CHAPTER 4

Intermediate SQL

Practice Exercises

4.1 Consider the following SQL query that seeks to find a list of titles of all courses taught in Spring 2017 along with the name of the instructor.

select name, title
from *instructor* **natural join** *teaches* **natural join** *section* **natural join** *course*
where *semester* = 'Spring' **and** year = 2017

What is wrong with this query?

4.2 Write the following queries in SQL:

a. Display a list of all instructors, showing each instructor's ID and the number of sections taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outer join, and should not use subqueries.

b. Write the same query as in part a, but using a scalar subquery and not using outer join.

c. Display the list of all course sections offered in Spring 2018, along with the ID and name of each instructor teaching the section. If a section has more than one instructor, that section should appear as many times in the result as it has instructors. If a section does not have any instructor, it should still appear in the result with the instructor name set to "—".

d. Display the list of all departments, with the total number of instructors in each department, without using subqueries. Make sure to show departments that have no instructors, and list those departments with an instructor count of zero.

4.3 Outer join expressions can be computed in SQL without using the SQL **outer join** operation. To illustrate this fact, show how to rewrite each of the following SQL queries without using the **outer join** expression.

a. **select** * **from** *student* **natural left outer join** *takes*

b. **select** * **from** *student* **natural full outer join** *takes*

4.4 Suppose we have three relations $r(A, B)$, $s(B, C)$, and $t(B, D)$, with all attributes declared as **not null**.

a. Give instances of relations r, s, and t such that in the result of
(r **natural left outer join** s) **natural left outer join** t
attribute C has a null value but attribute D has a non-null value.

b. Are there instances of r, s, and t such that the result of
r **natural left outer join** (s **natural left outer join** t)
has a null value for C but a non-null value for D? Explain why or why not.

4.5 **Testing SQL queries**: To test if a query specified in English has been correctly written in SQL, the SQL query is typically executed on multiple test databases, and a human checks if the SQL query result on each test database matches the intention of the specification in English.

a. In Section 4.1.1 we saw an example of an erroneous SQL query which was intended to find which courses had been taught by each instructor; the query computed the natural join of *instructor*, *teaches*, and *course*, and as a result it unintentionally equated the *dept_name* attribute of *instructor* and *course*. Give an example of a dataset that would help catch this particular error.

b. When creating test databases, it is important to create tuples in referenced relations that do not have any matching tuple in the referencing relation for each foreign key. Explain why, using an example query on the university database.

c. When creating test databases, it is important to create tuples with null values for foreign-key attributes, provided the attribute is nullable (SQL allows foreign-key attributes to take on null values, as long as they are not part of the primary key and have not been declared as **not null**). Explain why, using an example query on the university database.

Hint: Use the queries from Exercise 4.2.

4.6 Show how to define the view *student_grades* (*ID, GPA*) giving the grade-point average of each student, based on the query in Exercise 3.2; recall that we used a relation *grade_points*(*grade*, *points*) to get the numeric points associated with

employee (*ID*, *person_name*, *street*, *city*)
works (*ID*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*ID*, *manager_id*)

Figure 4.12 Employee database.

a letter grade. Make sure your view definition correctly handles the case of *null* values for the *grade* attribute of the *takes* relation.

4.7 Consider the employee database of Figure 4.12. Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition.

4.8 As discussed in Section 4.4.8, we expect the constraint "an instructor cannot teach sections in two different classrooms in a semester in the same time slot" to hold.

a. Write an SQL query that returns all (*instructor*, *section*) combinations that violate this constraint.

b. Write an SQL assertion to enforce this constraint (as discussed in Section 4.4.8, current generation database systems do not support such assertions, although they are part of the SQL standard).

4.9 SQL allows a foreign-key dependency to refer to the same relation, as in the following example:

```
create table manager
    (employee_ID      char(20),
     manager_ID       char(20),
     primary key employee_ID,
     foreign key (manager_ID) references manager(employee_ID)
                                  on delete cascade )
```

Here, *employee_ID* is a key to the table *manager*, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation *manager* is deleted.

4.10 Given the relations *a*(*name, address, title*) and *b*(*name, address, salary*), show how to express *a* **natural full outer join** *b* using the **full outer-join** operation with an **on** condition rather than using the **natural join** syntax. This can be done using the **coalesce** operation. Make sure that the result relation does not contain two

copies of the attributes *name* and *address* and that the solution is correct even if some tuples in *a* and *b* have null values for attributes *name* or *address*.

4.11 Operating systems usually offer only two types of authorization control for data files: read access and write access. Why do database systems offer so many kinds of authorization?

4.12 Suppose a user wants to grant **select** access on a relation to another user. Why should the user include (or not include) the clause **granted by current role** in the **grant** statement?

4.13 Consider a view *v* whose definition references only relation *r*.

- If a user is granted **select** authorization on *v*, does that user need to have **select** authorization on *r* as well? Why or why not?
- If a user is granted **update** authorization on *v*, does that user need to have **update** authorization on *r* as well? Why or why not?
- Give an example of an **insert** operation on a view *v* to add a tuple *t* that is not visible in the result of **select** * **from** *v*. Explain your answer.