Atomicity: This property ensures that either all operations in a transaction are successfully completed, or none of them are. In other words, a transaction is indivisible; if one part of the transaction fails, the entire transaction is rolled back to its initial state.

Consistency: This property ensures that the database remains in a consistent state before and after the transaction. The integrity constraints, such as foreign key constraints or unique constraints, are not violated during the execution of the transaction.

Isolation: This property ensures that the concurrent execution of transactions does not result in any interference. Each transaction should be isolated from other transactions until it is completed. Isolation prevents the so-called "lost updates," "dirty reads," "non-repeatable reads," and "phantom reads."

Durability: This property ensures that once a transaction is committed, its changes are permanently saved in the database and will not be lost, even in the event of a system failure.

A transaction in SQL with locking and different isolation levels:

```
CREATE TABLE BankAccount (
    AccountID INT PRIMARY KEY,
    Balance DECIMAL(10, 2)
);

-- Insert initial data
INSERT INTO BankAccount (AccountID, Balance) VALUES (1, 1000);

-- Transaction with locking
BEGIN TRANSACTION;
    -- Select and lock the account for update
    SELECT * FROM BankAccount WHERE AccountID = 1 FOR UPDATE;

    -- Simulate some processing time
    -- Update balance or perform other operations here
```

```
  -- Update the balance
   UPDATE BankAccount SET Balance = Balance - 100 WHERE AccountID = 1;


   -- Commit the transaction
COMMIT;


-- Different isolation levels demonstration
 Session 1:
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
   SELECT * FROM BankAccount WHERE AccountID = 1;
COMMIT;
 Session 2:
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;
   SELECT * FROM BankAccount WHERE AccountID = 1;
COMMIT;
```

In this example:

We first create a table BankAccount with an AccountID and Balance.

Then we insert an initial record into the BankAccount table.

We demonstrate a transaction with locking by updating the balance of an account. We use SELECT FOR UPDATE to lock the row during the transaction.

Different isolation levels (READ COMMITTED and READ UNCOMMITTED) by performing SELECT statements on the same row in two separate sessions.