• Assertions
• Date and time types
• Default values
• Indices
• Large objects
• User-defined types
• Domains
• Catalogs
• Schemas
• Authorization
• Privileges
  ◦ select
  ◦ insert
  ◦ update
  ◦ all privileges
  ◦ Granting of privileges
  ◦ Revoking of privileges
  ◦ Privilege to grant privileges
  ◦ Grant option
• Roles
• Authorization on views
• Execute authorization
• Invoker privileges
• Row-level authorization

Practice Exercises

4.1 Write the following queries in SQL:

a. Display a list of all instructors, showing their ID, name, and the number of sections that they have taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outerjoin, and should not use scalar subqueries.

b. Write the same query as above, but using a scalar subquery, without outerjoin.

c. Display the list of all course sections offered in Spring 2010, along with the names of the instructors teaching the section. If a section has more than one instructor, it should appear as many times in the result as it has instructors. If it 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 scalar subqueries. Make sure to correctly handle departments with no instructors.

4.2 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
Practicing Exercises 153

4.3 Suppose we have three relations \( r(A, B) \), \( s(B, C) \), and \( t(B, D) \), with all attributes declared as not null. Consider the expressions

- \( r \) natural left outer join \( (s \) natural left outer join \( t) \), and
- \( (r \) natural left outer join \( s) \) natural left outer join \( t \)

a. Give instances of relations \( r \), \( s \) and \( t \) such that in the result of the second expression, attribute \( C \) has a null value but attribute \( D \) has a non-null value.

b. Is the above pattern, with \( C \) null and \( D \) not null possible in the result of the first expression? Explain why or why not.

4.4 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 3.3.3 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 \( \text{instructor}, \text{teaches}, \text{and course} \), and as a result unintentionally equated the \( \text{dept_name} \) attribute of \( \text{instructor} \) and \( \text{course} \). 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 4.1.

4.5 Show how to define the view \( \text{student_grades} \) \((ID, GPA)\) giving the grade-point average of each student, based on the query in Exercise 3.2; recall that we used a relation \( \text{grade_points}(\text{grade, points}) \) to get the numeric points associated with a letter grade. Make sure your view definition correctly handles the case of null values for the \( \text{grade} \) attribute of the \( \text{takes} \) relation.

4.6 Complete the SQL DDL definition of the university database of Figure 4.8 to include the relations \( \text{student}, \text{takes}, \text{advisor}, \) and \( \text{prereq} \).
Consider the relational database of Figure 4.11. Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition.

As discussed in Section 4.4.7, 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 4.4.7, current generation database systems do not support such assertions, although they are part of the SQL standard).

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

```
create table manager
  (employee_name varchar(20) not null,
   manager_name varchar(20) not null,
   primary key employee_name,
   foreign key (manager_name) references manager
     on delete cascade )
```

Here, employee_name 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.

SQL provides an n-ary operation called coalesce, which is defined as follows: coalesce($A_1, A_2, \ldots, A_n$) returns the first nonnull $A_i$ in the list $A_1, A_2, \ldots, A_n$, and returns null if all of $A_1, A_2, \ldots, A_n$ are null.

Let $a$ and $b$ be relations with the schemas $A(name, address, title)$, and $B(name, address, salary)$, respectively. Show how to express a natural full outer join $b$ using the full outer-join operation with an on condition and 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.
Some researchers have proposed the concept of marked nulls. A marked null $\bot_i$ is equal to itself, but if $i \neq j$, then $\bot_i \neq \bot_j$. One application of marked nulls is to allow certain updates through views. Consider the view instructor info (Section 4.2). Show how you can use marked nulls to allow the insertion of the tuple (99999, “Johnson”, “Music”) through instructor info.

Exercises

4.12 For the database of Figure 4.11, write a query to find those employees 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.

4.13 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?

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

4.15 Show how to express the coalesce operation from Exercise 4.10 using the case operation.

4.16 Referential-integrity constraints as defined in this chapter involve exactly two relations. Consider a database that includes the relations shown in Figure 4.12. Suppose that we wish to require that every name that appears in address appears in either salaried worker or hourly worker, but not necessarily in both.

a. Propose a syntax for expressing such constraints.

b. Discuss the actions that the system must take to enforce a constraint of this form.

4.17 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.
4.18 Suppose user $A$, who has all authorizations 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.

4.19 Database systems that store each relation in a separate operating-system file may use the operating system’s authorization scheme, instead of defining a special scheme themselves. Discuss an advantage and a disadvantage of such an approach.

**Bibliographical Notes**

See the bibliographic notes of Chapter 3 for SQL reference material.

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