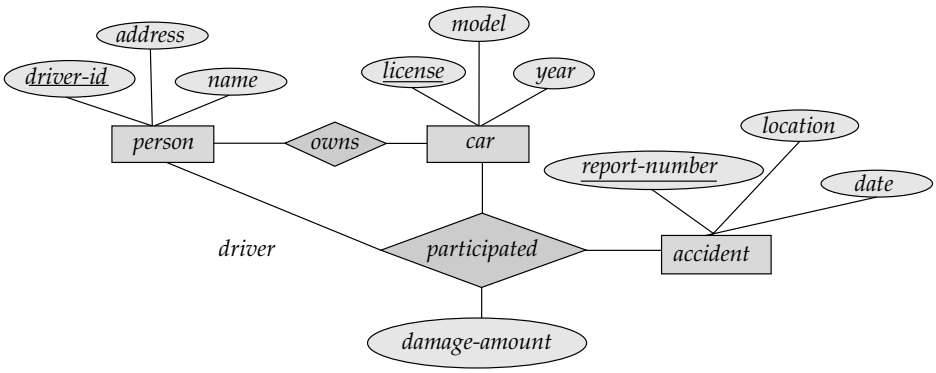

Figure 3.38. E-R diagram.

Exercises

3.1 Design a relational database for a university registrar's office. The office maintains data about each class, including the instructor, the number of students enrolled, and the time and place of the class meetings. For each student–class pair, a grade is recorded.
Answer: Underlined attributes indicate the primary key.

student (student-id, name, program)
course (courseno, title, syllabus, credits)
course-offering (courseno, secno, year, semester, time, room)
instructor (instructor-id, name, dept, title)
enrols (student-id, courseno, secno, semester, year, grade)
teaches (courseno, secno, semester, year, instructor-id)
requires (maincourse, prerequisite)

3.2 Describe the differences in meaning between the terms *relation* and *relation schema*. Illustrate your answer by referring to your solution to Exercise 3.1.
Answer: A relation schema is a type definition, and a relation is an instance of that schema. For example, *student* (*ss#*, *name*) is a relation schema and

ss#	name
123-45-6789	Tom Jones
456-78-9123	Joe Brown

is a relation based on that schema.

3.3 Design a relational database corresponding to the E-R diagram of Figure 3.38.
Answer: The relational database schema is given below.

person (driver-id, *name*, *address*)
car (license, *year*, *model*)
accident (report-number, *location*, *date*)
owns (driver-id, license)
participated (report-number *driver-id*, *license*, *damage-amount*)

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. $\Pi_{person\text{-}name,\ street,\ city}$
$(\sigma_{(company\text{-}name\ =\ \text{“First Bank Corporation”}\ \wedge\ salary\ >\ 10000)}$
$works \bowtie employee)$

d. $\Pi_{person\text{-}name}\ (employee \bowtie works \bowtie company)$

e. $\Pi_{person\text{-}name}\ ((employee \bowtie manages)$
$\bowtie_{(manager\text{-}name\ =\ employee2.person\text{-}name\ \wedge\ employee.street\ =\ employee2.street}$
$_{\wedge\ employee.city\ =\ employee2.city)}(\rho_{employee2}\ (employee)))$

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.

$\Pi_{person\text{-}name}\ (\sigma_{company\text{-}name\ \neq\ \text{“First Bank Corporation”}}(works))$

If people may not work for any company:

$\Pi_{person\text{-}name}(employee)\ -\ \Pi_{person\text{-}name}$
$(\sigma_{(company\text{-}name\ =\ \text{“First Bank Corporation”})}(works))$

g. $\Pi_{person\text{-}name}\ (works)\ -\ (\Pi_{works.person\text{-}name}\ (works$
$\bowtie_{(works.salary\ \leq works2.salary\ \wedge\ works2.company\text{-}name\ =\ \text{“Small Bank Corporation”})}$
$\rho_{works2}(works)))$

h. Note: Small Bank Corporation will be included in each answer.

$\Pi_{company\text{-}name}\ (company \div$
$(\Pi_{city}\ (\sigma_{company\text{-}name\ =\ \text{“Small Bank Corporation”}}\ (company))))$

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

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*) $(r \ \exists\!\!\bowtie_\theta \ s)$ can be defined as
$(r \bowtie_\theta s) \cup ((r - \Pi_R(r \bowtie_\theta s)) \times (null, null, \ldots, null))$
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*) $(r \bowtie\!\!\sqsubset_\theta \ s)$ can be defined as
$(r \bowtie_\theta s) \cup ((null, null, \ldots, null) \times (s - \Pi_S(r \bowtie_\theta s)))$
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*) $(r \ \exists\!\!\bowtie\!\!\sqsubset_\theta \ s)$ 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:

a. $employee \leftarrow \Pi_{person\text{-}name,street,\text{“}Newtown\text{”}}$
$(\sigma_{person\text{-}name=\text{“Jones”}}(employee))$
$\cup (employee - \sigma_{person\text{-}name=\text{“Jones”}}(employee))$

b. $works \leftarrow \Pi_{person\text{-}name,company\text{-}name,1.1*salary}($
$\sigma_{(company\text{-}name=\text{“First Bank Corporation”})}(works))$
$\cup (works - \sigma_{company\text{-}name=\text{“First Bank Corporation”}}(works))$

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.

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

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

$works \leftarrow (works - t_1) \cup t_2$

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. $works \leftarrow works - \sigma_{company-name=\text{“Small Bank Corporation”}}(works)$

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. $t_1 \leftarrow {}_{account\text{-}number}\mathcal{G}_{\textbf{count}\ customer\text{-}name}(depositor)$
$\Pi_{account\text{-}number}\ (\sigma_{num\text{-}holders>2}\ (\rho_{account\text{-}holders(account\text{-}number,num\text{-}holders)}(t_1)))$

b. $t_1 \leftarrow (\rho_{d1}(depositor) \times \rho_{d2}(depositor) \times \rho_{d3}(depositor))$
$t_2 \leftarrow \sigma_{(d1.account\text{-}number=d2.account\text{-}number=d3.account\text{-}number)}(t_1)$
$\Pi_{d1.account\text{-}number}(\sigma_{(d1.customer\text{-}name\neq d2.customer\text{-}name\ \wedge}$
$_{d2.customer\text{-}name\neq d3.customer\text{-}name\ \wedge d3.customer\text{-}name\neq d1.customer\text{-}name)}(t_2))$

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. $t_1 \leftarrow {}_{company\text{-}name}\mathcal{G}_{\textbf{count-distinct}\ person\text{-}name}(works)$
$t_2 \leftarrow \textbf{max}_{num\text{-}employees}(\rho_{company\text{-}strength(company\text{-}name,num\text{-}employees)}(t_1))$
$\Pi_{company\text{-}name}(\rho_{t_3(company\text{-}name,num\text{-}employees)}(t_1) \bowtie \rho_{t_4(num\text{-}employees)}(t_2))$

b. $t_1 \leftarrow {}_{company\text{-}name}\mathcal{G}_{\textbf{sum}\ salary}(works)$
$t_2 \leftarrow \textbf{min}_{payroll}(\rho_{company\text{-}payroll(company\text{-}name,payroll)}(t_1))$
$\Pi_{company\text{-}name}(\rho_{t_3(company\text{-}name,payroll)}(t_1) \bowtie \rho_{t_4(payroll)}(t_2))$

c. $t_1 \leftarrow {}_{company\text{-}name}\mathcal{G}_{\textbf{avg}\ salary}(works)$
$t_2 \leftarrow \sigma_{company\text{-}name\ =\ \text{“First Bank Corporation”}}(t_1)$

$$\Pi_{t_3.\textit{company-name}}((\rho_{t_3(\textit{company-name},\textit{avg-salary})}(t_1)) \bowtie_{t_3.\textit{avg-salary} > \textit{first-bank.avg-salary}} (\rho_{\textit{first-bank}(\textit{company-name},\textit{avg-salary})}(t_2)))$$

3.11 List two reasons why we may choose to define a view.
Answer:

a. Security conditions may require that the entire logical database be not visible to all users.
b. We may wish to create a personalized collection of relations that is better matched to a certain user's intuition than is the actual logical model.

3.12 List two major problems with processing update operations expressed in terms of views.
Answer: Views present significant problems if updates are expressed with them. The difficulty is that a modification to the database expressed in terms of a view must be translated to a modification to the actual relations in the logical model of the database.

a. Since the view may not have all the attributes of the underlying tables, insertion of a tuple into the view will insert tuples into the underlying tables, with those attributes not participating in the view getting null values. This may not be desirable, especially if the attribute in question is part of the primary key of the table.
b. If a view is a join of several underlying tables and an insertion results in tuples with nulls in the join columns, the desired effect of the insertion will not be achieved. In other words, an update to a view may not be expressible at all as updates to base relations. For an explanatory example, see the *loan-info* updation example in Section 3.5.2.

3.13 Let the following relation schemas be given:

$$\begin{aligned} R &= (A, B, C) \\ S &= (D, E, F) \end{aligned}$$

Let relations $r(R)$ and $s(S)$ be given. Give an expression in the tuple relational calculus that is equivalent to each of the following:

a. $\Pi_A(r)$
b. $\sigma_{B=17}(r)$
c. $r \times s$
d. $\Pi_{A,F}(\sigma_{C=D}(r \times s))$

Answer:

a. $\{t \mid \exists\, q \in r\, (q[A] = t[A])\}$
b. $\{t \mid t \in r \wedge t[B] = 17\}$
c. $\{t \mid \exists\, p \in r\, \exists\, q \in s\, (t[A] = p[A] \wedge t[B] = p[B] \wedge t[C] = p[C] \wedge t[D] = q[D] \wedge t[E] = q[E] \wedge t[F] = q[F])\}$
d. $\{t \mid \exists\, p \in r\, \exists\, q \in s\, (t[A] = p[A] \wedge t[F] = q[F] \wedge p[C] = q[D]\}$

3.14 Let $R = (A, B, C)$, and let r_1 and r_2 both be relations on schema R. Give an expression in the domain relational calculus that is equivalent to each of the following:

a. $\Pi_A(r_1)$
b. $\sigma_{B=17}(r_1)$
c. $r_1 \cup r_2$
d. $r_1 \cap r_2$
e. $r_1 - r_2$
f. $\Pi_{A,B}(r_1) \bowtie \Pi_{B,C}(r_2)$

Answer:

a. $\{< t > \mid \exists\, p,\ q\ (< t, p, q > \in r_1)\}$
b. $\{< a, b, c > \mid < a, b, c > \in r_1 \wedge b = 17\}$
c. $\{< a, b, c > \mid < a, b, c > \in r_1 \vee < a, b, c > \in r_2\}$
d. $\{< a, b, c > \mid < a, b, c > \in r_1 \wedge < a, b, c > \in r_2\}$
e. $\{< a, b, c > \mid < a, b, c > \in r_1 \wedge < a, b, c > \notin r_2\}$
f. $\{< a, b, c > \mid \exists\, p,\ q\ (< a, b, p > \in r_1 \wedge < q, b, c > \in r_2)\}$

3.15 Repeat Exercise 3.5 using the tuple relational calculus and the domain relational calculus.
Answer:

a. Find the names of all employees who work for First Bank Corporation:-
  i. $\{t \mid \exists\, s \in works\ (t[person\text{-}name] = s[person\text{-}name] \wedge s[company\text{-}name] = \text{“First Bank Corporation”})\}$
  ii. $\{< p > \mid \exists\, c,\ s\ (< p, c, s > \in works \wedge c = \text{“First Bank Corporation”})\}$

b. Find the names and cities of residence of all employees who work for First Bank Corporation:-
  i. $\{t \mid \exists\, r \in employee\, \exists\, s \in works\ (\ t[person\text{-}name] = r[person\text{-}name] \wedge t[city] = r[city] \wedge r[person\text{-}name] = s[person\text{-}name] \wedge s[company\text{-}name] = \text{“First Bank Corporation”})\}$
  ii. $\{< p, c > \mid \exists\, co,\ sa,\ st\ (< p, co, sa > \in works \wedge < p, st, c > \in employee \wedge co = \text{“First Bank Corporation”})\}$

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:-
  i. $\{t \mid t \in employee \wedge (\exists\, s \in works\ (\ s[person\text{-}name] = t[person\text{-}name] \wedge s[company\text{-}name] = \text{“First Bank Corporation”} \wedge s[salary] > 10000))\}$
  ii. $\{< p, s, c > \mid < p, s, c > \in employee \wedge \exists\, co,\ sa\ (< p, co, sa > \in works \wedge co = \text{“First Bank Corporation”} \wedge sa > 10000)\}$

d. Find the names of all employees in this database who live in the same city as the company for which they work:-
  i. $\{t \mid \exists\, e \in employee\, \exists\, w \in works\, \exists\, c \in company\ (t[person\text{-}name] = e[person\text{-}name] \wedge e[person\text{-}name] = w[person\text{-}name] \wedge w[company\text{-}name] = c[company\text{-}name] \wedge e[city] = c[city])\}$

ii. $\{< p > \mid \exists\, st,\ c,\ co,\ sa\ (< p, st, c > \in\ employee$
$\wedge\ < p, co, sa > \in\ works \wedge\ < co, c > \in\ company)\}$

e. Find the names of all employees who live in the same city and on the same street as do their managers:-

i. $\{\ t \mid \exists\, l\ \in\ employee\, \exists\, m\ \in\ manages\, \exists\, r\ \in\ employee$
$(l[person\text{-}name] = m[person\text{-}name] \wedge m[manager\text{-}name] = r[person\text{-}name]$
$\wedge\ l[street] = r[street] \wedge l[city] = r[city] \wedge t[person\text{-}name] = l[person\text{-}name])\}$

ii. $\{< t > \mid \exists\, s,\ c,\ m\ (< t, s, c > \in\ employee \wedge\ < t, m > \in\ manages \wedge\ < m, s, c > \in\ employee)\}$

f. Find the names of all employees in this database who do not work for First Bank Corporation:-
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.

i. $\{\ t \mid \exists\, w\ \in\ works\ (\ w[company\text{-}name] \neq$ "First Bank Corporation"
$\wedge\ t[person\text{-}name] = w[person\text{-}name])\}$

ii. $\{< p > \mid \exists\, c,\ s\ (< p, c, s > \in\ works \wedge\ c \neq$ "First Bank Corporation")$\}$

If people may not work for any company:

i. $\{\ t \mid \exists\, e\ \in\ employee\ (\ t[person\text{-}name] = e[person\text{-}name] \wedge \neg\, \exists\, w\ \in\ works$
$(w[company\text{-}name] =$ "First Bank Corporation"
$\wedge\, w[person\text{-}name] = t[person\text{-}name]))\}$

ii. $\{< p > \mid \exists\, s,\ c\ (< p, s, c > \in\ employee) \wedge\ \neg\, \exists\, x,\ y$
$(y =$ "First Bank Corporation"$\wedge\ < p, y, x > \in\ works)\}$

g. Find the names of all employees who earn more than every employee of Small Bank Corporation:-

i. $\{t \mid \exists\, w\ \in\ works\ (t[person\text{-}name] = w[person\text{-}name] \wedge\ \forall\, s\ \in\ works$
$(s[company\text{-}name] =$ "Small Bank Corporation" $\Rightarrow\ w[salary] > s[salary]))\}$

ii. $\{< p > \mid \exists\, c,\ s\ (< p, c, s > \in\ works\ \wedge\ \forall\, p_2,\ c_2,\ s_2$
$(< p_2, c_2, s_2 > \notin\ works \vee\ c_2 \neq$ "Small Bank Corporation" $\vee\ s > s_2))\}$

h. Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.
Note: Small Bank Corporation will be included in each answer.

i. $\{t \mid \forall\, s\ \in\ company\ (s[company\text{-}name] =$ "Small Bank Corporation" $\Rightarrow$
$\exists\, r\ \in\ company\, (t[company\text{-}name] = r[company\text{-}name] \wedge r[city] = s[city]))\}$

ii. $\{< co > \mid \forall\, co_2,\ ci_2\ (< co_2, ci_2 > \notin\ company$
$\vee\ co_2 \neq$ "Small Bank Corporation" $\vee\ < co, ci_2 > \in\ company)\}$

3.16 Let $R = (A,\ B)$ and $S = (A,\ C)$, and let $r(R)$ and $s(S)$ be relations. Write relational-algebra expressions equivalent to the following domain-relational-calculus expressions:

a. $\{< a > \mid \exists\, b\, (< a, b > \in r \wedge b = 17)\}$
b. $\{< a, b, c > \mid < a, b > \in r \wedge < a, c > \in s\}$
c. $\{< a > \mid \exists\, b\, (< a, b > \in r) \vee \forall\, c\, (\exists\, d\, (< d, c > \in s) \Rightarrow < a, c > \in s)\}$
d. $\{< a > \mid \exists\, c\, (< a, c > \in s \wedge \exists\, b_1, b_2\, (< a, b_1 > \in r \wedge < c, b_2 > \in r \wedge b_1 > b_2))\}$

Answer:

a. $\Pi_A\, (\sigma_{B\,=\,17}\, (r))$
b. $r \bowtie s$
c. $\Pi_A(r) \cup (r \div \sigma_B(\Pi_C(s)))$
d. $\Pi_{r.A}\, ((r \bowtie s) \bowtie_{c\,=\,r2.A\,\wedge\,r.B\,>\,r2.B}\, (\rho_{r_2}(r)))$

It is interesting to note that (*d*) is an abstraction of the notorious query "Find all employees who earn more than their manager." Let $R = (emp,\ sal)$, $S = (emp,\ mgr)$ to observe this.

3.17 Let $R = (A,\ B)$ and $S = (A,\ C)$, and let $r(R)$ and $s(S)$ be relations. Using the special constant *null*, write tuple-relational-calculus expressions equivalent to each of the following:

a. $r \bowtie\!\!\!\!\sqsubset s$
b. $r \rhd\!\!\!\!\bowtie\!\!\!\!\sqsubset s$
c. $r \rhd\!\!\!\!\bowtie s$

Answer:

a. $\{t \mid \exists r \in R\, \exists s \in S\, (r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists s \in S(\neg \exists r \in R(r[A] = s[A]) \wedge t[A] = s[A] \wedge t[C] = s[C] \wedge t[B] = null)\}$
b. $\{t \mid \exists r \in R\, \exists s \in S\, (r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists r \in R(\neg \exists s \in S(r[A] = s[A]) \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = null) \vee \exists s \in S(\neg \exists r \in R(r[A] = s[A]) \wedge t[A] = s[A] \wedge t[C] = s[C] \wedge t[B] = null)\}$
c. $\{t \mid \exists r \in R\, \exists s \in S\, (r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists r \in R(\neg \exists s \in S(r[A] = s[A]) \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = null)\}$

3.18 List two reasons why null values might be introduced into the database.
Answer: Nulls may be introduced into the database because the actual value is either unknown or does not exist. For example, an employee whose address has changed and whose new address is not yet known should be retained with a null address. If employee tuples have a composite attribute *dependents*, and a particular employee has no dependents, then that tuple's *dependents* attribute should be given a null value.

3.19 Certain systems allow *marked* nulls. A marked null $\perp_i$ is equal to itself, but if $i \neq j$, then $\perp_i \neq \perp_j$. One application of marked nulls is to allow certain updates through views. Consider the view *loan-info* (Section 3.5). Show how you can use marked nulls to allow the insertion of the tuple ("Johnson", 1900) through *loan-info*.
Answer: To insert the tuple ("Johnson", 1900) into the view *loan-info*, we can do