# 概述

MySQL 中的handler接口是一个抽象层，它提供了一组用于在低级别与存储引擎交互的功能。它允许绕过SQL层直接操作表数据。handler接口的底层原理可以解释如下：

1、Table Opening：当使用handler接口访问表时，调用存储引擎的ha\_open()函数。此函数初始化访问表所需的必要数据结构和资源。

2、记录导航（Record Navigation）：handler接口提供类似ha\_index\_read()和ha\_index\_next()的函数，用于在表的索引记录中导航。这些函数检索和遍历索引条目，允许根据提供的条件访问特定行。

3、数据操作（Data Manipulation）：handler接口提供ha\_write\_row()、ha\_delete\_row()和ha\_update\_row()等函数来执行数据操作。这些函数直接与存储引擎交互以插入、删除或更新表中的行。

4、事务处理：handler接口支持事务操作。它允许您使用 ha\_start\_stmt()、ha\_commit\_trans()和ha\_rollback\_trans()等函数启动、提交或回滚事务。

5、Table Closing：对表的操作完成后，handler接口调用存储引擎的ha\_close()函数释放与表相关的所有资源。

请务必注意，handler接口的行为和功能因所使用的特定存储引擎（例如 InnoDB、MyISAM）而异。每个存储引擎都以自己的方式实现handle接口，提供不同的功能和优化。

使用handler接口需要深入了解存储引擎的内部结构和数据结构。它通常由需要对数据访问进行细粒度控制并希望直接与存储引擎层交互的存储引擎开发人员或高级用户使用。

# 原理

## Handler API继承关系

sql\_alloc

->handler

->ha\_archive

->ha\_blackhole

->ha\_example

->ha\_federated

->ha\_heap

->ha\_innobase

->ha\_myisam

->ha\_ndbcluster

->ha\_partition

### Sql\_alloc

Sql\_alloc：不要被其名称所误导，这个类主要实现的是小内存的分配。

在修改MySQL时，申请小内存尽量使用sql\_alloc进行分配，而不是my\_malloc()。Sql\_alloc是从会话连接的线程池中分配内存的，而my\_malloc仅仅是标准malloc的包装，sql\_alloc分配的内存只持续到查询结束，如果想要获取永久内存块，则需要my\_malloc进行分配。

### Handlerton

Handlerton（handler.cc）：提供的是存储引擎的一些特性，为所有存储引擎提供标准接口。比如checkpoint、commit、rollback。同一个引擎跨表的操作需要在handlerton里面完成，比如说引擎的初始化，跨表的事务。

name(const char\*)：使用show engines等命令时，返回的存储引擎名

comment(const char\*)：存储引擎的注释

db\_type(enum legacy\_db\_type)：标识该库使用了何种存储引擎

savepoint\_offset(uint)：为该存储引擎准备的回滚保留点大小

Flags(uint32)：标识符，标识存储引擎支持的特性

binlog\_func：回调二进制日志处理函数

bnlog\_log\_query：查询二进制日志

sow\_status：返回存储引擎的状态和基本信息

prtition\_flags：返回分区状态

fll\_files\_table：MySQL Cluster填写文件内容

set\_cursor\_read\_view：转化到一个特定的游标视图

sart\_consistent\_snapshot：需要并发读取数据时调用的方法

### Handler

Handler（handler.cc）：提供存储引擎数据处理的方法（表的基本操作）。比如打开或者关闭表、扫描索引和数据，都是在handler里面定义的接口。

它与存储引擎之间依赖handlerton标准进行通讯。每个handler的子类进行初始化对象的时候，必须向构造函数传递一个handlerton对象的引用。Handler类有很多纯虚函数，有些必须在handler子类里面实现。这些函数处理存储引擎的底层IO方式。

#### 重要字段

ht(const handlerton\*)：该Handler使用的存储引擎

ref(byte\*)：指向当前行的指针

dupp\_ref(byte\*)：指向重复行的指针

data\_file\_lengh(ulonglong)：数据文件的长度

max\_data\_file\_length(ulonglong)：数据文件的最大长度

index\_file\_length(ulionglong)：索引文件的长度

max\_index\_file+length：索引文件的最大长度

delete\_length(ulonglong)：数据文件空闲字节

auto\_increment\_value(ulonglong)：当前自增值

records(ha\_rows/ulong)：表中的记录数

deleted(ha\_rows)：表中被删除的记录数

mean\_rec\_length(ulong)：物理记录的长度

create\_time/check\_time/update\_time(time\_t)：表的创建、检查、修改时间

#### 主要接口

##### 初始化

ha\_index\_init：索引访问开始前调用

ha\_index\_end：索引访问结束前调用

ha\_rnd\_init：随机读取开始前调用

ha\_rnd\_end：随机读取结束前调用

ha\_reset：在表打开后重置文件状态，这个函数在每个语句调用了它所有需要的表后被调用

start\_bulk\_insert/update/delete：大批量插入、修改、删除启动前调用

exec\_bulk\_update/delete：大批量插入、修改、删除执行前调用

end\_bulk\_update/delete：大批量插入、修改、删除后做必要的状态重置和清理

##### 索引操作

index\_read：顺序读取指定长度的索引

index\_read\_idx：从指定位置读取指定长度的索引

index\_next：转移索引指针到下一个位置

index\_prev：转移索引指针到前一个位置

index\_first：转移索引指针到第一个为止

index\_last：转移索引指针到最后一格为止

##### 读记录操作

read\_range\_list：读取两个结束点之间的第一行数据

read\_range\_next：读取两个结束点之间的下一行数据

rnd\_pos：跳转到任意一行的指针

rnd\_next：跳转到下一行指针

read\_first\_row：读取第一行数据

##### 写记录操作

write\_rows：写入一行数据

update\_row：修改一行数据

delete\_row：删除一行数据

##### 表操作

ha\_repair：REPAIR命令执行时调用

rename\_table：RENAME TABLE时调用

delete\_table：DROP TABLE时调用

drop\_table：只有ha\_heap重载的这个，用于临时表删除

delete\_all\_rows：TRUNCATE TABLE时调用

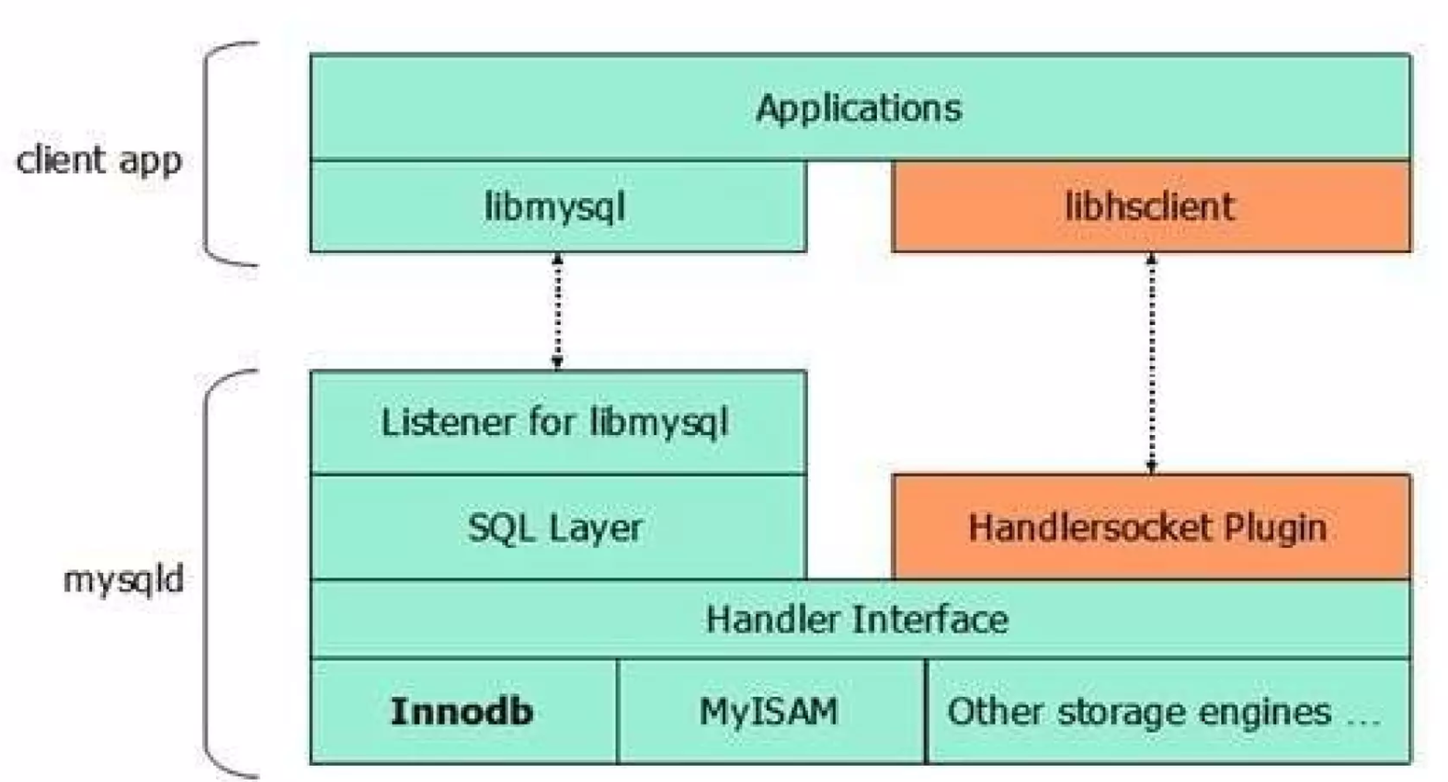
create：创建新表时调用

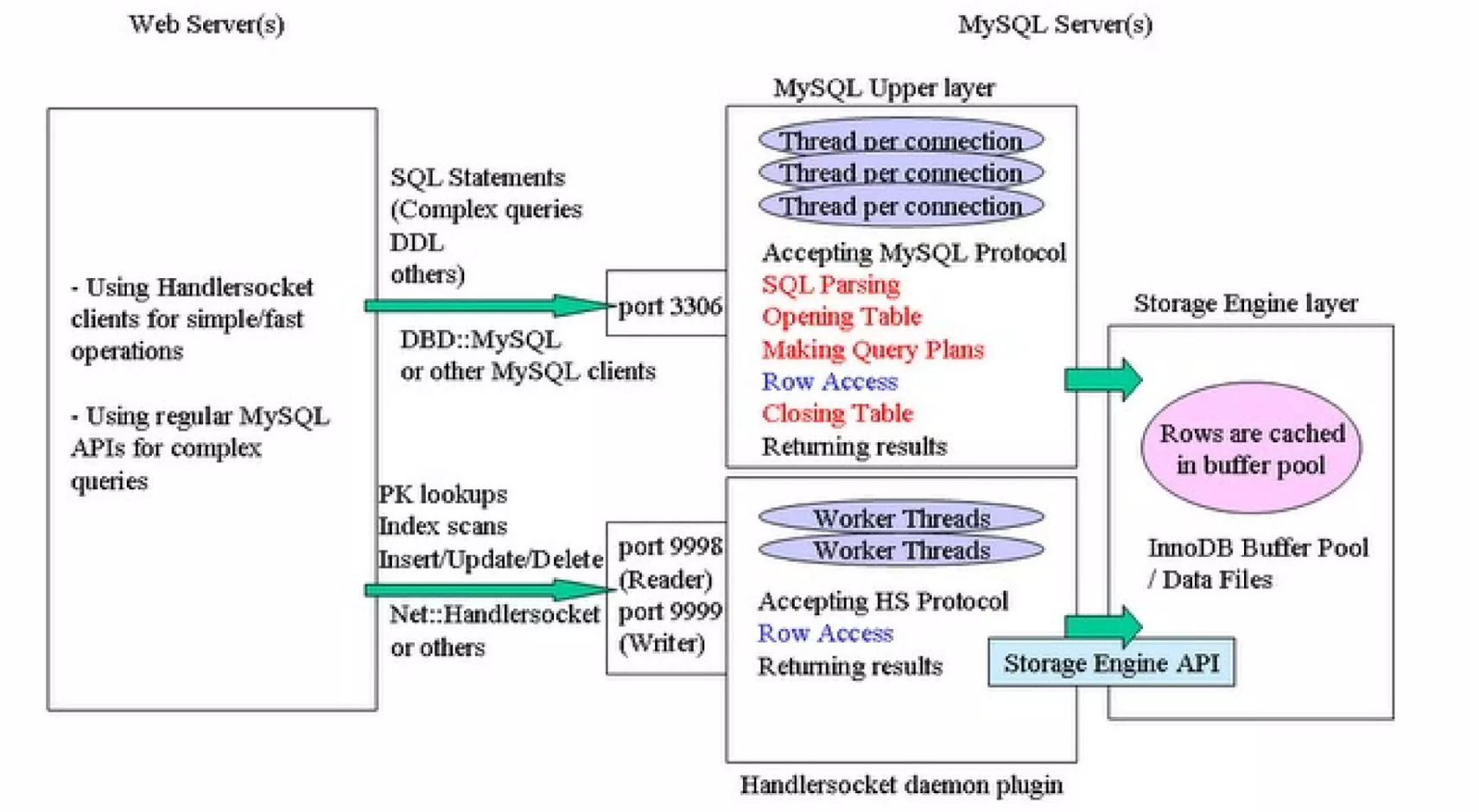
## 架构

与存储引擎交互有两种方式：

1. SQL
2. Handler接口

### Handler Socket





### SQL vs Handler

MySQL SQL和Handler执行在数据访问的方式和性能上存在一些区别和联系：

1、SQL执行：传统的MySQL查询通过SQL语句进行操作，包括解析、优化和执行等步骤。SQL语句由应用程序发送到MySQL服务器，服务器会解析语句、确定执行计划并执行查询，然后返回结果给应用程序。SQL执行适用于广泛的查询场景，可以进行复杂的连接、聚合、排序等操作。

2、Handler执行：Handler Socket是直接与InnoDB存储引擎进行交互的插件，绕过SQL层，以内存级别的速度进行数据访问。Handler Socket通过TCP/IP套接字连接到MySQL服务器，并使用一组简单的命令来执行读取、写入、更新和删除操作。Handler执行适用于以键为基础的数据访问，快速按主键或范围检索数据的场景。

#### 联系

- Handler Socket和SQL执行都是用于访问MySQL数据库的方式，它们都是通过与MySQL服务器进行通信来操作数据。

- 在某些特定场景下，Handler Socket可以替代SQL执行，通过直接与存储引擎交互来提高查询性能。

- Handler Socket可以作为一种选择，在需要高速读取或写入大量数据的情况下，通过绕过SQL层减少不必要的开销。

#### 区别

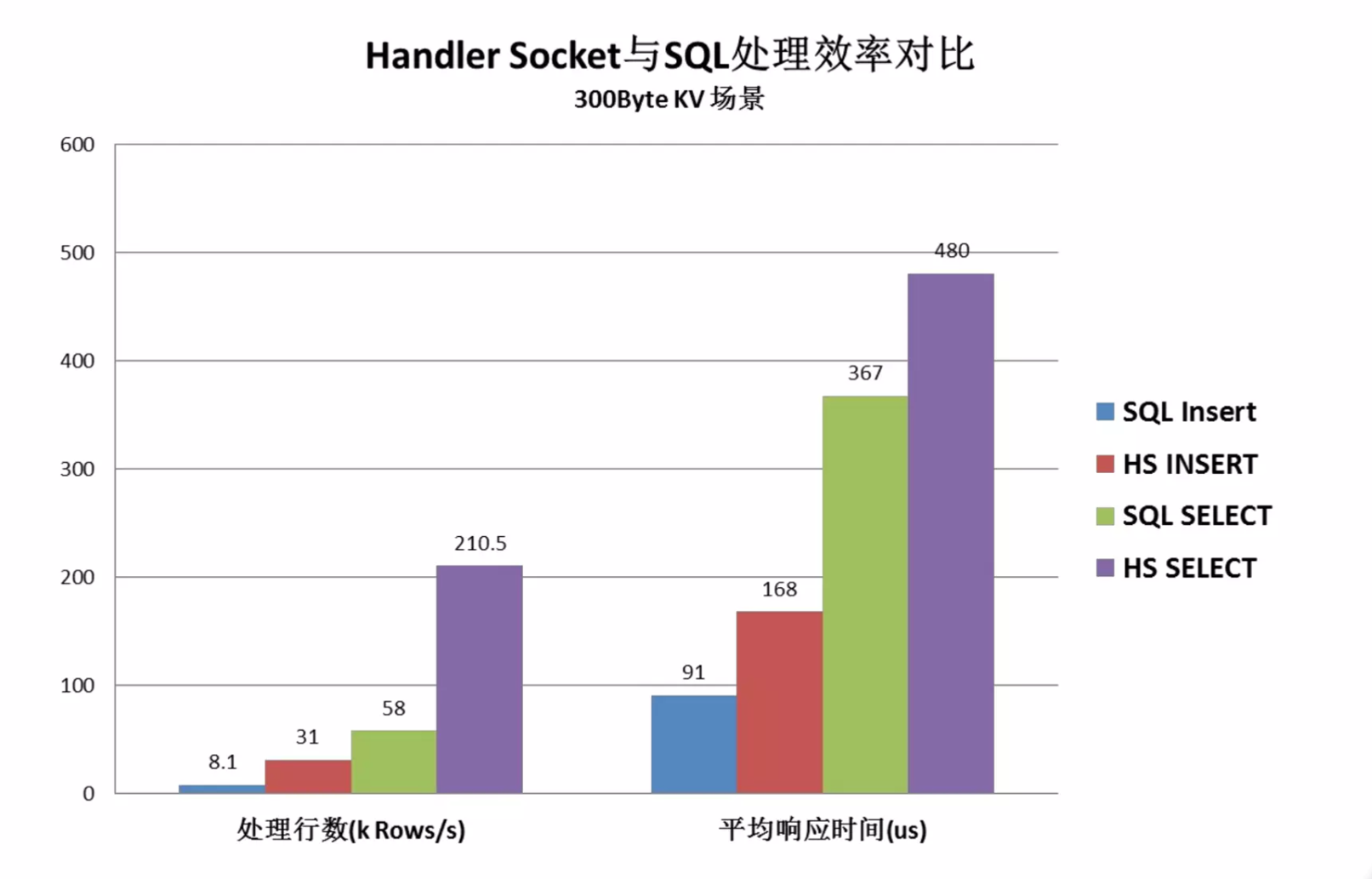
- SQL执行涉及到SQL语句的解析、优化和执行，而Handler Socket绕过了这些步骤，直接与存储引擎交互，从而避免了SQL层的开销。

- SQL执行适用于各种复杂查询，可以进行连接、聚合、排序等操作；而Handler Socket适用于以键为基础的简单数据访问，主要用于快速读取或写入数据。

- 使用SQL执行可以利用数据库的完整功能，如事务、触发器、存储过程等；而Handler Socket主要关注于基本的数据读写操作，不涉及数据库的高级特性。

根据具体的需求和场景，选择合适的数据访问方式可以提升查询性能和满足应用程序的需求。

性能比对：



举例：

SELECT \* FROM t LIMIT 1;

* read\_first\_row：直接读取数据文件第一行数据

SELECT c1 FROM t ORDER BY c1 LIMIT 1;

* index\_read\_last：从c1索引末尾读取一行数据

SELCT \* FROM t WHERE c1 BETWEEN #start# AND #end# LIMIT 10;

* index\_read读取#start#的索引位置i1，index\_read读取#end#的索引位置i2，调用read\_range\_first(i1,i2)读取第一行，然后调用read\_range\_next一直读取10行，并且compare\_key没有返回越界。

## Handler API

### Sql\_cmd\_handler\_open

### Sql\_cmd\_handler\_read

### Sql\_cmd\_handler\_close

# 使用

参考：

<https://developer.aliyun.com/article/41172>

## HANDLER OPEN

## HANDLER READ

## HANDLER CLOSE