# Data transaction

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a BEGIN or similar statement, the semantics of SQL are implementation-dependent.

# COMMIT

The COMMIT command makes a transaction's changes to the database permanent. It is used to start all transactions.

COMMIT is used to end a transaction and:

- Write all updates to the database.
- Make the transaction's changes visible to subsequent SNAPSHOT transactions or READ COMMITTED transactions.
- Close open cursors, unless the RETAIN argument is used.

After executing a transaction with [F9] or the

icon, and all operations in the transaction have been successfully performed by the server, the changes to the database must be explicitly committed. This can be done using [Ctrl + Alt + C] or the

icon.

Of course, those competent in SQL can also enter the command directly in SQL Editor.

**Syntax**

```
COMMIT [WORK] [TRANSACTION name] [RELEASE] [RETAIN [SNAPSHOT]];
```

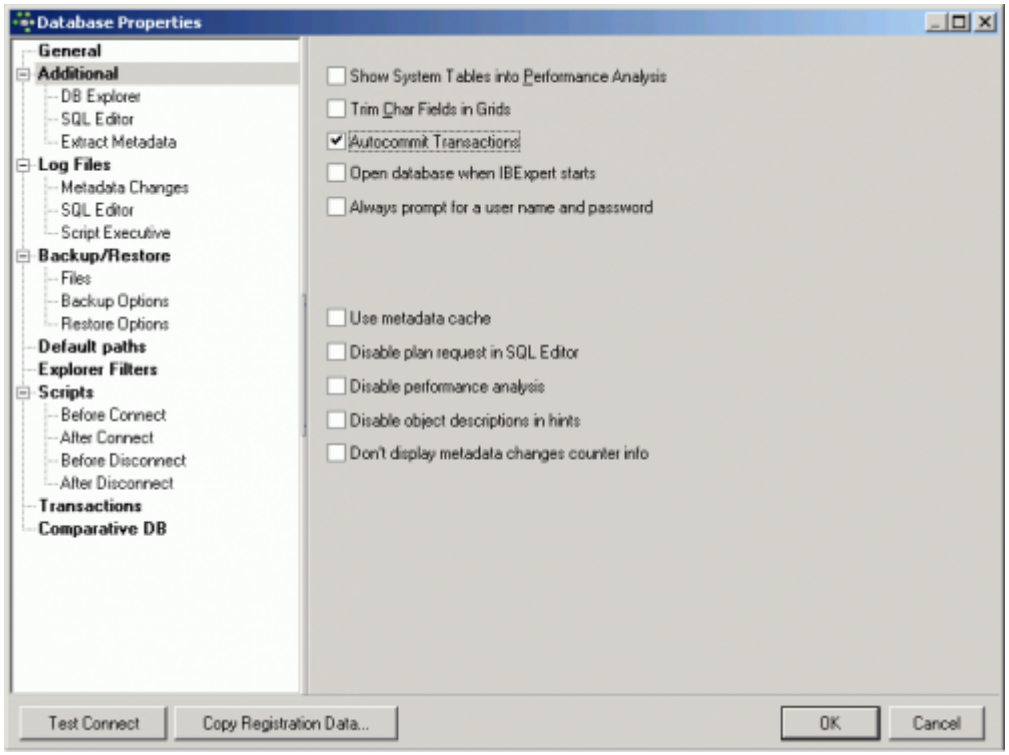
| | |
|---|---|
| WORK | An optional work used for compatibility with other relational databases that require it. |
| TRANSACTION name | Commits a transaction name to database. Without this option, COMMIT affects the default transaction. |
| RELEASE | Available for compatibility with earlier versions of Firebird/InterBase®. |
| RETAIN [SNAPSHOT] | Commits changes and retains current transaction context. |

The transaction name is only valid in an embedded SQL application using SQL or DSQL, where more than one transaction can be active at a time.

A transaction ending with COMMIT is considered a successful termination. Always use COMMIT or ROLLBACK to end the default transaction. Tip: after read-only transactions, which make no database changes, use COMMIT rather than ROLLBACK. The effect is the same, but the performance of subsequent transactions is better and the system resources used by them are reduced.

This statement is not valid inside a trigger, because a trigger is started automatically as part of a larger transaction, with other triggers perhaps firing after it. It is also not valid inside a stored procedure because the procedure might be invoked from a trigger.

In IBExpert it is possible to force all commands to be automatically committed, by checking the *Autocommit Transactions* box in the Database Properties dialog / Additional (menu item: Database / Database Registration Info...):




However, this is *NOT* recommended, as it is all too easy to accidentally drop a database (instead of a database field for example), as the developer is no longer asked for confirmation before committing.

See also:

- Compile, Commit, Rollback
- DCL - Data Control Language
- DDL - Data Definition Language
- DML - Data Manipulation Language
- Data Retrieval

back to top of page

# ROLLBACK

If a transaction's operations did not all complete successfully or satisfactorily, it is possible to roll back the transaction. A rollback restores the data to the state it was in before the transaction started. All changes made by insertions, updates and deletions are reversed.

The ROLLBACK is performed in IBExpert using the

icon or [Ctrl + Alt + R].

Rolling back can of course also be specified by issuing the following statement:

```
ROLLBACK [TRANSACTION name];
```

The transaction name is only required in embedded SQL applications using SQL or DSQL, where more than one transaction can be active at any one time.

It is important to note that when a transaction is rolled back, the changes performed by that transaction are not immediately deleted. Instead, InterBase® flags the transaction associated with that entry as having been rolled back in the Transaction Inventory Page (TIP). Subsequent queries must then reconstruct the row using the version history.

When Firebird/InterBase® performs a garbage collection or database sweep, the server detects that the row entry for the current version does not in fact contain the complete current version. It is then updated and the various data segments and version history relinked to ensure that the current version of the row is stored in the correct place, so that back versions do not need to be read each time.

back to top of page

### ROLLBACK RETAIN

Source: Firebird 2.0 Language Reference Update

*Available in*: DSQL

*Added in*: 2.0

#### Description

Undoes all the database changes carried out in the transaction without closing it. User variables set with RDB$SET_CONTEXT() remain unchanged.

#### Syntax

```
ROLLBACK [WORK] RETAIN [SNAPSHOT]
```

### Note

The functionality provided by ROLLBACK RETAIN has been present since InterBase® 6, but the only way to access it was through the API call isc_rollback_retaining().

back to top of page

## ROLLBACK TO SAVEPOINT

Source: Firebird 2.0 Language Reference Update

*Available in*: DSQL

*Added in*: 1.5

### Description

Undoes everything that happened in a transaction since the creation of the savepoint.

### Syntax

```
ROLLBACK [WORK] TO [SAVEPOINT] name
```

ROLLBACK TO SAVEPOINT performs the following operations:

- All the database mutations performed within the transaction since the savepoint was created are undone. User variables set with RDB$SET_CONTEXT() remain unchanged.
- All savepoints created after the one named are destroyed. All earlier savepoints are preserved, as is the savepoint itself. This means that you can rollback to the same savepoint several times.
- All implicit and explicit record locks acquired since the savepoint are released. Other transactions that have requested access to rows locked after the savepoint must continue to wait until the transaction is committed or rolled back. Other transactions that have not already requested the rows can request and access the unlocked rows immediately.

For a full discussion of savepoints, see SAVEPOINT.