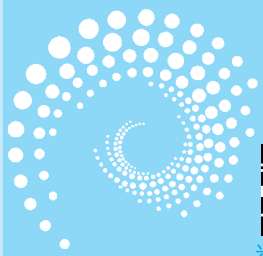
项目五

数据定义

PROJECT 5



兴趣阅读五

情景导入

MySQL 环境搭建完成后，开发组长说：“现在该创建电商平台系统数据库了！我们需要创建一个名为 eshop 的库，里面包含客户表、商品表、订单表等。每个表的字段该怎么定义？比如商品价格用什么数据类型？客户电话的长度多少合适？还要考虑主键、外键约束——这一步直接决定数据的存储方式，必须严谨。”

项目目标

- 掌握 MySQL 的数据类型；理解数据定义语言（DDL）的功能；掌握数据库和表的创建、修改、删除语法；理解完整性约束的作用。
- 能根据业务需求选择合适的数据类型；会用 SQL 创建电商平台系统数据库和表；会添加和删除表字段和约束；能使用 Navicat 可视化定义数据库和表。
- 培养根据业务特性精准定义数据结构的严谨性；树立通过数据约束来保障数据质量的意识；形成 SQL 与可视化工具结合使用的高效习惯。

任务 5.1　数据定义语言

【任务提出】

需要通过语言定义数据库对象（库、表、约束），（数据定义语言）

是 MySQL 中实现这一功能的工具,也是创建和管理数据库结构的基础,掌握其语法才能实现“从设计到落地”的转换。

【知识储备】

数据定义语言是数据库语言的一个子集,专门用于定义、修改和删除数据库结构本身。数据定义语言关注的是数据库的“骨架”或“蓝图”,而不是其中的具体数据内容,它定义了数据的组织方式和存储结构。结构化查询语言(SQL)是与关系数据库交互的标准语言,其中的数据定义语言主要通过 CREATE、ALTER、DROP 及 TRUNCATE 语句来实现操作数据库、表、视图、索引、约束、过程、函数、触发器等对象,使用数据定义语言定义对象时,核心要素是数据类型和完整性约束,它们共同决定了数据的格式、关系和有效性规则。

(1) CREATE 语句:创建新的数据库、表、视图、索引、过程、函数、触发器等数据库对象。

(2) ALTER 语句:修改现有数据库对象的结构。

(3) DROP /TRUNCATE 语句:删除数据库对象(及其包含的所有数据)。

任务 5.2 数据类型

数据类型决定了数据表所存储数据的格式和大小,选择合适的数据类型不仅可以提高数据的存储效率,还能优化查询性能。MySQL 支持多种数据类型,大致可以分为 5 类:数值、字符串、日期与时间、枚举与集合及 JSON 类型。

【任务提出】

不同数据(如商品价格、创建时间)需用合适的类型存储,否则可能导致精度丢失或存储空间浪费。正确选择数据类型是表结构设计的核心,直接影响数据的存储效率和查询准确性。

【知识储备】

5.2.1 数值类型

1. 整数类型

整数类型如表 5-1 所示。

表 5-1 整数类型

类 型	字节	有符号范围	无符号范围	说 明
TINYINT	1	−128～127	0～255	较小整数(如状态码)
SMALLINT	2	−32768～32767	0～65535	小整数值

续表

类　型	字节	有符号范围	无符号范围	说　明
MEDIUMINT	3	−8388608～8388607	0～16777215	中等整数
INT	4	−2147483648～2147483647	0～4294967295	最常用整数
BIGINT	8	-2^{63}～$2^{63}-1$	0～$2^{64}-1$	大整数(如 ID)

可添加 UNSIGNED 关键字定义无符号整数(如 INT UNSIGNED),括号内数字仅表示显示宽度,如 INT(5),不影响存储范围。

2. 浮点数和定点数

浮点数和定点数如表 5-2 所示。

表 5-2　浮点数和定点数

类　型	字节	说　明	示　例
FLOAT(*M*, *D*)	4	单精度浮点数,精度约 7 位	FLOAT(5, 2)→123.45
DOUBLE(*M*, *D*)	8	双精度浮点数,精度约 15 位	DOUBLE(10, 4)
DECIMAL(*M*, *D*)	变长	精确小数(如金融计算)	DECIMAL(10, 2)

M 表示总位数(整数＋小数),*D* 表示小数位数,如 DECIMAL(5, 2) 可存储 123.45,但不可存储 1234.5(超出总位数)。浮点数相对于定点数的优点是在长度一定的情况下,浮点数类型的取值范围大,但是不精准,适用于需要取值范围大,又可以容忍微小误差的科学计算场景(比如计算化学、分子建模、流体动力学等)。定点数类型的取值范围相对小,但是精准且无误差,适合对精度要求极高的场景(比如涉及金额计算的场景)。

5.2.2　字符串类型

1. 短文本和定长字符串

短文本和定长字符串如表 5-3 所示。

表 5-3　短文本和定长字符串

类　型	最 大 长 度	说　明
CHAR(*N*)	255 字符	定长字符串(空格填充),适合固定长度(如手机号)
VARCHAR(*N*)	65535 字节	变长字符串(节省空间),需预留长度(如用户名)

N 表示字符数(与字符集相关,如 utf8mb4 中 1 字符＝4 字节),VARCHAR 实际占用空间 ＝ 字符串长度 ＋ 1～2 字节(长度记录,长度大于 255 时用 2 字节)。字符类型的数据通常被放在一对单引号('')中。

2. 长文本类型

长文本类型如表 5-4 所示。

表 5-4 长文本类型

类 型	最大长度	特 点
TINYTEXT	255B	短文本
TEXT	64KB(65535 字节)	常用长文本(文章内容)
MEDIUMTEXT	16MB	较大文本(如书籍)
LONGTEXT	4GB	超大文本

3. 二进制数据

二进制数据如表 5-5 所示。

表 5-5 二进制数据

类 型	说 明
BINARY(*N*)	定长二进制数据
VARBINARY(*N*)	变长二进制数据
BLOB	存储图片、视频、音频、附件等二进制数据

5.2.3 日期与时间类型

日期与时间类型如表 5-6 所示。

表 5-6 日期与时间类型

类型	格 式	范 围	示 例
DATE	YYYY-MM-DD	1000-01-01～9999-12-31	'2023-10-01'
TIME	HH:MM:SS[.微秒]	−838:59:59～838:59:59	'14:30:00'
DATETIME	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00～9999-12-31 23:59:59	'2023-10-01 14:30:00'
TIMESTAMP	同 DATETIME	1970-01-01 00:00:01 UTC～2038-01-19 03:14:07 UTC	自动时区转换
YEAR	YYYY	1901～2155	'2023'

TIMESTAMP 占用 4 字节,它会自动转换时区,根据用户的时区不同,显示不同的结果,底层存储的是距离 1970-1-1 0:0:0 0 的毫秒值,两个日期比较大小或日期计算时,TIMESTAMP 更方便、更快。DATETIME 占用 8 字节,存储时间的字面值。

5.2.4 枚举与集合类型

枚举与集合类型如表 5-7 所示。

表 5-7　枚举与集合类型

类　　型	说　　明	示　　例
ENUM('v1','v2',...)	单选字符串值	ENUM('Red','Green','Blue')
SET('v1','v2',...)	多选字符串值(逗号分隔)	SET('Music','Movie','Sport')

ENUM 和 SET 效率较高,但扩展性差(修改需重建表)。

5.2.5　JSON 类型

JSON 类型存储结构化的 JSON 数据,支持路径查询(如 SELECT data->'$.user.name')和自动验证 JSON 格式。

任务 5.3　数据库的定义

【任务提出】

电商平台系统需要独立的数据库 eshop 存储所有业务数据,并定义其字符集、排序规则等。数据库是表的容器,合理定义数据库参数能避免后续出现数据存储问题。

【知识储备】

5.3.1　数据库

数据库是一个逻辑容器,用于组织和存储一组相关的数据表、视图、索引、存储过程、函数、触发器等数据库对象。数据库使数据管理更有条理,将不同应用或模块的数据分隔开,避免命名冲突和意外访问,同时还可以以数据库为单位对用户或角色授予权限。

MySQL 中每个数据库在文件系统中通常对应一个目录,该目录中包含该库所有表的结构文件(如.frm)和数据/索引文件(如.ibd)。MySQL 安装后默认会创建 information_schema、mysql、performance_schema 及 sys 数据库,它们用于存储元数据、权限信息和性能指标等核心系统数据。

数据库通过存储引擎执行数据的创建、查询、更新和删除操作。MySQL 提供了多种存储引擎选项,其中 InnoDB、MyISAM 和 Memory 较为常用。

InnoDB:作为 MySQL 的默认引擎,它提供了事务支持、自动增长列(AUTO_INCREMENT)、外键约束(FOREIGN KEY)以及行级锁定功能,并具备事务提交与回滚、崩溃恢复以及多版本并发控制(MVCC)能力。

MyISAM:该引擎以较高的插入和查询速度见长,但其不足之处在于不支持事务处理。

Memory:此类型引擎将表数据完全存储在内存中,因此能提供极快的数据访问速度。

5.3.2 创建数据库

MySQL 创建数据库的基本语法是:

```
CREATE {DATABASE|SCHEMA}[IF NOT EXISTS] database_name
[CHARACTER SET [=] charset_name]
[COLLATE [=]collation_name];
```

[](方括号):标记可选项。省略时,MySQL 使用默认值。

{}(花括号):标记必填项。必须提供。

|(竖线):表示"或",在多个选项中选择其一。

;(分号):语句终止符,标记语句结束。

语句主体(CREATE DATABASE|SCHEMA):两者功能相同。MySQL 对关键字的大小写不敏感,但建议统一使用大写以增强可读性。

IF NOT EXISTS:可选子句。仅在目标数据库不存在时执行创建,避免因重名报错(仅产生警告);省略时若数据库已存在则报错并中断。

database_name:指定新建数据库的名称。需符合操作系统文件夹的命名规则及 MySQL 标识符规范(推荐小写字母、数字和下画线组合,避免保留字和特殊字符)。建议使用具有实际意义的名称。可使用反引号(`)包裹名称,避免与关键字冲突。

charset_name:指定数据库的默认字符集(如 utf8mb4, latin1, gbk)。强烈推荐使用 utf8mb4(支持完整的 Unicode,包括表情符号 Emoji)。

collation_name:设定数据库的默认排序规则(例如 utf8mb4_general_ci、utf8mb4_0900_ai_ci、utf8mb4_bin)。该规则定义了字符的排序和比较方式,包括是否区分大小写(如 ci 表示不区分)、重音等属性。排序规则直接影响查询结果的顺序以及字符串比较、连接等操作的行为。若指定了 CHARACTER SET 但省略 COLLATE,则使用该字符集的默认排序规则。若两者均省略,则采用服务器级别的默认设置(可在 my.ini/my.cnf 中配置)。

创建 mydb 数据库,默认字符集为 utf8mb4,默认排序规则为 utf8mb4_general_ci 的 SQL 语句:

```
CREATE DATABASE IF NOT EXISTS `mydb`
CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
```

5.3.3 查看数据库

查看当前服务器所有数据库:

```
SHOW DATABASES;
```

查看当前使用的数据库:

```
SELECT DATABASE();
```

选定使用的数据库：

```
USE database_name;
```

database_name 是要选定使用或切换的数据库的名称，后续的操作都会在选定的数据库上执行。

查看数据库的创建语句：

```
SHOW CREATE DATABASE database_name;
```

database_name 是要显示的数据库的名称。

查看系统默认字符集：

```
SHOW VARIABLES LIKE 'character_set_database';
```

查看系统默认排序规则：

```
SHOW VARIABLES LIKE 'collation_database';
```

查看数据库支持的字符集：

```
SHOW CHARSET;
```

查看数据库支持的排序规则：

```
SHOW COLLATION;
```

查看数据库的进程列表：

```
SHOW PROCESSLIST;
```

查看数据库所有支持的引擎：

```
SHOW ENGINES;
```

5.3.4　修改数据库

修改数据库主要用于修改字符集和排序规则，其语法如下：

```
ALTER DATABASE database_name
[CHARACTER SET [=] charset_name]
[COLLATE [=] collation_name];
```

database_name：要修改的数据库名称。

charset_name：需要指定的字符集名称。

collation_name：字符集排序规则名称。

提示：数据库修改后会影响之后在该数据库中新创建的对象(如表)的默认设置，但不会改变已有对象的字符集/排序规则。

5.3.5 删除数据库

删除数据库的语法如下：

```
DROP DATABASE [IF EXISTS] database_name;
```

IF EXISTS：可选，避免删除不存在的数据库时报错。

database_name：要删除的数据库的名称。

提示：删除数据库属于危险操作，会删除数据库及其包含的所有对象(表、视图、存储过程等)和其中的数据。

【任务实施】

5.3.6 电商平台系统数据库的定义

1. 创建电商平台系统数据库

为电商平台系统创建数据库 eshop，字符集编码和排序规则使用默认值。SQL 语句如下：

```
CREATE DATABASE IF NOT EXISTS eshop;
```

2. 修改电商平台系统数据库

修改现有电商平台系统数据库，使其字符集编码为 utf8mb4，排序规则为 utf8mb4_general_ci。SQL 语句如下：

```
ALTER DATABASE eshop
CHARACTER SET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

任务 5.4 数据表的定义

【任务提出】

需将逻辑设计中的关系模型(如分类表、商品表)转换为 MySQL 中的实际表，明确字段、类型和约束。表是数据存储的基本单位，定义表结构是数据库存储数据的前提。

【知识储备】

5.4.1 表

表是数据库中存储数据的基本单元，由列(字段)和行(记录)组成。列就称为字段，每一

列都有特定的名称、数据类型和长度等属性，用于存储特定类型的数据。例如，用户表可能包含“用户名(VARCHAR 类型，用于存储可变长度的字符串，长度可能定义为 50，表示可以存储最多 50 个字符的用户名)”“密码(VARCHAR 类型)”等列。表中的行就是记录，它是一组相关字段的集合，代表一个完整的数据实体。例如，在用户表中，一条记录可能包含一个用户名和密码等信息。

5.4.2　创建数据表

创建数据表的 SQL 语句的语法如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name (
    column1_name datatype [constraints] [COMMENT 'string' ],
    column2_name datatype [constraints] [COMMENT 'string' ],
    ...
    columnN_name datatype [constraints] [COMMENT  'string' ],
    [table_constraints]
) [CHARACTER SET [=] charset_name]
[COLLATE [=] collation_name]
[ENGINE  [=] engine_name]
[COMMENT [=] 'string'];
```

,(逗号)：表示定义项之间的分隔符，最后一项后面不能有逗号。

...(省略号)：表示和前面形式相同的更多项。

''(单引号)：字符或字符串常量界定符。

TEMPORARY：有这个关键字表示创建的是临时表，否则就是永久表。

table_name：要创建的表的名称。

column1_name、column2_name、columnN_name 等：表中列的具体名称。

datatype：每列的数据类型。如 INT、VARCHAR、DATE 等。

constraints：列级约束。该约束是在定义列的时候直接指定的约束，这些约束仅作用于特定的列。

IF NOT EXISTS：创建的表的名称不存在才执行创建表语句。

CHARACTER SET[=]charset_name：指定表的字符集，不指定情况下将使用 MySQL 的默认字符集。

COLLATE[=]collation_name：指定表的字符集的校对规则，不指定情况下将使用 MySQL 的默认字符集的校对规则。

ENGINE[=]engine_name：指定表的存储引擎，不指定情况下将使用 MySQL 的默认存储引擎。

COMMENT[=]'string'：注释。如列或表说明，string 为说明文字。

table_constraints：表级约束。表级约束是指在所有列定义完成后单独定义的约束，可以涉及一个或多个列，通常用于定义涉及多个列的复杂约束。

提示：为安全起见，建议将列名、表名等自定义的名称用反引号(`)括起来，防止名称因包含某些特殊字符或关键字而出问题。

假设项目三中客户实体和账户实体之间的 1∶1 关系在转换时与客户实体合并,则客户关系模式为:

客户(客户编号,客户名称,客户邮箱,客户电话,创建时间,最后登录时间,是否活跃,客户年龄,客户积分,客户等级,客户账户编号);

"客户账户编号"是客户关系模式的键,作为账户关系模式的外键。

SQL 建表语句如下:

```
CREATE TABLE customers (
  customer_no INT COMMENT '客户编号',
  customer_name VARCHAR (50) COMMENT '客户名称',
  email VARCHAR (100) COMMENT '客户邮箱',
  phone CHAR(11) COMMENT '客户电话',
  created_at DATETIME COMMENT '创建时间',
  last_login TIMESTAMP COMMENT '最后登录时间',
  is_active TINYINT (1) COMMENT '是否活跃',
  age TINYINT UNSIGNED COMMENT '客户年龄',
  points MEDIUMINT COMMENT '客户积分',
  membership_level ENUM ('bronze', 'silver', 'gold') COMMENT '客户等级',
account_no INT COMMENT '客户账户编号'
  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci COMMENT =
'客户';
```

上述创建了一个客户表(customers)用于保存客户的基本信息,每列都声明了类型,用于存储对应类型的数据,数据的正确性和可靠性得到了一定的保障,但还存在一些问题,如客户编号怎么保证唯一性、客户名称怎么保证不能为空等,要解决这些问题需要加上 constraints(列级约束)或者 table_constraints(表级约束)对表进行完整性约束定义。

5.4.3 完整性约束

完整性约束(Integrity Constraint) 简称约束,是数据库管理系统中用于强制保证数据准确性、有效性和一致性的一组规则。它定义了数据在数据库表中必须满足的条件,确保数据的质量符合业务逻辑和现实世界的规则。完整性约束是数据库设计的基石,它通过预定义规则、自动强制执行、覆盖增删改操作等方式确保数据始终处于正确、有效、一致的状态,从根源上减少"脏数据"的产生,降低应用层的校验负担。

完整性约束分为列级约束和表级约束。列级约束一般对一个数据列建立约束,表级约束对多个数据列建立约束。表级约束只能在列定义之后声明,列级约束既可以在列定义时声明,也可以在列定义之后声明。

MySQL 中常见的完整性约束有非空约束(NOT NULL)、默认约束(DEFAULT)、唯一约束(UNIQUE)、主键约束(PRIMARY KEY)、外键约束(FOREIGN KEY)、自增约束(AUTO_INCREMENT)、检查约束(CHECK),如表 5-8 所示。

表 5-8 MySQL 中常见的完整性约束

约束类型	关 键 字	作 用
非空约束	NOT NULL	强制指定表中的某一列不能存储 NULL 值,确保关键字段必须有值。注意:NULL 不等于空白字符串或空格