4.7

```
CREATE TABLE employee (
    ID INT PRIMARY KEY,
    person_name VARCHAR(255),
    street VARCHAR(255),
    city VARCHAR(255)
);

CREATE TABLE company (
    company_name VARCHAR(255) PRIMARY KEY,
    city VARCHAR(255)
);

CREATE TABLE works (
    ID INT PRIMARY KEY,
    company_name VARCHAR(255),
    salary DECIMAL(10,2),
    FOREIGN KEY (ID) REFERENCES employee(ID) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (company_name) REFERENCES company(company_name) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE manages (
    ID INT PRIMARY KEY,
    manager_id INT,
    FOREIGN KEY (ID) REFERENCES employee(ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

4.9

删除操作的具体过程：

1. **初始删除**：
   - 假设删除 `employee_ID = 'E1'` 的员工（假设 `E1` 是一个经理）。
   - 数据库会首先删除 `employee_ID = 'E1'` 的元组。
2. **级联删除**：
   - 由于 `manager_ID` 引用了 `employee_ID`，所有 `manager_ID = 'E1'` 的员工（即 `E1` 的直接下属）也会被删除。
   - 如果这些被删除的员工本身也是经理，那么他们的下属（即第二级员工）也会被删除。
   - 这个过程会一直持续，直到所有直接和间接依赖于 `E1` 的员工都被删除。

4.12

1. 包含 `GRANTED BY CURRENT_ROLE` 的情况：

- 当使用 `GRANTED BY CURRENT_ROLE` 时，权限的授予是基于当前角色的权限。
- 这意味着授予的权限与当前角色的权限相关联，而不是与执行 `GRANT` 语句的特定用户相关联。
- **长期效果**：
  - 如果执行 `GRANT` 语句的用户离开或账户被终止，授予的权限仍然有效，因为这些权限是基于角色的。
  - 这种方式更适合企业环境，因为它确保了权限管理的连续性和一致性。

2. 不包含 `GRANTED BY CURRENT_ROLE` 的情况：

- 如果不使用 `GRANTED BY CURRENT_ROLE`，权限的授予是基于执行 `GRANT` 语句的特定用户的权限。
- **长期效果**：
  - 如果执行 `GRANT` 语句的用户离开或账户被终止，授予的权限可能会受到影响，因为这些权限与特定用户相关联。
  - 这种方式可能会导致权限管理的不连续性，特别是在用户变动频繁的环境中。

5.6

```
CREATE TRIGGER insert_into_branch_cust_via_depositor
AFTER INSERT ON depositor
REFERENCING NEW ROW AS inserted
FOR EACH ROW
BEGIN
INSERT INTO branch_cust
    SELECT branch_name, inserted.customer_name
    FROM account
    WHERE inserted.account_number = account.account_number;
END;

CREATE TRIGGER insert_into_branch_cust_via_account
AFTER INSERT ON account
REFERENCING NEW ROW AS inserted
FOR EACH STATEMENT
BEGIN
INSERT INTO branch_cust
    SELECT inserted.branch_name, customer_name
    FROM depositor
    WHERE depositor.account_number = inserted.account_number;
END;
```

5.15

a.

```
CREATE FUNCTION avg_salary(ename VARCHAR(15))
    RETURNS INTEGER
    DECLARE result INTEGER;
    SELECT AVG(salary) INTO result
    FROM works
    WHERE works.company_name = ename;
    RETURN result;
END;

SELECT company_name
FROM works
WHERE avg_salary(company_name) > avg_salary('First Bank Corporation');
```

b.

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

5.19

我们可以为表 `s` 创建一个 `AFTER DELETE` 触发器，在删除表 `s` 中的记录时，自动删除表 `r` 中相关的记录。

```
CREATE TRIGGER cascade_delete_on_s
AFTER DELETE ON s
FOR EACH ROW
BEGIN
    DELETE FROM r
    WHERE r.B = OLD.A;
END;
```