**ECE 567 Project SQL Design**

**Database Design and Management**

**Grant Kimes**

**5/2/16**

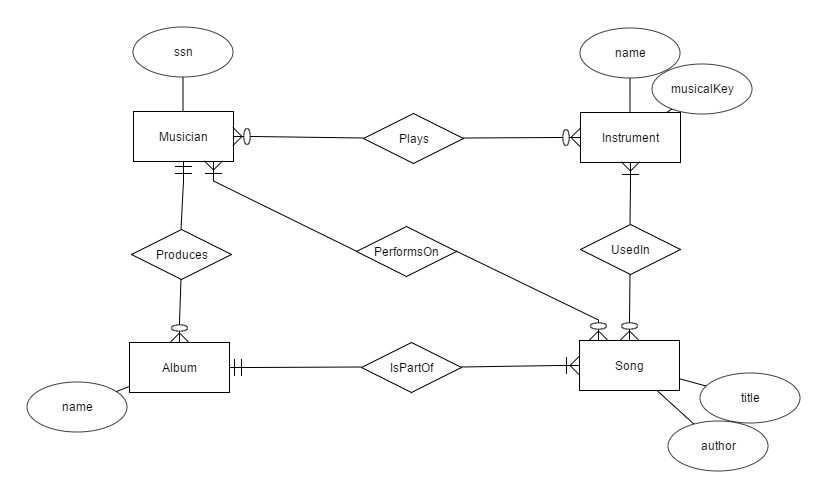
**Final Product**

In MusicianQueries.java, the user is given a prompt for all the sample queries that can be executed. They then choose a given query number, and are prompted for any additional information necessary for the query. The query then executes, and the user is shown the results. They are then back to the command line and additional queries can be performed.

The user is not expected to know SQL statements in order to perform these operations. They are prompted in simple English, and the program manages all the query syntax for them.

**Project Stages**

The initial conceptual model was not quite correct. Some of the relations given would not have allows for certain sample queries to be performed. Address was initially an attribute of Musician, and this lead to transitive functional dependency, so eventually Address was moved to a new table. The relation between instrument and song was unecessary. It is not needed to know which instruments were used on which songs.



*Initial Conceptual Model*

The logical model greatly improved upon this design, by creating more clear relations between Musician, Song, and Instrument. A musician can play on many songs, and a song can have many musicians. A musician can play many instruments, and an instrument can be played by many musicians. So this required a separate table for each of these relations. This model clarified exactly which relations would be involved with each other and which would have foreign keys.

C:\Users\Grant\Downloads\ECE567 Musician ER Diagram.png

*Logical Model*

**Table Creation**

CREATE TABLE Musician (

musicianNo int,

ssn int,

fName varchar(64),

lName varchar(64),

phoneNum int,

PRIMARY KEY (musicianNo),

FOREIGN KEY (phoneNum) REFERENCES Place (phoneNum)

);

CREATE TABLE Place (

phoneNum int,

street varchar(64),

city varchar(64),

state varchar(64),

zipCode int,

PRIMARY KEY (phoneNum)

);

CREATE TABLE Instrument (

instrumentNo int,

name varchar(64),

musicalKey varchar(64),

PRIMARY KEY (instrumentNo)

);

CREATE TABLE Ability (

musicianNo int,

instrumentNo int,

PRIMARY KEY (musicianNo, instrumentNo),

FOREIGN KEY (musicianNo) REFERENCES Musician (musicianNo),

FOREIGN KEY (instrumentNo) REFERENCES Instrument (instrumentNo)

);

CREATE TABLE Song (

songNo int,

title varchar(64),

author varchar(64),

albumNo int,

PRIMARY KEY (songNo),

FOREIGN KEY (albumNo) REFERENCES Album (albumNo)

);

CREATE TABLE Performance (

musicianNo int,

songNo int,

PRIMARY KEY (musicianNo, songNo),

FOREIGN KEY (musicianNo) REFERENCES Musician (musicianNo) ON DELETE CASCADE,

FOREIGN KEY (songNo) REFERENCES Song (songNo) ON DELETE CASCADE

);

CREATE TABLE Album (

albumNo int,

title varchar(64),

copyrightDate date,

format varchar(64),

producerNo int,

PRIMARY KEY (albumNo),

FOREIGN KEY (producerNo) REFERENCES Musician (musicianNo)

);

**Sample Data**

INSERT INTO Musician VALUES (1, 122223, 'Grant', 'Kimes', 2223331111);

INSERT INTO Musician VALUES (2, 334433, 'Kieran', 'Harrison', 2223331111);

INSERT INTO Musician VALUES (3, 123321, 'Emily', 'Kordick', 4442229999);

INSERT INTO Place VALUES (2223331111, '123 Home St.', 'Chicago', 'IL', 60011);

INSERT INTO Place VALUES (4442229999, '34 Walk Dr', 'Miami', 'FL', 11223);

INSERT INTO Instrument VALUES (1, 'Saxophone', 'E');

INSERT INTO Instrument VALUES (2, 'Saxophone', 'B flat');

INSERT INTO Instrument VALUES (3, 'Piano', 'C');

INSERT INTO Instrument VALUES (4, 'Clarinet', 'A flat');

INSERT INTO Ability VALUES (1, 1);

INSERT INTO Ability VALUES (1, 2);

INSERT INTO Ability VALUES (1, 3);

INSERT INTO Ability VALUES (2, 4);

INSERT INTO Ability VALUES (3, 3);

INSERT INTO Ability VALUES (3, 4);

INSERT INTO Album VALUES (1, 'The First Album', DATE '2010-01-01', 'CD', 1);

INSERT INTO Album VALUES (2, 'Two is a Date', DATE '2011-01-01', 'CD', 1);

INSERT INTO Album VALUES (3, 'Now We Are Talking', DATE '2011-02-02', 'CD', 2);

INSERT INTO Album VALUES (4, 'Jeezy', DATE '2011-03-03', 'CD', 3);

INSERT INTO Song VALUES (1, 'Numba One', 'Grakim', 1);

INSERT INTO Song VALUES (2, 'Johnny is Home', 'Grakim', 1);

INSERT INTO Song VALUES (3, 'The Crew', 'Grakim', 2);

INSERT INTO Song VALUES (4, 'One of the Best Songs', 'Grakim', 2);

INSERT INTO Song VALUES (5, 'Still Improving', 'Jenkins', 3);

INSERT INTO Song VALUES (6, 'Finale', 'Jenkins', 4);

INSERT INTO Performance VALUES (1, 1);

INSERT INTO Performance VALUES (1, 2);

INSERT INTO Performance VALUES (1, 3);

INSERT INTO Performance VALUES (1, 4);

INSERT INTO Performance VALUES (2, 5);

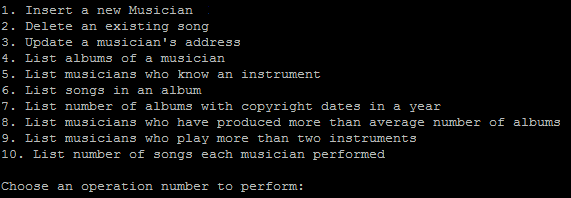
INSERT INTO Performance VALUES (3, 1);

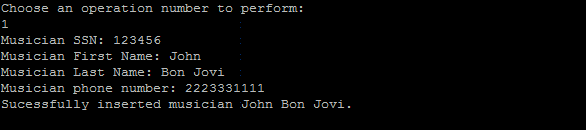
INSERT INTO Performance VALUES (3, 2);

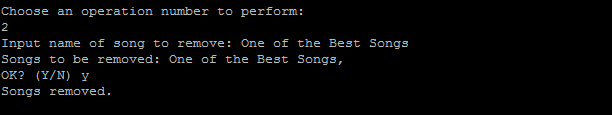
INSERT INTO Performance VALUES (3, 6);

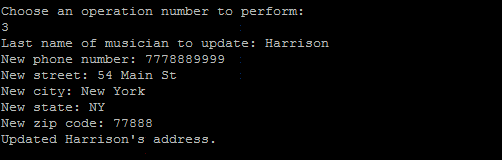
**Program Output for Sample Queries**

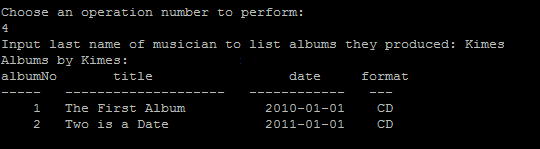
List of queries

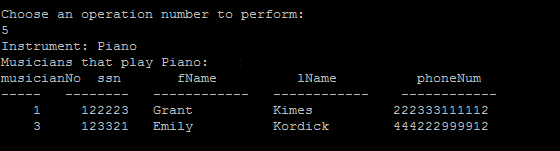


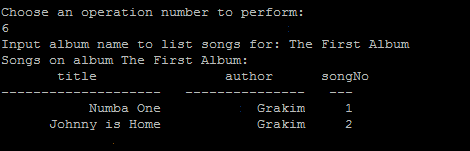


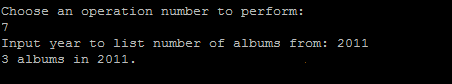


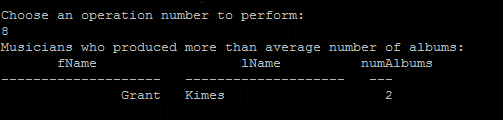


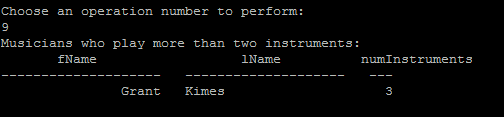


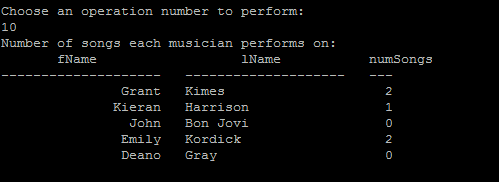












**Java Embedded SQL Source Code (MusicianQueries.java)**

**Available in home directory of grki567**

/\* Grant Kimes

\* ECE 567 Database Design

\* Project: Musician Queries

\*/

import java.io.\*;

import java.sql.\*;

import oracle.jdbc.\*;

import oracle.jdbc.pool.OracleDataSource;

class MusicianQueries

{

private static final String USERNAME = "grki567";

private static final String PASSWORD = "c09680";

private static final String HOST = "localhost"; // Host

private static final String PORT = "1521"; // Default port

private static final String SID = System.getenv("ORACLE\_SID"); // Oracle SID

private static Connection conn;

private static BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));

public static void main (String args[]) throws SQLException

{

OracleDataSource ods = new OracleDataSource();

ods.setUser(USERNAME);

ods.setPassword(PASSWORD);

ods.setURL("jdbc:oracle:thin:" + "@" + HOST + ":" + PORT + ":" + SID);

conn = ods.getConnection();

System.out.print("\n1. Insert a new Musician\n"

+"2. Delete an existing song\n"

+"3. Update a musician's address\n"

+"4. List albums of a musician\n"

+"5. List musicians who know an instrument\n"

+"6. List songs in an album\n"

+"7. List number of albums with copyright dates in a year\n"

+"8. List musicians who have produced more than average number of albums\n"

+"9. List musicians who play more than two instruments\n"

+"10. List number of songs each musician performed\n");

while (true) {

try {

String input;

System.out.println("\nChoose an operation number to perform: ");

input = stdIn.readLine();

switch(input) {

case "1":

insertMusician();

break;

case "2":

removeSong();

break;

case "3":

updateAddress();

break;

case "4":

listAlbums();

break;

case "5":

listInstrumentPlayers();

break;

case "6":

listAlbumSongs();

break;

case "7":

listAlbumsCopyrightYear();

break;

case "8":

listMoreThanAvgAlbums();

break;

case "9":

listMultipleInstrumentPlayers();

break;

case "10":

listNumSongsPerformed();

break;

default:

System.out.println("Not a valid input number.");

break;

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

private static void insertMusician() {

try {

System.out.print("Musician SSN: ");

String ssn2 = stdIn.readLine();

int ssn = Integer.parseInt(ssn2);

System.out.print("Musician First Name: ");

String fName = stdIn.readLine();

System.out.print("Musician Last Name: ");

String lName = stdIn.readLine();

System.out.print("Musician phone number: ");

String phoneNum2 = stdIn.readLine();

long phoneNum = Long.parseLong(phoneNum2);

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT COUNT(\*) FROM Musician; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

cursor.next();

int newMusicianNo = cursor.getInt(1) + 1;

Statement stmt = conn.createStatement();

String sql = "INSERT INTO Musician VALUES ("+newMusicianNo+","+ssn+",'"+fName+"','"+lName+"',"+phoneNum+")";

stmt.executeUpdate(sql);

System.out.println("Sucessfully inserted musician " + fName + " " + lName + ".");

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to insert musician.");

}

}

private static void removeSong() {

try {

System.out.print("Input name of song to remove: ");

String songName = stdIn.readLine();

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT title FROM Song WHERE title LIKE '%"+songName+"%'; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.print("Songs to be removed: ");

while (cursor.next ()) {

System.out.printf("%s, ", cursor.getString(1));

}

System.out.print("\nOK? (Y/N) ");

String answer = stdIn.readLine();

if (answer.equalsIgnoreCase("Y")) {

Statement stmt = conn.createStatement();

String sql = "DELETE FROM Song WHERE title LIKE '%"+songName+"%'";

stmt.executeUpdate(sql);

System.out.println("Songs removed.");

}

else {

System.out.println("Not removing songs.");

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to remove song.");

}

}

private static void updateAddress() {

try {

System.out.print("Last name of musician to update: ");

String lName = stdIn.readLine();

System.out.print("New phone number: ");

long phoneNum = Long.parseLong(stdIn.readLine());

System.out.print("New street: ");

String street = stdIn.readLine();

System.out.print("New city: ");

String city = stdIn.readLine();

System.out.print("New state: ");

String state = stdIn.readLine();

System.out.print("New zip code: ");

int zipCode = Integer.parseInt(stdIn.readLine());

Statement stmt = conn.createStatement();

String sql = "INSERT INTO Place VALUES ("+phoneNum+", '"+street+"', '"+city+"', '"+state+"', "+zipCode+")";

stmt.executeUpdate(sql);

sql = "UPDATE Musician SET phoneNum="+phoneNum+" WHERE lName='"+lName+"'";

stmt.executeUpdate(sql);

System.out.println("Updated "+lName+"'s address.");

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to update address.");

}

}

private static void listAlbums() {

try {

System.out.print("Input last name of musician to list albums they produced: ");

String lName = stdIn.readLine();

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT albumNo, title, copyrightDate, format FROM Album a, Musician m "

+"WHERE lName='"+lName+"' AND producerNo=musicianNo; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Albums by "+lName+": ");

System.out.println("albumNo title date format");

System.out.println("----- -------------------- ------------ --- ");

while (cursor.next ()) {

System.out.printf("%5d %-20s %12s %3s\n", cursor.getInt(1), cursor.getString(2), cursor.getDate(3), cursor.getString(4));

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list albums.");

}

}

private static void listInstrumentPlayers() {

try {

System.out.print("Instrument: ");

String instrument = stdIn.readLine();

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT DISTINCT m.musicianNo, m.ssn, m.fName, m.lName, m.phoneNum FROM Musician m, Ability a, Instrument i "

+"WHERE a.instrumentNo=i.instrumentNo AND i.name='"+instrument+"' AND m.musicianNo=a.musicianNo "

+"ORDER BY m.lName; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Musicians that play " + instrument + ": ");

System.out.println("musicianNo ssn fName lName phoneNum");

System.out.println("----- -------- ------------ ------------ ------------");

while (cursor.next ()) {

System.out.printf("%5d %8d %-12s %-12s %d12 \n", cursor.getInt(1), cursor.getInt(2), cursor.getString(3), cursor.getString(4), cursor.getLong(5));

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list players of given instrument.");

}

}

private static void listAlbumSongs() {

try {

System.out.print("Input album name to list songs for: ");

String albumName = stdIn.readLine();

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT s.title, s.author, s.songNo FROM Song s, Album a "

+"WHERE s.albumNo=a.albumNo AND a.title='"+albumName+"'; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Songs on album " + albumName + ": ");

System.out.println(" title author songNo");

System.out.println("-------------------- --------------- ---");

while (cursor.next ()) {

System.out.printf("%20s %15s %3d \n", cursor.getString(1), cursor.getString(2), cursor.getInt(3) );

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list songs on album.");

}

}

private static void listAlbumsCopyrightYear() {

try {

System.out.print("Input year to list number of albums from: ");

int year = Integer.parseInt(stdIn.readLine());

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT COUNT(\*) FROM Album "

+"WHERE EXTRACT(YEAR FROM copyrightDate)="+year+"; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

cursor.next();

System.out.println(cursor.getInt(1) + " albums in " + year + ".");

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list albums copyrighted in a year.");

}

}

private static void listMoreThanAvgAlbums() {

try {

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT fName, lName, COUNT(producerNo) AS numAlbums FROM Musician m, Album a "

+"WHERE m.musicianNo=a.producerNo GROUP BY lName, fName "

+"HAVING COUNT(producerNo)>(SELECT CAST(COUNT(albumNo) AS DECIMAL) / COUNT(DISTINCT producerNo) FROM Album); end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Musicians who produced more than average number of albums: ");

System.out.println(" fName lName numAlbums");

System.out.println("-------------------- -------------------- ---");

while (cursor.next ()) {

System.out.printf("%20s %-20s %3d \n", cursor.getString(1), cursor.getString(2), cursor.getInt(3) );

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list players who produced more than average number of albums.");

}

}

private static void listMultipleInstrumentPlayers() {

try {

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT fName, lName, COUNT(instrumentNo) AS numInstruments FROM Musician m "

+"LEFT JOIN Ability a ON m.musicianNo=a.musicianNo GROUP BY lName, fName HAVING COUNT(instrumentNo)>2; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Musicians who play more than two instruments: ");

System.out.println(" fName lName numInstruments");

System.out.println("-------------------- -------------------- ---");

while (cursor.next ()) {

System.out.printf("%20s %-20s %3d \n", cursor.getString(1), cursor.getString(2), cursor.getInt(3) );

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed to list multiple instrument players.");

}

}

private static void listNumSongsPerformed() {

try {

CallableStatement query;

ResultSet cursor;

query = conn.prepareCall("BEGIN OPEN ? FOR SELECT fName, lName, COUNT(songNo) AS numSongs FROM Musician m "

+"LEFT JOIN Performance p ON m.musicianNo=p.musicianNo GROUP BY fName, lName; end;");

query.registerOutParameter(1, OracleTypes.CURSOR);

query.execute();

cursor = ((OracleCallableStatement)query).getCursor(1);

System.out.println("Number of songs each musician performs on: ");

System.out.println(" fName lName numSongs");

System.out.println("-------------------- -------------------- ---");

while (cursor.next ()) {

System.out.printf("%20s %-20s %3d \n", cursor.getString(1), cursor.getString(2), cursor.getInt(3) );

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("Failed in listing number of songs of each musician.");

}

}

}