2.8SQLProfiler_SQLServer2005数据库开发详解 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/140/2021_2022_28SQLProf_c100_140305.htm 2.8 SQL Profiler通过 SQL Profiler 工具程序，可监控应用程序如何访问数据库引擎。普通来说，当系统性能需要优化或是应用程序对数据库访问的结果不合预期，都可以使用该工具确认视图问题所在。新版的 SQL Profiler 除了依然可以用来监视数据库引擎正在执行的工作，更增加了获取 Analysis Services 实例运行时所发生的事件。通过新增的跟踪选项，一样可以挑选需要录制的 AS事件、数据元以及设置过滤条件。此外可将跟踪的事件结果以 XML 格式存储，以及通过图形的方式显示与分析死锁（Deadlock）事件。我们接着就来示范一下如何在 Profiler 中以图形显示死锁的发生。打开 SQL Server Profiler 后点选工具栏上的“ 新建跟踪 ”按钮，接着在“ 连接到服务器 ”窗口设置服务器类型为“ Database Engine ”、设置服务器名称以及登录服务实例的验证方式，完成后点选“ 连接 ”按钮。接着在“ 跟踪属性 ”窗口内点选“ 事件选择范围 ”页签。在“ 事件选择范围 ”标记右下角勾取“ 显示所有事件 ”选项，接着在图2-23上方展开 Locks 事件后勾取“ Deadlock graph ”事件，如图2-23所示。完成后点选“ 执行 ”按钮，开始录制。完成跟踪事件设置后，该工具程序便开始监控目前数据库引擎的运行情况。接下来，笔者使用Management Studio 工具程序，制造死锁事件做为示范。打开 Management Studio 并在该环境内创建两条数据库连接，执行下列语法，模拟死锁环境。在工具栏上重复点选“ 新建查询 ”按钮，打开两个T-SQL 语法的编辑窗口，分别执行下列

语法。图2-23 在“ 跟踪属性 ”窗口内勾选“ Deadlock graph ”事件－－A 窗口begin tran 0update Production.Product set ListPrice=2 where ProductID=1 和－－B 窗口begin tran 0update Production.Product set ListPrice=2 where ProductID=2接着，依序在A 窗口内执行以下语法：0select * from Production.Product where ProductID=2然后B 窗口内执行： 0select * from Production.Product where ProductID=1你会在某条连接执行结果的窗口内发现死锁的报错信息，内容如下： 图2-24 死锁的报错信息当数据库内发生死锁时，Profiler 工具程序会将结果以图表方式显示，如图2-25所示。图2-25 数据库引擎事件录制的结果你可看到图形化死锁发生的原因，描述死锁锁定的资源，连接的细节，以及被牺牲掉的连接。在上图2-23的“ 跟踪属性 ”对话框内勾选“ Deadlock graph ”事件时，Profiler 会自动在“ 事件提取范围 ”标记后方增加一个“ 事件提取设置 ”页签。你可以在该页签内的“ 死锁 XML ”项目下，设置将死锁发生的细节单独存储成 xdl 文件，供后续分析使用。或是在主菜单上选择“ 文件 ”－“ 导出 ”－“ 提取 SQL Server 事件 ”－“ 提取死锁事件 ”，一样也可以将死锁的结果存储起来。使用Management Studio 即可打开 xdl 结果文件，如图2-26所示。SQL Server 2005 的 Profiler 工具可集成操作系统的性能监视器跟踪结果，通过图形化的交互引用显示，提供用户更便利的分析环境。接下来介绍如何将 Windows 系统提供的性能监视器所跟踪的计数器（Performance Counter）数值导入到 Profiler 一并分析。图2-26 使用 Management Studio 显示与分析死锁首先，必须在执行图2-23的跟踪事件中，打开系统的性能监视器，同时设置并进行记录的工作。（路径为“

开始”—“系统管理工具”—“性能”），接着在控制台窗格内点选“计数器日志”后，以鼠标右键点选详细数据窗格空白处，在快捷菜单内选择“新建日志设置”。如图2-27所示：图2-27 通过性能监视器新建记录内容在弹出的窗口内给一个跟踪日志名称，笔者在此命名为 trace，接着在下一个窗口内点选“添加”，如图2-28所示。在这里分别选择了Processor、Memory、PhysicalDisk等对象，各对象下的计数器依序为%Processor Time、Available Mbytes、%Disk Time，整个设置如图2-28所示：图2-28 设置需要跟踪的对象及其下的计数器完成设置后启动该设置文件的记录工作，开始监控目前操作系统内的对象运行情形，并在启动 Profiler 记录后，执行访问 SQL Server 的应用程序，例如 Management Studio。当跟踪一段时间后点选停止按钮，默认系统会将跟踪的事件记录在C盘的 perflogs 文件夹内。接着，将图2-23的结果保存成 xml 数据文件。然后再使用 Profiler 重新打开它，选择主菜单内“文件”—“导入性能数据”菜单。在对话框内选择刚刚性能监视器所存储的数据，即可看到如图2-29的结果。当然，Profiler 和性能监视器两者需要同时录制，也就是两个输出的结果其时间字段位内容需要重叠，否则就没有比较的意义了。在图2-29中，你可以通过鼠标点选最上方窗格内的事件，在中间的窗口中便会显示你所选择录制和当时服务器性能计数器的值。据此可以判断该执行语法与服务器资源使用的因果关系。在以往，我们会需要将 Profiler 和性能计数器的结果都导入到 SQL Server 的数据表内，才容易通过 T-SQL 语法做完整的比较，现在直观且方便多了。