

Databases Homework 2 report

M Hasibur Rahman, 295459, 31/05/2020

Personal statement:

'I certify that this assignment is entirely my own work, performed independently and without any help from the sources which are not allowed.'

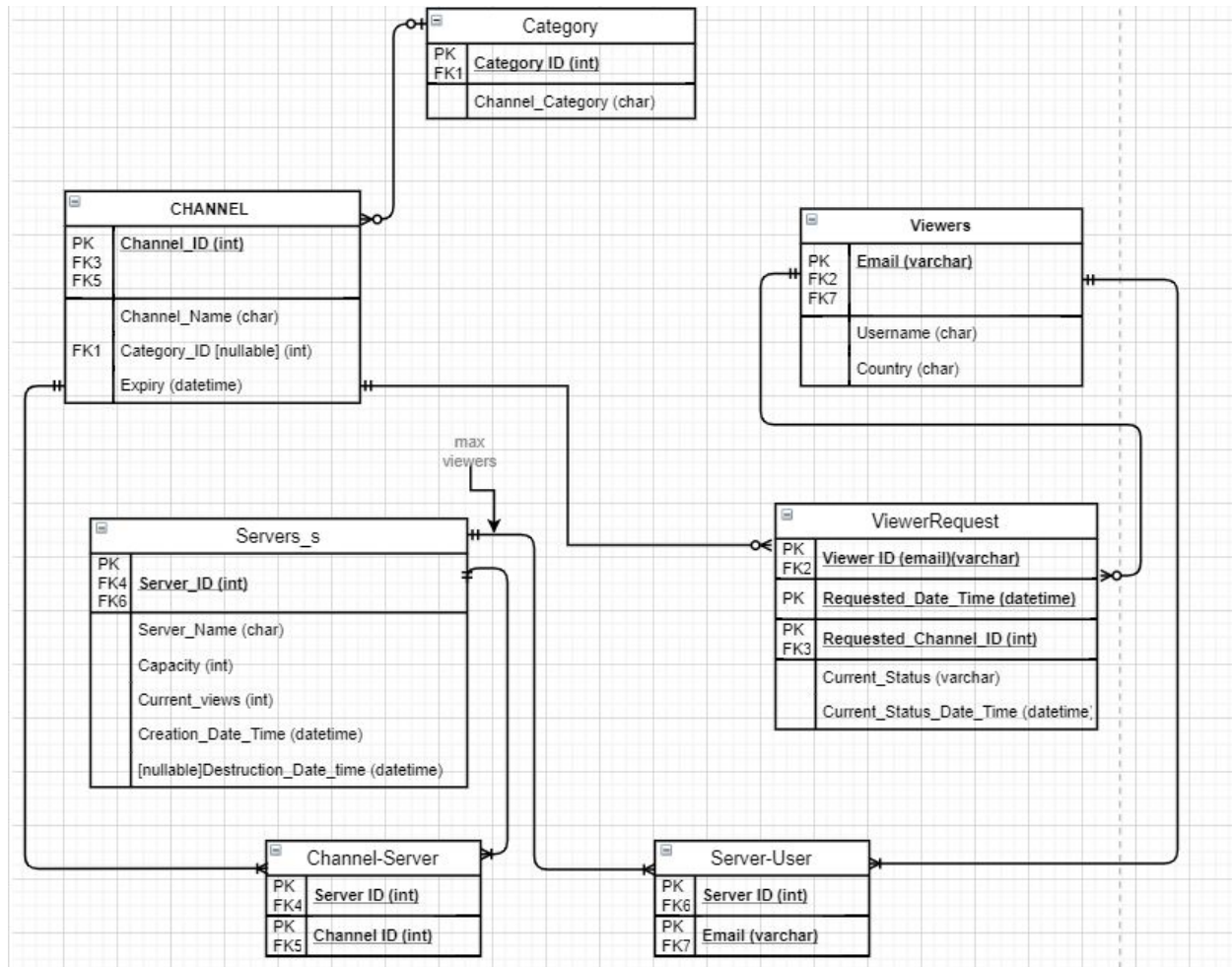
Signed: M Hasibur Rahman

Note:

Attached is "QueryForTask2" query file which has all the queries and more that are presented in this report

Part 1

Entity relation (ER) diagram



Part 2 Q.1and2

Above tables were made by the following sql queries:

```
--CREATE TABLE Category (  
--  Category_ID int NOT NULL PRIMARY KEY,  
--  Channel_Category char(30) NOT NULL,  
--);  
  
--CREATE TABLE Channel (  
--  Channel_ID int NOT NULL PRIMARY KEY,  
--  Channel_Name char(30) NOT NULL,  
--  Category_ID int, CONSTRAINT FK_1 FOREIGN KEY (Category_ID) REFERENCES  
Category(Category_ID),  
--  Expiry datetime NOT NULL
```

```
--);
```

```
--CREATE TABLE Viewers (  
-- Email varchar(50) NOT NULL PRIMARY KEY,  
-- Username char(30) NOT NULL,  
-- Country char(30) NOT NULL  
--);
```

```
--CREATE TABLE ViewerRequest (  
-- Viewer_ID varchar(50) NOT NULL, CONSTRAINT FK_2 FOREIGN KEY (Viewer_ID)  
REFERENCES Viewers(Email),  
-- Requested_Date_Time datetime NOT NULL,  
-- Requested_Channel_ID int NOT NULL CONSTRAINT FK_3 FOREIGN KEY  
(Requested_Channel_ID) REFERENCES Channel(Channel_ID),  
-- Current_Status varchar(50) NOT NULL,  
-- Current_Status_Date_Time datetime NOT NULL,  
-- primary key(Viewer_ID,Requested_Date_Time,Requested_Channel_ID)  
--);
```

```
--CREATE TABLE Server_s (  
-- Server_ID int NOT NULL PRIMARY KEY,  
-- Server_Name char(30) NOT NULL,  
-- Capacity int NOT NULL,  
-- Current_Views int NOT NULL,  
-- Creation_Date_Time datetime NOT NULL,  
-- Destruction_Date_Time datetime  
--);
```

```
--CREATE TABLE Server_User (  
-- Server_ID int NOT NULL, CONSTRAINT FK_6 FOREIGN KEY (Server_ID) REFERENCES  
Server_s(Server_ID),  
-- Email varchar(50) NOT NULL, CONSTRAINT FK_7 FOREIGN KEY (Email) REFERENCES  
Viewers(Email),  
-- primary key(Server_ID,Email)  
--);
```

```
--CREATE TABLE Channel_Server (  
-- Server_ID int NOT NULL, CONSTRAINT FK_4 FOREIGN KEY (Server_ID) REFERENCES  
Server_s(Server_ID),  
-- Channel_ID int NOT NULL, CONSTRAINT FK_5 FOREIGN KEY (Channel_ID)  
REFERENCES Channel(Channel_ID),  
-- primary key(Server_ID,Channel_ID));
```

Example of insertion for each table:

```
Insert into dbo.Channel(Channel_ID, Channel_Name, Category_ID, Expiry)
values (1,'HBO',10,'20240423 10:34:09 AM');
```

```
Insert into dbo.Category(Category_ID, Channel_Category)
values (10,'Comedy');
```

```
Insert into dbo.Viewers(Email, Username, Country)
values ('rhss@gmail.com','Rohans','India');
```

```
Insert into dbo.ViewerRequest(Viewer_ID, Requested_Date_Time, Requested_Channel_ID,
Current_Status, Current_Status_Date_Time)
values ('rhss@gmail.com','20210423 10:34:09 AM',2,'served','20210621 11:34:09 PM');
```

```
Insert into dbo.Server_User(Server_ID, Email)
values (41,'sven@gmail.com');
```

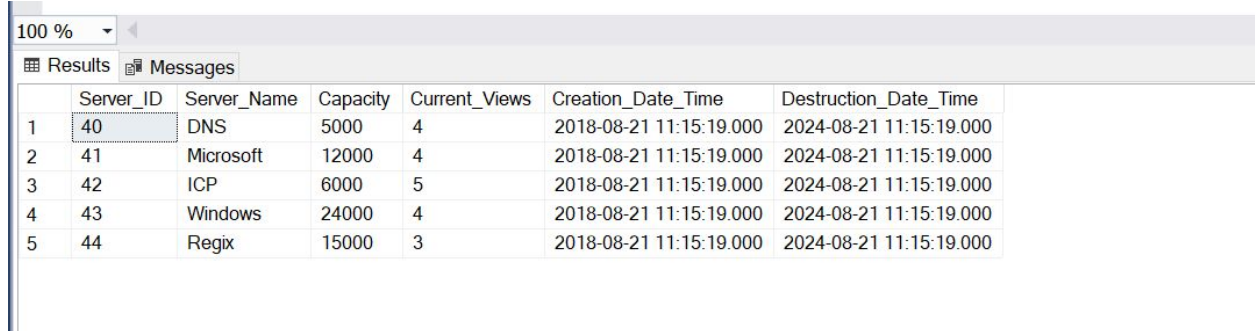
```
Insert into dbo.Channel_Server(Server_ID, Channel_ID)
values (40,2);
```

```
Insert into dbo.Server_s(Server_ID, Server_Name, Capacity, Current_VIEWS,
Creation_Date_Time, Destruction_Date_Time)
values (40,'DNS',5000,1543,'20180821 11:15:19 AM','20180821 11:15:19 AM');
```

Part 2. Q.3

```
update Server_s
set Current_VIEWS= (select count(*) from Server_User
where Server_User.Server_ID=Server_s.Server_ID);
```

This is the output we get showing after a new server user has been added.



	Server_ID	Server_Name	Capacity	Current_VIEWS	Creation_Date_Time	Destruction_Date_Time
1	40	DNS	5000	4	2018-08-21 11:15:19.000	2024-08-21 11:15:19.000
2	41	Microsoft	12000	4	2018-08-21 11:15:19.000	2024-08-21 11:15:19.000
3	42	ICP	6000	5	2018-08-21 11:15:19.000	2024-08-21 11:15:19.000
4	43	Windows	24000	4	2018-08-21 11:15:19.000	2024-08-21 11:15:19.000
5	44	Regix	15000	3	2018-08-21 11:15:19.000	2024-08-21 11:15:19.000

Part 3

--Design of Indexes

--First of all we can agree that Primary keys are already clustered indexes and that foreign keys that are not primary should be indexed as non-clustered indexes.

--We can create indexes on Channel_Server's Server_ID,Channel_ID and reorganize data by Server_ID then Channel_ID as it will enable us to check a Channel User faster and what Server it is watching and then the corresponding Channel_ID. This is unique.

--And so we create a non-clustered index of reflecting above information:

```
Create NONCLUSTERED Index ChannelUserServerID  
ON Channel_Server (Server_ID,Channel_ID)
```

--Similarly, with the same reason above for Server_User we

--Create an index on Server_User Server_ID,Email and reorganize data by Server_ID then Channel_ID which is non-clustered and is not unique

```
Create NONCLUSTERED Index ServerUserEmail  
ON Server_User (Server_ID,Email)
```

--We can create the below index since, Channel_Category is not that frequent and is not updated that frequently

```
Create NONCLUSTERED Index CatIDinChannel  
ON Channel (Category_ID)
```

--We can create an index on Category based on Category_ID to find channels based on that category faster.

--non-clustered including channel Category

```
Create NONCLUSTERED Index ChanCatinCat  
ON Category (Category_ID)  
INCLUDE (Channel_Category)
```

--We can index Viewers based on the email/viewer_id(which is also email) to query the viewers existence as a viewer and its requests

--non-clustered and unique

```
Create NONCLUSTERED Index Viewer_user_id  
ON Viewers (Email)
```

--non-clustered and not unique

```
Create NONCLUSTERED Index Viewer_user_id  
ON ViewerRequest (Viewer_ID)
```

--We can create index to check whether a server is about to be destructed or not

--non-clustured and unique

Create NONCLUSTERED Index serverdest

ON Server_s (Server_ID,Destruction_Date_Time)

--Similarly a index of viewer request and its status can be made

--non-clustured and unique

Create NONCLUSTERED Index viewerstat

ON ViewerRequest (Viewer_ID,Current_Status)

Part 4

Q.1)

select Viewers.Email, Channel.Channel_Name from Viewers, Channel,ViewerRequest where
(Viewers.Email=ViewerRequest.Viewer_ID and (Channel.Channel_ID not in
(select ViewerRequest.Requested_Channel_ID as Channel_ID from ViewerRequest where
ViewerRequest.Viewer_ID=Viewers.Email)))
group by Channel_Name,Viewers.Email

Email	Channel_Name
1 frk19@gmail.com	ABC
2 goji@gmail.com	ABC
3 rlen@gmail.com	ABC
4 sadf@gmail.com	ABC
5 sdae@gmail.com	ABC
6 sdes@gmail.com	ABC
7 sdf@gmail.com	ABC
8 simon_98@gmail.com	ABC
9 sven@gmail.com	ABC
10 rowa@gmail.com	CartoonNetwork
14 sven@gmail.com	CartoonNetwork
15 Vlen@gmail.com	CartoonNetwork
16 frk19@gmail.com	HBO
17 rlen@gmail.com	HBO
22 sjfie@gmail.com	HBO
23 Vlen@gmail.com	HBO
24 frk19@gmail.com	MTV
25 goji@gmail.com	MTV
26 rhss@gmail.com	MTV

It shows e.g. user: frk19@gmail.com has not requested channel ABC,MTV,HBO. And to prove that is true if we look at frk19@gmail.com we see that 2,5 are requested which are channels:CartoonNetwork and SevenNetwork. Shown below

Viewer_ID	Requested_Date_Time	Requested_Channel_ID	Current_Status	Current_Status_Date_Time
1 frk19@gmail.com	2021-04-23 10:34:09.000	2	served	2020-05-01 17:37:22.000
2 frk19@gmail.com	2021-04-23 10:34:09.000	5	open	2021-09-06 15:56:22.000
3 goji@gmail.com	2021-04-23 10:34:09.000	1	closed	2019-05-21 12:45:29.000
4 goji@gmail.com	2021-04-23 10:34:09.000	2	closed	2022-08-21 23:15:19.000
5 goji@gmail.com	2098-01-01 13:00:00.000	2	rejected	2098-01-01 13:00:00.000

Part 5

```
--create or alter procedure storproc @req datetime, @mail varchar(30)

--as

--begin

--    declare @id int;

--    declare @date datetime;

--    declare @i int = 0;


--    declare res cursor local for

--    select ch.Channel_ID, ch.Expiry from Channel ch

--    order by ch.Expiry ASC


--    open res


--    fetch next from res into @id, @date


--    while @@FETCH_STATUS=0

--    Begin

--        IF (@date > @req)

--            BEGIN

--                SET @i = 1

--                --- found

--                IF ((select s.Capacity  from Server_s as s

--                    where s.Server_ID IN (select Server_ID from

Channel_Server cs where cs.Channel_ID = @id))> (select s.Current_Views  from Server_s as s

--                    where s.Server_ID IN (select Server_ID from

Channel_Server cs where cs.Channel_ID = @id)))
```

```

--                                     BEGIN
--                                     PRINT 'KK';
--                                     END
--                                     ELSE
--                                     BEGIN
--                                     INSERT INTO Server_s values (5, 'ran', 5, 0,
getdate(), NULL);
--                                     INSERT INTO Channel_Server values (5, @id);
--                                     END
--                                     Break
--
--                                     END
--                                     ELSE
--                                     BEGIN
--                                     fetch next from res into @id,@date
--                                     continue
--                                     END
--                                     End

--                                     IF (@i = 0)
--                                     BEGIN
--                                     Insert into dbo.ViewerRequest(Viewer_ID, Requested_Date_Time,
Requested_Channel_ID, Current_Status, Current_Status_Date_Time)
--                                     values (@mail,@req,2,'rejected',@req);
--                                     RETURN
--                                     END
--                                     close res
--                                     deallocate res;

```


--end

The query is more visible in the query file attached with the mail.

This query basically Requests for viewing channels are only accepted if the requested date is less than the channel expire_date. If the request is rejected the procedure insert a new request of inserted date which is @req in our case and with status 'rejected'. I have also implemented code for the if statement for the statement: "If the request is accepted, then the procedure should check for a request's channel if there are free server resources for that channel by comparing max_viewers with current_viewers." But I was not able to solve part 5. Question 1. I did try to implement question 5.

To show the proof of when the request is not accepted e.g. when i ran the above code with a date greater than any of the channel expiry date I get the following:

The code I ran for storproc and the inputs:

```
exec storproc @req='2098-01-01 13:00:00', @mail = 'goji@gmail.com'
```

--goji@gmail.com is a viewer that already exists in viewer table and @req=date that is way bigger than any of the channel expiry date

I got the following:

Results Messages Client Statistics					
	Viewer_ID	Requested_Date_Time	Requested_Channel_ID	Current_Status	Current_Status_Date_Time
1	goji@gmail.com	2098-01-01 13:00:00.000	2	rejected	2098-01-01 13:00:00.000
2	goji@gmail.com	2021-04-23 10:34:09.000	2	closed	2022-08-21 23:15:19.000

As we can see the code worked as we got an insertion of a new request with status=rejected since the requested date is bigger than all channel expiry dates.