开放数据库互联(ODBC)
技术与应用

文必龙　邵　庆　编

科学出版社
龍門書局

开放数据库互联(ODBC)技术与应用

文必龙　邵　庆　编

燕卫华　　审订

科学出版社

龍門書局

1997

(京)新登字 092 号

内容简介

开放数据库互联(ODBC)为数据库应用程序访问异构型数据库提供了统一的数据存取 API,应用程序不必重新编译、连接就可以与不同的 DBMS 相联。目前支持 ODBC 的有 Oracle, Access, X-Base 等 10 多种流行的 DBMS。

本书介绍了 ODBC 的基本原理及 SQL 语言,并从两个角度介绍 ODBC 技术:一是从 ODBC 应用程序设计者的角度,将应用和开发中通常要用到的编程环节先总结成框架形式,然后分步骤讲解;另一是从 ODBC 驱动程序开发人员的角度,介绍如何开发 ODBC 驱动程序。本书还介绍了有关 ODBC 软件安装及数据源配置的方法。

本书适用于广大数据库应用人员和开发人员,也可供大、中专院校师生参考。

需要得到有关本书的技术支持或购买本书的读者,请与 010-62562329, 010-62541992 联系,或传真(010)62562329。

开放数据库互联(ODBC)技术与应用

文必龙 邵 庆 编
燕卫华

责任编辑:陆卫民

科学出版社
龙门书局 出版

北京东黄城根北街 16 号
邮政编码:100717

施园印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

1997 年 2 月第 一 版 开本:787×1092 1/16
1997 年 2 月第一次印刷 印张:9
印数:1—3000 字数:207000

ISBN7-03-005588-8/TP. 656

定价: 13.00 元

目　录

第一章　ODBC 基本原理

1.1　ODBC 简 介

异构型数据库之间的数据共享多年来一直是人们研究的课题，SQL(Structured Query Language)标准的制定为应用程序的移植带来一线希望，但各个 DBMS 定义出来的 SQL“方言”却在不同的 DBMS 之上的应用软件之间树起了一道隔墙。Microsoft 推出的 ODBC 解决了这些问题。ODBC(Open Database Connectivity，开放数据库互联)允许应用程序以 SQL 为数据存取标准，来存取 DBMS(Database Management Systems)管理的数据。ODBC 具有最大的互操作性，一个应用程序可以存取不同的数据库管理系统，而应用程序不必和 DBMS 绑在一起进行编译、连接、运行，而只要在应用程序中通过选择一个叫作数据库驱动程序的模块就可以把应用程序与所选的 DBMS 连接在一起了。

传统数据库管理应用程序经常用于完成工资管理、财务分析、辅助决策等。由于内嵌式 SQL 不受硬件环境和操作系统环境的限制，这些应用程序一般都使用内嵌式 SQL。但是，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。

当用户要对存储在数据库中的数据进行分析时，一般都愿意选择类似于 Microsoft Excel 电子表格这样的工具而不愿使用 DBMS 直接编写应用软件。按传统的数据存取方式，一个版本 Microsoft Excel 只能使用一个 DBMS 的预编译器进行编译连接，因而用户买一个产品只能使用一种 DBMS。ODBC 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，引入一个新的思想：数据库驱动程序(Database Drivers)，该驱动程序是一个动态链接库(Dynamic-Link Libraries，DLL)，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。ODBC 提供了一个标准接口，使应用程序可在各种应用和数据源之间传递数据。

ODBC 接口定义了如下内容：

(1) ODBC 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。

(2) SQL 语法。遵循标准“X/Open and SQL Access Group (SAG) SQL CAE specification (1992)”。

(3) 错误代码。

(4) 连接、登录 DBMS。

(5) 数据类型。

ODBC 接口具有相当的灵活性：

(1) 构成 SQL 语句的字符串可以在源程序中直接给出，也可以在运行时动态生成。

(2) 同一个程序可以存取不同的 DBMS。

(3) 应用程序不必关心 ODBC 与 DBMS 之间的底层通信协议。

(4) 数据值可以用应用程序最方便的格式传递。

ODBC 接口提供两种类型的函数调用:

(1) 核心函数(Core functions),基于标准"X/Open and SQL Access Group Call Level Interface specification"。

(2) 扩展函数,支持附加的函数,包括光标控制和异步进程。

向 ODBC 提交的 SQL 语句是以 ODBC 函数的参数形式给出的,SQL 语句不依赖于具体的 DBMS。在附录 B"SQL 语法"中,给出了基于"X/Open and SQL Access Group Call Level Interface specification"标准的 SQL 语法,应用程序应该只按这些语法使用 SQL,以保证最大的互操作性。

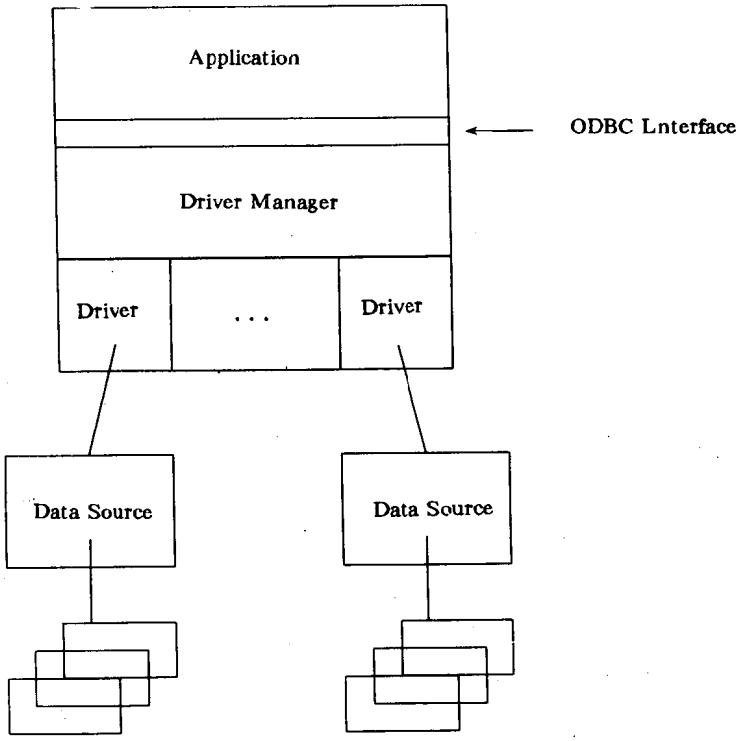


图 1.1 ODBC 驱动程序组件

1.2 ODBC 的组成部分

ODBC 包括四个组件:

(1) 应用程序(Application):负责调用 ODBC 函数来提交 SQL 语句,提取结果。

(2) 驱动程序管理器(Driver Manager):为应用程序加载驱动程序。

(3) 驱动程序(Driver):处理 ODBC 函数调用,向数据源提交 SQL 请求,向应用程序返回结果,必要时驱动程序将 SQL 语法翻译成符合 DBMS 语法规定的格式。

(4) 数据源(Data Source):由用户想要存取的数据、操作系统、DBMS、网络平台等组成。

对应用程序来说,在处理函数调用时,驱动程序和驱动程序管理器是一个整体。图 1.1 说明了这四个部分的关系。

应用程序用 ODBC API 完成以下任务:

(1) 请求与数据源建立联系,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) 事务提交或事务撤销。

(8) 中断与数据源的连接。

驱动程序管理器是一个 DLL,由 Microsoft 提供,是一个带有入口函数库的动态链接库,驱动程序管理器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) 参数合法性检验。

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。当应用程序调用函数 SQLBrowseConnect, SQLConnect, SQLDriverConnect 时,驱动程序管理器加载驱动程序。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,完成数据格式转换。

(4) 把结果返回给应用程序。

(5) 把错误整理成错误代码,并返回给应用程序。

(6) 申请并控制光标(这些操作对应用程序来说是透明的,除非应用程序发出了对光标名的存取请求)。

(7) 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)。

数据源是 DBMS、操作系统、网络平台的一个组合体。

1.3 ODBC 驱动程序的类型

ODBC 定义了两种驱动程序:

(1) 单层驱动程序:驱动程序既处理 ODBC 函数调用又处理 SQL 语句,相当于完成一部分数据源的任务。

(2) 多层驱动程序:驱动程序只处理 ODBC 函数调用,而把 SQL 语句交给数据源去处理。

一个系统可同时配有两种类型的驱动程序。

1.3.1 单层驱动程序的配置

对单层驱动程序,数据库文件是直接由驱动程序来处理的,驱动程序处理 SQL 语句并从数据库中提取结果,一个典型的单层驱动程序的例子是管理 XBASE 文件的驱动程序。

图 1.2 显示了两种单层驱动程序的配置。

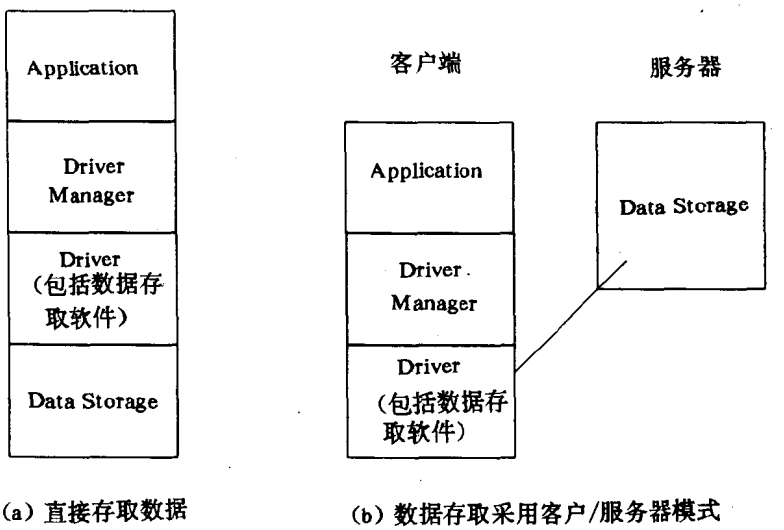


图 1.2 ODBC 驱动程序的单层结构

1.3.2 多层驱动程序的配置

在多层驱动程序的配置中,驱动程序把 SQL 请求传送到服务器中支处理。即使 ODBC 所有的组成部分都装在一个系统中,它们也被分成若干层而分布在系统里。应用程序,驱动程序,驱动程序管理器在一个系统中,叫作客户;数据库及控制数据存取的软件在另一个系统里,叫作服务器。

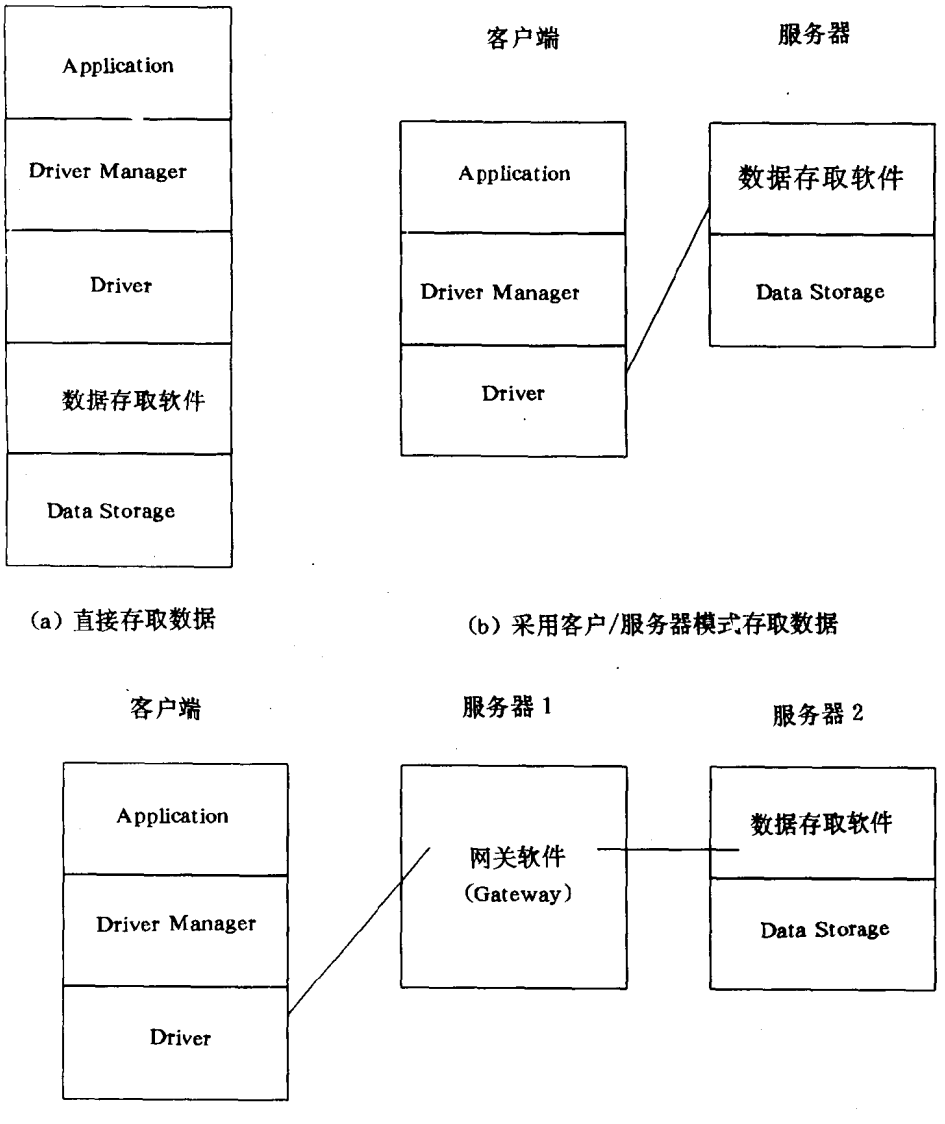
另外一种多层次配置是基于网关机制的,驱动程序把 SQL 请求送到网关进程,由网关进程把请求送到数据源。

此外,ODBC 还可在网络系统上进行配置,包括单一网络和广域网上两种情况。

1.4 ODBC 的符合性级别

ODBC 的优点是具有互操作性,在确定数据源之前程序员就可以编写应用程序,数据源的选择可以在程序运行后再确定。从应用程序的角度看,最理想的情况是所有的数据源都支持相同的函数调用和 SQL 语句。不幸的是,不同数据源及相关的驱动程序提供的函数和 SQL 语法有所不同,为此 ODBC 定义了符合性级别(conformance levels),符合性级别定义了一个具体的驱动程序应支持哪些函数及 SQL 语句。

图 1.3 给出了三种多层次配置,从应用程序的角度看,所有三种配置都是相同的。



(a) 直接存取数据

(b) 采用客户/服务器模式存取数据

(c) 通过网关存取数据

图 1.3 ODBC 驱动程序的多层次配置

ODBC 从两个方面来定义符合性级别:ODBC API 和 ODBC SQL 语法(包括 ODBC SQL 数据类型)。符合性级别通过建立一套标准的功能来帮助应用程序开发者和驱动程序开发者。应用程序就可很容易知道一个驱动程序是否支持它所需要的功能。

如果一个驱动程序声明它支持某一个 API 符合性级别或某一个 SQL 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,而不管这些功能是否被相应的 DBMS 支持。符合性级别并不限制驱动程序只能支持规定的功能,而是鼓励驱动程序开发者支持尽可能多的功能。应用程序可以通过调用函数 SQLGetInfo, SQLGetFunctions 和 SQLGetTypeInfo 来查询一个驱动程序到底支持哪些功能。

1.4.1 API 的符合性级别

对应于标准"X/Open and SQL Access Group Call Level Interface specification",ODBC API 定义了一套核心级函数,另外又定义了两个扩展级别:LEVEL1 和 LEVEL2。

大多数应用程序都要求驱动程序至少支持 LEVEL 1 API 符合性级别,为了保证 ODBC 驱动程序在应用程序中较好地完成工作,驱动程序开发者应实现 LEVEL1 中的所有功能。

核心级 API 具有如下功能:

(1) 分配和释放环境句柄、连接句柄、语句句柄。

(2) 连接数据源,在一个 连接中可以作用多个语句。

(3) 准备并 SQL 执行语句,并可立即 SQL 执行语句。

(4) 为 SQL 语句的参数和结果分配缓冲区。

(5) 提取结果及有关信息。

(6) 提交或撤销事务。

(7) 提取错误信息。

LEVEL 1 API 的功能:

(1) 具有核心级 API 的全部功能。

(2) 用驱动程序规定的对话框连接数据源。

(3) 传送一个参数值的部分或全部(这对长数据类型来说是很有用的)。

(4) 提取一个列值的部分或全部(这对长数据类型来说是很有用的)。

(5) 提取词典信息(列、统计、表等)。

(6) 提取驱动程序和数据源的兼容性信息,如支持的数据类型、数量函数、ODBC 函数等。

LEVEL 2 API 的功能:

(1) 具有核心级和 LEVEL 1 的功能。

(2) 浏览连接信息和可用的数据源清单。

(3) 传送若干组参数值,提取若干组列值。

(4) 提取参数个数及单个参数的描述信息。

(5) 使用可滚动光标。

(6) 提取 SQL 格式。

(7) 提取词典信息(权限、关键词、过程)。

(8) 调用翻译程序 DLL。

1.4.2 SQL 的符合性级别

ODBC 定义了一个大致相当于标准"X/Open and SQL Access Group SQL CAE specification (1992)"的核心级 SQL 语法,ODBC 还定义了最小级 SQL 以满足 ODBC 符合性级别的基本需要;ODBC 定义了扩充级 SQL。

最小级 SQL 语法:

(1) 数据定义语言(DDL):CREATE TABLE 和 DROP TABLE。

(2) 数据操纵语言(DML):简易 SQL,INSERT, UPDATE SEARCHED 和 DELETE SEARCHED。

(3) 表达式:简易型(例如 A>B+C)。

(4) 数据类型:CHAR, VARCHAR, LONG VARCHAR。

核心级 SQL 语法:

(1) 最小级 SQL 语法和数据类型。

(2) 数据定义语言(DDL):ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT 和 REVOKE。

(3) 数据操纵语言(DML):全部 SELECT。

(4) 表达式:子查询及一组函数(如 SUM 和 MIN)。

(5) 数据类型:DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION。

扩充 SQL 语法:

(1) 最小级和核心级 SQL 语法和数据类型。

(2)数据操纵语言(DML):外连接、定位 UPDATE、定位 DELETE、SELECT FOR UPDATE、联合操作。在 ODBC 1.0 中,定位 UPDATE、定位 DELETE、SELECT FOR UPDATE、联合操作是核心级的语法,在 ODBC 2.0 中定义到了扩充级。

(3) 表达式:数量函数(scalar functions),如(SUBSTRING、ABS、日期、时间等)。

(4) 数据类型:BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME, TIMESTAMP。

(5) 批量 SQL 语句。

(6) 过程调用。

1.4.3 怎样选择 ODBC 功能

ODBC 驱动程序支持的函数和 SQL 语句一般取决于数据源的兼容性,驱动程序开发应该实现尽可能多的函数。但对应用程序来说明在使用符合性级别内的功能外,使用哪些函数好呢?这取决于以下一些因素:

(1) 应用程序功能的需要。

(2) 应用程序性能的需要。

(3) 应用程序正在使用的数据源与其它的数据源具有互操作性。

(4) 驱动程序的能力。

因为不同的驱动程序支持不同级别的功能,应用程序开发者应对以上各因素作出适当的权衡。例如,一个应用程序需要显示一个表中的数据让用户进行查看或编辑,可以用函数 SQLColumnPrivileges 来查询哪一列可以修改,从而可以保护那些禁止修改的列,如果有的驱动程序不支持 SQLColumnPrivileges 函数,很显然,程序不能决定哪些列是禁止修改而应该保护起来。这时,程序员可以按如下方式去做:

(1) 使用所有的驱动程序,不保护任何一列。这样应用程序可以适合于所有的数据源,但减少了一些功能:用户可能修改那些他本没有修改权限的列。这样就会产生错误。

(2) 应用程序只使用支持 SQLColumnPrivileges 函数的驱动程序。

(3) 应用程序可使用所有的驱动程序。当某一驱动程序支持 SQLColumnPrivileges 时则保护那些禁止修改的列;当某一驱动程序支持 SQLColumnPrivileges 时,由于不能决定

哪些列是禁止修改的,只是给用户一个警告信息,他可能修改不了所有的列。

(4) 应用程序可使用所有的驱动程序,但总是保护那些禁止修改的列。当驱动程序不支持SQLColumnPrivileges 函数时,应用程序自己实现 SQLColumnPrivileges 的功能。

在做系统分析和系统计设时就选择好使用驱动程序的哪些功能,这对应用程序的性能优化及可移植性等有着很重要的作用。

1.4.4 连接与事务

应用程序在使用 ODBC 前,必须初始化 ODBC 并申请一个环境句柄(environment handle,henv),在与数据源连接前必须申请一个连接句柄(connection handle ,hdbc),在以后的 ODBC 函数调用中经常要指定环境句柄和连接句柄。

在一个应用程序中,为了连接多个数据源可能同时要申请多个连接句柄,每一个连接都占有一个独立的事务空间。

一个活动的连接可以有一个或多个语句。

驱动程序为每一个活动的连接管理一个事务。应用程序可以要求 SQL 语句执行完后自动提交事务,也可以要求在应用程序发出专门的事务请求(提交或撤销)后在处理事务。驱动程序在执行事务处理操作后,要对连接中的所有语句请求进行复位。

驱动程序管理器允许应用程序从当前连接中转到另一个连接。

第二章 SQL 简介

ODBC 是一个用于访问数据库系统的界面标准，受到 Oracle，Sybase，Informix 等十多种关系型数据库管理系统的支持，ODBC 之所以能够操纵如此众多的数据库，是因为当前绝大部分数据库都全部或部分地遵从 E. F. Codd 提出的关系数据库概念，甚至基于文件管理系统的产品都向关系模型靠拢，ODBC 正是抓住了这一共性。由于典型的应用程序真正用于访问数据库的基本函数不会很多(10 多个就够用)，加上 ODBC 采用结构化查询语言(SQL)，大大简化了 ODBC API，使 ODBC 用起来并不复杂。SQL 语言经历了 SQL-86、SQL-89、SQL-92，直到今后的 SQL3，表征着当今计算机技术的发展状况 SQL 作为关系代数语言，简单易学，是掌握使用关系型数据库管理系统和 ODBC 的基础。本章对 SQL 作一些简单介绍。

2.1 ANSI SQL 1989

SQL(Structured Query Language)，是一个被广泛采用的工业标准，该标准包括数据定义、数据操纵、数据管理、存取保护、事务控制等，SQL 源于关系型数据库，用表、索引、关键字、行、列来标识或定义数据。

SQL 本身并不是一个完整的程序设计语言(比如 SQL 不提供流程控制)，因而 SQL 总是与一个传统的程序设计语言(如 C 语言)联合使用。

SQL 的第一个标准是由美国国家标准委员会 ANSI 于 1986 年 10 月颁布的，即 X3.135—1986，紧接着，ISO 也作出了同样的决定，公布了 IS9075:1986 标准，这就是我们通常所说的 SQL-86。SQL-86 为软件制造商提供了一种极大的可能性，那就是无论在哪种机器平台上，还是何种数据库系统，都可采用 SQL 作为共同的数据存取或标准接口，该标准独立于任何程序设计语言。在 SQL-86 正式公布了名为 IS9075:1989 的“数据库语言 SQL”的修订标准，ANSI 也发表了同样的校订版标准，名为 X.3.135-1989，这个标准也叫 SQL-89，此后 ANSI 还发表了第二个标准 X3.169-1989。“嵌入式 SQL 语言”，而 ISO 则没有发表相关的标准，只是增加了一个关于嵌入式 SQL 定义的附录。SQL-89 在数据库发展发展史上起过相当的作用，许多数据库产品，如著名的 Oracle 等，都是因为采用了 SQL 标准而流行至今。当前的标准是 ANSI 1989，该标准为 SQL 定义了三个接口：

(1) 模块化语言(Module language)，允许在编译的程序内定义过程(即模块化)，这些过程被传统的程序设计语言的调用，并用参数返回值。

(2) 内嵌式 SQL 语言(Embedded SQL)，允许 SQL 语句被嵌入程序中，该标准为 COBOL，FORTRAN，Pascal，PL/1 都定义内嵌式 SQL 语句。

(3) 直接激活方式(Direct invocation)，存取方式由实现定义。

2.2 ANSI SQL 1992

内嵌式 SQL 允许程序员用标准的程序设计语言(称为宿主语言,如 COBOL 或 Pascal)把 SQL 语句写入程序,SQL 语句位于专门的开始语句和结束语句之间,开始语句和结束语句也是用宿主语言定义的,最终的程序包含 SQL 语言的代码和宿主语言的代码。

在编译嵌有 SQL 语句的程序时,预编译器先把 SQL 语句翻译成宿主语句的源代码,再用宿主语言的编译器进行编译。

在 ANSI 1989 中,内嵌式 SQL 是静态的。静态的 SQL 具有以下特征:

(1) 静态 SQL 语句是用程序源代码定义的,在编译前定义好列的个数及数据类型。

(2) 变量(指宿主语言中的变量 ,称为宿主变量)对宿主语言和 SQL 语言都是可存取的,但宿主变量不能用作列名或表名,宿主变量在编译前必须定义好长度和数据类型。

(3) 如果提交的 SQL 请求返回多行数据,可以定义光标来指定每次操作的是哪一行。

(4) 程序的每一次运行都执行同样的 SQL 请求,变化的只是宿主变量,所有的表名,列名对每次执行来说都是一样的,不能变化。如果要使用新的表名或列名,只能对程序源码进行修改,并重新编译。

(5) 使用标准的数据缓冲区来存储状态和错误信息。

静态 SQL 在完成一次预编译后,运行时都不再进行编译,因执行效率较高,但应用程序在编译时就与特定 DBMS 绑在一起了,不能动态地更换 DBMS。

静态 SQL 不能在运行时定义 SQL 语句,因此静态 SQL 不适合选作客户/服务器配置。

2.3 新一代的 SQL3

到 1992 年底,ANSI 的 X.3.135-1992、ISO 的 IS9075:1992 标准相断问世,成为当前的最新标准,这就是 SQL-92,SQL-92 是 SQL-89 的超集,主要是对语言表达式作了较大的扩充,SQL-92 定义了三个级别的功能:初始级(Entry)、中间级(Intermediate)、完全集(Full),更详细地说,SQL-92 在 SQL-86 和 SQL-89 基础上扩展了以下几个方面:

(1) 增加了许多新的数据类型,包括可变长的字符串类型、位串、复合字符集、日期、时间、内部值等。

(2) 数据库连接环境满足客户/服务器机制的要求。

(3) 支持动态 SQL。

(4) 可滚动光标(完全级)。

(5) 外连接(Outer joins,中间级和完全级)。

2.4 内 嵌 式 SQL

就在 SQL-92 还未正式公布之前,一个代号为 SQL3 的新规范已开始研制。当前的 SQL 语言虽然已经成为数据库的通用语言,但是随着计算机应用领域的不断扩展,关系模型表现出很多局限性。数据库技术中对象模型已深入人心,人们发现,用对象模型代替关系模型,将关系

"存储"到对象的属性和行为特性中去，不但能够更好地表达客观事物，完成数据的查询和操纵功能，保证数据的正交性，而且还可以实现数据的永久性定义，实现封装、继承、命名存储等机制。SQL3 仍是 SQL-92 的一个超集，到目前为止，SQL3 主要对 SQL 语言作了三方面的扩充，为对象相似提供了可能：

(1) 在程序设计方面的扩展。

到目前为止，SQL 语言不论在交互式下，抑或是嵌入到高级语言中，都是单条顺序执行的，这无疑带来了程序设计的不灵活性。SQL3 在这一方面所作的扩展是增加了大量的复合语句，因此确切地说，将来的 SQL2 仍是面向结构的语言，而不是面向对象的，但 SQL3 是支持对象管理的。SQL3 的复合语句包括 COMPOUND 语句、CASE 语句、IF 语句、LOOP 语句、FOR 语句以及 ASSIGNMENT、SIGNAL、RESIGNAL 等语句。

(2) 在类型系统方面的扩展。

SQL3 定义了抽象数据类型 ADT(Abstract Data Type)，可以说是"SQL 本身的一场革命"，SQL3 也因此将 SQL 语言从关系型数据库推向了对象数据库。SQL3 的类型系统包括以下五个方面：

① SQL3 通过使用 CREATE TYPE 来声明一个抽象数据类型 ADT。

② SQL3 引进了子类型和超类型的概念。

③ SQL3 中可以用一个已存在的 ADT 或生成类型通过 CREATE DISTINCTTYPE 语句来定义新的类型，即异类型。

④ SQL3 提供了"样板"功能。

⑤ SQL3 提供了集合类型 SET、MULTISET 和 LIST。

(3) SQL3 在查询语言方面的扩展。

查询是数据库的基本操作，SQL3 也将在这一方面作出相应的扩展，包括触发器规则、递归执行，此外还增加了一些新的谓词。

SQL3 本身还处于研制之中，将来的 SQL3 也许还会包含一些更新的技术。

2.5 动态 SQL

最新 ANSI 标准中定义的动态 SQL 允许应用程序在运行时产生和执行 SQL 语句。

动态 SQL 可以用一个函数调用进行准备，当准备好一个语句后，数据库环境产生一个可存取的计划和有关结果的描述，该语句可以按这个计划执行多次。

在动态 SQL 语句中可以含多个参数，参数的功能与内嵌式 SQL 相同，在执行前把值赋给合参数在缓冲区所处的位置即可，与静态 SQL 不同的是，动态 SQL 在编译前不必固定其长度和数据类型。

动态 SQL 的执行效率没有静态 SQL 高，但对用户来说更有用：

(1) 可在运行时灵活构筑 SQL 语句。

(2) 可延迟到运行时才与数据库连接。

2.6 SQL 调用级接口(CLI)

SQL 的调用级接口(CLI,Call Level Interface),由支持 SQL 语句的函数库组成,ODBC 接口就属于调用级接口。

CLI 通常用于数据的动态存取,X/Open and SQL Access Group 定义的 CLI 类似于在 X/Open and SQL Access Group SQL CAE specification(1992)规范中定义动态内嵌式 SQL 版本。

ODBC 接口可直接在应用程序中调用,而不必像内嵌式 SQL 那样进行预处理,函数调用也不需要宿主变量和其他内嵌式 SQL 概念,这对熟悉函数库的程序员来说用起来更自然,直接截了当。

CLI 不需要进行预编译,如要提交 SQL 请求,只需要把 SQL 命令以字符串的形式存入一个文本缓冲,然后把该缓冲的地址作为参数传给函数。对错误信息的查询是通过函数调用的返回代码或专门的错误处理函数来进行的。

CLI 可以在结果可用之前或可用之后定义存储结果的缓冲区,称为变量连接(Binding)。

CLI 的互操作性表现在:

(1) 所有的客户和数据源都遵守同一标准接口。

(2) 所有的客户都遵守相同的标准接口,驱动程序负责为特定的数据源解释各个命令。

第二个特性允许驱动程序掩盖了 ODBC 与数据库之间在功能、数据库协议、网络等方面的差异,由于 ODBC 具有第二个特性,所以 ODBC 可以充分发挥标准数据库协议和网络协议的优越性。