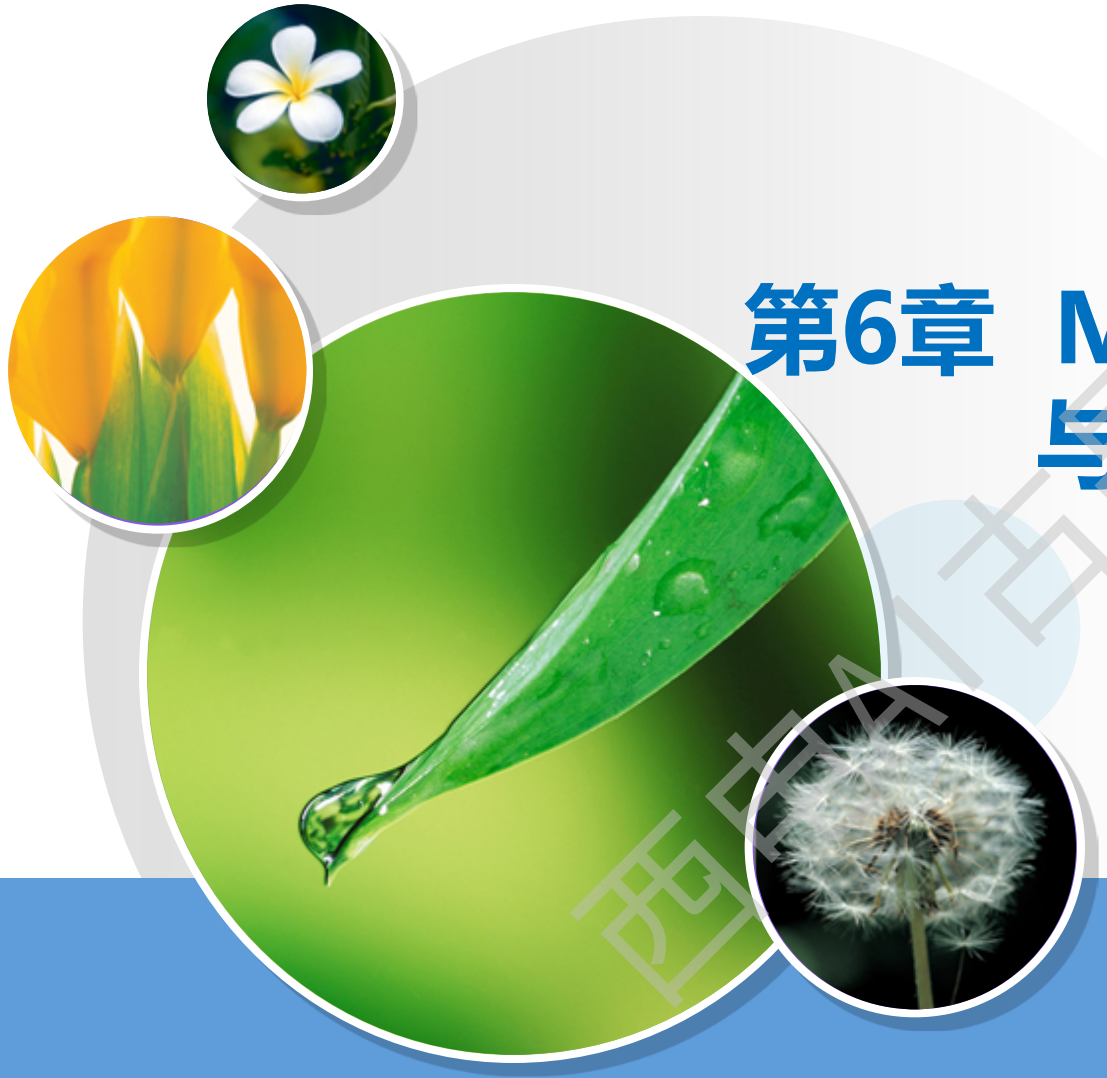


第6章 MySQL存储引擎与数据库操作管理

人工智能学院

古 晶

西安电子科技大学

第6章 MySQL存储引擎与数据库操作管理

- 6.1 存储引擎
- 6.2 字符集
- 6.3 MySQL数据库操作管理
- 6.4 知识点小结
- 本章实验

第6章 MySQL存储引擎与数据库操作管理

- 数据库是存储数据库对象的容器，是指长期存储在计算机内，有组织和可共享的数据的集合。
- 数据库的存储方式有特定的规律。
- **MySQL数据库的管理**主要包括数据库的创建、选择当前操作的数据库、显示数据库结构以及删除数据库等操作。
- 本章将介绍MySQL**存储引擎**与**数据库操作管理**。


第6章 MySQL存储引擎与数据库操作管理

6.1 MySQL存储引擎

- **存储引擎**就是指**表**的**类型**。
- 数据库的存储类型**决定**了表在计算机中的**存储方式**。
- 用户可以根据不同的存储方式、是否进行事务处理等来选择合适的存储引擎。
- 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，也可以根据自己的需要编写自己的存储引擎。
- MySQL提供了**插件式(pluggable)**的存储引擎，存储引擎是基于表的。同一个数据库，不同的表，存储引擎可以不同。甚至，同一个数据库表在不同的场合可以应用不同的存储引擎。
- 建表可以指定存储引擎，可以使用不同引擎，优化和逻辑时选着合适的存储引擎来存储的一种形式。**同一个表可以使用多个存储引擎。**

6.1 MySQL存储引擎

- 1. 查看MySQL支持的存储引擎：**SHOW ENGINES**
- 2. 查看显示支持的存储引擎信息：**SHOW VARIABLES LIKE 'have%'**
- 3. 查看默认的存储引擎：**SHOW VARIABLES LIKE 'storage_engine'**
- 在命令行的界面中，“\g”或“\G”作用与**分号作用相同**，” \G”可以让结果更加美观。比如：SHOW ENGINES **\G;**

6.1 MySQL存储引擎

InnoDB存储引擎：

- 为处理**巨大数据量**时的最大性能设计。
- 完全与MySQL服务器整合，InnoDB存储引擎为在主内存中缓存数据和索引而维持它自己的缓冲池。
- 是**事务（Transaction）安全**的，并且支持**外键**（foreign key）。
- InnoDB表空间分为**共享表空间**与**独享表空间**。其中，表空间是数据库的逻辑划分，一个表空间只能属于一个数据库。所有的数据库对象都存放在指定的表空间中。

6.1 MySQL存储引擎

- InnoDB存储引擎的**特点**：
  - 给MySQL数据库提供事务，包括**回滚**，包括**修复**能力，多版本并发控制事务安全。
  - 创建表结构是存储在“.fim”文件中，数据和索引，数据分别存储在innodb表空间中。
  - **缺点**是读写**读取效率比较低，占用的空间也比较大。**

6.1 MySQL存储引擎

□MyISAM存储引擎：

- 是**默认**存储引擎，它是基于传统的ISAM类型，ISAM是Indexed Sequential Access Method (有索引的顺序访问方法) 的缩写，它是存储记录和文件的标准方法。
- MyISAM具有**检查**和**修复表格**的大多数工具。
- MyISAM表格可以被**压缩**，而且它们**支持全文搜索**。
- 它们不是事务安全的，而且也**不支持外键**。
- 如果事务回滚将造成**不完全回滚**，**不具有原子性**。
- 如果执行**大量的查询操作**时，MyISAM是更好的选择。

6.1 MySQL存储引擎

- MyISAM存储引擎**特点**：
  - 创建表时会存储为3个文件，文件的名字与表名**相同**，分别是 “frm MYD MYI”，**frm存储表结构**，**MYD存储表数据**，**MYI存储表索引**分别保存在这三个文件中。
  - Myisam的表支持三种存储格形分别是：**静态**型，**动态**型，**压缩**型。
  - **静态是myisam默认存储格式**，字段是**固定长度**；动态包括变长字段，字段长度是不固定的；压缩型要选择工具dimpaik创建占用磁盘空间比较小。
  - Myisam的优势在于**磁盘空间比较小**，**处理速度快**。
  - **缺点**是不支持事务，没有事务的完整性，安全性以及事务的并发性处理。

□MEMORY存储引擎：

- 将表中的数据存放在内存中，如果数据库重启或发生崩溃，表中的数据都将消失。
- 非常适合用于存储临时数据的**临时表**，以及数据仓库中的**纬度表**。
- 默认使用**哈希(HASH)索引**，而不是B+树索引。
- **速度非常快**，但在使用上还是有一定的限制：只支持表锁，**并发性能较差**，并且**不支持TEXT和BLOB列类型**。最重要的是，存储变长字段（varchar）时是按照定常字段（char）的方式进行的，因此会**浪费内存**。

6.1 MySQL存储引擎

- MEMORY存储引擎**特点**：
  - 1）MySQL的特殊存储引擎，它是使用在存储下内存中的内容来创建表，而且所有的数据也是存储在内存中**速度快**。
  - 2）基于memory的存储引擎实际对应一个磁盘文件，文件名与表名相同，类型也与ifim但**只存储表结构**，数据存储在内存中有利于快速处理，可以提高表的处理效率。
  - 3）需要注意服务器需要足够的内存来维持数据引擎来使用，不想使用时需要删除不然系统内存不够。
  - 4）**缺点**是数据存储在内存上，如果意外断电或发生意外时**会造成数据丢失**。Memory很少使用到，生命周期比较短，一般都是一次性的，为了我们快速读取。

□MERGE存储引擎：

- **是一组MyISAM表的组合**，这些MyISAM表必须结构完全相同。
- MERGE表**本身没有数据**，对MERGE类型的表可以进行查询、更新、删除操作，实际上是**对内部的MyISAM表**进行的。
- 对MERGE表进行DROP操作，这个操作只是**删除MERGE的定义**，**对内部的表没有任何影响**。

MySQL中的其他存储引擎

(1)**BLACKHOLE存储引擎**

使用场合：

- 验证存储文件**语法**的正确性。
- 来自二进制日志记录的**开销测量**，通过比较，允许二进制日志功能的BLACKHOLE的性能与禁止二进制日志功能的BLACKHOLE的性能。
- 被用来查找与存储和引擎自身不相关的性能瓶颈。

(2)**CSV存储引擎**

其主要用途就是通过数据库中的数据导出成一份**报表文件**。

(3)**ARCHIVE存储引擎**

ARCHIVE 存储引擎主要用于通过较小的存储空间来存放过期的很少访问的历史数据。

MySQL存储引擎的选择

（1）**MyISAM**

适用场景是**不需要事务支持**、**并发相对较低**、**数据修改相对较少**、**以读为主**、**数据一致性要求不是非常高**。

（2）**InnoDB**

适用场景是需要**事务支持**、**行级锁定对高并发有很好的适应能力**，但需要确保查询是通过**索引**完成、数据更新较为**频繁**。

（3）**MEMORY**

适用场景是需要很快的**读写速度**、对数据的**安全性要求较低**。MEMORY存储引擎对表的大小有要求，不能是太大的表。

- 6.1 存储引擎
- **6.2 字符集**
- 6.3 MySQL数据库操作管理
- 6.4 知识点小结
- 本章实验

6.2 MySQL字符集

- **字符集**：一套文字符号及其编码、比较规则的集合。
- MySQL支持的字符集：
  - 默认是**latin1**（西欧ISO_8859_1字符集的别名）
  - latin1字符集是**单字符**编码，而汉字是双字节编码，由此可能导致MySQL数据库不支持中文字符查询或者中文字符乱码等问题。
  - MySQL服务器可以支持多种字符集，在同一台服务器、同一个数据库甚至同一个表的不同字段都可以使用**不相同或相同**的字符集。
  - **查看字符集**的两种方式：
    - **show character set**；
    - 使用**information_schema.character_sets** 表用于查看字符集的详细信息，比如显示所有的字符集和该字符集默认校对规则。

6.2 MySQL字符集

- MySQL字符集包括**字符集**和**校对规则**两个概念。
  - 字符集用来定义MySQL**存储字符串**的方式。
  - 校对规则定义**比较字符串**的方式。
  - 字符集和校对规则是**一对多**的关系，两个不同的字符集不能有相同的校对规则，每个字符集有一个默认校对规则。
  - MySQL支持30多种字符集的70多种校对规则。每个字符集至少对应一个校对规则。
  - 查看相关字符集的**校对规则**的方法：
    - 通过SHOW COLLATION LIKE ’ ***’;命令。
    - 通过系统表information_schema.COLLATIONS来查看。

MySQL字符集的选择

- 选择MySQL字符集的**考虑因素**：

(1) 满足应用支持**语言**的要求，如果应用要处理的语言种类多，要在不同语言的国家发布，就应该选择Unicode字符集，就目前对MySQL来说，选择**utf-8**。

(2) 如果应用中涉及**已有数据的导入**，就要充分考虑数据库字符集对已有数据的兼容性。假若已经有数据是GBK文字，如果选择uft-8作为数据库字符集，就会出现汉字无法正确导入或显示的问题。

MySQL字符集的选择

(3) 如果数据库只需要支持一般**中文**，**数据量很大**，**性能要求也很高**，那就应该选择双字GBK。因为，相对于UTF-8而言，GBK比较“小”，每个汉字占用2个字节，而UTF-8汉字编码需要3个字节，这样可以减少磁盘I/O、数据库Cache以及网络传输的时间。如果主要处理的英文字符，只要少量汉字，那么选择UTF-8比较好。

(4) 如果数据库需要做大量的**字符运算**，如比较、排序等，那么选择定长字符集可能更好，因为定常字符集的处理速度要比变长字符集的处理速度快。

(5) 考虑客户端所使用的**字符集编码格式**，如果所有客户端都支持相同的字符集，则应该优先选择字符集作为数据库字符集。这样可以比避免因字符集转化带来的性能开销和数据损失。

MySQL字符集的设置

- MySQL的字符集和校对规则有**4个级别**的默认设置：
  - **服务器级**字符集和校对规则
  - **数据库级**字符集和校对规则
  - **表级**字符集和校对规则
  - **字段级**字符集和校对规则

服务器级字符集和校对规则

- 查询当前的**服务器**的字符集：
  - 使用” **show variables like 'character_set_server' ;**” 命令。
- 查看**校对规则**：
  - 使用 “**show variables like 'collation_server';**” 命令。
- 服务器字符集和校对规则，可以在MySQL服务启动的时候确定。
  - **例**：若要指定字符集为gbk（校验规则为与其对应的默认的校验规则）
    - 在**my.cnf**配置文件中设置：

      [mysqld]

      character-set-server=gbk;
    - 在**启动时**指定字符集为gbk，命令如下:

      mysqld –character-set-server=gbk

数据库字符集和校验规则

- 数据库的字符集和校验规则在**创建数据库**的时候指定，也可以在创建完数据库后通过**“alter database”命令进行修改**。
- 如果数据库中已经存在数据，因为修改字符集并不能将已有的数据按照新的字符集进行存放，所以**不能通过修改数据库的字符集直接修改数据的内容**。
- 显示当前数据库字符集和校验规则可用以下两条命名分别查看:
  - show variables like ‘character_set_database ‘
  - show variables like ‘collation_database’

数据库字符集和校验规则

- 设置的**规则**：
  - 如果指定了字符集和校对规则，则使用指定的字符集和校对规则：
  - 如果指定了字符集没有指定校对规则，则使用指定字符集的**默认**校对规则；
  - 如果指定了校对规则但未指定字符集，则字符集使用与该校对规则**关联**的字符集；
  - 如果没有指定字符集和校对规则，则使用服务器字符集和校对规则作为数据库的字符和校对规则。

表字符集和校验规则

- 在**创建表**的时候指定；
- 可以通过**alter table**命令进行**修改**；
- 如果表中已有记录，修改字符集对原有的记录并没有影响，不会按照新的字符集进行存放。
- 要显示当前表字符集和校验规则可用以下如下命名查看:
  - show create table 表名;

表字符集和校验规则

- 设置表的字符集的**规则**：
  - 如果指定了字符集和校对规则，使用**指定的**字符集和校对规则；
  - 如果指定了字符集没有指定校对规则，使用指定字符集的**默认**校对规则；
  - 如果指定了校对规则但未指定字符集，则字符集使用与该校对规则**关联**的字符集；
  - 如果没有指定字符集和校对规则，使用**数据库**字符集和校对规则作为表的字符集和校对规则。

列字符集和校验规则

- MySQL可以定义**列级**别的字符集和校对规则，主要是针对相同的表不同字段需要使用不同的字符集的情况。
- 列字符集和校对规则的定义可以在**创建表**时指定，或者在**修改表时调整**，如果在创建表的时候没有特别指定字符集和校对规则，则默认使用表的字符集和校对规则。

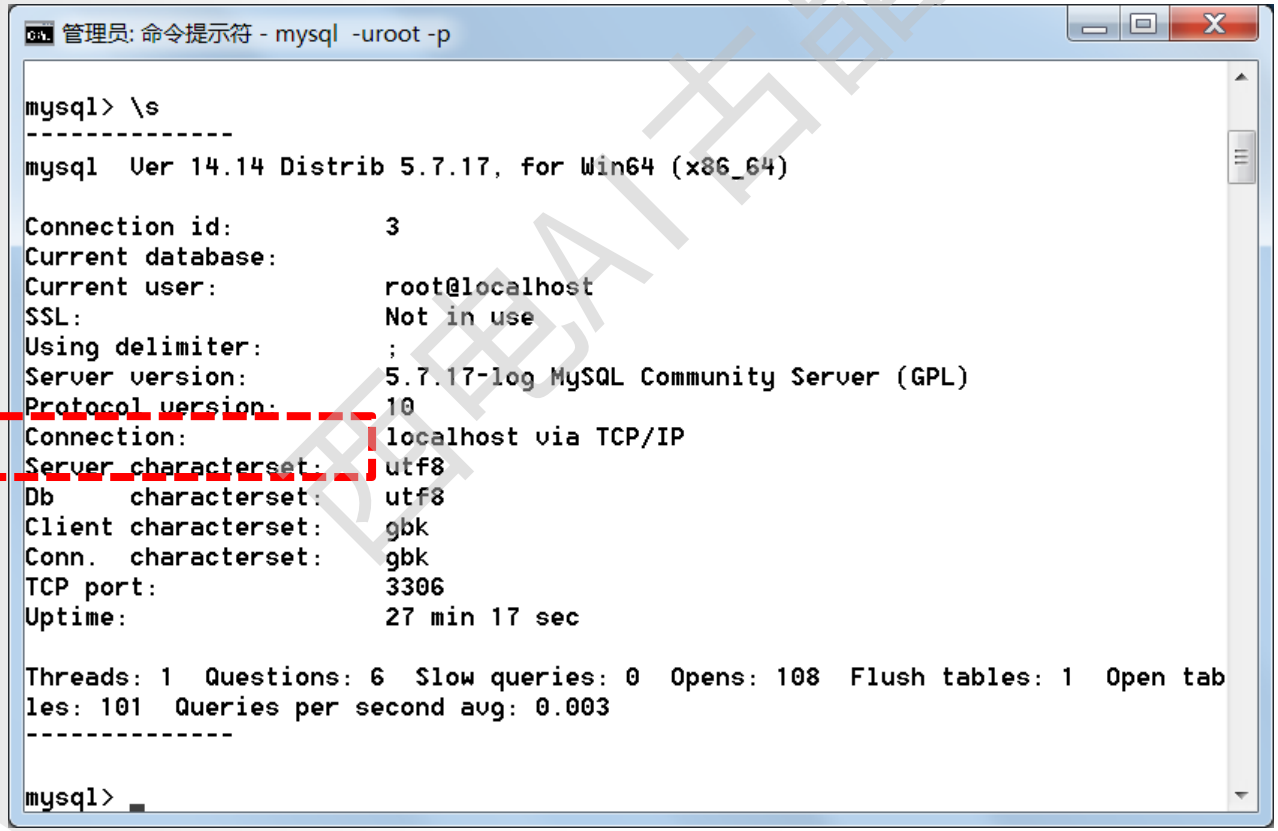
连接字符集和校验规则

- 客户端和服务器之间交互的字符集和校对规则的设置。
- MySQL提供了3个不同的参数：character_set_client、character_set_connection和character_set_results，分别代表**客户端、连接和返回结果的字符集**。通常情况下，**这3个字符集应该是相同的，才可以确保用户写入的数据可以正确地读出**，特别是对于中文字符，不同的写入字符集和返回结果字符集将导致写入的记录不能正确读出。
- 同时修改这3个参数的值：set names ***;命令
- 在my.cnf中设置一下语句

```
[mysql]
default-character-set=gbk;
```

查看MySQL的配置参数

- 可以在登录状态下，用**status**或者**\s**命令查看配置的参数



通过my.ini更改默认字符集

- 将“my-default.ini”复制到另外一个位置，文件名更为“**my.ini**”，双击打开；
- 在 **[mysqld]** 下面写上

  character-set-server=utf8
- 在 **[client]** 下面 加上

  default-character-set=utf8
- 保持后，复制到安装位置，重启MySQL服务才能生效：
  - ✓ 先**关闭服务**，以管理员身份进入控制台，然后输入：**net stop MySQL57**，MySQL的服务就停止了
  - ✓ 再**开启服务**，输入：**net start MySQL57**，启动服务

- 6.1 存储引擎
- 6.2 字符集
- **6.3 MySQL数据库操作管理**
- 6.4 知识点小结
- 本章实验

MySQL数据库操作管理

- 创建、查看、选择数据库以及查看数据库的定义
- 修改数据库的编码方式
- 删除数据库

创建数据库

- **创建数据库**是通过SQL语句**CREATE DATABASE**或**CREATE SCHEMA**命令来实现的。
- 每一个数据库都有一个数据库字符集和一个数据校验规则，**不能够为空**。
- **语法格式**：

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS]db_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFALUT] COLLATE collation_name]

例如：创建schoolDB 数据库，编码为GBK 。

CREATE DATABASE schoolDB DEFAULT CHARACTER SET GBK;

创建数据库

口 说明：

- 数据库名：在文件系统中，MySQL的数据库存储区将以目录方式表示MySQL数据库。因此，命令中的数据库名字必须符合操作系统文件夹命名规则。
- 在MySQL中**不区分大小写**，在一定程度上方便使用。
- 如果指定了CHARACTER SET charset_name 和COLLATE collation_name,那么采用指定的字符集charset_name和校验规则collation_name，如果没有指定，那么会采用**默认**的值。

创建数据库

- 查看**所有**的数据库：
  - 使用**SHOW DATABASES| SCHEMAS;**命令。

  例如： **SHOW DATABASES;**
- 查看数据库的**详细**信息：
  - SHOW CREATE {DATABASE|SCHEMA} 数据库名称 ;

  例如： SHOW CREATE DATABASE test;
- **使用**某个数据库：
  - 使用USE db_name;命令。

  例如：USE test;

修改数据库编码方式

- **修改数据库**名称：
  - ✓ 如果MySQL数据库的存储引擎是MyISAM，那么只要修改DATA目录下的库名文件夹就可以了。
  - ✓ 如果存储引擎是**InnoDB**，是**无法修改**数据库名称的，只能修改字符集和校对规则。
- 修改指定数据库编码方式的语法格式：

  ALTER {DATABASE | SCHEMA} [db_name] [DEFAULT CHARACTER SET charset_name]| [[DEFAULT] COLLATE collation_name]

例：将schoolDB的编码方式有GBK修改为UTF8。

```
ALTER DATABASE schoolDB CHARACTER SET UTF8;
```

- 说明：
  - ➢ ALTER DATABASE用于更改数据库的全局特性，用户必须有数据库**修改权限**，才可以使用ALTER DATABASE修改数据库。

删除数据库

- **删除数据库**是指在数据库系统删除**已经存在**的数据库。
- 删除数据库成功后，原来分配的空间将被收回。在删除数据库时，会删除数据库中的所有的表和所有的数据，因此，删除数据库时需要慎重考虑。
- 如果要删除某个数据库，可以先将该数据库**备份**，然后再进行删除。
- 删除数据库语法格式：

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

例如：删除schoolDB数据库。

```
DROP DATABASE schoolDB；
```

对数据库的常用操作示例总结

- 创建数据库：**CREATE DATABASE mydb;**
- 切换（使用）数据库：**USE mydb;**
- 查询数据库：**SHOW DATABASES;**
- 查看数据库定义：**SHOW CREATE DATABASE mydb;**
- 查看当前打开数据库名称：**SELECT DATABASE();或 SELECT SCHEMA();**
- 删除数据库：**DROP DATABASE mydb;**
- 查看上一步操作产生的警告信息:**SHOW WARNINGS;**

- 6.1 存储引擎
- 6.2 字符集
- 6.3 MySQL数据库操作管理
- **6.4 知识点小结**
- 本章实验

知识点小结

本章知识小结：

- MySQL中的存储引擎的概念和几种常见的存储引擎
- 字符集和校对规则的概念、设置方法
- 数据库的创建、修改、和删除的命令以及使用注意事项

- 6.1 存储引擎
- 6.2 字符集
- 6.3 MySQL数据库操作管理
- 6.4 知识点小结
- **本章实验**

本章实验

- 实验内容：
  （1）MySQL安装；
  （2）数据库创建、查看、修改与删除。

Thanks !

See you