31 May 2020

Emilia Wróblewska
291674

Databases Task II Report

Problem Description

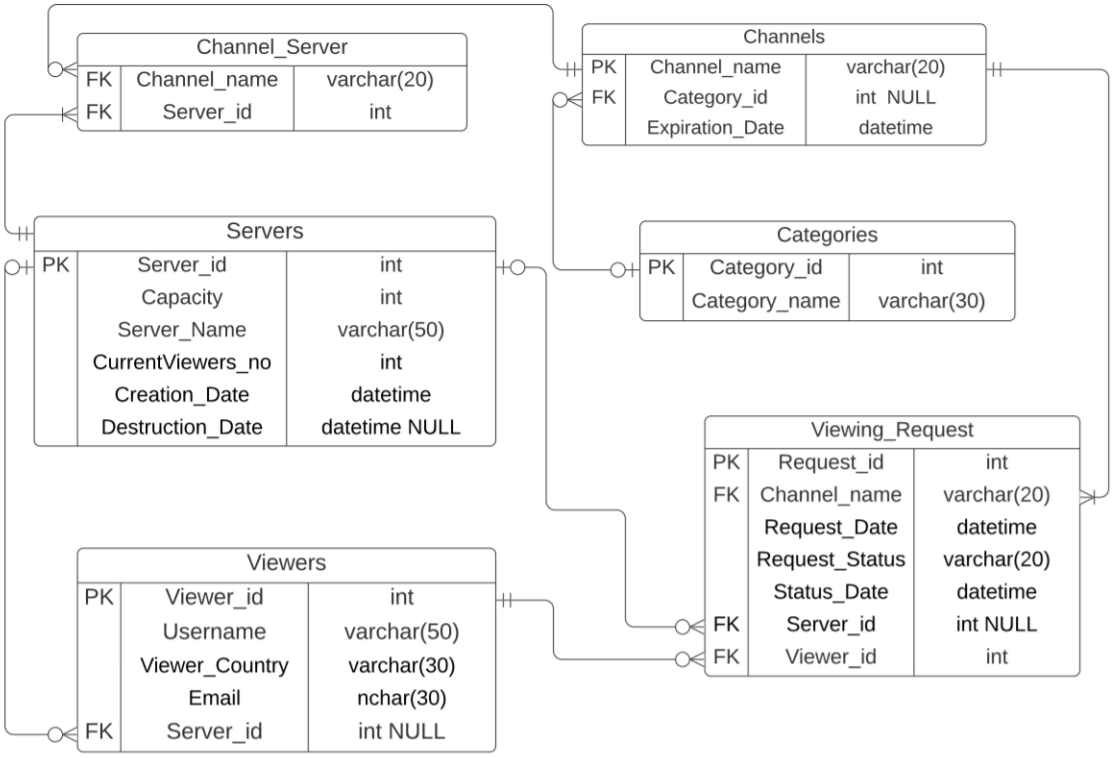
You work for the video streaming company. Your task is to design a small database for a system processing online tv viewing requests. Company offers multiple viewing channels. There are many channel categories (like sport, news, movies, music, kids, etcâ€¦) Each channel belongs to at most one category, some channels may be uncategorized. Each channel has an expiration datetime after which no view request is accepted by the system. Viewers are described by username, viewer country and an e-mail which must be unique. The Company has multiple virtual video streaming servers. Each server can stream multiple channels to multiple users, however, is has limited capacity which is maximum number of viewers connected to the server at the same time. Each server is described by its id, name, capacity, number of current viewers connected to this server and date and time of creation and date and time of destruction (active servers would have this empty). When a viewer attempts to view a channel a viewing request is created and stored in the database. The request has viewer id, request date and time, requested channel and current status (served â€“ currently being served, closed â€“ successfully served in the past, open â€“ awaiting for server assignment, rejected â€“ for some reasons i.e. missing payments) and a current status date and time. If there are servers currently serving the requested channel, with free capacity than one of them is assigned to the request and the request is considered served. Servers may be created and destroyed when required by streaming management system.

1 Part 1

My database *VideoStreaming* consists of 6 tables:

1. **Categories** - simple table with list of possible channels' categories names and IDs.
2. **Channels** - table with available channels each described by channel name, category (can be NULL) and expiration date after which channel cannot be streamed.
3. **Channel_Server** - mapping table of available channels and servers. It specifies which channels can be viewed on which servers.
4. **Servers** - table with servers available in the *VideoStreaming* company. Each server has it's unique ID, capacity (maximal number of viewers that can access this server at the same time), name, number of current viewers, creation and destruction dates. Destruction date is set to NULL for servers which are still working (were not destroyed).
5. **Viewers** - table of company's clients. Each viewer has their own unique ID, Username, country they are from, unique email and ID of currently assigned server. Server ID is set to NULL if the viewer currently is not watching anything, otherwise it's set to ID of server with requested channel.
6. **Viewing_Request** - table of requests made by viewers. One request consists of: request ID, requested channel name, request date, status, status date (date of most recent status update), viewer ID and the ID of server (can be NULL if no server is assigned).

The Entity Relation Diagram with indicated multiplicities of all relations between tables is presented below and included in file **Part1-Diagram.pdf**.



2 Part 2

2.1 Creation of tables with PK and FK specifications

Code for this section can be found file: **Part2.sql**. This query is responsible for creating the whole database and appropriate tables, inserting sample data to the tables and setting up all primary and foreign keys. The whole query should be executed at once. Below I present an exemplary query used for creation of *Channels* table. As we can see Primary key is set up during creation on unique fields that will be used to differentiate records.

```
01 | /****** Object:  Table [dbo].[Channels]  Date: 27.05.2020 ******/
02 | SET ANSI_NULLS ON
03 | GO
04 | SET QUOTED_IDENTIFIER ON
05 | GO
06 | CREATE TABLE [dbo].[Channels](
07 |         [Channel_name] [varchar](20) NOT NULL, --(PK)
08 |         [Category_id] [int] NULL,  --(FK)
09 |         [Expiration_Date] [datetime] NOT NULL,
10 |  CONSTRAINT [PK_Channels] PRIMARY KEY CLUSTERED
11 | (
12 |         [Channel_name] ASC
13 | )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
       = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
14 | ) ON [PRIMARY]
15 | GO
```

Foreign keys are created at the end, after data insertion. The exemplary query to set up foreign keys in *Channels* table is presented below.

```
01 | /***** Foreign keys for [dbo].[Channels] *****/
02 | GO
03 | ALTER TABLE [dbo].[Channels]  WITH CHECK ADD
04 | CONSTRAINT [FK_Channels_Category]
05 | FOREIGN KEY([Category_id]) REFERENCES [dbo].[Categories] ([
       Category_id])
06 | GO
07 | ALTER TABLE [dbo].[Channels] CHECK CONSTRAINT [FK_Channels_Category]
08 | GO
```

All created Forgein Keys have specific name that is build in the following way: FK_Tablename_ColumnFromOtherTable.

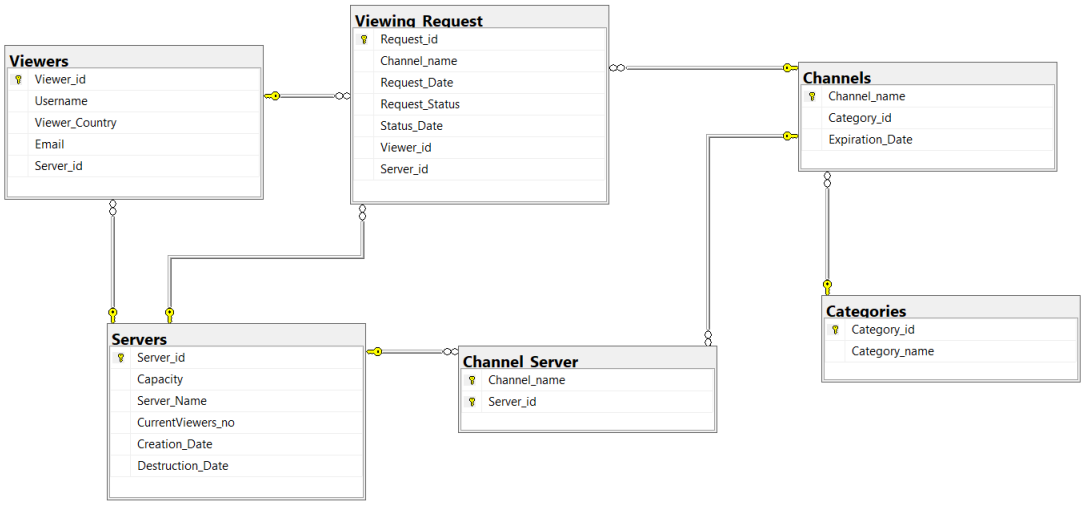


Diagram generated in SQL Server Managment Studio

2.2 Data insertion

This part is also included in file: **Part2.sql**. All tables have 6 to 7 records for easiness in searching. The only table with over 30 records is *Viewing_Request* table, since it was necessary for functional check of queries from Part 4. Initial tables are presented below:

	Category_id	Category_name
1	1	Sport
2	2	News
3	3	Kids
4	4	Movies
5	5	Music
6	6	Cooking

	Channel_name	Category_id	Expiration_Date
1	CALENDAR	NULL	2020-11-01 09:30:00.000
2	Eurosport	1	2020-10-01 10:30:00.000
3	Master Chef	6	2019-12-31 09:30:00.000
4	MusicTV	5	2020-12-31 09:30:00.000
5	NewsTV	2	2021-01-30 10:30:00.000
6	POP	NULL	2020-01-30 10:30:00.000

Categories and Channels

	Channel_name	Server_id
1	CALENDAR	101
2	Eurosport	101
3	MusicTV	101
4	NewsTV	101
5	CALENDAR	102
6	MusicTV	102
7	CALENDAR	103
8	NewsTV	103
9	POP	103
10	Master Chef	104
11	POP	104
12	CALENDAR	105
13	Eurosport	105
14	Master Chef	105
15	MusicTV	106
16	NewsTV	106

Channel_Server

	Server_id	Capacity	Server_Name	CurrentViewers_no	Creation_Date	Destruction_Date
1	101	4	server1	2	2016-10-30 13:20:00.000	NULL
2	102	3	Server2	1	2017-04-07 14:20:00.000	NULL
3	103	100	MainServer	0	2017-06-10 14:20:00.000	NULL
4	104	20	server4	0	2017-08-19 12:40:00.000	2019-01-01 14:20:13.000
5	105	7	server5	1	2018-10-10 13:20:00.000	NULL
6	106	42	Server6	0	2018-12-30 14:15:00.000	NULL

Servers

	Viewer_id	Username	Viewer_Country	Email	Server_id
1	11	JohnSmith	US	jsmith@us.com	NULL
2	12	flower55	Poland	12345@pl.pl	102
3	13	KLausP	Germany	KLAUS@ger...	105
4	14	JackSpa...	Poland	pirate@pl.pl	NULL
5	15	Atlantis	US	atlanta@us.c...	101
6	16	ALFKI	Poland	alf@xx.pl	NULL
7	17	misterX	Russia	sdfrusopen...	101

Viewers

	Request_id	Channel_name	Request_Date	Request_Status	Status_Date	Viewer_id	Server_id
1	1	POP	2019-01-01 23:09:22.000	closed	2019-01-02 01:01:04.000	11	105
2	2	Eurosport	2019-01-11 14:09:00.000	closed	2019-01-11 16:15:10.000	17	101
3	3	Master Chef	2019-02-22 18:07:07.000	rejected	2019-02-22 18:07:38.000	12	NULL
4	4	CALENDAR	2019-03-05 22:47:02.000	closed	2019-03-06 02:30:45.000	14	103
5	5	CALENDAR	2019-04-07 19:08:07.000	closed	2019-04-07 23:48:36.000	11	102
6	6	MusicTV	2019-04-21 10:06:20.000	closed	2019-04-21 12:06:30.000	11	102
7	7	NewsTV	2019-05-01 20:20:09.000	closed	2019-05-02 00:40:09.000	13	103
8	8	Master Chef	2019-06-23 19:35:29.000	closed	2019-06-23 23:40:08.000	11	105
9	9	Eurosport	2019-07-01 19:45:17.000	rejected	2019-07-01 19:45:34.000	14	NULL
10	10	NewsTV	2019-08-20 15:17:08.000	closed	2019-08-20 20:11:05.000	11	106
11	11	MusicTV	2019-09-07 23:34:08.000	closed	2019-09-07 23:34:20.000	12	105
12	12	Master Chef	2019-09-16 19:12:44.000	closed	2019-09-16 19:12:55.000	14	105
13	13	Eurosport	2019-10-07 17:45:08.000	closed	2019-10-07 21:58:20.000	11	105
14	14	Master Chef	2019-12-06 16:57:34.000	closed	2019-12-06 19:28:14.000	17	105
15	15	POP	2019-12-25 15:06:20.000	closed	2019-12-25 18:47:22.000	16	103
16	16	POP	2020-02-11 20:06:20.000	rejected	2020-02-11 20:06:30.000	15	NULL
17	17	CALENDAR	2020-02-22 18:20:05.000	closed	2020-02-22 22:37:00.000	14	102
18	18	Master Chef	2020-03-08 20:07:20.000	rejected	2020-03-08 20:07:27.000	17	NULL
19	19	CALENDAR	2020-03-25 21:20:05.000	closed	2020-04-25 21:37:00.000	13	103
20	20	CALENDAR	2020-04-05 15:28:05.000	closed	2020-04-05 15:28:14.000	12	102
21	21	MusicTV	2020-04-19 20:57:26.000	closed	2020-04-19 20:57:29.000	15	101
22	22	NewsTV	2020-04-27 21:48:05.000	closed	2020-04-27 21:37:00.000	16	103
23	23	Eurosport	2020-05-25 10:24:30.000	closed	2020-05-25 10:24:40.000	13	105
24	24	MusicTV	2020-05-27 13:40:09.000	closed	2020-05-27 18:40:18.000	12	102
25	25	NewsTV	2020-05-28 16:17:03.000	closed	2020-05-28 16:39:17.000	17	101
26	26	CALENDAR	2020-05-28 16:40:19.000	closed	2020-05-28 16:40:24.000	15	101
27	27	Eurosport	2020-05-28 18:19:03.000	served	2020-05-28 18:19:17.000	13	105
28	28	MusicTV	2020-05-28 19:02:12.000	served	2020-05-28 19:02:24.000	12	102
29	29	NewsTV	2020-05-28 21:16:05.000	served	2020-05-28 21:16:09.000	17	101
30	30	CALENDAR	2020-05-28 23:44:12.000	served	2020-05-28 23:44:24.000	15	101
31	31	NewsTV	2020-05-30 18:46:53.000	open	2020-05-30 18:46:53.000	16	NULL
32	32	Eurosport	2020-05-30 18:48:22.000	open	2020-05-30 18:48:22.000	11	NULL

Viewing_Request

2.3 Simple Transaction

Code for this part can be found in file: **Part2-query3.sgl**. This query for each server_id in *Servers* table updates the current_viewers_number column with the total number of viewers currently assigned with that server_id.

```
01 |  BEGIN TRANSACTION
02 |          UPDATE Servers SET CurrentViewers_no = CAST(Capacity as int);
03 |
04 |          -- rollback --no changes
05 |          COMMIT
06 |                  Select * from Servers
```

The file contains also transaction for coming back to previous state of *Servers* table.

3 Part 3

Code for this part can be found in the file: **Part3.sql**.
Proposing the structure of indexes in my *VideoStreaming* database, I considered using such data in real life situations i.e. when real Streaming company would have tables with much more records accessed almost constantly.

- Clustered indexes are automatically setup on Primary Keys (during creation of tables even if the user does not specify them, they will be created). Hence, I defined unique clustered index also for column combination used to frequently access group of records i.e. our mapping table *Channel_Server*. This table will not be updated and would be very frequently accessed while serving the requests. However, in our situation there is no need to add other clustered indexes on any column in any table since the most important fields in all tables are Primary Keys which already have them.

- Non-clustered indexes are setup for all Foreign Keys, since these are the most frequently used for searching/joining operations. Only Category_id FK in *Channels* is never used in this task so we can omit this index. However, in real life situation this field would be regularly used for e.g. collecting statistics on popularity of each category so it would be good idea to set up a non-clustered index then.

- We can also add non-clustered indexes for columns often used for search operations. In this case I added them on Request_Date and Status_Date in *Viewing_Request* (since they are often use for calculation and searching) and on Expiration_Date in *Channels* (often checking if channel has expired). It would be also a good idea to add non-clustered indexes on Capacity and CurrentViewers_no in *Servers*, because they are important for serving the requests and establishing new servers if needed.

- We can also use indexes to enforce unique constraints. In this case I setup unique non-clustered index on Email column in *Viewers* table, since this column has to be unique.

4 Part 4

Code for this part is included in file: **Part4.sql**. For queries 2,3 and 5 I defined additional queries displaying more data than the final one to check if the result of final query is correct. The outputs of all queries are presented below:

1. For each viewer show channels that have never been requested by that viewer

	Viewer_id	Username	Channel_name
1	12	flower55	Eurosport
2	12	flower55	NewsTV
3	12	flower55	POP
4	13	KLausP	Master Chef
5	13	KLausP	MusicTV
6	13	KLausP	POP
7	14	JackSparrow3	MusicTV
8	14	JackSparrow3	NewsTV
9	14	JackSparrow3	POP
10	15	Atlantis	Eurosport
11	15	Atlantis	Master Chef
12	15	Atlantis	NewsTV
13	16	ALFKI	CALENDAR
14	16	ALFKI	Eurosport
15	16	ALFKI	Master Chef
16	16	ALFKI	MusicTV
17	17	misterX	CALENDAR
18	17	misterX	MusicTV
19	17	misterX	POP

2. First query displays all customers with the number of distinct channels they watched for at least an hour in 2019. The final query limits the output to viewers who viewed every channel for total at least an hour in 2019.

	Viewer_id	Username	NumberOfChannelsIn2019
1	11	JohnSmith	6
2	12	flower55	0
3	13	KLausP	1
4	14	JackSparrow3	1
5	15	Atlantis	0
6	16	ALFKI	1
7	17	misterX	2

	Viewer_id	Username	NumberOfChannelsIn2019
1	11	JohnSmith	6

3. First query displays daily popularity increase of all channels. The final query limits the output to the one with lowest ratio. (Here the ratio is equal to NULL since my *Viewing_Request* table does not contain any records with date after 30.05.2020 and the amount of current day total channel views is calculated using GETDATE() function.)

	Channel_name	TodayTotalViews	DailyAverage
1	CALENDAR	NULL	NULL
2	Eurosport	NULL	NULL
3	Master Chef	NULL	NULL
4	MusicTV	NULL	NULL
5	NewsTV	NULL	NULL
6	POP	NULL	NULL

	Channel_name	Ratio
1	Eurosport	NULL

4. For each country list top 3 most unpopular channels with lowest number of unique viewers - here I firstly create temporary table with countries, channel names and amount of unique viewers for each channel in ascending order. Then display first 3 rows for each country.

	Viewer_Country	Channel_name	UniqueViewers	CountryRowNo
1	Germany	Eurosport	1	1
2	Germany	CALENDAR	1	2
3	Germany	NewsTV	1	3
4	Poland	NewsTV	1	1
5	Poland	MusicTV	1	2
6	Poland	POP	1	3
7	Russia	Master Chef	1	1
8	Russia	Eurosport	1	2
9	Russia	NewsTV	1	3
10	US	NewsTV	1	1
11	US	Eurosport	1	2
12	US	Master Chef	1	3

5. First query displays number of short (less than 15s) views for all channels with status='served' or status='closed' (since if we take only 'served' requests the amount of data will be insufficient) in descending order. Then we limit it to first 3 rows.

	Channel_name	NumberOfShortViews
1	CALENDAR	3
2	MusicTV	3
3	NewsTV	2
4	Eurosport	2
5	Master Chef	1

	Channel_name	NumberOfShortViews
1	MusicTV	3
2	CALENDAR	3
3	Eurosport	2

5 Part 5

Code for this part can be found in: **Part5.sql**. It developa a stored procedure *serveRequsts* which serves viewing requests from *Viewing_Request* table. The procedure utilizes cursor to retrieve appropriate data *Viewing_Request* table - we need Channel_name, Request_Date, Request_Status, Status_date and Viewer_id of the user.
Then we loop over the whole table and search for requests with status='open', since all other requests have already been served.
When we find an 'open' request we check if requested channel expired. If yes - request is automatically rejected (field status set to 'rejected' and status date updated to current date). If no, we search, using a query, for available servers streaming requested channel in *Channel_Server* table. If we find a free server (i.e. with CurrentViewers_no $<$ Capacity) we assign it to the request and the user, update request status and date and update CurrentViewers_no in appropriate server.
If there are no available servers, we reject the request and create a new server with capacity set to 10, CurrentViewers_no set to 0 and Creation_Date set to the current time.

To demonstrate the implemetation of the procedure I will present tables: *Viewing_Request*, *Servers* and *Viewers* before and after executing *serveRequsts*.

Before:

	Server_id	Capacity	Server_Name	CurrentViewers_no	Creation_Date	Destruction_Date
1	101	4	server1	2	2016-10-30 13:20:00.000	NULL
2	102	3	Server2	1	2017-04-07 14:20:00.000	NULL
3	103	100	MainServer	0	2017-06-10 14:20:00.000	NULL
4	104	20	server4	0	2017-08-19 12:40:00.000	2019-01-01 14:20:13.000
5	105	7	server5	1	2018-10-10 13:20:00.000	NULL
6	106	42	Server6	0	2018-12-30 14:15:00.000	NULL

Servers

	Viewer_id	Username	Viewer_Country	Email	Server_id
1	11	JohnSmith	US	jsmith@us.com	NULL
2	12	flower55	Poland	12345@pl.pl	102
3	13	KLausP	Germany	KLAUS@ger...	105
4	14	JackSpa...	Poland	pirate@pl.pl	NULL
5	15	Atlantis	US	atlanta@us.c...	101
6	16	ALFKI	Poland	alf@xx.pl	NULL
7	17	misterX	Russia	sdfrusopen...	101

Viewers

	Request_id	Channel_name	Request_Date	Request_Status	Status_Date	Viewer_id	Server_id
1	1	POP	2019-01-01 23:09:22.000	closed	2019-01-02 01:01:04.000	11	105
2	2	Eurosport	2019-01-11 14:09:00.000	closed	2019-01-11 16:15:10.000	17	101
3	3	Master Chef	2019-02-22 18:07:07.000	rejected	2019-02-22 18:07:38.000	12	NULL
4	4	CALENDAR	2019-03-05 22:47:02.000	closed	2019-03-06 02:30:45.000	14	103
5	5	CALENDAR	2019-04-07 19:08:07.000	closed	2019-04-07 23:48:36.000	11	102
6	6	MusicTV	2019-04-21 10:06:20.000	closed	2019-04-21 12:06:30.000	11	102
7	7	NewsTV	2019-05-01 20:20:09.000	closed	2019-05-02 00:40:09.000	13	103
8	8	Master Chef	2019-06-23 19:35:29.000	closed	2019-06-23 23:40:08.000	11	105
9	9	Eurosport	2019-07-01 19:45:17.000	rejected	2019-07-01 19:45:34.000	14	NULL
10	10	NewsTV	2019-08-20 15:17:08.000	closed	2019-08-20 20:11:05.000	11	106
11	11	MusicTV	2019-09-07 23:34:08.000	closed	2019-09-07 23:34:20.000	12	105
12	12	Master Chef	2019-09-16 19:12:44.000	closed	2019-09-16 19:12:55.000	14	105
13	13	Eurosport	2019-10-07 17:45:08.000	closed	2019-10-07 21:58:20.000	11	105
14	14	Master Chef	2019-12-06 16:57:34.000	closed	2019-12-06 19:28:14.000	17	105
15	15	POP	2019-12-25 15:06:20.000	closed	2019-12-25 18:47:22.000	16	103
16	16	POP	2020-02-11 20:06:20.000	rejected	2020-02-11 20:06:30.000	15	NULL
17	17	CALENDAR	2020-02-22 18:20:05.000	closed	2020-02-22 22:37:00.000	14	102
18	18	Master Chef	2020-03-08 20:07:20.000	rejected	2020-03-08 20:07:27.000	17	NULL
19	19	CALENDAR	2020-03-25 21:20:05.000	closed	2020-04-25 21:37:00.000	13	103
20	20	CALENDAR	2020-04-05 15:28:05.000	closed	2020-04-05 15:28:14.000	12	102
21	21	MusicTV	2020-04-19 20:57:26.000	closed	2020-04-19 20:57:29.000	15	101
22	22	NewsTV	2020-04-27 21:48:05.000	closed	2020-04-27 21:37:00.000	16	103
23	23	Eurosport	2020-05-25 10:24:30.000	closed	2020-05-25 10:24:40.000	13	105
24	24	MusicTV	2020-05-27 13:40:09.000	closed	2020-05-27 18:40:18.000	12	102
25	25	NewsTV	2020-05-28 16:17:03.000	closed	2020-05-28 16:39:17.000	17	101
26	26	CALENDAR	2020-05-28 16:40:19.000	closed	2020-05-28 16:40:24.000	15	101
27	27	Eurosport	2020-05-28 18:19:03.000	served	2020-05-28 18:19:17.000	13	105
28	28	MusicTV	2020-05-28 19:02:12.000	served	2020-05-28 19:02:24.000	12	102
29	29	NewsTV	2020-05-28 21:16:05.000	served	2020-05-28 21:16:09.000	17	101
30	30	CALENDAR	2020-05-28 23:44:12.000	served	2020-05-28 23:44:24.000	15	101
31	31	NewsTV	2020-05-30 18:46:53.000	open	2020-05-30 18:46:53.000	16	NULL
32	32	Eurosport	2020-05-30 18:48:22.000	open	2020-05-30 18:48:22.000	11	NULL

Viewing_Request

After:

	Server_id	Capacity	Server_Name	CurrentViewers_no	Creation_Date	Destruction_Date
1	101	4	server1	4	2016-10-30 13:20:00.000	NULL
2	102	3	Server2	1	2017-04-07 14:20:00.000	NULL
3	103	100	MainServer	0	2017-06-10 14:20:00.000	NULL
4	104	20	server4	0	2017-08-19 12:40:00.000	2019-01-01 14:...
5	105	7	server5	1	2018-10-10 13:20:00.000	NULL
6	106	42	Server6	0	2018-12-30 14:15:00.000	NULL

Servers

	Viewer_id	Username	Viewer_Country	Email	Server_id
1	11	JohnSmith	US	jsmith@us.com	101
2	12	flower55	Poland	12345@pl.pl	102
3	13	KLausP	Germany	KLAUS@germany.com	105
4	14	JackSparrow3	Poland	pirate@pl.pl	NULL
5	15	Atlantis	US	atlanta@us.com	101
6	16	ALFKI	Poland	alf@xx.pl	101
7	17	misterX	Russia	sdfrusopen@plot.com	101

Viewers

	Request_id	Channel_name	Request_Date	Request_Status	Status_Date	Viewer_id	Server_id
1	1	POP	2019-01-01 23:09:22.000	closed	2019-01-02 01:01:04.000	11	105
2	2	Eurosport	2019-01-11 14:09:00.000	served	2020-05-31 19:06:23.073	17	101
3	3	Master Chef	2019-02-22 18:07:07.000	rejected	2019-02-22 18:07:38.000	12	NULL
4	4	CALENDAR	2019-03-05 22:47:02.000	closed	2019-03-06 02:30:45.000	14	103
5	5	CALENDAR	2019-04-07 19:08:07.000	closed	2019-04-07 23:48:36.000	11	102
6	6	MusicTV	2019-04-21 10:06:20.000	closed	2019-04-21 12:06:30.000	11	102
7	7	NewsTV	2019-05-01 20:20:09.000	served	2020-05-31 19:06:23.073	13	101
8	8	Master Chef	2019-06-23 19:35:29.000	closed	2019-06-23 23:40:08.000	11	105
9	9	Eurosport	2019-07-01 19:45:17.000	served	2020-05-31 19:06:23.073	14	101
10	10	NewsTV	2019-08-20 15:17:08.000	served	2020-05-31 19:06:23.073	11	101
11	11	MusicTV	2019-09-07 23:34:08.000	closed	2019-09-07 23:34:20.000	12	105
12	12	Master Chef	2019-09-16 19:12:44.000	closed	2019-09-16 19:12:55.000	14	105
13	13	Eurosport	2019-10-07 17:45:08.000	served	2020-05-31 19:06:23.073	11	101
14	14	Master Chef	2019-12-06 16:57:34.000	closed	2019-12-06 19:28:14.000	17	105
15	15	POP	2019-12-25 15:06:20.000	closed	2019-12-25 18:47:22.000	16	103
16	16	POP	2020-02-11 20:06:20.000	rejected	2020-02-11 20:06:30.000	15	NULL
17	17	CALENDAR	2020-02-22 18:20:05.000	closed	2020-02-22 22:37:00.000	14	102
18	18	Master Chef	2020-03-08 20:07:20.000	rejected	2020-03-08 20:07:27.000	17	NULL
19	19	CALENDAR	2020-03-25 21:20:05.000	closed	2020-04-25 21:37:00.000	13	103
20	20	CALENDAR	2020-04-05 15:28:05.000	closed	2020-04-05 15:28:14.000	12	102
21	21	MusicTV	2020-04-19 20:57:26.000	closed	2020-04-19 20:57:29.000	15	101
22	22	NewsTV	2020-04-27 21:48:05.000	served	2020-05-31 19:06:23.073	16	101
23	23	Eurosport	2020-05-25 10:24:30.000	served	2020-05-31 19:06:23.073	13	101
24	24	MusicTV	2020-05-27 13:40:09.000	closed	2020-05-27 18:40:18.000	12	102
25	25	NewsTV	2020-05-28 16:17:03.000	served	2020-05-31 19:06:23.073	17	101
26	26	CALENDAR	2020-05-28 16:40:19.000	closed	2020-05-28 16:40:24.000	15	101
27	27	Eurosport	2020-05-28 18:19:03.000	served	2020-05-31 19:06:23.073	13	101
28	28	MusicTV	2020-05-28 19:02:12.000	served	2020-05-28 19:02:24.000	12	102
29	29	NewsTV	2020-05-28 21:16:05.000	served	2020-05-31 19:06:23.073	17	101
30	30	CALENDAR	2020-05-28 23:44:12.000	served	2020-05-28 23:44:24.000	15	101
31	31	NewsTV	2020-05-30 18:46:53.000	served	2020-05-31 19:06:23.073	16	101
32	32	Eurosport	2020-05-30 18:48:22.000	served	2020-05-31 19:06:23.073	11	101

Viewing_Request

6 Attached Files

- Part1_Diagram.pdf
- Part2.sql
- Part2-query3.sql
- Part3.sql
- Part4.sql
- Part5.sql