RUSH

Beneath, Between, and Behind the Unlimited Ceiling of Rush

A Database Design

Designed By Jonathan Pistilli

December 2, 2013

table of contents

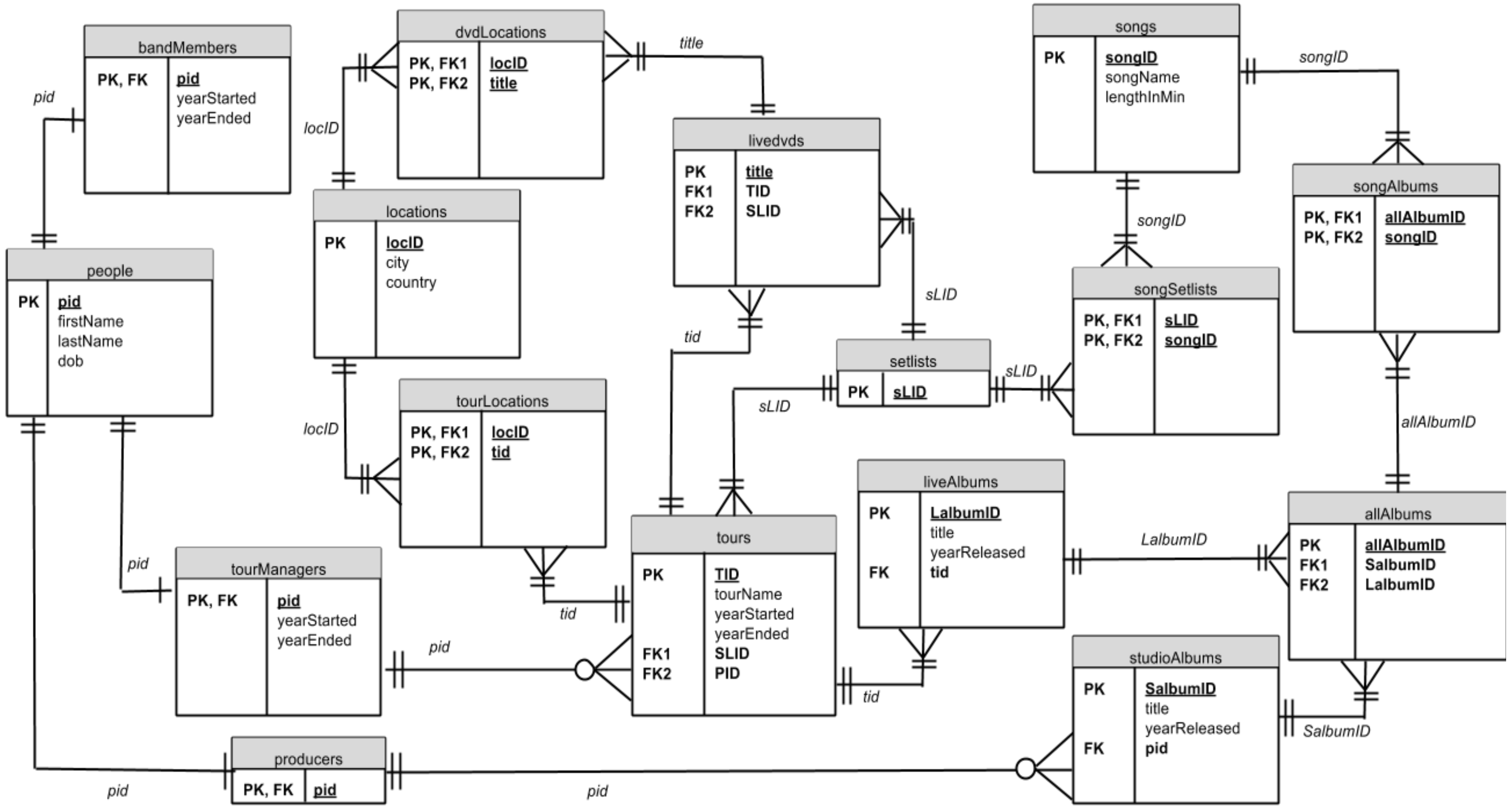
executive summary

With over one hundred and fifty songs, Rush has spanned the last forty years with their ever changing, ever the same style of music. Their number of tours, dvds, and albums are larger than many other bands from the past or present. Information on the Rush world is vast. A relational database is a perfect fit for the mass amount of data that Rush has and will produce over the years.

This design allows easier access and easier storing of information. This includes data such as song names, length of songs, albums, live albums, tours, tour managers, band members, and much more.

The structure of the database will be discussed below. This will show how tables are created, along with sample data, how reports are generated, triggers to contribute to data integrity, security measures, known problems and notes on the project design.

entity relationship diagram



create table statements

people table

Since band members, tour managers, and producers share the same qualities (i.e. first name, last name), their basic information is placed in this table.

```
CREATE TABLE people (
     pid       SERIAL         NOT NULL UNIQUE,
     firstName VARCHAR(30)    NOT NULL,
     lastName  VARCHAR(30)    NOT NULL,
     dob       DATE           NOT NULL,
PRIMARY KEY (pid)
);
```

functional dependencies

```
     pid → firstName, lastName, dob
```

sample data

Pid	firstName	lastName	dob
1	Jeffrey	Jones	1953-09-20
2	Alex	Živojinović	1953-08-27
3	John	Rutsey	1953-05-14
4	Gary	Weinrib	1953-07-29
5	Neil	Peart	1952-09-12
6	Terry	Brown	1947-08-07
7	Liam	Birt	1950-08-05

create table statements

bandMembers table

```
CREATE TABLE bandMembers (
     pid            INTEGER   NOT NULL,
     yearJoined     CHAR(4),
     yearLeft       CHAR(4),
PRIMARY KEY (pid),
FOREIGN KEY (pid) REFERENCES people(pid)
);
```

functional dependencies

```
     pid → yearJoined, yearLeft
```

sample data

pid	yearJoined	yearLeft
1	1968	1968
2	1968	
3	1968	1974
4	1968	
5	1974	

create table statements

producers table

```
CREATE TABLE producers (
     pid            INTEGER   NOT NULL,
PRIMARY KEY (pid),
FOREIGN KEY (pid) REFERENCES people(pid)
);
```

functional dependencies

```
     pid →
```

sample data

pid
6
9
10
11
12
13

create table statements

tourManagers table

```
CREATE TABLE producers (
     pid            INTEGER   NOT NULL,
     yearJoined     CHAR(4),
     yearLeft       CHAR(4),
PRIMARY KEY (pid),
FOREIGN KEY (pid) REFERENCES people(pid)
);
```

functional dependencies

```
     pid → yearJoined, yearLeft
```

sample data

pid	yearJoined	yearLeft
7	2002	2013
14	1985	1988
15	1989	1994
16	1995	1998
17	1968	1984

create table statements

songs table

A list of all songs by Rush.

```
CREATE TABLE songs (
     songID         SERIAL    NOT NULL,
     songName       CHAR(50)  NOT NULL,
     lengthInMin    time,
PRIMARY KEY (songID)
);
```

functional dependencies

```
     songID → songName, lengthInMin
```

sample data

songID	songName	lengthInMin
1	You Can't Fight It	2:50
10	Working Man	7:10
57	Limelight	4:19
124	Double Agent	4:51
110	Face Up	3:54
183	Headlong Flight	7:20
33	2112 – I. Presentation	3:42

create table statements

setlists table

This gives setlists and setlist ID to track specific setlists.

```
CREATE TABLE setlists (
     slID          SERIAL    NOT NULL,
PRIMARY KEY (slID)
);
```

functional dependencies

```
     slID →
```

sample data

slID
2
5
3
21
12
7

create table statements

songSetlists table

This is an associative table between songs and setlists that allows songs to match up to setlists.

```
CREATE TABLE songSetlists (
     slID      INTEGER   NOT NULL,
     songID    INTEGER   NOT NULL,
PRIMARY KEY (slID, songID),
FOREIGN KEY (slID)   REFERENCES setlists(slID),
FOREIGN KEY (songID) REFERENCES songs(songID)
);
```

functional dependencies

```
     {slID, songID} →
```

sample data

slID	songID
1	1
1	5
1	8
21	1
21	185
21	76
17	30

create table statements

tours table

A list of all tours by Rush.

```
CREATE TABLE tours (
     tid            SERIAL         NOT NULL,
     tourName       VARCHAR(50)    NOT NULL,
     yearStarted    CHAR(4),
     yearEnded      CHAR(4),
     slID           INTEGER,
     pid            INTEGER,
PRIMARY KEY (tid),
FOREIGN KEY (slID)  REFERENCES setlists(slID),
FOREIGN KEY (pid)   REFERENCES tourManagers(pid)
);
```

functional dependencies

```
     tid → tourName, yearStarted, yearEnded, slID, pid
```

sample data

tid	tourName	yearStarted	yearEnded	slID	pid
1	Rush	1974	1975	1	17
5	All The World's A Stage	1976	1977	4	17
8	Hemispheres	1978	1979	6	17
12	Signals	1982	1983	9	17
14	Power Windows	1985	1986	11	14
19	Test for Echo	1996	1997	16	16
25	Clockwork Angels	2012	2013	21	7

create table statements

locations table

A list of all locations that Rush has played at, as far as city and country.

```
CREATE TABLE locations (
     locID     SERIAL         NOT NULL,
     city      VARCHAR(50),
     country   VARCHAR(50),
PRIMARY KEY (locID)
);
```

functional dependencies

```
     locID → city, country
```

sample data

locID	city	country
1	Toronto	Canada
5	New York	United States
3	Quebec	Canada
25	Berlin	Germany
32	Milan	Italy
19	London	England
53	Rio de Janeiro	Brazil

create table statements

tourLocations table

This is an associative table between tours and locations that allows locations to be matched with tours.

```
CREATE TABLE tourLocations (
     tid       INTEGER   NOT NULL,
     locID     INTEGER   NOT NULL,
PRIMARY KEY (tid, locID),
FOREIGN KEY (tid)   REFERENCES tours(tid),
FOREIGN KEY (locID) REFERENCES locations(locID)
);
```

functional dependencies

```
     {tid, locID} →
```

sample data

tid	locID
1	1
5	7
25	1
25	25
12	25
12	44
15	21

create table statements

livedvds table

A list of all dvds that were recorded by Rush while on tour.

```
CREATE TABLE livedvds (
     title     VARCHAR(50)    NOT NULL,
     tid       INTEGER,
     slID      INTEGER,
PRIMARY KEY (title),
FOREIGN KEY (tid)   REFERENCES tours(tid),
FOREIGN KEY (slID)  REFERENCES setlists(slID)
);
```

functional dependencies

```
     title → tid, slID
```

sample data

title	tid	slID
Exit…Stage Left	13	5
Grace Under Pressure	8	8
A Show of Hands	15	12
Rush in Rio	21	17
R30	22	18
Snakes & Arrows Live	23	19
Time Machine	24	20

create table statements

dvdLocations table

This is an associative table between dvds and locations allowing dvds to be matched with the location they were recorded at.

```
CREATE TABLE dvdLocations (
     title     VARCHAR(50)    NOT NULL,
     locID     INTEGER,
PRIMARY KEY (title, locID)
FOREIGN KEY (locID) REFERENCES locations(locID),
FOREIGN KEY (title) REFERENCES dvds(title)
);
```

functional dependencies

```
     title → locID
```

sample data

title	locID
Exit…Stage Left	8
Grace Under Pressure	9
A Show of Hands	16
Rush in Rio	53
R30	25
Snakes & Arrows Live	81
Time Machine	36

create table statements

studioAlbums table

A list of all studio albums by Rush.

```
CREATE TABLE studioAlbums (
     SalbumID       SERIAL         NOT NULL,
     title          VARCHAR(50)    NOT NULL,
     yearReleased   CHAR(4),
     pid            INTEGER,
PRIMARY KEY (SalbumID),
FOREIGN KEY (pid) REFERENCES producers(pid)
);
```

functional dependencies

```
     SalbumID → title, yearReleased, pid
```

sample data

SalbumID	title	yearReleased	pid
1	Rush	1974	4
5	A Farewell to Kings	1977	8
7	Permanent Waves	1980	8
14	Roll the Bones	1991	10
20	Clockwork Angels	2012	11
10	Grace Under Pressure	1984	15
18	Feedback	2004	17

create table statements

liveAlbums table

A list of all live albums by Rush.

```
CREATE TABLE liveAlbums (
     LalbumID        SERIAL         NOT NULL,
     title           VARCHAR(50)    NOT NULL,
     yearReleased    CHAR(4),
     tid             INTEGER,
PRIMARY KEY (LalbumID),
FOREIGN KEY (tid) REFERENCES producers(tid)
);
```

functional dependencies

```
     LalbumID → title, yearReleased, tid
```

sample data

LalbumID	title	yearReleased	tid
2	Exit…Stage Left	1981	13
10	Moving Pictures: Live 2011	2011	24
1	All The World's A Stage	1976	5
12	Time Machine	2011	24
14	Clockwork Angels	2012	25
9	R30	2005	21
8	Rush in Rio	2003	20

create table statements

allAlbums table

A list of albums by Rush, including only studio and live albums.

```
CREATE TABLE allAlbums (
     allAlbumID     SERIAL    NOT NULL,
     SalbumID       INTEGER,
     LalbumID       INTEGER,
PRIMARY KEY (allAlbumID),
FOREIGN KEY (SalbumID) REFERENCES studioAlbums(SalbumID),
FOREIGN KEY (LalbumID) REFERENCES liveAlbums(LalbumID)
);
```

functional dependencies

```
     allAlbumID → SalbumID, LalbumID
```

sample data

allAlbumID	SalbumID	LalbumID
1	1	
8	8	
5	5	
19	19	
24		4
30		10
31		11

create table statements

songAlbums table

An associative table between songs and allAlbums which allows songs to be matched with their particular albums.

```
CREATE TABLE songAlbums (
     allAlbumID     INTEGER   NOT NULL,
     songID         INTEGER   NOT NULL,
PRIMARY KEY (allAlbumID, songID),
FOREIGN KEY (allAlbumID) REFERENCES allAlbums(allAlbumID),
FOREIGN KEY (songID)     REFERENCES songs(songID)
);
```

functional dependencies

```
     {allAlbumID, songID} →
```

sample data

allAlbumID	songID
2	22
2	25
2	19
6	44
6	43
29	173
29	30

view

Creates a view of all of the studio songs ever recorded by Rush.

```
CREATE OR REPLACE VIEW studioSongs (Song, Album, SongLength, YearReleased,
ProducerLastName )
as select distinct s.songName, studAlb.title, s.lengthInMin,
studAlb.yearReleased, p.lastname
from people p, producers prod, songs s, songAlbums sa, allAlbums aa,
studioAlbums studAlb
where s.songID = sa.songID
  AND sa.allAlbumID = aa.allAlbumID
  AND aa.SalbumID = studAlb.SalbumID
  AND p.pid = prod.pid
  AND prod.pid = studAlb.pid
order by yearReleased ASC;
```

stored procedure

This stored procedure returns the number of years that a member is active in the band Rush.

```
CREATE OR REPLACE FUNCTION yearsActive(year1 INTEGER, person1 CHAR, year2
INTEGER, person2 CHAR)
RETURNS INT AS $$
DECLARE
    yearBegan INT;
    yearOver INT;

BEGIN

SELECT   yearStarted
INTO yearBegan
FROM bandMembers
WHERE    pid = year1
  AND    yearEnded = person1;

SELECT   yearEnded
INTO yearOver
FROM bandMembers
WHERE    pid = year2
  AND    yearEnded = person2;

RETURN (yearOver - yearBegan);

END   $$ LANGUAGE plpgsql;
```

triggers

Insert trigger for songs, to add new songs from upcoming albums.

```
CREATE OR REPLACE FUNCTION songsInsert()
RETURNS trigger as $$

BEGIN

INSERT INTO songs ( songID, songName, lengthInMin )
             VALUES ( NEW.songID, NEW.songName, NEW.lengthInMin )

RETURN NEW;

END;  $$ LANGUAGE plpgsql;

CREATE TRIGGER songsInsert_trigger
AFTER INSERT ON songs
FOR EACH ROW EXECUTE PROCEDURE songsInsert();
```

reports

This report is useful for viewing the most common song details that most people would like to know in chronological order of album releases.

```
SELECT songName, title as album, yearReleased, lengthInMin
FROM   songs s, songAlbums sa, allalbums aa, studioAlbums studAlb
WHERE  s.songID = sa.songID
  AND  sa.allalbumID = aa.allalbumID
  AND  aa.SalbumID = studAlb.SalbumID

ORDER BY yearReleased ASC;
```

sample data

songName	Album	yearReleased	lengthInMin
Different Strings	Permanent Waves	1980	3:48
Natural Science	Permanent Waves	1980	9:17
Tom Sawyer	Moving Pictures	1981	4:33
Red Barchetta	Moving Pictures	1981	6:06
YYZ	Moving Pictures	1981	4:24
Limelight	Moving Pictures	1981	4:19
The Camera Eye	Moving Pictures	1981	10:56

reports

This report is shows all songs that were only performed live and never recorded on a studio album.

```
SELECT songName, lengthInMin
FROM   songs s, allAlbums a, songAlbums sa
WHERE  s.songID = sa.songID
  AND  sa.allAlbumID = a.allAlbumID
  AND  SalbumID is null

ORDER BY songName ASC;
```

sample data

songName	lengthInMin
Fancy Dancer	3:55
Garden Road	3:07
O’Malley’s Break	1:39

security

Since there is not any sensitive data in this database, there will only be two levels of users, database administrator(s) and users. Administrators will have the capability to edit information if need be. Users will be restricted from editing any information on the database, however they should be able to contact the administrator by email or another form of communication.

Administrator Grants

```
CREATE ROLE admin

GRANT SELECT, INSERT, UPDATE, DELETE

ON ALL TABLES IN SCHEMA PUBLIC

TO admin
```

User Grants

```
CREATE ROLE user

GRANT SELECT

ON ALL TABLES IN SCHEMA PUBLIC

TO user
```

final notes

- While the system itself seems like it could be very simple, it could have gotten very complex.
    - There could have been many more tables of information with interesting relationships.
- There is probably some way to make a person's years active as an interval instead of something like (yearStarted and yearEnded).
- The use of high security on a database like this is clearly not extremely essential.
- There might have been a simpler way to relate songs to albums
- Some dvds and live albums have multiple setlists, which would cause a problem with my relationships in the database
- If Rush tours the same place twice, there would have to be a date included to separate the data