MySQL的系统，引擎，锁机制

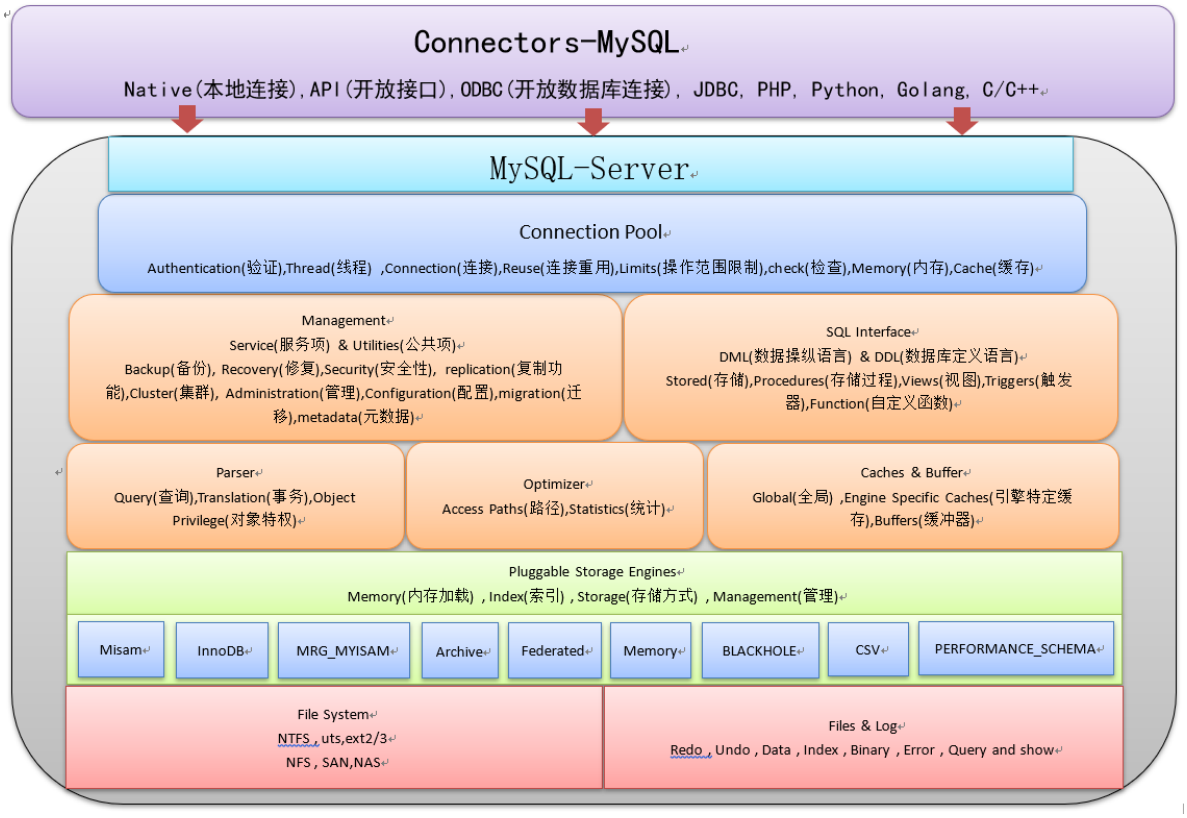
一、MySQL逻辑架构

1、总体分层

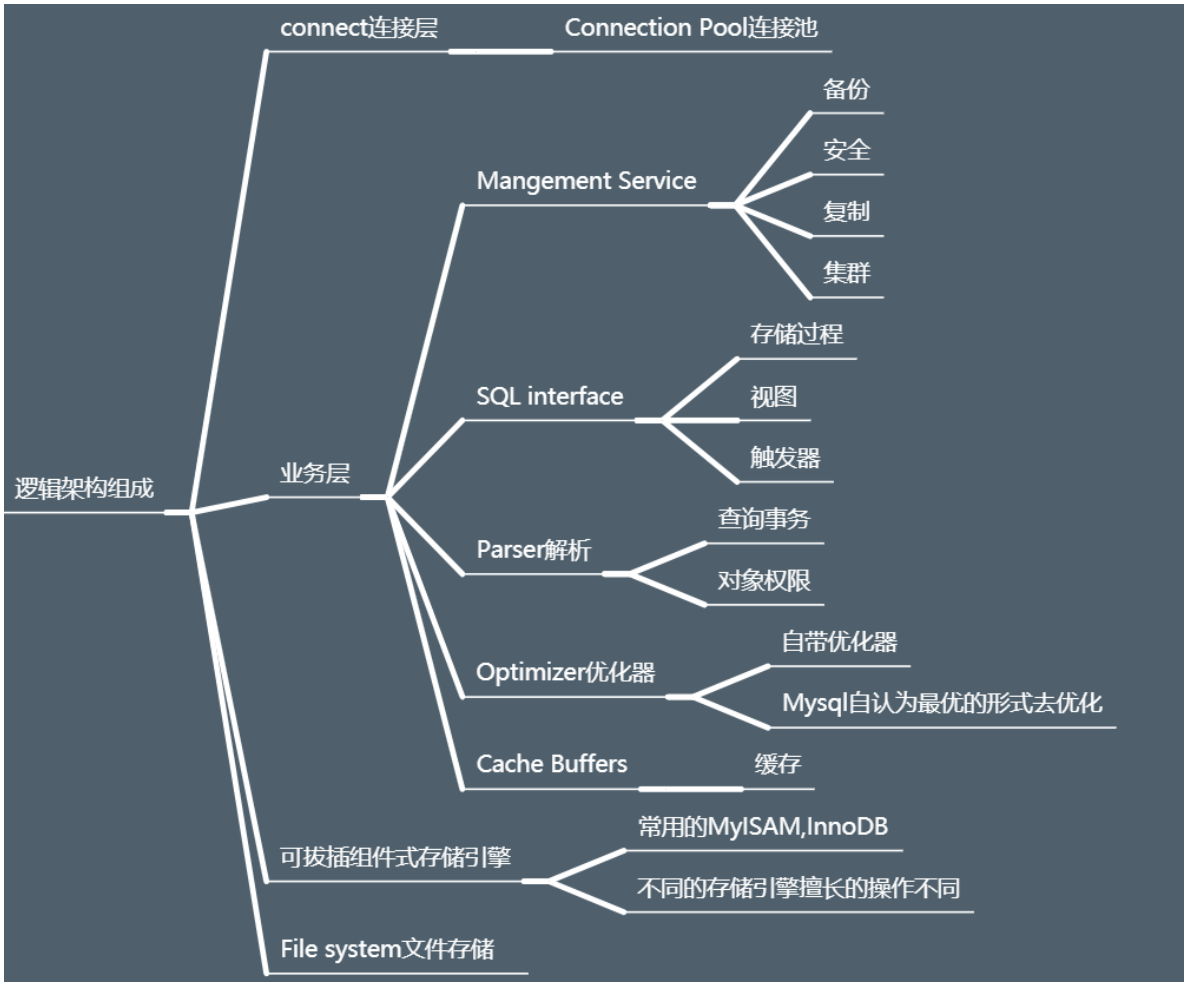
- 连接层
  - 与客户端建立连接的服务
  - 完成一些连接处理，授权认证及相关的安全方案
  - 在该层上引入了连接池的概念
- 服务层
  - 提供核心的服务功能
  - 复制sql接口，完成缓存的查询，sql的分析和优化部分及内置函数的执行
  - 所有跨存储引擎的功能都在这一层实现
  - 解析查询并创建相应的解析权，并完成优化，生成相应的操作
  - 合理分配内部的缓存，解决大量读操作优化问题
- 引擎层
  - 真正负责MYSQL中数据的存储和提取，通过API与存储引擎通信
  - 不同的存储引擎功能不同

- 存储层
  - 主要将数据存储的文件运行在计算机文件系统中，完成于引擎的交互

2、逻辑架构图



3、逻辑架构分层



4、逻辑分层的好处

- 1.功能分布明确，哪里出问题找哪里
- 2.插件式的存储引擎，将查询处理和其它任务系统任务已经数据提取相分离
- 3.可以根据业务的需求和业务的需要选择合适的存储引擎

二、存储引擎

1、存储引擎的概念

- MySQL中的数据用各种不同的技术存储在文件(或者内存)中。

- 这些技术中的每一种技术都使用不同的存储机制、索引技巧、锁定水平并且最终提供广泛的不同的功能和能力。
- 通过选择不同的技术，你能够获得额外的速度或者功能，从而改善你的应用的整体功能。
- 这些不同的技术以及配套的相关功能在 MySQL中被称作存储引擎

查看当前使用的引擎

```
show engines;          #查看当前版本中默认支持的数据库引擎

show variables like '%storage_engine';
#查看当前数据库使用的引擎
```

1. **InnoDB**

   事务型数据库的首选引擎，支持事务安全表（ACID），支持行锁定和外键，InnoDB是默认的MySQL引擎。

   InnoDB主要特性有：

   1. InnoDB 给 MySQL 提供了具有提交、回滚、崩溃恢复能力的事务安全存储引擎。
   2. InnoDB 是为处理巨大数据量的最大性能设计。它的 CPU 效率比其他基于磁盘的关系型数据库引擎高。
   3. InnoDB 存储引擎自带缓冲池，可以将数据和索引缓存在内存中。

4. InnoDB 支持外键完整性约束。
5. InnoDB 被用在众多需要高性能的大型数据库站点上使用
6. InnoDB 支持行级锁

2. **MyISAM**

   MyISAM 基于 ISAM 存储引擎，并对其进行扩展。它是在Web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM 拥有较高的插入、查询速度，但不支持事务。

   MyISAM主要特性有：

   1. 大文件支持更好
   2. 当删除、更新、插入混用时，产生更少碎片。
   3. 每个 MyISAM 表最大索引数是64，这可以通过重新编译来改变。每个索引最大的列数是16
   4. 最大的键长度是1000字节。
   5. BLOB和TEXT列可以被索引
   6. NULL 被允许在索引的列中，这个值占每个键的0~1个字节
   7. 所有数字键值以高字节优先被存储以允许一个更高的索引压缩
   8. MyISAM 类型表的 AUTO_INCREMENT 列更新比 InnoDB 类型的 AUTO_INCREMENT 更快
   9. 可以把数据文件和索引文件放在不同目录
   10. 每个字符列可以有不同的字符集
   11. 有 VARCHAR 的表可以固定或动态记录长度
   12. VARCHAR 和 CHAR 列可以多达 64KB

13. 只支持表锁
14. 支持FULLTEXT KEY(全文索引)

3. **MEMORY**

   MEMORY 存储引擎将表中的数据存储到内存中，为查询和引用其他表数据提供快速访问。

4. **BLACKHOLE**

   黑洞引擎一般是作为伪主机，用来缓解master的压力的一总主从架构的中间机器使用，

   只产生日志，不保存数据。

2、存储引擎的选择

一般来说，对插入和并发性能要求较高的，或者需要外键，或者需要事务支持的情况下，需要选择 InnoDB，

插入较少，查询较多的场景，优先考虑 MyISAM。

3、指定引擎

一般在建表时添加

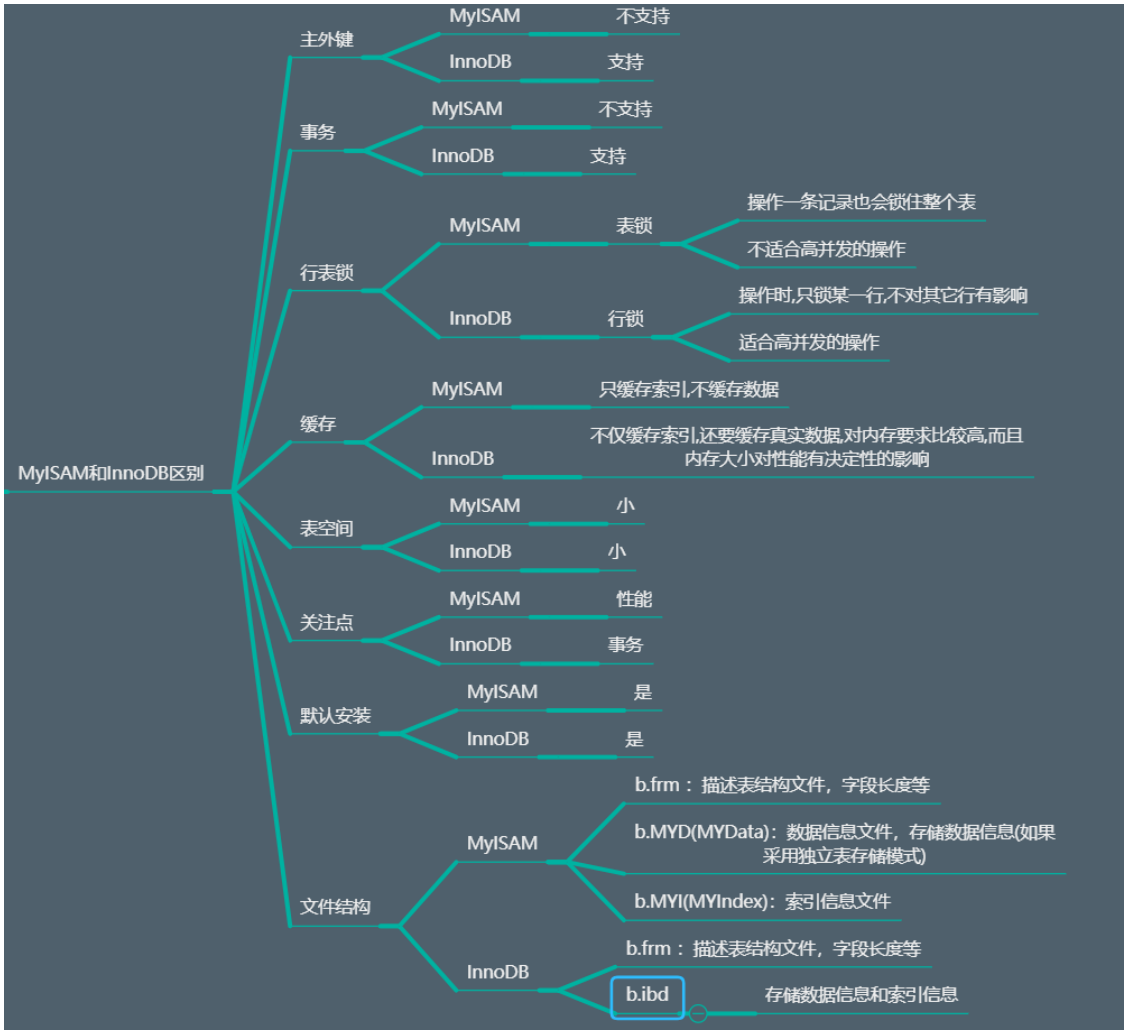
```
create table abc (
    name char(10)
) engine=MyISAM charset=utf8;

create table xyz (
    name char(10)
) engine=InnoDB charset=utf8;

alter table table_name
engine=innodb|myisam;
```

4、InnoDB 和 MyISAM 在文件方面的区别

1. InnoDB 将一张表存储为两个文件
    - demo.frm -> 存储表的结构和索引
    - demo.ibd -> 存储数据，ibd 存储是有限的, 存储不足自动创建 ibd1, ibd2
2. MyISAM 将一张表存储为三个文件
    - demo.frm -> 存储表的结构
    - demo.myd -> 存储数据
    - demo.myi -> 存储表的索引



三、锁机制

锁是计算机协调多个进程或线程并发访问某一资源的机制。

锁保证数据并发访问的一致性、有效性；

锁冲突也是影响数据库并发访问性能的一个重要因素。

锁是Mysql在服务器层和存储引擎层的的并发控制。

名词分类

- 行级锁

    - 行级锁是Mysql中锁定粒度最细的一种锁，表示只针对当前操作的行进行加锁。
    - 行级锁只有 InnoDB 引擎支持。
    - 行级锁能大大减少数据库操作的冲突。其加锁粒度最小，但加锁的开销也最大。
    - 特点：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低，并发度也最高。
- 表级锁
    - 表级锁是MySQL中锁定粒度最大的一种锁
    - 对当前操作的整张表加锁，它实现简单，资源消耗较少，被大部分MySQL引擎支持。
    - 特点：开销小，加锁快；不会出现死锁；锁定粒度大，发出锁冲突的概率最高，并发度最低。
- 页面锁
    - 开销和加锁速度介于表锁和行锁之间；会出现死锁；锁定粒度介于表锁和行锁之间，并发度一般
    - 页面锁在BDB引擎中才有，由于BDB已经被InnoDB所取代，我们只讨论MyISAM表锁和InnoDB行锁的问题
- 共享锁 (读锁)
    - 其他用户可以并发读取数据，但任何事务都不能对数据进行修改，直到已释放所有共享锁。
- 排他锁 (写锁)
    - 如果事务 T 对数据 A 加上排他锁后，则其他事务不能再对 A 加任何类型的封锁。
    - 持有排他锁的事务既能读数据，又能修改数据。

- 乐观锁(Optimistic Lock)

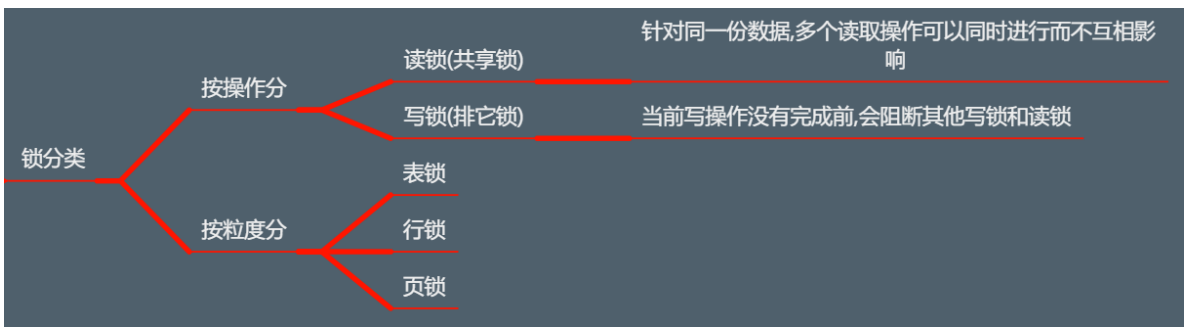
  假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。 乐观锁不能解决脏读的问题。

  乐观锁, 顾名思义，就是很乐观，每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断一下在此期间别人有没有去更新这个数据，可以使用版本号等机制。乐观锁适用于多读的应用类型，这样可以提高吞吐量，像数据库如果提供类似于write_condition机制的其实都是提供的乐观锁。

- 悲观锁(Pessimistic Lock)

  假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。

  悲观锁，顾名思义，就是很悲观，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会block直到它拿到锁。传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。



表锁操作

偏向于myisam存储引擎(innodb中也有),开销小, 加锁快, 无死锁,锁定粒度大, 发生锁冲突的概率最高, 并发支持最差。

特点：一张表只能一个人读，其他人排队。

示例

```
#查看表中有没有被锁
show open tables;

#加上读写锁
lock table stuinfo read;
lock table stuinfo write;

#解除锁
unlock tables;
```

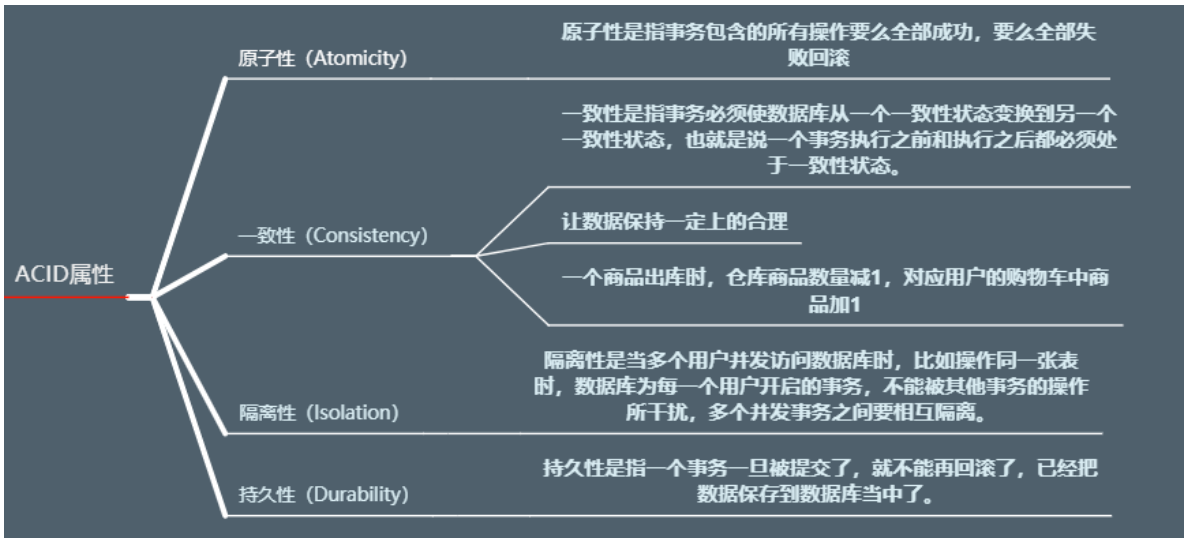
添加读锁(共享锁)，当锁是开启状态的时候

- 在当前的终端连接当中
  - 可以使用查询语句查询当前表，但是不可以进行其它操作
  - 不可以对其它表进行操作
- 在其它终端中进行操作
  - 可以使用查询语句查询当前表，可以使用其它操作，但是会一直阻塞到锁关闭以后在执行
  - 不可以对其它表进行操作

添加写锁(排它锁)，当锁是开启状态的时候

- 当前的终端链接的时候
  - 可以查询和操作主动被锁的当前表
  - 不可以从查询操作没有被主动锁(自动锁)的其它表
- 在其它终端中进行操作
  - 可以使用查询语句查询当前表，但是会一直阻塞到锁关闭以后在执行
  - 可以使用其它动作类型型操作.
- 使用方法:在查询的一瞬间锁上,查询完成以后释放

行锁操作



偏向InnoDB存储引擎,开销大, 加锁慢, 会出现死锁;锁定粒度最小, 发生锁冲突的概率最底,并发度也最高。

InnoDB与MyISAM的最大不同点:一是支持事务, 二是采用了行级锁。

并发事务中具体出现的问题

- 1.更新丢失问题

两个或多个事务选择同一行,然后基于最初选定的值更新该行时,由于每个事务都不知道其它事务的存在,就会发生丢失更新 问题,最后 的更新覆盖了其它事务所做的更新。

- 2.脏读

> 老板给你们发工资,2w/月
>
> 老板不小心输入出错20w,短信发送给你了,事务还没提交,修改回来在提交
>
> 入账2w

- 3.不可重复读

> 工资卡10w,买单,对方的系统中检测你卡中的余额足够
>
> 对象,把你的钱全部拿走,钱走了
>
> 余额不足

- 4.幻读

> 花费了2000,对象查帐10w
>
> 2w买了一台电脑
>
> 消费记录变成2.2w

查看隔离级别

- Read uncommitted （读未提交），就是一个事务可以读取另一个未提交事务的数据
- Read committed （读提交,不可重复读），一个事务要等另一个事务提交后才能读取数据
- Repeatable read (可重复度,效率高，已经可以防止各种问题了，默认级别)，就是在开始读取数据（事务开启）时，不再允许修改操作。
- Serializable (串行化的,效率低，非常有效，不是默认的级别)，在该级别下，事务串行化顺序执行，可以避免脏读、不可重复读与幻读。

```
#查看当前的隔离级别
select
@@global.tx_isolation,@@tx_isolation;
```

设置隔离级别

在测试的时候我们需要把自动提交事务给关闭，在mysql5.5以后默认是自动提交事务的

```
show variables like 'autocommit';     #查看事务的开启状态

set autocommit = 0     #0是关闭，1是开启
```

```
#全局设置
set global transaction isolation level
Serializable;

#当前会话终端设置
set session transaction isolation level
Serializable;
```

- 行锁必须是有添加过索引的列才能触发
- 在innodb中如果没有给某一个列或多个添加索引,那么默认使用表锁

间隙锁

范围查找的时候使用的多级行锁

```
update student set money=10000 where id>1
and id<5;
```