COMP 421 PROJECT 1

Group 58: Uber Eats Management and Data System

Jiachen Huo 260779330; Yuhao Cao 260779670;

Jiuyue Cai 260772961; Peiyuan Huang 260685635

1. Requirement analysis

1.1. Introduction

1.1.1. Purpose

This perfect application focused on building a database of Uber eats which is an online ordering and delivery platform, mostly based on managing the relationship between drivers, restaurants and users in the case of an order. By taking advantage of the currently used database model, Uber eats is improving its order processing power and impeccable customer service system.

1.1.2. Scope and special requirements

Distinguished from traditional food delivery platforms, Uber eats send different notifications to users according to their distance away from restaurants in diverse regions. Thus we focused on both order processing and the process of customer services in this application.

1.1.3. Resources

Course materials covered in Comp 421

1.2. Database Description

1.2.1. Entities and their attributes

Driver: Driver is the person who delivers food to customers. A driver has several attributes, including his driver's license, Driver name, Driver ID, rating, and Driver phone number. A driver can be identified by the key attribute Driver ID.

Vehicle: A vehicle captures the information of a car that a driver uses to deliver food. The vehicle entity includes attributes: Vehicle ID, year, insurance, category. Each vehicle is identified by its vehicle ID.

Order: An order has several attributes: order ID, address, fee, tips, price, order time, estimated arrival time, actual arrival time, comment. Order is a record for transaction among restaurant, customer and driver. Each order is identified by a unique order ID.

Restaurant: Restaurant provides food for customers. Every restaurant has an opening hour so this entity has opening time and closing time. Restaurants also need to specify the minimum amount of money spent to deliver the food. When users search on the app, they will see a description about the restaurant and contact information. Thus, the restaurant entity has 7 attributes: Restaurant address, description, open time, close time, minimum amount of order, Restaurant phone number, and a unique Restaurant ID as primary key.

Account: account is used to collect or receive money, which is . We create this entity to memorize the account information in order to make payment or receiving money more. Therefore, we create several attributes according to the real bank card information: Account number, CVV, exp date, Acct name, Birth date. Each account is identified by a unique account number.

Document: document entity records complaints filed by user. We need to record the time user filed complaint, the file ID,file content, open date, version number, and file ID which is a primary key.

Dish: Item is one specific dish of a certain restaurant. Thus, item is a weak entity. Each item has four attributes: discount rate, dish Name, Dish category, Dish price. Each item can only exist by a restaurant.

User: User entity is the customer who makes an order. Users need to provide his or her name and contact information. The app will automatically generate a unique ID, which is the primary key. Therefore, User entity has the following attributes: last name, first name, User phone number, email, and User ID.

Region: Region entity describes the area information such as region code (the first three bit of post code), tax rate, policy, Region Name. Each region is recognized by its region code.

Admin: Admin will contact the user driver and restaurant when an issue happens. It has two attributes: admin ID, admin name. The key attribute is admin ID.

1.2.2. Relationships

Belong To-Driver-Vehicle: A driver drives a car to deliver. This is a one to one relationship, because a driver has to drive one specific car to perform delivery. One can't drive two cars at the same time and two drivers can't drive the same car at the same time.

Allocate: A driver that is dispatched to deliver one order. This is a one to many relationship: a driver can take more than one order or no order, but for each order only one driver is assigned.

Belong To-Driver-Account: The account for each driver. The account is setup to pay delivery fees to drivers. This is a one to one relationship, because a driver only has one account, and each account only corresponds to one person.

Belong To-User-Account: The account for each user. The account is setup for users to make payments for their orders. This is a many to one relationship, because a user may have many accounts, but each account only corresponds to one person.

Belong To-Restaurant-Account: The account for each restaurant. The account is setup for restaurants to receive payments based on the orders. This is a one to one relationship, because a restaurant only has one account, and each account can corresponds to one restaurant only

Make: A user that makes an order. This is a many to one relationship. A user can make many orders, but each order must be made by one user only.

Contain-Order-Dish: An order that includes a dish. This is a many to many relationship. An order can have several amounts of the same dish, and a dish can be ordered in different orders.

Contain-Dish-Restaurant: A restaurant that includes this dish. This is a many to one relationship. A restaurant can have more than two dishes at the same, and each dish can be included in one restaurant only.

Belong To-Restaurant-Region: The restaurant that belongs to one region. This is a many to one relationship. A restaurant must belong to one region, but one region can have any number of restaurants.

Service-Order-Document-Admin: An administrator which participates in a service that might involve orders, and involves documents of the service. This is a many to many to one relationship: a service might not be involved with an irrelevant admin or an irrelevant order. However for every doc there must be at

least an order and at least an admin corresponds to it.

Negotiate-Admin-User: The admin makes decisions for the user based on some issues. This is a many to many relationship. An admin can negotiate and make decisions to many users and a user can be negotiated and informed with decisions from many admins.

Negotiate-Admin-Driver: The admin makes a decision to the driver based on some issues. This is a many to many relationship. An admin can negotiate and make decisions for many drivers and a driver can be negotiated and informed with decisions from many admins.

Negotiate-Admin-Restaurant: The admin makes a decision to the restaurant owner based on some issues. This is a many to many relationship. An admin can negotiate and make decisions to many restaurants and a restaurant can be negotiated and informed with decisions from many admins.

2. Application description

2.1. Algorithm Description

The application aims to fulfill the need of operating an online food delivery platform, namely uber. It seeks to allocate resources efficiently, archive each order information, and track the responsibility for each party involved in each transaction. The application is supported by an optimization algorithm described below.

The core algorithm is the one that retrieves the live recurrent data from the database and performs optimal predictions. One can consider each order as a multidimensional recurrent data coming up in a continuous time measure, where the inputs are the driver, user, and restaurant's information. Our algorithm will perform live-prediction based on these inputs for various order awaiting and give an optimal assignment (assign a driver to deliver an order) as response and update the solution to units involved, as well as updating the database. This algorithm mainly requires methods that involves high dimensional statistical learning with time series technique (Recurrent Neural Network, for example) to support such complex algorithm.

2.2. Variables to be Determined

1. The live traffic information, including weather condition, traffic jams, road construction, etc
2. Accidents may happen on the road that interrupts food delivery

3. Relations

3.1. Entities

account(Account number, Acct name, CVV, Birth date, Exp date)

vehicle(Vehicle ID, year, insurance, category)

restaurant(restaurant ID, description, open time, close time, minimum amount of order, restaurant phone number, restaurant address,region code,account number)(region code ref region, account number ref account)

user(user ID, last name, first name, user phone number, email)

driver(driver ID, driver license, driver phone number, driver name, rating, vehicle ID, account number)(vehicle ID ref vehicle, account number ref account)

region(region code, region name, tax rate, policy)

document(file ID, open date, version number, file content)

order(order ID, address, comment, tip, fee, order time, estimated arrival time, actual arrival time,user ID, driver ID) (user ID ref user, driver ref driver)

admin(admin ID, admin name)

3.2. Weak Entities

dish(dish name, restaurant ID, dish category, discount rate, dish price) (restaurant ID ref restaurant)

3.3. Relationships

service(file ID, order ID, admin ID) "file ID ref document" "order ID ref order" "admin ID ref admin"

Allocate (orderID, driverID) :

order (order ID, address, comment, tip, fee, order time, estimated arrival time, actual arrival time,user ID, driver ID, driver ID) "driverID ref driver"

driver (driver ID, driver license, driver phone number, driver name, rating, vehicle ID, account number)

Belong-To-driver-vehicle (Driver ID, Vehicle ID) "Driver ID ref driver" "Vehicle ID ref vehicle"

Belong-To-driver-account:

driver(driver ID, driver license, driver phone number, driver name, rating, vehicle ID, account number) "account number ref account"

account(Account number, Acct name, CVV, Birth date, Exp date)

Belong-To-user-account (user IDaccount number) participation constraint on user "account number ref account" "user ID ref user"

Belong-To-restaurant-account:

Restaurant (restaurant ID, description, open time, close time, minimum amount of order, restaurant phone number, restaurant address,region code,account number, account number)"Account number ref account"

account(Account number, Acct name, CVV, Birth date, Exp date)

Belong-To-restaurant-region:

Restaurant (restaurant ID, description, open time, close time, minimum amount of order, restaurant phone number, restaurant address,region code,account number, region code)"Region code ref region"

region(region code,region name, tax rate, policy)

Make-Order-User:

order(order ID, address, comment, tip, fee, order time, estimated arrival time, actual arrival time,user ID) "User id ref user"

user(user ID, last name, first name, user phone number, email)

Contain-dish-order(dish name, order id) participation constraint on order "dish name ref dish" "order id ref order"

Contain-dish-restaurant: participation constraint on restaurant

dish(dish name, restaurant ID, dish category, discount rate, dish price) "restaurant ID ref restaurant"

restaurant(restaurant ID, description, open time, close time, minimum amount of order, restaurant phone

number, restaurant address,region code,account number)

Negotiate-user (admin ID, user ID) "Admin ID ref admin" "user ID ref user"

Negotiate-driver (admin ID, driver ID) "Admin ID ref admin" "driver ID ref driver"

Negotiate-restaurant (admin ID, restaurant ID) "Admin ID ref admin" "restaurant ID ref restaurant"

4. Creativity and Complexity the Design

For the creativity of this database application, we used some properties of weak entity, ternary relationship and constraints as the followings:

1. Weak entity: Dish. Based on constraints that each dish depends on it's belonged restaurant, thus it's a weak entity.

2. Ternary relationship: The relationship is built between user, administration of the uber eats platform, and document file created. As for dealing with any complaints that user could make, it keeps tracking the updated information created by the file.

3. Constraints:

Participation only: user– > account;

restaurant–>dish; order– > dish

Key +participation: restaurant–>account;

driver–>account; vehicle–>driver;

driver–>vehicle; dish–>restaurant;

restaurant− > region; order–>user;

order–>driver

Key only: account–>driver