join, having, aggregate functions, distinct keyword, row num column.

Simple view:-

→ Read only:- It cannot be updated

→ updatable view:- updates can be done

To drop a view:-

drop view view_name;

⇒ Complex views have joins, group by, distinct, keyword

⇒ we cannot update views if they viewed in complex views

Data Dictionary:-

⇒ It gives information about base table, no. of view tables.

Imp views:-

1) User_objects:-
It will show the only one user's objects

2) DBA_objects:-
It will show the all user's objects that present in the software

3) All_objects:-
It will show use the objects of a user and objects of other user which are accessed by current user.

How to create user:-

Create user 'user_name' identified by 'password';

Ex:-
Create user 'lovely' identified by 'abc';

⇒ As user was created he donot poses all privileages or rights to perform different actions or accessing objects.

To give privilege to a new user:-

grant "privilege name" to username;

Ex:- grant create session to lovely;

To give all privileges to a new user:-

Syntax:-
grant all privileges to username;

* If we want to view all the objects of a user :-

Select * from dba_objects where owner='username';

9/10/2017 INT-306

VIEWS:-

We cannot perform INSERT, UPDATE, DELETE

If we want to see only the tables of particular user:-

```
Select * from user_tables;
```

If we want to see information of columns of particular user:-

```
Select * from user_tab_columns;
```

To view all the constraints of particular user for all the tables:-

```
Select * from user_constraints;
```

To view on which particular column the constraint has applied:-

```
Select * from user_cons_columns;
```

How to add a comment to a table:-

* Comment on table tablename is 'comments';

To see the comments of a table

* Select * from all_tab_comments;

To add comment on column of a table:-

Comment on column Employee Eid is 'Employee ID';

Syntax:-

Comment on column tablename.columnname is 'comment';

To see the comments on columns:-

```
Select * from user_col_comments where
table_name = 'TABLENAME';
```

CREATING SEQUENCES, SYNONYMS AND INDEXES

Syntax for creating sequence:-

```
create sequence sequence name
[ {increment by | start with} integer
| {maxvalue integer | nomaxvalue}
| {minvalue integer | nominvalue}
| {cycle | NO cycle}
| {cache integer | NOcache}
| {order | NOorder}
];
```

if we dont give max value then it will go upto the default value i.e. 10^{27}

How to use sequence for a table(s)

any table:-

insert into tablename values (sequencename.nextval, valuesofcolname, ...);

Nextval - will insert the next value of the sequence.

Currval :- will insert the present value of the sequence.

we can set sequence default value to column:-

Ex: create table _ Student
(
Rno int default s5.nextval;
Name char(10),
course char(5),
);

11/10/2023

DNT 305

If we want to know the information of the sequences:-

select * from user_sequences;

Synonyms:-

Advantages:-
- we can have one more name
- shorten then length of tablename/objectname.
- hide the identity and location of the object.

Syntax:-

create [public] synonym synonym_name for object/table;

Ex: create synonym ef for emps;

⇒ If we create public synonym, then that synonym can be accessed by other user also.

⇒ If we don't write public, by default it takes as private synonym.