How to model a Movies + TV database

[-5] [2] David542

[2021-12-31 06:06:57]

[database-design relational-database database-schema]

[https://stackoverflow.com/questions/70539480/how-to-model-a-movies-tv-database]

I have implemented a few databases that store Movies+TV data, similar conceptually to an IMDb, iTunes, Netflix, etc. There are various ways to do this, and it gets quite complex when a single title can have tons of versions, languages, promotional pricing, etc. But let's start with the basic case where items can be classified as either a Movie or TV (to the end user):

Basic types:

- TV
- Movie

The (usual) Hierarchy for TV is: TV Series (Friends) > TV Season (Friends, Season 1) > TV Episode (Friends, Season 1, Episode 7). And, perhaps we can extrapolate to having three main 'types':

- Title (A single unit that can be purchased)
- Collection (One or more units that can be purchased, yes sometimes there are single-episode seasons, or an episode + some bonus material, can also be movie bundles)
- Series (Often a helpful parent-level identifier showing where Titles or Collections may be connected)

While this seems simple in the abstract case, there are tons of edge cases, and any platform you can think of has inconsistencies, even IMDb. Some questions to illustrate this:

- How would you categorize the "2017 World Series" that contains a title for each game played? What about various purchasable Nascar races or WWE fights, should it show up in the Movie store or the TV store?
- How would you classify 15-minute animated cartoons? What if 4 of those are bundled together into a 'movie'?

If you had to store information for Movies and TV, what would the database look like? Here are a few fields to get started...

Series

- Name

Collection

- Name
- (TitleFK, Position)[]

Collection Example: "Star Wars Original Trilogy": [(A New Hope, 1), (Empire Strikes Back, 2), (Return of the Jedi, 3)]

Title

- Name
- Year

How would you structure it and why?

(2) What are you trying to accomplish? You create a data model, and when real life shows you your model is inadequate, you adjust the model. That's what IMDB did and does. - **Gilbert Le Blanc**
@GilbertLeBlanc sure, I've created and consumed several models before and current. I'm more interested in how someone else might structure things. - **David542**
(1) Fine. I'd structure things as simple as possible to get the job done and make the most profit for myself or my employer. - **Gilbert Le Blanc**

[+3] [2021-12-31 11:46:10] PerformanceDBA

Context

1.1 Anti-Relational

That is, 1960's Record Filing Systems, characterised by `RecordIDs`, which is a physical-isation of the data, placed in an SQL container for convenience (access; backups; etc), fraudulently marketed as "relational". With no Relational Power; no Relational Integrity; and no Relational Speed.

- Sure, they are quite happy "refactoring" every month, and re-modelling the "database" every time a new type of query that comes along. Not to mention the massive code re-writes.
- IMDb; NetFlix; SO; Google; etc, are all in that category. Notice the inability to detect and prevent data duplication.

I can't give you that.

1.2 Codd's Relational Model

Noting the `relational-database` and `database-design` tags. For those of us who consider the data precious, and can develop a permanent database, the structure of which doesn't change (not code re-writes), and is easy to expand (add columns and tables). I might be unable to do anything but *properly show* the notation.

High End

This subject has been done to death, since the 1970's. Think about high-end museums and international auction houses. I have done two such projects, plus one in the public domain. Elaborate, handles all those issues such as version; languages; different titles for the same movie marketed by different organisations; adaptations; etc.

- Notably, the public one was in response to an academic that wrote a paper that said "it can't be done using the *RM*, so I used RFS + Ontology + Description Logics". RFS in the back-end and massive middle-tier code. I modelled the entire requirement, about 80 tables, beyond the 12 files he had in his academic paper, proving him utterly wrong.
- If interested, you can take that as the gold standard, and chop out the bits that you do not need. Say so, and I will provide a short summary and links.

Low End

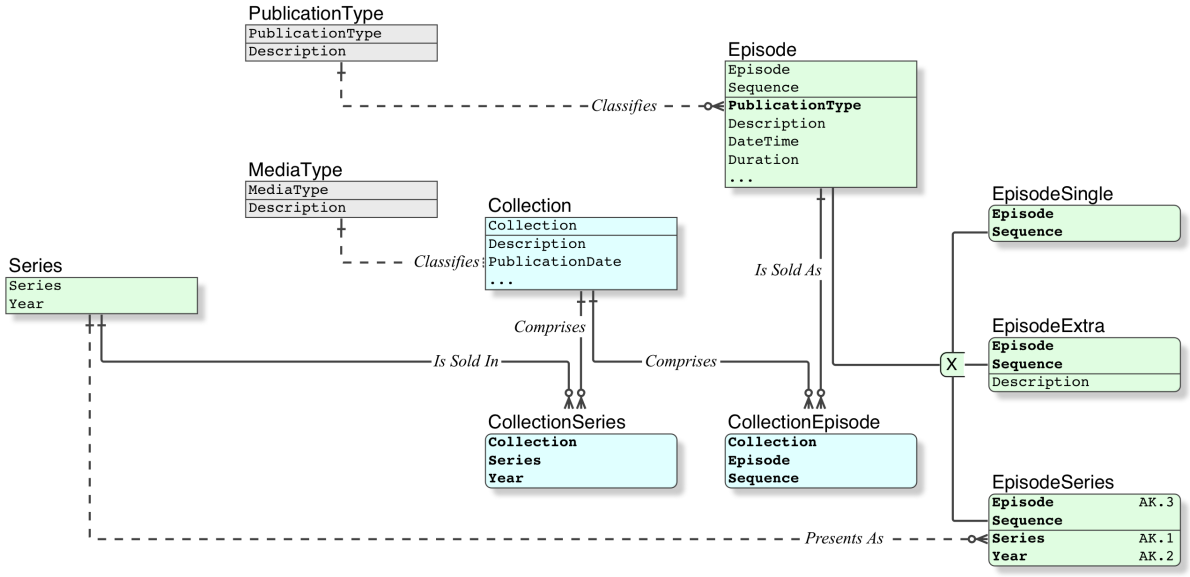
That is, to answer just your question, without the considerations that you have mentioned in your first paragraph. Rather than working from the high-end down to your requirement, the scope is only your stated requirement, working up until it is satisfactory.

- This appears to be more a data analysis question, than a modelling one. You need to post more detail, including sample data. Which may expand the scope. Understanding your data is the very first step.
- Of course, that data analysis can be substantially enhanced by using Relational Data Modelling, noting it is very early stage. The bulk of modelling, and indeed correct determination of any Fact, requires determination of the Key (the Law of Identity, of the Four Laws of Thought). Thus the model is at the Key level (not attribute; not Domain/Datatype levels).
- For data analysis or the first stages of modelling, normally I would start with simpler diagrams, eg. entity level, and proceed in increments, but that is not feasible on this site. So we got straight to the key level. Again, getting it right is very much dependent on using good sample data.

Relational Data Modelling

---

2 Data Model Vo.1



- The data model in a **PDF** [1], it has sample data.
- All my models are rendered in **IDEF1X** [2], the Standard for Relational Data Modelling.
- The **IDEF1X Introduction** [3] is essential reading for beginners.

Over to you. Happy New Year.

---

3 Comments

> I would definitely be interested in looking at your work combining RFS + Ontology + Description Logics. That puts a preliminary diagram up a notch.

In an ordinary Client-Server implementation, with a Relational database in the back-end, and a decent GUI in the front-end, there are two mutually exclusive options:

A Implement according to Science; Logic; Architecture; Standards

Complexity made simple via Science & Standards, code deployed in the appropriate location (Architecture):

- data is precious, it is owned by the corporation, it is curated
- both Rdb and code Normalised
- implement the **Open Architecture Standard** [4]
- Genuine Relational database, containing all Constraints; business rules; and ACID Transactions. The database is a single recovery unit, it is independent of all client apps
- All definitions of the data, in Logical (FOPC) any number of apps (GUIs) that access the data and execute Transactions
- the structure-of-data is completely logical, reflect reality, and thus are enduring, the code is stable. (It can be added to, without affecting extant code)
- Any cheap reporting tool can be used
- virtually no middle-tier code (only simple wrappers for _callingTransactions, or simple data caching)

B Implement according marketed info (ignore the Standards)

Complexity retained, ignorance of Science & Standards, a monolith instead of Architecture

- data is sawdust, owned by the app
- both RFS and code segments un-Normalised
- one RFS per app, commonly accessible only via the app, it is "closed"
- the RFS is dependent on the app (reversal of the natural order)
- "persistence" (for ever-changing Objects) and CRUD
- no concept of ACID Transactions
- since the RFS is physical, not logical, an Ontology to define the non-logical data
- and a Description Logics to obtain meaning from the non-logical data structures
- report tools cannot be used directly because there is no logic defining the data. Thus at the high-end (BusinessObjects) an "universe" that logically defines the data must be built, at the low-end, an Ontology plus a Description Logics
- the structure-of-data is transient: every time there is a new requirement, structure changes, and extant code has to be re-written
- every time the app changes, the RFS has to be "refactored", and the Ontology, and the Description Logics with it
- a massive middle-tier bunch of code, which also has to change every time.

Therefore it is not "combining RFS + Ontology + Description Logics" (which is a great idea while sitting squarely in [B]), but implementing the whole Relational database + app according to Standard, such the an Ontology + Description Logics is not required; eliminated. Like "refactoring" and CRUD are not required, like syphilis is not required.

If you have further questions re the Standards-based implementation, please ask a new Question, so that it can be answered properly.

> imagine we have millions of titles from across the world (think IMDb or iTunes)

Table population is irrelevant. It is 2022, we have indices covering the data (sure, the freeware and Oracle are suh-low overall when compared with the genuine SQL Platforms). The other mythology, that “joins are expensive” is likewise false: joins cost nothing, it is the size of the result set that is relevant.

Please appreciate that I have modelled the stated requirement. Data Modelling is an iterative process: each iteration implements the (thus far) stated requirement, and exposes new issues that need to be resolved ... in the next iteration. This exercise is typical.

So now you have noticed the exposed issues, and you have additional requirements. No problem at

all. Inform me as to how you would resolve that in the real world, and I will produce the next iteration of the data model. The PDF has sample data, which I have carefully constructed, feel free to expand that or add your own. As stated from the outset:

- this is more of a data analysis task
- the bulk of data modelling regards the determination of Identifiers (Keys)

> (1) how would you uniquely identify the Series? Series+Year would obviously be insufficient (type in any common word in IMDb and you should see hundreds or even thousands of matches).

What precisely identifies a `Series` uniquely ?

On the one hand, it is correct to obtain the thousands of matches from the Rdb as is (V0.1), because that is true. But then we must nominate attributes such that those thousands are reduced to one. What are those attributes ?

> (2) Even more so with Episode, Episode+Sequence would be insufficient, actually I've seen collisions on much longer combinations than this in the real world.

What precisely identifies a `Episode` uniquely ?

I knew `Episode` would be duplicated (refer to the PDF), so I gave you the simplest of methods to make it unique: `Sequence`. Inform me as to what attributes make `Episode` unique, and jettison `Sequence`.

> (3) Sequence is not immutable, episodes may be in one position in one container and another position in another one.

`Sequence` [V0.1]

The only purpose was to make `Episode` unique. It is slated for replacement [1][2]. It is not a `Position` or `Track`, which should be in the subordinates to `Collection{Series|Episode}`, which I had not modelled. No point in determining the child Key until the parent Key is stable.

> (4) For a Movie the Sequence is always 1 or rather null as it's not really applicable.

Yes and no. Two responses, at separate levels, both of which need consideration for the next iteration.

- At the level of your comment, wherein we have the one `[Episode]Title` for both Movies + TV, as per [V0.1]
  - Yes, and we do not store nulls (gross Normalisation error), so `Sequence` is an optional attribute
  - even in the current data model [V0.1], it can be appreciated that `Title` has to have more attributes to make it unique
- At the next level, the next data model [V0.2] below
  - you should perceive that `MovieTitle` and `EpisodeTitle` are quite different, the determinants (attributes that make each unique) are different
  - it may be Title has to be determined independent of both Movie and TV,
    xor
    `MovieTitle` and `EpisodeTitle` are hierarchs for `Movie` and `Episode`, respectively. Pending clarification from you, I would tend towards the latter.

> A somewhat similar model for Movies+TV data is how iTunes does it at

resources.organicfruitapps.com/documentation/... (diagrams are a bit down the page).

What a mess, how un-Apple that is. Made even more messy due to it being a pathetic RFS, not an RDb. That is what happens when a schizophrenic takes over from Steve Jobs, and slowly destroys the company from within. So the evidenced fact is, they do not *do* it, they *mess* with it, and the mess keeps changing, and they impose those constant changes on the external Partner developers.

Second, they make the classic, hysterical mistake that academics have made for fifty years: they fail to differentiate between base relations (stored) and views, which are result sets (projections; not stored). (Their illogic: because a relation is an abstraction and there is only one kind of abstraction.) This is one instance of how the insane impose their insanity on the sane.

Third, since it is not a formal data model but a picture, their relations are lost, so they have a *second diagram* to inform us that the relations are too messy (spaghetti) to draw properly. Good for making the hair curl. You can't make this stuff up.

Each clause above is a classic hallmark of insanity.

I will take it only as attributes that should be considered, not as attributes that belong to entities as declared.

3.1 External Reference

If you are using external sources as reference, then it is high time that you start using worthy or Standard ones, instead of rotten Apples. Such as the high-end, Movie Title data model I mentioned above, for an European Movie Museum, for the various types of movies that are catalogued.

- **Movie Title Entity** [5]
  About 80 tables on a single A2 page. This is the Contextual Index.

- **Movie Title Data Model** [6]
  Attribute level, on a single A1 page.

- **Movie_Title_Data_Model_A3** [7] All levels, in a single A3 PDF for printing, or for online navigation. It is a bit squeezed, to fit his A3 requirement.

  - I usually include a Data Hierarchy and an Alphabetic Index.

  - Note that each collapsed symbol is clickable: it will go to the expansion. Each page is a SubjectArea, a logical subset of the database.

  - (This partially answers your other question re documenting a database.)

- **Movie Title Discussion** [8]
  The entire discussion, from whoa to go, which is really a Relational Modelling exercise, between the user identifying requirements incrementally, and me as modeller. In this case the user was an academic, who like you, thought it couldn't be done under the *Relational Model*, and insisted it could be done with an RFS + Ontology + Description Logics, despite having no implementation experience whatsoever.

  - Warning: academics love to split hairs and leave things unresolved (ambiguous), whereas I apply the Four Laws and resolve every little thing, so be prepared for academic argumentation. About 80 tables in 14 iterations, via 98 posts.

  - I am not recommending that you read this discourse, I am providing it for completeness.

I am not saying you should start at the high-end and strip out what you don't need, although that is an option. I am saying, if you use a reference, use a good one.

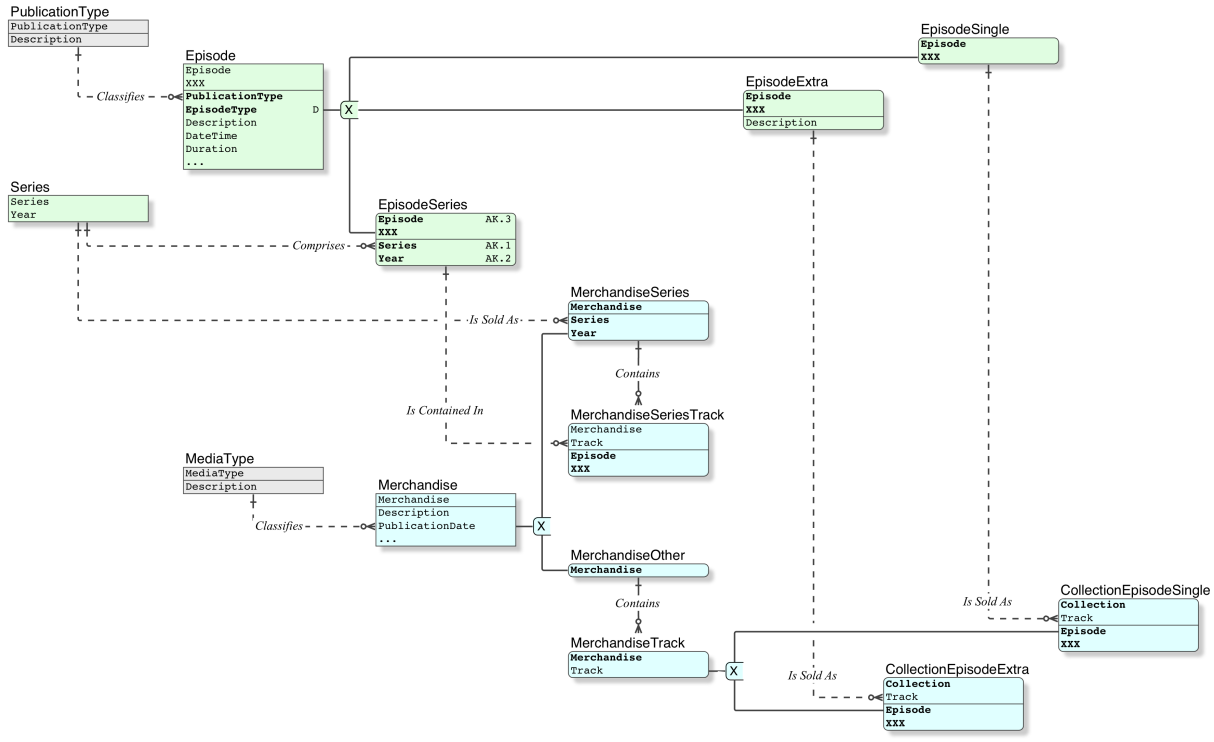
> The difficult part of "breaking everything up"

“Breaking everything up” is (a) Normalisation, that is demanded, and (b) ordinary data modelling (both pre- and post-Relationalo. Deal with it. Strictly, we need to break up only the Key components, but you do not know the data well enough to do that directly, so we must produce a few iterations, in order to expose the issues, which assists in determination of the Key components, and discrete Facts. Conversely if you avoid breaking the mess down to its atomic components, you end up with a grid that is chock-full of nulls, which leads to complex code, a well-known and guaranteed maintenance nightmare.

> is when you then have to tie in Prices, Genres, Cast/Crew, etc to different entities -- especially for entities where it's debatable where they should go (such as some of the examples from the question).

Actually, no. The opposite is true. Once the central Facts in the subject area are modelled and stable, it is quite easy to “tie in Prices, Genres, Cast/Crew, etc to different entities”, because the entity to which they apply is clear. The debate remains only while those central Facts are not determined correctly. Watch and see.

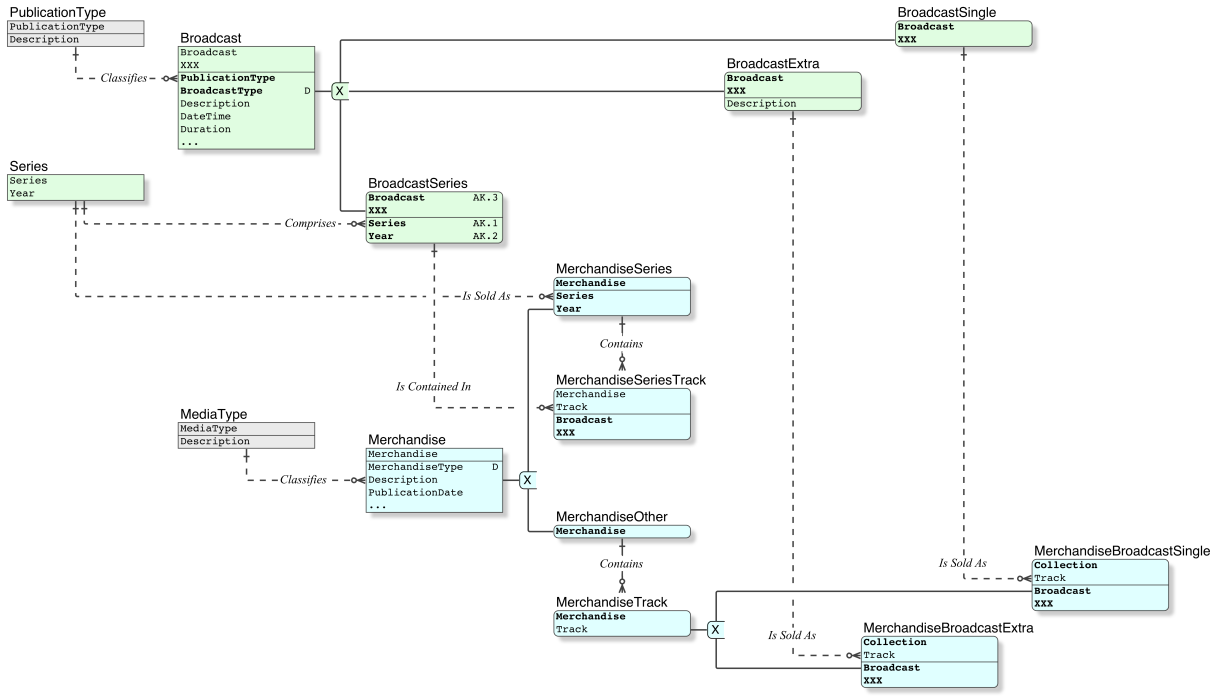
4 Data Model V0.2



- The data model in a **PDF** [9], it includes sample data that must be analysed carefully, in order to validate it.
- I have exposed the `Track` issue so that you can engage it.

5 Data Model V0.3

- I found a couple of minor errors, corrected. No substantial change.
- Naming is very important because it conveys meaning and identity. I have improved the tables names.



- The data model in a **PDF** [10], it includes sample data that must be analysed carefully, in order to validate the model.

Over to you.

[1] https://www.softwaregems.com.au/Documents/Student_Resolutions_2020/David542/Movie%20Collection%201.pdf
[2] https://www.iso.org/standard/60614.html
[3] https://www.softwaregems.com.au/Documents/Documentary%20Examples/IDEF1X%20Introduction.pdf
[4] https://www.softwaregems.com.au/Documents/Article/Application%20Architecture/Open%20Architecture.pdf
[5] http://www.softwaregems.com.au/Documents/Article/Database/Movie%20Title/Movie%20Title%20TR%20V0_14.pdf
[6] http://www.softwaregems.com.au/Documents/Article/Database/Movie%20Title/Movie%20Title%20TA%20V0_14.pdf
[7] http://www.softwaregems.com.au/Documents/Article/Database/Movie%20Title/Movie%20Title%20TA%20V0_14%20A3.pdf
[8] https://groups.google.com/g/comp.databases.theory/c/2GHCadeG5sA
[9] https://www.softwaregems.com.au/Documents/Student_Resolutions_2020/David542/Movie%20Collection%202.pdf
[10] https://www.softwaregems.com.au/Documents/Student_Resolutions_2020/David542/Movie%20Collection%203.pdf

I would definitely be interested in looking at your work combining RFS + Ontology + Description Logics. That puts a preliminary diagram up a notch. Please if you do not mind to inform when posting a link when your work is available. Thank you. - **Mugé**
Thanks for putting in the time! A few comments though: imagine we have millions of titles from across the world (think IMDb or iTunes), (1) how would you uniquely identify the Series? `Series+Year` would obviously be insufficient (type in any common word in IMDb and you should see hundreds or even thousands of matches). (2) Even more so with Episode, Episode+Sequence would be insufficient, actually I've seen collisions on much longer combinations than this in the real world. (3) Sequence is not immutable, episodes may be in one position in one container and another position in another one. - **David542**
(4) For a Movie the Sequence is always 1 or rather null as it's not really applicable. A somewhat similar model for Movies+TV data is how iTunes does it at resources.organicfruitapps.com/documentation/... (diagrams are a bit down

the page). The difficult part of "breaking everything up" is when you then have to tie in Prices, Genres, Cast/Crew, etc to different entities -- especially for entities where it's debatable where they should go (such as some of the examples from the question). - **David542**
@Mugé I have added a section to the Answer, in response to your comment. - **PerformanceDBA**
@David542 You are most welcome. I have added a section to the Answer, in response to your comments. - **PerformanceDBA**
@David542 I have upgraded the data model. Please review, and answer the outstanding questions. - **PerformanceDBA**

1

[+1] [2021-12-31 07:09:44] Mugé

Since no one answered until now and putting my hat on from own experience, you are right, it looks simple, but the data can grow tremendously and complex. Yet, although it seems confusing at first, taking the long haul is most times the better solution in the long run.

Meaning, first you need to pick the data apart. I call it categories. Categories can never be long wound descriptions. Every category has a table.

First to answer some of your questions, you mentioned language at first. Language on the hierachy is one of the highest located, higher than the movie name/title.

Just to go over what you already mention by first categorizing: TV > before you even go to the series, it first needs to be categorized by genre (drama, comedy, musical, sports, reality, podcast, etc) > country of origin (this can be inter-changeable on the hierarchy maybe even before genre) > and all the rest like Series/Title/Year/Director/Cast/Description/Rating/Price/Images/Poster/Video, and so on.
(All these are separate tables)

Movie: not going into too much detail here because the above will give some insight.

Position: do you mean, where the product is located? (I am unclear on this one)

"2017 World Series" and Nascar races or WWE fights --> They are reality (not categorizing as TV or Movie because they are a category on their own, like documentary.

Think this way, books are categorized into fiction and non-fiction.

Cartoons have their own category, on the same level as TV, Movie, Documentary, Reality Events, etc.

15-minute animated cartoons: Short films, short cartoons.

If they have a sequence, it should have a unique identifier that combines them automatically, so that they show up together when running a query.

I hope this gives some starting point to organize things mentally first and then it will all come together.

Position I mean the container position or track number. Episode 1 might be "Position 1 in season 1". No, the limitation is it needs to be sold as either the Movies or TV Store, like Google Play has it: gyazo.com/540eaf2e06ae2ade9d78442632cce53a - **David542**
I think Prime just bunches it all into 'Latest Releases' or something, on the very first row, and it is all mixed which I like. I think calling it 'Movie Store' makes it more interesting than a 'TV Store'. I am sorry that I am still not clear about the track number because that will automatically in ascending order when all categories are called into a query, they will have to be lined up. The rest is design work. - **Mugé**
that's wrong. Most stores categorize content as movie or tv, including Amazon: gyazo.com/9fc89c93f51d3b016824f9672b0a70d5 - **David542**

I agree, it just gives the idea what it is about. It's practical to call it that way. When Netflix started it was about Movies, not TV series. Yes, go by the conventional way. - **Mugé**
As the end effect, did I answer your question? - **Mugé**
hey Mugé, I appreciate your time putting this all together! No unfortunately not helpful for what I was looking to learn though. - **David542**

[2]
