# 第3章 chapter 3

# 数据库物理设计与实施

数据库最终需要存储在物理设备上，它在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于给定的计算机系统。对于给定的逻辑数据模型，选取一个最适合应用环境的物理结构的过程，称为数据库的物理设计。不同的数据库产品所提供的物理环境、存取方法和存储结构有很大差别，能供设计人员使用的设计变量、参数范围也很不相同。因此，没有通用的物理设计方法可遵循，只能给出一般的设计内容和原则，希望能设计出优化的物理数据库结构，使得在数据库上运行的各种事务响应时间短、存储空间利用率高、事务吞吐率大。此阶段是以逻辑设计的结果作为输入，结合具体 DBMS 的特点与存储设备的特性进行设计，选定数据库在物理设备上的存储结构和存取方法。

数据库的物理设计可分为三步：

（1）分析影响物理数据库设计的因素。

（2）确定物理结构，在关系数据库中主要指存取方法和存储结构。

（3）评价物理结构，评价的重点是时间和空间效率。

如果评价结果满足原设计要求，则可以进行物理实施，否则应该重新设计或修改物理结构，有时甚至返回逻辑设计阶段修改数据模型。

本章首先介绍开源数据库管理系统 MySQL，其次介绍数据库的物理设计及其建立的三个步骤，以帮助读者建立物理数据库设计的整体思路；然后，介绍物理表的设计，从数据库设计的最基础工作开始讲解，进而使读者了解如何设计物理表以减少不必要的系统开支；在基本表建立后，进一步介绍数据库的性能优化方法，主要从索引技术、存储结构设计和其他方面进行调优；最后，介绍数据库的故障判断及排除，列举了一些在数据库设计的过程中经常出现的错误。

## 3.1 数据库管理系统实例——MySQL 简介

随着开源软件的广泛使用，开源数据库的出现成为必然。当 Oracle、IBM DB2、Microsoft SQL Server、Sybase 等几大数据库巨头在数据库领域处于垄断地位的时候，出现了以 MySQL、PostgreSQL 等为代表的开源数据库系统，大大推动了开源软件事业的发展。MySQL 是一个精巧的 SQL 数据库管理系统。由于它的功能强大、操作灵活、系

统结构精巧，因此受到了广大自由软件爱好者甚至是商业软件用户的青睐，特别是它与Apache的结合，为建立基于数据库的动态网站提供了强大动力。最关键的是用户可以从Internet上免费下载并使用的社区版，足够胜任一般中小型甚至大型应用。MySQL的主要特点是快速、健壮和易用。它可媲美任何昂贵的大型数据库系统，并且速度更快。

MySQL是一个开放源码的小型关系型数据库管理系统，由MySQL AB公司所开发。目前，MySQL已被广泛应用在Internet上的中小型网站开发中。由于其拥有体积小、速度快、总体成本低，尤其是开放源码等特点，许多中小型网站为了降低网站总体开发成本而选择了MySQL作为网站数据库。

### 3.1.1 MySQL的安装过程

按图3-1至图3-12所示的步骤安装MySQL 5.0，其中有些选项和默认值是不一样的。如果重新安装MySQL，要注意先备份好MySQL目录中的data目录，这个目录是所有用户的数据目录，非常重要，不可大意。

重装时，备份好数据后，应先卸载掉旧版本的MySQL，并删除其安装目录。

MySQL 5.0的安装步骤如下：

进入MySQL 5.0安装界面后单击Next，出现图3-1所示的界面。


**图3-1 MySQL 5.0的安装步骤1**

选择Custom选项后单击Next，出现图3-2所示的个性化设置界面。

选择好要安装的部件后单击Change，出现图3-3所示的设置安装目录的界面。

请注意，为了数据安全，建议用户不要把MySQL安装在系统盘。单击OK，在Ready to Install the Program界面单击Install。然后在两个MySQL Enterprise界面都单击Next，在Setup Wizard界面单击Finish。在图3-4所示的MySQL Server Instance Configuration Wizard界面选择服务器类型后单击Next。

在图3-5所示的界面中，单击Next。

在图3-6所示的界面中，单击Next。

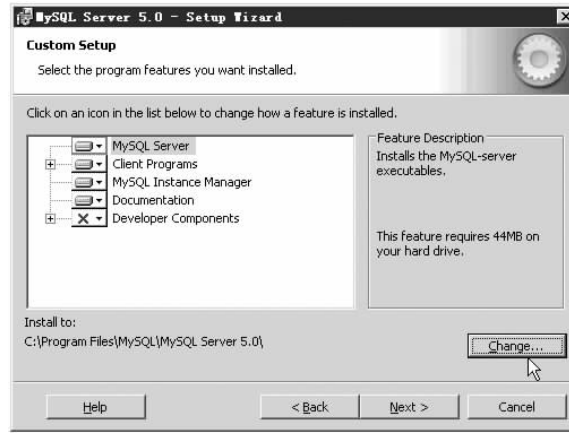


图 3-2　MySQL 5.0 的安装步骤 2

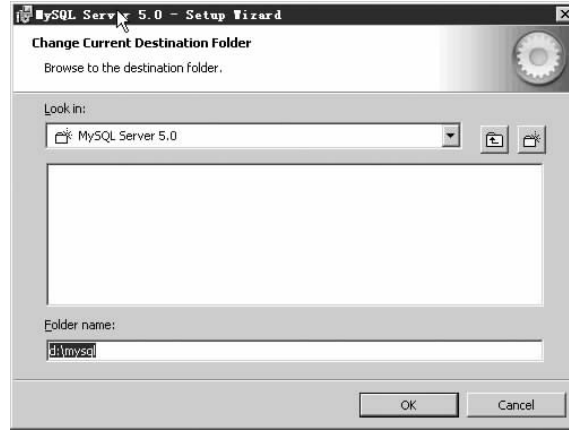


图 3-3　MySQL 5.0 的安装步骤 3


图 3-4　MySQL 5.0 的安装步骤 4

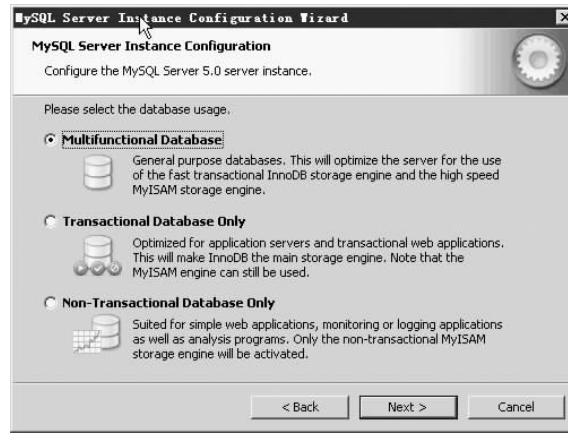


图 3-5 MySQL 5.0 的安装步骤 5

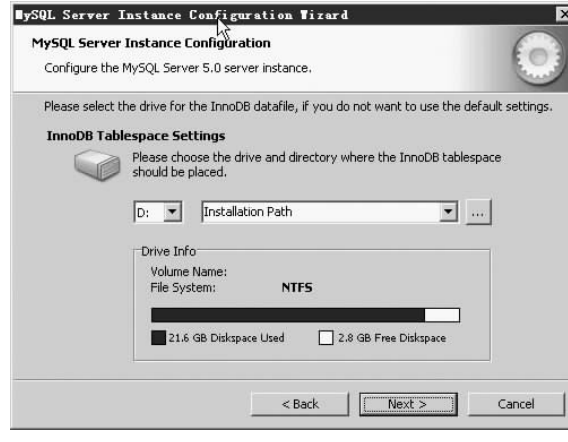


图 3-6 MySQL 5.0 的安装步骤 6

请注意图 3-7 中的设置,把 MySQL 的同时连接数改为了 1000,单击 Next。

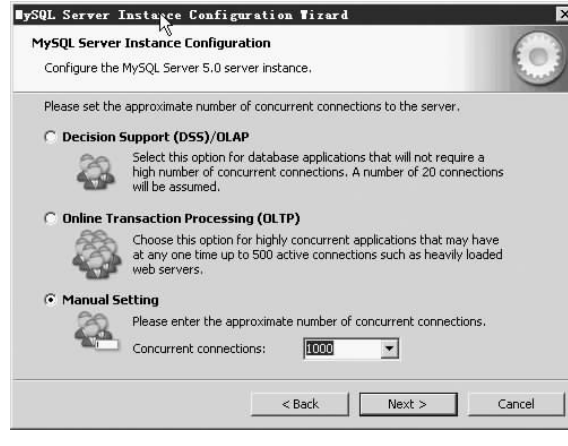


图 3-7 MySQL 5.0 的安装步骤 7

在图 3-8 所示的界面中，单击 Next。

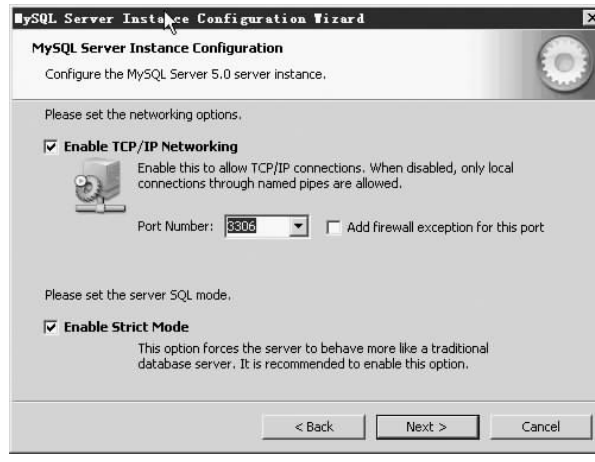


**图 3-8　MySQL 5.0 的安装步骤 8**

在图 3-9 所示的界面中，单击 Next。

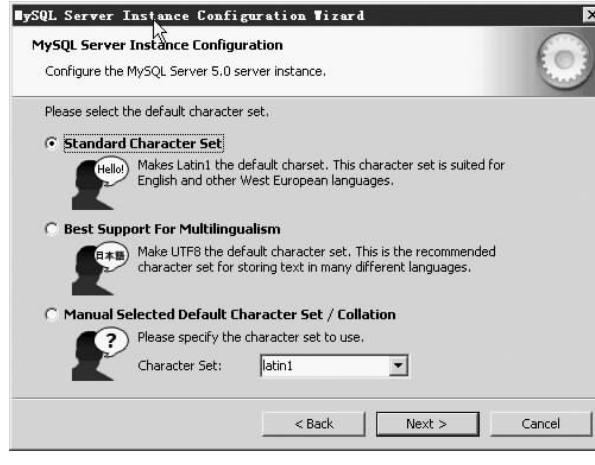


**图 3-9　MySQL 5.0 的安装步骤 9**

请注意在图 3-10 中，MySQL 的服务名称选为“MySQL”，不要使用其他符号，单击 Next。在图 3-11 中所填密码即是 MySQL 的管理员 Root 的密码，单击 Next。然后在 MySQL Server Instance Configuration Wizard 界面单击 Execute。至此 MySQL 安装成功，在图 3-12 所示的界面中单击 Finish。

## 3.1.2　MySQL 中游标的使用

游标是系统为用户开设的一个数据缓冲区，用来存放 SQL 语句的执行结果，由系统或用户以变量的形式定义。用户可以用 SQL 语句逐一从游标中获取记录，并赋给主变量，交由主语言进一步处理。主语言是面向记录的，一组主变量一次只能存放一条记录。


图 3-10 MySQL 5.0 的安装步骤 10


图 3-11 MySQL 5.0 的安装步骤 11


图 3-12 MySQL 5.0 的安装步骤 12

在某些情况下,需要把数据从存放在磁盘的表中调到计算机内存中进行处理,最后将处理结果显示出来或最终写回数据库。这样数据处理的速度才会提高,否则频繁的磁盘数据交换会降低效率。用数据库语言来描述游标就是映射在结果集中的一行数据上的位置实体。有了游标,用户就可以访问结果集中的任意一行数据了。将游标放置到某行后,即可对该行数据进行操作,例如提取当前行的数据等。

游标的使用可分为 4 个步骤。

**1. 定义游标**

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

**2. 打开游标**

```
OPEN cursor_name;
```

**3. 取数并推进游标指针**

```
FETCH cursor_name INTO variable list;
```

FETCH 获取游标当前指针的记录,并传给指定变量列表,注意变量数必须与 MySQL 游标返回的字段数一致。要获得多行数据,可使用循环语句去执行 FETCH。

**4. 关闭游标**

```
CLOSE cursor_name;
```

**注意**:MySQL 的游标是向前只读的。也就是说,只能从前往后顺序读取结果集,不能从后往前,也不能直接跳到中间的记录。

## 3.2 物理表的设计

创建一个模式,就建立了一个数据库的命名空间,即一个框架。在这个空间中首先要定义的是该模式包含的数据库基本表。

MySQL 语言中使用 CREATE TABLE 语句定义基本表,其基本格式如下:

```
CREATE TABLE<表名>
(<列名><数据类型>[列级完整性约束条件],
<列名><数据类型>[列级完整性约束条件],
<列名><数据类型>[列级完整性约束条件]……);
```

建表的同时通常还可以定义与该表有关的完整性约束条件,这些完整性约束条件被存入系统的数据字典中。当用户操作表中数据时,由 RDBMS 自动检查该操作是否违背这些完整性约束条件。如果完整性约束条件涉及该表的多个属性列,则必须定义在表级上;否则既可以定义在列级,也可以定义在表级。

### 3.2.1 主键和外键的设计

**例 3-1** 建立一个“读者”表 reader，将 reader_id 定义为主键，reader_type_code 定义为与表 reader_type_code 相关的外键，in_user_id 和 out_user_id 定义为不能为空。

代码如下：

```
CREATE  TABLE  reader
(reader_id  varchar(20) not null primary key,
in_user_id  varchar(20) not null,
out_user_id varchar(20) not null,
reader_type_code varchar(10) not null,
reader_name varchar(20),
reader_tel varchar(20),
reader_email varchar(50),
create_date datetime,
delete_time datetime,
borrow_count int,
pause_date datetime,
user_state varchar(20),
password varchar(20),
foreign key(reader_type_code) references reader_type(reader_type_code));
```

**1. 定义**

(1) 主键

表通常具有能包含唯一标识表中每一行的值的一列或多列，这样的一列或多列称为表的主键（PRIMARY KEY，简称 PK），用于强制表的实体完整性。在创建或修改表时，可以通过定义 PRIMARY KEY 约束来创建主键，如例 3-1 的 reader_id 字段。

PRIMARY KEY 约束要求构成主键的某列或者某几列的值必须唯一，以保证数据的唯一性，所以约束中的列不能接受空值。如果为表指定了 PRIMARY KEY 约束，则数据库引擎将通过为主键列创建唯一索引来强制数据的唯一性。

**注意**：如果对多列定义了 PRIMARY KEY 约束，即一个表由有多列组成复合主键，则其中某一列或几列的值允许重复，但构成主键的所有列值的组合必须唯一。

(2) 外键

外键（FOREIGN KEY，简称 FK）是用于建立和加强两个表数据之间的链接的一列或多列。创建或修改表时可通过定义 FOREIGN KEY 约束来创建外键，如例 3-1 的 reader_type_code 字段。

在外键引用中，当一个表的列被引用作为另一个表的主键值的列时，就在两表之间创建了链接，这个列就成为第二个表的外键。

**注意**：FOREIGN KEY 约束可以包含空值。但是，如果任何组合 FOREIGN KEY 约束的列包含空值，则将跳过组成 FOREIGN KEY 约束的所有值的验证。若要确保验

证了组合 FOREIGN KEY 约束的所有值，请将所有参与列指定为 NOT NULL。

### 2. 设计原则

将数据库从逻辑结构转换为物理结构时，主键和外键的设计是非常重要的，对物理数据库的性能和可用性都有着重大的影响。一旦将数据库投入使用，再对主键和外键进行修改就比较麻烦，所以在物理设计阶段要特别重视主键和外键的结构设计。

主键在物理层面上有两个用途：

- 作为每条记录唯一的标识，并且使用主键进行查询时，利用主键的唯一索引可以对数据进行快速访问。
- 作为一个可以被外键有效引用的对象。

因此，在注重数据库的性能与可用性方面，主键的设计一般要遵循一些原则。

(1) 尽量避免复合主键，只选择单列作为主键

单列主键对提高连接和筛选操作的效率是明显的，并且复合主键常常会导致不良的外键。例如，连接表成为另一个从表的主表，而使用多个外部键来作为这个表主键的一部分，然后这个表又有可能再成为其他从表的主表，其主键又有可能成为其他从表主键的一部分。如此传递下去，越靠后的从表，其主键包含的列就会越多。

例如在图书管理系统的数据库中，有一张“借阅表”，它记录了与读者借阅图书相关的所有信息，是连接读者与图书馆的最重要的表之一。因为“借阅”并不是一个实物，所以没有一个唯一能标识的属性，有一种设计主键的方法就是：以“借书证号”和“图书编号”作为复合主键，用于标识一次借阅事件。在不考虑多次借同一本书的情况下，这似乎是一种可行的方案，如表 3-1 所示。

**表 3-1　借阅表**

| 字　段　名 | 中 文 含 义 | 类　型 | 长度 | 说　　明 |
|---|---|---|---|---|
| book_id | 图书编号 | Varchar | 20 | 强制性、主键、外键 |
| reader_id | 借书证号 | Varchar | 20 | 强制性、主键、外键 |
| borrow_user_id | 借阅操作员代码 | Varchar | 20 | 强制性、外键 |
| return_user_id | 归还操作员代码 | Varchar | 20 | 强制性、外键 |
| borrow_date | 借阅日期 | Datetime | | |
| return_date | 归还日期 | Datetime | | |
| delaying_payment | 超期费用 | Float | | |

在很多情况下，数据库设计人员都喜欢采用这种方案。比如一个教学管理系统，“授课表”中可以由“课程号”、“授课教师号”和“学期”三者共同组成一个复合主键。部分设计者这样做的目的是为了使主键具有“实际意义”，然而这个“实际意义”不但不能提高查询效率，有时反而给人为入侵数据库提供了方便。并且当一个关系元数越高，主键就可能越复杂。

在图书管理系统中，借阅表以“借书证号”和“图书编号”作为复合主键，不能解决同

一事件重复发生的问题，即读者多次借阅同一本书。这时，必须加入一个“流水号”，作为借阅事件的唯一标识，每次借阅都会产生唯一的“流水号”来识别该次事件。这时“借书证号”和“图书编号”将不再作为主键，而由“流水号”取而代之，如表 3-2 所示。

**表 3-2 改进后的借阅表**

| 字 段 名 | 中 文 含 义 | 类 型 | 长度 | 说 明 |
|---|---|---|---|---|
| book_id | 图书编号 | Varchar | 20 | 强制性、外键 |
| reader_id | 借书证号 | Varchar | 20 | 强制性、外键 |
| borrow_user_id | 借阅操作员代码 | Varchar | 20 | 强制性、外键 |
| return_user_id | 归还操作员代码 | Varchar | 20 | 强制性、外键 |
| borrow_date | 借阅日期 | Datetime | | |
| return_date | 归还日期 | Datetime | | |
| delaying_payment | 超期费用 | Float | | |
| Seq_id | 流水号 | Int | | 强制性、主键 |

（2）主键不应包含动态变化的数据

如时间戳、创建时间列、修改时间列等。

（3）尽量不要更新主键

一般情况下，主键的功能只是唯一地标识一行，而不会再有其他用途，所以也就没有必要去对它进行更新。

**3. 外键设计**

数据表中主键的设计是必需的，而对外键的设计目前还没有广泛的共识。外键的作用是保持数据完整性，但是它约束了数据，也降低了性能。用与不用外键需要根据具体的项目环境而定。

一个严格使用 CDM、PDM 模型设计的数据库加上精确的程序控制或触发器控制，也能够很好地完成外键保持数据完整性的功能，因此很多设计是在程序中控制数据的完整性，而且性能更好。使用程序精确控制数据完整性的前提是全面了解整个系统的数据流向，此时设计过程中的文档是非常重要的，所以在对于效率要求较高，而对于数据完整性、一致性要求不是很高的情况下，可以考虑不使用外键。如果开发能力没有达到这个程度，那么使用外键就是必要的。使用外键即使在数据库服务器死机或者出现其他问题的时候，也能够最大限度地保证数据的一致性和完整性。因此在性能要求不高，而安全要求较高的情况下，应当合理使用外键。

## 3.2.2 数据类型的选择

关系模式中一个很重要的概念是域。每个属性来自一个域，它的取值必须是域中的值。

在 MySQL 中域的概念用数据类型来实现。定义表的各个属性时需要指明其数据类型及长度。MySQL 提供了一些主要的数据类型，如表 3-3 所示。要注意，不同的 RDBMS 中支持的数据类型不完全相同。

一个属性选用哪种数据类型需根据实际情况而定。一般需从两个方面来考虑：一是取值范围，二是运算类型。例如，对于年龄(Sage)属性，可以采用 CHAR(3)作为数据类型，但考虑要在年龄上做算术运算，所以采用整数类型比较合适。整数又有长整数和短整数两种，因为一个人的年龄在百岁左右，所以选用短整数作为年龄的数据类型。

**表 3-3　MySQL 的主要数据类型**

| 数据类型 | 含　义 |
|---|---|
| CHAR(n) | 长度为 n 的定长字符串 |
| VARCHAR(n) | 最大长度为 n 的变长字符串 |
| INT | 长整数(也可以写作 INTEGER) |
| SMALLINT | 短整数 |
| NUMERIC(p,d) | 定点数，由 p 位数字(不包括符号、小数点)组成，小数后面有 d 位数字 |
| REAL | 取决于机器精度的浮点数 |
| Double Precision | 取决于机器精度的双精度浮点数 |
| FLOAT(n) | 浮点数，精度至少为 n 位数字 |
| DATE | 日期，包含年、月、日，格式为 YYYY-MM-DD |
| TIME | 时间，包含一日的时、分、秒，格式为 HH:MM:SS |

### 1. 常规数据类型

字段的常规数据类型包括整数型、浮点数型、字符串型、二进制类型、日期和时间类型等，如表 3-3 所示，数据类型决定了数据的存储格式、约束条件和有效范围。表中的每个字段都有数据类型。MySQL 中，ALTER TABLE 语句也可以修改字段的数据类型。其基本语法如下：

```
ALTER TABLE 表名 MODIFY 属性名 数据类型
```

其中，“表名”参数指所要修改的表的名称；“属性名”参数指需要修改的字段的名称；“数据类型”参数指修改后的新数据类型。

**例 3-2**　修改 reader 表中 reader_name 字段的数据类型。SQL 代码如下：

```
ALTER TABLE reader MODIFY reader_name VARCHAR(30);
```

如果代码运行成功，user 表中 reader_name 字段的数据类型变为 VARCHAR(30)。在执行代码之前，先用 DESC 语句查看 user 表的结构。其中可以看到 reader_name 字段

修改前的数据类型，以便与修改后进行对比。DESC 语句执行后的显示结果如图 3-13 所示。

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| reader_id | varchar(20) | NO | PRI | NULL | |
| in_user_id | varchar(20) | NO | | NULL | |
| out_user_id | varchar(20) | YES | | NULL | |
| reader_type_code | varchar(10) | YES | MUL | NULL | |
| reader_name | varchar(20) | YES | | NULL | |
| reader_tel | varchar(20) | YES | | NULL | |
| reader_email | varchar(50) | YES | | NULL | |
| create_date | datetime | YES | | NULL | |
| delete_date | datetime | YES | | NULL | |
| allow_count | int(11) | YES | | NULL | |
| borrow_count | int(11) | YES | | NULL | |
| pause_date | datetime | YES | | NULL | |
| user_state | varchar(20) | YES | | NULL | |
| password | varchar(20) | YES | | NULL | |

**图 3-13 DESC 语句执行结果**

从查询结果可以看出，reader_name 字段的数据类型为 VARCHAR(20)。然后，执行 ALTER TABLE 语句修改字段数据类型。执行结果如图 3-14 所示。

```
mysql> ALTER TABLE reader MODIFY reader_name varchar(30);
Query OK, 6 rows affected (0.35 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

**图 3-14 ALTER TABLE 语句执行结果**

代码执行完毕，结果显示修改成功。为检验 reader_name 字段的数据类型是否已经改变，使用 DESC 语句重新查看 reader 表。查看结果如图 3-15 所示。

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| reader_id | varchar(20) | NO | PRI | NULL | |
| in_user_id | varchar(20) | NO | | NULL | |
| out_user_id | varchar(20) | YES | | NULL | |
| reader_type_code | varchar(10) | YES | MUL | NULL | |
| reader_name | varchar(30) | YES | | NULL | |
| reader_tel | varchar(20) | YES | | NULL | |
| reader_email | varchar(50) | YES | | NULL | |
| create_date | datetime | YES | | NULL | |
| delete_date | datetime | YES | | NULL | |
| allow_count | int(11) | YES | | NULL | |
| borrow_count | int(11) | YES | | NULL | |
| pause_date | datetime | YES | | NULL | |
| user_state | varchar(20) | YES | | NULL | |
| password | varchar(20) | YES | | NULL | |

**图 3-15 修改后的 reader 表结构**

查询结果显示，reader_name 字段的数据类型已经变为 VARCHAR(30)，修改完毕。

### 2. 大文本(BLOB)字段及在图书管理系统中的应用

如果表里有大文本字段，当需要对这个字段进行查询时，一般这种长字段是很难进行索引的。下面介绍合成索引的方法。

(1) 创建表 test，id、contest、hash_value 字段，类型分别为 INT、BLOB、VARCHAR(40)。

```
MySQL>create table test (id int,context blob,hash_value varchar(40));
```

(2) 在 test 中插入测试数据,其中 hash_value 用来存放 context 列的 md5 散列值,其中 repeat()为字符串函数。

```
MySQL>insert into test values(1,repeat('shanghai',2),md5(context));
MySQL>insert into test values(1,repeat('sasdffdd',10),md5(context));
MySQL>insert into test values(1,repeat('nanjing2002',2),md5(context));
```

(3) 如果要查询 context 值为 nanjing2002nanjing2002 的记录,可以通过相应的散列值来查询。

```
MySQL>select * from test where hash_value=md5(repeat('nanjing2002',2));
```

上面的例子展示了合成索引的用法。由于这种技术只能用于精确匹配,在一定程度上减少了 I/O,从而提高了查询效率。如果需要对 BLOB 或者 TEXT 进行模糊查询,MySQL 提供了前缀索引。

有时候需要对一个很长的字符串的列建立索引,这会导致索引变大并且变慢。一种策略是建立哈希索引,而另一种策略就是建立前缀索引。对这列的前几个字符建立索引,而不是全部。索引的选择性是指索引中不同的值的个数和所有行数的比率。使用高选择性的索引比较好,因为能够过滤掉更多的行。前缀索引如果选择性足够好,就可以获得很高的性能。如果使用 BLOB 或者 TEXT,或者非常长的 VARCHAR 列,就必须定义前缀索引,因为 MySQL 不允许在全部长度上建立索引。

**3. 二进制数据字段**

二进制字段是指任何二进制类型的文件,都可以保存在数据库的字段中,如图像文件或者 Word 文档文件等。

对二进制字段,在 Access 数据库中一般设计为 OLE 对象,而在 SQL Server 中则设计为 Image 等类型。MySQL 把 BLOB 数据列和带 BINARY 属性的 CHAR 和 VARCHAR 数据列中的数据当作二进制值。

**4. 自定义类型字段**

定义表的各个属性而难以判断需要指明的数据的类型及长度时,可以使用自定义类型字段。将数据类型定义为 mediumtext 类型,一个 BLOB 或是 TEXT 列,最大长度为 65535($2^{16}-1$)。

### 3.2.3 表联系的设计

在第 2 章中,已经介绍了建立表逻辑关系时对联系的设计。在设计物理表的联系时,不可避免要对表的结构做一些修改,这些改进将会使得表的结构更完善,并且使冗余降到最小。合理的联系能够确保数据库完整性的实施。表 3-4 为已经确定好的映射数量表,下面以此为基础介绍物理表联系的设计。

表 3-4 实体映射数量表

| | 读者 | 管理员 | 图书 | 读者类型 | 读者单位 |
|---|---|---|---|---|---|
| 读者 | | | n | 1 | 1 |
| 管理员 | | | | | |
| 图书 | n | | | | |
| 读者类型 | n | | | | |
| 读者单位 | 1 | | | | |

### 1. 一对一联系

在一对一联系中，通常习惯将一对表的关系设计成为一个父表和一个子表。子表依赖于父表，父表中的每一条数据，都与子表中的一条数据对应。首先父表中必须有一条数据，使子表中的一条数据与之对应，如“读者”与“读者单位”。一般情况下以读者表作为父表，读者单位表作为子表。首先读者表中必须存在一条某位读者的数据，读者单位表中才能有一条数据与之对应；如果该读者数据被删除，那么读者单位表中相对应的数据就没有实际意义了，应当一起删除。

为了保证子表与父表之间数据的一一对应，通常将父表的主键放在子表中作为外键，并且在子表中，该外键通常也同时作为主键，这样就能确保数据的唯一性及一一对应。如图 3-16 所示，一个读者在注册时填写的资料只能有一个读者单位(或不填写读者单位，除“借书证号”字段，其他字段为空)，所以，它们永远是一一对应的关系，两个表具有相同的主键。

当然，在一对一联系中，将外键作为主键的情况也并不是不变的。例如图 3-17 中的例子，经理表作为父表，部门表作为子表(如果需要，部门表作为父表也是可以的。这个例子可以根据具体环境灵活变动，但是联系设计的原理是一样的)。虽然经理表与部门表是一一对应的关系，一个部门只有一个经理且一个经理只对应一个部门，然而在一个公司里，部门是数量有限并且事先部署好的，所以可以单独定义一个“部门 ID”作为主键。这样做的好处是，当一个经理辞职的时候，暂时没有人来接管该部门，但是该部门数据不受影响(因为外键可以为空)。

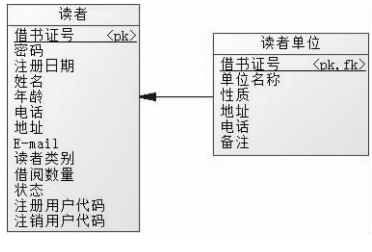


图 3-16 一对一联系

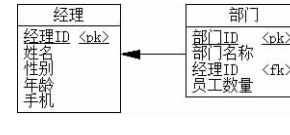


图 3-17 一对一联系，外键不作为主键

### 2. 一对多联系

一对多联系的设计可以参考一对一联系。联系的“一”端作为父表,“多”端作为子表。从父表中取其主键放到子表中做外键。如“读者类型”与“读者”是一对多的关系,一个读者类型可以有多个读者共享,而一个读者在同一时间内有且只有一个读者类型,所以将读者类型表作为父表,读者表作为子表,取读者类型表的主键“读者类型 ID”放到读者表中,如图 3-18 所示。

### 3. 多对多联系

建立多对多联系时需要使用到“联系表”,此“联系表”即在数据库逻辑设计阶段 E-R 模型中实体之间的联系转变而来。如读者与图书之间是多对多联系,它们的联系是“借阅”,“联系”也可以有自己的属性,同样可以产生一个独自的数据表,称之为联系表或连接表(注:第 3 章 E-R 图中“借阅”实为读者、图书、图书管理员三个实体之间的联系,此处简化,方便读者理解)。图 3-19 是使用“借阅”联系自有的属性产生的一个联系表,主键“借阅 ID”只作为主键,无实际意义。

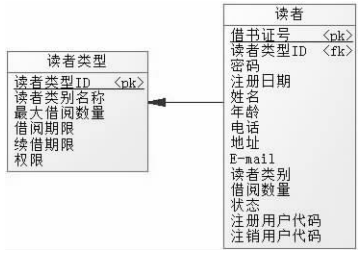


图 3-18 一对多联系

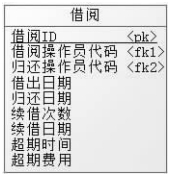


图 3-19 借阅联系表

多对多联系的设计过程是,从需要建立联系的两个表中取出主键,即读者表中的借书证号与图书表中的图书 ID,放入借阅表中成为外键。这里对借阅表的设计有两种方案:第一,使用引入的外键作为复合主键,即借书证号与图书 ID 作为复合主键。但在这里有一个问题,假如同一个读者重复借阅同一本书,则会产生相同的主键,违背了主键唯一的原则,需要加入新的属性共同组成复合主键,这就增加了数据库设计以及使用的难度。图 3-20 是设计完成的多对多联系。

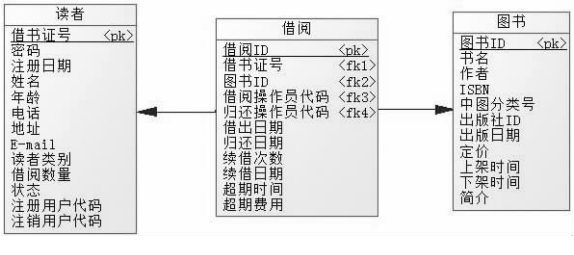


图 3-20 多对多联系

上述步骤建立了物理表的联系，下面还需要进一步调整表的结构与联系，确保每个表与它们之间的联系都符合要求。

理想的物理表有以下特点：

- 只有一个主键；
- 只代表单个主题，可以是具体的事物（读者）或事件（借阅）；
- 不包含多个成分或多值字段；
- 不包含计算字段；
- 数据冗余最小。

**4. 设计联系的规则**

联系的规则决定了删除联系的表中数据的方式。在删除一个父表的数据时，与其相关联的子表数据必定会受到影响，所以制定删除规则对于实施数据库完整性是至关重要的。根据数据库管理系统的执行动作，删除规则有如下 5 种：

（1）拒绝。接收到删除命令时，数据库管理系统物理上不删除父表中的数据，而是以特定标识符标志其为“删除”状态。例如需要删除某图书管理员数据时，一般情况下为了以后方便查找该员工的工作情况，物理上并不删除该数据，而是标志其为“已删除”。

（2）限制。接收到删除命令时，如果子表存在关联数据，则数据库管理系统不删除父表数据，除非首先删除子表数据。例如图书管理系统升级需要删除某些以后用不到的“读者类型”数据，则数据库管理系统首先需要检查目前是否还有读者正在使用这些读者类型，假如有，则不能删除相应的读者类型。

（3）级联。接收到删除命令后，数据库管理系统在删除父表数据的同时，自动查找删除相关联的子表数据。例如在删除读者数据时，同时删除该读者所有借阅记录。

（4）置空。接收到删除命令后，数据库管理系统在删除父表数据的同时，将子表中相关联的数据的外键设为空值，前提是该外键不是主键且允许为空。例如某出版社不存在以后，将图书表中的“出版社编号”设为空。

（5）设置默认值。接收到删除命令后，数据库管理系统在删除父表数据的同时，将子表中相关联的数据的外键设为一个“默认值”，用户从该“默认值”能够获知该数据对应的父表数据已经被删除。

## 3.2.4 约束的设计

前面已经提到过，数据的完整性是指数据的正确性和一致性，可以通过定义完整性约束来实现。完整性约束是一种规则。完整性约束存在数据字典中，不占用任何数据库空间。用户可以自己决定何时使用完整性约束。使用约束时，可以增强数据的完整性，但是也可以通过规则、索引、触发器等实现数据的完整性。完整性约束按照使用范围分为两类：行级和表级，处理机制是一样的。行级约束放在列后，表级约束放在表后，多个列共用的约束放在表后。

完整性约束分为 6 种：主键约束（PRIMARY KEY 约束）、外键约束（FOREIGN KEY 约束）、唯一约束（UNIQUE 约束）、非空约束（NOT NULL 约束）、检查约束（CHECK 约束）和默认值约束（DEFAULT 约束）。

设计完整性约束前，首先介绍关系模式的完整性分类。

- 实体完整性：实体完整性就是将表中的每一行看作一个实体并且保证实体的唯一性。实体完整性是针对表而言的，故它保证的是表的标示列或主键的完整性。可以通过创建 PRIMARY KEY 约束、UNIQUE 约束、唯一索引以及列的 IDENTITY 属性来实施实体完整性。
- 域完整性：域完整性是指保证列的输入值在有效范围内，包括正确的数据类型、格式和有效的取值范围。域完整性可以通过 FOREIGN KEY 约束、CHECK 约束、DEFAULT 约束、NOT NULL 约束和规则来实现。
- 参照完整性：参照完整性用于保证相关联的表数据的一致性，一般来说它主要通过 PRIMARY KEY 约束和 FOREIGN KEY 约束来实现。参照完整性避免了修改一个表的值后，其相关联的表存在无效的值。
- 用户定义完整性：用户定义完整性不同于前面三种完整性，而是由用户不属于其他任何完整性分类的特定业务规则，也可以简单说成是一种强制的数据约束。

### 1. 主键约束（PRIMARY KEY 约束）

主键表示唯一的标识，本身不能为空。例如图书管理系统中的“借书证号”，它能够唯一标识每个读者，并且在任何情况下都不可能存在一个没有“借书证号”的读者数据。通过将“借书证号”指定为主键，可以强制读者表的实体完整性。

在使用 PRIMARY KEY 约束时，该列的“是否为空”属性必须定义为 NOT NULL，也就是说有主键的那一列，不能为空。由于 PRIMARY KEY 约束确保唯一数据，所以经常用来定义标识列。另外当 PRIMARY KEY 约束由另一张表的 FOREIGN KEY 约束引用时，不能删除 PRIMARY KEY 约束；要删除它，必须先删除 FOREIGN KEY 约束。通常建立一列约束时，称之为列级 PRIMARY KEY 约束，应用于多列时，称之为表级 PRIMARY KEY 约束。

**注意**：在 Microsoft SQL 数据库中，建立 PRIMARY KEY 约束除了保证表的数据完整性的同时还为主键创建唯一索引以强制数据的唯一性。默认情况下所建立的索引为簇索引。该索引只能通过删除 PRIMARY KEY 约束或其相关表的方法来删除，而不能使用 DROP INDEX 语句删除。

**例 3-3** PRIMARY KEY 约束实例如下：

```
#创建读者表并将给借书证号添加 PRIMARY KEY 约束
create table 读者
(  借书证号   varchar(20) not null,
   用户名   varchar(20),
   密码   varchar(16),
   primary key (借书证号));
```

PRIMARY KEY 约束的功能主要体现在使用插入（Insert）添加数据时，主键为空或者重复时数据都是添加不进去的。

### 2. 外键约束（FOREIGN KEY 约束）

FOREIGN KEY 约束是在两个相关联的表中进行约束操作。通常与 PRIMARY

KEY 约束或者 UNIQUE 约束同时使用。在使用 FOREIGN KEY 约束时,需要注意以下几点:

- 一个表最多只能参照 253 个不同的数据表,每个表也最多只能有 253 个 FOREIGN KEY 约束。
- FOREIGN KEY 约束不能应用于临时表。
- 在实施 FOREIGN KEY 约束时,用户必须至少拥有被参照表中参照列的 SELECT 或者 REFERENCES 权限。
- FOREIGN KEY 约束同时也可以参照自身表中的其他列。
- FOREIGN KEY 约束,只能参照本身数据库中的某个表,而不能参照其他数据库中的表。跨数据库的参照只能通过触发器来实现。

**例 3-4** FOREIGN KEY 约束实例如下:

```
#创建一个借阅表,并参照已有的读者、图书和图书管理员表添加 4 个 PRIMARY KEY 约束
create table 借阅
(  借阅 ID  bigint not null auto_increment,  #自增列主键
   借书证号  varchar(20),
   图书 ID  bigint,
   借阅管理员 ID  bigint,
   归还管理员 ID  bigint,
   primary key (借阅 ID));
alter table 借阅 add constraint FK_Reference_1 foreign key (借书证号)
    references 读者 (借书证号) on delete restrict on update restrict;
alter table 借阅 add constraint FK_Reference_2 foreign key (图书 ID)
    references 图书 (图书 ID) on delete restrict on update restrict;
alter table 借阅 add constraint FK_Reference_3 foreign key (借阅管理员 ID)
    references 图书管理员 (管理员 ID) on delete restrict on update restrict;
alter table 借阅 add constraint FK_Reference_4 foreign key (归还管理员 ID)
    references 图书管理员 (管理员 ID) on delete restrict on update restrict;
```

这里需要重点注意 on delete 和 on update 后面的取值的含义,MySQL 与 Microsoft SQL Server 不同。MySQL 可能的取值有 4 种: No Action、Cascade、Set Null 和 Restrict。

(1) 为 on delete 之后的取值时,含义如下:

- 取值为 No Action 或者 Restrict 时,则在父表中删除对应数据时,首先检查该数据是否有对应外键。如果有则不允许删除,即联系设计规则中的"拒绝"。
- 取值为 Cascade 时,则在父表中删除对应数据时,首先检查该数据是否有对应外键。如果有则删除该外键在子表中相应的数据,即联系设计规则中的"级联"。
- 取值为 Set Null 时,则在父表中删除对应数据时,首先检查该数据是否有对应外键。如果有则设置子表中该外键值为 null(前提是该外键允许取值为 null),即联系设计规则中的"置空"。

(2) 为 on update 之后的取值时,含义如下:

- 取值为 No Action 或者 Restrict 时,则在父表中更新数据时,首先检查该数据是

否有对应外键。如果有则不允许更新，即联系设计规则中的“拒绝”。

- 取值为 Cascade 时，则在父表中更新数据时，首先检查该数据是否有对应外键。如果有则同时更新外键在子表中相应的数据，即联系设计规则中的“级联”。
- 取值为 Set Null 时，则在父表中更新数据时，首先检查该数据是否有对应外键。如果有则设置子表中该外键值为 null(前提是该外键允许取值为 null)，即联系设计规则中的“置空”。

**注意**：在 Microsoft SQL Server 数据库中，该取值有 No Action、Cascade、Set Null、Set Default，其中 Set Default 表示当父表数据被更新或删除时，子表中的相应数据被设置成默认值(前提是子表中的相应列设置有默认值)。

### 3. 唯一约束(UNIQUE 约束)

在一个表中只可以建立一个 PRIMARY KEY 约束，而其他列不希望重复的话可以使用 UNIQUE 约束，所以该约束应用于表中的非主键列。UNIQUE 约束保证一列或者多列的数据完整性，确保这些数据不会有相同的值。它与 PRIMARY KEY 约束的不同之处在于：UNIQUE 约束可以建立多个，而 PRIMARY KEY 约束在一个表中只能有一个。对于一列的 UNIQUE 约束，称之为列级 UNIQUE 约束。对于多列的 UNIQUE 约束，称之为表级 UNIQUE 约束。

例如，读者表中“用户名”可以作为读者自己设定的名称，方便用于网站的登录操作，显然“用户名”不能取相同的值但又不是作为主键，这时可以在此列上建立 UNIQUE 约束，确保不会有重复的用户名。

**例 3-5** UNIQUE 约束实例如下：

```
#创建一个读者表,借书证号添加 PRIMARY KEY 约束,用户名添加 UNIQUE 约束
create table 读者
(  借书证号   varchar(20) not null,
   用户名   varchar(20),
   密码   varchar(16) not null,
   primary key (借书证号),
   key AK_UQ_UserName (用户名));
```

### 4. 非空约束(NOT NULL 约束)

NOT NULL 约束用法比较简单，比如主键和密码这样的列一般都要设置为 NOT NULL。

### 5. 检查约束(CHECK 约束)

CHECK 约束的主要作用是确保输入一个列的内容合法，从而保证数据的域完整性。一个表允许建立多个 CHECK 约束，CHECK 约束中可以包含搜索条件，但不能包含子查询。同样可以为表中的一个列建立一个或多个 CHECK 约束，但是如果使用 CREATE TABLE 语句，只能为每个列建立一个 CHECK 约束。CHECK 约束被应用于多列时，必

须被定义为表级 CHECK 约束。

在表达式中,可以输入搜索条件,条件中可以包括 AND 或者 OR 一类的连接词。列级 CHECK 约束只能参照被约束列,而表级 CHECK 约束则只能参照表中列,它不能参照其他表中列。

**例 3-6** CHECK 约束实例如下:

```
#创建一个读者表,为电话添加 CHECK 约束保证全为数字
create table 读者
(  借书证号  varchar(20) not null,
   用户名  varchar(20),
   密码  varchar(16) not null,
   电话  char(20),
   primary key (借书证号),
   key AK_UQ_UserName (用户名),
   check (电话 like '[0-9]* ') );
```

**6. 默认值约束(DEFAULT 约束)**

DEFAULT 约束的功能是用户在插入数据但没有为列提供数据值时,系统会将默认值赋给该列。例如读者表中的密码列,在管理员新建一个读者数据时,系统自动填上六个零“000000”作为初始密码,以便读者进行首次登录。DEFAULT 约束所提供的默认值可以为常量、函数、系统零进函数、空值(NULL)等。

**例 3-7** DEFAULT 约束实例如下:

```
#创建一个读者表,为密码和注册日期添加 DEFAULT 约束,密码默认值为'000000',注册日期默认值为当天日期,使用 getdate() 函数获得
create table 读者
(  借书证号  varchar(20) not null,
   用户名  varchar(20),
   密码  varchar(16) not null default '000000',
   注册日期  date default 'getdate()',
   primary key (借书证号),
   key AK_UQ_UserName (用户名));
```

**注意**:每一列只能有一个默认约束,约束表达式不能参照表中的其他列和其他表、视图或存储过程。

## 3.2.5 使用 PowerDesigner 设计物理表

在 PowerDesigner 中建立物理模型一般有以下 4 种途径:

- 重新设计表来建立物理模型;
- 由设计好的概念模型生成物理模型;
- 由设计好的逻辑模型生成物理模型;