Group 14

John Keller

7 December 2018

Database System Concepts

Database Systems Group 14 Project Report

**Introduction:**

For our final project in Database Systems Concepts, our group decided to model a chain of car dealerships. This model includes a table of dealerships and the employees, addresses, sales, customers, cars, and car features associated with each dealership. Our final projects allows potential users to get information about customers, get information about employees, search details about particular car makes and models, search features associated with individual cars, analyze the sales performance of each individual employee, analyze the sales performance of each individual dealership, organize employees by the employees with the greatest total sales, and organize customers by the greatest total purchases.

Our project is designed to be used by both dealership employees (both individuals involved in upper management and salespeople who wish to assist customers in purchasing a vehicle). Managers at dealerships could use our application to analyze employee or customer information to determine the most consistently high-performing employees or most consistently loyal customers so that they reward such individuals. Additionally, higher-level managers could use our application to determine the highest-performing dealerships so that they can invest more resources at these locations. A salesperson, on the other hand, could use our application for basic car-searching needs. Customers can use the application to search for cars based on some of the most commonly searched details of a car including, make, model, color, year, mileage, previous owner counts, and the current asking price of the vehicle. Once the salesperson has identified a car that suits the customer’s particular needs, the car can be purchased and the purchase is added to our database.

**Technical Description:**

To begin work on our project, we first created eight tables within Microsoft SQL Server Management Studio. The tables included are as follows: Demo. Feature, Demo.[Address], Demo. Dealership, Demo.Car, Demo.CarFeature, Demo.Employee, Demo.Customer, and Demo.Sale. These tables provide a foundation for our database.

Once we had created the tables for our database, we populated the table with data included in our data.sql file. The majority of the data that we created was retrieved from Mockaroo.com. With the mock data provided on this website, we were able to populate our tables with 500 customers and 500 employees along with first names, last names, various IDs, phone numbers, email addresses, employee salaries, and employee job titles that relate to each person within our database. In addition to the information that we generated for each person in the database, we generated physical address information (city, street name, and zip code) information where necessary. Finally, we generated information about each car, including year information, car makes, car models, car colors, car mileage, asking prices, and dealership IDs for the dealerships associated with each car.

Once we had populated all of the tables in our database with mock information, we began to create procedures. In our procedures.sql file, we wrote 16 SQL procedures with which we could modify and organize the information within our database. Among the functions of these procedures are the ability to gather customer purchases, get the features associated with particular cars, search for cars based upon their attributes, and gather information about the performance of dealerships.

Finally, once we had established the structure of our database and created procedures with which we could modify the data in the database, we developed the functionality of our user interface with C# code in Microsoft Visual Studio 2017. We began by creating a central Windows Form that acts as our homepage and gives users access to the various functionalities of our application. Branching off of our homepage, we included eight other windows forms that bring users to windows that allow them to search for cars, assess dealership performance, assess employee performance, search cars by feature, gather customer information, organize top employees, search employees, and make purchases. Each form includes a corresponding page of code written in C# that controls the functionality of the page. These functionalities include calling database procedures through an established connection, creating data grids, and organizing information retrieved from procedure calls so that it is more easily interpreted by users.

**Database Design:**

To properly model all of the data needed for our database, we needed 7 regular tables and 1 linking tables. These tables represent employees, dealerships, addresses, customers, sales, cars, and car features. All the different tables that we used are as follows:

1. Employee

The employee table contains a primary key, employee id. It also has foreign keys to connect it to the address and dealership tables so that each employee can be associated with an address and dealership. The table also contains employee information for first/last name, salary, title, and phone number. Finally, there is a unique key representing the employee’s email. We use this to input into sales to indicate who the salesman is, so it is important that it is unique.

1. Dealership

Every dealership has a primary key of a dealership id. It also contains a foreign key to connect it to the address table. Finally, every dealership has a name and a phone number associated with it.

1. Address

Addresses are used to keep track of where things are. Every address has a primary key of address id. Addresses then contain information about the city, street, street2, and zipcode that the address represents.

1. Customer

Customer information is kept track of in our database using a primary key customer id. It also contains a foreign key that assigns addresses to employees. The customer also has information such as name, phone number, and email address.

1. Sale

Every sale is kept track of in our database with a primary key of sale id for each one. There are foreign keys that assign an employee, customer, and car to each sale. The information of sale amount and sale date are also stored in the database.

1. Car

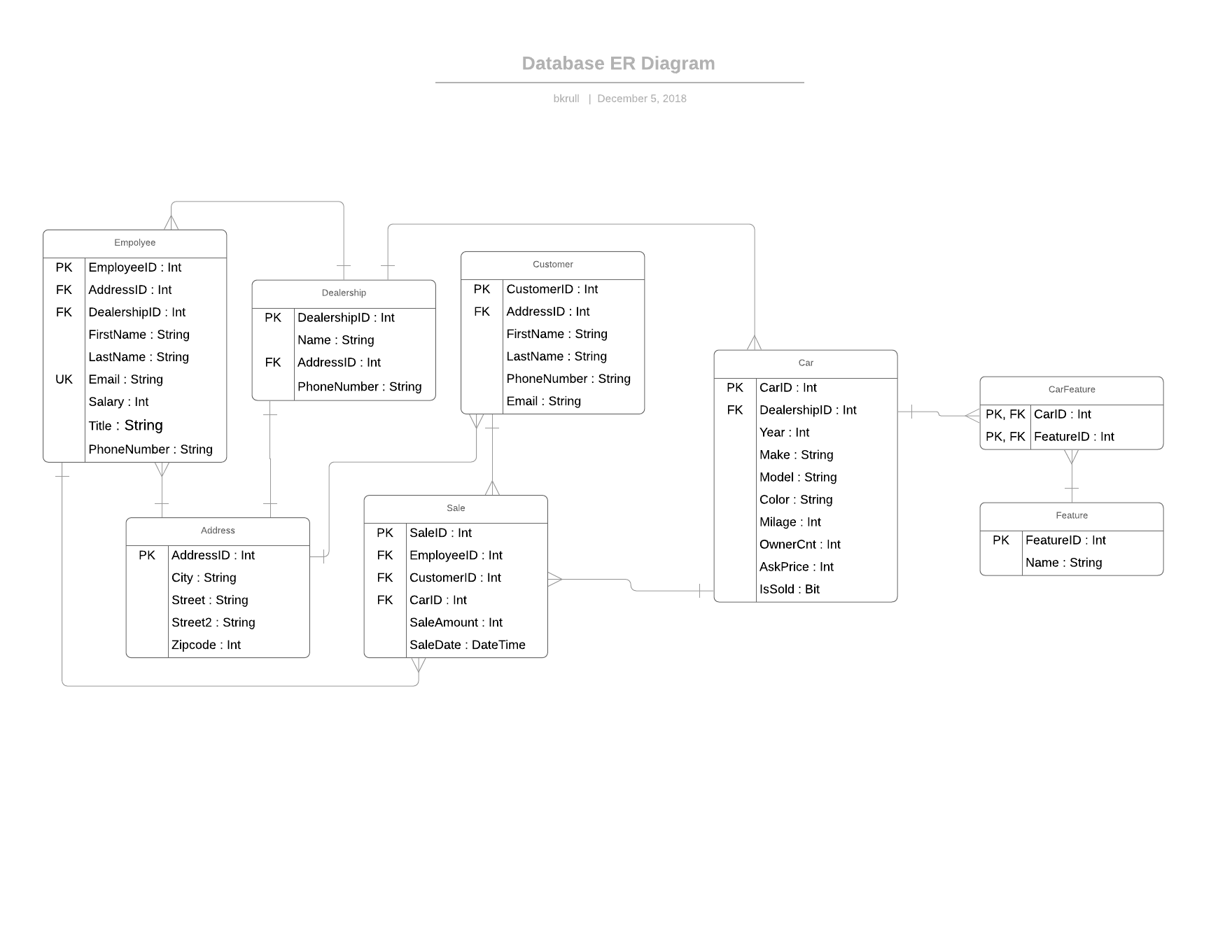
Cars are one of the most important parts of the database, and they are kept track of using a primary key of car id. There is also a foreign key that keeps track of which dealership the car is located at. There is a load of information on the car in the database, including year, make, model, color, mileage, owner count, asking price, and a flag to indicate whether or not it has been sold.

1. CarFeature

CarFeature is a linking table that is used to connect each car with its corresponding feature. It contains a car id and feature id, which are both foreign keys and combine to make the primary key. We used this because there were going to be a lot of different feature combinations for different cars, and we wanted to minimize the amount of data that was repeated in the database.

1. Feature

Finally, the feature table contains the information about specific features that a car can have. It has a primary key of feature id as well as a string to give the name of the feature. Every feature can be represented in many CarFeatures, and many CarFeatures can reference a single car.

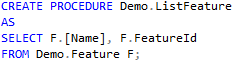
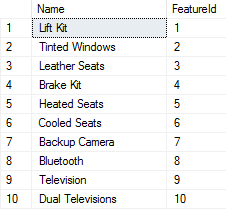


**System Design:**

**System Features and Usage:**

**Queries and Reports:**

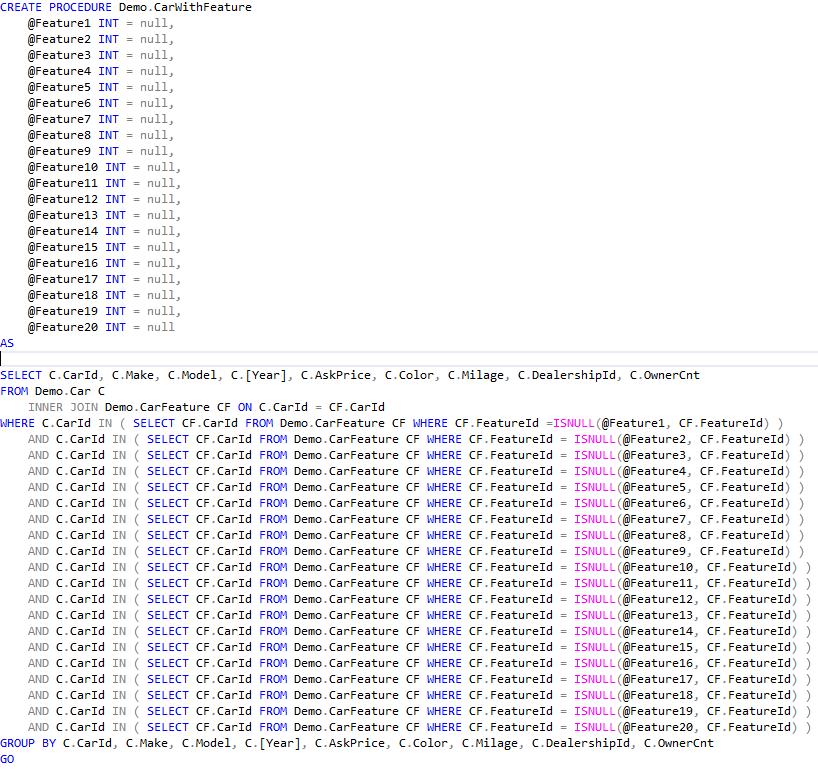
Demo.ListFeature is a question-type query. This procedure retrieves the names of all features so that they can be listed in the FeatureSearchForm.



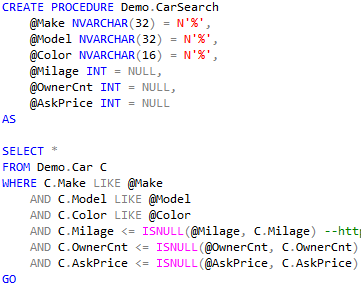
Demo.CarWithFeatures is a question-type query. This procedure allows users to lookup cars based on which features they have.



This table shows the results for determining cars that have feature 19 and feature 20.

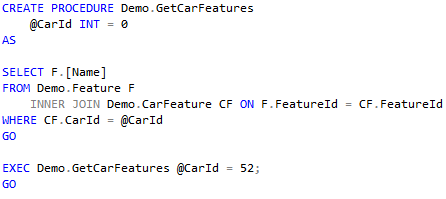


Demo.CarSearch is a question-type query. This procedure allows user to lookup cars based on features that the user specifies.

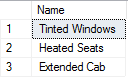


This table shows the result of the query with @make set to Ford and @Color set to Teal.

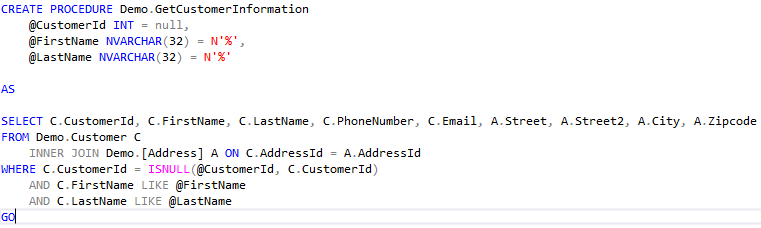
Demo.GetCarFeatures is a question-type query. This procedure allows the user to find a particular car’s features.



This table shows the features belonging to the car with ID = 52.

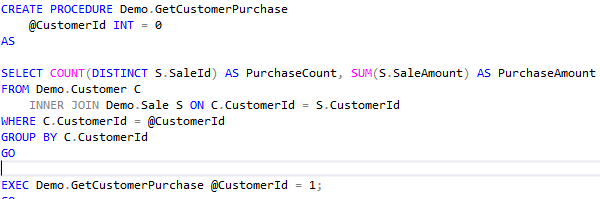


Demo.GetCustomerInformation is a question-type query. This procedure allows the user to search a customer and proceeds to retrieve all information about a customer.



This table shows customer information based on searching customerId =1.

Demo.GetCustomerPurchase is a question-type query. This retrieves a customer’s purchase count and purchase amount based upon a customer ID.

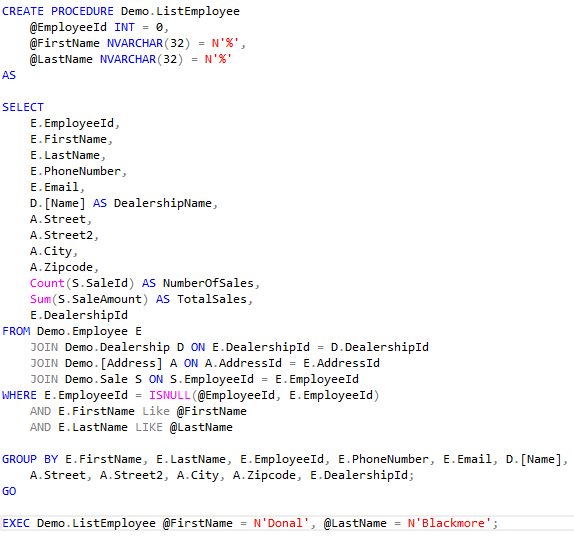


This table shows purchase information based on searching customerId =1.



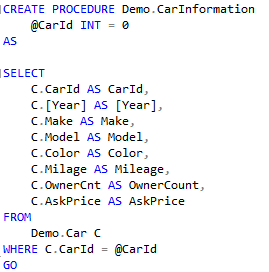
Demo.ListEmployee is a question-type query. This procedure allows the user to search an employee and provides information about that employee.

This table shows the employee information of EmployeeId = 2.

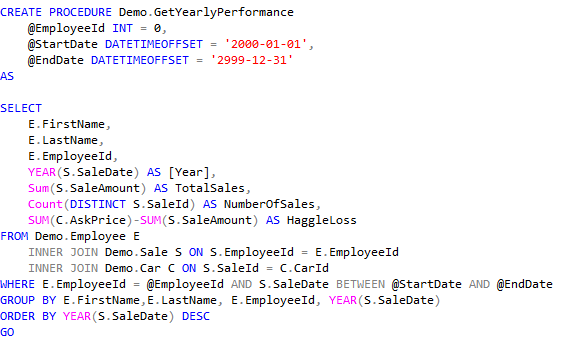


Demo.CarInformation is a question type query. This procedure allows the user to search for the details associated with a car.

This table shows the car information of CarId = 2.

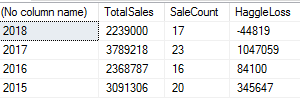
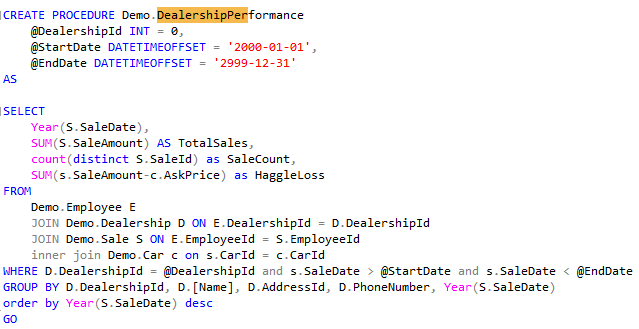


Demo.GetYearlyPerformance is a report type query. This procedure combines all the sales an employee has made in a data range and returns the yearly employee information of sales count, sales amount, and money lost on Ask vs Sale price. (Similar query for monthly results)



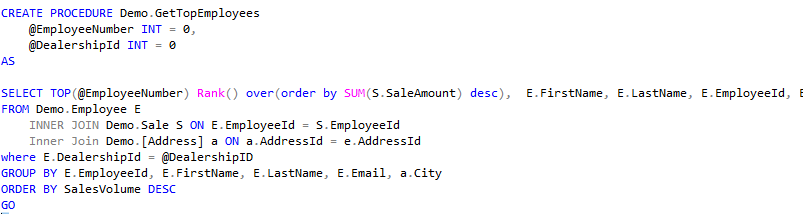
This table shows the employee performance of EmployeeId = 2, 2000-2018.

Demo.DealershipPerformance is a report type query. This procedure combines all the sales a dealership has made in a date range and returns the yearly information of sales count, sales amount, and money lost on Ask vs Sale price. (Similar query for monthly results)



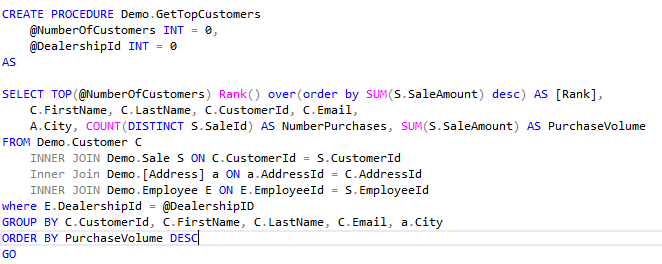
This table shows the dealership performance of Dealerhip = 2, 2000-2018.

Demo.GetTopEmployees is a report type query. This procedure combines all the sales of employees at a dealership and returns the top X employees.



This table shows the top 3 employees from dealership 2.

Demo.GetTopCustomers is a report type query. This procedure combines all of the purchases of customers at a dealership and returns the top X employees.



This table shows the top 3 customers from dealership 1.