Project I: Design & Implement a Relational Database

Problem description

Problem Statement

Many photographers have two types of accounts, namely social media accounts, and photos stock website accounts, which require them to upload the same photo twice on different platforms. If photographers are able to upload their photos to two platforms simultaneously, then they no longer need to open two websites and click repeatedly.

Objective

A database that manage the photos that photographers uploaded to it, while storing the pictures with the related EXIF data for future updating purposes. The process will read photos uploaded to social media account, store them to the database, and upload to photo stock websites with a click.

Tables and Their Roles:

1. Photos: Stores information about each photo including its dimensions, location where it was taken, and other descriptive data.
2. Photo tags: Contains tags associated with each photo for easier categorization and searchability.
3. Photographers: Holds details about photographers such as their name and contact information.
4. Photographer account: Links photographers to their accounts on social media platforms and photo stock websites.
5. Time for posting: Manages scheduling for posts, allowing photos to be uploaded at specific times.
6. Authentication: Stores authentication details necessary for accessing the artist's social media and photo stock website accounts securely through APIs.
7. Camera information: Stores information about the camera used for taking a photograph, including model name and lens type.

Identified Nouns:

- Photographers
- Photos
- Process
- Database
- Photo stock websites
- Social media account

Identified Actions

- update
- simplifies
- upload
- storing
- enable

Feature Summary

1. Uploading Photos: Photographers can upload photos directly through this platform instead of visiting each platform individually.
2. Batch Uploading: Photographers can upload a batch (series) of photo through this platform to different platform.
3. Scheduling Uploads: Allows setting specific times when photos should be posted automatically.

UML Diagram

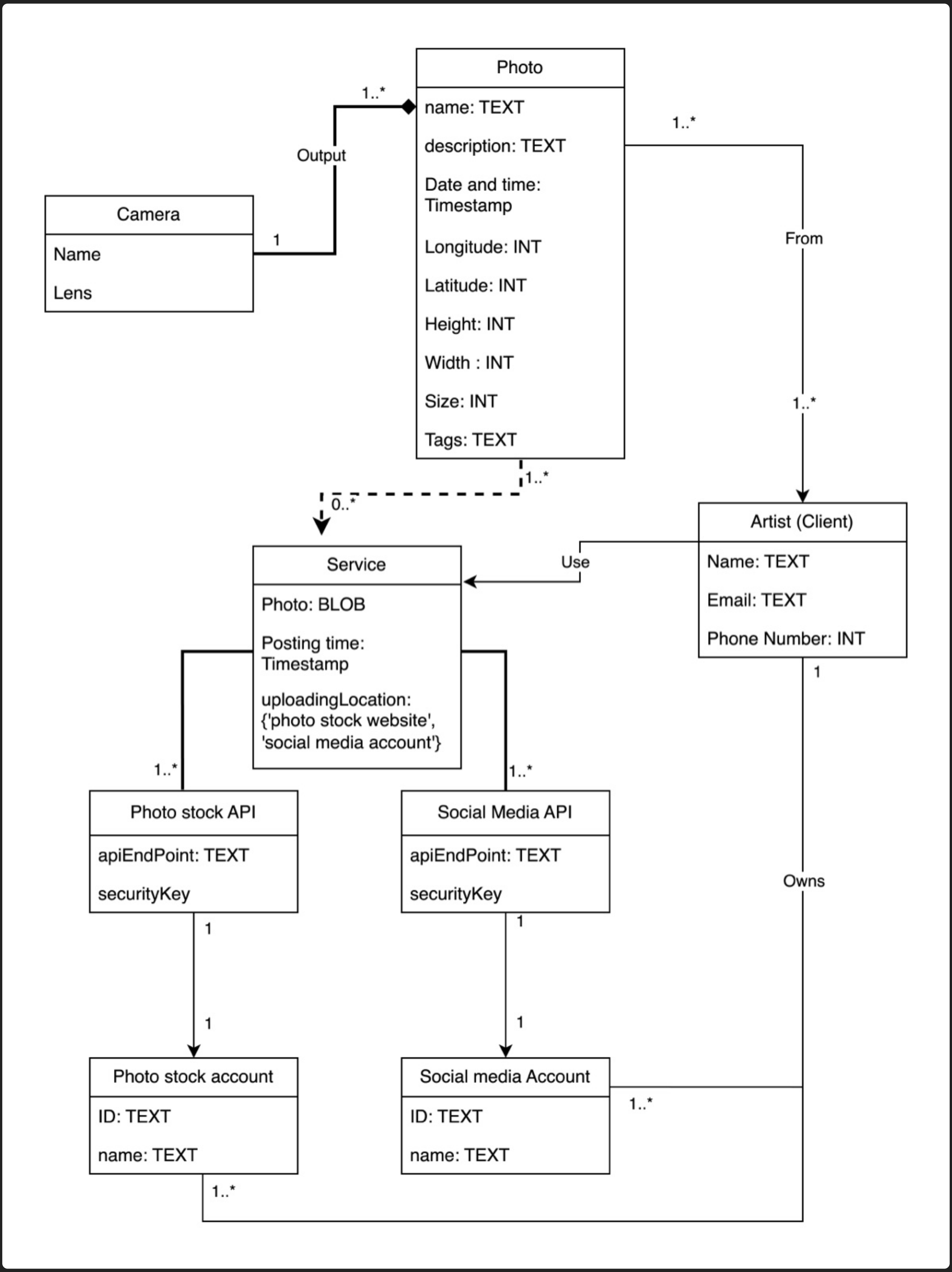
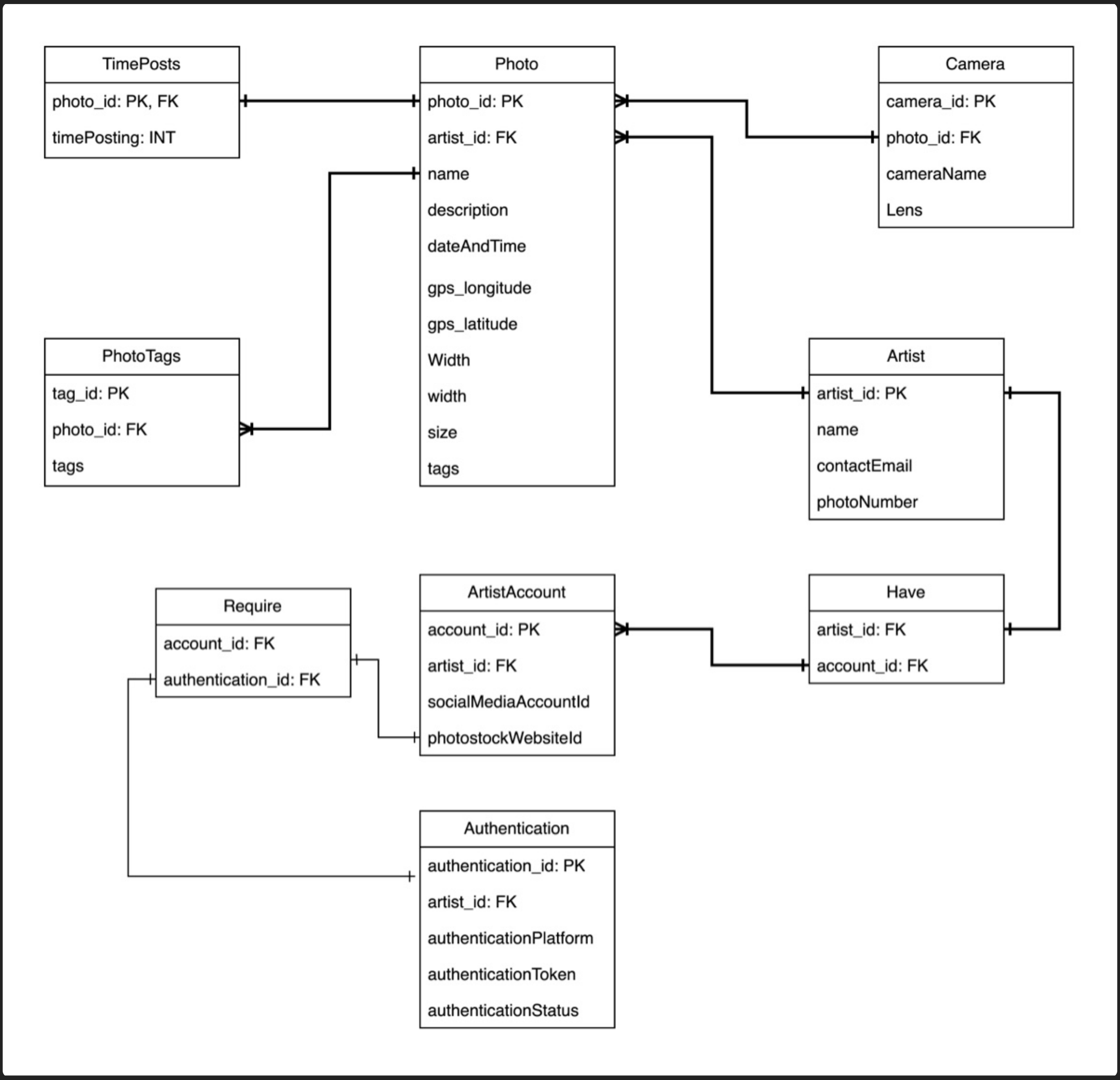


Photo
name: TEXT
description: TEXT
Date and time: Timestamp
Longitude: INT
Latitude: INT
Height: INT
Width : INT
Size: INT
Tags: TEXT
Camera
Name
Lens
Output
1
1..*
From
1..*
1..*
1..*
0..*
Artist (Client)
Name: TEXT
Email: TEXT
Phone Number: INT
Use
Service
Photo: BLOB
Posting time: Timestamp
uploadingLocation: {'photo stock website', 'social media account'}
1..*
1..*
Photo stock API
apiEndPoint: TEXT
securityKey
Social Media API
apiEndPoint: TEXT
securityKey
1
1
1
1
Photo stock account
ID: TEXT
name: TEXT
Social media Account
ID: TEXT
name: TEXT
1
Owns
1..*
1..*

Entity-Relation Diagram



drive.google.com
https://drive.google.com/file/d/1GuhTgij9uKoI4wRT_NRLEbu7XHggXcGE/view?usp=drive_link

https://drive.google.com/file/d/1GuhTgij9uKoI4wRT_NRLEbu7XHggXcGE/view?usp=drive_link

Relational Schema

1. Photos (photo_id PK, artist_id FK, title, description, dateAndTimeTaken, gps_longitude, gps_latitude, length, width, size)
2. PhotoTags (tag_id PK, photo_id FK, tags)
3. Artist (artist_id PK, name, contactEmail, phoneNumber)

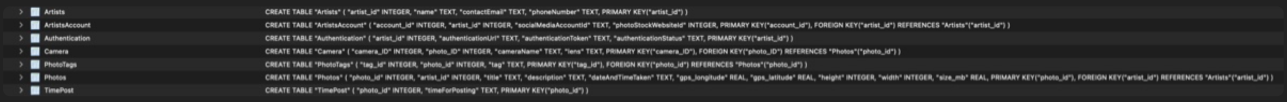
4. ArtistAccount (account_id PK, artist_id FK, socialMediaAccountId , photoStockWebsiteId)
5. TimePosts (photo_id PK FK, timeForPosting)
6. Authentication(artist_id PK FK ,authenticationUrl , authenticationToken , authenticationStatus)
7. Camera(camera_ID PK , photo_ID FK, cameraName, lens)

Functional Dependencies Identification

1. Photos: photo_id → artist_id, title, description, dateAndTimeTaken, gps_longitude, gps_latitude, length, width, size
2. PhotoTags: tag_id → photo_id, tags
3. tag_id → photo_id, tags
4. account_Id → artist_Id , socialMediaAccountId , photoStockWebsiteId
5. photo_Id → timeForPosting
6. artist_Id → authenticationUrl , authenticationToken , authenticationStatus
7. camera_ID → photo_ID , cameraName , lens

- It could be seen that attributes in each relation are dependent and only depend on the primary key, so the BCNF standard is followed.

SQL database creation

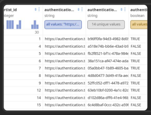


```
Artists          CREATE TABLE "Artists" ( "artist_id" INTEGER, "name" TEXT, "contactEmail" TEXT, "phoneNumber" TEXT, PRIMARY KEY("artist_id") )
ArtistsAccount   CREATE TABLE "ArtistsAccount" ( "account_id" INTEGER, "artist_id" INTEGER, "socialMediaAccountId" TEXT, "photoStockWebsiteId" INTEGER, PRIMARY KEY("account_id"), FOREIGN KEY("artist_id") REFERENCES "Artists"("artist_id") )
Authentication   CREATE TABLE "Authentication" ( "artist_id" INTEGER, "authenticationUrl" TEXT, "authenticationToken" TEXT, "authenticationStatus" TEXT, PRIMARY KEY("artist_id") )
Camera           CREATE TABLE "Camera" ( "camera_ID" INTEGER, "photo_ID" INTEGER, "cameraName" TEXT, "lens" TEXT, PRIMARY KEY("camera_ID"), FOREIGN KEY("photo_ID") REFERENCES "Photos"("photo_id") )
PhotoTags        CREATE TABLE "PhotoTags" ( "tag_id" INTEGER, "photo_id" INTEGER, "tag" TEXT, PRIMARY KEY("tag_id"), FOREIGN KEY("photo_id") REFERENCES "Photos"("photo_id") )
Photos           CREATE TABLE "Photos" ( "photo_id" INTEGER, "artist_id" INTEGER, "title" TEXT, "description" TEXT, "dateAndTimeTaken" TEXT, "gps_longitude" REAL, "gps_latitude" REAL, "height" INTEGER, "width" INTEGER, "size_mb" REAL, PRIMARY KEY("photo_id"), FOREIGN KEY("artist_id") REFERENCES "Artists"("artist_id") )
TimePost         CREATE TABLE "TimePost" ( "photo_id" INTEGER, "timeForPosting" TEXT, PRIMARY KEY("photo_id") )
```

Database created by using DB Browser

```
CREATE TABLE "Artists" (
    "artist_id"   INTEGER,
    "name" TEXT,
    "contactEmail"   TEXT,
    "phoneNumber" TEXT,
    PRIMARY KEY("artist_id")
);

CREATE TABLE "ArtistsAccount" (
    "account_id"  INTEGER,
    "artist_id"   INTEGER,
    "socialMediaAccountId"  TEXT,
    "photoStockWebsiteId"   INTEGER,
    FOREIGN KEY("artist_id") REFERENCES "Artists"("artist_id"),
    PRIMARY KEY("account_id")
);

CREATE TABLE "Authentication" (
    "artist_id"   INTEGER,
    "authenticationUrl"  TEXT,
    "authenticationToken"   TEXT,
    "authenticationStatus"  TEXT,
    PRIMARY KEY("artist_id")
);

CREATE TABLE "Camera" (
    "camera_ID"   INTEGER,
    "photo_ID" INTEGER,
    "cameraName"  TEXT,
    "lens" TEXT,
    FOREIGN KEY("photo_ID") REFERENCES "Photos"("photo_id"),
    PRIMARY KEY("camera_ID")
);

CREATE TABLE "PhotoTags" (
    "tag_id"   INTEGER,
    "photo_id" INTEGER,
    "tag"  TEXT,
    PRIMARY KEY("tag_id")
);

CREATE TABLE "Photos" (
    "photo_id" INTEGER,
    "artist_id"   INTEGER,
    "title"   TEXT,
    "description" TEXT,
    "dateAndTimeTaken"   TEXT,
    "gps_longitude"  REAL,
    "gps_latitude"   REAL,
    "height"   INTEGER,
    "width"    INTEGER,
    "size_mb"  REAL,
    FOREIGN KEY("artist_id") REFERENCES "Artists"("artist_id"),
    PRIMARY KEY("photo_id")
);

CREATE TABLE "TimePost" (
    "photo_id" INTEGER,
    "timeForPosting"  TEXT,
    PRIMARY KEY("photo_id")
```

SQL Queries



BUILD: Project I / Design & Implement a Relational Database

7. Define and execute at least five queries that show your database. (10 pts) Define and execute at least f...

https://observablehq.com/@borisshi/build-project-i-design-implement-a-relational-database