1. **MySQL的DELETE、TRUNCATE、DROP分析。**

**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rollback segement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rollback segement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)**

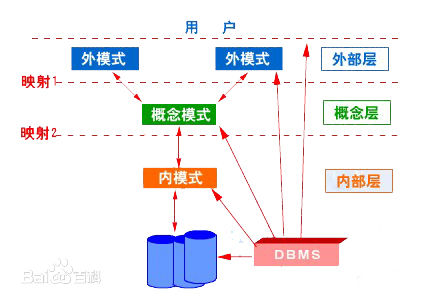
**(3)可重复读取(Repeatable read)：一个事务读取时，不允许更新，但允许插入。可以防止丢失更新、脏读、不可重复读，但不能防止幻读。**

**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。**

**索引！**

**表连接**

**子查询**

**数据库优化**

**排它锁、共享锁**