MET CS 669 Database Design and Implementation for Business
Term Project Iteration 3

Name: Megha Samala

Table of Contents

Project Direction Overview

I would like to create a web application designed to track a user's media consumption across several different mediums. The application, which I will call "TunedIn," will be a place where users can log the different media they're currently consuming, have consumed, or plan to consume. The media in question can be movies, TV shows, books, music (in album form), and video games. Additionally, the user will be able to track information about the vendor of the media and use this information to organize and revisit their media buys as well as give recommendations to others

I believe an application like this is important because an overall media tracking system does not appear to exist, leaving people to log their consumption on multiple different websites dedicated to only a specific medium (think Letterboxd or MyAnimeList), or create a Twitter thread or blog dedicated to tracking, or alternatively not track their watches/reads/listens at all. As someone who considers herself to be scatterbrained, I also often jump from media to media without finishing a series or completing a playthrough, and I forget what episode I left on or what chapter of a book I was on, and sometimes I completely forget that I was in the middle of watching or reading something. When people ask me what I've been watching or reading or playing, or what my favorite movie or show is, I often blank and fail to answer the question. I can safely presume that I am not the only one with this issue, and therefore an application dedicated to tracking the status of someone's varied media consumption is a convenient solution. I also believe that tracking media consumption encourages users to think more critically about what they are consuming and how they feel about it, rather than finishing a series or book or game and simply moving on to the next one.

TunedIn will offer a place for users to input information about a piece of media, the main focus being on a name, a rating (if completed/desired), and a status on the consumption. Additional information can be added based on relevance, such as author information for a book, or the URL for something consumed online so it can easily be provided to a friend when recommending the media to them, or a link to another existing database-esque website which may contain a blurb/critic ratings of the media (again, think Letterboxd or MyAnimeList or IMDB, etc), and more. Users will be able to log what chapter of a book they are on, or what episode of a show they are on, so that if they drop it and choose to resume it later, they can easily pick up from where they left off. Additionally, users can track when they started and finished a piece of media, to log past watches, and/or determine if it might be time to revisit something they previously consumed and enjoyed. Each kind of media has information that is specific to it, as well as information that can be applicable to multiple kinds of media, such as a genre and a review. Through my multiple project iterations, I will be able to refine my scope and determine what information is most necessary and relevant for the user to have logged, and create the most useful version of my database.

Use Cases and Fields

Use Case #1: Account Setup

Because TunedIn will be a web application, it will be necessary to register to keep track of each individual user's media.

Field	What it stores	Why it's needed
username	This stores a username associated with each account	Users can share usernames to see what another user is

		consuming, and users can also have multiple accounts if they would prefer
first_name	This stores the user's first name.	This can be displayed on screen and shared with other users.
last_name	This stores the user's first name.	This can be displayed on screen and shared with other users.
account_created	This stores the date on which the user created the account.	Users may want to know how long they have been using the application to track their media consumption.

Use Case #2: Logging a Movie
A movie will be one of the media types that the application will encourage users to log, and has fields that are specific to it. The other media types will track similar information and will be able to be differentiated in the database using a flag.

Field	What it stores	Why it's needed
movie_name	This stores the name of the movie.	This is necessary for the user to know what movie they watched.
director	This stores the name of the director of the movie.	This is useful for the user if they want to track how much they like movies by a certain director or want to sort by a certain director.
production_company	This stores the name of the production company of the movie, such as A24 or Studio Ghibli.	This is useful for the user to track how much they like the works of a certain company or sort by them.
date_released	This stores the date that the movie was released.	This helps the user determine their viewing trends by date and also distinguish between potential movie remakes or remasters with different release dates.

Use Case #3: Logging a Genre
Since genres are applicable to any media type, users will be able to log them for any media consumed.

Field	What it stores	Why it's needed
category	This is a catch-all for the different media types where the category/large genres can be stored, such as things like fiction or nonfiction for books, documentary or short film for movies, RPG or shooter for video games, etc.	This allows for users to sort and search for certain pieces of media in a broad way.
main_genre	This stores the primary genre of a piece of media, which can differ in definition based on the user, but is generally a broader category such as horror, action, etc.	This allows for users to sort media by genre when organizing and also looking for previously logged media.
subgenre	This stores the subgenre of a piece of a media, which can be a specifier, like historical fantasy vs urban fantasy, etc.	This is optional but allows for more specific searching and sorting for the user.

Use Case #4 - Logging a Review
Since reviews are applicable to any media type, users will be able to log them for any media consumed.

Field	What it stores	Why it's needed
rating	This stores a number rating out of 10.	This is a typical method of rating not just media, but anything, and will let users reflect on what they enjoyed or disliked.
review	This stores a text review.	This lets the user give clarification on the number rating and offers more information about the media.
review_link	This stores an optional link that leads to an external blog review or a website like Letterbox or Goodreads that the user may have posted their review on.	This lets the user organize relevant info in one place by allowing them to keep track of different websites they may have used to review a piece of media.

Use Case #5: Logging a Status
Each piece of media will have a consumption status that can be logged by the user.

Field	What it stores	Why it's needed
overall_status	This stores a general status like Planned, In Progress, Completed, Dropped, or On Hold.	This helps users keep track of all the media they want to watch, are watching, and have watched, as well as media they chose to not complete and media they plan to return to, which is handy for not forgetting their progress.
section	This stores a larger indicator of progress, such as a season of a TV show, a chapter for a book, a section/act of a video game, etc	This is helpful for giving context to the part of the media the user is on, or is an indicator that they finished it.
subsection	This stores a smaller indicator of progress, such as an episode of a TV show, a page of a book, a level of a video game, etc.	This is helpful for tracking the exact part of a piece of media that the user left off on so they can jump back in.
date_started	This stores the date when the user started consuming the media.	This is helpful for the user to know when they first began a piece of media and for tracking how long it is taking them to complete it.
date_finished	This stores the last date the user interacted with/finished the media.	This is helpful for tracking the amount of time the user has spent with the media and for tracking how long ago they interacted with the media to see if it is worth a revisit.

Use Case #6: Logging a Vendor
Media is offered by multiple vendors and these vendors can be frequented by users, and therefore are logged in the database.

Field	What it stores	Why it's needed

vendor_name	This stores the name of the vendor.	This is needed for the user to know what vendors they buyt from.
vendor_type	This stores the type of vendor, such as website, streaming service, or physical storefront.	This is useful for the user to know where they acquired a piece of media.
vendor_link	This stores a link to the vendor website, if available.	This can help the user track past visits to an online vendor and allow them to revisit it or recommend it.

Use Case #7: Vendor-Media Relationship
This connects the vendor to the piece of media and provides useful information about the purchase.

Field	What it stores	Why it's needed
vendor_name	This stores the name of the vendor and will reference the vendor table.	This lets users know where they bought or can buy a piece of media.
media_name	This stores the name of the media and will reference the media table.	This lets users know what media they bought/can buy from the vendor.
date_purchased	This stores the date of the purchase of the media from the vendor.	This can assist with tracking the consumption of media by time.
media_link	This stores a direct link to the media on the vendor site, if available/applicable	This can help the user revisit their purchase easier or recommend the purchase to a friend.
price	This stores the price of the media purchase, if applicable	This can help users budget and track their money spent on media and also play into their review of the media by determining if it was worth the price they paid for it.

Use Case #8: Series, Anthology, Collection
Some pieces of media are part of a series or anthology or collection, such as a book series or short film anthology, and users can specify which series a piece of media is part of.

Field	What it stores	Why it's needed
SAC_name	This stores the name of the series, anthology, collection or other extended body of work that a piece of media is part of.	This helps users see how their media is connected as well as track their consumption of a series or the like.
sub_SAC_name	This is optional and stores the name of a subseries/sub-extended body of work within a larger series.	This further helps users organize and search for media.
SAC_part	This is optional and stores a number to indicate which part of a chronological series the piece of media is (such as the second book in a trilogy, etc)	This helps users track completion of a connected/chronological body of media.

Structural Database Rules

Add in specialization-generalization into your structural database rules. Also enhance any existing rules as needed. Note that at least 10 entities should be identified in your structural database rules in this iteration – the 8 or more you had from the last iteration, and at least two subtypes from this iteration. Further note that your specialization-generalization rule should have a subtype and at least two subtypes.

1) Each user may log zero to many types of media, and each media type may be logged by zero to many users.
2) A media type is a book, album, TV show, movie, video game, or none of these.
3) Each book may have one review, and each review will apply to one book.
4) Each album may have one review, and each review will apply to one album.
5) Each TV show may have one review, and each review will apply to one TV show.
6) Each movie may have one review, and each review will apply to one movie.
7) Each video game may have one review, and each review will apply to one video game.
8) Each book will have one status, and each status will apply to one book.
9) Each album will have one status, and each status will apply to one album.
10) Each TV show will have one status, and each status will apply to one TV show.
11) Each movie will have one status, and each status will apply to one movie.
12) Each video game will have one status, and each status will apply to one video game.
13) Each book will have one genre, and each genre can apply to zero to many books.
14) Each album will have one genre, and each genre can apply to zero to many albums.
15) Each TV show will have one genre, and each genre can apply to zero to many TV shows.
16) Each movie will have one genre, and each genre can apply to zero to many movies.
17) Each video game will have one genre, and each genre can apply to zero to many video games.

18) Each book can be part of one series/anthologies/collections, and each series/anthology/collection can contain many books.
19) Each album can be part of one series/anthologies/collections, and each series/anthology/collection can contain many albums.
20) Each TV show can be part of one series/anthologies/collections, and each series/anthology/collection can contain many TV shows.
21) Each movie can be part of one series/anthologies/collections, and each series/anthology/collection can contain many movies.
22) Each video game can be part of one series/anthologies/collections, and each series/anthology/collection can contain many movies.
23) Each book is purchased/rented from one to many vendors, and each vendor can offer one to many books.
24) Each album is purchased/streamed from one to many vendors, and each vendor can offer one to many albums.
25) Each TV show is purchased/streamed from one to many vendors, and each vendor can offer one to many TV shows.
26) Each movie is purchased/streamed from/watched at one to many vendors, and each vendor can offer one to many movies.
27) Each video game is purchased from one to many vendors, and each vendor can offer one to many video games.

Conceptual Entity-Relationship Diagram

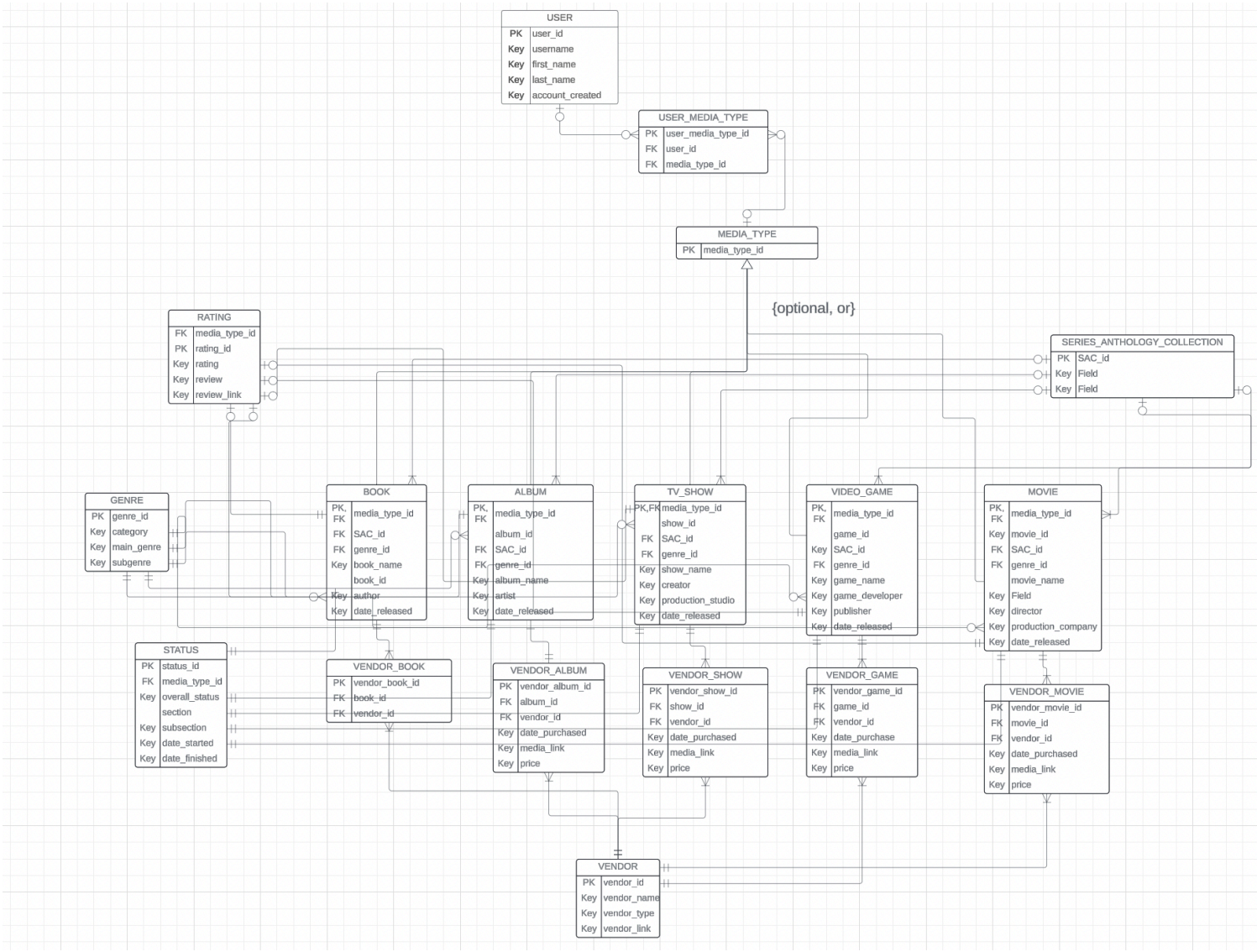
Below is my specialization-generalization in my conceptual ERD, with existing entities edited to reflect new/changed relationships from new structural database rules, and the additional use case of a series/anthology/collection added as a new entity.

This features bridging entities with a one to many relationship between the Vendor and Book, Album, TV Show, Movie, and Video Game entities, because the relationship between the Vendor entity and these media type entities is many to many.

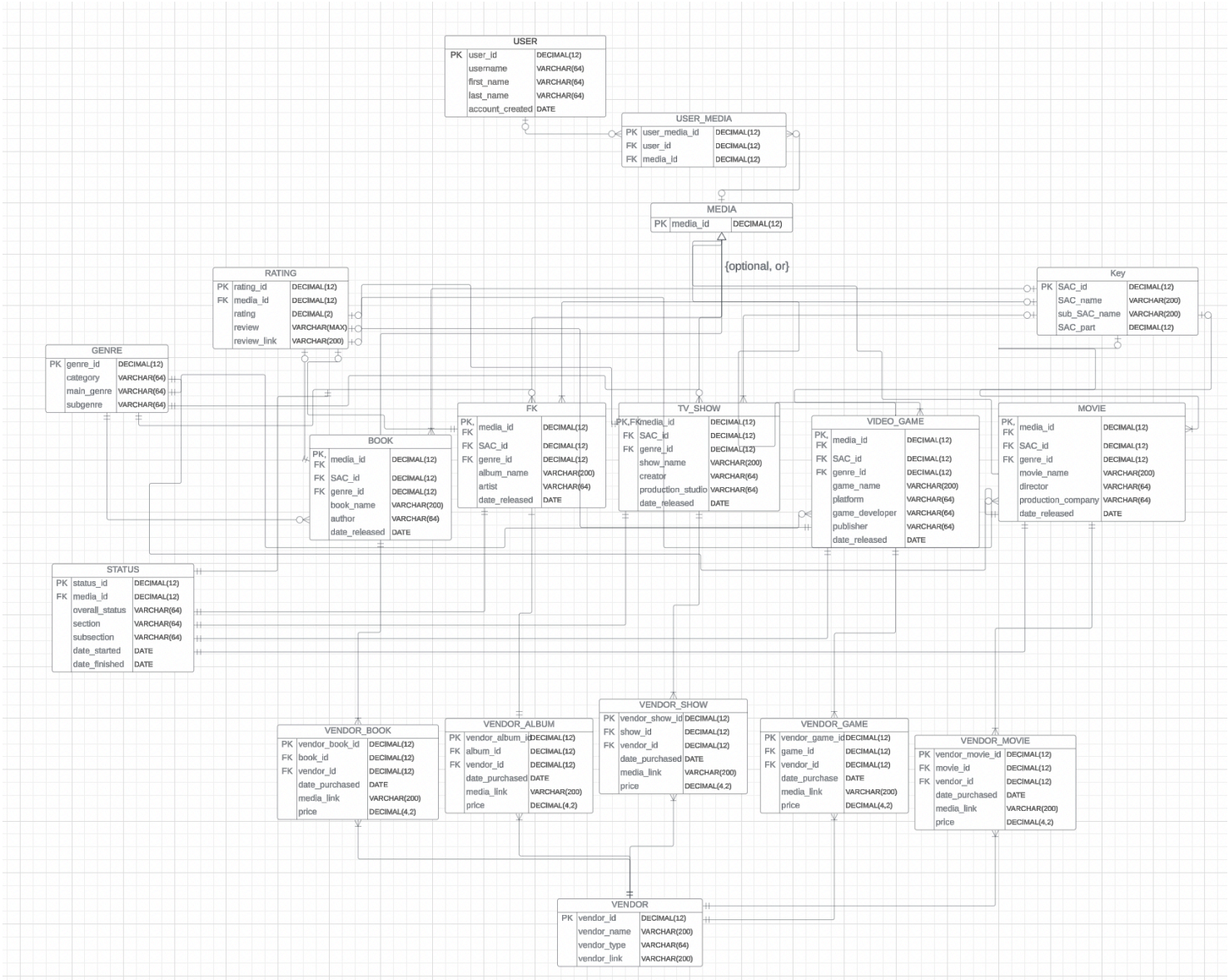
While it should seem like Genre and Status entities should have a many to many and one to many relationship with the media type entities respectively, both have a one to many and a one to one relationship respectively. It makes sense that, conceptually, a genre and a status could be similar or the same or applicable to multiple pieces of media, it is because of the amount of entry-specific information contained within both of the Genre and Status entities (such as subgenre or date_started and date_finished) that makes each entry in these tables only apply to one piece of media.

My conceptual ERD has a lot of the information that would go into a physical ERD and therefore there will not be a lot changing between the two, as all of the entities and relationships will be the same because I chose to depict all of them in my conceptual ERD.

Additionally, I had to represent my supertype-subtype relationship in UML format because I could not find the correct indicator for Crow's Foot in LucidChart.


USER
PK user_id
Key username
Key first_name
Key last_name
Key account_created
USER_MEDIA_TYPE
PK user_media_type_id
FK user_id
FK media_type_id
MEDIA_TYPE
PK media_type_id
{optional, or}
RATING
FK media_type_id
PK rating_id
Key rating
Key review
Key review_link
SERIES_ANTHOLOGY_COLLECTION
PK SAC_id
Key Field
Key Field
GENRE
PK genre_id
Key category
Key main_genre
Key subgenre
BOOK
PK, FK media_type_id
FK SAC_id
FK genre_id
Key book_name
book_id
Key author
Key date_released
ALBUM
PK, FK media_type_id
album_id
FK SAC_id
FK genre_id
Key album_name
Key artist
Key date_released
TV_SHOW
PK,FK media_type_id
show_id
FK SAC_id
FK genre_id
Key show_name
Key creator
Key production_studio
Key date_released
VIDEO_GAME
PK, FK media_type_id
game_id
Key SAC_id
FK genre_id
Key game_name
Key game_developer
Key publisher
Key date_released
MOVIE
PK, FK media_type_id
Key movie_id
FK SAC_id
FK genre_id
movie_name
Key Field
Key director
Key production_company
Key date_released
STATUS
PK status_id
FK media_type_id
Key overall_status
section
Key subsection
Key date_started
Key date_finished
VENDOR_BOOK
PK vendor_book_id
FK book_id
FK vendor_id
VENDOR_ALBUM
PK vendor_album_id
FK album_id
FK vendor_id
Key date_purchased
Key media_link
Key price
VENDOR_SHOW
PK vendor_show_id
FK show_id
FK vendor_id
Key date_purchased
Key media_link
Key price
VENDOR_GAME
PK vendor_game_id
FK game_id
FK vendor_id
Key date_purchase
Key media_link
Key price
VENDOR_MOVIE
PK vendor_movie_id
FK movie_id
FK vendor_id
Key date_purchased
Key media_link
Key price
VENDOR
PK vendor_id
Key vendor_name
Key vendor_type
Key vendor_link

Initial DBMS Physical ERD



The data types I chose for each field in each entity are pretty self explanatory based on the title of the field. For fields that would contain a smaller number of characters, I used VARCHAR(64) because that seems like common practice, while for fields that might require more characters, such as titles and links, I increased the character limit to 200, and for the review field in the Rating entity, I used VARCHAR(MAX) to accommodate a larger number of characters because I looked it up and this is best practice because the TEXT datatype is deprecated. For all of the primary and foreign keys I chose to use DECIMAL(12) because this can accommodate for a lot of entries in the database and is commonly used, and for prices I used DECIMAL(4,2) because I imagine most media will cost less than thousands of dollars. For all of the dates I used DATE, and for rating in the Rating entity I used DECIMAL(2) because the max rating is 10.

I also changed the name of the Media Type entity to Media so that the primary key name would be less confusing because leaving the name as media_type would sort of imply that there are only 5 primary keys because of the five specified media types. This also eliminates the need for a subtype-specific ID key for each of the subtypes.

As mentioned above, my initial physical ERD is quite similar to my conceptual ERD.

Summary and Reflection

My database is for a web application called “TunedIn” in which users can track their media consumption across various mediums, as well as media purchases from various vendors. The application allows users to rate their media and track their progress in consuming it.

This week I created more structural database rules and also fixed existing ones by thinking more about the proper relationship between entities. I also added a new use case and a new entity for series/anthology/collection that will be very useful to the user of my application. I decided against making the Production/Publisher entity that I contemplated last week because I realized that the dynamics of publishers and producers vary too much in their workings between TV, video games, and movies to just put into one entity that would apply to all three media types. I was also able to create a physical ERD from my conceptual ERD and I look forward to iterating upon it and improving it come next week.

I want to think about changing the way my Genre and Status entities work by perhaps changing the relationships for each of them back to many to many and one to many respectively, and this might be done by reducing the number of fields in both of those entities and placing that information into bridging entities to fix the specification issue. This way it would make more sense for each genre and status to be applicable to multiple media entries instead of just one. I also have to think about which fields in my physical ERD will be able to be null so I can specify in my next iteration and then use all of this information to start building with SQL.