Craig S. Mullins

Return to Home Page

December 2003 / January 2004

The Resource for Users of IBM zSeries & S/390 Systems

z/JOURNAL

zData Perspectives

by Craig S. Mullins

DB2 V8: Sequence Objects and Identity Columns

When designing DB2 databases a frequent request is for a column to contain sequentially generated numbers. Every new row added to the table requires a new value, one greater than the previous value, to be generated. These numbers might be used as a key or simply to differentiate data rows.

DB2 provides identity columns (V6 refresh) and sequence objects (V8) to meet this need. Without such features an application program can implement similar functionality, but usually not in a manner that can perform and scale properly. In this column we will learn a little bit about each of these methods of creating sequential values.

Identity Columns

An identity column is defined to a DB2 column using the IDENTITY parameter. A column thusly defined will cause DB2 to automatically generate a unique, sequential value for that column when a row is added to the table. For example, identity columns might be used to generate unique primary key values or a value that somewhat mimics Oracle's row number capability. When inserting data into a table that uses an identity column, the program or user will not provide a value for the identity column. Instead, DB2 automatically generates the appropriate value to be inserted.

Only one identity column can be defined per DB2 table. Additionally, the data type of the column must be SMALLINT, INTEGER, or DECIMAL with a zero scale (or a user-defined type based on one of those data types). You have control over the starting point for the generated sequential values, and the number by which the count is incremented. An example follows:

```
CREATE TABLE EXAMPLE
   (ID_COL INTEGER NOT NULL
           GENERATED ALWAYS AS IDENTITY
           START WITH 100   INCREMENT BY
10
   ...);
```

In this example, the identity column is named ID_COL. The first value stored in the column will be 100 and subsequent INSERTs will add 10 to the last value. So the identity column values generated will be 100, 110, 120, 130, and so on.

To retrieve the value of an identity column immediately after it is inserted you must use the IDENTITY_VAL_LOCAL() function. For example, run the following statement immediately after the INSERT statement that sets the identity value:

```
     VALUES IDENTITY_VAL_LOCAL() INTO
:IVAR;
```

The host variable IVAR will contain the value of the identity column. But this will only work after a singleton INSERT. You cannot use INSERT INTO

SELECT FROM or LOAD, if you need to rely on this function.

There are other problems with using identity columns, as well. Loading data into a table that automatically generates identity values is troublesome. Each LOAD will generate different identity values. If you can live with these caveats, then identity columns might be useful. However, in general, these "problems" make identity columns a very niche solution. Sequence objects are probably more useful than identity columns. But you have to wait for V8 to use them.

Sequence Objects

A sequence object is a separate structure that generates sequential numbers. It is not assigned to a single column like the identity property. A sequence object is created using the CREATE SEQUENCE statement.

When the SEQUENCE is created it can be used by applications to “grab” a next sequential value for use in a table. Sequence objects are ideal for generating sequential, unique numeric key values. A sequence can be accessed and incremented by many applications concurrently without the hot spots and performance degradation associated with other methods of generating sequential values.

Sequences are efficient and can be used by many users at the same time without causing performance problems. Multiple users can concurrently and efficiently access SEQUENCE objects because DB2 does not wait for a transaction to COMMIT before allowing the sequence to be incremented again by another transaction. An example creating a SEQUENCE object follows:

```
CREATE SEQUENCE ACTNO_SEQ
   AS SMALLINT
   START WITH 1  INCREMENT BY 1
   NOMAXVALUE    NOCYCLE
   CACHE 10;
```

This creates the SEQUENCE object named ACTNO_SEQ. Now it can be used to generate a new sequential value, for example

```
INSERT INTO DSN8810.ACT
   (ACTNO, ACTKWD, ACTDESC)
   VALUES (NEXT VALUE FOR ACTNO_SEQ,
‘TEST’, ‘Test activity’);
```

The NEXT VALUE FOR clause is known as a sequence expression. Coding the sequence expression causes DB2 to use the named SEQUENCE object to automatically generate the next value. You can use the PREVIOUS VALUE FOR sequence expression to request the previous value that was generated, too.

Like identity columns, sequence objects also have parameters to control the starting point for the generated values and the number by which the count is incremented. Additionally, you can specify how the SEQUENCE should handle running out of values when the maximum value is hit.

As you can tell by now, sequence objects are more flexible and generally useful than identity columns. Unlike sequence objects, identity columns must adhere to certain rigid requirements. An identity column is always defined on a single table and each table can have at most one identity column. Furthermore, when you create an identity column, the data type for that column must be numeric; not so for sequences. If you used a sequence object to generate a value you could put that generated value into a CHAR column, if you wish.

Summary

DB2 provides us with several options for generating sequential values for our tables. Identity columns and sequence objects make designing DB2 database and applications easier than ever before. Be sure to understand the unique advantages these features offer

From zJournal, December 2003 / January 2004.