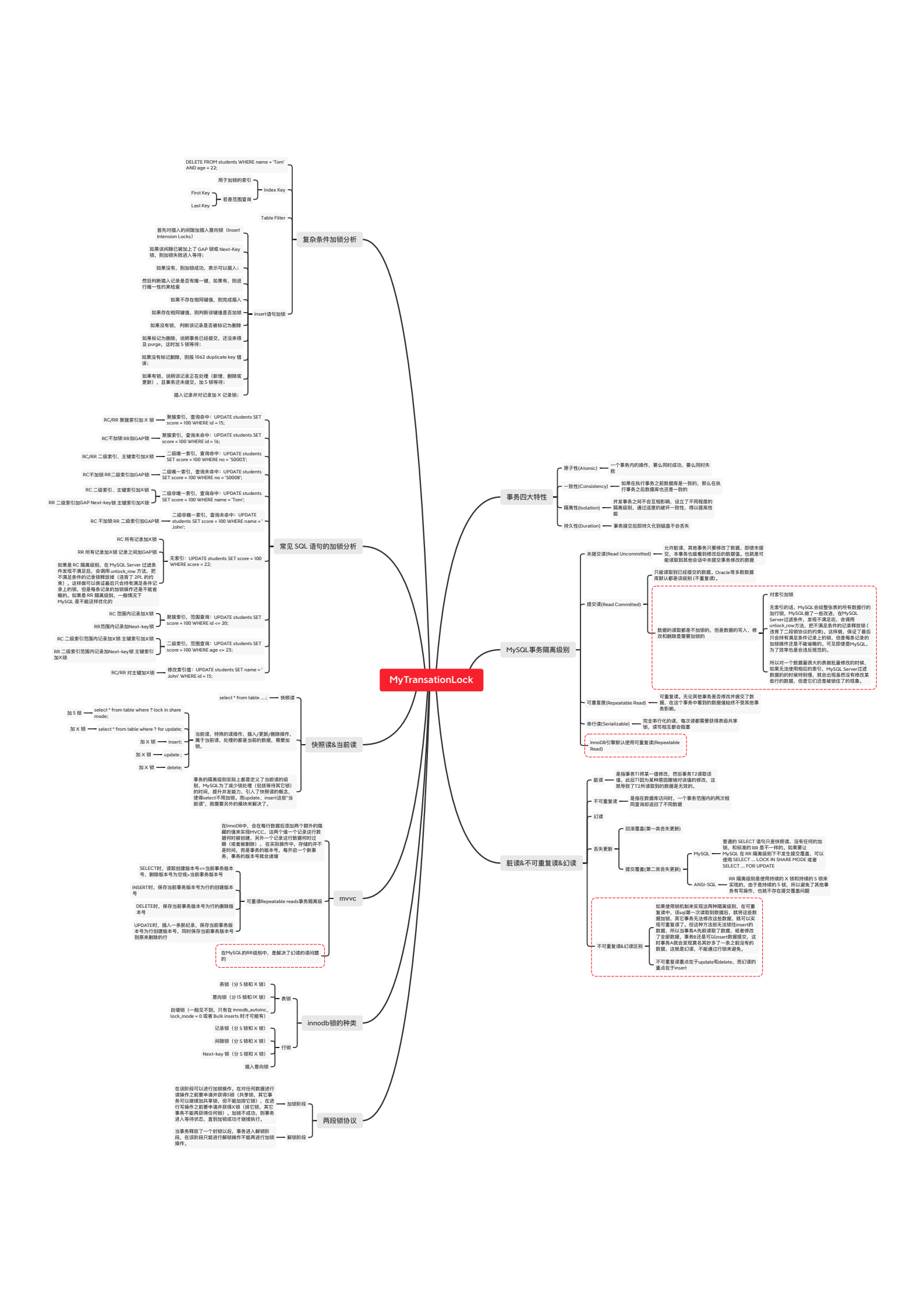
MyTransationLock
复杂条件加锁分析
DELETE FROM students WHERE name = 'Tom' AND age = 22;
用于加锁的索引
Index Key
First Key
Last Key
若是范围查询
Table Filter
首先对插入的间隙加插入意向锁（Insert Intension Locks）
如果该间隙已被加上了 GAP 锁或 Next-Key 锁，则加锁失败进入等待；
如果没有，则加锁成功，表示可以插入；
然后判断插入记录是否有唯一键，如果有，则进行唯一性的索检查
如果不存在相同键值，则完成插入
insert语句加锁
如果存在相同键值，则判断该键值是否加锁
如果没有锁，判断该记录是否被标记为删除
如果标记为删除，说明事务已经提交，还没来得及 purge，这时加 S 锁等待；
如果没有标记删除，则报 1062 duplicate key 错误；
如果有锁，说明该记录正在处理（新增、删除或更新），且事务还未提交，加 S 锁等待；
插入记录并对记录加 X 记录锁；
常见 SQL 语句的加锁分析
聚簇索引，查询命中：UPDATE students SET score = 100 WHERE id = 15;
RC/RR 聚簇索引加 X 锁
聚簇索引，查询未命中：UPDATE students SET score = 100 WHERE id = 16;
RC不加锁 RR加GAP锁
二级唯一索引，查询命中：UPDATE students SET score = 100 WHERE no = 'S0003';
RC/RR 二级索引、主键索引加X锁
二级唯一索引，查询未命中：UPDATE students SET score = 100 WHERE no = 'S0008';
RC不加锁 RR二级索引加GAP锁
RC 二级索引、主键索引加X锁
二级非唯一索引，查询命中：UPDATE students SET score = 100 WHERE name = 'Tom';
RR 二级索引加GAP Next-key锁 主键索引加X锁
二级非唯一索引，查询未命中：UPDATE students SET score = 100 WHERE name = 'John';
RC 不加锁 RR 二级索引加GAP锁
RC 所有记录加X锁
RR 所有记录加X锁 记录之间加GAP锁
无索引：UPDATE students SET score = 100 WHERE score = 22;
如果是 RC 隔离级别，在 MySQL Server 过滤条件发现不满足后，会调用 unlock_row 方法，把不满足条件的记录锁释放掉（违背了 2PL 的约束）。这样做可以保证最后只会持有满足条件记录上的锁，但是每条记录的加锁操作还是不能省略的。如果是 RR 隔离级别，一般情况下 MySQL 是不能这样优化的
RC 范围内记录加X锁
聚簇索引，范围查询：UPDATE students SET score = 100 WHERE id <= 20;
RR范围内记录加Next-key锁
RC 二级索引范围内记录加X锁 主键索引加X锁
二级索引，范围查询：UPDATE students SET score = 100 WHERE age <= 23;
RR 二级索引范围内记录加Next-key锁 主键索引加X锁
RC/RR 对主键加X锁
修改索引值：UPDATE students SET name = 'John' WHERE id = 15;
快照读&当前读
select * from table ....;
快照读
加 S 锁
select * from table where ? lock in share mode;
加 X 锁
select * from table where ? for update;
当前读，特殊的读操作，插入/更新/删除操作，属于当前读，处理的都是当前的数据，需要加锁。
加 X 锁
insert;
加 X 锁
update ;
加 X 锁
delete;
事务的隔离级别实际上都是定义了当前读的级别，MySQL为了减少锁处理（包括等待其它锁）的时间，提升并发能力，引入了快照读的概念，使得select不用加锁。而update、insert这些"当前读"，就需要另外的模块来解决了。
mvvc
在InnoDB中，会在每行数据后添加两个额外的隐藏的值来实现MVCC，这两个值一个记录这行数据何时被创建，另外一个记录这行数据何时过期（或者被删除）。在实际操作中，存储的并不是时间，而是事务的版本号，每开启一个新事务，事务的版本号就会递增
可重读Repeatable reads事务隔离级
SELECT时，读取创建版本号<=当前事务版本号，删除版本号为空或>当前事务版本号
INSERT时，保存当前事务版本号为行的创建版本号
DELETE时，保存当前事务版本号为行的删除版本号
UPDATE时，插入一条新纪录，保存当前事务版本号为行创建版本号，同时保存当前事务版本号到原来删除的行
在MySQL的RR级别中，是解决了幻读的读问题的
innodb锁的种类
表锁
表锁（分 S 锁和 X 锁）
意向锁（分 IS 锁和 IX 锁）
自增锁（一般见不到，只有在 innodb_autoinc_lock_mode = 0 或者 Bulk inserts 时才可能有）
行锁
记录锁（分 S 锁和 X 锁）
间隙锁（分 S 锁和 X 锁）
Next-key 锁（分 S 锁和 X 锁）
插入意向锁
两段锁协议
加锁阶段
在该阶段可以进行加锁操作。在对任何数据进行读操作之前要申请并获得S锁（共享锁，其它事务可以继续加共享锁，但不能加排它锁），在进行写操作之前要申请并获得X锁（排它锁，其它事务不能再获得任何锁），加锁不成功，则事务进入等待状态，直到加锁成功才继续执行。
解锁阶段
当事务释放了一个封锁以后，事务进入解锁阶段，在该阶段只能进行解锁操作不能再进行加锁操作。
事务四大特性
原子性(Atomic)
一个事务内的操作，要么同时成功，要么同时失败
一致性(Consistency)
如果在执行事务之前数据库是一致的，那么在执行事务之后数据库也还是一致的
隔离性(Isolation)
并发事务之间不会互相影响，设立了不同程度的隔离级别，通过适度的破坏一致性，得以提高性能
持久性(Duration)
事务提交后即持久化到磁盘不会丢失
MySQL事务隔离级别
未提交读(Read Uncommitted)
允许脏读，其他事务只要修改了数据，即使未提交，本事务也能看到修改后的数据值，也就是可能读取到其他会话中未提交事务修改的数据
提交读(Read Committed)
只能读取到已经提交的数据。Oracle等多数数据库默认都是该级别 (不重复读)。
数据的读取都是不加锁的，但是数据的写入、修改和删除是需要加锁的
对索引加锁
无索引的话，MySQL会给整张表的所有数据行的加行锁，MySQL做了一些改进，在MySQL Server过滤条件，发现不满足后，会调用unlock_row方法，把不满足条件记录上的锁。但是每条记录的加锁操作还是不能省略的。可见即使是MySQL，为了效率也是会违反规范的。
所以对一个数据量很大的表做批量修改的时候，如果无法使用相应的索引，MySQL Server过滤数据的的时候特别慢，就会出现虽然没有修改某些行的数据，但是它们还是被锁住了的现象。
可重复度(Repeatable Read)
可重复读。无论其他事务是否修改并提交了数据，在这个事务中看到的数据值始终不受其他事务影响。
串行读(Serializable)
完全串行化的读，每次读都需要获得表级共享锁，读写相互都会阻塞
InnoDB引擎默认使用可重复读(Repeatable Read)
脏读&不可重复读&幻读
脏读
是指事务T1将某一值修改，然后事务T2读取该值，此后T1因为某种原因撤销对该值的修改，这就导致了T2所读取到的数据是无效的。
不可重复读
是指在数据库访问时，一个事务范围内的两次相同查询却返回了不同数据
幻读
丢失更新
回滚覆盖(第一类丢失更新)
提交覆盖(第二类丢失更新)
MySQL
普通的 SELECT 语句只是快照读，没有任何的加锁，和标准的 RR 是不一样的。如果要让 MySQL 在 RR 隔离级别下不发生提交覆盖，可以使用 SELECT ... LOCK IN SHARE MODE 或者 SELECT ... FOR UPDATE
ANSI-SQL
RR 隔离级别是使用持续的 X 锁和持续的 S 锁来实现的，由于是持续的 S 锁，所以避免了其他事务有写操作，也就不存在提交覆盖问题
不可重复读&幻读区别
如果使用锁机制来实现这两种隔离级别，在可重复读中，该sql第一次读取到数据后，就将这些数据加锁，其它事务无法修改这些数据，就可以实现可重复读了。但这种方法却无法锁住insert的数据，所以当事务A先前读取了数据，或者修改了全部数据，事务B还是可以insert数据提交，这时事务A就会发现莫名其妙多了一条之前没有的数据，这就是幻读，不能通过行锁来避免。
不可重复读重点在于update和delete，而幻读的重点在于insert