T_i : write(Q)

$TS(T_i) < WTS(Q)$: write operation and no rollback (Allowed This is ignored.

$TS(T_i) < RTS(Q)$; Reject & T_i rollback.

Thomas Write Rule (Formal Definition)

- Modified version of the time-stamping-ordering protocol in which obsolete write operations may be ignored under certain circumstances.
- When T_i attempts to write data item Q, if $TS(T_i) < W\text{-timestamp}(Q)$, then T_i is attempting to write an obsolete value of {Q}.
  - Rather than rolling back T_i as the timestamp ordering protocol would have done this {write} operation can be ignored.
- Otherwise this protocol is same as the timestamp ordering protocol.
- Thomas Write Rule allows greater potential concurrency.
  - Allows some view-serializable schedules that are not conflict serializable

(10) T_1	(20) T_2
R(A)	
	W(A)
W(A)	

Obsolete write → $TS(T_1) < WTS(A)$

$10 < 20$; not allowed under TSP

But allowed under Thomas write rule.

$T_1 \Rightarrow W(A)$: ignored

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

Allowed under TWR.

→ More concurrency means this schedule is not conflict serializable, this is not allowed under TSP but allowed by Thomas Write Rule.

eg:

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

This is allowed under Thomas write rule.

Obsolete read.

(10) T_1	(20) T_2
R(B)	
	W(B)
	W(A)
W(A)	
R(A)	

$T_1 \rightarrow T_2$

Check TWR:

T_1: W(A) $TS(T_1) < WTS(A)$ — yes allowed in TWR

T_1: R(A) $\rightarrow TS(T_1) < WTS(A)$ — not in TWR.

Not TSP, Not TWR

Time Stamp protocol: ensure Serializability, deadlock free, but Starvation Possible.

Deadlock Prevention Algorithm:

(i) Wait–Die (ii) Wound–wait

Older Younger.

T_1 (10)	T_2 (20)	
R(A)		
	W(A)	Not in TSP
W(A)		But in TWR

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

Strict Time Stamp Ordering protocol = TSP + Strict Schedule.

↓

It Ensures: Serializability
Recoverability
Cascadeless (no cascading rollback)
Strict Recoverability

Q1. In which of the following lock scheme deadlock cannot occur?

Ans Conservative 2PL

Q2. Consider the following schedule: $r_1(x), r_2(y), r_2(x), w_1(z), r_1(y), w_3(y)$ $r_3(z), w_2(y), w_3(x)$.

Which of the time stamp ordering not allowed to execute schedule using Thomas Write Rule time stamping ordering protocol?

Ans $(T_1, T_2, T_3) = (10, 20, 30)$.

Q3. Consider the two statements about database transaction schedules:

I. Strict two phase locking protocol generates conflict serializable schedules that are also recoverable.

II. Timestamp ordering concurrency control protocol with Thomas write rule can generate view serializable schedules that are not conflict serializable.

Which of above statements is/are true?

Ans Both I and II.

Q4. Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?

1. 2-phase locking

II. Time stamp ordering.

Ans II Only.

Date 23/8/23

ER MODEL

ER Model: Entity-Relational Model.

→ **Entity:** Object in the real world and is distinguishable from other.

→ **Entity set:** Collection of similar entries that share the same properties.

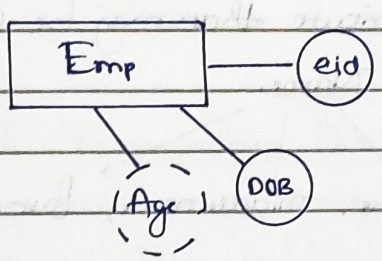
eg: Set of all people, companies.

* Entities have attribute

eg: a person has name, address.

* Entity set is a set of entities of same type denoted by a rectangle or box in ER diagram. Entity can be identified by a list of attributes which are placed in ovals.

eg:



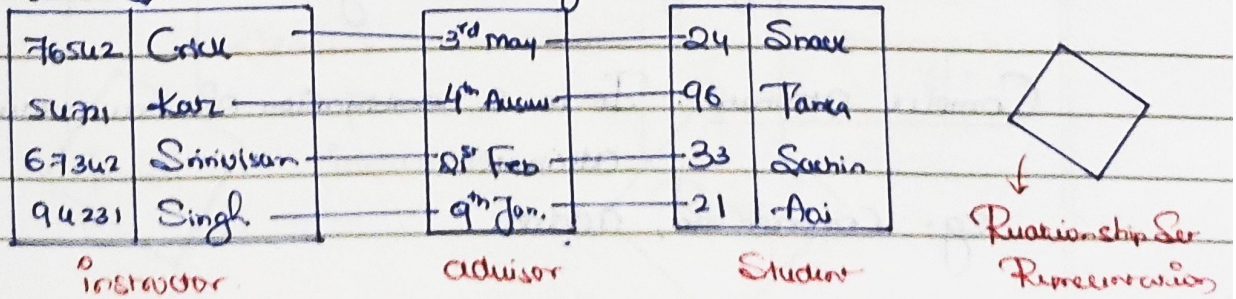
Relationship: A relationship is an association among several entities.

eg: 44553 (Peltier) — advisor — 22222 (Einstein)

44553 ↑ Student entity. advisor ↑ relationship set 22222 ↑ Instructor entity

Relationship Set: It is a mathematical relation among n≥2 entities each taken from entity sets.

* An attribute can also be property of a relationship set.
* For instance, the advisor relationship set between entity sets instructor and student may have the attribute date which tracks when the student started being associated with the advisor.



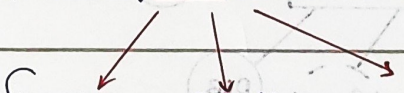
Attributes: Attributes are properties used to describe an entity.

Attribute types are:

1. Simple and Composite attributes.
2. Single valued and multivalued attributes
3. Stored and derived attributes.
4. Key attribute
5. Complex attribute
6. Descriptive attribute.

Simple attribute: Each entity has a single atomic value for the attribute. For example: Roll no. It cannot be further divided.

Composite attribute: The attribute that may be composed of several components. For example: Name



Single valued attribute: which takes one value per entity.
eg: gender, roll no.

Multi valued attribute: which takes more than one value per entity.
eg: email, phone no

Stored attribute: which does not require any updation.
eg: DOB, DOJ.

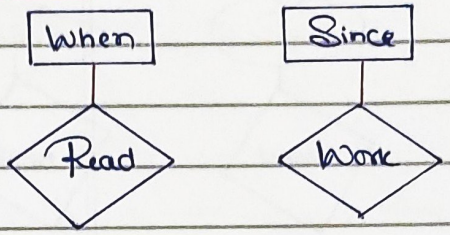
Derived attribute: the value of the attribute derived from other attributes. eg: years of service, age.

Complex attribute: It is a combination of multivalued + composite attribute.
eg: Contact no, address.

Contract no. → Composite attribute {SID + TID}
+
→ Multivalued attribute. {Sim1, Sim2}

Key attribute: which uniquely identify an entity in the entity set.
eg: roll number.

Descriptive attribute: which give information about the relationships set-s.

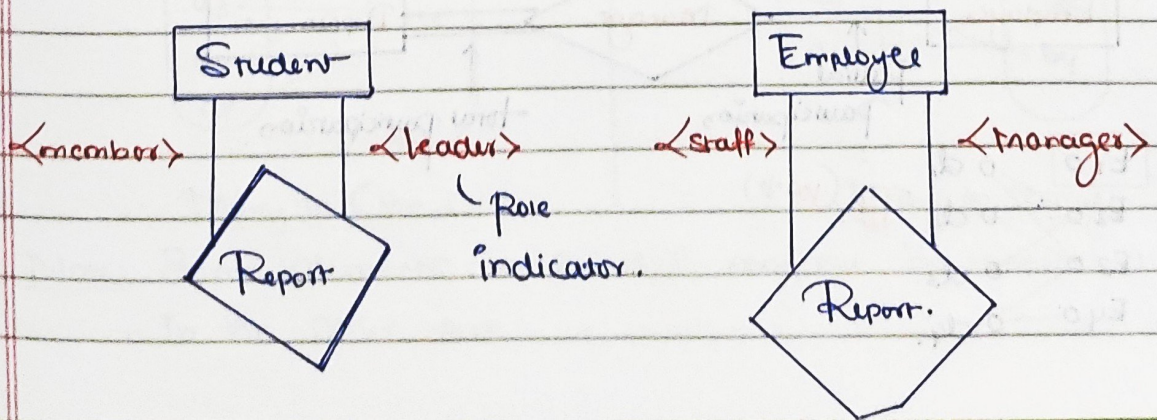


Degree of Relationship Set.
How many number of entity set participate in a relationship set.

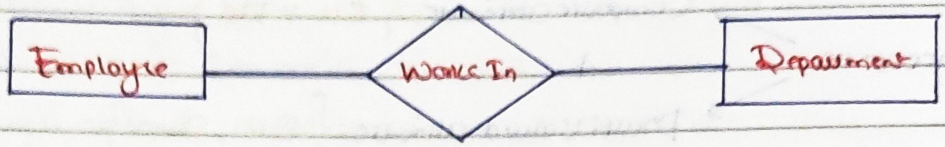
1. Unary (only one Entity set)
2. Binary (two entity set)
3. Ternary (three entity set)
4. N-ary (N-th entity set).

Degree of relationship set: Specifies the number of Entity set participate in a relationship set.
(Formal definition).

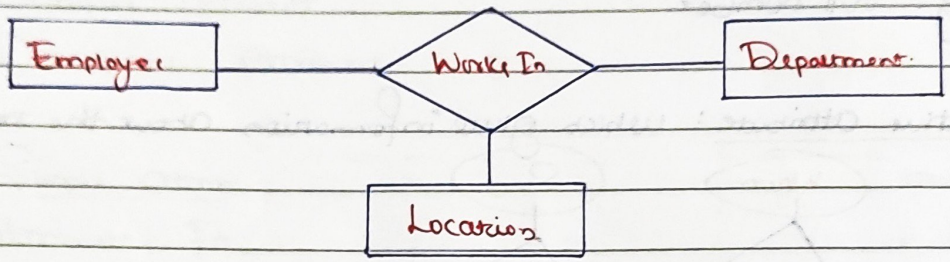
(1.) UNARY: Relation among two entities of the same entity set.
(Recursive Relationship Set).



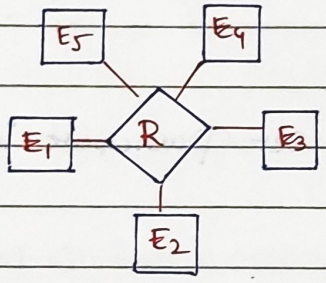
② BINARY : The relationship among two entity set.



③ TERNARY : The relationship among three entity set.



④ N-ary relationship - The relationship among n-entity set.

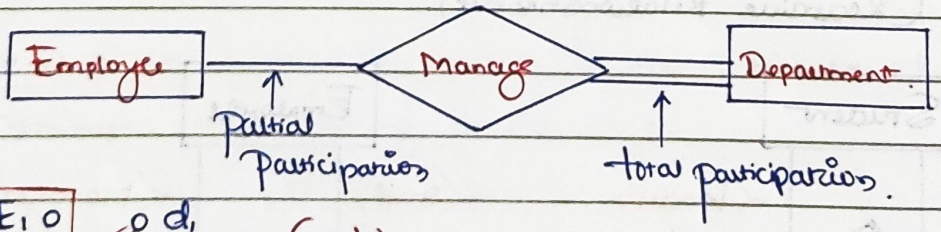


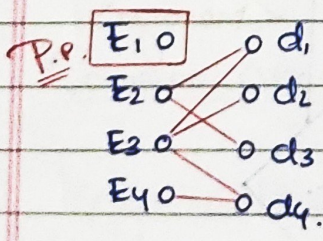
Participation : If every entities of entity set all participated in relationship set then it is total participation (100%) otherwise partial participation (<100%).

Participation Constraint:

* If every entity in the entity set participated in a relationship set is called Participation Constraint.
* Total participation denoted by double line (thick line) otherwise it is called partial participation (thin line or single line).

eg: Each department is managed by atleast one employee.

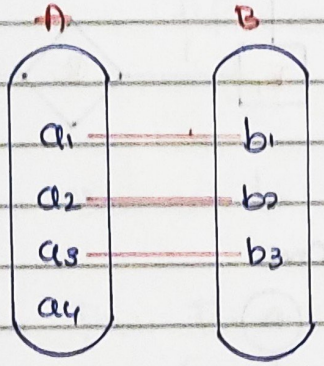




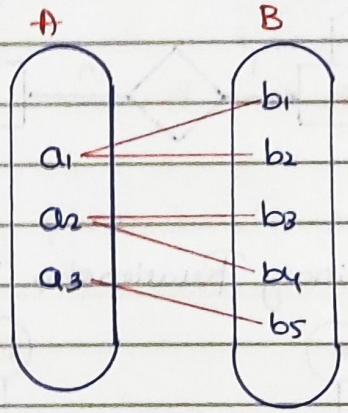
Mapping Cardinality Constraints :

- Express the number of entities to which other entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of following types.
  - One to one (1:1)
  - One to many (1:many)
  - Many to one (m:1)
  - Many to many. (m:m)

Mapping Cardinalities.

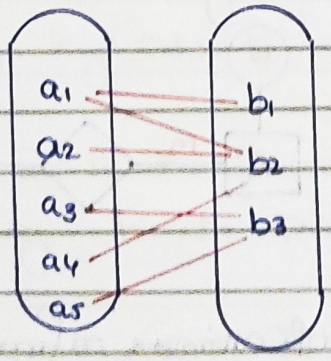


One to One

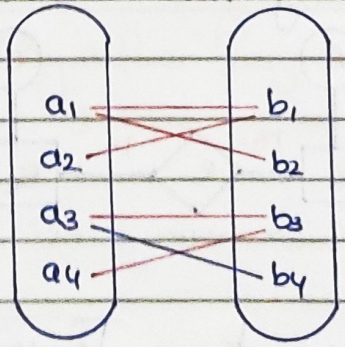


One to many.

Note: Some elements in A and B may not be mapped to any elements in the other set.



Many to One



Many to many.

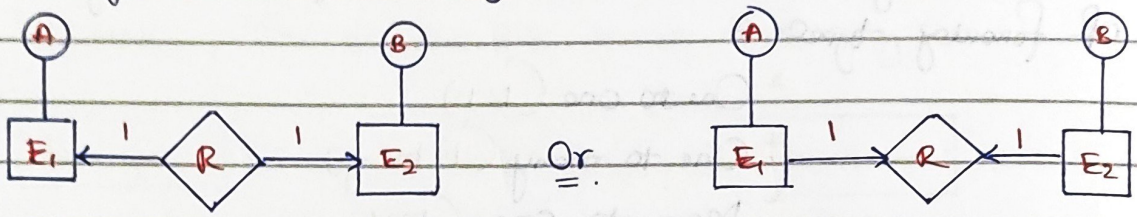
Note: Some elements in A and B may not be mapped to any elements in the other set.

Mapping (Cardinality Constraints of relationship Set)

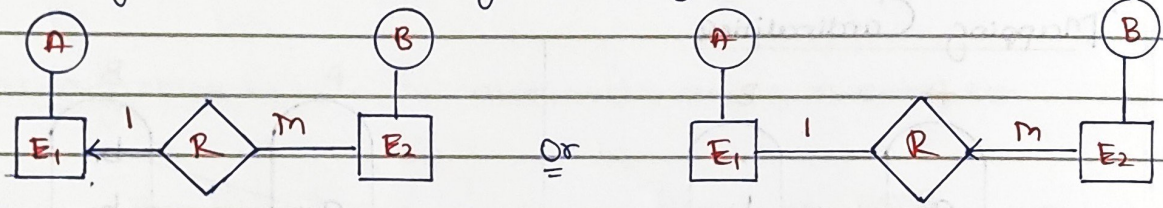
One mapping : At most One (0 or 1)

Many mapping : 0 or more (0....*).

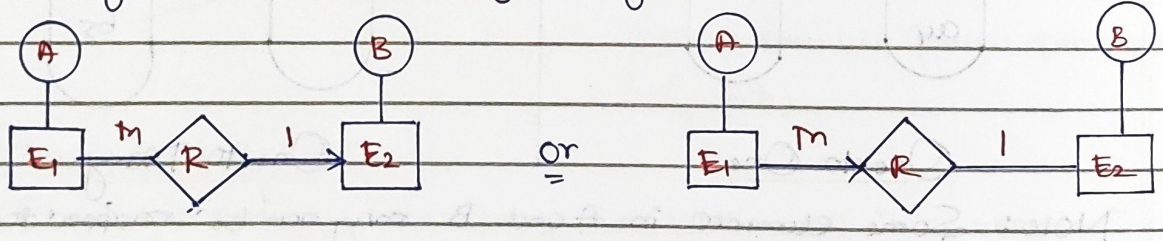
Binary Relationship Mapping (One : One)



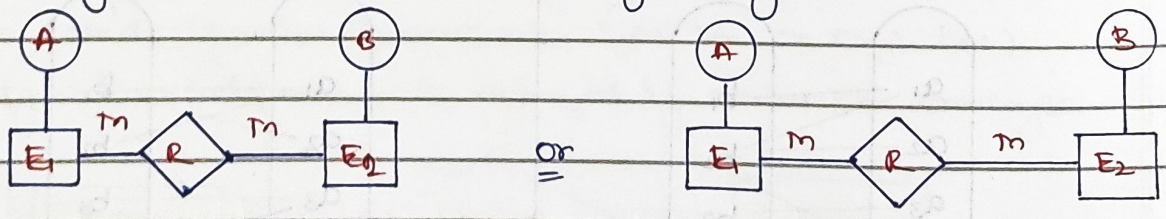
Binary Relationship Mapping (One : many)



Binary Relationship Mapping (Many : One)



Binary Relationship Mapping (Many : Many)



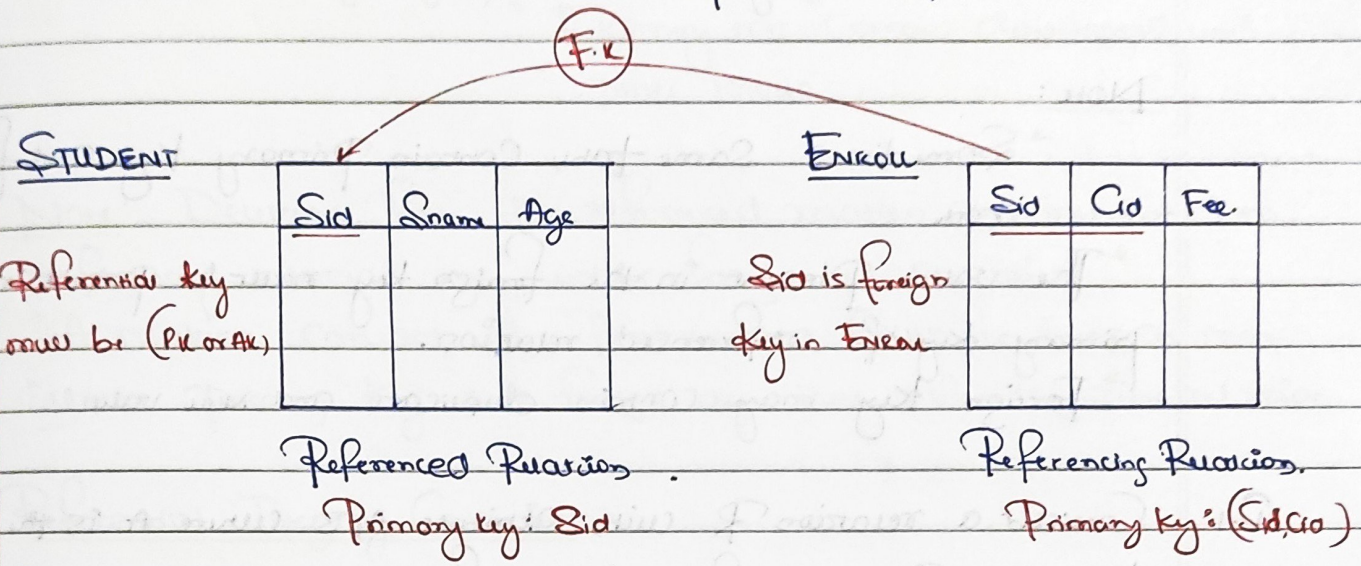
Foreign key: Foreign key is a set of attribute refrence to the Primary key or alternate key of same table or some other table.

It is used to relate or relation (table) with other or same relation.

Date 28/8/23

Referencing Relation: The table which contains the foreign key is known as referencing relation (child table/relation).

Referenced Table: The table which is referenced by foreign key is known as referenced relation (parent table/relation).



Note:

* Atmost there can be one primary key per relation (table).

→ Create Table Enrolled
Sid Varchar (10)
Cid Varchar (10)
Fees Integer (11)
Primary key (Sid Cid)
Foreign key (Sid) Reference Student

(per)
Atmost one primary/relation so directly write name of table.

→ By default foreign key reference to primary key of refd relation

When Sid is the primary key of Student
Let login is the primary key and Sid is alternative key then

Foreign key (Sid) Reference Student (Sid). → When Sid is not primary key but alternative key.
∴ many AK possible so write attribute name

RDBMS Constraints

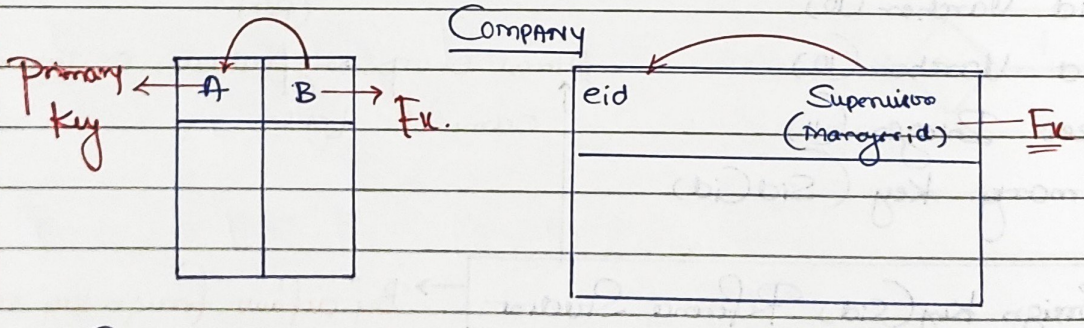
Domain Constraint - (Atomic)
Key Constraint - (Uniqueness)
Entity Integrity Constraint - (primary key not null)
Referential Integrity Constraint - (foreign key concept)

Note:

* Sometimes same table contain primary key and foreign key both
* The value present in the foreign key must be present in the primary key of referenced relation.
* Foreign key may contain duplicate and Null values.

Q.1 Consider a relation R with attribute A, B where A is the primary key and B is the foreign key, reference to primary key 'A' on same table. Which of the following is correct according to constraints? or which of the row can be sucessfuly inserted into R?

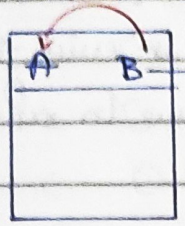
Ans (1,2) (2,3) (3,4) (4,2) (5,3) (6,5) (7,4)



Same table with different name Concept.

* The value present in foreign key must be present in primary key of referenced relation

Parent Table	Child Table
Referenced Table	Referencing Relation
✓ Insert <4 D ECE>	✗ Insert <105 DSA 67>
✗ Delete <1 A CSE>	✓ Delete <103 CO 3>



Constraints

* Primary key not contain null value
* The value present in F.K must be present in primary key of referenced relation.
* F.K may not contain duplicates and Null values.

Note: * Deletion from the referenced relation and insertion into referencing relation may violate foreign key constraint.
* A relation can act as PARENT and CHILD ie Relation may contain primary key and foreign key that refer to same relation

Referential Integrity Constraint:

1. Referenced Relation
   (i) Insertion: No violation
   (ii) Deletion: May cause violation if primary key is used by referencing relation.
      a. On delete no action
      *b. On delete Cascade
      c. On delete Set null

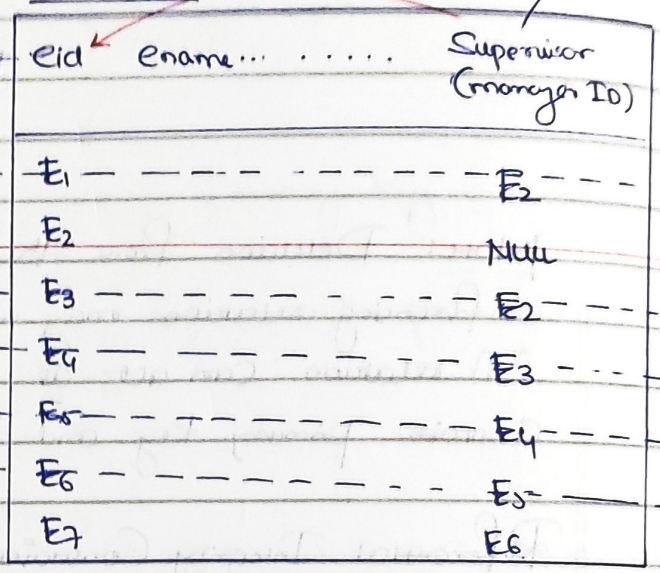
Foreign key	Referenced Relation

(i) Insertion: No violation
(ii) Deletion: May cause violation.
   (a) On delete no action: means if it cause problem on delete then deletion is not allowed on table.
   (b) On delete Cascade: If we want to delete primary key value from referenced table then it will delete their value from referencing table also.

(C). On delete set null: If we want to delete primary key value from referenced table then it will try to set the NULL values in place of that value in referencing table.

Foreign key (Ingrid) reference EMP ON DELETE CASCADE

EMPLOYEE



Q (E2, NULL) P.K E2 deleted

→ Due to P.K E2 → [E1, E2] [E3, E2] deleted

Now E1 and E3 deleted

→ Due to P.K E3 → [E4, E3] deleted

Now P.K E4 deleted

→ Due to P.K E4 → [E5, E4] deleted

Now P.K E5 deleted

→ Due to P.K E5 → [E6, E5] deleted

Now P.K E6 deleted.

→ Due to P.K E6 → [E7, E6] deleted.

Note: For good database design Sometimes On delete cascade not suggestable.

Because like in previous example deletion of one tuple (On delete cascade) result leads to deletion of complete table

(iii) Updation : (May cause violation)

a. On update no action
b. On update Cascade
c. On update Set null.

Referencing Relation :

1. Insertion : May cause violation
2. Deletion : No violation
3. Updation : May cause violation.

Note : If integrity violation occure because of insertion or updation in referencing table then restrict insertion and updation

Q2. The following table has 2 attributes A and C where A is the primary key and C is the foreign key referencing A with On-delete Cascade. The set of all tuples that must be additionally deleted to preserve referral integrity when the tuple (2,4) is deleted is :

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

Primary key 2 is deleted so, (5,2) (7,2) got deleted
Now Primary key 5, 7 deleted
Due to PK 5 → (9,5) deleted, so '9' also got deleted

∴ (5,2) (7,2) and (9,5) got deleted additionally.

E-R model : Entity Relationship Diagrams used to represent diagramatic design (High level design) of Database.

DB Design Steps :

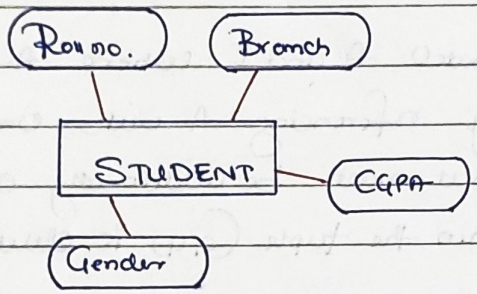
1. Requirement : What type of data stored and what operations required etc. } High level design
2. Conceptual and Logical : [ER Diagram]
3. ER Diagram to RDBMS table design and apply normalization. Physical DB Design (Indexing Design) User Interface Design and Security Design. } Low level design

ER MODEL → RDBMS

ER MODEL
- Multivalued attribute
- Composite attribute
- Weak Entity Concept.

RDBMS
- No Multivalued Attribute
- No Composite Attribute
- No weak entity concept.

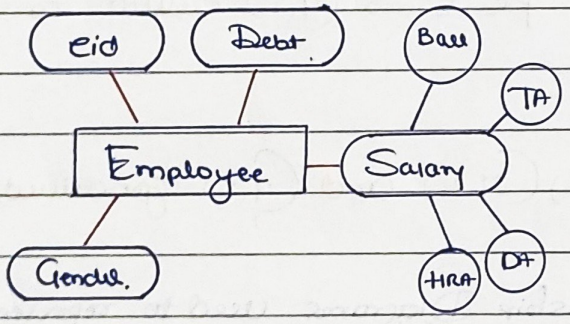
1. Entity Set. → Relation (Table)
Key attribute → Primary key
Entity. → Tuple



STUDENT

Roll no.	Branch	CGPA	Gender.

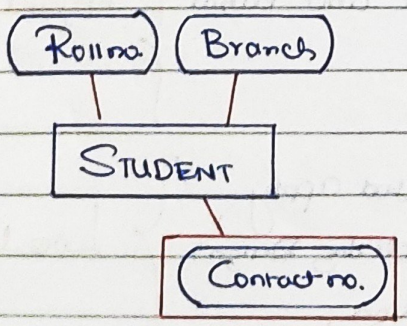
2. Composite Attribute → Set of Simple Components



Employee.

eid	Debt	Gender	Basic	TA	DA	HRA

3. Multivalued Attribute. → 2 Table.
→ 1. Key attribute + All other Attribute.
→ 2. Key attribute + All multivalued Attribute



1.

Roll no.	Branch	CGPA

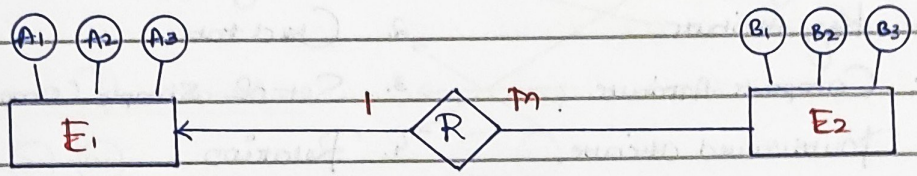
2.

Roll no.	Contact

ER to DBMS Conversion:

ER → RDBMS

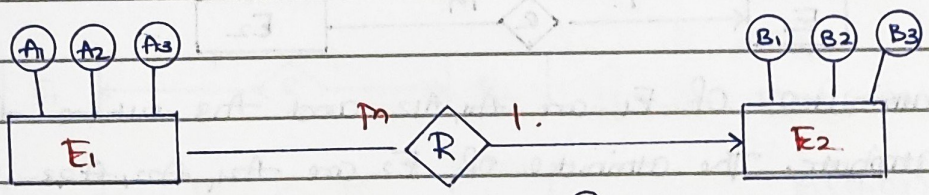
1. One to Many ⟶ 2 Table Required (Relationship set merge with many side).
2. Many to One ⟶ 2 Table Required (Relationship set merge with many side).
3. Many to Many ⟶ 3 Table Required (Seperate table for relationship set)
4. One to One ⟶ 2 Table Required. (Relationship set merge any side)



Two table Required

$E_1(\underline{A_1}\ A_2\ A_3)$ $R_{E_2}(B_1\ B_2\ B_3\ \underline{A_1})$ ← FK

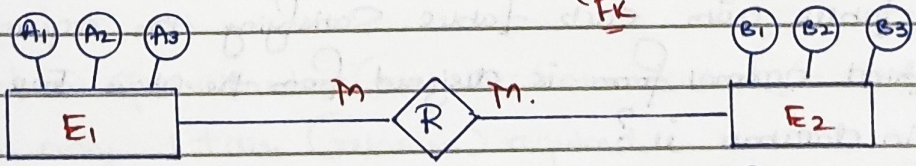
and 1 foreign key.



Two table required & 1 foreign key.

$E_1\ R(\underline{A_1}\ A_2\ A_3\ B_1)$ $E_2(\underline{B_1}\ B_2\ B_3)$

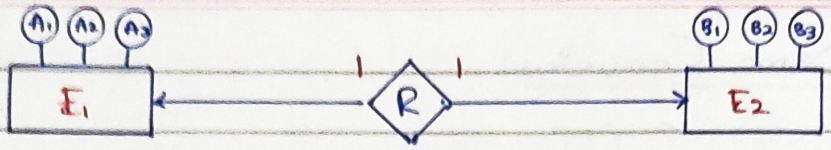
FK



Three Table required and two foreign key

$E_1(\underline{A_1}\ A_2\ A_3)$ $R(\underline{A_1}\ \underline{B_1})$ $E_2(\underline{B_1}\ B_2\ B_3)$

FK FK



Two table required and one foreign key.

$E_1 R(A_1 A_2 A_3 B_1)$ and $E_2(B_1 B_2 B_3)$

(FK)

or

$E_1(A_1 A_2 A_3)$ & $RE_2(B_1 B_2 B_3 A_1)$

(FK)

Q1. The term in list A have been mapped to list B so that it corresponds to the mapping process of ER model into relational. Which of the following represent the mapping process?

A. Entity type → 1. Primary key (or alternate key)

B. Key Attribute → 2. Child table

C. Composite Attribute → 3. Set of Simple Component Attributes

D. Multivalued Attribute → 4. Relation.

Q2. Consider the following entity relationship diagram (ERD) where two entities E_1 and E_2 have a relation R of cardinality 1:m

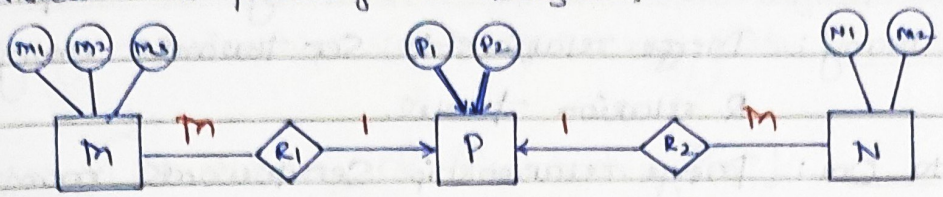
The attributes of E_1 are A_{11}, A_{12} and A_{13} where A_{11} is the key attribute. The attributes of E_2 are A_{21}, A_{22}, A_{23} where A_{21} is the key attribute and A_{23} is multi valued attribute. Relation R does not have any attribute. A relational database containing minimum no. of tables with each table satisfying the requirements of the third normal form is designed from the above ERD. The no of table in database is?

Ans $E_1(A_{11}\ A_{12}\ A_{13})$ $RE_2(A_{21}\ A_{22}\ A_{11})$

$E_2(A_{21}\ A_{23})$

∴ 3 Table required.

Q3. Consider the following ER Diagram.



MR1 (m1, m2, m3, P1)
P (P1, P2)
NR2 (N1, N2, P1)

(i) The min. no of table required to represent M, N, P, R1, R2 is: 3

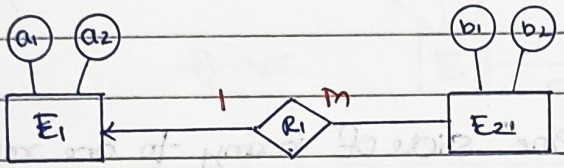
(ii) Which of the following is a correct attribute set for one of the table for the correct answer to the above question?

Ans {m1, m2, m3, P1}

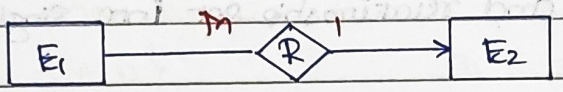
Participation Constraint

1. Partial Participation (—)
2. Total/Full Participation (═)

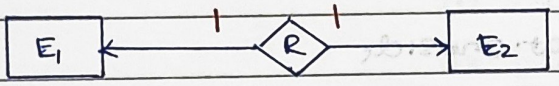
(I) Partial Participation



2 Table E1(a1 a2) RE2(b1, b2, a1) — FK



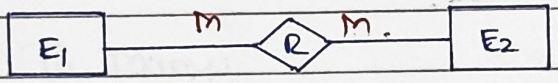
2 Table E1R(a1 a2 b1) E2(b1, b2) — FK



2 Table E1R(a1 a2 b1) & E2(b1 b2) — FK

or

E1(a1 a2) & RE2(b1 b2 a1)



3 Table E1(a1 a2) R(a1, b1) E2(b1, b2)

Note:

1. If total participation (═) at both side (in 1:M, M:1, 1:1 & M:N) then only 1 table (relation) required
2. If total participation at 'one side' (in 1 to M & M to 1) then only 1 table (relation) required.
3. If total participation at any one side of in many to many then 2 table (2 relation are required)
4. All other cases are same as normal case (As usual).

Partial Participation on both side of binary relationship

One to Many: Merge relationship set towards many side. So 2 relation tables.

Many to One: Merge relationship set towards many side. So 2 relational table.

One to One: Merge relationship set any one side. So, 2 relational tables.

Many to Many: Seperate table for each entity set and relation-ship set. So, 3 relational tables.

II. Full Participation on both sides of relationship:

1:1
1:M
M:1
M:N

} Merge the entity sets and relationship into single relational table so, 1 relational table.

1. Full Participation on "One" side of many to one relationship.

→ Merge the entities and relationship set into single relational table. So, 1 table.

1:M
M:1
1:1

} Tp at one side
1 Table.

2. Full Participation on any "One" side in One-One relationship.

→ Merge the entity set and relationship set into single table. So, 1 table.

3. Full Participation on any "Many" side of Many-Many relat. p.

→ Merge relationship set towards any "Many" side of relationship. So, 2 table.

20/8/23

4. Full Participation on "many" side of many-to-One relationship?
→ Merge relationship set towards many side. So, 2 relational-tables.

Mapping [cardinality of Relationship Set]

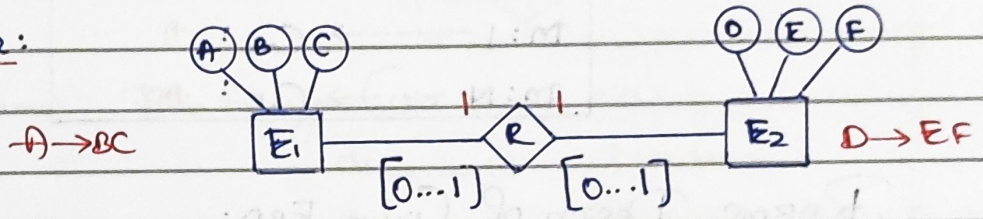
One: [Almost one] [0...1]
Many: [0 or more] [0...m]

Possible Mapping of binary relationship sets.

1. One : One
2. One : many
3. Many : One
4. many : many

} Candidate key of Relationship set is based on mapping.

One to One:



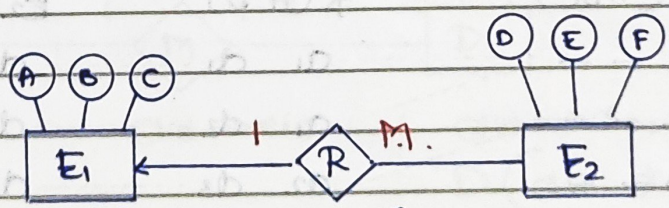
Candidate keys = 2
{A, D}.

R (A D) = {A, D}

a_1	d_1
a_2	d_2
a_3	d_5

1:1 Mapping Candidate key of relationship set.

One To Many:



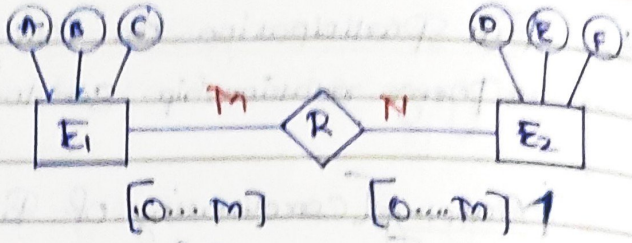
Candidate key of
R (A D) = {D}

a_1	d_1
a_1	d_2
a_2	d_3

[0...1] Each object of E_2 allowed to pair by almost one entity of E_1

[0...m] Each entity of E_1 can pair with many E_2 entity.

Many to Many:



R(A D) = {AD}

a_1	d_1
a_1	d_2
a_2	d_2
a_3	d_3

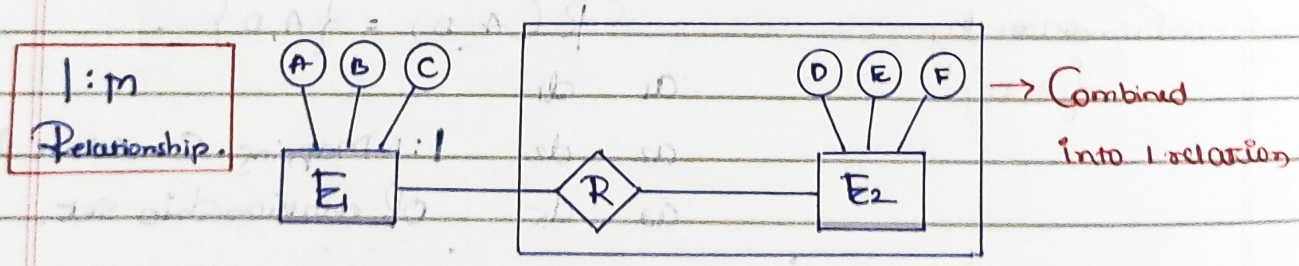
M:N

For many to many mapping Candidate key for relationship set is {AD}.

Summary:

1 : 1	C.K : A or D
1 : M	C.K : D
M : 1	C.K : A
M : N	C.K : AD.

RDBMS Design of Given ERD:



$E_1(ABC)$	$R(AD)$	$E_2(DEF)$
a_1 — —	a_1 d_1	d_1 — —
a_2 — —	a_1 d_2	d_2 — —
a_3 — —	a_2 d_3	d_3 — —
a_4 — —		d_4 — —
$A \rightarrow BC$	$D \rightarrow A$	$D \rightarrow EF$
{A}	{D}	{D}

{A → BC} E_1 (ABC) E_2 R(DEFA) {D → AEF}

a_1 — — d_1 — — a_1

a_2 — — d_2 — — a_1

a_3 — — d_3 — — a_2

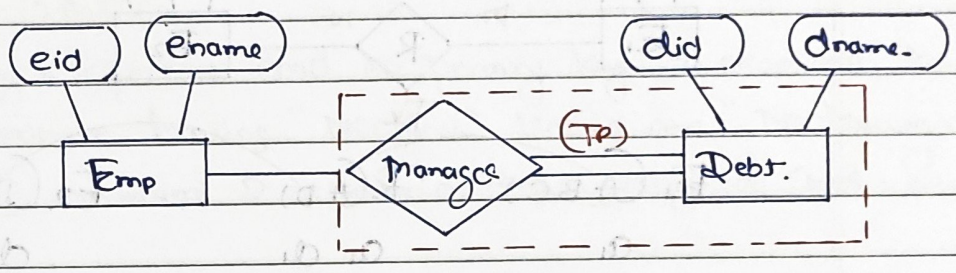
a_4 — — d_4 — — Null

Partial participation between E_2 and R

↳ due to partial participation.

∴ Minimum 2 relational table and 1 Foreign key required for given ERD.

Given ERD example:



Emp (eid ename) Dep Manager (did, dname eid)

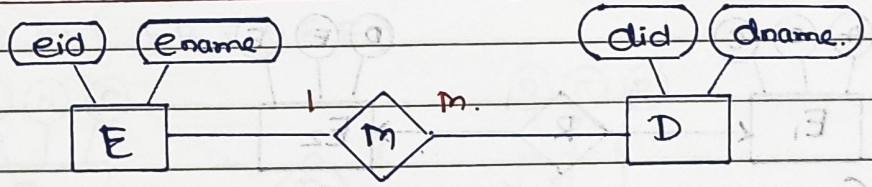
d_1 — — e_1 NOT NULL

d_2 — — e_2 If act as total participation at e_2 side.

d_3 — — e_3

d_4 — — e_4

→ If 1:m relationship set merge into left side: Entity set:



eid → ename. did → eid did → dname

D (did dname)

E_m (eid ename did)

e_1	A	d_1
e_1	A	d_2
e_2	B	d_2
e_1	C	d_3
e_1	D	Null

Actually 2 Table: E(eid, ename)

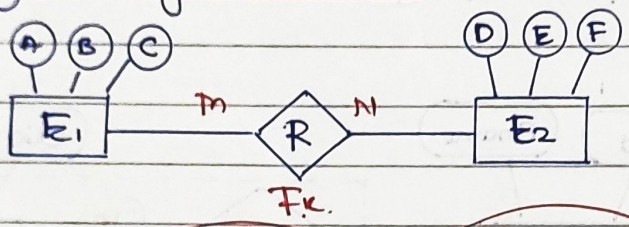
D(did, dname, eid)

→ Not valid as did is key of E-m (partial participation) [lost at E side]

Disadvantage:

1. Data Redundancy problems/ Occurs
   eid → ename
   Not Super Key (S.K)
2. Partial participation will be lost.
   (not possible to insert Employees who are not manager of dept).

Many : Many Relationships

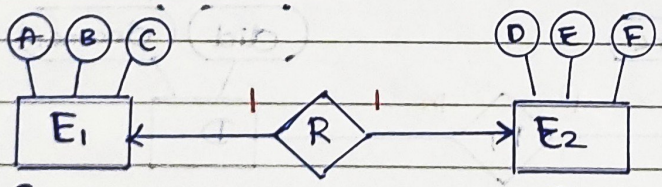


E1 (A B C) R (A D) E2 (D E F)

E1	R	E2
a1	a1 d1	d1
a2	a2 d2	d2
a3	d2 d2	d3
(Key)	(Not Key)	(Key)

Minimum 3 relational table and 2 foreign key required. M:N relationship set not allowed to merge with any entity set.

One : One Relationship



E1 (A B C) R (A D) E2 (D E F)

E1	R	E2
a1	a1 d2	d1
a2	a2 d3	d2
a3	a3 d4	d3
a4	(Key)	d4
(Key)	[A → D	(Key)
[A → BC]	D → A]	[D → EF]

E1, E2 entity sets and R relationship set related between E1 and E2 with 1:1 Cardinality and Partial participation at both side.

E1 (A B C) & R E2 (D E F A)

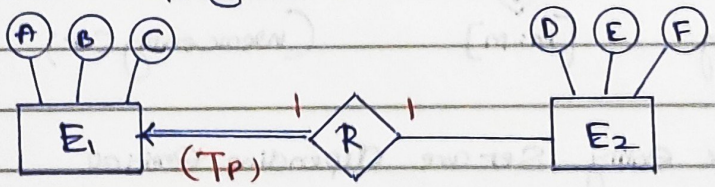
If all merge with Single table..

A	B	C	D	E	F
a_1	–	–	d_2	–	–
a_2	–	–	d_3	–	–
a_3	–	–	d_4	–	–
a_4	–	–	Null	Null	Null
Null	Null		d_1		

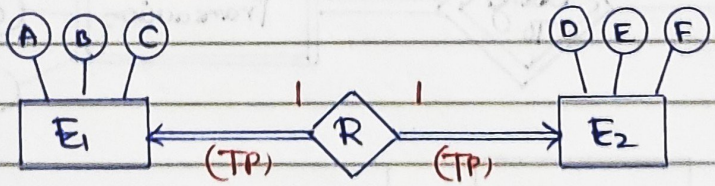
Candidate key = {A, D}.

No Candidate key is present and no primary key. A relational table in which no attribute having "NOT NULL" value one "Not" allowed in RDBMS. So minimum 2 relational table required and 1 foreign key.

E_1 E_2 Entity Set R Rel Set between E_1 E_2
With 1:1 mapping and at least one end having total participation.

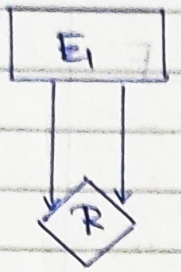


E_1 R E_2 (A B C D E F)

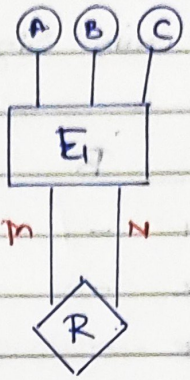


E_1 R E_2 (A B C D E F)
(primary key) (Unique & NOT NULL (alternative key))

Recursive Entity Set:



1:m, m:1
1:1 then only one table required.

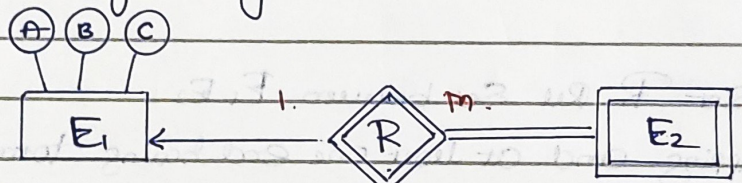


2 Table required
E1 (ABC)
R (A1, A2)

Weak Entity Set:

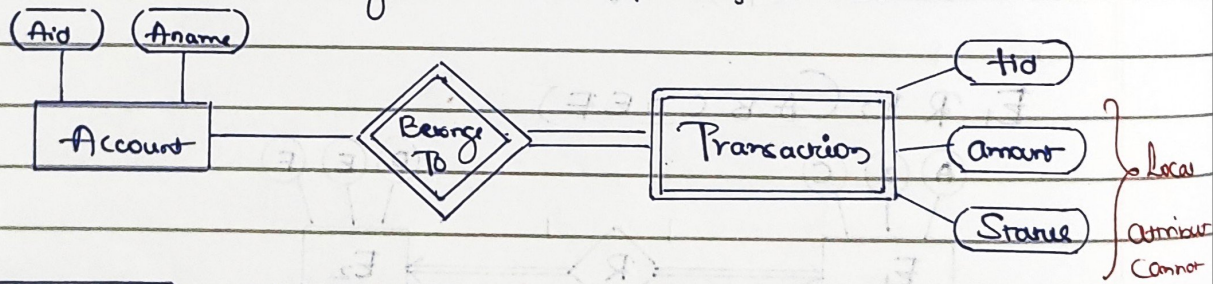
* Entity set with no key. [attributes of weak entity set not sufficient to differentiate entities uniquely].

* For each weak entity set there must be OWNER entity set - which is strongly entity.



OWNER entity Set. [m:m] (weak entity set)

* Entities of weak entity set are depending entities.



Aid	Aname
A1	
A2	
A3	
A4	

Aid	Tid
A1	t1
A2	t1
A2	t2
A3	t2

Tid	Amount	Status
t1	5000	debit
t1	4000	Credit
t2	5000	debit
t2	5000	debit

[1:M]
Ambiguity.

- Relationship set between weak entity set and identifier entity is also weak relationship set. ◇
- Participation towards weak entity set and must be Total participation.
- Mapping between identifier set must be One : many
- RDBMS Design:
  Transaction belongs (Aid tid Amount State)
  Account (Aid Aname).

Note: Weak entity set and multivalued attributes allowed to represent in ERD, but not allowed in RDBMS Table.

04/9/24

QUERY LANGUAGE

Query Language

1. Procedural Q.L
   - Relational Algebra
2. Non Procedural Q.L
   - SQL
   - TRC and DRC

Procedural Query Language	Non-procedural Query language
Formulation of how to access data from the database table and what data required to retrieve from DB tables.	Formulation of what data retrieve from DB tables
"Relational Algebra"	"Relational Calculus" "SQL".

Relational Algebra

- The basic idea of query language is Query executed on a DB table (relation) tuple by tuple (step by step). One tuple at a time.
- Relational algebra always generates distinct tuples.
- Relational algebra refers to a procedural query language that takes relational instances as input and returns relational instance as output.

Basic Operators

1. Selection (σ)
2. Projection (π)
3. Cross Product [$\times$]
4. Union [$\cup$]
5. Set difference [$-$]
6. Rename [ρ]

Derived Operators

1. JOIN & its types ($\bowtie$)
2. Division ($/$)
3. Intersection ($\cap$)

$R \cap S = R - (R - S)$

Selection [σ]: It select the tuples/Record/Row based on Specified Condition.

Syntax: $\sigma_{condition}(Relation)$ eg: $\sigma_{CGPA>9}(STUDENT)$

Projection [π]: It select fields/Attribute/Attribute list from relation.

Syntax: $\pi_{Attribute}(Relation)$ eg: $\pi_{Sname}(STUDENT)$

Q1. W.A.Q to find name of 'Red' color pens.

$\pi_{Pname}[\sigma_{Color=Red}(Pens)]$

Pid	Pname	Color
1	A	Red
2	B	Green
3	C	Red
4	D	Green
5	A	Yellow

O/P:

Pname
A
C

Q2. $\pi_{Sname, rating}[\sigma_{rating>8}(S2)]$

S2

Sid	Sname	Rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

O/P:

Sname	rating
yuppy	9
rusty	10

$\pi_{age}(S2)$ O/P:

age
35.0
55.0

} Only distinct value

① $\sigma_{C3}(\sigma_{C2}(\sigma_{C1}(R))) \equiv \sigma_{C2}(\sigma_{C1}(\sigma_{C3}(R)))$

C_1, C_2, C_3 } Conditions

* Selection [σ] is Commutative

② $\sigma_{C_2}(\sigma_{C_1}(R)) \equiv \sigma_{C_1} \wedge \sigma_{C_2}(R)$ ('AND')

③ $\Pi_{A_1}[\sigma_{C_1}(R)] \neq \sigma_{C_1}[\Pi_{A_1}(R)]$

④ $\sigma_{C_1}[\Pi_{A_1}(R)] \xrightarrow{\text{(Transformed)}} \Pi_{A_1}[\sigma_{C_1}(R)]$

In this (LHS Pair) firstly ~~Attribute~~ A_1 is selected then condition C_1 is applied on ~~Attribute~~ A_1.

⑤ $\Pi_A[\Pi_{AB}(R)] \equiv \Pi_A(R)$

⑥ $\Pi_a(R) \equiv \Pi_a(\Pi_b(R))$ ↳ $a \subseteq b$ (a is a subset of b)

⑦ $\Pi_{A_{11}}[\sigma_C(R)] = \sigma_C[\Pi_{A_{11}}(R)]$ * if it includes an Attribute used in C[condition]

Set Operators [∪, ∩, −]

* To apply set operators Relation must be Union Compatible (Type Compatable).
* R and Relation S is Union Compitable if:
  (i) Arity (Degree (#Attribute) of R & S must be Same
  (ii) Range (Domain) of Attribute must be Similar.

eg:

'C' Form.

Roll no.	Name.
4	Jimmy
2	Pit

∪, ∩, −

'J' Form

Roll no	Name
7	Ghost
3	Tim

- ~~R~~ Arity is same
- Domain is Same (datatype)

(datatype) ↓ (now we can apply Set operators)

* Same domain with different name allowed.
* Relation $R(A_1, A_2 \ldots A_n)$ & $S(B_1, B_2 \ldots B_n)$ is Union Compatible if:
  1. Arity of R = Arity of S
  2. Domain of A_i = Domain B_i (Corresponding Pair have Same domain)

eg: R

A
2.
2.
2.
3.

S

B
2
2
4

$R \cup S$ =

2
3
4

Set Union Operation.

Only unique values are displayed because it is relational algebra.

→ Assume Relation R and S consist m and N tuple respectively

1. Range of tuples in $R \cup S$ = max(m,N) to m+N
2. Range of tuples in $R \cup S$ = ϕ to min(m,N)
3. Range of tuples in $R - S$ = ϕ to m
4. Range of tuples in $S - R$ = ϕ to N.

Union Operation

Notation : $r \cup s$

Defined as : $r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$

Set Difference

Notation : $r - s$

Defined as : $r - s = \{ t \mid t \in r \text{ and } t \notin s \}$

Intersection Operation :

Notation : $r \cap s$.

$r \cap s = s \cap r$ Commutative.

eg:

R

A	B
α	1
α	2
β	1

S

C	B
α	2
β	3

① Union Operation

$R \cup S$

A	B
α	1
α	2
β	1
β	3

$S \cup R$

C	B
α	2
β	3
α	1
β	1

$R \cup S = S \cup R$ Commutative

② Minus Operation

$R - S$ =

α	1
β	1

$S - R$ =

β	3

③ Intersection

$R \cap S$, $S \cap R$ = α 2 Commutative.

Cross/Cartesian Product (R×S)

R	S	R×S
n_1 tuple	n_2 tuple	$n_1 \times n_2$ tuple
C_1 attribute	C_2 attribute	$C_1 + C_2$ attribute

R×S : It results all attributes of R followed by all attributes of S, and each record of R paired with every record of S.

Degree (R×S) = Degree (R) + Degree (S)

$|(R \times S)| = |S| \times |S|$

eg: ① R(A B C)

A	B	C
1	2	3
4	5	6

② S(D E F)

D	E	F
7	8	9
10	11	12

R×S (Cross Product)

A	B	C	D	E	F
1	2	3	7	8	9
1	2	3	10	11	13
4	5	6	7	8	9
4	5	6	10	11	13

Note: If R has 'C' & S also 'C' (attribute name) then in resultant table "R.C" and "S.C" will be the table attribute.

Note:
* Relation R with n tuples and Relation S with 0 tuple then → no. of tuple in R×S = 0 tuple
* For Set operator [∪, ∩, −] both relations must be Union (Type) Compatible
* For Cross/Cartesian Product (Cross Join) relation not required to be Union (Type) Compatible.

Q.1 Why Cross Product is required?

1. For performing the join operations (Inner join and Outer join)
2. If we require R.C < S.D, We need to do Cross product. (explained with example (P.T.O))

eg:

R

A	B	C
1	2	4
3	4	6

S

B	C	D
2	4	8
2	7	4.

If we require R.C < S.D, it is not possible in query language directly because Query execution/executed on a db table Tuple by Tuple, one tuple at a time.

So it fetches a tuple either from relation R or from relation S at a time So not possible to compare that is why we do Cross Product to convert them into one tuple.

But, R: 500 Tuple S: 400 Tuple

R×S = 500×400 = 200000 (2 lakh) Tuple

Only Cross Product is not meaningful until and unless we are not applying any condition (selection) and not selecting any attribute.

JOIN Operations (⋈)

① Natural Join ($R \bowtie S$): It's derived operator performed in 3 steps

Step 1: Cross product of R and S.

Step 2: Select the tuple which satisfy equality condition on all common attributes of R and S (From R×S)

Step 3: Projection of distinct attributes.

$$R \bowtie S = \Pi_{\text{distinct attribute}} \left[\sigma_{\substack{\text{equality condition} \\ \text{on all common} \\ \text{attributes}}} (R \times S) \right]$$

② Conditional Join ($R \bowtie_C S$)

$$R \bowtie_C S = \Pi_{\text{all attributes}} \left[\sigma_{\text{given Condition}} (R \times S) \right]$$

JOIN - OPERATIONS

* Join Operation is used to combine information from two or more relations (Tables)
* Join Operation performed by the cross product followed by selection and projection.

Inner Join

1. Conditional (Theta) Join [$\bowtie_c$]
2. Equi Join
3. Natural Join

Cross - Join

1. Cross product - [$R \times S$]

Outer Join

1. (Loj) Left outer join ($R \bowtie S$) or $R \overset{\circ}{\bowtie}_L S$
2. (Roj) Right outer join ($R \bowtie S$) or $R \overset{\circ}{\bowtie}_R S$
3. Full outer join ($R \bowtie S$) or $R \overset{\circ}{\bowtie} S$

Full Outer Join = Outer Join ($R \bowtie S$) or ($R \bowtie S$)

Dangling Tuple: It is a tuple (row) that fail to match (satisfy) any tuple of other relation in common attribute or given conditions.

Superious Tuple: Extra tuple (in join)

eg:

R

A	B	C
1	3	5
4	6	7
7	9	11

S

D	E	F	G
4	6	7	8
7	7	8	9

$R \bowtie_{R_B < S.D}$ (Conditional Join)

S →

R_A	R_B	R_C	SD	SE	SF	SG
✓ 1	3	5	4	6	7	8
✓ 1	3	5	7	7	8	9
✗ 4	6	7	4	6	7	8
✓ 4	6	7	7	7	8	9
✗ 7	9	11	4	6	7	8
✗ 7	9	11	7	7	8	9

3. Equi Join ($R \bowtie_{\theta} S$): It is a subset of conditional or similar to conditional join but here "only equality condition" applied.

$$R \bowtie_{\theta} S = \pi_{\text{Attributes}} \left[\sigma_{\substack{\text{equality} \\ \text{conditions}}} (R \times S) \right]$$

Note: Natural join equal to cross-product if join condition is empty
If no common attribute then $R \bowtie S = (R \times S)$

Outer Join	+
1. Left outer join [LOJ]	Outer join is done because in inner join some tuple failed to satisfying the conditions (Join condition) [Dangling Tuple] So loss of data. (For getting the complete table data)
2. Right outer join [ROJ]	
3. Full Outer join	

Outer Join ≡ Natural Join + Dangling Tuples

Left Outer Join ($R \bowtie S$)

Natural left outer Join

$R \bowtie S = R \bowtie S$ && The tuples from left side relation R those failed to satisfy join condition.

Or

$R \bowtie S = R \bowtie S$ + Include dangling tuple of left side relation R.

Conditional left outer Join

$R \bowtie_c S = R \bowtie_c S$ + Include dangling tuple of R

Right Outer Join: Natural Right Outer Join. $R \bowtie S = R \bowtie S$ && Include dangling tuple of S.

Conditional right outer Join

$R \bowtie_c S = R \bowtie_c S$ + Include dangling tuple of relation S

Full Outer Join

$$R \overline{\bowtie} S = R \bowtie S \cup R \overline{\bowtie} S$$

or

$R \overline{\bowtie} S$ = $R \bowtie S$ && Include Dangling tuple of relation R & S.

Conditional full outer Join: $R \overline{\bowtie}_c S = R \bowtie_c S$ + Include dangling tuples of relation R and S.
(Conditional Join)

Rename Operator [ρ]: It is used to rename table name and attribute names for query processing.

eg: STUD (Sid, Name, Age)

1. ρ (Temp, STUD) ⟶ Temp (Sid, Sname, Age) — Table Name Rename
2. ρ (S, N, A) STUD ⟶ STUD (S, N, A) — Renaming Attribute
3. $\rho_{1 \to S,\ 2 \to N}$ (STUD) ⟶ STUD (S, N, Age) — Renaming particular attributes.

Assignment Operator: Temp1 ← {Any Result}.

Basic Operator:
- π : Projection
- σ : Selection
- × : Cross Product
- ∪ : Union
- − : Set-difference
- ρ : rename Operator

Derived Operators
- ∩ : intersection {using "−"}
- ⋈ : Join {using ×, σ}
- / or ÷ : division {using π, ×, −}

Division Operator [/] : It is a derived operator.

$\frac{\Pi_{A,B}(R)}{\Pi_{B}(S)}$ => Quot (A,)

* It is used to retrieve attribute value of R which has paired with every attribute value of other relation S.
* $\Pi_{A,B}(R) / \Pi_{B}(S)$: It will retrieve value of attribute 'A' from R which there must be pairing 'B' value for every 'B' of S.

Q1. Retrieve all Students who are Enrolled in Some course or any course or atleast One Course?

Enrolled

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_3
S_3	C_1

Course

Cid
C_1
C_2
C_3

Solution : Π_{Sid} (Employee)

Sid
S_1
S_2
S_3

$\frac{\Pi_{AB}(r_1)}{\Pi_{B}(r_2)}$ = Quotient (A)

- value of 'A' which pair with every value B of r_2.

To retrieve all Students who are enrolled in every course:

$\Pi_{Sid, Cid}$ (Enrolled) / Π_{Cid} (Course)

Sid — Find

Cid — 2nd attribute must be same.

Ans = S_1

Another way : Π_{Sid} (Enrolled) − Π_{Sid} [Π_{Sid} (Enrolled) × Π_{Cid} (course) − Enrolled]

not enrolled every course

Π_{Cid} (Enrolled) − Π_{Sid} [Π_{Sid} (Enrolled) × Course − Enrolled]

$$\Pi_{AB}(R) / \Pi_{B}(S) = \Pi_{A}(R) - \Pi_{A}\left[\Pi_{A}(R) \times \Pi_{B}(S) - R\right]$$

To find — Connection.

$$\Pi_{ABCD}(R) / \Pi_{CD}(S) = \Pi_{AB}(R) - \Pi_{AB}\left[\Pi_{AB}(R) \times \Pi_{CD}(S) - R\right]$$

1. $\dfrac{\Pi_{AB}(R)}{\Pi_{B}(S)} = \text{Quotient}(A)$ — Getting 'A' which pair with every B value of S in relation R.

$$\Pi_{A}(R) - \Pi_{A}\left[\Pi_{A}(R) \times \Pi_{B}(S) - R\right]$$

2. $\dfrac{\Pi_{AB}(R)}{\Pi_{A}(S)} = \text{Quotient}(B)$ — Getting 'B' which pair with every value of S in relation R.

$$\Pi_{B}(R) - \Pi_{B}\left[\Pi_{B}(R) \times \Pi_{A}(S) - R\right]$$

Page No.
Date. 06/09/23

QUERY LANGUAGE (CONTINUED)

Q1. Consider the following three relations in a relational database: Employee (eId, Name), Brand (bId, bName), Own (eId, bId). Which of following relational algebra expressions return the set of eIds who own all the brands?

Ans $\Pi_{eId}(\Pi_{eId,bId}(OWN) / \Pi_{bId}(Brand))$

$\Pi_{EID}(OWN) - \Pi_{eID}((\Pi_{EID}(OWN) \times \Pi_{bID}(BRAND)) - \Pi_{eID,bID}(OWN))$

Q2. Consider the two relation Suppliers and Parts are given below.

Suppliers

Sno	Pno
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

Parts

Pno.
P_2
P_4

Step 1: $S \div P_2, P_4$

S/P_2

S_1
S_2
S_3
S_4

Step 2: $/P_4$

S_1
S_4

Ans

$\Pi_{Sno, Pno}(Suppliers) / \Pi_{Pno}(Parts)$

The number of tuples are there in the result when the above relational algebra query executed is 2.

Q3. Retrieve all Students who are Enrolled in every Course?

Enrolled

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_3
S_3	C_1

Course

Cid
C_1
C_2
C_3

Enrolled / C_1, C_2, C_3

E/C_1

S_1
S_2
S_3

$/C_2$

S_1

$/C_3$.

S_1

(Ans

Q4 $\Pi_{AB}(R) / \Pi_{B}(S)$ If R has m tuple and S has n tuple then what is minimum and maximum no of tuple in o/p?

Page No.
Date. / /

Assume R and S both are non empty.

Ans Minimum = 0

Maximum = $\left\lfloor \frac{m}{n} \right\rfloor$

Q2. $\Pi_{AB}(R) / \Pi_{B}(S)$ if R has 'm' tuple and S has n tuple then what is minimum and maximum no of tuples in the output? (Assume R non empty and S is empty)

Ans Minimum : 1

Maximum : m.

$$\frac{\Pi_{AB}(R)}{\Pi_{B}(S)} = \underbrace{\Pi_{A}(R)}_{\text{'m'}} - \underbrace{\left[\Pi_{A}\left[\Pi_{A}(R) \times \underbrace{S}_{\text{'0'}} - R \right] \right]}_{\text{Empty}(\phi)}$$

('0' → Empty, Empty − R → Empty)

SUMMARY

OPERATION	PURPOSE	NOTATION
Select	Select all tuples that satisfy selection Condition from relation R.	$\sigma_{\langle \text{selection condition} \rangle}(R)$
Project	Produces a new relation with only some of the attributes of R and removes duplicate tuples.	$\Pi_{\langle \text{attribute list} \rangle}(R)$
Theta Join	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\langle \text{condition} \rangle} R_2$
Equi Join	Produces all combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{C_1} R_2$ or $R_1 \bowtie_{C_1, C_2} R_2$
Natural Join	Same as equijoin except that the join attributes of R_2 are not included in the resulting relation. If the join attributes have same	$R_1 * \langle \text{condition} \rangle R_2$ or $R_1 * \langle \text{join attributes 1} \rangle \langle \text{join attributes 2} \rangle R_2$ or $R_1 * R_2$

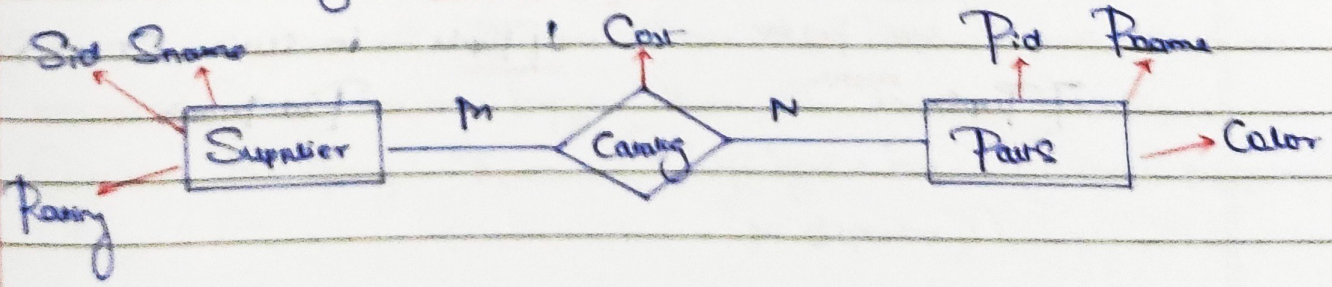
	names, they do not have to be specified at all	
Union	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2. R_1 and R_2 must be Union Compatible	$R_1 \cup R_2$
Intersection	Produces a relation that includes all the tuples in both R_1 and R_2. R_1 and R_2 must be union Compatible	$R_1 \cap R_2$
Difference	Produces a relation that includes all the tuples in R_1 that are not in R_2. R_1 and R_2 must be Union Compatible	$R_1 - R_2$
Cartesian Product	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples as combinations of tuples from R_1 and R_2	$R_1 \times R_2$
Division	Produces a relation $R(x)$ that includes all tuples $t[x]$ in $R_1(z)$ that appear in R_1 in combination with every tuple from $R_2(y)$ where $z = x \cup y$	$R_1(z) \div R_2(y)$

Q. Consider the database with relations:

Supplier (Sid, Sname, Rating)

Parts (Pid, Pname, Color)

Catalog (Sid, Pid, Cost)



(i) Find the Sid of Supplier whose, rating greater than 9?

Ans $\Pi_{Sid}\left[\sigma_{Rating>9}(Supplier)\right]$

(ii) Find the Pid of red color Parts

Ans $\Pi_{pid}\left[\sigma_{Color='Red'}(Parts)\right]$

(iii) Find the Supplier whose Cost is greater than 20,000?

Ans $\Pi_{Sid}\left[\sigma_{Cost>20,000}(Catalog)\right]$

(iv) Retrieve Sid of Supplier who supplied Some Red color Parts.

→ We need to use Parts, Catalog table, In parts table we find Pid Color='Red' then that pid match with Catalog table pid then Project (find) Sid.

$$\Pi_{Sid}\left[\sigma_{\substack{C.Pid=P.pid \\ P.color='Red'}} \wedge (Catalog \times Parts)\right]$$

(v) Retrieve Sname of Supplier who Supplied Some Red color parts?

Ans $$\Pi_{Sname}\left[\sigma_{\substack{P.pid=C.pid \wedge \\ C.Sid=S.sid \wedge \\ P.color=Red}} \left[Suppliers \times Parts \times Catalog\right]\right]$$

Practical Example:

Parts — 1000 Tuple — Total 'Red' — 11 Tuple — Color 'Red' but Pid not match — 7 Tuple

Catalog — 500 Tuple — (P.pid = C.pid) — 200 Tuple in which Pid match — 4 Tuple → in which Color Red and Pid match

Page No.
Date. / /

Retrieve Sid of Supplier who Supplied Some red Color Parts?

Ans

Query I : 4 $\quad \Pi_{Sid}\left[\sigma_{\substack{P.pid=C.Cid \\ P.color='Red'}}\left[P_{400} \times C_{300}\right]\right]$

Query II : 4 $\quad \Pi_{Sid}\left[\sigma_{P.pid=C.pid}\left[\sigma_{Color='Red'}(Parts) \times Catalog\right]\right]$ (11 × 500 = 5500)

Query III : 4 $\quad \Pi_{Sid}\left[\sigma_{Color=Red}\left[Parts \underset{Color=Red}{\bowtie} Catalog\right]\right]$ (200 tuple)

Query IV : 4 $\quad \Pi_{Sid}\left[\sigma_{Color=Red}(Parts) \bowtie Catalog\right]$ (11 tuple)

(Most efficient Query)

Note:

* Let an attribute A belongs to R only then

$\sigma_{A='a'}(R \bowtie S) = \sigma_{A='a'}(R) \bowtie S \rightarrow$ more efficient query

* Let an attribute A belongs to R only and attribute B belongs to S only then

$\sigma_{A='a' \wedge B='b'}(R \bowtie S) = \sigma_{A='a'}(R) \bowtie \sigma_{B='b'}(S) \rightarrow$ most efficient Query.

Q2 Consider the following relation schemas:

b-Schema = (b-name, b-city, assets)

a-Schema = (a-num, b-name, bal)

d-Schema = (c-name, a-number)

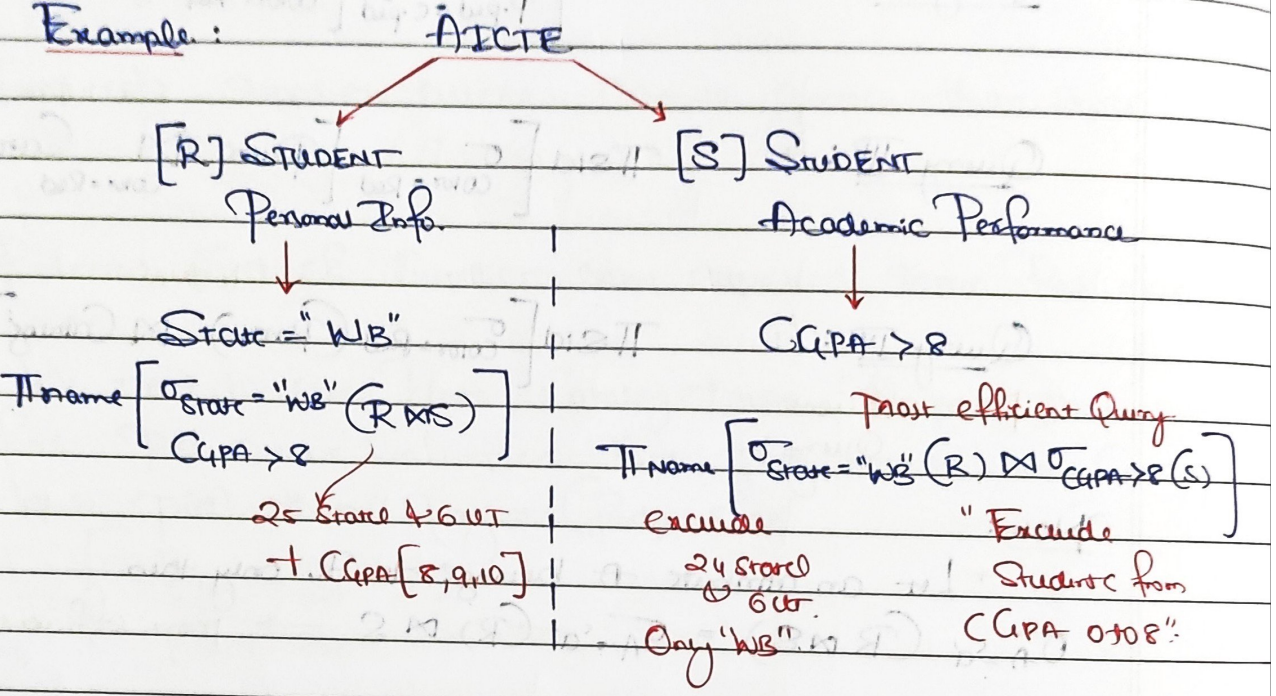
Let branch, account and depositor be respectively instances of the above schemas. Assume that the account and depositor relations are much bigger than the branch relations

Consider the query: $\Pi_{c\text{-name}}(\sigma_{b\text{city}='agra' \wedge bal<0}(branch \bowtie account \bowtie depositor))$.

Which of the following queries is the most efficient version of the above query?

Ans $\Pi_{Cname}(\sigma_{bcity="agra"}(branch \bowtie (\sigma_{bal<0}(account)) \bowtie depositer))$

Example:



Page No.
Date. 06/09/23

STRUCTURED QUERY LANGUAGE

SQL (Structured Query Language)

- DDL (Data Definition Language): Modifications allowed at Schema (Definition) level
  - Create, Alter, Drop Table
- DML (Data Manipulation Language): Modifications allowed at data level
  - Insert, Update, Delete.
- DCL (Data Control Language): Control Transactional Operations
  - Commit, Abort.
- DQL (Data Query Language): Used to retrieve the data from DB.
  - Select, From, Where.

SQL		RA
SELECT [DISTINCT] $A_1, A_2 \dots A_n$ (Optional Clause)	=	Projection [π]
FROM $R_1, R_2, R_3 \dots R_n$	=	Cross Product [$\times$]
WHERE Condition [Predicate]	=	Selection [σ]

Relational Algebra: $\pi_{A_1 A_2 \dots A_n}\left(\sigma_{\text{Condition}}\left(R_1 \times R_2 \times R_3 \dots R_n\right)\right)$

Note:
* RA is a procedural query language where as SQL is non procedural, declarative query language
* SQL is not case-sensitive (Keywords) but Strings are case sensitive
* SELECT in SQL, retain duplicate where as π[projection] in RA by default eliminates duplicates
* Number of SQL clause that are mandatory is "two", that are "SELECT" and "FROM".

Page No.
Date. 07/09/23

SQL Clause

SELECT [DISTINCT] A1, A2 ... An [ ] ← Optional Clause.
FROM R1, R2 ... Rn
[WHERE P]
[GROUP BY ATTRIBUTE [HAVING Condition]]
[ORDER BY Attribute [DESC]]

Query Execution:

1. **FROM Clause**: It is the first executable clause. It just simply relation or Cross Product of two or more relation.
2. **WHERE Clause**: It is the second executable clause. It select the tuple based on specified condition.
3. **GROUP BY Clause**: It is the third executable clause if used in the query. It groups the table based on the specified attribute.

Q1. Why SQL is most widely used?

Ans. Because Relational Algebra works on 'Set Theory' and SQL works on multi-set (Bag) where duplicate are allowed

eg: * Set [1, 2, 3, 4]
* multiset [1, 1, 2, 3, 3, 4]

Q2. Commercial Database works on Multiset (Bag) why?

Ans If we want to perform Union of two table (Relation), SQL just simply One table and simply add the data of another table (∵ it's fastest). But in relation we first eliminate duplicate then perform operation (very time consuming).

* In R.A it eliminate the duplicate so we cannot find the answer of multiple tuple value and removing duplicate is expensive
* In SQL aggregate functions are allowed but it not allowed in relational algebra!
* SQL have lots of feature that are not supported by R.A