**Oracle Lab 12: Transaction**

***Assoc. Prof. Dr. Nguyen Thai Nghe***

1. **Write a report about Database Transactions, save it to Lab12.docx**

Study from the following links (or others):

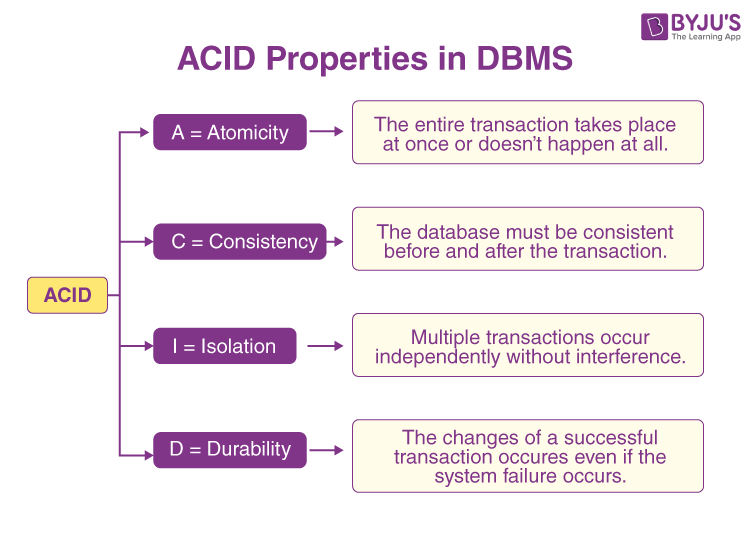
<https://docs.oracle.com/cd/B19306_01/server.102/b14220/transact.htm>

* What is Transaction?

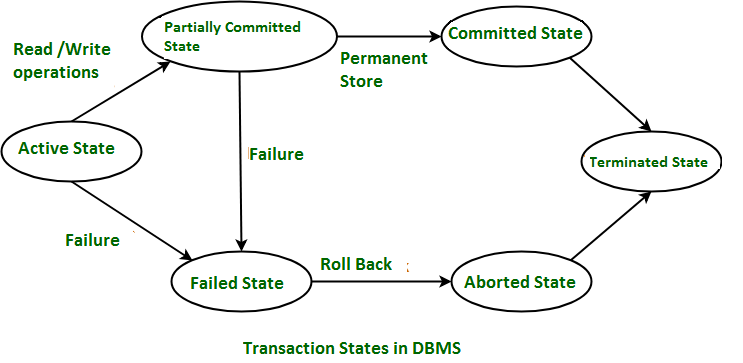
Answer: A **transaction** is a logical unit of work that contains one or more SQL statements. A transaction is an atomic unit. The effects of all the SQL statements in a transaction can be either all **committed** (applied to the database) or all **rolled back** (undone from the database).

* ACID properties

A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.



* States of Transactions



Active − In this state, the transaction is being executed. This is the initial state of every transaction.

Partially Committed − When a transaction executes its final operation, it is said to be in a partially committed state.

Failed − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

Aborted − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts : - Re-start the transaction  
 - Kill the transaction

Committed − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

<https://www.tutorialspoint.com/dbms/dbms_transaction.htm>

Concurrency control

<https://www.tutorialspoint.com/dbms/dbms_concurrency_control.htm>

We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories:

1. Lock based protocols

There are 2 kinds: 1. Binary locks

2. Shared/Exclusive

There are 4 types: 1.Simplistic Lock Protocol

2. Pre-claiming Lock Protocol

3. Two-Phase Locking 2PL

4. Strict Two-Phase Locking

1. Timestamp based protocols

Deadlock

<https://www.tutorialspoint.com/dbms/dbms_deadlock.htm>

In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

Deadlock Prevention: The DBMS inspects the operations and analyzes if they   
can create a deadlock situation. If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed. Ex: Wait-Die Scheme, Wound-Wait Scheme,...

Deadlock avoidance: Those mechanisms can be used to detect any deadlock situation in advance. Methods like "wait-for graph" are available but they are suitable for only those systems where transactions are lightweight having fewer instances of resource. In a bulky system, deadlock prevention techniques may work well.

1. **Practice (***Save to* ***Lab12****.sql***)**

* **Redo all of the examples from this link** <https://www.tutorialspoint.com/plsql/plsql_transactions.htm>

*Replace the Customer table in the above link to Emp table.*

* **Redo all of the examples from this link** [*https://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT038*](https://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT038)

*In file B2111933-lab12-lab13.sql*