

## Re: Behavior of Connection.commit()

---

*Source:*

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.jdbcdriver/2006-01/msg00051.html>

---

- *From:* Joe Weinstein <joeNOSPAM@xxxxxxx>
  - *Date:* Mon, 23 Jan 2006 11:48:08 -0800
- 

Angel Saenz-Badillos[MS] wrote:

Sam,

The problem is the same as above. If we set `set_implicit_transactions` on we will no longer be able to tell that a serious issue happened on commit and we would not throw an exception:

```
//start transaction
//insert data 1
//execute ddl
exception. At this time insert data 1 has been rolled back!
//insert data 2 //with set_implicit_transactions on this will
create a NEW transaction under the covers.
```

Understood. One other thing to consider is if the driver can know from the DBMS message when (the rare case such as deadlock or DDL) when the DBMS has already killed the whole tx, and in that case, throw an exception from any subsequent connection, statement or result set method that would/could do an update or query (because it may hold locks) (from this connection) saying "The DBMS has killed the current transaction. No further DBMS access is allowed until you call `rollback()`".

```
//COMMIT will no longer throw an exception, it will commit the
transaction started with insert data 2 and you will silently
ignore the fact that data 1 has been rolled back.
```

Correct, but there is a problem as soon as the tx has been rolled back, and there can be troubles even allowing the insert data 2 to proceed. The user application may depend on holding locks in order, and once it is allowed to proceed obtaining the lock for data 2, it may then go on to

## Re: Behavior of Connection.commit()

obtain other locks before trying to commit, and this not-expected order of locking (data 1 and any other data previously locked are now unlocked) may cause deadlocks that could kill other innocent, correct transactions.  
HTH,  
Joe