解析SQLServer数据体系和应用程序逻辑 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/264/2021_2022__E8_A7_A3_E6_9E_90SQLS_c97_264238.htm 在许多用SQL Server实现的新的企业系统设计中，系统设计师需要在给数据结构和管理应用程序逻辑的定位上做出具有关键性意义的决定。SQL Server有它自己的编程语言（Transact-SQL，即TSQL），开发者可以用它来管理数据访问、代码事务逻辑和交易控制。 使用TSQL，开发者可以创建保存过程，在保存过程中用一段可重用、预编译而且拥有自己的许可设置的代码块来封装数据访问。数据库中每个表格都有一组叫做triggers的特殊的保存过程。当底层数据库发生特定的数据库事件（如Insert、Delete或者Update）时，trigger就被“触发”了。使用triggers，开发者就可以编写基于事件的事务逻辑，这样，给定表格的Insert、Delete和Update事件就可以驱动其它表格的变化。 既然有了这样的灵活性，那么我们为什么不尽可能用TSQL写更多的事物逻辑呢？ 使用TSQL来开发应用程序逻辑存储 TSQL不仅可以作为单个应用程序的逻辑仓库，它也可以是一个访问相同数据的应用程序组的逻辑仓库这有几个逻辑上的原因。通过对数据的集中处理和管理SQL server中数据的规则，你可以配置这样的安全体系即应用程序在通过事务规则之前，不可以访问底层数据库。 这是大多数两层客户服务器应用程序的常见数据库范例。该体系把所有的事务逻辑和数据访问交给后端的服务器而把丰富的表示逻辑交给客户端。客户管理事务过程和数据的视（view），但不在本地处理除显示之外的其它事务。如果把所有的事务逻辑放到中央

仓库去，那么这个体系还有降低管理成本的潜力，但这会付出降低了可测性的代价。 我最近接触了一个客户，它花了数百个人月（一个人工作一个月的工作量）和数以千计的美元来设计一个非常复杂的、用TSQL管理所有应用程序逻辑的应用程序。尽管该体系非常精巧、在10到15个用户的情况下也运行良好，但是如果有20个用户，速度就非常慢。通过给SQL server增加处理器的方法，该系统可以允许60个用户同时使用。但是这距离100个用户的设计目标还有很大一段距离，这就使得该公司在Internet上开放该应用程序的计划无法实施下去。由于存储过程和trigger只能操作本地数据，该公司无法把该应用程序分解成多个SQL server以提高可测性。结果，该公司不得不大规模的修改它。 在应用程序逻辑中使用.NET类 上面那家公司在经过一段曲折后所发现的问题，大多数体系设计师在体系设计阶段都会重新认识到应用程序逻辑包含在一组.NET类的n层体系可以增加该应用程序的灵活性和可测性。由于TSQL是一种以管理数据为主要目的的语言，因此它不够灵活，但是我们仍可以用TSQL编写出复杂的事务逻辑。 如果开发者使用.NET框架，那么他们可以在开发核心事务过程时做出自己的语言选择。这个灵活性可以让你对应用程序要求和开发语言或者资源进行最合理的搭配。而且如果适当开发，封住这些事务过程的对象可以在多台机器上运行并共享同样的底层数据库server。在与处理TSQL事务逻辑无关的情况下，SQL server可以应付大量的并发请求。 行操作（row operation）和集操作（set operations） 在规划体系阶段时判断使用行操作还是集操作的一个指导思想就是：如果使用TSQL就使用集操作，如果使用.NET则进行行操作。通过网

络连接来提供大量的数据会影响应用程序的整体性能，所以只要有可能就使用server来处理它们这样做是很有意义的。但是从内存和处理能力的角度来看，SQL Server的指针（cursor）是非常昂贵的对象，因此创建一个指针来遍历集合中的所有记录并依次处理这些记录一般来说并没有多大意义。 当你需要执行基于行的处理，而这些处理包括了复杂的程序逻辑或者占用CPU比较厉害的操作时，你就应该从server中查询这些行并在中间层来处理它们。 如果你想通过一个例子来看看如何把数据访问逻辑封装到一个中间层对象中去，请从MSDN中下载数据访问应用程序模块。这是一个提供代码的、可重用的数据访问子系统，你可以根据它来编写自己的数据库或者特性应用程序的数据访问对象。 通过创建可重用的.NET应用程序框架来处理大多数应用程序逻辑、并用基于TSQL的保存过程来作为服务器端的集操作的安全限制和机制，那么你就可以创建同时拥有TSQL和.NET这两者优点的应用程序了。 