Recordset: How Recordsets Update Records (ODBC)

Article • 08/02/2021

This topic applies to the MFC ODBC classes.

Besides their ability to select records from a data source, recordsets can (optionally) update or delete the selected records or add new records. Three factors determine a recordset's updateability: whether the connected data source is updateable, the options you specify when you create a recordset object, and the SQL that is created.

> **Note**
>
> The SQL on which your `CRecordset` object is based can affect your recordset's updateability. For example, if your SQL contains a join or a **GROUP BY** clause, MFC sets the updateability to FALSE.

> **Note**
>
> This topic applies to objects derived from `CRecordset` in which bulk row fetching has not been implemented. If you are using bulk row fetching, see **Recordset: Fetching Records in Bulk (ODBC)**.

This topic explains:

- Your role in recordset updating and what the framework does for you.
- The recordset as an edit buffer and the differences between dynasets and snapshots.

Recordset: How AddNew, Edit, and Delete Work (ODBC) describes the actions of these functions from the point of view of the recordset.

Recordset: More About Updates (ODBC) completes the recordset update story by explaining how transactions affect updates, how closing a recordset or scrolling affects updates in progress, and how your updates interact with the updates of other users.

Your Role in Recordset Updating

The following table shows your role in using recordsets to add, edit, or delete records, along with what the framework does for you.

Recordset Updating: You and the Framework

Expand table

You	The framework
Determine whether the data source is updateable (or appendable).	Supplies CDatabase member functions for testing the data source's updateability or appendability.
Open an updatable recordset (of any type).	
Determine whether the recordset is updatable by calling `CRecordset` update functions such as `CanUpdate` or `CanAppend`.	
Call recordset member functions to add, edit, and delete records.	Manages the mechanics of exchanging data between your recordset object and the data source.
Optionally, use transactions to control the update process.	Supplies `CDatabase` member functions to support transactions.

For more information about transactions, see Transaction (ODBC).

The Edit Buffer

Taken collectively, the field data members of a recordset serve as an edit buffer that contains one record — the current record. Update operations use this buffer to operate on the current record.

- When you add a record, the edit buffer is used to build a new record. When you finish adding the record, the record that was previously current becomes current again.
- When you update (edit) a record, the edit buffer is used to set the field data

members of the recordset to new values. When you finish updating, the updated record is still current.

When you call AddNew or Edit, the current record is stored so it can be restored later as needed. When you call Delete, the current record is not stored but is marked as deleted and you must scroll to another record.

ⓘ **Note**

The edit buffer plays no role in record deletion. When you delete the current record, the record is marked as deleted, and the recordset is "not on a record" until you scroll to a different record.

Dynasets and Snapshots

Dynasets refresh a record's contents as you scroll to the record. Snapshots are static representations of the records, so a record's contents are not refreshed unless you call Requery. To use all the functionality of dynasets, you must be working with an ODBC driver that conforms to the correct level of ODBC API support. For more information, see ODBC and Dynaset.

See also

Recordset (ODBC)
Recordset: How AddNew, Edit, and Delete Work (ODBC)