项目 3

微营销小程序系统数据库创建和管理

CHAPTER 3

知识探索与学习信念

有了 MySQL 数据库管理平台的支撑，用户就可以创建和管理自己的项目数据库。

MySQL 是关系型数据库，以关系二维表描述实体对象并存储数据记录。每创建一个用户数据库就会在 MySQL 数据目录（Data）中生成一个子目录，而数据表则以文件形式存放在此子目录中。如用户数据表、商品数据表、订单数据表等分别以表名生成对应的数据文件名。通过数据库统一集中管理数据表，提高了数据的一致性、可靠性、易维护性。

本项目讲解如何启动和停止 MySQL 服务，对 MySQL 自带的系统数据库的用途作了介绍，并对数据库物理存储结构以及数据文件生存和存储方式进行了详细描述。通过实训任务掌握如何使用命令语句创建和管理用户数据库，如何使用图形化软件工具管理数据库，并对 MySQL 的几种数据库引擎作了对比介绍。

通过本项目学习，建立对数据库的物理空间概念。要养成持续良好的学习习惯和严谨的治学态度，按时高质量完成实训任务，深入掌握 MySQL 数据库管理知识，为后续项目开发打好基础，立志成为一名合格的数据库工程师。

本章技能知识导图如图 3-1 所示。

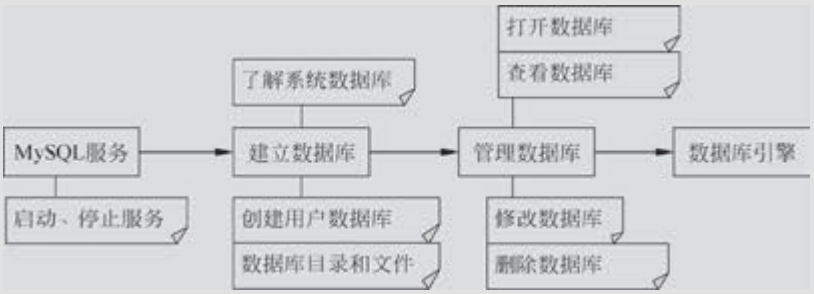

图 3-1　数据库创建和管理知识导图

素质目标

(1) 开拓进取,独立完成任务。

(2) 培养探源溯流的能力,清楚 MySQL 数据库管理系统的运行机制、数据库物理存储形式、存储引擎的作用以及系统数据库存在的意义。

(3) 培养深入钻研、思考问题和理解问题的能力。

学习目标

(1) 了解何为 MySQL 服务,掌握启动和停止 MySQL 服务的方法。

(2) 熟练掌握 MySQL 的远程登录连接指令、查看数据库的指令。

(3) 了解 MySQL 管理系统中系统数据库的用途。

(4) 了解 MySQL 数据存储目录及数据文件存储形式。

(5) 学会使用 MySQL 命令创建和管理用户项目数据库。

(6) 学会使用图形化管理工具创建和管理数据库。

(7) 了解 MySQL 数据库存储引擎及作用。

重点与难点

- 重点:MySQL 服务,数据库连接查看指令,MySQL 系统数据库,创建和管理用户数据库。
- 难点:MySQL 系统数据库,用户数据库,数据库引擎。

3.1 启动和停止 MySQL 服务

3.1.1 任务场景

由于教学计算机不仅用于数据库课程教学实训，还用于其他软件和网络专业课程的教学，安装了种类繁多的软件工具，所以为了减少内存资源占用，可能管理员将 MySQL 服务设计为手动启动，而不是开机自动启动。如果 MySQL 服务没有正常启动，连接 MySQL 数据库就会出错，例如连接本机 Windows 系统上安装的 MySQL 数据库，会出现如图 3-2 所示的错误信息。

图 3-2 连接 MySQL 数据库出错

所以首先了解如何启动和停止 MySQL 服务。

将 MySQL 服务正常启动后，进行连接，如果出现如图 3-3 所示的 mysql 提示符，则表示连接成功。

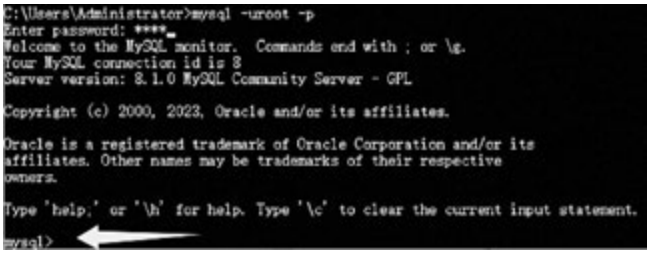

图 3-3 通过 Windows 命令窗口成功连接 MySQL 数据库示意图

3.1.2 任务 1 在 Windows 系统中启动和停止 MySQL 服务

在 Windows 系统中，要能够熟练掌握以下两种方法启动和停止 MySQL 服务。

方法一，通过进入“控制面板”→“管理工具”→“服务”功能窗口查看、启动或停止 MySQL 服务。

在图 3-4 所示的“服务”窗口中的 MySQL81 即是本教程安装的 MySQL 服务名称。选中此服务，通过单击左侧的“启动”“停止”等功能进行操作，或右击菜单选择“启动”“停止”等功能操作。

方法二，以命令行方式启动和停止 MySQL 服务。

(1) 在 Windows 的 cmd 窗口输入如下命令行语句可以启动 MySQL 服务。

```
net start MySQL81
```

启动成功的信息如图 3-5 所示。

(2) 可在 cmd 窗口中输入如下命令行语句停止 MySQL 服务。

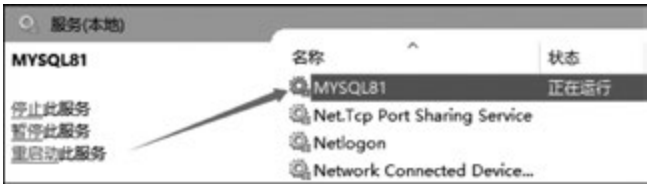

图 3-4　通过 Windows 系统的服务窗口启动或停止 MySQL 服务

图 3-5　MySQL 服务成功启动信息

```
net stop MySQL81
```

也可以使用 MySQLadmin 命令停止服务，命令语句如下：

```
MySQLadmin -u root -p shutdown
```

3.1.3　任务 2　在 Debian 系统中启动和停止 MySQL 服务

在 Debian 系统中，需要 root 账号权限才能停止服务，所以用普通账号登录后，要切换到 root 账号进行实验操作。注意，在 Debian 系统中命令用小写字母。

先用 service 命令查看 MySQL 服务是否启动：

```
service mysql status
```

如果已启动，则状态信息如图 3-6 所示。

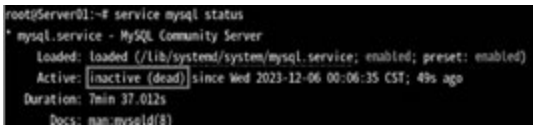

图 3-6　MySQL 服务启动的状态信息

1. 启动 MySQL 服务的命令语句

```
service mysql start
```

2. 停止 MySQL 服务的命令语句

```
service mysql stop
```

MySQL 服务启动或停止成功后，都不像旧版本那样会返回已启动或已停止的信息。要再次使用命令查看其状态。

Debian 系统中也可以使用 mysqladmin 命令停止服务，命令语句如下：

```
mysqladmin -u root -p shutdown
```

3.1.4 任务 3 连接访问 MySQL 数据库

MySQL 服务启动后，才可以进行连接操作。

利用 mysql 命令演示如何利用 Debian 系统终端工具连接本地 MySQL 数据库和连接远程 MySQL 数据库。

本机连接：

```
mysql -u root - p
```

远程连接：

```
mysql -h 192.168.192.128 -P 3306 -u root - p
```

远程连接 MySQL 除了提供账号和密码外，还需要指明服务器 IP 地址和端口号。

连接成功后，可以通过如下命令语句显示 MySQL 系统中已有的数据库：

```
show databases;
```

以超级管理员 root 账号连接服务器的执行结果如图 3-7 所示，能看到所有数据库。

图 3-7 显示 MySQL 服务器中的数据库信息

图 3-7 列出了新安装的 MySQL 8.0 数据库管理系统中的四个系统数据库（或架构模式），分别是 information_schema、mysql、performace_schema 和 sys。

3.2 介绍 MySQL 数据库架构

在 MySQL 中，模式就是数据库。MySQL 数据库管理系统中存在两种类型的模式（数据库）：系统自带的平台支撑数据库和用户自己创建的数据库。系统数据库组成架构如图 3-8 所示。

3.2.1 任务 1 了解 information_schema 数据库

information_schema 用于存储有关 MySQL 服务器维护的所有其他数据库的信息（例如，提供数据库、表的名称，列的数据类型、访问权限等）。information_schema 数据库包含

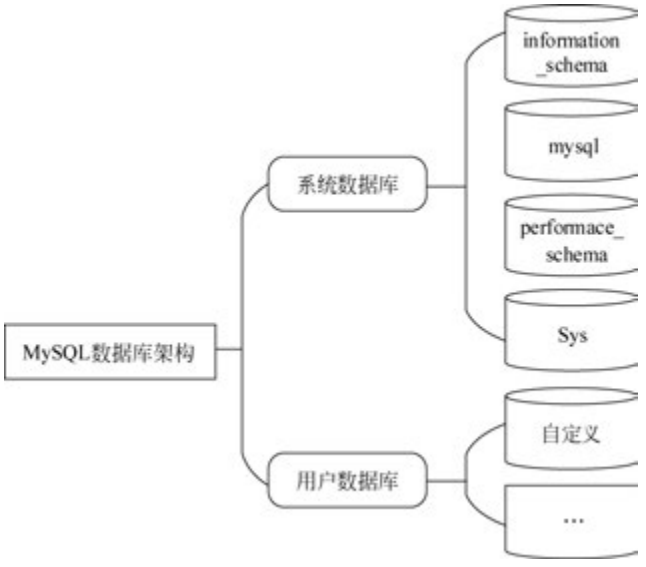

图 3-8　MySQL 数据库架构图

多个只读表，它们实际上是视图，而不是基表，因此没有与它们关联的文件，并且无法为其设置触发器。此外，没有 information_schema 名称的数据库目录。

可以使用 USE 语句将 information_schema 作为当前数据库，但只能读取其表中的内容，而不能对其执行 INSERT、UPDATE 或 DELETE 操作。

【例 3.1】 利用 information_schema 数据库的 tables 表查看 mysql 架构(数据库)中创建的所有表的信息。

通过 Navicat 工具远程连接 Debian 系统中的 MySQL 服务器，执行以下查询语句：

```
SELECT * FROM information_schema.tables WHERE table_schema = 'mysql';
```

打开 Navicat 工具，双击“information_schema”数据库，然后单击“查询”菜单，再单击下方出现的“新建查询”按钮，打开查询编辑器，如图 3-9 所示。

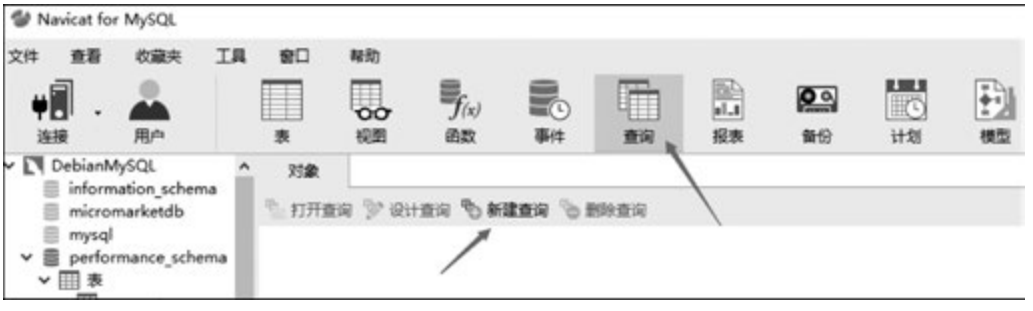

图 3-9　使用 Navicat 工具的查询编辑器

在查询编辑器中输入命令语句，执行查询语句的结果如图 3-10 所示。

图 3-10 列出了系统数据库 mysql 的所有表信息。同样，可以查询其他数据库的表信息。

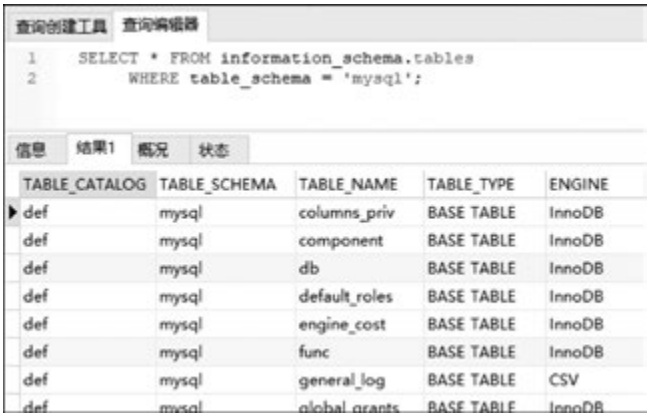

图 3-10 查询编辑器中执行查询语句输出结果

3.2.2 任务 2 了解 MySQL 数据库

MySQL 数据库中包含存储 MySQL 服务器运行时所需信息的表。MySQL 模式包含存储数据库对象元数据的数据字典表,以及用于其他操作目的的系统表。

MySQL 系统表和数据字典表驻留在 MySQL 数据目录中名为 mysql.ibd 的单个 InnoDB 表空间文件中。从 Debian 系统终端进入“/var/lib/mysql”目录能够看到此文件,如图 3-11 所示。

图 3-11 在 Debian 系统中查看 mysql 表物理存储文件

数据字典表是不可见的,它们无法使用 SELECT 语句读取、不会出现在 SHOW TABLES 的输出中、不会在 information_schema.tables 表中列出等。比如用 SELECT 读取 MySQL 数据库的字典表 schemata 会出现如图 3-12 所示错误信息。

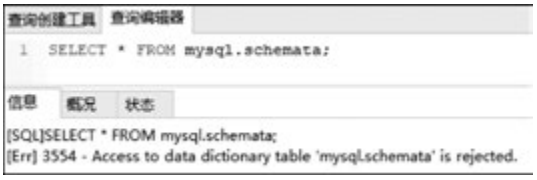

图 3-12 拒绝使用 SELECT 语句查询数据字典表

不过,大多数情况下都有相应的 information_schema 表可以查询。information_schema 提供了 MySQL 公开数据字典元数据的视图,从相应的 information_schema 表中读取能获得正确信息,SQL 语句如下:

```
SELECT * FROM information_schema.schemata;
```

3.2.3 任务 3 了解 performance_schema 数据库

performance_schema 用于收集数据库服务器性能参数,此模式用于在低级别监控

MySQL 服务器的执行情况。性能模式具有以下特征：

(1) 提供一种在运行时检查服务器内部执行情况的方法。它是使用 performance_schema 存储引擎和 performance_schema 数据库实现。性能模式主要关注性能数据，与 information_schema 不同，后者用于元数据检查。

(2) 性能模式监视服务器事件。一般来说，事件可以是函数调用、等待操作系统、SQL 语句执行的一个阶段(例如，解析或排序)或者整个语句或语句组。事件收集提供对有关服务器和多个存储引擎的同步调用(例如，互斥锁)、文件和表 I/O、表锁等信息的访问。

(3) 性能模式事件特定于 MySQL 服务器的给定实例。性能模式表被视为服务器本地表，对它们的更改不会复制或写入二进制日志。

(4) performance_schema 存储引擎使用服务器源代码中的“检测点”收集事件数据。

(5) 可以通过 SQL 语句更新 performance_schema 数据库中的表来动态修改 performance_schema 配置。配置更改会立即影响数据收集。

(6) 性能模式中的表是内存表，不使用持久磁盘存储。这些内容在服务器启动时开始重新填充，并在服务器关闭时丢弃。

注意：MySQL 的 sys 架构是一组对象，可方便地访问性能架构收集的数据。MySQL 默认情况下安装了 sys 架构。有关 sys 数据库在 3.2.4 节说明。

3.2.4　任务 4　了解 sys 数据库

sys 数据库(架构)是一组帮助 DBA 和开发人员解释性能模式收集的数据的对象。sys 模式对象可用于典型的调优和诊断用例。此架构中的对象包括：视图、存储过程、存储函数。

要完全访问 sys 架构，用户必须具有以下权限：

(1) 对所有系统表和视图具有 SELECT(查询)权限。

(2) 对所有系统存储过程和函数具有 EXECUTE(执行)权限。

(3) 如果要对其进行更改，则对 sys_config 表具有 INSERT(插入)和 UPDATE(更新)权限。

关于用户权限设置技术，将在项目 9 学习。实验过程中使用 root 账号操作，所以不存在权限限制问题。例 3.2 为 sys 数据库的几个应用。

【例 3.2】 利用 sys 数据库对象查看用户访问服务器的会话信息。

打开在 Debian 系统终端，首先登录 MySQL 服务器，使用 USE 命令将 sys 设置为当前数据库：

```
USE sys;
```

然后使用 SELECT 命令查询 session 视图，即可获得用户访问信息：

```
SELECT * FROM session;
```

执行结果如图 3-13 所示。

在以上操作语句中，挑选了用户访问过哪些数据库(db)、访问的用户名及 IP(user)和访问时长(time)三个字段的信息。

sys 架构包含许多视图，以各种方式汇总性能架构表。例如，host_summary_by_file_io 视图总结了按主机分组的文件 I/O，并显示从皮秒转换为更易读的值（带单位）的延迟。查询视图显示结果如图 3-14 所示。

```
mysql> SELECT user,db,time FROM sys.session;
+-------------------+--------------------+------+
| user              | db                 | time |
+-------------------+--------------------+------+
| root@192.168.13.1 | mysql              | 5013 |
| root@192.168.13.1 | NULL               | 4746 |
| root@192.168.13.1 | information_schema | 4745 |
| root@192.168.13.1 | performance_schema | 1353 |
| root@192.168.13.1 | sys                |  217 |
| root@192.168.13.1 | sys                |  198 |
| root@192.168.13.1 | sys                |  183 |
| root@192.168.13.1 | sys                |  152 |
| root@192.168.13.1 | sys                |  152 |
| root@192.168.13.1 | performance_schema |   72 |
| root@192.168.13.1 | performance_schema |   18 |
| root@localhost    | NULL               |    0 |
+-------------------+--------------------+------+
12 rows in set (0.06 sec)
```

图 3-13　查看用户访问 MySQL 服务器的信息

```
mysql> SELECT * FROM host_summary_by_file_io;
+--------------+------+------------+
| host         | ios  | io_latency |
+--------------+------+------------+
| background   | 1826 | 535.40 ms  |
| 192.168.13.1 |  151 | 263.14 ms  |
| localhost    |    2 | 90.40 us   |
+--------------+------+------------+
3 rows in set (0.01 sec)
```

图 3-14　通过 host_summary_by_file_io 视图查看各主机 I/O 性能

如果要检查 sys 架构中对象（视图、函数等）的定义，则需要使用 SHOW 命令语句查询。

【例 3.3】 利用 sys 数据库检查 session（会话视图）和 format_bytes()函数的定义。

由于在终端不能清晰查看视图、函数的定义信息，因此通过 Navicat 工具远程查看。

打开 Navicat 工具，双击“sys”数据库，单击“查询”菜单，再单击下方出现的“新建”按钮，打开查询编辑器，然后分别输入执行以下两个命令语句：

```
SHOW CREATE VIEW session;
SHOW CREATE FUNCTION format_bytes;
```

执行语句，可以查看创建视图或函数的名称、SQL 语句等信息，如图 3-15 所示。

图 3-15　使用 show 语句查看数据库中视图或函数的定义

3.2.5　任务 5　了解 Windows 系统中数据库目录及文件

在 MySQL 数据库系统中创建新的用户数据库，同样会在 MySQL 的 data 目录下生成一个与数据库名称相同的子目录，此目录用来存放数据库中的数据表文件。MySQL 8.0 不支持手动在 data 目录下创建数据库目录。创建数据库时，让服务器管理目录和其中的文

件。直接操作数据库目录和文件可能会导致不一致和意外结果。MySQL 对数据库数量没有限制，但底层文件系统(操作系统)可能对目录数量有限制。

在 MySQL 8.0 之后，以 InnoDB 引擎创建的数据表结构信息存储在“.ibd”或“.ibdata”文件中，而在 MySQL 8.0 之前的版本是“.frm”文件后缀名。以 MyISAM 引擎创建的表结构文件后缀名为“.myd”。

通过 InnoDB 引擎创建的数据库，数据存储方式可以配置成共享表空间存放数据，也可以配置成独享表空间存放数据。

共享表存储方式，使用“.ibdata“文件来存放表数据，可以配置为所有表共同使用一个(或多个)“.ibdata”文件。

独享表空间存储方式使用“.ibd”文件来存放表数据，且每个表就是一个“.ibd”文件。

在 Windows 系统中，数据库的默认存放目录为：C:\ProgramData\MySQL\MySQL Server 8.1\Data。

3.2.6　任务 6　了解 Debian 系统中的数据库目录及文件

登录 MySQL 服务器，使用以下命令可以查看 Debian 系统中 MySQL 数据库安装的相关目录：

```
show variables like '%dir%';
```

Debian 系统中的数据库目录及文件结构如图 3-16 所示。

```
mysql> show variables like '%dir%';
+----------------------------------------+--------------------------------+
| Variable_name                          | Value                          |
+----------------------------------------+--------------------------------+
| basedir                                | /usr/                          |
| binlog_direct_non_transactional_updates | OFF                            |
| character_sets_dir                     | /usr/share/mysql-8.0/charsets/ |
| datadir                                | /var/lib/mysql/                |
| innodb_data_home_dir                   |                                |
| innodb_directories                     |                                |
| innodb_doublewrite_dir                 |                                |
| innodb_log_group_home_dir              | ./                             |
| innodb_max_dirty_pages_pct             | 90.000000                      |
| innodb_max_dirty_pages_pct_lwm         | 10.000000                      |
| innodb_redo_log_archive_dirs           |                                |
| innodb_temp_tablespaces_dir            | ./#innodb_temp/                |
```

图 3-16　Debian 系统中的数据库目录及文件结构图

其中，datadir 对应的值“/var/lib/mysql/”就是数据库主目录。

退出 mysql，返回 Debian 终端，列表查看数据库主目录中创建的所有与数据库名同名的目录，以及一些生成的二进制备份文件等。操作命令语句如下：

```
ls /var/lib/mysql/
```

进入一个数据库目录，比如 sys 目录，可以看到此数据库上的表文件。操作命令如下：

```
ls /var/lib/mysql/sys
```

3.3 创建和管理用户数据库

将通过以下任务创建和管理微营销小程序系统的数据库。

3.3.1 任务 1 利用 MySQL 命令创建用户数据库

连接 MySQL 服务器,使用 MySQL 的 CREATE DATABASE 命令创建数据库。

1. 以默认方式简捷创建数据库

创建数据库的 SQL 语句如下:

```
CREATE DATABASE databasename;
```

其中,databasename 为数据库的名称。此命令采用系统默认的配置创建数据库。

【例 3.4】 创建微营销小程序系统的数据库,名称为 micromarketdb。

语句格式如下:

```
create database micromarketdb;
```

创建成功,会返回如下信息:

```
Query OK,1 row affected (0.01 sec)
```

2. 带选项参数创建数据库

【例 3.5】 创建微营销小程序系统数据库,要求判断要创建的数据库是否已经存在,同时设置数据库的字符集为 gb2312。

创建数据库的 SQL 语句如下:

```
CREATE DATABASE IF NOT EXISTS micromarketdb CHARACTER SET gb2312;
```

创建数据库时,先判断 micromarketdb 数据库是否存在,如果已经存在,则不会创建。

完成以上数据库创建后,可以看到在"/var/lib/mysql/"目录中生成了 micromarketdb 数据库子目录。由于没有在数据库上创建表结构,所以此目录为空。如何创建表结构将在项目 4 介绍。

创建数据库的语法格式及说明

语法格式如下:

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] databasename
 [[DEFAULT]|CHARACTER SET 字符集名][[DEFAULT] COLLATE 校对规则名]
```

参数说明:

"[ ]"内为可选项,"{|}"表示可选其中之一。

DATABASE|SCHEMA：表示创建 DATABASE 形式或 SCHEMA 形式的数据库。一般都是采用默认的 database 形式。

IF NOT EXISTS：表示在创建数据时判断此数据库是否已经存在，若不存在，则创建；若存在，则不能创建，执行也不会提示出错。

databasename：数据库名，数据库以目录的形式存储在文件系统中，所以要求按照操作系统文件夹的命名规则进行命名。

DEFAULT：指定默认值。

CHARACTER SET：指定 MySQL 支持的字符集。如，CHARACTER SET GB2312。

COLLATE：指定字符集校验规则。如，collate gb2312_chines_ci。

MySQL 系统的命令解释器对大小写不敏感，所以编写代码没有大小写的要求。以上命令中带下划线的中文部分为选项值，用具体的内容替换。

注意，在 MySQL 命令窗口执行 sql 语句一定要在语句行结尾加分号“;”表示结束。

3.3.2　任务 2　管理用户数据库

1. 打开数据库

要对数据库进行操作，首先需要打开此数据库。MySQL 使用 USE 命令打开数据库，最后打开的数据库成为当前工作的数据库。

命令格式：

```
USE databasename;
```

【例 3.6】 打开 micromarketdb 数据库。

```
USE micromarketdb;
```

2. 显示数据库

(1) 列出当前用户有权限查看的所有数据库。

```
SHOW DATABASES;
```

(2) 显示指定数据库的创建信息。

【例 3.7】 显示 micromarketdb 数据库的创建信息。

```
SHOW CREATE DATABASE micromarketdb;
```

3. 修改数据库

用 ALTER DATABASE 命令可以对已经创建的数据库进行修改。修改数据库时可以使用创建数据库命令中的相应选项，具体示例如下。

【例 3.8】 修改 micromarketdb 数据库，将字符集修改为 gb2312，校对规则改为

gb2312_chinese_ci。

```
ALTER DATABASE micromarketdb
DEFAULT CHARACTER SET gb2312
COLLATE gb2312_chinese_ci;
```

再查看此数据库结构，发现已经从 utf8 字符集改为 gb2312 字符集：

```
SHOW CREATE DATABASE micromarketdb;
```

4. 删除数据库

用 DROP DATABASE 命令删除数据库。

【例 3.9】 删除 micromarketdb 数据库。

```
DROP DATABASE micromarketdb;
```

再次查看数据库，发现数据库列表中已经没有此数据库了。

3.3.3 任务 3 使用 Navicat 管理用户数据库

通过 2.2.4 节的学习，已经掌握如何使用 Navicat 工具连接 MySQL 服务器。下面学习在 Navicat 图形界面中创建和管理数据库。

1. 新建数据库

打开项目 2 在 Navicat 中建立的远程链接 DebianMySQL，列出其所有数据库。

右击 DebianMySQL 连接，在菜单中选择“新建数据库”，弹出新建数据库窗口。输入数据库名（比如，micromarketdb），单击“确定”按钮，则创建此用户数据库，如图 3-17 所示。

图 3-17 Navicat 工具中新建用户数据库的操作界面

由于 3.3.1 节已经通过 SQL 命令创建了以上用户数据库，所以再创建此数据库时，会出现如图 3-18 所示错误提示信息，表示数据库已经存在，不能再创建。

2. 打开数据库

双击 micromarketdb 数据库名，数据库图标从灰色变成绿色，表示已经打开此数据库。数据库名称下面展示出多个功能菜单项，如“表”、“视图”和“函数”等，如图 3-19 所示，在后

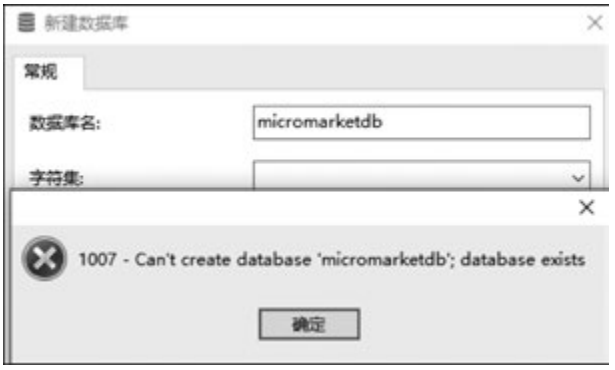

图 3-18　不能新建已经存在的数据库

续项目任务中将逐步学习使用这些功能。

图 3-19　Navicat 工具中丰富的数据库对象管理功能

3. 修改数据库

右击 micromarketdb 数据库，在菜单中选择“数据库属性”，弹出数据库属性窗口，在此窗口中可以修改数据库的字符集、排序规则，如图 3-20 所示。

图 3-20　Navicat 工具中修改数据库常规属性信息

4. 删除数据库

同样，右击需要删除的数据库，然后在菜单中选择“删除数据库”，即可删除。

3.4 介绍 MySQL 数据库存储引擎

存储引擎是 MySQL 组件，用于处理不同表类型的 SQL 操作。InnoDB 是默认且最通用的存储引擎。MySQL Server 使用可插拔存储引擎架构，允许在正在运行的 MySQL 服务器中加载和卸载存储引擎。要确定您的服务器支持哪些存储引擎，请使用 SHOW ENGINES 语句。

```
SHOW ENGINES;
```

执行命令语句，显示 MySQL 系统中所有数据库引擎如图 3-21 所示。

```
mysql> SHOW ENGINES;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| ndbcluster         | NO      | Clustered, fault-tolerant tables                               | NULL         | NULL | NULL       |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| ndbinfo            | NO      | MySQL Cluster system information storage engine                | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
11 rows in set (0.00 sec)
```

图 3-21 MySQL 数据库系统的存储引擎列表示意图

显示结果信息表中，“支持”(Support)列的值表示是否可以使用引擎，值为 YES 表示引擎可用、NO 表示引擎不可用、DEFAULT 表示引擎可用并被设置为默认存储引擎。

3.4.1 InnoDB 存储引擎

InnoDB 是一种平衡高可靠性和高性能的通用存储引擎，是现今 MySQL 默认的存储引擎。如果没有配置其他存储引擎为默认存储引擎，则使用不带 ENGINE 子句的创建数据库表(CREATE TABLE)语句时，将创建一个 InnoDB 表。

1. InnoDB 主要优势

(1) 其 DML 操作遵循 ACID 模型，事务具有提交、回滚和崩溃恢复功能，以保护用户数据。

(2) 行级锁定和 Oracle 风格的一致性读取提高了多用户并发性和性能。请参见 8.4 节并发事务与锁机制。

(3) InnoDB 表在磁盘上排列数据是基于主键优化查询。每个 InnoDB 表都有一个称

为聚集索引的主键索引，用于组织数据最大限度地减少主键查找的 I/O。

(4) 为了保持数据完整性，InnoDB 支持 FOREIGN KEY 约束。使用外键，会检查插入、更新和删除，以确保它们不会导致相关表之间出现数据不一致。

ACID 模型

ACID 模型具备四种特性：原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability)。

原子性：包括自动提交设置，COMMIT 语句，ROLLBACK 语句。一个事务中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚(Rollback)到事务开始前的状态，就像这个事务从来没有执行过一样。

一致性：主要涉及 InnoDB 内部处理以保护数据免于崩溃。在事务开始之前和事务结束以后，数据库的完整性限制没有被破坏。

隔离：主要涉及 InnoDB 事务，特别是应用于每个事务的隔离级别。相关 MySQL 功能包括：自动提交设置，事务隔离级别和 SET TRANSACTION 语句。当两个或者多个事务并发访问(此处访问指查询和修改的操作)数据库的同一数据时所表现出的相互关系。事务隔离分为不同级别，包括读未提交(Read uncommitted)、读提交(read committed)、可重复读(repeatable read)和串行化(Serializable)。将在项目 8 学习。

耐用性(持久性)：涉及与特定硬件配置交互的 MySQL 软件功能。由于 CPU、网络和存储设备的功能存在多种可能性，因此在这方面提供具体指南是最复杂的。在事务完成以后，该事务对数据库所做的更改持久地保存在数据库之中。相关的 MySQL 功能包括：InnoDB 双写缓冲区；innodb_flush_log_at_trx_commit、sync_binlog、innodb_file_per_table 变量；存储设备(例如磁盘驱动器、SSD 或 RAID 阵列)中的写入缓冲区；存储设备中由电池供电的缓存，不间断电源(UPS)保护运行 MySQL 服务器和存储 MySQL 数据的所有计算机服务器和存储设备的电源；备份策略，例如备份频率和类型以及备份保留期；对于分布式或托管数据应用程序，MySQL 服务器硬件所在的数据中心的特定特征以及数据中心之间的网络连接。

2. 使用 InnoDB 表的好处

(1) 如果服务器由于硬件或软件问题而意外退出，无论当时数据库发生了什么情况，重新启动数据库后都不需要执行任何特殊操作。InnoDB 会自动恢复完成崩溃之前提交的更改，并撤销正在进行但未提交的更改，从而允许用户重新启动并从中断的地方继续。

(2) InnoDB 存储引擎维护自己的缓冲池，在访问数据时将表和索引数据缓存在主内存中。经常使用的数据直接在内存中处理。此缓存适用于多种类型的信息并能够加快处理速度。在专用数据库服务器上，高达 80%的物理内存通常分配给缓冲池。

(3) 如果将相关数据拆分到不同的表中，则可以设置外键来强制引用完整性。

(4) 如果磁盘或内存中的数据损坏，校验和机制会在使用数据之前提醒用户注意这些

数据。innodb_checksum_algorithm 变量定义 InnoDB 使用的校验和算法。

(5) 为每个表设计具有适当主键列的数据库时,涉及这些列的操作会自动优化。在 WHERE 子句、ORDER BY 子句、GROUP BY 子句和连接操作中引用主键列的速度非常快。

(6) 插入、更新和删除操作通过称为更改缓冲的自动机制进行优化。InnoDB 不仅允许对同一个表进行并发读写访问,它还可以缓存更改的数据以简化磁盘 I/O。

(7) 性能优势不仅限于具有长时间运行查询的大型表。当从表中反复访问相同的行时,自适应哈希索引将接管以加快这些查找,就像它们来自哈希表一样。

(8) 可以压缩表和关联的索引。

(9) 可以加密数据。

(10) 可以创建和删除索引以及执行其他 DDL 操作,而对性能和可用性的影响很小。

(11) 表数据的存储布局对于 BLOB 和长文本字段来说更有效,采用 DYNAMIC 行格式。

(12) 可以通过查询 INFORMATION_SCHEMA 表来监视存储引擎的内部工作情况、性能。

(13) 可以将 InnoDB 表与其他 MySQL 存储引擎的表混合使用,甚至在同一个语句中也是如此。例如,您可以使用连接操作将 InnoDB 和 MEMORY 表中的数据合并到单个查询中。

(14) InnoDB 专为处理大数据量时的 CPU 效率和最大性能而设计。

对于 InnoDB 特定的调优技术,可以将其应用于 MySQL 服务器和应用程序代码。

3.4.2 MyISAM 存储引擎

MyISAM 是基于较旧的、不再使用的 ISAM 存储引擎的扩展。此引擎用于创建的表占用空间很小,通过表级锁限制读/写工作负载性能,通常用于 Web 和数据仓库配置中的只读或以读为主的工作负载。

每个 MyISAM 表都以两个文件的形式存储在磁盘上。这些文件的名称以表名开头,并具有指示文件类型的扩展名。数据文件的扩展名为 .MYD(MYData)。索引文件的扩展名为.MYI(MYIndex)。表定义存储在 MySQL 数据字典中。

由于 InnoDB 是默认引擎,创建表时要明确指定 MyISAM 表,使用 ENGINE 选项来创建,示例如下:

```
CREATE TABLE tb (id INT) ENGINE = MYISAM;
```

创建表的相关知识和技术将在项目 4 详细学习。

1. MyISAM 表的特点

(1) 所有数据值均以低字节在前存储。

(2) 所有数字键值都首先存储高字节,以实现更好的索引压缩。

(3) 支持大文件的文件系统和操作系统支持大文件(最多 63 位文件长度)。

(4) MyISAM 表中的行数限制为 2^{64} 行。

(5) 每个 MyISAM 表的最大索引数为 64,每个索引的最大列数为 16。

(6) 当按排序顺序插入行时(如使用 AUTO_INCRMENT 列时),索引树将被拆分,以便高节点仅包含一个键。这提高了索引树中的空间利用率。

(7) 支持每个表一个 AUTO_INCREMENT 列的内部处理。MyISAM 自动更新此列以进行 INSERT 和 UPDATE 操作。这使得 AUTO_INCRMENT 列更快。

(8) MyISAM 支持并发插入:如果表的数据文件中间没有空闲块,则可以在其他线程从表中读取数据的同时向其中插入新行。

(9) 可以将数据文件和索引文件放在不同物理设备上的不同目录中,以便使用 CREATE TABLE 的 DATA DIRECTORY 和 INDEX DIRECTORY 表选项来提高速度。

(10) BLOB 和 TEXT 列可以建立索引。

(11) 索引列中允许使用 NULL 值。

(12) 每个字符列可以有不同的字符集。

2. MyISAM 支持的功能

(1) 支持真正的 VARCHAR 类型;VARCHAR 列以存储在一个或两个字节中的长度开始。

(2) 具有 VARCHAR 列的表可能具有固定或动态的行长度。

(3) 表中 VARCHAR 和 CHAR 列的长度总和最多可达 64KB。

(4) 任意长度 UNIQUE 约束。

3.4.3　MEMORY 存储引擎

MEMORY 存储引擎(也称为 HEAP)使用存储在内存中的内容创建专用表。由于数据容易受到崩溃、硬件问题或断电的影响,因此仅将这些表用作临时工作区或只读缓存,用于从其他表中提取数据。MEMORY 表无法分区。

1. 何时使用 MEMORY 或 NDB Cluster

(1) 涉及瞬态、非关键数据的操作,例如,会话管理或缓存。当 MySQL 服务器停止或重新启动时,MEMORY 表中的数据将丢失。

(2) 内存存储可实现快速访问和低延迟。数据量可以完全容纳在内存中,而不会导致操作系统兑换虚拟内存页面。

(3) 只读或主要读取数据访问模式(有限更新)。

NDB Cluster 提供与 MEMORY 引擎相同的功能,但性能更高,并提供 MEMORY 不具备的附加功能:

(1) 行级锁定和多线程操作可减少客户端之间的争用。

(2) 包含语句混合也具有可扩展性。

(3) 可选的磁盘支持操作可实现数据持久性。

(4) 无共享架构和多主机操作,无单点故障,实现 99.999%的可用性。

(5) 跨节点自动分配数据;应用程序开发人员无须设计自定义分片或分区解决方案。

(6) 支持 MEMORY 不支持的可变长度数据类型(包括 BLOB 和 TEXT)。

2. MEMORY 表的特点

MEMORY 存储引擎不会在磁盘上创建任何文件。表定义存储在 MySQL 数据字典中。

(1) MEMORY 表的空间以小块的形式分配。表使用 100%动态哈希进行插入。不需要溢出区域或额外的密钥空间。

(2) MEMORY 表使用固定长度的行存储格式。可变长度类型(例如,VARCHAR)使用固定长度存储。

(3) MEMORY 表不能包含 BLOB 或 TEXT 列。

(4) MEMORY 包括对 AUTO_INCRMENT 列的支持。

(5) 非临时内存表在所有客户端之间共享,就像任何其他非临时表一样。

MySQL NDB Cluster

MySQL NDB Cluster 是适用于分布式计算环境的 MySQL 的高可用性、高冗余版本,使用 NDB 存储引擎(也称为 NDB Cluster)可以在集群中运行带有 MySQL 服务器和其他软件的多台计算机。从版本 8.0.19 开始,NDB Cluster 已作为通用(GA)版本提供。NDB 7.4 和旧版本系列不再受支持或维护。NDB 8.0 和 NDB 8.1 在生产中均受支持,建议用于新部署。

注意:MySQL NDB Cluster 不支持 InnoDB Cluster,必须使用带有 InnoDB 存储引擎的 MySQL Server 8.0 以及 NDB Cluster 发行版中未包含的其他应用程序来部署 InnoDB Cluster。

本教程使用的是 GPL(通用公共许可证)版,非商业许可,不提供 NDB Cluster 功能。

本教程采用 MySQL 默认存储引擎 InnoDB 创建项目表,其他诸多存储引擎不再一一详述。

【例 3.10】 修改 MySQL 的默认存储引擎为 MyISAM:

```
SET default_storage_engine = MyISAM;
```

3.5 实训拓展——创建网上订票系统数据库

由于网上订票系统交易数据量特别大,对数据库服务器的性能和硬盘存储空间要求很高,所以一般不会直接将数据存储在服务器系统硬盘分区上,而是采用单独的光纤高速传输磁盘阵列存储数据。一般也不会租赁第三方云服务器、共享网络带宽,而是搭建专用的服务器(群)和网络专线,确保数据安全、高速。

随着互联网应用普及和云计算技术的快速发展,分布式数据库成为了数据库领域的新热点。分布式环境中多个计算机节点构成一个数据库集群,共同完成大规模数据存储和处

理任务，实现数据的高可用性、高吞吐量及良好的扩展性。MySQL 作为开源关系型数据库管理系统，在分布式数据库中也有着重要的地位和应用。

分布式数据库架构

1. 基于 Master-Slave 架构的分布式数据库

该架构具有一个主数据库以及多个从数据库的结构。主数据库负责处理用户的读写请求，从数据库用于备份和复制主数据库的数据。各从数据库之间不直接通信，而是通过主数据库来实现数据同步。当主数据库出现故障时，在从数据库中选举一个备用节点作为主节点，继续服务。

在 MySQL 中，使用主从复制（Master-Slave Replication）来实现分布式数据库的 Master-Slave 架构。Master 节点接收用户的读写请求，并将数据同步到 Slave 节点上。读请求可以由 Master 或 Slave 处理，写请求只能由 Master 处理。在 Master 节点上进行数据操作后，数据会自动同步到 Slave 节点上。对于读密集型的业务，可以将读请求分配给 Slave 节点处理，减轻 Master 节点的压力。

2. 基于 Cluster 架构的分布式数据库

该架构中所有的节点都是相同的，共同组成一个集群，所有节点对外提供服务。数据在整个集群内是分片存储的，每个节点负责处理自己的数据。当有节点故障时，系统会自动将数据转移到其他节点保证服务的连续性。

在 MySQL 中，使用 Cluster 来实现分布式数据库的 Cluster 架构。MySQL Cluster 是基于 NDB 引擎的，它支持数据分片存储和自动分布式管理，并具有高性能、高可用性和强一致性等特点。MySQL Cluster 由三部分组成：数据存储、内存数据管理和查询处理。数据存储和内存数据管理运行在节点上，查询处理运行在 SQL 节点上。

【例 3.11】 将网上订票系统的数据库命名为 zt12306db，在 Debian 系统中用 mysql 命令创建此数据库。创建好后在 Windows 系统中打开 Navicat for MySQL 工具远程查看此数据库信息。

解析：

（1）运行虚拟机中的 Debian 操作系统。

（2）打开终端，连接上 MySQL。

```
mysql -uroot -p
```

（3）查看当前环境 MySQL 数据库文件存放路径。

```
SHOW VARIABLES LIKE 'datadir';
```

（4）创建 zt12306db 数据库，并显示数据库列表。

```
CREATE DATABASE zt12306db;
SHOW DATABASES;
```

(5) 创建成功后,退出 MySQL,回到终端。

(6) 进入 MySQL 数据库目录(/var/lib/mysql),查看是否存在新创建的数据库 zt12306db 目录。终端执行命令语句及列出的文件目录如图 3-22 所示。

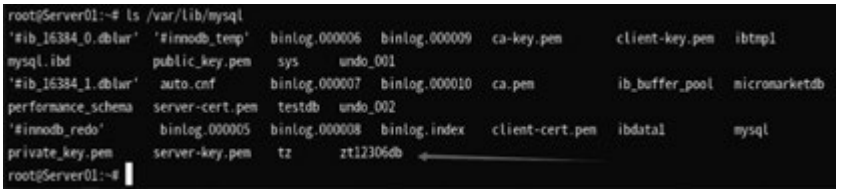

图 3-22 查看新建数据库的物理存储目录

(7) 在 Windows 系统中打开 Navicat for MySQL 工具远程查看此新数据库的信息。打开用 root 账号创建的远程连接(DebianMySQL),看到 zt12306db 数据库存在,如图 3-23 所示。

图 3-23 远程查看新建数据库

项目小结

本项目详细讲解了启动和停止 MySQL 服务器的方法,怎么使用 MySQL 命令连接 MySQL 服务器,如何查看数据库。详细了解了 MySQL 数据库架构,MySQL 物理数据库目录和文件管理机制,以及数据库表结构是以文件形式存放在数据库目录中。重点学习使用 MySQL 命令语句创建和管理用户数据库,同时学会使用 Navicat 管理工具创建和管理用户数据库。拓展了分布式数据库部署原理。在 Debian 系统中连接 MySQL 数据库,并创建了网上订票系统数据库。最后学习了 MySQL 数据库存储引擎相关知识。

习题

1. 选择题

(1) 在 Windows 命令窗口提示符下停止本机 MySQL 服务器的命令语句是(　　)。

A. STOP MYSQL　　B. NET STOP

C. NET START MYSQL　　D. NET STOP MYSQL

（2）连接 MySQL 数据库服务器的默认端口号是(　　)。

A. 80　　B. 8080　　C. 3306　　D. 1433

（3）在 Windows 命令窗口中远程连接 MySQL 数据库，使用(　　)参数指定端口号。

A. -h　　B. -u　　C. -p　　D. -P

（4）MySQL 的 InnoDB 引擎使用独享表空间存储方式将用户表的相关信息保留在 Data 目录下对应的数据库文件夹中，表数据文件的扩展名形式是(　　)。

A. .ibd　　B. .ibdata　　C. .CSV　　D. .myd

（5）以下不属于 MySQL 安装时自动创建的系统数据库是(　　)。

A. Information_schema　　B. mysql

C. sys　　D. mydb

（6）以下创建用户数据库语句错误的是(　　)。

A. CREATE DATEBASE testdb　　B. CREATE DATEBASE my.testdb

C. CREATE DATEBASE my_testdb　　D. CREATE Datebase _testdb

2. 填空题

（1）创建名为 studentsdb 数据库的命令语句是：_____；查看此数据库创建信息的命令语句是：_____。

（2）删除名为 studentsdb 数据库的命令语句是：_____。

（3）MySQL 系统数据库中，存储系统权限的数据库是：_____。

（4）在创建数据库时，如果使用_____命令子句，则会判断要创建的数据库是否存在，如果不存在则创建，如果存在则不会创建。

（5）MySQL 数据库的默认引擎是：_____。

实训

1. 实训目的

（1）学会登录数据库系统，查看数据库和数据库引擎。

（2）会使用不同工具创建和管理数据库。

（3）掌握创建和管理数据库的 SQL 命令语句。

2. 实训内容

（1）登录 MySQL 客户端，查看当前存在的数据库。

（2）查看当前安装的 MySQL 系统支持的存储引擎。

（3）创建 micromarketdb 数据库，指定字符集为 utf8，字符集校验规则为 utf8_general_ci。

（4）将 micromarketdb 数据库的字符集修改为 gb2312，字符集校验规则修改为 gb2312_chines_ci。

（5）删除 micromarketdb 数据库，并再次查看数据库，确认此数据库已经被删除。