عنوان: **Identity و مباحث مربوطه (قسمت دوم) نحوه بدست آوردن مقادیر Identity**
نویسنده: فرید طاهری
تاریخ: ۱۳۹۲/۰۳/۲۵ ۱۷:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, T-SQL, Identity

همانگونه که می‌دانید مقدار Identity پس از درج به آن تخصیص می‌یابد چنانچه بخواهید به این مقدار دسترسی پیدا کنید چندین روش به ازای اینکار وجود دارد که ما در این مقاله سه روش معمول را بررسی خواهیم نمود.

1- استفاده از متغییر سیستمی @@Identity

2- استفاده از تابع Scope_Identity ()

3- استفاده از تابع Ident_Current

هر سه این توابع مقدار Identity ایجاد شده برای جداول را نمایش می‌دهند. اما تفاوت هایی باهم دارند که در ادامه مقاله این تفاوت‌ها بررسی شده است.

1- متغییر سیستمی @@Identity : این متغییر سیستمی حاوی آخرین Identity ایجاد شده به ازای Session جاری شما است. لازم به ذکر است اگر به واسته Insert شما، Identity دیگری در یک حوزه دیگر (مانند یک Trigger) ایجاد شود مقدار موجود در این متغییر حاوی آخرین Identity ایجاد شده است. (یعنی Identity ایجاد شده توسط آن تریگر و نه خود جدول). لازم به ذکر است این موضوع به طور کامل در ادامه مقاله شرح داده شده است.

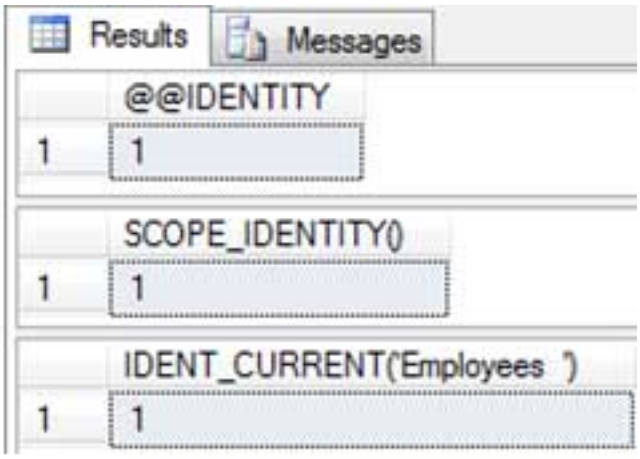
2- استفاده از تابع() Scope_Identity : با استفاده از این تابع می‌توانیم آخرین Identify ایجا دشده به ازای Session جاری را بدست آوریم. لازم به ذکر است مقادیر Identity ایجاد شده توسط سایر حوزه‌ها تاثیر در مقدار بازگشتی توسط این تابع ندارد. در ادامه مقاله این موضوع به طور کامل بررسی شده است.

3- استفاده از تابع ident_Current : این تابع آخرین مقدار Identity موجود در یک جدول را نمایش می‌دهد. ذکر این نکته ضروری است که Identity ایجاد شده توسط سایر Sessionها هم روی خروجی این تابع تاثیرگذار است. چون این تابع آخرین Identity موجود در جدول را به شما نمایش می‌دهد و نه Identity ایجاد شده به ازای یکSession را.

برای بدست آوردن یک Identity کافی است که پس از درج رکورد در جدول مورد نظر متغییر سیستمی @@Identity و یا توابع Scope_Identity و یا Ident_Current را همانند مثال زیر Select کنید.

```
USE TEMPDB
GO
IF OBJECT_ID(N'Employees', N'U') IS NOT NULL
    DROP TABLE Employees1;
GO
CREATE TABLE Employees
(
ID int IDENTITY,
FirstName NVARCHAR(50),
LastName NVARCHAR(50)
)
GO
INSERT INTO Employees (FirstName,LastName) VALUES (N'مسعود',N'طاهری')
GO
SELECT @@IDENTITY AS [@@IDENTITY]
SELECT SCOPE_IDENTITY() AS [SCOPE_IDENTITY()]
SELECT IDENT_CURRENT('Employees1') AS [IDENT_CURRENT('Employees1')]
GO
```

خروجی دستورات بالا پس از درج رکورد مورد نظر به صورت زیر است.



اما ممکن است از خودتان این سوال را بپرسید که آیا این توابع در سطح شبکه آخرین مقدار Identity درج شده توسط سایر Sessionها را نمایش می‌دهند و یا Session جاری را؟ (منظور Sessionی که درخواست مقدار موجود در identity را نموده است).

برای دریافت پاسخ این سوال مطابق مراحل اسکریپ‌های زیر را اجرا نمایید.

-1ایجاد جدول Employees1

```
USE TEMPDB GO
IF OBJECT_ID(N'Employees1', N'U') IS NOT NULL
    DROP TABLE Employees1;
GO
CREATE TABLE Employees1
(
ID int IDENTITY(1,1),
FirstName NVARCHAR(50),
LastName NVARCHAR(50)
)
GO
```

همانطور که مشاهده می‌کنید مقدار شروع برای Identity برابر 1 و گام افزایش هم برابر 1 در نظر گرفته شده است(Identity(1,1)) .

-2در Sessionی جدید دستورات زیر را اجرا نمایید. (درج رکورد جدید در جدول Employees1 و واکشی مقدار Identity)

USE tempdb

```
GO
INSERT INTO Employees1(FirstName,LastName) VALUES (N'فرید',N'طاهری')
GO
SELECT @@IDENTITY AS [@@IDENTITY]
SELECT SCOPE_IDENTITY() AS [SCOPE_IDENTITY()]
SELECT IDENT_CURRENT('Employees1') AS [IDENT_CURRENT('Employees1')]
GO
```

SPID مربوط به
Connection جاری

SQLQuery2.sql - (lo...Administrator (55))* | SQLQuery1.sql - (lo...Administrator (53))* | Ider

```
USE tempdb
GO
INSERT INTO Employees1(FirstName,LastName) VALUES (N'فرید',N'ظاهری'
GO
SELECT @@IDENTITY AS [@@IDENTITY]
SELECT SCOPE_IDENTITY() AS [SCOPE_IDENTITY()]
SELECT IDENT_CURRENT('Employees1') AS [IDENT_CURRENT('Employees1')]
GO
```

100 %

Results | Messages

	@@IDENTITY
1	1

	SCOPE_IDENTITY()
1	1

	IDENT_CURRENT('Employees1')
1	1

همانگونه که ملاحظه می‌کنید @@Scope_Identity ، Identity() و Ident_Current هر سه مقدار Identity (عدد 1) ایجاد شده بوسیله دستور Insert را به شما نمایش می‌دهند.

1- و در انتها در یک Session دیگر دستورات زیر را اجرا نمایید. (واکشی مقدار Identity)

```
USE tempdb
GO
SELECT @@IDENTITY AS [@@IDENTITY]
SELECT SCOPE_IDENTITY() AS [SCOPE_IDENTITY()]
SELECT IDENT_CURRENT('Employees1') AS [IDENT_CURRENT('Employees1')]
GO
```

همانطور که مشاهده می‌کنید در این Seesion ما از SQL خواسته‌ایم آخرین مقدار Identity را به ما نشان داده شود. باید به این نکته توجه کنید با توجه به اینکه در این Session عملیات درجی هنوز انجام نگرفته است که ما Identity ایجاد شده را مشاهده نماییم. بنابراین صرفاً تابع Iden_Current مقدار Identity موجود در جدول را به ما نمایش می‌دهد.

پس می‌توان به این نکته رسید که

Idnetity@@ و Identity : **Scope_Identity** ایجاد به ازای Session جاری را نمایش داده و به مقادیر تولید شده توسط سایر Session های دیگر دسترسی ندارد.

Ident_Current : آخرین Identity موجود در جدول را به شما نمایش می‌دهد. بنابراین باید این نکته را در نظر داشته باشید که Identity ها ایجاد شده توسط سایر Session ها روی مقدار بازگشتی این تابع تاثیرگذار است.

اما یکی دیگر از مباحث مهم درباره Identity تاثیر Scope بر مقدار Identity است (یعنی چه!) . برای اینکه با مفهوم این موضوع آشنا شوید اسکریپت‌های مربوط به مثال زیر را بدقت اجرا کنید.

1- ایجاد جدول Employees1

```
USE TEMPDB
GO
IF OBJECT_ID(N'Employees1', N'U') IS NOT NULL
    DROP TABLE Employees1;
GO
CREATE TABLE Employees1
(
ID int IDENTITY(1,1),
FirstName NVARCHAR(50),
LastName NVARCHAR(50)
)
GO
```

همانطور که مشاهده می‌کنید مقدار شروع برای Identity برابر 1 و گام افزایش هم برابر 1 در نظر گرفته شده است((Identity(1,1)) .

2- ایجاد جدول Employees2

```
USE TEMPDB
GO
IF OBJECT_ID(N'Employees2', N'U') IS NOT NULL
    DROP TABLE Employees2;
GO
CREATE TABLE Employees2
(
ID int IDENTITY(100,1),
FirstName NVARCHAR(50),
LastName NVARCHAR(50)
)
GO
```

همانطور که مشاهده می‌کنید مقدار شروع برای Identity برابر 100 و گام افزایش هم برابر 1 در نظر گرفته شده است((Identity(100,1)).

3- ایجاد یک Trigger به ازای جدول Employees1

```
USE tempdb
GO
CREATE TRIGGER Employees1_Insert ON Employees1 FOR INSERT
AS
BEGIN
   INSERT Employees2(FirstName,LastName)
SELECT FirstName,LastName FROM INSERTED
END;
GO
```

Trigger ایجاد شده به ازای جدول Employees1 به ازای عملیات Insert اجرا می‌شود. همچنین مقادیر درج شده در جدول Employees1 بوسیله جدول Inserted در دسترس است. لازم به ذکر است جدول Inserted یک جدول موقت بوده که توسط Trigger ایجاد شده و داخل خود آن معتبر است.

هدف ما از ایجاد این Trigger تهیه یک کپی از رکوردهایی که در جدول Employees1 درج می‌شوند است. این کپی قرار است با استفاده از دستور Insert...Select در جدول Employees2 ایجاد گردد.

4- درج یک رکورد در جدول Employees1 و واکشی مقدار Identity

```
USE tempdb
GO
INSERT INTO Employees1(FirstName,LastName) VALUES (N'مسعود',N'طاهری')
GO
SELECT @@IDENTITY AS [@@IDENTITY]
SELECT SCOPE_IDENTITY() AS [SCOPE_IDENTITY()]
SELECT IDENT_CURRENT('Employees1') AS [IDENT_CURRENT('Employees1')]
SELECT IDENT_CURRENT('Employees2') AS [IDENT_CURRENT('Employees2')]
GO
```

Results | Messages

	@@IDENTITY
1	100

	SCOPE_IDENTITY()
1	1

	IDENT_CURRENT('Employees1')
1	1

	IDENT_CURRENT('Employees2')
1	100

مقادیر استخراج شده به ازای Identity به شرح زیر است

1- @@Identity : پس از درج رکورد در جدول Employees1 متغییر سیستمی @@Identity مقدار 100 را نمایش داده است دلیل این موضوع بر می‌گردد به Trigger موجود در جدول Employees1 .

با توجه به اینکه جدول Employees1 دارای یک فیلد Identity بوده است هنگام درج رکورد در جدول مقدار @@Identity=1 است اما چون این جدول دارای Trigger ی است که این Trigger خود با جدولی دیگری درگیر است که دارای Identity است مقدار متغییر @@identity=100 خواهد شد.

2- Scope_Identity() : مقدار نمایش داده شده توسط تابع Scope_Identity() برابر با مقدار Identity تخصیص (عدد 1) داده شده به ازای رکورد شما می‌باشد که این موضوع در اغلب موارد مد نظر برنامه‌نویسان می‌باشد.

3- Ident_Current('Employees1') : مقدار نمایش شده توسط تابع Ident_Current آخرین مقدار Identity (عدد 1) موجود در جدول Employees1 است.

4- Ident_Current('Employees2') : مقدار نمایش شده توسط تابع Ident_Current آخرین مقدار Identity (عدد 100) موجود در جدول Employees2 است.

چند نکته مهم

1- مقدار بازگردانده شده توسط تابع Ident_Current آخرین مقدار Identity موجود در جدول مورد نظر شما بوده است و عملیات درج سایر کاربران در این مقدار تاثیر گذار است.

2- برای بدست آوردن مقدار Identity درست بهتر است از تابع Scope_Identity() استفاده نماییم. معمولاً در بیشتر مواقع مقدار بازگردانده شده توسط این تابع مد نظر برنامه نویسان است.

3- EntityFramework و Nhibernate هم برای بدست آوردن Identity از تابع Scope_Identity استفاده می‌کند.

نظرات خوانندگان

نویسنده: کاربر
تاریخ: ۱۳۹۲/۰۴/۳۱ ۴:۵۰

ببین دوست من مطلبتون رو خوندم هم اینو و هم قبلی رو، ازش خوشم اومد اما چیزی راجب درج صریح یا بروز رسانی مقادیر Identity ننوشته بودین. یا اینکه نمیشه در یک جدول دو identity property داشت.
من بلدم با set identity_insert table_name on/off کاری کنم که خودم دستی مقداری را برای خصیصه identity لحاظ کنم. ولی متاسفانه نتونستم مقدار یک ستون با خصیصه Identity رو بروز رسانی (یا همون update) کنم. لطفا بهم بگید که اصلا این کار ممکنه یا من بلد نیستم. البته براساس query زیر بمن SQL Server گفته که نمیشه این ستون را update کرد که ظاهرا هم همین طور(ستون id همانطور که در پیام آمده از نوع identity هست)

```
update t
set id = new_id
from (select id, row_number() over(order by id) new_id from #temp)t

--Cannot update identity column 'id'.
```

اصلا اجازه بدین یه جور دیگه سوال رو مطرح کنم من نیاز دارم تمام مقادیر identity رو بروز رسانی کنم تا کاملا پشت سر هم و متوالی بشن این کار را میتونم با یک تابع row_number و یک derived table انجام بدم (اگر بذارن!) همانطور که قبلا نشان دادم، یا با روش زیر این کار را بکنم که البته اجرا نمیشه به این دلیل که در یک جدول نمیشه دو identity property داشت. با فرض اجرا شدن دستور select into باز هم در دستور update با مشکل بر می‌خوردیم (چون نمیشه ستون id را بروز رسانی کرد)

```
select id, identity(int, 1,1) new_id
into #temptable
from #temp
order by id asc

/*
cannot add identity column, using the SELECT INTO statement, to table '#temptable',
 which already has column 'id' that inherits the identity property.
*/
update t
set id = new_id
from #temp t
join #temptable d
on t.id = d.id;
```

البته یک راهی برای حل این مساله هست اونم اینه که ابتدا بیاییم تمام داده‌ها جدول را در جدول دیگه ای درج کنیم سپس تمام داده‌های جدول را حذف کنیم سپس داده‌های حذف شده را با id جدید و مرتب شده در جدول اول درج کنیم. به این شکل

```
declare @t table(id int)

insert into @t
select id from #temp

delete from #temp

set identity_insert #temp on
insert #temp (id)
select row_number() over(order by id) from @t
set identity_insert #temp off
```

اما مشکلی که وجود داره اینه که اگر جدول ما parent باشه با مشکل واجه میشیم تمام سطرهای جداول child یتیم میشن.

من قصد ندارم صورت مساله نقد و بررسی بشه و اصولی بودن یا صحیح بودنش مورد ارزیابی قرار بگیره فقط برام این یک سوال شده.

مساله عمومی که راجب این ستون وجود داره استفاده کردن از Gapهای حاصل شده در این ستون برای درج‌های بعدی است. که query آن نیز بسیار ساده و در دسترس است.
آیا شما میدانید که چگونه این مشکل با sequence ای که در نسخه 2012 معرفی شده است حل می‌شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۳۱ ۱۴:۵۵

- خیر. چندین نوع استراتژی برای تعیین PK وجود دارند که یکی از آن‌ها فیلدهای Identity است و این تنها روش و الزاما بهترین روش نیست.
- مثلا زمانیکه با ORMها کار می‌کنید استفاده از فیلدهای Identity در حین ثبت تعداد بالایی از رکوردها مشکل ساز می‌شوند. چون این فیلدها تحت کنترل دیتابیس هستند و نه برنامه، ORM نیاز دارد پس از هربار Insert یکبار آخرین Id را از بانک اطلاعاتی واکشی کند. همین مساله یعنی افت سرعت در تعداد بالای Insertها (چون یکبار کوئری Insert باید ارسال شود و یکبار هم یک Select اضافی دوم برای دریافت Id تولیدی توسط دیتابیس).
- روش دوم تعیین PK استفاده از نوع Guid است. در این حالت، هم مشکل حذف رکوردها و خالی شدن یک شماره را در این بین ندارید و هم چون عموما تحت کنترل برنامه است، سرعت کار کردن با آن بالاتر است. فقط تنها مشکل آن زیبا نبودنش است در مقایسه با یک عدد ساده فیلدهای Identity.

در مورد فیلدهای Identity، تغییر شماره Id به صلاح نیست چون:
الف) همانطور که عنوان کردید روابط بین جداول را به هم خواهد ریخت.
ب) در یک وب سایت و یا هر برنامه‌ای، کلا آدرس‌ها و ارجاعات قدیمی را از بین می‌برد. مثلا فرض کنید شماره این مطلب 1381 است و شما آن‌را یادداشت کرده‌اید. در روزی بعد، برنامه نویس شماره Idها را کلا ریست کرده. در نتیجه یک هفته بعد شما به شماره 1381 ایی خواهید رسید که تطابقی با مطلب مدنظر شما ندارد (حالا فرض کنید که این عدد شماره پرونده یک شخص بوده یا شماره کاربری او و نتایج و خسارات حاصل را درنظر بگیرید).
ج) این خوب است که در بین اطلاعات یک ردیف خالی وجود دارد. چون بر این اساس می‌توان بررسی کرد که آیا واقعا رکوردی حذف شده یا خیر. گاهی از اوقات کاربران ادعا می‌کنند که اطلاعات ارسالی آن‌ها نیست در حالیکه نبود این رکوردها به دلیل حذف بوده و نه عدم ثبت آن‌ها. با بررسی این Idها می‌شود با کاربران در این مورد بحث کرد و پاسخ مناسبی را ارائه داد.

و اگر شماره‌ای که به کاربر نمایش می‌دهید فقط یک شماره ردیف است (و از این لحاظ می‌خواهید که حتما پشت سرهم باشد)، بهتر است یک View جدید ایجاد کنید تا این Id خود افزاینده را تولید کند (بدون استفاده از pk جدول).

پ.ن.
هدف من از این توضیحات صرفا عنوان این بود که به PK به شکل یک فیلد read only نگاه کنید. این دقیقا برخوردی است که Entity framework با این مفهوم دارد و صحیح است و اصولی. اگر در یک کشور هر روزه عده‌ای به رحمت ایزدی می‌روند به این معنا نیست که سازمان ثبت احوال باید شماره شناسنامه‌ها را هر ماه ریست کند!

نویسنده: فرید طاهری
تاریخ: ۱۳۹۲/۰۴/۳۱ ۲۰:۰

با تشکر از آقای نصیری و پاسخ مناسبی که ارائه کرده اند
در مورد استفاده از GUID به جای identity باید به یک نکته هم اشاره کنم که در بیشتر مواقع اگر مقدار GUIDی که به ازای یک فیلد UNIQUEIDENTIFIER تنظیم می‌کنید به صورت SEQUNTIAL نباشد باعث Fragment شدن ایندکس خواهد شد.
برای مقایسه بهتر بین Fragmentation ایندکس مربوط به Identity و GUID به مثال زیر دقت کنید. هر دو مثال فیلد ID خود را به شکل Clustered Index دارند بعد از درج تعدادی رکورد مساوی در دو جدول Fragmentation مربوط به جدولی که دارای GUID است به شدت بالا است که این موضوع باعث کاهش کارایی خواهد شد

```
USE TEMPDB
GO
```

```
IF OBJECT_ID('TABLE_GUID')>0
DROP TABLE TABLE_GUID
GO
CREATE TABLE TABLE_GUID
(
ID UNIQUEIDENTIFIER PRIMARY KEY,
FirstName NVARCHAR(1000),
LastName NVARCHAR(1000)
)
GO
IF OBJECT_ID('TABLE_IDENTITY')>0
DROP TABLE TABLE_IDENTITY
GO
CREATE TABLE TABLE_IDENTITY
(
ID INT IDENTITY PRIMARY KEY,
FirstName NVARCHAR(1000),
LastName NVARCHAR(1000)
)
GO
INSERT INTO TABLE_GUID(ID,FirstName,LastName) VALUES
(NEWID(),REPLICATE('FARID*',100),REPLICATE('Taheri*',100))
GO 10000

INSERT INTO TABLE_IDENTITY(FirstName,LastName) VALUES
(REPLICATE('FARID*',100),REPLICATE('Taheri*',100))
GO 10000

--Fragmentation بررسی وضعیت
SELECT * FROM sys.dm_db_index_physical_stats(DB_ID(),OBJECT_ID('TABLE_GUID'),NULL,NULL,'DETAILED')
DBCC SHOWCONTIG(TABLE_GUID)
GO
SELECT * FROM sys.dm_db_index_physical_stats(DB_ID(),OBJECT_ID('TABLE_IDENTITY'),NULL,NULL,'DETAILED')
DBCC SHOWCONTIG(TABLE_IDENTITY)
GO
```

خوب برای اینکه Fragmentation این نوع جداول را رفع کنید چند راه داریم

1- تولید GUID به صورت Sequential (لازم می‌دانم اشاره کنم این قابلیت در SQL Server وجود دارد ولی مقدار تولید شده باید به شکل یک Default Constraint باشد که این موضوع نیازمند این است که شما اگر در سورس به این GUID نیاز پیدا کنید مجبور به زدن Select و... شوید. اگر بخواهید در سورس این کار را انجام دهید باید از Extentionهایی که برای اینکار وجود دارند استفاده کنید فکر کنم Nhibernate این حالت رو پشتیبانی کنه در مورد EF دقیقا اطلاع ندارم باید اهل فن نظر بدن)

2- تنظیم مقدار Fillfactor به ازای ایندکس

3-Rebuild و یا Reorganize دوره ای ایندکس