Chapter 2

RELATIONAL OPERATORS

The update operations defined in the last chapter are not so much operations on relations as operations on tuples. In this chapter we shall consider operators that involve the entire relation. First, we see how the usual Boolean operations on sets apply to relations, and second, we consider three operators particular to relations: select, project, and join.

2.1 BOOLEAN OPERATIONS

Two relations on the same scheme can be considered sets over the same universe, the set of all possible tuples on the relation scheme. Thus, Boolean operations can be applied to two such relations. If r and s are relations on the scheme R, then $r \cap s$, $r \cup s$ and $r - s$ are all the obvious relations on R. The set $r \cap s$ is the relation $q(R)$ containing all tuples that are in both r and s, $r \cup s$ is the relation $q(R)$ containing all tuples that are in either r or s, and $r - s$ is the relation $q(R)$ containing those tuples that are in r but not in s. Note that intersection can be defined in terms of set difference: $r \cap s = r - (r - s)$.

Let $dom(R)$ be the set of all tuples over the attributes of R and their domains. We can define the *complement* of a relation $r(R)$ as $\bar{r} = dom(R) - r$. However, if any attribute A in R has an infinite domain, $\bar{r}$ will also be infinite and not a relation in our sense. We define a modified version of complementation that always yields a relation. If $r(A_1 A_2 \cdots A_n)$ is a relation and $D_i = dom(A_i)$, $1 \leq i \leq n$, the *active domain* of A_i relative to r is the set

$$adom(A_i, r) = \{ d \in D_i \mid \text{there exists } t \in r \text{ with } t(A_i) = d \}.$$

Let $adom(R, r)$ be the set of all tuples over the attributes of R and their active domains relative to r. The *active complement* of r is $\tilde{r} = adom(R,r) - r$. Note that $\tilde{r}$ is always a relation.

Example 2.1 The following are two relations, r and s, on the scheme ABC:

r

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_2

s

A	B	C
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

The results of the operations $r \cap s$, $r \cup s$, and $r - s$ are shown below.

$r \cap s =$

A	B	C
a_1	b_2	c_1

$r \cup s =$

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2

$r - s =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

Given $\{a_1, a_2\}$, $\{b_1, b_2, b_3\}$, and $\{c_1, c_2\}$ as the domains of A, B, and C, the domain of R and the complement of r derived from the domain of R are as shown:

$dom(R) =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_1	b_3	c_1
a_1	b_3	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2
a_2	b_3	c_1
a_2	b_3	c_2

$\bar{r} = dom(R) - r =$

A	B	C
a_1	b_1	c_2
a_1	b_2	c_2
a_1	b_3	c_1
a_1	b_3	c_2
a_2	b_1	c_1
a_2	b_2	c_1
a_2	b_2	c_2
a_2	b_3	c_1
a_2	b_3	c_2

To derive the active complement of r (that is, $\tilde{r}$), note that the active domain of B in the relation r does not contain b_3. The active domain of r, and the active complement of r, are:

$adom(R) =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2

$\tilde{r} = adom(R) - r =$

A	B	C
a_1	b_1	c_2
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_2	c_1
a_2	b_2	c_2

It is difficult to imagine a natural situation where the complement of a relation would be meaningful, except perhaps for a unary (one-attribute) relation. Active complement might arise naturally. Suppose a company has a training program that has a group of employees working two weeks in each department. The information on who in the training program has completed time in which department could be stored in a relation *done*(EMPLOYEE TRAINED-IN). The relation $\widetilde{done}$ would tell who had not completed training in what department, provided every employee in the program and every department is mentioned in *done*. Active complement can also be used as a storage compression device, when the active complement of a relation has fewer tuples than the relation itself.

The set of all relations on a given scheme is closed under union, intersection, set difference, and active complement. However, not all these operations preserve keys (see Exercise 2.3).

2.2 THE SELECT OPERATOR

Select is a unary operator on relations. When applied to a relation r, it yields another relation that is the subset of tuples of r with a certain value on a specified attribute. Let r be a relation on scheme R, A an attribute in R, and a an element of $dom(A)$. Using mapping notation, $\sigma_{A=a}(r)$ ("select A equal to a on r") is the relation $r'(R) = \{t \in r \mid t(A) = a\}$.

Example 2.2 Table 2.1 duplicates the relation in Table 1.2.

Table 2.1 New version of *sched*(FLIGHTS).

sched(NUMBER	FROM	TO	DEPARTS	ARRIVES)
84	O'Hare	JFK	3:00p	5:55p
109	JFK	Los Angeles	9:40p	2:42a
117	Atlanta	Boston	10:05p	12:43a
213	JFK	Boston	11:43a	12:45p
214	Boston	JFK	2:20p	3:12p

Table 2.2 is the result of applying $\sigma_{\text{FROM}=\text{JFK}}$ to *sched*.

Table 2.2 Result of applying $\sigma_{\text{FROM}=\text{JFK}}$ to *sched*(FLIGHTS).

$\sigma_{\text{FROM}=\text{JFK}}(sched) =$	NUMBER	FROM	TO	DEPARTS	ARRIVES
	109	JFK	Los Angeles	9:40p	2:42a
	213	JFK	Boston	11:43a	12:45p

Select operators commute under composition. Let $r(R)$ be a relation and let A and B be attributes in R, with $a \in dom(A)$ and $b \in dom(B)$. The identity

$$\sigma_{A=a}(\sigma_{B=b}(r)) = \sigma_{B=b}(\sigma_{A=a}(r))$$

always holds. This property follows readily from the definition of select:

$$\sigma_{A=a}(\sigma_{B=b}(r)) = \sigma_{A=a}(\{t \in r \mid t(B) = b\}) =$$
$$\{t' \in \{t \in r \mid t(B) = b\} \mid t'(A) = a\} = \{t \in r \mid t(A) = a \text{ and } t(B) = b\} =$$
$$\{t' \in \{t \in r \mid t(A) = a\} \mid t'(B) = b\} = \sigma_{B=b}(\sigma_{A=a}(r)).$$

Since the order of selection is unimportant, we write $\sigma_{A=a} \circ \sigma_{B=b}$ as $\sigma_{A=a, B=b}$ and $\sigma_{A_1=a_1} \circ \sigma_{A_2=a_2} \circ \cdots \circ \sigma_{A_n=a_n}$ as $\sigma_{A_1=a_1, A_2=a_2, \ldots, A_n=a_n}$. (The A_i's need not be distinct. See Exercise 2.4.) If X is a set of attributes and x is an X-value, $\sigma_{X=x}$ is also legitimate notation, if we interpret x as a sequence of values rather than a mapping.

Select is distributive over the binary Boolean operations:

$$\sigma_{A=a}(r \gamma s) = \sigma_{A=a}(r) \gamma \sigma_{A=a}(s)$$

where $\gamma = \cap$, $\cup$, or $-$, and r and s are relations over the same scheme. We prove $\sigma_{A=a}(r \cap s) = \sigma_{A=a}(r) \cap \sigma_{A=a}(s)$:

$\sigma_{A=a}(r \cap s) = \sigma_{A=a}(\{t | t \in r \text{ and } t \in s\}) =$
$\{t' \in \{t | t \in r \text{ and } t \in s\} | t'(A) = a\} =$
$\{t | t \in r \text{ and } t(A) = a\} \cap \{t | t \in s \text{ and } t(A) = a\} =$
$\sigma_{A=a}(\{t | t \in r\}) \cap \sigma_{A=a}(\{t | t \in s\}) = \sigma_{A=a}(r) \cap \sigma_{A=a}(s).$

The order of selection and complementation does make a difference in the result (see Exercise 2.5).

2.3 THE PROJECT OPERATOR

Project is also a unary operator on relations. Where select chooses a subset of the rows in a relation, project chooses a subset of the columns. Let r be a relation on scheme R, and let X be a subset of R. The *projection of* r *onto* X, written $\pi_X(r)$, is the relation $r'(X)$ obtained by striking out columns corresponding to attributes in $R - X$ and removing duplicate tuples in what remains. In mapping notation, $\pi_X(r)$ is the relation $r'(X) = \{t(X) | t \in r\}$.

If two projections are performed in a row, the latter subsumes the former: If π_Y is applied to the result of applying π_X to r, the result is the same as if π_Y were applied directly to r, if the original application of π_Y was proper. More precisely, given $r(R)$ and $Y \subseteq X \subseteq R$, $\pi_Y(\pi_X(r)) = \pi_Y(r)$. Similarly, for a string of projections, only the outermost need be considered for evaluation. If $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_m \subseteq R$, then

$$\pi_{X_1}(\pi_{X_2}(\cdots(\pi_{X_m}(r))\cdots)) = \pi_{X_1}(r).$$

Example 2.3 The following are the relations

1. $\pi_{\{\text{DEPARTS},\text{ARRIVES}\}}(sched)$,
2. $\pi_{\text{DEPARTS}}(\pi_{\{\text{DEPARTS},\text{ARRIVES}\}}(sched)) = \pi_{\text{DEPARTS}}(sched)$, and
3. $\pi_{\text{FROM}}(sched)$,

for the relation *sched* in Table 2.1.

1. $\pi_{\{\text{DEPARTS},\text{ARRIVES}\}}(sched) =$

(DEPARTS	ARRIVES)
3:00p	5:55p
9:40p	2:42a
10:05p	12:43a
11:43a	12:45p
2:20p	3:12p

2. $\pi_{\text{DEPARTS}}(sched) =$

DEPARTS
3:00p
9:40p
10:05p
11:43a
2:20p

3. $\pi_{\text{FROM}}(sched) =$

FROM
O'Hare
JFK
Atlanta
Boston

Projection commutes with selection when the attribute or attributes for selection are among the attributes in the set onto which the projection is taking place. If $A \in X$, $X \subseteq R$, and r is a relation on R, then

$$\pi_X(\sigma_{A=a}(r)) = \pi_X(\{t \in r | t(A) = a\}) = \{t'(X) | t' \in \{t \in r | t(A) = a\}\} = \{t(X)\ t \in r \text{ and } t(A) = a\} = \sigma_{A=a}(\{t(X) | t \in r\}) = \sigma_{A=a}(\pi_X(r)).$$

This identity does not hold when A is not an element of X (see Exercise 2.7).

The connection between projection and Boolean operations is treated in Exercises 2.8 and 2.9.

2.4 THE JOIN OPERATOR

Join is a binary operator for combining two relations. We illustrate its workings with an example. Suppose our imaginary airline maintains a list of which types of aircraft may be used on each flight and a list of the types of aircraft each pilot is certified to fly. These lists are stored as the relations *usable*(FLIGHT EQUIPMENT) and *certified*(PILOT EQUIPMENT). Table 2.3 shows sample states of these relations.

Table 2.3 Sample states of the relations *usable*(FLIGHT EQUIPMENT) and *certified*(PILOT EQUIPMENT).

usable(FLIGHT	EQUIPMENT)	*certified*(PILOT	EQUIPMENT)
83	727	Simmons	707
83	747	Simmons	727
84	727	Barth	747
84	747	Hill	727
109	707	Hill	747

We want a list showing which pilots can be used for each flight. We create a relation *options* on the scheme {FLIGHT, EQUIPMENT, PILOT} from the relations *usable* and *certified* by combining rows with the same value for EQUIPMENT. *Options* is shown in Table 2.4.

Once the relations are combined, if the EQUIPMENT values are no longer needed, we can compute $\pi_{\{\text{FLIGHT},\text{PILOT}\}}(options)$, as shown in Table 2.5.

In general, join combines two relations on all their common attributes. Start with relations $r(R)$ and $s(S)$, with $RS = T$. The *join of* r *and* s, written $r \bowtie s$, is the relation $q(T)$ of all tuples t over T such that there are tuples $t_r \in r$ and $t_s \in s$ with $t_r = t(R)$ and $t_s = t(S)$. Since $R \cap S$ is a subset of both R and S, as a consequence of the definition $t_r(R \cap S) = t_s(R \cap S)$. Thus, every tuple in q is a combination of a tuple from r and a tuple from s with equal $(R \cap S)$-values.

Returning to Table 2.4, we see $options = usable \bowtie certified$. The definition of join does not require that R and S have a non-empty intersection. If $R \cap S = \emptyset$, then $r \bowtie s$ is the Cartesian product of r and s. Actually, the Cartesian product of two relations would be a set of ordered pairs of tuples.

Table 2.4 The relation *options* on the scheme {FLIGHT, EQUIPMENT, PILOT}.

options(FLIGHT	EQUIPMENT	PILOT)
83	727	Simmons
83	727	Hill
83	747	Barth
83	747	Hill
84	727	Simmons
84	727	Hill
84	747	Barth
84	747	Hill
109	707	Simmons

Table 2.5 Computation of $\pi_{\{\text{FLIGHT},\text{PILOT}\}}(options)$.

$\pi_{\{\text{FLIGHT},\text{PILOT}\}}(options) =$ (	FLIGHT	PILOT)
	83	Simmons
	83	Hill
	83	Barth
	84	Simmons
	84	Hill
	84	Barth
	109	Simmons

By Cartesian product we shall mean Cartesian product followed by the natural isomorphism from pairs of R-tuples and S-tuples to RS-tuples.

Example 2.4 Let r and s be as shown:

$r(A$	$B)$
a_1	b_1
a_2	b_1

$s(C$	$D)$
c_1	d_1
c_2	d_1
c_2	d_2

Then $r \bowtie s$ is seen to be:

$r \bowtie s =$

$(A$	B	C	$D)$
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_1
a_1	b_1	c_2	d_2
a_2	b_1	c_1	d_1
a_2	b_1	c_2	d_1
a_2	b_1	c_2	d_2

2.5 PROPERTIES OF JOIN

There are more properties of join than we have room to list. We shall give some of them here, use others for exercises, and leave the rest for the reader to discover.

The join operation can be used to simulate selection. Given $r(R)$, suppose we wish to find $\sigma_{A=a}(r)$. First define a new relation $s(A)$, with a single tuple t, such that $t(A) = a$. Then $r \bowtie s$ is the same as $\sigma_{A=a}(r)$. The intersection of R and A is A, so

$$r \bowtie s = \{t \mid \text{there exists } t_r \in r \text{ and } t_s \in s \text{ such that } t_r = t(R) \text{ and } t_s = t(A)\} = \{t \mid \text{there exists } t_r \in r \text{ with } t_r = t(R) \text{ and } t(A) = a\} = \{t \in r \mid t(A) = a\} = \sigma_{A=a}(r).$$

We can also manufacture a generalized select operation using join. Let $s(A)$ now be a relation with k tuples, $t_1, t_2, \ldots, t_k$, where $t_i(A) = a_i$ and $a_i \in dom(A)$, $1 \leq i \leq k$. Then

$$r \bowtie s = \sigma_{A=a_1}(r) \cup \sigma_{A=a_2}(r) \cup \cdots \cup \sigma_{A=a_k}(r).$$

If we choose two attributes A and B from R and let $s(AB)$ be the relation with the single tuple t such that $t(A) = a$ and $t(B) = b$, then

$$r \bowtie s = \sigma_{A=a, B=b}(r).$$

There are other variations of selection available by adding columns and tuples to s.

It can be seen that the join operator is commutative from the symmetry in its definition. It is also associative. Given relations q, r, and s,

$$(q \bowtie r) \bowtie s = q \bowtie (r \bowtie s)$$

(see Exercise 2.11). Hence, we can write an unparenthesized string of joins without ambiguity.

We introduce some notation for multiple joins. Let $s_1(S_1)$, $s_2(S_2)$, $\ldots$, $s_m(S_m)$ be relations, with $R = S_1 \cup S_2 \cup \cdots \cup S_m$ and let **S** be the sequence $S_1, S_2, \ldots, S_m$. Let $t_1, t_2, \ldots, t_m$ be a sequence of tuples with $t_i \in s_i$, $1 \leq i \leq m$. We say tuples $t_1, t_2, \ldots, t_m$ are *joinable on* **S** if there is a tuple t on R such that $t_i = t(S_i)$, $1 \leq i \leq m$. Tuple t is the *result* of joining $t_1, t_2, \ldots, t_m$ on **S**.

Example 2.5 Tuples $\langle a_1\ b_1 \rangle$, $\langle b_1\ c_2 \rangle$, and $\langle a_1\ c_2 \rangle$ from relations s_1, s_2, and s_3, as shown

s_1(A	B)
a_1	b_1
a_1	b_2
a_2	b_1

s_2(B	C)
b_1	c_2
b_2	c_1

s_3(A	C)
a_1	c_2
a_2	c_2

are joinable with result $\langle a_1\ b_1\ c_2 \rangle$, and tuples $\langle a_1\ b_1 \rangle$, $\langle b_1\ c_2 \rangle$, and $\langle a_2\ c_2 \rangle$ are joinable with result $\langle a_2\ b_1\ c_2 \rangle$:

$s_1 \bowtie s_2 \bowtie s_3 =$

(A	B	C)
a_1	b_1	c_2
a_2	b_1	c_2

If $m = 2$ in the definition above, then if tuples t_1 and t_2 are joinable on $\mathbf{S} = S_1, S_2$ with result t, then $t_1 = t(S_1)$ and $t_2 = t(S_2)$. From the definition of join, t must be in $s_1 \bowtie s_2$. Conversely, if t is a tuple in $s_1 \bowtie s_2$, then there must be tuples t_1 and t_2 in s_1 and s_2, respectively, with t_1, t_2 joinable on **S** with result t. Hence, $s_1 \bowtie s_2$ consists of those tuples t that are the result of joining two tuples t_1, t_2 that are joinable on **S**.

Using the associativity of the join and induction, it is straightforward (but tedious) to prove the following result.

Lemma 2.1 The relation $s_1 \bowtie s_2 \bowtie \ldots \bowtie s_m$ consists of all tuples t that are the result of joining tuples $t_1, t_2, \ldots, t_m$ that are joinable on $\mathbf{S} = S_1, S_2, \ldots, S_m$.

Not every tuple of every relation may enter into the join. The relations $s_1, s_2, \ldots, s_m$ *join completely* if every tuple in each relation is a member of some list of tuples joinable on **S**.

Example 2.6 Example 2.5 shows a three-way join where the relations do not join completely. Tuple $\langle a_1\ b_2 \rangle$ of s_1 and tuple $\langle b_2\ c_1 \rangle$ of s_2 are left out of the join, for instance. If tuple $\langle a_1\ c_1 \rangle$ is added to s_3, then the relations do join completely, as shown below.

s_1

A	B
a_1	b_1
a_1	b_2
a_2	b_1

s_2

B	C
b_1	c_2
b_2	c_1

s_3

A	C
a_1	c_1
a_1	c_2
a_2	c_2

$s_1 \bowtie s_2 \bowtie s_3 =$

A	B	C
a_1	b_1	c_2
a_1	b_2	c_1
a_2	b_1	c_2

The join and project operators, though not inverses, do perform complementary functions. Let $r(R)$ and $s(S)$ be relations and let $q = r \bowtie s$. The scheme for q is RS. Let $r' = \pi_R(q)$. Is there any connection between r and r'? Yes, $r' \subseteq r$, since for any tuples t to be in q, $t(R)$ must be a tuple of r, and $r' = \{t(R) | t \in q\}$.

Example 2.7 The following shows that the containment is sometimes proper ($r' \subset r$).

$$
r(\underline{A \quad B}) \qquad s(\underline{B \quad C}) \qquad r \bowtie s = q(\underline{A \quad B \quad C})
$$
$$
\begin{array}{cc} a & b \\ a & b' \end{array} \qquad\qquad \begin{array}{cc} b & c \end{array} \qquad\qquad\qquad\qquad \begin{array}{ccc} a & b & c \end{array}
$$

$$
\pi_{AB}(q) = r'(\underline{A \quad B}) \qquad \begin{array}{cc} a & b \end{array}
$$

Next is a case where equality holds ($r' = r$).

$$
r(\underline{A \quad B}) \qquad s(\underline{B \quad C}) \qquad r \bowtie s = q(\underline{A \quad B \quad C})
$$
$$
\begin{array}{cc} a & b' \\ a & b' \end{array} \qquad\qquad \begin{array}{cc} b & c \\ b' & c' \end{array} \qquad\qquad\qquad\qquad \begin{array}{ccc} a & b & c \\ a & b' & c' \end{array}
$$

$$
\pi_{AB}(q) = r'(\underline{A \quad B}) \qquad \begin{array}{cc} a & b \\ a & b' \end{array}
$$

The containment becomes equality when for each tuple $t_r \in r$ there is a tuple $t_s \in s$ with $t_r(R \cap S) = t_s(R \cap S)$. Containment can also become equality without r and s joining completely (see Exercise 2.14). However, if $s' = \pi_S(q)$, then the condition $r = r'$ and $s = s'$ is exactly the same as r and s joining completely. This result generalizes to more than two relations (see Exercise 2.15).

What happens when we reverse the order and project and then join? Let q be a relation on RS, with $r = \pi_R(q)$ and $s = \pi_S(q)$. Let $q' = r \bowtie s$. If t is a tuple of q, then $t(R)$ is in r and $t(S)$ is in s, so t is also in q'. Therefore, $q' \supseteq q$. If $q' = q$, we say relation q *decomposes losslessly* onto schemes R and S. Lossless decomposition for more than two relations is treated in Chapter 6.

Example 2.8 The relation q in the second part of Example 2.7 decomposes losslessly onto AB and BC.

We can go one step further. Let $r' = \pi_R(q')$, $s' = \pi_S(q')$ and $q'' = r' \bowtie s'$. We are performing the project-join procedure twice on q to get q''. Let T be the intersection of R and S. Then $\pi_T(r) = \pi_T(\pi_R(q)) = \pi_T(q) = \pi_T(\pi_S(q)) = \pi_T(s)$, so r and s join completely, since for any tuple t_r in r, there must be a tuple t_s in s with $t_r(T) = t_s(T)$, and vice versa. Hence $r = r'$ and $s = s'$, so q' and q'' must be the same. Thus, the project-join procedure is idempotent: the result of applying it once is the result of applying it twice. This project-join procedure will be treated in more detail in Chapter 8.

Finally, we examine join in connection with union. Let r and r' be relations on R and let s be a relation on S. We claim that $(r \cup r') \bowtie s = (r \bowtie s) \cup (r' \bowtie s)$. Call the left side of the equation q and the right side q'. Given a tuple $t \in q$, there must be tuples t_r and t_s that are joinable with result t, with t_r in r or r' and $t_s \in s$. If $t_r \in r$, then t is in $r \bowtie s$. Otherwise, t is in $r' \bowtie s$. We have shown $q \subseteq q'$. For the other containment, $q \supseteq q'$, if t is in q', then t is in $r \bowtie s$ or $r' \bowtie s$. In either case, t is in $(r \cup r') \bowtie s$.

2.6 EXERCISES

2.1 Let $r(ABC)$ and $s(BCD)$ be relations, with a in $dom(A)$ and b in $dom(B)$. Which of the following expressions are properly formed?

(a) $r \cup s$
(b) $\pi_B(r) - \pi_B(s)$
(c) $\sigma_{B=b}(r)$
(d) $\sigma_{A=a, B=b}(s)$
(e) $r \bowtie s$
(f) $\pi_A(r) \bowtie \pi_D(s)$.

2.2 Relations r and s are given below.

$r(A$	B	$C)$
a	b	c
a	b'	c'
a	b''	c'
a'	b'	c

$s(B$	C	$D)$
b'	c'	d
b''	c'	c'
b''	c	d

Compute the values of the following expressions.

(a) $\tilde{r}$
(b) $\tilde{s}$
(c) $\sigma_{A=a}(r)$
(d) all the properly formed expressions in Exercise 2.1.

2.3 Let r and s be relations on scheme R with key K. Which of the following relations must necessarily have key K?

(a) $r \cup s$
(b) $r \cap s$

(c) $r - s$
(d) $\bar{r}$, where $\bar{r}$ is a relation
(e) $\pi_K(r)$
(f) $r \bowtie s$.

2.4 Let $r(R)$ be a relation with $A \in R$ and let $a, a' \in dom(A)$. Prove

$$[\sigma_{A=a,A=a'}(r) = \emptyset \text{ or } \sigma_{A=a,A=a'}(r) = \sigma_{A=a}(r)].$$

2.5 Let r and s be relations on R with $A \in R$ and let $a \in dom(A)$. Prove or disprove the following.

(a) $\sigma_{A=a}(\bar{r}) = \overline{\sigma_{A=a}(r)}$
(b) $\sigma_{A=a}(r \cap s) = \sigma_{A=a}(r) \cap s$.

2.6 Let r be a relation on $R[\underline{A}\ B\ C]$. What can be said about the size of $\sigma_{A=a}(r)$?

2.7 Let X be a subset of R, $A \in R$, $A \notin X$ and let r be a relation on R. Find a counterexample to

$$\pi_X(\sigma_{A=a}(r)) = \sigma_{A=a}(\pi_X(r)).$$

2.8 Let X be a subset of R and let r and s be relations on R. Prove or disprove the following equalities.

(a) $\pi_X(r \cap s) = \pi_X(r) \cap \pi_X(s)$
(b) $\pi_X(r \cup s) = \pi_X(r) \cup \pi_X(s)$
(c) $\pi_X(r - s) = \pi_X(r) - \pi_X(s)$
(d) $\pi_X(\bar{r}) = \overline{\pi_X(r)}$, where $\bar{r}$ is a relation.

2.9 For each of the disproved equalities in Exercise 2.8, try to prove containment in one direction.

2.10 Let A be an attribute in R, let $R' = R - A$ and let r be a relation on R. What relationships exist between the sizes of the relations r, $\sigma_{A=a}(r)$, $\pi_A(r)$, $\pi_{R'}(r)$ and $\sigma_{A=a}(\pi_A(r))$?

2.11 Given relations q, r and s, show

(a) $(q \bowtie r) \bowtie s = q \bowtie (r \bowtie s)$
(b) $q \bowtie q = q$
(c) $q \bowtie r = q \bowtie (q \bowtie r)$.

2.12 Let $r(R)$ and $s(S)$ be relations with $A \in R$. Prove

$$\sigma_{A=a}(r \bowtie s) = \sigma_{A=a}(r) \bowtie s.$$

2.13 Let r and r' be relations on R, and let s be a relation on S. Prove or disprove:

(a) $(r \cap r') \bowtie s = (r \bowtie s) \cap (r' \bowtie s)$
(b) $(r - r') \bowtie s = (r \bowtie s) - (r' \bowtie s)$
(c) $\tilde{r} \bowtie \tilde{s} = \widetilde{r \bowtie s}$.

2.14 Given relations $r(R)$, $s(S)$ and $q = r \bowtie s$, show that $\pi_R(q) = r$ can hold without r and s joining completely.

2.15* Let $s_1(S_1)$, $s_2(S_2)$, $\ldots$, $s_m(S_m)$ be relations and let $q = s_1 \bowtie s_2 \bowtie \cdots \bowtie s_m$. Prove that $s_1, s_2, \ldots, s_m$ join completely if and only if $s_i = \pi_{S_i}(q)$, $1 \leq i \leq m$.

2.16 Let $q(R)$ be a relation and let S_i be a subset of R, $1 \leq i \leq m$. Define $s_i = \pi_{S_i}(q)$, $1 \leq i \leq m$. Prove $s_1, s_2, \ldots, s_m$ join completely.

2.17 Let q be a relation on RS. Give an example of when the containment

$$q \subseteq \pi_R(q) \bowtie \pi_S(q)$$

is proper.

2.18 Given relations $r(R)$, $s(S)$ and $q = r \bowtie s$, define $r' = \pi_R(q)$ and $s' = \pi_S(q)$. Prove

$$q = r' \bowtie s'.$$

2.19* Given a relation $q(RS)$, find a sufficient condition for

$$q = \pi_R(q) \bowtie \pi_S(q).$$

Is your condition necessary?

2.7 BIBLIOGRAPHY AND COMMENTS

The relational operators select, project, and join in the form here were introduced by Codd [1970, 1972b], although analogs are given by Childs [1968] for a slightly different model. Join is sometimes called *natural join* to distinguish from other join-like operations, which we shall see in Chapter 3. In some sources, relations are treated in the traditional mathematical fashion, with ordered tuples and component denoted by number. We shall not make use of this treatment.