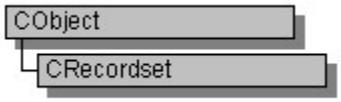
CRecordset 类详解



CRecordset 对象代表从一个数据源选择的一组记录的集合,被称作“记录集”.CRecordset 对象可以以两种形式使用:动态集和快照.动态集是与其它用户的数据更新保持同步的动态数据集.快照是数据的静态视图.每一种形式都代表打开记录集时固定的一组记录,但是当滚动到动态集中的一个记录时,动态集将反映后来由其它用户或由应用程序中其它记录集对此记录所做的改变.

注意:如果正在使用数据访问对象（DAO）类,而不是打开数据库连接（ODBC）类,请使用类 CDaoRecordset 来代替.有关的更多信息,请参阅联机文档“Visual C++程序员指南”中的“数据库主题（通用）”与“DAO 和 MFC”.

要使用任何一种记录集,通常需要从 CRecordset 派生一个应用程序指定的记录集类.记录集从一个数据源中选择记录,然后用户就可以:

- **在这些记录中滚动.**
- **更新记录并指定一种加锁模式.**
- **过滤记录集,以获得那些从数据源中选择出来的可利用的记录.**
- **排序记录集.**
- **参数化该记录集以定制它的具有要直到运行时才知道的信息的选项.**

要使用的类,打开一个数据库并构造一个记录集对象,给构造函数传递一个指向 CDatabase 对象的指针.然后调用记录集的 Open 成员函数,在此可以指定该对象是一个动态集还是一个快照.调用 Open 来从数据源中选择数据.在记录集对象被打开之后,用它的成员函数和数据成员来滚动和操作记录.可用的操作根据对象是一个动态集还是一个快照（这依赖于打开数据库连接（ODBC）数据源的性能）,是可更新的还是只读的,是否实现了成组行检取而不同.为了刷新从调用 Open 以来可能被改变或添加的记录,可以调用对象的 Requery 成员函数.当使用完对象之后,调用对象的 Close 成员函数,并销毁此对象.

在一个派生的 CRecordset 类中,使用记录字段交换（RFX）或成组记录字段交换（Bulk RFX）来支持读取和更新记录字段.

一 SQL 查询

记录集的建立实际上主要是一个查询过程,SQL 的 SELECT 语句用来查询数据源.在建立记录集时,CRecordset 会根据一些参数构造一个 SELECT 语句来查询数据源,并用查询的结果创建记录集.明白这一点对理解 CRecordset 至关重要. SELECT 语句的句法如下:

```
SELECT rfx-field-list FROM table-name [WHERE m_strFilter]
[ORDER BY m_strSort]
```

其中 table-name 是表名,rfx-field-list 是选择的列（字段）. WHERE 和 ORDER BY 是两个子句,分别用来过滤和排序.下面是 SELECT 语句的一些例子:

```
SELECT CourseID, InstructorID FROM Section
SELECT * FROM Section WHERE CourseID= ‘MATH202’ AND Capacity=15
SELECT InstructorID FROM Section ORDER BY CourseID ASC
```

其中第一个语句从 Section 表中选择 CourseID 和 InstructorID 字段. 第二个语句从 Section 表中选择 CourseID 为 MATH202 且 Capacity 等于 15 的记录,在该语句中使用了象 " AND " 或 " OR " 这样的逻辑连接符.要注意在 SQL 语句中引用字符串、日期或时间等类型的数据时要用单引号括起来,而数值型数据则不用. 第三个语句从 Section 表中选择 InstructorID 列并且按 CourseID 的升序排列,若要降序排列,可使用关键字 DESC.

二 建立记录集

```
#include <afxdb.h>
CRecordset::Open
virtual BOOL Open( UINT nOpenType = AFX_DB_USE_DEFAULT_TYPE, LPCTSTR lpszSQL = NULL, DWORD dwOptions = none );
```

throw(CDBException, CMemoryException);

返回值:

如果 CRecordset 对象被成功打开,则返回非零值;否则,如果 CDatabase::Open（如果被调用了）返回 0,则返回 0.

参数:

1 nOpenType

接收缺省值 AFX_DB_USE_DEFAULT_TYPE,或使用下列 enumOpenType 枚举值之一:

- CRecordset::dynaset 可以双向滚动的记录集.当记录集被打开时,记录集的全体成员和记录的顺序都被确定,但是伴随一次获取操作,其它用户对数据值所做的改变是可见的.dynaset 对键值驱动型记录集亦有效.
- CRecordset::snapshot 可以双向滚动的静态记录集.当记录集被打开时,记录集的全体成员和记录的顺序都被确定了;当记录被获取时,其数据值是确定的.其它用户在记录集未被关闭和重新打开之前是不可见的.
- CRecordset::dynamic 可以双向滚动的记录集.伴随一次获取操作,其它用户对成员,顺序和数据值所做的改变是可见的.注意,许多 ODBC 驱动器是不支持这种类型的记录集的.
- CRecordset::forwardOnly 只能向前滚动的只读的记录集.

对于 CRecordset 来说,缺省的值是 CRecordset::snapshot.这种缺省值机制,允许 Visual C++向导与具有不同缺省值的 ODBC CRecordset 和 DAO CDaoRecordset 进行交互.

更多参考信息,请查看“Visual C++程序员指南”中的文章“Recordset (ODBC)”.与此有关的信息,参见“ODBC SDK 程序员参考”中文章的“使用块和可滚动游标”.

提示:如果不支持要求的类型,框架将抛出一个异常.

2 lpszSQL

一个包含下列值之一的字符串指针:

- 一个 NULL 指针.
- 一个表名.
- 一条 SQL SELECT 语句（可选择带一条 SQL WHERE 或 ORDER BY 子句）.
- 一条 CALL 语句,指定一个预定义查询（存储过程）名.注意,不能在卷括号和 CALL 关键字之间插入空格.

有关这个字符串的更多信息,请参见说明下的表,以及对 ClassWizard 的作用的讨论.

注意:

在用户结果集中,列的顺序必须与在的重载的 DoFieldExchange 或 DoBulkFieldExchange 函数中的 RFX 和 Bulk RFX 函数调用的顺序相匹配.

3 dwOptions

一个 bitmask,它可以指定下列值的组合.这些值中的某些是相互独立的.缺省的值是 none.

- CRecordset::none 未设置选项.此参数值与所有其它的值是相互独立的.缺省的,记录集可以用 Edit 或 Delete 来更新,并允许用 AddNew 来添加新记录.这种可更新性依赖于数据源,就像所指定的 nOpenType 选项.成组添加的优化是没有用的.成组行检取不会被实现.在记录集中导航时,不能略过被删除的记录.书签是没有用的.实现了自动脏字段检查.
- CRecordset::appendOnly 不允许在记录集中进行 Edit 或 Delete.值允许 AddNew.此选项与 CRecordset::readOnly 相互独立.
- CRecordset::readOnly 用只读方式打开记录集.此选项与 CRecordset::appendOnly 相互独立.
- CRecordset::optimizeBulkAdd 使用预备的 SQL 语句来优化一次添加多个记录.只有在不使用 ODBC API 函数 SQLSetPos 来更新记录集时才使用.第一次更新确定将哪一个字段标记为脏的.此选项与 CRecordset::useMultiRowFetch 相互独立.
- CRecordset::useMultiRowFetch 实现成组行检取,允许在一次检取操作中获取多行.这是一个高级特征,是设计来提高性能的;但是,ClassWizard 不支持成组记录字段交互.此选项与 CRecordset::optimiazBulkAdd 相互独立.注意,如果指定了 CRecordset::useMultiRowFetch,则选项 CRecordset::noDirtyFieldCheck 将自动被返回（双缓冲将无效）;对于只向前记录集,选项 CRecordset::useExtendedFetch 将被自动返回.
- CRecordset::skipDeletedRecords 当在记录集中导航时,略过所有已被删除的记录.这将使某个相关获取的性能下降.在一个只向前的记录集中,此选项是无效的.注意,CRecordset::skipDeletedRecords 与“驱动器包装”相似,驱动器包装意味着被删除的行被从记录集中移走.但是,如果的驱动器包装记录,则它将只略过那些被删除的记录;而不会

略过在记录集打开时被其它用户删除的记录.CRecordset::skipDeletedRecords 将会略过被其它用户删除的记录.

• CRecordset::useBookarks 允许在记录集中使用书签,如果支持书签的话.书签会使数据检取变慢,但是会提高数据导航的性能.在只向前的记录集中此选项是无效的.

• CRecordset::noDirtyFieldCheck 关闭自动脏数据检查(双缓冲).这将提高性能;但是,必须通过调用 SetFieldDirty 和 SetFieldNull 成员函数来手动标记变脏的字段.注意,CRecordset 类中的双缓冲类似于 CDaoRecordset 中的双缓冲.但是,在 CRecordset 中,不能在独立字段中使双缓冲有效;只能对所有字段使它有效或无效.注意,如果指定了选项 CRecordset::useMultiRowFetch,则 CRecordset::noDirtyFieldCheck 将被自动返回;但是,SetFieldDirty 和 SetFieldNull 不能在实现成组行检取的记录集中使用.

• CRecordset::executeDirect 不要使用预备的 SQL 语句.如果永远不会调用 Requery 成员函数,则指定此选项来提高性能.

• CRecordset::useExtendedFetch 实现 SQLException 来代替 SQLFetch.这是设计来在只向前的记录集中实现成组行检取的.如果在一个只向前的记录集中指定了选项 CRecordset::userMultiRowFetch, 则 CRecordset::useExtendedFetch 将被自动返回.

• CRecordset::userAllocMultiRowBuffers 用户将为数据分配存储缓存.如果想分配自己的存储区,将此选项用与 CRecordset::useMultiRowFetch 连接;否则,框架将自动分配必要的存储区.注意,指定了 CRecordset::userAllocMutiRowBuffers,而没有指定 CRecordset::useMultiRowFetch,将导致一个失败断言.

说明:

必须调用此成员函数来运行记录集定义的查询.在调用 Open 之前,必须构造记录集对象.

此记录集与数据源的连接依赖于在调用 Open 之前是如何构造这个记录集的.如果将一个没有连接到一个数据源的 CDatabase 对象传递给记录集构造函数,此成员函数使用 GetDefaultConnect 来尝试打开该数据库对象.如果将 NULL 传递给记录集构造函数,则此成员函数为构造一个 CDatabase 对象,并且 Open 试图连接给数据库对象.有关关闭记录集和在这些不同的环境下的连接的细节,请参见 Close.

注意:

通过一个 CRecordset 对象对数据源的访问总是被共享的.不像 CDaoRecordset 类,不能使用一个 CRecordset 对象来打开一个具有独占访问的数据源.

建立记录集后,用户可以随时调用 Requery 成员函数来重新查询和建立记录集. Requery()有两个重要用途:

- 使记录集能反映用户对数据源的改变.
- 按照新的过滤或排序方法查询记录并重新建立记录集.

在调用 Requery 之前,可调用 CanRestart 来判断记录集是否支持 Requery 操作. 要记住 Requery 只能在成功调用 Open 后调用,所以程序应调用 IsOpen 来判断记录集是否已建立. 函数的声明为:

```
virtual BOOL Requery( );throw( CDBException, CMemoryException );
```

返回 TRUE 表明记录集建立成功,否则返回 FALSE. 若函数内部出错则产生异常.

```
BOOL CanRestart( ) const; //若支持 Requery 则返回 TRUE
BOOL IsOpen( ) const; //若记录集已建立则返回 TRUE
```

CRecordset 类有两个公共数据成员 m_strFilter 和 m_strSort 用来设置对记录的过滤和排序. 在调用 Open 或 Requery 前,如果在这两个数据成员中指定了过滤或排序,那么 Open 和 Requery 将按这两个数据成员指定的过滤和排序来查询数据源.

成员 m_strFilter 用于指定过滤器.m_strFilter 实际上包含了 SQL 的 WHERE 子句的内容,但它不含 WHERE 关键字.

使用 m_strFilter 的一个例子为:

```
m_pSet->m_strFilter=“CourseID=‘MATH101’”; //只选择 CourseID 为 MATH101 的记录
if(m_pSet->Open(CRecordset::snapshot, “Section”))
. . . . . .
```

成员 m_strSort 用于指定排序. m_strSort 实际上包含了 ORDER BY 子句的内容,但它不含 ORDER BY 关键字.

使用 m_strSort 的一个例子为:

```
m_pSet->m_strSort=“CourseID DESC”; //按 CourseID 的降序排列记录
m_pSet->Open();
. . . . . .
```

事实上,Open 函数在构造 SELECT 语句时,会把 m_strFilter 和 m_strSort 的内容放入 SELECT 语句的 WHERE 和 ORDER BY 子句中. 如果在 Open 的 lpszSQL 参数中已包括了 WHERE 和 ORDER BY 子句,那么 m_strFilter 和 m_strSort 必需为空.

调用无参数成员函数 Close 可以关闭记录集. 在调用了 Close 函数后,程序可以再次调用 Open 建立新的记录集. CRecordset 的析构函数会调用 Close 函数,所以当删除 CRecordset 对象时记录集也随之关闭.

三 滚动记录

CRecordset 提供了几个成员函数用来在记录集中滚动,如下所示.当用这些函数滚动到一个新记录时,框架会自动地把新记录的内容拷贝到域数据成员中.

```
void MoveNext(); //前进一个记录
void MovePrev(); //后退一个记录
void MoveFirst(); //滚动到记录集中的第一个记录
void MoveLast(); //滚动到记录集中的最后一个记录

void SetAbsolutePosition(long nRows);
```

该函数用于滚动到由参数 nRows 指定的绝对位置处. 若 nRows 为负数,则从后往前滚动. 例如,当 nRows 为-1 时,函数就滚动到记录集的末尾. 注意,该函数不会跳过被删除的记录.

```
virtual void Move( long nRows, WORD wFetchType = SQL_FETCH_RELATIVE );
```

该函数功能强大. 通过将 wFetchType 参数指定为 SQL_FETCH_NEXT、SQL_FETCH_PRIOR、SQL_FETCH_FIRST、SQL_FETCH_LAST 和 SQL_FETCH_ABSOLUTE,可以完成上面五个函数的功能.若 wFetchType 为 SQL_FETCH_RELATIVE,那么将相对当前记录移动,若 nRows 为正数,则向前移动,若 nRows 为负数,则向后移动.

如果在建立记录集时选择了 CRecordset::skipDeletedRecords 选项,那么除了 SetAbsolutePosition 外,在滚动记录时将跳过被删除的记录,这一点对象 FoxPro 这样的数据库十分重要.

如果记录集是空的,那么调用上述函数将产生异常. 另外,必须保证滚动没有超出记录集的边界. 调用 IsEOF 和 IsBOF 可以进行这方面的检测.

```
BOOL IsEOF() const;
```

如果记录集为空或滚动过了最后一个记录,那么函数返回 TRUE,否则返回 FALSE.

```
BOOL IsBOF() const;
```

如果记录集为空或滚动过了第一个记录,那么函数返回 TRUE,否则返回 FALSE.

下面是一个使用 IsEOF 的例子:

```
while(!m_pSet->IsEOF())
m_pSet->MoveNext();
```

调用 GetRecordCound 可获得记录集中的记录总数,该函数的声明为:

```
long GetRecordCount() const;
```

要注意这个函数返回的实际上是用户在记录集中滚动的最远距离. 要想真正返回记录总数,只有调用 MoveNext()移动到记录集的末尾(用 MoveLast()达不到这种要求).

四 修改、添加和删除记录

要修改当前记录,应该按下列步骤进行:

调用 Edit 成员函数. 调用该函数后就进入了编辑模式,程序可以修改域数据成员. 注意不要在一个空的记录集中调用 Edit,否则会产生异常. Edit 函数会把当前域数据成员的内容保存在一个缓冲区中,这样做有两个目的,一是可以与域数据成员作比较以判断哪些字段被改变了,二是在必要的时侯可以恢复域数据成员原来的值. 若再次调用 Edit,则将从缓冲区中恢复域数据成员,调用后程序仍处于编辑模式. 调用 Move(AFX_MOVE_REFRESH)或 Move(0)可退出编辑模式(AFX_MOVE_REFRESH 的值为 0),同时该函数会从缓冲区中恢复域数据成员.

设置域数据成员的新值的方法:调用 Update()完成编辑.Update()把变化后的记录写入数据源并结束编辑模式.

要向记录集中添加新的记录,应该按下列步骤进行:

调用 AddNew 成员函数.调用该函数后就进入了添加模式,该函数把所有的域数据成员都设置成 NULL(注意,在数据库术语中,NULL 是指没有值,这与 C++的 NULL 是不同的).与 Edit 一样,AddNew 会把当前域数据成员的内容保存在一个缓冲区中,在必要的时侯,程序可以再次调用 AddNew 取消添加操作并恢复域数据成员原来的值,调用后程序仍处于添加模式. 调用 Move(AFX_MOVE_REFRESH)可退出添加模式,同时该函数会从缓冲区中恢复域数据成员. 调用 Update(),Update()把域数据成员中的内容作为新记录写入数据源,从而结束了添加. 如果记录集是快照,那么在添加一个新的记录后,需要调用 Requery 重新查询,因为快照无法反映添加操作.

要删除记录集的当前记录,应按下面两步进行:

1 调用 Delete 成员函数. 该函数会同时给记录集和数据源中当前记录加上删除标记. 注意不要在一个空记录集中调用 Delete,否则会产生一个异常.

2 滚动到另一个记录上以跳过删除记录.

上面提到的函数声明为:

```
virtual void Edit( );throw( CDBException, CMemoryException );
virtual void AddNew( );throw( CDBException );
virtual void Delete( );throw( CDBException );
virtual BOOL Update( );throw( CDBException );
```

若更新失败则函数返回 FALSE,且会产生一个异常. 在对记录集进行更改以前,程序也许要调用下列函数来判断记录集是否是可以更改的,因为如果在不能更改的记录集中进行修改、添加或删除将导致异常的产生.

```
BOOL CanUpdate( ) const; //返回 TRUE 表明记录是可以修改、添加和删除的.
BOOL CanAppend( ) const; //返回 TRUE 则表明可以添加记录.
```