Yuwen Sang

COMP 3421: A6

E/R Schema of the Auto Insurance Company Database:

Coverage (**coverageID,** cAmount**,** planName**,** price)
States (**stateName,** climate, pDensity)
Driving Records (**recordID,** DRdate, DRtype)
Customers (**customerID**, cname, age, gender, SSN, dlNum, **coverageID, stateName**)
Cars (**VINcode,** brand, color, ctype, **customerID**)
Premium (**premiumID**, paymentPeriod, cAmount**, recordID, customerID**)

1. Table Creation Script and the insertion that violate the foreign key constraint

```
show databases;
drop database if exists carInsurance;
create database carInsurance;
use carInsurance;

# create 6 tables
#1. Coverage
drop table if exists Coverage;
create table Coverage(
        coverageID int NOT NULL,
        cAmount varchar(10) NOT NULL,
   planName varchar(15) NOT NULL,
   price int NOT NULL,
   primary key (coverageID));
describe Coverage;

#2. States (here we use states instead)
drop table if exists States;
create table States(
   stateName varchar(20) NOT NULL,
   climate varchar(15) NOT NULL,
   pDensity varchar(10) NOT NULL,
   primary key (stateName));
describe States;

#3. DrivingRecords
drop table if exists DrivingRecords;
create table DrivingRecords(
   recordID int NOT NULL,
   DRdate varchar(15),
   DRtype varchar(20) NOT NULL,
   primary key (recordID));
describe DrivingRecords;
```

```
#4. Customers
drop table if exists Customers;
create table Customers(
        customerID int NOT NULL,
   cname varchar(30),
   age int,
   gender varchar(10),
   ssn int,
   dlNum int,
   coverageID int NOT NULL,
   stateName varchar(20) NOT NULL,
   primary key (customerID),
   foreign key (coverageID) references Coverage(coverageID),
   foreign key (stateName) references States(stateName));
describe Customers;

#5. Cars
drop table if exists Cars;
create table Cars(
        VINcode varchar(20) NOT NULL,
   brand varchar(15),
   color varchar(15),
   ctype varchar(10),
   customerID int NOT NULL,
   primary key (VINcode),
   foreign key (customerID) references Customers(customerID));
describe Cars;

#6. Premium
drop table if exists Premium;
create table Premium(
        premiumID int NOT NULL,
   paymentPeriod varchar(15),
   cAmount varchar(10) NOT NULL,
   recordID int,
   customerID int NOT NULL,
   primary key (premiumID),
   foreign key (recordID) references DrivingRecords(recordID),
   foreign key (customerID) references Customers(customerID));
describe Premium;
```

Suppose we want to directly insert a tuple into the Premium table, we will get the foreign key constraint violation warning as follow:

```
mysql> insert into Premium values(402720513,"month","low",10000,30000);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`carinsurance`.`premium`, CONSTRAI
NT `premium_ibfk_1` FOREIGN KEY (`recordID`) REFERENCES `drivingrecords` (`recordID`))
mysql>
```

2. Create A Procedure

The code below shows the procedure that outputs the total number of customers have had some specific type of accident (accident_type, equals to “serious”, “medium”, or “slight”) and age younger than the cust_age years old.

Create The Procedure:

```
drop procedure if exists accident_proc;
delimiter //
create procedure accident_proc(IN accident_type varchar(20), IN cust_age INT, OUT
cust_count INT)
begin
    select count(*) into cust_count from Customers C, DrivingRecords DR, Premium P
    where C.customerID = P.customerID and DR.recordID = P.recordID and DR.DRtype
    = accident_type and C.age < cust_age;
end //
delimiter ;
```

Run The Procedure:

```
set @accident_type = "medium";
set @cust_age = 30;
set @cust_count = 0;
call accident_proc(@accident_type, @cust_age, @cust_count);
```

Outputs:

```
mysql> drop procedure if exists accident_proc;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> delimiter //
mysql> create procedure accident_proc(IN accident_type varchar(20), IN cust_age INT, OUT cust_count INT)
    -> begin
    -> select count(*) into cust_count
    -> from Customers C, DrivingRecords DR, Premium P
    -> where C.customerID = P.customerID and DR.recordID = P.recordID and DR.DRtype = accident_type and C.age < cust_age;
    -> end //
Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;
mysql> set @accident_type = "medium";
Query OK, 0 rows affected (0.00 sec)

mysql> set @cust_age = 30;
Query OK, 0 rows affected (0.00 sec)

mysql> set @cust_count = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> call accident_proc(@accident_type, @cust_age, @cust_count);
Query OK, 1 row affected (0.03 sec)

mysql> select @cust_count;
+-------------+
| @cust_count |
+-------------+
|          91 |
+-------------+
1 row in set (0.00 sec)
```

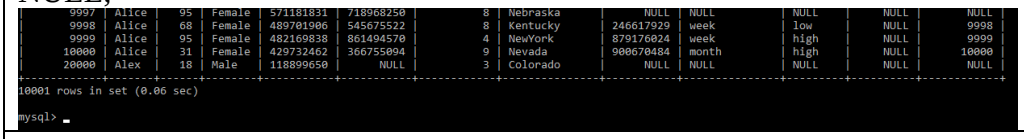
3. Show The Speed Difference between Using and Not Using Index in The Selection Statement

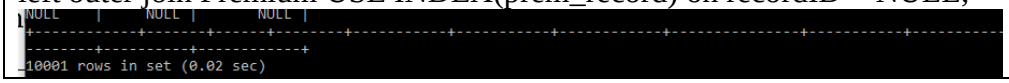
Test Select statement:

1) Simple statement: Find the customers that customerID smaller than 10001 (10000 results in the database).

Selection on a single relation	*Time Cost*
Not Use Index Code: Select C.customerID from Customers C where c.customerID < 10001;	*0.03sec*
Use Index Code: create index cust_index on Customers(customerID); Select customerID from Customers USE INDEX (cust_index) where customerID < 10001;	*0.02sec*

2) Find the customers their personal information and the premium information who don't have the driving accident record (10001 results in the database)

Selection Involves a Join	Time Cost
Not Use Index Code: Select * from Customers C left outer join Premium P on C.customerID = P.customerID and P.recordID is NULL; 	0.06 sec

Use Index Code:
 	
create index prem_record on Premium(customerID, recordID);
 	
Select *
from Customers
left outer join Premium USE INDEX(prem_record) on recordID = NULL;
 
	0.02 sec