**MySQL索引/事务与存储引擎**

# 数据库索引概述

## 一、数据库索引

在数据库中，索引使数据库程序无须对整个表进行扫描，就可以在其中找到所有数据。数据库的索引是某个表中一列或者若干列值的集合，以及物理标识这些值的数据页的**逻辑指针清单**。

## 二、索引的作用

（1）设置了合适的索引之后，数据库利用各种快速的定位技术，能够**加快查询速率**，特别是当表很大时，或者查询涉及到多个表时，使用索引可使查询加快成千倍。

（2）可以**降低**数据库的**IO成本**，并且索引还可以降低数据库的排序成本。

（3）通过创建唯一性索引保证数据表数据的唯一性，可以**加快表与表之间的连接**。

（4）在使用分组和排序时，可大大**减少分组和排序时间**。

## 三、索引的分类

（1）普通索引

这是最基本的索引类型，而且没有唯一性之类的限制

（2）唯一性索引

与普通索引基本相同，区别在于：索引列的所有值都只能出现一次，即必须唯一，但可为空。

（3）主键

是一种特殊的唯一索引，必须指定为“PRIMARY KEY”，具有唯一性的同时不能为空。

（4）全文索引

MySQL从3.23.23版开始支持全文索引和全文检索。在MySQL中，全文索引的类型为FULLTEXT，全文索引可以在VARCHAR或者TEXT类型的列上创建。贴吧的文本内容，和一些小型的网站网页内容，存放在数据库中即为全文索引模式。

（5）单列索引与多列索引

索引可以是单列上创建的索引，也可以是多列上创建的索引。

## 四、创建索引的原则依据

1、表的主键、外键必须有索引

2、数量超过300行的表应该有索引

3、经常与其他表进行连接的表，在连接字段上应该建立索引

4、唯一性太差的字段不适合建立索引

5、更新太频繁的字段不适合创建索引

6、经常出现在where字句中的字段，特别是大表的字段，应该建立索引

7、索引应该建在选择性高的字段上

8、索引应该建立在小字段上，对于大的文本字段甚至超长字段，不要建立索引

## 五、创建、查看、删除索引的方法

根据企业需求选择了合适的索引之后，可以使用CREATE INDEX创建索引，CREATE INDEX加上各个索引关键字便可创建各个类型的索引。

### 案例：建立公司IT运营部工资数据表

**需求描述：为公司建立员工工资数据库imployee\_salary，在imployee\_salary数据库中，建立IT\_salary数据表，以保存IT运营部员工的工资信息，如下表所示：**

****

CentOS7里面MySQL变成MariaDB：

**[root@localhost ~]# yum -y install mariadb mariadb-devel mariadb-server**

**[root@localhost ~]# vim /etc/my.cnf**

**[client]**

**default-character-set=utf8**

**此配置使客户端可以正确识别中文**

**[root@localhost ~]# systemctl restart mariadb**

**[root@localhost ~]# netstat -lnpt | grep :3306**

**tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 67322/mysqld**

**建立数据库imployee\_salary**

**[root@localhost ~]# mysql**

**MariaDB> create database imployee\_salary default charset utf8;**

**Query OK, 1 row affected (0.00 sec)**

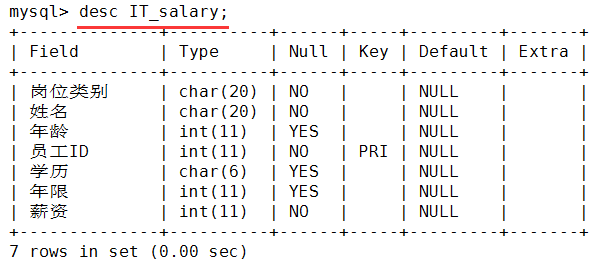
**MariaDB> use imployee\_salary;**

**Database changed**

**建立数据表IT\_salary**

**create table IT\_salary(岗位类别 char(20) not null,姓名 char(20) not null, 年龄 int, 员工ID int not null,学历 char(6),年限 int,薪资 int not null, primary key(员工ID));**

**//int数字类型、char 字符串类型、not null 不能为空、char() 指定最多字节个数、primary key() 指定索引字段**



**将IT运营部的员工工资信息插入到IT\_salary**

**insert into IT\_salary(岗位类别,姓名,年龄,员工ID,学历,年限,薪资) values('网络工程师','孙空武',27,011,'本科',3,4800);**

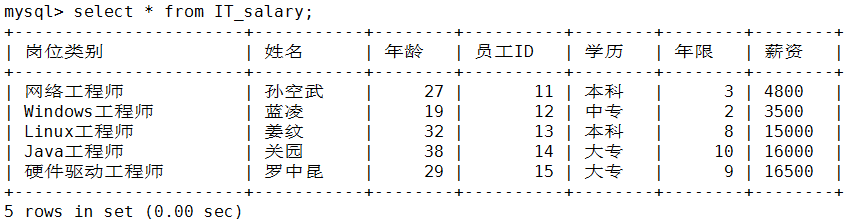
**insert into IT\_salary(岗位类别,姓名,年龄,员工ID,学历,年限,薪资) values('Windows工程师','蓝凌',19,012,'中专',2,3500);**

**insert into IT\_salary(岗位类别,姓名,年龄,员工ID,学历,年限,薪资) values('Linux工程师','姜纹',32,013,'本科',8,15000);**

**insert into IT\_salary(岗位类别,姓名,年龄,员工ID,学历,年限,薪资) values('Java软件工程师','关园',38,014,'大专',10,16000);**

**insert into IT\_salary(岗位类别,姓名,年龄,员工ID,学历,年限,薪资) values('硬件驱动工程师','罗中昆',29,015,'大专',9,16500);**

**查看核对信息**



1、创建普通索引

格式：CREATE INDEX <索引的名字> ON 表名(字段);

**MariaDB>create index salary\_index on IT\_salary(薪资);**

**Query OK, 0 rows affected (0.24 sec)**

**Records: 0 Duplicates: 0 Warnings: 0**

**MariaDB>show index from IT\_salary \G; //或show keys from IT\_salary \G;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: IT\_salary**

**Non\_unique: 0 #是否唯一索引，0代表是唯一索引**

**Key\_name: PRIMARY #员工ID字段是自动创建的主键索引**

**Seq\_in\_index: 1 #字段在索引中的序号，单字段索引此值都是1**

**Column\_name: 员工ID #索引字段**

**Collation: A #排序方法，A是不区分大小写的排序，查询时大小写的都显示**

**Cardinality: 2**

**Sub\_part: NULL**

**Packed: NULL**

**Null:**

**Index\_type: BTREE**

**Comment:**

**Index\_comment:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: IT\_salary**

**Non\_unique: 1**

**Key\_name: salary\_index**

**Seq\_in\_index: 1**

**Column\_name: 薪资**

**Collation: A**

**Cardinality: 5**

**Sub\_part: NULL**

**Packed: NULL**

**Null:**

**Index\_type: BTREE**

**Comment:**

**Index\_comment:**

**2 rows in set (0.00 sec)**

2、创建唯一性索引

格式：CREATE UNIQUE INDEX <索引的名字> ON 表名(字段);

**MariaDB> create unique index salary\_unique\_index on IT\_salary(姓名);**

**Query OK, 0 rows affected (0.06 sec)**

**Records: 0 Duplicates: 0 Warnings: 0**

**MariaDB>show keys from IT\_salary \G;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: IT\_salary**

**Non\_unique: 0**

**Key\_name: PRIMARY**

**Seq\_in\_index: 1**

**Column\_name: 员工ID**

**Collation: A**

**Cardinality: 5**

**Sub\_part: NULL**

**Packed: NULL**

**Null:**

**Index\_type: BTREE**

**Comment:**

**Index\_comment:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: IT\_salary**

**Non\_unique: 0**

**Key\_name: salary\_unique\_index**

**Seq\_in\_index: 1**

**Column\_name: 姓名**

**Collation: A**

**Cardinality: 5**

**Sub\_part: NULL**

**Packed: NULL**

**Null:**

**Index\_type: BTREE**

**Comment:**

**Index\_comment:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: IT\_salary**

**Non\_unique: 1**

**Key\_name: salary\_index**

**Seq\_in\_index: 1**

**Column\_name: 薪资**

**Collation: A**

**Cardinality: 5**

**Sub\_part: NULL**

**Packed: NULL**

**Null:**

**Index\_type: BTREE**

**Comment:**

**Index\_comment:**

**3 rows in set (0.00 sec)**

3、创建主键索引

格式：CREATE TABLE 表名([…],PRIMARY KEY(字段));

ALTER TABLE 表名 ADD PRIMARY KEY(字段);

示例：

**MariaDB> create table IT\_salary(岗位类别 char(20) not null,姓名 char(30) not null,年龄int(3),员工ID int not null,学历 char(6),年限int(3),薪资 char(10) not null,primary key(员工ID));**

若在新建表时忘记创建主键，可以进行如下操作：

**MariaDB>alter table IT\_salary add primary key(员工ID);**

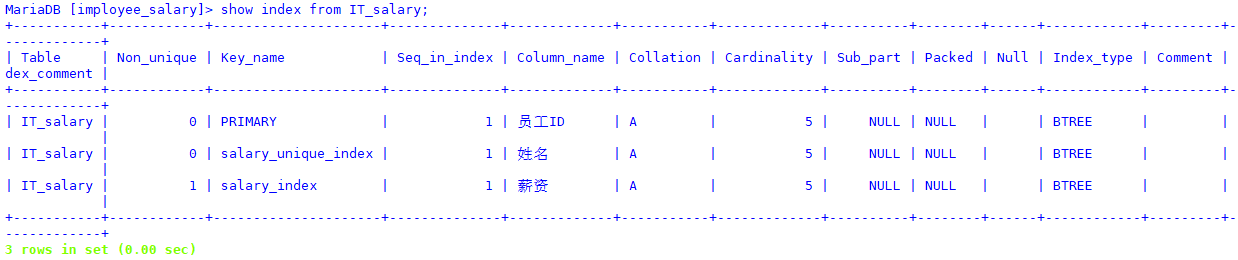
**Query OK, 0 rows affected (0.07 sec)**

**Records: 0 Duplicates: 0 Warnings: 0**

4、查看索引

格式：SHOW INDEX FROM 表名;

SHOW KEYS FROM 表名;



5、删除索引

格式：DROP INDEX <索引的名称> ON 表名;

ALTER TABLE 表名DROP INDEX <索引的名称>;

ALTER TABLE 表名DROP PRIMARY KEY;

注：前两条命令作用相同，由于每个表只允许有一个PRIMARY KEY，所以不用指定名称。

**MariaDB>drop index salary\_index on IT\_salary;**

**Query OK, 0 rows affected (0.10 sec)**

**Records: 0 Duplicates: 0 Warnings: 0**

**MariaDB>alter table IT\_salary drop index salary\_unique\_index;**

**Query OK, 0 rows affected (0.01 sec)**

**Records: 0 Duplicates: 0 Warnings: 0**

**MariaDB> alter table IT\_salary drop primary key;**

**Query OK, 5 rows affected (0.07 sec)**

**Records: 5 Duplicates: 0 Warnings: 0**

**MariaDB> show keys from IT\_salary;**

**Empty set (0.00 sec)**

# 数据库事务的概念

事务是一种机制、一个操作序列，包含了一组数据库操作命令，并且把所有的命令作为一个整体一起向系统提交或撤销操作请求，即这一组数据库命令要么都执行，要么不都执行。

事务是一个不可分割的工作逻辑单元，在数据库系统上执行并发操作时，事务是最小的控制单元。

事务适用于多用户同时操作的数据库系统的场景，如银行、保险公司及证券交易系统等等。通过事务的完整性以保证数据的一致性。

## 一、事务的ACID特点

事务具有四个属性：ACID

* 原子性（Atomicity）
* 一致性（Consistency）
* 隔离性（Isolation）
* 持久性（Durability）

### 1、原子性

事务是一个完整的操作，事务的各元素是不可分的（原子的），事务的所有元素必须作为一个整体提交或回滚。如果事务中的任何元素失败，则整个事务将失败。

### 2、一致性

当事务完成时，数据必须处于一致状态：在事务开始之前，数据库汇总存储的数据处于一致状态；在正在进行的事务中，数据可能处于不一致的状态；当事务完成时，数据必须再次回到已知的一致状态。

### 3、隔离性

对数据进行修改的所有并发事务是彼此隔离的，这表明事务必须是独立的，它不应该以任何方式依赖于或影响其他事务。修改数据的事务可以在另一个使用相同数据的事务开始之前访问这些数据，或者在另一个使用相同数据的事务结束之后访问这些数据。

### 4、持久性

事务的持久性指不管系统是否发生了故障，事务处理的结果都是永久的。一旦事务被提交，事务的效果会被永久地保留在数据库中。

## 二、事务的操作

默认情况下MySQL的事务是自动提交的，当sql语句提交时事务便自动提交。

手动对事务进行控制的方法：

* 事务处理命令控制
* 使用set设置事务处理方式

事务处理命令控制事务：

begin 开始一个事务

commit 提交一个事务

rollback 回滚一个事务（撤销）

MySQL事务的操作必须基于Innodb存储引擎

commit示例：

**MariaDB> create database auth;**

**Query OK, 1 row affected (0.00 sec)**

**MariaDB> use auth;**

**Database changed**

**MariaDB> create table users(user\_name char(18) not null,user\_passwd char(50) default '', primary key (user\_name)) engine=innodb;**

**Query OK, 0 rows affected (0.46 sec)**

**MariaDB>begin; //事务开始**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB>insert into users values('lisi',password('123123'));**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB>** **insert into users values('wangwu',password('654321'));**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB> insert into users values('zhangsan',password('123456'));**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB>commit; //事务结束并提交**

**Query OK, 0 rows affected (0.02 sec)**

**MariaDB>select \* from users;**

**+-----------+------------------------------------------+**

**| user\_name | user\_passwd |**

**+-----------+------------------------------------------+**

**| lisi | \*E56A114692FE0DE073F9A1DD68A00EEB9703F3F |**

**| wangwu | \*2A032F7C5BA932872F0F045E0CF6B53CF702F2C |**

**| zhangsan | \*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD |**

**+-----------+------------------------------------------+**

**3 rows in set (0.00 sec)**

rollback实例：

**MariaDB> use auth;**

**Database changed**

**MariaDB>begin; //事务开始**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB> insert into users values('zhaoliu',password('123123'));**

**Query OK, 1 row affected (0.00 sec)**

**Rows matched: 1 Changed: 1 Warnings: 0**

**MariaDB>rollback; //回滚（撤销操作），从begin开始的所有命令都将被撤销**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB> select \* from users;**

**+-----------+------------------------------------------+**

**| user\_name | user\_passwd |**

**+-----------+------------------------------------------+**

**| lisi | \*E56A114692FE0DE073F9A1DD68A00EEB9703F3F |**

**| wangwu | \*2A032F7C5BA932872F0F045E0CF6B53CF702F2C |**

**| zhangsan | \*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD |**

**+-----------+------------------------------------------+**

**3 rows in set (0.00 sec)**

2、使用set命令进行控制

set autocommit=0 禁止自动提交，默认值，提交前的所有命令是一个事务

set autocommit=1 开启自动提交，每一条命令都是一条单独的事务并自动提交

**MariaDB>set autocommit = 0; //禁止自动提交**

**Query OK, 0 rows affected (0.02 sec)**

**MariaDB> use auth;**

**Database changed**

**MariaDB>** **insert into users values('zhaoliu',password('123123'));**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB> insert into users values('lilili',password('111111'));**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB>rollback; //提交前面两条SQL语句，为一个事务**

**Query OK, 2 rows affected (0.04 sec)**

**MariaDB>select \* from users;**

**+-----------+------------------------------------------+**

**| user\_name | user\_passwd |**

**+-----------+------------------------------------------+**

**| lisi | \*E56A114692FE0DE073F9A1DD68A00EEB9703F3F |**

**| wangwu | \*2A032F7C5BA932872F0F045E0CF6B53CF702F2C |**

**| zhangsan | \*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD |**

**+-----------+------------------------------------------+**

**5 rows in set (0.00 sec)**

**MariaDB>set autocommit = 1; //开启自动提交**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB> use auth;**

**Database changed**

**MariaDB> insert into users values('zhaoliu',password('123123')); //第一个事务**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB> insert into users values('lilili',password('111111')); //第二个事务**

**Query OK, 1 row affected, 1 warning (0.00 sec)**

**MariaDB> rollback;**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB>select \* from users;**

**+-----------+------------------------------------------+**

**| user\_name | user\_passwd |**

**+-----------+------------------------------------------+**

**| lilili | \*FD571203974BA9AFE270FE62151AE967ECA5E0A |**

**| lisi | \*E56A114692FE0DE073F9A1DD68A00EEB9703F3F |**

**| wangwu | \*2A032F7C5BA932872F0F045E0CF6B53CF702F2C |**

**| zhangsan | \*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD |**

**| zhaoliu | \*E56A114692FE0DE073F9A1DD68A00EEB9703F3F |**

**+-----------+------------------------------------------+**

**7 rows in set (0.00 sec)**

**已插入成功，回滚失败**

# 存储引擎概念

MySQL数据库中使用各种不同的技术存储数据到文件系统中，每一种技术都使用不同的存储机制、索引技巧，锁定水平并最终提供不同的功能和能力，这些不同的技术以及配套的功能在MySQL中称为存储引擎。

存储引擎就是MySQL将数据存储在文件系统中的存储方式或者存储格式

目前MySQL常用的两种存储引擎：MyISAM、InnoDB

MySQL存储引擎是MySQL数据库服务器中的组件，负责为数据库执行实际的数据I/O操作，使用特殊存储引擎的主要优点之一在于仅需提供特殊应用所需的特性，数据库中的系统开销较小，具有更有效和更高的数据库性能。

MySQL系统中，存储引擎处于文件系统之上，在数据保存到数据文件之前会传输到存储引擎，之后按照各个存储引擎的存储格式进行文件的物理存储。

MySQL日志种类：

* 错误日志：程序运行，启动类的故障
* 二进制日志：保存数据库变更语句
* 中继日志：缓存二进制日志数据
* 查询日志：记录查询语句的（通用日志）
* 慢查询日志：记录超时的查询语句（调优）

## 一、MyISAM存储引擎

## 1、MyISAM概述

MyISAM存储引擎是MySQL关系数据库系统5.5版本之前默认的存储引擎，前身是ISAM。

ISAM是一个定义明确且经历时间考验的数据表格管理方法，在设计之时就考虑到数据库被查询的次数要远大于更新的次数。

ISAM的特点：ISAM执行读取操作的速度很快,而且占用不大量的内存和存储资源，它不支持事务处理，数据不能够容错。

MyISAM管理非事务表，是ISAM的扩展格式，提供ISAM里所没有的索引和字段管理的大量功能。

MyISAM使用一种表格锁定的机制，以优化多个并发的读写操作。MyISAM提供高速存储和检索，以及全文搜索能力，受到web开发的青睐。

## 2、MyISAM的特点

（1）不支持事务。

（2）表级锁定形式，数据在更新时锁定整个表。

（3）数据库在读写过程中相互阻塞（会在数据写入的过程中阻塞用户数据的读取，也会在数据读取的过程中阻塞用户的数据写入）。

（4）可以通过key\_buffer\_size来设置缓存索引，提高访问性能，减少磁盘IO的压力，但缓存只会缓存索引文件，不会缓存数据。

（5）采用MyISAM存储引擎数据单独写入或读取，速度过程较快而且占用资源相对较少。

（6）MyISAM存储引擎不支持外键约束，只支持全文索引

（7）每个MyISAM在磁盘上存储成三个文件，每一个文件的名字以表的名字开始，扩展名指出文件类型。

**.frm**文件存储**表定义**

**.MYD**文件存储**数据（MYData）**

**.MYI**文件存**储索引文件（MYIndex）**

## 3、MyISAM使用的生产场景

（1）公司业务不需要事务支持

（2）一般单方读取数据比较多的业务，或单方面写入数据比较多的业务，如：www.blog，图片信息数据库，用户数据库，商品库等业务，MyISAM存储引擎数据读写都比较频繁的场景不适合。

（3）对数据业务一致性要求不是非常高的业务

（4）使用读写并发访问相对较低的业务

（5）数据修改相对较少的业务

（6）服务器硬件资源相对比较差

## 二、InnoDB介绍

### 1、InnoDB的特点

（1）支持事务：支持4个事务隔离级别

（2）行级锁定，但是全表扫描仍然会是表级锁定

（3）读写阻塞与事务隔离级别相关

（4）具有非常高效的缓存特性：能缓存索引，也能缓存数据

（5）表与主键以簇的方式存储

（6）支持分区、表空间，类似oracle数据库

（7）支持外键约束，5.5以前不支持全文索引，5.5版本以后支持全文索引

（8）对硬件资源要求比较高

### 2、InnoDB使用的生产场景

（1）业务需要事务的支持

（2）行级锁定对高并发有很好的适应能力，但需要确保查询是通过索引来完成

（3）业务数据更新较为频繁的场景，如：论坛，微博

（4）业务数据一致性要求较高，如：银行业务

（5）硬件设备内存较大，利用InnoDB较好的缓存能力来提高内存利用率，减少磁盘IO的压力。

## 三、企业选择存储引擎的依据

1、需要考虑每个存储引擎提供了哪些不同的核心功能及应用场景

2、支持的字段和数据类型，所有引擎都支持通用的数据类型，但不是所有引擎都支持其他的字段类型，如二进制对象。

3、锁定类型：不同的存储引擎支持不同级别的锁定

表锁定：MyISAM、MEMORY

行锁定：InnoDB

页锁定：BDB

4、索引的支持

建立索引在搜索和恢复数据库中的数据的时候能显著提高性能

不同的存储引擎提供不同的制作索引的技术

有些存储引擎根本不支持索引

5、事务处理的支持

事务处理功能通过提供在相关表中更新和插入信息期间的可靠性

可根据企业业务是否要支持事务选择存储引擎

## 五、配置适合的存储引擎

1、查看数据库可配置的存储引擎

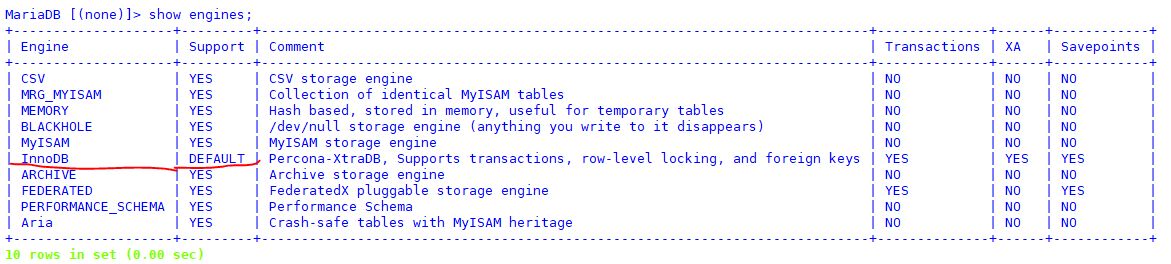
方法：登录MySQL，使用show engines;查看系统所支持的引擎

MariaDB [(none)]> show engines\G;

2、查看表正在使用的存储引擎

（1）查看当前MySQL的默认引擎





（2）查看表使用的存储引擎

方法一：SHOW TABLE STATUS FROM 库名 WHERE name = ‘表名’;

MariaDB [(none)]> show table status from auth where name='users'\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: users

Engine: InnoDB

Version: 10

Row\_format: Compact

Rows: 4

Avg\_row\_length: 4096

Data\_length: 16384

Max\_data\_length: 0

Index\_length: 0

Data\_free: 9437184

Auto\_increment: NULL

Create\_time: 2022-03-23 21:20:29

Update\_time: NULL

Check\_time: NULL

Collation: latin1\_swedish\_ci

Checksum: NULL

Create\_options:

Comment:

1 row in set (0.00 sec)

方法二：SHOW CREATE TABLE 表名;

MariaDB [(none)]> show create table auth.users\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: users

Create Table: CREATE TABLE `users` (

`user\_name` char(18) NOT NULL,

`user\_passwd` char(50) DEFAULT '',

PRIMARY KEY (`user\_name`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

1 row in set (0.00 sec)

3、配置存储引擎为所选择的类型

**方法一：ALTER TABLE 表名 ENGINE=引擎;**

**MariaDB [(none)]> alter table auth.users engine=MYISAM;**

**Query OK, 4 rows affected (0.01 sec)**

**Records: 4 Duplicates: 0 Warnings: 0**

**MariaDB [(none)]> show create table auth.users\G;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: users**

**Create Table: CREATE TABLE `users` (**

**`user\_name` char(18) NOT NULL,**

**`user\_passwd` char(50) DEFAULT '',**

**PRIMARY KEY (`user\_name`)**

**) ENGINE=MyISAM DEFAULT CHARSET=latin1**

**1 row in set (0.00 sec)**

**方法二：修改my.cnf的default-storage-engine为引擎**

**[root@mysql ~]# vim /etc/my.cnf**

**[mysqld]**

**default-storage-engine = InnoDB**

**[root@localhost ~]# systemctl restart mariadb**

**[root@mysql ~]# mysql -uroot -e 'show engines \G;'**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Engine: MRG\_MYISAM**

**Support: YES**

**Comment: Collection of identical MyISAM tables**

**Transactions: NO**

**XA: NO**

**Savepoints: NO**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Engine: CSV**

**Support: YES**

**Comment: CSV storage engine**

**Transactions: NO**

**XA: NO**

**Savepoints: NO**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Engine: MyISAM**

**Support: YES**

**Comment: Default engine as of MySQL 3.23 with great performance**

**Transactions: NO**

**XA: NO**

**Savepoints: NO**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Engine: InnoDB**

**Support: DEFAULT**

**Comment: Supports transactions, row-level locking, and foreign keys**

**Transactions: YES**

**XA: YES**

**Savepoints: YES**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Engine: MEMORY**

**Support: YES**

**Comment: Hash based, stored in memory, useful for temporary tables**

**Transactions: NO**

**XA: NO**

**Savepoints: NO**

**MariaDB> use auth;**

**MariaDB> create table user(name char(10),id int);**

**MariaDB> show create table user \G;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: user**

**Create Table: CREATE TABLE `user` (**

**`name` char(10) DEFAULT NULL,**

**`id` int(11) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8 //此时新建的表已经默认为InnoDB引擎了**

**1 row in set (0.00 sec)**

**方法三：CREATE TABLE 建立表时使用 engine=引擎**

**MariaDB> use auth;**

**MariaDB> create table id(id int) engine=myisam;**

**MariaDB> show create table auth.id;**

**+-------+--------------------------------------------------------------------------------------+**

**| Table | Create Table |**

**+-------+--------------------------------------------------------------------------------------+**

**| id | CREATE TABLE `id` (**

**`id` int(11) DEFAULT NULL**

**) ENGINE=MyISAM DEFAULT CHARSET=utf8 |**

**+-------+--------------------------------------------------------------------------------------+**

**1 row in set (0.00 sec)**

**方法四：mysql\_convert\_table\_format --user=root --password=密码 --engine=引擎 库名 表名**

**[root@web ~]# mysql\_convert\_table\_format --user=root --engine=Myisam auth user**

**[root@web ~]# mysql -e "show create table auth.user\G;"**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Table: user**

**Create Table: CREATE TABLE `user` (**

**`name` char(10) DEFAULT NULL,**

**`id` int(11) DEFAULT NULL**

**) ENGINE=MyISAM DEFAULT CHARSET=latin1**

**DBI报错解决方案：**

**[root@localhost ~]# mysql\_convert\_table\_format --user=root --engine=Myisam auth user**

**Can't locate DBI.pm in @INC (@INC contains: /usr/local/lib64/perl5 /usr/local/share/perl5 /usr/lib64/perl5/vendor\_perl /usr/share/perl5/vendor\_perl /usr/lib64/perl5 /usr/share/perl5 .) at /usr/local/mysql/bin/mysql\_convert\_table\_format line 19.**

**BEGIN failed--compilation aborted at /usr/local/mysql/bin/mysql\_convert\_table\_format line 19.**

**[root@localhost ~]# yum -y install perl-DBD-MySQL**