 作为一名数据库管理员，在进行代码迁移之前，我总是尽力给提交于开发环境的代码一个完整的面貌。但是，不得不承认，我不能保证不发生任何可能破坏开发系统的事情。当这种情况发生时，可能的补救措施是恢复到目标代码的前一版本，目标代码可能是存储过程、函数等等。 如果可能的话，你不想做但又不得不做的事情是从备份的数据库中恢复代码，但是如果备份的数据库存储在磁带上，这种方法可能因花费太长的时间而不能使用。如果数据库庞大的话，要花费相当长的时间来恢复，更不用说你还要找一台足够大的服务器来存储备份的文件。不过，还有更好的方法。 很久前我找到的一种解决方法是备份数据库代码到一个独立的数据表中，这样如果在我们开发的代码发生错误时，就可以从数据表恢复出错的过程或函数。这种方法确实节省了大量的时间。 在SQL Server 2000中，这种方法可以这样实现：对特别的数据库做一个完整的syscomments数据表备份，然后将备份的数据表放入档案表中。我通常保存最近两周的重要过程代码。利用这种技术唯一的麻烦是：如果代码对象十分大，那么可能要对代码进行重构。因为如果代码过大将会被存储到syscomments表中不同的行中，有时这可能是件令人感到头痛的事。 SQL Server 2005 新增加的众多功能之一是可以利用一个系统函数返回某个对象的完整代码，这个系统函数将使得存档你的过程代码变得十分简单。 OBJECT_DEFINITION SQL Server 2005新增的系统函

数OBJECT_DEFINITION根据提供给该函数的对象ID返回对象的TSQL代码。为了更好的理解这个函数的工作过程，让我举个例子。首先我们创建一个用户自定义函数，该函数的脚本如下： CREATE FUNCTION udf_Multiply (@Val1 INT, @Val2 INT) RETURNS INT AS BEGIN DECLARE @RetVal INT SET @RetVal = (@Val1 * @Val2) RETURN(@RetVal) END 这是一个很简单的小函数。因为它仅仅处理两个参数，但是已足够为我们演示OBJECT_DEFINITION函数是如何工作。测试该系统函数的脚本如下： DECLARE @ObjectID INT SET @ObjectID = OBJECT_ID(udf_Multiply) SELECT OBJECT_DEFINITION(@ObjectID) 在这个例子中，我们实际上用了两个系统函数。首先，我们要得到前面创建的udf_Multiply函数的OBJECT_ID，在SQL Server 数据库引擎中，OBJECT_ID是一个对象的系统标识符。然后我们将这个ID传给系统函数OBJECT_DEFINITION，这一系统函数将返回提供给它的ID对象的代码，即返回值是我们以前为udf_Multiply 函数写的TSQL代码。 既然我们对OBJECT_DEFINITION函数的工作原理有了很好的了解，接下来让我们看看如何利用这个函数来存档我们数据库中的过程代码。首先，运行列表A中的脚本程序在测试数据库中创建20个存储过程。 DECLARE @i INT SET @i = 1 WHILE @i BEGIN EXECUTE (IF OBJECT_ID(usp_TestProcedure @i)>0 DROP PROCEDURE usp_TestProcedure @i) EXECUTE (CREATE PROCEDURE usp_TestProcedure @i AS BEGIN PRINT The name of this procedure is CAST(OBJECT_NAME(@@PROCID) AS VARCHAR(20)) END

) SET @i = @i 1 END 你将看到在上面的脚本中，我们使用了动态SQL语句。当创建动态SQl语句时，我习惯用系统存储过程sp_executesql，因为该过程能够很好地在系统中缓存SQL语句。但是，在我们这一例子中，EXECUTE命令就能很好地完成任务。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com