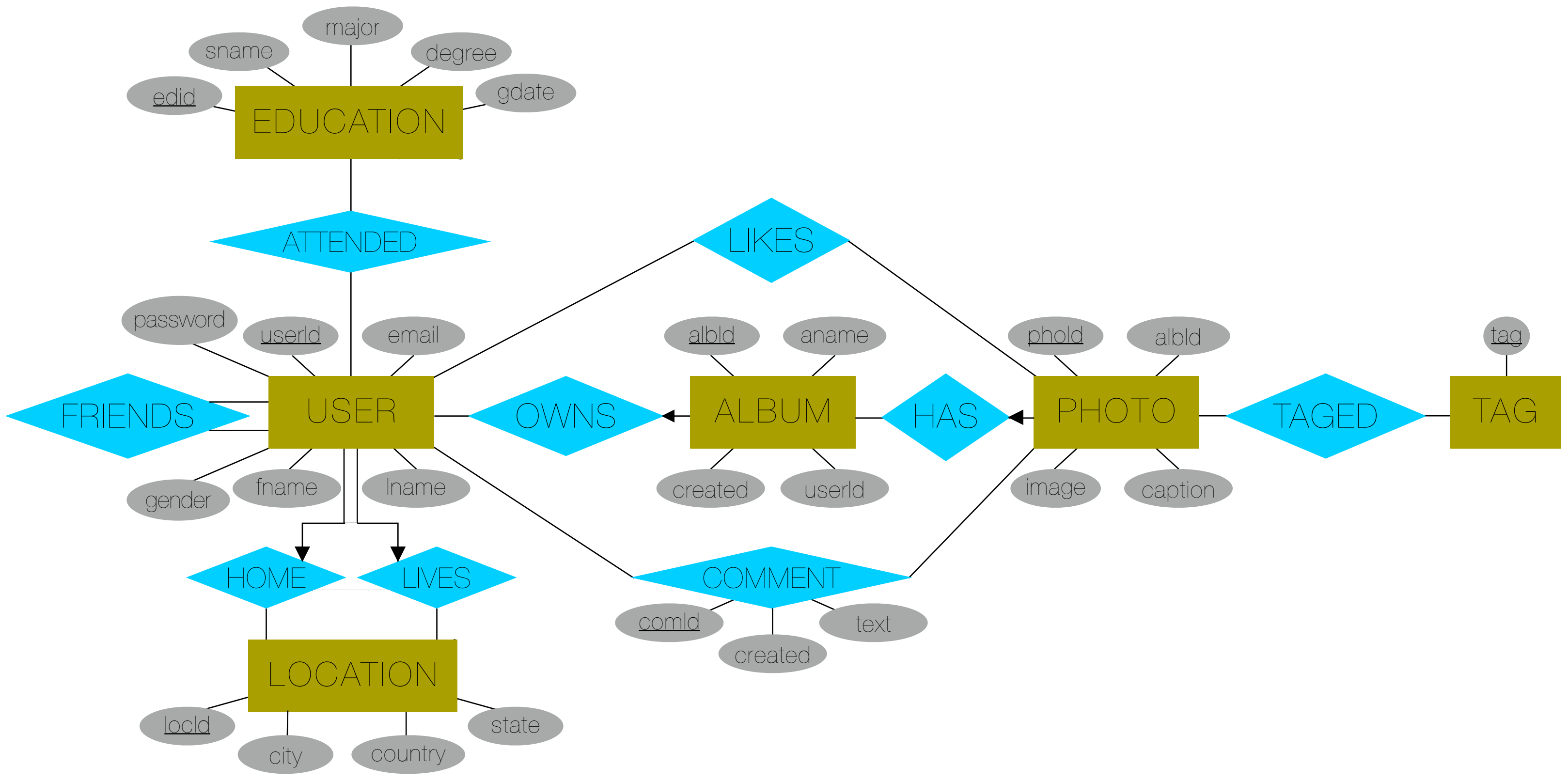
CS460

Nicholas Lippis

Programming Assignment 1

Part 1

E-R Diagram



ASSUMPTIONS

User Deletion

The application specification does not provide a user story for deleting a users account, so no functionality for account deletion is provided.

User Location & Education

Friends often live in close proximity to each other; so storing the hometown, current location and education of each user in the user entity will create extraneous duplicates. In order to mitigate this I have created the Location and Education

Entities. The Location entity stores a city, state, and country name; it is referenced in the user table through the home and lives attributes so locations are stored once. The Education entity stores school name, major, degree and graduation date; it is related to the user through the attended relation so educations are stored once.

Likes

Though it was not part of the schema specification, the application is supposed to keep track of likes. In order to implement this I created a Likes relation, which simply stores the primary keys of the user and the liked photo.

Comments

Comments are dependent on both the user who wrote them and the photo they are associated with in a 1:1 relationship; so instead of making comments an entity I made it a relation between the two.

Tags

Though tags could simply be stored in just the taged relation, if for some reason all photos with that tag are removed then the database will have no record of that tags existence. So in order to provide a concrete record of all tags that have been created, I have created the Tag entity and related it to photos through the taged relation.

Other

Metrics such as most popular tags and most active users can be generated by queries on page load. No need to store the data in the database

SQL Commands

Entities

User Table	Assumptions

CREATE TABLE user
(
    userId INTEGER PRIMARY KEY,
    email VARCHAR(50) UNIQUE,
    fname VARCHAR(40) NOT NULL,
    lname VARCHAR(40) NOT NULL,
    gender VARCHAR(6) NOT NULL,
    password VARCHAR(20) NOT NULL,
    home INTEGER,
    lives INTEGER,
    FOREIGN KEY home REFERENCES location(locId),
    FOREIGN KEY lives REFERENCES location(locId)
)
	The specification requires there to be an error message when a duplicate value for email is being entered into the table. I have made email a candidate key in order to achieve this.
	
	Gender is either the string male or female, so i restrict its length to a maximum of 6 characters
	
	I require all fields to be not null so that the database maintains a consistent data representation of each user.
	
	For each user there is one home & lives location so a reference to the location table is stored in the user table

Location Table	Assumptions

CREATE TABLE location
(
    locId INTEGER PRIMARY KEY,
    city VARCHAR(20) NOT NULL,
    state VARCHAR(20) NOT NULL,
    country VARCHAR(20) NOT NULL
)
	I require all fields to be not null so that the database maintains a consistent data representation of each user.

Education Table	Assumptions

CREATE TABLE education
(
    edId INTEGER PRIMARY KEY,
    schoolname VARCHAR(40) NOT NULL,
    major VARCHAR(40),
    degree VARCHAR(40),
    gdate DATE
)
	In order to cater to both university and high school students I only require the school name to be provided, and users have the option of providing their major, degree, and graduation date, which gives a appropriate representation of their education if they have attended a university for undergraduate or graduate studies.

Album Table	Assumptions

CREATE TABLE album
(
    albId INTEGER PRIMARY KEY,
    aname VARCHAR(40) NOT NULL,
    created DATE NOT NULL,
    userId INTEGER,
    FOREIGN KEY (userId) REFERENCES user(userId)
)
	For each album there is one owner so a reference to the user is stored in the album table.
	
	I require all fields to be not null so that the database maintains a consistent data representation of each user.

Photo Table	Assumptions

CREATE TABLE photo
(
    phoId INTEGER PRIMARY KEY,
    image BINARY NOT NULL,
    caption VARCHAR(500),
    albId INTEGER,
    FOREIGN KEY (albId) REFERENCES album(albId)
    ON DELETE CASCADE
)
	For each photo there is one associated album, so a reference to the album is stored in the photo table.
	
	I require the image file to be provided because otherwise there would be nothing to show. The caption, however is optional.
	
	If an photo’s associated album is deleted then the photo is deleted

Tag Table	Assumptions

CREATE TABLE tag
(
    tag VARCHAR(20) PRIMARY KEY
)
	Tag just one word that can be up to 20 characters long
	
	Because the same tag can be used for pictures across all accounts there is no need to provide a guid

SQL Commands

Relations

Friends Relation	Assumptions

CREATE TABLE friends
(
    friendId1 INTEGER,
    friendId2 INTEGER,
    FOREIGN KEY (friendId1) REFERENCES user(userId)
    FOREIGN KEY (friendid2) REFERENCES user(userid)
    PRIMARY KEY (friendId1, friendId2)
)
	Each user can have many friends and vice versa, so a relation table is required.

Attended Relation	Assumptions

CREATE TABLE attended
(
    eId INTEGER,
    userId INTEGER,
    FOREIGN KEY (eId) REFERENCES education(eId)
    FOREIGN KEY (userid) REFERENCES user(userid)
    PRIMARY KEY (eId, userId)
)
	Each user can have attended multiple schools (high school, undergraduate, masters) so the relation is many to many. Thus a relation table is required.

Likes Relation	Assumptions

CREATE TABLE likes
(
    userId INTEGER,
    phoId INTEGER,
    FOREIGN KEY (userId) REFERENCES user(userId)
    FOREIGN KEY (phoId) REFERENCES photo(phoid)
    ON DELETE CASCADE,
    PRIMARY KEY (userId, phoId)
)
	Each user can have liked many photos and vice versa so a relation table is required.
	
	If a photo is deleted then the like is deleted

Comment Relation	Assumptions
CREATE TABLE comment (comId INTEGER PRIMARY KEY, userId INTEGER, phoId INTEGER, text VARCHAR(500) NOT NULL, created DATE NOT NULL, FOREIGN KEY (userId) REFERENCES user(userId) FOREIGN KEY (phoId) REFERENCES photo(phoid) ON DELETE CASCADE, PRIMARY KEY (userId, phoId))	Each user can have made many comments and vice versa so a relation table is required. I require all fields to be not null so that the database maintains a consistent data representation of each user. The user can make a 500 character long comment (arbitrarily picked). If a photo is deleted the comment is also deleted.

Taged Relation	Assumptions
CREATE TABLE taged (tag VARCHAR(20), phoId INTEGER, FOREIGN KEY (tag) REFERENCES tag(tag) FOREIGN KEY (phoId) REFERENCES photo(phoId) ON DELETE CASCADE, PRIMARY KEY (tag, phoId))	Each photo can have multiple tags and vice versa so a relation table is required. If a photo is deleted then the tag relation is deleted

Integrity Constraints

Assertions

No Self Friends Assertion	Assumptions

CREATE ASSERTION noSelfFreinds
    CHECK NOT EXISTS
    (
        SELECT *
        FROM friends f
        WHERE f.friendId1 = f.friendId2
)
	A friend should not be friends with themselves

User Cant Comment Own Photo Assertion	Assumptions

CREATE ASSERTION userCantCommentOwnPhoto
    CHECK NOT EXISTS
    (
        SELECT *
        FROM album a, photo p, comment c
        WHERE a.albId = p.albId AND
                p.phoId = c.phoId AND
                a.userId = c.userId
)
	The specification stated that a user can not comment on their own photo. This assertion makes sure that it does not exist.