应用 ODBC 技术实现数据库系统的互连

高存宝 唐祯敏 康海生 （北方交通大学自动化所 100044）

摘要：本文介绍了 ODBC 开放式数据库互连技术的基本思想、原理和体系结构，并结合 C 语言和 Visual Basic 4 介绍了 ODBC 技术的应用。

关键词：ODBC SQL 驱动程序管理器 ODBC 驱动程序 数据源

一、ODBC 概述

目前，软件市场上有各种不同的数据库产品，它们在性能、价格和应用范围上各有千秋，而一个综合信息系统的各部门由于需求差异等原因，往往会存在多种数据库。这就带来了一个实际问题：如何实现这些数据库之间的互连访问？起初，各数据库厂商往往提供嵌入式 SQL (Structured Query)API。为了实现互连，用户一般要使用这些内嵌式 SQL 语句。这样，如果应用程序要移植到一个新的环境，其源码必须在新的环境下重新编译，因此可移植性比较差。为了进一步解决这一问题，Microsoft 推出了 ODBC(Open DataBase Connectivity)。ODBC 允许应用程序以 SQL 语言来存取 DBMS 管理的数据。它采用了一种新的途径：使用一个单独的程序来提取数据库信息，再提供一种方法让应用程序读取数据。ODBC 应用数据通信方法、数据传输协议、DBMS 等多种技术定义了一个标准的接口，使应用程序可以在各种应用和数据源之间传递数据。它引入了一个新的思想：数据库驱动程序，该驱动程序是一个动态链接库，就像在 Windows 下的打印机驱动程序一样。应用程序可以根据需要来选择一个数据源，而不必和 DBMS 绑在一起进行编译、连接、运行。

二、ODBC 的基本思想和体系结构

ODBC 的基本思想是为用户提供简单、标准、透明的数据库连接的公共编程接口，开发厂商根据 ODBC 的标准去实现底层的驱动程序，这个驱动程序对用户来说是透明的，并允许根据不同的 DBMS 采用不同的技术加以优化实现，这就利于不断吸收新的技术而趋完善。

ODBC 带来了数据库连接方式的变革。在传统方式中，开发人员需要熟悉多个 DBMS 及其 API，一旦 DBMS 端出现变动，则往往导致用户端系统重新编建或者源代码的修改，这就给开发和维护工作带来了很大困难。在 ODBC 方式中，不管底层网络环境如何，也无论采用何种 DBMS，用户在程序中都使用同一套标准代码，无需逐个了解各 DBMS 及其 API 的特点，源程序不会因底层的变化而重建或修改，使用户程序有很高的互操作性，从而减轻了开发维护的工作量，缩短了开发周期。

ODBC 是依靠分层结构来实现的，这样可保证其标准性和开放性。图 1 所示为 ODBC 的体系结构，它共分为四层：

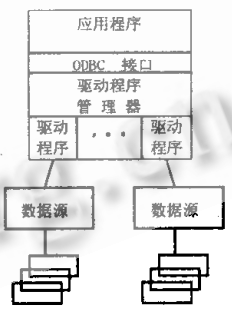

图 1 ODBC 体系结构

1. 应用程序

负责调用 ODBC 函数来提交 SQL 语句，提取结果。

使用 ODBC 接口的应用程序可执行以下任务：

(1)请求与数据源建立联接，创建一个对话；

(2)向数据源发出 SQL 请求；

(3)定义一个缓冲区和数据格式，用来存储 SQL 请求的结果；

(4)提取结果；

(5)处理各种错误；

(6)给用户报告结果;

(7)事务提交和事务撤销;

(8)中断与数据源的连接;

2.驱动程序管理器

驱动程序管理器是一个由 Microsoft 提供的带有一个入口函数库的动态链接库 ODBC.DLL,它的基本任务是加载驱动程序。此外还具有以下功能:

(1)根据 ODBC.INI 文件,把数据源名映射到相应的驱动程序 DLL;

(2)处理几个 ODBC 初始化函数;

(3)为 ODBC 调用提供参数合法性检查。

3.驱动程序

驱动程序是一个 DLL,用来完成 ODBC 函数调用并与数据源进行对话,并向数据源提交 SQL 请求,向应用程序返回结果,必要时驱动程序还将 SQL 语法翻译成符合 DBMS 语法规定的格式。根据应用程序的要求,驱动程序完成以下任务:

(1)建立与数据源的连接;

(2)向数据源提交请求;

(3)根据应用程序的需要,完成数据格式的转换;

(4)返回结果给应用程序;

(5)将运行错误格式化为标准代码返回给应用程序;

(6)根据数据源的需要,完成事务初始化(这一操作对应用程序是透明的)。

4.数据源

数据源由用户想要存取的数据、相应的 DBMS、DBMS 所在的系统平台及网络环境组成。每个数据源都需要驱动程序提供一定的信息,这些信息包括数据源名称、用户名及口令等。在安装数据源时将建立一个 ODBC.INI 文件,该文件中有一段文本专门用来列出可用的数据源,对每一个数据源都有一段文本对之进行描述,如定义驱动程序名、说明,以及一些驱动程序连接数据源时必要的信息。例如用于 OracleDemo 数据源的入口与下面的类似:

```
[ODBC Data Sources]
OracleDemo = Oracle7.1
[OracleDemo]
Drive = C: \ WINDOWS \ SYSTEM \ sqora71.dll
Description = Oracle Demo Data
Server = 2:
```

在应用程序向数据源发出 SQL 请求时,驱动程序管理器将根据此 ODBC.INI 文件找到数据源所对应的驱动程序并加载之,再由驱动程序来完成与数据源的连接、请求的提交以及结果的返回。

ODBC 接口定义了一个供应用程序调用的 ODBC API 函数库,利用这些函数,应用程序可以连接 DBMS,执行 SQL 语句,提取查询结果,而不必关心 ODBC 与 DBMS 之间的底层通信协议。此外,ODBC 接口还定义了遵循"X/Open and SQL Access Group (SAG) SQL CAE specification(1992)"标准的 SQL 语法。

为使 ODBC 具有最大的互操作性,从应用程序的角度看,最理想的情况是所有的数据源和驱动程序都支持同一套 ODBC 函数调用和 SQL 语句。但由于不同的 DBMS 在实现上有很大差异,它们所依赖的系统和环境也各不相同,所提供的 ODBC 函数和 SQL 语法也就不一致。为此 ODBC 定义了符合性级别,符合性级别建立了对众多功能的标准划分,定义了一个具体的驱动程序应支持哪些函数及 SQL 语句。如果一个驱动程序声明它支持某一个符合性级别,那么该驱动程序就应该支持该符合性级别中定义的全部功能。因此只要安装的驱动程序支持相同的符合性级别,就可以较为容易地实现相应数据库之间的互连访问,而在其顶层的应用程序的可移植性也能大大的提高。

三、ODBC 的基本应用

ODBC 接口定义的 ODBC API 函数均是以前缀"SQL"开头的一系列函数,对于应用程序开发者而言,由于只需调用这些 API 函数而不必关心底层的具体实现,所以开发一个 ODBC 应用程序就显得比较容易,下面以 C 语言为例简单介绍 ODBC 应用程序的基本开发步骤:

(1)安装和配置 ODBC 驱动程序。通常情况下,ODBC 驱动程序供应商会提供一个安装程序在你的系统上安装 ODBC 驱动程序。只要在 WINDOWS 环境下运行安装程序,再按提示一步步地去做,就可完成驱动程序的安装和相应数据源的设置工作,同时还会在前述的 ODBC.INI 文件中添加数据源的相关信息;

(2)调用函数 SQLAllocEnv 初始化 ODBC 环境,建立环境句柄;

(3)调用函数 SQLAllocConnect 分配连接句柄;

(4)调用函数 SQLConnect 连接数据源,并把数据源名称、用户名和口令等内容传给驱动程序。

当应用程序调用 SQLConnect 时,驱动程序管理器在文件 ODBC.INI 中用数据源名来查找对应的驱动程序 DLL,然后加载该 DLL,并把 SQL Connect 的参数传给

它。如果驱动程序还需要更多的信息，管理器则从 ODBC.INI 文件读取相应的内容；

(5)调用函数 SQL AllocStmt 为一个 SQL 语句分配 ODBC 语句句柄；

(6)发出 SQL 请求，提取和报告查询结果。一般是调用 SQL ExecDirect 函数或 SQL Execute 函数根据 ODBC 接口执行 SQL 语句。若 SQL 语句执行成功，应用程序则通过函数 SQL Fetch 提取结果；

(7)在完成对数据源的程序后，调用相应函数中断与数据源的连接。

以下是一个简单的例子(要使用 ODBC 数据，需包括头文件"SQL.H")。

本程序在 Acer Pentium 100 微机、Windows3.1、Visual C++ 1.5 环境下调试通过。

```
#define MAX-STMT-LEN 100
HENV henv;
HDBC hdbc;
HSTMT hstmt;
RETCODE retcode;
UCHAR SQL[MAX-STMT-LEN];
SDWORD Quantity;
SDWORD Total;
Total = 0;
retcode = SQLAllocEnv(&henv);
if(retcode = = SQL-SUCCESS){
retcode = SQLAllocConnect(henv, &hdbc);
if(retcode = = SQL-SUCCESS){
retcode = SQL Connect(hdbc, "OracledDemo", SQL-NTS,
        "Demo", SQL-NTS, "Demo", SQL-NTS);
    if(retcode = = SQL-SUCCESS||retcode = = SQL-SUCCESS-WITH-INFO){
        retcode = SQLAllocStmt(hdbc, &hstmt);
        if(retcode = = SQL-SUCCESS){
         Istrcpy(SQL, "select quantity from item");
         if(SQLExecDirect(hstmt, SQL, SQL-NTS) = = SQL-SUCCESS)
            {SQLBindCol(hstmt, 1, SQL-C-SLONG, &Quantity, 4, SQL-NULL-DATA);/*把缓冲区联结到数据列*/
            while((retcode = SQLFetch(hstmt)) = = SQL-SUCCESS)
                Total = Total + Quantity;
            if(retcode = = SQL-NO-DATA-FOUND)
                printf("Total = %ld/n", Total);}
        SQLFreeStmt(hstmt, SQL-DROP);
        }
    SQLDisconnect(hdbc);
    }
    SQLFreeConnect(hdbc);
    }
SQLFreeEnv(henv);
}
```

从以上例子可以看出，尽管调用 ODBC API 函数能为程序开发者编写 ODBC 应用程序提供一定的便利，但由于需要熟悉复杂、繁多且难用的 ODBC 数据，对一般的应用人员来说仍不是一件容易的事。其实在 Visual Basic 4 中，通过 Microsoft Jet 引擎来使用 ODBC 能够使应用程序开发人员动用熟悉的 DAO(数据访问对象)例程来操作数据库。

Microsoft Jet 引擎提供了一个介于应用程序和 ODBC 驱动程序管理器之间的抽象层。它与 ODBC 应用抽象组件之间的关系见图 2。

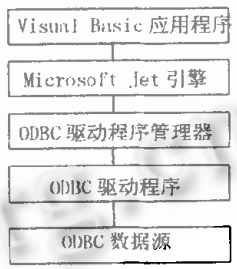

图 2 ODBC 的 Jet 层

上述 C 语言的程序通过 Microsoft Jet 引擎运用 VB4 DAO 对象实现的事件代码如下：

```
Private Sub cmdConnect-Click()
    Dim db As Database
    Dim rs As Recordset
    Dim Total As Integer
    Total = 0
  set db = OpenDatabase("", False, False, "ODBC;DSN = OracleDemo;
            UID = Demo;PWD = Demo")
  set rs = db.OpenRecordset("select quantity from item",
```

```
    dbOpensnapshot)
  Do Until rs.EOF
    Total = Total + rs.Fields("Quantity")
    rs.MoveNext
  Loop
  db.Close
  MsgBox "Total = / & Str $(Total)
End Sub
```

本程序在 Acer pentium 100 微机、Windows95 和 Visual Basic 4 环境下运行通过。

对比上述两程序可发现，使用 Microsoft Jet 引擎来编写 ODBC 例程确实快捷，但这种快捷有一定代价，即用 Jet 层书写的 ODBC 应用程序要比直接用 ODBC API 编写的应用程序慢一些。因此在实际应用时，若追求简单快捷，则可使用 VB4 中的 Microsoft Jet 引擎，若追求代码的高效率，则应直接使用 ODBC API 函数。

四、结束语

ODBC是一项很重要的技术，它为异构型数据库的互连提供了一个重要的思想：即只要安装不同的 ODBC 驱动程序就可存取相应的数据库产品，而不管用户使用何种前台软件，也不管后台是何种数据库，这个存取的过程是一致的。目前 ODBC 已为数据库供应商组织内部所认可，同时为众多应用软件厂商和第三方开发商所使用，微软也承诺进一步改进 ODBC 技术。相信随着 SQL 的推广和规范，用户和开发商会更加依赖于这一技术。

参考文献

[1] 文必龙 邵庆，开放数据库互连(ODBC)技术与应用，科学出版社 1997.2

[2] Brad Shannon, Visual Basic 4 Developer's Guide，机械工业出版社 19971.1

[3] 岳红宇 金以慧，全面了解 ODBC 技术，计算机世界 1995.12

(来稿时间：1997 年 5 月)