Chapter 5 :- Database constraint and Relational Database Design.

Functional Dependencies.

- They are the fundamental to the process of Normalization.
- Here, functional dependency plays a key role in differentiating good database design from bad database design.
- It describes the relationship between attributes in a table.
- For attributes X and Y in Relation R, functional dependency between X and Y is shown as :-

$$X \rightarrow Y$$

Here, X is determinant
Y is the functional dependent on X.

eg:-

eid	ename	age	salary
101	Ram	25	15000
102	Shyam	28	20000
103	Sita	26	20000

Here, eid → ename ; eid uniquely determines ename.

ename → eid ; is not always true.

eid → age ; True
eid → salary ; True

age → salary ; False

age → eid ; False

employee (eid, project NO, Hours, ename, pname, Plocation)

Here, eid → ename
project NO → {pname, Plocation}

{eid. project NO} → Hours.

Types of functional dependencies

i) Full functional dependency

- X → Y is a full functional dependency if the removal of any attribute A from X removes the dependency.
- A full functional dependency occours when it is already functional dependency and the set of attributes on the left side can't be reduced any further.
- example.

orderNO	LineNO	qty	price
A001	L001	10	200
A002	L001	20	400
A002	L002	30	800
A004	L001	15	300

Here, {orderNO, LineNO} → qty

{orderNO, LineNO} → price are the full functional dependencies.

ii) partial functional dependency.

- A functional dependency X → Y is partial dependency if some attributes A ∈ X can be removed from X and the dependency still holds.

eg:-
{eid, phone} → name

Here,
remove Phone, then {eid} → name still holds true, so it is partial functional dependency.

iii) Transitive Dependency

- It occours where there is an indirect relationship that causes a functional dependency.
- If X → Y and Y → Z, then X → Z is a transitive dependency.

iv) Trivial dependency

- It occurs when the functional dependency of an attribute is described on collection of attributes that includes the original attribute.
- Functional depencency is trivial if RHS is a subset of LHS

eg:-
{name, SSN} → SSN

v) Non-trivial dependency

- It is the one that is not trivial

eg:-
{S#, P#} → {S#, qty}

Axioms or Rules of inference for Functional Dependencies.

- It provides a simple technique for reasoning about functional dependies.

a) Reflexivity Rule

- If Y is a subset of X then X → Y holds.

b) Augmentation Rule

- If X → Y holds and Z is a set of attribute then ZX → ZY holds.

c) Transitivity Rule

If X → Y and Y → Z holds then X → Z holds

d) union Rule

- If X → Y holds and X → Z holds then X → YZ holds

e) Decomposition Rule

- If X → YZ holds then X → Y and X → Z holds.

f) pseudo Transitivity Rule

- If X → Y holds and ZY → P holds then XZ → P holds.

g) self determinaton

A → A

eg:- R = (A, B, C, G, H, I)

set of functional dependencies F{A → B, A → C, CG → H, CG → I, B → H}

Then, we have.

i) A → B, A → C then A → BC {union Rule}

ii) A → B, B → H then, A → H {Transitivity Rule}

iii) CG → H, CG → I then, CG → HI {union Rule}

iv) AG → I then, {pseudo Transitivity Rule}

eg:- If $F = \{A \to B, C \to X, BX \to Z\}$

Prove or disprove $AC \to Z$

Soln

$C \to X, BX \to Z \Rightarrow BC \to Z$: {using, pseudo-transitivity Rule}

$A \to B, CB \to Z \Rightarrow AC \to Z$

Closure of a set of functional dependencies.

- The set of functional dependencies that is logically implied by F is called closure of F and is written as F^+.
- The closure of F denoted by F^+ is the set of all Functional dependencies entailed by F.
- $F^+ = \{x \to y \mid F \models X \to Y\}$ (pic)
- If F is a set of Functional dependencies, we can compute F^+ directly from formal defination of functional dependency.
- If F were large, this process be lengthy and difficult so, Armstrong axioms useful for fast and accurate result.

eg:- $R\{A, B, C, D\}$

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

$F^+ = ?$

Soln

$A \to B, A \to C$ then $A \to BC$ {union Rule}

$A \to B, BC \to D$ then, $AC \to D$ {pseudo Transitive Rule}

$A \to C, BC \to D$ then, $AB \to D$ {pseudo Transitive Rule}

eg:- $R = \{A, B, C, D, E, F\}$

$F = \{A \to BC, B \to E, CD \to EF\}$

List F^+ and prove $AD \to F$

Soln

$A \to BC$ then $A \to B, A \to C$ {Decomposition Rule}

$CD \to EF$ then, $CD \to E, CD \to F$ {Decomposition Rule}

$AD \to E, AD \to F$ .proved //

Closure of attribute set

- Closure of attribute set is the set of attributes which are functionally dependent on the attribute set.
- Given a set of attributes $A_1 \ldots A_n$ the closure of $\{A_1 \ldots A_n\}^+$ is the set of attributes B such that $A_1 \ldots A_n \to B$

eg:-

name → color

category → department

color, category → price

Now, $\{name\}^+ = \{name, color\}$

$\{color\}^+ = \{color\}$

$\{name, category\}^+ = \{name, category, color, department, price\}$

An Algorithm to compute α^+, the closure of α under F.

result = α

while (changes to result) do

for each functional dependency $\beta \to \gamma$ in F do

begin

if $\beta \subseteq$ result then

result = result $\cup$ γ

end

eg:- $R = \{A, B, C, D, E, H, I\}$

$F = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

Compute $(AG)^+$

soln Let,

result = $\{AG\}$

1st iteration

For $A \to B$

$A \subseteq \{AG\}$; True so,

$\therefore$ result = $\{ABG\}$

For $A \to C$

$A \subseteq \{ABG\}$; True so,

$\therefore$ result = $\{ABCG\}$

For $CG \to H$

$CG \subseteq \{ABCG\}$; True

$\therefore$ result = $\{ABCGH\}$

For $CG \to I$

$CG \subseteq \{ABCGH\}$; True

$\therefore$ result = $\{ABCGHI\}$

For $B \to H$

$B \subseteq \{ABCGHI\}$; True so,

$\therefore$ result = $\{ABCGHI\}$

2nd iterations :-

we get the final result

= $\{ABCGHI\}$

The result doesn't change further

$\therefore (AG)^+ = \{ABCGHI\}$

eg:- $R = \{A, B, C, D, E, F\}$

$F = \{A \to BC, E \to CF, B \to E, CD \to EF\}$

Compute $(AB)^+$

Soln

1st iteration

Let, result = AB then,

For $A \to BC$

$A \subseteq \{AB\}$; true so,

$\therefore$ result = $\{ABC\}$

For $E \to CF$

$E \subseteq \{ABC\}$; False so,

$\therefore$ result = $\{ABC\}$

For $B \to E$

$B \subseteq \{A, B, C\}$; True so,

$\therefore$ result = $\{ABCE\}$

For $CD \to EF$

$CD \subseteq \{ABCE\}$; False so,

$\therefore$ result = $\{ABCE\}$

2nd iteration.

For $A \to BC$

$A \subseteq \{ABCE\}$ True so,

$\therefore$ result = $\{ABCE\}$

For $E \to CF$

$E \subseteq \{ABCE\}$ True

$\therefore$ result = $\{ABCEF\}$

For $B \to E$

$B \subseteq \{ABCEF\}$ True

$\therefore$ result = $\{ABCEF\}$

For $CD \to EF$

$CD \subseteq \{ABCEF\}$ False

$\therefore$ result = $\{ABCEF\}$

3rd iteration

Then we get final result

$= \{ABCEF\}$ The result

doesn't change further $(AB)^+ = \{ABCEF\}$.

Extraneous Attribute :-

- An attribute of functional dependency is extraneous if we can remove it without changing the closure of the set of functional dependencies.

Formal defination.
consider F as set of functional dependency and Function dependency $\alpha \to \beta$ in F. Then.

- attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - (\alpha \to \beta)) \cup ((\alpha - A) \to \beta)$
- attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - (\alpha \to \beta)) \cup (\alpha \to (\beta - A))$ logically implies F.

a) To test if attribute A is extraneous in α,

i) compute $(\alpha - A)^+$ using dependencies in F,
ii) check that $(\alpha - A)^+$ contains A; if it does, A is extraneous.

b) To test if attribute A is extraneous in β.

i) compute α^+ using only dependencies in F'.
Here, $F' = (F - (\alpha \to \beta)) \cup ((\alpha \to (\beta - A))$

ii) check that $(\alpha)^+$ contains A, if it does, A is

extraneous.

eg :- $F = \{A \to B, B \to C, AC \to D\}$.

for $AC \to D$

$\alpha = AC$

Let, take an attribute A then,

$= (AC - A)^+$

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

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

∴ A is not extraneous.

$(AC - C)^+$

$= A^+ = \{ABCD\}$

$C \in A^+ = \{ABCD\}$

∴ C is extraneous.

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

eg :- $F = \{A \to B, B \to C, A \to CD\}$

Soln

$A \to CD$

Let, $\beta = CD$ $\alpha = A$

Let, take an attribute C than,

$\alpha^+ = A^+$

$F' = \left(\{ A \to B, B \to C, A \to CD \} - (A \to CD) \right) \cup (A \to (CD - D))$

$F' = (\{ A \to B, B \to C, A \to C \})$

$\therefore A^+ = \{ABC\}$

$\therefore C \in A^+ = \{ABC\}$

$\therefore$ 'C' is extraneous.

For attribute 'd' we get,

$D \notin A^+ = \{ABC\}$

$\therefore$ D is not extraneous.

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

Determining candidate Key.

1) Compute closure of each attributes.
2) Any attributes is called candidate key of any relation if attribute closure is equal to the relation.

- Attribute that are part of candidate key are called prime attribute.
- attribute that are not part of candidate key are called non-prime attribute.

eg:- $R = (A, B, C, D)$

$F = \{ AB \to C, C \to D, D \to A \}$

List all candidate keys, prime and non-prime attributes.

soln

For A

$A^+ = \{ A \}$; A is not candidate key.

For B

$B^+ = \{ B \}$; The result doesn't match so, it is not candidate key.

For C

$C^+ = \{ CDA \}$; C is not candidate key.

For D

$D^+ = \{DA\}$; 'D' is not candidate key.

For AB

$(AB)^+ = \{ ABCD \}$; AB is candidate key.

For AC

$(AC)^+ = \{ACD\}$; AC is not candidate key.

For (AD)

$(AD)^+ = \{AD\}$; not a candidate key.

For BC

$(BC)^+ = \{BCDA\}$; BC is not (yes) candidate key.

For BD

$(BD)^+ = \{BDAC\}$; BD is a candidate key.

For CD

$(CD)^+ = \{CDA\}$; CD is not a candidate key.

∴ candidate key = {AB, BC, BD}

prime attribute = {A, B, C, D}.

Decomposition

- It refers to the breaking down of one table into multiple tables.

→ Desirable properties of decomposition

a) Attribute prevention
b) Dependency prevention
c) Lack of redundancy
d) Lossy Decomposition
e) Non-loss or lossless Decomposition.

a) Attribute prevention

- preserving all the attributes of relation being decomposed.

b) Dependency prevention.

- Let F be the dependencies on a relation R which is decomposed in $R_1, R_2 \ldots R_n$.

- We can partition the dependencies given by F such that $F_1, F_2, \ldots F_n$ are dependencies that only involve attributes from relation $R_1, R_2 \ldots R_n$ respectively.

- If the union of dependencies imply all the dependencies in F, then we say decomposition has preserved dependencies otherwise not.

- If the decomposition doesnot preserve the dependencies F, then the decomposed relation may contain relation that do not satisfy F.

c) Lack of Redundancy

- Redundancy should be avoided as much as possible.

d) lossy decomposition

- Loss of information due to decomposition is called lossy decomposition.

e) Non-loss decomposition

- It refers to decomposition where all information is preserved

eg:-

ABC

A	B	C
a1	100	c_1
a2	200	c_2
a3	300	c_3
a4	200	c_4

⇒

AB

A	B
a_1	100
a_2	200
a_3	300
a_4	200

BC

B	C
100	c_1
200	c_2
300	c_3
200	c_4

⇓

AUBUC

A	B	C
a_1	100	c_1
a_2	200	c_2
a_2	200	c_4
a_3	300	c_3
a_4	200	c_2
a_4	200	c_4

Normalization

- Database normalization is a technique of organizing data in the database.
- It is the process of removing redundancy and undesirable characteristics like insertion, update and deletion anamolies.
- Mainly two purposes of normalization are:-
  1) Eliminating redundant data.
  2) Ensuring data dependencies make sense.
- It involves dividing table into two or more tables and defining relationship between the tables.

Normal forms

- It represents the guidelines for record design.
- E.F codd originally established three normal forms. 1NF, 2NF, 3NF.
- others FORMS are BCNF (Boyce codd Normal Form) 4NF, 5NF, DKNF (DoMain Key Normalization)

unNormalized Form

- It contains one or more repeating groups or each row may contain multiple set of values for some columns.
- Multiple values in a single row are also called non-atomic values.

eg:-

cid	Name	Address	phone
101	Ram	Ktm	9823048543
			9861977612
102	Sita	Patan	982323404
103	Hari	Dharan	9803540511
			9863944091

Fig:- unNormalized Form (UNF)

First Normal Form (1NF)

- A relation is in 1NF if and only if all columns are atomic i.e. no repeating values in columns.
- Now, Fig 1 will become.

Cid	Name	address	phone
101	Ram	K+m	9823048543
101	Ram	K+m	9861977612
102	Sita	patan	982323904
103	Hari	Dharan	9803540511
103	Hari	Dharan	9863444041

- In 1NF there should be no repeating columns.

eg:-

id Name Phone Child1 Chaild2 Child3

↓

id Name phone child.

- values of each attribute is atomic.
- No composite values.
- All entries in any columns must be of same kind.
- each column must have unique name.
- No two rows are identical.

Second Normal Form (2NF)

- A relation is in 2NF if,
  - it is in 1NF
  - all attributes depend on full primary key

eg:-

personID	projectID	Name	projectName	phone
1	1	Ram	Database	9863444041
2	1	Sita	Database	9803540511
1	2	Ram	web	9823622245
2	2	Sita	web	9808580414

Fig:-2

In Fig:-2 Given relation is in 1NF but not in 2NF because all attributes are not fully dependent on primary key (personID, projectID).

Now decompose above table as follows.

personID	Name	Phone
1	Ram	9863444041
2	Sita	9803540511

table 1

projectID	projectName
1	Database
2	web

table2

id	personID	projectID
1	1	1
2	2	1
3	1	2
4	2	2

table 3

Third Normal Form (3NF)

- A relation is in 3NF if
  - it is is in 2NF.
  - There is no transitive functional dependency i.e. There should not be case that non prime attribute is determined by another non prime attribute.

eg:-

St_id	St_name	DoB	Zip	city
1	Ram	2055/5/5	01	Kathmandu
2	Sita	2055/2/1	01	Kathmandu
3	Gita	2040/3/1	02	patan
4	Hari	2045/06/06	03	Dharan

Fig:- 3

In above fig:-3

St_id is a primary key. all other attributes are dependent on St_id, so it is in 2NF but, city, non-prime attribute is dependent on Zip, another non prime attribute. so it is not in 3NF.

Now, decompose above table as below:-

St_id	St_name	DoB	Zip
1	Ram	2055/5/5	01
2	Sita	2055/2/1	01
3	Gita	2040/3/1	02
4	Hari	2045/06/06	03

table 1

Zip	city
01	Kathmandu
02	patan
03	Dharan

table 2

BCNF

- A relation is in BCNF if,
  - it is in 3NF
  - every determinant in that table is candidate key.
- Advance Form of 3NF also referred as 3.5NF.
- If table contain only one candidate key, 3NF and BCNF are equivalent.

eg:-

Student	Course	Teacher
Ram	DB	Shyam
Ram	e.com	Sita
Gita	DB	Shyam
Gita	e.com	Sita

Fig:- 4

Here
Key = {student, course}
F.D i) {student, course} → Teacher
Teacher → course.

in ii) Teacher is not a candidate key but determines course, above figure 4 is Not in BCNF.

Now, Decompose it as follows

Teacher	course
Shyam	DB
Sita	e.com

table 1

Student	Course
Ram	DB
Ram	e.com
Gita	DB
Gita	e.com

table 2

Multivalued Dependency

- In a Relation R(A,B,C) if each A value has associated with it a set of B values and a set of C values such that B and C values are independent of each other, then the relation is said to have Multivalued dependency.

$$A \twoheadrightarrow B,\ A \twoheadrightarrow C$$

eg:-

Model	Year	color
M1	2007	Red
M1	2008	Blue
M2	2012	Red
M2	2016	Blue
M3	2018	Red
M3	2019	Blue

Here Model $\twoheadrightarrow$ year and Model $\twoheadrightarrow$ color.

#4NF

* A relation R is in 4NF if
  - it is is in 3NF or BCNF
  - R contains no Multivalued attributes.

eg:-

ename	pname	dname
Ram	X	John
Ram	Y	sita
Ram	X	Sita
Ram	Y	John

Here ename $\twoheadrightarrow$ pname ename $\twoheadrightarrow$ dname.

∴ to convert it into 4NF, decompose into

ename	pname
Ram	x
Ram	y

table 1

ename	dname
Ram	John
Ram	sita

table 2

DeNormalization.

- It is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data.
- In relational database, denormalization is an approach to speed up read performance in which the administrator selectively adds back specific instances of redundant data after the data structure has been normalized.
- It is needed when multiple joins in some query can have negative impact on performance.
- denormalization should take place after a satisfactory level of normalization has taken and that any required constraints and/or rules have been created to deal with the inherent anamolies in design.

eg:- Database Normalization Example.

employee(Name, project, task, office, floor, phone)

Name	project	task	office	floor	phone
Ram	xyz	T1	400	3	1400
Ram	ABC	T1	400	3	1400
Ram	ABC	T2	400	3	1442
Sita	xyz	T3	442	3	1442
Sita	ABC	T3	442	3	1442
Sita	pqR	T3	442	3	1558
Hari	xyz	T2	558	4	

Table 1

- Is Table1 in unnormalized form? ⇒ No
- Is it in 1NF? ⇒ yes

Ans:-

- Is it in 2NF? No, because with name only we can figure out office, floor and phone. i.e. all prime attributes are not dependent on fully primary key.

∴ Split it into two relations

a) Employee-project-task (name, project, task)

b) Employee-office-phone (name, office, floor, phone)

So, Tables are:-

Name	project	task
Ram	xyz	T1
Ram	ABC	T1
Ram	ABC	T2
Sita	xyz	T3
Sita	ABC	T3
Sita	pqR	T3
Hari	xyz	T2

Table a

Name	office	floor	phone
Ram	400	3	1400
Sita	442	3	1442
Hari	558	4	1558

Table b

Is table a in 3NF? yes

Is table b in 3NF? No, office can determine phone. i.e. non prime attribute can determine another non prime attribute.

∴ Split it in two tables.

c) Employee-office (name, office floor)

d) Employee-phone (office, phone)

Name	office	floor
Ram	400	3
Sita	442	3
Hari	558	4

Table c

office	phone
400	1400
442	1442
558	1558

Table d.

- Are all tables table (b), (c), (d) in BCNF? Yes.

- Are all tables in 4NF?
  only table (c) and (d) are in 4NF
  Table a, b are not in 4NF because of Multivalued Attribute.

∴ Split into two Relations

e) employee_project (name, project)
f) employee_task (name, task)

Name	project
Ram	xyz
sita	xyz
Hari	xyz
Ram	ABC
sita	par
Sita	ABC

Table e)

Name	task
Ram	T1
Ram	T2
Sita	T3
Hari	T2

Table f)

Refrential integrity :-
- It refers to the accuracy and consistency of data within a relationship.

Chapter-6 Security

- unauthorized access or manipulation of database creates problems for organization.

- Security refers to protection of data against unauthorized disclosure, alteration or destruction.

- Main objectives of designing secure db system
  a) secrecy b) Integrity c) Availability

- Database security is about controlling access to information i.e. some information be avilable freely and other information be available to certain authorized people or groups.

- Database security system stores authorization rules and enforces them for database access.

- Database security can be :-
  a) physical security

  - It refers to security of hardware and protection of site where computer resides.

  b) logical security

  - It refers to software safeguards for organization system including user identification, password access, access rights, authority levels.

Database Security levels

- To ensure db security, security at different levels should be Maintained.
- A weakness at low level of security allows strict high level security Measures.

i) Database system
- db system have to ensure that authorization restrictions are not voilated.

ii) operating system
- weakness in os security is concerned with unauthorized access to db.

iii) Network
- software level security within network system is important

iv) physical
- sites with computer system Must be physically secured against armed entry by intruders.

v) Human
- user Must be authorized carfully.

Authorization

- It is a security Mechanism used to determine user/client privilege or access level related to system resources.
- During authorization, system verifies authenticated user's access rules and either grant or refuse resource access.

⇒ Authorization includes

- permitting only certian users to access, process or alter data.
- Applying varying limitations on users access or actions
- Limitations placed on users can apply to object such as schema, tables, rows etc.

⇒ Authorization in data includes

- authorization to read data
- authorization to insert new data
- authorization to update data
- authorization to delete data

- Each of these type of authorization is called privilege.
- users an authorized all or none or combination of these types of privileges or specified parts of db such as relation or view.
- In addition to authorization on data, users are granted authorization on db schema, allowing them to create, Modify

or drop relations.

- The ultimate form of authority is that given to DBA
- DBA may authorize new users, restructive the database etc.

Granting and Revoking privileges

- SQL standard includes the privileges select, insert, update, delete.
- 'All privileges' can be used for all allowable privileges.
- A user who creates a new relation is given all privileges automatically.

Grant statement

```
grant <privilege list>
on <relation or view>
to <user/role list>
```

- To read tuples in relation, 'select' authorization is required

eg:-
```
grant select
on employee
to Ram, Hari;
```

- To update any tuple in relation, 'update' authorization is required.
- 'update' authorization may be given either on all attributes of relations or only some.

eg:-
```
grant update
on employee
to sita, Gita
```

```
grant update (salary)
on employee
to sita, Rita
```


- To insert tuples into the relation, insert authorization is used.
- 'insert' privilege may also specify a list of attributes.
- The system either gives default values or Null for remaining attributes.

eg:-
```
grant insert
on employee
to Ram, sita
```

```
grant insert (new address)
on employee
to Ram, Gita
```

- To delete tuples from relation, 'delete' authorization is used.

eg:-
```
grand delete
on employee
to Ram, Hari
```

- privileges granted to 'public' are implicitly granted to all (current and future users.
- cryptosystem:
- Aim to solve problem by modifying data being transmitted in a manner that it become unintelligible to anyone but not for intended recipient.
- Data Encryption
- It is the strong and transmitting data is in encrypted form.
- original data is called plain text.

- plaintext is encrypted using encryption algorithm; whose inputs are plain text and encryption key.
- output is called cipher text.
- Encryption refers to the process of transforming data into a form that is unreadable unless the reverse process of decryption is applied.
- Encryption algorithm use an encryption key to perform encryption and requires a decryption key to perform decryption.
- Encryption is widely used today for protecting data in transit in a variety of applications such as data transfer on internet, cellular phone networks.
- Encryption is also used to carry out other tasks like authentication.
- In database encryption is used to store data in secure way so that if the data is acquired by unauthorized users, the data will not be accessible without a decryption key.
- A good encryption technique has following technique.

→ It is relatively suitable and simple for authorized users to encrypt and decrypt data.

→ It depends upon encryption key used to encrypt data.

- In symmetric key encryption, encryption key is also used to decrypt.
- In asymmetric key encryption, two different keys public and private key are used to encrypt and decrypt data.
- Its decryption key is extremely difficult for an intruder to determine, even if intruder has encrypted data.
- In asymmetric key encryption, its difficult to infer private key even if public key is avilable.

symmetric Key Encryption

- Authorized users must be provided with encryption key via secure mechanisms.
- eg:- AES (Advance Encryption standard), DES (Data Encryption standard)

Asymmetric Key Encryption

- Two keys private and public.
- public key are published.
- private key is known to only to user to whom key belongs.
- If user1 wants to store encrypted data, user1 encrypts them using public key E1, decryption requires private key D1.
- eg RSA

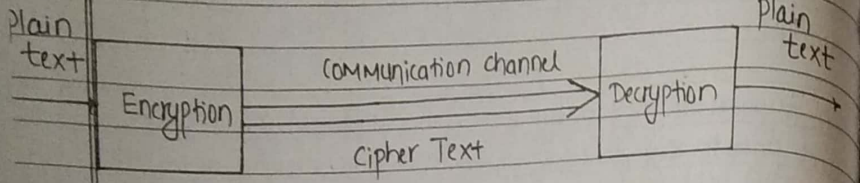


Fig:- Data Encryption and Decryption process.

Chapter-8
Insertion into B+ Tree.

- While inserting values into node, if node is full then follow the following two rules:

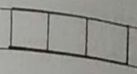
- If node is leaf then break down the node into two partitions.
- The first partition should hold ceil value of (N-1)/2 Key values. N is the number of pointers.
- Second partition can hold rest of the Key values.
- Then copy the smallest key element from second partition to parent node.
- If node is non-leaf then break down the node into two partitions.
- The first partition should hold ceil of (N/2)-1 Key values.
- second partition can hold rest of the values.
- Then Move smallest element from second partition to parent node.

Insert the flowing data into B+ Tree indexing (No. of Pointers = 4)
(2, 5, 7, 10, 13, 16, 20, 22, 23, 24)

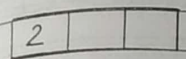
Soln

Here, No of pointer (N) = 4 that means the maximum 3 data can be filled at the Node.

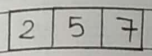
When N = 4 then



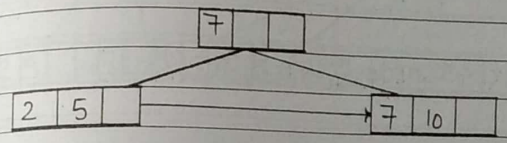
Step I :- Insert 2 in the empty node.



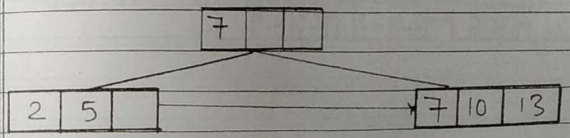
Step II :- Insert 5 and 7



Here this has no any space to insert, this is leaf node, so we break it into two nodes.
The first partition holding ceil value of $(N-1)/2 = \frac{3}{2} = 1.5 \approx 2$ so, and inserting 10 and keeping parent the smallest among second node.

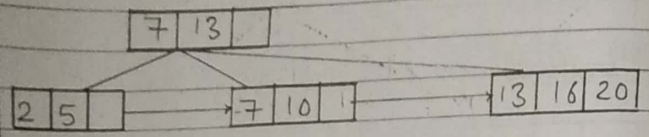


Step III :- Insert 13 then

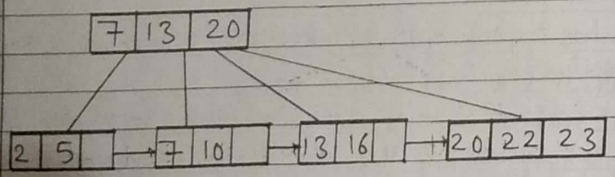


second node is full so break down that node in two segments.

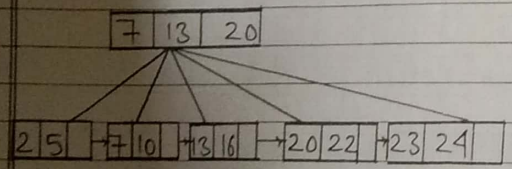
Step IV :- Insert 16 and 20 then,



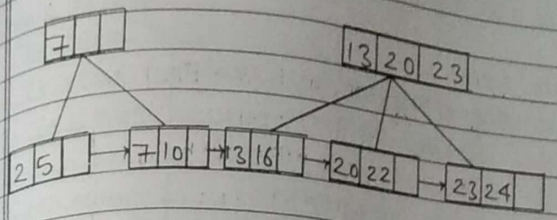
Step V :- Break third node and Insert 22 and 23 then



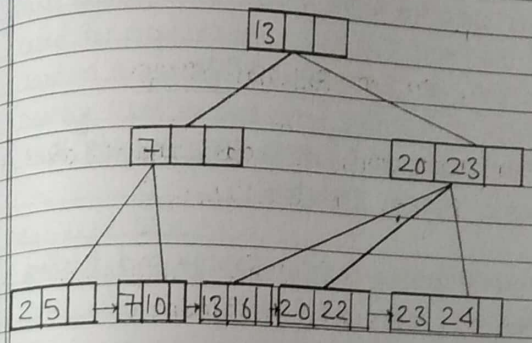
Step VI :- Break 4th node then, insert 24



Step VII :- Non-leaf node is full so break this and 1st partition hold ceil value of $\left(\frac{N}{2}-1\right) = \frac{4}{2}-1 = 1$ then,



Step IX :- Moving the smallest value from second partition to parent node.



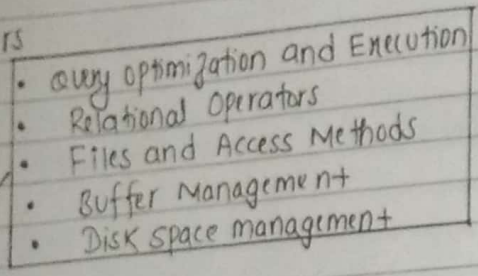
Complete B^+ tree //

Chapter 8 :- File organization and Indexing.

File organization

- It is a method of arranging the records in a file when the file is stored on disk. A relation is typically stored as a file of records.

- DBMS Layers



Stores records in a file in a collection of disk pages. Keep track of pages allocated to each file. Tracks available space within pages allocated to the file.

Data on External storage

- A DBMS stores vast amounts of data and the data has to be preserved across program executions.
- Therefore, data is stored on external storage and fetched into main Memory as needed for processing.
- The unit of information that is read and written to a disk in page.
- Higher layer of DBMS views this pages as unified files and can read or write records to these files.

Storage Media.

- Cache :-
  - The fastest and most costly form of storage.
- Main Memory :-
  - The general purpose machine instructions operate in main memory.
  - It is used for data that are avilable to be operated.
  - It is volatile in nature.
- Flash Memory :-
  - is non-volatile in nature.
  - Reading data from flash memory is at speed of that of main memory.
  - Writing data to flash memory is slower.
  - more expensive than disks.
  - used for applications with read workloads that requires fast random acesses.
  - Mostly used in embedded systems like in hand held devices and other digital electronic devices.
- Magnetic Disk :-
  - secondary storage, used for long terms storage of data.
  - Data should be moved to main memory before in use.
  - It is non-volatile in nature.
  - Direct memory Access.
- optical Storage :-
  - Random access of data.
  - used as secondary access of data for long term.
  - mostly found in CD and DVD.
  - Data are stored optically on a disk and read by laser.

- capable of storing high data in GB.
- Found in both read only and read write form is as CD-R, CD-RW, DVD-R, DVD-RW.

- Magnetic Tape :-
  - It is reffered as sequential access storage.
  - primarily used for backup and archival data.
  - cheaper that than disks but also access to data is slower.

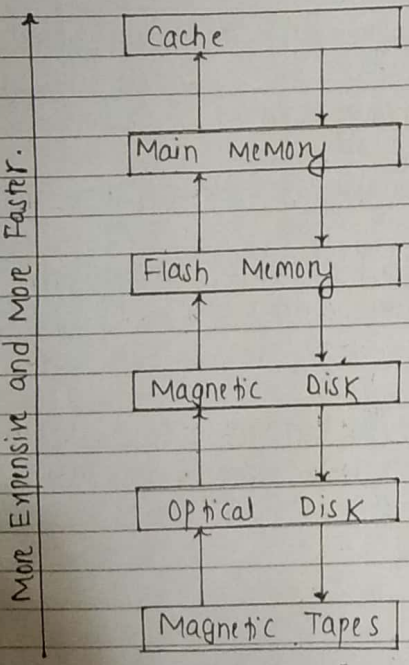


Fig:- Generation of storage media.

File organization in Details Explanations.

- Method for arranging a collection of records and supporting the concept of a file.

- A file is organized logically as a sequence of records. These records are mapped into the disk blocks.

- Each File is logically partitioned into fixed length storage units called blocks, which are the units of both storage allocation and data transfer.

- Normally the block sizes ranges from 4 to 8KB by default to 6B.

- Blocks contains records that is determined by the form of physical data organization being used.

- The size of the records should not be greater than block considering the fact that the field might be of image and it can varies in size dramatically.

- The another fact is that the entire record should be contained in one block rather than divided into separate blocks. This will benefits in speeding up the data access.

- Records may varies in size in one particular file system, so it is to be addressed.

Fixed length Records.

- Giving the attributes fixed size for an entity in terms of byte.
- For the file student, record may contain id int, name varchar (20), age int. For each record the space required is 24 bytes.

- Problems :-

- The file record are of the same record type, but one or more of the fields are of varying size.

- The file records are of the same record type, but one or more of the fields may have multiple values for individual's records.

- The file records are of same record types but one or more of the fields are optional.

- But it is very hard to occupy the block size of multiple of 24 bytes. so, it may be possible that when we the block size obtain its max limit, it might be possible that the information of records may shift to another block.

- while deleting the records from the file, it should be either marked as deleted or should be occupied by the next new record.

- Solutions :-
- The records should be entered into the block after the block size computation (dividing block size with the record size),

by leaving the remaining block size unused.

- while deleting record, we could move the record that came after it into the space formerly occupied by the deleted record and so on, until every record following the deleted record has been moved ahead. or it might be easier to move the final record of the file into the space occupied by the deleted record.

- The file header is used for storing the information of deleted records as well as available records. While inserting the new record, if the file contains the deleted spaces it will insert the record into that position and so on, if it does not contain any deleted position the record is placed at the end of the file.

Record 0	A-102	Kalimati	4000
Record 1	A-201	Patan	5000
Record 2	A-302	Bhaktapur	6000
Record 3	A-402	Kalanki	4000
Record 4	A-103	Kalimati	5000
Record 5	A-502	Kritipur	3000
Record 6	A-105	Kalimati	5000
Record 7	A-202	Patan	4000
Record 8	A-403	Kalanki	4000

Record 0	A-102	Kalimati	4000
Record 1	A-201	Patan	5000
Record 3	A-402	Kalanki	4000
Record 4	A-103	Kalimati	5000
Record 5	A-502	Kritiputr	3000
Record 6	A-105	Kalimati	5000
Record 7	A-202	Patan	4000
Record 8	A-403	Kalanki	4000

Record 2 deleted and all records moved up.

Record 0	A-102	Kalimati	4000
Record 1	A-201	Patan	5000
Record 8	A-403	Kalanki	4000
Record 3	A-402	Kalanki	4000
Record 4	A-103	Kalimati	5000
Record 5	A-502	Kritipur	3000
Record 6	A-105	Kalimati	5000
Record 7	A-202	Patan	4000

header			
Record 0	A-102	Kalimati	4000
Record 1			
Record 2	A-302	Bhaktapur	6000
Record 3	A-402	Kalanki	4000
Record 4			
Record 5	A-502	kritipur	3000
Record 6			
Record 7	A-202	Patan	4000
Record 8	A-403	Kalanki	4000

Free list after deletion of records 1, 4, 6

Variable Length Records

- It is used for :-
  - storing multiple record type in a file.
  - record type that allow variable lengths for one or more fields.
  - record types that allow repeating fields, such as arrays.

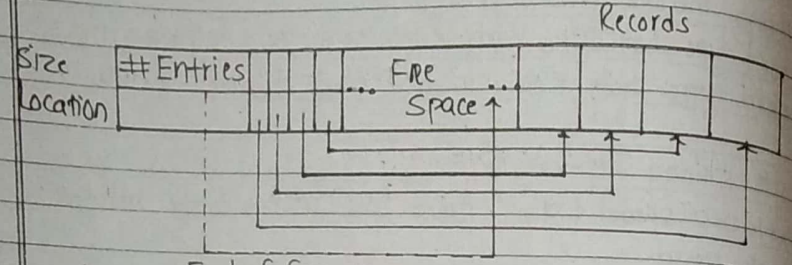
- Byte string Representation :-
  - Attach a special end of records (⊥) symbol to the end of each record.
  - store each record as a string of consecutive bytes.

- Disadvantages :-
  - Not easy to reuse space occupied formerly by a deleted record.
  - No space for records to grow longer.

0	Kalimati	A-102	4000	A-103	5000	A-105	5000 ⊥
1	Patan	A-201	5000	A-202	4000	⊥	
2	Bhaktapur	A-303	6000	⊥			
3	Kalanki	A-402	4000	A-403	4000	⊥	
4	Kritipur	A-502	3000	⊥			

- Slotted page structure :-
  - Header consists of
    - number of record entries.
    - end of free space in the block
    - location and size of each record.
  - Records can be moved around within a page to keep them contiguous with no empty space between them; entry in the header must be updated.

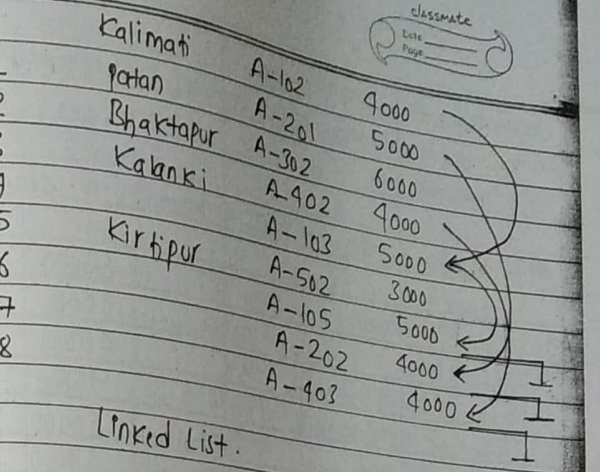
- pointers should not point directly to record - instead they should point to the entry for the record in header.



- Fixed length Representation :-
  - Two ways :-
    - Reserved space.
    - List Representain.

0	Kalimati	A-102	4000	A-103	5000	A-105	5000
1	Patan	A-201	5000	A-202	4000	⊥	⊥
2	Bhaktapur	A-303	6000	⊥	⊥	⊥	⊥
3	Kalanki	A-402	4000	A-403	4000	⊥	⊥
4	Kritipur	A-502	3000	⊥	⊥	⊥	⊥

Reserved space Method.



Linked List.

Organization of Records in Files.

- Heap
  - a record can be placed anywhere in the file where there is space.
- sequential
  - store records in sequential order, based on the value of the search key of each record.
- Hashing
  - a hash function computed on some attributed of each record; the result specifies in which block of the file the record should be placed.
- Records of each relation may be stored in a seperate file. In a multitable clustering file organization records of several different relations can be stored in the same file.
  - Motivation: store related records on the same block to Minimize I/O.

Sequential File organization.

- Suitable for applications that requires sequential processing of the entire file.
- The records in the file are ordered by a search-key

10101	Srinivasan	Comp. Sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Phyics	95000
32343	El said	History	60000
33456	Gold	physics	87000
45565	Katz	Comp. sci	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci	92000
98345	Kim	Elec. Eng.	80000

- Deletion – use pointer chains
- Insertion – locate the positions where the record is to be inserted
  - if there is free space insert there.
  - if no free space, insert the record in an overflow block.
  - In either case, pointer chain must be updated.
- Need to reorganize the file from time to time to restore sequential order.

10101	Srinivasan	Comp. Sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci	92000
98345	Kim	Elec. Eng	80000
32222	Verdi	Music	48000

Multitable clustering File organization.

- Store several relations in one file using a multitable clustering file organization.

department

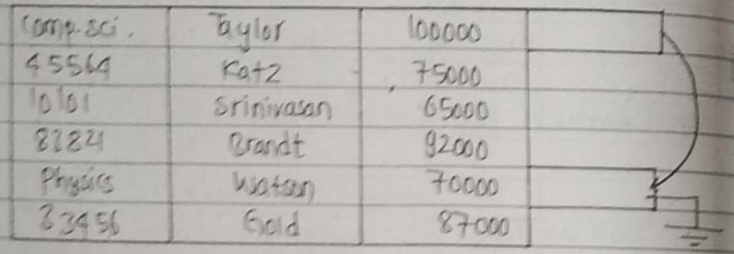
dept_name	building	budget
Comp_Sci	Taylor	100000
Physics	Watson	70000

instructor

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

	comp.sci	Taylor	100000
	45564	Katz	75000
mutitable	10101	srinivasan	65000
clustering of	83821	Brandt	92000
department and	Physics	Watson	70000
Instructor	33456	Gold	87000

- Good for queries involving department instructor, and for queries involving one single department and its instructors
- Bad for queries involving only departments
- result is variable size records.
- can add pointers chain to link records of a particular relation



Clustering File structure with pointer chain.

Indexing

- Indexing mechanisms are used to speed up access to desired data.
- search key - attribute to set of attributes used to look up records in a file.
- An index file consist of a records (called index entries) of the form

search-key	pointer

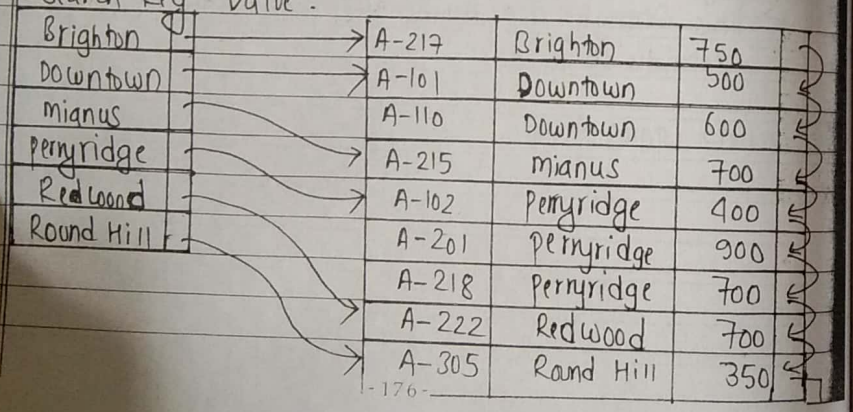
- Index files are typically much smaller than the original file.
- Two basic kinds of indices:
  - ordered indices: search keys are stored in sorted order.
  - Hash indices: search keys are distributed uniformly across "buckets" using a "hash functions".
- Index Evaluation Metrics:-
  - Access time
  - Insertion time
  - Deletion time
  - space overhead
  - Access types supported efficiently. Eg,
    - records with a specified value in the attribute
    - or records with an attribute value falling in a specified range of values.
    - This strongly influences the choice of index, and depends on usage.

1) ordered indices

- In the ordered indices, index entries and stored sorted on the search key value. Eg, author catalog in library.
- Primary index:- in a sequential ordered file, the index whose search key specifies the sequential order of the file.
  - Also called clustering index.
  - The search key of a primary index is usually but not necessarily the primary key.
- Secondary index:- an index whose search key specifies an order different from the sequential order of the file.
  - Also called non-clustering index.
- Index-sequential file: ordered sequential file with a primary index.

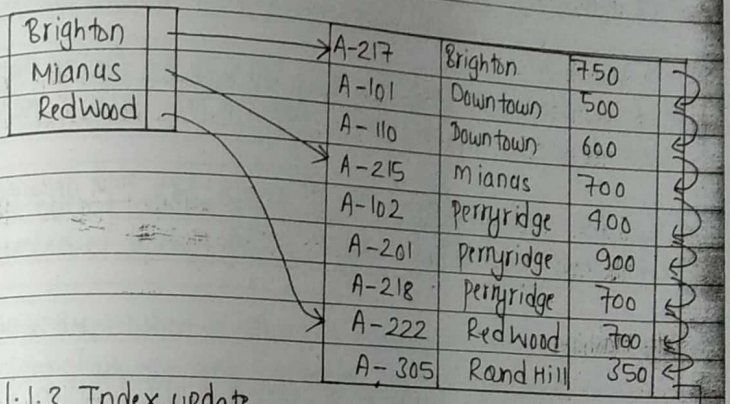
1.1.1 Dense index

- Index record appears for every search-key value in the file.
- In dense primary index, the index record with that search key value and a pointer to the first data record with that search key value.



1.1.2 Sparse Index.

- Index records appears for only some search-key values.
- Here also the index record contains the search key value and a pointer to the first data record with that search key value.
- only applicable when records are sequentially ordered on search-key.
- To locate a record with search-key value K we.
- Find index record with largest search-key value <= K.
- Search file sequentially starting at the record to which the index record points



1.1.3 Index update

- Deletion
  - If deleted record was the only record in the file with its particular search-key value, the search-key is deleted from the index also.
  - Single-level index deletion:
    - Dense indicies - deletion of search-key: similiar to file record deletion.
    - Sparse indices -
      - if an entry for the search key exists in the index, it is deleted by replacing the entry in the index with the next search-key value in the file (in search-key order).

- If the next search-key value already has an index entry, the entry is deleted instead of being replaced.

- Insertion
- Single-level index insertion:
  - perform a lookup using the search-key value appearing in the record to be inserted.
  - Dense indices:- if the search-key value does not appear in the index, insert it.
  - Sparse indices:-
    - if index stores an entry for each block of the file, no change needs to be made to the index unless a new block is created.
    - If a new block is created, the 1st search-key value appearing in the new block is inserted into the index.

Hashing

- Hashing is an effective technique to calculate direct location of data record on the disk without using index structure.
- It uses a function, called hash functions and generates address when called with search key as parameters. Hash function computes the location of desired data on the disk.
- Hash organization:
  - Bucket:- Hash file stores data in bucket format. Bucket is considered a unit of storage. Bucket typically stores one complete disk block, which in turn can store one or more records.

Hash functions: A hash function h, is a mapping function that maps all set of search-keys K to the address where the actual records are placed. It is a function from search keys to bucket addresses.

- Choose hash functions that assign search-key values to buckets in such a way that distribution is either uniform or random.

Static Hashing

In static hashing, when a search-key value is provided the hash function always computes the same address.

- eg:- if mod-4 hash function is used then it shall generate only 5 values.
- The output address shall always be same for that function. The numbers of buckets provided remain same at all times.

- Insertion: when a record is required to be entered using static hash, the hash function h, computes the bucket's address for search key K, where the record will be stored.
- Search: When a record needs to be retrieved the same hash functions can be used to retrieve the address of bucket where the data is stored.
- Delete:- This is simply search followed by deletion operation.

=> Example of Hash file organization

Name	Address	phone	salary	Post
Gill	KTM	456789	20000	Engineer
Van	BKT	654321	20000	Engineer
Robert	KTM	456789	20000	Engineer
Brown	BKT	654321	10000	Overseer
Albert	KTM	454545	10000	officer

~~Salary is dependent on post so,~~

a non-prime attribute is dependent on another non-prime attribute
It has Transitive Dependency.

Dividing the table.

Table 1:

Name	Address	Phone
Gill	ktm	456789
Van	BKT	654321
Robert	Ktm	456789
Brown	BKT	654321
Albert	Ktm	45454

Table 2:

salary	post
2000	

phone no can be Address, we can't find phone.

Name	salary	post
Gill	20000	Engineer
Van	20000	Engineer
Robert	20000	Engineer
Brown	10000	Engineer
Albert	10000	Owrseer
		officer

Phone	Address
456789	

~~Name~~ → Roll No → Name,

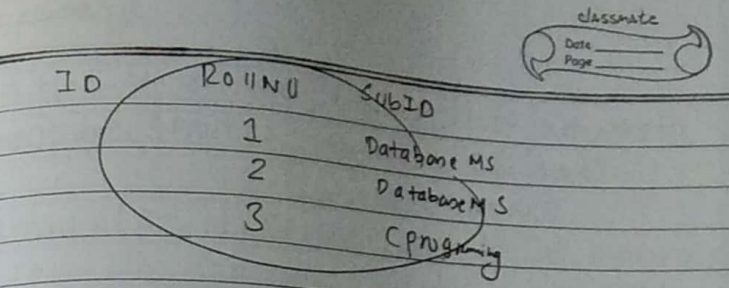
SubID → SubName, Feepaid.

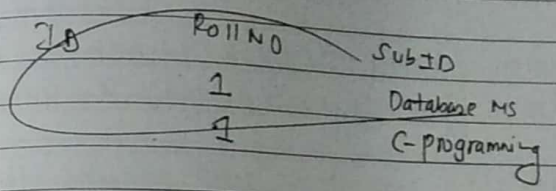
Roll No	Name
1	Hariman Dangol
2	Mohan Prasad sah
3	Indira Rimal

table 1

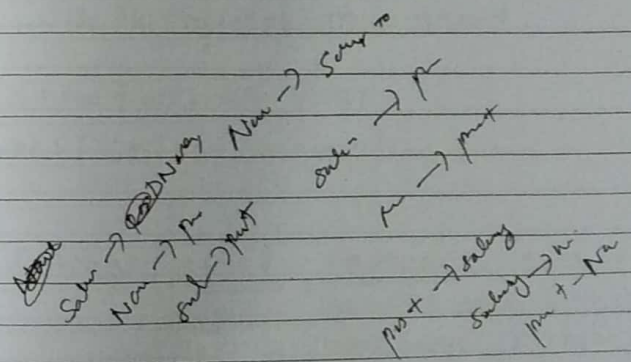
SubID	SubName	Feepaid
DBMS	Database MS	2000
DBMS	Database	2000
DBMS	Database	2000
CPROG	Cprogramming	2000
CPROG	Cprogram	15000
Math	mathematics	8000

ID → Roll No, SubID





ID	Roll No	SubID
1	1	DBMS
2	2	DBMS
3	3	DBMS
4	1	CPROG
5	2	CPROG
6	3	Math



Stdmaster (st_id, st-name, Instructor-id, Inst_name, Course_id1, Cane-name1, courseid2, Cansename2, courseid3, Casename3)

1) Process for 1NF

Stdmaster (st-id, St_name, instructor-id, instructor-name, Course_id1, Course-name1, courseid2, Course_name2, Course-id3, Course-name3)

- In the stdmaster table, the repeating group is a course information. A student can take many courses.
- Remove the repeating group. In this case, it's the course information for each student.
- Identify the primary key for your new table.
- The primary key must uniquely identify the attribute value. (st-id, Course_id1, courseid2, courseid3)
- After removing all the attributes related to the course and student, you are left with the student course table (Student+Course).
- The Student table (student) is now in 1st normal form with the repeating groups removed.
- The two new tables are shown

student (st-id, St_name)

Student+Cause (st_id, courseid1, course_id3, courseid2, instructor_name, instructor_id, Cause-name1, Cause-name2, Cansename_3)

2) Process for 2NF

- To move to 2NF, a table must be in 1NF.
- The student table is already in 1NF. because it has a single-column primary key.
- when examining the student course table, we see that not all attributes are fully dependent on the primary key; specifically, all course information. ~~The only attribute that is fully~~
- Identify the new table that contains the course information.
- Identify the primary key for the new table.
- The three new tables are shown in new table.

Student (st_id, st_name)

Course_Id (st_id, course_id1, course_id2, course_id3, course_name1, course_name2, course_name3)

CourseInstructor (st_id,

Chapter 7, 8 / 9, 10 11, 2

3, 5

Let table be UnNormalized then, we are taking taking table in first Normal form 1NF

A table is in 1NF if there is no repeating values in a column.

then, that table 1 is in 1NF.

Now, lets check for 2NF.

1NF and

RollNO → Name

SubID → SubName, Feepaid

RollNO	Name	SubID	SubName	Feepaid
1	Hariman...	DBMS		
2	mohan Prasad	DBMS		
3	Indira ...	DBMS		
		Cprog		
		Cprog		
		nah		

Normalization Vs Denormalization

Normalization	DeNormalization
• It is used to remove redundant data from the db and to store non-redundant and consistent data into it.	• Denormalization is used to combine multiple table data into one so that it can be queried quickly.
• It mainly focuses on clearing the db from unused data and to reduce the data redundancy and inconsistency.	• It on the other hand focus on to achive the faster execution of queries through introducing redundancy.
• During Normalization as data is reduced so a number of tables are deleted from the database hence data tables are lesser in number.	• It is integrated in the same db and hence a number of tables to store that data increases in number.
• Normalization uses optimized memory and hence faster in performance.	• Denormalization introduces some sort of wastage of memory.
• Normalization maintains data integrity.	• It doesn't maintain any data integrity.
• It is generally used where number of insert/update/delete operations are performed and joins of those tables are not expensive.	• Denormalization is used where joins are expensive and frequent query is executed on the tables.

SQL Queries

2020 Fall

b) Write SQL Statements

i) Create a table named Automotor with chasis_number as primary key and following attributes: veh_brand, veh_name, veh_model, veh_year, veh_cost, veh_color, veh_weight

```
create table Automotor (Chasis_number int primary key, veh_brand varchar(50), veh_name varchar(50), veh_model varchar(50), veh_year year(4), veh_cost int, veh_color varchar(50), veh_weight int )
```

ii) Enter a full detailed information of an automotor.

```
Insert into Automotor values (1, 'Bajaj', 'Chary', 'Ferari', 2012, 200000, 'Red', 3000),
(2, 'Yamaha', 'Polis', 'Ferari', 2015, 50000, 'yellow', 2000)
(3, 'Maruti', 'Audi', 'Bmw', 2020, 700000, 'Red', 3050)
```

iii) Change any Automotor's year to 2019

```
Update Automotor set veh_year = '2019' where Chasis_number=1
```

iv) Remove all Automotor records whose model contains character 'i' in last position.

```
Delete from Automotor where veh_model like '%i'
```

v) Display the total cost of all vechicles of the table Automotor

```
Select sum(veh_cost) from Automotor
```

vi) Create a view from above table having vehicles only red color.

```
Create view vehicle as select * from Automotor where
veh_color = 'red'
```

vii) Display details of Automotor ordering on descending manner by brand name and by ascending on model when brand matches

thomas.

```
Select * from Automotor
group by veh_model Asc
order by veh_brand DESC
```

viii) Change data type of color so that it only takes one character

```
Alter table thomas.Automotor
modify column veh_color char(1)
```


i) Create a database named hotel and table relation

```
// Create database hotel

// create table Hotel_details (hotel_id int primary key,
hotel_name varchar(50), estb_year year(4),
hotel_star varchar(50), hotel_worth varchar(30))
```

ii) Create a view named price which shows hotel name and its worth

```
Create view price as select hotel_name, hotel_worth
```

iii) Modify the data so that Hotel Chitwan is now four star level.

```
Update Hotel_details set hotel_star = 'Four' where
hotel_name = 'Hotel Chitwan'
```

iv) Delete the records of all hotels having worth more than 9M.

```
Alter table Hotel_details
Drop column hotel_name where hotel_worth,
'9M'
```

2019 Fall

2b)

i) Display ID of patient admitted to hospital at pokhara and whose name ends with s.

```
Select patientID from patients where
Address = 'pokhara' and patientName Like '%s'
```

ii) Delete the record of doctors whose salary is greater than average salary of doctors.

```
Delete from Doctors where salary > AVG(salary)
```

iii) Increase the salary of doctors by 18.5% who works in opd departements.

```
Update Doctors set salary = Salary * 1.185 where
Department = 'opd'.
```


iv) Find the average salary of doctors for each address who have average salary more than 55K.

```
select avg(salary), address from Doctors
where avg(salary) > 55K
```

2018 - spring

2b) write sql statements for followings.

i) create a table named vehicle with veh-number as a primary key and following attributes:

veh_type, veh_brand, veh_year, veh_mileage, veh_owner, veh_photo, veh_price.

```
Create table vehicle ( veh_type varchar(50), veh_brand
varchar(50), veh_year year(4), veh_mileage int,
veh_owner varchar(50), veh_Photo varBinary(max),
veh_price int )
```

ii) Enter a full detailed information of a vehicle.

```
insert into vehicle values ( SUV, ford, 2010, 120,
Hariram, 0x3721, 200000),
(UMATRUCK, Figo, 2111, 70, pikachu, 0x973210,
7000000),
(crane, Ferari, 2019, 20, Rachel, 0x9722001,
10000000)
```

iii) Increment vehicle price by 10,000

update vehicle set veh_price = veh_price + 10000,

iv) Remove all vehicle's records whose brand contains character 'o' in second position.

Delete from vechile where ven_brand like '_o%'

v) Display the total price of all vehicles.

Select sum(veh_price) from vehicles.

vi) Create a view from above table.

creat view automotor

vii) Display details of vehicles ordering on descending manner in brand and by milage when brand matches.

Select all from vehicles
group by veh_~~brand~~ Mileage Desc
order by veh_brand Desc

viii) Change the data type of year to datetime.

Alter table Vehicle
modify column veh_year datetime

Doctor (Name, age, address)
works (Name, Depart_no, salary)
Department (Depart_no, depname, floor, room)

i) Display the name of doctor who do not work in any department.

works
Select name from ~~doctor~~ where depart_no
not in (select depart_no from Department)

ii) modify the database so that Dr. Hari lives in pokahara.

Doctor
Upadte ~~address~~ set address = Dr. Hari when
address = pokhara

iii) Delet all record of Doctor working in opD department

Delect from Doctor

iv) Display the name of doctors who work in at least two departments.

Select name from ~~doctor~~ inner join works
on Doctor.Dame = works.Name innerjoin
department on works.Depart_no = department.Deprt_no
having count (department) >= 2

Old is gold Solution of Chapter 5

2011 Fall (VVI)

a) What is refrential integrity? Explain with an example about functional dependencies and multivalued dependencies.

⇒ Refrential integrity

- Refrential integrity refers to the accuracy and consistency of data with a relationship.
- It means the refrence from a row in one table to another table must be valid.
- For eg:- If we delete row number 15 in a primary table, we need to be sure that there's no foreign key in any related table with the value of 15. We should only be able to delete a primary key if there are no associated rows. otherwise, we would end up with an orphaned record.

Primary Table

Companyid	Company Name
1	Apple
2	Samsung

✓ = Associated Record

X = orphaned record

?

Related Table

companyid	Productid	Product Name	
1	1	iphone	✓
15	2	Mustang	X

- Here, the related table contains foreign key value that doesn't exist in the primary key field of primary table (i.e. the "companyid" field). This has resulted in an "orphaned record".

⇒ Refrential integrity will prevent users from :-

- Adding rows to a related table if there is no associated row in the primary table.
- Changing values in a primary table that result in orphaned records in a related table.
- Deleting rows from a primary table if there are matching related rows.

Functional dependencies

- It is the fundamental to the process of Normalization which describes the relationship between attributes in a table.
- It helps to maintain the quality of data in the database.
- It plays a vital role to find the difference between good and bad database design.
- A functional dependencies is denoted by an arrow "→".
- For attributes X and Y in a Relation R, functional dependency between X and Y are shown as:-

$$X \rightarrow Y$$

Here

- X is a determinant and,
- Y is the functional dependent on X.

- For example

eid	ename	age	salary
101	Ram	35	150000
102	Sita	30	100000

Here, eid → ename ; eid uniquely determines ename.
ename → eid ; is not always true.
eid → age ; True
age → eid ; False
eid → salary ; True
age → salary ; False.

(PTO 1)

Multivalued dependencies

- Multivalued dependency occurs in the situation where there are multiple independent multivalued attribute in the single table.
- A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.
- In a Relation R(A,B,C) if each 'A' value has associated with it a set of 'B' values and a set of C values such that 'B' and 'C' values are independent of each other, then the relation is said to have multivalued dependency.

$$A \twoheadrightarrow B \ , \ A \twoheadrightarrow C$$

eg:-

Model	year	color
M1	2000	Red
M2	2012	Black
M3	2016	Blue
M4	2021	White

Here Model → year and Model → color.

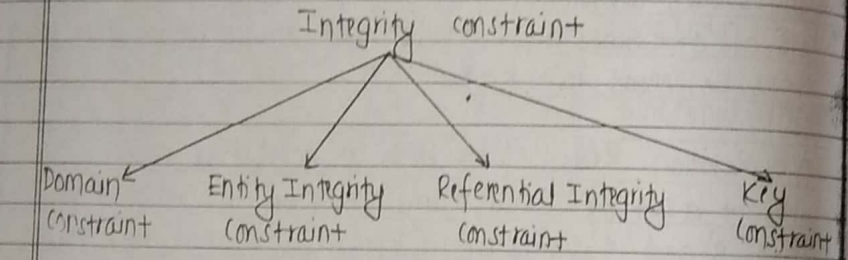
2011-spring (VVI) (2014 Fall) (2013 spring) All time asked

3a) What is integrity constraints? In how many categories the integrity constraints can be classified? Explain each.

Ans:-

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of integrity constraint



i) Domain constraint

- Domain constraints can be defined as the defination of a valid set of values for an attribute.
- The datatype of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in corresponding domain.

eg:-

ID	name	sem	Age
1	Tom	1st	17
3	Cely	2nd	19
4	Kate	8th	B

Not Allowed, because Here, AGE is an integer attribute.

ii) Entity Integrity constraint

- It states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

eg:-

Emp_Id	Emp_Name	Salary
123	Jack	300000
143	Harry	500000
154	Jakson	700000
	Zayn	200000

↓

Not allowed as primary key can't contain a Null value.

(PTO2) pg 2

iii) Referential integrity constraint

iv) Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.

- An entity key set contains multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

eg:-

ID	Name	sem	Age
1	Jack	1st	17
2	Sonia	3rd	18
3	Kate	5th	20
2	Morgan	8th	22

↓ Not allowed. Because all row must be unique.

3b) How redundancy is reduced in databases? Explain basic three steps.

- Redundancy is reduced in databases by Normalizintion them.

Normalization

- Database normalization is a technique of organizing data in the database.
- It is the process of removing redundancy and undesirable and characteristics like insertion, update and deletion anamolies
- It involves dividing table into two or more tables and defining relationship between the tables.
- mainly two purpose are:-
  - Eliminating redundant data.
  - Ensuring data dependencies make sense.

First Normal Form (1NF)

- A relation is in 1NF if and only if all columns are atomic i.e. no repeating values in columns

eg:- Let us take a table of unNormalized Form which contains one or more repeating groups or each row may contain multiple set of values for some columns.

- multiple values in a single row are also called as non-atomic values.

eg:-

cid	Name	Address	Phone
101	Ram	K+m	9863444041
			9879644321
102	Sita	Bhaktpur	9863547633

Fig 1:- UNF

Now, Fig1 in 1NF then,

cid	Name	Address	Phone
101	Ram	K+m	9863444041
101	Ram	K+m	9879644321
102	Sita	Bhaktpur	9863547633

- In 1NF there should not be no repeating columns.

eg:- id Name Phone child1 child2 child3

↓

id Name Phone child

- Values of each attribute is atomic.
- No composite values and no two rows are identical.
- Each column must have unique name.
- All entries in any columns must be of same kind.

Second Normal Form (2NF)

- A relation is in 2NF if,
  - it is in 1NF
  - all attributes depends on full primary key.

eg:-

personID	projectID	Name	projectName	phone
1	1	Ram	Database	9863444041
2	1	Sita	Database	9823048543
1	2	Ram	Web	9863444041
2	2	Sita	Web	9823048543

Fig:-2

In Fig:-2 Given relation is in 1NF but not in 2NF because all attributes are not fully dependent on primary key (personID, projectID)
Now decompose above table as follows:-

personID	Name	phone
1	Ram	9863444041
2	Sita	9823048543

Table 1

id	personID	projectID
1	1	1
2	2	1
3	1	2
4	2	2

Table 3

projectID	projectName
1	Database
2	Web

Table2

Third Normal Form (3NF).

- A relation is in 3NF if:-
  - it is in 2NF
  - There is no transitive functional depencency i.e. There should not be case that non prime attribute is determined by another non prime attribute.

eg:-

st_id	st_name	DOB	zip	city
1	Ram	2055/5/5	01	Kathmandu
2	Sita	2055/2/1	01	Kathmandu
3	Gita	2040/3/1	02	Patan
4	Hari	2045/06/06	03	Dharan

Fig:-3

st_id is a primary key. All other attributes are dependent on st_id, so it is in 2NF but, city, non-prime attribute is dependent on zip, another non prime attribute. so, it is not in 3NF.
Now, decompose abon table as below:-

st_id	st_name	DOB	zip
1	Ram	2055/5/5	01
2	sita	2055/2/1	01
3	Gita	2040/3/1	02
4	Hari	2045/06/06	03

Table 1

zip	city
01	Kathmandu
02	Patan
03	Dharan

Table2

BCNF

- A relation is in BCNF if,
  - it is in 3NF.
  - every determinant in that table is candidate key.
- Advance form of 3NF also referd as 3.5NF
- If table contain only a one candidate key, 3NF and BCNF

eg:-

Student	Course	Teacher
Ram	DB	Shyam
Ram	e-com	sita
Gita	DB	Shyam
Gita	e-com	sita

Here,
Key = {student, course}

F.D i) {student, course} → Teacher
Teacher → course

ii) Teacher is not a candidate key but determines course, above fig '4' is not in BCNF.

Now, Decompose it as follows:-

Teacher	Course
Shyam	DB
Sita	e-com

Table 1

student	course
Ram	DB
Ram	e-com
Gita	DB
Gita	e-com

Table 2

4NF

- A relation R is in 4NF if
  - it is in 3NF or BCNF
  - R contains no multivalued attributes

eg:-

ename	pname	dname
Ram	x	John
Ram	y	sita
Ram	x	Sita
Ram	y	John

Here, ename →→ pname , ename →→ dname

∴ To convert it into 4NF, decompose into

ename	pname
Ram	x
Ram	y

Table 1

ename	dname
Ram	John
Ram	Sita

Table 2

OR

What do you mean by decomposition of relational schema? Explain lossless and lossless decomposition with example.

Decomposition

- when a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies and redundancy.

⇒ Types of Decomposition Properties

i) Lossless Decomposition (Property)

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantee that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

eg:-

- Employee-Department table

Emp-ID	Emp-Name	Emp-Age	Emp-Loc	Dept-ID	Dept-N
1	Thomas	29	K+m	DP+1	Operation
2	Henry	32	K+m	DPT2	HR
3	Tom	22	Texus	DPT3	Finance

Decompose the above tables into two tables.

- Emp-Details table1

Emp-ID	Emp-Name	Emp-Age	Emp-Loc
1	Thomas	29	K+m
2	Henry	32	K+m
3	Tom	22	Texus

- Dept-Details table2

Dept-ID	Emp-ID	Dept-N
DP+1	1	operation
DP+2	2	HR
DP+3	3	Finance

Now, Natural join is applied on above table the the result be:-

Emp-ID	Emp-Name	Emp-Age	Emp-Loc	Dept-ID	Dept-N
1	Thomas	29	K+m	DP+1	operation
2	Henry	32	K+m	DP+2	HR
3	Tom	22	Texus	DP+3	Finance

∴ The above relation has lossless decomposition i.e. no loss of information.

It's different from Lossy Decomposition.

ii) Lossy Decomposition (Dependency preserving property)

- When a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.
- Atleast one decomposed table must satisfy every dependency.

eg:-

- Emp_Info table

Emp_ID	Emp_name	Emp_Age	Dept_ID	Dept_Name
1	Jacob	29	Dpt1	Operations
2	Henry	32	Dpt2	HR

Decompose into two tables:

- Emp_Details table 1

Emp_ID	Emp_name	Emp_Age
1	Jacob	29
2	Henry	32

- Dept_Details table 2

Dept_ID	Dept_Name
Dpt1	operations
Dpt2	HR

- Now, we can't join the above two table because Emp_ID isn't part of the DeptDetails table2.

so, ∴ The above relation has lossy decomposition.

2012 Fall

3a) state and explain about functional dependencies. considering the suitable example.

- It is the fundamental to the process of Normalization which describes the relationship between attributes in a table.

(pTO1.) Page 1

- Types of functional dependencies

1) Full functional dependency

- $X \rightarrow Y$ is a full functional dependency if the removal of any attribute 'A' from 'X' removes the dependency.
- A full functional dependencies occours when it is already functional dependency and the set of attributes on the left side can't be reduced any further

eg:-

orderNO	LineNO	qty	price
01	L1	10	200
02	L2	20	400
03	L3	30	800
04	L4	15	300

Here, $\{orderNO, LineNO\} \rightarrow qty$

$\{orderNO, LineNO\} \rightarrow price$

are the full full functional dependencies.

ii) Partial functional dependency

- A functional dependency $X \to Y$ is a partial dependency if some attributes $A \in X$ can be removed from X and the dependency still holds.

eg:-

$\{eid, Phone\} \to name$

Here,
Remove Phone, then $\{eid\} \to name$ still holds true, so it is partial functional dependency.

iii) Transitive Dependency

- It occurs where there is an indirect relationship that causes a functional dependency.

- If $X \to Y$ and $Y \to Z$, then $X \to Z$ is a transitive dependency.

iv) Trivial dependency

- It occurs when the function dependency of an attribute is described on collection of attributes that indicates the original attribute.

- Functional dependency is Trivial if R.H.S is a subset of LHS.

eg:- $\{name, SSN\} \to SSN$

v) Non-Trivial dependency

- If a F.D $X \to Y$ holds true where 'X' intersection 'Y' is null then the dependency is said to be completely Non-Trivial dependency.

eg:-

$\{S\#, P\#\} \to \{S\#, qty\}$

3b) Define third normal form. Convert the following 2NF relation into 3NF. (consider Name as a primary key.)

Name	Address	Phone	Salary	Post
Gill	K+M	456789	20000	Engineer
Van	BK+	654321	20000	Engineer
Robert	K+M	456789	20000	Engineer
Brown	BK+	654321	10000	overseer
Albert	K+M	454545	10000	officer

(Repeated)

PTO 2

⇒ examples and Explanation of Entity Integrity constraints

- Entity integrity requires that each entity have a unique key.

- eg, if every row in a table represents relationships for a unique entity, the table should have one column or a set of columns that provides a unique identifier for the rows of the table.

- Defining the parent key is called entity integrity.

- Every relation must have PK. Every PK is non-null and unique.

- unique means, value of the PK must be different not be same.

- Null is a value, suppose no other value not assign or apply.

- Nil is not zero(0) or blank ("").

2012 Spring

2b) How does data integrity constraints differs with Data security.

Data security

- Data security defines the prevention of data corruption through the use of controlled access mechanism.
- It deals with the protection of data. Data security is making sure only the people who should have access to the data are the only ones who can access the data.
- Data security refers to making sure that data is accessed by its intended users, thus ensuring the privacy and protection of data.
- Authentication/authorization, encryptions, and masking are some of the popular means of data security.
- It is the protection of data from an unauthorized users. only the authorized users are allowed to access the data.
- For eg:- If we have an account in the "yahoo.com", then we have to give our correct username and password to access our account or email.
- Similiarly, when we insert our ATM card into the ATM machine, the machine reads our ID number printed on the card and then asks us to enter our pin code (or password). In this way, we can access our account.

Data Integrity

- Data Integrity defines the quality of data, which guarantees the data is complete and has a whole structure.
- Data Integrity deals with the validity of data. Data integrity is making sure the data is correct and not corrupt.
- Data Integrity refers to the structure of the data and how it matches the schema of the database
- Backing up, designing a suitable user interface and error detection/correction in data are some of the means to preserve integrity.
- Data integrity means that the data contained in the database is both correct and consistent. For this purpose, the data stored in the database must satisfy certain types of constraints (rules).
- Data in a database must be correct and consistent. So, data stored in the database must satisfy certain rules. and DBMS provides different ways to implement such types of constraints(rules) which improves the data integrity in database.
- For eg:- A balance for any account must not be less than zero. such constraints are enforced in the system by adding appropriate code in application programs. But, when new constraints are added such as balance should not be less than Rs 5000, application programs need to be changed. But it is not an easy task to

3b) Explain about 5th normal form [4 marks]

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as project-join normal form (PJ/NF).

eg:-

Subject	Lecturer	Semester
computer	Anshika	Sem 1
computer	John	sem 1
math	John	sem 1
math	Ashka	sem 2
Chemistry	Praveen	sem 1

In the abon table, John takes both computer and math class from sem 1 but he doesn't take math class for sem2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new semester as sem 3 but do not know about the subject so we leave Lecturer and subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So, to make the abon table into 5NF, we can decompose it into three relations P1, P2 and P3.

P1

Semester	Subject
Sem 1	computer
sem 1	Math
Sem 1	Chemistry
Sem2	Math

P2

Subject	Lecturer
Computer	Anshika
Computer	John
math	John
math	Ashka
Chemistry	praveen

P3

Semester	Lecturer
sem 1	Anshika
Sem 1	John
Sem 1	John
sem 2	Ashka
sem 1	praveen.

2013 Fall

3a) Explain joined dependencies [5 Marks]

- If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in join Dependency.
- It is a generelization of Multivalued Dependency.

Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it can't be decomposed further.

eg:-

- Employee table

EmpName	EmpSkills	EmpJob (Assigned work)
Tom	Networking	EJ001
Harry	Web developement	EJ002
Katie	Programming	EJ002

The above table can be decomposed into the following three tables; ∴ it is not in 5NF:

- EmployeeSkills table.

EmpName	EmpSkills
Tom	Networking
Harry	Web developement
Katie	Programming

- EmployeeJob Table.

EmpName	EmpJob
Tom	EJ001
Harry	EJ002
Katie	EJ002

JobsKills table

EmpSkills	EmpJob
Networking	EJ001
Web developement	EJ002
programming	EJ002

- our Join Dependency -

✓ {(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}

- The above relation have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation Employee table.

2013 spring

3a) What do you mean by Functional dependency, Multivalued dependency and Transitive dependency? Why normalization is needed? Assume the un-Normalized relation given below and find the final normalized logical ER diagram normalizing the un-normalized relation up to 3NF explaining what you are going to check at each steps.

Roll NO	Name	SubID	SubName	FeePaid
1	Hariman Dangol	DBMS	Database MS	200000
2	Mohan Prasad sah	DBMS	Database MS	200000
3	Indira Rimal	DBMS	DatabaseMS	300000
1	Hariman Dangol	CPROG	C-Programming	200000
2	mohan Prasad sah	CPROG	C-Programming	150000
3	Indira Rimal	MATH	Mathematics	300000

Normalization is needed for :-

- To minimize data redundancy i.e. no unnecessarily duplication of data.
- To make database structure flexible i.e. it should be possible to add new data values and rows without reorganizing the database structure.
- Data should be consistent throught the database i.e. it should not suffer from following anamolies.

- Insert Anomaly - Due to lack of data i.e., all the data avilable for insertion such that null values in keys should be avoided. This kind of anomaly can seriously damage database.

- Update Anomaly - It is due to data redundancy i.e. multiple occurrences of some values in a column. This can lead to inefficiency.

- Deletion Anomaly - It leads to loss of data for rows that are not stored else where. It could result in loss of vital data.
- complex queries required by the user should be easy to handle.

Let us consider,
The table is in unNormalized then to maken the table in 1NF.

- A relation is in 1NF if and only if all columns are atomic i.e. No repeating values in columns.

Now,
Here the table is in 1NF because No repeating values in

column.

Here To convert the table to 2NF. A table is in 2NF only if :- - It is an 1NF.
- all attributes depends on fully primary key.

Here, the table is in 1NF but Here, all attributes are not fully dependent on primary key

- RollNo depends on Name (Roll → Name)
- SubID depends on SubName, Feepaid (SubID → SubName, Feepaid)

but our decomposition is Lossless decomposition, there should not be the loss of information when table is breakdown.
Now, breaking table.

RollNo	Name
1	Hariman Dangol
2	Mohan Prasad sah
3	Indira Rimal

Table 1

SubID	SubName	Feepaid
DBMS	Database Ms	2 00000
DBMS	Database Ms	300000
CPROG	Cprogramming	2000000
CPROG	C-programming	150000
Math	Mathematics	300000

Table 2

ID	Roll NO	SubID
1	1	DBMS
2	2	DBMS
3	3	DBMS
4	1	CPROG
5	2	CPROG
6	3	Math

Table 3

Here TO convert table to 3NF
A Table is in 3NF if only if
- It is in 1NF
- Non prime attribute should not determine non-prime attribute (No Transitive dependency should occur)

And, Here all tables Table 1, Table 2, Table 3 are in 3NF because it satisfy all the above condition.

2012-Fall

3b) convert the following 2NF relation into 3NF (Name as a PK)

Name	Address	phone	Salary	Post
Gill	K+M	456789	20000	Engineer
van	BK+	654321	20000	Engineer
Robert	K+M	456789	20000	Engineer
Brown	BK+	654321	10000	Overseer
Albert	K+M	454545	10000	officer

Ans:-

Here, table is in 2NF because it is in one NF and all attributes fully dependent on the primary key Name

Now,
converting the table to 3NF

A relation is in 3NF only if
- It is in 2NF.
- There is no transitive functional dependency. i.e. There should not be case that non-prime attribute is determined by another non-prime attribute.

Here, Name is a primary key, and Here showing that attributes post and salary has transitive functional dependency.
Here, salary is determined by the post even if they both are non-prime attribute. so, Here one non-prime attribute determines another non-prime attribute so, the table is not in 3NF Now, breaking the table,

Name	Address	phone'	post
Gill	K+M	456789	Engineer
van	BK+	654321	Engineer
Robert	K+M	456789	Engineer
Brown	BK+	654321	overseer
Albert	K+M	454545	officer

Table 1

post	salary
Engineer	20000
overseer	10000
officer	10000

 Table 2

Here, Table 1 (Name, Address, Phone, Post)
Table2 (post, salary)

- Both table are in 3NF because it satisfies the above 3NF criteria.

2014 Fall all repeated.

2014 spring 4a repeated

4b) What is Normalization and why it is done? Give an example to normalize table to 3NF if table is not in 3NF.
Repeated.

2015 Fall

3a) Repeated.

6a) Explain the roles of Assertions and Triggers in SQL - when Triggers are not appropriate to use? Give an example.

⇒ Assertions

- An assertions is a statement in SQL that ensures a certain condition will always exist in the database.

- Assertions are like column and table constraints, except that they are specified separately from table definations. However assertions are CHECKED only when UPDATE or INSERT actions are performed against the table.

- We can use Assertions when we know that the given particular condition is always true. Assertions do not maintain any track of changes made in table. It have small syntax as compared to triggers and Modern database do not use assertions

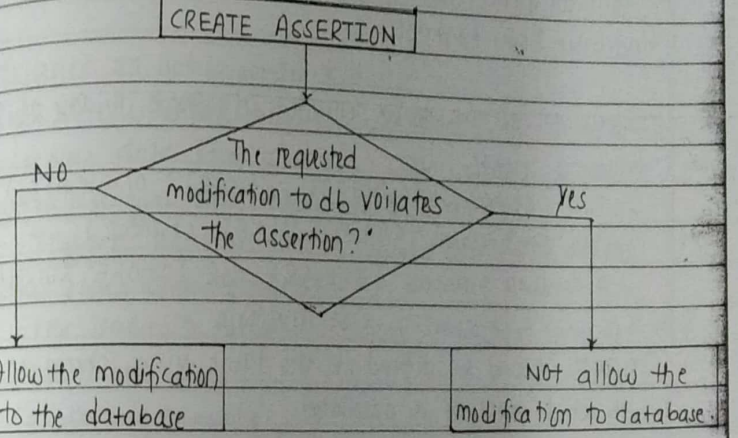
- SYNTAX:- CREATE ASSERTION [assertion-name] // TO CHECK ([condition]); // create assertion
DROP ASSERTION <assertion-name> // TO delete assertion

- Domain constraints and referential integrity are special form of assertions.

- When an Assertions are created, the system tests it for validity. If the assertion is valid, any further modification to the database is allowed only if it does not cause that assertion to be violated.

- Assertions can't modify the data and they are not linked to any specific tables or events in the database.

- simple flow-chart on working of Assertion



- When a constraints can't be expressed in the Domain constraints, functional dependency and referential integrity we use assertions eg:-

Ensuring the sum of loan amounts for each branch is less than the sum of all acounts balances at the branch.

Create assertion sum-constraints check

(not exists (select * from branch where
(select sum) amount) from loan
where (loan.bname = branch.bname >= (select sum)
amount) from account where
(account.bname = branch.bname)))

⇒ Triggers

- A triggers is a database object that is associated with the table, it will be activated when a defined action is executed for the table.
- Triggers are stored programs, which are automatically executed or fired when some event occours.
- Triggers are written to be executed in response to any of the following events.
  - DML statements (DELETE, INSERT, or UPDATE).
  - DDL statements (CREATE, ALTER, or DROP)
  - A database operations (SERVERERROR, LOGOFF, STARTUP, or SHUTDOWN).
- Triggers could be defined on the table, view, schema, or database with which the event is associated.
- We can use Triggers even particular condition may or may not be true. Triggers maintain track of all changes occoured in table. They have large syntax to indicate each specific of the created triggers. Triggers are very well used in modern databases.
- Triggers are more powerfull because they can check conditions and

also modify the data within the tables insides a database; and it can be invoked before or after the event.

⇒ Advantages of triggers

- It generates some derived column values automatically and also Enforces referential integrity.
- Auditing, Event logging and storing information on table access, also synchronous replication of tables.
- Imposing security authorizations, Preventing invalid transactions.

⇒ Disadvantages of triggers

- cannot replace all validations, Impose load on server, Invisible from client applications, Not recommended for high velocity of data.

⇒ Triggers are not appropriate to use

- Risk of unintended execution of triggers, for example, when
  - loading data from a backup copy.
  - Replicating updates at a remote side (Site)
  - Triggers execution can be disabled before such actions.
- other risks with triggers:
  - Error leading to failure of critical transactions that set off the triggers.
  - Casacading execution.

⇒ Restrictions applied to Mysql triggers -

- only one triggers for each timing/event, Return statement is not permitted, Foreign key restrictions, Out dated Metadata
- Cannot use 'CALL' statement, cannot create a TEMPORARY table or a view, Not activated by changes in INFORMATION_SCHEMA

SYNTAX:- Create trigger [trigger_name]
[before | after]
{insert | update | delete} on
[table_name]
[for each row]
[trigger_body]

eg:-

```
Create trigger t1 before UPDATE on sailors

for each row

begin

  if new.age>60 then
     set new.age =old.age;

  else
     setnew.age=new.age;

  end if;
  end;
$
```

- In the above example we are creating triggers before update. so, if the new age is greater than 60 we should not update else we should update. We can call the trigger by using "$" symbol.

2015 spring 4 a,b Repeated 2016 spring 4,a,b
2016 Fall 3 a,b Repeated

2017 Fall 4a) repeated.

5b) What is a database anomaly? Explain different types of databases anomaly with suitable example.

- Anomalies are problems that can occour in poorly planned, un-normalized databases where all the data is stored in one table (a flat-file database)
- Tables that have redundant data have problems known as anomalies.
- An anomaly is an abnormality, a blip on the screen of life that doesn't fit with the rest of the pattern.
- There are three types of anamolies in DBMS they are:-

i) Insert Anomalies :- It occours when certain attributes can't be inserted into the database without the presence of other attribute.
- It is the inability to add data to the database due to the absence of other data.

- The nature of the database may be such that it is not possible to add a requred piece of data unless another piece of unavilable data is also added.

eg:-
The primary key for this table is the combination of Emp-ID and Course-Title.

EmpID	Ename	Dept Name	course_Title	Salary
100	Ali	Marketing	c++spss	48000
100	Ali	marketing	surveys	48000
140	Khalid	Infosystem	c++	50000
150	ahemed	Infosystem	Java	80000

New table

Emp-ID	Ename	Dept-Name	course_Title	Salary
100	Ali	Marketing	NULL	90000

This is an insert anamoly.

Here, insert anamoly occours because, in the new record Emp-ID is available but course_Title is NULL. As per the entity integrity rule primary key must not have Null value.

ii) Delete Anomaly :-

- A deletion anamolies is the unintended loss of data due to deletion of other data.
- A deletion anomaly occurs when we delete a record that may contain attributes that shouldn't be deleted.
- A record of data can legitimately be deleted from a database and the deletion can result in the deletion of the only instance of other, required data.

eg:-

EmployeeKey	Employee Name	Project Name	Project Description
4123	Brown Richard	AI	Bulding the low cost robert
4142	Anu Rokka	Datascience	Tune the point of data to any scale.
4215	Thomas Basyal	Machinelearning	To develope new Algorithm for AI

- If Anu Rokka were the only person on project Datascience, deleting her from database would also delete the project information.

iii) Modification Anomaly (updated anomalies)

- Incorrect data, may have to change, which could involve many records having to be changed, leading to the possibility of some changes being made incorrectly.
- When duplicate data is updated at one instance and not across all instances where it was duplicated is called update anomaly.
- An update anomaly is a data inconsistency that results from data redundancy and a partial update.
- It happens when the person charged with the task of keeping all the records current and accurate is asked. for eg:- To change the emplmt + due to Promotion.

eg:- consider a relation want to update (CEIT) department Manager.

E#	Ename	Address	D#	Dname	Dmgr#
1	Jay	india	1	CEIT	M1
2	samantha	india	2	IT	M2
3	Thomas	America	2	CEIT	M2

Suppose the manager of a (CEIT) department has changed, this requires that the Dmgr# in all the tuples corresponding to that department must be changed to reflect the new status.

If we fail to update all the tuples of given department, then two different records of employee working in the same department might show different Dmgr# lead to inconsistency in the database.

2017 spring

3b) Design relational database for the Dept. of computer engineering (DoCE) at pokhara university. your database should have at least (3) relations. Describe referential integrity constraint based on the above database of DOCE.

2018 Fall 3b repeated

4a) What are the roles of Assertions and Triggers in SQL? consider following bank database:

Branch-schema = (branch-name, branch-city, assets)
loan-schema = (loan-number, branch-name, amount)

write an Assertions for the bank database to insure that the Assets value for the Koteshwor branch is equal to the sum of all the amounts lent by the Koteshwor branch.

Ans:-

```
CREATE Assertion ensure CHECK
    (not exists (select * from Branch-schema where
     branch_name = 'Koteshwor' and
     assets = (select sum(amount) from
     loan-schema where branch_name = Koteshwor')))
```

2018 spring

3b) Write down the properties of decomposition. (Repeated) (In Note copy)

2019 Fall

3b) Suppose we are given schema $R = \{A, B, C, G, H, I\}$ and set of functional dependencies $F = \{A \to B, A \to C, CG \to H, B \to H, CG \to I\}$. Find the closures of functional dependency F.
also computing $(AG)^+$

Ans:-

Soln

Let result = $\{AG\}$

1st iteration

For A→B
A ⊆ {AG} ; True so,
∴ result = {ABG}

For A→C
A ⊆ {ABG} ; True so,
∴ result = {ABCG}

For CG→H
CG ⊆ {ABCG} ; True so,
∴ result = {ABCGH}

For CG→I
CG ⊆ {ABCGH} ; True so,
∴ result = {ABCGHI}

For B→H
B ⊆ {ABCGHI} ; True so,
∴ result = {ABCGHI}

2nd iterations
For A→B
A ⊆ {ABCGHI} true so,
∴ result = {ABCGHI}

Similiarly for other result = {ABCGHI} The result does not change further
∴ result = {ABCGHI}

Now, Closures (F^+) of a Functional dependency.

i) $A \to B$, $A \to C$ then $A \to BC$ {Union Rule}
ii) $A \to B$, $B \to H$ then $A \to H$ {Transitivity Rule}
iii) $CG \to H$, $CG \to I$ then, $CG \to HI$ {union Rule}
iv) $A \to C$, $CG \to I$ then, $AG \to I$ {pseudo Transitivity Rule}

2020 Fall all repeated

2019 spring

3b) Explain Closure set of functional dependency with example. Write properties of decomposition. write one example of Multivalued dependency. How will you make the given table Std master with attributes: St_id, st_name, instructor_id, Inst_name, course_id1, course_name1, course_id2, Course_name2, Courseid3, Course_name3, in 1st, 2nd and 3rd normal form. write the steps.

Closure set of functional dependency.

- The set of functional dependencies that is logically implied by F is called closure of F and is written as F^+.
- The closure of F denoted by F^+ is the set of all Functional dependencies entailed by F.
- $F^+ = \{ x \to y \mid F \models X \to Y \}$ (pie)
- If F is a set of Functional dependencies, we can compute F^+ directly from formal defination of function dependency

- If F were large, this process be lengthy and difficult so Armstrong axioms useful for fast and accurate result.

eg:- $R = \{A, B, C, D\}$

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

$F^+ = ?$

soln

$A \to B$, $A \to C$ then $A \to BC$ {union Rule}

$A \to B$, $BC \to D$ then, $AC \to D$ {pseudo Transitive Rule}

$A \to C$, $BC \to D$ then, $AB \to D$ {pseudo Transitivive Rule}

Closure of attribute set

- Closure of attribute set is the set of attributes which are functionally dependent on the attribute set.

- Given a set of attributes $A_1 \ldots A_n$ the closure of $\{A_1 \ldots A_n\}$ is the set of attributes B such that $A_1 \cdots A_n \to B$.

eg:-

name → color
category → department
color, category → price

Now,

$\{name\}^+ = \{name, color\}$

$\{color\}^+ = \{color\}$

$\{name, category\}^+ = \{name, category, color, department, \ldots\}$

Properties of decomposition.

a) Attribute prevention :-

- preserving all the attributes of relation being decomposed.

b) Dependency prevention

- Let F be the dependencies on a relation R which is decomposed in $R_1, R_2 \ldots R_n$.

- we can partition the dependencies given by F such that $F_1, F_2, \ldots F_n$ are dependencies that only involve attribute from relation $R_1, R_2 \ldots R_n$ respectively.

- If the union of dependencies imply all the dependencies in F, then we say decomposition has preserved dependencies otherwise not.

- If the decomposition don't preserve the dependencies, F then the decomposed relation may contain relation that do not satisfy F.

c) Lack of Redundancy

- Redundancy should be avoided as much as possible.

d) lossy decomposition

- Loss of information due to decomposition is called as lossy decomposition.

e) Non-loss decomposition

- It refers to decomposition where all information is preserved.

Example of Multivalued dependencies

Name	SSN	PhoneNO	Course
Fred	123-321-99	986501	CSE-444
Fred	123-321-99	986501	CSE-391
Fred	123-321-99	982132	CSE-444
Fred	123-321-99	982132	CSE-391

The multivalued dependencies are

Name, SSN $\twoheadrightarrow$ PhoneNO

Name, SSN $\twoheadrightarrow$ Course

Stdmaster (St_id, St_name, instructor_id, inst_name, course_id,1, course_name1, course_id2, course_name2, course_id3, course_name3)

- If the relation is in unNormalized form ie it contains one or more repeating groups or each row may contain multiple set of values for some columns.

- To take the table into 1NF following criteria must be satisfied:
  - If there is no repeating values in a columns.
  - It must not have repeating columns

- Here course_id1, course_id2, course_id3 are repeating columns
  Changing it to only course_id and
  Similarly, course_name1, course_name2, course_name3 into course_name only then,

the 1NF table relation schema will be :-

Stdmaster (St_id, St_name, instructor_id, inst_name, course_id, course_name)

Now this is in 1NF

To change this relation schema in 2NF. it must have certain criteria,
- it is in 1NF and
- all attributes fully depend on primary key.

The above relation is in 1NF and
all attributes are not fully dependent on primary key.

Now, breaking the table in such a way and preserving its dependency & no loss of information

The relation schema with PK St_id, instructor_id, course_id.

Now,

P1 (st_id, st_name)

P2 (instructor_id, inst_name)

P3 (course_id, course_name)

P4 (id, st_id, instructor_id, course_id)

Here Now all table is in 2NF and 3NF also because Relation schema P1, P2, P3, P4 all satisfy 3NF criteria i.e. It is in 2NF and there is no any non-prime attribute that depends on another non-prime attribute (no transitive dependency)

Example of Multivalued dependencies

Name	SSN	Phone NO	course
Fred	123-321-99	986501	CSE-444
Fred	123-321-99	986501	CSE-341
Fred	123-321-99	982132	CSE-444
Fred	123-321-99	982132	CSE-341

The multivalued dependencies are

Name, SSN →→ Phone NO

Name, SSN →→ course

Stdmaster (st_id, st_name, instructor_id, inst_name, course_id,1, course_name1, course_id2, course_name2, course_id3, course_name3)

- If the relation is in unNormalized form ie it contains one or more repeating groups or each row may contain multiple set of values for some columns.

- To take the table into 1NF following criteria must be satisfied:
  - If there is no repeating values in a columns.
  - It must not have repeating columns

- Here course_id1, course_id2, course_id3 are repeating columns
  Changing it to only course_id and
  Similarly, course_name1, course_name2, course_name3 into course_name only then,

the 1NF table relation schema will be :-

Stdmaster (st_id, st_name, instructor_id, inst_name, course_id, course_name)

Now this is in 1NF

- To change this relation schema in 2NF. it must have certain criteria,
  - it is in 1NF and
  - all attributes fully depend on primary key.

- The above relation is in 1NF and all attributes are not fully dependent on primary key.

- Now, breaking the table in such a way and preserving its dependency & no loss of information

The relation schema with PK st_id, instructor_id, course_id.

Now,

P1 (st_id, st_name)

P2 (instructor_id, inst_name)

P3 (course_id, course_name)

P4 (id, st_id, instructor_id, course_id)

Here Now all table is in 2NF and 3NF also because Relation schema P1, P2, P3, P4 all satisfy 3NF criteria ie. It is in 2NF and there is no any non-prime attribute that depends on another non-prime attribute (no transitive dependency)

Example of Multivalued dependencies

Name	SSN	phone NO	course
Fred	123-321-99	98650	CSE-444
Fred	123-321-99	98650	CSE-241
Fred	123-321-99	98232	CSE-444
Fred	123-321-99	98232	CSE-241

The multivalued dependencies are

Name, SSN → PhoneNO

Name, SSN → course

Stdmaster (st_id, st_name, instructor_id, inst_name, course_id1, course_name1, course_id2, course_name2, course_id3, course_name3)

- If the relation is in unNormalized form ie it contains one or more repeating groups or each row may contain multiple set of values for some columns.

- To take the table into 1NF following criteria must be satisfied:
  - If there is no repeating values in a columns.
  - It must not have repeating columns

- Here course_id1, course_id2, course_id3 are repeating columns
  Changing it to only course_id and
  Similarly, course_name1, course_name2, course_name3 into course_name only then,

The 1NF table relation schema will be :-

Stdmaster (st_id, st_name, instructor_id, inst_name, course_id, course_name)

Now this is in 1NF

To change this relation schema in 2NF. It must have certain criteria,
- it is in 1NF and
- all attributes fully depend on primary key

The above relation is in 1NF and all attributes are not fully dependent on primary key.

Now, breaking the table in such a way and preserving its dependency & no loss of information

The relation schema with PK st_id, instructor_id, course_id.

Now,

P1 (st_id, st_name)

P2 (instructor_id, inst_name)

P3 (course_id, course_name)

P4 (id, st_id, instructor_id, course_id)

Here Now all table is in 2NF and 3NF also because Relation schema P1, P2, P3, P4 all satisfy 3NF criteria ie. It is in 2NF and there is no any non-prime attribute that depends on another non-prime attribute (no transitive dependency)