```
employee (person-name, street, city)
works (person-name, company-name, salary)
company (company-name, city)
manages (person-name, manager-name)
```

**Figure 3.39**. Relational database for Exercises 3.5, 3.8 and 3.10.

- **3.4** In Chapter 2, we saw how to represent many-to-many, many-to-one, one-to-many, and one-to-one relationship sets. Explain how primary keys help us to represent such relationship sets in the relational model.
  - **Answer:** Suppose the primary key of relation schema R is  $\{A_{i_1}, A_{i_2}, ..., A_{i_n}\}$  and the primary key of relation schema S is  $\{B_{i_1}, B_{i_2}, ..., B_{i_m}\}$ . Then a relationship between the 2 sets can be represented as a tuple  $(A_{i_1}, A_{i_2}, ..., A_{i_n}, B_{i_1}, B_{i_2}, ..., B_{i_m})$ . In a one-to-one relationship, each value on  $\{A_{i_1}, A_{i_2}, ..., A_{i_n}\}$  will appear in exactly one tuple and likewise for  $\{B_{i_1}, B_{i_2}, ..., B_{i_m}\}$ . In a many-to-one relationship (e.g., many A one B), each value on  $\{A_{i_1}, A_{i_2}, ..., A_{i_n}\}$  will appear once, and each value on  $\{B_{i_1}, B_{i_2}, ..., B_{i_n}\}$  may appear many times. In a many-to-many relationship, values on both  $\{A_{i_1}, A_{i_2}, ..., A_{i_n}\}$  and  $\{B_{i_1}, B_{i_2}, ..., B_{i_m}\}$  will appear many times. However, in all the above cases  $\{A_{i_1}, A_{i_2}, ..., A_{i_n}, B_{i_1}, B_{i_2}, ..., B_{i_m}\}$  is a primary key, so no tuple on  $(A_{j_1}, ..., A_{j_n}, B_{k_1}, ..., B_{k_m})$  will appear more than once.
- **3.5** Consider the relational database of Figure 3.39, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries:
  - **a.** Find the names of all employees who work for First Bank Corporation.
  - **b.** Find the names and cities of residence of all employees who work for First Bank Corporation.
  - **c.** Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than \$10,000 per annum.
  - **d.** Find the names of all employees in this database who live in the same city as the company for which they work.
  - **e.** Find the names of all employees who live in the same city and on the same street as do their managers.
  - **f.** Find the names of all employees in this database who do not work for First Bank Corporation.
  - **g.** Find the names of all employees who earn more than every employee of Small Bank Corporation.
  - **h.** Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

#### Answer:

```
a. \Pi_{person\text{-}name} (\sigma_{company\text{-}name} = "First Bank Corporation" (works))

b. \Pi_{person\text{-}name, city} (employee \bowtie (\sigma_{company\text{-}name} = "First Bank Corporation" (works)))
```

```
c. \Pi_{person\text{-}name, street, city}
(\sigma_{(company\text{-}name} = \text{``First Bank Corporation''} \land salary > 10000)
works \bowtie employee)
d. \Pi_{person\text{-}name} \ (employee \bowtie works \bowtie company)
e. \Pi_{person\text{-}name} \ ((employee \bowtie manages))
(manager\text{-}name = employee2.person\text{-}name \land employee.street} = employee2.street)
```

 $\land$  employee.city = employee2.city) ( $\rho$ employee2 (employee))) **f.** The following solutions assume that all people work for exactly one company. If one allows people to appear in the database (e.g. in *employee*) but not appear in *works*, the problem is more complicated. We give solutions for this more realistic case later

- **3.6** Consider the relation of Figure 3.21, which shows the result of the query "Find the names of all customers who have a loan at the bank." Rewrite the query to include not only the name, but also the city of residence for each customer. Observe that now customer Jackson no longer appears in the result, even though Jackson does in fact have a loan from the bank.
  - **a.** Explain why Jackson does not appear in the result.
  - **b.** Suppose that you want Jackson to appear in the result. How would you modify the database to achieve this effect?
  - **c.** Again, suppose that you want Jackson to appear in the result. Write a query using an outer join that accomplishes this desire without your having to modify the database.

# **Answer:** The rewritten query is

 $\Pi_{customer\text{-}name, customer\text{-}city, amount}(borrower \bowtie loan \bowtie customer)$ 

- a. Although Jackson does have a loan, no address is given for Jackson in the customer relation. Since no tuple in customer joins with the Jackson tuple of borrower, Jackson does not appear in the result.
- b. The best solution is to insert Jackson's address into the *customer* relation. If the address is unknown, null values may be used. If the database system does not support nulls, a special value may be used (such as **unknown**) for Jackson's street and city. The special value chosen must not be a plausible name for an actual city or street.

- c.  $\Pi_{customer-name, customer-city, amount}((borrower \bowtie loan) \bowtie customer)$
- **3.7** The outer-join operations extend the natural-join operation so that tuples from the participating relations are not lost in the result of the join. Describe how the theta join operation can be extended so that tuples from the left, right, or both relations are not lost from the result of a theta join.

#### **Answer:**

- **a.** The left outer theta join of r(R) and s(S)  $(r \bowtie_{\theta} s)$  can be defined as  $(r \bowtie_{\theta} s) \cup ((r \prod_{R} (r \bowtie_{\theta} s)) \times (null, null, \dots, null))$  The tuple of nulls is of size equal to the number of attributes in S.
- **b.** The right outer theta join of r(R) and s(S)  $(r \bowtie_{\theta} s)$  can be defined as  $(r \bowtie_{\theta} s) \cup ((null, null, \dots, null) \times (s \Pi_S(r \bowtie_{\theta} s)))$  The tuple of nulls is of size equal to the number of attributes in R.
- **c.** The full outer theta join of r(R) and s(S)  $(r \bowtie_{\theta} s)$  can be defined as  $(r \bowtie_{\theta} s) \cup ((null, null, \ldots, null) \times (s \Pi_S(r \bowtie_{\theta} s))) \cup ((r \Pi_R(r \bowtie_{\theta} s)) \times (null, null, \ldots, null))$  The first tuple of nulls is of size equal to the number of attributes in R, and the second one is of size equal to the number of attributes in S.
- **3.8** Consider the relational database of Figure 3.39. Give an expression in the relational algebra for each request:
  - **a.** Modify the database so that Jones now lives in Newtown.
  - **b.** Give all employees of First Bank Corporation a 10 percent salary raise.
  - **c.** Give all managers in this database a 10 percent salary raise.
  - **d.** Give all managers in this database a 10 percent salary raise, unless the salary would be greater than \$100,000. In such cases, give only a 3 percent raise.
  - **e.** Delete all tuples in the *works* relation for employees of Small Bank Corporation.

### **Answer:**

- $\begin{array}{l} \textbf{a.} \ employee \leftarrow \Pi_{person\text{-}name,street,"Newtown"} \\ (\sigma_{person\text{-}name} = \text{"Jones"}(employee)) \\ \cup (employee \sigma_{person\text{-}name} = \text{"Jones"}(employee)) \\ \textbf{b.} \ works \leftarrow \Pi_{person\text{-}name,company\text{-}name,1.1*salary}( \\ \sigma_{(company\text{-}name} = \text{"First Bank Corporation"})(works)) \\ \cup (works \sigma_{company\text{-}name} = \text{"First Bank Corporation"}(works)) \\ \end{array}$
- c. The update syntax allows reference to a single relation only. Since this update requires access to both the relation to be updated (works) and the manages relation, we must use several steps. First we identify the tuples of works to be updated and store them in a temporary relation ( $t_1$ ). Then we create a temporary relation containing the new tuples ( $t_2$ ). Finally, we delete the tuples in  $t_1$ , from works and insert the tuples of  $t_2$ .

```
t_1 \leftarrow \Pi_{works.person-name,company-name,salary} (\sigma_{works.person-name=manager-name}(works \times manages))
```

$$t_2 \leftarrow \Pi_{person-name, company-name, 1.1*salary}(t_1)$$
  
 $works \leftarrow (works - t_1) \cup t_2$ 

**d.** The same situation arises here. As before,  $t_1$ , holds the tuples to be updated and  $t_2$  holds these tuples in their updated form.

```
t_1 \leftarrow \Pi_{works.person\text{-}name,company\text{-}name,salary} \\ (\sigma_{works.person\text{-}name=manager\text{-}name}(works \times manages))
t_2 \leftarrow \Pi_{works.person\text{-}name,company\text{-}name,salary*1.03} \\ (\sigma_{t_1.salary*1.1} > 100000(t_1))
t_2 \leftarrow t_2 \cup (\Pi_{works.person\text{-}name,company\text{-}name,salary*1.1} \\ (\sigma_{t_1.salary*1.1} \leq 100000(t_1)))
works \leftarrow (works - t_1) \cup t_2
e. works \leftarrow works - \sigma_{company\text{-}name}="Small Bank Corporation" (works)
```

- **3.9** Using the bank example, write relational-algebra queries to find the accounts held by more than two customers in the following ways:
  - a. Using an aggregate function.
  - b. Without using any aggregate functions.

## **Answer:**

```
a. t_1 \leftarrow account\text{-}number\mathcal{G}_{\textbf{count}} \ customer\text{-}name(depositor)
\Pi_{account\text{-}number} \left(\sigma_{num\text{-}holders>2} \left(\rho_{account\text{-}holders(account\text{-}number,num\text{-}holders)}(t_1)\right)\right)
b. t_1 \leftarrow \left(\rho_{d1}(depositor) \times \rho_{d2}(depositor) \times \rho_{d3}(depositor)\right)
t_2 \leftarrow \sigma_{(d1.account\text{-}number=d2.account\text{-}number=d3.account\text{-}number)}(t_1)
\Pi_{d1.account\text{-}number}\left(\sigma_{(d1.customer\text{-}name\neq d2.customer\text{-}name\neq d1.customer\text{-}name)}(t_2)\right)
d2.customer\text{-}name\neq d3.customer\text{-}name \land d3.customer\text{-}name\neq d1.customer\text{-}name)}(t_2)
```

- **3.10** Consider the relational database of Figure 3.39. Give a relational-algebra expression for each of the following queries:
  - **a.** Find the company with the most employees.
  - **b.** Find the company with the smallest payroll.
  - **c.** Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

# **Answer:**

```
a. t_1 \leftarrow_{company-name} \mathcal{G}_{count-distinct\ person-name}(works)
t_2 \leftarrow_{max_{num-employees}}(\rho_{company-strength(company-name,num-employees)}(t_1))
\Pi_{company-name}(\rho_{t_3(company-name,num-employees)}(t_1) \bowtie \rho_{t_4(num-employees)}(t_2))
b. t_1 \leftarrow_{company-name} \mathcal{G}_{sum\ salary}(works)
t_2 \leftarrow_{min_{payroll}}(\rho_{company-payroll(company-name,payroll)}(t_1))
\Pi_{company-name}(\rho_{t_3(company-name,payroll)}(t_1) \bowtie \rho_{t_4(payroll)}(t_2))
c. t_1 \leftarrow_{company-name} \mathcal{G}_{avg\ salary}(works)
t_2 \leftarrow_{company-name} \mathcal{G}_{avg\ salary}(works)
t_3 \leftarrow_{company-name} \mathcal{G}_{avg\ salary}(works)
```