2.2 定义问题

在出现问题时，有了冷静的态度之后，关键就是要定位问题了。这里假定你已经可以区分是否存在性能问题。

如何定位问题？答案就是通过当前现象和日常、临时收集的数据进行分析。这种收集是通过大量工具收集并汇总来实现的。常见的定位工具如下。

- 性能监视器，在第11章介绍。
- 动态管理对象（DMO，SQL Server 2005才出现），贯穿全书。
- SQL Profiler/SQL Trace，在第12章介绍。
- PowerShell，在第12章介绍。
- Extended Events，在第13章介绍。

其他工具将在第14章介绍。可能在本书编写的过程还会出现部分工具而本书未介绍到，如果读者想要进一步了解，请参考相关资料和书籍。

2.2.1 使用工具找到性能瓶颈

可以借助上面介绍的工具，收集数据，并找出异常。比如使用性能监视器监控内存方面的指标，如果发现某个指标值明显超过警戒线，这就需要引起注意，检查其产生的原因了；或者通过DMO，查询等待状态，检查系统正在等待什么资源，通过等待信息一步一步分析问题。

需要注意的是，表象有误导性，不应该通过一两个指标来定位性能问题的根源。当然，这会需要有一定的知识和经验了，但是SQL Server已经问世十几年，大量的问题已经在网上有记载，所以多学习多积累，问题还是不难解决的。

2.2.2 通过性能数据进行分类

性能问题有很多种，包含资源配置或使用问题、设计问题、编码问题、管理问题等。每种问题的处理手段和侧重点都不同，所以需要进行分类，然后再做处理。本书将会介绍在获取到性能数据之后，如何对数据进行分类。

这里需要特别提醒的是，如果处理完一个性能问题之后就以为万事大吉，不做好分类、总结甚至一些处理预案，下一次还是有可能再次出现问题的。所以强烈建议，在处理现有的性能问题时，要对问题进行深度分析，找到根源，尽可能避免重复出现，这样才能做到最终的性能优化。同时，在处理完问题后，把问题归档，即使面对的是一些无法避免的问题，有了解决方法的归档和分类，也可以减少侦测性能瓶颈并搜索解决方法的花费时间和降低难度。

性能问题的主要优化步骤如下。

- 分析实例级别的等待
- 组合等待和队列
- 确定方向，然后确定优化方案
- 细化到数据库、文件级别
- 细化到进程级别
- 优化索引/查询

1．分析实例级别的等待

优化方法论的第一步通常是从实例级别找出是哪些等待类型占用了大部分的等待时间。从SQL Server 2005开始，可以使用DMV（sys.dm_os_wait_stats）来实现，如果是SQL Server 2000，要通过DBCC SQLPERF（WAITSTATS）命令来实现。等待信息是很好的问题切入点，而其他工具可能返回的信息过多，容易产生误导。这部分将在第7章中详细介绍。通过等待信息，可以发现比如锁、闩锁、IO、事务日志、内存等对象相关的等待。但是需要提醒的是，大部分的资源问题（比如CPU、内存、IO）都可能是因为设计不合理、编码低效引起的，并不一定是资源真的有问题。

2．组合等待和队列

一旦找到实例级别最高的几个等待类型，就可以把研究面缩小。接下来就是组合等待类型和队列找出有问题的资源了，这一步主要使用性能监视器，跟踪比如I/O队列、缓存命中率、内存等相关计数器，也可以使用SQL Server 2005开始提供的sys.dm_os_performance_counters这个DMV来查看常用的计数器。对于其他有用但是暂时未提供的计数器，还是需要使用性能监视器来监控。这一步在第11章介绍。

3．确定方向，然后确定方案

通过前面的步骤得到准确的信息之后，就要根据这些信息定好优化的方向（比如索引、查询、设计等），然后确定方案。但是如果你发现存在的是一些资源问题，或者阻塞、编译重编译等问题，那么采取的方案就不一样了。

图2-1　性能优化步骤