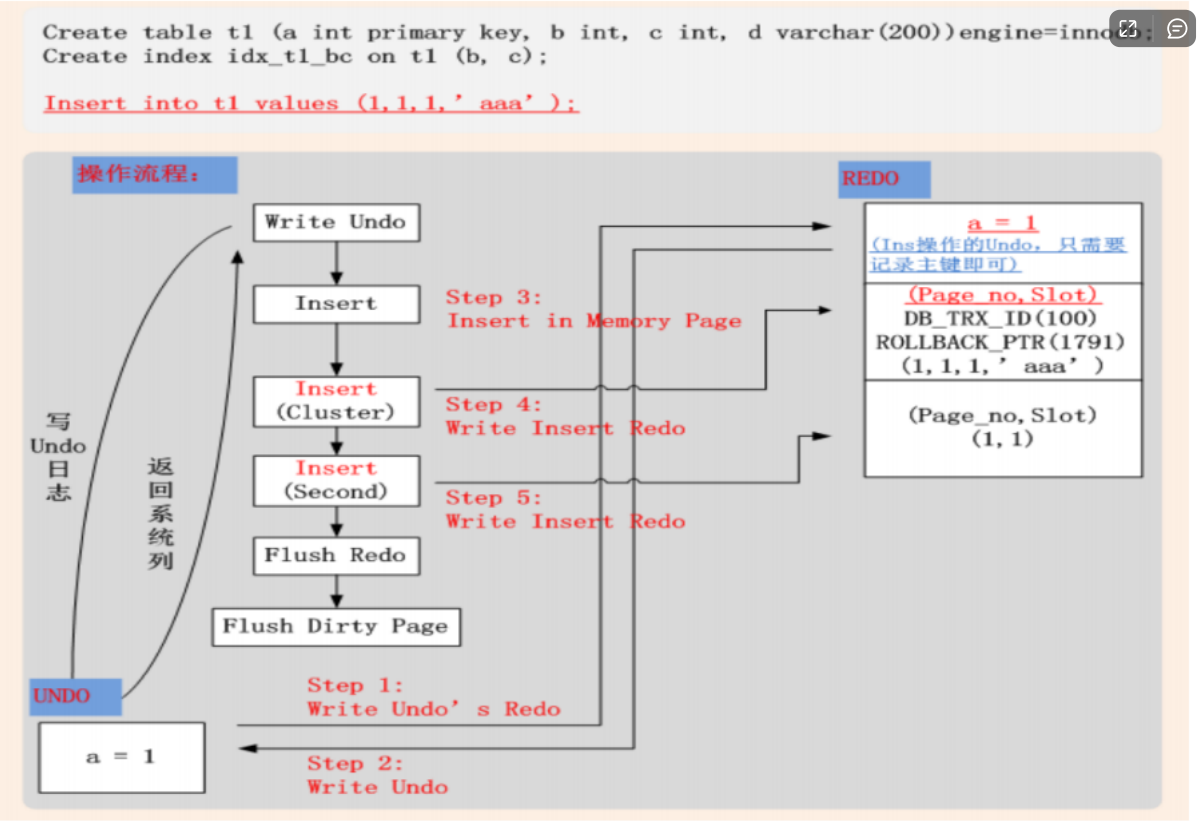
20.MySQL-InnoDB中Undo，Redo在i，u，d操作时的如何工作√

1. Insert

- Undo
  - 将插入记录的主键值，写入Undo；
- Redo
  - 将[space_id, page_no, 完整插入记录, 系统列, ...]写入Redo；
  - space_id, page_no 组合代表了日志操作的页面；



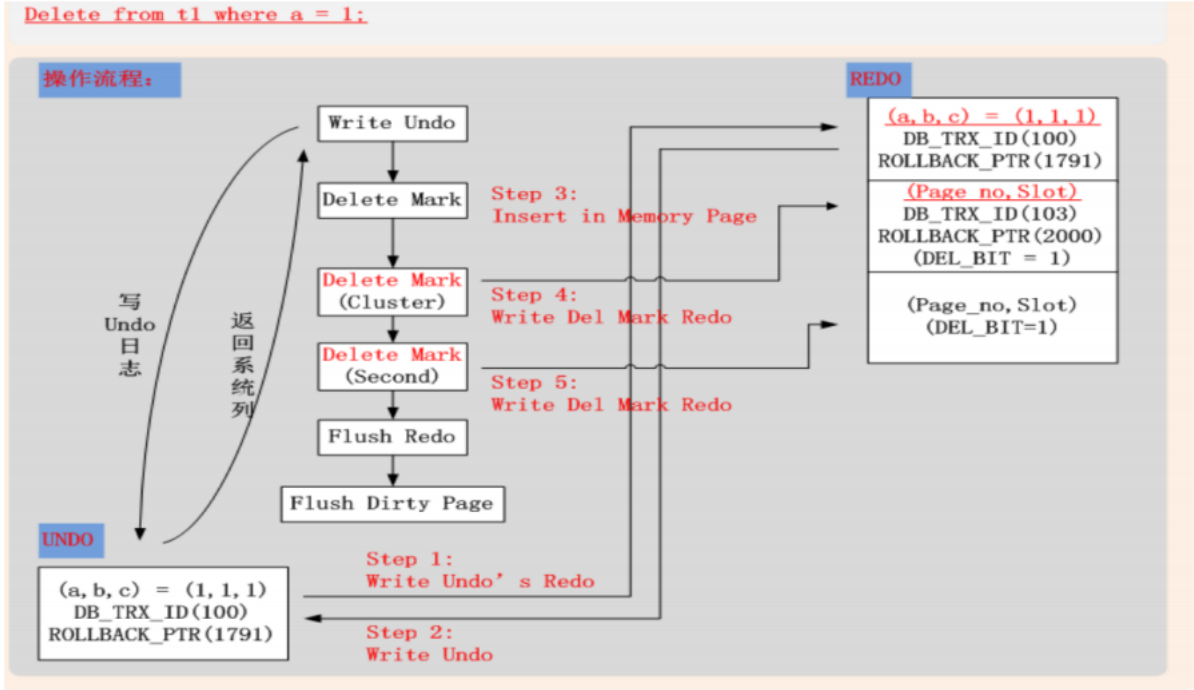
2. DELETE

- Undo

- 1. Delete，在InnoDB内部为Delete Mark操作，将记录上标识Delete_Bit，而不删除记录；
- 2. 将当前记录的系统列写入Undo (DB_TRX_ID, ROLLBACK_PTR, ...)；
- 3. 将当前记录的主键列写入Undo；
- 4. 将当前记录的所有索引列写入Undo (why? for what?)；
- 5. 将Undo Page的修改，写入Redo；

— Redo

- 将[space_id, page_no, 系统列, 记录在页面中的Slot, ...]写入Redo；



3.UPDATE

3.1 情况一：Update(未修改聚簇索引键值，属性列长度未变化)

— Undo (聚簇索引)

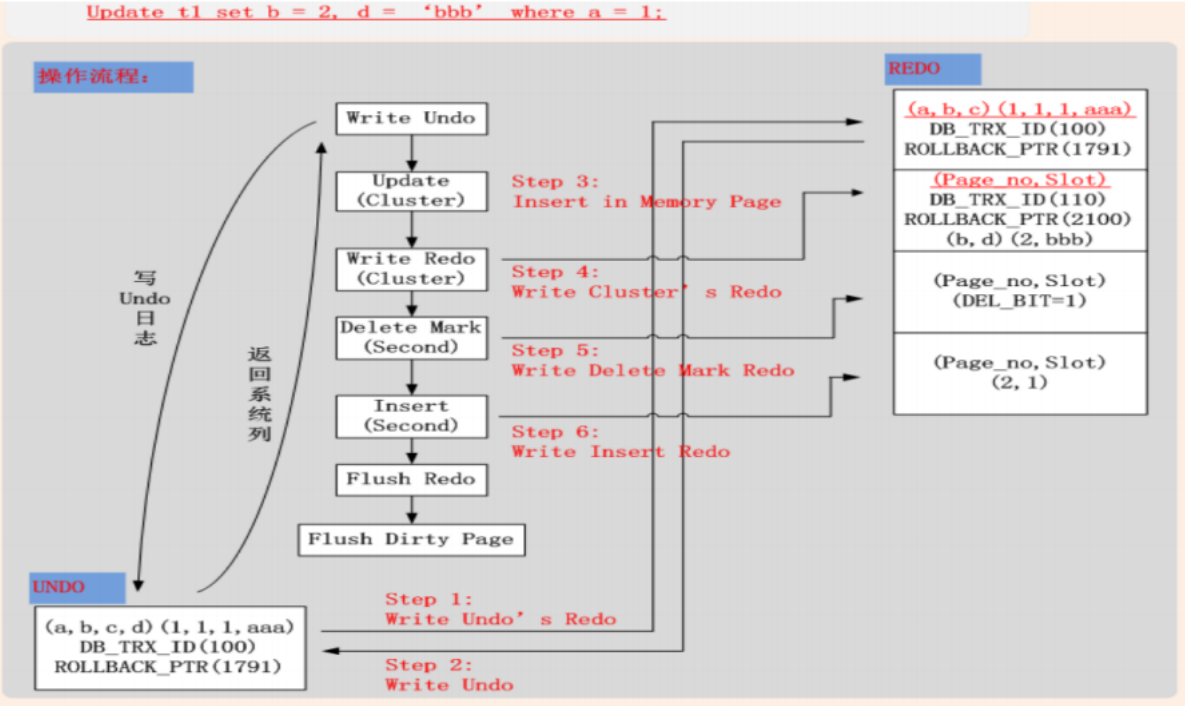
- 1. 将当前记录的系统列写入Undo (DB_TRX_ID, ROLLBACK_PTR, ...)；
- 2. 将当前记录的主键列写入Undo；
- 3. 将当前Update列的前镜像写入Undo；
- 4. 若Update列中包含二级索引列，则将二级索引其他未修改列写入Undo；

- 5. 将Undo页面的修改，写入Redo；

— Redo

- 进行In Place Update，记录Update Redo日志(聚簇索引)；
- 若更新列包含二级索引列，二级索引肯定不能进行In Place Update，记录Delete Mark +

Insert Redo日志；



3.2 情况二： Update(未修改聚簇索引键值，属性列长度发生变化)

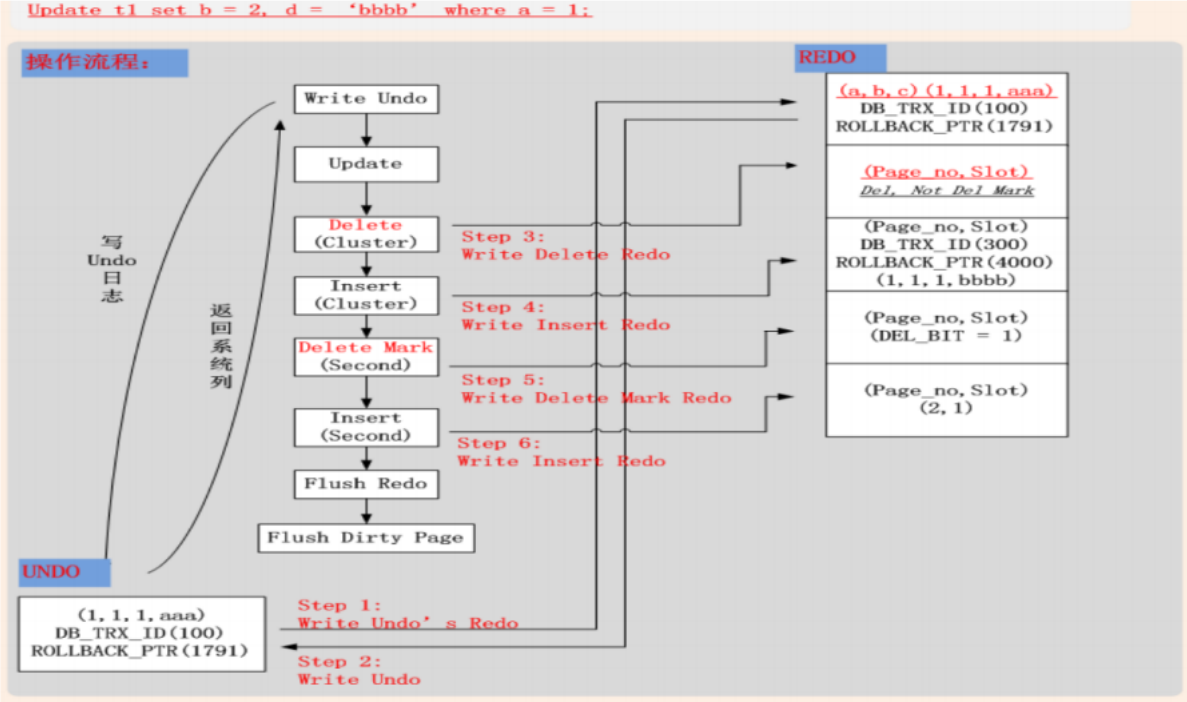
— Undo (聚簇索引)

- 1. 将当前记录的系统列写入Undo (DB_TRX_ID, ROLLBACK_PTR, ...)；
- 2. 将当前记录的主键列写入Undo；
- 3. 将当前Update列的前镜像写入Undo；
- 4. 若Update列中包含二级索引列，则将二级索引其他未修改列写入Undo；
- 5. 将Undo页面的修改，写入Redo；

— Redo

- 不可进行In Place Update，记录Delete + Insert Redo日志(聚簇索引)；
- 若更新列包含二级索引列，二级索引肯定不能进行In Place Update，记录Delete Mark +

Insert Redo日志；



3.3 情况三：Update(修改聚簇索引键值)

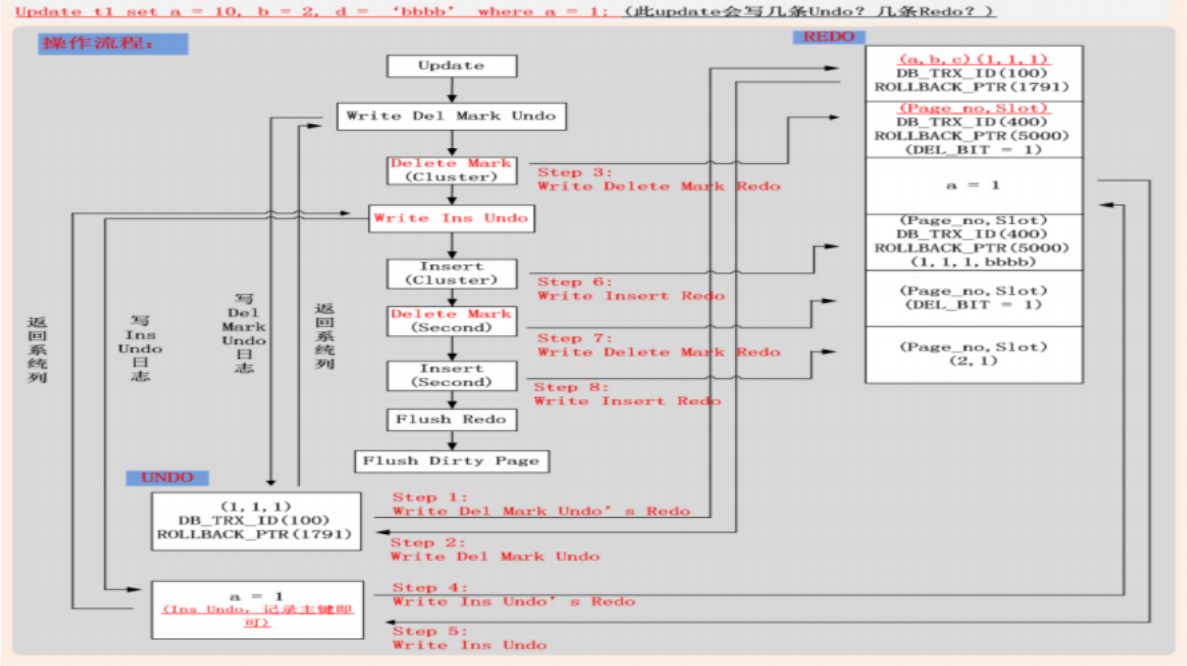
— Undo (聚簇索引)

- 1. 不可进行In Place Update。Update = Delete Mark + Insert；
- 2. 对原有记录进行Delete Mark操作，写入Delete Mark操作Undo；
- 3. 将新纪录插入聚簇索引，写入Insert操作Undo；
- 4. 将Undo页面的修改，写入Redo；

— Redo

- 不可进行In Place Update，记录Delete Mark + Insert Redo日志(聚簇索引)；
- 若更新列包含二级索引列，二级索引肯定不能进行In Place Update，记录Delete Mark +

Insert Redo日志；



InnoDB%E4%B8%ADUndo%EF%BC%8CRedo%E5%9C%A8i%EF%BC%8Cu%EF%BC%8Cd%E6%93%8D%E4%BD%9C%E6%97%B6%E7%9A%84%E5%A6%82%E4%BD%95%E5%B7%A5%E4%BD%9C%E2%8