|  |  |  |
| --- | --- | --- |
| **Database System(CSE 235)**  **Lab 04** | | |
| **CSE 235(02)** | | |
| **Database System** | | |
| DATE | 22.11.26 | |
| NAME | 2018136121 | 조원석 |
| 2018136111 | 장영진 |
| 2019136154 | 이제윤 |
| 2018136003 | 강희권 |
|  | | |

**INDEX**

* **Tasks-1 : develop Java Application for your Database**
  + Establish a connection between your data base and Java Application using JDBC driver
  + Create an appropriate GUI using Java language
    - display information about your database (tables, functions, stored procedures etc)
    - navigate (see) records (data) from each table in your database
    - run functions to insert, update and delete records for each table in your database
    - print reports (data from different tables)
* **Tasks-2: Submit project report** 
  + Introduction
  + Database Design
  + Server side Programming (Procedure, functions, triggers etc)
  + Client side Programming (Java Application)
* **Tasks-3: Prepare a video presentation for the project (5-7 minuets)**

**Tasks-1 : develop Java Application for your Database**

|  |
| --- |
| **1.1 Adding User Library** |
| Build Path -> Add Libraries -> User Library -> Add External JARs… |
|  |
|  |
|  |
|  |

1. **Establish a connection between your database and Java Application using JDBC driver**

|  |
| --- |
| **1.2 Make connection** |
| **Login set** |
|  |
| MyConnection.java |
| public static Connection makeConnection() {  . . .  // load and register the Driver  // using "encrypt=false" to avoid ssh authorization  Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  String url = "jdbc:sqlserver://DESKTOP-OPBS2NM:1433;databaseName=RewardCrowdFunding;**encrypt=false**";  // Get connection using URL through Driver Manager  con = DriverManager.getConnection(**url, "iqeq126", "1320"**);  . . . |
| DBGUI.java / Metadata.java |
| public class DBGUI extends Application or public class Metadata {  . . .  private Connection con = **MyConnection.makeConnection();**  . . . |
| **1.3 Result** |
|  |

1. **Create an appropriate GUI using Java language**
   1. display information about your database (tables, functions, stored procedures etc)

**[Press <ctrl> & <click>]**

[**..\Video\Task1.2.A.mp4**](../Video/Task1.2.A.mp4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

* 1. navigate (see) records (data) from each table in your database

**[Press <ctrl> & <click>]**

[**..\Video\Task1.2.B.mp4**](../Video/Task1.2.B.mp4)

* 1. run functions to insert, update and delete records for each table in your database

**[Press <ctrl> & <click>], [Sample\_Channel\_Table]**

[**..\Video\Task1.2.C.mp4**](../Video/Task1.2.C.mp4)

* 1. print reports (data from different tables)

**[Press <ctrl> & <click>]**

[**..\Video\Task1.2.D.mp4**](../Video/Task1.2.D.mp4)

**Tasks-2: Submit project report**

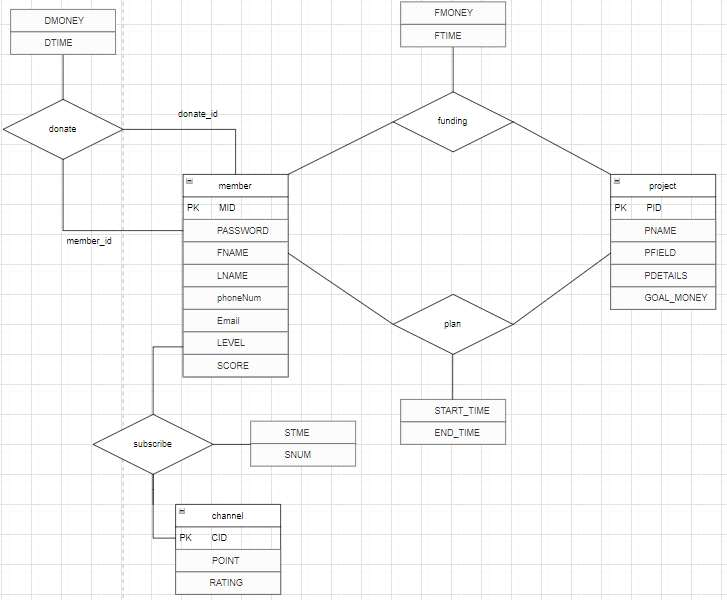
1. **Introduction**

We took a class on the database. We took the following class that Database Design, Server side Programming, and Client side Programming. And we created our own projects for each assignment.

We came up with a crowdfunding database for a specific project. This database is about being donated by the people working on the project.

So, Let me introduce our own Database.

1. **Database Design**

Introduce our database design. Our database’s ER Diagram. 

List of entity and attribute

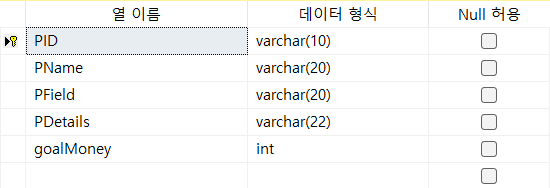
We proceeded with Logical Design with this ER diagram.

Explained entity and attribute of the database that proceeded to Logical Design.

* Project

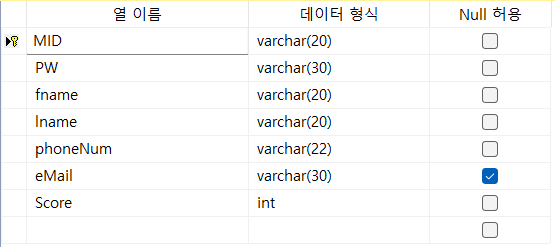
project that will receive crowdfunding. It has PID, PNAME, PField, PDetails, goalMoney.

PID is primary key.



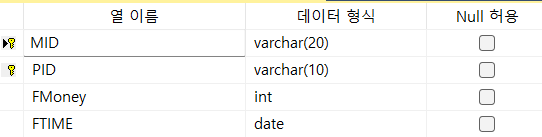
* Member

Member is the object of funding. It has member id, password, phoneNumber, eMail, and funding Score. MID is primary key.



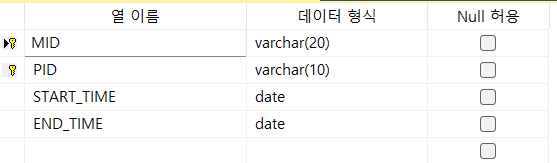
* Funding

Related to funding. Receives the MID, PID as a foreign key. It has attribute of a funding money, funding time. MID and PID is primary key.



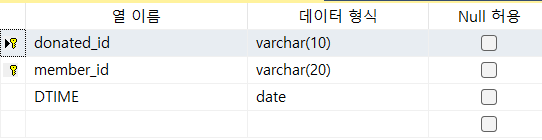
* Plan\_

Plan is funding with members. Receives the MID, PID as a foreign key. It has attribute of a start time, end time. MID and PID is primary key.



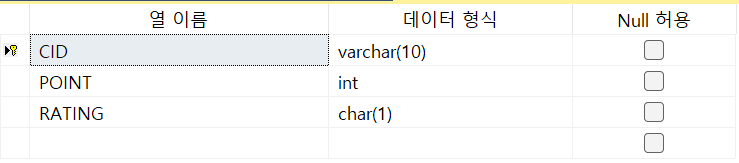
* Donate

Donate is an object related to donation. Receives the member\_id as a foreign key. It has attribute of donate id, donate time. Donated\_id is primary key.



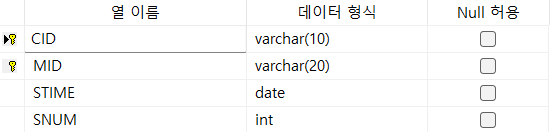
* Channel

Channel is platform with subscribe (like youtube channel). Point is the number of human subscribed to the channel. Rating is the rank based on the number of Point. Channel Id is primary key.



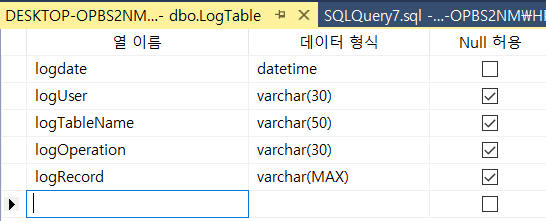
* Subscribe

Subscribe is entity which channel a particular member subscribed to. CID, MID is foreign key.



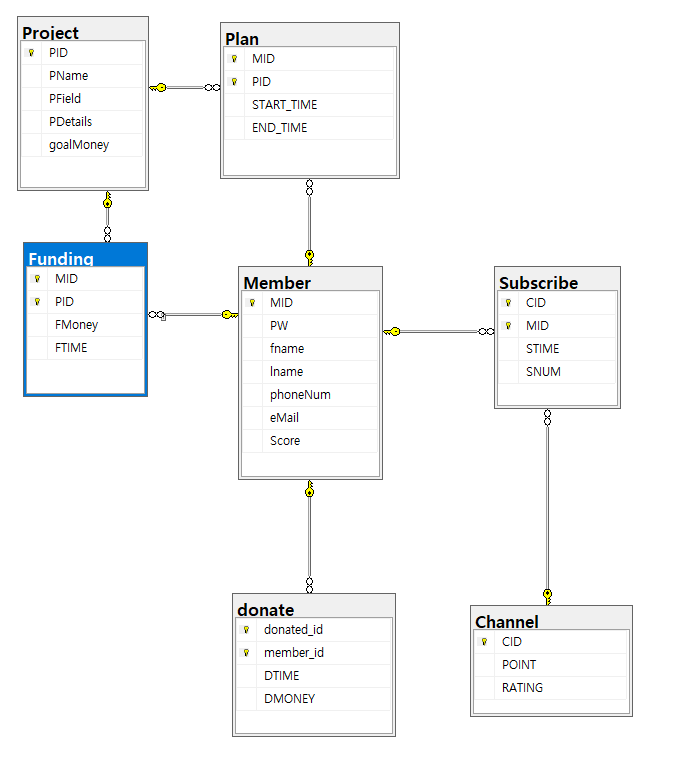
* LogTable

Logdata has logdata, logUser, logTableName, logPeration, logRecord



* Full Table

below is the overall table diagram and connection relationship



* 1. **Table Create Quary**

|  |
| --- |
| Member |
| CREATE TABLE Member(  MID VARCHAR(20) NOT NULL, -- NOT NULL Constraint  PW VARCHAR(30) NOT NULL, -- NOT NULL Constraint  fname VARCHAR(20) NOT NULL, -- NOT NULL Constraint  lname VARCHAR(20) NOT NULL, -- NOT NULL Constraint  phoneNum VARCHAR(22) NOT NULL, -- NOT NULL Constraint  eMail VARCHAR(30) NULL,  Score INT NOT NULL , -- NOT NULL Constraint  PRIMARY KEY (MID), -- PRIMARY Key Constraint  -- CHECK (branchNo like 'B%') -- CHECK Constraint  );  go |
| Project |
| CREATE TABLE Project(  PID VARCHAR(10) NOT NULL, -- NOT NULL Constraint  PName VARCHAR(20) NOT NULL, -- NOT NULL Constraint  PField VARCHAR(20) NOT NULL, -- NOT NULL Constraint  PDetails VARCHAR(22) NOT NULL, -- NOT NULL Constraint  goalMoney INT NOT NULL, -- NOT NULL Constraint  PRIMARY KEY (PID), -- PRIMARY Key Constraint  CHECK (goalMoney >= 100000)  );  go |
| Funding |
| CREATE TABLE Funding(  MID VARCHAR(20) NOT NULL, -- NOT NULL Constraint  PID VARCHAR(10) NOT NULL, -- NOT NULL Constraint  FMoney INT NOT NULL, -- NOT NULL Constraint  FTIME DATE DEFAULT GETDATE() NOT NULL, -- DEFAULT, NOT NULL Constraint  PRIMARY KEY (MID, PID), -- PRIMARY Key Constraint  foreign key (MID) REFERENCES Member(MID), -- FORENIGN Key Constraint  foreign key (PID) REFERENCES Project(PID) -- FORENIGN Key Constraint  );  go |
| Plan\_ |
| CREATE TABLE Plan\_(  MID VARCHAR(20) NOT NULL,  PID VARCHAR(10) NOT NULL,  START\_TIME DATE NOT NULL,  END\_TIME DATE NOT NULL,  PRIMARY KEY(MID,PID),  foreign key (MID) REFERENCES Member(MID), -- FORENIGN Key Constraint  foreign key (PID) REFERENCES Project(PID), -- FORENIGN Key Constraint  CHECK ( END\_TIME > START\_TIME),  CHECK ( START\_TIME > GETDATE())  );  go |
| Channel |
| CREATE TABLE Channel(  CID VARCHAR(10) NOT NULL PRIMARY KEY,  POINT INT NOT NULL DEFAULT 0,  RATING CHAR(1) NOT NULL DEFAULT 'C' CHECK(RATING IN('S', 'A', 'B', 'C'))  );  go |
| Subscribe |
| CREATE TABLE Subscribe(  CID VARCHAR(10) NOT NULL,  MID VARCHAR(20) NOT NULL,  STIME DATE DEFAULT GETDATE() NOT NULL,  SNUM INT NOT NULL,  PRIMARY KEY(CID, MID),-- primary key constraint  foreign key (CID) REFERENCES Channel(CID), -- foreign key constraint  foreign key (MID) REFERENCES Member (MID), -- foreign key constraint  CHECK (SNUM >= 0),  );  go  donate |
| CREATE TABLE donate(  donated\_id VARCHAR(10) NOT NULL,  member\_id VARCHAR(20) NOT NULL,  DTIME DATE DEFAULT GETDATE() NOT NULL,  PRIMARY KEY (donated\_id, member\_id), -- primary key constraint  FOREIGN KEY (member\_id) REFERENCES Member(MID) -- foreign key constraint  );  go |
| LogTable : update / delete Trigger’s Log store |
| CREATE TABLE LogTable(  logdate datetime NOT NULL,  logUser varchar(30) NULL,  logTableName varchar(50) NULL,  logOperation varchar(30) NULL,  logRecord varchar(max) NULL  )  go |

1. **Server side Programming**
   1. **Procedure**

|  |
| --- |
| Explanation : This procedures is the procedure responsible for deletion at each table. |
| create proc sp\_deleteChannel @c1 varchar(10)    as  begin  delete from dbo.Channel  where CID = @c1;  end    create proc sp\_deletedonate @c1 int, @c2 varchar(20)    as  begin  delete from dbo.donate  where donated\_id = @c1 AND member\_id = @c2;  end    create proc sp\_deletedFunding @c1 varchar(20), @c2 varchar(20)    as  begin  delete from dbo.Funding  where MID = @c1 AND PID = @c2;  end    create proc sp\_deletedMember @c1 varchar(20)    as  begin  delete from dbo.Member  where MID = @c1;  end    create proc sp\_deletedPlan\_ @c1 varchar(20), @c2 varchar(20)    as  begin  delete from dbo.Plan\_  where MID = @c1 AND PID = @c2;  end    create proc sp\_deletedProject @c1 varchar(20)    as  begin  delete from dbo.Project  where PID = @c1;  end    create proc sp\_deletedSubsribe @c1 varchar(10), @c2 varchar(20)    as  begin  delete from dbo.Subscribe  where CID = @c1 AND MID = @c2;  end |
| Explanation : These procedures are procedures responsible for inserting data into each table. |
| create proc sp\_insertdonate  @c2 varchar(20),  @c3 date,  @c4 int  as  begin  insert into dbo.donate values (@c2,@c3,@c4)  end    create proc sp\_insertFunding  @c1 varchar(20),  @c2 varchar(20),  @c3 int,  @c4 date  as  begin  insert into dbo.Funding values (@c1,@c2,@c3,@c4)  end    create proc sp\_insertMember  @c1 varchar(20),  @c2 varchar(30),  @c3 varchar(20),  @c4 varchar(20),  @c5 varchar(22),  @c6 varchar(30),  @c7 int  as  begin  insert into dbo.Member values (@c1,@c2,@c3,@c4, @c5, @c6, @c7)  end    create proc sp\_insertPlan\_  @c1 varchar(20),  @c2 varchar(20),  @c3 date,  @c4 date  as  begin  insert into dbo.Plan\_ values (@c1,@c2,@c3,@c4)  end    create proc sp\_insertProject  @c1 varchar(20),  @c2 varchar(20),  @c3 varchar(20),  @c4 varchar(50),  @c5 int  as  begin  insert into dbo.Project values (@c1,@c2,@c3,@c4,@c5)  end    create proc sp\_insertSubscribe  @c1 varchar(10),  @c2 varchar(20),  @c3 date,  @c4 int  as  begin  insert into dbo.Subscribe values (@c1,@c2,@c3,@c4)  end |
| Explanation : These procedures serve to update data in the table. |
| use RewardCrowdfunding  go    create proc sp\_updateChannel  @c1 varchar(10),  @c2 int,  @c3 char(1)  as  begin  update dbo.Channel set POINT = @c2, RATING = @c3  where CID = @c1;  end    create proc sp\_updatedonate  @c1 int,  @c2 varchar(20),  @c3 date,  @c4 int  as  begin  update dbo.donate set DTIME = @c3, DMONEY = @c4  where donated\_id = @c1 AND member\_id = @c2;  end    create proc sp\_updateFunding  @c1 varchar(20),  @c2 varchar(20),  @c3 int,  @c4 date  as  begin  update dbo.Funding set FMoney = @c3, FTIME = @c4  where MID = @c1 AND PID = @c2;  end    create proc sp\_updateMember  @c1 varchar(20),  @c2 varchar(30),  @c3 varchar(20),  @c4 varchar(20),  @c5 varchar(22),  @c6 varchar(30),  @c7 int  as  begin  update dbo.Member set PW = @c2, fname = @c3, lname = @c4, phoneNum = @c5, eMail = @c6, Score = @c7  where MID = @c1;  end    create proc sp\_updatePlan\_  @c1 varchar(20),  @c2 varchar(20),  @c3 date,  @c4 date  as  begin  update dbo.Plan\_ set START\_TIME = @c3, END\_TIME = @c4  where MID = @c1 AND PID = @c2;  end    create proc sp\_updateProject  @c1 varchar(20),  @c2 varchar(20),  @c3 varchar(20),  @c4 varchar(50),  @c5 int  as  begin  update dbo.Project set PName = @c2, PField = @c3, PDetails = @c4, goalMoney = @c5  where PID = @c1  end    create proc sp\_updateSubscribe  @c1 varchar(10),  @c2 varchar(20),  @c3 date,  @c4 int  as  begin  update dbo.Subscribe set STIME = @c3, SNUM = @c4  where CID = @c1 AND MID = @c2;  end |

* 1. **Functions**

|  |
| --- |
| Explanation : This function returns a list of columns in a table. |
| create function [dbo].[getColumnsList](@tname as varchar(40))  returns table  as return  select SCHEMA\_NAME(schema\_id) AS schema\_name  ,o.name AS object\_name, o.type, o.type\_desc  ,c.name AS column\_name  ,TYPE\_NAME(c.user\_type\_id) AS column\_type  ,c.max\_length    FROM sys.objects AS o, sys.columns AS c    WHERE o.object\_id = c.object\_id and o.object\_id = OBJECT\_ID(@tname) |
|  |

|  |
| --- |
| Explanation : This function returns a Parameters in a table |
| create function [dbo].[getParameters](@tname as varchar(30))  returns table  as return  select SCHEMA\_NAME(schema\_id) AS schema\_name  ,o.name AS object\_name, o.type, o.type\_desc  ,p.name AS parameter\_name  ,TYPE\_NAME(p.user\_type\_id) AS parameter\_type  ,p.max\_length  ,p.precision    FROM sys.objects AS o, sys.columns AS p    WHERE o.object\_id = p.object\_id and o.object\_id = OBJECT\_ID(@tname) |
|  |

|  |
| --- |
| Explanation : This function returns a list of table |
| create function [dbo].[getListTables]()  returns table  as  return SELECT name AS object\_name, SCHEMA\_NAME(schema\_id) AS schema\_name  ,type, type\_desc, create\_date, modify\_date  FROM sys.objects  where type = 'U' |
|  |

|  |
| --- |
| Explanation : This function returns information from the generated table. |
| create function [dbo].[task1]()  returns table  return select \* from INFORMATION\_SCHEMA.TABLES |
|  |

|  |
| --- |
| Explanation : This function returns all Primary-keys and Foreign-Keys |
| create function [dbo].[task2]()  returns table  return select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS  where CONSTRAINT\_TYPE='PRIMARY KEY'or CONSTRAINT\_TYPE='foreign key' |
|  |

|  |
| --- |
| Explanation : This function returns all user-defined triggers |
| create function [dbo].[task3]()  returns table  return select \* from sys.objects  where type = 'TR' |
|  |

|  |
| --- |
| Explanation : This function returns row-count for all user-defined tables in the database |
| create function [dbo].[task4]()  returns table  return select \* from sys.objects  where type = 'U' |
|  |

|  |
| --- |
| Explanation : This function returns the list of all user defined functions. |
| create function [dbo].[task5]()  returns table  return select \* from sys.objects  where type = 'IF' or type ='FN' |
|  |

|  |
| --- |
| Explanation : This function takes a table name as input and returns the column names with their types |
| create function [dbo].[task6](@frname as varchar(30))    returns table  as return  select SCHEMA\_NAME(schema\_id) AS schema\_name,  o.name as object\_name, o.type,o.type\_desc  ,c.name as column\_name  ,TYPE\_NAME(c.user\_type\_id)as column\_type  ,c.max\_length  from sys.objects as o,sys.columns as c  where o.object\_id = c.object\_id and o.object\_id=OBJECT\_ID(@frname) |
|  |

|  |
| --- |
| Explanation :: This function takes a function name as input and returns input parameters for that function with their types |
| create function [dbo].[task7](@frname as varchar(30))  returns table  return select o.name as object\_name  ,o.type,o.type\_desc,p.name as parameter\_name,  TYPE\_NAME(p.user\_type\_id) as parameter\_type  ,p.max\_length,p.precision  from sys.objects as o,sys.parameters as p  where o.object\_id=p.object\_id and o.object\_id=OBJECT\_ID(@frname) |
|  |

|  |
| --- |
| Explanation : This function shows the three most expensive funding products. |
| create function [dbo].[UDF1]()    returns table  return select TOP(3) \* from Funding order by FMoney desc |
|  |

|  |
| --- |
| Explanation : This function outputs subscribers who subscribe to a particular channel. |
| create function [dbo].[UDF2](@myID varchar(10))  returns table  return select MID as '구독자', fname, lname from Member  where MID in (select MID from Subscribe where CID= @myID) |
|  |

|  |
| --- |
| Explanation : This function outputs the member ID that funding the project. |
| create function [dbo].[UDF3](@name varchar(40))  returns table  return select MID as '펀딩한 사람' from Funding  where PID = (select PID from Project where PName = @name) |
|  |

|  |
| --- |
| Explanation : This function calculates the sum of the funds for the entered project. |
| create function [dbo].[UDF4](@name varchar(40)) returns table  as  return select sum(FMoney) as '총 금액' from Funding  where PID in ( select PID from Project where PName=@name) |
|  |

|  |
| --- |
| Explanation : This function searches for members with n points or more. |
| create function [dbo].[UDF5](@num int)-  returns table  as  return select \* from Member  where Score >= @num |
|  |

* 1. **Trigger**

|  |
| --- |
| **3.3.1 Create Trigger**  **We created 4 triggers each for 7 tables, excluding logTable**  **We will explain the Channel table as an example..** |
| Explanation : This prints message that change table |
| CREATE trigger [dbo].[myTrigger]  on [dbo].[Channel]  after insert, delete, update  as  begin  print 'there is a change in Channel table '  end |
| Explanation : This is delete trigger |
| CREATE trigger [dbo].[tr\_deleteChannel]  on [dbo].[Channel]  for **delete**  as  begin  declare  @c1 datetime,  @c2 varchar(30),  @c3 varchar(50),  @c4 varchar(30),  @c5 varchar(max);  set @c1 = getdate();  set @c2 = SUSER\_NAME();  set @c3 = 'dbo.Channel';  set @c4 = **'delete'**;  select @c5 = CID + ', ' + convert(varchar, POINT) + ', ' + RATING from **deleted**;  insert into LogTable values(@c1, @c2, @c3, @c4, @c5);  end |
| Explanation : This is insert trigger |
| CREATE trigger [dbo].[tr\_insertChannel]  on [dbo].[Channel]  for **insert**  as  begin  declare  @c1 datetime,  @c2 varchar(30),  @c3 varchar(50),  @c4 varchar(30),  @c5 varchar(max);  set @c1 = getdate();  set @c2 = suser\_name();  set @c3 = 'dbo.Channel';  set @c4 = **'insert';**  select @c5 = CID + ', ' + convert(varchar, POINT) + ', ' + RATING from **inserted**;  insert into LogTable values(@c1, @c2, @c3, @c4, @c5);  end |
| Explanation : This is update trigger |
| ALTER trigger [dbo].[tr\_updateChannel]  on [dbo].[Channel] for **update**  as  begin  declare  @c1 datetime,  @c2 varchar(30),  @c3 varchar(50),  @c4 varchar(30),  @c5 varchar(max);  set @c1 = getdate();  set @c2 = suser\_name();  set @c3 = 'dbo.Channel';  set @c4 = **'update'**;  select @c5 = CID + ', ' + convert(varchar, POINT) + ', ' + RATING from **inserted**;  insert into LogTable values(@c1, @c2, @c3, @c4, @c5);  end |

1. **Client side Programming**

|  |
| --- |
| **4.1 Channel Table** |
|  |
| **Channel Table JassperViewer** |
|  |
| **4.2 donate Table** |
|  |
| **donate Table** **JasperViewer** |
|  |
| **4.3 Funding Table** |
|  |
| **Funding Table JasperViewer** |
|  |
| **4.4 Member Table** |
|  |
| **Member Table JasperViewer** |
|  |
| **4.5 Plan Table** |
|  |
| **Plan Table JasperViewer** |
|  |
| **4.6 Project Table** |
|  |
| **Project Table JasperViewer** |
|  |
| **4.7 Subscribe Table** |
|  |
| **Subscribe Table JasperViewer** |
|  |
| **4.8 LogData Table** |
|  |
| **LogData Table JasperViewer** |
|  |

**Tasks-3: Prepare a video presentation for the project (5-7 minuets)**

**[Press <ctrl> & <click>]**

[**..\Video\Task3.mp4**](../Video/Task3.mp4)