Database Programming with PL/SQL

3-4
Using Transaction Control Statements
Objectives

This lesson covers the following objectives:

• Define a transaction and provide an example

• Construct and execute a transaction control statement in PL/SQL

• Since Oracle Application Express automatically commits changes, the following information will be presented as if you were issuing the commands in an installed/local environment with the ability to use COMMIT and ROLLBACK.
Purpose

• In this lesson, you learn how to include transaction control statements such as COMMIT, ROLLBACK, and SAVEPOINT in PL/SQL.

• Transactions often have multiple parts or steps.

• If something happens to prevent, say, the third step from completing, database integrity is maintained because steps one and two didn't actually change the database because a COMMIT statement was never issued.

• ROLLBACK and SAVEPOINT statements give you additional control in processing transactions.
Database Transaction

• A transaction is an inseparable list of database operations that must be executed either in its entirety or not at all.

• Transactions maintain data integrity and guarantee that the database is always in a consistent state.
Example of a Transaction

- To illustrate the concept of a transaction, consider a banking database.
- When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operations:
  - Decrease savings account balance.
  - Increase checking account balance.
  - Record the transaction in the transaction journal.
Example of a Transaction

• What would happen if there were insufficient funds in the savings account?
• Would the funds still be added to the checking account?
• Would an entry be logged in the transaction journal?
• What do you think should happen?

| Decrease savings account balance. | UPDATE savings_accounts SET balance = balance - 500 WHERE account = 3209; |
| Increase checking account balance. | UPDATE checking_accounts SET balance = balance + 500 WHERE account = 3208; |
| Record the transaction in the transaction journal. | INSERT INTO journal VALUES (journal_seq.NEXTVAL, '1B' 3209, 3208, 500); |
Example of a Transaction

If all three SQL statements can be performed to maintain the accounts in proper balance, the effects of the transaction can be committed, or applied to the database tables.

1. Decrease savings account.
2. Increase checking account.
3. Record the transaction in the transaction journal.

If OK, then COMMIT!
Example of a Transaction

However, if a problem, such as insufficient funds, invalid account number, or a hardware failure prevents one or two of the statements in the transaction from completing, the entire transaction must be rolled back (reversed out) so that the balance of all accounts is correct.

- Decrease savings account.
- Increase checking account.
- Record the transaction in the transaction journal.

If OK, then

If not OK then

ROLLBACK!
Transaction Control Statements

• You use transaction control statements to make the changes to the database permanent or to discard them.

• The three main transaction control statements are:
  – COMMIT
  – ROLLBACK
  – SAVEPOINT

• The transaction control commands are valid in PL/SQL and therefore can be used directly in the executable or exception section of a PL/SQL block.
COMMIT

- **COMMIT** is used to make the database changes permanent.
- If a transaction ends with a **COMMIT** statement, all the changes made to the database during that transaction are made permanent.

```plsql
BEGIN
    INSERT INTO pairtable VALUES (1, 2);
    COMMIT;
END;
```

- The keyword **END** signals the end of a PL/SQL block, not the end of a transaction.
- A transaction may include multiple PL/SQL blocks and nested PL/SQL blocks.
ROLLBACK

• **ROLLBACK** is for discarding any changes that were made to the database after the last **COMMIT**.

• If the transaction fails, or ends with a **ROLLBACK**, then none of the statements take effect.

• In the example, only the second **INSERT** statement adds a row of data.

```sql
BEGIN
    INSERT INTO pairtable VALUES (3, 4);
    ROLLBACK;
    INSERT INTO pairtable VALUES (5, 6);
    COMMIT;
END;
```
SAVEPOINT

• **SAVEPOINT** is used to mark an intermediate point in transaction processing.

• **ROLLBACK** is used to return the data values to the point of a **SAVEPOINT**.

```sql
BEGIN
    INSERT INTO pairtable VALUES (7, 8);
    SAVEPOINT my_sp_1;
    INSERT INTO pairtable VALUES (9, 10);
    SAVEPOINT my_sp_2;
    INSERT INTO pairtable VALUES (11, 12);
    ROLLBACK to my_sp_1;
    INSERT INTO pairtable VALUES (13, 14);
    COMMIT;
END;
```
Terminology

Key terms used in this lesson included:

• COMMIT
• END
• ROLLBACK
• SAVEPOINT
• Transaction
Summary

In this lesson, you should have learned how to:
• Define a transaction and provide an example
• Construct and execute a transaction control statement in PL/SQL