한 지붕 두 가족『MS SQL Server 2012』Identity와 Sequence

㈜엑셈 컨설팅본부/SQL Server팀 이 제춘

1992 년 MS 가 Windows NT 에서 운용되는 첫 번째 SQL Server(4.2) 를[2] 출시한 이후 20 년이 흘렀다. 그간 꾸준한 발전을 통해 후발주자임에도 Oracle 과 함께 관계형 DBMS 시장의 양대 산맥으로 자리매김 하였다. 그리고 지난 2012 년 또 하나의 새로운 버전 SQL Server 2012 를 출시 하였다.

이 SQL Server 2012 에서는 고가용성 향상을 위한 Always on 기능 및 BI 의 Power View, Cloud 환경 지원 등 굵직한 새 기능들이 포함되었다. 이러한 기능들 이외에도 프로그래밍 기능 향상과 같은 MS 의 세심함을 느낄 수 있는 부분들이 있는데 그 중 하나가 Sequence 이다. 사실 Sequence 는 지금까지 Oracle 에서 사용되어 왔던 자동증가 기능이었다. 이전부터 SQL Server 와 Oracle 의 다른 점으로 거론된 부분 중 하나가 자동증가 컨트롤에 대해 SQL Server 는 Identity 를 Oracle 에서는 Sequence 를 사용한다는 것이었다. 헌데 이 두 가지 모두를 SQL Server 2012 부터 사용 가능하게 된 것이다.

자동증가 기능이란 1, 2, 3, 4…혹은 2, 4, 6, 8…등과 같이 정해진 시작 값에서 일정한 값 만큼 자동으로 증가하는 기능이다. 자동으로 값이 증가 되므로 중복되는 값이 없어 테이블을 운영하며 Primary key 를 적용 할 수 있는 일련번호 등으로 많이 사용 된다.

SQL Server 2012 라는 한 지붕에 살게 된 Identity 와 Sequence 에 대해 알아보도록 하자.

---

[2] Microsoft 는 Sybase 와 1987 년부터 협력을 맺었고 1989 년 OS/2 기반 SQL Server 1.0 을 발표한다. 1992 년 3 월 OS/2 용 SQL Server 4.2 를 발표했는데 이 버전에 있는 데이터베이스 엔진의 소스 코드가 MS 가 Sybase 로부터 받은 마지막 코드이다. 발표 직후 양사는 개발팀을 분업하였고 1992 년 10 월 MS 개발팀에서 개발한 Windows NT 용 SQL Server 4.2 를 출시한다. 1994 년 4 월 Microsoft 와 Sybase 는 협력관계를 마감한다. – Kalen Delaney 저, Microsoft Inside SQL Server 2000 발췌

안방마님 Identity

Identity 는 이전부터 SQL Server 에서 사용한 자동증가 방식이다. 선언하는 방법과 사용은 아래와 같다.

```
CREATE TABLE Identity_Test
(
        Idt INT IDENTITY(1,1),
        Val VARCHAR(5)
)
GO

INSERT INTO Identity_Test(Val) VALUES ('test')
GO 3

SELECT * FROM Identity_Test
GO
Idt          Val
-----------------------------------------------------------
1           test
2           test
3           test
```

테이블을 생성해주면서 사용하고자 하는 칼럼의 자료형 속성 뒤에 IDENTITY(초기 값, 증가 값)를 추가해주기만 하면 끝이 난다. 굉장히 심플하다. 예시 코드에서 IDENTITY(1,1)이므로 초기 값 1 부터 1 씩 증가 된다. 예시에서 INSERT 구문을 통해 두 개의 데이터를 삽입하였고 삽입된 모습을 보면 IDENTITY 속성이 있는 Idt 칼럼이 자동으로 증가하여 삽입되었음을 확인 할 수 있다. 주의 할 점은 하나의 테이블에는 반드시 하나의 IDENTITY 속성만을 사용 할 수 있다는 것이다. 또한 기본적으로 INSERT 작업 시 IDENTITY 속성이 있는 칼럼은 기재하지 말아야 한다.

드물게 IDENTITY 속성이 적용되어 있는 칼럼의 값을 직접 입력해야 하는 경우가 있는데

```
SET IDENTITY_INSERT 테이블명 ON
```

위 명령어를 통해 IDENTITY 속성이 적용된 칼럼에도 직접 값을 입력 할 수 있다. 이 때 제약이 설정 되어있지 않은 경우 이미 기록된 값도 다시 입력 될 수 있는데 이 같은 상황에서는 중복 값이 없다는 부분은 거짓말이 된다. 통상적으로 이렇게 사용되지 않기 때문에 하는 말이다. IDENTITY_INSERT 속성을 OFF 시키면 다시 자동증가 기능이 실행된다. 이 때 IDENTITY 값은

마지막 입력된 값이 아닌 해당 칼럼의 MAX 값으로 설정된다. (증가값이 음수일 경우는 MIN 값으로 설정) 자동증가 기능이 활성화 된 상황에서 다시 입력이 될 경우 [해당 칼럼의 MAX 값 + IDENTITY 증가값]이 삽입 된다. 필요한 경우 이 IDENTITY 값도 변경이 가능한데 이는 DBCC CHECKIDENT 구문을 사용하여 확인 및 변경이 가능하다.

```
DBCC CHECKIDENT (
      table_name
                [, { NORESEED | { RESEED, new_reseed_value]}}]
)
```

다음 구문을 통해 IDENTITY 값을 자동으로 칼럼의 MAX 값으로 변경시켜 주거나 현재 IDENTITY 값을 확인, 또는 직접 IDENTITY 값을 지정해 줄 수 있다.

뒤쪽에서 언급한 IDENTITY 값을 컨트롤하는 내용들은 사실 자주 사용되는 기능은 아니다. 기본적으로 사용되는 구문만 놓고 본다면 IDENTITY 속성은 참으로 간단하고도 편리한 자동증가 기능이라 볼 수 있다.

굴러온 돌 Sequence

SQL Server 2012 에서 새롭게 추가 된 Sequence 는 테이블 밖에서 선언된다. 위에서 Identity 를 설명했을 때와 같은 결과가 나오도록 하나의 Sequence 와 함께 연동 될 테이블을 만들어보자.

```
CREATE TABLE Sequence_Test
(
       Seq INT,
       Val VARCHAR(5)
)
GO
CREATE SEQUENCE SeqTest
       START WITH 1
       INCREMENT BY 1 ;
GO

INSERT INTO Sequence_Test(Seq, Val)
```

```
        VALUES (NEXT VALUE FOR SeqTest, 'test')
GO 3


SELECT * FROM Sequence_Test
GO
Seq         Val
----------- ----------
1           test
2           test
3           test
```

예시에서 보다시피 Sequence 는 테이블과 별도로 생성 된다. 그러므로 하나의 테이블에 여러 개의 Sequence 가 들어갈 수 있고 하나의 Sequence 를 여러 개의 테이블에서 사용 할 수도 있다. START WITH 문에서 초기값을 정해주고 INCREMENT BY 문을 통해 증가 값을 세트 할 수 있다. INSERT 구문을 보면 알겠지만 NEXT VALUE FOR 문을 호출 할 때마다 Sequence 가 증가 된다.

기본적으로 Sequence 는 하나의 개체이고 여러 인수를 갖고 있다. 간략한 설명을 위해 MSDN 에서 제공하는 Sequence 기본 구문을 인용 하였다.

```
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 } ]
        [ ; ]
```

CREATE SEQUENCE 기본구문을 보면 Sequence 의 자료 형식을 지정해 줄 수 있고 최소값, 최대값 지정과 함께 최대값을 초과할 경우 반복 여부도 선택 가능하다. 해당 기능들과 함께 SQL Server 에서 기본 50 으로 지정되는 CACHE 의 크기도 지정 할 수 있다. 캐시는 Sequencer 를 생성하는 데 필요한 디스크 IO 를 최소화 시켜 SQL 의 성능을 향상시킨다. 그러나 전원오류와 같은 비정상적 종료가 발생 할 경우 캐시에 미리 저장했던 값들은 사라지고 시스템 테이블에 있는 다음 번호가 Sequence 번호로 할당 된다. 예를 들어 캐시 10 을 설정 할 경우 1, 2, 3 이 입력된 후 비정상적으로 중지 되면 캐시에 할당 된 시퀀스 번호는 모두 사라진다. 1, 2, 3 을 입력 받고 바로 11 부터 다시 Sequence 가 할당 된다. NO CACHE 옵션을 사용 할 경우 캐시 영역을

사용하지 않기 때문에 누락되는 번호가 발생하지 않지만 LOCK 으로 인해 성능 면에서 취약해 질 수 있다.

이 외에도 유용한 기능으로 값이 들어가 있는 기존 사용 테이블에도 OVER 절을 통해 특정 열을 정렬하여 Sequence 를 적용 할 수 있다. 아래는 SeqTest 로 선언 한 Sequence 를 ID 오름차순으로 하여 Sequence_test 테이블에 입력하는 예제 이다. 자주 사용되지는 않겠지만 운영중인 테이블에 기본 키가 없어 곤란한 상황에서는 두손들고 반길 만한 기능이다.

```
SELECT NEXT VALUE FOR SeqTest
OVER (ORDER BY ID) AS 'Seq', ID, Phone, Address
FROM Sequence_test
```

이같이 Sequence 는 사용자에게 높은 프로그래밍 자유도를 제공하고 있다. 여러 면에서 Sequence 는 Identity 에 비해 나아 보인다. 그렇다면 앞으로 Sequence 가 Identity 를 대체하고 Identity 는 사라지게 될 것인가? 앞서 Identity 와 Sequence 를 설명하며 많은 부분에 대해 이미 알았겠지만 직접적인 비교를 통해 더 알아보고자 한다.

Identity vs. Sequence ?

사실 이 둘을 동등한 입장에 두고 비교한다는 자체가 우둔한 일이 아닐 수 없다. 서로 비슷한 일을 할 뿐이지 근본적으로 다르기 때문이다. 대결 구도가 안 된다는 말이다. 그도 그럴 것이 Identity 는 테이블에 종속된 속성(Attribute)중 하나일 뿐이고 Sequence 는 하나의 개체(Object)이기 때문이다. 그래도 구태여 비교해 보이자면 아래와 같은 도표로 나타낼 수 있다.

<표1> Identity vs. Sequence

	Identity Attribute	Sequence Object
코드 작성	간단	복잡
명확성(명료성)	높음	낮음
프로그래밍 자유도	낮음	높음
옵션 기능	적음	많음

Identity 는 테이블에 종속되는 속성에 불과하기에 작성이 매우 간단하다. 하나의 테이블에 단 하나만 존재 할 수 있고 테이블 생성시 함께 속성을 설정 하기에 명확하게 나타내는 데에도 유용하다. Sequence 는 테이블과 별개로 사용이 가능한 개체이다. 그렇기에 테이블에 대해 개수 제약 없이 사용할 수 있고 여러 개의 테이블에서 하나의 Sequence 를 사용 할 수도 있다. 무엇보다도 강력한 것은 다양한 기능을 사용할 수 있는 프로그래밍 자유도 이다.

몇 가지 예를 들어 본다면 Identity 의 경우 INSERT 되기 전에는 입력 될 값을 미리 알고 사용할 수 없지만 Sequence 의 경우 INSERT 와 관계 없이 쿼리 상 어디서든 값을 증분 하여 사용할 수 있다. Sequence 는 복수의 테이블에 하나의 시퀀스로 채번 할 수 있고 이미 여러 데이터가 입력되어 있는 테이블에 대해서도 채번 할 수 있다. 문자와 조합된 조합문자 형식의 채번 또한 수월하게 할 수 있다.

'C000001' 과 같은 문자조합 채번을 Identity 와 Sequence 로 각 각 구현하여 비교해 봤다.

```
CREATE TABLE TableName(
        Idt INT IDENTITY not null PRIMARY KEY,
        Name VARCHAR(20)
)
GO
-- Identity 칼럼을 인수로 받아 Sequence 로 가공
CREATE FUNCTION FuncName(@id INT) RETURNS CHAR(7) AS
BEGIN
        RETURN 'C' + right('000000' + convert(VALCHAR(6), @id), 6)
END
GO
-- Sequence 로 가공 한 값을 넣어 줄 칼럼 테이블에 추가
ALTER TABLE TableName ADD SeqNumber AS dbo.FuncName(idt)
GO

INSERT INTO TableName (Name) VALUES ('name')
GO 3

SELECT * FROM TableName
GO
```

Idt	Name	SeqNumber
1	name	C000001

```
2           name                C000002
3           name                C000003
```

먼저 Identity 를 이용한 쿼리이다. 3 Function 을 이용한 방법으로 Identity 를 이용해 조합문자를 생성하는 여러 방법 중 하나이다. 테이블에서 조합문자를 위해 칼럼을 하나 더 사용하고 함수를 통해 재가공하고 있다. Trigger 혹은 채번 테이블을 통해 조합문자를 생성하는 방법도 있지만 다른 방법들도 마찬가지로 Identity 속성이 있는 칼럼에 대해 직접적인 변경작업을 할 수 없어 참조해야만 한다. 다음은 Sequence 를 사용해 문자조합 채번을 한 쿼리이다.

```
CREATE TABLE Sequence_Test
(
       CustomSeq varchar(10) not null PRIMARY KEY,
       Name varchar(20)
)
GO

CREATE SEQUENCE SeqTest
       START WITH 1
       INCREMENT BY 1
GO
-- INSERT 구문 내에서 문자조합을 만들어 바로 입력
INSERT INTO Sequence_Test(CustomSeq, Name)
       VALUES ('C' + right('000000' + CONVERT(VARCHAR(6), NEXT VALUE FOR
SeqTest),6), 'name')
GO 3

SELECT * FROM Sequence_Test
GO
CustomSeq   Name
C000001     name
C000002     name
C000003     name
```

[3] 원본 출처 : By Jeff Smith, http://www.sqlteam.com/article/custom-auto-generated-sequences-with-sql-server
해당 사이트에서 Identity 를 이용해 더 복잡한 Custom Sequence Number 를 생성하는 방법도 제공하고 있다.

구조를 자세히 보면 Sequence 를 소개하면서 만들어 본 쿼리와 동일하다. 단지 INSERT 부분만 이 다를 뿐이다. 쿼리의 길이로만 본다면 identity 와 크게 다를 바 없어 보이지만 채번을 위해 따로 칼럼을 사용하지 않았고 함수를 사용하지도 않았다. 쿼리 분석도 훨씬 직관적이다.

앞서 언급한 예들 이외에도 다양한 기능을 사용하거나 옵션을 두고자 할 때, 혹은 Identity 로 작성하기 불편했던 많은 상황을 Sequence 로 해결 할 수 있다.

넝쿨째 굴러들어온 복, Sequence

SQL Server 2012 의 New Feature, Sequence 는 굴러온 돌이 아닌 굴러들어온 복덩이다. 왜 이제서야 나타났는지 싶다. Identity 의 단점인 프로그래밍 취약점을 Sequence 를 통해 커버 할 수 있게 되었다. 지금까지 Identity 를 여러 방법으로 변형해서 사용하는 수고로움을 덜 수 있게 된 것이다. 그렇다고 Identity 를 버리고 모두 Sequence 를 사용 할 필요는 없다. 숫자로 된 일련번호 목적으로만 사용 할 경우 편리하고 명확한 Identity 를 사용하고 추가적인 기능이 필요하다면 Sequence 를 사용하면 된다. 닭 잡는데 소 잡는 칼을 쓸 필요 없지 않나. MS 에서 제공하는 가이드나 권장사항은 없지만 앞서 말한 용도에 맞게 Identity 와 Sequence 를 사용하는 것이 좋겠다.