IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB8005

Chapter XIII

Set Comparison Queries in SQL

Mohammad Dadashzadeh
Wichita State University, USA

INTRODUCTION

One of the most important promises of the relational data model has been that it frees the decision maker, the manager, from the necessity of resorting to an intermediary, the programmer, in retrieving information from the organization's database in response to unanticipated needs. That promise is founded on the availability of very high-level relational query languages such as SQL. Unfortunately, the current specification of the SQL standard fails to support users adequately in formulating complex queries involving set comparison that tend to arise in on-line analytical processing (OLAP) situations. As pointed out by Rao et al. (1996): "SQL's syntax is too restricted to express quantified queries. While SQL allows subqueries to form sets, the relationships that can be expressed over sets are limited, and must be written in awkward and complicated ways." This chapter presents a systematic approach for teaching users how to formulate in SQL complex set comparison queries encountered in ad-hoc decision-making scenarios.

BACKGROUND AND MOTIVATION

Consider the following relational database about suppliers and parts. (The primary key of each relation is underlined.)

SUPPLIER(S#, Supplier_Name, Supplier_City)
PART(P#, Part_Name, Part_Color)
SHIPMENT(S#, P#)
SUPPLY(S#, P#)

The relation SHIPMENT records information on what parts are *currently* shipped by each supplier, while the relation SUPPLY indicates what parts can be supplied, *in the future*, by each supplier. An instance of the relations SHIPMENT and SUPPLY is depicted below.

CURRENT *SHIPMENT*		FUTURE *SUPPLY*	
S1	P1	S1	P1
S1	P2	S1	P2
S1	P3	S1	P3
S1	P5	S1	P4
		S1	P5
S2	P1	S2	P1
S2	P2	S2	P2
S2	P3		
S3	P1	S3	P2
S4	P1	S4	P1
S4	P2	S4	P2
S5	P5		
		S6	P5
		S6	P6

Now, consider the following queries:

Q1: Which suppliers are shipping *at least one* red part?
Q2: Which suppliers are shipping *no* red parts?
Q3: Which suppliers are shipping *only* red parts?
Q4: Which suppliers are shipping *every* red part?
Q5: Which suppliers are shipping *exactly* the red parts?
Q6: Which suppliers are shipping *no* part that they will supply in the future?
Q7: Which suppliers will not continue to supply the same parts that they are currently shipping?

Of the queries listed, Q2-Q7 are considered ***set comparison queries*** since their result sets (i.e., the desired supplier numbers) can only be determined by comparing two sets (e.g., the set of part numbers shipped by each supplier against the set of part numbers for red parts). In contrast, the result set for Q1 can be obtained by merely matching (i.e., joining) the part number from a SHIPMENT row with that of a red PART row as shown below:

Q1: Which suppliers are shipping *at least one* red part?

```
SELECT   DISTINCT S#
FROM     SHIPMENT, PART
WHERE    (SHIPMENT.P# = PART.P#) AND (PART_COLOR = 'RED');
```

Despite their innocuous appearances, queries involving set comparison are very difficult to formulate in relational query languages (Blanning, 1993; Celko, 1997;