Recordset: Creating and Closing Recordsets (ODBC)

Article • 08/02/2021

> **Note**
>
> The MFC ODBC Consumer wizard is not available in Visual Studio 2019 and later. You can still create a consumer manually.

This topic applies to the MFC ODBC classes.

To use a recordset, construct a recordset object and then call its `Open` member function to run the recordset's query and select records. When you finish with the recordset, close and destroy the object.

This topic explains:

- When and how to create a recordset object.
- When and how you can qualify the recordset's behavior by parameterizing, filtering, sorting, or locking it.
- When and how to close a recordset object.

Creating Recordsets at Run Time

Before you can create recordset objects in your program, you typically write application-specific recordset classes. For more information about this preliminary step, see Adding an MFC ODBC Consumer.

Open a dynaset or snapshot object when you need to select records from a data source. The type of object to create depends on what you need to do with the data in your application and on what your ODBC driver supports. For more information, see Dynaset and Snapshot.

To open a recordset

1. Construct an object of your `CRecordset`-derived class.

   You can construct the object on the heap or on the stack frame of a function.

2. Optionally modify the default recordset behavior. For the available options, see Setting Recordset Options.

3. Call the object's Open member function.

In the constructor, pass a pointer to a `CDatabase` object or pass NULL to use a temporary database object that the framework constructs and opens based on the connection string returned by the GetDefaultConnect member function. The `CDatabase` object might already be connected to a data source.

The call to `Open` uses SQL to select records from the data source. The first record selected (if any) is the current record. The values of this record's fields are stored in the recordset object's field data members. If any records were selected, both the `IsBOF` and `IsEOF` member functions return 0.

In your Open call, you can:

- Specify whether the recordset is a dynaset or snapshot. Recordsets open as snapshots by default. Or, you can specify a forward-only recordset, which allows only forward scrolling, one record at a time.

  By default, a recordset uses the default type stored in the `CRecordset` data member `m_nDefaultType`. Wizards write code to initialize `m_nDefaultType` to the recordset type you choose in the wizard. Rather than accepting this default, you can substitute another recordset type.

- Specify a string to replace the default SQL **SELECT** statement that the recordset constructs.

- Specify whether the recordset is read-only or append-only. Recordsets allow full updating by default, but you can limit that to adding new records only or you can disallow all updates.

The following example shows how to open a read-only snapshot object of class `CStudentSet`, an application-specific class:

```
C++
```

```
// Construct the snapshot object
CStudentSet rsStudent( NULL );
// Set options if desired, then open the recordset
if(!rsStudent.Open(CRecordset::snapshot, NULL, CRecordset::readOnly))
    return FALSE;
// Use the snapshot to operate on its records...
```

After you call `Open`, use the member functions and data members of the object to work with the records. In some cases, you might want to requery or refresh the recordset to include changes that have occurred on the data source. For more information, see Recordset: Requerying a Recordset (ODBC).

Tip

The connect string you use during development might not be the same connect string that your eventual users need. For ideas about generalizing your application in this regard, see **Data Source: Managing Connections (ODBC)**.

Setting Recordset Options

After you construct your recordset object but before you call `Open` to select records, you might want to set some options to control the recordset's behavior. For all recordsets, you can:

- Specify a filter to constrain record selection.
- Specify a sort order for the records.
- Specify parameters so you can select records using information obtained or calculated at run time.

You can also set the following option if conditions are right:

- If the recordset is updateable and supports locking options, specify the locking method used for updates.

Note

To affect record selection, you must set these options before you call the `Open` member function.

Closing a Recordset

When you finish with your recordset, you must dispose of it and deallocate its memory.

To close a recordset

1. Call its Close member function.
2. Destroy the recordset object.

   If you declared it on the stack frame of a function, the object is destroyed automatically when the object goes out of scope. Otherwise, use the **`delete`** operator.

`Close` frees the recordset's `HSTMT` handle. It does not destroy the C++ object.

See also

Recordset (ODBC)
Recordset: Scrolling (ODBC)
Recordset: Adding, Updating, and Deleting Records (ODBC)