CMPT 355 Class Project

Team 6

Daniel Morris

Kevin Noonan

Qi Guo

Wei Zhang

Nam Pham

**Part A**.   
Analyzation of each our current database:

group project e-r parta high level (v2) (2).png

According to the E-R diagrams of each of us database,   
**Works:**

All of our museum have work identifiers, work name, work type, work subtype, creator for each work, completion date of each work, acquisition date of each work, insurance value of each work, description of each work and ownership of each work. Hence we suggest that we should keep the above information in the new database as works information.  
 Two of the databases have only one medium for each work. But others have multiple media information associated to each work. Thus we suggest that we could combine all the media information together in one media table to suit all the museums databases.  
 Other than that, each of our museums all has a attribute described the expertise of each museum specializes in. This attribute help to identify an important characteristic of our works in each museum: One museum’s characteristic attribute is about identifying whether the work are abstract or not. And One museum’s characteristic attribute is about identifying whether the work are aboriginal or not. And One museum’s characteristic attribute is about identifying whether the work are poetry or not.  
Hence we suggest that the new database’s works table would add one attribute called ‘wk-question’ contain all museum’s characteristic attributes information. And add another attribute ‘wk-answer’ contain the data corresponding to that characteristic attribute information.   
In this way, we only add extra two attributes to represent all 5 different characteristic attributes   
information.

**Work Keepers:**

And we also have same information for workskeeper. It is the institutions and owners information. Hence we suggest to combine this two information together. A new table called worksKeepers to store all the ownership information.   
We have the owner’s name, the work identifier, the start date and the end date when the work owned, and the owner contact like: the address, the email address and the phone number.   
  
**Exhibitions:**

For the Exhibitions information. All of us have the name, the start date, the end date and the description of the exhibition. Moreover, we find the exhibition need to mentioned if it is a travelling exhibition or not, hence we suggest to create a new attribute to indicate that.

**Locations:**

For the location information, all of our museum have the location name in our museum, and we find the location name is same for some museums. Hence we suggest to add a new attribute to indicate when different museum have same location name.   
Also we find the travelling exhibition have multiple locations, and all our current database regard the temporal location name as location name. Hence we suggest to connect travelling exhibition information to the locations and exhibitions. All the travelling locations information can be find in the locations.

**Doors:**

For doors information in each of our museum, we realize that we have same locations names for some museum, hence we suggest adding museum name information in door table to distinguish different locations.

The first location the door connect from, the second location the door connect to. And make all of the above 3 attributes as primary keys. In order to distinguish different museums’ locations

Below are a high level diagram and a lower level diagram for Part A.  
group project e-r parta (v4).png

**Part B**.

The database is being made such that the history of the database is stored at all times. The current database design allows for the history of **work locations**, **work keepers**, **Temporary Exhibitions Locations, exhibitions**, **exhibition locations**, and **exhibition works.** Throughout the database design this temporal data is made possible through the use of start dates and end dates. The start date is always necessary and the end date is optional based on whether or not the end date of the temporal attribute is known. The following is a breakdown of how each temporal structure will be implemented:

**Exhibitions:**  When exhibitions are created it is necessary that their start date is specified. This start date will be essential for storing the temporal location of the exhibition and the works showcased in the exhibition. The end date of an exhibition is not necessary because it is sometimes not planned for when an exhibition will end.

**Exhibit Locations:**  Once an exhibition is made the location of the exhibition can be set using this table. The location of an exhibit will always be known. Since exhibitions can be either travelling or non-traveling this will need to be implemented into this table. If the exhibition is non-traveling the database will use the start date and end date of the exhibition for the start and end date of this exhibit location. If the exhibition is traveling, then the start date and end date of when the exhibition will be in the specified location will also need to be stated. When data is being imputed to this table the database will perform checks to ensure that conflicts do not occur in the database such as dates overlapping where they are not supposed to be. Also when a location for an exhibition is set or changed the database will make the necessary changes to all the works that have already been placed into the exhibition.

**Exhibit Works:**  Once an exhibition has been created in the museum then works can be scheduled to be placed into it. When works are entered into an exhibition the database will check to see if the exhibition has a specified location. If the location for the exhibition is specified, then the database will update the works locations table to account for where and when the works will be in the location specified.

**Work Locations:**  The location of a work will always be known. The works locations table will contain the start date-time and end date-time for a work’s location. A work’s location is necessary to be tracked at a minute to minute basis due to insurance reasons. A work’s location can be either inserted manually or automatically. As previously stated the works location will be automatically updated when the work is in an exhibition with a specified location.

**Work Keepers:** The keeper of a work will always be known for he can be the lender or the borrower of the item. Since the keeper of a work can change it is necessary for insurance reasons that the history of all works is recorded. When a keeper of a work is defined it is necessary that the start date is specified but the end date is optional since it is often not known when a keeper will not be responsible for a work anymore.

**Temporary Exhibitions Locations:** When travel exhibitions are created, it is necessary that

their start date is specified. The temporary location always need to keep track the data of the sponsor, security, the insurance value and the address when it is on the travelling because when a temporary can be used multiple times in many exhibitions, its other data can also be changed. So, we would want to keep track of the history of being used of a temporary location.

Through further analysis of the database it has become apparent that our preliminary

database design does not account for some other data that could also be temporal. This data

includes the state of the work and the insurance value of the work. The state of the work

includes things such as “In good condition”, “Stolen”, “Damaged” etc. Since the state of the

work could change it is important that all the previous states of each work are recorded. It is

also apparent that the insurance value of a work could change as well. It would be nice to see

the previous value for a work so that functionality will be added to the database. The following is a breakdown of how each temporal structure will be implemented:

**Work State:** When a work is first entered into the museum it’s state will also be inputted presumably as “In good condition”. The start date of the work’s state will be set to the date of acquisition of the work. If the state of the work changes then the end date of that state will be saved and a new state for that work will be saved.

**Work Insurance Value:** When a work is first entered into the museum it’s insurance value will also be inputted. The start date of this value will be set to the date of acquisition of the work. If the insurance value of the work changes then the end date of that value will be saved and a new value for that work will be saved.

Adding the functionality for these two extra temporal data types requires the creation

of two more tables that relate to works. The following is a diagram of how that relationship would look. The new tables are highlighted to easily show where they fit into the whole database design.

e-r Works WorksState WorksInsuranceValue.png

group project e-r partb (v1).png

**Part C**.

This section of the report will be split up into two sections. The first section will be describing the purposes of the domains we have chosen for the new database structure. The second section will be describing the tables and attributes and their individual purposes and design.

**Domains:**

For each new domain that we created for the database structure we made sure that the largest possible value that could be represented was the largest type value out of all of our databases so that we have no errors when combining. Also for the domains we used a general naming system that follows using the table without plurals as the prefix followed by a good representation of the data combined by the five databases. Additionally there is one domain that is not created under a table and that is the museumName domain. The reason for this is it is represented in a various amount of tables and will always be the same value in each case.

*Works Table:*

In the works table the domains are fairly similar to the attribute as there aren’t very many differences between what the attribute represents throughout all five databases. We made the names for each domain as simple as possible to know exactly what the domain represents. The main change we made to one of the domain names was the artist domain. Since four out of five of our museums are art with the other being a car museum we decided on the domain creator for the artist domain. Additionally two domains we have “question” and “answer” were named as such as they are exactly what they represent. This was the only way we could think of to represent the characteristic entity that was found for each of our work descriptions. Our domains are as follows in order of the E-R diagram list: IDAlpha, IDNumeric, workName, type, subtype, creator, completionDate, acquisitionDate, insuranceValue, workdescription, ownership, question and answer.

*Media Table:*

The media table only has one new created domain. This domain is the media domain and we named it material. Each of our works is made of materials so it was a logical decision to name the domain. The two domains that are not created in this table are represented by foreign keys connecting tothe works table.

*WorksKeeper Table:*

The worksKeeper table has seven new attributes that are coming from different tables in different databases. Most of the databases share the same idea for the domain names though. We decided to use the tables name as a prefix for the domains that had been previously used in the database such Name and Date. These are domains are appropriately named workKeeperName, workKeeperStartDate, workKeeperAddress, workKeeperEndDate. Additionally there are three attributes that do not share the prefix mainly because the suffix is not used anywhere else in the database. These domains are email, phoneNumber and status. The two domains that are not created in this table are represented by foreign keys connecting to the works table.

*WorksLocations Table:*

The worksLocations table creates two new domains to represent the date attributes with a timestamp. These domains are named with the prefix of the table and the representation of what the domain is. They are workLocationStartDateTime and workLocationEndDateTime. The four domains that are not created in this table are represented by foreign keys connecting to the works table and the locations table.

*Locations Table:*

The locations table creates five new domains with one of the domains representing a tab. Most of the domains created for the locations table do not follow the prefix system except for locationName. This is because the domains names are only used in the locations table. The domains are as follows locationName, dimension, minNumWorks, maxNumWorks, availableDate. The dimension domain is different as it is of type room. Room is a type we created to have a tab that holds the length, width and height of the room.

*TempExhibitionsLocations Table:*

The tempExhibitionsLocation table creates six new domains. Three of the domains are represented by the table as a prefix. Those domains are tempExhibitionLocationStartDate, tempExhibitionLocationEndDate, tempExhibitionLocationAddress. The other three domains are named for what they represent and the name only exists within this table. They are sponsor, security and insurance.

*Exhibitions Table:*

The exhibitions table creates five new domains. Four of the domains are represented by the table exhibitions as the prefix as the names exist elsewhere in the database. These domains are exhibitionName, exhibitionStartDate, exhibitionDescription and exhbitionEndDate. The remaining domain is the isTravelling domain that represents whether the exhibition is travelling or not.

*WorksInsuranceValue Table:*

The worksinsurancevalue table creates two new domains. Both of these domains use the table as a prefix and both represent a date value. The names are workInsuranceValueStartDate and workInsuranceValueEndDate.

WorksState Table:

The worksstate table creates three new domains. Two of the domains use the table as a prefix and both represent a date value. The names are workStateStartDate and workStateValueEndDate. The remaining domain is the state domain which is simply named to represent the state value.

**Tables and Attributes:**

For each of the tables the attributes are represented in taking the short form of the table name and representing that as a prefix for the attribute. The suffix is then the domain name.

*Works Table:*

The works table is the most important table in the new database. It has a one to many relationship with the primary keys wk-IDAlpha and wk-IDNumeric to the rest of the connecting tables. This was the easiest way to go about the connection between all of the works tables. Additionally we decided to remove the insurance value attribute. This was a good move for the future of the database as the insurance value attribute is temporal data. We decided to create a column for the question to our characteristic value wk-answer. This solved the problem of trying to find a good name for the characteristic column. Instead we can generalize the answer and create a specific question for each characteristic. Two of the databases we are combining with also had a media attribute in the works table. These have been removed and created in a separate table as it would only create blank spaces if left in the works table.

*Locations Table:*

The locations table is the table that stores all of the works, exhibitions, temporary exhibitions and doors locations throughout each individual museum. It has a one to many relationship with the primary keys lc-Name and lc-museumName to the rest of the connecting tables. This was decided because each of the locations tables have to use the location name and the associated museum name to find exact location of the exhibition, work and or door. We decided to add the dimension attribute to the locations table in the form of a tab type. This created a more organized value for all the of the museum size that each individual database had. We also created two date attributes within the table that represent the date of a temporaryExhibitionLocation. This attributes are lc-telStartDate and lc-telEndDate. The lc-telStartDate attribute is possibly and foreign key to temporaryExhbitionsLocations table the depending factor is whether or not the location value is not a part of the museum and is a temporary location.

*WorksLocations Table:*

The worksLocations table is the table that contains the works values and associates those values with the locations that they are placed in. The table has a one to one relationship with both the works table and the locations table as there can be more than one work and more than one location. Additionally the worksLocations table also stores the values of the start and end date that the work is in that location with a timestamp to know exactly when that work has been moved. The reason why the attribute was designed that way is because the works move around so fast that it’s possible a work could move in one day and we would need to narrow that down to a time and place. We also add the museum name as a foreign key to locations so that we could narrow down the location to a specific museum.

*Doors Table:*

The doors table is a table that contains the locations that a door is located in and connects to. This table also stores the museum that the location is in. Originally we had planned to have the door named after numbers but decided that would become confusing even with the museum narrowing it down. Instead we got rid the door number attribute and simplified it so that we have two locations and the door is between and the museum those locations are in. Additionally the doors table has a one to one relationship with the locations table as there can be multiple locations and museums.

*WorksKeepers Table:*

The doors table is a table that contains the locations that a door is located in and connects to. This table also stores the museum that the location is in. Originally we had planned to have the door named after numbers but decided that would become confusing even with the museum narrowing it down. Instead we got rid the door number attribute and simplified it so that we have two locations and the door is between and the museum those locations are in. Additionally the doors table has a zero to many relationship with the locations table as there can be multiple locations and museums.

*TempExhibitionsLocations Table:*

The tempExhibitionsLocations table is a table that contains the information of the exhibitions that are travelling outside the museum to different locations. This table is essentially an extension of the locations table. The reason for separating the table was for the additional attributes that came with the temporary locations. We needed to add a sponsor, security and insurance value to the travelling exhibition. Also this table has a location value and musuem value. The location value tel-lcName has was made so that the location can be a city or town. The museum attribute tel-lcMuseumName is there so that the travelling exhibition is still associated with that museum no matter the location it is in. This table made it difficult to keep track of the location of the exhibitions and works. To solve this issue we created start and end date values. Tel-tempExhibtionLocationStartDate is a primary key in this table as there will always be a different start date for each exhibition and it will never be null. Additionally tel-tempExhibtionLocationEndDate was also created but as a regular attribute. This attribute has the possiblility to be null as we may not know when an exhibition in the future is going to end as well as the possibility of exhibitions ending at the same time. This table also has a zero to many relationship with the locations table as there can be many locations, museums and different times.

*Exhibitions Table:*

The exhibitions table is a table that is the main point for the works that are being showcased in the exhibitions and the locations that the exhibitions are being held in. The original attributes for this table that have been kept the same are the exhibitions name, description, start date and end date. The two attributes that have been added are the ex-museumName because we would like to know which musuem the exhibition is taking place and the new boolean attribute isTravelling. The reason behind the isTravelling attribute is because in the exhibitionsLocations table the el-exStartDate attribute could be different than the ex-exhibitionStartDate attribute in the exhibitions table. The isTravelling attribute was made to solve this issue so that we can check if the exhibition is travelling. If it is and the value is true then value for el-exStartDate in ExhibitionsLocations table will possibly be different than the value in Exhibitions. If it is false then both attributes will be the same date value. Additionally the table has one to one relationships with the primary keys ex-exhibitionName, ex-exhibitionStartDate and ex-museumName to it’s connected tables.

*ExhibitionsLocations Table:*

The exhibitionsLocations table is a table that contains the information of the exhibitions and the locations they are placed in. This table cause alot of issues in terms of dates and time. We found a way to accurately track the date of each exhibition and it’s location by creating four different date attributes. These attributes are el-exStartDate, el-exEndDate, el-lctelStartDate and el-lctelEndDate. The first two are the dates that are in relation to the exhibitions table. The start date is the foreign key that will possibly be different depending on whether the exhibition is travelling or not. If it is then that date will go to el-lctelStartDate instead so we can keep track of that date. Each end date has the possibility to be null and or the same value as another end date. Additionally another attribute that has been added to this table is the ex-lcexMuseumName which we can use to keep track of which exhibition is part of which museum. The table also a zero to many relationship with both the locations table and the exhbitions table as there are many possible locations, museums and start dates.

*ExhibitionsWorks Table:*

The exhibitionsworks table is a table that contains the information of the works that are being showcased and which exhibitions they are being showcased in. The only main additions that had to be made to this table was the two date attributes ew-exStartDate and ew-exEndDate. These date values unline other date values throughout the database only apply to the exhibition itself no matter if it is travelling or not. This is our for sure way of knowing what the exhibitions start date and end date are. The end date is still a regular attribute as it has a possibility of being null or duplicate value but the start date does not. The remaming attribute that has been added is the ew-exMusuemName attribute just so we can check which exhibition the museum is associated with. Additionally the table as a zero to many relationship with the works table and the exhibitions table because there can be many works in different exhibitions and different exhibitions with different start times.

*Media Table:*

The media table is a table that contains the material make up of the work that is located in the works table. This is a very simple table, the reason for it’s creation is there more than one material per work which doesn’t work well in the works table. This way we can have a more organized view of the materials per work. In this table are the foreign keys to the wk-IDAlpha and wk-IDNumeric attributes of the works table and the md-material attribute that has the values of the media from the works table. Additionally this table has a zero to many relationship with the works table because there can be many different works.

*WorksState Table:*

The worksstate table is a table that contains the state of the work and when the work was put into that state. We added this table as it was possible for the state of the work to change. This table also has the foreign keys wk-IDAlpha and wk-IDNumeric of the works table and two date values ws-workStateStartDate amd ws-workStateEndDate. This table is purely to serve the purpose of checking whether the work is in good condition or not with the attribute ws-state.

*WorksInsuranceValue Table:*

The worksinsurancevalue table is a table that contains the insurance value of a work if it was stolen or damaged and has now been changed. This table was created when we realized that the insurance value in the works table could be temporal. This table has the foreign keys wk-IDAlpha and wk-IDNumeric to be associated with the work. It also has two dates to track the start date that the insurance value was changed and when it ends if it does. These attributes are known as wiv-workUnsuranceValueStartDate and wiv-workInsuranceValueEndDate. The start date being a primary key value and the end date being a possible null value as we are not sure when the insurance value will drop back down. To track the insurance value we use the attribute wiv-insurancevalue.

group project e-r partc high level (v1).png

group project e-r partc (v4).png

**Part D**.

**1. Kevin Noonan’s report**

Currently my database is comprised of ten tables. The new structure that I will be converting my database to has twelve tables. The new tables that are being added to the structure are the Media table, the WorksState table, the WorksInsuranceValue table and a new combined table called WorksKeepers. The WorksKeepers table takes attributes from the onloanto table and owners table from my previous database to make one organized table. Most of the tables have one or two attributes that need to be changed to accommodate the new structure I will now describe the changes to the tables and the testing requirements for each.

**Works:**

One attribute that will be changed in the works table is the wksdonor. This attribute is not needed as it is already represented in the wksowners table. I will be removing this attribute from the table. Also I will be removing wksengine attribute which represents the media of the table and the wksinsurancevalue. I will be taking the data from the wksengine column and adding it to the new Media table and the wksinsurancevalue will go to the WorksInsuranceValue table. A new attribute that will be added to the works table is the wk-question attribute which adds a question column with the attribute wkscountry being the renamed wk-answer in the new structure. I will test these new changes but creating a query to check for just the columns wk-question and wk-answer. I can also create a query to test for wksengine to see if it still exists. That query should return with no value if it was removed correctly.

**WorksLocations:**

The workslocations table will have two additional attributes added to the table. One attribute will be wl-lcmuseumname which is the museums name that is associated with that location. The second attribute will be and end date time with a timestamp. As a start date time already exists adding an end date time is essential. These two attributes will be tested by using a query to call three columns wl-lcmuseumName, wk-workLocationEndDateTime and wl-lcName and making sure that the first two are associated with the wl-lcName.

**Locations:**

The locations table has multiple changes added to the table. The new attribute will be added called lc-museumName that represents the museums name to that location. Two attributes that currently exist in my table are locroomwidth and locroomlength. These attributes will be deleted and a new domain dimension tab under the attribute lc-dimension will be created, that has the attributes length, width and height. Additionally two new attributes will be added to the locations table called lc-telStartDate and lc-telEndDate. Testing for the new lc-dimension will be to query the attribute to see if the proper values are stored in the length, width and height. The lc-telStartDate and lc-telEndDate attributes will be tested by calling the query with those two attributes and the two date values in the table TempExhibitionsLocations to see if they are the same value.

**TempExhibitionsLocations:**

The TempExhbitionsLocations table will have the following additional attributes to the table. The templocadd attribute in my current table holds a city as it’s address. This will be changed to tel-lcname attribute in the new table and the newly named tel-tempExhibitionLocationAddress will contain an exact address. Also a new attribute that will be added to the table is the tel-lcMuseumName which will contain the museum that is associated with the temporary exhibition. The testing for this table will be to create a query that checks the tel-lcname, tel-lcMuseumName and tel-tempExhibitionLocationAddress and cross reference that with the attributes in the Locations table lc-name and lc-museumName.

**Exhibitions:**

The exhibitions table will have two new additional attributes added. The attribute ex-museumName that associates the museum name with the exhibition and the boolean attribute ex-isTraveling that holds the value of if the exhibit is in the musuem or not. These attributes will be tested by using a query to compare ex-isTraveling with ex-exhibitionName and ex-exhibitionStartDate. Additionally these values will be queried with the locations attribute lc-telStartDate. If the ex-isTraveling value is true then the ex-exhibitionStartDate will be five days after the lc-telStartDate. Otherwise the value should be shown as false and the ex-exhbitionName will be located in the museum.

**ExhibitionsLocations:**

The exhibitionslocations table has five additional attributes that need to be added to work with the new database structure. Four date attributes need to be added. Two of these date attributes are for the exhibitions start date and end date which are named el-exStartDate and el-exEndDate. The other two date attributes are for the TempExhibitionsLocations table exhibitions which are named el-lctelStartDate and el-lctelEndDate. The other attribute that needs to be added is the museum name as el-lcexMuseumName that is associated with the exhibition location. The testing for these new attributes will be to create a query the el-exStartDate and el-exEndDate with the el-exName and el-lcexMuseumName to check that the dates match with the exhibitions and the associated museum. I will create a separate query to check the el-lctelStartDate and el-lctelEndDate with the el-exName to check that those dates match with their respective exhibitions.

**ExhibitionsWorks:**

The exhibitionworks table will be have three additional attributes added to the table. The new attributes are two date attributes ew-exStartDate and ew-exEndDate that track the date of the exhibition. The third attribute is the museum name as ew-exMuseumName that is associated with each exhibition that exists. These attributes will be tested by creating a query with the ew-exName and the eq-exMuseumName and make sure that each exhibition is with it’s associated museum. I will then add the ew-exStartDate and ew-exEndDate and make sure those dates match with the associated exhibitions.

**Owners:**

This table will not exist in the new database structure but the values from my database will be transferred to the new WorksKeepers table. These attributes are ownersname, ownersworkssl, ownersworkssn, ownersacq. These attributes will be renamed as wkk-workKeeperName, wkk-wkIDAlpha, wkk-wkIDNumeric and wkk-workKeeperStateDate respectively. The testing for these values will be answered in the WorksKeepers section.

**Doors:**

The doors table will have an attribute removed and then replaced with a new attribute. The attribute that is being removed is the drsnumber attribute. The new attribute will be the dr-lcMuseumName which will hold the museums name for the associated door location. This will be tested by selecting the doors table and cross checking the location attributes are matched with their associated museum.

**OnLoanTo:**

This table will not exist in the new database structure but the values from my database will be transferred to the new WorksKeepers table. These attributes are onloanname, onloanaddress, onloannum, onloanemail, onloanstart and onloanend. These attributes will be put into the following new attributes in the WorksKeepers table: wkk-workKeeperName, wkk-workKeeperAddress, wkk-phoneNumber, wkk-email, wkkKeeperStartDate and wkkKeeperEndDate respectively. The testing for these values will be explained in the WorksKeepers section.

**Media:**

The media table is a new table that is added to the database structure. This table will hold three attributes. Two of the attributes are foreign keys to the works attributes wk-IDAlpha and wk-IDNumeric. These will be named md-wkIDAlpha and md-wkIDNumeric. The third attribute added will be the wksEngine attribute that was removed from the works table to be placed in this new Media table as md-material. This will be test by simply selecting the new Media table and making sure that the md-wkIDAlpha and md-wkIDNumeric matches with it’s respective md-material.

**WorksKeepers:**

The workskeepers table will be created from combing two of my old tables; onLoanto and owners. This new table will have nine new attributes. All of the attributes already exist which is except for one which is wkk-status. This attribute will hold the value of whether the value is loaned, borrowed or potentially borrowed. After that attribute is created it is simply a matter of updating the table and renaming the attributes. The attributes that are being added have been described in the owners and onloanto section. The testing will be to create a query that selects all the attributes and then cross reference them with the old tables onloan to and owners to make sure that the values are correct. Additionally a query will be made with wkk-wkIDAlpha and wkk-wkIDNumber and wkk-status to check that each work has a status of loaned, borrowed or potentially borrowed.

**WorksState:**

The worksstate table is a new table that is created with two foreign keys to the works table. These two attributes ws-wkIDAlpha and ws-wkIDNumeric will come from the wk-IDAlpha and wk-IDNumeric attributes. Additionally there are three other attributes that will be added to this table. Two the attributes are date values named ws-workStateStartDate and ws-workStateEndDate. The other attribute is wk-state which holds the state of the work such as “damaged” or “in good condition”. These will be tested by setting the wk-state of each work to “in good condition”, selecting the table and making sure that each work id is set to in good condition as none of the works have had that value changed yet. Also we will make sure that the end date value is set to null.

**WorksInsuranceValue:**

The worksInsuranceValue is a new table that is created with two foreign keys to the works table. These two attributes wiv-wkIDAlpha and wiv-wkIDNumeric will come from the wk-IDAlpha and wk-IDNumeric attributes. Additionally there are three other attributes that will be added to this table. Two the attributes are date values named wiv-workInsuranceValueStartDate and ws-workInsuranceValueEndDate. The third attribute is the wiv-insuranceValue which will come from the works table. The testing for this will be similar to the WorksState table except no values will be added except for the end date attribute to null. I will then create a query that compares the wk-IDAlpha and wk-IDNumeric with the work attributes and compare the insurance value to each to make that each id has the same insurance value.

**2. Daniel Morris’s report**

There is quite a bit I need to change to my database before it will be ready to merge with the design stated in part C above. Below is a list of the changes and the testing plan for each of the defined tables above compared to my assignment 3 design.

- Works table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **Works** | | wkIdAlpha  wkIdNum | | wkName  wkType  wkSubtype  wkArtist  wkCompDate  wkAcqDate  wkValue  wkDesc  wkBorrowStatus  wkIsAboriginal | | *New group table:*   |  | | --- | | **Works** | | wk-IDAlpha  wk-IDNumeric | | wk-worksName  wk-type  wk-subtype  wk-Creator  wk-completionDate  wk-acquisitionDate  wk-workDexcription  wk-ownership  wk-question  wk-answer | |

There are a few changes to be made to my Works table. The value of a Work will no longer be stored in the Works table. I will create the new table WorksInsuranceValue and create a new entry in that table for each of my works. Since the insurance value of all my works has remained constant I will just insert every work into the WorksInsuranceValue table with their respective insurance values and set the start date to the work’s date of acquisition and the end date to null. My wkIsAboriginal attribute of my works table will now be splitting into two attributes in the new table. wk-question will always be “Is the work aboriginal?” for every work in my database and wk-answer will be the same as wkIsAboriginal: either yes or no.

To test the insurance value attribute changes in the table I will create a query to select the wkIdAlpha, wkIdNum, wkAcqDate, and wkValue from the old Works table and wiv-wkIdAlpha, wiv-wkIdNumeric, wiv-workInsuranceValueStartDate, wiv-insuranceValue from the new WorkInsuranceValue table. I will check to make sure these values match up. To test the wk-question and wk-answer attributes I will select the wkIdAlpha, wkIdNum, and wkIsAboriginal from the old Works table and wk-IDAlpha, wk-IDNumeric, and wk-answer from the new Works table. I will check to make sure these values match up. I will also select every wk-question from the new Works table to check if it is set to “Is the work Aboriginal?”.

- Locations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Rooms** | | rmName | | rmWidth  rmLength  rmSuggMax  rmSuggMin  rmAvailabilityDate | | *New:*   |  | | --- | | **Locations** | | lc-name  lc-museumName | | lc-telStartDate  lc-telEndDate  lc-dimension  lc-minNumWorks  lc-maxNumWorks  ls-availableDate | |

For the locations table I will need to add a new attribute to store the museum in which the location is in. I will set it to the default the name of my museum. The lc-telStartDate and telEndDate will be set to null for each location that isn’t a traveling exhibition. For my traveling exhibition though I will need to add a location for each temporary location. Since I did not implement a startDate and endDate to my travelingExhibition locations I will need to make these dates up. In my Rooms table the rmWidth, and rmLength will be both combined into lc-dimention in the new Locations table.

To test the changes to the room dimensions I will query the old Rooms table selecting the length and width, and the new Locations table selecting the dimension to confirm the date has been mapped accordingly. To test the lc-telStartDate I will query all of the temporary locations in my traveling exhibition and confirm that their start dates and end dates add up to the total duration of the exhibition.

- Exhibitions table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitions** | | exName | | exDesc  exStart  exEnd | | *New:*   |  | | --- | | **exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate  ex-museumName | | ex-isTraveling  ex-exhibitionDescription  ex-exhibitonEndDate | |

For the exhibitions table I will have to add two new attributes. The first is the attribute ex-MuseumName which will be the same for all of my exhibitions. The other is a Boolean value ex-isTraveling, this will be manually set to false for all of my static exhibitions and true for my traveling exhibition.

To test these changes I will select ex-exhibitionName, ex-museumName, ex-isTraveling and check to see that the correct values for these attributes have been set.

Temporal data Table:

- TempExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **travelingExhibitLocations** | | treName FK-ex | | treSponsor  treAddress  treSecurity  treInsurance | | *New:*   |  | | --- | | **TempExhibitionsLocations** | | tellcname FK-lc  tel-lcMuseumName FK-lc  tel-tempexhibitionLocationStartDate | | tel-sponsor  tel-security  tel-insurance  tel-tempExhibitionLocationEndDate  tel-tempExhibitionLocationAddress | |

For the TempExhibitionsLocations table I will also need to add another attribute for the museum name. I will also need to add an attribute for the start date and end date. I will acquire these dates from the previously defined location that corresponds to the temp location.

To test this table, I will first ensure that every entry in the table has the name of my museum in the tel-lcMuesumName attribute. I will then query the location table’s lc-telStartDate and lc-telEndDate and make sure that there is a corresponding tempExhibitionsLocation for all of these locations.

- WorksLocation table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **worksLocations** | | locIdAlpha FK-wk  locIdNum FK-wk  locName  locMoveDate | |  | | *New:*   |  | | --- | | **WorksLocations** | | wl-lcName FK-lc  wl-wkIDAlpha FK-wk  wl-wkIDNumeric FK-wk  wl-lcmuseumName FK-lc  wk-workLocationStartDateTime | | wk- workLocationEndDateTime | |

For the worksLocations table I will need to add a new attribute for the museum name. It was also decided in our group that all of our temporal data be closed-closed temporal data so I will need to convert my closed-open temporal data to closed-closed. This also means adding the wk- workLocationEndDateTime attribute.

To test this table, I will first ensure that every entry in the table has the name of my museum in the wl-lcMuesumName attribute. I will then query my worksLocations table and the new worksLocations table sorting by locIdAlpha, locIdNum, locStartDate ordered by locStartDate. This will allow me to see very quickly if the dates have been converted correctly.

- ExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitLocation** | | exLocName FK – ex  exLocloc FK - rm | |  | | *New:*   |  | | --- | | **ExhibitionsLocations** | | el-lcName FK-lc  el-exName FK-ex  el-exStartDate FK-ex  el-lcexMuseumName FK-lc, ex | | el-lctelStartDate  el-lctelEndDate  el-exEndDate | |

For my exhibitLocation table I will need to add a few new attributes. I will need to add an attribute for the museum name, an attribute for the start date of the exhibition, the end date of the exhibition, and the start and date of the temporary location if this location is a temporary one. The el-lcexMuseumName attribute will be the same for every exhibitionLocation. The el-exStartDate and el-exEndDate attributes will be acquired from the exhibitions table. The el-lctelStartDate el-lctelEndDate will be acquired from the locations table previously defined.

To test this table, I will first ensure that every entry in the table has the name of my museum in the el-lcexMuesumName attribute. I will then query the exhibitionsLocations table with the exhibitions table matching together the exhibition names and will check that their start dates and end dates also match. I will also query the exhibitionsLocations table with the locations table matching together the location name and will check that the tempStartDates and tempEndDates also match.

- ExhibitionsWorks table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitWorks** | | ewIdAlpha FK-wk  ewIdNum FK-wk  ewName FK-ex | |  | | *New:*   |  | | --- | | **ExhibitionsWorks** | | ew-exName FK-ex  ew-wkIDAlpha FK-wk  ew-wkIDNumeric FK-wk  ew-exStartDate FK-ex  ew-exMuseumName FK-ex | | ew-endDate | |

For my exhibitionsWorks table I will need to add a few new attributes. I will need to add an attribute for the museum name, an attribute for the start date of the exhibition,and the end date of the exhibition. The ew-lcexMuseumName attribute will be the same for every exhibitionLocation. The ew-exStartDate and ew-exEndDate attributes will be acquired from the exhibitions table.

To test this table, I will first ensure that every entry in the table has the name of my museum in the ew-exMuesumName attribute. I will then query the exhibitionsWorks table with the exhibitions table matching together the exhibition names and will check that their start dates and end dates also match.

- WorksKeeper table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **workOwner** | | wkoIdAlpha FK – wk  wkoIdNum FK – wk  wkoName  wkoDate | |  | | **workLoan** | | wklIdAlpha FK – wk  wkIdNum FK – wk  wklName | | wklAddress  wklPhoneNum  wklEmail  wklStart\_date  wklEnd\_date | | *New:*   |  | | --- | | **WorksKeeper** | | wkk-workKeeperName  wkk-wkIDNumberic FK-wk  wkk-wkIDAlpha FK-wk  wkk-workKeeperStartDate  wkk-status | | wkk-workKeeperAddress  wkk-workKeeperEndDate  wkk-email  wkkphonenumber | |

The worksKeeper table is the combination of the workOwner and workLoan table. Because this 2 tables have the same attributes and can work in the same way, all we need to do is to add another attribute to know whether or not the items are lent or borrowed. In my workOwner table I had wkoDate as a closed-open temporal data structure. I will need to convert this to closed-closed to work with the new WorksKeeper table.

To test this new table I will query it against the old tables I had and make sure the attributes have been moved over sussessfully. The wkk-status attributed should be borrowed, potentially borrowed, or loaded out.

- Doors table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Doors** | | doorNum  doorRm1 FK – rm  doorRm2 FK - rm | |  | | *New:*   |  | | --- | | **Doors** | | dr-lcMuseumName FK-lc  dr-lcName1 FK-lc  dr-lcname2 FK-lc | |  | |

The doors table doesn’t have many major change. The doorNum was decided to be not needed by our group so I will simply delete it.

To test this table I will simply select all the doors in the table and manually check that all the data is correct.

- Media table

|  |
| --- |
| **Media** |
| medIdAlpha FK-wk  medlIdNum FK-wk  medMedia |
|  |

For this table, it is identical to my current one and have the same primary keys and foreign keys. Therefore, I can transfer my data to this table easily.

**New tables:**

- WorksState table

|  |
| --- |
| **WorksState** |
| ws-wkIDNumeric FK-wk  ws-wkIDAlpha FK-wk  ws-workStateStartDate |
| ws-state  ws-workStateEndDate |

To input my data into this new table shouldn’t be very difficult. All of the primary key attributes can come from the works table. ws-state will be set to “In good condition” for all my works and ws-workStateEndDate will be set to null.

To test this table I can simply select the entire table and make sure that every work is “In good condition”.

- WorksInsuranceValue table

|  |
| --- |
| **WorksInsuranceValue** |
| wiv-wkIDNumeric FK-wk  wiv-wkIDAlpha FK-wk  wiv-workInsuranceValueStartDate |
| wiv-insuranceValue  wiv- workInsuranceValueEndDate |

To input my data into this new table shouldn’t be very difficult. wiv- workInsuranceValueEndDate will be set to null and all the other attributes will come directly from the works table.

To test this table, I will query this table and my old works table selecting the IaAlpha and IdNumeric and the insuranceValue, ordering by idAlpha and idNumeric from both tables. It will then be easy to see if there are any discrepancies.

**3. Qi Guo’s report**

I’ll update my Works, Locations, Exhibitions, ExhibitionsLocations, ExhibitionsWorks tables and add WorksInsuranceValue, WorksState, WorksKeepers tables, TempExhibitionsLocations tables before merger.

- Works table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **Works** | | wkIDLetter  wkIDNumber | | wkName  wkType  wkSubtype  wkAuthor  wkCompletionDate  wkAcquisitionDate  wkInsuranceValue  wkDescription  wkOwnership  wkCharacter | | *New group table:*   |  | | --- | | **Works** | | wk-IDAlpha  wk-IDNumeric | | wk-worksName  wk-type  wk-subtype  wk-Creator  wk-completionDate  wk-acquisitionDate  wk-workDexcription  wk-ownership  wk-question  wk-answer | |

I plan to update my Works table from the following aspects. Remove wkInsuranceValue attribute from Works table and make a new temporal table named WorksInsuranceValue table to keep track of changes in insurance value from time to time. Add a new attribute to Works table named wk-question with a new domain called question, which is a varchar of length 50. “What is the character of this work?” will be inserted to wk-question. Since all group members have different custom attributes from assignment 2, we decide to add an attribute called answer composing of the data from custom attribute to Works table and an attribute with a name of question containing corresponding questions based on answers.

Tests planfor changes in Workstable includes querying all data in newly added attribute wk-answer, along with wk-IDAlpha, wk-IDNumeric from old Works table and new Works table to check whether those data match up and querying all data in wk-question to see if those data are all set to “What is the character of this work?”.

- WorksInsuranceValue table

*New group table:*

|  |
| --- |
| **WorksInsuranceValue** |
| wiv-wkIDNumeric FK-wk  wiv-wkIDAlpha FK-wk  wiv-workInsuranceValueStartDate |
| wiv-insuranceValue  wiv- workInsuranceValueEndDate |

A new table WorksInsuranceValuementioned above will be needed so that the changes in insurance value can be recorded. This table will contain the following attributes: wiv-wkIDAlpha, wiv-wkIDNumeric, wiv-workInsuranceValueStartDate, wiv-insuranceValue, wiv-workInsuranceValueEndDate.

Test plan for changes in WorksInsuranceValue table will be query all data in WorksInsuranceValue table to check if are temporal data are stored correctly.

- WorksState table

*New group table:*

|  |
| --- |
| **WorksState** |
| ws-wkIDNumeric FK-wk  ws-wkIDAlpha FK-wk  ws-workStateStartDate |
| ws-state  ws-workStateEndDate |

A new table WorksStateis needed to keep track of different work states, namely, damaged, stolen, in good condition. This table will contain the following attributes: ws-wkIDAlpha, ws-wkIDNumeric, ws-workStateStartDate, ws-workStateEndDate, ws-state.

Test plan for changes in WorksState table is querying all data in this table to make sure that there are no time gap between ws-workStateStartDate and ws-workStateEndDate.

- Locations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **Locations** | | lcName  lcMuseum | | lcDimension  lcMinNumWork  lcMaxNumWork  lcIsAvailable | | *New group table:*   |  | | --- | | **Locations** | | lc-name  lc-museumName | | lc-telStartDate  lc-telEndDate  lc-dimension  lc-minNumWorks  lc-maxNumWorks  ls-availableDate | |

The following changes are my plans for Locations table. Update domain to lc-dimension from numeric to a custom domain named dimension, which is a set of data namely, length, width and height. Add a new attribute lc-availableDate to take place of Boolean attribute named lcIsAvailable. A new domain called availableDate of type date will be applied to this attribute. Add a new attribute lc-telStartDate to keep track of temporary exhibition locations. A new domain called tempExhibitionLocationStartDate will be applied to this attribute. Add a new attribute lc-telEndDate to keep track of temporary exhibition locations. A new domain called tempExhibitionLocationEndDate will be applied to this attribute.

Tests plan for Locations table includes querying all data in lc-dimension to see if length, height and width are all recorded; querying all data in lc-telStartDate, lc-telEndDate to see if they are consistent with the start date and end date in TempExhibitionsLocationstable**.**

querying all data in lc-availableDate of Locations table and data in el-exEndDate from ExhibitionsLocations table to check if lc-available is the next day of el-exEndDate.

- TempExhibitionsLocations table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *New group table:*   |  | | --- | | **TempExhibitionsLocations** | | tel-lcname FK-lc  tel-lcMuseumName FK-lc  tel-tempexhibitionLocationStartDate | | tel-sponsor  tel-security  tel-insurance  tel-tempExhibitionLocationEndDate  tel-tempExhibitionLocationAddress | |

A new temporal table named TempExhibitionsLocations will be added to keep track of all the changes in locations for travelling exhibitions. This new table involves the following attributes: tel-lcName, tel-lcMuseumName, tel-tempExhibitionLocationStartDate, tel-sponsor, tel-security, tel-insurance, tel-tempExhibitionLocationEndDate, tel-tempExhibitionLoationAddress.

Tests plan for TempExhibitionsLocations table includes querying all data in this table to see if all 5 records are stored.

- ExhibitonsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **ExhibitionsLocations** | |  | | elLocation FK-lc  elExhibition FK-ex  elexStartDate FK-ex  elexEndDate  ellcMuseum FK-lc | | *New group table:*   |  | | --- | | **ExhibitionsLocations** | | el-lcName FK-lc  el-exName FK-ex  el-exStartDate FK-ex  el-lcexMuseumName FK-lc, ex | | el-lctelStartDate  el-lctelEndDate  el-exEndDate | |

For my ExhibitionsLocations table, I will add two attributes, el-lctelStartDate, el-lctelEndDate, to keep track of all the changes in locations for travelling exhibitions.

Tests plan for ExhibitionsLocations table is querying all data in el-lcName, el-lctelStartDate, el-lctelEndDate from ExhibitionsLocations table and all data in tel-lcName, tel-lctempExhibitionLocationStartDate, tel-lctempExhibitionLocationEndDate in TempExhibitionsLocations table to check when el-lcName and tel-lcName are the same, which means that it is a travelling exhibition, data in el-lctelStartDate and tel-lctempExhibitionLocationStartDate, el-lctelEndDate and tel-lctempExhibitionLocationEndDate match up.

- Exhibitions table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **Exhibitions** | | exName  exStartDate | | exDescription  exEndDate | | *New group table:*   |  | | --- | | **Exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate  ex-museumName | | ex-istraveling  ex-exhibitionDescription  ex-exhibitonEndDate | |

There are two changes needed to be done to Exhibitions table. Add a new attribute ex-museumName to keep track of museum name for normal exhibitions and travelling exhibitions. Add a new Boolean attribute ex-isTravelling to distinguish travelling exhibitions from normal exhibitions.

Tests plan for Exhibitions table consists of querying all data in ex-museumName from Exhibitions table and lc-museumName from Locations table to see if they are of the same amount; querying all data in ex-museumName, ex-exhibitionStartDate, ex-isTravelling from Exhibitions table and all data in lc-museumName, lc-telStartDate from Locations table to check if when ex-museumName is the same as lc-museumName and ex-exhibitionStartDate is five days after lc-telStartDate, ex-isTravelling is true; otherwise, ex-isTravelling is false.

- ExhibitionsWorks table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **ExhibitionsWorks** | |  | | ewExhibition FK-ex  ewIDLetter FK-wk  ewIDNumber FK-wk  ewStartDate FK-ex  ewEndDate | | *New group table:*   |  | | --- | | **ExhibitionsWorks** | | ew-exName FK-ex  ew-wkIDAlpha FK-wk  ew-wkIDNumeric FK-wk  ew-exStartDate FK-ex  ew-exMuseumName FK-ex | | ew-endDate | |

For ExhibitionsWorks table, I’ll add a new attribute named ew-exMuseumName for museum name of normal exhibitions and travelling exhibitions.

Tests planfor ExhibitionsWorkstable includes querying all data in ew-exName, ew-exStartDate, ew-exMuseumName from ExhibitionsWorks table and all data in ex-exhibitionName, ex-exhibitionStartDate, ex-museumName from Exhibitions table to check if when ew-exName and ex-exhibitionName, ew-exStartDate and ex-exhibitionStartDate are identical, ex-exhibitionName and ex-museumName are same.

- WorksKeepers table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **Owners** | |  | | ownwkIDLetter FK-wk  ownwkIDNumber FK-wk  ownName |        |  | | --- | | **LoanedWorks** | | lwwkIDLetter FK-wk  lwwkIDNumber FK-wk | | lwInstituteName  lwAddress  lwPhoneNumber  lwEmail  lwStartDateTime  lwEndDateTime | | *New group table:*   |  | | --- | | **WorksKeepers** | | wkk-workKeeperName  wkk-wkIDNumberic FK-wk  wkk-wkIDAlpha FK-wk  wkk-workKeeperStartDate  wkk-status | | wkk-workKeeperAddress  wkk-workKeeperEndDate  wkk-email  wkkphonenumber | |

My Owners table and LoanedWorks table will be merged to a new table called WorksKeepers table since those two old tables contain information regarding either owners or obligors for different works, which have similar functionality.

Tests plan for WorksKeepers table is querying all data in WorksKeepers table and all data in wk-IDAlpha, wk-IDNumeric from Works table to check if all works stored in Works table have associated record in WorksKeepers table.

**4. Wei Zhang’s report**

Descriptions of what changes I should do for each of my table And How I will test that my changes worked.  
1. For the **Works** table, delete the attribute media, because we will create the media table to put media information in there.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **museum** | | muletter  munumber | | muname  mudescription  mutype  musubtype  Muauthor  mumedia  mudate\_complete  mudate\_accquire  muinsurance\_value  muownership  mucharacteristic | | *New group table:*   |  | | --- | | **Works** | | wk-IDAlpha  wk-IDNumeric | | wk-worksName  wk-type  wk-subtype  wk-Creator  wk-completionDate  wk-acquisitionDate  wk-workDexcription  wk-ownership  wk-question  wk-answer | |

Add the 2 new attributes called 'wk-question', 'wk-characteristic', Then in this way we can combine everyone's character of description. The 'wk-question' is about all member's characteristic definition. And the 'wk-characteristic' is the data value for the characteristic.   
And also create the domain for ‘wk-question’ as ‘question’ and make the domain type as varchar(100)   
In addition, make the attribute ’wk-answer’ as ‘answer’ and make the domain type as varchar(50)  
  
~ To test if the changes to works table worked, just query this table by selecting the names of attributes which were deleted and the attributes were added..

To test the media attribute changes in the table query this table to select the wkIdAlpha, wkIdNum, wkAcqDate, and media from the old Works table and wiv-wkIdAlpha, wiv-wkIdNumeric, wiv-workInsuranceValueStartDate, wiv-insuranceValue from the new media table. I will check to make sure these values match up. To test the wk-question and wk-answer attributes I will select the wkIdAlpha, wkIdNum, and wkchatacteristic from the old Works table and wk-IDAlpha, wk-IDNumeric, and wk-answer from the new Works table. I want to make sure these values match. I will also select every wk-question from the new Works table to check if it is set to “Is the work abstract?”.

2. For the **Locations** table, add additional attribute called lc-telStartDate as the location start to use date,add additional attribute called lc-telStartDate as the location end to use date  
And also create the domain for ‘lc-museumname as ‘museumname’ and make the domain type as varchar(50)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Locations** | | lc-name | | lc-dimension  lc-min\_capacity  lc-max\_capacity | | *New:*   |  | | --- | | **Locations** | | lc-name  lc-museumName | | lc-telStartDate  lc-telEndDate  lc-dimension  lc-minNumWorks  lc-maxNumWorks  ls-availableDate | |

Because in this way we can distinguish the same location name with different museum name. Add additional attribute called ‘lc-availabledate’ to indicate when and where the location will be available to use in future. Also create the domain for attribute ‘lc-availableDate’ as ‘availableDate’ as date.  
  
~To test if the changes to locations table worked, just query this table by selecting the names of attributes which are added.

To test the changes to the lc-availabledate I will query the old location table selecting the length and width, and the new Locations table selecting the dimension to confirm the date has been mapped accordingly. To test the lc-telStartDate I will query all of the temporary locations in my traveling exhibition and confirm that their start dates and end dates add up to the total duration of the exhibition.  
  
3.For the **Exhibitions** table, add attribute ‘ex-isTravelling’ to indicate whether the exhibition is travelling exhibition or not.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate | | ex-ex\_descripton  ex-exhibitonEndDate | | *New:*   |  | | --- | | **exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate  ex-museumName | | ex-istraveling  ex-exhibitionDescription  ex-exhibitonEndDate | |

Also create domain isTraveling as Boolean.  
Also drop table exhgala. Because the data in exhgala can get from exhibitionlocation. Exhgala table is redundant.

~ To test if the changes to exhibitions table worked, just query this table by selecting the added names of attributes and the deleted names of attribute. And display the table to see if the changes works. And also display the entire database to see if the exhgala table dropped.

To test the changes to the isTravelling I will query the old travelling exhibition table selecting the exhibitionname and the new exhibition table selecting the ex-exhibiitionname and ex-istravelling to confirm the exhibition name has been mapped accordingly.

4.For the **WorksLocations** table, add additional attribute called 'wk-lcmuseumNme' as primary key and it’s also a foreign key.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **workslocation** | | wl-museumetter FK-wk wl-museumnumber FK-wk wl-current\_location  wl-date\_in | | wl-date\_out  wl-time\_in  wl-time\_out | | *New:*   |  | | --- | | **WorksLocations** | | wl-lcName FK-lc  wl-wkIDAlpha FK-wk  wl-wkIDNumeric FK-wk  wl-lcmuseumName FK-lc  wk-workLocationStartDateTime | | wk- workLocationEndDateTime | |

Because other member's museums can have the same locationname.   
Also change the domain workLocationStartDate to workLocationStartDateTime as timestamp;  
Also change the domain workLocationEndDate to workLocationEndDateTime as timestamp;  
  
~ To test if the changes to workslocations table worked, just query this table by selecting the added names of attributes. And display the table to see if the changes works.  
  
5. For the **ExhibitionsWorks** table, delete the redundant attribute 'moveintime', because that information can get from the workslocations table.

~ To test if the changes to worksexhibitions table worked, just query this table by selecting the deleted name of attributes. And display the table to see if the changes works.  
  
6. For the **ExhibitionsLocations** table, add the additional attribute 'el-lcexmuseumname' as a primary key.  
Because other member's museum could have the same location name.   
Also add the attribute ‘el-lctelEndDate’ to store the end date of travelling exhibition.  
Create the domain tempExhibitionLocationStartDate as Date.  
Also add the attribute ‘el-lcexMuseumName’ to store the start date of travelling exhibition.   
Create the domain tempExhibitionLocationEndDate as Date.  
  
Because the exhibitionsLocations is a temporal table we need to keep history records of traveling exhibitions.

~ To test if the changes to exhibitionsLocations table worked, just query this table by selecting the added name of attributes. And display the table to see if the changes works.  
  
7. For the **Doors** table. Add the attribute dr-lcMuseumName as a primary key,   
because that is much easier to check which door connected to which doors in others museum.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Doors** | | dr-door\_from FK-lc  dr-door\_to FK-lc | |  | | *New:*   |  | | --- | | **Doors** | | dr-lcMuseumName FK-lc  dr-lcName1 FK-lc  dr-lcname2 FK-lc | |  | |

~ To test if the changes to doors table worked, just query this table by selecting the added name of attributes. And display the table to see if the changes works.  
  
8. For the **WorksKeeper** table. Add the attributes wkk-workKeeperStartDate and wwk-workKeeperenddate,wwk-status. Also make wkk-workKeeperStartDate and workKeeperenddate as primary keys.

*New:*

|  |
| --- |
| **WorksKeepers** |
| wkk-workKeeperName  wkk-wkIDNumberic FK-wk  wkk-wkIDAlpha FK-wk  wkk-workKeeperStartDate  wkk-status |
| wkk-workKeeperAddress  wkk-workKeeperEndDate  wkk-email  wkkphonenumber |

Create the domain workKeeperStartDate as Date.  
Create the domain workKeeperEndDate as Date.  
Because the same work could be kept by the same person by different time.

~ To test if the changes to worksKeeper table worked, just query this table by selecting the added name of attributes. And query the table to display if the changes works.  
  
9. Also add a **Media** table in my database. Because other museum works have more than one materials for some works.   
Also add the attributes md-wkIDalpha, md-wkIDnumber, mtmaterial into the materials table. And mtidalpha, mtidnumber are the primary keys.  
~ To test if the changes to Media table worked, display the database to see if the made changes works and query the table by selecting the added attributes  
  
10. Create a new table called **TempExhibitionsLocations** to store the temporal travelling exhibition information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *New:*   |  | | --- | | **TempExhibitionsLocations** | | tel-lcname FK-lc  tel-lcMuseumName FK-lc  tel-tempexhibitionLocationStartDate | | tel-sponsor  tel-security  tel-insurance  tel-tempExhibitionLocationEndDate  tel-tempExhibitionLocationAddress | |

Also made the attributes ‘tel-lcname’, ‘tel-lcMuseumName’, ‘tel-tempExhibitionLocationStartdate’ as primary keys  
Also made the attributes ‘tel-lcname’, ‘tel-lcMuseumName’ as foreign keys  
Also add the ‘tel-sponsor’, ‘tel-security’, ‘tel-insurance’, ‘tel-tempExhibitionLocationEnddate’, ‘tel-tempExhibitionLocationAddres  
attributes to store the corresponding information for the travelling exhibitions.

~ To test if the changes to TempExhibitionsLocations table worked, just query this table by selecting the added name of attributes. And display the table to see if the changes works.

11. Create a new table called **WorksState** that store the information about the works state. Add the attributes ‘ws-wkIDNumeric’, ‘ws-wkIDAlpha’, ‘ws-workStateStartDate’ as primary keys. Also add the ‘ws-state’, ‘ws-workStateEndDate’ attributes to store the state information. Also the ws-state could be stolen, damaged, restoration, in good condition.   
Create domain for attribute ‘ws-state’ called state as varchar(30); And create constrain for the state could be only one of (stolen, damages, restoration, in good condition)

~ To test if the changes to worksState table worked, just query this table by selecting the added name of attributes. And display the table to see if the changes works  
  
12. Create a new table called **WorksInsuranceValue** that store information about the works insurance information. Make the ‘wiv-wkIDNumeric’, ‘wiv-wkIDAlpha’, ‘wiv-workinsuranceValueStartDate’ as primary keys. Also add attribute ‘wiv-insuranceValue’, ‘wiv-workInsuranceVaueEndDate’ to record the information about the insurance of each works. Because the insurance value of each works could change.  
Also create domain for attribute’ wiv-insuranceValue’ called workInsuranceValueStartDate as date;  
Also create domain for attribute‘wiv-insuranceValue’ called workInsuranceValueEndDate as date;

~ To test if the changes to worksInsuranceValue table worked, query this table by selecting the added name of attributes. And display the table to see if the changes works  
  
By using Dbvisualiser to display the E-R diagram of my new database and compare it with our planed E-R diagram. If they have same structure, then my database is correct.

**5. Nam Pham’s report**

I will have to change some part of my database before I can transfer the data to the new one.

Main table:

- Works table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **All\_works** | | wk-itemalpha  wk-itemnumber | | wk-name  wk-description  wk-type  wk-subtype  wk-author  wk-date\_complete  wk-date\_accquire  wk-insurance\_value  wk-claim  wk-potray\_with | | *New group table:*   |  | | --- | | **Works** | | wk-IDAlpha  wk-IDNumeric | | wk-worksName  wk-type  wk-subtype  wk-Creator  wk-completionDate  wk-acquisitionDate  wk-workDexcription  wk-ownership  wk-question  wk-answer | |

For this table, my current table already look identical to it, including the primary key, although there are still some differences.

The table that the group agree on will not contain the **insurance\_value** value of the item because that value will be record as the temporal data for the new table *WorksInsuranceValue*. So, I will need to add all the current item value to that table and set the same date range for all of them. Because there aren’t any change in the insurance value, so all the end\_date value will be null.

*Test*: I will create a query that choose the idnumber and idalpha from the new database and find it whether the data of the new database catch up. For the insurance value which are now in the new table, I will also make it one part of the query even when it can cause multiple row.

- Locations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Galleries** | | lc-name | | lc-dimension  lc-min\_capacity  lc-max\_capacity | | *New:*   |  | | --- | | **Locations** | | lc-name  lc-museumName | | lc-telStartDate  lc-telEndDate  lc-dimension  lc-minNumWorks  lc-maxNumWorks  ls-availableDate | |

For the location table, because we will combine the data of all the group members together, we need to distinguish each member location by adding the attribute **museumName** and make it as a part of the PK. So, even when our locations the same names, it does not violet the PK unique rule. Therefore, I just need to add the name of my museum for each row of location information.

Now, even though my old table already have the attributes like: dimension, min and max capacity, we decided to add more attributes for the locations of the traveling exhibitions later. So, the **telStartdate** and **telEndDate** will be used to tell, for a temporary location, when the exhibition will be held and end at that place. So, because I have stored the date information in the *exhibition\_info* table of my old database, I will use it for this 2 attribute. However, for the locations that are in the main museum, they will never held a traveling exhibition and therefore, this 2 attributes will be null for them.

The **availableDate** is used for when the location is not occupied and can be used. It is most likely the end date of the exhibitions. So, for now I will only need to update this attribute with the date after the exhibitions end, and then later I can create a trigger or a rule that can used this attribute to plan the exhibition in the future.

Test: I will create a query that include the location name to check whether or not the data catch up. The tables that will be used from the old database to check is the galleries table and the temporary location table.

- Exhibitions table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitions\_info** | | ex-exhibition\_id | | ex-theme  ex-ex\_descripton  ex-number\_of\_items  ex-start\_date  ex-end\_date  ex-location\_name | | *New:*   |  | | --- | | **exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate  ex-museumName | | ex-istraveling  ex-exhibitionDescription  ex-exhibitonEndDate | |

In my old table, I used the **exhibition\_id** as the set of 3 characters to counter the problem that is 2 exhibitions can have the same name. However, our group decide to go with using the **exhibitionName** and the **exhibitionStartDate** as a part of the PK to counter the problem. We also add the **museumName** attribute to distinguish our exhibitions from each other. So, for me, I have to create a new set of PK and like in the Location tables, I will add my museum name for each row of exhibitions information. I will also have to drop the **exhibition\_id** as I won’t use it in the future.

I also have to drop the number\_of\_items and location\_name as they will be store elsewhere in the database. However, I will have another attribute **istraveling** which is a Boolean. My data will have difference from my group as for a travelling exhibition, they will use it as a whole, but I divided it into 5 exhibitions for each temporary location with the different start date. So, I can add **istraveling** to tell whether or not the 5 exhibitions is actually one or not, but the way I represent the traveling exhibition will be different from my groupmates.

Test: I will create a query that use the museum name to check whether or not the data in the new database catch up or not. The table from the old database that is used to compare is exhibitions\_info table.

Temporal data Tables: The tests in these tables will be most likely use a query to find out that the data has matched up or not because we only need to change the PK set of them.

- TempExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Temporary\_location** | | tel-name\_location FK-lc | | tel-sponsor  tel-address  tel-head\_of\_security  tel-insurance | | *New:*   |  | | --- | | **TempExhibitionsLocations** | | tel-lcname FK-lc  tel-lcMuseumName FK-lc  tel-tempexhibitionLocationStartDate | | tel-sponsor  tel-security  tel-insurance  tel-tempExhibitionLocationEndDate  tel-tempExhibitionLocationAddress | |

In my old table database, I forgot to add the situation that the temporary location can be reused in the future and this table can be used as a temporal data storage. Now, our group decide to add the **tempexhibitionLocationStartDate** attribute to know the time the temporary location is used and also make it a part of the PK. Moreover, because this table is connected to the *TempExhibitionsLocations* table it also need to add the **lcMuseumName** attribute to be a part of the PK and the FK to *locations* table as the PK in the *locations* table is changed. So, what I need to do here is to change the part of add the museum attribute and change the set of PK and the FK in the *Temporary\_location* table.

The **startDate** and **endDate** attributes in this table will have the also have the same value as the one in the *locations* table, so I can easily update them.

- WorksLocation table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **work\_location** | | wl-itemalpha\_ex FK-wk wl-itemnumber\_ex FK-wk wl-current\_location  wl-date\_in | | wl-date\_out  wl-time\_in  wl-time\_out | | *New:*   |  | | --- | | **WorksLocations** | | wl-lcName FK-lc  wl-wkIDAlpha FK-wk  wl-wkIDNumeric FK-wk  wl-lcmuseumName FK-lc  wk-workLocationStartDateTime | | wk- workLocationEndDateTime | |

For this table, because of the change of PK set in *locations* table, I will need to add the **lcmuseumName** attribute to connect it with the location table.

The other thing I need to do is to combine the 2 attributes **date\_in** and **time\_in**, so that the attribute can show both date and time. After that I can make the result attribute as a part of the PK in this table. The same will be done for **date\_out** and **time\_out**, but we don’t need it as part of the PK. No further change need to be made for this table.

- ExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Exhibition\_location** | | el-Ex\_id FK-ex  el-current\_location FK-lc | | el-no\_of\_items | | *New:*   |  | | --- | | **ExhibitionsLocations** | | el-lcName FK-lc  el-exName FK-ex  el-exStartDate FK-ex  el-lcexMuseumName FK-lc, ex | | el-lctelStartDate  el-lctelEndDate  el-exEndDate | |

Because the set of PK in both *Exhibitions* table and *Locations* table are changed, I will need to add the more attributes to make a new PK. All of the data for these new attributes are already available in *locations* and *exhibitions* and therefore, can be updated from those tables. Because **Ex\_id** is not needed in the new database, I will also drop it.

The no\_of\_items attribute is also not needed for this table, so I will also drop it.

The 2 new special attributes like: **lctelStartDate** and **lctelEndDate** are used to store the date of the traveling exhibition. For this dates, I can find them both in the *exhibitions* table or the *temporary location* table, therefore I can update it from there. However, for the temporary locations, the value of this attribute will be null.

- ExhibitionsWorks table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Exhibition\_works** | | ew-itemalpha\_ex FK-wk  ew-itemnumber\_ex FK-wk  ew-ex\_id FK-ex | |  | | *New:*   |  | | --- | | **ExhibitionsWorks** | | ew-exName FK-ex  ew-wkIDAlpha FK-wk  ew-wkIDNumeric FK-wk  ew-exStartDate FK-ex  ew-exMuseumName FK-ex | | ew-endDate | |

This table has to be changed due to the change of the PK set from the exhibitions table. So, after I added all the new attributes, I can update the information from the exhibitions table.

- WorksKeepers table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **donors** | | do-itemalpha\_don FK-wk  do-itemnumber\_don FK-wk | | do-donor\_name |        |  | | --- | | **loan\_institution** | | lo-itemalpha\_loan FK-wk  lo-itemnumber\_loan FK-wk | | lo-name  lo-address  lo-phonenum  lo-email  lo-start\_date  lo-end\_date | | *New:*   |  | | --- | | **WorksKeepers** | | wkk-workKeeperName  wkk-wkIDNumberic FK-wk  wkk-wkIDAlpha FK-wk  wkk-workKeeperStartDate  wkk-status | | wkk-workKeeperAddress  wkk-workKeeperEndDate  wkk-email  wkkphonenumber | |

The *workskeeper* table is the combination of the donors and *loan\_institution* table. Because this 2 tables have the same attributes and can work in the same way, all we need to do is to add another attribute **status** to know whether or not the items are lend or borrowed. All the data can be taken from this 2 old tables after the merge.

When I do these 2 tables, I didn’t think to make them the temporal data storage, so I only make the item identification to be the PK. However, in this project, making it a temporal data storage mean I have to change the PK set like the above, but all the information can be update easily.

- Doors table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Doors** | | dr-door\_from FK-lc  dr-door\_to FK-lc | | dr-door\_name | | *New:*   |  | | --- | | **Doors** | | dr-lcMuseumName FK-lc  dr-lcName1 FK-lc  dr-lcname2 FK-lc | |  | |

The doors table doesn’t have many major change.

I will need to drop the **doorname** as it is considered unnecessary.

Then, because the PK set in the locations table is changed, I will need to add the **museum name** as part of the table PK. This will also be a way to distinguished the doors between the museum. All the data can be taken from *locations* table.

- Media table

|  |
| --- |
| **Media** |
| md-wklIdAlpha FK-wk  md-wklIdNumber FK-wk  md-material |
|  |

For this table, it is identical to my current one and have the same primary key as foreign key. Therefore, I can transfer my data to this table easily.

New tables:

- WorksState table

|  |
| --- |
| **WorksState** |
| ws-wkIDNumeric FK-wk  ws-wkIDAlpha FK-wk  ws-workStateStartDate |
| ws-state  ws-workStateEndDate |

This table is used to check the current state of the item which including: in good condition, stolen, damaged, etc and the time range they are in that condition. So, for all of my current item now, I will just make them into ‘in good condition’ state.

- WorksInsuranceValue table

|  |
| --- |
| **WorksInsuranceValue** |
| wiv-wkIDNumeric FK-wk  wiv-wkIDAlpha FK-wk  wiv-workInsuranceValueStartDate |
| wiv-insuranceValue  wiv- workInsuranceValueEndDate |

This table will keep the record of the table insurance value that will be changed by the time. However, because all my insurance value have not been changed yet, I will only keep the current insurance and only insert new one when it is changed.