第5章

数据库实践

本章概要

数据库管理与应用在大数据时代发挥着至关重要的作用，为数据的存储、组织、访问、保护和分析提供了强有力的支持。本章以实际应用为例，从数据库的设计与创建、数据查询与更新、数据管理与保护等方面开展实践，达到掌握利用关系型数据库进行数据管理和数据处理的能力。

学习目标

（1）掌握通过 SQL 语言创建数据库与数据表。
（2）掌握数据查询与数据更新的方法。
（3）掌握运用结构化查询语言解决数据问题。
（4）了解数据库安全及其管理方法。

5.1 数据库与数据表的创建

数据表也被称为表或基本表，是数据库最基本的用于存储数据的对象。可以认为关系数据库中的数据表是以行和列组成的二维表格，通常人们将行称为记录，将列称为字段。

在创建数据表时需要用到数据类型。因此在介绍创建表之前，先介绍一下 MySQL 支持的数据类型。

5.1.1 数据类型

MySQL 中定义数据字段的类型对数据库的优化是非常重要的。

MySQL 支持多种类型，大致可以分为四类：数值、字符串（字符）、日期/时间和复合类型。

1. 数值类型

数值型数据就是通常所说的数字，它可以由 0～9 的数字、正负符号与小数点（.）组成，例如，10，13.134，－12，－23.45 等。存放数值型数据的数据类型就是数值类型。MySQL 支持所有标准 SQL 数值数据类型。这些类型包括严格数值

数据类型(INTEGER、SMALLINT、DECIMAL 和 NUMERIC),以及近似数值数据类型(FLOAT、REAL 和 DOUBLE PRECISION)。

关键字 INT 是 INTEGER 的同义词,关键字 DEC 是 DECIMAL 的同义词。

作为 SQL 标准的扩展,MySQL 也支持整数类型 TINYINT、MEDIUMINT 和 BIGINT。表 5-1 显示了需要的每个整数类型的大小、范围和用途。

表 5-1 数值类型的大小、范围和用途

类 型	大 小	范围(有符号)	范围(无符号)	用 途
TINYINT	1B	−128～127	0～255	小整数值
SMALLINT	2B	−32 768～32 767	0～65 535	大整数值
MEDIUMINT	3B	−8 388 608～8 388 607	0～16 777 215	大整数值
INT 或 INTEGER	4B	−2 147 483 648～2 147 483 647	0～4 294 967 295	大整数值
BIGINT	8B	−9 223 372 036 854 775 808～9 223 372 036 854 775 807	0～18 446 744 073 709 551 615	极大整数值
FLOAT	4B	−3.402 823 466E+38～−1.175 494 351E−38 0, 1.175 494 351E−38～3.402 823 466 351E+38	0, 1.175 494 351E−38～3.402 823 466E+38	单精度浮点数值
DOUBLE	8B	−1.797 693 134 862 3157E+308～−2.225 073 858 507 201 4E−308 0, 2.225 073 858 507 201 4E−308～1.797 693 134 862 3157E+308	0, 2.225 073 858 507 2014E−308～1.797 693 134 862 3157E+308	双精度浮点数值
DECIMAL[(*M*,[*D*])]或 NUMERIC(*M*,*D*)	对 DECIMAL(*M*,*D*),如果 *M* > *D*,为 *M*+2,否则为 *D*+2	依赖于 *M* 和 *D* 的值	依赖于 *M* 和 *D* 的值	可变精度浮点数值,*M* 代表包括小数点的数字的长度(但不包括负号);*D* 代表小数点右边的位数。*M* 默认为 10,*D* 默认为 0

2. 字符串类型

字符型数据是数据库中常用的数据类型之一,有时也将其称为字符串。例如,在一个存储图书信息的表中,书名、出版社、ISBN 都是字符型数据。字符型数据可由字母、数字、空白符、标点、特殊字符与汉字等符号组成。在 SQL 中,字符型数据被放在一对半角单引号('')中,用于区别其他类型的数据。

字符串类型就是用来存放字符型数据的。字符串类型指 CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT。表 5-2 描述了这些类型的大小和用途。

表 5-2　字符串类型的大小和用途表

类　型	大　小	用　途
CHAR	0～255B	定长字符串
VARCHAR	0～65 535B	变长字符串
TINYBLOB	0～255B	不超过 255 个字符的二进制字符串
TINYTEXT	0～255B	短文本字符串
BLOB	0～65 535B	二进制形式的长文本数据
TEXT	0～65 535B	文本数据
MEDIUMBLOB	0～16 777 215B	二进制形式的中等长度文本数据
MEDIUMTEXT	0～16 777 215B	中等长度文本数据
LONGBLOB	0～4 294 967 295B	二进制形式的极大文本数据
LONGTEXT	0～4294967295B	极大文本数据

3. 日期与时间类型

表示时间值的日期和时间类型为 DATETIME、DATE、TIMESTAMP、TIME 和 YEAR。每个时间类型都有一个有效值范围和一个“零”值，当指定不合法的 MySQL 不能表示的值时使用“零”值。TIMESTAMP 类型有专有的自动更新特性。表 5-3 描述了这些类型的大小、范围、格式和用途。

表 5-3　日期与时间类型的大小、范围、格式和用途

类　型	大小/B	范　围	格　式	用　途
DATE	3	1000-01-01/9999-12-31	YYYY-MM-DD	日期值
TIME	3	'－838:59:59'/'838:59:59'	HH:MM:SS	时间值或持续时间
YEAR	1	1901/2155	YYYY	年份值
DATETIME	8	1000-01-01 00:00:00/ 9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	混合日期和时间值
TIMESTAMP	4	1970-01-01 00:00:00/2038 结束时间是第 2 147 483 647 秒， 北京时间 2038-1-1911:14:07， 格林尼治时间 2038 年 1 月 19 日 凌晨 03:14:07	YYYY-MM-DD HH-MM-SS	混合日期和时间值， 时间戳

4. 复合类型

(1) ENUM('value1','value2',…)类型：用于存储从预先定义的字符集合中选取互斥的数据值，可以有 65 535 个不同的值。

(2) SET('value1','value2',…)类型：用于存储从预先定义的字符集合中选取任意数目的值，可以有 64 个成员。

5.1.2　数据表基础

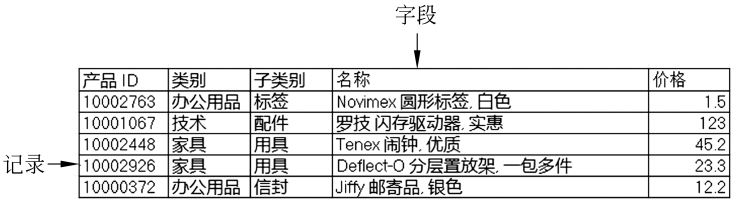
当你将资料放入自己的文件柜时，并不是随便将它们扔进某个抽屉，而是在文件柜中创

建文件夹，然后将相关的资料放入特定的文件夹中，在数据库领域中这种文件称为数据表(Table)。

数据表又被称为表。在关系数据库系统中，一个关系就是一个表，表结构指的就是数据库的关系模型，表示若干列(Column)和若干行(Row)的集合，每一行代表一个唯一的记录，每一列代表一个字段。在确定表结构时首先要定义表的字段，即定义字段名、数据类型及其宽度，其次输入行(记录)。

1. 数据表中的记录和字段

关系数据库中的数据表其实很像生活中的二维表格，甚至有人会说它就是二维表格。数据表由行和列组成，通常人们将行称为记录，将列称为字段，如图 5-1 所示。



产品 ID	类别	子类别	名称	价格
10002763	办公用品	标签	Novimex 圆形标签, 白色	1.5
10001067	技术	配件	罗技 闪存驱动器, 实惠	123
10002448	家具	用具	Tenex 闹钟, 优质	45.2
10002926	家具	用具	Deflect-O 分层置放架, 一包多件	23.3
10000372	办公用品	信封	Jiffy 邮寄品, 银色	12.2

图 5-1 数据表

表由列组成。列中存储着表中某部分的信息，是表中的一个字段。所有表都是由一个或多个列组成的。每个字段中的数据必须具有相同的数据类型，且每个字段都有字段名，如图 5-1 中的“产品 ID”“名称”等就是字段名。关系数据库中规定在同一个表内不能有重复的字段。表中的数据是按行存储的，所保存的每个记录存储在自己的行内。实际上，表内也不应该有重复的记录，只是多数数据库管理系统不会强制这一点而已。

2. 表结构

一个非空数据表实际上是由三部分组成的，分别是表名、表结构和表的记录。表结构由表中所有字段的字段信息组成，这些信息包括字段名、字段类型、字段大小和字段约束、表约束等。创建一个数据表，其实就是在创建其表结构。

5.1.3 表逻辑设计

数据表的设计是数据库设计的主要部分，表逻辑设计的好坏将会影响数据库系统最终的运行效果、数据安全以及完整性。表的逻辑结构设计必须满足用户的需求，使用户准确理解数据的本质且容易掌握，并且没有二义性。E-R 模型则能帮助系统开发人员很好地完成表逻辑设计。

1. E-R 模型图

E-R 模型图也称实体-联系图(Entity-Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。它是描述现实世界关系概念模型的有效方法，是表示概念关联模型的一种方式。

E-R 模型图用“矩形框”表示实体型，矩形框内写明实体名称；用“椭圆图框”或圆角矩形表示实体的属性，并用“实心线段”将其与相应关联的“实体型”连接起来；用“菱形框”表示实体型之间的联系成因，在菱形框内写明联系名，并用“实心线段”分别与有关实体型连接起来，同时在“实心线段”旁标上联系的类型(1∶1，1∶n 或 m∶n)。

E-R 模型图是一种自上而下的数据库设计方法。一个完整的数据库系统的 E-R 模型图是由若干局部 E-R 模型图组合而成的。

在 E-R 模型图方法中，将局部概念结构图称为局部 E-R 模型图。局部 E-R 模式的设计过程如图 5-2 所示。

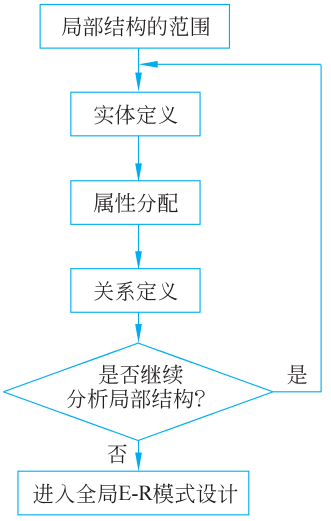


图 5-2　局部 E-R 模式的设计过程

例如，公司销售系统数据库中的商品管理与员工管理的局部 E-R 模型图，分别如图 5-3 和图 5-4 所示。

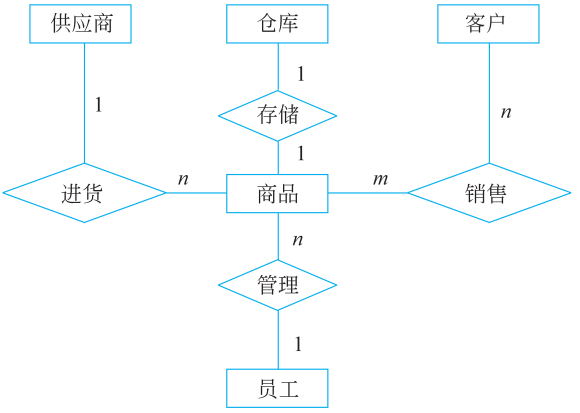


图 5-3　商品管理局部 E-R 模型图

将局部 E-R 模型图合并成全局 E-R 模型图的方法有两种：一种是一次合并多个局部 E-R 模型图，另一种是逐步合并局部 E-R 模型图，如图 5-5 所示。

无论采用哪种方法，合并局部 E-R 模型图的准则是先解决局部 E-R 模型图的冲突，合并成初步 E-R 模型图，然后进行初步 E-R 模型图的优化与修改，最终得到全局的 E-R 模型图。例如，合并商品管理和员工管理局部 E-R 模型图后，得到的 E-R 模型图如图 5-6 所示。

2. 规范化与范式

规范化是一种用来产生数据表集合的技术，通过规范化，表将具有符合用户需求的属

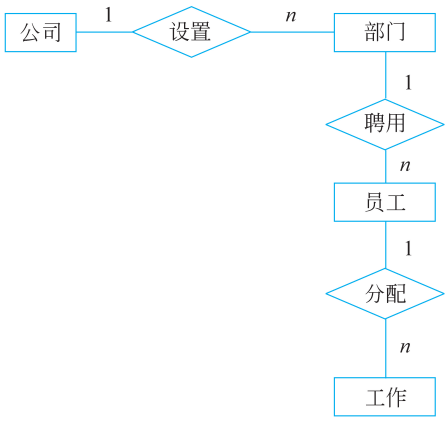


图 5-4　员工管理局部 E-R 模型图

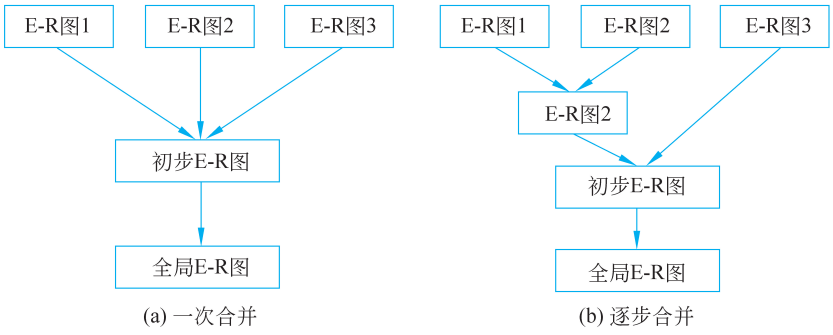


图 5-5　合并局部 E-R 模型图的方法

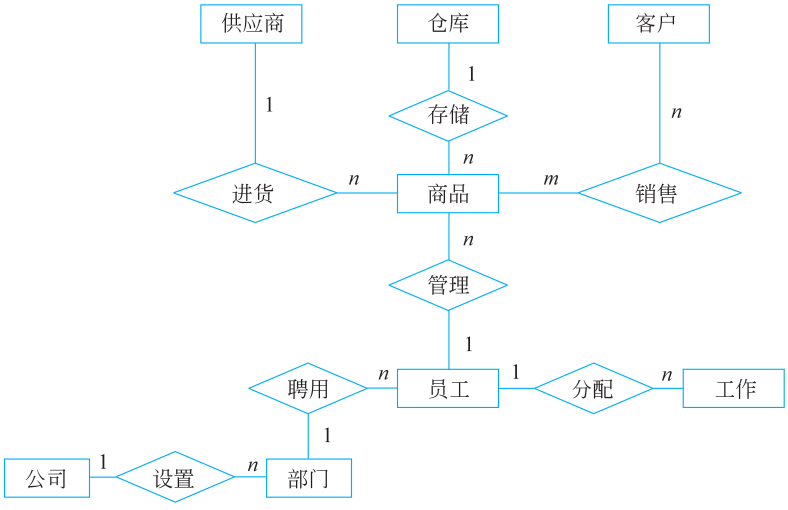


图 5-6　合并后的 E-R 模型图

性。规范化通常作为对表结构的一系列测试来决定其是否满足和符合给定范式。数据库逻辑结构设计产生的结果应该满足规范化要求，以使关系模式的设计合理，达到冗余少和提高查询效率的目的，所以对数据库进行规范化非常重要。对数据库的规范化要确定规范化级

别，然后按要求进行并且要达到这一级别。

一般情况下，规范化处理主要进行以下三个步骤。

（1）确定数据依赖：通过数据依赖表示出数据项之间的关系，此项工作在需求分析阶段完成。

（2）定义键并消除冗余的关系：此项工作在概要设计阶段完成。

（3）确定范式级别：规范化必须要达到范式级别。

范式简称 NF(Normal Form)，是满足一定条件的关系模式。范式是规范化确定的级别，数据库设计的范式有多种，常用的有第一范式(1NF)、第二范式(2NF)和第三范式(3NF)。所有范式都基于数据表中字段之间的关系。

第一范式：若关系模式 P 的所有属性的值域中每个值都是不可再分解的值，则称 P 为第一范式。第一范式是最低的规范化要求，数据表不能存在相同的记录，需设定一个关键字，并且要求每个字段都不可再分解。

第二范式：若关系模式 P 是第一范式，P 的表以及每个非主键字段都可以由构成主键的全部的字段得到，则称 P 为第二范式。第二范式可以消除大量的冗余数据，并对数据表可以进行异常的插入和删除。

第三范式：若关系模式 P 是第二范式，且每个非主属性都不传递依赖于 P 的候选键，则称 P 是第三范式。第三范式的关系不具有多义性，其属性值唯一，且每个非主属性必须依赖于整个主键而不能依赖于其他关系中的属性。

5.1.4　创建数据库与数据表

在创建数据库对象之前必须先创建数据库，数据库中包含数据表视图、查询规则、默认值等数据库对象，并且对这些对象进行统一管理。

1. 创建数据库

在面向对象的关系数据库管理系统中，一般情况下，用户使用 DBMS 环境中的工具创建数据库。用户也可以使用 SQL 语句中的 CREATE DATABASE 语句创建数据库，基本语法格式如下。

```
CREATE DATABASE  <databasename>;
```

例 5-1：在 MySQL Workbench 中，使用 CREATE DATABASE 语句创建一个 test 数据库。

```
CREATE DATABASE test;
```

运行结果如图 5-7 所示。

2. 删除数据库

删除数据库可以使用 DROP DATABASE 语句，基本语法格式如下。

```
DROP DATABASE  <databasename>;
```

例 5-2：在 MySQL Workbench 中，使用 DROP DATABASE 语句删除 test 数据库。

```
DROP DATABASE test;
```

3. 创建表

SQL 中创建表用 CREATE TABLE 语句来实现。CREATE TABLE 语句可以定义表

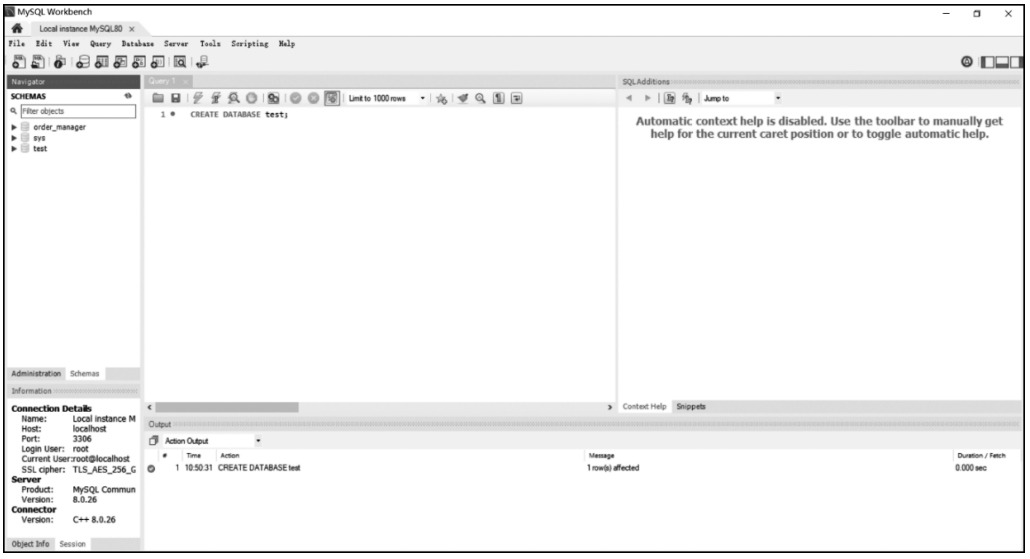


图 5-7　创建 test 数据库

的结构、约束以及继承等内容。

CREATE TABLE 语句的功能是在当前数据库中创建一个新的数据表，创建表需要表名、表字段名和定义每个表字段等信息，基本语法格式如下。

```
CREATE TABLE <表名>(
    <字段名 1><数据类型> [NOT NULL][DEFAULT<默认值>],
    <字段名 2><数据类型> [NOT NULL][DEFAULT<默认值>],
    …
    <字段名 n><数据类型> [NOT NULL][DEFAULT<默认值>],
);
```

- NOT NULL：为可选项，如果在某字段后加上该项，则向表添加数据时，必须给该字段输入内容，即不能为空。
- DEFAULT＜默认值＞：为可选项，如果在某字段后加上该项，则向表添加数据时；如果不向该字段添加数据，系统就会自动用默认值填充该字段。

例 5-3：创建一个 goods(商品)表，设置商品 ID、商品名、商品价格、商品描述、类别。表结构描述如表 5-4 所示。

表 5-4　商品表的表结构

列	数据类型	说　明
goods_id	varchar(4)	唯一的商品 ID
goods_name	varchar(50)	商品名
price	double	商品价格
goods_desc	varchar(255)	商品描述
category	varchar(20)	类别

创建语句如下。

```
CREATE TABLE goods(
    goods_idvarchar(4) NOT NULL,
    goods_name varchar(50) NOT NULL,
    price double DEFAULT 0,
    goods_desc varchar(255),
    category varchar(20),
    PRIMARY KEY (goods_id)
);
```

在数据表中能够唯一识别记录的字段都会被人们设置为主键。当某个字段被设置为主键后，该字段中就不能再有重复值，也不能有空值。数据库管理系统将强制执行这一规则，这就是主键约束。如果想设置多个字段为主键，只需要在语句 PRIMARY KEY(字段名 1，字段名 2…)中增加字段名即可。

4. 修改表

在操作数据库时，可能需要更改表结构，如修改某字段的数据类型、添加新字段、删除指定字段等。可以使用 ALTER TABLE 语句完成上述要求。

1) 添加字段的语法格式

```
ALTER TABLE 表名
ADD
字段名 数据类型[(长度)];
```

其中，字段名是需要添加的字段名称。数据类型是需要添加字段的数据类型。长度是需要添加字段的长度，该项为可选项，当需要添加的字段类型为带长度的数据类型时必须定义其长度，如字符类型。

2) 修改字段的语法格式

```
ALTER TABLE 表名
MODIFY
字段名 数据类型[(长度)];
```

其中，字段名是需要修改的字段名称；数据类型是需要添加字段的数据类型；长度是需要修改字段的长度。

3) 删除字段的语法格式

```
ALTER TABLE 表名
DROP
字段名;
```

其中，字段名为要删除的字段名。

例 5-4：为例 5-3 创建的商品表添加字段“产地”用来存储商品的产地，修改商品名字段将长度改为 100，删除“商品描述”字段。

SQL 语句如下：

(1) 添加“产地”字段。

```
ALTER TABLE goods
ADD
origin varchar(20);
```

（2）修改商品名字段长度。

```
ALTER TABLE goods
MODIFY
goods_name varchar(100);
```

（3）删除“商品描述”字段。

```
ALTER TABLE goods
DROP
goods_desc;
```

5. 删除表

当不再需要数据库中的某表时，就应当删除该表，释放该表所占有的资源。在 SQL 中，删除数据表使用 DROP TABLE 语句。

使用 DROP TABLE 语句会将表彻底删除，包括表内的数据和表本身，但有时会只希望删除表中数据而不删除表本身，这时可以使用 TRUNCATE 语句将表截断，即删除表内的所有数据。

1）删除表的语法格式

```
DROP TABLE 表名;
```

2）截断表的语法格式

```
TRUNCATE TABLE 表名;
```

6. 创建本书使用的数据表

本书后面的大多数实例都使用 order_manager 数据库的 goods、customer、orders 数据表，下面列出创建这些数据库表的 SQL。

1）创建 goods 表的 SQL 语句

```
CREATE TABLE goods(
goods_id varchar(4) NOT NULL,
goods_name varchar(50) NOT NULL,
price double DEFAULT 0,
goods_desc varchar(255),
category varchar(20),
PRIMARY KEY (goods_id)
);
```

2）创建 customer 表的 SQL 语句

```
CREATE TABLE customer (
  cust_id varchar(4) NOT NULL,
  cust_name varchar(100) NOT NULL ,
  address varchar(100),
  city varchar(50),
  tel varchar(18),
  fax varchar(18),
  contact varchar(100),
  email varchar(100),
  PRIMARY KEY (cust_id)
);
```