《数据库原理及应用教程（MySQL）》

第4章 数据库的创建和管理

01 OPTION MySQL数据库的存储引擎

02 OPTION MySQL数据库的字符集

03 OPTION MySQL数据库管理

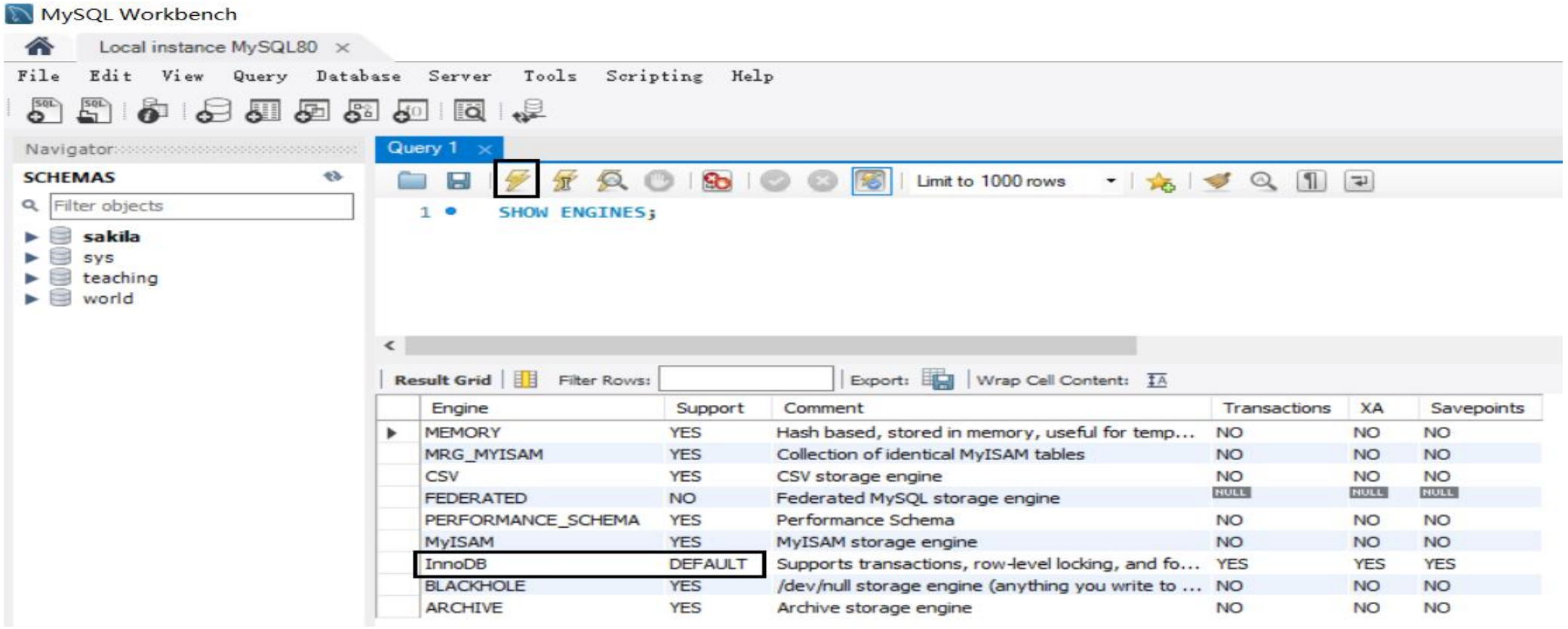
1. 存储引擎概述

- ✓ 存储引擎是决定如何存储数据库中的数据、如何为数据建立索引、如何更新和查询数据的机制
- ✓ MySQL数据库管理系统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常用的存储引擎有InnoDB、MyISAM、MEMORY和MERGE等
- ✓ 可以查看MySQL支持的存储引擎，查看命令如下：

```
SHOW ENGINES;
```

1. 存储引擎概述

✓ 在MySQL Workbench查询窗口输入“SHOW ENGINES;”，单击“执行”按钮，即可查看各存储引擎的相关信息。



Engine	Support	Comment	Transactions	XA	Savepoints
MEMORY	YES	Hash based, stored in memory, useful for temp...	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 fo...	YES	YES	YES
BLACKHOLE	YES	/dev/null storage engine (anything you write to ...	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO

1. 存储引擎概述

- ✓ MySQL的默认存储引擎是InnoDB，如果想把其他存储引擎设置为默认存储引擎，可以使用如下命令：

```
SET DEFAULT_STORAGE_ENGINE=存储引擎名;
```

- ✓ 如果不确定MySQL当前默认的存储引擎，可以使用如下命令：

```
SHOW VARIABLES LIKE '%storage_engine%';
```

- ✓ 在MySQL Workbench中执行上述命令的结果如图所示：

	Variable_name	Value
▶	default_storage_engine	InnoDB
	default_tmp_storage_engine	InnoDB
	disabled_storage_engines	
	internal_tmp_mem_storage_engine	TempTable

存储引擎	类比	主要特点	优点	缺点	典型用途
InnoDB	银行	支持事务、外键、行级锁	数据安全性高；支持事务回滚；支持外键约束；并发性能好	占用空间较大；结构复杂	电商系统（订单、支付）、银行系统、用户系统、需要数据一致性的业务系统
MyISAM	图书馆	查询速度快、表级锁	查询性能好；存储结构简单；占用空间较小；支持全文索引	不支持事务；并发写入性能差；数据安全性较低	新闻网站、博客系统、日志查询系统、以查询为主的系统
MEMORY	内存仓库	数据存储在内存中	读写速度极快；适合高频访问数据	服务器重启数据丢失；内存容量有限；不适合长期存储	临时表、缓存数据、统计中间结果、高速计算数据
MERGE	表合并	多个MyISAM表组合成一个逻辑表	可以统一查询多个表；方便管理大量分表数据	只能用于MyISAM表；使用场景较少	按年份或月份分表的日志系统，例如 `log_2023`、`log_2024` 合并查询
BLACKHOLE	黑洞	写入数据后立即丢弃	可用于性能测试；用于主从复制架构中的数据转发	数据不会保存	测试数据库性能、测试SQL语句、主从复制实验
CSV	Excel文件	数据以CSV文件形式存储	文件格式通用；便于数据导出和交换；可被Excel等软件直接读取	不支持索引；查询效率低；功能简单	数据导出报表、数据交换、与其他软件共享数据
ARCHIVE	档案馆	高压缩存储历史数据	占用空间小；适合长期保存数据	只支持插入和查询；不能更新或删除	历史日志存储、归档数据、很少访问的历史记录

3 压力测试

可以理解为：

> “只测系统速度，不测存储能力”

因为：

BLACKHOLE 不写磁盘。

所以可以测试：

- SQL执行能力
- 网络吞吐
- MySQL处理能力

例如：

```sql
INSERT INTO blackhole_table VALUES (...)
```

每秒可以测试 **几十万次写入。**

2. InnoDB存储引擎

- ✓ MySQL5.5之后，InnoDB是MySQL的默认存储引擎
- ✓ InnoDB是事务型数据库的首选引擎，具有提交、回滚和崩溃修复能力
- ✓ InnoDB提供专门的缓冲池，是为处理巨大数据量时的最大性能设计
- ✓ InnoDB支持外键约束，是 MySQL 上第一个提供外键约束的存储引擎
- ✓ InnoDB存储引擎将表和索引存储在一个表空间中，表空间可以包含多个文件（或原始磁盘分区）

3. MyISAM存储引擎

- ✓ MySQL5.5之前，MyISAM是MySQL的默认存储引擎
- ✓ MyISAM不支持事物处理，也不支持外键约束，但是，MyISAM具有高效的查询速度，插入数据的速度也很快，是在Web、数据仓储等应用环境中最常使用的存储引擎之一
- ✓ MyISAM的修复时间与数据量的多少成正比，随着数据量的增加，MyISAM的恢复能力的性能会变弱
- ✓ MyISAM不提供专门的缓冲池，必须依靠操作系统来管理读取与写入的缓存，因此在某些情况下，其数据访问效率会比InnoDB低
- ✓ 使用MyISAM创建数据库，将生成三个文件。文件的主文件名与表名相同，扩展名包括“.frm”、“.myd”和“.myi”

4. MEMORY存储引擎

- ✓ MEMORY类型的表中的数据存储在内存中，如果数据库重启或者发生崩溃，表中的数据都将消失
- ✓ MEMORY类型的表适用于暂时存放数据的临时表、作为统计操作的中间表，以及数据仓库中的维度表
- ✓ 每个MEMORY类型的表对应于一个文件，其主文件名与表名相同，扩展名为“.frm”，该文件只存储数据表的定义，而数据表中的数据存储在内存中，这样可以有效地提高数据的处理速度
- ✓ MEMORY默认使用哈希（HASH）索引

5. 其他存储引擎

✓ MERGE存储引擎

- MERGE存储引擎是一组具有相同结构的MyISAM表的组合。MERGE表本身没有数据，对MERGE表可以进行查询、更新和删除操作，这些操作实际上是对内部的MyISAM表进行的

✓ BLACKHOLE存储引擎

- BLACKHOLE存储引擎可以用来验证存储文件语法的正确性。还可以对二进制日志记录进行开销测量，通过比较，允许与禁止二进制日志功能的BLACKHOLE的性能。可以用来查找与存储和引擎自身不相关的性能瓶颈

✓ CSV存储引擎

- CSV存储引擎实际上操作的是一个标准的CSV文件，不支持索引。CSV文件是很多软件都支持的较为标准的格式，当需要把数据库中的数据导出成一份报表文件时，可以先在数据库中建立一张CVS表，然后将生成的报表信息插入到该表，得到CSV报表文件

✓ 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储过期的很少访问的历史数据

6. MySQL存储引擎的选择

✓ 在实际工作中，可以根据应用场景的不同，对各种存储引擎的特点进行对比和分析，选择适合的存储引擎

- 如果实际应用需要事物处理，在并发操作时要求保持数据的一致性，而且除了查询和插入操作，还经常要进行更新和删除操作，这种情况可以选择InnoDB，可以有效降低更新和删除操作导致的锁定，并且可以确保事务的完整性提交和回滚
- 如果实际应用不需要事物处理，以查询和插入操作为主，更新和删除操作较少，并且对事物的完整性和并发性要求不是很高，可以选择MyISAM
- 如果实际应用不需要事物处理，需要很快的读写速度，并且对数据的安全性要求较低，可以选择MEMOEY。它对表的大小有要求，不能建立太大的表。所以，MEMORY适用于创建相对较小的数据库表

01 OPTION MySQL数据库的存储引擎

02 OPTION MySQL数据库的字符集

03 OPTION MySQL数据库管理

1. MySQL字符集概述

- ✓ 针对数据的存储，MySQL提供了多种字符集
- ✓ 针对同一字符集内字符之间的比较，MySQL提供了与之对应的多种校对规则
- ✓ 一个字符集对应至少一种校对规则（通常是一对多的关系），两个不同的字符集不能有相同的校对规则，而且，每个字符集都设置默认的校对规则
- ✓ 可以通过如下命令查看MySQL支持的所有字符集：

```
SHOW CHARACTER SET;
```

- ✓ 或者使用系统表infromation_schema中的CHARACTER_SETS，如下：

```
use information_schema;
SELECT * FROM CHARACTER_SETS;
```

1. MySQL字符集概述

✓ 在MS DOS窗口或者MySQL Shell窗口执行上述命令，可以得到如图所示的MySQL8.0字符集列表

Charset	Description	Default collation	Maxlen
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
ascii	US ASCII	ascii_general_ci	1
big5	Big5 Traditional Chinese	big5_chinese_ci	2
binary	Binary pseudo charset	binary	1
cp1250	Windows Central European	cp1250_general_ci	1
cp1251	Windows Cyrillic	cp1251_general_ci	1
cp1256	Windows Arabic	cp1256_general_ci	1
cp1257	Windows Baltic	cp1257_general_ci	1
cp850	DOS West European	cp850_general_ci	1
cp852	DOS Central European	cp852_general_ci	1
cp866	DOS Russian	cp866_general_ci	1
cp932	SJIS for Windows Japanese	cp932_japanese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
eucjpms	UJIS for Windows Japanese	eucjpms_japanese_ci	3
euckr	EUC-KR Korean	euckr_korean_ci	2
gb18030	China National Standard GB18030	gb18030_chinese_ci	4
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
geostd8	GEOSTD8 Georgian	geostd8_general_ci	1
greek	ISO 8859-7 Greek	greek_general_ci	1
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
hp8	HP West European	hp8_english_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
latin7	ISO 8859-13 Baltic	latin7_general_ci	1
macce	Mac Central European	macce_general_ci	1
macroman	Mac West European	macroman_general_ci	1
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
swe7	7bit Swedish	swe7_swedish_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
ucs2	UCS-2 Unicode	ucs2_general_ci	2
ujis	EUC-JP Japanese	ujis_japanese_ci	3
utf16	UTF-16 Unicode	utf16_general_ci	4
utf16le	UTF-16LE Unicode	utf16le_general_ci	4
utf32	UTF-32 Unicode	utf32_general_ci	4
utf8	UTF-8 Unicode	utf8_general_ci	3
utf8mb4	UTF-8 Unicode	utf8mb4_0900_ai_ci	4

1. MySQL字符集概述

- ✓ 通过如下命令可以查看MySQL支持的所有校对规则：

```
SHOW COLLATION;
```

- ✓ 或者使用系统表infromation_schema中的COLLATIONS，如下：

```
USE information_schema;
SELECT * FROM COLLATIONS;
```

- ✓ 如果需要查看某一种特定的字符集，例如utf8字符集的校对规则，可以使用如下命令：

```
SHOW COLLATION WHERE Charset = 'utf8';
```

- ✓ 或者使用系统表infromation_schema中的COLLATIONS，如下：

```
USE information_schema;
SELECT * FROM COLLATIONS WHERE CHARACTER_SET_NAME = 'utf8';
```

这些命令的本质作用是：

> **查看 MySQL 中有哪些"字符排序规则（Collation）"。**

简单理解：

- **字符集（Charset）**：决定 **文字怎么存储**
- **校对规则（Collation）**：决定 **文字怎么比较和排序**

例如：

```
apple
Apple
```

有的规则认为它们一样
有的规则认为不一样。

这些命令就是 **用来查看数据库支持哪些排序规则。**

命令	做什么	通俗理解	典型用途
`SHOW COLLATION;`	显示 MySQL 支持的所有排序规则	查看数据库所有"文字排序方式"	了解数据库支持哪些排序规则
`USE information_schema; SELECT * FROM COLLATIONS;`	从系统表查看所有排序规则	通过系统数据库查看详细信息	查看更完整的排序规则信息
`SHOW COLLATION WHERE Charset='utf8';`	查看某个字符集的排序规则	只看 **utf8字符集有哪些排序方式**	选择适合的utf8排序规则
`SELECT * FROM COLLATIONS WHERE CHARACTER_SET_NAME='utf8';`	从系统表查询utf8的排序规则	用SQL方式筛选utf8排序规则	数据库管理员或开发者查询

1. MySQL字符集概述

- ✓ 在MS DOS窗口或者MySQL Shell窗口执行上述命令，可以得到如图所示的utf8字符集的校对规则
- ✓ 可以看出，utf8字符集的校对规则有28个，其中，“utf8_general_ci”是默认校对规则。
- ✓ “utf8_general_ci”结尾的“ci”表示大小写不敏感；如果是“cs”，表示大小写敏感；如果是“bin”，表示按编码值比较。

COLLATION_NAME	CHARACTER_SET_NAME	ID	IS_DEFAULT	IS_COMPILED	SORTLEN	PAD_ATTRIBUTE
utf8_general_ci	utf8	33	Yes	Yes	1	PAD SPACE
utf8_tolower_ci	utf8	76		Yes	1	PAD SPACE
utf8_bin	utf8	83		Yes	1	PAD SPACE
utf8_unicode_ci	utf8	192		Yes	8	PAD SPACE
utf8_icelandic_ci	utf8	193		Yes	8	PAD SPACE
utf8_latvian_ci	utf8	194		Yes	8	PAD SPACE
utf8_romanian_ci	utf8	195		Yes	8	PAD SPACE
utf8_slovenian_ci	utf8	196		Yes	8	PAD SPACE
utf8_polish_ci	utf8	197		Yes	8	PAD SPACE
utf8_estonian_ci	utf8	198		Yes	8	PAD SPACE
utf8_spanish_ci	utf8	199		Yes	8	PAD SPACE
utf8_swedish_ci	utf8	200		Yes	8	PAD SPACE
utf8_turkish_ci	utf8	201		Yes	8	PAD SPACE
utf8_czech_ci	utf8	202		Yes	8	PAD SPACE
utf8_danish_ci	utf8	203		Yes	8	PAD SPACE
utf8_lithuanian_ci	utf8	204		Yes	8	PAD SPACE
utf8_slovak_ci	utf8	205		Yes	8	PAD SPACE
utf8_spanish2_ci	utf8	206		Yes	8	PAD SPACE
utf8_roman_ci	utf8	207		Yes	8	PAD SPACE
utf8_persian_ci	utf8	208		Yes	8	PAD SPACE
utf8_esperanto_ci	utf8	209		Yes	8	PAD SPACE
utf8_hungarian_ci	utf8	210		Yes	8	PAD SPACE
utf8_sinhala_ci	utf8	211		Yes	8	PAD SPACE
utf8_german2_ci	utf8	212		Yes	8	PAD SPACE
utf8_croatian_ci	utf8	213		Yes	8	PAD SPACE
utf8_unicode_520_ci	utf8	214		Yes	8	PAD SPACE
utf8_vietnamese_ci	utf8	215		Yes	8	PAD SPACE
utf8_general_mysql500_ci	utf8	223		Yes	1	PAD SPACE

情况1： `utf8mb4_general_ci`

`ci` = case insensitive（不区分大小写）

```sql
SELECT * FROM t ORDER BY name;
```

排序可能是：

```
apple
Apple
banana
Banana
```

数据库认为：

```
apple = Apple
```

情况2： `utf8mb4_bin`

`bin` = binary（按二进制比较）

排序可能是：

```
Apple
Banana
apple
banana
```

因为 ASCII：

```
A = 65
a = 97
```

所以大写在前。

`_ci`	case insensitive（不区分大小写）
`_cs`	case sensitive（区分大小写）
`_bin`	二进制比较

表:

```SQL
CREATE TABLE users(
    name VARCHAR(50)
) COLLATE utf8mb4_general_ci;
```

插入:

```SQL
INSERT INTO users VALUES ('Tom');
INSERT INTO users VALUES ('tom');
```

查询:

```SQL
SELECT * FROM users WHERE name = 'tom';
```

结果:

```
Tom
tom
```

因为:

```
general_ci 不区分大小写
```

2. MySQL字符集设置

- ✓ MySQL对于字符集的设置分为4个级别：服务器（Server）、数据库（DataBase）、数据表（Table）和连接（Connection）
- ✓ 可以查看MySQL字符集在各个级别上的默认设置，查看命令如下：

```
SHOW VARIABLES LIKE 'character%';
```

- ✓ 在MS DOS窗口或者MySQL Shell窗口执行上述命令，可以得到如图所示的各个级别的默认字符集

```
+--------------------------+----------------------------------------------------+
| Variable_name            | Value                                              |
+--------------------------+----------------------------------------------------+
| character_set_client     | gbk                                                |
| character_set_connection | gbk                                                |
| character_set_database   | utf8                                               |
| character_set_filesystem | binary                                             |
| character_set_results    | gbk                                                |
| character_set_server     | utf8mb4                                            |
| character_set_system     | utf8                                               |
| character_sets_dir       | C:\Program Files\MySQL\MySQL Server 8.0\share\charsets\ |
+--------------------------+----------------------------------------------------+
```

这些 **字符集级别（Server、Database、Table、Connection）** 的作用，本质上是：

> **确保数据库在存储、读取、传输文字时不会出现乱码。**

MySQL在不同层级都可以设置字符集，是为了 **控制不同范围的数据编码方式。**

级别	控制范围	通俗理解	典型用途
Server（服务器级）	整个MySQL服务器	整个数据库系统默认使用什么编码	新建数据库时默认使用的字符集
Database（数据库级）	某一个数据库	该数据库默认使用什么编码	创建数据库时指定编码
Table（表级）	某一个数据表	表中数据使用什么编码	不同表存不同语言数据
Connection（连接级）	客户端连接	客户端与数据库通信使用什么编码	防止程序连接数据库时乱码

你写入一句话：

```
你好世界
```

这个过程会经过[illegible]

1 客户端发送（

程序发送：

```
你好世界
```

如果客户端是 **UT**

数据库是 **Latin1**

就可能乱码。

2 数据库存储（Table / Database）

数据库需要决定：

数据用什么编码存。

例如：

```
utf8
utf8mb4
gbk
latin1
```

3 默认继承关系

如果没有指定：

```
Server
 ↓
Database
 ↓
Table
 ↓
Column
```

会 **逐级继承默认字符集。**

2. MySQL字符集设置

- ✓ 可以单独查看某个特定级别的字符集默认设置，例如，查看服务器级的字符集默认设置的命令如下：

```
SHOW VARIABLES LIKE 'character_set_server';
```

- ✓ 可以查看MySQL校对规则在各个级别上的默认设置，查看命令如下：

```
SHOW VARIABLES LIKE 'collation%';
```

- ✓ 在MS DOS窗口或者MySQL Shell窗口执行上述命令，可以得到如图所示的各个级别的默认校对规则

```
+----------------------+----------------------+
| Variable_name        | Value                |
+----------------------+----------------------+
| collation_connection | gbk_chinese_ci       |
| collation_database   | utf8_general_ci      |
| collation_server     | utf8mb4_0900_ai_ci   |
+----------------------+----------------------+
```

这个命令：

```
SQL
SHOW VARIABLES LIKE 'character%';
```

作用是：

> **查看当前MySQL所有字符集设置。**

例如常见结果：

变量	含义
character_set_server	服务器默认字符集
character_set_database	当前数据库字符集
character_set_connection	连接字符集
character_set_client	客户端字符集
character_set_results	查询结果字符集

3. MySQL字符集常见问题

✓ 在数据库系统开发中，MySQL乱码一直是困扰开发者的主要问题。主要表现为：

- 数据录入时为正常编码数据，但存入的数据库的数据后呈现乱码数据
- 数据库中存储的是正常编码的数据，但读取后的数据呈现乱码形态

3. MySQL字符集常见问题

✓ 可以从数据流向的角度，分析出现上述乱码问题的主要原因：

- 数据输入端问题：在终端对用户录入的数据进行编码时，如果选择了与数据存储端不同的编码方式，则在传输后对数据进行解码过程时导致数据出现乱码
- 网络问题：对于在线运行的数据库系统，可能因网络服务中断、网络服务质量不可靠等原因，出现数据接收不完整等现象，如果接收端不对数据的完整性进行校验，会导致数据库中存储了编码不完整的数据
- 数据存储端问题：数据存储端主要是运行在服务器或者本地系统中的数据库，数据库存储的编码涉及多个层面，例如，若数据库管理系统采用Latin编码，而数据库层面未设置默认编码，则会继承使用数据库管理系统的编码，导致当存储中文数据时，会出现乱码

01 OPTION MySQL数据库的存储引擎

02 OPTION MySQL数据库的字符集

03 OPTION MySQL数据库管理

1. 创建数据库

✓ 在MySQL中，创建数据库的语法格式如下：

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

【例4-1】创建名称为teaching的数据库，设置默认字符集为utf8mb4，设置默认校对规则为utf8mb4_0900_ai_ci。

```
CREATE DATABASE IF NOT EXISTS teaching
DEFAULT CHARACTER SET utf8mb4
DEFAULT COLLATE utf8mb4_0900_ai_ci;
```

1. COLLATE 的作用

`COLLATE` 用于定义 **字符的比较规则**，主要影响：

- **字符串排序（ORDER BY）**
- **字符串比较（=、LIKE、>、<）**
- **是否区分大小写**
- **是否区分重音符号**

SQL

```
CREATE DATABASE db1
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
```

意思是：

创建数据库 `db1`

并设置默认：

- 字符编码：`utf8mb4`
- 字符排序规则：`utf8mb4_general_ci`

之后在该数据库创建的表，如果不指定，就会继承这个规则。

2. 查看数据库

- ✓ 查看数据库的语法格式如下：

```
SHOW CREATE DATABASE db_name;
```

- ✓ 创建完数据库之后，可以在MySQL安装盘符下的“\ProgramData\MySQL\MySQL Server 8.0\Data”文件下看到以数据库名称命名的文件夹，该文件夹最初是空文件夹，之后，在数据库中创建的数据表等的相关文件会存储在该文件夹中

例如，查看数据库teaching，在MS DOS窗口或者MySQL Shell窗口执行以下命令

```
SHOW CREATE DATABASE teaching;
```

可以得到如图所示的执行结果，其中展示了数据库teaching的创建命令和参数设置。

```
+----------+-----------------------------------------------------------------------------------------+
| Database | Create Database                                                                         |
+----------+-----------------------------------------------------------------------------------------+
| teaching | CREATE DATABASE `teaching` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ |
+----------+-----------------------------------------------------------------------------------------+
```

3. 修改数据库

- ✓ 数据库创建之后，可以根据需要修改数据库的参数
- ✓ 如果MySQL的默认存储引擎是InnoDB，无法修改数据库名，只能修改字符集和校对规则
- ✓ 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
```

- ✓ 需要注意，用户必须有数据库的修改权限，才能使用ALTER DATABASE命令修改数据库

【例4-2】将数据库teaching的默认字符集修改为gbk，默认校对规则修改为gbk_chinese_ci。

```
ALTER DATABASE teaching
DEFAULT CHARACTER SET gbk
DEFAULT COLLATE gbk_chinese_ci;
```

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，删除成功之后，原来分配的空间将被收回
- ✓ 如果数据库中已经包含了数据表和数据，则删除数据库时，这些内容也会被删除，因此，删除数据库之前最好先对数据库进行备份操作。
- ✓ 删除数据库的语法格式如下：

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

【例4-3】删除数据库teaching。

```
DROP DATABASE IF EXISTS teaching
```