MySQL InnoDB小结

数据库中事务的概念

一、事务

所谓事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。例如，在关系数据库中，一个事务可以是一条SQL语句、一组SQL语句或者整个程序。

事务和程序是两个概念。一般的讲，一个程序中包含多个事务。
事务的开始和结束可以由用户显示控制。如果用户没有显示的定义事务，则有DBMS按照缺省规定自动划分事务。
在SQL中，定义事务的语句有3条：

BEGIN TRANSACTION 、COMMIT 、ROLLBACK

事务通常以BEGIN TRANSACTION开始，以COMMIT或ROLLBACK结束。COMMIT表示提交，即提交事务的所有操作。具体的说就是将事务中所有对数据库的更新写回到磁盘上的物理数据库中去，事务正常结束。ROLLBACK表示回滚，即在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销，回滚到事务开始时的状态。这里的操作指对数据库的更新操作。

二、事务的特性

事务具有四个特性：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（ACID properties）。

1.原子性

事务是数据库的逻辑工作单位，事务中包括的诸操作要么都做，要么都不做。

2.一致性

事务执行的结果必须是使数据库从一个一致性状态变成另一个一致性状态。因此当数据库指包含成功事务提供的结果时，就说数据库处于一致性状态。如果数据库系统运行中发生故障，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一写入物理数据库，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。

3.隔离性

一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对其他并发事务是隔离的，并发执行的各个事务之间不能相互干扰。

4.持续性

持续性也称永久性（Permanence），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。接下来的其他操作或故障不应该对其执行结果有任何影响。

事务是恢复和并发控制的基本单位。所以下面的考虑均与事务为对象。
保证事务ACID特性是事务管理的重要任务。事务ACID特性可能遭到破坏的因素有：

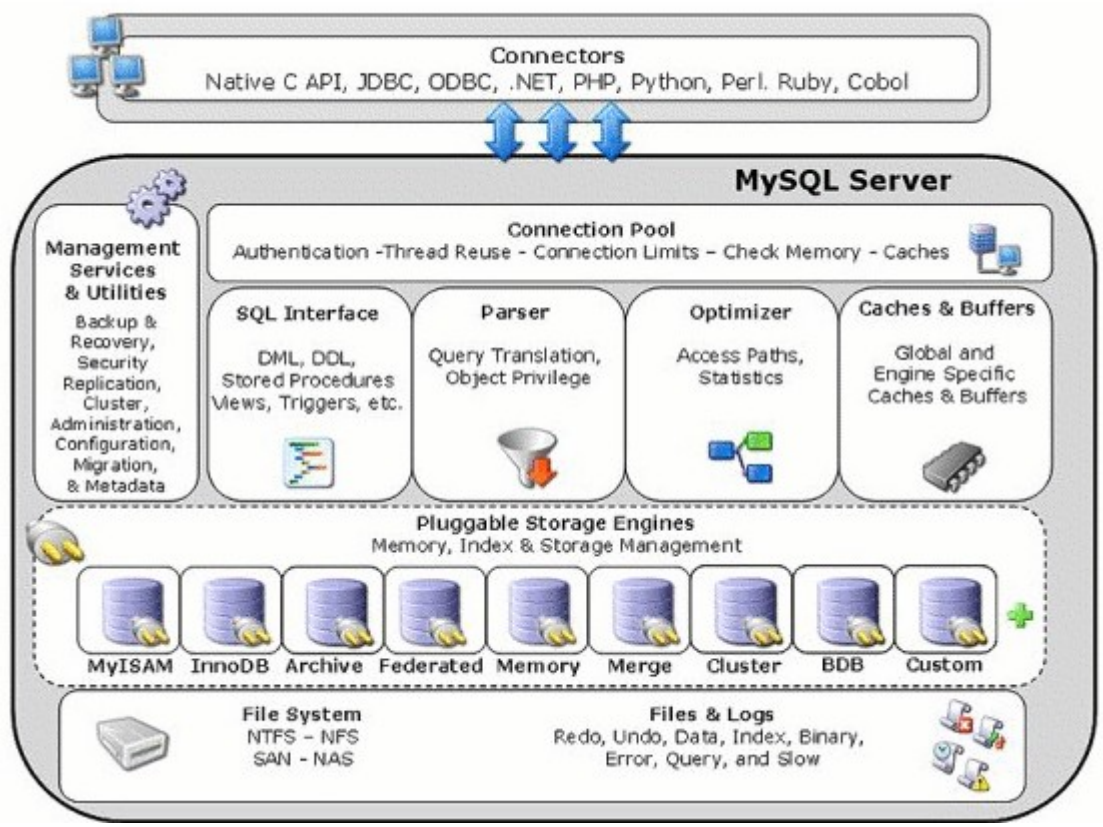
（1）多个事务并行运行时，不同事务的操作交叉执行；
（2）事务在运行过程中被强行停止。

在第一种情况下，数据库管理熊必须保证多个事务的交叉运行不影响这些事务的原子性。在第二种情况下，数据库管理熊必须保证被强行终止的事务对数据库和其他事务没有任何影响。

这些就是数据库管理系统中恢复机制和并发控制机制的责任。

MySQL体系结构

MySQL特点：插件式体系结构。



- 管理和服务组件
- 连接池组件
- sql接口组件
- 查询分析器组件
- 优化器组件
- 缓冲组件
- 插件式存储引擎
- 物理文件

MySQL各个存储引擎

- InnoDB

  面向OLTP（Online Transaction Processing，在线事务处理）、行锁、支持外键、非锁定读、默认采用repeaable（可重复读）级别、通过next-key locking策略避免幻读、插入缓冲、二次写、自适应哈希索引、预读

- MyISAM

  不支持事务、表锁、全文索引、适合OLAP（Online Analysis Processing，在线分析处理）。其中myd:放数据文件，myi:放索引文件

- ndb

  集群存储引擎，share nothing，可提高可用性

- memory

  数据存放在内存中，表锁，并发性能差，默认使用哈希索引

- archive

  只支持insert和select zlib算法压缩1：10，适合存储归档数据如日志等、行锁

- maria

  目的取代myisam、缓存数据和索引、行锁、mvcc

Feature	MyISAM	BDB	Memory	InnoDB	Archive	NDB
Storage Limits	No	No	Yes	64TB	No	Yes
Transactions (commit, rollback, etc.)		✔		✔		
Locking granularity	Table	Page	Table	Row	Row	Row
MVCC/Snapshot Read				✔	✔	✔
Geospatial support	✔					
B-Tree indexes	✔	✔	✔	✔		✔
Hash indexes			✔	✔		✔
Full text search index	✔					
Clustered index				✔		
Data Caches			✔	✔		✔
Index Caches	✔		✔	✔		✔
Compressed data	✔				✔	
Encrypted data (via function)	✔	✔	✔	✔	✔	✔
Storage cost (space used)	Low	Low	N/A	High	Very Low	Low
Memory cost	Low	Low	Medium	High	Low	High
Bulk Insert Speed	High	High	High	Low	Very High	High
Cluster database support						✔
Replication support	✔	✔	✔	✔	✔	✔
Foreign key support				✔		
Backup/Point-in-time recovery	✔	✔	✔	✔	✔	✔
Query cache support	✔	✔	✔	✔	✔	✔
Update Statistics for Data Dictionary	✔	✔	✔	✔	✔	✔

InnoDB特性

主体系结构

默认7个后台线程。

4个IO Thread

- insert buffer
- log
- read
- write

1个Master Thread（优先级最高）、

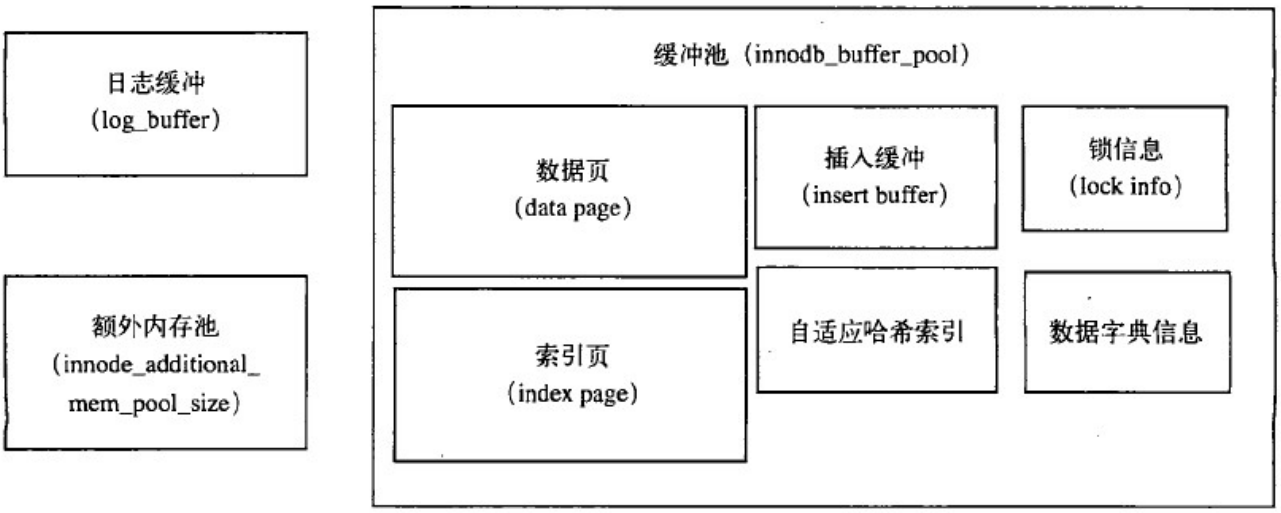
1个锁（lock）监控线程、

1个错误监控线程，可以通过 `SHOW ENGINE innodb STATUS` 来查看。

新版本已对默认的read thread和write thread增大到4个，可以通过 `SHOW VARIABLES LIKE 'innodb_io_thread%'` 查看。

存储引擎组成

- 缓冲池（buffer pool）
- 重做日志缓冲（redo log buffer）
- 额外的内存池（additional memory pool）



缓冲池

最大块内存，用来存放各种数据的缓存包括：

- 有索引页
- 数据页
- undo页
- 插入缓冲
- 自适应哈希索引
- innodb存储的锁信息
- 数据字典信息等。

工作方式总是将数据库文件按页（每页16K）读取到缓冲池，然后按最近最少使用（LRU）的算法来保留在缓冲池中的缓存数据。如果数据库文件需要修改，总是首先修改在缓存池中的页（发生修改后即为脏页），然后再按照一定的频率将缓冲池的脏页刷新到文件。

日志缓冲

将重做日志先放入这个缓冲区，然后按照一定的频率刷新到重做日志文件。

Master Thread

主循环（loop）执行每秒一次的操作：

- 日志缓冲刷新到磁盘，即使这个事务还没有提交（总是执行，所以再大的事务commit的时间也是很快的）
- 合并插入缓冲（innodb当前一秒发生的io次数小于5次则执行）
- 至多刷新100个innodb的缓冲池中的脏页到磁盘（超过配置的脏页所占缓冲池比例则执行，在配置文件中innodb_max_dirty_pages_pac决定，默认是90，新版本是75，google建议是80）
- 如果当前没用用户活动，切换到backgroud loop

主循环每10秒一次执行的操作：

- 刷新100个脏页到磁盘（过去10秒IO操作小于200次则执行）
- 合并至多5个插入缓冲（总是）
- 将日志缓冲到磁盘（总是）
- 删除无用的Undo页（总是）
- 刷新100个或者10个脏页到磁盘（有超过70%的脏页，刷新100个脏页，否则刷新10个脏页）
- 产生一个检查点

background loop，若当前没有用户活动（数据库空闲）时或者数据库关闭时，就会切换到这个循环：

- 删除无用的undo页（总是）
- 合并20个插入缓冲（总是）
- 跳回到主循环（总是）
- 不断刷新100个页，直到符合条件（可能在flush loop中完成）

如果flush loop中也没有什么事情可以做了，InnoDB存储引擎会切换到suspend_loop，将master thread挂起，等待事件的发生。若启用了InnoDB存储引擎，却没有使用任何InnoDB存储引擎的表，那么master thread总是处于挂起状态。

插入缓冲

Insert Buffer是物理页的一个组成部分，而不是缓冲池的一部分。根据B+树算法的特点，插入数据的时候主键索引是顺序的，不会造成数据库的随机读取。而对于非聚集索引（即辅助索引），叶子节点的插入不再是顺序的了，这时需要离散的访问非聚集索引，插入性能降低。

InnoDB引入插入缓冲，判断非聚集索引页是否在缓冲池中，如果在则直接插入，如果不在则先放在插入缓冲区中，然后根据上述master thread介绍的，会有一定频率的将插入缓冲合并。

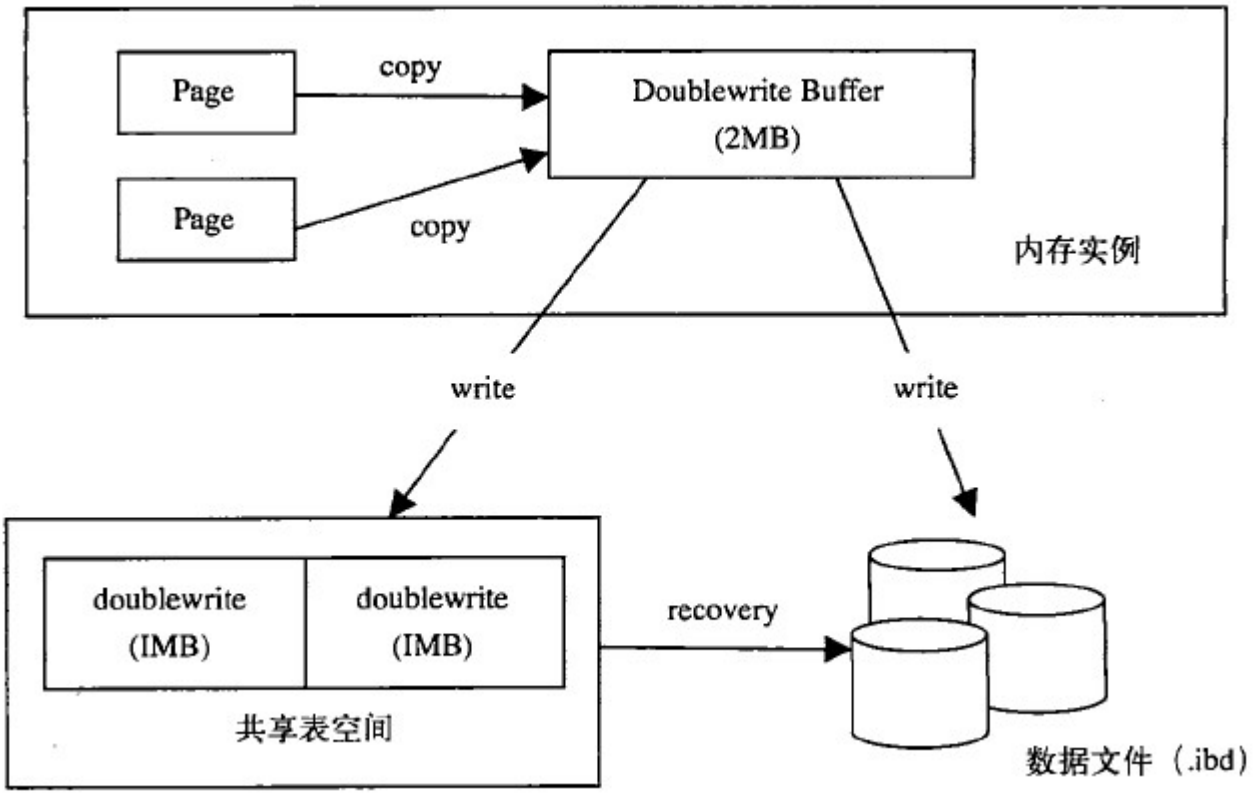
此外，辅助索引并不是唯一的，因为在插入到插入缓冲时，并不去查找索引页的情况，否则仍然会造成随机读，失去插入缓冲的意义了。

插入缓冲可能会占缓冲池中内存，默认也能会占到1/2，所以可以将这个值调小点，到1/3。通过IBUF_POOL_SIZE_PER_MAX_SIZE来设置，2表示1/2,3表示1/3。

两次写

它带来InnoDB数据的可靠性。如果写失效，可以通过重做日志进行恢复，但是重做日志中记录的是对页的物理操作，如果页本身损坏，再对其进行重做是没有意义的。所以，在应用重做日志之前，需要一个页的副本，当写入失效发生时，先通过页的副本来还原该页，再进行重做，这就是double write。

恢复数据 = 页副本 + 重做日志



自适应哈希索引

InnoDB存储引擎提出一种自适应哈希索引，存储引擎会监控对表上索引的查找，如果观察到建立建立哈希索引会带来速度的提升，则建立哈希索引，所以称之为自适应的。自适应哈希索引只能用来搜索等值的查询，如 `SELECT * FROM table WHERE index_col = '***'` , 此外自适应哈希是由InnoDB存储引擎控制的，我们只能通过innodb_adaptive_hash_index来禁用或启用，默认开启。

MySQL文件

参数文件

告诉Mysql实例启动时在哪里可以找到数据库文件，并且指定某些初始化参数，这些参数定义了某种内存结构的大小等设置。用文件存储，可编辑，若启动时加载不到则不能成功启动（与其他数据库不同）。

参数有动态和静态之分，静态相当于只读，动态是可以set的。如我们通过 `SHOW VARIABLES LIKE '***'` 查出来的key、value值，是可以通过 `SET key = value` 直接修改的。同时，修改时还有作用域之分，即这个session个有效和全局有效，在对应的key前加上session或global即可，如：
`SELECT @@seesion.read_buffer_size`
`SET @@global.read_buffer_size`

日志文件

用来记录Mysql实例对某种条件做出响应时写入的文件。如错误日志文件、二进制日志文件、慢查询日志文件、查询日志文件等。

二进制文件

不记录查询，只记录对数据库所有的修改操作。目的是为了恢复（point-in-time修复）和复制。

重做日志文件

实例和介质失败，重做日志文件就能派上用场，如数据库掉电，InnoDB存储引擎会使用重做日志恢复到掉电前的时刻，以此来保证数据的完整性。

二进制日志和重做日志的区别：

- 首先，二进制日志会记录所有与Mysql有关的日志记录，包括InnoDB、MyISAM、Heap等其他存储引擎的日志。而InnoDB存储引擎重做日志只存储有关其本身的事务日志；
- 其次内容不同，不管将二进制日志文件记录的格式设为STATEMENT还是ROW，又或者是MIXED，其记录的都是关于一个事务的具体操作内容。而InnoDB存储引擎的重做日志文件记录的关于每个页的更改的物理情况；
- 此外，写入时间不同。二进制日志文件是在事务提交前进行记录的，而在事务进行的过程中，不断有重做日志条目被 写入重做日志文件中。

MySQL InnoDB表

表空间：表空间可看做是InnoDB存储引擎逻辑结构的最高层。

段：表空间由各个段组成，常见的段有数据段、索引段、回滚段等。

区：由64个连续的页组成，每个页大小为16kb，即每个区大小为1MB。

页：每页16kb，且不能更改。常见的页类型有：数据页、Undo页、系统页、事务数据页、插入缓冲位图页、插入缓冲空闲列表页、未压缩的二进制大对象页、压缩的二进制大对象页。

行：InnoDB存储引擎是面向行的(row-oriented)，每页最多允许存放7992行数据。

行记录格式：常见两种行记录格式Compact和Redundant，mysql5.1版本后，主要是Compact行记录格式。对于Compact，不管是char型还是varchar型，null型都是不占用存储空间的；对于Redudant,varchar的null不占用空间，char的null型是占用存储空间的。

varchar类型的长度限制是65535，其实达不到，会有别的开销，一般是65530左右，这还跟选取的字符集有关。此外这个长度限制是一整行的，例如：

```
create table test(a varchar(22000), b varchar(22000), c varchar(22000)) charset=latin1 engine=innodb
```

也会报错。

sql语句应该考虑哪些安全性？

（1）防止sql注入，对特殊字符进行转义，过滤或者使用预编译的sql语句绑定变量。

（2）最小权限原则，特别是不要用root账户，为不同的类型的动作或者组建使用不同的账户。

（3）当sql运行出错时，不要把数据库返回的错误信息全部显示给用户，以防止泄漏服务器和数据库相关信息。

简单描述MySQL中索引、主键、唯一索引、联合索引的区别，对数据库的性能有什么影响

（1）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。

（2）普通索引（由关键字KEY或INDEX定义的索引）的唯一任务是加快对数据的访问速度。

（3）普通索引允许被索引的数据列包含重复的值，如果能确定某个数据列只包含彼此各不相同的值，在为这个数据索引创建索引的时候就应该用关键字UNIQE把它定义为一个唯一所以，唯一索引可以保证数据记录的唯一性。

（4）主键，一种特殊的唯一索引，在一张表中只能定义一个主键索引，逐渐用于唯一标识一条记录，是用关键字PRIMARY KEY来创建。

（5）索引可以覆盖多个数据列，如像INDEX索引，这就是联合索引。

（6）索引可以极大的提高数据的查询速度，但是会降低插入删除更新表的速度，因为在执行这些写操作时，还要操作索引文件。