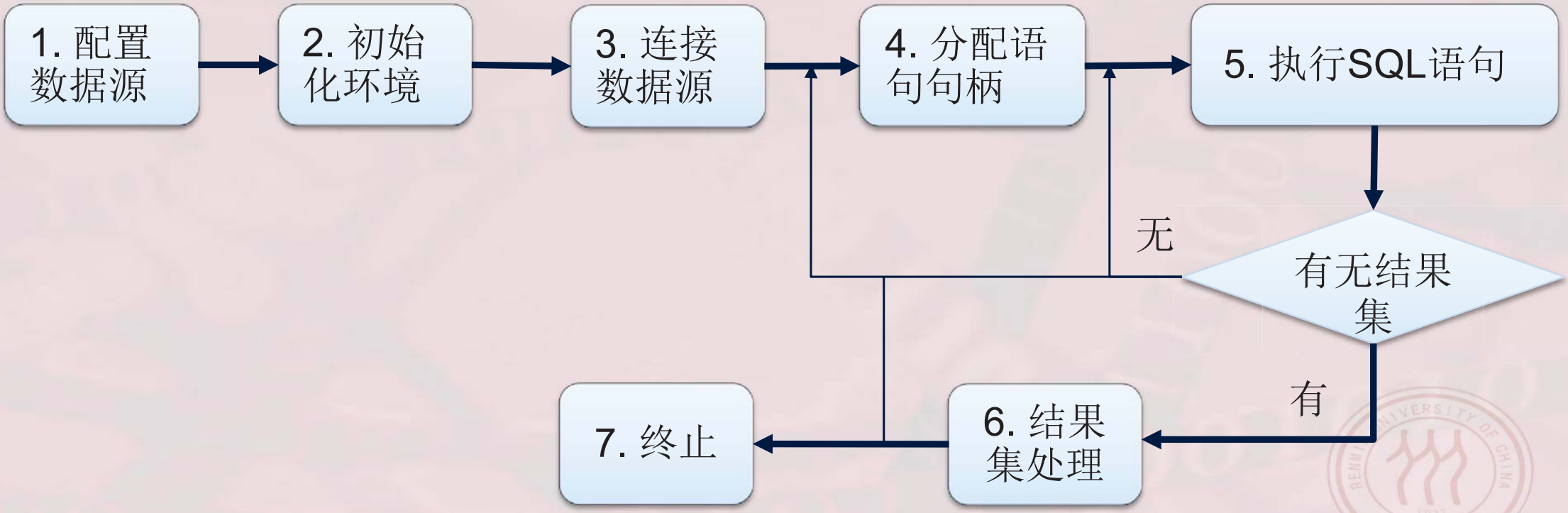
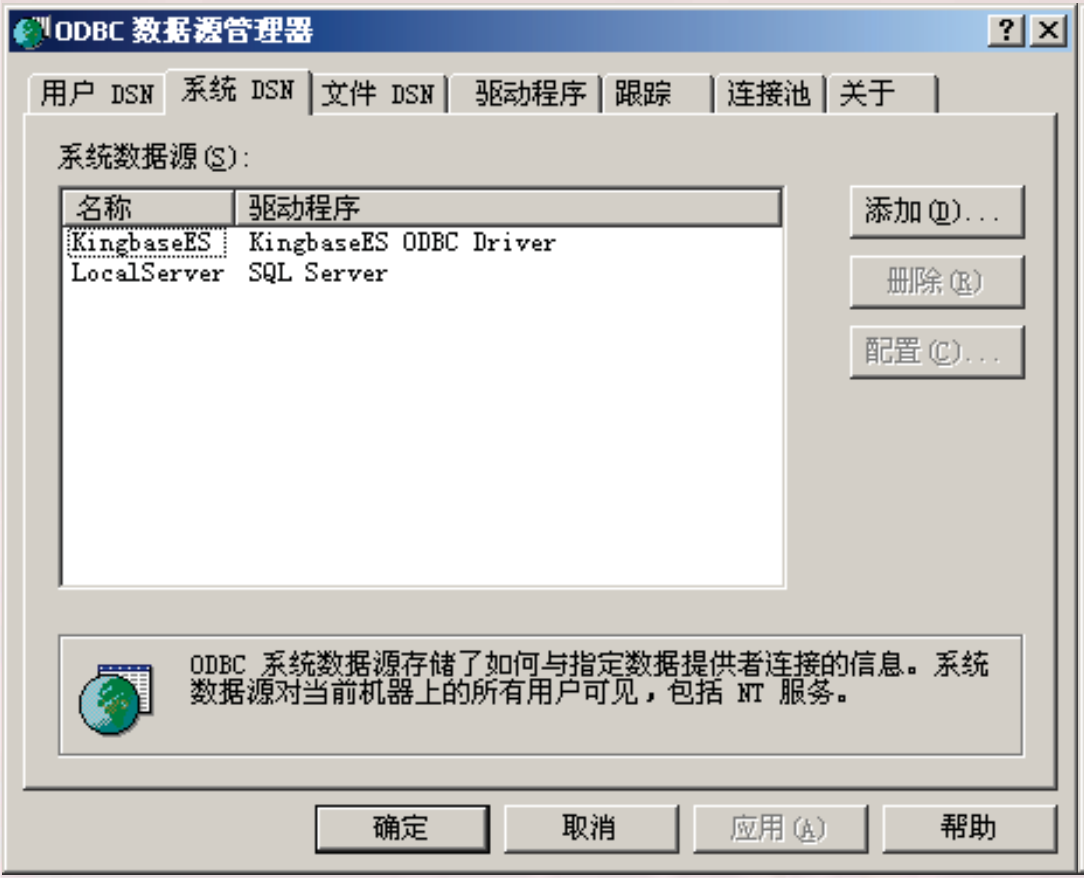
Video 13-3：ODBC 工作流程
（教科书 8.4.3）

8.4.3 ODBC的工作流程

❖ODBC的工作流程



1. 配置数据源


ODBC 数据源管理器
用户 DSN
系统 DSN
文件 DSN
驱动程序
跟踪
连接池
关于
系统数据源(S):
名称
驱动程序
KingbaseES
KingbaseES ODBC Driver
LocalServer
SQL Server
添加(D)...
删除(R)
配置(C)...
ODBC 系统数据源存储了如何与指定数据提供者连接的信息。系统数据源对当前机器上的所有用户可见，包括 NT 服务。
确定
取消
应用(A)
帮助

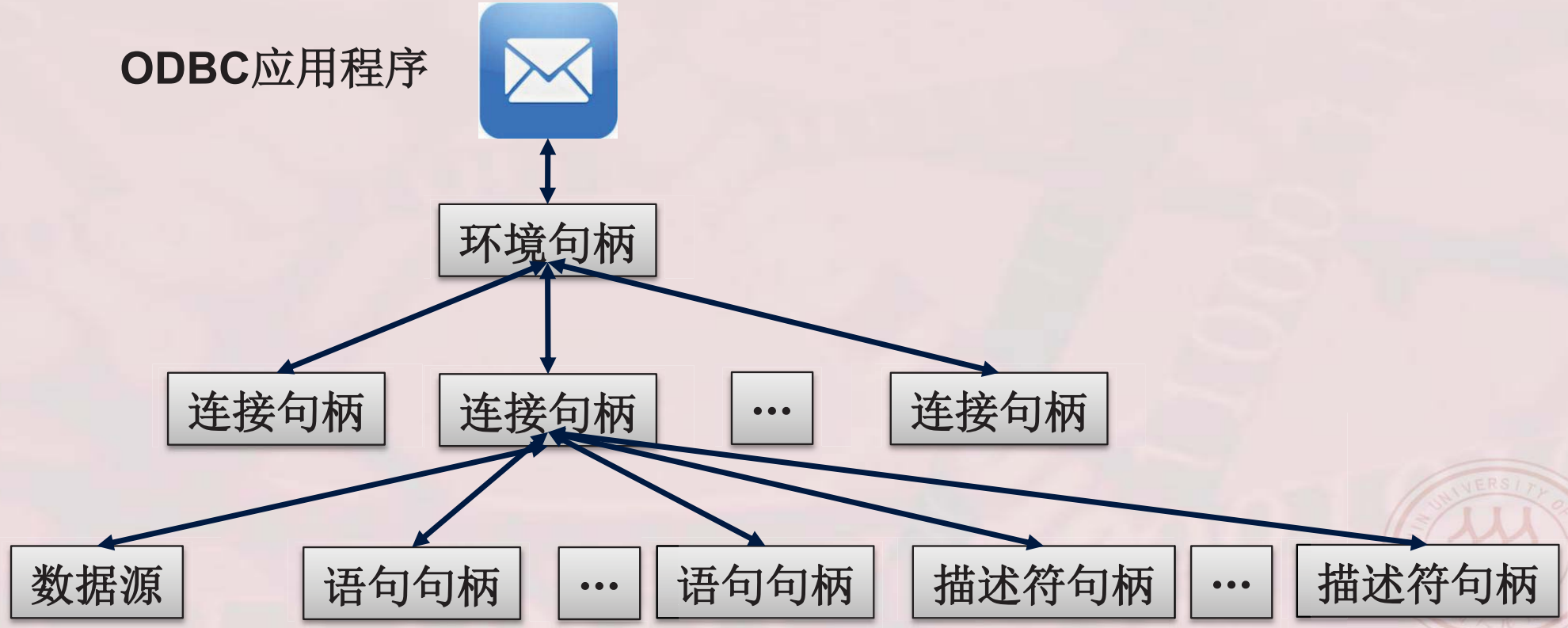

创建新数据源
选择您想为其安装数据源的驱动程序(S):
名称
Driver da Microsoft para arquivos texto (*.
Driver do Microsoft Access (*.mdb)
Driver do Microsoft dBase (*.dbf)
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
KingbaseES ODBC driver
Microsoft Access Driver (*.mdb)
Microsoft Access-Treiber (*.mdb)
Microsoft dBase Driver (*.dbf)
< 上一步(B)
完成
取消

1. 配置数据源


KingbaseES ODBC 数据源配置
数据源名称：
KingbaseES ODBC
描述：
KingbaseES-SQL Server
数据库名称：
TEST
服务器：
localhost
端口：
54321
用户名：
SYSTEM
口令：
****
保存
取消
测试连接
高级选项
确定
取消
应用(A)
帮助
选择您想为其安装数据源的驱动程序(S):
名称
Driver da Microsoft para arquivos texto (*.
Driver do Microsoft Access (*.mdb)
Success
连接成功!
确定
取消
< 上一步(B)
完成
取消

2. 初始化环境

❖环境句柄

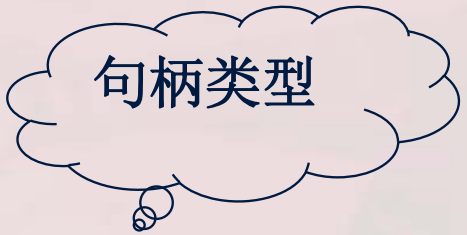


2. 初始化环境（续）

- 没有和具体的驱动程序相关联，由Driver Manager来进行控制，并配置环境属性
- 应用程序通过调用连接函数和某个数据源进行连接后，Driver Manager才调用所连的驱动程序中的SQLAllocHandle，来真正分配环境句柄的数据结构

2. 初始化环境（续）

- 环境句柄
  - 声明环境句柄变量
    - SQLHENV henv;
  - 创建环境句柄变量
    - SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
  - 初始化环境句柄变量
    - SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);



3. 建立连接

- 连接句柄
  - 声明连接句柄变量
    - SQLHDBC hdbc;
  - 创建连接句柄变量
    - SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
  - 初始化连接句柄变量：连接事先创建的数据源
    - SQLConnect(hdbc, “DNS-NAME",SQL_NTS, "SYSTEM",SQL_NTS, "MANAGER",SQL_NTS);

3. 建立连接

- 连接句柄
  - 声明连接句柄变量
    - SQLHDBC hdbc;
  - 创建连接句柄变量
    - SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
  - 初始化连接句柄变量：连接到指定的数据源
    - SQLConnect(hdbc, “DNS-NAME”,SQL_NTS, "SYSTEM",SQL_NTS, "MANAGER",SQL_NTS);

数据源名称

用户名

密码

4. 分配语句句柄

- 语句句柄
  - 声明语句句柄变量
    - SQLHSTMT hstmt;
  - 创建语句句柄变量
    - SQLAllocHandle(SQL_HANDLE_STMT, hdbc, & hstmt);
  - 初始化语句句柄变量：设置语句句柄的属性
    - SQLSetStmtAttr(hstmt,SQL_ATTR_ROW_BIND_TYPE,(SQLPOINTER)SQL_BIND_BY_COLUMN,SQL_IS_INTEGER);

5. 执行SQL语句

- 应用程序处理SQL语句的两种方式
  - 直接执行（SQLExecDirect适用于单条语句执行）
    - SQLExecDirect(hstmt,"SELECT * FROM STUDENT",SQL_NTS);
  - 预处理（SQLPrepare、SQLExecute适用于批量语句执行）
    - SQLPrepare(hstmt,“INSERT INTO STUDENT (SNO,SNAME,SSEX,SAGE,SDEPT) VALUES (‘20140091’,’刘晨’,’男’,20,’MA’)", SQL_NTS);
    - SQLPrepare(hstmt,“INSERT INTO STUDENT (SNO,SNAME,SSEX,SAGE,SDEPT) VALUES (‘20140092’,’王雪梅’,’女’,19,’CS’)", SQL_NTS);
    - SQLExecute(hstmt);

5. 执行SQL语句（续）

- 如果SQL语句含有参数，应用程序为每个参数调用SQLBindParameter，并把它们绑定至应用程序变量
  - SQLPrepare(hstmt,"INSERT INTO STUDENT (SNO,SNAME) VALUES (?, ?)", SQL_NTS);
  - SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,SQL_CHAR, SNO_LEN, 0, sSno, 0, &SQL_NTS);
  - SQLBindParameter(serverhstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,SQL_CHAR, NAME_LEN, 0, sName, 0, &SQL_NTS);
  - 设置sSno和sName的值
  - SQLExecute(hstmt);

SQLBindParameter

- SQLRETURN SQLBindParameter(
  SQLHSTMT StatementHandle,
  SQLUSMALLINT ParameterNumber,
  SQLSMALLINT InputOutputType,
  SQLSMALLINT ValueType,
  SQLSMALLINT ParameterType,
  SQLUINTEGER ColumnSize,
  SQLSMALLINT DecimalDigits,
  SQLPOINTER ParameterValuePtr,
  SQLINTEGER BufferLength,
  SQLINTEGER * StrLen_or_IndPtr);

SQLBindParameter（续）

- StatementHandle：执行SQL语句STMT句柄。
- ParameterNumber：指明要将变量与第几个参数绑定，从1开始计算。
- InputOutputType：指明是输入还是输出参数。可以取值的范围为：SQL_PARAM_INPUT，SQL_PARAM_OUTPUT，SQL_PARAM_INPUT_OUTPUT。
- ValueType：指明用于和参数绑定的C语言数据类型。
- ParameterType：指明SQL数据类型。
- ColumnSize：指明接收数据的宽度，对于字符串和结构需要指明数据的宽度，而对于普通的变量如SQLINTEGER，SQLFLOAT等设置为0就可以了。
- DecimalDigits ：当数据类型为SQL_NUMERIC，SQL_DECIMAL时指明数字小数点的精度，否则填0。
- ParameterValuePtr：在作为输入参数指明参数的指针，在作为输出参数时指明接收数据的变量指针。
- BufferLength：指明参数指针所指向的缓冲区的字节数大小。对于字符串和结构需要指明大小，而对于普通的变量如SQLINTEGER，SQLFLOAT等设置为0就可以了。
- StrLen_or_IndPtr：作为输入参数时指明数据的字节数大小，对于普通的定长变量如SQLINTEGER，SQLFLOAT等设置为0就可以了，对于字符号串需要在此参数中指定字符串数据的长度，或者设置为SQL_NULL_DATA表明此参数为空值，或者设置为SQL_NTS表明字符串以NULL字符结尾，对于结构需要指明结构的长度。当作为输出参数时，当SQL执行完毕后会在这个参数中返回拷贝的缓冲区的数据的字节数。

执行SQL语句（续）

- 应用程序根据语句类型进行的处理
  - 有结果集的语句（**select**或是编目函数），则进行结果集处理
  - 没有结果集的函数，可以直接利用本语句句柄继续执行新的语句或是获取行计数（本次执行所影响的行数）之后继续执行

6. 结果集处理

- 应用程序可以通过SQLNumResultCols来获取结果集中的列数
- 通过SQL DescribeCol或是SQLColAttrbute函数来获取结果集每一列的名称、数据类型、精度和范围

结果集处理（续）

- ODBC中使用游标来处理结果集数据
- ODBC中游标类型
  - Forward-only游标，是ODBC的默认游标类型
  - 可滚动（Scroll）游标
    - 静态（static）
    - 动态（dynamic）
    - 码集驱动（keyset-driven）
    - 混合型（mixed）

结果集处理（续）

- 结果集处理步骤
  - ODBC游标的打开方式不同于嵌入式SQL，不是显式声明而是系统自动产生一个游标，当结果集刚刚生成时，游标指向第一行数据之前
  - 应用程序通过SQLBindCol把查询结果绑定到应用程序缓冲区中，通过SQLFetch或是SQLFetchScroll来移动游标获取结果集中的每一行数据
  - 对于如图像这类特别的数据类型，当一个缓冲区不足以容纳所有的数据时，可以通过SQLGetdata分多次获取
  - 最后通过SQLClosecursor来关闭游标

结果集处理（续）

- 结果集处理
  - SQLBindCol把查询结果绑定到应用程序缓冲区中

```
SQLBindCol(hstmt, 1, SQL_C_CHAR,sSno,SNO_LEN, &SQLINTEGER);
SQLBindCol(hstmt, 2, SQL_C_CHAR, sName, NAME_LEN, &SQLINTEGER);
SQLBindCol(hstmt, 3, SQL_C_CHAR, sSex, SSEX_LEN, &SQLINTEGER);
SQLBindCol(hstmt, 4, SQL_C_LONG, &sAge, 0, &SQLINTEGER);
SQLBindCol(hstmt, 5, SQL_C_CHAR, sDepart, DEPART_LEN, &SQLINTEGER);
```

  - 通过SQLFetch移动游标获取结果集中的每一行数据

```
while ((ret=SQLFetch(hstmt)) !=SQL_NO_DATA_FOUND)
    process(sSno, sName, sSex, sAge, sDepart);
```

7. 终止处理

- 应用程序终止步骤
  - 释放语句句柄
  - 释放数据库连接
  - 与数据库服务器断开
  - 释放ODBC环境

终止处理（续）

- 创建数据源---第七步：终止处理

```
SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
SQLDisconnect(hdbc);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
SQLFreeHandle(SQL_HANDLE_ENV,serverhenv);
```