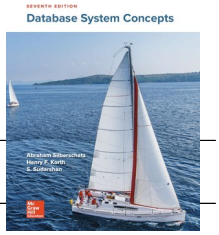
Database System Concepts

Chapter 4



REVISION 2

Terms (GNU AGPLv3)

This is **just a sample**!
I have tried to put things like **summary**, and obviously this is not the solution that I will submit or recommended! I am no way responsible for any illegal use of this file.

If you need direct solutions, please ask **DeepSeek, ChatGPT**, **Google Gemini**, **Anthropic Claude**, **Hugging Chat**, **Le Chat Mistral** or any other predictive model.

Chapter 4 | Practice Exercises

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?

Here instructor and course share the same attribute, named “dept_name” and as natural join is used, they will be joined only if the value is same, skipping other possible columns. So basically every teacher will be listed in his own department only.
A better approach would be to specify the join conditions explicitly to ensure only the intended columns are matched.

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.

```
select ID, count(sec_id) as Number_of_sections
from instructor natural left outer join teaches
group by ID;
```

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

```
select ID,
        (select count(*) as Number_of_sections
        from teaches T where T.id = I.id)
from instructor I;
```

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 "-".

```
select course_id, sec_id, ID,
        coalesce(name,'--') as name
from (section natural left outer join teaches)
natural left outer join instructor
where semester='Spring' and year = 2018;
```

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.

```
SELECT d.dept_name, COUNT(i.ID) AS num_instructors
FROM department d
LEFT OUTER JOIN instructor i ON d.dept_name = i.dept_name
GROUP BY d.dept_name;
```

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

```
SELECT *
FROM student
NATURAL JOIN takes
UNION
SELECT student.ID, name, dept_name, tot_cred,
    NULL, NULL, NULL, NULL, NULL
FROM student
WHERE student.ID NOT IN (SELECT ID FROM takes);
```

b. select * from student natural full outer join takes

```
SELECT *
FROM student
NATURAL JOIN takes
UNION
SELECT student.ID, name, dept_name, tot_cred,
    NULL, NULL, NULL, NULL, NULL
FROM student
WHERE ID NOT IN (SELECT ID FROM takes)
UNION
SELECT NULL, NULL, NULL, NULL,
    takes.ID, course_id, sec_id, semester, year
FROM takes
WHERE ID NOT IN (SELECT ID FROM student);
```

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.

Yes, it is possible that C has a null value, but not D. Let's consider,

r(A,B) = {(1, 1)}
s(B,C) = {(2, 1)}
t(B,D) = {(1, 3)}

And the result of the operation will be, {(1, 1, Null, 3)}

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.

This is not possilbe. If we are to join s and t first then obviously there will be a value of C, to join with the corresponding D via B. And we know that all attributes are declared as not null, from the question.

Let's consider,

r(A,B) = {(1, 1)}
s(B,C) = {(2, 1)}
t(B,D) = {(1, 3)}

And the result of the operation will be, {(1, 1, Null, Null)}
If we need D, then without matching with a C won't make sense.

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.

Let's consider a scenario,

instructor = {('12345','Gauss', 'Physics', 10000)}
teaches = {('12345', 'EE321', 1, 'Spring', 2017)}
course = {('EE321', 'Magnetism', 'Ele . Eng.', 6)}

Here it won't be combined because instructor's *dept_name* and course's *dept_name* are different, hence will produce an 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.

If we create etries in the referenced relation without any childrens, then it'll be a good test subject to check SQL queries like left join or right join or outer full join.

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.

This kind of tuple is important as in SQL, Null is not equal to Null. So values with Null value will be totally skipped in operations like,

```
SELECT * FROM teaches NATURAL JOIN instructor;
```

```
-- (will appear in results)
INSERT INTO teaches VALUES ('101', 'CS-101', '1', 'Fall', 2023);

-- (will disappear from results)
INSERT INTO teaches VALUES (NULL, 'CS-101', '2', 'Fall', 2023);
```

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.

```
create view student_grades(ID, GPA) as
select
 ID, sum(credits * points)/sum(credits) as GPA
from
 takes, course, grade_points
where
 takes.grade = grade_points.grade
and takes.course_id = course.course_id
group by ID;
```

I didn't understood how to handle the null value case. Hint :: **coalesce**.

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.

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

Figure 4.12 Employee database.

```
CREATE TABLE employee (
        ID NUMERIC(5,0),
   person_name VARCHAR(255) NOT NULL,
   street VARCHAR(255),
   city VARCHAR(255),
   PRIMARY KEY (ID)
);

CREATE TABLE company (
```

```
  company_name VARCHAR(255) NOT NULL,
  city VARCHAR(255),
  PRIMARY KEY (company_name)
);

CREATE TABLE works (
        ID NUMERIC(5,0),
  person_name VARCHAR(255) NOT NULL,
  company_name VARCHAR(255),
  salary DECIMAL(10,2),
  PRIMARY KEY (ID),
  FOREIGN KEY (ID) REFERENCES employee(ID),
  FOREIGN KEY (company_name) REFERENCES company(company_name)
);

CREATE TABLE manages (
        ID NUMERIC(5,0),
  manager_id NUMERIC(5,0),
  PRIMARY KEY (ID),
  FOREIGN KEY (ID) REFERENCES employee(ID),
  FOREIGN KEY (manager_id) REFERENCES employee(ID)
);
```

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.

```
SELECT
  instructor.ID,
  instructor.name,
  teaches.sec_id,
  teaches.semester,
  teaches.year,
  section.time_slot_id,
  COUNT(DISTINCT section.building, section.room_number) AS location_count
FROM
  instructor
  NATURAL JOIN teaches
  NATURAL JOIN section
GROUP BY
  instructor.ID,
  instructor.name,
  teaches.sec_id,
  teaches.semester,
  teaches.year,
  section.time_slot_id
HAVING
  COUNT(DISTINCT section.building, section.room_number) > 1;
```

b. Write an SQL assertion to enforce this constraint.

```
create assertion check not exists
(SELECT
   instructor.ID,
   instructor.name,
   teaches.sec_id,
   teaches.semester,
   teaches.year,
   section.time_slot_id,
   COUNT(DISTINCT section.building, section.room_number) AS location_count
FROM
   instructor
   NATURAL JOIN teaches
   NATURAL JOIN section
GROUP BY
   instructor.ID,
   instructor.name,
   teaches.sec_id,
   teaches.semester,
   teaches.year,
   section.time_slot_id
HAVING
   COUNT(DISTINCT section.building, section.room_number) > 1);
```

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.

The tuples of all employees of the manager, at all levels, get deleted as well! This deletion works recursively and thus further relations will also be deleted.

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.

```sql
SELECT
  COALESCE(a.name, b.name) AS name,
  COALESCE(a.address, b.address) AS address,
  a.title,
  b.salary
FROM
  a FULL OUTER JOIN b
    ON a.name = b.name
    AND a.address = b.address;
```

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?

Database systems offer more authorization types because:

1. Databases have more complex operations than simple file read/write (SELECT, INSERT, UPDATE, DELETE, etc.)
2. Granular control is needed for security in multi-user environments
3. Different operations have different risk profiles (e.g., SELECT vs DELETE)
4. Need to control schema modification (ALTER, REFERENCES) separately from data access
5. Support for row-level and column-level security requirements
6. Need to manage privileges for views, stored procedures, and other database objects
7. Requirement to delegate privileges (GRANT OPTION)

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?

If a user grants permission though a role then, even if the user leaves, the permission still persists. That's why it's a good practice in the long run.

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?

No, the user doesn't need SELECT on the base table (r).

• 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?

Yes, the user also needs update permission on r as well, because this update operation is performed there.

• 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.

```sql
CREATE VIEW v AS SELECT * FROM r WHERE rating > 5;
INSERT INTO v VALUES (101, 'John', 3); -- Tuple won't appear in SELECT * FROM v
```

Explanation: The inserted tuple has rating=3 which doesn't satisfy the view's condition (rating > 5), so it won't be visible in the view even though it exists in the base table.

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.

The query already groups by course_id, semester, year, and sec_id. Here sec_id is the key attributes of section. So the section relation doesn't contain any attributes that would affect the average of tot_cred.

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.

```
select * from section join classroom using (building, room_number);
```

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).

```
select s.ID
from student s left outer join takes t on s.ID = t.ID
where t.ID is null
```

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
```

```
select s.ID
from student s left outer join advisor a on s.ID = a.s_id
where a.s_id is null or a.i_ID is null
```

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.

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

Figure 4.12 Employee database.

With outer join,

```
select e.person_name
from employee e left outer join manages m on e.ID = m.ID
where m.manager_id is null;
```

Without outer join,

```
SELECT e.person_name
FROM employee e
WHERE NOT EXISTS (
  SELECT 1
  FROM manages m
  WHERE e.ID = m.ID
);
```

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?

The query would include tuples with null title values when:

1. There are students who haven't taken any courses (their takes attributes would be null)
2. There are courses that no students have taken (their student attributes would be null)

The natural join fails to match on dept_name between student and course (if student's dept_name doesn't match course's dept_name)

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

```
create view tot_credits (year, num_credits) as
select year, sum(credits)
from takes natural join course
where grade is not null and grade <> 'F'
group by year;
```

21. For the view of Exercise 4.20, explain why the database system would not allow a tuple to be inserted into the database through this view.

We will get the following error,
Error Code: 1471. The target table tot_credits of the INSERT is not insertable-into

The view is not updatable because:

1. It contains aggregation (SUM)
2. It involves a GROUP BY clause
3. It joins multiple tables

So it's really not ideal to pull update operations through view.

22. Show how to express the coalesce function using the case construct.

```
-- COALESCE(a,b,c) is equivalent to:
case when a is not null then a
   when b is not null then b
   else c
end
```

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.

If grant is done by a role then if Satoshi leaves the organization, privileges granted by their role will persist as it maintains separation between personal and role-based privileges. It allows for proper privilege inheritance and revocation.

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.

As A has all the priviledges beforehand B, B won't be able to grant select on r to A again. So it wont' create a cycle.

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?

The user needs REFERENCES privilege on r2 because,

1. The foreign key creates a dependency between the relations.
2. It prevents the referenced rows in r2 from being deleted while referenced.
3. Without this requirement, users could create constraints on data they don't own

Suppose an user creates a foreign key in a relation r1 referencing the dept name attribute of the r2 relation and then inserts a tuple into r pertaining to the Geology department. It is no longer possible to delete the Geology department from the department relation without also modifying relation r1. Thus, the definition of a foreign key by an user restricts future activity by other users; therefore, there is a need for the references privilege.

26. Explain the difference between integrity constraints and authorization constraints.

Feature	Integrity Constraints	Authorization Constraints
Definition	Rules enforced to ensure **data correctness** and **consistency** within the database.	Restrictions that control **who can access or modify** data.
Purpose	Prevent invalid or inconsistent data from being stored.	Restrict unauthorized users from performing certain actions.
Scope	Applied to **data values and relationships** between tables.	Applied to **users and their privileges** on database objects.
Enforcement	Enforced **automatically** by the database system.	Enforced **based on user roles and privileges**.
Examples	- **Primary Key** (ensures uniqueness) - **Foreign Key** (maintains referential integrity) - **Check Constraints** (enforces specific conditions) - **Not Null** (ensures presence of values)	- **GRANT/REVOKE statements** (assign/restrict permissions) - **Access control on tables/views** (read/write restrictions) - **REFERENCES privilege** (needed for foreign key creation)
Focus	**Data correctness and logical consistency.**	**User access control and security.**
Modification Impact	Prevents changes that would violate constraints.	Prevents unauthorized users from making changes.