
Shape up Gyms Database

Database Management Systems
University of Wales Trinity Saint David

Siobhan Stephens

1800709

Abstract

Shape up Gyms provides access to modern gyms and affordable membership plans. The company began as a small business operating out of Swansea but their success has led them to expand their services to other parts of Wales. To support their growing customer base and improve services, they are in the process of upgrading their IT systems.

This report outlines the suggested performance enhancements and a distributed database fragmentation schema based on the current centralised Up Gyms database system, to aid in their growing customer base and regional based gym management system, whilst allowing their finance department and head office to remain in Swansea.

Table of Contents

Abstract	2
1. Introduction	4
2. Performance Tuning	4
2.1. Clustering Indexes	4
2.2. Secondary Indexes	4
3. Distributed Database Design	6
3.1. Fragmentation Discussion	6
3.2. Fragmentation Schema	6
3.3. Allocation and Replication	9
4. Conclusion	10
5. References	11

1. Introduction

This report will outline suggested performance tuning enhancements along with a proposed distributed database design for Shape Up Gyms, based on previously provided requirements. The main requirements are noted below.

- a. Implement distributed database model
- b. Distribute customer base and gym membership system across regional areas
- c. Finance department remains in Swansea

The centralised database structure has been provided and implemented, and is outlined in Appendix A.

2. Performance Tuning

The following section outlines how indexes could be used by Shape Up Gyms to improve the performance of their database queries.

2.1. Clustering Indexes

Clustered indexes are used to order the rows in a given table, allowing the data to be sorted, therefore aiding in the efficient and quick access to data.

Customer information will be required frequently by Shape Up Gyms across all sites (Gyms and finance office), by both the customers themselves and by the staff. This access will likely be spread across the day, with no specific peak times.

Shape Up Gyms Staff may choose to use views such as 's_all_customers' (SQL File 2) to search for all customer information, sorted by last name in descending order. As a result, to speed up this query it would be suitable to add a clustering index on the Customer.LastName field.

CityName from the City table is also often queried, in order to find Customer's home address city or a gym's city, and so, an ordered clustered index should be used, sorted by CityName in descending order.

2.2. Secondary Indexes

In addition to the afore mentioned clustering indexes, secondary indexes should be added to tables and fields which are often queried using attributes other than the primary keys, allowing the DBMS to access data more efficiently on non-ordered tables.

However, secondary indexes should not be used on fields which are commonly updated, due to the overhead of maintaining these indexes, which would degrade update performance.

In order to decide on the necessary secondary indexes, the author considered the most common queries that would be run on the system, along with how those fields would be used and how an improvement in performance would affect those fields.

The most important transaction in the system is to create a new membership purchase (NewMembershipPurchase stored procedure). This relies on data from other tables linked by foreign keys, and contains inner SQL SELECT statements to the MembershipType table to gather information that must be stored in MembershipPurchase.

The author concluded that fields from the MembershipType table would be used frequently, to display all fixed-term memberships, all day passes, and to find contracts of a specific duration, such as 6 months. DurationType and Contract type are also essential to the creation of new MembershipPurchase records. These fields are likely to be used throughout the day, with peak times in the morning, when people visit the gym before work, and in the evening, when people visit after work. As a result, both these fields should be indexed.

The MembershipPurchase table's Status field is also often used in querying, such as to find all 'Active' or 'Elapsed' memberships. As this field is not often updated, it is also a good candidate for a secondary index.

In addition, OneTimeFee, JoiningFee and MonthlyPaymentFee from the GymMembershipPrice table will be frequently accessed when creating a new MembershipPurchase, as their values will be queried and stored in the MembershipPurchase table so that the historical price can be stored. As a result, to increase the performance of the create MembershipPurchase stored procedure these fields should also be indexed.

3. Distributed Database Design

According to M. T. Özsu et al, a distributed database is a collection of multiple, logically interrelated databases located at the nodes of a distributed system. A distributed database management system (distributed DBMS) is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users [1].

Many organisations, including Shape Up Gyms, are physically divided across multiple locations, with each location requiring its own set of local data, thus to maintain performance and ease of access the overall database should be distributed. However, the multiple locations will require shared data, and so there is a requirement to ensure the necessary tables remain synchronised.

3.1. Fragmentation Discussion

3.1.1. Advantages

Applying fragmentation to a distributed database allows data to be stored close to where it is most frequently used, aiding in the performance of querying, along with minimising communication costs and improving reliability and availability. Also, data that is not needed by the local application is not stored, minimising unnecessary storage costs.

3.1.2. Disadvantages

Global applications that require data from several of the fragments, which are physically separated across different sites can suffer from decreased performance. Controlling the integrity of data may also be more difficult if the data and their functional dependencies are separated across different sites.

In addition, maintaining consistent data is more difficult when a data item changes and needs to move from one fragment to another.

3.2. Fragmentation Schema

Three rules must be followed to ensure a successful fragmentation schema.

Completeness

If Relation R is split into fragments $R_1, R_n \dots$, each data found in R must appear in at least one fragment. This will ensure there is no data loss [2].

Reconstruction

In order to ensure functional dependencies are preserved, it must be possible to reconstruct the original relation from the fragments using relational operations.

Disjoint

Finally, to ensure there is only minimal data redundancy, it is important that if data item d_i appears in fragment R_i , then it should not appear in any other fragment.

In order to decide on the most appropriate fragmentation schema for Shape Up Gyms, both quantitative and qualitative information has been considered, including the number of estimated queries, peak times, who will rely on the information, and the data's importance [3].

3.2.1. City Table

It has been decided not to fragment the City table. This is because the table contains only a limited number of rows and will be required across for use by multiple tables and queries, across multiple physical sites including the finance department in Swansea along with the Gym branches across the country.

Fragmentation would not improve performance of the database, and in fact could slow performance as the data would not be readily available at the required site.

3.2.2. Region Table

It has been decided not to fragment the Region table, for the reasons given above in 2.1.1.

3.2.3. MembershipType Table

It has been decided not to fragment the MembershipType table, for the reasons given above in 2.1.1.

3.2.4. GymMembershipPrice Table

The GymMembership price table stores the prices set by individual gyms for each membership type. This table will be used frequently by individual gyms when displaying prices, and also when creating a new membership purchase.

It is not required that each gym has access to other gym's prices, and therefore it would be most suitable to fragment the table into each Gym.

Therefore, it is suggested to use primary horizontal fragmentation for GymMembershipPrice with minterm predicates. The chosen horizontal fragments are:

GMP_1	GymID = 'G000001'
GMP_2	GymID = 'G000002'
GMP_3	GymID = 'G000003'
GMP_4	GymID = 'G000004'
GMP_5	GymID = 'G000005'
GMP_6	GymID = 'G000006'

Reconstruction: $GMP_1 \cup GMP_2 \cup GMP_3 \cup GMP_4 \cup GMP_5 \cup GMP_6$

3.2.5. Customer Table

The Customer table stores personal, gym and bank details for each customer. This table will be used by both the finance department and personnel employees at the customer's own gym.

As above, it is not required that each gym has access to every customer, instead it should only have access to its own customers. This will reduce replication, improve query times and will improve data integrity as only those who should have access to the data, do have access.

The finance department does not require full customer details to process the payments and so they have only limited columns.

Therefore, it has been suggested that the customer table be decomposed using vertical fragmentation:

C_1 (Personnel): $\Pi_{CustomerID, GymID, FirstName, LastName, HomeAddress, HomeCityCode, PostCode, DateOfBirth, ContactPhoneNo, MedicalSummary}(Customer)$

C_2 (Finance): $\Pi_{CustomerID, BankAccountNo, SortCode}(Customer)$

Then, use derived fragmentation on fragment C_1 , which is based on GymID to locate customer information at the same location as the Gym's location fragment:

$$C_{1i}: C_1 \triangleright_{GymID} GMP_i \quad 1 \leq i \leq 6$$

Reconstruction: $(C_{11} \cup C_{12} \cup C_{13} \cup C_{14} \cup C_{15} \cup C_{16}) \bowtie_{CustomerID} C_2$

3.2.6. MembershipPurchase Table

The MembershipPurchase table is a transaction table used to store details of each membership purchase, and links to many tables such as Customer, MembershipType, GymMembershipPrice etc.

The most common use case of this table is for each gym to find a record of all its memberships, and therefore the table should be decomposed into the records for each Gym.

Therefore, it has been decided to use primary horizontal fragmentation with minterm predicates. Proposed horizontal fragments are:

MP ₁	Customer.GymID = 'G000001'
MP ₂	Customer.GymID = 'G000002'
MP ₃	Customer.GymID = 'G000003'
MP ₄	Customer.GymID = 'G000004'
MP ₅	Customer.GymID = 'G000005'
MP ₆	Customer.GymID = 'G000006'

Reconstruction: $MP_1 \cup MP_2 \cup MP_3 \cup MP_4 \cup MP_5 \cup MP_6$

In addition, the finance department will require access to the table to process payments, produce invoices etc. For this, it is suggested that vertical fragmentation be used to provide information for the finance department on a per gym basis.

MP_{i1} (Finance): $\Pi_{PurchaseNo, CustomerID, MembershipTypeID, StartDate, EndDate, Status, JoiningFeePrice, MonthlyFeePrice, OneTimeFeePrice}(MembershipPurchase)$

3.2.7. Gym Table

The Gym table stores name, location and contact information for each of the Shape Up Gym sites.

This information may be used by other local gyms, and therefore, it is suggested that primary horizontal fragmentation is used with minterm predicates, to separate the Gyms by Region. The suggested horizontal fragments are as follows:

G ₁	Region.RegionName = 'Wales'
G ₂	Region.RegionName = 'West Midlands'
G ₃	Region.RegionName = 'South West'
G ₄	Region.RegionName = 'South East'
G ₅	Region.RegionName = 'London'
G ₆	Region.RegionName = 'North East'
G ₇	Region.RegionName = 'North West'

Where Region is linked via an inner join to the Gym table (Gym.CityCode = City.CityCode, City.RegionCode = Region.RegionCode)

Reconstruction: $G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$

3.3. Allocation and Replication

It is suggested that the distributed database for Shape Up Gyms have selective replication, with the following proposed for each table.

- a. Region

- Table to replicate across each site
- b. City
 - Table to replicate across each site
- c. Gym
 - Each fragment to be stored at necessary site
- d. MembershipType
 - Table to replicate across each site
- e. GymMembershipPrice
 - Each fragment to be stored at necessary site
- f. Customer
 - Each fragment to be stored at necessary site
- g. MembershipPurchase
 - Each fragment to be stored at necessary site

As it is proposed that Region, City and MembershipType are not fragmented, they must be replicated across each site. This will offer the best read performance, which is important for these tables given their estimated usage. Despite replicating across each site, as the tables contain data that will not frequently be updated, and there are only limited numbers of records, the storage costs should remain low. If the record set was to grow significantly, this design may be reconsidered [4].

The other fragments should be split across all appropriate sites by Region/City/Gym. This will offer Shape Up Gyms the best-read performance as the data is closest to the site, along with low storage costs. In addition, data integrity will be improved with sites not having access to another's data [5].

4. Conclusion

The proposed indexing and fragmentation solutions should allow Shape Up Gyms to maintain a reliable, well performing distributed database, so that they can continue to expand their business across new sites, whilst maintaining the best cost/performance balance.

In the future, if the business needs are to change or the record set is to expand significantly, it would be suitable to redesign the proposed solution.

5. References

- [1] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*. Springer International Publishing, 2020.
- [2] D. J. Reid and M. Orłowska, "Allocating relations in a distributed database system," *Math. Comput. Model.*, vol. 22, no. 8, pp. 33–47, Oct. 1995.
- [3] Ozsu and P. Valduriez, "Distributed Database Systems." [Online]. Available: <https://www.comp.nus.edu.sg/~tankl/cs5225/2008/design6.pdf>. [Accessed: 13-Mar-2020].
- [4] "Distributed DBMS - Distributed Databases - Tutorialspoint." [Online]. Available: https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_databases.htm. [Accessed: 13-Mar-2020].
- [5] A. Singh and K. Singh Kahlon, "Nonreplicated Static Data Allocation in Distributed Databases Using Biogeography-Based Optimization," 2014.

Appendix A

Region

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
RegionCode	Unique ID of the region	Primary	CHAR(5)	5 Characters	No	Yes	R0002	LIKE 'R[0-9][0-9][0-9][0-9]'
RegionName	Name of the region		VARCHAR(90)	Up to 90 Characters	No	Yes	South West	Length < 90 and Length > 2

City

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
CityCode	Unique ID of the City	Primary	CHAR(7)	7 Characters	No	Yes	CC00056	LIKE 'CC[0-9][0-9][0-9][0-9][0-9]'
CityName	The name of the city		VARCHAR(90)	Up to 90 Characters	No	Yes	Bristol	Length >2 and < 90
CountyName	The name of the county		VARCHAR(90)	Up to 90 Characters	No	No	Gloucestershire	Length >2 and < 90
RegionCode	Foreign key to the Region table	Foreign	CHAR(5)	5 Characters	No	No	R0174	

Gym

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
GymBranchID	The unique ID of the gym branch	Primary	CHAR(7)	7 Characters	No	Yes	G000386	LIKE 'G[0-9][0-9][0-9][0-9][0-9]'
Name	The name of the gym		VARCHAR(70)	Up to 70 Characters	No	No	Keep Fit Gym	Length < 70 and length > 2
Address	First line of the address		VARCHAR(150)	Up to 150 Characters	No	No	45 Hunter Road	Length < 150 and length > 3
CityCode	The gym's city address (foreign key to city)	Foreign	CHAR(7)	7 Characters	No	No	CC09876	
PostCode	The gym's postcode		VARCHAR(8)	Up to 8 Characters	No	No	SA12 3ND	LIKE '[A-Z][A-Z][0-9][A-Z] [0-9][A-Z][A-Z]' OR LIKE '[A-Z][0-9][A-Z] [0-9][A-Z][A-Z]' OR LIKE '[A-Z][0-9][0-9][A-Z][A-Z]' OR LIKE '[A-Z][0-9][0-9][0-9][A-Z][A-Z]' OR LIKE '[A-Z][A-Z][0-9][0-9][A-Z][A-Z]' OR LIKE '[A-Z][A-Z][0-9][0-9][0-9][A-Z][A-Z]'
TelephoneNo	The gym's telephone number (multiple telephone formats)		VARCHAR(17)	Up to 17 Characters	No	No	07488345234	LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

MembershipType

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
MembershipTypeID	The unique ID of the membership type	Primary	CHAR(7)	7 Characters	No	Yes	MT00129	LIKE 'MT[0-9][0-9][0-9][0-9]'
PackageName	The name of the membership package		VARCHAR(70)	Up to 70 Characters	No	No	New 12 Month Member	Length < 70 and >2
Description	The description of the membership package		VARCHAR(220)	Up to 220 Characters	No	No	New member only. 12 month membership at any branch.	Length < 220 and >2
ContractType	The type of contract		VARCHAR(30)	Up to 30 Characters	No	No	Fixed-Term OR Rolling	LIKE 'Fixed-Term' OR LIKE 'Rolling'
DurationType	The duration type of the contract (Year, Month etc)		VARCHAR(30)	Up to 30 Characters	No	No	Month OR Year	LIKE 'Year' OR LIKE 'Month'
DurationQuantity	The quantity of the duration, eg 2 (years)		INTEGER(3)		No	No	36	Integer
OffPeakMembership	If the membership is off peak (yes or no)		BIT	1 Integer	No	No	0 OR 1	1 OR 0
CancelAnytime	If the membership can be cancelled at any time (yes or no)		BIT	1 Integer	No	No	0 OR 1	1 OR 0

GymMembershipPrice

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
MembershipTypeID	The unique ID of the membership type - foreign key to MembershipType	Composite Primary, Foreign	CHAR(7)	7 Characters	No	No	MT00129	
GymID	The unique ID of the membership type - foreign key to MembershipType	Composite Primary, Foreign	CHAR(7)	7 Characters	No	No	G000386	
OneTimePaymentFee	The joining fee amount if paid as a one-off payment		DECIMAL(19, 2)		No	No	110.99	>= 0 AND <= 9999
MonthlyPaymentFee	The fee amount if paid monthly		DECIMAL(19, 2)		No	No	19.99	>= 0 AND <= 9999
JoiningFee	The fee amount for joining		DECIMAL(19, 2)		No	No	15	>= 0 AND <= 9999

Customer

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
CustomerID	The unique ID for the customer	Primary	CHAR(8)	8 Characters	No	Yes	C0000735	REGEXP '^C[0-9]{7}\$'
GymID	The gym they attend (References GymGymBranchID)	Foreign	CHAR(7)	7 Characters	No	No	G000386	
FirstName	Customer's first name		VARCHAR(90)	Up to 90 Characters	No	No	John	length <= 90 and >= 1
LastName	Customer's last name		VARCHAR(90)	Up to 90 Characters	No	No	Higgins	length <= 90 and >= 1
DateOfBirth	Customer's date of birth		DATE		No	No	1989-04-06	Age >= 16
EmailAddress	Customer's email address		VARCHAR(320)	Up to 320 Characters (up to 64 for local, 1 for @, up to 255 for domain)	No	Yes	john.higgins@emailprovider.com	LIKE '%____@____%' and length < 320
Passcode	Used to enter the gym (4 digit key code)		INT(4)	4 Digits	No	No	9834	LIKE '[0-9][0-9][0-9][0-9]'
HomeAddress	The first line of their address		VARCHAR(220)	Up to 220 Characters	No	No	34 Kings Lane	length <220 characters and length > 2
HomeCityCode	Their city (address)	Foreign	CHAR(7)	7 Characters	No	No	CC00056	
Postcode	Their postcode		VARCHAR(8)	Up to 8 Characters	No	No	SA12 3ND	PostCode LIKE '[A-Z][A-Z][0-9][A-Z][0-9][A-Z][A-Z]' OR PostCode LIKE '[A-Z][0-9][A-Z][0-9][A-Z][A-Z]' OR PostCode LIKE '[A-Z][0-9][0-9][A-Z][A-Z]' OR PostCode LIKE '[A-Z][0-9][0-9][A-Z][A-Z]' OR PostCode LIKE '[A-Z][A-Z][0-9][0-9][A-Z][A-Z]' OR PostCode LIKE '[A-Z][A-Z][0-9][0-9][A-Z][A-Z]'
ContactPhoneNo	The telephone number of their emergency contact		VARCHAR(17)	Up to 17 Characters	No	No	07488345234	LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
MedicalSummary	Details of their medical history/conditions if applicable		VARCHAR(500)	Up to 500 characters	Yes	No	Type 2 Diabetes OR null	Length < 500
JoinDate	The date the customer first joined (first gym/membership)		DATE		No	No	02/05/2019	
BankAccountNo	The UK bank account number (7 digit accounts left pad 0)		CHAR(8)	8 Numeric Characters	No	No	3456753	LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
SortCode	The UK sort code (format 00-00-00)		CHAR(8)	8 Characters	No	No	23-41-93	LIKE '([0-9][0-9]-[0-9][0-9]-[0-9][0-9])'

MembershipPurchase

Field Name	Description	Key	Data Type	Length	Nullable?	Unique?	Valid Value Example	Constraints
PurchaseNo	The unique ID for the purchase	Primary	CHAR(9)	9 Characters	No	Yes	P00004572	LIKE 'P[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
CustomerID	The unique ID of the customer	Foreign	CHAR(8)	8 Characters	No	No	C0000735	
MembershipTypeID	The unique ID of the membership type	Foreign	CHAR(7)	7 Characters	No	No	MT00129	
StartDate	The membership start date		DATE	10 Characters	No	No	2020-01-04	Greater than or equal to todays date
EndDate	The membership end date		DATE	10 Characters	No	No	2022-01-04	Greater than StartDate
PurchaseDate	The date the membership was purchased		DATE	10 Characters	No	No	2020-01-02	Less than or Equal to StartDate
JoiningFeePrice	Joining fee at the time of purchase (from GymMembershipPrice)		DECIMAL(19, 2)		No	No	110.99	Between 0 and 9999
OneTimeFeePrice	One time fee at the time of purchase (from GymMembershipPrice)		DECIMAL(19, 2)		No	No	10.99	Between 0 and 9999
MonthlyFeePrice	Monthly fee at the time of purchase (from GymMembershipPrice)		DECIMAL(19, 2)		No	No	21.99	Between 0 and 9999
MonthlyPaymentDay	The day (digit) of the month the payment should be taken (if not valid roll to next valid date)		INTEGER(2)	2 digits	Yes	No	12	null or Between 1 and 31
LastPaymentDate	The last payment date calculated from the MonthlyPaymentDay and EndDate		DATE		No	No	01/01/2022	Greater than NOW and StartDate
Status	Whether or not the contract is active or not		VARCHAR(20)	Up to 20 Characters	No	No	'Cancelled' OR 'Active' OR 'Elapsed'	Must be 'Cancelled' OR 'Active' OR 'Elapsed'