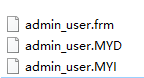
**第十一节知识点 :   
 mysql体系结构**

**第十二节课知识点:**

**MyISAM：**

1. **Mysql5.5之前是默认的**
2. **也是mysql系统表和临时表使用的引擎**
3. **这里说的临时表是在排序分组等操作中，当数量超过以丁的大小之后,由查询优化器建立的临时表；而不是用create table 建立的临时表**
4. **它将表结构存放在frm文件中，将数据放在myd文件中，将索引存放在myi文件中**
5. **特性:**
   1. **并发行与锁级别（共享锁？）**

**共享锁就是一个锁多个进程可以用，但是不能加其他的锁，（进程只能读）**

**排他锁就是一个锁只能一个进程用，但是不能加其他的锁；（进程可以读或写）**

* + 1. **锁是表锁，说明我们修改表数据的时候会对整个表加X（排他）锁，**
    2. **对读写混合的并发性不太好，但是只是读的话还不错；因为读的时候加的是共享锁**
    3. **表损坏修复**
       1. **查看表是否损坏 check table user**
       2. **修复表 repair table user**
       3. **命令行工具修复myisam表**

**使用Myisamchk时 需要停止mysql服务**

* 1. **Myisam支持的索引类型**
     + 1. **全文索引**
  2. **Myisam 支持数据压缩**

**Myisampack –b –f myisam.myi 命令**

* 1. **使用场景:**
     1. **非事务型应用 （如不要求数据的完整性）**
     2. **对于只读的应用（因为可以压缩表）**
     3. **空间类应用 （）**

**第十三节知识点  
 InnoDB  
 在mysql5.5 或 5.8 后 默认引擎为 InnoDB  
 1. 支持事务。**

**2. innodb 使用表空间进行数据存储**

**Innodb\_file\_per\_table 参数决定innodb使用什么形式的数据存储  
查看参数的命令:**

**Show variables like ‘innodb\_file\_per\_table’;**

**修改参数的命令:**

**Set global innodb\_file\_per\_table=off;**

**为 ON： 使用独立表空间 : 表名.ibd**

**每个表都有自己独立的表空间**

**为 off : 系统表空间 : ibdataX**

**系统表空间是多个表在表空间混合储存**

**比较:**

1. **系统表空间无法简单的收缩文件的大小。**

**系统表空间就算把表删除了系统表的大小也是不会改变的。只能通过瘦身的方法进行文件瘦身**

1. **系统表空间因为都放在一个文件中，多个表的读写操作都在这一个文件中进行就会造成IO瓶颈**
2. **独立表空间可以收缩文件大小**
3. **独立表空间可以同时向多个文件进行读写**

**将系统表孔见中的表转入到独立表空间步骤**

1. **使用mysqldump导出所有数据库表数据**
2. **停止mysql服务，修改参数，并删除INNOdb相关文件。**
3. **重启mysql服务重建innodb系统表空间**
4. **重新导入数据**

**第十四节知识点  
 Innodb的系统表空间中存了哪些重要数据?**

1. **Innodb 数据字典信息（如，表，列，索引）**
2. **Undo 回滚段**

**Innodb 存储引擎的特性：**

1. **Innodb是一种事务性存储引擎**
2. **完全支持事务的ACID特性**

**Innodb是如何支持事务的?**

1. **Redo Log**
2. **Undo log**

**Innodb 支持行级锁**

**行级锁可以最大成都的支持并发**

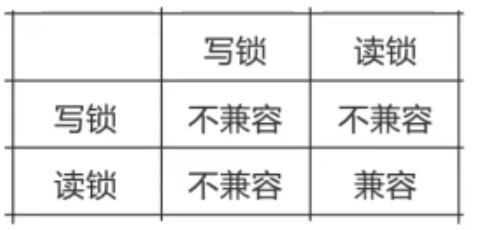
**行级锁是有存储引擎层实现的；**

**什么是锁:**

1. **锁对主要作用是管理共享资源的并发访问**
2. **锁用于实现事务的隔离性**

**锁的类型**

1. **共享锁 （读锁）**
2. **独占锁 （写锁）**



**锁的粒度:**

1. **表级锁**
2. **行级锁**
3. **列级锁**

**第十五节知识点**

**阻塞和死锁:**

1. **阻塞**
   1. **一个锁在等待另一个锁的释放**
   2. **阻塞是保证事务的并发和正常运行**
   3. **而阻塞意味着可能有慢查询**
2. **死锁**

**Innodb状态检查:**

**Show engine innodb status;**

**使用场景:**

**Innodb适合于大多数OLTP应用**

**在mysql5.7开始innodb也支持全文索引和空间函数**

**第十六节知识点:**

**CSV 存储引擎**

1. **数据以文件方式存储在问文件中；**
2. **.csv 文件存储表内容**
   1. **(以逗号分割，双引号包含数据，表字段不能为null)**
3. **.csm 文件存储表的元数据如表状态和数据量**
4. **.frm 文件存储表结构信息**
5. **不支持索引**
6. **可以对数据文件直接编辑**

**使用场景:**

1. **适合作为数据交换的中间表**





1. **Excel 转 CSV**

[**https://jingyan.baidu.com/article/5bbb5a1be25d4b13eba1791e.html**](https://jingyan.baidu.com/article/5bbb5a1be25d4b13eba1791e.html)

**第十七节知识点:**

**Archive引擎 （如果只有 写操作可以用这个）**

1. **以zlib对表数据进行压缩，磁盘I/O更少**
2. **数据存储在ARZ为后缀的文件中**
3. **只支持insert和select操作（支持行级所和缓冲区，可以实现高并发的插入）**
4. **只允许在自增ID列上加索引**
5. **Archive表比MyISAM表要小大约75%，比支持事务处理的InnoDB表小大约83%。当数据量非常大的时候Archive的插入性能表现会较MyISAM为佳。**
6. **Archive表的性能是否可能超过MyISAM？答案是肯定的。根据MySQL工程师的资料，当表内的数据达到1.5GB这个量级，CPU又比较快的时候，Archive表的执行性能就会超越MyISAM表。因为这个时候，CPU会取代I/O子系统成为性能瓶颈。别忘了Archive表比其他任何类型的表执行的物理I/O操作都要少。**

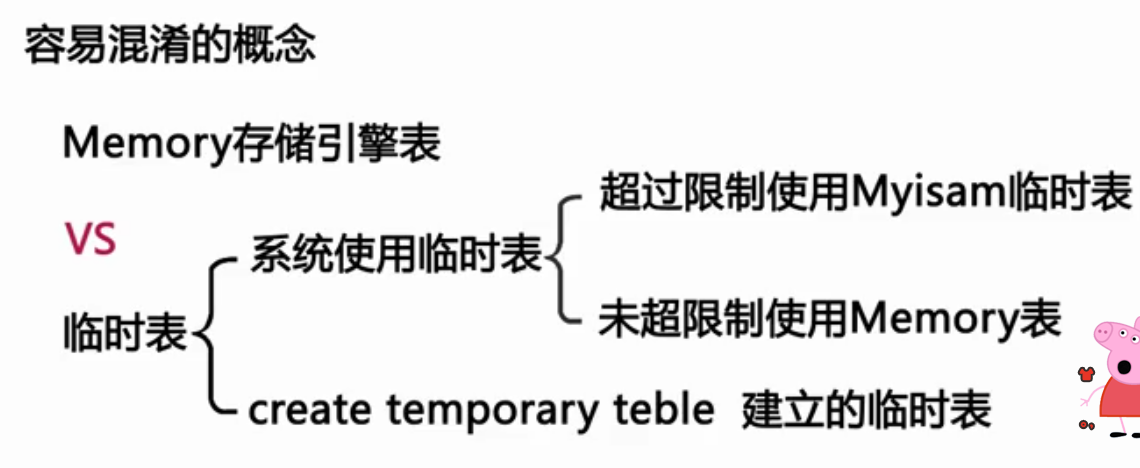
**使用场景:**

1. **日志;**

**第十八节知识点:**

**Memory引擎**

1. **数据都是保存在内存中（容易丢失）**
2. **表结构存储在磁盘上 （表结构不会丢失）**
3. **支持 hash索引 和 btree 索引**
4. **Hash 用来做等值查询 Btree 用来范围查找**
5. **所有的字段都是固定的 不管你定的是varchar 直接转 char**
6. **Memory 存储引擎用表级锁**
7. **最大大小由 max\_heap\_table\_size参数决定（当然之前创建过的表不会在改变）**



**使用场景（memory 数据是易丢失，所以要求数据能够再生）**

1. **等值查找，如 ；邮编，地区**
2. **用于保存数据分析中产生的中间表**
3. **用于缓存周期性聚合数据的结果表**

**索引的添加:**

1. **Create** [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引的名

ON 表名（字段名[长度]）

https://www.2cto.com/database/201801/713186.html

1. **Alter table 表名 add 索引类型 [索引的名] （表名（长度））**

**索引的删除:**

1. **Drop index 索引的名 ON 表名**
2. **Alter table 表名 index 索引的名**

**查看索引信息**

1. **Show index from 表名；**

**查看表状态命令:**

1. **show table status like ‘user’\G;**

**查看创建表的命令的命令:**

1. **show create table user \G;**

**查看表索引  
 5. Show index from 表名\G**

**新手学习索引网站**

**https://blog.csdn.net/u012954706/article/details/81241049**

**第十九节知识点:**

**Federated引擎**

**特点:**

1. **提供了访问远程mysql服务器上表的方法**
2. **本地不存储数据，数据全部在远程服务器上**
3. **本地需要保存表结构和远程服务器的连接信息**

**怎么使用:**

1. **首先 show engines ;查看一下是否支持federated;如果不支持则在mysql配置文件中加入 federated =1；**
2. **Mysql：//user\_name[:password]@**

**host\_name[:port\_name]/db\_name/table\_name**

**使用案例:**

**在a库中创建一个aa表；**

**create table aa(**

**id int auto\_increment not null,**

**name varchar(40) not null,**

**primary key(id));**

**添加几条数据**

**insert into aa(name) values ('aa'),('bb'),('cc');**

**给一个用户授权:**

**Grant select,update,insert,delete on a.aa to fred\_link@'127.0.0.1' identified by '123456';**

**在b库中创建dd表 并连接a库中的aa表**

**create table dd(**

**id int not null auto\_increment,**

**name varchar(20) not null,**

**primary key(id)**

**)engine=federated connection='mysql://fred\_link:123456@127.0.0.1:3306/a/aa';**

**然后dd 表就能使用aa表了；**

**使用场景：**

**偶尔的统计分析 和 手动查询；**

**第二十节知识点：**

**如何选择存储引擎：**

**大部分情况下使用innodb引擎;当innodb做不到的一些情况下可以参考别的引擎；**

**选择引擎参考条件:**

1. **判断应用是否事务; innodb**
2. **如果不进行事务只是insert select 可以用myisam**
3. **备份; innodb 有免费方案；**

**Mysqldump是逻辑备份，为保证数据的一致性它要加锁，所以他不是一个在线热备的方案；**

1. **崩溃恢复: innodb会更好**
2. **存储引擎的特有特性**

**减少引擎混合;**

**因为有时候我们两表联查时候一个是innodb一个是myisam，当数据回滚也只会回滚innodb的表所以会造成数据的混乱；混合引擎也无法是实现热备；**