MS SQL 2011 (Denali) – Sequence

Sequence (NEW) - последовательность

Возможность которой не удивишь нынче пользователей Oracle, DB2, PostgreSQL и множества других реляционных баз данных, наконец-то появилась и в SQL Server. На арене Sequence!

Sequence – генерирует последовательность чисел так же как и identity. Однако основным плюсом sequence является то, что последовательность не зависит от какой-либо конкретной таблицы и является объектом базы данных.

Рассмотрим пример скрипта написанного на SQL Server 2008. Создание простой таблицы с двумя колонками, одна из которых будет авто инкрементальной.

```
Create Table WithOutSequence1
(
        EmpId int identity not null primary key
        ,EmpName varchar(50) not null
)

Insert into WithOutSequence1
Select 'Violet' Union All
Select 'Tape'

Select * from WithOutSequence1
```

Похожим образом создадим еще одну таблицу.

```
Create Table WithOutSequence2
(
        EmpId int identity not null primary key
        ,EmpName varchar(50) not null
)

Insert into WithOutSequence2
Select 'Violet' Union All
Select 'Tape'

Select * from WithOutSequence2
```

Как можно заметить из примеров, мы записали значения в таблицу при этом значение инкрементального поля автоматически и независимо от нас заполнилось. Мы не можем повторно использовать значение этого поля в другой таблице. Давайте посмотрим какой выход дает Sequence из этой ситуации.

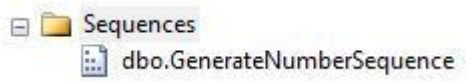
Общий синтаксис для команды выглядит так:

```
CREATE SEQUENCE [schema_name . ] sequence_name
  [ AS { built_in_integer_type | user-defined_integer_type } ]
  | START WITH <constant>
  | INCREMENT BY <constant>
  | { MINVALUE <constant> | NO MINVALUE }
  | { MAXVALUE <constant> | NO MAXVALUE }
  | { CYCLE | NO CYCLE }
  | { CACHE [<constant> ] | NO CACHE }
```

Создадим последовательность чисел:

```
IF EXISTS (SELECT * FROM sys.sequences WHERE NAME = N'GenerateNumberSequence' AND
TYPE='SO')
    DROP Sequence GenerateNumberSequence
GO
SET ANSI_NULLS ON
GO
CREATE SEQUENCE GenerateNumberSequence
START WITH 1
INCREMENT BY 1;
GO
```

После выполнения указанного скрипта, в браузере объектов базы, в узле Sequences можно найти наш объект.



После того как объект создан, можно его использовать в создании и заполнении таблиц как показано ниже:

```
Create Table WithSequence1
(
EmpId int not null primary key
,EmpName varchar(50) not null
);

Insert into WithSequence1(EmpId, EmpName)
VALUES (NEXT VALUE FOR GenerateNumberSequence, 'Violet'),
(NEXT VALUE FOR GenerateNumberSequence, 'Tape')

SELECT * FROM WithSequence1;
```

Если создать вторую таблицу в таком же духе, то можно снова использовать **GenerateNumberSequence** и получать сквозную нумерацию объектов.

```
Create Table WithSequence2
(
EmpId int not null primary key
,EmpName varchar(50) not null
);

Insert into WithSequence2(EmpId, EmpName)
VALUES (NEXT VALUE FOR GenerateNumberSequence, 'Niladri'),
(NEXT VALUE FOR GenerateNumberSequence, 'Deepak')

SELECT * FROM WithSequence2;
```

Последовательность (Sequence) которую мы создали, можно посмотреть в системном каталоге **sys.sequences**.

```
SELECT
        Name
        ,Object_ID
        ,Type
        ,Type_Desc
        ,Start_Value
        ,Increment
        ,Minimum_Value
        ,Maximum_Value
```

```
        ,Current_Value
        ,Is_Exhausted
FROM sys.sequences
```

	Name	Object_ID	Type	Type_Desc	Start_Value	Increment	Minimum_Value	Maximum_Value	Current_Value	Is_Exhausted
1	Sequence-20110205-163323	946102411	SO	SEQUENCE_OBJECT	-2147483648	1	-2147483648	2147483647	-2147483648	0
2	GenerateNumberSequence	1419152101	SO	SEQUENCE_OBJECT	1	1	-2147483648	2147483647	1	0

Это не вся доступная информация по sequence, просто эти колонки нам понадобятся далее. Чтобы получить всю информацию замените имена колонок на звездочку. =) Про Is_Exhausted будет упомянуто позднее.

Sequence может быть следующих типов:

- Int
- Smallint
- Tinyint
- Bigint
- Decimal
- Numeric

Не обязательно начинать последовательность с единицы. Можно начинать с любого числа в пределах возможных значений объявленного типа. Например, для целочисленных значений это может быть от -2147483648 до 2147483647.

Проверим на практике, что скажет SQL Server при задании начального числа вне допустимого диапазона. Начнем с левой границы.

```
CREATE SEQUENCE GenerateNumberSequence
START WITH -2147483649 --outside the range of the int datatype boundary
INCREMENT BY 1;
```

An invalid value was specified for argument 'START WITH' for the given data type.

Что и ожидалось. Теперь нарушим правую границу.

```
CREATE SEQUENCE GenerateNumberSequence
START WITH 2147483647 --the max range of the int datatype
INCREMENT BY 1;
```

Сервер сообщит нам об ошибке так:

The sequence object 'GenerateNumberSequence' cache size is greater than the number of available values; the cache size has been automatically set to accommodate the remaining sequence values.

И если мы обратим внимание на колонку Is_Exhausted в каталоге sys.sequences, то увидим, что значение стало равно 1. Что говорит нам о невозможности дальнейшего использования данной последовательности.

	Name	Object_ID	Type	Type_Desc	Start_Value	Increment	Minimum_Value	Maximum_Value	Current_Value	Is_Exhausted
1	GenerateNumberSequence	1531152500	SO	SEQUENCE_OBJECT	2147483647	1	-2147483648	2147483647	2147483647	1

При попытке создать таблицу с использованием такой последовательности, сервер выдаст ошибку:

The sequence object 'GenerateNumberSequence' has reached its minimum or maximum value. Restart the sequence object to allow new values to be generated.

Это можно трактовать как просьбу движка рестартовать указанную последовательность. Для этого необходимо воспользоваться конструкцией **RESTART WITH**.

```
ALTER SEQUENCE dbo.GenerateNumberSequence
RESTART WITH 1;
```

Значение должно быть в пределах допустимого диапазона объявленного типа. Далее последовательность начнется с указанного значения, не со следующего.

Т.е. если задать

```
ALTER SEQUENCE dbo.GenerateNumberSequence
RESTART WITH 10;
```

А потом выполнить скрипт:

```
Insert into WithSequence1(EmpId, EmpName)
VALUES (NEXT VALUE FOR GenerateNumberSequence, 'violet'),
(NEXT VALUE FOR GenerateNumberSequence, 'tape')

SELECT * FROM WithSequence1;
```

То результат будет таким:

```
EmpId    EmpName
-----    -------
10       violet
11       tape
```

Последовательность началась с заданного значения.

Получить минимальные и максимальные значения можно из каталога **sys.sequences**.

MIN и MAX значения

Для последовательностей можно задавать границы допустимых значений. Попробуем выполнить такой скрипт ниже.

```
CREATE SEQUENCE GenerateNumberSequence
START WITH 1
INCREMENT BY 1
MINVALUE 10
MAXVALUE 20
```

Минимальное значение равняется 10, максимальное – 20, но мы пытаемся задать начальное значение равное единице. Это за пределами допустимого диапазона и поэтому нас порадуют сообщением:

The start value for sequence object 'GenerateNumberSequence' must be between the minimum and maximum value of the sequence object.

Далее можем представить, что следующее значение в последовательности нарушает границу. В таком случае получим ошибку:

The sequence object 'GenerateNumberSequence' has reached its minimum or maximum value. Restart the sequence object to allow new values to be generated.

Для решения проблемы есть два пути:

- Использовать служебные слова Restart или Restart With.
- Использовать опцию CYCLE

Опция CYCLE

Данная опция зацикливает последовательность и, достигнув максимального значения, последовательность продолжается с минимального. Например:

```
CREATE SEQUENCE GenerateNumberSequence
START WITH 20
INCREMENT BY 1
MINVALUE 10
MAXVALUE 20
CYCLE
```

После того как максимальное значение было достигнуто, результаты станут такими:

```
EmpId    EmpName
-----    -------
10       Tape
20       Violet
```

Для выборки использовался запрос:

```
Insert into WithSequence1(EmpId, EmpName)
VALUES (NEXT VALUE FOR GenerateNumberSequence, 'Violet'),
(NEXT VALUE FOR GenerateNumberSequence, 'Tape')

SELECT * FROM WithSequence1;
```

Если внимательно посмотреть на вывод, то можно заметить, что записи были перепутаны. Если бы мы не использовали последовательности, то вывод был бы

```
EmpId    EmpName
-----    -------
20       Violet
21       Tape
```

Но из-за того, что вторая запись пересекла диапазон допустим значений, номер был сброшен на минимальное значение, заданное для последовательности (10). Если сейчас посмотреть в каталог sys.sequences, то будет видно, что текущее значение равняется 10.

В следующий раз, заполнение таблицы могло бы быть таким:

```
EmpId    EmpName
----     -------
11       Violet
12       Tape
```

В этот момент Sequence проверит порядок в котором записи будут вставлены и так как “Violet” идет раньше “Tape” и текущий номер равен 10, записи будут вставлены как:

Следующее_занчение =Текущее_значение +Сдвиг т.е. 10 +1 будет присвоено для “Violet”. Теперь значение Sequence = 11 и для второй записи значение будет 12 следуя то же самой формуле.

Опция NO CYCLE

Поведение такой опции уже рассматривалось в самом начале, и является значением по умолчанию при создании Sequence.

Sequence в сочетании с Over()

Можно использовать последовательность вместе с выражением Over для генерирования порядковых номеров как показано ниже:

```
--Declare a table
Declare @tblEmp Table
(
EmpId int identity
,EmpName varchar(50) not null
)
--Populate some records
Insert Into @tblEmp
Select 'Niladri' Union All
Select 'Arina' Union All
Select 'Deepak' Union All
Select 'Debasis' Union All
Select 'Sachin' Union All
Select 'Gaurav' Union All
Select 'Rahul' Union All
Select 'Jacob' Union All
Select 'Williams' Union All
Select 'Henry'

--Fire a query
SELECT
        e.*
        , Seq = NEXT VALUE FOR GenerateNumberSequence OVER (ORDER BY EmpName)
FROM @tblEmp e
```

Результат:

	EmpId	EmpName	Seq
1	2	Arina	1
2	4	Debasis	2
3	3	Deepak	3
4	6	Gaurav	4
5	10	Henry	5
6	8	Jacob	6
7	1	Niladri	7
8	7	Rahul	8
9	5	Sachin	9
10	9	Williams	10

Можно заметить, что записи были отсортированы и последовательность была применена верно к сохраненным данным. Это означает, что записи сначала сортируются и только потом применяется нумерация последовательности.

Ограничения использования Next Value для функций.

Sequence ни в каких случаях нельзя использовать в сочетании с:

- Проверкой ограничений (constraints)
- Значениями по умолчанию
- Вычисляемыми колонками
- Представлениями (views)
- Пользовательскими функциями
- Пользовательскими функциями агрегации
- Подзапросами
- CTE (Common Table Expression)
- Подтаблицами
- Выражением TOP
- Выражением Over
- Выражением Output
- Выражением On
- Выражением Where
- Выражением Group By
- Выражением Having
- Выражением Order By
- Выражением Compute
- Выражением Compute By

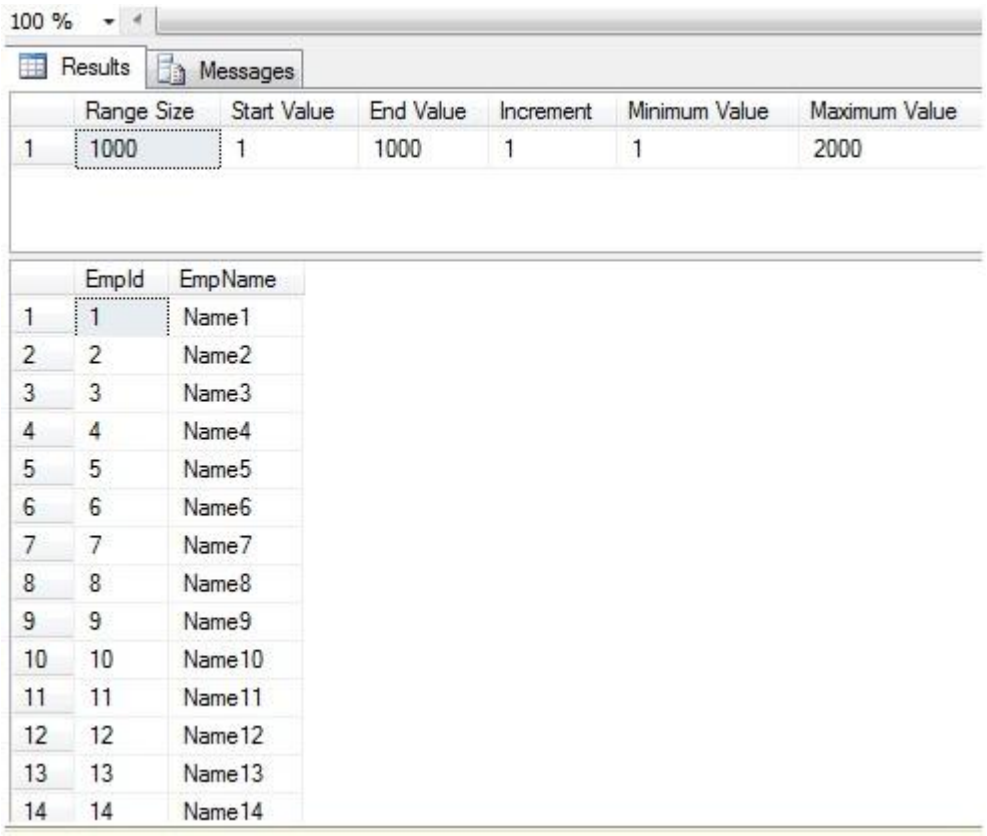
Функция sp_sequence_get_range

Если рассмотреть все использованные выше подходы к добавлению строк в таблицы используя **NEXT VALUE FOR**, то становится заметно, что это выражение присутствует в каждом уровне VALUES, что выглядит несколько утомительно. Вместо этого можно использовать функцию sp_sequence_get_range для получения необходимого диапазона значений, которые можно использовать впоследствии. Сейчас продемонстрирую как это можно осуществить.

```
--Drop the Sequence object if it exists
-- удаляем последовательность, если она существует
IF EXISTS (SELECT * FROM sys.sequences WHERE NAME = N'GenerateRangeNumberSequence' AND
TYPE='SO')
    DROP Sequence GenerateRangeNumberSequence
GO

-- Drop the table if it exists
-- удаляем таблицу, если она существует
IF EXISTS (SELECT * FROM sys.objects WHERE name = N'tbl_RangeSequence' AND type = 'U')
    DROP TABLE tbl_RangeSequence
GO
SET ANSI_NULLS ON
GO

--Create the sequence
-- создаем последовательность
CREATE SEQUENCE GenerateRangeNumberSequence
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 2000
CYCLE
GO

-- создаем таблицу
```

```
CREATE TABLE [dbo].[tbl_RangeSequence](
        [EmpId] [int] NOT NULL,
        [EmpName] [varchar](50) NOT NULL,
PRIMARY KEY CLUSTERED
(
        [EmpId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

--Declare the needed parameter for the sp_sequence_get_range
-- объявляем необходимые параметры для процедуры sp_sequence_get_range
DECLARE
  @sequence_name nvarchar(100) = N'GenerateRangeNumberSequence',
  @range_size int = 1000,
  @range_first_value sql_variant,
  @range_last_value sql_variant,
  @sequence_increment sql_variant,
  @sequence_min_value sql_variant,
  @sequence_max_value sql_variant;

--Execute the stored procedure sp_sequence_get_range
-- запускаем процедуру sp_sequence_get_range
EXEC sp_sequence_get_range
  @sequence_name = @sequence_name,
  @range_size = @range_size,
  @range_first_value = @range_first_value OUTPUT,
  @range_last_value = @range_last_value OUTPUT,
  @sequence_increment = @sequence_increment OUTPUT,
  @sequence_min_value = @sequence_min_value OUTPUT,
  @sequence_max_value = @sequence_max_value OUTPUT;

-- Display the values
-- показываем значения
SELECT
  @range_size AS [Range Size],
  @range_first_value AS [Start Value],
  @range_last_value AS [End Value],
  @sequence_increment AS [Increment],
  @sequence_min_value AS [Minimum Value],
  @sequence_max_value AS [Maximum Value];

--Build the range of values in the CTE
-- строим массив значений с помощью CTE
;With Cte As
(
Select Rn = 1, SeqValue = Cast(@range_first_value as int)
Union All
Select Rn+1, Cast(SeqValue as int) + Cast( @sequence_increment as int)
From Cte
Where Rn<@range_last_value
)

--Insert 100 Records
-- вставляем 100 строк
Insert into tbl_RangeSequence(EmpId, EmpName)
Select SeqValue,'Name' + Cast(SeqValue as varchar(3))
From Cte
Where SeqValue<=100
Option (MaxRecursion 0)

----Display the result
-- показываем результат
SELECT * FROM tbl_RangeSequence
```

Вот что будет в результате выполнения:



100 %

Results | Messages

	Range Size	Start Value	End Value	Increment	Minimum Value	Maximum Value
1	1000	1	1000	1	1	2000

	EmpId	EmpName
1	1	Name1
2	2	Name2
3	3	Name3
4	4	Name4
5	5	Name5
6	6	Name6
7	7	Name7
8	8	Name8
9	9	Name9
10	10	Name10
11	11	Name11
12	12	Name12
13	13	Name13
14	14	Name14

Здесь можно увидеть, что последовательность была увеличена до 1000 и пропущенные значения не были использованы нигде без нашего ведома. В данном случае мы их использовали для вставки значений.

Сравнение между Sequence и Identity

Не стоит ставить между ними глобальный знак равенства из-за следующих факторов:

- Identity относится к таблице и является ее частью неотделимой, Sequence – независимый объект базы данных.
- Можно получить набор последовательности с помощью sp_sequence_get_range, что в принципе невозможно с Identity.
- Для Sequence можно определять границы значений, что так же невозможно для Identity.
- Цикличность значений можно задать так же только для Sequence.

И еще несколько слов про Sequence.

- Sequence дает больший прирост производительности по сравнению с Identity. Сравнение и результаты в статье Аарона Бертарнда (Aaron Bertrand)
- Можно задавать права доступа к Sequence, так же как и к другим объектам базы.

Почитать дополнительно о Sequence можно на MSDN:

1. CREATE SEQUENCE
2. Creating and Using Sequence Numbers
3. sp sequence get range

Дальше будет интереснее, так что оставайтесь на связи. Перевод.

Hard’n’heavy!

Violet Tape