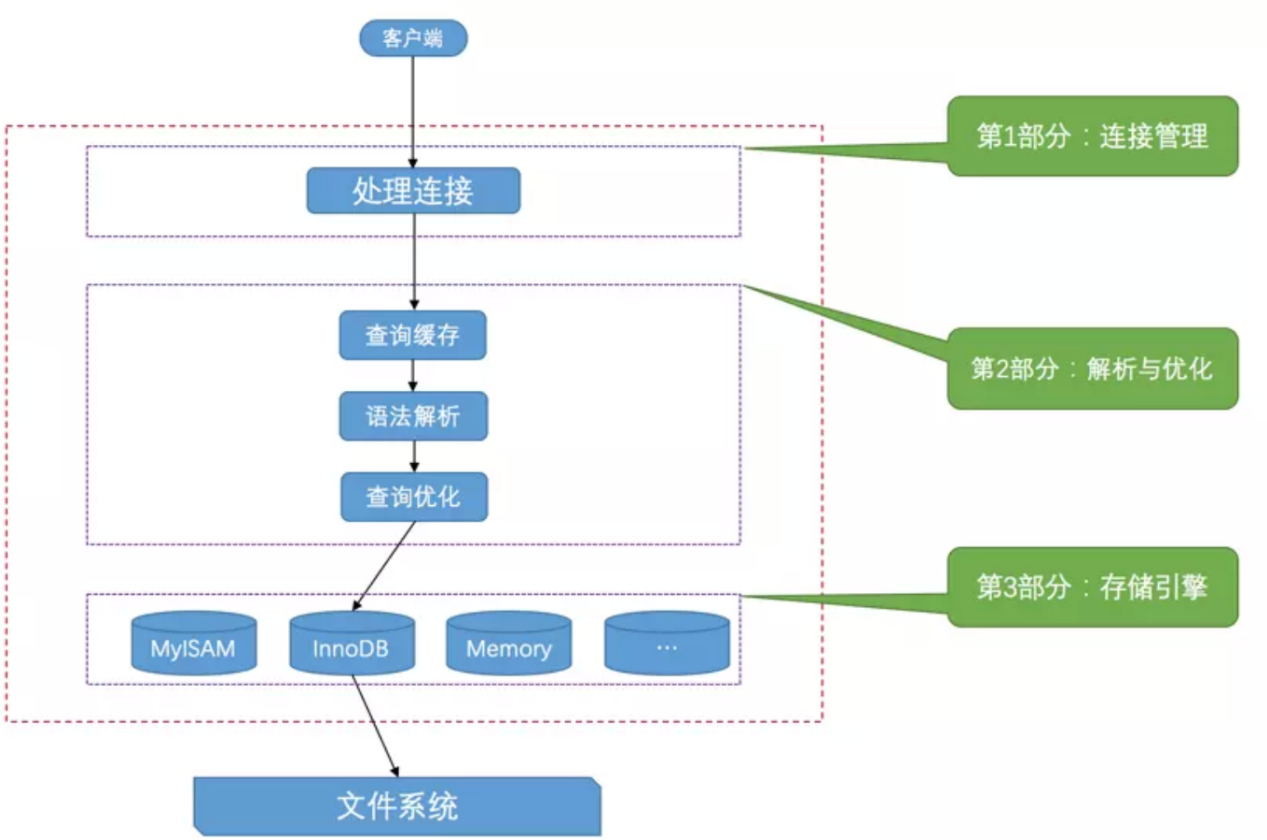
Mysql数据库表引擎与字符集

1.服务器处理客户端请求

其实不论客户端进程和服务器进程是采用哪种方式进行通信，最后实现的效果都是：**客户端进程向服务器进程发送一段文本（MySQL语句），服务器进程处理后再向客户端进程发送一段文本（处理结果）。**那服务器进程对客户端进程发送的请求做了什么处理，才能产生最后的处理结果呢？客户端可以向服务器发送增删改查各类请求，我们这里以比较复杂的查询请求为例来画个图展示一下大致的过程：



> 虽然查询缓存有时可以提升系统性能，但也不得不因维护这块缓存而造成一些开销，比如每次都要去查询缓存中检索，查询请求处理完需要更新查询缓存，维护该查询缓存对应的内存区域。从MySQL 5.7.20开始，不推荐使用查询缓存，并在MySQL 8.0中删除。

2.存储引擎

`MySQL`服务器把数据的存储和提取操作都封装到了一个叫`存储引擎`的模块里。我们知道`表`是由一行一行的记录组成的，但这只是一个逻辑上的概念，物理上如何表示记录，怎么从表中读取数据，怎么把数据写入具体的物理存储器上，这都是`存储引擎`负责的事情。为了实现不同的功能，`MySQL`提供了各式各样的`存储引擎`，不同`存储引擎`管理的表具体的存储结构可能不同，采用的存取算法也可能不同。

> 存储引擎以前叫做`表处理器`，它的功能就是接收上层传下来的指令，然后对表中的数据进行提取或写入操作。

为了管理方便，人们把`连接管理`、`查询缓存`、`语法解析`、`查询优化`这些并不涉及真实数据存储的功能划分为`MySQL server`的功能，把真实存取数据的功能划分为`存储引擎`的功能。各种不同的存储引擎向上边的`MySQL server`层提供统一的调用接口（也就是存储引擎API），包含了几十个底层函数，像"读取索引第一条内容"、"读取索引下一条内容"、"插入记录"等等。

所以在`MySQL server`完成了查询优化后，只需按照生成的执行计划调用底层存储引擎提供的API，获取到数据后返回给客户端就好了。

`MySQL`支持非常多种存储引擎：

存储引擎	描述
`ARCHIVE`	用于数据存档（行被插入后不能再修改）
`BLACKHOLE`	丢弃写操作，读操作会返回空内容
`CSV`	在存储数据时，以逗号分隔各个数据项
`FEDERATED`	用来访问远程表
`InnoDB`	具备外键支持功能的事务存储引擎
`MEMORY`	置于内存的表
`MERGE`	用来管理多个MyISAM表构成的表集合
`MyISAM`	主要的非事务处理存储引擎
`NDB`	MySQL集群专用存储引擎

3,MyISAM和InnoDB表引擎的区别

1) 事务支持

MyISAM不支持事务，而InnoDB支持。

> 事物：访问并更新数据库中数据的执行单元。事物操作中，要么都执行要么都不执行

2) 存储结构

MyISAM：每个MyISAM在磁盘上存储成三个文件。

- .frm文件存储表结构。
- .MYD文件存储数据。

- .MYI文件存储索引。

InnoDB：主要分为两种文件进行存储

- .frm 存储表结构
- .ibd 存储数据和索引 （也可能是多个.ibd文件，或者是独立的表空间文件）

3) 表锁差异

MyISAM：只支持表级锁，用户在操作myisam表时，select，update，delete，insert语句都会给表自动加锁，如果加锁以后的表满足insert并发的情况下，可以在表的尾部插入新的数据。 **InnoDB：支持事务和行级锁，是innodb的最大特色。**行锁大幅度提高了多用户并发操作的新能。但是InnoDB的行锁，只是在WHERE的主键是有效的，非主键的WHERE都会锁全表的。

4) 表主键

MyISAM：允许没有任何索引和主键的表存在，索引都是保存行的地址。 InnoDB：如果没有设定主键或者非空唯一索引，就会自动生成一个6字节的主键(用户不可见)，数据是主索引的一部分，附加索引保存的是主索引的值。InnoDB的主键范围更大，最大是MyISAM的2倍。

5) 表的具体行数

MyISAM：保存有表的总行数，如果select count() from table;会直接取出出该值。 InnoDB：没有保存表的总行数(只能遍历)，如果使用select count() from table；就会遍历整个表，消耗相当大，但是在加了wehre条件后，myisam和innodb处理的方式都一样。

6) CURD操作

MyISAM：如果执行大量的SELECT，MyISAM是更好的选择。 InnoDB：如果你的数据执行大量的INSERT或UPDATE，出于性能方面的考虑，应该使用InnoDB表。DELETE 从性能上InnoDB更优，但DELETE FROM table时，InnoDB不会重新建立表，而是一行一行的删除，在innodb上如果要清空保存有大量数据的表，最好使用truncate table这个命令。

7) 外键

MyISAM：不支持 InnoDB：支持

8) 查询效率

MyISAM相对简单，所以在效率上要优于InnoDB，小型应用可以考虑使用MyISAM。

推荐考虑使用InnoDB来替代MyISAM引擎，原因是InnoDB自身很多良好的特点，比如事务支持、存储 过程、视图、行级锁定等等，在并发很多的情况下，相信InnoDB的表现肯定要比MyISAM强很多。

另外，任何一种表都不是万能的，只用恰当的针对业务类型来选择合适的表类型，才能最大的发挥MySQL的性能优势。如果不是很复杂的Web应用，非关键应用，还是可以继续考虑MyISAM的，这个具体情况可以自己斟酌。

9）MyISAM和InnoDB两者的应用场景：

MyISAM管理非事务表。它提供高速存储和检索，以及全文搜索能力。如果应用中需要执行大量的SELECT查询，那么MyISAM是更好的选择。 InnoDB用于事务处理应用程序，具有众多特性，包括ACID事务支持。如果应用中需要执行大量的INSERT或UPDATE操作，则应该使用InnoDB，这样可以提高多用户并发操作的性能。现在默认使用InnoDB。

4.了解一下字符集和乱码

字符集简介

我们知道在计算机中只能存储二进制数据，那该怎么存储字符串呢？当然是建立字符与二进制数据的映射关系了，建立这个关系最起码要搞清楚两件事儿：

1. 你要把哪些字符映射成二进制数据？

   也就是界定清楚字符范围。

2. 怎么映射？

   将一个字符映射成一个二进制数据的过程也叫做`编码`，将一个二进制数据映射到一个字符的过程叫做`解码`。

人们抽象出一个`字符集`的概念来描述某个字符范围的编码规则

我们看一下一些常用字符集的情况：

- `ASCII`字符集

  共收录128个字符，包括空格、标点符号、数字、大小写字母和一些不可见字符。由于总共才128个字符，所以可以使用1个字节来进行编码，我们看一些字符的编码方式：

  ```
  'L' ->  01001100 (十六进制：0x4C，十进制：76)
  'M' ->  01001101 (十六进制：0x4D，十进制：77)
  ```

- `ISO 8859-1`字符集

  共收录256个字符，是在`ASCII`字符集的基础上又扩充了128个西欧常用字符(包括德法两国的字母)，也可以使用1个字节来进行编码。这个字符集也有一个别名`latin1`。

- `GB2312`字符集

  收录了汉字以及拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母。其中收录汉字6763个，其他文字符号682个。同时这种字符集又兼容`ASCII`字符集，所以在编码方式上显得有些奇怪：

  - 如果该字符在`ASCII`字符集中，则采用1字节编码。
  - 否则采用2字节编码。

  这种表示一个字符需要的字节数可能不同的编码方式称为`变长编码方式`。比方说字符串`'爱u'`，其中`'爱'`需要用2个字节进行编码，编码后的十六进制表示为`0xCED2`，`'u'`需要用1个字节进行编码，编码后的十六进制表示为`0x75`，所以拼合起来就是`0xCED275`。

  > 小贴士：我们怎么区分某个字节代表一个单独的字符还是代表某个字符的一部分呢？别忘了`ASCII`字符集只收录128个字符，使用0～127就可以表示全部字符，所以如果某个字节是在0～127之内的，就意味着一个字节代表一个单独的字符，否则就是两个字节代表一个单独的字符。

- `GBK`字符集

  `GBK`字符集只是在收录字符范围上对`GB2312`字符集作了扩充，编码方式上兼容`GB2312`。

- `utf8`字符集

  收录地球上能想到的所有字符，而且还在不断扩充。这种字符集兼容`ASCII`字符集，采用变长编码方式，编码一个字符需要使用1～4个字节，比方说这样：

```
'L' ->  01001100 (十六进制：0x4C)
'啊' ->  111001011001010110001010 (十六进制：0xE5958A)
```

> 小贴士： 其实准确的说，utf8只是Unicode字符集的一种编码方案，Unicode字符集可以采用utf8、utf16、utf32这几种编码方案，utf8使用1～4个字节编码一个字符，utf16使用2个或4个字节编码一个字符，utf32使用4个字节编码一个字符。更详细的Unicode和其编码方案的知识不是本书的重点，大家上网查查哈～ MySQL中并不区分字符集和编码方案的概念，所以后边唠叨的时候把utf8、utf16、utf32都当作一种字符集对待。

对于同一个字符，不同字符集也可能有不同的编码方式。比如对于汉字`'我'`来说，`ASCII`字符集中根本没有收录这个字符，`utf8`和`gb2312`字符集对汉字`我`的编码方式如下：

```
utf8编码：111001101000100010010001 (3个字节，十六进制表示是：0xE68891)
gb2312编码：1100111011010010 (2个字节，十六进制表示是：0xCED2)
```

5.MySQL中的utf8和utf8mb4

我们上边说`utf8`字符集表示一个字符需要使用1～4个字节，但是我们常用的一些字符使用1～3个字节就可以表示了。而在`MySQL`中字符集表示一个字符所用最大字节长度在某些方面会影响系统的存储和性能，所以设计`MySQL`的大叔偷偷的定义了两个概念：

- `utf8mb3`：阉割过的`utf8`字符集，只使用1～3个字节表示字符。
- `utf8mb4`：正宗的`utf8`字符集，使用1～4个字节表示字符。

有一点需要大家十分的注意，在`MySQL`中`utf8`是`utf8mb3`的别名，所以之后在`MySQL`中提到`utf8`就意味着使用1~3个字节来表示一个字符，如果大家有使用4字节编码一个字符的情况，比如存储一些emoji表情啥的，那请使用`utf8mb4`。

字符集的查看

`MySQL`支持好多好多种字符集，查看当前`MySQL`中支持的字符集可以用下边这个语句：

```
show charset;
```