design.pdf

Jessica McManus, Alexander Haufe, Aleksei Weinberg
CSC 460
Program 4

1. updated ER diagram with constraints (its updated but might go in the tar file instead of here)
   One of our main constraints with the ER diagram was our lack of ability to show foreign key relationships. It made it difficult to figure out how to connect and label different aspects. Furthermore, we had a lot of Tables that had similar relations to each other, so naming relation attributes became difficult.

2. our schema (dropCreateTable.sql)

TotalOrder

orderID	memberID	reservationID	orderTime	totalPrice	payment Status	orderDate	orderStatus

MenuItem

menuID	orderID	name	price

Member

memberID	lastName	firstName	DoB	email	membershipTier	emergencyContactAreaCode	emergency Contact	phoneNoAreaCode	phoneNo

EventBooking

bookingID	custID	bookDate	eventID	attendance Status	paymentStatus	membershipTier

Reservation

reservationID	customerID	roomID	resDate	startTime	duration	inStatus	membershipTier

Room

roomID	maxCapacity	petType	purpose	location

Event

eventID	eventName	description	roomID	empID	maxCapacity	eventDate	time	attendance Fee	regCust

Employee

empID	firstName	lastName	position

HealthRecord

recordID	petID	employeeID	recordDate	recordType	description	nextDueDate	validity	invalidReason

Pet

petID	name	species	breed	age	arrivalDate	temperament	spNeeds	currStat

Adoption

appID	custID	petID	empID	appDate	status	price

3. provide all FDs and justify why the tables adhere to 3NF/BCNF
   TotalOrder {orderID -> orderID, memberID, reservationID, orderTime, totalPrice, paymentStatus, orderDate, orderStatus}

   MenuItem {menuID -> menuID, orderID, name, price}

   Member {memberID -> memberID, lastName, firstName, DoB, email, membershipTier, emergencyContactAreaCode, emergencyContact phoneNoAreaCode, phoneNo}

   EventBooking { bookingID -> bookingID, custID, bookDate, eventID, attendanceStatus, paymentStatus, membershipTier}

   Reservation { reservationID -> reservationID, customerID, roomID, resDate, startTime, duration, inStatus, membershipTier}

   Room { roomID -> roomID, maxCapacity, petType, purpose, location}

   Event { eventID -> eventID, eventName, description, roomID, empID, maxCapacity, eventDate, time, attendanceFee, regCust}

Employee {empID -> empID, firstName, lastName, position}

HealthRecord { recordID-> recordID, petID, employeeID, recordDate, recordType. Description, nextDueDate, validity, invalidReason}

Pet {petID -> petID, name, species, breed, age, arrivalDate, temperament, spNeeds, currStat}

Adoption {appID -> appID, custID, petID, empID, appDate, status, price}

Our database is in both third normal form and Boyce-Codd normal form. We were careful not to store anything that could be calculated from another such as membershipTier and membershipDiscount. This prevented us from having something be on the left side of the functional dependencies other than the primary key. As the only attributes on the left side of any of our functional dependencies are primary keys, the database is in both 3NF and BCNF.

4. Description of query 4:

   Query 4 prompts the user to input an employee’s first and last name. The query uses the Employee and Adoption tables to connect the employee’s name to the pending adoption applications assigned to them. The query then uses the Adoption and Pet tables to determine the name, species and breed of the pets with pending adoption forms assigned to that employee. Then the Query prints the results.

ReadMe.txt

Jessica McManus, Alexander Haufe, Aleksei Weinberg
CSC 460
Program 4

Compilation and execution instructions

Add the Oracle JDBC driver to your CLASSPATH environment variable:

```
   export CLASSPATH=/usr/lib/oracle/19.8/client64/lib/ojdbc8.jar:${CLASSPATH}
```

The code can be compiled by compiling all java files in the directory with;

```
javac ./*.java
```

The “Main” code to execute is Prog4.java, it takes no arguments when ran and can be ran with;

```
java ./Prog4
```

When running instructions will be on the terminal and most required inputs are done through a number selection.

workload distribution (who did what)

For Workload distribution, everyone helped with planning, ER-Diagram, and some form of code testing.

Alexander Haufe coded the initial schema and csv reader to populate the tables. He made queries 3 and 4. He also made the logic that deletes from the database.

Aleksei Weinberg did code for the main method in Prog4.java, as well as query 1, modifytable, and all the methods in common.java

Jessica McManus primarily did the code for AddToTable, runQuery(), and query 2, as well as working on the schema in the design pdf.