## **Function**

Create Function <function-name> (new-col-name DataType)

Returns DataType

Begin

Declare new-col-name1 DataType1

Declare new-col-name2 DataType2

Select col-name **Into** new-col-name1

From TableName

IF condition on new-col-name1Then

Set new-col-name2 = ….

ELSEIF condition on new-col-name1Then

Set new-col-name2 = ….

ELSE

Set new-col-name2 = ….

END if

Return new-col-name2

End

Select function-name(col-name)

From table-name

## **Procedure**

Create Procedure <procedure-name> (IN new-col-name1 DataType,

OUT new-col-name2 DataType)

Begin

Update <tablename>

Set col-name = (Given Condition)

Where condition on IN col

Select sum(col-name) **into** new col-name2

From tablename

Where condition on IN col

End

Declare new-col-name3 DataType

Call <procedure-name>(value, new-col-name3)

## **User Management**

Individual Privilage 🡪 GRANT Insert, Update, Delete ON <Tablename/view-name> TO <User1, User2>

Create Roles 🡪 Create Role <rolename>

Grant Roles to users 🡪 GRANT <rolename> TO <User1, User2>

Withdraw Individual privilages 🡪 REVOKE Insert, Update, Delete ON <tablename> FROM <U1, U2>

Witdraw Role 🡪 REVOKE <rolename> FROM <U1>

GRANT <rolename> TO <U2>

## **View**

Create View <view-name> as (SQL Queries)

## **Integrity Constraints**

Composite Primary Key 🡪 PRIMARY KEY(col-1, col-2)

Foreign Key (new-col-name) References <tablename>(col-name)

CHECK (P)

P=Predicate

CHECK (city IN (‘Dhaka’, ‘Rajshahi’, ‘Sylhet’))

CHECK (cgpa BETWEEN 0 AND 4)

## **Normalization**

1NF: No Multivalued Column

2NF: No Partial Dependency (Primary Key 🡪 Non Key)

3NF: No Transitive Dependency (Non Key 🡪 Non Key)

## **Index**

Create Index <indexname> ON <tablename> (col-name)

Clustered Index: Ordered Index and Ordered Disk

Dense Index: each an every col of Index-File point the related row

Sparse Index: Some Key is inserted in Index-file, points🡪 Blocks

Secondary Index: Index-file (non-key) points🡪Bucket 🡪 particular search key values

(Sparse+Dense = Dense)

Insertion (Dense) 🡪 Insert in Index-file + insert also in Datafile

Insertion (Sparse) 🡪 Search in Index file insert key + insert also in datatable

Deletation (Dense) 🡪 Delete from index file + delete from data-table

Deletion (Sparse) 🡪

if not in indextable 🡪No need to delete from Index-file + Delete from Data-file

if exists in index-file 🡪 Binary-Search to find largest key value < particular key + then delete the date from Data-table + In index file replace the address vaule with immidiate next.

SELECT \* FROM instructor WHERE id = 83821 (DENSE)

Step-1: Find Physical Address(pointer) of id = 83821 by searching index-file

Step-2: By using the physical address(pointer) find the data from disk, then the row data fetched to the memory

SELECT \* FROM instructor WHERE id = 83821 (Sparse)

Step-1: Find largest id of Physical Address(pointer) condition id < 83821 by searching index-file

Step-2: By using the physical address(pointer) and linear searchig in the disk, then the row data fetched to the memory

## **SQL Operations**

Insertion 🡪 Insert <Tablename> (col-1, col-2)

Select col-1, col-2

From tablename

Deletaion 🡪 Delete From <tablename>

Where Condition on Column

Update 🡪 Update <tablename>

Set col-name = Condition/SQL

## **SQL Join**

Wise 🡪 Group By

(course-id)

All students who have taken any course taken by Abid 🡪 IN

All students who have taken all courses taken by Abid 🡪 Not Exists

(branch\_name)

### All customers who have accounts in all branches in comilla city 🡪 Not Exists

all customers who have both loan and account. Join

all customers who have account but no loan.

Customer INNER JOIN with Account

Customer Left Outer Join with Loan, where loan\_numeber IS NULL;

all employees whether he/she have child or not. Employee Left Outer Join Child

all projects with employee or no employee and all employees work to project or not.

Full Outer Join