RAW IS DATABASE:

An Employee Database for World Wrestling Entertainment

Designed by Ryan Fredericks

Database Management – Spring 2015

5/1/15

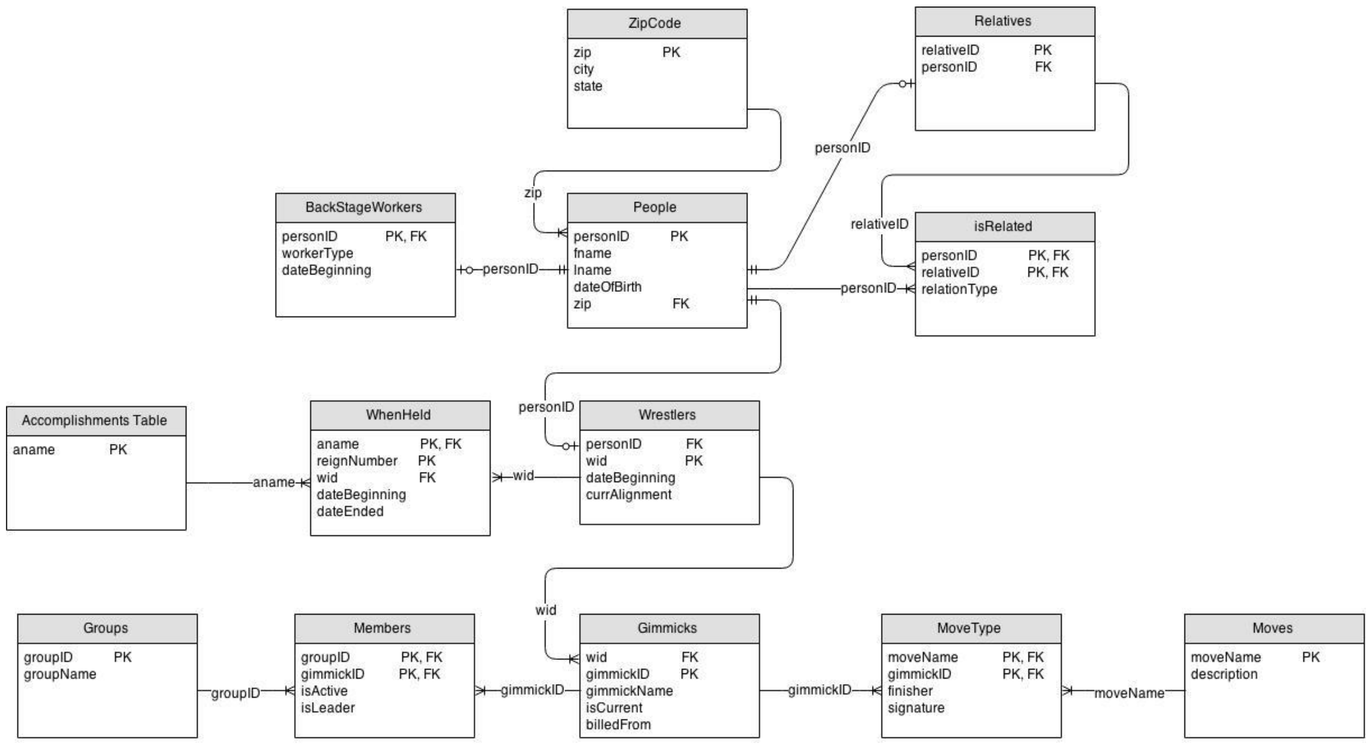
Table of Contents

EXECUTIVE SUMMARY

World Wrestling Entertainment, formally WWF and WWWF, is the largest wrestling promotion in the world, and has worked with thousands of employees and hundreds of wrestlers throughout its tenure that has reached across four decades. Wrestlers, writers, and executives have all worked for this global corporation, and some have worked in many different roles within this company. This document is meant to show the inner workings of the company, and how many careers have stretched decades. This is obviously an incomplete document, as there have been literally thousands of employees and hundreds of wrestlers in the company, some of which have more than one gimmick within the weekly television show. Since there are more than three hundred shows run per year by the million dollar industry, the need to organize all the information about employees in an easy to access way is extremely important to the smooth running of the company.

Shown below is the entity relationship diagram which shows how employees, both past and present, within the organization are organized, as well as the tables within the database and the functional dependencies that are defined within them. In addition to these, some views, reports, and stored procedures are defined within the SQL code in order to help find statistics about the wrestlers within the company. Security clearances are also created, as well as a report of what differences could be made to the database in order to improve it in the future, including comments on any issues that are known in this particular database.

Entity Relationship Diagram



TABLES

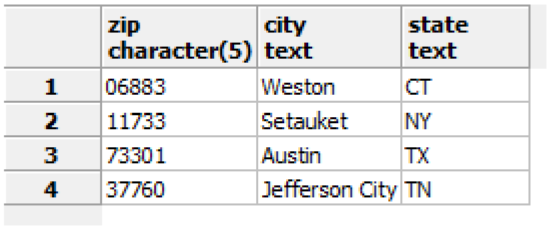
ZipCodes table

The ZipCodes table holds all the zip codes of current and former members of the company, regardless of which position they were in.

```
CREATE TABLE zipCodes (
  zip            char(5) not null,
  city           text not null,
  state          text not null,

  primary key(zip)
);
```

Functional Dependencies: zip -> city, state

Sample Data:



	zip character(5)	city text	state text
1	06883	Weston	CT
2	11733	Setauket	NY
3	73301	Austin	TX
4	37760	Jefferson City	TN

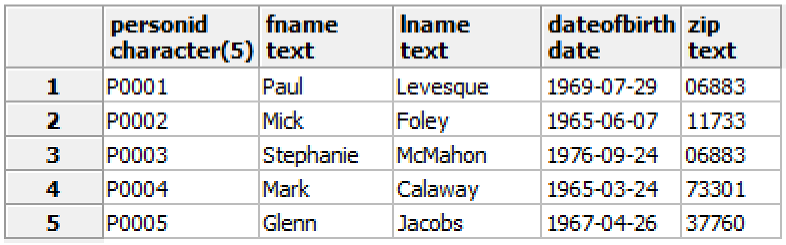
People table

The People table holds the personal information of everyone who is involved in the company. It has three entity subtypes, which are BackstageWorkers, Relatives, and Wrestlers.

```
CREATE TABLE people (
  personID        char(5) not null,
  fname           text not null,
  lname           text not null,
  dateOfBirth     date not null,
  zip             text not null references zipCodes(zip),

  primary key(personID)
);
```

Functional Dependencies: <u>personID</u> -> fname, lname, dateOfBirth, zip

Sample Data:



	personid character(5)	fname text	lname text	dateofbirth date	zip text
1	P0001	Paul	Levesque	1969-07-29	06883
2	P0002	Mick	Foley	1965-06-07	11733
3	P0003	Stephanie	McMahon	1976-09-24	06883
4	P0004	Mark	Calaway	1965-03-24	73301
5	P0005	Glenn	Jacobs	1967-04-26	37760

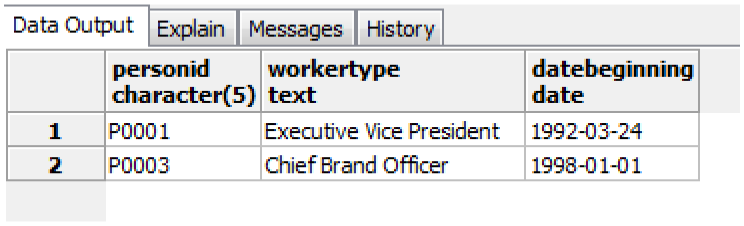
BackstageWorkers Table

The BackstageWorkers tables is an entity subtype which contains all people employed, either currently or formally, as behind-the-scenes crew (i.e. executives, trainers, writers).

```
CREATE TABLE backstageWorkers (
  personID        char(5) not null references people(personID),
  workerType      text not null,
  dateBeginning date not null,
  primary key(personID)
);
```

Functional Dependencies: <u>personID</u> ->workerType, dateBeginning

Sample Data:



Data Output | Explain | Messages | History

	personid character(5)	workertype text	datebeginning date
1	P0001	Executive Vice President	1992-03-24
2	P0003	Chief Brand Officer	1998-01-01

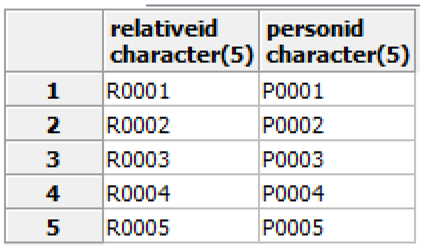
Relatives Table

The relatives table is an entity subtype that gives every current and former person involved with the company a relativeID.

```
CREATE TABLE relatives (
  relativeID        char(5) not null,
  personID          char(5) not null references people(personID),

  primary key(relativeID)
);
```

Functional Dependencies: relativeID -> personID

Sample Data:



	relativeid character(5)	personid character(5)
1	R0001	P0001
2	R0002	P0002
3	R0003	P0003
4	R0004	P0004
5	R0005	P0005

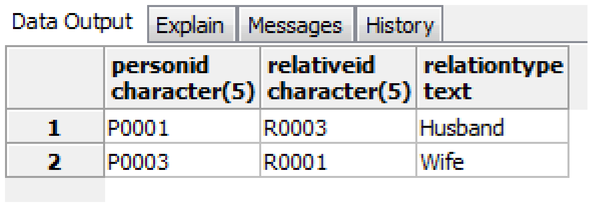
IsRelated Table

The IsRelated table combines a personID and a relativeID to show a relationship between two people who are involved in the company, as well as defining what that relationship is.

```
CREATE TABLE isRelated (
  personID        char(5) not null references people(personID),
  relativeID      char(5) not null references relatives(relativeID),
  relationType    text not null,

  primary key (personID, relativeID)
);
```

Functional Dependencies: <u>personID</u>, <u>relativeID</u> -> relationType

Sample Data:



	personid character(5)	relativeid character(5)	relationtype text
1	P0001	R0003	Husband
2	P0003	R0001	Wife

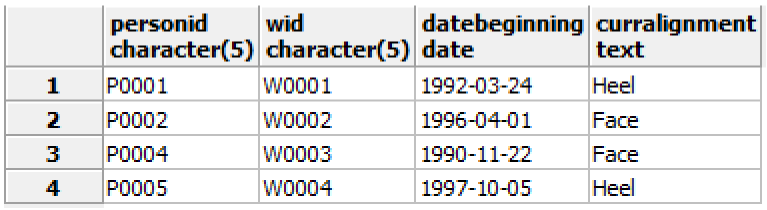
Wrestlers Table

The Wrestlers table defines people who have or are currently working as a Superstar or Diva within the company, as well as defining when they started and their current alignment within the product.

```
CREATE TABLE wrestlers (
  personID        char(5) not null references people(personID),
  wid             char(5) not null,
  dateBeginning date not null,
  currAlignment text not null,
  CHECK (currAlignment = 'Face' OR currAlignment = 'Heel'),
  primary key(wid)
);
```

Functional Dependencies : wid -> personID, dateBeginning, currAlignment

Sample Data:



	personid character(5)	wid character(5)	datebeginning date	curralignment text
1	P0001	W0001	1992-03-24	Heel
2	P0002	W0002	1996-04-01	Face
3	P0004	W0003	1990-11-22	Face
4	P0005	W0004	1997-10-05	Heel

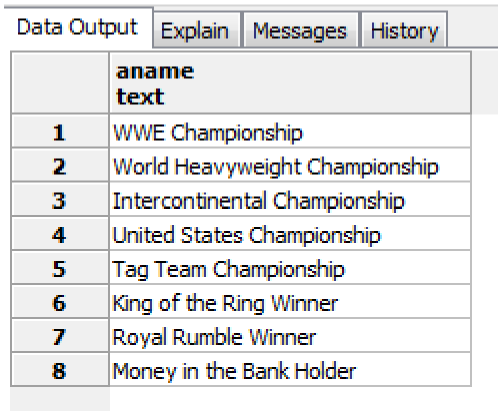
Accomplishments Table

The Accomplishments table defines titles or other such accomplishments that a wrestler can accomplish during their time in WWE. The names of titles will always be unique.

```
CREATE TABLE accomplishments (
  aname           text not null,

  primary key (aname)
);
```

Functional Dependencies: aname ->

Sample Data:



	aname text
1	WWE Championship
2	World Heavyweight Championship
3	Intercontinental Championship
4	United States Championship
5	Tag Team Championship
6	King of the Ring Winner
7	Royal Rumble Winner
8	Money in the Bank Holder

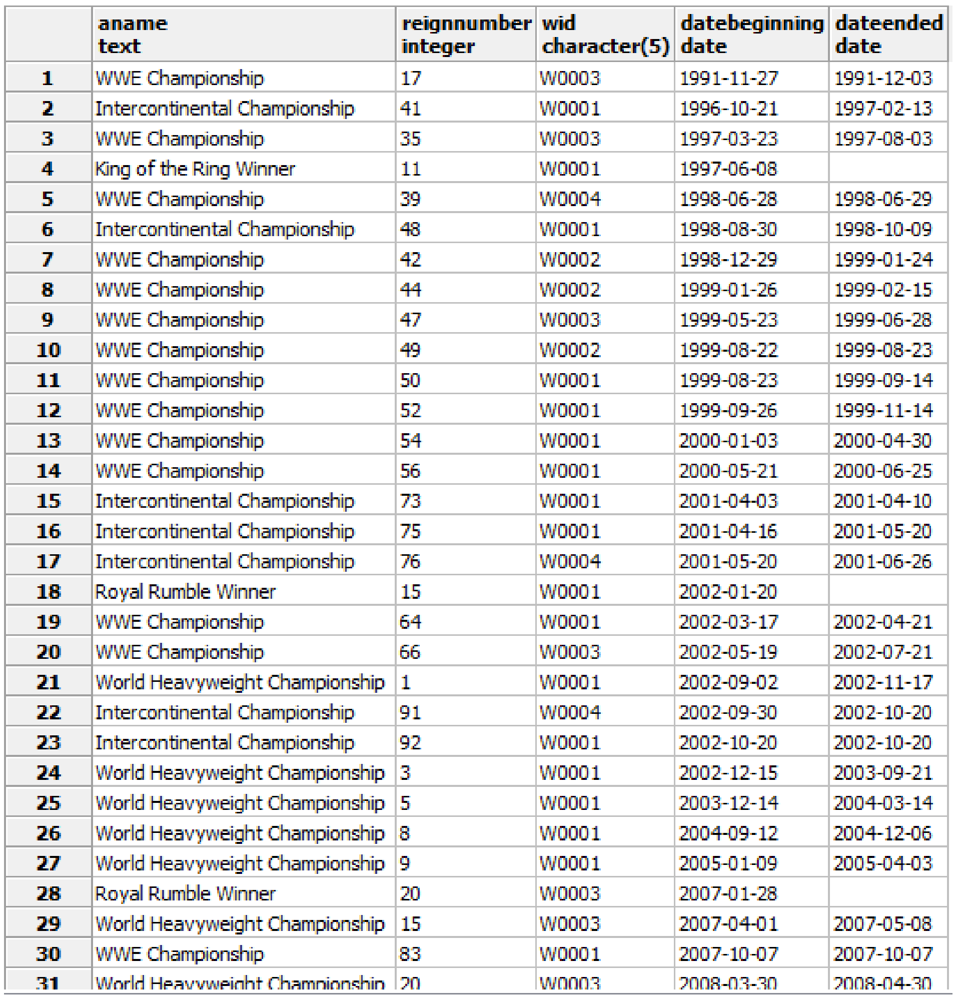
WhenHeld Table

The WhenHeld table defines when a wrestler within the company has either held a title or won an event. It is defined by the accomplishment itself and the number of the reign in regards to the history of the company, while stating when that reign began and ended.

```
CREATE TABLE whenHeld (
  aname          text not null references accomplishments(aname),
  reignNumber    integer not null,
  wid            char(5) not null references wrestlers(wid),
  dateBeginning  date not null,
  dateEnded      date,

  primary key (aname, reignNumber)
);
```

Functional Dependencies: aname, reignNumber -> wid, dateBeginning, dateEnded

Sample Data (ordered by dateBeginning ASC):



	aname text	reignnumber integer	wid character(5)	datebeginning date	dateended date
1	WWE Championship	17	W0003	1991-11-27	1991-12-03
2	Intercontinental Championship	41	W0001	1996-10-21	1997-02-13
3	WWE Championship	35	W0003	1997-03-23	1997-08-03
4	King of the Ring Winner	11	W0001	1997-06-08	
5	WWE Championship	39	W0004	1998-06-28	1998-06-29
6	Intercontinental Championship	48	W0001	1998-08-30	1998-10-09
7	WWE Championship	42	W0002	1998-12-29	1999-01-24
8	WWE Championship	44	W0002	1999-01-26	1999-02-15
9	WWE Championship	47	W0003	1999-05-23	1999-06-28
10	WWE Championship	49	W0002	1999-08-22	1999-08-23
11	WWE Championship	50	W0001	1999-08-23	1999-09-14
12	WWE Championship	52	W0001	1999-09-26	1999-11-14
13	WWE Championship	54	W0001	2000-01-03	2000-04-30
14	WWE Championship	56	W0001	2000-05-21	2000-06-25
15	Intercontinental Championship	73	W0001	2001-04-03	2001-04-10
16	Intercontinental Championship	75	W0001	2001-04-16	2001-05-20
17	Intercontinental Championship	76	W0004	2001-05-20	2001-06-26
18	Royal Rumble Winner	15	W0001	2002-01-20	
19	WWE Championship	64	W0001	2002-03-17	2002-04-21
20	WWE Championship	66	W0003	2002-05-19	2002-07-21
21	World Heavyweight Championship	1	W0001	2002-09-02	2002-11-17
22	Intercontinental Championship	91	W0004	2002-09-30	2002-10-20
23	Intercontinental Championship	92	W0001	2002-10-20	2002-10-20
24	World Heavyweight Championship	3	W0001	2002-12-15	2003-09-21
25	World Heavyweight Championship	5	W0001	2003-12-14	2004-03-14
26	World Heavyweight Championship	8	W0001	2004-09-12	2004-12-06
27	World Heavyweight Championship	9	W0001	2005-01-09	2005-04-03
28	Royal Rumble Winner	20	W0003	2007-01-28	
29	World Heavyweight Championship	15	W0003	2007-04-01	2007-05-08
30	WWE Championship	83	W0001	2007-10-07	2007-10-07
31	World Heavyweight Championship	20	W0003	2008-03-30	2008-04-30

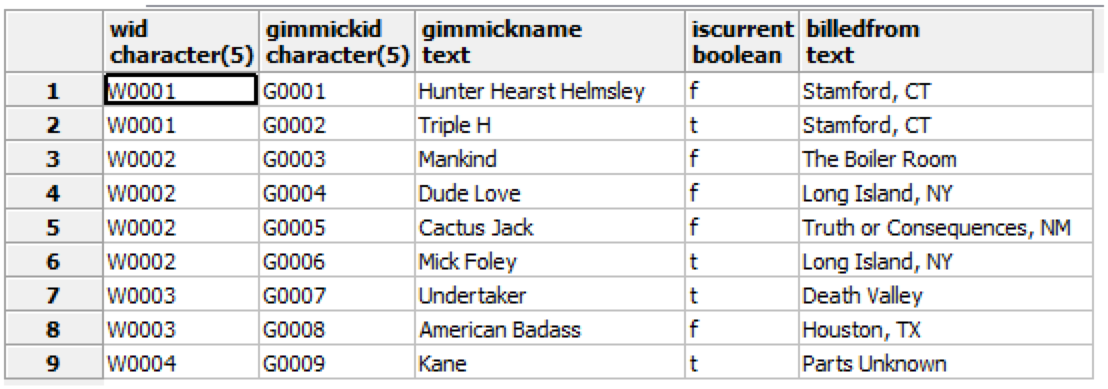
Gimmicks Table

The Gimmicks table shows the gimmicks that a wrestler has used throughout their career, while defining if it is their current gimmick, where the gimmick is billed from, and that gimmick's name.

```
CREATE TABLE gimmicks (
  wid            char(5) not null references wrestlers(wrestlerID),
  gimmickID      char(5) not null,
  gimmickName text not null UNIQUE,
  isCurrent      boolean not null,
  billedFrom     text not null,
  primary key (gimmickID)
);
```

Functional Dependencies: gimmickID -> wid, gimmickName, isCurrent, billedFrom

Sample Data:



	wid character(5)	gimmickid character(5)	gimmickname text	iscurrent boolean	billedfrom text
1	W0001	G0001	Hunter Hearst Helmsley	f	Stamford, CT
2	W0001	G0002	Triple H	t	Stamford, CT
3	W0002	G0003	Mankind	f	The Boiler Room
4	W0002	G0004	Dude Love	f	Long Island, NY
5	W0002	G0005	Cactus Jack	f	Truth or Consequences, NM
6	W0002	G0006	Mick Foley	t	Long Island, NY
7	W0003	G0007	Undertaker	t	Death Valley
8	W0003	G0008	American Badass	f	Houston, TX
9	W0004	G0009	Kane	t	Parts Unknown

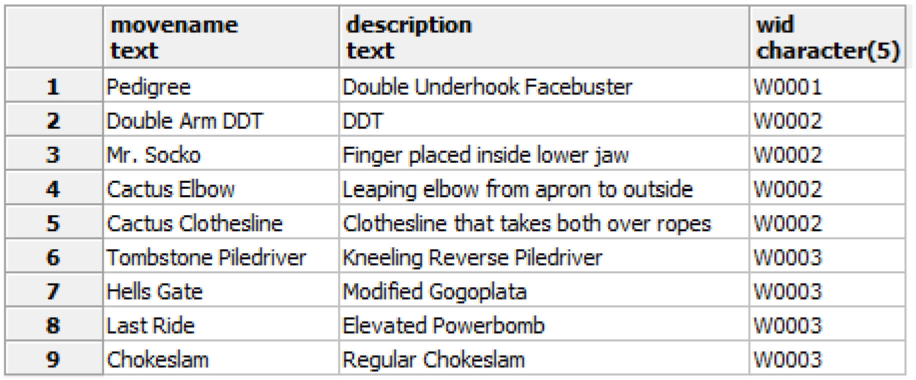
Moves Table

The moves table defines what major moves wrestlers have used throughout their career.

```
CREATE TABLE moves (
  moveName       text not null,
  description    text not null,
  primary key (moveName)
);
```

Functional Dependencies: <u>moveName</u> ->description

Sample Data:



	movename text	description text	wid character(5)
1	Pedigree	Double Underhook Facebuster	W0001
2	Double Arm DDT	DDT	W0002
3	Mr. Socko	Finger placed inside lower jaw	W0002
4	Cactus Elbow	Leaping elbow from apron to outside	W0002
5	Cactus Clothesline	Clothesline that takes both over ropes	W0002
6	Tombstone Piledriver	Kneeling Reverse Piledriver	W0003
7	Hells Gate	Modified Gogoplata	W0003
8	Last Ride	Elevated Powerbomb	W0003
9	Chokeslam	Regular Chokeslam	W0003

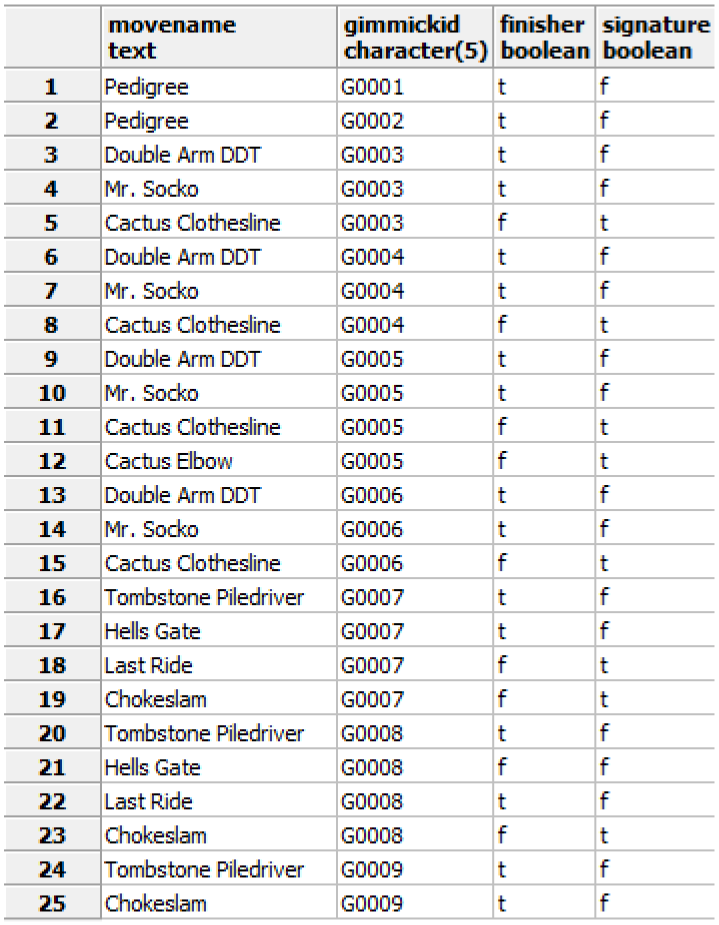
MoveType Table

The MoveType table defines which wrestlers in which gimmicks used certain moves, as well as defining whether that move was a signature or finisher for them.

```
CREATE TABLE moveType (
  moveName      text not null references moves(moveName),
  gimmickID     char(5) not null references gimmicks(gimmickID),
  finisher      boolean not null,
  signature     boolean not null,

  primary key (moveName, gimmickID)
);
```

Functional Dependencies: <u>moveName</u>, <u>gimmickID</u> -> finisher, signature

Sample Data:



	movename text	gimmickid character(5)	finisher boolean	signature boolean
1	Pedigree	G0001	t	f
2	Pedigree	G0002	t	f
3	Double Arm DDT	G0003	t	f
4	Mr. Socko	G0003	t	f
5	Cactus Clothesline	G0003	f	t
6	Double Arm DDT	G0004	t	f
7	Mr. Socko	G0004	t	f
8	Cactus Clothesline	G0004	f	t
9	Double Arm DDT	G0005	t	f
10	Mr. Socko	G0005	t	f
11	Cactus Clothesline	G0005	f	t
12	Cactus Elbow	G0005	f	t
13	Double Arm DDT	G0006	t	f
14	Mr. Socko	G0006	t	f
15	Cactus Clothesline	G0006	f	t
16	Tombstone Piledriver	G0007	t	f
17	Hells Gate	G0007	t	f
18	Last Ride	G0007	f	t
19	Chokeslam	G0007	f	t
20	Tombstone Piledriver	G0008	t	f
21	Hells Gate	G0008	f	f
22	Last Ride	G0008	t	f
23	Chokeslam	G0008	f	t
24	Tombstone Piledriver	G0009	t	f
25	Chokeslam	G0009	t	f

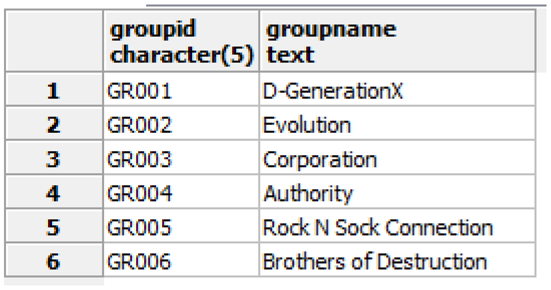
Groups Table

The Groups table defines groups or factions throughout the history of WWE.

```
CREATE TABLE groups (
  groupID          char(5) not null,
  groupName      text not null,

  primary key (groupID)
);
```

Functional Dependencies: groupID -> groupName

Sample Data:



	groupid character(5)	groupname text
1	GR001	D-GenerationX
2	GR002	Evolution
3	GR003	Corporation
4	GR004	Authority
5	GR005	Rock N Sock Connection
6	GR006	Brothers of Destruction

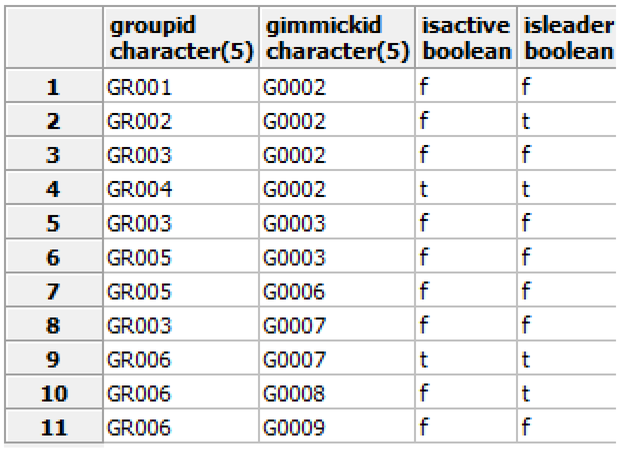
Members Table

The Members table defines the wrestlers in different gimmicks that were part of different groups, as well as if they are currently active in that group and if they are the leader.

```
CREATE TABLE members (
  groupID        char(5) not null references groups(groupID),
  gimmickID      char(5) not null references gimmicks(gimmickID),
  isActive       boolean not null,
  isLeader       boolean not null,

  primary key (groupID, gimmickID)
);
```

Functional Dependencies: groupID, gimmickID -> isActive, isLeader

Sample Data:



	groupid character(5)	gimmickid character(5)	isactive boolean	isleader boolean
1	GR001	G0002	f	f
2	GR002	G0002	f	t
3	GR003	G0002	f	f
4	GR004	G0002	t	t
5	GR003	G0003	f	f
6	GR005	G0003	f	f
7	GR005	G0006	f	f
8	GR003	G0007	f	f
9	GR006	G0007	t	t
10	GR006	G0008	f	t
11	GR006	G0009	f	f

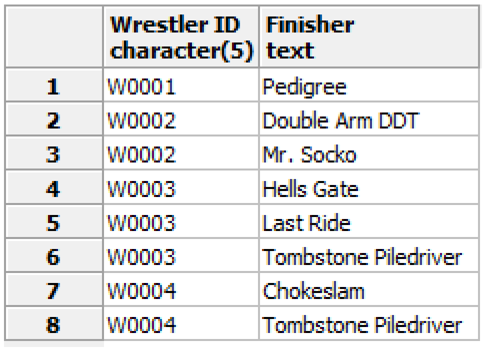
VIEWS

Finisher View

This view shows ANY finisher that a wrestler has used throughout their career, regardless of the gimmick.

```
CREATE VIEW Finishers

AS

SELECT DISTINCT w.wid AS "Wrestler ID", mt.moveName as "Finisher"

FROM wrestlers w, gimmicks gi, moveType mt

WHERE w.wid = gi.wid AND gi.gimmickID = mt.gimmickID  AND finisher = TRUE

GROUP BY w.wid, mt.moveName;
```

Sample Data:

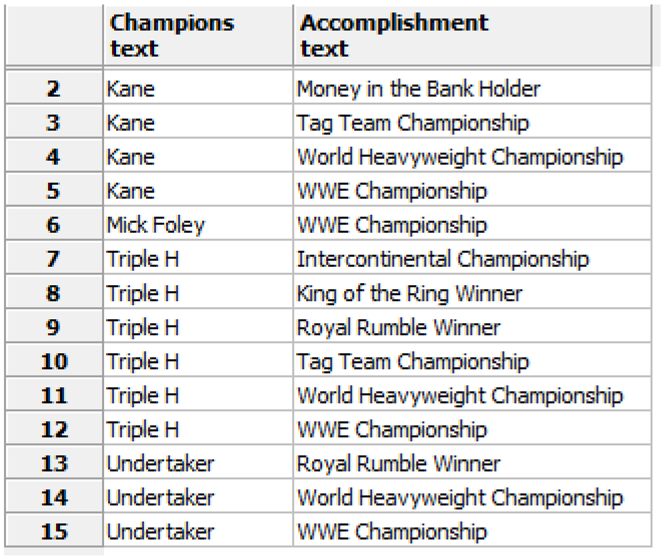


	Wrestler ID character(5)	Finisher text
1	W0001	Pedigree
2	W0002	Double Arm DDT
3	W0002	Mr. Socko
4	W0003	Hells Gate
5	W0003	Last Ride
6	W0003	Tombstone Piledriver
7	W0004	Chokeslam
8	W0004	Tombstone Piledriver

Champions View

This view shows each accomplishment that each wrestler in the database has gotten, based off their most current gimmick.

```
CREATE VIEW Champions
AS
SELECT DISTINCT gi.gimmickName AS "Champions", wh.aname as "Accomplishment"
FROM wrestlers w, gimmicks gi, whenHeld wh, accomplishments a
WHERE w.wid = gi.wid
        AND w.wid = wh.wid
        AND a.aname = wh.aname
        AND gi.isCurrent = TRUE
GROUP BY gi.gimmickName, wh.aname;
```



	Champions text	Accomplishment text
2	Kane	Money in the Bank Holder
3	Kane	Tag Team Championship
4	Kane	World Heavyweight Championship
5	Kane	WWE Championship
6	Mick Foley	WWE Championship
7	Triple H	Intercontinental Championship
8	Triple H	King of the Ring Winner
9	Triple H	Royal Rumble Winner
10	Triple H	Tag Team Championship
11	Triple H	World Heavyweight Championship
12	Triple H	WWE Championship
13	Undertaker	Royal Rumble Winner
14	Undertaker	World Heavyweight Championship
15	Undertaker	WWE Championship

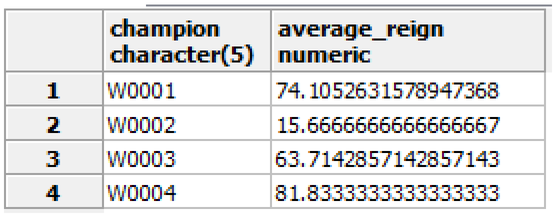
REPORTS

Average Championship Reigns

This report shows the average championship reign of all wrestlers in the system that have at least one championship reign, as long as there is an end date.   Money in the Bank does not count toward this as it is not a championship.

```
SELECT  wh.wid as Champion, avg(wh.dateEnded - wh.dateBeginning) AS Average_Reign
FROM whenHeld wh
WHERE wh.dateEnded IS NOT NULL AND wh.aname != 'Money in the Bank Holder'
GROUP BY wh.wid
ORDER BY wid ASC;
```

Sample Data:

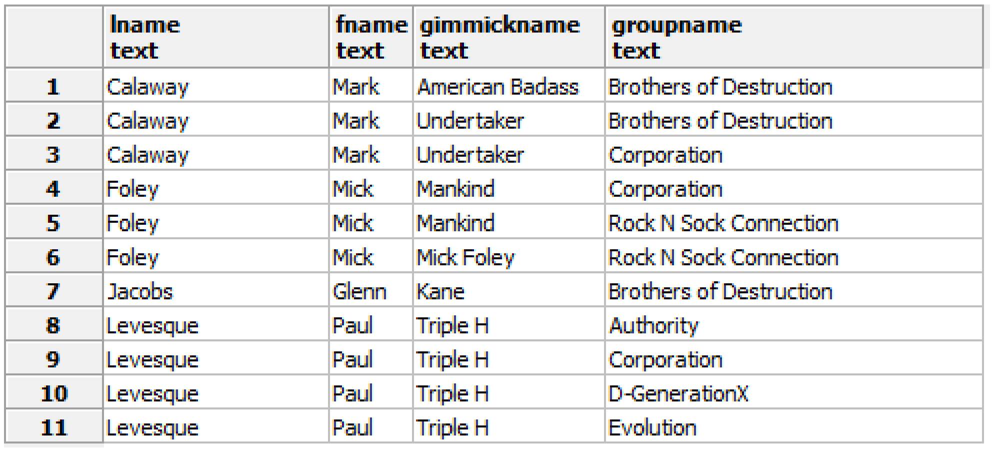


	champion character(5)	average_reign numeric
1	W0001	74.1052631578947368
2	W0002	15.6666666666666667
3	W0003	63.7142857142857143
4	W0004	81.8333333333333333

Real Life Gimmicks and Groups

This report shows the real life names of people within the company, as well as any groups that any of their gimmicks have been a part of.

```
SELECT  p.lname, p.fname, g.gimmickName, gr.groupName
FROM people p, wrestlers w, gimmicks g, members m, groups gr
WHERE p.personID = w.personID AND
            w.wid = g.wid AND
            g.gimmickID = m.gimmickID AND
            gr.groupID = m.groupID
GROUP BY p.lname, p.fname, g.gimmickName, gr.groupName
Order BY p.lname ASC;
```



	lname text	fname text	gimmickname text	groupname text
1	Calaway	Mark	American Badass	Brothers of Destruction
2	Calaway	Mark	Undertaker	Brothers of Destruction
3	Calaway	Mark	Undertaker	Corporation
4	Foley	Mick	Mankind	Corporation
5	Foley	Mick	Mankind	Rock N Sock Connection
6	Foley	Mick	Mick Foley	Rock N Sock Connection
7	Jacobs	Glenn	Kane	Brothers of Destruction
8	Levesque	Paul	Triple H	Authority
9	Levesque	Paul	Triple H	Corporation
10	Levesque	Paul	Triple H	D-GenerationX
11	Levesque	Paul	Triple H	Evolution

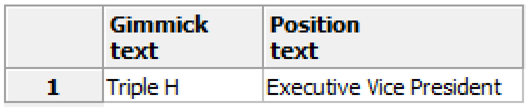
STORED PROCEDURES

Full Time Employee Stored Procedure

This stored procedure takes any employee that has both an active gimmick within the WWE and is a member of the backstage personnel (With the data in this database currently, only Triple H applies).

```
create or replace function FullTimeEmployee(REFCURSOR) returns refcursor as

$$

declare

  resultset       REFCURSOR := $1;

begin
  open resultset for
    select gi.gimmickName, bw.workerType
     from  gimmicks gi, backstageWorkers bw, people p, wrestlers w
     where       gi.isCurrent = TRUE
                 AND bw.personID IS NOT NULL
                 AND p.personID = bw.personID
                 AND p.personID = w.personID
                 AND w.wid = gi.wid;
  return resultset;
end;
$$
language plpgsql;
```

Sample Output:



	Gimmick text	Position text
1	Triple H	Executive Vice President

TRIGGERS

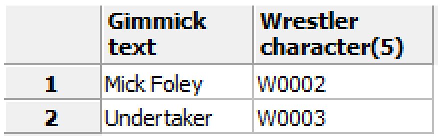
```
CREATE TRIGGER reviewFullTime
AFTER UPDATE ON people
FOR EACH ROW EXECUTE PROCEDURE FullTimeEmployees();
```

Faces Stored Procedure

This stored procedure looks into the database and finds which active wrestlers are faces, showing their gimmicks and their wrestling ID.

```
create or replace function Faces(REFCURSOR) returns refcursor as
$$
declare
  resultset        REFCURSOR := $1;
begin
  open resultset for
    select gi.gimmickName as "Gimmick", w.wid as "Wrestler"
     from   gimmicks gi, wrestlers w
     where        gi.isCurrent = TRUE  AND
                  w.wid = gi.wid  AND
                  w.currAlignment = 'Face';
  return resultset;
end;
$$
language plpgsql;
```

Sample Output:



	Gimmick text	Wrestler character(5)
1	Mick Foley	W0002
2	Undertaker	W0003

TRIGGERS

```
CREATE TRIGGER reviewFaces
AFTER UPDATE ON wrestlers or gimmicks
FOR EACH ROW EXECUTE PROCEDURE Faces();
```

SECURITY

There should be three types of users who would access the data involved in this database. These users are the administrators, the website staff, and the writing staff.

The administrator should have all privileges within the database, and therefore should have insert, update, and alter powers.

```
CREATE ROLE administrator
GRANT SELECT, INSERT, UPDATE, ALTER, DELETE
ON ALL TABLES IN SCHEMA PUBLIC
TO administrator
```

The website staff should be able to select the database, as they should not be making changes throughout to it

```
CREATE ROLE webstaff
GRANT SELECT
ON ALL TABLES IN SCHEMA PUBLIC
TO webstaff
```

The writing staff should be able to select, insert, and update the database, as they should be able to change the information within it in order to write for the show, but should not be able to alter the database as a whole.

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

Implementation Notes

In order to more easily recognize the people employed by the company, as well to be more comfortable in the future for updating, personIDs should be kept track of strictly in order to avoid duplicate keys within the database. The employees in the company should be filled in as they are introduced, and someone would need to keep on track of their successes and how they are currently functioning within the company. A writer's presence may be necessary in order to know for certain the changes in titles throughout the company. The company must also make sure that, even though there is a need to constantly update the database, not to infringe on the history documented in it. The implementation besides these small problems should be very smooth, however, as there is little to no redundancy and all of the tables within the database function well as a whole and separately.

Known Problems

Some of the problems in the database deal mostly with how the company will update it in the future, as it is incomplete in the grand scheme of the company itself. Whether the WWE would want this database to include all of the facets of the organization, or just of the employees themselves, would be major factors in determining how the database improves in the future. One of the major problems in the database is that due to the smaller sample size of people shown in this snapshot of the database, there could be some issues if a title reign in voided, and there are issues with title reigns themselves, as the reignNumber is generated after the fact, and is not in the proper order that should be shown within the database. This is due to the lack of wrestlers defined in the database, as there are very significant missing reigns. This is an issue that can easily be cleared up when more employees are added to the database.

Future Improvements

There are many significant improvements that could be made to the database to not only make it more complete, but could also make it significantly easier to use.

- The tables need to be filled with all employees or at the very least all wrestlers that have been involved with the company. While this may not be important in regards to the individual employees themselves, it is extremely essential in order to document the moves and accomplishments that have become staples of the history of the company.
- Splitting the people table into both former and current talent would be extremely helpful when it comes to finding information both in the history and current scope of the WWE.
- Replace wrestlers table with in-ring talents, as to highlight managers, announcers, and ring announcers/interviewers through entity subtypes.
- Find a way to document feuds and relationships between in-ring talent, in order to better grasp characters when looking at the database.
- Expand database to include other facets of the WWE besides employees, such as the ring gear, weapons, types of matches, and shows.
- Be able to update miscellaneous facts in order to better implement stored procedures.