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

Figure 4.12 Employee database.

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 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.

Exercises

4.14 Consider the query

```
select course_id, semester, year, sec_id, avg (tot_cred)
from takes natural join student
where year = 2017
group by course_id, semester, year, sec_id
having count (ID) >= 2;
```

Explain why appending **natural join** *section* in the **from** clause would not change the result.

4.15 Rewrite the query

```
select *
from section natural join classroom
```

without using a natural join but instead using an inner join with a **using** condition.

4.16 Write an SQL query using the university schema to find the ID of each student who has never taken a course at the university. Do this using no subqueries and no set operations (use an outer join).

4.17 Express the following query in SQL using no subqueries and no set operations.

```
select ID
from student
except
select s_id
from advisor
where i_ID is not null
```

4.18 For the database of Figure 4.12, write a query to find the ID of each employee with no manager. Note that an employee may simply have no manager listed or may have a *null* manager. Write your query using an outer join and then write it again using no outer join at all.

4.19 Under what circumstances would the query

```
select *
from student natural full outer join takes
                    natural full outer join course
```

include tuples with null values for the *title* attribute?

4.20 Show how to define a view *tot_credits* (*year, num_credits*), giving the total number of credits taken in each year.

4.21 For the view of Exercise 4.18, explain why the database system would not allow a tuple to be inserted into the database through this view.

4.22 Show how to express the **coalesce** function using the **case** construct.

4.23 Explain why, when a manager, say Satoshi, grants an authorization, the grant should be done by the manager role, rather than by the user Satoshi.

4.24 Suppose user A, who has all authorization privileges on a relation r, grants **select** on relation r to **public** with grant option. Suppose user B then grants **select** on r to A. Does this cause a cycle in the authorization graph? Explain why.

4.25 Suppose a user creates a new relation $r1$ with a foreign key referencing another relation $r2$. What authorization privilege does the user need on $r2$? Why should this not simply be allowed without any such authorization?

4.26 Explain the difference between integrity constraints and authorization constraints.

Further Reading

General SQL references were provided in Chapter 3. As noted earlier, many systems implement features in a non-standard manner, and, for that reason, a reference specific to the database system you are using is an essential guide. Most vendors also provide extensive support on the web.

The rules used by SQL to determine the updatability of a view, and how updates are reflected on the underlying database relations appeared in SQL:1999 and are summarized in [Melton and Simon (2001)].

The original SQL proposals for assertions date back to [Astrahan et al. (1976)], [Chamberlin et al. (1976)], and [Chamberlin et al. (1981)].

Bibliography

[Astrahan et al. (1976)] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson, "System R, A Relational Approach to Data Base