Nam report:

To change my current database to the team database, I have to rely on the information I have on my old database. So, after my team agree on the database design, we have created the text files that can create the tables and domains of the team database. Then, I will use that files to create the team database and use the SQL code to transfer all my data from the old design to the new one. Then after I done everything, I will copy all the data to text files, fix some errors and update them on the team database. This is the more detail of how I have done the work in each table:

1. **Institutions 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:*   |  | | --- | | **Institutions** | | in\_Name | | In\_email  In\_ phoneNumber  in\_streetAddress  In\_city  in\_countrySubdivision  In\_country  in\_postalCode | |

As you can see, the institutions table is the combination of 2 tables donors and loan\_institution from my old data base.

For the loan\_institution table, it is easy to get the information from it when it already has the name of the institution, the phone number and the email. The only think I had to change in this table is that I will have to divide the address to smaller part like: street address, city, country subdivision, country, and postal code, but I can update it manually very fast because there are only 5 loan institutions. It also doesn’t have any duplicate names so, but I still use distinct just in case.

For the donors table, as you can see, it lacks everything when it only has the institutions name. Not only that, some name is duplicate because one institution can lend you 2 items. So, right after I get the name for all the institutions, I had to update the other information for them. And because there are 32 donors, it was a long process. Though, most of my donors is not real, so the information is fake anyway.

Because the institution table has the institution name as primary key, I have me to realize there is duplicate value inside my donors table when I insert from it. After everything was done, I just need to query it to show all the table and get the number of rows.

1. **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:*   |  | | --- | | ns-**Works** | | ns-wk-IDAlpha  ns-wk-IDNumeric  ns-wk-IDinstitution | | ns-wk-worksName  ns-wk-Creator  ns-wk-completionDate  ns-wk-acquisitionDate  ns-wk-workDescription  ns-Wk-ownershipStatus  ns-Wk-theme  ns-Wk-subject  ns-Wk-culture  ns-Wk-colour  ns-wk-carType  ns-wk-transmission | |

For this new table, the first problem occur is that the new attribute that will represent type and subtype. This is the result from combining the type and subtype of all the team members and now, the table will have new attributes which represent the team members type and subtype with a new meaningful name. Though, this is not a big proplem because I can make them null when I use the insert transaction.

Beside from the problem above, all my data from the old table is sufficient enough for the new one and it didn’t take me much time to transfer all the data from the old table to the new one.

The other change is that now, because I have combine thee museum with other teammate, my items now had to associate with my museum name from now on. However, this is also not a big issue when it can be easily done in the insert form.

So, even when this new table have connection with the institutions table, because I have insert all the data in one swoop, it doesn’t cause any problem.

Finally, I do the query and have all the data in organize and don’t cause any error.

1. **Locations TABLE:**

*New:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Galleries** | | lc-name | | lc-dimension  lc-min\_capacity  lc-max\_capacity |  |  | | --- | | **Temporary\_location** | | tel-name\_location          FK-lc | | tel-sponsor  tel-address  tel-head\_of\_security  tel-insurance | | **Locations** |
| ns-lc-name  ns-inInstitutionName |
| ns-lc-length  ns-lc-height  ns-lc-width  ns-lc-minNumWorks  ns-lc-maxNumWorks  ns-ls-availabilityDate  ns-lc-sponsor  ns-lc-security  ns-lc-insurance  ns-lc-streetAddress  ns-lc-city  ns-lc-country  ns-lc-postalCode |

The locations table of the team database now is a combination of mu 2 previous tables: galleries and temporary\_location and they cause me to have some change.

In my old database, the temporary location is the sub-table of the galleries table and therefore, the galleries table have the names of all the temporary locations. So, when I insert the data into the new table, it will also contain the name of the temporary location, but it won’t take other information because I need to change them before I insert the new one.

For the galleries table, I can only take the name of the location, and the min and max capacity because the dimension will be divided by 3 values in the new table. Not only that, I also add my museum name when I was performing the insert. After I have done the insert for this normal location, I had to update the dimension into 3 new values and also the availability date, which is the next day of its last exhibition (I can get this information from the old exhibitions locations table). So, for this table, I had to double check the value before I can update it to the new data.

For the temporary location, I have already got all theirs name in the new table, the next thing I need to do is to update the value manually because now the address one again had to be separated into new attributes. The update did not take that much time when I can get them from the old tables, even when I had to do it manually.

Finally, I do the query to show the whole table and have all the data in organize and don’t cause any error.

1. **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** | | ns-ex-exhibitionName  ns-ex-exhibitionStartDate  ns-ex-inInstitutionName | | ns-ex-istraveling  ns-ex-exhibitionDescription  ns-ex-exhibitonEndDate | |

In my old table, it got most of the necessary information for new one like the name of the exhibition, thee description, the start and end date. So, I only need to deal with 2 new attribute which are: isTraveling and the museum name. The museum once again, is not a major problem in my data when I can deal with it in the insertion like the other previous tables. The one I had to deal with was the isTraveling attribute.

Because isTraveling will show whether or not the exhibitions is going to be traveling or not and it is a Boolean, I have to do the insertion 2 times for 2 kinds of exhibition. It is not a major problem because this 2 insertion have the same format, but the only difference is the value of isTraveling is true or false.

This table have not much to check when it doesn’t have much data. I only need to have a query that show the whole table to check whether it is organized or not.

1. **Works\_Locations 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  wklcinInstitutionName             FK-lc,wk  wk-workLocationStartDateTime | | wk- workLocationEndDateTime | |

Because my old table have the separate attribute for the time and the date, I had to combine them together, so that I can insert them into the new database. To do this, I had to change my old table by adding a new attribute that now can store the timestamp value. Then I had to do the most time-consuming part of my database, that is I had to combine and update them manually. Though it is lucky that there are many rows have the same time and date value because there are many items have the same time and date to go into the exhibition or go to the new locations, but it was not enough when I had to update for 2 attributes (timestamp when it is in and timestamp when it is out). Every time I done an update, I had to go to another look for the new date that haven’t got combine yet and sometimes, I miscounted the row and had to do it over again. However, thanks to this, I also found out some error in my old data and then I can delete them from the database.

After I have done all the combining, I finally can the insert the data from that old table of mine to the new one. There is not so much difference between my old and the new table now when the only thing I had to do is adding my museum name for each of the row.

Because this is the most massive table that I have, I have some queries to check a certain of works to see whether theirs date is reasonable and possible. I got some error occur and I had to fix some part of the data.

1. **Works\_Exhibitions 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 | |

My old table does not contain the information of date period of the exhibitions as well as the name of the exhibition, therefore, I had to change my old table (just like in works\_location) to have more attribute that contain those data. So, I add the exhibition name column and the start date and end date column, then I get the update from the exhibitions table. Because they have the same exhibition\_id, it is easy to update that and this process won’t take long like the previous one.

After I got all the information I need in the old tables, I will simply do the insert from my tables to the new one, and of course, I will also insert my museum name.

I only need the query showing the whole table and compare it with the old one to find some problem it may have.

1. **Works\_Value TABLE:**

This table is the new table and only contain the value of my item. And because my works value had not been changed, it only need one record for each item. So, I just need to get the value, the work identification and their acquisition date to insert into the new table.

However, I still need to do some end date update in the end because some of my items have been sold to another institution, therefore, the end data of this item value is the date it be sold because from now on, I won’t have any connection with it (except if they sold it back to me).

1. **Works\_State TABLE:**

This table will show the current state of the items in a period of time. But I haven’t change anything to my works state so like the works value table, I will just update all the state to ‘in good condition’ while inserting the identification and theirs acquisition date.

However, I still need to do some end date update in the end because some of my items have been sold to another institution, therefore, the end data of this item value is the date it be sold because from now on, I won’t have any connection with it (except if they sold it back to me).

1. **Works\_Media TABLE:**

My media table look exactly like the team database table. The only difference is that now I had to add my institution name and nothing else. So, it is an easy to check situation when I only need a query to show the whole table.

1. **Exhibitions\_Locations 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 | |

Like the works exhibitions table, my old exhibition location table does not contain the information about the exhibition name and its period of time. The period of time for the traveling exhibition at each location also does not concern me much because in my old exhibitions table, I have keep record of all of them.

So, the first thing I do is to add more columns into my old table and then update them to get the new information from the exhibitions table. This is the exact thing that I do at the works\_exhibitions table, which have the same format as the old exhibition location table.

After I got all information in the exhibition location table, I just need to use insert to get the data to the new table and I will also add my museum name in this insert.

In the end the separate date for the traveling exhibition, which can be update manually.

I only need the query showing the whole table and compare it with the old one to find some problem it may have.

1. **Locations\_doors TABLE:**

My doors table look exactly like the team database table. The only difference is that now I had to add my institution name and nothing else. So, it is an easy to check situation when I only need a query to show the whole table.

1. **Works\_Ownership TABLE:**

In my old works table, it has the claim attribute which can show that whether the items is OWNED, BORROWED, POTENTIALLY BORROWED or LOAN. Therefore, based on this information, I can update my ownership table.

First the owned and loan means that my works is already mine, therefore, I will be its owner. So, by using the acquisition date as start date of the ownership, I have upload the item as mine.

Then there is the owner of the BORRWED work. The name of this institutions is in my donors table along with their identification of the item they lend us, therefore I can easily track their name down and insert into the table. However, because I do not know the date their ownership start, so I just choose randomly the date 2016-6-1.

I only need the query showing the whole table and compare it with the old one to find some problem it may have.

1. **Transactions TABLE:**

This is the new table that will record my museum transaction with other institutions including my team database. However, I have lost track of all my database at this point, so I will only update the transaction when I am selling the item to other institutions.

The trigger used:

The trigger will be divided for its function to the whole table:

* The trigger that is only used for update the new value of one table:
* CheckValue on works\_Value table:

When the value of the item change, it means that the end date of the previous value will also going to end. This trigger will help you to update the final date of the previous value by giving it the start date of the new value as long as the end date of the previous value is NULL.

* ChangState on works\_ State table:

When the state of the item change, it means that the end date of the previous state will also going to end. This trigger will help you to update the final date of the previous state by giving it the start date of the new state, as long as the end date of the previous state is NULL.

* ChangeWorkLocationEndDate on works\_location table:

When the location of the item change, it means that the end date of the previous location will also going to end. This trigger will help you to update the final date of the previous state by giving it the start date of the new location, as long as the end date of the previous location is NULL.

* The triggers that will be used to plan for a whole new exhibition, in other work, they are related to each other:
* changeWorkLocation on works\_exhibitions table:

This trigger is not necessary needed, however, it is still efficient and can reduce the works of the trigger after it. In other word, let say you are planning for a new exhibition and you know all the works that you can use in that exhibitions. The first thing you do is to insert the information of this new exhibition to the exhibitions table. However, at this step, you must not declare the isTraveling attribute to be true or false because it is an essential part for the trigger after. Next, you will add the identification of your items with the name of the exhibition to the works\_exhibitions table, then this is the time that the trigger does its job. It adds the work identification along with other necessary information to the works\_locations table. However, the location for this row will only be a temporary one and it will be the storage of each individual museum. Right after we done with the next triggers, this temporary location will be updated and change into the location that the exhibition is held.

* Ex\_schedule on exhibitions table:

Now, the next step to set up an exhibition is to update the value isTraveling of the exhibition. After we done the update to tell whether the new exhibition is a traveling exhibition or not, this trigger will be activated. Sadly, this trigger will only perform the setting location for the normal exhibition, because the temporary locations for traveling exhibition do not have min and max capacity or available date. Moreover, the location for the traveling exhibition can be multiple, therefore, it should be insert manually to the exhibitions\_locations.

Back to the trigger, it will do the job to find the most suitable location for your exhibition based on the available date and the location capacity. After it finds the best location to set the exhibition, it will add all the information to the exhibitions locations table and activate the next trigger.

* updateWorkLocation on exhibitions locations table:

After we insert a new row to the exhibition location table by the Ex\_schedule trigger (for the normal exhibition) or by hand (for the traveling exhibition), this will activate another trigger that will change the data to the works\_location table.

This trigger can recognize the whether the exhibition got insert in the exhibitions\_locations table is a traveling or not by checking the traveling date. If is null, it is a normal exhibition, but if it is not, it is a traveling one. Base on this information, the trigger will have different action to store the information because these 2 kinds exhibitions are different. Fow example, the date the item go to the location will be 2 days before the exhibitions held, but for traveling exhibition, the date it come to the temporary location is the traveling date, not the exhibition start date. So, for this trigger, it helps us to sort out the kinds of exhibitions.

* The trigger that bring the new works to the storage:
* changeWorkLocation on works table:

When the new item is added to the museum, it will always first go to the storage, however, this is not the case with the POTENTIALLY BORROWED item when they are not technically in our museum yet. However, this trigger can recognize whether this item is just BORROWED or not by looking at its acquisition date. If this date is NULL, then it means it is just POTENTIALLY BORROWED and therefore, no need to add into the works\_location table. The opposite will be done for the other. So, the item will only go into the museum storage if only if we update its acquisition date.