Marist College Music Department

A Database Design Proposal

By Andrew Bauman

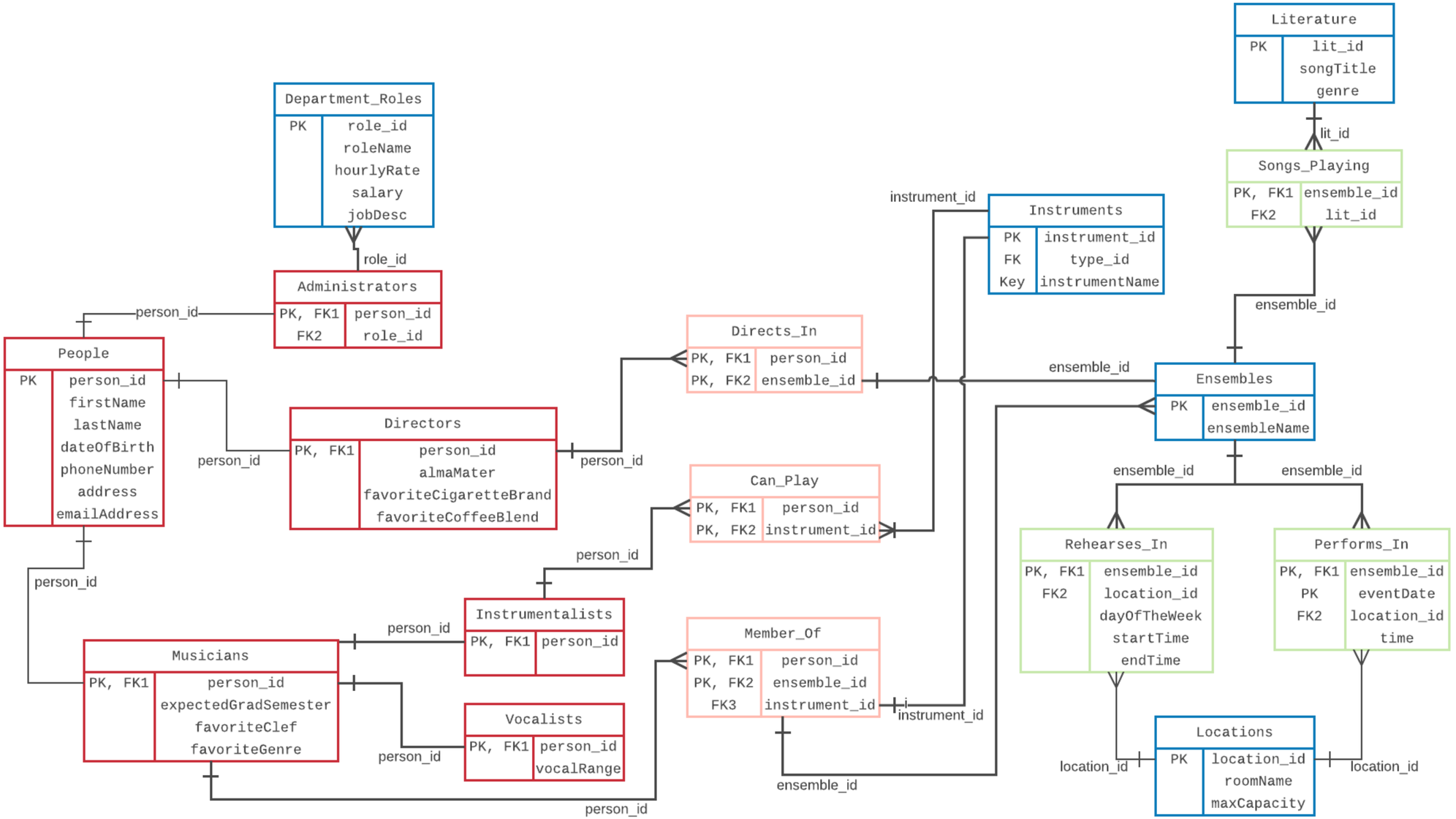
CMPT 308 - Fall 2017

Table of Contents

The Marist Music Department is comprised of both instrumental and vocal performance ensembles containing nearly 400 students.

The purpose of this database system is to simplify the process of managing many of the daily bookkeeping tasks within the Marist College Music Department to afford the department the ability of remaining focused on producing music rather than becoming distracted by day-to-day tasks.

Executive Summary



Entity-Relationship Diagram

People

Stores the general information for the people within the database.

Fundamental Dependencies

person_id → firstName, lastName, dateOfBirth, phoneNumber, address, emailAddress

```
CREATE TABLE People (
  person_id        char(8) not null,
  firstName        text not null,
  lastName         text not null,
  dateOfBirth        date,
  phoneNumber        text,
  address            text,
  emailAddress     text not null,
 primary key(person_id)
);
```

person_id	firstname	lastname	dateofbirth	phonenumber	address	emailaddress
20083100	Andrew	Bauman	1998-03-01	845-123-4567	3399 North Road	andrew.bauman1@marist.edu
20083101	Marsha	Mellow	1999-05-14	660-948-9199	132 Market Lane	m.mellow@email.com
20083102	Cadence	Smith	1997-08-13	309-962-4101	2455 Spring Row	csmith@gmail.com
20083103	Adagio	Turner	1997-04-28	330-569-9560	24 Phoenix Route	adagio@gmail.com
20083104	Melody	Wind	1998-02-14	571-814-6974	49 Kingwood Passage	MelodyWind123@gmail.com
20083105	I.M.	Tired	1998-12-25	202-956-8464	13 Orchard Lane	TiredIm@gmail.com
20083106	Rita	Book	1999-01-24	401-534-7890	464 South Row	ritab@yahoo.com
20083107	Isabelle	Ringing	1999-05-20	304-914-5975	90 Bridgewater Lane	bellsRinging@aol.com
20083108	Maureen	Biologist	1996-03-23	305-668-9127	24 Grime Lane	dolphin6132@hotmail.com
20083109	Ann	Chovey	1999-07-21	762-222-4549	242 Royalty Boulevard	choveya@ibm.com
20083110	Johnny	Cash	1998-05-21	610-570-7873	24 Polygon Avenue	jcash@gmail.com
30042001	Alan	Labouseur	1980-01-01	845-575-3832	191 Marist Drive	alan@labouseur.com
30042002	Art	Himmelberger	1953-11-28	845-575-3000 ext. 2839	3399 North Road	arthur.himmelberger@marist.edu
30042003	Sarah	Williams	1975-04-24	845-575-3000 ext. 2944	3399 North Road	sarah.williams@marist.edu
30042004	Mikey	Napolitano	1983-03-23	845-575-3000 ext. 2142	3399 North Road	michael.napolitano@marist.edu
30042005	MusicStaff	MemberOne	1999-02-23	845-575-3000	3399 North Road	music@marist.edu

(16 rows)

Create Table Statements

Musicians

Includes those who perform in ensembles.

Fundamental Dependencies

person_id → expectedGradSemester, favoriteClef, favoriteGenre

person_id	expectedgradsemester	favoriteclef	favoritegenre
20083100	Spring 2020	Bass	Funk
20083101	Fall 2019	Treble	Pop
20083102	Spring 2018	Tenor	Marches
20083103	Spring 2021	Treble	Classical
20083104	Fall 2020	Bass	Jazz
20083105	Fall 2020	TAB	Hard Rock
20083106	Spring 2019	Treble	Swing
20083107	Fall 2018	Bass	Soft Rock
20083108	Spring 2018	Bass	Jazz
20083109	Fall 2021	Treble	Pop
20083110	Spring 2019	Tenor	Metal

```
CREATE TABLE Musicians (
  person_id             char(8) not null references People(person_id),
  expectedGradSemester  text,
  favoriteClef          text,
  favoriteGenre         text,
 primary key(person_id)
);
```

Create Table Statements

Instrumentalists

References those who perform
in band ensembles.

```
 person_id
-----------
 20083100
 20083101
 20083102
 20083103
 20083104
 20083105
```

```
CREATE TABLE Instrumentalists (
  person_id  char(8) not null references Musicians(person_id),
 primary key(person_id)
);
```


Vocalists

References those who perform
in vocal ensembles

Fundamental Dependencies
person_id → **vocalRange**

person_id	vocalrange
20083106	Tenor
20083107	Soprano
20083108	Baritone
20083109	Alto
20083101	Baritone
20083110	Alto

```
CREATE TABLE Vocalists (
  person_id     char(8) not null references Musicians(person_id),
  vocalRange    text,
 primary key(person_id)
);
```


Directors

person_id	almamater	favoritecigarettebrand	favoritecoffeeblend
30042002	University of Michigan	Lucky Strike Menthol	black
30042003	Music U		Starbucks Holiday Blend
30042001	Marist College		

Includes those who direct ensembles.

Fundamental Dependencies

person_id → almaMater, favoriteCigaretteBrand, favoriteCoffeeBlend

```
CREATE TABLE Directors (
  person_id                 char(8) not null references People(person_id),
  almaMater                 text,
  favoriteCigaretteBrand    text,
  favoriteCoffeeBlend       text,
 primary key(person_id)
);
```

Create Table Statements

Department_Roles

Stores the possible roles and salaries office staff may have

Fundamental Dependencies

role_id → **roleName, hourlyRate, salary, jobDesc**

```
CREATE TABLE Department_Roles (
  role_id        char(4) not null,
  roleName       text,
  hourlyRate     numeric(10,2),
  salary         numeric(10,2),
  jobDesc        text,
 primary key (role_id)
);
```

role_id	rolename	hourlyrate	salary	jobdesc
0001	Director of Music and Director of Bands		73000.00	Responsible for the management and direction of the Marist Band and Music Department
0002	Director of Choral Activities		75000.00	Responsible for the management and direction of the Marist Singers Program and related ensembles
0003	Operations Manager	18.00		Manages the operations regarding the Marist Music Department

Administrators

Includes those who hold roles within the department.

person_id	role_id
30042002	0001
30042003	0002
30042004	0003

Fundamental Dependencies

person_id → role_id

```
CREATE TABLE Administrators (
  person_id char(8) not null references People(person_id),
  role_id   char(4) not null references Department_Roles(role_id),
 primary key(person_id)
);
```


Ensembles

Includes the ensembles offered within the department.

Fundamental Dependencies

ensemble_id → **ensembleName**

```
CREATE TABLE Ensembles (
  ensemble_id   char(5) not null,
  ensembleName  text,
 primary key(ensemble_id)
);
```

ensemble_id	ensemblename
00001	Symphonic Band
00002	Wind Symphony
00003	Orchestra
00004	Jazz Foxes
00005	Brass Ensemble
00006	Flute Choir
00007	Marist Singers
00008	Time Check
00009	Sirens
00010	Pit Band
00011	Computer Society

Instruments

Includes the instruments which students can play in the ensembles

Fundamental Dependencies

instrument_id → instrumentName, instrumentFamily

```
CREATE TABLE Instruments (
  instrument_id      char(4) not null,
  instrumentName     text not null,
  instrumentFamily   text not null,
 primary key(instrument_id)
);
```

instrument_id	instrumentname	instrumentfamily
0001	Euphonium	Brass
0002	Trombone	Brass
0003	Clarinet	Woodwind
0004	Flute	Woodwind
0005	Drums	Percussion
0006	Harpsichord	Keyboards
0007	Tenor Sax	Woodwind
0008	Baritone Sax	Woodwind
0009	Apple II	Keyboards
0010	Marimba	Percussion

Member_Of

Determines who is in which ensemble and if they play an instrument in it.

Fundamental Dependencies

person_id, ensemble_id → **instrument_id**

person_id	ensemble_id	instrument_id
20083100	00001	0001
20083100	00002	0002
20083101	00003	0004
20083101	00006	0004
20083101	00002	0004
20083102	00001	0003
20083102	00002	0003
20083102	00003	0003
20083103	00003	0006
20083104	00001	0008
20083104	00002	0008
20083104	00003	0008
20083105	00004	0010
20083106	00007	
20083107	00008	
20083107	00007	
20083108	00009	
20083108	00007	
20083109	00010	
20083109	00007	
20083110	00007	

```
CREATE TABLE Member_of (
  person_id     char(8) not null references Musicians(person_id),
  ensemble_id   char(5) not null references Ensembles(ensemble_id),
  instrument_id char(4) references Instruments(instrument_id),
 primary key(person_id, ensemble_id)
);
```

Create Table Statements

Can_Play

Stores which instruments each instrumentalist can play

Fundamental Dependencies

person_id → instrument_id

person_id	instrument_id
20083100	0001
20083100	0002
20083101	0004
20083102	0003
20083103	0006
20083104	0008
20083105	0010

```
CREATE TABLE Can_Play (
  person_id          char(8) not null references Instrumentalists(person_id),
  instrument_id      char(4) not null references Instruments(instrument_id),
 primary key(person_id, instrument_id)
);
```

Create Table Statements

Directs_In

References which people direct ensembles

Fundamental Dependencies

person_id → ensemble_id

person_id	ensemble_id
30042002	00001
30042002	00002
30042002	00003
30042002	00004
30042002	00005
30042002	00006
30042002	00010
30042003	00007
30042003	00008
30042003	00009
30042001	00011

```
CREATE TABLE Directs_In (
  person_id     char(8) not null references Directors(person_id),
  ensemble_id   char(5) not null references Ensembles(ensemble_id),
 primary key(person_id, ensemble_id)
);
```

Create Table Statements

Locations

Stores the locations where ensembles can perform and rehearse in.

Fundamental Dependencies

location_id → **roomName, maxCapacity**

```
CREATE TABLE Locations (
  location_id   char(4) not null,
  roomName      text,
  maxCapacity   text,
 primary key(location_id)
);
```

location_id	roomname	maxcapacity
0001	Symphonic Hall	410
0002	Fusco Recital Hall	314
0003	Nelly Golleti Theatre	400
0004	River Rooms	350
0005	Bardavon	800
0006	CIA Recital Hall	900
0007	Longview Park	1000
0008	The Caberet	210
0009	HC2020	40
0010	Small Ensembles Room	100

Rehearses_In

References the locations where ensembles rehearse.

Fundamental Dependencies

ensemble_id → location_id, dayOfTheWeek, startTime, endTime

```
 ensemble_id | location_id | dayoftheweek | starttime | endtime
-------------+-------------+--------------+-----------+----------
 00001       | 0001        | Monday       | 20:00:00  | 22:00:00
 00002       | 0001        | Thursday     | 21:00:00  | 23:00:00
 00003       | 0001        | Wednesday    | 18:30:00  | 20:30:00
 00004       | 0010        | Wednesday    | 21:00:00  | 23:30:00
 00005       | 0010        | Friday       | 18:00:00  | 20:00:00
 00006       | 0010        | Wednesday    | 21:00:00  | 23:30:00
 00007       | 0002        | Monday       | 17:30:00  | 19:45:00
 00008       | 0002        | Tuesday      | 19:00:00  | 20:45:00
 00009       | 0002        | Monday       | 20:00:00  | 22:00:00
```

```sql
CREATE TABLE Rehearses_In (
  ensemble_id   char(5) not null references Ensembles(ensemble_id),
  location_id   char(4) not null references Locations(location_id),
  dayOfTheWeek  text not null, --CheckRestraints
  startTime     time not null,
  endTime       time not null,
 primary key(ensemble_id)
);
```

Create Table Statements

Performs_In

Stores the department's music repertoire

Fundamental Dependencies

lit_id → songTitle, genre

```
CREATE TABLE Literature (
  lit_id    char(5) not null,
  songTitle text,
  genre     text,
 primary key(lit_id)
);
```

lit_id	songtitle	genre
00001	Sleigh Ride	Christmas
00002	The Marist College Fight Song	
00003	The Marist Song	
00004	The Great Locomotive Chase	
00005	Everlong	
00006	We Wish you a Merry Christmas	Christmas
00007	First Suite in E Flat	
00008	Second Suite in F	
00009	The Planets	
00010	All Star	

Create Table Statements

Songs_Playing

References which songs each ensemble is playing

```
CREATE TABLE Songs_Playing (
  ensemble_id   char(5) not null references Ensembles(ensemble_id),
  lit_id        char(5) not null references Literature(lit_id),
 primary key(ensemble_id, lit_id)
);
```

```
 ensemble_id | lit_id
-------------+--------
 00001       | 00001
 00001       | 00002
 00001       | 00007
 00002       | 00004
 00002       | 00003
 00002       | 00006
 00002       | 00010
 00011       | 00008
 00007       | 00002
 00007       | 00003
 00008       | 00006
 00009       | 00010
 00003       | 00009
```


v_Administrators

Returns a list of administrators containing their names, position, and contact information.

```
CREATE VIEW v_Administrators AS
    SELECT firstName, lastName, roleName, emailAddress
    FROM People, Administrators, Department_Roles
    WHERE people.person_id = administrators.person_id
    AND administrators.role_id = department_roles.role_id;
```

```
 firstname |   lastname   |                rolename                |          emailaddress
-----------+--------------+----------------------------------------+--------------------------------
 Art       | Himmelberger | Director of Music and Director of Bands | arthur.himmelberger@marist.edu
 Sarah     | Williams     | Director of Choral Activities           | sarah.williams@marist.edu
 Mikey     | Napolitano   | Operations Manager                      | michael.napolitano@marist.edu
```

v_BandMembers

Returns the names, instrument, and graduation semesters of each band student.

```
CREATE VIEW v_BandMembers AS
    SELECT firstName, lastname, expectedGradSemester, instrumentName
    FROM People p, Musicians m, Instrumentalists i, Can_Play, Instruments
    WHERE p.person_id = m.person_id
    AND m.person_id = i.person_id
    AND i.person_id = Can_Play.person_id
    AND Can_Play.instrument_id = Instruments.instrument_id
    Order by lastname ASC;
```

```
 firstname | lastname | expectedgradsemester | instrumentname
-----------+----------+----------------------+---------------
 Andrew    | Bauman   | Spring 2020          | Trombone
 Andrew    | Bauman   | Spring 2020          | Euphonium
 Marsha    | Mellow   | Fall 2019            | Flute
 Cadence   | Smith    | Spring 2018          | Clarinet
 I.M.      | Tired    | Fall 2020            | Marimba
 Adagio    | Turner   | Spring 2021          | Harpsichord
 Melody    | Wind     | Fall 2020            | Baritone Sax
```


v_Singers

Returns the names, instrument, and graduation semesters of each student in Singers.

```
CREATE VIEW v_Singers AS
    SELECT firstName, lastName, expectedGradSemester, vocalRange
    FROM People p, Musicians m, Vocalists v
    WHERE p.person_id = m.person_id
    AND m.person_id = v.person_id
    Order by lastname ASC;
```

firstname	lastname	expectedgradsemester	vocalrange
Maureen	Biologist	Spring 2018	Baritone
Rita	Book	Spring 2019	Tenor
Johnny	Cash	Spring 2019	Alto
Ann	Chovey	Fall 2021	Alto
Marsha	Mellow	Fall 2019	Baritone
Isabelle	Ringing	Fall 2018	Soprano

MembersIn(ensemble_id, ‘results’)

Returns a roster for a given ensemble including Students’ Names and Expected Graduation Semesters.

```
CREATE OR REPLACE FUNCTION MembersIn(char(5), REFCURSOR) RETURNS refcursor AS
$$
DECLARE
    inputNumber char(5)       := $1;
    resultSet   REFCURSOR     := $2;
BEGIN
    OPEN resultSet for
        SELECT firstName, lastName, expectedGradSemester
        FROM   People p, Musicians m, Member_Of, Ensembles
        WHERE  p.person_id = m.person_id
        AND    m.person_id = member_of.person_id
    AND     member_of.ensemble_id = ensembles.ensemble_id
    AND     ensembles.ensemble_id = inputNumber;
    RETURN resultSet;
END;
$$
LANGUAGE plpgsql;
```

```
SELECT MembersIn('00001', 'results');
FETCH ALL FROM results;
```

firstname text	**lastname** text	**expectedgradsemester** text
Andrew	Bauman	Spring 2020
Cadence	Smith	Spring 2018
Melody	Wind	Fall 2020

PerformsIn(location_id, ‘results’)

Returns a list of the ensembles performing in a specified location along with their respective date and times

```
CREATE OR REPLACE FUNCTION PerformsIn(char(4), REFCURSOR) RETURNS refcursor AS
$$
DECLARE
    inputNumber char(4)       := $1;
    resultSet   REFCURSOR     := $2;
BEGIN
    OPEN resultSet for
        SELECT eventDate, eventTime, ensembleName
        FROM Ensembles e, Performs_In, Locations l
        WHERE l.location_id = performs_In.location_id
        AND e.ensemble_id = performs_In.ensemble_id
        AND l.location_id = inputNumber;
    RETURN resultSet;
END;
$$
LANGUAGE plpgsql;
```

```
SELECT PerformsIn('0003', 'results');
FETCH ALL FROM results;
```

eventdate date	**eventtime** time without time zone	**ensemblename** text
2017-12-02	14:00:00	Symphonic Band
2017-12-03	14:00:00	Orchestra

Returns a list of where and when a given ensemble is practicing in.
In this case, Symphonic Band.

```
SELECT dayOfTheWeek, startTime, endTime, roomName
FROM Ensembles e, Rehearses_In r, Locations l
WHERE l.location_id = r.location_id
AND e.ensemble_id = r.ensemble_id
AND e.ensemble_id = '00001';
```

dayoftheweek text	**starttime** time without time zone	**endtime** time without time zone	**roomname** text
Monday	20:00:00	22:00:00	Symphonic Hall

Returns a list of the songs a given ensemble is playing.
In this case, Symphonic Band

```
SELECT songTitle
FROM Literature l, Songs_Playing s, Ensembles e
WHERE l.lit_id = s.lit_id
AND s.ensemble_id = e.ensemble_id
AND e.ensemble_id = '00001';
```

songtitle text
Sleigh Ride
The Marist College Fight Song
First Suite in E Flat

Reports/Interesting Queries

Operation: Save Alan

This trigger prevents anyone from removing anyone named Alan Labouseur from the database

```
CREATE OR REPLACE FUNCTION saveAlan() RETURNS TRIGGER AS $$
BEGIN
    IF OLD.firstName = 'Alan'AND OLD.lastName = 'Labouseur'
    THEN RAISE EXCEPTION 'CANNOT DELETE ALAN, THE GREATEST DATABASE PROFESSOR WHO EVER LIVED';
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER savingAlan BEFORE DELETE ON People
FOR EACH ROW EXECUTE PROCEDURE saveAlan();
```

```
DELETE FROM People
where lastName = 'Labouseur';
```

```
ERROR:  CANNOT DELETE ALAN, THE GREATEST DATABASE PROFESSOR WHO EVER LIVED
CONTEXT:  PL/pgSQL function savealan() line 4 at RAISE
********** Error **********
```


```
CREATE ROLE Administrator;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC TO Administrator;
```

The administrators should have the ability to modify all tables as well as to create new views and stored procedures to be used by other users.

```
CREATE ROLE OfficeStaff;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES
IN SCHEMA PUBLIC TO OfficeStaff;
```

The office staff should have the ability to query data as well as upgrade records, they should not have the ability to delete records

```
CREATE ROLE Eboard;
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM Eboard;
GRANT SELECT ON Musicians, Instrumentalists, Vocalists,
                Can_Play, Instruments, Member_Of, Ensembles,
                Rehearses_In, Performs_In, Locations, Literature,
                Songs_Playing
                TO Eboard;
```

Band and Singers eBoard members should have the ability to generate rosters, view the music library, and check to make sure rehearsals and performances are scheduled. They should not be capable of viewing administrators information

Security

The purpose of this database system is to manage many of the daily bookkeeping tasks for the Marist College Music Department.

If this database were fully implemented, person_id would most likely be replaced by a CWID number, which it is already equipped to handle.

Implementation Notes

As of this moment, there is no way to confirm that the instruments a student is playing in an ensemble is one of the instruments they have reported knowing how to play.

Known Problems

This is a small-scale representation of what the larger, fully developed database would look like.

If someone were to continue developing this project, I would imagine the database would have **more check constraints** to ensure the database remain normalized. Additionally **more views and stored procedures** should exist to allow for a simplified user experience. The developer can also **take into account semesters**, in that, some ensembles may not be offered every semester. **Additional data could be tracked for Literature**, such as instrument parts, how many times a piece has been performed, the last time a piece has been performed, etc. The database could also begin to handle **time clock and payroll information for office staff members**.

Future Enhancements