Parking Lot Application Project Description
CSC 540 - Database Application Programming
(Team Project)

Final Due Date: Oct 21

Next steps:

1. Begin individual studying of project description and start developing your questions or issues.
2. Form teams (online forum has been set up) and discuss these issues and see if a discussion helps you resolve them or post on the project question forum. IBefore posting, please look to see if the question was already asked and answered. **Questions will be entertained up until Sep 19.**
3. If you have not found a team **by Sep 9,** let the TAs know to help place you in one.
4. Initial project plan which includes, the names and unityids of your team members **Team size is 4**. Initial ER model (not necessarily the complete and final one). One to two paragraphs describing your project plan - remaining tasks as you see them currently and possible task assignment. **Deadline for this is Sep 21**.
5. Sample data to be used for demo will be provided later (approx. one week before the due date).
6. Submission instructions are provided on the last page.

Application Description

The goal of this project is for you to develop a database application to help University Parking Services (UPS) manage the campus parking lots and its users. The UPS issues parking permits to employees, students and visitors and there are different eligibility constraints for parking permits in the different lots as well as time restrictions for eligibility. In addition to the permits, UPS issues tickets/citations for parking violations and collects fees for them. University students and employees all have a *univid* (integer) which is a unique identifier for identifying them and linking them to their vehicles as well as an attribute status that is either ‘S’ or ‘E’ or ‘A’ depending on whether a student or an employee or administrator (who is also an employee but works with UPS).

Parking lots are identified by a *unique name* and *an address* and a *zone’s* designation that determines who can park in the lot. Each zone has an identifier that is at most two characters long: A, B, C, D, AS, BS, CS, DS, V. A parking lot can have a designation that includes multiple zones. For example, a lot with zone designation $A / B / C$ is a lot that combines three zones. Zones fall under two categories: visitor and non-visitor (employee or student - we do not worry about the UPS employee parking) parking zones. Zones A, B, C D are for employees, zones with suffix ‘S’ - for students, V - for visitors.

A parking lot is made up of a set of spaces, each uniquely identified within the lot. In any lot that has V as part of its zone designation, a set of spaces within that lot are dedicated to the V zone. For example, say a lot is designated with A / B / V zones and there are 100 spaces in the lot, spaces 90 - 100 may be dedicated to visitors zone V (spaces 1 to 89 in this case would be considered both A and B spaces). Some spaces in a lot can have a dedicated type e.g. *electric vehicle* or *handicap*, otherwise they are just *regular* spaces.

Permits fall broadly into two categories *Visitor* and *NonVisitor*. Each permit has a *unique permit identifier* (8 character string length, begins with two digits for the year, one or two characters for the zone, then a combination of numbers and characters for the remainder), *zone identifier*, *a car license number*, *start date*, *expiration date*, *expiration time*, and a *space type designation* e.g. "handicap" (default is "regular"). Nonvisitor permits are associated with a *univid*. For a nonvisitor employee permit, one additional vehicle may be added to the permit (for a total of up to 2 vehicles). For a visitor permit, *the space number*, *start time* and *lot* is also included in the permit. The expiration date for employee permits is 11:59 pm, 12 months from start date and for students it is 11:59 pm, 4 months from start date. The expiration date and time for a visitor's permit is computed as a function of request time plus the duration requested (some more details of this is given below). Vehicle information associated with all permits include car manufacturer, model, year, color, license plate (unique). While employee and student permits are assigned by UPS employees, visitor's permits are obtained by users invoking a procedure with appropriate input GetVisitorPermit (Lot, duration, time beginning, type = default is standard, …). Below are some functions that need to be supported. For all functions, we give a suggestion of inputs. However, you are to think through and decide the final list of inputs and outputs for each function. Below, we include a general description and which user role can execute it.

- ExitLot(): takes a permit number as input makes the status of a visitor space available and computes any time overages for the reservation (in the case that a visitor overstayed their booked amount of time). This can just be recorded as a time overage for the permit.
- ChangeVehicleList(): this is to allow university users (employees) modify the vehicles on their permit. They may remove or add vehicles. However, note the students can only have one vehicle on a permit at a time, employees up to 2. The input to this should be the permit number and univid (use for verifying if the correct user is making the modification).

The following are some rules and restrictions concerning permits and their usage in lots.

1. To park in a lot, a user's permit must have as its zone, one of the zones designated for that lot. For example, a lot with zone designation A / B / C allows users with permits either A, B or C.
2. Employees can have only one permit which allows them to park in any of the regular spaces in a lot that includes their permit's zone as one of its designation (but in the dedicated visitor spaces region). Employees may also park in a zone designated for students (S), but students cannot park in employee zones except after work hours i.e. after 5pm.

3. A visitor permit allows a specific car in a specific visitor space and must include the appropriate special type designation to park in dedicated type visitor spaces like for electric cars. A visitor permit becomes invalid for any parking after permit expiration time.

Citations are issued to vehicles by UPS employees that violate parking regulations (the process of issuing is discussed below). A citation includes a unique citation *number*, *car license number*, *model*, *color*, *date*, *lot*, *time*, *violation category*, *appropriate fee* for the category and payment due date (which is usually 30days after citation date). A citation also has a status which is *paid* or *unpaid*. When it is initially issued, status is unpaid. The following are the categories and fees:

- Invalid Permit ($20)
- Expired Permit ($25)
- No Permit ($40)

IssueCitation () is a procedure that generates with appropriate information as described above. A citation should alert the user via a notification. No integration with mail or sms is required. However, notification should be achieved by inserting a tuple in a "notifications" table. The structure of such a table is at your discretion but should include just basic information including the contact information which for university entities it is their univid and for visitors we will assume they include a phone number when requesting a permit.
Anyone can pay their citation by invoking the PayCitation ( ) procedure. (For our purposes, all this citation will do is change the status of citation from unpaid to paid).

The administrators of the system (i.e. university employees in the UPS and are designated 'A') have a few functions that they can perform.

- AddLot (…): This function should take lot identifier and address of lot, number of spaces in lot, beginning space number, initial zone designation (single zone initially).
- AssignZoneToLot (…):
- AssignTypeToSpace (…….):
- AssignPermit (univid, type, zone, …….): assigns a permit to user with univid for parking zone given in input and type (default is used if no input is given). Things like expiration etc are computed based on the description given earlier.
- CheckVValidParking (CurrentTime, Date,Space#, Lot, License#): checks if car has valid permit in visitors lot.
- CheckNVValidParking(time, permit#): checks valid parking in nonvisitor parking lot.

User Interfaces assume that your system has a main entry screen where users can select an option of what role they want to play (admin, university user, visitor). Then, for each role, the list of functions that they can perform will be listed and a user can select which function they want to perform. University users should have a login step before their functions are displayed. (Visitors should not need to log in because they dont have univids. Their functions should be available once they choose that role). These user interfaces do not have to be graphical user

interfaces (but if you wish to do so that's fine). A menu driven list e.g. numbered list of options and users selecting the corresponding number for the option they want, should suffice.
For example, one the main screen you may present options like

"1. UPS Admin Role
2. Employee Role
3. Student Role
…….
Please enter the number for the menu option desired.
A similar strategy can be used for all screens.
Remember to have as one of the options of the main screens an option to select and execute the preassigned queries given.

Sample Queries

Queries on your database will help assess the quality of database design. However, it isn't possible to leave that implementation to only demo day. Therefore, you are being asked to implement the queries given below as part of your project ahead of the demo. Note that on the demo day, you will be given 2 - 3 queries not listed here to implement directly on your database.

1. Show the list of zones for each lot as tuple pairs (lot, zone).
2. Get permit information for a given employee with UnivID: 1006020
3. Get vehicle information for a particular UnivID: 1006003
4. Find an available space# for Visitor for an electric vehicle in a specific parking lot: Justice Lot
5. Find any cars that are currently in violation
6. How many employees have permits for parking zone D.

Reporting Queries

1. For each lot, generate the total number of citations given in all zones in the lot for a three month period (07/01/2020 - 09/30/2020).
2. For Justice Lot, generate the number of visitor permits in a date range: 08/12/2020 - 08/20/2020, grouped by permit type e.g. regular, electric, handicapped.
3. For each visitor's parking zone, show the total amount of revenue generated (including pending citation fines) for each day in August 2020.

What to turn in:

The final version of the project is to be turned in as a single zip file on the course Moodle page.

1) Final Report & Demo: **Oct 21, 2020 (midnight)**
(Find available time slots for demo HERE).

    1. Constraints: A description of constraints that were not implemented as part of table definitions and justification on how they were implemented in the final design. In particular, a separate subsection here should highlight constraints that couldn't be implemented in the database at all and had to be implemented in application code. Note that a key part of assessing your design is how well you used the DBMS to implement constraints v/s implementing in application code.
    2. Two SQL files:
        a. Triggers, tables, constraints, procedures
        b. Queries for populating the tables with the sample data
    3. Executable file (e.g. executable JAR file) and source Java Code.
    4. README.txt file that contains the names of the team members and explains with any additional instructions on how to compile and execute your code.
    5. Everything should be in a **single zip file** so that when we unzip it, we can read the README file, follow the directions, and run your project.

2) Peer review of each member of your team that should include your own evaluation of all members (score out of 100). **A separate link for submission is provided on Moodle.**

Grading:

1. Preliminary Report: **5 points**
2. Final Report:
    a. ER design: **15 points**
    b. Relational design, including constraints and indices (SQL definition file): **15 points**
    c. Functional dependencies **10 points**
    d. Application development (Executable code & SQL Query files) and Demo (including new queries): **40 points**
3. Discussion and justification of design choices, instructions for running demo etc.: **15 points**
4. Peer Review: % of average of peer review grades