Open Database Connectivity (ODBC)

- We'd like more power to manipulate DBs
  - looping, updating
- Programs can benefit from using DBs
  - statistical analysis: use info from DB rather reading text file or hard coding in header

What is ODBC?

- API that database vendors can implement for their system via ODBC driver
- Your program makes requests via the API
- Driver talks to the DBMS using SQL
- Available drivers:
  - SQL Server, Access, FoxPro, Excel, dBase, Paradox, text files

MFC's Database Classes

- MFC provides interface on top of the ODBC API, for use in C++
- CDatabase
- CRecordset

How to connect to a datasource...

- Use ODBC data source administrator in the Control Panel to register your datasource
- Create classes derived from the Crecordset type for each table in your database
- Instantiate a Cdatabase object and use it to connect to your datasource
- Initialize your Crecordsets with the connected database

CDatabase

- Member functions:
  - Open("Name of data source")
  - Close
  - IsOpen
  - BeginTrans, CommitTrans, Rollback
  - ExecuteSQL("SQL statement")
- Examples...

CRecordset

- High level view…
  - Hardcode the table name to connect
  - Object exposes one record of the table at time
  - Member variable for each column in table
  - These members exchange data with the associated column in the current record

CRecordset

- Member Vars:
  - One for each column
  - m_strFilter, a string holding WHERE clause
- Member Functions
  - Crecordset(&Cdatabase)
  - Open, Close
  - IsOpen, IsBof, IsEof
  - AddNew, Delete, Update, Edit
  - MoveFirst, MoveLast, MoveNext, MovePrev
  - IsFieldNull, IsFieldDirty, Requery

Example: Connect with Student Registration DB

- Instructor (InstructorID, Name, RoomNo)
- Student (StudentID, Name, GradYear)
- Course (CourseID, CourseTitle, Hours)
- Section (CourseID,SectionNo, InstructorID, RoomNo, Schedule, Capacity)

Student Crecordset

```
class Students1 : public CRecordset
{
public:
	Students1(CDatabase* pDatabase = NULL);
	DECLARE_DYNAMIC(Students1)

// Field/Param Data
	//{{AFX_FIELD(Students1, CRecordset)
	long m_StudentID;
	CString	m_Name;
	int	m_GradYear;
	//}}AFX_FIELD
}
```

```
Students1::Students1(CDatabase* pdb)
	: CRecordset(pdb)
{
	//{{AFX_FIELD_INIT(Students1)
	m_StudentID = 0;
	m_Name = _T("");
	m_GradYear = 0;
	m_nFields = 3;
	//}}AFX_FIELD_INIT
	m_nDefaultType = dynaset;
}


CString Students1::GetDefaultConnect()
{
	return _T("ODBC;DSN=Stdregistration");
}

CString Students1::GetDefaultSQL()
{
	return _T("[Student]");
}

void Students1::DoFieldExchange(CFieldExchange* pFX)
{
	//{{AFX_FIELD_MAP(Students1)
	pFX->SetFieldType(CFieldExchange::outputColumn);
	RFX_Long(pFX, _T("[StudentID]"), m_StudentID);
	RFX_Text(pFX, _T("[Name]"), m_Name);
	RFX_Int(pFX, _T("[GradYear]"), m_GradYear);
	//}}AFX_FIELD_MAP
}
```

```
class Sections : public CRecordset
{
public:
	Sections(CDatabase* pDatabase = NULL);
	DECLARE_DYNAMIC(Sections)

// Field/Param Data
	//{{AFX_FIELD(Sections, CRecordset)
	CString	m_CourseID;
	CString	m_SectionNo;
	CString	m_InstructorID;
	CString	m_RoomNo;
	CString	m_Schedule;
	int	m_Capacity;
	//}}AFX_FIELD
}
```

```
Sections::Sections(CDatabase* pdb)
	: CRecordset(pdb)
{
	//{{AFX_FIELD_INIT(Sections)
	m_CourseID = _T("");
	m_SectionNo = _T("");
	m_InstructorID = _T("");
	m_RoomNo = _T("");
	m_Schedule = _T("");
	m_Capacity = 0;
	m_nFields = 6;
	//}}AFX_FIELD_INIT
	m_nDefaultType = dynaset;
}


CString Sections::GetDefaultConnect()
{
	return _T("ODBC;DSN=Stdregistration");
}

CString Sections::GetDefaultSQL()
{
	return _T("[Section]");
}

void Sections::DoFieldExchange(CFieldExchange* pFX)
{
	//{{AFX_FIELD_MAP(Sections)
	pFX->SetFieldType(CFieldExchange::outputColumn);
	RFX_Text(pFX, _T("[CourseID]"), m_CourseID);
	RFX_Text(pFX, _T("[SectionNo]"), m_SectionNo);
	RFX_Text(pFX, _T("[InstructorID]"), m_InstructorID);
	RFX_Text(pFX, _T("[RoomNo]"), m_RoomNo);
	RFX_Text(pFX, _T("[Schedule]"), m_Schedule);
	RFX_Int(pFX, _T("[Capacity]"), m_Capacity);
	//}}AFX_FIELD_MAP
}
```

```
CMyprojDoc::CMyprojDoc()
{
	CDatabase StudDB;
	StudDB.OpenEx(_T("ODBC;DSN=Stdregistration"));
	Students1 StudentTable(&StudDB);
	StudentTable.Open(CRecordset::dynaset, "Students");

	//Add a new student
	StudentTable.AddNew();
	StudentTable.m_StudentID = 97999999;
	StudentTable.m_Name = "John Smith";
	StudentTable.m_GradYear = 1999;
	StudentTable.Update();

	//Cycle through sections and edit
	Sections SectionTable(&StudDB);
	SectionTable.Open();
	SectionTable.m_strFilter = "InstructorID = 'ROGERSN'";
	SectionTable.Requery();
	SectionTable.MoveFirst();

	while (!SectionTable.IsEOF()) {
		SectionTable.m_InstructorID = "SMITHJ";
		SectionTable.MoveNext();
	}
	SectionTable.Update();

	//Add a new table and attach it to query
	CString strSQL = "CREATE TABLE InstructorCourses (InstructorID
VARCHAR(15), CourseTitle VARCHAR(15) )";
	StudDB.ExecuteSQL(strSQL);

	CString strSQL = "INSERT INTO InstructorCourses (InstructorID,
CourseTitle)  SELECT InstructorID, CourseTitle FROM Section, Instructor
WHERE Section.InstructorID = Instructor.InstructorID";
	StudDB.ExecuteSQL(strSQL);

	InsCourses InsCourseTable(&StudDB);
	InsCourseTable.Open("InstructorCourses");

	StudDB.Close();
}
```