

```

Q4A:  SELECT  DISTINCT Pnumber
      FROM    PROJECT
      WHERE   Pnumber IN
            ( SELECT  Pnumber
              FROM    PROJECT, DEPARTMENT, EMPLOYEE
              WHERE   Dnum = Dnumber AND
                    Mgr_ssn = Ssn AND Lname = 'Smith' )

      OR

      Pnumber IN
            ( SELECT  Pno
              FROM    WORKS_ON, EMPLOYEE
              WHERE   Essn = Ssn AND Lname = 'Smith' );
    
```

If a nested query returns a single attribute *and* a single tuple, the query result will be a single (scalar) value. In such cases, it is permissible to use = instead of IN for the comparison operator. In general, the nested query will return a table (relation), which is a set or multiset of tuples.

SQL allows the use of tuples of values in comparisons by placing them within parentheses. To illustrate this, consider the following query:

```

SELECT  DISTINCT Essn
FROM    WORKS_ON
WHERE   (Pno, Hours) IN
      ( SELECT  Pno, Hours
        FROM    WORKS_ON
        WHERE   Essn = '123456789' );
    
```

This query will select the Essns of all employees who work the same (project, hours) combination on some project that employee 'John Smith' (whose Ssn = '123456789') works on. In this example, the IN operator compares the subtuple of values in parentheses (Pno, Hours) within each tuple in WORKS\_ON with the set of type-compatible tuples produced by the nested query.

In addition to the IN operator, a number of other comparison operators can be used to compare a single value  $v$  (typically an attribute name) to a set or multiset  $V$  (typically a nested query). The = ANY (or = SOME) operator returns TRUE if the value  $v$  is equal to *some value* in the set  $V$  and is hence equivalent to IN. The two keywords ANY and SOME have the same effect. Other operators that can be combined with ANY (or SOME) include >, >=, <, <=, and <>. The keyword ALL can also be combined with each of these operators. For example, the comparison condition ( $v > ALL V$ ) returns TRUE if the value  $v$  is greater than *all* the values in the set (or multiset)  $V$ . An example is the following query, which returns the names of employees whose salary is greater than the salary of all the employees in department 5:

```

SELECT  Lname, Fname
FROM    EMPLOYEE
WHERE   Salary > ALL
      ( SELECT  Salary
        FROM    EMPLOYEE
        WHERE   Dno = 5 );
    
```

In general, a query written with nested select-from-where blocks and using the = or IN comparison operators can *always* be expressed as a single block query. For example, Q16 may be written as in Q16A:

```

Q16A:   SELECT   E.Fname, E.Lname
           FROM     EMPLOYEE AS E, DEPENDENT AS D
           WHERE    E.Ssn = D.Essn AND E.Sex = D.Sex
                AND E.Fname = D.Dependent_name;
    
```

### 7.1.4 The EXISTS and UNIQUE Functions in SQL

EXISTS and UNIQUE are Boolean functions that return TRUE or FALSE; hence, they can be used in a WHERE clause condition. The EXISTS function in SQL is used to check whether the result of a nested query is *empty* (contains no tuples) or not. The result of EXISTS is a Boolean value **TRUE** if the nested query result contains at least one tuple, or **FALSE** if the nested query result contains no tuples. We illustrate the use of EXISTS—and NOT EXISTS—with some examples. First, we formulate Query 16 in an alternative form that uses EXISTS as in Q16B:

```

Q16B:   SELECT   E.Fname, E.Lname
           FROM     EMPLOYEE AS E
           WHERE    EXISTS ( SELECT *
                            FROM   DEPENDENT AS D
                            WHERE  E.Ssn = D.Essn AND E.Sex = D.Sex
                            AND E.Fname = D.Dependent_name);
    
```

EXISTS and NOT EXISTS are typically used in conjunction with a *correlated* nested query. In Q16B, the nested query references the Ssn, Fname, and Sex attributes of the EMPLOYEE relation from the outer query. We can think of Q16B as follows: For each EMPLOYEE tuple, evaluate the nested query, which retrieves all DEPENDENT tuples with the same Essn, Sex, and Dependent\_name as the EMPLOYEE tuple; if at least one tuple EXISTS in the result of the nested query, then select that EMPLOYEE tuple. EXISTS(Q) returns **TRUE** if there is *at least one tuple* in the result of the nested query Q, and returns **FALSE** otherwise. On the other hand, NOT EXISTS(Q) returns **TRUE** if there are *no tuples* in the result of nested query Q, and returns **FALSE** otherwise. Next, we illustrate the use of NOT EXISTS.

**Query 6.** Retrieve the names of employees who have no dependents.

```

Q6:     SELECT   Fname, Lname
           FROM     EMPLOYEE
           WHERE    NOT EXISTS ( SELECT *
                                FROM   DEPENDENT
                                WHERE  Ssn = Essn );
    
```

In Q6, the correlated nested query retrieves all DEPENDENT tuples related to a particular EMPLOYEE tuple. If *none exist*, the EMPLOYEE tuple is selected because the **WHERE**-clause condition will evaluate to **TRUE** in this case. We can explain Q6 as follows: For *each* EMPLOYEE tuple, the correlated nested query selects all

The second option is shown as Q3B. Notice that we need two-level nesting in Q3B and that this formulation is quite a bit more complex than Q3A.

```

Q3B:  SELECT  Lname, Fname
        FROM    EMPLOYEE
        WHERE   NOT EXISTS ( SELECT  *
                               FROM    WORKS_ON B
                               WHERE   ( B.Pno IN ( SELECT  Pnumber
                                                       FROM    PROJECT
                                                       WHERE   Dnum = 5 )
                               AND
                               NOT EXISTS ( SELECT  *
                                             FROM    WORKS_ON C
                                             WHERE   C.Essn = Ssn
                                             AND     C.Pno = B.Pno )));

```

In Q3B, the outer nested query selects any WORKS\_ON (B) tuples whose Pno is of a project controlled by department 5, *if* there is not a WORKS\_ON (C) tuple with the same Pno and the same Ssn as that of the EMPLOYEE tuple under consideration in the outer query. If no such tuple exists, we select the EMPLOYEE tuple. The form of Q3B matches the following rephrasing of Query 3: Select each employee such that there does not exist a project controlled by department 5 that the employee does not work on. It corresponds to the way we will write this query in tuple relation calculus (see Section 8.6.7).

There is another SQL function, UNIQUE(Q), which returns TRUE if there are no duplicate tuples in the result of query Q; otherwise, it returns FALSE. This can be used to test whether the result of a nested query is a set (no duplicates) or a multiset (duplicates exist).

## 7.1.5 Explicit Sets and Renaming in SQL

We have seen several queries with a nested query in the WHERE clause. It is also possible to use an **explicit set of values** in the WHERE clause, rather than a nested query. Such a set is enclosed in parentheses in SQL.

**Query 17.** Retrieve the Social Security numbers of all employees who work on project numbers 1, 2, or 3.

```

Q17:  SELECT  DISTINCT Essn
        FROM    WORKS_ON
        WHERE   Pno IN (1, 2, 3);

```

In SQL, it is possible to **rename** any attribute that appears in the result of a query by adding the qualifier AS followed by the desired new name. Hence, the AS construct can be used to alias both attribute and relation names in general, and it can be used in appropriate parts of a query. For example, Q8A shows how query Q8 from Section 4.3.2 can be slightly changed to retrieve the last name of each employee and his or her supervisor while renaming the resulting attribute names