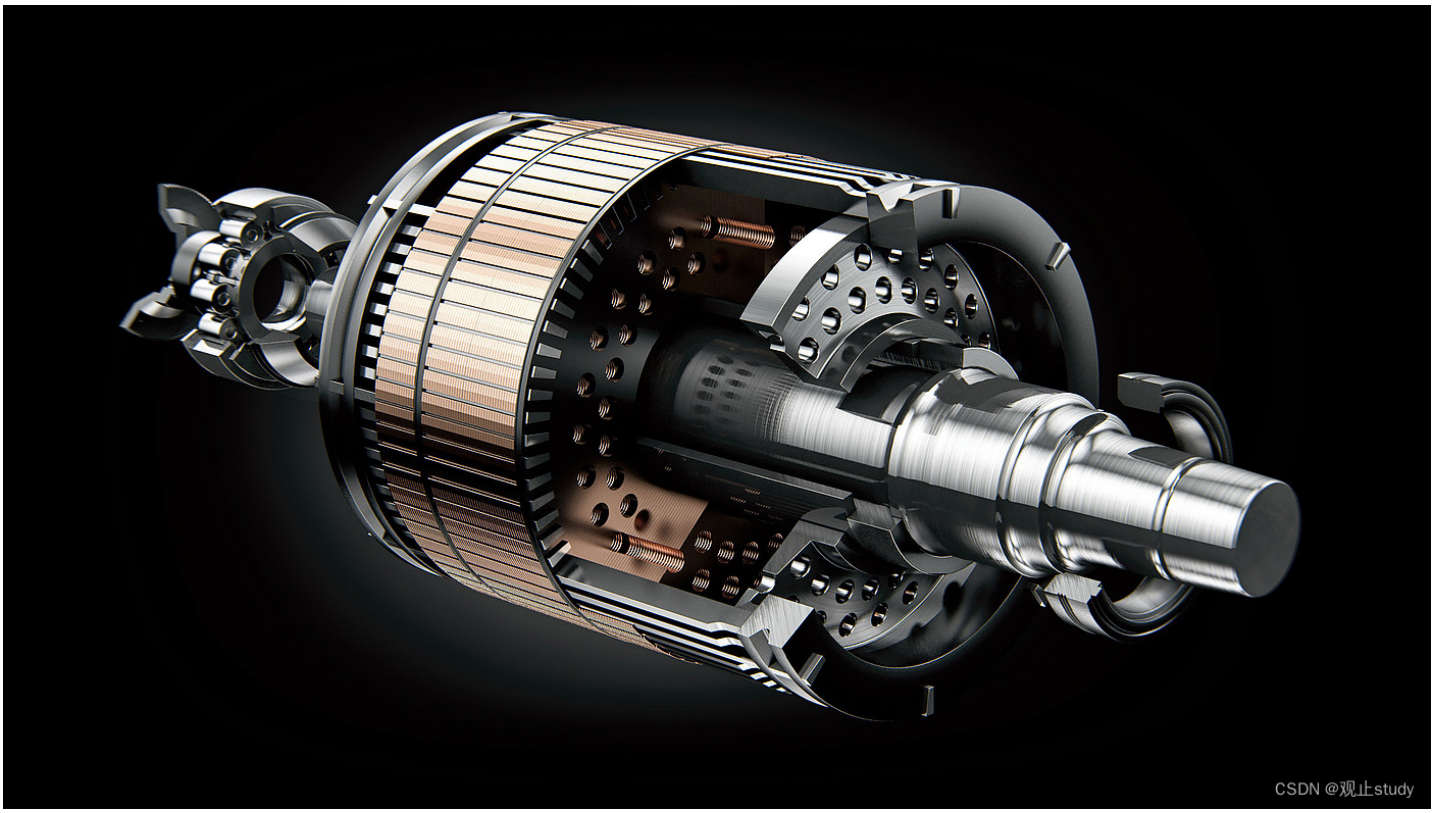
【MySQL】存储引擎（六）

> 🚙MySQL学习·第六站~
> 🚩本文已收录至专栏：MySQL通关路
> ❤️‍🩹文末附全文思维导图，感谢各位点赞收藏支持~
> ⭐学习汇总贴，超详细思维导图：【MySQL】学习汇总(完整思维导图)

一.引入

大家可能没有听说过存储引擎，但是一定听过引擎这个词，引擎就是发动机，是一个机器的核心组件。 比如，对于舰载机、直升机、火箭来说，他们都有各自的引擎，是他们最为核心的组件。



而我们在选择引擎的时候，需要在合适的场景，选择合适的存储引擎，就像在直升机上，我们不能选择舰载机的引擎 一样。而对于存储引擎，也是一样，他**是mysql数据库的核心**，我们也需要在合适的场景选择合适的存储引擎。不同的引擎具有不同的应用场景，它们之间没有好坏之分，我们只需在合适的场景选择合适的引擎。

存储引擎就是存储数据、建立索引、更新/查询数据等技术的实现方式 。存储引擎是基于表的，而不是基于库的，所以存储引擎也可被称为表类型。我们可以在创建表的时候，来指定选择的存储引擎，如果没有指定将自动选择默认的存储引擎。

二.存储引擎介绍

存储引擎有着很多种，但是我们常用的只有其中三种：`InnoDB、MyISAM、Memory`。

(1) InnoDB

`InnoDB`是一种兼顾高可靠性和高性能的通用存储引擎，在 MySQL 5.5 之后，**InnoDB 是 MySQL 默认的存储引擎**。

它有着如下特点：

- DML（增删改）操作遵循ACID模型，支持**事务**；
- **行级锁**，提高并发访问性能；
- 支持**外键**FOREIGN KEY约束，可以保证数据的完整性和正确性；

默认情况下，innoDB引擎的每张表都会对应这样一个表空间文件（`表名.ibd`），存储该表的表结构（frm-早期 、sdi-新版）、数据和索引。

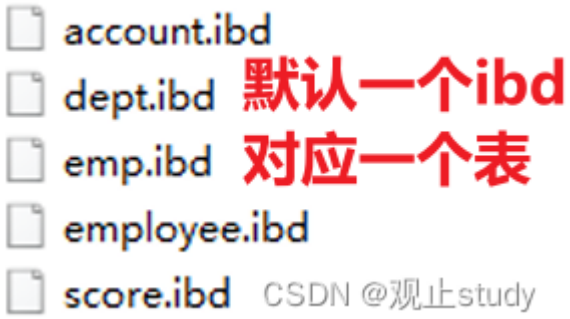
它是由参数：`innodb_file_per_table` 所控制：

```
-- ON 代表对于InnoDB引擎的表，每一张表都对应一个ibd文件
show variables  like 'innodb_file_per_table';
```

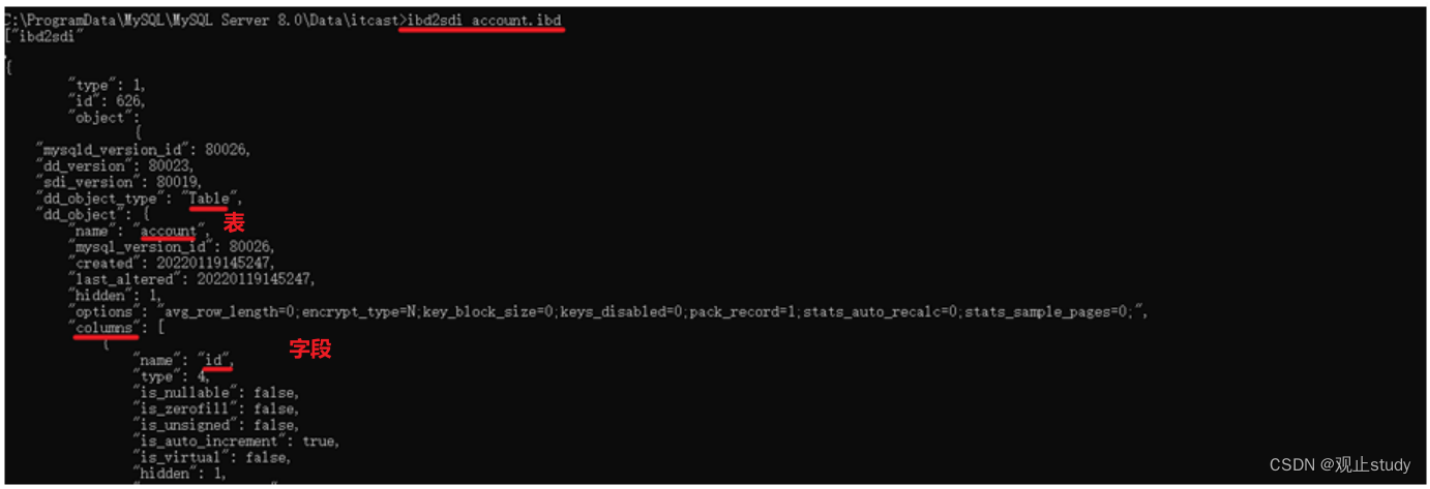
Variable_name	Value
innodb_file_per_table	ON

打开

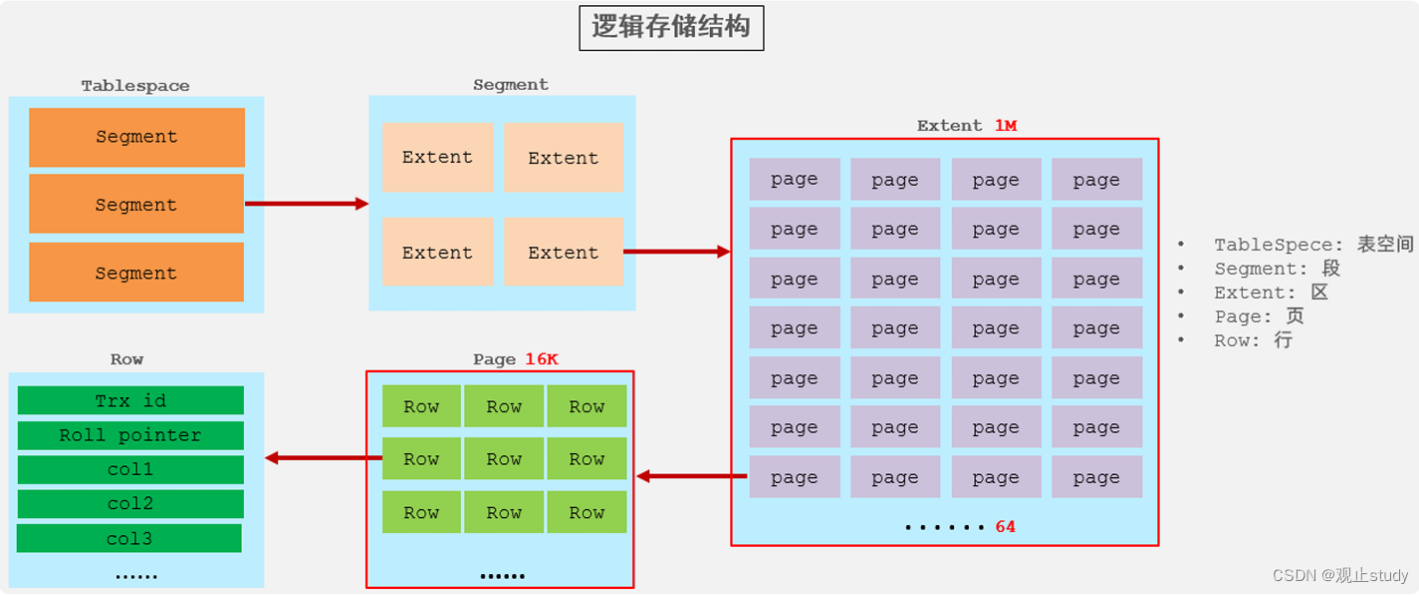
我们在文件夹 `C:\ProgramData\MySQL\MySQL Server 8.0\Data` 中查看发现确实如此：



在每个`ibd`文件中不仅存放表结构、数据，还会存放该表对应的 索引信息。 该文件是基于二进制存储的，不能直接基于记事本打开，我们可以使用mysql提供的一 个指令 `ibd2sdi` ，通过该指令就可以从ibd文件中提取sdi信息，而sdi数据字典信息中就包含该表的表结构。



它的逻辑存储结构是由表空间、段、区、页、行组成的.



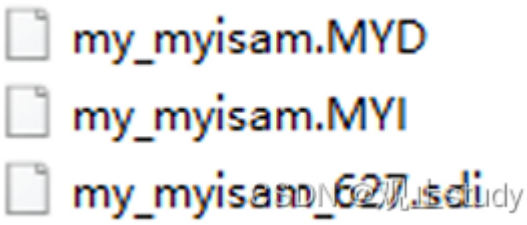
- **表空间**：InnoDB存储引擎逻辑结构的最高层，ibd文件其实就是表空间文件，在表空间中可以 包含多个Segment段。
- **段**：表空间是由各个段组成的， 常见的段有数据段、索引段、回滚段等。InnoDB中对于段的管 理，都是引擎自身完成，不需要人为对其控制，一个段中包含多个区。
- **区**：区是表空间的单元结构，每个区的大小为1M。 默认情况下， InnoDB存储引擎页大小为 16K， 即一个区中一共有64个连续的页。
- **页**：页是组成区的最小单元，页也是InnoDB 存储引擎磁盘管理的最小单元，每个页的大小默 认为 16KB。为了保证页的连续性，InnoDB 存储引擎每次从磁盘申请 4-5 个区。
- **行**：InnoDB 存储引擎是面向行的，也就是说数据是按行进行存放的，在每一行中除了定义表时 所指定的字段以外，还包含两个隐藏字段。

(2) MyISAM

`MyISAM`是MySQL早期的默认存储引擎。它不支持事务，不支持外键，不支持行锁，但是它支持表锁，而且**访问速度很快**。

它的每张表对应着三个文件：

- `表名.sdi`： 存储表结构信息
- `表名.MYD`: 存储数据
- `表名.MYI`: 存储索引



(3) Memory

`Memory`引擎的表数据时**存储在内存中**的，由于受到硬件问题、或断电问题的影响，只能将这些表作为 临时表或缓存使用，他的索引**默认为hash索引**。

它通过 `表名.sdi`文件存储表结构信息。

(4) 对比选择

特点	InnoDB	MyISAM	Memory
存储限制	64TB	有	有
事务安全	支持	×	×
锁机制	行锁	表锁	表锁
B+tree索引	支持	支持	支持
Hash索引	×	×	支持
全文索引	支持(5.6版本之后)	支持	×
空间使用	高	低	N/A
内存使用	高	低	中等
批量插入速度	低	高	高
支持外键	支持	×	×

在选择存储引擎时，应该根据应用系统的特点选择合适的存储引擎。对于复杂的应用系统，还可以根据 实际情况选择多种存储引擎进行组合。

- `InnoDB`: 是Mysql的默认存储引擎，支持事务、外键。如果应用**对事务的完整性有比较高的要求，在并发条件下要求数据的一致**性，数据操作除了插入和查询之外，还包含很多的更新、删除操作，那么InnoDB存储引擎是比较合适的选择。
- `MyISAM`： 如果应用是**以读操作和插入操作为主**，只有很少的更新和删除操作，并且对事务的完整性、并发性要求不是很高，那么选择这个存储引擎是非常合适的。例如记录日志信息的表文件。
- `MEMORY`：将所有数据保存在内存中，访问速度快，**通常用于临时表及缓存**。MEMORY的缺陷就是 对表的大小有限制，太大的表无法缓存在内存中，而且无法保障数据的安全性。

由于目前市面上有更好的`MongoDB`替代`MyISAM`存储引擎的功能以及`Redis`替代`MEMORY`存储引擎的功能，导致其使用的比较少。

三.相关操作

(1) 查询当前数据库支持的存储引擎

- 语法

```
show engines;
```

```
1 show engines;
```

信息 | 结果1 | 概况 | 状态

Engine	Support	Comment	Transactions	XA	Savepoints
MEMORY	YES	Hash based, stored in me	NO	NO	NO
MRG_MYISAM	YES	Collection of identical My	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
FEDERATED	NO	Federated MySQL storag	(Null)	(Null)	(Null)
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, ro	YES	YES	YES
BLACKHOLE	YES	/dev/null storage engine	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO

(2) 建表时指定存储引擎

- 语法

```
CREATE TABLE  表名(
字段1  字段1类型  [ COMMENT  字段1注释 ],
 ......
字段n  字段n类型  [COMMENT  字段n注释 ]
) ENGINE = 存储引擎  [ COMMENT  表注释 ];
```

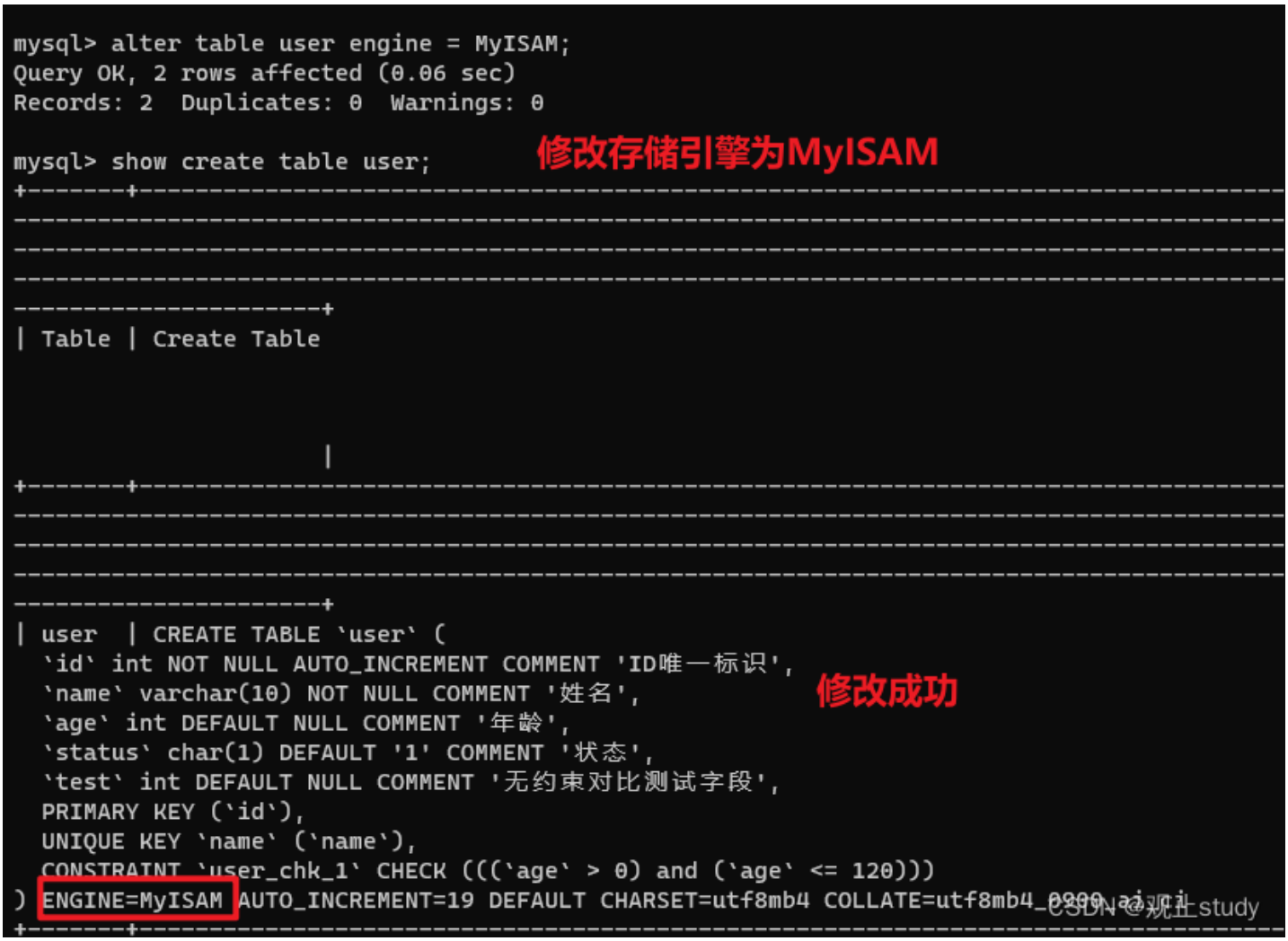
- 当我们不指定存储引擎的时候，数据库默认为InnoDB

```
-- 默认为ENGINE = InnoDB
CREATE TABLE  表名(
字段1  字段1类型  [ COMMENT  字段1注释 ],
 ......
字段n  字段n类型  [COMMENT  字段n注释 ]
)  [ COMMENT  表注释 ];
```

(3) 建表后修改存储引擎

- 语法

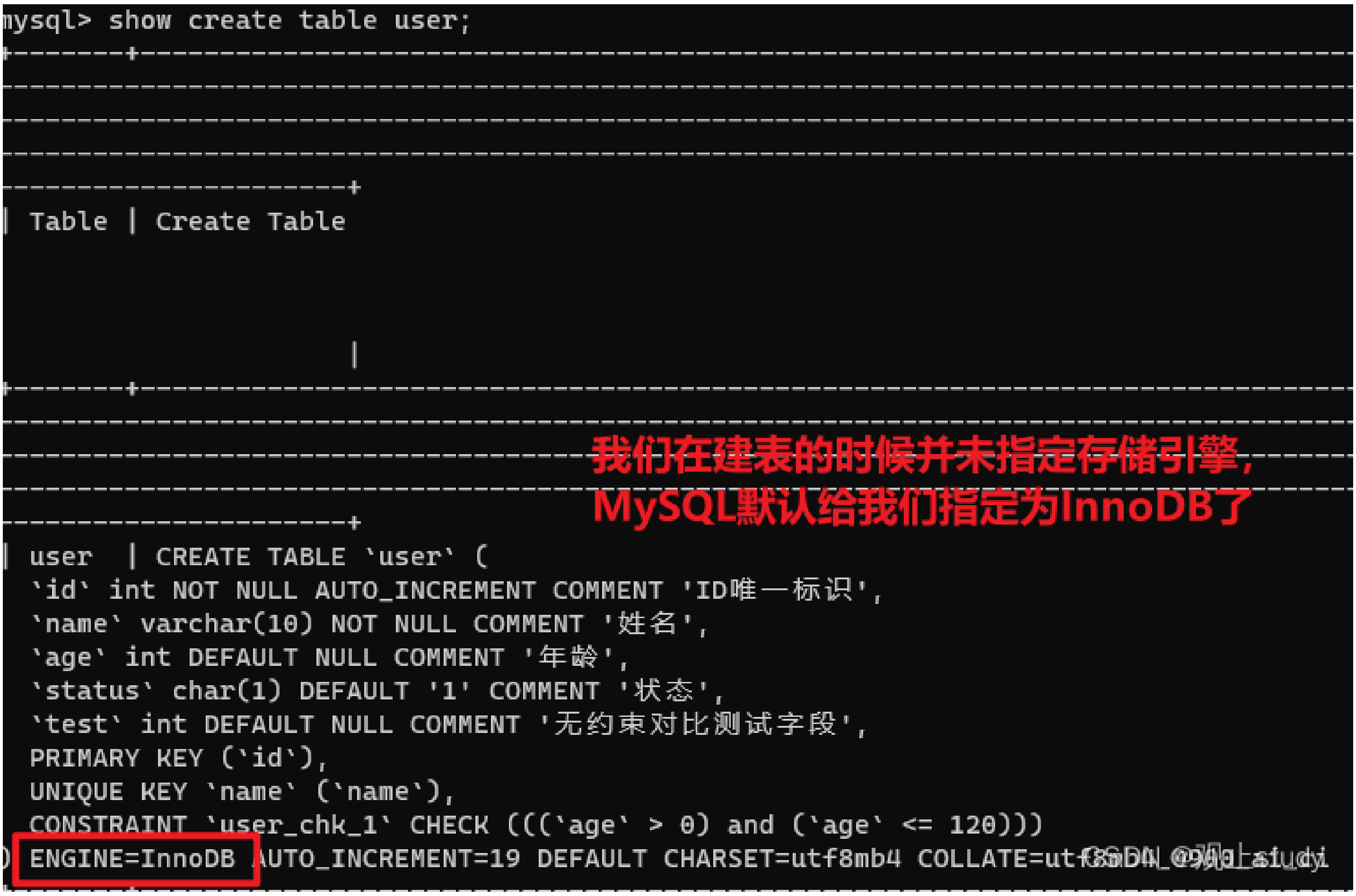
```
alter table 表名 ENGINE = 存储引擎;
```



(4) 查看表的存储引擎

- 我们可以通过查看建表语句来查看当前表所使用的存储引擎

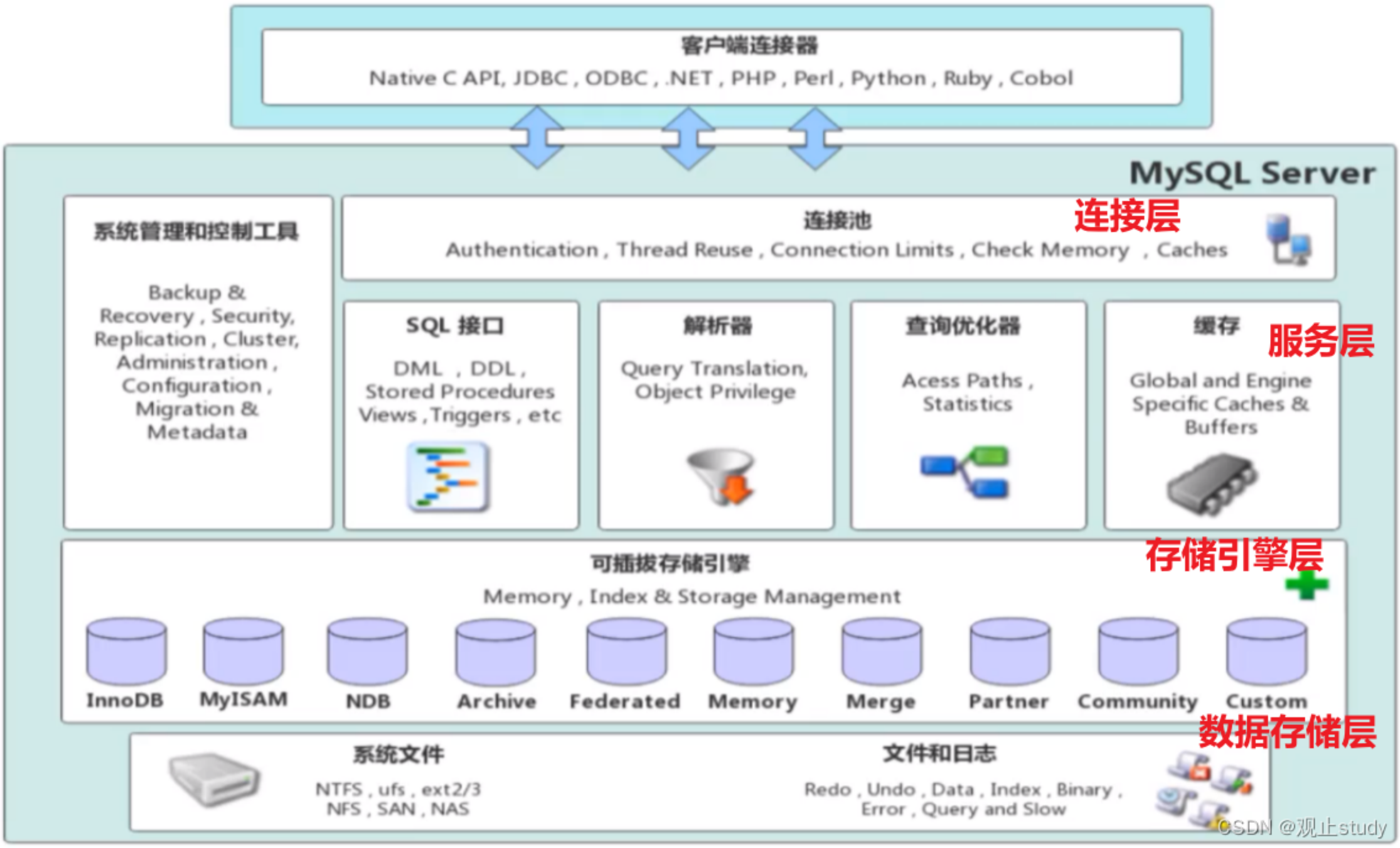
```
show create table 表名;
```



```
mysql> show create table user;
+-------+--------------------------------------------------
| Table | Create Table
                      |
+-------+--------------------------------------------------
| user  | CREATE TABLE `user` (
  `id` int NOT NULL AUTO_INCREMENT COMMENT 'ID唯一标识',
  `name` varchar(10) NOT NULL COMMENT '姓名',
  `age` int DEFAULT NULL COMMENT '年龄',
  `status` char(1) DEFAULT '1' COMMENT '状态',
  `test` int DEFAULT NULL COMMENT '无约束对比测试字段',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`),
  CONSTRAINT `user_chk_1` CHECK (((`age` > 0) and (`age` <= 120)))
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```


四. 体系结构(补充)

MySQL体系结构分为：**连接层 、 服务层、存储引擎层、数据存储层**。



- **连接层**：**负责处理一些客户端和链接服务**，包含本地sock通信和大多数基于客户端/服务端工具实现的类似于TCP/IP的通信。主要是**接收客户端的一些连接请求，完成一些类似于连接处理、授权认证及相关的安全方案**。该层引入了线程池的概念，为通过认证安全接入的客户端提供线程。同样在该层上可以实现基于SSL的安全链接。**服务器也会为安全接入的**

每个客户端验证它所具有的操作权限。例如：我们连接数据库的时候需要输入账号密码，之后便会在连接层校验我们的用户名和密码是否正确以及所拥有的权限，即能操作哪些数据库、表。

- **服务层**：**负责完成大多数的核心服务功能，如SQL接口，查询缓存，SQL的分析和优化，部分内置函数的执行**。所有跨存储引擎的功能也在这一层实现，如过程、函数等。在该层，服务器会解析查询并创建相应的内部解析树，并对其完成相应的优化如确定表的查询的顺序，是否利用索引等， 最后生成相应的执行操作。如果是select语句，服务器还会查询内部的缓存，如果缓存空间足够大， 这 样在解决大量读操作的环境中能够很好的提升系统的性能。
- **存储引擎层**：**负责了MySQL中数据的存储和提取**，服务器通过API和存储引擎进行通信。不同的存储引擎具有不同的功能，我们可以根据自己的需要，来选取合适的存储引擎。**数据库中的索引是在存储引擎层实现的，也就是说不同的存储引擎索引的结构是不一样的**。
- **数据存储层**：存储引擎层控制的是数据如何存、如何取，而**数据存储层则是负责将数据(如: redolog、undolog、数据、索引、二进制日志、错误日志、查询 日志、慢查询日志等)存储在文件系统之上**，并完成与存储引擎的交互。

和其他数据库相比，MySQL有点与众不同，它的架构可以在多种不同场景中应用并发挥良好作用。**这主要体现在存储引擎上，插件式的存储引擎架构，将查询处理和其他的系统任务以及数据的存储提取分离**。 这种架构可以根据业务的需求和实际需要选择合适的存储引擎。

五.全文概览