CHAPTER 5

Other Relational Languages

Exercises

5.1 Answer: The *participated* relation relates car(s) and accidents. Assume the *date* attribute is of the form "YYYY-MM-DD".

a. Find the total number of people who owned cars that were involved in accidents in 1989.

accident	*report-number*	*date*	*location*
	_report	*_date*	

participated	*driver-id*	*car*	*report-number*	damage-amount
	P.CNT.UNQ.ALL		*_report*	

conditions
_date = ($\geq$ 1989-00-00 **and** $\leq$ 1989-12-31)

b. Find the number of accidents in which the cars belonging to "John Smith" were involved.

person	*driver-id*	*name*	*address*
	_driver	John Smith	

participated	*driver-id*	*car*	*report-number*	*damage-amount*
	_driver		P.CNT.ALL	

c. Add a new accident to the database; assume any values for required attributes.

We assume that the driver was "Williams", although it could have been someone else. Also assume that "Williams" has only one Toyota.

accident	*report-number*	*date*	*location*
I.	4007	1997-01-01	Berkeley

participated	*driver-id*	*car*	*report-number*	*damage-amount*
I.	_driver	_license	4007	3000

owns	*driver-id*	*license*
	_driver	_license

car	*license*	*year*	*model*
	_license	_year	Toyota

person	*driver-id*	*name*	*address*
	_driver	Williams	

d. Delete the car "Mazda" that belongs to "John Smith."

person	*driver-id*	*name*	*address*
	_driver	John Smith	

owns	*driver-id*	*license*
	_driver	_license

car	*license*	*year*	*model*
D.	_license		Mazda

e. Update the damage amount for the car with license number "AABB2000" in the accident with report number "AR2197" to $3000.

owns	*driver-id*	*license*
	_driver	"AABB2000"

participated	*driver-id*	*car*	*report-number*	*damage-amount*
	_driver		"AR2197"	U.3000

5.3 Answer:

a. Find all employees who earn more than the average salary of all employees of their company.

The following solution assumes that all people work for at most one company.

works	*person-name*	*company-name*	*salary*
	P.	_y	_x
		_y	_z

conditions
$\_x > \text{AVG.ALL.}\_z$

b. Find the company that has the most employees.

works	*person-name*	*company-name*	*salary*
	_x	P.G.	
	_y	G.	

conditions
CNT.UNQ.$\_x \geq$ MAX.CNT.UNQ.ALL.$\_y$

c. Find the company that has the smallest payroll.

works	*person-name*	*company-name*	*salary*
		P.G.	_x
		G.	_y

conditions
SUM.ALL.$\_x \leq$ MIN.SUM.ALL.$\_y$

d. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

works	*person-name*	*company-name*	*salary*
		P.G.	_x
		First Bank Corporation	_y

conditions
AVG.ALL.$\_x >$ AVG.ALL.$\_y$

5.5 Answer:

a. $\Pi_A(r)$

i.

r	A	B	C
	P.		

ii. $query\,(X)$:- $r\,(X,\ Y,\ Z)$

b. $\sigma_{B\,=\,17}\;(r)$

i.

r	A	B	C
P.		17	

ii. $query\,(X,\,Y,\,Z)$:- $r\,(X,\,Y,\,Z),\;Y\;=\;17$

c. $r\;\times\;s$

i.

$result$	A	B	C	D	E	F
P.	_a	_b	_c	_d	_e	_f

r	A	B	C
	_a	_b	_c

s	D	E	F
	_d	_e	_f

ii. $query\,(X,\,Y,\,Z,\,U,\,V,\,W)$:- $r\,(X,\,Y,\,Z),\;s\,(U,\,V,\,W)$

d. $\Pi_{A,F}\,(\sigma_{C\,=\,D}(r\;\times\;s))$

i.

$result$	A	F
P.	_a	_f

r	A	B	C
	_a		_c

s	D	E	F
	_c		_f

ii. $query\,(X,\,Y)$:- $r\,(X, V,\,W),\;s\,(W,\,Z,\,Y)$

5.7 Answer:

a. $\{< a > \;|\; \exists\, b\,(< a, b > \in\; r\;\wedge\; b\;=\;17)\}$

i.

r	A	B
	P.	17

ii. $query\,(X)$:- $r\,(X,\,17)$

b. $\{< a, b, c > \;|\; < a, b > \in\; r \wedge\; < a, c > \in\; s\}$

i.

r	A	B
	$\_a$	$\_b$

s	A	C
	$\_a$	$\_c$

$result$	A	B	C
P.	$\_a$	$\_b$	$\_c$

ii. *query(X, Y, Z) :- r(X, Y), s(X, Z)*

c. $\{< a > \mid \exists\, c\, (< a, c > \in\ s \wedge \exists\, b_1, b_2\, (< a, b_1 > \in\ r \wedge\ < c, b_2 > \in\ r \wedge b_1\ > b_2))\}$

i.

r	A	B
	$\_a$	$> \_s$
	$\_c$	$\_s$

s	A	C
	P.$\_a$	$\_c$

ii. $query\,(X)$:- $s\,(X,\,Y),\ r\,(X,\,Z),\ r\,(Y,\,W),\ Z\ >\ W$

5.8 Answer:

a. Find all employees who work (directly or indirectly) under the manager "Jones".

$query\,(X)$:- $p\,(X)$
$p\,(X)$:- $manages\,(X,$ "Jones")
$p\,(X)$:- $manages\,(X,\,Y),\ p\,(Y)$

b. Find all cities of residence of all employees who work (directly or indirectly) under the manager "Jones".

query(X, C) :- p(X), employee(X, S, C)
p(X) :- manages(X, "Jones")
p(X) :- manages(X, Y), p(Y)

c. Find all pairs of employees who have a (direct or indirect) manager in common.

query(X, Y) :- p(X, W), p(Y, W)
p(X, Y) :- manages(X, Y)
p(X, Y) :- manages(X, Z), p(Z, Y)

d. Find all pairs of employees who have a (direct or indirect) manager in common, and are at the same number of levels of supervision below the common manager.

query(X, Y) :- p(X, Y)
p(X, Y) :- manages(X, Z), manages(Y, Z)
p(X, Y) :- manages(X, V), manages(Y, W), p(V, W)

5.10 Answer: A Datalog rule has two parts, the *head* and the *body*. The body is a comma separated list of *literals*. A *positive literal* has the form $p(t_1,\ t_2, \ldots,\ t_n\ )$ where p is the name of a relation with n attributes, and $t_1,\ t_2, \ldots,\ t_n$ are either constants or variables. A *negative literal* has the form $\neg p(t_1,\ t_2, \ldots,\ t_n)$ where p has n attributes. In the case of arithmetic literals, p will be an arithmetic operator like $>$, $=$ etc.

We consider only safe rules; see Section 5.2.4 for the definition of safety of Datalog rules. Further, we assume that every variable that occurs in an arithmetic literal also occurs in a positive non-arithmetic literal.

Consider first a rule without any negative literals. To express the rule as an extended relational-algebra view, we write it as a join of all the relations referred to in the (positive) non-arithmetic literals in the body, followed by a selection. The selection condition is a conjunction obtained as follows. If $p_1\ (X,\ Y),\ p_2\ (Y,\ Z)$ occur in the body, where p_1 is of the schema $(A,\ B)$ and p_2 is of the schema $(C,\ D)$, then $p_1.B\ =\ p_2.C$ should belong to the conjunction. The arithmetic literals can then be added to the condition.

As an example, the Datalog query

query(X, Y) :- works(X, C, S1), works(Y, C, S2), S1 > S2, manages(X, Y)

becomes the following relational-algebra expression:

$$\begin{aligned} E_1 \ = \ & \sigma_{(w1.company\text{-}name\,=\,w2.company\text{-}name\ \wedge\ w1.salary > w2.salary\ \wedge} \\ & \quad {}_{manages.person\text{-}name\,=\,w1.person\text{-}name\ \wedge\ manages.manager\text{-}name\,=\,w2.person\text{-}name)} \\ & (\rho_{w1}(works)\ \times\ \rho_{w2}(works)\ \times\ manages) \end{aligned}$$

Now suppose the given rule has negative literals. First suppose that there are no constants in the negative literals; recall that all variables in a negative literal must also occur in a positive literal. Let $\neg q(X, Y)$ be the first negative literal, and let it be of the schema $(E,\ F)$. Let E_i be the relational algebra expression obtained after all positive and arithmetic literals have been handled. To handle this negative literal, we generate the expression

$$E_j \quad = \quad E_i \bowtie (\Pi_{A_1, A_2}(E_i) - q)$$

where A_1 and A_2 are the attribute names of two columns in E_i which correspond to X and Y respectively.

Now let us consider constants occurring in a negative literal. Consider a negative literal of the form $\neg q(a, b, Y)$ where a and b are constants. Then, in the above expression defining E_j we replace q by $\sigma_{A_1 = a \wedge A_2 = b}(q)$.

Proceeding in a similar fashion, the remaining negative literals are processed, finally resulting in an expression E_w.

Finally the desired attributes are projected out of the expression. The attributes in E_w corresponding to the variables in the head of the rule become the projection attributes.

Thus our example rule finally becomes the view:-

create view *query* **as**
$$\Pi_{w1.person\text{-}name,\, w2.person-name}(E_2)$$

If there are multiple rules for the same predicate, the relational-algebra expression defining the view is the union of the expressions corresponding to the individual rules.

The above conversion can be extended to handle rules that satisfy some weaker forms of the safety conditions, and where some restricted cases where the variables in arithmetic predicates do not appear in a positive non-arithmetic literal.