This is the fourth chapter in a series about retrieval features of SQL. It is a logical continuation of Chapters 4, 5, and 8.

First, we revisit subqueries, beginning with an introduction to the three operators \texttt{ANY}, \texttt{ALL}, and \texttt{EXISTS}. These operators allow you to create a special relationship between main queries and subqueries, as opposed to using the \texttt{IN} operator or standard comparison operators. You will also learn about \textit{correlated subqueries}, which are subqueries where some subquery clauses refer to column expressions from the main query.

In Sections 9.2 and 9.3, we will look at subqueries in query components other than the \texttt{WHERE} clause: the \texttt{SELECT} and the \texttt{FROM} clauses. In Section 9.4 we will discuss the \texttt{WITH} clause, also referred to as \textit{subquery factoring}, which allows you to define one or more subqueries in the beginning of your SQL commands, and then to reference them by name in the remainder of your SQL command.

We continue with \textit{hierarchical queries}. Relational tables are essentially flat structures, but they can represent hierarchical data structures; for example, by using foreign key constraints referring to the primary key of the same table. The \texttt{MGR} column of the \texttt{EMPLOYEES} table is a classic example of such a hierarchical relationship. Oracle SQL supports explicit syntax to simplify retrieval of hierarchical data structures.

The next subject we investigate is \textit{analytical functions}. Within the context of a single row (or tuple variable), you can reference data in other rows and use it for comparisons and calculations.

Finally, this chapter discusses a helpful Oracle SQL feature allowing you to travel back in time: \textit{flashback queries}.

\section{Subqueries Continued}

Chapter 4 discussed various examples of subqueries, using the \texttt{IN} operator or standard logical comparison operators. As a refresher, let’s start with two standard subquery examples.

The subquery in Listing 9-1 shows all 13 registrations we have for build courses; that is, for course category \texttt{"BLD"}.

\textbf{Listing 9-1. Subquery Using the IN Operator}

\begin{verbatim}
select r.attendee, r.course, r.begindate 
from registrations r 
where r.course in (select c.code 
    from courses c 
    where c.category='BLD');
\end{verbatim}
Listing 9-2 shows how you can retrieve all employees who are younger than colleague 7566.

Listing 9-2. Single-Row Subquery Using a Comparison Operator

```
select e.empno, e.ename, e.init, e.bdate
from employees e
where e.bdate > (select x.bdate
    from employees x
    where x.empno = 7566);
```

Listing 9-2 shows an example of a single-row subquery. The subquery must return a single row, because the comparison operator (> in the third line would fail otherwise. If subqueries of this type nevertheless return more than a single row, you get an Oracle error message, as you discovered in Chapter 4 (see Listing 4-38).

This section continues the discussion of subqueries by explaining the possibilities of the ANY, ALL, and EXISTS operators. You’ll also learn about correlated subqueries.

The ANY and ALL Operators

SQL allows you to combine standard comparison operators (<, >, =, and so on) with subqueries returning any number of rows. You can do that by specifying ANY or ALL between the comparison operator and the subquery. Listing 9-3 shows an example of using the ANY operator, showing all employees with a monthly salary that is higher than at least one manager.