CS4320 HW3
Name: Yiwen Huang (yh385), Tin Kuo (yk638), Jason Han (jh995)

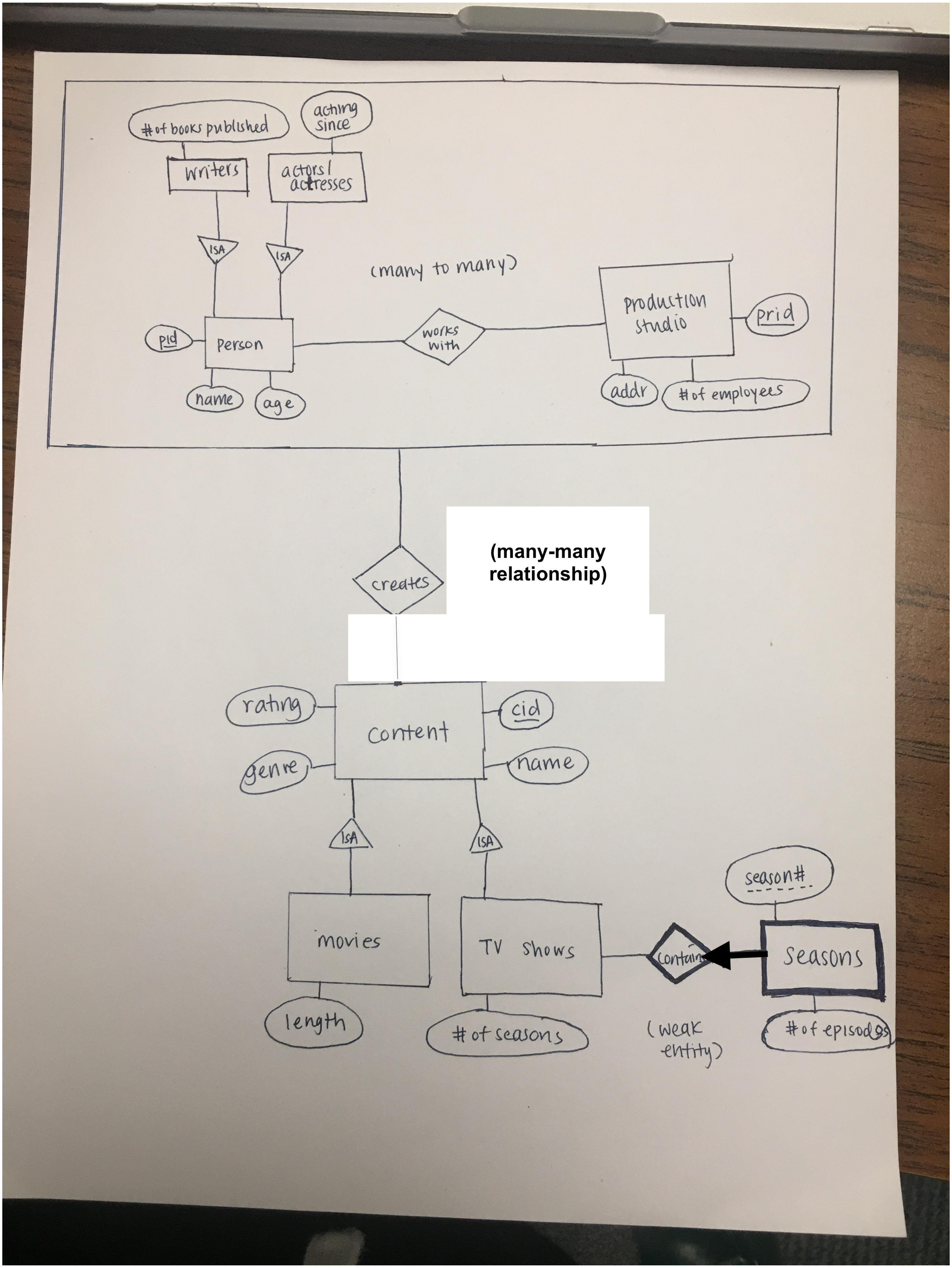
Description

Netflix is a widely popular media distribution platform. Netflix hosts two different kinds of content, TV Shows and Movies. Each of these has a rating, a genre, and a name. Movies have a length in minutes and TV shows have a count of the number of seasons. These TV shows are composed of seasons, each season is identified by its season number and the number of episodes. These two types of content are both created by actors/actresses, writers, and production studios. Actors/Actresses are both types of people that have names, and ages. These people work with the production studios which each have an address and a number of employees.

SQL Query

```
CREATE TABLE Person ( -- entity
    pid INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    age INTEGER NOT NULL CHECK (age ≥ 0)
);

CREATE TABLE Content ( -- entity
    cid INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    rating INTEGER,
    genre VARCHAR(100) NOT NULL,
);

CREATE TABLE Writer ( -- subclass
    pid INTEGER PRIMARY KEY,
    num_books INTEGER NOT NULL CHECK (num_books ≥ 0),
    FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE ActorAndActress ( -- subclass
    pid INTEGER PRIMARY KEY,
    since DATE NOT NULL,
    FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE ProductionStudio ( -- entity
    prid INTEGER PRIMARY KEY,
    addr VARCHAR(500) NOT NULL,
    num_employee INTEGER NOT NULL CHECK (num_employee ≥ 0)
);

CREATE TABLE WorksWith ( --many-many relationship
    pid INTEGER NOT NULL,
    prid INTEGER NOT NULL,
    PRIMARY KEY (pid, prid),
    FOREIGN KEY (pid) REFERENCES Person(pid),
    FOREIGN KEY (prid) REFERENCES ProductionStudio(prid)
);

CREATE TABLE Creates ( --relates work with to contents one-one relationship
    pid INTEGER NOT NULL,
    prid INTEGER NOT NULL,
    cid INTEGER NOT NULL,
    PRIMARY KEY (pid, prid, cid),
    FOREIGN KEY (pid, prid) REFERENCES WorksWith(pid, prid),
    FOREIGN KEY (cid) REFERENCES Content(cid)
);
```

```
CREATE TABLE Movies ( -- subclass
    cid INTEGER PRIMARY KEY,
    length INTEGER NOT NULL CHECK (length ≥ 0),
    FOREIGN KEY (cid) REFERENCES Content(cid)
);

CREATE TABLE TVShows ( --subclass
    cid INTEGER PRIMARY KEY,
    num_seasons INTEGER NOT NULL CHECK (num_seasons > 0),
    FOREIGN KEY (cid) REFERENCES Content(cid)
);

CREATE TABLE Seasons ( --weak entity
    cid INTEGER NOT NULL,
    season_num INTEGER NOT NULL,
    num_episodes INTEGER NOT NULL CHECK (num_episodes ≥ 1),
    PRIMARY KEY (cid, season_num),
    FOREIGN KEY (cid) REFERENCES TVShows(cid) ON DELETE CASCADE
);
```



(many-many relationship)

The entity *content* represents all the different types of things you can watch on Netflix, each tuple is given a "content id" (cid) which is also the primary key for this table. This allows us to identify each content individually. The other attributes of this table is *name*, *rating*, *genre* which was required by the problem. Then since there are two different types of content as stated in the problem, (i.e. movies and TV shows), there are two subclasses for each type which is shown in the diagram and each of the subclasses have their own special attributes in addition to the attributes in *contents* (i.e. movies have length while TV shows have number of seasons). The two subclasses are represented by the table *movies* and $TVShows$ respectively and the primary key for both of those tables the primary key is *cid*. Then we have the weak entity *seasons* whose owner entity is $TVShows$. This means that for one TV Show, there can be many seasons and seasons must total participation in this identifying relationship. In addition, seasons can not be uniquely identified without knowing which TV Show it is (i.e. we need the primary key *cid* and the partial key *season_num*), therefore it is a weak entity represented by the table *seasons* whose primary key is $(cid, season\_num)$ and this captures the key constraint.

Then for the upper-part of the ER diagram, we have the entity *person* whose primary key is *pid* and it has the attributes *name* and *age*. Then we have the two subclasses of *person* which are writers and actors/actresses. Both of these things have the attributes *name* and *age* to satisfy the requirements given in the problem, but they can also have their own special properties that can only pertain to themselves (e.g. writers can have number of books published while actors and actresses can have a date since they first started acting). And so the tables to represent those two subclasses are *writers* and *actorsAndActresses* respectively, and for both of the table the primary key is *pid* which is the person id number (the primary key from *person* table). Next we have the *productionStudio* entity which has the attributes *address* and *number of employees* as required by the problem. Then we set up a primary key for this entity table *prid* which is production id so we can uniquely identify each production studio. Then we have the relationship *works with* which is a many-many relationship between *person* and *productionStudio* because one person can work with many production studios and a production studio can work with many people. To represent this relationship, we have the table *workswith* whose primary key is $(pid, prid)$ to allow multiple combinations of people and production studios.

This then becomes an aggregated relationship. We assume that a "content" can have many production/person pairs and that the production/person pair can produce many "contents". We also did not impose a participation constraint on both ways because it is possible for people and production studio to collaborate and have no yet produced a content yet. And we assume it is possible for a content to not have a producer (this was discussed with a T.A.). Therefore this aggregation results in a many-many relationship with content. This is represented in the relationship *creates* and the many-many relation ship is represented in our SQL query when we make the primary key of the table *creates* (pid, prid, cid) to allow all combination. In addition, we require that *pid*, *prid* and *cid* is NOT NULL so that it is only only in this table if a person worked with a production studio on a "content". All this comes together to correctly model the information given from the problem.