Chapter-1

→ DBMS is a collection of interrelated data and a set of programs to access those data.

Advantages

1. Data redundancy and inconsistency removed.
2. Ease in accessing data
3. Data isolation
4. Integrity (जैसे library में होता)
5. Atomicity ~~problem~~ (transaction)
6. Concurrent access
7. Security

Types of DBMS

1. Hierarchical → Tree-like structure. Data is stored Hierarchical (top down / bottom up) format.
   → Data is represented using parent-child relationship
   → parent may have many children, children have only one parent.

2. Network model:
   → child to have multiple parents.
   → entities are organized in graph, which can be accessed through several paths.

3. Relational
   → based on normalizing data in rows and columns of table.
   → stored in fixed structures & manipulated using SQL

④ Object-Oriented Model.

→ data stored in form of objects.

→ classes displays data within it

→ it defines databases as a collection of objects which stores both data members values and operations.

Data abstraction / level of Database.

① Physical / Internal

② Logical / Conceptual

③ View / External.

example : University database.

Types of level.	Implementation
① External / view	view1 : CourseInfo (cid: int, cname: string) View2: studentInfo (id: int, name: string)
② Logical/Conceptual	Students (id: int, name: string, login: string, age: integer) courses (id: int, cname: string, credit: int) enrolled (id: int, grade: string)
③ Physical	→ Relations stored as unordered files. → Index on the first column of student

Data independence → D.I. is the property of DBMS, that

① Physical D.I.
② Logical D.I.

helps you to change the database schema at one level of a database system without requiring to change the schema at higher level.

① Physical D.I.

→ helps to separate conceptual levels from the internal/physical level.
→ helps to keep data separated from all programs that make use of it.

→ allows you to provide a logical description of database without the need to specify physical structure.
→ compared to Conceptual/Logical D.I. it is easy to achieve.

→ P.D.I. is the power to change physical storage structures or devices with an effect on the conceptual schema
→ Any change done would be absorbed by the mapping between the conceptual and internal levels.

- examples

① Using a new storage device like Hard Drive or magnetic tapes.
② Modifying the file organization technique in the Database.
③ Switching to different data structures.
④ Changing the access method
⑤ Modifying indexes.
⑥ Change of location of database from C drive to D drive.

Logical data independence

→ is the ability to change the conceptual schema without changing External views, external API or programs.

→ Any change made will be absorbed by the mapping between external and conceptual level.

Due to Logical D.I., any of the following change will not affect the external layer.

① Add/modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application program.

② Merging two records into one.

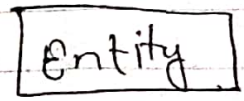
③ Breaking an existing record into two or more records.

	Logical D.I.	Physical D.I.
①	Is mainly concerned with the structure or changing the data definition.	concerned with storage of data.
②	is difficult as the retrieving of data is mainly dependent on the logical structure of data.	is easy to retrieve.

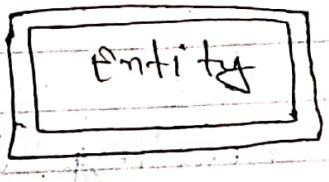
ER diagram

key words

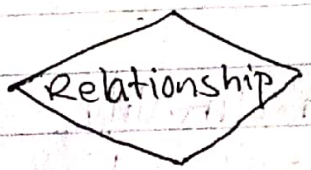
① Entity – an object or concept about which you want to store information.
→ rectangle



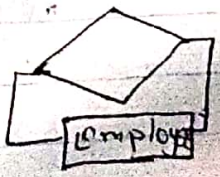
Weak entity → an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attribute alone.



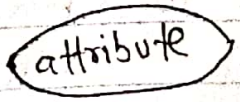
② Actions → shows how two entities share info in database.
→ diamond.



→ entities can be self linked
e.g. employees can supervise other employees.



③ Attribute. → Ovals



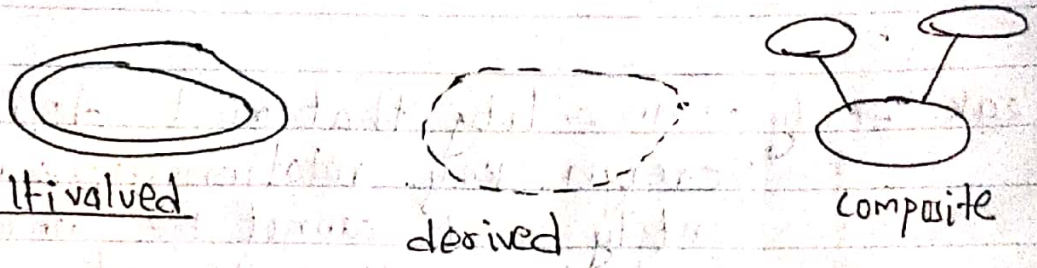
→

multivalued attribute → can have many value for particular entity attribute. e.g. mobile number.

Derived attribute → attribute based on another attribute.
e.g. employees monthly salary is based on employees annual salary.

composite attribute: attribute which can be divided into subparts.

eg. Name can be divided into subparts: FirstName & LastName



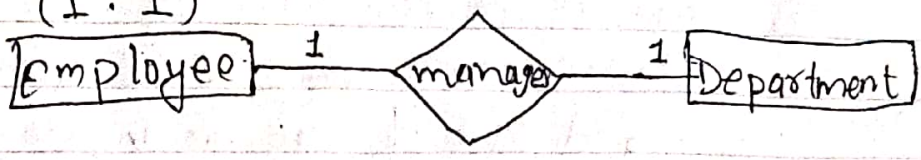
④ connecting lines – solid lines
→ connect attributes to show relationship of entities in diagram.

⑤ Cardinality
→ specifies how many instances of an entity relate to one instance of another entity.
→ specifies occurance of a relationship.
→ ordinality describes relationship as either mandatory or optional.

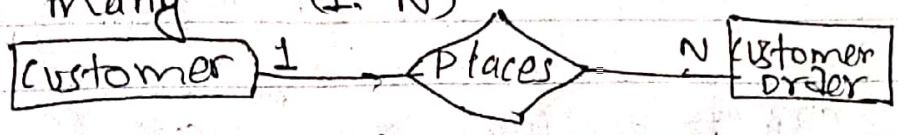
→ Cardinality specifies maximum no. of relationship.
Ordinality " the absolute minimum no. of. "

chen style

① one-to-one (1:1)



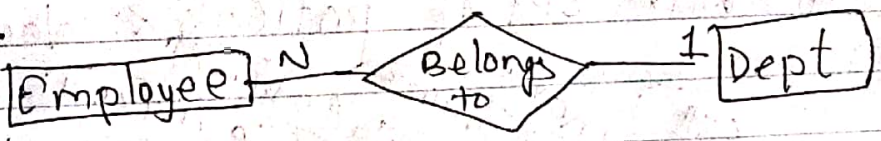
② One to many (1: N)



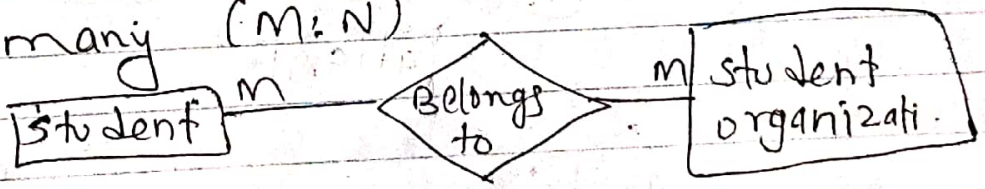
customer may place many orders, but each ~~can~~ order can be placed by one customer only.

③ many to one. (N:1)

many employee may belong to ~~ma~~ one departm
but one particular employee can belong to one dept. only.



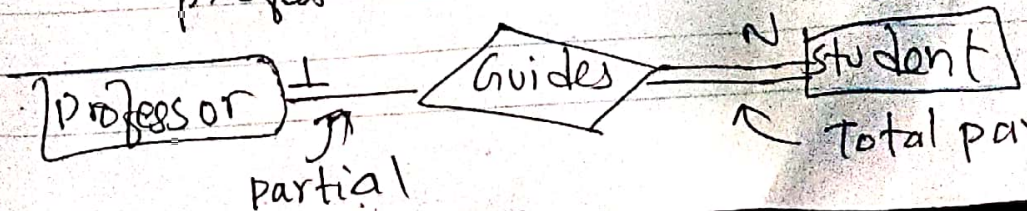
④ many-to-many (M:N)



participation

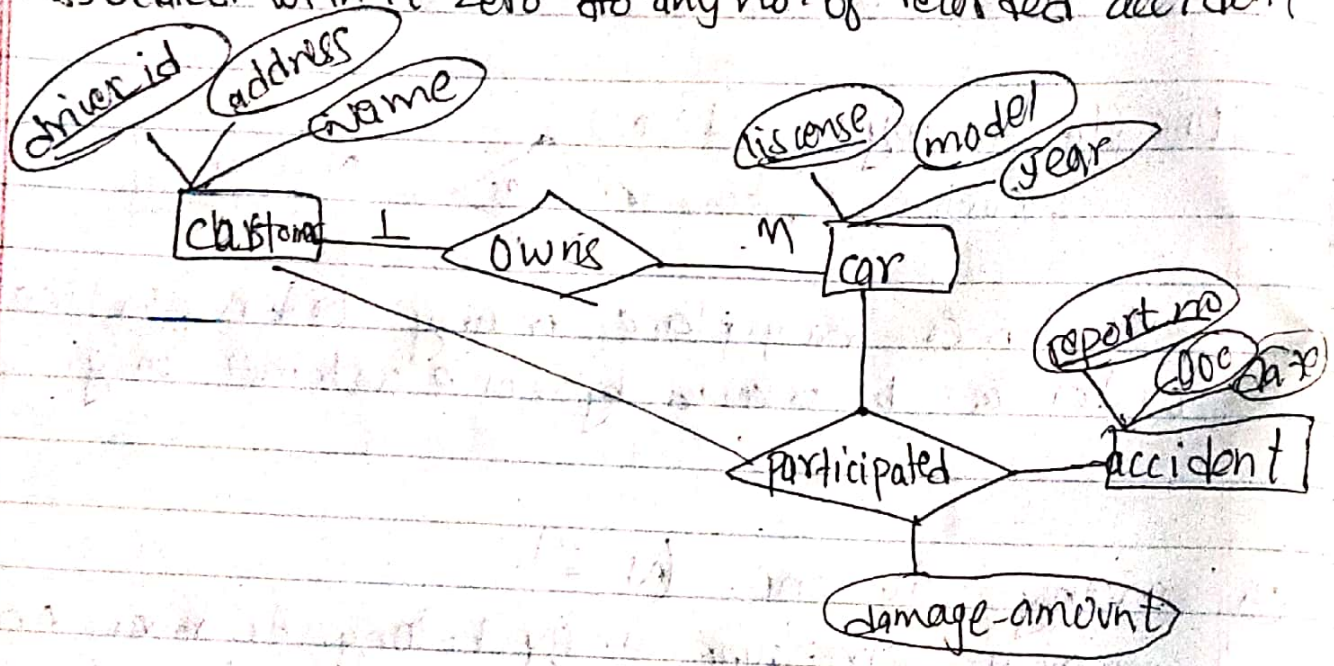
① Total/Full participation → every entity in a set is involved in the relationship.

e.g. each student must be guided by a professor (there are no students who are not guided by any professor).

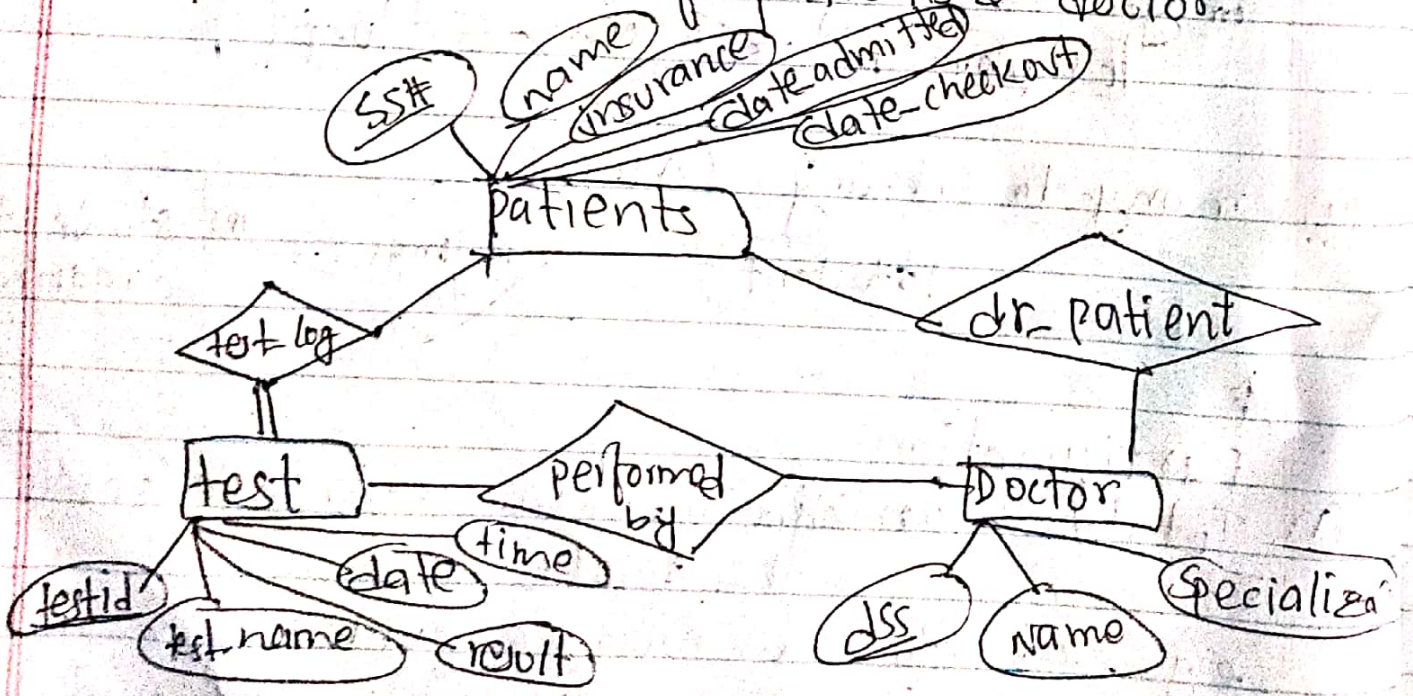


ER example

① car-insurance company:
customers own one or more cars each. Each car has associated with it zero or any no. of recorded accident



② hospital with set of patients & doctor.



data models

* Tuple Relational calculus
→ non-procedural query language
→ explains what to do but not how to do.

syntax : $\{t \mid P(t)\}$
→ t = resulting tuple
P(t) = predicate, used to fetch t.

P(t) may have various conditions logically combined with OR (∨) AND (∧) NOT (¬).
It also uses quantifiers:
$\exists t \in r\,(Q(t))$ = "there exists" a tuple t in r such that predicate Q(t) is true.
$\forall t \in r\,(Q(t) \Rightarrow$ Q(t) is true "for all" tuples in r.

* e.g. To find loan_number, branch, amount of loans of greater than or equal to 10000 amount (only selection)
loan(loan_number, branch, amount)
$\{t \mid t \in loan \wedge t[amount] >= 10000\}$

* Find loan number for each loan of an amount greater than $1500. (selection with projection).

$\{t \mid \exists l \in loan\,(t[loan\_number] = l[loan\_number] \wedge l[amount] > 1500)\}$

borrower (customer_name, loan_number)

Q. Find names of all customer who have a loan from pune branch

$\{t \mid \exists b \in borrower\,(t[customer\_name] = b[customer\_name] \wedge \exists l \in loan\,(l[loan\_number] = b[loan\_number] \wedge l[branch = "Pune"])\}$

∃ existential
∀ universal quantifier

Domain Relational calculus.

→ filtering variable uses the domain of attributes.

$$\{< x_1, x_2, x_3, \cdots, x_n > | P(x_1, x_2, x_3, \cdots, x_n)\}$$

$< x_1, x_2, x_3, \cdots >$ ⇒ resulting domain variables
$P(x_1, x_2, x_3, \cdots x_n)$ ⇒ predicate

loan (loan_number, branch, amount)

eg 1 $\{< l, b, a > | < l, b, a > \in loan \wedge a > 1500\}$

→ यसमा l, b, a सबै column.

Find all loan_number for loan with an amount greater than 1500

$$\{< l > | \exists b, a (< l, b, a > \in loan \wedge a > 1500)\}$$

→ यसमा loan_number attribute मात्र।

Data Dictionary

→ contains metadata i.e. data about the database.
→ d.d. is very imp as it contains information such that as what is in the database, who is allowed to access it, where is the database physically stored etc.
→ The users of database normally don't interact with the d.d., it is only handled by database administrator.
→ d.d. in general contains inform about following:
- names of all the database tables and their schemas.
- details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
- Physical inform about the tables such as where they are stored and how.
- Table constraints such as p.k, f.k etc.
- about db views that are visible.

⟹ DBMS needs d.d. to access data within db

Types of data dictionary.

① Active d.d.
→ any changes made on db should be reflected on d.d. → This is responsibl of dbms in which d.d. resides.
→ d.d. is automatically updated by dbms when any changes are made in db → is kla active / self-updating

② Passive
→ automatically update होता नहीं।
→ manually updated to match db.

DDL DML Dcl & TCL.

→ DDL → Data definition language
→ used to create and modify structure of db objects

CREATE, ALTER, DROP, TRUNCATE

DML Data manipulation
→ used to retrieve, store, modify, delete, insert and update data in database

→ SELECT, INSERT, UPDATE, DELETE

DCL data control language
→ used to create roles, permissions
→ used to control access to db by sewring it.

GRANT, REVOKE

TCL : Transaction control language
→ to manage different transaction occuring within database.

COMMIT → saves work done in transaction
ROLLBACK → Restores db to original state since last commit command in transaction.
SAVE TRANSACTION → sets a save point within a transaction.

QBE

→ QBE is a feature involved along with various database applications which gives user-friendly techniques of running database queries.

→ Generally, without QBE, a user must write input commands using correct SQL syntax. SQL is standard language in which nearly all database program support.

→ If the syntax is slightly incorrect the query might return the wrong results or may not run at all.

→ QBE feature gives a simple interface for a user to enter queries.

Instead of writing an whole SQL command, the user can just fill in blanks or select items to describe the query, that s/he needs to perform.

For example, a user may need to select an entry from a table called Table1 with an ID of 123.

Using SQL, the user would require to input the command "Select * From Table1 where ID=123".

The QBE interface might ~~perform~~ permit the user to just click on Table1, type in "123" in the ID field and click "search".

→ QBE is offered along with most database programs by the interface is often different among application

→ Whatever, QBE implementation gives

ⓐ to make it easier to run database queries.
ⓑ To prevent from frustrations of SQL errors.
ⓒ make it easier to get data
ⓓ Allow users to use SQL without understanding it.

multi-valued dependency

→ occurs when two attributes in a table are independent of each other but, both depend on third attribute.

e.g.

Bike model	manu_year	color
M2014	2008	white
M2015	2008	Black
M3014	2010	white

Here, columns color and manu_year are dependent on Bike_model and independent of each other.

In this case, these two columns can be called as multivalued dependent on Bike model.

These depency is represented by

Bike_mode →→ manu_year

Bike_model →→ color

Decomposition of Relation

→ Decomposition → process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.

The decomposition of a relation schema R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R.

→ Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistancy and anomalies

Types (2) (2) lossless

(1) lossy Decomposition

→ decomposition of R into R_1 and R_2 is lossy when join of R_1 and R_2 does not yield the same relation in R.

→ some information is lost during retrieval of original relation

e.g. R: Student

RollNo	Sname	Dept
111	Angel	computer
222	~~Paru~~ Angel	Electrical

R_1

Roll_no	Sname
111	Angel
222	Paru

R_2

Sname	Dept
Angel	Comput
~~Paru~~ Angel	Elect

$R_1 \bowtie R_2$

RollNo	Sname	Dept
111	Angel	computer
111	Angel	Electrical
222	Angel	computer
222	Angel	electrical

In lossy spurious, garbage tuples are generated when natural join is applied.

→ it is bad decom

chapter-01

1. Object-Oriented Model.

→ data and ~~ret~~ their relationships contained in object, which are instances of class.

Components

① object structure

→ structure of object refers to properties that an object is made up of.

→ properties of object — attribute

→ object is real world entity with certain attributes that makes up object structure.

→ object encapsulates data code into single unit — which provides data abstraction by hiding implementation details from user.

→ Object structure further consists: Message, methods, variables

Ⓐ Messages.

→ provides interface/communication medium bet" object and outside world. message can be of two types:

→ Read-only → if invoked method does not change value of variable.

→ update → invoked method changes value of variable

Ⓑ methods - when message is passed, then body of code that is executed is called method. It can be of two types

→ Read-only method - value of variable is not affected.

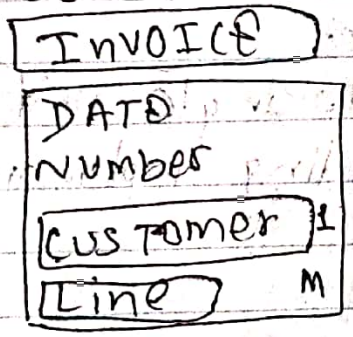
→ update " → value of variable changes by method.

(C) (3) variable.
→ stores data of an object.

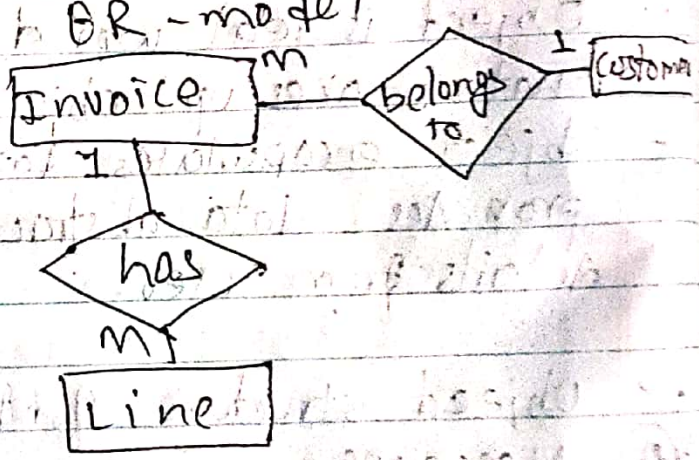
(2) object classes.
→ real world entity

→ OODM provides numerous facilities to its users.
→ It supports concept of programming paradigms.

OO data-model



ER-model



Advantage

(1) Add semantic content
(2) Database integrity
(3) Both structural and data independence.

Disadvantage

(1) Lack of OODM standard
(2) Complex navigational data access
(3) Steep learning curve.
(4) High system overhead slow transactions.

② Object- Relational data model (ORM)

→ It is combination of a object oriented database model and a Relational database model.

→ It supports objects, classes, inheritance etc. just like object-oriented model and has support for data types, tabular structures etc. like Relational data model.

→ One of major goals is to close gap between relational dm & ORM.

Advantages of ORM

① Inheritance

→ allows to inherit objects & tables

② Complex data types

→ complex data types can be formed using existing data types.

③ Extensibility

→ inheritance & complex data type use ~~गरेर~~ functionality can be extended.

④ Disadv

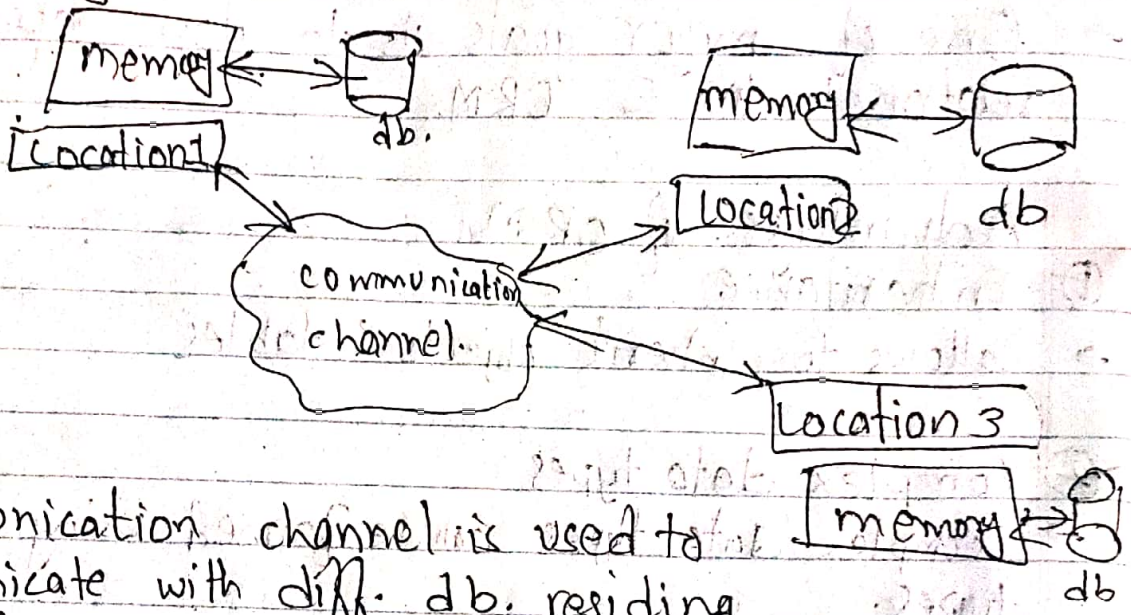
→ complicated

- difficult to handle.

③ Distributed databases

→ collection of multiple interconnected databases which are spread physically across various locations that communicate via a computer network

→ Distributed DBMS manages the distributed database in a manner so that it appears as one single db to users.



→ communication channel is used to communicate with diff. db. residing at diff. locations and every system has its own memory and database.

Goals To improve:

① Reliability → In this, if one system fails down or stops working for some time another system can complete task.

② Availability: If system fails, another system is available to ensure reliability.

③ Performance: Increases

Types of Distributed D.B.

Two types

① Homogenous

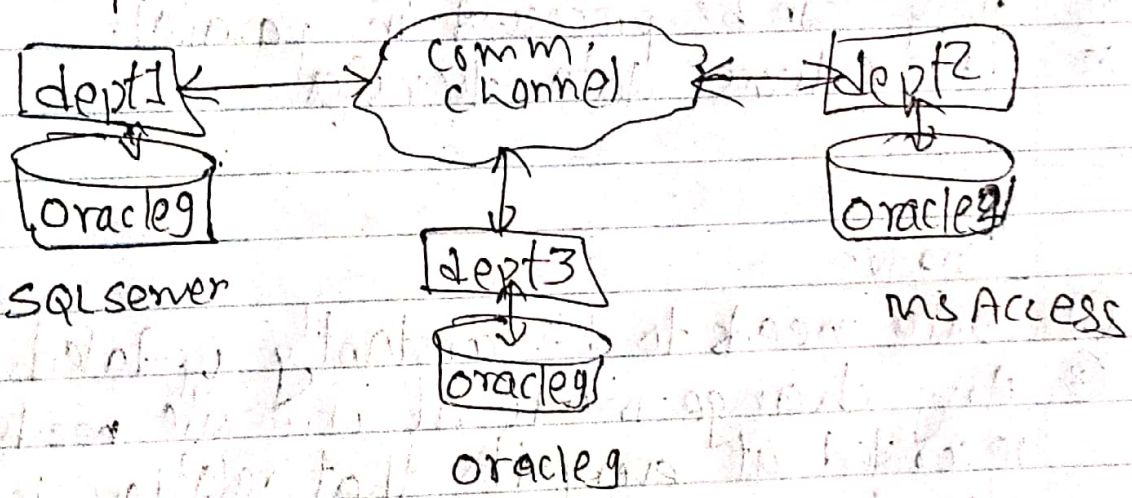
→ All sites ~~are~~ have identical software.

→ Are aware of each other and agree to cooperate in processing user request.

→ Easy to handle.

→ Each sites surrenders part of its autonomy in terms of right to change schema or software.

→ If some changes ~~are~~ made in one department then it would update other department also.



② Heterogenous

→ Different software at diff. sites.

→ Difference in software is major problem for query processing

→ Diff. in software is major problem for transaction processing

Distributed Data storage

2 ways in which data can be stored.

① Replication

→ System maintains multiple copies of data, stored in different sites.

→ If entire database is available at all sites, it is fully redundant database.

Advantage

① Increases Availability of data at diff. sites.
② query rep can be processed in parallel.
③ It reduces data transfer.

Disadv.

① Data needs to be constantly updated.
② Any change made at one site needs to be recorded at every sites that relation is stored or else it may lead to inconsistency.
③ concurrency control becomes complex.

Fragmentation

→ Division of relation R into fragments $r_1, r_2, r_3, \ldots$ which contain sufficient information to reconstruct relation R.

→ In this, Relation R is divided into smaller parts and stored in different sites where they are required.

→ It doesnot ~~to~~ create copies of data, consistency is not a problem.

Fragmentation can be done in 2 ways:

① Horizontal
- Splitting by rows
- allows parallel processing on fragments of a relation
- allows a relation to be split so that tuples are located where they are most frequently accessed.

जस्तो row 1-10 एउटा मा, 11-20 अर्कोमा)

Union को use गर्नुपर्छ, to retrieve data.

② Vertical
- Splitting by columns
- allows tuples to be split so that each part of tuple is stored where it is most freqty accessed
- → allows parallel processing on a relation.
- → column हुटा हुटै ठाउमा
- → To retrieve data, join को concept use गर्ने।
- Join lossless हुनपर्छ।

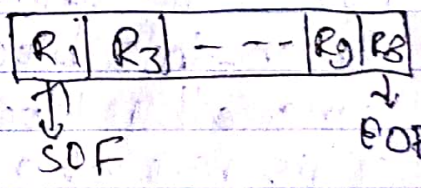
chapter - 8

→ File is collection of records.
→ FO is logical relationship among various records. This method defines how file records are mapped onto disks block.

Types

① Sequential
→ files are stored sequentially. It can be implemented by two ways:

(A) File File Method:
→ In this, record are stored in a sequence, i.e. one after another.
→ record will be inserted in the order in which they are inserted into tables.
→ In case of updating or deleting of any records the record will be searched in memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



Insertion of new record
→ new record is placed at the end of file.

② sorted method.
→ new record is always inserted at the file's end and then it will sort the sequence in ascending or descending order based on primary or any other key.

Advantage

① contains a fast and efficient method for huge data
② files can be easily stored in cheaper storage like magnetic tape.
③ simple in design. requires no much effort to store data.
④ used for report generation or statistical calculation.

Disadv

① It will waste time as we cannot jump on a particular record that is required but we have to move sequentially
② sorted file method takes more time & space

② Heap FO

→ works with data blocks
→ data are inserted at file's end and does not require the sorting and ordering.
→ When data block is full, new record is stored in some other blocks.
This new data block need not to be very next data block, but it can select any data block in memory to store new records.
→ It is DBMS responsibility to store and manage new records.
→ To search, update or delete, we need to traverse the data from starting of file till we get requested record

Adv	Disadv
→ useful for bulk insertion.	time consuming.

③ Hash functions

→ effective technique to calculate direct location of a data record on the disk without using index structure.

→ Hashing uses hash function with search keys as parameter to generate address of a data record.

Hash organization

- Bucket – hash file stores data in bucket format.
  - is considered as unit of storage.
  - stores one complete disk block.

Hash Function:- h.f. h is a mapping function that maps all set of search keys K to the address where actual records are placed.

→ It is a function from search keys to bucket address.

Types

① Static Hashing

→ when search-key is provided, h.f. always computes same address.

e.g. if mod-4 h.f. is used, it will generate only 5 values.

→ O/p address shall always be same for that function.

→ No. of buckets provided remains unchanged at all time.

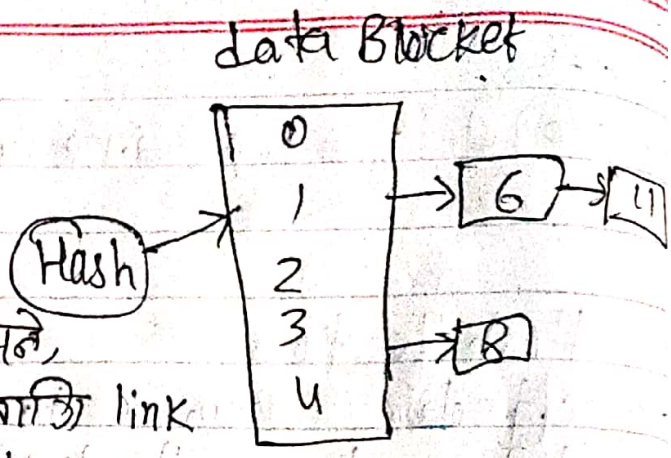
Bucket overflow:

→ It is kla collision

Remedies

(A) Overflow chaining

यदि bucket full भयो भने, same hash result को लागि link गर्दै पछाडी थप्दै जाने।



→ This mechanism is called closed Hashing.

(B) Linear probing

→ hash f. ले same address generate गऱ्यो भने र त्यो add. मा already data छ भने next free bucket खोज्दै जाने DSA तीर गरेको जस्तै

→ This is called open hashing.

खोज्दै जाने

(2) Dynamic hashing

→ static मा bucket fix हुन्छ so collision पनि हुने

→ So, यसमा data buckets dynamically added हुँदै जान्छ।

→ also kla extending hashing.

B+ tree

→ advanced method of Indexed Sequential Access Method
→ uses tree-like structure to store records in File.

→ All records are stored only at leaf node.
→ Intermediate node.. act as index to the leaf node.

Pros

① searching becomes very easy.
② Traversing through the tree structure is easier & faster.
③ Size of B+ tree has no restrictions. no. of record can increase or decrease and B+ tree can grow or shrink.
④ Is a balanced tree-structure so, insert/delete/update doesnot affect performance of tree.

Cons

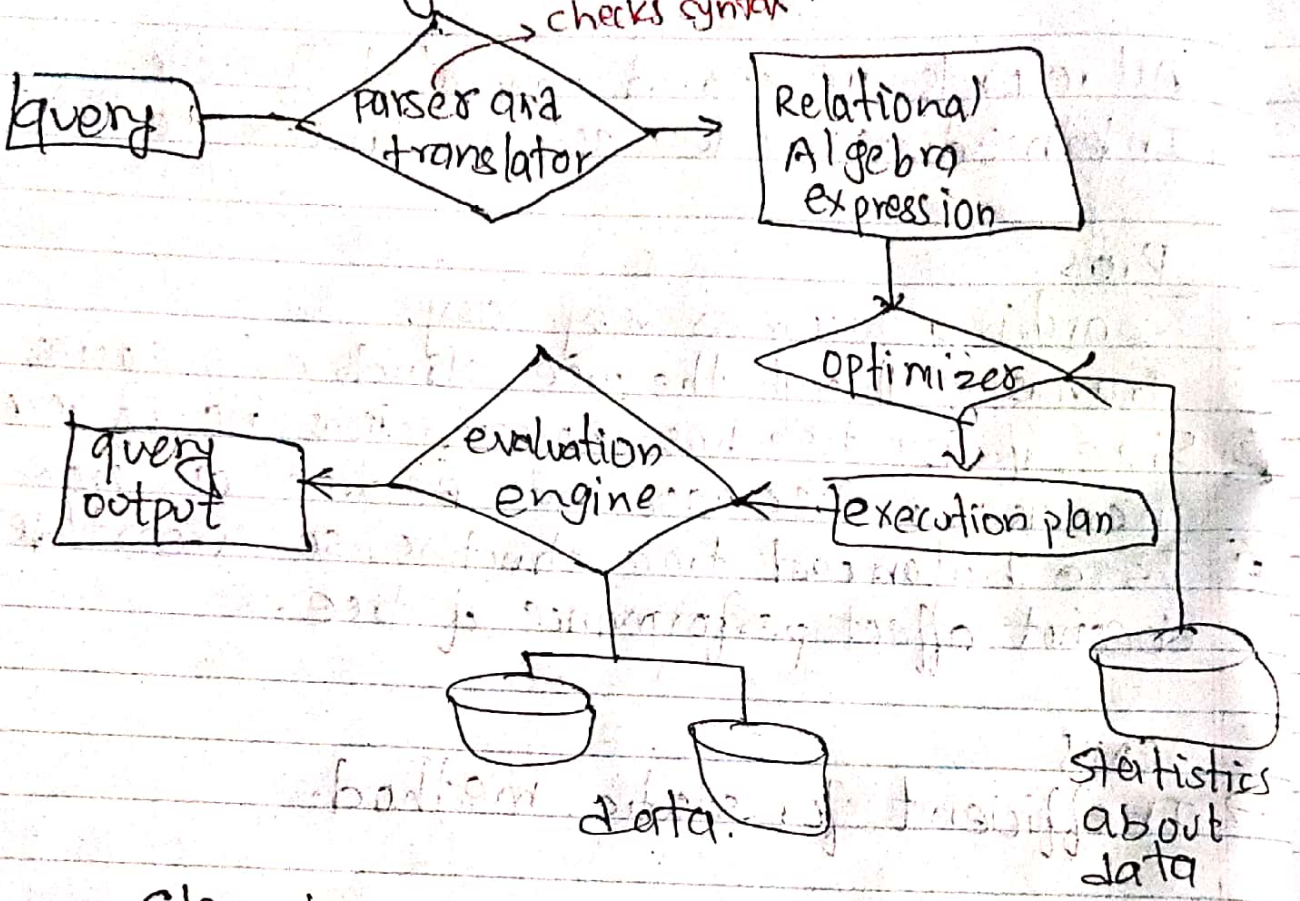
→ Inefficient for static method.

ISAM

Query processing

Query processing

(POE) Parsing translation Optimization Evaluatio



Steps in query processing

① $\sigma_{salary>5000}(\Pi_{salary}(Employee))$

② $\Pi_{salary}(\sigma_{salary>5000}(Employee))$

Security

Physical level

→ Protection of equipment from floods, power failure
→ ~~Prot~~ from theft, erasure, physical damage.
→ ताल्या चाबी use गर्ने।
→ चोरी हुनलाइको याहा दिने software use गर्ने।

Human level

(1) Lock Based protocol

→ One way to ensure isolation is to require that data items be accessed in a mutually exclusive manner, i.e. while one transaction is accessing a data item, no other transaction can modify that data item.

→ To implement this requirement, most common method is to allow a transaction to access a data item only if it is currently holding a lock on that item.

modes of lock

① shared (S) lock mode:
If transaction Ti has obtained a shared-mode-lock on item Q, then Ti can only read, but cannot write Q.

→ Transaction requests shared lock on Q by executing lock-S(Q) instruction.

② Exclusive (X) mode
In this mode, Transaction Ti can both read and write Q.

→ Transaction requests exclusive lock on Q by executing lock-X(Q) instruction.

→ Every transaction requests a lock in an appropriate mode on data item Q, depending on the types of operations that it will perform on Q.

Transaction requests → concurrency-control Manager grants lock to transaction

	S	X
S	TRUE	False
X	False	False

lock-compability matrix.

(2)

Hence, shared mode is compatible only with shared mode.

→ If any transaction holds an X-lock on item, no other transaction may hold any lock on that item.

→ If a lock can't be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released the lock and is then granted.

→ Transaction can unlock a data item Q by instruction unlock(Q).

→ A transaction must hold a lock on data item as long as it access item.

e.g.

```
T: lock-S(A)
   Read(A)
   unlock(A)
                lock-S(B)
   read(B)
   unlock(B)
   Display(A+B)
```

A locking protocol is set of rules followed by all transactions while requesting and releasing locks.

→ locking protocol restricts the set of possible schedule.

→ locking protocol must ensure serialization.

e.g. let value of account A & B are 100 & 100 resp. T_1 transfer 20 from A to B. T_2 display sum of A and B.

(3)

Now, Here, T_1: lock-X(A) read(A) A = A-20 write(A) unlock(A) lock-X(B) read(B) B = B+20 write(B) Unlock-X(B)

T_2: lock-S(A) read(A) unlock(A) lock-S(B) read(B) unlock(B) display(A+B)

→ If T_1 and T_2 are executed serially, we get correct result
→ But, concurrent execution of T_1 and T_2 may result incorrect value like:

T_1	T_2	concurrency control manager
lock-X(A)		grant-X(A, T_1)
read(A)		
A = A-20		
write(A)		
unlock(A)		
	lock-S(A)	grant-S(A, T_2)
	read(A)	
	unlock(A)	
	lock-S(B)	grant-S(B, T_2)
	read(B)	
	unlock(B)	
	display(A+B)	
lock-X(B)		grant-X(B, T_1)
B = B+20		
write(B)		
unlock(B)		

fig: schedule -x

(4)

* lock-based cntd...

→ Assume that unlocking is delayed at the end of transaction. Then,

T_1 becomes

```
T3 : lock-X(A)
     read(A)
     A = A-20
     write(A)
     lock-X(B)
     Read(B)
     B = B+20
     write(B)
     unlock(A)
     unlock(B)
```

T_2 becomes

```
T4 : lock-S(A)
     read(A)
     lock-S(B)
     read(B)
     display(A+B)
     unlock(A)
     unlock(B)
```

→ with T_3 and T_4, we can't get schedule X as above and for any other schedule, it produces correct value like:

T_1	T_2
lock-X(A)	
read(A)	
A = A-20	
Write(A)	
lock-X(B)	
read(B)	
B = B+20	
write(B)	
unlock(A)	
	lock-S(A)
	read(A)
unlock(B)	
	lock-S(B)
	read(B)
	display(A+B)
	unlock(A)
	unlock(B)

(5) dirty read भयो भने cascade rollback हुन्छ.

Two-phase locking protocol.

→ This protocol requests that each transaction in a schedule will be two phased, growing phase and shrinking phase.

→ In growing phase, transaction can only obtain like lock but can not release any lock.

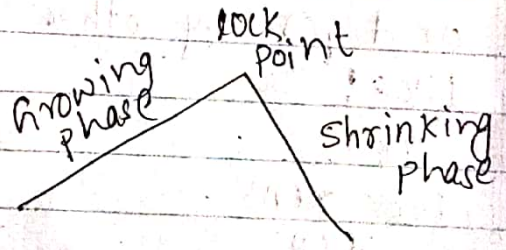
→ In shrinking phase transaction can only release locks but can not obtain any lock.

→ Transaction can perform read/write operation both in growing & shrinking phase.

→ Ensures C.S./V.S., the order of serializability is the order in which, transaction reaches lock point

→ May generate unrecoverable schedule and cascading rollback.

→ Do not ensure freedom from deadlock.



S

T_1	T_2
lock-S(A)	
Read(A)	
	lock-X(B)
	read(B)
	write(B)
lock-X(B)	
	lock-X(A)

(6) Schedules

When transactions are executing concurrently in an interleaved fashion, then the order of execution of operations from various transaction is known as schedule.

A schedule specifies the chronological order in which the operations of concurrent transaction are ~~occurs~~ executed.

Serial schedule

→ Transaction are executed non-interleaved i.e. serial schedule is one in which no transaction starts untill a running transaction has ended.

e.g. Suppose there are two transaction T_1 and T_2 which have some operations. If it has no interleaving of operations, then there are following two possible outcome:

(a) Execute all operation of T_1 which was followed by operations of T_2.

(b) Execute all operation of T_2 which was followed by operations of T_1.

T_1	T_2
read(A);	
A := A - N;	
write(A);	
Read(B);	
B := B + N	
write(B);	
	read(A);
	A := A + M;
	write(A);

S

T_1	T_2	T_3
R(X)		
W(X)		
com.		
	R(X)	
	W(X)	
	com.	
		R(Z)
		W(Z)
		com.

T_1	T_2
	Read(A)
	A := A + M;
	write(A);
Read(A);	
A := A - N;	
write(A);	
Read(B);	
B := B + N	
write(B)	

(2) Non-serial

If interleaving operations is allowed, then there will be non-serial schedule.

It contains many possible orders in which the system can execute the individual operations.

e.g.

T_1	T_2
read(A);	
A := A-N;	
	read(A);
	A := A+M;
write(A)	
read(B)	
	write(A);
B := B+N;	
write(B)	

T_1	T_2
read(A);	
A := A-N;	
write(A);	
	read(A);
	A := A+M;
	write(A)
read(B);	
B := B+N;	
write(B)	

(3) Serializable schedule.

The serializibility of schedules is used to find non-serial schedules that allow the transaction to execute concurently without interferring with one another.

A non-serial schedule isill be serializable if its result is equal to the result of its transaction executed serially.

(6)

Conflict serializable schedule.

→ a schedule is called conflict ~~serla~~ c.s. if after swapping of non-conflicting operations (instruction) it can transform into a serial schedule.

→ schedule will be c.s. if it is conflict equivalent to a serial schedule.

Conflicting operations:

Two operations become conflicting if all cond^n satisfy:

1. Both belong to separate transaction
2. They have the same data item.
3. They contain at least one write operation.

Conflict equivalent

→ one ~~transact~~ schedule can be transformed to another by swapping non-conflicting operations.

Two schedules are conflict equivalent iff:

1. contain same set of transaction
2. If each pair of conflict operations are ordered in same way.