Chapter 3

MORE OPERATIONS ON RELATIONS

In this chapter we shall study some relational operators that are less elementary than those in Chapter 2. Some of the operators are generalizations of those in Chapter 2; others can be shown equivalent to a series of those operators. These operators, along with a set of relations and constants, will form a relational algebra. We shall see that we can restrict the set of operators and still retain the expressive power of relational algebra. Finally, we examine two operators that, while not part of the algebra, are sometimes useful in database implementations.

3.1 THE DIVIDE OPERATOR

The divide operator has a rather complex definition, but it does have some applications in natural situations.

Definition 3.1 Let $r(R)$ and $s(S)$ be relations, with $S \subseteq R$. Let $R' = R - S$. Then r *divided by* s, written $r \div s$, is the relation

$$r'(R') = \{t | \text{for every tuple } t_s \in s \text{ there is a tuple } t_r \in r \text{ with } t_r(R') = t \text{ and } t_r(S) = t_s\}.$$

Relation r' is the *quotient* of r divided by s. Another way to state the definition is that $r \div s$ is the maximal subset r' of $\pi_{R'}(r)$ such that $r' \bowtie s$ is contained in r. The join in this case is a Cartesian product. An example should clarify the definition.

Example 3.1 Table 3.1 is another instance of the relation *certified*(PILOT EQUIPMENT) given in Table 2.3. Suppose we want to find those pilots who can fly all the types of aircraft in some set. Let q(EQUIPMENT) and s(EQUIPMENT) be as follows:

q(EQUIPMENT)
707
727
747

s(EQUIPMENT)
707

Table 3.1 An instance of the relation *certified*(PILOT EQUIPMENT).

certified (PILOT	EQUIPMENT)
Desmond	707
Desmond	727
Desmond	747
Doyle	707
Doyle	727
Davis	707
Davis	727
Davis	747
Davis	1011
Dow	727

Division can then be used to garner information on what pilots can fly the types of aircraft in q, or to find what pilots can fly the aircraft in s.

certified $\div q = q'$(PILOT)	*certified* $\div s = s'$(PILOT)
Desmond	Desmond
Davis	Doyle
	Davis

Division can be expressed in terms of the operators from Chapter 2 (see Exercise 3.3).

3.2 CONSTANT RELATIONS

In discussing join in the last chapter, we showed that the effect of select can be obtained by join with a constant relation. We have a notation for representing constant relations directly in expressions. If $A_1, A_2, \cdots, A_n$ are distinct attributes, and c_i is a constant from $dom(A_i)$ for $1 \leq i \leq n$, then

$$\langle c_1 : A_1 \; c_2 : A_2 \cdots c_n : A_n \rangle$$

represents the constant tuple $\langle c_1 \; c_2 \cdots c_n \rangle$ over scheme $A_1 \; A_2 \cdots A_n$. We represent a constant relation over scheme $A_1 \; A_2 \cdots A_n$ as a set of tuples. Let c_{ij} be a constant in $dom(A_i)$ for $1 \leq i \leq n$ and $1 \leq j \leq k$. Then

$$\{\langle c_{11} : A_1 \; c_{21} : A_2 \cdots c_{n1} : A_n \rangle,$$
$$\langle c_{12} : A_1 \; c_{22} : A_2 \cdots c_{n2} : A_n \rangle, \cdots$$
$$\langle c_{1k} : A_1 \; c_{2k} : A_2 \cdots c_{nk} : A_n \rangle\}$$

represents the relation we would normally write as

$(A_1$	A_2	$\cdots$	$A_n)$
c_{11}	c_{21}	$\cdots$	c_{n1}
c_{12}	c_{22}	$\cdots$	c_{n2}
$\vdots$	$\vdots$		$\vdots$
c_{1k}	c_{2k}	$\cdots$	c_{nk}.

In the case of a single-tuple constant relation, we shall sometimes omit the set brackets. For a single-attribute tuple, we shall sometimes omit the wickets ("⟨" and "⟩").

A constant relation of any number of tuples and any number of attributes can be built up from single-tuple, single-attribute constant relations through join and union.

Example 3.2 The relation shown below

(PILOT	EQUIPMENT)
Desmond	707
Davis	707

can be represented as

$$(\langle \text{Desmond:PILOT} \rangle \bowtie \langle \text{707:EQUIPMENT} \rangle) \cup (\langle \text{Davis:PILOT} \rangle \bowtie \langle \text{707:EQUIPMENT} \rangle).$$

3.3 RENAMING ATTRIBUTES

Consider the relation *usedfor* in Table 3.2, which tells what plane will be used for a given flight on a given day. Suppose we want to know all the pairs of flights that are scheduled to use the same plane on the same day. What we need is a join of *usedfor* with itself, but ignoring connections on the FLIGHT column. We can accomplish this join with a copy of *usedfor* where FLIGHT is renamed to, say, FLIGHT2.

To specify such a relation, we introduce a *renaming* operator δ. Let r be a relation on scheme R, where A is an attribute in R and B is an attribute not in $R - A$. Let $R' = (R - A)B$. Then r *with* A *renamed to* B, denoted $\delta_{A \leftarrow B}(r)$, is the relation

$$r'(R') = \{t' \mid \text{there is a tuple } t \in r \text{ with } t'(R - A) = t(R - A) \text{ and } t'(B) = t(A)\}.$$

We require that A and B have the same domain.

Table 3.2 The relation *usedfor*, telling what plane will be used for a given flight.

usedfor(FLIGHT	DATE	PLANENUM)
12	6 Jan	707-82
12	7 Jan	707-82
13	6 Jan	707-82
26	6 Jan	747-16
26	7 Jan	747-18
27	6 Jan	747-16
27	7 Jan	747-2
60	6 Jan	707-82
60	7 Jan	727-6

Example 3.3 An expression that denotes the relation with the desired pairs of flights is

$$s = \pi_{\{\text{FLIGHT},\text{FLIGHT2}\}}(\mathit{usedfor} \bowtie \delta_{\text{FLIGHT}-\text{FLIGHT2}}(\mathit{usedfor})).$$

The value for s using *usedfor* as in Table 3.2 is given in Table 3.3. In Section 3.5.1 we shall see a generalization of the select operator that can be used to remove the redundancy in relation s (see Exercise 3.7).

Table 3.3 Relation s, showing what pairs of flights use the same plane.

s(FLIGHT	FLIGHT2)
12	13
13	12
12	60
60	12
13	60
60	13
12	12
13	13
60	60
26	27
27	26
26	26
27	27

Let r be a relation on R. Let $A_1, A_2, \ldots, A_k$ be distinct attributes in R and let $B_1, B_2, \ldots, B_k$ be distinct attributes not in $R - (A_1\, A_2 \cdots A_k)$, where

$dom(A_i) = dom(B_i)$ for $1 \leq i \leq k$. We denote the simultaneous renaming of the attributes $A_1, A_2, \ldots, A_k$ to $B_1, B_2, \ldots, B_k$, respectively, in r by

$$\delta_{A_1, A_2, \ldots, A_k \leftarrow B_1, B_2, \ldots, B_k}(r).$$

Note that a simultaneous renaming sometimes cannot be written as a sequence of single-attribute renamings without introducing another attribute symbol. The renaming $\delta_{A,B \leftarrow B,A}$ is an example.

3.4 THE EQUIJOIN OPERATOR

As the join operator was defined in Chapter 2, relations may only be combined on identically named columns and must be combined on all such columns. In the last section we saw how to join on a subset of those columns. Relations can also be combined on columns with different attribute names but equal domains.

Example 3.4 Consider the relations *routes* and *based* in Table 3.4 and Table 3.5.

Table 3.4 The relation *routes*.

routes (NUMBER	FROM	TO)
84	O'Hare	JFK
109	JFK	Los Angeles
117	Atlanta	Boston
213	JFK	Boston
214	Boston	JFK

Table 3.5 The relation *based*.

based (PILOT	AIRPORT)
Terhune	JFK
Temple	Atlanta
Taylor	Atlanta
Tarbell	Boston
Todd	Los Angeles
Truman	O'Hare

Routes is a projection of the relation *sched* in Table 2.1. *Based* gives the home base for each pilot. Suppose we want to assign pilots to flights that originate at the pilots' home bases. We need a relation showing which pilots are based in the origin city of each flight. Table 3.6 shows such a relation.

Table 3.6 The relation *canfly*, showing which pilots live in the origin city of each flight.

canfly(NUMBER	FROM	TO	PILOT	AIRPORT)
84	O'Hare	JFK	Truman	O'Hare
109	JFK	Los Angeles	Terhune	JFK
117	Atlanta	Boston	Temple	Atlanta
117	Atlanta	Boston	Taylor	Atlanta
213	JFK	Boston	Terhune	JFK
214	Boston	JFK	Tarbell	Boston

We have taken an *equijoin* on the columns corresponding to attribute names FROM and AIRPORT.

We give a general description of equijoin. Let $r(R)$ and $s(S)$ be relations with $A_i \in R$, $B_i \in S$, and $dom(A_i) = dom(B_i)$, $1 \leq i \leq m$. The A_i's need not be distinct, nor need the B_i's. The equijoin of r and s on $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_m$, written $r[A_1 = B_1, A_2 = B_2, \ldots, A_m = B_m]s$, is the relation

$$q(RS) = \{t | \text{there exists } t_r \in r \text{ and } t_s \in s \text{ with } t(R) = t_r \text{ and } t(S) = t_s \text{ and } t(A_i) = t(B_i),\ 1 \leq i \leq m\}.$$

Example 3.5 The relation *canfly* in Table 3.6 is

$$routes[\text{FROM}=\text{AIRPORT}]based.$$

This definition needs a little refinement. There could be an attribute A such that $A \in R$ and $A \in S$. In the equijoin of r and s, we want a column for each occurrence of A. We require that $R \cap S = \emptyset$ in the definition. This is not a great restriction, since if R and S do have a non-empty intersection, we can rename attributes in r or s to make the intersection of schemes empty. Note that there need not be any comparisons in the equijoin; m can be 0 in the definition. The equijoin $r[]s$ is simply the Cartesian product of r and s.

To emphasize the distinction between join as defined in Chapter 2 and equijoin, we sometimes call the former *natural join*. Equijoin is mainly a con-

venience, for it can be expressed in terms of renaming and natural join (see Exercise 3.5). Natural join can also be expressed using equijoin. For example, given relations $r(ABC)$ and $s(BCD)$ and attributes B' and C' with $dom(B) = dom(B')$ and $dom(C) = dom(C')$,

$$r \bowtie s = \pi_{ABCD}(r[B = B', C = C'] \delta_{B,C-B',C'}(s)).$$

The main difference between natural join and equijoin is that natural join does not repeat the connected columns.

3.5 EXTENSIONS FOR OTHER COMPARISONS ON DOMAINS

Up to this point, the only comparison between domain values we have been making is one for equality. We could also compare domain values using inequality. Often, domains are ordered, and in those cases, the comparisons $<$, $\leq$, $\geq$, and $>$ also make sense. For a general treatment of such comparisons, we posit a set Θ of *comparators*: binary relations (in the mathematical sense) over pairs of domains. If θ is a comparator in Θ, and A and B are attributes, we say A is *θ-comparable* with B if θ is over $dom(A) \times dom(B)$. We write "A is θ-comparable" to mean A is θ-comparable with itself. We assume every attribute A is equality-comparable and inequality-comparable.

We generally will only be concerned with the comparators $=$, $\neq$, $<$, $\leq$, $\geq$, and $>$ over a single domain. However, we use these symbols in a generic sense; for example, "$=$" actually represents different equality comparators for different domains. We shall use comparators to generalize selection and join to comparisons other than equality.

It is a somewhat artificial restriction to require that our comparators be binary relations. There are reasonable tests we might like to make that are represented by mathematical relations of degree other than two. For example, we might want the unary relation m, on the domain of times, where $h \in m$ means h is a morning time; or the ternary relation w on integers, where $w(i, j, k)$ means $i \leq j \leq k$. Any unary relation θ can be represented by a binary relation θ', where $\theta(a)$ if and only if $\theta'(a, a)$, and for no a, b where $a \neq b$ does $\theta'(a, b)$ hold. Some ternary and higher order relations, such as w, can be represented as the conjunction of binary relations, while others cannot (see Exercise 3.9). While the extension to comparisons based on relations other than binary is straightforward, the notation is messy, and we wish to keep our theorems neat.

3.5.1 Extending Selection

We extend our notation for the select operator to be $\sigma_{A\theta a}$, where θ is a comparator in Θ. If r is a relation on scheme R, and A an attribute of R, and a is a constant in $dom(B)$, where A and B are θ-comparable, then $\sigma_{A\theta a}(r) = \{t \in r | t(A)\ \theta\ a\}$. We use infix notation for comparators: $t(A)\theta a$ means $\theta(t(A), a)$.

Example 3.6 A relation *times*, which is a projection of the relation *sched* in Table 2.1, is shown below.

times(NUMBER	DEPARTS	ARRIVES)
84	3:00p	5:55p
109	9:40p	2:42a
117	10:05p	12:43a
213	11:43a	12:45p
214	2:20p	3:12p

Relation $s = \sigma_{\text{ARRIVES} \leq 1:00\text{p}}(times)$, assuming $\leq$ orders the hours of the day from 12:01a to midnight, is as follows:

$\sigma_{\text{ARRIVES} \leq 1:00\text{p}}(times) = s$

(NUMBER	DEPARTS	ARRIVES)
109	9:49p	2:42a
117	10:05p	12:43a
213	11:43a	12:45p

It is a list of all flights and times that arrive at or before 1:00p.

Besides comparisons between an attribute and a constant, we also allow comparisons between two attributes. Let r be a relation on R, where A and B are attributes in R. Let $\theta \in \Theta$ be a comparator such that A and B are θ-comparable. Then $\sigma_{A\theta B}(r) = \{t \in r | t(A)\ \theta\ t(B)\}$.

Example 3.7 Let "$\ll$" be the comparator on times of day meaning "precedes by at least 2 hours." Then, for *times* as given in Example 3.6, $s = \sigma_{\text{DEPARTS} \ll \text{ARRIVES}}(times)$ is given below.

s(NUMBER	DEPARTS	ARRIVES)
84	3:00p	5:55p
109	9:40p	2:42a
117	10:05p	12:43a

We let times of day wrap around midnight for $\ll$.

As before, we can abbreviate a series of selections. For example, $\sigma_{A \leq a}(\sigma_{B>D}(\sigma_{C=c}(r)))$ becomes $\sigma_{A \leq a, B>D, C=c}(r)$.

To give ourselves even more convenience, we allow the logical connectives $\wedge$, $\vee$, $\neg$ (and, or, not), and parentheses. For example, $\sigma_{((A=a) \vee (A>c)) \wedge (B \neq b)}(r)$.

The commas we used before were actually implicit ands. The logical connectives, while convenient, do not add any expressive power to our set of relational operators (see Exercise 3.10).

3.5.2 The Theta-Join Operator

The equijoin extends the join operator to handle comparisons between columns with different attribute names. With other comparators, we need not restrict ourselves merely to comparing for equality.

Example 3.8 Suppose we have a list of flights and times from city a to city b, and a similar list of flights and times from city b to city c. Table 3.7 and Table 3.8 show these lists, represented by two relations, *timesab* and *timesbc*.

Table 3.7 Flights between city a and city b.

timesab(NUMBER	DEPARTS	ARRIVES)
60	9:40a	11:45a
91	12:50p	2:47p
112	4:05p	6:15p
306	8:30p	10:25p
420	9:15p	11:11p

Table 3.8 Flights between city b and city c.

timesbc(NUMBER	DEPARTS	ARRIVES)
11	8:30a	9:52a
60	12:25p	1:43p
156	4:20p	5:40p
158	7:10p	8:35p

We want to know which flights from a to b connect with flights from b to c. We combine tuples from *timesab* and *timesbc* when the flight from a to b arrives at b before the flight from b to c departs from b. Table 3.9 shows the result, relation *connectac*. Note that we must first rename attributes in *timesbc*, and that we are not looking for connections over midnight.

Table 3.9 Flight connections between city a and city c at city b.

connectac (NUMBER	DEPARTS	ARRIVES	NUMBER′	DEPARTS′	ARRIVES′)
60	9:40a	11:45a	60	12:25p	1:43p
60	9:40a	11:45a	156	4:20p	5:40p
60	9:40a	11:45a	158	7:10p	8:35p
91	12:50p	2:47p	156	4:20p	5:40p
91	12:50p	2:47p	158	7:10p	8:35p
112	4:05p	6:15p	158	7:10p	8:35p

Let $r(R)$ and $s(S)$ be two relations we want to combine, where $R \cap S = \emptyset$. Let $A \in R$ and $B \in S$ be θ-comparable for θ in Θ. Then $r[A\theta B]s$ is the relation

$$q(RS) = \{t | \text{for some } t_r \in r \text{ and some } t_s \in s \text{ such that } t_r(A)\ \theta\ t_s(B), \\ t(R) = t_r \text{ and } t(S) = t_s\}.$$

Example 3.9 For the relation in Table 3.9,

$$connectac = timesab[\text{ARRIVES} < \text{DEPARTS}']timesbc',$$

where

$$timesbc' = \delta_{\text{NUMBER,ARRIVES,DEPARTS}\leftarrow\text{NUMBER}',\text{ARRIVES}',\text{DEPARTS}'}(timesbc).$$

When we want a number of comparisons to take place, we write them all between the brackets. For example, $r[A_1 < B_1, A_2 = B_2, A_3 \geq B_2]s$. We call any such join a *theta-join*. Equijoin is a special case of theta-join.

3.6 RELATIONAL ALGEBRA

We refer to the operators union, intersection, difference, active complement, select, project, natural join, division, renaming, and theta-join, along with constant relations and regular relations, as the *relational algebra*. Any expression legally formed using these operators and relations is an *algebraic expression*. Given an algebraic expression E, and the current values of all the relations in E, we can evaluate E to yield a single relation. E represents a mapping from sets of relations to single relations.

Actually, the set of attributes, the domains, and the set of comparators we use limit the mappings we may define. In Chapter 10, where we compare the expressive power of relational algebra to other systems for operating on relations, these parameters will make a difference. In such cases, we must be a bit more formal.

Definition 3.2 Let $\mathbf{U}$ be a set of attributes, called the *universe*. Let $\mathfrak{D}$ be a set of domains, and let *dom* be a total function from $\mathbf{U}$ to $\mathfrak{D}$. Let $\boldsymbol{R} = \{R_1,$

$R_2, \ldots, R_p\}$ be a set of distinct relation schemes, where $R_i \subseteq \mathbf{U}$ for $1 \leq i \leq p$. Let $d = \{r_1, r_2, \ldots, r_p\}$ be a set of relations, such that r_i is a relation on R_i, $1 \leq i \leq p$. Let Θ be a set of comparators over domains in $\mathfrak{D}$, including at least the equality and inequality comparators for every domain. The *relational algebra over* $\mathbf{U}$, $\mathfrak{D}$, dom, $\mathbf{R}$, d, *and* Θ is the 7-tuple $\mathfrak{R} = (\mathbf{U}, \mathfrak{D}, dom, \mathbf{R}, d, \Theta, O)$, where O is the set of operators union, intersect, difference, active complement, project, natural join, and divide, and renaming using attributes in $\mathbf{U}$, select using comparators in Θ, and logical connectives and theta-join using comparators in Θ. An *algebraic expression over* $\mathfrak{R}$ is any expression formed legally (according to the restrictions on the operators) from the relations in d and constant relations over schemes in $\mathbf{U}$, using the operators in O.

We allow parentheses in algebraic expressions, and assume no precedence of the binary operators, except for the usual precedence of $\cap$ over $\cup$. We also may omit parentheses for strings of relations connected by the same operator, if the operation is associative. Note that we do not allow two relations with the same scheme. We discuss this restriction again in Chapter 12.

The relation names $r_1, r_2, \ldots, r_p$ are analogous to program variables, where r_i ranges over relations on scheme R_i. Our notation is a bit ambiguous, in that we use r_i both as a relation name and to denote the current state of a relation. The same ambiguity arises when discussing variables in programs; this is the problem denotational semantics tries to address. The ambiguity only gets clumsy when we view an algebraic expression as a mapping.

3.6.1 Algebraic Expressions as Mappings

Since the result of every relational operation we use is a single relation, every algebraic expression defines a function that maps a set of relations to a single relation. The scheme of the single relation depends only on the schemes for the set of relations. Let the *scheme* of an algebraic expression E, denoted $sch(E)$, be the relation scheme of the relation.

We can define $sch(E)$ recursively according to the following rules.

1. If E is r_i, then $sch(E)$ is the relation scheme for r_i.
2. If E is a constant relation, $sch(E)$ is the scheme for the constant relation.
3. If $E = E_1 \cup E_2$, $E_1 \cap E_2$, $E_1 - E_2$, $\tilde{E}_1$, or $\sigma_C(E_1)$, where C is some set of conditions, then $sch(E) = sch(E_1)$.
4. If $E = \pi_X(E_1)$, then $sch(E) = X$.
5. If $E = E_1 \div E_2$, then $sch(E) = sch(E_1) - sch(E_2)$.

6. If $E = E_1 \bowtie E_2$ or $E_1[C]E_2$, for some set of conditions C, then $sch(E) = sch(E_1) \cup sch(E_2)$.
7. If $E = \delta_{A_1,A_2,\ldots,A_k \leftarrow B_1,B_2,\ldots,B_k}(E_1)$, then $sch(E) = (sch(E_1) - A_1A_2 \cdots A_k)\, B_1B_2 \cdots B_k$.

If E is an algebraic expression involving relation names $s_1, s_2, \ldots, s_q$, corresponding to schemes $S_1, S_2, \ldots, S_q$, then E is a mapping

$$E\colon Rel(S_1) \times Rel(S_2) \times \cdots \times Rel(S_q) \rightarrow Rel(sch(E)),$$

where $Rel(R)$ is the set of all relations with scheme R. We shall sometimes use $E(s_1, s_2, \ldots, s_q)$ to denote the value of E on the set of relations named by $s_1, s_1, \ldots, s_q$.

Sometimes we shall want to use the complement operator in expressions. If we add complement to our set of operators, we get a *relational algebra with complement.* An algebraic expression E involving complement potentially maps a set of relations to an infinite relation. We shall not use complement after this chapter until Chapter 10.

3.6.2 Restricting the Set of Operators

As we have seen numerous times, the relational operators are in no sense independent. There are restricted sets of operators that have all the power of the full set. One such set is given by the next theorem.

Theorem 3.1 Let E be an expression over relational algebra $\mathfrak{R}$ that uses relation names $s_1, s_2, \ldots, s_q$. There is an expression E' over $\mathfrak{R}$ that defines the same function of $s_1, s_2, \ldots, s_q$ and uses only single-attribute, single-tuple constant relations, select with a single comparison, natural join, project, union, difference, and renaming.

Proof By what we noted in Section 3.2, we can replace every constant relation in E by an expression involving union, join, and single-attribute, single-tuple constant relations. Exercise 3.13 shows that theta-join can be replaced by natural join and selection. Exercise 3.10 shows how to replace any generalized selection with an expression involving single-comparison selections and other relational operators, not including theta-join. Exercise 3.3a shows how to express division in terms of operators from Chapter 2.

In Section 2.1 we saw that intersection can be replaced by difference. The only operator left to replace in E to get E' is active complement. Active complement can be expressed with project, join, and difference. For example, suppose E_1 is an algebraic expression where $sch(E_1) = ABC$. Then $\tilde{E}_1$ is

$$(\pi_A(E_1) \bowtie \pi_B(E_1) \bowtie \pi_C(E_1)) - E_1.$$

Note that the joins are Cartesian products.

Corollary Let E be an expression over relational algebra $\mathfrak{R}$ with complement that uses relation names $s_1, s_2, \ldots, s_q$. There is an expression E' over $\mathfrak{R}$ that defines the same function of $s_1, s_2, \ldots, s_q$ and uses only single-attribute, single-tuple constant relations, select with a single comparison, natural join, project, union, complement, and renaming.

Proof By Theorem 3.1, the only operator that must be removed from E is difference. Note that $E_1 - E_2 = \overline{E_1 \cup E_2}$.

3.7 THE SPLIT OPERATOR

The split operator takes one relation as an argument and returns a pair of relations. We do not include it in relational algebra since we want the value of every expression in the algebra to be a single relation. Let r be a relation on scheme R and let $\beta(t)$ be a Boolean predicate on tuples over R. Then r *split on* β, written $\text{SPLIT}_\beta(r)$, is the pair of relations (s, s'), both with scheme R, where $s = \{t \in r | \beta(t)\}$ and $s' = \{t \in r | \text{not } \beta(t)\}$. Clearly, $s' = r - s$. We put no restrictions on what the predicate β may be, except that its value may only depend on tuple t and not on the state of r.

Example 3.10 The predicate, $\beta(t) =$ there exists t' in r with $t(A) \neq t'(A)$ would not be permissible, since it depends on other tuples in r.

Example 3.11 Consider the relation *certified* in Table 3.3. Let $\beta(t) =$ (t(EQUIPMENT)= 707 or t(EQUIPMENT) = 727). The relations s and s', where $\text{SPLIT}_\beta(\textit{certified}) = (s, s')$, are shown below.

s(PILOT	EQUIPMENT)
Desmond	707
Desmond	727
Doyle	707
Doyle	727
Davis	707
Davis	727
Dow	727

s'(PILOT	EQUIPMENT)
Desmond	747
Davis	747
Davis	1011

3.8 THE FACTOR OPERATOR

The *factor* operator takes one relation as an argument and generates two relations. The two relations, when joined, yield the original relation with an added column. We shall first demonstrate the factor operator by example.

Example 3.12 Consider a flight roster showing all the passengers booked on a flight, what class they are flying, and whether they are in the smoking or non-smoking section. We represent the flight roster as a relation *roster* on the scheme {PASSENGER, CLASS, SMOKING} as shown.

roster

PASSENGER	CLASS	SMOKING
Salazar	first	yes
Schick	first	no
Shockley	coach	no
Stewart	first	yes
Sayers	coach	no
Sands	coach	no
Sachs	coach	yes

There are only four possible {CLASS, SMOKING}-values. We can represent the same information in less space by splitting off the CLASS and SMOKING columns, and creating a new column, LINK, as shown below.

roster1

PASSENGER	LINK
Salazar	1
Schick	2
Shockley	4
Stewart	1
Sayers	4
Sands	4
Sachs	3

roster2

LINK	CLASS	SMOKING
1	first	yes
2	first	no
3	coach	yes
4	coach	no

It is easy to check that $roster = \pi_{\{\text{PASSENGER,CLASS,SMOKING}\}}(roster1 \bowtie roster2)$.

If r is a relation on scheme R and $B_1, B_2, \ldots, B_m$ are attributes of R, and L is an attribute not in R, we use the notation

$$\text{FACTOR}(r; B_1, B_2, \ldots, B_m; L)$$

to denote the operation of removing the columns corresponding to B_1, B_2,

..., B_m from r to form a new relation, and adding an extra column labeled L to r and the new relation on which to join. The relations *roster1* and *roster2* are the result of FACTOR(*roster*; CLASS, SMOKING; LINK).

We shall not specify the factor operator more formally. It main use is as a conceptual tool for finding efficient ways to store a relation. Again, we do not include this operator in the relational algebra, because it does not yield a single relation as its result.

3.9 EXERCISES

3.1 Let $r(R)$ and $s(S)$ be relations where $R \cap S = \emptyset$. Prove

$$(r \bowtie s) \div s = r.$$

3.2 Let r be a relation on scheme R and let s and s' be relations on scheme S, where $R \supseteq S$. Show that if $s \subseteq s'$, then

$$r \div s \supseteq r \div s'.$$

Show that the converse is false.

3.3* Let $r(R)$ and $s(S)$ be relations with $R \supseteq S$ and let $R' = R - S$. Prove the identities

a) $r \div s = \pi_{R'}(r) - \pi_{R'}((\pi_{R'}(r) \bowtie s) - r)$.
b) $r \div s = \bigcap_{t \in s} \pi_{R'}(\sigma_{S=t}(r))$.

3.4 For relation r with the scheme shown in Table 3.2, give an expression that, for a given flight f, evaluates to a relation on scheme FLIGHT giving all the flights that use the same plane as flight f on every date for flight f listed in r.

3.5 Show that any equijoin can be specified in terms of natural join and renaming, given sufficient extra attributes with the correct domains.

3.6 It is sometimes meaningful to equijoin a relation with itself. Compute relation r = *routes*[TO = FROM']*routes'* where *routes* is the relation in Table 3.4, and *routes'* is *routes* with all attributes renamed to primed versions. Using r, compute the relation $s = \pi_{\{\text{FROM,TO}'\}}(r)$. What meaning can be assigned to the tuples in s? Find an operation that will remove tuples such as ⟨JFK JFK⟩ from s.

3.7 In Example 3.3, let the domain of FLIGHT (and FLIGHT2) be $<$-comparable. Use selection as extended in Section 3.5.1 to give an expression that denotes s without the redundant information. That is, each pair should occur once, and pairs such as $\langle 12\ 12\rangle$ should be removed.

3.8 Compute

$$\sigma_{(\text{DEPARTS} \geq 11:00\text{a} \wedge \text{ARRIVES} \leq 2:00\text{p}) \vee (\text{DEPARTS} \leq 5:00\text{p})}(times)$$

for the relation *times* in Example 3.6.

3.9 Give a ternary relation (in the mathematical sense) that cannot be represented as the conjunction of binary relations without introducing new domains.

3.10 Show that the effect of any selection operation can be achieved using the select operator in the form $\sigma_{A\theta a}$ or $\sigma_{A\theta B}$ and the operators from Chapter 2 except for select. Do not assume the set of comparators Θ is closed under negation.

3.11 Compute

$$\sigma_{\text{ARRIVES}' \leq 2:00\text{p}}(times[\text{ARRIVES} < \text{DEPARTS}']times')$$

where *times* is the relation in Example 3.6 and *times'* is the same relation with all attributes renamed to primed versions. Assume time of day runs from 12:01a to midnight.

3.12 Compute

$$sched[\text{TO} = \text{FROM}',\ \text{ARRIVES} <\cdot\ \text{DEPARTS}']sched'$$

where *sched* is the relation in Table 2.1, *sched'* is *sched* with all attributes renamed to primed versions, and $<\cdot$ is the comparator "earlier by up to 3 hours" that wraps around midnight.

3.13 Show that any theta-join can be expressed using natural join and generalized selection.

3.14 Given relations $r(ABC)$ and $s(BCD)$, what is $sch(E)$ for

$$E = \pi_A(\sigma_{B=b}(r)) \bowtie \pi_B(\pi_{BC}(r) - \pi_{BC}(s)).$$

3.15 Let $\mathfrak{R}$ be the relational algebra

$(\mathbf{U}, \mathfrak{D}, dom, \boldsymbol{R}, d, \Theta, O)$.

(a) Show that if Θ contains arbitrary comparators, then for no proper subset of the operations in Theorem 3.1 is the theorem true.
(b) Show that if Θ contains only equality and inequality comparators, then selection can be restricted to the form $\sigma_{A=B}$.

3.16 Show that if $\text{SPLIT}_\beta(r) = (s, s')$, then $r = s \cup s'$.

3.17 Let r and r' be relations on R. Let $s = r \cup r'$. Show that there does not necessarily exist a predicate β such that $\text{SPLIT}_\beta(s) = (r, r')$.

3.18 Let r be a relation on scheme R, let $\{B_1, B_2, \ldots, B_m\}$ be a subset of R, and let L be an attribute not in R. Let $p_i = |dom(B_i)|$, $1 \leq i \leq m$, and assume all the p_i's are finite. Suppose every value in a tuple of r requires one byte of storage and there are k tuples in r. Give an inequality involving m, k, and $p_1, p_2, \ldots, p_m$ that will indicate when the relations generated by $\text{FACTOR}(r; B_1, B_2, \ldots, B_m; L)$ will require less space than r.

3.10 BIBLIOGRAPHY AND COMMENTS

Codd [1972b] defines the relational algebra as given here, with the exception of renaming. Hall, Hitchcock, and Todd [1975] explore some generalizations of the algebraic operators. Beck [1978] discusses minimal sets of operators. The split operation is from Fagin [1980b].

Exercise 3.3b was suggested by Jon Shultis.