Page No.
Date 16/1/2024

→ Data - Raw facts/figures ; → Info - processed data that is meaningful.
↳ Image, sound, characters ↳ Arranging raw data into meaningful data.

→ Database - collection of related data that is well organised which is related in a meaningful way which can be accessed by different users but stored only once.

S.No	Name	Address	Contact No.
1	A	W	0 ---
2	B	X	1 ---
3	C	Y	2 ---
4	D	Z	3 ---

} Database

Relational Database - Database in tables
Relation - Combination of rows and columns (table)

No. of Operations → (i) Insertion (ii) Deletion (iii) Updation (iv) Search
(v) Sort

Traditional Way of storing :- File Systems
File - combination of application programs ; eg - files made in drive.
★ DATABASE WORKS ON CLIENT-SERVER ARCHITECTURE
→ User that can access resources - ~~Case~~ CLIENT

DISADV OF FILE SYSTEMS →
(i) Data Redundancy - Same data baar baar alag alag files mein likhenge to ~~redundancy~~ size badhega.
(ii) Less scalable
(iii) Too much data needs large space.
(iv) No backup/recovery
(v) Data dependency - ek change karna nai, sabme alag alag krna.
(vi) Concurrency problem - ek time pe multiple users file access karenge to data inconsistency aaegi.

~~Sub Systems~~

(vii) Data Security (viii) Time consuming

~~Sub Systems~~

CHARACTERISTICS OF DATABASES

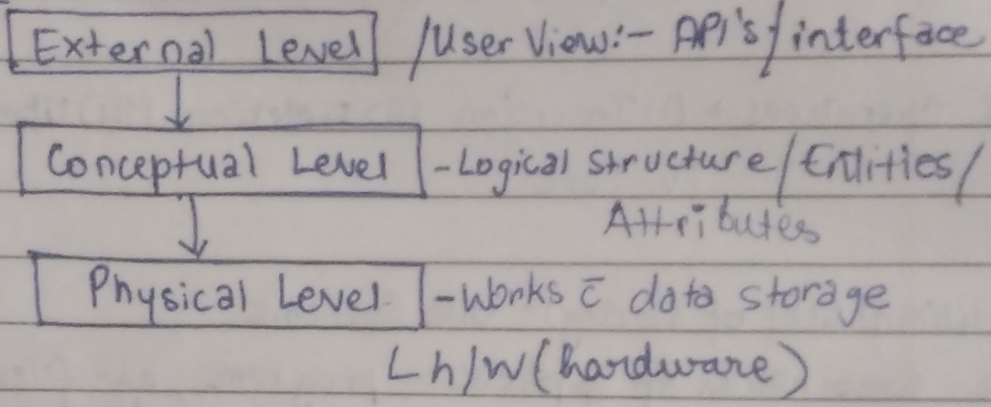
(i) organised data / related (ii) More scalable

(iii) integrity (correct) (iv) Non-Redundancy (v) Consistency.

(vi) Shareable (vii) Data Independancy.

DATABASE ARCHITECTURE ~~AAA~~ V.V.V. Imp

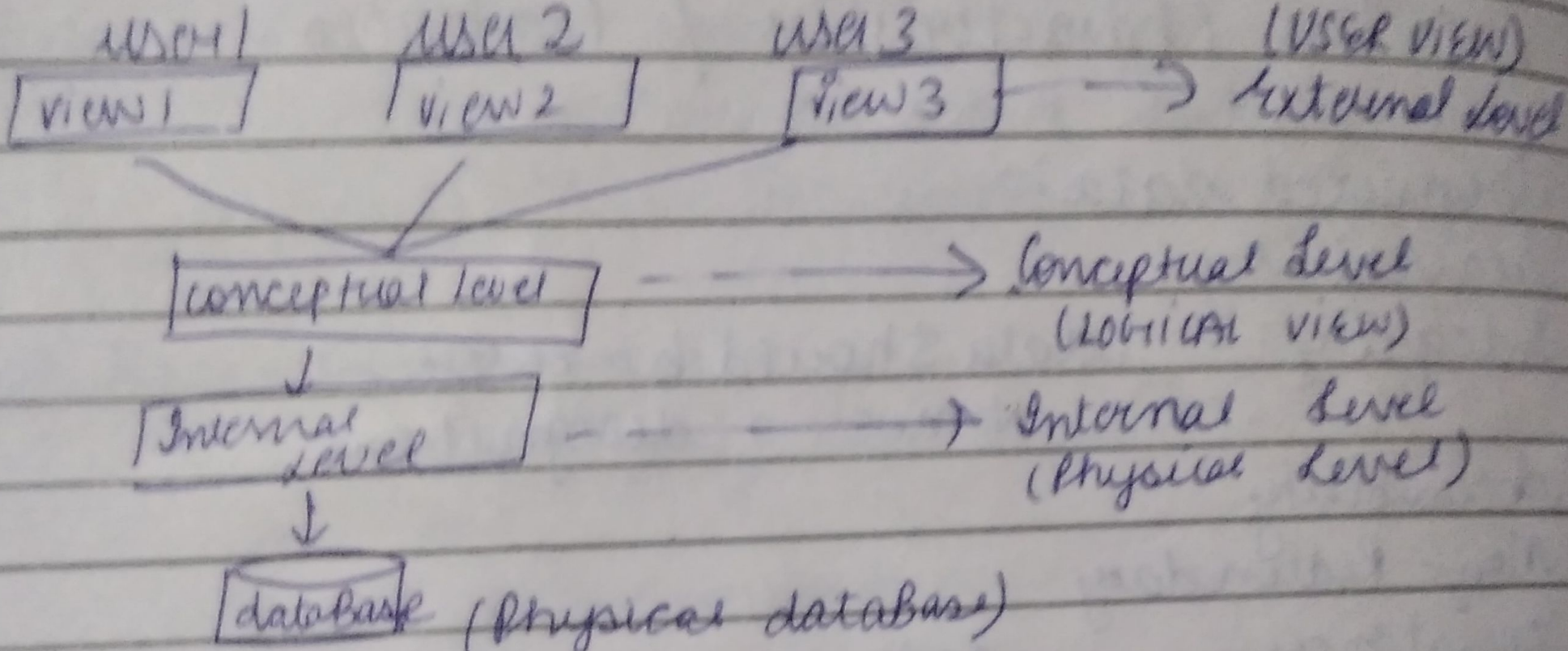
3 level Architecture → ~~fff~~



DATA INDEPENDENCE => means a change of data at one level should not affect another level.

3 tier - architecture of DBMS

3 Levels -



TYPES →

(1) Physical Independence :- Any change in physical location of table and indices should not affect conceptual or external level.

(2) Logical/Conceptual Independence :- Change in conceptual level does not affect external level.

Wednesday

Data Base Schema :- planning

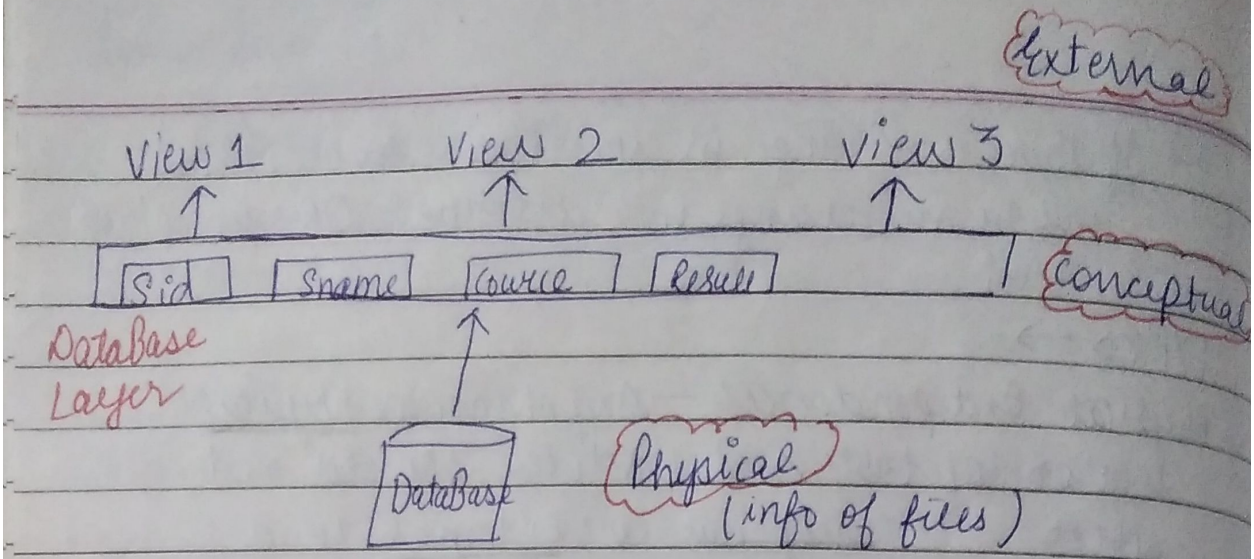
Define :- DataBase schema is the logical representation of DB which shows how the data is stored logically in entire DB.

A schema does not physically contain data itself. instead it gives info about shape of data and how it can be related to other tables/models.

TYPES → 3 types

1) External / User View Schema → It interact b/w extenal & logical dbms.
2) Logical / Conceptual Schema → represents data in the as tables.
3) Internal / Physical Schema | in conceptual schema to users

unorganised data is placed



Define → Main PDF

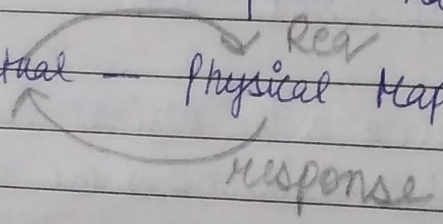
MAPPING :→

tells connectivity b/w all 3 levels.

Define :- Mapping used to transform the request and response between various DB levels of architecture (structure)

TYPES → 2 types

1) External – Conceptual Mapping
2) ~~Conceptual~~ – Physical Mapping



1) External-Conceptual Mapping → It define the correspondance between particular external/user view and conceptual view. The external-conceptual mapping tells dbms which object on conceptual level correspond to the object requested on particular user's external view.

2) Conceptual - Internal / Physical Level :- It define correspondance b/w particular conceptual and physical view) The database is store on physical storage device. It describes how conceptual records are stored and retrieved to and from storage device. This means that conceptual - internal mapping tells dbms how conceptual records are represented.

table

30/01/2024

1. RELATION - All in relational database is stored in the form of relation. Each relation has attributes. A relation represents 2-D table where rows of the table represent individual record & columns represent attributes.

2. ATTRIBUTES - An attribute is a characteristic of relation that helps in interpreting meaning of data values in each row. No 2 attributes of same relation can have identical names.

3. TUPPLES - Rows / Record in a table. No two tupples are identical in their names.

4. DATA VALUES :- The smallest unit of data in a table.

5. CARDINALITY → Total no. of rows / tupples at any one time is known as a cardinality of relation.

6. DEGREE → No. of attributes / columns that comprise a relation.
unary
binary
ternary
n-ary

7. DOMAIN → possible / range of values of each attribute.
values
↳ sometimes called attribute specification.

31/01/2024

Keys

	Roll	Name	Age	Course
Key →	1	ABC	20	C_1
	2	CDE	20	C_1
Key →	3	ABC	20	C_1

An attribute that gives unique identity.

KEY is a set of 1 or more columns whose combined values are unique among all occurances in a given table.

Page No.
Date / 20

TYPES OF KEYS ⇒

(i) SUPER KEY:- It are those attributes of a relation which have the properties of uniqueness.
Let 'K' be a set of attributes of relation R. then K is a superkey for 'R' iff (if and only if) it possesses the property of Uniqueness.
UNIQUENESS:- No legal value of R ever contains 2 distinct tuples with the same value for K.
Eg:- The table drawn behind. Therefore Roll No is the superkey.
☆ Key can be a comb^n of two columns too.

(Roll + Name) ✓ can be a key.
(Name + Age) ✗ Can't be a key cz same values.
(Roll + Age) ✓
(Roll + Age) ✓
(Roll + Name + Age) ✓

All the possible comb^n that can make record unique - SUPER KEY

(ii) CANDIDATE KEYS:- Those attributes of relation that have the property of uniqueness and irr-reducability.
IRR-REDUCABILITY:- Let K be a set of attributes of relation R then K is a candidate key for 'R' iff ——
prop. of uniqueness & irr ——.
↳ No proper subset of 'K' has the uniqueness property. This property means that if a candidate key is composite key (consisting of >1), no individual attribute of candidate key is unique.

☆ A single (non-composite key) is by default a Candidate key.
☆ A table can have more than 1 candidate key.

05/02/2024

Page No.
Date / 20

Relational Integrity Rules

(1.) Entity Integrity Rule ⇒ States that in a base relation, value of attribute of a primary key cannot be null.

(2.) Refrential Integrity Rule ⇒ Refrential Integrity states that if either a foreign key exists in a relation, the foreign key value must match the primary key value of tuples in its own relation or the foreign key values must be null.

(3.) Enterprise I.R ⇒ Additional rules specified by the users/ database administrator of a database.

CREATING A TABLE

```
CREATE TABLE tablename(
    column_name1 datatype [constraint],
    column_name2 datatype);
```

Varchar2 ?

Eg →

Roll	Name	Course

```
CREATE TABLE Student(
    Roll_no  number,
    Name     varchar2(10)
    Course   varchar2(10));
```

DESC table_name [for detailed view of table]

CONSTRAINTS - set of rules applied to tables/attributes to restrict data entry. it helps in maintaining data integrity of table at database level.

Page No.
Date / 20

TYPES OF CONSTRAINTS :-

(i) Column-level ⇒ constraints that apply on a single column.
↳ defined as a part of column definition.

(ii) Table-Level ⇒ specified at the end of 'CREATE TABLE' Statement to constraint multiple columns.

Constraints ⇒

(1.) Not null constraints :-

```
CREATE TABLE tablename(
  column1 datatype,
  name varchar2(10) NOT NULL
);
```

} Column-level

```
CREATE TABLE tablename(
 column1 datatype,
 colum2 datatype,
 NOT NULL(column1, column2)
 );
```

} Table-level

(2.) Unique Constraints :- can contain null value.

```
CREATE ------(
 column1 datatype unique
 ---  ---------);
```

} column-level

```
CREATE-------(
 column---
 column-----
 column----
 unique(---,---,--- ));
```

3.) Primary Key Constraint ⇒ (→ unique + not null)

```
● --------(
----------PRIMARY KEY,
);
```

4) Foreign Key Constraint → A foreign key constraint specifies that the values stored in foreign key column corresponds to value of the primary key in the other table. Also called referential integrity constraint. The table containing FK is known as depending table or refrencing table & the table that is refrenced by foreign ~~table~~ key is base / refrenced table.

For eg:-
```
CREATE TABLE Dept(
Dno int PrimaryKey,
Deptname varchar2(10),
Loc varchar2(10)
);
```
} BASE TABLE

→ P.K

Dno	Deptname	Loc
10	Marketing	D
20	Research	C
30	Sales	M

```
CREATE TABLE Emp(
eno int primarykey,
ename varchar2(10),
sal number(5),
Dno int References dept(Dno)
);
```
} dependent table

← F.K

Eno	E.name	Sal	D.No

← F.K

COMMIT; – To save Table's data.

Page No.
Date / 20

5) Check constraint ⇒ Check constraints specify a cond" which must be True for the data being inserted into the table. A single column can have multiple checks. Checks involving multipk columns are known as table level check constraint.

```
CREATE TABLE Dept(
Dno int Primary Key,
Dname varchar2(10) CHECK(Dname IN(—,—,—)),
Loc varchar2(10)
);
```

6) Default constraint ⇒

```
LOC varchar2(10) default 'Delhi',
```

↳ 'Delhi' in all the ~~columns~~ rows.

DDL COMMANDS

ALTER ⇒ It is used to change the structure of existing table.

ALTER TABLE tablename add address varchar(10);

⇒ ALTER TABLE tablename add column-name datatype;

⇒ ALTER TABLE tablename DROP column-KW column_name;

⇒ ALTER TABLE tablename RENAME COLUMN Column_name To new_name;

⇒ ALTER TABLE tablename ADD ~~CONSTRAINT~~ constraint name(column_name);

⇒ ALTER TABLE tablename MODIFY columnname new data-type;

Page No.
Date / 20

** Don't use " " ! Use ' ' !

DML COMMANDS

INSERT ⇒ INSERT INTO tablename VALUES(---------);
INSERT INTO tablename(column_names) VALUES
(-----);

06/02/2024

A query result can be directly inserted into the table too.
INSERT INTO tablename; → jisme value daalni hai.
SELECT (column_name) FROM tablename WHERE cond^n;
↳ selected query from table

~~DELETE~~

TRUNCATE ⇒

DELETE ⇒ to delete single/multiple data (or rows) from table.
↳ * nhi aata

DELETE FROM tablename WHERE condition;

Ques → Delt (all record) from employee.
DELETE FROM employee.

Ques → Delete the records of clerks.
DELETE FROM employee WHERE job = "CLERK";

Ques → Delete the records of employees who belong to account department. (diff table)
DELETE FROM emp where deptno IN(SELECT deptno from dept WHERE dname = "Account");

TRUNCATE Vs DELETE

Page No.
Date / 20

UPDATE → → Table keyword not to be written.

UPDATE ~~TABLE~~ tablename SET column_name = <new value> WHERE condition;

Ques → Update the salary of emp no 7902 to Rs 3500.

UPDATE ~~TABLE~~ emp SET sale = 3500 WHERE empno = 7902;

Ques → Update empname & salary of emp no 7900 to Sam and 2500 respectively.

UPDATE ~~TABLE~~ emp SET name = 'Sam', empno = 2500 WHERE empno = 7900;

07/02/2024

PL-SQL (Procedural SQL) | SET SERVEROUTPUT ON; → Don't semicolon. | SQL was declarative

↳ Programming concepts + SQL

ADVANTAGE ⇒ branching, loops, functions, error and exception handling.

- multiple statements go as a block hence reducing calling. whereas in SQL, all statements were calling to the Oracle server.
- less data traffic & networking traffic.
- used in web developmental etc.

Declare: (no: ?)

Begin:

Exception:

end;

Modules

Page No.
Date / 20

Declare section → all variables etc; (optional section)
VARIABLE name datatype;

Begin section → executable code (mandatory)
If nothing to execute, write NULL.

Exception section → (optional)
End section → end;

To print → DBMS_OUTPUT.PUT_LINE('whatever u print');
(DBMS_OUTPUT: package containing of procedures/func^n; PUT_LINE: procedure in DBMS_OUTPUT)

Ques To print hello world.
Sol^n
```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello World');
    END;
```

BASIC ELEMENTS OF PL SQL

1. Characterset - strings, numeric data, special characters.
2. Literals - values assigned to constants/variable.
In a=100, 100 is literal, a is variable.

(1) String literals → only strings, (2) Numeric literal, (3) Char literal.
→ A value that is not represented by an identifier. It is simply value used when you initialize constants, variables, etc.
(4) Logical literal - true/false

3. Comments - to improve readibility of code.
Multiline, single line.
Single line → --(double hyphen) → --Here....

Page No.
Date / 20

Multi line → /* */

4. OPERATORS — AND, OR, NOT (no symbol)
(a) Logical operator, (ii) Relational operator,
(c) Arithmetic (d) Expression operator - assignment, string concatenation, range specification.

* In PL-SQL, == is not there, use = for comparison.
→ For assignment → a := 10

→ for range specification → ..
1..n → range 1 to n.

→ Concatenation → ||
First name || Last Name

⇒ Additional operators → IN, BETWEEN etcetc.

5. IDENTIFIERS - names of variables/ constants/constraints.

6. KEYWORDS , 7. DATATYPES

9/02/2024

7. VARIABLES AND CONSTANTS ⇒
variable datatype; → variable
Constantname constant datype := <value>; → constant
* Can't declare multiple in a single line: int a, b; ×
&
* a := &b; (will ask value during execution)

Ques → Write a program to calculate product of 2 nos.
Declare:
VARIABLE a int;
VARIABLE b int;
VARIABLE c int;

* <> - not equal to *

Page No.
Date / 20

12/02/2024 Sub Queries

emp table

EId	Ename	Dept	Salary
1	Ram	HR	10,000
2	Amrit	MKT	20,000
3	Ravi	HR	30,000
4	Nitin	MKT	40,000
5	Ansh	IT	50,000

Eg → Write a SQL query to display maximum salary from emp table.

```
SELECT MAX(salary) FROM emp;
```

Ques → Write a SQL query to display employ name who is taking maximum salary.

```
SELECT ename from emp where salary = (SELECT MAX(salary)
                                      from emp);
```

* will solve inner ~~query~~ query first

Query → Write a sql query to display second highest salary from emp table.

```
SELECT max(salary) FROM emp WHERE salary <> (select max
                                             (salary) FROM emp);
```

Query → Write a SQL query to display employ name who is taking second highest salary.

```
SELECT ename FROM emp WHERE salary = ( SELECT max(salary) from
emp where salary <> (select max(salary) from emp));
```

Query → Display all the dept names along with no. of employees.

```
SELECT DISTINCT(dept), count (*) from emp group by dept;
```

uery → Write a query to display all the departments where no. of employees where no. of employees are <2.

```
SELECT distinct dept, count * FROM emp GROUP BY dept
                    HAVING count (*) < 2;
```

Query → Write a query to display highest salary dept wise and name of the employee who is taking that salary.

```
(SELECT dept, max (salary) from emp GROUP BY dept );
SELECT ename from emp WHERE salary IN
```

★ IN here cz we want multiple values here.
★ (=) for single value.

13th Feb 2024

CONTROL STRUCTURES ⇒

(i) Conditional Controls:- cond^ based programs
- (a) if then
- (b) if then else
- (c) ~~else~~ if then else if

If then ⇒
```
if (cond^n) then
    statement;
endif;
```
—— zaroori hai

If then else ⇒
```
if (cond^n) then
    Statement;
else
    Statement;
endif;
```

Page No.
Date / 20

Query → WAP to check whether no. is even or odd.

```
declare
    n number(2) := &n;
begin
    if (n mod 2 = 0) then
        DBMS_OUTPUT.PUT_LINE (n || 'is even');
    else
        DBMS_OUTPUT.PUT_LINE(n || 'is odd');
    endif; end if;
    end;
```

If then else if ⇒

```
if (cond^n) then
    Statement;
else if (cond^n) then
    Statement 2;
else
    Statement 3;
endif;
```

ITERATIVE STRUCTURES ⇒

(a) Simple loop
(b) while loop
(c) for loop

Simple loop ⇒

```
loop
    Statement;
    exit statement;
end loop;
```

exit is of 2 types ⇒
(i) Unconditional exit
(ii) Exit with cond^ (exit when)

Page No.
Date / 20

Exit ⇒ The exit statement forces a loop to complete unconditionally. When an exit statement encountered, loop completes immediately & the control passes to the next statement after the end loop statement.

Exit when ⇒ Exit when is a cond^ exit statement which causes the control to exit the enclosing loop and shift to next statement after the loop when the specified cond^ is satisfied.

Query - Write AP which executes a loop 3 times using exit statement.

```
declare:
    n number(2) := 0;
begin:
    loop
    n := n+1;
    if n > 3 then
    exit;
    endif;
    dbms_output.put_line("loop executive" || n || "times");
    endloop;
    end;
```

WHILE loop

14/02/24

VARIABLE ATTRIBUTES

%Type → Assigning variable with table's attribute

<variable_name> <tablename>.<column_name> %type;

Salary emp.sal %Type
↳ emp table ke salary attribute

The percentage type attribute is used to declare variable acc to the already declared variable or database column. The datatype and precision of the variable declared using %Type is same as that of the column that is referred from a given table.

%RowType →

<variablename> <tablename> %RowType;
↓ Record ↓ table ka record.

Employee Emp %Rowtype; → Employee.sal:=1000;
Employee.job:='Clerk';

Select Into → Table ke data like variable mein store.

name emp.ename %Type;

SELECT<column_name(s)> INTO <variable_name> FROM <table_name> WHERE condition;

CURSORS => SELECT empno, ename FROM emp WHERE deptno = 20.

↳ Pointer which gives o/p's one by one.

↳ A cursor is a type of pointer built into PL-SQL for querying the database, retrieving a set of records and allowing a developer to access the active data set; a row at a time. This allows the programmers to accomplish tasks that require procedural code to be performed in each record in the result set individually.

→ ~~Cursor~~ Cursor size depends on by oracle's engine built-in memory management & amount of ~~RAM~~ RAM available.

★ SQL - an automatic cursor

★ Active Data Set =>

TYPES

Implicit	Explicit
(Current status of cursor / active data set)	
(i) SQL %. Found (A.d.s present or not) TRUE - data present, else false	
(ii) SQL %. NOT FOUND (Inverse of found)	
(iii) SQL %. ROW COUNT (No. of rows retrieved)	
(iv) SQL %. OPEN (Cursor open hua ke	

ID	Name	Age	Address	Salary
1	Ramesh	22	Agra	2000
2	Hardik	25	Delhi	3000
3	Komal	25	Mumbai	2000
4	Kunal	28	Delhi	4000
5	Ram	27	Agra	5000
6	Rahul	32	MP	1500

Query → Write to ~~update~~ increase the salary of each employee by 500 & use SQL row count attribute to determine the no. of rows effected.

```
Declare: n int;
Begin:
update employee SET salary = salary + 500;
if SQL%FOUND then
n = SQL%rowcount; ---- (n||"rows updated");
else
DMBS output ---- ("No rows updated")
endif;
end;
```

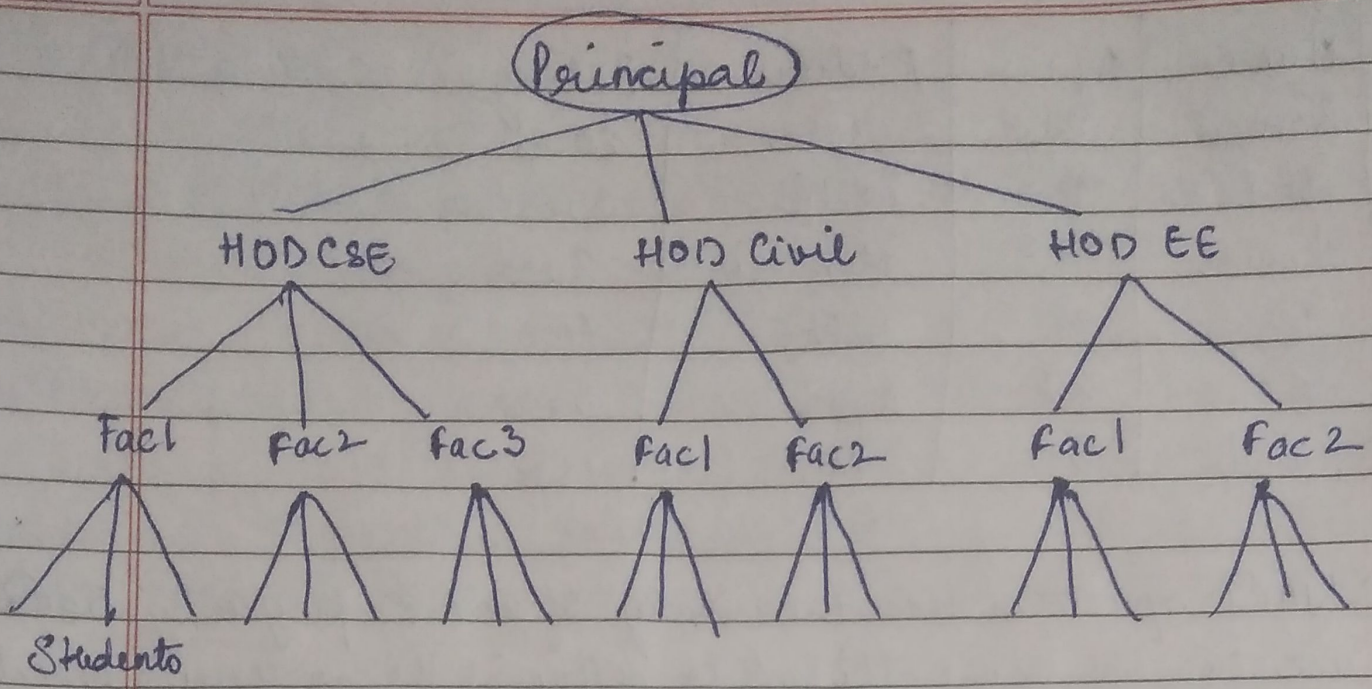
21/02/24 — 6M — Diff — RM

Diff b/w their operations

DATA MODELS → tells how to structurise the data.
↳ how is data organised.

- Hierarchial Model
- N/W Model
- Relational Model

HIERARCHIAL MODELS =>



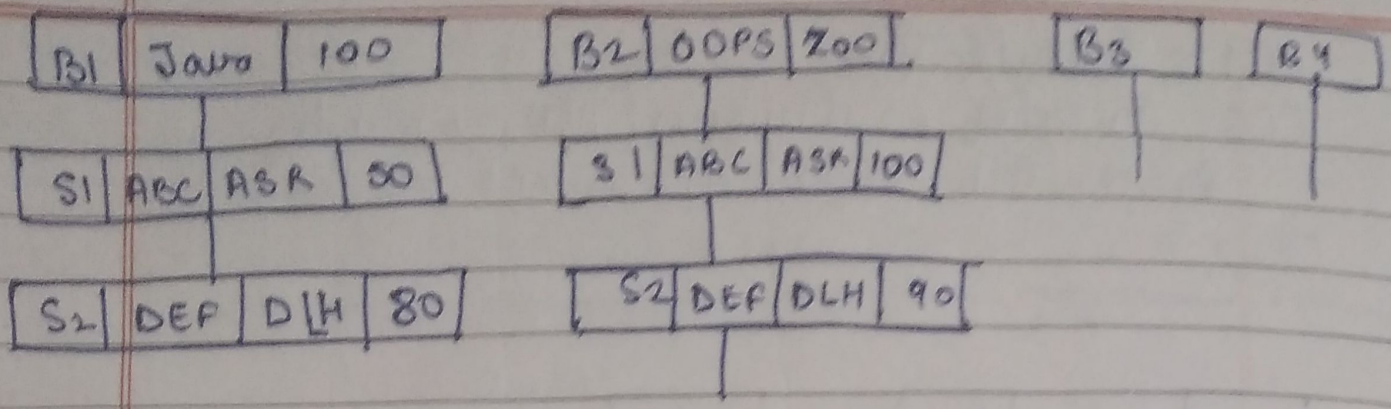
→ Parent child relation → One to many relation

→ Records - parent child

Bid	Bname	Price
B_1	Java	100
B_2	OOPS	200
B_3	C	300
B_4	RDBMS	400

Sid	S_name	City
S_1	ABC	Asr
S_2	DEF	Delhi
S_3	XYZ	Ldh

Sid ~~Bid~~	Bid ~~Sid~~	Qty
S_1	B_1	50
S_1	B_2	100
S_1	B_3	70
S_2	B_1	80
S_2	B_2	90
S_3	B_2	100



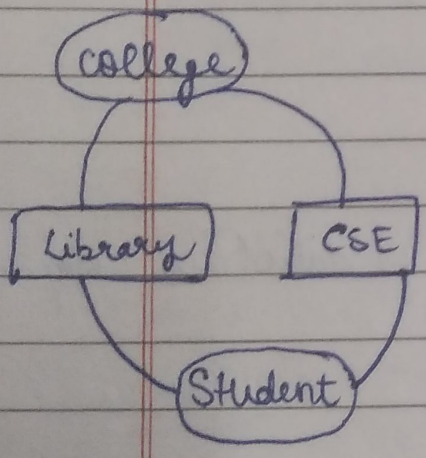
Operations - insert, delete, update, retrieval.

INSERT

UPDATE

24/02/24

NETWORK MODEL ⇒ many to many relation



Eg → Suppliers

S.No	Sname	City
S_1		
S_2		
S_3		

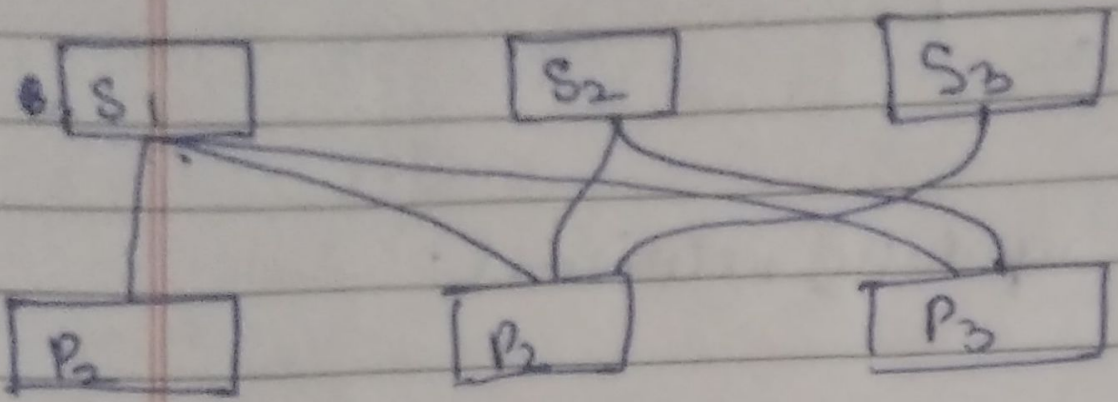
Parts

P_{no}	P_{name}	Weight
P_1	Nut	
P_2	Bolt	
P_3	Screw	

Shipment

Sno	Pno	Qty
S1	P_1	200
S1	P_2	300
S1	P_3	500
S2	P_2	
S2	P_3	
S3	P_2	

CO1 - Architecture
CO2 - Queries, Subqueries. All commands
PLSQL till loops (cursor not included)
CO3 -



Owner - Member Record

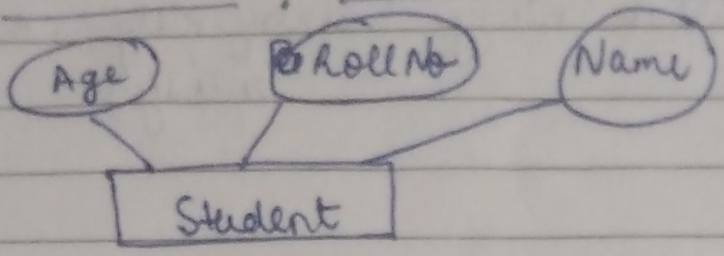
RELATIONAL MODELS → Currently being used.

26/02/2024

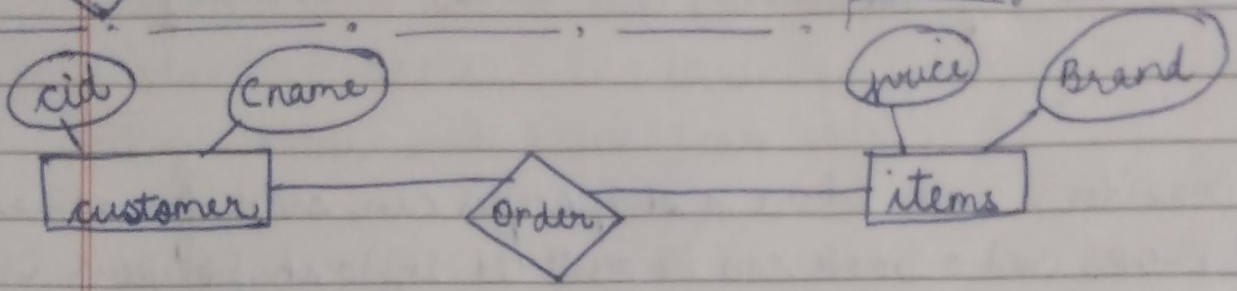
E.R. Diagrams

- rectangular box = entity

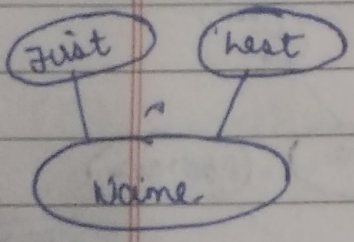
- attributes



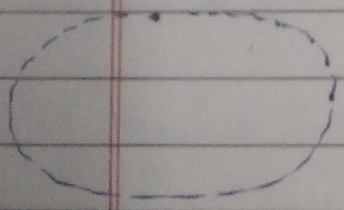
- Relationship denoted by Rhombus.



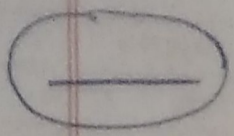
- multi-value attribute (double ellipse)



- composite attribute



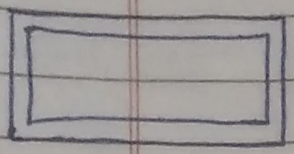
- Derived attribute

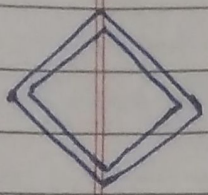
 - Key attribute

Page No.
Date / 20

Strong Entity - Jo uniquely identify krsake rows ko
eg:- primary key.

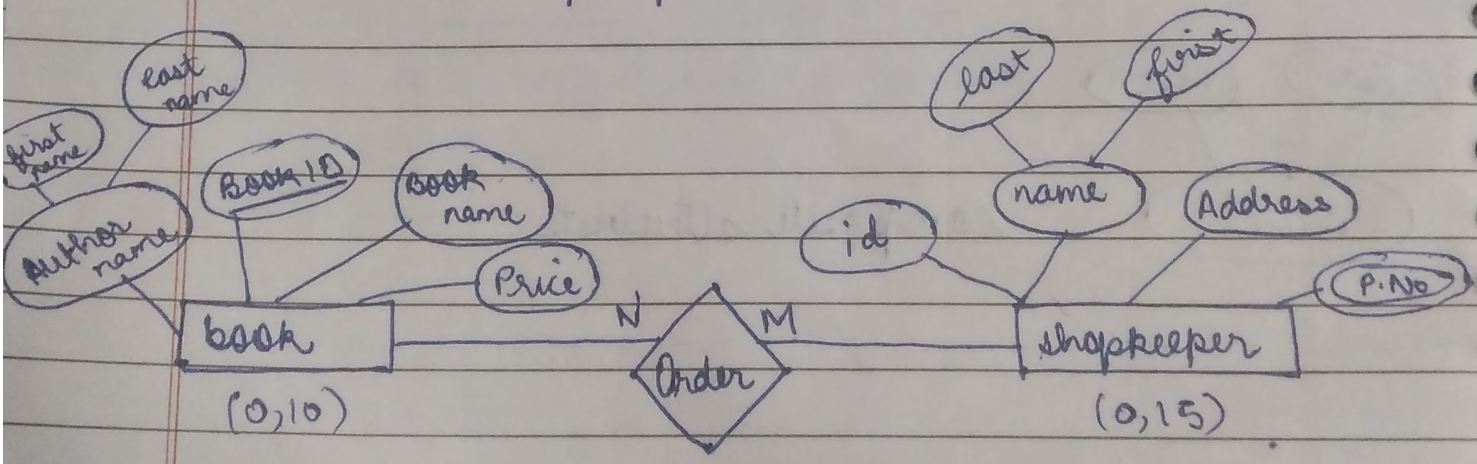
Weak entity - jiske paas apni P.K nhi hai; dusre pe depend
karna padta hai ::

 - Weak entity (Strong entity is the normal entity)

 [association b/w a weak and a strong entity.
any association that includes weak entity.

Ques Consider a case where a shopkeeper can order for atmost 15 books and a book can be ordered by 10 shopkeepers. The two entities types book and shopkeeper are connected by a relationship orders.

Entites - book, shopkeeper



1 Shopkeeper, 10 many books
1 book - many shopkeeper] many to many

Page No.
Date / 20

DATA DICTIONARY - data about a data (metadata)
↳ access of it is with database administrator.

VIEWS - virtual tables (syntax ahead in 08/02/24 lecture)
↳ kitna portion of the whole database do we want to let the user see.

CODD's Twelve Rule :- in any software, 6 of these 12 should be met.

(1) Information Rule - jitna data hai woh tables ki form mein hona chahiye.

(2) Guaranteed Access Rule - ek unique / primary key honi chahiye table mein to differentiate.

(e3) Systematic Treatment of Null Values - says that null values are not zero; it means data there is missing.

(4) Database description rule - ~~data~~ data dictionary should be there in addition to data. ~~that~~
System Catalogue

(5) Comprehensive Data Sub Language Rule - all the diff. language should be able to run the database rules.

(6) Views updating rule - views ko query ke through manipulate kar sake.

(7) Physical Data Independance -

(8) Logical data independance -

(9) High level inserts, delete & update - relational o

(10) Integrity Independency - Correct & consistent data

(11) ~~Dec~~ Distribution Independance -

(12) Non-Subversion Rule -

27/02/24

★ IMP ★

RELATIONAL ALGEBRA

- Procedural language used to manipulate data
- Theoretical Model

OPERATIONS ⇒ — Basic
— Derived

Basic → UNION, Intersection, Difference, Cross Join / Cartesian Product

~~UNION~~ →

ID	Name
501	Amit
503	Anand
5084	Amrit
507	Ram
510	Kapil
512	Ranjit

ID	Name
503	Anand
504	Amrit
506	Madhav
510	Kapil

Page No.
Date / 20

UNION → Concatenate both & remove duplicate (P ∪ Q)
- attributes of both the table should be same.
- Commutative & Associative

P ∪ Q = Q ∪ P P ∪ (Q ∪ R) = (P ∪ Q) ∪ R

Ans in Table form

INTERSECTION → Retrieves common tupples (P ∩ Q)
- commutative & associative

DIFFERENCE → Also known as SET DIFFERENCE
- Q - P
- Resultant is the tupple of the left table that are not in right table
- 501, 507, 512.
- Not commutative
- To find ~~union~~ intersection using ~~intersection~~ difference → Q = (Q - P)

CARTESIAN PRODUCT → P × Q

503	Anand	501	Amit
503	Anand	503	Anand

Degree of Relationship → No. of attributes in a table relationship
- In cartesian product, resultant table degree is addition of individual degrees of table.
- ~~tupple~~ tupple in p * ~~tupples~~ tupples in q = Total tupples

Page No.
Date / 20

Derived / Special →

PROJECTION (π) - retrieves data from relationship.
- $\pi_{(\text{what to retrieve})}(\text{Tablename})$
- $\pi_{(Z4)}(Q)$ - Project Id's from Q.
- Applies on columns

SELECTION (σ) - to set cond^n while projecting.
- $\sigma_{<\text{cond}^n>}(\text{Tablename})$
- When σ and π together, σ will come first, then π
- It applies on tables rows / tupple

Ques Retrieve the name of the student whose roll no is 507 from Q.

$\pi_{\text{name}}((\sigma_{<\text{Roll}=507>} Q))$

28/02/24

DIVISION OPERATOR

enrolled

SID	CID
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

Course

Cid
C_1
C_2

Ques → Retrieve Sid of students who enrolled in every course.

EVERY / ALL → DIVISION OPERATOR

Page No.
Date / 20

Step1 → ~~Cartesian~~ Cartesian product (every × every)

$\pi_{sid}(enrolled) \times \pi_{cid}(course)$

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_2
S_1	C_1
S_1	C_2
S_3	C_1
S_3	C_2

Step 2 → Minus karo

S_1 C_1		S_1 C_1
S_1 C_2		S_2 C_1
S_2 C_1		S_1 C_2
S_2 C_2	−	S_3 C_2
S_1 C_1		
S_1 C_2		
S_3 C_1		
S_3 C_2		

$(\pi_{sid}(enrolled) \times \pi_{cid}(course)) - \pi_{sid,cid}(enrolled)$

S_2 C_2
S_3 C_1

→ resultant will be the student who has enrolled in only one course.

$\pi_{sid}(enrolled) - \pi_{sid}[(\pi_{sid}(enrolled) \times \pi_{cid}(course)) - \pi_{sid,cid}(enrolled)]$

VIEWS

student1

SID	Name	Address
1	Hari	Kolkata
2	Priya	Hyd
3	Divya	Chennai
4	Khushi	Mumbai
5	Ram	Bangalore

CREATE VIEW tablename_view AS
SELECT columnname FROM tablename WHERE cond^{n} ;

CREATE VIEW student_view AS SELECT Name, Address FROM Student WHERE Sid<5;

To see → SELECT * FROM ~~tabla~~ student_view.

Suppose, we have another table

Sid	Name	Marks	Age
1		96	
2		90	
3		94	
4		92	
5		95	

Now, we want marks along ~~c~~ Student1 table

CREATE VIEW student_view AS
SELECT student1.name, Student1.address, Result.marks,
FROM Student, result WHERE student.SID = result.SID

(LAB)

Page No.
Date / 20

29/02/24

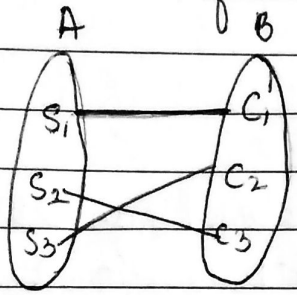
CARDINALITY AND CONNECTIVITY

~~Table 1 ki bitni entr~~

Cardinality of Relationship quantifies the relationship btw entities by measuring how many instances of one entity type are related to a single instance of another relation.

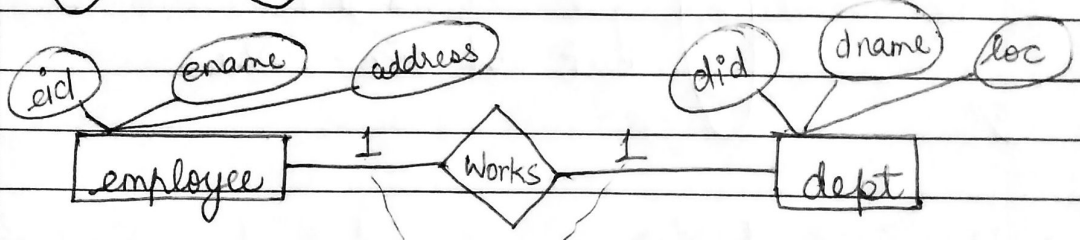
Connectivity is used to describe the classification of relationships which can be classified by one-one, one-many, many-many, many-one. ★ One table, one kind of connectivity.

ONE-ONE - Single instance of relation 1 is associated with exactly one instance of another relation 2.



one elt of A will only be connected to another element of B.

Eg



one to one.

Make table from this → employee

Eid	Ename	Address
E_1	A	Asr
E_2	B	Dli
E_3	C	Mum
E_4	A	Kol
E_5	B	Asr

Dept

Did	Dname	Location
D_1	IT	Bang
D_2	Mkt	DLH
D_3	Prod	Hyd

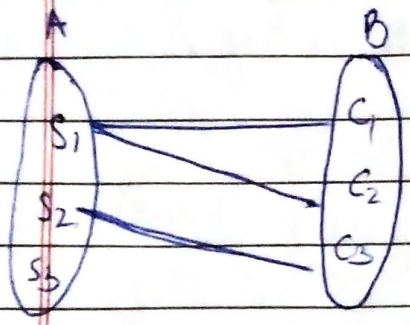
Page No.
Date / 20

Here the relationship is Works. The attributes of that table would be Primary key of both tables.

Works

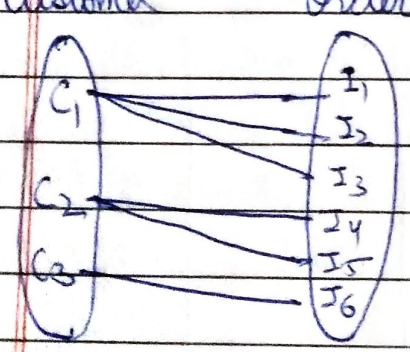
Eid	Did
E_1	D_1
E_2	D_3
E_4	D_2

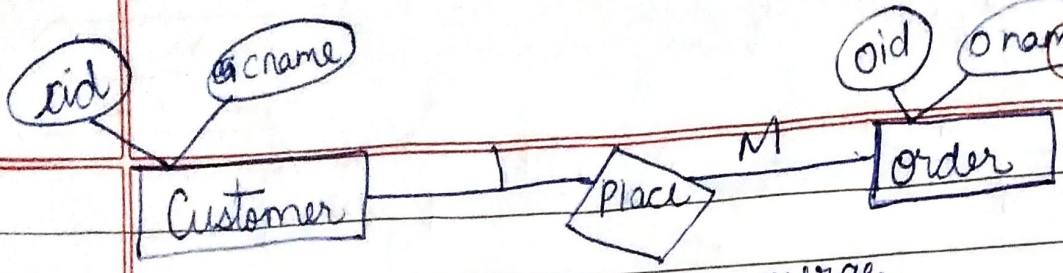
ONE - MANY



One instance of relation 1 can have multiple relations with relation 2. But not vice-versa.

Eg → 1 customer 4 orders
1 order of 1 customer only. (same order 2 customers ka nhi ho sakta.)





merge

Customer

Cid	Customer
C_1	A
C_2	B
C_3	C
C_4	A

Place

Cid	Oid
C_1	O_1
C_1	O_2
C_2	O_3
C_2	O_4

+

Order

Oid	Oname
O_1	X
O_2	Y
O_3	Z
O_4	P

↑ Primary Key

- PK would be many side of the table.

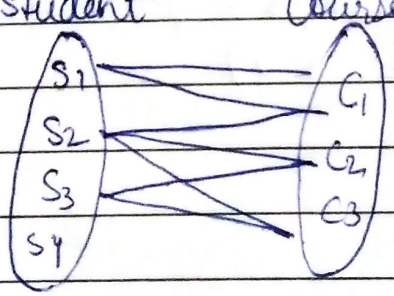
Order + Place

Oid	Oname	Cid
O_1	X	C_1
O_2	Y	C_1
O_3	Z	C_2
O_4	P	C_2

Table reduced.

MANY-ONE — Same as one-many.

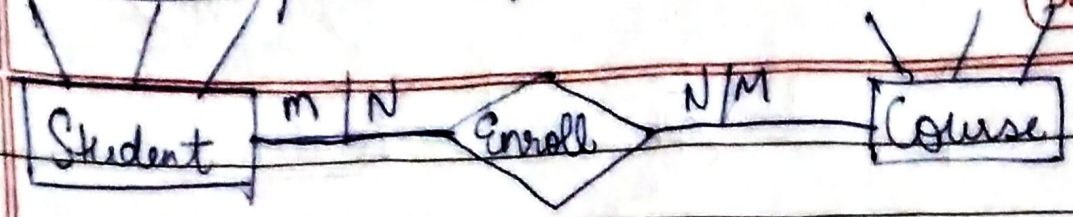
MANY-MANY



One student many course
One course many students

M:M - Same no of rows
M:N - Diff rows / N:M - Diff rows

Page No.
Date / 20



Student

Sid	Sname	Age
S_1	A	
S_2	B	
S_3	B	
S_4	C	

Enroll

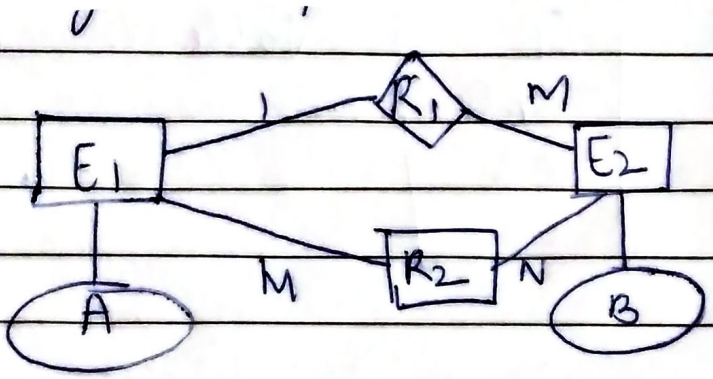
Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_3

Course

Cid	CName
C_1	XYZ
C_2	ABC
C_3	ADE
C_4	XXX

Composite Primary Key (Sid + Cid)

Ques



Q→ What is minimum no of tables required to represent this ER model into relational model. (3)

LAB

Oracle Installation:- ↗ Tool

Language - SQL ★ SQL*Plus - Interface where we will write in SQL.

Oracle → SQL*Plus → SQL

① Username - SYSTEM
P/W - CAREER
Lab

Semicolon

② SHOW USER

③ To create user - ALTER SESSION SET "_ORACLE_SCRIPT"=TRUE;
CREATE USER AKSHITA IDENTIFIED BY 4002;

★ SQL commands are universal, SQL PLUS is only for oracle.
★ User likhne ke baad not seedha connect cz system has not granted it. So,

④ CONN SYSTEM/CAREER
⑤ GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO AKSHITA;
⑥ CONN AKSHITA/4002.

COMMAND TO SHOW USERS - SELECT USERNAME FROM ALL_USERS;

SCOTT CONN SCOTT/MANAGER

SELECT * FROM EMP;
> SET LINESIZE 120; ↗ Changes in all tables.
> SET PAGESIZE 90;
> SELECT * FROM EMP;
> SHOW USER
> GRANT SELECT ON EMP TO PUBLIC;
> " " " DEPT TO PUBLIC;
> SELECT * FROM TAB; → To check if there's a table

Page No.
Date / 20

To copy from SCOTT.

CREATE TABLE EMP AS SELECT * FROM SCOTT.EMP;
CREATE TABLE DEPT AS SELECT * FROM SCOTT.DEPT;

Description command
> DESC EMP

Database lang divide in 4 parts →
1. DDL - Data Definition Language - Structure creation of database.
2. DML - Data Manipulation Language - Insertion
3. DCL - Data Control Language - Permission
4. TCL - Transaction Control Language -

DDL (Table) commands :- (i) CREATE (ii) ALTER (iii) DROP
(iv) RENAME

DML (Data) commands :- (i) INSERT (ii) DELETE (iii) UPDATE
(iv) SELECT

DCL commands :- (i) GRANT - give (ii) REVOKE - take back permission

TCL commands :- (i) COMMIT - permenently save the transac
~~(ii) Rot~~
(ii) SAVEPOINT - check point
(iii) ROLLBACK - undo till the last savepoint
(iv) SET Transaction -

DATATYPES →

(1) CHAR(n) Storing characters upto 10 (no. of characters) char can be stored.

(ii) VARCHAR2(n) – 2000 char.
(iii) NUMBER(m) ; ~~NUM~~ NUMBER(p,s) –
p → Precision, s → scale (after decimal)

NUMBER (5,2) – 234.16
max 5 – 24.12

(iv) DATE – DD-MM-YY
(v) LONG – numerical; Range > Number (2 gbytes)
(vi) LONG RAW – Graphics & Sound files (2 gb)
(vii) BOOLEAN – True/False/NULL
(viii) LOB – Large Object (large amount of data)
(a) CLOB – Character large object (to store books)
(b) BLOB – Binary large object (to store photos)
(c) BFILE – Binary files (movies etc) (4GB)

25/1/24

DML – Data Manipulation Language.

- ~~Insert~~
- SELECT – To retrieve data
- To retrieve data ~~c~~ a cond^n.

SELECT ColumnName FROM tablename WHERE condition;
SELECT * FROM tablename WHERE condition;

Ques → List the employees whose name starts with 'S'.
SELECT ENAME FROM EMP WHERE ename LIKE 'S%';
LIKE → pattern matching from column.

SELECT ename FROM emp WHERE ename LIKE '%S';
(end).

Ques → Name has 5 character name.
SELECT ename FROM EMP WHERE ename
LIKE '_____';

Ques → 2nd character 'D'.
WHERE ename LIKE '_D%'.

Page No.
Date / 20

→ 2 a in name
WHERE ename '%A%A'; : '%A%A%'
NO PARANTHESES ~~first~~ A, last A kahi bhi

★ BETWEEN – to check a value in a range.
While using the between operator, it must be remembered that both the values will be included in the range.

Ques → List the empno, empname, salary, ~~where~~ - salaries lie btw 1500-2000.
SELECT empno, empname, sal FROM emp WHERE sal BETWEEN 1500 and 2000;

★ IN – to check value within SET.

Ques → List the name of emp who belong to dept either 10 or 20.

SELECT ename FROM emp WHERE DEPTNO IN (10, 20);

Diff btw BTW & IN => BTW - Beech ki values bhi
IN - Jo di hai sirf wohi

Ques → List empno, empname of the employees who don't have name FORD, JAMES or JONES.
↳ WHERE ename NOT IN ('FORD', 'JAMES', 'JONES');

Ques → List name of emp, job & commission of those emp who do the job of either clerk or salesman and get no comission
↳ WHERE job IN('CLERK', 'SALESMAN') AND COMM IS NULL;

Page No.
Date / 20

1st Feb, 2024

SORTING → ORDER BY

SELECT column_name FROM table_name ORDER BY column_name DESC;

* by default it is ascending.

Ques Print the employee name & their date in desc order of hier date.

SELECT ename, hiredate FROM emp ORDER BY hiredate DESC;

Ques Retrieve all the rows from emp table for dept 30 and order by name.

SELECT * FROM emp WHERE dept=30 ORDER BY ename;

→ when doing order by null values.

* Jinme null value hongi, woh last mein aega. *

ASC or DESC ; last mein hi aega.

0 hoga to first.

NULL Vs 0

Single Row Functions :-

GROUP / AGGREGATE FUNCTIONS

Aggregate - multiple data leke single value O/P.

eg:- total of all the salaries.

(i) COUNT (ii) SUM (iii) MAX (iv) MIN (v) AVG

(i) COUNT ⇒ It determines the no of rows or non-NULL column values

Select count(column_name/*) FROM table_name;

Page No.
Date / 20

Ques list no of employees working with company.
SELECT count(*) FROM emp;

Ques list total salaries payable to employees.
SELECT

Ques list no of jobs available in the emp table.
SELECT count(DISTINCT job) from emp;
↓
no duplicacy.

2) SUM ⇒ it returns the sum of values for the selected list of column.
SELECT SUM(column_name) FROM table_name;

Ques list total salaries payable to employees.
SELECT SUM(SAL) FROM emp;

3) MAX → SELECT MAX(column_name) FROM table_name.

Ques list max salary of employees working as salesman.
SELECT MAX(SAL) FROM emp WHERE job='Salesman';

4) MIN ⇒ SELECT MIN(column_name) FROM table_name;

5) AVG ⇒ SELECT AVG(column_name) FROM table_name;

Ques List average salary & no of employees working in dept 20.
SELECT Avg(Sal), COUNT(*) FROM emp WHERE deptno=20;

WHERE V/S HAVING

Page No.
Date / 20

GROUP BY CLAUSE

SELECT col_name, group_function(col_name) FROM table_name WHERE condition GROUP BY group_by_expression ORDER BY ~~colob~~ column_name.

Ques List dept no & total salary of each dept.
SELECT deptno, SUM(sal) FROM emp GROUP BY deptno;

Ques List the avg salary of each job from emp table.
SELECT job, AVG(sal) FROM emp GROUP BY job;

Ques List the max salary of each dept.
SELECT deptno, MAX(sal) FROM emp GROUP BY deptno;

HAVING CLAUSE - condition in groupby.

Same like groupby;
Just add HAVING cond^ before ORDER BY.

Ques Find max sal of each dept but show only the depts having salary more than 2900.
SELECT deptno, max(sal) FROM emp GROUP BY deptno HAVING max(Sal~~ary~~)>2900;

Ques List total salary, max and min salary & avg salary of employees jobwise for dept no 20 & display only those rows having average salary greater than 1000 ag.
~~SELECT sum(~~ SELECT job, SUM(sal), MAX(sal), MIN(sal), AVG(sal) FROM emp WHERE deptno=20 GROUP BY job HAVING AG(sal)>1000;

Page No.
Date / 20

08/02/2024

JOIN (SQL)

Student

PK → Roll	Name	Age
1	ABC	18
2	PQR	19
3	ABC	19
4	XYZ	20

Course

Course ID	C name	Roll No ← FK
10	Comp.	1
20	Law	2
30	Physics	3

Ques → We need info about student whose course is "Computer".

TYPES OF JOINS

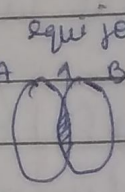
(i) Equi join / Equality Join => Matches common columns and matches.

Emp

Eno	Ename	Address
1	ABC	Delhi
2	DEF	Asr
3	ABC	Chd
4	XYZ	Mum

Dept

Dno	Dname	Eno
D1	HR	1
D2	IT	2
D3	MRKT	4



Ques → find name of emp who are working in any dept.
here equal columns are Eno and data matched is 1, 2, 4.

SELECT ename FROM (emp, dept) WHERE emp.eno = dept.eno;

(emp, dept) → cross prod / cartesian prod → emp × dept

emp.eno → emp ka eno

dept.eno → dept ka eno.

Page No.
Date / 20

Emp x Dept ⇒ (emp.eno, ename, city, dept.eno, dname, mgr)

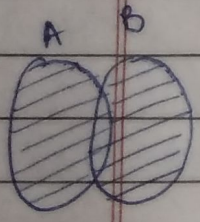
1	ABC	Delhi	D_1	HR	1	Row 1 ✓
1	ABC	Delhi	D_2	IT	2	
1	ABC	Delhi	D_3	Mrk	4	
2	DEF	Asr	D_1	HR	1	Row 2 ✓
2	DEF	Asr	D_2	IT	2	
2	DEF	Asr	D_3	Mrkt	4	
3	ABC	Chd	D_1	HR	1	Row 3
3	ABC	Chd	D_2	IT	2	
3	ABC	Chd	D_3	Mrk	4	
4	XYZ	Mum	D_1	HR	1	Row 4 ✓
4	XYZ	Mum	D_2	IT	2	
4	XYZ	Mum	D_3	Mrkt	4	

ABC, DEF, XYZ ← Ans

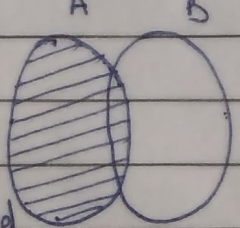
(ii) CROSS JOIN | CARTESIAN JOIN ⇒ Will not match
↳ equi join cont cond^n

↳ a cartesian join occurs when data is selected from 2 or more tables and there is no cond^n specified in the WHERE clause. It joins each row from the first to every row in the 2nd table.

Total records = m x n
(m ← no of records in first table; n ← no of records in second table.)



(iii) LEFT OUTER JOIN ⇒



The left join returns all records from table 1 (left table) and the matching record from table 2 (table right). The result is zero records from the right side if there is no match.

Page No.
Date / 20

PREVIOUS TABLE REFERENCE

Ques→ Display list of employees working in each dept. Display the dept info even if no employee belongs to that dept.

SYNTAX ⇒ SELECT columnname FROM tablename1 LEFT JOIN tablename2 ON tablename1.columnname = tablename2.column_name.

SELECT empname, address, DName FROM emp LEFT JOIN dept ON emp.eno = dept.eno;

RESULT→

ename	address	Dname
ABC	Delhi	~~Comp~~ HR
DEF	Agr	IT
XYZ	Mum	Mark
ABC	Chd	

→ Query returns null values for the columns in B for the cases where records don't match

(iv) RIGHT OUTER JOIN ⇒

SYNTAX ⇒ SELECT column_name FROM tablename1 RIGHT JOIN tablename2 ON tablename1.column_name = table2.column_name.

(v) SELF JOIN ⇒

Page No.
Date / 20

15/02/24

for loop:- for counter in [Reverse] lowerbound..upperbound
loop statement(s);
end loop;

counter variable (i) — optional — range operator
Reverse → to move the loop backward.
loop, end loop → keyword

eg → for i in 1..10

* In PL SQL, increment happens automatically in loop. *

Ques → To print table of a number.

Ans

~~begin~~
declare:
VARIABLE a int := 2;
begin
for i in 1..10
loop a*i; DBMS_OUTPUT.PUT_LINE(a);
end loop;
end; ~~DBMS_OUTPUT.PUT~~

NO NEED

OR

DBMS_OUTPUT.PUT_LINE(a||'*'||i||'='||a*i);

POINTS TO REMEMBER:-

(i) Don't declare counter variable - PL SQL implicitly declares the counter variable as a local variable ~~of~~ with datatype integer. Scope of counter → inside loop.

(ii) Counter always uses integer literals.

(iii) The loop index in for loop statement is always incremented/decremented by 1. So, if you want to specify a diff increment

Or decrement value, then you will have to write a code for that.

Eg ⇒
```
for i in 1..50 loop
if i mod 5 = 0 then
DBMS_OUTPUT.PUT_LINE (i);
endloop;
end;
```

3. To forcefully break the loop before the range has ended; use exit cond".

⇒ Loop Label ⇒

<<Outer>> — Name of the loop.

```
for i in 1 to 1..10 loop
  DBMS_OUTPUT.PUT_LINE (i);
<<Inter>>
for j in 1..5 loop
DBMS_OUTPUT.PUT_LINE (j);
endloop; OR end inner;
endloop; OR end outer;
end;
```

⇒ SEQUENTIAL CONTROL ⇒

GOTO STATEMENT ⇒ To go from label to label

```
begin:
dbms_output.put_line ('free first');
goto third;
dbms_output.put_line ('second');
<<third>>
                    ('third');
```

24/02/24 Special Operation - JOIN (Algebra)

Cross product + Condition = JOIN Symbol - ⋈

$P \underset{\text{<Condition>}}{\bowtie} Q$ - P joins Q with condition

* Any ordinary join = THETA JOIN *
* If cond^n of equality = EQUI JOIN *
* Anything instead of equality (<, >, ≤, ≥) = NON-EQUI JOIN

Theta is one of the comparison operators such that =, !=, <=, >=, >, <. A join operation C such a general join cond^n is called theta join.

THETA JOIN
- EQUI JOIN
- NON-EQUI JOIN

Ques

Employee

Eid	Ename	Age	Did
101	Ankush		02
103	Ankur		03
106	Amrit		01

Dept

Dno	Dept
01	Mkt
02	Prod
03	IT

We need EID, Ename, Age, Dname

① Cross product

EMP ⋈ DEPT (Did = Dept)

Eid	Did	Did
101	02	01
101	02	02
101	02	03
103	03	01
103	03	02
103	03	03
106	01	01
106	01	02
106	01	03

Impressions
Date
Page

(8) Answer the queries based on EMP and DEPT tables.

(a) List all the columns of EMP table.

⇒ SELECT * FROM EMP;

(b) Display employees whose salary is less than 2500.

⇒ SELECT * FROM EMP WHERE SAL > 2500;

(c) Display employees, whose name ends with the letter 'S' and who belongs to dept 10 and 20.

⇒ SELECT * FROM EMP WHERE ENAME LIKE '%S' AND DEPTNO IN(10,20);

(d) Display empno, name and job of all employee who do not earn a commission.

⇒ SELECT ~~ENAME~~ EMPNO, ENAME, JOB FROM EMP WHERE COMM IS NULL;

(e) Display total number of employees maxim-um and minimum salaries dept-wise.

⇒ SELECT COUNT(*), MAX(SAL), MIN(SAL) FROM EMP GROUP BY DEPTNO;

(6) Display average salary job-wise excluding 'SALES MAN'.

=> SELECT AVG(SAL) FROM EMP WHERE JOB NOT IN ('SALESMAN') GROUP BY JOB;

(7) Display average salary for all dept employing more then 3 people.

=> SELECT DEPTNO, AVG(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(*) > 3;

Set Operators :→

Set Operators can be used to select data from multiple tables. Set operators basically combine a result of 2 queries into 1. These queries are known as Compound Queries

The Data Type of resulting columns should match in both queries

4 types of set operators ---→

1) Union
2) Union All
3) Minus
4) Intersect

UNION :→

The Union Operator merge the output of 2 or more queries in a single set of rows or columns. It eliminates the duplicate values of 2 or more queries.

Syntax :→

```
select <statement 1>
UNION
Select < statement 2> ;
```

Define :- The union all operater merge the output of 2 or more queries in a single set of rows and columns.
It does not eliminate duplicate Values.

Syntax → select <statement1>
UNION ALL
select <statement2>

Ques - Displays jobs in dept = 20 and dept = 30 using UNION ALL operator.
select job from emp where dept = 20
UNION ALL
select job from emp where dept = 30 ;

(A - B)

MINUS → The minus operator returns the rows which are unique to the first query

Syntax → select <statement 1>
MINUS
select <statement 2>

Ques - List the jobs which are common to dept 20 and

select job from emp where dept = 20
MINUS
select job from emp where dept = 30
MINUS
select job from emp where dept = 10 ;

(U can use IN)

dept 10	dept 20	dept 30
Mgr	Clerk	sales
Prsi	Mgr	Mgr
Sales	Analyst	clerk

INTERSECT → It gives only those rows as output from both the queries which have common records among them.

Syntax :- select <statement 1>
INTERSECT
select <statement 2> ;