CS1555/ CS2055 – Database Management Systems (Fall 2018)
Dept. of Computer Science, University of Pittsburgh

Project: MyAuction

Release:	Nov. 1, 2018
Team Declaration Due:	Tuesday, Nov. 6, 2018 @ 8:00 PM
Phase 1 Due:	Monday, Nov. 19, 2018 @ 8:00 PM
Phase 2 Due:	Monday, Dec. 3, 2018 @ 8:00 PM
Phase 3 Due:	Saturday, Dec. 8, 2018 @ 8:00 PM

Goal

The goal of this project is to design and implement an electronic auctioning system, similar to "ebay.com". The core of such a system is a database system. You are expected to do your project using Oracle, Java and JDBC in groups of three.

Our system, called "MyAuction", will facilitate auctioning by allowing registered users to post sale announcements, and submit bids. The system needs to keep track of information of registered users, products that have been put for auction, bidding history and selling history of each product on auction.

Some of the intended uses of the database system include:

- For each registered user, record his/her name, address, email, a unique login-name and password.
- For each product, put for auction, record its name, an (optional) description and one or several categories that it belongs to: 'books-and-records', 'software', 'automobiles', 'appliances' . You need to make sure the category given matches one of the available categories. Each product should have a unique auction-id.
- Keep track of information about a product for auction such as who is selling it, the minimum acceptable price, auction starting date and its status - i.e., one of the following values : 'under auction', 'sold', 'withdrawn', 'closed'.
- Keep track of every bid made by registered users, such as the bidder's name, the date when the bid was made, and the amount of the bid.
- If a product was sold successfully, we want to know who bought the product with what bidding price, and when it was sold.
- For each product category, record its (unique) name. We want to organize the categories into a hierarchy structure such that one category can contain 0 or more subcategories.

Database Description

We will use the database whose schema is described below. In addition, you must maintain our system time (not the real system time) in one of your tables. The reason to make our system time different from the real system time is to make it easy to generate scenarios to debug and test your project. All timing is based on our system time (i.e., the ourSysDATE table).

You need to create scripts to create tables in your Oracle server account. You are required to define all of the structural and semantic integrity constraints and their modes of evaluation. **You mush follow the table names and attribute names strictly.**

- ourSysDATE (c_date)
  PK (c_date)
  Datatype
  - c_date: date
- Customer(login, password, name, address, email)
  PK (login)
  Datatype:
  - login varchar2(10)
  - password varchar2(10)
  - name varchar2(20)
  - address varchar2(30)
  - email varchar2(20)
- Administrator(login, password, name, address, email)
  PK (login)
  Datatype:
  - login varchar2(10)
  - password varchar2(10)
  - name varchar2(20)
  - address varchar2(30)
  - email varchar2(20)
- Product(auction_id, name, description, seller, start_date, min_price, number_of_days, status, buyer, sell_date, amount)
  PK (auction-id)
  FK (seller) → Customer(login)
  FK (buyer) → Customer(login)
  Datatype:
  - auction_id int
  - name varchar2(20)
  - description varchar2(30)
  - seller varchar2(10)
  - start_date date
  - min_price int
  - number_of_days int
  - status varchar2(15) (not null)
  - buyer varchar2(10)
  - sell_date date
  - amount int
- Bidlog(bidsn, auction_id, bidder, bid_time, amount)
  PK (bidsn)

FK (auction_id) → Product(auction_id)
FK (bidder) → Customer(login)
Datatype:

- bidsn int
- auction_id int
- bidder varchar2(10)
- bid_time date
- amount int

- Category(name, parent_category)
  PK (name)
  FK (parent_category) → Category(name)
  Datatype:
  - name varchar2(20)
  - parent_category varchar2(20)
- BelongsTo(auction_id, category)
  PK (auction_id, category)
  FK (auction_id) → Product(auction_id)
  FK (category) → Category(name)
  Datatype
  - auction_id int
  - category varchar2(20)

Once you have created a schema and integrity constraints for storing all of this information, you should generate sample data to insert into your tables. Generate the data to represent at least 1 ourSysDATE, 3 Customer, 1 Administrator, 5 Product, 10 Bidlog, 5 Category, 5 BelongsTo. You can also use these inserts to test the integrity constraints and triggers (described in the next section).

Interface and Functions

You are expected to develop your project in SQL and Java. For the Java part, use JDBC to interact with the Oracle server. You can develop your project on either Unixs, Class3 or Windows environments, but we will only test your code on Class3 machine. So even though your code works in a windows environment, you should make sure it works on Class3 before you submit your project.

You need to design the **SQL transactions** appropriately and when necessary, use the concurrency control mechanisms supported by Oracle (e.g., isolation level, locking modes) to make sure that inconsistent state will not occur.

For this project, you need to design two easy-to-use interfaces for two different user groups: **administrators** and **customers**. A simple text menu with different options is sufficient enough for this project. You may design nice graphic interfaces, but it is not required and it carries **no bonus points**. A good design may be a two level menu such that the higher level menu provides options for administrators login and normal user login. The lower level menu provides different options for different users, depending on whether the user is an administrator or a customer.

Each menu contains a set of tasks. The tasks are described as follows:

1. Customer interface

   (a) Browsing products
   Customers are allowed to look at the list of products still under auction in a category of their choice; they can also ask for products to be sorted on the highest bid amount, or alphabetically by product name. To provide the list of categories, you should start by providing all categories in a root position of the category hierarchy so that customers can refine their choices step by step. When listing products, you should list all key attributes, e.g, auction id, name, description (if through search as mentioned in task (b)), highest bid amount (if through order by highest bid).

   (b) Searching for product by text
   Customers can specify up to two keywords; our system has to return all products (despite their status) whose descriptions contain ALL these keywords.

   (c) Putting products for auction
   Customers can put products for auction. For each product, information such as product name, (optional) description, categories it belongs to and number of days for auction need to be provided. Your system needs to return an unique auction_id, which in general should be equal to the largest previous auction_id plus 1. Your system should also use our system current date/time to set the starting date. Note that the categories provided by customers must be smallest subcategories (i.e., leaf node in the category hierarchy). You are asked to create a stored procedure called **proc_putProduct** and use that stored procedure to implement this task in your system. The procedure takes as inputs all the valid information that the user provides, infers other information (i.e., auction_id and starting date), and records all information to the database.

   (d) Bidding on products
   A customer can bid on a product by providing a specific auction_id and an amount before the auction for the product closes. There is a constraint that the amount of the new bid on a product must be higher than the currently highest bid on that product (recorded on the *Amount* attribute of the *Product* table). In addition to recording information about the valid bid to the database, you need to do the followings:

   - Create a trigger called **trig_bidTimeUpdate** that, when a new bid is inserted, advances the system time by 5 seconds to simulate the time consumed on this bid.
   - Create a trigger called **trig_updateHighBid** that updates the *Amount* attribute in the Product table for the product that is bidden on. The new value for this attribute is the amount specified in the new bid.
   - Resolve concurrency conflicts: Multiple concurrent transactions might bid on the same product, which might lead to inconsistency. For example, consider 2 transactions T1 and T2 trying to bid on a product P with amounts of 250 and 200, respectively. Assume that currently the highest bid on P is 100, so both transactions are valid to proceed. T2 then might be able to insert to Bidlog first, and hence update the amount of P to 250. T1 then, not being aware of the change in the amount, inserts a tuple to Bidlog and updates the amount to 200, which is not true since the highest bid on P now is 250. You need to design the transaction appropriately and when necessary, use the the concurrency control mechanisms supported

by Oracle (e.g., isolation level, locking modes) to make sure that such inconsistent state will not occurs.

(e) Selling products
This function is provided only to the customer who is the seller of the product. For simplicity, here you should manually terminate the auction for the product. Your system should then display the **second** highest bid, and let the seller choose either to sell the product or withdraw the auction. Update the product status according to the user's choice. If the seller decides to sell the product, the Amount attribute of the product should be set to the selling price (i.e., the second highest bid). If there is only one bid on the product, the final price is the amount of that only bid.

(f) Suggestions
If a customer X asks, the system can provide suggestions on which products he/she should look at. The system looks at the bidding history of customer X; then it determines the "bidding friends" of X, that is, those other customers who bid on at least one of the products that X did; then, our system lists a union of the products that the "bidding friend" are currently bidding on. The products should be listed in decreasing "desirability" order: Desirability of a product is defined as the number of distinct "bidding friends" that have bid on this product.

2. Administrator interface

(a) New customer registration
Each new customer has to provide his/her name, address, email address, preferred login name and password. You should also specify whether the new user is an administrator. Your database should prevent two customers selecting the same login name.

(b) Update system date
The system date is set to a new date value with second accuracy (i.e., mm.dd/yyy hh:mm:ss). It can be assumed that the system date cannot be set "backwards" so you do not have to worry about handling transactions that happened in the past when the time is moved back.

(c) Product statistics
This function will generate a report of products to include information about product name, its status, current highest bid amount and the corresponding bidder's login name if it is not sold, highest bid amount and buyer's login name if it was sold. Your system should provide options for listing all products or products from a specified customer.

(d) Statistics
Create the following stored functions in Oracle:

- **func_productCount**: counts the number of products sold in the past x months for a specific categories c, where x and c are the function's inputs.
- **func_bidCount**: counts the number of bids a specific user u has placed in the past x months, where x and u are the function's inputs.

- **func_buyingAmount**: calculates the total dollar amount a specific user u has spent in the past x months, where x and u are the function's inputs.

Now using the functions created above when applicable, support the following statistics for the past x months:

i. the top k highest volume categories (highest count of products sold), here we only care categories that do not contain any other subcategories (i.e, leaf nodes in the category hierarchy).

ii. the top k highest volume categories (highest count of products sold), we only care categories that do not belong to any other category (root nodes in the category hierarchy).

iii. the top k most active bidders (highest count of bids placed)

iv. the top k most active buyers (highest total dollar amount spent)

(k and x should be input parameters that can be specified by users.)

3. Additional Functional Requirements

   (a) You must verify the login name and password of all users at the beginning. By default, we assume an pre-exist administrator with login name "admin" and password "root". This info can be inserted into your table after you create database.

   (b) Create a trigger called **trig_closeAuctions** that executes when the system time is updated. This trigger should check all the products in the system and close the auctions (i.e., change the status to 'close' if it is 'under auction') of all products whose sell-date falls before the new system time.

Project Phases

Phase 1 The first phase covers the SQL part of the project. Specifically, you need to create the SQL files to create the tables, integrity constraints, triggers, functions, procedures, and any other database objects (e.g., views) that you think are needed.

Phase 2 The second phase covers the Java part of the project as well as transactions. Specifically, the text UI, the functions to do the tasks, display of meaningful error message when certain operation is not allowed or failed.

Phase 3 The third phase should contain a Java driver program to demonstrate the correctness of your code by calling all of the above functions and display corresponding results. It may prove quite handy to write this driver as you develop the functions as a way to test them.

You should also include a benchmark program to stress test the stability of your system by calling each function automatically for multiple time with reasonably large amount of data and display corresponding results each time a function is being called.

Now this may not seem like a lot for a third phase (especially since it is stated that you may want to do this work alongside Phase 2). The reasoning for this is to allow you to focus on incorporating feedback from the TA regarding Phase 1 and 2 into your project as part of Phase 3. This will be the primary focus of Phase 3.

Project Submission

Phase 1 Phase 1 should produce the following three files:
schema.sql the script to create the tables.
trigger.sql the script to create other database objects, e.g., trigger, functions, procedures, etc.
insert.sql the script to insert sample records into the database, that test triggers and IC etc.

Note that after the first phase submission, you should continue working on your project without waiting for our feedback. Furthermore, you should feel free to correct and enhance your SQL part with new views, functions, procedures etc.

Phase 2 Phase 2 should produce, in addition to Phase 1, the following files:
MyAuction.java the file containing your main class and all the functions.
Readme.txt the file with instructions of how to run your system.

Phase 3 Phase 3 should produce, in addition to Phase 2, the below files:
Driver.java the driver file to show correctness of your functions.
Benchmark.java the benchmark file to stress test your functions.

The project will be collected by the TA via the GitHub repository. Therefore, at the beginning of the project, you need to do two things:

1. Create one common **private** repository as a team, where all team members are contributing and use it to develop the project.
2. Give full permission of the project repository to your TA (GitHub ID: xzhangedu) and your instructor (GitHub ID: panos4pittcs).

To turn in your code, you must do three things by each deadline:

1. Make a commit to your project repository that represents what should be graded as your group's submission for that phase. The message for this commit should be "Phase X submission" where X is 1,2 or 3.
2. Push that commit to the GitHub repository that you have shared with the instructor and TA.
3. Send an email to cs1555-staff@cs.pitt.edu with the title "[CS1555] Project Phase X submission" that includes a link to your GitHub repository and Git commit ID for the commit that should be graded. Commit ID can be found on GitHub or as part of output of "gitlog" command.

Multiple submissions are allowed for each phase for each team. The last submission before the corresponding deadline will be graded. **NO late submission is allowed**.

Group Rules

- You are asked to form groups of three persons. You need to notify the instructor and the TA by emailing group information to cs1555-staff@cs.pitt.edu (Remember that you need to send the email from your pitt email address, otherwise the email will not go through). The email should include the names of the team members and should be CC-ed to all team members (otherwise the team will not be accepted).

- The deadline to declare teams is **8:00 PM, Tuesday, Nov. 6, 2018** . If no email was sent before this time, you will be assigned a team member by the instructor. Each group will be assigned a unique number which will be sent by email upon receiving the notification email.
- It is expected that all members of a group will be involved in all aspects of the project development and contribute equally. Division of labor in any way (e.g., with regard to phases, or SQL vs Java code) is not acceptable since each member of the group will be evaluated on all phases.

Grading

Feedback will be given based on Phase 1 and 2 submissions about incorrect implementation, missing functionality, etc. Final grade will be mainly based on Phase 3 submission.

The project will be graded on correctness (e.g., coping with violation of integrity constraints), robustness (e.g., coping with failed transactions) and readability. You will not be graded on efficient code with respect to speed although bad programming will certainly lead to incorrect programs. Programs that fail to compile or run or connect to the database server earn **zero** and **no partial points.**

Academic Honesty

The work in this assignment is to be done **independently** by each team. Discussions with other students or teams on the project should be limited to understanding the statement of the problem. Cheating in any way, including giving your work to someone else will result in an F for the course and a report to the appropriate University authority.

Enjoy your class project!