عنوان مقاله: تغییرات زبان T-SQL در SQL Server ۲۰۲۲ – پارت اول

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: ۱۱ دی ۱۴۰۱

منبع: https://nikamooz.com/ t-sql-language-changes-in-sql-server-۲۰۲۲-part-۱

پایگاه داده SQL Server 2022 از آگوست ۲۰۲۲ در مرحله پیش‌نمایش است و احتمالاً این ورژن اواخر امسال منتشر خواهد شد. من این پلتفرم را با این ایده آزمایش می‌کنم که قصد دارم در سال آینده چند سرور را ارتقا دهم. تغییرات زیادی در پلتفرم پایگاه داده به وجود آمده اما تغییراتی که مربوط به روال توسعه می‌شود، برای من جذاب‌تر است.

این مقاله برخی از تغییرات T-SQL را که در SQL Server 2022 انجام شده را پوشش می‌دهد. مواردی که ارائه خواهم داد عبارتند از DISTINCT FROM، DATE_BUCKET، GENERATE_SERIES، GREATEST/LEAST، STRING_SPLIT و DATETRUNC.

DISTINCT FROM

اگر چه این تغییر خیلی به نظرم جالب نبود اما پس از انتشار از آن استفاده کردم و متوجه شدم که ممکن است مفید باشد. برطبق مستندات ارائه شده برای IS [NOT] DISTINCT FROM این تابع، برابری دو عبارت را مقایسه و درست یا نادرست بودن را بررسی می‌کند؛ حتی اگر یکی از پارامترهای مورد بررسی NULL باشد.

به مثال زیر دقت کنید:

```
DECLARE @a INT, @b INT
SELECT @a = 1, @b = 1
IF @a = @b
 SELECT 'equal', @a, @b
ELSE
 SELECT 'unequal', @a, @b
SELECT @a = 1, @b = 2
IF @a = @b
 SELECT 'equal', @a, @b
ELSE
 SELECT 'unequal', @a, @b
 SELECT @a = 1, @b = null
IF @a = @b
 SELECT 'equal', @a, @b
ELSE
 SELECT 'unequal', @a, @b
```

نتایج زیر را خواهیم داشت:

Results Messages

	(No column name)	(No column name)	(No column name)
1	equal	1	1

	(No column name)	(No column name)	(No column name)
1	unequal	1	2

	(No column name)	(No column name)	(No column name)
1	unequal	1	NULL

برای مثال دوم از کد زیر استفاده می‌کنم:

```
DECLARE
  @a INT
, @b INT;
SELECT @a = 1, @b = 1;
IF @a = @b
  SELECT 'equal', @a, @b;
ELSE IF @a IS NULL OR @b IS NULL
  SELECT 'unknown', @a, @b;
ELSE
  SELECT 'unequal', @a, @b;
SELECT @a = 1, @b = 2;
IF @a = @b
  SELECT 'equal', @a, @b;
ELSE IF @a IS NULL OR @b IS NULL
  SELECT 'unknown', @a, @b;
ELSE
  SELECT 'unequal', @a, @b;
SELECT @a = 1, @b = NULL;
IF @a = @b
  SELECT 'equal', @a, @b;
ELSE IF @a IS NULL OR @b IS NULL
  SELECT 'unknown', @a, @b;
ELSE
  SELECT 'unequal', @a, @b;
```

این کد نتایج مناسب را به صورت "equal" ، "unequal" و "unknown" می‌دهد.

بدون این تابع جدید، وقتی موارد را مقایسه می‌کنیم، می‌توانیم سه حالت true، false یا NULL داشته باشیم. با وجود این تابع جدید، می‌توانیم روال را به صورت زیر انجام دهیم:

```
DECLARE
  @a INT
, @b INT;
SELECT @a = 1, @b = NULL;
IF @a IS DISTINCT FROM @b
  SELECT 'unequal', @a, @b;
ELSE
  SELECT 'equal', @a, @b;
```

نتیجه در اینجا "unequal" است حتی اگر یکی از مقادیر NULL باشد.

این تابع وقتی کاربردی می‌شود که نمی‌خواهم در داخل کد ابتدا NULL بودن یک عبارت را چک کنم و سپس کار مقایسه را انجام دهم. بااستفاده از تابع DISTINCT FROM بدون کنترل NULL بودن، می‌توان مقایسه را انجام داد. به مثال زیر دقت کنید:

```
DECLARE @cc INT = 5618;
SELECT soh.SalesOrderID
     , soh.OrderDate
     , soh.SalesOrderNumber
     , soh.CustomerID
     , soh.CreditCardID
     , soh.CreditCardApprovalCode
 FROM sales.SalesOrderHeader AS soh
 WHERE soh.CreditCardID = @cc
```

این کد چند سفارش را به من برمی‌گرداند. با این حال، اگر ورودی را به NULL تغییر دهم، به صورت زیر خواهد بود:

```
DECLARE @cc INT = null;
SELECT soh.SalesOrderID
     , soh.OrderDate
     , soh.SalesOrderNumber
     , soh.CustomerID
     , soh.CreditCardID
     , soh.CreditCardApprovalCode
 FROM sales.SalesOrderHeader AS soh
 WHERE soh.CreditCardID = @cc
```

Messages

SalesOrderID	OrderDate	SalesOrderNumber	CustomerID

در SQL Server می‌توانم تغییر زیر را اعمال کنم. در این مورد، از IS NOT DISTINCT FROM استفاده می‌کنم. این کد باعث می‌شود ردیف‌های NULL مشخص شوند.

```
DECLARE @cc INT = null;
SELECT soh.SalesOrderID
     , soh.OrderDate
     , soh.SalesOrderNumber
     , soh.CustomerID
     , soh.CreditCardID
     , soh.CreditCardApprovalCode
 FROM sales.SalesOrderHeader AS soh
 WHERE soh.CreditCardID IS NOT DISTINCT FROM @cc
```

Messages

SalesOrderID	OrderDate	SalesOrderNumber	CustomerID	CreditCardID	CreditCardApprova
43737	2005-07-11 00:00:00.000	SO43737	13261	NULL	NULL
43739	2005-07-11 00:00:00.000	SO43739	13563	NULL	NULL
43787	2005-07-22 00:00:00.000	SO43787	29385	NULL	NULL
43837	2005-07-30 00:00:00.000	SO43837	11009	NULL	NULL
43924	2005-08-02 00:00:00.000	SO43924	12132	NULL	NULL
43970	2005-08-11 00:00:00.000	SO43970	16684	NULL	NULL

بدون این تابع، من باید ابتدا یک عبارت OR اضافه می‌کردم تا ISNULL را بررسی کند، هم برای پارامتر ورودی و هم برای مقداری که بررسی می‌شود. در حالی که با تابع IS NOT DISTINCT FROM نیاز به این روال‌های اضافی نیست.

DATE_BUCKET

تابع DATE_BUCKET تاریخ شروع یک دوره زمانی را براساس window یا bucket ای که شما مشخص کرده‌اید به شما می‌دهد. این تابع برای موقعیت‌هایی که دوره‌های زمانی را گروه‌بندی یا محاسبه می‌کنید مفید است. در این قسمت با ارائه چند مثال ساده، کاربرد این تابع را ارائه می‌دهم.

فرض کنید من می‌خواهم سال را به bucket های ٤ هفته‌ای تقسیم کنم. در تقویم ۲۰۲۲، این bucket های ٤ هفته به صورت زیر درنظر گرفته شد:

- 1 Jan to 28 Jan
- 29 Jan to 25 Feb
- 26 Feb to 25 Mar

به همین ترتیب دوره‌های زمانی در نظر گرفته می‌شود.

اگر تاریخ شروع خود را ۱ ژانویه تنظیم کنم و تاریخ را در یک بازه زمانی انتخاب کنم، شروع هر bucket برگردانده می‌شود. من ٤ تاریخ را در قسمت‌های مختلف برای چند ماه اول، برای نشان دادن این موضوع انتخاب کرده‌ام. کد به صورت زیر است:

```
DECLARE @origin DATE = '2022/01/01';
DECLARE @bucketsize INT = 4
SELECT
 date_bucket(week, @bucketsize, CAST('2022/01/15' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/01/30' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/02/25' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/03/04' as date), @origin)
```

نتایج را همان‌طور که در تصویر زیر می‌بینید، اولین روز از هر bucket چهار هفته‌ای برگردانده می‌شود:

```
DECLARE @origin DATE = '2022/01/01';
DECLARE @bucketsize INT = 4

SELECT
 date_bucket(week, @bucketsize, CAST('2022/01/15' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/01/30' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/02/25' as date), @origin),
 date_bucket(week, @bucketsize, CAST('2022/03/04' as date), @origin)
```

Messages

(No column name)	(No column name)	(No column name)	(No column name)
2022-01-01	2022-01-29	2022-01-29	2022-02-26

اگر نوع bucket را به ماه تغییر دهم و interval را به ۱ کاهش دهم، اولین روز هرماه را دریافت می‌کنم.

```
DECLARE @origin DATE = '2022/01/01';
DECLARE @bucketsize INT = 1

SELECT
 date_bucket(month, @bucketsize, CAST('2022/01/15' as date), @origin),
 date_bucket(month, @bucketsize, CAST('2022/01/30' as date), @origin),
 date_bucket(month, @bucketsize, CAST('2022/02/25' as date), @origin),
 date_bucket(month, @bucketsize, CAST('2022/03/04' as date), @origin)
```

Messages

(No column name)	(No column name)	(No column name)	(No column name)
2022-01-01	2022-01-01	2022-02-01	2022-03-01

GENERATE_SERIES

تابع GENERATE_SERIES جدولی از اعداد را به من برمی‌گرداند که می‌توان از آن در جای مناسب استفاده نمود. سینتکس بسیار ساده‌ای دارد و توسعه خوبی روی آن انجام گرفته است. سینتکس به صورت زیر است:

GENERATE_SERIES (start, stop [, step])

اگر start < stop باشد به صورت پیش‌فرض مقدار step برابر ۱ خواهد بود و در غیراینصورت مقدار step برابر ۱- تنظیم می‌شود. دقت داشته باشید مقدار step هرگز نمی‌تواند برابر صفر باشد.

اگر لیستی از اعداد ۱ تا ۵ را بخواهم، کد زیر را می توانم اجرا کنم:

```
SELECT * FROM GENERATE_SERIES(1, 5)
```

Messages

value
1
2
3
4
5

نکته جالب در مورد این تابع این است که می‌توانید از اعداد اعشاری نیز استفاده کنید. فرض کنید می‌خواهم کاری را با اعداد اعشاری برای درصد انجام دهم. کد زیر لیستی از درصدهای ۱ تا ۱۰۰ را برمی‌گرداند.

```
SELECT * FROM GENERATE_SERIES(0.0, 1.0, 0.01)
```

Messages

value
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
0.11

نام ستون value است و می‌توانید از آن به همان شیوه جدول tally استفاده کنید.

GREATEST/LEAST

این دو تابع برای من توابع جالبی هستند که باعث می‌شوند کد بسیار تمیزتر باشد. خوشحالم که GREATEST() و LEAST()به پلتفرم اضافه شدند. این توابع تعدادی از پارامترها را باهم مقایسه کرده و بزرگترین یا کوچکترین آن‌ها را برمی‌گرداند.

چند مثال برای کاربردهای این دو تابع ارائه می‌کنم:

```
SELECT GREATEST(1, 2, 4.3, 19),
       LEAST(1, 2, 4.3, 19);
```

sults Messages

(No column name)	(No column name)
19.0	1.0

مشاهده می‌کنید که بزرگترین و کوچکترین اعداد توسط این توابع برگردانده می‌شوند. اما یک تبدیل datatype نیز اتفاق افتاده است. این روال براساس اولویت datatype است.

این توابع برای string ها هم کار می‌کند:

```
SELECT GREATEST('B','A','C','Z','F'),
       LEAST('20220501', '20221012','20220914');
```

sults Messages

(No column name)	(No column name)
Z	20220501

مقادیر موجود در تابع LEAST() به صورت string هستند نه. date ما می‌توانیم از date نیز در این توابع استفاده کنیم، اما باید پارامترهای ارسالی از نوع date باشند. همه مقادیر ارسال شده باید ازطریق یک implicit conversion ارسال شوند.

نکته کاربردی این توابع برای date این است که NULL ها را نادیده می‌گیرند. بنابراین اگر تعدادی تاریخ برای سفارش داشته باشم، مانند تاریخ‌های سفارش، بسته‌بندی، ارسال و تحویل، می‌توانم آخرین تاریخ را به راحتی دریافت کنم:

```
DECLARE @ORDERDATE DATE = '20220301';
DECLARE @BACKORDERDATE DATE = null
DECLARE @PACKDATE  DATE = '20220305'
DECLARE @SHIPDATE DATE = '20220310'
DECLARE @DELIVERYDATE DATE = null

SELECT GREATEST(@ORDERDATE,
                @BACKORDERDATE,
                @PACKDATE,
                @SHIPDATE,
                @DELIVERYDATE
                )
```

esults Messages

(No column name)
2022-03-10

در اینجا من تاریخ‌های تابع را طوری تنظیم کرده‌ام که معمولاً به‌روزرسانی می‌شوند اما نیازی نیست نگران مقادیر NULL برای سفارش برگشتی و تاریخ‌های تحویل باشم. این تابع به من آخرین تاریخ را می‌دهد.

STRING_SPLIT

تابع STRING_SPLIT در چند ورژن قبلی SQL Server بوده است اما یکی از دردسرها این بود که وقتی یک رشته را تقسیم می‌کنید، هیچ امکانی برای مرتب‌سازی ندارید. این روال در SQL Server 2022 تغییر کرده است. یک پارامتر سوم اختیاری، اضافه شده که می‌توانید آن را با عدد ۱ تنظیم کنید تا مقدار ترتیبی را تابع برگرداند.

```
DECLARE @S VARCHAR(20) = 'apples,pears,bananas';
SELECT * FROM STRING_SPLIT(@S,',',1) AS ss
```

ults | Messages

value	ordinal
apples	1
pears	2
bananas	3

ستون "value" برای رشته‌های تقسیم شده و ستون "ordinal" برای ترتیب قرارگیری است.

DATETRUNC

من اغلب در بسیاری از مواقع به تابع DATETRUNC نیاز ندارم، اما به هرحال یک تابع کاربردی است. این تابع هنگامی مورد نیاز است که می‌خواهید از مقدار datetime تنها عدد مربوط به ماه را دریافت کنید و یا عدد مربوط به روز را دریافت کنید و برای بقیه مقادیر هم به همین ترتیب. یعنی قسمت‌های مختلف یک datetime را جداگانه کنترل کنید. به عنوان مثال می‌توانم روز را از تاریخ حذف کنم با مقدار ۰۱ جایگزین کنم و سال و ماه را نگه دارم. یا حتی روز و ماه را از تاریخ حذف کنم و با مقدار ۰۱ جایگزین کنم و تنها سال را نگه دارم. در مورد دقیقه، ثانیه و غیره نیز می‌توانید این کنترل را داشته باشید.

مثال زیر حالت‌های مختلف را نشان می‌دهد.

```
DECLARE @d DATETIME = '20220704 16:24:32.1'
SELECT
    DATETRUNC(year, @d),
    DATETRUNC(quarter, @d)
UNION
SELECT
    DATETRUNC(month, @d),
    DATETRUNC(day, @d)
UNION
SELECT
    DATETRUNC(week, @d),
    DATETRUNC(hour, @d)
UNION
SELECT
    DATETRUNC(minute, @d),
    DATETRUNC(second, @d)
```

Results | Messages

(No column name)	(No column name)
2022-01-01 00:00:00.000	2022-07-01 00:00:00.000
2022-07-01 00:00:00.000	2022-07-04 00:00:00.000
2022-07-03 00:00:00.000	2022-07-04 16:00:00.000
2022-07-04 16:24:00.000	2022-07-04 16:24:32.000

خلاصه

این‌ها تعدادی از تغییرات T-SQL برای SQL Server 2022 بودند. از GENERATE_SERIES استقبال می‌کنم؛ زیرا من از جداول tally استفاده می‌کنم و این تابع می‌تواند جایگزین خوبی باشد. من مطمئن هستم که استفاده از STRING_SPLITنیز برای مرتب‌سازی رشته‌های خروجی کاربرد دارد.

DATE_BUCKET جالب به نظر می‌رسد، به خصوص برای محاسبات دوره‌های زمانی مختلف مفید خواهد بود. توانایی تعیین دوره‌های زمانی مختلف به گزارش‌های مالی کمک می‌کند که گاهی اوقات از BUCKET های غیرمعمول استفاده کنند. من در مورد DATETRUNC مطمئن نیستم خیلی به دردم بخورد.

DISTINCT FROM و GREATEST/LEAST احتمالاً توابعی هستند که من بیشتر از بقیه توابع ارائه شده در این مقاله استفاده خواهم کرد. من دورنمای خوبی را با این تغییرات جدید می‌بینم که با این پیشرفت‌ها کدنویسی ساده‌تر و تمیزتر شود.