 



Database Programming with SQL 18-1: Database Transactions Practice Activities

# Objectives

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

# Vocabulary

Identify the vocabulary word for each definition below

|  |  |
| --- | --- |
| **ROLLBACK** | Ends the current transaction making all pending data changes permanent |
| **ROLLBACK** | Enables the user to discard changes made to the database |
| **SAVEPOINT** | Creates a marker in a transaction, which divides the transaction into smaller pieces |
| **Read consistency**[*Read consistency - is an automatic implementation, guarantees that readers of the data see consistent data that is not currently undergoing change.*] | guarantees a consistent view of the data by all users at all times |
| **Lock**[*Locking: Implicit locking occurs in oracle in all statements except SELECT to avoid destructive interaction b/w transactions accessing same resource. EXPLICIT locking is also possible in oracle.*] | Mechanisms that prevent destructive interaction between transactions accessing the same resource that can be granted to the user |
| **Transaction** | a collection of DML statements that form a logical unit of work |

# Try It / Solve It

1. Define the COMMIT, ROLLBACK, and SAVEPOINT statements as they relate to data transactions.

* ·         **COMMIT, ROLLBACK and SAVEPOINT are known as TCL (Transaction Control Language).**
* ·         **COMMIT - make all pending changes permanent.**
* ·         **ROLLBACK - discard the pending changes made. Automatic rollback will occur in case of system failure to protect data integrity.**
* ·         **SAVEPOINT - marker to divide transaction into smaller pieces, ROLLBACK to SAVEPOINT is possible. SAVEPOINT is not a schema object and can't be referenced in data dictionary, lost after commit or rollback (transaction complete).**
* ·         **TRANSACTION-**

·         **Starts with:**

o   **DML (INSERT UPDATE DELETE MERGE)**

·         **And ends with:**

o   **COMMIT,**

o   **Exit normally (implicit commit),**

o   **ROLLBACK,**

o   **DDL [CREATE DROP ALTER RENAME TRUNCATE]**

o   **DCL [GRANT OR REVOKE].**

1. What data will be committed after the following statements are issued?

INSERT INTO R values (5, 6);

SAVEPOINT my\_savepoint\_1; INSERT INTO R values (7, 8);

SAVEPOINT my\_savepoint\_2; INSERT INTO R values (9, 10);

ROLLBACK TO my\_savepoint\_1; INSERT INTO R values (11, 12); COMMIT;



1. Construct a SQL statement for the DJs on Demand D\_SONGS table that deletes the song “All These Years,” inserts a new Country song called ‘Happy Birthday Sunshine’ by “The Sunsets” with a duration of 4 min and an ID = 60. Make sure that all data can be recovered before any changes to the table are made.

**type\_code is not nullable in d\_songs. Purpose of this problem seems to be: show that, since INSERT fails and I am making a transaction, delete will also rollback.**

**Point  e)  or f) below is the solution to this problem, rest is to understand the issue.**

*a)*      *If I use simply below mentioned SQL script, delete is committed, even if insert fails:*

*DELETE FROM d\_songs*

*WHERE title = 'All These Years';*

*INSERT INTO d\_songs (id, title, duration, artist)*

*VALUES (60, 'Happy Birthday Sunshine', '4 min',  'The Sunsets');*

*COMMIT;*

SELECT \*  FROM d\_songs

WHERE title = 'All These Years';



*b)*      *Now I recreated the whole schema using schema creation scripts provided.*

*c)*      *The most straight forward way to verify rollback is below mentioned executed as a script:*

*DELETE FROM d\_songs*

*WHERE title = 'All These Years';*

*INSERT INTO d\_songs (id, title, duration, artist)*

*VALUES (60, 'Happy Birthday Sunshine', '4 min',  'The Sunsets');*

*ROLLBACK;*

**How to: when INSERT fails delete should also rollback.**

*d)*      *Now I recreated the whole schema using schema creation scripts provided.*

**e)**      **Alternative1: Now make both the delete and insert one unit under a block: Save below mentioned code as script in apex:**

**BEGIN**

**DELETE FROM d\_songs**

**WHERE title = 'All These Years';**

**INSERT INTO d\_songs (id, title, duration, artist)**

**VALUES (60, 'Happy Birthday Sunshine', '4 min',  'The Sunsets');**

**COMMIT;**

**END;**

**Even if I skip COMMIT here, I should get same results.**

1. Write an SQL statement that will issue an automatic commit.

**DELETE FROM d\_songs**

**WHERE title = 'All These Years';**

**Изображение выглядит как текст

Автоматически созданное описание**

In apex, above statement will auto-commit in SQL commands window.

I tried  ROLLBACK in SQL commands window:

Rollback statement not applicable. All statements are automatically committed.

1. Give two examples of businesses other than banks that rely on transaction control processes. Describe why each business needs transaction processing control.

·         **Say, I am booking a plane ticket. If this is not a transaction, I could get my money deducted from my debit card, even if the assignment of seat to my unique id failed.**

·         **Say, a passenger airplane has to take off, and it follows a sequence (which includes right from fuel to weather check kind of activities) before it is actually triggered to run for flying, even if one step fails in the sequence, taking off must be cancelled, if it were not a transaction, I won't ever like to be present in a passenger airplane to die.**

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.