

ACCA5026 Database Assignment Julian Milligan

by Julian Milligan

FILE	16671_JULIAN_MILLIGAN_ACCA5026_DATABASE_ASSIGNMENT_JULIAN_MILLIGAN_349706_447066609.DOCX (334.12K)		
TIME SUBMITTED	03-MAY-2021 12:34PM (UTC+0100)	WORD COUNT	3137
SUBMISSION ID	151408340	CHARACTER COUNT	17657

Name – Julian Milligan

Student number – 1904804

Course – Bachelor of Science (Part-time): Computing (Software Engineering)

Module – L5 - Databases and Application Development

Module code - ACCA5026

Date – 11/04/2021

Abstract

Shape Up Gyms have requested the report author to help create a new database system to help them manage their gym memberships across their new branches around Wales. The database will be written in SQL and the tables will be designed to 3rd Normal Factor to allow expansion of new branches and membership types. The system will also take advantage of Distribution Database model.

Table of Contents

Abstract	1
Introduction	3
Security.....	5
Views	6
Day visits	6
Members	7
Constraints	9
Stored Procedures.....	10
[dbo].[spCreateCustomer]	10
[dbo].[spSearchCustomer]	10
Moving Forward.....	12
Performance Tuning.....	13
Distributed Database Solution	14
References.....	17
Appendix	18
Appendix 1	18
Appendix 2	18
Appendix 3	20
Appendix 4	20
Appendix 5	21
Files Attached.....	22
SQL File1.sql	22
SQL File2.sql	22
SQL File3.sql	22

Introduction

The database has been designed with third normal form in mind. Branch data is stored in it's own table and so is customer data and costing. The only table that will have repeatable data is [dbo].[membership_number] but because it is using foreign keys to other tables then if a customer changes their name then only one main record need be updated in [dbo].[customer] and no need to change multiple records in [dbo].[membership_number].

The database consist of 5 tables :-

[dbo].[branch]

This will hold all the relevant information about each branch such as Name , Address , Email , Telephone and Region which will be used to distribute the database. There is also a column called membership_cost_fk which is the foreign key to the table [dbo].[membership_cost] .

[dbo].[customer]

This table holds all the relevant data for the customer. Such as preferred Branch , membership type , and various personal information. It does include their bank details however as mentioned in the security section of this report access to this information is controlled

[dbo].[membership_cost]

This tables holds the data for the costings of a Yearly membership , monthly and any joining fees. There are 8 branches at the moment and 8 entries setup. However some gyms are new to Shape Up Gyms so they are using the same pricing structure as some of the older gyms. Creating the database this has the benefit of if one pricing structure is updated then any Gym using it will automatically be updated.

[dbo].[membership_number]

This table marries the customer , branch and membership type together with their unique membership number. The Store Procedure [dbo].[spCreateCustomer] inserts a new customer and will also generate a membership number by concatenating 'SUG'+Branch Number + row number + customer ID. This way each new yearly or monthly customer has an unique membership ID and every day visit is also recorded. A copy of the SP is in [Appendix 2](#).

[dbo].[membership_type]

This is a small table but in the future could be expanded to included classes or other membership types. Currently it includes a yearly membership which can be paid in a one of payment or monthly. It all so has a Day session.

An ERD for the purposed database is in [Appendix 1](#) . The design of the database did alter from the ERD in the build stage as the designer felt certain features would not meet the 3NF rule. The main area affected was the table [dbo].[membership_cost] . Including the branch id as a foreign key , this would not allow multiple branches to use the same pricing structure. [Appendix 3](#) shows a diagram generated by SQL Management Studio that shows the changes.

Security

As quoted by Paul Wilton

“In today’s world, the security of the data in your database is a primary concern. Privacy concerns and laws often make exposure of corporate data a catastrophic problem. Exposure of personal information such as social security numbers or medical information can bring costly lawsuits or even government fines and litigation.” [1]

Within the tables there is data that certain members of staff do not need to see this includes the customers bank details which only the accounts teams would need to see. To over come this a number of views have been created. There are 3 levels of security

1. Accounts – they are able to see records of daily visits , any membership that need renewing and access to all membership.
2. Branch Access – each branch has two views
 - a. Day visits
 - b. All yearly members
3. Region managers – each region manager has access to two views
 - a. Daily figures for all the branches in their region
 - b. Yearly figures for the branches in their regions
 - c. Day visits and member visits for the branches in their region.

Please see the file SQL3.sql for the script used to create these.

Views

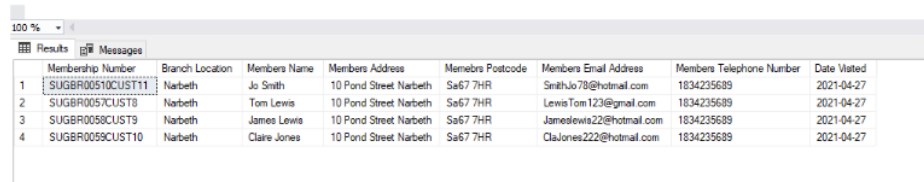
Day visits

This view is designed to show some very basic information about all daily visits. It uses 2 inner joins to display the branch name and not the branch code and the membership number of the customer. The difference between each one for the branches is in the WHERE clause. Changing the customer.branch_fk will shows the records for each branch.

Code

```
SELECT TOP (100) PERCENT membership_number.membership_number_pk AS  
[Membership Number], branch.branch_location AS [Branch Location],  
customer.customer_name AS [Members Name],  
customer.customer_address AS [Members Address],  
customer.customer_postcode AS [Members Postcode], customer.customer_email AS  
[Members Email Address],  
customer.customer_telephone AS [Members Telephone Number],  
CONVERT(date, customer.customer_start_date) AS [Date Visited]  
FROM dbo.customer  
INNER JOIN dbo.branch ON customer.branch_fk = branch.branch_number  
INNER JOIN dbo.membership_number ON customer.customer_id_pk =  
membership_number.customer_id_fk  
WHERE (customer.branch_fk = 'BR005') AND (customer.membership_type_fk = '2')  
ORDER BY 'Members Name'
```

Result



	Membership Number	Branch Location	Members Name	Members Address	Members Postcode	Members Email Address	Members Telephone Number	Date Visited
1	SUGBR00510CUST11	Narbeth	Jo Smith	10 Pond Street Narbeth	SA67 7HR	SmithJo78@hotmail.com	1834235689	2021-04-27
2	SUGBR0057CUST8	Narbeth	Tom Lewis	10 Pond Street Narbeth	SA67 7HR	LewisTom123@gmail.com	1834235689	2021-04-27
3	SUGBR0058CUST9	Narbeth	James Lewis	10 Pond Street Narbeth	SA67 7HR	JamesLewis22@hotmail.com	1834235689	2021-04-27
4	SUGBR0059CUST10	Narbeth	Claire Jones	10 Pond Street Narbeth	SA67 7HR	ClaJones222@hotmail.com	1834235689	2021-04-27

Members

The members view is similar to the day visits but instead of the last column being called Date Visited it is now called Joining Date. The rest of the columns are identical and so is the functionality between each branch.

Code

```
SELECT TOP (100) PERCENT membership_number.membership_number_pk AS  
[Membership Number], branch.branch_location AS [Branch Location],  
customer.customer_name AS [Members Name],  
  
customer.customer_address AS [Members Address],  
customer.customer_postcode AS [Members Postcode], customer.customer_email AS  
[Members Email Address],  
  
customer.customer_telephone AS [Members Telephone Number],  
CONVERT(date, customer.customer_start_date) AS [Joining Date]  
  
FROM dbo.customer INNER JOIN  
  
dbo.branch ON customer.branch_fk = branch.branch_number INNER JOIN  
  
dbo.membership_number ON customer.customer_id_pk =  
membership_number.customer_id_fk  
  
WHERE (customer.branch_fk = 'BR001') AND (customer.membership_type_fk = '1')  
  
ORDER BY [Members Name]
```

Results

	Membership Number	Branch Location	Members Name	Members Address	Members Postcode	Members Email Address	Members Telephone Number	Joining Date
1	SUGBR00111CUST12	Swansea	Ben Grice	30 Camarthen Road Swansea	SA1 6TR	BGrice@hotmail.com	1834235689	2021-04-27
2	SUGBR00112CUST13	Swansea	George Morgan	15 Kings Road Swansea	SA1 6TR	GMorgan@hotmail.com	1792568952	2021-04-27
3	SUGBR00113CUST14	Swansea	Cath Morton	67 New Road Swansea	SA1 7GF	CMorton@hotmail.com	1792849515	2021-04-27
4	SUGBR0011CUST1	Swansea	Julian Milligan	7 Grugos Avenue Pontyberem LLanel	SA15 5AF	julian.milligan25@gmail.com	1267235689	2021-04-25
5	SUGBR0012CUST2	Swansea	Kyle Milligan	7 Grugos Avenue Pontyberem LLanel	SA15 5AF	Kyle.milligan@gmail.com	1267235689	2021-04-25
6	SUGBR0013CUST3	Swansea	Sian Milligan	7 Grugos Avenue Pontyberem LLanel	SA15 5AF	Sian.milligan@gmail.com	1267235689	2021-04-25
7	SUGBR0014CUST4	Swansea	Rhannon Milligan	7 Grugos Avenue Pontyberem LLanel	SA15 5AF	Sian.milligan@gmail.com	1267235689	2021-04-25

Mailing List

Returns any customer that has 'Yes' in the column customer_mailing_list. Again it only returns information needed for the mailing list such as Membership Number , Branch , Members name and email address.

	Membership Number	Branch Location	Members Name	Members Email Address
1	SUGBR00111CUST12	Swansea	Ben Grice	BGrice@hotmail.com
2	SUGBR00113CUST14	Swansea	Cath Morton	CMorton@hotmail.com
3	SUGBR00217CUST18	Camarthen	Julian Williams	jwilliams34@hotmail.com
4	SUGBR00218CUST19	Camarthen	Justin Williams	jwilliams34@hotmail.com
5	SUGBR00510CUST11	Narbeth	Jo Smith	SmithJo78@hotmail.com
6	SUGBR0057CUST8	Narbeth	Tom Lewis	LewisTom123@gmail.com
7	SUGBR0058CUST9	Narbeth	James Lewis	Jameslewis22@hotmail.com
8	SUGBR0059CUST10	Narbeth	Claire Jones	ClaJones222@hotmail.com
9	SUGBR0076CUST7	Ammanford	James Greenwell	GreenersJames@gmail.com

Accounting views

There are two views for the accounting team which are very different. One shows all the daily visits and the cost the other shows all yearly members, but an expiry date is now included. The expiry date is not included in any table but is generated by using the SQL function DATEADD which adds a year to the joining date. This will allow the Accounts team to query the data and look for any accounts that maybe expiring within 30 days and the send out reminders or auto renew their account if they have selected that option.

Daily visits

	Membership Number	Branch Location	Members Name	Members Address	Members Postcode	Members Email Address	Members Telephone Number	Date Visted	Daily Charge
1	SUGBR0020CUST5	Camarthen	Kirsty Ball	7 Park Hall Camarthen	SA31 3er	kirsty.ball@hotmail.com	1267235689	2021-04-25	10
2	SUGBR00510CUST11	Narbeth	Jo Smith	10 Pond Street Narbeth	SA67 7HR	SmithJo78@hotmail.com	1834235689	2021-04-27	12
3	SUGBR0057CUST8	Narbeth	Tom Lewis	10 Pond Street Narbeth	SA67 7HR	LewisTom123@gmail.com	1834235689	2021-04-27	12
4	SUGBR0058CUST9	Narbeth	James Lewis	10 Pond Street Narbeth	SA67 7HR	Jameslewis22@hotmail.com	1834235689	2021-04-27	12
5	SUGBR0059CUST10	Narbeth	Claire Jones	10 Pond Street Narbeth	SA67 7HR	ClaJones222@hotmail.com	1834235689	2021-04-27	12

Renewals

	Members Telephone Number	Joining Date	Renewal Date	Bank Name	Bank Account Number	Bank Sort Code	Renew Account
	1834235689	2021-04-27	2022-04-27	HSBC	45958615	51-54-53	Yes
	1792568952	2021-04-27	2022-04-27	Lloyds	78894556	88-55-44	Yes
	1792849515	2021-04-27	2022-04-27	Barclays	78451958	74-56-12	Yes
	1267235687	2021-04-27	2022-04-27	Barclays	36251478	12-45-89	Yes
m	1267234391	2021-04-27	2022-04-27	HSBC	731598442	79-13-82	Yes
m	1267234391	2021-04-27	2022-04-27	HSBC	48592648	78-65-48	Yes
	1554785689	2021-04-27	2022-04-27	Barclays	23568974	20-15-25	Yes
.com	1554785689	2021-04-27	2022-04-27	Barclays	78561245	21-17-78	Yes

Constraints

While creating the tables a number of columns were set to NOT NULL these were used for data that needs to be entered in order for the system to function correctly. Some examples of these were Name , Address , telephone and start date in the customer table. Items like the bank details were allowed null values as for day visits there is no need to record these. The Branch table only two items were allowed NULLS , these were email and website. This is because Shape Up Gym`s are currently taking over a number of smaller gyms and bring them under the main company umbrella and at the time of this report they did not have their own websites.

Two further constrains were added one to make sure each Membership number is unique and the second to make sure only a set number of payment types are inserted. The 2nd one will assist the accounts team when they need to query the data. Knowing that there will only be a certain number of options means they could create reports that query the data via SQL Reporting Services.

Further reading on SQL Constraints at w3schools [2] allowed the designer to implement the two constraints above. The constraints added were column based however a table constraint can be added but the designer felt that a full table constraint would not benefit this project as each table column is unique and there was no global constraint that could be added.

Stored Procedures

Two store procedures were written to help insert a new customer record and search for a customer.

[dbo].[spCreateCustomer]

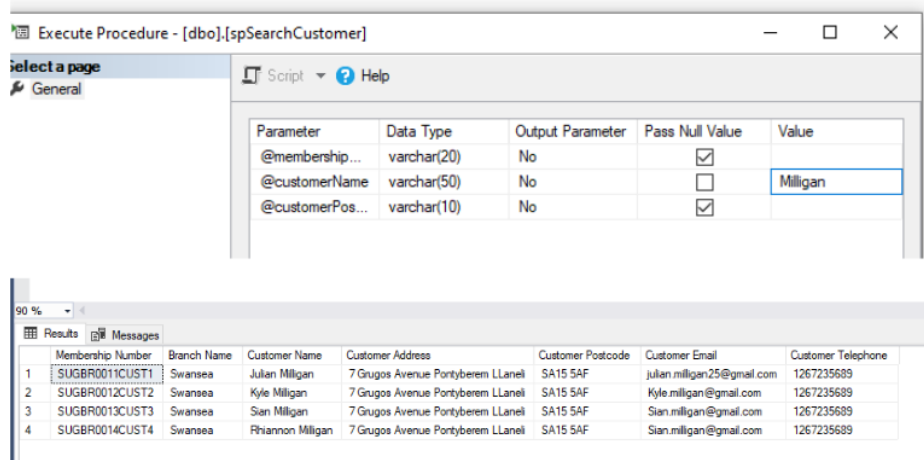
A number of variables are created which match their columns in the customer table and these are filled when the SP is executed. Below these some internal variables are create and populated via just below, these are used to dynamically create the customer ID this is done by counting the number of rows and adding 1 then concatenating it the string 'CUST'. The next part creates the membership number this is a little more complex as the concatenation adds 'SUG' + branch number + row count + customer ID to create a unique membership number.

Following on from creating the customer ID and Membership Id the customers details are inserted using the Begin Transaction and Commit Transaction functions. Once that has been committed then the membership number is inserted using the same method.

[dbo].[spSearchCustomer]

This SP can be used to search for a customer record from one of three bits of data membership number, customer surname or customer postcode. It uses an IF Else statement to determine which bit of data is passed. Currently only one bit of data can be used at once, further development could be done where multiple bits of data could be passed.

Example below shows searching for the surname 'Milligan'

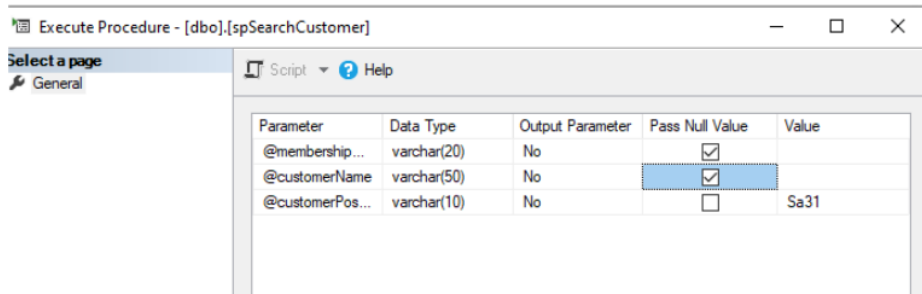


Execute Procedure - [dbo].[spSearchCustomer]

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@membership...	varchar(20)	No	☒	
@customerName	varchar(50)	No	☐	Milligan
@customerPos...	varchar(10)	No	☒	

	Membership Number	Branch Name	Customer Name	Customer Address	Customer Postcode	Customer Email	Customer Telephone
1	SUGBR0011CUST1	Swansea	Julian Milligan	7 Grugos Avenue Pontyberem LLaneli	SA15 5AF	julian.milligan25@gmail.com	1267235689
2	SUGBR0012CUST2	Swansea	Kyle Milligan	7 Grugos Avenue Pontyberem LLaneli	SA15 5AF	Kyle.milligan@gmail.com	1267235689
3	SUGBR0013CUST3	Swansea	Sian Milligan	7 Grugos Avenue Pontyberem LLaneli	SA15 5AF	Sian.milligan@gmail.com	1267235689
4	SUGBR0014CUST4	Swansea	Rhiannon Milligan	7 Grugos Avenue Pontyberem LLaneli	SA15 5AF	Sian.milligan@gmail.com	1267235689

The example below shows searching for a partial postcode of 'SA31'



Results							
Messages							
	Membership Number	Branch Name	Customer Name	Customer Address	Customer Postcode	Customer Email	Customer Telephone
1	SUGBR0020CUST5	Camrathen	Kirsty Ball	7 Park Hall Camrathen	Sa31 3er	kirsty.ball@hotmail.com	1267235689
2	SUGBR00214CUST15	Camrathen	Andy Fisher	67 Maple Crescent Camrathen	Sa31 8YH	AFisher@gmail.com	1267235687
3	SUGBR00215CUST16	Camrathen	Rob Clement	6 New Road Camrathen	Sa31 3RT	RClement@gmail.com	1267235687
4	SUGBR00216CUST17	Camrathen	Janis Davies	10 Town Road Camrathen	Sa31 8YH	jdavies123@hotmail.com	1267237451
5	SUGBR00217CUST18	Camrathen	Julian Williams	56 Park Hall Camrathen	Sa31 3KU	jwilliams34@hotmail.com	1267234391
6	SUGBR00218CUST19	Camrathen	Justin Williams	56 Park Hall Camrathen	Sa31 3KU	jwilliams34@hotmail.com	1267234391

It only brings back enough information for the member of staff to validate the customer.

Moving Forward

Using the SP's and views a back office program could be written in ASP .Net Core using Entity Framework and Kendo Grids ([ASP.NET Core Grid Key Features Demo](#) | [Telerik UI for ASP.NET Core](#)). Using the security already created this would allow staff to see up to date information for their branch and allow customers to create their own accounts on-line. Also using ASP forms we could then force options for the customer / staff adding even more security on the data that is being entered.

Of course added protection would be need to prevent against SQL injection from the website. Following some of the guideline layout by Mircosoft [3] SQL injection can be prevented by

1. Securing the Connection
2. Restrict access to configuration files
3. Do not allow untrusted users access to the site.
4. Log all Selects , updates , Inserts and Deletes
5. Parameterize any data passed to an Entity Command.

Performance Tuning

Performance is one of the most important aspects to consider when designing a database, having all that data there but queries that take an age to return results are what the users will see and not the tables and code being it. Based on the designer's experience two Indexes were created one for the membership number which is a unique index

```
CREATE UNIQUE INDEX index_membershipNumber  
ON [dbo].[membership_number] (membership_number_pk)
```

This was created because one the queries that maybe performed on a repeated basis is search for a customer via their membership number. For example when they arrive at the Gym and swipe a membership card or staff search for a customer on the till system.

```
CREATE INDEX index_customerName  
ON [dbo].[customer] (customer_name)
```

The other was a nonclustered index on the customer name in the customer table. This one was chosen because if a customer does not have their membership card then the most common item the staff will search on is their name, with all the branches expected to be very busy after lockdown this would be the best option to help keep the speed of the database at optimum.

There are further indexes that could be added in the future, extra reading was done on indexes and why it best practice to use them. [4] [5] [6].

Indexes are used to speed up querying the database, meeting with the users to work out their needs and what they maybe querying most often is very important. Indexes can be dropped just as a table and will not affect any data currently in the tables.

Distributed Database Solution

As there is currently no back office solution in place for Shape Up Gyms then they will need to access the database directly to query it. As the locations are spread across Wales and internet speeds verifying then a the database will need to distributed to each site via their Region. As Jim Walker says

“ Distributed SQL is a single logical database deployed across multiple physical nodes in a single data center or across many data centers if need be; all of which allow it to deliver elastic scale and bulletproof resilience.” [7]

If we take the head office as the Swansea branch and they will house the main SQL Server then out of the 5 tables only 2 tables need to be distributed to the other sites. This is because all the admin work on membership_cost , membership_type and branch will be done at Swansea. This means each region will only need access to the customer table and membership_number.

Based on the article written by Dave Pinal there are two types of policy when considering a distributed database system.

“Arbitrary Distribution: This is when data is distributed across database instances, but without any consideration or understanding for specific application requirements and how the data will be used by users or the application;

Declarative, Policy-Based Distribution: This is when data is distributed across database instances, but in a way that specifically understands all application requirements, data relationships, transactions, and how the data is used in reads and writes by the application.” [8]

We will use the Declarative , Policy-Based Distribution policy as it will align with the schema of the database , and will more efficient because it will lead to less nodes and use of the network infructure.

To implement the distribution across the 4 regions two procedures will need to be done , one to create the connection and the other to give access to the users.

Create the connection

```
1. sp_addlinkedserver
2. [ @server= ] 'server'
3. [ , [ @srvproduct= ] 'product_name'
4. [ , [ @provider= ] 'provider_name'
5. [ , [ @datasrc= ] 'data_source'
6. [ , [ @location= ] 'location' ]
7. [ , [ @provstr= ] 'provider_string'
8. [ , [ @catalog= ] 'catalog' ]
```

To add the users

```
1. sp_addlinkedserver [ @rmtsrvname = ] 'rmtsrvname'  
2. [ , [ @useself = ] 'useself' ]  
3. [ , [ @locallogin = ] 'locallogin' ]  
4. [ , [ @rmtuser = ] 'rmtuser' ]  
5. [ , [ @rmtpassword = ] 'rmtpassword' ]
```

The following code and guidance is from c-sharpcorner [9]

Conclusion

Currently the amount of data in the tables is small but as the branches reopen and members start coming back and signing up to Shape Up Gyms the records will increase, because of the 3NF design it will allow a customer to become a member of the Carmarthen Gym on a yearly membership but if needed they could use the Cardigan branch as a daily visit but only have one customer record. If a branch was to close updating a customers record and moving them to another branch will simple as only one column would need to be changed.

References

[P. Wilton, Beginning SQL, Wrox, 2005.

1

]

["SQL Create Constraints," [Online]. Available:

2 https://www.w3schools.com/sql/sql_constraints.asp. [Accessed 04 2021].

]

["Security Considerations (Entity Framework)," Microsoft, 30 03 2017. [Online]. Available:

3 [https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-](https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations)

4 [considerations](https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations). [Accessed 02 05 2021].

]

["SQL Server Indexes," SQL Server Tutorial.net, [Online]. Available:

4 <https://www.sqlservertutorial.net/sql-server-indexes/>. [Accessed 05 2021].

]

["W3 School," W3 School, [Online]. Available:

5 https://www.w3schools.com/sql/sql_create_index.asp. [Accessed 05 2021].

]

["Indexes," Microsoft, 21 12 2016. [Online]. Available: [https://docs.microsoft.com/en-](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-ver15)

6 [us/sql/relational-databases/indexes/indexes?view=sql-server-ver15](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-ver15). [Accessed 05 2021].

]

[J. Walker, "What is Distributed SQL? An Evolution of the Database," 18 08 2020. [Online].

7 Available: [https://www.cockroachlabs.com/blog/what-is-distributed-](https://www.cockroachlabs.com/blog/what-is-distributed-sql/#:~:text=%20What%20comprises%20a%20Distributed%20SQL%20database%3F%20,of%20data%20throughout%20a%20complex%2C%20widely...%20More%20)

8 [sql/#:~:text=%20What%20comprises%20a%20Distributed%20SQL%20database%3F%20,of%20data%20throughout%20a%20complex%2C%20widely...%20More%20](https://www.cockroachlabs.com/blog/what-is-distributed-sql/#:~:text=%20What%20comprises%20a%20Distributed%20SQL%20database%3F%20,of%20data%20throughout%20a%20complex%2C%20widely...%20More%20). [Accessed 05 2021].

[P. Dave, "How to Create a Distributed Relational SQL Database," 04 09 2014. [Online].

8 Available: [https://blog.sqlauthority.com/2014/10/06/mysql-how-to-create-a-distributed-](https://blog.sqlauthority.com/2014/10/06/mysql-how-to-create-a-distributed-relational-sql-database/)

9 [relational-sql-database/](https://blog.sqlauthority.com/2014/10/06/mysql-how-to-create-a-distributed-relational-sql-database/). [Accessed 05 2021].

]

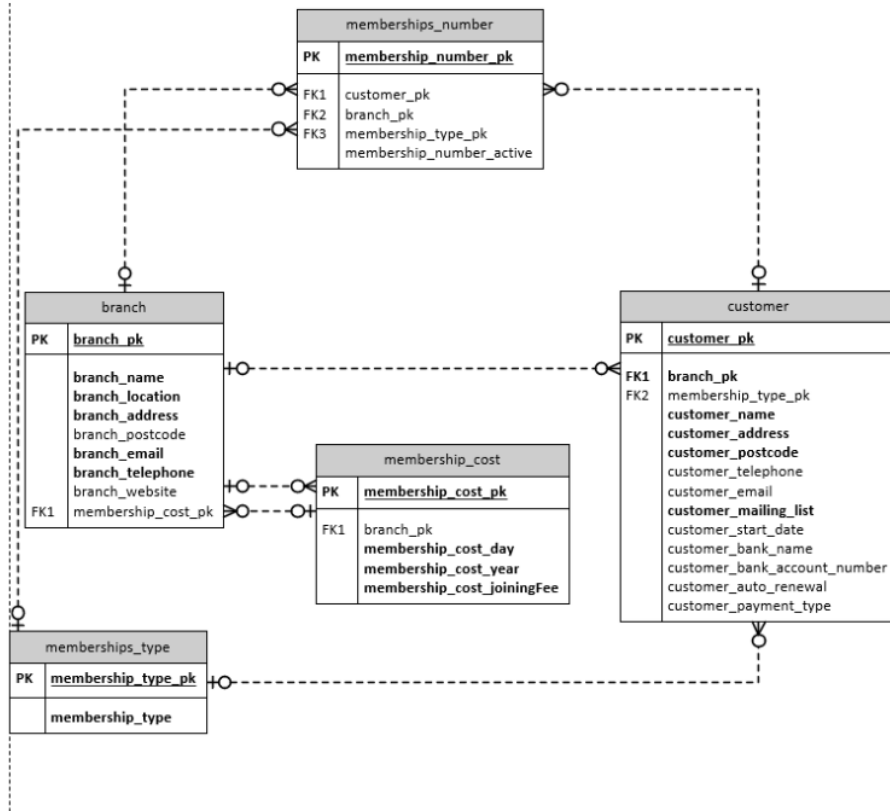
[J. C. Olamendy, "Distributed Databases in SQL Server," 10 07 2019. [Online]. Available:

9 <https://www.c-sharpcorner.com/article/distributed-databases-in-sql-server-2005/>.

10 [\[Accessed 05 2021\].](https://www.c-sharpcorner.com/article/distributed-databases-in-sql-server-2005/)

Appendix

Appendix 1



Appendix 2

```
/****** Object: StoredProcedure [dbo].[spCreateCustomer]    Script Date:
30/04/2021 12:58:15 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Julian Milligan>
-- Create Date: <25/04/2021>
-- Description: Insert a new customer>
-- =====
ALTER PROCEDURE [dbo].[spCreateCustomer]

    @branch_fk varchar(10),
    @membership_type_fk varchar(10) ,
    @customer_name varchar(50),
    @customer_address varchar(100),
    @customer_postcode varchar(8),
```

```

        @customer_telephone int,
        @customer_email varchar(100),
        @customer_mailing_list varchar(10),
        @customer_bank_name varchar(25),
        @customer_bank_account_number int,
        @customer_bank_sortcode varchar(8),
        @customer_auto_renewal varchar(10),
        @customer_payment_type varchar(10)

AS
BEGIN
    --Auto create customer id and membership number
    --Declare internal variables
    DECLARE @customer_id_pk varchar(10)
    DECLARE @rowCountCust int
    DECLARE @rowCountMemb int
    DECLARE @membership_number varchar (20)

    --Set cust id
    SET @rowCountCust = (SELECT COUNT (*) FROM [dbo].[customer])
    SET @customer_id_pk = ( SELECT CONCAT ('CUST' , CAST(@rowCountCust + 1 AS
varchar)))

    --Set membership number
    SET @rowCountMemb = (SELECT COUNT (*) FROM [dbo].[membership_number])
    SET @membership_number = (SELECT CONCAT ('SUG' , @branch_fk , @rowCountMemb
, @customer_id_pk ))

    BEGIN TRANSACTION

        --Insert customer
        INSERT INTO [dbo].[customer] (customer_id_pk , branch_fk ,
membership_type_fk , customer_name , customer_address , customer_postcode ,
customer_telephone , customer_email , customer_mailing_list , customer_start_date
, customer_bank_name , customer_bank_account_number , customer_bank_sort_code ,
customer_auto_renewal , customer_payment_type )
        VALUES (
            @customer_id_pk,
            @branch_fk,
            @membership_type_fk,
            @customer_name,
            @customer_address,
            @customer_postcode,
            @customer_telephone,
            @customer_email,
            @customer_mailing_list,
            GETDATE(),
            @customer_bank_name,
            @customer_bank_account_number,
            @customer_bank_sortcode,
            @customer_auto_renewal,
            @customer_payment_type
        )

    COMMIT TRANSACTION

    BEGIN TRANSACTION

        INSERT INTO [dbo].[membership_number] ( membership_number_pk ,
customer_id_fk , branch_fk , membership_type_fk )

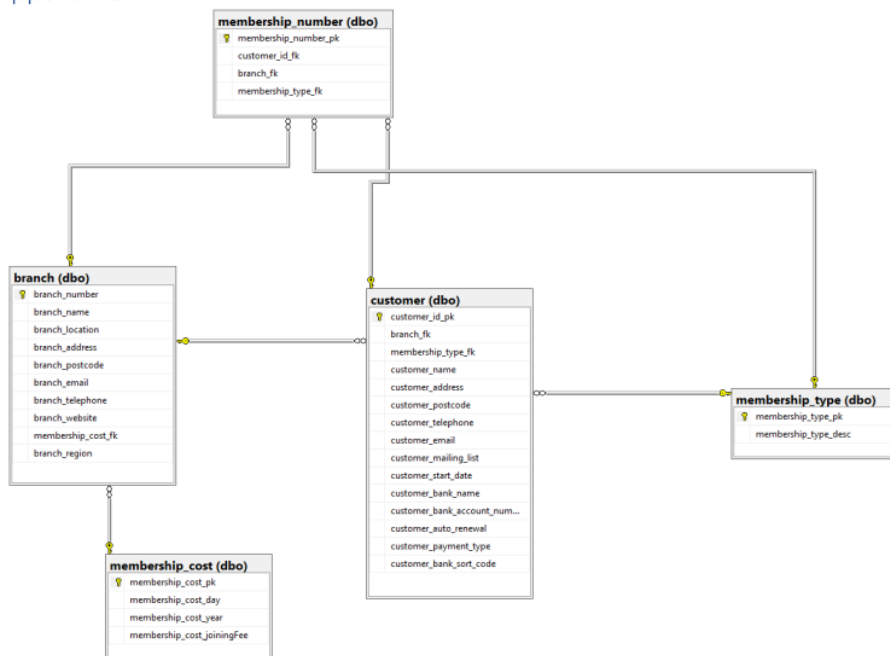
```

```
VALUES (
    @membership_number,
    @customer_id_pk,
    @branch_fk,
    @membership_type_fk
)
```

```
COMMIT TRANSACTION
```

```
END
```

Appendix 3



Appendix 4

```
Msg 2627, Level 14, State 1, Line 909
Violation of PRIMARY KEY constraint 'pk_customer_id'. Cannot insert duplicate key in object 'dbo.customer'. The duplicate key value is (CUST4).
The statement has been terminated.

Completion time: 2021-04-30T14:10:20.8896197+01:00
```

Appendix 5

```
Msg 847, Level 16, State 0, Line 321
The INSERT statement conflicted with the CHECK constraint "CHK_Payment". The conflict occurred in database "Assignment", table "dbo.customer", column 'customer_payment_type'.
The statement has been terminated.

Completion time: 2021-04-20T14:11:26.5594632+01:00
|
```

Files Attached

SQL File1.sql

Shows the code to create

1. All the tables.
2. Add the extra constraints.
3. Create the indexes.

SQL File2.sql

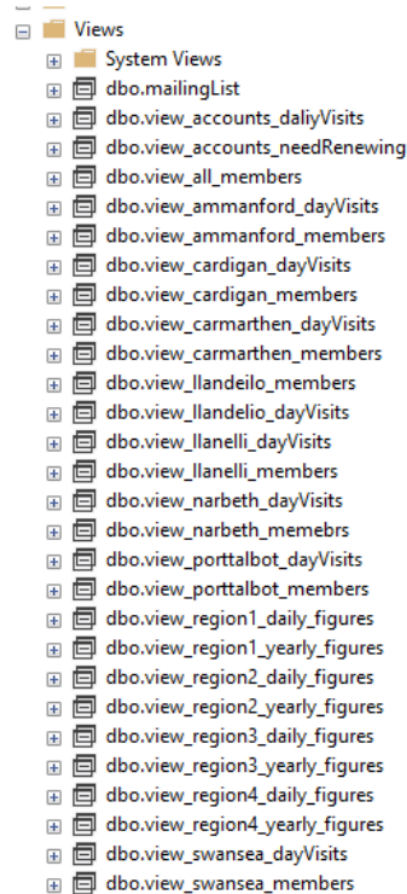
Shows the code for

1. Insert Branch data
2. Insert Customer data
3. Insert Membership Costs
4. There is no code to insert data into the membership table as this is created dynamically via the stored procedure called [dbo].[spCreateCustomer]
5. Insert Membership Types.
6. Reinserts to test the constraint no duplicate membership number.
7. Reinsert to test the constraint to test payment type.

SQL File3.sql

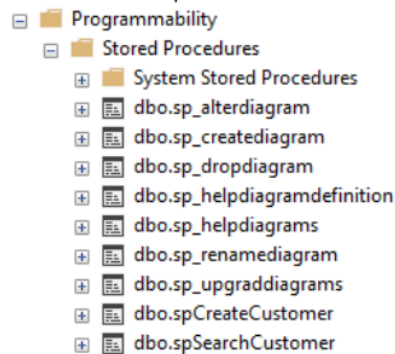
Shows the code used to create

1. All Login and User
2. The code to grant permissions to each view
3. A selection of views (all the other views are variants of the ones shown, the WHERE clause just needs to be altered.)



a.

4. The creation of the store procedure.



a.

ACCA5026 Database Assignment Julian Milligan

GRADEMARK REPORT

FINAL GRADE

60/100

GENERAL COMMENTS

Instructor

You have designed a suitable schema but to fully address the requirements you need to consider the staff within the gyms and some consideration of the facilities and how customers / staff link to the facilities.

That said you demonstrate a sound understanding of the SQL development environment and you have built and tested all your table correctly. You have demonstrated by example the use of indexing and transaction management. Perhaps more narrative on distributed databases and some theoretical examples would help with your considerations.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

