CA1: Database Design and Development

Module Title: Database Design & Development

Module Code: B8IT113

Module Leader: Jennifer Byrne

Student Name: Ross Maguire

Student Code: 10556781

Contents

CA1: Database Design and Development 1

1. Project Overview/Scope 2

2. Entity Relationship Diagram 2

3. Assumptions Made 3

4 Data Dictionary 3

5. Technology Used 5

6. Test Plan 7

7. Reflections on Learning 9

8. References 10

9. SQL 10

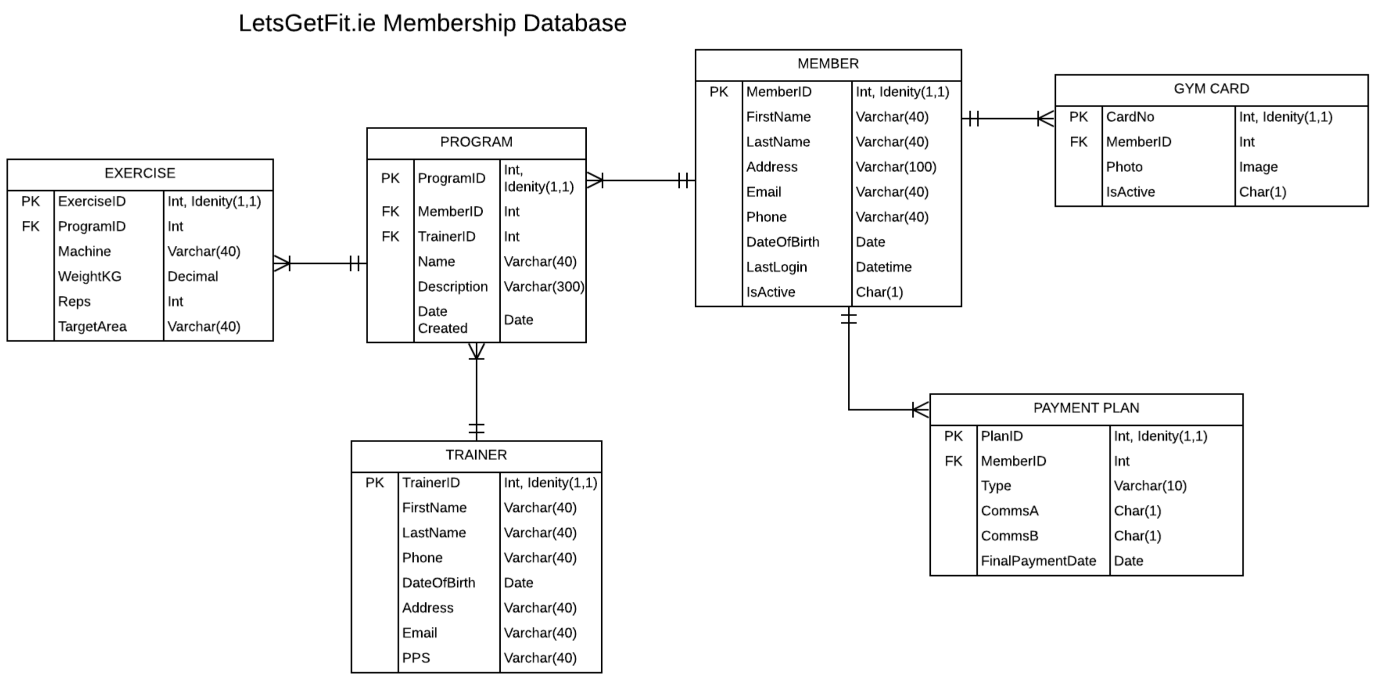
1. Project Overview/Scope

This project provides a technical design and provision of a new database server and the development of a new database for LetsGetFitGym.

This design is for Phase 1 - mapping out the data that needs to be managed by the gym and setting up the database to be used by a front-end application. Total number of phases is yet to be decided.

This phase will look at capturing the current paper-based system into an SQL database that can be queried by an application for employees to use.

2. Entity Relationship Diagram



3. Assumptions Made

- A member can only have 1 card at a time but replace them if lost/stolen
- A trainer is an employee of the gym and has access to the application to record a new program
- A trainer draws up the details of the program in-person based on a real-world discussion
- Program details are saved in the database by a trainer creating Exercise entities that all have matching ProgramIDs. This way the program can be modified by adding or deleting exercises
- External card scanning hardware is installed in the gym that connects to the database for updating 'Last Login' in the member table by matching the Card No to MemberID
- If a member loses a card, we assume the gym can create a new one for them and keep record of the old cards so that others can't use them by making them inactive
- If a member has chosen to allow the gym to keep business critical info kept on file i.e soft delete (Comms B) then when they are deleted, the info being deleted will be replaced with 'GDPR'. Email is the mechanism they will use to contact the customer for further offers so wish to keep this on file along with name and last login
- When being hard deleted, if a member final payment date is not older than 30 days we return a message to say 'will be deleted after final payment date is 30 days' and assume that our application will do something here – to be developed at a later stage of the project
- A program automatically has a date saved from the application so is not entered by the trainer

4. Data Dictionary

Member

Attribute	Datatype	Required	Description
MemberID	int	Yes	Primary Key for member
FirstName	varchar(40)	Yes	Members First Name
LastName	varchar(40)	Yes	Members Last Name
Address	varchar(100)	Yes	Members Address
Email	varchar(40)	Yes	Members Email for Comms
Phone	varchar(40)	Yes	Phone Number
Date Of Birth	date	Yes	Member DOB
Last Login	datetime	No	Records when they login with card – not required as will be null to begin with
isActive	char(1)	Yes	Y or N if membership is active or not

Gym Card

Attribute	Datatype	Required	Description
CardNo	int	Yes	Primary Key for gym card number
MemberID	int	yes	Foreign Key to identify which user owns this card
Photo	image	Yes	Users photo to identify
isActive	char(1)	Yes	Y or N if this card is active or not

Payment Plan

Attribute	Datatype	Required	Desc
PlanID	int	Yes	Primary Key specific to the members plan
MemberID	int	Yes	Foreign Key to id member who is on this plan
Type	varchar(10)	Yes	Monthly, Quarterly or Yearly
Comms A	char(1)	Yes	Y or N on emails and texts
Comms B	char(1)	Yes	Y or N on hard or soft delete
Final Payment Date	date	Yes	The final date they are to pay or have paid which we measure for hard delete after 30 days

Trainer

Attribute	Datatype	Required	Description
TrainerID	int	Yes	Primary Key Specific to trainer
FirstName	varchar(40)	Yes	First Name of Trainer
LastName	varchar(40)		Last Name of Trainer
Phone	varchar(40)	Yes	Phone Number
Date Of Birth	date	Yes	DOB of Trainer
Address	varchar(100)	Yes	Address of Trainer
Email	varchar(40)	Yes	Trainer Email
PPS No	varchar(40)	Yes	PPS for employee record

Program

Attribute	Datatype	Required	Desc
ProgramID	int	Yes	Primary Key specific ID for program
MemberID	int	Yes	Foreign Key to relate the member who this program is for
TrainerID	int	Yes	Foreign Key to relate the trainer who set up the program
Name	varchar(40)	Yes	Name given to the program
Description	varchar(300)	No	Optional program description
Date Created	date	Yes	The date recorded automatically when the program was set up

Exercise

Attribute	Datatype	Required	Description
ExcersiseID	int	Yes	Primary Key for specific exercise
Name	varchar	Yes	Name given by trainer to this execrsise
Machine	varchar	Yes	Machine or Weight Name
WeightKG	decimal	Yes	Weight amount in KG
Reps	int	Yes	Number of reps with this machine or weight
Target Area	varchar(40)	No	Specific area of the body – can be null
ProgramID	int	Yes	Foreign Key relating to program this exersise is a part of

5. Technology Used

SQl Server

Microsoft Word

Lucid Chart

6. Test Plan

Item Tested	Test Run	Expected Result	Actual Result

Form a Program by selecting all its exercises by ProgramID	SELECT ExerciseID, Machine, WeightKG, Reps, TargetArea
	FROM Exercise
	WHERE
	Exercise.ProgramID = '4'
		Returns all program ascertaining to a collection of exercises, making a programs contents viewable	Returned a table with the exercise listed with ProgramID '4'

Test the MI Single View	SELECT * FROM VW_MI_Single_View	Return all active Members, their Program details and Trainer details	View executed successfully. Table returned
Create a new member via PSP	EXEC usp_NewMember @FirstName = **** , @LastName = **** , @Address = '30 Main Street, Newcastle', @Email = ' **** @ **** , @Phone = ' **** ', @DOB = '10/16/1984'	Add a new member entity to the member table – id automatically generated	Member created in the table
Create a new Program via PSP	EXEC usp_NewProgram @MemberID = '15', @TrainerID = '4', @ProgramName = 'Intermediate Cardio', @ProgramDesc = 'Some medium weight cardio exercises'	Add a new Program entity to the program table – id automatically generated	Program created in table

Create a new Exercise via PSP and make it part of a Program	EXEC usp_AddExercise @ProgramID = '6', @Machine = 'Tread Mill', @WeightKG = NULL, @Reps = NULL, @TargetArea = 'Run for 45 mins on treadmill'	Add a new Exercise entity to the exercise table – id automatically generated	Exercise created with program ID of '6' so when reproducing test 2 using ProgramID '6' this exercise appears
Create New Trainer Via SP	EXEC usp_NewTrainer @FirstName = **** , @LastName = **** , @Phone = **** , @DOB = '05-14-1990', @Address = '21 **** Manor, **** , @Email = '**** @gmail.com', @PPS = '8020184Y'	Create a new trainer in the db	New trainer created in the members table
Test the hard delete part of the delete SP by using a member who has a payment over 30 days and comms b not ticked	EXEC usp_Hard_Delete_Member @MemberID = '7', @Email = ' **** @hotmail.com'	Record should be deleted	Record deleted
Test the soft delete part of the delete SP by using a member who has ticked comms b	EXEC usp_Soft_Delete_Member @MemberID = 4, @Email = ' **** @gmail.com'	Address, Phone shouldbe set to GDPR and DatOfBirth set to Null Active set to N	Record Deleted
Test Generic delete SP which will check if the member has final payment over 30 days and comms b not ticked so will be hard deleted	EXEC usp_Delete_Member @MemberID = '6', @Email = **** @gmail.com'	usp_Delete_Member SP should trigger usp_Hard_Delete_Member and delete the record	Successful and record deleted
Test Generic Delete Member SP with a member who final payment is less 30 days and comms b is not ticked so will return a message to say eventually be deleted	EXEC usp_Delete_Member @MemberID = '3', @Email = **** @gmail.com'	usp_Delete_Member SP should trigger usp_Hard_Delete_Member and add a message as they are less than 30 days final payment	Returned message 'Will be deleted when final payment date is past 30 days'

Test Generic Delete Member SP with member who comms b is ticked so will be soft deleted	EXEC usp_Delete_Member @MemberID = '5', @Email = **** @hotmail.com'	usp_Delete_Member should trigger usp_Soft_Delete_Member and add GDRP in relevant fields	Successful and record changes to GDPR in relevant fields
Test the Deleted Member View	SELECT * FROM VW_Deleted_Members	Should return a list of members who were soft deleted	Successfully return the members who were soft deleted during testing and have their relevant details changed to 'GDPR'

7. Reflections on Learning

When Importing the SQL from Lucid Chart it gave me a few realisations about my ERD

- I needed to order my CREATE TABLE statements in such a way so that the table was created before it was referenced elsewhere as a Foreign Key

- This also gave me insight to be able to rework my ERD in Lucid Chart even after export to better visualise the tables and how they are linked in my Technical Document

Once I started writing my SQl statements I realised a couple of things:

- It would be more efficient to change the column names to not have spaces in writing code. So I rename in SQL Server and went back to my ERD & Data Dictionary to change the names

- It would be more efficient to write SQL if I had have named my columns unique so I wouldn't have ambiguous table names occur as an error anytime I didn't specify the table I was referencing.

- Had to revise my ERD after writing statements as I realised I didn't need certain foreign keys in certain tables

- I left exporting my ERD until the last minute because I had some realisations along the way thereafter. However, the overall model didn't change and no new entities were added or relationships created. More so around Keys, DataTypes and Naming

- When writing conditional statements for SP say for example the delete SP which triggers the hard or soft delete depending on the relating Final Payment Date and Comms B in that members payment plan row – I pondered whether an application layer on top of this database could do some of the heavy lifting in terms of deciding if a user would be hard deleted or not by reading the database seperately.

8. References

Caleb Curry YouTube Channel

Socratica YouTube Channel

microsoft.com

geeksforgeeks.org

w3schools.com

stackoverflow.com

moodle

9. SQL

letsgetFitGym-SQL.sql

This file should be used to setup the db. Please execute the file in order:

- CREATE Tables
- INSERT data
- CREATE Stored Procedures
- CREATE Views

test-data-SQL.sql

This file provides all the code in order to follow the test plan above (heading 6). Please execute in order the statements appear in the file.