SQL SERVER 2005 基于事务日志的备份与恢复深入研究①

向 猛, 谢立靖

(湖南商务职业技术学院 实训部, 长沙 410205)

摘 要: SQL SERVER 2005 的备份/恢复功能表面上看似简单, 但实际上其实现机制相当复杂并且微软公司并没有对外公布其备份/恢复机制的具体实现细节. 包括专业数据库管理员在内很少有人能将 SQL SERVER 2005 的备份/恢复机制解释清楚, 力图从一个较深入的层次研究分析 SQL SERVER 2005 的事务日志以及基于事务日志的备份与恢复.

关键词: 数据库管理系统; 备份; 恢复; 事务日志; 最小恢复日志序列号

In-Depth Research of SQL SERVER 2005 Backup and Recovery Based on the Log of Transactions

XIANG Meng, XIE Li-Jing

(Training Department, Hunan Vocational College of Commerce, Changsha 410205, China)

Abstract: SQL SERVER 2005 backup and recovery function looks very simple on the surface, but in fact, the realization mechanism is very complex and Microsoft did not announce concrete implementation details of backup and recovery. Including professional database administrator, very few people can explain SQL SERVER backup/recovery's implementation details clearly, this paper strive to research and analysis of the SQL SERVER Transaction Log and backup and recovery based on the log of transaction from a deeper level.

Key words: DBMS; backup; recovery; transaction log; minimum recovery LSN

SQL SERVER 2005 备份与恢复功能是保证企业数据灾难恢复的关键功能之一. 其备份与恢复功能表面上看似简单, 但实际上, 简单的外衣下隐藏了很多复杂的实现原理、实施细节等方面的内容, 如果不懂其备份与恢复的实现原理, 没有对各种故障类型下的备份恢复做深入细致的研究, 而仅仅按照一般 SQL SERVER2005 管理书籍所列出的步骤去简单的备份与恢复, 可能会将企业宝贵的数据资产置于很大的安全风险之下, 造成难以估量的损失.

1 日志文件的逻辑结构

在 SQL SERVER2005 中, 一个数据库至少需要一个日志文件来包含(或者曾经包含)所有事务的日志信息, 并且一个物理上的日志文件被划分成多个逻辑上的虚拟日志文件(virtual log files, VLFs)来管理. SQL SERVER 对日志文件进行诸如收缩、日志截断等管理操作都是以 VLFs 为最小单位的. SQL SERVER 日志文件的逻辑结构如图 1 所示:

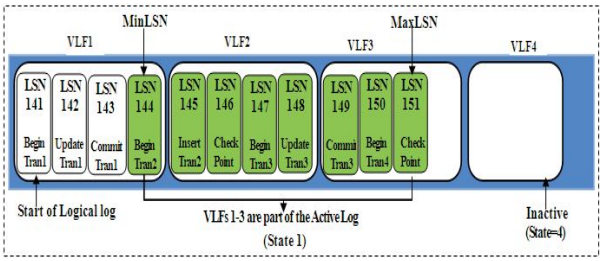

图 1

① 收稿时间:2012-11-01;收到修改稿时间:2012-12-02

SQL Server 在写事务日志到日志文件时, 先写第一个可用的 VLF (VLF1), VLF1 写满之后再写 VLF2, 按顺序以此类推. 如果所有 VLFs 都写满了之后, 而 VLF1 又允许覆盖重写的话, 则会对 VLF1 再次写入. 如果所有 VLFs 都写满之后, 而 VLF1 内容又不允许被覆盖, 此时为了存放即将到来的日志记录, 则存储引擎不得不分配更多的 VLFs 来存储日志记录. VLFs 里面的日志记录在何种情况下能够被覆盖以及怎样被覆盖, 与数据库的恢复模式、日志备份策略以及 VLFs 所处状态等息息相关.

1.1 日志记录

SQL SERVER 的每条日志记录都有一个称为日志序列号 (LSN)唯一编号. 日志记录在日志文件中的记录方式如上面日志文件结构图所示, 在逻辑上是线性的, 新的日志记录都是写在日志文件的逻辑末端, 不像数据那样可能会写到磁盘的各个地方. 由于 LSN 是按日志记录产生的先后顺序不断增长且其逻辑线性的记录方式, 因此, LSN 本质上是标识了日志记录在日志文件中位置, 在恢复时是通过 LSN 找到所需要的日志记录.

如果使用 DBCC LOG(DB_NAME,3)查看当前日志文件中各日志记录的详细信息, 可以发现每条日志记录中都包含了该日志所引用的数据页的页码、上一次引用该数据页的日志记录的 LSN、日志记录所属事务的事务 ID、事务名以及同一事务前一条日志记录的 LSN 等众多信息. 在日志记录中包含同一事务上一条日志记录的 LSN 使得属于同一个事务的日志记录逐步向前逻辑上组成一个完整的链条, 以便在需要回滚时, 能够按事务原来相反的顺序从最后的日志记录逐条向前回滚, 直至事务的开始.

在上面所示精简的日志文件逻辑结构图中, 最后一个记录的 LSN 是 151, 对应于最后的 Checkpoint 操作(Checkpoint: 将缓冲中所有脏页刷新到磁盘的操作), 最开始的一个日志记录 LSN141 对应于事务 1 的 Begin tran 语句.

1.2 活动日志

活动日志是在崩溃恢复时或介质恢复的 Recovery 阶段时为保证事务 ACID 属性而进行前滚(Redo)或回滚(Undo)所需要的那部分日志. 在日志文件中就是 Minimum Recovery LSN 到日志文件结尾 LSN 之间的这段日志.

Minimum Recovery LSN(MinLSN)是在崩溃恢复或介质恢复的 Recovery 阶段应用事务日志进行 Redo 或 Undo 的起点. SQL SERVER 中的 MinLSN 是最后一次 Checkpoint 时, 日志文件中还未结束的事务(活动事务)当中最老的一个事务的起始 LSN 和最后一次 Checkpoint 操作本身的起始 LSN 两者中的较小者. 对于上面的日志文件结构图, 最老活动事务(事务 2)的开始 LSN144 小于最后 Checkpoint 的起始 LSN151, 因此 Minimum Recovery LSN 就是事务 2 的起始 LSN144. 随着当前数据库未提交事务的不断提交, 日志文件中最老活动事务的不断易主以及 Checkpoint 的不断执行, MinLSN 因此也得以不断向前推进. 如果系统中存在长时间执行的事务, 其开始执行后一直未结束, 则该事务会阻止 MinLSN 向前推进. 在上面日志文件逻辑结构图中, 从 MinLSN 到最后 LSN, 是当前活动日志(Active Log). 并且对于虚拟日志文件而言, 如果某个 VLF 中不包含任何 MinLSN 之后的日志, 其状态就是非活动状态(inactive), 而如果 VLF 中包含了任何 MinLSN 之后的日志记录, 其就是活动状态的(active). 对于不同状态的 VLFs, 存储引擎可以采取不同的处理方式, 比如日志截断、日志备份以及日志收缩等.

2 日志管理

2.1 日志截断

日志文件所包含的 VLFs 数目, 因为性能等原因不可能无限增长, 因此, 为了重复利用有限 VLFs 的空间, 需要将一些对于保证事务 ACID 属性无关紧要的日志记录从 VLFs 中清除出去(日志截断)或者将这些日志进行存档(日志备份). 将非活动 VLFs 标记为可以重用状态的逻辑过程称为日志截断. 被标记为可重用状态的 VLFs 里面的日志记录可以被新的日志记录覆盖, 这个覆盖过程在存储引擎写新日志时自动处理, 不需要人工干预.

需要指出, 对于保证事务 ACID 属性无关紧要的日志是指 MinLSN 之前的日志, 这些日志对应事务都已提交并且实际修改已写入磁盘, 因此截断的是 MinLSN 之前的日志. 活动日志对应的各事务还没有得到妥善的处理, 在进行数据库恢复时, 为了妥善处理这些事务保证其 ACID 属性需要使用到这部分日志, 因此任何情况下都不能被截断.

非活动 VLFs 里面的日志记录能否被自动截断主

要取决于为数据库所设置的恢复模式, 不同的恢复模式对非活动 VLFs 有不同的处理方式. 当然非活动 VLFs 里面的日志记录也能通过执行命令手工进行日志截断.

此外, 日志截断是一个逻辑上的过程, 并不会减少日志文件中已分配 VLFs 的个数, 也不会物理上释放已经为各个 VLFs 分配的磁盘空间. 如果需要实实在在看到已经十分巨大的日志文件变小, 需要执行一个称为日志收缩的过程, 这是一个减少非活动 VLFs 数量, 并释放这些 VLFs 占用空间的物理过程. 如果系统配置良好且备份策略得当, 日志文件一般不会无缘无故持续增加直到撑爆硬盘. 日志收缩一般用来处理一些日志文件相关的紧急情况.

SQL SERVER 中的日志截断以 VLF 为单位的, 截断的范围是从第一个非活动 VLF 起, 到最后一个非活动 VLF 为止.

2.2 简单恢复模式下的日志管理

在简单恢复模式下, 已结束事务的日志在 VLFs 中是不会长久保存的, 每一次 Checkpoint 时, SQL SERVER 都会自动调用截断进程来检查是否有 Inactive 的 VLFs, 如果有 Inactive 状态的 VLFs, 就将其标记为可重用(截断的本质). 在有新的日志记录需要 VLF 时, 这些可重用状态的 VLFs 中的日志将被新的日志记录覆盖, 以前的日志记录在还没有存档的情况下就不复存在了. 正是由于日志还未存档就丢失的原因, SQL SERVER2005 干脆不支持在简单恢复模式下进行日志备份, 因为备份出来的日志也没有任何意义, 可能缺少丢失的日志, 是不连续, 不完整的. 所以在该恢复模式下事务日志文件是不具备传统意义上的备份恢复功能的, 其事务日志仅仅用于保持事务的 ACID 属性.

2.3 完整及大容量日志恢复模式下的日志管理

在完整及大容量日志恢复模式下, 每一次 Checkpoint 时, 系统不会在事务日志还未存档的情况下自动截断 MinLSN 之前的日志, 不会覆盖非活动 VLFs 中的日志记录. 如果日志空间耗尽, 其通过增加新的 VLFs 使得日志记录得以继续进行. 由于该两种模式阻止了非活动 VLFs 中的内容被自动覆盖, 使得在不进行事务日志备份或手工执行日志截断的情况下, 系统不断为日志文件分配新的 VLFs, 使得日志文件持续增大直至撑爆整个磁盘空间, 同时也会出现文件碎片过多影响性能等问题.

在这两种恢复模式下, 当用户手工执行了日志截断命令或者进行日志备份(日志备份默认选项执行日志截断), 才截断这些事务日志. 在该两种恢复模式下, 完整备份与差异备份默认只进行备份, 并不附带执行日志截断. 在 SQL SERVER 2005 中, 需要注意的一个比较特殊的情况是: 在创建一个新数据库后, 如果从未对该数据库做过完整备份, 即使该数据库是完整恢复模式或大容量日志恢复模式, 存储引擎也会像对待简单恢复模式一样, 在需要时自动截断事务日志. 只有在对数据库做过一次完整备份后, 日志管理才回归完整/大容量日志恢复模式的“正常”情况.

3 备份

数据库及事务日志的备份到底做了那些事? 都备份了那些内容? 从微软公布的资料来看, 数据库备份一般经历两个步骤, 首先读取数据文件中的数据, 然后将其写入备份设备; 接着读取某些事务日志, 然后将其写入备份设备. 根据备份类型的不同, 所读取的数据文件及日志文件的内容会有一定的差异.

对于完全数据库备份, 在备份操作开始之前, 其首先强制执行一个检查点操作(backup Checkpoint), 然后开始从数据文件中读取全部已分配的数据页并写入备份设备, 强制执行一个检查点操作的目的是为了在备份操作开始之前将缓存中的所有数据进行存档, 以便最小化以后的恢复时间. 数据文件读取完毕并写入备份设备后, 然后开始读取日志文件, 完全备份中包含的日志记录开始于备份检查点(backup Checkpoint)时刻日志文件中的 MinLSN, 结束于读数据文件操作完毕时日志文件中存在的最后日志记录(在 backup Checkpoint 日志之后,备份本身日志记录开始之前的某个 LSN). 日志写入备份设备完成后, 数据库完整备份结束. 因此完整备份中保存的是备份操作读数据文件结束时刻的状态.

SQL SERVER 2005 中一个比较特殊的情况: 是在进行第一次完整数据库备份时, 备份操作会截断 MinLSN 之前的日志记录, 对于之后的完整数据库备份, 则不会截断任何日志.

对于事务日志的备份, 其备份完整备份 LastLSN 或者最后一次日志备份 LastLSN 到当前日志文件最后 LSN 之间的所有日志. 本质上就是将当前非活动

VLFs 中的日志存档. 日志备份的备份链开始于一个完整备份, 而一个计划外的完整备份可能会破坏预先设计的日志备份链, 导致该计划外完整备份之后的日志备份对于计划的备份策略来说变成无用的垃圾文件. 一般情况下, 如果不熟悉当前正在实施的整个备份策略, 而又由于各种原因非要对数据库作完整备份, 此时最好加上 copy_only 选项, 该选项使计划外的完整备份不会破坏原来的日志备份链, 不会对原来的备份策略产生影响.

4 数据库恢复过程详细分析

4.1 数据库恢复需要处理的两类不正确情况

根据事务在最后 Checkpoint 及故障点时的完成情况, 无论是崩溃恢复, 还是介质恢复的 Recovery 阶段都需要处理图 2 所示的两类处于不正确状态的事务.

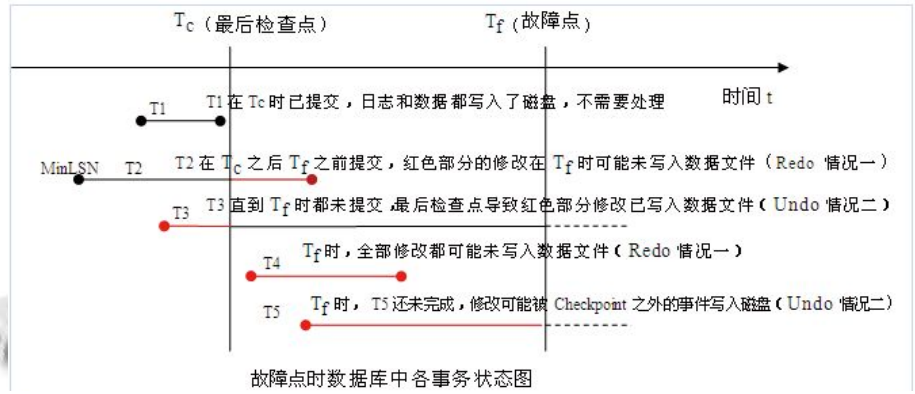

图 2

4.1.1 需前滚(重做)的情况

由于日志和数据写入磁盘完全是异步进行的, 因此有可能在系统发生故障时数据库中存在“某事务虽已提交, 但其数据更改并未真实反映在数据库中”的情况. 假设某个事务(上面的事务 T2 和 T4)现在被提交, 其完整日志记录会被写入了磁盘日志文件中, 而实际的脏数据页可能还未来得及一个检查点就发生了故障, 导致在故障时, 该事务“不正确”(虽然已提交, 却没有实际产生效果). 在这种情况下, 需要应用已经存在于日志文件中的该事务的日志进行前滚(重做), 以便保持事务的持久性.

4.1.2 需回滚(撤消)的情况

如果某个事务直到故障点时都还未提交(上图中的事务 T3 和 T5), 而此前的一个 checkpoint 操作, 将该未提交事务的部分更改已经被写入数据文件, 随后, 实例在该事务提交前崩溃了. 因此, 数据库在崩溃时刻存在“事务还未提交, 但是部分修改已经写入磁盘”的情况. 恢复过程需要找出该未提交事务的日志记录, 然后撤消对数据的修改.

4.2 崩溃恢复的起始 LSN--MinLSN

崩溃恢复(重启恢复、实例恢复)是在遭遇系统故障后, SQL SERVER 实例重新启动过程中自动执行的一种恢复. 系统故障是发生诸如硬件故障(如 CPU 故障), 操作系统故障, DBMS 代码故障等等, 导致当前内存中的部分或全部事务非正常终止, 但并不使数据库本身遭受到破坏, 存储数据文件、日志文件的磁盘依然正常的一种故障. 在这些故障修复后, SQL SERVER 实例仍然能够正常启动.

发生这些故障后, 为了使非正常终止的事务得到妥善处理, 需要经历前滚(Redo)和回滚(Undo)两个阶段. 前滚是针对故障点时虽已提交而修改未写入磁盘的事务而言的(第一种不正确情况, 事务 T2、T4), 未写入磁盘的修改需要通过日志文件中的日志记录重做一遍. 而回滚需要处理的是故障点时还未提交, 但其修改已经写入磁盘的事务而言的(T3, T5). 无论是前滚还是回滚, 都需要找出前滚或回滚的起始 LSN(MinLSN). 在 SQL SERVER 2005 中, 数据库根页(boot page)中的 dbi_checkptLSN 参数(该参数由 m_fSeqNo, m_ block Offset, m_slotId 三部分构成)总是记录数据库最后一个 Checkpoint 操作本身的起始 LSN, 可以使用 DBCC TRACEON(3604),DBCC DBINFO(‘数据库名’)来查看 boot page 中的该参数值. 通过该 dbi_checkptLSN 参

数值，就可以找到系统故障时，日志文件中最后 Checkpoint 操作的日志记录(LOP_BEGIN_CKPT, LOP_END_CKPT). 如果该最后 Checkpoint 执行时刻，数据库中还存在一个或多个未结束的活动事务，系统就会在该 Checkpoint 操作的 LOP_BEGIN_CKPT 和 LOP_END_CKPT 日志记录之间插入一条操作为 LOP_XACT_CKPT 的日志记录，在该 LOP_XACT_CKPT 记录的 log Record 字段中记录了最后 Checkpoint 执行时所有未提交事务的日志信息，包括事务起始日志记录的 LSN 以及事务内各更改语句的 LSN. 找出 log Record 字段中最小的 LSN 即为前滚的起点 LSN(MinLSN); 如果最后 Checkpoint 执行时刻，数据库中不存在未提交的活动事务，则该 Checkpoint 起始 LSN 就是前滚的起点 LSN(MinLSN).

4.3 崩溃恢复的前滚过程

崩溃恢复的前滚过程是：恢复进程首先按上面过程找到 MinLSN，然后从 MinLSN 开始按顺序依次处理后面的每一条日志记录，直到达到日志文件的最后一条日志记录. 在处理每条日志记录时，对属于已提交事务的日志记录来说，当它所引用的数据页的当前 LSN(在数据页页头表示为 M_LSN)大于或等于日志记录的本身的 LSN 时，什么也不需要做，因为日志记录的更改已经永久的反映到磁盘上了. 当它所引用的数据页的当前 LSN(M_LSN)小于日志记录本身的 LSN(CurrentLSN)时，则该日志记录必须被重做以确保事务更改是永久的；对于活动日志中，属于未提交事务的日志记录，也需要进行重做，尽管这些日志记录在后面的回滚阶段又被撤消.

4.4 崩溃恢复的回滚过程

执行完前滚操作后，数据库还可能存在上面提到的第二种情况，需要执行所谓的回滚过程. 回滚操作的过程是：恢复进程根据前面所述过程首先找到 log Record 字段中记录的最后 Checkpoint 执行时所有未提交事务的日志信息(该 LOP_XACT_CKPT 中记录的最后 Checkpoint 时的活动事务可以等同的认为是故障点时未提交事务)，再结合各事务内的日志记录链条，从未提交事务的最后 LSN 开始，沿着日志记录链条一直到事务的起始 LSN，做与原来操作相反的操作(即：UNDO 操作)，当 log Record 字段中记录的所有未提交事务都经过这样的处理后，数据库的回滚操作宣告完成，此时，数据库处于联机状态，对用户可用.

崩溃恢复所需要的时间取决于 log Record 字段中记录活动日志的日志量，活动日志越大所需要的恢复时间越长，如果某个事务 Begin 了很长时间，但一直没有 Commit 或在 Rollback，而事务在日志文件中产生了大量日志记录，在系统故障后，实例重启时，系统需要对该事务产生的大量日志进行前滚和回滚，花费很长的处理时间.

4.5 介质恢复

介质故障一般是存储数据文件、日志文件或系统文件的介质发生故障，导致 SQL SERVER 实例甚至操作系统都无法启动，此种故障危害性很大，无法通过过崩溃恢复由 SQL SERVER 自身来解决，可能需要在全新的软硬件环境下人工使用备份来执行介质恢复操作. 上述崩溃恢复的前滚和回滚过程在介质恢复中同样存在，在 SQL SERVER 2005 中介质恢复一般由称之为 Restore 和 Recovery 两个阶段组成，Restore 阶段用于重演以用户视角来看毫无问题的事务，将数据库当前状态不断向前推进，而介质恢复的 recovery 阶段，情况类似于崩溃恢复的过程，需要处理到各备份完成时刻，数据库中存在的几种“不正确”事务.

5 一个完整备份+多个日志备份策略分析

在实际生产环境中，备份策略的制定主要决定于用户所能容忍的丢失数据量，所能容忍的丢失数据量由两次日志备份之间的时间间隔来控制，如果用户最多只能容忍 30 分钟的数据丢失，则两次日志备份之间的间隔不应该超过 30 分钟. 某企业某一天当中备份情况如图 3 所示:

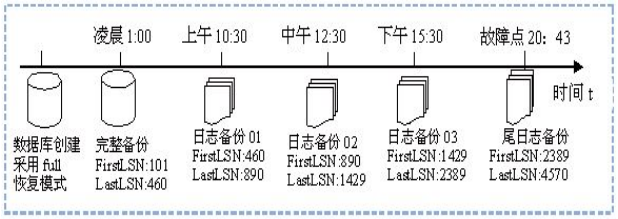

图 3

首先执行完整备份，将数据库在 1: 00 时的完整状态备份下来. 完整备份文件中除了包含所有数据之外，还包含备份检查点时刻日志文件中的活动日志. 在 10: 30，执行第一次事务日志备份，日志备份的备份链的起点是完整备份的 FirstLSN(101)，结束于当前日志文

件中的最后一条事务日志. 接下来的 12: 30 时刻的日志备份开始于上一次日志备份的 LastLSN(890)结束于当前日志文件最后一条日志(1429). 在上述策略中, 从 1: 00 到 15: 30 这个时间段内产生的每一条日志记录都妥善的保存在了备份文件中. 由于日志备份操作默认是截断日志的, 所以在每次日志备份后非活动 VFLs 里面记录可能被新的日志覆盖掉了.

如果在 20: 43 存储数据文件的磁盘发生了介质故障(在生产环境中, 一般将数据文件和日志文件分开存放), 而事务日志文件仍完好无损. 此种情况下, 是能够将数据库恢复到故障点状态的, 但前提是, 在执行恢复操作之前, 首先要进行尾日志备份(最后一次日志备份的 LastLSN 到故障点时日志文件的最后一个 LSN). 在 SQL SERVER 2005 中进行介质恢复时, 如果系统检测到在恢复时恢复的目标数据库还有日志记录没有备份, 会提示先进行尾日志备份, 并不允许进行还原操作. 除非使用 REPLACE 选项跳过尾日志备份而强制执行还原操作.

对于尾日志的备份, 如果在故障严重程度较低, SQL SERVER 实例还能正常启动的情况下, 使用常规的日志备份语句 Backup log 加上 no_trancate 选项来完成. 但如果故障严重程度较高, 导致了 SQL SERVER 实例甚至服务器操作系统都无法启动的情况下, 尾日志备份则困难得多. 一种常出现的情况是: 在原服务器上通过特殊手段获得数据库的日志文件, 然后在拥有相同颁布号 SQL SERVER 实例的其他服务器上进行恢复. 最差的一种情况是, 服务器彻底报废, 尾日志永远丢失, 这样就无法恢复最后日志备份到故障点之间的事务.

在进行数据库恢复时, 首先使用完整备份进行还原, 将数据库置于 1:00 时的状态, 在还原时注意使用 NORecovery 选项指定还原结束时不让数据库经历 Recovery 阶段, 以便可以继续应用后续的日志备份进行还原, 将数据库的状态向前推. 接着再按照备份的顺序依次还原日志备份 01、日志备份 02、日志备份 03、尾日志备份, 将数据库的状态不断向前推直至故障点时的状态. 在使用前面的日志备份 01、日志备份 02、日志备份 03 进行还原操作时, 实际执行的操作很简单, 就是把这些日志备份中的所有记录进行重做. 但在使用最后一个日志备份进行还原的时候, 为了使数据库处于一致、联机状态, 需要使用 RECOVERY 选项让数据库经历 recovery 阶段. recovery 阶段的处理过程如同前面崩溃恢复时进行的相应处理过程.

如果在 20: 43 时刻是存储日志文件的磁盘发生介质故障, 可以使用以前的备份将数据库恢复到最后一个日志备份的末尾状态, 而最后一个日志备份到故障点之间的所有事务将永远全部丢失. 在故障程度较轻的情况下, 也可以采取某种特殊的方法仅仅使用 MDF 数据文件进行恢复, 只不过此方法并非传统意义上的恢复过程.

在使用上面基于事务日志的备份恢复策略时, 需要确保各备份文件的日志记录是连续的, 绝对不允许出现 LSN 断开的情况. 可以在备份后使用 Restore headeronly 命令来检查各备份文件的 LSN 是否出现断开. 日志链断开一般出现在恢复模式切换时, 特别是简单恢复模式到完整/大容量日志记录模式切换时或备份计划外执行的完整/差异备份等情况. 比如: 一开始, 将数据库置于简单恢复模式, 并在该恢复模式下执行了完整备份. 完整备份后, 再执行日志备份, 这时发现简单恢复模式下, 不能执行日志备份. 于是把恢复模式设置为完整恢复模式或大容量日志恢复模式, 再执行日志备份. 在这种情况下, 后面执行的日志备份其实没有任何意义了, 因为简单恢复模式的自动日志截断导致日志记录链肯定断了.

6 总语

SQL SERVER 2005 的备份恢复功能复杂而强大, 还有诸如系统数据库的备份与恢复、差异备份与恢复、文件及文件组的备份与恢复、备份与恢复的自动化等等. 数据管理人员只有在深刻理解备份恢复原理的情况下, 才能对数据存储管理做到了然于胸.

参考文献

1 Delaney K. Inside Microsoft® SQL Server™ 2005: The Storage Engine. US: Microsoft Press, 2006.

2 赵松涛.SQL SERVER 2005 奥秘.北京:电子工业出版社, 2007.

3 李爱武.SQL Server 实例恢复中重做日志记录定位机制研究.现代计算机,2009,315:107–109.

4 Paul SR. Understanding SQL Server Backups. [2011-08-14]. http://technet.microsoft.com/zh-cn/magazine/2009.07.sqlbackup.aspx#MtViewDropDownText.

(下转第 74 页)

语音输入, 具有回声抑制功能, 还可直接驱动 500mW 的喇叭, 因而便于实现系统音频警示提醒功能等;

g) 终端单元电源管理模块: 选用了 National Semiconductor 公司的 LP3907 芯片, 支持 2 路 Bulk, 开关频率高达 2.1MHz, 效率高, 并有 2 路 LDO 输出, 符合该系统的电源设计需求.

5.5 系统监控视频数据的保存

本设计专门预留了通道 MII2(如图 2), 用来实时保存 6 路监控视频数据. 因为 MPC8270 的 MII1 和 MII2 之间可以工作网络交换状态, 所以 MII2 得到的是 6 路未经过高压缩率处理的视频数据. 同时, 用户通过终端单元可以选定保存数据的通道编号, 还可以设定通道数据保存的方式, 如定期保存、长期保存或者不保存等.

5.6 系统的应用领域

此无线网络的视频监控系统支持同时监视 6 路视频画面, 并具有无线视频传输功能, 而且系统视频数据传输是通过 3G 网络的基站完成的, 覆盖面广阔. 那么, 在国内使用此系统终端单元就没有距离限制. 例如智能家居安防系统、车载无线视频监控系统等均可以采用此方案.

6 结束语

本文完成了多路监控系统的监控单元和终端单元设计工作, 并将互联网有线网络和 3G 无线通信网络通过监控单元的网络交换机等成功结合, 实现了一种基于 3G 无线网络的六路视频监控系统的方案设计.

参考文献

1 付少华,付红桥,王政.基于 3G 网络的手机移动视频监控系统的设计.计算机应用,2011,31(1):70–72.

2 李臻,胡平.基于 3G 网络的车载视频监控终端实现.煤炭技术,2011,30(5):153–155.

3 胡平,韩兴.基于 3G 的无线实时视频监控系统设计.计算机工程与设计,2011,32(12):4015–4018.

4 朱海华,陈自刚.基于 3G 的无线视频家居安防系统设计.计算机测量与控制,2011,19(12):2982–2988.

(上接第 23 页)

5 Suhas D. Truncating a Transaction Log File. [2011-12-21]. http://blogs.msdn.com/b/suhde/archive/2009/07/18/revealing-the-secrets-truncating-a-transaction-log-file.aspx.

6 Randal PS. Understanding Logging and Recovery in SQL Server. [2011-12-26]. http://technet.microsoft.com/en-us/magazine/2009.02.logging.aspx.

7 Randal PS. Using fn_dblog to tell if a transaction is contained in a backup. [2012-02-17]. http:// sqlskills.com/BLOGS/PAUL/post/Search-Engine-QA-6-Using-fn_dblog-to-tell-if-a-transaction-is-contained-in-a-backup.aspx

8 Randal PS. How do checkpoints work and what gets logged. [2012-02-11].http://sqlskills.com/BLOGS/PAUL/post/How-do-checkpoints-work.aspx

9 Randal PS.Debunking a couple of myths around full database backups. [2012-02-11]. http://www.sqlskills.com/BLOGS/PAUL/post/Debunking-a-couple-of-myths-around-full-database-backups.aspx

10 Randal PS. More on how much transaction log a full backup includes. [2012-04-18]. http://www.sqlskills.com/BLOGS/PAUL/post/More-on-how-much-transaction-log-a-full-backup-includes.aspx

11 宋沄剑.SQL Server 中灾难时备份结尾日志(Tail of log)的两种方法. [2012-04-19]. http://www.cnblogs.com/CareySon/ archive/2012/02/23/2365006.html