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 than 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)**
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);
    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();
}