Database Creation and Management

Transaction Management

Sasmita Kumari Nayak
CSE Department

TRANSACTION CONCEPTS

- A transaction is an action, or series of actions, carried out by a single user or application program, which reads or updates (writes) the contents of the database.
- It is initiated by a user program written in a high level data manipulation language or programming language like C, C++, java etc.
- It must see a consistent database.
- During transaction execution the database may be temporarily inconsistent.

Cont...

- When the transaction completes successfully (is committed), the database must be consistent.
- After a transaction commits, the changes it has made to the database persist (keep on), even if there are system failures.
- Multiple transactions can execute in parallel.
- Two main issues to deal with:
  - Failures of various kinds, such as hardware failures and system crashes
  - Concurrent execution of multiple transactions

TRANSACTION EXAMPLE

- A transaction is a **logical unit of work** on the database.
- A transaction can have one of two outcomes.
  - If it completes successfully, the transaction is said to have **committed** and the database reaches a new consistent state.
  - If the transaction does not execute successfully, the transaction is **aborted**.
- If a transaction is aborted, the database must be restored to the consistent state it was in before the transaction started. Such a transaction is **rolled back** or **undone**.

Cont…

- For example if we want to give all employees a pay rise of 15%, the operations to perform this action is shown in figure 7.1.
- The BEGIN TRANSACTION and COMMIT statements define the statement in a transaction.
- Any other SQL statements between them are part of the transaction.
- In the transactions consists of two database operations that is the READ and WRITE.

```
Begin_Transaction
    Read EmpNo, Salary
    Salary = Salary * 1.15
    Write EmpNo, Salary
Commit
```

Figure 7.1: A sample transaction

Cont...

- **Read Operation:** Data object is first brought into Main Memory from Disk, and then value is copied into a Program.
- **Write Operation:** Data base object copy in Main Memory is first modified then written to Disk.
- If a transaction completes successfully as illustrated in Figure 7.1 it will commit and the database returns to a new consistent state.

TRANSACTION STATES

- There are the following six states in which a transaction may exist:

1. **Active:** The initial state when the transaction has just started execution.
2. **Partially Committed:** After the final statement has been executed the transaction enters into this state. At any given point of time if the transaction is executing properly, then it is going towards it COMMIT POINT. The values generated during the execution are all stored in secondary or volatile storage.

Cont...

3. **Failed:** If the transaction fails for some reason. The temporary values are no longer required, and the transaction is set to **ROLLBACK**. If the failed transaction has withdrawn Rs. 100/- from account A, then the ROLLBACK operation should add Rs 100/- to account A.
4. **Aborted:** When the ROLLBACK operation is over, the database restored to its state prior to the start of the transaction (can be done only if no internal logical error) or kill the transaction. The transaction is now said to have been aborted.

Cont...

5. **Committed:** If no failure occurs then the transaction reaches the COMMIT POINT. All the temporary values are written to the stable storage and the transaction is said to have been committed.
6. **Terminated:** Either committed or aborted, the transaction finally reaches this state.

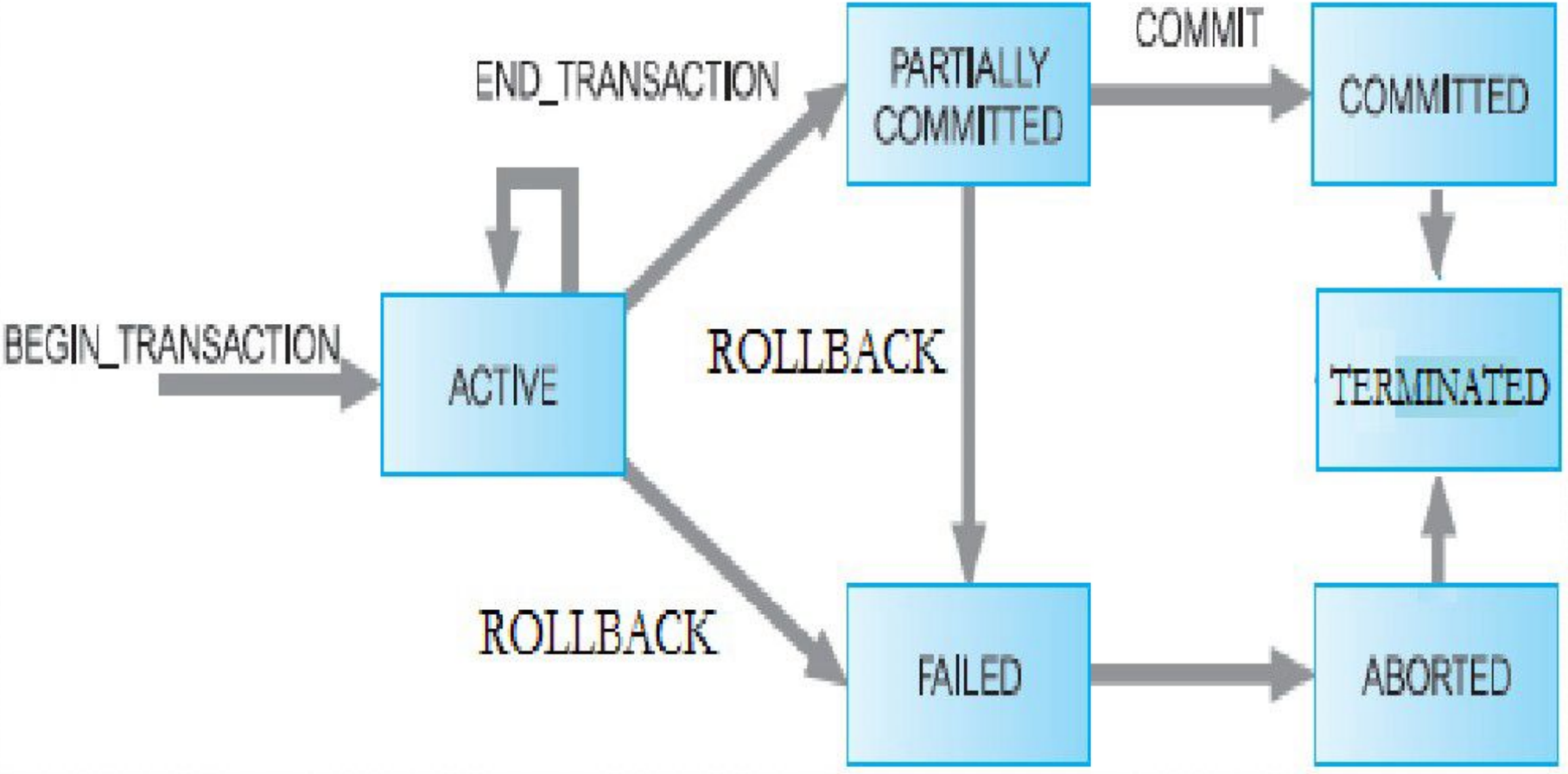

Figure 7.3: State transition diagram for a transaction

Example of state transaction diagram:

- Transfer Rs. 1000 from account A to account B. In active state the transaction can be executed as shown in skeleton of transaction.
- After execution of transaction either it will go to partially committed state or failed state.
- If it enters into partially Committed state and executing properly, then it is going towards it COMMIT POINT.

Transaction:
Begin_Transaction
A's balance - = 1000
B's balance + = 1000
Commit

Cont...

- The values generated during the execution are all stored in secondary or volatile storage. In some situation, consider that B's account cannot be found then it will enter into failed state. If the failed transaction has withdrawn Rs. 1000/- from account A, then the ROLLBACK operation should add Rs 1000/- to account A. Once the transaction failed, it will enter into abort state. In this state the adjustment to A's balance is undone. In both the cases the transaction finally reaches to terminate state

Properties of Transactions (ACID Properties)

- Every transaction, for whatever purpose it is being used, has the following four properties. They are:
  1. Atomicity
  2. Consistency
  3. Isolation
  4. Durability
- Taking the initial letters of these four properties we collectively call them the **ACID Properties**.

Cont..

Atomicity:

- It follows the 'all or nothing' property. This means that either all of the instructions within the transaction will be reflected in the database, or none of them will be reflected.

Consistency:

- A transaction must transform the database from one consistent state to another.
- It is the responsibility of both the DBMS and the application developers to ensure consistency.

Cont...

- The DBMS can ensure consistency by enforcing all the constraints that have been specified on the database schema, such as integrity and enterprise constraints.
- For example, suppose we have a transaction that is intended to transfer money from one bank account to another and the programmer makes an error in the transaction logic and debits one account but credits the wrong account, then the database is in an inconsistent state.
- However, the DBMS would not have been responsible for introducing this inconsistency and would have had no ability to detect the error.

Cont…

Isolation:

- Transactions execute independently of one another. In other words, the partial effects of incomplete transactions should not be visible to other transactions.
- The data used during the execution of a transaction cannot be used by a 2nd transaction until the 1st one complete.
- It is the responsibility of the concurrency control subsystem to ensure isolation.

Cont...

- In case multiple transactions are executing concurrently and trying to access a sharable resource at the same time, the system should create an ordering in their execution so that they should not create any anomaly in the value stored at the sharable resource.
- There are several ways to achieve this and the most popular one is using some kind of locking mechanism. It states that a transaction must first lock the data item that it wishes to access, and release the lock when the accessing is no longer required. Once a transaction locks the data item, other transactions wishing to access the same data item must wait until the lock is released.

Cont...

Durability:

- The effects of a successfully completed (committed) transaction are permanently recorded in the database and must not be lost because of a subsequent failure. In other words, it states that once a transaction has been complete the changes it has made should be permanent.
- It is the responsibility of the recovery subsystem to ensure durability.

Example of ACID property

- For example, we have two accounts A and B, each containing Rs 1000/-. We now start a transaction to deposit Rs 100/- from account A to Account B.
- The transaction has 6 instructions to extract the amount from A and submit it to B. After execution it will show Rs 900/- in A and Rs 1100/- in B.

Transaction:
Read A;
A = A – 100;
Write A;
Read B;
B = B + 100;
Write B;

Atomicity:

- Now, suppose there is a power failure just after instruction 3 (Write A) has been complete. What happens now?

Cont...

- After the system recovers it will show Rs 900/- in A, but the same Rs 1000/- in B.
- It would be said that Rs 100/- evaporated in thin air for the power failure. Clearly such a situation is not acceptable.
- The solution is to keep every value calculated by the instruction of the transaction not in any stable storage (hard disc) but in a volatile storage (RAM), until the transaction completes its last instruction.

Cont...

- When we see that there has not been any error we do something known as a **COMMIT** operation. Its job is to write every temporarily calculated value from the volatile storage on to the stable storage.
- In this way, even if power fails at instruction 3, the post recovery image of the database will show accounts A and B both containing Rs 1000/-, as if the failed transaction had never occurred.

Consistency:

- The sum of A and B is unchanged by the execution of the transaction.

Cont…

Isolation:

- If between steps 3 and 6, another transaction is allowed to access the partially updated database, it will see an inconsistent database (the sum A + B will be less than it should be).
- Isolation can be ensured trivially by running transactions serially, that is one after another.
- However, executing multiple transactions concurrently has significant benefits.

Cont…

Durability:

- Once the user has been notified that the transaction has completed (i.e. the transfer of the Rs. 100 has taken place), the updates to the database by the transaction must persist despite failures.
- As we have seen in the explanation of the Atomicity property, the transaction, if completes successfully, is committed.

Cont…

- Once the COMMIT is done, the changes which the transaction has made to the database are immediately written into permanent storage. So, after the transaction has been committed successfully, there is no question of any loss of information even if the power fails. Committing a transaction guarantees that the transaction has been successfully executed.