* JOINS: → to our Sake → it is called as operator

⇒ JOINS in SQL is a clause used to combine the elements from different tables (or) relations based on the matching (or) intersection between them.

⇒ If the elements are not matched then they will be represented as 'NULL'.

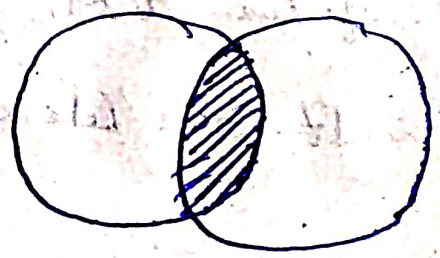
⇒ To perform join operation there exists a condition to satisfy the matching the elements b/w two relations are combined to provide the O/P.

⇒ They are 4 types:-

1. Inner Join
2. Left JOIN
3. Right Join
4. full join.

i) Inner Join:-

⇒ Inner join combines the rows (or) data elements from different tables, satisfying the condition that there should be matching elements.



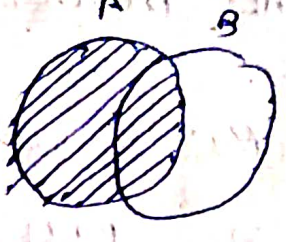
Syn:

```
Select column1, cl2, cln from
table1 innerjoin table2 where
condition;
```

2) Left Join:

→ Left join is a clause (or) operation used to join elements from two tables where it provides all the values of left side table and matching values of right table.

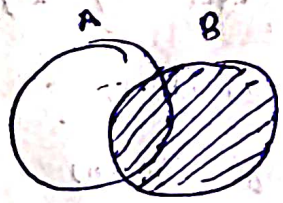
Syn:- select col1, col2, coln from table1
leftjoin table2 where condition;



3) Right join:

→ Right join is a part of join in which all the elements of right table are provided as result with matching elements of left table.

Syn: select col1, col2, coln from table1
right join table where condition;

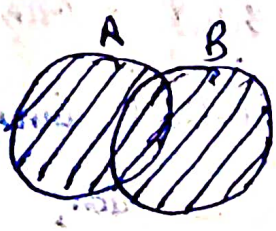


4) Full Join:

→ Syn: select col1, col2, coln from table1 full join table2 where condition;

→ full join is a part of join clause used to combine all the elements of two tables, inspite of a matching condition.

→ It acts like an union all operator.



02/4/22

Group By:

⇒ It is a clause used to group similar values in a relation.

→ Used to it to group specific elements of a column/ relation.

Syn: select columns, col2, col3 from table name where condition group by (columns);

Ex: Customers:

ID	Name	Age	Salary	Country
1	Roy	35	35000	Ind
2	Moy	25	60000	Us
3	Koy	18	18000	Ind
4	Noy	19	19000	Ger
5	Ro	16	5000	Pak
6	Ra	15	6000	Afg

Syn: select ID, NAME from customers where Age <20 group by country = 'IND';

O/P

ID	NAME
3	KOY

* Having:

→ It is a clause used to provide aggregate operations

→ Since where clause cannot be used ~~to~~ in terms of aggregate operator having comes to play.

→ Used after where clause and groupby in a query.

Syn: select cols, cols, cols from table name where condition group by column having operator (column);

Ex: Select country, (ID, NAME) from customers where Age >20 group by salary Having count (salary < 60,000);

o/p:

ID	NAME
1	ROY

* Views:-

→ It is a part of SQL used to create user specification on a relation.

→ It is called as virtual tables.

→ Depending on the user requirements views can be created on a single relation (tables or from multiple tables/relations)

04/12/2

Syn:

Create view view_name AS select column1, col2, coln from Table name WHERE condition;

Eg

~~CREATE VIEW~~

Eg Create view ID, ~~Name from~~ AS select ID, ~~Name from~~ Student where Age > 15;

Table

Student

ID	NAME	Age	Sex	Mail
1	Roy	17	m	xx
2	Ro	18	f	yy
3	Ra	18	F	zz
4	Ja	16	m	mm
5	Pa	20	m	nn

O/P

IDV

ID	NAME
1	Roy
2	Ro
3	Ra
4	Ja

Eg

Create view SF AS select ID, Name, FNAME, SUB from Student, faculty ~~where~~ order by ID;

Table faculty

FID	FNAME	Sub
001	Vijaya	IOT
002	Chand	AI
003	Vinod	Cloud
004	Abhi	ML
005	Shiva	Robotics

O/P

SF

ID	NAME	FNAME	SUB
1	Roy	Vijaya	IOT
2	Ro	Chand	AI
3	Ra	Vinod	Cloud
4	Ja	Abhi	ML
5	Pa	Shiva	Robotics

* To delete view:

Syn Drop view viewname;

→ Views are of 2 types:-

i) User views

ii) System views:-

i) User views:-

→ User views are created based on the requirements of a database user.

ii) System views:-

→ These are created automatically based on the application specifications.

→ Additional db elements, temporary data base, Oracle db.

* TCL:-

→ Transaction control language.

→ Any query, involving in a operation will go through series of execution in a database session.

→ Transaction means complete series of steps involved in a execution process

⇒ In SQL this transactions are controlled by 3 commands

① Commit

② Roll back

③ Save points.

* Commit:

⇒ It is a TCL command used to permanantly store the elements of a relation on to the disk space.

⇒ As a user once commit is performed further changes in the same session / further roll back can't be performed.

Syn: COMMIT;

* Save Points:-

⇒ It is a TCL command used to create pointers for on a relation.

⇒ Save points are pointers created for future changes which can be accorded with the relation using save point name.

Syn SAVE POINT SAVEPOINT (NAME);

Ex SAVEPOINT SP1;

UPDATE STUDENT SET AGE = 15 where ID = 5;

SQL> SAVEPOINT SP1;

SQL> Insertion : ... ;

SAVEPOINT SP2;

SQL> Rollback SP1;

* Rollback:

→ It is a TCL command used to roll back to previous state of a relation using savepoint name.

Syn Rollback savepoint (name);

Ex SQL> Rollback SP2;

* Cursor: (Pointer/Reference):

⇒ Cursor is a temporary storage space created in the server.

⇒ Where it acts as a space to work and a pointer to fetch elements from relation/table.

⇒ Cursors are performed in 2 ways

i) Implicit Cursor

ii) Explicit Cursor.

i) Implicit Cursor:-

⇒ These are the cursors which are defined automatically by the database system when insert, update, delete statements are executed.

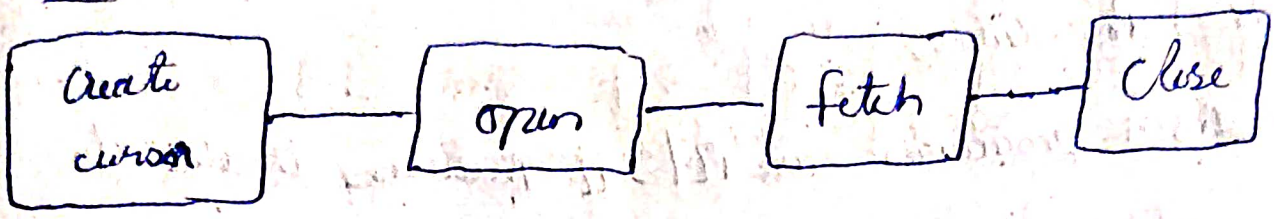
⇒ If a select statement works on 1 row then implicit cursor can be defined.

06/12/22

Explicit Cursor:-

⇒ These are the cursor which are defined manually by the user where the select statement works on multiple rows.

Steps involved in a cursor:-



Eg:-

```
Set Serveroutput ON;
declare V_Sname sailors.sname %type;
  V_age sailors.age %type;
cursor c2 is select sname, age from sailors;
begin;
open c2;
loop
fetch c2 into V_Sname, V_age;
exit;
when c2%rowcount >3;
dbms_output.putline(V_Sname || " " ||V_age);
End loop
Close c2;
End;
```

o/p:-

Dustin 38
Brutus 48
Lubber 68

* To deallocate the cursor space.

Syn

SQL> deallocate cursor name;

Ex deallocate c2;

* Stored Procedures:

→ Stored procedures are PL/SQL programs created to evaluate and work for specific purpose.

⇒ Whenever the user wanted a statement to be executed the users can execute the stored procedures at that time.

Syn create (or) replace procedure (Procedure name/Parameter);

declare

main statement;

Exit;

Ex set serveroutput on;

Create or replace Procedure

Sail (c_id int) is

declaration { v_sname varchar (18);
v_age number (18);

begin

Select sname, age into v_sname, v_age from salors;

dbms.output.putline ('Sname:' || V_Sname);

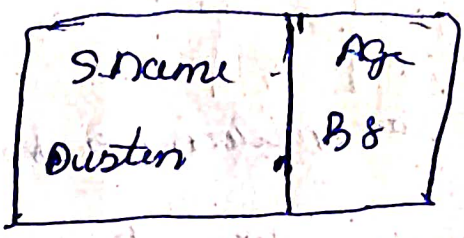
dbms.output.putline ('Age:' || V_age);

/

Execute (x);

End;

O/P



→ Trigger is for event & stored procedure used for purpose created a

Stored Procedure is used to define a program which is used again and again for the most important purpose of a database.

UNIT-

9/12/23

Schema Refinement and Normalisation

→ Schema → structure / definition of a table / relation

→ Problem in terms of Anomalies (Insertion / deletion / updation)

Problem → Redundancies

→ Refinement → By decomposition → breaking down the original table / relation. data elements into multiple relations with equivalency → means based rules on axioms.

→ Axioms rule → depends on functional dependency is the work.

→ Normalisation → it process

→ It is procedure to normalize the values of the relation

→ These are of 6 types.

i) 1NF → first normal form
ii) 2NF → second normal form
iii) 3NF → third normal form
iv) BCNF → Boyce codd
v) 4NF → fourth normal form
vi) 5NF → fifth normal form

} based on axioms rules

11/12/23

* Schema Refinements:-

→ Schema refinement is a process in database used to refine database when the data suffers from redundancy and anamolies.

→ Unordered and un-structured databases with redundant data are refined to eliminate unwanted redundancy and inconsistency.

→ Redundancy means same copy of data exists in multiple location.

→ Anamolies are problems due to inconsistency and redundancy by which a poorly planned database (or) un-normalised database.

→ Schema refinement is an process achieved through techniques of decomposition of database and Normalisation.

* Anamolies of database:-

→ There are 3 types of anamolies:-

① Insertion } These anamolies happens when there is a
② Deletion } possibility of redundancy

③ Updation → This happens when updation is performed it leads the database in an inconsistant state.

Student

SID	SNAME	CID	CNAME	FEES
1	Roy	4	CS	5K
2	Moy	C_2	AI	5K
3	Koy	C_3	ML	8K
1	Roy	C_2	AI	5K
2	Moy	C_2	AI	5K
4	Joy	C_3	ML	8K

* Insertion Anamoly:

→ It is a scenario, when trying to insert a new value in columns, the dummy values and null values also to be inserted, into a relation, and this creates the problem

→ Insert a new course C4;

insert into student values ("NULL", "NULL", "C4", "IOT", "15K");

⇒ This scenario leads to insertion anamoly, leaving relation or database in a inconsistant state.

* Deletion Anamoly:-

→ If a deletion of values in a columns has to be performed the entire row to be deleted

⇒ This leads the database to inconsistant state.

⇒ Delete sid = 4

* Updation Anamoly:-

→ If specific column value is to updated, then all the values associated with it should be updated,

→ If not leads to updation anamoly.

⇒ Update fee 5k to 8k ✗ ↳ it will be problem because the data has 3, 5k rows

→ This will lead database as inconsistant data.

⇒ Further decomposition will solve all the anamolies problems.

* Functional Dependency:- Based on Armstrong Axiom rules
① Reflexivity ② Augumentation ③ Transitivity

⇒ It is rule that attribute values one set is functionally dipend on another set.

⇒ It preserves integrity constraints of a data base.

$t_1.x = t_2.x$ Then $t_1.y = t_2.y$

where t_1, t_2 are tuples and x, y are attributes.

⇒ If $A \rightarrow B$ then, B is functionally dependent on A.

=) Say set R contains value (A, B, C, D) and divided into $R_1(ABC)$ and $R_2(ADD)$, then it is said that

$A \longrightarrow BC$ and $A \longrightarrow BD$, where

R_1 is a part of R, then it is depends on sets

Revers when divided.

* Decomposition:-

=) Breaking down the database into multiple tables to eleminate redundancy & Inconsistency.

SID	SNAME	CID
1	Roy	C1
2	Moy	C2
3	Koy	C3
1	Roy	C2
2	Moy	C2
4	Joy	C3

⋈

CID	CNAME	FEES
C1	CS	5k
C2	AI	9k
C3	ML	8k

2 types of decomposition:

① Lossless decomposition

② ~~Decomposition~~ Dependency Preserving

③ Lossy

13/12/23

SID	SNAME

SID	CID

CID	CNAME	fees

* Functional Dependencies:-

→ functional dependencies is a rule based concept imposed by Ed.f. Codd in relational database.

→ It is used to handle the redundancy and bad design of relational database.

→ It is based of AXIOM's RULE derived by William Armstrong which is mathematical entity for solving problems

* ~~Armstrong's Axioms Rule~~

~~→ It is a rule based~~

→ functional dependency is used to find the relationship between attributes of relation.

→ If X and Y are attributes of a relation R then there exist a functional dependency between X and Y as $X \rightarrow Y$ where $X \rightarrow Y$ where X is the determinant and Y is the dependent

ARMSTRONG AXIOM's RULES:

→ It is a rule based theory and mathematical entity for handling complex structures.

① Reflexivity (X→Y):

→ If Y is a subset of X then it holds, X→Y by reflexivity rule, where Y depends on X

Eg:-

Table

RollNo	NAME	Dept_name	Dept_block
1	R0	CSE	M
2	X0	ME	C
3	M0	IOT	D
4	K0	CSE	M
5	L0	EC	C
6	J0	IOT	E
7	Y0	IOE	C

Ex: {RollNo} → {Name}

{RollNo, Name} → {RollNo}

where {Name} (or) {RollNo} is a subset of {RollNo, Name}

② Augmentation (ZX→YZ):

→ If X holds attributes Y and Z, the functional dependency ZX→YZ, then it is said to be valid dependency by

augmentation rule.

Ex:- {Dept name, Name} → {Dept_name}

{name, dept_name, dept block} → {dept name}

③ Transitivity (X→Y)(Y→Z)(Z→X) :-

→ If X holds Y and Y holds Z then X holds Z by transitivity rule (X→Y)(Y→Z)(Z→X)

Ex

{Roll no, name} → {Name}

{Name, dept name, dept block} → {dept block}

{Roll no, Name} → {dept. block}

In a relation there exists attributes and relationship b/w these attributes & 1 attribute determining another attribute.

* Types of Functional Dependencies

① Trivial functional dependency. (insignificant)

② Non-Trivial functional dependency

③ Multivalued functional dependency

④ Transitive functional dependency.

① In trivial functional dependency, if X holds Y, functional dependency can be said as $X \rightarrow Y$.

Roll No	Name	Age
1	Ro	23
2	Gro	24
3	Jo	32
4	Po	31

→ Y is a subset X, and Y is dependent on X

Eg $\{\underset{X}{\text{Roll no, Name}}\} \rightarrow \{\underset{Y}{\text{Name}}\}$

$X \rightarrow Y$

② In non-trivial functional dependency, the dependent is not a subset of determinent. i.e Y is not a subset of X.

Eg : $\{\underset{X}{\text{Roll no, Name}}\} \rightarrow \{\underset{Y}{\text{Age}}\}$

where {Age} is not a subset of {Roll no, name}

③ In multivalued dependency, the dependent is a subset of determinant, but the attributes of dependent will not depend on each other.

$X \rightarrow Y, Z$

where X holds YZ, YZ depends on X and YZ will not dependent each other.

Eg: $\{Roll\ no, name, age\} \longrightarrow \{name, age\}$ (X → YZ)

But

{Name, Age} is set where they are not dependent.

(ii) In transitive dependency X holds Y and Y holds Z and then X holds Z by transitivity rule of Armstrong Axioms

Eg

X Y
Roll no → Name

Y Z
Name → Age

X Z
Roll no → Age

* Valid Dependencies:

18/12/23

Normalisations :-

⇒ It is a procedure / guideline based on functional dependency used to normalize tables / relation with problems / redundancy and Inconsistency.

⇒ Normalization is used to remove anomalies (Insertion, Updation, Deletion)

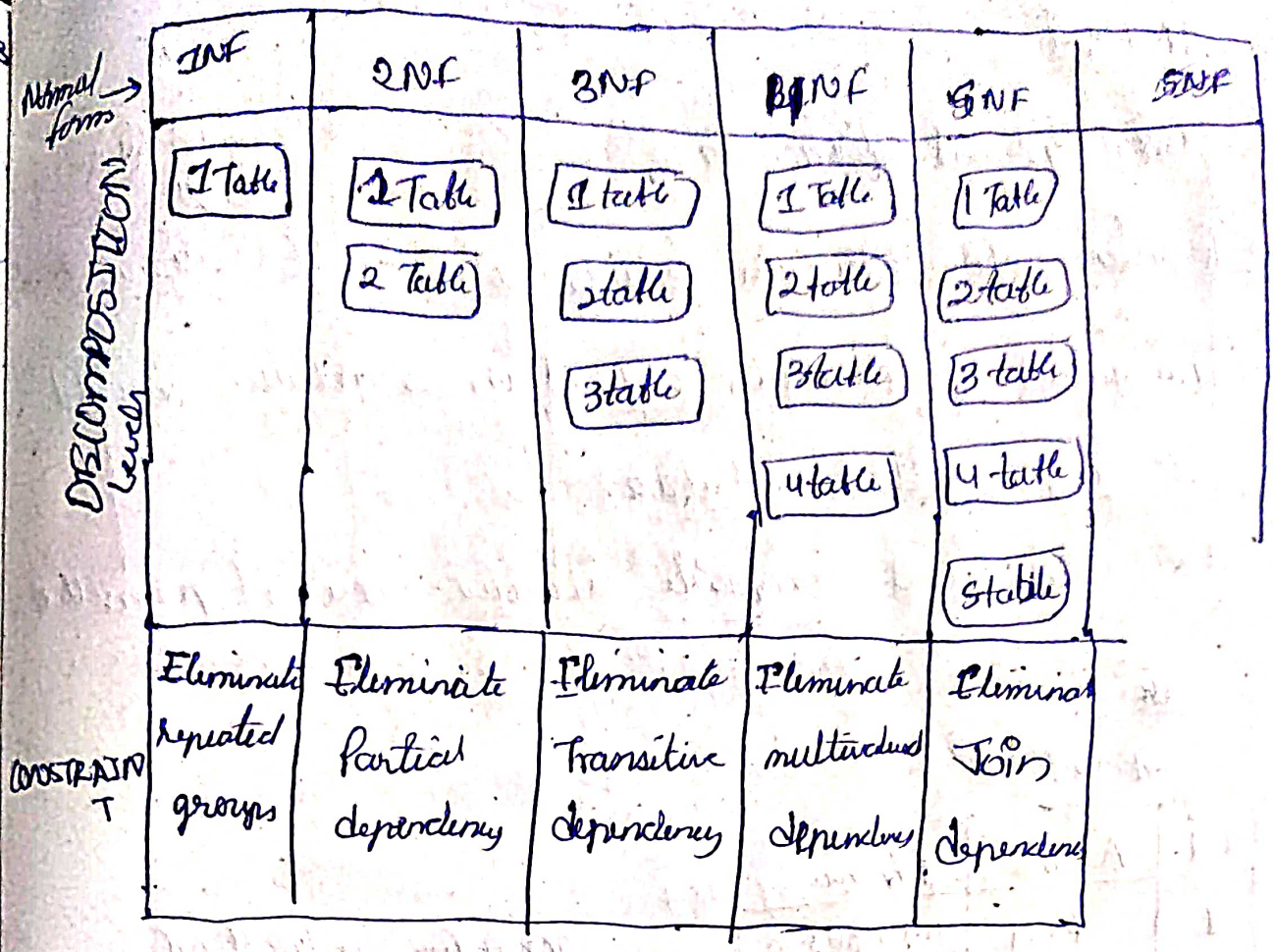
⇒ Normalization follows the principle of dividing a relation with complex structure into ease of use.

⇒ It was invented by Edgar F. Codd to sort complex data base design.

⇒ Normalization comes with guidelines (rule / constraint) called normal forms

⇒ Types of normal forms include :-

- 1NF
- 2NF
- 3NF
- BCNF → Boyce & Codd normal form
- 4NF
- 5NF



* Advantages:-

1) Eliminates redundancy & inconsistency
2) Eliminates inconsistency
3) Integrity is preserved.
4) Achieves great flexibility. → makes the user to feel ease with the data base

* Disadvantages:-

1) When higher level of normalization performs it leads to problem.
2) Higher level of normalization ie 4NF, 5NF leads to delay in Retrieval of record.

① 1NF:-

⇒ First normal form :

⇒ Relation with attributes values should be atomic. (single value)

⇒ All values of attributes defined in a relation should be single valued attribute.

⇒ Multivalued and composite attribute are not permitted.

Ex

Student

Id	Name	Ph_no
1	Roy	98486858 & 4565432851
2	Roy	9912531514
3	Varsha	7869333542

⇒ Converting into 1NF

ID	Nam	Phno
1	Roy	98486858
1	Roy	4565432851
2	Roy	9912531514
3	Varsha	7869333542

Reason → since the above ex suffers from multivalues attribute Ph_no

* 2NF:-

⇒ A relation is in 2NF if it is in 1NF and no partial functional dependency exists.

browser

20/12/23

Student

SID	S_Name	Age
S_1	Roy	5
S_2	Kavy	6
S_3	Moy	7

$x \rightarrow y$ → Fully FD

$x \rightarrow z$ → Partially FD

So we decompose it

→ Since the above table/relation does not have full functional dependency bt/s SID & Age we decompose it

Student

SID	SName
S_1	Roy
S_2	Kavy
S_3	Moy

StudentAge

SID	Age
S_1	5
S_2	6
S_3	7

* 3NF:-

→ A relation is in 3NF then it should be in 2NF and it does not hold a transitive dependency bt/s attributes.

→ A relation is in 3NF if it holds a condition for all non-trivial functional dependency $x \rightarrow y$

where → x is a super key

→ Y is a prime attribute (all the elements of Y is a part of candidate key)

* for Emp:

EmpID	Emp_Name	Emp_Zip	Emp_State	Emp_City
01	Jeni	60014	Chikago	NY
02	Keni	50005	Dalas	XY
03	Reni	70015	Texes	YY
04	Meni	89004	Southern	NY
05	Deni	59001	Northean	MY

Super key: {EmpID}, {EmpId, EmpName}, {Emp_Id, EmpName, Emp_Zip}.... so on....

Candidate key : {EmpId}

* Since the above table holds a transitive dependency bcs

EmpID → {Emp_State, Emp_City}, we decompose it.

* Emp - details:-

EmpId	Emp_name	Emp_zip
01	Jeni	60014
02	Keni	50005
03	Reni	70015
04	Meni	89004
05	Deni	59001

Emp-address:

Emp_zip	Emp_state	Emp_city
60014	Chicago	NY
50005	Dallas	XY
70015	Texas	YY
89004	Southern	NY
59001	Northeast	NY

* BCNF:

→ BCNF is an higher form of 3NF and a relation is in BCNF if all non-trivial functional dependency $a \rightarrow y$ ensure and x is a superkey and it is 3NF

* Ex

Student

Id	Teacher	Subject
Roy	John	Database
Roy	Luke	C
Ravi	John	Database
Ravi	Max	C

Candidate keys:

(Student, teacher)

(Student, subject)

FD: (Student, Teacher) → subject

FD: (Student, subject) → teacher

Teacher → subject

x → y

Student	Teacher
Roy	John
Roy	Luke
Ravi	John
Ravi	Max

Teacher	Subject
John	Database
Luke	C
Max	C

23/12/24

* 4NF:-

→ If a relation is in 4NF then it should be in 3NF or BCNF

→ No multivalued dependency present in the given relation

Ex R: Student

StdId	Subject	Hobby
21	Math	Singing
21	Computer	Dancing
34	Math	Singing
64	Chemistry	Swimming

Std - course

StdId	Subject
21	Math
21	Computer
34	Math
64	Chemistry

Std - Hobby

StdId	Hobby
21	Singing
21	Dancing
34	singing
64	swimming

Ex Since the relation R student sustains with multivalued dependency it is decomposed into 2 tables to remove the problem.

5NF: (Project Join Normal-form)

→ If a relation is in 5NF, then it should be in 4NF

→ NO join dependency should exist and Join is lossless

* Ex

Subject	Lecturer	Semester
Maths	John	Sem1
Computer	John	sem1
Maths	Duke	Sem1
Chemistry	Mary	Sem2
Maths	Coan	sem 2

Candidate key : {Subject, lecturer, semester}

Primary key

→ Since the above relation persist with multivalued and Join dependency it is decomposed into 3 tables

Semester	Subject
sem1	maths
sem1	computer
sem2	maths
sem2	Chemistry

Subject	lecturer
Maths	John
Maths	Duke
Computer	John
Chemistry	Mary
Maths	Coan

lecturer	Subject
John	sem1
Duke	Sem1
Mary	sem2
Coan	sem2

27/01/23

* Decomposition Types in DBMS:-

→ It is a method used in DBMS, where the relation with problems can be divided into multiple tables for ease of use and retrieval.

→ It is implemented in Normalization:-

* Types:

i) lossy

ii) loseless

iii) Dependency Preserving

i) lossy:-

→ Decomposition of lossy type will exhibit loss of data when combined after a relation is divided.

[Ex]

StdId	NAME	DEPT	COURSE
01	Roy	CSE	ML
02	Indra	CSE	IOT
03	Moy	CSM	ML
01	Indra	CSE	ML
02	Chandra	CSD	AI
04	Bandra	CSC	IOT

Student

StuId	Name
01	Roy
02	Indra
03	Moy
01	Indra
02	Chandra
04	Bandra

Dept

Deptt	Course
CSE	ML
CSE	IOT
CSM	ML
CSE	ML
CSD	AI
CSC	IOT

* Lossless :-

→ It is a method used to preserve the originality of a relation, when it is decomposed into multiple tables.

→ if the data elements into multiple table is Lossless.

Ex

Stud_Id	Name	Dept_ID	Dept	Course
01	Ra	D_1	CSE	ML
02	ka	D_2	CSM	IOT
03	Ja	D_3	CSD	ML
04	Pa	D_2	CSM	ML
02	Ra	D_4	CSC	AI

→ Since the given table holds redundancy & anomalies, we decompose it.

Stid → Primary Ke

StuId	Name	DeptId → foreign key
01	Ra	D_1
02	ka	D_2
03	Ja	D_3
04	Pa	D_2
02	Ra	D_4

Primary key

DeptId	Dept	Course
D_1	CSE	ML
D_2	CSM	IOT
D_3	CSD	ML
D_2	CSM	ML
D_4	CSC	AI

Dependency Preserving:- (making faster for a long time)

→ A relation R is said to be in dependency preserving if a Relation R is sub divided in R_1 & R_2 then either the data elements from $R_1 \cup R_2$ should be a part of Relation R.

Ex: R (ABCD) → FD → A → BC

elements

R (ABCD)

R_1 (A, B, C) R_2 (AD)

UNIT-5

Transaction Processing:-

⇒ T→ all the executable statements accomplishment is called Transaction.

→ In DBMS transaction means

i) Read
ii) Fetch
iii) Retrieve
iv) Save

* ACID Properties of DBMS:-

⇒ There are properties used in DBMS to ensure quality of data and maintaining Integrity constraints over relations.

⇒ Ensuring DBMS performance.

* Properties:-

i) Atomicity
ii) Consistency
iii) Isolation
iv) Durability

← Out of this these 4 are imp

7 factors of good database sys

1) Atomicity
2) Redundancy
3) Security
4) Consistency
5) Isolation
6) Concurrency
7) Integrity

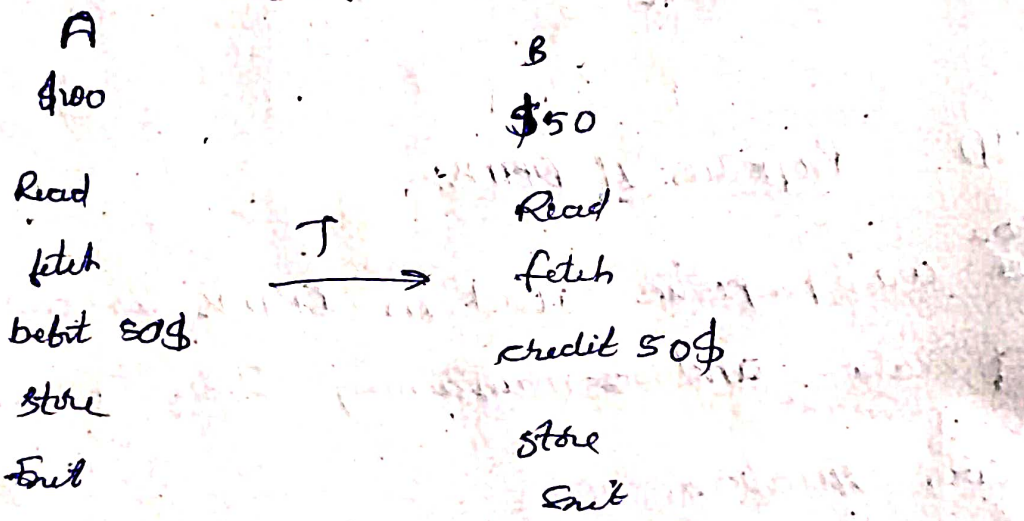
1) Atomicity :-

⇒ The data values which persist in a database, should insure that all the values are atomic in nature.

⇒ When it comes to transaction the atomicity is measured by either complete successful transaction (or) the transaction is rolled back.

Ex:

Atomicity



⇒ Atomicity won't leave any transaction in a intermediate state.

⇒ If the transaction fails atomicity is ensured by bringing A's account to its original state.

2) Consistency:-

→ Ensuring the changes of data values in a database after every transaction in a consistent mode.

→ Every change made in database is consistent and the relation (or) data in it is readily available for next transaction without any problem.

Eg:

A's acc
$300
debit B - 50$
debit C - 50$
fetch - 200$
store - 300$

B's account - $75
credit - 50$

C's account - 50$
credit - 50$

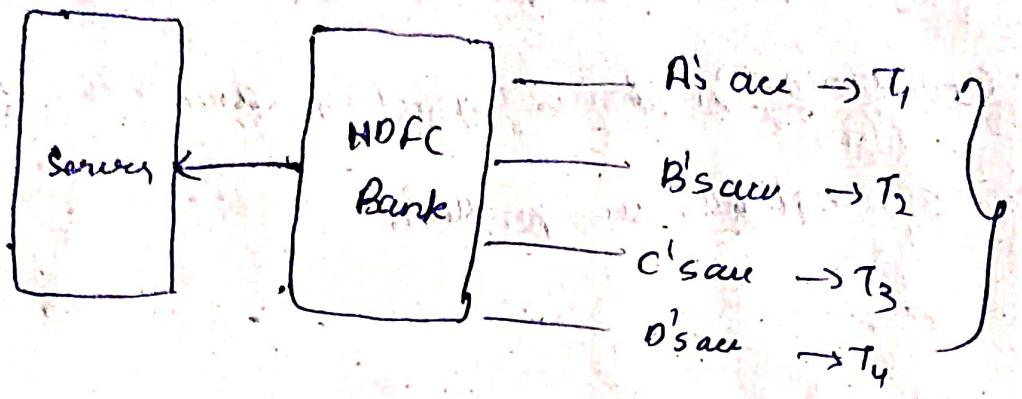
3) Isolation:-

→ Isolation is an ACID property where the transaction working in a database, perform its function in a isolated way.

→ i.e One transaction does not disturb or Modulate another transaction.

Ex

⇒ Banking database contains 'n' no of transactions in any given time. Each transaction expected independent of any another.



Q) Durability:

⇒ It is a ACID property used to ensure the quality of data in a database by ensuring the sustainability of database even when a failure or crash occurs.

⇒ Durability is to ensure durability either copy of same database is available at another location.

Or Crashed recovery programme is displayed.

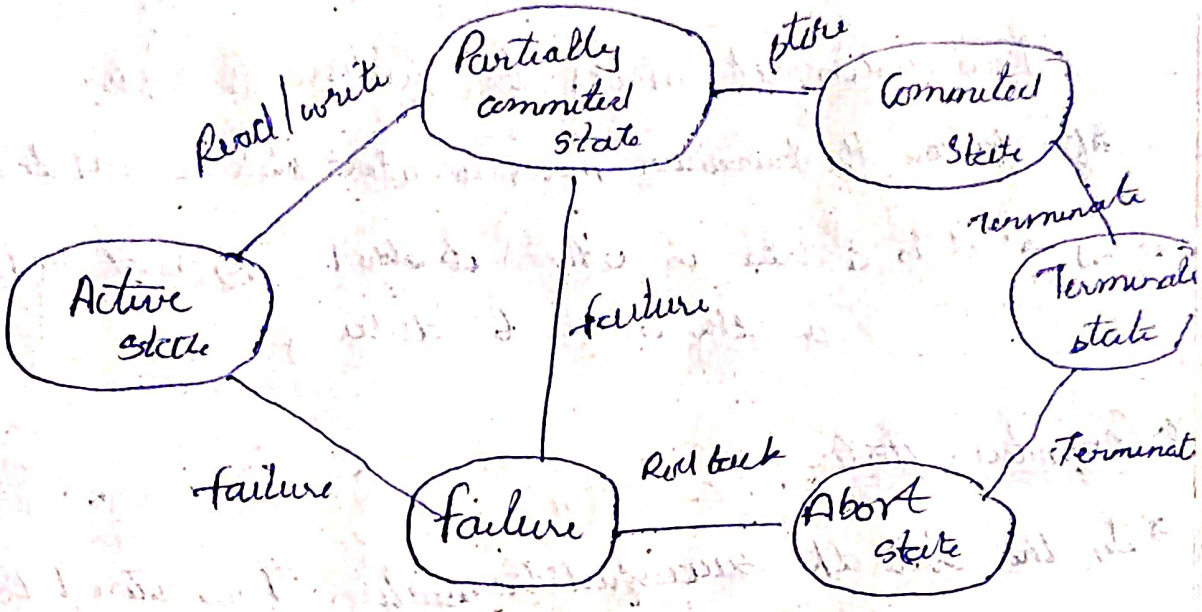
↓

ensured by database program or DBA.

Transaction States:

⇒ Transaction state: is a policy which records the lifetime of a transaction until it is state and completes.

⇒ Every database system ensures a database log, where all your transaction matters are maintained.



i) Active State:

⇒ It is the first state of a transaction, where the instruction of a transactions tries to read/write the data elements in a data base.

ii) Partial Committed State:

⇒ If read/write is successfull it moves to partially committed state.

⇒ Transaction data is in buffer (s) main memory

⇒ It records the successfull read/write

* Failure state:-

→ If the read/write is unsuccessfull in a active state it moves failure ~~transaction~~ states.

⇒ If failure is a problem to commit the successfull transaction it moves to failure states.

→ After failure the transaction moves to abort state & roll backed.

⇒ log → it is a data in which it stored by what is happening & the entries of data

* Commited state:-

⇒ In this state the successfull transaction T is stored to the database and it moves transaction to terminated state, whes transaction is terminated.

Database works on → multi-user environment
- serial → one after the another
- concurrent → at the same time we can access

Interleaving → One access the data & the other access the data based on the result of T^t one and same continues.

08/11/24

* Concurrency in transaction:-

→ Concurrency → simultaneous accessing/operating on a base (common def)

* Concurrent Accessing of database:-

→ It works in multi-user-system, where no of reads and writes are performed by users.

→ Read(A) means Read value of (A) and store in the buffer of main memory.

→ Write(A) means write value (of) A to the database from buffer

* Problems in Concurrent Execution:-

→ When the accessing of database is concurrent and the transaction trying to access are not interleaved in fashion then there arises series of problems, which we call as concurrency problems in database.

→ The most typical problems are

1) lost-update Problem (LU-U) (write-write)

2) Dirty read Problem (R-W) (Read-write)

3) Unrepeatable read Problem (W-R)

Lost update Problems: (W-W):-

→ It is concurrency problem happens in an concurrent execution of transactions, where the concurrency leads to loss of final update of data in a database.

Ex

→ Say A has = $300 and two transactions tries to concurrently execute on A.

time	Tx	Ty
t1	Read (A)	
t2	A = A-50	
t3	-	Read (A)
t4		A = A+100
t5	write (A) (i.e 250)	
t6		write (A = $400)

Result will be the final write operation of Transaction (Ty)

A = $400) which leads to inconsistency

→ from this example it is clear that the previous update of write (Tx) is lost.

Dirty Read Problem (R-W) :-

- Dirty read problem in concurrent execution happens after the final write of a transaction if the server goes down, then the transaction is the rolled back.
- which leads to the dirty read i.e two different read is performed & moving further to an inconsistant state.

A = $250

Time	Tx	Ty
t_1	Read (A)	
t_2	A = A-50	
t_3	write (A)	
t_4		Read (A)
		A = A+50
t_5	Server down → rolled back to initial state (i.e = 250$)	
t_6		write (A)

t_3 writes (A) i.e 300

- It happens when the result of transaction T_x is read by T_y after final write of t_1 and the server goes down for T_x it rolls back to initial state
- Now T_y is in dirty read problem.

3) Unrepeatable Read Problem (URP):

→ It happens in concurrent execution of transactions when a single transaction reads the value of database multiple time when multiple transactions are working on commit of database.

Eg:

Ac $100

Time	Tx	Ty
t_1	Read (A)	
t_2		Read (A)
t_3		A = A - 50
t_4		write (A) → $50
t_5 50$	Read (A)	

⇒ At time t_1 & t_3 the Transaction Tx confuses it which value it should consider.

⇒ So it leads to inconsistancy in data.

* How to overcome the concurrent execution problem:

- These are algorithms to overcome these problems in a variety of scenarios.

1. Time stamp protocal
2. Lock based protocal.
3. Validation Rous Protocal.