

# 前言

谁说MySQL查询千万级别的数据很拉跨？我今天就要好好的和你拉拉家常，畅谈至深夜，一起过除夕！这篇文章也是年前的最后一篇，希望能给大家些许收获，不知不觉查找文档和参考实体书籍就写了这么多，自己都感觉到意外。不禁感慨到，知道的越多，才知道不知道的更多。

开发人员或者是DBA都应该关注MySQL使用的存储引擎，选择合适存储引擎对你的应用性能提升是明显的。在阅读到本文的时候，肯定是一定的MySQL或者其它数据库基础的，不然有些地方看着会很费劲。重点地方，我都进行了加粗处理，这样更容易获取关键知识点。

关于存储引擎，一篇文章也不可能面面俱到，对个人觉得比较重要、于工作有益的方面进行阐述。如果真的去深挖，估计得一本书的篇幅。顺带还介绍一些数据类型选择、字符集设置、索引的使用；视图、存储过程、函数以及触发器啊等等会在下一篇博文进行详细的描述。但本文不会做太详细的叙述。本篇文章以存储引擎的选择为核心，如果有出现瑕疵的地方，希望您能留下宝贵的建议。

使用MyISAM存储引擎创建的表tolove，查询存储有1kw数据的表tolove。

```
MySQL [(none)]> select count(*) from test.tolove;
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (0.000 sec)
```

再看演示使用InnoDB存储引擎创建的表test，同样为了演示，事先随机生成了1kw条数据。

```
MySQL [(none)]> select count(*) from test.test;
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (3.080 sec)
```

今天发现了一个神奇的参数：`-site:xxxx.net`

# 正文

## 一、存储引擎的选择（表类型）

### 1、存储引擎的介绍

与到多数关系型数据库的区别在于MySQL有一个存储引擎的概念，针对不同的存储需求可以选择最合适的存储引擎。MySQL中的插件式的存储引擎是其一大特色，用户可以根据应用的需求选择如何存储、是否索引，是否使用事务。嘿嘿，你也可以根据业务环境去适配最适合自己的存储引擎。

Oracle从中嗅到了商机，收购了MySQL，从此有了企业版（商业支持）。社区版依旧可以免费下载。另一大魅力也是因为开源，社区高度活跃，人人都可贡献。接下来介绍几种使用比较多的存储引擎，存储引擎并无优劣之分，有的只是谁更适合对应的生产业务环境。

MySQL5.0中支持的存储引擎有FEDERATED、**MRG\_MYISAM**、**MyISAM**、BLACKHOLE、CSV、**MEMORY**、ARCHIVE、**NDB Cluster**、BDB、EXAMPLE、**InnoDB**（MySQL5.5以及MariaDB10.2之后的默认存储引擎）、PERFORMANCE\_SCHEMA（非常规存储数据引擎）。下面给出MySQL与MariaDB支持的存储引擎的对比，可以看出MariaDB新增了Aria引擎：

```
1 SHOW ENGINES; /** 查看当前数据库支持的存储引擎 **/
2 SELECT VERSION(); /** 查看数据库版本 **/
```

Linux版本MySQL5.6.51

Engine	Support	Comment	Transactions	XA	Savepoints
FEDERATED	NO	Federated MySQL	(NULL)	(NULL)	(NULL)
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine	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
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

```
48 /** 查询当前数据库支持的引擎 **/
49 -- 参数文件中设置default-table-type
50 SHOW ENGINES; /** 查看当前数据库支持的存储引擎 **/
51 SELECT VERSION(); /** 查看数据库版本 **/
```

Windows版本MariaDB10.5.6

Engine	Support	Comment	Transactions	XA	Savepoints
CSV	YES	Stores tables as CSV files	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
Aria	YES	Crash-safe table engine, built on the MyISAM storage engine	NO	NO	NO
MyISAM	YES	Non-transactional MyISAM tables	NO	NO	NO
SEQUENCE	YES	Generated table engine	YES	NO	YES
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

## 查看存储引擎

通过MySQL登录自带的字符界面输入 `show engines\G;` 或者使用支持MySQL查询的工具SQLyog、phpMyAdmin、MySQL workbench等查询支持的引擎，这里只展示部分哟：

```
[test@cnwangk ~]$ mysql -uroot -p
Enter password:
mysql> show engines\G;
***** 2. row *****
    Engine: MRG_MYISAM
    Support: YES
    Comment: Collection of identical MyISAM tables
Transactions: NO
      XA: NO
Savepoints: NO
***** 3. row *****
    Engine: MyISAM
    Support: YES
    Comment: MyISAM storage engine
Transactions: NO
      XA: NO
Savepoints: NO
***** 6. row *****
    Engine: MEMORY
    Support: YES
    Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
      XA: NO
Savepoints: NO
***** 8. row *****
    Engine: InnoDB
    Support: DEFAULT
    Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
      XA: YES
```

```
Savepoints: YES
9 rows in set (0.00 sec)
```

#### 作用描述：

- Engine：引擎名称（描述）；
- Support：当前版本数据库是否支持该存储引擎，YES：支持、NO：不支持；**Supports transactions, row-level locking, and foreign keys**，个人字面上翻译这段话：**支持事务、行级锁和外键**；
- Comment：对该存储引擎的详情描述，比如描述该引擎否支持事务和外键；
- Transactions：对该存储引擎是否支持事务的描述，YES：支持、NO：不支持；
- XA：是否满足XA规范。XA规范是开放群组关于分布式事务处理(DTP)的规范。YES：支持、NO：不支持；
- Savepoints：字面意思是保存点，对事物控制是否支持，YES：支持、NO：不支持。

小声哔哔，如果你能阅读明白官方的一些英文文档，这将有助于你对MySQL存储引擎的进一步理解，养成阅读源码或者文档的能力。

顺带的提一下MySQL的妹妹MariaDB。在MySQL的复刻版本MariaDB中10.2之前使用的自带的新引擎Aria，在MariaDB10.2之后使用的默认存储引擎也是InnoDB，足以看出InnoDB存储引擎的优秀之处。MariaDB的API和协议兼容MySQL，另外又添加了一些功能，以支持本地的非阻塞操作和进度报告。这意味着，所有使用MySQL的连接器、程序库和应用程序也将可以在MariaDB下工作。在此基础上，由于担心甲骨文MySQL的一个更加封闭的软件项目，Fedora等Linux发行版已经在最新版本中以MariaDB取代MySQL，维基媒体基金会的服务器同样也使用MariaDB取代了MySQL。

#### 主要需要了解的几种存储引擎：

- MyISAM
- InnoDB
- MEMORY
- MERGE

下面将着重介绍我最近看书认识的几种常用的存储引擎，对比各个存储引擎之间的区别，帮助我们理解不同存储引擎的使用方式。更多详情可以参考MySQL的官方文档。

## 2、部分存储引擎的特性

存储引擎/支持特性	存储限制	事务安全	锁机制	B树索引	哈希索引	全文索引	集群索引	数据缓存	索引缓存	数据可压缩	空间使用	内存使用	批量插入速度	外键支持
MyISAM	有		表锁	支持		支持			支持	支持	低	低	高	
InnoDB	64TB	支持	行锁	支持		支持 (5.6)	支持	支持	支持		高	高	低	支持
MEMORY	有		表锁	支持	支持			支持	支持		N/A	中等	高	
MERGE	没有		表锁	支持					支持		低	低	高	
NDB	有		行锁	支持				支持	支持		低	高	高	

InnoDB存储引擎在MySQL5.6版本开始支持全文索引。在MySQL5.7推出了虚拟列，MySQL8.0新特性加入了函数索引支持。

## 2.1、MyISAM存储引擎

MyISAM是MySQL5.5之前默认的存储引擎。MyISAM不支持事务、不支持外键。优势在于访问速度快，对事务完整性没有特殊要求或者以select和insert为主的应用基本上可以使用MyISAM作为存储引擎创建表。我们先弄个例子出来演示，事先准备了一张数据千万级别的表，看看这个存储引擎的特性：

我已经创建好了数据库为test，在test中分别创建了两张表test和tolove。test表在创建的时候指定默认存储引擎为MyISAM，tolove表指定存储引擎为InnoDB。

使用MyISAM存储引擎创建的表tolove，查询存储有1kw数据的表tolove。

**tips：**你可以使用 `use test`，切换到test数据库，就不用像我这样查询tolove表去指定test数据库了哟！

```
MySQL [(none)]> select count(*) from test.tolove;
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (0.000 sec)
```

再看演示使用InnoDB存储引擎创建的表test，同样为了演示，事先随机生成了1kw条数据。




```
MySQL [(none)]> select count(*) from test.test;
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (3.080 sec)
```

进行对比同样存储1kw条数据的表，使用MyISAM作为存储引擎查询速度堪称光速1 row in set (0.000 sec)，使用InnoDB存储引擎查询速度稍逊一筹1 row in set (3.080 sec)。

MyISAM在磁盘中的存储的文件：

每个MyISAM在磁盘上存储成3个文件，其文件名和表名都相同，扩展名分别是：

- .frm：存储表定义；
- .MYD：MYData，存储数据；
- .MYI：MYindex，存储索引。

 tolove.frm	2022-10-10 10:10:10	FRM 文件	2 KB
 tolove.MYD	2022-10-10 10:10:10	MYD 文件	195,313 KB
 tolove.MYI	2022-10-10 10:10:10	MYI 文件	100,375 KB

数据文件和索引文件可以存放在不同的目录，平均分布IO，获得更快的速度，提升性能。需要指定索引文件和数据文件存储的路径，创建表时通过DATA DIRECTORY和INDEX DIRECTORY参数指定，表明不同MyISAM表的索引文件和数据文件可以存放在不同的路径下。当然，需要给予该路径的访问权限。

MyISAM损坏处理方式：

MyISAM类型的表可能会损坏，原因多种多样。损坏后的表有可能不能被访问，会提示需要修复或者访问后提示返回错误结果。MyISAM类型的表，可以通过提供的修复工具**执行CHECK TABLE语句检查MyISAM表的健康程度**，使用**REPAIR TABLE语句修复**一个损坏的表。**表损坏可能会导致数据库异常重新启动**，需要尽快修复并确定原因好做应对策略。

Table	Op	Msg_type	Msg_text
test.tolove	check	status	OK 2B

## 执行：CHECK TABLE TABLE\_NAME

使用MyISAM存储引擎的表支持3种不同的存储格式，如下：

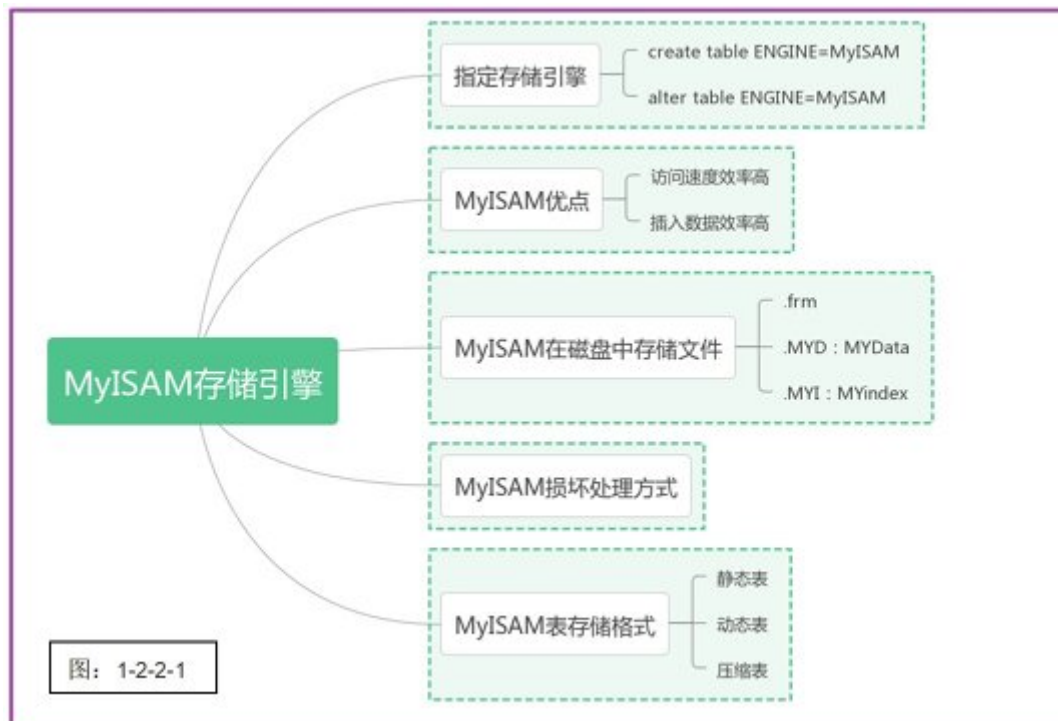
- 静态表，固定长度；
- 动态表
- 压缩表

**静态表**是MyISAM存储引擎的默认存储格式，字段长度是定长，记录都是固定长度。优势在于**存储迅速、容易缓存、出现故障易恢复**；缺点是相对耗存储空间。**需要注意的是：如需保存内容后面的空格，默认返回结果会去掉后面的空格。**

**动态表**包含变长字段，记录不是固定长度，存储优势：**占用空间相对较小、但频繁删除和更新记录会产生碎片**。这时，需要定期执行 `optimize table` 语句或者 `myisamchk -r` 命令来改善性能，出现故障恢复相对较难。

**压缩表**由**mysiampack工具**创建，占用磁盘空间很小。因为每个记录是被单独压缩，所以访问开始非常小。

梳理一下MyISAM存储引擎的要点，如下图1-2-2-1所示：



图：1-2-2-1

顺带安利一波，前段时间发现WPS也能够制作精美的思维导图，并且支持一键导入到doc文件中。普通用户最多可存储150个文件。之前也用过XMind、processon、gitmind等等，现在使用WPS更方便了。



## 2.2、InnoDB存储引擎

**优点与缺点：**InnoDB存储引擎提供了具有**提交**（commit）、**回滚**（rollback）和**崩溃恢复能力的事务安全**。但对比MyISAM存储引擎，InnoDB写的处理效率相对差一些，并且会占用更多的磁盘空间保留数据和索引。下图是我存储了1kw条数据的表，并且使用的是InnoDB存储引擎。student01表同样使用了InnoDB存储引擎，存储数据为100w条。从下图可以看出**存储数据索引在.ibd文件中、表结构则存在.frm文件中**。

student01.frm	student01存储了100w数据1	FRM 文件	1 KB
student01.ibd	2021	IBD 文件	65,536 KB
test.frm	test存储了1kw条数据	FRM 文件	2 KB
test.ibd	2022	IBD 文件	843,776 KB

### 2.2.1、自动增长列

InnoDB表的自动增长列可以手工插入，但插入的值为空或者0，则实际插入的将是自动自动增长后的值。

本来想继续使用bols那张表作为演示的，想来想去还是正经一点。为了演示，我又新增了一张表为autoincre\_test，表示id设置为主键且自增长，存储引擎选择InnoDB。然后插入了3条数据进行演示。查询当前线程最后插入数据的记录使用值：

```
MySQL [test]> create table autoincre_test(id int not null auto_increment,name
varchar(16),primary key(id))engine=innodb;
Query OK, 0 rows affected (0.018 sec)

MySQL [test]> insert into autoincre_test values(1,'1'),(0,'2'),(null,'3');
Query OK, 3 rows affected (0.007 sec)
Records: 3 Duplicates: 0 Warnings: 0

MySQL [test]> select * from autoincre_test;
+----+-----+
| id | name |
+----+-----+
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
+----+-----+
3 rows in set (0.000 sec)

select last_insert_id();
MySQL [test]> select last_insert_id();
+-----+
| last_insert_id() |
+-----+
| 2 |
+-----+
1 row in set (0.000 sec)
```

**tips：**可以通过 `alter table table_name=n;` 语句强制设置自动增长列的初始值，默认从1开始，但该**强制的默认值是保留在内存中的**，如果使用该值之前数据库重新启动，强制默认值则会丢失，就需要重新设置，毕竟使用内存没有加载到磁盘中。

通过上面的演示，你会发现插入记录是0或者空时，实际插入的将是自动增长后的值。通过 `last_insert_id()` 函数可以查询当前线程最后插入数据的记录使用值。如果一次插入多条记录，则返回的是第一条记录使用的自动增长值，这里就不演示插入多条数据了。记住一点，可以使用 `last_insert_id()` 去查询id记录值。

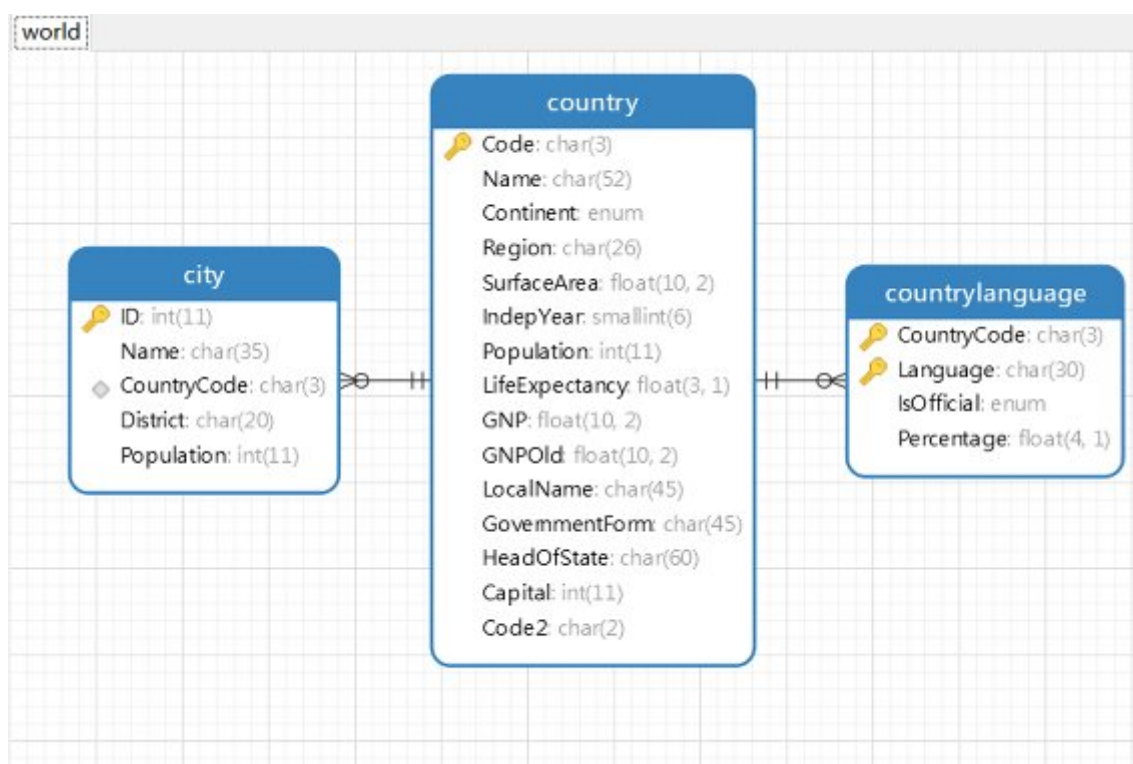
对于InnoDB表，自动增长列必须是索引。如果是组合索引，也必须是组合索引的第一列。但对于MyISAM表，自动增长列可以是组合索引的其它列。这样插入记录后，自动增长列是按照组合索引的前面几列排序后递增的。你可以创建一张表指定MyISAM存储引擎，然后将两列字段组合索引进行测试验证。

### 2.2.2、外键约束

在MySQL中，**目前支持外键约束的存储引擎只有InnoDB**。创建外键的时候，要求父表必须有对应的索引。子表创建外键的时候，也会自动创建对应的索引。下面将通过实例进行讲解。可以从MySQL官网下载示例数据库world和sakila进行参考。

- city表，FOREIGN KEY ( CountryCode ) REFERENCES country ( Code )
- country表
- countrylanguage表，FOREIGN KEY ( CountryCode ) REFERENCES country ( Code )

通过MySQL workbench或者Navicat逆向生成物理模型进行参考，更加直观。插一句，在MySQL的官网同样有一个sakila数据库是关于演员电影的，也[提供了sakila的ERR物理模型图](#)，这句话做了超链接，[可以直接访问](#)。给出我之前逆向生成的world数据库的物理模型：



在创建索引时，可以指定在删除、更新父表时，对子表进行的相应操作包含：

- restrict
- cascade
- set null和no action

其中restrict和no action相同，**restrict限制在子表有关联记录的情况下父表无法更新**；**cascade**表示在父表更新或删除的时候，**级联更新或者删除子表对应记录**；set null表示在父表更新或删除的时候，子表的对应字段被set null。**选择cascade以及set null时需要谨慎操作**，有可能导致数据丢失。

在导入多个表的数据时，如果忽略表之前的导入顺序，可以暂时关闭外键检查；同样**执行load data和alter table时也可以暂时关闭外键检查**加快处理的速度，提升效率。关闭外键检查的命令为：

```
set foreign_key_checks=0;
```

执行完导入数据或者修改表的操作后，通过开启外键检查命令改回来：

```
set foreign_key_checks=1;
```

对于InnoDB类型的表，外键信息可以通过 `show create table` 或者 `show table status` 查看。比如查找world数据库中的city表：

```
MySQL [sakila]> show table status like 'city'\G
```

关于外键约束就提这么多，没有演示创建以及删除，因为贴太多的SQL语句太占篇幅了。可以到MySQL官网下载world和sakila数据库进行测试。

### 2.2.3、存储方式

InnoDB存储表和索引有两种方式：

- 共享表空间存储
- 多表空间存储

使用**共享表空间存储**，这种方式创建的表的表结构保存在.frm文件中，数据和索引保存在innodb\_data\_home\_dir和innodb\_data\_file\_path定义的表空间中，可以是多个文件。在开头介绍InnoDB存储引擎时也提到过文件存储位置。

使用**多表空间存储**，这种方式创建的表的表结构仍然**保存在.frm文件中**，但每个表的数据和索引单独保存在.ibd文件中。如果是**分区表**，则每个分区对应单独的.ibd文件，**文件名为表名+分区名**。可以在创建分区的时候指定每个分区的数据文件位置，以此来平均分布磁盘的IO，达到缓解磁盘压力的目的。如下是在Windows下使用InnoDB存储了海量数据的文件：

student01.frm	student01存储了100w数据	2021	FRM 文件	1 KB
student01.ibd		2021	IBD 文件	65,536 KB
test.frm	test存储了1kw条数据	2022	FRM 文件	2 KB
test.ibd		2022	IBD 文件	843,776 KB

使用多表空间存储需要设置参数 `innodb_file_per_table`，重启数据库服务器才能生效哟。多表空间的参数生效后，只对新建的表生效。多表空间的数据文件无大小限制，无需设置初始大小，也不需设置文件的最大限制与扩展大小等参数。使用**多表空间存储优势在于方便单表备份和恢复操作**。虽然不能直接复制.frm和.ibd文件达到目的，但可以使用如下命令操作：

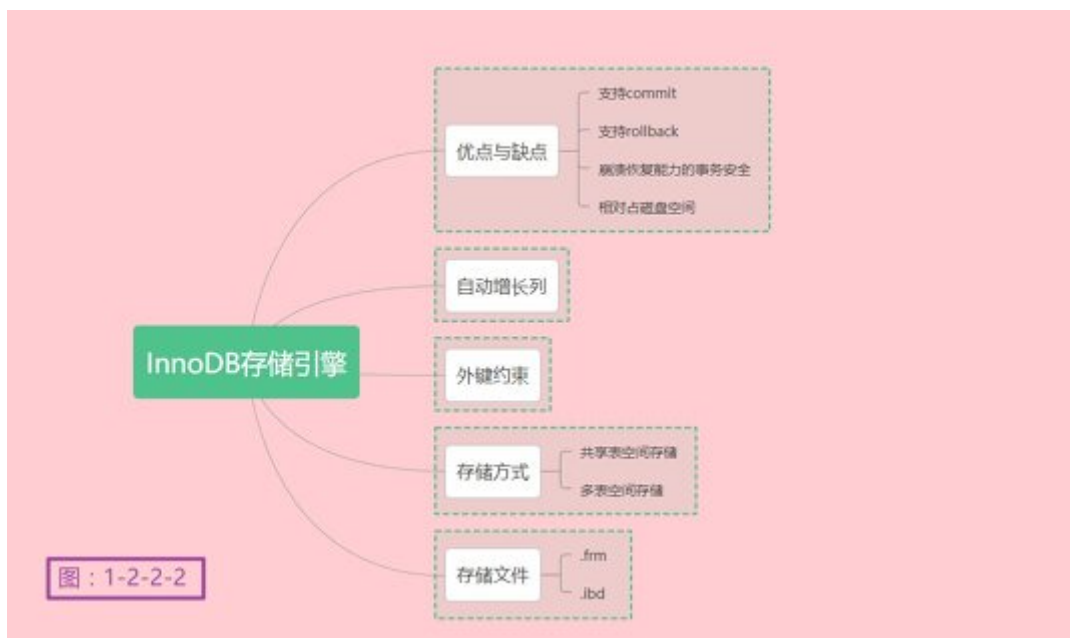
```
alter table table_name discard tablespace;  
alter table table_name import tablespace;
```

将备份恢复到数据库中，单表备份，只能恢复到原来所在的数据库中，无法恢复到其它数据库中。如过需要将单表恢复至其它目标数据库中，则需要通过mysqldump和mysqlimport来实现。

**注意：**即便多表存储更有优势，但是**共享表存储空间依旧是必须的**，InnoDB将**内部数据字典和在线重做日志**存在这个文件中。

梳理一下InnoDB存储引擎的要点，如下图1-2-2-2所示：





关于InnoDB存储引擎就介绍到此处了，更多详情可以参考MySQL的官方文档。是不是发现了我只在**MyISAM和InnoDB存储引擎**做了总结的思维导图。没错，只做了这两个，因为这俩最常用。至于为啥是粉色背景，因为老夫有一颗少女心！

## 2.3、MEMORY存储引擎

MEMORY存储引擎使用存在与内存中的内容来创建表。每个MEMORY表只对应一个磁盘文件，格式是.frm。MEMORY类型的表访问速度极快，存在内存中当然快。这就是Redis为什么这么快？不仅小？还能持久？咱回到正题，**MEMORY存在内存中并默认使用hash索引**，一旦服务关闭，表中数据会丢失。创建一张名为GIRLS的表指定存储引擎为MEMORY，**注意了**在UNIX和Linux操作系统下，**是对字段和表名大小是写敏感的，关键字不影响**。

```
CREATE TABLE GIRLS (  
  ID int NOT NULL,GIRE_NAME varchar(64) NOT NULL,GIRL_AGE varchar(10) NOT NULL,  
  CUP_SIZE varchar(2) NOT NULL,PRIMARY KEY (ID)  
) ENGINE=MEMORY DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

还记得在介绍存储引擎做的那会张表格吗，有介绍到MEMORY支持B TREE索引。虽然MEMORY默认使用的索引是hash索引，但是你可以手动指定索引类型。例如默认手动指定**使用关键字USING HASH**：

```
-- 创建索引指定索引类型为hash。  
create index mem_hash USING HASH on GIRLS(ID);  
-- 查询索引类型，简化了一下，只展示了部分参数。  
mysql> SHOW TABLE STATUS LIKE 'GIRLS'\G  
***** 1. row *****  
      Name: GIRLS  
      Engine: MEMORY  
    Version: 10  
   Row_format: Fixed  
 1 row in set (0.00 sec)
```

虽然MEMORY容易丢失数据，但是在启动MySQL服务的时候，我们可以使用--init-file选项，将**insert into ... select或者load data infile**这样的语句存放在这个指定的文件中，就可以在服务启动时从持久稳固的数据源装载表。

服务器需要提供足够的内存来维持所有在同一时间使用的MEMORY表，当不在需要MEMORY表内容之时，释放被MEMORY表使用的内存。仔细思考一下，如果内存用了不释放那将有多可怕。此时可以**执行 delete form 或truncate table亦或完整地删除整个表**，使用drop table。这里提一点，在Oracle中也同样支持truncate，使用truncate的好处在于不用再去考虑回滚（rollback），效率更高。使用truncate需要在命令模式下使用，其它客户端工具可能不支持。

每个MEMORY表中存放的数据量大小，受**max\_heap\_table\_size**系统变量约束，初始值为16MB，可以根据需求调整。通过**max\_rows**可以指定表的最大行数。

MEMORY存储引擎最大特色是快，主要用于内容变化不频繁的代码表，或者是为了做统计提供的中间表，效率更高。使用MEMORY时需谨慎，万一忘了这厮重启数据就没了就尴尬了。所以在使用时，考虑好重启服务器后如何取得数据。

关于MEMORY存储引擎就介绍到这里，大部分都是些理论知识，更多的需要自己去实践测试。

## 2.4、MERGE存储引擎

**MERGE存储引擎**是一组MyISAM表的组合，**这些MyISAM表必须结果完全相同**，MERGE表本身没有数据，对MERGE类型的表可以进行查询、更新、删除操作，**实际上是对内部的MyISAM表进行操作的**。对于MERGE类型表的插入操作，通过**insert\_method**子句定义插入的，可以有3个不同的值，使用first或last插入操作对应开始与最后一个表上。如果不定义这个子句，或者定义为NO，表示不能对MERGE表进行操作。

对MERGE表进行DROP操作，只是对MERGE的定义进行删除，对内部表没有任何影响。MERGE表上保留两个文件，文件名以表的名字开始，分别为：

- .frm文件存储表定义；
- .mrg文件包含组合表的信息，包含表组成、插入数据依据。

可以通过修改.mrg文件来修改表，但修改后需要使用**flush tables**刷新。测试可以先创建两张存储引擎为MyISAM的表，再建一张存储引擎为MERGE存储引擎的表。如下所示**创建demo为总表指定引擎为MERGE**，demo01和demo02为分表：

```
CREATE TABLE `merge_demo` (  
  `ID` INT(11) NOT NULL AUTO_INCREMENT, `NAME` VARCHAR(16) COLLATE utf8_bin NOT NULL,  
  PRIMARY KEY (`ID`)) ENGINE=MERGE UNION=(merge_demo01,merge_demo02)  
INSERT_METHOD=LAST DEFAULT CHARSET=utf8 COLLATE=utf8_bin  
  
CREATE TABLE `merge_demo01` (  
  `ID` INT(11) NOT NULL AUTO_INCREMENT, `NAME` VARCHAR(16) COLLATE utf8_bin NOT NULL,  
  PRIMARY KEY (`ID`)) ENGINE=MYISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin  
  
CREATE TABLE `merge_demo02` (  
  `ID` INT(11) NOT NULL AUTO_INCREMENT, `NAME` VARCHAR(16) COLLATE utf8_bin NOT NULL,  
  PRIMARY KEY (`ID`)) ENGINE=MYISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin
```

merge_demo.frm	2022-03-15 15:05:15	FRM 文件	1 KB
merge_demo.MRG	2022-03-15 15:05:15	MRG 文件	1 KB
merge_demo01.frm	2022-03-15 15:05:15	FRM 文件	1 KB
merge_demo01.MYD	2022-03-15 15:05:15	MYD 文件	0 KB
merge_demo01.MYI	2022-03-15 15:05:15	MYI 文件	1 KB
merge_demo02.frm	2022-03-15 15:05:15	FRM 文件	1 KB
merge_demo02.MYD	2022-03-15 15:05:15	MYD 文件	0 KB
merge_demo02.MYI	2022-03-15 15:05:15	MYI 文件	1 KB

通过插入数据验证MERGE确实是一个MyISAM的组合，就是这么神奇。如下所示，只对demo01和demo02进行插入：

```
INSERT INTO study.`merge_demo01` VALUES(1,'demo01');
INSERT INTO study.`merge_demo02` VALUES(1,'demo02');
mysql [study]> select * from merge_demo;
+----+-----+
| ID | NAME |
+----+-----+
|  1 | demo01 |
|  1 | demo02 |
+----+-----+
2 rows in set (0.000 sec)
```

插入完数据，分别查看demo01和demo02各只有一条数据，总表可以看到俩分表的全部数据。关键是指定了insert\_method=last。MERGE表和分区表的区别，MERGE并不能智能地将记录插入到对应表中，而分区表可以做到。通常我们使用MERGE表来透明的对多个表进行查询和更新操作。可以自己在下面测试总表插入数据，看分表的情况，我这里就不贴代码了。

关于MySQL自带的几款常用存储引擎就介绍到此，感兴趣的可以私下测试验证，更多参考请到官网获取API或者DOC文档。

除了MySQL自带的一些存储引擎之外，常见优秀的第三方存储引擎有TokuDB，一款开源的高性能存储引擎，适用于MySQL和MariaDB。更多详情可以去[TokuDB官网](#)了解哟。

## 2.5、修改表的存储引擎

创建新表时，如果不指定存储引擎，系统会使用默认存储引擎。在MySQL5.5之前默认的存储引擎为MyISAM，在MySQL5.5之后默认的存储引擎为InnoDB。如果想修改默认存储引擎，可以通过配置文件指定 default-table-type 的参数。关于存储引擎的查看，在上面介绍存储引擎的时候已经有说明了。

### 方法一：建表即指定当前表的存储引擎

在创建tolove表的时候就指定存储引擎，例如指定存储引擎为MyISAM，默认编码为utf8：

```
-- Create Table
CREATE TABLE `tolove` (
  `ID` int(11) NOT NULL AUTO_INCREMENT, `GIRL_NAME` varchar(64) COLLATE utf8_bin
  DEFAULT NULL,
  `GIRL_AGE` varchar(64) COLLATE utf8_bin DEFAULT NULL, `CUP_SIZE` varchar(10)
  COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM AUTO_INCREMENT=20000001 DEFAULT CHARSET=utf8 COLLATE=utf8_bin
```

测试生成的数据量比较大，随机生成了1千万条数据。查询（select）业务相对较多，在建表的时候就指定默认存储引擎MyISAM，统计（count）的效率很高。以我的渣渣电脑，使用INNODB存储引擎，统计一次需要2~3秒左右。在上面讲到MYISAM的时候，已经将查询时间进行过对比。

## 方法二：使用alter table修改当前表的存储引擎

修改创建的tolove表为MYISAM引擎进行测试。

```
-- 修改创建的tolove表为MYISAM引擎进行测试
ALTER TABLE test.`tolove` ENGINE=MYISAM;
```

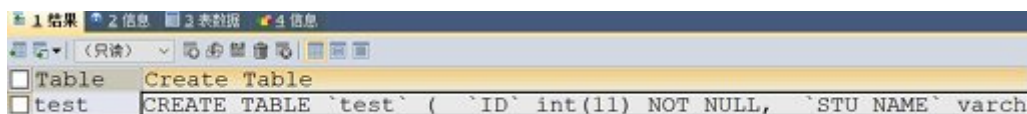
修改test表的存储引擎为INNODB进行测试。

```
-- 修改表的存储引擎为INNODB进行测试
ALTER TABLE test.`test` ENGINE=INNODB;
```

SHOW CREATE TABLE查询表的存储引擎，分别查询test表和tolove表，在讲存储引擎为MyISAM的时候，有演示过哟！

```
SHOW CREATE TABLE test.`test`;
SHOW CREATE TABLE test.`tolove`;
```

如果在工具中无法看全，可以导出成xml、csv、html等查询，以下是我查询出自己创建表时设置的存储引擎为InnoDB：



```
-- 显示出我创建的test表的SQL语句存储引擎为InnoDB
CREATE TABLE `test` ( `ID` int(11) NOT NULL AUTO_INCREMENT, `STU_NAME`
varchar(50) NOT NULL, `SCORE` int(11) NOT NULL, `CREATETIME` timestamp NOT NULL
DEFAULT current_timestamp() ON UPDATE current_timestamp(), PRIMARY KEY (`ID`) )
ENGINE=InnoDB AUTO_INCREMENT=20000001 DEFAULT CHARSET=utf8
-- 显示出我创建的tolove表的SQL语句，存储引擎为MyISAM
CREATE TABLE `tolove` ( `ID` int(11) NOT NULL AUTO_INCREMENT, `GIRL_NAME`
varchar(64) COLLATE utf8_bin DEFAULT NULL, `GIRL_AGE` varchar(64) COLLATE
utf8_bin DEFAULT NULL, `CUP_SIZE` varchar(10) COLLATE utf8_bin DEFAULT NULL,
PRIMARY KEY (`ID`) ) ENGINE=MyISAM AUTO_INCREMENT=20000001 DEFAULT CHARSET=utf8
COLLATE=utf8_bin
```

存储引擎的修改就介绍这么多，看到我的自增长列（AUTO\_INCREMENT）ID到了20000001，之前随机生成过一次1kw条数据哟！有一部分解释说明我写在了代码块中，看起来更加舒服。

## 3、存储引擎的选择

在选择合适的存储引擎时，应根据应用特点选择合适的存储引擎。对于复杂的应用系统，你可以选择多种存储引擎满足不同的应用场景需求。如何选择合适的存储引擎呢？存储引擎的选择真的很重要吗？

确实应该好好思考，在并不复杂的应用场景下，可能MyISAM存储引擎就能满足日常开销。或许在另外一种场景之下InnoDB才是最佳选择，综合性能更好，满足更多需求。

**MyISAM**是MySQL的默认的插件式存储引擎，是MySQL在5.5之前的默认存储引擎。如果应用以读和插入操作居多，只有很少的更新和删除操作，对事务完整性、并发性没有很高的需求，此时首选是MyISAM存储引擎。在web和数据仓库最常用的存储引擎之一。

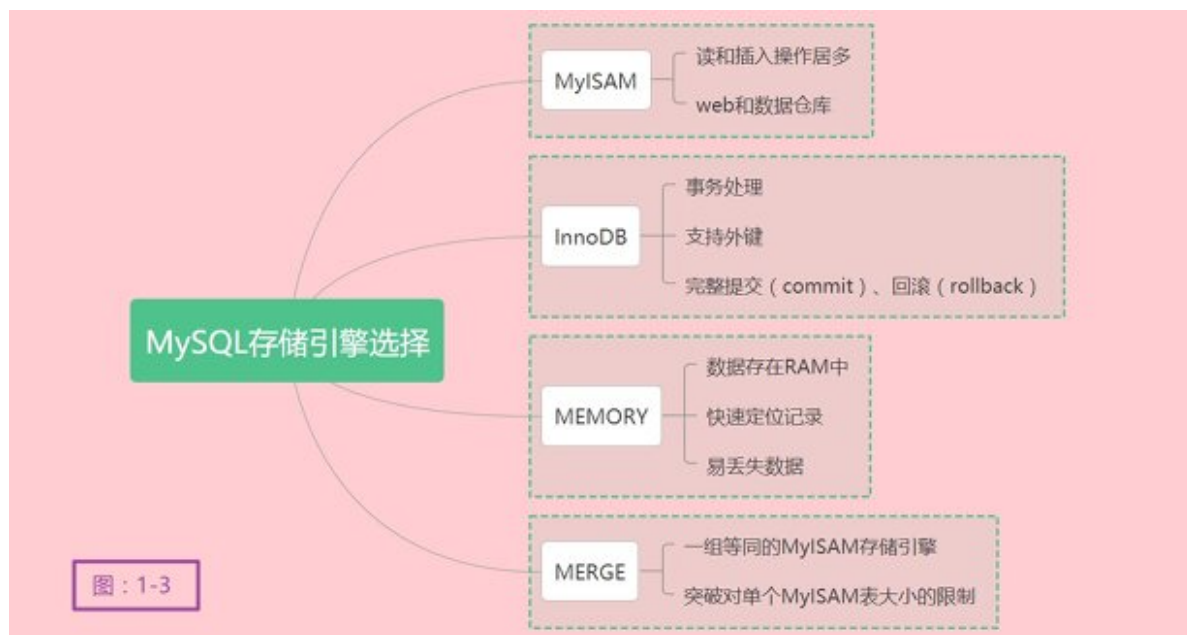
**InnoDB用于事务处理应用程序，并且支持外键。**是MySQL在5.5之后的默认存储引擎，同样也是MariaDB在10.2之后的默认存储引擎，足见InnoDB的优秀之处。如果应用对事务完整性有较高的要求，在并发情况下要求数据高度一致性。数据除了插入和查询以外，还包括很多的更新和删除操作，那么InnoDB应该会比较合适的存储引擎选择。InnoDB除了有效地降低由于删除和更新导致的锁定，还可以确保事务的完整提交（commit）、回滚（rollback）。对类似计费系统或者财务系统等对数据准确性要求比较高的系统，InnoDB也是合适的选择。插点题外话，本人在工作中使用Oracle数据库也有一段时间，Oracle的事务确实很强大，处理大数据压力很强。

**MEMORY**存储引擎将所有的数据存在RAM中，在需要快速定位记录和其它类似数据的环境下，可提供极快的访问。MEMORY的缺陷在于对表的大小有限制，太大的表无法缓存在内存中，其次是要确保表的数据可以恢复，数据库异常重启后表的数据是可恢复的。MEMORY表通常用于更新不太频繁的小表，快速定位访问结果。

**MERGE**用于将一组等同的MyISAM存储引擎的表以逻辑方式组合在一起，并作为一个对象应用它们。MERGE表的优点在于可以突破对单个MyISAM表大小的限制，并通过将不同的表分布在多个磁盘上，改善MERGE表的访问效率。对数据局仓库等VLDB环境很适合。

最后，关于存储引擎的选择都是根据别人实际经验去总结的。并不是一定契合你的应用场景，**最终需要用户对各自应用进行测试，通过测试来获取最合适的结果。**就像我开始列举的示例，数据量很庞大，对查询和插入业务比较频繁，我就开始对MyISAM存储引擎进行测试，确实比较符合我的应用场景。

关于存储引擎的选择，总结简化如下图1-3：



## 4、表的优化（碎片整理）

在开始介绍存MyISAM和InnoDB储引擎的时候，我也展示过存储大量数据所占的磁盘空间。使用**OPTIMIZE TABLE**来优化test数据库下的test表，**优化之前，这张表占据磁盘空间大概在824M**；通过优化之后，有明显的改善，系统回收了没有利用的空间，test表所耗磁盘空间明显下降，**优化之后只有456M**。这里就不贴磁盘所占空间的截图了。

```
OPTIMIZE TABLE test.`test`;
```

**优化之后**，统计（count）数据效率也有所提升，大概在2.5sec左右：



```
mysql [test]> select count(*) from test; -- 使用的是innodb存储引擎测试
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (2.468 sec)
```

**优化之前**，统计数据大概在3.080 sec。**经过对比，效率提升是可观的。**

你也可以使用**explain**执行计划对查询语句进行优化。关于MySQL优化方面的知识，并不是本文的重点，就不做过多描述。

## 二、索引设计与使用

### 1、索引简介

在涉及到MySQL的面试当中，会提到**最左前缀索引**，都被玩成梗了。

MySQL所有列类型都可以被索引，对相关列合理的使用索引是提高查询（select）操作性能的最佳方法。根据引擎可以定义每张表的最大索引数和最大索引长度，MySQL的每种存储引擎（MyISAM、InnoDB等等）对每张表至少支持16个索引，总索引长度至少为256字节。大多数存储引擎有更高的限制。

**MyISAM和InnoDB存储引擎默认创建的表都是BTREE索引。在MySQL8.0之前是不只支持函数索引的，MySQL5.7推出了虚拟列功能，在MySQL8.0开始支持函数索引，也是8.0版本的新特性之一。**

MySQL支持前缀索引，对索引字段的前N个字符创建索引，前缀索引长度和存储引擎有关。有很多人经常会问到，**MySQL支持全文索引吗**？我的回答是：支持。MySQL5.6之前MyISAM存储引擎支持全文索引（FULLTEXT），5.6之后InnoDB开始支持全文索引。

为test表创建10个字节的前缀索引，创建索引的语法如下：

```
CREATE INDEX girl_name ON table_name(test(10));
```

同样可以使用**alter table**语句去新增索引，给girl表的字段girl\_name新增一个索引：

```
ALTER TABLE test.`girl` ADD INDEX idx_girlname(girl_name);
```

对于使用索引的验证可以使用**explain**执行计划去判断。关于索引的简述就介绍这么多，更多基础知识可以参考官方文档或者权威书籍。

### 2、设计索引原则

索引的设计可以遵循一些已有的原则，创建索引的时候请尽量考虑符合这些原则。有助于提升索引的使用效率。

**搜索的索引列**，不一定是所要选择的列。**最合适的索引列，往往是出现在where子句中的列，或者是连接子句中指定的列**，而不是出现在select后选择列表中的列。

**使用唯一索引。**考虑某列中值的分布，索引列的基数越大，索引效果越好。

**使用短索引。**如果对字符串列进行索引，应指定一个前缀长度。比如char(100)，思考一下，重复度的问题。是全部索引来的快，还是对部分字符进行索引更优？

**利用最左前缀。**在创建一个N列的索引时，实际上是创建了MySQL可利用的N个索引。多列索引可以起几个索引的作用，**利用索引中最左边的列表来匹配行**。这样的列集称为最左前缀。都快被涉及到MySQL的面试玩成梗了，哈哈。

**注意不要过度使用索引。**不要以为使用索引好处多多，就在所有的列上全部使用索引，过度使用索引反而会适得其反。每个额外的索引会占用磁盘空间，对磁盘写操作性能造成损耗。在重构的时候，索引也得更新，造成不必要的时间浪费。

**InnoDB存储引擎的表。**对于使用InnoDB存储引擎的表，记录默认按一定的顺序保存。有如下几种情况：

- 如果有明确定义的主键，则遵循主键顺序保存；
- 在没有主键，但有唯一索引的情况下，会遵循唯一索引顺序保存；
- 既没有主键又没有唯一索引，表中会自动生成一个内部列，并遵循这个列的顺序保存。

以上就是对索引设计原则的简单介绍。

### 3、B-TREE与HASH索引

使用这些索引时，**应该考虑索引是否当前使用条件下生效**！在使用MEMORY存储引擎的表中可以选择使用HASH索引或者B-TREE索引，两种不同的索引有其各自适用的范围。

**HASH索引。**只用于这类关系操作符：`=`、`<=>`的操作比较，**优化器不能使用HASH索引来加速order by**操作。MySQL不能确定在两个值之间大约有多少行。

**B-TREE索引。**对于B-TREE索引，使用`>`、`<`、`>=`、`<=`、`BETWEEN`、`!=`或者`<>`、亦或是使用like 'condition'。其中'condition'不以通配符开始的操作符时，都可以使用相关列上的索引。

关于索引就介绍到这里。合理的使用索引将有助于提升效率，但并不是使用的索引越多越好。

## 三、数据类型选择

- 字符串类型char与varchar
- 浮点数和定点数
- 日期类型

工作中，个人使用经验。Oracle里面使用BLOB存储大字段比较频繁，TEXT相对少见，使用VARCHAR2类型比较多。但在MySQL中是不支持VARCHAR2类型的。

### 1、CHAR与VARCHAR

char和varchar类型类似，用于存储字符串，但它们保存和检索的方式不同。char类型属于固定长度（定长）类型的字符串，varchar属于可变长度的字符串类型。在MySQL的严格模式中，使用的char和varchar，超过列长度的值不会被保存，并且出现错误提示。

**char优缺点。**char是固定长度，处理速度比varchar要快，但缺点是浪费存储空间，没有varchar那么灵活。**varchar。**随着MySQL的不断升级，varchar类型也在不断优化，性能也在提升，被用于更多的应用中。

**MyISAM存储引擎：**建议使用固定长度的数据列代替可变长度的数据列。

**InnoDB存储引擎：**建议使用VARCHAR类型。

**MEMORY存储引擎：**使用固定长度数据类型存储。

## 2、TEXT与BLOB

一般情况，存储少量的字符串时，会选择char和varchar类型。而在保存较大文本时，通常选择TEXT或者BLOB大字段，二者之间的区别在于**BLOB能存二进制数据**，比如：照片，**TEXT类型只能存字符数据**。这也是为什么我在开始的时候提及到个人工作中见到BLOB类型相对较多。TEXT和BLOB还包括不同类型：

- TEXT、LONGTEXT、MEDIUMINT、MEDIUMTEXT、TINYTEXT；
- BLOB、LONGBLOB、MEDIUMBLOB、TINYBLOB。

**区别在于存储文本长度和字节不同。**

**需要注意的点：**

- BLOB和TEXT值会引起一些性能问题，尤其是执行大量删除操作时；
- 可以使用合成索引提高大字段的查询性能；
- 在不必要的时候避免检索大字段；
- 将BLOB和TEXT分离到不同的表中。

## 3、浮点数与定点数

浮点类型一般用于表示含有小数部分的值。列举一些示例：

- double类型：用于浮点数（双精度）；
- decimal类型：MySQL中表示定点数；
- float类型：用于浮点数（单精度）。

学过Java语言的同学，对这些浮点类型并不陌生吧。

**注意点：**浮点数存在误差问题，对精度比较敏感的数据，避免对浮点类型做比较。

## 4、日期类型

谈到日期类型，又让我想起了7年前学Java语言的时候，会写一个工具类（Utils.java），将常用的处理日期的方法写进去然后调用。经常用到的一个方法（SimpleDateFormat），对时间戳进行转换格式化。

**MySQL中常用的日期类型有：**

- DATE
- DATETIME
- TIME
- TIMESTAMP

如果需要记录年月日时分秒，并且记录的年份比较久远，最好用DATETIME，而不要使用TIMESTAMP时间戳。**TIMESTAMP表示的范围比DATETIME短得多。**

## 四、字符集（字符编码）设置

从本质上来说，计算机只能是被二进制代码（010101）。因此，不论是计算机程序还是处理的数据，最终都会转换成二进制代码，计算机才能识别。为了让计算机不仅能做科学计算，也能处理文字信息，于是计算机字符集诞生了。

**字符编码**（英语：Character encoding）、**字集码**是把**字符集**中的字符编码为指定集合中某一对象）（例如：比特模式、自然数序列、8位组或者电脉冲），以便文本在计算机中存储和通过通信网络的传递。常见的例子包括将拉丁字母表编码成摩斯电码和ASCII。其中，ASCII将字母、数字和其它符号编号，并用7比特的二进制来表示这个整数。通常会额外使用一个扩充的比特，以便于以1个字节的方式存储。

在计算机技术发展的早期，如ASCII（1963年）和EBCDIC（1964年）这样的字符集逐渐成为标准。但这些字符集的局限很快就变得明显，于是人们开发了许多方法来扩展它们。对于支持包括东亚CJK字符家族在内的写作系统的要求能支持更大量的字符，并且需要一种系统而不是临时的方法实现这些字符的编码。

引用自维基百科对字符编码的介绍。

## 1、Unicode

Unicode是什么？是统一编码，是计算机科学领域的业界标准。从最初的1.0.0到目前最新的14.0版本，对应ISO/IEC 10646-N:xxxx。说一下UTF-8、UTF-16、UTF-16LE、UTF-32BE、UTF-32LE等等大家应该很熟悉了。

## 2、常见字符集

常见的字符集：

- UTF-8：泛用性最广泛；
- GBK：对中文支持非常友好，在GB2312基础上进行了扩充；
- GB2312：对中文字符集支持，；
- GB18030：支持中文字符集，解决GBK强制力不够的问题。

## 3、MySQL支持的字符集

通过 `show character set;` 命令可以查看MySQL支持的字符集。我只展示部分：

```
mysql [test]> show character set;
```

gbk	GBK Simplified Chinese	gbk_chinese_ci	2
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
utf8	UTF-8 Unicode	utf8_general_ci	3
utf8mb4	UTF-8 Unicode	utf8mb4_general_ci	4
utf16	UTF-16 Unicode	utf16_general_ci	4
utf32	UTF-32 Unicode	utf32_general_ci	4

或者你还可以使用 `DESC information_schema.CHARACTER_SETS` 查看所有字符集和字符集默认的校对规则。

查看相关字符集校对规则，可以使用**SHOW COLLATION**配合 LIKE模糊搜索gbk字符集。

```
SHOW COLLATION LIKE 'gbk%';
```

**MySQL字符集设置**：默认可以过配置文件设置**character-set-server**参数。

- Linux发行版中安装一般在my.cnf中配置；
- Windows下在my.ini文件中配置

```
[mysqld]
character-set-server=utf-8
character-set-server=gbk
```

额外再提一点，判断字符集所占字节，可以使用函数**LENGTH()**：

```
SELECT LENGTH('中');
```

如果使用的是UTF-8编码，默认汉字是占用3个字节，使用GBK则占用2个字节。字符编码就介绍到这里。

## 五、MySQL示例数据库sakila

视图、存储过程、函数、触发器。这里给出我自己随机生成海量数据用到的函数和存储过程。

### 1、函数

**创建函数**，使用DELIMITER声明，使用**CREATE FUNCTION创建函数**，tolove表的创建在介绍存储引擎过程中已经有展示过。

```
/** 创建函数 生成学号 **/  
DELIMITER $  
CREATE FUNCTION rand_number() RETURNS INT  
BEGIN  
    DECLARE i INT DEFAULT 0;  
    SET i= FLOOR(1+RAND()*100);  
    RETURN i;  
END $  
DELIMITER $
```

创建函数：用于生成姓名随机字符串

```
/** 创建函数 生成姓名随机字符串 **/  
DELIMITER $  
CREATE FUNCTION rand_name(n INT) RETURNS VARCHAR(255)  
BEGIN  
    DECLARE chars_str VARCHAR(100) DEFAULT  
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';  
    DECLARE return_str VARCHAR(255) DEFAULT '';  
    DECLARE i INT DEFAULT 0;  
    WHILE i < n DO  
        SET return_str =  
CONCAT(return_str,SUBSTRING(chars_str,FLOOR(1+RAND()*52),1));  
        SET i = i+1;  
    END WHILE;  
    RETURN return_str;  
END $  
DELIMITER $
```

### 2、存储过程

**创建存储过程**，使用**CREATE PROCEDURE**创建：

```
/** 创建存储过程 **/  
DELIMITER $  
CREATE PROCEDURE insert_tolove(IN max_num INT(10))  
BEGIN  
    DECLARE i INT DEFAULT 0;  
    DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;  
    START TRANSACTION;  
    WHILE i< max_num DO
```



```

INSERT INTO test.`tolove`(ID,GIRL_NAME,GIRL_AGE,CUP_SIZE)
VALUES(NULL,rand_name(5),rand_number(),NULL);
SET i = i + 1;
END WHILE;
COMMIT;
END $
DELIMITER $

```

使用**CALL**调用存储过程，随机生成百万数据：

```

/** 调用存储过程 */
CALL insert_tolove(100*10000);

```

删除函数或者存储过程，使用**DROP**关键字

```

-- 删除函数rand_name
DROP FUNCTION rand_name;
-- 删除存储过程insert_tolove
DROP PROCEDURE insert_tolove;

```

### 3、触发器

创建触发器使用**CREATE TRIGGER**，这里就引用sakila数据库实例。如果存在，使用了判断语句 **IF EXISTS**，然后删除**DROP TRIGGER**已经存在的触发器。

```

DELIMITER $$
USE `sakila`$$
DROP TRIGGER /*!50032 IF EXISTS */ `customer_create_date`$$
CREATE
    /*!50017 DEFINER = 'root'@'%' */
    TRIGGER `customer_create_date` BEFORE INSERT ON `customer`
    FOR EACH ROW SET NEW.create_date = NOW();
$$
DELIMITER ;

```

### 4、sakila数据库

在文中我反复提到了MySQL的示例数据库sakila，是一个完整的学习MySQL的好例子。包含了视图、存储过程、函数和触发器。可以去MySQL的官网获取SQL脚本。



文末留一个神秘的参数，通过此种方式可以过滤你不想看到的内容哟！无论在手机端或者PC端都可生效，亲测可用。

```
xxxx(检索的内容) -site:xxn.net  
-- 或者  
xxxx(检索的内容) -site:xxshu.com
```

持续更新优化中...

## 总结

以上就是此次文章的所有内容的，希望能对你的工作有所帮助。感觉写的好，就拿出你的一键三连。在公众号上更新的可能要快一点，目前还在完善中。**能看到这里的，都是帅哥靓妹**。如果感觉总结的不到位，也希望能留下您宝贵的意见，我会在文章中进行调整优化。



原创不易，转载也请标明出处和作者，尊重原创。不定期上传到github或者gitee。认准龙腾万里sky，如果看见其它平台不是这个ID发出我的文章，就是转载的。**linux系列文章：《初学者如何入门linux，原来linux还可以这样学》**已经上传至github和gitee。个人github仓库地址，一般会先更新PDF文件，然后再上传markdown文件。如果访问github太慢，可以使用gitee进行克隆。

**tips**：使用hexo搭建的静态博客也会定期更新维护。

<https://github.com/cnwangk/SQL-study>

**作者**：[龙腾万里sky](#)