**Chapter 8**

**Advanced SQL**

|  |
| --- |
| **NOTE**  Several points are worth emphasizing:   * Chapter 8 focuses on creating database structures and manipulating the data in tables. The material covers creating databases and objects within the databases, such as tables, indexes, and views. * We have provided the SQL scripts for this chapter. These scripts are intended to facilitate the flow of the material presented to the class. However, given the comments made by our students, the scripts should **not** replace the manual typing of the SQL commands by students. Some students learn SQL better when they have a chance to type their own commands and get the feedback provided by their errors. We recommend that the students use their lab time to practice the commands manually. * In this chapter, the stored procedures and triggers are executed in the Oracle RDBMS. Unlike SQL, which is standardized, languages for creating stored procedures and triggers are not standardized across different DBMS products. For example, while PL/SQL in Oracle and TSQL (Transact SQL) in SQL Server perform similar tasks in roughly similar ways, the syntax and keywords for these languages are very different. This material is presented to help students understand the nature of the tasks performed by these program modules. The programs shown in the text are to illustrate these concepts. The concepts are common across most DBMS products, even though the actual syntax for the languages is different. Even if instructors do not use Oracle or do not teach the syntax of PL/SQL that is presented in the chapter, students still benefit from understanding the need for programs such as the ones presented and the nature of how these programs are implemented. |

**Answers to Review Questions**

1. **What type of integrity is enforced when a primary key is declared?**

Creating a primary key constraint enforces **entity integrity** (i.e. no part of the primary key can contain af null and the primary key values must be unique).

1. **Explain why it might be more appropriate to declare an attribute that contains only digits as a character data type instead of a numeric data type.**

An attribute that contains only digits may be properly defined as character data when the values are nominal; that is, the values do not have numerical significance but serve only as labels such as ZIP codes and telephone numbers. One easy test is to consider whether or not a leading zero should be retained. For the ZIP code 03133, the leading zero should be retained; therefore, it is appropriate to define it as character data. For the quantity on hand of 120, we would not expect to retain a leading zero such as 0120; therefore, it is appropriate to define the quantity on hand as a numeric data type.

1. **What is the difference between a column constraint and a table constraint?**

A column constraint can refer to only the attribute with which it is specified. A table constraint can refer to any attributes in the table.

1. **What are “referential constraint actions”?**

Referential constraint actions, such as ON DELETE CASCADE, are default actions that the DBMS should take when a DML command would result in a referential integrity constraint violation. Without referential constraint actions, DML commands that would result in a violation of referential integrity will fail with an error indicating that the referential integrity constrain cannot be violated. Referential constraint actions can allow the DML command to successfully complete while making the designated changes to the related records to maintain referential integrity.

1. **What is the purpose of a CHECK constraint?**

A CHECK constraint is used to limit the values that can appear in an attribute. It performs the function of enforcing a domain.

1. **Explain when an ALTER TABLE command might be needed.**

ALTER TABLE is used to modify the structure of an existing table by adding, removing, or modifying column definitions and, in some cases, constraints. Many database structures have long, useful lives in an organization. It is not uncommon for a database to exist in organizational systems for decades. If the existing database structure needs to be modified to accommodate changes in business requirements or the integration of new systems, the existing structure will be modified with ALTER TABLE commands. This preserves the existing data in the table, as opposed to dropping the table and then re-creating it.

1. **What is the difference between an INSERT command and an UPDATE command?**

The INSERT command is used to add a new row to a table. The UPDATE command changes the values in attributes of an existing row. UPDATE will not increase the number of rows in a table, but INSERT will.

1. **What is the difference between using a subquery with a CREATE TABLE command and using a subquery with an INSERT command?**

Using a subquery with a CREATE TABLE command is a DDL command and will create a new database table. The table will be structured to match the structure of the data returned by the subquery, and the data from the subquery will be placed in the table. Therefore, using a subquery with CREATE TABLE will both create the structure and place data inside that structure.

Using a subquery with an INSERT command is a DML command and will add data to an existing table. This operation requires that the target table where the data should be stored must already exist. The programmer must ensure that the structure of the data being returned by the subquery is appropriate in terms of data types and constraints for the structure of the table where the results are to be stored.

1. **What is the difference between a view and a materialized view?**

A view defines a query to retrieve data, but it does not create another copy of the data. Whenever the view is used, the defined query is executed to retrieve the current data from the base tables. A materialized view also defines a query, but it also stores another copy of the data in the materialized view. When the materialized view is used, the data from the secondary copy is returned. A materialized view must be periodically be refreshed as the data in the base tables changes over time.

1. **What is a sequence? Write its syntax.**

A sequence is a special type of object that generates unique numeric values in ascending or descending order. You can use a sequence to assign values to a primary key field in a table.

A sequence provides functionality *similar* to the Autonumber data type in MS Access. For example, both, sequences and Autonumber data types provide unique ascending or descending values. However, there are some subtle differences between the two:

* In MS Access an Autonumber is a *data type*; in Oracle a sequence is *a completely independent object*, rather than a data type.
* In MS Access, you can only have one Autonumber per table; in Oracle you can have as many sequences as you want and they are not tied to any particular table.
* In MS Access, the Autonumber data type is tied to a field in a table; in Oracle, the sequence-generated value is not tied to any field in any table and can, therefore, be used on any attribute in any table.

The syntax used to create a sequence is:

CREATE SEQUENCE CUS\_NUM\_SEQ START WITH 100 INCREMENT BY 10 NOCACHE;

1. **What is a trigger, and what is its purpose? Give an example.**

A trigger is a block of PL/SQL code that is automatically invoked by the DBMS upon the occurrence of a data manipulation event (INSERT, UPDATE or DELETE.) Triggers are always associated with a table and are invoked before or after a data row is inserted, updated, or deleted. Any table can have one or more triggers.

Triggers provide a method of enforcing business rules such as:

* A customer making a credit purchase must have an active account.
* A student taking a class with a prerequisite must have completed that prerequisite with a B grade.
* To be scheduled for a flight, a pilot must have a valid medical certificate and a valid training completion record.

Triggers are also excellent for enforcing data constraints that cannot be directly enforced by the data model. For example, suppose that you must enforce the following business rule:

If the quantity on hand of a product falls below the minimum quantity, the P\_REORDER attribute must the automatically set to 1.

To enforce this business rule, you can create the following TRG\_PRODUCT\_REORDER trigger:

CREATE OR REPLACE TRIGGER TRG\_PRODUCT\_REORDER

BEFORE INSERT OR UPDATE OF P\_ONHAND, P\_MIN ON PRODUCT

FOR EACH ROW

BEGIN

IF :NEW.P\_ONHAND <= :NEW.P\_MIN THEN

NEW.P\_REORDER := 1;

ELSE

:NEW.P\_REORDER := 0;

END IF;

END;

1. **What is a stored procedure, and why is it particularly useful? Give an example.**

A stored procedure is a named block of PL/SQL and SQL statements. One of the major advantages of stored procedures is that they can be used to encapsulate and represent business transactions. For example, you can create a stored procedure to represent a product sale, a credit update, or the addition of a new customer. You can encapsulate SQL statements within a single stored procedure and execute them as a single transaction.

There are two clear advantages to the use of stored procedures:

1. Stored procedures substantially reduce network traffic and increase performance. Because the stored procedure is stored at the server, there is no transmission of individual SQL statements over the network.
2. Stored procedures help reduce code duplication through code isolation and code sharing (creating unique PL/SQL modules that are called by application programs), thereby minimizing the chance of errors and the cost of application development and maintenance.

For example, the following PRC\_LINE\_ADD stored procedure will add a new invoice line to the LINE table and it will automatically retrieve the correct price from the PRODUCT table.

CREATE OR REPLACE PROCEDURE PRC\_LINE\_ADD

(W\_LN IN NUMBER, W\_P\_CODE IN VARCHAR2, W\_LU NUMBER)

AS

W\_LP NUMBER := 0.00;

BEGIN

-- GET THE PRODUCT PRICE

SELECT P\_PRICE INTO W\_LP

FROM PRODUCT

WHERE P\_CODE = W\_P\_CODE;

-- ADDS THE NEW LINE ROW

INSERT INTO LINE

VALUES(INV\_NUMBER\_SEQ.CURRVAL, W\_LN, W\_P\_CODE, W\_LU, W\_LP);

DBMS\_OUTPUT.PUT\_LINE('Invoice line ' || W\_LN || ' added');

END;

**Problem Solutions**

All of the problems in the Problem section require writing SQL or PL/SQL code. Since there are minor differences in the code based on the DBMS used, solutions for problems are provided in separate files for Oracle, MySQL, and Microsoft SQL Server. Solutions for Microsoft Access are provided in .mdb files for each data model used in the problem section. A very few of the problems do not apply to all DBMS products. For example, MySQL is installed in “autocommit” mode by default, therefore, issuing COMMIT commands are not necessary. On the other hand, Oracle does not use autocommit by default and does require COMMIT commands to make DML command results permanent in the database. Therefore, instructions about issuing commands to make DML changes permanent do not apply to MySQL, but are necessary for Oracle. Also, since only PL/SQL is presented in the text for creating stored procedures and triggers, problems that require creating these types of modules are only provided in PL/SQL for Oracle.

The files are in the “**Teacher**” data files that accompany the book, and are named as follows:

Oracle: **Ch08\_ProblemSolutions\_ORA.txt**

MySQL: **Ch08\_ProblemSolutions\_MySQL.txt**

SQL Server: **Ch08\_ProblemSolutions\_SQL.txt**

MS Access: **Ch08\_AviaCo.mdb**

**Ch08\_ConstructCo.mdb**

**Ch08\_MovieCo.mdb**

**Ch08\_SaleCo.mdb**

**Ch08\_SimpleCo.mdb**