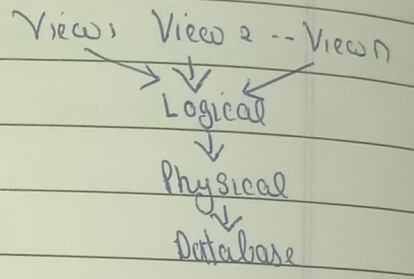
Gaurav DBMS Revision Notes

DATE: / /

Drawback of file System :

- Data redundancy and inconsistency
- Difficult to access data
- Data isolation
- Integrity problem : Hard to add or change new constraints
- Atomicity of updates : Failure can cause partial update.
- Concurrent access by multiple users
- Security

Level of Abstraction :



① Physical level : How data/record stored
② Logical level : Store relation of data
③ View level : Hide detail of data types
→ stored in data dictionary

Schema : Logical structure of database (attribute, relation, etc)

Physical Schema : Database at physical level
Logical Schema : — // — Logical level

Instance : Actual database content at a point of time

Physical Data Independence

- Concern with data storage
- Easy to retrieve
- Easy to achive ~~pto~~ physical ~~this~~
- Change here not affect Logical
- Concern with internal schema

Esc : Change algo, storage devic

- Capacity to change ~~Logical~~ physical with out change logical

Logical — // —

- Concern with structure, data defination
- Difficult to retrieve
- Difficult to achive - this
- First need to change app program
- Concern to external schema

Esc : Add modify attribut

- Capacity to change logical without physical

PAGE :
DATE : / /

Data Model :

① Relation model ② ER (Enity Relationship model)
③ Object based data model ④ Semi Structured data model (xml)
⑤ Network model ⑥ Hierarchical model

DML (Data manupulation Laung.) : Known as query language

- Procedural : what data and how to get it, is specified
- Declarative (Non-procedural) : Only what data is specified.

DDL (Data Defination Language) :

- Specific notation for defining database schema. Ex: Create table
- DDL compiles generate set of table store in dictionary
- Data dict. contain meta data {schema, authorization, constraint}

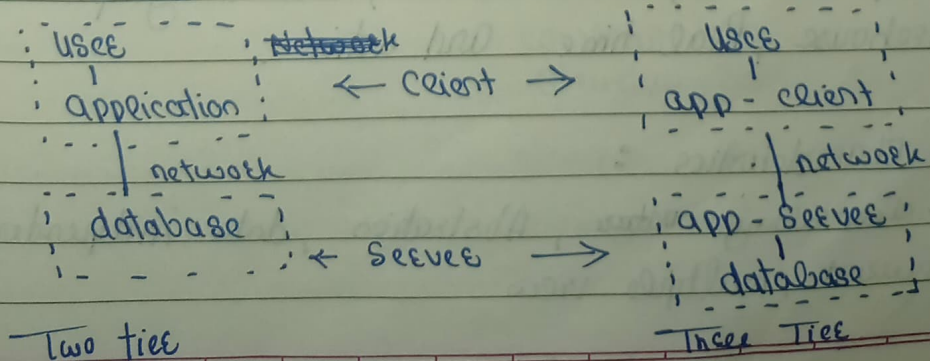
Q. Bal. of all acc. with customer id 156-78

Select account.balance from depositor, account
where depositor.customer_id = 156-78 and
depositor.account_number = custce.account_nunber.

ER model : Model an enterprise as collection of entity & relation

Entity : An object (distinguishable) (set of attributes)
Relation : Association between several entity.

Architecture :



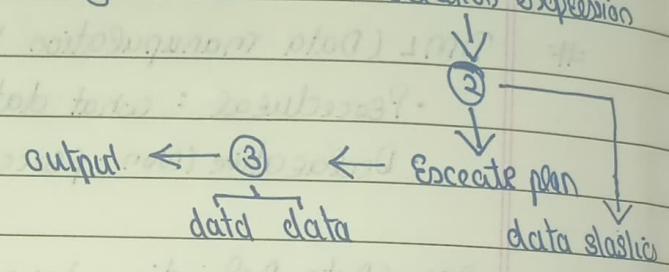
Distributed Environment: Now know as client server based system as support a set of database seever and clients.
- Homogenous
- Hetro -11-
- multi database system

PAGE:
DATE: / /

DBMS Internal

- Storage management: Provide interface bet" low level data in database app and queries submitted
  Interact with file manager, efficient storage, update, etc

- Query Processing:
  1) Passing and Translation
  2) Optimization
  3) Evaluation



- Transaction Managment
  Collection of operation that perform single logical fn.
  Manage consistency, concurrency, Power failure

Database Users:

① Naive: People accessing database over web.
② Sophisticated: form ~~data~~ request on database query.
③ Specialized: write database app.
④ Application Programmer: interact system through DML case.
⑤ Database Administrator

Architecture: Centralized, client server, parallel, distributed

Database Type:

Numeric & Textual, Multimedia, GIS (Geographic info sys)
Datawarehouse, Real time and active

Database Characteristics:

- Self-describing nature, Abstration, data independence
  concurrent, multiple views

- Data dict / repo: Store schema and other info like app. program.
  Active: Accessed by ~~DBMS~~ user / DBA, DBMS software
  Passive: Accessed by DBA / user only.

PAGE :
DATE : / /

Hierarchical Model :

+ve
- Simple to construct and operate, correspond to no. of hierarchy
  Eg: assembly in company. Ex: Get, Getnext, Get unique, etc

-ve
- Navigation and procedural nature, visualized as linear arrangement of record, ~~the~~ little scope for query optimize

Network Model :

+ve
- Able to model complex relation, can handle most situation using record / relation type. Navigational language like Find member, Get., Find owner, Find next within set, etc

-ve
- Navigational and procedural, contain complex pointer array, little scope for query optimization

Database Schema

Prog.

Course No.	Prog. No

Database instance

Prog

Course No	Prog. No
CS30	MATH31C
CS31	Eng 382
⋮	⋮

Database State : Initial, Valid.

- Distinction : database schema change very infrequently. database state change every time data update
- Schema is called intension, instance state is extension

Three Schema Archi :

End User

External View ... Ext. View

↘ ↙

constraint / structure, logical level ← conceptual Schema

Physical Storage / level ← Internal Schema

↓

Database

Two —

Relational Model

PAGE :
DATE : / /

Domain : Set of allowed value for each attr.
Attribute value required to be atomic means indivisible

Categorical Language: Procedural, Non procedural
"Pure" Language: Relational Algebra, Tuple relation calculus
Domain relation calculus

Relational Algebra : A∪B so A and B must have same no of attribute means Same arity and ↑ exc: 2nd coln of A is val then 2nd coln of B also val
Select : σ , Project : Π , Union : ∪ , Rename : ρ
Set difference : − , cartesian product : X
Set intersection : A∩B = A−(A−B) ↳ make pair of every tupple with every other

Select used to specify which tupple we want
Project used to specify which attr. column we want.
Rename : $\rho_x(E)$ ⇒ return expression E under name X
$\rho_{x(A_1, A_2 \cdots A_n)}(E)$ ⇒ —"— under name x and attribute name as $A_1, A_2 \cdots A_n$
Assignment operation : Assign variable : temp ← $\Pi_{\varepsilon\text{-}S}(\varepsilon)$

branch (b-name, b-city, asset)
customer (c-name, c-street, c-city)
account (a-number, b-name, balance)
loan (l-number, b-name, amount)
depositor (c-name, a-number)
borrower (c-name, l-number)

Q. All loan > 1200 : $\sigma_{amount>1200}(loan)$
Q. Loan number for above : $\Pi_{l\text{-}number}(\sigma_{amount>1200}(loan))$
Q. Find c-name who have loan or account from bank :
$\Pi_{c\text{-}name}(borrower) \cup \Pi_{c\text{-}name}(depositor)$

$\Pi_{c.name}$ (borrower ⋈ loan)

PAGE :
DATE : / /

Q. Name of all cust. who have loan at Perry branch

⇒ $\Pi_{c\text{-}name}$ ($\sigma_{b\text{-}name = \text{"perry"}}$ ($\sigma_{borrower.l\text{-}number = loan.l\text{-}numb}$ (borrower × loan)))

Q. Find c_name with loan at 'Pe' branch and do not have account at branch

⇒ $\Pi_{c\text{-}name}$ ($\sigma_{b\text{-}name = 'Pe}$ ($\sigma_{borr.l\text{-}num = loan.l\text{-}num}$ (borrower × loan))) − $\Pi_{c\text{-}name}$ (depositor)

• Set intersection : A ∩ B ⇒ common tuple of A, B selected

• Natural Join : E ⋈ S ⇒ Only make pair of common tuple

A	B	D
α	1	a
β	2	a
α	2	b

⋈

B	D	E
1	a	γ
1	a	γ
2	b	γ

⇒

A	B	D	E
α	1	a	γ
α	1	a	γ
α	2	b	γ

(A⋈B)⋈C = A⋈(B⋈C)

E ⋈ S defined as: $\Pi_{SA, SB, SE, SD}$ ($\sigma_{E.B = S.B \wedge E.D = S.D}$ (E × S))

$E \bowtie_{\theta} S = \sigma_{\theta}(E \times S)$ ⇒ θ (predicate)

Aggregate Function : Take collection of value and return 1 value

avg, min, max, sum, count

$G_1 G_2 \ldots G_n \; \mathcal{G}_{F_1(A_1), F_2(A_2), \ldots F_n(A_n)} (E)$

↳ attribute which to group ; F_1 = aggregate F_n, A_n = attribute name

• Sum on relation account group by b_name : b_name $\mathcal{G}$ sum(balance) (account)

To change sum attribute name : b_name $\mathcal{G}$ sum(bal..) as sum-ba (account)

⊖ Aggregate Ignore Null value.

Division (÷) : Binary operation used for query that have "for all"

Let E(R) and S(S) be relation S ⊆ R means every attr of S in R. Relation E ÷ S is relation on E-S means (E ÷ S will have only attr which are in E but not in S). Also for every tuple in S there tuple in E which satisfy a) $t_E[S] = t_S[S]$ b) $t_E[E-S] = t$

Q. Customer name who have acc in all branch of 'Brooklyn' city

$\Pi_{c\text{-}name, b\text{-}name}$ (depositor ⋈ account) ÷ $\Pi_{b\text{-}name}$ ($\sigma_{b\text{-}city = \text{"Bro"}}$ (branch))

Q. C-name who have account from "Downtown" and "Uptown" branch

⇒ $\Pi_{C.name}(\sigma_{b\text{-}name = "Downtown"}(depo \bowtie account)) \cap \Pi_{cname}(\sigma_{bname = uptown}(depo \bowtie account))$

⇒ $\Pi_{cname, bname}(depositor \bowtie account) \div \rho_{temp(bname)}(\{"Downtown", "Uptown"\})$

Generalized Projection: Extend projection operation by allowing arithmatic f^n $\Pi_{F_1, F_2, F_3 \cdots F_n}(E)$

Exc: Given credit-info(c-name, limit, credit-bal) find how much more each person can spend: $\Pi_{c\text{-}name, limit - credit\text{-}bal}(credit\_info)$

Deletion: $\mathcal{E} \leftarrow \mathcal{E} - E$ -- (E = Relational alg. query)

Q. Delete all record of "Perry" branch:

⇒ account ← account − $\sigma_{b\text{-}name = "Perry"}(account)$

Q. Delete all acc. at branch located in Needam.

⇒ $\mathcal{E}_1 \leftarrow \sigma_{branch\_city = "Needam"}(branch \bowtie account)$

$\mathcal{E}_2 \leftarrow \Pi_{acc\text{-}number, b.name, bal}(\mathcal{E}_1)$

account ← account − $\mathcal{E}_2$

$\mathcal{E}_3 \leftarrow \Pi_{c\_name, a\text{-}number}(\mathcal{E}_2 \bowtie depositor)$

~~account ← account − $\mathcal{E}_2$~~

depositor ← depositor − $\mathcal{E}_3$

Insertion: $\mathcal{E} \leftarrow \mathcal{E} \cup E$

Q. Provide 200$ Saving acc to all loan customer at "Perry" Loan no = account no. for new saving account:

⇒ $\mathcal{E}_1 \leftarrow \sigma_{bname = perry}(borrower \bowtie loan)$

account ← account ∪ $\Pi_{loan\text{-}num, b\text{-}name, 200}(\mathcal{E}_1)$

depositor ← depositor ∪ $\Pi_{c\_name, a\text{-}number}(\mathcal{E}_1)$

PAGE :
DATE : / /

Updating : $E \leftarrow \Pi_{F_1, F_2 \cdots F_n}(E)$

Q. Pay all account balance over 10,000 with 6% interest and all other with 5% interest.

$\Rightarrow$ account $\leftarrow \Pi_{a\text{-}num, b\text{-}name, bal * 1.06}(\sigma_{bal > 10000}(account)) \cup \Pi_{anum, bname, bal * 1.05}(\sigma_{bal \leq 10000}(account))$

Cartesian Product : (X)

R {A, B, C} ; S {A, C, D}

R x S contain { R.A , S.A, B, R.C, S.C , D }

all possible pair all present.

SQL

Create table r (A1 D1, A2 D2 ...)
r → table name
A1 = attribute name, D1 = data type Exc: c_id varchar(10)
data type: char(), integer(), varchar()
↳ int(), small int()
- numeric(p, d): precision of p digit, d digit or sign of decimal.
- real, double precision: floating pt. and double precision floating pt.
- float(n): n is user specified precision.

Integrity constraint:
Exc: c-id var(10) not null, primary key (c-id)

insert into account
values ('A-20', 'Perry', 1000)

delete all tuple from account, delete from account

drop table: delete all info. about dropped ~~table from~~ relation & from database.

alter table: Add attr. to existing relation, or drop an attribute
Exc: alter table r add A D
A = attr name, D = domain of A. Initally all value are assigned null to A
Exc: alter table r drop A ⇒ This dropping may not be supported
r → attr.

Select A_1
From R → Relation table
Where P → predicate
} equivalent to ⇒ $\Pi_{A_1}(\sigma_P(R))$

Select distinct b.name from loan

Select all — " —
↳ duplicate will not be removed

PAGE :
DATE : / /

↱ all attributes

Select * from loan

Select can have arithmatic expression Exc: Select balance * 100

where specify condition and use logical connection 'and' 'or' 'not'

Select * from borrower, loan ⇒ this means borrower × loan

Rename (as) : Select b_name as ben from branch
Exc: Find name of all branch that have more asset than some branch in Brooklyn :
⇒ Select distinct T.b_name
from branch as T, branch as S
where T.asset > S.asset and S.b_city = "Brooklyn"
Keyword "as" is optional : branch as T = branch T but not in oracle

String Operation : Operator "like" use pattern described using :
① (%) : % match any substring ② (_) : underscore match any char
Exc: Name of cust whose street have substring "Main" :
⇒ select c_name from customer where c_street like '%Main%'
To match name "Main%" escape Exc: like 'Main\%' escape '\'
• Concatenation using '||'

Order by :
Exc : List in alphabatic order name of all cust. with loan in "Perry"
Select distinct c_name from borrower, loan
where borrower.loan_number = loan.loan_number and b_city = "Perry"
order by ~~customer~~ c_name
^

Use order by c_name desc for descending order

PAGE :
DATE : / /

Set operation : $\cup, \cap, -$ means union, intersect and except respectively
This operation automatically remove duplicate
If not remove duplicate use union all, intersect all, except all

Q. Find customer who have loan, account or both
⇒ (Select c_name from depositor) UNION (Select c_name from borrower)
Similarly, for loan and account use intersect
and for account but no loan so except

Aggregate Function :
Exc: Select avg(bal), Exc: Select Count(*)
Exc: Select count (distinct c_name)

Q. Find no. of depositor for each branch
Select b_name, count (distinct c_name)
from depositor, account
where depositor.acc_number = acc.acc_number
group by branch_name

Q. Name of all branch with avg(bal) > 1200
⇒ Select b_name, avg(bal)
from account
groupby b_name
having avg(bal) > 1200

↳ This is applied after group formation
where is applied before group formation

PAGE :
DATE : / /

Q. Find C-name who have loan ~~and~~ account but not bank account.

⇒ Select c_name from borrower where c_name not in
(Select c_name from depositor)

Q. Find nth highest sal : Use Scott Schema

Select name, sal from emp e1 where N-1 = (Select count(dist sal) from emp e2 where e2.sal > e1.sal

Sub Query : ① Single Row ② Multiple row ③ Multi column

For single row we use : =, >, >=, <=, <>

For multi row we use : IN, ANY, ALL

Q. Find branch which have greater asset than some branch in Brooklyn

⇒ Select b_name from branch where asset > some (
Select asset from branch where b_city = "Brooklyn")

For all branch which have greater than all branch of brook use "all"

Exists

Unique

delete from account where b_name = "Perr"

insert into account values ('A-77', 'Perr', null)

update account set balance = balance * 1.06 where balance > 1000

! Similar way

update account set balance = case when balance > 1000 then bal * 106
else balance * 105 end.

PAGE :
DATE : / /

Joins : ↱ Here same attr of both table R, S is A then in answer R.A and S.A is present

① Inner Join : loan inner join on loan.l-num = borro.l-num
Here only common tuple joins

② Left outer Join : all left table attribute present and if no value for a attr. then null.

③ Natural Join : Same as inner join but no repeat attribute

④ Right Outer Join : All right table attr present default value null.

- Natural Join can get in problem if same name attr in both R, S so use loan full outer join borrower using (loan_number)

Relation not conceptual but made visible to user is view.
create view v as <query expression>

Q. View of all branches and their customer
create view all_cust as
(select b_name, c_name from depositor, account
where depos. acc-numb = account. acc-numb)

One view can be used to define another.
- V_2 used to define V_1 then V_1 directly depend on V_2
- If V_1 depend direct on V_2 or path from V_1 to V_2 then V_1 depend on V_2
- V is recursive if it depend on itself.

With clause provide a way of define temporary view
Find all account with max balance
- with max_bal (value) as select max (balance) from account
Select a_number from acc, max_bal where acc.bal = max_bal.val

Functional Dependencies and Normalization

PAGE :
DATE : / /

Semantic of Relation Attr.

① Each tupple in relation should represent one entity
② Design schema that does not suffer from insert, delete anomly
③ Relation should have NULL value as low as possible
④ Relation should be design to statisfy lossless join. No spurious tupple should be generated by natural join.

• Spurious Tupple: Bad decision for database may result in problem for some Join operation. Use looseless join to guarantee meaningfull result.

E

x	y	z
x_1	y_1	z_1
x_2	y_2	z_2
x_3	y_2	z_3

Check if $X \to Y$ and $Z \to Y$ decomposition is looseless

A

x	y
x_1	y_1
x_2	y_2
x_3	y_2

B

y	z
y_1	z_1
y_2	z_2
y_2	z_3

A Join B ⇒

x	y	z
x_1	y_1	z_1
x_2	y_2	z_2
x_2	y_2	z_3
x_3	y_2	z_2
x_3	y_2	z_3

Spurious Tupple

FD are used to define normal form for relation, they are constraint derived from interrelation of data.

$x \to y$ hold if whenever 2 tupple have same value of x also have same y means $t_1[x] = t_2[x]$ then $t_1[y] = t_2[y]$

Example: Social Security Number ⇒ Employee Name.

∴ $A \to A$ always hold functional dependency.

Inference Rule for FD

① IR_1 (Reflexive): Y subset of X then $X \to Y$
② IR_2 (Augment): $X \to Y$ then $XZ \to YZ$
③ IR_3 (Transitive): $X \to Y, Y \to Z$ then $X \to Z$

} Armstrong Rule

④ Decomposition: $X \to YZ$ then $X \to Y, X \to Z$
⑤ Union: $X \to Y, X \to Z, X \to YZ$
⑥ Pseudotransitivity: $X \to Y, WY \to Z$ then $XY \to Z$
⑦ Closure F^+

PAGE :
DATE : / /

If closure of an attr contain all attr then it is super key
Candidate key have min no. of key possible.

F and G are equivalent if $F^+ = G^+$
F covers G if G^+ subset of F^+

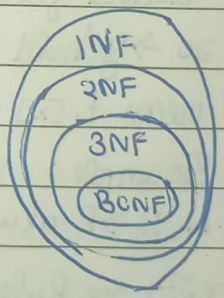
Attr is prime if it is member of some candidate key

Full FD means if from $Y \to Z$ if any attr of Y remove then FD fails

Non Trival FD : $X \to Y$ where Y not subset of X
Trival FD : $X \to Y$ then Y subset of X

Normalization : Process of decomposing bad relation by breaking attr. into smaller relation.
Denormalization : Process of storing join of higher normal as base relation which is in lower normal form.

1NF : Relation schema R is in 1NF if all attr are atomic. There should be no attr with multi value



2NF : R in 2NF if every non prime attr is fully dependent of primary key
$AB \to C$ } partial dependency
$A \to C$

PAGE :
DATE : / /

~~3NF : X→Z can be derived from X→Y and Y→Z~~

3NF : R in 3NF if it is in 2NF and no non-prime attr A is transitively dependent on primary key

Exc : X→Y and Y→Z if X is ~~can~~ primary key and Z non prime then not in 3NF if Y is candidate key then in 3NF

Means X→A hold 3NF if

- X is superkey OR A is prime attr.
- One non-prime should not determine another non prime

Boyce Codd Normal Form (BCNF) : Should be in 3NF wherever if X→A then X is superkey

BCNF can be achieved by decomposition

Exc : AB→C , C→A

AB is candidate key. This in 3NF but not in BCNF so decompose as CA and CB

Set of FD is minimal if :

① Every dependency in F have single attr in RHS

② We cannot remove any dependency F and have set of equi to F.

③ Cannot replace X→A with Y→A where Y subset of X

Canonical Cover : If any update occur then it must not violate and FD. If it does then roll back. To check for violation just find some short dependency which can closure to all attributes

PAGE :
DATE : / /

Storage and File Structure

Physical Storage : Magnetic disk, RAID, Tertiary Storage, File organization, Data-Dictionary, Records

Classification done on basis of : data access speed, cost per unit of data, reliability, volatile or non volatile.
↳ data loss on power off.

Storage media : Cache (volatile), Main Memory (RAM) (volatile)
- Flash memory (Non volatile, data can be written to location once, location can be erased and written again) (write is slow)
  NOR Flash : Fast read, very slow, lower capacity, store program code.
  NAND Flash : High capacity, widely used as data storage.
- Magnetic Disk (non volatile, store whole database, read write any order)
- Optical storage (read using laser CD-R, CDRW, DVD-R, DVDRW)
- Juke Box : System with many removable disk
- Tape Storage : Non volatile, use for backup, sequential access slow than disk, high capacity

cache → MM → Flash → Magnetic Disk → Optical Storage → Magnetic Tape

Hierarchy : Primary (cache, mm), Secondary (Flash, Disk), Tertiary (tape, optical)
↳ online storage ↳ off line storage

Magnetic Disk : Surface, Circular track, Sector (512 byte)
↳ 50k track ↳ 1000 sector

Disk interface family : ATA, Serial ATA, SCSI

Disk ARM Scheduling : minimize disk arm movement
Elevator algo. : move arm in one direction until request of that direction over then in other direction

PAGE :
DATE : / /

File organization : optimize block access-time by organizing block. File may get fragment means data store sequential location get scatter due to insert, delete.

Non volatile ~~RAM~~ buffer : Battery backed RAM, Flash memory

Log disk : Disk devoted for logs used like non volatile RAM (NV-RAM)

RAID (Redundant Array of independ disk) :
High capacity, High speed, multiple disk use parallel.

Redundancy : Store extra info. which can be used ~~to ref~~ at failure.

Mirroring : Duplicate every thing. Every write is done on two disk one is for backup

Parallelism : ① Load balance multiple small access to ↑ throughput
② Parallelize large access to reduce response time
Improve transfer rate by stripping access multiple disk.
Bit level stripping, Block level stripping.

Raid Level 0 : Block strip, non redundant

Raid Level 1 : Mirror disk, block strip (used for log storing)

—11— 2 : Memory style error correction, bit stripping

—11— 3 : Bit Interleaved Parity (Parity bit used) (Fast transfer rate)

—11— 4 : Block Interleaved Parity (keep parity block on seperate disk)
Fast I/O rate than level 3

—11— 5 : Block Interleaved Distributed Parity
5 disk, Parity for nth set of block in (n mod 5) + 1

—11— 6 : P+Q redundancy, simple to level 5 but extra redundant information.

PAGE :
DATE : / /

Choice of RAID Level :
Monetary, performance, Perf. during failure, Perf. during rebuild
Raid 0 when data safety not imp.
Raid 2 and 4 never used as subsumed by 3 and 5
Raid 3 not used as bit stripp more arm movement
Raid 6 easily used.
Raid 5 preffered for app. with low update rate, large data
Raid 1 low data, fast update rate.

Hot Swapping : Replace disk while system running

Buffer : Portion of mm store copies of block
Buffer Manager : If block already in buffer return address, if not allocate space
Buffer Replace Policy : Last recent used.
Pinned Block : memory block not allowed to return back
Tossed Block immediate : free space as final tupple processed
LRU (Most recent used)

Database collection of File. File is sequence of record, Record are sequence of fields.
Fix Length record : Sequence each tupple by record number.
Free List : Store first deleted block add, one delete block point other del.
Variable Length record.
Slotted page in variable length record contain :
① no. of record entries ② end of free space ③ location, size of record
Heap, Sequential, Hashing, multiable clustering organization
↳ store several relation in one file
Data Dictionary contain : Relational-metadata, Attribute-metadata, user metadata, index metadata, view metadata.