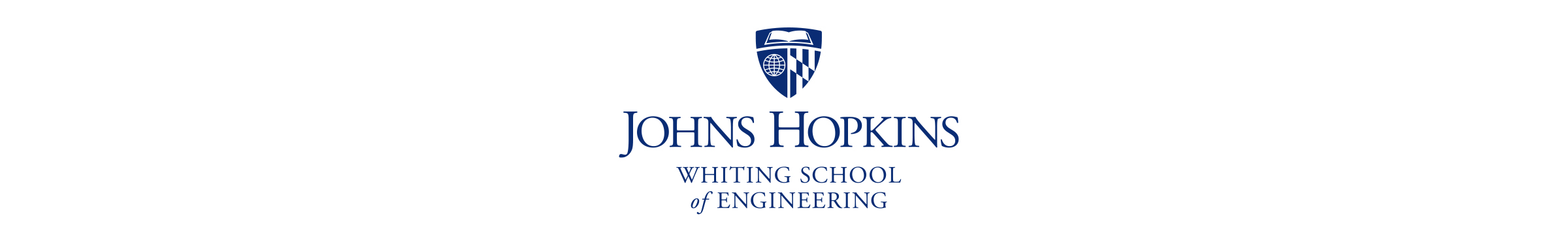
|  |
| --- |
|  |
| MovieDbLite |
| Course Section: CS605.641  Spring, 2020 |
| Prepared by |
| **Anderson, Steven** |
| **04/22/2020** |



|  |
| --- |
| Database Design Project Document |

**Table of Contents**

[1. Introduction 4](#_Toc516748910)

[1.1. Scope and Purpose of Document 4](#_Toc516748911)

[1.2. Project Objective 4](#_Toc516748912)

[2. System Requirements 4](#_Toc516748913)

[2.1 Hardware Requirements 4](#_Toc516748914)

[2.2 Software Requirements 4](#_Toc516748915)

[2.3 Functional Requirements 4](#_Toc516748916)

[2.4 Database Requirements 4](#_Toc516748917)

[3. Database Design Description 4](#_Toc516748918)

[3.1 Design Rationale 4](#_Toc516748919)

[3.2 E/R Model 4](#_Toc516748920)

[3.2.1 Entities 5](#_Toc516748921)

[3.2.2 Relationships 5](#_Toc516748922)

[3.2.3 E/R Diagram 5](#_Toc516748923)

[3.3 Relational Model 5](#_Toc516748924)

[3.3.1 Data Dictionary 5](#_Toc516748925)

[3.3.2 Integrity Rules 5](#_Toc516748926)

[3.3.3 Operational Rules 5](#_Toc516748927)

[3.3.4 Operations 5](#_Toc516748928)

[3.4 Security 5](#_Toc516748929)

[3.5 Database Backup and Recovery 5](#_Toc516748930)

[3.6 Using Database Design or CASE Tool 6](#_Toc516748931)

[3.7 Other Possible E/R Relationships 6](#_Toc516748932)

[4. Implementation Description 6](#_Toc516748933)

[4.1 Data Dictionary 6](#_Toc516748934)

[4.2 Advanced Features 6](#_Toc516748935)

[4.3 Queries 6](#_Toc516748936)

[4.3.1 Customer Bills 6](#_Toc516748937)

[4.3.2 Customer Rental History 6](#_Toc516748938)

[4.3.3 Movie Rental History 6](#_Toc516748939)

[4.3.4 List all videos by movie category 6](#_Toc516748940)

[4.3.5 List video usage by movie category 6](#_Toc516748941)

[4.3.6 List videos by format (Laser Disc or VHS) 6](#_Toc516748942)

[4.3.7 List defective videos 7](#_Toc516748943)

[4.3.8 List twenty popular videos by category for customers’ recommendations 7](#_Toc516748944)

[5. CRUD Matrix 7](#_Toc516748945)

[5.1 List of Entity Types 7](#_Toc516748946)

[5.2 List of Functions 7](#_Toc516748947)

[6. Concluding Remarks 7](#_Toc516748948)

[Appendices 8](#_Toc516748949)

[References 8](#_Toc516748950)

(Note: Please provide some inputs for the sections in your DB design document template even though some of them may not be suitable for your project. The purpose for all different sections is to ask you to think about different perspectives of a database project. If a section doesn't apply to your project, you should make something up. Please don't skip them.

For instance, Database Backup and Recovery is commonly implemented in a real database project. Even if you don’t implement it, most RDBMSs may have incorporated backup and recovery tools into their interfaces and infrastructure. Give a short description of your RDBMS solution to demonstrate your thought process.)

# Introduction

Movies are a common form of entertainment in the lives of many people. With the evolution of the internet and various streaming services, the access to films are more available than ever. The MovieDbLite project aims to be simple, easy-to-use website for viewing the details about movies and credits, as well as allowing users to rate and review the movies they have seen. In alignment with the website, the database itself is intended to be simple and intuitive for people with database experience.

For myself personally, I am a huge movie buff, and still go to the theaters multiple times a month with my wife to watch new releases. I chose this project because of my interest in movies and the people involved in delivering these films. In addition, the ownership of a movie database allows for me to personalize the reports or features that may not be found in other popular movie database applications. Furthermore, it opens up the possibility to create APIs (Application Programming Interfaces) for consumption of the data by other systems, use cases, and applications (e.g. trivia games).

## Scope and Purpose of Document

The purpose of this document is to describe design, development, requirement, and implementation details of the MovieDbLite database and website application, with the primary focus being on the database that is used to store movie and film industry data. Specifically, when it comes to the design of the database, the focus is on the conceptual and logical design and implementation. An existing, industry-standard Relational Database Management System (RDBMS) environment that handles the physical implementation of the database will be used. Therefore, lower-level details about physical storage and implementation of the database can be found via online resources of the respective RDBMS environment.

Overall, this document is bounded to describing the MovieDbLite database and front-end website for accessing basic details and reports of data within database. Accordingly, information about external applications that may consume the database data (e.g. through an API) is not intended to be covered by this document.

## Project Objective

The objective of this project is to design and implement a simple, yet useful movie database that tracks information related to the film industry. Further, the project intends to provide a front-end website to deliver an easy-to-use user interface for allowing users to view data and rate movies. This project shall be built using modern RDBMS and web-based technologies to captures the data elements and functional requirements that are mentioned in the upcoming sections. Ultimately, this database is intended to serve as a stable and maintainable back-end to enable future opportunities for APIs to be built for external applications to consume.

# System Requirements

The MovieDbLite is composed of three high-level system components that have their distinct set of system requirements:

1. The SQL Server hosted MovieDbLite Database
   1. Hosted on a remote server
2. The MovieDbLite Web Application Environment (Back-End)
   1. Hosted on a remote server
3. The MovieDbLite Website (Front-End)
   1. Accessed by an end-user via Web Browser

This section will cover the System Requirements (software and hardware) for the three major system components of the MovieDbLite system, as well as the functional requirements that make apart building these three components.

Although the Database (system component #1) and Web Application (system component #2) are hosted remotely, this does not necessarily mean they will be hosted on the same remote server/machine. In other words, the database and web application may be hosted on different server environments. Therefore, the following sections may distinguish and discretely separate the requirements for these respective components as appropriate.

## Hardware Requirements

This section describes the hardware requirements for the three components of the MovieDbLite system. In some of the hardware listings, a minimum hardware requirement is listed, as well as a recommended. The minimum hardware requirement is meant to indicate the minimum possible hardware specification to run or host the component. The recommended listing is the suggested minimum to mitigate deterioration of component performance. Overall, the system will support hardware specifications with larger amounts than the minimum/recommended listings, which would subsequently increase or maintain component performance and/or reliability.

### Database Server Requirements

The database server will require hardware able to host SQL Server Standard Edition 2017. The full server hardware requirements can be found in Microsoft’s documentation [here](https://docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15#hardware-requirements).

This is the overall breakdown of the hardware requirements for SQL Server Standard Edition:

* **Memory** - Minimum of 1GB, recommended of 4GB
* **Hard Drive**
  + Computer with NTFS or ReFS file format for security reasons.
    - TODO: Provide links for NTFS and ReFS
  + Minimum of 6GB of available hard-disk space
    - Note: [Hard Disk Space Requirements](https://docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15#HardDiskSpace) will apply depending on which components are installed
* **Monitor -** Super-VGA (800x600) or higher resolution monitor
* **Internet -** Internet access is required
* **Processor**
  + Speed: Minimum of 1.4GHz, recommended of 2.0+ GHz
  + Type: x64 Processor (e.g. AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support)
* **Drive** – A DVD drive is required if installation is from disc

### Web Application Server Requirements

The Web Application environment (back-end to the website) will require an environment that can host an ASP.NET Website targeting .NET Core v3.1. The baseline hardware requirements can be found [here](https://help.syncfusion.com/aspnet-core/installation-and-upgrade/system-requirements#hardware-environment).

* **Memory** - Minimum of 512MB, recommended of 1GB
* **Hard Drive**
  + Up to 2GB of available space may be required
  + 300MB of free space is required in boot drive
* **Internet** – Internet access is required
* **Processor**
  + x86 and x64 processors are supported
  + Although specifications on speed is not provided, modern web servers recommend processor of 2.0+ GHz

### Website Front-End Access Requirements

The end-user will require hardware compatible to run a modern web browser (such as Chrome, FireFox, Edge) in order to access the website.

The following is the minimum hardware requirements for Chrome, which is generally the minimum requirements amongst other browser types:

* **Hard Drive** – Minimum 100MB of free hard drive space is required
* **Memory** – Minimum 128MB of RAM
* **Internet –** Network access is required to connect
* **Processor –** Pentium 4 and above
* **Keyboard (or equivalent) –** Typing will be required to perform searches
* **Mouse (or equivalent) –** Clicking will be required to navigate the web pages

More information can be found at this [link](•%09https:/smallbusiness.chron.com/google-chrome-software-requirements-48820.html).

TODO: Android/iOS requirements?

TODO: Development environment requirements (Visual Studio and SSMS)

## Software Requirements

This section covers the software required to operate the components of MovieDbLite. In addition to Software application requirements, this section may list the Operating System (OS) requirements – as appropriate.

### Database Server Requirements

The database server will require software installed to host SQL Server Standard Edition 2017. The full server software requirements can be found in Microsoft’s documentation [here](https://docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15#hwswr).

In summary, the following two software components are required to be installed on the server hosting the database:

1. .NET Framework (v4.6)
2. Network Software that support the following protocols: Shared Memory, Named Pipes, TCP/IP, and VIA.

The following [link](https://docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15#operating-system-support) outlines the supported OS versions to host SQL Server 2017. Overall, the operating system must be a Windows Server 2012+ (2012, 2016, 2019) environment. Although Windows 8 and Windows 10 are listed as well, these are not intended to be used for remote hosting of the database.

### Web Application Server Requirements

The Web Application environment (back-end to the website) will require an Operation System environment that can host an ASP.NET Website targeting .NET Core v3.1. The full supported OS versions can be found [here](https://github.com/dotnet/core/blob/master/release-notes/3.1/3.1-supported-os.md). This section will focus on the remote server environments hosted by Windows, to remain consistent with Windows hosted database environment using SQL Server.

The following are the recommended operating system environments:

* Windows Server 2012 R2+
  + x64 and x86
* Windows Nano Server 1803+
  + x64 and ARM32

### Website Front-End Access Requirements

A modern web browser that supports HTML5 is required for the end-user to access the website.

The following are the preferred browsers to access the website:

* Google Chrome
* Microsoft Edge
* Mozilla Firefox
* Safari

Note: Although some features of the website may work in Internet Explorer, it is recommended to use one of the listed browsers above for full support.

## Functional Requirements

The MovieDbLite website will contain basic functionality to allow users to view details about movies and members of the film industry. Users may also create user accounts and rate/review movies. Administrative access will be available to allow for modification of movie and film industry details to keep the database’s data current.

This section covers the different functional requirements of the MovieDbLite web application in detail below.

* Track movie/film basic information (e.g. title, description, release date, duration, etc.)
  + Note: full data details to be described in Data Dictionary section. TODO: Link to this
* Track the different film members that are involved in the film industry, as well as basic information about them (preferred name, gender, date of birth, etc.)
* Track cast and crew members of a movie by their role (director, producer, actor, costume designer, choreographer, etc.)
* Track the different restriction ratings (e.g. G, PG, PG-13, R, etc.) that a film can be assigned to.
* Track the different international languages (English, Spanish, German, etc.) that a film can be assigned to.
* Track different annual award show events and the particular awards they give out
* Track the film members and movies that win awards at annual award show instances
* Track the different movie genres and allow movies to be classified as multiple of these genres
* Allow user to create a user account login (user name and password)
* Allow user access to modify details of their user account (user name, email address, password)
* Allow user to search for movies by movie title
* Allow user to perform an advanced movie search by other fields (description, release date, average user rating, restriction rating)
* Allow user to search for film members by their name
* Allow user to write a review and assign a user-rating (1-5 stars) to a movie
* Allow user to mark a review as helpful or unhelpful
* Allow user to sort reviews by date, helpfulness, or rating
* Reports
  + Report a movie’s basic details (title, restriction rating, language, release date, etc.)
  + Report the film members (actors, producers, crew members, etc.) of a movie
  + Report the movies a film member has been a part of
  + Report the awards given to a certain movie
  + Report the awards given to a certain film member
  + Report the movies with the highest user rating
  + Report the user reviews for a movie
* Administratively allow adding/editing movie information
* Administratively allow adding/editing the film members of a movie
* Administratively allow inputting results from award show instances

## Database Requirements

The MovieDbLite Database is hosted using Microsoft’s SQL Server 2017 (Standard Edition).

The development/management environment used to perform DML and DDL operations against the database is Microsoft SQL Server Management Studio 17 (SSMS). The full details of SSMS are seen below.



## Additional Development Details

The ASP.NET Core Web Application is Developed using the Visual Studio 2019 IDE. The programming languages are C#, JavaScript, HTML, and CSS using the MVC (Model-View-Controller) design pattern.



# Database Design Description

This database consists of a set of static and dynamic tables. Static tables can be defined as tables that are pre-defined and seeded with data upfront, and remain mostly static (meaning that the data in it is not added/modified/deleted often). The primary purpose of these tables is to avoid redundantly storing the names/descriptors for the values within other entities in the database (which may vary in length), where the value can simply be referenced by a static identifier. It also serves as a consistent/standard way of setting and viewing data in the database. For example, if these types did not exist, one may enter “Sci Fi” as a genre for a given movie, whereas another person may spell it out as “Science Fiction”. This would result in two different values being stored that mean the same thing.

The semantics about whether a table is static or dynamic is not too important, however, I chose to organize the following by it.

The static tables in the database, as well as their purpose, are as follows:

1. Award
   1. The purpose of this table is to track the different types of movie/film member awards that can be presented at annual award show events. Each award is tied to an Award Show. The Award contains a required AwardName column and optional Description.
   2. A Unique key/index constraint is added to the table for the combination of {AwardShowId, AwardName}. In other words, the AwardName should be unique for the given AwardShow. Consequently, this does also imply the AwardName by itself is not unique. For example, the Oscars and Critics’ Choice Awards both have an award named “Best Actress” (amongst other awards that also share the same names).
   3. This is considered a static table because (for the most part) the records in this table are not modified often, and have historically been fairly consistent on an annual basis when the award shows are hosted.
2. AwardShow
   1. The purpose of this table is to track the different types of annual movie/film award shows that present awards to movies/film members. The AwardShow contains a required ShowName column and optional Description.
   2. This is considered a static table because (for the most part) the records in this table are not modified often, and it is not often the case that Award Shows change. The standard ones have been around for a while (e.g. Oscars, Golden Globes, Critics’ Choice Awards, etc.)
3. FilmRole
   1. The purpose of this table is to track the different role types of the personnel/film members that contribute to making films/movies. A FilmRole contains a required RoleName column and optional Description.
   2. This is considered a static table because the records of this table will not be modified often, as the film roles in the industry have been well established over the years (e.g. Director, Writer, Producer, Actor, Costume Designer, etc.)
4. Genre
   1. The purpose of this table is to track the different genre types that movies can be classified by. A Genre contains a required GenreName and optional Description.
   2. This is considered a static table because the records of this table will not be modified often, as it is uncommon that new movie genres will be created. The existing movie genres have remained relatively unchanged for a while now (e.g. Action, Comedy, Horror, Romance, Drama, etc.)
5. Language
   1. The purpose of this table is to track the languages that the dialogue of a movie can be classified as. A Language contains a required Name column.
   2. This is considered a static table because the records of this table will not be modified often, as it is uncommon that new languages are created or established to be used in the film industry.
6. RestrictionRating

Dynamic

1. AwardShowInstance
   1. The purpose of this table is to keep track of each instance of an AwardShow. In other words, the actual award show event that takes place at a particular year. Each AwardShowInstance contains an Award Show it is in an instance of. It also contains a required Year field (the year the awards are being presented for) and DateHosted field (the Date the event instance took place).
   2. A Unique key/index constraint is added to the table for the combination of {AwardShowId, Year}. In other words, there should only be one instance of an AwardShow for a given Year. It has been the standard for a long time that these Award Shows are done on an annual basis, which is the reasoning behind the constraint.
   3. This table could borderline be considered static, however, multiple entries will be added to this on at least an annual basis, and for that reason, it can be classified as dynamic.
2. AwardWinner
   1. The purpose of this table is to keep track of the winners/recipients of Awards of a particular Award Show Instance. Each AwardWinner record tracks the AwardShowInstance it belongs to, the actual Award that was won, the Film Member it was presented to, and the Movie it was for. All of these columns are required.
   2. The combination of {AwardShowInstance, AwardId, FilmMemberId} make up the PK for the table. Multiple film members may be awarded a particular award (at an award show instance), and so this column must make up the unique primary key. For example, sometimes multiple people are awarded Best Costume Design or Best Cinematography for a particular movie.
   3. Both a FilmMember and a Movie are required to be entered for a record. This is because each Award that is presented is toward a particular Movie (there are currently not awards that exist otherwise, or at least are not tracked by this database). In addition, every Award is accepted/presented to a certain Film Member. This includes “Best Picture” award that may seemingly appear to be a “movie-only” award. However, the Producer(s) accept this award on behalf of the Movie.
3. FilmMember
   1. The purpose of this table is to track all the persons participating in the making of movies/films. This table tracks several fields for each FilmMember, including a Prefix (e.g. Dr., Mr., Mrs., Ms., etc.) (optional), FirstName (required), MiddleName (optional), LastName (required), Suffix (e.g. Jr., Sr., IV, etc.) (optional), PreferredFullName (required), Gender (M or F) (required), DateOfBirth (optional), DateOfDeath (optional), and Biography. The PreferredFullName field is an important field because this is what is intended to be used in most areas of the application to display the name of the FilmMember. This field is synonymous with “StageName”, and may differ from the person’s concatenated prefix/first/middle/last/suffix name. E.g. Katheryn Elizabeth Hudson goes by “Katy Perry”.
   2. An index (non-unique) is added to the PreferredFullName field as it will be common for searches to be performed against this field.
4. Movie
5. Movie\_Genre
6. MovieCastMember
7. MovieCrewMember
8. MovieUserReview
9. MovieUserReviewHelpful
10. User

This database

* Track Annual Award Shows
* Track Different Awards
* Track Movies and Film Members that receive awards
* Track Movies
* Track what Language a Movie is in
* Track when a Movie is or will be released
* Track the Restriction Rating on a movie
* Track a Movie’s Genres
* Allow Users to Sign-Up for Site
* Allow Users to Review Movies
* Allow Users to mark other reviews as helpful/unhelpful
* Track Actors in Movies
* Track various Crew Members that work on the set of Movies

Standards:

1. Unique Key Constraints and Index Naming
2. Column Naming
   1. PascalCased
   2. Standard Column Naming
      1. {Blank}Name (Required)
      2. Description (Nullable)
3. Indexing Guidelines
   1. Natural Key on each table
4. Table Naming
   1. PascalCased
5. Identity Keys vs. Non-Identity Keys
6. Foreign Key Naming
7. Primary Key Naming
   1. Clustered
8. View Naming
9. Table Naming
10. Surrogate keys vs. non-surrogate
    1. If a table is being FK referenced, a surrogate key is used
11. Why Film vs Movie in some instances?

Assumptions:

1. Award Shows are annual

## Design Rationale

Why did you choose a particular ER design? Think about how your database design may be challenged and how you can defend your design. For example, did you use artificial primary keys and why? Why did you choose non-identifying relationship for two entities when it could be modeled as identifying relationship? Why?

## E/R Model

General E/R model descriptions

### Entities

Detailed descriptions of the entities

### Relationships

Detailed descriptions of the relationships

### E/R Diagram

E/R diagram

## Relational Model

### Data Dictionary

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? | Valid Values |
| --- | --- | --- | --- | --- | --- | --- |
| SSN | Social Security Number | Varchar | 9 | Primary Key | Y | 9 numeric digits |

### Integrity Rules

How did you handle the mandatory fields, data formatting/conversion, and valid values for the data? How are the referential integrity established? For example, which table references which other table?

### Operational Rules

What are the constraints for some operations? For example, will the users be able to delete a patron’s information if he/she has outstanding videos? Can a patron be associated multiple records of checked-out videos? Will the application allow the users to enter a patron if the patron has the same first name and last name as an existing patron in the system?

* UserName constraints

### Operations

Describe what operations are involved for a particular use case. For example, does checking out videos involve insert/delete/update/retrieve?

## Security

* Storing of passwords in the database are hashed
* SQL Injection is prevented or mitigated via ORM model of accessing the database, or parameterized SQL queries using ADO.NET

## Database Backup and Recovery

Explain any database backup and recovery you may implement. Most RDBMSs may have incorporated backup and recovery tools into their interfaces and infrastructure. You may give a short description of.

## Using Database Design or CASE Tool

Software engineering tools provide automated or semiautomatic support for software development. A CASE (Computer-Aided Software Engineering) tool sometimes is more important than hardware for achieving good quality and productivity. Describe the tools you used to produce the artifacts for your project.

## Other Possible E/R Relationships

What were the other alternatives you considered when you designed your database?

# Implementation Description

General implementation requirements

## Data Dictionary

Data dictionary after you have your database set up. This dictionary is less detailed than the one in the previous section. Usually, a “DESCRIBE” operation in your database will provide the information needed for this section. If your RDBMS has a tool to show data dictionary; it is also fine too.

## Advanced Features

Describe any triggers, stored procedures, functions, or others used in the project to implement business rules specified in your database project; (You may include user interfaces with screen shots if you have implemented a database application).

## Queries

General query information (Your specific database queries or reports)

### Customer Bills

How did you query customer bills? Provide SQL statements.

### Customer Rental History

Same as above.

### Movie Rental History

Same as above.

### List all videos by movie category

Same as above.

### List video usage by movie category

Same as above

### List videos by format (Laser Disc or VHS)

Same as above.

### List defective videos

Same as above

### List twenty popular videos by category for customers’ recommendations

Same as above

# CRUD Matrix

CRUD matrix as described in one presentation by Dr. Kung

## List of Entity Types

## List of Functions

# Concluding Remarks

Lessons learned and strengths and weaknesses, what you may add to the database project if you have more time.

More data fields.

1. AwardShowInstance could have a location column
2. Track Production companies
3. File Attachments

Appendices

Additional information, such as known defects

Appendix A - DDL, INSERT, SELECT Statements

CREATE statements for creating database objects; INSERT statements to populate test data into the database; SELECT statements to display the test data

Appendix B - Data Dictionary Index

Index to the data dictionary (e.g., column\_name in alphabetical order, table\_name))

References

Reference material