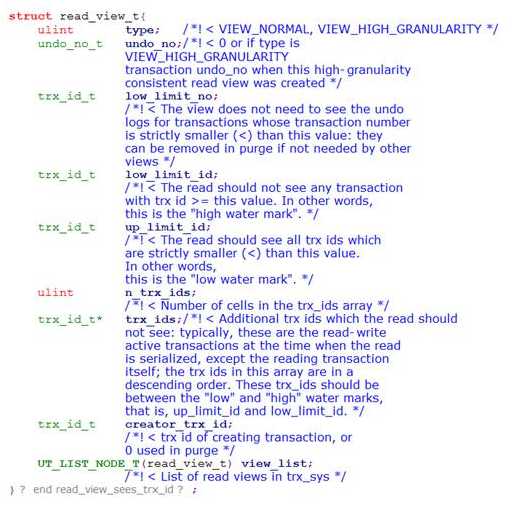
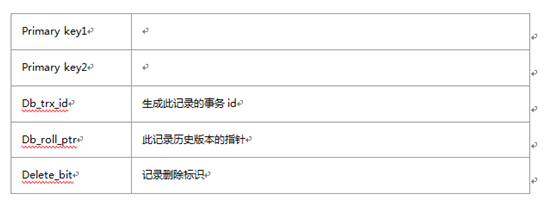
1. **Read view的结构**

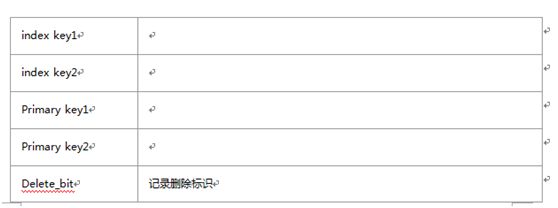
前文中我们首先介绍了read view结构如下：



* low\_limit\_id：trx->id >=low\_limit\_id的事务，当前view不可见
* up\_limit\_id：trx->id<up\_limit\_id的事务，当前view一定可见
* trx\_ids：系统中所有活跃事务id列表
* view\_list：view列表

聚集索引行结构



二级索引记录结构

在二级索引页面中，有DB\_MAX\_ID：标识修改索引页面的最大事务ID。

1. **插入和更新操作对索引的影响**

创建表和初始数据如下：

*–create table and index*

*create table test (id int , comment char(50), primary key(id), key test(comment)) engine=InnoDB;*

*–Insert*

*insert into test values(1, ‘aaa’);*

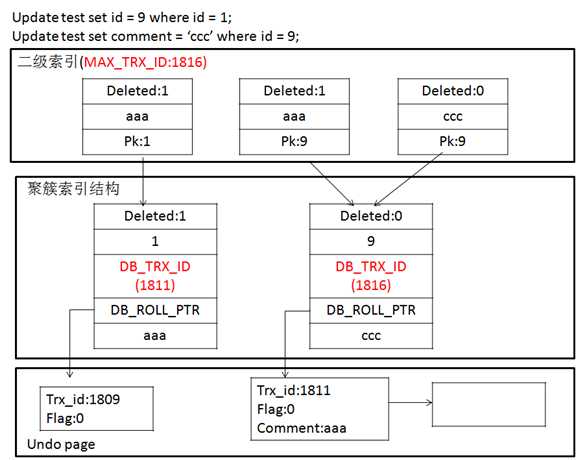
*insert into test values(2, ‘bbb’);*

分别进行主键修改和辅助索引key修改

*update test set id = 9 where id = 1;*

*update test set comment = ‘ccc’ where id = 9;*

索引结构变为如下：

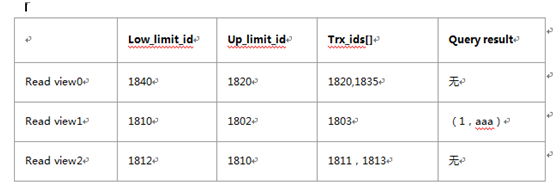


* 更新主键，聚簇索引/二级索引均无法进行in place update，均会产生两个版本；
* 更新非主键索引，聚簇索引可以in place update；二级索引产生两个版本；
* 聚簇索引记录undo，二级索引不记录undo；
* 更新聚簇索引，记录旧版本会进入rollback segment undo page
* 更新二级索引，同时需要判断是否修改索引页面的MAX\_TRX\_ID
* 属于同一事务的undo记录，在undo page中保存逆向指针

1. **查询（查询主键索引、二级索引）**
2. **查询主键索引**

select \* from test where id = 1;

查询的结果和当前查询所属的read view 的值有关系，read view值不同，产生的结果也不同



对于read view0，db\_trx\_id(1811) < view.up\_limit\_id，则当前记录对view可见，即该记录删除可见，查询结果为空；

对于read view1， db\_trx\_id(1811) > view.low\_limit\_id， 则当前记录对view不可见，根据db\_roll\_ptr回溯历史记录，trx\_id=1809，对1809在view中判断可见性，1809通过up\_limit)id和low\_limit\_id不能够判断可见性，则在trx\_ids[]中查询，发现不在trx\_ids中，则该记录可见，返回（1，aaa）。

对于read view2， db\_trx\_id(1811)通过up\_limit\_id和low\_limit\_id不能够判断可见性，则在trx\_ids[]中查询，发现在trx\_ids中，则该记录不可见，返回结果为空。

总结：

* 通过主键查找记录，需要配合read\_view，记录DATA\_TRX\_ID，记录DATA\_ROLL\_PTR指针共同判断。
* read\_view用于判断当前记录是否可见(判断DATA\_TRX\_ID)。DATA\_ROLL\_PTR用于将当前记录回滚到前一版本。

1. **查询二级索引**

*select \* from test where comment = 'aaa';*

查询的结果和当前查询所属的read view 的值有关系，read view值不同，产生的结果也不同，此时二级索引中的MAX\_TRX\_ID派上了用场

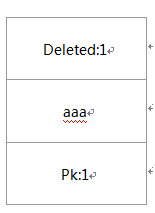


对于read view0， MAX\_TRX\_ID(1816)<view.up\_limit\_id，则全部可见，过滤记录中的有效记录，返回结果为空；

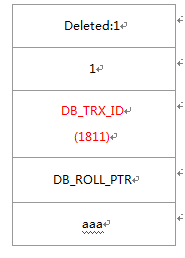
对于read view2， MAX\_TRX\_ID(1816)根据up\_limit\_id和low\_limit\_id不能判断

其可见性，则依次遍历每一条记录来判断可见性

1. 对于第一条记录



反查到聚簇索引记录

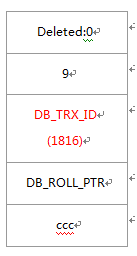


，对该调记录判断其可见性，1811通过low\_limit\_id和up\_limit\_id判断不了，且不在trx\_ids中，则该条记录可见，即aaa已经删除；

1. 对于第二条记录

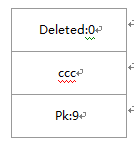


反查到聚簇索引记录：

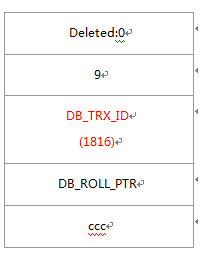


对该调记录判断其可见性，1816通过low\_limit\_id和up\_limit\_id判断不了，且在trx\_ids中，则该条记录不可见，通过db\_roll\_ptr进行判断，1811通过low\_limit\_id和up\_limit\_id判断不了，且不在trx\_ids中，则该条记录可见，返回（9，aaa）。

1. 对于第三条记录



反查到聚簇索引记录



对该调记录判断其可见性，1816通过low\_limit\_id和up\_limit\_id判断不了，且在trx\_ids中，则该条记录不可见，通过db\_roll\_ptr进行判断，1811通过low\_limit\_id和up\_limit\_id判断不了，且不在trx\_ids中，则该条记录可见，值为aaa，同时发现该值aaa和二级索引中的值ccc不相同，则该条记录不返回。

综上所述最后返回（9，aaa）

总结：二级索引的可见性判断需要通过聚簇索引来完成；

1. **总结**

辅助索引查询的MVCC判断如下：

* 如果该Key的整个辅助索引记录的最大事务max\_trx\_id小于当前活跃最老事务up\_limit\_id，说明整个索引都对当前read view可见；
* 否则需要依次对索引中的每条记录回表找到聚簇索引行数据，以判断其可见性；
* 在返回前判断当前可见记录的辅助索引Key是否与回表时辅助索引key相同，如果不同则说明数据不对应，不返回。

所以我们在对**事务并发**的情况下通过**辅助索引查询**，尽管我们在select中返回字段仅有辅助索引和聚簇索引（覆盖），但是其read view的构建（无论是RR还是RC）仍然可能需要**回表查询聚簇索引**，所以事务并发情况下尽量使用聚簇索引查询，否则使用辅助索引可能回表，效率较低。

**参考**

1. 关于InnoDB中mvcc和覆盖索引查询的困惑？<https://www.zhihu.com/question/27674363>