CS2102: Database Systems

Project (25%)

August 31, 2020

The objective of this team project is for you to apply what you have learned in class to design and develop a web-based database application. The project is to be done in teams of five students.

1 Overview of Application

This section provides a brief specification of the database application. As the specification is not meant to be complete, each team has the freedom to decide on the application's data requirement as well as the application's functionalities/features to be implemented. Your database design and implementation for the application should demonstrate non-trivial usage of SQL and database features.

1.1 Pet Caring

You are to develop a database application for pet caring service (PCS). The PCS application allows pet owner to search for care takers for their pets for certain periods of time. An example of this is https://dogvacay.com. The application supports *at least* three kinds of users: Pet Owner, Care Taker, and PCS Administrator. Note that the Pet Owner and Care Taker may use the same account and when they are mentioned as one, we will use the term User.

Each User must have a profile. Additionally, we want the Pet to also have a profile. Each Pet are classified into categories (*e.g.*, cats, dogs, big dogs, lizards, *etc.*) to facilitate browsing. They may also have special requirements related to how they need to be taken care of (*e.g.*, daily walk, types of food, *etc.*).

Each Care Taker can advertise their availability. This should minimally include the days they are available, the kind of pets they can take care, their daily price for each kind of Pet, *etc.*. A Care Taker should not take care of pets they cannot care for but may take care of more than one pets at any given time.

Pet Owner can browse or search for Care Taker and bid for their services. The successful bidder could either be chosen by the Care Taker or automatically selected by the system based on some criteria. Both the Pet Owner and Care Taker should agree on how to transfer the Pet, which can be one of the following three (*or more, if you have other possible ideas*):

1. Pet Owner deliver.
2. Care Taker pick up.
3. Transfer through the physical building of PCS.

The cost of caring for a Pet is the number of days times the daily price stated by the Care Taker. Once selected by the Care Taker, the Pet Owner must pay for the amount upfront either by pre-

registered credit card or paying cash. At the end of the care period, the Pet Owner can post review on and give rating to the Care Taker. The review and rating will only be for the specific transaction. As such an Pet Owner may submit multiple review/rating for a Care Taker if the Care Taker has taken care of the Pet Owner's Pet multiple times, including for the same pet. The reviews will be available to all User.

Each Care Taker can be either a full-time or a part-time employee. Each full-time Care Taker must work for a minimum of 2×150 consecutive days a year (since it includes weekends, so there are plenty of leave days available). A full-time Care Taker are treated as available until they apply for leave. They cannot apply for leave if there is at least one Pet under their care. We assume there will be no emergency leave but you may cater for such scenario if you have time. Full-time Care Taker have a limit of up to 5 Pet at any one time. When bid by any Pet Owner, a full-time Care Taker will always accept the job immediately if possible

For each part-time Care Taker, they should be able to specify their availability for the current year and the next year. For instance, on 1 January 2020, a part-time Care Taker can already specify their availability until 31 December 2021. At any single point in time, a part-time Care Taker cannot take care of more than 2 Pet *unless* they have a good rating (*e.g.*, 4 out of 5, or any such threshold you define) and they cannot have more than 5 Pet regardless of rating.

The base daily price for a full-time Care Taker is already specified by PCS Administrator for each pet type. This price increases with the rating of the Care Taker but will never be below the base price. The salary of full-time Care Taker depends on how many Pet are taken care of in a given month for how many days. We call this value pet-day. A full-time Care Taker will receive a salary of $3000 per month for up to 60 pet-day. For any excess pet-day, they will receive 80% of their price as bonus. For part-time Care Taker, the PCS will take 25% of their price as payment (*i.e.*, the Care Taker receives 75% of their stated price).

2 Application Requirements

Your PCS application should not be limited by the functionalities described in the previous section. You are free to introduce additional functionalities and *realistic* data constraints to make your application interesting and non-trivial. You are also free to use any of the database's features and other SQL constructs beyond what are covered in class. Your application must contain at least 3 appropriate applications of triggers. Additionally, your application must provide at least the following functionalities:

1. Support the creation/deletion/update of data for the different users (Pet Owner, Care Taker, and PCS Administrator).
2. Support data access for the different users (*e.g.*, Pet Owner can view reviews of Care Taker, their own past orders, *etc.*; Care Taker can see their review, their past jobs, their salary, *etc.*).
3. Support the browsing/searching of Care Taker by Pet Owner.
4. Support the browsing of summary information for PCS Administrator. For instance, it can be one of the following:
   (a) Total number of Pet taken care of in the month.
   (b) Total salary to be paid to all Care Taker for the given month.
   (c) The month with the highest number of jobs.
   (d) The *underperforming*[1] full-time Care Taker.
   (e) *etc.*
5. Support the browsing of summary information for Care Taker. For instance, it can be one of the following:

[1] You may give your definition.

(a) Total number of pet-day this month.

(b) Their expected salary for this month.

(c) *etc.*

6. Support the browsing of summary information for Pet Owner. For instance, it can be one of the following:

   (a) All Care Taker in their *area* (*if information is available*).

   (b) Other Pet Owner nearby.

   (c) Their Pet information.

   (d) *etc.*

For your final project demo, your application's database should be loaded with *reasonably* large tables. To generate data for your application, you may use online data generators (*e.g.*, https://mockaroo.com/ or https://www.generatedata.com). Alternatively, you may write your own program to generate mock data.

3 Software Tools

The architecture of your application follows the typical three-tier (clients, application server, database server) architecture of a Web-based database application shown in Figure 1 with Web browsers as the clients and a Web server as the application server.

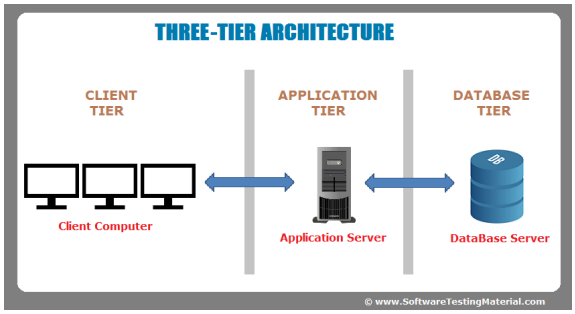

Figure 1: Three-Tier Web-based Application Architecture

You may use any SQL server, frontend, and backend development stack of your choice but we recommend the following combinations:

1. PostgreSQL
2. NodeJS

Additionally, your project must not use any Object-Relational Mapping (ORM) framework. Note that you are required to host your project online which may serve as your portfolio. We recommend the use of Heroku (https://www.heroku.com/) if you are using the recommended combinations above. If you wish to use other stack, you need to find a suitable host on your own (*preferably a free one*).

4 Deadlines & Deliverables

There are four deliverables to be completed:

Week	Project Task
Week 3	Team registration
Week 6	Preliminary constraints, ER diagrams, with justification for any serial type
Week 12	Project report to Git
Week 13	Demo video to Git (*up to 10 mins video*)

The preliminary constraints and ER diagrams[2] (deadline end of **Week 6**) will be reviewed by end of week 8 with possible weaknesses/improvements highlighted. By that period, your private Git repo should be set up with TAs added into the team for grading purposes[3]. You may view the file `GitItRight.pptx` to guide you on how to use Git[4].

The final project deliverables will be at the end of **Week 12**, with extension given on a case-by-case basis. The project report (*up to a maximum of 20 pages in PDF format with at least 10-point font size*) should be uploaded to your Git repo and should include the following:

1. Names and student numbers of all team members and project team number (*on the first page*).
2. A listing of the project responsibilities of each team member.
3. A description of your application's data requirements and functionalities. Highlight any interesting/non-trivial aspects of your application's functionalities/implementation. List down all the application's data constraints.
4. The ER model of your application. If your ER model is too large (*i.e.*, spanning more than a page), you may want to include a single-page simplified ER model (*with non-key attributes omitted*) before presenting the detailed ER model. Provide justifications for any non-trivial design decisions in your ER model. List down all the application's constraints that are not captured by your ER model.
5. The relational schema derived from your ER data model (*i.e.*, show the SQL `CREATE TABLE` statements and possibly `ALTER TABLE` if any for all your database tables). List down all the application's constraints that are not enforced by your relational schema (*e.g.*, the constraints that are enforced using triggers, or cannot be enforced at all).
6. Discussion on whether your database is in 3NF or BCNF. Provide justifications for tables that are not in 3NF or BCNF.
7. Present the details of three non-trivial/interesting triggers used in your application by providing an English description of the constraints enforced by each trigger and showing the code of the trigger implementation.
8. Show the SQL code of three of the most complex queries implemented in your application. Provide an English description of each of these queries.
9. Specification of the software tools/frameworks used in your project.
10. Two or three representative screenshots of your application in action.
11. A summary of any difficulties encountered and lessons learned from the project.

[2] Note that non-trivial constraints are removed from this requirements as *triggers* are likely to only be discussed in Week 7.

[3] We can be removed after the end of the semester.

[4] Courtesy of Akshay Narayan from CS2103/T.

5 Evaluation Criteria

The maximum score for the project is 25 marks with the following breakdown. A simplified grading rubric is given to show the expected minimum work with + being full mark and ∘ being minimum expected work achieving almost full mark. In some cases, we add − to show what you should avoid.

1. Preliminary constraints, ER diagrams, with justification for any serial type **5 marks**
   (a) Preliminary constraints *2 marks*
      +⟩ Clear and logical description of constraints that match real-world constraints.
      ∘⟩ Some unclear or unrealistic constraints.
   (b) Preliminary ER diagram *2 marks*
      +⟩ Realistic model with reasonable complexity in terms of number entity/relationship sets (*i.e.*, minimum of 15) with at least 1 weak entity sets and using appropriate constructs; satisfy most constraints given above that can be satisfied; serial types are well-justified.
      ∘⟩ Does not met the minimum number of entity/relationship sets or did not have weak entity sets; many constraints are not satisfied that could have been satisfied; serial types are not well-justified.
   (c) System setup *1 mark*
      +⟩ Systems are set up on time.
2. Final project report **15 marks**
   (a) ER data model (*i.e.*, *updated ER diagram from feedback*) *2 marks*
      +⟩ Realistic model with reasonable complexity in terms of number entity/relationship sets (*i.e.*, minimum of 15) with at least 1 weak entity sets and using appropriate constructs; satisfy most constraints given above that can be satisfied; serial types are well-justified.
      ∘⟩ Does not met the minimum number of entity/relationship sets or did not have weak entity sets; many constraints are not satisfied that could have been satisfied; serial types are not well-justified.
   (b) Relational schema *3 marks*
      +⟩ Relational mapping is consistent with ER model; appropriate use of table/column constraints to enforce constraints whenever possible; some discussion of the normal form of its schema.
      ∘⟩ Some inconsistencies between ER model and relational mapping; some mapping decision is sub-optimal (*e.g.*, could have been enforced by a table constraints if two or more tables had been merged into a single table).
   (c) Interesting queries *3 marks*

      up to 1 mark per query (*only for top three most interesting queries*). If you give more than three, we assume the top three are given by ordering from most interesting to least interesting.
      +⟩ Demonstrate the application of advanced SQL constructs that provides insight into application that could improve business decision:
         – Subqueries
         – Group by aggregations (*possibly with having clause*)
         – Common table expressions (*but not more than 2*)
         – Transactions
         – Stored procedures
      ∘⟩ Demonstrate the application of advanced SQL constructs without insight into application.
      −⟩ SQL queries can be *rewritten* easily into one without the advanced constructs above.

(d) Triggers for complex constraints *3 marks*

up to 1 mark per triggers (*only for top three most interesting triggers*). If you give more than three, we assume the top three are given by ordering from most interesting to least interesting.

+⟩ Correct triggers enforcing correct constraints that cannot be enforced with table/-column constraints.

−⟩ Constraints could have been enforced using table/column constraints.

(e) User interface *2 marks*

+⟩ Interface are well thought out and easy to use.

○⟩ Data are displayed only using html tables (*e.g.*, `<table></table>`); all functionalities are connected to home page *without manually typing specific address* but can be several clicks away; bland design.

(f) Report Write-Up *2 marks*

+⟩ Report is well organized and well written; meets all 11 requirements.

○⟩ Report is poorly organized or poorly written; many aspects of the project are not documented.

3. Project demo **5 marks**

(a) Demo video *3 marks*

+⟩ Demo presentation is organized and coherent; functionalities worked as advertised (*and replicable in web site*); application did not crash. Database has at least 1000 records of both Care Taker and Pet Owner combined.

○⟩ Some reported functionalities are not working or not fully implemented or not replicable in web site.

(b) Q&A *2 marks*

+⟩ Demo is so well done that there is no need to ask.

+⟩ Team is able to answer clarification questions clearly when asked.

○⟩ Some answers are not clear when asked.

Additionally, there is a bonus mark of **1 mark** for projects in top 10. The team will receive the bonus mark if their total project marks is *under 25 marks*. Evaluation criteria includes:

1. Realistic ER data model with corresponding relational schema.
2. Applicable interesting queries that captures important business decision.
3. Highly polished user interface.
4. Well designed video (*e.g.*, think of it like some mini promo video).