

❑ 开放数据库互连 (Open DataBase Connectivity, ODBC)

- 用于应用程序与数据库服务通信的标准

- 标准定义了一个API

1. 建立一个和服务器的连接
2. 发送查询、更新请求等
3. 获取返回结果

❑ 应用程序（如，图形界面、电子表格等）可以使用相同的ODBC API来访问一个支持ODBC标准的数据库

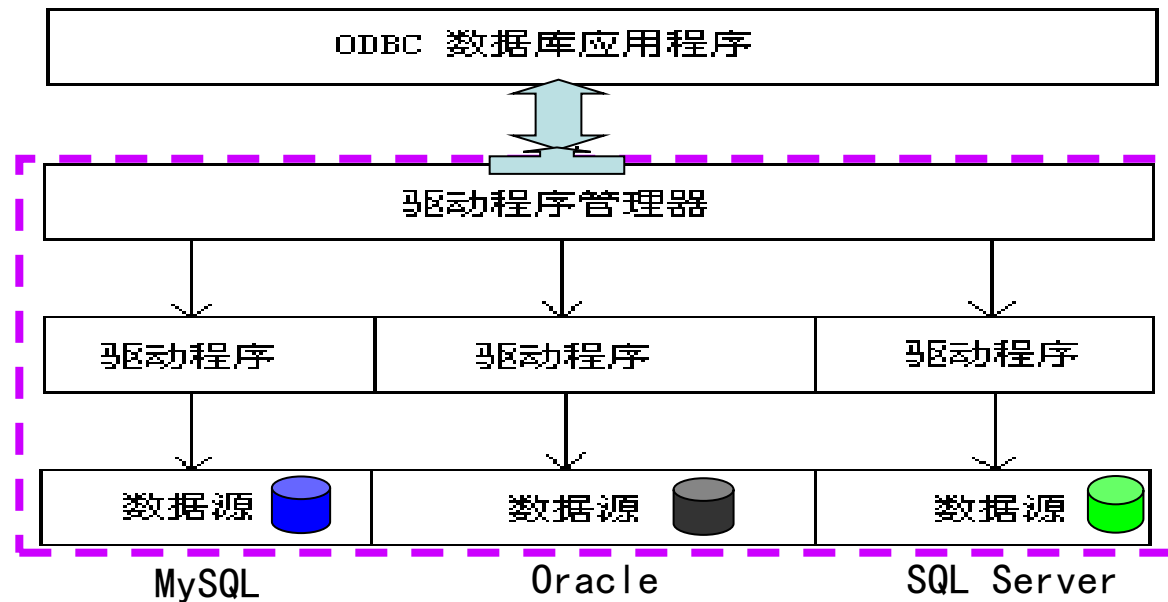
□ 嵌入式SQL和ODBC

- 嵌入式SQL：程序在编译前必须由一个特殊的预处理器进行处理
- ODBC标准为应用程序连接数据库服务器定义了一个API
 - 与具体DBMS无关
 - 不需要预编译

- ❑ ODBC提供了一个公共的、与具体数据库无关的应用程序设计接口API (Application Programming Interface)。所谓公共的接口API就是为开发者提供单一的编程接口，这样同一个应用程序就可以访问不同的数据库服务器
- ❑ 使用ODBC访问数据库的方法：
 - ODBC API访问数据库
 - Visual C++的MFC提供了丰富的ODBC类，它们封装了大量的函数用以完成数据库的大部分应用

□ 访问数据库的其他方法

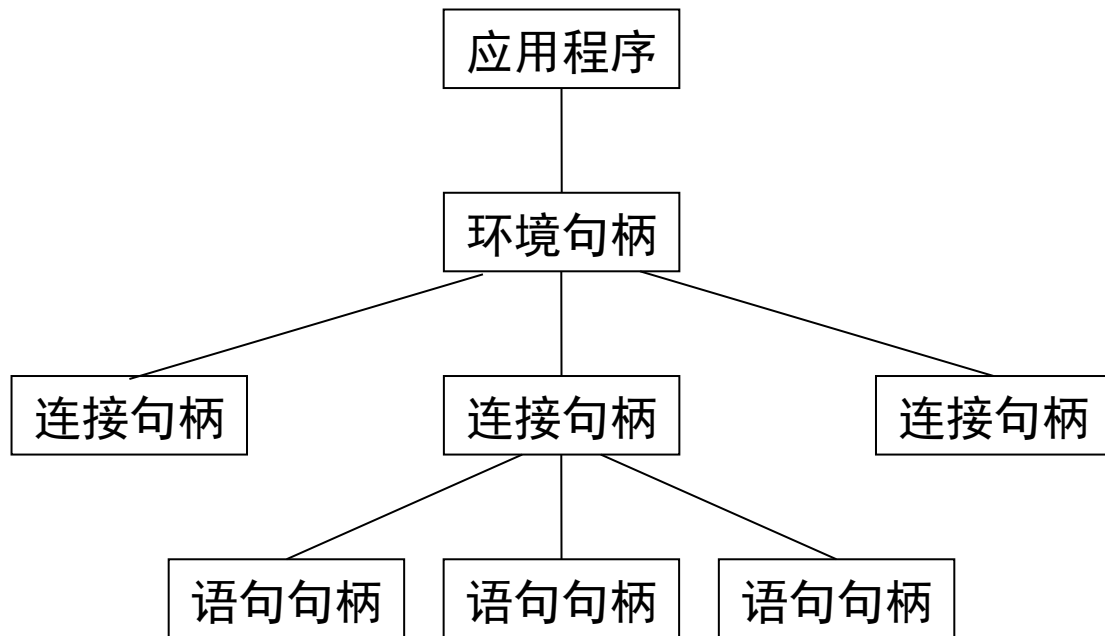
- OLE DB (Object Link and Embedding DataBase), 是一套通过COM (Component Object Model, 组件对象模型) 接口访问数据库的ActiveX底层接口技术, 速度快, 支持关系型和非关系型数据库, 编程量大
- ADO (ActiveX Data Objects), 基于COM, 建立在OLE DB之上, 更易于使用
- DAO (Data Access Objects), 基于Microsoft Jet引擎, 访问Access数据库 (即*.MDB文件) 时具有很好的性能, DAO类与ODBC类相比具有很多相似之处



- 定义ODBC数据源：控制面板—>管理工具—>
数据源(ODBC) —> DSN(data source name)

- ❑ ODBC程序会先分配一个SQL的环境变量，然后是一个数据库连接句柄
- ❑ ODBC程序随后会利用SQLConnect打开和数据库的连接，这个调用有几个参数，包括：
 - 数据库的连接句柄
 - 要连接的服务器
 - 用户的身份
 - 密码
- ❑ 还必须指定参数的类型
 - SQL_NTS表示前面参数是一个以null结尾的字符串

□ ODBC接口定义了三种句柄类型



❑ 1. 分配环境句柄

```
HENV henv;  
SQLAllocEnv (&henv );
```

❑ 2. 分配连接句柄

```
HDBC hdbc;  
SQLAllocConnect (henv, &hdbc);
```

❑ 3. 用已分配的连接句柄连接数据源

```
SQLConnect (hdbc, szDSN, cbDSN, szUID, cbUID, zAuthStr, cbAuthStr);
```

说明：hdbc是一个已分配的连接句柄

- szDSN和cbDSN分别表示系统所要连接的数据源名称字符串及其长度
- szUID和cbUID分别表示连接数据源的用户名字符串及其长度
- szAuthStr和cbAuthStr分别表示连接数据源的权限字符串及其长度

❑ 4. 分配语句句柄

```
HSTMT hstmt;  
SQLAllocStmt (hdbc, &hstmt);
```

❑ 5.1 直接执行SQL语句

```
SQLExecDirect (hstmt, szSqlStr, cbSqlStr );
```

说明：hstmt是一个有效的语句句柄

■ szSqlStr和cbSqlStr分别表示将要执行的SQL语句的字符串及其长度

■ 例，retcode=SQLExecDirect(hstmt, “delete from book where ISBN=1” , SQL_NTS);

— 说明：删除book表中ISBN=1的记录。SQL_NTS是ODBC的一个常数，当字符串是以NULL结束时，可用它来表示字符串的长度

□ 5.2 有准备地执行SQL语句

- 如果SQL语句需要执行几次，则采用有准备的执行更好，避免了SQL语句的多次分析。有准备的执行需要两个函数

`SQLPrepare (hstmt, szSqlStr, cbSqlStr);`

— 说明：SQL语句准备函数，参数同`SQLExecDirect`

`SQLExecute (hstmt);`

— 说明：SQL语句执行函数

❑ 6. 查询结果的获取

`SQLFetch (hstmt);`

说明：把游标移到下一行，当查询语句执行后第一次调用时移到结果集的第一行

`SQLGetData (hstmt, icol, fCType, rgbValue, cbValueMax, pcbValue);`

说明：读取游标指向行的列值

- `icol`和`fCType`分别表示结果集的列号和类型
- `rgbValue`和`cbValueMax`是接收数据存储区的指针和最大长度
- `pcbValue`是返回参数，表示本次调用后实际接收到的数据的字节数

□ 7. 释放语句句柄

`SQLfreeStmt(hstmt, foption);`

说明: `foption`指定选项, 一个选项是用`SQL_DROP`表示释放所有与该句柄相关的资源

□ 8. 断开数据源连接

`SQLDisconnect(hdbc);`

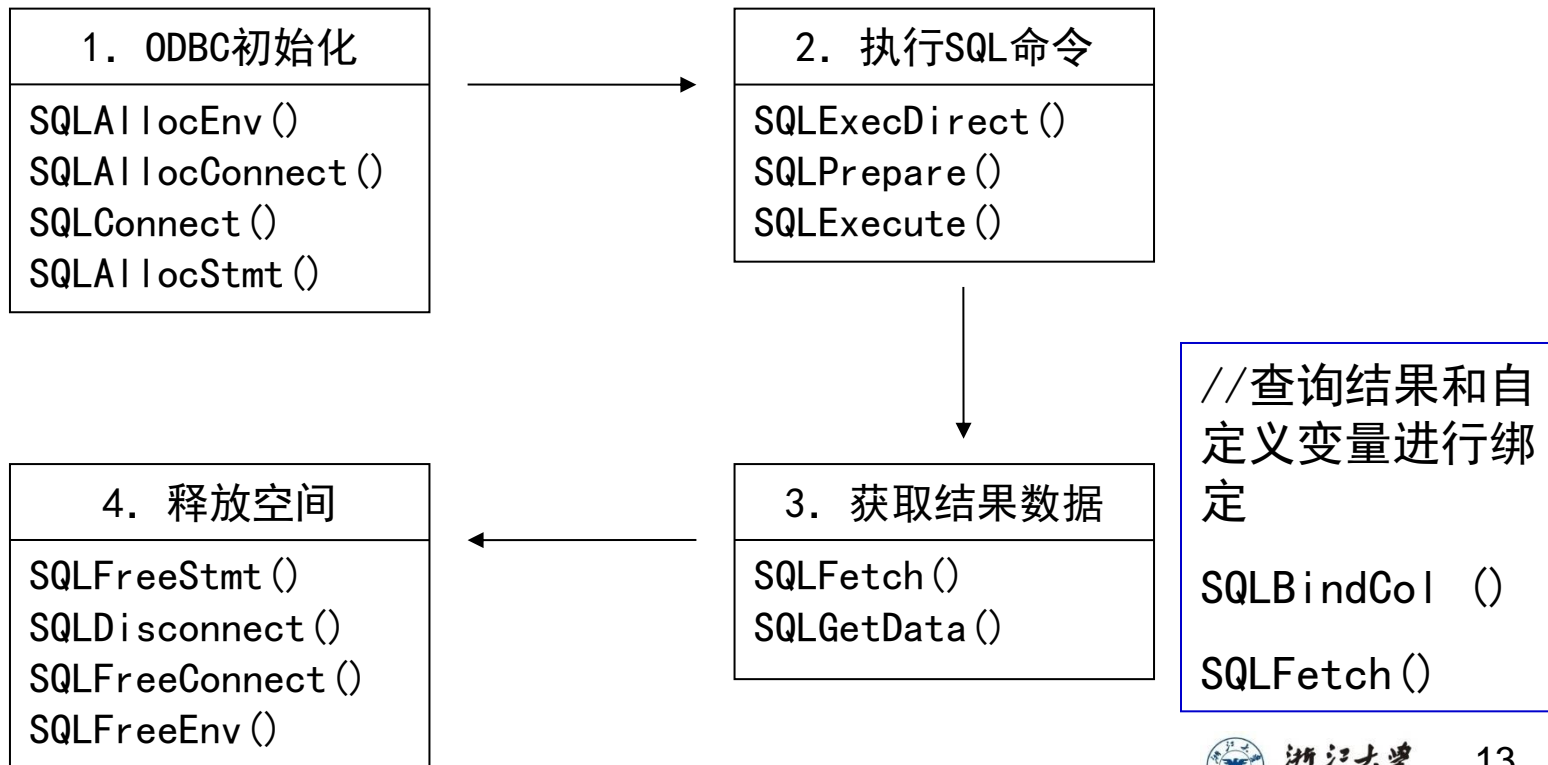
□ 9. 释放连接句柄

`SQLFreeConnect(hdbc);`

□ 10. 释放环境句柄

`SQLFreeEnv(henv);`

□ ODBC编程的基本流程



□ ODBC代码示例

```

int ODBCexample()          //程序结构
{
    RETCODE error;
    HENV      env;          /* environment */
    HDBC      conn;         /* database connection */
    SQLAllocEnv(&env);
    SQLAllocConnect(env, &conn); /*建立连接句柄 */
    SQLConnect(conn, "MySQLServer", SQL_NTS, "user",
    SQL_NTS, "password", SQL_NTS);
    /* 建立用户user与数据源的连接, SQL_NTS表示前一参量以null结尾 */
    { ... Main body of program ... } //见下页
    SQLDisconnect(conn);
    SQLFreeConnect(conn);
    SQLFreeEnv(env);
}

```

❑ ODBC代码示例 (Main body of program)

```
{  
    char deptname[80];  
    float salary;  
    int lenOut1, lenOut2;  
    HSTMT stmt;  
    SQLAllocStmt (conn, &stmt);  
    /*为该连接建立数据区, 将来存放查询结果*/  
    char * sqlquery = "select dept_name, sum (salary)  
                        from instructor  
                        group by dept_name" ; /*装配SQL语句*/  
    error = SQLExecDirect (stmt, sqlquery, SQL_NTS);  
    /*执行sql语句, 查询结果存放到数据区stmt , 同时sql语句执行状态的  
    返回值送变量error*/
```

❑ ODBC代码示例 (Main body of program)

```

.....
if (error == SQL_SUCCESS) {
    SQLBindCol(stmt, 1, SQL_C_CHAR, deptname, 80, &lenOut1);
    SQLBindCol(stmt, 2, SQL_C_FLOAT, &salary, 0, &lenOut2);
    /*对stmt中的返回结果数据加以分离，并与相应变量绑定。第1项数据转
    换为C的字符类型，送变量deptname(最大长度为80)，lenOut1 为实际字
    符串长度(若=-1代表null)，第2项数据转换为C的浮点类型送变量
    salary中 */
    while (SQLFetch (stmt) >= SQL_SUCCESS) {
        /*逐行从数据区stmt中取数据，放到绑定变量中*/
        printf ( " %s %f\n" , deptname, salary);
        /*对取出的数据进行处理*/
        .....
    }
    SQLFreeStmt (stmt, SQL_DROP); /* 释放数据区*/
}

```


- ❑ 应用程序通过SQLExecDirect语句把命令发送到数据库
- ❑ 使用SQLFetch语句取回查询结果中的元组
- ❑ SQLBindCol将C语言的变量和查询结果的属性绑定
 - 当结果中的元组被取出时，它的属性值会自动存储到对应的C语言变量里
 - SQLBindCol函数中的参数。例，

```
SQLBindCol(stmt, 1, SQL_C_CHAR, deptname, 80, &lenOut1);
```

- ODBC变量: stmt
- 选择属性中哪一个位置的值
- SQL把属性转化成什么类型的C变量
- 变量地址

- SQLBindCol函数中的参数。例，

```
SQLBindCol(stmt, 1, SQL_C_CHAR, deptname, 80, &lenOut1);
```

对于诸如字符数组这样的变长类型，最后两个参数还要给出：

- 变量的最大长度
- 存放元组取回时的实际长度
- 注意：如果长度域返回一个负值，那么代表这个值为空（null）

- 好的编程风格要求检查每一个函数的结果，确保它们没有错误，为了简洁，我们在这里忽略了大部分检查

- ❑ ODBC标准定义了**符合性级别**（**Conformance Levels**），用于指定标准定义的功能的子集（不同版本ODBC提供不同等级的标准）
 - 核心级（core level）
 - level 1需要支持取得目录的有关信息
 - level 2需要更多的特性，如发送和提取参数值数组以及检索有关目录的更详细信息的能力
- ❑ SQL标准定义了**调用级接口**（**Call Level Interface, CLI**），它与ODBC接口类似。如，Oracle call interface（OCI）

- ❑ SQL查询可以从宿主语言通过嵌入式SQL激发
- ❑ ODBC标准给C、JAVA等语言的应用程序定义接入SQL数据库的应用程序接口，程序员越来越多地通过这些API来访问数据库

谢谢！