01 July
Wednesday

2015
Week 7th Day 183

Important

8.00

9.00

10.00

11.00

12.00

1.00

2.00

3.00

4.00

Turned
table
from memory
while keeping
structure
intact

DBMS

domain constraint
weak, strong
entities

2015

nday		6	13	20	27
uesday		7	14	21	28
Wednesday	1	8	15	22	29
Thursday	2	9	16	23	30
Friday	3	10	17	24	31
Saturday	4	11	18	25	
Sunday	5	12	19	26	

2015 July 04
Week 27th Day 185th Saturday

Scala

Database System (consists of) → (operations) DBMS

(collection of related data) Database

RDBMS
- Structured
  - IRCTC
  - University
- Unstructured
  - Webpages

DBMS
- SQL server
- Oracle
- MySQL (by Oracle)
- DB_2 (by IBM)

Structured :-
Structured data is stored in form of tables ie relations (RDBMS)
RDBMS is a system that allows us to insert, delete, update and perform many other operations on relations

Unstructured :-
There is no particular way or format to store data. Most of the tech these days use unstructured data.

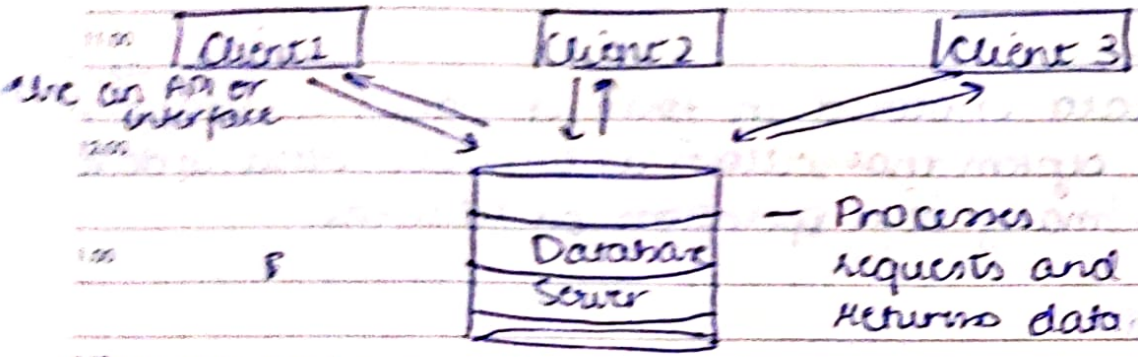
Why did we start using DBMS more than file system?
1. Searching is faster and uses memory utilisation is efficient because using SQL queries, we fetch only the needed data whereas in file system, entire file would have been fetched.
2. To access files, we need to first have attributes/metadata. But DBMS is independent and we send request to server using SQL queries.
3. Concurrency : Multiple people can access same data at same point of time as there is no such protocol in file system. There could be inconsistency in data of file system.
4. Security : There is role based security in DBMS. No such security in file system as once you enter your PC, anyone can access any data.

AUG SEP OCT NOV DEC

05 July Sunday

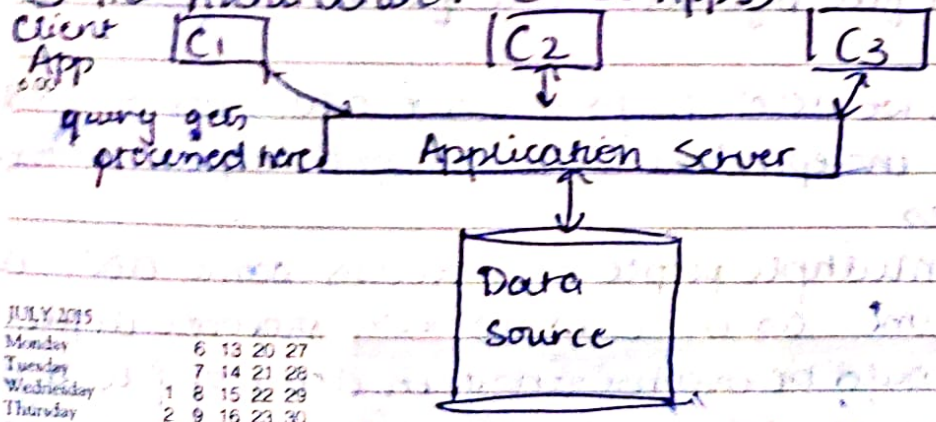
5) Boundary :- There are many constraints like foreign key, primary key etc which help in data integrity, to ensure data isn't redundant

2 Tier Architecture :- (Banks)
2 layers — Client layer
Database server



Advantages :-
The maintenance is easy.
Disadvantages :-
Scalability : Many clients can't use it together ..
Security :- Direct interaction of client-server thus harmful

3 Tier Architecture :- (Web Apps)



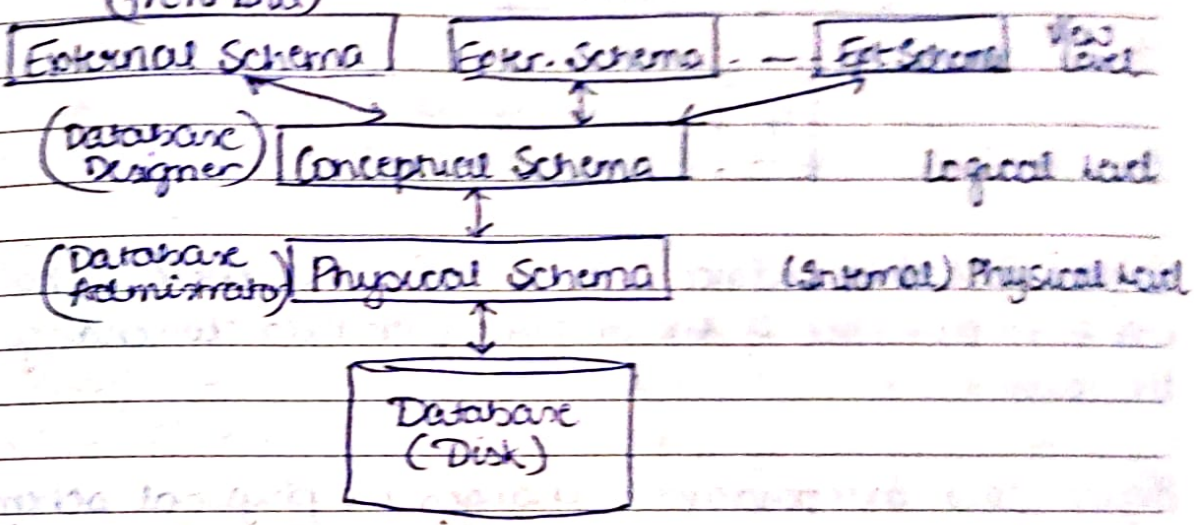
Maintenance is high but secure and scalable

2015

Date 06
Monday

Schema in Database :-
It is logical representation of database from database
Student schema : | Roll no | name | address |
This schema can be implemented using SQL (Structured query lang.) We use various DDL (Data definition commands) to implement schema like create table, etc

Three Schema Architecture / Three level Abstraction :-



Main motive : Data independence ie user must not know where the data is
View level : How data should be visible to users. This view is different for all kinds of users like student, teachers are different
Conceptual Schema :- Structure of data in database. | Eno | Name |
Relationships b/w different Tables of database etc
Physical Schema :- It contains database administrator which decide what data is to be stored and at which locations
Data can be stored in 2 ways : centralised (entire data at 1 location) and distributed (data stored widely all across globe)

07 July 2015
Tuesday 4 Week 28th Day

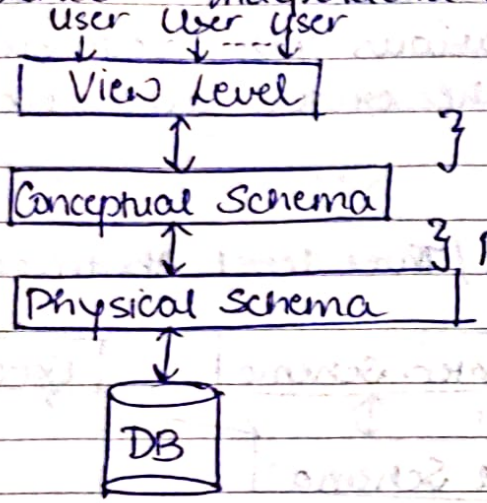
Important

User sees the data in form of tables but in the database (disk) it is stored as files.

8.00

Data independence :- Independence of data from user.

9.00



10.00 } Logical Data Independence

11.00 } Physical Data Independence

12.00

1.00

2.00 Logical Data Independence :- To ensure that there is no change on view level if one or other user tries to change the columns.

3.00

4.00 Physical Data independence :- Changes in physical schema will not affect the conceptual schema. Lets say data is

5.00 in hard disk 1 and now we move it to hard disk 2, this doesn't mean that name and structure of tables will change

6.00 Storage structure change, DS change, index change will not affect the conceptual schema.

Key :-

It is an attribute from table, which helps to uniquely identify tuple. (Single row of the table)

EG :- <Aadhar Card, Roll no., Email>

JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
dnesday	1	8	15	22	29	
sday	2	9	16	23	30	
	3	10	17	24	31	
day	4	11	18	25		
ay	5	12	19	26		

collection of all these uniquely identifying keys is called candidate key.
We choose one that is most unique and call it the primary key.
The remaining are called alternative keys.
Primary key is chosen from set of candidate key.

Primary Key :-
{ unique + NOT NULL }
There can be only 1 primary key in database because it is tough to create it since it should be both unique and not null. But there can be many candidate keys.

Foreign Key :-
It is an attribute or set of attributes that references to primary key of same table or another table (relation)
It maintains referential integrity.

PK ← FK

Students

RNo	Name	addr.
1	A	Delhi
2	B	Chd.
3	C	Bombay

Courses

Course Id	C.Name	RNo
C1	DBMS	1
C2	OS	2

Students is the referenced table and courses is the referencing table. We cannot add RNo 10 in the referencing table directly because it doesn't exist in the referenced table.

09 July 2015
Thursday
Important

FK should have same values as primary key or lesser

6 Week 28th Day 190th

Creating table with FK

⇒ create table Courses
(course id varchar(10),
C.Name varchar(20),
RNo int references
Students (R.No)
);

Altering already made table

⇒ alter table courses
ADD constraint FK
foreign key (RNo)
references Student (RNo),

Operations on Referenced table (Base Table)

1. Insertion :- No violation

2. Deletion :- may cause violation

Solution :-

- On delete cascade :- Delete the related data from all other related tables also. (best)
- On delete set NULL :- Set the Foreign key value as NULL but it could by PK of that table.
- On delete No action :- The row does not get deleted from base table before we delete the related data.

3 Updation :- May cause violation

Set :- on update cascade
on update set NULL
on update no action

Operations on Referencing table (Foreign key table)

1. Insertion :- May cause violation
2. Deletion :- No violation
3. Updation :- may cause violation if we try to update the foreign key. Rest are fine.

JULY 2015

Monday	6	13	20	27
Tuesday	7	14	21	28
	1	8	15	22	29
	2	9	16	23	30
	3	10	17	24	31
	4	11	18	25	
	5	12	19	26	

Super key :- It is combination of all possible attributes which can uniquely identify two tuples in a table.
Candidate key + any other key/keys = superkey.

Super set of any candidate key is a super key.

R(A1, A2, A3, A4 ..... An) then how super keys are possible ?

① If A1 is candidate key.

② A1, A2 are candidate key.

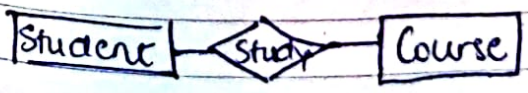
Sol(1) :- Here A1 has to be included all the time.
∴ possible super keys = 2^{n-1}

Sol(2) $\underset{A_1 \text{ always}}{\left(2^{n-1}\right)} + \underset{A_2 \text{ always}}{\left(2^{n-1}\right)} - \underset{A_1 \text{ and } A_2 \text{ common}}{\left(2^{n-2}\right)}$

Q :- R(A1, A2 ..... An), then find possible super keys if A1A2, A3A4 are candidate keys.

Sol :- $\underbrace{2^{n-2}}_{A_1A_2 \text{ fixed}} + \underbrace{2^{n-2}}_{A_3A_4 \text{ fixed}} - \underbrace{2^{n-4}}_{\text{common } A_1A_2A_3A_4}$

E-R Model (Entity Relationship Model)
This is for logical representation of data ie conceptual view.
Entity :- Any object having physical existance
Eg Student :- attributes - name, age, R.No. & schema.



11 July 2015
Saturday
Week 28th Day

Important

Entity | Attribute type | Relationship

Types of Attributes in ER model:
Attributes tell about characteristics of an entity.

1. Single V/S multivalued
Reg-No | Phone no. — 9816426076, 7876775747

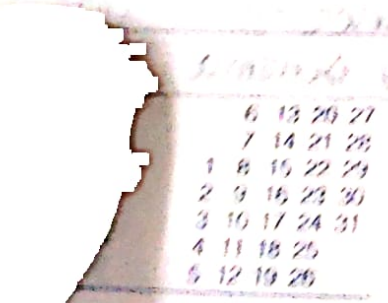
2. Simple V/S Composite
Age | Name — First name, Last name

3. Stored V/S Derived
DOB | Age — Can be derived from DOB automatically

4. Key V/S Non-Key
Key helps to uniquely identify each row ie Roll no is unique
Non Key can or cannot be unique ie name, P. No

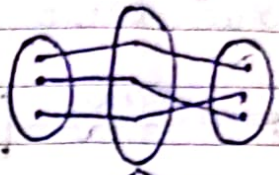
5 Required V/S Optional Attribute
Required means it is mandatory to fill it — Name
Optional means it is not compulsory — Phone No.

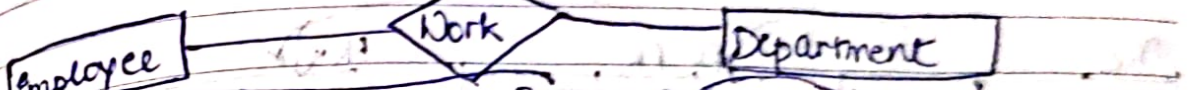
6 Complex
Composite + Multivalued



Degree of Relationship / Cardinality :-
one to one ; One to many ; Many to one ; many to many

One to One





PK

EID	EName	Age
E1	A	20
E2	B	25
E3	C	28

PK, PK

EID	DID
E1	D1
E3	D2
E2	D3

PK

DID	DName	Loc
D1	IT	Bangl
D2	HR	Del
D3	Product	Del

In the relation table, the primary key can be either of the foreign keys.

Reduced no: of tables :- 2 tables

EID	EName	Age	DID
E1	A	20	D1
E2	B	25	D3
E3	C	28	D2

DID	DName	Loc.
D1	IT	Bangl.
D2	HR	Delhi
D3	Product	Delhi

attributes of relation table are called descriptive attributes
We can merge the relationship table only with the table having primary key.

One / Many Relationship :-
We merge tables towards many side ie the primary key of relationship derived from which table

PK	Name	City
C_1	A	Barp.
C_2	B	Mumb.
C_3	C	Delh.
C_4	D	Delh.

FK	FK	
ID	ONO.	Date
C_1	O_1	
C_1	O_2	
C_2	O_3	
C_2	O_4	

PK		
ONo	Item	Cost
O_1	Bucket	100
O_2	Shoes	250
O_3	Shirt	150
O_4	Jeans	200

PK = ONO (unique + Not null)

Reduced table - 2.

ID	ONo.	Date	Item	Cost
C_1	O_1	-	Buck.	100
C_1	O_2	-	Shoe	250
C_2	O_3	-	Shirt	150
C_2	O_4	-	Jeans	200

ID	Name	City
C_1	A	Barp.
C_2	B	Mumb.
C_3	C	Delh.
C_4	D	Delh.

Many to many Relationship :-
In the relationship table or Referencing table, we cannot make any separate PK. We make composite key ie (RNo + CID).

No reduction of table is possible here as no table has a key (RNo + CID).

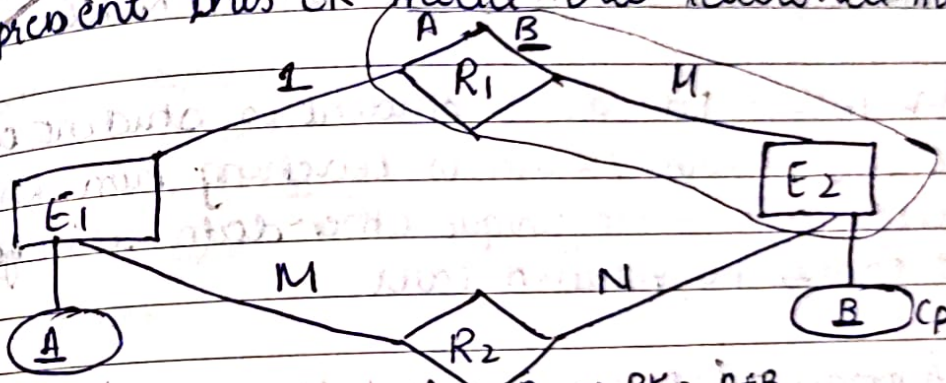
Many to many Relationship (M-N) :-

PK		
RNo	Name	Age
1	A	16
2	B	17
3	A	17
4	C	15

FK	FK
RNo.	C-id
1	C_1
2	C_2
1	C_1
2	C_2

PK		
CID	Name	Credit
C_1	Maths	4
C_2	Phy	4
C_3	Chem	4
C_4	Hindi	4

Ques :- What is the min. no of tables required to represent this ER model into relational model?



B/C of 1-M

E_1, E_2 E_1E_2
$= E_2$

E_1 E_2R_1

$\therefore T_1 = E_1$; $T_2 = R_1E_2$; $T_3 = R_2$.

Normalisation :-
It is a technique to remove or reduce redundancy or duplicacy from the table.
It is of 2 types :- Row level ; Column level

15 July Wednesday

Update Student Set SName = "Amit" where SID = 3

Important

Row level :-

SID	SName	Age
1	Ram	20
2	Vijay	25
1	Ram	20

Column level :-

PK SID	SName	CID	CName	FID	FName	Salary
1	Ram	C1	DBMS	F1	John	30K
2	Ravi	C2	Java	F2	Bob	40K
3	Nitin	C1	DBMS	F1	John	30K

We use primary key to remove row level duplicacy.

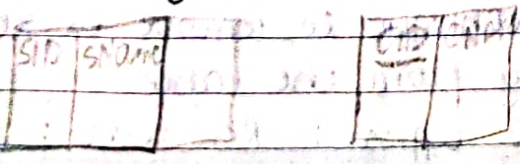
Problems related to column level :-

1. Insertion Anomaly :- There is a primary key already in column level and we want to add just course and ID. But we cannot do that as PK cannot be NULL.

2. Deletion Anomaly :- Lets say a student is student a unique course and unique teacher is teaching him. When we try to delete data these unique other data also gets deleted and cannot be retrieved later.

3. Updation Anomaly :- Lets say we want to update salary of F1. So due to column duplicacy it will check each data and then make change but change should have happened just once.

We can remove these anomalies from column level by dividing the table.



JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
Wednesday	1	8	15	22	29	
Thursday	2	9	16	23	30	
Friday	3	10	17	24	31	
Saturday	4	11	18	25		
Sunday	5	12	19	26		

Lossless / Lossy Join Decomposition

Whenever we normalise a table, we need to decompose it. There are 2 rules to follow:

1. Decomposition to be lossless
2. Dependency preserving decomposition

R

A	B	C
1	2	1
2	2	2
3	3	2

→ $R_1(AB)$

R_1

A	B
1	2
2	2
3	3

→ $R_2(BC)$

R_2

B	C
2	1
2	2
3	2

During decomposition, at least one column should be common. So that, if needed, during future queries, we can combine them using this common column.

Query: Find value of C where value of A is 1.

Select $R_2.C$ from R_2 Natural Join R_1 where $R_1.A = 1$.

Join means first we do cross product and then apply a condition

In cross product, multiply the first row with all rows and so on. Total rows = X*Y

17 July Friday 2015

14 Week 29th Day

Important

Natural Join = Cross product + check if B is same after cross

A	B	B	C	
1	2	2	1	✓
1	2	2	2	✓
~~1~~	~~2~~	~~3~~	~~2~~	
2	2	2	1	✓
2	2	2	2	✓
~~2~~	~~2~~	~~3~~	~~2~~	
~~3~~	~~3~~	~~2~~	~~1~~	
~~3~~	~~3~~	~~2~~	~~2~~	
3	3	3	2	✓

∴ R

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

extra tuples ie spurious tuples

But, in original table there were only 3 values now, there are 5 ∴ it is lossy decomposition as there is loss of consistency.

Solution :

Common attribute should be CK or Super Key of either R1 or R2 or both.

∴ A can be chosen.

A	B
1	2
2	2
3	3

A	C
1	1
2	2
3	2

∴ Recor Join

A	B	C
1	2	1
2	2	2
3	3	2

Natural Cross

A	B	A	C	
1	2	1	1	✓
~~1~~	~~2~~	~~2~~	~~2~~	
~~1~~	~~2~~	~~3~~	~~2~~	
~~2~~	~~2~~	~~1~~	~~1~~	
2	2	2	2	✓
2	2	3	2	
~~3~~	~~3~~	~~1~~	~~1~~	
~~3~~	~~3~~	~~2~~	~~2~~	
3	3	3	2	✓

∴ No loss of consistency

⇒ Lossless Decomposition

JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
Wednesday	1	8	15	22	29	
Thursday	2	9	16	23	30	
Friday	3	10	17	24	31	
Saturday	4	11	18	25		
Sunday	5	12	19	26		

2015 July 18 Saturday

3rd form always has dependency preserving decomposition But not BCNF

Conditions for Lossless Decomposition :-

i) $R_1 \cup R_2 \equiv R$

$AB \cup AC \equiv ABC \equiv ABC$

ii) $R_1 \cap R_2 \neq \phi$

$AB \cap AC = A \neq \phi$

iii) R_1 CK or R_2 CK or Both.

Eg :-
R(ABCD)
AB → CD, D → A
CK = {AB, BD}
$AB^+ = ABCD$
$CB^+ = CB$
$DB^+ = DBAC$
PA = {A, B, D}
NPA = {C}
It is not in BCNF so, decompose
AB
BCD {BD → C}
DA {D → A}
Reject AB → CD but there AB → AB
not preserved

4th Normal form

It should be in BCNF and there should be no multi-valued dependency.

Varun → 3 phone no.
Varun → 3 mails

V	M_1	E_1
V	M_1	E_2
⋮	⋮	⋮

5th Normal form

D = ABC

It should be in 4th Normal form.
There should be lossless decomposition.

Ques:- For the following FD, find correct minimal cover :-
{A → B, C → B, D → ABC, AC → D}.

1. ⇒ {A → B, C → B, D → A, [D → B] (redundant) D → C, AC → D}

Find redundant keys. :- $A^+ = A$; $C^+ = C$; $D^+ = DBC$; $D^+ = DABC$

2. {A → B, C → B, D → A, D → C, AC → D}

3 Single Attribute on left side. :-

AC → D A^+ doesn't have C
C^+ doesn't have A ∴ not removable

Ans :-
{A → B, C → B, D → AC, AC → D}

AUGUST 2015

Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

19 July Sunday

Important

2015 Week 29th

Ques :- R(ABCDEF), check the highest normal form?
FD: {AB → C, C → DE, E → F, F → A}.

Solution :-

Step 1 :- Find all CKs in relation

CK : {AB, FB, EB, CB}.

Step 2 :- All prime attributes

{A, B, C, E, F}.

Step 3 :- All non-prime attributes — DE is non prime attribute.

{D}.

Step 4 :- FD: {AB → C, C → DE, E → F, F → A}

C is proper subset of CK.

	AB → C	C → DE	E → F	F → A
BNCF	✓	X	X	X
3rd	✓	X	✓	✓
2nd	✓	X	✓	✓
1st	✓	✓	✓	✓

Highest is 1st normal form.

Ques :- How to find out the normal form of a relation?

R (ABCDEF)

CK = {AB, FB, EB, CB}

Prime attributes {A, B, C, E, F}

Non prime attributes {D}

FDs : {AB → C, C → D, C → E, E → F, F → A}.

Solution

Already in first normal form with highest redundancy.

Now, for second normal form :-

C → D is in partial dependency.

∴ We will decompose form 2nd normal form.

⇒

JULY 2015

Monday		6	13	20	27
Tuesday		7	14	21	28
Wednesday	1	8	15	22	29
Thursday	2	9	16	23	30
Friday	3	10	17	24	31
Saturday	4	11	18	25	
Sunday	5	12	19	26	

R(ABCDEF)

R_1 (ABCEF) — R_2 (C, D)

R_1 {AB → C, C → E, E → F, F → A} — {C → D}

CK = {AB, FB, EB, CB} — CK = {C}

We included C in R_2 so that we can have lossless join as a common attribute / CK is needed for that.

Now, for 3rd normal form :-

LHS = CK or SK OR RHS = PA

Conditions satisfied ∴ in normal form.

now for BCNF :-

R_1 {AB → C, C → E, E → F, F → A} R_2 {C → D}

Here, LHS should be CK which isn't true for all ∴

We decompose R_1

R_1 (ABCEF)

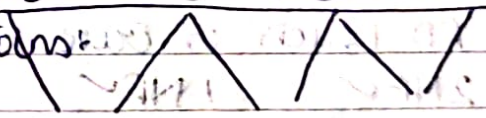
R_3 {AB → C}, R_4 {C → E}, R_5 {E → F}, R_6 {F → A} R_2 {C → D}

ABC, CE, EF, FA CD

At this point, we has min / 0 redundancy.

CK = AB C E F C.

For joins :-



C is in both E in both F in both

But if we want to join R_3 and R_5 its not possible directly as nothing common

Thus,

R_{34} = ABCE R_5 = EF

Now E common and is CK of R_5 ∴

R_{345} = ABCEF. (lossless join).

21 July 2015
Tuesday Week 30th Day 204

Important

Ques: Which of following is true?

(A) A relation is in 3rd NF then its always in BCNF

(B) A relation is in 2nd NF then its not in 1st NF

(C) A relation is in BCNF, then it is in 2NF.

(D) A relation is in 2NF, then it contains partial dependency.

Ques: Relation R has eight attributes ABCDEFGH.
FD: {CH → G, A → BC, B → CFH, E → A, F → EG}.
How many candidate keys in R?

$AD^+ = ADBCHFEG$

$ED^+ = EDABCFHG$

$FD^+ = ABCDEFGH$

$BD^+ = ABCDEFGH$

∴ 4 CKs.

Ques: Find the highest Normal form possible:-

1. Schema: Registration (roll no, courses)
Non-trivial FD: {roll no → courses}.
BCNF: LHS should be CK of FD which is true
∴ BCNF ✓ 3rd NF ✓ 2NF ✓ 1NF ✓
Since BCNF is valid ∴ all other forms valid too

2. Schema: Registration (roll no, course ID, email)
Non-Trivial FD: {roll no, Course ID → email, e-mail → roll no}
CK = {roll no + Course ID}
PA = {roll no, Course ID}
NPA = {email}
BCNF: Not true for email → roll no.

JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
Wednesday	1	8	15	22	29	
Thursday	2	9	16	23	30	
Friday	3	10	17	24	31	
Saturday	4	11	18	25		
Sunday	5	12	19			

LHS = CK OR RHS = PA

3NF ⊢ Here for email → roll no. condition satisfied

∴ BCNF ✗ 3NF ✓ 2NF ✓ 1NF ✓

3/ Schema 3: Reg.(roll no., course ID, marks, grade)

Non trivial FD: {roll no + course ID → marks, grade
marks → grade}.

~~BCNF ✗~~ CK ⊢ {roll no + C ID}

~~3NF ✗~~ PA ⊢ {roll no, Course ID}

NPA ⊢ {marks, grade}.

BCNF → ✗ 3NF ✗ 2NF ✓ 1NF ✓

2NF ⊢ Partial dependency ∴ LHS = proper subset of CK +
RHS = non PA

Here, marks → grade
False True = False
∴ not Par. depend.

∴ 2NF ✓

4/ Schema ⊢ Registration (roll no, course ID, credit)

non-trivial FD {roll no + course id → credit},
course ID → credit}.
True True

CK = {roll no + Course ID}

PA = {roll no, course ID}

NPA = {credit}

BCNF ✗ 3NF ✗ 2NF ✗ 1NF ✓

AUG SEP OCT NOV DEC

AUGUST 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

23 July Thursday 2015
26 Week 30th Day 2

Important

Equivalence of function dependency

$X = \{A \to B, B \to C\}$ $Y = \{A \to B, B \to C, A \to C\}$

Check if X covers Y $X \supseteq Y$ (1)

Y covers X $X \subseteq Y$ (2)

$\Rightarrow X \equiv Y$

(1) Take closure of Y from X

$A^+ \Rightarrow ABC$; $B^+ = BC$; $\Rightarrow X \supseteq Y$

valid (x) valid

(2) Take closure of X from Y

$A^+ = ABC$; $B^+ = BC$ $\Rightarrow Y \supseteq X$

valid valid

∴ Equivalent

Ques $X = \{AB \to CD, B \to C, C \to D\}$ $Y = \{AB \to C, AB \to D, C \to D\}$

(1) $X \supseteq Y$ (True)

$AB^+ = ABCD$; $C^+ = CD$

valid valid

(2) $Y \supseteq X$ (False)

$AB^+ = ABCD$; $B^+ = B$; $C^+ \Rightarrow CD$

valid not valid valid

∴ not equivalent

JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
Wednesday	1	8	15	22	29	
Thursday	2	9	16	23	30	
Friday	3	10	17	24	31	
Saturday	4	11	18	25		
Sunday	5	12	19	26		

Dependency Preserving Decomposition

R (FD)

R1 (FD1) R2 (FD2)

FD1 ∪ FD2 = FD+ , then preserved

Ques: Let R(ABCD) with FD
{A→B, B→C, C→D, D→B}.
R is decomposed into R1(AB), R2(BC), R3(BD)

Ans

R1(AB)	R2(BC)	R3(BD)
~~A→A~~	B→C ✓	~~B→B~~
A→B ✓	~~B→B~~	B→D ✓
~~B→A~~ ✗	C→B ✓	~~D→D~~
~~B→B~~	~~C→C~~	D→B ✓

(C→D→B; B→C→D)

We will not take trivial dependencies

Final list : A→B, B→C, C→B, B→D, D→B

Check R(FD) in regard with final list
{A→B, B→C, C→D, D→B}

For checking C→D, its not directly there in final list
∴ C+ = CBD ∴ D is here ⇒ C→D ✓
∴ It is preserved

Ques: R(ABCD) with FD {AB→CD, D→A}
Decomposed R1(AD) & R2(BCD)
First check after R1 ∪ R2 all elements are preserved ✓ (Yes)

R1(AD)	R2(BCD)
A→D ✗	B→CD ✗
D→A ✓	C→DB ✗
	D→BC ✗
	BC→D ✗
	BD→C ✓

CD→B ✗
~~BCD→BCD~~ ✓

25 July 2015
Saturday
Week 30th Day 208

Important

$BD^+ = \overrightarrow{BD}\overrightarrow{A}\overrightarrow{C}$; $CD^+ = CDA$

Final list : $\{D \to A, BD \to C\}$ → Because of this we cannot determine actual func. dependencies

8.00 $R(FD) = \{D \to A, AB \to CD\}$

∴ not preserved

9.00

10 Joins :

Relational algebra : Mathematical way of representing joins

11.00 Joins : When we have to combine 2 or more tables to find info which isn't possible individually

12.00 It can only be used if there is a common attribute b/o the 2 attributes.

1.00 Join = Cross product + select statement

1. Cross join
2. Natural join
3. Conditional join
4. 3.00 Equ. join
5. Self join
6. 4.00 Outer Join → Left, Right, full.

5.00

6 Natural Join :

When we have to equalise the values of common atti. in Tables, then we use natural join

Ques Find the emp. names who are working in a department.

Tables :

Monday		6	13	20	27
Tuesday		7	14	21	28
Wednesday	1	8	15	22	29
Thursday	2	9	16	23	30
Friday	3	10	17	24	31
Saturday	4	11	18	25	
Sunday	5	12	19	26	

Emp.

ENO	EName	Add.
1	Ram	Delhi
2	Varun	Chd
3	Ravi	Chd
4	Amit	Delhi

Dep.

Dep Name	ENO
D_1 HR	1
D_2 IT	2
D_3 MRKT	4

ENO	EName	DepNo	ENO
1	Ram	D_1	1
~~1~~	~~Ram~~	~~D_2~~	~~2~~
~~1~~	~~Ram~~	~~D_3~~	~~4~~
~~2~~	~~Varun~~	~~D_1~~	~~1~~
2	Varun	D_2	2
~~2~~	~~Varun~~	~~D_3~~	~~4~~
~~3~~	~~Ravi~~	~~D_1~~	~~1~~
~~3~~	~~Ravi~~	~~D_2~~	~~2~~
~~3~~	~~Ravi~~	~~D_3~~	~~4~~
~~4~~	~~Amit~~	~~D_1~~	~~1~~
~~4~~	~~Amit~~	~~D_2~~	~~2~~
~~4~~	~~Amit~~	~~D_3~~	~~4~~

Query :-
Select E.name from Emp, Dep where
Emp.ENo = Dep.ENO
OR
Select Ename from Emp Natural Join Dept
Select Ename from Emp, Dept
where Emp.ENo = Dept.ENO ;

Self Join
Table joined with itself
Ques: Find student ID who is enrolled in at least 2 courses

Study (Student, Course)

Sid	Cid	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Sid	Cid	Sid	Cid
~~S_1~~	~~C_1~~	~~S_1~~	~~C_1~~
~~S_1~~	~~C_1~~	~~S_2~~	~~C_2~~
~~S_1~~	~~C_1~~	S_1	C_2
~~S_2~~	~~C_2~~	~~S_1~~	~~C_1~~
~~S_2~~	~~C_2~~	~~S_2~~	~~C_2~~
~~S_2~~	~~C_2~~	~~S_1~~	~~C_2~~
~~S_1~~	~~C_2~~	~~S_1~~	~~C_1~~
~~S_1~~	~~C_2~~	~~S_2~~	~~C_2~~
~~S_1~~	~~C_2~~	~~S_1~~	~~C_2~~

Query :
Select T_1.SID from Study as T_1, Study as T_2
where T_1.SID = T_2.SID and
T_1.CID <> T_2.CID.
≠

Important

~~Equi~~ join : Equates any random attributes unlike natural

Ques: Find the emp. name who worked in dept having location same as their address?

Emp ← Dep

PK E.NO	EName	Addr.	DepNo	Location	ENo FK
1	Ram	Delhi	D_1	Delhi	1
2	Vaun	Chd	D_2	Pune	2
3	Ravi	Chd	D_3	Patna	3
4	Amit	Delhi			

Ans

1 Ram Delhi D_1 Delhi
~~1 Ram Delhi D_2 Pune 2~~
~~1 Ram Delhi D_3 Patna 3~~
~~2 Vaun Chd D_1 Delhi~~
~~2 Vaun Chd D_2 Pune 2~~
~~2 Vaun Chd D_3 Patna 3~~
~~3 Ravi Chd D_1 Delhi~~
~~3 Ravi Chd D_2 Pune~~
~~3 Ravi Chd D_3 Patna 3~~
~~4 Amit Delhi D_1 Delhi~~
~~4 Amit Delhi D_2 Pune 2~~
~~4 Amit Delhi D_3 Patna 3~~

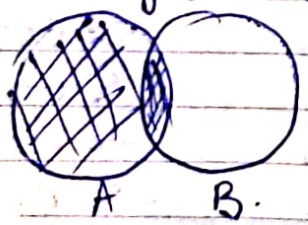
Query:

```
Select E.name from Employee, Department
where Emp.E_NO = Dept.E_No
and Emp.Addr. = Dept.Location
```

~~Left~~ Outer Join :-

It gives the matching rows and some which are in left table but not in right table

natural join + left table



Tables:

JULY 2015

Monday		6	13	20	27
Tuesday		7	14	21	28
Wednesday	1	8	15	22	29
Thursday	2	9	16	23	30
Friday	3	10	17	24	31
Saturday	4	11	18	25	
Sunday	5	12	19	26	

Emp

EmpNo	EName	DeptNo
E1	Varun	D1
E2	Amit	D2
E3	Ravi	D1
E4	Nitin	—

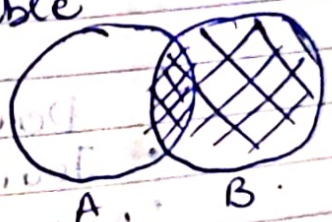
Dept

DeptNo	DName	Loc
D1	IT	Delh
D2	HR	Hyd.
D3	Fin.	Pune
D4	Test	Noida

EmpNO	Ename	Dname	Loc
E1	Varun	D1	Del
E2	Amit	D2	Hyd.
E3	Ravi	D1	Delh
E4	Nitin	-	-

Query :-
Select emp_no., ename, d_name, loc. from emp left outer join dept. on (emp.depno = dept.depno)

Right Outer Join
It gives matching rows and rows which are in right table but not in left table



Use Above tables only :-
Query :- Select emp.no, ename, dname, loc from emp right outer join dept on (emp.deptno = dept.deptno)
Sol :-

Emp No	Ename	Dname	Location	Dname
E1	Varun	D1	Delhi	IT
E2	Amit	D2	Hyd	HR
-	-	D3	Pune	Finance
-	-	D4	Noida	Test

29 July 2015
Wednesday 32. Week 31st Day 210
Important

Full outer Join :-
left outer join + Right outer join output
8.00 OR Union
ie common once and remaining too.
9.00

10.00 Relational Algebra
It is also called formal query lang. or procedural query
11.00 lang.
It is called proc. query lang as the user has to mention
12.00 2 things :- What to do? How to do?
We use relational algebra to access data.
1.00 It is base of SQL.

2.00 Operations (6) (2)

3.00 Basic Op. — Derived Op.
- Projection (π) — σ — → Join (⋈)
4.00 selection (σ) — × — → intersect (∩)
- cross product (X) — ∪ — (X ∩ Y) = X-(X-Y)
5.00 Union (∪) — − — → Division (/, ÷)
- Rename (ρ) — ρ
6.00 Set diff. (−)

Projection (π) :- Column
It fetches any data from table
It works only on distinct format ie no duplicate data will be shown.

Student

Roll No.	Name	Age
1	A	20
2	B	21
3	A	19

Query Ques'n Retrieve roll no.
π_{rollno} (Student)

1
2
3

Query Retrieve name
π_{name} (Student)

A
B

π (Student)
Not true.

π (of student) name, Rollno = 2

Selection (σ) :- Works on row (tuples)
Query :- Retrieve name of student whose Rollno = '2'.

$\pi_{name}(\sigma_{Rollno=2.}(Student))$

π name

Cross product (X) :- To join tables

R_1

A	B	C
1	2	3
2	1	4

tuples

R_2

C	D	E
3	4	5
2	1	2

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

Atleast one column should be common for joining

Columns in new table = 6 (3+3)
Rows (tuples) = $R_1 \times R_2$
= 2 x 2
= 4

31 Friday

2015 Week 31st Day 214

Important

Set Difference -

(A-B) = A but not B

= $A \cap B'$

$A - B \neq B - A$ ie not commut.

Conditions :

1. No. of colums should be same
2. Domain of each column must be same

Ques: Find name of person who is student but not employee.

$\Pi_{name}(Student)$

Roll no	Name
1	A
2	B
3	C

Student

Emp No	Name
7	E
1	A

Employee

Π_{name}

Answer

Roll no
B
C

Query : $\Pi_{name}(Student) - \Pi_{name}(Employee)$

Union :

S_1 (1,2,3) S_2 (3,4,5)

$S_1 \cup S_2 = 1, 2, 4, 5, 3$

Conditions :

1. No. of columns should be same.
2. Domain of each column must be same (numeric etc)
3. Column name after union will be of first table.

Use the above table :

Union

Roll no	Name
1	A
2	B
3	C
7	E

JULY 2015

Monday		6	13	20	27	
Tuesday		7	14	21	28	
Wednesday	1	8	15	22	29	
Thursday	2	9	16	23	30	
Friday	3	10	17	24	31	
Saturday	4	11	18	25		
Sunday	5	12	19	26		

Ques:- Find the roll no and name of people who are student, employee or both.

Query :-

$\Pi_{name, Rollno}(Student) \cup \Pi_{name, Emp\#}(Employee)$

Division :

Enrolled (E)

sid	cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

Course (C)

cid
C_1
C_2

Tells the actual data value ← helper table

E(Sid/Cid) / C(Cid)

A(X,Y)/B(Y) = it results x values for that there should be tuple (x,y) for every y value of relation.

Ques:- Retrieve sid of students who enrolled in every / all courses

Query :-

$\Pi_{sid}(Enrolled) - (\Pi_{sid}(((\Pi_{sid}(Enrolled)) \times \Pi_{cid}(course)) - (Enrolled)))$

① ← ②

$S_1 \times C_1$
$C_2 \Rightarrow$

S_2

S_3

Assume all students in all course

$S_1 C_1$ ~~$S_1 C_1$~~
$S_1 C_2$ ~~$S_1 C_2$~~ → $S_2 C_1 \Rightarrow S_3 C_1 \Rightarrow S_3$
$S_2 C_1$ ~~$S_2 C_1$~~
$S_2 C_2$ $S_1 C_2$
$S_3 C_1$ $S_3 C_2$
$S_3 C_2$ enrolled but no course

S_1 S_2
S_2 – S_3
S_3

$\Rightarrow$ S_1 Answer

E(Sid/Cid) / C(Cid)
↓ from where data is to be taken
what helps in getting data

02 August
Sunday

Important

2015
36 Week 31st Day 216

Tuple relational calculas
It is a non procedural query lang. → It tells what to do

8.00

9.00

Relational Calculas (RA)

Tuple relational (TRA) — focuses on rows or tuples

10.00

Domain relational — focuses on columns or attributes

11.00

Query in TRC :-

12.00 $\{t \mid P(t)\}$ → t = resulting tuple.

P(t) : predicate ie conditions to fetch t.

1.00

Operations :

Atomic functions : where we give only one condition to retrieve data.

3.00 OR (V), AND (∧), NOT (¬), ∃ (there exists), ∀ (for all)

4.00

$\forall \exists \neg \wedge \vee$

Unsafe expression

5.00 $\{S.name \mid \neg Suppliers(s)\}$

6.00 This will be stuck in infinite loop as this asks for all supplier names not in supplier table and we don't have that data.

Both TRA and RA have same expressive powers but TRA should not have unsafe expressions. But SQL has higher power.

August 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

2015
August
Monday 03
37

Examples:

Given: Supplier (SID, Sname, address)
Parts (PID, Pname, color)
Catalog (SID, PID)

Q1 Display Sname of suppliers.
{S.sname | Suppliers (s)}

Q2 Display Pname of parts whose color is red
{p. partname | Parts (p) ∧ p.color = 'Red'}

Q3 Display SID of suppliers whose SID is varun and add is Chd.
{S.SID | Suppliers (s) ∧ S.name = 'Varun' ∧ S.add = 'Chd'}

Q4 Display SID of suppliers who supplied some parts?
{C.Sid | Catalog (C)}

Q5 Display S.Name of suppliers who supplied some parts?
{S.Sname | Supplier (s) ∧ ∃C (Catalog (C)) ∧ S.Sid = C.Sid}

Q6 Display Sname of suppliers who supplied some Red color parts?
{S.Sname | Supplier (s) ∧ (∃c)(catalog(c)) (∃p)(parts(p)) ∧
S.Sid = C.Sid ∧
p.pid = C.pid) ∧
p.color = 'Red'}

General purpose lang - It can be used at multiple places.

04 August 2015
Tuesday

procedural lang - What to do, how to do

EF Codd — DBMS father

Important

38 Week 32nd Day 216

User → lang. → Database

SQL (Structured Query Language)

- Earlier is was called SEQUEL ie simple english query language.
- It is a domain specific language. It can only be used on relational data
- It is declarative lang. It tells what to do
- DDL, DML, DCL, TCL are diff commands to create, insert, fetch, control etc the data.

Types of the SQL commands

Data Definition lang (5)	Data manipul. lang (4)	Data Control lang (2)	Transaction Control lang (3)	Constraints (6)
create	Select	Grant	Commit	Primary key
alter	Insert	Revoke	Rollback	Foreign key
drop	Update		Savepoint	Check
Truncate	Delete			Unique
rename				Default
				not null

DDL commands

Deal with schema ie tables. How the table is formed.

DML commands

We can change the data using this. But only after we have defined the data

DCL -

Who can control the data.

AUGUST 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

Create Table :-

```
create table <table_name>
(
column1 name datatype,
column2 name datatype,
column3 name datatype,
);
desc table_name;
```

salary number (10, 0)
scale precision

```
create table emp
(
id int;
name varchar(20),
salary number(10)
);
desc emp;
```

Alter command : (6)

add columns ; remove columns; modify datatypes; modify datatype length; add constraint; remove constraint; rename remove column/table :

Example :-

- alter table Student add (address varchar(30));
- alter table employee drop column address;
- alter table employee modify id varchar(30);
- alter table employee rename column id to roll_no;
- alter table employee rename to emp;
- alter table employee add primary_key (roll_no);

Alter	Update
DDL command It changes the things related to columns ie column names or table.	DML command. It is used to change the data related to the table ie the data inside the table

06 August Thursday

Important

Rollback: bringing back deleted data from the log which is created after 2015 deleting.

Week 32nd Day

Delete	Drop	Truncate
DML command	DDL	DDL
It is used to delete entire row ie tuples specifically	It is used to delete entire table structure.	It deletes all rows in one go.
delete from Student; where ID='1'.	drop table student;	Truncate student;
Slower due to rollback.		faster
Rollback possible before commit.	No rollback	Rollback not possible
	View not exists	View exists
	Integrity constraints removed	not removed

Constraints in SQL

Restrictions before adding data in tables. We apply these conditions on the columns or attributes like roll-no, name.

1) Unique : diff data
2) Not Null : not empty
3) primary key : unique + not null
4) Check : check if condition met like check age >18.
5) Foreign key : referential integrity
6) default : if no data given then what to be filled

Eid	Ename	Dept	Salary
1	Ram	HR	10000
2	Amit	Mkt.	20000
3	Ravi	HR	30000
4	Nitin	Mkt	40000
5	Varun	IT	50000

AUGUST 2019

Monday	31	3	10	17	24	
Tuesday		4	11	18		
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

means comparing with single value 2 = 2
means comparing with multiple value 2 in (1, 2, 3, 4)

August
Friday 07
41

Ques: Write SQL Query to display max salary from emp table.
Ans: select Max(salary) from emp.

Ques: Write a SQL query to display employee name taking maximum salary.
Ans: select E-name from emp
where salary = (select Max(salary) from Emp)
} Eg. of nested salary.

Ques: Write query to display second highest salary from emp table.
Ans: Select Max(salary) from emp where
salary <> (select max(salary) from Emp;

Ques: Write query to display employee name taking second max salary.
Ans: select E-name from emp where
salary = select max(salary) from emp where
salary <> (select max(salary) from emp;

Ques: Write query to display all dept name along with no of employee working in that
Ans: select Dept-name, count(*) from emp
group by dept;

SEPTEMBER 2015

Monday	7	14	21	28	
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

SEP OCT NOV DEC

Eg group by :- We can use select with same attribute used with group by directly and aggregate fun

If we need to use where avoid group by, we use having

08 August Saturday 2015

Important

42 Week 32nd Day 2

Ques: Write a query to display all dept names where no. of emps are less than 2.

Sol: select dept from emp
group by dept having
count(*) < 2;

Ques: Write query to display all employee names of dept. having no. of emp. less than 2.

Sol: select Ename from emp
where dept In
(select dept from emp group by dept
having count(*) < 2);

Ques: Write a query to display highest salary dept wise and name of emp who is taking that salary.

Sol: select Ename from emp where salary in
(select max(salary) from emp group by dept);

In / Not In :-

Emp :-

Eid	Ename	Add.
1	Ravi	Chd.
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bang.
5	Ammy	Chd.

Project

Eid	Pid	Pname	Location
1	P1	IOT	Banglore
5	P2	Bigdata	Delhi
3	P3	Retail	Mumbai
4	P4	Android	Hyderabad

AUGUST 2015

Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

In 'in' inner query is executed once and then its values compared to outer table

Exist / Not exist used for correlated nested query.
One row of outer query is compared to all rows of inner query. Top to bottom approach
Rest are bottom to top approach.

Ques: Display detail of employee where address is either Delhi or Chd or Pune.
Select * from Emp where
Address in ('Delhi', 'Chd', 'Pune')

Ques: Find the name of Emp who are working on a project.
Sol: Select Ename from Emp where Eid in
(Selected (distinct Eid) from Project).

Ques: Find name of Emp who are not working on project.
Sol: Select Ename from Emp where Eid not in
(Select Eid from Project).

Ques: Find detail of emp who is working on at least one project.
Sol: Select * from emp where
Eid exists (Select Eid from Project where
Emp.Eid = Project.Eid)

Aggregate functions:
max, min, Count, Sum, Avg

Eid	Ename	Dept	Salary
1	Ram	HR	10000
2	Amit	MKT	20000
3	Ravi	HR	30000
4	Nitin	MKT	30000
5	Varun	IT	50000
6	Sandy	Testing	Null

10 August Monday 2015

Important

nth highest salary :-
select min(salary) from
(select distinct salary from emp order by salary desc) where rownum <= n;

Ques: Find total number of entries / tuples / rows in the table.

Sol: Select count(*) from Emp; ⇒ 6

Select count(salary) from Emp; ⇒ 5.
Select distinct count(salary) from Emp; ⇒ 4

Ques Find sum of all salaries.

Sol: Select sum(salary)) from Emp; → 140000
Select distinct(sum(salary)) from Emp; → 110000.

Ques Find avg. of all salaries.

Sol: Select Avg(salary) from Emp; 140000/5
Select distinct avg(salary) from Emp; 110000/4.

1. Basic subquery (scalar subquery.)

2. Correlated subquery (synchronized query).
It is a subquery that uses value from outer query. It uses top to down approach. exist connector is used.

Emp:-

Eid	Name	Address
1	A	Delhi
2	B	Pune
3	A	Chd
4	B	Delhi
5	C	Pune
6	D	Mumbai
7	E	Hyd

Dept:

Did	Dname	Eid
D1	HR	1
D2	IT	2
D3	HKT	3
D4	TEST	4

AUGUST 2015
Mo 3 10 17 24
Tu 4 11 18 25
We 5 12 19 26
Th 6 13 20 27
Fr 7 14 21 28
Sa 1 8 15 22 29
Su 2 9 16 23 30

Like command % = anything _/A/% = BACB
_ = fixed position anything Anything
--A/ 2 things before A and any no of char after it.

2015 August 11
Week 33rd Day 223rd Tuesday 45

Ques Find all employees detail who work in a dept
Sol: Select * from Emp where exists
(Select * from dept where dept.Eid = Emp.Eid)

Nested Subquery	Correlated Subquery	Joins
Bottoms up	Top to down	crossproduct + condition
n compare	mxn compare	

Ques: Display details of all emp who work in any dept.

Select * from emp where e-id in (select eid from dept)	Select * from emp where exists (select eid from dep where emp.eid = dep.eid)	Select * from Emp, dep where emp.eid = dept.eid
	lesser space but more time	lesser time but more space

Ques Find n^{th} highest salary using SQL
Sol: select id, salary from emp e_1
where N-1 = (select count (distinct salary)
from emp e_2
where e_2.salary > e_1.salary)

Q: You need to display last name of employees who have 'A' as second character in their names.
Sol: Select last_name from emp where last_name like '_A%'.

12 August Wednesday 2015

Week 33rd Day 224

Important

Q command to remove relation from SQL database.

Sol:- drop table <table name>;

8.00

Q. In the following schema R is R(a,b)

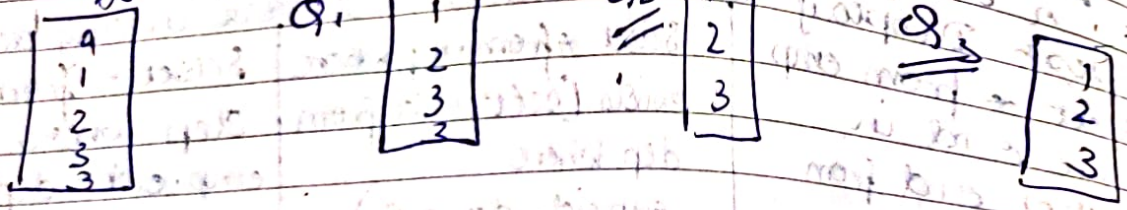
9.00 Q_1 select * from R

Q_2 (select * from R) intersect (select * from R)

10.00 Q_3 select distinct * from R.

Sol:- Q_2, Q_3 → same result

11.00



12.00

1.00

2.00 PL-SQL

It is procedural SQL ie it tells what to do and how to do.

3.00

In PL-SQL, query is called block or code. It is divided into 3 parts :- declaration (a int), executable code, exception handling

4.00

5.00

Declaration	a int
Executable code	begin
	end
Exception handling (error)	

6.00

AUGUST 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

PL SQL

Function

Procedure

Trigger

Cursor

2015 August 13
Week 33rd Day 225th Thursday 49

CPU doesn't directly work with hard disk as speed of Hard disk is very slow as compared to CPU

Transaction :
It is set of operations used to perform a logical unit of work.
A transaction generally represents change in database.

main transaction operations :- read, write, commit, rollback

A → B transaction

Steps :

In RAM
R(A) → 1000
A = A − 500
W(A) → 500
R(B) → 2000
B = B + 500
W(B) → 2500
Commit

Hard Disk Database
A = ~~1000~~ 500
B = ~~2000~~ 2500

Acid Properties of Transaction

Atomicity : Either all or none. Either all operations will run until commit or else even if one transaction fails it will rollback and start from 1st again ie Transaction cannot resume, it will restart.

Consistency : Before transaction starts and after the trans. ends, sum of money should be same.

Isolation : We try to convert parallel schedule into serial schedule to avoid interference of various transactions. This is because serial schedule is consistent always.

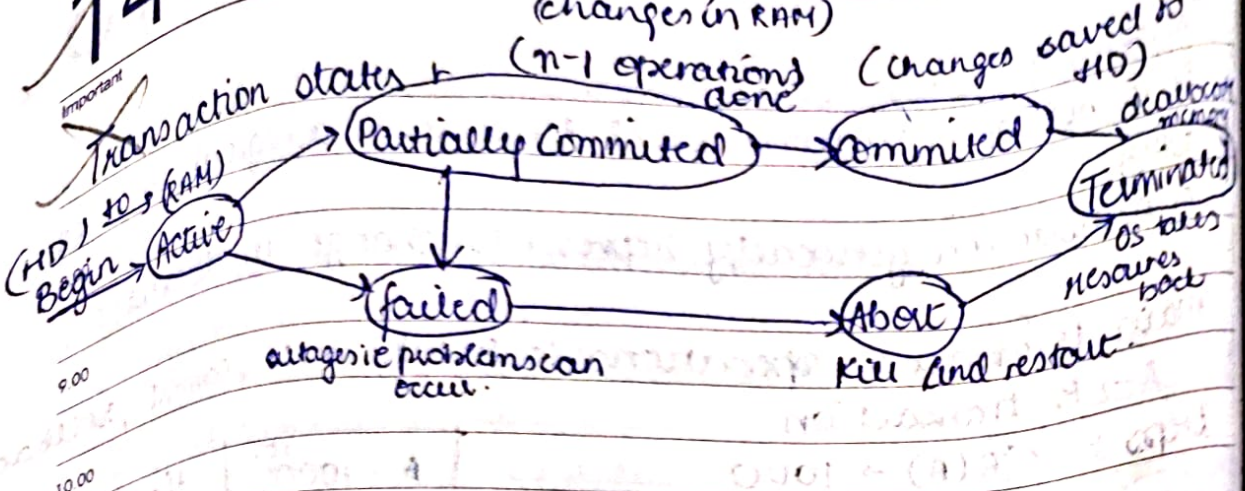
Durability : All the changes in database will be permanent. This is why we store it on hard disk.

14 August Friday

50 Week 33rd Day 226

Important

Transaction states :-



What is schedule?

It is chronological execution sequence of multiple transactions.

Eg $T_2 \longrightarrow T_1 \longrightarrow T_8 \longrightarrow T_3$ ---.

Serial schedule :-

One transaction will start and till the time it doesn't end, no new transaction will start.

Advantages :-

Consistency and secure

Disadvantages :-

Waiting time which degrades performance of the system

This decreases throughput ie no of transactions executed per unit time. Throughput ∝ performance.

Parallel schedule :-

Multiple transactions can come and start at same time by switching b/w diff transactions.

Advantages :-

Throughput increases as there is not waiting. Thus performance also increases.

Disadvantage :-

Inconsistency can occur.

2015 August 15 Saturday 51

RAW: read after write

Dirty read
phantom
unrepeatable

Lost update
incorrect summary

Types of concurrency:
concurrency means multiple transactions at same time ie parallel schedule.

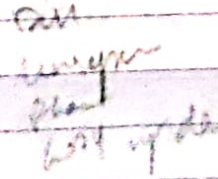
1. Dirty Read or uncommited Read or RAW (WR)
Here if a transaction starts on T_1 makes changes and these changes are read by T_2 and then finally T_2 commits by T_1 fails in future, then we will rollback the transaction. But T_2 gave wrong value answer.

2. Incorrect summary
This occurs when one transaction is going on and other transaction comes and performs its aggregate fns.

3. Lost Update: (WW)
The changes made by T_1 were updated by T_2 so the changes made by T_1 got lost.

4. Unrepeatable read: (RW)
When a transaction reads value 2 or more times and all times it is different.

5. Phantom read
When T_1 reads data, then T_2 reads it. After that T_1 deletes it and T_2 tries to delete it too.



16 August Sunday 2015

Important

Read-write conflict or unrepeatable read.

On same data:
- R R - no problem
- R W
- W R
- W W

	U_1	U_2
	T_1	T_2
	(2) R(A)	A=2
		R(A) (2)
		W(A) A=A-2 =0
10.00		Commit ⇒ A=0 ↓ database
	(0) R(A)	
11.00	W(A)	

RW or WR problem or WW commit

12.00 Here T_1 reads, then T_2 reads, writes and commits. The T_1 reads 0 and sees there is diff in the value of A but T_1 didn't perform any result. Thus roll back happens

2.00

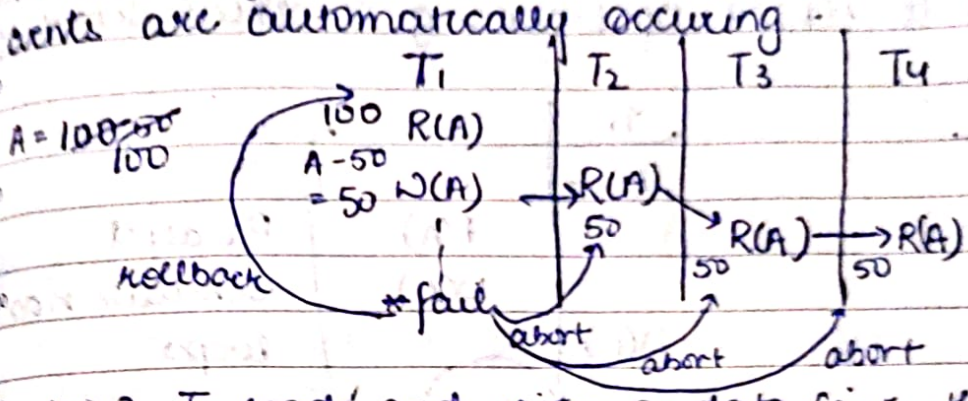
Irrecoverable Schedule

When changes made by some transaction cannot be recovered or regained, they are irrecoverable schedule

	T_1	T_2
5.00	10 R(A)	A=10 → register / HD.
	5 A=A-5	3
6.00	5 W(A)	10 (due to roll back
		R(A) 5
	rollback	A=A-2 3
		W(A) 3
		commit 3
	R(B)	
	fail	

Now 3 cannot be retrieved ie change made by T_2

Cascading : due to occurence of one event, multiple events are automatically occuring.



Here T1 reads and writes on data first, then T2, T3, T4 read the same data ie 50. But then T1 fails at the end and due to atomicity rollback. T2, T3, T4 are still reading 50 but 50 doesn't exist. So to avoid this read-write conflict, we abort T2, T3, T4 after T1 fails. This occurence of aborting T2, T3, T4 after T1 failure is called cascading. The schedule of T1, T2, T3, T4 is cascading schedule

Disadvantage :
CPU utilisation not proper as CPU cycle is wasted. Thus peformance reduced

Cascadeless schedule (W-W problem remains/~~R-R~~ WR solved)
We will not allow T2, T3, T4 to read A till T1 gets commited or fails or gets aborted.

Problems :
W W problems ie lost updation will still occur.
Cascadeless schedule says after write operation of T1 we cannot read T2, T3 & T4 before it gets finished ie commited

18 August 2015
Tuesday Week 34th Day 230

Schedule : Collection of transactions

$n!$ = no. of possibilities of serials

$T_1 \to T_2 \to T_3$; $T_1 \to T_3 \to T_2$; $T_2 \to T_1 \to T_3$,
$T_2 \to T_3 \to T_1$; $T_3 \to T_1 \to T_2$; $T_3 \to T_2 \to T_1$

Important

$3 \Rightarrow 6$

Serialisability :
Does a schedule capable of becoming a serial schedule

S

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)

Serial

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A)	

Parallel schedule has loops

$T_1 \to T_2$

$T_1 \leftrightarrow T_2$

To convert $T_1 \leftrightarrow T_2$ to serial schedule we can convert to $T_1 \to T_2$ or $T_1 \leftarrow T_2$.

There are 2 ways of checking if we can make a serialisable schedule :
Conflict, View

Conflict Equivalent Schedule :

If we are given a schedule, we try to find its conflict equivalent or if 2 schedules are given, we try to find if they are conflict equivalent or not.

R(A)	R(A)	R(A) W(A)
R(B)	R(A)	W(A) R(A)
R(B)	W(A)	W(A) W(A)
W(B)	R(A)	
W(A)	W(B)	

Non Conflict

Conflict

2015

August 19 Wednesday 55

Check if $S \equiv S'$

S	
T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

S'	
T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

S	
T_1	T_2
R(A)	
W(A)	
	R(A)
R(B)	
	W(A)

$\Rightarrow$

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

III, S'

Adjacent non conflict pairs can be swapped

Adjacent no conflict pairs not swapped.

$S \rightarrow S'$ (exists) then S' is serialisable ie serial schedule

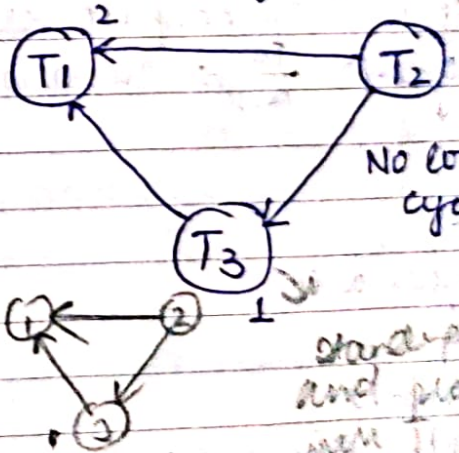
Conflict Serialisability

Check if given schedule is conflict serialisable

T_1	T_2	T_3
R(x)		
		R(y)
		R(x) for this W(x) in T_1
	R(y)	
	R(z)	
		W(y) check for W(y), R(y) conflicts
	W(z)	
R(z)		
W(x)		
W(z)		

Check conflict pairs in other transactions & draw edges

Precedence graph



Check for loop or cycle in precedence graph

No loop $\Rightarrow$ conflict serialisable $\Rightarrow$ consistent

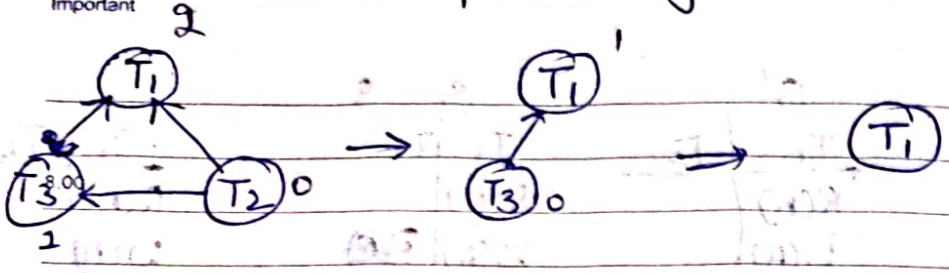
Check incagree for the actual serial possible

Now;

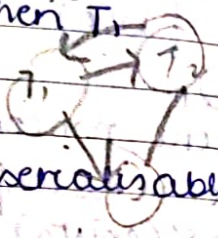
20 August
Thursday
Important

5t 2015
Week 34th Day 232

Keep removing lowest indegree



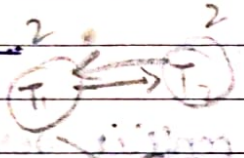
Ans: $T_2 \rightarrow T_3 \rightarrow T_1$ This is sequence for conflict serialisable
ie first finish operations of T_2 then T_3 then T_1



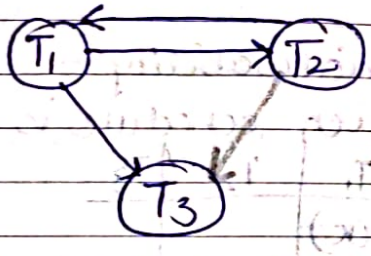
Ques: Check whether schedule is conflict serialisable or not?

S

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)



Precedence graph



Now here, there is a loop from T_1 to T_2 and back.
∴ it is not conflict serialisable.

No loop → Yes exists → no answer, use view.
None ↓

C S
↓
Serialisable
↓
Consistent

When there is a loop, schedule is not conflict serialisable but we check for serialisable for view serialisable. Here we check for the answer with values

AUGUST 2015

Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

2015

Week 34th Day 233rd

August 21 Friday

U(A) unlock

A = 100

T_1	T_2	T_3
100 R(A)		
-40 = 20 W(A)	-40 = 60 W(A)	
		-20 = 0 W(A)

View ≡ Equivalent

T_1	T_2	T_3
100 R(A)		
-40 = 60 W(A)		
	-40 = 20 W(A)	
		-20 = 0 W(A)

Give same ans. ∴ view serialisable

Concurrency Control protocols.

Used to achieve serialisability and recoverability.

They make schedules serialisable and recoverable using locking protocols (simple, 2PL, rigourous 2PL, strict 2PL, time stamp protocols). This helps to achieve consistency.

Shared-Exclusive locking (simple protocol)

→ Shared lock (S) :- if transaction locked data item in shared mode, then allowed to read only.

→ Exclusive lock (X) :- if transaction locked data item in exclusive mode then allowed to read and write both.

T	T
S(A)	X(A)
R(A)	R(A)
U(A)	W(A)
	U(A)

grant \ request	S	X
S	✓ R-R	X R-W
X	X W-R	X W-W

Comparability table

Problems in S/X locking

1) may not be sufficient to produce only serialisable schedule
2) may not be free from irrecoverability.
3) may not be free from deadlock.
4) may not be free from starvation

22 August Saturday 2015

58 Week 34th Day 234

Important

Problem (1)

T_1	T_2
X(A)	
R(A)	
W(A)	S(A) - only when X(A) unlocks
U(A)	R(A)
X(B)	U(A)
R(B)	
W(B)	
U(B)	

It doesn't became serializable as it is $T_1 \rightarrow T_2 \rightarrow T_1$

Problem (2)

T_1	T_2
X(A)	
R(A)	
W(A)	S(A)
U(A)	R(A)
	U(A)
*fail	Commit

not aborted ∴ dirty read by T_2

Problem (3)

Two transactions waiting for resources and both are waiting in infinite loop.

T_1	T_2
Granted X(A)	
	X(B) Granted
Wait X(B)	
	X(A) Wait

Problem (4) Finite time waiting :- Starvation

T_1	T_2	T_3	T_4
X(A)	S(A)	S(A)	S(A)
	U(A)	U(A)	U(A)

now gets resource

2) Phase locking (2PL) [Modification of S/X lock].
Growing phase → locks are acquired and no locks are released.
Shrinking phase → locks are released and no locks are acquired.
This helps to achieve serialisebility.
Any transaction following 2PL will always be serialisable

Schedule S

T_1	T_2
↓ Lock S(A)	
	Lock S(A) ↓
Lock X(B)	
shrink starts → Unlock (A)	
Lp *	
	Lock X(B) ← shrink
Unlock (B)	* Lp
	Unlock (A) ← shrinking
	Unlock (B)

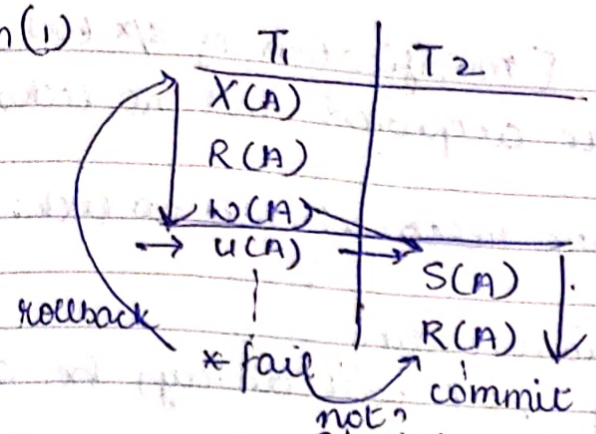
Growing

Lock pt → where transaction starts the first unlock.
It helps to find schedule of serialisability.
∴ Serial $T_1 \rightarrow T_2$

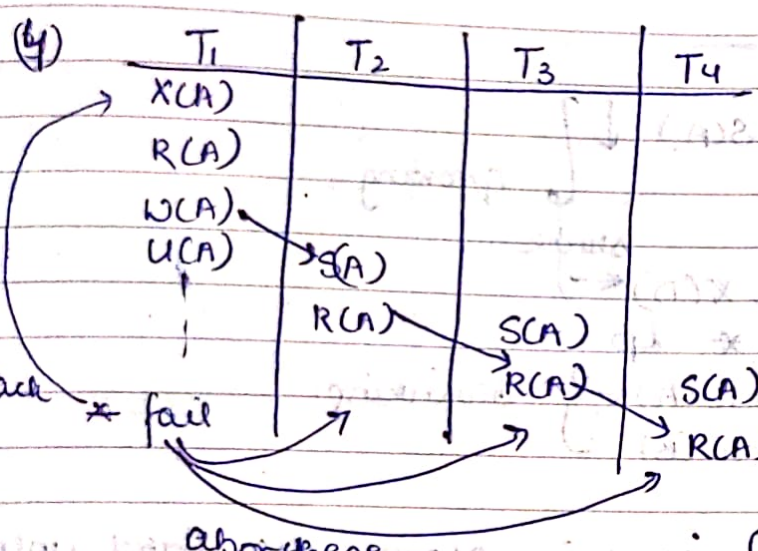
Advantages →
Always ensures serialisability.

Disadvantages →
May not be free from cascaded irrecoverability.
Not free from deadlocks
Not free from starvation
Not free from cascading rollback

Problem (1)



Problem (4)



5.00 Problem (2), (3) same as S/X lock.

6.00 Strict 2PL and Rigorous 2PL are used to solve the problem of irrecoverability and cascading rollback.

Conservation 2PL removes all the problems as it says give all resources ie locks to T_1 first so in this way T_2, T_3 do not have to wait so no deadlock. But we do not know from start which locks will be needed.

AUGUST 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	

2015
Week 35th Day 237th

Tuesday 25/8

Strict 2PL: (Cascadeless, Strict Recoverable always)
It should satisfy basic 2PL and all exclusive locks should be held until commit / abort and later release

Rigorous 2PL: (Same as above)
It should satisfy basic 2PL and all shared, exclusive locks should be held until commit / abort

Timestamp ordering protocol: transaction entering first will finish first
Unique value assigned to every transaction
It tells the order (when they enter into system)
It ensures that the transaction entering first should only finish first.
Read_TS(RTS) = last (latest) transaction no. which has performed read successfully.
Write_TS(WTS) = last (latest) transaction no. which performed write successfully.

10 T_1	20 T_2	30 T_3
R(A)		
	R(A)	
		R(A)

RTS(A) = 30

10 T_1	20 T_2	30 T_3
W(A)		
		W(A)
	W(A)	

WTS(A) = 20

Rules
1) Transaction T_i issues a Read(A) operation — Timestamps of Transaction
   a) if WTS(A) > TS(T_i), rollback T_i.
   b) otherwise execute R(A) operation
      Set RTS(A) = max{RTS(A), TS(T_i)}
2) Transaction T_i issues write(A) operation
   a) if RTS(A) > TS(T_i) then rollback T_i
   b) if WTS(A) > TS(T_i) then rollback T_i
   c) otherwise execute write(A) operation
      Set WTS(A) = TS(T_i)

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

SEP OCT NOV DEC

26 August Wednesday

2015

62 Week 35th Day 238

Important

T_1	T_2
W(A)	R(A)

roll back

T_1	T_2
R(A)	W(A)

RB

T_1	T_2
W(A)	W(A)

RB

Conflict occurs when younger reads first [R(B) before W(A)] as then elder younger gets commited and younger works on wrong data.

Ques:

100 T_1	200 T_2	300 young T_3
10.00 R(A)		
	R(B)	
11.00 W(C)		R(B)
	rollback	
12.00 R(C)	W(B)	
1.00		W(A)

	A	B	C
RTS	~~0~~ 100	~~0~~ 200 300	~~0~~ 100
WTS	~~0~~ 300	0	~~0~~ 100

Comparisons of above schedule acc. to rules:

0 > 100

300 > 200

0 > 100

400 0 > 100

0 > 300

500 100 > 100

[300 > 200 True] ∴ roll back of T_2

600 100 > 300

0 > 300

$T_1 \rightarrow T_3 \rightarrow T_2$

↓

later after restarting.

AUGUST 2015

Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

2015

Thursday 21

Why indexing is used? It helps minimize disk access required for query processing
CPU gets the query but data is actually stored in the memory. We permanently store data in Hard disk. But for CPU to interact with hard disk RAM is used. OS divides hard disk into blocks/pages and inserts data in it.

Data can be stored in sorted or unsorted way.
First whole block comes to RAM, if we get searched data, it is hit otherwise miss. This is I/O cost. More the blocks called more is the I/O cost. Thus our motive is to reduce the no. of blocks called. Now if we use index I/O cost will be reduced.

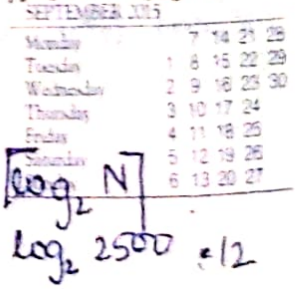
Question :-
Consider a hard disk in which block size = 1000 bytes, each record is of size = 250 bytes. If total no of records are 10000 and the data entered in hard disk without any order (unordered). What is avg time complexity to search a record from HD?

Sol :-
No. of Records we can put in every block = $\frac{1000}{250}$
= 4 records.

No. of blocks required = $\frac{10000}{4}$ = 2500 blocks.

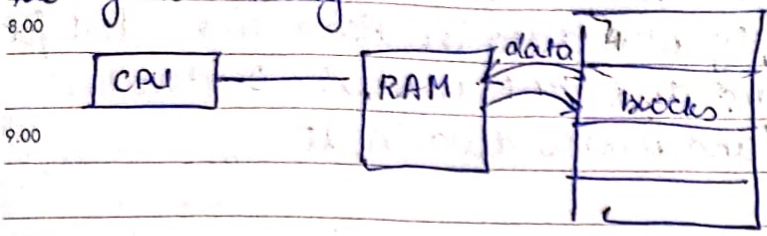
Best case time complexity = 1 based on I/O cost i.e. no of blocks called
Worst case time complexity = 2500
Avg. case time complexity = $N/2$ = 1250.

If data is sorted, we use binary search then $\lceil \log_2 N \rceil$
= $\log_2 2500$ = 12



28 August 2015
Friday
Important
64 Week 35th Day 240

But we can reduce it to cost more than binary search using indexing.



Index : same as no block

Key, pointer

Key tells the data to be searched
Pointer tells the page number

Number of records in one block of index are categorised:
Dense, Sparse

Dense :- For each entry in HD, index table has same no. of entries. Index record for every search key.

Sparse :- For each block of HD, there will be 1 pointer each which points to the first entry ie anchor records ie no of entries in index = no. of blocks in HD. It can be used only in ordered records.

Ques :- consider a hard disk in which block size = 1000 byte each record is of size = 250 bytes. If total no. of records are 10000 and the data entered in hard disk with and without order (2 cases). What is avg. time complexity to search a record from index table if index table entry = 20 B. (10 B pointer, 10 B key)

Block size in Disk = Block size in IT index

Sol:-

Each block of index table = $\frac{1000}{20} = 50$ records

no of records in each block = 1000/250 = 4
∴ no. of blocks needed for 10000 records = 10000/4 = 2500.

Dense indexing (uses no. of records)
e.g. No. of blocks of index = 10000/50
= 200
∴ search time = $\lceil \log_2 200 \rceil + 1$
= 8 + 1 = 9.

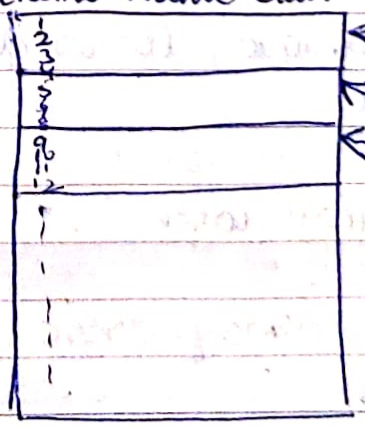
Sparse indexing (Uses no. of blocks)
No. of blocks needed = 2500/50
= 50
∴ Search time = $\lceil \log_2 50 \rceil + 1$
= 6 + 1 = 7

Types of indexes :-
Key & Uniqueness
1 primary ~~key~~ indep · at most 1 in table } Ordered file
2 Clustered
3 Secondary Unordered file

	Key	Non key
Ordered file	Primary (Sparse) 1/3	2/3 Clustered (Sparse)
Unordered file	Secondary	Secondary

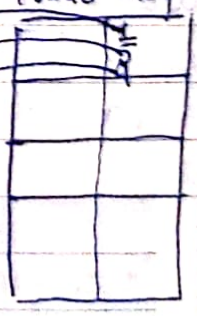
Primary Indep
- Data should be ordered and unique.
- no. of entries in index table = No of blocks in HD.
- Search time = $(\log_2 n) + 1$, n = no. of blocks in IT → to search any block in HD of primary
- We can use sparse

unique, ordered [Rollno Name class

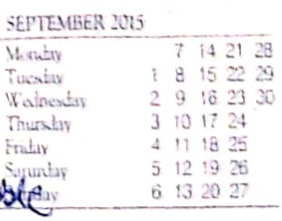


Hard disk

Pointer Key (Search key = Roll no)



Index Table



30 August Sunday 2015

67 Week 35th Day 242nd

Important

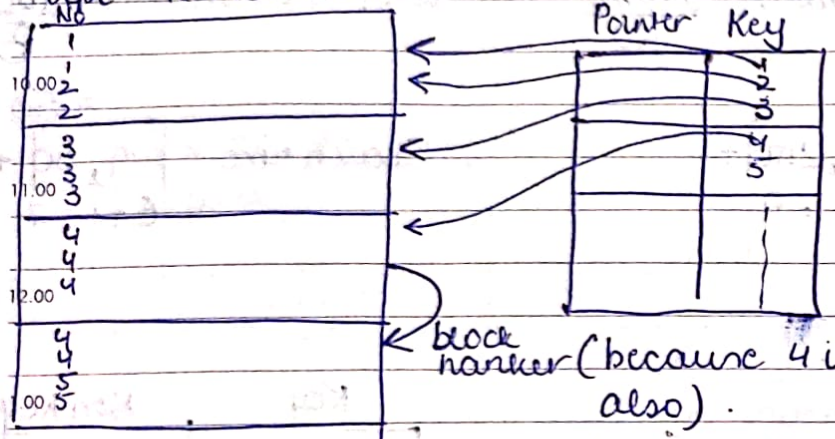
Clustered index :

Ordered and non unique data / non key

8.00 Searching time = $\log_2 N + 1 + 1$ - because there can be block number.

No. of entries in IT = no. of unique data keys.

9.00 Dept Name PNO.



We can use sparse over here

2.00

Secondary index.

Unordered data and unique/non unique non key / CK

4.00 In index we will take the key data sorted only. The data is ~~sparse~~ dense always.

5.00 No. of records in index table = no. of records in hard disk.

Search time = $\log_2 N + 1$ } unordered and unique

6.00

In case of unordered and non unique, it is dense and sparse both, but we consider dense.

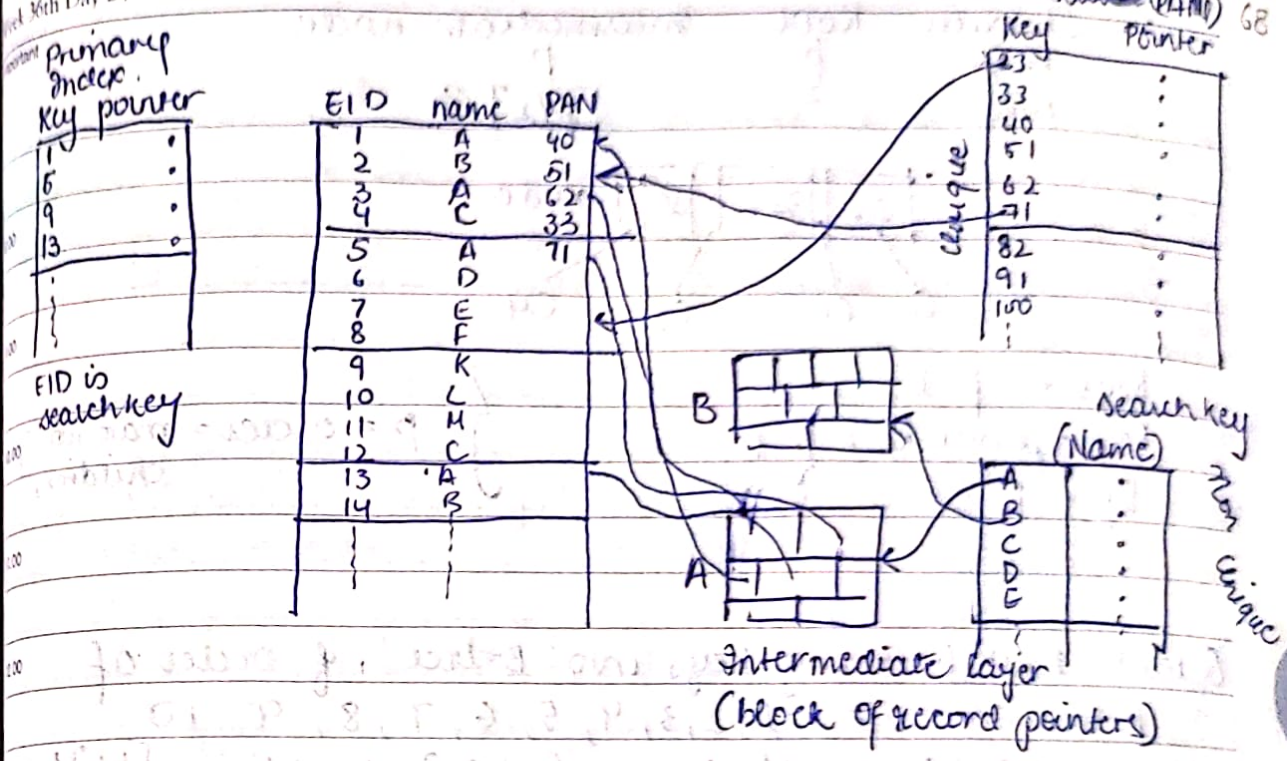
Search time = $\log_2 N + 1 + 1$

for intermediate layer

Secondary index is always dense

AUGUST 2015

Monday	31	3	10	17	24	
Tuesday		4	11	18	25	
Wednesday		5	12	19	26	
Thursday		6	13	20	27	
Friday		7	14	21	28	
Saturday	1	8	15	22	29	
Sunday	2	9	16	23	30	



B Tree → (Dynamic multilevel index)

When index size increases, we convert it to multilevel ie the index of index and so on are made. When the indexes increases, insertion and deletion get tough. Lets say we insert some data in secondary memory, then lots of changes have to be made in all indexes. To better this thing, we use Balanced tree.

Nomenclature :-

1) Block pointer / tree pointer :- There can be multiple values in one node and they further point to their child. We use tree pointer for them.

2) Key :- searching criteria and corresponding to them we have record pointer which points to actual data in SM.

Key = Record pointers (no.)

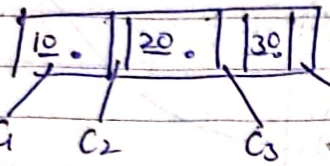
Children = Block pointer (no. of).

Order = p = max no. of children

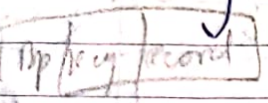
01 September
Tuesday
2015
69 Week 36th Day 244

Important

Children Root Intermediate node
Max. p p
Min. 2 $\lceil p/2 \rceil$

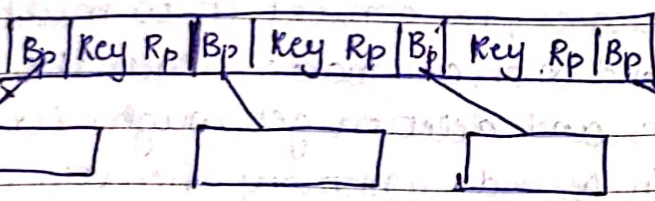


Keys = p-1
Record pointer Rp = p-1
Bp = p
} p = order = max no. of children

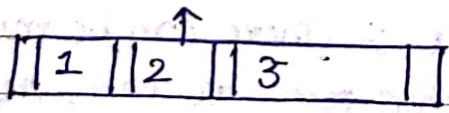


Ques: Insert following keys into B-tree, if order of B-tree = 4 :: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

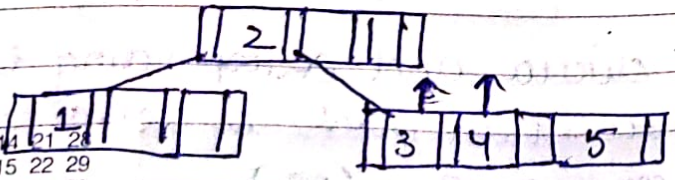
Here, max keys = p-1 = 4-1 = 3 ; min = $\lceil p/2 \rceil - 1 = 2-1 = 1$
Rp = p-1 = 3
Bp = p = 4



max children = max Bp = 4
max keys = max Rp = 3
Like binary search tree left < root < right



4 overflow $\lceil \frac{n}{2} \rceil = 2$

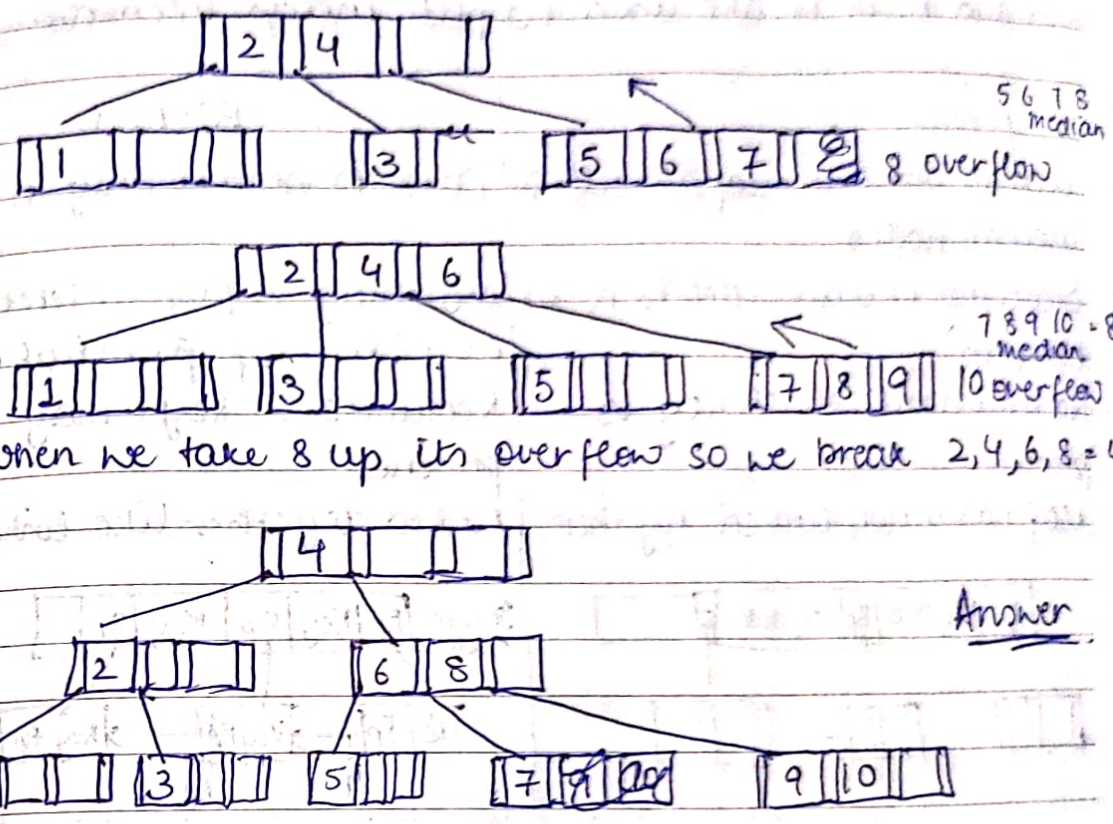


6 overflow $\lceil \frac{n}{2} \rceil = 3$ (4th median)

SEPTEMBER 2015

Monday	7	14	21	28	
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

2015 September
Wednesday 02



When we take 8 up its overflow so we break 2,4,6,8 = 4

Answer

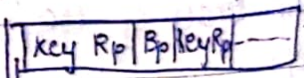
Ques: Consider a B-tree with key size = 10 bytes, block size = 512 bytes, data pointer is of size 8 bytes, block pointer = 5 bytes. find order of B-tree?

Sol: no of block pointers = n

Total size of Bp = n × Bp.

Key = no of Bp - 1 = (n-1) key

Rp = no of keys - 1 = (n-1) Rp



Block

	Root	Internal
max	23	23
min	2	$\frac{23}{2} = 12$

Key = order - 1

Rp = order - 1

$\Rightarrow n \times Bp + (n-1)key + (n-1)Rp \leq$ block size

$(n \times 5) + (n-1)10 + (n-1)8 \leq 512$

$5n + 18n - 18 \leq 512$

$23n - 18 \leq 512$

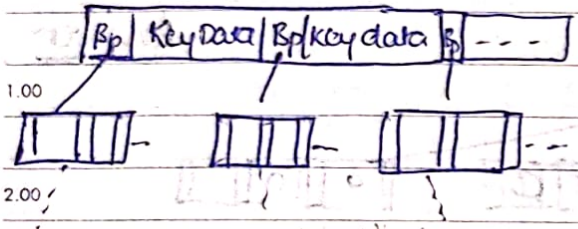
$23n \leq 530$

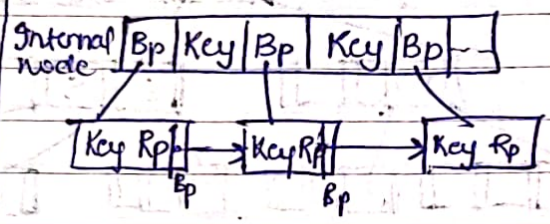
$n \leq \frac{530}{23}$, 23.04 $\therefore$ $n = 23$

No of block pointer = n = order of B-tree = 23 Ans

B and B+ trees are used to put index records.

B tree	B+ tree
1. Data is stored in leaf as well as internal nodes	Data is stored only in leaf nodes
2. Searching is slower, deletion is complex	Searching is faster, deletion is easy (discrete from leaf node)
3. No redundant search key is present	Redundant keys may be present
4. Leaf nodes not linked together	Linked together like linked list.

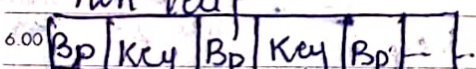




$n Bp + n Key \leq 512$

Ques:
Consider a B+ tree with key size = 10 bytes, block size = 512 bytes, data pointer = 8 bytes and block pointer = 5 bytes. What is the order of leaf and non leaf node.

Non leaf



$n \times Bp + (n-1) key \leq Block\ size$

$n \times 5 + (n-1)\ 10 \leq 512$

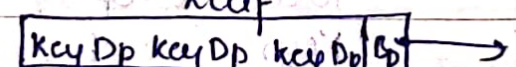
$15n \leq 522$

$n \leq \frac{522}{15} = 34.8$

$n = 34$.

∴ max children = 34. Order of non leaf

Leaf



$x(key) + x(Dp) + Bp \leq Block\ size$

$x(10+8) + 5 \leq 512$

$18x \leq 507$

$x \leq \frac{507}{18}, 28.2$

$x = 28$.

Order of leaf node = 28

In relational mode, no. of attributes is called degree.

Log is file where what is done by transaction is stored.

Immediate Database Modification (log Based Recovery).
As soon as written, ie W(A), the database value also changes. We won't wait for transaction to be commited

T_1	In deferred only new value stored	
A = 100	$\langle T_1, Start \rangle$	$\langle T_1, Start \rangle$
B = 200	$\langle T_1, A, 100, 200 \rangle$ (old, new)	$\langle T_1, A, 1000, 2000 \rangle$
R(A)	$\langle T_1, B, 200, 400 \rangle$	$\langle T_1, B, 5000, 6000 \rangle$
A = A+100	$\langle T_1, commit \rangle$	$\langle T_1, commit \rangle$
W(A)		$\langle T_2, Start \rangle$
R(B)	System 1 log	$\langle T_2, C, 700, 800 \rangle$
B = B+200		
W(B)		
Commit		

Redo

Undo

If system 1 fails, then recovery manager will first check if T_1 was started and commited both. If yes, then it will redo. ie it will save new values in database. If already saved, then it will overwrite them with same values.
But, if not commited at end, and failure occurs then it only starts and not commits, so it will undo ie retrieve old values from the old, new.
Thus, it is also called Redo/ undo strategy.

Q: Let R(a, b, c) and S(d, e, f) be 2 relations. 'd' is a foreign key of S that refer to primary key of R. Consider four operations on 'R' and 'S'.
(i) insert into R (ii) insert into S
(iii) delete from R (iv) delete from S
Which of these can violate referential integrity
Ans: (ii), (iii)

current data

05 September 2015
Saturday
Important

73 Week 36th Day 248

Ques:- The view of total database content is called conceptual view.

8.00

Ques:- The view of partial database content is called external view.

9.00

Ques:- In relational mode, cardinality is no. of Tuples

11.00

Ques:- Referential key constraint is used to maintain consistency among tuples in 2 relations.

12.00

Ques:- What do datawarehouse support?

1.00 Data Warehouse is the place where we keep data from that is integrated from diff. sources. Later, this data is 2.00 analysed. This is used to retain customers using cookies.

∴ Data Warehouse supports OLAP (Online Analytical 3 Processing.

Ques:- Hadoop is framework that works with variety of related tools. Common group includes ___, ___, ___.

Ans:- 5.00 It is a Big data tool ie data is in peta bytes. We use hadoop to process this unstructured data.

6 Mapreduce:- Reduces data into small batches and peforms multiprocessing.

Hive:- It is used to write SQL commands.

Hbase:- It is used for data storing.

SEPTEMBER 2015

Monday		7	14	21	28	
Tuesday	1	8	15	22	29	
Wednesday	2	9	16	23	30	
Thursday	3	10	17	24		
Friday	4	11	18	25		
Saturday	5	12	19	26		
Sunday	6	13	20	27		

Ques: All of the following accurately describe hadoop, except opensource (under apache), it is not real time, it does batch processing, java based, distributed computed approach/ grid computing.

Ques: 5 Vs of Big Data:-
Volume : Amount of data
Velocity : speed with which data is increasing
Variety : Types like structured, un.
Value : Value of Database of Big Data.
Veracity : Trustworthyness

Log Based Recovery : If system fails, how can we recover it. Using log means log is small sized file which stores actions performed by transactions.

Deferred Database modification (No Undo / Redo)
The changes after write are made only in RAM, they are reflected in DB after it is commited.

A = 100 B = 200 DB	T1		
	R(A)	<T1, start>	(T1, Start)
	A = A + 100	<T1, A, 200 new>	<T1, A, 200>
	W(A)	<T1, B, 400>	<T1, B, 400>
	R(B)	<T1, Commit>	<T1, Commit>
	B = B + 200		<T2, Start>
	W(B)		<T2, C, 500>
	Commit		

If Start + Commit ⇒ Redo
If Start + no commit ⇒ No redo + rollback ie old values
Redo means new values will be stored to DB.

OCTOBER 2015
Monday 5 12 19 26
Tuesday 6 13 20 27
Wednesday 7 14 21 28
Thursday 8 15 22 29
Friday 9
Saturday 3 10 17 24 31
Sunday 4 11 18 25

OCT NOV DEC

07 September 2015
Monday 75 Week 37th Day 250
Important

Like Command in SQL :-
Helps to search data.

Emp	
ID	Name
1	Varun
2	Arun
3	Kavya
4	Amit
5	Ranjeet
6	Ajeet

8.00
1. Find employee details whose name starts with A
Sol :- Select * from employee where
9.00 name like 'A%'.

10.00
2. Find emp detail whose name ending with 'n'.
Sol :- Select * from emp. where
11.00 name like '%n'.

12.00
3. Whose name contains 'ee'.
Sol :- Select * from emp where
1.00 name like '%ee%'.

2.00
4. Whose name contains a in 2nd place.
Sol :- Select * from emp where name like '_a%'.
3.00

5. Whose name contains a in second place and name should
4.00 contain 5 characters.
Sol :- Select * from emp where name like '_a___'.
5.00

6.00
PL SQL programs-2
Programe 1 :- Find sum of 2 numbers.

```
declare
a int;
b int
c int;
begin
a:=&a;
b:=&b;
```

& helps to take value from user.

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

c := a+b;
dbms_output.put_line('sum of a and b = '||c);
end;

Program 2: Greatest of 2 numbers

```
declare
a int;
b int;
begin
a := &a;
b := &b;
if (a>b)
then
dbms_output.put_line ('a is greater').
else
dbms_output.put_line ('b is greater').
end if;
end;
```

Program 3: For loop

set server output on is used to activate output

```
declare
a number(2);   -- size
begin
for a in 0.. 10   .. means default 1 step each
loop
dbms_output.put_line (a);
end loop;
end;
```

09 September
Wednesday

2015
77 Week 37th Day 252

Important

Program 4; While loop:

```
declare
a int;
b int;
begin
a := 0;
b := &b) = 7
while a < b
loop
a := a+1;
dbms_output.put_line ('value of A' || a);
end loop;
end;
```

Output

0
1
2
3
4
5
6
7

Output
0
1
2
3
4
5
6

Single Row Fcns → Row ⇒ Tuple
It means fcn is applicable on single row and gives single output.

Multi Row fcn
It is applicate on multiple rows and gives a single output

RAID :- Redundant array of independent disks
We need performance, security

RAID-0	RAID-1	RAID 1+0	Raid 3	Raid 4	Raid 5
data striping A → b1 disk1 b2 disk2 Best performance	mirroring If data on one disk fails we prefer security	mixed raid most important	data broken into blocks and parity also stored for data recovery Only 1 disk fail		Distributed parity :- no over usage of 1 disk Raid 6 2 parity fcn

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

2015
September 10 Thursday

SQL functions

- Single Row function
  - Number fxns
  - Char. fxns
    - Case manipul.
      - lower
      - upper
      - initcap
    - Char. manipul.
      - concat
      - substr
      - length
      - instr
      - LPAD/RPAD
      - Trim
      - Replace
  - Date fxn
    - Months_between
    - Add_Months
    - NEXT_DAY
    - LAST_DAY
    - ROUND
    - TRUNC
  - General fxns
    - NVL
    - NVL2
    - NULLIF
    - COALESCE
    - CASE
    - DECODE
  - Data Type Conversion
    - TO_CHAR
    - TO_NUMBER
    - TO_DATE
- Multi Row function
  - Aggregate
    - sum
    - max
    - min
    - count
    - avg

Character fxn

Case fxn manipulation
- lower
- upper
- initcap

Character manipulation
- Concat ('Varun', 'Singla')
- Substr ('Varun', 2, 4) ⇒ aru
- Instr ('Varun', 'u') ⇒ 4
- length ('Varun') ⇒ 5
- LPAD ('Varun', 10, '*') ⇒ *****Varun
- RPAD
- Trim ('V' from 'Varun') ⇒ arun
- Replace ('Varun', 'V', 'T') ⇒ Tarun

11 September Friday 2015

Important

79 Week 37th Day 254

(Materialised view) Takes space if data is on remote server and we want to keep a copy on our local server of full data or parts of it

create table emp
(
8.00 id int,
name varchar); (id, name)
9.00 insert into emp values (1, 'Vipasha Rana');
select concat (id, name) from emp.
10.00 select substr(name, 2, 5) from emp. ⇒ Gave smashers ate s

11.00

View in Database ? Subset of data from one or more tables

12.00 It is virtual table.
It is result set of stored query
1.00 Read-only Vs updated views
Materialised view

Create view V1
as select id
from students.

2.00 Changes of base table will be reflected in the view table whereas changes in view will not be visible on base table
3.00 In case of ~~updated~~ read only view but allowed in updated view.

4.00 We cannot apply DDL commands but DML commands only on view.
5.00 We can bring data of more than one table on view.
We can show lesser columns or rows with view.
6.00

Advantages of View:
- To restrict data access
- To make complex queries easy.
- To provide data independence
- To present different views of same data.

SEPTEMBER
Monday 7 14 21 28
Tuesday 1 8 15 22 29
Wednesday 2 9 16 23 30
Thursday 3 10 17 24
Friday 4 11 18 25
Saturday 5 12 19 26
Sunday 6 13 20 27

Saturday 12/20

1. Create table from existing table
create table emp2 as select * from emp emp;

2. fetch all emp with salary b/w 10000 to 50000
select * from emp where esal between 10000 and 50000

3. WHERE VS HAVING

where is used to filter records based on specified condition	having is used to filter records from groups based on conditions
It cannot have aggregate	It can operate on agg fns
implemented on rows	On columns
Used before group by and	Used after groups are made
after having	

4. find lowest salary of emp in each dept.
select dept, min(salary) as lowest_salary from emp
group by dept;

5. Write query to fetch unique value from column in table.
select distinct city from emp.

6. Write SQL query to fetch unique dept and print their length.
select distinct dept, length(dept) as length_dept from emp

7. What is use of DATEDIFF in SQL.
Returns no. of days b/w 2 date, datetime or timestamp
Eg: select datediff('2021-04-10', '2021-03-30');

13 September
Sunday

Important

81

Q Write an SQL query to display depts that have more than 2 employees.

8.00 select dept, count(emp_id) from emp. group by dept having count(emp_id) > 2

9.00

Q Display details of employees for all dept except marketing

10.00 select * from emp where dept != / <> 'Marketing';

Q Find employee with 3rd highest salary.

select * from (select * from employee order by salary

12.00 desc limit 3) as T order by salary limit 1;

Q Print all alternate records in table.

with CTE as

2.00 {

select *, ROW_NUMBER() OVER

3.00 (order by emp_id) as rn from emp

}

4.00 select * from CTE where rn % 2 == 1;

OR

5.00 select * from emp where emp_id % 2 = 1;

Q Write query to fetch all duplicate rows in table.

select e_id, name, age, count(*) as dup_count from dup_employees group by e_id, name, age having count(e_id) > 1 and count(name) > and count(age) >

SEPTEMBER 2015

Monday		7	14	21	28	
Tuesday	1	8	15	22	29	
Wednesday	2	9	16	23	30	
Thursday	3	10	17	24		
Friday	4	11	18	25		
Saturday	5	12	19	26		
Sunday	6	13	20	27		

Q Display employees with exactly 2 A's in their name.

select * from employee where length(name) - length(replace(upper(name), 'A', '')) = 2;

2015
258th Day 259th

1	Garry	11000	3	0
2	Girish	10000	September 14	
3	Smith	19000	Monday	
4	Lalaan	25000		
5	Thrisha	12000	3	

82

8. Given a string, how will you extract four characters starting from second position.

```
select substr ("Michael",2,4);
```

9. How does self join work?

It joins a table to itself. The table must contain column(x) that acts as primary key and different column(y) that stores values that can be matched up with values in column x.

10. Return name of manager for each employee in same row

```
select e.Emp_id, e.Emp_name, e.managerId, m.Emp_name as
manager_Name from emp as e Join emp as m on
e.manager_id = m.Emp_id;
```

11. Write a query to fetch list of employees with same salary.

```
select distinct e.emp_Id, e.emp_name, e.salary from
employees e, employees e1 where e.salary = e1.salary
and e.emp_Id != e1.empId.
```

12. Write a query to print one row twice in results from table

```
select emp_name, dept from employees e where e.Dept = 'HR'
union all select emp_name, dept from employees e1 where
e1.dept = 'HR'.
```

13. Use emp table and write a query to add 10 where age = 0, 20 when age = 1, else print number itself.

```
select age,
case when age = 0 then age+10
     when age = 1 then age+20
     else age
end as case_age
from emp.
```

OCTOBER 2015

Monday		5	12	19	26	
Tuesday		6	13	20	27	
Wednesday		7	14	21	28	
Thursday	1	8	15	22	29	
Friday	2	9	16	23	30	
Saturday	3	10	17	24	31	
Sunday	4	11	18	25		

OCT NOV DEC

15 September 2015
Tuesday
Important
Week 38th Day 258

Horsey

select sum(case when num 83

Q. Given a table, write query to find sum of all positives & negatives separately.

select sum

8.00 select sum(case when num > 0 then num else 0 end) as sum_of_positive,
9.00 sum(case when num < 0 then num else 0 end) as sum_of_negative
from employee;
10.00

Q. Diff b/w primary key and foreign key

11.00

Primary	Foreign
uniquely + not null	can be null
12.00 Only one in table	can be many
uniquely identifies a row in table	field in table that is primary key in other table.

Q. 2.

Primary key	Union
Only one in table	can be more than one
3.00 Can't accept null	can accept null.
create clustered index	create non clustered index.

4.00

Q. What is check constraint in SQL?

5.00 It is used to limit values that can be inserted into column.
Eg: To allow to insert row where city = Mumbai & age > 10.
6.00 create table dummy (c_id int PRIMARY KEY, city VARCHAR(
check (city = "Mumbai"), age int check (age > 10);

Q. Given 2 tables A and B, write query to fetch values in table B that are not in A.

select id from B left join A using (id)
where A.id IS NULL.

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Q) There a 2 tables order and customers. Find customers with no orders.

```
select customerid from customers
where customerid not in (select customerid from orders);
```

Q) using the Orders table, find month end orders. (ie last day)

```
select employeeid, orderid, orderdate from orders
where orderdate = EOMONTH(orderdate).
order by Employeeid, Orderid
```

Q) Find top 5 countries with highest freight charges in year 1997

```
select top 5 shipcountry, avg(freight) as average_freight
from orders where year(OrderDate) = 1997
groupby Shipcountry
orderby average-freight desc;
```

Q) Display total products in each category (tables: categories, products)

```
select Categoryname, count(*) as total_products
from products as p inner join categories as c
on p.category_id = c.category_id
groupby categoryname
order by count(*) desc;
```

Q) Find list of late orders for all employees (Table: Employee orders)

```
Select e.employeeid, firstname, count(*) as late_orders
from orders as o inner join employees as e
On e.Employeeid = o.EmployeeId
where RequiredDate <= Shipped date
groupby e.Employeeid, e.firstname
order by late_orders desc;
```

OCTOBER 2015

Monday		5	12	19	26	
Tuesday		6	13	20	27	
Wednesday		7	14	21	28	
Thursday	1	8	15	22	29	
Friday	2	9	16	23	30	
Saturday	3	10	17	24	31	
Sunday	4	11	18	25		

OCT NOV DEC

17 September
Thursday

Important

Joins → Equi → Inner (Select rows based on condition specified)
→ Outer (Select rows based on join ⊕ rows not satisfying condition)
→ non Equi

2045
Week 38th Day 260

PL SQL
In order have benefits of fors, loops etc, PLSQL is extension of SQL

1. Procedure
It is named block of statement which may/maynot return value

Syntax
Create [or replace] Procedure procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
[IS | AS]
BEGIN
<procedure_body>
END procedure_name;

2. Cursor
It is temporary area created in main memory when SQL statement is executed. It contains info on select statement and rows of data accessed by it.
It can hold more than one row but process only one row at time
Set of rows it holds = active set
a) Implicit cursor : They get created when we execute DML queries like select, insert etc
Eg :- %Rowcount, %Found
b) Explicit :- They must be created when you execute select statement that returns more than one row in PL/SQL procedure

Or often
These cursors only differ in way they are accessed

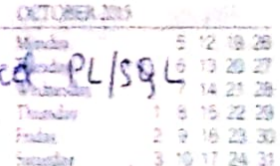
SEPTEMBER 2045
Monday 7 14 21 28
Tuesday 1 8 15 22 29
Wednesday 2 9 16 23 30
Thursday 3 10 17 24
Friday 4 11 18 25
Saturday 5 12 19 26
Sunday 6 13 20 27

2015
Day 261st
September
Friday 18

5. Function
It is same as procedure except that it always returns a value.
Syntax
Create [or replace] fun fun_name
[(param_name [IN | OUT | IN OUT] type [, ...])]
Return return_datatype {IS | AS}
Begin
<fun_body>
End fun_name;

6. Trigger
They are stored routines which are automatically executed when some event occurs.
They can be written for DML, DDL.
Syntax:
Create [or replace] Trigger trigger_name
Before / After Insert / Delete / Update
On table_name [For each row]
Declare
... variable declarations
Begin
... Trigger code
Exception when
... exception handling
End;

7. Package
It is schema object that groups logically related PL/SQL types, items & subprograms.
Parts: package specification & package body or definition



OCT
NOV
DEC

19 September Saturday 2015

Week 38th Day 262

Important

select product Id, max(quantity) group by

Exception :
an error condition during program execution.
System defined and user defined

Sales Data

1 Orders Table : (details)
Order Id . Order Date Property Id Product Id Quan

2. Property :-
Property Id Property City Property State

3. Product :-
Product Id Product Name Product Category Price

Basic Questions :

1 Find maximum quantity sold in a transaction
select max(quantity) from orderdetails;

2. Find count of transactions with max quantity
select max(quantity), count(*) from orderdetails;

3 Find unique products in all transactions
select distinct ProductId from orderdetails;

4. Find max quantities sold for unique products in all transactio
select distinct productId, Quantity from orderdetails where quantity = 3 order by ProductId asc, Quantity desc;

5. Find unique properties
select distinct propertyId from order details;

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

2015

Sunday 20

28th Day 263rd

select max(

select product category, select product max(

Intermediate ques :-

1. Find product category that has max products
select productcategory, count(*) as Counting from products
groupby productcategory order by Count desc limit 1;

2. Find state where most stores are present
select propertystate, count(*) as Counting from property
groupby propertystate orderby Count desc;

3. Find top 5 product Ids that did maximum sales in terms of quantity
select productId, sum(Quantity) as Total from orderdetails
groupby productId orderby 2 desc limit 5;

Intermediate II ques :-

1. Find top 5 products names that did maximum sales in term of quantity.
select p.productname, sum(o.Quantity) as Total from orderdetails as o left join products as p on o.productId = p.productId groupby p.productname orderby 2 desc limit 5;

2. Find top 5 products that did maximum sales.
select p.productname, sum(o.quantity * p.price) as sales from orderdetails as o left join products as p on o.productId = p.productId
groupby p.productname
orderby 2 desc limit 5;

3 Find top 5 cities that did maximum sales.

```
select pi.propertycity, sum(o.quantity * p.price) as sales
from orderdetails as o left join products as p on
o.productId = p.productId left join property as pi on
o.propertyId = pi.propertyid
groupby pi.propertycity
orderby sales desc limit 5;
```

Advanced ques

1. Find top 5 products in each city.

```
select pi.propertycity, p.productname, sum(o.quantity x p.price)
as sales from orderdetails as o leftjoin products as p on
o.productId = p.productId left join property as pi on
o.propertyId = pi.propertyId
where pi.propertycity = 'Arlington'
groupby pi.propertycity, p.productname
orderby sales desc limit 5;
```

Questions:

1. Write a query to add multiple ~~cols~~ rows in table

```
insert into users values
(null, 'Akash', 32, 32), (null, 'Aditya', 30, 28);
```

2. Write a query to add/delete multiple cols in table

```
alter table users
add email varchar(255) NOT NULL UNIQUE
```

```
alter table users
drop column email, drop column passcode.
```

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

2015
39th Day 265th
Tuesday 22

limit 1,1
start from 2nd movie and print 1

Write a query to find 2nd highest value in table
select * from movies order by score desc limit 1,1.

Find max value without using order by.
select name, budget from movies where budget =
(select max(budget) from movies);
} Independent query

Write a query to change the datatype of column.
alter table movies
modify column budget bigint;

Find manager of all employees.
select e1.fname, e2.fname from employees e1 JOIN employees e2
on e1.manager_id = e2.id;

Use of wild cards.
select * from movies where name like 'A____';
select * from movies where name like '%man';

If else based question.
select name, (sibsp + parch) as 'size',
case
when (sibsp + parch) <= 2 then 'small',
when (sibsp + parch) > 2 and (sibsp+parch) < 4 then 'medium'
else 'large'
end as 'type'
from titanic

OCTOBER 2015

Monday		5	12	19	26	
Tuesday		6	13	20	27	
Wednesday		7	14	21	28	
Thursday	1	8	15	22	29	
Friday	2	9	16	23	30	
Saturday	3	10	17	24	31	
Sunday	4	11	18	25		

23 September 2015
Wednesday
Week 39th Day 266
Important

9. Find category-wise top value.
select genre, name, score from movies m1 where score =
8.00 (select max(score) from movies m2 where m2.genre = m1.genre);

10.00 Find / delete all duplicate values
delete from contacts where id NOT IN (select min(id) from
10.00 contacts group by first_name, last_name, email);

11.00 Joining tables on users, memberships, groups. (User, Grp name)
select name, age, group name from users u JOIN membership m
12.00 on u.user_id = m.user_id JOIN groups g on m.grp_id = g.grp_id;

1.00 Theory questions:-
1. SQL VS NOSQL databases
2.00 SQL is used on structured data ie tabular data.
Eg:- MySQL, Oracle, Postgre
3.00 When we have unstructured data like insta post then NOSQL.
Eg:- MongoDB
4.00

2. Tables required for many to many relationships
5.00 Three tables needed. Eg:- Students & Courses

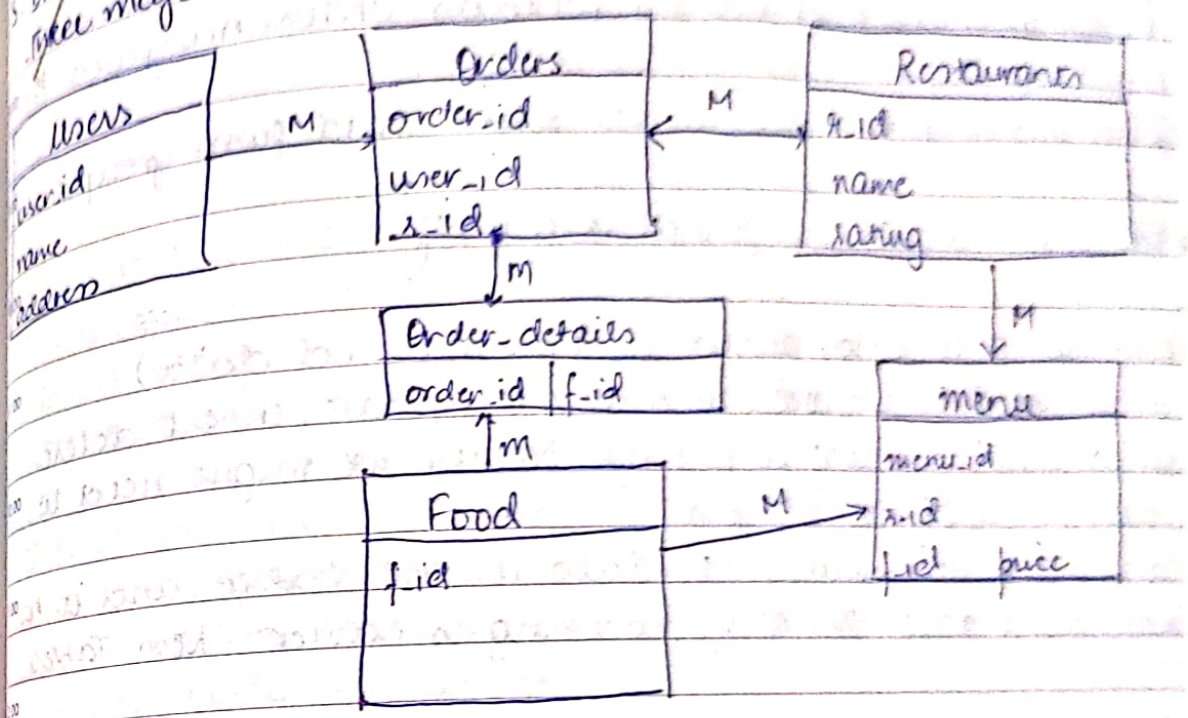
3. 6.00 Tables required for one to many relationships
Two tables needed Eg:- customer & addresses,
student & branches

4. Tables required for one to one relationship.
One table needed Eg:- Customer & Aadhar.

SEPTEMBER 2015

Monday		7	14	21	28	
Tuesday	1	8	15	22	29	
Wednesday	2	9	16	23	30	
Thursday	3	10	17	24		
Friday	4	11	18	25		
Saturday	5	12	19	26		
Sunday	6	13	20	27		

1. Draw schema design for swiggy.
Three major tables : users, restaurants, food items



2. Types of Joins → Union vs Union All
1. Cross join : It is cartesian product and it tells all possibilities
select * from users u CROSS JOIN groups g.

2. Inner join : It represents the common of 2 or more tables
select * from users u INNER JOIN membership m on
u.user_id = m.userid .

3. left join : It represents common of tables + all from left table
select * from users u LEFT JOIN membership m on u.user = m.user.

4. Right join :

5. Outer/full join : left join ∪ right join
select * from user u FULL OUTER JOIN member m on u.user = m.user

25 September Friday 2015
Week 39th Day 268th
Important

MYSQL doesn't support full outer join. So we need to do left and right join separately and then UNION them.
UNION gives distinct values whereas UNION ALL gives all duplicates too.
Select userid from users UNION select grp_id from groups.

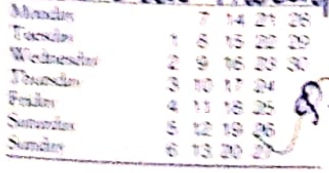
Self join :- The table is joined with itself.

7. OLAP vs OLTP (Normalised vs Denormalised data)
In normalised data write operations ie update, insert, delete are easy but reading it is tough because we might need to access many tables via joins.
Denormalisation means all data in one table and it has redundant data. The query processing is easier. Less Tables.

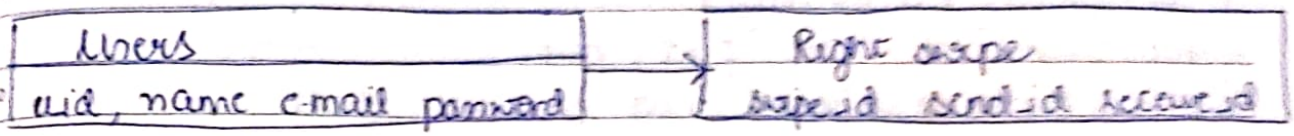
OLTP :- The databases are called online transaction processing on which websites work. It has normalised data.
OLAP :- Eg in datawarehouse. It is used by analysis team to read data. This has denormalised data. Disk wastage.

ETL : Extract Transform Load
For analysis, the data is extracted from different tables of database (OLTP) and transformed into redundant denormalised table later, it loaded on datawarehouse to analyse it.

Case Study :- Tinder
There are multiple users. Users can send any number of requests to other users.
1. Convert business scenario to schema for this one



Here, it seems like many to many but there is only one entity ie users. So instead of 3, we will make 2 tables only



Q2. Write a query to print name and details of all users to whom a particular user has sent request.

```
Select * from rightswipe r JOIN users u on r.receive_id=u.uid
where r.send_id = (select u.uid from users where name = 'Virat');
```

Q3. Write query to print name of user who has sent 2nd most number of requests. Also print count

```
Select u.name, count(send_id) from rightswipe r join users u on
r.send_id = u.uid group by send_id order by count(send_id)
desc limit 1,1.
```

Q Find all matches for a particular user.

```
select u1.name, u2.name from rightswipe r1 JOIN
rightswipe r2 on r1.send_id = r2.receive_id JOIN
users u1 on r1.send_id = u1.uid JOIN
users u2 on r1.receive_id = u2.uid WHERE
r1.receive_id = r2.send_id AND (r1.send_id = 1 OR r1.receive_id = 1);
```

Case Study – Swiggy

Given :

six data tables

27 September Sunday 2015

Week 39th Day 270th

Important

users :- user_id, name, email, password
restaurants :- r_id, r_name, cusine, rating
food :- f_id, f_name, type
menu :- menu_id, r_id, f_id, price
orders :- order_id, user_id, r_id, amount, date
order_details :- id, order_id, f_id

10.00

Q1 Find customers who have never ordered

11.00 select name from users where user_id not in (select user_id from orders);

12.00

Q2. Average price / dish.

1.00 select f_name, avg(price) as 'Avg Price' from menu m Join food f on m.f_id = f.f_id
2.00 group by m.f_id;

Q3. Find top restaurants in terms of number of orders in a given month (June)

4.00 select r.r_name, count(*) as 'count'
from orders o
5.00 Join restaurants r on o.r_id = r.r_id
where MONTHNAME(date) LIKE 'June'
6.00 Group by o.r_id order by count(*) desc limit 1

Q4. Restaurants with monthly sale > x ~~=~~ (x = 500)

select r.r_name, sum(amount) as 'revenue' from orders o join restaurants r on o.r_id = r.r_id where MONTHNAME(date) like 'June' group by o.r_id having revenue > 500

SEPTEMBER

Monday		7	14	21	28	
Tuesday	1	8	15	22	29	
Wednesday	2	9	16	23	30	
Thursday	3	10	17	24		
Friday	4	11	18	25		
Saturday	5	12	19	26		
Sunday	6	13	20	27		

Q5 Show all orders with order details for a particular customer in particular date range.

```
select o.order_id, r.r_name, f.f_name
from orders o JOIN restaurants r on r.r_id = o.r_id JOIN
order_details od on o.order_id = od.order_id JOIN
food f on f.f_id = od.f_id
where user_id = (select user_id from users where name like
'Ankit') and date > '2022-06-10' and date < '2022-07-10');
```

Q6 Find restaurants with max repeated customers.

```
select r.r_name, count(*) as 'loyal customers'
from (select r.id, user_id, count(*) as 'visit'
      from orders group by r_id, user_id having visits > 1) t
join restaurants r on r.r_id = t.r_id group by t.r_id
order by loyal customers desc limit 1;
```

Q7. Month over month revenue growth of swiggy.

```
select month, ((revenue - prev)/prev)*100 from (
  with sales as (
      select monthname(date) as 'month', sum(amount) as 'revenue'
      from orders group by month order by month(date)
  )
  select month, revenue, lag(revenue,1) over(order by revenue)
  as prev from sales
)t
```

Q8 Customer → favourite food

29 September Tuesday 2015
Week 40th Day 272nd

Important

If alias has space then use " " or [ ]

```
with temps as(
    select o.user_id, od.f_id, count(*) as 'frequency'
8.00 from orders o join order_details od on
    o.order_id = od.order_id group by o.user_id, od.f_id
9.00 )
  select u.name, t1.frequency, f.f_name from temps t1 JOIN
10.00 users u on u.user_id = t1.user_id join food f on
  f.fid = t1.f_id where t1.frequency = ( select max(frequency)
11.00 from temps t2 where t2.user_id = t1.user_id);
```

Ques:- Select all customers with city starting with 'b', 's', 'p'.

```
Select * from Customers where City like '[bsp]%';
1.00
```

Ques:- Select all customers with 'a-c' as starting.

```
2 Select * from Customers where City like '[a-c]%';
```

Ques:- Select all customers with b, s, p not starting.

```
select * from customers where City like '[!bsp]%';
4.00
```

h[^oa]t ⇒ hit not hot, hat

```
5.00
```

Ques:- Order b/w '01-July-1996' and '31-July-1996'.

```
Select * from orders where date b/w #07/01/1996# and #07/31/1996#
```

Ques:- using Alias.

```
Select contactname as [name_last] from Customers;
```

SEPTEMBER 2015

Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Ques:- Copy suppliers table into customers.

```
insert into customers (name, city) select
  name0, city from Suppliers
```

2015

Week 40th Day 273rd

September 30 Wednesday

Important Ques:- Copy german suppliers to Customers.
insert into customers (name, city, country) select
supplyname, city, country from suppliers where country = "Germ.".

Ques:- Create full copy.
select * into Backup from Customers.
select * into Backup in 'Backup.mdb' from Customers.

Ques:- Create new empty table.
select * into new from old where 1=0;

Ques:- Order customers by city but if null then by country.
select name, city, country from Customers Order by
(case when city is Null then Country.
Else city) End);

Ques:- Return alternative if UnitOnOrder is NULL.
select name, price*(unit * IfNull(UnitOnOrder, 0));
or IsNull(); Coalesce()

Ques:- Stored procedure :-
Create procedure SelectAll @City varchar(30), @ code varchar(10)
as
select * from Customers where City = @City and code = @ code
go;

Execution :-
Exec SelectAll @ City = "London", @Code = "WA12";

Ques:- Comments.
-- (Single)
/*(many) */

OCTOBER 2015

Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	
Friday	2	9	16	23	
Saturday	3	10	17	24	
Sunday	4	11	18	25	

OCT NOV DEC

Alter Table Persons → Add Column (name)
Drop Column-name)
Alter Column Datatn

01 October 2015
Thursday
Week 40th Day 274th

Important

Ques: SQL Backup DB.
backup database test
To disk = 'D:\back\test.db'

Back up only changes.
backup database test
To disk =
with differential.

Ques: Add unique constraint
Alter table Person
Add constraint UC_person Unique (ID, name);
drop index UC_person

Ques Drop Primary key
Alter Table Person
drop primary key

Alter table person
Drop constraint PK_person

Ques: Add check constraint:-
Add constraint Chk_person Check (Age >= 18 & name like 'A-B%')

Ques: Default
City varchar(255) default 'Asia';
Add constraint df_city default 'Asia' for City
Alter Column City Drop Default;

Ques Create index.
Create index idx on
persons (lastname);

create index idex on
persons (lastname, first)

Ques: Drop index
drop index name on table;
drop index table_name.index name

Alter table (name)
drop index (name);

OCTOBER 2015

Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	31
Sunday	4	11	18	25	

Ques: Auto increment start, increase by2
#personId int identity (1,1)
#Alter table persons Auto-Increment = 5;
#create tablename
personId int not null auto-increment ;

2015 select city from supplier

October 02 Friday

Week 40th Day 275th

Important SQL dates :- Date YYYY-MM-DD
Datetime YYYY-MM-DD HH-MM-SS
Timestamp - same - small Date Time
Year YYYY or YY

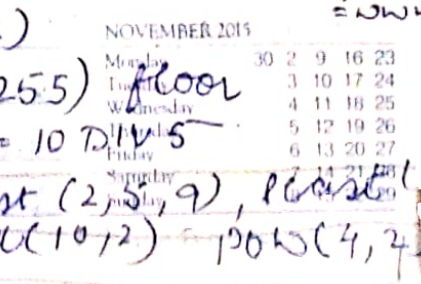
Ques:- View
create view [new] as select name from customers where ID = 1;
select * from [new];
update view ⇒ create or replace [new] as ----
drop view [new];

Ques:- Select top, limit, rownum.
select * from Customers where rownum <= 3;
select * from Customers limit 3;
select top 3 * from Customers.

Ques:- Set
update customers set name = "Alfo", city = "France" where ID = 1;

Ascii (character)
char_length (character)
concat (A, ' ', B);
field (value, val1, val2) (1 means 1st place position)
find_in_set ('a', 's,a,l')
format (80.1129, 2)
insert ("W3schools", 1, 9 "Example", Example)
instr ("W3school", 3) Ans = 2
lower(), space(10), Ucase()
left ("SQL", 2); right ("SQL", 2);
locate ("3", W3 School) Ans = 2
lpad ("SQL", 20, "AB(-)") ⇒ ABCABCABCSQL
ltrim (" SQL")
Rand(); Round (25.1, 1)

Mid ("SQL Tut", 5, 3)
position ("3" in "W3School")
repeat ("SQL", 3)
replace ("SQL", "QL", "LI")
reverse ("SQL")
strcmp ("SQL", "SQL") = 0
substr ("SQLBB", 3, 2)
substring_index ("www.w3cap", ".", 1) = www
ABS (-2)
ceil (255.5) floor
10 / 5 = 10 DIV 5
Greatest (2, 5, 9), Least()
mod (10, 2) pow (4, 2)



OCT NOV DEC

03 October Saturday

2015 101

Week 40th Day 276th

Important SQRT(64)

Date_Add( ), Sub Date ("2017-06-15", : 2017-06-05"
Add date ("2017-06-05", INTERVAL 10DAY) = 2017-06-15
Add time ("207-06-15 09:34:21.001", "5 2:10:5.003") = 2017-06-2
Curdate()+1 = 2022-08-28 OR Current_date()+1
Current_time() or Curtime()
Current_timestamp() : 2022-08-28 05:38:00
Date ("2017-06-05 05:11:39") = 2017:06:05
Datediff("2017-06-25", "2017-06-15") : 10 (return no. of days)
Date_formate ("2017-06-15", "%Y") = 2017
"%W %M %e %Y" = Thursday June 15, 2017
%a abbraviated weekday %b = Jan %c: 0-12
%D = 1st, 2nd, %d, %e = 1,2-31 %f = microsec = 999999
%H = 00 to 23 %h = 00 to 12 %h = %I %i = mins 00 to 59
%j: day of year %k = hr (0 to 23) %l = hr (0-12) %M = Month January
%m = (00 to 12) %p = AM or PM %r = (hh:mm:ss AM)
%S or %s = 00 to 59 sec %T = hh:mm:ss %U (1 = week 00 to 53)
%V = week (01 to 53) %W = Monday %w = (0 to 6 monday)
%X = year %Y = YYYY %y = YY
Date_Sub ("2017-06-15 09:34:21", interval 15 minute)
Day ("2017-06-15") = 15
Dayname ("2017-06-15") = Thursday dayname(curdate())
Dayofmonth ("2017-06-15") = 15
Day of week ( ) = 05 (Thursday)
Day of year ("2017-06-15") = 166
Extract (week from "2017-06-15") = 24
From Days (730500) = 2136-12-08
Last Day ("2017-12-15") = 2017-12-31

October						
Monday						
Tuesday						
Wednesday						
Thursday						
Friday						
Saturday						
Sunday						

localtime(), localtime stamp()
makedate(2022, 60) = 2022-03-01
maketime(9, 35, 4) Select now()
Monthname() = January.

Addy period_add(yyyymm, 15) = 201806 (months)
period_diff(201803, 201703)

Quarter("2017-06-15") = 2
Sec_to_time(1) : 00:00:01
Str_to_data("August 10 2017", "%M %d %Y"); = 2017-08-10
Sysdate(); Sysdate(); Time("2017-08-15 19:30:10.000")
Time_format("19:30:10", "%T") - 19:30:10
Time_to_sec("19:30:10") = 70210
Timediff("13:10:11", "13:10:10") = 00:00:01
To_days("2017-06-20") Week("2017-06-15") = 24
Weekofyear("2017-06-15") = 24
Yearweek("2017-06-15") = 201724
Cast("2017-10-4" As date) . Convert()
Coalesce(null, null, "w3", null, "learn");
Conv(4, 10, 2);
Current_user(), , database()
If(strcmp("hello", "bye")=0, "Yes", "No");
Ifnull("none", "w3") = none ; Ifnull(null, "w3") = w3
Isnull(null) = 1 ; nullif("Hello", "World") = hello

Ques :-
create table branch(
foreign key(mgr_id) references employee(empid) on
delete set null); or on delete cascade;

decimal(3,2) → 2 after decimal.
3 before
describe table