CS 340 Project

Natalia Quintero Echeverri
http://web.engr.oregonstate.edu/~quintern/home

Content

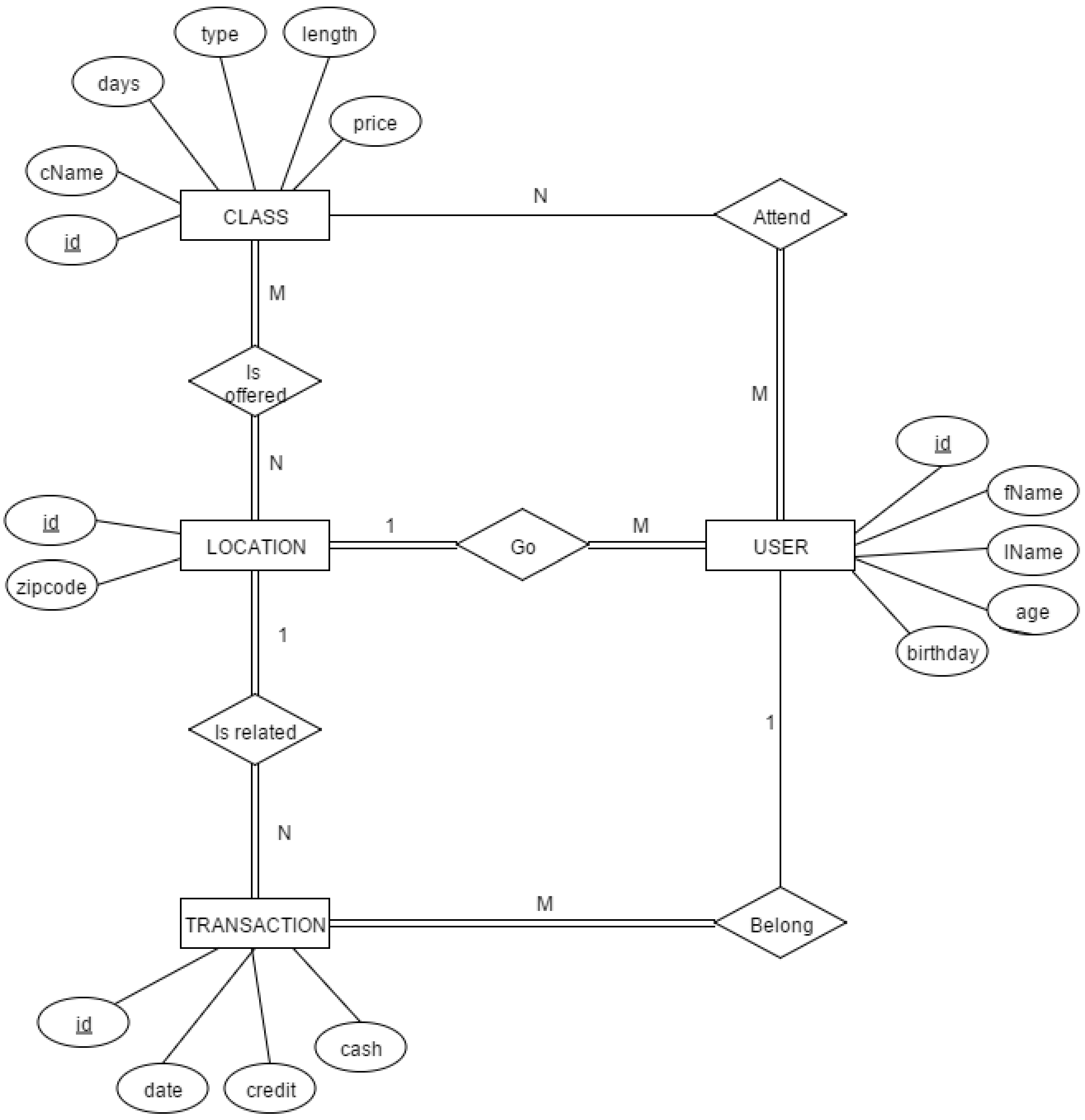
This is a data base for a dance studio that keeps track of the users and classes in different locations. It provides the studio with client-data reports and will be useful for business intelligence.
This data base tracks attendance trends, offers a client profile and displays the schedule of classes in different locations.

Situation

- A user has an ID, name, last name, birthday, and age. An ID uniquely identifies each user.
- A class has an ID, name, a class type (solo or with a partner), length, price, and days offered. An ID uniquely identifies each class.
- A location has an ID and a zip code. Each location is identified by its location ID.
- A transaction has an ID, cash payment or credit card, and a transaction date. Each transaction is uniquely identified by its ID.

- Each class is offered in multiple locations, and each location can offer multiple classes. (m – n)
- A user can attend many classes and a class can be attended by zero or more users. (m – n)
- Each user may have zero, one, or multiple transactions but a transaction belongs to only one user.
- A user can go to one location, and a location can have many users.
- A location is related to many transactions but a transaction is related to exactly one location.

ER Diagram


type
length
days
price
cName
CLASS
N
Attend
id
M
Is offered
M
N
id
id
fName
LOCATION
1
Go
M
USER
lName
zipcode
age
1
birthday
Is related
1
N
M
TRANSACTION
Belong
id
cash
date
credit

Schema

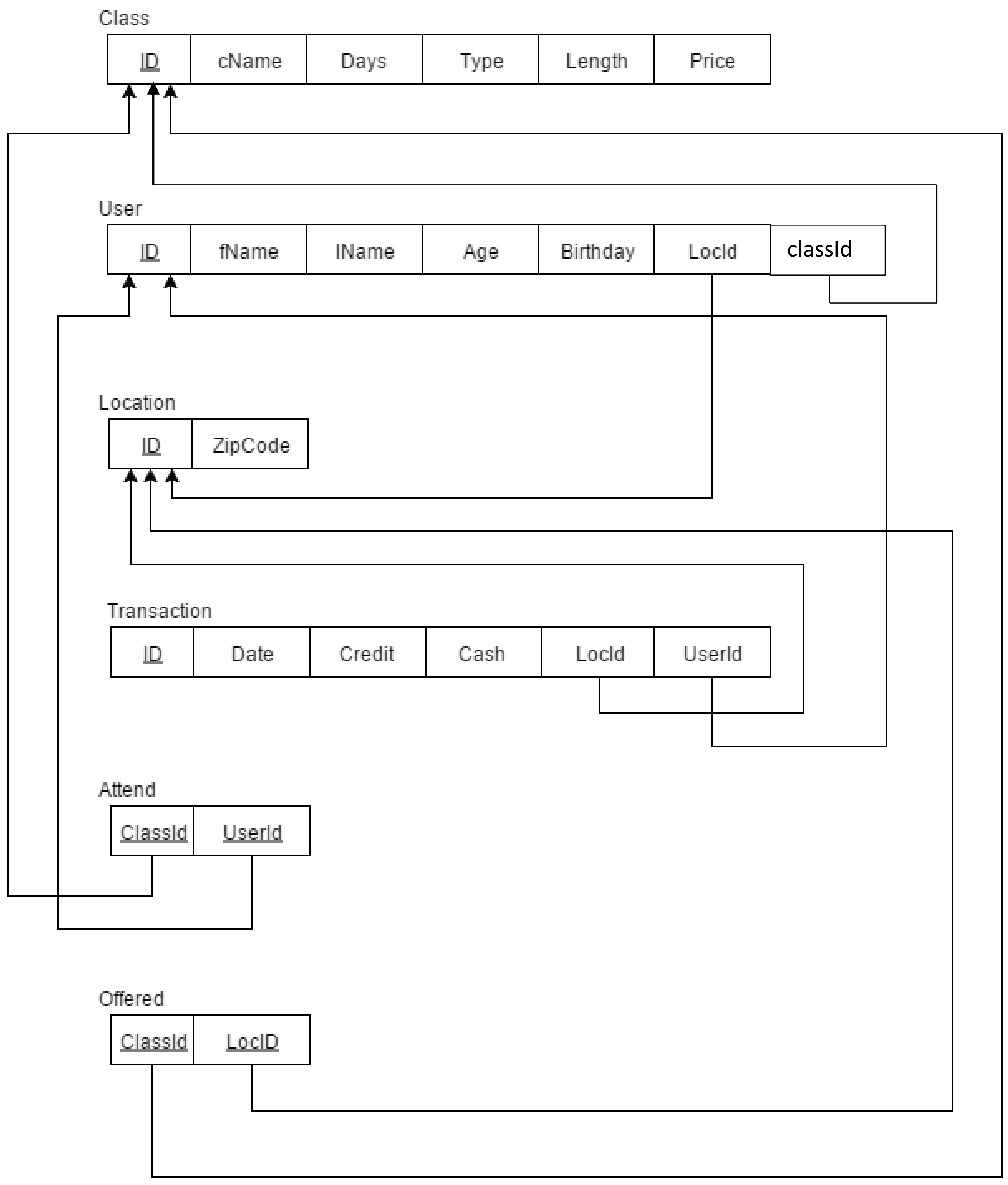

Class
ID
cName
Days
Type
Length
Price
User
ID
fName
lName
Age
Birthday
LocId
classId
Location
ID
ZipCode
Transaction
ID
Date
Credit
Cash
LocId
UserId
Attend
ClassId
UserId
Offered
ClassId
LocID

Table Creation Queries

```
DROP TABLE IF EXISTS ‘class’;
DROP TABLE IF EXISTS ‘user’;
DROP TABLE IF EXISTS ‘location’;
DROP TABLE IF EXISTS ‘transaction’;
DROP Table IF EXISTS ‘attend’;
DROP TABLE IF EXISTS ‘offered’;


CREATE TABLE class(
classId INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
cName VARCHAR(20) NOT NULL,
days VARCHAR(20) NOT NULL,
type ENUM('Solo', 'Partner'),
length SMALLINT UNSIGNED NOT NULL,
price DECIMAL(5,2) NOT NULL,
classId INT,
location INT UNSIGNED NOT NULL,
PRIMARY KEY (classId)
)ENGINE=InnoDB;


CREATE TABLE location(
locationId INT (11) UNSIGNED NOT NULL AUTO_INCREMENT,
Zipcode MEDIUMINT UNSIGNED DEFAULT NULL,
PRIMARY KEY (locationId)
)ENGINE=InnoDB;


CREATE TABLE user(
userId INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
fName VARCHAR(20) NOT NULL,
lName VARCHAR(20),
age TINYINT UNSIGNED,
birthday DATE,
locationId INT(11) UNSIGNED NOT NULL,
PRIMARY KEY (userId),
KEY idx_fk_locationId (locationId),
CONSTRAINT fk_user_location FOREIGN KEY (locationId) REFERENCES location (locationId) ON
DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;
```

```
CREATE TABLE transaction(
transId INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
transdate DATE NOT NULL,
credit DECIMAL(5,2),
cash DECIMAL(5,2),
locationId INT(11) UNSIGNED NOT NULL,
userId INT(11) UNSIGNED NOT NULL,
PRIMARY KEY (transId),
KEY idx_fk_locationId (locationId),
KEY idx_fk_userId (userId),
CONSTRAINT fk_trans_location FOREIGN KEY (locationId) REFERENCES location (locationId) ON
DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT fk_trans_user FOREIGN KEY (userId) REFERENCES user (userId) ON DELETE CASCADE ON
UPDATE CASCADE
) ENGINE=INNODB;


CREATE TABLE attend(
classId INT(11) UNSIGNED NOT NULL,
userId INT(11) UNSIGNED NOT NULL,
PRIMARY KEY (classId, userId),
KEY idx_userId (userId),
CONSTRAINT fk_attend_class FOREIGN KEY (classId) REFERENCES class (classId) ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT fk_attend_user FOREIGN KEY (userId) REFERENCES user (userId) ON DELETE CASCADE
ON UPDATE CASCADE
)ENGINE=InnoDB;


CREATE TABLE offered(
classId INT(11) UNSIGNED NOT NULL,
locationId INT(11) UNSIGNED NOT NULL,
PRIMARY KEY (classId, locationId),
KEY idx_classId (classId),
CONSTRAINT fk_offered_class FOREIGN KEY (classId) REFERENCES class (classId) ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT fk_offered_loc FOREIGN KEY (locationId) REFERENCES location (locationId) ON DELETE
CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;
```

General Use Queries

Add/ Insert queries

```
INSERT INTO class (cName, days, type, length, price)
VALUES ([name_of_class], [day_of_week], [type], [length], [price]);

INSERT INTO location (Zipcode)
VALUES ([Zip_code]);

INSERT INTO user(fName, lName, age, birthday, locationId, classId)
VALUES ([client_first_name], [client_last_name], [age], [birthday], [location_id], [class_Id] );

INSERT INTO transaction(transdate, credit, cash, locationId, userId)
VALUES ([transaction_date], [credit_amount], [cash_amount], [location_id], [user_id]);

INSERT INTO attend( classId, userId )
SELECT class.classId, user.userId
FROM class
INNER JOIN user ON class.classId = user.classId;

INSERT INTO offered (classId, locationId)
SELECT class.classId, location.locationId
FROM class
INNER JOIN location ON class.location = location.locationId;
```

Select queries

```
-- Find which classes are available on a given day.
SELECT cName
FROM class
WHERE days = [day_input];

-- Find what classes users are taking.
SELECT user.fName, user.lName, class.cName
FROM user
INNER JOIN class ON class.classId = user.classId
ORDER BY userId;
```

***This query will display a complete list of users, if looking for a particular client add to the query: WHERE user.fname=[first_name_input] AND user.lName=[last_name_input];*

```
-- Find the schedule of classes in all locations.
SELECT class.cName, class.days, location.zipcode
FROM class
INNER JOIN offered ON class.classId = offered.classId
INNER JOIN location ON location.locationId = offered.locationId
ORDER BY zipcode;

-- Find the total amount of transactions at each location. Add all credit and cash columns for
payments made partially.
SELECT transaction.locationId, SUM(transaction.credit + transaction.cash) AS ‘Total Amount’
FROM transaction
GROUP BY locationId;
```

Update queries

```
-- Update day and price of a class.
UPDATE class
SET days = [day_input], price = [price_input]
WHERE cName = [class_name_input];

-- Moving a class to a different location.
UPDATE class
SET location = [location_id_input]
WHERE classId = [class_id_input];
```

Delete querie

```
-- Delete a class.
DELETE FROM class
WHERE cName = [class_name_input];
```