**Database Design Document For the Museum Patron**

**version 1.2**

**prepared by: Team 8**

# **Outline**

1. Introduction
   1. Scope
   2. Purpose
   3. Brief overview of merged museum
2. Database Basic functionalities
   1. Functionalities relating to pieces of work for any of the Museums
   2. Functionalities relating to Exhibitions of the Museum
   3. Functionalities relating to Locations of the Museum
   4. Functionalities relating to Traveling Exhibition
   5. Functionalities relating to a Transaction for a work
3. Simplified E-R Diagram
4. Database Security and Privacy
5. Addition of temporal data the database

**Introduction**

**Scope:**

The scope of this report is to provide a high level architectural overview for a New Museum Database Management System (NMDBS) which is able to combine the information contained in the database of five separate museums (Figure 1.1). This report uses a high level database design to give a general overview on the handling and accessing information of all the museums.

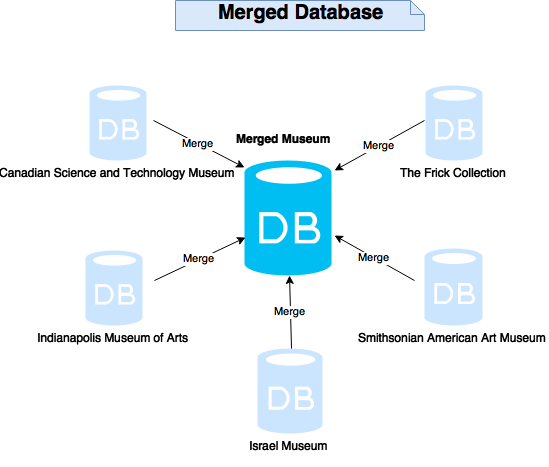
**Purpose:**

The purpose of the NMDBS project is to provide a robust and efficient museum database management system that can accurately record and later access data for each museum without losing any important information.

**Brief overview of the merged museum:**

The project consists of five different museums. Each museum specializes in one or more fields of art, science, and technology. All the museums are to be combined into the NMDBS. A short description of the what each museum specializes in is listed below.

1. The Frick Collection: specializes in different categories of art like painting, textile, sculpture, photographs, carvings, and clocks, ceramics, and metal works.
2. Indianapolis museum of Arts: Specializes in art from different time frames and regions including African, American, Asian, Mediterranean, and Contemporary art.
3. Smithsonian American Art museum: specializes in different categories of American artworks, from the colonial period to the present.
4. Israel Museum: specializes in different categories of chemistry, Physics, and African artwork with almost all its collections being paintings and sculptures.
5. Canadian science and technology museum: specializes in different science and technology inventions and creations related to past, current, and astronomy.



(Figure 1.1 New database system for merging five different museums. All relevant information from each of the partner museum database will be combined and adapted into an optimal new database)

**Database basic functionalities**

The database design was chosen so as to retain all relevant information relevant to works from all 5 partner museums, while keeping the most relevant information easily accessible and reducing the amount of confusing and disorganized features relating less relevant pieces of information. Listed below are the functionalities of the combined museum database.

**Functionalities relating to pieces of work for any of the museums**

|  |  |
| --- | --- |
| Functionalities | Functionality description |
| Recording and accessing all basic information for each piece of work | The new database design records basic categorical information, the name, the author or source, the year of creation, the date of acquisition, a description and an entry location for which data was originally entered relating to each piece of work owned by one of the museums. Some of the museums had physical properties of the pieces of work recorded, including it being a painting, sculpture, carving etc. as well as a further classification of each piece. (For example a painting can be a portrait, a landscape, etc.). In keeping this data no defining classification information of the works. |
| Secure recording of the insurance value for each piece of work | The new database records current and past insurance value for piece of work in all the partner museums while preventing non-authorized personnel from accessing it. |
| Keeps track of transactions involving a piece of work and any of the five museums. | The new database design keeps a detailed history of all works borrowed to, lent out by, sold by, or bought by any of the museums. The date of each transaction as well as a return date if the work has been lent out by or lent to one the museums can always be accessed to enable access of information related to individual transactions. Information on whether a work has been exhibitions, transactions, and whether a work is not under the ownership of one of the partner museums but has the potential to be borrowed can also be accessed. |
| Records ownership of each piece of work | The database holds information on the owner of a piece of work for all works which sold, bought by a museum, can be borrowed or have been borrowed. This can resolve issues relating to when a purchaser a work or one of the museums gained possession of an item. |
| Organized and descriptive categorization for each piece of work | The new database categorizes each piece of work according the physical properties of the piece of work with a deeper classification. Pieces of work may also be categorized based on conceptual themes which may describe features of them including the era or region a work comes from, or whether a piece of work relates to another area of life. These themes can aid in planning of current, past, and obtaining information on a piece of work by enabling a user to find a work based on a descriptive word. This information may have been derived from past classification of pieces of work in the old databases or from the descriptions for the pieces of work. All information describing what a piece is about or any non-exclusive classifications can be recorded with this feature. |
| Records a detailed history for the time and place of each piece of work | For security and insurance reasons, the location of a piece of work at any point in time since the work has been entered into any of the museums databases can be accessed, including instances where a work belonging to the museum travels out of the museum. This feature can also be used to obtain the date a piece of work was acquired by one of the partner museums. |

**Functionalities relating to exhibitions of the museums**

|  |  |
| --- | --- |
| Functionalities | Functionality description |
| Exhibition planning | The new database design gives full control over planning and management of exhibitions. This includes the ability to:   * plan and assign new exhibitions to any of the existing partner museums * assign an exhibition to take place in multiple locations (i.e. galleries) within the same museum. The same exhibition can be planned for separate locations connected by a door. * Re-plan an exhibition multiple times with the option of expanding and updating the exhibition’s collection each time. * Have exhibitions to present pieces of work borrowed from one of the partner museums or from another owner. * Access a detailed record of all past, current, and future exhibitions including the date each exhibition started, ended, or is planned to be started as well as a description, the location of and works presented in each exhibition. This feature is included to enable access to information of exhibitions which have occurred throughout each museum’s history or will occur in the museum’s future. Exhibitions by the same name at different points in time can be distinguished. |
|
| Travelling Exhibition planning | The new database design allows gives full control over planning and management of traveling exhibitions. This includes the ability to   * Create a new traveling exhibition or assign a traveling exhibition for one of the museums exhibitions. * Access information relating to enter, edit, or future traveling exhibitions including each of the locations for the exhibition, the time it was in that location, the departure and return date of the exhibition from one of the museums, the insured value of the exhibition, as well as those sponsoring the exhibition and security personnel will the ability to update and add data on future traveling exhibitions. * Allowing traveling exhibitions to have multiple sponsors. |
|
|
|
|
|
|
|

**Functionalities relating to locations of the museum**

|  |  |
| --- | --- |
| Functionalities | Functionality description |
| The ability to handle information relating to each location of the partner museums. | The new database gives the ability to assign, update, andenter, edit, or remove information on locations relating to each museum. It can handle information on the name, suggested maximum and minimum capacity, and the dimensions for a location. |
| Holding general information about doors between rooms | The new database gives the ability to enter, edit or remove information on doors between locations in the museum including rooms connected by a door and if a room is not available due to maintenance, another planned exhibition or other reasons. |

**Functionalities related to traveling exhibitions**

|  |  |
| --- | --- |
| Functionalities | Functionalities description |
| The ability to handle information identifying traveling exhibitions | The new database gives the ability to assign and update and view information identifying the traveling exhibitions including the exhibition name and start and end date of the exhibition. Information can be viewed for any traveling exhibition planned in the museum's history |
| Handling of information relating to the locations of traveling exhibitions | The database can IT, security, or view information relating to each location of a traveling exhibition including the place of the location, the date the exhibition began and ended in a location, and a contact phone number for someone at the destination. |
| Handling of information relating to sponsors | The new database can handle information relating to multiple sponsors of a traveling exhibition |
| Handling of information relating to the insurance of traveling exhibitions | Travelling exhibitions are more at risk for potential mishandling of a piece of work and have an increased insurance value based on the insurance values of the pieces of work |

**Functionalities relating to a Transaction for a Work**

|  |  |
| --- | --- |
| Functionalities | Functionalities description |
| Purchasing a work | When purchasing a piece of work, the piece is added into work data with the owner and museum assigned to the museum which it arrived at. This enables access to information which can be quickly in a museum's possession |
| Selling a work | The location of the work is assigned to be not in the museum, the owner of the work is assigned the buyer in this transaction. By keeping the work in the database information can be accessed regarding past uses of this piece of work as well as information for the buyer in case this person must be contacted. |
| Borrowing a work | If a work has never been borrowed the data for it is entered into the database system. The location of a work is always updated when a work it borrowed and the start and end dates of the borrowing are recording. The end date can be edited if this is needed. |
| Loaning a work | The location of the work is set to be not in the museum and the start and end dates of the loan are recorded. |

**Simplified E-R Diagram**

Patron Diagram.png

**Database Security and Privacy**

For security and insurance reasons the location of a work at any time since the museum has been dealing with a work can be accessed on a date and time basis. For items moving between locations in the same museum the time it starts in one location is recorded as the end date for its previous location; however, this does not happen for transfers between partner museums.

https://lh6.googleusercontent.com/wmqYSDwJr6TbPSFR_QHe_Hj4gju92bGSS238MPN_THAu3qv3y6mwFkjxem8cB60j9INdF_Xa50JeZYZwlab0TaZZRozW2K_I8-Ic8Gvyu3BauERb-Rm5OUW4YtPnVD3Ia19fKbk9

(Figure 1.4 authority for different role and positions)

The figure 1.4 shows that different role could access or view different parts of database.

The merged museum database is used by different people with different responsibilities and privileges relating to it. People can view different information from the database depending on their role. Roles people may have pertaining to the database can be employee roles such a patron, work, exhibition, and building maintenance or roles of an outsider including a visitor and sponsor of the traveling museum. Each person does not require access to all information and some information can be dangerous to freely give out, such as allowing a visitor to view insurance values of a piece of work. Below is a description of roles and what this person may access pertaining to the database as shown in figure 1.4.

|  |  |
| --- | --- |
| Role | Role Description |
| Patron | The patron is the museum manager and has privileges of accessing all information from the database containing information of all 5 partner museums. |
| IT | IT need access to all parts of the database for the develop of software such as the website of the museum, developing applications and managing the database itself. |
| Security | Security are restricted to viewing information on locations, exhibitions, items, and visitors including traveling exhibitions in order to maintain safety and to have the ability to access and provide information in the case of an unfortunate incident. |
| Building maintenance | The building maintenance are restricted to accessing and editing information on the locations and doors because they only need to perform maintenance on the physical structures of the building itself. |
| Visitor | Visitors are people who visit the museum, and view online sources such as a website or museum app which access the database. These people have limited authority and cannot edit or create information but are able to view information on pieces of sold, borrowed, or locations of the museums with the exception of private information like the insurance value of a work and current and upcoming exhibitions. |
| Sponsor | Sponsors support the partner museums by sponsoring traveling exhibitions. These people therefore can have access to information relating to the traveling exhibitions and pieces of work in case they want further input on what works are involved in a traveling exhibition. |

**Management of Historical Information**

By attaching a time period to data, it becomes possible to store history relating to different states of data. A first step in our design towards a temporal database thus is to attach a time and date to entities.

Below is a list of all data that will accurately record the date-time history of all events happening in all museums combined:

1. Our database stores temporal data for the location of any piece of work. It is common in for piece to be moved from one location to another in the same museum when for example, works are moved from a gallery to storage. In our combined museum database it is likely for an item to be transferred from one museum to another partner museum. Our database design keeps track of the history of all transfers within a museum and between museums. For every work, the database records the date and time the work is inserted into or moved out of its location. Because moving an item within a museum takes only a short time, the start date and time an item was moved into a location is recorded as the end date and time for the previous location as long as the item only changed location within the same museum. Our database also allows planning future locations of any work in the museum collections. (i.e. works can be planned to be inserted in a certain gallery of a certain museum at a future date and time.). This aids in security of a work and can be helpful for insurance purposes because the place of a work at any point in time in while it was in the museums' system can be located, with the exception of when a work is given to someone else because then the work is the responsibility of this person. The exhibition of each work can also be accessed apart from the location of the exhibition itself.
2. Our database stores temporal data for all changes in works ownership. This includes keeping track of purchased, borrowed, loaned, and sold items. Once an item is sold, future changes in ownership are not recorded as they do not the museum unless the work returns to the museum. By recording the owner at all points in time if a person should be contacted pertaining to the work this person owns, this can be done. If a work can be potentially borrowed this is not temporal data as a date cannot be attached to it, but any work can be identified as to whether it is “borrowable” or not.
3. Transactions have a date and time attached to them as multiple transactions can happen for the same item. It can also be helpful to have the date and time of a transaction for purposes of security, taxes, etc. Transactions include if a work is purchased, sold, lent, or lent out. Entering a transaction which is a buy or a sale of a work automatically creates temporal data for the owner of a work. This feature can also be used to record the date and time a work was damaged, was found to be missing, or was donated which can give more detailed information for security and insurance purposes including police reports. Temporal data for location of a work is automatically entered as appropriate depending on whether a work is bought, sold, lent or borrowed.
4. Exhibitions and the location of exhibitions are also temporal data as the same exhibition can occur at different dates also different locations and therefore the time of an exhibition is needed to differentiate between occurrences of similar exhibitions by the same name.

Below is a more detailed explanation of information pertaining to a time frame for data and the benefits of having information for the given data at every point in time.

**The Location and Exhibition of a piece of work**

* For some of the museum databases this will involve the implementation of new temporal data

|  |  |
| --- | --- |
| Timeframe | Explanation and benefits of data recorded in the time frame |
| Past | Keeps track of all history relating to the history of a piece of work. This can be useful in situations of security and insurance where a work could have been harmed or gone missing. |
| Current | Records the current location of a piece of work. This is useful for identifying the location of any work in the museum's database and can aid in planning exhibitions. |
| Future | Records where works are planned to be at future dates. This can aid in planning of exhibitions and other functions because all advance notice of where items should be can be accessed. |

**Exhibitions,** including the works and locations of exhibitions

|  |  |
| --- | --- |
| Timeframe | Explanation and benefits of data recorded in the time frame |
| Past | Keeps a detailed history for the dates of past exhibitions which can aid in planning of future exhibitions. Tracks when works and locations are had been booked for the benefit of consideration of future exhibitions or traveling exhibitions and can aid in planning of museum resources. |
| Current | Keep track of all information pertaining to exhibitions which are ongoing and gives better a more detailed handling of works which are currently out of use due to exhibitions. In the case of traveling exhibitions, data which is current can aid in control of the status of the traveling exhibition, such as the need to locate the current point of a traveling exhibition. This can also be helpful for visitors of the museum to know what is ongoing. |
| Future | Tracks when works and locations are booked for future exhibitions or traveling exhibitions and can aid in museum planning of resources. Visitors also appreciate being able to access information about exhibitions planned in the future. |

The **owner** of a work

|  |  |
| --- | --- |
| Timeframe | Explanation and benefits of data recorded in the time frame |
| Past | The owner of a piece of work is recorded when the museum is entered into the museum database. When transferring owners it can be helpful to know who has owned it lasts for security and insurance reasons. |
| Current | Keeps track of the work’s owner and gives control and organization over locating a work. |

All **transactions** will be past data; however there are multiple types of transactions for which it can be useful to access past data. Below is a detailed table explaining the handling of transaction data.

|  |  |
| --- | --- |
| Type of Transaction | Detail |
| Buy and Sell | When one of the museums buys or sells a work, at any point of time it is necessary to be able to access this information for security, insurance, and accounting purposes. |
| Loan and Borrow | All information for loans and borrowing can be accessed at any point and will be helpful for security and insurance purposes. |
| Missing | The date when a work was first noticed to be missing is   helpful for many reasons relating to history about the work. In this case the owner of the work would not automatically change. |
| Donated | Useful in determining how a work came into possession of the museum if for example the donor should be commemorated some day for their contribution to the museum. In the case of a donation the previous owner should be recorded with the same date the piece was given to the museum as the actual date the donor received the work may not be accessible. |
| Damaged | Records all points in time for which an item was noticed to be damaged and can help in determining what methods of organization of the works and other factors can contribute to the damaging of a work and possibly changes in insurance value of a work. |

Thank-you for the opportunity to design a functional database for the museum merger.

Following is a table of links which will redirect you to the official website of each museum

Appendix A

|  |  |
| --- | --- |
| Museum name | Link |
| The Frick Collection | <http://www.frick.org/> |
| Indianapolis Museum of Art | <http://www.imamuseum.org/> |
| Smithsonian American Art Museum | <http://americanart.si.edu/> |
| Israel Museum | <http://www.imj.org.il/en/> |
| Canadian Science and Technology Museum | <http://cstmuseum.techno-science.ca/en/> |

Database Design Document

version 1.2

prepared by: Team 8

# Outline

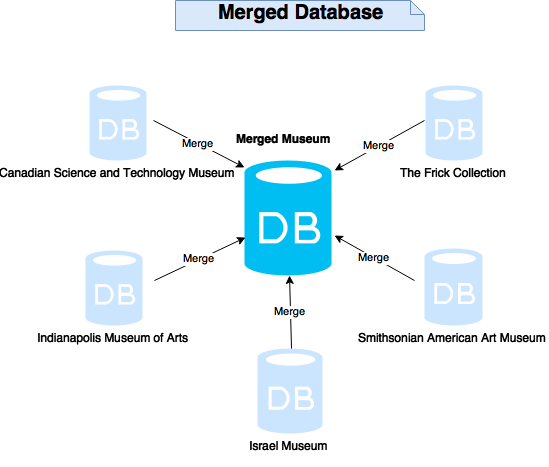
1. Introduction
   1. Scope
   2. Purpose
   3. Brief overview of merged museum
   4. Requirement of Environment
2. Domain
3. Simplified Diagram
   * 1. Core Diagram
     2. Traveling Centered Diagram
     3. Transaction Centered Diagram
4. Tables
5. Temporal Data
6. Views
7. Future Consideration

Appendix A

Introduction

###### **1.1 Scope:**

The scope of this document is to provide a high level architectural overview for a New Museum Database Management System (NMDBS) that combines the databases and information content of five different museums (Figure 1.1). This document only focuses on a high-level database design and gives a general overview on handling and accessing information of all museums to be combined.



(Figure 1.1 New database system for merging five different museums. All relevant information from each of the partner museum database will be combined and adapted into an optimal new database )

###### **1. Purpose:**

The purpose of the NMDBS project is to provide a robust and efficient museums' database management system that can accurately record and access all the information and data history pre-contained in each individual museum database without losing any important information. Specifically, our NMDBS is designed to securely handle and manage all information relating to the museums works, works locations, works owners, exhibitions, exhibition works, exhibitions locations, traveling exhibitions, traveling exhibition sponsors and how works are borrowed and loaned.

###### **1.3 Brief overview of the merged museum:**

The project consists of five different museums. Each museum specializes in one or more fields of art, science, and technology. All of the museums are to be combined into the NMDBS. A short description of the what each museum specializes in is listed below.

1. The Frick Collection: specializes in different categories of art like painting, textile, sculpture, photographs, carvings, and clocks, ceramics, and metal works.
2. Indianapolis museum of Arts: Specializes in art from different timeframes and regions such as African, American, entity, relationships, and Contemporary art.
3. Smithsonian American Art museum: specializes in different categories of American artworks, from the colonial period to the present.
4. Israel Museum: specializes in different categories of chemistry, Physics, and African artwork with the majority of its collections being paintings and sculptures.
5. Canadian science and technology museum: specializes in different science and technology inventions and creations related to past, current, and astronomy.

**1.4 Requirement of Environment**

SQL file or Text files of SQL commands, dbvisualizer, and the PostgresSQL database provided to you on the Department PostgresSql server.

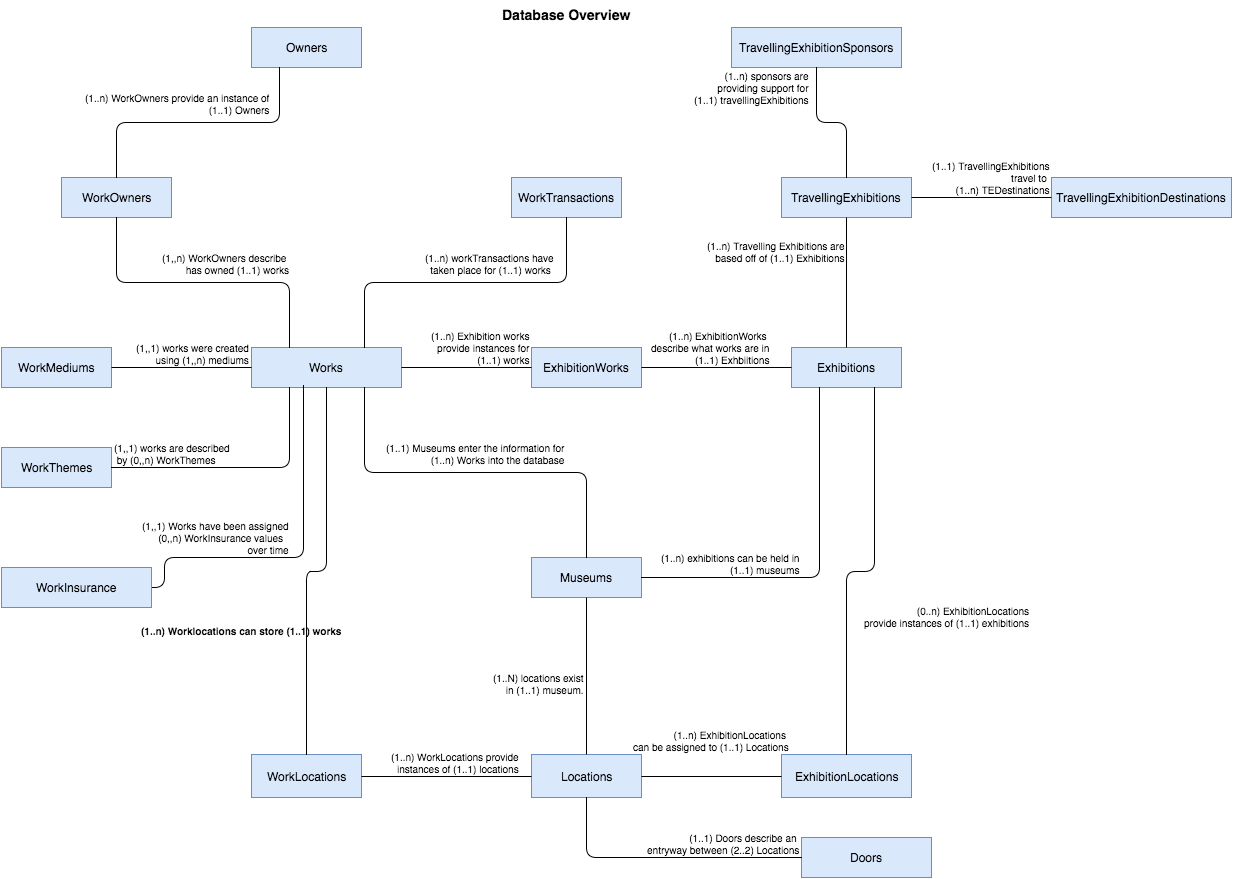
**1.5 Domains:**

Defining domains which will be used in the database is an important process because the domains must accurately describe the attributes which they are defining and have an appropriate data type which can hold the information it will be given. Listed below is table which defines all domains which will be used in the database along their type and the purpose for the domain.

|  |  |  |
| --- | --- | --- |
| Domain | Type | Purpose for the domain |
| WorkCharID | char(10) | This domain is used to check the character identifier of the primary key for museum works/items. The char data type was chosen because entries are alphabetical entries with no number |
| WorkNumID | Bigint | This domain is used to check the numeric identifier of the primary key for museum works/items, entries are only numeric and some might be large numbers. |
| DatabaseEntryLocationAddress | varchar(100) | This domain is used to check the physical address of where a work was first entered into the database. It is a varchar because because the address will likely have numbers and letters in it. |
| DatabaseEntryLocation | varchar(60) | Defines attributes which check the location (one of the five museums) of where the museum work/item was first introduced into the database. This is used both as a location stamp and as a part of the primary key for a work to prevent the collision of keys for two works possessing the same alpha and numeric keys upon combining the database. |
| WorkDescription | Text | Defines attributes which check a description of a museum item/work that is stored in the works table. |
| WorkPhysicalProperty | varchar(30) | Defines attributes which sort museum items/works by storing the physical shape/style, such as whether the work is physically a painting, sculpture, electrical, etc. |
| WorkClassification | varchar(30) | Defines attributes which sort museum items/works that share the same physical property by classifying them based on more specific characteristics (for example, a painting may be a portrait, a landscape etc.) |
| WorkCreationDate | date | Defines attributes which check the date at which a museum work was created by the original author. For museums entity, relationships, and the creation is a range or a non-specific value(such as ‘the 1800s’) The latest date from that time period will be chosen because it is known that the work is at least this old. |
| WorkCreator | varchar(50) | This domain is used to check the name of the original author/creator of a museum work. |
| WorkMedium | varchar(40) | Defines attributes which check a type of physical medium that a museum work is made of. Is a varchar 40 because it can only check one word (or two in the case of an example like ‘refined silver’). |
| LocationName | varchar(30) | Define attributes which check the name of a location within or on a travelling exhibition. |
| MuseumName | varchar(50) | Defines attributes which check the name of a museum. It is used to both refer to the database entry location and to differentiate works that share the same primary key. |
| MuseumLocation | varchar(100) | Defines attributes which check the address of a museum. |
| WorkInsureValue | float | This domain stores the insurance value of a particular museum work/item in case the work is locations, works, and damaged. It is a float because it is a value for money |
| WorkTravelInsureValue | float | Defines attributes which check the insurance value of a particular museum work/item while it is traveling for the case in which it may be stolen or destroyed. |
| OwnerName | varchar(50) | Defines attributes which check the name of a person or organization who does or has owned a work in the museum database. |
| OwnerEmail | varchar(80) | Defines attributes which will check the contact email of a person or organization who does or has owned a work in the museum database. |
| OwnershipStartDate | timestamp | Defines attributes which check the initial time in which a museum work came into the possession of a certain person or organization. Chosen as a timestamp for more accurate tracking of when a work changed ownership. It was named OwnershipStartDate over something like WorkOwnerStartDate to be more clear as to what exactly is starting. |
| OwnershipEndDate | timestamp | Defines attributes which check the final time in which a museum work was in the possession of a certain person or organization. |
| LocationName | varchar(80) | Defines attributes which stocheck re the name of a location in the real world, possibly the name of a gallery in the museum, or the name of a location that a traveling exhibition visits. |
| LocationSuggestedCapacity | int | Defines attributes which check either the upper or lower bound of the suggested amount of museum's works that a certain location can hold |
| LocationWidth | float | Defines attributes which check the width of a location in a museum in meters. |
| LocationLength | float | Defines attributes which check the length of a location in a museum in meters. |
| WorkArrivalTime | timestamp | Defines attributes which check the date and time that a particular museum work arrived at a location. Is is recorded as a timestamp for more accurate information of where works were for purposes such as security or insurance. |
| WorkDepartureTime | timestamp | Defines attributes which check the date and time that a particular museum work departed from a location. |
| ExhibitName | varchar(150) | Defines attributes which check the name of the exhibits that a museum may host. |
| ExhibitDescription | text | Defines attributes which describe to a museum curator and to the public what an exhibit is about. |
| ExhibitStartDate | DATE | Defines attributes which check the starting date of an exhibition. |
| ExhibitEndDate | DATE | Defines attributes which check the ending date of an exhibition. |
| TransactionType | varchar(100) | This domain checks a possible transaction type that the museum may be involved in, including selling a work, or changing its location. |
| TransactionTime | timestamp | Defines attributes which check the time that a transaction on a work took place. It is set as a timestamp to keep accurate data down to the second of how a work changes. |
| WorkBorrowable | Boolean | Defines attributes which determine whether a work can be borrowed or not. Used in the planning of determining whether a work is available to a museum. |
| SecurityName | varchar(200) | Defines attributes which describe the person who is in charge of security for a travelling exhibition. |
| ExhibitDepartureDate | timestamp | Defines attributes which determine the date that an exhibit left a specific travelling destination. Timestamp to keep track of the works on the road for insurance purposes. |
| DestinationPhone | bigint | Defines attributes which determine the phone number of a location that a travelling exhibition may visit. Bigint to deal with that works may be sent to international locations where phone number formats are not the same. |
| ExhibitArrivalDate | timestamp | Defines attributes which determine the arrival date of the items of a travelling exhibition to a particular location. Timestamp so as to track the items for insurance purposes. |
| SponsorName | varchar(100) | Defines attributes which determine the name of a person or organization who sponsors one or more travelling exhibitions. |
| SponsorAmount | float | Defines attributes which determine the amount of money that a sponsor has donated to sponsor a travelling exhibition. Float to keep track of every last cent that may exist in the sponsorship. |
| InsureStartDate | timestamp | Defines attributes which determine the initial date that a work held a specific insurance value. Timestamp to keep track if multiple changes are made to the same work on the same day. |
| InsureEndDate | timestamp | Defines attributes which determine the final time/date that a work held a particular insurance value. |
| Keyword | varchar(60) | Defines attributes which determine specific keywords that may be related to specific works. Used in the sorting of works, and the planning of exhibitions. |

Table 1: Each domain to be used in the database is listed in the table with the domain name on the left, the datatype of the domain in the middle and the purpose for the domain on the right and reasons for the data type chosen.

**1.6 Simplified Diagram**



**Figure 1: This** figure shows all entities which are required for the database with relationships between entities which includes multiplicity

**Main ER Diagram**

CoreDatabaseDiagram(4).png

Figure 2: This is an entity relationship diagram which shows the main entities which will be present in the museum database along with the attributes of each entity, relationships and relationships between entities including multiplicities. Each entity box contains the name of the entity in the top section of the box, all attributes which are primary keys are in the section in the middle section, and all attributes which are not primary keys are in the bottom section

Traveling centered Diagram

TravellingExhibitions(1).png

Figure 2: This entity-relationship diagram follows the same style as figure 1 and contains the entities which are used in handling travelling exhibitions, and the attributes for those entities. The TravellingExhibitions entity references the same exhibitions entity shown in figure 1.

Transaction centered diagram



Figure 3: This entity-relationship diagram follows the same style as figure 1 and contains the entities which are used in handling transactions, and the attributes for those entities. The works entity referenced is the entity shown in figure 1.

Tables in the museum database systems

**Museums:** This table stores non-temporal information about the five partner museums which are now being merged into one database system. It was added for the sake of uniquely identifying locations, works and exhibitions which may have the same name. We chose to add a museum table which other tables could reference as opposed to changing the values which are the same for both museums because if for example the character ID for a work is changed, then it can be difficult to look up any historical data pertaining to a work. This ensures all historical information for a work is accurate.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| mus-MuseumName  (Primary Key) | MuseumName | Not NULL | Stores the name of one of the five partner museums in the database.  Must be not null because this is a primary key and important in identifying the museum |
| mus-MuseumLocation  (Primary Key) | MuseumLocation | Not NULL | Stores the physical address of one of the five partner museums in the database  This must be a primary key in the event two museums in a merger have the same name and can not be null |

**Works:** The works table stores non-temporal basic information about works/items that are in the museum database. This includes information about the

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| wor-WorkCharID  (Primary Key) | WorkCharID | Not NULL | Stores the character identifier of a museum work  Not null because it is a primary key used in identifying the piece. |
| wor-WorkNumID  (Primary Key) | WorkNumID | Not NULL | Stores the numerical identifier of a museum work  Not null because it is a primary key used in identifying the piece. |
| wor-DatabaseEntrylocation  (Primary Key)  Foreign Key=> references **Museums** table. | DatabaseEntrylocation | Not NULL | Stores the location in which the work originally existed before it’s inserted into the new database.  Not null because it is a primary key used in identifying the piece, was added because the museums involved in the merge may have items with the same WorkCharID and WorkNumID and this makes each item unique. |
| wor-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Museums** table. | DatabaseEntryLocationAddress | Not NULL | Stores the physical address of the original museum where the work first entered into the new database system.  Required when accessing the DatabaseEntryLocation and thus must be a primary key |
| wor-WorkDescription | WorkDescription | None | Records a brief description about the work. This may includes information about how the work physically looks like, what the work is trying to describe, how different material/medium are used to build the work |
| wor-WorkPhysicalProperty | WorkPhysicalProperty | None | Sort museum items/works by storing what type of work the piece physically is. A work physical property can be: Metal Work, Furniture  Carving, Ceramics  Painting, Textile, Electrical Work, or Mechanical Work. This is useful because these values do not describe conceptual themes about the work which are stored elsewhere. |
| wor-WorkClassification | WorkClassification | None | Categorizes works which share the same physical property by classifying them based on more specific characteristics (for example, a painting may be a portrait, a landscape etc.). This attribute was chosen in order to maintain information on subtypes of some of the museums which served a similar purpose |
| wor-WorkCreator | WorkCreator | None | Stores the original creator of the museum work |
| wor-WorkCreationDate | WorkCreationDate | None | Stores the date that the work was created |
| wor-WorkBorrowable | WorkBorrowable | None | This attribute records whether a work in our database is a ‘potentially borrowed’ (i.e. isn’t owned by any of our five partner museums) This is required because whether a work locations, works, and be borrowed is not temporal data. Some works have been sold and will most likely not be borrowable and items owned by the museum do not need to be borrowed but items not owned by the museum must indicate if they can be borrowed. |

**WorkMediums:** This table stores information about the materials and media used in creating each of the museum’s works. This table must be present because many of the works from each of the museums have multiple stolen, destroyed, orAsian, Mediterranean, and they have been made of.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| wme-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table. | WorkCharID | Not Null;  foreign key constraints | Needed because it is a unique character identifier of a work |
| wme-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table. | WorkNumID | Not Null;  foreign key constraints | Needed because it is a unique numerical identifier of a work |
| wme-DatabaseEntryLocation  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntryLocation | Not Null;  foreign key constraints | Stores the location in which the work was originally entered into the new database. Must be added to all medium table to because it is a primary key from the works table that uniquely identifies works which may have the same WorkNumID and WorkCharID |
| wme-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | Stores the physical address of the original museum where the work first entered the new database system. |
| wme-WorkMedium | WorkMedium | Not Null (each work must at least have one medium) | The physical materials and media the work are created from. |

**Location:** This table stores non-temporal information about each of the locations that physically exist in each museum. This information includes location name, dimensions, suggested maximum and minimum works capacity, and the partner museum where the location exists.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| loc-LocationName  (Primary Key) | LocationName | Not Null | Stores the name of the museum room or location (ex: museum galleries) |
| loc-MuseumName  (Primary Key)  Foreign Key=> references **Museums** table. | MuseumName | Not Null;  foreign key constraints | Stores the name of the museum that the location exists in. |
| loc-MuseumLocation  (Primary Key)  Foreign Key=> references Museums table. | LocationName | Not Null;  foreign key constraints | Stores the physical address of the museum, where the location exists in. |
| loc-LocationSuggestedCapacityMin | LocationSuggestedCapacityMin | Value Must be greater than or equal to zero. | Stores the suggested minimum number of works that can exist in a location |
| loc-LocationSuggestedCapacityMax | LocationSuggestedCapacityMax | Value Must be greater than or equal to zero. | Stores the suggested maximum number of works that can fit in a location |
| loc-LocationWidth | LocationWidth | Value Must be greater than or equal to zero. | Stores the width of the location measured in meters. |
| loc-LocationLength | LocationLength | Value Must be greater than or equal to zero. | Stores the length of the location measured in meters |

**WorkLocations:** This is table records temporal information regarding the works’ location. This includes recording past, current, and future locations of a work. It provides a full location history for all works in the five partner museums. The table is necessary in order to obtain information on where a work was located at different times and that is why the location of a work must be temporal data. The table mainly has attributes for foreign keys which reference a location and a work and also possesses the time of arrival and time of departure for a work.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| wol-WorkCharID  (Primary Key)  Foreign Key=> references **Works** tabl | WorkCharID | Not Null;  foreign key constraints | Stores the character identifier of a museum work |
| wol-WorkNumID  (Primary Key)  Foreign Key=> references **Works** tabl | WorkNumID | Not Null;  foreign key constraints | Stores the numerical identifier of a museum work |
| wol-DatabaseEntrylocation  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntrylocation | Not Null;  foreign key constraints | Stores the location from where the work originally entered into the new database. This was added because a DatabaseEntryLocation is primary key in the works table. |
| wol-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | Stores the physical address of the museum where the work first entered the system |
| wol-LocationName  (Primary Key)  Foreign Key=> references **Locations** table. | LocationName | Not NULL | Stores the name of the museum room or location (ex: museum galleries) that the work is stored or was stored in |
| wol-MuseumName  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumName | Not Null;  foreign key constraints | Name of the museum that has the location where the work was or is being stored in |
| wol-MuseumLocation  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumLocation | Not Null;  foreign key constraints | Stores the physical address of the museum, where the work’s location exists in |
| wol-WorkArrivalTime  (Primary Key) | WorkArrivalTime | Not Null | Stores the time and date that a museum work arrived at a specific location, required as a primary key to differentiate occasions of the same work showing up in the same location. |
| wol-WorkDepartureTime | WorkDepartureTime | None | Stores the time and date that a museum work departed from a specific location in order to know the extent that an item was in a location. |

**Doors:** This table stores non-temporal information about the pathways between rooms and/r gallery in each of the five partner museums. This table allows a location (i.e. room or gallery) to be connected to one or more other locations in the same museum. This table helps in planning exhibitions that need to use more than one location (i.e. gallery) in a museum. For every door physically in the museum between two rooms, room a and room b for this example, 2 entries are made into the database showing that there is a door leading from room a to room b and a door leading from room b to room a excluding cases of a one way door. This decision was made so that it could be specified if a door is only way because there would only be one record for the door in the database and the original location would not be reachable by the end location.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| dor-MuseumName  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumName | Not Null;  foreign key constraints | Stores the name of one of the five partner museums in the database. This attribute is not need for both locations because locations in separate museums would not be connected by a door. |
| dor-MuseumLocation  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumLocation | Not Null;  foreign key constraints | Stores the physical address of one of the five partner museums in the database |
| dor-LocationNameOrigin  (Primary Key)  Foreign Key=> references **Locations** table. | LocationName | Not Null;  foreign key constraints | A museum’ location name that represents the origin or start of the connection (pathway). |
| dor-LocationNameEnd  (Primary Key)  Foreign Key=> references **Locations** table. | LocationName | Not Null;  foreign key constraints | A museum’s location name that represents the end or final point of the connection (pathway). |

**Exhibitions:** This table stores basic and temporal information about all current and past exhibitions that has or will be displayed in any of the five partner museums. This information include the name of exhibitions, a brief description of the exhibition and its collection, the museum in which the exhibition has occurred, currently occurring or will occur in the future and most important the date and time in which each exhibition starts and ends. It was chosen to be temporal data because exhibitions with the same name may happen more than once so a startdate is needed to differentiate them

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| exh-ExhibitName  (Primary Key) | ExhibitName | Not Null; | Stores the name of an exhibition that is either held in one of the five museums or as a traveling exhibition abroad. |
| exh-ExhibitDescription | ExhibitDescription | None | Stores a brief description of the exhibition and the exhibition works’ collection. |
| exh-ExhibitStartDate  (Primary Key) | ExhibitStartDate | Not Null; | The start date that an exhibition is displayed to the public. |
| exh-ExhibitEndDate | ExhibitEndDate | Not Null;  Value > StartDate | The end date that an exhibition is displayed to the public. |
| exh-MuseumName  (Primary Key)  Foreign Key=> references Museums table. | MuseumName | Not Null;  foreign key constraints | Stores the name of one of the five partner museums in which the exhibition is or will be held. Was added as a primary key in order the differentiate exhibitions which may occur or have occurred at separate museums on the same date. |
| exh-MuseumLocation  (Primary Key)  Foreign Key=> references **Museums** table. | MuseumLocation | Not Null;  foreign key constraints | Stores the physical address of one of the five partner museums in which the exhibition was, currently, or will be held. |

**ExhibitionLocations:** This table stores temporal information about the locations that exhibits are, have, or will be occupying. It is temporal because it is accessing temporal data from the exhibitions table which is needed to differentiate exhibitions by the same name in the same museum. All attributes are needed to reference the exhibition and location for which an Exhibition\_Location record will refer to.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| elo-ExhibitName  (Primary Key)  Foreign Key=> references Exhibitions table. | ExhibitName | Not Null; | Stores the name of an exhibit that is either displayed in one of the five museums or as a traveling exhibition abroad. |
| elo-ExhibitStartDate  (Primary Key)  Foreign Key=> references **Exhibitions** table. | ExhibitStartDate | Not Null;  foreign key constraints | The start date that the exhibit was displayed at a certain location. Sets apart exhibitions at the same museum by the same name and also gives a timeline for when an exhibition will be run. |
| elo-LocationName  (Primary Key)  Foreign Key=> references **Locations** table. | LocationName | Not Null;  foreign key constraints | Name of a location within one of the five partner museums that an exhibit is displayed. |
| elo-MuseumName  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumName | Not Null;  foreign key constraints | Stores the name of one of the five partner museums in which the exhibition will be held.. |
| elo-MuseumLocation  (Primary Key)  Foreign Key=> references **Locations** table. | MuseumLocation | Not Null;  foreign key constraints | Stores the physical address of one of the five partner museums in which the exhibition was, will be held. |

**ExhibitionWorks**: This table stores non-temporal information about the works contained in each Exhibition. Note that: Exhibitions are identified by their name, starting date and the museum they are located at. The date an item entered the exhibition is not needed because only the location is needed for security and insurance purposes so it was decided a start date for the item would not be included. All attributes are needed as they are needed to uniquely identify exhibitions and works for which the table refers to.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| elo-ExhibitName  (Primary Key)  Foreign Key=> references **Exhibitions** table. | ExhibitName | Not Null; | Stores the name of an exhibit that is either displayed in one of the five museums or as a traveling exhibition abroad. |
| elo-ExhibitStartDate  (Primary Key)  Foreign Key=> references **Exhibitions** table. | ExhibitStartDate | Not Null;  foreign key constraints | The start date that the exhibit was displayed at a certain location. |
| exw-MuseumName  (Primary Key)  Foreign Key=> references **Exhibitions** table. | MuseumName | Not Null;  foreign key constraints | Name of the museum that the exhibit was contained in. |
| exw-MuseumLocation  Foreign Key=> references **Exhibitions** table. | MuseumLocation | Part of the primary key so it cannot be null. | Physical address of the museum that the exhibition was contained in. |
| exw-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table. | WorkCharID | Not Null;  foreign key constraints | Stores the character identifier of a museum work |
| exw-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table. | WorkNumID | Not Null;  foreign key constraints | Stores the numerical identifier of a museum work |
| exw-DatabaseEntrylocation  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntrylocation | Not Null;  foreign key constraints | Stores the location in which the work originally existed before it’s inserted into the new database. |

**Owners:** This table stores non-temporal information about the organizations or people who currently possess or have owned a work/item in the museum’s database. This was chosen to be one of the tables for the sake of a lack of redundancy of information. A work can have multiple owners over its existence so it is not sufficient to enter an owner attribute into the works table. An owner can also have multiple works so if there is an owner's table that a Works\_Owners table must reference then this will make finding information owned by an owner easier. If, for example, a Works\_Owners table did not have to reference a record from an owner’s table then the owner may be misspelled sometimes when being entered into the database and this would create problems when trying to access all works by the same owner.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| own-OwnerName  (Primary Key) | OwnerName | Not Null | Stores the name of a person or organization who owns or has owned any of the works in the five museums collection. Is a primary key to differentiate owners using the same email. |
| own-OwnerEmail  (Primary Key) | OwnerEmail | Not Null | Stores the email address of a person or organization who owns or has owned any of the works in the five museums collection. Included to contact the owner if needed and is primary to differentiate owners by the same name. |

**WorkOwners:** This table stores temporal information about the organizations or people who have owned or currently own a specific museum work/item in the museum database. All attributes except for OwnershipStartDate and OwnershipEndDate are required because they reference the primary keys of other tables. OwnershipStartDate is still required as a primary key however in the event that an owner purchases and sells a work multiple times.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| won-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table. | WorkCharID | Not Null;  foreign key constraints | References the character identifier of a museum work. |
| won-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table. | WorkNumID | Not Null;  foreign key constraints | References the numerical identifier of a museum work. |
| won-DatabaseEntryLocation  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntryLocation | Not Null;  foreign key constraints | stores the name of the museum that first introduced the work into the system |
| won-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table. | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | references the physical address of the museum where the work first entered the system. |
| won-OwnerName  (primary key)  Foreign Key=> references **Owners** table. | OwnerName | Not Null;  foreign key constraints | The name of the person or organization who possesses the work at a given point in time. |
| won-OwnerEmail  (primary key)  Foreign Key=> references **Owners** table. | OwnerEmail | Not Null;  foreign key constraints | The email of the person or organization who possesses the work at a given point in time. |
| won-OwnershipStartDate  (Primary Key) | OwnershipStartDate | Not Null | The initial date that a work came into a person or organization’s possession. |
| won-OwnershipEndDate | OwnershipEndDate | None | The last date that a work was in a people possession. Helpful in determing how long a museum owned a work. Will not have values for works sold from the museum most of the time because it is not the museums business to know how long an outside owner has owned a work. It can therefore be null. |

**WorkTransactions:** This table stores temporal information about transactions involving museums’ works. Works’ transaction include purchasing a new work, loaning, borrowing, and selling a work as well as incidence of a work being donated, damaged, or when a work has gone missing.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| wtr-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table. | WorkCharID | Not Null;  foreign key constraints | References the character identifier of a museum work which was involved in the transaction. |
| wtr-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table. | WorkNumID | Not Null;  foreign key constraints | References the numberical identifier of a museum work which was involved in the transaction. |
| wtr-DatabaseEntryLocation  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocation | Not Null;  foreign key constraints | references the location that the museum work which was involved in the transaction initially entered the database. |
| wtr-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | references the physical address of the museum where the work first entered the system. |
| wtr-TransactionType  (Primary Key) | TransactionType | Not Null | Stores the type of transaction happened on the work. A transaction can be either purchased, loaned, borrowed, sold, gone missing, donated, or damaged |
| wtr-TransactionTime  (Primary Key) | TransactionTime | Not Null | Stores the date and time that the transaction took place. Required as a primary key in order to differentiate items which have the same kind of transaction more than once. |

**TravellingExhibitions:** This table stores temporal information about exhibitions that are set to be travelling. Each travelling exhibition must originally belong to one of our five partner museums. It was chosen to be it’s own table apart from the exhibitions table because there are attributes for the travelling exhibitions which are not required by exhibitions. A travelling exhibition is still an exhibition however, so it refers to the exhibitions table.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| tre-ExhibitName  (Primary Key)  Foreign Key=> references **Exhibitions** table | ExhibitName | Not Null;  foreign key constraints | Name of an exhibition that is sent to various locations (traveling). |
| tre-MuseumName  (Primary Key)  Foreign Key=> references **Exhibitions** table | MuseumName | Not Null;  foreign key constraints | Name of one of the five museums that is in charge of a travelling exhibition. |
| tre-MuseumLocation  (Primary Key)  Foreign Key=> references **Exhibitions** table | MuseumLocation | Not Null;  foreign key constraints | The address of one of the five museums that is in charge of a travelling exhibition. |
| tre-ExhibitStartDateOriginal  (Primary Key)  Foreign Key=> references **Exhibitions** table | ExhibitStartDate | Not Null;  foreign key constraints | The date the exhibition originally ran in any of our five partner museums. |
| tre-ExhibitDepartureDate | ExhibitDepartureDate | Not Null;  Value > than the value of ExhibitStartDateOriginal | The date the works departed from the original museum and were sent out to be displayed in a traveling exhibition. |
| tre-Security | SecurityName | None | Name of the person in charge of security for the exhibit |

**TravellingExhibitionDestinations:** This table stores temporal information about the specific locations abroad that a traveling exhibition has visited. It is included because each travelling exhibition has multiple locations for which it visits, which all have unique information. It is also temporal because a travelling exhibition can visit the same location more than once; therefore, along with the start date for the whole travelling exhibition the time spent in the destination must also be recorded and must be part of the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| ted-ExhibitName  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | ExhibitName | Not Null;  foreign key constraints | The name of the travelling exhibit that is being displayed at a certain destination |
| ted-MuseumName  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | MuseumName | Not Null;  foreign key constraints | Name of the original museum that is in charge of the exhibition. |
| ted-MuseumLocation  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | MuseumLocation | Not Null;  foreign key constraints | Physical address of the museum that is in charge of the exhibition. |
| ted-ExhibitStartDateOriginal  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | ExhibitStartDate | Not Null;  foreign key constraints | The initial date in which the traveling exhibition as a whole began being displayed to the public. |
| ted-DestinationPhone | DestinationPhone | None | The phone number of the destination that the travelling exhibition is being displayed at was included in this table as opposed to having another table with destination phone numbers for each location because a different phone number may need to be accessed every time a location is visited, even if the location is visited more than once on the same tour. |
| ted-LocationName  (Primary Key) | LocationName | Not Null | The name of the location that the travelling exhibition is being displayed at. |
| ted-LocationAddress  (Primary Key) | LocationAddress | Not Null | The physical address of the location that the travelling exhibition is being displayed at.  Note: two locations can have the same name, but they physically exists in two different addresses. |
| ted-ExhibitArrivalDate  (Primary Key) | ExhibitArrivalDate | Not Null  Value > than the value of ted-ExhibitStartDateOriginal and tre-ExhibitDepartureDate | stores the initial date that the traveling exhibit was at that location. |
| ted-ExhibitDepartureDate  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | ExhibitDepartureDate | Not Null;  foreign key constraints  Value > than the value of ted-ExhibitArrivalDate | The last date that the travelling exhibition was at a specific location. |

**TravellingExhibitionSponsors:** This table stores temporal information about the people or organizations who sponsor specific traveling exhibits. It has it’s own table because there can be multiple sponsors for a travelling exhibition so it is not appropriate to have a sponsor attribute in the travelling exhibition table. All attributes except sponsorAmount are required because they are either foreign keys which refer to other tables primary keys for the travelling exhibition or in the case of sponsor name, it is a unique identifier for

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| spo-SponsorName  (Primary Key) | SponsorName | Not Null | Holds the name of a person or organization sponsoring a travelling exhibition. |
| spo-ExhibitName  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | ExhibitName | Not Null;  foreign key constraints | The name of the travelling exhibit the sponsor is sponsoring. |
| spo-MuseumName  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | MuseumName | Not Null;  foreign key constraints | The name of the museum that is responsible for the traveling exhibition that the person or organization is sponsoring. |
| spo-MuseumLocation  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | MuseumLocation | Not Null;  foreign key constraints | The physical address of the museum that is responsible for the traveling that the person or organization is sponsoring. |
| spo-ExhibitStartDateOriginal  (Primary Key)  Foreign Key=> references **TravellingExhibitions** table | ExhibitStartDate | Not Null;  foreign key constraints | The original starting date of the traveling exhibit that the person or organization is sponsoring. |
| spo-SponsorAmount | SponsorAmount | None | The amount of money that the sponsor paid to sponsor the exhibit. Included because it is helpful to know some of the financial benefits of the travelling exhibitions. |

**WorkThemes:**  This table stores non-temporal information about keywords and concepts related to museum works/items that helps in the organization and planning of future exhibitions. A work may fit into multiple themes so this table was created and the themes are based on what conceptual qualities a piece may have such as it being “fine brush be commemoratedwork”, a theme may describe a region or era a piece comes from. Themes do not fall under the same category as Physical property or classification because they are describing things about the work which are more relative terms and do not describe anything about what type of work the piece is.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| wth-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table | WorkCharID | Not Null;  foreign key constraints | stores the reference to the character identifier of a museum work. |
| wth-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table | WorkNumID | Not Null;  foreign key constraints | stores the reference to the numerical identifier |
| wth-DatabaseEntryLocation  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocation | Not Null;  foreign key constraints | stores the reference to the location where the work initially entered the database. |
| wth-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | references the physical address of the museum where the work first entered the system. |
| wth-Theme  (Primary Key) | Theme | None | stores the theme that is linked to a particular work |

**WorksInsurance:** This table stores temporal information about the insurance value of a particular museum work/item, including its insurance value when traveling abroad. It shows the changes the item may experience over time and this reason it is stored as temporal data.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | Domain Name | Attribute constraints | Attribute description |
| win-WorkCharID  (Primary Key)  Foreign Key=> references **Works** table | WorkCharID | Not Null;  foreign key constraints | references the character identifier of a museum work. |
| win-WorkNumID  (Primary Key)  Foreign Key=> references **Works** table | WorkNumID | Not Null;  foreign key constraints | References the numerical identifier of a museum work. |
| win-DatabaseEntryLocation  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocation | Not Null;  foreign key constraints | references the museum which entered the work into the database originally. |
| win-DatabaseEntryLocationAddress  (Primary Key)  Foreign Key=> references **Works** table | DatabaseEntryLocationAddress | Not Null;  foreign key constraints | references the physical address of the museum which entered the work into the database originally |
| win-WorkInsureValue | WorkInsureValue | Monetary insurance based so it must be greater than or equal to 0. | The insurance value that a work held at a particular point in time. |
| win-WorkTravelInsureValue | WorkTravelInsureValue | Based on the assignment. If no value is entered the default value is 110% of the original value of the work but this value may be higher. | The insurance value that a work held at a particular point in time if it was traveling. |
| win-InsureStartDate  (Primary Key) | InsureStartDate | Part of the primary key so it cannot be null. | The initial date that a work held a particular insurance value. |
| win-InsureEndDate | InsureEndDate | NONE | The last date that a work held a particular insurance value. |

Primary Key win-WorkCharID, win-WorkNumID, win-DatabaseEntryLocation, win-DatabaseEntryLocationAddress, win-InsureStartDate

Foreign key win-WorkCharID, win-WorkNumID, win-DatabaseEntryLocation, win-DatabaseEntryLocationAddress references Works.

###### **Temporal Data**

Below is a detailed explanation of information pertaining to a timeframe for data and the benefits of having information for the given data at every point in time organized by tables requiring temporal data.

**Work location**

Keeps track the work location including the history location, current location and future planned location. A datetime attribute is necessary because the exact time of an item needs to be known. When works are moved between locations in the same museum a trigger will change the end date of the last location to the start date of the new location because moves within a museum are short. The same trigger will not apply for moves between museums.

|  |  |
| --- | --- |
| Past | Keeps track of all history relating to the history of a piece of work. This can be useful in situations of security and insurance where a work could have been harmed or gone missing. |
| Current | Records the current location of a piece of work. This is useful for identifying the location of any work in the museum's database and can aid in planning exhibitions. |
| Future | Records where works are planned to be at future dates. This can aid in planning of exhibitions and other functions because all advance notice of where items should be can be accessed. |

**Exhibitions**

An exhibition is temporal data because exhibitions that have the same name. The start date of an exhibition is therefore recorded as a primary key to differentiate between these exhibitions.

|  |  |
| --- | --- |
| Past | Keeps a detailed history for the dates of past exhibitions which can aid in planning of future exhibitions. |
| Current | Keep track of all information pertaining to exhibitions which are ongoing. In the case of travelling exhibitions, data which is current can aid in control of the status of the travelling exhibition, such as the need to locate the current point of a travelling exhibition. This can also be helpful for visitors of the museum to know what is ongoing. |
| Future | Visitors appreciate being able to access information about exhibitions planned in the future and museum staff can benefit from knowledge of future exhibitions. |

**Exhibition locations** (references exhibition and location)

This data is temporal by extension of it referencing the exhibition table

|  |  |
| --- | --- |
| Past | Can answer questions of interest for which exhibition was occurring in which location. |
| Current | Gives better a more detailed handling of locations which are currently out of use due to exhibitions |
| Future | Tracks when locations are booked for future exhibitions or travelling exhibitions and can aid in museum planning of resources. |

**Exhibition works**

|  |  |
| --- | --- |
| Past | Tracks when works have been booked for the benefit of consideration of future exhibitions or travelling exhibitions and can aid in planning of museum resources. |
| Current | Gives better a more detailed handling of works which are currently out of use due to exhibitions |
| Future | Tracks when works are booked for future exhibitions or travelling exhibitions and can aid in museum planning of resources. |

**Works owner**

|  |  |
| --- | --- |
| Past | The owner of a piece of work is recorded when the museum is entered into the museum database. When transferring owners it can be helpful to know who has owned it last for security and insurance reasons. |
| Current | Keeps track of the work’s owner and gives control and organization over locating a work. |

**Work transactions**

|  |  |
| --- | --- |
| Type of Transaction | Detail |
| Buy and Sell | When one of the museums buys or sells a work, at any point of time it is necessary to be able to access this information for security, insurance, and accounting purposes. For this reason it is necessary to be able to query all instances of purchases or sales. |
| Loan and Borrow | All information for loans and borrowing can be accessed at any point and will be helpful for security and insurance purposes. All loans and works borrowed should be recorded to be able to easily query for how many loans are occurring. |
| Missing | The date when a work was first noticed to be missing is very helpful for many reasons relating to history about the work. In this case the owner of the work would not automatically change. It can be helpful to query a history of when items have gone missing for security purposes. |
| Donated | Useful in determining how a work came into possession of the museum if for example the donor should whether some day for their contribution to the museum. In the case of a donation the previous owner should be recorded with the same date the piece was given to the museum as the actual date the donor received the work may not be accessible. |
| Damaged | Records all points in time for which an item was noticed to be damaged and can help in determining what methods of organization of the works and other factors can contribute to the damaging of a work and possibly changes in insurance value of a work. It can be helpful to be able to query when a work was damaged for security purposes |

**Traveling Exhibits**

Travelling exhibitions are temporal data for the same reason that exhibitions are temporal data, because a travelling exhibition by the same name can occur more than once and the dates attached to the primary key help to differentiate the travelling exhibitions.

**Travelling destinations**

Destinations are temporal data because the same destination can be travelled to within the same travelling exhibition, even on the same tour and the dates of arrival at these destinations need to differentiate them.

Which tables are needed to add new temporal (improve design), why

1. For the WorkLocations table
   1. We need to add a timestamp (i.e. time) to the moving in date and moving out dates. This will allow our database not to not only record a more accurate time for when an item is inserted in a location or moved out of it.
   2. A detailed date-time works location history is important for museum administrators and insurance companies for the purpose of tracking the location of an item minute by minute
2. For the Exhibitions table we need to add temporal information about the time an exhibition starts and time it ends. This temporal data is helpful for the museum visitors to know what times they can expect to visit the exhibitions and what times it closes.
3. For the ExhibitionWorks table we need to to add time to the attributes containing date.

###### **Views**

There are a number of benefits to implementing views in this database

**A view could hide complexity**

Which means if we need to query multiple tables or perform complex logic or calculations to get a result we can simply code the logic into a view. Views can also improve the security of a database by limiting access to certain information depending on the responsibilities and privileges each person has pertaining to the database.

**Views can perform a task**

Listed below are a number of task views of database can have.

**Task views**:

1. Listing of all publicly available data on all works in each exhibition sorted by the exhibition and by the name of the work
2. Listing of works sorted by when they are available for use in a new exhibition, by classification, and by the name of the work
3. Listing of the amount of additional works that could be added to each exhibition based on the unused capacity of the galleries that they are currently in. This listing should just have the amount of additional works that could be added and the name of the exhibit,
4. Listing all current and future exhibitions including the name of the exhibition, the dates it is happening (starting and ending dates), the maximum recommended capacity of the locations it is using, and amounts of works planned for it
5. Listing the different locations that a given work was/is/will be in between two dates
6. Listing all the exhibitions that make use of a location between two dates

And save those into view for further usage

Below is a list of views which can be used by different roles of people accessing the database which can create better security for the database.

**View could be used as security mechanism**

by allowing users to access data through the view, without granting the user permission to directly access the underlying tables or information the user does not need



( Figure 4 )

The merged museum database is used by different people with different responsibilities and privileges relating to it. People are able to view different information from the database depending on their role. Roles people may have pertaining to the database can be employee roles such a patron, IT, security, or building maintenance or roles of an outsider such as a visitor and sponsor of the travelling museum. Each person does not require access to all information and some information can be dangerous to freely give out, such as allowing a visitor to view insurance values of a piece of work. Below is a description of roles and what this person may access pertaining to the database as shown in figure 4..

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Role Description | View Name | View Detail |
| Patron | Museum manager | PatronView | The patron is the museum manager and has privileges of accessing all information from the database containing information of all 5 partner museums. The patron will not want to however, deal with the underlying tables. |
| IT | It a representative of the IT Department which maintains things like the museum website or app, | ITView | IT need access to all parts of the database for the develop of software such as the website of the museum, developing applications and managing the database itself. |
| Security/Guard | Is a representative of the security department who are responsible for safety in the museum and items which belong to the museum. | SecurityView | Security are restricted to viewing information on locations, exhibitions, items, and visitors including travelling exhibitions in order to maintain safety and to have the ability to access and provide information in the case of an unfortunate incident. |
| Building maintenance | Is representative of the maintenance department and may not be an employee of the museum, but should still have access to information about the location and doors to perform maintenance | BuildingMaintenance | The building maintenance are restricted to accessing and editing information on the locations and doors because they only need to perform maintenance on the physical structures of the building itself. |
| Visitor | Are representatives of a group people who visit the museum or should not be able to edit or create information but should be able to view it. | VisitorView | Visitors are people who visit the museum, and view online sources such as a website or museum app which access the database. These people have limited authority and cannot edit or create information but are able to view information on pieces of work, exhibition, and locations of the museums with the exception of private information like the insurance value of a work and current and upcoming exhibitions. |
| Sponsor | It a representative of a group people who sponsor or support our museum by sponsoring one or more traveling exhibition (s) | SponsorView | Sponsors support the partner museums by sponsoring travelling exhibitions. These people therefore can have access to information relating to the travelling exhibitions and also pieces of work in case they want further input on what works are involved in a travelling exhibition. |

###### **Future Consideration**

Thinking of database design can be a heavy subject, and the goal of the team database is for the database to be efficient and scalable.

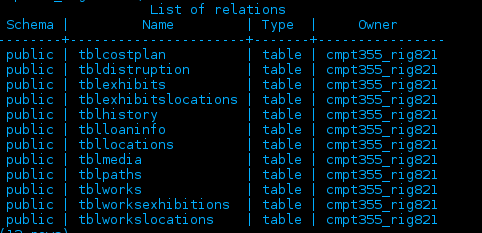
Indexes can be created for frequently used queries on big tables, which will improve the speed of data retrieval operations.

**Individual Reports**

**PART D**

**PART D: Database Migration Report- Richard Granger: Canadian Museum of Science And Technology**

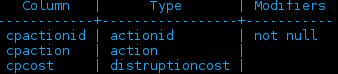
The following is a report on how I plan to adjust my database to fit the new amalgamated museum database design that our group has developed. As my database stood prior to the group project, it contained 12 tables, as evidenced here:



In this report I will examine the structure of each of the tables in my current database, and will explain how I intend to migrate the data into the new database design. In the last section I will also explain my tests table by table.

**Table 1: tblcostplan**

**The following was the original structure of this table:**



**Purpose:**

This table was designed for storing alternate options for calculating distruption costs in regard to planning exhibits.

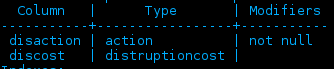
**Future Plans:**

I designed this table for the bonus part of assignment 3, and since it does not contain any useful data, the table will remain part of the legacy database and no information will be carried forward to the new system.

Testing: No testing is required since no data is being moved forward.

**Table 2: tbldistruption**

**The following was the original structure of this table:**



**Purpose:**

This table was designed for storing the cost of each action with regard to stopping or postponing exhibitions with regards to the planning of disruption costs of new exhibitions.

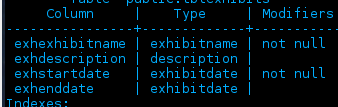
**Future Plans:**

This table was designed for the bonus section of assignment 3, and since it does not contain any useful data, will not be moved forward to the new database. All data in this table will remain in the old database exclusively.

Testing: No testing is required since no data is being moved forward.

**Table 3: tblExhibits**

**The following was the original structure for this table:**



**Purpose:** This table was designed to hold the information related to the exhibitions that my museum had, whether they were showcased within the museum or abroad as a travelling exhibition.

**This table will primarily be moved to the exhibitions table:**

Exhexhibitname- this attribute is the name of the exhibit, and so we can simply move over this field into the new exh-ExhibitName attribute of the unified database.

Exhdescription- this attribute is the description of the exhibit and can be simply moved over to exh-ExhibitDescription. They are the same datatype and should have no trouble moving with the names.

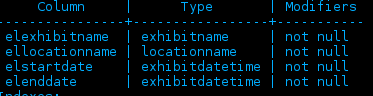
exhStartDate- this attribute is the starting date of the exhibit and can also be moved forward to exh-ExhibitStartDate. They are similiar datatypes so transferring them should be easy.

Exhenddate- this attribute is the end date of an exhibit, and it shares the similar data type, so it can easily be transferred to the exh-ExhibitEndDate field in the new database structure.

**Testing**: To test this migration, we will produce a report of the old tblExhibits and the new Exhibits table. Their structures are almost identical so an examination of the two tables will show whether the data transferred successfully. We simply create a report of both tables and the data of the two tables will be exactly the same.

**Table 4: tblExhibitsLocations**

The following was the original structure for this table:



Purpose: This table stored the locations of the exhibits that my museum displayed, including the locations of travelling exhibitions.

This table will be moved to the Exhibitions\_Locations table in the unified database, with the elo-MuseumLocation, and elo-MuseumName which did not exist in my own database, will be added to the new table manually by referencing the new Museums table. As for the attributes in my current table:

elexhibitname-This is the name of the exhibition and can simply be moved over to the new table easily

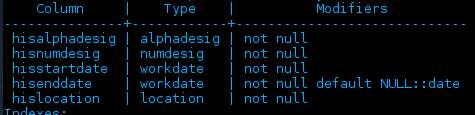
ellocationname-This is the name of the location that the exhibit was at that location. This can simply moved manually to the new table, as the names of the locations are still the same.

Elstartdate, elenddate- This two attributes can simply be moved to the new table, but since it is a timestamp instead of a date, the time may need to be adjusted to ensure that no data is simply created.

Testing: To ensure that this data has been moved forward successfully, we can create a report of the table from the old database, and a table from the new database with specific examples of the locations. Since the data is the same, the extra data about the museums will have to be included in the report due to the new attributes in the unified database.

**Table 5: tblHistory**

**The following was the design of this table in my database:**



**Purpose:** This table was intended to store the historical locations of the works.

This table will primarily be moved to the Works\_Locations of the unified database. The DatabaseEntryLocation and DatabaseEntryAddress will have to be added manually here since they did not exist in our database.

Hisalphadesig, hisnumdesig- these two identifiers of the primary key of a work can simply transferred over to the Works\_Locations table of the new database

hisStartDate, hisEndDate- These can be moved over to the wol-WorkArrivalTime, and wol-WorkDepartureTime, but like the others, the new attribute is a timestamp so the times may need to be adjusted as necessary.

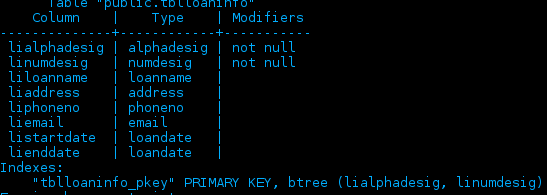
hisLocation- Our museum used a character identifier for locations, so this data will have to be substituted by hand with the actual location name from the locations table.

All other attributes that exist in the new database design will be added from their foreign tables since they are data that are added elsewhere.

**Testing:** To test this we will create a report that will generate the information of specific works locations in both the original and new tables. The transferred data must match.

**Table 6: tblLoanInfo**

**The following was the design of this table in my database:**

**Purpose**: This table in my original database stored the loan info of works and when and who they were loaned to.

**The following changes will be made:**

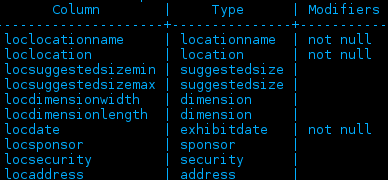
Lialphadesig, linumdesig-these will already be in the works table of the new database, so they will not be moved forward.

Liaddress, liphoneno- this are not used as part of the new database, so they will remain in the original database but they will not move forward.

The other data will remain in the old database, but will be moved to other tables as necessary.

**Tests**: We will produce a report that queries the proper tables to get the information on works where the wor-DatabaseEntryLocation is our museum, and compare the results of that report to the results of a report of the loans from the old database.

**Table 7: tblLocation**

This was the structure of the locations table within the database of our museum:

**Purpose:** This table in our database stored the information on locations both within our museum and locations that were visited by.

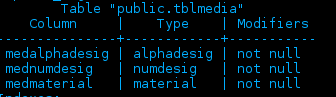
**The following changes will be made:**

Loclocationname, locsuggestedsizemin, locsuggestedsizeman, locdimensionwidth, and locdimension length can easily be moved to the new Locations table, with additional manual input of data where necessary, such as for the museum name.

The locations within this table that are not in our museum, we have agreed to move all appropriate data to the Travelling\_Destinations table of the new database. This will have to be done by manually going over the table and moving the proper location to the proper table

**Testing**: Testing this migration will be a tad more complex, but it will consist of two different tests, we will produce reports of the appropriate locations from both the communal database Locations table, and the Travelling\_Destinations table. We will then compare the transferred data to the data in the original database.

**Table 8: tblmedia**

The following were the attributes contained in this table in our museum’s database:

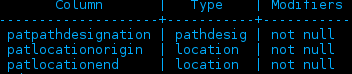
**Purpose:** This table stored the medium that the works in the museum were made out of, basically the same function as the Work\_Mediums table in the database.

**The Following changes will be made**:

Medalphadesig, mednumdesig, medmaterial all correspond to wme-WorkCharID, wme-WorkNumID, and wme-WorkMedium in the Work\_Medium table. The additional fields in the new table will be filled in manually with the proper data as necessary, such as museum location and museum address.

**Testing:** This will consist of creating a report that will pull the identifier of a work and the name of the material that the work is made of. We can then compare the two reports and if they are the same we know that the transfer was conducted successfully.

**Table 9: tblpaths**

The following were the attributes contained in this table of our museum’s database:

**Purpose**: This table was responsible for storing the information about the connections that existed between rooms in our own museum.

**The following changes will be made:**

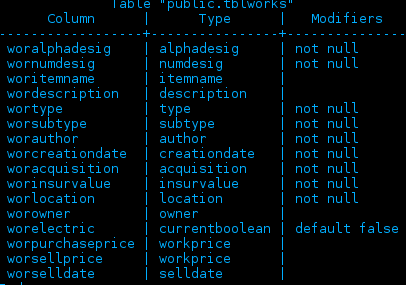
Patpathdesignation was a numerary indicator of path, it was redundant data and since it is not used and also redundant there is no need for it to be carried forward.

Patlocationorigin and patlocationend correspond to lcn-LocationNameOrigin and lcn-LocationNameEnd of the Doors table and thus can be moved forward by manually referring to the actual names of the locations.

**Testing**: To test the successful transfer of data, we will produce a report by querying the new database for the doors that correspond to our museum. We will then cross-reference that information with our original data to ensure that everything transferred successfully.

**Table 10: tblworks**

The following was the design for the works table in our museum:



**Purpose:** This table contained the data on the museums works, as well as other various data about them.

**The following changes will be made:**

Woralphadesig, wornumdesig- these attributes will be the wor-CharID, and wor-NumID in the new works table in the unified database. Same data type so simply moving them over should not be a problem.

Wor-itemname, wordescription- these can be moved as well to the Works table as wor-WorkName and wor-WorkDescription, a simple copy-over and they will move without issue.

Wortype, worsubtype- these classified the types and subtypes of my works, but the unified database has a new unified system of categories, so these attributes will not be moved forward and instead I will have to reclassify every work manually.

Worauthor, worcreationdate-these can be moved to the Works table as the attributes wor-WorkCreator and wor-CreationDate. The datatype of the date is different so the dates may need to be adjusted manually to ensure that there is no loss of data

Worinsurevalue-this will be moved manually to the Works\_Insurance table and placed with the appropriate dates into the table.

worLocation-This was old data that was made redundant by the history table in our database, so there is no need to move it forward.

worowner-This data will have to be manually moved to the Owners table and the dates looked up and put in the right place.

worelectric-This attribute was the custom one produced for assignment 3, and it has been amalgamated into the physical properties of the new database structure. The works that had it will have to be manually adjusted, but none of this data needs to be carried forward.

Worpurchaseprice, worselldate, worsellprice-The worselldate will be added as a change in ownership in the Works\_Owners table. Worselldate will be used as a basis, and I will adjust any data as necessary. Worsellprice is not included in the new database and will not be moved forward.

**Testing**: To test the successful transfer of data the following must be done:

-Create a report on the three works that my museum has sold from the WorksOwners table and compare that to worselldate from the old database.

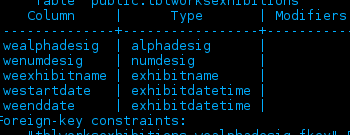
-Create a report on the works that feature electric in their property and make sure that they line up with the works that have the electric attribute in the old database.

-Query the Works\_Owners table and the works table to ensure that the works and their owners current and previous line up properly and the data is the same.

-Query the works\_insurance table and the Works table and do the same to the old tblworks and compare the two results to ensure that they are the same

**Table 11: tblworksexhibitions**

The following is the structure of the table in our database:



**Purpose:** This table was responsible for storing the information on the works that belonged to each exhibit.

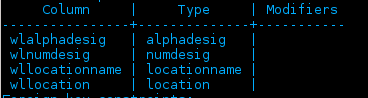
**The following changes will be made:**

Wealphadesig, wenumdesig-This data will already exist in the works table so there is no need to retransfer it again as it would just cause errors.

Weexhibitname, weexhibitstartdate, weexhibitenddate- These attributes will have already been transferred (already exist in the exhibits table) so I can rebuild the data here in the new database within the references by using the proper manual insert statements.

**Testing**: Once we have rebuilt the exhibits and works together in the new table, we will create a report and compare the items of the first exhibition that our museum had, and create the same report in the original database and compare the two reports.

**Table 12: tblworkslocations**

The design of this table in my database was as followed

**Purpose:** This table was originally designed to replace the tblhistory table, but was not fully implemented, and thus contains no useful data of any kind

**The following changes will be made**:

No changes need to be made as the table is functionally similar to tblhistory and it contains no data. Nothing needs to be carried forward.

**Testing**: Nothing was moved so nothing needs to be tested.

**Additional Notes:**

-In the museums table I will manually add the data for our museums.

-Each change to a work, throughout all the tables will have to be added manually to the works\_transactions table.

Individual Report (Part D)

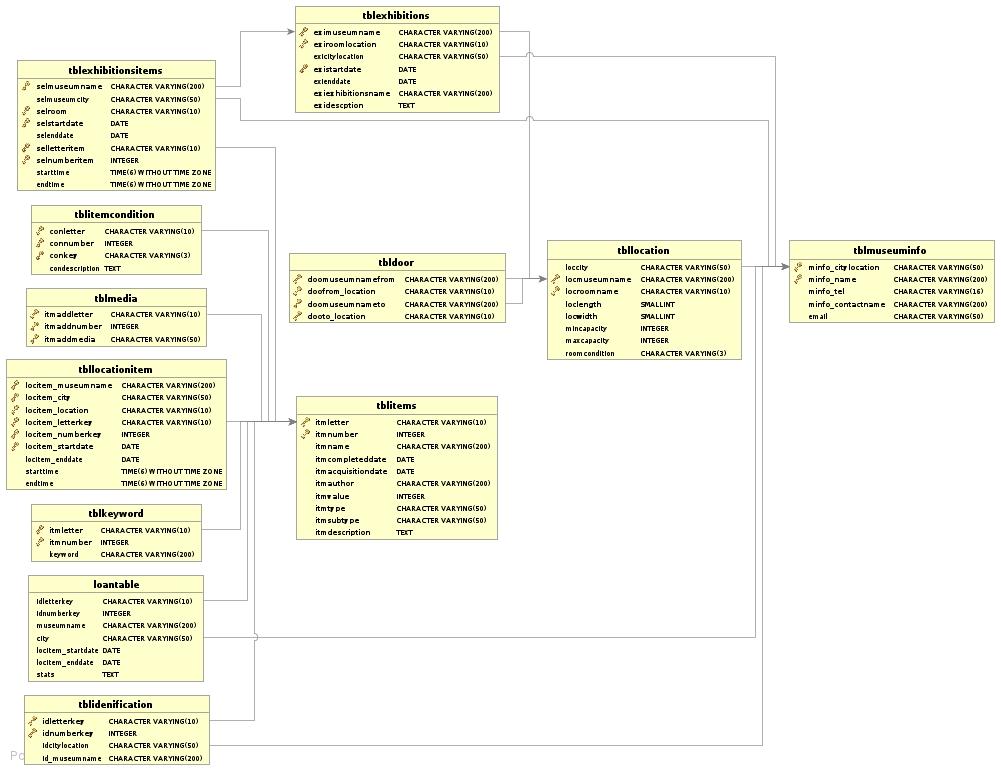
## Name: Zang, JiaWei

## NSID: 11148860

Team Database Structure



My Database Structure



### Difference between database

Core functionalities

Works

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| wor-WorkCharID | My database has those but in different name | NONE | NONE |
| wor-WorkNumID | My database has those but in different name | NONE | NONE |
| wor-DatabaseEntryLocation | My database Does not have it | When I insert works information into team database on this table from query museum information, I need to added the location information | Query this data from my information and check does it have database entry location which is same with my museum name. |
| wor-DatabaseEntryLocationAddress | My database Does not have it | When I insert works information into team database on this table, I need to added the location address information | Query this data from this table and check does it have database entry location which is same with my museum address. |
| wor-description | My database has those but in different name | NONE | NONE |
| wor-PhysicalProperty | My database has those but in different name.  However, the standard is different | Need to change my type fit into the physical property standard | Query this column and check this result are all in the standard.  It should be painting or sculpture. |
| wor-workClassification | My database has those but in different name.  However, the standard is different | Need to change my type fit into the Classification  standard | Query this column and check this result are all in the classifications standard |
| wor-DateCreated | My database has those but in different name | NONE | NONE |
| wor-InsuranceValue | My database has those but in different name | NONE | NONE |
| wor-WorkBorrowable | Does not have this attributes. | When I insert my data, I need to check the loanwork table which the table contain all the loan works | Check this out the loanable items are all in the loanwork table in my original database |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing or wrong value in the query result, make sure they are all there, and make sure they are all valid by evaluating the result. | | |

WorkMediums

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| wme-WorkCharID references Works (WorkCharID) | My database has those but in different name | NONE | NONE |
| wme-WorkNumID references Works (WorkNumID) | My database has those but in different name | NONE | NONE |
| wme-WorkDatabaseEntryLocation references Works (DatabaseEntryLocation) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location information | Query this table to check this table have item with char and number have same name with my museum name |
| wme-DatabaseEntryLocationAddress  references Works (DatabaseEntryLocationAddress)  wme-Medium | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location address information | Query this table to check this table have item with char and number have save with my museum address |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

WorkThemes

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| wth-WorkCharID references Works (WorkCharID) | My database has those but in different name | NONE | NONE |
| wth-WorkNumID references Works (WorkNumID) | My database has those but in different name | NONE | NONE |
| wth-WorkDatabaseEntryLocation references Works (DatabaseEntryLocation) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location information | Query this date to check this table have item which are belong to me with char and number have museum name |
| wth-DatabaseEntryLocationAddress  references Works (DatabaseEntryLocationAddress) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location address information | Query this data to check this table have item which belong to me with char and number have museum address |
| wth-Theme | Does not have this attributes. | Need to copy what I have in works table and insert into this table. | Query the item belonged to me, and make sure that my original themes are changed into standard theme. |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

WorkLocations

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| won-WorkCharID references Works (WorkCharID) | My database has those but in different name | NONE | NONE |
| wol-WorkDepartureTime | My database has those but in different name | NONE | NONE |
| won-WorkNumID references Works (WorkNumID) | My database has those but in different name | NONE | NONE |
| won-WorkDatabaseEntryLocation references Works (WorkDatabaseEntryLocation) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location information | Query this data to check this table have item which are belong to me with char and number have museum name |
| won-DatabaseEntryLocationAddress  references Works | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location address information | Query this data to check this table have item which belong to me with char and number have museum address |
| wol-WorkArrivalTime | My database has those but in different name | NONE | NONE |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

ExhibitionWorks

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| exw-ExhibitName references Exhibitions (ExhibitName) | My database has those but in different name | NONE | NONE |
| exw-MuseumName references Exhibitions (MuseumName) | My database has those but in different name | NONE | NONE |
| exw-Museumlocation references  Exhibitions (MuseumLocation) | My database has those but in different name | NONE | NONE |
| exw-ExhibitStartDate references  Exhibitions (ExhibitStartDate) | My database has those but in different name | NONE | NONE |
| exw-WorkCharID references Works (WorkCharID) | My database has those but in different name | NONE | NONE |
| exw-WorkNumID references Works (WorkNumID) | My database has those but in different name | NONE | NONE |
| exw-WorkDatabaseEntryLocation references Works (DatabaseEntryLocation) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location information | Query this data to check this table have item which are belong to me with char and number have museum name |
| exw-DatabaseEntryLocationAddress  references Works (DatabaseEntryLocationAddress) | Does not have this attributes. | When I insert works information into team database on this table, I need to added the location address information | Query this data to check this table have item which belong to me with char and number have museum address |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Museums

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| mus-museumName | My database has those but in different name | NONE | NONE |
| mus-museumLocation | My database has those but in different name | NONE | NONE |
| mus-phoneNumber | My database has those but in different name | NONE | NONE |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Locations

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| loc-LocationName | My database has those but in different name | NONE | NONE |
| loc-MuseumName | My database has those but in different name | NONE | NONE |
| loc-MuseumLocation | My database has those but in different name | NONE | NONE |
| loc-LocationSuggestedCapacityMin | My database has those but in different name | NONE | NONE |
| loc-LocationWidth | My database has those but in different name | NONE | NONE |
| loc-LocationLength | My database has those but in different name | NONE | NONE |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Door

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| dor-LocationNameOrigin | My database has those but in different name | NONE | NONE |
| dor-LocationNameEnd | My database has those but in different name | NONE | NONE |
| dor-MuseumName references Exhibitions (MuseumName) | My database has those but in different name | NONE | NONE |
| dor-Museumlocation references  Exhibitions (MuseumLocation) | My database has those but in different name | NONE | NONE |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

ExhibitionLocations

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| elo-ExhibitName references Exhibitions (ExhibitName) | Does not have this attributes.  Because I stored the information about exhibition location in exhibition table | My exhibition location information is stored in exhibition table, mean I need move the exhibition name information into this table, when I insert my information in team database | Query from my original database, and check the exhibition table check they are same in exhibition location table |
| elo-MuseumName references Exhibitions (MuseumName) | Does not have this attributes.  Because I stored the information about exhibition location in exhibition table | My exhibition location information is stored in exhibition table, mean I need move the museum name information into this table, when I insert my information in team database | Query form the museum name table to check this is same with the new team database.  And query this and check this with query from my exhibition table to get the name and make sure they are same |
| elo-Museumlocation references  Exhibitions (MuseumLocation) | Does not have this attributes.  Because I stored the information about exhibition location in exhibition table | My exhibition location information is stored in exhibition table, which is mean I need move the museum location information into this table, when I insert my information in team database | Query form the museum location table to check this is same with the new team database in museumlocation |
| elo-ExhibitStartDate references  Exhibitions (ExhibitStartDate) | Does not have this attributes.  Because I stored the information about exhibition location in exhibition table | My exhibition location information is stored in exhibition table, which is mean I need move the start date information into this table, when I insert my information in team database | Query from the exhibition table with the same format in new team database, and check this is same with the query result from start date exhibition table |
| elo-LocationName references Locations(LocationName) | Does not have this attributes.  Because I stored the information about exhibition location in exhibition table | My exhibition location information is stored in exhibition table, which is mean I need move the location name information into this table, when I insert my information in team database | Query the data which what are mine, and check this is same with the query the data which are in exhibition table. |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Exhibitions

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| exh-ExhibitionName | My database has those but in different name | NONE | NONE |
| exh-MuseumName | My database has those but in different name | NONE | NONE |
| exh-Museumlocation | My database has those but in different name | NONE | NONE |
| exh-ExhibitStartDate | My database has those but in different name | NONE | NONE |
| exh-ExhibitEndDate | My database has those but in different name | NONE | NONE |
| exh-Description | My database has those but in different name | NONE | NONE |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Transactions



Works

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| wor-WorkCharID | During database has those but in different name | NONE | NONE |
| wor-WorkNumID | My database has those but in different name | NONE | NONE |
| wor-DatabaseEntryLocation | My database Does not have it | When I insert works information into team database on this table, I need to added the location information | Query this information from this information and check does it have database entry location which is my museum name. |
| wor-DatabaseEntryLocationAddress | My database Does not have it | When I insert works information into team database on this table, I need to added the location address information | Query this information from this information and check does it have database entry location which is my museum address. |
| wor-description | My database has those but in different name | NONE | NONE |
| wor-PhysicalProperty | My database has those but in different name.  However the standard is different | I need change my type fit into the physical property standard | Query this column and check this result are all in the standard.  It should be painting or sculpture. |
| wor-workClassification | My database has those but in different name.  However the standard is different | I need change my type fit into the Classification  standard | Query this column and check this result are all in the classifications standard |
| wor-DateCreated | My database has those but in different name | NONE | NONE |
| wor-InsuranceValue | My database has those but in different name | NONE | NONE |
| wor-WorkBorrowable | Does not have this attributes. | When I insert my data, I need to check the loanwork table which the table contain all the loan works | Check this out the loanable items are all in the loanwork table in my original database |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

WorkOwners

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| won-WorkCharID references Works (WorkCharID) | My database has those but in different name. | NONE | NONE |
| won-WorkNumID references Works (WorkNumID) | My database has those but in different name. | NONE | NONE |
| won-WorkDatabaseEntryLocation references Works (WorkDatabaseEntryLocation) | I don’t have this attribute, therefore, I need to create this attribute | When I insert works information into team database on this table, I need to added the location information | Query this table, and check out the museum name information about the item, check out the is this my museum name |
| won-DatabaseEntryLocationAddress  references Works (DatabaseEntryLocationAddress) | I don’t have this attribute, therefore, I need to create this attribute | When I insert works information into team database on this table, I need to added the location address information | Query this table, and check out the museum address information about the item, check out the is this my museum address |
| won-ownerName references Owners (ownerName) | I don’t have this attribute, therefore, I need to create this attribute | Add new attribute change the institution name into this attribute | Query this modification attributes and check out this is valid name. |
| won-ownerEmail references Owners (ownerEmail) | I don’t have this attribute, therefore, I need to create this attribute | Create new data information about the email | Query this data which is new data that I originally don’t have this information, and make sure those are email address |
| won-OwnershipStartDate | I don’t have this attribute, therefore, I need to create this attribute | Create new date into new table when I insert into this team database | Query this data which is new data, that I originally don’t have this information, and make sure those are start date |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

Owner

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| own-ownerName | I don’t have this table, but I save this owner information as institutions | Needed to change to valid name when we insert into team database | Query this data, and compare with the owner result in works owner table. |
| own-ownerEmail | I don’t have this table, I save this owner information as institutions with name and address | Needed to create email information when I insert this data into team database | Query this data, and compare with the owner result in works owner table. |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

WorkTransactions

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| wtr-WorkCharID references Works (WorkCharID) | My table to contain this information is in item condition table | Needed to select this information from condition table and insert into transactions table. | Query this form my item table and make sure those are my work |
| wtr-WorkNumID references Works (WorkNumID) | My table to contain this information is in item condition table | Needed to select this information from condition table and insert into transactions table. | Query this form my item table and make sure those are my work |
| wtr-WorkDatabaseEntryLocation references Works (WorkDatabaseEntryLocation) | This table does not have this attribute | When I insert works information into team database on this table, I need to added the location information | Query this from mine in museum information table museum name is same value, and make sure thosesure those are same |
| wtr-DatabaseEntryLocationAddress references Works (DatabaseEntryLocationAddress) | This table does not have this attribute | When I insert works information into team database on this table, I need to added the location address information | Query this from my museum table which called museum address is same value, and make sure those are same |
| wtr-TransactionType | My database has those but in different name. | NONE | NONE |
| wtr-TransactionTime | My database has those but in different name. And this is in loan table | Need to query this from loantable to get the start date into this table. | Query this data and make sure they are all same date with loan table in my original table |
| wtr-Enddate | My has those but it attribute in item table, and item condition table | Need to query this from location item table and item condition which is borrowed items to get the end date of the borrow items end date | Query this data and query the tblitemcondition table to find the borrowed items and find those item in location item table and find the end date to make sure those are the same. |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

## 

## Traveling Exhibition

TravellingExhibitions

|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| tre-ExhibitName references Exhibitions (ExhibitName) | I don’ t have this table, because I stored this traveling exhibition information into the exhibition table | Query this from exhibition table which is traveling exhibition and insert into team database | Query this data and compare the traveling exhibition which in the exhibition table, and make sure they are same |
| tre-Security | I am missing this information, therefore, I need to add security name.  Because I save this information in museum information take | Add security name into the team database by make the name up. | Query this information and compare with the name in the table of museum information table, and make sure they don’t exist in the table, because my original database does not have those name information about security |
| tre-MuseumName references Exhibitions(MuseumName) | I don’t have this in the table but in different name in different table. | NONE | NONE |
| tre-MuseumLocation references  Exhibitions(MuseumLocation) | I don’t have this in the table but in different name in different table. | Query this from exhibition table which is traveling exhibition and insert into team database | Query this data and compare with the exhibition table which exhibition are traveling exhibition, and make sure they are same. |
| tre-ExhibitStartDateOriginal references Exhibitions(ExhibitStartDate) | I don’t have this in the table but in different name in different table. | Query this from exhibition table which is traveling exhibition and insert into team database | Query this data and compare with the start date in the exhibition table (which are traveling exhibition), and make sure they are same |
| tre-ExhibitDepartureDate | I don’t have this in the table but in different name in different table. | Query this from exhibition table which is traveling exhibition and insert into team database | Query this data and compare with the end date in the exhibition table (which are traveling exhibition), and make sure they are same |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

TravellingExhibitionSponsors

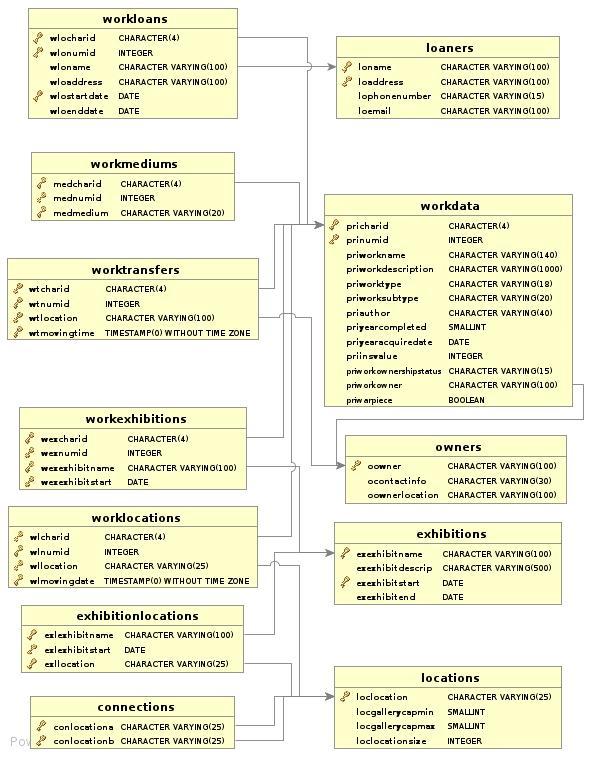
|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| spo-ExhibitName references TravellingExhibitions(ExhibitName) | I don’t have this table in my database design, but this attribute is in exhibition table | Needed to query from exhibition table which is traveling, and insert this into team database. | Query this data and compare this with exhibition table which are traveling exhibition and make sure they are all same |
| spo-MuseumName references TravellingExhibitions (MuseumName) | I don’t have this table in my database design, but this attribute is in exhibition table | Needed to query from exhibition table which is traveling, and insert this into team database. | Query this data and compare this with exhibition table which are traveling exhibition and make sure they are all same |
| spo-MuseumLocation references TravellingExhibitions (MuseumLocation) | I don’t have this table in my database design, but this attribute is in exhibition table | Needed to query from exhibition table which is traveling, and insert this into team database. | Query this data and compare this with exhibition table which are traveling exhibition and make sure they are all same |
| spo-ExhibitStartDateOriginal references TravellingExhibitions (ExhibitStartDateOriginal) | I don’t have this table in my database design, but this attribute is in exhibition table | Needed to query from exhibition table which is traveling, and insert this into team database. | Query this data and compare this with exhibition table which are traveling exhibition and make sure they are all same |
| spo-ExhibitDepartureDate references TravellingExhibitions (ExhibitDepartureDate) | I don’t have this table in my database design, but this attribute is in exhibition table | Needed to query from exhibition table which is traveling, and insert this into team database. | Query this data and compare this with exhibition table which are traveling exhibition and make sure they are all same |
| spo-sponsorName | I don’t have this attribute | Needed to create a value for information about sponsor name. | Query this data and make sure those are new data, because I don’t have this.  Or compare with museum information table, because this table contain contractor information. |
| spo-SponsorAmount | I don’t have this attribute | Needed to calculate this value when I insert the value. | Query this data and make sure those are match the insurance values are greater than the total value of insurance |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

TravellingExhibitionDestinations

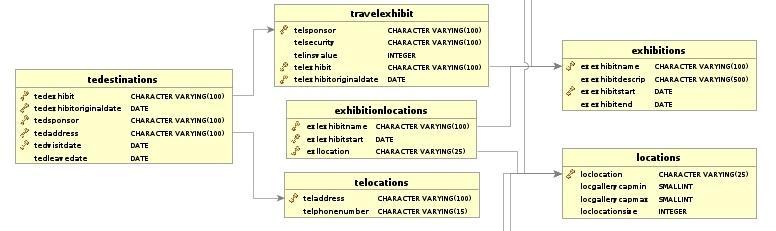
|  |  |  |  |
| --- | --- | --- | --- |
| Team Database | My Database | Modifications | Modifications Evaluation |
| spo-ExhibitName references TravellingExhibitions(ExhibitName) | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the exhibition name information from exhibition traveling, and insert into the team database | Query this data and compare with the exhibition table which are traveling. And make sure they are same |
| spo-MuseumName references TravellingExhibitions (MuseumName) | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the museum name information from exhibition which is traveling, and insert into the team database | Query the museum name and compare with query the museum name which are traveling exhibition, and make sure they are same. |
| spo-MuseumLocation references TravellingExhibitions (MuseumLocation) | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the museum location information from exhibition which is traveling, and insert into the team database | Query this data and compare this information with museum information table and |
| spo-ExhibitStartDateOriginal references TravellingExhibitions (ExhibitStartDateOriginal) | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the exhibition end date information from exhibition which is traveling, and insert into the team database | Query this data and compare with the exhibition table which are traveling exhibition and make sure they are same |
| spo-ExhibitDepartureDate references TravellingExhibitions (ExhibitDepartureDate) | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the exhibition end date information from exhibition which is traveling, and insert into the team database | Query this data and compare with the exhibition table which are traveling exhibition and make sure they are same |
| ted-LocationName | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling | Query the exhibition location information from exhibition which is traveling, and insert into the team database | Query this data and query the original data and check the location to be same. |
| ted-ExhibitArrivalDate | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling.  And I separate the one exhibition into different exhibition and stored as different exhibition. | Query the exhibition start in exhibition table which is traveling information and insert into the team database | Query this value and compare the exhibitions are exist in the exhibition table which are traveling exhibitions |
| ted-DestinationPhoneNumber | I don’t have this think that exhibition table, because I stored this information in the museum information table, which I museum information could also contain the information about the museum contractor of traveling | Query the information from the information in table museum information table | Query this data and compare the query result in museum information table, and make sure they are same |
| ted-ExhibitDepartureDateLocation | I don’t have this table because I’m using exhibition table because exhibition table contain the information about exhibition information including traveling.  And I separate the one exhibition into different exhibition and stored as different exhibition. | Query the exhibition end date in exhibition table which is traveling information and insert into the team database | Query this value and compare the exhibitions are exist in the exhibition table which are traveling exhibitions |
| How to transformations | During modification of the attributes, and evaluate the result of modification, then Select all the information which I need and save as view, and insert into the team database. | | |
| Check the new database is correct | Query all attributes in this table, and check the order and there is no missing value in the query result, make sure they are all there | | |

**Database Conversion Report – Warren Fehr**

**(PART D)**

A chart showing my db design as of the end of Assignment 3 for reference

Another chart, showing how I handled TravellingExhibitions:



**Overall Changes**

We have made a lot of tiny changes to prefixes and names to standardize the database structure between groups. For example, I used “pri” as the prefix for my WorksData table, but our group decided on the prefix “wor” for the equivalent table in the merged database. All of these will be changed to the standards found the previous report unless stated in the individual table report.

**New Table (Museums)**

I added the table Museums to my database. It will have all the attributes as stated in the general report: the name of the museum, the location of the museum (which are both primary keys), and the phone number for the museum. I will also need to add the relevant domains to make the table as necessary. This will only have one entry, for my own museum, which will be a primary key for all works,exhibitions, and locations.

**Changes to WorksData Table (Now called Works), & creation of WorksInsurers**

Beyond renaming this table to Works, I will need to make changes to several attributes. First, I will remove “priwarpiece” attribute, because its information will be moved to the WorkThemes table (as told in the WorkThemes section below). I will also remove the “priownership” and “priowner” attributes, and move all that data to the Owners, Work Owners and and WorkTransactions tables (which will be described in greater depth in that part of the report). We will add the “wor-classification” and “wor-physicalproperty” attributes as well. All works in my museum are paintings, so all the PhysicalProperty enteries will be “Painting”. The classifications will be set to whatever the type attribute was originally(Such as “Landscape” or “Portrait”). After that, we can remove the unnecessary “priworktype” and “prisubtype” attributes. We will need to add the foreign/primary keys “DatabaseEntryLocation” and “DatabaseEntryLocationAddress” for all entries, which will be the ones we entered into the Museum table. The insurance values can be removed from here and put into the new WorksInsurance table.  All other attributes just need to be renamed to the ones we have specified in the Museums table for our museum.

**Changes to WorkMediums**

All I need to do are add the foreign keys from the Museums table and touch up attribute names as needed.

**Changes to Exhibitions Table**

All I need to do are add the foreign keys from the Museums table and touch up attribute names as needed.

**Changes to Locations Table**

After adding the foreign keys from Museums, I also will have to change the “loclocationsize”, which represents the total square meters of the locations into the two attributes representing length and width and create domains for them. After that, all I need to do is to just touch up attribute names.

**Changes to Connections Table (Now called Doors)**

All I need to do are add the new foreign keys from the Locations(Museums) table and touch up attribute names as needed.

**Changes to ExhibitionLocations and WorkExhibitions (Now called ExhibitionWorks)**

After changing WorkExhibitions to ExhibitionWorks and adding all the new foreignkeys in their parent tables, all I need to do to these two tables is touch up names as needed.

**Changes to WorkLocations**

In the original database, I made the assumption that because moving works between locations in the same museum is basically instantaneous, I only needed an arrival timestamp for works moved between locations. I'll need to add an attribute for departure for that in my table. The values for it will be the time it arrived at the other location. For example, if I moved an item from Location A to Location B, the departure time of the entry for A would be the arrival time for B. This will need to be touched up for all WorkLocation transactions. After that, all I need to do is add the new foreign keys in the parent tables and touch up names.

**New Table (WorkThemes)**

I need to add to add a table for WorkThemes to my museum. Considering my museum specializes in American art, I can put an entry in here for all works with theme “American”. I can also put an entry for all my pieces that are marked as war pieces by putting an entry in here for all of them with theme “War”. I will also add a few themes for each piece based on what they are based on their description to provide the data needed for themes to be useful in the new database.

**How to handle work transactions**

A lot of the information I had for handling transactions with works was scattered throughout the database in several tables (Owners, Loaners, WorkTransfers, WorkLoans). We can compile all these into the new Owners, WorkTransactions, and WorkOwners without too much trouble. All data from my Owners and Loaners tables can be compiled into the new Owners table. WorkLoans data can be transferred over to the new standard by making an entry in the WorkTransactions table (an entry showing a tranasction with the trasaction type  “LOANED” with start and end dates) and a entry into the WorkOwners table (showing the ownership transfer to the new party, as well as start and end dates of it). WorkTransfers, holding data about sold and bought works, can have its data transferred in the same manner. This would be by making an entry in the works WorkTransactionsTable with “BOUGHT” or “SOLD” as the transactiontype and appropraite start and end dates, and an entry in the workOwners table as well. This also helps to handle the inital acquire date of the work and the issue of owners, which was handled by the Works table before this. After this, all that really needs to be done is changing attribute names and adding the museum foreign keys.

**Changes to Travelling Exhibitions & Adding TravellingExhibitSponsors**

I can offload all of the sponsor data I have in this table to the new TravellingExhibitionSponsors table. I have no idea what the sponsors donated before this point in time for my data (and I shouldn't be making this amount up), so all these new sponsor entries will have the SponsorAmount be NULL. After that all I need to do are add the new foreign keys from the Exhibitions table and touch up attribute names as needed.

**Changes to TEDestinations (Now called TravellingExhibitionDestinations) &**

**TELocations**

The new database schema doesn't have a tedestinations table, but instead covers all that information in the TravellingExhibitionDestinations instead. I can transfer over the data from TELocations over to TravellingExhibitionDestinations without any problem. Other then that, just routine renaming, as well as carrying over the new foreign keys from TravellingExhibitions.

**Testing & Verification**

Beyond the data simply fitting into the new database schema, I'll need to run some tests and checks to verify it will work properly in the merged database. I've devised a major and a minor test to help verify the data is up to standard.

**Full Transactional Integrity**

I will devise a query to check if each work has full transactional integrity. What this means is that there should be an unbroken record of what the location and who the owner are for each work up until the most recent recorded activity. This is very important for our client because if there is missing information, we may have troubles if something happens to a work and we need to collect insurance on it. For a quick example, here's the sequence of events that should happen for a typical work we've bought:

* Work is bought (Entry in WorkTransactions for Sale). The item is also recorded into the Works Table at this time under the Museum that bought it, as well as an entry in the WorksOwners table (the EndTime should be NULL, because we don't know when we will stop owning the work). The WorkOwners and WorkTransactions times should match.
* An entry is made in the WorksLocation table is made when it arrives at the Museum and is placed into Storage.
* At some point, it will be entered into an Exhibition. When it is moved, the workDeparture time from the Storage entry should match the arrival time at the exhibit location. If we decide that moving times aren't instantaneous for the museum, we would decide on how long a move could be at most, and if we have a period longer then that, it would fail.
* After the Exhibition is done, it is moved back to storage. We would apply the same checks here.

We would continue like this until we hit a discrepency or we determine there is a clear chain of events. There are a lot of possibilities to cover (Loans, Sales, Donated Works, Damaged Works), but in all those cases, the new schema is able to describe the clean chain of ownership and location, and I should be able to confirm that all my data has no holes once I have placed it into it. If there is any works that fail the test, I will have to take a look and either check if it is a mere transcribing error because of moving the data, or a logic error (which may indicate a problem with the schema) and fix it accordingly. For transcribing errors it is simple, but logic errors can be used to point out if we haven't covered our cases properly.

**Themes Checking**

For themes to be useful for clients and visitors, we need to have quite a few themes for each work. All works will have the “American” theme, as my museum is an American Art, and a few works were identified as “war” pieces in the process of doing Assignment 3. I'll need to check that each piece has at least 3 themes associated with it, and I'll need to devise new ones to describe the work based on what information I have.

CMPT 355 REPORT

Part D (Revised)& PART E

Mohamed Kassem

NSID: mgk787

Student No.: 11129292

**Introduction**

In this report I introduce the architecture for my individual assigned museum database. The report focuses on the information contained in my museum databases and how this information can be transferred to the new team database with the least amount of information losses. It’s not meant to provide a comprehensive **technical explanation** about everything in my individual database or how it can be transferred to fit the new team database design using SQL.

**Report Outlines**

**A. Museum’s Works Classification (Part D revised)**

**B. Transferring data to the new team database (Part D revised)**

§ Changes needed to transform my database and its data content to the new team database?

§ How information will be migrated to the new database without being lost.

**C. Evaluating and testing the correctness of the new database. (Part D revised)**

**A. Museum’s Works Classification (Part D revised)**

This section focuses on the differences between the classification of works in my individual database and the new team database, and shall not discuss in details any of databases tables, attributes or domains. The main focus will be on how my works will be classified under the new team design, and how my works’ classification can fit under broader categories in the new design.

The Frick Collection (i.e. my personal assigned museum) is a museum of art work. It has a collection of works that can be generally classified in to one of the following categories:

1. Painting

2. Textile

3. Metal works

4. Clocks

5. Sculpture

These categories are what I refer in my database as **work’s types**. The word ‘types’ is not indicative enough to how the work is exactly classified, and that’s why in the new team database works will be classified according to different attributes. Each of the above mentioned work’s types are also classified according to my database into one or more of types.

Below is a list of classifications that any work in my database can categorized under

**I.** **Painting: can be either**

i. Portrait

ii. Landscape

iii. Genre: a style of painting depicting scenes from ordinary life, especially domestic situations

iv. Still-Life: is a work of art depicting mostly inanimate subject matter, typically commonplace objects which may be either natural (food, flowers, dead animals, plants, rocks, or shells) or man-made (drinking glasses, books, vases, jewelry, coins, pipes, and so on)

v. History: Painting that describes historical and religious events or scenes

**II.** **Textile: can be either**

I. Carpets

II. Tapestry

**III.** **Metalwork: can be only (no other types of metal works in my database)**

i. Jewellery

**IV.** **Clocks: can be either**

i. Bracket

ii. Mantel

iii. Long-case

iv. Pendant

v. Astronomical

**V.** **Sculpture: can be either**

i. **Round:** sculpture in the round, free-standing sculpture, such as statues, not attached to any other surface

ii. **Relief:** sculptures, which are at least partly attached to a background surface.

On the other side, the new team database classify works based on physical property (i.e. physical characteristics of the work, example a painting is a physical property for an item), sub-class of the physical property, geographic region (not to be confused with the country of origin), related field of science (if any), and works themes

**Themes** are conceptual qualities that can describe the related work, the idea of work’s theme comes from literature in theme is the central message or insight that is conveyed through a piece of writing. This same definition is applied to works in the new team database.

**Physical property** describes the physical characteristic of the work, what exactly is the work. Example for physical properties are: Metal Work, Furniture, Carving, Ceramics, Painting, Textile, Electrical Work, or Mechanical Work are all examples that can fit under the physical property of the work.

**Sub-class of the physical property (**it’s called **WorkClassification** in the **NS\_work table).** This is a second level classification (granularity of classification) for the physical property of the work. Examples for a sub-class of the physical property are a painting being either a portrait, genre or landscape; textile maybe a carpet, cloths or tapestry etc.). This classification was chosen in order to maintain information on subtypes in some of our team museums’ database.

**Geographic region** (it’s stored in **worWorkGeographicRegionOfOrigin** in the **NS\_work** table**).**  The new team database differentiates between the concept of country of origin and the geographic region that piece of work can be related to. An example for that is a piece of an art work that is created by a French artist, in China (country of origin) and classified under Asian artwork. Another example to clarify this point is a piece of an artwork created by an American artist, in Italy (country of origin) and classified under Mediterranean artwork.

**Related Field of Science** (it’s stored in **worWorkFieldOfScience** attribute in **NS\_Works** table). Under this classification, works are classified based on their related field of science. This classification is most suitable for science and technology related works and inventions, in which a work can be classified to be related to science field like Chemistry, Physics, Biology, and etc.

From the above discussion, one can see that the classification of works in my individual database, can fit under the new works’ classification in the team database. Information about works’ types and subtypes in my database are mapped with no changes to be under **Physical property** and **Sub-class of the physical property** in the new team **database**. My database doesn’t have a direct information about how works can be classified by themes, related science field or by geographic region. These new classification fields (i.e. attributes) are important information that need to be collected by my museum staff and admin to fit under the new database classification.

A careful search in the the description of works trying to look for important keywords, that could identify one or more work’s themes or identify a related science field has resulted in the following. The majority of my works can be classified with themes like Spirituality, Religions, Christianity, Wars and European life style, while all of them can fit under being related to Social Sciences, History and Fine Arts.

**B. Transferring data into the new team database (Part D revised)**

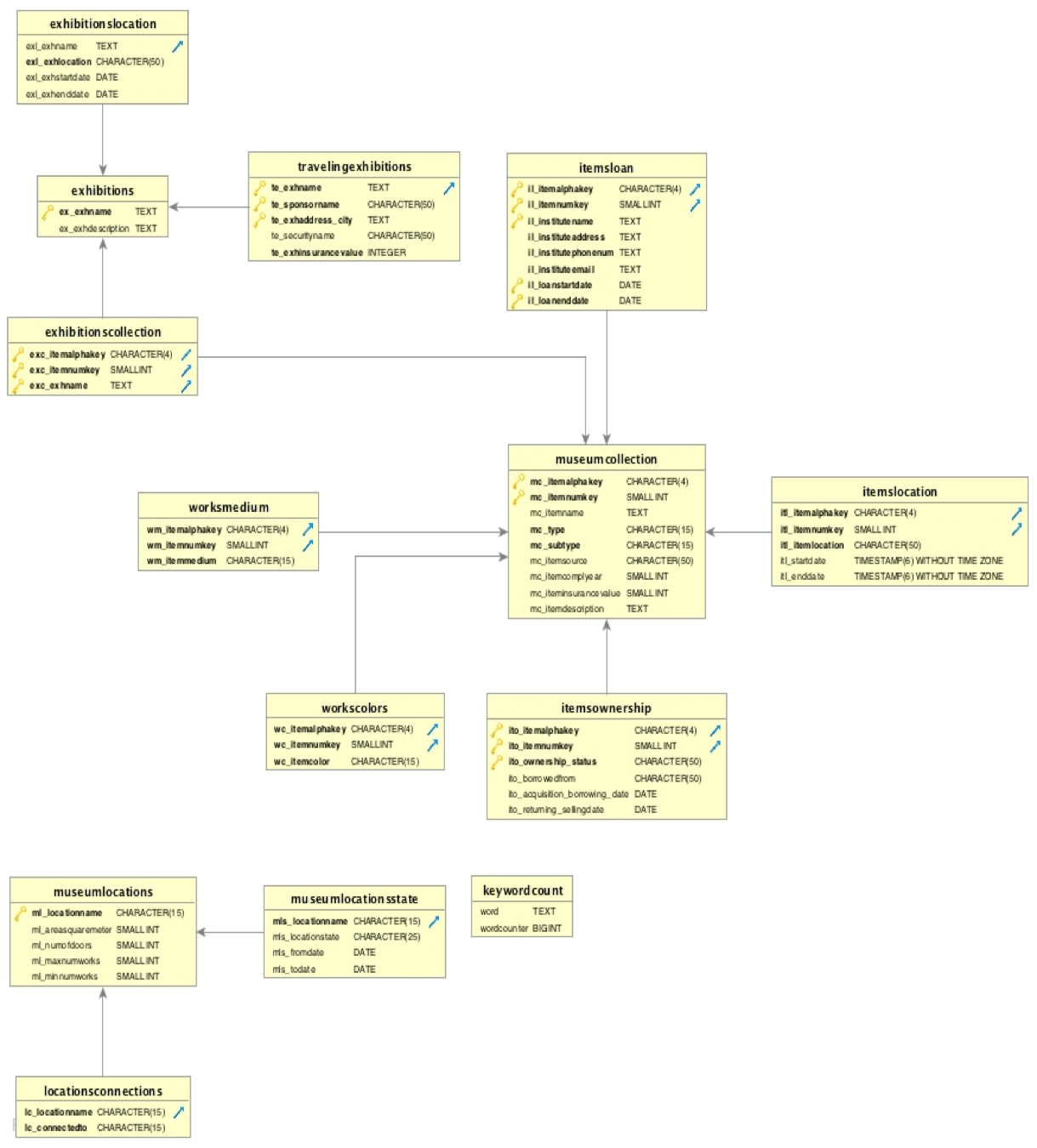
This section of the report discuss the necessary requirements and changes needed to transfer information in my database into the new team database. The details of how tables are queried to get all of the newly formatted data out of them and the process of creating the unified team database is discussed in details in Part E. However, this section will only discuss the transfer of information between two database, and whether there will be any information losses or not, however, it will not discuss the how part. The focus is whether my database has enough information to fit in the new database without the need of adding new information to the system. Part E shall have more details about how the information is logically pulled out of my database table and transferred to the new database.

My individual database system consists of a number of core tables that essentially records all information regarding my museum works, works’ location, museum’s locations, exhibitions, exhibition’s locations, exhibition’s works, travelling exhibitions and etc. Figure 1, shows a complete E-R diagram for my individual museum database.

According to the ER, and the database design the **MuseumCollection** table play a central role in the database system, as most of the database table need to reference the museum works. **MuseumCollection** table stores basic non-temporal information about works in my museum collection. This includes data about the Item character key, Item numeric key, Item’s name, Item’s type, Item’s subtype, Item author/Source, Year of completion, Item’s Insurance value and a description.

The new team database has an equivalent table, called **NS\_Works** that servers the same functionality of my **MuseumCollection** table, but with more attributes to classify works.

Information in **MuseumCollection** table can be mapped (i.e. transferred by simple queries) directly in to the **NS\_Works** table in the new team database without any information loss. However, the **NS\_Works** table contains more information fields to store data about the geographic region of the work (discussed earlier in section A of my individual report), related field of science, country of origin and whether a work can be borrowed or not. Work’s type and sub-types discussed at the beginning of this report, are mapped to PhysicalProperty and Classification, as these attributes in **NS\_Works** categorize works based on their physical property and a sub-class of physical property.

(Figure 1: E-R diagram for the Fick’s collection database)

The ER diagram shows a number of tables that are directly related to the **MuseumCollection** table discussed above. The description of these tables and how the information stored in the them will be transferred to the new database will be discussed as follow.

**WorksMedium** table: This table record information about the materials and media used in the creation of the work. It allows each work in the museum to have one or more type of media. **WorksMedium** table is related to NS\_Works by (1:n) relation, where for (1,,1) work in **MuseumCollection** table can have (1,,n) type of media in **WorksMedium** table)

In the new database, the **NS\_WorksMedium** serves the same purpose of the **WorksMedium** table in my database, and so the information in this table can be mapped directly into the **NS\_WorksMedium** table in the new team database without any information loss.

**ItemsOwnership table:** The name of this table in my database design was a wrong choice. This table was **not** designed to record the history of ownership of works in my museum, but rather records a history (start and end dates) of ownership related transactions that can be done on a work. A work can be initially purchased by my museum, then at a later it can be sold, then at sometime in the future it could be borrowed from another museum. The types of transactions that can only be handled by this table are : OWNED-PURCHASED', 'LONG TERM BORROWED', 'POTENTIAL BORROWING', 'SOLD'. Also, it worth to mention that this table does not deal with items ON LOAN, and did not consider works ON LOAN as an ownership related transactions (in my database Items on loans are handles separately by a separate table). This was a poor choice in my database design that will be corrected by migrating the information into the new team database.

The new team database design directly handles ownership related transactions by having a separate table for such purpose called **NS\_WorkTransaction.** The **NS\_WorkTransaction** table stores temporal information about the WorkCharID, WorkNumID, Museum Involved (i.e. the museum initiating the transaction), Client museum (i.e. The museum receiving the service from the transaction -beneficiary museum-), Transactiontype and Transaction time. The type of transactions that this table can deal with are: 'Purchased', 'Sold', 'Loaned', 'Borrowed', 'Returned', 'Missing', 'Destroyed', 'Sent for Repairs or Restoration'.

From that, one can see that Information in my **ItemsOwnership** table we discussed above can be transferred into **NS\_WorkTransaction** table in the new database. However, in order to do that firstly the **Ito\_ownership\_status**  attribute in **ItemsOwnership** table has to be abandoned and replaced by the correct attribute for **Transactiontype** as per the new team database design**.** My **ItemsOwnership** table will have to store information about all different types of ownership related transaction in **Ito\_ownership\_status**  attribute, like 'Purchased', 'Sold', 'Loaned', 'Borrowed', 'Returned', 'Missing', 'Destroyed', 'Sent for Repairs or Restoration' **instead of of just** OWNED-PURCHASED', 'LONG TERM BORROWED', 'POTENTIAL BORROWING', 'SOLD'. So, domain constraints on **Ito\_ownership\_status** has to be changed to allow all sort of ownership related transaction. Secondly, the **Ito\_borrowed\_from** attribute in my **ItemsOwnership** table has to store information that can directly be mapped into **wtrClient** (i.e. beneficiary museum) attribute in the new team **NS\_WorkTransaction** table. Thirdly, and lastly information about clients museums (i.e. currently stored in **Ito\_borrowed\_from**  attribute**)** in my database has to be transferred as well beforehand to **NS\_Owners** tables in the new team database.

In order to make this last point clear, the new team database, has a two tables to deal ownership of works **NS\_Owners** and **NS\_WorkOwners.**  The first stores information about the owners of the works themselves, including their names, phone number, e-mail and etc. These owener can be normal people, museums, organization or etc. While, the second table (i.e. **NS\_WorkOwners)** stores temporal history of the works ownership. For example, if you want to know who owns a particular work from some start date to some other end date, the **NS\_WorkOwners** is to serve you. So, in order to correctly perform **step 3** in migrating data from my **ItemsOwnership** table to the **NS\_WorkTransaction** table in the new database. All my museum clients (i.e beneficiary museum) stored in **Ito\_borrowed\_from**  attribute must be considered as works’ owners and information about them has to be manually recorded in **NS\_Owners**  table in the new database before transferring any data from  **ItemsOwnership** table .

However, direct information about who owns a particular work and when he owns it (i.e. history of works ownership) is not clearly recorded in my previous database base design, and it has to be extracted from several tables including **ItemsOwnership** and **ItemsLoans** tables (**ItemsLoans** tables will be discussed as follows). However this problem is fixed by the new team database since we he have **NS\_WorkOwners** that does that job.

Items on loans in my database design are treated separately. **ItemsLoans** table in my database records dated information about all loaned works (i.e. works ON LOAN) and the organizations our museum are loaning to (i.e. these are considered beneficiary museums). This includes data to identify the works itself (i.e. Work’s identification keys), the organization name, organization address, phone number, email address, starting date of the loan period and the ending date of the loan period.

From my previous discussion, it should be clear now that the new team database considered works on loan (i.e. loaned items) as an ownership related transaction, and it stores information about that in **NS\_WorkTransaction** table, while it stores information about owners (beneficiary museums included) and ownership history in **NS\_Owners** and **NS\_WorkOwners** tables. So, firstly the names of organizations/or institute and their personal information stored in my **ItemsLoans** table has to be transferred into **NS\_Owners** table in the new database. Secondly, all works in my **ItemsLoans** table has to be recorded as ‘Loaned’ transaction in **NS\_WorkTransaction** table in the new database.

**MuseumLocations table**: This table in my museum database records basic information about the locations/ and or rooms available in my individual museum. This includes data about the name of the location (example, Gallery\_A, Gallery\_B, Storage, Lobby and etc.), suggested minimum and maximum number of works that the location can hold, number of doors exist in the location and Area size (in meter square) of the whole location.

Information in this table can be exactly mapped directly with few minor modifications into **NS\_Locations** table in the new team database. Firstly, **NS\_Locations** table in the new database doesn’t record the Area size of the location, however, it records the physical dimensions (i.e. length and width) of the location in meters. So, in order not to lose information about the sizes of locations in my database, I need to explicitly records the physical dimensions of each location as length and width instead of using Area (i.e. **ML\_AreaSquaremeter** attribute in my **MuseumLocations** table must be dropped and replaced by two other attributes that describes length and width of the locations). Also, **NS\_Locations** table in the new database doesn’t record number of doors that a location has, however, this information can be calculated by counting how many connections a location has in **NS\_Doors** table (more about this point in the next point). So, dropping data about the number of doors a location has, and stored in ML\_NumDoors attribute in my **MuseumLocations** table is necessary in order to fit the new database design. Also, it worth to mention that **NS\_Locations** table in the new database can store locations inside any of our five partner museums or location outside our partner museums(useful for works in travelling exhibitions and works loaned to other museums outside our group) . So, it has a more broader view for the concept of locations

**LocationsConnection** table in my database records the connections (i.e. pathways) between different locations/rooms in my museums.

Information in this table can be mapped directly without modifications into **NS\_Doors** table in the new team database. It should be noted here, that the data we dropped earlier about the number of doors a location has, can be calculated from this table without the need to be recorded in a separate attribute, as it’s currently done in my database.

**MuseumLocationState**: This table in my database records dated history about the status/or condition of each location in the museum. In other words, it records by date only when a location is **open** (means active and currently displaying work), **closed** (might be closed to prepare for an exhibition), **under construction** and **in-maintenance**. Any changes in the status/or condition of the location is recorded in this table (example, a location might be opened for a certain period of time and for another period it’s recorded to be in-maintenance). Note: OPEN, UNDER\_CONSTRUCTION, IN-MAINTENANCE, PREPEXHIBITION are all values that can be stored in mls\_locationstatus attribute in this table

Information in this table can’t be transferred to the new team database tables, as there’s no tables in the new team database that can hold such information. It was assessed by team members not to be done. So, yes, information about the location status of my museum’s locations will not be recorded in the new team database be lost here.

**Exhibitions and ExhibitionLocation** tables in my database are responsible for all information regarding arranging new exhibitiona and managing its location. The first of these two tables stores information about exhibitions’ names and a brief description on the exhibition and its collection. While, **ExhibitionsLocation** table stores temporal information about the location of each exhibition ran by my museum. This includes data about the name of the exhibition, its intended location in the museum (example, which gallery this exhibition will be or was held in), starting date and time of the exhibition and end date and time. The main purpose of this table is to provide a detailed timestamped history of the locations an exhibition has over any period of time. This include travelling exhibitions locations as well.

Information in these last two tables can be transferred without modification to the database to the new database by recording their data content into **NS\_Exhibitions** and **NS\_ExhibitionLocations** tables in the new team database. My **ExhibitionLocations** table is meant to provide a temporal history of exhibitions’ locations regardless of whether ‘this’ location is in my museum (i.e. the location is a museum gallery) or outside the museum. While the **Exhibitions** table is used only to set up a new Exhibition by providing its name and description ONLY.

On the other side, the new team database **NS\_ExhibitionLocations** table records temporal history of exhibitions’ locations. These locations can be any of the locations (i.e. rooms) that exist in any of our five partner museums, or locations that exist outside our group in another museum because exhibition is on travel (Recall: locations in the new database can be any location in any of five group museums or a location of an external museum outside our group). While **NS\_Exhibitions** is used to fully organize and plan for an exhibition, this include the name of the exhibition, its starting and ending date, the museum it should be held in, whether it’s a travelling exhibition or no. This is different from my structure for **Exhibitions** table**,** however, during the process of data transfer from my database to the new database, there will be no changes needed to be done on my database structure, tables, attributes or domain. Information about exhibition will need only to be pulled out from my **Exhibitions**  and **ExhibitionLocations** tables using SQL queries and transferred to the **NS\_Exhibitions** and **NS\_ExhibitionLocations** in the new database. This will be explained in details **in PART E.**

**TravelingExhibition**  table in my database stores non-temporal information about exhibitions that my museum admin decided to set on travel. This includes the name of the traveling exhibition (the exhibition must originally exit in the museum database i.e. **Exhibitions** table), sponsor name, the city the exhibition will be held in (cities are used to identify the location of travelling exhibition), the name of the head of security responsible to secure the exhibition and finally the total insurance value for the traveling exhibition. Note: according to my database design the starting and end date and time of any exhibition regardless of whether it’s traveling or no are stored in **ExhibitionLocation** table because date and time of an exhibition worth nothing for a museum admin without mentioning the location factor according to my personal design.

Travelling exhibitions in the new database design is treated as normal exhibitions (i.e. stored in **NS\_Exhibition** table) only except that the museum they’re held at is a museum outside of group museums. So, their locations exists outside any of the locations of our five partner museums. Information about travelling exhibition sponsors in the new database is stored in **NS\_SponsorExhibitions**, this include data about the name of sponsor, the name of the sponsored museum and amount paid to sponsor any of exhibition.

From the above discussion in order to correctly transfer information about travelling exhibition to the new database, i have to abandon this **TravelingExhibition** table except for the sponsor name and store information about travelling exhibition normally as i store any regular exhibition in my database. Also, my **MuseumLocations** table have to start to store information about locations outside my museum.If this is done, then travelling exhibition will be transferred normally as i discussed earlier for a regular exhibition.

**ExhibitionsCollection** table in my database records information about the works in each exhibitions. It stores the exhibition name along with the work character and numeric identifier.

Information in this table can be transferred directly into **NS\_ExhibitionWorks** table in the new team database without the need of any modification in the database structure. However, **NS\_ExhibitionWorks**  table stores more information than just the name of the exhibitions and the work identification. It stores when an item is added to an exhibition (this might be different from the start date of the exhibition) and when it’s removed (this might be different from the end date of the exhibition). To fill up these missing information about when the work is added to an exhibition and when it’s remove, i have to query my **ItemsLocations** table (explained below) to check when a work in my museum is added to a specified location and when it’s taken out of this location. This has to be done for each work, and each location an exhibition has once been held in.

**ItemsLocation** table records temporal data about the location of each work in the museum. The purpose of this table is to provide a timestamped history for the locations of all works in the museums. Each work is recorded where it exists during a certain period of time between two dates.

Information in this table can be mapped and stored directly into **NS\_WorksLocation** table in the new team database without the need of any modification in my database structure. In the new database, the **NS\_WorksLocation** table records the location of work by identifying the location it physically existed/or exists at, the museum in which this location exists, and the time period (starting and ending date) the work was/is in this location. This is the same structure as my **ItemsLocation** table, with the exception that i need to record the name of the museum in which the work was/is in one of its location. Since, most of my works already exists in locations inside my museum. Most of the data will transfer to the new database **NS\_WorksLocation** table with ‘The Frick Collection’ as a value for **wolMuseumName** in **NS\_WorkLocations**

**C. Evaluating and testing the correctness of the new database. (Part D revised)**

There are several test cases that can be conducted on the new database tables in order to ensure that all my data was transferred correctly according to claims mentioned in this section. In Section, i will discuss some of the major test cases that can verify that the information content in my database was transferred correctly. However, the details of how these test cases are implemented and how it was verified to ensure the correctness of the database are mentioned in more details in PART E of this report.

**Test case 1:** Process a query on the new database **NS\_Works** table to produce all works in the database that has WorkCharId one of the following values (PNTG, TXTL, CLCK, MWJW and SCLP) and worDatabaseEntryMuseum =The Frick collection. This report should produce all works in the new database related to my museum. This report should be compared to a query on my **MuseumCollection** table, that produce the full content of the **MuseumCollection** table.

**Test case 2:** Process a query on the new database **NS\_WorkLocations** table to produce the history of locations for a specific item in my museum collection between two specific start and end dates. The new database should produce the same history of locations for the specified work, if the same query were to run on **Itemlocations** table in my database.

**Test case 3:** Process a query on the new database **NS\_ExhibitionLocations** table to produce the history of locations for a specific exhibition that belong to my museum between two specific start and end dates. I suggest producing this query for an exhibition named: Still life in European Art. This is exhibition specifically has existed in multiple location in my museum, and also was set on travel. The report produced from this query should be compared to a similar query on **ExhbitionLocation** table in my database. The history of locations should be identical including the name of the cities this exhibition was at, when it was travelling.

**Test case 4:** Process a query on the **NS\_WorkTransaction**  table in the new team database to produce all works in the new database that are related to my museum ‘The Frick Collection’ where the transaction type is set once to be Sold, once to be Borrowed, and once to be loaned. This report should be compared to the following two reports:

Samuel Germain

Student No.: 11131725

NSID: sjg970

CMPT 355

PART D

I will be adding a **museum** table to the database, which holds primary keys of the name of the museum and the address of the museum. I will test for these changes entering the museum into the database and checking for the existence of the museum records with a query.

For my **works** table the alpha ID, numerical ID, work name and description will not need to be edited as they easily fit into the new works table under attributes of the same data type. My subtypes mostly describe an era or a region works came from and will need to be entered as theme records for the works. I will test for my subtypes converting to themes by entering all the subtypes into my themes table and then testing for the existence of all subtypes within the themes table as well as check for if a few work is matched to the particular subtype they had previously. My types fit under the category of physical property because they describe that a piece of work is something physical like a painting, a sculpture etc. This data can be easily added as an attribute to the physical property. I will create classification for my works because any of my works can be further classified on attributes pertaining to their physical property (for example many of my paintings will be able to be classified as portraits, landscapes etc.) I will test for this by searching for the existence of all classifications for which I have entered into the database and change those which are misspelled words and search fro resons for why some may not have ben entered in. Because we are going with a date data type for the creation date of a work I will need to change many of my date attributes because they are unspecific (for example some creation dates are recorded as something such as the 18th century) I will test for this by querying for all attributes in this field which do not fit the proper date format, such as being too short or containing a character. For my creation dates I will enter dates in for them that coincide to the last possible date for a period of time (if the date is the 18th century then I will enter a date of 1799/12/31. All of my values for artists will fall under the creator attribute in the new database. For my dates of acquisition I will need to create owner records for when the museum first came into possession of a work I will test for this by making sure the number of transactions that share the same acquisition date and same primary keys for a work is the number. Ideally there should be one transaction like this for each one of my works. For all my insurance values I will need to create insurance records because insurance values are becoming temporal data and I will enter the date as the current date because I know that work had this insurance value on this day. I will again test for this by checking that my records for insurance values are equal in number to the amount of works I have. I will need to move all my values for owners into owner records because this is now temporal data. I will choose the start dates of the owner records to be today because I know that the owner of that work owns the work on this date. I will test for this by making sure the number of owner records which I have is equal to the distinct attributes of owners in my work table. For my borrow status attribute I will make the records which record borrowed and create borrowed transaction records with the date it was borrowed as the current date because I know that it was being borrowed on this date. For those works recorded as potentially borrowed I will enter these into the new database works table under the borrowable attribute. I will also need to add the data entry museum name and data entry museum address as attributes and primary keys to my works table, which will reference the Museums table. I will test for this by checking that the same amount of records that I have for borrowed works matches those works which have borrowed recorded as an attribute.

For my **media** table I can easily put all these records into the new WorksMediums table because the attributes in my media table match the attributes in the mediums table and both sets of attributes use the same data types with the exceptions of adding attributes of the museum name and museum address to account for the new primary keys pertaining to the works table

For my **locations** table I will not need to change anything because my attributes and datatypes for my attributes sync with the attributes and domains for the attributes decided for the new museum with the exceptions of adding the museum name and museum address for all the locations

For my **doors** I will need to double the amount of records because my museum only has one entry for each door referencing the two locations the door connects but our collective database design had decided to have two entries for each door where each entry designates that there is a path from one location to the next. I will test for this by running queries which check that for every door leading from one room to another there will be a door leading from the other room to the first room.

For my **WorkLocations** table I will need to add in the data entry museum name and data entry museum location for each WorkLocation record because they are primary keys for the work that workLocations must reference. As well I will need to add the attributes that hold the value for the museum that is the primary key of the location.

For my **exhibitions** table I will need to add an attribute for all the records that says what museum the exhibition originated out of. I will give the value of ‘Indianpolis Museum of Art’ to all my exhibition records. I will need to add the corresponding address as well. To my ExhibitionLocations table I will need to add the same attributes, which will reference the museum attributes from the exhibitions table for the corresponding record.

My **ExhibitionWorks** table will need to have attributes added indicating the data entry museum of the exhibition and the work each record references. The values of my museum will be given for both.

My **transactions** table can retain what it has because it’s attributes and data types sync with the design of the new database. There are no records indicating a transaction was a borrow and a loan but the table already specifies that the transaction was a sale or a buy so the new values or borrow and loan will not cause any problems.

For all my tables where the museum name and museum address are added I will test to see if the number of records returning for those works matching my museum is the same as the number of records in the table total.

For my **loanees** table I will move all of the data referring to individuals into the owners’ table. All information referring to the loaning of a work will be recorded into the transactions table.. I will check for this by querying to see if all the owners listed in the loans table are all in the owners table.

I will need to add attributes for the data entry museum and the data entry museum address to my **travelling exhibitions** table and set the attributes to be my museum for every record. I will also need to change my travelling exhibitions so that they reference the exhibition, which they were based on of because now they reference a record that is, and exhibition record for the travelling museum. I will need to split information in my travelling exhibitions table. I will need to take all information relating to a location of a travelling exhibition and put it into a travelling locations table which I will need to create. The remainder of the travelling exhibition information can stay as it is because it will be able to fit into the new database travelling exhibitions table. I will test for this by querying for data on the original exhibition of my travelling exhibition and by querying to check that my travellingExhibitionLocations has the same amount of entries as my travelling exhibitions table which basically describes the locations my travelling exhibition will visit.

My table called **conception**, which I created a table to describe a more abstract way of categorizing works, I will not need to alter because it will fall perfectly under the themes table.

These changes will prepare my database to have all of its information transferred to the new database.

**Part E**

Once everyone had finished preparing their databases for the merge, we used dbVisualizer's export function, which exported our databases to sql files, which contained INSERTS, UPDATES and CREATES which would recreate our database. We then created schemas for each team member on the central database. After we modified the sql files slightly to account for a few errors caused by the export process, each person checked them for errors in their data on the central server, they used select statements to insert their data into our central schema and set of tables.

Essentially a copy was made of all the tables and domains which were designed for the new database but existing within our own database (all tables prefixed with NS). Then we could essentially do queries to copy each attribute to a new attribute of the same name to an identical table which was not prefixed with ns. Once all data was in the database there were some instances of certain data not being copied over. We did a check for all data which should have been in the new database and made sure records existed for all five museums. This is because although information may have transferred over successfully into the tables within our own database this does not mean the transition worked. We also checked for all attributes which should have existing values and located the ones which did not. We then did a check for what museum these records originated from and assigned the team member to fix the attributes. One problem which was run into by a few team members was the lack of recording of workOwners is because many works owned by the museums had their ownership recorded as an acquisition date. It was forgotten then, by a few to record temporal data relating to the museums coming into possession of the work.

**PART E: Smithsonian American Museum of Art**

**By: Warren Fehr**

The following actions were taken for moving the data from our old database schema to the new unified museum database design agreed upon by the 5 museums.

Explanation for how I got the information for each table.

**Owners:**

I was able to get information from a loners table, and a loaners table, and easily select all the data and transfer it into a version of the new owners table. The major problem I had was reformatting that the address data was not properly formatted, so i had to go in and personally reformat it using UPDATES and INSERTS. I added an entry for my museum to this table, as well.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**Works:**

No major changes needed. I selected all the data from my old tables, and added pertinent values to fill the new attributes(namely setting the databaseentrylocation to 'Smithsonian American Art Museum', the physicalproperty to 'Painting', and the geographic region and country to 'United States'.).

Testing: I produced a query of the new records and compared it by hand to the prior data.

**Locations:**

I added the data from my old locations table without any issue. I had a few travelling exhibition locations in a separate table in my old db, which was problematic. Thankfully, there weren't many of them, so I just created a few INSERTS with their data manually and inserted it into the locations table.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**Exhibitions:**

The regular exhibitions were selected from the old tables and added into to the new tables without any issues. I had one travelling exhibition in a seperate table, so i converted the information it had and INSERTed by hand.

Testing: I produced a query of the new records and compared it by hand to the prior data.

Doors & WorksMediums:

All I needed was to simply select from the old table with a slight change to account for databaseentrylocation primary keys. No modifications were needed for the data.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**SponsorExhibitions:**

This is a brand new table. I only had one entry in it from the travelling exhibition I had, and I inserted it with no problems.

Testing: I checked the single entry and compared it by hand to the prior data.

**ExhibitionLocations**

I selected the data from the old table, added in the new primary keys for the museums, and copied it in without any worries. I had to take the entries from a travellingExhibitionLocations, format them to fit the exhibitionlocations format by hand, then manually inserted them in afterwards.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**WorkOwners**

I made this new table from scratch. I made an entry for all items with their ownership changing to me on the day I acquired them, and then referenced an old transfers/transactions table I had and added info for borrowed,loaned, and sold works.

Testing: I created this from scratch, so all I did was the check that all the transaction info was correct.

**WorksInsurance**

I made this new table using the insurance values from my works table, with the insurance being issued on the day I acquired it.. I did not have any individual insurance values stored for my travelling exhibitions, so I had nothing to reference for that.

Testing: I produced a query of the new records and compared it by hand to the prior data in Works.

**WorkLocations**

This table needed a lot of work. My previous version of this table only had arrival times, which was not adequate enough. In the end, I basically had to recreate most the work moving data by hand, from scratch, with inserts and updates, which was long and gruelling.

Testing: I checked every single item to make sure it had a flawless chain of transfers: It should start in Storage when it was first bought, and should have a set of entries that show its set of movements (without any gaps) to its current location, where it should have a departure time of null (which means it hasn't moved).

**ExhibtionWorks**

I could basically select the information from my old exhibitionworks table and insert it without any major problems. I had to create new entries for all the items that were in my travelling exhibition and add them in as well.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**WorkTransactions**

I had several tables covering various transactions, such as loans, borrowing, and sales. I combined them into this one via select statements, marking the transaction type and client for each transaction depending on which table they came from.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**Themes**

This is brandnew, I simply added an entry for all my works as an 'American' theme, and added an entry for the pieces marked as war pieces as 'War'.

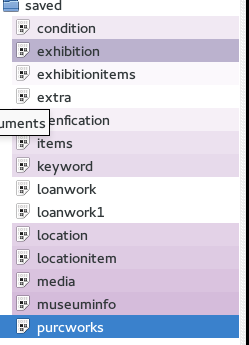
Testing: Not much here, just make sure all the pieces that are war pieces are tagged properly.

NSID : jiz457

NAME : Zang, JiaWei

Step 1

Save all data of original database into sql and txt, to make sure that I don’ t lose anything.



Step 2

Delete all original database by dbVisualizer or command line

PS: I Evaluated DbVisualizer to professional Version, then I could do actions.

Step 3

Run code again without extra works related to bonus question

Reset the database to be just have assignment 1 - 3 Because my database contain extra information about extra works by running original code I saved.

Step 4

Evaluation for Database structure, which does not contain extra works

By checking the code in extra work is there, including exhibitions which I added to test the cost. If there is not exhibition which I added in the extra section, then the Reset the database is evaluated.

Step 5

Create new Schemas and in this case named this called “test”, and let this schema to be temp schema the same database structure with team database. Use the original database and use sql query insert into the “test” schema same with the team database which including domain and tables

Step 6

Domains, I run the scripts which create domain on Dbvisualizer professional.

Step 7

Transfer data into new database or new schema

Owners

* To transfer original data into this table, I am using insert to into the new schema. Basically I am missing the state, street name, building number, so I leave those into empty. Insert rest of them into new table. Basically, I saved all the information about owner table, because, in this table I am not getting any issue.

Locations

* To transfer original data into this table, I am using the location table as source in my original schema, And only thing new for me is traveling exhibition location to this part I divided into two parts, one part is in Israel and one part is not. If a location city is not Israel then this location is traveling location.

Source from location table

Doors

* To transfer original data into this table, I am using door table which in original table in my original schema. Because these are similar, I just need to insert into new database with original table data. This table is same table I had.

Source from door table

Works

* To transfer original data into this table, I am using identification table and item table, because the type and sub type are changed,, so and I am using the sub type to check the physical property, for example, if the sub type is Asian Art Painting, then the physical property is Painting and the work classification will be same with the original sub type.
* The additional information the keyword is does not important (useless), they are all “ART” etc al. And this is does not mean anything.
* Therefore, I ignore this out of new schema.

Source from item and identification

Exhibitions

* My original table contain exhibition location information, therefore, I need to take those out.
* Because in this table, there is repeat exhibition entry in exhibition name attributes which called “TOP NOTCHED show”. For this record, I just take one. And I still missing the security person name, and my original table does not have it, but this record is important, therefore, I make it up.
* And, for the attributes which is called is traveling exhibition, they are all false by default, and I sql command.
* For the description, I just copy and paste from original exhibition description into new exhibition table. And for the attributes which called security person name I made this up, since I did have this one in my original database table.

Exhibition Locations

* This information was in exhibition table, therefore, I just get the exhibition and their location information into new exhibition location table.

Sponsor Exhibitions

* I was missing this sponsor for exhibition, therefore, I need to make the sponsor information first. In this case, I made the Bill and zang, Time up for the sponsor name.
* And I divided those into two part, one part is for the “'TOP NOTCHED show'”, because this there are two entries in original table,

Exhibition Works

* Using the sql query to new schema, I divided into two parts, and select from original tables.

Work owners

* Firstly, I insert data which is own by my museum, which is own by me all the time.
* Secondly, I insert data which are belonged to loaned work, which is mean the ownership is outside museum, which is mean that works ownership did change. And I find the assignment requirement, and set the start date and set the end date to be null.
* Thirdly, insert the date which is purchased and sold, in this case, I set the seller to be unknow and buyer to be unknow. And set the selling items for seller ownership end date to be '2016-10-25, and set the buyer ownership start date to be '2016-10-25, and set the end of ownership to be null and vise versa.

Themes

* In this new table, I insert the type item table from my original schema into this table.

Work locations

* In this new table, I insert this table from my original location item table by using the sql code.

Works insurance

* In this table, I am miss insurance value date start and end date, therefore, I made this up to be 2013 01 01 to 2022 01 01, which will be reasonable. And insert the insurance value from my original table which is called item table.

Works medium

* In this table, just insert my original data from media table.

Work transaction

* I am missing this table, because I am using the location item, condition, and item identification tables to holding this information. Therefore, I am using query select those information from the item condition and item identification.
* And I am missing the sold and purchase and loan or return information, in my case, I have P for purchase, SOD for sold, PB, and LB are for borrowed. And I add interval 12 hours, which make this to be into time stamp.
* I divided those into loan and return section, and purchase and sold section.
* And I for sold item, I have sold item for record.
* In this case, purchase item action is happen on 2016-10-25, then I insert this into new database. And insert add this into sold record by same record.
* In this case, sold item action is happen on 2016-10-21, then I insert this into new database. And insert add this into purchase record by same record.
* In this case, loan item action is happen on 2015-05-30, then I insert this into new database. And insert add this date into borrow record by same record.

PS: And I have loantable, keyword, loanwork, purwork table which is unimportant, and they are all repeat information. And keyword is useless for this situation because the information stored in this table are all “art”, “Israel”, therefore, I ignore the data stored in those table.

Step 7

Evaluate the data

Part A

Preliminary Evaluation

Evaluating the tables, domains and basic database elements.

* Evaluation Environment
  + Windows + putty / dbvisualizer
  + Linux + Command line / dbvisualizer
* Testing for team table and structure.
  + Reason I testing it is I need to make sure, team member won’t have trouble with the those.
  + Errors I found:
    - Repeat Domains
      * Fixed by delete those
    - Missing Constraints
      * Fixed by adding constraints
    - Missing keys in some table
      * Fixed by adding and update the key in those table.

Part B

Team Peer Review Evaluation

Let team member to read my records in the team database. And I got feedback from them.

* Evaluation Environment
  + Team members can access the team database
* Log: Mistake found in owner information, which should be only one to be my museum which is “Israel museum’
  + Fixed: By update the true into false excluding the israel museum
* Missing Phone number in owner information
  + Fixed: By update the information about the owner information

Part C

Pre-Transfer Evaluation

This testing or evaluation is for before I loading data into team database, I doing this testing because I don’t want to mass the team database.

* Evaluation Environment
  + Windows + putty / dbvisualizer
  + Linux + Command line / dbvisualizer
  + In cmpt355\_jiz457 public schema
* Owner table
  + Query this table which is matching team database table
    - Check that the there is new attributes added on and there are 3 attributes are null, because those I did have it.
      * Evaluating source in additional file
* Check table structure contain all attributes
* Check empty attributes are missing, because I don’t have those
* Check email attribute which have this attributes
* Locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
      * Check all attributes is there, and check the data is correct
      * Check the travelling location is five.
* Doors
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Don’t need to check it because new team table and my original are same
* Works
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is no record for 3 attributes
      * Check the physical property and work classification are matching
        + For example, Painting is match for Asian Art Painting and so on
* Exhibitions
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
* Exhibition Locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check that there are 17 record.
      * Check that there are 2 record for muti-room exhibition (in this case is should be 2)
* Sponsor Exhibitions
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
* Exhibition Works
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
* Themes
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
* Work locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
* Work owners
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
* Works insurance
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
* Works medium
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
* Work transaction
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there are just four type in the transaction
      * Check the count of purchase and sold item records are the same
      * Check the count of loan and borrow item records

PS: Those evaluation is in additional file.

Part D

Post-Transfer Evaluation

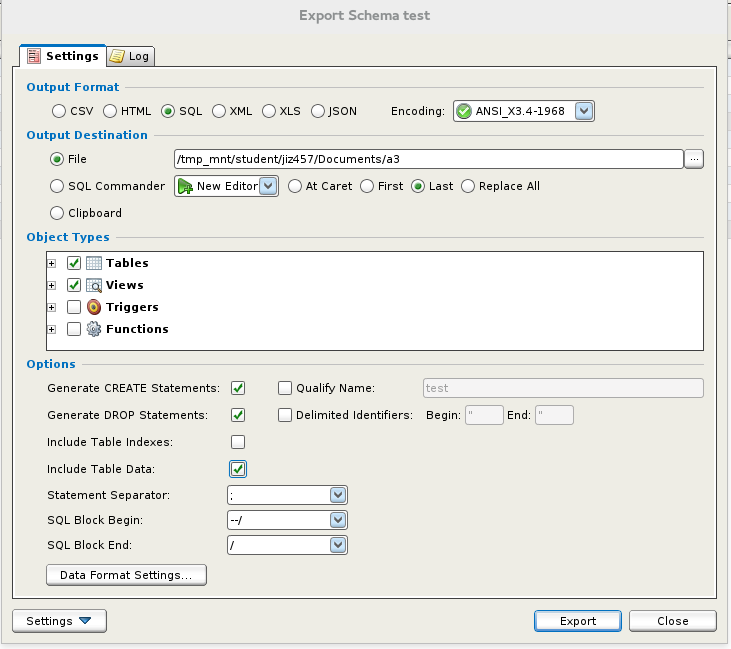
This testing or evaluation is for after I loading data into database. Make sure my data is correct. AKA this test will check

* Evaluation Environment
  + Windows + putty / dbvisualizer
  + Linux + Command line / dbvisualizer
  + In cmpt355\_jiz457 public schema
* Owner table
  + Query this table which is matching team database table
    - Check that the there is new attributes added on and there are 3 attributes are null, because those I did have it.
      * Evaluating source in additional file
* Check table structure contain all attributes
* Check empty attributes are missing, because I don’t have those
* Check email attribute which have this attributes
* Check the my data is there
* Locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
      * Check all attributes is there, and check the data is correct
      * Check the travelling location is five.
      * Check the my data is there
* Doors
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Don’t need to check it because new team table and my original are same
      * Check the my data is there
* Works
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is no record for 3 attributes
      * Check the physical property and work classification are matching
        + For example, Painting is match for Asian Art Painting and so on
        + Check the my data is there
* Exhibitions
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check the my data is there
* Exhibition Locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check that there are 17 record.
      * Check that there are 2 record for muti-room exhibition (in this case is should be 2)
      * Check the my data is there
* Sponsor Exhibitions
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check the my data is there
* Exhibition Works
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
    - Check the my data is there
* Themes
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
    - Check the my data is there
* Work locations
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
    - Check there is entry museum is there
    - Check the my data is there
* Work owners
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
      * Check the my data is there
* Works insurance
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
      * Check the my data is there
* Works medium
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there is entry museum is there
      * Check the my data is there
* Work transaction
  + Query this table which is matching team database table
    - Check table structure contain all attributes
    - Check all attributes is there, and check the data is correct
      * Check there are just four type in the transaction
      * Check the count of purchase and sold item records are the same
      * Check the count of loan and borrow item records
      * Check the my data is there

PS: In Evaluation Team sql file, which should run on team08 server and in public schema.

Step 8

Export the schema, select the including data, select the SQL. and export the sql file



Step 9

Create new schema for the debugging the sql code is running perfectly

Debug the sql file, tried to run this file in SQL Commander three time.

Step 10

Import into team database

Step 11

Done.

**Steps to transfer data into team database**

Following the step 8, 9 and 10

AKA

Export my database into sql in Dbvisualizer

Then create schema in team server.

And run this in new schema in team server.

And query from this new schema and insert into team database.

**PART E: Canadian Museum of Science and Technology Database Migration Report**

By: Richard Granger

The following actions were taken for moving the data from our old database schema to the new unified museum database design agreed upon by the 5 museums. We will outline how the data was moved to each table in the new design one by one:

**Doors:**

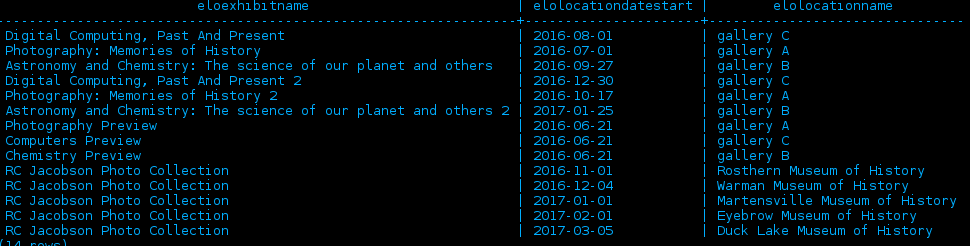
My previous database design relied on character identifiers for locations, and so for this table the data was migrated manually record by record.

Testing: I produced a query of the new records and compared it by hand to the prior data.

**ExhibitionLocations:**

My previous database design was fairly consistent with the new database design, so most of the information was easily copied over, with the museum name being filled in with our name. We did notice that parts of the travelling exhibition were missing, so they were then inserted by hand into the new database.

Testing: We produced a query that described the exhibitionlocations within the new database design, and it showed that all data was moved properly and that the erroneous exhibits had been corrected. As shown here:



**Exhibitions:**

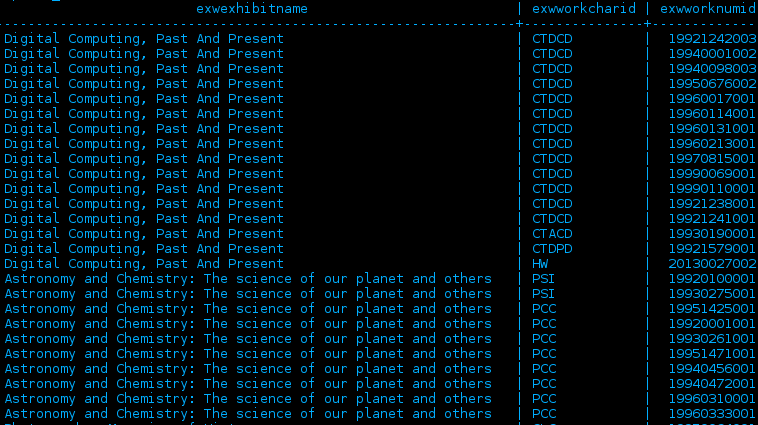
The design of our exhibitions table was mostly similar to the design of the team database, so most data was copied over without issue. We had a slight issue with travelling exhibitions, but after a manual adjustment the data was corrected.

Testing: We produced a query of the old and new data and examined both to be consistent, and thus were satisfied with the quality of the data transfer.

**ExhibitionWorks:**

This data was almost perfectly identical with the new database design, and required no readjustment, it was simply added to the database without having to change anything.

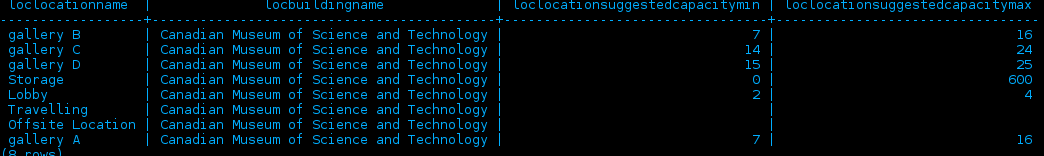
Testing: The data was perfectly preserved when we did a query in the new design as shown below:



**Locations:**

The data in our museum did not record a person for each location, so we had to mess with the data a little bit here. We mass transferred the data, using temporary owners, then adding the proper owners and changing the locations to the right owners. The locations in the new design now had their proper owners according to the new database design.

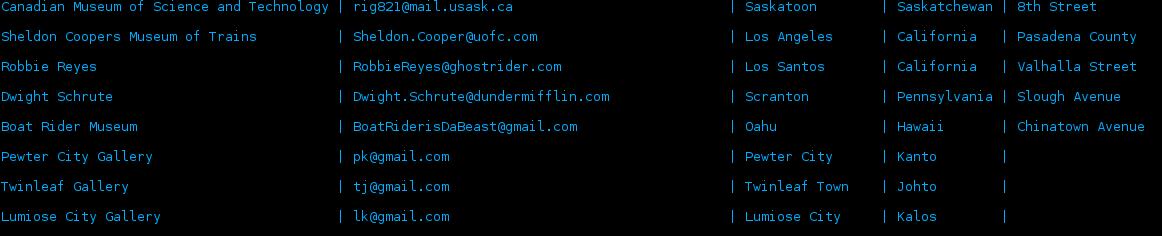
Testing: Producing a query of the locations within our museum and cross-referencing the information with our old database showed that the data had transferred properly.



**Owners:**

The data for owners in our museum was both in the owners table and in the loaninfo table. We moved over the data from both tables into the new database. We were also required to add new owners for the building’s in which we had sent works abroad and works that we had sold.

Testing: We tested this by producing a query and comparing it to the original database to ensure that BOTH loaninfo and owners had been moved over successfully. The query was correct and produced the following:



As you can see places that we have loaned to and places that have had travelling exhibitions, it has all transferred successfully.

**SponsorExhibitions:**

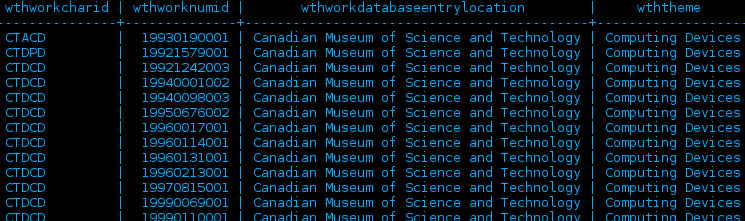
We fairly easily added this data from our old database, and cross-referenced old budget slips to enter the amount that Rogers Incorporated paid to sponsor each exhibition. This information was entered into the database along with the data that we already had.

Testing: A query showed that all 5 exhibitions that were sponsored showed up with the correct exhibit name and sponsor amount.

**Themes:**

This data had not existed prior in our own database. We studied works carefully and added appropriate themes to certain works as the saw appropriately. The data was then added to the database.

Testing: This did not exist prior, so there is nothing to reference it to. We did however query the new design to ensure that the data was put in correctly:



**WorkLocations:**

This data was mostly easy to transfer to the new database, however upon transferring we did notice some inconsistencies in the table, which we corrected manually. The data was then looked over manually and found to be accurate.

Testing: We produced a query that showed works locations comparing them to the original data in the database. The fixes that we implemented were shown in the new data and the correct data had transferred over properly.

**WorkOwners:**

This data transferred over relatively straighforwardely, with few manual changes needed.

Testing: Here we tested in queries to ensure that the ownership remained the same. We also produced a query to ensure that sold works showed a transfer in ownership, and they did, so everything transferred properly.

**Works:**

Transfering this data was very easy, simple copying, with manually inserting some data as necessary to the new database. Our old category fit nicely with the field of science way to sort works, so I ported some of those applicable category names to the new field of science part of the works table. I also had to create a function that ported my 4 digit year of creation and made it into a date to be in standard with the new database.

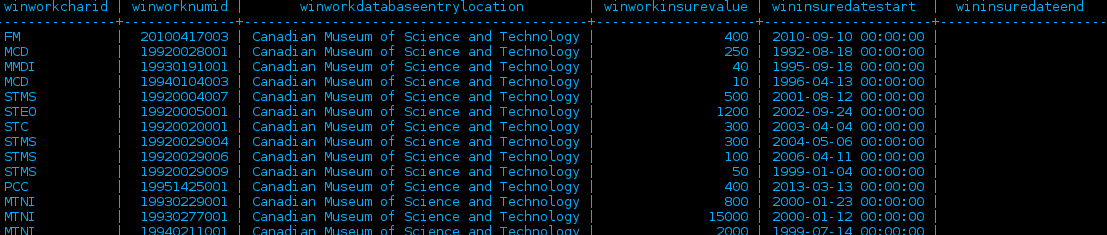
Testing: I produced a query and compared it to the data of the original database to ensure that everything had transferred properly. As shown as a sample below, it did:



**WorksInsurance:**

Transferring this data over mainly consisted of moving the insurance value from the works table along with the appropriate data. We had to manually change some entries due to the value of the certain works changing due to being sent out in a travelling exhibition.

Testing: I produced a query, shown like the one below that ensured the new temporal data of insurance was proper, and compared it to my past database to ensure that there was no errors.



**WorksMedium:**

This one required almost no editing as the structure was almost identical, merely adding a museum field to the data as required. Everything migrated without issue here.

Testing: Produced a query of the materials that certain works were made out of and compared them to the original table to ensure data integrity.

**WorkTransaction:**

This data did not exist in our database prior, so going through physical records the entirety of the new data was meticulously constructed manually. This includes all events that our past museum had for transactions, they have now been entered into the new database schema.

Testing: No data existed prior, so we produced a query to ensure that the data made sense and was not corrupted, referencing it with physical records to check accuracy. An example is shown below:



**All data is therefore now transferred to the new database schema properly.**

Samuel Germain

11131725

sjg970

Part E

**Mostly Unaltered Tables**

For the tables Doors, Media, workLocations, exhibitionlocations, conception, and exhibitionlocations, all that was added was the museum name which was ‘Indianapolis Museum of Art’ to uniquely identify the works and locations for each record in each of these tables. The records in each of these tables were copied to identical tables other the museum name attributes with the exception that records in conception were copied to the themes table, and records in media were copied to a Medium table.

**Testing**

I tested to see that the number of records which I had in my tables of my previous database matched the number of records for the identical tables in the new database, while also checking that unwanted duplicate records were not created because this meant everything copied over successfully. I also did count test by counting the number of records in my new tables compared to my old tables because there should have been identical numbers of records.

**Exhibitions Table**

**Added attributes**

**Exh\_istravellingExhibition**

This was added because all exhibitions were to be recorded as travelling exhibitions. My database already had this structure but it could only be checked that an exhibition was travelling by checking if the location it went to was named ‘Travelling’. Because the new database is not set up like this I had to add this attribute and check it as true for my travelling exhibition.

**Testing**

Testing for this was simple as it only should have been true for one exhibition, so I just had to check that this was the correct one.

**Exh\_SecurityPersonName**

The travelexhibitionlocations table in my database was removed. It recorded information about extra information for travelling exhibitions including the security person for the travelling exhibition. Because of this an attribute for the security person was added to the exhibition table and the values of the security were entered in for those exhibitions which were travelling

**Travelling**

I only had one security person and he was only assigned to one exhibition so in testing I just confirmed that he was being security for just this one exhibition

**Exh\_MuseumName**

I added that all my exhibitions were from the Indianapolis Museum of Art and tested to make sure they all had this value

For all other attributes in the exhibitions table I simply copied the records to my new database and performed count queries to make sure that I had the right number of exhibitions in my new database, while also checking that unwanted duplicate records did not copy over.

**Locations Table**

Most of my information for the locations table was able to be transferred over. I was not able to transport over when my locations were available as this function was not noticed until too late in the implementation of the database although I would have liked to transfer this over as it’s own table. Height of the room was also removed because it was never actually needed in the database.

**Added Attributes**

**Loc\_isTravelling**

This attribute was added because the travelexhibitionlocations table was removed, so it was needed to keep track of which locations were not physically in one of the museums. Values for width length and capacity were left as null for travelling exhibitions.

**Testing**

I checked to make sure only those values which were part of the travelling exhibition had the true value as all other ones should have been false.

**Loc\_BuildingName**

I entered Indianapolis Museum of Art for all the locations located in my museum. For travelling locations, I added the owner of the building for which the travelling exhibition was visiting, which were other museums. My database had the city listed as the travelling exhibition locations, so I added this as the location name while adding the name of the museum which was to be visited as the building name.

**Testing**

I checked to make sure all my locations were listed as the Indianapolis museum of art while all other locations had a museum which made sense for the location listed as the building name.

**Added Values**

To account for the location of works which were borrowed, which the museum would care about, I added locations for the museum which borrowed the work as the building name, with on loan as the location name. My old database handled the location of works by simply listing the location as on loan, but the new database had the building name as part of the primary key so it was necessary to add locations of on loan for each museum borrowing into the new database.

**WorkTransaction Table**

The attributes in the WorkTransaction table for the charid, the numid, the client, the transactiontime and the transactiontype were copied from the values stored under my attributes traaccsym, traaccnum, traowner, trasaledate and tratransactiontype respectively. I also included transactions of loan from my loanees table which were loans of my works lent out to other museums and because the start date, the date, and the person involved in buying the works were already available data this did not create any problems in transferring the data. I added the value of ‘Indianapolis Museum of Art’ as both the workdatabaseentrymuseum and the museuminvolved because it was my museum involved in all these transactions. I did not record any transactions of my museum borrowing any works, works that have been damaged or works, or works which have gone missing because my museum did not have any records of this data.

**Works Table**

**Added and Altered Attributes**

**Wor\_workclassification**

This attribute was added to my database in order to further classify the type of work something was. More specific information can now be found about a piece such as the ability to find whether a painting is an portrait, landscape or still life and instead of knowing that a work is a piece of furniture, it can be known that the piece is specifically a wardrobe or a chair.

**Testing**

As this information was all new information for those works, which were paintings, I simply had to check that all paintings and furniture had a value for classification which made sense pertaining to the work. For other works such as all types of sculptures I compared the new works table to my old works table and made sure that all types for these types of works in the old works table matched the subtypes for the new works table.

**Wor\_workphysicalproperty**

This attribute replace the type attribute from my database. Most of my attributes could be copied directly into this attribute. Because a classification attribute was added some of my types which were physical properties went into the classification attribute because they fit more as a subtype of the type of work the piece was. The types of ‘mold’, ‘carving’, and ‘metal work’ were put as classifications and the physical property was assigned to sculpture because these values fit more as classifications.

**Testing**

For all works which were paintings and furniture I compared my new works table to my old works table to make sure the types in works matched physicalproperty is ns\_works. For all works which were types of sculptures I check to make sure their physical property was indeed a sculpture. For all works which were clothing I checked to make sure the physical property was textile.

**Wor\_workcreator**

Some attributes were removed from work creator because they essentially described where the country of origin for a work was. This includes values such as Egyptian culture and Greek culture, because with the addition of the country of origin attribute, listing the culture a work is from under the creator is from becomes redundant.

**Testing**

I checked to make sure the creator attribute in ns\_works matched the artist attribute in my works table. As there were only a handful of works for which I removed the value assigned to artist, I checked to make sure these works had a culture which was also a country listed in my artist table.

**Wor\_workcreationdate and Wor\_workCreationTimePeriod**

Althought an attribute existed within my database to record the creation date, it did not give a specific date, it gave a range of time or a year and so the . Because of this, works where not able to be categorized or compared based on when they were created. Under the creation date attribute I took the latest date possible for the time period listed for my works (i.e. a work recorded as 1930-1940 was given the creation Dec 31 1940) because I knew that my works were at least this old. My team decided to add the workCreationTimePeriod attribute because without it I lost a good amount of information. If for example I had a work recorded as being from the 14th to 16th century, this is a wide range, and although it is useful to record this value as the latest date in the 16th century for purposes of comparing data, it is also useful to keep the data stating that a work was created anywhere between 200 years.

**Testing**

I checked to make sure the values in my workcreationtimeperiod attribute in ns\_works matched those for my creationdate in my works table. For my creationdate attribute in my ns\_works table, I checked to make sure the date was Dec 31 of the latest year in the range of years for the value in my creationdate field for my works table.

**Wor\_workBorrowable**

My database recorded whether a work was owned, was borrowed, or could be potentially borrowed. I replaced this with an attribute in the work table which just said if a work was borrowable. This was suffice because the owners table records if a work is owned by the museum, and the transactions table can be used to find if a work is currently borrowed. All that needed to be known then is if a work could be borrowed so all works which were recorded as ‘borrowed’ or ‘potentially borrowed’ were given the value ‘TRUE’ for wor\_workborrowable. The date when a work was borrowed was not available and so no data was added for borrow transactions which were not recorded as this would be incorrect and fraudulent data.

**Testing**

In testing this I checked to make sure those attributes, which were potentially\_borrowed and borrowed in my works tables were recorded as true in my ns\_works table and false otherwise.

**Wor\_workcountryoforigin**

Much of this information for the country where a work came from was able to be extracted from my database under the subtype attributes. This was able to be done for all attributes which stated a type of American art. However those works which were listed as African, Asian, European and from the Americas, new data had to be entered in.

**Testing**

I checked that those works which were a style of American art(American impressionism, American modernism etc. ) said that the country of origin was the united states and for other works I checked that the country of origin within my ns\_works table, residing within the geographic region which was listed under the subtype for my works table.

**Wor\_workgeographicregionoforigin**

This was mainly able to be extracted from the subtype attribute within my works table, as my subtypes mostly indicated a region of the world (Ancient art of the Mediterranean, native arts of the Americas, Asian art, African art, American modernism …). For the American arts I listed the region as north America.

To avoid losing information about a work being ancient in the case of ‘Ancient art of the Meditteranean, or a work being native, in ‘native arts of the americas’. I added a themes of native arts and ancient arts to my themes table.

**Testing**

I checked to make sure that the values for geographicregionoforigin for my ns\_works table attribute matched the region which was stated in my subtypes for my works table.

**Wor\_workfieldOfScience**

I left all these values as null for my works, as none of them pertained to a field of science.

**Wor\_DatabaseEntryLocation**

I simply entered that each of my works was entered in at the ‘Indianapolis Museum of Art’ and checked to make sure each work got this value.

**Removed Attributes From Works**

**DateOfAcquistion**

This attribute, which was present in my old database as wkdoacq, was not included in my new database as the date of acquisition could be obtained by looking at the transactions table for when the museum obtained the work.

**Insurance Value**

A insurance value field was not included be a value table was made. This is because the insurance value should be temporal data, as the value may change over time for reasons such as the work becoming part of a travelling exhibition, inflation, a work being damaged, etc. I entered records for my insurance values of works into my values table with the current date, because I knew the works have an insurance value of what was stated in the works table on the current date. My database had a lump insurance value for travelling exhibitions which was equal to twice the value of each work on the travelling exhibition so for the periods when my works where on travelling exhibitions I entered records for the double the regular insurance value into my value table and records for the regular insurance value scheduled for when the works returned from the travelling exhibition.

**Testing**

I queried the Insurance value table to check for records of the insurance values of my works, as well the changing insurance values while on a travelling exhibition

**Unchanged attributes From Works**

For the worknumid, workcharid, worname, and workdescription, I had these values stored under a different name in my old database and simply checked to make sure they moved into the new attributes.

**PART F: Triggers**

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. For example, when a new record is added to the NS\_ExhibitionWorks, new records should also be created in the NS\_WorkLocations table.

There are three main multi table triggers implemented in our new database

* **Trigger 1:** Activated upon inserting to ExhibitionWorks.

Trigger functionality: When a new work is added to the ExhibitionWorks table (when a new work is added to an exhibition), the trigger function will trigger the following events:

1 - Locates the most recent record in the WorkLocations table which has the same CharID, NumID and DataEntryEntryMuseum and updates the attribute wolWorkTimeDeparture to be equal to the date the work was added to the exhibition(i.e. equal to NEW **exwDateWorkAdded**  attribute in **ExhibitionWorks table**).

2- Inserts a new record into the WorkLocations table which records the work as being in the location of the exhibition at the same time as it was entered into the exhibition(ex. If a work is entered into ‘Summer Exhibition’ at 12:00 pm July 1, which takes place in Gallery D, then a record is inserted into the workLocations table which says the work was in Gallery D at 12:00 pm July 1)

This is so records are not left with inaccurate ending dates. If the end date is left null then it appears that a work is still in the previous location. Because the value can be left as null, a user of the database could easily forget to update the end date of the previous work location. As well a user should not be expected to update the previous end date because this is a tedious task which can easily be accomplished by a trigger.

* **Trigger 2:** Activated upon inserting into WorkTransaction

When a new work transaction record is added to the NS\_WorkTransaction table with a transaction type of Sold, Loaned, or Returned, the trigger function will trigger the following events:

1- Update the **wonOwnershipEndDate** attribute in the **WorkOwners** table for the same specified Work of the transaction to be equal to the NEW **wtrTransactionTime** attribute in **WorkTransaction**.

This is because a transaction involves changing the ownership of a work. If for example a partner museum sells a work it makes no sense to keep the work recorded as belonging to the museum. A user would expect ownership in the system to change upon recording a transaction involving the change in ownership of a work and it would be tedious and redundant to also change the ownership for every transaction.

2- Insert a new record in the **WorkOwners** table for the specified work with an owner name equal to the new transaction client (the **wonOwnerName** attribute in **WorkOwners** is equal to the **wtrClient** attributein the **WorkTransaction)**, with an ownership start date equal to the added transaction time (i.e. wonOwnershipDateStart in **NS\_WorkOwners** table equal to the **wtrTransactionTime** attribute in **NS\_WorkTransaction),** and wonOwnershipDateEnd equal to NULL (NULL indicate that this new owner currently hold the item in his possession).

As indicated previously this is to allow easability of transactions. A user of the database should not be expected to change ownership of a work as this is tedious and prone to error. An error in not recording ownership could potentially lead to major legal problems if for example the museum is still claiming ownership for works or a customer would like a history of works which he or she came into possession to from the museum.

This trigger does not however, change the location of a work, because the owner of a work may change while the location remains the same (If for example, someone purchases a work but chooses to continue having it displayed in the museum). It would be inaccurate then to design a trigger to change the location upon a transaction.

* **Trigger 3**: Activated upon inserting into ExhibitionWorks

When a new work is added to the **ExhibitionWorks** table, where the exhibition referenced by the record is a travelling exhibition (when a new work is added to an exhibition), the trigger function will trigger the following events:

1- update the **winInsureDateEnd** attribute in the **WorksInsurance** table to be equal to **exwExhibitDateStart** attribute in **ExhibitionWorks** table.

2- Insert a new record into the **WorksInsurance** table for the specified added item with a **winWorkInsureValue** equal to 110% of its last insurance values, winInsureDateStart equal to exwExhibitDateStart in **ExhibitionWorks**, and winInsureDateEnd =NULL.

This is to ensure safety for the value of works for travelling exhibitions. As a works insurance value is guaranteed to go up to 110% of its current value when on a travelling exhibition by the insurance company, the accurate insurance values should always be recorded. This trigger does not allow for errors in recording of insurance values and thus leads to less problems with insurance claims when handling the database.

**PART G: Transactions for inserting, updating, and deleting data**

This section of the report is focused on the set of transactions developed for our database. They are divided into two sections, transactions that were used in our own previous databases, and new ones that were written for our new database design. They are now outlined below:

**Part 1: Transactions that were used in prior museum databases**

**Insertion Transactions:**

**Insert into doors:** This is used to insert a new entry into the doors table. This form allows entry into each field of the table manually, to insert new paths, such as when a new gallery added, so a new path would possibly be needed. This transaction then inserts the user inputted data into the appropriate table in the database.

Tables Affected: Doors

Testing: This table was tested by attempting insertion with valid and invalid values to ensure that the transaction/script worked properly.

**Insert into ExhibitionLocations:** This is used to insert a new entry into the exhibitionlocations table. This form allows the assignment of an exhibition to a specific location or locations, so it is useful for inserting into the database where an exhibition will be held. The form allows for the input of information into the database in a friendly user interface.

Tables Affected: ExhibitionLocations

Testing: Testing was conducted by inserting into the database some test data, and viewed in the database to ensure that it works properly.

**Insert into Exhibitions:** This is used to insert a new exhibition into the database system. This can specify the name and details of an exhibition.

Tables Affected: Exhibitions

Testing: Testing for this table was accomplished by submitting test data and querying the table to ensure that everything was inserted into the database correctly.

**Insert into exhibitionworks:** This transaction script is used input a work that will be a part of a certain exhibition. It allows every field to be inputted for more meticulous editing.

Tables Affected: exhibitionworks.

**Insert into Locations:** This transaction script allows the entry of new locations into the database system, whether they may be for new museum locations, or places that a travelling exhibit may visit. Every field is accessible for insertion for more meticulous editing.

Tables Affected: Locations

Testing: Testing for this was accomplished by using test data, in association with other scripts to produce data that was queried to ensure that it was input properly.

**Insert into owners:** This transaction script allows the entry of new owners into the new database. This is done through a graphical form that allows the customization of information for every field. This can be used for situations such as transactions that involve someone new coming into the museum’s database.

Tables Affected: Owners

Testing: This was tested like the other scripts by inserting test data and querying the applicable table to ensure that the data was submitted correctly.

**Insert into sponsorexhibitions:** This script allows for the insertion into the database of a sponsor of an exhibition, including the entry of how much money that the sponsor may have donated. The script displays a form which allows for database entry, and once filled, the new information is inserted into the sponsorexhibitions table.

Tables Affected: sponsorexhibitions

Testing: To test this script, a test sponsor was added for one of our exhibitions, which when queried, did properly link up to the information that we had inputted.

**Insert into themes:** This script allows for the insertion of new themes into the database and which works relate to them. This allows the input of new themes which aids in the sorting of works. Due to the database constraints, nothing will be submitted if the work doesn’t exist.

Tables affected: Themes

Testing: New themes were inserted for certain works, and then were queried and displayed correctly, signifying that the inputted information had been input into the database correctly.

**Insert into worklocations:** This allows the manual insertion of the information of where a work was into the database. It allows entry for all fields, and is more administrative in nature, allowing for the meticulous input of new worklocation data.

Tables Affected: worklocations

Testing: Testing was accomplished by adding test data and querying the worklocations table to ensure that the new data was properly added.

**Insert into workowners:** This allows for the manual insertion of new data into the workowners table. This can be used to insert new data about the owner of a work, when a work is inserted, or when a work changes owners.

Tables Affected: workowners

Testing: Testing was done by inserting new data into the workowners table using test data and querying the respective table to ensure that it was properly put into the database.

**Insert into works:** This allows the input of new museum works to the database, it allows for all the fields to be filled in. This allows for insertion of new works with all applicable data fields available to be filled in.

Tables Affected: works

Testing: This was tested normally, by inserting test data into the works table using the transaction, and querying to ensure that the data was correct.

**Insert into worksinsurance:** This allows for the entry of data into the worksinsurance table. Every field is editable for meticulous insertion if necessary. This generally won't happen all that often, but it is handy to have in emergencies.

Tables Affected: worksinsurance

Testing: Data was inserted and the subsequently queried from the worksinsurance table to ensure that data was inserted correctly.

**Insert into worksmedium:** This allows for the insertion of new medium types for works in the database. This allows the museum to insert new mediums for the sorting of works.

Tables affected: worksmedium

Testing: Testing for this was accomplished by inserting test data into the database and querying the data to ensure that it was input correctly.

**Insert into worktransaction:** This allows the insertion of manual transactions into the table. Traditionally it would not happen all that often, but this is more intended to add transactions manually in the possible case of a power outage, etc.

Tables Affected: worktransaction

Testing: Testing was accomplished by submitting a test transaction and ensuring that all the data was submitted into the museum’s database correctly.

**Deletion Transactions:**

**Delete from doors:** This allows a user of the system to delete from the doors table. This can be useful if the structure of a building changes. It may not be used often, but it is useful to have.

Tables Affected: doors

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from exhibitionlocations:** This allows a user of the system to remove an entry from the exhibitionlocations table. This may be done if an exhibition is suddenly forced to change locations, or if incorrect data is entered into the system.

Tables Affected: exhibitionlocations

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed. Originally noticed a slight issue where a certain item would not be deleted, it was corrected.

**Delete from exhibitions:** This allows a user to remove an exhibition from the system. A user may wish to do this if an exhibition is suddenly cancelled before taking place, or if incorrect information is entered into the system.

Tables Affected: Exhibitions

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from exhibitionworks:** This allows the removal of data from the exhibitionworks table. A user may wish to do this while planning the museum works that may go into a certain exhibition that is being planned for the future.

Tables Affected: exhibitionworks

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed. Once this was achieved we knew that the function worked.

**Delete from locations:** This allows the removal of data from the locations table. This commonly would only be used by the user if a location is entered incorrectly into the system.

Tables Affected: Locations

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from owners:** This allows for the deletion of an owner from the owners table. A user of the database may wish to do this if data was entered incorrectly into the database.

Tables Affected: owners

Testing: This was tested by deleting some test data that was inserted, and querying the owners table to ensure that the data was properly removed.

**Delete from sponsorexhibitions:**  This allows the removal of data from sponsorexhibitions. This may be done for various rare cases, but is still necessary to have for the odd case that it may be needed.

Tables Affected: sponsorexhibitions

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from themes:** This allows for data to be removed from the themes table. This allows a user to remove themes that may no longer apply to works or were perhaps inserted incorrectly.

Tables affected: themes

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from worklocations:** This allows a user to remove data from the worklocations table. This will rarely be used, except in the rare cases that past false data may need to be removed.

Tables Affected: worklocations

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from workowners:** This allows data about who a specific work belonged to to be removed from the database. This will rarely be done except in the odd case of false historical data.

Tables Affected: workowners

Testing: An owner was inserted into the table using the counterpart transaction, and this transaction was used to delete it, with a query confirming its deletion.

**Delete from works:** This allows a work to be deleted from the database. This allows a user to find a specific work, and perhaps because of incorrect data, remove that work from the database completely provided that the work is not referenced in another table.

Tables Affected: works

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed. This test was proved successful, so the data was properly removed as expected.

**Delete from worksinsurance:** This allows for the deletion of the insurance history on a specific work in the database. This will rarely be used, perhaps in the odd case that information that is historically incorrect needing to be deleted

Tables Affected: worksinsurance

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Delete from worksmedium:** This allows for the removal of medium data on a specific work. A user may delete one entry from the worksmedium table using this method. A user may wish to do this to remove incorrect data from the database, but due to the fact that museum works traditionally don’t change materials, it likely won’t be used all that often.

Tables Affected: worksmedium

Testing: Testing for this was accomplished by manually inserting a new medium to an existing work, and then using this method to delete it, finally querying the specific table to ensure the data was removed.

**Delete from worktransaction:** This allows the deletion of specific transactions from the museums database. Generally this will only be used in the rarest of cases, and by a system administrator as transactions should usually be kept on record.

Tables Affected: worktransaction

Testing: This was tested in conjunction with its corresponding insert function. Data was put into the table and this was used to remove said data, with a query then being done to confirm that it was removed.

**Update Functions:**

**Update exhibitionlocations:** This allows for the updating of data on where an exhibition is held, in the case that an exhibition may change locations or if data was input into the system incorrectly and needed to be changed. It also allows all data of it to be edited if need be

Tables Affected: exhibitionlocations

Testing: Testing was done by editing all the fields of test data and querying the table to ensure that the information had been changed correctly.

**Update exhibitions:** This allows all the fields of data related to an exhibition to be changed by a user. Useful in the case in which more information may be added, or incorrect information needed to be changed.

Tables Affected: exhibitions

Testing: Testing was done by editing test data that was entered into the system, and querying the new data to ensure that it was changed correctly.

**Update exhibitionworks:** This allows the editing of data of a specific entry in the exhibitionworks table. This is useful if data is incorrect and needs to be changed, or possible new information needs to be added.

Tables Affected: exhibitionworks

Testing: Testing was accomplished by changing data that had been inserted for testing purposes, and then ensuring through a report that the data had been changed appropriately.

**Update locations:** This allows for the updating of information on a location. Each field is editable for various purposes, such as when more data is obtained or if a location gets a new owner.

Tables Affected: locations

Testing: Testing for this was done by creating test date, and editing it, finally doing a report on the test data to ensure that it was changed correctly.

**Update owners**: This allows for the editing of data within a specific record of the owners table. A user may want to do this if more information is added, or if data needs to be changed for whatever reason.

Tables Affected: owners

Testing: Testing was accomplished by making edits to test data, and then querying the owners table to make sure that the new data that was updated into the database was there correctly.

**Update sponsorexhibitions:** This allows the updating of information in the sponsorexhibitions table. A user may wish to do this if a sponsor perhaps donates more money to a specific exhibition and the data needs to be changed.

Tables Affected: sponsorexhibitions

Testing: Testing on this table was accomplished by editing the amount of money that a specific sponsor had donated to an exhibition, and then querying to ensure that the update had occurred properly.

**Update themes:** This allows the name of specific themes that relate to a work to be updated. A user may wish to do this if new themes replace old themes from within the system.

Tables Affected: themes

Testing: A theme for a specific work was changed in the themes table, and the data was queried to ensure data had updated properly.

**Update worklocations:** This allows a user to manually update or change data in the worklocations table. They may wish to do this to update data in the database in the case that something changes. This was primarily used to add an end date

Tables Affected: worklocations

Testing: Testing was accomplished by editing the end date of a worklocation record and ensuring through a query that the data was changed correctly.

**Update workowners:** This allows a user to manually change data related to the owner of a work. This temporal data likely wont be changed often, but can be used to update the final date field to keep relevant temporal information.

Tables Affected: workowners

Testing: Data was entered and edited with this function, and queried to ensure that data was changed as intended.

**Update works:** This allows a user to add or change information about a specific museum work. They may wish to do this if more data about the work was obtained or if incorrect data was added and needed to be changed.

Tables affected: works

Testing: New data was inserted for a work that was obtained, and the work was queried from the table to ensure that the new data was entered correctly

**Update worksinsurance:** This allows a user to update data in a specific record in the worksinsurance table. A user will primarily use this to add an end date when a works insurance value changes but it may be used for other purposes as well

Tables Affected: worksinsurance

Testing: Test data was updated using this method and subsequently queried to ensure that the information changed correctly.

**Update worksmedium:** This allows a user to update data in a specific record in the worksmedium table. This commonly wont be used but it may be used if more data is obtained, such as if a metal is discovered to be a more specific element.

Tables Affected: worksmedium

Testing: This testing was accomplished by editing test data and querying that corresponding data to ensure that it was changed correctly.

**Update worktransaction:** This allows the updating of specific transactions on the odd cases that something needs to be changed.

Tables Affected: worktransaction

Testing: A test transaction record was created and edited using this function, and then queried to ensure proper data editing.

**Part 2: New Transaction Structures**

**Museum Purchasing a work**: This is a multi-table transaction that is higher level, allowing the database to process the purchase of a new work for the museum. In a single form, a user can enter the information of a work, and it will automatically be entered into the works table, recorded as a new insurance record in the workinsurance table, and be logged as a purchase transaction in the worktransaction table.

Tables Affected: Works, worksinsurance, worktransaction

Testing: Testing this required using the form to input data into the database, and then querying the three respective tables to ensure that data was inserted into the table properly.

**Museum selling a work:** This is a multi-table transaction that is meant to be user friendly to process selling a work that belongs to a museum. Through filling out the form, the data entered by the user automatically logs the work into the worktransaction table as being sold, and creates a new record of ownership and sets and end date of the previous record in the workowners table.

Tables Affected: workowners, worktransaction

Testing: This was tested by inputting a test work labeled AAAA-4444 into the database system, and using this function to sell the work to another owner. The appropriate tables were then queried to ensure that the data had been added and updated appropriately.

**Loan out a work:** This is a multi-table transaction that is meant to be user friendly to allow a user to process the loaning of a museum work to another museum. Through filling out the form, a transaction is created, and the location of the work is updated in the worklocations table by inserting the new location and setting the end date for the old location.

Tables Affected: worktransaction, worklocations

Testing: This was tested by inputting a test work into the system, and then loaning it to another owner and location. The data was then queried to ensure that it was logged as a transaction and that the work changed locations.

**Borrow A work:** This is a multi-table transaction that is meant to be user friendly. It allows a user to create a log in the system of borrowing a work. By filling out the form, the work being borrowed is recorded as a transaction and its location is updated and previous location ended, similiar to the loaning of a work

Tables Affected: worktransaction, worklocations

Testing: This was tested by inputting a test work into the system, and then borrowing it from another owner and location. The data was then queried to ensure that it was logged as a transaction and that the work changed locations.

**Inter-museum selling:** This enables the processing of transactions for works that are sold between 2 museums within the conglomerate. This will involve a user filling out a form, which when submitted will automatically change over the ownership of the work, move the work to the purchasers storage area (or another) and log it as a transaction in the transactions table.

Tables Affected: workowners, worklocations, worktransaction

Testing: Testing involved putting in a test work and using this transaction, then subsequently querying the appropriate tables to ensure that the data had been created and processed properly.

**Selling a work to a new owner:** This transaction is meant for a new owner purchasing a work. Throughout filling a single form, a new owner is added to the system, the ownership of the work updated and transferred to them, and the transaction of this logged in the transactions table.

Tables Affected: Owners, Workowners, Transactions

Testing: Testing for this function was accomplished by producing test data and querying the applicable tables to ensure that the data had been created and input properly.

**Part H**

This section of the report is dedicated to the queries and views we have for obtaining information from the database.

**Old Queries & Views**

All of us had queries from before the central database merge: these were adapted for the new schema we have made. We have a view for every single table we have, as well as few that were made for a few basic views that would cover simple use cases. These are:

* A view for the curator to see all items currently in storage, and what they are appraised at.
* A view for the public to see all the exhibits, and how big they are, and where they are located.
* A view to show to public information on all works, based on which exhibit the work is in.
* A view to show which works are free for use, and when.
* A view to show how many more works we can add to exhibitions

**New Queries and Views**

We have expanded our capabilities accordingly with a lot of new queries and views that cover a lot of common use cases that would come up for the curators, managers, and employees of our museums. They fit into a few categories:

**Current item status**

While we have a lot of information on where our items have been and who has owned them over time, a lot of employees will want information on what their current status is, so they can find a piece or check its current exhibition. We have created several views to accommodate this:

* A view that states the current locations of all works
* A view that states its current exhibition of all works
* A view that states the current insured value of all works
* A view that summarizes all of these.

(and several things in the background to handle these).

In addition we have created procedures that can look up any item based on its identifiers(CharID,Numid, and initalEntrymuseum) to accommodate a look up system. If work was continued on the system going forward, these could be easily hooked up to a GUI for employees to use.

**History Lookup**

Our system can hold a lot of information about what happens to works over time. If anyone needs to look up this information (to check the paper trail of a work for insurance purposes, for example), our system can accommodate those needs with a set of functions:

* A function to look up the entire ownership history of an item.
* A function to look up the entire insurance history of an item, which is useful when an item is on a travelling exhibition,
* A function to look up the entire movement history of an item, to track when and where it is moved to.

**Daily and period inventories**

Sometimes, a curator or manager may want to see what business has happened over a particular day, or extended span of time. If multiple works are being bought, sold and loaned out of the system in short period of time, a summary would be helpful in checking all of them. We have accommodated this with a set of functions:

* A function to look up all transactions that occurred on a particular date as well as a variant to cover a user-definable period of time.
* A function to look up all ownership changes that occurred on a particular date as well as a variant to cover a user-definable period of time.
* A function to look up all work movements that occurred on a particular date as well as a variant to cover a user-definable period of time.

**Global Search Functions**

In addition, we have also added a few simple, common search functions to supplement the older ones we have to cover a few of the new functionalities in our central database. These are:

* A function that displays the current status and location of all works in a user-given museum.
* A function that shows all works of a given physical type in a given museum.
* A function that shows all works in a given exhibition.