Recordset: Working with Large Data Items (ODBC)

Article • 08/02/2021

This topic applies to both the MFC ODBC classes and the MFC DAO classes.

> **Note**
>
> If you are using the MFC DAO classes, manage your large data items with class **CByteArray** rather than class **CLongBinary**. If you are using the MFC ODBC classes with bulk row fetching, use `CLongBinary` rather than `CByteArray`. For more information about bulk row fetching, see **Recordset: Fetching Records in Bulk (ODBC)**.

Suppose your database can store large pieces of data, such as bitmaps (employee photographs, maps, pictures of products, OLE objects, and so on). This kind of data is often referred to as a Binary Large Object (or BLOB) because:

- Each field value is large.
- Unlike numbers and other simple data types, it has no predictable size.
- The data is formless from the perspective of your program.

This topic explains what support the database classes provide for working with such objects.

Managing Large Objects

Recordsets have two ways to solve the special difficulty of managing binary large objects. You can use class CByteArray or you can use class CLongBinary. In general, `CByteArray` is the preferred way to manage large binary data.

`CByteArray` requires more overhead than `CLongBinary` but is more capable, as described in The CByteArray Class. `CLongBinary` is described briefly in The CLongBinary Class.

For detailed information about using `CByteArray` to work with large data items, see

Technical Note 45.

CByteArray Class

`CByteArray` is one of the MFC collection classes. A `CByteArray` object stores a dynamic array of bytes — the array can grow if needed. The class provides fast access by index, as with built-in C++ arrays. `CByteArray` objects can be serialized and dumped for diagnostic purposes. The class supplies member functions for getting and setting specified bytes, inserting and appending bytes, and removing one byte or all bytes. These facilities make parsing the binary data easier. For example, if the binary object is an OLE object, you might have to work through some header bytes to reach the actual object.

Using CByteArray in Recordsets

By giving a field data member of your recordset the type `CByteArray`, you provide a fixed base from which RFX can manage the transfer of such an object between your recordset and the data source and through which you can manipulate the data inside the object. RFX needs a specific site for retrieved data, and you need a way to access the underlying data.

For detailed information about using `CByteArray` to work with large data items, see Technical Note 45.

CLongBinary Class

A CLongBinary object is a simple shell around an `HGLOBAL` handle to a block of storage allocated on the heap. When it binds a table column containing a binary large object, RFX allocates the `HGLOBAL` handle when it needs to transfer the data to the recordset and stores the handle in the `CLongBinary` field of the recordset.

In turn, you use the `HGLOBAL` handle, `m_hData`, to work with the data itself, operating on it as you would on any handle data. This is where CByteArray adds capabilities.

⊗ Caution

CLongBinary objects cannot be used as parameters in function calls. In addition, their implementation, which calls `::SQLGetData`, necessarily slows scrolling performance for a scrollable snapshot. This might also be true when you use an `::SQLGetData` call yourself to retrieve dynamic schema columns.

See also

Recordset (ODBC)
Recordset: Obtaining SUMs and Other Aggregate Results (ODBC)
Record Field Exchange (RFX)