CISC 332

QBnB Project Deliverable 2

Zachary Baum 10090150

Vinyas Harish 10089169

March 10th 2016

Table of Contents

Continued Assumptions 1

Updated Relational Schema & Table Dumps 2

Required SQL Statements & Sample Outputs 3

State Machine Diagram 4

User Interface Flow 5

Main Page 5

Sign Up Page 5

Sign In Page 5

User Dashboard 5

Settings Panel 6

Administrator Panel 6

# Continued Assumptions

1. We have used member\_ID as the primary key for our Member table, restricting each email to being associated with one account on QBnB. We are assuming since this is a Queen’s alumni database we will not have an integer overflow with the number of registered users.
2. Users will not enter invalid information or forms that are incomplete when using our application.
3. All Properties in the Property table must be owned by some member of QBnB.
4. One member may own multiple properties, but multiple members cannot own the same property.
5. Each property belongs to one and only one district, but a district may encompass multiple properties.
6. All bookings must be approved (or denied). Users can only cancel bookings if they are “pending”.
7. The member that owns a rentable property is able to approve (or deny) any bookings, and all bookings need only be approved once.
8. Owner\_ID corresponds to the Member\_ID of the owner of the property in question.
9. Any member may comment on a property, but the owner of the property is the only user who may reply directly to comments on their property. In order to comment on a property, a member must have booked it at some point in time. Members can only comment based on their booking once.
10. Administrator accounts will be limited to five, and will be the first five accounts created in the Member table (Member\_IDs 1, 2, 3, 4 and 5).
11. When leaving feedback on a property, a comment is required, but a rating (out of five) is optional.

# Updated Relational Schema & Table Dumps

CREATE DATABASE QBnB;

USE QBnB;

CREATE TABLE Member(

Member\_ID int NOT NULL AUTO\_INCREMENT,

F\_Name varchar(20) NOT NULL,

L\_Name varchar(20) NOT NULL,

Email varchar(50) NOT NULL,

Phone\_No bigint(15) NOT NULL,

Grad\_Year int(4) NOT NULL,

Faculty varchar(20) NOT NULL,

Degree\_Type char(10) NOT NULL,

Password varchar(180) NOT NULL,

PRIMARY KEY (Member\_ID),

UNIQUE (Email)

);

CREATE TABLE District(

District\_Name varchar(30) NOT NULL,

PRIMARY KEY (District\_Name)

);

CREATE TABLE POI(

District\_Name varchar(30) NOT NULL,

POI\_Name varchar (30) NOT NUll,

POI\_Description varchar(500) DEFAULT NULL,

PRIMARY KEY (District\_Name, POI\_Name),

FOREIGN KEY (District\_Name) references District(District\_Name)

);

CREATE TABLE Property(

Property\_ID int NOT NULL AUTO\_INCREMENT,

Owner\_ID int NOT NULL,

Street\_No int NOT NULL,

Street\_Name varchar(30) NOT NULL,

City varchar(30) NOT NULL,

Country varchar(30) NOT NULL,

Postal\_Code varchar(10) NOT NULL,

District\_Name varchar(30) NOT NULL,

Type varchar(15) NOT NULL,

Price decimal NOT NULL,

PRIMARY KEY (Property\_ID),

FOREIGN KEY (District\_Name) references District(District\_Name),

FOREIGN KEY (Owner\_ID) references Member(Member\_ID)

);

CREATE TABLE Feature(

Property\_ID int NOT NUll,

Feature\_Name varchar(100) NOT NULL,

Feature\_Description varchar(500) DEFAULT NUll,

PRIMARY KEY (Property\_ID, Feature\_Name),

FOREIGN KEY (Property\_ID) references Property(Property\_ID)

);

CREATE TABLE Booking(

Booking\_ID int NOT NULL AUTO\_INCREMENT,

Property\_ID int NOT NULL,

Member\_ID int NOT NULL,

Owner\_ID int NOT NULL,

Booking\_Start datetime NOT NULL,

Booking\_Status char(15) NOT NULL,

PRIMARY KEY (Booking\_ID, Property\_ID, Member\_ID),

FOREIGN KEY (Property\_ID) references Property(Property\_ID),

FOREIGN KEY (Member\_ID) references Member(Member\_ID),

FOREIGN KEY (Owner\_ID) references Property(Owner\_ID)

);

CREATE TABLE Comment(

Booking\_ID int NOT NUll,

Comment\_Time timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

Member\_ID int NOT NULL,

Rating int DEFAULT NULL,

Comment\_Text varchar(500) NOT NULL,

Owner\_Reply varchar(500) DEFAULT NULL,

PRIMARY KEY (Booking\_ID, Comment\_Time),

FOREIGN KEY (Booking\_ID) references Booking(Booking\_ID),

FOREIGN KEY (Member\_ID) references Member(Member\_ID)

);

## Initial Table Dumps











# Required SQL Statements & Sample Outputs

## Generic Statements (organized by user type)

### MEMBER

#Register as member

INSERT INTO `Member` (`F\_Name`,`L\_Name`,`Email`,`Phone\_No`,`Grad\_Year`,`Faculty`,`Degree\_Type`,`Password`,`Member\_ID`)

VALUES('value1','value2','value3',value4,value5,'value6','value7','value8',value9)

1. SIGN IN QUERY ????

#Update anything in profile

UPDATE `Member`

SET attribute1=value1,..

WHERE Member\_ID=theirMemberID

#Remove their membership

DELETE FROM `Member`

WHERE Member\_ID = theirMemberID

### Owner

#Create new accommodation

INSERT INTO `Property` (`Street\_No`,`Street\_Name`,`City`,`Country`,`Postal\_Code`,`District\_Name`,`Type`,`Price`,`Property\_ID`,`Owner\_ID`)

VALUES (value1,'value2','value3','value4','value5','value6','value7', value8, value9)

#Update existing accommodation

UPDATE `Property`

SET attribute1 = value1,...

WHERE Property\_ID = thePropertyID

#Remove accommodation

DELETE FROM `Property`

WHERE Property\_ID = thePropertyID

#Approve booking

UPDATE `Booking`

SET Booking\_Status = 'Approved'

WHERE Booking\_ID = theBookingID AND Booking\_Start = theBookingStart

#Reject booking

UPDATE `Booking`

SET Booking\_Status ='Rejected'

WHERE Booking\_ID = theBookingID AND BookingStart = theBookingStart

#Reply to comment

UPDATE `Comment`

SET Owner\_Reply = ownerCommentText

WHERE Member\_ID = commentersMemberID AND Booking\_ID = theBookingID AND Comment\_Time = theCommentTime

#List all owned properties + comments / ratings

SELECT Street\_No, Street\_Name, City, Price, Property\_ID, Comment\_Text, Rating,Owner\_Reply

FROM Property LEFT JOIN Comment

WHERE Owner\_ID = supplierMemberID

CONSUMER

#Search by district

SELECT DISTINCT Street\_No, Street\_Name, City, Country, District\_Name, Type, Price, Property\_ID

FROM Property

WHERE District\_Name = selectedDistrict

#Search by type

SELECT DISTINCT Street\_No,Street\_Name,City,Country,District\_Name,Type,Price,Property\_ID

FROM Property

WHERE Type = selectedType

#Search by features (CAN WE DO THIS?)

SELECT DISTINCT Street\_No, Street\_Name, City, Country, District\_Name, Type, Price, Property\_ID

FROM Property

WHERE Feature\_Name = selectedFeature

#Search by price

SELECT DISTINCT Street\_No, Street\_Name, City, Country, District\_Name, Type, Price, Property\_ID

FROM Property

WHERE Price <= maxPriceEntered AND Price >= minPriceEntered

#List all ratings and comments for a listing

SELECT DISTINCT Comment\_Text, Owner\_Reply, Rating

FROM Comment

WHERE Property\_ID = thePropertyID

ORDER BY Comment\_Time

#List availability of rental

SELECT Booking\_ID, Booking\_Status

FROM Booking

WHERE Booking\_Start = currentWeekStart AND Property\_ID = thePropertyID

#Show owner details

SELECT DISTINCT F\_Name,L\_Name,Email,Phone\_No

FROM Member NATURAL JOIN Property

WHERE Member\_ID = ownerOfSelectedProperty

#Place booking request (Booking is a table containing all booking for all properties)

INSERT INTO `Booking` (`Property\_ID`,`Booking\_Start`,`Booking\_Status`,`Member\_ID`,`Owner\_ID`)

VALUES (value1,value2,'Pending',value3,value4)

#List all my bookings

SELECT Street\_No,Street\_Name,City,Booking\_Start,Booking\_Status,Property\_ID

FROM Booking NATURAL JOIN Property

WHERE Member\_ID = consumerMemberID

GROUP BY Booking\_Status

ORDER BY Booking\_Start

#Show details of one booking

SELECT DISTINCT Street\_No, Street\_Name, City, Postal\_Code, District\_Name, Type, Feature\_Name, Price, Booking\_Start, Booking\_Status, Owner\_ID

FROM Booking NATURAL JOIN Property

WHERE Member\_ID = consumerMemberID AND Property\_ID = thePropertyID

#Add comment and rating for an accommodation

INSERT INTO `Comment` (`Member\_ID`,`Property\_ID`,`Rating`,`Comment\_Text`)

VALUES (commenterMemberID,thePropertyID,ratingValue,'commentText')

#Cancel a booking

DELETE `Booking`

WHERE Booking\_Start = theBookingStart AND Property\_ID = thePropertyID

### ADMIN

#Delete a member and associated listings

DELETE `Booking`

WHERE Member\_ID = theirMemberID OR Property\_ID in (SELECT Property\_ID

FROM Property

WHERE Owner\_ID = theirMemberID)

DELETE `Comment`

WHERE Member\_ID=theirMemberID OR Property\_ID in (SELECT Property\_ID

FROM Property

WHERE Owner\_ID = theirMemberID)

DELETE `Property`

WHERE Owner\_ID = theirMemberID

DELETE `Member`

WHERE Member\_ID = theirMemberID

#Delete an accommodation

DELETE `Booking`

WHERE Property\_ID=thePropertyID

DELETE `Comment`

WHERE Property\_ID=thePropertyID

DELETE `Feature`

WHERE Property\_ID=thePropertyID

DELETE `Property`

WHERE Property\_ID=thePropertyID

#Summarize bookings and ratings per accomodation

SELECT Booking\_Start, Booking\_Status, Avg(Rating)

FROM Booking NATURAL JOIN Comment

WHERE Property\_ID = thePropertyID

GROUP BY Booking\_Status

ORDER BY Booking\_Start

#Summarize bookings and ratings per supplier

SELECT Booking\_Start, Booking\_Status, Avg(Rating)

FROM Booking NATURAL JOIN Comment

WHERE Owner\_ID=supplierMemberID

GROUP BY Booking\_Status

ORDER BY Booking\_Start

#Summarize booking activity per customer

SELECT Booking\_Start, Booking\_Status

FROM Booking

WHERE Member\_ID = customerMemberID

GROUP BY Booking\_Status

ORDER BY Booking\_Start

## Demo Statements (organized by user type)

Note that the timestamps on comments may vary from those shown in the original table dumps because the database sometimes had to be dropped as part of the process in order to get the queries working.

### MEMBER

INSERT INTO `Member` (`F\_Name`,`L\_Name`,`Email`,`Phone\_No`,`Grad\_Year`,`Faculty`,`Degree\_Type`,`Password`)

VALUES ('Elon','Musk','e.musk@tesla.com',4567866787,1985,'Commerce','BComm','money')

UPDATE `Member`

SET Password = 'cash'

WHERE Member\_ID = 6

1. Note: Elon Musk was still in the table prior to this statement

DELETE FROM `Member`

WHERE Member\_ID = 6

1. INSERT INTO `Property` (`Street\_No`,`Street\_Name`,`City`,`Country`,`Postal\_Code`,`District\_Name`,`Type`,`Price`,`Owner\_ID`)

VALUES (15,'Wynd Drive','Toronto','Canada','M7T G6H','Forest Hill','House',1000, 2)

1. UPDATE `Property`

SET Price = 1500

WHERE Property\_ID = 6

1. ☹
2. UPDATE `Booking`

SET Booking\_Status = 'Approved'

WHERE Property\_ID = 4 AND Booking\_Start = "2016-06-11 12:30:00"

1. UPDATE `Booking`

SET Booking\_Status='Rejected'

WHERE Booking\_ID = 2 AND Booking\_Start = "2016-06-11 12:30:00"

1. UPDATE `Comment`

SET Owner\_Reply = "Glad you liked it!"

WHERE Member\_ID = 3 AND Booking\_ID = 5 AND Comment\_Time = "2016-03-08 22:33:41"

1. ☹
2. SELECT DISTINCT Street\_No,Street\_Name,City,Country,District\_Name,Type,Feature\_Name,Price,Property\_ID

FROM Property

WHERE District\_Name = "Entertainment District"

1. SELECT DISTINCT Street\_No,Street\_Name,City,Country,District\_Name,Type,Price,Property\_ID

FROM Property

WHERE Type = "Loft"

1. ☹
2. SELECT DISTINCT Street\_No,Street\_Name,City,Country,District\_Name,Type,Price,Property\_ID

FROM Property

WHERE Price<=75 AND Price>=15

1. ☹
2. SELECT Booking\_ID, Booking\_Status

FROM Booking

WHERE Booking\_Start = "2016-06-11 12:30:00" AND Property\_ID = 4

1. SELECT DISTINCT F\_Name, L\_Name, Email, Phone\_No

FROM Member NATURAL JOIN Property

WHERE Member\_ID = 2

1. INSERT INTO `Booking` (`Property\_ID`,`Booking\_Start`,`Booking\_Status`,`Member\_ID`,`Owner\_ID`)

VALUES (2,'2016-08-17 16:30:00','Pending',5,2)

1. SELECT Booking\_ID, Street\_No,Street\_Name,City,Booking\_Start,Booking\_Status,Property\_ID

FROM Booking NATURAL JOIN Property

WHERE Booking.Member\_ID = 3

GROUP BY Booking\_Status

ORDER BY Booking\_Start

1. ☹ Need features
2. INSERT INTO `Comment` (`Booking\_ID`,`Member\_ID`,`Rating`,`Comment\_Text`)

VALUES (6,5,5,'Never wanted to leave! Wonderful home.')

1. ☹
2. ☹
3. ☹
4. SELECT Booking\_Start,Booking\_Status, Avg(Rating)

FROM Booking NATURAL JOIN Comment

WHERE Property\_ID = 4

GROUP BY Booking\_Status

ORDER BY Booking\_Start

1. SELECT Booking\_Start,Booking\_Status, Avg(Rating)

FROM Booking NATURAL JOIN Comment

WHERE Owner\_ID = 4

GROUP BY Booking\_Status

ORDER BY Booking\_Start

1. SELECT Booking\_Start,Booking\_Status

FROM Booking

WHERE Member\_ID = 2

GROUP BY Booking\_Status

ORDER BY Booking\_Start

# Required SQL Statements & Sample Outputs

# State Machine Diagram

# User Interface Flow

## Main Page

From the Main Page, there are a few options for the user to choose from:

* Browse through the main page and view additional information on QBnB
* Enter their email into the displayed textboxes and click “Sign Me Up!”
  + This option will take them to the Sign Up Page
* Select the “Already a Member” option to sign into QBnB
  + This option will take them to the Sign In Page

Users are not able to access this page if they are logged in, and it is the main page when accessing the website via the sites’ domain.

## Sign Up Page

From the Sign Up Page, the user is able to enter their information:

* Upon entering valid information in all fields of the page, the user can select Sign Up, and will be taken to the User Dashboard
* If the user does not complete all fields, or tries to create an account with an email that is already in use, the sign up process will not complete and the user will be required to enter all required, or different information

From this page, the user is also able to access the Sign In Page by clicking “Already a Member”, or to return to the home page by clicking the QBnB logo.

## Sign In Page

From the Sign In Page, the user is able to enter their account credentials:

* If they enter a valid email and password, the user is taken to the User Dashboard
* If they enter incorrect information, the user is prompted to try again with correct information

If the user does not want to sign in, they are able to return to the Main Page by clicking the QBnB logo.

## User Dashboard

From the User Dashboard, there are a variety of options available to the user.

* The user may click on “Settings

## Settings Page

From the Settings Page, the user is able to update any parts of their account information. Once they have entered the new information, they can select the confirm button, and may repeat this if desired. Once the user is done updating their information, they can be taken back to the User Dashboard by clicking “Go Back”.

## Administrator Dashboard

Within the Administrator Dashboard, administrators have several options and controls at their disposal. The Administrator Dashboard allows Admins to:

* Remove a given accommodation from the list of all properties (and all associated features, comments and bookings)
* Remove a given user from the list of all members (and all associated accommodations, comments and bookings)
* View all bookings made by a given user
* View all bookings, comments and ratings made on a given property
* View all bookings, comments and ratings made on all properties listed by a given supplier

Additionally, the Administrators are able to leave the Administrator Dashboard and return to the User Dashboard.