1. 在REPEATABLE READ隔离级别下，InnoDB,MVCC具体是如何操作的
2. SELECT
   1. 只查询版本早于当前事务版本的数据行（即行的系统版本号小于等于事务的版本号），这样可以确保事务读取的行，要么是在事务开始前已经存在，要么是事务自身插入或修改过的。
   2. 行的删除版本号要么未定义，要么大于当前事务版本号，这样就可以确保事务读取到的行，在事务开始之前未被删除（如大于说明开始后才被删除的，仍然要返回）。
3. INSERT
   1. 为新插入的每一行保存当前系统版本号为行版本号。这样SELECT事务就不会读取插入的行，因为新插入行版本号高。
4. DELETE
   1. 为删除的每一行保存当前系统版本号作为删除标识。
5. UPDATE
   1. 为插入一行新纪录，保存当前系统版本号作为行版本号，同时保存当前系统版本号到原来的行作为行删除标识！即UPDATE是先插入在删除！怎么保证主键唯一？噢！每行都是通过额外空间存储的，update后还是一行！

5）版本控制只用于REA COMMITED和REPEATABLE READ，因为别的更本用不上。

1. InnoDB是数据库默认引擎，被设计用来处理大量的短期事务，短期事务大部分情况是正常提交的，很少会被回滚。innoDB的性能和自动奔溃恢复特性，使得它在非事务存储中的需求也很流行。
2. innoDB的数据存储在表空间中，表空间是由innoDB管理的一个黑盒子，由一系列数据文件组成。在MySql4.1以后，innoDB可以将每个表的数据和索引存放在不同的文件中。
3. innoDB采用MVCC来支持高并发，并且实现了四个标准的隔离级别。默认是REPEATABLE READ，并且通过间隙锁策略防止幻读，即间隙锁不仅仅锁定查询涉及的行，还会对索引中的间隙进行锁定。如锁定 1 4 6，同时锁定了235，防止235插入。
4. archive引擎：用于需要快速插入的场景，且查询是全表，适合日志和数据采集类应用，这些应用做数据分析是往往需要全表扫描的场景。
5. csv引擎：用于数据交换机制，非常有用。
6. memory引擎：快速访问数据，且数据不会被修改。重启后表结构还在，但数据丢失。用于：
   1. 查找或映射表，例如将邮编和州名映射的表
   2. 用于缓存周期性聚合数据的结果
   3. 用于保存数据分析产生的中间数据
   4. 支持hash索引，查询非常快，但是表级锁，并发写性能低。不支持BLOB和TEXT类型，且列长短是固定的。
   5. 临时表可以使用任何引擎，内部保存中间结果用的就是memory引擎，但中间结果超出memory时，或含有BLOB或TEXT时，会转成MyISAM引擎。临时表使用完表结构也会不存在。区别于memory引擎的表。
7. NDB集群引擎
8. 如何选择存储引擎
   1. 除非需要用到某些InnoDB不具备的特性，并没有办法替代，否则都应该优先选择InnoDB。
   2. 若要用到全文索引，优先考虑 innoDB+Sphinx组合，而不是使用MyISAM.
   3. 如果不在乎可扩展能力和并发能力，也不在乎崩溃后的数据丢失问题，却对innoDB的占用空间过多比较敏感，可以选择MyISAM。
   4. 除非万不得已，否则建议不要混合使用多种存储引擎。