Part 5 存储引擎

1. 查看存储引擎

```
show engines;
# 或
show engines\G
```

2. 设置系统默认的存储引擎

- 查看默认的存储引擎：

```
show variables like '%storage_engine%';
#或
SELECT @@default_storage_engine;
```

- 修改默认的存储引擎

如果在创建表的语句中没有显式指定表的存储引擎的话，那就会默认使用`InnoDB`作为表的存储引擎。

```
SET DEFAULT_STORAGE_ENGINE=MyISAM;
```

或者修改`my.cnf`文件：

```
default-storage-engine=MyISAM
```

```
# 重启服务
systemctl restart mysqld.service
```

3. 设置表的存储引擎

存储引擎是负责对表中的数据进行提取和写入工作的，我们可以为`不同的表设置不同的存储引擎`，也就是说不同的表可以有不同的物理存储结构，不同的提取和写入方式。

3.1 创建表时指定存储引擎

```
CREATE TABLE 表名(
    建表语句;
) ENGINE = 存储引擎名称;
```

3.2 修改表的存储引擎

```
ALTER TABLE 表名 ENGINE = 存储引擎名称;
```

4. 引擎介绍

4.1 InnoDB 引擎：具备外键支持功能的事务存储引擎

- MySQL从3.23.34a开始就包含InnoDB存储引擎。`大于等于5.5之后，默认采用InnoDB引擎`。
- InnoDB是MySQL的`默认事务型引擎`，它被设计用来处理大量的短期(short-lived)事务。可以确保事务的完整提交(Commit)和回滚(Rollback)。
- 除了增加和查询外，还需要更新、删除操作，那么，应优先选择InnoDB存储引擎。
- **除非有非常特别的原因需要使用其他的存储引擎，否则应该优先考虑InnoDB引擎。**
- 数据文件结构：
  - 表名.frm 存储表结构（MySQL8.0时，合并在表名.ibd中）
  - 表名.ibd 存储数据和索引
- InnoDB是`为处理巨大数据量的最大性能设计`。
  - 在以前的版本中，字典数据以元数据文件、非事务表等来存储。现在这些元数据文件被删除了。比如：`.frm`，`.par`，`.trn`，`.isl`，`.db.opt`等都在MySQL8.0中不存在了。
- 对比MyISAM的存储引擎，`InnoDB写的处理效率差一些`，并且会占用更多的磁盘空间以保存数据和索引。
- MyISAM只缓存索引，不缓存真实数据；InnoDB不仅缓存索引还要缓存真实数据，`对内存要求较高`，而且内存大小对性能有决定性的影响。

4.2 MyISAM 引擎：主要的非事务处理存储引擎

- MyISAM提供了大量的特性，包括全文索引、压缩、空间函数(GIS)等，但MyISAM`不支持事务、行级锁、外键`，有一个毫无疑问的缺陷就是`崩溃后无法安全恢复`。
- `5.5之前默认的存储引擎`
- 优势是访问的`速度快`，对事务完整性没有要求或者以SELECT、INSERT为主的应用
- 针对数据统计有额外的常数存储。故而 count(*) 的查询效率很高
- 数据文件结构：
  - 表名.frm 存储表结构
  - 表名.MYD 存储数据 (MYData)
  - 表名.MYI 存储索引 (MYIndex)
- 应用场景：只读应用或者以读为主的业务

4.3 Archive 引擎：用于数据存档

- archive 是 归档 的意思，仅仅支持 插入 和 查询 两种功能（行被插入后不能再修改）。
- 在MySQL5.5以后 支持索引 功能。
- 拥有很好的压缩机制，使用 zlib压缩库，在记录请求的时候实时的进行压缩，经常被用来作为仓库使用。
- 创建ARCHIVE表时，存储引擎会创建名称以表名开头的文件。数据文件的扩展名为 .ARZ。
- 根据英文的测试结论来看，同样数据量下，Archive表比MyISAM表要小大约75%，比支持事务处理的InnoDB表小大约83%。
- ARCHIVE存储引擎采用了 行级锁。该ARCHIVE引擎支持 AUTO_INCREMENT 列属性。 AUTO_INCREMENT列可以具有唯一索引或非唯一索引。尝试在任何其他列上创建索引会导致错误。
- Archive表 适合日志和数据采集（档案）类应用；**适合存储大量的独立的作为历史记录的数据。**拥有 很高的插入速度，但是对查询的支持较差。

特征	支持
B树索引	不支持
备份/时间点恢复 （在服务器中实现，而不是在存储引擎中）	支持
集群数据库支持	不支持
聚集索引	不支持
压缩数据	支持
数据缓存	不支持
加密数据（加密功能在服务器中实现）	支持
外键支持	不支持
全文检索索引	不支持
地理空间数据类型支持	支持
地理空间索引支持	不支持
哈希索引	不支持
索引缓存	不支持
锁粒度	行锁
MVCC	不支持
存储限制	没有任何限制
交易	不支持

4.4 Blackhole 引擎：丢弃写操作，读操作会返回空内容

4.5 CSV 引擎：存储数据时，以逗号分隔各个数据项

- CSV引擎可以将普通的CSV文件作为MySQL的表来处理，但不支持索引。
- CSV引擎可以作为一种数据交换的机制，非常有用。
- CSV存储的数据直接可以在操作系统里，用文本编辑器，或者excel读取。
- 对于数据的快速导入、导出是有明显优势的。

创建CSV表时，服务器会创建一个纯文本数据文件，其名称以表名开头并带有.CSV扩展名。当你将数据存储到表中时，存储引擎将其以逗号分隔值格式保存到数据文件中。

4.6 Memory 引擎：置于内存的表

概述：

Memory采用的逻辑介质是内存，响应速度很快，但是当mysqld守护进程崩溃的时候数据会丢失。另外，要求存储的数据是数据长度不变的格式，比如，Blob和Text类型的数据不可用(长度不固定的)。

主要特征：

- Memory同时支持哈希（HASH）索引和B+树索引。
  - 哈希索引相等的比较快，但是对于范围的比较慢很多。
  - 默认使用哈希（HASH）索引，其速度要比使用B型树（BTREE）索引快。
  - 如果希望使用B树索引，可以在创建索引时选择使用。
- Memory表至少比MyISAM表要快一个数量级。
- MEMORY表的大小是受到限制的。表的大小主要取决于两个参数，分别是max_rows和max_heap_table_size。其中，max_rows可以在创建表时指定；max_heap_table_size的大小默认为16MB，可以按需要进行扩大。
- 数据文件与索引文件分开存储。
  - 每个基于MEMORY存储引擎的表实际对应一个磁盘文件，该文件的文件名与表名相同，类型为frm 类型，该文件中只存储表的结构，而其数据文件都是存储在内存中的。
  - 这样有利于数据的快速处理，提供整个表的处理效率。
- 缺点：其数据易丢失，生命周期短。基于这个缺陷，选择MEMORY存储引擎时需要特别小心。

使用Memory存储引擎的场景：

1. 目标数据比较小，而且非常频繁的进行访问，在内存中存放数据，如果太大的数据会造成内存溢出。可以通过参数max_heap_table_size控制Memory表的大小，限制Memory表的最大的大小。
2. 如果数据是临时的，而且必须立即可用得到，那么就可以放在内存中。
3. 存储在Memory表中的数据如果突然间丢失的话也没有太大的关系。

4.7 Federated 引擎：访问远程表

- Federated引擎是访问其他MySQL服务器的一个代理，尽管该引擎看起来提供了一种很好的跨服务器的灵活性，但也经常带来问题，因此默认是禁用的。

4.8 Merge引擎：管理多个MyISAM表构成的表集合

4.9 NDB引擎：MySQL集群专用存储引擎

5. MyISAM和InnoDB

对比项	MyISAM	InnoDB
外键	不支持	支持
事务	不支持	支持
行表锁	表锁，即使操作一条记录也会锁住整个表，不适合高并发的操作	行锁，操作时只锁某一行，不对其它行有影响，适合高并发的操作
缓存	只缓存索引，不缓存真实数据	不仅缓存索引还要缓存真实数据，对内存要求较高，而且内存大小对性能有决定性的影响
自带系统表使用	Y	N
关注点	性能：节省资源、消耗少、简单业务	事务：并发写、事务、更大资源
默认安装	Y	Y
默认使用	N	Y