Recordset: Filtering Records (ODBC)

Article • 08/02/2021

This topic applies to the MFC ODBC classes.

This topic explains how to filter a recordset so that it selects only a particular subset of the available records. For example, you might want to select only the class sections for a particular course, such as MATH101. A filter is a search condition defined by the contents of a SQL **WHERE** clause. When the framework appends it to the recordset's SQL statement, the **WHERE** clause constrains the selection.

You must establish a recordset object's filter after you construct the object but before you call its `Open` member function (or before you call the `Requery` member function for an existing recordset object whose `Open` member function has been called previously).

To specify a filter for a recordset object

1. Construct a new recordset object (or prepare to call `Requery` for an existing object).

2. Set the value of the object's m_strFilter data member.

   The filter is a null-terminated string that contains the contents of the SQL **WHERE** clause but not the keyword **WHERE**. For example, use:

   ```
   m_pSet->m_strFilter = "CourseID = 'MATH101'";
   ```

   not

   ```
   m_pSet->m_strFilter = "WHERE CourseID = 'MATH101'";
   ```

   > **Note**
   >
   > The literal string "MATH101" is shown with single quotation marks above. In

the ODBC SQL specification, single quotes are used to denote a character string literal. Check your ODBC driver documentation for the quoting requirements of your DBMS in this situation. This syntax is also discussed further near the end of this topic.

3. Set any other options you need, such as sort order, locking mode, or parameters. Specifying a parameter is especially useful. For information about parameterizing your filter, see Recordset: Parameterizing a Recordset (ODBC).

4. Call `Open` for the new object (or `Requery` for a previously opened object).

Tip

Using parameters in your filter is potentially the most efficient method for retrieving records.

Tip

Recordset filters are useful for **joining** tables and for using **parameters** based on information obtained or calculated at run time.

The recordset selects only those records that meet the search condition you specified. For example, to specify the course filter described above (assuming a variable `strCourseID` currently set, for instance, to "MATH101"), do the following:

```
// Using the recordset pointed to by m_pSet

// Set the filter
m_pSet->m_strFilter = "CourseID = " + strCourseID;

// Run the query with the filter in place
if ( m_pSet->Open( CRecordset::snapshot, NULL, CRecordset::readOnly ) )

// Use the recordset
```

The recordset contains records for all class sections for MATH101.

Notice how the filter string was set in the example above, using a string variable. This is the typical usage. But suppose you wanted to specify the literal value 100 for the course ID. The following code shows how to set the filter string correctly with a literal value:

```
m_strFilter = "StudentID = '100'";   // correct
```

Note the use of single quote characters; if you set the filter string directly, the filter string is **not**:

```
m_strFilter = "StudentID = 100";   // incorrect for some drivers
```

The quoting shown above conforms to the ODBC specification, but some DBMSs might require other quote characters. For more information, see SQL: Customizing Your Recordset's SQL Statement (ODBC).

> **Note**
>
> If you choose to override the recordset's default SQL string by passing your own SQL string to `Open`, you should not set a filter if your custom string has a **WHERE** clause. For more information about overriding the default SQL, see **SQL: Customizing Your Recordset's SQL Statement (ODBC)**.

See also

Recordset (ODBC)
Recordset: Sorting Records (ODBC)
Recordset: How Recordsets Select Records (ODBC)
Recordset: How Recordsets Update Records (ODBC)
Recordset: Locking Records (ODBC)