Identity Property vs Sequence Object

Feature	Identity Property	Sequence Object
Type	Tied to a specific table column	User-defined database object, not tied to any table
Shared across tables	Cannot be shared across multiple tables	Can be shared across multiple tables
Next value generation	Requires inserting a row into the table	Does not require a row insertion
Max value specification	Maximum value = column data type's max value	Can specify a `MAXVALUE`
Automatic restart	No option to restart automatically	`CYCLE` option allows automatic restart

---

Identity Property Example

An **Identity property** is tied to the table and auto-generates IDs upon row insertion.

Create a Table with Identity Property:

sql Copy code

```sql
CREATE TABLE Employees ( Id INT PRIMARY KEY IDENTITY(1,1), Name NVARCHAR(50), Gender NVARCHAR(10) );
```

Insert Data:

sql Copy code

```sql
INSERT INTO Employees VALUES ('Todd', 'Male');
```

Output:

Id	Name	Gender
1	Todd	Male

Sequence Object Example

A **Sequence object** is independent of any table and can be shared across multiple tables.

Step 1: Create Customers and Users Tables

sql Copy code

```sql
CREATE TABLE Customers ( Id INT PRIMARY KEY, Name NVARCHAR(50), Gender NVARCHAR(10) ); CREATE TABLE Users ( Id INT PRIMARY KEY, Name NVARCHAR(50), Gender NVARCHAR(10) );
```

Step 2: Create a Sequence Object

sql Copy code

```sql
CREATE SEQUENCE [dbo].[SequenceObject] AS INT START WITH 1 INCREMENT BY 1;
```

Step 3: Insert Data Using the Sequence Object

sql Copy code

```sql
INSERT INTO Customers VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Ben', 'Male'); INSERT INTO Customers VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Sara', 'Female'); INSERT INTO Users VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Tom', 'Male'); INSERT INTO Users VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Pam', 'Female'); INSERT INTO Users VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'David', 'Male');
```

Step 4: Query the Tables

```sql
SELECT * FROM Customers; SELECT * FROM Users;
```

Output:

Customers Table:

Id	Name	Gender
1	Ben	Male
2	Sara	Female

Users Table:

Id	Name	Gender
3	Tom	Male
4	Pam	Female
5	David	Male

CYCLE Option in Sequence Object

The **CYCLE** option allows the sequence to restart automatically from the `MINVALUE` after reaching the `MAXVALUE`.

Example: Creating a Sequence with the CYCLE Option

```sql
CREATE SEQUENCE [dbo].[SequenceObject] START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 5 CYCLE;
```

Step 1: Generate Values Beyond the Max Value

```sql
```

```
SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 1 SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 2 SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 3 SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 4 SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 5 SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 1 (restarts) SELECT NEXT VALUE FOR [dbo].[SequenceObject]; -- Output: 2
```

Explanation:

- When the sequence reaches the `MAXVALUE` (5), the **CYCLE** option causes it to restart from the `MINVALUE` (1).
- This is useful for cyclic numbering systems or scenarios where ID reuse is acceptable.

Output Sequence:

Call	Generated Value
1st Call	1
2nd Call	2
3rd Call	3
4th Call	4
5th Call	5
6th Call	1
7th Call	2

Key Takeaways

- **Identity Property** is simpler but tied to a table and lacks advanced features like sharing or cycling.
- **Sequence Object** is flexible, supports sharing across tables, allows `MAXVALUE` and `MINVALUE`, and can cycle automatically when values are exceeded.