SQL | SEQUENCES

SQL sequences specifies the properties of a sequence object while creating it. An object bound to a user-defined schema called a sequence produces a series of numerical values in accordance with the specification used to create it. The series of numerical values can be configured to restart (cycle) when it runs out and is generated in either ascending or descending order at a predetermined interval. Contrary to identity columns, sequences are not linked to particular tables. Applications use a sequence object to access the next value in the sequence. The application has control over how sequences and tables interact. A sequence object can be referred to by user applications, and the values can be coordinated across various rows and tables.

Let's suppose that sequence is a set of integers 1, 2, 3, ... that are generated and supported by some database systems to produce unique values on demands.

- A sequence is a user-defined schema-bound object that generates a series of numeric values.
- Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.
- The sequence of numeric values is generated in an ascending or descending order at defined intervals and can be configured to restart when it exceeds max_value.

Different Features of Sequences

1. A sequence is a database object that generates and produces integer values in sequential order.
2. It automatically generates the primary key and unique key values.
3. It may be in ascending or descending order.
4. It can be used for multiple tables.
5. Sequence numbers are stored and generated independently of tables.
6. It saves time by reducing application code.
7. It is used to generate unique integers.
8. It is used to create an auto number field.
9. Useful when you need to create a unique number to act as a primary key.

10. Oracle provides an object called a Sequence that can generate numeric values. The value generated can have maximum of 38 digits
11. Provide intervals between numbers.

Syntax:

CREATE SEQUENCE sequence_name

START WITH initial_value

INCREMENT BY increment_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

Following is the sequence query creating a sequence in ascending order.

Example 1:

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

The above query will create a sequence named *sequence_1*. The sequence will start from 1 and will be incremented by 1 having maximum value of 100. The sequence will repeat itself from the start value after exceeding 100.

Example 2: Following is the sequence query creating a sequence in descending order.

```
CREATE SEQUENCE sequence_2
start with 100
increment by -1
```

```
min value 1
max value 100
cycle;
```

The above query will create a sequence named *sequence_2*. The sequence will start from 100 and should be less than or equal to a maximum value and will be incremented by -1 having a minimum value of 1.

Example To Use Sequence:

Create a table named students with columns as id and name.

```
CREATE TABLE students
(
ID number(10),
NAME char(20)
);
```

Now insert values into a table

```
INSERT into students VALUES
(sequence_1.nextval,'Shubham');
INSERT into students VALUES
(sequence_1.nextval,'Aman');
```

where *sequence_1.nextval* will insert id's in the id column in a sequence as defined in sequence_1.

Output:

```
ID  | NAME
-------------
1   | Shubham
2   | Aman
```

Cache Management

To enhance performance, SQL Server employs cache management for sequence numbers. The CACHE argument determines the number of sequence numbers pre-allocated by the server.

For instance, let's consider a new sequence with a starting value of 1 and a cache size of 15. When the first number is required, values 1 through 15 are fetched from memory and made available. The last cached value (15) is written to the system tables on disk. As all 15 numbers are utilized, the next request (for number 16) triggers the allocation of a new cache. The latest cached value (30) is then stored in the system tables.

In the event of a SQL Server shutdown after using 22 numbers, the next intended sequence number (23) in memory replaces the previously stored number in the system tables. Upon restarting, when a sequence number is needed, the starting number (23) is read from the system tables. A cache of 15 numbers (23-38) is allocated to memory, and the next number outside the cache (39) is written to the system tables.

If an abnormal shutdown occurs, such as a power failure, the sequence restarts from the number read from the system tables (39). Any sequence numbers allocated to memory but not requested by users or applications are lost. This behavior can result in gaps, but it ensures that a single sequence object will never issue the same value twice unless it is defined as CYCLE or manually restarted.

The cache is managed in memory by tracking the current value (the last issued value) and the remaining values in the cache. Hence, the cache consumes memory equivalent to two instances of the sequence object's data type.

When the cache argument is set to NO CACHE, the current sequence value is written to the system tables every time a sequence is used. This approach may slow down performance due to increased disk access, but it reduces the likelihood of unintended gaps. However, gaps can still occur if numbers are requested using the NEXT VALUE FOR or sp_sequence_get_range functions but are either unused or used within uncommitted transactions.

This article is contributed by **ARSHPREET SINGH**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Jun, 2023

67

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between SQL and T-SQL