Department of Computer Science,
New Mexico State University
C S 482/502 Database Management Systems I

Project

Deadline for the first phase: 11:59pm Oct. 14, 2016, Friday;
Deadline for the whole project: 11:59pm Nov. 11, 2016, Friday

Requirements

- You should form **a group with 1 or 2 people** to work on this project. Each group just needs to submit one copy of your program.
- You need to finish the tasks stated in the following two phases.

Scenario

Assume that you get the following relational schemas and the constraints for the real estate company in Las Cruces.

- User (UserID: varchar(16), Password: varchar(8), FirstName: varchar(32), LastName: varchar(32), Birthday: date, Email: varchar(32))
  - Constraint: FirstName should NOT be NULL.
  - Constraint: LastName should NOT be NULL.
- Tenant (UserID: varchar(16), TenantID: char(8))
  - Foreign key: UserID references User(UserID).
- Manager(UserID: varchar(16), EmployeeID: char(8), OfficeID: integer)
  - Foreign key: UserID references User(UserID).
  - Foreign key: OfficeID references Office(OfficeID).
  - Constraint: EmployeeID needs to be unique.
- Staff (UserID: varchar(16), EmployeeID: char(8), Title: varchar(16), OfficeID: integer)
  - Foreign key: UserID references User(UserID)
  - Foreign key: OfficeID references Office(OfficeID)
  - Constraint: Title should NOT be NULL.
  - Constraint: EmployeeID needs to be unique.
- UserPhoneNumber (UserID: varchar(16), PhoneNumber: integer)
  - Foreign key: UserID references User(UserID).
  - Constraint: PhoneNumber should NOT be negative.
- Office (OfficeID: integer, PhoneNumber: integer, StreetName: varchar(16), StreetNumber: integer, City: varchar(16), State: char(2), Zip: integer)
  - Constraint: OfficeID should NOT be negative.
  - Constraint: PhoneNumber should NOT be NULL.
- Property (PropertyID: integer, PropertyName: varchar(16), StreetName: varchar(16), StreetNumber: integer, City: varchar(16), State: char(2), Zip: integer, ManagerUID: varchar(16))

- Foreign key: ManagerUID references Manager(UserID).
- Constraint: PropertyID should NOT be negative.

- PropertyUnit (PropertyID: integer, ApartmentNumber: integer, Rent: integer, Availability: char(1), NumberOfBedRoom: integer, NumberOfBathRoom: integer)
  - Foreign key: PropertyID references Property(PropertyID)
  - Constraint: PropertyID should NOT be negative.
  - Constraint: ApartmentNumber should NOT be negative.
  - Constraint: Availability should only take values from {'Y, 'N'}.
- Payment (TransactionID: integer, PaymentMethod: varchar(16), Amount: integer, Date: date)
  - Constraint: TransactionID should NOT be negative.
  - Constraint: PaymentMethod should only take values from {"credit", "debit", "check"}.
- MaintainReq (JobID: integer, Date: date, RequestedJob: varchar(16), ManagerUID: varchar(16), PropertyID: integer, ApartmentNumber: integer)
  - Foreign key: ManagerUID references Manager(UserID).
  - Foreign key: (PropertyID , ApartmentNumber) references PropertyUnit(PropertyID, ApartmentNumber).
  - Constraint: JobID should NOT be negative.
  - Constraint: RequestedJob should only take values from {"carpet shampoo", "repairs", "replacement"}.
- StayIn (TenantUID: varchar(16), StartDate: date, PropertyID: integer, ApartmentNumber: integer, EndDate: date)
  - Foreign key: TenantUID references Tenant(UserID)
  - Foreign key: (PropertyID , ApartmentNumber) references PropertyUnit(PropertyID, ApartmentNumber)
  - Constraint: If a certain Tenant is still occupying a particular property, then the end date should be 01/01/1111.
- MakePayment (TenantUID: varchar(16), PropertyID: integer, ApartmentNumber: integer, TransactionID: integer)
  - Foreign key: TenantUID references Tenant(UserID)
  - Foreign key: (PropertyID, ApartmentNumber) references PropertyUnit(PropertyID, ApartmentNumber)
  - Foreign key: TransactionID references Payment(TransactionID)

For your reference, the data types provided by MySQL are used. (However, you can use other compatible data types).

- integer: http://dev.mysql.com/doc/refman/5.1/en/integer-types.html
- char, varchar: http://dev.mysql.com/doc/refman/5.1/en/char.html
- blob: http://dev.mysql.com/doc/refman/5.1/en/blob.html
- date, datetime: http://dev.mysql.com/doc/refman/5.1/en/datetime.html

Phase 1 (20 points)

- (15 points) Implement the database using MySQL. Write proper SQL statements to create the required tables in the database and to set all the required constraints (e.g., primary key, foreign keys, and value constraints). Put all the SQL statements to "CreateDB.sql".
- (5 points) Insert into each table 3-5 records. Put all the insertion SQL statements to "DataDB.sql".

Note: All your submitted SQL statements should run correctly (syntactically and semantically) at the department's MySQL server. If an SQL statement can not be run in the MySQL server, that SQL statement is graded as **ZERO** (i.e., no partial score).

Phase 2 (80 points)

Create a small web application using PHP to manage the above database.
Note: You have some flexibilities to design this web application. If you are not clear on the requirements, please discuss with the TAs.

Your program should contain a main file named **index.html** or **index.php**. From this main file, a user should be able to go to the following sub-systems. In your project, you are required to implement **three sub-systems**.

(i) (25 points) Tenant management sub-system. In this sub-system, a tenant should be able to

A. (5 points) **Request a new account**: if this tenant does not have an account yet. The requesting account interface should allow the user to enter his/her user id, email, password, first name, last name, and all the other information on the *User* relation. Note that the user id needs to be more than 8 characters and less than 16 characters. The system should make sure that the given user id is unique and is not shared by any other users of the system. The system should assign a unique 8-character tenant id to this tenant once his/her account has been successfully created.

B. (2 points) **Log in**: if a tenant already has an account, he/she should be able to log in using this interface. The log in should ask for the tenant's user id and password.

C. (9 points) **View:** After a tenant logs in, this tenant should be able to view
   - the information in the *User* relation along with his tenant id and the information in the *StayIn* relation.
   - the tenant should be able to see his/her payment history by accessing both *MakePayment* relation and *Payment* relation.
   - the tenant should be able to see the manager of their property unit, the manager's office, and all the contact details of that office (full address and phone number).

   Please make sure that your interface design is user-friendly.

D. (3 points) **Edit**: After a tenant logs in, the tenant should be able to edit his/her email and phone number(s). The tenant should also be able to reset his/her password by providing their user id. Please design your interface in a user-friendly manner.

E. (5 points) **Pay**: After a tenant logs in, they should be able to make payments towards their property unit. This part could simply let the user enter an amount in dollars, choose a payment method, and click pay. The property information should be automatically picked up from the relevant relation, the date should be the system date and your program should generate a unique transaction id for this payment transaction. This step should also add a relevant tenant-payment-property entry in the *MakePayment* relation.

F. (1 points) **Logout**: a tenant should be able to logout.

(ii) (30 points) Manager administration sub-system. In this sub-system, a manager should be able to

A. (5 points) **Create a new account** if this manager does not have an account yet. The interface should allow the manager to enter his/her user id email, password, first name, last name and every other information on the *User* relation. Note that the user id needs to be more than 8 characters and less than 16 characters. The system should make sure that the given user id is unique and is not shared by any other users of the system. The system should assign a unique 8-character employee id to this manager once his/her account has been successfully created.

B. (2 points) **Log in**: if a manager already has an account, he/she should be able to log in using his/her user id and password, and use the following functionality in this sub-system.

C. (6 points) **View**:

- The manager should be able to *view* his/her information which include first name, last name, email, and all the other information in the *User* relation along with the manager's employee id.
- The manager should also be able to see all the information of the office he works at including the details of all the staff that work in that office (note that the manager should NOT be able to see the user id and password for any of the staff members). For simplicity, you could only display one phone number for a staff rather than all of their numbers.

Design your interface in a user-friendly manner.

D. (3 points) **Edit:** the manager should be able to *reset* his/her password by providing their user id and employee id, and *edit* his/her email and phone number(s).
Design your interface in a user-friendly manner.

E. (5 points) **View, add, update, and delete maintenance requests**: a manager should be able to see all the maintenance requests initiated for a particular property unit, add a maintenance request, update a maintenance request, or delete an existing maintenance request.
When deleting a maintenance request, the request can only be deleted if a manager correctly supplies his/her user id, employee id, and password.

F. (8 points) **Assigns property units**: once a tenant has created their account, the manager should be able to be able to see all such tenants who doesn't have property unit assigned to them (this can be easily done by performing a simple search on both *StayIn* and *Tenants* relations) and then assign an empty property unit to the tenant. The manager assigning a property to a tenant triggers the following events: (1) The *Availability* of the property unit changes to *available*, (2) a relevant tenant-property entry is added to the *StayIn* relation. The start date for the *StayIn* relation should be automatically picked up from the system date.

G. (1 point) **Logout**: a manager should be able to logout.

(iii) (25 points) Staff administration sub-system. In this sub-system, a staff should be able to

A. (5 points) **Create a new account** if this staff does not have an account yet. The interface should allow the staff to enter his/her user id, email, password, first name, last name and every other information on the *User* relation. Note that the user id needs to be more than 8 characters and less than 16 characters. The system should make sure that the given user id is unique and is not shared by any other users of the system. The system should assign a unique 8-character employee id to this staff once his/her account has been successfully created.

B. (2 points) **Log in**: if a staff already has an account, he/she should be able to log in into this sub-system by providing his/her user id and password.

C. (5 Points) **View**: After A staff logs in,

- this staff should be able to *view* his/her information which include their first name, last name, email, and all the other information in the *User* relation along with their employee id.
- The staff should also be able to see all the information of the office he/she works at including the details of the manager who works in this office (note that the staff should NOT be able to see the user id, employee id, and password for the manager). For simplicity, you could only display one phone number for the manager rather than all of their numbers.

Design your interface in a user-friendly manner.

D. (3 points) **Edit**: After a staff logs in, this staff should be able to *reset* his/her password by providing their user id and employee id. This staff should also be able to *edit* his/her email and phone number(s).
Design your interface in a user-friendly manner.

E. (9 points) **Data Entry - adding and updating properties, property units, and offices**: After a staff logs in, this staff should be able to add and update all entries of a property, an office, and/or a property unit.

F. (1 point) **Logout**: a staff should be able to logout.

Handing in & Grading:

- You have to hand in your assignment electronically (through Canvas). Printed copies are not accepted. Combine and compress all the source files (sql, php, and other related files) to a .zip file with name proj-(your banner ids). See course syllabus for policies on late submission and plagiarism.
- Demonstration. You should demonstrate your project to your TA in **one week after the due date of this project** (inclusive).