Database Management system (DBMS)

DBMS (Basics)

→ Data : Raw and isolated fact about an entity (recorded)
eg : text, image, video, document, map etc.

→ Information : Processed, meaningful and useful data.

→ data $\xrightarrow[\text{processing, usability inc}\uparrow\uparrow]{}$ information

→ Database : collection of similar / related data.

→ DBMS : s/w used to create, manipulate and delete database.

Disadvantages of file system

① Data Redundancy : single data being found at multiple places (data scattering).

② Data inconsistency : one data at multiple places being changed at one place may / may not change other value resulting in inconsistency

③ Difficulty in accessing data : problem being solved by DBMS queries.

④ Data isolation : isolated data can't be handled by OS as they are dependant.

⑤ Security problems ↑

⑥ Atomicity problem : data comes in inconsistent state when OS gets closed due to some reason.

⑦ Concurrent access anomalies: only one user can access the data at once but that can't be a problem in DBMS.

⑧ Integrity problem: automatic triggers that can't be allowed can ~~be~~ sustain in DBMS.

Types of ~~OLAP~~ Database Management system.

OLAP	OLTP
→ online analytical processing	→ online transaction processing
→ holds historical data	→ hold day-to day operational data
→ used in decision making, getting prediction	→ used for day to day operations.
→ subject oriented	→ application oriented.
→ large size (TB, PB)	→ small size (MB, GB)
→ CEO, MD, GM deals with it	→ clerks & manager deal with it
→ only read operation	→ read & write data

Why OLAP and OLTP separated?

→ 95% data was historical data that was not much in use

→ 5% data is day to day use needed.

→ if all data is stored together, then speed for day to day needed data gets decreased

→ thus we divide our whole data into OLAP and OLTP.

ER diagram (Entity Relationship diagram)

- → A non technical design method works on conceptual level based on perception of real world.
- → consist of collection of basic objects called entities and relationship among these objects & attributes define property
- → free of ambiguities
- → easy to understand (as diagram based)

Entity: real world objects distinguishable from each other on the basis of values of attributes.

Entities

← →

tangible (physically exist)

intangible (virtually exist)

Entity set: collection of same ~~sets~~ entities.

→ schema: structure consisting of entity set

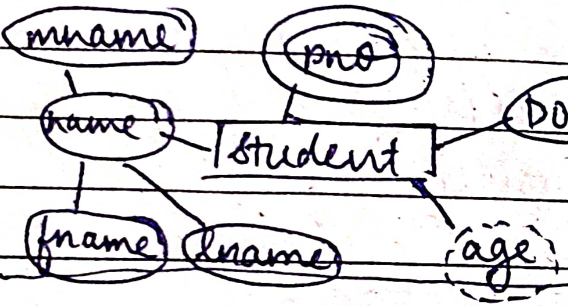
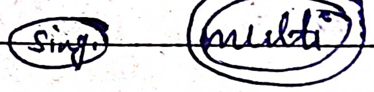
→ entity set is represented by rectangle

~~→ attributes:~~

→ attributes in ER diagram

- → units that describes properties in an entity
- → for each attribute, there is a permitted domains
- → represented by ovals.

Types of attribute

simple-composite	single-multivalued	stored/derived
→ simple can't be divided further and is represented by oval. → composite can be further divided into simple attributes, rep. by oval connected to oval 	→ only one instance at a time: single valued. → more than one instance at a time: multivalued → single oval: single valued → multivalued: double ~~valued~~ oval 	→ attributes that values are known and stored in database → derived attri-butes are derived from stored attributes → derived attributes are drawn using dotted oval

Relationship

→ association between 2 or more entites of same or different entity set.
→ not represented in ER diagram.
→ represented using a row in a table

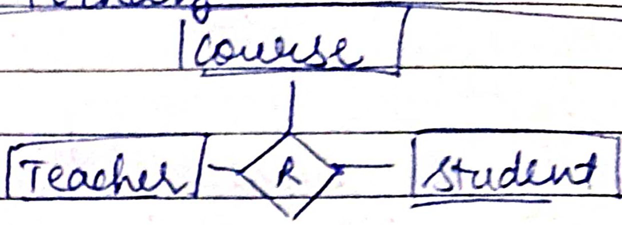
Relationship set/type

→ set of similar type of relationship
→ represented using diamond in ER diagram.
→ every relationship type has 3 components
* Name
* Degree
* Cardinality ratio / Participation constraints

Degree of relationship
→ no. of participating entity set.

Binary

Ternary

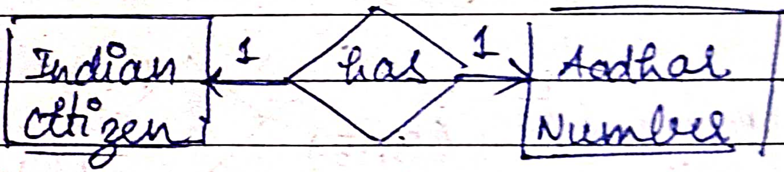


→ most of relationships are binary relationships
→ however there may arise some cases when we have more than 2 relationships.

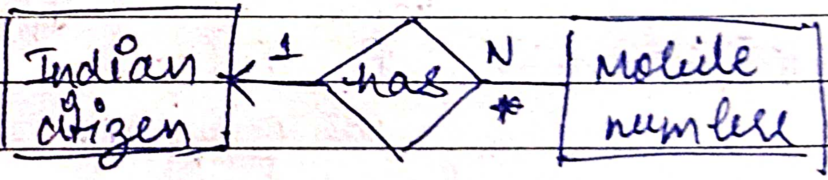
Cardinality Ratio / Mapping cardinality

→ number of entities to which other entity can be related via a relationship.
→ it is more useful in binary relationships

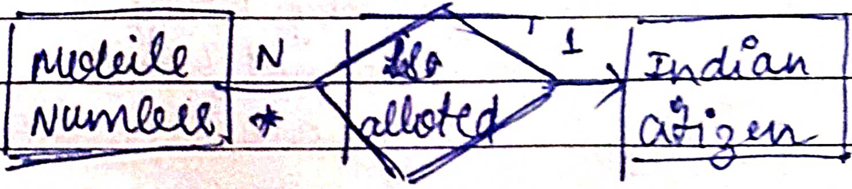
* One to one



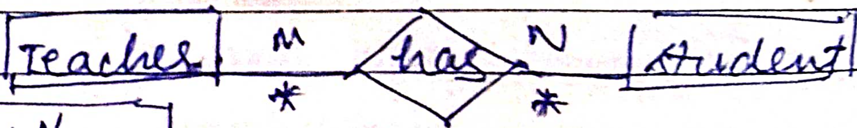
* one to many

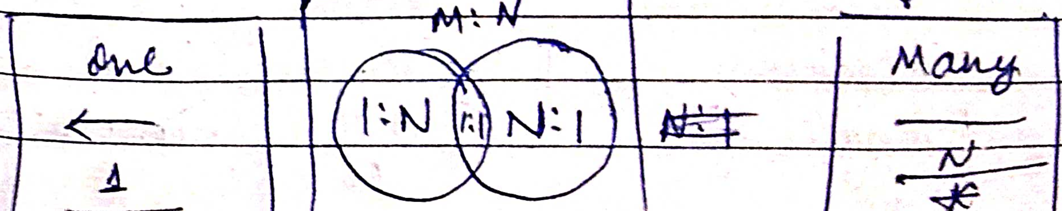


* many to one



* many to one





Participation constraints.

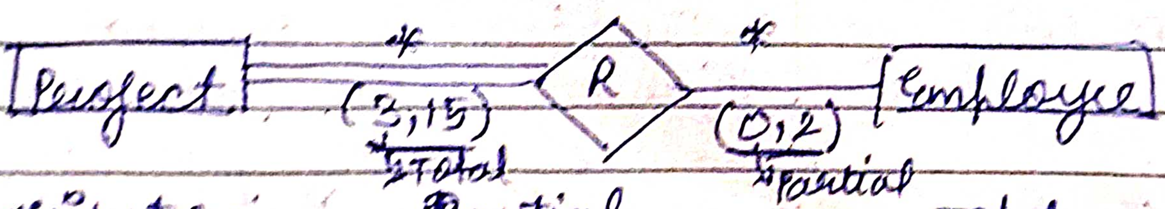
→ does any entity needs to participate in any relationship to be able to exist in system?

→ specifies whether the existance of an entity depends on its being related to another entity via relationship type.

→ constraints specify minimum and maximum number of relationships instance that each entity can participate in.

* Max cardinality defines maximum number of times an entity occurrence participate in relationship.

* Min cardinality defines minimum number of times an entity occurrence participate in relationship.



Participation:	Partial Participation	Total Participation
	* if any of the entity does not participate in relationship * that side of relationship is partial * single line.	* If all entity participate in relationship * that side of relationship is total. * double line (not recommended to use)

Functional Dependencies

→ if there is a functional dependancy from α to β in a relational table, then we can search value of β from relational table using α.

→ denote: $fd(\alpha) \rightarrow \beta$ and $\alpha \subseteq R$ & $\beta \subseteq R$

→ if $t_1[\alpha] = t_2[\alpha]$ then $t_1[\beta] = t_2[\beta]$

$$\underset{\text{determinant}}{\alpha} \longrightarrow \underset{\text{depandant}}{\beta}$$

→ FD can be breaked in 2 types: ① trivial ② non trivia

① trivial functional dependancies.

→ if $\alpha \rightarrow \beta$ and $\beta \subseteq \alpha$, then dependancy is trivial.

→ $AB \rightarrow A$

② non trivial functional dependancies

→ If $\alpha \rightarrow \beta$ and $\beta \not\subseteq \alpha$, then dependancy is non trivia

→ $AB \rightarrow AB\textcircled{C}$ → new attribute.

Problem, which options are correct:

✓ Ⓐ A → BC   ✗ Ⓒ C → DE

✓ Ⓑ DE → C   ✓ Ⓓ BC → A

R

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

Ⓐ is correct.   Ⓓ is correct.

Ⓑ is also correct

Shortcut to find whether dependancy is true

① check value of α.

② if all values of α are unique, dependancy is true

③ check value of β

④ if all values of β are same, dependancy is true

Question : which optoinsare valid

X	Y	Z
1	4	2
1	5	3
1	6	3
3	2	2

Ⓐ $XY \to Z$ && $Z \to Y$ ✗
Ⓑ $YZ \to X$ && $Y \to Z$ ✓
Ⓒ $YZ \to X$ && $X \to Z$ ✗
Ⓓ $XZ \to Y$ && $Y \to Z$ ✗

Question: which options are valid

A	B	C
1	2	4
3	5	4
3	7	2
1	4	2

Ⓐ $A \to B$ && $BC \to A$ ✗
Ⓑ $C \to B$ && $CA \to B$ ✗
Ⓒ $B \to C$ && $AB \to C$ ✓
Ⓓ $A \to C$ && $BC \to A$. ✗

attribute closure / closure of attributes set / closure of attribute set.

if $A \to B$ and $B \to C$ then $A \to BC$
these A is called closure set $(A)^+$.
$(A)^+$ here = ABC

Question R(ABCDEF)

$A \to B$
$C \to DE$
$AC \to F$
$D \to AF$
$E \to CF$.

$(D)^+ = DAFB = \boxed{ABDF}$
$(DE)^+ = DAFBCE = \boxed{ABCDEF}$

Armstrong Axioms / Rules

Primary Rule (RAT)
① Reflexivity : $Y \subseteq X \Rightarrow X \to Y$
② Augmentation: $X \to Y \Rightarrow XZ \to YZ$.
③ Transitivity : $X \to Y$ && $Y \to Z \Rightarrow X \to Z$

Secondary Rule.
① Union: $X \to Y$ & $X \to Z \Rightarrow X \to YZ$
② Decomposition, $X \to YZ \Rightarrow X \to Y$ & $Y \to$
③ Pseudotransitivity
$X \to Y$ & $WY \to Z$
then $WX \to Z$

Equivalence of functional dependancies

F) R(ACDEH)

F:
$A \rightarrow C$
$AC \rightarrow D$
$E \rightarrow AD$
$E \rightarrow H$

G) → $A \rightarrow CD$
$E \rightarrow AH$

Which one is true?
(a) $F \subseteq G$
(b) $F \supseteq G$
(c) $F = G$
(d) $F \neq G$

$(A)^+ \rightarrow ACD$
$(AC)^+ \rightarrow ACD$
$(E)^+ \rightarrow EAHCD \rightarrow F \subseteq G$

$(A)^+ = ACD$
$(E)^+ = EADHC$ → $G \subseteq F$

Q. R(PQRS)

X: $P \rightarrow Q$ ✓
$Q \rightarrow R$ ✗
$R \rightarrow S$ ✓

$(P)^+ = PQRS$
$(Q)^+ = Q$
$(R)^+ = RS$
thus $X \subseteq Y$

Y: $P \rightarrow QR$ ✓
$R \rightarrow S$ ✓

$(P)^+ = PQRS$
$(R)^+ = RS$
thus $Y \subseteq X$

Which one is true
~~a)~~ $X \subset Y$
✓b) $Y \in X$
~~c)~~ $X = Y$
d) $X \neq Y$

Q R(ABC)

F: $A \rightarrow B$ ✓
$B \rightarrow C$ ✓
$C \rightarrow A$ ✓

$(A)^+ = ABC$
$(B)^+ = BCA$
$(C)^+ = CAB$
~~$F \subseteq G$~~

G: $A \rightarrow BC$ ✓
$B \rightarrow A$ ✓
$C \rightarrow A$ ✓

$(A)^+ = ABC$
$(B)^+ = BCA$
$(C)^+ = CAB$
$G \subseteq F$

$F = G$

Q. R(VWXYZ)

F: $W \rightarrow X$ ✓
$WX \rightarrow Y$ ✓
$Z \rightarrow WY$ ✓
$Z \rightarrow VX$ ✗

$(W)^+ = WXY$
$(WX)^+ = WXY$
$(Z)^+ = $ ~~Z~~WXY

$F \nsubseteq G$

G: $W \rightarrow XY$ ✓
$Z \rightarrow WX$ ✓

$(W)^+ = WXY$
$(Z)^+ = ZWYX$ ✓

$G \subseteq F$

Irreducible set of functional dependancies (canonical form)

→ if any functional dependancies contains some redundant elements, then we must remove it

$R(WXYZ)$

$X \to W$	$X \to W^*$ ✓	$(X)^+ = XW$
$WZ \to XY$	~~$WZ \to X$~~	$(X)^+ = X$ (ignoring first rel)
$Y \to WXZ$	$WZ \to Y^*$ ✓	this shows that the dependancies
decomposed in diff relations	~~$Y \to W$~~	is essential
	$Y \to X^*$ ✓	$(WZ)^+ = WZXY$) this means
	$Y \to Z^*$ ✓	$(WZ)^+ = WZYX$ $WZ \to X$ is redunda

Rule

① Apply decomposition rule to each and every depend

② Find and compare closure of ~~any~~ any dependan with and without dependancies

③ If it is same, then the dependancies is non essential and viceversa

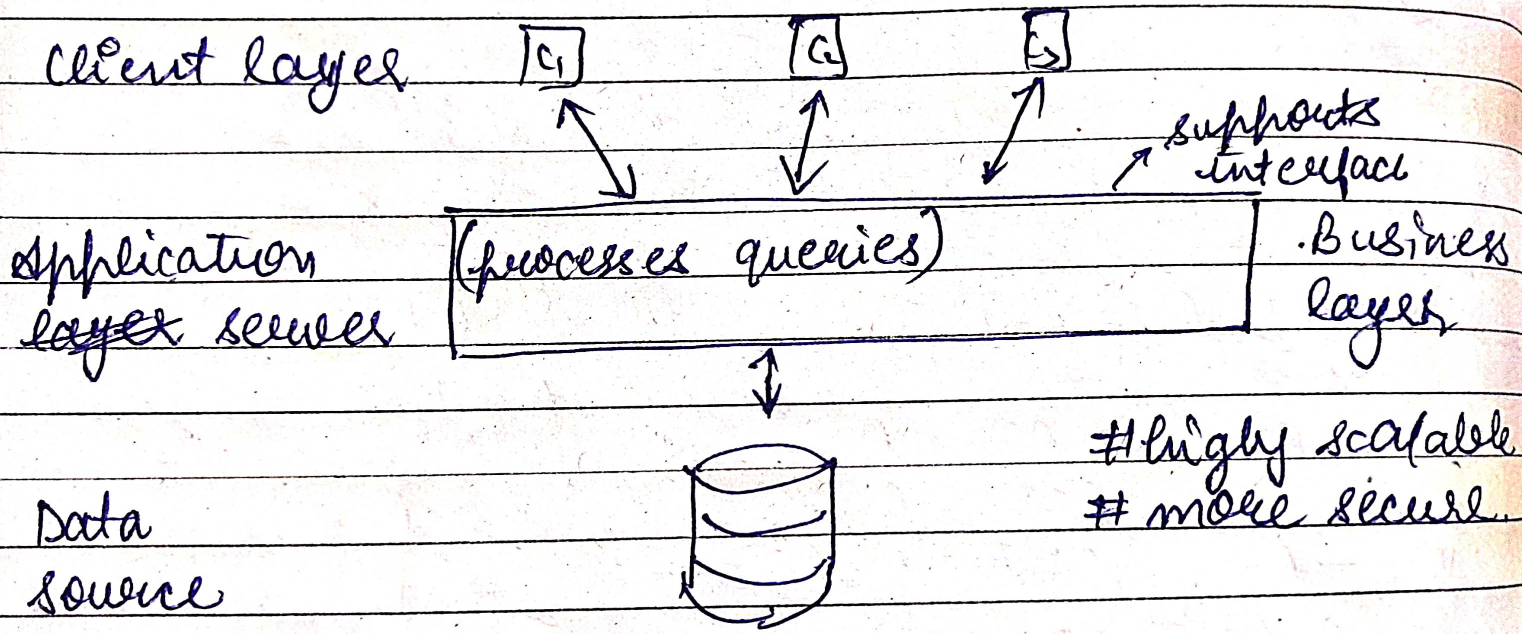
Q. $R(ABCD)$ → find minimal canonical form

$A \to B$	$A \to B$ ✓	$A^+ = AB$	$A^+ = A$
$C \to B$	$C \to B$ ✓	$C^+ = CB$	$C^+ = C$
$D \to ABC$	$D \to A$ ✓	$D^+ = DABC$	$D^+ = DBC$
$AC \to D$	~~$D \to B$~~	$D^+ = DBAC$	$D^+ = DACB$
①	$D \to C$ ✓	$D^+ = DCAB$	$D^+ = DAB$
	$AC \to D$ ✓	$AC^+ = ACDB$	$AC^+ = ACB$

$A \to B$	$(AC)^+ = ACDB$	Minimal cover
$C \to B$	$(A)^+ = AB$	→ $A \to B$
$D \to A$	$(C)^+ = B$	$C \to B$
$D \to C$		$D \to AC$
$AC \to D$		$AC \to D$

Date : Page :

3 tier architecture (3 level architecture)



#highly scalable
more secure.

3 schema architecture
→ Schema: structure (what is structure to store data)

1. Primary key.

→ used to identify one and only instance of entity uniquely

→ most suited key out of a set of keys.

→ selection is based on requirements developers

2. Candidate key.

→ attribute / set of attributes uniquely identify tuple

→ remaining attribute except primary key are considered as candidate key.

3. super key.

→ set of attributes that can uniquely identify tuple.

→ candidate key is superkey but superkey can't be candidate.

4. Alternate key.

→ candidate key other than primary key is called alternate key.

5. Foreign key.

→ If an attribute can only take values present as values of some other attributes, it'll be foreign key to attributes to which it refers.

Normalization

→ remove redundancy / reduce redundancy & duplicate for a relation / set of relations

→ redundancy causes insertion, deletion and updation anomalies.

1) First Normal Form (1NF)

→ table should not contain any multivalued attributes.

→ a table is in 1NF if every attribute in relation is single valued attribute

Closure Method

→ used to find all ~~prima~~ candidate keys in a table

R(A BC D)

$\{A \to B, B \to C, C \to D\}$ CK = {A} ~~AB~~

$A^+ = ABCD$ $B^+ = BCD$ $C^+ = CD$ $D^+ = D$

AB = ABCD → (But AB must be minimal so AB cannot be ~~prim~~ candidate key)

2) Second Normal Form (2NF)

→ table must be 1NF.

→ table must not contain partial dependencies.

Partial dependencies: non prime attributes is dependant on proper subset of candidate key.

AB → CK
but A → C thus there is partial dependancy

R(ABCDEF)

FD $\{C \to F, E \to A, EC \to D, A \to B\}$

CK: ~~FADB~~ EC = FADB

$EC^+ = ABCDEF$

3) Third Normal Form (3NF)

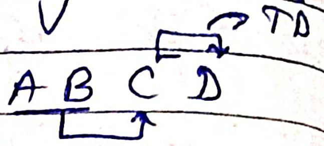
→ a table is in 3NF, if there are no transitive dependencies for non prime attributes

→ and table should be in 2NF.

transitive dependancy.

→ if any non prime attribute is being determined by another non prime attribute

* non prime attribute : not participating in CK.



R(ABCD)   FD: AB → C   C → D

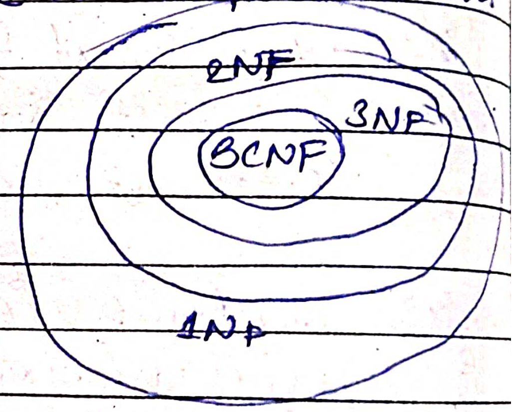
CK = AB   PA = A, B   NPA = C, D

This table is not in 3NF

check for 3NF →

① LHS must be CK or SK. ② RHS must be prime attribute.

4. Boyce codd normal form (BCNF)
- → special case of 3^{nd} NF.
- → Table must be in 3NF.
- → every functional dependencies must have candidate key on LHS.
- → candidate can only determine all dependencies



SQL.
- → most popular & widely used query language.
- → domain specific & not general purpose.
- → can define database, modify, insert and delete elements/table/database
- → based on relational algebra

Parts of SQL
① DDL (data definition language)
- → define a table in database
- → can create, define, delete, modify schema of table

eg.

② Data Manipulation Language (DML)
- → provide ability to query info from DB,

→ insert, delete and modify tuple of database
→ works off instance

③ TCL (Transaction control language):
includes commands for beginning and ending of transaction
① rollback ② commit ③ checkpoint.

~~Int Dat~~ DQL (Data query language)
→ for retrieving a data from DB.
→ use of SELECT operation,. FROM operation and WHERE operation

I/p — | SQL query | — O/p (one table)
(at least one)

SELECT (column name) from (Relation name) where (Condition)
→ SQL is case insensitive.
→ SQL supports duplication in both I/p and O/p.

SELECT clause.
→ pick column reqd in result of query out of all columns in relation.
→ all columns (*)
→ supports simple arithmetic operation (no change in database)

select all ~~d~~etails from branches.
>> SELECT * from BRANCHES;
~~>> SELECT~~
find the name of all customer who have account
>> select c.name from depositer;
^distinct

find each l.no along with amount
>> select lno, amount from loan ;

find all account no. and balance with 6% year
>> select account, balance *1.06 from account in

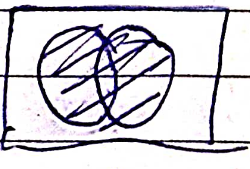
where clause.
- → specify condition/ predicate.
- → can allow comparision operation (<, <=, >, >=,
- → logical operators OR ~~and~~, AND and NOT
- → between clause

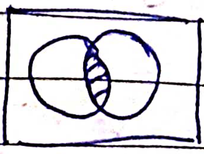
Q. all a.no where balance is less than 1000
>> SELECT ano. from account where balance < 10

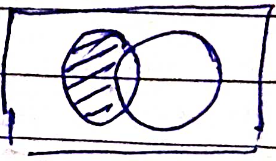
Q. find b.name which is situated in delhi and having assets < 100,000
>> select b.name from branch where B.city = 'Delhi' and ~~[illegible]~~ assets < 100,000;

Q. Find branch and A.no where balance is >=1 and <= 1000
>> Select brname, A.no from account where balance between 100 and 1001

SET operator.
→ Union, Intersect, except, minus, set diff


AUB


A∩B


A−B

→ union intersection & minus automatically removes duplicate values.

→ If we want to retain duplicates even after operation we write all after operation

① Find all the customer name having an account or loan or both

⇒ select cname from depositor union select cname from Borrower

* if we want not to eliminate duplicates, we use union all.

② Find all the customer having both account and loan

⇒ Select cname from depositor intersect select caname from borrower.

③ Find all the customer having an account but not loan

⇒ select cname from depositor minus select cname from borrower.

→ set operations are only applicable to 2 tables if they are compatible.

→ they must have same no. of collumns and same domain

Caertesian Product ('x')

→ Query on multiple relation

→ combine each and every tuple from first relation with every possible tuple of second relation

① Find name of customers having branch in North delhi

⇒ Select cname from Account, Depositor where bname = 'N D'

and account.Ano = Depositor. Ano

match

② Find loan no and amount of loans which are from branch situated in delhi

>> select lno, amount from loan, branch where B.city = 'Delhi' and loan.bname = branch.bnam

③ find customer name which have loan from branch with assets < 1,00,000

```
>> select cname
   from branch, loan, Borrower
   where assets < 100000
   and    Branch.Bname = loan.Bname
   and loan.lno = Borrower.lno.
```

Natural join.

→ works same as cartesian product but considers only those pair of tuples with same value on attributes that appear of schema

above queries with natural join

①

```
select cname
from account natural join depositor
where branch name = 'N_D'
```

```
② select loan_no, amount
   from loan natural join Branch
   where b.city = "Delhi"
```

```
③ select cname
   from Branch naturaljoin loan natural join borrower
   where assets < 100000
```

→ no prefix required
→ commutative in nature
→ lead to data loss if attribute to be joined has different name or attribute present only 1 tab

Inner join
→ works same as cartesian product if not use "using" keyword
→ allow us to choose attribute in order to eliminate redundant tuples

Q find the name of customer having loan amount < 1000

```
select c.name
from borrower join loan using(lno)
where amount <1000
```

On keyword (more powerful than using)
→ allows us to use a general predicate over being joined relations.

Q. write a SQL query to find all customer name having loan amount <1000.

```
>> select cname
from loan [illegible] join borrower ~~where on~~
on loan.lno= borrower.lno and
~~where loan~~ amount <1000.
```

→ no use of where
→ can express any predicate
→ improves readability of query.

Outer join
→ manage data loss from inner/natural join

left outer join (⟕), right outer join (⟖)
full outer join (⟗)

left outer join
→ all the tuples from left table and matching tuples from right, in case there are no matching values, tuple is filled with null values

right outer join
→ all the tuples from right table and matching tuples from left, in cas there are no matching values, tuples is filled with null values

full outer join
→ union of left and right outer join.

Rename operator.
→ provide column of table with temporary name
→ only exist for a duration of query
→ used for comparing table with itself
→ used to provide name to column that got created as a result of operations on one or more columns

① Find account number and balance with 8% interest if bal <1000

```
select a.no, (balance) ~~fun~~ * (1.08) as bewi
from account
where balance < 1000.
```

Q. Find customer name and loan amount.
-> select c_name as customername, amount as loan_amount
from loan, borrower where loan.lno = Borrower.lno

Q. Find account no, bname and bcity
-> Select Ano, Bname as A, Bcity
from Account, Branch as A
where A.bname = B.bname.

Q. Find maximum loan amount from bank.
-> (select A. amount
from Loan as A, Loan as B
where A. amount < B. amount.) ①

-> select amount from loan minus ①

- SQL aliases are used to give temporary name to a table or column
- exist for duration of query.
- used for comparing table with itself.
- does not change anything in table.

String Operation.
- enclose string in single quotes
- equality operation may/may not be case sensitive
- operation: concatenation, substring, change case, find ~~operation~~ length
- pattern matching can be performed with strings using like operation.
- like is case sensitive
- % : match any substring
  _ : match with any character.

→ specify double comma if [illegible] single
comma in string
'doesn't' → "doesn''t"

→ if want to use special character in string, use backslash, (same with _).
"ab%c" → 'ab\%c'.

Ordering the display of tuples (Order By) clause
→ sort the results of query based on column
→ ASC and DESC

① Find the name of all branches which are situated in "Gilgit" in alphabet order.
>> select B-name from Branch where b.city = 'Gilgit' order by B-name.

→ ordering can be done by multiple attributes.

Query language.
→ Languages in which user request some info from database
→ types: procedural & non procedural
① procedural : user instruct system to perform a sequence of operations in order to produce desired result.
→ defines what to and how to retrieve data
② Non procedural : user describe desire info without giving any instruction of how to perform it.

→ Relational model is conceptual framework and RDBMS is its implementation

Relational model	RDBMS
RA, RC.	SQL
Algo	code
conceptual	Reality
Theoretical	Practical

Relational Algebra.

- → theoretical framework.
- → associated with relational model.
- → defines no. of operations and use relation as operan
- → every operator takes 1/2 table as i/p, it yields out results as table without name.
- → it is based on set theory thus no duplicates possib
- → It is procedural query language.
- → no use of english keywords.

Basic	derived
select (σ)	Natural join ($\bowtie$)
project (π)	⟕, ⟖, ~~⟗~~
union ($\cup$)	Intersection ($\cap$)
set difference ($-$)	division ($\div$)
cartesian product ($\times$)	
Rename (ρ)	

select operator. (σ)

- → unary operator so can take only one table.
- → fundamental operator.
- → find tuples/rows from table which satisfy a particular condition
- → $\sigma_{condition}$ (table-name)