Seattle Badminton Club Database

Database Systems (CSS 475)
Professor: Dr. Erika Parsons
Semester: Fall 2014
Group Name: YODA Ltd (p8)

Group Members:
Yauheniya (Eugenia) Zapryvaryna
Omar Idris
Daejung Kim
Carlos Aldridge

December 11, 2015

1. Introduction

The project we, members of Yoda Ltd, worked on involved developing an intensive database for the Seattle Badminton Club (SBC) located in Kirkland Washington. This database hold the club's guests' information (such as their names and contact info), membership information (including the membership card's details), the inventory of the club's pro-shop, guests' orders and their details, vendors, owners, and employees data, and class schedule including the attendance records for each class.

1.1. Background

The reason our team took chose to do this project is that at the start one of our member, Carlos Aldridge, noticed multiple issues caused by the SBC's sole reliance on forms and paper trails to keep track of its data that can be simply solved by a database while he worked at the club. Some of the problems and solutions are highlighted in the following sections.

1.1.1. Problems Caused by Paper Trail System

One issue with the paper trail system currently employed by SBC is the difficulty to keep track of an order placed by a guest/member of the club. The order may be placed by the staff but after that the front desk has not information on the progress of the item until it gets to the club. Another the staff at the club face is the lack of an ability to verify the status of a guest membership (whether they are a member or not), which leads to possibility of guest saying they are members, when they are not, only to get the member discounts.

1.1.2. What the Database Solves

Simply put a majority of the issues are solved by a database inherent characteristic to be able to have data stored in a single location and provide easy accessibility to it by generating multiple views.

1.2. Overview

The database will be used directly by all of the employees of the Badminton club. Owners will use it to access any information on guests, employees, vendors, and the inventory. And the regular employees will use it for the same purposes with the exception that they won't be able to access personal data about each other, unless authorized by their position (such as an accountant or a general manager).

1.2.1. Schedule:

This is the general tentative schedule our team followed:

- ➔ Week 1 (Iteration 1): 11/1/2015 - 11/8/2015
  - ◆ Contract
  - ◆ Project Proposal
  - ◆ Design (ER Diagram, Relational Diagram)
- ➔ Week 2 : 11/9/2015 - 11/16/2015

        - Create a list of Cascading Deletes
        - Create Database with Constraints
        - Create all database entities
    - Week 3 (Iteration 2): 11/16/2015 - 11/25/2015
        - Updated Project Proposal based on Feedback
        - Write SQL Statements
        - Populate the database with entities
    - Week 4 : 11/27/2015 - 12/6/2015
        - Prepare Powerpoint Presentation
    - Week 5 (Iteration 3): 12/7/2015 - 12/13/2015
        - Present Powerpoint Presentation
        - Finalize Document

1.3. Assumptions and Constraints

While designing the Seattle Badminton Club database we came up with the following assumptions and the constraints on the design:

- We don’t need to keep record of the state people (except for vendors) are in because the club is only located in Kirkland, Washington and the members are only from Washington.
- Guests are not necessarily members. There are guests who don’t have a membership, and so the MembershipID can be NULL.
- the orders guests place are not delivered to them, but to the club itself. The club then notifies the guests of the delivery of their order and that they could pick them up straight at the club. That's why we have the boolean attribute wasTheGuestContacted.
- Owner is not always an employee

2. Design

2.1. Approach

We used a bottom-up approach to the database design strategy where we took the current details (what known to us about the Seattle Badminton Club) and came up with possible attributes for the future relations, by using NOUN strategy. We then grouped the associations of the attributes into relations, analyzed what types of relationships we might end up with between the entities entities. We came up with an ER diagram and the RDM, and then later normalized the relations in the database. The reason why we went with a bottoms up approach was because we wanted to match the database as close as possible to the actual services and entities at play at SBC. This approach did not pan out as great as we hoped as we had to add a few details that were not a part of the club's actual operation such as the Vendors (SBC is a direct retail for the YONEX badminton gear).

We used MySQL and did not feel the need to use Azure, as the Seattle Badminton Club has its own localized server. Furthermore if actually implemented the database would house some sensitive information, which we can’t risk to lose.

Visual C# was used to come up with the console application for the database.

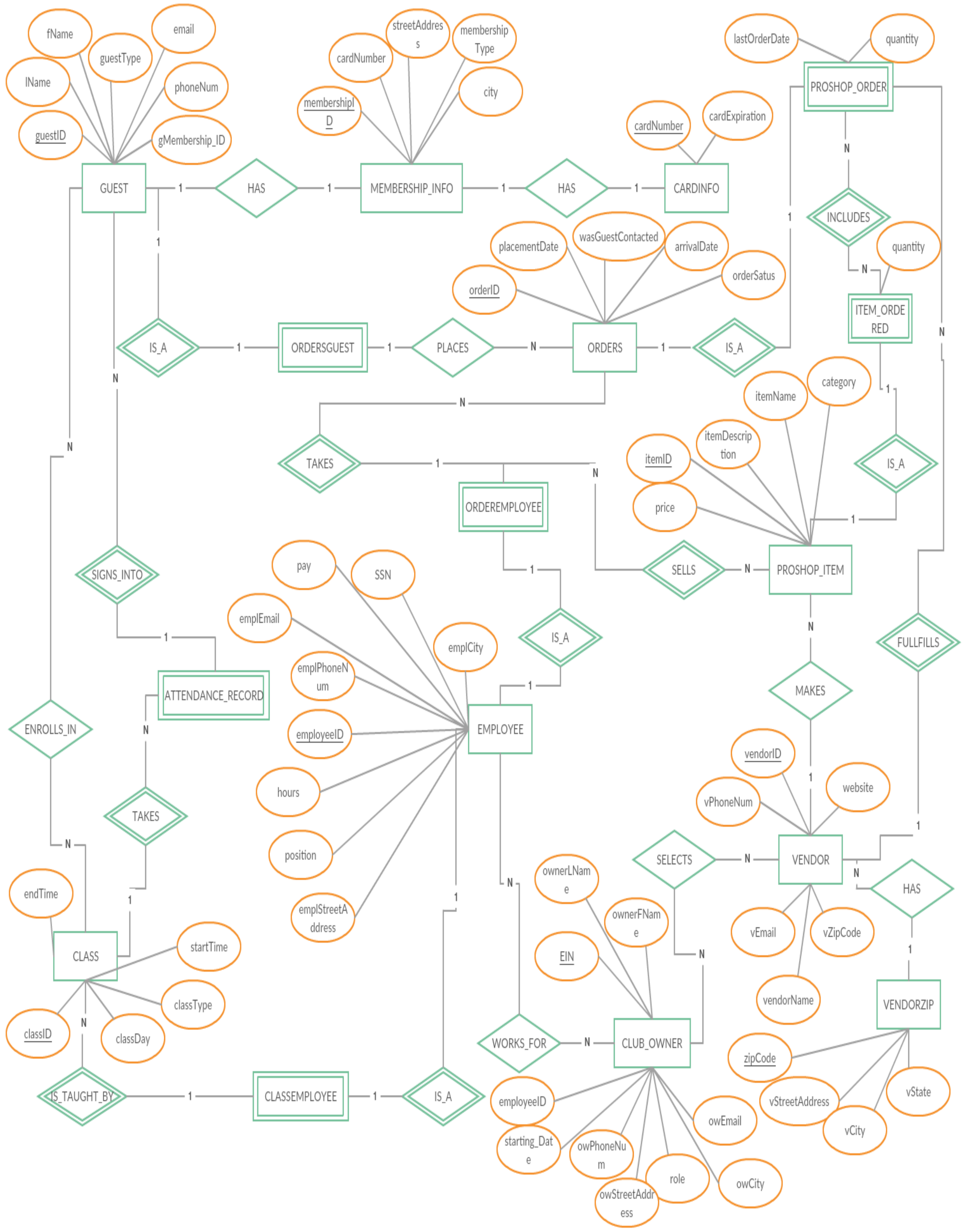
2.2. Use Case

The SBC database is designed for the use of the employees and the owners. While owners get full access to the database, it is restricted for the employees depending on the position.

2.3. Design Documentation

Our design documentation included the Entity-Relationship diagram (Figure 1) as well as the Relational Data Model (Figure 2), based on the ER diagram. Just as in the software design, the diagrams had to be remade as the project unfolded.

Figure 1: Entity Relationship Diagram that was updated since the previous iteration to include the updated entities in 3NF form and the newly created entities that came from the normalization process



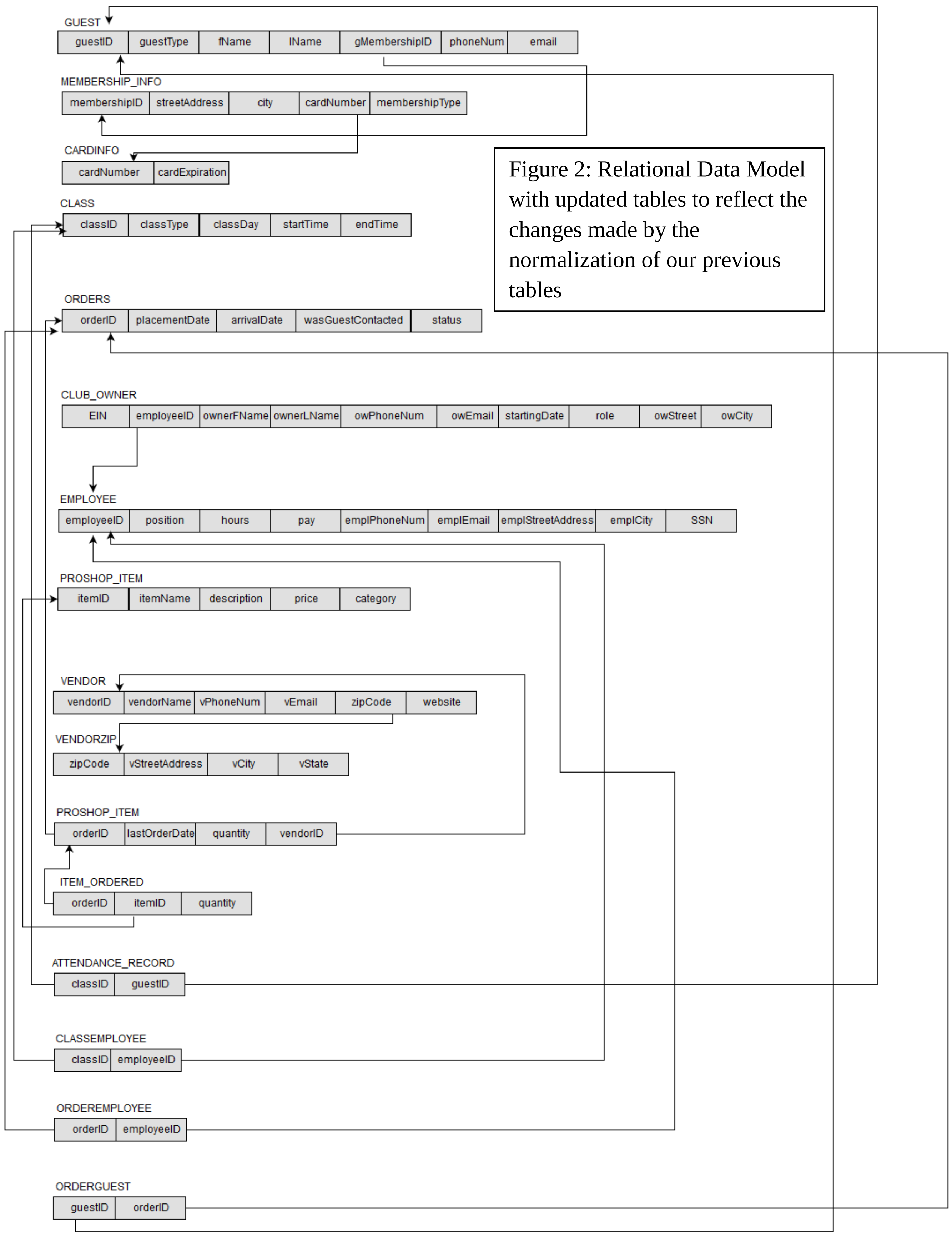


Figure 2: Relational Data Model with updated tables to reflect the changes made by the normalization of our previous tables

2.4. Data That is To Be Stored By Database

The Database is designed to hold the following data to reflect the curre

- Badminton Classes (ID, schedules, types (private or regular), which instructor teaches them, and the attendance of each class)
- Inventory: (ID, vendor ID that supplies it, name, description, price, quantity, price, category, the last order date).
- Orders:(guest ID, employee who placed the order for the guest, order ID, placement and arrival dates, status, and whether the guest was contacted).
- Guests (guest ID , membership ID, guest type - regular or member, name, and contact information) and their Membership information (ID, type, card number, card expiration, guest's address).
- Employees (ID, name, position - manager, coach, accountant, front desk, hours, pay, SSN, and contact information)
- Owners (EIN, employee ID, name, contact information, role, starting date).
- Vendors (ID, name, contact information, website).

2.5. Queries it will be able to handle

2.5.1. Add/Remove/Update

- Guests: adding a guest, removing a guest, changing any information about a particular guest, such as an email or a type of guest.
- Proshop Item: adding or deleting an inventory item, updating any information regarding a specific item.
- Class: adding or removing a class, changing information of a particular class, such as its time of start or end, its instructor etc.
- Order: adding or deleting an order, changing the order as per guest's request (for example, the quantity of a proshop item ordered, adding more items to the order or removing items from the order).
- Employee: adding or removing a specific employee, updating any information of a specific employee, such as the contact information, pay etc.
- Owner: adding or removing an owner, updating the information of a specific owner, such as contact information, or the role.
- Vendor: adding or removing a vendor from the database, changing information about a particular vendor, such as the phone number, or website.

2.5.2. Display

- Order status, the employee who took the order, and what a guest ordered it.
- Employee information, hours worked by each, employee schedule.
- Guest Membership information.
- Class schedule.
- Class Attendance.
- Inventory of Sporting Equipment and Apparel
- Order History (orders placed during a specific period of time).

- Owners info (SSNs, addresses, names, phone numbers, etc.)
- Vendor list (names, numbers, addresses).
- Guest list (all the info about the guests)

2.5.3. Normalization

On our initial database design, some of our tables were in 1NF. Those tables include Class, Order, Proshop_Item, and Item_Ordered. We split these tables into smaller ones that satisfied the 3NF. These are a set of new tables created for our database. After reworking our tables, we went through each one and verified that they were all in 3NF based on the definitions of the NF:

- 1NF: No multi-valued attributes
- 2NF: 1NF + All non-key attributes are fully functionally dependant on primary key.
- 3NF: 2NF + No transitive functional dependency.

2.6. Data Gathering

Data was gathered by Carlos Aldridge, the actual employee of the Seattle Badminton Club in Kirkland, WA. Although he had to adjust certain sensitive private information for the database. But the type of data matches what the SBC's current database contains. Furthermore, the data is limited by the

2.7. Testing

The testing component was done by using the data and running queries on the data to see whether the outcome matches with the information we had.

3. SQL

3.1. Creation

Table 1 shows the MySQL statements we used to make the database

Table 1: Create statements for Database

SQL Statement
create table cardInfo (cardNumber varchar(16), cardExpiration date not null, primary key(cardNumber));
create table MEMBERSHIP_INFO (membershipID int not null, streetAddress Varchar(30) not null, city varchar(20) not null, cardNumber varchar(16), membershipType varchar(20) not null, primary key(membershipID), constraint MEMBERSHIPFK1 foreign key (cardNumber) references cardInfo(cardNumber));
create table GUEST (guestID int not null, guestType varchar(15) not null, lName varchar(20) not null, fName varchar(20) not null, gMembership_ID int, email varchar(30) not null, phoneNum varchar(11) not null, primary key(guestID), constraint GUESTFK1 foreign key (gMembership_ID) references MEMBERSHIP_INFO(membershipID) on delete set null on update cascade);
create table EMPLOYEE (employeeID int not null, position varchar(30) not null, hours int not null, pay double not null,

```
emplPhoneNum varchar(11) not null,
emplEmail varchar(40) not null,
emplStreetAddress varchar(50) not null,
emplCity varchar(20) not null,
SSN int not null,
primary key (employeeID));
```

```
create table ClassEmployee
(
classID int not null,
classEmployeeID int not null,
constraint CLASSEMPLOYEEFK1
foreign key (classEmployeeID) references
EMPLOYEE(employeeID)
on delete cascade
on update cascade,
constraint CLASSEMPLOYEEFK2
foreign key (classID) references CLASS(classID)
on delete cascade
on update cascade
);
```

```
create table CLASS
(
classID int not null,
classType varchar(20) not null,
classDay varchar(20) not null,
startTime varchar(20) not null,
endTime varchar(20) not null,
primary key(classID)
);
```

```
create table ATTENDANCE_RECORD
(
attClassID int not null,
attGuestID int not null,
constraint ATTFK1
foreign key (attClassID) references CLASS(classID)
on delete cascade
on update cascade,
constraint ATTFK2
foreign key (attGuestID) references GUEST(guestID)
on delete cascade
on update cascade
);
```

```
create table ORDERS
(
orderID int not null,
placementDate date not null,
arrivalDate date,
wasGuestContacted bool not null default 0,
orderStatus varchar(30) not null,
primary key(orderID)
);
```

```
create table ORDERSGUEST
(orderID int not null,
orderGuestID int,
constraint ORDERGUESTFK1
foreign key (orderGuestID) references GUEST(guestID),
constraint ORDERGUESTFK2
foreign key (orderID) references ORDERS(orderID)
);
```

```
create table ORDEREMPLOYEE
(
orderID int not null,
orderEmployeeID int,
constraint ORDEREMPLOYEEFK1
foreign key (orderEmployeeID) references
EMPLOYEE(employeeID),
constraint ORDEREMPLOYEEFK2
foreign key (orderID) references ORDERS(orderID)
);
```

```
create Table VENDOR
(vendorID int not null,
vendorName varchar(20) not null,
vPhoneNum varchar(11) not null,
vEmail varchar(30) not null,
vzipCode int(5) not null,
vebsite varchar(30) not null,
primary key (vendorID),
constraint VENDORFK1
foreign key (vzipCode) references VENDORZIP(zipCode)
);
```

```
create table VENDORZIP
(
zipCode int(5) not null,
vStreetAddress varchar(50) not null,
vCity varchar(20) not null,
vState varchar(20) not null,
primary key (zipCode)
);
```

```
create table PROSHOP_ITEM
(itemID int not null,
itemName varchar(25) not null,
itemDescription varchar(50) not null,
price double not null,
category varchar(20) not null,
primary key (itemID)
);
```

```
create table PROSHOP_ORDER
(
itemID int not null,
vendorID int not null,
lastOrderDate date not null,
quantity int not null,
constraint PROSHOP_ORDER1
foreign key (vendorID) references VENDOR(vendorID)
on delete cascade
on update cascade,
constraint PROSHOP_ORDER2
foreign key (itemID) references PROSHOP_ITEM(itemID)
on delete cascade
on update cascade
);
```

```
create table ITEM_ORDERED
(orderID int not null,
itemID int,
quantity int not null,
constraint ITEM_ORDEREDFK1
foreign key (orderID) references ORDERS(orderID)
on delete cascade
on update cascade,
constraint ITEM_ORDEREDFK2
foreign key (itemID) references PROSHOP_ITEM(itemID)
on delete set null
on update cascade
);
```

```
create table CLUB_OWNER
(EIN int not null, ownerFName varchar(20) not null,
ownerLName varchar(20) not null, owPhoneNum varchar(11)
not null,
owEmail varchar(40) not null, starting_Date Date not null,
role varchar(30), owStreetAddress Varchar(40) not null,
owCity Varchar(20) not null, employeeID int,
primary key(EIN),
constraint CLUB_OWNERFK1
foreign key (employeeID) references
EMPLOYEE(employeeID)
on delete set null
on update cascade
);
```

3.2. Population

Here Table 2 covers the insert statements that we used to populate our newest version of the database after our normalization reform of it.

Table 2: Inserts to populate SBC database

SQL Statement
insert into cardInfo value(1234567890, 20180301);
insert into cardInfo value(1111111111, 20160701);
insert into cardInfo value(2222222222, 20200101);
insert into MEMBERSHIP_INFO value(11, '24222 Sunrise', 'Bothell', 1234567890, 'A');
insert into MEMBERSHIP_INFO value(12, '11042 NE Bothell DR', 'Bothell', 1111111111, 'A');
insert into MEMBERSHIP_INFO value(13, '131 South Park', 'Seattle', 2222222222, 'B');
insert into EMPLOYEE value(100, 'CFO', 45, 45000, 2223334444, 'sfs32@yahoo.com', '4223 Wood DR', 'Woodinville', 123456789);
insert into EMPLOYEE value(101, 'Finance', 48, 38000, 2223334444, 's5321@uw.edu', '13812 Bellevue road', 'Bellevue', 0987654321);
insert into EMPLOYEE value(102, 'Statician', 54, 43000, 2223334444, 'jdae2f@gmail.com', '322 Rest square', 'Bothell', 1111111111);
insert into EMPLOYEE value(221133, 'Owner', 45, 50000, 2223332222, 'Deathstar@j.net', 'DSA villege 123', 'Bellevue', 222333444);
insert into GUEST value(100, 'A', 'Luke', 'Walker', 11, 'aewws234@gmail.com', 2223332222);
insert into GUEST value(101, 'B', 'Ow', 'Wes', 11, 'khurdq@gmail.com', 2244444422);
insert into GUEST value(102, 'C', 'Micro', 'Soft', 11, 'gasewt434534@ms.com', 1234567890);
insert into CLASS value(1000, 'BasicBadminton', 'Tuesday', '12:00', '14:00');
insert into CLASS value(1021, '2:2 Match', 'Thursday', '8:00', '10:00');
insert into CLASS value(1042, 'Badminton 101', 'Friday', '19:00', '21:00');
insert into ClassEmployee value(1000, 100);

insert into ClassEmployee value(1021, 101);
insert into ClassEmployee value(1042, 102);
insert into attendance_record value(1000, 100);
insert into attendance_record value(1000, 101);
insert into ORDERS value(12412, '2015-8-12', '2015-08-15', true, "Arrived");
insert into ORDERS value(11322, '2015-9-17', '2015-09-22', false, "Arrived");
insert into ORDERS value(13211, '2015-12-8', '2015-08-10', true, "Departed");
insert into ORDERSGUEST value(11322, 101);
insert into ORDERSGUEST value(12412, 100);
insert into ORDERSGUEST value(13211, 101);
insert into ORDEREMPLOYEE value (11322, 102);
insert into ORDEREMPLOYEE value (12412, 102);
insert into ORDEREMPLOYEE value (13211, 102);
insert into VENDORZIP value(98021, 'Canyon Park Place 2211', 'Bothell', 'WA');
insert into VENDORZIP value(98052, 'Village Square NE 87st', 'Redmond', 'WA');
insert into VENDORZIP value(98103, 'ABC building 20th floor', 'Seattle', 'WA');
insert into VENDOR value(1000, 'A-company', '4253332222', 'sdfkjh452@gmail.com', 98021, 'Do not have');
insert into VENDOR value(1001, 'B-company', '4254441112', 'gsdfsr@yahoo.com', 98103, 'Do not have');
insert into VENDOR value(1002, 'C-company', '4255224442', '42gsdfe@gmail.com', 98052, 'Do not have');
insert into PROSHOP_ITEM value(10000, 'Super Racket', 'Very Strong Racket', 19.99, 'Equipment');
insert into PROSHOP_ITEM value(20000, 'Badminton 101', 'Boring textbook like a college book.', 24.99, 'Book');
insert into PROSHOP_ITEM value(30000, 'Badminton suite', 'Lee style yellow exercise suite', 19.99, 'Apparel');

insert into PROSHOP_ORDER value(20000, 1001, '2015-9-22', 50);
insert into PROSHOP_ORDER value(30000, 1002, '2015-11-19', 100);
insert into PROSHOP_ORDER value(10000, 1000, '2015-10-1', 20);
insert into ITEM_ORDERED value(12412, 10000, 1);
insert into ITEM_ORDERED value(11322, 20000, 1);
insert into ITEM_ORDERED value(13211, 30000, 3);
insert into CLUB_OWNER value(221133, 'Das', 'Vader', '2223332222', 'Deathstar@j.net', '2005-1-1', ", 'DSA villege 123', 'Bellevue', 221133);

3.3. *Queries of Interest*

- display guests that didn't attend a specific class
- display the order history of a guest
- display members whose payment information is about to expire
- display coaches that are teaching a class at a specific time
- display employee hours worked per week

4. Interface

The interface we designed (Figure 3) was purely for the Owner use case as it has some functionality that should not be accessible to any regular employee of the club.

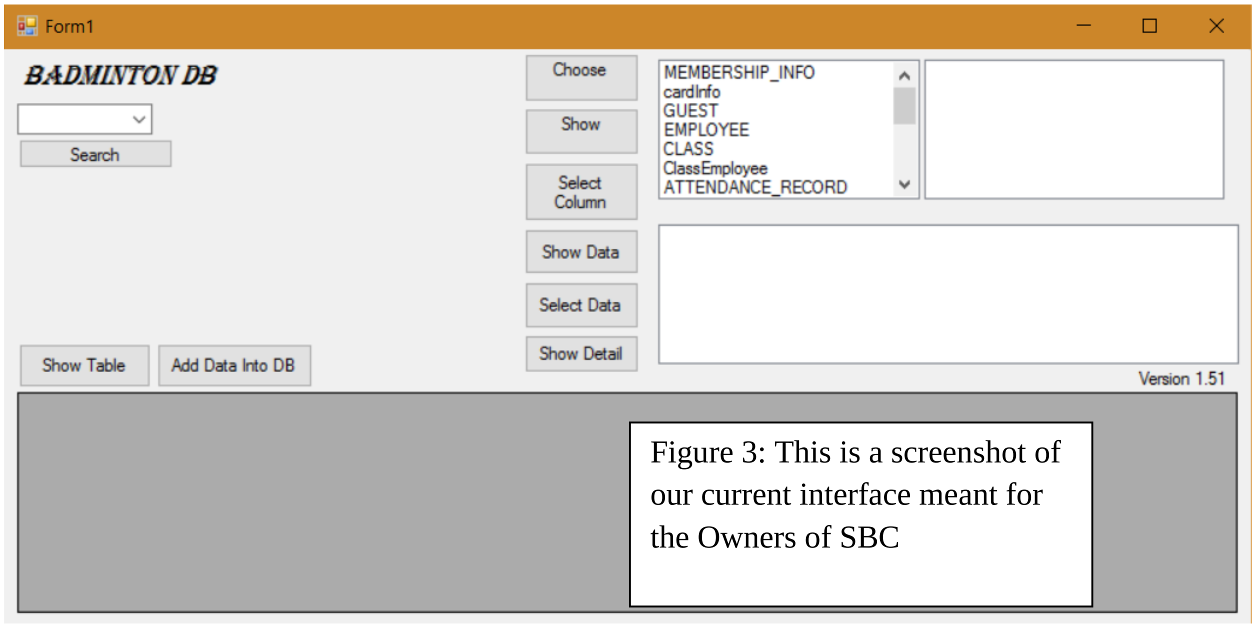


Figure 3: This is a screenshot of our current interface meant for the Owners of SBC

4.1 Function

The goal of this application :Searching data from DB and add new data into DB tables

- Choose button : Selecting data table. There is an existed list of the database' tables.
- Show button: Show all columns of the selected table on the second box which is on right side of the table list box.
- Select column button: Selecting column.
- Show data button: show all data of the selected column. There is a search value on the first line of the box.
- Select data button: selecting data.
- Show data button: show about information of selected data based on its table. if user chose search value, they can search data they want.
- Search button: There are 3 selections for operating detail search.
- Show Table button: Show table which user chose.
- Add Data into DB: insert new data into the chosen table.

4.2 Usage

1. Download application file and install it, then execute Application_DB.app.
2. Insert correct user database information. For example, server: localhost, userID: root, password: yourpassword, database: databaseName. If user failed to insert correct information, this application failed to connect MySql database and will close itself.
3. Search button has 3 functions, and it triggers 3 select statment.
    a. Order Info : searching specific order information.
    b. Order Ongoing : searching order information which is currently processing.
    c. Check Card Expiration: show card number which was expired.
    d. At the first time. user should choose table. Except select function, all functions will works after choosing table.
    e. There are 6 buttons on left side. These are for searching a specific and detailed data and table sheet. The process is choosing table -> Show column -> Select Column -> Show Data -> Select Data-> show details this process is just clicking button from top to bottom with selecting table, column, and data.
    f. Show table button will show selected table on the bottom gray box. This button can be used after choosing table.
    g. Add data into DB button will helps inserting new data into table. This button can be used after choosing table. After clicking this button, application will ask users inserting new data at each column of selected table.

5. Project Evaluation

5.1. Effort Spent Overall

Table 3: Contribution Outline

Name	Contributions
Yauheniya (Eugenia) Zapryvaryna	Initial ER and RDM, Iteration Reports, Interesting SQL Statements, Final

	Report, Final ER, Final RDM
Omar Idris	Initial ER and RDM, Normalization
Daejung Kim	Initial ER and RDM, Normalization, Initial SQL Create Statements, Interface, Application Manual
Carlos Aldridge	Initial ER and RDM, Initial SQL Create Statements, SQL Insert Statements, Iteration Report, Final Report

5.2. What Went Right

We were able to successful accomplishment of the database for the SBC. Furthermore, and although it is basic, we were able to implement an interface for the database.

5.3. What Went Wrong

- schedule issues among the team members
- some miscommunication
- the difficulty of the concurrent classes of some team members interfered with the CSS 475
- the addition of the fourth member happened later than when the project initiated cause serious reformating of original role assignments
- different levels of effort from the team members

5.4. Possible Improvements

If we had more time to work on this project we would try to accomplish the following:

- If possible it would be great to gather more data that we would be able to design more intensive test to.
- Include triggers to make our database more robust
- Implement an interface for the Employees
- Improve the current interface for Owners