**Persistent Storage**

The persistent data structure used in the cuACS system is a sqlite3 database. Using a database connected to QtCreator allows the program to efficiently and swiftly save data without the hassle of having to parse a plain-text file. Using a database also significantly increases the scalability of the program, allowing for many other animals and clients to be stored. When the program is loaded, a check for a database is performed, searching the operator’s host desktop directory for an existing database to load from. If a database does not exist, one is created and placed on the desktop, with the name *“cuACS.db”*. Providing the database on the desktop allows for simple access, if needed, for the operator or in-house IT staff; it also provides the operator with assurance that a database has been created for them, in a spot that they are familiar with. After the database is created, 6 tables are created– clientStorage, dogStorage, catStorage, lizardStorage, birdStorage and rabbitStorage, with corresponding schemas. (*See section* ***Table Schema*** *to view each table’s associated schema.)* A check for these 6 tables is also performed if an existing database is present, with the correct schema. After the database and tables have been established, the tables are populated with clients (25) and animals (27).

Each client/animal profile is represented in the database as a row, in the appropriate table – i.e. dogs are stored in the table titled ‘dogStorage’, clients in clientStorage, etc. The columns for each of the tables represent a client/animal’s attributes, such as for animals: name, breed, history, hypoallergenic, etc.; and for clients: name, address, wantsBird, has children, etc. Each of the attributes are stored in the database tables as a sqlite INTEGER type or sqlite TEXT type. For the case of an animal’s breed, the associated vector of breeds is converted into the a string in the format: [ (breed1)(breed2)(…)(breedn)], for the n breeds which are associated with an animal, this newly formed string is saved as a sqlite TEXT type. Booleans are stored as integers.

As the tables are read, and the data is loaded into the cuACS system, each table is read, row by row. When a row is being read, an object is created, and the corresponding column value is assigned to the objects attribute. For example, when lizardStorage is being read, the lizard object’s “space” attribute – denoting how much space the lizard needs to survive – is set by the value stored in the database for the column titled “space.” This is done in the exact same way for all animals, and all clients. When a row is finished being read, the object is complete. Next, the newly created object is pushed into the internal Animal or Client list. Using an internal list speeds up processing time and prevents having to query the database for every time a profile is viewed.

When adding a new client or animal to the persistent storage, all of the appropriate text-fields, radio-buttons and check boxes are read, and an animal or client object is created. When created, the object is added to the appropriate internal list (client or storage), as well as sent to the database class to be added to the cuACS database. Here, the object is deconstructed into its attributes for the object, and are stored into the database, by column, based on the profile they are (i.e. client, dog, bird, etc.). A unique identifying number is created for each client or animal in the system, called *idNum*. The idNum is used for distinguishing animals and clients from each other – this is used as the primary key in the database. A primary key is the key that is unique for each record (row) in the database. Assigning each animal and client a unique idNum prevents the data from being over written when a new animal or client is added to the appropriate table.

Adding a new client or animal to the persistent storage creates the appropriate object, (with a new, unique idNum) and send it’s to the databaseStorage class, to add it to the database. Here, we deconstruct the object into its attributes, storing each of them inside the table with columns representing the attributes. When it comes to editing a profile for a client or animal, we refer to the profile by its unique idNum identifier, to ensure that we only edit the profile desired and not any others – this is ensured because the idNum is used as the primary key, and cannot be replicated, for an entry into the database. An idNum is composed of 2 parts, the profile code, followed by an identifying number. Profile codes are as follows: ﻿Client = 1, Dog = 2, Cat = 3, Bird = 4, Lizard = 5, Rabbit = 6, and is the first digit for the idNum. The identifying number is calculated by determining how many clients or objects currently exist in the system, and for a newly created profile is incremented by one. By default, the identifying numbers start at 100. As an example, the first animal in the system would have idNum = *profilecode+*100, adding another animal would create the idNum *profileCode*+101, and so on. An example of the clients’ idNum is as follows: if there are 12 clients in the system and another one is being added, the last client idNum would have been 1112, the resulting idNum of the new client will be 1113. Creating and utilizing an idNum in this way allows assurance that no 2 keys are the same, maintaining uniqueness between all animals and clients.

Having these unique idNum keys, allows for easy edits of clients or animals to occur. Simply querying the database table for the idNum of an edited animal or client, allows the row to be easily updated. In order to ensure that all user changes are updated, the all profile data is pushed to the database table and stored for the given idNum.

**Table Schemas**

Below are the schema for each table in the sqlite3 database for cuACS, as printed out using the *.schema <table\_name>* command in sqlite3 from a bash shell.

**Table**: clientStorage

CREATE TABLE clientStorage(idNum INTEGER PRIMARY KEY, fName TEXT, lName TEXT, address TEXT, phone TEXT, email TEXT, city TEXT, prov TEXT, dwelling INTEGER, location INTEGER, workSchedule INTEGER, activity INTEGER, hasChildren INTEGER, hasAnimals INTEGER, travels INTEGER, children INTEGER, goodWAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, wantsDog INTEGER, hasDogAllergies INTEGER, dogBreeds TEXT, dogAge INTEGER, dogSize INTEGER, dogGender INTEGER, followsCommandsDog INTEGER, houseTrained INTEGER, wantsCat, hasCatAllergies INTEGER, catBreeds TEXT, catAge INTEGER, catSize INTEGER, catGender INTEGER, isCurious INTEGER, followCommandsCat INTEGER, doesntShed INTEGER, wantsBird INTEGER, hasBirdAllergies INTEGER, birdBreeds TEXT, birdAge INTEGER, birdSize INTEGER, birdGender INTEGER, isQuietBird INTEGER, isSocialBird INTEGER, birdColour TEXT, wantsLizard INTEGER, hasLizardAllergies INTEGER, lizardBreeds TEXT, lizardAge INTEGER, lizardSize INTEGER, lizardGender INTEGER, easyToFeed INTEGER, simpleLiving INTEGER, lizardColour TEXT, wantsRabbit INTEGER, hasRabbitAllergies INTEGER, rabbitBreeds INTEGER, rabbitAge INTEGER, rabbitSize INTEGER, rabbitGender INTEGER, isSocialRabbit INTEGER, needsGrooming INTEGER, rabbitColour TEXT, dogFur INTEGER ,catFur INTEGER, birdFur INTEGER, lizardFur INTEGER, rabbitFur INTEGER, quietness INTEGER, age INTEGER);

**Table**: dogStorage

CREATE TABLE dogStorage(idNum INTEGER PRIMARY KEY, breed TEXT, name TEXT, size INTEGER, age INTEGER, gender INTEGER, fur INTEGER, travels INTEGER, children INTEGER, goodWithAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, nocturnal INTEGER, hypo INTEGER, lifeStyle INTEGER, history INTEGER, barks INTEGER, training INTEGER, bathroomTrained INTEGER, goodBoy INTEGER, filepath TEXT);

**Table**: catStorage

CREATE TABLE catStorage(idNum INTEGER PRIMARY KEY, breed TEXT, name TEXT, size INTEGER, age INTEGER, gender INTEGER, fur INTEGER, travels INTEGER, children INTEGER, goodWithAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, nocturnal INTEGER, hypo INTEGER, lifeStyle INTEGER, history INTEGER, curiosity INTEGER, trained INTEGER, shedding INTEGER, filepath TEXT);

**Table**: birdStorage

CREATE TABLE birdStorage(idNum INTEGER PRIMARY KEY, breed TEXT, name TEXT, size INTEGER, age INTEGER, gender INTEGER, fur INTEGER, travels INTEGER, children INTEGER, goodWithAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, nocturnal INTEGER, hypo INTEGER, lifeStyle INTEGER, history INTEGER, loud INTEGER, social INTEGER, colour TEXT, filepath TEXT);

**Table**: lizardStorage

CREATE TABLE lizardStorage(idNum INTEGER PRIMARY KEY, breed TEXT, name TEXT, size INTEGER, age INTEGER, gender INTEGER, fur INTEGER, travels INTEGER, children INTEGER, goodWithAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, nocturnal INTEGER, hypo INTEGER, lifeStyle INTEGER, history INTEGER, diet TEXT, colour TEXT, feed TEXT, space INTEGER, light INTEGER, filepath TEXT);

**Table**: rabbitStorage

CREATE TABLE rabbitStorage(idNum INTEGER PRIMARY KEY, breed TEXT, name TEXT, size INTEGER, age INTEGER, gender INTEGER, fur INTEGER, travels INTEGER, children INTEGER, goodWithAnimals INTEGER, strangers INTEGER, crowds INTEGER, noises INTEGER, protector INTEGER, energy INTEGER, fearful INTEGER, affection INTEGER, messy INTEGER, nocturnal INTEGER, hypo INTEGER, lifeStyle INTEGER, history INTEGER, pattern TEXT, colour TEXT, grooming INTEGER, attention INTEGER, filepath TEXT);

Example of Adding Client and Animal into Persistent Storage

﻿INSERT INTO clientStorage VALUES(1100,'Mitch','Marner','40 Bay St','4166291616','mitch16@gmail.com','Toronto','Ontario',0,0,5,4,0,0,3,2,3,2,3,2,1,4,2,1,3,1,0,'(Cocker Spaniel)(Pug)(Shih Tzu)',1,1,-1,4,3,1,0,'(Bengal)(Russian Blue)',-1,-1,-1,1,4,2,0,0,'',-1,-1,-1,0,0,'Red',0,0,'',-1,-1,-1,0,0,'Red',0,0,'',-1,-1,-1,0,0,'None',1,-1,-1,-1,-1,2,22);

INSERT INTO dogStorage VALUES(2105,'Golden Retriever','Goldy',1,1,'M',1,4,4,4,4,3,2,1,4,1,4,4,0,0,'Outdoor','Street Animal',3,0,1,1,'animalPhotos/Goldy.jpg');

INSERT INTO catStorage VALUES(3100,'Bengal','Cinnamon',1,2,'F',1,1,2,3,2,2,3,3,2,4,1,3,1,1,'Indoor','Brought by owner',4,0,4,'animalPhotos/Cinnamon.jpg');

INSERT INTO birdStorage VALUES(4103,'Cockatoo','Scream',1,4,'F',3,0,1,1,3,2,1,3,2,4,2,1,0,1,'Indoor','Brought by owner',4,4,'White','animalPhotos/Scream.jpg');

INSERT INTO lizardStorage VALUES(5101,'Bearded Dragon','Boswer',2,14,'M',6,1,2,1,3,3,2,4,2,1,3,2,1,1,'Indoor','Brought by owner','Mice','Green','Daily',1,1,'animalPhotos/Bowser.jpg');

INSERT INTO rabbitStorage VALUES(6102,'Belgian Hare','Spot',2,7,'M',1,2,1,3,4,4,2,3,2,1,0,2,0,1,'Indoor','Brought by owner','Spotted','White',4,4,'animalPhotos/Spot.jpg');