IST 652

Joseph Davis

**Project Overview**

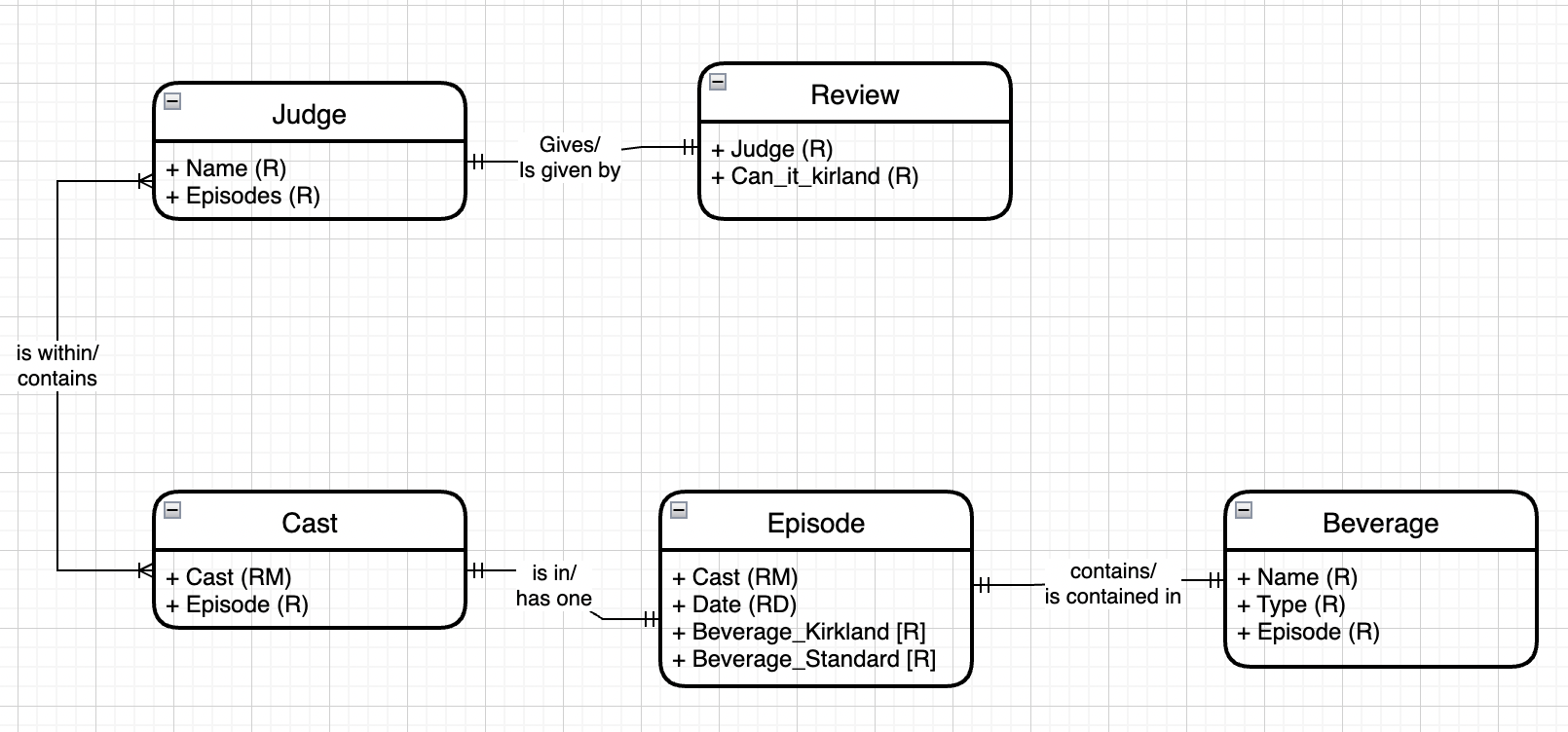
The intent of this database was to store the data generated from the tik tok review show *Can it Kirkland?*. The show has a basic premise; the contestants try a Kirkland brand alcoholic drink and its name brand counterpart and try to pick which one is Kirkland. If they guess that the Kirkland brand is the name brand that is a point for Kirkland. If they guess properly then that’s a point against Kirkland. These points determine “can it Kirkland?”. The intent is to use the date, episode, cast, beverage, and reviews of the judges to build an application that will allow users to look up the reviews for a particular kind of drink in Costco and see how it compares to the standard brand.

**Team Log**

All work was done by Joseph Davis April 7th-April 9th.

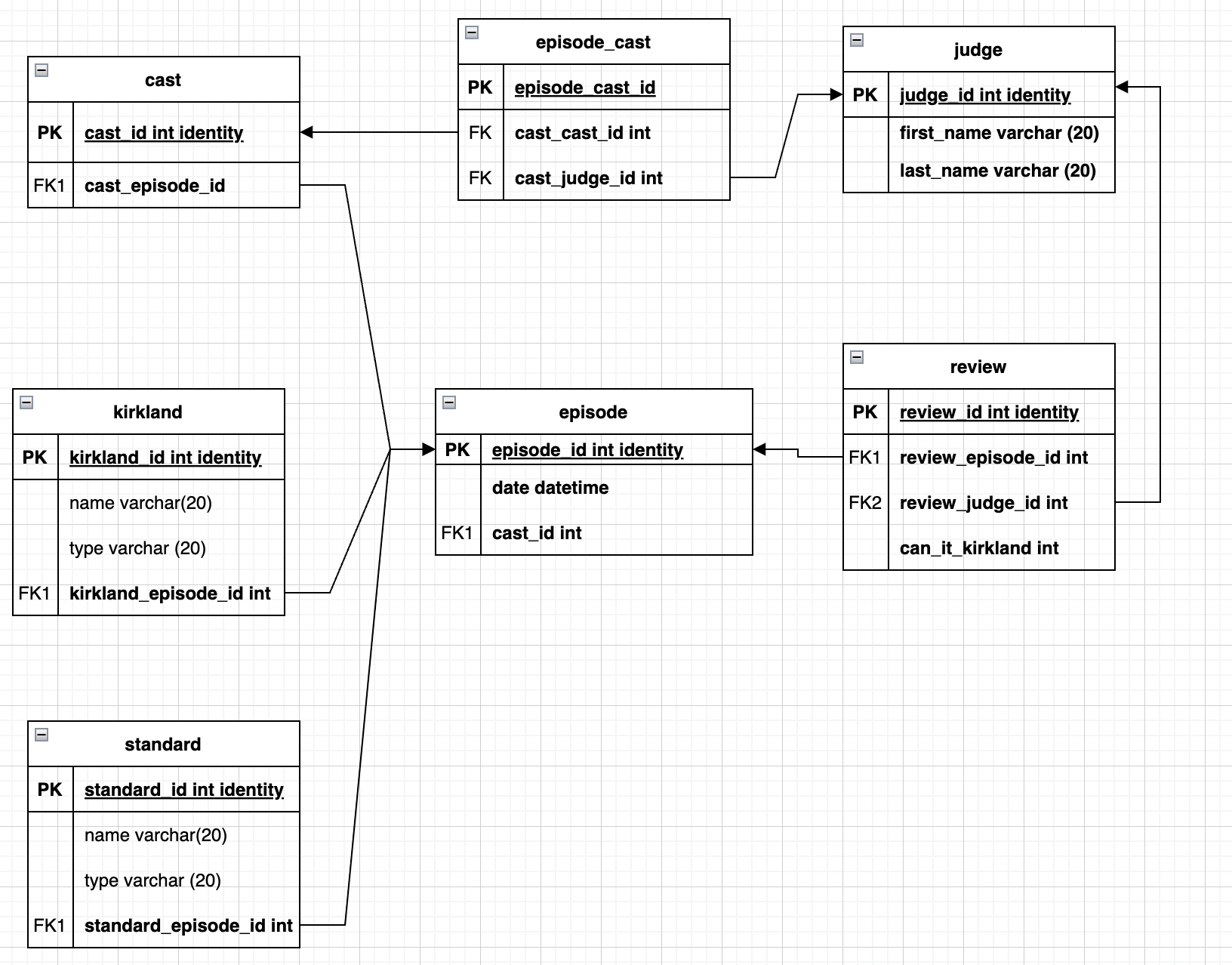
**Conceptual Model**

As stated earlier the conceptual model will house a few entities. These are the reviews given, the judges, the cast for that episode, the episode itself, and the kind of beverage. A conceptual model has been created and can be seen below.



As we can see, the review itself will be given by one judge and one judge only, and a judge can only give one review per drink. The review just consists of guessing which drink is which, if the judges guess correctly more than incorrectly then it cannot Kirkland. The judges include repeats for each episode and sometimes guest judges. Therefore, a judge can be on one or multiple casts for the episode. The cast includes one or many judges. The cast can appear on one and only one episode. There are repeat casts, however each cast will appear on one episode once. The episode will contain one cast only, making this a 1 to 1 relationship. The episode consists of the cast, date, and the beverages. Finally, each episode has one and only one beverage and the beverages are not repeated between episodes so this is also a 1 to 1 relationship.

Next a logical model was created from this conceptual model. This model can be seen below.



The focal point of this logical model centers around the episode. The episode\_id is referenced in many other tables. The tables themselves are very straightforward. The judges will have a table, and that will be referenced in both the reviews and episode\_cast. The episode\_cast will bridge the cast and judge’s tables. The cast then connects to the episode table. The Kirkland and standard tables will reference the episode table as each episode will have a different drink. Finally, the review table uses the episode\_id as a foreign key to connect the episode to the reviews.

**Reflection**

In conclusion the start of a potential database was created. It may have utility for people who regularly shop at Costco and shows different relationships between different entities. I don’t have enough time to put data into the database, create a presentation, or implement the application. Given more time, I would of course like to complete these things as well as a video presentation, however that isn’t feasible. I hope you can at least follow the conceptual beginnings of a future useful application.

**SQL Script**

create database FINAL

GO

alter table standard

drop CONSTRAINT if exists standard\_episode

GO

alter table review

drop CONSTRAINT if exists review\_judge

GO

alter table review

drop CONSTRAINT if exists review\_episode

GO

alter table kirkland

drop CONSTRAINT if exists kirkland\_episode

GO

alter table episode\_cast

drop CONSTRAINT if exists episode\_cast\_judge

GO

alter table episode\_cast

drop CONSTRAINT if exists episode\_cast\_cast

GO

alter table cast

drop CONSTRAINT if exists cast\_episode

GO

drop table if exists cast

GO

CREATE TABLE cast (

cast\_id int NOT NULL,

cast\_episode\_id int NOT NULL,

CONSTRAINT cast\_pk PRIMARY KEY (cast\_id)

);

-- Table: episode

drop table if exists episode

GO

CREATE TABLE episode (

episode\_id int NOT NULL,

date date NOT NULL,

cast\_id int NOT NULL,

CONSTRAINT episode\_pk PRIMARY KEY (episode\_id)

);

-- Table: episode\_cast

drop table if exists episode\_cast

GO

CREATE TABLE episode\_cast (

episode\_cast\_id int NOT NULL,

cast\_cast\_id int NOT NULL,

cast\_judge\_id int NOT NULL,

CONSTRAINT episode\_cast\_pk PRIMARY KEY (episode\_cast\_id)

);

-- Table: judge

drop table if exists judge

GO

CREATE TABLE judge (

judge\_id int NOT NULL,

first\_name varchar(20) NOT NULL,

last\_name varchar(20) NOT NULL,

CONSTRAINT judge\_pk PRIMARY KEY (judge\_id)

);

-- Table: kirkland

drop table if exists kirkland

GO

CREATE TABLE kirkland (

name varchar(20) NOT NULL,

type varchar(20) NOT NULL,

kirkland\_id int NOT NULL,

kirkland\_episode\_id int NOT NULL,

CONSTRAINT kirkland\_pk PRIMARY KEY (kirkland\_id)

);

-- Table: review

drop table if exists review

GO

CREATE TABLE review (

review\_id int NOT NULL,

can\_it\_kirkland int NOT NULL,

review\_episode\_id int NOT NULL,

review\_judge\_id int NOT NULL,

CONSTRAINT review\_pk PRIMARY KEY (review\_id)

);

-- Table: standard

drop table if exists standard

GO

CREATE TABLE standard (

name varchar(20) NOT NULL,

type varchar(20) NOT NULL,

standard\_episode\_id int NOT NULL,

standard\_id int NOT NULL,

CONSTRAINT standard\_pk PRIMARY KEY (standard\_id)

);

-- foreign keys

-- Reference: cast\_episode (table: cast)

ALTER TABLE cast ADD CONSTRAINT cast\_episode

FOREIGN KEY (cast\_episode\_id)

REFERENCES episode (episode\_id);

-- Reference: episode\_cast\_cast (table: episode\_cast)

ALTER TABLE episode\_cast ADD CONSTRAINT episode\_cast\_cast

FOREIGN KEY (cast\_cast\_id)

REFERENCES cast (cast\_id);

-- Reference: episode\_cast\_judge (table: episode\_cast)

ALTER TABLE episode\_cast ADD CONSTRAINT episode\_cast\_judge

FOREIGN KEY (cast\_judge\_id)

REFERENCES judge (judge\_id);

-- Reference: kirkland\_episode (table: kirkland)

ALTER TABLE kirkland ADD CONSTRAINT kirkland\_episode

FOREIGN KEY (kirkland\_episode\_id)

REFERENCES episode (episode\_id);

-- Reference: review\_episode (table: review)

ALTER TABLE review ADD CONSTRAINT review\_episode

FOREIGN KEY (review\_episode\_id)

REFERENCES episode (episode\_id);

-- Reference: review\_judge (table: review)

ALTER TABLE review ADD CONSTRAINT review\_judge

FOREIGN KEY (review\_judge\_id)

REFERENCES judge (judge\_id);

-- Reference: standard\_episode (table: standard)

ALTER TABLE standard ADD CONSTRAINT standard\_episode

FOREIGN KEY (standard\_episode\_id)

REFERENCES episode (episode\_id);

-- End of file.