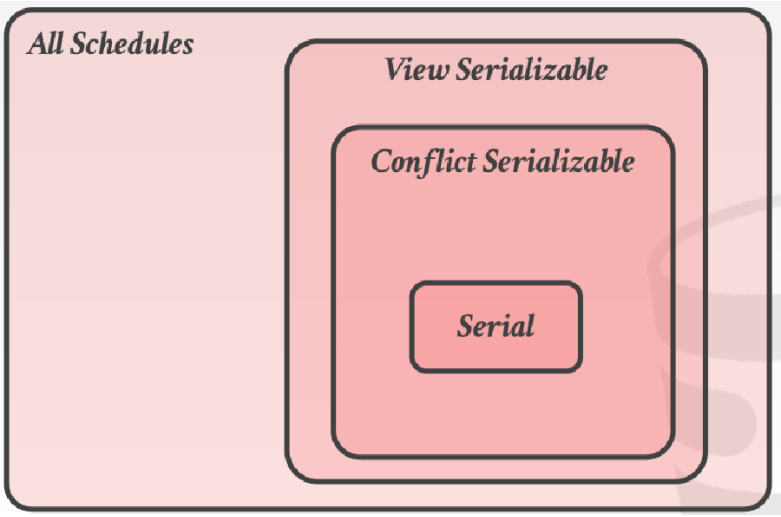
11 事务

- 要点
  - Concept of transaction and ACID　ACID 原子性、一致性、隔离性、持久性
  - Relation between conflict serializability and view serializability 冲突可串行化（比较重要），视图可串行化（知道概念即可）冲突可串行化是视图可串行化的一个子集
- 基本概念
  - **事务ACID特性**
    - 原子性：要么全部做完，要么不做
    - 一致性：保持数据库一致性
    - 隔离性：多个事务并发执行，但事务自己感觉不到
    - 持久性：完成之后它对数据库的改变是永久的
  - 事务的状态
    - 活动的
    - 部分提交的
    - 失败的
    - 中止的
    - 提交的
- 事务调度



  - 串行调度
  - 并发调度
    - 问题：可能破坏数据一致性，即使单个事务的执行保证一致性，但多个并发调度也会有可能破坏
    - 问题原因：事务隔离性未得到保证，并发不能保证事务可以看到另一事务对数据课的全部修改
    - 导致三类数据不一致性

        - 丢失修改
            - 两个事务从数据库读入同一个数据之后分别进行修改，先后提交，后提交的覆盖了先提交的修改
        - 不可重复读
            - 事务1读取某些数据，事务2读取并修改（更新插入删除），当事务1再去读取时，得到的和之前的不一样
            - 也就是说，在一个事务读取两次同样数据中间，另一事务对这个数据进行了修改
        - 读“脏”数据
            - 事务1修改某数据并写入，但尚未提交
            - 事务2读取同一数据，得到的是**新值**
            - 但事务1提交失败，数据库中的数据恢复到旧值
            - 事务2读到的数据和数据库最终数据不一致
- 可串行化
    - 假设：每个事物都保持数据库一致性
    - 如果一种调度等同于串行调度，则它是可串行化的
    - 不同形式的调度产生
        - 冲突可串行化
        - 查看可串行化
    - 简化事务视图——忽略读写之外的指令
    - 冲突可串行化
        - 不同事务在相同数据项上操作，并且有至少一个是write指令，我们说两个事务冲突
        - 若两条指令I和J是某调度S的两条连续指令。若I和J是属于不同事务的指令且不冲突，则可以交换I和J的顺序而得到一个新的调度
        - 不断进行等价交换之后，变成了一个串行调度，那么原调度就是冲突可串行化的
    - 简单方法确定一个调度是否冲突可串行化——优先图
        - 顶点集是所有参与调度的事务，边集是由满足下列三个条件之一的边$T_i->T_j$组成，T表示的是事务吧
            - 在T_j执行read之前，T_i执行write
            - 在T_j执行write之前，执行read/write
        - 若优先图中存在边$T_i->T_j$，则在任何等价调度中，T_i必须出现在T_j之前
        - 如果优先图中有换，则该调度是非冲突可串行化的；如果**无环**，则是冲突可串行化的
- 事务隔离性和原子性
    - 可恢复调度
        - 对于每对事务，如果T_j读取了之前由T_i所写的数据项，则T_i先于T_j提交

    - 无级联调度
        - 对于每对事务，如果T_j读取了之前由T_i所写的数据项，则T_i必须在T_j读取操作之前就提交
- 事务隔离级别
    - 并发控制
        - 目标——开发可确保可串行化的并发控制协议。
    - 隔离级别
        - 可串行化
        - 可重复读：只允许读已提交的数据，在一个事务两次读取一个数据时，其他事务不可以更改数据
        - 已提交读：只允许读已提交数据，但不要求可以重复读，……期间其他事务可以更新数据
        - 未提交读：允许读取未提交的数据，最低
        - 所有都不允许脏写
- 隔离性级别的实现
    - 锁
    - 时间戳
    - 多版本和快照隔离

以上内容整理于 幕布文档