GATE CSE NOTES

by

UseMyNotes

* Topics.

→ ER Diagram
→ Conversion of E-R Diagram into relational model
→ Basics of relational model & functional dependency
→ Idea about keys / types
→ Normalization (INF – BCNF)
→ Lossless decomposition & dependency preserving
→ Indexing & Physical Structure (B, B^+ tree)
→ SQL, R.A., R.C. [Relational algebra, calculus]
→ Transition
→ Concurrency Control.

* Data : Raw & isolated facts about an entity (recorded)

Information : Processed, meaningful, usable data.

Database : Collection of smilar / related data

DBMS : S/W used to create, manipulate & delete database.

* Disadvantages of File System.

1. Data redundancy
2. Data inconsistency
3. Difficulty in access
4. Data isolation
5. Security problem
6. Atomicity problem
7. Concurrent access anomalies
8. Integrity problem.
9. Representing complex relationships among data
10. Providing multiple user interfaces.

* OLAP

Online analytical processing

→ historical data
→ subject oriented
→ decision making
→ size in TBs & PBs
→ dealt by CEO, MD, GM.
→ only read

OLTP

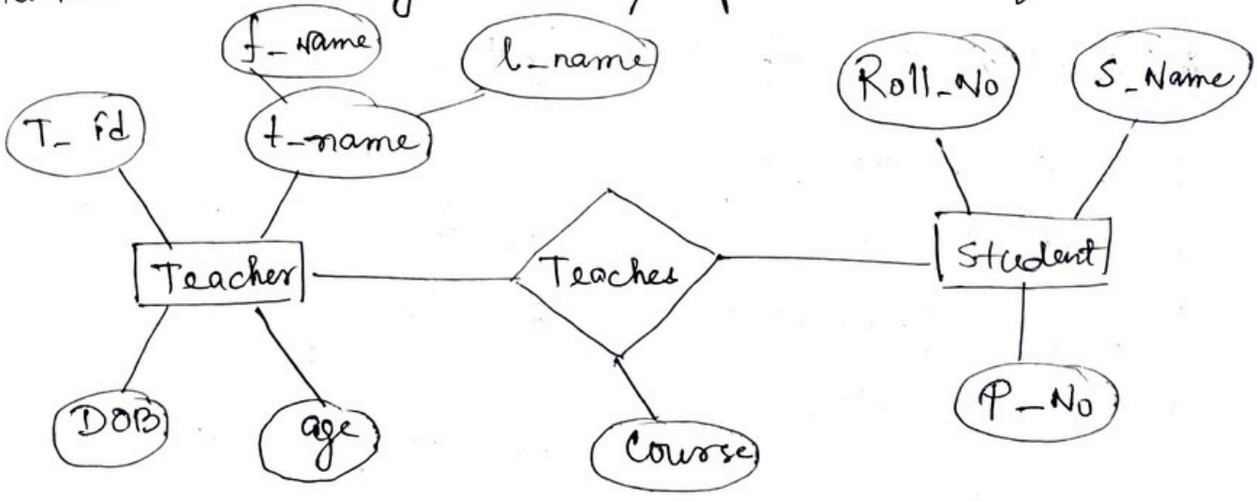
Online transaction processing

→ current data
→ application oriented
→ day to day operations
→ size in MBs, GBs
→ dealt by clerks, managers.
→ read / write.

* Entity Relationship Diagram (ER Diagram)

(Intro, Entity & entity set, attributes & types, relationship, strong & weak entity set, traps, conversion)

Non-technical design method that works on conceptual level based on the perception of real world. Consists of collection of basic objects called entities & of relationships among these objects & attributes that define their properties. Free from ambiguities & provides a standard & logical way of visualising data.



Entity: An entity is a thing or an object in the real world that is distinguishable from other objects based on the values of the attributes it possesses.

Types of entities

→ **Tangible** : Entities which physically exists in real world.
eg. car, pen, bank locker.

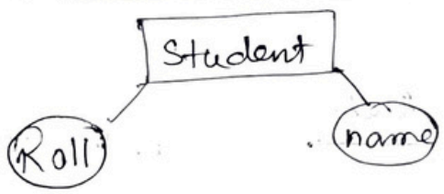
→ **Intangible** : Entities which exist logically.
eg. bank account

Entity set: Collection of same types of entities i.e. that share same properties or attributes called entity set.

ER Diagram

→ Entity can't be represented in an ER diagram as it is instance / data.

→ Entity set is represented by rectangle in an ER diagram.



Relational model:

→ Entity can be represented in an relational model by row / tuple / record.

→ Entity set is represented by table in relational model.

Student

Name	Roll
A	1
B	2
⋮	⋮

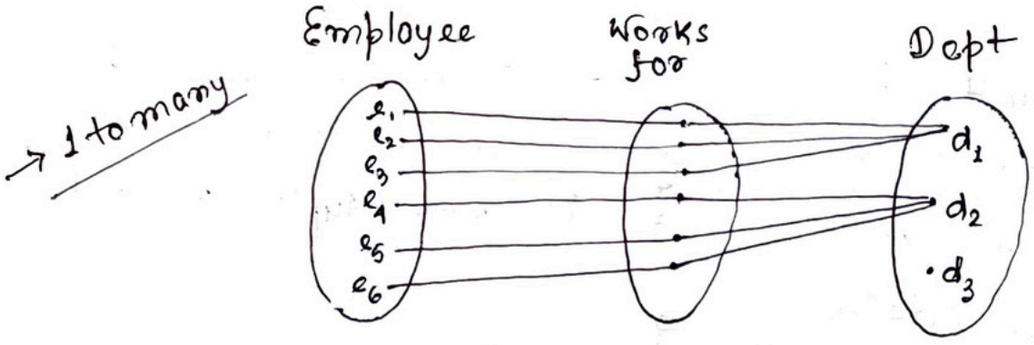
* Attributes.

Classifications –

1. Composite & Simple
2. Single valued & multi valued.
3. Stored & derived.
   (DOB) (Age ← DOB)
4. Complex attributes (composite & multivalued)

* Requirement analysis

Every employee works for exactly a dept., & a dept. can have many employees. New dept. need not have any employee.

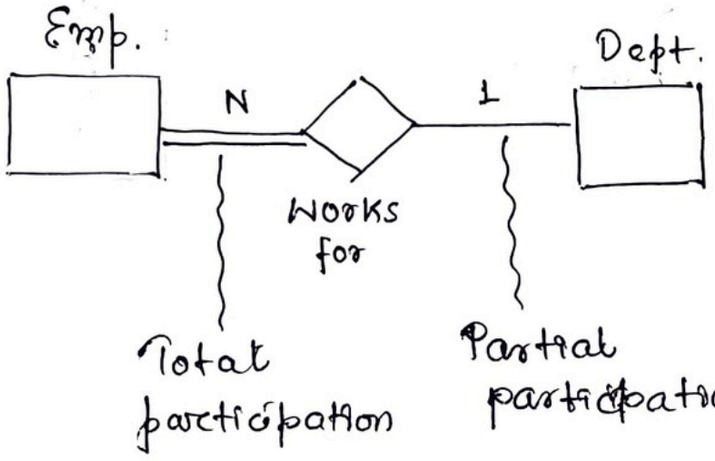


→ Degree = 2 (2 entities)

→ Cardinality ratio (Max) = 1e Nd.

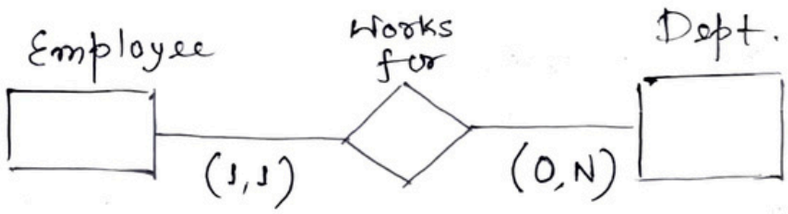
→ Participation or existence (Min) = 1e 0d

e - employee
d - dept.



Cardinality ratio / single-double line representation

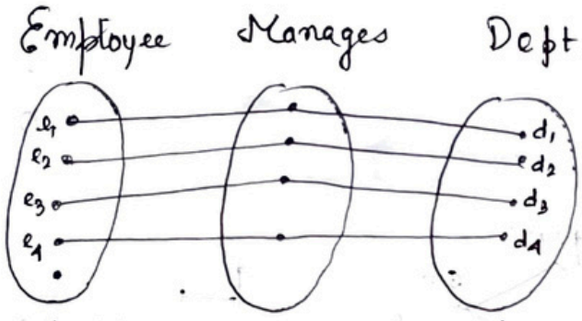
→ 1 to many relationship.



min-max representation.

→ One to one relationship.

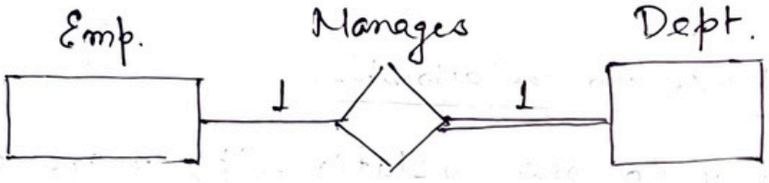
Every dept should have a manager & only one employee manages a dept.



Degree 2.

Cardinality 1 1

Participation 0 1



→ Many to many relationship

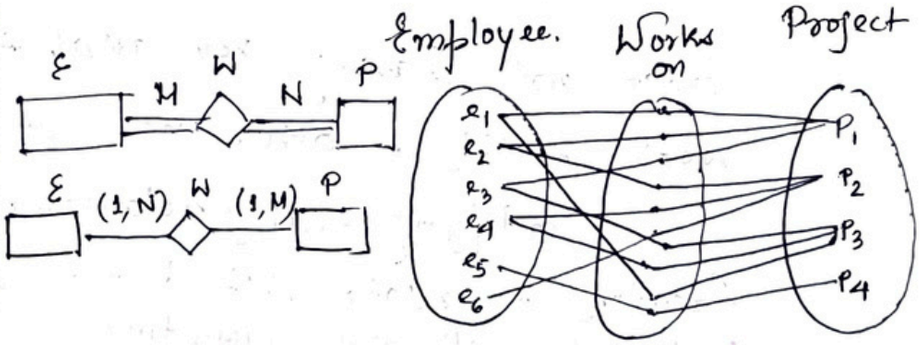
Each employee works on atleast 1 & at max all projects. Each project can have at least 1 & at max all employees.

P_1 : e_1, e_2, e_3

P_2 : e_2, e_4, e_6

P_3 : e_3, e_4, e_1

P_4 : e_5

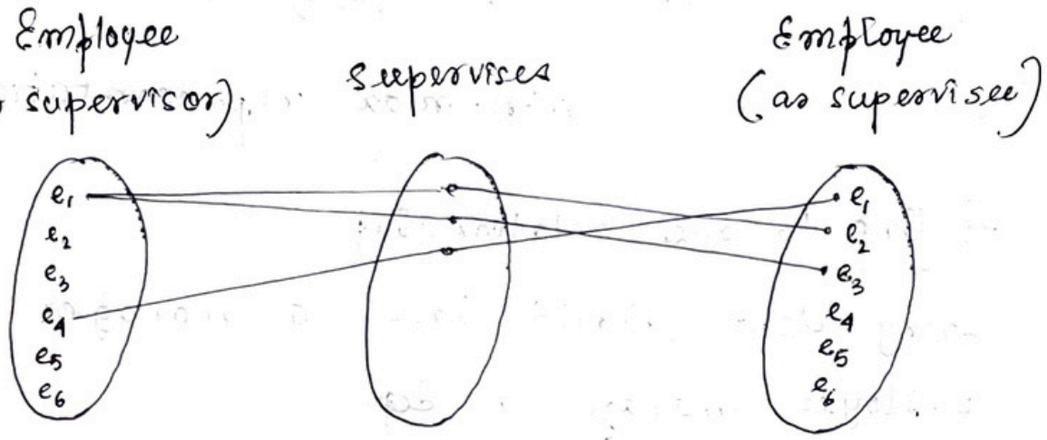


Degree 2

Cardinality N M

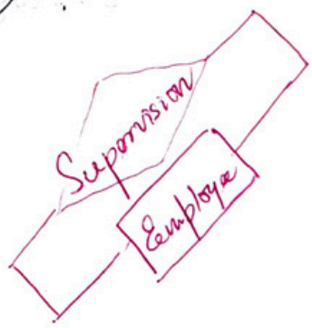
Participation 1 1

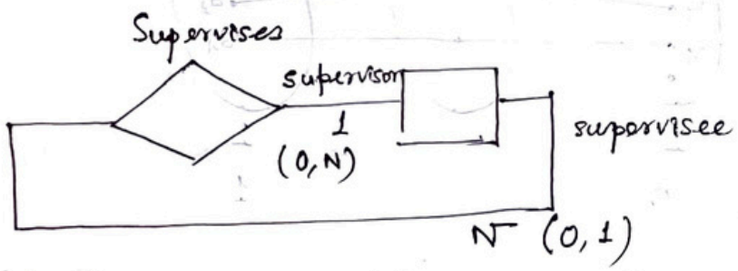
→ Recursive Relationships.



Degree 2

Cardinality	N	1
Participation	0	0





→ Attributes to relationship.

For many to one relationship, more attribute to the many (N) side.

For one to one, anywhere.

For many to many, shouldn't move attribute to entities. Keep it to relationships.

* Weak Entity.

Key attribute – uniquely identifies the entities.

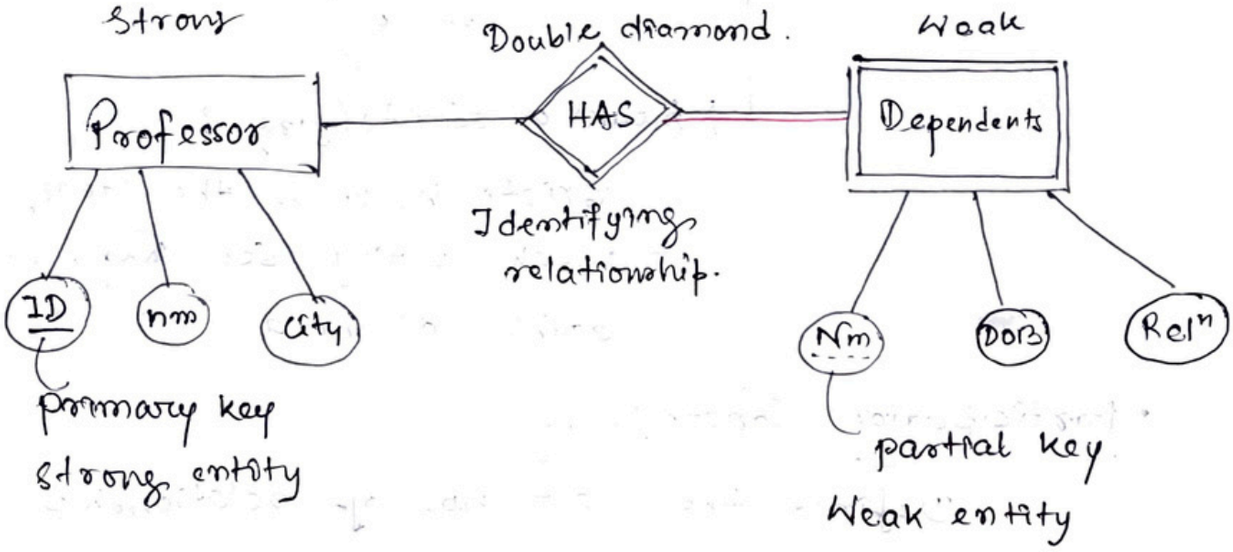
Entity not having key attributes is called weak entity, otherwise strong entity.

Identifying entity relationships are used to identify a relationship between strong & weak entity.

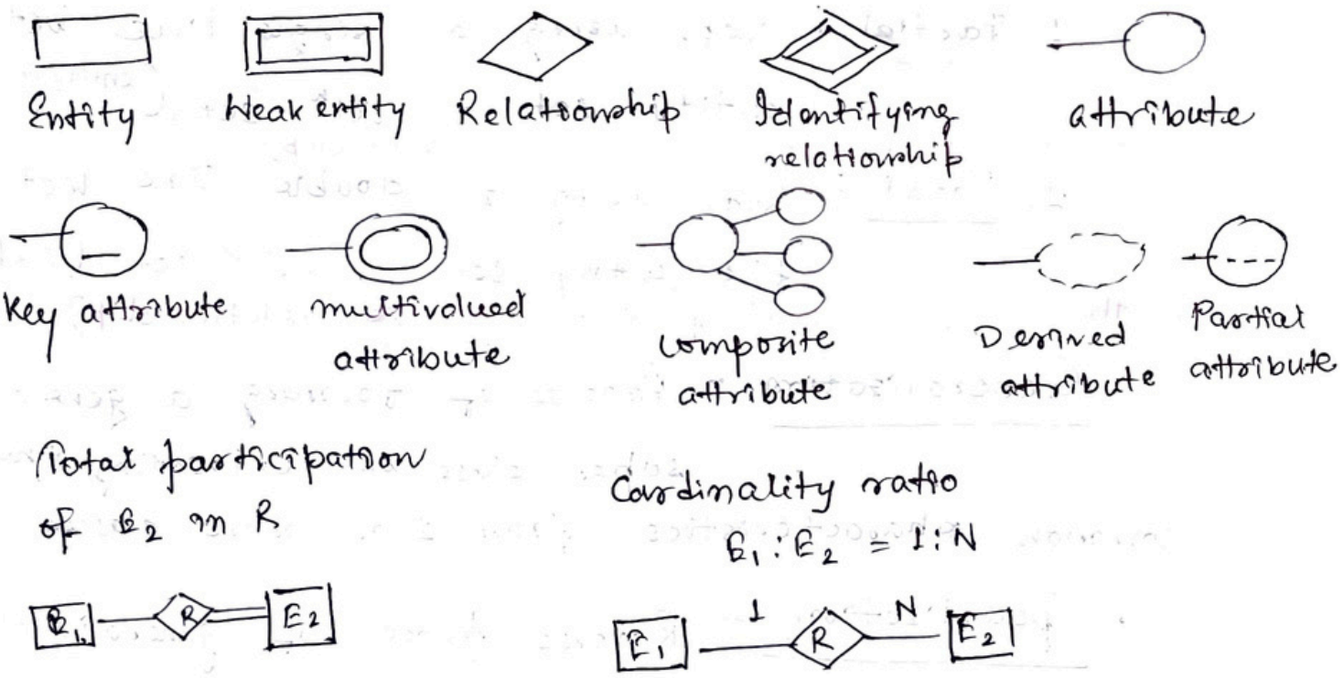
Participation of weak entity type is always total. Relationship between weak entity type & its identifying strong entity type is called identifying relationship (double diamond).

eg. Professor — Strong entity

Professor-dependents — Weake entity



ER diagram notations.



Total participation of E_2 in R

Cardinality ratio

$E_1 : E_2 = 1 : N$

- Components of ER Diagram.

i) Entity sets ii) Attributes iii) Relationship set

- Entity set → Strong : Possesses its own primary key. Represented using a single rectangle.
  → Weak : Does not possess own primary key. Rep. using a double rectangle.

- Relationship sets
  - Strong: Strong relationship exists between 2 strong entity sets. (Diamond symbol)
  - Weak or Identifying: Exists between the strong & weak entity set. Rep. using double diamond.

- Participation Constraints. (min)

Defines the least no. of relationship instances in which an entity has to necessarily participate.

1. Partial: Rep. using a single line between entity set & rel^n set. (entity in the entity set may or may not participate in the relationship)
2. Total: Rep. using a double line between the entity set & rel^n set. (each entity in the entity set must participate in the relationship)

- Generalisation. - Process of forming a generalised super class by extracting the common characteristics from 2 or more classes.

- Specialization. - Reverse process of generalization where a super class is divided into sub classes by assigning the specific characteristics of subclasses to them.

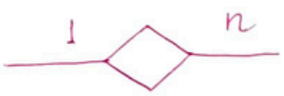
- Cardinality Constraints/ratios. (max)

Defines the max. no. of relationship instances in which an entity can participate.

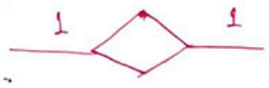
i) many to many (m:n)

ii) many to one (m : 1)

iii) one to many (1 : n)



iv) one to one. (1 : 1)



• A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities. It contains a partial key called discriminator. Discriminator can identify a group of entities from the entity set.

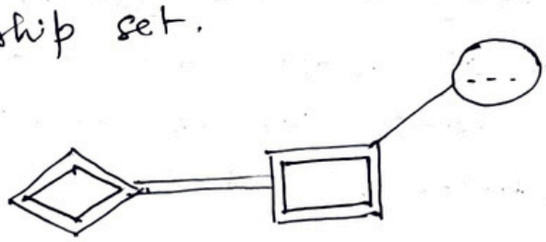
The combination of discriminator & primary key of the strong entity set makes it possible to uniquely identify all entities of the weak entity set.

Primary key of weak entity set	=	Its own discriminator	+	Primary key of strong entity set.

• Total participation may or may not exist in the strong entity set relationship.

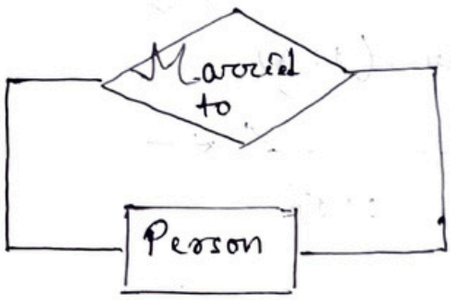
Always exists in the identifying relationship.

So, in ER diagram, weak entity set is always present in total participation with the identifying relationship set.



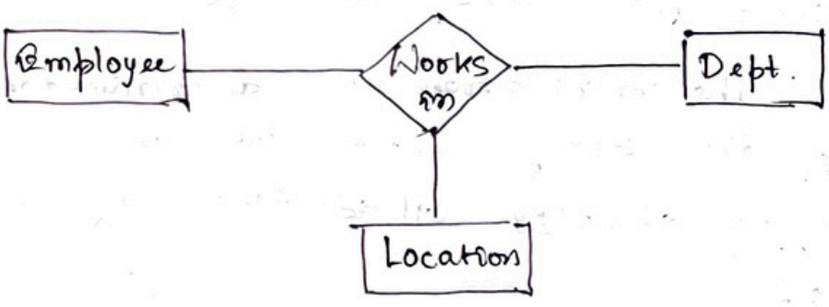
• Degree of a relationship set. : No. of entity sets participating in a relationship set.

- Unary relationship: One person married to only one person.



Binary relationship: Student enrolled in a course.

Ternary relationship:



- Cardinality Constraints: Max. no. of relationship instances in which an entity can participate.

i) m:n : An entity in set A can be associated with any no of entities of set B. Same for B.

ii) m:1 : An entity in set A can be associated with at most one entity in set B. An entity in set B can be associated with any no. of entities in set A.

iii) 1:m : Opposite of m:1.

iv) 1:1 : An entity in set A can be associated with at most one entity in set B. Same for B.

• Participation constraints.

Least no. of relationship instances in which an entity must compulsorily participate.

i) Total participation. : Each entity in the entity set must compulsorily participate in at least ~~once~~ one relationship instance in the relationship set. (mandatory participation)

ii) Partial participation: Each entity ~~set~~ in the entity set may or may not participate in the relationship instance in that relationship set. (optional participation)

(i.e participation).

• Minimum cardinality tells whether the participation is partial or total.

- If minimum cardinality = 0, then it signifies partial participation.

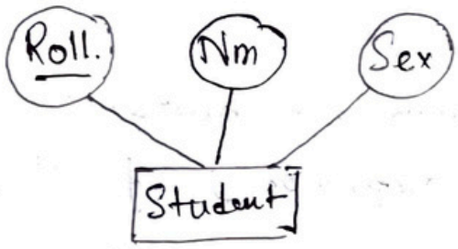
- If minimum cardinality = 1, then it signifies total participation.

Maximum cardinality tells the max. no. of entities that participates in a relationship set.

* Converting ER diagram to tables.

ER diagram is converted into the tables in relational model. This is because relational models can be easily implemented by RDBMS like MySQL, Oracle etc.

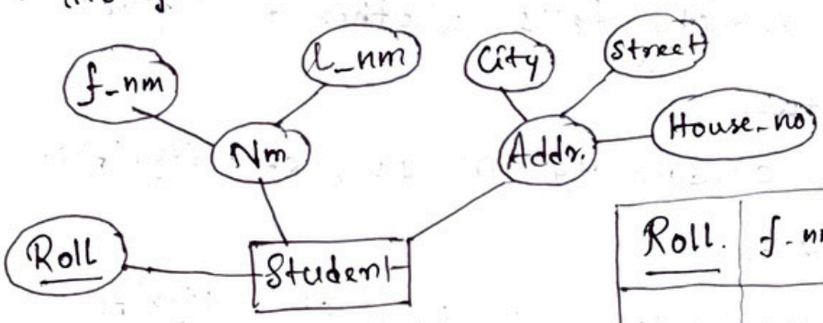
Rule 1: A strong entity set with only simple attributes will require only one table in Relational model. Attributes of the table will be the attributes of entity set. The primary key of the table will be the key attribute of the entity set.



Roll	Nm	Sex

Schema: Student

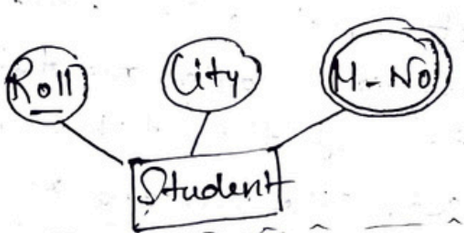
Rule 2. A strong entity set with any no. of composite attributes will require only one table in relational model. While conversion, simple attributes of the composite attributes are taken into account & not the composite attribute itself.



Roll.	f_nm.	l_nm	city	street	house_no

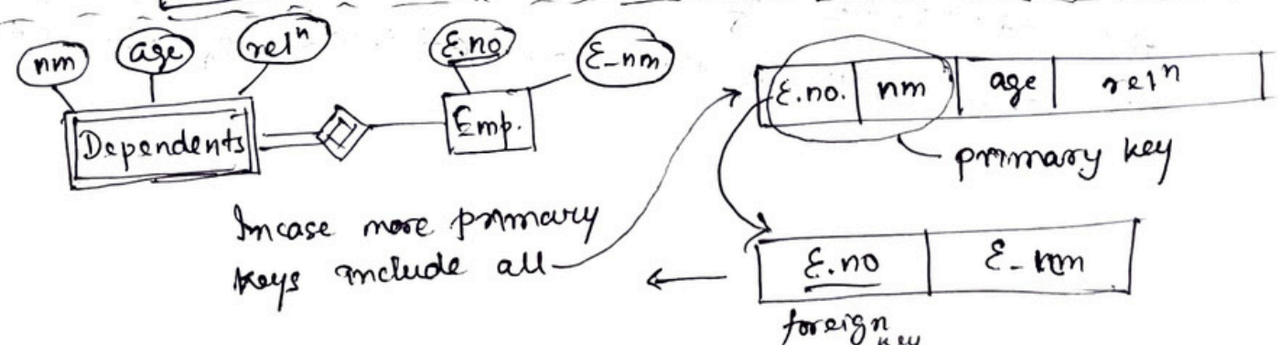
Schema : Student.

Rule 3. A strong entity set with any no. of multivalued attributes will require 2 tables in relational model. One table will contain all the simple attributes with the primary key. Other table will contain the primary key & all the multivalued variables.



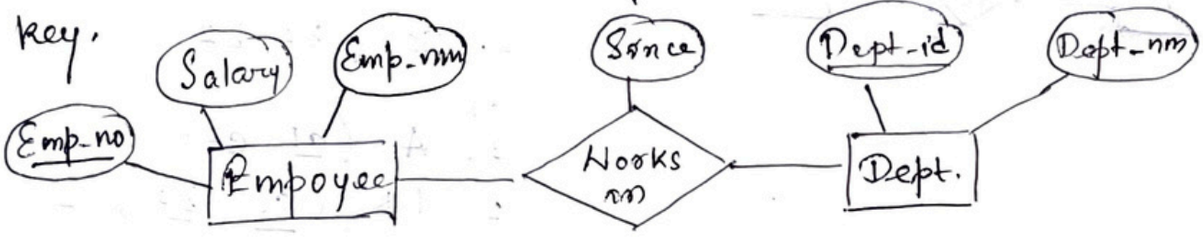
Roll	City

Roll	M_No



Rule 4. Translating relationship set into a table.

A relationship set will require one table in the relationship model. Attributes of the table are primary key attributes of the participating entity sets, its own descriptive attributes if any. Set of non-descriptive attributes will be the primary key.



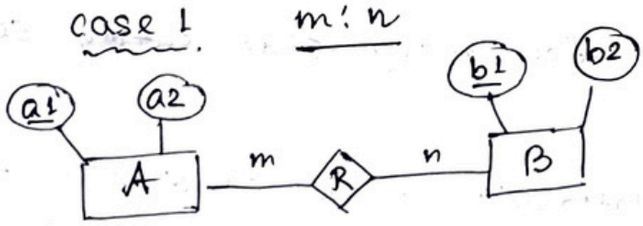
Emp_no	Dept_id	since

Schema: Works in

If we consider the overall ER diagram, three tables will be reqd. in relational model. One table for entity set 'Employee', one for 'Dept', one for rel. set 'Works in'.

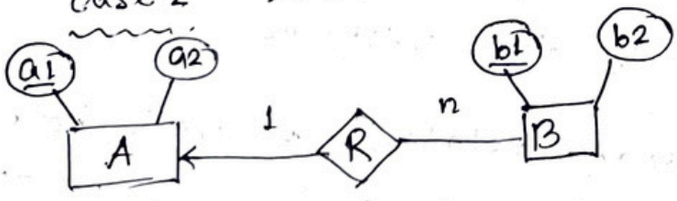
Rule 5. For binary relationships with cardinality ratios

case 1 m:n



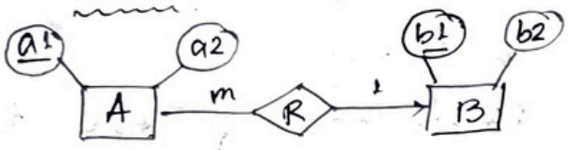
Three tables –
A (a1, a2)
R (a1, b1)
B (b1, b2)

case 2 1:n



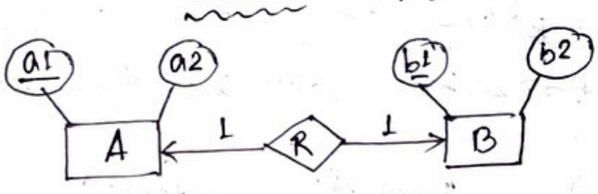
2 tables –
A (a1, a2)
BR (a1, b1, b2)
combined table

case 3 m:1



2 tables –
1) AR (a1, a2, b1)
2) B (b1, b2)

case 4 1:1



2 tables
1. AR (a1, a2, b1)
2. B (b1, b2)

or
1. A (a1, a2)
2. BR (a1, b1, b2)

• While determining the minimum number of tables reqd. for binary relationships with given cardinality ratios, following thumb rules must be kept in mind –

→ For binary relationship with cardinality ratio m:n, separate & individual tables will be drawn for each entity set & relationship.

→ For binary relationship with cardinality ratio either m:1 or 1:n, always remember many side will consume the relationship. i.e. a combined table will be drawn for many side entity set & relationship set.

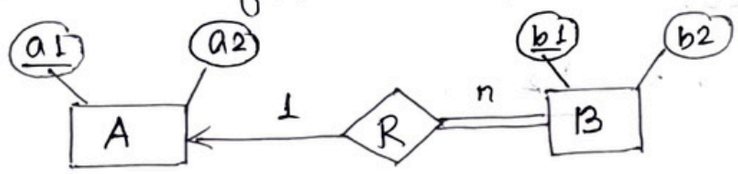
→ for binary relationship with cardinality ratio 1:1, two tables will be reqd. Combine the relationship set with any one of the entity sets.

Rule 6 For binary relationship with both cardinality constraints & participation constraints –

Cardinality constraints will be implemented as discussed in rule 5. Because of the total participation constraint, foreign key acquires NOT NULL constraints. i.e. now foreign key cannot be null. (Foreign keys are

the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.)

case-1. For binary relationship with cardinality constraint & total participation constraint from one side -



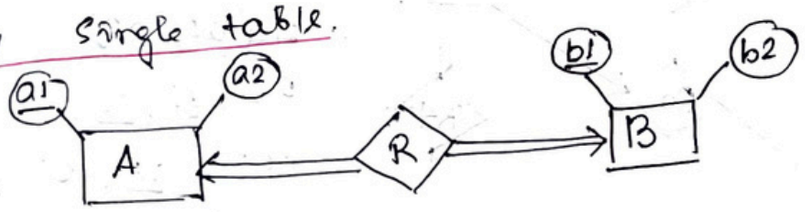
Because cardinality ratio = 1:n, so we will combine the entity set B & relationship set R. Two tables reqd. -

1. A (a1, a2)
2. BR (a1, b1, b2)

Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.

case-2 for binary relationships with cardinality constraint & total participation constraint from both sides -

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.



ARB (a1, a2, b1, b2)

Rule 7. For binary relationship with weak entity set.

Weak entity set always appears in association with identifying relationship with total participation constant.

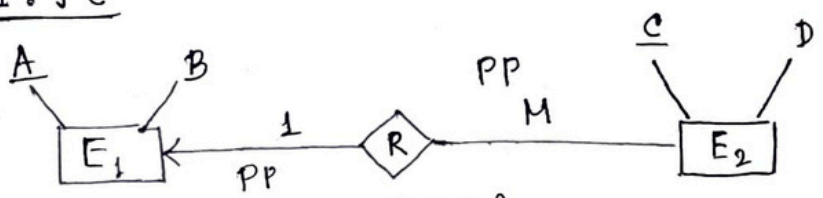
2 tables reqd.

i) A (a1, a2)

ii) BR (a1, b2, b1)

Minimum RDBMS.

① 1 : MC



(A C)

C → A

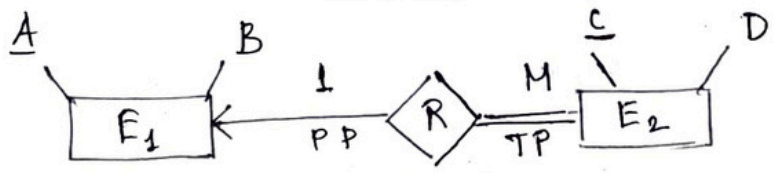
C → D.

E_1 (A B) E_2 R (C D A) FK

[Should allow NULL]

Min 2 tables & 1 FK.

② 1 : M



R(A C)

C → A

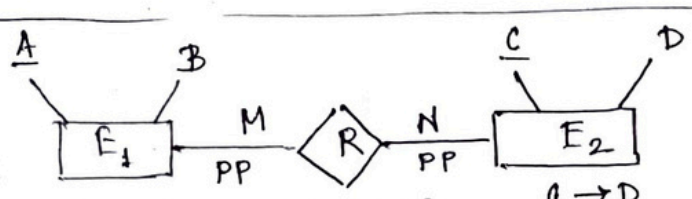
E_1 (A B) E_2 R (C D A) [No null]

A → B

C → DA

Min 2 tables & 1 FK.

③ M : N



A → B

R (A C)

C → D

No nontrivial FD

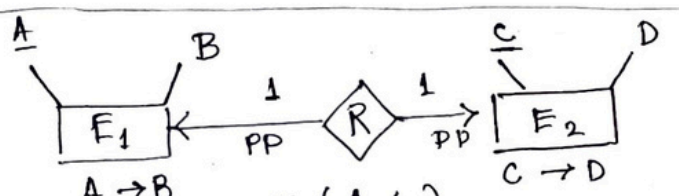
E_1 (A B) R (A C) E_2 (C D

FK

Min 3 tables

2 FKs.

④ 1 : 1



A → B

R (A, C)

A → C, C → A

C → D

(i) E_1R (A B C) E_2 (C D)

A → B

C → A

C → D.

(ii) E_1 (A B) E_2R (C D A)

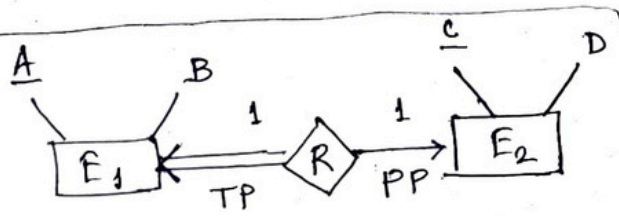
A → B

C → DA

A → C

Min 2 tables, 1 FK.

⑤ 1 : 1



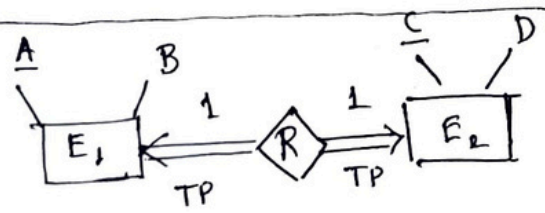
E_1RE_2 (A B C D)

A → BC

C → AD

Min 1 table, no FK.

⑥ 1 : 1



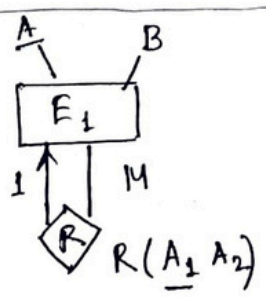
E_1RE_2 (A B C D)

A → BC

C → AD

Min 1 table, no FK.

⑦ 1 : M

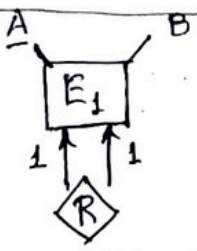


R(A1 A2)

E_1R (A B A1)

Min 1 table, 1 FK.

⑧ 1 : 1



R (A1, A2)

E_1R (A B A1) {A, A1} candidate key

Min 1 table,

1 FK.

(9)

M:N

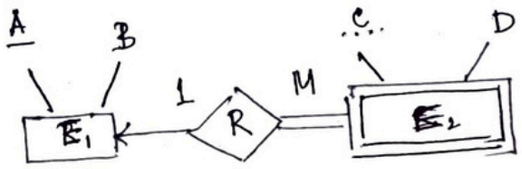
E_1 ($\underline{A}$ $\underline{B}$) R ($\underline{A_1}$ $\underline{A_2}$)

Min 2 tables

2 FKs

---

(10)



1:M

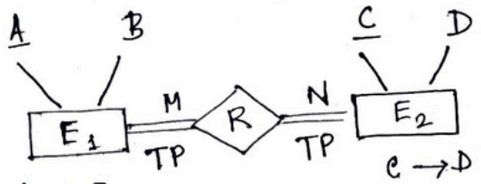
E_1 ($\underline{A}$ B) E_2 R ($\underline{A}$ $\underline{C}$ D)

Min 2 tables, 1 FK.

---

(11)

m:n



R ($\underline{A}$ $\underline{C}$)

No nontrivial FD

Min Rel^n for 1NF

E_1 R E_2 (A B C D)

A → B C → D

$\underline{AC}$ not 2NF

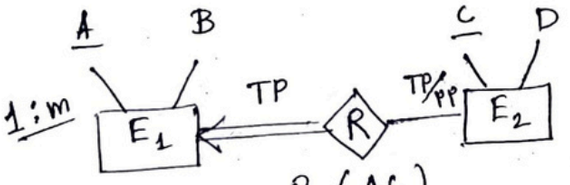
Min Rel^n for BCNF

E_1 ($\underline{A}$ $\underline{B}$) R ($\underline{A}$ $\underline{C}$)

A → B

E_2 ($\underline{C}$ D)

C → D



R ($\underline{A}$ $\underline{C}$)

E_1 R (A B C D)

A → B, C → A D

$\underline{C}$: Key

1NF ✓

2NF ✓

3NF X

E_1 ($\underline{A}$ B) A → B

E_2 R ($\underline{C}$ A D) C → A D

- For not applicable or unknown values in table we set that NULL.

* Relational Constraints.

Restrictions imposed on the database contents & operations. They ensure the correctness of data in the database.

Types -

1) Domain constraint: Value taken by the attribute must be the atomic value from its domain.

2) Tuple uniqueness constraint: All the tuples must be necessarily unique in any relation.

3) Key constraint: All the values of primary key must be unique.
The value of primary key must not be NULL.

4) Entity integrity: No attribute of primary key ~~must~~ should contain a null value in any relation. Presence of null value in the primary key violates the uniqueness property.

5) Referential integrity constraint:
This constraint is enforced when a foreign key references the primary key of a rel^n. It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.
(Foreign key must reference a valid, existing primary key in the parent table.)

• We can't insert a record into a referencing relation if the corresponding record does not exist in the referenced rel^{n}.

• We can't delete or update a record of the referenced relation if the corresponding record exists in the referencing rel^{n}.

• Referencial Integrity Constraint Violation :

Cause-1. Insertion in a referencing relation –

It is allowed to insert only those values in the referencing ~~relationship~~ attribute which are already present in the value of the referenced attribute.

Inserting a value in the referencing attribute which isn't present in the value of the referenced attribute violates the referential integrity constraint.

Cause-2 Deletion from a referenced relation –

It is not allowed to delete a row from the referenced relation if the referencing attribute uses the value of the referenced attribute of that row. Such deletion violates referential integrity constraint.

Handling this violation –

method 1. Simultaneous deleting those tuples from the referencing relation where the referencing attribute uses the value of referenced attribute being deleted. (On delete cascade)

method 2. Aborting or deleting the request.

method 3. Setting the value to NULL or some other value in the referencing relation if the referencing attribute uses the value.

Cause-3. Updation on a referenced relation.-

It's not allowed to update a row of the referenced relation of the referencing attribute uses the value of the referenced attribute of that row. Such updates violate referential integrity comtraint.

Handling this violation-

1. Simultaneous update of those tuples of the referencing rel^n where the referencing attribute uses the referenced attribute value being updated. (On update cascade)

2. Aborting or deleting the request.

3. Setting the value to NULL or some other value.

*Keys. A key is a set of attributes that can identify each tuple uniquely on the given relation.

| Relation ≡ Table
| Tuple ≡ Record

Different keys -

1. Super key: Set of attributes that can identify each tuple uniquely on the given relation. A super key is not restricted to have any specific no. of attributes. A super key may consist of any no. of attributes.

✓ All the attributes in a super key are definitely sufficient to identify each tuple uniquely on the given relation but all of them may not be necessary.

Any superset of a key is a super key.

Every rel^n must have a superkey & in the worst case, whole superkey will be the key.

2. Candidate key : A minimal super key is called as a candidate key. This is a set of minimal attribute(s) that can identify each tuple uniquely in the given rel^n.

All the attributes in a candidate key are sufficient as well as necessary to identify each tuple uniquely. Removing any attribute fails in identifying each tuple uniquely. The value of candidate key must always be unique. The value of candidate key can never be NULL. It's possible to have multiple candidate keys in a relation. Those attributes which appear in some candidate key are called as prime attributes.

There can be multiple CKs for a single rel^n.

eg. Student (roll, name, sex, age, addr., class)

Super keys examples –
(roll, name, sex)
(roll, age, addr.) ... etc.

No. of candidate keys = $2^n - 1$ (possible) [When n attrib.s]

Candidate keys examples –
(class, section, roll)
(name, addr.)

each set consists of minimal attrib.s reqd. to identify each student uniquely

3. Primary Key : A candidate key that the database designer selects while designing the db.

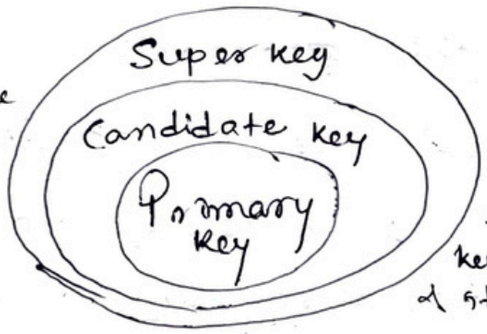
The value of primary key can never be NULL. The value of primary key must always be unique. Values of primary key can never be changed i.e. no updation possible. Value of primary key

In a single rel^n, different candidate keys may have some different set of attributes. (like A, and Be) N=1

must be assigned when inserting a record.

A rel^n is allowed to have only one primary key.

There can only be one primary key for any rel^n in a schema.



One of the candidate keys gets qualified to become a primary key. Rules that a CK must have to become a PK are that the key value should never be null & it must be unique for all tuples.

4. Alternate Key: Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys.

5. Foreign Key: An attribute X is called a foreign key to some other attribute Y when its values are dependent on the values of Y.

The attrib. X can assume only those values which are assumed by Y.

The rel^n in which Y is present is called the referenced rel^n. The rel^n in which X is present is called the referencing rel^n. The attribute Y might be present in the same table or in some other table.

eg.

Teacher (t_no, t_nm, t_age, t_dept)

Department (dept_no, dept_nm)

Referencing rel^n

foreign key

Referenced rel^n.

t_dept can take only those values that are present in dept_no in department table.

Foreign key references the primary key of the table. Foreign key can take only those values that are present on the primary key of the referenced relⁿ. Foreign key may have a name other than that of a primary key. Foreign key can take the NULL value. There is no restriction on a foreign key to be unique. Foreign key is not unique most of the time. Referenced relⁿ may also be called as the master table or primary table. Referencing relⁿ may also be called the foreign table.

6. Partial Key : A key using which all the records of the table cannot be identified uniquely. However, a bunch of related tuples can be selected from the table using partial key.

eg. Dept. (Emp_no, dependent_nm, relation)

Emp_no	dependent_nm	reldtion
e1	Suman	Mother
e1	Ajay	Father
e2	Bijay	Father
e2	Raj	Son

Using partial key Emp_no, we can't identify a tuple uniquely but we can select a bunch of tuples from the table.

7. Composite Key : A primary key comprising of multiple attribs & not just a single attrib.

8. Unique Key: Key with following properties-

- It's unique for all the records of table.
- ✓ Once assigned, its value can't be changed (non-updatable).
- ✓ It may have a NULL value.

eg. Aadhaar card numbers

9. Surrogate Key: Key with properties —

- It's unique for all the records.
- ✓ It's updatable.
- ✓ It can't be NULL.

eg. mobile no. of students in a class where every student owns a mobile phone.

10. Secondary Key: Required for the indexing purpose for better & faster searching.

Q. Schema $R(A_1, A_2, \ldots, A_n)$.

Candidate key $\{A_1\}$.

How many super keys possible?

→ Every superset of a key is a superkey.

A_2 A_3 $\ldots$ A_n

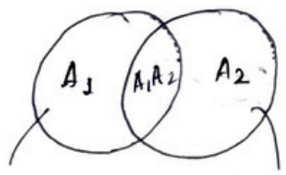
$2 \times 2 \times \ldots \times 2$ n-1 times

$= 2^{n-1}$ no. of superkeys.

Q. $R(A_1, A_2, \ldots, A_n)$ Relation.

Candidate key - $\{A_1, A_2\}$

How many superkeys ?

→ 

Superkeys for A_1 | Superkeys for A_2

$SK(A_1) + SK(A_2) - SK(A_1A_2)$

$= 2^{n-1} + 2^{n-1} - 2^{n-2}$

Q. $R(A_1, A_2, \ldots, A_n)$

$CK = \{A_1, A_2A_3\}$ No. of superkeys ?

→ $SK(A_1) + SK(A_2A_3) - SK(A_1A_2A_3)$.

$= 2^{n-1} + 2^{n-2} - 2^{n-3}$

Q. $R(A_1, A_2, \ldots, A_n)$

$CK = (A_1, A_2, A_3)$

No. of superkeys.?

→ $SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1A_2) - SK(A_2A_3) - SK(A_3A_1) + SK(A_1A_2A_3)$

Q. $R(A, B, CD)$

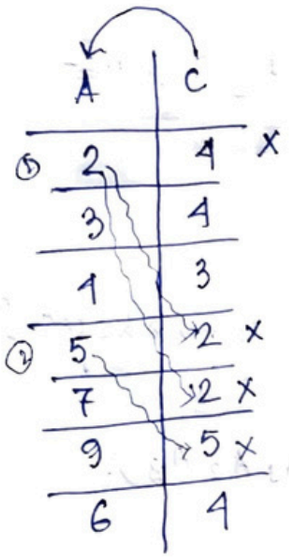
$CK = \cancel{(A_2BC)}$ (A, BC)

No. of superkeys?

→ $8 + 4 - 2$

$= 10.$

eg. A is the primary key & C is the foreign key referencing A with on delete cascade. The set of all tuples that must be additionally deleted to preserve referential integrity when (2,4) is deleted —

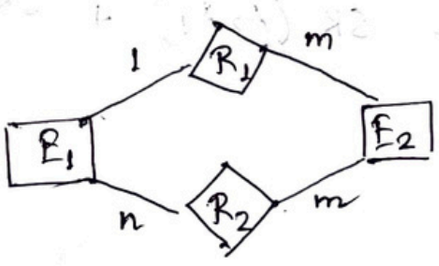


(5,2), (7,2), (9,5)

(On delete cascade in case of foreign keys)

Q. E_1, E_2 two entities. R_1, R_2 are 2 relationships b/w E_1 & E_2. R_1 is one to many & R_2 is many to many. R_1 & R_2 does not have any attributes of their own. What is min. no. of tables reqd.?

→



E_1, E_2 have simple single valued attrib.s. (assume)

E_1
E_2 R_1
1 table for R_2 (min)

} 3 tables

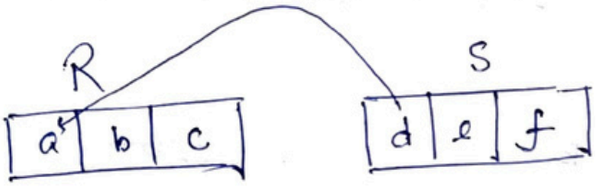
Q.

Let R(a,b,c) & S(d,e,f) be two relations. 'd' is a foreign key of S that references to the primary key of R. Consider following operations on R & S —

a) Insert into R
c) Delete from R
b) Insert into S
d) Delete from S

Which of these can cause violation of referential integrity?

Assume 'a' to be primary key.



* Functional Dependency.

In any relation, a functional dependency $\alpha \rightarrow \beta$ holds if two tuples having same value of attribute α also have same value for attribute β.

If α & β are 2 sets of attribs in a relational table R where $\alpha \subseteq R$ & $\beta \subseteq R$, then for a functional dependency to exist from α to β, following cond^n must be true

$$\text{if } t1[\alpha] = t2[\alpha], \text{ then } t1[\beta] = t2[\beta]$$

$$f_d : \alpha \rightarrow \beta.$$

- FD is a relationship that exists between 2 attributes. It typically exists between the primary key & non-key attribute within a table.

- $X \rightarrow Y$
  - X: determinant
  - Y: dependent

- FD
  - Trivial: $AB \rightarrow A$, $A \rightarrow A$
  - Non-trivial: $A \rightarrow B$ (Non trivial)
  - Semi-non-trivial: $AB \rightarrow BC$

→ Trivial Functional Dependency.

An FD $X \to Y$ is said to be trivial if & only if $Y \subseteq X$. If RHS of FD is a subset of LHS, it is trivial.

$AB \to A$, $A \to A$.

→ Non-trivial functional dependency

$X \to Y$ is non-trivial only if $Y \not\subseteq X$.

If there exists at least one attrib. on the RHS of a functional dependency that is not a part of LHS, then it's called non-trivial.

$AB \to BC$, $A \to B$

Q.

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

Which FDs are satisfied?

i) $XY \to Z$, $Z \to Y$
ii) $YZ \to X$, $Y \to Z$ ✓
iii) $YZ \to X$, $X \to Z$
iv) $XZ \to Y$, $Y \to X$

→ If all determinants (LHS) are different, then that FD is valid on that instance. (eg. $XY \to Z$ here)

* Inference Rule (IR)

The Armstrong's axioms are the basis inference rule. Armstrong's axioms are used to conclude functional dependencies on a relational database. The inference rule is a type of assertion. It can be applied to a set of FD to derive other FD. Using inference rule we can derive additional functional dependency from the initial set.

1. Reflexive rule.

If Y is a subset of X, then X determine Y.

If $X \supseteq Y$, then $X \rightarrow Y$.

2. Augmentation rule / Partial dependency

If $X \rightarrow Y$, then $XZ \rightarrow YZ$.

3. Transitive rule

If $X \rightarrow Y$ & $Y \rightarrow Z$, then $X \rightarrow Z$.

4. Union rule

If $X \rightarrow Y$ & $X \rightarrow Z$ then $X \rightarrow YZ$.

Proof $X \rightarrow Y$, $X \rightarrow Z$

$\Rightarrow X \rightarrow XY$ (augmentation $XX \rightarrow XY$, $X \rightarrow XY$)

$\Rightarrow XY \rightarrow YZ$

Using transitive rule,

$X \rightarrow YZ$.

5. Decomposition rule / Project rule

It is reverse of union rule.

If $X \rightarrow YZ$, then $X \rightarrow Y$ & $X \rightarrow Z$.

proof $X \rightarrow YZ$

$YZ \rightarrow Y$ (reflexive rule)

$X \rightarrow Y$ (Transitive rule)

6. Pseudo transitive rule

If $X \rightarrow Y$ & $YZ \rightarrow W$ then $XZ \rightarrow W$

proof $X \rightarrow Y$ $\quad YZ \rightarrow W$

$\Rightarrow XZ \rightarrow YZ$ (Augmentation)

$XZ \rightarrow W$ (Transitive rule).

* Rules for Functional Dependency

i) **Rule 01** : An FD $X \rightarrow Y$ will always hold if all the values of X are unique irrespective of the values of Y.

attribs A, B, C, D, E

All values of A attrib. are different, then

A → any combination of attrib. A, B, C, D, E.

ii) **Rule 02** : An FD $X \rightarrow Y$ will always hold if all the values of Y are same irrespective of the values of X.

All the values of attrib. C are same.

Any combination of attribs A, B, C, D, E → C

· In general, a functional dependency $\alpha \rightarrow \beta$ always holds if either all values of α are unique or if all values of β are same or both.

iii) For an FD $X \rightarrow Y$ to hold, if 2 tuples in the table agree on the value of attrib. X, then they must also agree on the value of attrib. Y.

iv) For an FD $X \rightarrow Y$, violation will occur only when for 2 or more same values of X, the corresponding Y values are different.

* Closure of an Attribute Set.

The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.

Closure of attribute set $\{X\}$ is denoted as $\{X\}^+$.

Steps to find the closure of an attribute set –

1. Add the attributes contained in the attribute set for which closure is being calculated to the result set.

2. Recursively add the attributes to the result set which can be functionally determined from the attributes ~~attrs~~ already contained in the result set.

eg. Relation $R(A, B, C, D, E, F, G)$. with FDs

$A \to BC$, $BC \to DE$, $D \to F$, $CF \to G$.

$A^+ = \{A\} = \{A, B, C\}$ using $A \to BC$

$= \{A, B, C, D, E\}$ using $BC \to DE$

$= \{A, B, C, D, E, F\}$ using $D \to F$

$= \{A, B, C, D, E, F, G\}$ using $CF \to G$.

$D^+ = \{D\} = \{D, F\}$

$\{B, C\}^+ = \{B, C\} = \{B, C, D, E\} = \{B, C, D, E, F\}$

$= \{B, C, D, E, F, G\}$.

- Finding the keys using closure.

→ Super key. If the closure result of an attribute set contains all the attributes of the relⁿ, then that attrib. set is called a super key of that relⁿ.

Thus, the closure of a super key is the entire relation schema.

→ Candidate key. If there exists no subset of an attribute set whose closure contains all the attributes of the relⁿ, then that attrib. set is a candidate key of that relation.

eg. $R(A, B, C, D, E, F)$

FDs $(C \to F,\ E \to A,\ EC \to D,\ A \to B)$

What can be key?

a) CD b) EC ✓ c) AE d) AC.

Q. $R(E, F, G, H, I, J, K, L, M, N)$

FDs $\{E, F\} \to \{G\}$ $\quad \{K\} \to \{M\}$

$\{F\} \to \{I, J\}$ $\quad \{L\} \to \{N\}$.

$\{E, H\} \to \{K, L\}$

What is key?

a) $\{E, F\}$ b) $\{E, F, H\}$ c) $\{E, F, H, K, L\}$ ✓ d) $\{E\}$.

→ $(E, F, H)^+ = (EFHGIJKLMN)$.

It's a key { candidate key too. (only one.)

Q. $R(A, B, C, D, E, H)$

FDs $\{A \to B,\ BC \to D,\ E \to C;\ D \to A\}$

What are candidate keys?

a) AE, BE c) AEH, BEH, BCH

b) AE, BE, DE d) AEH, BEH, DEH.

→ $(EH)^+ = \{E, H, C\}$

$(AEH)^+ = \{AEHBCD\}$ ✓

$(BEH)^+ = \{\ ,\ \}$.

$(CEH)^+ = \{CEH,\ \}$

$(DEH)^+ = \{ABCDEH\}$ ✓

Start from the attributes that are not already determined by any determinant (as they need to produce themselves)

Q. R (A B C D E F G H) G' 2013 | Normalisation obs

*** FDs = { CH → G, A → BC, B → CFH, E → A, F → EG }

How many candidate keys does the rel^n have?

(4)

→ $D^+ = \{D\}$

- ✓ $(AD)^+ = \{A\,B\,C\,D\,E\,F\,G\,H\}$
- ✓ $(BD)^+ = \{A\,B\,D\,C\,F\,H\,E\,G\}$
- $(CD)^+ = \{C\,D\}$
- ✓ $(ED)^+ = \{D\,E\,A\,B\,C\,F\,H\,G\}$
- ✓ $(FD)^+ = \{D\,F\,G\,E\,A\,B\,C\,H\}$

$(GD)^+ = \{G\,D\}$

$(HD)^+ = \{D\,H\}$

(CD) (GD) (HD)

Add C, H, G.
if add A, B, E, F then will become super keys.

$(GCD)^+ = \{G\,C\,D\}$

$(HCD)^+ = \{H\,C\,D\,G\}$

$(GHD)^+ = \{G\,H\,D\}$

$(CGHD)^+ = (C\,G\,H\,D)$

eg. R (A B C D E)

AB → C, C → D, B → E

✓ $(AB)^+ = \{A\,B\,C\,D\,E\}$ candidate key | 2^3 Super keys

eg. R (A B C D E F)

A → B, C → D, E → F

$(ACE)^+ = \{A\,C\,E\,B\,D\,F\}$ candidate key. | 2^3 Super keys.

eg. R = (A B C D) FDs = { AB → CD, C → A, D → B }

$A^+ = \{A\}$ $B^+ = \{B\}$ $C^+ = \{C\,A\}$ $D^+ = \{D\,B\}$

✓ $AB^+ = \{A\,B\,C\,D\}$ ✓ $BC^+ = B\,C\,A\,D$

$AC^+ = \{A\,C\}$ $BD^+ = B\,D$

✓ $AD^+ = \{A\,D\,B\,C\}$ ✓ $CD^+ = C\,D\,B\,A$

eg. R (ABCDEF)

FDs = {AB → C, C → D, D → E, E → F, F → A}

$B^+ = \{B\}$

$AB^+ = \{ABCDEF\}$ $EB^+ = \{EBFACD\}$

$CB^+ = \{BCDEFA\}$ $FB^+ = \{FBACDE\}$

$(DB)^+ = \{DBEFAC\}$

eg. R (ABCDEF)

FDs = {AB → C, C → D, D → EB, E → F, F → A}

$A^+ = \{A\}$ $D^+ = \{DEBFAC\}$

$B^+ = \{B\}$ $E^+ = \{EFA\}$

$C^+ = \{CDEBFA\}$ $F^+ = \{FA\}$

$AB^+ = \{ABCDEF\}$ $BE^+ = \{BEFACD\}$

$BF^+ = \{BFACDE\}$

$EF^+ = \{EFA\}$

$AE^+ = \{AEF\}$

$AF^+ = \{AF\}$

$AEF^+ = \{AEF\}$

Candidate keys.

{C, D, AB, BE, AE}.

eg. R (ABCDEF)

A → BCDEF

BC → ADEF

DEF → ABC

$A^+ = \{ABCDEF\}$

$B^+ = \{B\}$ $E^+ = \{E\}$

$C^+ = \{C\}$ $F^+ = \{F\}$

$D^+ = \{D\}$

$BC^+ = \{BCADEF\}$.

$DEF^+ = \{DEFABC\}$

Can't form 4 element CK.

therefore no 5 or 6 element CK

(A, BC, DEF) CKs

Q. $R = ABCDE$

$FDs = \{A \to B, B \to C, C \to D, D \to A\}$

$\times E^+ = \{E\}$ 4 Candidate keys.

$\checkmark AE^+ = \{AEBCD\}$

$\checkmark BE^+ = \{BECDA\}$

$\checkmark CE^+ = \{CEDAB\}$

$\checkmark DE^+ = \{DEABC\}$

Q. $R = (ABCDE)$

$A \to BC$

$CD \to E$

$B \to D$

$E \to A$

$\checkmark A^+ = ABCDE$

$B^+ = BD$

$C^+ = C$

$D^+ = D$

$\checkmark E^+ = EABCD$

$\checkmark BC^+ = BCDEA$

$\checkmark CD^+ = CDEAB$

$BD^+ = BD$

~~$BCD^+ = BCDEA$~~

CKs = A, E, BC, CD.

Q. $R(ABCD)$ $\{AB \to CD, D \to A\}$

What are CKs of sub relation $R_1(BCD)$?

→ $B^+ = B$ $C^+ = C$ $D^+ = DA$

$BC^+ = BC$ $CD^+ = CDA$ $\checkmark BD^+ = BDAC$

$CK = \{BD\}$

Q $R(ABCDEF)$

$AB \to C, B \to D, AD \to F, C \to D, D \to E, E \to F, E \to D$

CKs for $R_1(DEF)$?

→ $\checkmark D^+ = DEF$ Only CKs D, E.

$\checkmark E^+ = EFD$

$F = F$

Q R(ABCDE)

$A \to BC,\ CD \to E,\ B \to D,\ E \to A$

CKs of R_1 (ABCE) ?

→ ✓ $A^+ = ABCDE$

$B^+ = BD$

$C^+ = C$

✓ $E^+ = EABCD$

✓ $BC^+ = BCDEA$

CKs = {A, E, BC}

Q R(AB)

Total no. of FDs possible.

→ $\phi \to \phi$ $\phi \to A$ $\phi \to B$ $\phi \to AB$

$A \to B$ $A \to A$ $A \to \phi$ $A \to AB$

$B \to B$ $B \to A$ $B \to \phi$ $B \to AB$

$AB \to \phi$ $AB \to A$ $AB \to B$ $AB \to AB$.

Q R(ABC)

Given FDs $\{A \to B, B \to C\}$ what are additional FDs we can determine from this ?

→ $\phi \to \phi$ 1

$A \to \{ABC\}$ $2^3 = 8$

$B \to \{BC\}$ $2^2 = 4$

$C \to \{C\}$ 2.

$AB \to \{ABC\}$ $2^3 = 8$

$BC \to \{BC\}$ $2^2 = 4$

$CA \to \{ABC\}$ $2^3 = 8$.

$ABC \to \{ABC\}$ $2^3 = 8$

$X \to Y$
$ABC \to ABC$
$2^3 \quad 2^3$
$8 \times 8 = 64$

$(1 + 8 + 4 + 2 + 8 + 4 + 8 + 8) = 43$ additional FDs (that are valid)

Q. In a schema with attributes A, B, C, D & E following set of FDs are given

$A \to B$, $A \to C$, $CD \to E$, $B \to D$, $E \to A$

Which of the following FDs is not implied by the above set? ~~$CD \to AC$~~, ~~$BD \to CD$~~, ~~$BC \to CD$~~, ~~$AC \to BC$~~

→ $CD^+ = CDEAB$

$AC^+ = ACBDE$

$BD^+ = BD$

$BC^+ = BCDEA$

* Equivalence of two sets of Functional Dependencies.

Two different sets of FDs for a given relation may or may not be equivalent. If F & G are the 2 sets of FDs, then following 3 cases are possible—

1. F covers G ($F \supseteq G$)
2. G covers F ($G \supseteq F$)
3. Both F & G cover each other ($F = G$).

1. Determining whether F covers G. :

a) Take FDs of set G into consideration. For each FD $X \to Y$, find the closure of X using the FDs of set G.

b) Take FDs of set G into consideration. For each FD $X \to Y$, find the closure of X using the FDs of set F.

c) If the FDs of set F has determined all those attribs that were determined by the FDs of set G, then it means F covers G.

Thus we conclude $F \supseteq G$, otherwise not.

$G \subseteq F$

2. Determining whether G covers F.

Similar to 1.

3. Determining whether both F & G cover each other.

If $F \supseteq G$ & $G \supseteq F$, we conclude $F = G$.

eg: R (A, C, D, E, H).

Two functional dependency sets F & G.

$F = \{A \to C, AC \to D, E \to AD, E \to H\}$

$G = \{A \to CD, E \to AH\}$.

Which is true?

a) $G \supseteq F$ b) $F \supseteq G$ c) $F = G$ d) all.

→ $F \supseteq G$?

$A^+ = \{A C D\}$ $E^+ = \{E A D H C\}$ using F

So, $F \supseteq G$.

$G \supseteq F$?

$A^+ = \{A C D\}$ using G

$AC^+ = \{A C D\}$

$E^+ = \{E A H C D\}$ So, $G \supseteq F$

$E^+ = \{E A H C D\}$

$\therefore F = G$.

Q F : $\{A \to B, B \to C, C \to D\}$

G : $\{A \to BC, C \to D\}$

→ F ⊇ G ?

$A^+ \Rightarrow (A\ B\ C\ D)$

$C^+ = (C\ D)$

G ⊇ F ?

$A^+ = \{A\ B\ C\ D\}$

$B^+ = (B)$ ×

$C^+ = (C\ D)$

Not equivalent. As F ⊇ G but G ⊉ F.

Q. F : $\{A \to B, AB \to C, D \to AC, D \to E\}$

G : $\{A \to BC, D \to AB\}$.

→ F ⊇ G?

$A^+ = \{A\ B\ C\}$

$D^+ = \{D\ A\ C\ B\ E\}$

F ⊇ G.

G ⊇ F

$A^+ = \{A\ B\ C\}$

$AB^+ = \{A\ B\ C\}$

$D^+ = \{D\ A\ B\ C\}$

$E^+ = \{E\}$ ×.

G ⊉ F.

Q. F : $\{A \to B, B \to C, C \to A\}$

G : $\{A \to BC, B \to A, C \to A\}$.

→ F ⊇ G ? ✓

$A^+ = \{A\ B\ C\}$

$B^+ = \{B\ C\ A\}$

$C^+ = \{C\ A\ B\}$

G ⊇ F ? ✓

$A^+ = \{A\ B\ C\}$

$B^+ = \{B\ A\ C\}$

$C^+ = \{C\ A\ B\}$

F = G.

* Canonical Cover

A canonical cover is a simplified & reduced version of the given set of functional dependencies. Since it is a reduced version, it's also called irreducible set.

Canonical cover is free from all the extraneous FDs (an attrib. of a FD is extraneous if we can remove it without changing the closure of the set of FDs). The closure of canonical cover is same as that of the given set of FDs. Canonical cover is not unique & may be more than one for a given set of FDs.

Need for CC.

i) Working with the set containing extraneous FDs increases the computation time.

ii) Therefore, the given set is reduced by eliminating the useless FDs.

iii) This reduces the computation time & working with the irreducible set becomes easier.

Steps to find CC

1. Writing set of FDs in such a way that each FD contains exactly one attrib. on its right side.

eg. $X \rightarrow YZ$ to

$X \rightarrow Y$

$X \rightarrow Z$.

2. Determining each FD to be essential or non-essential.

To come to know whether an FD is essential or not, compute the closure of its left side —

- Once by considering that the particular FD is present in the set.
- Once by considering that the particular FD is not present in the set.

Then following 2 cases are possible —

Case 1 If results come out to be same, it suggests that the presence or absence of that FD does not create any difference. Thus it is non-essential. Eliminate FD from set. Eliminate non-essential FDs as soon as it is discovered. *

Case 2 If results come out to be different, it means that the FDs are essential.

3. Check if there is any FD that contains more than one attribute on its left side.

Case 1 No There contains no FD with 2 or more attrib.s on left side. Previous step result is the cc.

Case 2 Yes Consider all FDs one by one. Check if left side can be reduced. (Checking — compute closure of all the possible subsets of the left side of that FD. If any of the subsets produce the same closure result as produced by the entire left side, then replace the left side with that subset).

Set obtained now is cc.

* Do not consider it while checking the essentiality of other FDs.

eg. $R(W, X, Y, Z)$

$FDs = \{ X \to W,\ WZ \to XY,\ Y \to WXZ \}$

→ Step 1

$X \to W$ ✓, $WZ \to X$ ✗, $WZ \to Y$ ✓, $Y \to W$ ✗, $Y \to X$ ✓, $Y \to Z$ ✓.

Step 2

$X^+ = \{X\}$ without considering $X \to W$

So, $X \to W$ is essential. [as we're not getting W]

$WZ^+ = \{WZYX\}$ without considering $WZ \to X$.

So, $WZ \to X$ is non essential. [as we're getting X not considering the FD $WZ \to X$]

$WZ^+ = \{WZ\}$ essential. $WZ \to Y$

$Y^+ = \{YWXZ\}$ non essential. $Y \to W$

$Y^+ = \{YZ\}$ essential $Y \to X$

$Y^+ = \{YXW\}$ essential $Y \to Z$

We get after step 2,

$X \to W,\ WZ \to Y,\ Y \to X,\ Y \to Z$

Step 3

$WZ \to Y$.

None of subsets of WZ produced Y in their closure.

$W^+ = \{W\}$

$Z^+ = \{Z\}$

So, $WZ \to Y$ essential.

So, CC. ~ $X \to W$,
$WZ \to Y$
$Y \to X$
$Y \to Z$

Q $\{AB \to C, D \to E, E \to C\}$ (F) is the minimal cover of $\{AB \to C, D \to E, AB \to E, E \to C\}$ (G) — T/F

→ 1 way here best way

FD equivalence.

$F \supseteq G$. ? $AB^+ = ABC$ not equivalent

So, can't be minimal cover.

$AB \to C, D \to E, AB \to E, E \to C$

$AB^+ = \{ABEC\}$ $AB \to C$ redundant

$D^+ = \{D\}$ $D \to E$ essential

$AB^+ = \{AB\}$ $AB \to E$ essential

$E^+ = \{E\}$ $E \to C$ essential.

$D \to E, AB \to E, E \to C$..

↳ Minimal cover.

$A^+ = \{A\}$

$B^+ = \{B\}$.

* Decomposition of a Relation.

The process of breaking up or dividing a single relation into two or more subrelations.

- Properties

i) Lossless : No information is lost from the original relation during decomposition When the subrelations are joined back, the same relation is obtained that was decomposed. Every decomposition must always be lossless.

ii) Dependency preservation : None of the FDs that holds on the original relation are lost.

- Types of Decomposition.

(1) Relation R decomposed into subrelations $R_1, R_2, \ldots, R_n$. It is lossless join decomposition when the join of the subrelations results in the same relation R that was decomposed.

For lossless join decomposition,

$$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n = R.$$

$\bowtie$ – natural join operator

#. Lossless join decomposition is also known as non-additive join decomposition. This is because the resultant relation after joining the subrelations is same as the decomposed relation. No extraneous tuples appear after joining of the sub-relations.

eg. R (A, B, C)

A	B	C
1	2	1
2	5	3
3	3	3

Decomposed into $R_1(A,B)$ & $R_2(B,C)$

A	B
1	2
2	5
3	3

B	C
2	1
5	3
3	3

natural join

A	B	C
1	2	1
2	5	3
3	3	3

Same as original relation R

(2) Relation R decomposed into $R_1, R_2, \ldots, R_n$.

This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed. The natural join of the sub relations is always found to have some extraneous tuples.

$$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n \supset R$$

Lossy join decomposition is known as careless decomposition. This is because extraneous tuples get introduced on the natural join of the sub relations.

eg. R (A, B, C)

A	B	C
1	2	1
2	5	3
3	3	3

R_1 (A, C)

A	C
1	1
2	3
3	3

R_2 (B, C)

B	C
2	1
5	3
3	3

$\bowtie$

A	B	C	
1	2	1	
2	5	3	
2	3	3	← Extraneous
3	5	3	←
3	3	3	

$R_1 \bowtie R_2 \supset R$

- Determining whether decomposition is lossless or lossy.

If following conditions satisfy, the decomposition is lossless. If any fails, then lossy.

1. Union of both the subrelations must contain all the attributes that are present in the original relation R.

$$R_1 \cup R_2 = R$$

2. Intersection of both the subrelations must not be null. There must be some common attribute which is present in both the subrelations.

$$R_1 \cap R_2 \neq \phi$$

3. Intersection of both the subrelations must be a superkey of either R_1 or R_2 or both.

$$R_1 \cap R_2 = \text{Super Key of } R_1 \text{ or } R_2 \text{ or both}$$

Q. Consider a rel^n schema R(A, B, C, D) with the FDs $A \rightarrow B$ & $C \rightarrow D$. Determine whether the decomposition of R into $R_1(A, B)$ & $R_2(C, D)$ is lossless or lossy.

→ $R_1(A, B) \cup R_2(C, D)$

$= R(A, B, C, D)$

$\rightsquigarrow R_1(A,B) \cap R_2(C,D)$

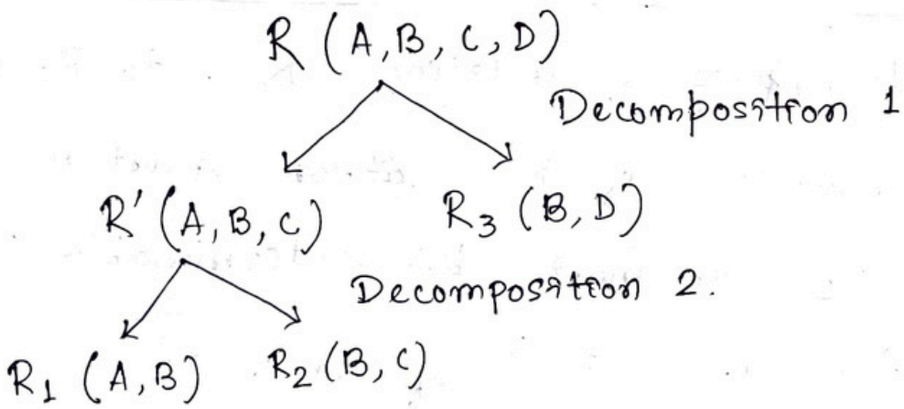
$= \phi$

Decomposition is lossy.

⑤ $R(A,B,C,D)$ with FDs $\{A \to B, B \to C, C \to D, D \to B\}$. Determine whether the decomposition of R into $R_1(A,B)$, $R_2(B,C)$, $R_3(B,D)$ is lossless or lossy.

→ # When a given relation is decomposed into more than two subrelations, then consider any possible way in which the relⁿ might have been decomposed into those subrelations. First, divide the given relⁿ into two subrelations. Then, divide the subrelations acc. to the subrelations given in problem.

As a thumb rule, any relⁿ can be decomposed only into two subrelations at a time.



Decomposition 1..

$\rightsquigarrow R'(A,B,C) \cup R_3(B,D)$

$= R(A,B,C,D)$

$\rightsquigarrow R'(A,B,C) \cap R_3(B,D)$

$= B$ Now, $B^+ = \{B, C, D\}$

$\rightsquigarrow R'(A,B,C) \cap R_2(B,D)$

$= B \neq \phi$

Attribute B can't determine attribute A of subrel^{n} R'. So, B is not super key of R'.

B can determine all attributes of R_3. Thus, it is a super key of sub relation R_3.

So, as one of the subrelations has the intersection as the super key, it is lossless.

Decomposition 2.

$\rightsquigarrow R_1(A,B) \cup R_2(B,C) = R'(A,B,C)$

$\rightsquigarrow R_1(A,B) \cap R_2(B,C) = B \neq \phi.$

$B^+ = \{B, C, D\}$

B is a super key of R_2.

So, decomposition 2 is lossless.

∴ Overall decomposition is lossless.

- Dependency preserving Decomposition.

If we decompose a relation R into R_1 & R_2, all dependencies of R either must be a part of R_1 or R_2 or must be determinable from combination of FD's of R_1 & R_2.

Q. $R(A,B,C,D)$ & FDs $\{A \rightarrow B, C \rightarrow D\}$.

Decomposition of R into $R_1(A,B)$ & $R_2(C,D)$ is —

$\rightsquigarrow$ $R_1(A,B) \cup R_2(C,D)$

$= R(A,B,C,D)$

$R_1(A,B) \cap R_2(C,D) = \phi$ lossy decomposition.

$R_1(A,B)$ $\quad A \to B,\ C \to D$

Possible FDs

- $A \to B$
- $B \to A$
- $C \to D$
- $D \to C$.

So, answer is dependency preserving but not lossless.

Q. $R(A,B,C,D)$

FDs $\{AB \to CD,\ D \to A\}$

Decomposition example 3. rbr.

(?) Decomposed into $R_1(A,D)$, $R_2(B,C,D)$.

$R_2(B,C,D)$ As A not present in $R_2(B,C,D)$

$\rightarrow$ $R_1(A,D)$

$A \to D$

$D \to A$

$A^+ = \{A\}$

$D^+ = \{D,A\}$

$B^+ = \{B\}$

$C^+ = \{C\}$

$D^+ = \{D,A\}$

$BC^+ = \{B,C\}$

$CD^+ = \{C,D,A\}$

$BD^+ = \{B,D,A,C\}$

$B \to C$

$C \to D$

$D \to B$

$BC \to CD$

$CD \to BD$

$BD \to BC$.

$BD \to C$

Not dependency preserving.

(as $AB \to CD$ not preserved)

Q. $R(A,B,C,D)$ & FDs $\{AB \to C,\ C \to D,\ D \to A\}$.

Decomposed into $R_1(A,B,C)$ & $R_2(C,D)$. Check dependency preservation.

$\rightarrow$ F_1 be the set of FDs for R_1 & that for R_2 is F_2.

For F_1, all combination of A, B, C.

$A^+ = \{A\}$ trivial $\quad B^+ = \{B\}$ trivial $\quad C^+ = \{C,D,A\} = \{C,A\}$ as D not present in R_1

$\therefore$ $C \to A$

$(AB)^+ = \{A, B, C, D\} = \{A, B, C\}$ remove D, as not in R_1

$AB \rightarrow C$

$(BC)^+ = \{B, C, D, A\} = \{A, B, C\}$

$BC \rightarrow A$

$(AC)^+ = \{A, C, D\} = \{A, C\}$ $AC \rightarrow AC$ trivial

$\therefore F_1 = \{C \rightarrow A, AB \rightarrow C, BC \rightarrow A\}$.

For F_2,

$C^+ = \{C, D, A\} = \{C, D\}$ $C \rightarrow D$

$D^+ = \{D, A\} = \{D\}$

$\therefore F_2 = \{C \rightarrow D\}$

Original FDs $\{AB \rightarrow C, C \rightarrow D, D \rightarrow A\}$

$D \rightarrow A$ not preserved.

∴ Decomposition is not dependency preserving.

→ Working formula - Checking FD preservation

R be decomposed to R_1, R_2. with FD set F.
R_1 & R_2 have FD set F_1, F_2.

If $R_1 = \{A, B, C\}$, $R_2 = \{C, D\}$ then to get F_1 & F_2 take every combination of attributes from R_1 & R_2 separately & compute closure of them using F.

i.e. A^+ AB^+ BC^+ CA^+ for F_1 | C^+ D^+ for F_2.

(not ABC, as it will be trivial) (not CD as it will be trivial)

While computing closures for different relations if the closure contains some attribute that is not present in the relation itself, then reject it. (i.e. if $AB^+ = \{A, B, C, D\}$ then we reject D as it is not an attribute of R_1). $\Rightarrow AB^+ = \{A, B, C\}$.

We should not consider trivial FDs.

After doing this, we need to check whether $F_1 \cup F_2 = F$ or not.

If $F_1 \cup F_2 = F$ then it's FD preserving.

$F_1 \cup F_2 \subset F$ not preserving.

$F_1 \cup F_2 \supset F$ not possible.

Q $R(A, B, C, D, E, [illegible], G)$.

FDs : $\{AB \to C,\ AC \to B,\ AD \to E,\ B \to D,\ BC \to A,\ E \to G\}$

Decomposed into $R_1(ABC)$, $R_2(ABDE)$, $R_3(EG)$.

→ $R(A, B, C, D, E, [illegible], G)$

Decom 1: $R'(ABCDE)$ and $R_3(EG)$

Decom 2: $R_1(ABC)$ and $R_2(ABDE)$

Decom 1

~~$R_1(A,B,C) \cup R_2(A,B,D,E) \cup R_3(E,G)$~~

~~$= R''(A,B,C,D,E,G)$~~

~~Not considered.~~

$R_3(EG) \cup R'(ABCDE)$

$= R(A, B, C, D, E, G)$.