CHAPTER 5

Advanced SQL

Practice Exercises

5.1 Consider the following relations for a company database:

- *emp* (*ename*, *dname*, *salary*)
- *mgr* (*ename*, *mname*)

and the Java code in Figure 5.20, which uses the JDBC API. Assume that the userid, password, machine name, etc. are all okay. Describe in concise English what the Java program does. (That is, produce an English sentence like "It finds the manager of the toy department," not a line-by-line description of what each Java statement does.)

5.2 Write a Java method using JDBC metadata features that takes a ResultSet as an input parameter and prints out the result in tabular form, with appropriate names as column headings.

5.3 Suppose that we wish to find all courses that must be taken before some given course. That means finding not only the prerequisites of that course, but prerequisites of prerequisites, and so on. Write a complete Java program using JDBC that:

- Takes a *course_id* value from the keyboard.
- Finds prerequisites of that course using an SQL query submitted via JDBC.
- For each course returned, finds its prerequisites and continues this process iteratively until no new prerequisite courses are found.
- Prints out the result.

For this exercise, do not use a recursive SQL query, but rather use the iterative approach described previously. A well-developed solution will be robust to the error case where a university has accidentally created a cycle of prerequisites

```
import java.sql.*;
public class Mystery {
  public static void main(String[] args) {
      try (
          Connection con=DriverManager.getConnection(
              "jdbc:oracle:thin:star/X@//edgar.cse.lehigh.edu:1521/XE");
          q = "select mname from mgr where ename = ?";
          PreparedStatement stmt=con.prepareStatement();
      )
      {
          String q;
          String empName = "dog";
          boolean more;
          ResultSet result;
          do {
              stmt.setString(1, empName);
              result = stmt.executeQuery(q);
              more = result.next();
              if (more) {
                  empName = result.getString("mname");
                  System.out.println (empName);
              }
          } while (more);
          s.close();
          con.close();
      }
      catch(Exception e){
          e.printStackTrace();
      }
  }
}
```

Figure 5.20 Java code for Exercise 5.1 (using Oracle JDBC).

(that is, for example, course A is a prerequisite for course B, course B is a prerequisite for course C, and course C is a prerequisite for course A).

5.4 Describe the circumstances in which you would choose to use embedded SQL rather than SQL alone or only a general-purpose programming language.

5.5 Show how to enforce the constraint "an instructor cannot teach two different sections in a semester in the same time slot." using a trigger (remember that the

branch (*branch_name*, *branch_city, assets*)
customer (*customer_name*, *customer_street, cust omer_city*)
loan (*loan_number*, *branch_name, amount*)
borrower (*customer_name*, *loan_number*)
account (*account_number*, *branch_name, balance*)
depositor (*customer_name*, *account_number*)

Figure 5.21 Banking database for Exercise 5.6.

constraint can be violated by changes to the *teaches* relation as well as to the *section* relation).

5.6 Consider the bank database of Figure 5.21. Let us define a view *branch_cust* as follows:

```
create view branch_cust as
    select branch_name, customer_name
    from depositor, account
    where depositor.account_number = account.account_number
```

Suppose that the view is *materialized*; that is, the view is computed and stored. Write triggers to *maintain* the view, that is, to keep it up-to-date on insertions to *depositor* or *account*. It is not necessary to handle deletions or updates. Note that, for simplicity, we have not required the elimination of duplicates.

5.7 Consider the bank database of Figure 5.21. Write an SQL trigger to carry out the following action: On **delete** of an account, for each customer-owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the *depositor* relation.

5.8 Given a relation $S(student, subject, marks)$, write a query to find the top 10 students by total marks, by using SQL ranking. Include all students tied for the final spot in the ranking, even if that results in more than 10 total students.

5.9 Given a relation *nyse*(*year, month, day, shares_traded, dollar_volume*) with trading data from the New York Stock Exchange, list each trading day in order of number of shares traded, and show each day's rank.

5.10 Using the relation from Exercise 5.9, write an SQL query to generate a report showing the number of shares traded, number of trades, and total dollar volume broken down by year, each month of each year, and each trading day.

5.11 Show how to express **group by cube**(a, b, c, d) using **rollup**; your answer should have only one **group by** clause.