COLORADO SCHOOL OF MINES

Famous Artwork

Adam Stogsdill, John Capper, Daniel Brouillet, Shaine Carroll-Frey

CSCI 403 Final Project

May 1, 2019

1 INTRODUCTION

Art has existed probably for as long as humans have existed, and as such is integral to humanity's collective cultural development. As such, we decided to create a Java application to display data about various famous artworks. Our goal was to provide an easy to use application for people to learn about and appreciate famous art.

Our database contains information about famous artworks and artists, each contained in their own separate tables. There is also a table containing information about where an artwork is located (museum, private collection, etc). The information was chosen based on what would be useful to include in the Java application like URLs for the images of artworks.

The application that we created for the database allows the user to filter through the data and allows the user to display a piece of art and display some other information of that artwork using pre-written queries from the database.

2 DESCRIPTION OF APPLICATION

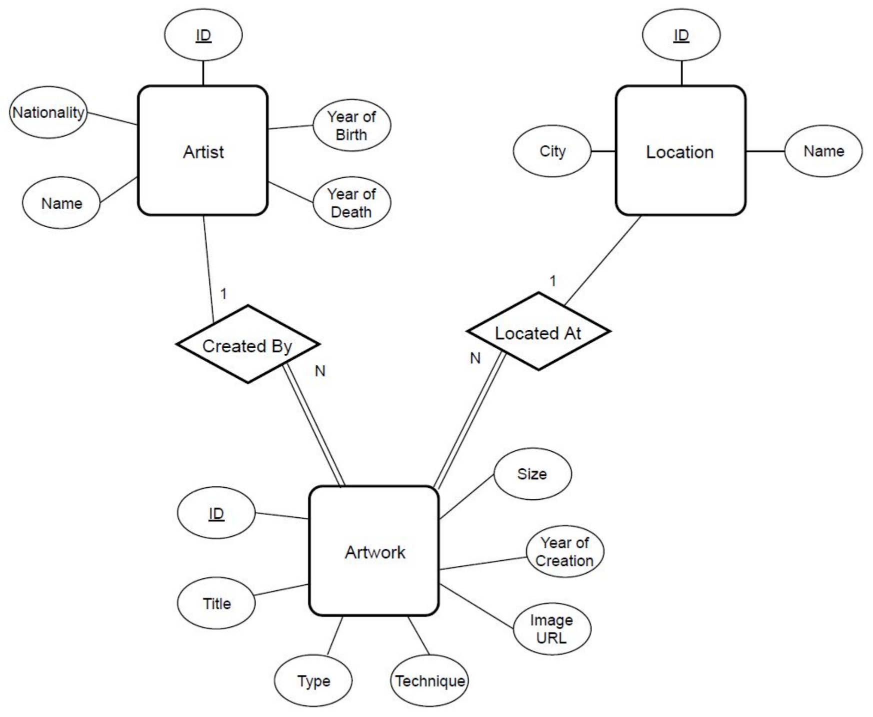


Figure 1. Entity Relationship Diagram of Database

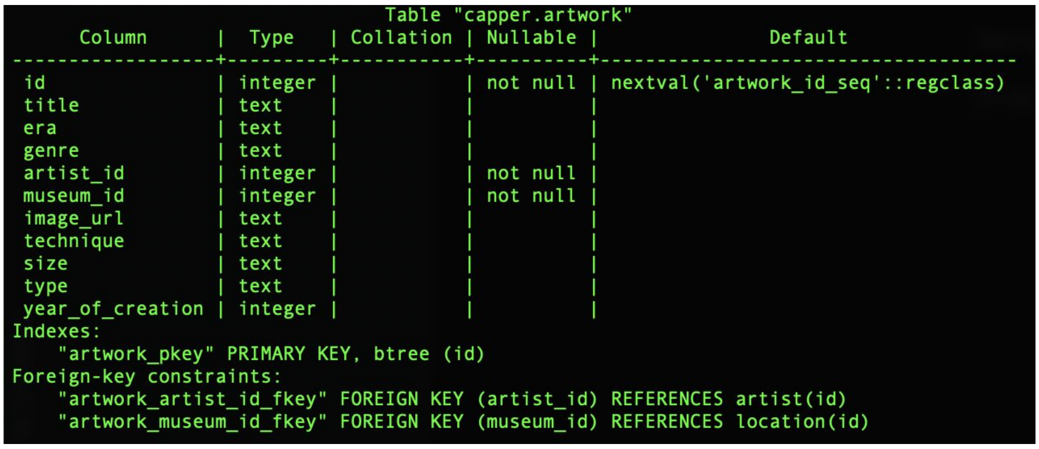
Figure 1 shows the entity relationship diagram (ERD) of the database. The ERD consists of three main entities: Artwork, Artist, and Location. The whole database surrounds the entity Artwork. Artwork contains the attributes. *id*, *title*, *type*, *technique*, *year of creation*, and *size*. The attribute *id* is a key so that an instance can be uniquely identified if there are multiple occurrences of the same title. The attribute *type* represents the main category of artwork a piece falls in (painting, sculpture, ceramics, etc). The attribute *genre* represents the genre of a piece which can be, for example, religious, historical, landscape, etc. The attribute *Image URL* represents the link to an image of the artwork.

The entity Artwork has a 1-to-N "Created By" relationship with Artist. The Artist entity partially participates in this relationship since an artist does not need to have an artwork(at a certain point in time) to exist. However, the Artwork entity fully participates in the relationship because it can't exist without being created by an artist. The Artist entity contains the attributes *id* (key), *name*, *nationality*, *year of birth*, and *year of death*.

The entity Artwork also has a "Located at" relationship with Location. This relationship is

1-to-N since one location can store multiple artworks an artwork can only be stored in one location. This relationship has partial participation on the Location side because a location can exist without containing any artwork. However, on the Artwork side, there is full participation since it can't exist without a location. The entity Location has the attributes *id* (key), *name*, and *city*.

Dataset



```
                              Table "capper.artwork"
     Column      |  Type   | Collation | Nullable |              Default
-----------------+---------+-----------+----------+------------------------------------
 id              | integer |           | not null | nextval('artwork_id_seq'::regclass)
 title           | text    |           |          |
 era             | text    |           |          |
 genre           | text    |           |          |
 artist_id       | integer |           | not null |
 museum_id       | integer |           | not null |
 image_url       | text    |           |          |
 technique       | text    |           |          |
 size            | text    |           |          |
 type            | text    |           |          |
 year_of_creation | integer |          |          |
Indexes:
    "artwork_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "artwork_artist_id_fkey" FOREIGN KEY (artist_id) REFERENCES artist(id)
    "artwork_museum_id_fkey" FOREIGN KEY (museum_id) REFERENCES location(id)
```

Figure 2. Artwork Table Schema

Our database consists of three tables: artwork, artist, and location. These tables were derived from the ERD shown in Figure 1.

Figure 2 shows the schema of the artwork table. The column *id* is primary key with the sequence *artwork_id_seq* that auto-increments the value on a new insertion so that each row has a unique value. The table has the columns *artist_id* and *museum_id* that contain foreign keys referencing the *id* columns of the artist and location tables respectively. Since the artist entity fully participates in both relationships, the columns *artist_id* and *museum_id* have "not null" constraints. Artist and Location tables were created as well, containing columns of their corresponding attributes in the ERD. Both tables have auto-incrementing *id* columns to ensure each row can be uniquely defined.

Collecting data was the main issue with the dataset. Many sources had lots of information relating to our topic but restricted the amount of data that can be obtained unless a subscription was purchased. In addition, using APIs was too difficult since individual requests needed to made to get information about each piece, which took a very long time.

Eventually, we found data from the Web Gallery of Art (WGA), a searchable database of fine arts from the 3rd to 19th centuries. WGA allowed the data to be downloaded free of charge and without license restrictions. The data was downloaded as a CSV file, that contained around 50 thousand unnormalized rows of artwork info. Since the entire database was able to be downloaded straight up, the info was grabbed quickly as opposed to needing to make many requests to an API. Once the data was downloaded, Python was used to parse and group the data then insert into the database.

Graphical User Interface

The program utilizes a query management system that holds the queries in memory to speed up the queries and hopefully relies less on the database once the program has already gathered that information. The program consists of 4 ”autocomplete” text boxes that allow the user to type into a corresponding category and sift through the artist, locations, year of creation, and the artwork. Using the artist, locations, and year of creation text boxes you can then filter the artwork list. (Only one can apply at a time because if we use multiple filters at once then the likelihood of actually getting at least one artwork is very low thus the functionality seemed not needed). Though if there are multiple instances of artwork and no filter was applied then a popup will show allowing the person to specify the artist in particular. There is also a help button that describes how to use the application.

The most difficult challenge was parsing the query into the actual GUI because it required me to parse the information using indexes because of my QueryConversionData class so I had to add some more parsing algorithms to avoid spending too much time parsing and just give the answers to the JPanels that I needed. There were also some O(n^2) algorithms that were being implemented because of the 2D ArrayList that I needed to parse the information. So I made some modifications to the code to do only O(n) algorithms on the datasets that were large (especially when parts of the dataset have > 40,000 points of information).

3 CONCLUSION

It goes without saying that art is an incredibly important component in the advancement of human society. However, many people may not have the means to view and learn about famous artwork, which was the motivation behind this project. By obtaining our data exclusively from the Web Gallery of Art, we were able to avoid paying any fee or dealing with any licensing restrictions on our data. Then, by parsing this data into a simple Java GUI, we were able to create an attractive, easy-to-use interface in which users can filter out what art they want to see and data about that specific artwork. If we had more time and/or resources, we may consider obtaining data about modern art from some other source in order to make our application more complete, as currently it only contains data pertaining to artwork created between the 3rd and 19th century. We also may consider adding a search function in which the user can simply type in the name of a piece or artist and view the piece and/or a list of pieces created by that artist. However, with approximately 50,000 pieces of art, we can confidently say that this application is a quality resource for anyone wanting to view and learn about famous artwork.