Logical and Physical Schema-Hotel Group case study

```
A2 <- dbConnect(RSQLite::SQLite(), "A2.sqlite")
```

A2 Logical data Schema

- **customers**(*cust_id*,customer_name, customer_first_name, customer_middle_name,customer_last_name, customer_cell_phone)
- **cars**(*car_id*,car_brand, car_is_new, car_color, car_mileage, car_age)
- **dealers**(*dealer_id*,dealer_name, dealer_address)
- **car_purchase_orders**(*po_id* ,*customer_id*, *car_id*, *dealer_id*,purchase_at, payment)
- **package_purchases**(*package_id*, *po_id*,package_fee, start_at, end_at)
- **repair_records**(*repair_id*, *dealer_id*, *po_id*, mileage, repair_at, labor_cost, spare_cost)

Like all other businesses, we need to record customer information, such as customer names and contact information. Thus, we need to create a 'customer' entity. Each car information, including brand, model, color, and so on, should be recorded in the 'car' entity. There are many dealers with different names and addresses available for customers to choose. So we include a 'dealer' entity to record different dealers' information. The dealer provides two kinds of service - car sales and car repair. Thus, we need two entities, 'car_purchase_order' and 'repair record' to record information for these two businesses. For car purchases, we can use customer_id, car_id, and dealer_id to locate a specific purchasing order. Package purchase is attached with the purchasing order. Thus, we use po_id as a foreign key for 'package_purchase entity'. It records the price and valid period of a service package. We assume we can get all dealers' information, then, for the 'repair_record' entity, a repair action must be after a purchase action. Thus, we use 'dealer_id' and 'po_id' as foreign keys to locate a repair record.

How many customers have stopped bringing their cars after the first encounter with the dealer?

Using po_id and dealer_id to combine the car_purchase_order and repair_record. We would like to count how many customers have stopped bringing their cars after the first encounter with the dealer. In other words, it includes two groups of customers: 1) Customers who purchased the car but never repair the car. 2) Customers who purchased cars from one dealer but repaired cars in another dealer. Since we use the LEFT JOIN

command, these two kinds of customers will have the “NULL” value for repair_id. Thus, we use ‘WHERE repair_id IS NULL’ to select the needed values. A customer may purchase multiple cars, which will lead to a cutomer_id attached with many po_id. Thus, we use ‘DISTINCT customer_id’ to ensure we only count each customer once.

```
SELECT count(DISTINCT customer_id) AS "#CustomerChurn"
FROM (
SELECT customer_id, rr.dealer_id, car_id
FROM car_purchase_order AS cp
JOIN repair_record AS rr
ON cp.po_id = rr.po_id
GROUP BY customer_id, rr.dealer_id, car_id
HAVING count(DISTINCT repair_id)=1 )
```

1 records

#CustomerChurn
0

What is the relationship between the price of the service and the age of the car in terms of (a) actual car age (e.g., mileage) and (b) time with thecurrent owner?

Service_cost is composited of labor_cost and spare_cost. The car’s mileage is computed through the initial mileage from the car record and the mileage when repairing the car. We choose ‘DATE’ data type for ‘repair_at’(repair date) and ‘purchase_at’(purchase date). Thus, we use julianday function to calculate the duration and divide 365.25 to convert unit from ‘days’ to ‘year’. We use ‘car_id’ and ‘po_id’ join three entities together to select the information we needed.

```
SELECT cp.car_id,labor_cost + spare_cost AS service_cost, rr.mileage -
c.car_mileage AS mileage, (julianday(repair_at) - julianday(purchase_at))/
365.25 years
FROM repair_record AS rr
JOIN car_purchase_order AS cp
ON rr.po_id = cp.po_id
JOIN car AS c
ON cp.car_id = c.car_id;
```

0 records

car_id	service_cost	mileage	years

```
A2 <- DBI::dbConnect(RSQLite::SQLite(),"A2.sqlite")
```

```sql
-- customer
CREATE TABLE 'customer' (
 'customer_id' NUMERIC PRIMARY KEY,
 'customer_name' VARCHAR,
 'customer_first_name' VARCHAR,
 'customer_middle_name' VARCHAR,
 'customer_last_name' VARCHAR,
 'customer_cell_phone' NUMERIC NULL
);

-- car
CREATE TABLE 'car' (
 'car_id' NUMERIC PRIMARY KEY,
 'car_brand' VARCHAR,
 'car_model' VARCHAR,
 'car_is_new' INTEGER,
 'car_color' VARCHAR,
 'car_mileage' REAL,
 'car_age' REAL
);

-- car
CREATE TABLE 'car' (
 'car_id' NUMERIC PRIMARY KEY,
 'car_brand' VARCHAR,
 'car_model' VARCHAR,
 'car_is_new' INTEGER,
 'car_color' VARCHAR,
 'car_mileage' REAL,
 'car_age' REAL
);

-- dealer
CREATE TABLE 'dealer' (
 'dealer_id' NUMERIC PRIMARY KEY,
 'dealer_name' VARCHAR,
 'dealer_address' VARCHAR
);

-- car purchase order
CREATE TABLE 'car_purchase_order' (
 'po_id' NUMERIC PRIMARY KEY,
 'purchase_at' DATE,
 'payment' REAL,
 'customer_id' NUMERIC,
 'car_id' NUMERIC,
 'dealer_id' NUMERIC,
 FOREIGN KEY('customer_id')
   REFERENCES customer ('customer_id'),
 FOREIGN KEY('car_id')
   REFERENCES car ('car_id'),
```

```
 FOREIGN KEY('dealer_id')
   REFERENCES dealer ('dealer_id')
);

-- package purchase
CREATE TABLE 'package_purchase' (
 'package_id' NUMERIC PRIMARY KEY,
 'package_fee' REAL,
 'start_at' DATE,
 'end_at' DATE,
 'po_id' NUMERIC,
 FOREIGN KEY('po_id')
   REFERENCES car_purchase_order ('po_id')
);

-- repair record
CREATE TABLE 'repair_record' (
 'repair_id' NUMERIC PRIMARY KEY,
 'mileage' REAL,
 'repair_at' DATE,
 'labor_cost' REAL,
 'spare_cost' REAL,
 'dealer_id' NUMERIC,
 'po_id' NUMERIC,
 FOREIGN KEY('po_id')
   REFERENCES car_purchase_order ('po_id'),
 FOREIGN KEY('dealer_id')
   REFERENCES dealer ('dealer_id')
);
```