

Assignment

by Steven Alexander Moon

FILE	17554_STEVEN_ALEXANDER_MOON_ASSIGNMENT_349706_1866899250.D OCX (128.15K)		
TIME SUBMITTED	04-MAY-2021 08:41AM (UTC+0100)	WORD COUNT	2314
SUBMISSION ID	151522907	CHARACTER COUNT	16853

Database and Application Development

Steven Moon (1904741)

Table of Contents

Section Title	Section Number	Page Number
Introduction	1.0	2
Data Query Requirements	2.0	2
Frequent Queries	2.1.0	2
Finance Queries	2.1.1	2
Gym Queries	2.1.2	3
Fragmentation	3.0	3
Gym Fragment	3.1.0	3
Customer Fragment	3.1.1	4
Remaining Fragments	3.1.2	4
Conclusion	4.0	5
References	5.0	5
Appendix	6.0	6
File 1	6.1	6
File 2	6.2	7
ERD	6.3	12

1.0 Introduction

The following report will look into the performance of the ShapeUp Gyms database and offer insights into performance improvements and identify the most commonly used data queries that are likely to be used within the database and offer a view of how this can be fragmented to improve speed of queries across the various sites.

2.0 Data Query Requirements

When looking at the requirements of ShapeUp Gyms, considerations need to be made for the type of information that is going to be required at the different locations and where the departments accessing this information will be located. The other requirement is to ensure that the routinely run queries operate at an acceptable speed for those users.

2.1.0 Frequent Queries

Below we will look at some of the queries that would be run by each department frequently and therefore need to be tuned to perform at acceptable levels.

2.1.1 Finance

The main finance office will be frequently looking at the overall income of the different locations and also how much as a company is being made per month or per quarter.

Given the above a frequent query may look as below:

```
SELECT m.LocationID, SUM(p.PaymentAmount)
FROM ShapeUpGym.dbo.Payment p
LEFT JOIN ShapeUpGym.dbo.Membership m on p.MembershipID=m.MembershipID
GROUP BY m.LocationID
```

This would allow the finance staff to see which gym locations are producing the most revenue for the company, the query could also be modified to include a date range for a reporting period. The above query would benefit from a index field to increase performance, this index would take place on the MembershipID and the payment amount fields. This is a secondary index that improves the speed of the query but would benefit aswell from the fragmentation schema discussed later in this paper. The other improvement that could be made for this query is the addition on a "MonthPaid" field that is then indexed to allow for the grouping of payments for quarterly returns and reporting on end of month figures.

2.1.2 Gym Queries

The individual gym locations will be interested to see the number of clients that are currently signed up to their location with an active membership and also those who have an elapsed membership so that they are able to try and contact those individuals to renew and generate additional income.

A query to return all those with active memberships may look something as below being limited to the individual's location.

```
SELECT MembershipID, CustomerID, LocationID, EndDate  
FROM ShapeUpGym.dbo.Membership  
WHERE LocationID = '1'  
and EndDate > GETDATE()
```

This query benefits from having the primary key of the table (MembershipID) being indexed and would not require additional secondary indexes to improve access as all information is contained within the same table structure.

If we consider looking for those memberships that have expired in order for the team to contact them then we would join on the CustomerID field which would then benefit from having an index placed on its primary key. This has been implemented within the database structure presented in this paper.

3.0 Fragmentation

Another key element to the database management systems that allow for improved performance is the use of a fragmentation design that places information closer to user who will frequently use the information by storing separate fragments of the overall database on different network machines [1]. There are many benefits to the use of fragmentation in large databases and these include efficiency and security of data as it is stored in smaller fragments based on need and only stored at locations that have security rights to access [2]. There are also some downsides to performance when using a fragmented model which affects those applications or queries that use information from a number of fragments stored in different areas of the network.

3.1.0 Gym Fragment

When looking at the frequent query used by the individual gym location then the implementation of horizontal fragmentation makes the most logical sense as the Membership table can be easily split "WHERE" each gym location holds its own membership details. This would mean all information about a gym's membership are held within its local network area rather than being accessed

across the internet. The global table can then be re-created using a simple union of all the different fragments.

There is a downside to this method of fragmentation for the finance team when compiling information for all sites as the query would need to access fragments distributed across the wider network.

3.1.1 Customer Fragment

When we consider the best way to fragment the customer table we need to consider the use case for the information across the organisation, this leads to the need for a mixed fragmentation model to allow for personal information such as email address and name to be replicated across the individual gym locations.

Given the above the recommended fragmentation model would be firstly vertical with the below columns stored at the Swansea office

CustomerID,DebitDetails,DebitAmount

The following would then be stored locally at the gym locations using replication to allow quick access to the personal information

CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile

This type of fragmentation would allow for quick access to the required information across all sites and for the information to be compiled using a simple join to the fragment stored at the gym location. The downside to this fragment design is some network overhead to replicate any changes across the network to each of the gym locations.

3.1.2 Remaining Fragments

The remaining tables within the database would be best stored centrally within the Swansea office as the table size is small and is unlikely to be searched frequently. This is true of the GymLocation table that is a simple lookup that would only be required when creating additional views of the information which would then be fragmented in a different manner.

4.0 Conclusion

In summary of this report a well designed and implemented database management system allows for the ShapeUp Gyms business to run more effective queries and maintain a clearer dataset that is possible using a manual system or something based within excel. The implementation suggested in this report will allow for continued expansion and maintenance of the database system would be minimal given the current indices and fragmentation design.

5.0 References

- [1 A. H. A.-S. e. al, 05 2017. [Online]. Available:
] https://www.researchgate.net/publication/320634432_A_comparative_analysis_of_data_fragmentation_in_distributed_database. [Accessed 27 04 2021].
- [2 D. Thakur, "What is Fragmentation in Database? Advantages & Disadvantages of fragmentation,"
] [Online]. Available: <https://ecomputernotes.com/database-system/adv-database/fragmentation>.
[Accessed 27 04 2021].

6.0 Appendix

6.1 File 1

```
--Create ShapeUpGym database
CREATE DATABASE ShapeUpGym

GO

--Create Tables for Database

USE ShapeUpGym

--Creation of Customer table and required constraints
CREATE TABLE Customer(
    CustomerID          CHAR(6) NOT NULL CONSTRAINT PK_Customer PRIMARY KEY
    CONSTRAINT CHK_CustomerID CHECK (LEFT(CustomerID,1) = 'C' and (RIGHT(CustomerID,5)
between '10000' and '99999')),
    FirstName          VARCHAR(30) NOT NULL,
    Surname            VARCHAR(30) NOT NULL,
    DOB               DATE NOT NULL,
    Postcode          VARCHAR(10) CONSTRAINT CHK_PCDLen CHECK
(LEN(Postcode) >=6),
    Email            VARCHAR(100) NOT NULL CONSTRAINT CHK_Email CHECK
(Email like '%@%'),
    Mobile          VARCHAR(12) NOT NULL,
    DebitDetails    VARCHAR(20),
    DebitAmount     DECIMAL(4,2)
);

GO

--Creation of Location table
CREATE TABLE GymLocation(
    LocationID          INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_Location
PRIMARY KEY,
    City              VARCHAR(30) NOT NULL,
    Postcode          VARCHAR(10) NOT NULL CONSTRAINT CHK_PCDLenGym CHECK
(LEN(Postcode) >=6),
    JoiningFee        DECIMAL(5,2) NOT NULL,
    MonthlyFee        DECIMAL(5,2) NOT NULL,
    DailyFee          DECIMAL(5,2) NOT NULL,
);

GO

--Creation of Membership table
CREATE TABLE Membership(
    MembershipID        CHAR(6) NOT NULL CONSTRAINT PK_Membership PRIMARY KEY,
    CustomerID          CHAR(6) NOT NULL CONSTRAINT FK_Membership_Customer
FOREIGN KEY REFERENCES Customer(CustomerID),
    LocationID          INT NOT NULL CONSTRAINT FK_Membership_Location
FOREIGN KEY REFERENCES GymLocation(LocationID),
    MembershipType      VARCHAR(30) NOT NULL,
    MembershipLength    INT NOT NULL,
    EndDate            DATE NOT NULL,
    PaymentType         VARCHAR(7),
    RemainingAmount     DECIMAL (5,2)
);
```

GO

--Creation of Payment table

```
CREATE TABLE Payment(  
    PaymentID INT IDENTITY(1, 1) NOT NULL CONSTRAINT PK_Payment  
    PRIMARY KEY,  
    MembershipID CHAR(6) NOT NULL CONSTRAINT FK_Payment_Membership FOREIGN  
    KEY REFERENCES Membership(MembershipID),  
    PaymentAmount DECIMAL(5,2) NOT NULL,  
    DatePaid DATE NOT NULL  
);
```

GO

--Create non-clustered index of customerID and LocationID on the Membership table

```
CREATE INDEX CustID on ShapeUpGym.dbo.Membership (CustomerID);  
CREATE INDEX LocID on ShapeUpGym.dbo.Membership (LocationID);
```

--Index-only plan for amount paid by membership id

```
CREATE INDEX Paid on ShapeUpGym.dbo.Payment (MembershipID,PaymentAmount);
```

6.2 File 2

--Insert customer records into table

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10000','Steven','Moon','1994-05-  
08','LD30DG','Moony8594@hotmail.com','07896980844','1123456635','0.00')
```

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10001','Bev','Holland','1970-10-  
04','SY210FG','BHolland@hotmail.com','07856284371','1129507452','0.00')
```

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10002','Ffion','Howells','1995-11-  
26','LD30DG','FHowells@hotmail.com','07857194756','4629456535','0.00')
```

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10003','Gaynor','Thomas','1984-02-  
07','SY160FH','GThomas@hotmail.com','07018465844','1101846635','0.00')
```

--Testing constraints for customer insert

```
BEGIN TRANSACTION
```

```
BEGIN TRY
```

--Already in database table

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10003','Gaynor','Thomas','1984-02-  
07','SY160FH','GThomas@hotmail.com','07018465844','1101846635','0.00')
```

--Not in database table

```
INSERT INTO ShapeUpGym.dbo.Customer  
(CustomerID,FirstName,Surname,DOB,Postcode,Email,Mobile,DebitDetails,DebitAmount)  
VALUES('C10006','Aled','Falvey','1966-02-  
07','LD50FH','AFalvey@hotmail.com','0701736444','1246346635','0.00')
```

```

COMMIT TRANSACTION ; -- commit once all queries executed and none have failed
END TRY

BEGIN CATCH
-- An error occurred, rollback ANY changes made during this transaction
ROLLBACK TRANSACTION ;
PRINT('Transaction has not been committed due to an error');
THROW;
END CATCH;

--Insert location records
INSERT INTO ShapeUpGym.dbo.GymLocation
(City,Postcode,JoiningFee,MonthlyFee,DailyFee)
VALUES('Swansea','SY105PL','20.00','30.00','3.00')

INSERT INTO ShapeUpGym.dbo.GymLocation
(City,Postcode,JoiningFee,MonthlyFee,DailyFee)
VALUES('Cardiff','CF130FJ','25.00','35.00','3.00')

INSERT INTO ShapeUpGym.dbo.GymLocation
(City,Postcode,JoiningFee,MonthlyFee,DailyFee)
VALUES('Brecon','LD30LU','15.00','30.00','3.00')

INSERT INTO ShapeUpGym.dbo.GymLocation
(City,Postcode,JoiningFee,MonthlyFee,DailyFee)
VALUES('Newtown','NW054KH','10.00','25.00','3.00')

--Testing postcode constraint
BEGIN TRANSACTION
BEGIN TRY
--Postcode not valid
INSERT INTO ShapeUpGym.dbo.GymLocation
(City,Postcode,JoiningFee,MonthlyFee,DailyFee)
VALUES('Welshpool','WP11','20.00','30.00','3.00')

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed as the postcode is too short');
THROW;
END CATCH

--Insert some test membership details with business rules
--Insert valid monthly membership for customer C10000 paying monthly
BEGIN TRANSACTION
BEGIN TRY
--Check if already has a membership
IF (SELECT ISNULL(EndDate,'1990-01-01') FROM ShapeUpGym.dbo.Membership WHERE
CustomerID = 'C10000') > GETDATE()
THROW 51000, 'Transaction has not been committed as the customer already has a
membership', 1;
INSERT INTO ShapeUpGym.dbo.Membership
(MembershipID,CustomerID,LocationID,MembershipType,MembershipLength,EndDate,PaymentType)
VALUES('M00001','C10000','1','Monthly','6',DATEADD(month,6,GETDATE()),'Monthly')

UPDATE ShapeUpGym.dbo.Membership

```

```

SET RemainingAmount = (MembershipLength * (SELECT MonthlyFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')) + (SELECT JoiningFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')
WHERE MembershipID = 'M00001'

UPDATE ShapeUpGym.dbo.Customer
SET DebitAmount = (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE
MembershipID='M00001') / ((SELECT MembershipLength FROM ShapeUpGym.dbo.Membership
WHERE MembershipID='M00001'))
WHERE CustomerID = 'C10000'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed as the customer already has a membership');
THROW;
END CATCH

--Insert valid monthly membership for customer C10001 paying Monthly
BEGIN TRANSACTION
BEGIN TRY
--Check if already has a membership
IF (SELECT ISNULL(EndDate, '1990-01-01') FROM ShapeUpGym.dbo.Membership WHERE
CustomerID = 'C10001') > GETDATE()
THROW 51000, 'Transaction has not been committed as the customer already has a
membership', 1;
INSERT INTO ShapeUpGym.dbo.Membership
(MembershipID, CustomerID, LocationID, MembershipType, MembershipLength, EndDate, PaymentType)
VALUES ('M00002', 'C10001', '1', 'Monthly', '12', DATEADD(month, 12, GETDATE()), 'Monthly')

UPDATE ShapeUpGym.dbo.Membership
SET RemainingAmount = (MembershipLength * (SELECT MonthlyFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')) + (SELECT JoiningFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')
WHERE MembershipID = 'M00002'

UPDATE ShapeUpGym.dbo.Customer
SET DebitAmount = (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE
MembershipID='M00002') / ((SELECT MembershipLength FROM ShapeUpGym.dbo.Membership
WHERE MembershipID='M00002'))
WHERE CustomerID = 'C10001'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed as the customer already has a membership');
THROW;
END CATCH

--Insert valid monthly membership for customer C10002 paying full amount of daily rate
BEGIN TRANSACTION
BEGIN TRY
--Check if already has a membership
IF (SELECT ISNULL(EndDate, '1990-01-01') FROM ShapeUpGym.dbo.Membership WHERE
CustomerID = 'C10002') > GETDATE()
THROW 51000, 'Transaction has not been committed as the customer already has a
membership', 1;

```

```

INSERT INTO ShapeUpGym.dbo.Membership
(MembershipID, CustomerID, LocationID, MembershipType, MembershipLength, EndDate, PaymentType)
VALUES ('M00003', 'C10002', '1', 'Daily', '3', DATEADD(day, 3, GETDATE()), 'Full')

UPDATE ShapeUpGym.dbo.Membership
SET RemainingAmount = (MembershipLength * (SELECT DailyFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1'))
WHERE MembershipID = 'M00003'

UPDATE ShapeUpGym.dbo.Customer
SET DebitAmount = (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE
MembershipID='M00003')
WHERE CustomerID = 'C10002'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed as the customer already has a membership');
THROW;
END CATCH

--Attempting to add a monthly fee to C10002
BEGIN TRANSACTION
BEGIN TRY
--Check if already has a membership
IF (SELECT ISNULL(EndDate, '1990-01-01') FROM ShapeUpGym.dbo.Membership WHERE
CustomerID = 'C10002') > GETDATE()
THROW 51000, 'Transaction has not been committed as the customer already has a
membership', 1;
INSERT INTO ShapeUpGym.dbo.Membership
(MembershipID, CustomerID, LocationID, MembershipType, MembershipLength, EndDate, PaymentType)
VALUES ('M00004', 'C10002', '1', 'Monthly', '6', DATEADD(month, 6, GETDATE()), 'Monthly')

UPDATE ShapeUpGym.dbo.Membership
SET RemainingAmount = (MembershipLength * (SELECT MonthlyFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')) + (SELECT JoiningFee FROM
ShapeUpGym.dbo.GymLocation WHERE LocationID = '1')
WHERE MembershipID = 'M00004'

UPDATE ShapeUpGym.dbo.Customer
SET DebitAmount = (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE
MembershipID='M00001') / ((SELECT MembershipLength FROM ShapeUpGym.dbo.Membership
WHERE MembershipID='M00001'))
WHERE CustomerID = 'C10002'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed as the customer already has a membership');
THROW;
END CATCH

--Customer making a payment
--Making a valid Monthly payment for Customer C10000 under membership M00001
BEGIN TRANSACTION
BEGIN TRY

```

```

IF (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE MembershipID =
'M00001') = 0
THROW 51000, 'Customer Does not owe any money', 1;

INSERT INTO ShapeUpGym.dbo.Payment
(MembershipID,PaymentAmount,DatePaid)
VALUES ('M00001',(SELECT DebitAmount FROM ShapeUpGym.dbo.Customer WHERE CustomerID =
'C10000'),GETDATE())

UPDATE ShapeUpGym.dbo.Membership
SET RemainingAmount = RemainingAmount - (SELECT DebitAmount FROM
ShapeUpGym.dbo.Customer WHERE CustomerID = 'C10000')
WHERE MembershipID = 'M00001'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed');
THROW;
END CATCH

--Making a full payment for MembershipID M00003
BEGIN TRANSACTION
BEGIN TRY
IF (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE MembershipID =
'M00003') = 0
THROW 51000, 'Customer Does not owe any money', 1;

INSERT INTO ShapeUpGym.dbo.Payment
(MembershipID,PaymentAmount,DatePaid)
VALUES ('M00003',(SELECT DebitAmount FROM ShapeUpGym.dbo.Customer WHERE CustomerID =
'C10002'),GETDATE())

UPDATE ShapeUpGym.dbo.Membership
SET RemainingAmount = RemainingAmount - (SELECT DebitAmount FROM
ShapeUpGym.dbo.Customer WHERE CustomerID = 'C10002')
WHERE MembershipID = 'M00003'

COMMIT TRANSACTION
END TRY

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed');
THROW;
END CATCH

--Attempting to repeat the above payment to test transaction check
BEGIN TRANSACTION
BEGIN TRY
IF (SELECT RemainingAmount FROM ShapeUpGym.dbo.Membership WHERE MembershipID =
'M00003') = 0
THROW 51000, 'Customer Does not owe any money', 1;

INSERT INTO ShapeUpGym.dbo.Payment
(MembershipID,PaymentAmount,DatePaid)
VALUES ('M00003',(SELECT DebitAmount FROM ShapeUpGym.dbo.Customer WHERE CustomerID =
'C10002'),GETDATE())

UPDATE ShapeUpGym.dbo.Membership

```

```

SET RemainingAmount = RemainingAmount - (SELECT DebitAmount FROM
ShapeUpGym.dbo.Customer WHERE CustomerID = 'C10002')
WHERE MembershipID = 'M00003'

```

```

COMMIT TRANSACTION
END TRY

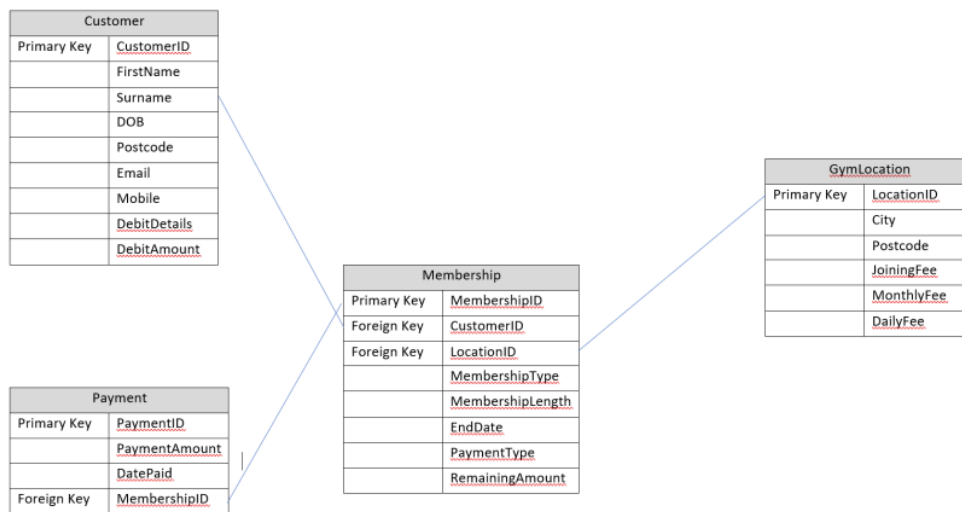
```

```

BEGIN CATCH
ROLLBACK TRANSACTION
PRINT('Transaction has not been committed');
THROW;
END CATCH

```

6.3 ERD



Assignment

GRADEMARK REPORT

FINAL GRADE

51 / 100

GENERAL COMMENTS

Instructor

Your schema design was somewhat simplistic, To fully address the requirements of the gym would need additional table - for example for staff, facilities and links for member bookings.

You do demonstrate a good understanding of eth SQL development environment and you have translated you design not SQL correctly; you have used this to implement test data and look at appropriate transactions and indexes as required by your design. You have discussed the attribute od distributed design, but you have now presented any examples in the form of data within your report or views in your database design within SQL.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12