



2016 US Women's Olympic Gymnastics Team

Reisha Puranik

Table of Contents

Executive Summary.....	3
ER Diagram.....	4
Tables.....	5
Views.....	11
Reports.....	14
Stored Procedures.....	18
Triggers.....	22
Security/Roles.....	23
Implementation Notes.....	26
Known Problems/Future Enhancements.....	27

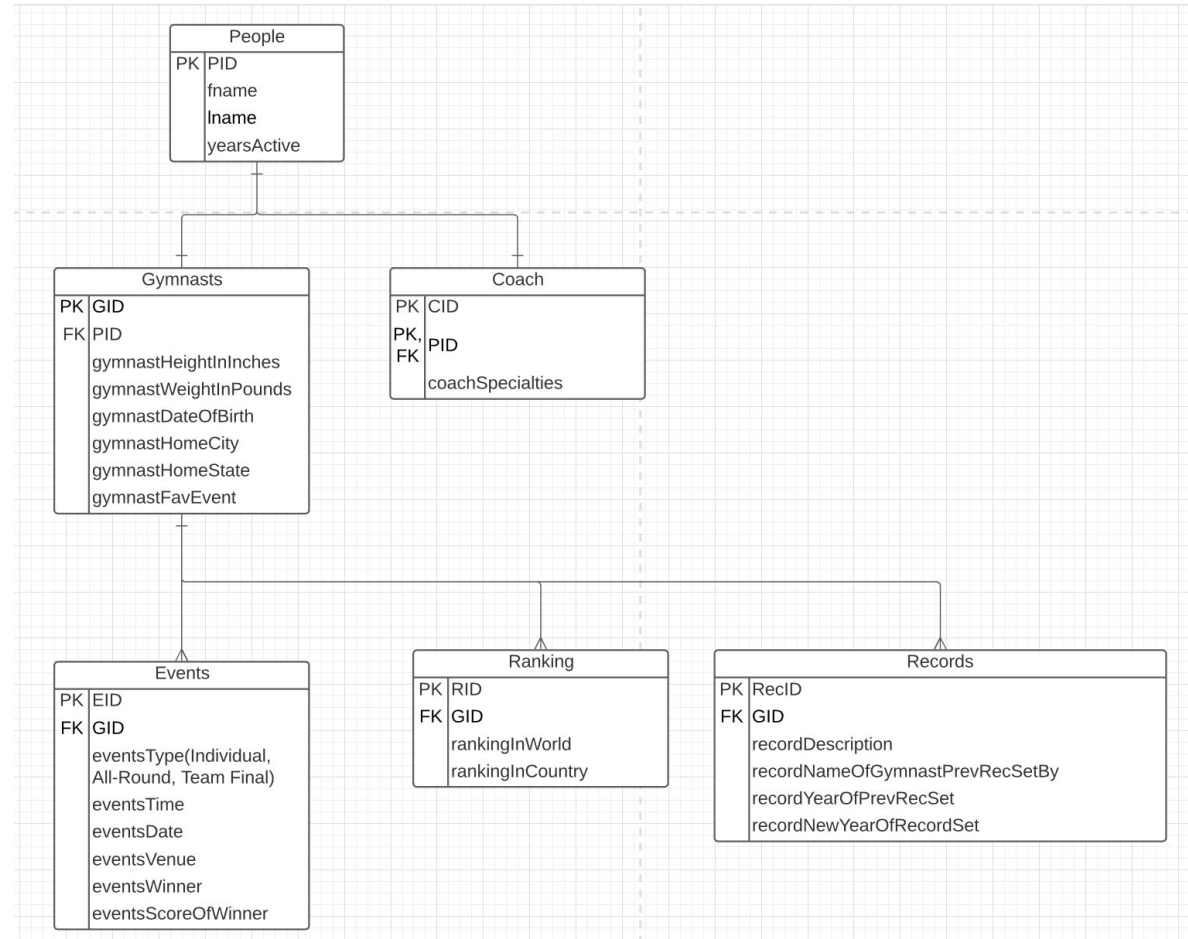
Executive Summary

The 2016 US Women's Olympic Gymnastics Team was very successful at the Rio Olympics. They broke records and they emerged victorious. The entire team contributed to the winnings of the events and in order to keep track of that, the US gymnastics team needed a database.

This paper outlines the database created in order to keep track of what the US gymnastics team won. This database was designed in pgAdmin and has various queries, views, reports, and stored procedures.

This database follows the five female US gymnasts who took part at the Rio Olympics and their scores as a team, and individually.

ER Diagram



People Table - The people table keeps track of all the people in the database with some basic information

Functional dependencies:

pid -> fname, lname, yearsActive

-- People --

```
CREATE TABLE People (  
    pid          int not null,  
    fname        text,  
    lname        text,  
    yearsActive  int,  
    primary key(pid)  
);
```

	pid [PK] integer	fname text	lname text	yearsactive integer
1	1	Simone	Biles	17
2	2	Laurie	Hernandez	15
3	3	Madison	Kocian	17
4	4	Gabby	Douglas	18
5	5	Aly	Raisman	20
6	6	Other	Country	0
7	7	Team	USA	0
8	8	Mihai	Brestyan	40
9	9	Laurent	Landi	20
10	10	Christian	Gallardo	15
11	11	Maggie	Haney	13
12	12	Cecile	Landi	20
13	13	Alan	Labouseur	18

Gymnasts Table - The gymnasts table compiles basic biometric and geographic information about each gymnast.

```
-- Gymnasts --  
CREATE TABLE Gymnasts (  
    gid                char(3) not null,  
    pid                int not null references People(pid),  
    HeightInInches     int,  
    WeightInPounds     int,  
    DateOfBirth        date,  
    homeCity           text,  
    homestate          text,  
    FavEvent           text,  
    primary key(gid),  
    CONSTRAINT CHK_FavEvent CHECK (FavEvent = 'Bars' OR  
                                    FavEvent = 'Beam' OR  
                                    FavEvent = 'Vault' OR  
                                    FavEvent = 'Floor')  
);
```

Functional Dependencies:

gid -> HeightInInches, WeightInPounds,
DateOfBirth, homeCity, homeState,
FavEvent

	gid [PK] character (3)	pid integer	heightinches integer	weightinpounds integer	dateofbirth date	homecity text	homestate text	favevent text
1	g01		56	104	1997-03-14	Columbus	Ohio	Floor
2	g02		61	106	2000-06-09	Old Bridge	New Jersey	Floor
3	g03		62	101	1997-06-15	Dallas	Texas	Bars
4	g04		62	108	1995-12-31	Newport News	Virginia	Beam
5	g05		62	115	1994-05-25	Needham	Massachusetts	Floor
6	g06		62	115	1994-05-25	Other	Country	Floor
7	g07		62	115	1994-05-25	Team	USA	Floor

Coach Table - The coach's table has information about each coach's specialties.

Functional Dependencies:

cid, pid(PK,FK) -> specialties

	cid [PK] character (3)	pid [PK] integer	specialties text
1	c01	8	Floor
2	c02	9	Bars
3	c03	10	Head Coach
4	c04	11	Head Coach
5	c05	12	Choreography
6	c06	13	Vault

-- Coach --

```
CREATE TABLE Coach (  
    cid char(3) not null,  
    pid int not null references People(pid),  
    specialties text,  
    primary key(cid, pid)  
);
```

Events Table - The events table contains specific information about each event that took place at the Rio Olympics.

Functional Dependencies:

eid -> eventType, eventTime,
eventDate, venue, winner, score

```
-- Events --
CREATE TABLE Events (
    eid          char(3) not null,
    gid          char(3) not null references Gymnasts(gid),
    eventType    text,
    eventTime    time,
    eventDate    date,
    venue        text,
    winner       text,
    score        decimal,
    primary key (eid),
    CONSTRAINT CHK_eventType CHECK (eventType = 'Individual: Bars' OR
    eventType = 'Individual: Beam' OR
    eventType = 'Individual: Vault' OR
    eventType = 'Individual: Floor' OR
    eventType = 'All-Round' OR eventType = 'Team Final')
);
```

	eid [PK] character (3)	gid character (3)	eventtype text	eventtime time without time zone	eventdate date	venue text	winner text	score numeric
1	e01	g06	Individual: Bars	14:00:00	2016-08-14	Arena Oli...	Aliya Must...	15.900
2	e02	g06	Individual: Bea...	12:00:00	2016-08-15	Arena Oli...	Sanne Wev...	15.466
3	e03	g01	Individual: Vault	08:00:00	2016-08-14	Arena Oli...	Simone Bil...	15.966
4	e04	g06	Individual: Floor	14:00:00	2016-08-16	Arena Oli...	Simone Bil...	15.966
5	e05	g06	All-Round	08:00:00	2016-08-11	Arena Oli...	Simone Bil...	62.198
6	e06	g07	Team Final	08:00:00	2016-08-09	Arena Oli...	USA	184.897

Ranking Table - The ranking table holds information about how each gymnast ranks in the world and the country.

Functional Dependencies:

rid -> rankingInWorld,
rankingInCountry

```
-- Ranking --  
CREATE TABLE Ranking (  
    rid                char(3) not null,  
    gid                char(3) not null references Gymnasts(gid),  
    rankingInWorld     int,  
    rankingInCountry   int,  
    primary key(rid)  
);
```

	rid [PK] character (3)	gid character (3)	rankinginworld integer	rankingincountry integer
1	r01	g01	1	1
2	r02	g02	3	3
3	r03	g03	5	5
4	r04	g04	4	4
5	r05	g05	2	2

Records Table - The records table holds information about gymnasts that have broken records and what records they have broken.

Functional Dependencies:

recID -> recordDescription,
nameOfGymnastPrevRecSetBy,
yearOfPrevRecSetBy, yearOfNewRecSet

```
-- Records --  
CREATE TABLE Records (  
    recID char(5) not null,  
    gid char(3) not null references Gymnasts(gid),  
    recordDescription text,  
    nameOfGymnastPrevRecSetBy text,  
    yearOfPrevRecSet date,  
    yearOfNewRecSet date,  
    primary key (recID)  
);
```

	recid [PK] character (5)	gid character (3)	recorddescription text	nameofgymnastprevrecsetby text	yearofprevrecset date	yearofnewrecset date
1	rec01	g01	Win of four gold medal...	Ecaterina Szabo	1984-08-20	2016-08-20
2	rec02	g01	Won the all around gold...	Lilia Podkopyeva	1996-08-20	2016-08-20

Views

Individual Event: This lists the winners and scores for the gymnasts of each individual event at the Olympics from highest to lowest score.




```
DROP VIEW IndividualEvent;  
CREATE VIEW IndividualEvent  
as  
select eid, gid, (eventType = 'Individual: Bars') AND  
        (eventType = 'Individual: Beam') AND  
        (eventType = 'Individual: Vault') AND  
        (eventType = 'Individual: Floor'), winner, score  
from Events  
where score < 60  
order by score DESC;  
  
select * from IndividualEvent;
```

	<div>eid</div> <div>character (3)</div>	<div>gid</div> <div>character (3)</div>	<div>?column?</div> <div>boolean</div>	<div>winner</div> <div>text</div>	<div>score</div> <div>numeric</div>
1	e03	g01	false	Simone Bil...	15.966
2	e04	g01	false	Simone Bil...	15.966
3	e01	g06	false	Aliya Must...	15.900
4	e02	g06	false	Sanne Wev...	15.466

Views

AllRoundEvent: This lists the winner of the all-round event and their score at the Olympics.





```
DROP VIEW AllRoundEvent;  
CREATE VIEW AllRoundEvent  
as  
select eid, gid, (eventType = 'All-Round') winner, score  
from Events  
where score > 60 AND score < 180;  
  
select * from AllRoundEvent;
```

	eid character (3) 	gid character (3) 	winner boolean 	score numeric 
1	e05	g01	true	62.198

Views

TeamFinalEvent: This lists the winner of the team final event and their team's score at the Olympics.

```
DROP VIEW TeamFinalEvent;  
CREATE VIEW TeamFinalEvent  
as  
select eid, gid, (eventType = 'Team Final') winner, score  
from Events  
where score > 180;  
  
select * from TeamFinalEvent;
```

	eid character (3) 	gid character (3) 	winner boolean 	score numeric 
1	e06	g07	true	184.897

Reports - Top 2 gymnasts in the world and which one has broken records

This query provides all the information about the top 2 gymnasts in the world. It also provides information on the records that they have set.

```
select *  
from Ranking r  
left outer join Gymnasts g on g.gid = r.gid  
left outer join Records rec on rec.gid = r.gid  
where rankingInWorld < 3;  
|
```






	<div>rid</div> <div>character (3)</div>	<div>gid</div> <div>character (3)</div>	<div>rankinginworld</div> <div>integer</div>	<div>rankingincountry</div> <div>integer</div>	<div>gid</div> <div>character (3)</div>	<div>pid</div> <div>integer</div>	<div>heightinches</div> <div>integer</div>	<div>weightinpounds</div> <div>integer</div>	<div>dateofbirth</div> <div>date</div>	<div>homeci</div> <div>text</div>
1	r01	g01	1	1	g01	1	56	104	1997-03-14	Columbu
2	r01	g01	1	1	g01	1	56	104	1997-03-14	Columbu
3	r05	g05	2	2	g05	5	62	115	1994-05-25	Needhan

<div>mecity</div> <div>t</div>	<div>homestate</div> <div>text</div>	<div>favevent</div> <div>text</div>	<div>recid</div> <div>character (5)</div>	<div>gid</div> <div>character (3)</div>	<div>recorddescription</div> <div>text</div>	<div>nameofgymnastprevrecsetby</div> <div>text</div>	<div>yearofprevrecset</div> <div>date</div>	<div>yearofnewrecset</div> <div>date</div>
Columbus	Ohio	Floor	rec01	g01	Win of four gold medal...	Ecaterina Szabo	1984-08-20	2016-08-20
Columbus	Ohio	Floor	rec02	g01	Won the all around gold...	Lilia Podkopayeva	1996-08-20	2016-08-20
Needham	Massachusetts	Floor	[null]	[null]	[null]	[null]	[null]	[null]

Reports - People who have 18+ years of experience

This query provides the first name, last name, years active, coach id, and specialties from the people data where they have at least 18 years of experience. This affected only two gymnasts and four coaches.

```
select fname, lname, yearsActive, cid, specialties
from People p
left join Gymnasts g on g.pid = p.pid
left join Coach c on c.pid = p.pid
where yearsActive >= 18;
```

	 fname text	 lname text	 yearsactive integer	 cid character (3)	 specialties text
1	Gabby	Douglas	18	[null]	[null]
2	Aly	Raisman	20	[null]	[null]
3	Cecile	Landi	20	c05	Choreography
4	Alan	Labouseur	18	c06	Vault
5	Mihai	Brestyan	40	c01	Floor
6	Laurent	Landi	20	c02	Bars

Reports - The coach whose specialty is floor

```
select *  
from People p inner join Coach c on p.pid = c.pid  
where specialties = 'Floor';
```

	pid integer	fname text	lname text	yearsactive integer	cid character (3)	pid integer	specialties text
1	8	Mihai	Brestyan	40	c01	8	Floor

This query provides all the data about the coach who specializes in the floor event using an inner join.

Reports - The gymnasts favorite events who rank below first in the world where they are grouped by height

```
select gid, FavEvent
from Gymnasts
where gid in (select gid from Ranking where rankingInWorld > 1)
group by HeightInInches, gymnasts.gid;
```

This query provides the gymnasts favorite events. The sub-query selects the gymnasts that are not ranked first and the query groups them by height.

	gid [PK] character (3)	favevent text
1	g05	Floor
2	g04	Beam
3	g02	Floor
4	g03	Bars

Stored Procedures

```
create or replace function CoachSpecialties(char(3), REFCURSOR) returns refcursor as
$$
declare
    cSpecialty          char(3)          := $1;
    resultset           REFCURSOR        := $2;
begin
    open resultset for
        select cid, specialties
        from    Coach
        where   cid = cSpecialty;
    return resultset;
end;
$$
language plpgsql;

select CoachSpecialties('c05', 'results');
Fetch all from results;
```

This stored procedure compiles the specialties for each coach after specifying the coach id.

	cid character (3) 	specialties  text
1	c05	Choreography

Stored Procedures

This stored procedure displays the gymnast's ranking in the world after specifying the gymnast id desired.

```
create or replace function RankingInWorld(char(3), REFCURSOR) returns refcursor as
$$
declare
    rRank          char(3)          := $1;
    resultset      REFCURSOR        := $2;
begin
    open resultset for
        select gid, rankingInWorld
        from   Ranking
        where  gid = rRank;
    return resultset;
end;
$$
language plpgsql;

select RankingInWorld('g04', 'results');
Fetch all from results;
```



	gid character (3) 	rankinginworld integer 
1	g04	4

Stored Procedures

```
create or replace function GymnastHomeState(char(3), REFCURSOR) returns refcursor as
$$
declare
    gHomeState          char(3)          := $1;
    resultset            REFCURSOR        := $2;
begin
    open resultset for
        select gid, homestate
        from    Gymnasts
        where   gid = gHomeState;
    return resultset;
end;
$$
language plpgsql;

select GymnastHomeState('g03', 'results');
Fetch all from results;
```

This stored procedure provides the gymnast's home state after specifying the gymnast id.

	gid [PK] character (3) 	homestate 
1	g03	Texas

Stored Procedures

```
create or replace function RankingIncrement() returns trigger as
$$
begin
    UPDATE Ranking SET rankingInWorld=rankingInWorld+1 where gid=NEW.gid;
end;
$$
language plpgsql;
```

This stored procedure returns a trigger and increments the gymnasts ranking in the world.

Triggers

```
create trigger RankIncreme BEFORE INSERT ON Ranking
FOR EACH ROW EXECUTE PROCEDURE RankingIncrement();
```

This trigger executes the RankingIncrement() function before data has been inserted into the table ranking.

Before

	rid [PK] character (3)	gid character (3)	rankinginworld integer	rankingincountry integer
1	r01	g01	1	1
2	r02	g02	2	3
3	r03	g03	3	5
4	r04	g04	4	4
5	r05	g05	5	2

After

	rid [PK] character (3)	gid character (3)	rankinginworld integer	rankingincountry integer
1	r01	g01	1	1
2	r02	g02	3	3
3	r03	g03	5	5
4	r04	g04	4	4
5	r05	g05	2	2

Security/Roles

Administrators: People in this role are granted full access to the data because they are in charge of each event and tracking the progress of each gymnast.

```
create role admin;  
GRANT ALL ON ALL TABLES  
IN SCHEMA PUBLIC  
TO admin;
```

Judges: Judges are able to select, insert, and update tables as needed.

```
create role judges;  
GRANT SELECT, INSERT, UPDATE ON ALL TABLES  
IN SCHEMA PUBLIC  
TO judges;
```

Security/Roles

Coach: This role is for the coaches to be able to have access to their gymnasts scores and other relevant data.

```
create role coach;  
GRANT SELECT ON ALL TABLES  
IN SCHEMA PUBLIC  
TO coach;
```

Gymnast: This role is for the gymnast to be able to have access to their scores and other relevant data.

```
create role gymnast;  
GRANT SELECT ON ALL TABLES  
IN SCHEMA PUBLIC  
TO gymnast;
```


Security/Roles

Commentators: This role is for the commentators who would like to be able to access the data and report it back to their audience.

```
create role commentators;  
GRANT SELECT ON ALL TABLES  
IN SCHEMA PUBLIC  
TO commentators;
```

Implementation Notes

The inserting data process was quite interesting, however, some of the data inserted was not accurate. Weight in pounds for a gymnast with short height should not be that low. A gymnast with the amount of training they go through should most likely be more. Although the information that I found to get this data was inaccurate, it was an observation I had made while inserting.

The tables, I believe, all had information that was interesting to the database created. I believe that the data queried was interesting to know and I learned a lot about doing different kinds of queries in the process.

Known Problems/Future Enhancements

When creating this database and implementing the data, there were few things that I took notice of.

One problem that I noticed was that when querying the data, it is usually the gymnast id that came along with the rest of the data. This might be hard to follow for others looking at this database because you would need to pay attention to what gymnast is associated with that id. Another problem was that since people id was the only int, it was hard to query data with other ids since the other ids were characters.

In the future I would like to add the scores of the other gymnasts, and not just the winner of that event. That would create more interesting queries.

As for queries and stored procedures, I would like to make more complex queries and stored procedures to make it even more interesting. I found it difficult to use the constraint type and multiple rows of data for just one type. In the future I would like some more calculations using these constraints.