CS3431 (Database Systems I)
Midterm Exam
C-term, 2013

Solution

Consider the following requirements for a ***library database***

- The library has many books, for each book we have its ISBN (unique), title, the number of copies, number of pages in this book, and the price.

- The library has two types of customers, either registered customers or un-registered customers. For all customers, we keep the ID (unique), name, DoB. For registered customers we keep additional information such as Registration ID, Tel. number, and the discount offer(s) available for that customer (can have several offers).

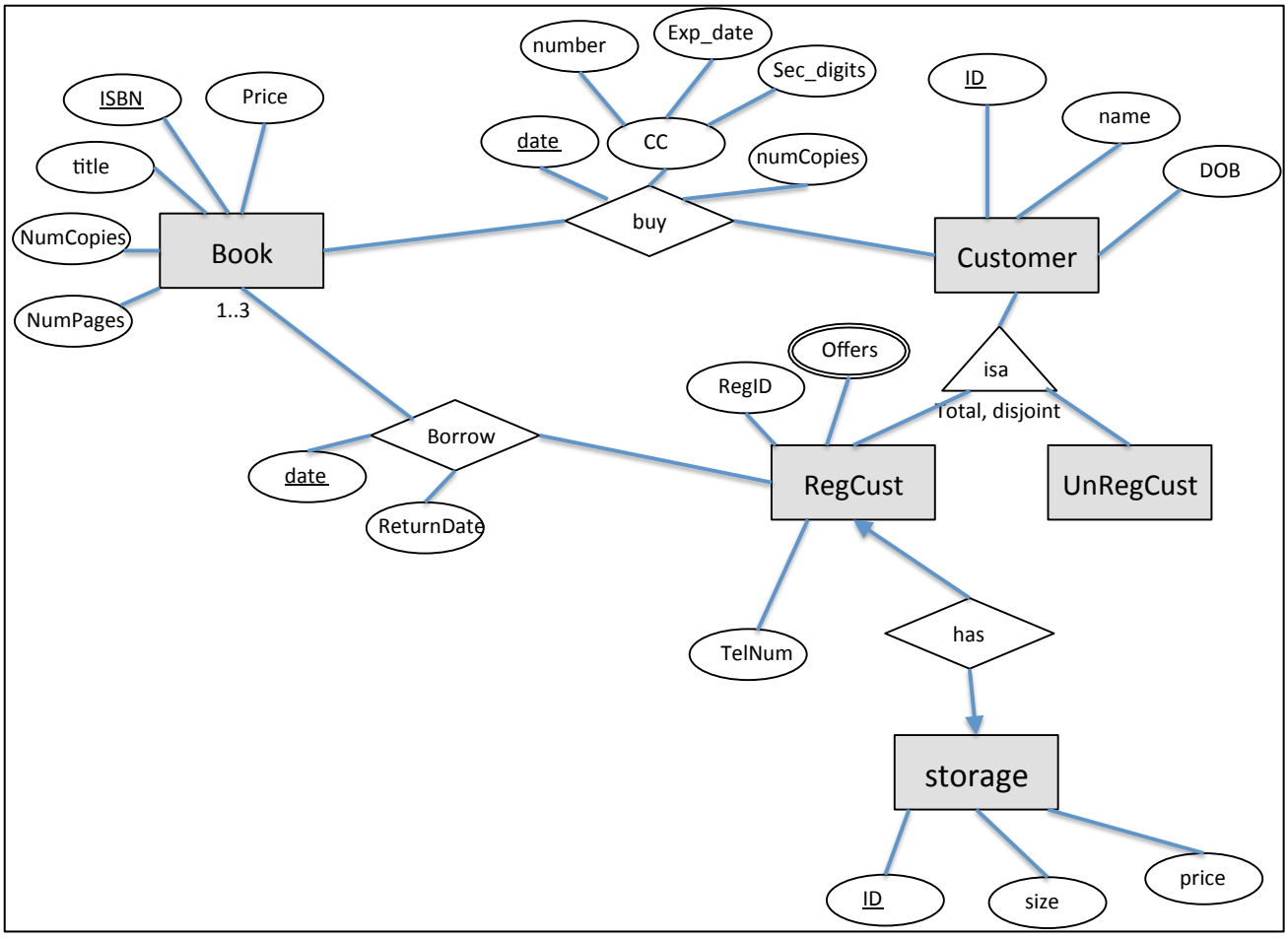
- Customers (either registered or not) can buy books; each customer can buy many books and can buy the same book on different dates. We want to capture the purchase date, the number of copies purchased, and the credit card (CC) info used in the transaction (CC number, expiry date, and 3-digit security number).

- Only registered customers can borrow books, where each borrow transaction has a borrow date and it can contain at most 3 books. Each borrowed book can be returned on a different date that we want to capture.

- The library maintains storage areas only for registered customers. Each area has an ID (unique), size, and rented price. Each area belongs to at most one customer and each customer can have at most one storage area.

Question 1 (ER Model) [20 points]

Create the Entity-Relationship model (ERD) for the application mentioned above.



Question 2 (Relational Model) [20 points]

- Convert the ERD you designed in Question 1 to its corresponding relational model. For each relation, write it in the form of R(A1, A2, ….An) and underline the primary keys.

- Also state the foreign keys in the form of: R.A1 (FK) References S.A2 (PK)

Book(ISBN, title, price, NumCopies, NumPages);

Customer(ID, name, DOB);

RegCustomer(CustID, RegID, TelNum, storageID);

RegCustOffer(CustID, Offer);

Storage(ID, size, price)

Buy(CustID, ISBN, date, numCopies, CC_exp_date, CC_number, CC_sec_digits);

Borrow(CustID, ISBN, date, returnDate);

Foreign Keys

RegCustomer.CustID (FK) references Customer.ID (PK)

RegCustomer.StorageID (FK) references Storage.ID (PK)

RegCustOffer.CustID (FK) references Customer.ID (PK)

Buy.CustID (FK) references Customer.ID (PK)

Buy.ISBN (FK) references Book.ISBN (PK)

Borrow.CustID (FK) references Customer.ID (PK)

Borrow.ISBN (FK) references Book.ISBN (PK)

Question 3 (Relational Algebra & SQL Queries)[50]

Given the following relations from a university registration system:

Department(ID, name, address) ---------- short name D

Faculty(ID, deptID, firstName, lastName, joinYear) ---------- short name F

Course(ID, deptID, name, numOfCredits) ---------- short name C

Student(ID, deptID, firstName, LastName, joinYear) ---------- short name S

Teaching(FacultyID, courseID, Year) ---------- short name T

Registration(studentID, courseID, Year, grade) ---------- short name R

The primary key in each relation is underlined. Each faculty member and each course belongs to exactly one department. Each student also belongs to one department. However, a student can take courses outside his/her department. Each faculty can teach many courses each year, and each taught course must have at least 10 students.

Write the *algebraic expression* and *SQL statement* for the following queries:

Only SQL statements are given. The algebraic expression should be straightforward from the Select statements**.

(1) [5 points]: Report faculty members who have joined after 2005. Sort them descending by their last names

```
Select *
From F
Where joinYear > 2005
Order By lastName desc;
```

(2) [5 points]: Report faculty members (Ids and names) who have taught courses outside their departments (that is, the course and the faculty belong to different departments)

```
Select F.ID, F.firstName, F.lastName
From F, C, T
Where F.ID = T.FacultyID
And C.ID = T.CourseID
And C.deptID != F.deptID;
```

Department(ID, name, address) ---------- short name D

Faculty(ID, deptID, firstName, lastName, joinYear) ---------- short name F

Course(ID, deptID, name, numOfCredits) ---------- short name C

Student(ID, deptID, firstName, LastName, joinYear) ---------- short name S

Teaching(FacultyID, courseID, Year) ---------- short name T

Registration(studentID, courseID, Year, grade) ---------- short name R

(3) [5 points]: Report the department DIs that have more than 20 faculty members and offer more than 10 courses with numOfCredits = 3.

```
    Select deptID
    From F
    Group By deptID
    Having count(*) > 20
Intersect
    Select deptID
    From C
    Where numOfCredits = 3
    Group By deptID
    Having count(*) > 10;
```

(4) [5 points]: Report the distinct course IDs that have been taught in 3 consecutive years between 2000 and 2010 (inclusive).

```
Select  distinct t1.courseID
From    T t1, T t2 , T t3
Where t1.courseID = t2.courseID
And     t2.courseID = t3.courseID
And  t1.year = t2.year -1
And  t2.year = t3.year -1
And t1.year >= 2000
And t3.year <= 2010;
```

Department(ID, name, address) ---------- short name D

Faculty(ID, deptID, firstName, lastName, joinYear) ---------- short name F

Course(ID, deptID, name, numOfCredits) ---------- short name C

Student(ID, deptID, firstName, LastName, joinYear) ---------- short name S

Teaching(FacultyID, courseID, Year) ---------- short name T

Registration(studentID, courseID, Year, grade) ---------- short name R

(5) [5 points]: Report the course IDs that have more than 50 registered students in year 2010

Select courseID
From R
Where year = 2010
Group By course ID
Having count(*) > 50;

(6) [5 points]: Update the join year for faculty ID = 12345 to be 2007.
Write only the SQL statement, no algebraic expression

Update Faculty
Set joinYear = 2007
Where ID = 12345;

Department(ID, name, address) ---------- short name D

Faculty(ID, deptID, firstName, lastName, joinYear) ---------- short name F

Course(ID, deptID, name, numOfCredits) ---------- short name C

Student(ID, deptID, firstName, LastName, joinYear) ---------- short name S

Teaching(FacultyID, courseID, Year) ---------- short name T

Registration(studentID, courseID, Year, grade) ---------- short name R

(7) [5 points]: Delete records from the *Registration* table for year 2000 and that have courses belong to department ID 'CS'.

Write only the SQL statement, no algebraic expression

```
Delete From R
Where year = 2000
And  courseID in (Select Id from C Where deptID = 'CS');
```

(8) [5 points]: Report the years that have either more than 50 courses or less than 5 courses taught by faculty. Report the year and the count of courses

```
Select year, count(*) as cnt
From  T
Group By year
Having count(*) > 50
Or count(*) < 5;
```

Department(ID, name, address) ---------- short name D

Faculty(ID, deptID, firstName, lastName, joinYear) ---------- short name F

Course(ID, deptID, name, numOfCredits) ---------- short name C

Student(ID, deptID, firstName, LastName, joinYear) ---------- short name S

Teaching(FacultyID, courseID, Year) ---------- short name T

Registration(studentID, courseID, Year, grade) ---------- short name R

(9) [5 points]: Report the student names (first and last) registered in the last offering of course ID = 'CS3431'

```
Select S.firstName, S.lastName
From R, S
Where R.studentID = S.ID
And R.courseID = 'CS3431'
And R.year = (Select  max(year)
              From R
              Where  courseID = 'CS3431');
```

(10) [5 points]: Report student IDs who have at least 3 A's (grades) in year 2010.

```
Select studentID
From R
Where year = 2010
And grade = 'A'
Group By studentID
Having count(*) >= 3;
```