2019

Database Design Project

CSU34041

JACK GILBRIDE – STUDENT NUMBER 17340868

Description

The database that I chose to design for this project was one for a professional wrestling company. The data seen in the database in this project models that of World Wrestling Entertainment Inc, or WWE. A lot of this information would be easily attainable by fans through applications like WWE.com or the WWE Network subscription model. While most of the database holds information that could be shared outside the company through applications, it would also contain private data, e.g. staff information. The relations that I decided to model are as follows:

STAFF

This would contain information about staff employed by WWE, such as their name, address, data of birth, salary and social security number (SSN). I have assumed that each staff member has a staff number which is unique; this is used as the primary key into the table.

WRESTLER

Many of the staff of WWE are professional wrestlers who require their own unique attributes. As wrestlers are inherently characters, many of these attributes are made up when the character is created such as their billed height, billed weight and the location they are billed from. Gender is also an attribute required for a wrestler to determine what matches they are eligible to fight in. The wrestler's ring name, which may or not be their real name, is the primary key to the table as WWE will not give two wrestlers the same ring name. Every wrestler is a staff member but not every staff member is a wrestler, thus there is a foreign key to the staff table (their staff number).

CHAMPIONSHIP

There are various championships that a wrestler may hold. These would require database entries to keep track of all of the championships in the company and who's holding them. Each championship has a unique name which would serve as the primary key in the tuple. The tuple's attributes are information that may be useful about the championship such as the date it was created, the plate colour and the strap colour of the belt. Each championship is held by a particular wrestler and belongs to a particular show, so there will be foreign keys to WRESTLER and SHOW respectively.

SHOW

WWE produces multiple weekly shows for television. Each show has a different name, airtime (the length of an episode), channel on which it is shown, and brand colour for promotional materials. These are all attributes of each tuple, where Name is the primary key as each show name is unique. Another attribute is the date on which the show first aired which may be considered valuable information to query.

EPISODE

Each show runs weekly, so has multiple episodes. The episode title is unique so it is the primary key. Other attributes of EPISODE would be the episode number, date and start time. The end time is not included as this can be calculated from start date and the airtime of the respective show. There are foreign keys to SHOW and ARENA, noting what show the episode is part of and what arena is hosting the show respectively.

ARENA

Assuming that each arena has a unique name, name the primary key to an ARENA tuple. The attendance of the arena is a useful attribute, as is the location, which is actually three attributes; street, city and state. An arena may host multiple episodes while each episode is only hosted by one arena.

MATCH

The final relation that I wanted to model was a wrestling match itself. I assumed a unique Match ID as the primary key of each tuple. The tuple would also record the start time, end time and outcome of the match (i.e. pinfall, submission finish, count-out or disqualification). During a match a championship can be defended, so a foreign key to the CHAMPIONSHIP table can go in the tuple. A match is also booked on a particular episode, so there is also a foreign key to the EPISODE table.

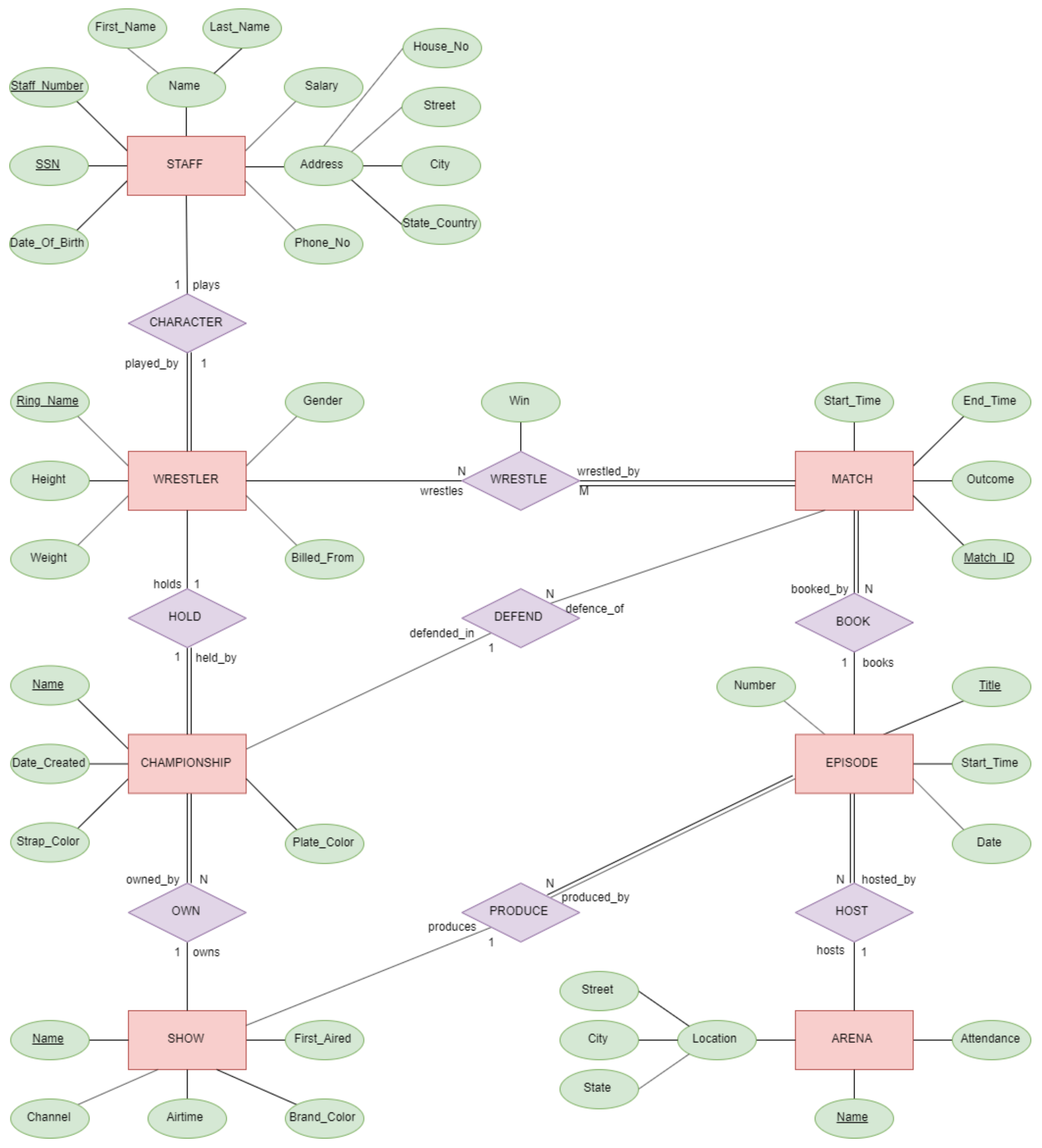
WRESTLER_IN_MATCH

A wrestler can wrestle M matches and a match can be wrestled by N wrestlers. As usual when modelling M..N relationships, the relationship is modelled as its own table. The win attribute is added to each tuple in this relation to identify whether a particular wrestler won a particular match. Multiple wrestlers may be recorded as winning the same match, e.g. a tag-team match.

Entity Relationship Diagram

I created an entity relationship diagram using the notation introduced in lectures. An entity type is represented by a rectangle with the relation name inside it, with its attributes surrounding it in ovals. Key attributes are represented by underlining the attribute. In this case the only key attributes were primary keys, apart from in STAFF where SSN could also be a unique key. There was no need to put uniqueness constraints on any other non-primary key attributes. Composite attributes are seen a number of times; as the Name and Address in STAFF, and the Location in ARENA.

Foreign keys which referred to other entity types were represented as relationships. The notation for a relationship between two entity types is a diamond between them. Most of the relationships have 1:N cardinality constraints, but there are also 1:1 constraints (STAFF:WRESTLER and WRESTLER:CHAMPIONSHIP) and an N:M constraint (MATCH:WRESTLER). These constraints are in line with the real-world semantics of the data. There are also participation constraints represented by a double line, e.g. a WRESTLER has to be played by a member of STAFF, and a MATCH has to be wrestled by N WRESTLERs. An example of a relationship with an attribute can also be seen in the WRESTLE relationship, where Win is an attribute.



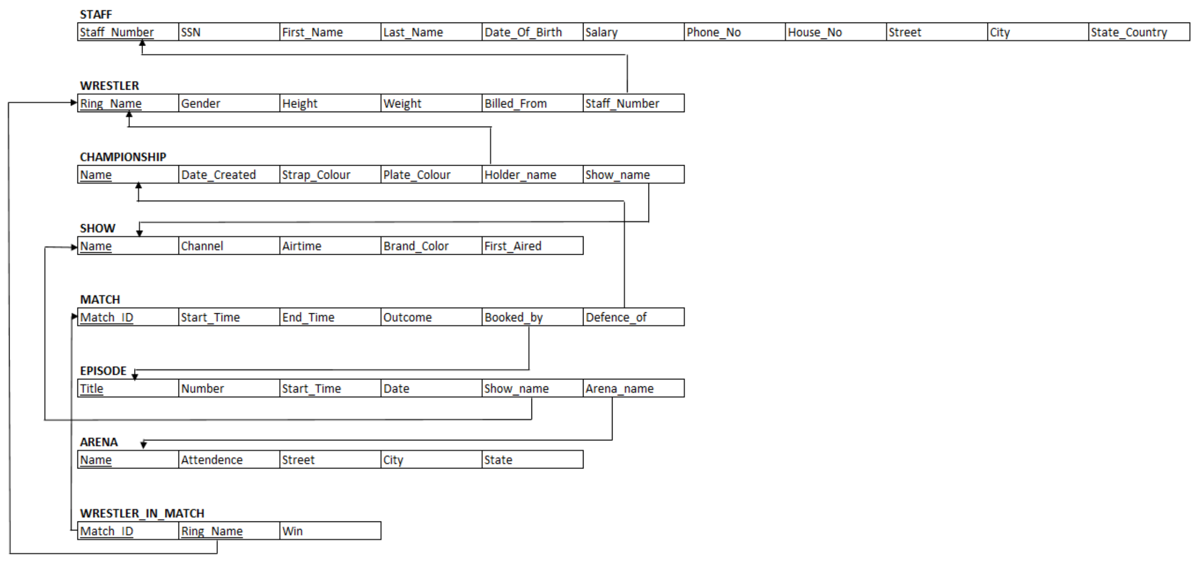
Relational Schema

The relational schema was translated from the entity relationship model as shown in lectures. Each entity type became a table, and each attribute became an attribute of that table. For each entity type, one of the key attributes was chosen as the primary key. In most cases there was only one key attribute to choose from, except in STAFF where there was both Staff_Number and SSN. Staff_Number seemed a more suitable primary key as it would be used for identification more often within the company, rather than SSN which would be used more in relation to the person's government identity.

There was only one 1:1 relationship; STAFF:WRESTLER. A merged relation approach could not be used for this as there was not total participation on both sides; any staff member who was not a wrestler would have many NULL attributes. As a result the foreign key approach was used. There was total participation on the WRESTLER side so the foreign key was put in WRESTLER pointing to STAFF.

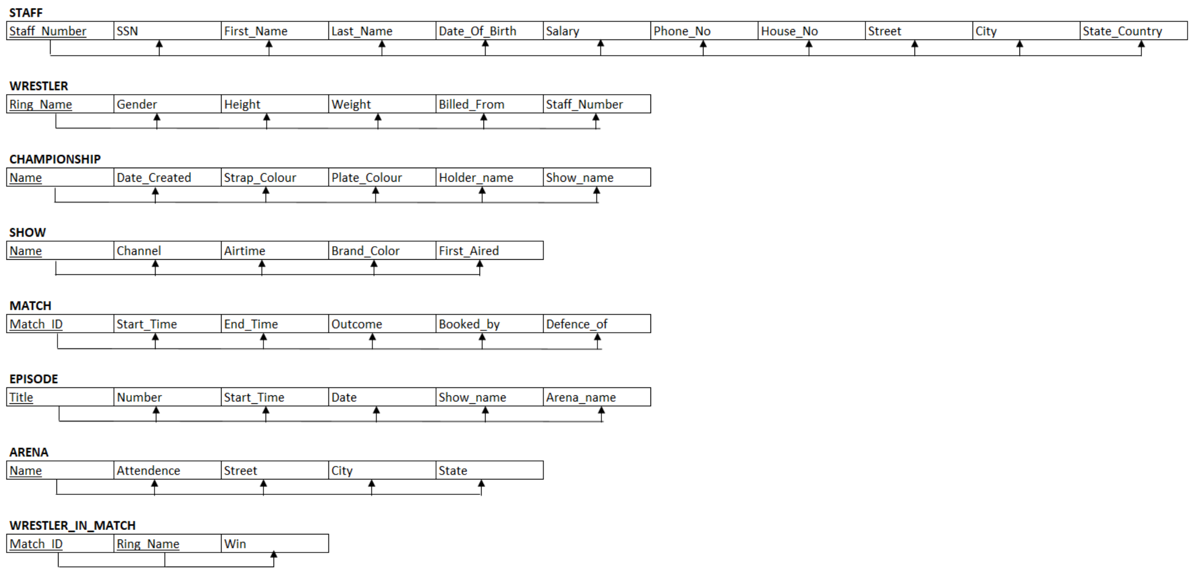
The database model is mostly comprised of 1:N relationships. These were all mapped by putting a foreign key on the N side, pointing to the primary key of the 1 side. This occurs multiple times in the model; one example being that there are N episodes of 1 show, so EPISODE has an attribute Show_name, which is a foreign key that points to SHOW.

There is an M:N relationship between WRESTLER and MATCH. For this, a new table WRESTLER_IN_MATCH was created. The primary key was a composite primary key made up of the foreign keys to WRESTLER and MATCH. An attribute of the relationship, Win, was also put into the table.


STAFF
Staff_Number | SSN | First_Name | Last_Name | Date_Of_Birth | Salary | Phone_No | House_No | Street | City | State_Country
WRESTLER
Ring_Name | Gender | Height | Weight | Billed_From | Staff_Number
CHAMPIONSHIP
Name | Date_Created | Strap_Colour | Plate_Colour | Holder_name | Show_name
SHOW
Name | Channel | Airtime | Brand_Color | First_Aired
MATCH
Match_ID | Start_Time | End_Time | Outcome | Booked_by | Defence_of
EPISODE
Title | Number | Start_Time | Date | Show_name | Arena_name
ARENA
Name | Attendence | Street | City | State
WRESTLER_IN_MATCH
Match_ID | Ring_Name | Win

Functional Dependency Diagram and Normalization

Due to the way in which the database was designed; i.e. an entity relationship model then translated into a relational schema using the rules above, the full design is in Boyce-Codd normal form. For every relation R, all attributes in R are *dependent on the key, the whole key, and nothing but the key*. We can see this more clearly by mapping out the functional dependencies as below. Each primary key fully determines the other attributes in the tuple. This eliminates redundancy and avoids unintended data loss when a tuple is modified/deleted.



Assumptions

A number of assumptions were taken about the real world semantics of the data on which the database was being modelled. These affected the planning of the database but also the implementation in SQL itself. Assumptions taken include the following:

- Staff phone numbers are U.S. phone numbers.
- Wrestler height is measured in centimetres.
- Wrestler weight is measured in kilograms.
- Multiple wrestlers can win one match, for example a tag-team match.
- Colours are stored in their hexadecimal encoding.
- Due to their variations in format from wrestler to wrestler, ring names and the location that they are billed from are stored as single strings.
- A wrestler's salary is their yearly salary in U.S. dollars.
- Each staff member has a unique staff number.
- Each match has a unique match ID.
- Each wrestler has a unique ring name.
- Each championship has a unique name.
- Each arena has a unique name.
- Each episode has a unique title.
- Gender is an attribute of wrestler to determine what matches they are eligible for, but not of staff as it is provides no useful information (general staff should discriminated against based on gender).

Semantic Constraints

Table Constraints

Some of the table constraints were chosen based on real world semantics. A good example of this is the STAFF relation. *Staff_Number* is a primary key, so it will be unique and not null. *SSN* is also unique and not null, as every staff member should have one, and duplicate SSNs would mean some sort of error. The *First_Name*, *Last_Name* and *Date_Of_Birth* fields should not be null as these are fundamental, basic pieces of information about the staff member, so there is no excuse for them to be null. Another constraint we see is a check that *Salary* is greater than zero, as it makes no sense to have a salary of zero or less. If there is no salary for a staff member for any reason, null is allowed (i.e. a guest appearance with a one off payment would mean no salary, rather than a salary of zero). The constraints on *Staff_Number*, *SSN*, *First_Name*, *Last_Name* and *Salary* are stated in this CREATE TABLE command:

```
CREATE TABLE STAFF (
    Staff_Number INTEGER NOT NULL,
    SSN INTEGER NOT NULL UNIQUE,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Date_Of_Birth DATE NOT NULL,
    Salary INTEGER,
    Phone_No CHAR(10),
    House_No VARCHAR(20),
    Street VARCHAR(20),
    City VARCHAR(20),
    State_Country VARCHAR(20),
    PRIMARY KEY (Staff_Number),
    CHECK (Salary > 0));
```

The participation constraints from the entity relationship model carried over to the creation of the tables also. For example, in the diagram we can see that a wrestler cannot exist without a corresponding staff member. So in the WRESTLER table, the foreign key to STAFF is NOT NULL. If the primary key of STAFF is updated (a staff member gets a new staff number), the update to cascade to the foreign key from WRESTLER, so this is specified when WRESTLER is created. A staff member should not be deleted if a wrestler instance still refers to them, so there is no ON DELETE statement. A design decision was made to use this approach (not null foreign keys which update on cascade) whenever mapping from participation constraints in the entity relationship model.

```
CREATE TABLE WRESTLER (
    …
    Staff_Number INTEGER NOT NULL,
    …
    FOREIGN KEY (Staff_Number) REFERENCES STAFF (Staff_Number) ON UPDATE
CASCADE,
    …);
```

The approach to constraints seen above (primary keys, real-life semantics and mapping from participation constraints) was taken when applying table constraints to any of the tables.

Triggers

Due to design decisions, most of the constraints implemented in the database were table constraints. It seemed clearer to have the constraints built into the table definition rather than a separate trigger on modification of a table. This is the reason that there are many constraints built into the CREATE statements above. However, another interesting use for triggers that was used was placing default values. I found this best to separate from the CREATE TABLE statements as ideally when a tuple is created, all known values will be specified. However, if either the creation date of a championship or the air date of a championship are left NULL, it would be reasonable to say that today's date is an expected value. So on both CHAMPIONSHIP and EPISODE, there are triggers which check if the date is NULL; if so, set the date to today's date.

```
CREATE TRIGGER Championship_Date
AFTER INSERT ON CHAMPIONSHIP
WHEN (new.Date_Created IS NULL)
DECLARE
    Championship_Name VARCHAR(50);
BEGIN
    Name := NEW.Championship_Name;
    UPDATE CHAMPIONSHIP
    SET Date_Created = CURRENT_DATE
    WHERE Name = Championship_Name;
END;
.
RUN;

CREATE TRIGGER Episode_Date
AFTER INSERT ON EPISODE
WHEN (new.Date IS NULL)
DECLARE
    Episode_Title VARCHAR(255);
BEGIN
    Episode_Title := NEW.Title;
    UPDATE EPISODE
    SET Date = CURRENT_DATE
    WHERE Title = Episode_Title;
END;
.
RUN;
```

Separating the more important constraints from dealing with default values breaks the code up and makes it easier to understand. Variables are also used in these triggers to make the code more readable; in this case they represent the name of the new championship or title of new episode after insert respectively.

Creating a View

Views are very useful mechanisms for SQL statements which may be repeated often, allowing users to query the view rather than repeating the more complicated statement again. They may also be used for security purposes to restrict the data that a user is shown. In this project I created three views on the database.

The first view is called PAYROLL, and can be interpreted as payroll information about a staff member, but without their personal information. The view contains three columns from STAFF, which are the staff number, SSN and salary. This view allows payroll information to be queried for a staff member without showing their name or other identifying information like their address.

```
CREATE VIEW PAYROLL (
    Staff_No, Staff_SSN, Staff_Salary)
    AS SELECT Staff_Number, SSN, Salary
    FROM STAFF;
```

The second view allows the personal information of a wrestler to be achieved by their ring name. This may be useful in the company if somebody wants to find out a wrestler's contact information via their ring name rather than their real name or staff number. As many wrestlers are commonly known by their ring names, this is a query that would be relatively common. This view does not contain any of the "payroll" columns above as these could be seen as more sensitive information and not information that should be queried by ring name.

```
CREATE VIEW WRESTLER_INFORMATION (
    Wrestler_Ring_Name, Wrestler_Gender, Wrestler_First_Name,
    Wrestler_Last_Name, Wrestler_Date_Of_Birth, Wrestler_Phone_No,
    Wrestler_House_No, Wrestler_Street, Wrestler_City,
    Wrestler_State_Country)
    AS SELECT WRESTLER.Ring_Name, WRESTLER.Gender, STAFF.First_Name,
      STAFF.Last_Name, STAFF.Date_Of_Birth, STAFF.Phone_No, STAFF.House_No,
      STAFF.Street, STAFF.City, STAFF.State_Country
    FROM WRESTLER, STAFF
    WHERE WRESTLER.Ring_Name = STAFF.Ring_Name;
```

Important information in a professional wrestling company is the history of title (championship) wins since the company started. The TITLE_WINS view shows the winner for every match where a championship was on the line. This brings back the wrestler's ring name, the championship name and the match ID, to allow the database to do further querying. The result is a history of all title wins in matches in WWE history.

```
CREATE VIEW TITLE_WINS (
    TITLE_WIN_WRESTLER, TITLE_WIN_CHAMPIONSHIP, TITLE_WIN_MATCH)
    AS SELECT WRESTLER_IN_MATCH.Ring_Name, MATCH.Defence_Of,
      WRESTLER_IN_MATCH.Match_ID
    FROM WRESTLER_IN_MATCH, MATCH
    WHERE WRESTLER_IN_MATCH.Match_ID = Match_ID
      AND MATCH.Defence_Of IS NOT NULL AND WRESTLER_IN_MATCH.Win = 1;
```

Appendix

Table Creation

```sql
CREATE TABLE STAFF (
    Staff_Number INTEGER NOT NULL,
    SSN INTEGER NOT NULL UNIQUE,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Date_Of_Birth DATE NOT NULL,
    Salary INTEGER,
    Phone_No CHAR(10),
    House_No VARCHAR(20),
    Street VARCHAR(20),
    City VARCHAR(20),
    State_Country VARCHAR(20),
    PRIMARY KEY (Staff_Number),
    CHECK (Salary > 0));

CREATE TABLE WRESTLER (
    Ring_Name VARCHAR(50) NOT NULL,
    Gender CHAR(1),
    Height INTEGER,
    Weight INTEGER,
    Billed_From VARCHAR(50),
    Staff_Number INTEGER NOT NULL,
    PRIMARY KEY (Ring_Name),
    FOREIGN KEY (Staff_Number) REFERENCES STAFF (Staff_Number)
      ON UPDATE CASCADE,
    CONSTRAINT CHK_Wrestler CHECK (Height > 0 AND Weight > 0));

CREATE TABLE SHOW (
    Name VARCHAR(20) NOT NULL,
    Channel VARCHAR(20),
    Airtime INTEGER,
    Brand_Color INTEGER,
    First_Aired DATE,
    PRIMARY KEY (Name));

CREATE TABLE CHAMPIONSHIP (
    Name VARCHAR(50) NOT NULL,
    Date_Created DATE,
    Strap_Colour INTEGER,
    Plate_Colour INTEGER,
    Holder_Name VARCHAR(50) NOT NULL,
    Show_Name VARCHAR(20) NOT NULL,
    PRIMARY KEY (Name),
    FOREIGN KEY (Holder_Name) REFERENCES WRESTLER (Ring_Name)
      ON UPDATE CASCADE,
    FOREIGN KEY (Show_Name) REFERENCES SHOW (Name) ON UPDATE CASCADE
);

CREATE TABLE ARENA (
    Name VARCHAR(255) NOT NULL,
    Attendence INTEGER,
    Street VARCHAR(20),
    City VARCHAR(20),
    State VARCHAR(20),
    PRIMARY KEY (Name),
    CHECK (Attendence > 0));

CREATE TABLE EPISODE(
    Title VARCHAR(255) NOT NULL,
    Number INTEGER,
    Start_Time TIME(0),
    End_Time TIME(0),
    Date DATE,
    Show_Name VARCHAR(20) NOT NULL,
    Arena_Name VARCHAR(255) NOT NULL,
    PRIMARY KEY (Title),
    FOREIGN KEY (Show_Name) REFERENCES SHOW (Name) ON UPDATE CASCADE,
    FOREIGN KEY (Arena_Name) REFERENCES ARENA (Name) ON UPDATE CASCADE,
    CHECK (Number >= 0));

CREATE TABLE MATCH (
    Match_ID INTEGER NOT NULL,
    Start_Time TIME(0),
    End_Time TIME(0),
    Outcome VARCHAR(20),
    Booked_By VARCHAR(255) NOT NULL,
    Defence_Of VARCHAR(50),
    PRIMARY KEY (Match_ID),
    FOREIGN KEY (Booked_By) REFERENCES EPISODE (Title) ON UPDATE CASCADE,
    FOREIGN KEY (Defence_Of) REFERENCES CHAMPIONSHIP (Name)
      ON UPDATE CASCADE);

CREATE TABLE WRESTLER_IN_MATCH (
    Match_ID INTEGER NOT NULL,
    Ring_Name VARCHAR(50) NOT NULL,
    Win BIT(1),
    PRIMARY KEY (Match_ID, Ring_Name),
    FOREIGN KEY (Match_ID) REFERENCES MATCH (Match_ID) ON UPDATE CASCADE,
    FOREIGN KEY (Ring_Name) REFERENCES WRESTLER (Ring_Name)
      ON UPDATE CASCADE);
```

Triggers

```
CREATE TRIGGER Championship_Date
AFTER INSERT ON CHAMPIONSHIP
WHEN (new.Date_Created IS NULL)
DECLARE
    Championship_Name VARCHAR(50);
BEGIN
    Name := NEW.Championship_Name;
    UPDATE CHAMPIONSHIP
    SET Date_Created = CURRENT_DATE
    WHERE Name = Championship_Name;
END;
.
RUN;

CREATE TRIGGER Episode_Date
AFTER INSERT ON EPISODE
WHEN (new.Date IS NULL)
DECLARE
    Episode_Title VARCHAR(255);
BEGIN
    Episode_Title := NEW.Title;
    UPDATE EPISODE
    SET Date = CURRENT_DATE
    WHERE Title = Episode_Title;
END;
.
RUN;
```

Views

```
CREATE VIEW PAYROLL (
    Staff_No, Staff_SSN, Staff_Salary)
    AS SELECT Staff_Number, SSN, Salary
    FROM STAFF;

CREATE VIEW WRESTLER_INFORMATION (
    Wrestler_Ring_Name, Wrestler_Gender, Wrestler_First_Name,
    Wrestler_Last_Name, Wrestler_Date_Of_Birth, Wrestler_Phone_No,
    Wrestler_House_No, Wrestler_Street, Wrestler_City, Wrestler_State_Country)
    AS SELECT WRESTLER.Ring_Name, WRESTLER.Gender, STAFF.First_Name,
      STAFF.Last_Name, STAFF.Date_Of_Birth, STAFF.Phone_No, STAFF.House_No,
      STAFF.Street, STAFF.City, STAFF.State_Country
    FROM WRESTLER, STAFF
    WHERE WRESTLER.Ring_Name = STAFF.Ring_Name;
```

```sql
CREATE VIEW TITLE_WINS (
    TITLE_WIN_WRESTLER, TITLE_WIN_CHAMPIONSHIP, TITLE_WIN_MATCH)
    AS SELECT WRESTLER_IN_MATCH.Ring_Name, MATCH.Defence_Of,
      WRESTLER_IN_MATCH.Match_ID
    FROM WRESTLER_IN_MATCH, MATCH
    WHERE WRESTLER_IN_MATCH.Match_ID = Match_ID
      AND MATCH.Defence_Of IS NOT NULL AND WRESTLER_IN_MATCH.Win = 1;
```

Insertions

```sql
INSERT INTO STAFF VALUES(10000000, 12345670, 'Colby', 'Lopez',
      '1986-05-28', 3000000, '2025550148', '2625', '56th Court',
      'Davenport', 'IA');
INSERT INTO STAFF VALUES(10000001, 12345669, 'Joe', 'Anoai',
      '1985-05-25', 5000000, '2025550160', '2291', 'Main Street',
      'Minneola', 'FL');
INSERT INTO STAFF VALUES(10000002, 12345668, 'Brock', 'Lesnar',
      '1977-06-12', 6500000, '2025550179', '5141', 'Mallard Lane',
      'Alexandria', 'MN');
INSERT INTO STAFF VALUES(10000003, 12345667, 'Fergal', 'Devitt',
      '1981-07-25', 1000000, '2025550135', '5055', 'Forsyth Commerce Road',
      'Orlando', 'FL');
INSERT INTO STAFF VALUES(10000004, 12345666, 'Rebecca', 'Quin',
      '1987-01-30', 250000, '2025550128', '3300', 'University Park Drive',
      'Orlando', 'FL');
INSERT INTO STAFF VALUES(10000005, 12345665, 'Ashley', 'Fliehr',
      '1986-04-05', 550000, '2025550177', '5055', 'Forsyth Commerce Road',
      'Orlando', 'FL');

INSERT INTO WRESTLER VALUES('Seth Rollins', 'M', 185, 98,
      'Davenport, Iowa', 10000000);
INSERT INTO WRESTLER VALUES('Roman Regins', 'M', 191, 120,
      'Pensacola, Florida', 10000001);
INSERT INTO WRESTLER VALUES('Brock Lesnar', 'M', 191, 130,
      'Minneapolis, Minnesota', 10000002);
INSERT INTO WRESTLER VALUES('Finn Balor', 'M', 180, 86,
      'Bray, County Wicklow, Ireland', 10000003);
INSERT INTO WRESTLER VALUES('Becky Lynch', 'F', 168, 61,
      'Dublin, Ireland', 1000004);
INSERT INTO WRESTLER VALUES('Charlotte Flair', 'F', 178, NULL,
      'The Queen City', 1000005);

INSERT INTO SHOW VALUES('Raw', 'USA Network', 3, 14876959, '1993-11-01');
INSERT INTO SHOW VALUES('SmackDown', 'Fox Sports 1', 2, 158408, '1999-04-29');
INSERT INTO SHOW VALUES('NXT', 'USA Network', 2, 15913216, '2012-02-29');


INSERT INTO CHAMPIONSHIP VALUES('WWE Championship', '1963-04-25',
      0, 13938487, 'Brock Lesnar', 'Raw');
INSERT INTO CHAMPIONSHIP VALUES('Universal Championship', '2016-07-25',
      255, 13938487, 'Roman Reigns', 'SmackDown');
INSERT INTO CHAMPIONSHIP VALUES('NXT Championship', '2014-06-01',
      0, 13938487, 'Finn Balor', 'NXT')
INSERT INTO CHAMPIONSHIP VALUES('Raw Women Championship', '2016-08-03',
      16777215, 16711680, 'Becky Lynch', 'Raw');

INSERT INTO ARENA VALUES('Madison Square Garden', 18500,
      'Pennsylvania Plaza', 'New York', 'NY');
INSERT INTO ARENA VALUES('Allstate Arena', 18200,
      'North Mannheim Road', 'Rosemont', 'IL');
INSERT INTO ARENA VALUES('Wells Fargo Center', 18000,
      'South Broad Street', 'Philadelphia', 'PA');
INSERT INTO ARENA VALUES('Staples Center', 21000,
      'South Figueroa Street', 'Los Angeles', 'CA');
INSERT INTO ARENA VALUES('TD Garden', 18000,
      'Legends Way', 'Boston', 'MA');

INSERT INTO EPISODE VALUES('Raw Pilot', 1, '20:00:00', '23:00:00',
      '1993-11-01', 'Raw', 'Madison Square Garden');
INSERT INTO EPISODE VALUES('SmackDown Pilot', 1, '20:00:00', '22:00:00',
      '1999-04-29', 'SmackDown', 'Allstate Arena');
INSERT INTO EPISODE VALUES('NXT Pilot', 1, '20:00:00', '22:00:00',
      '2012-02-29', 'NXT', 'Wells Fargo Center');

INSERT INTO MATCH VALUES(10001, '20:10:00', '20:40:00', 'Pinfall',
      'Raw Pilot', 'WWE Championship');
INSERT INTO MATCH VALUES(10002, '21:00:00', '20:20:00', 'Submission',
      'Raw Pilot', NULL);
INSERT INTO MATCH VALUES(20001, '20:15:00', '20:30:00', 'Pinfall',
      'SmackDown Pilot', NULL);
INSERT INTO MATCH VALUES(20002, '20:50:00', '21:15:00', 'Disqualification',
      'SmackDown Pilot', NULL);
INSERT INTO MATCH VALUES(30001, '20:05:00', '20:30:00', 'Submission',
      'NXT Pilot', NULL);
INSERT INTO MATCH VALUES(30002, '21:00:00', '21:40:00', 'Pinfall',
      'NXT Pilot', NULL);

INSERT INTO MATCH VALUES(10001, 'Brock Lesnar', '1');
INSERT INTO MATCH VALUES(10001, 'Roman Reigns', '0');
INSERT INTO MATCH VALUES(10002, 'Seth Rollins', '1');
INSERT INTO MATCH VALUES(10002, 'Finn Balor', '0');
INSERT INTO MATCH VALUES(20001, 'Becky Lynch', '1');
INSERT INTO MATCH VALUES(20001, 'Charlotte Flair', '0');
INSERT INTO MATCH VALUES(20002, 'Seth Rollins', '1');
INSERT INTO MATCH VALUES(20002, 'Finn Balor', '1');
INSERT INTO MATCH VALUES(20002, 'Roman Reigns', '0');
INSERT INTO MATCH VALUES(20002, 'Brock Lesnar', '0');
INSERT INTO MATCH VALUES(30001, 'Finn Balor', '1');
INSERT INTO MATCH VALUES(30001, 'Charlotte Flair', '1');
INSERT INTO MATCH VALUES(30001, 'Seth Rollins', '0');
INSERT INTO MATCH VALUES(30001, 'Becky Lynch', '0');
INSERT INTO MATCH VALUES(30002, 'Roman Reigns', '1');
INSERT INTO MATCH VALUES(30002, 'Seth Rollins', '0');
```