2011 年度十大

SQL Server 技巧文章

2011 年度十大 SQL Server 技巧文章

再过一周就是我们中国的传统节日春节了，在辞旧迎新之际，TechTarget 数据库网站编辑为您总结了过去一年中，关于微软 SQL Server 数据库的十大技巧文章，其中不乏一些 DBA 必须掌握但又容易忽视的知识。也希望通过我们的努力，为广大 DBA 送上更多更精彩的内容。敬请参考：2009 年度十大 SQL Server 技巧文章

SQL Server 目录视图教程

SQL Server 2008 提供了上百个系统视图，这些视图可以让你查看数据库和服务器元数据，而无需访问真正的数据源。系统分类视图的目录在通过数据库引擎提取信息时特别有用。

- SQL Server 目录视图教程：数据库和文件目录视图
- SQL Server 目录视图教程：对象目录视图
- SQL Server 目录视图教程：安全目录视图

深入了解 SQL Server 动态管理视图

随着 SQL Server 2005 版本的发布，带来了一种新的用于访问系统和数据库信息的方式，而无需创建复杂的查询或直接访问系统表。通过使用 SQL Server 动态管理视图(DMVs)，你可以查看 SQL Server 的实例信息，比如实例运行在什么系统上、实例中有那些数据库。

- 深入了解 SQL Server 动态管理视图（上）
- 深入了解 SQL Server 动态管理视图（下）

寻找 SQL Server 数据保护的方法

可用于 SQL Server 数据保护的技术有：更改跟踪，变更数据捕获，复制和数据库镜像，每种技术都有最适合自己的场合。本文不会讨论备份技术，而是讨论如何保存和访问特定对象的历史版本数据，如数据或表的早期版本。

- 寻找 SQL Server 数据保护的方法：变更跟踪
- 寻找 SQL Server 数据保护的方法：变化数据捕获
- 寻找 SQL Server 数据保护的方法：复制与镜像

第三方 SQL Server ETL 工具推介

许多组织都在使用第三方 ETL 工具，即使已经买了 SQL Server 的 License。有些人也许是因为对特定的产品很熟悉，另一些人是因为对 SSIS 不了解，或者他们的数据管理需求用 SSIS 不容易实现，如 ERP 和 CRM 功能。

- 第三方 SQL Server ETL 工具推介（上）
- 第三方 SQL Server ETL 工具推介（下）

SQL Server 数据访问策略

在谈论数据库架构的书籍，论坛，电子杂志甚至是博客上，往往认为设计并实现数据库，规划了高效的安全和索引策略就是数据库架构的全部，很少包括数据库访问策略方面的讨论。本文将讨论基于 SQL Server 作为后台数据库，数据访问方面的策略选择。

- SQL Server 数据访问策略：即席 SQL
- SQL Server 数据访问策略：存储过程
- SQL Server 数据访问策略：CLR

玩转自助式 BI：SQL Server Crescent

在 2010 年 11 月举行的 SQL Server 专家会议(PASS)上，微软公司推出了 Crescent 项目，它是一款新的商业智能(BI)工具，专为使非技术用户能更容易地创建视觉效果良好的报表而设计。它是下一代 SQL Server BI 核心。

- 玩转自助式 BI：SQL Server Crescent（上）
- 玩转自助式 BI：SQL Server Crescent（下）

提高 SQL Server 扩展性

扩展一个跨多个系统的 SQL Server 环境可以说是一项困难且复杂的系统工程，涉及到分区数据库、联邦等等。所以，当涉及到 SQL Server 可扩展性的时候，大多数组织更喜欢在设法解决之前采取扩展独立系统的方法。

- 提高 SQL Server 扩展性：最大化数据库性能
- 提高 SQL Server 扩展性：合理虚拟化
- 提高 SQL Server 扩展性：提升存储性能
- 提高 SQL Server 扩展性：购买新的服务器

把 SQL Server 数据导入到 Excel PowerPivot

微软公司的 PowerPivot for Excel 2010 是一款数据分析工具，可以让你很容易地从 SQL Server 数据库和其他数据源导入数据到 Excel 环境中。由于 PowerPivot 高效的压缩算法和内存分析功能，你可以导入和管理大数量的数据。

- 把 SQL Server 数据导入到 Excel PowerPivot（上）

- 把 SQL Server 数据导入到 Excel PowerPivot（下）

SQL Server 2012 容量管理

由于在大数据集中常常会出现扩展问题，微软公司即将推出的 SQL Server 2012 中大部分针对扩展性和容量管理的新功能都定位专门处理大负载：数据仓库 BI 和决策支持应用程序。把这样一些应用程序整合到一起是相当辛苦的，而维护它们的数据结构就更困难了。

- SQL Server 2012 容量管理：列存储索引
- SQL Server 2012 容量管理：更多的表分区
- SQL Server 2012 容量管理：AlwaysOn
- SQL Server 2012 容量管理：FILESTREAM 和 FileTable

SQL Server 硬件选择不能犯的几个错误

建立全新的 SQL Server 系统可能比较棘手。SQL Server 是真正注重利用硬件的产品，它的性能跟你如何配置服务器有很大的关系，尤其是如何配置你服务器的存储子系统。

- SQL Server 硬件选择不能犯的几个错误

SQL Server 目录视图教程：数据库和文件目录视图

SQL Server 2008 提供了上百个系统视图，这些视图可以让你查看数据库和服务器元数据，而无需访问真正的数据源。系统分类视图的目录在通过数据库引擎提取信息时特别有用。SQL Server 目录视图是视图的集合，为目录元数据提供通用的接口。

通过目录视图你能访问到的信息量是非常广泛的。例如，使用目录视图提取关于特定数据库中表和视图的信息。或者提取关于安全对象，全文检索，数据库镜像，分区或者各种其它目录相关的信息。

SQL Server 目录视图可以按功能线分成子分类，比如数据库对象视图和安全视图。在本文中，我为你提供了这样几种子分类的视图示例。我在 SQL Server 2008 的一个本地实例上创建了这些示例。在那些需要指定具体数据库的例子中，我使用的是“AdventureWorks 2008”数据库。然而，要注意，我在示例查询中展示的返回结果是与我的 SQL Server 实例有关的。如果你在你自己的环境中运行这些示例，你可能会得到不同的结果。

对于我演示的每个目录视图(以及所有一般目录视图)，SQL Server 在 SQL Server 联机丛书上提供了针对这些视图的帮助主题。由于每个视图返回的信息类型不同，你会希望通过访问该主题阅读每个视图的详细信息的。

数据库和文件目录视图

我们要了解的第一组 SQL Server 目录视图允许你访问关于 SQL Server 实例中安装的数据库信息，以及与那些数据库相关的文件信息。在下面的例子中，我利用“sys.databases”目录视图来提取数据库列表，要求查询名称以“adventureworks”开头的数据库。

```
SELECT

name AS DbName,
```

```
database_id AS DatabaseID,
user_access AS UserAccess,
user_access_desc AS AccessDescript
FROM
sys.databases
WHERE
name LIKE 'adventureworks%';
```

正如“SELECT”语句中显示的，你访问目录视图就跟你访问其它视图一样。在这种情况下，我在“FROM”从句中使用了“sys.databases ”视图来提取实例数据库的详细信息。要得到该视图支持的列清单，以及每列的描述信息，请参见该视图在 SQL Server 联机丛书中的主题。这些描述信息包括在列中用作值的代码解释。例如，该主题解释了存储在“user_access ”列中的代码，展示了数据库是允许多用户访问(用“0”表示)，还是只允许单用户访问(用“1”表示)，还是限制用户访问(用“2”表示)。

在上面的“SELECT”语句中提取了每个“adventureworks”数据库的名称和 ID，还有用户访问代码和代码描述。下面的表展示了我在我的 SQL Server 2008 本地实例中运行该查询时返回的结果：

DbName	DatabaseID	UserAccess	AccessDescript
AdventureWorks	7	0	MULTI_USER
AdventureWorks2008	8	0	MULTI_USER
AdventureWorksDW2008	10	0	MULTI_USER

可见，我在我的系统中安装了三个“adventureworks”数据库。默认情况下，用户访问代码“0”被赋给每个数据库。用户访问代码描述显示在列“user_access_desc”，它表示每个数据库都支持“MULTI_USER”用户访问。

SQL Server 还提供了一个目录视图，允许你提取关于数据库文件的信息。在下面的“SELECT”语句中，我利用“sys.master_files”视图来访问有关“AdventureWorks 2008”数据库文件的详细信息：

```
SELECT
file_id AS FileID,
name AS LogicalName,
type AS FileType,
type_desc AS TypeDescript
FROM
sys.master_files
WHERE
database_id IN
(
SELECT database_id
FROM sys.databases
WHERE name = 'AdventureWorks2008'
);
```

对于与“AdventureWorks2008”数据库关联的每个文件，我都提取了文件 ID，文件的逻辑名称，文件类型代码，以及解释类型代码的描述。另外，请参考与“sys.master_files”目录视图有关的主题获取关于该列的更多信息。

请注意，为了提取正确的数据，我的“WHERE”从句中包含有一个从“sys.databases ”查询数据库 ID 的子查询。这使我可以访问文件信息，无需知道 ID，只要知道数据库名称即可。

SQL Server 元数据函数

SQL Server 提供了一组元数据函数，可以使你执行诸如提取数据库 ID 这类操作。在本文中，我没有使用元数据函数，这样我可以更好地展示各种目录视图。您可以在 SQL Server 联机丛书的“元数据函数(Transact-SQL)”专题中找到关于这些函数的更多介绍。比如函数“DB_ID”可以提取数据库 ID，有助于避免使用子查询。

在我提取了数据库 ID 之后，我可以限制只查询我指定的数据库。下面的结果展示了关于“AdventureWorks 2008”数据库的返回信息：

FileID	LogicalName	FileType	TypeDescript
1	AdventureWorks2008_Data	0	ROWS
2	AdventureWorks2008_Log	1	LOG
65537	FileStreamDocuments	2	FILESTREAM

另一个 SQL Server 目录视图是“sys.database_files”。这个视图与“sys.master_files”视图类似，只是它需要指定一个数据库。在下面的例子中，我首先使用了一个“USE”语句，明确使用“AdventureWorks 2008”数据库，然后我运行“SELECT”语句来从“sys.database_files”提取数据：

在本例中，我提取了文件 ID，逻辑文件名，文件类型，当前状态，以及文件大小。下面的结果展示了该语句返回的信息(使用的是我的系统环境)：

FileID	FileName	TypeDescript	StateDescript	Size
1	AdventureWorks2008_Data	ROWS	ONLINE	24928
2	AdventureWorks2008_Log	LOG	ONLINE	256
65537	FileStreamDocuments	FILESTREAM	ONLINE	0

正如查询结果所展示的，有三个文件与“AdventureWorks 2008”数据库有关联。通过使用“ sys.database_files”目录视图，我能很容易地访问每个文件中的数据。您可以在 SQL Server 联机丛书中的“数据库与文件目录视图(Transact-SQL)”中获得关于该话题的更多信息，然后再查询指定的视图。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

SQL Server 目录视图教程：数据库和文件目录视图

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45246.htm

SQL Server 目录视图教程：对象目录视图

下一个 SQL Server 目录视图的子类型是对象视图。顾名思义，这些视图返回数据库对象的信息。这些视图中最有用的就是“sys.objects”，它可以返回指定数据库中用户定义的，模式范围内的对象。例如，在下面的“SELECT”语句中，我使用“sys.objects”视图来提取“AdventureWorks ”数据库中唯一性约束清单：

```
USE AdventureWorks2008;
GO
SELECT
a.name AS ObjectName,
(
SELECT b.name
FROM sys.objects b
WHERE b.object_id = a.parent_object_id
) AS ParentTable
FROM
sys.objects a
WHERE
a.type = 'UQ';
```

在该“SELECT”语句中，我提取了唯一性约束名称和与这些限制关联的父对象(表)的名称。为了提取该表的名称，我创建了一个子查询，把父对象 ID 与常规对象 ID 进行了关联。(父对象和子对象都能在“sys.objects”视图中查到。)

我还在该语句中加上了“WHERE”从句，只返回那些值类型是“UQ”的行。“UQ”代码代表的是唯一性约束(unique constraint)。正如下面的结果所示，“AdventureWorks 2008”数据库只包含唯一性约束，是在 document 表中定义的。

ObjectName	ParentTable
UQ__Document__F73921F730F848ED	Document

在对象子类别中另一个有用的视图是“sys.tables”，它返回数据库中配置的用户定义的表清单。下面的“SELECT”语句使用该视图在“AdventureWorks 2008”数据库对象“HumanResources ”中提取表清单：

```
USE AdventureWorks2008;
GO
SELECT
name AS TableName,
is_replicated AS IsReplicated,
is_tracked_by_cdc AS IsTracked
FROM
sys.tables
WHERE
schema_id IN
```

```
(
SELECT schema_id
FROM sys.schemas
WHERE name = 'HumanResources'
);
```

如你所见，我提取了表名称，以及它们是被 Change Data Capture 跟踪的还是复制的。请注意，为了限制只查询“”对象，我在“WHERE”从句中包含了一个子查询，基于对象名称提取对象 ID。下表展示了通过“sys.tables ”视图查询出来的结果：

TableName	IsReplicated	IsTracked
Department	0	0
Employee	0	0
EmployeeDepartmentHistory	0	0
EmployeePayHistory	0	0
JobCandidate	0	0
Shift	0	0

正如结果所示，在“HumanResources ”对象中有六个表。“is_replicated”和“is_tracked_by_cdc”列都被配置为 bit 列，其中“0”表示 false，“1”表示 true。表中数据全为“0”意味着没有一个表是被 Change Data Capture 复制的或者跟踪的。

下面这个例子与上面的类似，只是我使用了“sys.procedures”视图来提取“HumanResources ”对象中包含的存储过程清单：

```
USEAdventureWorks2008;
GO
```

```
SELECT
name AS ProcName,
type AS ProcType,
type_desc AS ProcDescript
FROM
sys.procedures
WHERE
schema_id IN
(
SELECT schema_id
FROM sys.schemas
WHERE name = 'HumanResources'
);
```

在本例中，我提取了存储过程的名称和类型(包括存储过程代码和描述)。与前面一样，我在“WHERE”从句中使用了子查询来限制只获取指定对象的数据。该“SELECT”语句返回结果如下表：

ProcName	ProcType	ProcDescript
uspUpdateEmployeeHireInfo	P	SQL_STORED_PROCEDURE
uspUpdateEmployeeLogin	P	SQL_STORED_PROCEDURE
uspUpdateEmployeePersonalInfo	P	SQL_STORED_PROCEDURE

如你所见，结果只包含 SQL 存储过程。然而，“sys.procedures”视图还将返回“公共语言运行时”存储过程，外部存储过程，以及复制过滤器存储过程，如果数据库中有的话。

如果你想提取表或视图中关于列的信息，请使用“sys.columns”视图。在下面的例子中，我提取了“AdventureWorks 2008”数据库中名称包含“history”(模糊匹配)的所有对象，表和列的名称：

```
USEAdventureWorks2008;
GO
SELECT
s.name AS SchemaName,
o.name AS ViewName,
c.name AS ColumnName
FROM
sys.columns c
INNER JOIN sys.objects o
ON c.object_id = o.object_id
```

```
INNER JOIN sys.schemas s

ON o.schema_id = s.schema_id

WHERE

o.type = 'V' AND

o.name LIKE '%history%'

ORDER BY

s.name, o.name, c.name;
```

为了提取视图和对象的名称，我对“sys.columns”视图和“sys.schemas”视图做了联合查询。然后，我使用“WHERE”从句限制查询类型 V 的对象(视图 views)要求对象名称中包含“history”。从执行结果来看，“AdventureWorks 2008”数据库只包含一个带有“history”字样的视图，如下面结果所示：

SchemaName	ViewName	ColumnName
HumanResources	vEmployeeDepartmentHistory	BusinessEntityID
HumanResources	vEmployeeDepartmentHistory	Department
HumanResources	vEmployeeDepartmentHistory	EndDate
HumanResources	vEmployeeDepartmentHistory	FirstName
HumanResources	vEmployeeDepartmentHistory	GroupName
HumanResources	vEmployeeDepartmentHistory	LastName
HumanResources	vEmployeeDepartmentHistory	MiddleName
HumanResources	vEmployeeDepartmentHistory	Shift
HumanResources	vEmployeeDepartmentHistory	StartDate
HumanResources	vEmployeeDepartmentHistory	Suffix
HumanResources	vEmployeeDepartmentHistory	Title

SQL Server 还提供了一个目录视图，允许你查看数据库中配置的索引。例如，在下面的“SELECT”语句中，我使用“sys.indexes”视图提取“AdventureWorks 2008”数据库“Person”表中定义的所有索引信息：

```
USEAdventureWorks2008;
GO
SELECT
name AS IndexName,
type_desc AS IndexType,
is_primary_key AS PrimaryKey,
is_unique AS UniqueIndex
FROM
sys.indexes
WHERE
object_id IN
(
SELECT object_id
FROM sys.objects
WHERE name = 'person'
);
```

在本例中，我提取了“Person”表每个索引的名称和类型，以及该索引是主键索引还是唯一索引的信息。为了限制只查询“Person”表的数据，我在“WHERE”从句中使用了子查询来提取与“Person”表关联的对象 ID。下面的结果展示了该表定义的索引：

IndexName	IndexType	PrimaryKey	UniqueIndex
PK_Person_BusinessEntityID	CLUSTERED	1	1
IX_Person_LastName_FirstName_MiddleName	NONCLUSTERED	0	0
AK_Person_rowguid	NONCLUSTERED	0	1
PXML_Person_AddContact	XML	0	0
PXML_Person_Demographics	XML	0	0
XMLPATH_Person_Demographics	XML	0	0
XMLPROPERTY_Person_Demographics	XML	0	0
XMLVALUE_Person_Demographics	XML	0	0

我在这里展示的例子只是 SQL Server 所提供对象目录视图中的一部分。还有与对象视图相关的装配模块，计算列，时间，识别列，触发器，同义词以及其他对象类型。要获取所有对象视图的清单，请参看 SQL Server 联机丛书相关主题“对象目录视图 (Transact-SQL)”。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

SQL Server 目录视图教程：对象目录视图

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45247.htm

SQL Server 目录视图教程：安全目录视图

另一个重要的目录视图子分类就是安全视图，这类视图允许你查看关于数据库安全，服务器安全，加密以及审计等方面的信息。例如，你可以利用“sys.database_principals”视图来提取关于数据库中每个安全原则相关的信息。在下面的“SELECT”语句中，我使用这个视图来提取“AdventureWorks 2008 ”数据库中每项安全规则的类型和名称：

```
USE AdventureWorks2008;

GO

SELECT

type_desc AS PrincipalType,

name AS PrincipalName

FROM

sys.database_principals

ORDER BY

type_desc, name;
```

如你所见，该语句非常简单。我只想提取类型和名称并按相应的结果排序，返回结果如下表所示：

PrincipalType	PrincipalName
DATABASE_ROLE	db_accessadmin
DATABASE_ROLE	db_backupoperator
DATABASE_ROLE	db_datareader
DATABASE_ROLE	db_datawriter
DATABASE_ROLE	db_ddladmin
DATABASE_ROLE	db_denydatareader
DATABASE_ROLE	db_denydatawriter
DATABASE_ROLE	db_owner
DATABASE_ROLE	db_securityadmin
DATABASE_ROLE	public
SQL_USER	guest
SQL_USER	INFORMATION_SCHEMA
SQL_USER	sys
WINDOWS_USER	dbo

另一个与具体数据库有关的安全视图是“sys.database_permissions”视图。不说你也知道，该视图返回关于数据库权限的信息。例如，在下面的“SELECT”语句中，我利用“sys.database_permissions”视图来提取在“AdventureWorks 2008”数据库中赋予“SELECT”权限的数量统计：

```
USEAdventureWorks2008;

GO

SELECT

class_desc AS ClassName,

permission_name AS PermissionName,

state_desc AS StateName,
```

```
COUNT(*) AS NumberOfSelect
FROM
sys.database_permissions
WHERE
type = 'SL' AND
state = 'G'
GROUP BY
class_desc,
permission_name,
state_desc;
```

在该语句中，我按照“class，permission 和 state”对结果进行分组，这样就可以得到合计值。我利用“WHERE”从句来限制只查询权限类型“SELECT(SL)”和状态为“GRANT(G)”的数据。如下面结果所示，129 个“SELECT”权限已经被授予到了“AdventureWorks 2008”数据库中：

ClassName	PermissionName	StateName	NumberOfSelect
OBJECT_OR_COLUMN	SELECT	GRANT	129

你还可以利用安全视图来提取服务器级别的信息。例如，在下面的例子中我利用“sys.server_principals ”视图来提 SQL Server 本地实例中的安全原则类型和名称：

```
SELECT
type_desc AS PrincipalType,
name AS PrincipalName
```

```
FROM

sys.server_principals

WHERE

type <> 'C'

ORDER BY

type_desc, name;
```

请注意，在这个语句中，我使用了“WHERE”从句把类型为“C”的数据从结果中排除掉了。类型 C 表示认证映射登录，我不想把这类数据包含进来。下面的结果展示了“sys.server_principals”视图返回的信息：

PrincipalType	PrincipalName
SERVER_ROLE	bulkadmin
SERVER_ROLE	dbcreator
SERVER_ROLE	diskadmin
SERVER_ROLE	processadmin
SERVER_ROLE	public
SERVER_ROLE	securityadmin
SERVER_ROLE	serveradmin
SERVER_ROLE	setupadmin
SERVER_ROLE	sysadmin
SQL_LOGIN	##MS_PolicyEventProcessingLogin##
SQL_LOGIN	##MS_PolicyTsqlExecutionLogin##
SQL_LOGIN	sa
WINDOWS_LOGIN	BOBE024\Administrator
WINDOWS_LOGIN	NT AUTHORITY\LOCAL SERVICE
WINDOWS_LOGIN	NT AUTHORITY\SYSTEM

你还可以通过“sys.server_permissions”视图查看服务器权限明细，请看下面的示例：

```
SELECT
class_desc AS ClassName,
permission_name AS PermissionName,
state_desc AS StateName,
COUNT(*) AS NumberOfGrant
FROM
sys.server_permissions
WHERE
type = 'CO' AND
state = 'G'
GROUP BY
class_desc,
permission_name,
state_desc;
```

在该语句中，我提取了已经赋予我本地 SQL Server 实例的“CONNECT”权限数量。正如你在前面的例子中看到的，我将按“class，permission 类型和 state”分组，这样就可以得出结果，请看下面的结果：

ClassName	PermissionName	StateName	NumberOfGrant
ENDPOINT	CONNECT	GRANT	4

除了我在这里展示的安全视图，SQL Server 还支持大量其它视图。你可以在 SQL Server 联机丛书“系统视图(Transact-SQL)”中看到那些视图的清单。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

SQL Server 目录视图教程：安全目录视图

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45248.htm

深入了解 SQL Server 动态管理视图（上）

随着 SQL Server 2005 版本的发布，带来了一种新的用于访问系统和数据库信息的方式，而无需创建复杂的查询或直接访问系统表。通过使用 SQL Server 动态管理视图(DMVs)，你可以查看 SQL Server 的实例信息，比如实例运行在什么系统上、实例中有那些数据库。

你可以在 Transact-SQL 语句中引用你想要查看的视图名称来调用 DMVs。所有的 DMVs 都存放在 SYS 架构中，这个预定义架构包含了系统视图。它们以字符 dm_开头，如 sys.dm_os_hosts。

与其他类型的视图一样，SQL Server DMVs 返回一个指定数据类型的列集合。但是，一个 DMV 的架构会随着 SQL Server 版本的变化而变化。因此，当编写调用 DMV 的代码时，你应该指定列名而不是使用类似 SELECT * FROM view_name 之类的语法。

SQL Server 动态管理视图可分为两大类：一类是描述数据库级别的信息，另一类是描述服务器级别的信息。在本文中，我将演示如何使用前者，后者我将会在以后的文章中加以阐述。本文例子所使用的系统环境是一个创建在本地的 SQL Server 2008 实例。例子中引用的数据库是 AdventureWorks 2008 示例数据库，除了一个例子使用了 SQL Server Reporting Services 数据库。对于每一个例子，我都会列出语句运行的结果集。很可能你的结果集会有所不同，这取决于你在哪种系统上运行 SQL Server 以及如何修改数据库的。

与数据库相关的 SQL Server DMVs

我们查看的第一个 SQL Server DMV 是 sys.dm_db_partition_stats，它返回当前数据库中的分区信息，每个分区一行。(如果一个索引、一个堆或者一个没有索引的表未被分区，它被认为是一个分区。)该视图返回的信息包括分区使用的页面数量和总行数。在下面的 SELECT 语句中，我使用这个 DMV 来获取 AdventureWorks 2008 数据库中 Production.Product 表的分区信息：

```
USE AdventureWorks2008;
```

```
GO

SELECT

index_id AS IndexID,

partition_number AS PartitionNum,

used_page_count AS UsedPageCount,

row_count AS TotalRows

FROM

sys.dm_db_partition_stats

WHERE

object_id = OBJECT_ID('Production.Product')

ORDER BY

IndexID,

PartitionNum;
```

如语句所示，sys.dm_db_partition_stats 动态管理视图返回的信息首先是索引或堆的识别号然后是分区号。换句话说，定义的分区号与特定的索引或堆相关，所以每个索引或堆可以有一个或多个分区。(这个分区号不同于数据库分配给每个分区的独一无二的分区 ID。)

紧接着索引 ID 和分区号，还将返回该分区的页面总数和使用的总行数，结果如下表所示：

IndexID	ParitionNum	UsedPageCount	TotalRows
1	1	15	504
2	1	4	504
3	1	5	504
4	1	4	504

如上表所示，在 PRODUCT 表中每个索引或堆都只有一个分区，每个分区有 504 行。

接下来的动态管理视图，我会介绍 sys.dm_sql_referenced_entities。该视图返回指定对象所引用的用户定义的数据库对象集合，每个对象一行。例如，如果一个存储过程引用了用户定义的表，则动态管理视图将为每个表返回一行。

（作者：Robert Sheldon 译者：沈宏 来源：TT 中国）

深入了解 SQL Server 动态管理视图（上）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45832.htm

深入了解 SQL Server 动态管理视图（下）

实际上，sys.dm_sql_referenced_entities 并不是一个视图，它是一个函数。在下面的 SELECT 语句中，我使用 sys.dm_sql_referenced_entities 函数来检索触发器 iuPerson 所引用的对象列表，如下所示：

```
USE AdventureWorks2008;

GO SELECT referenced_schema_name AS SchemaName,

referenced_entity_name AS EntityName,

referenced_class_desc AS ClassName

FROM sys.dm_sql_referenced_entities

('Person.iuPerson', 'OBJECT')

ORDER BY

SchemaName,

EntityName;
```

触发器 iuPerson 是 Person 架构的一部分，与 Person 表相关。当我调用该函数时，需要指定触发器名称和 OBJECT 参数选项。如果它是一个数据库级触发器，则需要指定 DATABASE_DDL_TRIGGER 参数选项。如果它是一个服务器级触发器，则需要指定 SERVER_DDL_TRIGGER 参数选项。下表显示 SELECT 语句返回的信息：

SchemaName	EntityName	ClassName
NULL	Person	OBJECT_OR_COLUMN
NULL	Person	OBJECT_OR_COLUMN
Demographics	exist	OBJECT_OR_COLUMN
Person	Person	OBJECT_OR_COLUMN
Person	Person	OBJECT_OR_COLUMN
Person	Person	OBJECT_OR_COLUMN

请注意，返回的是 AdventureWorks 2008 数据库中 iuPerson 触发器所引用对象的架构名称、实体名称和类别。

另一个你可能会发现有用的数据库管理函数是 sys.dm_sql_referencing_entities，它返回数据库中的某个实体引用另一个用户定义实体的信息，每个实体一行。例如，在下面的语句中使用该函数来检索 Production.ProductInventory 表所引用对象的架构、实体及类别：

```
USE AdventureWorks2008;

GO

SELECT

referencing_schema_name AS SchemaName,

referencing_entity_name AS EntityName,

referencing_class_desc AS ClassName

FROM

sys.dm_sql_referencing_entities
```

```
('Production.ProductInventory', 'OBJECT')

ORDER BY

SchemaName,

EntityName;
```

与前面的例子中，我提供的参考对象是 ProductInventory 表，并使用了 OBJECT 参数选项。下面的结果表明，该表引用了四个用户定义的对象：

SchemaName	EntityName	ClassName
dbo	fn_inventory	OBJECT_OR_COLUMN
dbo	ufnGetStock	OBJECT_OR_COLUMN
Production	CK_ProductInventory_Bin	OBJECT_OR_COLUMN
Production	CK_ProductInventory_Shelf	OBJECT_OR_COLUMN

还有一个动态管理函数是 sys.dm_db_index_physical_stats，它返回指定表或视图的索引和数据的大小和碎片信息。在下面的 SELECT 语句中，我将数据库名称和表名称作为函数的前两个参数：

```
USE AdventureWorks2008;

GO

SELECT

index_id AS IndexID,

index_type_desc AS IndexType,

fragment_count AS FragCount,

page_count AS TotalPages
```

```
FROM
sys.dm_db_index_physical_stats
(DB_ID('AdventureWorks2008'), OBJECT_ID('Person.Person'),
NULL, NULL, NULL)
ORDER BY
IndexID;
```

正如你所看到的，sys.dm_db_index_physical_stats 还需要三个额外的参数，在例子中已经被指定为 NULL 值。这三个参数分别代表：索引 ID、分区号和用于获取统计信息的扫描模式。因为我想了解表上所有索引和分区的信息，我为前两个参数指定了 NULL 值。关于模式，NULL 值表示将使用有限的扫描模式，而不是详尽或采样扫描模式。SELECT 语句返回以下结果：

IndexID	IndexType	FragCount	TotalPages
1	CLUSTERED INDEX	5	3807
2	NONCLUSTERED INDEX	2	103
3	NONCLUSTERED INDEX	2	57
256000	PRIMARY XML INDEX	1	3
256001	PRIMARY XML INDEX	7	2152
256002	XML INDEX	33	1386
256003	XML INDEX	33	1385
256004	XML INDEX	35	1386

返回结果包括与 Person 表相关的索引 ID、索引类型、碎片数量和页数。

现在，让我们回到SQL Server动态管理视图。动态管理视图sys.dm_db_index_usage_stats返回自上一次启动SQL Server服务启动以来不同的索引操作类型的统计信息，如检索和扫描。在下面的SELECT语句中，我查看本地SQL Server实例中最近的索引操作使用的统计信息。

```
USE AdventureWorks2008;
GO
SELECT
OBJECT_NAME(object_id) AS ObjectName,
index_id AS IndexID,
user_seeks AS UserSeeks,<
user_scans AS UserScans
FROM
sys.dm_db_index_usage_stats
WHERE
database_id = DB_ID('AdventureWorks2008')
ORDER BY
ObjectName,
IndexID;
```

注意，对于每个操作可查询到对象的名称、该对象对应的索引 ID 以及用户检索和扫描的次数。下面的结果表明，自 SQL Server 服务实例启动以来只有较少数量的检索和扫描：

ObjectName	IndexID	UserSeeks	UserScans
Document	1	1	0
JobCandidate	1	1	0
ProductReview	1	1	0
SalesOrderHeader	1	0	1

（作者：Robert Sheldon 译者：沈宏 来源：TT 中国）

深入了解 SQL Server 动态管理视图（下）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45833.htm

寻找 SQL Server 数据保护的方法：更改跟踪

大多数的 DBA 都在思考这个问题。在许多数据或表的架构发生巨大变化的时候，确实需要看看在数据变化之前和之后发生了什么。一个精心编写的数据库应用程序可以有自己的日志功能，或者会干脆地设定一种从来不进行数据覆盖的方法。如果没有采取任何措施，你还可以找出备份并与当前版本进行比较，但这意味着是一件令人头痛的事情。然而是否存在原生的 SQL Server 功能来找出不同之处呢?

可用于 SQL Server 数据保护的技术有：更改跟踪，变更数据捕获，复制和数据库镜像，每种技术都有最适合自己的场合。在这里我不会讨论备份技术，而是讨论如何保存和访问特定对象的历史版本数据，如数据或表的早期版本。

更改跟踪

SQL Server 更改跟踪是一种相当简单的方法。当该功能被启用时，任何时候一个数据操作语言(DML)动作(插入、更新或删除)发生，它就会记录哪张表的哪一行被修改以及事务发生的时间和日期;如果是更新操作还会记录哪些列被修改(前提是已经启用列跟踪)。可以查询 sys.dm_tran_commit_table 系统视图获得这些跟踪信息，还可以反过来用于从特殊命名的内部表获得给定表的信息，只要你在创建给定表时已经启用更改跟踪。

默认情况下，更改跟踪记录保持两天时间。你可以通过 ALTER DATABASE 命令或 SQL Server Management Studio 管理工具的数据库属性窗口对保留时间进行修改。最短的时间是一分钟;虽然在理论上没有最大值，但是在一个高访问量的数据库中设置该属性超过两天，我对此持怀疑态度，因为系统对象的增长存在失控的可能性。

更改跟踪最大的缺点是：很多关于变化数据和数据模式的信息不会被保留。例如，更改跟踪不保留任何关于给定行在你修改之前它的值是什么的信息。仅有该行最新的数据是什么，就是表本身。如果你删除一张表，所有关于该表的更改跟踪信息也被删除。

另一个缺点，但这是次要的，更改跟踪特性只适用于数据库兼容级别在 90 或更高的。如果小于 90，你仍然可以设置更改跟踪选项，但试图通过 CHANGETABLE 功能获得跟踪信息将会产生错误。

为此，更改跟踪不能作为有效的数据保护措施。它是一种有效的收集表统计信息的方式且其变化影响较小。如果你想在一个给定的数据库中获得最多或最少更新的列或表的真实信息，并想在此基础上优化你的数据库设计，采用这个特性就很方便。

（作者：Serdar Yegulalp 译者：沈宏 来源：TT 中国）

寻找 SQL Server 数据保护的方法：更改跟踪

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45688.htm

寻找 SQL Server 数据保护的方法：变化数据捕获

SQL Server 变化数据捕获不同于更改跟踪，因为它既保存 DML 所做的更改又保存被修改的数据。不像更改跟踪，被捕获的数据以与其表结构相同的格式进行存储(除了一些额外的列用于存储元数据)。其结果可通过一些表值函数进行查询，变化数据捕获是通过事务日志进行异步跟踪的，因此其跟踪过程对系统性能影响不大。默认情况下，变更数据被保留三天，但这可以逐步延长。

变化数据捕获也做出一个略微强一点的尝试试图存储在捕获过程中架构变化的信息。在变化数据捕获过程中任何新添加的列不被记录，任何删除的列将继续捕获并以 NULL 值替代。但是架构的变化也会被跟踪并存放在一张单独的表中，所以如果在捕获过程中表结构发生改变，它就会为给定表创建多个捕获表。因此如果添加或删除列，你将存在捕获给定表新旧结构的多个实例。但在任何时候一张表上不能存在多于两个的活跃捕获实例。

变化数据捕获应用程序就像一个数据保护系统要比单纯的更改跟踪更为实用，因为你可以通过变化数据捕获的存储过程来访问修改后的数据。但它并不是作为整个数据库的副本来执行真正的工作，它可以让你指定的时间点，看到过去数据的快照。(从理论上讲，可以编写一个应用程序利用变化数据捕获规划中的应用接口去做这项工作，但它不是开箱即用的。)

此外，对于如何与其他方法的协同工作有一些限制。变化数据捕获将其信息存储在不能复制的系统表中，更改跟踪信息只对源系统进行访问。

变化数据捕获对增量数据的加载和同步是很有用的，有时微软将此作为一个案例。最后，变化数据捕获在需要连续的变化流用于分析其他资源的场合相当有用，能够看到给定数据的单点实时状态。

(作者：Serdar Yegulalp 译者：沈宏 来源：TT 中国)

寻找 SQL Server 数据保护的方法：变化数据捕获

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45689.htm

寻找 SQL Server 数据保护的方法：复制与镜像

复制

复制的好处之一是：它可以让你合并来自多个数据源的数据。所以，如果你想保护来自多个数据库或数据提供者的数据状态，复制允许你覆盖一些数据。但是它作为一种 SQL Server 数据保护方法的价值取决于你使用哪种复制类型。

每当源数据库(“发行者”)的事务完成后，事务复制就将变化信息推送到目标数据库(“订购者”)。这允许低延迟更新的订购者获得中间的变化信息(如触发前后的数据状态)。微软建议：当需要一个数据库的只读副本，事务复制是最好的。

除复制是双向的之外，合并复制与事务复制相同：来自发行者和订购者的变化被跟踪，两者之间的冲突自动解决。然而，合并复制不跟踪中间数据状态;它仅跟踪给定行的最终值。

就 SQL Server 数据保护而言，当你需要对整个对象保持某种程度的不断更新的副本时，这些模式都是最适合的。它们并不适用于在指定时刻保留对象的独立状态。为此，你需要快照复制。正如其名称所示，快照复制获得某个时刻某个对象的快照并复制其状态，提供了完整的数据刷新。这是很好的进行实时数据保护的方式，但在生成快照并将其推送到订购者时需要一定的开销费用。基本上是在对整个对象生成一个副本时产生的开销，而且对那些不是太大的表是最好的。此外，相对于更改跟踪或变化数据捕获，它在对象变化的历史记录方面只有很少的量;快照是静态的。要获得历史信息只有将两个快照进行比较，这可能会很笨拙。

数据库镜像

从现象上看，SQL Server 数据库镜像类似于复制。一个数据库的两个副本被保存在单独的计算机上，并保持同步更新。所不同的是，镜像副本可以在数据库发生故障时使用，因此，如果你关注数据的可用性，防止在服务器离线的时候可能发生的连接丢失，那么最好使用它。在这种情况下，获得可用性有两种方法：这不是关于如何从数据库中获得数据的问题，而是能够发布变更的问题。

结论

手头上有这些技术的一个好处是它们可以混合以匹配你的 SQL Server 对数据保护的需要。变化数据捕获用于逐项修改特定行的数据，何时什么被覆盖，复制用于确保所有数据的最新副本并以一种统一的方式保持可用。所有这些方法的关键是在一个单一的、一致的数据视图与一个连续的变化信息流之间进行权衡。你需要知道要什么样的保护及其它在什么环境中。

（作者：Serdar Yegulalp 译者：沈宏 来源：TT 中国）

寻找 SQL Server 数据保护的方法：复制与镜像

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_45691.htm

第三方 SQL Server ETL 工具推介（上）

如果你的数据大部分都存在于 SQL Server 数据库中，而且源数据很容易访问到，那么使用 SSIS 执行抽取、转换、加载(ETL)操作是很好的。SSIS 是 SQL Server 中免费带的，很容易安装，是一个很棒的 ETL 工具。

然而，许多组织都在使用第三方 ETL 工具，即使已经买了 SQL Server 的 License。有些人也许是因为对特定的产品很熟悉，另一些人是因为对 SSIS 不了解，或者他们的数据管理需求用 SSIS 不容易实现，如 ERP 和 CRM 功能。

这样的产品一定不少。事实上，世面有太多的 ETL 工具，选型工作也是很烦琐的任务。考虑到这点，我这里主要针对五种流行的产品来进行 ETL 工具的比较。

在选择这些工具的时候，我先浏览了很多在线讨论和文章评论，对哪种产品存在什么样的问题有了一个基本认识。我排除了任何与特定数据库管理系统绑定的工具，例如 Oracle 的 OWB。注意，我并没有要支持或推荐某一款产品，也没有对这些产品的排名有任何倾向性。

Informatic PowerCenter

很少有产品像 Informatic PowerCenter 那样获得如此多的关注。PowerCenter 具有友好的用户接口和面向服务的架构，能支持数据迁移和数据集成、复制、同步和数据仓库。可以用 PowerCenter 从多种业务系统中检索数据，以批量方式在企业范围内，按照实时或者是按需的方式交付数据。

这个产品也支持全面的报告和审计能力。此外，整个组织的开发团队能跨项目和平台的方式共享和重用数据定义及数据映射逻辑。因为 PowerCenter 是元数据驱动的架构，定义可以是标准化的，可以将技术和业务元数据集成进一个单一的数据集成目录中。

SAS 企业数据集成服务器

数据集成是 SAS 企业数据集成服务器的简称。具有集成的开发和工作流结构，这个 ETL 工具可以访问多种数据源，包括数据库、企业应用、主流的数据源、消息中间件、结构化和半结构化的数据、静态和流 WEB 数据，以及许多文件类型等。所有这些数据源都在使用其本身的格式和开放的标准。

这个工具也能在数据库、应用程序、主机遗留文件以及其他数据源之间迁移和同步数据。由于是多线程并行处理结构，集成服务可以支持快速高效地移动数据，大量的关于数据的工作都可以用这个工具完成。转换库中有 300 多个预定义的表和列转换，在数据转换期间，能捕获和记录源数据和数据集成流程。公用的元数据仓库可以进行集中的存储和流程管理，这样就可以实现重用，减少开发的工作量。

BO 数据集成器

用户在寻找一个系统来解决他们的 ERP、CRM、BI 和数据迁移需求的时候，他们常常会找到 BODI。这个产品提供了企业元数据的统一视图，支持高级数据设置方便用户能理解数据结构，内容和数据质量。元数据集成支持端到端的影响分析，可以看到通过对源数据的改变如何来影响 ETL 和 BI 环境。

集成也可以让用户沿着 ETL 过程来审计数据。元数据集成的另外一个优点是产品固有的数据血统，让用户看到数据是如何计算的，何时更新的，数据来源是什么。象其他产品一样，数据集成支持团队开发并提供了一个集中的元数据仓库。

(作者：Robert Sheldon 译者：包春霞 来源：TT 中国)

第三方 SQL Server ETL 工具推介（上）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_47513.htm

第三方 SQL Server ETL 工具推介（下）

Pentaho 数据集成工具

现在许多组织都转向使用开源技术来满足他们的 ETL 需求。这类产品中最流行的一款是 Community 版的 Pentaho 数据集成工具，也就是众所周知的 Kettle。这个产品采用元数据驱动的方法来创建复杂的 Job，并在拖拽式的 GUI 环境中转换。结果，你不必生成客户代码，只需要访问 ERP 连接器，数据质量插件，有 150 多个封装好的映射对象支持高级数据仓库组件，如缓慢变化维和废弃维。

Kettle 也提供一个可扩展的基于标准架构的统一的 ETL、模型和可视化开发环境。注意，Community 版本是一个自支持的产品。如果想要得到技术支持、管理升级和企业特性，必须要升级到企业版才可。

Talend Open Studio

另外一个流行的开源 ETL 产品是 Talend Open Studio。象 Kettle 一样，Open Studio 也是一个元数据驱动的 Solution，支持数据迁移、集成和同步。Open Studio 采用自顶向下的方法来进行业务建模，让业务线的干系人参与集成过程设计并监控这些过程的开发。

开发环境提供了集成过程的图形和功能视图，包括处理各种类型的任务和操作的开源组件和连接器的图形化的工具箱。用户也可以跟踪整个转换流程的数据，自动地生成技术参考文档。此外，Open Studio 也对打包应用提供全面的连通性支持，如 ERP 和 CRM 等，还有数据仓库和在线事务处理(OLAP)应用。

选择第三方 ETL 工具

如你所看到的，除了我这里提到的以外，还有很多第三方 ETL 工具可选。如果你已经在组织中实施了 SQL Server，花些工夫配置好 SSIS 并将它用起来是值得的。然后，如果你的数据管理需求超过了 SSIS 所能提供的 ETL 功能，可能就需要考虑从众多的产品中选择一个适合的了。

在选择工具之前，你需要评估要从中抽取数据的业务系统的类型，分析数据的特征。也需要决定除了在系统实施所需要的 ETL 需求以外你还需要什么功能。例如，你可能需要合并 ERP 和 CRM 的能力。要决定你的组织是否愿意使用开源软件。有些干系人并不满意采用开源技术。最后，你还要负责评估采用最好的 ETL 工具对组织的必要性。

（作者：Robert Sheldon 译者：沈宏 来源：TT 中国）

第三方 SQL Server ETL 工具推介（下）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_47517.htm

SQL Server 数据访问策略：即席 SQL

在谈论数据库架构的书籍，论坛，电子杂志甚至是博客上，往往认为设计并实现数据库，规划了高效的安全和索引策略就是数据库架构的全部，很少包括数据库访问策略方面的讨论。本文将讨论基于 SQL Server 作为后台数据库，数据访问方面的策略选择。

基于微软 SQL Server 数据库应用程序开发，开发人员往往有三种选择，即席 SQL(Dynamic SQL)，存储过程(Stored Procedure)和 CLR(Common Language Runtime)。

即席 SQL 有时也被称为“动态 SQL”，通常指在客户端组织 SELECT,INSERT,UPDATE,DELETE 以及其他任何 SQL 语句，然后，这些 SQL 语句被发给 SQL Server engine,由 SQL Server 来检查语法，编译，生成查询计划，执行。 即席 SQL 可以分为两类，直接嵌套在 C#，VB 或者任何编程语言中，另外一种就是使用像 LINQ 这样的中间件生成的，这里不讨论这两种方法在生成动态 SQL 的方面的好处与坏处。

面试数据库开发人员的时候，当我问，谈谈存储过程的优点，面试者可以非常自信地说出很多优点;反之，如果问，谈谈存储过程的缺点，面试者还能口若悬河的就很少了;同样的，如果问动态 SQL 的坏处，面试者可以说出很多，反之，动态 SQL 的好处，能说一二的就更少了。

即席 SQL

相对于编译好的存储过程，使用未经编译的即席 SQL，确实有一些优点。即席 SQL 可以对查询的运行时控制，查询是在运行时创建的，这样可以根据当时场景，创建精确查询;如 SELECT 查询，你可以只查询你当前场景需要的数据，如当前需要根据产品 ID 或者产品的名字，我们只需要构建

```
"SELECT Name FROM Production.Product WHERE ProductID = " +
iProductId.ToString()
```

即可，我们不需要去执行一个通用的存储过程，之后获得其中的 Name 列的值，如下面的存储过程是根据产品 ID 或者产品的属性，

```
CREATE PROC [dbo].[uspGetProcutAttributeListByID]
```

```
@ProductId INT
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Name,
           ProductNumber,
           MakeFlag,
           FinishedGoodsFlag,
           Color,
           SafetyStockLevel,
           ReorderPoint,
           StandardCost,
           ListPrice,
           ProductLine,
           Class,
           Style,
           SellStartDate,
           SellEndDate
    FROM  Production.Product
    WHERE ProductID = @ProductId
```

END

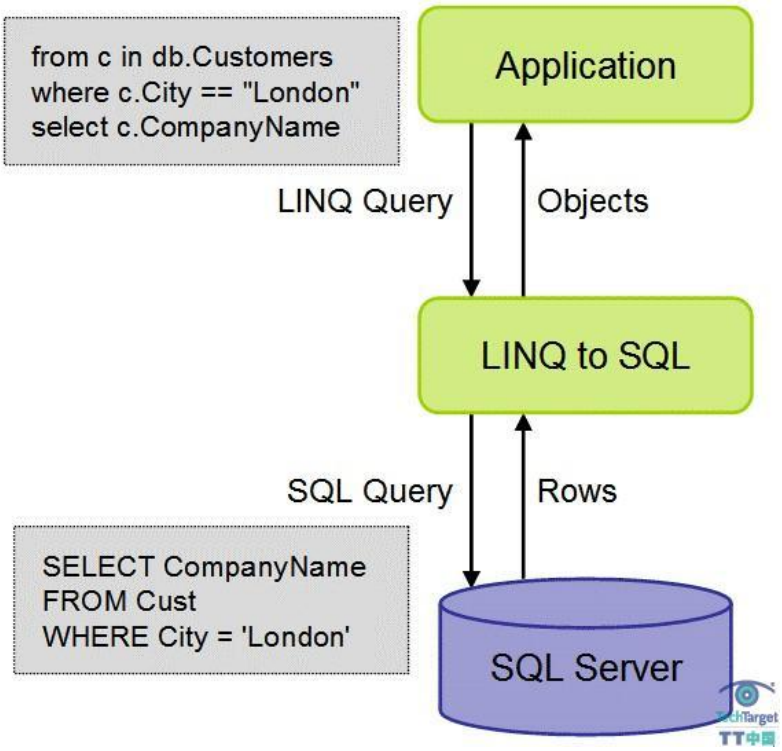
之后在编程语言中指定存储过程的名字以及所需参数，来执行下面类似语句来获得当前产品 ID 对应的产品名字；

```
EXEC [dbo].[uspGetProcutAttributeListByID] @ProductId = 1
```

当然同样的道理，对于 UPDATE 操作，通过构建即席 SQL 我们只修改需要更新数据，因为在客户端我们可以在数据被编辑的地方更容易知道数据是否被改变。

另外，灵活性也是即席 SQL 的一大优势，我们编程的时候常常会遇到这样的情况，根据不同的变量值，执行不同的 SQL 操作;即席 SQL 可以充分发挥编程语言的优势，灵活改变;而存储过程里面的 SQL 编程是面向过程编程，而且 SQL 的 IF - ELSE 操作会使得性能严重下降。

下图是 LINQ 访问 SQL Server 的一个简单架构图，从中我可以看出 LINQ 其实是自动产生动态 SQL 的一个中间件，可能这个叫法不太合适，不过他是根据 C# 语法最终产生 SQL，进而发送到 SQL Server。



即席 SQL 的劣势就简要说一下：即席 SQL 与系统的其他部分相互协作性不强，当前即席 SQL 只能适用于当前场景，重用性不强；即席 SQL 耦合度太高，一旦数据库中表定义发生改变，只要涉及到该表的即席 SQL 都可以面临改动;由 SQL 注入引发的安全问题更是大家立马就能想到的。

（作者：李爱华 来源：TT 中国）

SQL Server 数据访问策略：即席 SQL

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_48806.htm

SQL Server 数据访问策略：存储过程

存储过程是已编译的 SQL 代码，它可以被参数化以便重用。存储过程的封装性，安全性，性能等都是我们选择存储过程作为数据访问策略的理由，当然存储过程也不是完美无缺。

存储过程的封装性可以很好的隐蔽调用者不需要知道的执行细节，而且你可以根据需求变更重新编写存储过程，而不用担心会破坏任何客户端代码，换句话说，这意味着任何特性都可以改变，包括表结构，列名(当然返回的列名最好以别名，alias,形式出现)和编程方法，只要存储过程的名字以及参数不变，客户端不需要任何修改。我想这是我采用存储过程最重要的原因。

安全性，你可以针对 SQL Server 用户，对存储过程进行访问权限的设置，而不是授予用户对存储过程用到的资源进行访问权限设置;在实际工作中，你遇到过这样的情况吗?某些程序员或者用户修改不应该被修改的数据，删除了不该被删除的数据，之后跑到你这里来，我改怎么恢复过去?

我们通过下面的 SQL 语句针对指定存储过程，将其执行权限授予指定用户， 这样的我们可以在具体表，列，行的基础上，更好管理安全性。

```
GRANT EXECUTE ON [dbo].[uspGetProcutAttributeListByID] TO [MAX]
```

存储过程的参数可以使查询优化计划得到最大程度的利用，避免了检查语法，编译，生成机器代码，生成查询计划等，性能因此得到很大提升，同时客户端调用的只是存储过程的名字，在网络传输方面也要远比发送整段 SQL 代码要强。

相对于动态 SQL 的高耦合，对存储过程的修改不会或者很少对客户端产生影响。

存储过程有缺点吗？有！但绝对不是可一致性差，技术社区或者个人博客上在谈论存储过程缺点的时候，大部分人都在说存储过程可移植性差，这是因为他们将软件或者项目的业务逻辑封装在存储过程中了;针对存储过程的使用，我们应该扬长避短，SQL 使用来处理数据的，即 SELECT, INSERT,DELETE,UPDATE 操作，SQL 在对逻辑运算方面远逊于编程语言，因为我们应该将业务逻辑放在中间层而不是 DB 层。

存储过程的缺点也是 TSQL 的缺点，TSQL 是面向过程语言，是一种查询语言，因此它在程序分支或者条件控制方面(IF - ELSE,CASE)做的不好，而且效率很差。当存储过程中出现 IF ELSE 的时候，个人建议可以考虑将其分为两个存储过程了。如果你不想在客户端调用的时候出现根据条件对调用哪个存储过程进行判断，你也可以创建另外一个存储过程在里面进行条件判断，确定哪个存储过程需要被执行(业务逻辑不知不觉又跑到存储过程里面了)。

另外存储过程在执行 UPDATE 操作的时候，不知道哪些列真正的被修改了，而是根据客户端闯过来的所有参数，将对应列都进行 UPDATE，不够灵活。

(作者：李爱华 来源：TT 中国)

SQL Server 数据访问策略：存储过程

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_48808.htm

SQL Server 数据访问策略：CLR

相信每个数据库开发人员都知道，存储过程是用来对数据进行操作的，除此之外，一切东西都不应该放到存储过程中去执行，但是，一些非数据处理操作就是需要在存储过程中进行。如对 XML 的操作，虽然 SQL Server 2005 以后已经支持对 XML 的操作，但是当 XML 比较大的时候，TSQL 的处理效率就会很低；这个时候我们可以借助 CLR(Common Language Runtime)，CLR 在很大程度上解放了 TSQL 逻辑运算能力不足的问题，而且 CLR 拥有丰富的语言支持，C#，VB.NET 等；在.Net Framework 基础上，拥有复杂的过程逻辑和计算，这使其在对字符串的操作，统计运算等方面表现优秀；另外 Visual Studio 还为 SQL Server 对象开发提供了很多模板。CLR 可以创建函数，存储过程，触发器甚至用户自定义类型，说实话，本人只创建过 CLR 函数，就是这些 CLR 函数帮助我们在 SQL 中很好地完成了对 XML 的处理。

CLR 是运行在 SQL Server 上的 DLL，因此安全性成为了必须考虑的因素。编程人员有个很好的习惯，需要编写功能或者 DLL 的时候，通常先 Google 一下，如果能找到现成的 DLL 或者代码，直接拿过来，如果找到第三方 CLR，没有研究其背后的代码， 直接部署到 SQL Server 上，其潜在的危险性可想而知。

(作者：李爱华 来源：TT 中国)

SQL Server 数据访问策略：CLR

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_48809.htm

玩转自助式 BI：SQL Server Crescent（上）

在去年十一月举行的 SQL Server 专家会议(PASS)上，微软公司推出了 Crescent 项目，它是一款新的商业智能(BI)工具，专为使非技术用户能更容易地创建视觉效果良好的报表而设计。它是下一代 SQL Server BI 核心。

几个月后，我们仍然没有得到更多其它详细信息——Denali 的第一社区技术预览(CTP)没有出现 Crescent，CTP2 只是针对微软 MVP 的(MVP 必须签署相关保密协议)，而 CTP3 仍然在建设中。幸运的是，上月底在亚特兰大举行的 TechEd 北美大会上，微软公司在几场培训会上展示了其 Crescent 产品。从我见到的情况来看，我认为大家的等待是值得的。

首先，我们来讨论一下 Crescent 背后的设计目标。正如其口号所说的“面向大众的 BI”，微软希望提供这样一款工具，它能易于使用并且能为数据视图提供强大的实用的视觉效果。在设计界面时，该公司设立了非常雄心勃勃的目标：所有的功能都只要通过一次或两次点击就能完成。

一旦你用上了这款工具，你就会发现微软公司确实做到了。你打开一个模型，Crescent 会读取该模型的元数据，分析其中的关系并给你展现一些查看数据的选项，比如查看表和字段域。例如，Crescent 可以检测父子关系，你可以有几种途径钻取到子记录。

让我们来看看技术细节。考虑到 Crescent 是 PowerPivot 的下一代产品，或者正像微软公司给它的称呼，它是 PowerPivot 和 Excel 之间的连接点。因为它设计的就是要在 SharePoint 中运行的，但 SharePoint 企业版价格昂贵，而且安装也不简单，许多人把它看做是一种潜在的障碍。但是既然微软公司使用 SharePoint 来把各种 BI 服务连接到一起，你迟早不得不接受这一点——SharePoint 就在那儿。

有了 Crescent，你就有了一个直观的报表设计器，可以构建和预览报表。一旦你部署了某个报表，用户可以在浏览器中查看它，也可以运行在应用框架 Silverlight 中，Silverlight 是微软公司针对 Flash 的替代品。

至于数据源，我们有几种选择。你可以在已经上传到 SharePoint 的 PowerPivot 模型中访问数据，或者也可以使用分析服务中的 BI 语义模型(BISM)。(BISM 是一种可以使在 Crescent 中构建 BI 模型更容易的技术。)这两种模型提供了对数据最快速的访问。你还可以直接查询 SQL Server 表。但是如果你想这样做的话，微软公司推荐你使用列存储索引，它是微软公司采用 VertiPaq 技术实现的超级搜索功能，它可以保证查询的快速响应时间。

（作者：Roman Rehak 译者：冯昀晖 来源：TT 中国）

玩转自助式 BI：SQL Server Crescent（上）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50280.htm

玩转自助式 BI：SQL Server Crescent（下）

微软公司把 Crescent 称为“交互式数据研究和可视化展现体验”。在看了 Redmond 程序经理们在 TechEd 技术大会上的演示之后，我非常赞同这种提法。构建报表和分析数据看起来如此的有趣。除了所有常见的 BI 工具，你还能获得一个数据模型，会展示给你度量和可用域。你可以快速修改图表类型，也就是你查看数据的方式。你可以使用与在 Excel 中相同的图表——柱状图，条形图，稀疏图以及其它。要加上过滤功能也很容易：传统过滤功能中你只能“全选”或者选择单独的条件，但是现在你可以在你的过滤器中输入过滤条件，省得在几百个项目中滚动查找。

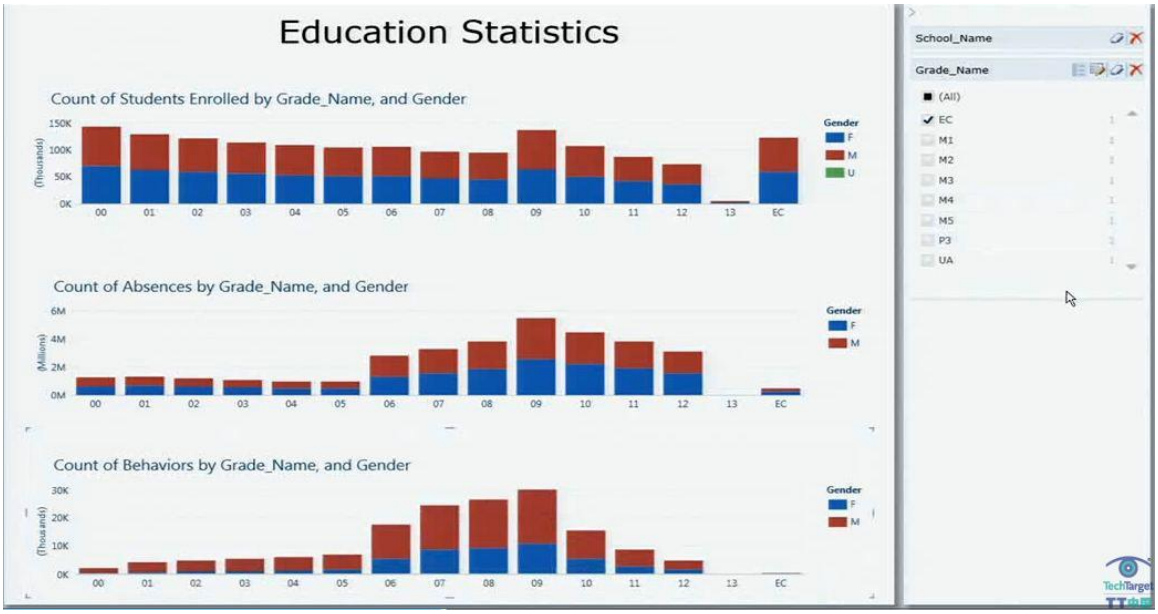


图 1。该图展示的是 Crescent 的报表设计器。该报表展示了三栏图标，每一栏展示了一种不同的度量值，还有过滤器。该图也按性别做了区分。（点击放大）

Crescent 还有几个令人印象深刻的视觉功能。你可以设置父记录按区块显示图像。例如，你可以滚动你的类别显示为区块。当你将鼠标悬浮到它们之上时，Crescent 会在明细区域显示分类明细，如图 2 所示。

图 2。该图展示了 Crescent 把分类按区块显示的效果。如果某个分类有图像，它会显示为图表形式。当你选择目录时，Cerscent 会自动显示分类细节。

下面将介绍 Crescent 的另一个优秀功能：当你浏览图表时，你可以点击图例条中的一个，Crescent 会自动按选中项进行过滤并高亮显示明细。不过，最吸引人的特性可能要数图表的动画展示了。把图表类型修改为散点图，然后设置值播放轴，可以设置一个时间度量，比如一年或者一个季度。在 Crescent 播放动画时，这些散列点会在图上移动，这样你就可以直观地看到所有季度或者年的取值段趋势。

虽然 Crescent 令人印象深刻，但它的定位并不是替代 Visual Studio 的报表构建器或者报表设计器，而是对它们的补充。Crescent 可以让你不费吹灰之力创建交互式的可视化报表;所有的工作都交给 Crescent 做了。但是轻松易用的代价就是你不能对最终报表的布局有完全绝对的控制了，不可能像 Visual Studio 和设计器那样的灵活。因此，你可能不得不等待将来的版本提供更多查看数据的方式或者对整体布局的更多控制了。如果你需要在你的报表中做复杂运算的话，你还是最好使用成熟的，经验证好用的工具吧。

总体而言，Crescent 的预展示证明了其承诺，微软公司似乎实现了它的目标：以互动的方式提供简单快速的方法显示数据，同时帮助实现了“面向大众的 BI”。

（作者：Roman Rehak 译者：冯昀晖 来源：TT 中国）

玩转自助式 BI：SQL Server Crescent（下）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50283.htm

提高 SQL Server 扩展性：最大化数据库性能

扩展一个跨多个系统的 SQL Server 环境可以说是一项困难且复杂的系统工程，涉及到分区数据库、联邦等等。所以，当涉及到 SQL Server 可扩展性的时候，大多数组织更喜欢在设法解决之前采取扩展独立系统的方法。这里有四种使扩展过程更轻松、更有效的技巧。

1、最大化 SQL Server 性能

每个服务器的性能可以归结为四个基本部件：

- 内存
- 磁盘存储
- 网络适配器
- 处理器

第一步，尽量满配现有服务器上的这四个部件。

从内存开始，它可以产生最大的影响，通常是服务器里最容易扩展的部分。只有一个限制：运行在 Windows 32 位版本上的服务器，没有理由安装超过 4GB 的 RAM，因为操作系统不能使用 4GB 以上的内存。在 64 位的计算机上运行 64 位版本的 Windows 和 SQL Server，需要尽可能多的安装内存，以使 SQL Server 获得最佳性能。

事实上，如果你拥有运行在 Windows 32 位版本上的 SQL Server 实例，将其迁移到 64 位计算机是你的第一个步骤，因为这可以让你的系统访问更多的内存空间，是你获得最大性能提升的一个方法。不要吝啬投资内存;根据服务器制造商的建议来采购内存，虽然常常是更昂贵、拥有纠错能力、高速的内存模块。但这值得付出。

接下来要看的存储。这是一个大课题，后面将介绍更详细的技巧。我只想说，存储性能越高始终是一件好事。

网络连接是第三个方面。许多 SQL Server 计算机的性能在网络适配器层面受到约束。如果负担得起，那么就配备多个网络适配器以提供多条网络路径。千兆以太网(GbE)适配器应该是用于 SQL Server 计算机的最低配置，如果网络支持 10 GbE，那么就使用它。特

别重要的是，你的计算机使用一个单独的网络，并至少为每个主要用途配备一块网络适配器。

例如，如果使用 iSCSI 作为存储通信协议，数据传输应该使用专有网络接口控制器(NIC)和专有网络，而不是与客户端流量共享网络。

最后，检查服务器的处理器。它排在清单最后面是有原因的：从效益成本来考虑，升级处理器是很罕见的。这是因为处理器必须与主板相匹配，而主板通常是为特定型号的CPU 专门设计的。要安装更快的处理器，通常不得不更换一块新的主板以及相匹配的新内存，其它一切都要换成新的。换句话说，就是一台全新的服务器。

如果服务器可以添加更多的处理器，那么就进行升级，更多的处理器总能更快些。但在这里，你的选择可能是有限的：大多数服务器在购买时都是满配的，不会留下任何额外的处理器空间。

(作者：Don Jones 译者：沈宏 来源：TT 中国)

提高 SQL Server 扩展性：最大化数据库性能

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50940.htm

提高 SQL Server 扩展性：合理虚拟化

信不信由你，虚拟化是一种聪明的给 SQL Server 带来性能优势的方式。这似乎违反直觉;毕竟，虚拟化的重点是在单台主机上运行多个工作负载。因此，把整台计算机完全分配给 SQL Server，不是可以得到更好的 SQL Server 性能吗?

一般来说，你会这样做。如果可以将整台物理服务器完全分配给一个 SQL Server 数据库，那么就这样做。然而，许多组织习惯在他们的 SQL Server 系统中安装多个数据库，甚至多个实例。这没有什么不妥当的;事实上，SQL Server 就是这样设计的，但它不能提供与工作负载相适应的可用硬件资源的灵活性。

相反，将这些数据库迁移到虚拟机后，占用的资源总量可能会少于整台物理服务器的资源。SQL Server 是一个最棒的虚拟机客户应用程序，可以将数据库切分成多个 SQL Server 实例，每个实例占用一个虚拟机，这样就可以获得较大的灵活性。使用实时迁移技术，可以迅速地将虚拟机从一台主机迁移到另一台主机，根据当前的工作负载，重新分配虚拟机以更好地利用现有硬件资源。

你可以使用类似集群技术的方法，包括 Windows 集群服务。将数据库分区到集群中不同的 SQL Server 实例上(归根到底，SQL Server 实例是虚拟化的一种形式)。然后，可以随意移动这些实例，而无需担心数据库的可用性。如果数据库 A 在某个下午需要扩大规模，则可以关闭该数据库群集节点上的其他实例，腾出资源供那个指定的实例使用。这种动态缩放的能力可以得到令人赞叹的性能，但也确实需要你的组织开发一个成熟的性能监控和响应模型。

如果你准备花费大量资金购买新的服务器硬件，结合一些资源共享规划(如集群或虚拟化)是非常有意义的。这样，可以让昂贵的新硬件在任何时候都尽可能地发挥其最大能力，满足用户的性能需求。

(作者：Don Jones 译者：沈宏　来源：TT 中国)

提高 SQL Server 扩展性：合理虚拟化

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50943.htm

提高 SQL Server 扩展性：提升存储性能

存储系统的容量是由数据库的容量需求所决定的;而存储系统的速度往往是被忽视的指标。在 SQL Server 世界里，磁盘存储速度为王。SQL Serve 对 I/O 能力的需求排在内存、网络适配器或处理器之前，成为最重要的一个因素。让存储尽可能的快。这意味着把存储区域网络(SAN)连接到 hyperfast 光纤连接的另一端，并使用诸如 iSCSI 之类的快速 SAN 协议与磁盘进行通信。

密切关注数据库实际处理的工作负载，让存储技术(如 RAID)与工作负载相匹配。例如，RAID 5 提供在一个设备发生故障时的可恢复性，但它会稍微增加写操作时间，因为额外的用于数据恢复的信息，必须在每次更新时写入。

快速的磁盘控制器可以帮助解决这个问题，通过缓存服务器发送的数据，然后迅速将数据写入磁盘。存储子系统的每个元素在性能方面都扮演着关键角色，如单碟旋转速度、裸设备 I / O、平均寻道时间、通信介质(铜缆或光纤)。与经验丰富的存储供应商一起努力，能够为一个系统提供最佳的 SQL Server 性能。

如果资金是一个问题，那么在削减存储之前，请先削减处理器，甚至内存。存储性能是扩展 SQL Server 的一条长期途径。

(作者：Don Jones 译者：沈宏 来源：TT 中国)

提高 SQL Server 扩展性：提升存储性能

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50945.htm

提高 SQL Server 扩展性：购买新的服务器

在某些时候，需要检查一下现有的服务器，以确认再也不能从中挤出更多的性能。存储 I/O 像预期的那样快。内存已经被耗尽。每一个处理器插槽都已插满。扩展槽中都是 10 GbE 网卡，有足够的冷却风扇提供动力。在这种情况下，很容易说服自己购买新的服务器（尽管你的财务总监还会与你就此进行争论）。

在发现一台服务器仍然有扩展的空间，却决定买一台新的来代替它，这个决策会相当艰难。当你看到有空闲的内存插槽、CPU 插座和 PCI 背板，似乎把这些空闲插槽用新的、性能更好的部件填满更符合成本效益。

有时候这是一个好主意。但有时候也需要抵制这种冲动。

首先，不要花钱升级一台 32 位计算机。还是换成 64 位系统，安装 64 位版本的 Windows 和 SQL Server。购买一台满配内存、拥有四个或更多的多核处理器以及高速网卡的新服务器。

最后，考虑 SQL Server 可扩展性：任何使用年限超过四、五年的服务器都应该被换掉，无需升级。给旧服务器增加内存或处理器的成本通常与购买一台全新的服务器的成本相当，一台新服务器会带来更多的性能提升：更快的 BIOS 电路，速度更快的芯片组，更快的内存桥梁。旧服务器仍然可以作为文件服务器来使用，也可以运行 SQL Server 处理其它的、工作负载较低的业务。

（作者：Don Jones 译者：沈宏 来源：TT 中国）

提高 SQL Server 扩展性：购买新的服务器

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_50950.htm

把 SQL Server 数据导入到 Excel PowerPivot（上）

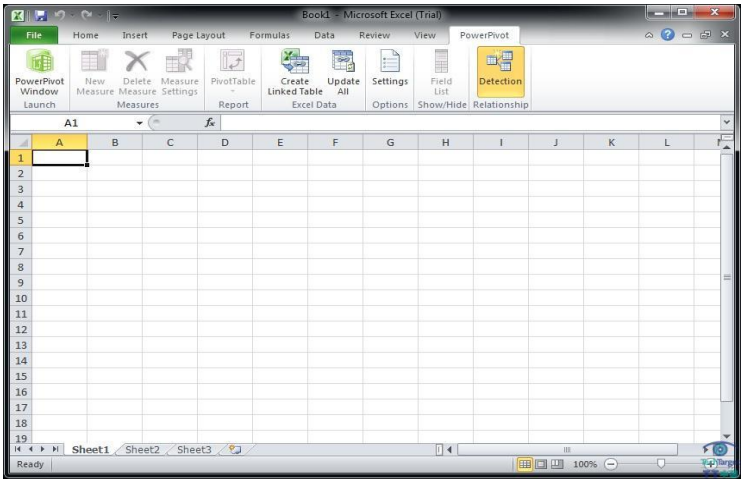
微软公司的 PowerPivot for Excel 2010 是一款数据分析工具，可以让你很容易地从 SQL Server 数据库和其他数据源导入数据到 Excel 环境中。由于 PowerPivot 高效的压缩算法和内存分析功能，你可以导入和管理大数量的数据。

此外，PowerPivot 采用了 VertiPaq 引擎，它是一种基于列的数据存储技术，来自于 SQL Server 分析服务，它可以把多核处理器功能和内存的强大能力推向你的指尖。因此，你可以用它在处理几千行数据的时间内处理数百万行数据。而且因为是在 Excel 中使用，所以你可以利用熟悉的、用户友好的 Excel 工具集。

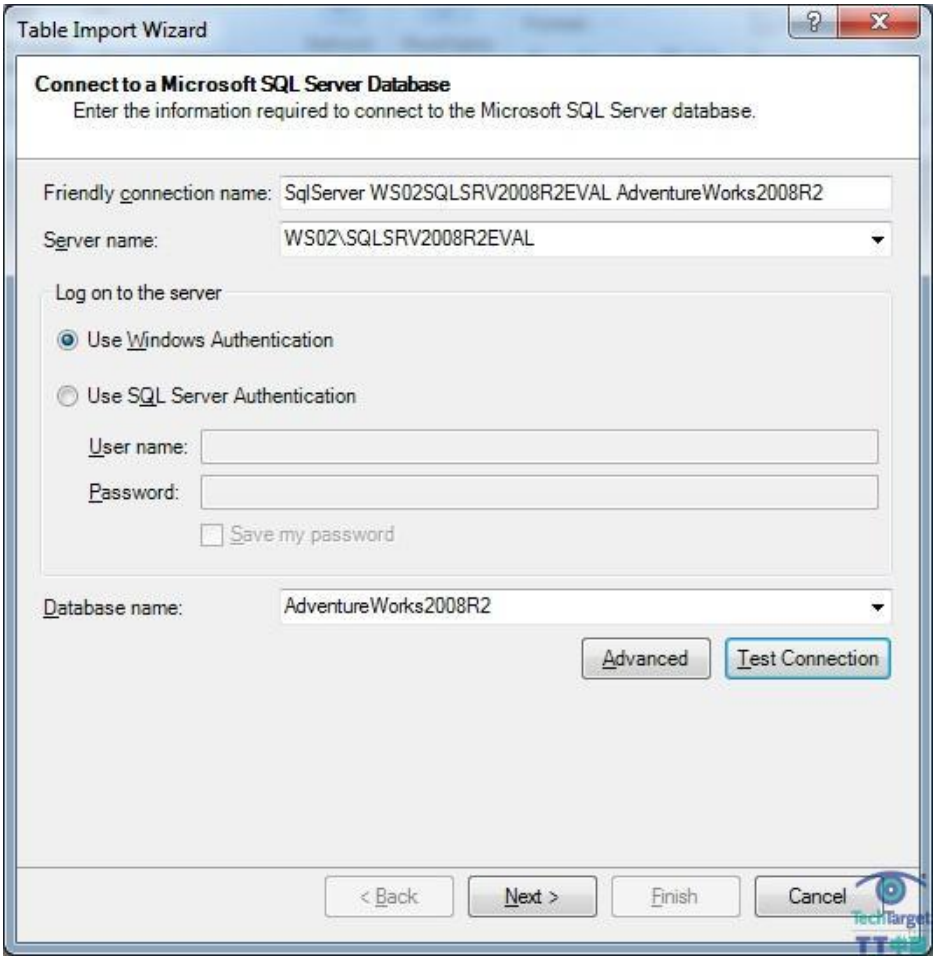
对于希望处理从 SQL Server 数据库导入到 Excel 中分析计算的复杂关系型数据的用户来说，这是个好消息。(该过程可以被集成到 SharePoint 工作流中进行协同工作)要使用 PowerPivot，你必须运行微软公司的 Office 2010，并安装 PowerPivot 插件。

在本文中，我在本地安装了一套 SQL Server 2008 R2 试用版实例，同时安装了“AdventureWorks2008R2”示例数据库，它是微软公司为测试和评估目的设计的示例数据库。我还在 pre-R2 SQL Server 2008 软件本地实例上针对“AdventureWorks2008R2”测试了 PowerPivot。

在你安装微软公司提供的 PowerPivot 插件后，会有一个 PowerPivot 功能区(也就是一个新的标签页)被添加到 Excel 工作簿中，如图 1 所示。

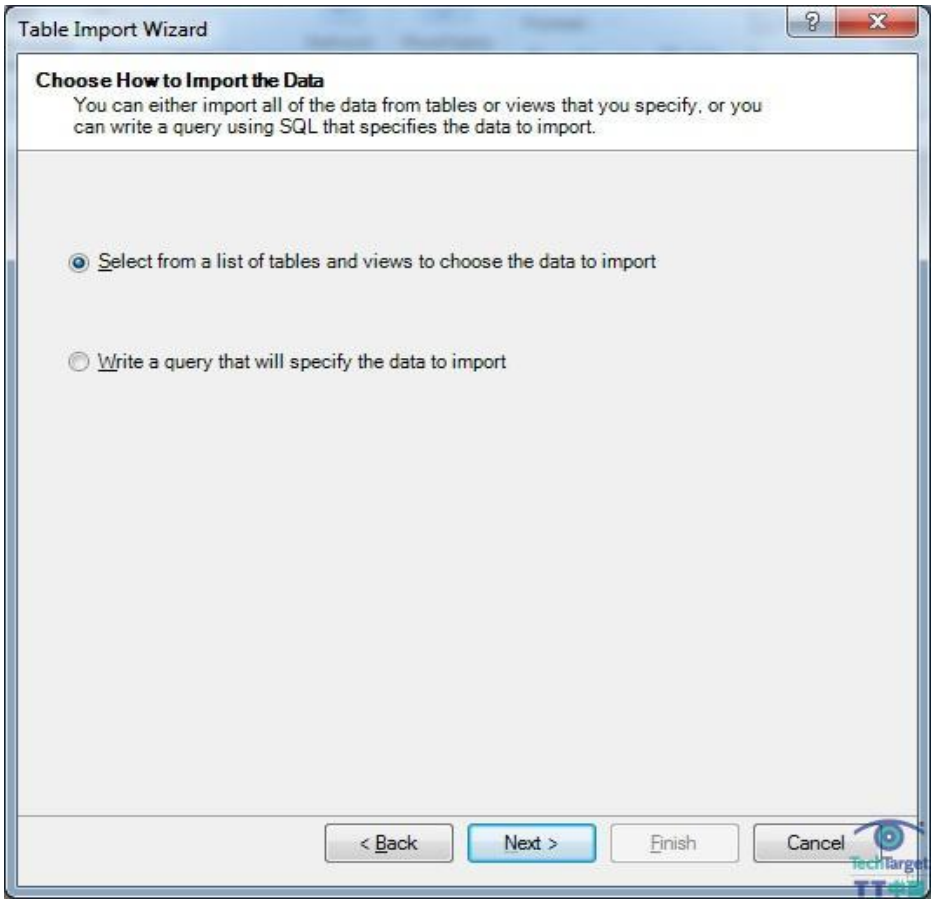


你可以通过该功能区访问 PowerPivot 窗口(见图 2)，在那里你可以导入并管理来自 SQL Server 数据库和其它数据源的数据。



从 SQL Server 数据库中提取数据

通过表格导入向导可以连接到数据库并导入 SQL Server 数据。要运行该向导，请打开 PowerPivot 窗口，在菜单上点击“从数据库提取”按钮，然后点击“从 SQL Serve 数据库提取”。表格导入向导的第一个界面是“连接到微软 SQL Server 数据库”，如图 3 所示。



当你连接到数据库时，请从下拉列表中选择 SQL Server 实例，选择认证类型并选择数据库。在填写完必要的参数之后，可以测试连接。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

把 SQL Server 数据导入到 Excel PowerPivot（上）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_51342.htm

把 SQL Server 数据导入到 Excel PowerPivot（下）

选择导入 SQL Server 数据的表

要导入 SQL Server 数据，你有两中选择：选择你想导入的表或视图;或者编写临时查询。图 4 展示了表导入向导的第二个界面。(默认情况下，该向导会让你从表和视图中进行选择)现在，我将选择第一种方案;后面我会给大家演示如何自定义查询。

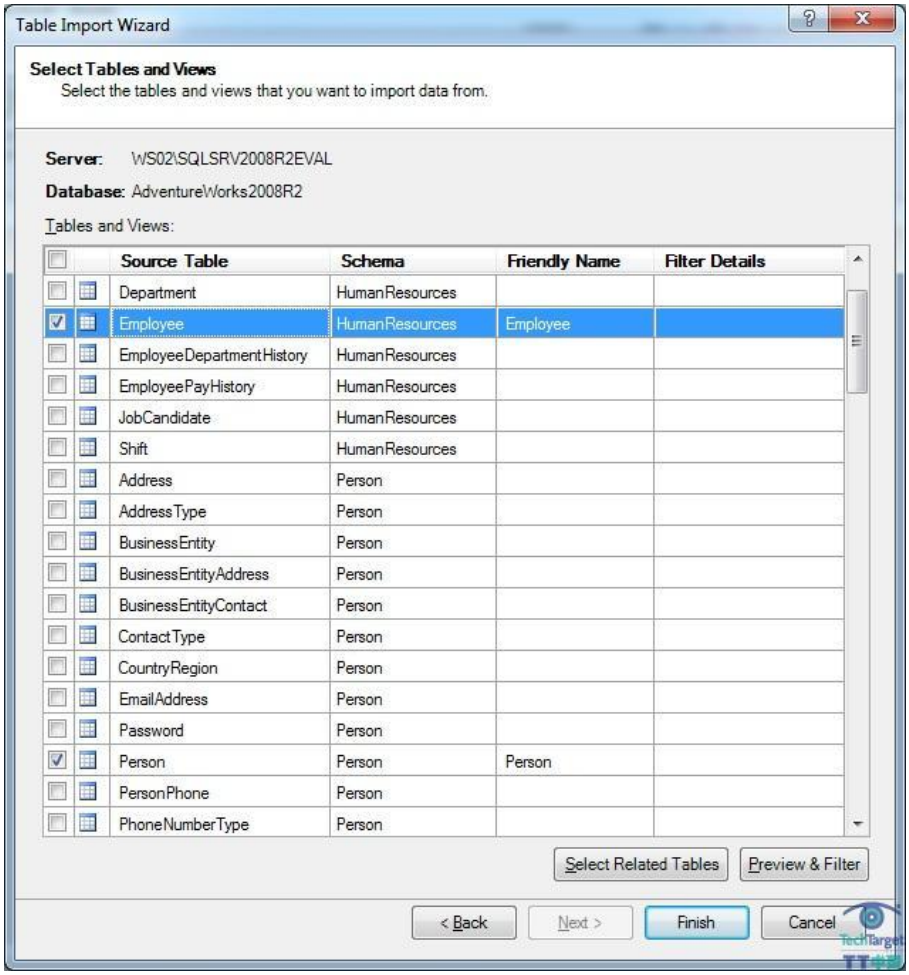
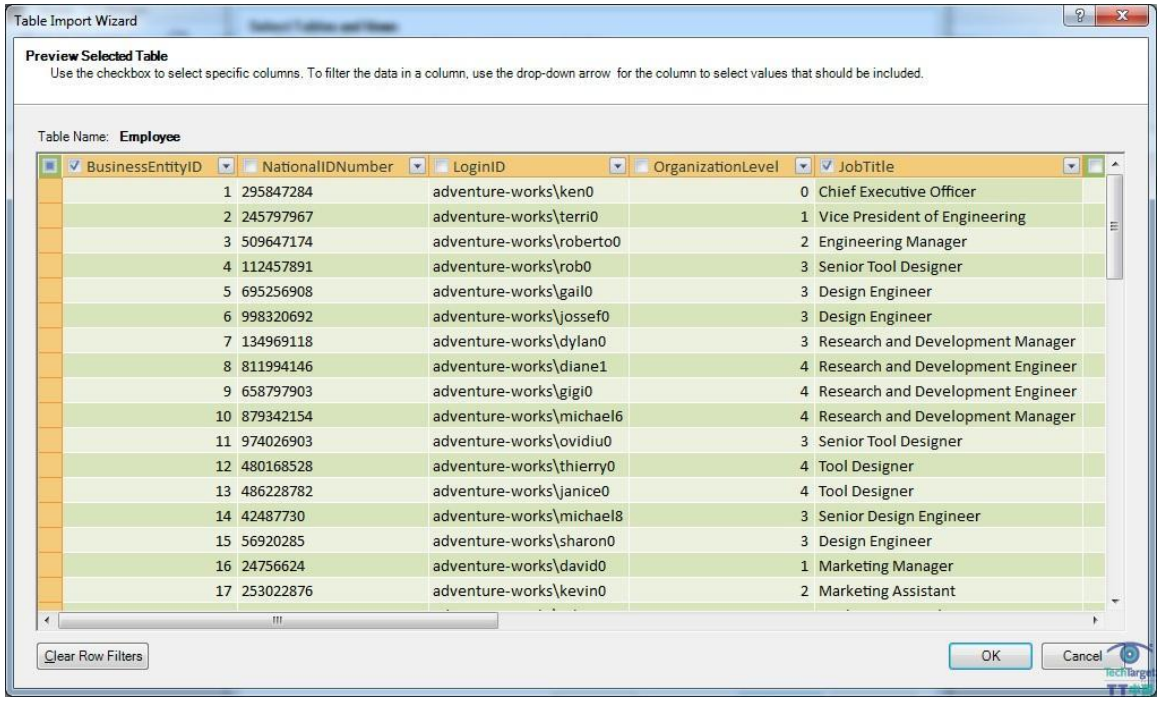


图 4 点击放大

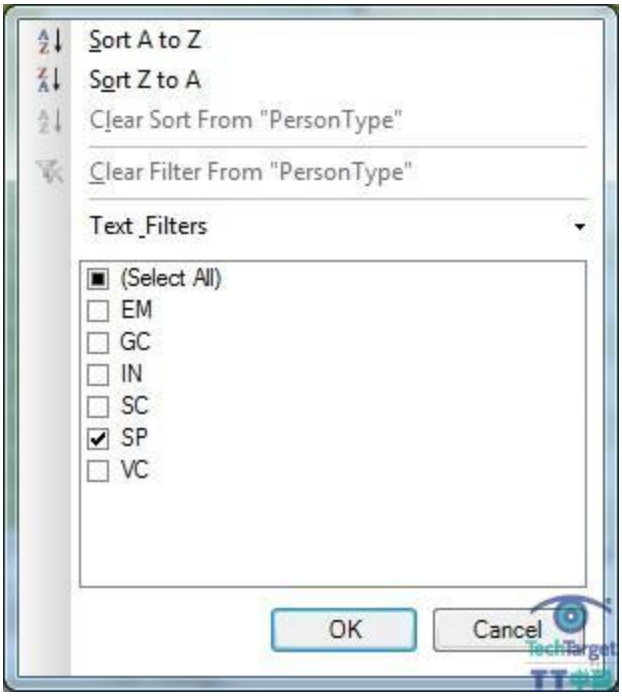
表导入向导如图 5 所示，它包括“AdventureWorks2008R2”数据库的表和视图列表。要从这些表或者视图中选择导入数据，请选中对象名称前面的复选框。在这里，我将选择“HumanResources.Employee”表和“Person.Person”表。



BusinessEntityID	NationalIDNumber	LoginID	OrganizationLevel	JobTitle
1	295847284	adventure-works\ken0	0	Chief Executive Officer
2	245797967	adventure-works\terri0	1	Vice President of Engineering
3	509647174	adventure-works\roberto0	2	Engineering Manager
4	112457891	adventure-works\rob0	3	Senior Tool Designer
5	695256908	adventure-works\gail0	3	Design Engineer
6	998320692	adventure-works\jossef0	3	Design Engineer
7	134969118	adventure-works\dylan0	3	Research and Development Manager
8	811994146	adventure-works\diane1	4	Research and Development Engineer
9	658797903	adventure-works\gigi0	4	Research and Development Engineer
10	879342154	adventure-works\michael6	4	Research and Development Manager
11	974026903	adventure-works\ovidiu0	3	Senior Tool Designer
12	480168528	adventure-works\thierry0	4	Tool Designer
13	486228782	adventure-works\janice0	4	Tool Designer
14	42487730	adventure-works\michael8	3	Senior Design Engineer
15	56920285	adventure-works\sharon0	3	Design Engineer
16	24756624	adventure-works\david0	1	Marketing Manager
17	253022876	adventure-works\kevin0	2	Marketing Assistant

图 5 点击放大

在这个地方，我就可以点击“完成”按钮来结束向导并从这两个表中导入完整的数据集到 PowerPivot。然而，我想过滤过滤一下数据，这样我就不必导入很多不必要的数据。要做到这一点，我们要高亮选中表或者视图，然后点击“预览和过滤器” 按钮来创建一个过滤器。这样就会打开“预览选中表”界面，如图 6 所示。



正如截图所示，我打开了“预览选中表”界面，打开的是“Employee”表。我现在可以浏览列头信息，把不想要的列反选掉。在本文中，我只选了“BusinessEntityID”列和“JobTitle”列。点击“确定”按钮返回表和视图列表界面。

接下来，我在“Person”表上创建了一个过滤器。我只选择了“BusinessEntityID”列，“FirstName”列和“LastName”列。然而，对于这个表，我还想限制行，只查询那些“PersonType”列值是“SP”的行(只查询销售人员)。要实现这样的效果，我需要在“PersonType”列的顶部点击下拉箭头。然后会打开一个小的过滤器窗口，如图 7 所示。然后清除掉除“SP”值意外的复选框并点击“确定”按钮。

点击“确定”按钮关闭“预览选中表”界面，然后在“选择表和视图”界面点击“完成”按钮。接下来就到了该向导的最后一个界面了，它报告了导入的状态。如果导入成功了，你将在每个表名旁边看到一个绿色的标记(如图 8 所示)。点击“关闭”按钮完成数据导入流程。

导入数据通常是一个简单的流程。我唯一一次遇到一个问题是源列超过了 PowerPivot 允许的长度。例如，当我试图导入整个“Person”表时，我接收到错误信息，提示“AdditionalContactInfo”列超出了 Excel 允许的最大允许大小。然而，在我创建了过滤器，过滤出我想要的列和其它信息之后，我就不会遇到导入数据的问题了。

提取 SQL Server 数据库数据：查询方法

要使用查询语句导入数据，请再次运行表导入向导，在“选择如何导入数据”界面选择“使用查询语句导入”，然后点击“下一步”按钮。下一个界面要求你指定一个 SQL 查询语句，如图 9 所示。

首先，给你的查询起个名字，然后指定一个“Transact-SQL”语句。在本文中，我创建了下面的查询来从“Sales.SalesOrderHeader”表中提取数据：

```
SELECT

SalesPersonID,

YEAR(OrderDate) AS OrderYear,

DATENAME(month, OrderDate) AS OrderMonth,

SubTotal

FROM

Sales.SalesOrderHeader

WHERE

SalesPersonID IS NOT NULL
```

在定义好查询语句之后，你可以进行验证然后点击“完成”按钮。表名“Sales”被添加到了“PowerPivot”窗口中。

在你导入数据之后，你可以采取其它步骤，比如配置你的关系或者创建计算列。你还可以创建图表和表格，包括 PivotChart 和 PivotTable。然而，请一定保存好你的数据。在 PowerPivot 窗口中的所有数据(包括你创建的任何图表和表格)，都只被保存在 Excel 电子簿文件中。保存好以后，你可以回来修改文件，根据你的需要做修改。

这里最出色的地方在于，不管你是否从头开始创建计算，还是修改了现存的内容，你都不必是一位 SQL Server 数据库专家或者商业智能专家。你需要做的所有事情就是利用丰富的 Excel 环境和 PowerPivot 的强大和高效来实现你的目标。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

把 SQL Server 数据导入到 Excel PowerPivot（下）

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_51346.htm

SQL Server 2012 容量管理：列存储索引

由于在大数据集中常常会出现扩展问题，微软公司即将推出的 SQL Server 2012 中大部分针对扩展性和容量管理的新功能都定位专门处理大负载：数据仓库，商业智能(BI)和决策支持应用程序。把这样一些应用程序整合到一起是相当辛苦的，而维护它们的数据结构就更困难了。

列存储索引

列存储索引被标榜为 SQL Server 2012 中最大的可扩展性改进，现在社区技术预览(CTP)中已经可用了。列存储索引为每个列把数据打包到磁盘页的独立集合中。这是相对于传统的每页存储多行的方法而言的。

微软公司宣称使用这种新的索引方法有几个优势。举例来说，存储列这种方式使得计算特殊查询需要的列能轻松快速地提取。这是因为该索引压缩了数据，平均来看列比整个行更冗余，因此可压缩性也更高。磁盘访问模式和缓冲允许数据被更加快速地提取和保留。在给定表中存在的列越多，或者你计划添加到给定表中的列越多，列索引给表带来的扩展效果就越好。

从扩展性的角度来看，优点是很明显的。你可以给数据库扩充容量，而无需担心这会对 BI 查询(查询的数据可能达到数十亿行)性能造成影响。微软公司自己前期工作发现，请求巨量行数据的各种查询订单程数量级激增，这跟你手头的数据类型有关。

需要说明的一点是，一旦你给表增加了列存储索引，它就变成只读的了。具体来说，你不能使用“Insert，Update，Delete”或者“Merge”语句，也不能使用批量操作添加数据。我猜这是避免列存储索引被重复更新的一种手段，进而避免损失摆在首位的性能优势。

如果你需要在没有列存储索引的情况下运行查询(例如，判断净性能增益)，你可以在 T-SQL 查询中使用“IGNORE_NONCLUSTERED_COLUMNSTORE_INDEX”选项，从该选项的名称也可以看出，微软公司已经计划推出集群的列存储索引。

(作者：Serdar Yegulalp 译者：冯昀晖 来源：TT 中国)

SQL Server 2012 容量管理：列存储索引

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_55619.htm

SQL Server 2012 容量管理：更多的表分区

在带有列存储索引的表中，如果无法使用“删掉索引，增加数据，再重新创建索引”的方法增加数据，那么增加数据的另一种方式就是：切换到一个新的表。默认情况下，SQL Server 中的表支持一千个分区。SQL Server 2012 把限制提升到了 1.5 万个默认分区，这就使得在需要处理大规模操作时切换分区更容易了。显然，微软公司希望给用户每天使用一个分区的功能支持，有 1.5 万个分区，你可以用这种方式管理数据达 41 年之久。

虽然增加分区能给你更多的扩容灵活性，但它也不是没有代价的。如果你计划使用大量分区，你的数据库应该有至少 16GB 内存。这不是外在的强迫限制;如果没有检测到一定数量的最小内存限制，SQL Server 将不会明确拒绝在特定计算机上设置超过一千个分区。但是，你设置的分区越多，你运行查询时遇到性能问题或者内存耗尽问题的机会就越大。

如果你想用 SQL Server 2012 CTP 作为这些新功能容量规划的测试床，要牢记的一件事就是：CTP 如何处理统计。微软公司说，如果创建或者重建了分区索引，“(SQL Server 2012 CTP)中的统计”不是通过扫描表中所有行来创建的。相反，查询优化器使用默认样例算法来生成统计。……要想在分区索引的情况下通过扫描表中所有行来获得统计，要使用“CREATE STATISTICS”或者“UPDATE STATISTICS”并加上“FULLSCAN ”从句。微软公司声称这些变更将不会影响查询性能，但是如果你的统计进度影响容量或者性能规划工作，那它们可能会出现问题。

这些功能在 SQL Server 2012 CTP3 中才被激活，在较早发布的 CTP 版本中(去年十一月以前发布的)是不可用的。另外，如果你使用的数据库仍然基于 32 位 x86 架构，要注意超过一千个分区的表或者索引可能在这类系统上是不支持的配置，这里要感谢微软公司对 64 位服务器的推动了。

(作者：Serdar Yegulalp 译者：冯昀晖 来源：TT 中国)

SQL Server 2012 容量管理：更多的表分区

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_55620.htm

SQL Server 2012 容量管理：AlwaysOn

AlwaysOn 是微软公司 SQL Server 2012 中的高可用及灾备恢复技术。它使用了相同的“可用性组”技术变体，该技术在最近的 Exchange 版本中已经使用了。

我在这里提到 AlwaysOn 有两个原因。第一，扩展性和容量规划会被你实现镜像和集群的方式影响。好消息是 Windows 种的集群不再严重依赖于同质硬件，所以保持集群工作并进行横向纵向扩容也容易了一些。这样你就不需用在一个地方存储一切或者全面使用相同的硬件了。

第二，AlwaysOn 可以被用于卸载备份到次级复制，作为缓解你主服务器群 I/O 和处理负载的一种方式。如果你没怎么考虑更换你的备份基础设施，这么做就是很好的一个借口。最终结果是：你可以降低在前端增加容量的需求，更有效地扩展你的后端。

（作者：Serdar Yegulalp 译者：冯昀晖 来源：TT 中国）

SQL Server 2012 容量管理：AlwaysOn

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_55621.htm

SQL Server 2012 容量管理：FILESTREAM 和 FileTable

SQL Server 2012 中可能影响扩展性和容量的另一项特色就是对在 SQL Server 2008 中引入的 FILESTREAM 功能有很大改进。FILESTREAM 支持 SQL Server 直接工作于巨大存储(新技术文件系统 NTFS 可容纳的最大量)，这样二进制大对象就可以在磁盘上保存为文件，但可以用 T-SQL 代码处理。这一特性使得 SQL Server 可以利用 NTFS 做自由形式的数据存储，超越了数据库引擎的限制，如果你以前把自由形式的数据直接塞进了数据库，那么这就是增强扩展一个很好的方式。

SQL Server 2012 用 FileTable 扩展了 FILESTREAM，它使用起来更有趣。它允许 SQL Server 应用程序把磁盘文件系统看作是数据库表，可以给它写文件，而且还有其它 Windows 应用程序(不只是其它 SQL Server 应用程序)可以识别这些文件。FileTable 使用标准的 Windows 共享来把这些文件暴露给整个 Windows 中的应用程序，这就使得 Windows 应用程序可以与它们方便地配合工作，你需要做的所有工作只是把它们指向新的共享。FileTable 甚至被实现为公共语言运行类，这样它也可以被.NET framework 方法访问。

我应该指出的是，虽然当前 CTP 中的新功能不可否认值得注意，但是不要指望在生产环境中能使用到任何一个功能，除非官方推出了 SQL Server 2012。如果你想尝鲜 2012 版本，那最好以独立的方式来做：在自己的计算机(或者虚拟计算机)上创建独立的数据库实例，使用数据副本工作，不要直接使用原始数据。微软公司有在预览版本技术就非常稳定的历史——例如，Windows 7 的与测试版本就惊人地稳定可靠——但是，你没有理由把你的数据或者你的数据库系统置于风险之中。

(作者：Serdar Yegulalp 译者：冯昀晖 来源：TT 中国)

SQL Server 2012 容量管理：FILESTREAM 和 FileTable

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_55622.htm

SQL Server 硬件选择不能犯的几个错误

建立全新的 SQL Server 系统可能比较棘手。SQL Server 是真正注重利用硬件的产品，它的性能跟你如何配置服务器有很大的关系，尤其是如何配置你服务器的存储子系统。考虑到这一点，我们在下面列出了一些人们在采购 SQL Server 硬件时最容易犯的错误：

1、选择 DIY 路线。不要购买零部件自己组装 SQL Server 计算机，除非你只是用于非生产环境的开发计算机。通常情况下，服务器尤其是 SQL Server 用的计算机需要处理器，芯片组，内存控制器卡等等零部件匹配性非常好。例如，你需要能支持高热的组件，它们必须从设计起就是考虑协同工作的。这并不是说构建自己的服务器是不可能的，但是买一台全集成的一体机会更容易一些，而且会有制造商售后支持。

2、没有性能预期。你不能简简单单地建立 SQL Server 系统，除非你知道它将承担什么样的负载。当然，你可以这么做，但是你将面临性能不足或性能过度冗余的问题，这两种情况都造成了一定资金浪费。如果你遇到的是性能不足的情况，你在将来某个时候一定需要提升服务器的处理能力，也就是说你必须再花钱进行升级(这与你的初始配置有关，升级甚至可能是行不通的)。如果你遇到的是性能过度冗余的情况，你实际上花了比你需要甚至预计需要支出的更多的钱。使用现存数据库，应用程序或者甚至供应商的基准，来获得一些性能预期，了解你期望每秒钟需要处理多少事务，并了解相应的硬件容量大小。

3、购买磁盘时只考虑容量，而不考虑磁盘性能。是的，SQL Server 通常需要大量的磁盘空间。但是，如果磁盘处理技术不够快，所有的磁盘空间都是没用的。把若干驱动器配置成 RAID5 阵列可能会满足你的空间和冗余需求，但是如果该阵列不能以一定的速度移动数据，它将成为你系统的主要性能瓶颈。如果你买不起你所需容量的快速磁盘，那你就满足不了 SQL Server 的需要。

理想情况下，数据库文件和事务日志应该放置在不同的磁盘上(或不同的阵列上)，SQL Server 应该通过不同的通道访问它们，比如磁盘控制器卡或者存储区域网络(SAN)连接。临时系统数据库如果使用频率较高，可能也需要自己独立的磁盘或阵列。

4、选择了错误的 RAID 方案。RAID 5 方案在写数据时会比较慢。大部分 RAID 控制器都尝试通过在控制器内存中缓存数据来克服这一问题(控制器内存为安全考虑通常有电池备份)，但是繁忙的 SQL Server 数据库可能会占满该缓存空间，并达到瓶颈。RAID 10 是可选的方案呢。它比 RAID 5 成本更高，但是它用数据条带连接了磁盘镜像，它能提供更高冗余和更快的读写速度。

5、购买的驱动器太少。如果你需要 XGB 或者 XTB 的存储空间，你会希望它能提供尽可能多的物理磁盘，这样能获得最快的吞吐速度。那是因为，拥有的磁盘数量越多(不管是小容量的还是大容量的)，比拥有少数大容量磁盘的性能会更好。对于条带式的阵列方案(RAID 5 和 RAID 10 都支持)，每多一块磁盘都会给 SQL Server 的性能提供一定贡献。例如，如果你可以选择购买 5 快 1TB 的硬盘或者 20 块 250GB 的硬盘，20 块 250GB 的硬盘性能(假定这些磁盘都被配置为条带式阵列，驱动器都具备相同的速度和传输率)一定会超过 5 块 1TB 硬盘的性能。

6、利用没有电池的磁盘控制器。如果你依赖磁盘控制器来缓冲写指令(例如，RAID 5 阵列)，要确保主板上有电池。要经常有计划地监视服务器电源自检测(有检测通知)屏幕，确保那些电池(通常是手表用的锂电池)持续有电。

7、盲目信任 SAN。SAN 并不是在所有情况下都是完美的选择。你必须确保有足够的吞吐量，SQL Server 不会与那么多其他服务器和应用程序共用资源，竞争带宽和吞吐量。SQL Server 需要快速存储访问，对大多数 SQL Server 主机来说，这是最大的性能瓶颈。请确保你知道 SAN 的配置(例如，要知道是 RAID 5 还是 RAID 10，要记得前面提到的错误)，要知道其吞吐量和其它细节，就像你想知道直接附加存储的这些信息一样。

8、选择 32 位。在 32 位 Windows 环境中，SQL Server 想利用大于 3GB 的内存会更困难一些，它必须采用一些页扩展技术，这种技术并不像原始手段直接访问大量内存一样高效。如果你选择了 64 位的硬件，那就在其上运行 64 位的操作系统。此外，要注意 Windows Server 2008 R2 以及 Windows 的后续版本都只对 64 位版本可用。

上面的许多错误似乎都与存储有关，SQL Server 的存储是人们最容易关注的领域，大家都太多地关注存储容量，而对其它因素(比如：吞吐量)不够重视。尤其是 SAN，存储变成了类似“我们的私有云服务”的东西，像空中的一个大魔盒，数据都在其中。

当然，SQL Server 的性能也不只是与存储有关，也跟其它因素有关，比如处理器架构和服务器内存容量。要从细节处理问题和分析性能。在为 SQL Server 购置硬件时要避免犯这些错误，这样才能得到性能更好的主机。

(作者：Don Jones 译者：冯昀晖 来源：TT 中国)

SQL Server 硬件选择不能犯的几个错误

TechTarget 中国原创内容，原文链接：

http://www.searchdatabase.com.cn/showcontent_53248.htm