

Homework #6 DB Programming

In this assignment, you will implement your SQL queries from HW #5 using interactive SQL-Server Express 2008/2012 AND Postgres, a simple scripting language, and a Java program using the JDBC/ODBC interface. Your queries can be implemented as a WebApp for Extra Credit.

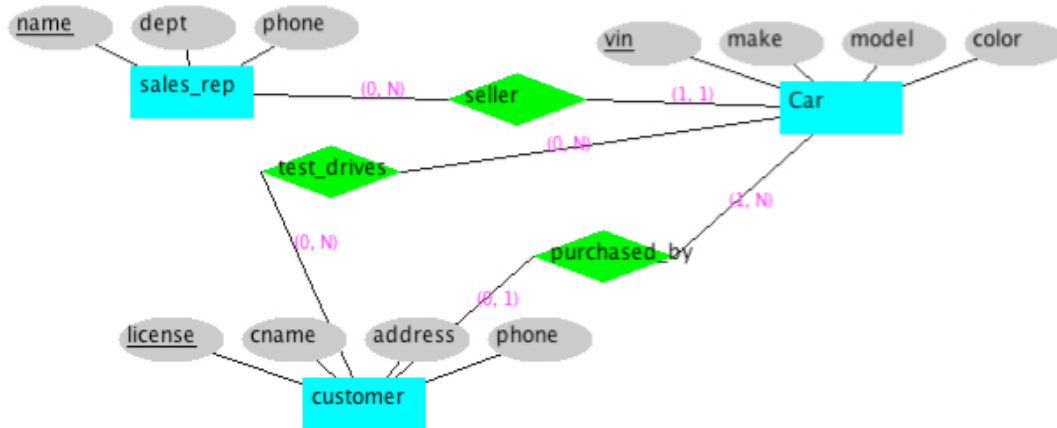
Here are the schema & queries for HW #5 (for reference):

Consider the following relational schema:

Sales_Rep (name, dept, phone)
Car (vin, make, model, color, seller, purchased_by)
 FK seller references Sales_Rep
 FK purchased_by references Customer
Customer (license, cname, address, phone)
Test_Drives (vin, license, date)
 FK vin references Car
 FK license references Customer


The Test-drives relation lists the date on which a car is test driven by a customer (note that customers can only test drive a particular car at most once!) While each car is purchased by a single Customer and sold by a single Sales_Rep, a Customer may purchase many cars and a Sales_Rep may sell many cars.

The equivalent ER Schema is given here for reference (note the attribute 'date' is missing from the test_drives relationship in this diagram!)



Write the following queries in SQL that is as simple as possible. ***Explain clearly why your relational algebra query should produce the correct results (independent of any particular data values).*** Include any additional output attributes necessary to unambiguously present the data requested (but do not include any unnecessary attributes)!

Include any additional output attributes necessary to unambiguously present the data requested (but do not include any unnecessary attributes)! ***Note that this is not a programming assignment – you will implement your SQL queries on the next assignment!***

1. Find the names and departments of all sales_reps who have sold a 'green' car.
2. Find the cnames of customers who have purchased a car that was sold by someone in the "Toy" department.
3. Find the names of all sales_reps who sold a car that was test driven (by anyone) before October 1, 2012.
4. Find the vins of all cars that were test driven by a customer who also purchased them.
5. Find the vins of cars that were test driven by a customer who did ***not***  also purchase them.
6. Find the cnames of customers that have test driven ***every*** car.
7. Find the names of sales reps that ***not*** sold any car test driven by a customer named "Psmith".
8. Find the cnames of customers that have test driven ***every*** red car and ***no*** blue cars.
9. Find the cnames and phones of customers that have ***not*** test driven any cars.
10. Find the names of customers who ***only*** test drive 'VW Bugs' (that is, cars with make = "VW" and model = "Bug").
11. Find the ***average*** number of cars that each sales-rep sells.
12. Find the name and address of the customer who has test driven the ***maximum*** number of cars.

I. (100 points) Implement your queries using interactive MS SQL-Server 2008/2012 (either Express or the full edition) AND Postgres. You should create test tables containing 2-3 tuples each, and then execute your SQL queries using the interactive SQL interface (as demonstrated in lecture). ***Your test output annotation should explain clearly why the output tuples are correct, given the input data. Design your test data so that each query has at least some output (you can use different test data for the different queries), and be sure to list all of the input data along with your query results for verification!***

II. (50 points) Implement **ONE** of your queries using the Java JDBC/ODBC interface; AND implement **ONE** of your queries using your favorite scripting language. ***Be sure to have appropriate comments in both your Java program and your script, and include complete design documentation in your writeup.***

III. Extra Credit I: Re-implement your solutions to Parts I & II using (a) MySQL and/or (b) SQLite and/or (c) Oracle. Be sure to include complete output annotation and a discussion of any differences in the SQL Query Syntax between the different SQL Engines.

IV. Extra Credit II: Implement a simple graphic interface (using either Java or a scripting language) that prompts the user for a query number, and then executes the corresponding query (more extra credit if the user is given more choices of queries to

execute from among the 12 given above). Extend your interface to allow the user to enter query parameters, for example the color of the car in Query #1.

V. *More Extra Credit:* Re-implement your user interface as a WebApp running on a browser on one machine with the DBS running on a different (back-end) server machine. Demonstrate that you can access the DBS from multiple client machines and explain the linkage & implementation techniques that you used in your documentation. See the 675 External Links page for a collection of tutorials covering all of the supporting technologies. Note that the 675 Virtual Box does not support remote access over the Internet, so you will need to host your WebApp on TheCity!