Transactions management

Transaction

A database is the subject of many operations. A transfer of funds from a current account to a saving account is a single operation when viewed form a customer's point of view.

A collection of operations that form a single unit of work is called a transaction.

Definition

A transaction is a logical unit of program execution that accesses and possibly updates various data items.

To ensure the integrity of data a database is required to maintain the following properties: atomicity (A), consistency (C), isolation (I) and durability (D).

Example of transaction

To illustrate the concept of a transaction, consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operations:

- Decrement the savings account
- Increment the checking account
- Record the transaction in the transaction journal

The transaction is illustrated below:

Begin Transaction

The SQL statement for the Decrement Savings Account is as follows:

```
UPDATE savings_accounts
SET balance = balance - 500
WHERE account = 3209;
```

The SQL statement for the Increment Checking Account is as follows:

```
UPDATE checking_accounts
SET balance = balance + 500
WHERE account = 3208;
```

The SQL statement for Record in Transaction Journal is as follows:

```
INSERT INTO journal VALUES
(journal_seq.NEXTVAL, '1B', 3209, 3208, 500);
```

The SQL statement for End Transaction is as follows:

```
COMMIT WORK;
```

End Transaction

You created this PDF from an application that is not licensed to print to novaPDF printer (http://www.novapdf.com)

Transaction states

During its execution a transaction passes through several states until it finally commits or aborts. A transaction must be in one of the following states:

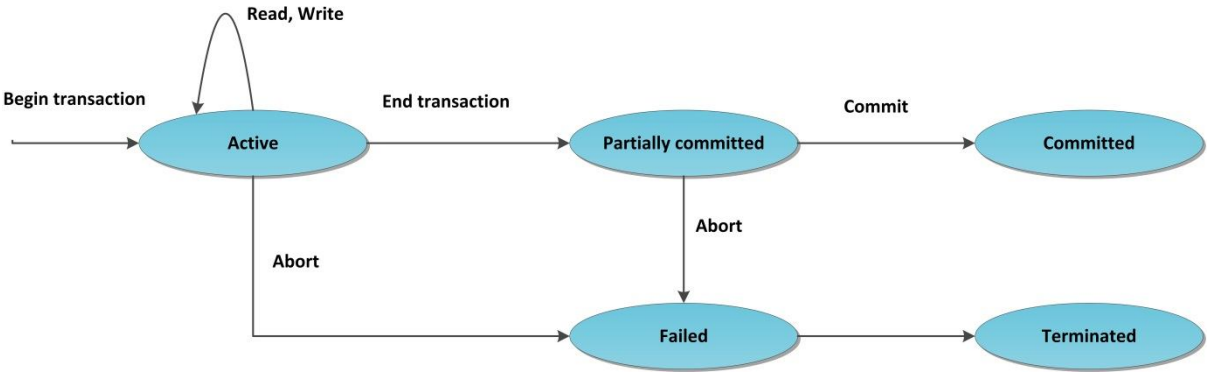
Active, the initial state; the transaction stays in this state while it is executing.

Partially committed, after the final statement has been executed.

Committed, after successful completion

Failed, after the discovery that normal execution can no longer proceed

Terminated/Aborted, after the transaction has rolled back and the database has been restored to its state prior to the start of the transaction.



ACID properties

Atomicity: either all operations of the transaction are reflected properly in the database, or none are. A transaction is an indivisible unit of work.

Consistency: transactions transform a database from one consistent state to another consistence state.

Isolation: transactions execute independently of one another i.e. the partial effect of transaction is not visible to other transactions.

Durability: the effect of a successfully completed (i.e. committed) transaction is permanently recorded in the database and cannot be undone (persistence).

You created this PDF from an application that is not licensed to print to novaPDF printer (http://www.novapdf.com)

Example

Let us consider a simplified banking system of several accounts and a set of transactions that access and update those accounts. Access to the database is accomplished by the following two operations:

Read(X), which transfers the data item X from the database to the area of the transaction that issued the **read** operation.

Write(X), which transfers the data item X from the area of the transaction that issued the **write** operation back to the database.

Let T_i be a transaction that transfers £50 from account A to account B. This transaction can be defined as

T_i: **read**(A);
A = A - 50;
write(A);
read(B);
B = B + 50 ;
write(B)

Let us consider the ACID requirements for this transaction:

Consistency: consistency requires that the sum of A and B is unchanged by the execution of the transaction. This is ensured by the programmer who codes the transaction.

Atomicity*:* suppose that initially the values of A and B are £1000 and £2000 respectively. The sum A+B must always be equal to £3000. Suppose that during the execution of transaction T_i a failure occurred that prevented T_i from completing its execution successfully. Further, suppose that the failure occurred/happened after the **write**(A) operation was executed, but before the **write**(B) operation was executed. In this case the value of accounts A and B in the database are £950 and £2000. £50 was lost/destroyed as a result of the failure. The sum A+B is no longer preserved. The database is in an inconsistent state.

Atomicity ensures that all actions are reflected in the database or none of them. Ensuring atomicity is the responsibility of the DBMS.

Isolation: even if consistency and atomicity are ensured for each transaction, if several transactions are executed concurrently, their operations may interleave in some undesirable way, resulting in an inconsistent state.

Durability: the durability property guarantees that once a transaction completes successfully, all updates that it carried out on the database persist, even if there is system failure after the transaction completes execution. . Ensuring durability is the responsibility of the DBMS.

You created this PDF from an application that is not licensed to print to novaPDF printer (http://www.novapdf.com)