浅谈MySql的存储引擎（表类型）

什么是MySql数据库

通常意义上，数据库也就是数据的集合，具体到计算机上数据库可以是存储器上一些文件的集合或者一些内存数据的集合。

我们通常说的MySql数据库，sql server数据库等等其实是数据库管理系统，它们可以存储数据，并提供查询和更新数据库中的数据的功能等等。根据数据库如何存储数据和如何操作数据的实现机制不同，这些数据库之间即有区别又有共同点。

MySql数据库是开放源代码的关系型数据库。目前，它可以提供的功能有：支持sql语言、子查询、存储过程、触发器、视图、索引、事务、锁、外键约束和影像复制等。在后期，我们会详细讲解这些功能。

同Oracle 和SQL Server等大型数据库系统一样，MySql也是客户/服务器系统并且是单进程多线程架构的数据库。

MySql区别于其它数据库系统的一个重要特点是支持插入式存储引擎。

那么什么是存储引擎呢?

存储引擎说白了就是如何**存储数据、如何为存储的数据建立索引和如何更新、查询数据**等技术的实现方法。因为在关系数据库中数据的存储是以表的形式存储的，所以存储引擎也可以称为表类型（即存储和操作此表的类型）。

在**Oracle 和SQL Server等数据库中只有一种存储引擎**，所有数据存储管理机制都是一样的。

而**MySql数据库提供了多种存储引擎。**用户可以根据不同的需求为数据表选择不同的存储引擎，用户也可以根据自己的需要编写自己的存储引擎。

MySql中有哪些存储引擎?

1 **MyISAM：**这种引擎是mysql最早提供的。这种引擎又可以分为静态MyISAM、动态MyISAM 和压缩MyISAM三种：

静态MyISAM：**如果数据表中的各数据列的长度都是预先固定好的，服务器将自动选择这种表类型。因为数据表中每一条记录所占用的空间都是一样的，所以这种表存取和更新的效率非常高。当数据受损时，恢复工作也比较容易做。

动态MyISAM：**如果数据表中出现varchar、xxxtext或xxxBLOB字段时，服务器将自动选择这种表类型。相对于静态MyISAM，这种表**存储空间比较小，但由于每条记录的长度不一，所以多次修改数据后，数据表中的数据就可能离散的存储在内存中，进而导致执行效率下降。**同时，内存中也可能会出现很多碎片。因此，这种类型的表要经常用optimize table 命令或优化工具来进行碎片整理。

压缩MyISAM：**以上说到的两种类型的表都可以用**myisamchk工具压缩。这种类型的表进一步减小了占用的存储，但是这种表压缩之后不能再被修改。另外，因为是压缩数据，所以这种表在读取的时候要先时行解压缩。**

但是，不管是何种MyISAM表，**目前它都不支持事务，行级锁和外键约束的功能。**

2 MyISAM Merge引擎：**这种类型是MyISAM类型的一种变种。**合并表是将几个相同的MyISAM表合并为一个**虚表。常应用于日志和数据仓库。**

3 InnoDB：

InnoDB表类型可以看作是对MyISAM的**进一步更新产品，它提供了事务、行级锁机制和外键约束的功能。**

我知道一般情况下插入查询比较多的适合用myisam， 更新比较多或者财务之类的表用innoDB比较好，

myisam 不支持事务，你试试开两个窗口，其中一个插入一条数据，再没有commit的情况下，另外一个也能看到。

innodb 支持事务，具体的我就不多说了，你应该懂得。

mysql中 myisam 引擎不支持事务的概念，多用于数据仓库这样查询多而事务少的情况，速度较快。

mysql中 innoDB 引擎支持事务的概念，多用于web网站后台等实时的中小型事务处理后台。

而oracle没有引擎的概念，oracle有OLTP和OLAP模式的区分，两者的差别不大，只有参数设置上的不同。

oracle无论哪种模式都是.....

---

4 memory(heap)：

这种类型的数据表**只存在于内存中。它使用散列索引，所以数据的存取速度非常快。**

因为是存在于内存中，所以这种类型常**应用于临时表中。**

5 archive：

这种类型**只支持select 和 insert语句，而且不支持索引。常应用于日志记录和聚合分析方面。**

当然MySql支持的表类型不止上面几种。

下面我们介绍一下如何查看和设置数据表类型。

MySql中关于存储引擎的操作

1 查看数据库可以支持的存储引擎 用show engines; 命令可以显示当前数据库支持的存储引擎情况，如图1所示：

图1 数据库的存储引擎

```
mysql> show engines;
+------------+---------+------------------------------------------------------------+--------------+------+------------+
| Engine     | Support | Comment                                                    | Transactions | XA   | Savepoints |
+------------+---------+------------------------------------------------------------+--------------+------+------------+
| InnoDB     | YES     | Supports transactions, row-level locking, and foreign keys | YES          | YES  | YES        |
| MRG_MYISAM | YES     | Collection of identical MyISAM tables                      | NO           | NO   | NO         |
| BLACKHOLE  | YES     | /dev/null storage engine (anything you write to it disappears) | NO       | NO   | NO         |
| CSV        | YES     | CSV storage engine                                         | NO           | NO   | NO         |
| MEMORY     | YES     | Hash based, stored in memory, useful for temporary tables  | NO           | NO   | NO         |
| FEDERATED  | NO      | Federated MySQL storage engine                             | NULL         | NULL | NULL       |
| ARCHIVE    | YES     | Archive storage engine                                     | NO           | NO   | NO         |
| MyISAM     | DEFAULT | Default engine as of MySQL 3.23 with great performance     | NO           | NO   | NO         |
+------------+---------+------------------------------------------------------------+--------------+------+------------+
8 rows in set (0.00 sec)
```

由上图可见当前系统的默认数据表类型是MyISAM。当然，我们可以通过修改数据库配置文件中的选项，设定默认表类型。 **2 查看表的结构等信息的若干命令** 要查看表的定义结构等信息可以使用以下几种命令： 2.1Desc[ribe] **tablename**; //查看数据表的结构 例如，查看表t1的结构，可得下图。

图2：查看表t1的结构

```
mysql> desc t1;
+-------+-----------+------+-----+-------------------+-----------------------------+
| Field | Type      | Null | Key | Default           | Extra                       |
+-------+-----------+------+-----+-------------------+-----------------------------+
| id    | int(11)   | NO   | PRI | NULL              | auto_increment              |
| sno   | char(20)  | NO   |     | NULL              |                             |
| cno   | int(11)   | YES  |     | NULL              |                             |
| ts    | timestamp | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-------+-----------+------+-----+-------------------+-----------------------------+
4 rows in set (0.01 sec)
```

2.2 Show create table **tablename**; //显示表的创建语句 同上查询表t1,得下图：

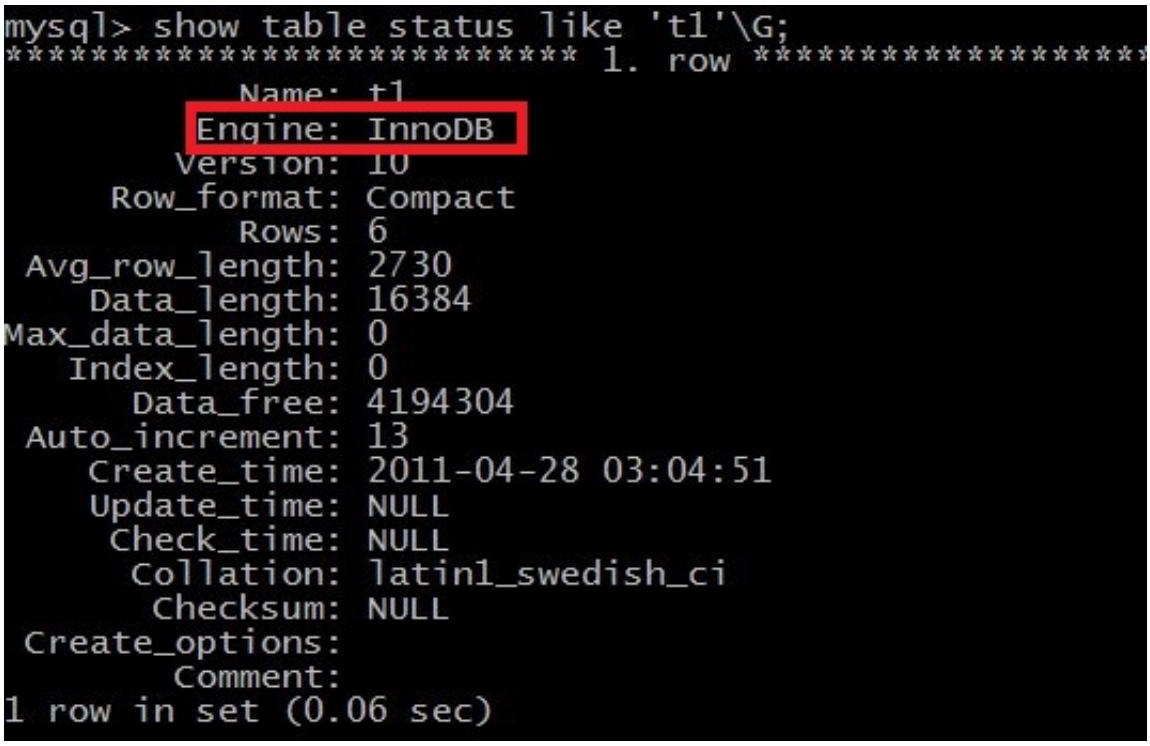
图3 显示创建表t1的语句

```
mysql> show create table t1;
| Table | Create Table |
| t1    | CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `sno` char(20) NOT NULL,
  `cno` int(11) DEFAULT NULL,
  `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=latin1 |
1 row in set (0.00 sec)
```

2.3 show table status like ‘**tablename**’\G显示表的当前状态值

同上查询表t1,得下图：

图4 显示表t1 的当前状态值



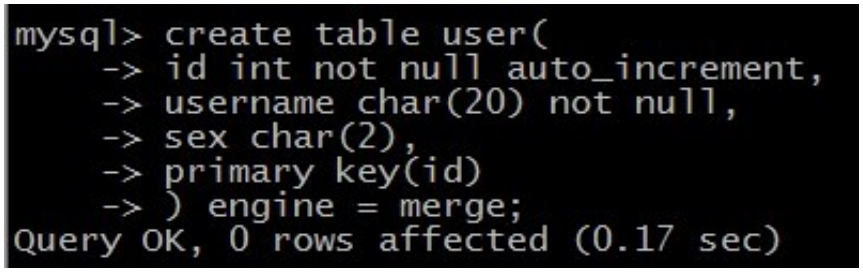
综上可见，后两种方式都可以帮助我们查看某一表的存储引擎类型（图中已用红色方框标出）。 **3 设置或修改表的存储引擎** 3.1创建数据库表时设置存储存储引擎的基本语法是： Create table ***tableName***(***columnName***(列名1) type(数据类型) attri(属性设置), ***columnName***(列名2) type(数据类型) attri(属性设置), ……..) engine = ***engineName***

例如,假设要创建一个名为user的表,此表包括id,用户名username和性别sex三个字段，并且要设置表类型为merge。则可用如下的方式创建此数据表,

```
create table user(

  id int not null auto_increment,

  username char(20) not null,

  sex char(2),

  primary key(id)

) engine=merge
```

具体执行结果见下图:

图5 创建表user



查看创建后表user的信息，可见表的当前存储引擎是merge，如图所示：

图6 显示表t1 的当前状态值

```
mysql> show table status like 'user'\G
*************************** 1. row ***************************
           Name: user
         Engine: MRG_MYISAM
        Version: 10
     Row_format: Fixed
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: 0
    Create_time: NULL
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.05 sec)
```

3.2修改存储引擎，可以用命令Alter table **tableName** engine =**engineName**

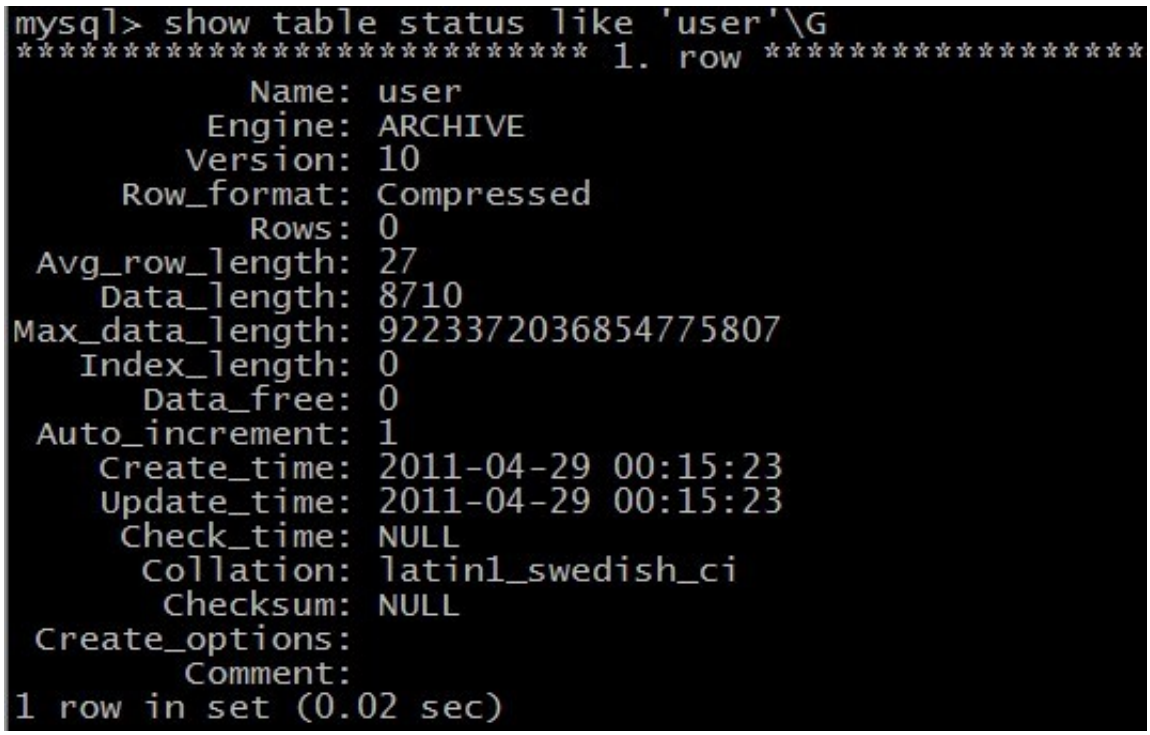
假如，若需要将表user的存储引擎修改为archive类型，则可使用命令alter table user engine=archive。如下图所示：

图7 修改表user的存储引擎

```
mysql> alter table user engine = archive;
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

查看修改后的表类型，可见表类型已经变为archive类型。

图8 显示表user修改后的状态值



小结

在本文中主要介绍了什么是MySql数据库，并进一步引出了它的一个重要特性， 即**插入式的多存储引擎机制。**然后，简单介绍了什么是存储引擎和MySql中几种主要的存储引擎。最后，介绍了如何查看数据库支持的所有存储引擎，如何查看数据库表的存储引擎类型及如何设置或修改表的存储引擎类型。刚刚入门学习MySql,文中有错误之处，还请大家多多指导!

---

mysql有多种存储引擎，目前常用的是 MyISAM 和 InnoDB 这两个引擎，除了这两个引擎以为还有许多其他引擎，有官方的，也有一些公司自己研发的。这篇文章主要简单概述一下常用常见的 MySQL 引擎，一则这是面试中常被问到的问题，二则这也是数据库设计中不可忽略的问题，用合适的引擎可以更好的适应业务场景，提高业务效率。

MyISAM

MyISAM 是 mysql 5.5.5 之前的默认引擎，它支持 B-tree/FullText/R-tree 索引类型。

锁级别为表锁，表锁优点是**开销小，加锁快；缺点是锁粒度大，发生锁冲动概率较高，容纳并发能力低，这个引擎适合查询为主的业务。**

此引擎不支持事务，也不支持外键。

MyISAM强调了快速读取操作。它存储表的行数，于是SELECT COUNT(*) FROM TABLE时只需要直接读取已经保存好的值而不需要进行全表扫描。

InnoDB

InnoDB 存储引擎最大的亮点就是**支持事务，支持回滚，它支持 Hash/B-tree /R-tree索引类型。**

锁级别为行锁，行锁优点是适用于高并发的频繁表修改，高并发是性能优于 MyISAM。缺点是系统消耗较大，索引**不仅缓存自身，也缓存数据，相比 MyISAM 需要更大的内存。**

InnoDB 中**不保存表的具体行数，也就是说，执行 select count(*) from table时，InnoDB 要扫描一遍整个表来计算有多少行。**

支持事务，支持外键。

ACID 事务

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。

Memory

Memory 是内存级别存储引擎，数据存储在内存中，所以他能够存储的数据量较小。

因为内存的特性，存储引擎对数据的一致性支持较差。锁级别为表锁，不支持事务。但访问速度非常快，并且默认使用 hash 索引。

Memory存储引擎使用存**在内存中的内容来创建表，每个Memory表只实际对应一个磁盘文件，在磁盘中表现为.frm文件。**

总结

	MyISAM	**InnoDB**
存储结构	每张表被存放在三个文件：**frm-格**定义MYD(MYData)-数据文件MYI(MYIndex)-索引文件	**所有的表都保存在同一个数据文件中（也可能是多个文件，或者是独立的表空间文件），InnoDB表的大小只受限于操作系统文件的大小，一般为2GB**
存储空间	MyISAM可被压缩，存储空间较小	InnoDB的表需要更多的内存和存储，它会在主内存中建立其专用的缓冲池用于高速缓冲数据和索引
可移植性、备份及恢复	由于MyISAM的数据是以文件的形式存储，所以在跨平台的数据转移中会很方便。在备份和恢复时可单独针对某个表进行操作	免费的方案可以是拷贝数据文件、备份 binlog，或者用mysqldump，在数据量达到几十G的时候就相对痛苦了
事务安全	不支持 每次查询具有原子性	支持 具有事务(commit)、回滚(rollback)和崩溃修复能力(crash recovery capabilities)的事务安全

		(transaction-safe (ACID compliant))型表
AUTO_INCREMENT	MyISAM表可以和其他字段一起建立联合索引	InnoDB中必须包含只有该字段的索引
SELECT	MyISAM更优	
INSERT		InnoDB更优
UPDATE		InnoDB更优
DELETE		InnoDB更优 它不会重新建立表，而是一行一行的删除
COUNT without WHERE	MyISAM更优。因为MyISAM保存了表的具体行数	InnoDB没有保存表的具体行数，需要逐行扫描统计，就很慢了
COUNT with WHERE	一样	一样，InnoDB也会锁表
锁	只支持表锁	支持表锁、行锁 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
外键	不支持	支持
FULLTEXT全文索引	支持	不支持（5.6.4以上支持英文全文索引） 可以通过使用Sphinx从InnoDB中获得全文索引，会慢一点

互联网项目中随着硬件成本的降低及缓存、中间件的应用，一般我们选择都以 InnoDB 存储引擎为主，很少再去选择 MyISAM 了。而业务真发展的一定程度时，自带的存储引擎无法满足时，这时公司应该是有实力去自主研发满足自己需求的存储引擎或者购买商用的存储引擎了。