**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, car types, cars, branches, employees, and rentals. Rentals is the primary relation where all the other relations “meet”, 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 cut down on repeated information and simplify pricing for the consumer and the system, 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.

By storing information in the minimal number of positions, we ensure all our relations are in BCNF and thus promote 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 went with a number of button presses to check availability and 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 of 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 the situations where one would need to delete it would be up to the database administrator to consider the ramifications of delete and alter other tables to change foreign keys or delete entries in which the foreign key is referenced.

**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

