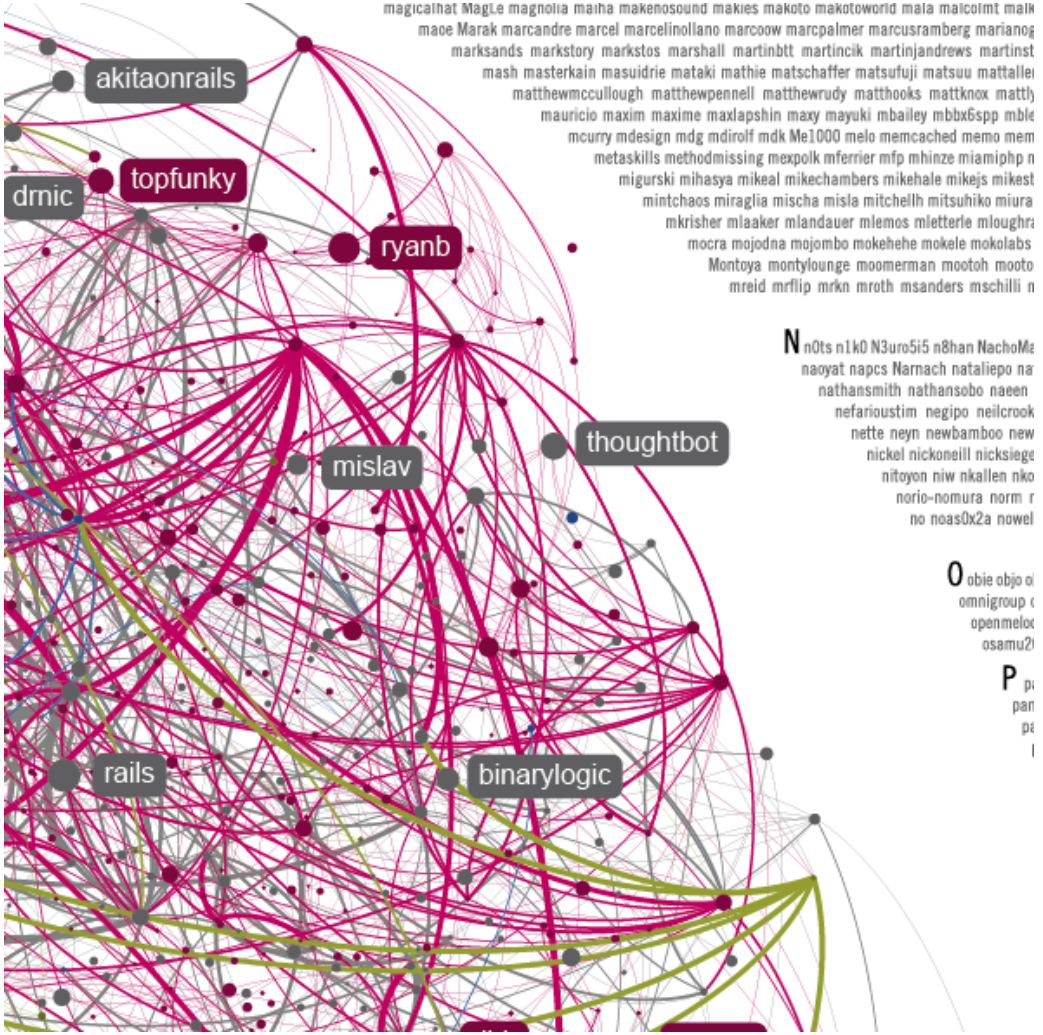
DATAGURU 炼数成金

DB2设计与性能优化

第5周



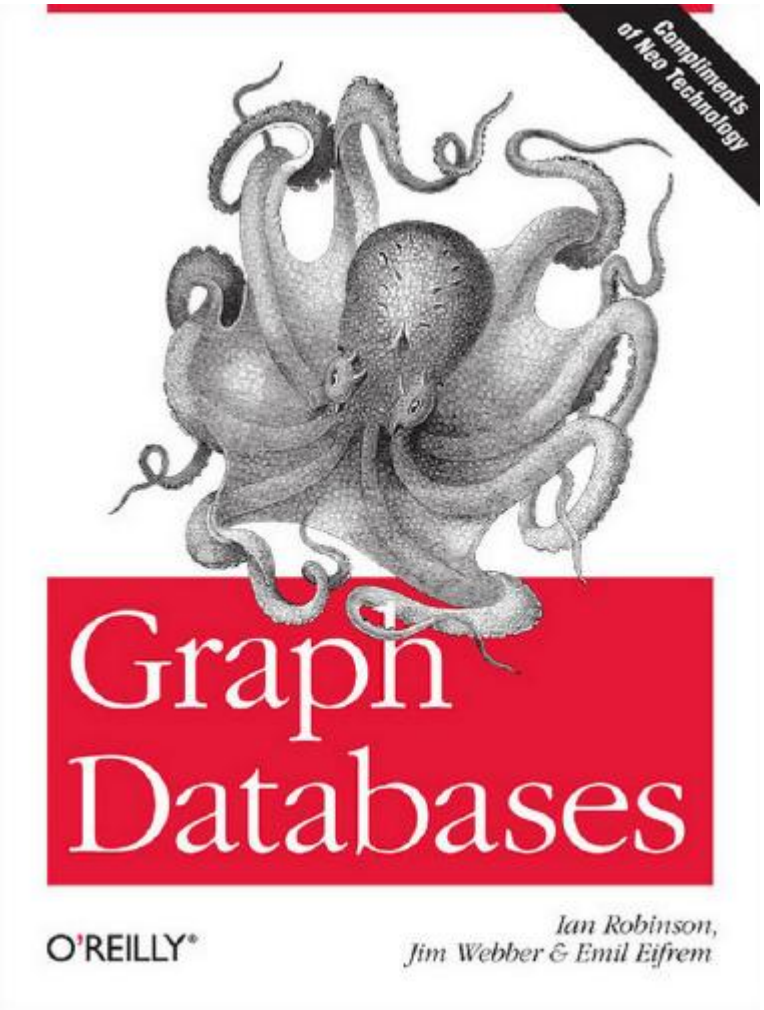
法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

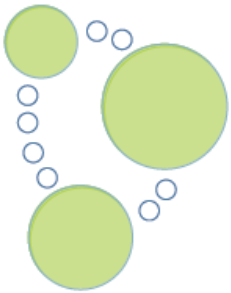
课程详情访问炼数成金培训网站

http://edu.dataguru.cn

图数据库参考书



The Neo4j Manual



neo technology
value in relationships

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

提纲

- 日志优化应知应会三原则
- DB2日志原理
- 参数配置与性能
- 性能监控与优化
- 小结与练习

日志优化应知应会三原则

- 日志先写原则
  - 日志是事务处理和恢复的必备机制
  - 日志记录每一行数据的变化
  - 写入磁盘顺序：日志为先，数据在后
- 日志与数据分离原则
  - 数据和日志是数据库处理必不可少的两部分
  - 为防止数据和日志磁盘同时损害而导致数据丢失
  - 为提高性能，防止IO竞争
- 日志最优待遇原则（VIP原则）
  - 日志性能对数据库系统的整体性能至关重要
  - 在每次事务提交时都必须将日志写入到磁盘
  - 给日志配置独立的物理磁盘，性能最好的磁盘

提纲

- 日志优化应知应会三原则
- DB2日志原理
- 参数配置与性能
- 性能监控与优化
- 小结与练习

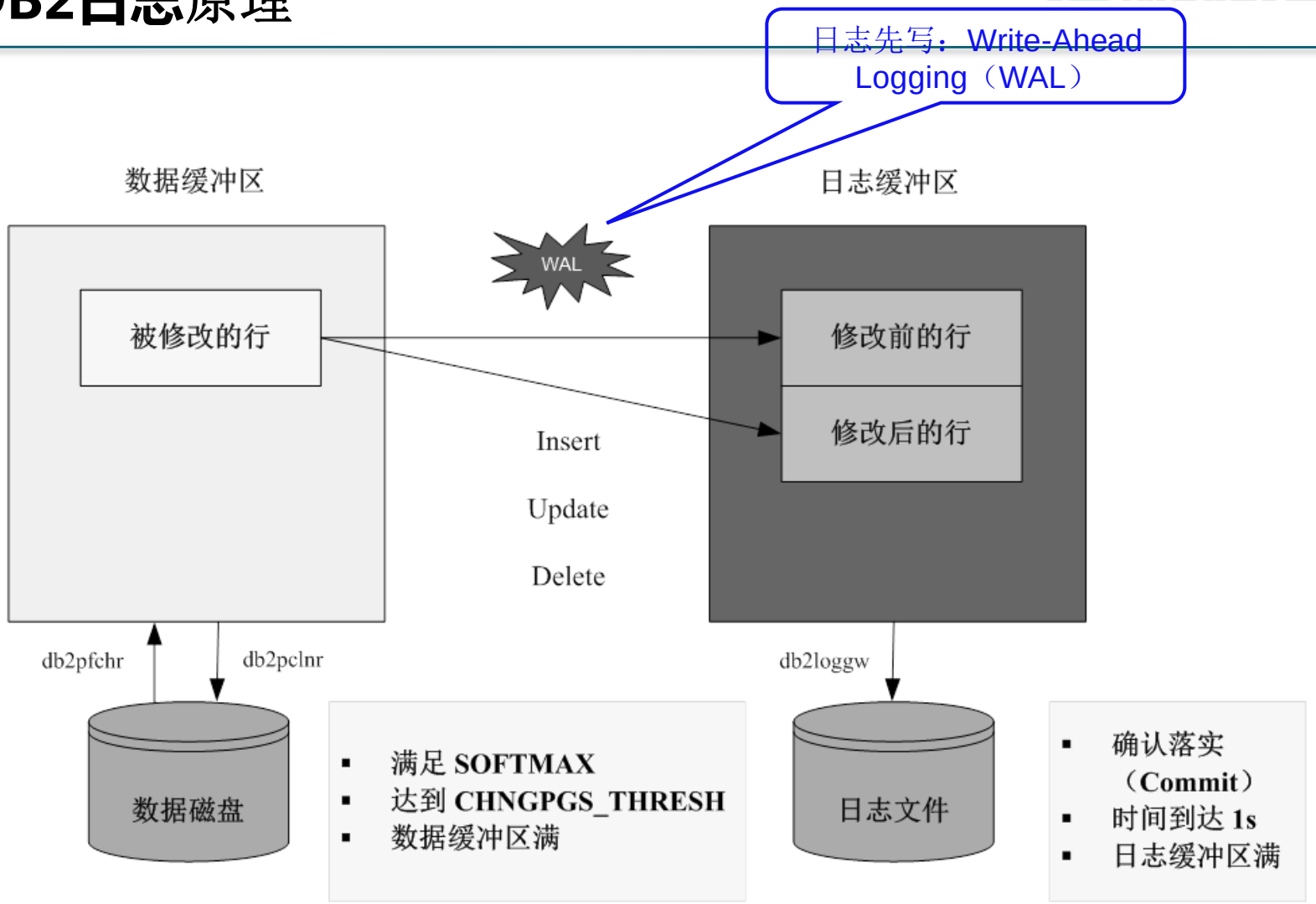
日志的基础知识

- 日志是数据库事务和数据库恢复的必备机制
  - 不论Oracle还是DB2，日志必不可少
  - 日志记录了数据库中数据的变化(插入、更新或删除)情况
  - 日志记录了事务的开始、Commit和Rollback事件
- 日志的作用
  - 用于系统崩溃的事务（或数据）一致性恢复
  - 系统灾难（如磁盘损坏）时，结合数据库备份并利用日志前滚将数据库恢复到当前状态
- 日志记录首先会被写入日志缓冲区，然后被写入日志文件:
  - 日志缓冲区满了
  - 当事务提交时，与该事务相关的日志会被写入日志文件
  - 数据页被写入磁盘 (为了保证在数据库在出问题时，改变的数据可以被会滚)

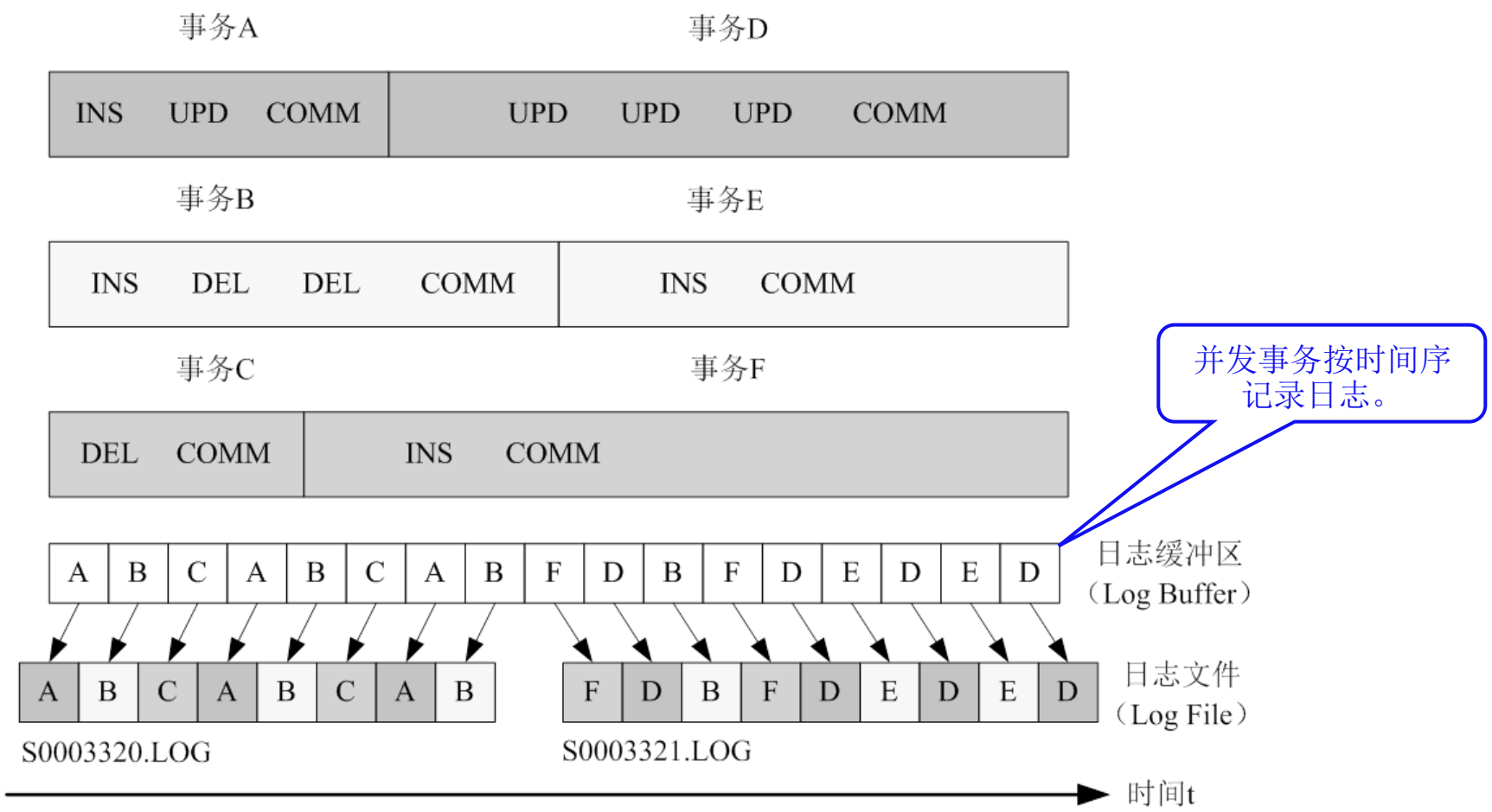
日志的基础知识(续)

- 当日志记录被写入日志文件时，需要保留额外的空间以防“回滚”事务发生
  - “Undo” 日志记录在回滚时被记入日志
  - 所需的空间大小与 “redo” 日志大小基本相同或略小
- 日志记录被写入日志文件
  - 每个日志文件的头两个页用于存放元数据（metadata ）
  - 其余的页用于存放日志记录
- 日志文件的命名以 S0000000.LOG开始

DB2日志原理


日志先写：Write-Ahead Logging（WAL）
数据缓冲区
日志缓冲区
WAL
被修改的行
修改前的行
修改后的行
Insert
Update
Delete
db2pfchr
db2pclnr
db2loggw
数据磁盘
日志文件
满足 SOFTMAX
达到 CHNGPGS_THRESH
数据缓冲区满
确认落实（Commit）
时间到达 1s
日志缓冲区满

DB2日志缓冲区与文件

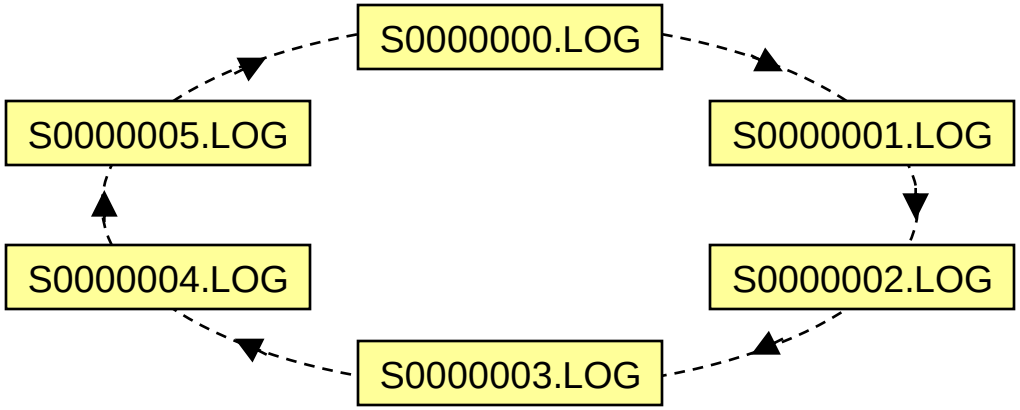


从日志缓冲区到磁盘文件

- 日志缓冲区
  - 日志记录先保存在缓冲区，在必要时写入磁盘
  - 日志记录写入程序（db2loggw）将日志从缓冲区写入到日志文件。
- 日志写入到磁盘的时机（WAL）
  - 当前事务被落实（Commit）时，此事务的日志必须写入到日志文件
  - 日志缓冲区满了，缓冲区中的日志会立即写入到磁盘
  - 数据缓冲区满时，脏数据页被写到磁盘前，相关的日志必须先写入磁盘
  - 每隔一秒，日志缓冲区也会将自身的日志写入到磁盘
- 日志写入磁盘后，数据何时写入磁盘与该日志无直接联系
- DB2支持两种方式的日志模式:

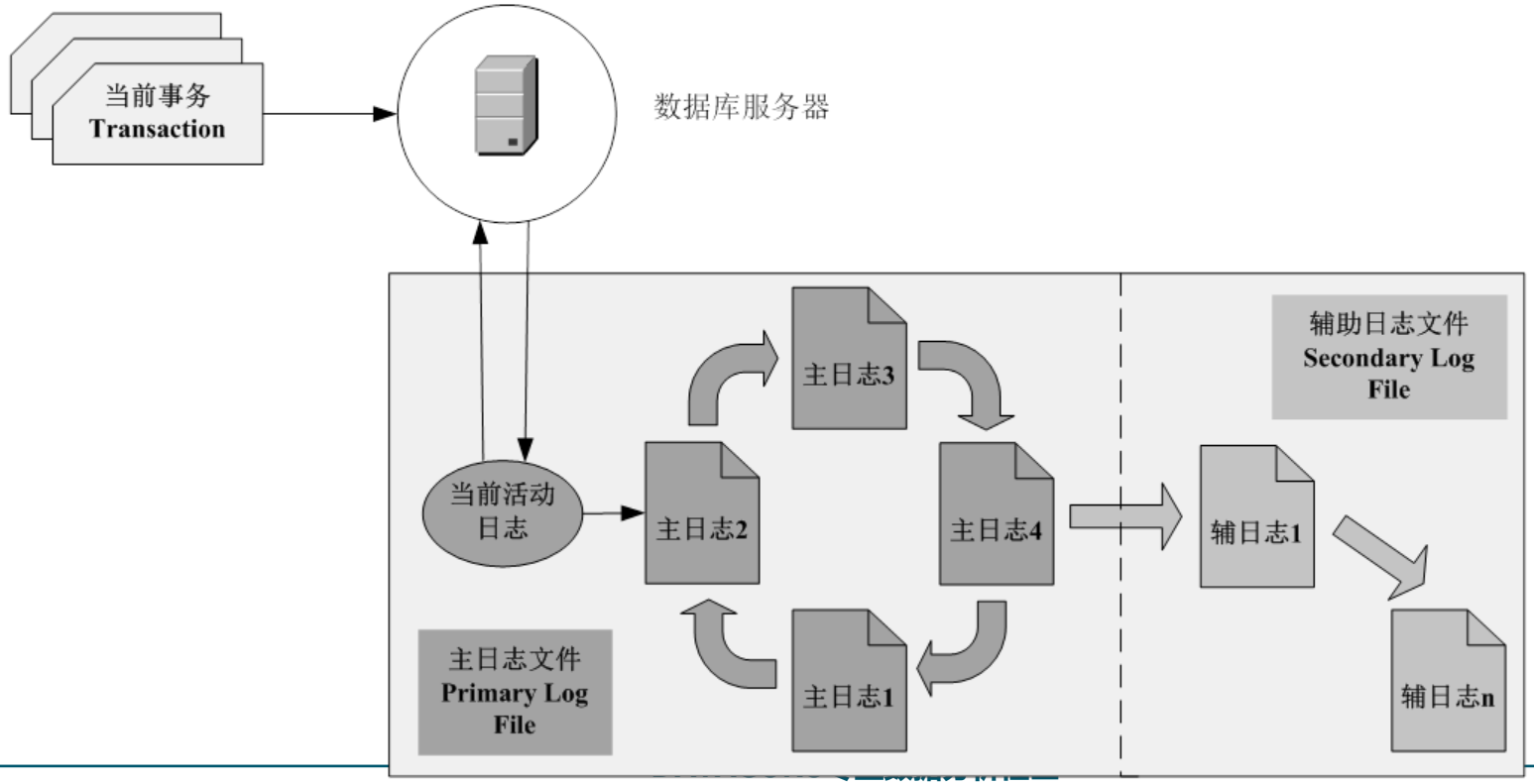
循环日志（Circular Logging）

- 日志文件的内容不会被长期保存
- 日志文件中的内容会被覆盖，当该文件中日志文件的相关数据已经被提交
- 只能使用离线数据库备份
- 不支持前滚



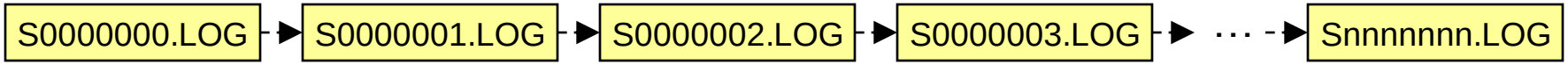
循环日志（续）

- 预先分配几个主日志文件，并依次写入日志记录，循环往复，重复使用
- 主日志文件空间全用完时，临时分配辅助日志文件
- 可用于系统崩溃时恢复当前的事务，只能做离线全备份、版本恢复



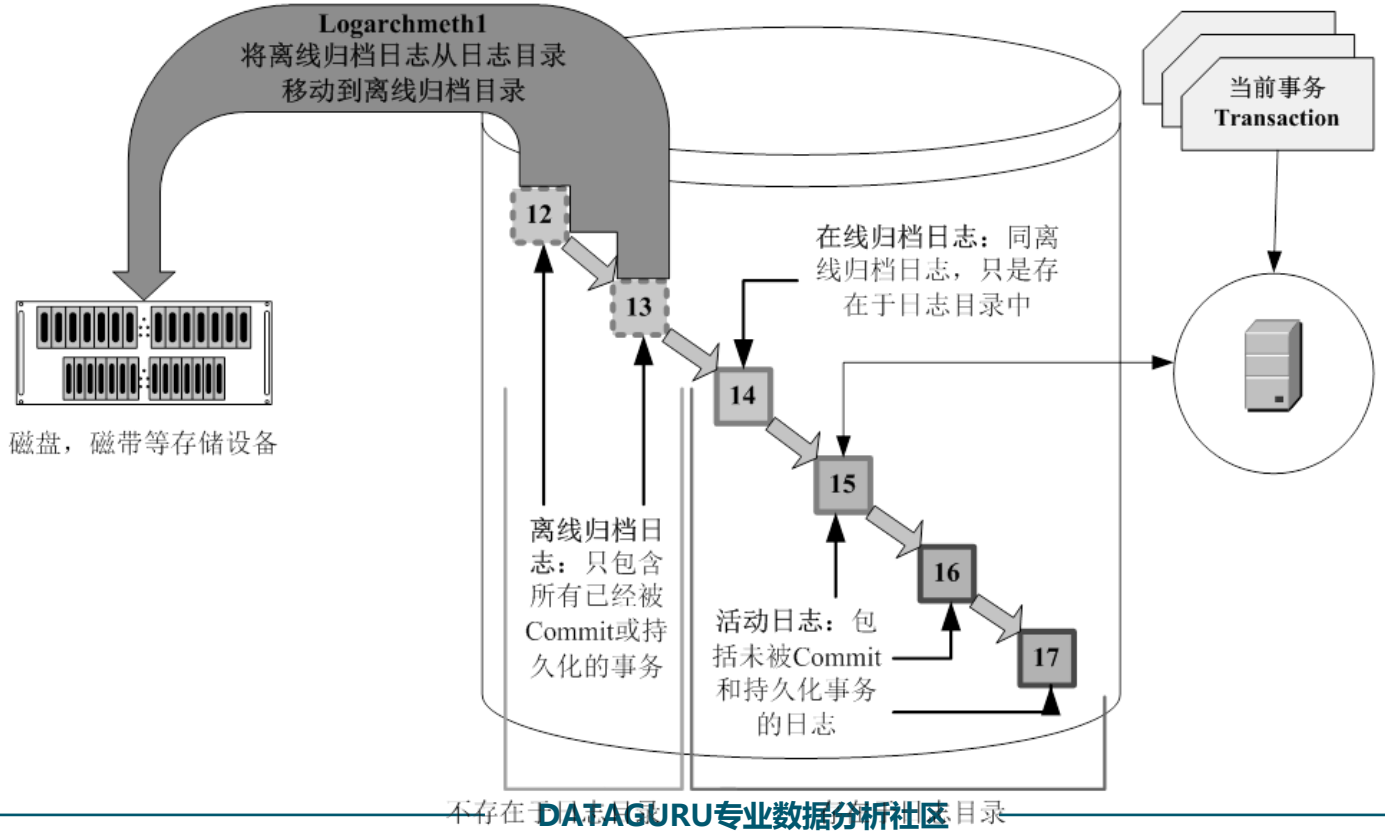
归档日志

- 通过设置参数 LOGRETAIN, USEREXIT, or LOGARCHMETH1/2开启归档日志功能
- 日志文件一直被保存，并且可被归档



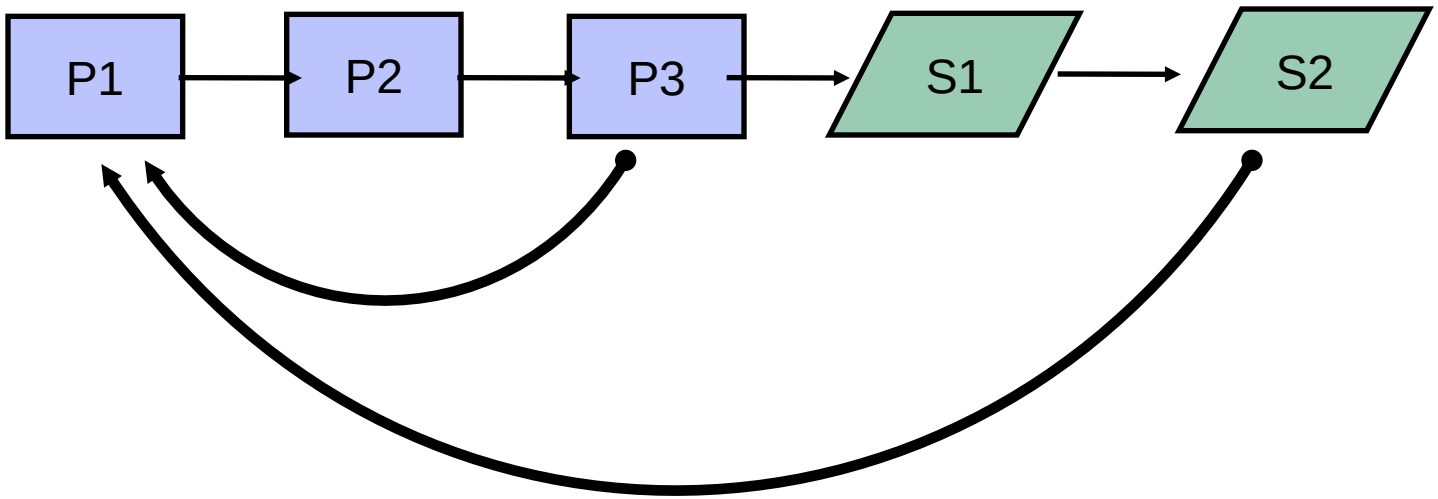
归档日志（续）

- 生产环境下的常用模式，日志文件不重用
- 依次写入日志记录，当日志文件处于非活动状态时，移动到归档目录
- 可用于灾难恢复，数据库在线备份，版本恢复、前滚恢复

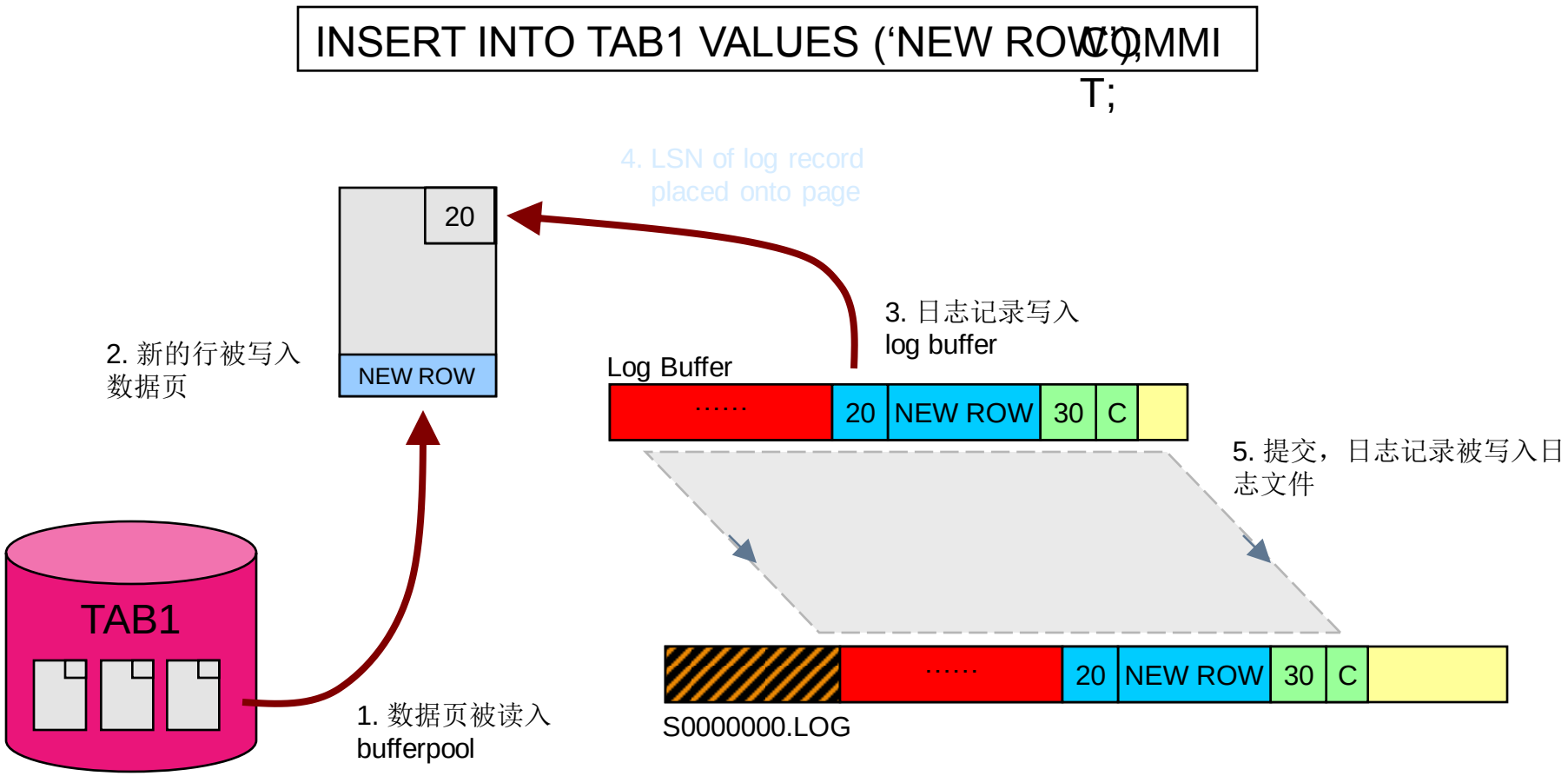


主日志和辅助日志

- 主日志是预先分配
- 辅助日志在需要时才即时分配，一般是长事务或者峰值工作负载
- 对于日常的数据库运行，一般需要保证只用主日志



示例：写日志的过程


INSERT INTO TAB1 VALUES ('NEW ROW');COMMIT;
4. LSN of log record placed onto page
20
NEW ROW
2. 新的行被写入数据页
3. 日志记录写入log buffer
Log Buffer
20
NEW ROW
30
C
5. 提交，日志记录被写入日志文件
TAB1
1. 数据页被读入bufferpool
20
NEW ROW
30
C
S0000000.LOG

提纲

- 日志优化应知应会三原则
- DB2日志原理
- 参数配置与性能
- 性能监控与优化
- 小结与练习

日志相关的配置参数

- NEWLOGPATH (Log Path)
- MIRRORLOGPATH
- OVERFLOWLOGPATH *

日志当前路径及其修改

- LOGRETAIN
- USEREXIT
- LOGARCHMETH1/2 *
- LOGARCHOPT1/2 *
- NUMARCHRETRY *
- ARCHRETRYDELAY *
- FAILARCHPATH *

归档方式和归档路径

- LOGBUFSZ *

日志缓冲区大小

- LOGPRIMARY *
- LOGSECOND * *
- LOGFILSIZ *

日志文件数据和大小设置

- BLK_LOG_DSK_FUL *
- MAX_LOG *
- NUM_LOG_SPAN *
- MINCOMMIT * *

组提交

- SOFTMAX *
- BLOCKNONLOGGED

* Can be changed dynamically

* Updated by Configuration Advisor (AUTOCONFIGURE)

日志文件路径及其修改

- LOGPATH: 当前日志文件路径
  - 当前日志文件所在的位置
  - 日志的I/O特征：大量顺序的写操作， 与DB2的容器(数据文件)非常不一样
  - 日志文件一般不要与数据文件或其他应用程序共享物理磁盘
  - 推荐使用RAID-10做日志，来减少由于磁盘故障导致日志丢失的情况
- NEWLOGPATH: 设置新的日志文件路径
  - 何时使用：
    - ✓ 分离数据和日志的存储，
    - ✓ 或日志存储空间不足
    - ✓ 或使用性能好磁盘做日志存储时
  - 只在重新设置日志文件位置时用，其它情况没有值
  - 日志路径的修改需重新激活数据库才能生效

```
db2 UPDATE DB CFG FOR sample USING NEWLOGPATH <path>
```

日志归档方式和归档路径

- LOGARCHMETH1：定义日志归档方法和归档路径
  - OFF（默认）：使用循环日志记录，且不可前滚恢复
  - DISK：将日志文件将归档当某文件目录
  - TSM：日志文件归档在本地 TSM 服务器
  - 修改此参数后，数据库处于backup_pending的状态，需要做全备份后，数据库才能用。

```
db2 UPDATE DB CFG FOR sample USING LOGARCHMETH1 DISK:C:\DB2LOGARCH
```

日志文件大小 (LOGFILSIZ)

- 参数特征
  - 定义每个主日志文件或辅助日志文件的大小，单位是4K的页
  - 默认1000， 范围为[4 - 1 048 572] （DB2V9.5 FP3）
  - 日志文件大小可达4GB(V9.5 FP3 之前是2GB)
- 选择文件大小时需要考虑的因素:
  - 需要频繁归档日志文件 (e.g. for DR and log shipping)
  - 因为有256个文件的限制，如果存在非常的活动日志，就需要较的日志文件
  - 比较大的文件，会导致在创建该文件时花费较长的时间,
- 建议：最小设置为 20 - 50 MB
- 对于有归档需要的，推荐使用大日志文件

对性能的影响

- 必须使日志文件的大小与主日志文件数平衡
- 如果数据库要运行大量更新、删除或插入事务，而这将导致日志文件很快变满，那么应增大 logfilsiz 的值
- 日志文件太小则会因归档旧日志文件、频繁分配新日志文件以及等待可用的日志文件的开销而影响系统性能。
- 如果磁盘空间不足（日志空间），那么应减小*logfilsiz* 的值，因为主日志是按此大小预分配的
- 太大的日志文件会减小管理归档日志文件和日志文件副本时的灵活性，因为某些媒体可能无法保存整个日志文件
- 大文件日志也可能因为一个文件损坏可能丢失很多数据

主日志数目(LOGPRIMARY)

- 主日志特征
  - 主日志文件的空间预先分配，用于记录事务处理
  - 主日志需要的的磁盘空间是一定的，不管它的内容是空还是写满的
  - 主日志文件的最大数目是256，默认是3
- 性能影响
  - 一般需要保证只用主日志就能满足日常的数据库运行
  - 如果LOGPRIMARY设置为很小的数目，那可能总遇到“log-full”的情况
  - 如果DB2总是分配辅助日志，说明主日志大小不够
  - 对于高负载的OLTP系统，建议增大此参数的设置避免日志分配等待

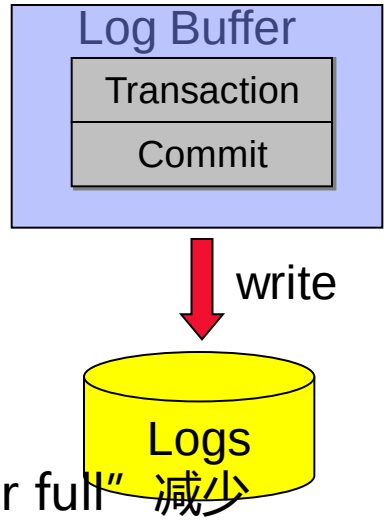
辅助日志数 (LOGSECOND)

- 辅助日志特征
  - 辅助日志文件最大数目由LOGSECOND设置
  - 当主日志文件写满时，为保证事务继续处理，辅助日志就会被即时分配（大小为logfilsiz）
  - 当分配的日志数到达规定的最大数目，还不能完成当前事务，则前事务会被撤销并返回错误
  - 辅助日志一旦分配，它们一直存在（不会删除）直到数据库去激活(deactivated)
- 参数设置
  - 默认为2，可设置为0-254
  - LOGSECOND设置为-1表示启用无限日志(“infinite logging”)
- 性能影响
  - 一般避免设置较大的辅助日志，等待分配空间是也影响性能。
  - 无限日志可能影响利用归档的日志文件做ROLLBACK的性能
- 最大的日志空间
  - 日志文件数为 (logprimary + logsecond)，上限为256
  - 日志空间最大为logfilsiz* (logprimary + logsecond)，上限为1024 GB（DB2 V9.5.3）

无限日志(“infinite logging”)

- 对于活动日志所占用的日志空间大小没有限制
- 通过设置参数 LOGSECOND 的值为 -1 启用
- 数据库必须被配置为归档日志模式
  - 通过设置参数 **LOGARCHMETH1**，使用DISK, TSM等
- 日志文件在写满后，会被归档, 但是会一直保存在日志路径中，直到下一个日志文件被写满。
  - DB2 会保存的主文件个数往往多于LOGPRIMARY
  - 这些而外的文件被称作“cached active log files”
- 可以通过设置参数 MAX_LOG 或者 NUM_LOG_SPAN 来防止应用程序产生太长的active事务

日志缓冲区大小(LOGBUFSZ)



- 定义日志缓冲区的内存大小，使日志I/O更高效
  - 日志缓冲已满是日志写回磁盘的条件之一
  - 它不受STMM (Self-Tuning Memory Manager)管理
  - 默认大小为8，单位为4K的页
  - 必须小于或等于 *dbheap* 参数
- 设置与性能
  - 默认值8对OLTP应用不够，256是调优的较好起点
  - 调优此参数时应使快照监控中的 “number log buffer full” 减少
  - 减少当前已落实（ currently committed ）从磁盘读日志的百分比(参考db2pd -logs)

```
db2 UPDATE DB CFG FOR sample USING LOGBUFSZ 256
```

日志缓冲区大小（续） -设置与性能

- OLAP系统推荐的初始值 (in 4KB pages):

机器的物理内存	LOGBUFSZ
< 16 GB RAM	**4096**
16 – 64 GB RAM	**8192**
> 64 GB RAM	**16384**

- 理想值得选择依赖于系统的工作负载
- 监控 “log buffer full”
  - 查看 **NUM_LOG_BUFFER_FULL** 监控项
- 并非“大”就一定好 (非常大的log buffer有可能对影响性能)
- 从数据库heap分配

组提交数目 (MINCOMMIT)

- 参数特征
  - 表示触发日志缓冲区写回日志文件的**COMMIT**语句数目
  - 默认值1，表示只要COMMIT，当前事务日志记录立即写回文件
  - 通过将多个COMMIT分组提交，减少了I/O操作数目
  - 参数范围为[1 – 25]
- 性能影响
  - 改进多个应用程序在短时间内频繁提交的性能
  - 只用在OLTP中批处理的事务数很高的特殊场景，一般不要设置太高，不高于3

```
db2 UPDATE DB CFG FOR sample USING MINCOMMIT 3
```

参数配置的一般建议

- 日志文件的位置
  - 数据与日志分离，需要为日志配置独占的物理磁盘
  - 推荐使用高速磁盘或者存储系统来做日志
- LOGFILSIZ
  - 一般要设置的比默认值大，比如5000 4K pages或更多
- LOGPRIMARY
  - 将所有的日志分配为主日志文件
  - 辅助日志LOGSECOND只在“紧急”情况下用
- LOGBUFSZ
  - 设置为256 – 1000 (OLTP)
- MINCOMMIT
  - 保留默认值1, 就是设置也不要大于 3

利用DB2 和 操作系统工具来进行日志活动监控和调优

提纲

- 日志优化应知应会三原则
- DB2日志原理
- 参数配置与性能
- 性能监控与优化
- 小结与练习

日志性能有哪些瓶颈？

- 日志的磁盘被共享了?
- 日志记录了太多数据？
- 日志I/O次数很高?
- 事务率太高?
- COMMIT太频繁了?
- Mincommit太低了?

日志瓶颈 – 磁盘空间和IO竞争

- 日志IO对系统性能有至关重要的影响，特别是OLTP环境
- 谁还在用日志所在的磁盘?
  - 检查表空间容器路径，数据库路径，其它应用
- IO所在磁盘的空间不够？
  - 检查日志利用率（快照或者V97管理视图）
- 调优措施：修改日志文件路径
  - 独立的磁盘，与数据表空间容器分离
  - 拥有高速写缓存的高速磁盘
  - RAID并行和冗余

确认日志是否与数据分离

1. df -k

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2             10847448   9537288    759132  93% /
/dev                    517576        96    517480   1% /dev
/dev/sdb1             38456308   1837808  34665000   6% /db2fs
```

2. 检查数据库配置参数'Path to log files'

```
db2 get db config for sample | grep -i 'path to log files'
Path to log files = /db2fs/db2inst1/NODE0000/SQL00006/SQLOGDIR/
```

3. 确认事务日志并不共享文件系统或者物理磁盘

在这个例子中，事务日志与表空间容器共享磁盘。

```
SELECT SUBSTR(TBSP_NAME,1,20) AS TBSP_NAME, INT(TBSP_ID) AS TBSP_ID,
       SUBSTR(CONTAINER_NAME,1,45) AS CONTAINER_NAME
           FROM SYSIBMADM.CONTAINER_UTILIZATION

TBSP_NAME            TBSP_ID     CONTAINER_NAME
-------------------- ----------- ---------------------------------------------
SYSCATSPACE                    0 /db2fs/db2inst1/NODE0000/SAMPLE/T0000000/C000
TEMPSPACE1                     1 /db2fs/db2inst1/NODE0000/SAMPLE/T0000001/C000
USERSPACE1                     2 /db2fs/db2inst1/NODE0000/SAMPLE/T0000002/C000
```

管理视图 – LOG_UTILIZATION (V97)

```
SELECT substr(db_name, 1,10) DB_NAME,
log_utilization_percent, total_log_used_kb,
total_log_available_kb
    FROM SYSIBMADM.LOG_UTILIZATION;
```

```
DB_NAME LOG_UTILIZATION_PERCENT TOTAL_LOG_USED_KB TOTAL_LOG_AVAILABLE_KB
------- ----------------------- ----------------- ------------------------
SAMPLE  21.65                   0                 19902

  1 record(s) selected.
```

总日志空间的利用率%!

该管理视图包含日志利用率的信息.

日志瓶颈 – 大量日志数据

- 怎样判定日志I/O数据量大?
  - iostat (或perfmon) 显示平均 I/O (e.g. > 8k), < 50 IO/s
  - 数据库快照中日志写等待的时间过长
- 调优措施：
  - 减少日志记录的数据
    - 通过NOT LOGGED选项对某些表或者字段不记录日志
    - 用LOAD导入数据和TRUNCATE清空表
    - 使用数据压缩技术减少日志
  - 配置更好的硬件磁盘

日志性能-减少日志数据量

- LOB/CLOB数据的NOT LOGGED选项
  - 大对象(CLOB) 列默认是记录在日志中的。 如果这些数据在数据库系统外部包含和恢复的对象，那么可以用NOT LOGGED标志这些列，来减少IUD时的日志数据
- ALTER TABLE... NOT LOGGED INITIALLY
  - 如果预计在一个表上大批量操作，可以将改表设置为 NOT LOGGED INITIALLY
- 临时表上的NOT LOGGED选项
  - 声明全局临时表：Declared Global Temporary Tables (DGTTs)
  - 创建全局临时表：Created Global Temporary Tables (CGTTs)
- 用LOAD工具导入量大的数据，而不是IMPORT
  - LOAD不通过SQL引擎，直接写数据页，因此不需要日志，而且速度快
  - IMPORT内部通过INSERT实现，需要记录日志
- 用TRUNCATE来清空表
  - Truncating表不用记录日志，比DELETE高效
- 用数据压缩技术来压缩数据和索引
  - 当使用数据压缩时，日志记录也是压缩的，因而将少了日志I/O

磁盘瓶颈 – 事务率高（High Transaction Rate）

- 磁盘IO次数高，或者事务提交的速率高
- 怎样判定事务速率高
  - iostat (或perfmon)显示日志设备每秒发生大于80-100 I/O请求，每次I/O 的平均大小~4 KB
  - 数据库快照和应用程序快照显示commits数目非常高
- 调优措施
  - 减少COMMIT的频率
  - 如果因为日志缓冲区太小导致日志磁盘IO，增加日志缓存的大小，可以用SYSIBMADM.SNAPDB NUM_LOG_BUFFER_FULL确认。
  - 在允许的情况下增加MINCOMMIT，一般只在大量应用程序(成百上千!)都频繁COMMIT时有效。

快照(SNAPSHOT) – Commits和Rollbacks

```
db2 get snapshot for all on sample
```

```
Commit statements attempted                    = 11
Rollback statements attempted                  = 0
Dynamic statements attempted                   = 524
Static statements attempted                    = 16
Failed statement operations                    = 0
Select SQL statements executed                 = 171
Xquery statements executed                     = 0
Update/Insert/Delete statements executed       = 9
DDL statements executed                        = 0
Inactive stmt history memory usage (bytes)     = 0
```

Commits次数

处理动态语句数目

1. GET SNAPSHOT FOR All ON sample
2. 找到Commits/Rollback的日志片段
3. 参考Commit, Rollback, Dynamic, Static, etc.
4. 分析应用类型(OLTP)和预测日志信息

快照 – 日志页（Log Pages）

```
db2 get snapshot for database on sample
```

```
Log space available to the database (Bytes)= 8286039447
Log space used by the database (Bytes)     = 37160553
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)       = 7720996823
Secondary logs allocated currently         = 0
Log pages read                             = 13000
Log read time (sec.ns)                     = 0.000000004
Log pages written                          = 12646941
Log write time (sec.ns)                    = 875.000000004
Number write log IOs                       = 1167739
Number read log IOs                        = 5
Number partial page log IOs                = 105768
Number log buffer full                     = 221
Log data found in buffer                   = 200
```

日志空间利用率

日志读页数

日志写页数

日志写总延迟时间

日志写IO次数

日志缓冲区满次数

1. GET SNAPSHOT FOR DATABSE ON sample
2. 找到日志读写片段
3. 分析日志信息：
   - 如果‘Number Read Log IOs’ 远大于 ‘Log Data found in buffer’,那么你需要增多LOGBUFSZ
   - 如果 ‘Number of log buffer full’很高，增多LOGBUFSZ
   - ‘Log write time/’Number write log IOs’ 很重要，最好<=2ms

管理视图 – SNAPDB

```
SELECT VARCHAR(DB_NAME,20) AS DBNAME,
     CASE WHEN (commit_sql_stmts + rollback_sql_stmts) > 0
     THEN DEC((1000 * (log_write_time_s / (commit_sql_stmts
+          rollback_sql_stmts))),5,0)
     ELSE NULL
     END AS LogWriteTime_PER_1000TRX,
     log_write_time_s AS LOGWTIME,
     commit_sql_stmts + rollback_sql_stmts AS
TOTALTRANSACTIONS
 FROM sysibmadm.snapdb;

DBNAME   LOGWRITETIME_PER_1000TRX           LOGWTIME   TOTALTRANSACTIONS
------   ---------------------------------  ---------------   -------
SAMPLE   10                                  20                  2000

  1 record(s) selected.
```

1000个事务的累计写日志等待时间

该管理视图包含一个代理等待日志从缓存写回磁盘的时间。

提纲

- 日志优化应知应会三原则
- DB2日志原理
- 参数配置与性能
- 性能监控与优化
- 小结与练习

日志优化应知应会-小结

- 日志是事务处理和恢复的必备机制
  - 数据和日志是数据库中必不可少的两部分
  - 日志性能对数据库的整体性能有至关重要的影响
- 日志先写原则
  - 日志记录每一行数据的变化
  - 写入磁盘顺序：日志为先，数据在后
- 日志与数据分离原则
  - 为提高性能，防止数据和日志IO竞争
  - 为防止数据和日志磁盘同时损害而导致数据丢失
- 日志最优待遇原则
  - 在每次事务提交时都必须将日志写入到磁盘，因此日志性能非常重要
  - 给日志配置独立的物理磁盘，性能最好的磁盘（VIP）
- 日志参数配置
  - 数据库日志相关的参数对日志性能有重要影响，特别是MINCOMMIT和LOGBUFSZ
- 日志监控和调优
  - 可用快照工具或管理视图监控日志性能
  - 日志数据量大，考虑减少日志记录的数据进行调优
  - 日志IO次数多，考虑提高COMMIT的频率或者增加MINCOMMIT的设置

Thanks

FAQ时间