Database Programming

Database Transactions

Objectives

In this lesson, you will learn to:

- Define the terms COMMIT, ROLLBACK, and SAVEPOINT as they relate to data transactions
- List three advantages of the COMMIT, ROLLBACK, and SAVEPOINT statements
- Explain why it is important, from a business perspective, to be able to control the flow of transaction processing

Purpose

What if a bank had no systematic process for recording deposits and withdrawals?

How would you know if a deposit was credited to your account before you needed to withdraw money? You can imagine the confusion it would cause.

Purpose (cont.)

Fortunately, banks control transaction processes to ensure data consistency.

In this lesson you will learn how the process of changing data is managed and how changes to a database are committed or cancelled.

Transactions

Transactions are a fundamental concept of all database systems. Transactions allow users to make changes to data and then decide whether to save or discard the work.

Transactions (cont.)

Database transactions bundle multiple steps into one logical unit of work. A transaction consists of one of the following:

- DML statements which constitute one consistent change to the data. The DML statements include INSERT, UPDATE, DELETE, and MERGE
- One DDL statement such as CREATE, ALTER, DROP, RENAME, or TRUNCATE
- One DCL statement such as GRANT or REVOKE

Transaction Analogy

A bank database contains balances for various customer accounts, as well as total deposit balances for other branches. Suppose a customer wants to withdraw and transfer money from his account and deposit it into another customer's account at a different branch.

Transaction Analogy (cont.)

There are several separate steps involved to accomplish this rather simple operation. Both bank branches want to be assured that either all steps in the transaction happen, or none of them happen, and if the system crashes, the transaction is not left partially complete. Grouping the withdrawal and deposit steps into one transaction provides this guarantee.

A transaction either happens completely or not at all.

Controlling Transactions

Transactions are controlled using the following statements:

COMMIT: Represents the point in time where the user has made all the changes he wants to have logically grouped together, and because no mistakes have been made, the user is ready to save the work. When a COMMIT statement is issued, the current transaction ends making all pending changes permanent.

Controlling Transactions (cont.)

Transactions are controlled using the following statements:

ROLLBACK: Enables the user to discard changes made to the database. When a ROLLBACK statement is issued, all pending changes are discarded.

SAVEPOINT: Creates a marker in a transaction, which divides the transaction into smaller pieces.

Controlling Transactions (cont.)

Transactions are controlled using the following statements:

ROLLBACK TO SAVEPOINT: Allows the user to roll back the current transaction to a specified savepoint. If an error was made, the user can issue a ROLLBACK TO SAVEPOINT statement discarding only those changes made after the SAVEPOINT was established.

Transaction Example

In the example shown, the user has issued an UPDATE statement and immediately created SAVEPOINT one.

```
UPDATE d_cds
SET cd_number = 96
WHERE title = 'Graduation Songbook';
SAVEPOINT one;
```

Transaction Example (cont.)

After an INSERT statement and an UPDATE statement, the user realized that a WHERE clause was not included in the last UPDATE. To remedy the mistake, the user issued a ROLLBACK TO SAVEPOINT one. The data is now restored to its state at SAVEPOINT one.

```
UPDATE d_cds
SET cd_number = 96
WHERE title = 'Graduation Songbook';
SAVEPOINT one;
INSERT INTO d_cds(cd_number, title, producer, year)
VALUES(100, 'Go For It', 'The Music Man', 2004) );
UPDATE d_cds
SET cd_number = 101;
ROLLBACK TO SAVEPOINT one;
COMMIT;
```

When Does a Transaction Start or End?

A transaction **begins** with the first DML (INSERT, UPDATE, DELETE or MERGE) statement.

A transaction **ends** when one of the following occurs:

- A COMMIT or ROLLBACK statement is issued
- A DDL(CREATE, ALTER, DROP, RENAME or TRUNCATE) statement is issued
- A DCL(GRANT or REVOKE) statement is issued
- The user exits iSQL*Plus or SQL*Plus
- A machine fails or the system crashes

When Does a Transaction Start or End? (cont.)

After one transaction ends, the next executable SQL statement automatically starts the next transaction. A DDL statement or a DCL statement is automatically committed and therefore implicitly ends a transaction. Every data change made during a transaction is temporary until the transaction is committed.

Data Consistency

Imagine spending several hours making changes

to employee data only to find out that someone else was entering information that conflicted with your changes!

To prevent such disruptions or conflicts and to allow multiple users to access the database at the same time, database systems employ an automatic implementation called "read consistency.“

Data Consistency (cont.)

Read consistency guarantees a consistent view of the data by all users at all times. Readers do not view data that is in the process of being changed. Writers are ensured that the changes to the database are done in a consistent way. Changes made by one writer do not destroy or conflict with changes another writer is making.

Read Consistency

Read consistency is an automatic implementation.

A partial copy of the database is kept in undo segments. When User A issues an insert, update, or delete operation to the database, the Oracle server takes a snapshot (copy) of the data before it is changed and writes it to an undo (rollback) segment.

User B still sees the database as it existed before the changes started; he views the undo segment's snapshot of the data.

Read Consistency (cont.)

Before changes are committed to the database, only the user who is changing the data sees the changes; everyone else sees the snapshot in the undo segment. This guarantees that readers of the data see consistent data that is not currently undergoing change.

Changes Visible

When a DML statement is committed, the change made to the database becomes visible to anyone executing a SELECT statement.

If the transaction is rolled back, the changes are undone:

- The original, older version of the data in the undo segment is written back to the table.
- All users see the database as it existed before the transaction began.

COMMIT, ROLLBACK, and SAVEPOINT

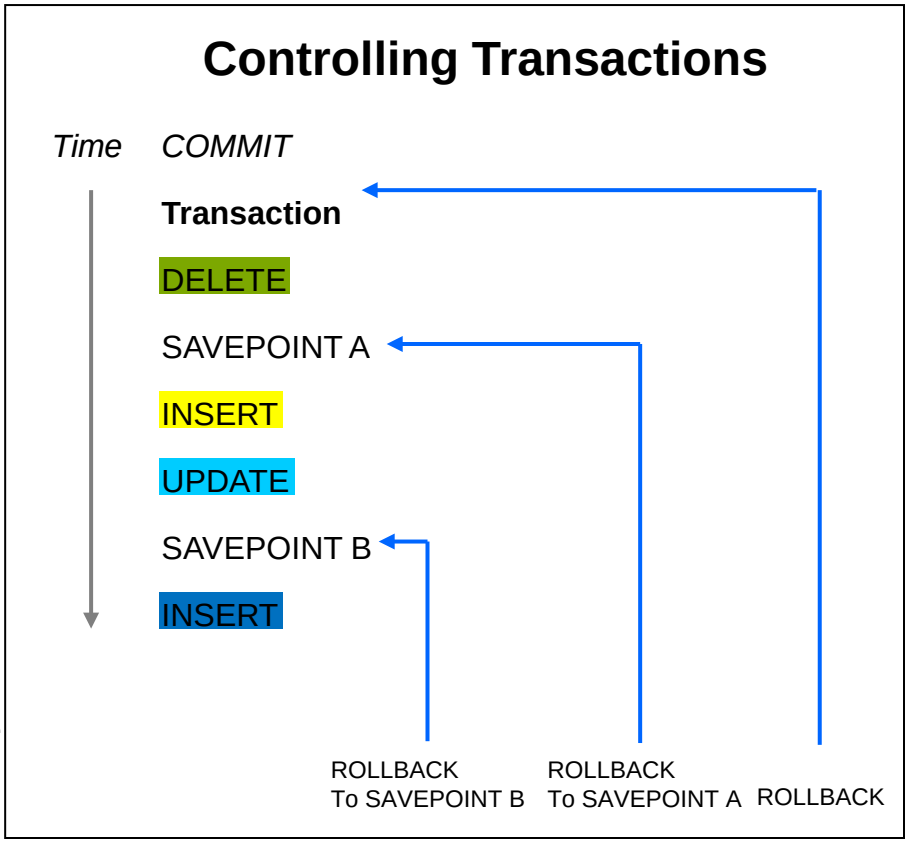
COMMIT and ROLLBACK ensure data consistency, making it possible both to preview data changes before making changes permanent, and to group logically related operations.

SAVEPOINT creates a point in a transaction to which we can rollback without having to undo the entire transaction.

However, SAVEPOINT is not supported in Oracle Application Express, due to the way Oracle Application Express manages connections to the database.

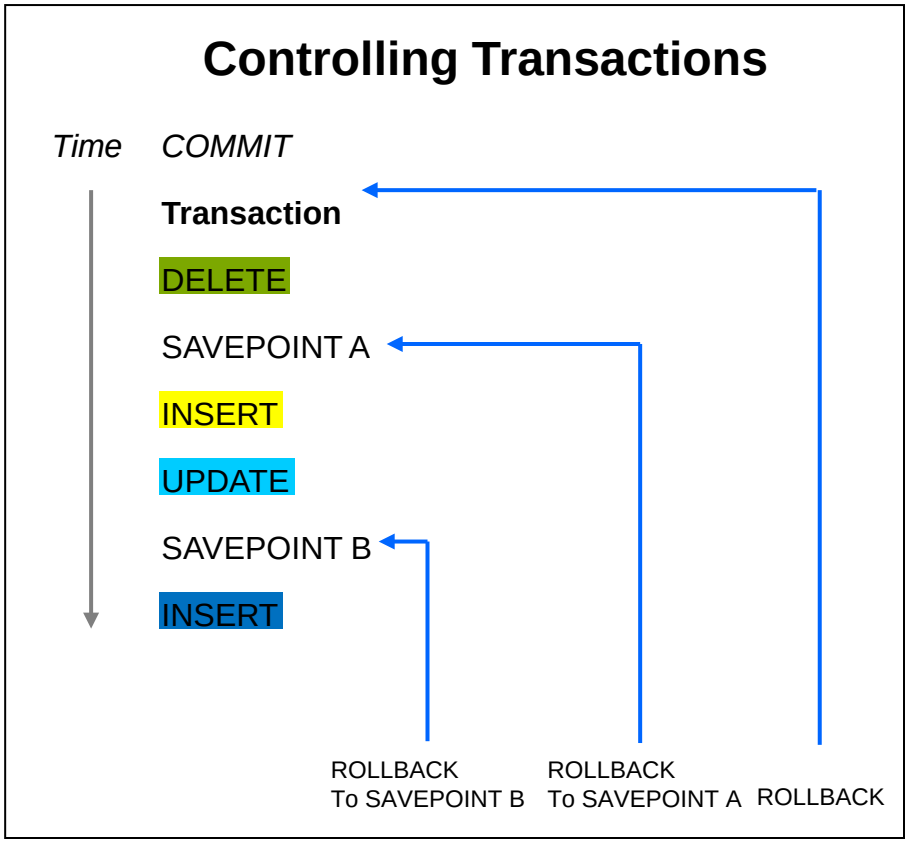
COMMIT, ROLLBACK, and SAVEPOINT (cont.)

In the transaction shown in the graphic, a DELETE statement was issued and then SAVEPOINT A was established. This SAVEPOINT acts like a marker that will allow the user to rollback any subsequent changes made in the data back to the state of the data as it existed at this point.



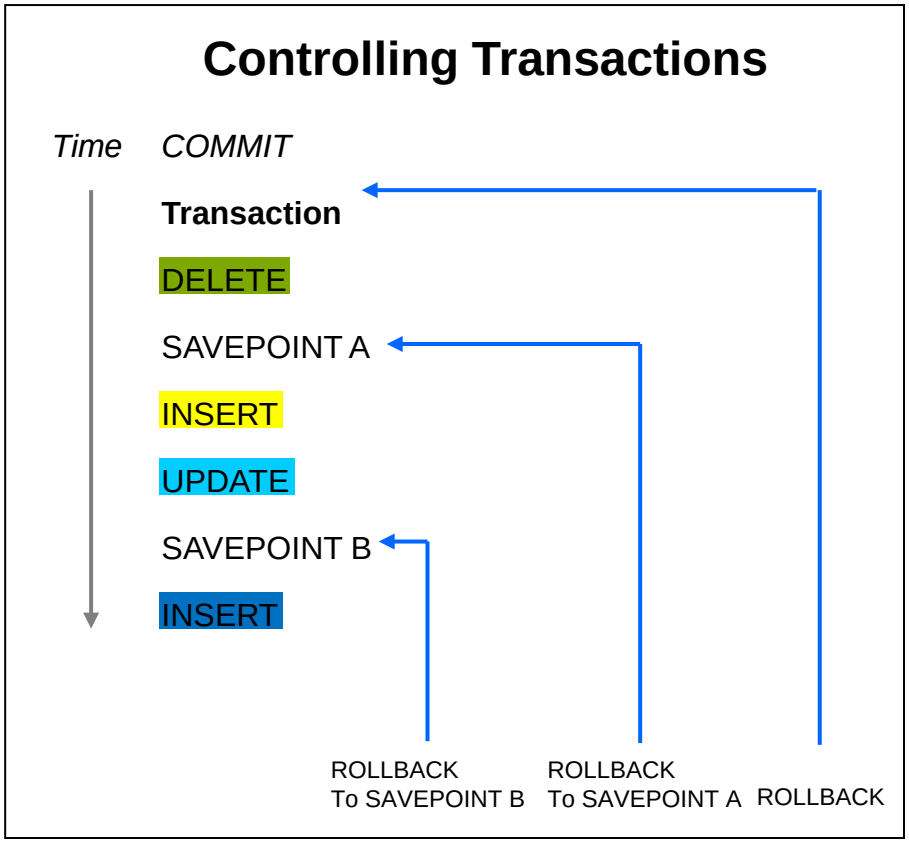
COMMIT, ROLLBACK, and SAVEPOINT (cont.)

In the example, following SAVEPOINT A, the user issues an INSERT and UPDATE statements, then establishes another rollback marker at SAVEPOINT B.



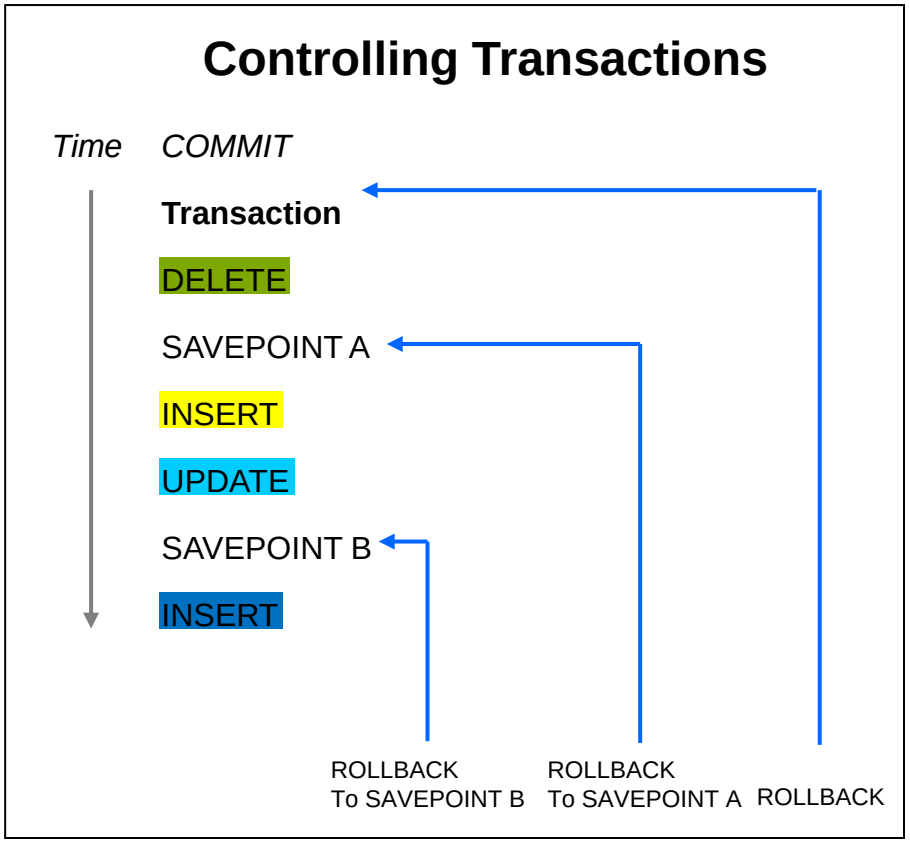
COMMIT, ROLLBACK, and SAVEPOINT (cont.)

If for some reason the user does not want these INSERT and/or UPDATE statements to occur, the user can issue a ROLLBACK TO SAVEPOINT A statement. This will rollback to the state of the data as it was at the SAVEPOINT A marker.



COMMIT, ROLLBACK, and SAVEPOINT (cont.)

Adding other SAVEPOINTS creates additional markers for rollback points. If a user issues a ROLLBACK without a ROLLBACK TO SAVEPOINT statement, the entire transaction is ended and all pending data changes are discarded.



Implicit Transaction Processing

Automatic commit of data changes occurs under the following circumstances:

- a DDL statement is issued
- a DCL statement is issued
- a normal exit from iSQL*Plus or SQL*Plus without explicitly issuing COMMIT or ROLLBACK statements
- When the Autocommit box is checked in Oracle Application Express

Implicit Transaction Processing (cont.)

Automatic rollback occurs under an abnormal termination of either iSQL*Plus or SQL*Plus, or when a system failure occurs.

This prevents any errors in the data from causing unwanted changes to the underlying tables. The integrity of the data is therefore protected.

Locking

It is important to prevent data from being changed by more than one user at a time. Oracle uses locks that prevent destructive interaction between transactions accessing the same resource, either a user object (such as tables or rows) or a system object not visible to users (such as shared data structures and data dictionary rows).

How the Oracle Database Locks Data

Oracle locking is performed automatically and requires no user action. Implicit locking occurs for SQL statements as necessary, depending on the action requested. Implicit locking occurs for all SQL statements except SELECT.

The users can also lock data manually, which is called explicit locking.

When a COMMIT or ROLLBACK statement is issued, locks on the affected rows are released.

Terminology

Key terms used in this lesson included:

- Transaction
- Commit
- Savepoint
- Rollback
- Read consistency
- Locks

Summary

In this lesson you have learned to:

- Define the terms COMMIT, ROLLBACK, and SAVEPOINT as they relate to data transactions
- List three advantages of the COMMIT, ROLLBACK, and SAVEPOINT statements
- Explain why it is important, from a business perspective, to be able to control the flow of transaction processing