**Fuzzy Waffle Auto ~ Design Document**

Group 1: Neal Hamacher, Jordan Beaubien, Kevin Ulliac, Andrew Dool, De Xie

The relational database system we built provides atomic fields for quick and simple building of queries. For example, names and addresses are split into distinct parts so string processing is avoided as much as possible.

Our main relations were Customers, CarTypes, Cars, Branches, Employees, and Rentals. Rentals is the primary relation where all the other relations intersect, taking info from car types, cars, customers, and branches. We decided that rentals would be based on car types; this mimics the way real-world car rental services present their vehicles. It also allowed us to reduce data redundancy and simplify pricing for the consumer and the system as we assigned pricing information for each Car to its CarType. Things such as features, baggage space, and passengers are universal to cars of the same types. It is only once a rental is confirmed that a specific car is selected.

There are several derived attributes – price in Rentals and vehicle stock in Branch being the main ones. The determination of these derived values is obtained by SQL queries in our application. The vehicle stock at a location is determined by looking at the last place a car was returned, because of this all cars must have a rental in the system before showing as available to rent. Due to this constraint, we created an internal customer ID, 1, to use as the internal code for transferring cars from stock to an initial branch, or for transfers between branches. This internal customer ID then was not included for any of the SQL queries determining number of rentals or rental income, as it is not actually a rental but rather an internal transfer.

We decided that all rentals must contain the customer ID of the renter, the branch it is rented from and returned to, and the specific car that is rented (through the car we can access the car type). This enables us to link a rental back to the renting customer and rented cars, in order to process rentals, track usage of specific cars, and track rental history of specific customers.

By minimizing data redundancy, we ensure that our relations are in Boyce-Codd Normal Form (BCNF), which promotes efficient updating and querying.

The application was designed with a distinctly early 2000’s vibe to be easy-to-use and reliable while still being relatively straightforward on the back end. We used one button to search for availability, and one to confirm bookings, so that an SQL query would only be sent out at predetermined times, which allowed us to check input before sending a query that might return errors.

Referential integrity was enforced through use of foreign keys in related tables since all our relations are one-to-many. We decided to use cascading on updates, and not allow deleting if a key is referenced in another table. In situations where deletion is necessary, it is the responsibility of the database administrator to consider the ramifications; they must alter other tables to change foreign keys or delete entries in which the foreign key is referenced.

We added relations for employee logins and customer logins to the ER diagram; these are included only to allow users to access our car rental system.

**Relational Tables**

Customer (customer\_id, first\_name, last\_name, house\_number, street, city, province, dob, age(), driver\_license)

Branch (branch\_id, building\_number, street, city, province, phone\_number, email, vehicle\_stock())

Employee (employee\_id, first\_name, last\_name, house\_number, street, city, province, position, salary, branch\_id)

* branch\_id references Branch

CarType (type, daily\_rate, weekly\_rate, monthly\_rate, dif\_branch\_ret\_price, features, passengers, baggage\_space)

Car (vin, make, model, year, colour, license\_plate, type)

* type references CarType

Rental (reservation\_id, from\_date, to\_date, customer\_id, vin, branch\_id\_pickup, branch\_id\_return, price())

* customer\_id references Customer
* vin references Car
* branch\_id\_pickup references Branch
* branch\_id\_return references Branch

CustomerPhone (customer\_id, phone\_number, type)

* ­customer\_id references Customer

CustomerEmail (customer\_id, email\_address, type)

* customer\_id references Customer

EmployeePhone (employee\_id, phone\_number, type)

* employee\_id references Employee

EmployeeEmail (employee\_id, email\_address, type)

* employee\_id references Employee

CustomerLogin (customer\_id, username, password)

EmployeeLogin (employee\_id, username, password)

**Integrity Constraints**

Rental entries must have full participation in Customer, Car, and Branch Relations.

Employee entries must have full participation in a Branch.

Car must have full participation in CarType.

Fuzzy Waffle Auto Database ER Diagram

