SQL Server 安全跨数据库所有权链接

所有权链接

大多数的时候，你创建的数据库对象所引用的其他对象都是在同一个数据库中。比如一个存储过程所访问的表在同一个数据库，一个视图所连接的表都在同一个数据库。但是，有时你需要创建对象访问其他数据库中的对象。在以前，跨数据库访问的安全规则遵循在单一数据库中的规则。访问用户需要对访问对象有必要的权限，连续的所有权链接允许 SQL Server 简化权限检查。毕竟，SQL Server 不能放松警惕无论对象存储在哪。类似单一数据库中，所有权链接有助于简化跨数据库的安全管理。

所有权链接工作的基本原理是一样的，不管你是在一个单一的数据库还是跨数据库。所有数据库对象都有一个所有者，并且所有者控制谁在对象上有权限。对象访问其他对象，比如存储过程在 SELECT 语句把多张表联接起来，形成一个连续的所有权链接，只要一个所有者拥有所有对象。

这种情况下用户只要有顶层对象(比如存储过程、视图)的权限，访问其他对象并不需要对底层对象有权限，只要存在一个连续的所有权链接就行。SQL Server 一旦证实用户对顶层对象有权限就会停止检查底层对象的权限。这种设计为用户提供了更多更好的控制权，因为用户只需要直接访问对象的权限。

提示：所有权链接只适用于对象的权限，如 SELECT、UPDATE 和 EXECUTE 操作。SQL Server 总是检查数据定义语言语句的权限，因为这些权限应用于语句而不是对象。

跨数据库所有权链接

跨数据库所有权链接是所有权链接的一个扩充，所有权链接的所有对象(包

括用户直接访问的对象以及底层相关的对象)在同一个数据库。唯一不同的是跨数据库所有权链接跨越数据库边界。因此你可以在数据库中有一个视图将多个数据库下的表联接在一起。或者是一个存储过程访问多个数据库下的对象。在这些情况下，用户直接访问的源对象依赖于另一个数据库中的对象。

这两种所有权链的唯一重要的区别是 ,主体可以跨库成为其他对象的所有者。数据库用户也是一个主体，是完全包含在一个单一数据库中。即使多个数据库中都具有相同名称的数据库用户，这些都是独立的，不同的主体，不能参与一个完整的所有权链接，除非所有这些用户都映射到相同的服务器级别登录名。所以相关的所有者是登录名，而不是数据库用户。一个关键的概念：一个连续的所有权链接中的对象的共同拥有者是一个服务器级的主体 ,而不是一个数据库级的主体。

SQL Server 内部通过 SID 而不是名称来区别对象的所有者。在一个单一的数据库，all objects owned by a single user have a single SID specified as the owner，因为在同一数据库中用户名不能重复。但在跨数据库中，SID 是终极所有者在服务器级别的登录名对应。不同的用户在不同的数据库下可以有不同的登录名，所以会有不同的 SID。这可能是用户与登录名潜在容易混乱的方面，所以要确保你清楚这一点！

一个连续的所有权链接要求所有对象(源对象和关联对象)的所有者映射到相同的登录名及相同的 SID。

探索跨数据库所有权链接

本篇的样例代码会说明如何使用跨数据库所有权链接并解释它的特性。代码首先会在服务器级别创建一个叫做 SharedLogin 登录名。之后的代码会使用此登录名作为所有权链接中对象的共享所有者。

```
    USE master;
    GO
    IF SUSER_SID('SharedLogin') IS NOT NULL DROP LOGIN Shar
edLogin;
    CREATE LOGIN SharedLogin WITH password = 'Y&2!@37z#F!l1
zB';
    GO
```

代码 7.1 创建 SharedLogin 登录名

我们需要两个数据库来演示跨数据库所有权链接，因此接下来代码会创建它们。用户直接访问对象的数据库叫做 SourceDB，关联对象的数据库叫做 ChainedDB。ChainedDB 包含一个 dbo.AlaskaCity 表，里面有 Alaska 三大城市的人口数据。代码 7.2 会创建 ChainedDB 库和 AlaskaCity 表，然后往表中插入部分数据：

```
    IF DB_ID('ChainedDB') IS NOT NULL DROP DATABASE Chained
DB;
    CREATE DATABASE ChainedDB;
    GO
    USE ChainedDB;
    GO
    -- Create a table for access from another database
    CREATE TABLE dbo.AlaskaCity
    (
        AlaskaCityID INT NOT NULL IDENTITY(1, 1),
        CityName NVARCHAR(20) NOT NULL,
        Population INT NOT NULL
        CONSTRAINT PK_AlaskaCity PRIMARY KEY (AlaskaCityID)
```

```
    );
    GO
    INSERT INTO dbo.AlaskaCity (CityName, Population)
        VALUES ('Fairbanks', 31535), ('Anchorage', 291826),
 ('Juneau', 31275);
    GO
```

代码 7.2 创建 ChainedDB 库和 AlaskaCity 表并插入测试数据

代码 7.3 创建 SourceDB 库和一个视图，用户直接访问视图以返回 ChainedDB 库 AlaskaCity 表中的数据。代码中包含一个查询语句测试视图返回数据是否正确。如果你是以 sysadmin 身份登录这个语句应该正常执行。作为管理员运行是很方便，但是在生产使用它不安全，通常一个用户访问的代码不会有所有对象的所有权！

```
    IF DB_ID('SourceDB') IS NOT NULL DROP DATABASE SourceD
B;
    CREATE DATABASE SourceDB;
    GO
    USE SourceDB;
    GO
    -- Create a view that accesses ChainedDB.dbo.AlaskaCity
    CREATE VIEW dbo.AlaskaCitiesView AS
        SELECT * FROM ChainedDB.dbo.AlaskaCity;
    GO
    SELECT * FROM dbo.AlaskaCitiesView ORDER BY Population
DESC;
```

代码 7.3 创建 SourceDB 库和 AlaskaCitiesView 视图

现在你有两个数据库，其中一个库下的对象会访问另一库下的对象，而且在sysadmin 下执行是正常。代码 7.4 在 SourceDB 库创建一个 SourceUser 用户映射到SharedLogin登录名 ,并且将AlaskaCitiesView视图的查询权限授予给SourceUser用户。然后代码更改执行上下文到 SharedLogin 并尝试访问 AlaskaCitiesView 视图。查询会成功吗？

```
    USE SourceDB;
    GO
    -- Create a user in the SourceDB who will access the vie
w
    CREATE USER SourceUser FOR LOGIN SharedLogin;
    GRANT SELECT ON dbo.AlaskaCitiesView TO SourceUser;
    GO
    -- Try accessing the view as SourceUser
    EXECUTE AS LOGIN = 'SharedLogin';
    SELECT * FROM dbo.AlaskaCitiesView ORDER BY Population
DESC;
    GO
    REVERT;
```

代码 7.4 创建一个 SourceUser 用户映射到 SharedLogin 登录名

查询抛出一个错误：服务器主体 "SharedLogin" 无法在当前安全上下文下访问数据库 "ChainedDB"。

我们有一个连续的所有权链接：视图和表共享相同的所有者——dbo 数据库用户。但是在服务器实例或新数据库上没有启用跨数据库所有权链接，下面我们将启用它。

启用跨数据库所有权链接

新安装的 SQL Server 实例跨数据库所有权链接选项默认是关闭的。这是因为启用这个选项会在实例的安全盔甲上打开几个漏洞 ;我将在这篇后面简要地讲解风险。当选项是禁用的 ,当代码依赖跨数据库所有权链接就会生成权限拒绝错误。(正如你稍后会看到的，在代码执行前这不是唯一需要修正的问题，但它是第一个我们将解决。)

服务器级别启用

在服务器级别为所有数据库启用跨数据库所有权链接。如果你启用它，它会应用到所有数据库上而且你不能在数据库上限制它。

使用 SSMS，对象资源管理器下右击实例->属性->安全性，在对话框的底部你会看到跨数据库所有权链接的选项，如图 7.1 所示，点击确定实例下的所有数据库就会启用。

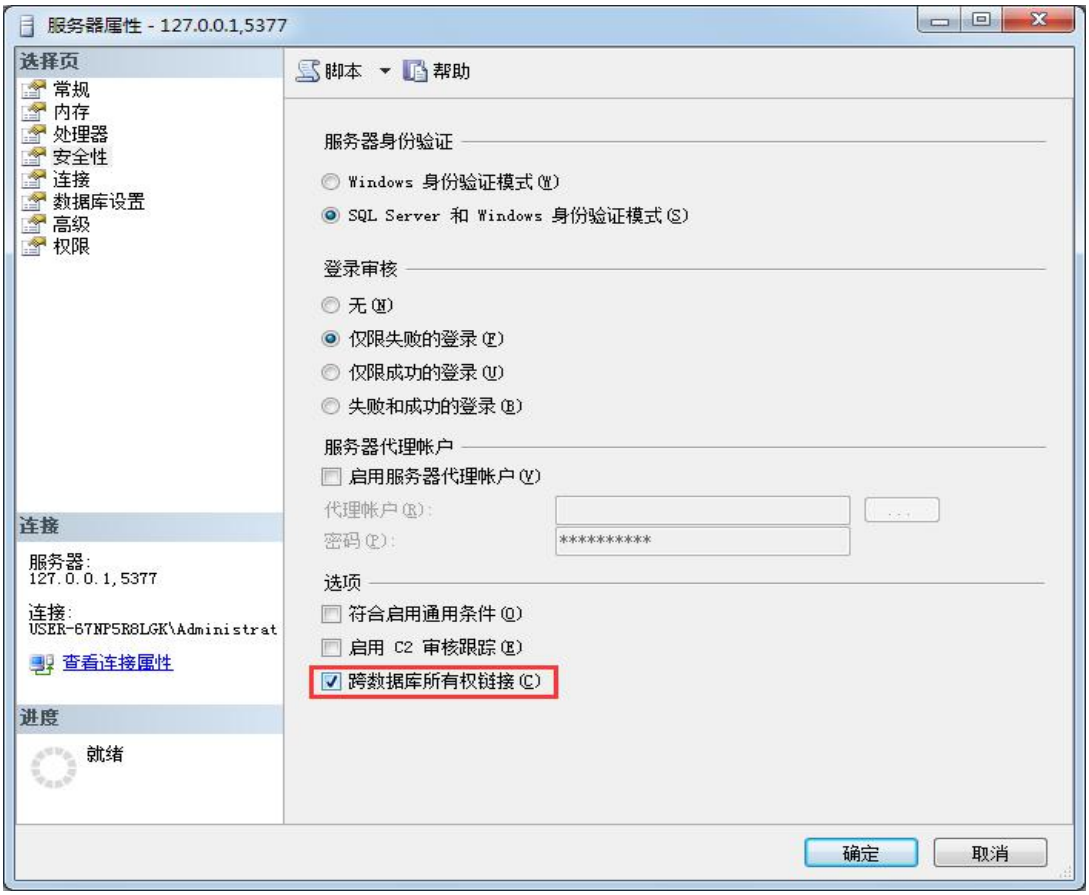

图 7.1 启用跨数据库所有权链接

你也可以用T-SQL语句启用 如代码7.5所示。比如在其他服务器实例将"cross db ownership chaining "值设为 1 启用，O 禁用。确保使用 RECONFIGURE 语句而不需要重启实例才能让更改生效。一旦你执行了这段代码，你就可以在 SQL Server 实例的任何数据库中使用跨数据库所有权链接。

```
USE master;
GO
EXECUTE sp_configure 'cross db ownership chaining', 1;
RECONFIGURE;
```

代码 7.5 使用 T-SQL 启用跨数据库所有权链接

如果你使用 SSMS 或 T-SQL 已启用了跨数据库所有权链接，请将它关闭。这

不是启用它的最好方法，除非你真的打算在实例的每个数据库下从源对象访问关联对象。如果不打算这样，在实例上启用太不安全，而且不是好主意。

作为替代，你应该针对需要的数据库启用跨数据库所有权链接。

数据库级别启用

在服务器级别禁用了跨数据库所有权链接，如果你需要使用的话，你必须在数据库级别启用它。如果一个数据库禁用了，它就不能作为源或链接数据库参与跨数据库所有权链接。为了让链接工作，在源和链接数据库都必须设置启用。

默认当你创建或附加用户数据库时跨数据库所有权链接是禁用的。但是 master、msdb 和 tempdb 系统数据库是启用的，model 数据库是禁用的。因为 SQL Server 用户内部使用跨数据库所有权链接，我们不能禁用或启用系统数据的选项。

提示：如果你分离然后再附加一个已经启用跨数据库所有权链接的数据库，你必须在数据库附加后重新启用选项。这不适合于服务器级别已启用的，服务器级别会自动应用到所有数据库上。

不幸地是，在 SSMS->数据库属性->选项，"跨数据库所有权链接已启用"选项是 False，如图 7.2 所示。为了更改数据库的这个值，你需要使用代码 7.6，启用 SourceDB 和 ChainedDB 数据库的设置。

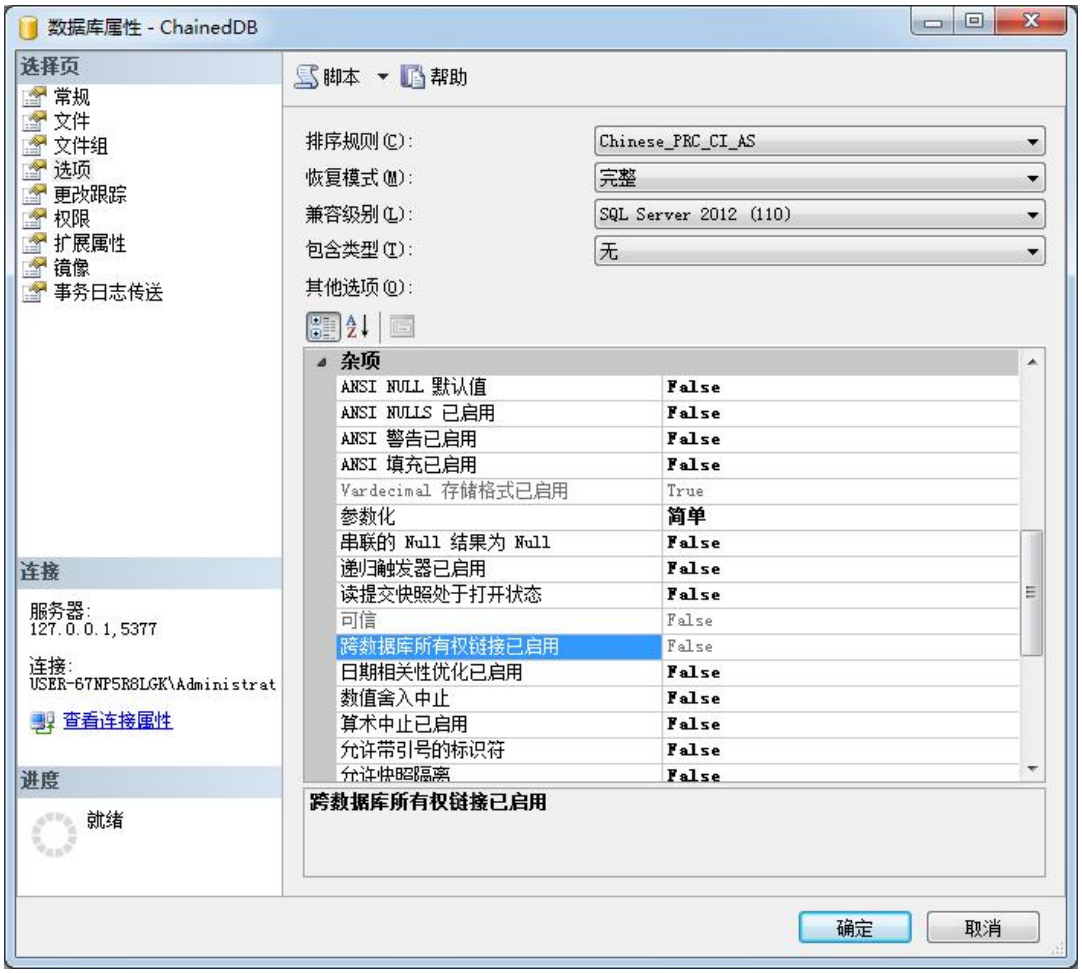

图 7.2 查看 ChainedDB 数据库跨数据库所有权链接已启用选项

```
ALTER DATABASE ChainedDB SET DB_CHAINING ON;

ALTER DATABASE SourceDB SET DB_CHAINING ON;
```

代码 7.6 启用跨数据库所有权链接

两个数据库都启用跨数据库所有权链接后，你可以尝试使用 SharedLogin 安全上下文再次访问视图：

```
USE SourceDB;

GO

EXECUTE AS LOGIN = 'SharedLogin';
```

```
    SELECT * FROM dbo.AlaskaCitiesView ORDER BY Population
DESC;

    GO

    REVERT;
```

代码 7.7 尝试使用跨数据库所有权链接跨过数据库边界访问对象

还是抛出同样的错误：服务器主体 "SharedLogin" 无法在当前安全上下文下访问数据库 "ChainedDB"。这次的问题是要满足一个需求。用户访问视图必须也能够访问 ChainedDB 数据库，用户不需底层对象的权限。事实上，用户不需要在那个数据库下的任何权限。

代码 7.8 展示了如何解决这个问题。在 ChainedDB 库创建一个 ChainedUser 用户映射到 SharedLogin 登录名。这给 SharedLogin 在 ChainedDB 库一个立足点，跨数据库链接工作的必要条件。为了验证 SharedLogin 在 AlaskaCity 表上没有查询权限，代码尝试直接从 AlaskaCity 表上读取数据。但是尝试失败，因为 SharedLogin 和 ChainedUser 在 ChainedDB 库没有任何权限。这次的错误是：拒绝了对对象 'AlaskaCity' (数据库 'ChainedDB'，架构 'dbo')的 SELECT 权限。

```
    USE ChainedDB;

    GO

    CREATE USER ChainedUser FOR LOGIN SharedLogin;

    -- Note that we're not granting the user any permission
s in the chained database.

    GO

    -- Verify that SharedLogin doesn't have direct access t
o the AlaskaCity table, even in the ChainedDB database cont
ext.

    EXECUTE AS LOGIN = 'SharedLogin';

    SELECT * FROM dbo.AlaskaCity;
```

```
GO
REVERT;
```

代码 7.8 ChainedDB 库创建一个 ChainedUser 用户映射到 SharedLogin 登录名

现在你可以返回执行代码 7.7，它会成功并返回视图中的数据！

使用代码 7.9 禁用数据库所有权链接——实际上你只需要执行代码中任何一条语句，因为要让跨数据库所有权链接正常工作两个数据库的都要设置成 ON。然后你再运行代码 7.7，此时对象所有者相同且用户能访问两个数据库。这次报错 :拒绝了对对象 'AlaskaCity' (数据库 'ChainedDB' ,架构 'dbo')的 SELECT 权限。

```
ALTER DATABASE ChainedDB SET DB_CHAINING OFF;
ALTER DATABASE SourceDB SET DB_CHAINING OFF;
```

代码 7.9 禁用数据库所有权链接

共用所有权

样例代码中的视图在启用跨数据库所有权链接后能正常执行的原因是 :视图和表共用所有权。样例代码假定是以 sysadmin 登录到 SSMS 编写的，因此代码所创建的对象都被 dbo 用户所拥有，用户都是映射到 sysadmin。你可以用代码 7.10 验证映射关系，显示架构名称、所有者用户名、所有者登录名。如果你在 SourceDB 和 ChainedDB 库下运行，你会发现所有者登录名是相同的，如图 7.3 和 7.4 所示，在我的例子中，dbo 用户映射到登录名 USER-67NP5R8LGK\Administrator.

```
SELECT
    so.[name] AS Object,
    sc.[name] AS [Schema],
```

```
       USER_NAME(COALESCE(so.principal_id, sc.principal_i
d)) AS OwnerUserName,

       sp.name AS OwnerLoginName,

       so.type_desc AS ObjectType

   FROM sys.objects so

       JOIN sys.schemas sc ON so.schema_id = sc.schema_id

       JOIN sys.database_principals dp ON dp.principal_id
= COALESCE(so.principal_id, sc.principal_id)

       LEFT JOIN master.sys.server_principals sp ON dp.sid
= sp.sid

   WHERE so.[type] IN ('U', 'V');
```

代码 7.10 显示对象架构名称、所有者用户名、所有者登录名

	Object	Schema	OwnerUserName	OwnerLoginName	ObjectType
1	AlaskaCitiesView	dbo	dbo	USER-67NP5R8LGK\Administrator	VIEW

图 7.3 SourceDB 库下运行结果

	Object	Schema	OwnerUserName	OwnerLoginName	ObjectType
1	AlaskaCity	dbo	dbo	USER-67NP5R8LGK\Administrator	USER_TABLE

图 7.4 ChainedDB 库下运行结果

要注意的是一个数据库的 dbo 用户不一定和另一个数据库下的 dbo 用户相同。它们可以被映射到完全不同的登录名或者不映射。重要的是链接中所有对象的拥有者有相同的 SID。你可以用代码 7.11 验证对象所有者和登录名有相同的 SID。

```
   SELECT name, sid FROM SourceDB.sys.database_principals
WHERE name = 'dbo'

   UNION ALL

   SELECT name, sid FROM ChainedDB.sys.database_principals
 WHERE name = 'dbo'

   UNION ALL
```

```
    SELECT 'USER-67NP5R8LGK\Administrator', SUSER_SID('USE
R-67NP5R8LGK\Administrator');
```

代码 7.11 验证对象所有者和登录名有相同的 SID

	name	sid
1	dbo	0x0105000000000005150000001AC4EF08FBEBE719223C840CF4010000
2	dbo	0x0105000000000005150000001AC4EF08FBEBE719223C840CF4010000
3	USER-67NP5R8LGK\Administrator	0x0105000000000005150000001AC4EF08FBEBE719223C840CF4010000

图 7.5 SourceDB 和 ChainedDB 库下 dbo 用户，及代码会话登录用户的 SID