**Relational Algebra Operations from Set Theory**

The UNION, INTERSECTION, and MINUS Operations

We can define the three operations UNION, INTERSECTION, and SET DIFFERENCE on two union-compatible relations R and S as follows:

■ UNION: The result of this operation, denoted by R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

■ INTERSECTION: The result of this operation, denoted by R ∩ S, is a relation that includes all tuples that are in both R and S.

■ SET DIFFERENCE (or MINUS): The result of this operation, denoted by R – S, is a relation that includes all tuples that are in R but not in S.

For example, to retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the UNION operation as follows:

DEP5\_EMPS ← σDno=5(EMPLOYEE)

RESULT1 ← πSsn(DEP5\_EMPS)

RESULT2(Ssn) ← πSuper\_ssn(DEP5\_EMPS)

RESULT ← RESULT1 ∪ RESULT2

The relation RESULT1 has the Ssn of all employees who work in department 5, whereas RESULT2 has the Ssn of all employees who directly supervise an employee who works in department 5.

The UNION operation produces the tuples that are in either RESULT1 or RESULT2 or both (see Figure 8.3) while eliminating any duplicates. Thus, the Ssn value ‘333445555’ appears only once in the result.

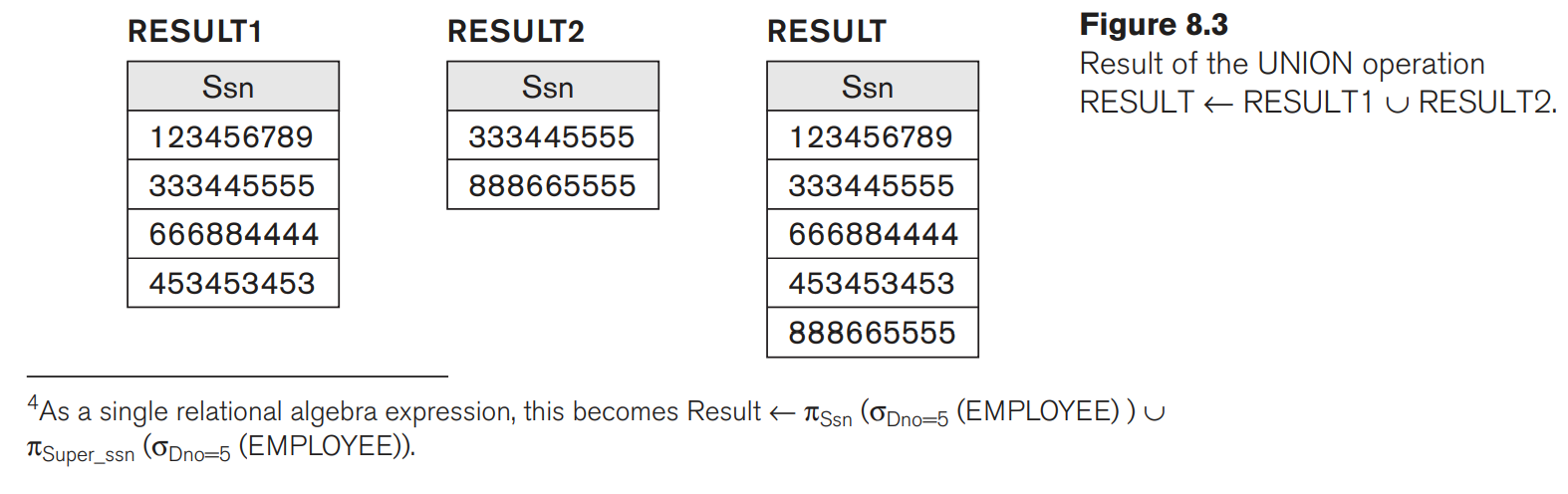


Figure 8.4 illustrates the three operations. The relations STUDENT and INSTRUCTOR in Figure 8.4(a) are union compatible and their tuples represent the names of students and the names of instructors, respectively.

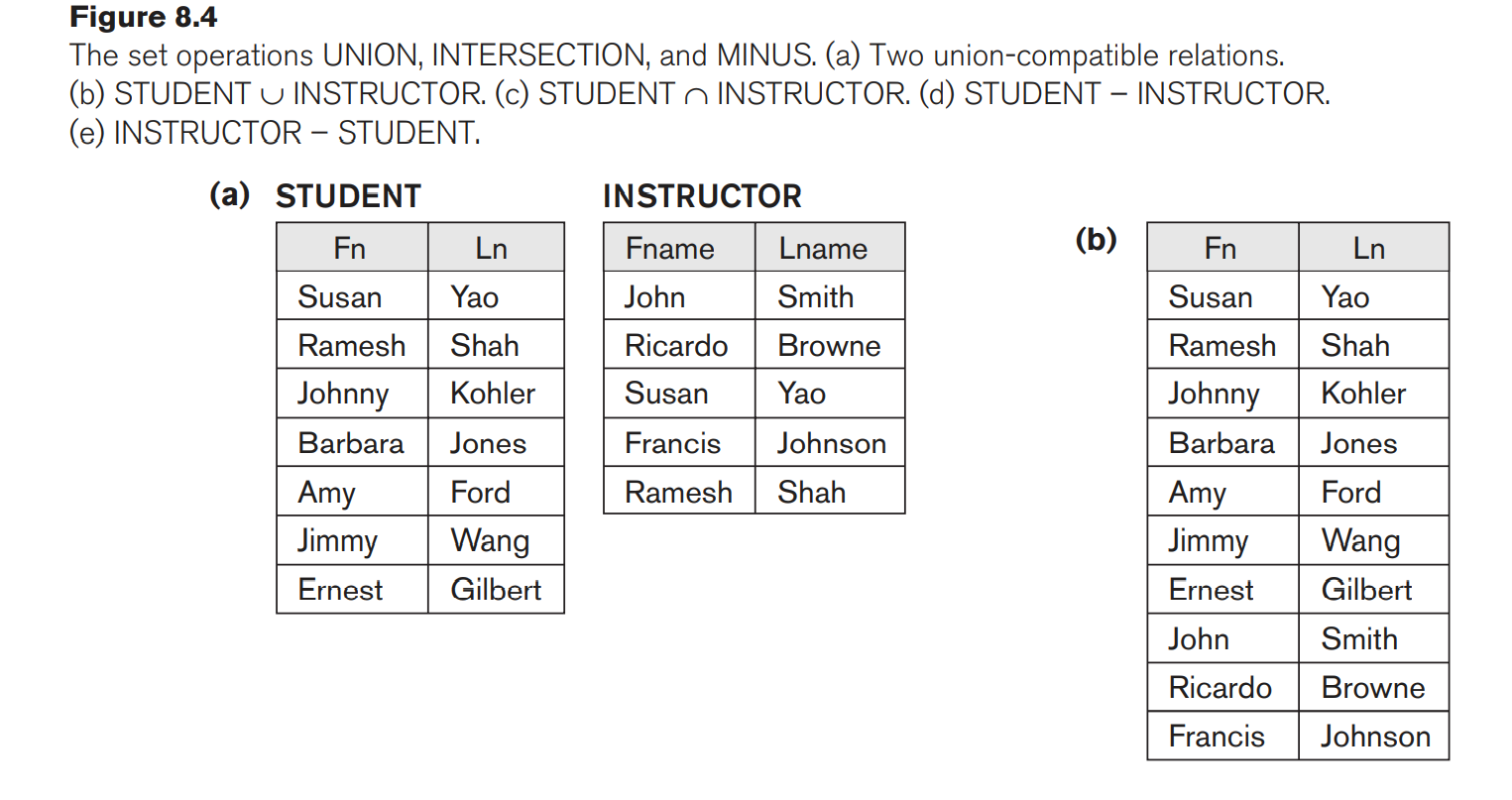
The result of the UNION operation in Figure 8.4(b) shows the names of all students and instructors. Note that duplicate tuples appear only once in the result. The result of the INTERSECTION operation (Figure 8.4(c)) includes only those who are both students and instructors.

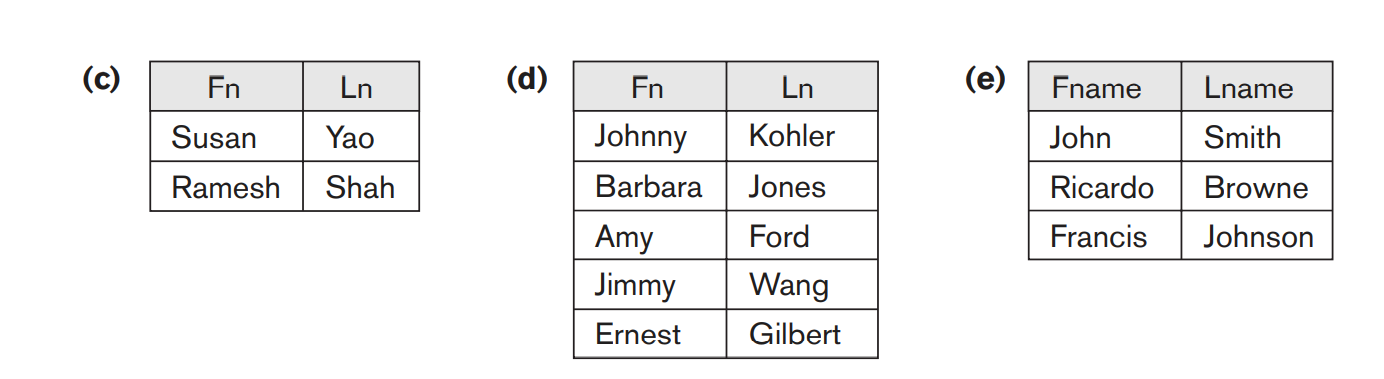
Notice that both UNION and INTERSECTION are commutative operations; that is,

R ∪ S = S ∪ R and R ∩ S = S ∩ R

Both UNION and INTERSECTION can be treated as n-ary operations applicable to any number of relations because both are also associative operations; that is,

R ∪ (S ∪ T ) = (R ∪ S) ∪ T and (R ∩ S) ∩ T = R ∩ (S ∩ T)





The MINUS operation is not commutative; that is, in general,

R − S ≠ S − R

Figure 8.4(d) shows the names of students who are not instructors, and Figure 8.4(e) shows the names of instructors who are not students.

**The CARTESIAN PRODUCT (CROSS PRODUCT) Operation**

CARTESIAN PRODUCT operation—also known as CROSS PRODUCT or CROSS JOIN—which is denoted by ×. This is also a binary set operation.

This set operation produces a new element by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set).

In general, the result of R(A1, A2, … , An) × S(B1, B2, … , Bm) is a relation Q with degree n + m attributes Q(A1, A2, … , An, B1, B2, … , Bm), in that order.

The resulting relation Q has one tuple for each combination of tuples—one from R and one from S. Hence, if R has nR tuples (denoted as |R| = nR), and S has nS tuples, then R × S will have nR \* nS tuples.

The CARTESIAN PRODUCT operation applied by itself is generally meaningless. It is mostly useful when followed by a selection that matches values of attributes coming from the component relations.

For example, suppose that we want to retrieve a list of names of each female employee’s dependents. We can do this as follows:

FEMALE\_EMPS ← σSex=‘F’(EMPLOYEE)

EMPNAMES ← πFname, Lname, Ssn(FEMALE\_EMPS)

EMP\_DEPENDENTS ← EMPNAMES × DEPENDENT

ACTUAL\_DEPENDENTS ← σSsn=Essn(EMP\_DEPENDENTS)

RESULT ← πFname, Lname, Dependent\_name(ACTUAL\_DEPENDENTS)The resulting relations from this sequence of operations are shown in Figure 8.5.

The EMP\_DEPENDENTS relation is the result of applying the CARTESIAN PRODUCT operation to EMPNAMES from Figure 8.5 with DEPENDENT from Figure 5.6.

In EMP\_DEPENDENTS, every tuple from EMPNAMES is combined with every tuple from DEPENDENT, giving a result that is not very meaningful (every dependent is combined with every female employee).

We want to combine a female employee tuple only with her particular dependents—namely, the DEPENDENT tuples whose Essn value match the Ssn value of the EMPLOYEE tuple. The ACTUAL\_DEPENDENTS relation accomplishes this.

