1. **MySQL的DELETE、TRUNCATE、DROP分析。**
2. [mysql语句中----删除表数据drop、truncate和delete的用法](http://www.cnblogs.com/shuaiandjun/p/6042600.html)
3. 程度从强到弱
4. 1、drop  table tb
5. drop将表格直接删除，没有办法找回
6. 2、truncate (table) tb
7. 删除表中的所有数据，不能与where一起使用
8. 3、delete from tb (where)
9. 删除表中的数据(可制定某一行)

**用户想要删除数据表可以使用DELETE、TRUNCATE以及DROP来删除数据。从功能上讲不带WHERE子句的DELETE、TRUNCATE和DROP是相似的。**

**三者的区别主要包括以下几点：**

1. **删除内容**

* **DROP操作会删除表结构、依赖的约束、索引以及触发器，并且会将依赖该表的所有存储过程和视图设置为invalid。**
* **TRUNCATE和DELETE则是只会删除表中的数据，并不会删除表结构。**

**因此，如果该表以后不再需要的话可以使用DROP删除，而如果后续还需要的话可以通过TRUNCATE或DELETE删除，因为这样可以不需要再重新建表。**

1. **删除空间**

* **DELETE操作并不会更改所占用的区的空间，高水位线不会发生改变。**
* **DROP操作将表所占用的空间全部释放。**
* **TRUNCATE操作是相对于先执行DROP操作，然后再执行CREATE操作，执行完成后会恢复初始表空间，高水位线复位（0）。**

1. **语句类型**

* **DELETE语句是DML（数据库操作语言），这操作会放到rollbacksegement中，事务提交后才生效；如果有相应的trigger，执行的时候将被触发。**
* **TRUNCATE、DROP语句是DDL（数据库定义语言），操作立即生效，原数据不放到rollbacksegement中，不能回滚，操作不触发trigger。**

1. **操作效率**

* **DROP > TRUNCATE > DELETE。DELETE操作是按照记录一行一行的删除的。**

1. **安全性**

* **使用DROP和TRUNCATE会导致整个表中的数据都被删除，使用需谨慎。**
* **使用上，想删除部分数据行用DELETE带上WHERE子句。想删除表用DROP。想保留表而将所有数据删除，如果和事务无关，用TRUNCATE即可。如果和事务有关，或者想触发trigger，使用DELETE。**

**PS:TRUNCATE 和DELETE重置 AUTO\_INCREMENT 情况分析。**

**区别：TRUNCATE能够快速清空一个表，并且重置AUTO\_INCREMENT的值。而DELETE只能一行一行的删除。**

**但对于不同的类型存储引擎需要注意的地方是：**

**A 对于MyIASM**

**TRUNCATE会重置AUTO\_INCREMENT的值为1。而DELETE后表仍然保持AUTO\_INCREMENT。**

**B 对于InnoDB**

**TRUNCATE会重置AUTO\_INCREMENT的值为1。DELETE后表仍然保持AUTO\_INCREMENT。但是在做DELETE整个表之后重启MySQL的话，则重启后的AUTO\_INCREMENT会被置为1。**

**也就是说，InnoDB的表本身是无法持久保存AUTO\_INCREMENT。DELETE表之后AUTO\_INCREMENT仍然保存在内存，但是重启后就丢失了，只能从1开始。实质上重启后的AUTO\_INCREMENT会从 SELECT 1+MAX(ai\_col) FROM t 开始。**

1. **MySQL查询某数据最值问题。**

* **SELECT MAX(字段名) FROM table\_name LIMIT 0,1 最大**
* **SELECT MIN(字段名) FROM table\_name LIMIT 0,1 最小**
* **SELECT \* FROM table\_name ORDER BY字段名 DESC LIMIT 0,1 最大**
* **SELECT \* FROM table\_name ORDER BY字段名 ASC LIMIT 0,1 最小**

**第二大值：**

* **SELECT MAX(字段名) FROM table\_name WHERE (字段名) NOT IN (SELECT MAX(字段名) FROM table\_name)**
* **SELECT \* FROM table\_name ORDER BY字段名 DESC LIMIT 1,1 最大**

**后面以此类推！**

1. **事务是什么？事务的四个特性。**

**①事务是应用程序中一系列严密的操作，所有操作必须成功完成，否则在每个操作中所作的所有更改都会被撤销。也就是事务具有原子性，一个事务中的一系列的操作要么全部成功，要么一个都不做。事务的结束有两种，当事务中的所有步骤全部成功执行时，事务提交。如果其中一个步骤失败，将发生回滚操作，撤销撤销之前到事务开始时的所有操作。事务通常以BEGIN TRANSACTION开始，以COMMIT或ROLLBACK结束。**

**②事务的ACID。**

**事务具有四个特征：原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这四个特性简称为ACID特性。**

**原子性：事务是数据库的逻辑工作单位，它对数据库的修改要么全部执行，要么全部不执行，是一个不可分割的工作单位。**

**一致性：事务前后，数据库的状态都满足所有的完整性约束。事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。比如，当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。如果数据库系统在运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是不一致的状态。**

**隔离性：并发执行的事务是隔离的，一个不影响一个。如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请求，使得在同一时间仅有一个请求用于同一数据。通过设置数据库的隔离级别，可以达到不同的隔离效果。**

**持续性：也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。接下来的其它操作或故障不应该对其执行结果有任何影响。**

**事务中的所有操作要么全部执行，要么都不执行；如果事务没有原子性的保证，那么在发生系统故障的情况下，数据库就有可能处于不一致状态。因而，事务的原子性与一致性是密切相关的。**

1. **事务的隔离级别。**

**数据库事务无非就两种：读取事务(SELECT)和修改事务(UPDATE、INSERT)。在没有事务隔离控制的时候，多个事务在同一时刻对同一数据的操作可能就会影响到最终期望的结果，通常有四种情况：**

**(1)修改时允许修改—丢失更新**

**(2)修改时允许读取—脏读**

**(3)读取时允许修改—不可重复读**

**(4)读取时允许插入—幻读**

**以上四种情况描述完毕，前三种是对同一条数据的并发操作，对程序的结果可能产生致命影响，尤其是金融等实时性，准确性要求极高的系统，绝不容许这三中情况的出现，相比第四种情况不会影响数据的真实性，在很多情况下是允许的，如社交论坛等实时性要求不高的系统。**

**四种情况问题严重性降低，但性能开销增加！因为不同的系统允许不同级别情况，所以就出现了事务的隔离级别—四种隔离级别：**

**(1)未授权读取(Read uncommitted)：一个事务更新时不允许更新，但允许读取。可以防止丢失更新，但不能防止脏读、不可重复读、幻读。(隔离级别最低)**

**(2)授权读取(Read committed):一个事务更新时不允许读取，必须等到更新事务提交后才能读取，可以防止丢失更新和脏读，但不能防止不可重复读、幻读。(隔离级别次低)**

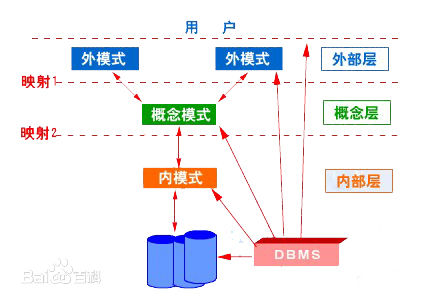
**(3)可重复读取(Repeatable read)：一个事务读取时，不允许更新，但允许插入。可以防止丢失更新、脏读、不可重复读，但不能防止幻读。**

**PS:这是MySQL的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会得到同样的数据行。InnoDB存储引擎通过多版本并发控制(MVCC)机制解决了幻读！**

**(4)序列化(Serializable)：提供严格的事务隔离，它要求事务序列化执行，事务只能一个接一个地执行，不能并发执行。**



**5.数据库模式关系表**



**用户级对应外模式，概念级对应概念模式，物理级对应内模式。数据库对应一个内模式，多个外模式。**

**6.SELECT语句的执行顺序：**

**SELECT的执行顺序：FROM ，WHERE ，GROUP BY , HAVING, (SELECT) ORDER BY , LIMIT**

**提纲：**

**WHERE ，HAVING 区别和用法。**

**主键，外键，唯一约束用法。（见笔记本）**

**索引**

索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。更通俗的说，数据库索引好比是一本书前面的目录，能加快数据库的查询速度。

**表连接**

**子查询**

**数据库索引的数据结构**

**数据库优化**

#### 1、选取最适用的字段属性

MySQL可以很好的支持大数据量的存取，但是一般说来，数据库中的表越小，在它上面执行的查询也就会越快。因此，在创建表的时候，为了获得更好的性能，我们可以将表中字段的宽度设得尽可能小。

例如，在定义邮政编码这个字段时，如果将其设置为CHAR(255),显然给数据库增加了不必要的空间，甚至使用VARCHAR这种类型也是多余的，因为CHAR(6)就可以很好的完成任务了。同样的，如果可以的话，我们应该使用MEDIUMINT而不是BIGIN来定义整型字段。

**另外一个提高效率的方法是在可能的情况下，应该尽量把字段设置为NOTNULL**，这样在将来执行查询的时候，数据库不用去比较NULL值。  
对于某些文本字段，例如“省份”或者“性别”，我们可以将它们定义为ENUM类型。因为在MySQL中，ENUM类型被当作数值型数据来处理，而数值型数据被处理起来的速度要比文本类型快得多。这样，我们又可以提高数据库的性能。

#### 2、使用连接（JOIN）来代替子查询(Sub-Queries)

MySQL从4.1开始支持SQL的子查询。这个技术可以使用SELECT语句来创建一个单列的查询结果，然后把这个结果作为过滤条件用在另一个查询中。例如，我们要将客户基本信息表中没有任何订单的客户删除掉，就可以利用子查询先从销售信息表中将所有发出订单的客户ID取出来，然后将结果传递给主查询，如下所示：

使用子查询可以一次性的完成很多逻辑上需要多个步骤才能完成的SQL操作，同时也可以避免事务或者表锁死，并且写起来也很容易。但是，有些情况下，子查询可以被更有效率的连接（JOIN）..替代。例如，假设我们要将所有没有订单记录的用户取出来，可以用下面这个查询完成：

连接（JOIN）..之所以更有效率一些，是因为MySQL不需要在内存中创建临时表来完成这个逻辑上的需要两个步骤的查询工作。

#### 3、使用联合(UNION)来代替手动创建的临时表

MySQL从4.0的版本开始支持union查询，它可以把需要使用临时表的两条或更多的select查询合并的一个查询中。在客户端的查询会话结束的时候，临时表会被自动删除，从而保证数据库整齐、高效。使用union来创建查询的时候，我们只需要用UNION作为关键字把多个select语句连接起来就可以了，要注意的是所有select语句中的字段数目要想同。下面的例子就演示了一个使用UNION的查询。

#### 4、事务

尽管我们可以使用子查询（Sub-Queries）、连接（JOIN）和联合（UNION）来创建各种各样的查询，但不是所有的数据库操作都可以只用一条或少数几条SQL语句就可以完成的。更多的时候是需要用到一系列的语句来完成某种工作。但是在这种情况下，当这个语句块中的某一条语句运行出错的时候，整个语句块的操作就会变得不确定起来。设想一下，要把某个数据同时插入两个相关联的表中，可能会出现这样的情况：第一个表中成功更新后，数据库突然出现意外状况，造成第二个表中的操作没有完成，这样，就会造成数据的不完整，甚至会破坏数据库中的数据。要避免这种情况，就应该使用事务，它的作用是：要么语句块中每条语句都操作成功，要么都失败。换句话说，就是可以保持数据库中数据的一致性和完整性。事物以BEGIN关键字开始，COMMIT关键字结束。在这之间的一条SQL操作失败，那么，ROLLBACK命令就可以把数据库恢复到BEGIN开始之前的状态。

事务的另一个重要作用是当多个用户同时使用相同的数据源时，它可以利用锁定数据库的方法来为用户提供一种安全的访问方式，这样可以保证用户的操作不被其它的用户所干扰。

#### 5、锁定表

尽管事务是维护数据库完整性的一个非常好的方法，但却因为它的独占性，有时会影响数据库的性能，尤其是在很大的应用系统中。由于在事务执行的过程中，数据库将会被锁定，因此其它的用户请求只能暂时等待直到该事务结束。如果一个数据库系统只有少数几个用户来使用，事务造成的影响不会成为一个太大的问题；但假设有成千上万的用户同时访问一个数据库系统，例如访问一个电子商务网站，就会产生比较严重的响应延迟。

其实，有些情况下我们可以通过锁定表的方法来获得更好的性能。下面的例子就用锁定表的方法来完成前面一个例子中事务的功能。

这里，我们用一个select语句取出初始数据，通过一些计算，用update语句将新值更新到表中。包含有WRITE关键字的LOCKTABLE语句可以保证在UNLOCKTABLES命令被执行之前，不会有其它的访问来对inventory进行插入、更新或者删除的操作。

#### 6、使用外键

**锁定表的方法可以维护数据的完整性，但是它却不能保证数据的关联性。这个时候我们就可以使用外键。**

例如，外键可以保证每一条销售记录都指向某一个存在的客户。在这里，外键可以把customerinfo表中的CustomerID映射到salesinfo表中CustomerID，任何一条没有合法CustomerID的记录都不会被更新或插入到salesinfo中。

该参数保证当customerinfo表中的一条客户记录被删除的时候，salesinfo表中所有与该客户相关的记录也会被自动删除。如果要在MySQL中使用外键，一定要记住在创建表的时候将表的类型定义为事务安全表InnoDB类型。该类型不是MySQL表的默认类型。定义的方法是在CREATETABLE语句中加上TYPE=INNODB。如例中所示。

#### 7、使用索引

索引是提高数据库性能的常用方法，它可以令数据库服务器以比没有索引快得多的速度检索特定的行，尤其是在查询语句当中包含有MAX(),MIN()和ORDERBY这些命令的时候，性能提高更为明显。

那该对哪些字段建立索引呢？

一般说来，索引应建立在那些将用于JOIN,WHERE判断和ORDERBY排序的字段上。尽量不要对数据库中某个含有大量重复的值的字段建立索引。对于一个ENUM类型的字段来说，出现大量重复值是很有可能的情况

例如customerinfo中的“province”..字段，在这样的字段上建立索引将不会有什么帮助；相反，还有可能降低数据库的性能。我们在创建表的时候可以同时创建合适的索引，也可以使用ALTERTABLE或CREATEINDEX在以后创建索引。此外，MySQL从版本3.23.23开始支持全文索引和搜索。全文索引在MySQL中是一个FULLTEXT类型索引，但仅能用于MyISAM类型的表。对于一个大的数据库，将数据装载到一个没有FULLTEXT索引的表中，然后再使用ALTERTABLE或CREATEINDEX创建索引，将是非常快的。但如果将数据装载到一个已经有FULLTEXT索引的表中，执行过程将会非常慢。

#### 8、优化的查询语句

绝大多数情况下，使用索引可以提高查询的速度，但如果SQL语句使用不恰当的话，索引将无法发挥它应有的作用。

下面是应该注意的几个方面。

* 首先，最好是在相同类型的字段间进行比较的操作。

在MySQL3.23版之前，这甚至是一个必须的条件。例如不能将一个建有索引的INT字段和BIGINT字段进行比较；但是作为特殊的情况，在CHAR类型的字段和VARCHAR类型字段的字段大小相同的时候，可以将它们进行比较。

* 其次，在建有索引的字段上尽量不要使用函数进行操作。

例如，在一个DATE类型的字段上使用YEAE()函数时，将会使索引不能发挥应有的作用。所以，下面的两个查询虽然返回的结果一样，但后者要比前者快得多。

* 第三，在搜索字符型字段时，我们有时会使用LIKE关键字和通配符，这种做法虽然简单，但却也是以牺牲系统性能为代价的。

**排它锁、共享锁**

**http://www.cnblogs.com/hoobey/p/5223636.html**

#### 锁

共享锁用于读取数据操作，它是非独占的，允许其他事务同时读取其锁定的资源，但不允许其他事务更新它。

独占锁也叫排他锁，适用于修改数据的场合。它所锁定的资源，其他事务不能读取也不能修改。

当一个事务访问某种数据库资源时，如果执行select语句，必须先获得共享锁，如果执行insert、update或delete语句，必须获得独占锁，这些锁用于锁定被操作的资源。

当第二个事务也要访问相同的资源时，如果执行select语句，也必须先获得共享锁，如果执行insert、update或delete语句，也必须获得独占锁。此时根据已经旋转在资源上的锁的类型，来决定第二个事务应该等待第一个事务解除对应资源的锁定，还是可以立刻获得锁。

**Mysql的触发器**

简单的说，就是一张表发生了某件事（插入、删除、更新操作），然后自动触发了预先编写好的若干条SQL语句的执行

特点及作用

特点：触发事件的操作和触发器里的SQL语句是一个事务操作，具有原子性,要么全部执行，要么都不执行；

作用：保证数据的完整性，起到约束的作用；