Joseph Valencia

Thomas Woods

CS 340 Final Project, Art Galleries

## Outline:

Our database contains various art galleries/museums around the world. These galleries contain various sections that display artwork painted by various artists, and customers can enter the sections and purchase art from sections that they have visited, which get stored as sales. Treating this as a database, we can track which sections/galleries each customer has gone to, along with the various art pieces they have purchased. This basic setup could easily be extended for any application involving inventory/sales, so it is an intriguing test system that can allow us to see some of the complexities of these systems.

## Database Outline:

The highest point in the hierarchy of the database is the gallery entity. Galleries contain zero or more section entities. A section must belong to one and only one gallery. Artwork represents individual paintings. An artwork has one artist, and artists can paint many artworks. An artwork can occupy one section, and a section can hold many artworks. An artwork must have an artist and must be in a section. Artwork may be sold to a customer. A customer may purchase many artworks, but each artwork can only be purchased by one customer. A customer can only purchase art from a section he or she has visited. Many customers can visit many sections.

### Webpage

The top of the webpage contains lists tables that contain information about the various galleries, sections, artwork, customers, artists, and sales contained in the database, as well as how many unique customers have been to each of the galleries in the database.

The forms in the next part of the page allow you add/remove entries of the various entities in the database, and all the forms on this page will post the field entries to a child page to perform the actual work. Galleries, artists, and customers may be added without any dependencies, while sections are dependent on the gallery that they are in, and artwork is dependent on the section that it is in as well as the artist that painted the piece. You can specify two fields when adding a gallery to the database: the name of the gallery and the city that it is located in. The fields for adding an artist are the first name and the last name along with the art Movement that that artist is most known for. Customer fields include a first name and a last name. Adding a section has a field for specifying the section’s name, and you must select the gallery that the section is located in. The fields for adding artwork include the title of the artwork, the year that it was created, the price of the piece, and the user must specify the sectioni/gallery that it is located in along with who the piece was painted by. Deleting any of these entries from the database from their respective delete sections also cascades to delete any of their dependencies.

The next section allows a user to visit various sections/galleries in the database. After the customer has visited one of these sections, he/she is able to purchase art from that section in the “Purchase Artwork” form below “Visit a Section”. The user enters a description for the type of sale that it is, and if the sale goes through it will be added to the list of sales and the dropdown for the artwork will display that the art has been “Sold”. The last section of the database includes filters that will display the list of purchases according to the customer that the user selects, the list of sections that a customer has visited, and there is a final section that filters customers out by their last name.

## ER Diagram



# Database Schema

## Table Creation Queries

The following queries were used to create tables for the Art Galleries Database (reproduced from the file Project\_DDL.sql:

-- ----------------------------------------------------------

-- CS340 Intrdoduction to Datbases

-- Fall 2016

-- Class Project

-- Joe Valencia

-- Tom Woods

-- ----------------------------------------------------------

-- Data Definition

-- ----------------------------------------------------------

-- Reference for the Dev Database on AWS. Comment out for deployment elswhere

-- use CS340;

-- Turn off foreign key checks to allow dropping tables that hold data

set FOREIGN\_KEY\_CHECKS = 0;

-- gallery

drop table if exists gallery;

create table gallery

(

galleryID int not null auto\_increment primary key,

galleryName varchar(200),

galleryCity varchar(200)

) engine = innodb;

-- section

-- One or more sections belong to a Gallery Floor

drop table if exists section;

create table section

(

sectionID int not null auto\_increment primary key,

sectionName varchar(50),

galleryID int not null,

Foreign key fk\_section\_gallery (galleryID) references gallery(galleryID)

on delete cascade

on update cascade

) engine = innodb;

-- art

-- each art is made by one artist

drop table if exists artwork;

create table artwork

(

artworkID int not null auto\_increment primary key,

artworkTitle varchar(50),

artworkYearCreated int,

artworkSectionID int,

artworkArtistID int,

artworkPrice int,

isSold int,

Foreign key fk\_artwork\_section (artworkSectionID) references section(sectionID)

on delete cascade

on update cascade,

Foreign key fk\_artwork\_artist (artworkArtistID) references artist(artistID)

on delete cascade

on update cascade

) engine = innodb;

-- artist

drop table if exists artist;

create table artist

(

artistID int not null auto\_increment primary key,

artistFirstName varchar(50),

artistLastName varchar(50),

artistMovement varchar(50)

) engine = innodb;

-- customer

drop table if exists customer;

create table customer

(

customerID int not null auto\_increment primary key,

customerFirstName varchar(50),

customerLastName varchar(50)

) engine = innodb;

-- sales

-- Sales are made in individual transactions to customers by a gallery

-- for a specific peice of Art

-- tmw: removed section, transactionType, amount per simplification discussion

drop table if exists sales;

create table sales

(

transactionID int not null auto\_increment primary key,

saleDescription varchar(250),

artworkID int,

customerID int,

Foreign key fk\_sale\_artwork (artworkID) references artwork(artworkID)

on delete cascade

on update cascade,

Foreign key fk\_sale\_customer (customerID) references customer(customerID)

on delete cascade

on update cascade

) engine = innodb;

-- visitorLog

-- table to track which customers have visited which museum

drop table if exists visitorLog;

create table visitorLog

(

logID int not null auto\_increment primary key,

customerID int,

sectionID int,

foreign key fk\_visitor\_customer (customerID) references customer(customerID)

on delete cascade

on update cascade,

foreign key fk\_visitor\_section (sectionID) references section(sectionID)

on delete cascade

on update cascade

) engine = innodb;

-- restore the foreign key checks

set FOREIGN\_KEY\_CHECKS = 1;

-- Triggers

-- Commented out because we don't have permissions to make them on anything we can reach

-- from the class assignment.

-- NOTE: The following triggers were commented out when we realized --- we didn’t have permission on our ONID-DB to create them:

/\*

-- clear the sold flag for every new artwork

CREATE TRIGGER set\_sold\_flag

AFTER INSERT ON artwork FOR EACH ROW

update artwork

set isSold = 0

where artworkID = NEW.artworkID;

-- set sold flag when a sale is made

CREATE TRIGGER set\_sold\_flag

AFTER INSERT ON sales FOR EACH ROW

update artwork

set isSold = 1

where artworkID = NEW.artworkID;

-- clear the sold flag if a sale is deleted

CREATE TRIGGER set\_sold\_flag

AFTER DELETE ON sales FOR EACH ROW

update artwork

set isSold = 0

where artworkID = NEW.artworkID;

\*/

## General Use Queries

The following queries operate our website:

### Selection Queries

* SELECT galleryName, galleryCity FROM gallery
* SELECT gallery.galleryName, section.sectionName FROM section INNER JOIN gallery ON section.galleryID=gallery.galleryID
* SELECT gallery.galleryName, artwork.artworkTitle, artist.artistFirstName, artist.artistLastName, artwork.artworkYearCreated, section.sectionName, artwork.artworkPrice FROM artwork INNER JOIN section ON artwork.artworkSectionID=section.sectionID INNER JOIN gallery ON section.galleryID=gallery.galleryID INNER JOIN artist ON artwork.artworkArtistID = artist.artistID
* SELECT customerFirstName, customerLastName FROM customer
* SELECT artistFirstName, artistLastName, artistMovement FROM artist
* SELECT c.customerFirstName, c.customerLastName, g.galleryName, se.sectionName, a.artworkTitle, ar.artistFirstName, ar.artistLastName, sa.saleDescription, a.artworkPrice   
  FROM customer c JOIN sales sa on c.customerID = sa.customerID JOIN artwork a on sa.artworkID = a.artworkID JOIN section se on a.artWorkSectionID = se.sectionID JOIN gallery g on se.galleryID = g.galleryID JOIN artist ar on a.artworkArtistID = ar.artistID
* SELECT g.galleryName, count(c.customerID) from gallery g JOIN section s ON s.galleryID = g.galleryID JOIN artwork a ON s.sectionID = a.artworkSectionID JOIN sales sa ON a.artworkID = sa.artworkID JOIN customer c ON sa.customerID = c.customerID  
  GROUP BY g.galleryName ORDER BY galleryName
* SELECT galleryID, galleryName FROM gallery
* SELECT artistID, artistFirstName, artistLastName FROM artist
* SELECT customerID, customerFirstName, customerLastName FROM customer
* SELECT galleryID, galleryName FROM gallery
* SELECT sectionID, galleryName, sectionName FROM section JOIN gallery ON section.galleryID = gallery.galleryID
* SELECT sectionID, sectionName, galleryName FROM section JOIN gallery on section.galleryID = gallery.galleryID
* SELECT artistID, artistFirstName, artistLastName FROM artist
* SELECT artworkID, artworkTitle, artistFirstName, artistLastName FROM artwork JOIN artist ON artwork.artworkArtistID = artist.artistID
* SELECT customerID, customerFirstName, customerLastName FROM customer
* SELECT sectionID, sectionName, galleryName FROM section JOIN gallery on section.galleryID = gallery.galleryID
* SELECT customerID, customerFirstName, customerLastName FROM customer
* SELECT artworkID, artworkTitle, artworkPrice, isSold FROM artwork
* SELECT customerID, customerFirstName, customerLastName FROM customer
* SELECT customerID, customerFirstName, customerLastName FROM customer
* SELECT c.customerFirstName, c.customerLastName, g.galleryName, se.sectionName, a.artworkTitle, ar.artistFirstName, ar.artistLastName, sa.saleDescription, a.artworkPrice  
  FROM customer c JOIN sales sa on c.customerID = sa.customerID JOIN artwork a on sa.artworkID = a.artworkID JOIN section se on a.artWorkSectionID = se.sectionID JOIN gallery g on se.galleryID = g.galleryID JOIN artist ar on a.artworkArtistID = ar.artistID  
  WHERE c.customerID = ['purchList']
* SELECT customerFirstName, customerLastName FROM customer where customerLastName like 'Customers'] order by customerFirstName
* SELECT sectionName, galleryName FROM customer JOIN visitorLog ON customer.customerID = visitorLog.customerID JOIN section ON section.sectionID = visitorLog.sectionID JOIN gallery on section.galleryID = gallery.galleryID WHERE customer.customerID = ['visitedSections']

### Insertion Queries

* INSERT INTO artwork (artworkTitle, artworkYearCreated, artworkPrice, artworkSectionID, artworkArtistID) VALUES  
  (['artName'],['yearCreated'],['price'],['addSection'],['Artist'])
* INSERT INTO artist (artistFirstName, artistLastName, artistMovement) VALUES  
  (['firstName'],['lastName'],['movement'])
* INSERT INTO customer (customerFirstName, customerLastName) VALUES  
  (['custFName'],['custLName'])
* INSERT INTO gallery (galleryName, galleryCity) VALUES   
  (['galName'],['galCity'])
* INSERT INTO section (sectionName, galleryID) VALUES  
  (['sectName'],['Gallery'])
* INSERT INTO sales (saleDescription, artworkID, customerID ) VALUES (['desc'], ['Artwork'],(select customerID from visitorLog where customerID = ['Customers'] and sectionID = (select artworkSectionID from artwork where artworkID = ? limit 1 )))
* INSERT INTO visitorLog (customerID, sectionID ) VALUES (['Customers'],['Sections'])

### Deletion Queries

* DELETE from artist where artistID = ['delArtist']
* DELETE from artwork where artworkID = ['delArtwork']
* DELETE from customer where customerID = ['delCustomer']
* DELETE FROM gallery WHERE galleryID = ['delGallery']
* DELETE from section where sectionID = ['delSection']
* DELETE from sales where transactionID = ['delSale']

### Update Queries

* UPDATE artwork SET isSold=1 WHERE artworkID = ['Artwork']