Database System Project

Deliverable: Database Implementation
Group3: Uber

Link: https://webdev.cs.uiowa.edu/~changzhan/

Description:

This is the final deliverable of our database system project. In this document, we included:

1. Cover page
2. Project website link
3. Individual contributions
4. Documentation for the database scheme design
5. Documentation for the queries

In addition, we also submitted all related project files:

1. SQL file with schema and sample data - *"schema.sql"*
2. PHP file with SQL queries - *"queries.php"*
3. First deliverable: Project description - *"Group3_Project_Description_Uber.pdf"*
4. Second deliverable: ER Model - *"Group3_(updated)ER_Model_Uber.pdf"*
5. Third deliverable: Relational Design - *"Group3_Relational_Model_Uber.pdf"*

Individual Contributions:

Changze Han:

- Made Cover page;
- Wrote Documentation for schema;
- Created webpage; load database schema; import PHP file;
- Updated & reformat PHP file.

Cory Skeers:

- PHP implementation;
- Wrote Documentation for queries;

- Constructed Queries;
- Schema debug.

Josh Kamp:

- Wrote entire database schema

Cameron Chen:

- Wrote Dummy Data;
- Ensure Correctness of Schema

Kaiqiang Zhang:

- MongoDB Model

Scheme Design Documentation:

Drivers:

We used a combination of two attributes driver_id and driver_license as primary key because they can uniquely identify a tuple. Other attributes of a driver can't.
Both driver_id and driver_license can't be null.

Cars:

We used a combination of two attributes plate_number and driver_id as primary key because they can uniquely identify a tuple. Other attributes of a car can't.
Both driver_id and plate_number can't be null.

Payment_account:

We used a foreign key driver_id as primary key because driver is the parent table. Payment account also has its own primary key bank_account_number. This way, we can uniquely identify a tuple, other attributes of a payment account can't do that. Both bank_account_number and driver_id can't be null. If a record in driver table is deleted, then the corresponding records in the payment account table will automatically be deleted.

Payments:

We used an artificial key payment_id as primary key because it can uniquely identify a tuple. Other attributes of a Payment can't.
Payment_id can't be null.

Driver_customer_payments:

We used three foreign key driver_id, payment_id, and customer_id as primary key. We designed this table to connect corresponding drivers, customers and payments together in order to conveniently do all kinds of queries. All three of them can't be null. If a record in either driver, customer, or payment is deleted, then the corresponding record in this child table will also be deleted automatically.

Driver_ratings:

We used a foreign key driver_id as primary key. This child table also has its own primary key which are rating_data and rating_time. In this way, we can uniquely identify a tuple. Other attributes of this table can't. If a record of driver is deleted, then the corresponding record in this child table will also be deleted automatically. All keys can't be null.

Regions:

We used three attributes region_zipcode, region_state, region_city as primary because this combination can uniquely identify a tuple. Either one of them alone can't do that. All of them can't be null.

Trips:

We used a trip_date, trip_time and an artificial key teip_id together as primary key, because this combination can uniquely identify a tuple. Other attributes in this table can't. All three of them can't be null.

Driver_customer_trips:

We used three foreign keys driver_id, trip_id, customer_id as primary key, because they can uniquely identify a tuple. This is a child table. If a record of either drivers, customers, or trips is deleted, then the corresponding record in this child will also be deleted automatically. All keys can't be null.

Customers:

We used an artificial key customer_id as primary key because it can uniquely identify a tuple. Other attributes of customer can't. Customer_id can't be null.

Credit_cards:

We used a foreign key customer_id as primary key. This is a child table, it also has its own primary key card_number. These two keys can uniquely identify a tuple. Other attributes of a credit_card can't. Both of them can't be null. If a record of customer is deleted, then the corresponding record in this table will also be deleted.

Customer_ratings:

We used a foreign key customer_id and two attributes cr_date, cr_time together as primary key, because they can uniquely identify a tuple. Other attributes of customer rating can't. All three of them can't be null.

Accidents:
We used an artificial key accident_id and two attributes accident_date, accident_time as primary key because they can uniquely identify a tuple. Other attributes of an accidents can't. Accident_id can't be null.

Reports:
We used an artificial key report_id as primary key because it can uniquely identify a tuple. Other attributes of reports can't. Report_id can't be null.

Queries Documentation:

Query1: For repeat customers, what is their average trip distance, trip price, and tip amount?
We select customer name, average trip distance, average payment amount, and the average of the division of payment tip by payment amount. These attributes are taken from the customers, trips, and payments tables, using the driver_customer_trips and driver_customer_payments tables as links between these. Finally, the results are grouped by customer ID and reduced to only those in which a customer ID appears more than once in the driver_customer_trips table, indicating that they have used the service more than a single time (and are thus a repeat customer).

Query2: Do drivers who maintain an average rating above a 3.0 receive greater average tips than those drivers with below a 3.0 average?
Two queries are used, one limited by a driver maintaining an average rating above 3.0, and one for drivers with an average rating below 3.0. The results from the driver_ratings table are joined with the payments table through driver_customer_payments in order to determine their average tip amount. These two queries are combined with a union to display both the above 3 and sub-3 driver tip amounts.

Query3: Which driver made the most money in the last three months, and how much was that?
The driver name and summed payment amount are selected from the driver and payments tables, joined through the driver_customer_payments table. The results are then ordered by the total summed payments (descending), and limited to 1 so that we received only the driver with the top payment amount. (Note: An additional subquery condition would need to be added to limit the

results to include only payments from the last three months, but this has been removed for this proof-of-concept display due to the nature of our available test data).

MongoDB Model:

```
Customer{
        _id: <0000>
        customer_id: 1,
        customer_name: “customer”,
        customer_address: “Shandong province”,
        customer_phone: 319-000-0000
}
drivers {
    _id: <0001>,
        driver_id: 1,
        driver_license: “license ID”,
        driver_name: “name”,
        driver_address: “address”,
        driver_phone: 319-999-9999
}
Cars {
        _id: <0002>,
        plate_number: “plate”,
        diver: <0001>,
        make: “MAKE”,
        year: 2019
}
payment_account{
        _ip: <0003>,
        bank_account_number: 111111,
        routing_number: 222222,
        ssn: 2343212,
        driver: <0001>
}
Payments{
    _ip: <0004>,
    payment_id: 1,
    payment_date: 13/12/2019,
    payment_time: 23:59:59,
    payment_amount: 99.99,
    payment_tip: 1.00,
    payment_tax: 9.99,
    payment_status: “PENDING”,
    }
driver_customer_payments{
        _id: <0005>,
        Payment: <0004>,
        Driver: <0001>,
        Customer:<0000>
}
driver_rating {
        _id: <0006>,
        driver: <0001>,
        rating_date: 13/12/2019,
        rating_time: 23:59:59,
        rate: 1.0,
        rating_comments:“too slow”,
}
Regions{
        _id: <0007>
        region_zipcode: 52241,
        region_state: “IA”,
        region_city: “CORALVILLE”
}
Trips{
        _id: <0008>
        Customer: <0000>
        Driver: <0001>
    trip_id: 1,
        trip_date: 13/12/2019,
        trip_time: 23:59:59,
        trip_distance: 1000,
        pickup_location:“SC”,
        dropoff_location: “Target”,
        payment: <0004>
}
driver_customer_trip{
```

```
          _id: <0009>
      trip: <0008>,
          customer: <0000>,
          driver: <0001>
}
credit_cards{
          _id: <0010>
      customer: <0000>,
          card_number: 0000 0000 0000 0000,
          card_expiration: “12/2099”,
          card_code: 233
}
customer_rating{
          _id<0011>
      Customer: <0000>,
          cr_date: 13/12/2019,
          cr_time: 23:59:59,
          cr_rate: 5.0,
          cr_description: “give tips”
}
Accidents{
          _id: <0012>
          accidenter: <0001>
      accident_id: 1,
          accident_date: 13/12/2019,
          accident_time 20:59:59,
          accident_description: “deadly injured”
}
report{
          _id: <0013>
          report_id: 1,
          trip: <0008>,
          report_date: 14/12/2019,
          report_time: 00:00:00,
          report_description: “boring”
}
```