Vibeka McGyver

Movie Collection Database

INFSCI 1022 Database Management Systems

12/12/18

Table of Contents

[Introduction 3](#_Toc531810630)

[E-R Model 4](#_Toc531810631)

[Business Rules 4](#_Toc531810632)

[Entity and Attribute Descriptions 6](#_Toc531810633)

[List of Questions 8](#_Toc531810634)

[Closing 9](#_Toc531810635)

Introduction

The database I created, titled movie\_collection, contains data for tracking and maintaining movie collections. This database is meant to be used by someone who has a personal movie collection they want to keep track of. I mainly collect VHS tapes, and I struggle with remembering what movies I own. Buying VHS tapes is so cheap that I often end up with too many copies of the same movie. This database will allow myself and other movie collectors a convenient way to keep track of what they already own, which will help build a more complete and high-quality collection.

In movie collecting two important factors are the object’s quality and its video format, because a complete collection contains one movie on many formats with excellent object quality. This database gives the user the ability to log multiple copies of the same film on different video formats and quality levels. For my personal collection, I like to own my favorite movies on the three most important formats: VHS, Blu-ray and LaserDisk. Each format has a unique viewing experience that another format can’t duplicate. Blu-ray offers a fine crisp HD view, but VHS shows a nostalgic muffled and original feeling view. Similarly, LaserDisks are a combination of a Blu-ray and VHS, but the experience of putting the record-sized disk in the machine is an experience of itself. One of the risks in collecting is that it’s likely to buy a VHS for $0.99, but later realize the tape is damaged, so replacing a poor-quality copy with a brand-new quality item is an important part of the process. Therefore, tracking formats and quality is an important aspect of my database.

Another important aspect of movie collecting is to log purchase-price. All collectors wonder what their collection is worth, and this function of the database makes keeping track easy. My personal collection mainly focuses on VHS tapes and LaserDisks, which I mostly source from thrift stores. One issue I have found is that I can’t remember where I find my best pieces and get the best deals. A function I added to my database is the ability to connect purchase location to an item. Therefore, I can see which thrift stores I am mostly getting ‘misses’, being sold over-priced items, and avoid those places. For any movie collector, this would be valuable and timesaving data. Along with purchase-price is the option to log a current resale value price. Although VHS tapes have very low resale value, it is still an interesting way to examine the value of the collection. For someone who collects Blu-ray box sets, this tool may be incredibly valuable if they have cheap box-set sourcing methods.

Movie collecting is fun, endlessly entertaining and can even have high resale value, if that is the collector’s intent. I have a collection of over 100 VHS tapes, which I started in high school before I had a DVD player or Netflix. Over time I have joined VHS collecting communities online and noticed a problem where collectors kept rebuying the same items and oversaturating their collection. The purpose of this movie collection database is to provide a tool for movie collectors who want to keep track of the physical copies of movies they own. Creating a simple, mobile interface for it would turn this database into a seriously useful product. I would make this a mobile application with easy search tools, to check and see if I already own a movie before buying it again for $0.99. The database I built would be very beneficial to movie collectors who want to condense, expand, or complete their collections.

# 

# E-R Model

# 

# 

# Business Rules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity 1** | **Entity 2** | **Cardinality on Entity 1 side** | **Cardinality on Entity 2 side** | **Business Rule(s)** |
| Film | Genre | 1..\*  (One film has one to many genres) | 1..\*  (One genre has one to many films) | A film must have one or many genres.  A genre must be associated with at least one film. |
| Film | Collection Item | 0..\* | 1..1 | A film can be related to one to many collection items. A collection item must have only one film associated with it. |
| Purchase Location | Collection Item | 0..\* | 0..1 | A purchase location can have zero to many collection items associated with it. A collection item may only have one purchase location but can have zero to account for possibilities such as a gift. |
| Current Resale | Collection Item | 1..1 | 0..\* | A current resale value must have one item associated with it. One collection item can have zero to many resale values. |
| Video Format | Collection Item | 0..\* | 1..1 | A video format can have zero to many collection items. One collection item only has one video format. |
| Video Condition | Collection Item | 0..\* | 1..1 | A video condition can have zero to many collection items. One collection item only has one video condition. |

# 

# Entity and Attribute Descriptions

|  |  |  |
| --- | --- | --- |
| **collection\_item**: enter information regarding a single item in the collection | | |
| pk\_item\_id | int(11) AI PK | An autoincrementing primary key for the collection item. |
| fk\_condition\_id | int(11) | Use a condition ID from the video\_condition table to track the object's condition. |
| fk\_format\_id | int(11) | Use a format ID from the video\_format table to track the object's format. |
| fk\_film\_id | int(11) | Use a film ID from the film table to track the object's film. |
| purchase\_year | year(4) | Year the item was purchased, if item was a gift than no need to enter purchase\_year. |
| purchase\_price | double | Price the item was purchase for, if not purchased than there is no need to enter a purchase\_price. |
| fk\_purchase\_location | int(11) | Use a purchase\_location\_id from the purchase\_location table to track the object's purchase location if necessary. |
| notes | varchar(500) | Additional notes regarding the item can be entered here. |

|  |  |  |
| --- | --- | --- |
| **current\_resale** | | |
| pk\_current\_resale\_id | int(11) AI PK | An autoincrementing primary key for the current\_resale. |
| fk\_item\_id | int(11) | A foreign key from the collection\_item table linking a new resale value to the item. |
| current\_value | double | The current resale value for the item. This price should be sourced from eBay under completed and sold items for an exactly similar item. |
| update\_time | timestamp | Automatically update the time for this current value entry due to the face that prices change over time. |
| **film:** a separate table for films allows simplicity in entering multiple copies of the same film | | |
| pk\_film\_id | int(11) AI PK | An autoincrementing primary key for the film. |
| film\_title | varchar(150) | Film's title. |
| release\_date | date | The film's release date. |
| director\_name | varchar(50) | The film's director's first and last name entered as one. |

|  |  |  |
| --- | --- | --- |
| **film\_genre:** film and genre junction table | | |
| fk\_film\_if | int(11) | The film\_id from the associated film. |
| fk\_genre\_id | int(11) | The genre\_id from the associate genre |

|  |  |  |
| --- | --- | --- |
| **genre** | | |
| pk\_genre\_id | int(11) AI PK | An autoincrementing primary key for the genre. |
| genre\_name | varchar(50) | Genre title, such as 'Action'. |

|  |  |  |
| --- | --- | --- |
| **purchase\_location** | | |
| pk\_location\_id | int(11) AI PK | An autoincrementing primary key for the location. |
| store\_name | varchar(50) | The store's name. |
| address\_line1 | varchar(50) | Location's address line 1. |
| address\_line2 | varchar(50) | Location's address line 2. |
| city | varchar(50) | Location's address city. |
| state | varchar(50) | Location's address state. |
| postal\_code | varchar(50) | Location's address postal code. |

|  |  |  |
| --- | --- | --- |
| **video\_condition** | | |
| pk\_condition\_id | int(11) AI PK | An autoincrementing primary key for the condition type. |
| condition\_title | varchar(50) | A title for the condition type. Optional titles, taken from eBay's classification system are: "Brand New", "Like New", "Very Good", "Good", and "Acceptable". |
| condition\_description | varchar(500) | The description for each condition type. An object must comply with the condition\_description to be considered that condition\_type. |

|  |  |  |
| --- | --- | --- |
| **video\_format** | | |
| pk\_format\_id | int(11) AI PK | An autoincrementing primary key for the format type. |
| format\_name | varchar(50) | The associated format type's name. |

# 

# List of Questions

1. What items do I own? This is the most important question to answer because it provides a list of all items in the collection. This query will answer that question and provide the results in an easy to read, alphabetized order as well as display the most important information about the item including format and condition.

select item\_id as 'item id', f.film\_title as 'film title', vf.format\_name as 'format',

vc.condition\_title as 'condition', notes from collection\_item ci

join film f on f.film\_id = ci.film\_id

join video\_format vf on vf.format\_id = ci.format\_id

join video\_condition vc on vc.condition\_id = ci.format\_id

order by f.film\_title asc;

1. Sometimes you want to watch a movie but you don't know what movie to decide on. This query allows the user to search within genres for a movie that might appeal. Running this query for example will produce a list of movies that fall under the genre of either science fiction or anime, and excludes any duplicates that come from movies having multiple genres.

select f.film\_title from film\_genre fg

join genre g on g.genre\_id = fg.genre\_id

join film f on f.film\_id = fg.film\_id

where genre\_name like '%science fiction%' or genre\_name like '%anime%'

group by film\_title;

1. The most useful reason to use a single table select would be to view the optional condition titles and condition descriptions. Using this select's results is necessary when entering new collection items because it is important to maintain consistent item conditions.

select \* from video\_condition;

1. In order to see which items in my collection have cost more money than they are currently worth, this query would be used. It is ordered in descending order. This query also shows details of the items that lost money, which can show valuable trends. Running this query shows 3 out of the 4 items I've lost money on have been bought at Half Price Books, so maybe it's a good idea to be careful what I purchase from them.

select cr.item\_id as 'Item ID', purchase\_price as 'Purchase', current\_value as 'Current', vf.format\_name as 'Format', f.film\_title as 'Title', pl.store\_name, cr.update\_time as 'Current Value Update Time' from current\_resale cr

join collection\_item ci on cr.item\_id = ci.item\_id

join film f on f.film\_id = ci. film\_id

join video\_format vf on ci.format\_id = vf.format\_id

join purchase\_location pl on ci.purchase\_location = pl.location\_id

where purchase\_price > current\_value

order by purchase\_price desc;

1. Knowing the value of a collection is interesting and important. The first query returns the total amount of money spent, and the second query shows the total current resale value based on the values inputed.

select sum(purchase\_price) as 'Total Spent' from collection\_item;

select sum(current\_value) as 'Current Value' from current\_resale;

1. Knowing how many items there are in the collection is another important question I want to answer. Using a count function, we can get a grand total of items. Using another count function and a group by we can get a count of items by their format.

select count(item\_id) from collection\_item;

select count(item\_id), video\_format.format\_name from collection\_item

left join video\_format on video\_format.format\_id = collection\_item.format\_id

group by video\_format.format\_name

order by count(item\_id);

1. I want to know which items are the most expensive that I bought, and how much I spent. The best way to do this is to order my items by purchase price. I don't want to see all of the items and their prices, so I will limit the search results to five, to get an idea of the top five most expensive things in my collection.

select purchase\_price as 'purchase price', f.film\_title as 'film title', vf.format\_name as 'format',

vc.condition\_title as 'condition', notes from collection\_item ci

join film f on f.film\_id = ci.film\_id

join video\_format vf on vf.format\_id = ci.format\_id

join video\_condition vc on vc.condition\_id = ci.format\_id

order by purchase\_price

desc

limit 5;

1. Today I want to go shopping for new items but I don't want to waste a lot of time at stores that are too expensive or usually have bad pieces. I would use this query to answer my question of where I should go today.

select \* from collection\_item

join film on film.film\_id = collection\_item.film\_id

inner join purchase\_location on purchase\_location.location\_id = collection\_item.purchase\_location

where purchase\_location IN

(select location\_id from purchase\_location where city like '%pittsburgh%') and purchase\_price <= 2.99 and condition\_id >= 3;

1. Due to having the same movie on different formats, I have many copies of the same movie. But how many copies is too many? Can I spare to sell one or trade it? This query will answer that question for me.

select film.film\_title, count(\*) as itemcount

from collection\_item ci

join film on film.film\_id = ci.film\_id

group by ci.film\_id

having itemcount >= 2

order by itemcount desc;

1. This query would tell me what the most popular genres in my collection are. The results of this query are valuable because it can tell me if I need to expand in certain areas or if my collection is consistent.

select fg.genre\_id, genre\_name, count(\*) from collection\_item ci

join film\_genre fg on fg.film\_id = ci.film\_id

join genre on genre.genre\_id = fg.genre\_id

group by genre\_id;

# Closing

I enjoyed using this project to create a personally meaningful database, which solidified the convenience and usefulness of databases in everyday life. I also enjoying using this project to see what it is like to completely design a database from the ground up. I had the most challenges with the initial database design, and it took a lot of trial and error to get it how I wanted it to be. One thing I would like to do is to see what it is like to design a front end for a database in order to make it useful for a casual user. I would also like to spend more time refining my database skills. From this project I have more of an interest in how databases work, and the process of organizing information.