GVSU Marching Band Database

Ron Foreman, Rob Sanchez, Victor Sun, Justin Wickenheiser

CIS 673 - 01

Fall 2017

**Table of Contents**

Requirements Specification 3

Enhanced Entity-Relationship (EER) Diagram 5

Relational Schema (BCNF) 6

Integrity Constraints Table 7

## Requirements Specification

**Nature of the Application**

The following database design is intended to support an application for creating and managing marching shows in multiple seasons, to be performed by Grand Valley State University’s Laker Marching Band. The application will be used to maintain the detailed plan for each Show, including each student’s involvement in different shows throughout the seasons along with specific Songs, uniforms and instruments.

**Student**  
The system tracks Students that are in the school marching band. Students have a unique student id, in addition to name and academic major attributes. Every Student in the system is either a Marcher or a Drum Major.

**Season**  
The system tracks the different marching Seasons. Seasons have a unique term code as well as a description. Each Season is composed of Shows.

**Show**  
The system will track marching Shows. Shows have a title and a performance date. A Show belongs to a specific Season and a Season can be composed of many shows.

**Song**  
The system will also track Songs. Songs have a unique Song ID, a title, tempo, measure count, and one or more composers. For any given Show in a given Season, there are Songs that make up the Show. A Show can have many Songs and a Song can be in multiple Shows. A Show must have at least 1 Song in it, but not every Song has to be in a Show. The system must also track the order of the Songs in each Show. The system will only track Songs that have a minimum tempo of 96 beats per minute and a minimum of 50 measures.

**Marcher**  
Marchers participate in Shows. Since some Marchers might be busy or otherwise unavailable during a Show, not every Marcher participates in a given Show. Additionally, there can be Shows that are planned for future Seasons, so not every Show has Marchers participating in it. A Marcher can participate in several Shows, and a Show has one or more Marchers. The system needs to track what instrument a Marcher plays for a given Show. A Marcher will only play one instrument per Show. The instrument that a Marcher can play must be one of the following: Piccolo, Clarinet, Alto Sax, Tenor Sax, Mellophone, Trumpet, Trombone, Baritone, Sousaphone, Percussion, Flag, or Twirler. A Marcher must play the same instrument for every Show they participate in for a given Season.

**Lead Conductor**  
Drum Majors “become” Lead Conductors for specific Songs of any given show. “Lead Conducting” is an event that occurs during a show, and the duration of that event could be one or many Songs.  “Lead Conductor” is a role a Drum Major takes on for the duration of a Song or Songs during a Show.  A Song only has one Lead Conductor per Show. Not every Drum Major has to be a Lead Conductor within a Show. Every Song within a Show must have a Lead Conductor; however, since Shows can be planned for future Seasons, not every Show will have Lead Conductors assigned to Songs.

**Uniform**  
The system will track the Uniforms that are assigned to Students. Uniforms are uniquely identified by their uniform ID, and the system will track the date on which each Uniform was purchased. Each Student checks out only one Uniform, and a Uniform can only be checked out by one Student. Not every Uniform in the band's inventory is necessarily checked out.

## Relational Schema (Boyce-Codd Normal Form)

**Design Choice**

Marcher and DrumMajor are subtypes of entity Student. We chose design alternative #2 from the EER.ppt lecture notes and created one table per subtype, with no supertype table. At the beginning of the design process, we wanted to avoid the need for joins on queries requiring subtype attributes, but ultimately we didn’t create such queries anyway. Since our sub-typing was not overlapping but disjoint instead, it turned out to be a satisfactory decision.

**Marcher**(StudentID, FirstName, LastName, Major, UniformID)

**DrumMajor**(StudentID, FirstName, LastName, Major, UniformID)

**Uniform**(UniformID, PurchaseDate)

**Season**(TermCode, Description)

**Show**(TermCode, ShowTitle, Date)

**Song**(SongID, Title, Tempo, MeasureCount)

**Composer**(SongID, Composer)

**Participation**(MarcherID, TermCode, ShowTitle, Instrument)

**LeadConductor**(TermCode, ShowTitle, SongID, drumMajorID)

**ShowLineup**(TermCode, ShowTitle, SongID, Order)

## Integrity Constraints Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IC Name & Tables** | **IC Type** | **English Statement** | **Page**  **Found** | **Page Tested** |
| marcher\_IC1  *table: Marcher* | Key | Marchers have a unique student ID | Page A1  Line 36 | Page A16  Line 956 |
| composer\_IC2  *table: Composer* | Foreign Key | The Song ID must be an existing Song; if a Song gets deleted, delete the composer for that Song also. | Page A2  Line 128 | Page A16  Line 990 |
| participation\_IC3  *table: Participation* | 1-attribute | The instrument that a Marcher can play must be one of the following: Piccolo, Clarinet, Alto Sax, Tenor Sax, Mellophone, Trumpet, Trombone, Baritone, Sousaphone, Percussion, Flag, or Twirler. | Page A3  Line 156 | Page A16  Line 1005 |
| song\_IC2  *table: Song* | 2-attribute, 1-row | If the Song has 200 or more measures then the tempo must be greater than or equal to 120. | Page A2  Line 111 | Page A17  Line 1026 |
| participation\_IC5\_tr  *table: Participation* | 2-row (trigger) | A Marcher must play the same instrument for every Show they participate in for a given Season. | Page A4  Line 200 | Page A17  Line 1039 |

SQL> /\*

SQL> CIS 673 - Database Design Project

SQL> Ron Foreman

SQL> Rob Sanchez

SQL> Victor Sun

SQL> Justin Wickenheiser

SQL> \*/

SQL> -- Set the linesize (only to help keep things clean)

SQL> SET LINESIZE 300;

SQL> --

SQL> -- CREATE TABLES

SQL> --

SQL> -- Create Uniform Table

SQL> CREATE TABLE uniform (

2 uniformId NUMBER (15),

3 purchaseDate DATE,

4 --

5 -- uniform\_IC1:

6 -- uniformId is the primary key

7 CONSTRAINT uniform\_IC1 PRIMARY KEY (uniformId)

8 );

Table created.

SQL> --

SQL> -- Create Marcher Table

SQL> CREATE TABLE marcher (

2 studentId NUMBER(15),

3 firstName VARCHAR2(30),

4 lastName VARCHAR2(30),

5 major VARCHAR2(30),

6 uniformId NUMBER(15),

7 --

8 -- marcher\_IC1:

9 -- studentId is the primary key

10 CONSTRAINT marcher\_IC1 PRIMARY KEY (studentId),

11 -- marcher\_IC2:

12 -- Every marcher has one uniform checked out

13 CONSTRAINT marcher\_IC2 FOREIGN KEY (uniformId)

14 REFERENCES uniform (uniformId)

15 );

Table created.

SQL> --

SQL> -- Create Drum Major Table

SQL> CREATE TABLE drumMajor (

2 studentId NUMBER(15),

3 firstName VARCHAR2(30),

4 lastName VARCHAR2(30),

5 major VARCHAR2(30),

6 uniformId NUMBER(15),

7 --

8 -- drumMajor\_IC1:

9 -- studentId is the primary key

10 CONSTRAINT drumMajor\_IC1 PRIMARY KEY (studentId),

11 -- drumMajor\_IC2:

12 -- Every marcher has one uniform checked out

13 CONSTRAINT drumMajor\_IC2 FOREIGN KEY (uniformId)

14 REFERENCES uniform (uniformId)

15 );

Table created.

SQL> --

SQL> -- Create Season Table

SQL> CREATE TABLE season (

2 termCode NUMBER(6),

3 description VARCHAR2(30),

4 --

5 -- season\_IC1:

6 -- termCode is the primary key

7 CONSTRAINT season\_IC1 PRIMARY KEY (termCode)

8 );

Table created.

SQL> --

SQL> -- Create Show Table

SQL> CREATE TABLE show (

2 termCode NUMBER(6),

3 title VARCHAR2(30),

4 performDate DATE,

5 --

6 -- show\_IC1:

7 -- termCode and title are the composite primary key

8 CONSTRAINT show\_IC1 PRIMARY KEY (termCode,title),

9 -- show\_IC2:

10 -- termCode must be an existing season's termCode

11 -- If a season is deleted, then show also gets deleted.

12 CONSTRAINT show\_IC2 FOREIGN KEY (termCode)

13 REFERENCES season (termCode)

14 ON DELETE CASCADE

15 );

Table created.

SQL> --

SQL> -- Create Song Table

SQL> CREATE TABLE song (

2 songId NUMBER(15),

3 title VARCHAR2(30),

4 tempo NUMBER(3),

5 measureCount NUMBER(3),

6 --

7 -- song\_IC1:

8 -- songId is the primary key

9 CONSTRAINT song\_IC1 PRIMARY KEY (songId),

10 -- song\_IC2:

11 -- If the song has >=200 measures then the tempo must be >= 120

12 CONSTRAINT song\_IC2 CHECK (NOT (measureCount >= 200 AND tempo < 120))

13 );

Table created.

SQL> --

SQL> -- Create Composer Table

SQL> CREATE TABLE composer (

2 songId NUMBER(15),

3 composer VARCHAR2(30),

4 --

5 -- composer\_IC1:

6 -- songId and composer are the composite primary key

7 CONSTRAINT composer\_IC1 PRIMARY KEY (songId,composer),

8 -- composer\_IC2:

9 -- The songId must be an existing song.

10 -- If a song gets deleted, delete the composer for that song.

11 CONSTRAINT composer\_IC2 FOREIGN KEY (songId)

12 REFERENCES song (songId)

13 ON DELETE CASCADE

14 );

Table created.

SQL> --

SQL> -- Create Participation Table

SQL> CREATE TABLE participation (

2 marcherId NUMBER(15),

3 termCode NUMBER(6),

4 showTitle VARCHAR2(30),

5 instrument VARCHAR2(30),

6 --

7 -- participation\_IC1:

8 -- marcherId, termCode, and showTitle are the composite primary key

9 CONSTRAINT participation\_IC1 PRIMARY KEY (marcherId,termCode,showTitle),

10 -- participation\_IC2:

11 -- The marcherId must be an existing marcher.

12 -- If a marcher is deleted, then their participation in shows are deleted.

13 CONSTRAINT participation\_IC2 FOREIGN KEY (marcherId)

14 REFERENCES marcher (studentId)

15 ON DELETE CASCADE,

16 -- participation\_IC3:

17 -- The instrument that a marcher can play must be one of the following:

18 -- Piccolo, Clarinet, Alto Sax, Tenor Sax, Mellophone,

19 -- Trumpet, Trombone, Baritone, Sousaphone, Percussion, Flag, or Twirler.

20 CONSTRAINT participation\_IC3 CHECK (instrument IN ('piccolo','clarinet','alto sax','tenor sax','mellophone','trumpet','trombone','baritone','sousaphone','percussion','flag','twirler'))

21 -- participation\_IC4:

22 -- The combination of termCode and showTitle must be an existing show.

23 -- This will need to be done via a trigger.

24 --

25 -- participation\_IC5:

26 -- A marcher must play the same instrument for every show they participate in for a given season.

27 -- This will need to be done via a trigger.

28 );

Table created.

SQL> --

SQL> -- Create pariticiaption\_IC4 Trigger

SQL> -- The combination of termCode and showTitle must be an existing show.

SQL> CREATE TRIGGER participation\_IC4\_tr

2 BEFORE INSERT OR UPDATE ON

3 participation

4 FOR EACH ROW

5 DECLARE

6 counter INTEGER; /\* counter variable \*/

7 BEGIN

8 SELECT

9 COUNT(1)

10 INTO

11 counter

12 FROM

13 show

14 WHERE

15 termCode = :new.termCode

16 AND title = :new.showTitle;

17

18 IF counter = 0

19 THEN

20 RAISE\_APPLICATION\_ERROR(-20001,'The show/termCode combination does not exist. ' || :new.showTitle || ' does not exist in term ' || :new.termCode);

21 END IF;

22 END;

23 /

Trigger created.

SQL> --

SQL> -- Create pariticiaption\_IC5 Trigger

SQL> -- A marcher must play the same instrument for every show they participate in for a given season.

SQL> CREATE TRIGGER participation\_IC5\_tr

2 BEFORE INSERT OR UPDATE ON

3 participation

4 FOR EACH ROW

5 DECLARE

6 PRAGMA AUTONOMOUS\_TRANSACTION;

7 numInstruments INTEGER;

8 BEGIN

9 numInstruments := 0;

10 -- Get the instrument that :new.marcherId used when participating in shows for the given season :new.termCode

11 SELECT

12 COUNT(DISTINCT instrument) AS numInstruments

13 INTO

14 numInstruments

15 FROM

16 participation

17 WHERE

18 termCode = :new.termCode

19 AND marcherId = :new.marcherId

20 ;

21

22 IF numInstruments > 0 AND LOWER(:old.instrument) != LOWER(:new.instrument)

23 THEN

24 RAISE\_APPLICATION\_ERROR(-20001,'Invalid instrument. The marcher has been using ' || :old.instrument || ' all of the ' || :new.termCode || ' term. You are trying to switch the instrument to ' || :new.instrument || '.');

25 END IF;

26 END;

27 /

Trigger created.

SQL> --

SQL> -- Create Lead Conductor Table

SQL> CREATE TABLE leadConductor (

2 termCode NUMBER(6),

3 showTitle VARCHAR2(30),

4 songId NUMBER(15),

5 drumMajorId NUMBER(15),

6 --

7 -- leadConductor\_IC1:

8 -- termCode, showTitle, and songId are the composite primary key

9 CONSTRAINT leadConductor\_IC1 PRIMARY KEY (termCode,showTitle,songId),

10 -- leadConductor\_IC2:

11 -- The songId must be of an existing song.

12 -- If the song is deleted, then the leadConductor gets deleted.

13 CONSTRAINT leadConductor\_IC2 FOREIGN KEY (songId)

14 REFERENCES song (songId)

15 ON DELETE CASCADE,

16 -- leadConductor\_IC3:

17 -- The drumMajorId must be an existing drum major.

18 -- If the drum major gets deleted, then the leadConductor gets deleted.

19 CONSTRAINT leadConductor\_IC3 FOREIGN KEY (drumMajorId)

20 REFERENCES drumMajor (studentId)

21 ON DELETE CASCADE

22 -- leadConductor\_IC4:

23 -- The combination of termCode and showTitle must be an existing show.

24 -- This will need to be done via a trigger.

25 );

Table created.

SQL> --

SQL> -- Create leadConductor\_IC4 Trigger

SQL> -- The combination of termCode and showTitle must be an existing show.

SQL> CREATE TRIGGER leadConductor\_IC4\_tr

2 BEFORE INSERT OR UPDATE ON

3 leadConductor

4 FOR EACH ROW

5 DECLARE

6 counter INTEGER; /\* counter variable \*/

7 BEGIN

8 SELECT

9 COUNT(1)

10 INTO

11 counter

12 FROM

13 show

14 WHERE

15 termCode = :new.termCode

16 AND title = :new.showTitle;

17

18 IF counter = 0

19 THEN

20 RAISE\_APPLICATION\_ERROR(-20001,'The show/termCode combination does not exist. ' || :new.showTitle || ' does not exist in term ' || :new.termCode);

21 END IF;

22 END;

23 /

Trigger created.

SQL> --

SQL> -- Create Show Line Up Table

SQL> CREATE TABLE showLineup (

2 termCode NUMBER(6),

3 showTitle VARCHAR2(30),

4 songId NUMBER(15),

5 orderBy INTEGER,

6 --

7 -- showLineup\_IC1:

8 -- termCode, showTitle, and songId are the composite primary key

9 CONSTRAINT showLineup\_IC1 PRIMARY KEY (termCode,showTitle,songId),

10 -- showLineup\_IC2:

11 -- The songId must be of an existing song.

12 -- If the song is deleted, then it is removed from the line up.

13 CONSTRAINT showLineup\_IC2 FOREIGN KEY (songId)

14 REFERENCES song (songId)

15 ON DELETE CASCADE

16 -- showLineup\_IC3:

17 -- The combination of termCode and showTitle must be an existing show.

18 -- This will need to be done via a trigger.

19 );

Table created.

SQL> --

SQL> -- Create showLineup\_IC3 Trigger

SQL> -- The combination of termCode and showTitle must be an existing show.

SQL> CREATE TRIGGER showLineup\_IC3\_tr

2 BEFORE INSERT OR UPDATE ON

3 showLineup

4 FOR EACH ROW

5 DECLARE

6 counter INTEGER; /\* counter variable \*/

7 BEGIN

8 SELECT

9 COUNT(1)

10 INTO

11 counter

12 FROM

13 show

14 WHERE

15 termCode = :new.termCode

16 AND title = :new.showTitle;

17

18 IF counter = 0

19 THEN

20 RAISE\_APPLICATION\_ERROR(-20001,'The show/termCode combination does not exist. ' || :new.showTitle || ' does not exist in term ' || :new.termCode);

21 END IF;

22 END;

23 /

Trigger created.

SQL> --

SQL> -- Populate the tables with simple test data

SQL> SET FEEDBACK OFF

SQL> -- Insert uniforms

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (1,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (2,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (3,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (4,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (5,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (6,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (7,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (8,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (9,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (10,TO\_DATE('05/17/2013','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (11,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (12,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (13,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (14,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (15,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (16,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (17,TO\_DATE('10/03/2015','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (18,TO\_DATE('12/09/2016','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (19,TO\_DATE('12/09/2016','mm/dd/yyyy'));

SQL> INSERT INTO uniform (uniformId,purchaseDate) VALUES (20,TO\_DATE('12/09/2016','mm/dd/yyyy'));

SQL> -- Insert marchers

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (1000,'James','Singleton','Accounting',1);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (1011,'Emily','Reed','Music Education',2);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (1012,'Cody','Dalm','Music Education',3);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (2104,'Kalie','Twilling','Ad PR',4);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (2194,'Katie','Salinas','Accounting',5);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (2202,'John','Stickroe','Psychology',6);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (3963,'Abbigail','Fox','Nursing',7);

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (3004,'Mason','Riley','Music Education',8);

SQL> -- Insert drum majors

SQL> INSERT INTO drumMajor (studentId,firstName,lastName,major,uniformId) VALUES (2945,'Zach','Lehman','Music Education',13);

SQL> INSERT INTO drumMajor (studentId,firstName,lastName,major,uniformId) VALUES (1855,'Tim','Grieme','Music Education',14);

SQL> INSERT INTO drumMajor (studentId,firstName,lastName,major,uniformId) VALUES (2264,'Brianne','Krom','Nursing',15);

SQL> -- Insert seasons

SQL> INSERT INTO season (termCode,description) VALUES (201710,'Fall 2017');

SQL> INSERT INTO season (termCode,description) VALUES (201810,'Fall 2018');

SQL> INSERT INTO season (termCode,description) VALUES (201910,'Fall 2019');

SQL> -- Insert shows

SQL> INSERT INTO show (termCode,title,performDate) VALUES (201710,'Show 1',TO\_DATE('08/26/2017','mm/dd/yyyy'));

SQL> INSERT INTO show (termCode,title,performDate) VALUES (201710,'Show 2',TO\_DATE('09/2/2017','mm/dd/yyyy'));

SQL> INSERT INTO show (termCode,title,performDate) VALUES (201810,'Show 1',TO\_DATE('08/25/2018','mm/dd/yyyy'));

SQL> -- Insert songs

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (1,'Queen Opener',120,50);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (2,'All I Do is Win',100,45);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (3,'Applause',120,70);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (4,'Victorious',140,63);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (5,'Come Fly with Me',104,87);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (6,'Night Train',124,33);

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (7,'Daft Punk Medley',116,112);

SQL> -- Insert composer

SQL> INSERT INTO composer (songId,composer) VALUES (1,'Tom Wallace');

SQL> INSERT INTO composer (songId,composer) VALUES (1,'Tony McCutchen');

SQL> INSERT INTO composer (songId,composer) VALUES (2,'Tom Wallace');

SQL> INSERT INTO composer (songId,composer) VALUES (3,'Michael Brown');

SQL> INSERT INTO composer (songId,composer) VALUES (3,'Will Rapp');

SQL> INSERT INTO composer (songId,composer) VALUES (4,'Matt Conaway');

SQL> INSERT INTO composer (songId,composer) VALUES (4,'Jack Holt');

SQL> INSERT INTO composer (songId,composer) VALUES (5,'Paul Murtha');

SQL> INSERT INTO composer (songId,composer) VALUES (5,'Will Rapp');

SQL> INSERT INTO composer (songId,composer) VALUES (6,'Tom Wallace');

SQL> INSERT INTO composer (songId,composer) VALUES (7,'Tom Wallace');

SQL> INSERT INTO composer (songId,composer) VALUES (7,'Tony McCutchen');

SQL> -- Insert participation

SQL> -- -- -- 201710 Show 1 -- -- --

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1000,201710,'Show 1','clarinet');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1011,201710,'Show 1','piccolo');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1012,201710,'Show 1','alto sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2104,201710,'Show 1','tenor sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2194,201710,'Show 1','tenor sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2202,201710,'Show 1','mellophone');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (3963,201710,'Show 1','percussion');

SQL> -- -- -- 201710 Show 2 -- -- --

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1000,201710,'Show 2','clarinet');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1011,201710,'Show 2','piccolo');

SQL> -- marcherId 1012 did not participate in 201710 Show 2

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2104,201710,'Show 2','tenor sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2194,201710,'Show 2','tenor sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2202,201710,'Show 2','mellophone');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (3963,201710,'Show 2','percussion');

SQL> -- -- -- 201810 Show 1 -- -- --

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1000,201810,'Show 1','clarinet');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1011,201810,'Show 1','piccolo');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (1012,201810,'Show 1','alto sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2104,201810,'Show 1','sousaphone');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2194,201810,'Show 1','tenor sax');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (2202,201810,'Show 1','mellophone');

SQL> INSERT INTO participation (marcherId,termCode,showTitle,instrument) VALUES (3963,201810,'Show 1','percussion');

SQL> -- Insert showLineup

SQL> -- -- -- 201710 Show 1 -- -- --

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 1',1,1);

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 1',3,2);

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 1',2,3);

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 1',4,4);

SQL> -- -- -- 201710 Show 2 -- -- --

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 2',7,1);

SQL> INSERT INTO showLineUp (termCode,showTitle,songId,orderBy) VALUES (201710,'Show 2',5,2);

SQL> -- Insert leadConductor

SQL> -- -- -- 201710 Show 1 -- -- --

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 1',1,2945);

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 1',2,1855);

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 1',3,2264);

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 1',4,2945);

SQL> -- -- -- 201710 Show 2 -- -- --

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 2',7,1855);

SQL> INSERT INTO leadConductor (termCode,showTitle,songId,drumMajorId) VALUES (201710,'Show 2',5,2264);

SQL> --

SQL> SET FEEDBACK ON

SQL> COMMIT;

Commit complete.

SQL> --

SQL> -- Display the tables

SQL> SELECT \* FROM uniform;

UNIFORMID PURCHASED

---------- ---------

1 17-MAY-13

2 17-MAY-13

3 17-MAY-13

4 17-MAY-13

5 17-MAY-13

6 17-MAY-13

7 17-MAY-13

8 17-MAY-13

9 17-MAY-13

10 17-MAY-13

11 03-OCT-15

UNIFORMID PURCHASED

---------- ---------

12 03-OCT-15

13 03-OCT-15

14 03-OCT-15

15 03-OCT-15

16 03-OCT-15

17 03-OCT-15

18 09-DEC-16

19 09-DEC-16

20 09-DEC-16

20 rows selected.

SQL> SELECT \* FROM marcher;

STUDENTID FIRSTNAME LASTNAME MAJOR UNIFORMID

---------- ------------------------------ ------------------------------ ------------------------------ ----------

1000 James Singleton Accounting 1

1011 Emily Reed Music Education 2

1012 Cody Dalm Music Education 3

2104 Kalie Twilling Ad PR 4

2194 Katie Salinas Accounting 5

2202 John Stickroe Psychology 6

3963 Abbigail Fox Nursing 7

3004 Mason Riley Music Education 8

8 rows selected.

SQL> SELECT \* FROM drumMajor;

STUDENTID FIRSTNAME LASTNAME MAJOR UNIFORMID

---------- ------------------------------ ------------------------------ ------------------------------ ----------

2945 Zach Lehman Music Education 13

1855 Tim Grieme Music Education 14

2264 Brianne Krom Nursing 15

3 rows selected.

SQL> SELECT \* FROM song;

SONGID TITLE TEMPO MEASURECOUNT

---------- ------------------------------ ---------- ------------

1 Queen Opener 120 50

2 All I Do is Win 100 45

3 Applause 120 70

4 Victorious 140 63

5 Come Fly with Me 104 87

6 Night Train 124 33

7 Daft Punk Medley 116 112

7 rows selected.

SQL> SELECT \* FROM composer;

SONGID COMPOSER

---------- ------------------------------

1 Tom Wallace

1 Tony McCutchen

2 Tom Wallace

3 Michael Brown

3 Will Rapp

4 Jack Holt

4 Matt Conaway

5 Paul Murtha

5 Will Rapp

6 Tom Wallace

7 Tom Wallace

SONGID COMPOSER

---------- ------------------------------

7 Tony McCutchen

12 rows selected.

SQL> SELECT \* FROM season;

TERMCODE DESCRIPTION

---------- ------------------------------

201710 Fall 2017

201810 Fall 2018

201910 Fall 2019

3 rows selected.

SQL> SELECT \* FROM show;

TERMCODE TITLE PERFORMDA

---------- ------------------------------ ---------

201710 Show 1 26-AUG-17

201710 Show 2 02-SEP-17

201810 Show 1 25-AUG-18

3 rows selected.

SQL> SELECT \* FROM showLineup;

TERMCODE SHOWTITLE SONGID ORDERBY

---------- ------------------------------ ---------- ----------

201710 Show 1 1 1

201710 Show 1 3 2

201710 Show 1 2 3

201710 Show 1 4 4

201710 Show 2 7 1

201710 Show 2 5 2

6 rows selected.

SQL> SELECT \* FROM participation;

MARCHERID TERMCODE SHOWTITLE INSTRUMENT

---------- ---------- ------------------------------ ------------------------------

1000 201710 Show 1 clarinet

1011 201710 Show 1 piccolo

1012 201710 Show 1 alto sax

2104 201710 Show 1 tenor sax

2194 201710 Show 1 tenor sax

2202 201710 Show 1 mellophone

3963 201710 Show 1 percussion

1000 201710 Show 2 clarinet

1011 201710 Show 2 piccolo

2104 201710 Show 2 tenor sax

2194 201710 Show 2 tenor sax

MARCHERID TERMCODE SHOWTITLE INSTRUMENT

---------- ---------- ------------------------------ ------------------------------

2202 201710 Show 2 mellophone

3963 201710 Show 2 percussion

1000 201810 Show 1 clarinet

1011 201810 Show 1 piccolo

1012 201810 Show 1 alto sax

2104 201810 Show 1 sousaphone

2194 201810 Show 1 tenor sax

2202 201810 Show 1 mellophone

3963 201810 Show 1 percussion

20 rows selected.

SQL> SELECT \* FROM leadConductor;

TERMCODE SHOWTITLE SONGID DRUMMAJORID

---------- ------------------------------ ---------- -----------

201710 Show 1 1 2945

201710 Show 1 2 1855

201710 Show 1 3 2264

201710 Show 1 4 2945

201710 Show 2 7 1855

201710 Show 2 5 2264

6 rows selected.

SQL> --

SQL> -- Queries

SQL> -- 1. Join involving at least four relations.

SQL> -- Find the instrument, show title, and season description for each show that Kalie Twilling particiapted in. Order the results by the season and then by show title.

SQL> SELECT

2 m.firstName,

3 m.lastName,

4 s.description AS Season,

5 sh.title AS Show,

6 p.instrument

7 FROM

8 season s,

9 show sh,

10 participation p,

11 marcher m

12 WHERE

13 s.termCode = sh.termCode

14 AND sh.termCode = p.termCode

15 AND sh.title = p.showTitle

16 AND p.marcherId = m.studentId

17 AND LOWER(m.firstName) = 'kalie'

18 AND LOWER(m.lastName) = 'twilling'

19 ORDER BY

20 s.description,

21 sh.title

22 ;

FIRSTNAME LASTNAME SEASON SHOW INSTRUMENT

------------------------------ ------------------------------ ------------------------------ ------------------------------ ------------------------------

Kalie Twilling Fall 2017 Show 1 tenor sax

Kalie Twilling Fall 2017 Show 2 tenor sax

Kalie Twilling Fall 2018 Show 1 sousaphone

3 rows selected.

SQL> --

SQL> -- 2. Self join

SQL> -- Find pairs of marchers that share the same major.

SQL> SELECT

2 m1.firstName || ' ' || m1.lastName AS Marcher\_1,

3 m2.firstName || ' ' || m2.lastName AS Marcher\_2,

4 m1.major

5 FROM

6 marcher m1,

7 marcher m2

8 WHERE

9 m1.major = m2.major

10 AND m1.studentId < m2.studentId

11 ORDER BY

12 m1.major

13 ;

MARCHER\_1 MARCHER\_2 MAJOR

------------------------------------------------------------- ------------------------------------------------------------- ------------------------------

James Singleton Katie Salinas Accounting

Emily Reed Mason Riley Music Education

Cody Dalm Mason Riley Music Education

Emily Reed Cody Dalm Music Education

4 rows selected.

SQL> --

SQL> -- 3. Union

SQL> -- Select the firstName and lastName of marchers and drum majors that are majoring in Music Education. Order by the lastName.

SQL> SELECT

2 firstName,

3 lastName

4 FROM

5 marcher

6 WHERE

7 LOWER(major) = 'music education'

8 UNION

9 SELECT

10 firstName,

11 lastName

12 FROM

13 drumMajor

14 WHERE

15 LOWER(major) = 'music education'

16 ORDER BY

17 lastName

18 ;

FIRSTNAME LASTNAME

------------------------------ ------------------------------

Cody Dalm

Tim Grieme

Zach Lehman

Emily Reed

Mason Riley

5 rows selected.

SQL> --

SQL> -- 4. SUM, AVG, MAX, and MIN

SQL> -- Find the total number of measures, the average number of measures, max and min number of measures across all songs.

SQL> SELECT

2 SUM(measureCount),

3 AVG(measureCount),

4 MAX(measureCount),

5 MIN(measureCount)

6 FROM

7 song

8 ;

SUM(MEASURECOUNT) AVG(MEASURECOUNT) MAX(MEASURECOUNT) MIN(MEASURECOUNT)

----------------- ----------------- ----------------- -----------------

460 65.7142857 112 33

1 row selected.

SQL> --

SQL> -- 5. GROUP BY, HAVING, and ORDER BY

SQL> -- Find the marchers that particiapted in only 1 show for each season. For each marcher, get their name and the season's description.

SQL> SELECT

2 m.firstName,

3 m.lastName,

4 s.description,

5 COUNT(p.showTitle) AS showsMarched

6 FROM

7 marcher m,

8 season s,

9 participation p

10 WHERE

11 m.studentId = p.marcherId

12 AND p.termCode = s.termCode

13 GROUP BY

14 m.firstName,

15 m.lastName,

16 s.description

17 HAVING

18 COUNT(p.showTitle) = 1

19 ORDER BY

20 s.description,

21 m.lastName,

22 m.firstName

23 ;

FIRSTNAME LASTNAME DESCRIPTION SHOWSMARCHED

------------------------------ ------------------------------ ------------------------------ ------------

Cody Dalm Fall 2017 1

Cody Dalm Fall 2018 1

Abbigail Fox Fall 2018 1

Emily Reed Fall 2018 1

Katie Salinas Fall 2018 1

James Singleton Fall 2018 1

John Stickroe Fall 2018 1

Kalie Twilling Fall 2018 1

8 rows selected.

SQL> --

SQL> -- 6. Correlated Subquery

SQL> -- Find the name of the marcher(s) who have not particiapted in any shows.

SQL> SELECT

2 m.firstName,

3 m.lastName

4 FROM

5 marcher m

6 WHERE

7 NOT EXISTS (

8 SELECT

9 \*

10 FROM

11 participation

12 WHERE

13 marcherId = m.studentId

14 )

15 ;

FIRSTNAME LASTNAME

------------------------------ ------------------------------

Mason Riley

1 row selected.

SQL> --

SQL> -- 7. Non-Correlated Subquery

SQL> -- Find the song(s) that are not a part of any show line up.

SQL> SELECT

2 title

3 FROM

4 song

5 WHERE

6 songId NOT IN (

7 SELECT

8 songId

9 FROM

10 showLineUp

11 )

12 ;

TITLE

------------------------------

Night Train

1 row selected.

SQL> --

SQL> -- 8. Relational DIVISION

SQL> -- Find the studentId and name of every drum major who has conducted every song composed by Will Rapp

SQL> SELECT

2 dm.studentId,

3 dm.firstName,

4 dm.lastName

5 FROM

6 drumMajor dm

7 WHERE

8 NOT EXISTS (

9 (

10 SELECT

11 c.songId

12 FROM

13 composer c

14 WHERE

15 LOWER(c.composer) = 'will rapp'

16 ) MINUS (

17 SELECT

18 l.songId

19 FROM

20 leadConductor l, composer c

21 WHERE

22 l.drumMajorId = dm.studentId

23 AND l.songId = c.songId

24 AND LOWER(c.composer) = 'will rapp'

25 )

26 )

27 ;

STUDENTID FIRSTNAME LASTNAME

---------- ------------------------------ ------------------------------

2264 Brianne Krom

1 row selected.

SQL> --

SQL> -- 9. Outer Join

SQL> -- Find the uniformId and purchase date of every uniform. Also show the students name for those who have them.

SQL> SELECT

2 u.uniformId,

3 u.purchaseDate,

4 m.firstName || ' ' || m.lastName AS marcher,

5 d.firstName || ' ' || d.lastName AS drumMajor

6 FROM

7 uniform u

8 LEFT OUTER JOIN

9 marcher m ON u.uniformId = m.uniformId

10 LEFT OUTER JOIN

11 drumMajor d ON u.uniformId = d.uniformId

12 ORDER BY

13 u.uniformId

14 ;

UNIFORMID PURCHASED MARCHER DRUMMAJOR

---------- --------- ------------------------------------------------------------- -------------------------------------------------------------

1 17-MAY-13 James Singleton

2 17-MAY-13 Emily Reed

3 17-MAY-13 Cody Dalm

4 17-MAY-13 Kalie Twilling

5 17-MAY-13 Katie Salinas

6 17-MAY-13 John Stickroe

7 17-MAY-13 Abbigail Fox

8 17-MAY-13 Mason Riley

9 17-MAY-13

10 17-MAY-13

11 03-OCT-15

UNIFORMID PURCHASED MARCHER DRUMMAJOR

---------- --------- ------------------------------------------------------------- -------------------------------------------------------------

12 03-OCT-15

13 03-OCT-15 Zach Lehman

14 03-OCT-15 Tim Grieme

15 03-OCT-15 Brianne Krom

16 03-OCT-15

17 03-OCT-15

18 09-DEC-16

19 09-DEC-16

20 09-DEC-16

20 rows selected.

SQL> --

SQL> -- 10. RANK Query

SQL> -- Find the RANK and DENSE RANK of the uniform purchase date of '09-DEC-16' among all purchase dates

SQL> SELECT

2 RANK('09-DEC-16') WITHIN GROUP (ORDER BY purchaseDate) AS "Rank of 09-DEC-16",

3 DENSE\_RANK('09-DEC-16') WITHIN GROUP (ORDER BY purchaseDate) AS "Dense Rank of 09-DEC-16"

4 FROM

5 uniform

6 ;

Rank of 09-DEC-16 Dense Rank of 09-DEC-16

----------------- -----------------------

18 3

1 row selected.

SQL> --

SQL> -- 11. Top-N Query

SQL> -- Find the title and tempo of the four fastest songs.

SQL> SELECT

2 title,

3 tempo

4 FROM

5 (

6 SELECT

7 title,

8 tempo

9 FROM

10 song

11 ORDER BY

12 tempo DESC

13 )

14 WHERE

15 ROWNUM <= 4

16 ;

TITLE TEMPO

------------------------------ ----------

Victorious 140

Night Train 124

Queen Opener 120

Applause 120

4 rows selected.

SQL> --

SQL> -- TESTING ICs

SQL> --

SQL> -- Testing: marcher\_IC1 (key)

SQL> INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (3004,'Emily','Ketchum','Accounting',9);

INSERT INTO marcher (studentId,firstName,lastName,major,uniformId) VALUES (3004,'Emily','Ketchum','Accounting',9)

\*

ERROR at line 1:

ORA-00001: unique constraint (FOREMARO.MARCHER\_IC1) violated

SQL> COMMIT;

Commit complete.

SQL> --

SQL> -- Testing: drumMajor\_IC2 (foreign key)

SQL> UPDATE

2 drumMajor

3 SET

4 uniformId = 99

5 WHERE

6 studentId = 2945

7 ;

UPDATE

\*

ERROR at line 1:

ORA-02291: integrity constraint (FOREMARO.DRUMMAJOR\_IC2) violated - parent key not found

SQL> COMMIT;

Commit complete.

SQL>

SQL> --

SQL> -- Testing: composer\_ic2 (foreign key)

SQL> INSERT INTO composer VALUES (99,'Biggie Smalls');

INSERT INTO composer VALUES (99,'Biggie Smalls')

\*

ERROR at line 1:

ORA-02291: integrity constraint (FOREMARO.COMPOSER\_IC2) violated - parent key not found

SQL> COMMIT;

Commit complete.

SQL>

SQL>

SQL>

SQL> --

SQL> -- Testing: participation\_IC3 (1-attribute)

SQL> UPDATE

2 participation

3 SET

4 instrument = 'flute'

5 WHERE

6 marcherId = 3963

7 ;

participation

\*

ERROR at line 2:

ORA-20001: Invalid instrument. The marcher has been using percussion all of the 201710 term. You are trying to switch the instrument to flute.

ORA-06512: at "FOREMARO.PARTICIPATION\_IC5\_TR", line 20

ORA-04088: error during execution of trigger 'FOREMARO.PARTICIPATION\_IC5\_TR'

SQL> COMMIT;

Commit complete.

SQL> --

SQL> -- Testing: song\_IC2 (2-attribute, 1 row)

SQL> INSERT INTO song (songId,title,tempo,measureCount) VALUES (8,'Wabash Cannonball',116,245);

INSERT INTO song (songId,title,tempo,measureCount) VALUES (8,'Wabash Cannonball',116,245)

\*

ERROR at line 1:

ORA-02290: check constraint (FOREMARO.SONG\_IC2) violated

SQL> COMMIT;

Commit complete.

SQL> --

SQL> -- Testing: participation\_IC5\_tr (2-row)

SQL> UPDATE

2 participation

3 SET

4 instrument = 'piccolo'

5 WHERE

6 marcherId = 1000

7 AND termCode = 201710

8 AND showTitle = 'Show 2'

9 ;

participation

\*

ERROR at line 2:

ORA-20001: Invalid instrument. The marcher has been using clarinet all of the 201710 term. You are trying to switch the instrument to piccolo.

ORA-06512: at "FOREMARO.PARTICIPATION\_IC5\_TR", line 20

ORA-04088: error during execution of trigger 'FOREMARO.PARTICIPATION\_IC5\_TR'

SQL> COMMIT;

Commit complete.

SQL> --

SQL> SPOOL OFF