INF-2700: Database Systems

Assignment 1

Magnus Lyngra
UiT id: mly004@uit.no
Git user: OpusMag
Exam id: i

September 24, 2023

1 Database schema

CREATE TABLE Customers (customerNumber INTEGER PRIMARY KEY, customerName TEXT NOT NULL, contactLastName TEXT NOT NULL, contactFirstName TEXT NOT NULL, phone TEXT NOT NULL, addressLine1 TEXT NOT NULL, addressLine2 TEXT NULL, city TEXT NOT NULL, state TEXT NULL, postalCode TEXT NULL, country TEXT NOT NULL, salesRepEmployeeNumber INTEGER NULL, creditLimit REAL NULL)

CREATE TABLE Employees (employeeNumber INTEGER PRIMARY KEY, lastName TEXT NOT NULL, firstName TEXT NOT NULL, extension TEXT NOT NULL, email TEXT NOT NULL, officeCode TEXT NOT NULL, reportsTo INTEGER NULL, jobTitle TEXT NOT NULL, FOREIGN KEY (reportsTo) REFERENCES Employees(employeeNumber))

CREATE TABLE Offices (officeCode TEXT PRIMARY KEY, city TEXT NOT NULL, phone TEXT NOT NULL, addressLine1 TEXT NOT NULL, addressLine2 TEXT NULL, state TEXT NULL, country TEXT NOT NULL, postalCode TEXT NOT NULL, territory TEXT NOT NULL)

CREATE TABLE OrderDetails (orderNumber INTEGER NOT NULL, productCode TEXT NOT NULL, quantityOrdered INTEGER NOT NULL, priceEach REAL NOT NULL, orderLineNumber INTEGER NOT NULL, PRIMARY KEY (orderNumber, productCode), FOREIGN KEY (productCode) REFERENCES Products)

CREATE TABLE Orders (orderNumber INTEGER PRIMARY KEY, orderDate TEXT NOT NULL, requiredDate TEXT NOT NULL, shippedDate TEXT NULL, status TEXT NOT NULL, comments TEXT NULL, customerNumber INTEGER NOT NULL, FOREIGN KEY (customerNumber) REFERENCES Customers)

CREATE TABLE Payments (customerNumber INTEGER NOT NULL, checkNumber TEXT NOT NULL, paymentDate TEXT NOT NULL, amount REAL NOT NULL, PRIMARY KEY (customerNumber, checkNumber), FOREIGN KEY (customerNumber) REFERENCES Customers)

CREATE TABLE ProductLines(productLine TEXT PRIMARY KEY, description TEXT NULL)

CREATE TABLE Products (productCode TEXT PRIMARY KEY, productName TEXT NOT NULL, productLine TEXT NOT NULL, productScale TEXT NOT NULL, productVendor TEXT NOT NULL, productDescription TEXT NOT NULL, quantityInStock INTEGER NOT NULL, buyPrice REAL NOT NULL, MSRP REAL NOT NULL, FOREIGN KEY (productLine) REFERENCES Productlines)

The aforementioned 8 CREATE TABLES statements form the database schema for the inf2700orders database.

2 SQL queries

[2]
a) SELECT customerName, contactLastName, contactFirstName FROM Customers;

This query returns customers names and their contact's first and last name from the Customers table. This produces a list of with the names of all customers and their contact's first and last name.

b) SELECT * FROM Orders WHERE shippedDate IS NULL;

This query returns all data from the table Orders, but only for orders that have not shipped yet.

c) SELECT C.customerName AS Customer, SUM(OD.quantityOrdered) AS Total FROM Orders O, Customers C, OrderDetails OD WHERE O.customerNumber = C.customerNumber AND O.orderNumber = OD.orderNumber GROUP BY O.customerNumber ORDER BY Total DESC; 2

This query returns the names of all customers and the sum of how much they have ordered. It also cross-references the customernumbers from the table Orders and Customers and cross-references the ordernumber from the table Orders and OrderDetails and then groups them by the ordernumber and sorts them in descending order by the sum they have ordered.

d) SELECT P.productName, T.totalQuantityOrdered FROM Products P NATURAL JOIN (SELECT productCode, SUM(quantityOrdered) AS totalQuantityOrdered FROM OrderDetails GROUP BY productCode) AS T WHERE T.totalQuantityOrdered ¿= 1000;

This query returns the product names from the Products table and then the sum of products ordered by using a natural join to get the sum of products ordered from the OrderDetails table and group it by the productcode. At last it limits what is to by returned by setting a threshold of a 1000 or more units for the sum of the products ordered.

3 C code

[1]
This first piece of code opens the database then checks if the operation was successful. If it wasn't it prints an error and closes the database.

```
sqlite3_open(testdb_file_name, &db);
    if (sqlite3_open(testdb_file_name, &db) != SQLITE_OK) {
      printf("Failed to open database: %s\n", sqlite3_errmsg(db));
      sqlite3_close(db);
      return 0;
    }
```

[3] This next piece of code is a function created to run the generated query. The function opens an in-memory SQLite database using sqlite3open. If the database can't be opened, an error message is given and 0 is returned. If the database is successfully opened, the function gets the query string connected to the query name using the getquerystr function. If the query string can't be found, an error message is given and 0 is returned. The function then prepares the query using sqlite3preparev2. If there is an error preparing the query an error message is given and 0 is returned. If the query is successfully prepared, the function executes the query using sqlite3step. If there is an error executing the query an error message is given and 0 is returned.

The function then iterates over the rows of the query result using a while loop. For each row, the function copies the first column of the result into the query result variable using sqlite3columntext. If there are multiple rows, the function separates them with a newline character.
If there is an error executing the query, an error message is given and 0 is returned. If the query is successfully executed, the function finalizes the query using sqlite3finalize, closes the database using sqlite3close, and returns 1. This logic does not function correctly at the moment. Running the tests results in empty output where an output is expected, which indicates that the query result is not stored properly in the query$_{r}esvariableasexpected.Severaldifferentimplementationsofthishasbeenattemptedinordertosuccessfullystorethequer$

```
int run_query(char const* const query_name, char* const query_res, const size_t res_len, char* const qu

  sqlite3 *db;
  sqlite3_stmt *res;

  int rc = sqlite3_open(":memory:", &db);

  if (rc != SQLITE_OK) {
    snprintf(query_err, err_len, "Cannot open database: %s\n", sqlite3_errmsg(db));
    sqlite3_close(db);
    return 0;
  }

  char const* query_p = get_query_str(query_name);
  if (!query_p) {
    snprintf(query_err, err_len, "test \"%s\" not found.", query_name);
    sqlite3_close(db);
    return 0;
  }

  rc = sqlite3_prepare_v2(db, query_p, -1, &res, 0);

  if (rc != SQLITE_OK) {
    snprintf(query_err, err_len, "Error preparing query: %s\n", sqlite3_errmsg(db));
    sqlite3_close(db);
    return 0;
  }

  rc = sqlite3_step(res);

  if (rc != SQLITE_DONE && rc != SQLITE_ROW) {
    snprintf(query_err, err_len, "Error executing query: %s\n", sqlite3_errmsg(db));
    sqlite3_finalize(res);
    sqlite3_close(db);
    return 0;
  }

  if (rc == SQLITE_ROW) {
    strncpy(query_res, (char*)sqlite3_column_text(res, 0), res_len);
    printf("%s\n", query_res);
  }

  sqlite3_finalize(res);
  sqlite3_close(db);

  return 1;
}
```

This final piece of code simply closes the database.

```
sqlite3_close(db);
```

4 Sources

References

[1] [SQLite C] SQLite C. An Introductory SQL Tutorial. Retrieved 16:50, September 16, 2023, from `https://zetcode.com/db/sqlitec/`

[2] [SQL Tutorial Introduction] Rachel Leist. (2022, March 21). An Introductory SQL Tutorial. Retrieved 12:40, September 24, 2023, from `https://blog.hubspot.com/marketing/sql-tutorial-introduction`

[3] [column blob] Result Values From a Query. Retrieved 19:10, September 24, 2023, from `https://www.sqlite.org/c3ref/column_blob.html`