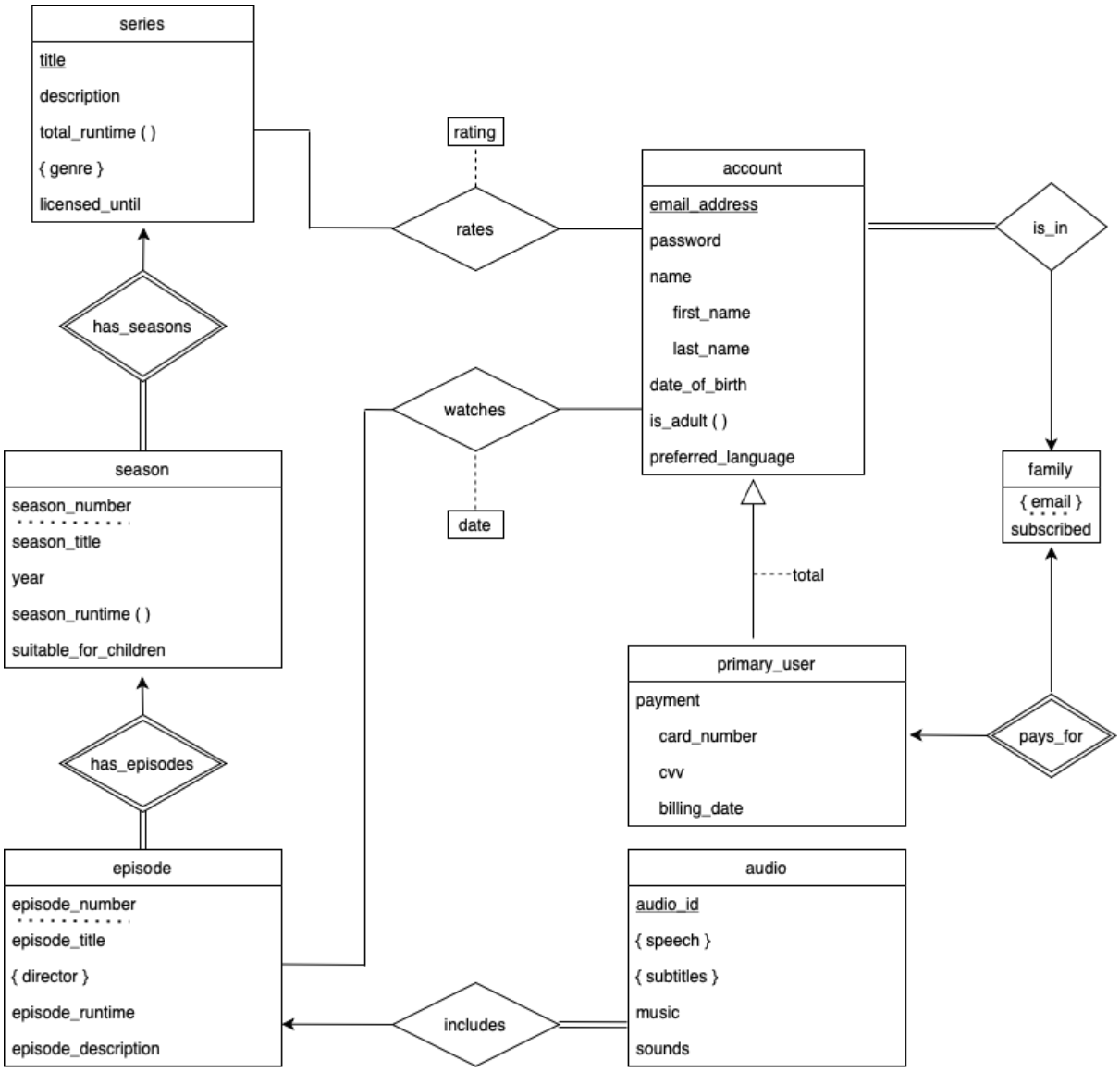
Database Design

CS3101 - Databases

E-R Diagram



My E-R diagram comprises seven entity sets and seven relationship sets. I have assumed that no two series will have the same name in order to use the series name as a primary key. Additionally, every series has seasons and every season has episodes where the series title is the primary key while the season number and episode number are the foreign keys. This allows episodes to be uniquely referenced in a logical chronological fashion.

The relation between non-primary accounts and the family entity introduces ambiguity. This is because all accounts either pay for or are a member of a family subscription. A user should only be able to rate and watch content if they are either a primary user or in a family with a primary user. This condition isn't represented in the database. My solution to this was to have a subscribed attribute in the family entity set which acts as a flag to tell the platform if the family includes a paying user. The relation between the account entity and the family entity has mandatory participation because accounts without a subscription cannot use the platform.

There are three weak entity sets: season, episode, and family. Season uses series as an identifying entity set instead of using it's own ID because the existence of seasons is dependent on the existence of the series. The same logic follows for the episode entity set; it is defined by the episode number key and its relation to the season entity which is in turn defined by the season number key and its relation to a series. The primary user entity set has total inheritance from the account entity set which allows the existence of accounts with and without payment information. Both the account and the primary user have relations to the family entity but with different participation and cardinality constraints. I chose to have the primary user as a subclass of account to avoid the redundancy of having two separate entity sets.

I gave audio its own entity set to store different sounds including music along with speech in different languages. This links with the account attribute preferred_language to provide the user with access to their native language.

My model uses derived attributes in two ways, firstly to calculate the season runtime and series runtime as the sum of the episode runtimes. Secondly, the account entity set has an is_adult attribute which will flag as true if the time since the user's date of birth exceeds 18 years.

Relational Model

Series is a strong entity set and so it is simply described by its attributes and its primary key. However, it contains a multivalued attribute, genre, which must be represented in its own relation schema since the relational model does not include multivalued attributes. The series_genre uses the series title as a foreign key and the genre as a discriminator. Season is a weak entity set and so it needs the series title as a foreign key and the season number as the discriminator. Episode is similar and uses both the series title and the season number as primary keys along with the discriminating attribute, episode number. The episode director is a multivalued attribute to allow for collaboration. This translates into the relational model as a schema with four candidate keys.

The primary user entity set inherits from account and so its representation in the model contains all the attributes from account plus the payment attributes which are stored in the primary user entity set. Family is a weak entity set which uses a multivalued attribute as the candidate key. The set of email addresses that belong to a family is the discriminating attribute which is represented in the relation schemas family and family emails.

Entities

- series (title: string, description: string, total_runtime: interval, licensed_until: datetime)
- series_genre (series_title*: string, genre: string) - separate schema for multivalued attribute ‘genre’
- season (title*: string, season_number: integer, year: integer, season_runtime: interval, suitable_for_children: boolean)
- episode (title*: string, season_number*: integer, episode_number: integer, episode_title: string, episode_runtime: interval, episode_description: string)
- episode_director (title*: string, season_number*: integer, episode_number*: integer, director: string)
- audio (audio_id: integer, music: blob, sounds: blob)
- episode_speech (audio_id*: integer, audio: blob)
- episode_subtitles (audio_id*: integer, subtitles: string)
- account (email_address: string, password: string, first_name: string, last_name: string, date_of_birth: date, is_adult: boolean, preferred_language: string)
- primary_user (email_address*: string, password: string, first_name: string, last_name: string, date_of_birth: date, is_adult: boolean, preferred_language: string, card_number: integer, cvv: integer, billing_date: datetime)
- family (email_address*: string, subscribed: boolean)
- family_emails (email_address*: string, email: string)

Relations

- has_seasons (title*: string, season_number: number, season_title: string, year: integer, season_runtime: interval, suitable_for_children: boolean)
- has_episodes (title*: string, season_number: integer, episode_number: integer, episode_title: string, director: string, episode_runtime: interval, episode_description: string)
- includes (title*: string, season_number*: string, episode_number: integer, audio_id: integer, speech: blob, subtitles: string, music: blob, sounds: blob)
- rates (title: string, email_address: string, rating: boolean)
- watches (title*: string, season_number*: integer, episode_number: integer, email_address: string, date: date)
- is_in (email_address: string, email: string)
- pays_for (email_address: string, email: string)

Non-Trivial Functional Dependencies

email_address → card_number

title → description

episode_number → episode_description

episode_number → audio_id

audio_id → speech

audio_id → subtitles

Card numbers are unique to each payment card holder and so card_number is functionally determined by age primary key, email address. Series is a strong entity set with a functional dependency from the primary key, title, to the series description which will be unique for each series. The audio entity set is also strong and has two attributes which will be unique for each episode, speech and subtitles, thus there are two functionally dependencies. Initially, I included season title and episode title as being functionally determined by their respective season and episode numbers. However, this implies that no two series can have the same season or episode names which is an unnecessary restriction to impose. Lastly, the discriminator of episode maps uniquely to both the episode description and the audio_id. This is because, regardless of shared music or sounds, each episode will map to only one instance of audio which is referenced using the audio_id.

Normalisation

First, second, third, or Boyce-Codd normal form with justification and reference to the dependencies.

The relational model is at in first normal form since every multivalued attribute is split into two relation schemas preserving the atomicity condition. Additionally, all values stored in each attribute are in the same domain, each attribute has a unique name, and the order in which the data is stored is not important.

Second normal form requires the model to be in first normal form and for each attribute to either be part of a candidate key or fully functionally dependent on a candidate key. My relational model does not satisfy this constraint and in fact breaks these conditions at least once in each entity set. Series could (in theory) have the same runtimes, genres, or licensing dates and audio instances could share music or sounds. Also, many accounts will share the same preferred language which is not functionally dependent on any candidate keys and is not a candidate_key itself.

Word Count

1,243