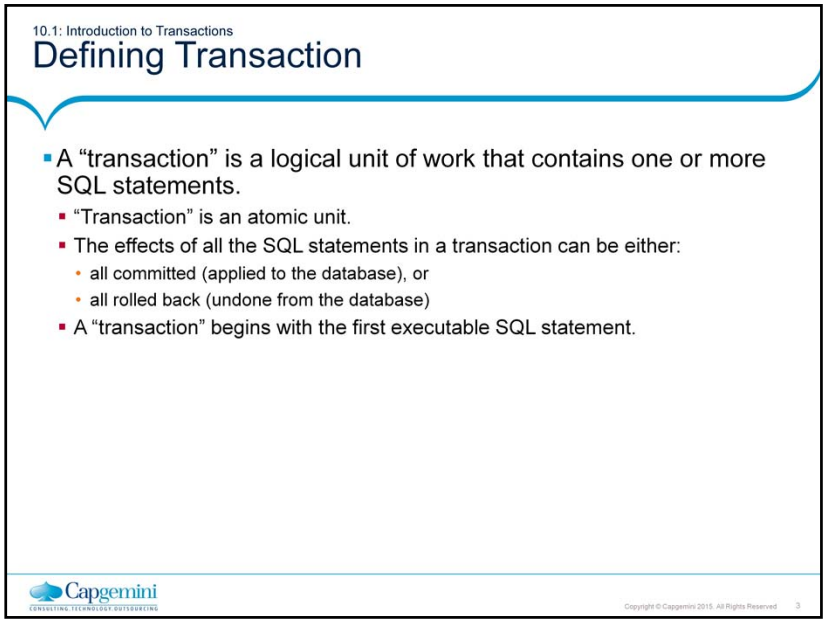
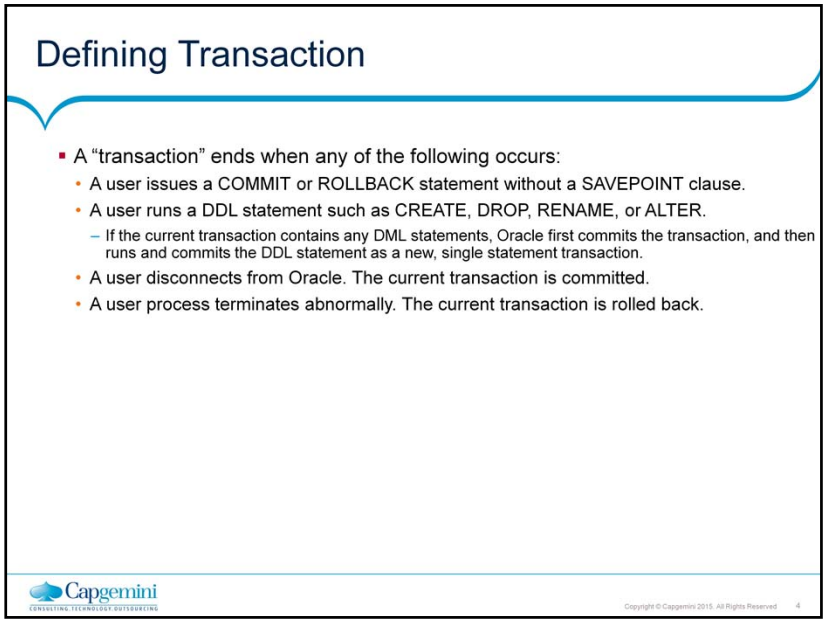
DBMS SQL

Lesson 11: Transaction Control Language

Lesson Objectives

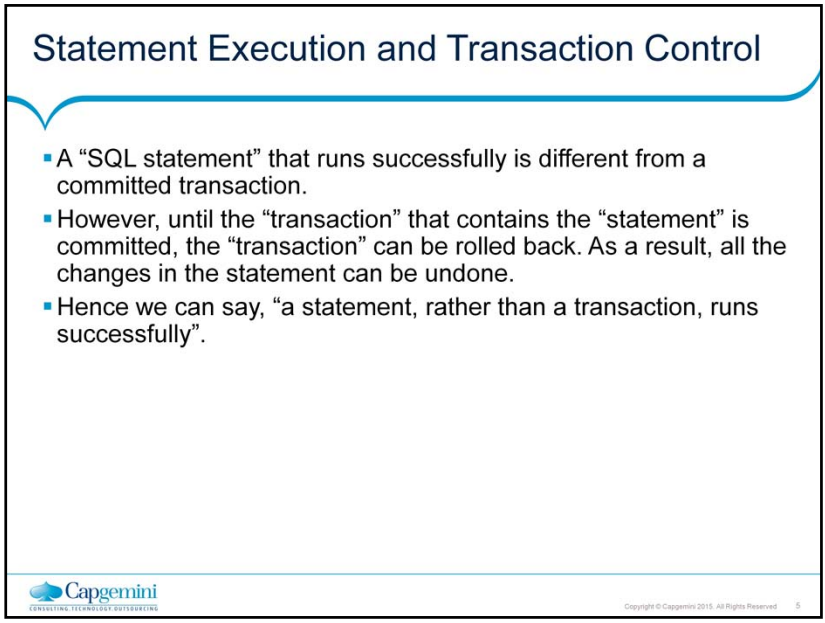
- To understand the following topics:
  - Transactions
    - Statement execution
    - Transaction control
    - Commit Transactions
    - Commit Command
    - Rollback transactions
    - Save points


10.1: Introduction to Transactions
Defining Transaction
A “transaction” is a logical unit of work that contains one or more SQL statements.
“Transaction” is an atomic unit.
The effects of all the SQL statements in a transaction can be either:
all committed (applied to the database), or
all rolled back (undone from the database)
A “transaction” begins with the first executable SQL statement.
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 3



Note:

After one transaction ends, the next executable SQL statement automatically starts the subsequent transaction.

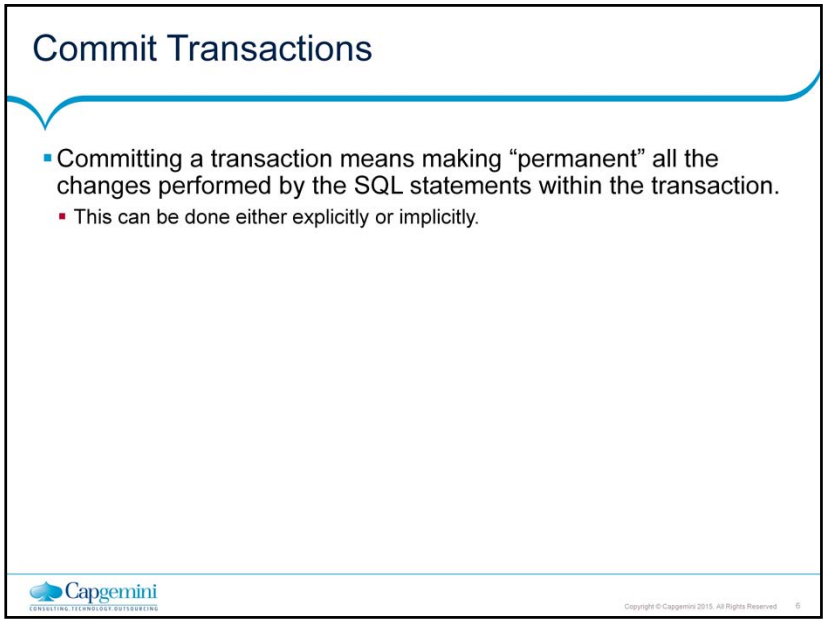


Statement Execution and Transaction Control:

Executing successfully means that a single statement was:

- Parsed
- Found to be a valid SQL construction
- Run without error as an atomic unit.

For example: All rows of a multi-row update are changed.



Note:

An “explicit request” occurs when the user issues a COMMIT statement.

An “implicit request” occurs after normal termination of an application or completion of a data definition language (DDL) operation.

The changes made by the SQL statement(s) of a transaction become permanent and visible to other users only after that transaction is committed. Queries that are issued after the transaction is committed will see the committed changes.

Statement-Level Rollback

If at any time during execution, a SQL statement causes an error, then all effects of the statement are rolled back.

The effect of the rollback is as if that statement had never been run. This operation is a statement-level rollback.

Errors discovered during SQL statement execution cause statement-level rollbacks.

For example: Attempting to insert a duplicate value in a primary key.

Single SQL statements involved in a deadlock (competition for the same data) can also cause a statement-level rollback.

Errors discovered during SQL statement parsing, such as a syntax error, have not yet been run, so they do not cause a statement-level rollback.

A SQL statement that fails causes a loss only of any work it would have performed by itself.

It does not cause the loss of any work that preceded it in the current transaction.

If the statement is a DDL statement, then the implicit commit that immediately preceded it is not undone.

Commit Transactions

- COMMIT statement makes “permanent” all the changes that are performed in the current transaction.
- Syntax:

```
COMMIT [WORK];
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Commit Transactions

- COMMIT types:
  - Implicit: Database issues an implicit COMMIT before and after any data definition language (DDL) statement
  - Explicit
- Example of COMMIT command:

```
DELETE FROM student_master
WHERE student_name = 'Amit';
COMMIT ;
```

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

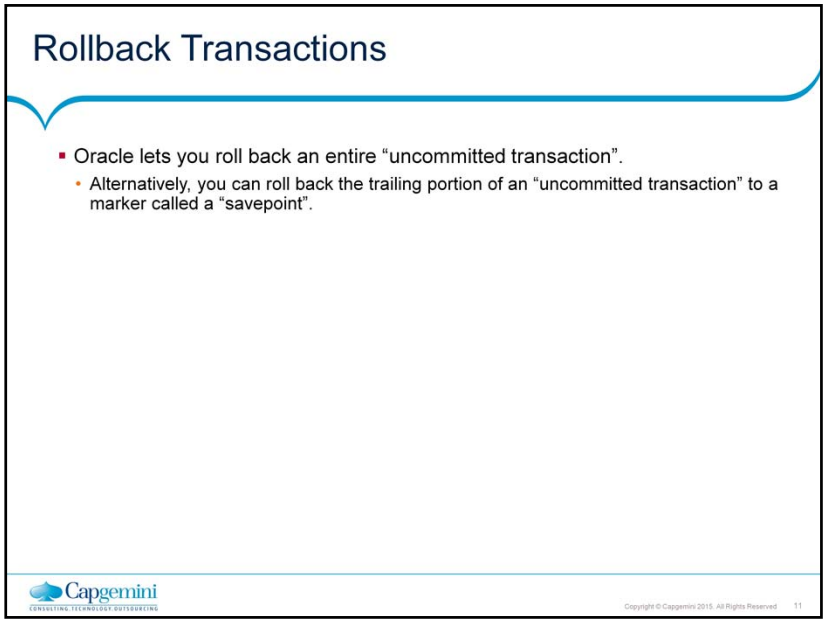
 9

Rollback Transactions

- Rolling back a transaction means “undoing changes” to data that have been performed by SQL statements within an “uncommitted transaction”.
  - Oracle uses “undo tablespaces” (or rollback segments) to store old values.
  - Oracle also uses the “redo log” that contains a record of changes.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



Types of Roll back:

All the following types of rollbacks use the same roll back procedure:

- Statement-level rollback (due to statement or deadlock execution error)
- Rollback to a savepoint
- Rollback of a transaction due to user request
- Rollback of a transaction due to abnormal process termination
- Rollback of all outstanding transactions when an instance terminates abnormally
- Rollback of incomplete transactions during recovery

In rolling back an entire transaction, without referencing any savepoints, there is an occurrence of the following sequence:

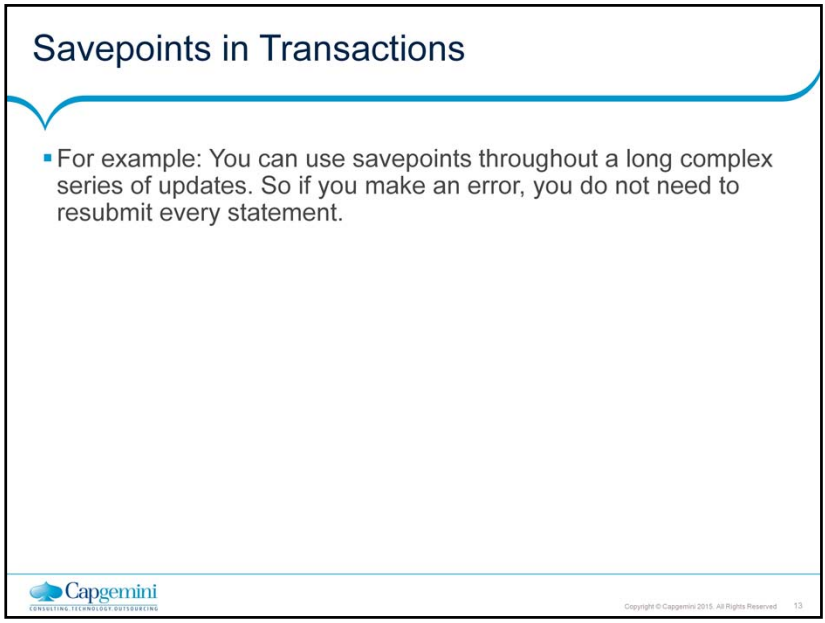
- Oracle undoes all changes made by all the SQL statements in the transaction by using the corresponding undo tablespace.
- Oracle releases all the locks of data for the transaction.
- The transaction ends.

Savepoints in Transactions

- In a transaction, you can declare intermediate markers called "savepoints" within the context of a transaction.
  - By using "savepoints", you can arbitrarily mark your work at any point within a long transaction.
  - In this manner, you can keep an option that is available later to roll back the work performed, however:
    - before the current point in the transaction, and
    - after a declared savepoint within the transaction

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 12



Savepoints in Transactions:

Savepoints are useful in application programs, as well. If a procedure contains several functions, then you can create a savepoint at the beginning of each function.

Then, if a function fails, it is easy:

to return the data to it's state before the function began, and

to re-run the function with revised parameters or perform a RECOVERY action.

After a rollback to a savepoint, Oracle releases the data locks obtained by rolled back statements. As a result:

Other transactions that were waiting for the previously locked resources can proceed.

Other transactions that want to update previously locked rows can do so.

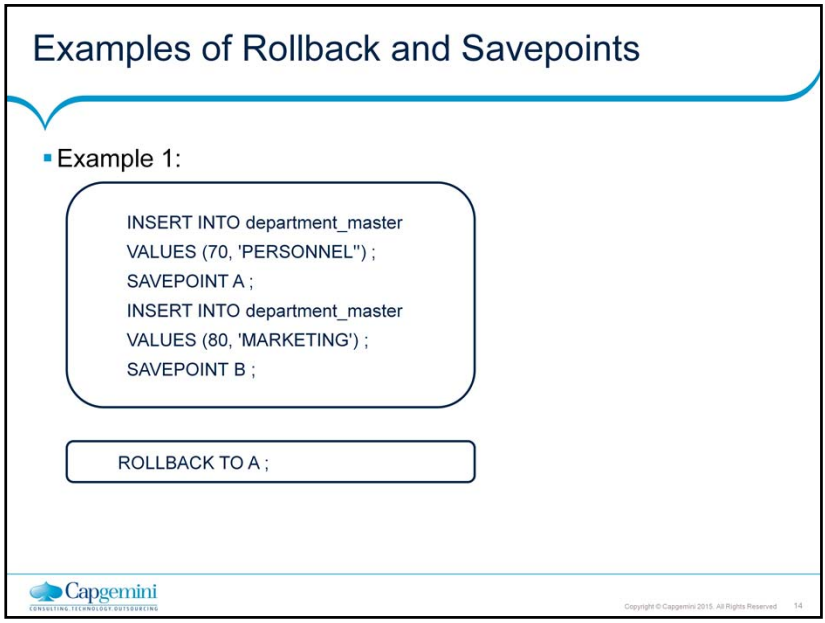
When a transaction is rolled back to a savepoint, there is an occurrence of the following sequence:

Oracle rolls back only the statements run after the savepoint.

Oracle preserves the specified savepoint, but all savepoints that were established after the specified savepoint are lost.

Oracle releases all table and row locks acquired since that savepoint, however, it retains all data locks acquired previous to the savepoint.

The transaction remains active and can be continued.



Examples of Rollback and Savepoints:

In the example in the above slide, after execution of 'ROLLBACK TO A' command, 'INSERT INTO department_master VALUES (70, 'PERSONNEL')' statement got committed (stored in a table permanently) and all other statements after the SAVEPOINT A, will get rolledback (undone).

Advantages of COMMIT and ROLLBACK statements:

With COMMIT and ROLLBACK statements, you can:

- ensure data consistency
- preview data changes before making changes permanent
- group logically related operations

State of the Data after COMMIT:

- Data changes are made permanent in the database.
- The previous state of the data is permanently lost.
- All users can view the results.
- Locks on the affected rows are released. Those rows are available for other users to manipulate.
- All savepoints are erased.

State of the Data after ROLLBACK:

DBMS discards all pending changes by using the ROLLBACK statement:

- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released

Syntax:

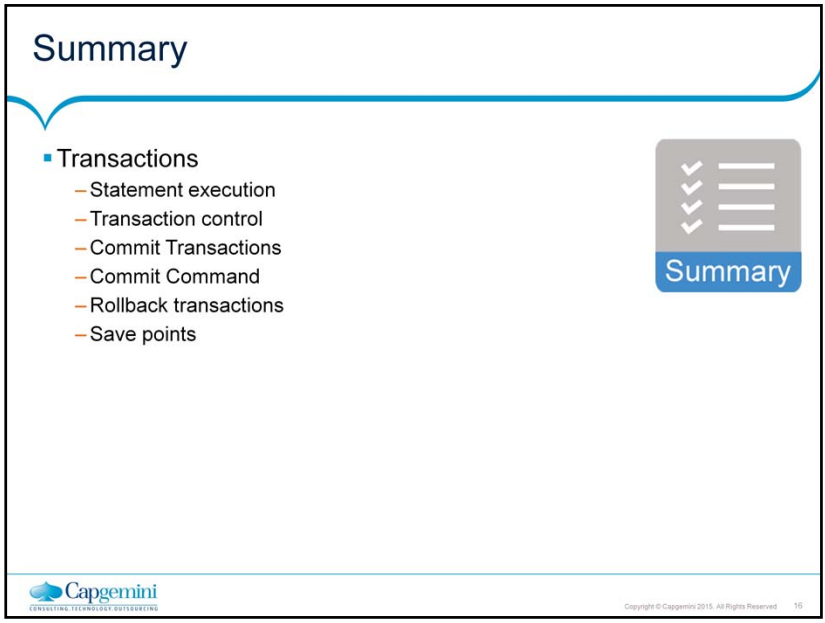
```
UPDATE...
COMMIT;
```

```
INSERT...
ROLLBACK;
```

Example of rolling back changes to a Marker:

- Create a marker in a current transaction by using the SAVEPOINT statement.
- Roll back to that marker by using the ROLLBACK TO SAVEPOINT statement.

```
UPDATE...
SAVEPOINT update_done;
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```


Summary
Transactions
Statement execution
Transaction control
Commit Transactions
Commit Command
Rollback transactions
Save points
Summary
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 16

Review – Questions

- Question 1 : ___ is a logical unit of work.
- Question 2: A transaction is committed when the user issues a DDL statement.
  - True/False
- Question 3: A transaction is rolled back when ___.
  - Option 1: rollback statement is issued
  - Option 2: the user session is abruptly terminated
  - Option 3: an error occurs in DML statement
  - Option 4: none of the above