mysql 默认的数据库myisam

三种使用指定数据库引擎方法

配置文件my.ini

[mysqld]

default-storage-engine=InnoDB

建表时指定

CREATE TABLE `rice\_ball\_star` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT COMMENT '明星id',

`img\_url` varchar(256) NOT NULL COMMENT '明星头像地址',

`name` varchar(56) NOT NULL COMMENT '明星名字',

`name\_pinyin` varchar(20) NOT NULL COMMENT '明星名字（拼音）',

`type` tinyint(4) NOT NULL COMMENT '明星类型：0：个人 1：cp',

`deleted` tinyint(2) NOT NULL DEFAULT '0' COMMENT '是否删除 0.否 1.是',

`gmt\_create` timestamp NOT NULL COMMENT '创建时间',

`gmt\_modified` timestamp NOT NULL ON UPDATE CURRENT\_TIMESTAMP COMMENT '修改时间',

PRIMARY KEY (`id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=480 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC COMMENT='明星表'

更改数据表引擎

alter table tablename engine=innodb

查看数据库引擎

SHOW ENGINES;

InnoDB是一个事务型的存储引擎，有行级锁定和外键约束。

Innodb引擎提供了对数据库ACID事务的支持，并且实现了SQL标准的四种隔离级别。该引擎还提供了行级锁和外键约束，它的设计目标是处理大容量数据库系统，它本身其实就是基于MySQL后台的完整数据库系统，

MySQL运行时Innodb会在内存中建立缓冲池，用于缓冲数据和索引。但是该引擎不支持FULLTEXT类型的索引，而且它没有保存表的行数，当SELECT COUNT(\*) FROM TABLE时需要扫描全表。

当需要使用数据库事务时，该引擎当然是首选。由于锁的粒度更小，写操作不会锁定全表，所以在并发较高时，使用Innodb引擎会提升效率。

但是使用行级锁也不是绝对的，如果在执行一个SQL语句时MySQL不能确定要扫描的范围，InnoDB表同样会锁全表。

ACID

A 事务的原子性(Atomicity)：指一个事务要么全部执行,要么不执行.也就是说一个事务不可能只执行了一半就停止了.比如你从取款机取钱,这个事务可以分成两个步骤:1划卡,2出钱.不可能划了卡,而钱却没出来.这两步必须同时完成.要么就不完成.

C 事务的一致性(Consistency)：指事务的运行并不改变数据库中数据的一致性.例如,完整性约束了a+b=10,一个事务改变了a,那么b也应该随之改变.

I 独立性(Isolation）:事务的独立性也有称作隔离性,是指两个以上的事务不会出现交错执行的状态.因为这样可能会导致数据不一致.

D 持久性(Durability）:事务的持久性是指事务执行成功以后,该事务所对数据库所作的更改便是持久的保存在数据库之中，不会无缘无故的回滚.

myisam 索引和数据分开，非聚集索引， 并且索引有压缩，内存使用率提高了不少，能加载更多索引

innodb 索引和数据在一起，索引也没有使用压缩。

MySQL能够允许你在表这一层应用数据库引擎，所以你可以只对需要事务处理的表格来进行性能优化，而把不需要事务处理的表格交给更加轻便的MyISAM引擎。对于 MySQL而言，灵活性才是关键。

聚簇索引

innodb b+树 索引和数据存放在一起， 叶子结点数据域存放的是数据本身而不是数据地址。

innodb数据文件就是 主索引

辅助索引数据域存储的是数据主键的值。

搜索过程：

将主键组织到一棵B+树中，而行数据就储存在叶子节点上，若使用”where id = 13”这样的条件查找主键，则按照B+树的检索算法即可查找到对应的叶节点，之后获得行数据。

若对Name列进行条件搜索，则需要两个步骤：第一步在辅助索引B+树中检索Name，到达其叶子节点获取对应的主键。第二步使用主键在主索引B+树种再执行一次B+树检索操作，最终到达叶子节点即可获取整行数据。

辅助索引应该不是使用的b+树

innodb的数据文件是结合主键形成的，所以innodb表必须要有主键。

如果没有显式指定，则MySQL系统会自动选择一个可以唯一标识数据记录的列作为主键，如果不存在这种列，则MySQL自动为InnoDB表生成一个隐含字段作为主键，这个字段长度为6个字节，类型为长整形。

非聚簇索引 myisam b+树 叶子结点的key包含所有的索引，叶子结点的数据域存储的是真正数据的地址

myisam中索引检索的算法为首先按照B+Tree搜索算法搜索索引，如果指定的Key存在，则取出其data域的值作为地址读取相应数据记录。

使用自增主键，b+树节点会被顺序的填满，而不会频繁调整b+树，可以有效的提升插入效率。

水平分表

使用myisam引擎和mrg\_myisam引擎

统一id问题处理

数据分配逻辑：可以按业务分，也可以随机均分

//用户表一

CREATE TABLE IF NOT EXISTS `user1` (

`id` int(11) NOT NULL ,

`name` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

//用户表二

CREATE TABLE IF NOT EXISTS `user2` (

`id` int(11) NOT NULL ,

`name` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

//分别插入两条测试数据先

INSERT INTO `user1` (`name`) VALUES('辅助');

INSERT INTO `user2` (`name`) VALUES('JackFrost');

//总表

CREATE TABLE IF NOT EXISTS `alluser` (

`id` int(11) NOT NULL ,

`name` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=MRG\_MYISAM

DEFAULT CHARSET=utf8

UNION=(user1,user2) ;

总表只有只读权限

3. 但是我们又想在总表去插入，怎么办呢？？

允许通过总表插入数据，数据存储在MRG文件列出的第一个分表之中。例如，执行以下SQL语句，将总表的INSERT\_METHOD修改为FIRST，然后通过总表插入一条数据：

//就是插入总表的时候，其实也是插入到第一个分表。

ALTER TABLE `test\_engine`.`alluser` INSERT\_METHOD = FIRST;

INSERT\_METHOD选项只会影响通过总表插入（INSERT）数据的行为，通过总表对数据进行删除（DELETE）、查询（SELECT）、修改（UPDATE）、清空（TRUNCATE）都不会受影响。

INSERT INTO `alluser` (id,`name`) VALUES(2,'插入到第一个分表');

而，允许通过总表插入数据，数据存储在MRG文件列出的最后一个分表之中。例如，执行以下SQL语句，将总表的INSERT\_METHOD修改为LAST，然后通过总表插入一条数据：

//就是插入总表的时候，其实也是插入到最后一个分表。

ALTER TABLE `test\_engine`.`alluser` INSERT\_METHOD = LAST;

基于MRG\_MYISAM存储引擎实现的分表机制，比较适用于插入和查询频率较高的场景。由于MyISAM具有表级别的锁机制，所以不适用于更新频率较高的场景。

其他的删除（DELETE）、查询（SELECT）、修改（UPDATE）、清空（TRUNCATE）等操作都可以通过总表alluser完成。

1）MYD文件是MyISAM表的数据文件；

2）MYI文件是MyISAM表的索引文件；

3）frm文件用于存储MyISAM表的表结构。

（3）MRG\_MYISAM分表的优点：

（一）适用于存储日志数据。例如，可以将不同月份的数据存入不同的表，然后使用一些工具压缩数据，最后通过一张MRG\_MYISAM表来查询这些数据。

（二）可以获得更快的速度。可以根据某种指标，将一张只读的大表分割成若干张小表，然后将这些小表分别放在不同的磁盘上存储。当需要读取数据时，MERGE表可以将这些小表的数据组织起来，就好像使用先前的大表一样，但是速度会快很多。

（三）可以提高搜索效率。可以根据某种指标将一张只读的大数据表分割为若干个小表，然后根据不同的查询维度，可以得到若干种小表的组合，然后再为这些组合分别创建不同的MERGE表。例如，有一张只读的大数据表T，分割为T1、T2、T3、T4，共4张小表，有两种查询维度A和B，A可以得到小表组合T1、T2和T3，B可以得到小表组合T2、T3和T4，分别为A和B创建两个MERGE表，也就是M1和M2，这两个MERGE表分别关联的小表是存在交叠的。

（四）可以更加有效的修复表。修复单个的小表要比修复大数据表更加容易。

（五）多个子表映射至一个总表的速度极快。因为MERGE表本身不会存储和维护任何索引，索引都是由各个关联的子表存储和维护的，所以创建和重新映射MERGE表的速度非常快。

（六）不受操作系统的文件大小限制。单个表会受到文件大小的限制，但是拆分成多个表，则可以无限扩容。

（七）MERGE表还可以用来给单个表创建别名，并且几乎不会影响性能。

（4）使用MRG\_MyISAM分表的必须思考问题：（针对总表）

（一）总表（MERGE表）必须使用MRG\_MyISAM存储引擎，子表必须使用MyISAM存储引擎，不可避免会受到MyISAM存储引擎的限制。（比如不支持事务）

（二）MERGE表不能使用某些MyISAM特性。例如，虽然可以为子表创建全文索引，但是却不能使用全文索引，只能通过MERGE表查询数据。

（三）若使用ALTER TABLE语句修改总表的存储引擎，那么会立即丢失总表和子表的映射关系，并且会将所有子表的数据拷贝至修改后的新表。

（四）总表和子表的主键都不能使用自动增长（auto increment）。

（五）子表之间不能保证唯一键约束，只能保证单个子表内部的唯一性约束。也就是说，直接查总表全部，id可能会重复。

（六）由于不能保证唯一键约束，导致REPLACE语句的行为会不可预期，INSERT … ON DUPLICATE KEY UPDATE语句也有类似问题。因此，只能使用路由策略，对子表使用这些语句，而不能对总表使用。

（七）当正在使用总表时，不能对任何子表执行ANALYZE TABLE、REPAIR TABLE、OPTIMIZE TABLE、ALTER TABLE、DROP TABLE、DELETE或TRUNCATE TABLE语句，否则会导致不可预期的结果。

（八）总表和子表的表结构必须完全一致。

（九）总表可以映射的所有子表的总行数上限为 2的64次方 行。

（十）不支持INSERT DELAYED语句。

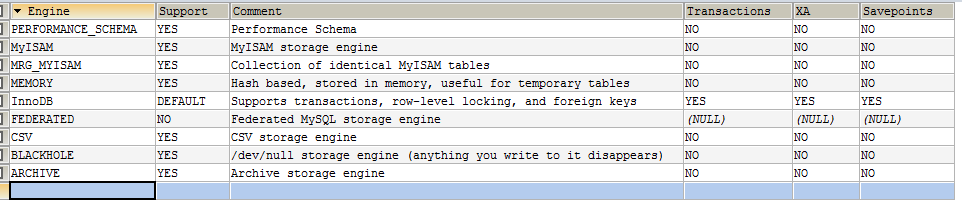
memory引擎只适合作为临时中间结果数据的存放，而且数据量不能太大

参考文档：

https://www.cnblogs.com/sunsky303/p/8274586.html

一些性能测试：

#show engines



delimiter $$

drop procedure if exists ptestmyisam$$

create procedure ptestmyisam()

begin

declare pid int ;

set pid = 1000000;

while pid>0

do

insert into testmyisam(name) values(concat("fuzhu", pid));

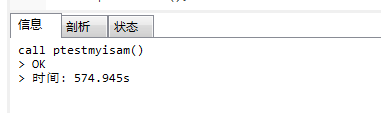
set pid = pid-1;

end while;

end $$

//使用存储过程：

call ptestmyisam();



//创建存储过程（尽量把Innodb的数量级压低，不然，，卡在那里半天也不奇怪）

delimiter $$

drop procedure if exists ptestInndb;

create procedure ptestInndb()

begin

declare pid int ;

set pid = 1000000;

while pid>0

do

insert into testinnodb(name) values(concat("fuzhu", pid));

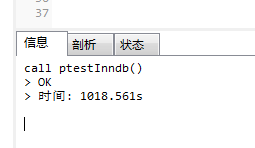
set pid = pid-1;

end while;

end $$

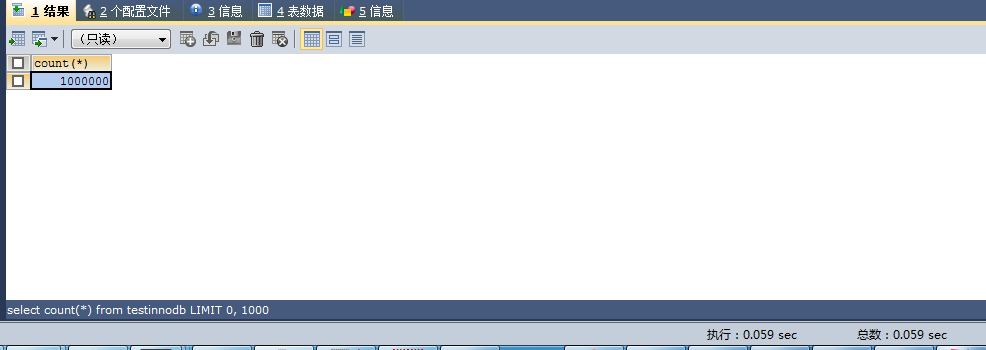
//使用存储过程：

call ptestInndb();



查询总数

SELECT COUNT(\*) FROM testinnodb;



SELECT COUNT(\*) FROM testmyisam;

