> 2127405048 方浩楠 SQL作业

3.1

> Write the following queries in SQL, using the university schema. (We suggest you actually run these queries on a database, using the sample data that we provide on the website of the book, db-book.com. Instructions for setting up a database, and loading sample data, are provided on the above website.)
>
> a. Find the titles of courses in the Comp. Sci. department that have 3 credits.
> b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.
> c. Find the highest salary of any instructor.
> d. Find all instructors earning the highest salary (there may be more than one with the same salary).

a

```sql
SELECT title
FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

b

```sql
SELECT DISTINCT takes.id
FROM takes JOIN teaches ON takes.course_id = teaches.course_id JOIN instructor
ON teaches.id = instructor.id
WHERE instructor.name = 'Einstein';
```

c

```sql
SELECT MAX(salary)
FROM instructor;
```

d

```sql
SELECT name
FROM instructor
WHERE salary = (
    SELECT MAX(salary)
    FROM instuctor
);
```

3.9

> Consider the relational database of Figure 3.19, where the primary keys are underlined. Give an expression in SQL for each of the following queries.

> a. Find the ID, name, and city of residence of each employee who works for "First Bank Corporation".
> b. Find the ID, name, and city of residence of each employee who works for "First Bank Corporation" and earns more than $10000.
> c. Find the ID of each employee who does not work for "First Bank Corporation".
> d. Find the ID of each employee who earns more than every employee of "Small Bank Corporation".
> e. Assume that companies may be located in several cities. Find the name of each company that is located in every city in which "Small Bank Corporation" is located.
> f. Find the name of the company that has the most employees (or companies, in the case where there is a tie for the most).
> g. Find the name of each company whose employees earn a higher salary, on average, than the average salary at "First Bank Corporation".

a

```sql
SELECT e.ID, e.person_name, city
FROM employee AS e JOIN works AS w ON e.ID = w.ID
WHERE w.company_name = 'First Bank Corporation';
```

b

```sql
SELECT e.ID, e.person_name, city
FROM employee AS e JOIN works AS w ON e.ID = w.ID
WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000;
```

c

```sql
SELECT ID
FROM works
WHERE company_name <> 'First Bank Corporation'
```

d

```sql
SELECT ID
FROM works
WHERE salary >  (
    SELECT MAX(salary)
    FROM works
    WHERE company_name = 'Small Bank Corporation'
);
```

e

```sql
SELECT S.company_name
FROM company AS S
WHERE NOT EXISTS (
    (
        SELECT city
        FROM company
```

```
        WHERE company_name = 'Small Bank Corporation'
    )
    EXCEPT
    (
        SELECT city
        FROM company AS T
        WHERE T.company_name = S.company_name
    )
);
```

f

```
SELECT company_name
FROM works
GROUP BY company_name
HAVING COUNT(DISTINCT ID) >= ALL (
    SELECT COUNT(DISTINCT ID)
    FROM works
    GROUP BY company_name
)
```

g

```
SELECT company_name
FROM works
GROUP BY company_name
HAVING AVG(salary) >  (
    SELECT AVG(salary)
    FROM works
    WHERE company_name = 'First Bank Corporation'
);
```

3.16

> Consider the employee database of Figure 3.19, where the primary keys are underlined. Given an expression in SQL for each of the following queries.
>
> a. Find ID and name of each employee who lives in the same city as the location of the company for which the employee works.
> b. Find ID and name of each employee who lives in the same city and on the same street as does her or his manager.
> c. Find ID and name of each employee who earns more than the average salary of all employees of her or his company.
> d. Find the company that has the smallest payroll.

a

```
SELECT e.ID, e.person_name
FROM employee AS e JOIN works AS w ON e.ID = w.ID JOIN company AS c ON
w.company_name = c.company_name
WHERE e.city = c.city;
```

b

```
SELECT
FROM employee AS e JOIN manager AS m on e.ID = m.ID JOIN employee AS m_of_e ON m.manager_ID = mr_of_e.ID
WHERE e.street = m_of_e.street AND e.city = m_of_e.street;
```

c

```
WITH average_salary_per_company(company_name, avg_salary) AS (
    SELECT company_name, AVG(salary)
    FROM works
    GROUP BY company_name
)
SELECT e.id, e.person_name
FROM employee AS e INNER JOIN works ON e.id = works.id
WHERE works.salary > (
    SELECT avg_salary
    FROM average_salary_per_company
    WHERE company_name = works.company_name
);
```

d

```
SELECT TOP 1 company_name, SUM(salary) AS total_payroll
FROM works
GROUP BY company_name
ORDER BY total_payroll ASC;
```

3.17

> Consider the employee database of Figure 3.19. Give an expression in SQL for each of the following queries.
>
> a. Give all employees for "First Bank Corporation" a 10 percent raise.
> b. Give all managers of "First Bank Corporation" a 10 percent raise.
> c. Delete all tuples in the *works* relation for employees of "Small Bank Corporation".

a

```
UPDATE works
SET salary = salary * 1.1
WHERE company_name = 'First Bank Corporation';
```

b

```sql
UPDATE works
SET salary = salary * 1.1
WHERE company_name = 'First Bank Corporation' AND id IN (
    SELECT manager_id
    FROM manages
);
```

c

```sql
DELETE FROM works
WHERE company_name = 'Small Bank Corporation';
```

3.21

> Consider the library database of Figure 3.20. Write the following queries in SQL.
>
> a. Find the member number and name of each member who has borrowed at least one book published by "McGraw-Hill".
> b. Find the member number and name of each member who has borrowed every book published by "McGraw-Hill".
> c. For each publisher, find the member number and name of each member who has borrowed more than five books of that publisher.
> d. Find the average number of books borrowed per member. Take into account that if a member does not borrow any books, then that member does not appear in the *borrowed* relation at all, but that member still counts in the average.

a

```sql
SELECT memb_no
FROM member AS m
WHERE EXISTS (
    SELECT 1
    FROM book JOIN borrowed ON book.isbn = borrowed.isbn
    WHERE book.publisher = 'McGraw-Hill' AND borrowed.memb_no = m.memb_no
)
```

b

```sql
SELECT memb_no, name
FROM member AS m
WHERE NOT EXISTS (
    (
        SELECT isbn
        FROM book
        WHERE publisher = 'McGraw-Hill'
    )
    EXCEPT
    (
        SELECT isbn
```

```
        FROM borrowed
        WHERE memb_no = m.memb_no
    )
)
```

c

```
WITH member_borrowed_book AS (
    SELECT member.memb_no, name, book.isbn, title, authors, publisher, date
    FROM member INNER JOIN borrowed ON member.memb_no = borrowed.memb_no
                INNER JOIN book ON borrowed.isbn = book.isbn
)
SELECT memb_no, memb_name, publisher, COUNT(isbn)
FROM member_borrowed_book
GROUP BY memb_no, memb_name, publisher
HAVING COUNT(isbn) > 5;
```

d

```
WITH number_of_books_borrowed AS (
    SELECT memb_no, name, (
        CASE
            WHEN NOT EXISTS (SELECT * FROM borrowed WHERE borrowed.memb_no =
member.memb_no) THEN 0
            ELSE (SELECT COUNT(*) FROM borrowed WHERE borrowed.memb_no =
member.memb_no)
        END
    )
    FROM member
)
SELECT AVG(number_of_books) AS average_number_of_books_borrowed_per_member
FROM number_of_books_borrowed
```