**IS 620 Advanced Database Projects**

Fall 2017

**Vacation Home Rental**

## Overview

You will form groups of two to five people in this project. It is recommended that each group consists of at least three people given the amount of work. Please read the whole document carefully before starting your project.

Your assignment is to design a database system for a XYZ vacation home rental company. You will design the database, insert some sample data, and implement a set of required features. Each feature will be implemented as one or more Oracle PL/SQL procedures. You do **NOT** need to write a graphic user interface.

**Assumptions:**

You can make the following assumptions in this project.

1. The system needs to store data about users (you can assume that each user needs to register with the system). Users could be hosts or guests (it is possible for a user to be both). Each user has user ID (for internal use), name, mailing address (including street, city, state, and zipcode), phone number, email, and a password (email is used as username). In addition, a host has an average rating (computed from reviews for the host) and a method of receiving payment (you can assume it is bank account number as varchar). A guest has an average rating and a profile (as varchar) stating some information about the guest (e.g., gender, family, job, hobby, etc.).
2. The system also stores information about listings. Each listing has a host, an address, zip code, city, state, type (house or apartment), maximal capacity (number of people can live there), number of bedrooms, number of beds, number of bathrooms, minimal number of nights of stay, check in time and check out time (you can use interval data type).
3. Each listing has a list of amenities such as Wifi, TV, washer and dryer, free parking, etc.
4. A host can enter several availability periods for a listing. Each period as a start date, and end date (if guest check out on end date the price still applies, but if the guest check in on end date the price for next period will be used), and a price per night if a guest stay in this period.   
   For example, suppose the availability periods are:  
   Period 1: 2016-9-7 to 2017-6-1, price per night is $80;  
   Period 2: 2017-6-1 to 2017-9-7, price per night is $110;  
   Period 3: 2017-9-7 to 2018-6-1, price per night is $90.

If a guest stay at this listing from 2017-9-1 (check in date) to 2017-9-10 (check out date). The guest stays 9 nights, 6 nights in period 2 and 3 nights in period 3. So the total charge of this stay is 6\*110+3\*90 = $930 plus a service fee (described next).

1. The rental company charges a 5% guest service fee for each booking and 3% host service fee. The guest service fee is added to a guest’s charge per booking. The host service fee is deducted from the payment (called payout) received by hosts.
2. Each guest can enter review for a host after staying at a listing owned by the host. Each host can enter review for a guest as well.
3. The database stores booking information, including booking ID, guest ID, listing ID, check in date, check out date, number of adults, number of children, payment method (as varchar), payment date (normally one day before check in), booking status (requested, approved or denied, paid, canceled), and payout status (whether host receives payment from the rental company). The guest first submits a booking request. The host can approve or deny it. If the request is approved, the guest makes payment to the rental company about one day before check in. The payment will be sent to host by the rental company (with guest and host service fee deducted) after some time.
4. A guest can cancel a booking before payment date. But not after that.
5. The system stores payout information (i.e., payment sent from rental company to host). The system usually computes total payment to a host once a month.
6. The system stores a message table, which includes user id, message date and message body.

**Features:**

**Features for account management:**

1. Register a user with the system. The customer needs to provide name, address, phone#, email, password, and user type (host or guest or both). The procedure should check whether the email already exists in user table. If so, please print a message saying the customer exists. Otherwise create an account with input values and return a new user ID.

2. Allow a user to login by providing email and password. Please check whether email exists and password matches. If not, please print a message to indicate the error. Otherwise print a message to indicate user has logged on. The procedure should return a value 1 for success login and 0 for unsuccessful log in.

3. Allow a user to read messages providing user id and a starting date. Print out messages for that user since that date.

**Features for listing management**

1. Allow a host to add a listing. The input includes host ID, address of the listing (street, city, state, zip), type of hosting, maximal capacity, number of bedrooms, beds, bathrooms, minimal stay, check in time and check out time, and a list of amenities (you can use varray data type). A new listing ID should be generated.
2. Allow a host to enter an availability period for a listing. The input includes listing ID, start date, end date, price per night.

**Booking**

1. \*\*\*(most difficult feature)  
   Look up available houses at a given city and state and in a given period. Input: city and state, check in date and check out date. Output: display house address, total price for listings that are available in the entire period and there is no booking (do not count those denied or canceled bookings) on the same house overlap with that period. The total price is computed by sum of per\_night\_price for every day in the period pluse a 5% service charge by the company.   
     
   Note that the given period (check in date to check out date) may overlap with multiple availability periods in listing\_availability table. For example, suppose the given period is 2017-9-1 to 2017-9-15, but there are two available period for a listing: one from 2007-6-1 to 2017-9-7 with cost per night 180 and the other from 2017-9-7 to 2018-6-1 with cost per night 100.

There are 6 nights in the first period with cost $180 per night and 8 in the second period with cost $100 per night.

So total cost = (180\*6 + 100\*8)\*1.05 = 1974.

However, if the two available period is from 2007-6-1 to 2017-9-7 and from 2017-9-10 to 2018-6-1, the given period is not completely covered (2017-9-8 to 2017-9-9), then this listing should not be in the result.

**Hint:** You can first fetch all listings that are the given city and state and do not have reservations intersect with given period. For each such listing, you can call a sub procedure to check whether the given period is completely covered by one or more available periods and compute the total price.

1. Booking request: a guest sends a booking request with listing ID, guest ID, checkin date, checkout date, number of adults, number of kids.

The feature automatically checks whether the listing is available (see feature 6 for how to check availability), and whether it meets maximal capacity and min stay requirement. Generate a message for the guest if any of these conditions are not satisfied. If all conditions are satisfied, insert a row into booking table with status = 'requested' and generate a message for the host if all conditions are satisfied. The message contains the request information along with average review rating of the guest.

1. Allow a host to approve or deny a booking request. Input: booking id, decision (approval or deny). Output: check if the booking exists. If not print out an error message. Otherwise update the booking status (set it to approved or denied) and insert a message to both host and guest indicating the request has been approved or denied.
2. Look up booking request for a host. Input: host id. Output: booking id, guest name, listing id, check in date, check out date, number of guests for listings owned by that host with status = requested.
3. Allow a guest to make payment. Input includes booking ID, payment method, and payment date. Check whether the booking exists and is approved and the payment date is at least one day before checkin date. If so, update payment status to paid and insert a message for the host and guest about this payment. Otherwise, print an error message.
4. Allow a guest to cancel a booking if not paid yet. Input includes booking ID. If the booking does not exist or the status is paid, print an error message and insert a message to guest saying why it cannot be canceled. Otherwise update status to canceled and insert a message for guest and host about the cancelation.
5. Allow the system to generate payout to host. Input includes host ID, payout date.

Find all bookings that are paid but with payout status be 0, update status to 1 (paid out) and compute total amount (sum of total for each booking/1.05 \* 0.97 to exclude service fees). Insert a payout record to payout table. Insert a message for the host.

(Note by Rashmi: According to above feature I think payout stauts initially should be zero ie., while inserting the values to table…then later when host checks for paymnmet he can update the staus to 1 if payment is made)

1. Allow a guest to enter a review for host and update average rating for the host as well.

Please check the guest indeed has a booking (paid) with a listing owned by the host.

1. Allow a host to enter a review to guest and update average rating for the guest as well.

Please check the guest indeed has a booking (paid) with a listing owned by the host.

**Analysis**

1. Report the following statistics: total number of users, hosts, guests, listings, bookings, top-k hosts with highest average ratings (k as input), top-k guests with highest average ratings, average length of stay per booking, average cost per booking.

**Deliverables:**

There will be 4 deliverables. D1 and D3 will be due before class starts on the due date. D2 and D4 are due midnight of the due date. Delayed submission will result in a penalty of 30% of your score (e.g., if your score for part 2 is 20 but you are late, your score will be 14). The final presentation is due at class time and no delay is allowed.

1. 10%. Due 9/19. Project Management Schedule.
   1. Include team members and a timeline showing each phase of your project with its tasks and time duration, for the entire effort.
   2. It is expected that every member should participate in all phases of the project. For example, every member should be involved in writing the code.
   3. Each task in the same phase may be assigned to different members. E.g., you can specify that features 1-5 are assigned to member X. Pay attention to difficulty of each feature as well (e.g., features with three \*\*\* are the most difficult and requires more work)
   4. Tasks should include system design, populating tables, writing code, testing code, running example queries, writing documents, preparing for presentation, etc. Smaller milestones shall be set for deliverable 3 and 4.
   5. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time is left for testing and integration).
2. 25%. Due 10/17. Design Document which includes the following:
   1. ER diagram of the database. You don’t have to follow exact notations of ER diagram, but need to show tables, columns, primary keys, and foreign key links.
   2. SQL statements to create database tables and to insert some sample data (at least 5 rows per table).
   3. Specification for each required feature. The specification should include a description of input parameters and output (usually screen outputs), and an example of how a user can use this feature (e.g., exec XXX(…) where XXX is the procedure name). You don’t need to implement any of these features at this point.
3. 30%. Due 12/19. Demonstration. Your work will be demonstrated to the class in real time, where you will present the design of your system and you will run a demo. You don’t need to submit anything.
4. 35% Due 12/19. The code should include:
   1. Drop table statements to drop tables if they exist (remember the order of drop should be inverse of create).
   2. Create table statements
   3. Insert statements
   4. Create procedure statements (with code for the procedures). Each feature can be implemented as one PL/SQL procedure (in the procedure you may call other procedures or functions). Please include some comments in your code explaining the major steps. You should use create or replace to avoid procedure name conflict.
   5. Demo script to show that all your features work correctly. The script shall include some examples to test different cases. E.g., for feature 1, one example for new user (email is not in database) and one example for existing user (using existing email). Please include:
   6. PL/SQL script to call the appropriate PL/SQL procedure for this feature. E.g., exec procedure-name(parameter values)
   7. Explanation of what should be the correct output. The output could be updated tables (you can have some select statement to show the updated tables), some print out, etc.
   8. Make sure you have tested your examples from beginning to end. Remember that database tables may have been changed in the process. So you may need to start with a clean database (i.e., right after you execute all the drop table, create table, and insert statements).

Grading Guidelines

What I look for while grading software code (deliverable 4):

1. Existence of code
2. Comments: Both descriptive and inline for every procedure/function
3. Software quality
   1. Whether it is correct (giving correct results).
   2. Whether it is complete and clear.
   3. Efficiency of code. You shall not use too many SQL statements, and you shall put as much work as possible in SQL. For example, if you can do a join, do not use two select statements and then do a join in your program.
   4. Whether it has considered all special cases such as whether a user has already registered in Feature 1.

Regarding the presentation of your project: Each student must participate in the project demonstration by presenting to the entire class some slides. You will be graded on:

1. Timeliness of presentation
2. Organization of the presentation
3. Delivery of your talk
4. Demo (running the code)

For the demo, you will be graded on the following items:

1. Existence of tables and data. You need to have at least 5 rows in each table.
2. The correctness of features. This can be shown by checking whether the screen output is correct and the database has been updated correctly.

Each member of the team shall contribute more or less equally. It is unfair for a few members to do most of the work while others do less. You will be asked to evaluate your teammate’s effort at the end of the project. The instructor will adjust the grade based on the evaluation. Normally if most of your teammates agree that you do not contribute at all or contribute too little (e.g., your group has 4 members and you contribute only 5%), you may lose up to 80% of your project grade. If your teammates agree that you contribute much more than anyone else (e.g., your group has 4 members and you contribute 40%), you may gain up to 20% of your project grade (but not exceeding 100% of project grade). A peer evaluation will be conducted at the end of the semester to determine the contribution of each team member.

Tips:

1. Be aware of you and your team members’ strengths and limitations. Some of you may have very little programming experiences, and some of you do. So each team shall have at least 2 people who are good at programming. Each team can discuss how to assign the workload fairly and at the same time matching the strengths of team members. Usually easier features can be assigned to members without much programming experience (they should take more responsibility on non programming part to ensure equal contribution). However, every member should write some code (2-3 features at minimal).
2. Start early. Do not wait until last month to start coding. Do not wait until one week before the demo to start putting things together. Past experiences show that more than 50% of time shall be devoted to testing and putting things together.
3. Learn how to debug SQL and PL/SQL code. You can insert screen output statements to check intermediate results. Oracle also returns error messages and error code. You can google the error messages and error code to find possible causes. You may also use Oracle SQL Developer which allows you to insert break points during debugging.
4. Use homework, in class exercises, and programs in slides as templates of your PL/SQL program. For example, if you need to write a cursor, find a cursor example and use it as a starting point.