GROUP 3

DBMS For Art Gallery Management

ABSTRACT

The Art Gallery Management System aims to efficiently manage and organize the database of an art gallery, providing accurate tracking and storage of art and paintings details. The system enhances safety and efficiency in managing exhibitions, gallery operations, and art stocks. Here's an abstract:

This project focuses on the design and implementation of an Art Gallery Management System, utilizing a MYSQL database platform with PHP and WAMP Server support. The system is developed to ensure effective management of art-related activities within the gallery. The primary functionalities include:

1. **Exhibition Management:** Tracks details of ongoing and upcoming exhibitions, including exhibit themes, dates, and participating artists.
2. **Gallery Management:** Manages gallery operations such as inventory management, artwork categorization, and gallery space utilization.
3. **Art Stocks:** Maintains a comprehensive database of art pieces and paintings, including details such as artist information, artwork descriptions, and pricing.

The application interface is developed using HTML5 and CSS3, with PHP facilitating dynamic content generation and interaction with the MYSQL database. Through this Art Gallery Management System, gallery administrators can streamline administrative tasks, enhance visitor experience, and effectively showcase the gallery's collection to art enthusiasts and patrons.

Group members:

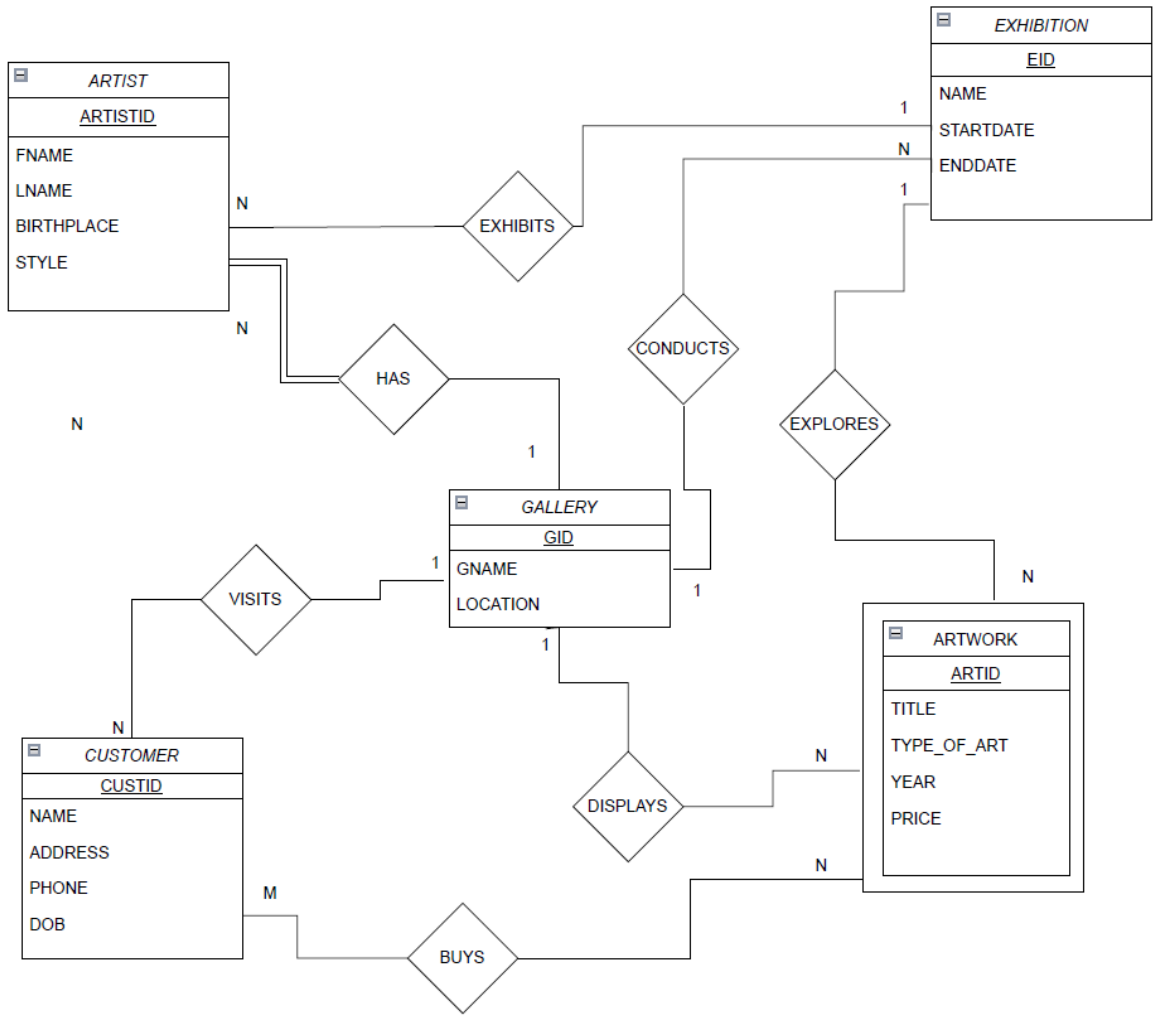
Member 1 AM.EN.U4AIE21006: Adithya S Nair

Member 2 AM.EN.U4AIE21015: Anoop Boby Manuel

Member 3 AM.EN.U4AIE21020 : Athul Gireesh

Member 4 AM.EN.U4AIE21047: Navneeth Krishna

ER-DIAGRAM



MAPPING OF ER DIAGRAM TO RELATIONS

STEP 1: Mapping of Regular Entities

For each regular entity type E in the ER schema, create relation R that includes all simple attributes of E.

GALLERY

GID	GNAME	LOCATION

EXHIBITION

EID	STARTDATE	ENDDATE

ARTIST

ARTISTID	FNAME	LNAME	BIRTHPLACE	STYLE

CUSTOMER

CUSTID	**ARTID**	FNAME1	LNAME1	ADDRESS	PHONE	DOB

STEP 2 : Mapping of Weak Entity Types

ARTWORK

ARTID	**ARTISTID**	TITLE	TYPE_OF_ART	YEAR	PRICE
	FK				

STEP 3: Mapping of 1:1 Relationship

Identify the relation S that represents the participating entity type at the 1-side of the relationship type.

Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

For each binary 1:1 relationship type R in ER schema, identify the relations S and T that correspond to the entity types participating in R if any.

There are **no** 1:1 relationship.

STEP 4 : Mapping of 1:N Relationship

EXHIBITION

EID	STARTDATE	ENDDATE	**GID**
			FK

ARTIST

ARTISTID	FNAME	LNAME	BIRTHPLACE	STYLE	**EID**	**GID**	**CUSTID**
						FK	**FK** **FK**

CUSTOMER

CUSTID	**ARTID**	FNAME1	LNAME1	STREET	CITY_ID	DOB	**GID**
	FK				**FK**	**FK**	

ARTWORK

ARTID	**ARTISTID**	TITLE	TYPE_OF_ART	YEAR	PRICE	**EID**	**GID**
	FK					**FK**	**FK**

CITY_ID

CITY_ID	CITY	STATE	ZIPCODE

STEP 5 : Mapping of M:N Relationship

Create a new relation S to represent R.

Include as foreign key attributes in S the primary key of the relations that represents the participating entity types their combination will form the primary key of S.

Also, include any simple attributes of the M:N relationship type as attributes of S.

STEP 6: Mapping of Multi-Valued Attributes

For each multivalued attributes A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-

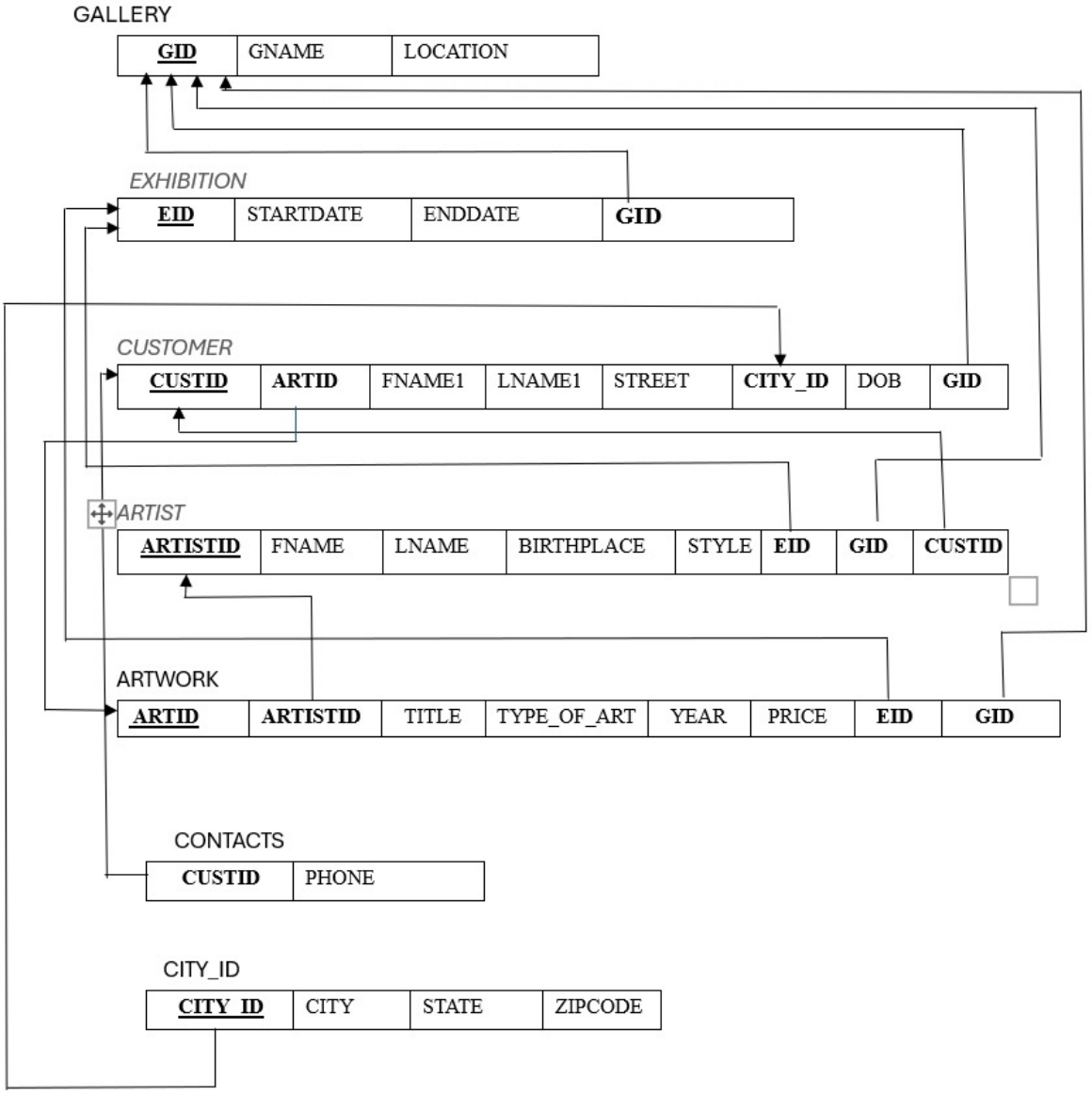
of the relation that represents the entity type of relationship type that has A as an attribute.

The Primary Key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

CONTACTS

CUSTID	PHONE

SCHEMA DIAGRAM



NORMALIZE THE RELATIONS

Database normalization, or simply normalization, is the process of organizing the columns(attributes) and tables(relations) of a relational database to reduce data redundancy and improve data integrity. Normalization involves arranging attributes in relations based on dependencies between attributes.

1. First Normal Form

As per First normal form, no two rows of data must contain repeating group of information. Each set of columns must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that will distinguishes it as unique.

Example:

GALLERY

GID	GNAME	LOCATION

All the tables in the database are normalized to 1NF as all the attributes are atomic.

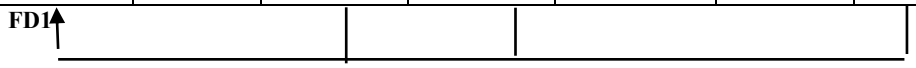
2. Second Normal Form (2NF)

A table is in 2NF if it is in 1NF and if all non-key attributes are fully functionally dependent on all of the key.

Example:

CUSTOMER

CUSTID	**ARTID**	FNAME1	LNAME1	STREET	**CITY_ID**	DOB	**GID**



FD1

CUSTID	FNAME1	LNAME1	DOB

3. Third Normal Form(3NF):

A table is in 3NF if it is in 2NF and if it has no transitive dependency. X->Y, Y->Z, X>Z

According to CODD’s definition a relation schema R is in 3NF. It satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key. All tables of database satisfies upto 3NF.

CREATION OF TABLES

Table 1: GALLERY

```
CREATE TABLE GALLERY (
   GID INT PRIMARY KEY,
   GNAME VARCHAR(100) NOT NULL,
   LOCATION VARCHAR(200) NOT NULL
);
```

Table 2: EXHIBITION

```
CREATE TABLE EXHIBITION (
   EID INT PRIMARY KEY,
   STARTDATE DATE NOT NULL,
   ENDDATE DATE NOT NULL,
   GID INT,
   FOREIGN KEY (GID) REFERENCES GALLERY(GID)
);
```

Table 3: ARTIST

```
CREATE TABLE ARTIST (
   ARTISTID INT PRIMARY KEY,
   FNAME VARCHAR(50) NOT NULL,
   LNAME VARCHAR(50) NOT NULL,
   BIRTHPLACE VARCHAR(100),
   STYLE VARCHAR(100)
);
```

Table 4: ARTWORK

```
CREATE TABLE ARTWORK (
   ARTID INT PRIMARY KEY,
   ARTISTID INT,
   TITLE VARCHAR(200) NOT NULL,
   TYPE_OF_ART VARCHAR(100),
   YEAR INT,
   PRICE DECIMAL(10, 2),
   EID INT,
   GID INT,
   FOREIGN KEY (ARTISTID) REFERENCES ARTIST(ARTISTID),
   FOREIGN KEY (EID) REFERENCES EXHIBITION(EID),
   FOREIGN KEY (GID) REFERENCES GALLERY(GID)
);
```

Table 5: CUSTOMER

```
CREATE TABLE CUSTOMER (
   CUSTID INT PRIMARY KEY,
   FNAME VARCHAR(50) NOT NULL,
   LNAME VARCHAR(50) NOT NULL,
   STREET VARCHAR(200),
   CITY_ID INT,
   DOB DATE,
   GID INT,
   FOREIGN KEY (GID) REFERENCES GALLERY(GID)
);
```

Table 6: CITY

```
CREATE TABLE CITY (
    CITY_ID INT PRIMARY KEY,
    CITY VARCHAR(100) NOT NULL,
    STATE VARCHAR(100) NOT NULL,
    ZIPCODE VARCHAR(20) NOT NULL
);
```

Table 7: CONTACTS

```
CREATE TABLE CONTACTS (
    CUSTID INT,
    PHONE VARCHAR(15),
    PRIMARY KEY (CUSTID, PHONE),
    FOREIGN KEY (CUSTID) REFERENCES CUSTOMER(CUSTID)
);
```

Additional Considerations

- All primary key constraints ensure that each record is uniquely identifiable.
- Foreign key constraints establish the relationships between tables, ensuring referential integrity.