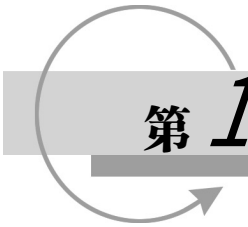
第一部分　MySQL 综述



第 1 章　数据库基础

为了更好地学习 MySQL，首先需要介绍数据库的基本概念。如果学习过数据库原理，那么本章数据库基础部分仅仅作为一个参考。

1.1　数据库基本概念

1.1.1　数据库及其系统

1. 数据库

数据库（DB）是存放数据的仓库，而且这些数据存在一定的关联，并按一定的格式存放在计算机内。例如，把一个学校的学生、课程、成绩等数据有序地组织并存放在计算机内，就可以构成一个数据库。

2. 数据库管理系统

数据库管理系统（DBMS）按一定的数据模型组织数据形成数据库，并对数据库进行管理。简单地说，DBMS 就是管理数据库的系统（软件）。数据库管理员（DBA）通过 DBMS 对数据库进行管理。

3. 数据库系统

数据、数据库、数据库管理系统与操作数据库的应用程序，加上支撑它们的硬件平台、软件平台和与数据库有关的人员一起构成了一个完整的数据库系统。如图 1.1 所示描述了数据库系统的构成。

1.1.2　数据模型

数据库管理系统组织数据库数据采用的数据模型主要有层次模型、网状模型和关系模型。

1. 层次模型

层次模型将数据组织成一对多关系的结构，采用关键字来访问其中每一层次的每一部分。它存取方便且速度快；结构清晰，容易理解；数据修改和数据库扩展容易实现；检索关键属性十分方便。但结构不够灵活；同一属性数据要存储多次，数据冗余大；不适合拓扑空间数据的组织。

如图 1.2 所示为按层次模型组织的数据示例。

2. 网状模型

网状模型是具有多对多类型的数据组织方式。它能明确而方便地表示数据间的复杂关系；数据

冗余小。但网状结构的复杂，增加了用户查询和定位的困难；需要存储数据间联系的指针，使得数据量增大；数据的修改不方便。

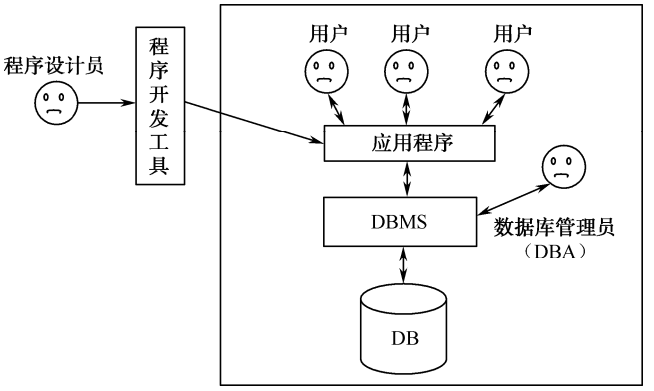


图 1.1　数据库系统的构成

如图 1.3 所示为按网状模型组织的数据示例。

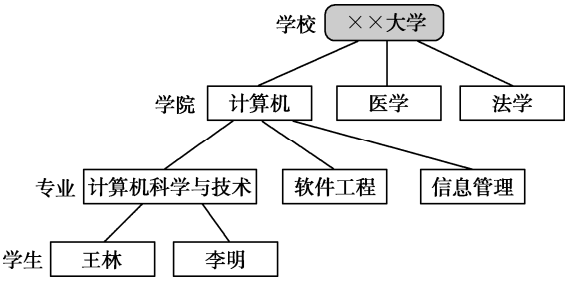


图 1.2　按层次模型组织的数据示例

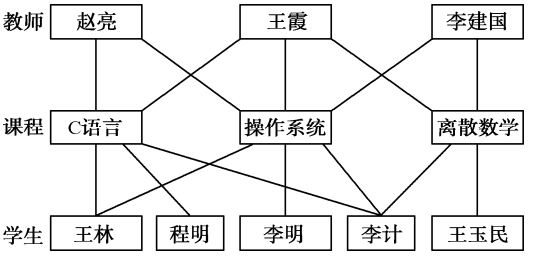


图 1.3　按网状模型组织的数据示例

3. 关系模型

关系模型以记录组或二维数据表的形式组织数据。例如，在学生成绩管理系统所涉及的“学生”“课程”和“成绩”3 个表中，“学生”表涉及的主要信息有学号、姓名、性别、出生时间、专业、总学分和备注；“课程”表涉及的主要信息有课程号、课程名、开课学期、学时和学分；“成绩”表涉及的主要信息有学号、课程号和成绩。如表 1.1、表 1.2 和表 1.3 所示分别描述了学生成绩管理系统中“学生”“课程”和“成绩”3 个表的部分数据。

表 1.1　“学生”表

学　号	姓　名	性　别	出生时间	专　业	总学分	备　注
081101	王林	男	1994-02-10	计算机	50	
081103	王燕	女	1993-10-06	计算机	50	
081108	林一帆	男	1993-08-05	计算机	52	已提前修完一门课程
081202	王林	男	1993-01-29	通信工程	40	有一门课程不及格，待补考
081204	马琳琳	女	1993-02-10	通信工程	42	

表 1.2　“课程”表

课 程 号	课 程 名	开 课 学 期	学　时	学　分
0101	计算机基础	1	80	5
0102	程序设计与语言	2	68	4
0206	离散数学	4	68	4

表 1.3　“成绩”表

学　号	课 程 号	成　绩	学　号	课 程 号	成　绩
081101	101	80	081108	101	85
081101	102	78	081108	102	64
081101	206	76	081108	206	87
081103	101	62	081202	101	65
081103	102	70	081204	101	91

表格中的一行称为一个记录，一列称为一个字段，每列的标题称为字段名。如果给每个关系表起一个名字，则有 *n* 个字段的关系表的结构可表示为如下形式：

关系表名(字段名 1,…,字段名 *n*)

通常把关系表的结构称为关系模式。

在关系表中，如果一个字段或几个字段组合的值可唯一标识其对应的记录，则称该字段或字段组合为码。

例如，表 1.1 中的“学号”可唯一标识每一个学生，表 1.2 中的“课程号”可唯一标识每一门课程，表 1.3 中的“学号”和“课程号”可唯一标识每一个学生每一门课程的成绩。

有时，一个表可能有多个码，比如表 1.1 中，如果姓名不重名，则“学号”“姓名”均是学生信息表的码。对于每一个关系表，通常可指定一个码为主码，在关系模式中，一般用下横线标出主码。

设表 1.1 的名字为 xs，关系模式可表示为如下形式：

xs(学号, 姓名, 性别, 出生时间, 专业, 总学分, 备注)

设表 1.2 的名字为 kc，关系模式可表示为如下形式：

kc(课程号, 课程名, 开课学期, 学时, 学分)

设表 1.3 的名字为 xs_kc，关系模式可表示为如下形式：

xs_kc(学号, 课程号, 成绩)

通过上面的分析可以看出，关系模型更适合组织数据，所以使用广泛。目前，主流的关系型数据库管理系统（RDBMS）包括 Oracle、SQL Server、DB2、Sybase、MySQL 等。MySQL 是目前流行的开放关系数据库管理系统之一。

1.1.3　关系型数据库语言

结构化查询语言（SQL）是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统，同时也是数据库脚本文件的扩展名。

结构化查询语言是高级的非过程化编程语言，允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同数据库系统可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言的语句可以嵌套，这使它具有极大的灵活性和强大的功能。

1986 年 10 月，美国国家标准协会对 SQL 进行规范后，以此作为关系型数据库管理系统的标准语言（ANSI X3. 135-1986），1987 年得到国际标准组织的支持并成为国际标准。不过各种通行的数据库系统在其实践过程中都对 SQL 规范进行了某些改写和扩充。所以，不同数据库系统之间的 SQL 实际上不能完全相互通用。

1.2 数据库设计

数据模型按不同的应用层次分为 3 种类型：概念模型、逻辑模型、物理模型。

1.2.1 概念模型

概念模型是面向数据库用户的现实世界的模型，主要用来描述世界的概念化结构，它使数据库的设计人员在设计的初始阶段摆脱计算机系统及 DBMS 的具体技术问题，集中精力分析数据以及数据之间的联系等，与具体的 DBMS 无关。概念模型必须换成逻辑模型后，才能在 DBMS 中实现。

概念模型用于信息世界的建模，常用的是 E-R 模型、扩充的 E-R 模型、面向对象模型及谓词模型。

通常，E-R 模型把每一类数据对象的个体称为“实体”，把每一类对象个体的集合称为“实体集”。例如，在学生成绩管理系统中主要涉及“学生”和“课程”两个实体集，其他非主要的实体可以很多，如班级、班长、任课教师、辅导员等实体。

把每个实体集涉及的信息项称为属性。就“学生”实体集而言，它的属性有学号、姓名、性别、出生时间、专业、总学分和备注。“课程”实体集属性有课程号、课程名、开课学期、学时和学分。

实体集中的实体彼此是可区别的。如果实体集中的属性或最小属性组合的值能唯一标识其对应实体，则将该属性或属性组合称为码。码可能有多个，对于每一个实体集，可指定一个码为主码。

实体集 *A* 和实体集 *B* 之间存在各种关系，通常把这些关系称为“联系”，将实体集及实体集联系的图示称为实体（Entity）-联系（Relationship）模型。

E-R 图就是 E-R 模型的描述方法，即实体-联系图。通常，关系数据库的设计者使用 E-R 图来对信息世界建模。在 E-R 图中，使用矩形表示实体型，使用椭圆表示属性，使用菱形表示联系，用线段连接实体集与属性，当一个属性或属性组合指定为主码时，在实体集与属性的连接线上标记一条斜线。如图 1.4 至图 1.7 所示，其中属性用圆角矩形表示。从分析用户项目涉及的数据对象及数据对象之间的联系出发，到获取 E-R 图的这一过程称为概念结构设计。

两个实体集 *A* 和 *B* 之间的联系可能是以下 3 种情况之一。

1. 一对一的联系（1：1）

A 中的一个实体最多与 *B* 中的一个实体相联系，*B* 中的一个实体也最多与 *A* 中的一个实体相联系。例如，“班级”与“班长”这两个实体集之间的联系是一对一的联系，因为一个班级只有一个班长，反过来，一个班长只属于一个班级。“班级”与“班长”两个实体集的 E-R 模型如图 1.5 所示。

2. 一对多的联系（1：*n*）

A 中的一个实体可以与 *B* 中的多个实体相联系，而 *B* 中的一个实体最多与 *A* 中的一个实体相联系。例如，“班级”与“学生”这两个实体集之间的联系是一对多的联系，因为一个班级可有若干学生，反过来，一个学生只能属于一个班级。“班级”与“学生”两个实体集的 E-R 模型如图 1.6 所示。

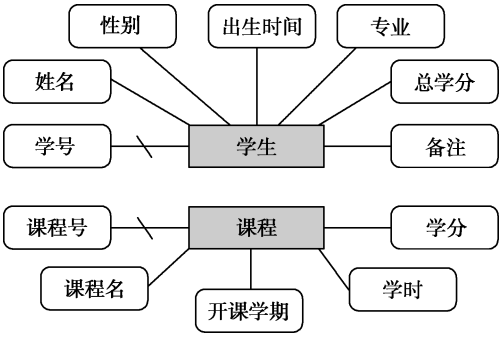


图 1.4 “学生”和“课程”实体集属性的描述

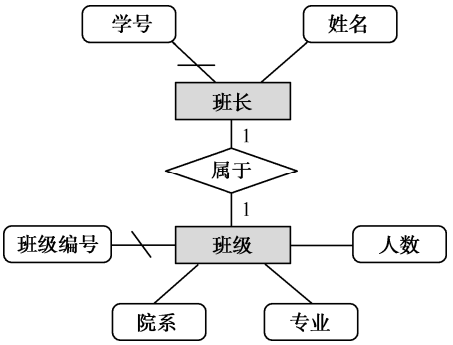


图 1.5 “班级”与“班长”两个实体集的 E-R 模型

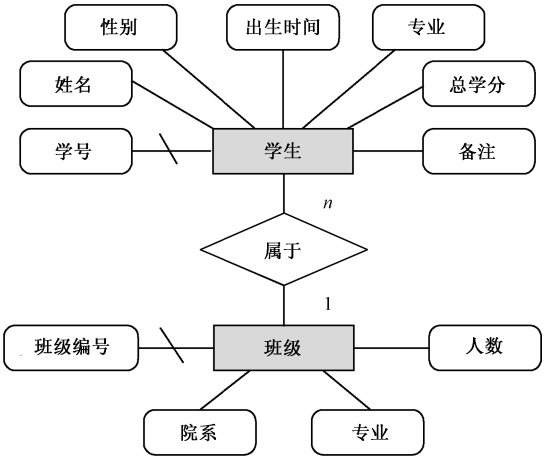


图 1.6 “班级”与“学生”两个实体集的 E-R 模型

3. 多对多的联系（m：n）

A 中的一个实体可以与 B 中的多个实体相联系，而 B 中的一个实体也可与 A 中的多个实体相联系。例如，“学生”与“课程”这两个实体集之间的联系是多对多的联系，因为一个学生可选修多门课程，反过来，一门课程可被多个学生选修。“学生”与“课程”两个实体集的 E-R 模型如图 1.7 所示。

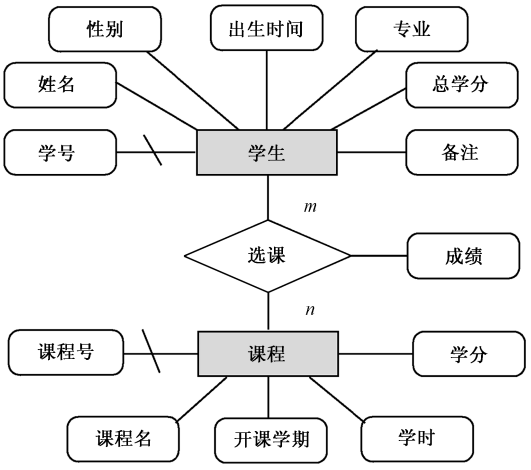


图 1.7 “学生”与“课程”两个实体集的 E-R 模型

1.2.2 逻辑模型

逻辑模型是用户从数据库所看到的模型，是具体的 DBMS 所支持的数据模型。此模型既要面向用户，又要面向系统，主要用于 DBMS 的实现。

前面用 E-R 图描述学生成绩管理系统中实体集与实体集之间的联系，为了设计关系型的学生成绩管理数据库，需要确定包含哪些表，每个表的结构是怎样的。下面将给出 3 种联系从 E-R 图获得关系模式的方法。

1.（1∶1）联系的 E-R 图到关系模式的转换

对于（1∶1）的联系，既可以单独对应一个关系模式，也可以不单独对应一个关系模式。

（1）联系单独对应一个关系模式，则由联系的属性、参与联系的各实体集的主码属性构成关系模式，其主码可选参与联系的实体集的任一方的主码。

例如，考虑图 1.5 描述的“班级（bj）”与“班长（bz）”实体集通过属于（sy）联系 E-R 模型，可设计如下关系模式（下横线表示该字段为主码）：

bj（班级编号，院系，专业，人数）

bz（学号，姓名）

sy（学号，班级编号）

（2）联系不单独对应一个关系模式，联系的属性及一方的主码加入另一方实体集对应的关系模式中。

例如，考虑图 1.5 描述的“班级（bj）”与“班长（bz）”实体集通过属于（sy）联系 E-R 模型，可设计如下关系模式：

bj（班级编号，院系，专业，人数）

bz（学号，姓名，班级编号）

或者：

bj（班级编号，院系，专业，人数，学号）

bz（学号，姓名）

2.（1∶*n*）联系的 E-R 图到关系模式的转换

对于（1∶*n*）的联系，既可以单独对应一个关系模式，也可以不单独对应一个关系模式。

（1）联系单独对应一个关系模式，则由联系的属性、参与联系的各实体集的主码属性构成关系模式，*n* 端的主码作为该关系模式的主码。

例如，考虑图 1.6 描述的“班级（bj）”与“学生（xs）”实体集 E-R 模型，可设计如下关系模式：

bj（班级编号，院系，专业，人数）

xs（学号，姓名，性别，出生时间，专业，总学分，备注）

sy（学号，班级编号）

（2）联系不单独对应一个关系模式，则将联系的属性及 1 端的主码加入 *n* 端实体集对应的关系模式中，主码仍为 *n* 端的主码。

例如，图 1.6 描述的“班级（bj）”与“学生（xs）”实体集 E-R 模型可设计如下关系模式：

bj（班级编号，院系，专业，人数）

xs（学号，姓名，性别，出生时间，专业，总学分，备注，班级编号）

3.（*m*∶*n*）联系的 E-R 图到关系模式的转换

对于（*m*∶*n*）的联系，单独对应一个关系模式，该关系模式包括联系的属性、参与联系的各实体集的主码属性，该关系模式的主码由各实体集的主码属性共同组成。

例如，图 1.7 描述的“学生（xs）”与“课程（kc）”实体集之间的联系可设计如下关系模式：

xs（学号，姓名，性别，出生时间，专业，总学分，备注）

kc（课程号，课程名，开课学期，学时，学分）

xs_kc（学号，课程号，成绩）

关系模式 xs_kc 的主码是由“学号”和“课程号”两个属性组合起来构成的一个主码，一个关系模式只能有一个主码。

至此，已介绍了根据 E-R 模型设计关系模式的方法。通常，这一设计过程称为逻辑结构设计。

在设计好一个项目的关系模式后，就可以在数据库管理系统环境下创建数据库、关系表及其他数据库对象，输入相应数据，并根据需要对数据库中的数据进行各种操作。

1.2.3　物理模型

物理模型是面向计算机物理表示的模型，描述了数据在储存介质上的组织结构，它不但与具体的 DBMS 有关，而且与操作系统和硬件有关。每一种逻辑数据模型在实现时都有其对应的物理数据模型。DBMS 为了保证其独立性与可移植性，大部分物理数据模型的实现工作由系统自动完成，而设计者只设计索引、聚集等特殊结构。

1.3　数据库应用系统

1.3.1　应用系统的数据接口

客户端应用程序或应用服务器向数据库服务器请求服务时，首先必须和数据库建立连接。虽然现有 DBMS 几乎都遵循 SQL 标准，但不同厂家开发的 DBMS 有差异，存在适应性和可移植性等方面的问题，为此，人们研究和开发了连接不同 DBMS 的通用方法、技术和软件接口。

1. ODBC 数据库接口

ODBC 即开放式数据库互连，是微软公司推出的一种实现应用程序和关系数据库之间通信的接口标准。符合该标准的数据库可以通过 SQL 语句编写的程序对数据库进行操作，但只针对关系数据库。目前所有的关系数据库都符合该标准。ODBC 本质上是一组数据库访问 API（应用程序编程接口），由一组函数调用组成，核心是 SQL 语句。

在具体操作时，首先必须用 ODBC 管理器注册一个数据源，管理器根据数据源提供的数据库位置、数据库类型及 ODBC 驱动程序等信息，建立起 ODBC 与具体数据库的联系。这样，只要应用程序将数据源名提供给 ODBC，ODBC 就能建立起与相应数据库的连接。

MySQL 是通过 MySQL Connector/ODBC（MyODBC 驱动程序系列）为 ODBC 提供支持的，如图 1.8 所示说明了 MySQL 使用 ODBC 连接方式的结构。

其中，ODBC 驱动管理器是用于管理 ODBC 应用程序和驱动程序间通信的库。ODBC.INI 是 ODBC 配置文件，其中保存了连接到 MySQL 服务器所需的驱动信息和数据库信息。ODBC 驱动管理器将使用它来确定加载哪个驱动程序（使用数据源名）。

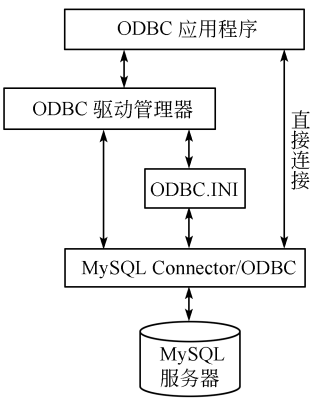


图 1.8　MyODBC 数据库接口

2. ADO 数据库接口

ADO（ActiveX Data Object）是微软公司开发的基于 COM 的数据库应用程序接口，通过 ADO 连接数据库，可以灵活地操作数据库中的数据。

使用 ADO 访问关系数据库有两种途径：一种是通过 ODBC 驱动程序，另一种是通过数据库专用的 OLE DB Provider，后者有更高的访问效率。

随着网络技术的发展，网络数据库及相关的操作技术也越来越多地应用到实际中，而数据库操作技术也在不断地发展完善。ADO 对象模型进一步发展成了 ADO.NET。ADO.NET 是.NET Framework SDK 中用于操作数据库的类库总称，ADO.NET 相对于 ADO 的最大优势在于对数据的更新修改可在与数据源完全断开连接的情况下进行，然后再把数据更新的结果和状态传回数据源，这样就大大减少了由于连接过多而对数据库服务器资源的占用。

3. ADO.NET 数据库接口

ADO.NET 数据模型从 ADO 发展而来，但它不只是对 ADO 的改进，而是采用了一种全新的技术，主要体现在以下几个方面。

（1）ADO.NET 采用的并不是 ActiveX 技术，而是与.NET 框架紧密结合的产物。

（2）ADO.NET 包含对 XML 标准的完全支持，这对于跨平台交换数据具有重要意义。

（3）ADO.NET 既能在与数据源连接的环境下工作，又能在与数据源断开连接的环境下工作。特别是第（3）个方面，非常适合于网络应用的需要，因为在网络环境下，始终做到保持与数据源连接，不符合网站的要求，不仅效率低，付出代价高，而且常会引发由于多个用户同时访问而带来的冲突。因此，ADO.NET 系统集中主要精力解决在与数据源断开连接的环境下数据处理的问题。

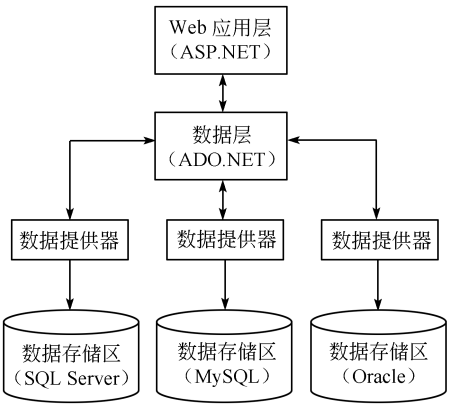


图 1.9　通过 ADO.NET 访问数据库的接口模型

ADO.NET 提供了面向对象的数据库视图，并且在其对象中封装了许多数据库属性和关系。最重要的是，它通过多种方式封装和隐藏了很多数据库访问的细节，可以完全不知道对象在与 ADO.NET 对象交互，也不用担心数据移动到另外一个数据库或者从另一个数据库获得数据等细节问题。如图 1.9 所示显示了通过 ADO.NET 访问数据库的接口模型。

数据层是实现 ADO.NET 断开式连接的核心，从数据源读取的数据先缓存到数据集中，然后被程序或控件调用。数据源可以是数据库或 XML 数据。

数据提供器用于建立数据源与数据集之间的联系，它能连接各种类型的数据源，并能按要求将数据源中的数据提供给数据集，或者从数据集向数据源返回编辑后的数据。

MySQL 使用 MySQL Connector/NET 实施了所需的 ADO.NET 接口，并将其集成到了 ADO.NET aware 工具中，从而使开发人员能够方便地创建需要安全和高性能数据连接（与 MySQL）的.NET 应用程序。MySQL Connector/NET 是用纯 C# 语言编写的可托管的 ADO.NET 驱动程序。

4. JDBC 数据库接口

JDBC（Java DataBase Connectivity）是 JavaSoft （原来 SUN 公司的业务部门）开发的、用 Java 语言编写的用于数据库连接和操作的类和接口，可为多种关系数据库提供统一的访问方式。通过 JDBC 对数据库的访问包括 4 个主要组件：Java 应用程序、JDBC 驱动管理器、驱动器和数

据源。

在 JDBC API 中有两层接口：应用程序层和驱动程序层，前者使开发人员可以通过 SQL 调用数据库和取得结果，后者处理与具体数据库驱动程序的所有通信。

使用 JDBC 接口操作数据库有如下优点。

（1）JDBC API 与 ODBC 十分相似，有利于用户理解。

（2）有利于编程人员从复杂的驱动器调用命令和函数中解脱出来，从而致力于应用程序功能的实现。

（3）JDBC 支持不同的关系数据库，增强了程序的可移植性。

使用 JDBC 的主要缺点：访问数据记录的速度会受到一定影响，此外，由于 JDBC 结构中包含了不同厂家的产品，这给对数据源的更改操作带来了较大麻烦。

MySQL 通过 MySQL Connector/J 驱动实现 JDBC 接口，它提供了与使用 Java 语言开发的客户端应用程序的连通性。MySQL Connector/J 是一种纯 Java 程序，能使用 MySQL 协议与 MySQL 服务器直接通信。

5. 数据库连接池技术

对于网络环境下的数据库应用，由于用户众多，使用传统的 JDBC 方式进行数据库连接，系统资源开销过大，易成为制约大型企业级应用效率的瓶颈；而采用数据库连接池技术对数据库连接进行管理，可以大大提高系统的效率和稳定性。

1.3.2　C/S 架构的应用系统

DBMS 通过命令和适合专业人员的界面操作数据库。对于 MySQL 数据库服务器，用户可以在客户端输入 SQL 命令，系统执行的结果再返回到客户端显示。用户还可以直接通过 MySQL 的界面管理工具来操作数据库。

对于一般的数据库应用系统，除了 DBMS 外，还需要设计适合普通人员操作数据库的界面。目前较为流行的开发数据库界面的工具主要有 Visual C++、Visual C#等。应用程序与数据库、数据库管理系统之间的关系如图 1.10 所示。

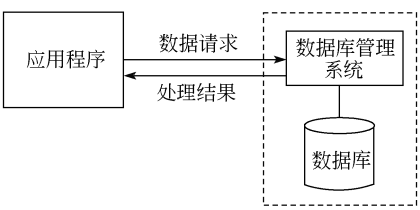


图 1.10　应用程序与数据库、数据库管理系统之间的关系

从图 1.10 可以看出，当应用程序需要处理数据库中的数据时，首先向数据库管理系统发送一个数据请求，数据库管理系统接收到这一请求后，对其进行分析，然后执行数据库操作，并把处理结果返回给应用程序。由于应用程序直接与用户交互，而数据库管理系统不直接与用户打交道，所以应用程序被称为“前台”，而数据库管理系统被称为“后台”。由于应用程序是向数据库管理系统提出服务请求，通常称为客户程序（Client），而数据库管理系统是为应用程序提供服务，通常称为服务器程序（Server），所以又将这一操作数据库的模式称为 C/S（客户/服务器）架构。

应用程序和数据库管理系统可以运行在同一台计算机上（单机方式），也可以运行在网络环境中。在网络环境下，数据库管理系统在网络中的一台主机上运行，应用程序可以在网络中的多台主机上运行，即一对多的方式。对于 MySQL，除了需要在服务器端安装数据库管理系统外，还需要在客户端安装客户程序。

例如，C/S 架构的学生成绩管理系统的学生信息查询界面如图 1.11 所示。

图 1.11　C/S 架构的学生成绩管理系统的学生信息查询界面

1.3.3　B/S 架构的应用系统

基于 Web 的数据库应用采用三层（浏览器/Web 服务器/数据库服务器）模式，也称 B/S 架构，如图 1.12 所示。其中，浏览器（Browser）是用户输入数据和显示结果的交互界面，用户先在浏览器表单中输入数据，然后将表单中的数据提交并发送到 Web 服务器，Web 服务器接收并处理用户的数据，通过数据库服务器，从数据库中查询需要的数据（或把数据录入数据库）送回 Web 服务器，Web 服务器把返回的结果插入 HTML 页面，传送给客户端并在浏览器中显示出来。

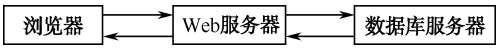


图 1.12　三层 B/S 架构

目前流行的开发数据库 Web 界面的工具主要有 PHP、Java EE、ASP.NET（C#）等。例如，B/S 架构的学生成绩管理系统的学生信息录入界面如图 1.13 所示。

图 1.13　B/S 架构的学生成绩管理系统的学生信息录入界面