主题五：恢复系统

1. 背景(Background)

数据库系统在运行的过程中难免会遇到故障。为了保证事务在此情况下的原子性、持久性和一致性，数据库系统需要恢复算法来实现此目标。

2. 故障简介(Crash Overview)

数据库管理系统的子系统相当复杂，因而可能发生故障的地方也会很多。其中，有的故障是可恢复的，而有的故障可发现却不可恢复。常见的故障分类如下：

- 事务故障：当事务遇到错误而必须中止时而产生。分为逻辑错误或者内部状态错误两种情况。
    - 逻辑错误：事务不满足限制。例如，非NULL限制或者check contraint。
    - 内部状态错误：死锁发生，锁管理器要求事务中止。
- 系统故障：当系统的硬件或软件发生故障而产生。
    - 软件错误：可能是数据库系统实现错误。例如，未捕获除0异常。
    - 硬件错误：例如，断电。
- 存储介质故障：当存储介质受损而发生的故障。它是不可恢复但可发现的故障。需要人工干预，可能从数据库备份中恢复。

3. 缓冲池管理策略(Buffer Pool Management Policy)

缓冲池管理策略规定了缓冲池管理器在什么时候将页强制写入磁盘和允许事务是否在未提交时将页写入磁盘

- 偷窃策略(Steal Policy)：是否允许事务在其未提交时覆写非易失性存储器上的一个对象的最新提交值。换言之，是否允许事务把另一个还未提交的事务所做的改变反映到磁盘上。
- 强制(Force Policy)：是否要求事务在提交前将所有的更新反应到磁盘上。

在非偷窃、强制的缓冲池管理策略下，

---

4. 日志(Log)

日志提供了恢复系统在恢复阶段判别事务是否需要撤销或重做的必要信息。在正常执行时，事务在执行更新操作前先写一个更新的日志项。在事务提交时，所有的日志项都应该被写入日志文件。这样的技术称为先写日志*(Write-Ahead Logging)*。先写日志保证了事务的原子性和持久性。日志的一个可能格式是：`<Transaction ID,Object ID,Before Value, After Value>`。

在不同的实现中，日志项的内容可能不同。有以下日志策略。

- 物理日志(Physical Logging)：日志项记录了对象的字节位置。例如，`Tansaction ID=1,Table = A,Page = 1,Offset=500,Before=AAA,After=BBB`
- 逻辑日志(Logical Logging)：日志项记录高级的操作。例如，`Query = UPDATE students SET grade=100 WHERE name='Hoo'`
- 物理逻辑日志(Physiological Logging)：综合了物理日志和逻辑日志。但不表现出页的组织。例如，`Transaction ID=1,Table=A,Page=1,Slot=5,Before=XXX,After=YYY`

---

5. 检查点(Check Point)

数据库系统在运行时周期性地使用检查点来记录其状态。通过使用检查点，数据库系统可以减少恢复阶段所需要的时间和日志存储所需要的空间。

在简单的检查点实现中，系统等待所有进行中的事务完成、阻止新的事务开始、将所有的脏页刷新到磁盘上。因为该实现需要阻塞系统运行，脏页很多的情况下，多系统造成可观的性能损失。一个性能更高的检查点实现是模糊检查点*(Fuzzy Checkpoint)*，在模糊检查点中，检查点有两个数据结构：DPT(Dirty Page Table)和ATT(Active Transaction Table)。

- 脏页表(Dirty Page Table)：记录了在检查点处的脏页。每个数据项的格式是：`<PageID>,<RecLSN>`，其中RecLSN指的是自从页被刷新以后最久的日志序号。
- 活跃事务表(Active Transaction Table)：记录了在检查点出活跃的事务。每个数据项的格式是：`<TransactionID>,<Status>,<LastLSN>`，其中LastLSN指的是事务最近的日志序号。

> 为了保证状态机在发生故障后能够恢复，系统需要在状态机正常运行时使用日志来记录其更新。但随着系统运行时间的增长，如果不加以额外处理，日志会越来越多，甚至会超过物理限制。因而我们需要在适当的时候进行日志压缩。数据库系统使用检查点来实现这一目标，而Raft中使用快照来实现。

6. ARIES恢复算法(ARIES Recovery Algorithm)

ARIES恢复算法是基于非偷窃、强制的缓冲区管理策略，使用先写日志技术的恢复算法。其被IBM DB2、MS SQL等数据库系统使用。ARIES的三个关键点：

- 先写日志：对对象的任何改变都应该先记录在日志中，在对象的改变被写入到磁盘前，日志必须要被写入到稳定存储器中。
- 在重做阶段重复历史：在故障后重启时，重新执行故障前的操作并且让数据库系统回到与故障发生前的系统一致的状态。然后撤销在故障发生时仍然活跃的事务所做的改变。
- 在撤销阶段记录改变：在撤销事务时，记录对数据库所做的操作并且确保该操作不会被继续执行。

6.1. 数据结构(Data Structure)

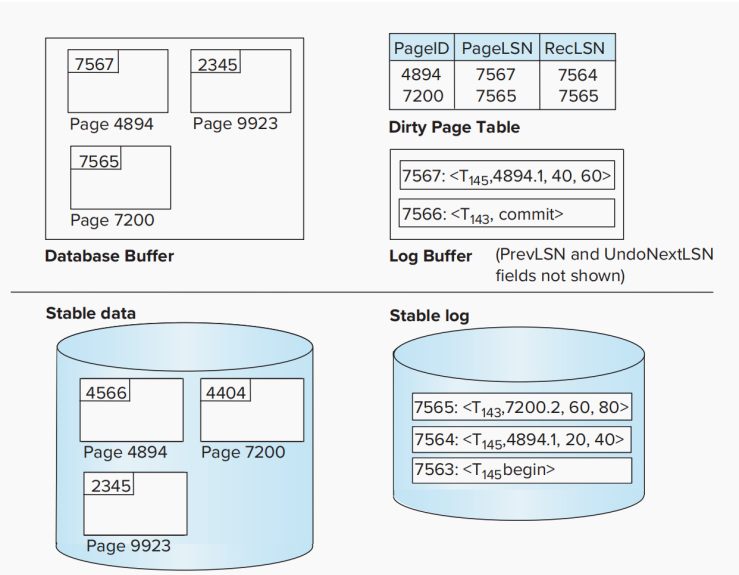
检查点：模糊检查点。

每个日志都有一个日志序号*(Log Sequence Number)* 并且还存储着PreLSN（同一事务的前一个日志序号）。

在撤销事务时，为撤销的动作记录补偿日志记录*(Compensation Log Record)*。补偿日志记录与一般的日志记录相比，多了一个UndoNext字段，该字段存储着下一个需要被撤销的日志序号。

主检查点日志序号(Master Checkpoint LSN)：最后一个检查点的日志序号。

每个页存储着一个PageLSN字段和RecLSN字段。PageLSN存储着该页的序号，RecLSN字段存储着最新的更新该页的日志项的日志序号。



6.2. 算法(Algorithm)

ARIES恢复算法在恢复时一共有三个阶段：分析、重做和撤销。

6.2.1. 分析阶段(Analyze Phase)

从数据库记录的主检查点开始。

1. 从检查点开始向后扫描。
2. 如果遇到< TXN-END> 就将该事务从ATT中移除。
3. 如果遇到< TXN-BEGIN > 就将该事务加入到ATT中并且标记状态为UNDO。如果遇到< TXN-COMMIT > 就把该事务在ATT中的状态设置为提交。
4. 如果遇到对页P的更新并且P不在DPT中，那么就将P加入DPT。并且改变P的RecLSN。

6.2.2. 重做阶段(Redo Phase)

该阶段重做所有更新以及CLR。从DPT中最小的RecLSN开始扫描。重新应用每一个日志项记录LogEntry的更新并且更新受影响页P的RecLSN，除非：

1. P不在DPT中，或者
2. P在DPT中，但LogEntry.LSN < P.RecLSN，或者
3. P在磁盘上的RecLSN > LogEntry.LSN。

一旦重做阶段完成，对于ATT中标记为COMMIT的事务，写入<TXN-END> 日志项并且将其从ATT中移除。

上述不重做操作限制的目的是为了避免不必要的执行。

6.2.3. 撤销阶段(Undo Phase)

该阶段的ATT中每一个事务都被撤销。从每个事务的LastLSN开始，在每个日志项的PreLSN字段帮助下反向撤销并且为撤销记录下CLR。

撤销阶段完成后，把日志缓冲区中的日志项写入稳定存储器中，然后开始数据库系统的正常运行。