第一章：

1. InnoDB的体系结构包括哪两大部分？
2. InnoDB后台线程的作用？

一、刷新缓存内存池到最新的状态；二、刷新脏数据到数据库保证数据一致性；三、保证当数据库发生破坏时恢复数据库

* 1. 1个Main Thread：保证数据一致性（缓存和数据库），包括脏页刷新、合并插入缓存、undo页回收等
  2. 4个IO Thread：InnoDB中存在很多来自客户端的异步IO请求（AIO），那么InnoDB接收请求后由于是异步，所以不会立刻回调结果，此时InnoDB就需要有IO Thread去处理这些请求
  3. 1个Purge Thread：清除那些已经提交了事务的无用的undo log
  4. 1个Page Cleaner Thread：完成脏页的刷新（内存脏数据刷新到磁盘）

1. InnoDB内存池的作用？

一、维护存储各个InnoDB线程在使用的时候所需要的各个内存结构（）

二、存储缓存，加快客户端与数据库间读写的速度

三、保存重做日志缓存

4. 内存池的主要组成部分：

数据页、索引页、插入缓冲、锁信息、自适应哈希索引、重做日志缓冲

5. 内存池的其他组成部分：

LRU List（存储热点页的List，保证在页面置换的时候替换非热点页。midpoint技术）

Free List（存储空白页）

Flush List（存储脏页，checkpoint刷新脏页就是将Flush List中的脏页刷新到磁盘）

1. 什么是LRU midpoint技术？
2. 为什么LRU List 要使用LRU midpoint insertion而不是用朴素的LRU技术？
3. 重做日志缓冲（redo log buffer）作用？
4. 内存池的重做日志缓冲中的redo log刷新到磁盘中重做日志缓冲的3种情况？
5. 为何使用checkpoint技术？（缩短恢复数据库时间、刷新脏页腾出足够的缓冲池空闲区域、刷新redolog保证redolog缓冲区域足够存储新的redolog）
6. 有多少种checkpoint技术？
   1. Sharp checkpoint
   2. Fuzzy checkpoint
      1. Main thread checkpoint
      2. Flush LRU List checkpoint
      3. Async/Sync Flush checkpoint
      4. Dirty page too much checkpoint
7. Main thread 的工作原理
   1. 主循环（主线程每秒都会做的操作、主线程每10秒都会做的操作）
   2. 后台循环
8. 为何需要插入缓冲技术？（插入一个新的非聚集索引到非聚集索引页要进行离散读整个非聚集索引，不像聚集索引那样直接插入，速度慢）
9. 引入了插入缓冲技术能带来什么好处？（解决了因为仅仅一次非聚集索引插入就单独进行非聚集索引插入的弊端，等很多次插入操作混合，在一块进行非聚集索引插入）
10. 插入缓冲技术怎么实现？（先判断，再进行插入缓冲技术）
11. 使用插入缓冲技术要符合哪两个条件？（非聚集索引，非唯一的非聚集索引）

第二章

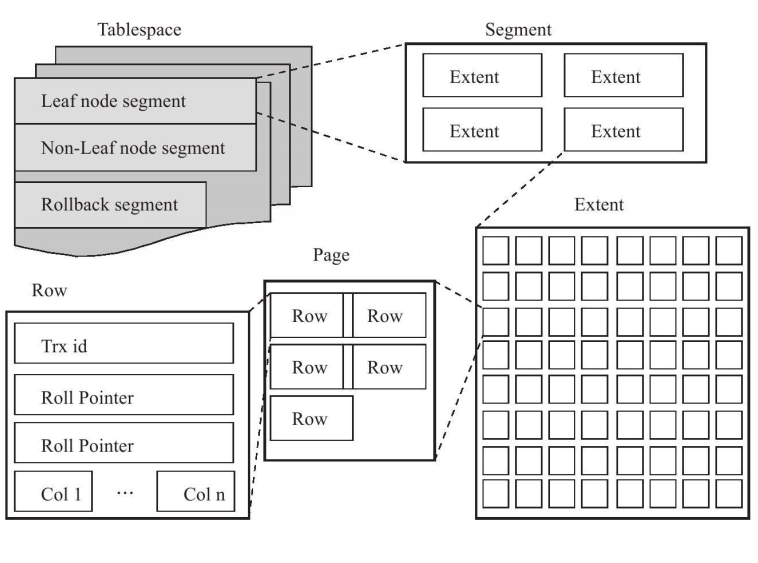
1. InnoDB的五个特性：插入缓冲、两次写、AHI自适应哈希索引、AIO异步IO、邻近页刷新
2. LRU List和自适应哈希AHI都是对热点页操作，不同之处：LRU List起到一个告知该替换哪个页的功能；而AHI起到一个直接定位搜索热点页访问热点页数据的功能
3. 插入索引的内部结构？
4. Insert buffer bitmap的作用？（记录每个辅助索引页的可用空间，当一个辅助索引页的bitmap显示可用空间快要满了，那么此时会立即进行插入缓冲的合并）
5. 插入缓冲的升级：change buffer，解决了插入缓冲宕机恢复数据慢以及写密集操作时占用内存空间大的问题，使得缓冲增加了insert buffer、delete buffer、purge buffer
6. Merge insert buffer发生的三个时间点？（索引页加载到缓冲池、bitmap空间不可用、master thread）
7. 非辅助索引的节点信息？（辅助索引-主键的映射）
8. 合并插入发生了什么？合并插入并非刷新到磁盘，而是将插入缓冲的内容插入到缓冲池对应的索引页上
9. 为什么需要两次写？（两次写和redo log共同保证了数据库宕机后数据的恢复问题，redo log仅仅是记录着脏页变化的物理信息）
10. AHI自适应哈希索引为什么叫自适应？（MySQL自动监控是否对某一些热点页新构建哈希索引）
11. AHI自适应哈希索引解决了什么问题？
12. AIO异步IO相比起同步IO的好处是什么（同步IO是阻塞式的IO，知道处理完同步IO系统才会接受下一个IO）
13. AIO具体做了什么（接受异步IO并不着急处理，而是等待多个IO的请求，进行合并再进行处理，数据库实例中的IO线程也是可以异步处理IO操作，只要后面回调处理结果就好）
14. 数据库何时使用异步IO？（read ahead操作时、脏页刷新也就是写入磁盘的时候）
15. 数据库的预读操作？

第三章

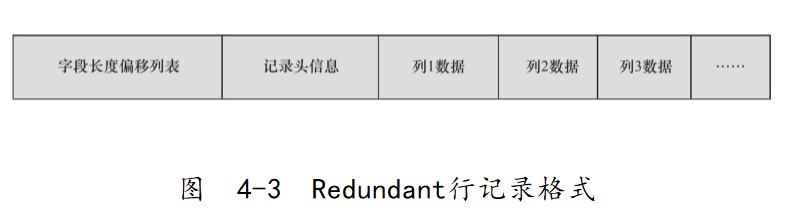
1. 什么是视图？（视图是一个虚拟表，其查询的数据来自于视图定义时的 as select xx 查询语句。视图的列来自于一个表或多个表，所以视图不可以和表名重名。通常用户可以保存在.frm文件下）。
2. 视图的作用？（简化用户查询操作；给数据库操作带来安全性；视图对重构数据库提供了一定程度的逻辑独立性）

第四章

1. 每张表都有一个主键，没指定就由系统指定
   1. 优先选择表中最早定义的非空的唯一索引作为主键（NOT NULL && UNIQUE KEY）
   2. 如果不存在这个索引，那么就新建一列（非空唯一递增的隐藏主键）
2. 表空间结构包括几个部分？（表空间，段，区，页，行）
3. 表空间中包含3种段segment（数据段、索引段、undo段回滚段），并不是每个表空间都有划分成段，但区和页是必然存在的
4. 一个段segment里面包含着4-5个区extent
5. 一个区extent的大小是1M，里面可以存储有64个页（区里面的页都是连续的），一个页大小是16K
6. 常见页的类型：数据页，undo-页，插入缓冲位图（insert buffer bitmap）



1. 一行记录保存由什么部分组成？（变长字段长、NULL标志位、记录头信息、列1、列2、........），除此之外每行也有两个隐藏列（事务ID和回滚指针）用于undo页事务回滚



1. 一页由什么部分组成？（文件头[记录当前页是什么类型的页等]、page头[记录当前页的一些基本状态信息]、上界和下界[这个页的物理地址哪里是保存数据的地方]、UserRecord和freeRecord、pageDirctory[记录这个page的记录的相对位置]、File尾部）
2. pageDirectory的作用？（用于在此页搜索数据时，用这个directory信息进行页数据的二分查找；注意：这指的是已经定位到某个页，找记录行的过程种起作用）
3. 约束的作用？（保证数据完整性）
4. 数据完整性包括哪三个方面？（实体完整性[主键]、域完整性[存储数据的特定条件]、参照完整性[表与表之间靠外键进行约束]）
5. MySQL具体的约束有哪些？（Primary key、Unique key、Foreign key、not null）
6. 什么是触发器？（create trigger，触发器指的是在insert、delete、update命令之前或之后自动调用sql命令的过程）
7. 外键约束具体有多少种形式？（级联cascade、on update、on delete、set null、on action）
8. 有外键的是子表，外键参考的是父表，子表是听父表的
9. 索引
10. 数据库中有三种常见的索引？
11. AHI为何叫自适应的？（存储引擎根据表的实际情况自己决定构建AHI，来代替B+树索引）
12. 聚集索引的B+树索引的叶子节点是数据页，定位了数据页后会将他加入缓冲区内存，由page directory进行查找相对应的行
13. 平衡二叉树和二叉搜索树的差别？优势？
14. B树（多路平衡查找树）和平衡二叉树的差别？优势？（此时不是二叉的，每个节点可能有多个分支，压缩了二叉树的深度，提高平均搜索速度）
15. B+树和B树的差别？优势（B树的叶子节点和非叶子节点都存储键和值，而B+树的非叶子节点只存储键，进一步压缩二叉树的深度，提高平均搜索速度）
16. B+树中，非叶子节点就是索引页，叶子节点就是数据页
17. B+树数据页插入的情况？（数据页索引页都没满--直接插入数据页；数据页满了索引页没满--分裂数据页，把中间索引提到索引页；数据页和索引页都满了，两个都分裂）
18. MySQL在B+树的插入操作中，由于经常需要分页分裂，导致所花费的时间很长。如何优化？（B+树索引页记录数满时就不进行按中间数据拆分上移，而是另一一个新的数据页在右侧；这个就需要插入记录是顺序的）
19. 什么是聚集索引？（聚集索引就是一张表，依靠这张表的主键信息进行构建一棵B+树，叶子节点里面存储的就是这个表中所有的数据行记录）
20. 什么是非聚集索引、辅助索引、二级索引？（B+树叶子节点并不存储这个表的具体行记录数据，而是存储的是对应于这个表中该记录的聚集索引的信息；B+树叶子节点的记录是按顺序进行排序的）
21. 每张表中可以有多个非聚集索引
22. B+树节点插入时，用向右分裂进行优化（不分裂另起一个索引和一个数据页只能用于聚集索引，而非聚集索引还是会进行一个分裂，但不是从最中间的数据节点进行分裂，这样损失空间，会造成更多的分裂）
23. Cardinality（表中某个索引数据里，不重复行的数量；用于判断是否构建索引，越高那么构建索引的效果就十分好）