Databse Homework 3

> 10185101210 陈俊潼

Chapter 3

3.11

a.

```
SELECT DISTINCT name FROM (student NATUTAL JOIN takes) JOIN course
USING(course_id) WHERE course.dept_name='Comp. Sci.';
```

b.

```
SELECT ID, name FROM student WHERE ID not in (SELECT ID FROM student
NATURAL JOIN takes WHERE (year < 2009 or (year = 2009 and
semester='Spring')));
```

c.

```
SELECT name, salary FROM instructor NATURAL  JOIN (SELECT MAX(salary) as
salary FROM instructor GROUP BY dept_name) as T;
```

d.

```
SELECT min(salary) as salary FROM
(SELECT name, salary FROM instructor NATURAL JOIN (SELECT MAX(salary) as
salary FROM instructor GROUP BY dept_name) as T
) as TT;
```

3.12

a.

```
INSERT INTO course (course_id, title, dept_name, credits) VALUES ('CS-
001','Weekly Seminar','Comp. Sci.',0);
```

b.

```
INSERT INTO section (course_id, sec_id, semester, year)VALUES('CS-001', 1,
'Autumn', 2009);
```

c.

```
INSERT INTO takes (id, course_id, sec_id, semester, year) (SELECT ID, 'CS-
001', 1, "Autumn", 2009 FROM student WHERE student.dept_name = 'Comp.
Sci.');
```

d.

```
DELETE FROM takes WHERE course_id='CS-001' AND sec_id = 1 AND year = 2009
AND semester = 'Autumn' AND id = (SELECT id FROM student WHERE name =
'Chavez');
```

e.

```
DELETE FROM course WHERE course_id='CS-001';
```

If we don't first delete all the section, there will be a foreign key violation and the operation will fail.

f.

```
DELETE FROM takes WHERE course_id in (SELECT course_id FROM course WHERE
title LIKE '%database%');
```

3.13

```
CREATE DATABASE insurance;

CREATE TABLE person(
    driver_id varchar(10) NOT NULL,
    name varchar(32) NOT NULL,
    address varchar(64) NOT NULL,
    PRIMARY KEY (driver_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE car(
    license varchar(10) NOT NULL,
  model varchar(20) NOT NULL,
  year varchar(4) NOT NUL,
  PRIMARY KEY (license)
);

CREATE TABLE accident(
    report_number varchar(10)   NOT NULL,
  date datetime NOT NULL,
  location varchar(20) NOT NUL,
  PRIMARY KEY (report_number)
);

CREATE TABLE owns(
    driver_id varchar(10)   NOT NULL,
  license varchar(10) NOT NULL,
  PRIMARY KEY (driver_id, license),
  FOREIGN KEY (driver id) REFERENCES person
    FOREIGN KEY (license) REFERENCES car
);

CREATE TABLE participated(
    report_number varchar(10)   NOT NULL,
  license varchar(10) NOT NULL,
  driver_id varchar(10) NOT NULL,
  damage_amount varchar(10) NOT NULL,
  PRIMARY KEY (report_number),
  FOREIGN KEY (license) REFERENCES car
);
```

3.14

a.

```
SELECT count(report_number) FROM (SELECT * FROM accident NATURAL JOIN
participated NATURAL JOIN person WHERE name = 'John Smith');
```

b.

```
UPDATE participate SET damage_amount = '$3000' WHERE license = 'AABB2000'
and report_number = 'AR2197';
```

3.15

a.**TODO**

```
SELECT
    customer_name
FROM
    customer AS C
WHERE (SELECT count(*) FROM branch WHERE branch_city='Brooklyn') = (SELECT
count(*) FROM (customer NATURAL JOIN depositor NATURAL JOIN account) AS D
WHERE branch.branch_city = 'Brooklyn' AND D.customer_name =
C.customer_name)
```

b.

```
SELECT sum(amount) FROM loan;
```

c.

```
SELECT branch_name FROM branch WHEN assets > SOME (SELECT assets FROM
branch WHERE branch_city = 'Brooklyn');
```

3.16

a.

```
SELECT employee_name FROM works WHERE company_name = 'First Bank';
```

b.

```
SELECT employee.employee_name FROM employee NATURAL JOIN works NATURAL
JOIN companny WHERE employee.city = company.city;
```

c.

```
SELECT employee.employee_name FROM employee e1, employee e2, manages m
WHERE e1.street = e2.street and e1.city = e2.city;
```

d.

```
SELECT employee_name FROM works AS WA WHERE salary > (SELECT avg(salary)
FROM works AS WB WHERE WA.company_name = WB.company_mame);
```

e.

```
SELECT company_name, min(payroll) FROM (SELECT company_name sum(salary) AS
payroll FROM works GROUP BY company_name) AS T;
```

3.17

a.

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

b.

```
UPDATE works SET salary = salary * 1.1 WHERE employee_name IN (SELECT
mannager_name FROM manages M NATURAL JOIN works  W WHERE W.company_name =
'First Bank Corporation') AS T;
```

c.

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

3.18

1. Sometimes the value of one attribute in a tuple can't be asured and is unknown.
2. It can stand for a value that's not exist.

3.19

S <> all P means S doesn't equal to eny element in P, S not in P means S is not a subset of P, which means for every element in P, it doesn't equal to S. So they are idetical.

3.20

```
CREATE TABLE employee(
    employee_name varcar(32) NOT NULL,
  street varchar(64) NOT NULL,
  city varchar(32) NOT NULL,
  PRIMARY KEY (employee_name),
);

CREATE TABLE works(
    employee_name varcar(32) NOT NULL,
  company_name varchar(64) NOT NULL,
  salary varchar(32) NOT NULL,
  PRIMARY KEY (employee_name),
  FOREIGN KEY (employee_name) REFERENCES employee,
  FOREIGN KEY (company_name) REFERENCES company
);
```

```
CREATE TABLE company(
    company_name varcar(32) NOT NULL,
  city varchar(32) NOT NULL,
  PRIMARY KEY (company_name),
);

CREATE TABLE manages(
    employee_name varcar(32) NOT NULL,
  manager_name varcar(32) NOT NULL,
  PRIMARY KEY (employee_name),
);
```

3.21

a.

```
SELECT DISTINCT name FROM (member NATURAL JOIN book NATURAL JOIN borrowed)
WHERE publisher = `McGraw-Hill`
```

b.

```
SELECT DISTINCT name FROM member AS M WHERE NOT EXISTS
    (SELECT isbn FROM book WHERE
    (book.publisher = 'McGraw-Hill' AND isbn NOT IN (SELECT isbn FROM
borrowed WHERE borrowed.memb_no = M.memb_no));
```

c.

```
SELECT publisher, name FROM (SELECT name, publisher, count(isbn) AS
counter FROM book NATURAL JOIN borrowed NATURAL JOIN member GROUP BY
publisher, name WHERE counter > 5);
```

d.

```
SELECT name, avg(num) FROM (SELECT name, sum(isbn) AS num FROM member AS M
NATURAL JOIN borrowed GOUPE BY name) WHERE (SELECT count(*) FROM borrowed
AS B WHERE B.memb_no = M.memb_no) > 0;
```

3.22

```
WHERE (SELECT count(title) FROM course) = (SELECT count(DISTINCT title)
FROM course);
```

3.23

The only attribute that is same between *section* and *course* is `course_id`. By joining section table, there won't be new rows in the result. Besides, since the attribute `semester` and `year` in section is identical to takes, so the result wont be affected.

3.24

```
SELECT DISTINCT dept_name FROM instructor AS I WHERE (SELECT sum(salary)
FROM instructor AS T WHERE dept_name = I.dept_name) >= (SELECT avg(salary)
FROM instructor AS TT);
```

Chapter 4

4.12

```
SELECT employee_name FROM employee LEFT OUTER JOIN manages ON
employee.employee_name = manages.employee_name WHERE manager_name is NULL;

SELECT employee_name FROM employee AS E WHERE NOT EXISTS (SELECT
employee_name FROM manages AS M WHERE E.employee_name = M.employee_name
AND M.manager_name IS NOT NULL);
```

4.13

If a student didn't take any course, there will be NULL value for the title attribute.

Another situation is that the title of the course if NULL itself.

4.14

```
CREATE VIEW tot_credits(year, nuum_credits) AS (SELECT year, SUM(credits)
AS num_credits FROM course NATURAL JOIN takes GROUP BY year);
```

4.15

```
SELECT
    WHEN A1 IS NOT NULL THEN A1
    WHEN A2 IS NOT NULL THEN A2
    WHEN A3 IS NOT NULL THEN A3
    ...
    ELSE NULL;
```

4.16

a.

```
CONSTRAINT check_exists CHECK (salaried_worker.name IN address.name OR
salaried_worker.name IN hourly_worker.name)
```

b.

Every time when user tries to insert or edit data in *salaried_worder*, the system should check if the name exists in table *hourly_worker* or *address*. If the requirements aren't satisfied, the system should reject the request.