

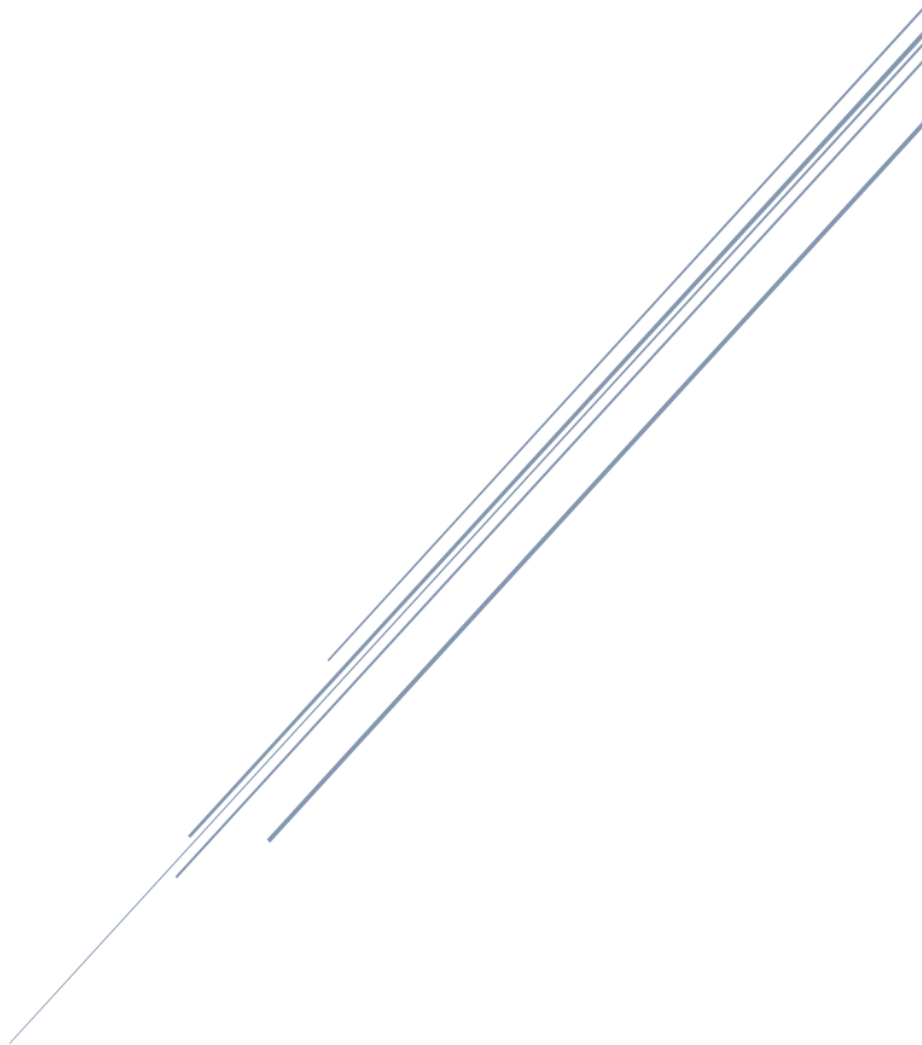
Databases and Application Development Report - Sharla Passariello

by Sharla Passariello

FILE	18703_SHARLA_PASSARIELLO_DATABASES_AND_APPLICATION_DEVELOPMENT_REPORT_-_SHARLA_PASSARIELLO_349706_1794975088.DOCX (364.81K)		
TIME SUBMITTED	10-MAY-2021 06:28PM (UTC+0100)	WORD COUNT	2250
SUBMISSION ID	152559062	CHARACTER COUNT	12748

DATABASES AND APPLICATION DEVELOPMENT

ACCA5026



Sharla Passariello
1903420

Contents

Abstract..... 2

Case Study - Shape Up Gyms 2

Shape Up Gyms Solution..... 2

 Distributed Database Solution 2

 Fragmentation Schema 4

 Performance Tuning..... 8

Conclusion..... 9

References 10

Abstract

A case study has been provided for a company known as Shape Up Gyms who are looking to upgrade their I.T systems. This report will look at distributed database solutions and performance tuning to provide Shape Up Gyms with a database solution that meets their criteria and streamlines their database performance.

Case Study - Shape Up Gyms

Shape Up Gyms is a small company operating out of Swansea who are expanding their service of providing access to modern gyms under affordable membership plans to other parts of Wales. To support a growing customer base and improve services, Shape Up Gyms are to upgrade their I.T systems and require the following from the new system:

- A distributed database model is used
- Each gym sets their membership prices
- Gyms may require a joining fee
- Customers have the choice between a daily pass or fixed term membership
- Membership fees can be paid all at once or monthly

Shape Up Gyms Solution

Two problems that Shape Up Gyms currently face is producing a distributed database solution and performance tuning for their most important transactions. The following sections will provide a proposed solution to both problems.

Distributed Database Solution

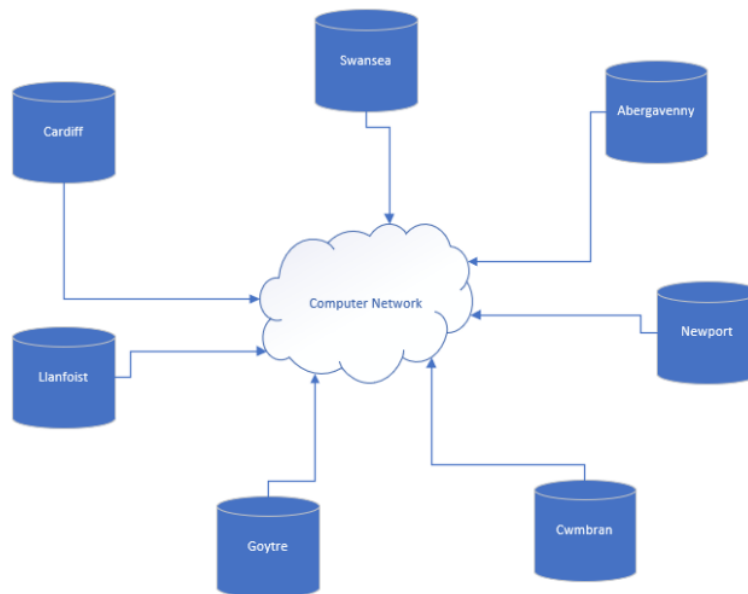
Per the case study requirements, a distributed database solution was required over a centralised database solution.

A centralised database solution stores the databases at a single location where it is maintained. This offers an advantage of maximum data integrity and minimal data redundancy as the data is stored in one location making it easier to coordinate consistent and accurate data whilst minimising redundant data entries. Whilst a central database solution provides advantages in terms of data quality, this fetches disadvantages of reduced performance as it takes longer to search and access the data, in-turn creating data access traffic which may bottleneck if multiple users attempt access simultaneously. A centralised database solution also carries more risk around disaster recovery. If system failures were to occur, all the data will be at risk as it is stored in the one location. [1]

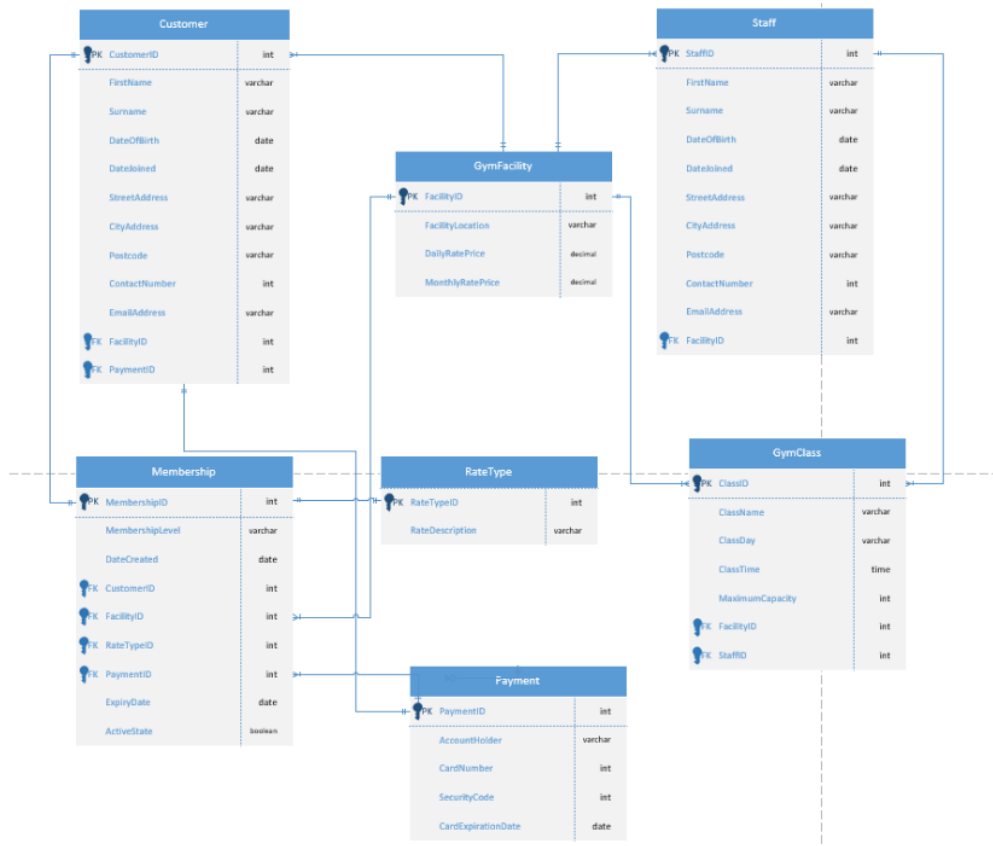
The distributed database solution enables a safer, more robust storage of the data by spreading it across multiple locations, resulting in less data being destroyed if a system failure were to occur in one of the locations. The solution provides faster performance when accessing data as the data access traffic is also spread across the different locations. Furthermore, it allows resilient growth of the databases as work on one site will not impact the others, resulting in easier expansion of the system. The disadvantage of the distributed data solution revolves around data integrity. If a communal data format is not followed, this can cause problems when retrieving data from the different locations. This can also increase the processing overhead as many messages need to be shared between sites to complete distributed transactions. [2] There are further complexities

involved with the distributed solution such as distinguishing failures and complex implementations, however the advantages of the distributed model are outweighing the disadvantages with this type of solution becoming more commonplace. [3]

It is recommended that Shape Up Gyms uses the homogeneous distributed database type. The homogeneous distributed database solution requires sites to use similar software. A Database Management System (DBMS) is a software application used for creating, managing, and administering databases. The DBMS that has been used for the Shape Up Gyms case study is Microsoft SQL Server. The homogeneous database type will mean that Shape Up Gyms regions are aware of all other sites and are using the same software to process requests. This will also provide incremental growth and allow increased performance. [4]



The following displays the proposed ERD for the databases within the distributed solution:



Fragmentation Schema

Fragmentation is the dividing of a relation (a table) into a set of smaller relations (tables) which are known as fragments. There are different types of fragmentation that can be used to breakdown relations: horizontal, vertical, mixed, and derivative.

Horizontal fragmentation divides relations by the row utilising a where clause. Vertical fragmentation divides relations by the columns. Mixed fragmentation divides relations using both the rows and columns combined. Derivative fragmentation is used to join fragments where a relation cannot be based on one of its own attributes but is derived from another relation.

To best choose what fragmentation type to use, the database must be analysed for its most important transactions. The analysis for fragmentation should be based on quantitative information such as frequency in which transactions are run and the site from which a transaction is run. The analysis should also be based on qualitative information such as the tables, columns and rows accessed, the type of access needed, and the predicates of the read operations. Once this analysis is complete, fragmentation can be designed to best provide the most relevant data needed for a scenario in an efficient manner.

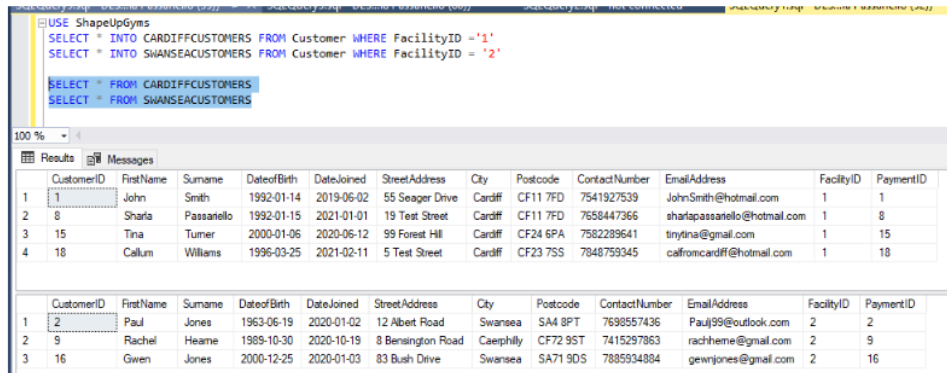
Based on the case study criteria and the design of the database and tables, the following fragmentations have been designed for Shape Up Gyms.

Horizontal Fragmentation

The Customer and Staff tables are designed to best suit the horizontal fragmentation design as per the following examples.

Searching for customers based on the region/gym facility:

Using the FacilityID for the WHERE clause will display customers only based in that facility/region. A gym in Cardiff will mostly be utilising customers from Cardiff and therefore it is logical to use horizontal fragmentation for this scenario.



```
USE ShapeUpGyms
SELECT * INTO CARDIFFCUSTOMERS FROM Customer WHERE FacilityID = '1'
SELECT * INTO SWANSEACUSTOMERS FROM Customer WHERE FacilityID = '2'

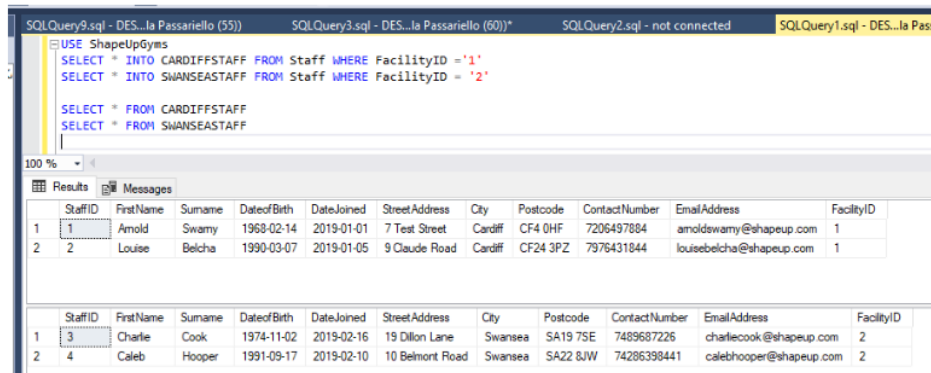
SELECT * FROM CARDIFFCUSTOMERS
SELECT * FROM SWANSEACUSTOMERS
```

	CustomerID	FirstName	Surname	DateOfBirth	DateJoined	StreetAddress	City	Postcode	ContactNumber	EmailAddress	FacilityID	PaymentID
1	1	John	Smith	1992-01-14	2019-06-02	55 Seager Drive	Cardiff	CF11 7FD	7541927539	JohnSmith@hotmail.com	1	1
2	8	Sharla	Passaniello	1992-01-15	2021-01-01	19 Test Street	Cardiff	CF11 7FD	7658447366	sharlpassaniello@hotmail.com	1	8
3	15	Tina	Turner	2000-01-06	2020-06-12	99 Forest Hill	Cardiff	CF24 6PA	7582289641	tinytina@gmail.com	1	15
4	18	Callum	Williams	1996-03-25	2021-02-11	5 Test Street	Cardiff	CF23 7SS	7848759345	caffromcardiff@hotmail.com	1	18

	CustomerID	FirstName	Surname	DateOfBirth	DateJoined	StreetAddress	City	Postcode	ContactNumber	EmailAddress	FacilityID	PaymentID
1	2	Paul	Jones	1963-06-19	2020-01-02	12 Albert Road	Swansea	SA4 8PT	7698557436	PaulJ99@outlook.com	2	2
2	9	Rachel	Heame	1989-10-30	2020-10-19	8 Bensington Road	Caerphilly	CF72 9ST	7415297863	rachheame@gmail.com	2	9
3	16	Gwen	Jones	2000-12-25	2020-01-03	83 Bush Drive	Swansea	SA71 9DS	7885934894	gwenjones@gmail.com	2	16

Searching for staff based on the region/gym facility:

Using the FacilityID for the WHERE clause will display staff members based in that facility/region. A gym in Cardiff will be looking at staff from Cardiff and therefore it is logical to use horizontal fragmentation for this scenario.



```
USE ShapeUpGyms
SELECT * INTO CARDIFFSTAFF FROM Staff WHERE FacilityID = '1'
SELECT * INTO SWANSEASTAFF FROM Staff WHERE FacilityID = '2'

SELECT * FROM CARDIFFSTAFF
SELECT * FROM SWANSEASTAFF
```

	StaffID	FirstName	Surname	DateOfBirth	DateJoined	StreetAddress	City	Postcode	ContactNumber	EmailAddress	FacilityID
1	1	Arnold	Swamy	1968-02-14	2019-01-01	7 Test Street	Cardiff	CF4 0HF	7206497884	arnoldswamy@shapeup.com	1
2	2	Louise	Belcha	1990-03-07	2019-01-05	9 Claude Road	Cardiff	CF24 3PZ	7976431844	louisebelcha@shapeup.com	1

	StaffID	FirstName	Surname	DateOfBirth	DateJoined	StreetAddress	City	Postcode	ContactNumber	EmailAddress	FacilityID
1	3	Charlie	Cook	1974-11-02	2019-02-16	19 Dillon Lane	Swansea	SA19 7SE	7489687226	charliecook@shapeup.com	2
2	4	Caleb	Hooper	1991-09-17	2019-02-10	10 Belmont Road	Swansea	SA22 8JW	74286398441	calebhooper@shapeup.com	2

Searching for gym class schedule for the region/gym facility:

Using the FacilityID for the WHERE clause will display the gym classes based in that facility/region. Much of the time, a gym in Cardiff will look at the class schedule of the Cardiff gym and therefore it is logical to use horizontal fragmentation for this scenario.

SQLQuery9.sql - DES...la Passariello (55)) SQLQuery3.sql - DES...la Passariello (60))* SQLQuery2.s

USE ShapeUpGyms

SELECT * INTO CARDIFFCLASSCHEDULE FROM GymClass WHERE FacilityID = '1'

SELECT * INTO SWANSEACLASSCHEDULE FROM GymClass WHERE FacilityID = '2'

SELECT * FROM CARDIFFCLASSCHEDULE

SELECT * FROM SWANSEACLASSCHEDULE

100 %

Results

Messages

	ClassID	ClassName	ClassDay	ClassTime	MaximumCapacity	FacilityID	StaffID
1	1	Pilates	Tuesday	09:00:00.0000000	30	1	2
2	2	Absolute Abs	Monday	13:00:00.0000000	20	1	1

	ClassID	ClassName	ClassDay	ClassTime	MaximumCapacity	FacilityID	StaffID
1	6	High Intensity Cardio	Friday	10:00:00.0000000	10	2	3

Vertical Fragmentation

The Payment table is designed to best suit the vertical fragmentation design as per the following examples.

Blocking the view of confidential payment information:

As payment details are stored in the database, there may be users whose access does not allow for the viewing of such confidential data. By using vertical fragmentation, the payment table can show only the PaymentID whilst the hiding sensitive data columns such as AccountHolder, CardNumber, SecurityCode, and CardExpirationDate.

SQLQuery3.sql - DES...la Passariello (60))* SQLQuery2.sql - not connected

SELECT PaymentID

INTO VERTPAYMENT

FROM Payment

SELECT * FROM VERTPAYMENT

100 %

Results

Messages

	PaymentID
1	4
2	2
3	13
4	9
5	11
6	18
7	7
8	12
9	8
10	6
11	16
12	17
13	10
14	1
15	5
16	3
17	14
18	15

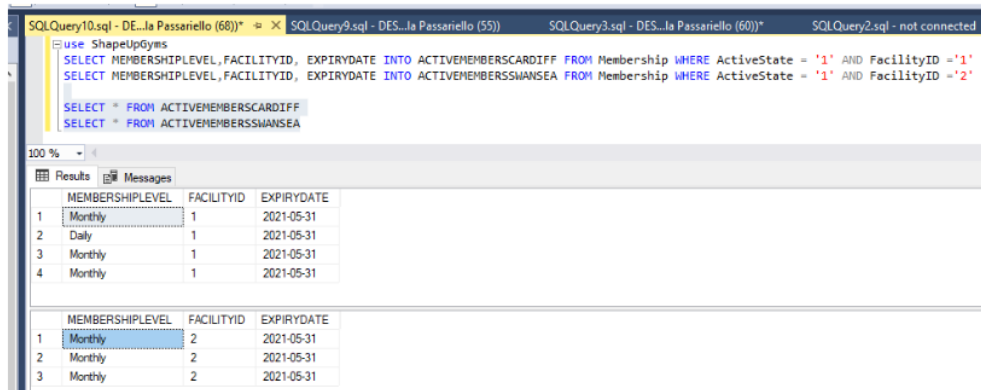
Using vertical fragmentation allows for data to be hidden from those who should not be accessing it and can also trim data for those who do not need to see certain columns.

Mixed Fragmentation

The Membership table is designed to best suit a mixed fragmentation as per the following examples.

Financing department to view active memberships by region/gym facility:

The finance department may wish to view how many active members each region/gym facility may currently have but will not need to see further data such as the DateCreated, CustomerID, or PaymentID. A mixed fragmentation can provide this data utilising both the horizontal and vertical relations.



```
SQLQuery10.sql - DE...la Passariello (68)) * SQLQuery9.sql - DE...la Passariello (55)) SQLQuery3.sql - DE...la Passariello (60)) * SQLQuery2.sql - not connected
```

```
use ShapeUpGyms
SELECT MEMBERSHIPLEVEL, FACILITYID, EXPIRYDATE INTO ACTIVEMEMBERSCARDIFF FROM Membership WHERE ActiveState = '1' AND FacilityID = '1'
SELECT MEMBERSHIPLEVEL, FACILITYID, EXPIRYDATE INTO ACTIVEMEMBERSSWANSEA FROM Membership WHERE ActiveState = '1' AND FacilityID = '2'

SELECT * FROM ACTIVEMEMBERSCARDIFF
SELECT * FROM ACTIVEMEMBERSSWANSEA
```

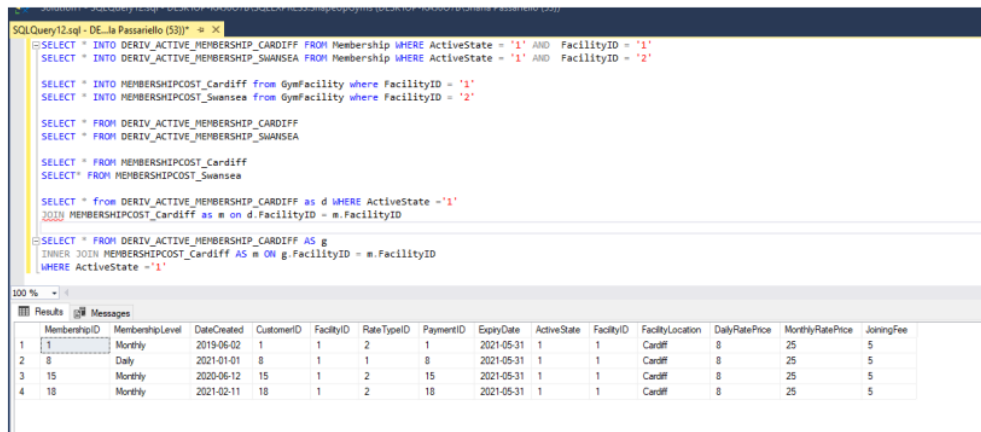
	MEMBERSHIPLEVEL	FACILITYID	EXPIRYDATE
1	Monthly	1	2021-05-31
2	Daily	1	2021-05-31
3	Monthly	1	2021-05-31
4	Monthly	1	2021-05-31

	MEMBERSHIPLEVEL	FACILITYID	EXPIRYDATE
1	Monthly	2	2021-05-31
2	Monthly	2	2021-05-31
3	Monthly	2	2021-05-31

Derived Fragmentation

It is required in the case study that each gym may set their own memberships prices. If the financing department wanted to view the active members and the prices they were paying per region/gym facility, a derived fragmentation could be used as per the following example.

Financing Department to view active members and how much they cost by region/gym facility:



```
SQLQuery12.sql - DE...la Passariello (53)) * *
```

```
SELECT * INTO DERIV_ACTIVE_MEMBERSHIP_CARDIFF FROM Membership WHERE ActiveState = '1' AND FacilityID = '1'
SELECT * INTO DERIV_ACTIVE_MEMBERSHIP_SWANSEA FROM Membership WHERE ActiveState = '1' AND FacilityID = '2'

SELECT * INTO MEMBERSHIPCOST_CARDIFF FROM GymFacility WHERE FacilityID = '1'
SELECT * INTO MEMBERSHIPCOST_SWANSEA FROM GymFacility WHERE FacilityID = '2'

SELECT * FROM DERIV_ACTIVE_MEMBERSHIP_CARDIFF
SELECT * FROM DERIV_ACTIVE_MEMBERSHIP_SWANSEA

SELECT * FROM MEMBERSHIPCOST_CARDIFF
SELECT * FROM MEMBERSHIPCOST_SWANSEA

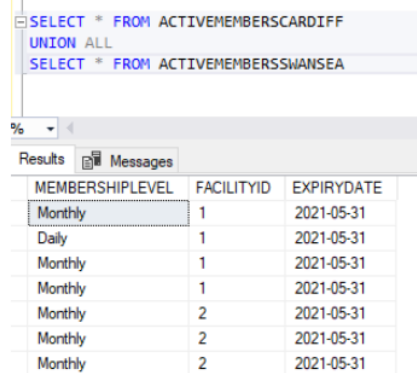
SELECT * FROM DERIV_ACTIVE_MEMBERSHIP_CARDIFF AS d WHERE ActiveState = '1'
JOIN MEMBERSHIPCOST_CARDIFF AS m ON d.FacilityID = m.FacilityID

SELECT * FROM DERIV_ACTIVE_MEMBERSHIP_CARDIFF AS g
INNER JOIN MEMBERSHIPCOST_CARDIFF AS m ON g.FacilityID = m.FacilityID
WHERE ActiveState = '1'
```

MembershipID	MembershipLevel	DateCreated	CustomerID	FacilityID	RateTypeID	PaymentID	ExpiryDate	ActiveState	FacilityID	FacilityLocation	DailyRatePrice	MonthlyRatePrice	JoiningFee
1	Monthly	2019-06-02	1	1	2	1	2021-05-31	1	1	Cardiff	8	25	5
2	Daily	2021-01-01	8	1	1	8	2021-05-31	1	1	Cardiff	8	25	5
3	Monthly	2020-06-12	15	1	2	15	2021-05-31	1	1	Cardiff	8	25	5
4	Monthly	2021-02-11	18	1	2	18	2021-05-31	1	1	Cardiff	8	25	5

Reconstruction

Table reconstruction of fragmentation can be executed using the UNION ALL function in SQL. This will bring back all values including duplicates. The example below shows reconstruction of the data for ACTIVEMEMBERSCARDIFF and ACTIVEMEMBERSWANSEA that were fragmented in the *Mixed Fragmentation* section:



```
SELECT * FROM ACTIVEMEMBERSCARDIFF
UNION ALL
SELECT * FROM ACTIVEMEMBERSWANSEA
```

MEMBERSHIPLEVEL	FACILITYID	EXPIRYDATE
Monthly	1	2021-05-31
Daily	1	2021-05-31
Monthly	1	2021-05-31
Monthly	1	2021-05-31
Monthly	2	2021-05-31
Monthly	2	2021-05-31
Monthly	2	2021-05-31

Performance Tuning

As data is being created and modified, it is important to analyse ways in which performance can be improved or kept as efficient as possible. A distributed database solution is the first step to helping query performance as spreads the data so that not every single record has to be read when a query is executed. For example, if Shape Up Gyms consisted of 1000 customers with only 200 of those customers registered to the Swansea region/gym facility, when searching for a customer from the Swansea region, as the Swansea records are going to be distributed to that location then the query being executed will only have to look through 200 records. However, in a centralised solution it would be required to look through all 1000 records in search of Swansea customers. The distributed database solution returns results more quickly than a centralised solution.

The design of the tables will also contribute to the performance of the database. Indexes are one of the best tools for retrieving records quickly. An index is a structure in the table used to identify and group records to target queries in a more time efficient manner. This prevents the SQL engine needing to read through all records and helps to find the path of least resistance, which can significantly improve query performance.

Clustered Indexes

In a clustered index the data is sequentially ordered by the specified field. When a primary key is created in an SQL table, this automatically becomes the clustered index for that table - a table may have only one clustered index.

All tables provided in the Shape Up Gyms database were given a primary key which was set as the clustered index. This allows for an accelerated retrieval of data where there may be columns that contain the same data results, such as customer names in the Customer table. By providing each customer with their own unique ID, this allows for faster retrieval as it will look and retrieve the one record as opposed to pulling back multiple customers who may share the same name.

Secondary Indexes

Adding a secondary index can also help retrieve data more efficiently. A secondary index will group and sort records by the chosen indexed columns. This allows the SQL engine to directly go to these grouped records instead of having to scan the whole table, thus returning results more efficiently.

In the Shape Up Gyms database, a secondary index would be useful where data input is large into tables and where JOIN queries will most likely be executed. For this database, secondary indexes were created for the following scenarios:

- Queries to retrieve customers by their names; indexing them on the Surname and FirstName columns as users will likely query for customers in the database. This may be prevalent with queries such as looking at customers located in specific regions/gym facilities, or searching for customers in relation to the payment table
- Queries to retrieve staff by their names; indexing them by the Surname and FirstName columns as users will likely query customers in the database by their names. This may be prevalent with queries such as looking at staff located in specific regions/gym facilities, or searching for staff in relation to the GymClass table
- Queries to retrieve memberships in specific regions/gym locations; indexing them by the FacilityID column on the Membership table to group memberships by the facility location

Secondary indexes should avoid columns that are frequently updated. As prices and membership details will likely change often, it was more suitable that secondary indexes remain on the FacilityID on the Membership table, and the Surname and FirstName columns of customers and staff, as names are unlikely to change as often as membership details. If indexes are formed on columns that often change, the indexes will need to be rebuilt.

Analysis of the database should continue throughout its lifecycle so that indexes can be monitored and tweaked to improve data retrieval more efficiently. As data is modified, the secondary indexes may no longer be efficient to the database and cause performance issues and therefore regular monitoring of index performance is important.

As well as indexing, query optimisation can produce a better return performance from the database. Executing more specific queries such as using SELECT Column1, Column2... instead of SELECT * FROM and adding in a WHERE statement can prevent the database from pulling back a lot of unwanted data, improving query response time.

Conclusion

In conclusion, combining a distributed database solution and indexing on tables that will be populated with large quantities of data and targeted frequently with JOIN statements can help with database performance for Shape Up Gyms databases. Database monitoring should become a regular task for those maintaining the database to analyse queries and response times so that indexes can be created and modified to efficiently retrieve the data.

References

- [1]A. Onsman, "Centralized Database Management System", *Tutorialspoint.com*, 2021. [Online]. Available: <https://www.tutorialspoint.com/Centralized-Database-Management-System>. [Accessed: 09- May- 2021].
- [2]S. Kandasamy, "Advantages and Disadvantages of Distributed Databases", *Exploredatabase.com*, 2021. [Online]. Available: [https://www.exploredatabase.com/2014/08/advantages-and-disadvantages-of-distributed-databases.html#:~:text=Increased%20reliability%20and%20availability%20%E2%80%93,to%20some%20sites%20\(servers\)](https://www.exploredatabase.com/2014/08/advantages-and-disadvantages-of-distributed-databases.html#:~:text=Increased%20reliability%20and%20availability%20%E2%80%93,to%20some%20sites%20(servers)). [Accessed: 08- May- 2021].
- [3]S. Schwiderski, "Monitoring the behaviour of distributed systems", University of Cambridge, 1996.
- [4]"Distributed DBMS - Database Environments - Tutorialspoint", *Tutorialspoint.com*, 2021. [Online]. Available: https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_database_environments.htm. [Accessed: 08- May- 2021].
- [5]B. Petrovic and V. &arr;, "SQL index overview and strategy", *SQL Shack - articles about database auditing, server performance, data recovery, and more*, 2021. [Online]. Available: <https://www.sqlshack.com/sql-index-overview-and-strategy/#:~:text=A%20SQL%20index%20is%20used,users%20need%20to%20search%20frequently>. [Accessed: 09- May- 2021].
- [6]E. Otoo and K. Wu, "Accelerating Queries on Very Large Datasets*", Lawrence Berkeley National Laboratory, 2021.
- [7]S. Mukherjee, "Indexes in Microsoft SQL Server", Chicago.
- [8]"Indexing in Databases | Set 1 - GeeksforGeeks", *GeeksforGeeks*, 2019. [Online]. Available: <https://www.geeksforgeeks.org/indexing-in-databases-set-1/>. [Accessed: 09- May- 2021].

Databases and Application Development Report - Sharla Passariello

GRADEMARK REPORT

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

Very good design of database schema. You have used this well within the SQL environment to build all of the tables. You have a good understanding of design and testing you have provided and demonstrated the inserting of data and tested out you overall structure. Very good analysis and documentation of distributed database requirement. You have used you table and SQL weel to demonstrate how this would work. A good overall structure to your report.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11