Database Programming

INSERT Statements

Objectives

In this lesson, you will learn to:

- Explain the importance of being able to alter the data in a database
- Construct and execute INSERT statements which insert a single row using a VALUES clause
- Construct and execute INSERT statements that use special values, null values, and date values
- Construct and execute INSERT statements that copy rows from one table to another using a subquery

Purpose

Up to now, you have been learning how to access data in a database. It's time to learn how to make changes to the database.

In business, databases are dynamic. They are constantly in the process of having data inserted, updated, and deleted. Think how many times the school's student database changes from day to day and year to year. Unless changes are made, the database would quickly lose its usefulness.

Purpose (cont.)

In this lesson, you will begin to use data manipulation language (DML) statements to make changes to a database.

Copy Tables Before Inserting

You will be responsible for altering tables in your schema. You will also be responsible for restoring them just as a real Database Administrator would assume that responsibility.

- To keep your schema tables in their original state, you will make a copy of each table before completing the practice activities in this and later lessons.
- In each practice activity, you will use the copy of the table that you create, not the original.
- If you inadvertently alter a table copy, you can use the original table to restore the copy.

Copy Tables Before Inserting (cont.)

- You should name each copied table: copy_tablename.
- The table copies will not inherit the associated primary-to-foreign-key integrity rules (relationship constraints) of the original tables. The column data types, however, are inherited in the copied tables.

Syntax to Create a Copy of a Table

CREATE TABLE copy_tablename
  AS (SELECT * FROM tablename);

For example:

```
CREATE TABLE copy_f_customers
  AS (SELECT * FROM f_customers);
```

Syntax to Create a Copy of a Table (cont.)

To verify and view the copy of the table, use the following DESCRIBE and SELECT statements:

```
DESCRIBE copy_f_customers;

SELECT * FROM copy_f_customers;
```

INSERT

The INSERT statement is used to add new rows to a table. The statement requires three values:

- the name of the table
- the names of the columns in the table to populate
- corresponding values for each column

How can we INSERT the data below to create a new customer in the copy_f_customers table?

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
145	Katie	Hernandez	92 Chico Way	LA	CA	98008	8586667641

INSERT (cont.)

The syntax shown on the next slide uses INSERT to add a new customer to a Global Fast Foods table. This statement explicitly lists each column as it appears in the table. The values for each column are listed in the same order. Note that number values are not enclosed in single quotation marks.

INSERT (cont.)

```
INSERT INTO copy_f_customers
   (id, first_name, last_name, address, city, state, zip, phone_number)
VALUES
  (145, 'Katie', 'Hernandez', '92 Chico Way', 'Los Angeles', 'CA',
98008, 8586667641);
```

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
145	Katie	Hernandez	92 Chico Way	LA	CA	98008	8586667641

INSERT (cont.)

Another way to insert values in a table is to implicitly add them by omitting the column names.

One precaution: the values for each column must match exactly the default order in which they appear in the table (as shown in a DESCRIBE statement), and a value must be provided for each column.

INSERT (cont.)

The INSERT statement in this example was written without explicitly naming the columns. For clarity, however, it is best to use the column names in an INSERT clause.

```
INSERT INTO copy_f_customers
VALUES (475, 'Angelina', 'Wright', '7425 Redwood St', 'San Francisco',
'CA', 94162, '4159982010');
```

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
475	Angelina	Wright	7425 Redwood St	San Francisco	CA	94162	4159982010

Check The Table First

Before inserting data into a table, you must check several table details. The DESCRIBE tablename statement will return a description of the table structure in the table summary chart.

COPY_F_STAFFS TABLE SUMMARY

Primary Key	Name	Type	Length	Precision	Scale	Nullable	Default	Comments
1	ID	Number		5	0			
	FIRST_NAME	Varchar2	25					
	LAST_NAME	Varchar2	35					
	BIRTHDATE	Date	7					
	SALARY	Number		8	2			
	OVERTIME_RATE	Number		5	2	√		
	TRAINING	Varchar2	50			√		

Table Summary

As shown in the example, the table summary provides information about each column in the table, such as:

Primary Key	Name	Type	Length	Precision	Scale	Nullable	Default	Comments
1	ID	Number		5	0			
	FIRST_NAME	Varchar2	25					
	LAST_NAME	Varchar2	35					
	BIRTHDATE	Date	7					
	SALARY	Number		8	2			
	OVERTIME_RATE	Number		5	2	√		
	TRAINING	Varchar2	50			√		

Table Summary (cont.)

- the allowance of duplicate values
- the type of data allowed
- the amount of data allowed
- the allowance of NULL values

Primary Key	Name	Type	Length	Precision	Scale	Nullable	Default	Comments
1	ID	Number		5	0			
	FIRST_NAME	Varchar2	25					
	LAST_NAME	Varchar2	35					
	BIRTHDATE	Date	7					
	SALARY	Number		8	2			
	OVERTIME_RATE	Number		5	2	√		
	TRAINING	Varchar2	50			√		

Table Summary (cont.)

Notice the Length column for characters and dates, and the Precision and Scale columns for numbers. Precision is the total number of digits, and Scale is the number of digits to the right of the decimal place.

Primary Key	Name	Type	Length	Precision	Scale	Nullable	Default	Comments
1	ID	Number		5	0			
	FIRST_NAME	Varchar2	25					
	LAST_NAME	Varchar2	35					
	BIRTHDATE	Date	7					
	SALARY	Number		8	2			
	OVERTIME_RATE	Number		5	2	√		
	TRAINING	Varchar2	50			√		

Table Summary (cont.)

The OVERTIME_RATE column allows numbers with a Precision of 5 and a Scale of 2. The maximum value allowed in this column is 999.99.

Primary Key	Name	Type	Length	Precision	Scale	Nullable	Default	Comments
1	ID	Number		5	0			
	FIRST_NAME	Varchar2	25					
	LAST_NAME	Varchar2	35					
	BIRTHDATE	Date	7					
	SALARY	Number		8	2			
	OVERTIME_RATE	Number		5	2	√		
	TRAINING	Varchar2	50			√		

Inserting Rows With Null Values

The INSERT statement need not specify every column—the Nullable columns may be excluded. If every column that requires a value is assigned a value, the insert works.

Inserting Rows With Null Values (cont.)

In our example, the SALARY column is defined as a NOT NULL column. An implicit attempt to add values to the table as shown would generate an error.

```
INSERT INTO copy_f_staffs
  ( id, first_name, last_name, birthdate, overtime_rate)
VALUES
  (15, 'Gregorz', 'Polanski', '25-SEP-1988', 17.50);
```

ORA-01400: cannot insert NULL into ("USQA_JOHN_SQL01_S01"."COPY_F_STAFFS"."SALARY")

Inserting Rows With Null Values (cont.)

An implicit insert will automatically insert a null value in columns that allow nulls. To explicitly add a null value to a column that allows nulls, use the NULL keyword in the VALUES list.

Inserting Rows With Null Values (cont.)

To specify empty strings and/or missing dates, use empty single quotation marks (with no spaces between them like this ‘’) for the missing data.

```
INSERT INTO copy_f_staffs
  ( id, first_name, last_name, birthdate, salary, overtime_rate)
VALUES
  (15, ‘Gregorz’, ‘Polanski’, ‘25-SEP-1988’, 224.25, null);
```

ID	FIRST_NAME	LAST_NAME	BIRTHDATE	SALARY	OVERTIME_RATE	TRAINING
15	Gregorz	Polanski	25-SEP-1988	224.25	(null)	(null)

Inserting Special Values

Special values such as SYSDATE and USER can be entered in the VALUES list of an INSERT statement.

SYSDATE will put the current date and time in a column.

USER will insert the current session's username, which is OAE_PUBLIC_USER in Oracle Application Express.

Inserting Special Values (cont.)

```
INSERT INTO copy_employees
  (employee_id, last_name, email, hire_date, job_id)
VALUES
  (1001, USER, 'Test', SYSDATE, 'IT_PROG');
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
1001	(null)	HTMLDB_ PUBLIC_USER	Test	(null)	12-MAR-2006	IT_PROG

Inserting Special Values (cont.)

In addition, functions and calculated expressions can also be used in the VALUES clause.

The example below illustrates two types of insertions into the Global Fast Foods copy_f_orders table. SYSDATE is used to insert the order_date, and a calculated expression is used to enter the order_total.

```
INSERT INTO copy_f_orders
   (order_number, order_date, order_total, cust_id, staff_id)
VALUES
   (1889, SYSDATE, 87.92*1.08, 123, 19)
```

Inserting Specific Date Values

The default format model for date is DD-MON-RR. With this format, recall that the century defaults to the nearest century (nearest to SYSDATE) with the default time of midnight (00:00:00).

Inserting Specific Date Values (cont.)

In an earlier section, we learned how to use the TO_CHAR function to convert a date to a character string when we want to retrieve and display a date value in a non-default format. Here is a reminder of TO_CHAR:

```
SELECT first_name, TO_CHAR(birthdate, 'Month fmDD, RRRR')
FROM  f_staffs
WHERE id = 12;
```

FIRST_NAME	TO_CHAR(BIRTHDATE, 'MONTH FMDD, RRRR')
Sue	July 1, 1980

Inserting Specific Date Values (cont.)

Similarly, if we want to INSERT a row with a non-default format for a date column, we must use the TO_DATE function to convert the date value (a character string) to a date.

```
INSERT INTO f_staffs
    (first_name, TO_DATE(birthdate, 'Month fmDD, RRRR')
VALUES
    (‘Sue’, ‘July 1, 1980’);
```

Inserting Specific Date Values (cont.)

A second example of TO_DATE allows the insertion of a specific time of day, overriding the default time of midnight.

```
INSERT INTO f_staffs
    (first_name, TO_DATE(birthdate, 'Month fmDD, RRRR HH24:MI')
VALUES
    (‘Sue’, ‘July 1, 1980 17:20’);
```

Using A Subquery To Copy Rows

Each INSERT statement we have seen so far adds only one row to the table. But suppose we want to copy 100 rows from one table to another.

We do not want to have to write and execute 100 separate INSERT statements, one after the other. That would be very time-consuming.

Using A Subquery To Copy Rows (cont.)

Fortunately, SQL allows us to use a subquery within an INSERT statement. All the results from the subquery are inserted into the table. So we can copy 100 rows – or 1000 rows – with one multiple-row subquery within the INSERT.

As you would expect, you don't need a VALUES clause when using a subquery to copy rows because the inserted values will be exactly the values returned by the subquery.

Using A Subquery To Copy Rows (cont.)

In the example shown, a new table called SALES_REPS is being populated with copies of some of the rows and columns from the EMPLOYEES table.

The WHERE clause is selecting those employees that have job IDs like '%REP%'.

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
    SELECT employee_id, last_name, salary, commission_pct
    FROM   employees
    WHERE  job_id LIKE '%REP%';
```

Using A Subquery To Copy Rows (cont.)

The number of columns and their data types in the column list of the INSERT clause must match the number of columns and their data types in the subquery.

The subquery is not enclosed in parentheses as is done with the subqueries in the WHERE clause of a SELECT statement.

Using A Subquery To Copy Rows (cont.)

If we want to copy all the data – all rows and all columns – the syntax is even simpler.

To select all rows from the EMPLOYEES table and insert them into the SALES_REPS table, the statement would be written as shown:

```
INSERT INTO sales_reps
   SELECT *
   FROM   employees;
```

Using A Subquery To Copy Rows (cont.)

Again, this will work only if both tables have the same number of columns with matching data types, and they are in the same order.

Terminology

Key terms used in this lesson included:

- INSERT INTO
- USER
- Transaction
- Explicit

Summary

In this lesson, you should have learned how to:

- Explain the importance of being able to alter the data in a database
- Construct and execute INSERT statements which insert a single row using a VALUES clause
- Construct and execute INSERT statements that use special values, null values, and date values
- Construct and execute INSERT statements that copy rows from one table to another using a subquery