CPSC 304 Project Cover Page

Milestone #: 4

Date: November 28, 2024

Group Number: 72

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Edward Liu	55997308	e9s4m	edwardtliu8@gmail.com
Kobe Shen	13079694	b0j3y	Shenkobe.111@gmail.com
Yang Yu	45834330	n5p8x	yuyang2003m@163.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Description:

- **Description:**
  Our project aims to model the whole pet adoption and post-adoption animal care ecosystem, allowing users to browse and adopt pets hosted at the shelter, allowing animal specialists like veterinarians to view, store and update a pet's medical documentation and for trainers to work with shelters and train their respective pets.

- **Accomplishment:**
  This project creates a Pet Shelter database with SQL script to drop, recreate, and reload multiple tables including Pet, Trainer, Shelter, etc. and inserts sufficient data. The program features a user-friendly GUI with clear user notifications. It allows users to insert documentation data, update pet information, delete trainers, and perform complex queries as listed below. Basic data sanitization and error handling are implemented to ensure security.

- **Final schema:**
  The final schema has generally remained **unchanged.** The Entities, Relationships, Keys, Attributes, etc. remain the same from the previous turned-in work. ON DELETE CASCADE constraint has been added to some of the foreign key relationships to ensure integrity and expected behavior during deletions.

List of SQL queries:

* Location in Code refers to the **start** of each query
* Only include SQL queries here, no parameter bindings or execution options

Query	Location in Code	SQL Queries And Description for 2.1.7 - 2.1.10
2.1.1 INSERT	appService.js, line 244	INSERT INTO Documentation (pid, veterinarianContact, id, ddescription, ddate) VALUES (:pid, :vetcon, :id, :ddesc, TO_DATE (:ddate, 'YYYY-MM-DD'));

2.1.2 UPDATE	appService.js, line 274	UPDATE Pet1
		SET pname = :newValue
		WHERE pname = :oldValue AND pid = :petID;

2.1.3 DELETE	appService.js, line 561	DELETE FROM Trainer1
		WHERE contact = :trainerContact;

2.1.4 Selection	appService.js, line 548	SELECT *
		FROM Shelter

		${queryString}\` (* WHERE is included in the query string in case of NO selection condition is specified.)

2.1.5 Projection	appService.js, line 258	SELECT ${selectedColumns}
		FROM Dog

2.1.6 Join	appService.js, line 408	SELECT *
		FROM POwner o, AdoptionApplication a
		WHERE o.oaddress = a.ownerAddress
		AND ${query}

2.1.7 Aggregation with GROUP BY	appService.js, line 574	SELECT P7.species, AVG(P3.age)
		FROM Pet7 P7, Pet3 P3
		WHERE P3.pid = P7.pid
		GROUP BY P7.species;
		
		This query finds the average age of pets group by species.

2.1.8 Aggregation with HAVING	appService.js, line 424	SELECT Pet7.species, AVG(Pet2.age) FROM Pet1, Pet2, Pet3, Pet4, Pet5, Pet6, Pet7, Pet8, Pet9, Pet10, Pet11 WHERE Pet1.pid = Pet3.pid AND Pet1.pid = Pet4.pid AND Pet1.pid = Pet5.pid AND Pet1.pid = Pet6.pid AND Pet1.pid = Pet7.pid AND Pet1.pid = Pet9.pid AND Pet1.pid = Pet10.pid AND Pet2.species = Pet7.species AND Pet2.age = Pet3.age AND Pet8.species = Pet7.species AND Pet8.breed = Pet9.breed AND Pet11.species = Pet7.species AND Pet11.dietaryRequirements = Pet2.dietaryRequirements AND Pet11.healthCondition = Pet4.healthCondition GROUP BY Pet7.species HAVING AVG(Pet2.age) > 5; This query finds the average age of each pet species which has average age greater than 5.

2.1.9 Nested aggregation with GROUP BY	appService.js, line 221	SELECT P1.pid, P1.age, P2.species
		FROM Pet3 P1
		JOIN Pet7 P2 ON P1.pid = P2.pid
		WHERE P1.age <= (SELECT AVG(P3.age)
		FROM Pet3 P3
		JOIN Pet7 P4 ON P3.pid = P4.pid
		WHERE P4.species = P2.species
		GROUP BY P4.species
		);
		
		This query finds pets whose age is less than or equal to the average age of all pets within the same species.

2.1.10 Division	appService.js, line 458	SELECT Shelter.saddress
		FROM Shelter
		WHERE NOT EXISTS (
		(SELECT Supplier1.saddress
		FROM Supplier1)
		MINUS
		(SELECT PurchasesFrom.supplierAddress
		FROM PurchasesFrom
		WHERE shelterAddress = Shelter.saddress);
		
		This query finds the addresses of shelters that have purchased from all suppliers.