第3章 MySQL 8.0数据库和表的基本操作

前面学习了设计数据库的基本理论，在此基础上就可以完成以教务管理数据库为例的MySQL 数据库的概念结构设计和逻辑结构设计部分，实现在 MySQL 8.0 的软件环境中创建和维护数据库的操作。

创建数据库的目的是存储、管理和查询数据，而表是存储数据最重要的载体。表是MySQL 数据库中最重要的数据库对象，也是构建高性能数据库的基础。

本章将学习设计数据库的基本过程，以及创建和管理数据库的基本操作，并通过建立教务管理数据库中的学生、课程、教师和成绩等数据表介绍各种数据表的创建、修改、管理、存储与数据格式转换，以及实现数据完整性的方法和基本操作。

3.1 MySQL 8.0 数据库概述

MySQL 数据库的管理主要包括数据库的创建、打开当前操作的数据库、显示数据库结构以及删除数据库等操作。MySQL 数据库具有可移植性好、超强的稳定性和强大的查询功能，能够支持大型数据库。

MySQL 数据库管理系统提供了许多命令行工具，这些工具可以用来管理 MySQL 服务器、对数据库进行访问控制、管理 MySQL 用户以及进行数据库备份和恢复等。MySQL 还提供了图形化管理工具，这使得对数据库的操作更加简单。

3.1.1 MySQL 数据库的基础知识

1. MySQL 数据库文件

数据库管理的主要任务包括创建、操作和支持数据库。在 MySQL 8.0 中，每个数据库的文件都对应存放在一个与数据库同名的文件夹中。数据库文件的默认存放位置是“C:\ProgramData\MySQL\MySQL Server 8.0\Data\”，用户可以通过配置向导或手工修改数据库的默认存放位置。由于 MySQL 8.0 InnoDB 引擎的重构，MySQL 8.0 中 MySQL 数据表的文件合并到数据根目录下的 mysql.ibd 中，数据库中的每个数据表的内容都存储在一个同名的扩展名为.ibd 的文件中。

2. MySQL 自动建立的数据库

在 MySQL 安装完成之后，会在其 data 目录下自动创建几个必需的数据库，用户可以

使用 show databases 命令来查看当前存在的所有系统数据库，如表 3-1 所示。

表 3-1 MySQL 系统数据库

数据库名称	数据库的作用
mysql	描述用户访问权限
information_schema	保存关于 MySQL 服务器所维护的所有其他数据库的信息，例如数据库名、数据库的表、表项的数据类型与访问权限等
performance_schema	主要用于收集数据库的服务器性能参数
sakila	MySQL 官方测试用的数据库
sys	sys 数据库里面包含了一系列的存储过程、自定义函数以及视图，存储了许多系统的元数据信息
world	存储当前世界上的主要城市、国家和语言信息

3. 查看数据库

在成功安装数据库后，可以使用 show databases 命令查看 MySQL 服务器中的所有数据库信息。

【例 3.1】 使用 show databases 语句查看 MySQL 服务器中的所有数据库。

命令和运行结果如下：

```
mysql> show databases;
       +--------------------+
       | Database           |
       +--------------------+
       | information_schema |
       | mysql              |
       | performance_schema |
       | sakila             |
       | sys                |
       | world              |
       +--------------------+
       6 rows in set (0.36 sec)
```

从例 3.1 的运行结果可以看出，通过 show databases 命令可以查看 MySQL 服务器中的所有数据库，结果显示了 MySQL 服务器中的 6 个数据库。

3.1.2 MySQL 存储引擎

数据库存储引擎是数据库底层软件组件，数据库管理系统使用数据引擎进行创建、查询、更新和删除数据的操作。MySQL 数据库提供了多种存储引擎，用户可以根据不同的需求为数据表选择不同的存储引擎，用户也可以根据需要编写自己的存储引擎，MySQL 的核心就是存储引擎。

数据库的存储引擎决定了表在计算机中的存储方式。存储引擎就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等的实现方法。因为在关系数据库中数据是以表的形式存储的，所以存储引擎简而言之就是指表的类型。

MySQL 8.0 支持的存储引擎有 InnoDB、MRG_MYISAM、MEMORY、BLACKHOLE、

MyISAM、CSV、ARCHIVE、FEDERATED 和 PERFORMANCE-SCHEMA 共 9 种。不同存储引擎有各自的特点，以适应不同的需求。MySQL 常用的存储引擎如表 3-2 所示。

表 3-2　MySQL 常用存储引擎的功能对比

功　　能	InnoDB	MyISAM	MEMORY
存储限制	64TB	256TB	RAM
支持事务	支持	不支持	不支持
空间使用	高	低	低
内存使用	高	低	高
支持数据缓存	支持	不支持	不支持
插入数据的速度	低	高	高
支持外键	支持	不支持	不支持

1. 查看数据库存储引擎

MySQL 的存储引擎是一种插入式的存储引擎，这决定了 MySQL 数据库中的表可以用不同的方式存储。用户可以根据自己的不同要求选择不同的存储方式以及是否进行事务处理等。MySQL 的默认存储引擎是 InnoDB，如果想设置其他存储引擎，可以使用以下 MySQL 命令：

```
set default_storage_engine=MyISAM;
```

该命令可以临时将 MySQL 当前会话的存储引擎设置为 MyISAM，使用 MySQL 命令“show engines;”可以查看当前 MySQL 服务实例默认的存储引擎。其命令和结果如图 3-1 所示。

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

图 3-1　查看 MySQL 8.0 数据库支持的存储引擎类型

2. 常用存储引擎介绍

1）存储引擎 InnoDB

MySQL 8.0 选择 InnoDB 作为默认存储引擎。InnoDB 是事务型数据库的首选引擎，兼具原子性、一致性、隔离性和持久性等事务的 4 个特性，是具有提交、回滚和崩溃恢复能力的事务安全存储引擎，并支持行级锁定和外键约束。

InnoDB 存储引擎还支持自动增长列 auto_increment 和全文检索功能，因此 InnoDB 存储引擎能够提供 OLTP 支持，如果表需要执行大量的添加、删除、修改数据的操作，出于事务安全方面的考虑，InnoDB 存储引擎是更好的选择。在存储表中的数据时，每张表的存储都按主键顺序存放，如果没有显示在定义表时指定主键，InnoDB 会为每行生成一个 6 字

节的 ROWID，并以此作为主键。

InnoDB 存储引擎完全与 MySQL 服务器整合，为在主内存中缓存数据和索引而维持它自己的缓冲池。InnoDB 将它的表和索引存储在一个逻辑表空间中， 表空间可以包含数个文件。

2）存储引擎 MyISAM

MyISAM 存储引擎不支持事务、外键约束，但访问速度快，对事务的完整性不要求，适合于以 select/insert 为主的表。MyISAM 表占用的空间很小。其表级锁定特性限制了读写工作负载中的性能，因此它通常用于 Web 和数据仓库配置中的只读或以读取为主的工作负载中。

3）存储引擎 MEMORY

MEMORY 存储引擎是 MySQL 中的一类特殊的存储引擎。其所有数据存储在 RAM 中，以便在需要快速查找非关键数据的环境中进行快速访问。每个 MEMORY 表可以放置数据量的大小受 max_heap_table_size 系统变量的约束，可按需求增加。此外，在定义 MEMORY 表时可通过 max_rows 子句定义表的最大行数。

值得注意的是，服务器需要有足够的内存来维持 MEMORY 存储引擎的表的使用。如果不需要使用了，可以释放这些内存，甚至可以删除不需要的表。

3. 如何选择存储引擎

在实际工作中，选择一个合适的存储引擎是一个很复杂的问题。每种存储引擎都有各自的优势，可以根据各种存储引擎的特点进行对比，给出不同情况下选择存储引擎的建议。

MySQL 中提到了存储引擎的概念，它是 MySQL 的一个特性，可简单理解为后面要介绍的表类型。每一个表都有一个存储引擎，可以在创建时指定，也可以使用 alter table 语句修改，都是通过 engine 关键字设置的。

3.2　MySQL 数据库的设计过程

数据库设计是指对于一个给定的应用环境，构造优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。

数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境。高效率的运行环境是指数据库数据的存取效率、数据库存储空间的利用率、数据库系统运行管理的效率等都是高的。

数据表设计的优劣将影响磁盘空间的使用效率、数据处理时内存的利用率以及数据的查询效率。在这个过程中要注意表结构的规范化、数据类型的正确选择，以及数据库和数据表字符集的统一问题。表的数据完整性规则是保证表中数据的正确性、精确性和可靠性的关键。

以高校的教务管理系统来说，需要数据库来存储学生的学籍信息、考试信息、教师信息、课程信息等。数据库技术可以更加有效地管理和存取大量的数据资源，以提高人力、物力和财力的利用率和工作效率。

3.2.1 数据库设计的基本过程

一般来说，按照数据库规范化设计的方法，数据库设计可分为需求分析、概念设计、逻辑设计和物理设计 4 个阶段，如图 3-2 所示。之后是软件开发阶段中的数据库创建和数据库运行与维护。在实际的项目开发中，如果系统的数据关系较复杂，数据存储量较大，设计的表较多，表和表之间的关系比较复杂，就需要首先考虑规范的数据库设计，然后再进行具体的创建库、创建表的工作。数据库设计主要包括以下内容。

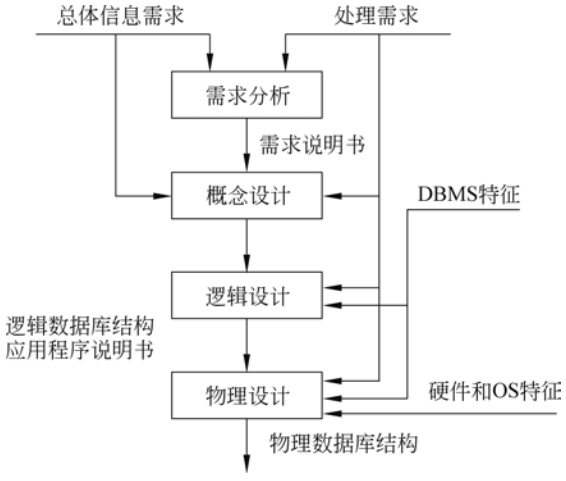

图 3-2　数据库设计的步骤

（1）需求分析：需求分析的目标是通过调查研究了解用户的数据要求和处理要求，并按一定的格式整理形成需求说明书。需求说明书是需求分析阶段的成果，也是以后设计的依据，它包括数据库所涉及的数据、数据的特征、数据量和使用频率的估计等。例如数据名、属性及其类型、主关键字属性、保密要求、完整性约束条件、使用频率、更改要求、数据量估计等。

（2）概念设计：概念设计是数据库设计的第 2 阶段，其目标是对需求说明书提供的所有数据和处理要求进行抽象与综合处理，按一定的方法构造反映用户环境的数据及其相互联系的概念模型。这种概念数据模型与 DBMS 无关，是面向现实世界的数据模型，极易为用户所理解。为保证所设计的概念数据模型能正确、完全地反映用户（单位）的数据及其相互关系，便于进行所要求的各种处理，在本阶段设计中可吸收用户参与和评议设计。

实体关系（E-R）的数据库设计方法是目前最常用的方法。基于实体关系的数据库设计方法的基本思想是在需求分析的基础上用 E-R 图构造一个纯粹反映现实世界实体之间内在关系的企业模式，然后将此企业模式转换成选定的 DBMS 上的概念模式。每个实体或联系将映射为一个数据表。

（3）逻辑设计：逻辑设计阶段的设计目标是把上一阶段得到的与 DBMS 无关的概念数据模型转换成等价的，并为某个特定的 DBMS 所接受的逻辑模型所表示的概念模式，同时将概念设计阶段得到的应用视图转换成特定 DBMS 下的应用视图。在转换过程中要进一步落实需求说明，并满足 DBMS 的各种限制。逻辑设计阶段的结果是 DBMS 提供的数据定

义语言（DDL）写成的数据模式。

（4）物理设计：物理设计阶段的任务是把逻辑设计阶段得到的逻辑数据库在物理上加以实现，其主要内容是根据 DBMS 提供的各种手段设计数据的存储形式和存取路径，例如文件结构、索引设计等，即设计数据库的内模式或存储模式。数据库的内模式对数据库的性能影响很大，应根据处理需求及 DBMS、操作系统和硬件的性能进行精心设计。

在数据库设计的基本过程中，每个阶段设计基本完成后，都要进行认真的检查，看看是否满足应用需求，是否符合前面已执行步骤的要求和满足后续步骤的需要，并分析设计结果的合理性。在数据库设计完成后，就可以利用 MySQL 创建数据库了。

3.2.2　教务管理数据库设计的规范化

数据库应用程序的性质和复杂性可以使数据库的设计过程变化很大。一个简单的数据库的设计可以依赖于设计者的技巧和经验，采用直接设计数据库的方式进行；而对于为成千上万的客户处理事务的数据库，数据库设计可能是长达数百页的正式文档，其中需要包含有关数据库的各种可能细节。要进行较复杂的数据库设计，必须遵守数据库设计规范化规则（Normalization Rule），并按照软件工程提供的规范才能进行。

按照规范化规则设计数据库，可以将数据冗余降至最低，使得应用程序软件可以在此数据库中轻松实现强制完整性，且很少包括执行涉及 4 个以上表的查询。规范化理论就是为了设计好的基本关系，使每个基本关系独立表示一个实体，并且尽量减少数据冗余。满足一定条件的关系模式称为范式（Normal Form，NF），一个低级范式的关系模式通过分解（投影）方法可转换成多个高一级范式的关系模式的集合，这个过程称为规范化。

1. 按照规范化规则设计数据库

数据依赖是一个关系内部属性与属性之间的一种约束关系。这种约束关系是通过属性间值的相等与否体现出来的数据间相互联系，它是现实世界属性间相互联系的抽象，是数据内在的性质，是语义的体现。人们提出了许多种类型的数据依赖，其中最重要的是函数依赖（Function Dependency，FD）和多值依赖（Multivalued Dependency，MVD）。函数依赖极为普遍地存在于现实世界中。比如一个学生的关系 student，可以由学号、姓名、性别、电话等几个属性描述。由于一个学号只对应一个学生，所以一旦“学号”值确定，学生的姓名、性别、电话等的值也就被唯一地确定了。

例如建立一个描述学校教务的数据库 teaching，该数据库涉及的对象包括学生学号、学生姓名、学生性别、电话、课程号、课程名和成绩等数据项。假设用一个单一的关系模式“学生”来表示，则该关系模式的属性集合为：

U={学生学号，学生姓名，学生性别，电话，课程号，课程名，成绩}

考察这个关系模式发现存在以下问题。

（1）数据冗余度大：课程号和课程名重复出现，重复次数与该班所有学生的所有课程成绩出现的次数相同。

（2）更新异常：由于数据冗余，当更新数据库中的数据时，系统要付出很大的代价来维护数据库的完整性，否则会面临数据不一致的危险。

（3）插入异常：如果一门课程刚开设，尚无学生选课记录，则系统无法把该课程信息存入数据库。

（4）删除异常：如果某一级的学生全部毕业了，在删除该班学生信息的同时也把这个课程的信息一起删除了。

鉴于存在以上种种问题，可以得出这样的结论：学生关系模式不是一个规范化的关系模式，一个规范化的关系模式应当不会发生插入异常、删除异常、更新异常，数据冗余度应尽可能地小。

2. 教务管理数据库的规范设计

为了避免上述诸多异常，在进行数据库设计时需要遵守称为数据库范式的规则。下面以教务管理数据库为例，介绍数据库设计中的范式（NF）理论。

（1）第 1 范式（1NF）：第 1 范式的目标是确保每列的原子性。如果每列都是不可再分的最小数据单元，则满足第 1 范式（1NF）。

现以学生表为例，设计学生表的结构如下：

学生（学生学号，学生姓名，学生性别，电话，课程号，课程名，成绩）

以上学生表中的各项都符合 1NF 条件。

（2）第 2 范式（2NF）：第 2 范式在第 1 范式的基础上要求确保表中的每列都和码相关，即每一个非主属性都要完全函数依赖于码。

分析学生关系模式，码应该为（学生学号，课程号），很明显，在该关系模式中学生姓名、课程名、学生性别、电话等只完全函数依赖于学生学号，因此对码（学生学号，课程号）是部分函数依赖，故该关系模式不满足第 2 范式。

如果把学生的相关属性单独拿出来，形成关系模式：

学生（学生学号，学生姓名，学生性别，电话，课程号，课程名）

选修（学生学号，课程号，成绩）

则以上两个关系模式都符合第 2 范式。

（3）第 3 范式（3NF）：第 3 范式是在第 2 范式的基础上要求确保表中的每列都和码直接相关，而不是间接相关。如果一个关系满足 2NF，并且除了码以外的其他列都不相互依赖，则满足第 3 范式（3NF）。

为了理解第 3 范式，需要根据 Armstrong 公理之一定义传递函数依赖。假设 A、B 和 C 是关系模式 R 的 3 个属性，如果 A→B 且 B→C，则从这些函数依赖中可以得出 A→C。

考察上述分解后的关系模式：

学生（学生学号，学生姓名，学生性别，电话，课程号，课程名）

可以得出课程名→课程号，而课程号→学生学号（假设学生选修该课程），因此存在课程名→学生学号的传递函数依赖，故该关系模式不符合第 3 范式。

如果把学生关系模式中课程的相关属性单独拿出来，形成关系模式：

学生（学生学号，学生姓名，学生性别，电话）

课程（课程号，课程名）

则以上两个关系模式都满足第 3 范式。

分析学生实体和课程实体之间的联系，并添加一些必要的属性，可以得出学生实体和课程实体之间是多对多（M:N）的联系，因此绘制学生实体和课程实体的“选修”关系局部 E-R 图如图 3-3 所示。如果再加上教师实体，并针对本系统的特点修改，则教务管理系统的 E-R 图如图 3-4 所示。

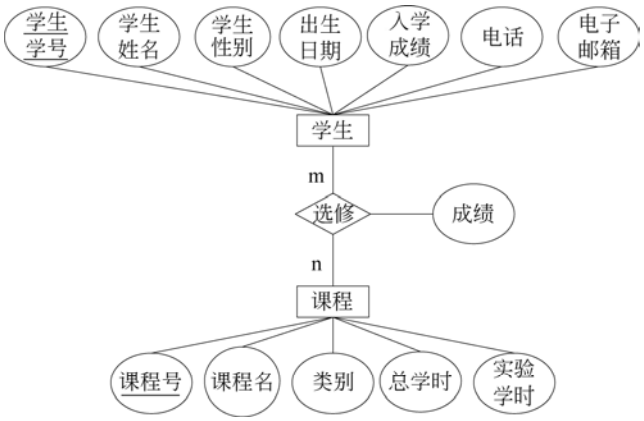

图 3-3 学生-课程“选修”关系 E-R 图

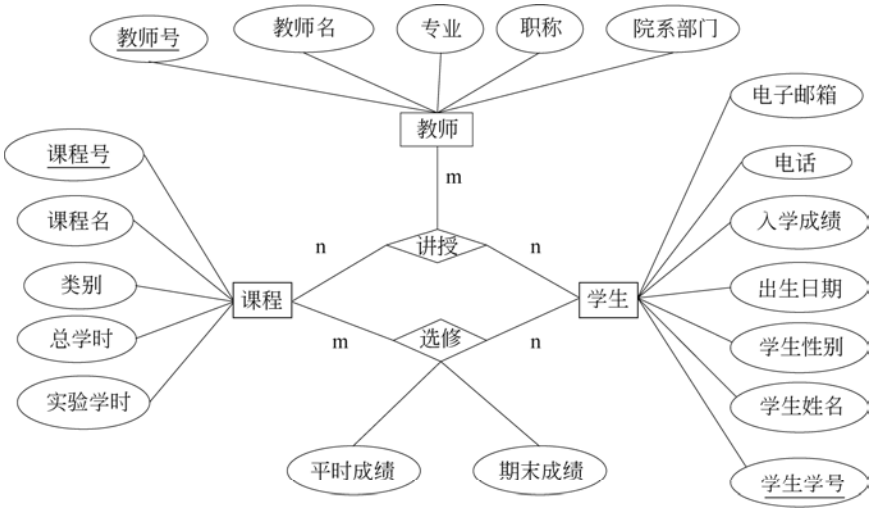

图 3-4 教务管理系统 E-R 图

之后就可以在此基础上根据 E-R 图的转换利用 MySQL 等软件创建 teaching 数据库和 student、score、course、teacher 等数据表了。

3.3 用户数据库的创建和管理

本节利用 MySQL 语句创建和维护教务管理数据库，用于存储学校的学生、课程、教师和成绩等基本数据。本节建立名为 teaching 的教务管理数据库，将在本书中作为数据库管理的示例数据库应用。

3.3.1 创建数据库


视频讲解

在创建数据库时，数据库的命名要符合标识符规则，名称最长可为 64 个字符，而别名最长可多达 256 个字符。另外，不能使用关键字作为数据库名、表名，以及已经存在的数据库名。

在默认情况下，Windows 下数据库名、表名的大小写是不敏感的，而在 Linux 下数据库名、表名的大小写是敏感的。为了便于数据库在平台间进行移植，可以用小写字母来定义数据库名和表名。

1. 创建数据库的语法结构

使用 create database 或 create schema 命令创建数据库。其语法结构如下：

```
create {database|schema}[if not exists]databasename
[[default]character set charset_name]
[|[default]collate collation_name];
```

说明：

（1）create database | schema：创建数据库的命令。在 MySQL 中 schema 也是指数据库。

（2）if not exists：如果已存在某个数据库，再创建一个同名的库，这时会出现错误信息。为避免出现错误信息，可以在建库前加上这一判断，只有在该库目前不存在时才执行 create database 操作。

（3）databasename：数据库标识符名。

（4）[default] character set charset_name：default character set 指定数据库的默认字符集（Charset），charset_name 为字符集名称。在创建数据库时最好指定字符集。

（5）[default] collate collation_name：collate 指定字符集的校对规则，collation_name 为校对规则名称。

2. 创建数据库示例

创建数据库是指在数据库系统中划分一块空间，用来存储相应的数据。这是进行表操作的基础，也是进行数据库管理的基础。在 MySQL 中，创建数据库是通过 SQL 语句 create database 实现的。

【例 3.2】 通过 create database 语句创建一个名为 mysqltest 的数据库。

命令和运行结果如下：

```
mysql> create database if not exists mysqltest;
       Query OK, 1 row affected (0.05 sec)
```

结果表明，创建 mysqltest 数据库成功。

【例 3.3】 创建教务管理数据库 teaching，并指定字符集为 utf8mb4，校对原则为 utf8mb4_0900_ai_ci。

命令和运行结果如下：

```
mysql> create database teaching
    -> default character set utf8mb4
    -> default collate utf8mb4_0900_ai_ci;
      Query OK, 1 row affected (0.13 sec)
```

成功创建数据库后，可以使用 show database 命令查看数据库，也可以在指定路径（或数据库的默认存放位置）下查看数据库。

3.3.2 管理数据库


视频讲解

1. 打开数据库

在数据库创建后，若要操作一个数据库，还需要使其成为当前的数据库，即打开数据库。用户可以使用 use 语句打开一个数据库，使其成为当前默认数据库。

例如选择名为 mysqltest 的数据库，设置其为当前默认的数据库。

命令和运行结果如下：

```
mysql> use mysqltest;
       Database changed
```

其中，Database changed 表明数据库 mysqltest 已经打开，变成了当前数据库，用户可以在数据库 mysqltest 中进行相关的操作了。

2. 修改数据库

在数据库创建后，如果需要，可以修改数据库的参数。

修改数据库的语法格式如下：

```
alter {database | schema} [db_name]
[default] character set charset_name]
[|[default] collate collation_name];
```

其中，alter 是修改数据库的命令关键字。

【例 3.4】 修改 mysqltest 库的字符集为 GB2312，校对原则为 gb2312_chinese_ci。

命令和运行结果如下：

```
mysql> alter database mysqltest
    -> default character set gb2312
    -> collate gb2312_chinese_ci;
      Query OK, 1 row affected (0.17 sec)
```

3. 显示数据库结构

如果要查看数据库的相关信息，例如 MySQL 版本 id、默认字符集等，使用 show 命令实现。

【例 3.5】 显示数据库 teaching 的结构信息。

命令和运行结果如下：

```
mysql> show create database teaching;
       +----------+-------------------------------------------------+
       | Database | Create Database                                 |
       +----------+-------------------------------------------------+
       | teaching | CREATE DATABASE 'teaching'
                       /*!40100 DEFAULT CHARACTER SET utf8mb4
                                COLLATE utf8mb4_0900_ai_ci */
                       /*!80016 DEFAULT ENCRYPTION='N' */ |
       +----------+-------------------------------------------------+
```

```
1 row in set (0.05 sec)
```

4. 删除数据库

删除数据库是指在数据库系统中删除已经存在的数据库。在删除数据库之后，原来分配的空间将被收回。删除数据库的语法格式如下：

```
drop database  [if exists] db_name
```

例如，删除 mysqltest 库的命令如下：

```
mysql> drop database mysqltest;
```

需要注意的是，删除数据库会删除该数据库中所有的表和所有数据，因此删除数据库前最好存有备份。

3.4 MySQL 数据库表的管理

在 MySQL 数据库系统中可以按照不同的标准对表进行分类。

（1）按照表的用途分类。

① 系统表：用于维护 MySQL 服务器和数据库正常工作的数据表。例如，在系统数据库 mysql 中就存在若干系统表。

② 用户表：由用户自己创建的用于各种数据库应用系统开发的表。

③ 分区表：分区表是将数据水平划分为多个单元的表，这些单元可以分布到数据库的多个文件组中。在维护整个集合的完整性时，使用分区可以快速而有效地访问或管理数据子集，从而使大型表或索引更易于管理。

（2）按照表的存储时间分类。

① 永久表：包括 SQL Server 的系统表和用户在数据库中创建的数据表，该类表除非人工删除，否则将一直存储在介质中。

② 临时表：临时表只有创建该表的用户在用来创建该表的连接中可见。当临时表关联的连接被关闭时，临时表自动被删除。如果服务器关闭，则所有临时表会被清空、关闭。

3.4.1 InnoDB 存储引擎的表空间

MySQL 8.0 的数据库表默认使用 InnoDB 存储引擎，InnoDB 表空间是 MySQL 数据表的存储空间的管理模式。在这种管理模式下，数据库中不仅存储数据文件和重做日志文件，还要管理这些文件的表空间文件。InnoDB 表空间分为共享表空间和独立表空间两种类型。

1. 表空间的基本概念

（1）共享表空间：MySQL 8.0 服务实例承载着数据库中所有 InnoDB 表的数据、索引、各种元数据以及事务回滚（undo）信息，全部存放在共享表空间文件中。在默认情况下，该文件位于数据库的根目录下，文件名是 ibdata1，初始大小为 12MB，能够自动扩展。也就是说，InnoDB 的所有文件共享一个表空间，其最大容量限制约为 64TB。

（2）独立表空间：每一个表都会以独立的文件方式来进行存储，每一个表都有一个.ibd 文件。该文件包括一个表单独的数据内容以及索引内容。

2. 查看数据库的表空间

利用以下命令可以查看数据库的表空间：

```
mysql> show variables like 'InnoDB_data%';
       +-----------------------+-----------------------+
       | Variable_name         | Value                 |
       +-----------------------+-----------------------+
       | innodb_data_file_path | ibdata1:12M:autoextend |
       | innodb_data_home_dir  |                       |
       +-----------------------+-----------------------+
       2 rows in set, 1 warning (0.35 sec)
```

如果用 autoextend 选项描述最后一个数据文件，当 InnoDB 用尽所有表自由空间后将会自动扩充最后一个数据文件，每次的增量约为 8 MB。

不管是共享表空间还是独立表空间，都会存在 InnoDB_data_file 文件，因为这些文件不仅要存放数据，还要存储事务回滚（undo）信息。

3. 共享表空间和独立表空间的比较

（1）共享表空间的特点：表空间可以分成多个文件存放在一起方便管理，且共享表空间分配后不能回缩。多个表及索引在表空间中混合存储，当数据量非常大的时候，表做了大量的删除操作后表空间中将会有大量的空隙，特别是对于统计分析，经常进行删除操作的这类应用最不适合用共享表空间。

（2）独立表空间的特点：每个表都有独立的表空间，每个表的数据和索引都会存储在自己的表空间中，可以实现单表在不同的数据库中移动。对于使用独立表空间的表，不管怎么删除，表空间的碎片都不会严重影响性能，而且还有机会处理。单表增加过大，当单表占用的空间过大时，存储空间会不足。drop table 操作能自动回收表空间，对于统计分析或者是日志表，删除大量数据的命令如下：

```
alter table TableName engine=innodb;
```

4. 共享表空间和独立表空间之间的转换

（1）查看当前数据库的表空间管理类型：可以通过以下命令查看。

```
mysql> show variables like "InnoDB_file_per_table";
```

对于独立表空间，如果将全局系统变量 InnoDB_file_per_table 的值设置为 on（InnoDB_file_per_table 的默认值为 off，on 代表独立表空间管理，off 代表共享表空间管理），那么之后再创建 InnoDB 存储引擎的新表，这些表的数据信息、索引信息都将保存到独立表空间文件。

（2）修改数据库的表空间管理方式：修改 InnoDB_file_per_table 的参数值（InnoDB_file_per_table=1 为使用独立表空间，InnoDB_file_per_table=0 为使用共享表空间）即可，但是修改不能影响之前已经使用过的共享表空间和独立表空间。

（3）共享表空间转换为独立表空间的方法（参数 InnoDB_file_per_table=1 需要设置）：单个表的转换操作可以用以下命令实现：

```
alter table table_name engine=innodb;
```

3.4.2 创建数据库表


视频讲解

在数据库创建之后，数据库是空的，是没有表的，可以用 show table 命令查看。

1. 创建表的语法结构

表决定了数据库的结构，表是存放数据的地方，一个库需要什么表，各数据库表中有什么样的列，是要合理设计的。创建表的语法结构如下：

```
create [temporary]table[if not exists]table_name
[([column_definition],…|[index_definition])]
[table_option][select_statement];
```

说明：

（1）temporary：使用该关键字表示创建临时表。

（2）if not exists：如果数据库中已存在某张表，再创建一个同名的表，这时会出现错误信息。为避免出现错误信息，可以在建表前加上这一判断，只有在该表目前不存在时才执行 create table 操作。

（3）table_name：要创建的表名。

（4）column_definition：字段的定义，包括指定字段名、数据类型、是否允许空值，指定默认值、主键约束、唯一性约束、注释字段名、是否为外键，以及字段类型的属性等。字段定义的具体格式如下：

```
col_name  type  [not null | null] [default default_value]
[auto_increment] [unique [key] | [primary] key]
[comment 'string'] [reference_definition]
```

其中：

① col_name：字段名。

② type：声明字段的数据类型。

③ null（not null）：表示字段是否可以是空值。

④ default：指定字段的默认值。

⑤ auto_increment：设置自增值属性，只有整数类型才能设置此属性。auto_increment 列值从 1 开始。每个表只能有一个 auto_increment 列，并且它必须被索引。

⑥ unique key：对字段指定唯一性约束。

⑦ primary key：对字段指定主键约束。

⑧ comment 'string'：注释字符串。

⑨ reference_definition：指定字段外键约束。

（5）index_definition：为表的相关字段指定索引。

（6）table_option：表的选项，包括存储引擎、字符集等。

（7）select_statement：用于定义表的查询语句。

2. 利用 SQL 语句创建数据表

本书的教务管理数据库 teaching 将根据前面的需求分析和简化创建 5 张表，即 student（学生表）、course（课程表）、score（成绩表）、teacher（教师表）和 teach_course（纽带表）。各表的结构如表 3-3～表 3-7 所示。

【例 3.6】 按照表 3-3 所示的结构创建 student 表。

表 3-3 student 表的结构

列序号	字 段 名	类 型	取 值 说 明	列 含 义
1	studentno	char(11)	主键	学生学号
2	sname	char(8)	否	学生姓名
3	sex	enum (2)	否	学生性别
4	birthdate	date	否	出生日期
5	entrance	int(3)	否	入学成绩
6	phone	varchar(12)	否	电话
7	Email	varchar(20)	否	电子邮箱

程序代码如下：

```
mysql> create table if not exists student
     (
      studentno  char(11)not null comment'学生学号',
      sname char(8)not null comment'学生姓名',
      sex enum('男', '女') default '男'comment'学生性别',
      birthdate date not null comment'出生日期',
      entrance int(3)null comment'入学成绩',
      phone varchar(12) not null comment'电话',
      Email varchar(20) not null comment'电子邮箱',
      primary key (studentno)
     );
```

【例 3.7】 利用 create table 命令建立课程表 course，表结构如表 3-4 所示。

表 3-4 course 表的结构

列序号	字 段 名	类 型	取 值 说 明	列 含 义
1	courseno	char(6)	主键	课程号
2	cname	char(6)	否	课程名
3	type	char(8)	否	类别
4	period	int(2)	否	总学时
5	exp	int(2)	否	实验学时
6	term	int(2)	否	开课学期

程序代码如下：

```
mysql> create table if not exists course
     (
       courseno  char(6) not null,
       cname  char(6) not null,
       type char(8) not null,
       period int(2) not null,
       exp int(2) not null,
       term int(2) not null,
       primary key (courseno)
     );
```

【例 3.8】 利用 create table 命令建立成绩表 score，表结构如表 3-5 所示。在该表中，主键由两个列构成。

表 3-5　score 表的结构

列 序 号	字 段 名	类　型	取 值 说 明	列 含 义
1	studentno	char(11)	主键	学生学号
2	courseno	char(6)		课程号
3	daily	float(4,1)	否	平时成绩
4	final	float(4,1)	否	期末成绩

利用 create table 语句在数据库 teaching 中建立成绩表 score 的程序代码如下：

```
mysql> create table if not exists score
     ( studentno  char(11) not null,
       courseno  char(6) not null,
       daily float(4,1) default 0,
       final float(4,1) default 0,
       primary key (studentno, courseno)
     );
```

【例 3.9】 利用 create table 命令建立教师表 teacher，表结构如表 3-6 所示。

表 3-6　teacher 表的结构

列 序 号	字 段 名	类　型	取 值 说 明	列 含 义
1	teacherno	char(6)	主键	教师号
2	tname	char(8)	否	教师名
3	major	char(10)	否	专业
4	prof	char(10)	否	职称
5	department	char(16)	否	院系部门

利用 create table 语句在数据库 teaching 中建立教师表 teacher 的程序代码如下：

```
mysql> create table if not exists teacher
     ( teacherno  char(6) not null comment '教师号',
       tname  char(8) not null comment '教师名',
```

```
  major  char(10) not null comment '专业',
  prof char(10) not null comment '职称',
  department char(16) not null comment '院系部门',
  primary key (teacherno)
);
```

【例 3.10】 为了完善 teaching 数据库的表间联系，创建表结构如表 3-7 所示的纽带表 teach_course。

表 3-7　teach_course 表的结构

列 序 号	字 段 名	类　型	取 值 说 明	列 含 义
1	teacherno	char(6)	主键	教师号
2	courseno	char(6)		课程号

程序代码如下：

```
mysql> create table if not exists teach_course
      (teacherno char(6) not null,
       courseno  char(6) not null,
       primary key(teacherno,courseno)
       );
```

说明：

（1）主键设置：primary key 表示设置字段为主键。例如在 student 表中，primary key (studentno)表示将 studentno 字段定义为主键。在 score 表中，primary key (studentno , courseno) 表示把 studentno、courseno 两个列一起作为复合主键。

（2）添加注释：comment'学号'表示对 studentno 字段增加注释“学号”。

（3）字段类型的选择：sex enum('男' ,'女')表示 sex 字段的类型是 enum，取值范围为'男'和'女'。对于取值固定的字段可以设置数据类型为 enum。例如，在 course 表中 type 字段表示课程的类型，一般是固定的几种类型。因此也可以把该字段的定义写成“type enum('必修课', '选修课') default '必修课'”。

（4）默认值的设置：default'男'表示默认值为“男”。

（5）设置精度：score 表中的 daily float(4,1)表示精度为 4，小数位数为 1 位。

（6）如果没有指定是 null 或是 not null，则列在创建时假设为 null。

3. 设置表的属性值自动增加


视频讲解

在 MySQL 数据表中，一个整数列可以拥有一个附加属性 auto_increment。auto_increment 也是一个特殊的约束条件，其主要用于为表中插入的新记录自动生成唯一的序列编码。在默认情况下，该字段的值从 1 开始自增，用户也可以自定义开始值。一个数据表只能有一个字段使用 auto_increment 约束，且该字段必须为主键的一部分。该自增主键的计数器持久化到重做日志中，每次计数器改变，都会将其写入重做日志中。如果数据库重启，InnoDB 会根据重做日志中的信息来初始化计数器的内存值。auto_increment 约束的字段可以是任何整数类型（如 tinyint、smallint、int、bigint 等）。

设置属性值字段增加的基本语法如下：

属性名　数据类型　auto_increment

【例 3.11】 在 teaching 库中创建选修表 sc，选课号 sc_no 是自动增量，选课时间默认为当前时间，其他字段分别是学生学号、课程号和教师号。

程序代码如下：

```
mysql> create table sc
     (  sc_no int(6) not null auto_increment,
        studentno  char(11) not null,
        courseno  char(6) not null,
        teacherno char(6) not null,
        score float(4,1) null,
        sc_time timestamp not null default now(),
        primary key(sc_no)
     );
```

3.4.3　查看表

在数据表创建后，就可以用 show tables 命令查询已创建表的情况，也可以查看表结构，即查看数据库中已存在的表的定义。查看表结构的语句包括 describe 语句和 show create table 语句。通过这两个语句可以查看表的字段名、字段的数据类型、完整性约束条件等。

（1）查看已经创建的表：其命令和运行结果如下。

```
mysql> show tables;
       +--------------------+
       | Tables_in_teaching |
       +--------------------+
       | course             |
       | sc                 |
       | score              |
       | student            |
       | teach_course       |
       | teacher            |
       +--------------------+
       6 rows in set (0.00 sec)
```

（2）查看表基本结构的语句 describe：在 MySQL 中，使用 describe 语句可以查看表的基本定义，其中包括字段名、字段的数据类型、是否为主键和默认值等。

describe 语句的命令和运行结果如下：

```
mysql> describe  student;
      +-----------+----------------+------+-----+--------+-------+
      | Field     | Type           | Null | Key | Default| Extra |
      +-----------+----------------+------+-----+--------+-------+
      | studentno | char(11)       | NO   | PRI | NULL   |       |
      | sname     | char(8)        | NO   |     | NULL   |       |
```

```
| sex       | enum('男','女')  | YES  |     | 男      |       |
| birthdate | date             | NO   |     | NULL    |       |
| entrance  | int(3)           | YES  |     | NULL    |       |
| phone     | varchar(12)      | NO   |     | NULL    |       |
| Email     | varchar(20)      | NO   |     | NULL    |       |
+-----------+------------------+------+-----+---------+------+
7 rows in set (0.05 sec)
```

（3）查看表详细结构的语句 show create table：在 MySQL 中，使用 show create table 语句可以查看表的详细定义，其中包括表的字段名、字段的数据类型、完整性约束条件等信息，除此之外还可以查看表默认的存储引擎和字符编码。

show create table 语句的命令和运行结果（整理格式）如下：

```
mysql> show  create  table  course;
+--------+-------------------------------------------------------+
| Table  | Create Table                                          |
+--------+-------------------------------------------------------+
| course | CREATE TABLE 'course' (
  'courseno' char(6) NOT NULL,   'cname' char(6) NOT NULL,
  'type' char(8) NOT NULL,      'period' int NOT NULL,
  'exp' int NOT NULL,          'term' int NOT NULL,
  PRIMARY KEY ('courseno')
)
 ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci|
+--------+-------------------------------------------------------+
 1 row in set (0.02 sec)
```

说明：

①“ENGINE=InnoDB”表示本表采用的存储引擎是 InnoDB，InnoDB 是 MySQL 在 Windows 平台上默认的存储引擎。

②“CHARSET=utf8mb4”表示本数据库表的字符集是 utf8mb4。

（4）当数据库表创建完毕后，也可以通过安装路径（例如“C:\ProgramData\MySQL\MySQL Server 8.0\Data\teaching”）查看磁盘文件数据库及其包含的数据表文件，如图 3-5 所示。

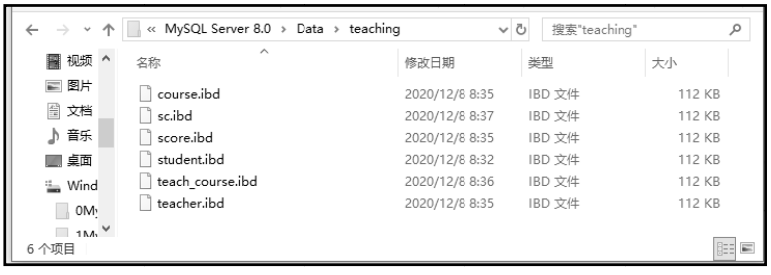

图 3-5 查看数据表文件

从图 3-5 中可以查看数据库 teaching 中创建的各个数据表文件，例如课程表 course、成绩表 score、选修表 sc 等。

3.4.4 修改数据库表


视频讲解

修改表是指修改数据库中已存在的表的定义。修改表比重新定义表简单，不需要重新加载数据，也不会影响正在进行的服务。在 MySQL 中通过 alter table 语句来修改表。修改表包括修改表名、修改字段的数据类型、修改字段名、增加字段、删除字段、修改字段的排列位置、更改默认存储引擎和删除表的外键约束等。

1. 修改表的语法格式

修改数据库表的语法格式如下：

```
alter [ignore] table tbl_name
alter_specification [, alter_specification] …
alter_specification:
add [column] column_definition [first | after col_name ] //添加字段
|alter [column]col_name{set default literal|drop default}//修改字段的默认值
|change [column] old_col_name column_definition        //重命名字段
[first|after col_name]
|modify [column]column_definition[first|after col_name]//修改字段的数据类型
|drop [column] col_name                                //删除列
|rename [TO] new_tbl_name                              //对表重命名
|order by col_name                                     //按字段排序
|convert TO character set charset_name[collate collation_name]
                                                       //将字符集转换为二进制
|[default] character set charset_name [collate collation_name]
                                                       //修改表的默认字符集
```

2. 修改数据库表的示例

alter table 语句用于更改原有表的结构，例如增加或删减字段、重新命名字段或表，以及修改默认字符集。

（1）增加字段：在创建表时，表中的字段就已经定义完成。如果要增加新的字段，可以通过 alter table 语句进行增加。增加表的字段可以实现以下功能：

- 增加无完整性约束条件的字段。
- 增加有完整性约束条件的字段。
- 在表的第一个位置增加字段。
- 在表的指定位置之后增加字段。

【例 3.12】 在 student 表的 Email 列后面增加一列 address。

命令和运行结果如下：

```
mysql> alter table student
    -> add address varchar(30) not null after Email;
Query OK, 0 rows affected (1.63 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结束添加操作后，也可以执行“describe student; ”查看结果。

（2）修改表名：表名可以在一个数据库中唯一确定一张表。数据库系统通过表名来区分不同的表。在 MySQL 中，修改表名是通过 SQL 语句 alter table 实现的。

【例 3.13】 将 sc 表重名为 se_course。

命令和运行结果如下：

```
mysql> alter table sc rename to se_course;
       Query OK, 0 rows affected (0.20 sec)
```

（3）修改字段的数据类型：alter table 语句也可以用来修改字段的数据类型。

【例 3.14】 修改 course 表中的 type 字段，该字段一般取固定值，因此也可以把该字段的定义写成“type enum ('必修, '选修') default '必修'”。

命令和运行结果如下：

```
mysql> alter  table course
    -> modify type enum('必修','选修') default '必修';
       Query OK, 0 rows affected (0.47 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

（4）删除字段：删除字段是指删除已经定义好的表中的某个字段。在 MySQL 中，alter table 语句也可以用来删除表中的字段。

【例 3.15】 删除 student 表中的字段 address。

命令和运行结果如下：

```
mysql> alter table student  drop address;
       Query OK, 0 rows affected (0.21 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

3.4.5　删除数据库表

删除表是指删除数据库中已存在的表。在删除表时会删除表中的所有数据，因此用户在删除表时要特别注意。在 MySQL 中通过 drop table 语句来删除表。

删除表的语法格式如下：

```
drop table  table_name
```

【例 3.16】 在 mytest 数据库中创建 example 表，然后删除 example 表。

代码和运行结果如下：

```
mysql> use mytest;
       Database changed
mysql> create  table  example(
    -> today datetime,
    -> name char(20)
    -> );
       Query OK, 0 rows affected (0.11 sec)
mysql> desc example;
```