Identity columns
- A new entry in Oracle® Database 12c -

Author: Corrado Piola, OCP

Last updated: September 2013

Introduction

Good news for software and database developers: Oracle® Database 12c introduced **Identity**, an auto-incremental (system-generated) column to be used - commonly - for surrogate Primary Keys. But not for Primary Keys only...

In the previous database versions (until 11g), you can implement an Identity by creating two additional objects: a Sequence and a Trigger. You can avoid the Trigger, if you want, but you have to manually invoke the NEXTVAL method of your Sequence object.
Now (from 12c onward) you can create your own Table and - as a part of its structure - define the column that has to be generated as an Identity. No more and no less. And there is an intermediate solution: in 12c you can manually create your own Sequence, and then use it as a default value for your Table's column.
It's interesting to note that Oracle introduced Identity column in this last release, almost at the same time in which Microsoft® SQL Server® 2012 introduced Sequence objects.

This document doesn't want to be complete or exhaustive, but it shows you how to implement this new feature. My name is neither Cary Millsap nor Jonathan Lewis so, please, forgive (and forget, as soon as possible!) any possible mistake that I could have made over the pages which follow.
Please, feel free to contact me at my e-mail address (at the end of the document).

Should you need more details about the Identity column, you can find almost everything in Oracle® database documentation and in other sources over the internet (Tim Hall, for example, has written really good articles on this topic, better than the one you are reading now).
I have written nothing new, obviously, but I hope you will find this document an interesting reading.

I will locally connect to an Oracle® 12c R1 database server, and I will use two different users: most of the time I will use my user CONRAD (with all the privileges to create the needed objects, and with unlimited quotas on my default Tablespace CONRAD_TAB, a Locally Managed Tablespace with Automatic Segment Space Management) and, whenever a privileged user is required, I will use the user SYS. I have not constrained my Tables with Primary Keys, because it's not necessary for my test purposes.
If you need to create your test user from scratch, please refer to **Appendix A** at the end of this document. **Appendix B** shows you how to clean your test environment.

Please, be careful: don't make your tests in a production environment, especially when connected with SYSDBA privileges, as you could seriously damage your own data.
In the following examples, when needed, I have formatted the output for better readability.

Now I will show you two pre-12c solutions and two 12c ones.

Pre-12c solutions

For our test cases we need the following objects: a Table, a Sequence and a before-insert row-level Trigger:

- **Table** T1 with a numeric column (we want to simulate the auto-incremental behavior)
- **Sequence** SEQ_T1_ID1 (to use for the number generation)
- **Trigger** TRG_T1_ID1 (that populates our column automatically)

Obviously you need the CREATE TABLE, CREATE SEQUENCE and - optionally - CREATE TRIGGER privileges, and some quotas (maybe unlimited) on the default Tablespace (I will use the default one).

First solution (Sequence + Trigger + Table)

First of all, I want show you the solution that includes the creation of a Trigger. This allows you to avoid the explicit call to your Sequence in every INSERT statement.

Let's start:

```
CONRAD@orcl> CREATE TABLE T1
  2  (ID1       NUMBER,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> CREATE SEQUENCE SEQ_T1_ID1;

Sequence created.

CONRAD@orcl> CREATE OR REPLACE TRIGGER TRG_T1_ID1
  2  BEFORE INSERT ON T1
  3  FOR EACH ROW
  4  BEGIN
  5     SELECT SEQ_T1_ID1.NEXTVAL INTO :NEW.ID1 FROM DUAL;
  6  END;
  7  /

Trigger created.

CONRAD@orcl>
```

And now you can insert your first record, without any reference to the Sequence:

```
CONRAD@orcl> INSERT INTO T1 (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T1 ORDER BY ID1;

       ID1 TNAME
---------- --------------------------------------------------
         1 VALUE 1

CONRAD@orcl>
```

If you specify a value for the ID1 column, such a value is just ignored (thanks to the Trigger definition):

```
CONRAD@orcl> INSERT INTO T1 (ID1, TNAME) VALUES (19, 'VALUE 2');

1 row created.
```

```
CONRAD@orcl> SELECT * FROM T1 ORDER BY ID1;

       ID1 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         2 VALUE 2

CONRAD@orcl>
```

Moreover, I can't insert a NULL, because the Trigger always generates the next value from the Sequence (whether I supply a value or not):

```
CONRAD@orcl> INSERT INTO T1 (ID1, TNAME) VALUES (NULL, 'VALUE 3');

1 row created.

CONRAD@orcl> SELECT * FROM T1 ORDER BY ID1;

       ID1 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         2 VALUE 2
         3 VALUE 3

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

As you can see, the Trigger has automatically populated the column **ID1** for us, for each case. That's all.

But I can define my Trigger so that it accepts a supplied value too, if present:

```
CONRAD@orcl> CREATE OR REPLACE TRIGGER TRG_T1_ID1
  2  BEFORE INSERT ON T1
  3  FOR EACH ROW
  4  WHEN (NEW.ID1 IS NULL)
  5  BEGIN
  6     SELECT SEQ_T1_ID1.NEXTVAL INTO :NEW.ID1 FROM DUAL;
  7  END;
  8  /

Trigger created.

CONRAD@orcl> INSERT INTO T1 (ID1, TNAME) VALUES (19, 'VALUE 4');

1 row created.

CONRAD@orcl> SELECT * FROM T1 ORDER BY ID1;

       ID1 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         2 VALUE 2
         3 VALUE 3
        19 VALUE 4

CONRAD@orcl>
```

Still I can't insert a NULL, because the Trigger generates a value from the Sequence every time I supply a NULL:

```
CONRAD@orcl> INSERT INTO T1 (ID1, TNAME) VALUES (NULL, 'VALUE 5');

1 row created.
```

```
CONRAD@orcl> SELECT * FROM T1 ORDER BY ID1;

       ID1 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         2 VALUE 2
         3 VALUE 3
         4 VALUE 5
        19 VALUE 4

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

Second solution (Sequence + Table)

The second solution, as I mentioned earlier, has no Trigger that generates the next value for you, so you need to make an explicit call to Sequence's NEXTVAL method.

```
CONRAD@orcl> CREATE TABLE T2
  2  (ID2       NUMBER,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> CREATE SEQUENCE SEQ_T2_ID2;

Sequence created.

CONRAD@orcl>
```

Now you have to manually specify the Sequence as part of the INSERT statement:

```
CONRAD@orcl> INSERT INTO T2 (ID2, TNAME) VALUES (SEQ_T2_ID2.NEXTVAL, 'VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T2 ORDER BY ID2;

       ID2 TNAME
---------- --------------------------------------------------
         1 VALUE 1

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

You can supply a NULL only if you have not defined a NOT NULL constraint on the ID2 column. Keep in mind that the Table and the Sequence are completely independent from each other.

New 12c solutions

I will show you the two solutions that I mentioned in the Introduction: in the first one you need to create a Sequence and your Table, in the second one you need your Table only.

First solution (Sequence + Table)

As a first approach I wish to show you the one I have previously called (see Introduction) as an intermediate solution. You create a Sequence, independent from the Table, and then you set it as the default value in the Table definition.

Let's start…

```
CONRAD@orcl> CREATE SEQUENCE SEQ_T3_ID3;

Sequence created.

CONRAD@orcl> CREATE TABLE T3
  2  (ID3       NUMBER       DEFAULT SEQ_T3_ID3.NEXTVAL,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl>
```

The Sequence is now part of the Table definition:

```
CONRAD@orcl> SELECT DBMS_METADATA.GET_DDL('TABLE', 'T3') FROM DUAL;

DBMS_METADATA.GET_DDL('TABLE','T3')
--------------------------------------------------------------------------------

  CREATE TABLE "CONRAD"."T3"
   (    "ID3" NUMBER DEFAULT "CONRAD"."SEQ_T3_ID3"."NEXTVAL",
        "TNAME" VARCHAR2(128)
   ) SEGMENT CREATION DEFERRED
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  TABLESPACE "CONRAD_TAB"

CONRAD@orcl>
```

And in the [USER|ALL|DBA]_TAB_COLUMNS views we can see the DATA_DEFAULT column set:

```
CONRAD@orcl> SELECT COLUMN_ID, COLUMN_NAME, DATA_DEFAULT
  2  FROM USER_TAB_COLUMNS
  3  WHERE TABLE_NAME = 'T3'
  4  ORDER BY COLUMN_ID;

 COLUMN_ID   COLUMN_NAME   DATA_DEFAULT
----------   -----------   --------------------------------
         1   ID3           "CONRAD"."SEQ_T3_ID3"."NEXTVAL"
         2   TNAME

CONRAD@orcl>
```

And we have no constraint:

```
CONRAD@orcl> SELECT *
  2  FROM USER_CONSTRAINTS
  3  WHERE TABLE_NAME = 'T3';

no rows selected

CONRAD@orcl>
```

We can finally insert our first value, without specifying the ID3 column:

```
CONRAD@orcl> INSERT INTO T3 (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T3;

       ID3 TNAME
---------- --------------------------------------------------
         1 VALUE 1

CONRAD@orcl>
```

If we explicitly specify our ID3 value, Oracle® doesn't use the DEFAULT property defined for the ID3 column in the Table definition.

```
CONRAD@orcl> INSERT INTO T3 (ID3, TNAME) VALUES (19, 'VALUE 2');

1 row created.

CONRAD@orcl> SELECT * FROM T3 ORDER BY ID3;

       ID3 TNAME
---------- --------------------------------------------------
         1 VALUE 1
        19 VALUE 2

CONRAD@orcl>
```

We can even insert a NULL, in the ID3 column (if I have not defined a NOT NULL constraint, obviously):

```
CONRAD@orcl> INSERT INTO T3 (ID3, TNAME) VALUES (NULL, 'VALUE 3');

1 row created.

CONRAD@orcl> SELECT * FROM T3 ORDER BY ID3;

       ID3 TNAME
---------- --------------------------------------------------
         1 VALUE 1
        19 VALUE 2
           VALUE 3

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

As we expected, the Sequence didn't miss any value:

```
CONRAD@orcl> SELECT SEQ_T3_ID3.NEXTVAL FROM DUAL;

   NEXTVAL
----------
         2

CONRAD@orcl>
```

But there's another available option: I can tell Oracle® to use its DEFAULT value (the Sequence NEXTVAL) even when I explicitly supply a NULL for the ID3 column: I use the **DEFAULT ON NULL** property.

Please note that I have to delete any record which has ID3 set to NULL, in order to alter the table structure as desired:

```
CONRAD@orcl> ALTER TABLE T3
  2  MODIFY (ID3       NUMBER       DEFAULT ON NULL SEQ_T3_ID3.NEXTVAL);
ALTER TABLE T3
*
ERROR at line 1:
ORA-02296: cannot enable (CONRAD.) - null values found

CONRAD@orcl> DELETE FROM T3 WHERE ID3 IS NULL;

1 row deleted.

CONRAD@orcl> ALTER TABLE T3
  2  MODIFY (ID3       NUMBER       DEFAULT ON NULL SEQ_T3_ID3.NEXTVAL);

Table altered.

CONRAD@orcl>
```

Now, when I insert a NULL I get the next available Sequence value:

```
CONRAD@orcl> INSERT INTO T3 (ID3, TNAME) VALUES (NULL, 'VALUE 3');

1 row created.

CONRAD@orcl> SELECT * FROM T3 ORDER BY ID3;

       ID3 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         3 VALUE 3
        19 VALUE 2

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl> SELECT COLUMN_ID, COLUMN_NAME, DATA_DEFAULT, DEFAULT_ON_NULL
  2  FROM USER_TAB_COLUMNS
  3  WHERE TABLE_NAME = 'T3'
  4  ORDER BY COLUMN_ID;

 COLUMN_ID   COLUMN_NAME   DATA_DEFAULT                          DEF
----------   -----------   -------------------------------   ---
         1   ID3           "CONRAD"."SEQ_T3_ID3"."NEXTVAL"   YES
         2   TNAME                                           NO

CONRAD@orcl>
```

And thanks to DBMS_METADATA.GET_DDL function we can see the new definition of the ID3 column:

```
    "ID3" NUMBER DEFAULT "CONRAD"."SEQ_T3_ID3"."NEXTVAL" NOT NULL ENABLE
```

As a matter of fact, now we can see a new constraint:

```
CONRAD@orcl> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION
  2  FROM USER_CONSTRAINTS
  3  WHERE TABLE_NAME = 'T3';

CONSTRAINT_NAME CONSTRAINT_TYPE SEARCH_CONDITION
--------------- --------------- -----------------
SYS_C0011098    C               "ID3" IS NOT NULL

CONRAD@orcl>
```

Another interesting thing to note is that I can drop the Sequence, even if it's used as a DEFAULT value in a Table definition:

```
CONRAD@orcl> DROP SEQUENCE SEQ_T3_ID3;

Sequence dropped.

CONRAD@orcl>
```

Obviously this DDL command has committed our Transaction, so we don't need to issue a COMMIT. Despite the fact that I have dropped the Sequence, I can still see it in the Table definition:

```
CONRAD@orcl> SELECT DBMS_METADATA.GET_DDL('TABLE', 'T3') FROM DUAL;

DBMS_METADATA.GET_DDL('TABLE','T3')
--------------------------------------------------------------------------------

  CREATE TABLE "CONRAD"."T3"
   (    "ID3" NUMBER DEFAULT "CONRAD"."SEQ_T3_ID3"."NEXTVAL",
        "TNAME" VARCHAR2(128)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 4194304 NEXT 4194304 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "CONRAD_TAB"

CONRAD@orcl>
```

But what happens when we try to insert a record into the Table?
It succeeds in the only case we explicitly specify a value for the ID3 column, otherwise it tries to use the Sequence and it fails:

```
CONRAD@orcl> INSERT INTO T3 (ID3, TNAME) VALUES (37, 'VALUE 4');

1 row created.

CONRAD@orcl> INSERT INTO T3 (TNAME) VALUES ('VALUE 5');
INSERT INTO T3 (TNAME) VALUES ('VALUE 5')
            *
ERROR at line 1:
ORA-02289: sequence does not exist

CONRAD@orcl>
```

So, finally, commit our transaction and let's see the final Table's content:

```
CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl> SELECT * FROM T3;

       ID3 TNAME
---------- --------------------------------------------------
         1 VALUE 1
         3 VALUE 3
        19 VALUE 2
        37 VALUE 4

CONRAD@orcl>
```

Second solution (Table)

This approach is really simple, because you don't need to create either a Trigger or a Sequence, and your Table is all that you need: a Table with a numeric column that has the Identity, auto-incremental, property. Oracle®, behind the scenes, creates the Sequence for you.
Please note: the owner of the Table (CONRAD, in my case) must have both the CREATE TABLE privilege, as well as the CREATE SEQUENCE. And when you create your Table you don't immediately need quotas on the Tablespace you create the Table in, not until you insert your first record (thanks to Deferred Segment Creation, from 11gR2 onward).
The Identity column must be numeric. You almost always define it as NUMBER (with neither precision nor scale), but you can specify a precision too, if you want. In this latter case pay attention that if you reach the limit that you specify, Oracle® throws an error (as you will see in one of the following examples) because the system-generated Sequence is independent of your column precision.

Let's try this…

```
CONRAD@orcl> CREATE TABLE T4
  2  (ID4       NUMBER      GENERATED AS IDENTITY,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T4 (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T4;

       ID4 TNAME
---------- --------------------------------------------------
         1 VALUE 1

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

We obtained the same behavior and the same result, but with much less work.

In the [USER|ALL|DBA]_TABLES views we can easily see whether a Table contains an Identity column (T4) or not (T1, T2, T3):

```
CONRAD@orcl> SELECT TABLE_NAME, HAS_IDENTITY
  2  FROM USER_TABLES
  3  WHERE TABLE_NAME IN ('T1', 'T2', 'T3', 'T4')
  4  ORDER BY TABLE_NAME;

TABLE_NAME   HAS
----------   ---
T1           NO
T2           NO
T3           NO
T4           YES

CONRAD@orcl>
```

And in the [USER|ALL|DBA]_TAB_COLUMNS views we can see which column has the Identity property (ID4):

```
CONRAD@orcl> SELECT TABLE_NAME, COLUMN_NAME, IDENTITY_COLUMN
  2  FROM USER_TAB_COLUMNS
  3  WHERE TABLE_NAME IN ('T1', 'T2', 'T3', 'T4')
  4  ORDER BY TABLE_NAME, COLUMN_ID;
```

```
TABLE_NAME   COLUMN_NAME   IDE
----------   -----------   ---
T1           ID1           NO
T1           TNAME         NO
T2           ID2           NO
T2           TNAME         NO
T3           ID3           NO
T3           TNAME         NO
T4           ID4           YES
T4           TNAME         NO

8 rows selected.

CONRAD@orcl>
```

Another way to look at the Identity columns is by the [USER|ALL|DBA]_TAB_IDENTITY_COLS views:

```
CONRAD@orcl> SELECT TABLE_NAME, COLUMN_NAME, GENERATION_TYPE, IDENTITY_OPTIONS
  2  FROM USER_TAB_IDENTITY_COLS
  3  ORDER BY TABLE_NAME;

TABLE_NAME   COLUMN_NAME   GENERATION   IDENTITY_OPTIONS
----------   -----------   ----------   -----------------
T4           ID4           ALWAYS       START WITH: 1, INCREMENT BY: 1,
                                        MAX_VALUE: 9999999999999999999999999999,
                                        MIN_VALUE: 1, CYCLE_FLAG: N,
                                        CACHE_SIZE: 20, ORDER_FLAG: N

CONRAD@orcl>
```

The GENERATION_TYPE column shows you that Oracle® will ALWAYS populate the ID4 column with a Sequence value for you. That's the default behavior, as you can see, because I have not specified the ALWAYS keyword earlier in my CREATE TABLE command. I could have created my Table as:

```
CREATE TABLE T4
(ID4       NUMBER      GENERATED ALWAYS AS IDENTITY,
 TNAME     VARCHAR2(128));
```

That's the same.
With the ALWAYS option, if I explicitly specify a value for the ID4 column in my INSERT statement, I get an error:

```
CONRAD@orcl> INSERT INTO T4 (ID4, TNAME) VALUES (19, 'VALUE 2');
INSERT INTO T4 (ID4, TNAME) VALUES (19, 'VALUE 2')
                *
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column

CONRAD@orcl>
```

Oracle® has automatically created a NOT NULL Check Constraint (I have not explicitly specified "NOT NULL" in my CREATE TABLE statement), and this constraint is "Not Deferrable" and "Immediate", too:

```
CONRAD@orcl> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME,
  2         SEARCH_CONDITION, DEFERRABLE, DEFERRED
  3  FROM USER_CONSTRAINTS
  4  WHERE TABLE_NAME IN ('T1', 'T2', 'T3', 'T4')
  5  ORDER BY TABLE_NAME, CONSTRAINT_NAME;

CONSTRAINT_NAME   C   TABLE_NAME   SEARCH_CONDITION    DEFERRABLE       DEFERRED
---------------   -   ----------   -----------------   --------------   ---------
SYS_C0011088      C   T4           "ID4" IS NOT NULL   NOT DEFERRABLE   IMMEDIATE

CONRAD@orcl>
```

And it has automatically created a Sequence for us, too. So, let's see all these Sequences (**manually** created and **system**-generated):

```
CONRAD@orcl> SELECT * FROM USER_SEQUENCES
  2  ORDER BY SEQUENCE_NAME;

SEQUENCE_NAME    MIN_VALUE                      MAX_VALUE    INCREMENT_BY    CYCLE_FLAG
-------------    ---------    ----------------------------    ------------    ----------
ISEQ$$_95174             1    99999999999999999999999999                1             N
SEQ_T1_ID1               1    99999999999999999999999999                1             N
SEQ_T2_ID2               1    99999999999999999999999999                1             N


ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                     20             21                       N               N
N                     20             21                       N               N
N                     20             21                       N               N

CONRAD@orcl>
```

Remember: we have three Sequences only - not four - because we have dropped one of them…

Furthermore, we can see the name of the automatically created Sequence in the DATA_DEFAULT column of the [USER|ALL|DBA]_TAB_COLUMNS views:

```
CONRAD@orcl> SELECT TABLE_NAME, COLUMN_NAME, DATA_DEFAULT
  2  FROM USER_TAB_COLUMNS
  3  WHERE IDENTITY_COLUMN = 'YES'
  4  ORDER BY TABLE_NAME, COLUMN_ID;

TABLE_NAME    COLUMN_NAME    DATA_DEFAULT
----------    -----------    --------------------------------
T4            ID4            "CONRAD"."ISEQ$$_95174".nextval

CONRAD@orcl>
```

Finally, for the sake of completeness, we obviously see that there's only one Trigger, as we expected: the one we have manually created in the first pre-12c solution, because the Identity doesn't need it:

```
CONRAD@orcl> SELECT TRIGGER_NAME, TRIGGER_TYPE FROM USER_TRIGGERS;

TRIGGER_NAME      TRIGGER_TYPE
------------      ---------------
TRG_T1_ID1        BEFORE EACH ROW

CONRAD@orcl>
```

From the USER_SEQUENCES output shown earlier, we can see that all the Sequence options are exactly the same for both (manual and automatic). So Oracle® has automatically created the Sequence with all the default options.

To be more precise, in SYS.SEQ$ we can see a different FLAGS bitmap (**the sixth bit** is set to "1" for the system-generated Sequence):
- In the "manually" created Sequence, we have a bitmap that's something like "0000**0**0**1**000"
- In the "automatically" created one, we have a different bitmap: something like "0000**1**0**1**000"

I might be wrong, however I have never seen the sixth bit set to "1" for Sequences other than system-generated ones.
You can find references to the **fourth bit** (decode(bitand(flags, 8), 8, 1, 0)) in the following Views:
- LOGSTDBY_SUPPORT (CURRENT_SBY column) in 10gR2
- LOGSTDBY_SUPPORT_SEQ (CURRENT_SBY column) in 11gR2 and 12cR1

In 12c we can see a reference to the **seventh bit** (SESSION_FLAG) and the **tenth bit** (KEEP_VALUE) in the [USER|ALL|DBA]_SEQUENCES views.

```
SYS@orcl> SELECT O.NAME, S.FLAGS
  2  FROM SYS.SEQ$ S, OBJ$ O
  3  WHERE S.OBJ# = O.OBJ#
  4    AND O.NAME IN ('ISEQ$$_95174', 'SEQ_T1_ID1', 'SEQ_T2_ID2')
  5  ORDER BY O.NAME;

NAME                 FLAGS
---------------   -------
ISEQ$$_95174            40
SEQ_T1_ID1               8
SEQ_T2_ID2               8

SYS@orcl>
```

The Sequence created by Oracle® behind the scenes has a system-generated name that refers to the Table's Object Id (95174 in my case):

```
CONRAD@orcl> SELECT OBJECT_NAME FROM USER_OBJECTS WHERE OBJECT_ID = 95174;

OBJECT_NAME
------------------------
T4

CONRAD@orcl>
```

The Sequence name contains the Object Id of the Table to which is bounded, so we can't have more than one Identity column per Table (this is a more than acceptable limit, because it's uncommon to have a situation in which you need more than one Identity column in the same Table).

We can't drop a system-generated Sequence:

```
CONRAD@orcl> DROP SEQUENCE ISEQ$$_95174;
DROP SEQUENCE ISEQ$$_95174
              *
ERROR at line 1:
ORA-32794: cannot drop a system-generated Sequence

CONRAD@orcl>
```

And we can't modify it:

```
CONRAD@orcl> ALTER SEQUENCE ISEQ$$_95174 INCREMENT BY 2;
ALTER SEQUENCE ISEQ$$_95174 INCREMENT BY 2
*
ERROR at line 1:
ORA-32793: cannot alter a system-generated Sequence

CONRAD@orcl>
```

Moreover, I can use this system-generated Sequence as if it were a regular Sequence, independently of the Table (for our purpose it does not make sense, of course).

Let's generate, for example, the next value:

```
CONRAD@orcl> SELECT ISEQ$$_95174.NEXTVAL FROM DUAL;

   NEXTVAL
----------
        21

CONRAD@orcl>
```

Now this value (21) has been lost, obviously, as we can see if we insert another record into the Table.
So, as with every Sequence object, we are never guaranteed about a sequential numbering.

```
CONRAD@orcl> INSERT INTO T4 (TNAME) VALUES ('VALUE 2');

1 row created.

CONRAD@orcl> SELECT * FROM T4;

       ID4 TNAME
---------- --------------------------------------------------
         1 VALUE 1
        22 VALUE 2

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

By default, Oracle® caches 20 values for a Sequence - we can see it from the CACHE_SIZE column - so the LAST_NUMBER value is the next value we'll get in case we lose the cached ones (for example after an Instance restart, or after a Shared Pool flushing):

```
CONRAD@orcl> SELECT SEQUENCE_NAME, CACHE_SIZE, LAST_NUMBER
  2  FROM USER_SEQUENCES
  3  ORDER BY SEQUENCE_NAME;

SEQUENCE_NAME   CACHE_SIZE   LAST_NUMBER
-------------   ----------   -----------
ISEQ$$_95174            20            41
SEQ_T1_ID1              20            21
SEQ_T2_ID2              20            21

CONRAD@orcl>
```

```
SYS@orcl> ALTER SYSTEM FLUSH SHARED_POOL;

System altered.

SYS@orcl>
```

The NEXTVAL will be 41, instead of 23:

```
CONRAD@orcl> SELECT ISEQ$$_95174.NEXTVAL FROM DUAL;

   NEXTVAL
----------
        41

CONRAD@orcl> SELECT SEQUENCE_NAME, CACHE_SIZE, LAST_NUMBER
  2  FROM USER_SEQUENCES
  3  ORDER BY SEQUENCE_NAME;

SEQUENCE_NAME   CACHE_SIZE   LAST_NUMBER
-------------   ----------   -----------
ISEQ$$_95174            20            61
SEQ_T1_ID1              20            21
SEQ_T2_ID2              20            21

CONRAD@orcl>
```

So, the LAST_NUMBER column is something like a High-water Mark of the Sequence (although it's a "next" value, and not a "last" value). In fact in the SYS.SEQ$ view, on which the [USER|ALL|DBA]_SEQUENCES views are based, is named HIGHWATER:

```
SYS@orcl> SELECT O.OBJ#, O.NAME, S.CACHE, S.HIGHWATER
  2  FROM SYS.SEQ$ S, SYS.OBJ$ O
  3  WHERE S.OBJ# = O.OBJ# AND O.NAME = 'ISEQ$$_95174';

 OBJ#   NAME           CACHE   HIGHWATER
-----   ------------   -----   ---------
95175   ISEQ$$_95174      20          61

SYS@orcl>
```

It's interesting to see how Oracle® created the Table with the Identity column:

```
CONRAD@orcl> SELECT DBMS_METADATA.GET_DDL('TABLE', 'T4') FROM DUAL;

DBMS_METADATA.GET_DDL('TABLE','T4')
--------------------------------------------------------------------------------

  CREATE TABLE "CONRAD"."T4"
   (    "ID4" NUMBER GENERATED ALWAYS AS IDENTITY
                     MINVALUE 1 MAXVALUE 9999999999999999999999999999
                     INCREMENT BY 1 START WITH 1 CACHE 20
                     NOORDER  NOCYCLE  NOT NULL ENABLE,
        "TNAME" VARCHAR2(128)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 4194304 NEXT 4194304 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "CONRAD_TAB"

CONRAD@orcl>
```

It contains all the parameters of a full Sequence definition.
This is useful, because we can define this Identity in a handful of ways. For example we could define our Identity so that it starts with a different value, or with a different incremental gap (INCREMENT BY).

For example, let's suppose we want to start with the value **500**, and we want an increment by **10**. With the default CACHE=20 we have cached in the Shared Pool all the values between 500 and 690.

```
CONRAD@orcl> CREATE TABLE T
  2  (ID        NUMBER      GENERATED ALWAYS AS IDENTITY
  3                         INCREMENT BY 10 START WITH 500,
  4   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 2');

1 row created.

CONRAD@orcl> SELECT * FROM T;

        ID TNAME
---------- --------------------------------------------------
       500 VALUE 1
       510 VALUE 2

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

Modifying the Identity parameters

We can modify the Table structure and its related Sequence parameters. In this case Oracle® modifies the Sequence definition but it obviously maintains the same name, because the Sequence is bound to the Table's Object Id.

Now we'll see that if we change the START WITH parameter, Oracle® recreates (resets) the Sequence (we can't change the START WITH parameter of a traditional Sequence, as a matter of fact).
But if we change some other parameters, Oracle® doesn't recreate the Sequence and keeps its current values.
After every change, we will see the output from the DBMS_METADATA.GET_DDL function and the USER_SEQUENCES view.
We can use the Table T from the previous example. So, first of all, we have to find the system-generated Sequence name:

```
CONRAD@orcl> SELECT COLUMN_NAME, DATA_DEFAULT
  2  FROM USER_TAB_COLUMNS
  3  WHERE TABLE_NAME = 'T' AND IDENTITY_COLUMN = 'YES';

COLUMN_NAME    DATA_DEFAULT
-----------    -------- ----------------------
ID             "CONRAD"."ISEQ$$_95196".nextval

CONRAD@orcl>
```

The last generated value, in the previous example, was 510. So, the next available value in the cache is 520. In case we lose all the cached values, we will get 700:

```
CONRAD@orcl> SELECT ISEQ$$_95196.NEXTVAL FROM DUAL;

   NEXTVAL
----------
       520

CONRAD@orcl>
```

Let's see the current Sequence settings (remember, from the previous example, that we have specified our custom, non-default, **increment** and **starting** values):

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 10
START WITH 500 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                      MAX_VALUE    INCREMENT_BY    CYCLE_FLAG
-------------    ---------    ----------------------------    ------------    ----------
ISEQ$$_95196             1    9999999999999999999999999999              10    N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                     20            700                       N               N
```

Now, for example, let's modify the increment (INCREMENT BY) and caching (CACHE) parameters:

```
CONRAD@orcl> ALTER TABLE T
  2  MODIFY (ID NUMBER       GENERATED ALWAYS AS IDENTITY
  3                          INCREMENT BY 50
  4                          NOCACHE);

Table altered.

CONRAD@orcl>
```

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 50
START WITH 500 NOCACHE  NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                      MAX_VALUE    INCREMENT_BY   CYCLE_FLAG
-------------    ---------   ----------------------------    ------------   ----------
ISEQ$$_95196             1   9999999999999999999999999999              50   N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                      0            570                       N               N
```

The last generated value was 520. But now we don't have cached values, so the next value to be generated - with the new increment of 50 - is 570, as we can see:

```
CONRAD@orcl> SELECT ISEQ$$_95196.NEXTVAL FROM DUAL;

   NEXTVAL
----------
       570

CONRAD@orcl>
```

So Oracle® has not reset the Sequence, but it has used the new `INCREMENT` value to calculate the next to be generated.

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 50
START WITH 500 NOCACHE  NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                      MAX_VALUE    INCREMENT_BY   CYCLE_FLAG
-------------    ---------   ----------------------------    ------------   ----------
ISEQ$$_95196             1   9999999999999999999999999999              50   N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                      0            620                       N               N
```

Now, let's modify the starting value (`START WITH`) only:

```
CONRAD@orcl> ALTER TABLE T
  2  MODIFY (ID NUMBER      GENERATED ALWAYS AS IDENTITY
  3                         START WITH 100);

Table altered.

CONRAD@orcl>
```

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1
START WITH 100 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                      MAX_VALUE    INCREMENT_BY   CYCLE_FLAG
-------------    ---------   ----------------------------    ------------   ----------
ISEQ$$_95196             1   9999999999999999999999999999               1   N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                     20            100                       N               N
```

As you can see, Oracle® has reset the current parameters (`INCREMENT BY` and `CACHE`) to the default ones. And then, with a new starting value of 100, the next value to be generated has to be 100.

```
CONRAD@orcl> SELECT ISEQ$$_95196.NEXTVAL FROM DUAL;

   NEXTVAL
----------
       100

CONRAD@orcl>
```

After the first number generation (100), thanks to the caching of the next 20 values, the next to be generated (in the case of loss of all the cached ones) is 120.

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1
START WITH 100 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                       MAX_VALUE    INCREMENT_BY   CYCLE_FLAG
-------------    ---------   -----------------------------    ------------   ----------
ISEQ$$_95196             1   999999999999999999999999999              1    N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                     20            120                       N               N
```

You have seen that by changing the START WITH parameter, Oracle® recreates the Sequence from scratch. In fact, you can't modify the starting value in a manually created Sequence, at least not with the ALTER SEQUENCE START WITH command:

```
CONRAD@orcl> CREATE SEQUENCE SEQ_T;

Sequence created.

CONRAD@orcl> ALTER SEQUENCE SEQ_T START WITH 4;
ALTER SEQUENCE SEQ_T START WITH 4
                   *
ERROR at line 1:
ORA-02283: cannot alter starting sequence number

CONRAD@orcl> DROP SEQUENCE SEQ_T;

Sequence dropped.

CONRAD@orcl>
```

Note: you can change the next value to be generated with a little work-around, without recreating the Sequence, playing with the INCREMENT BY setting (the START WITH setting remains unchanged, anyway). Keep in mind that there's no CREATE OR REPLACE SEQUENCE command, so should you need to recreate the Sequence you have to drop it, recreate it and re-grant the needed privileges to the users.

"By Default" clause

By default (the ALWAYS clause), as previously explained, we can't insert a value into a column created as Identity:

```
CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 3');
INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 3')
               *
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column

CONRAD@orcl>
```

If we need to insert such "free" values, independently of the Identity mechanism, we have to specify the BY DEFAULT clause, instead of **ALWAYS**.

```
CONRAD@orcl> ALTER TABLE T
  2  MODIFY (ID NUMBER      GENERATED BY DEFAULT AS IDENTITY);

Table altered.

CONRAD@orcl>
```

This command has not altered any Sequence parameter. They are the same as before the DDL command.

```
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1
START WITH 100 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE
```

```
SEQUENCE_NAME    MIN_VALUE                       MAX_VALUE    INCREMENT_BY   CYCLE_FLAG
-------------    ---------   -----------------------------    ------------   ----------
ISEQ$$_95196             1   9999999999999999999999999999                1   N

ORDER_FLAG    CACHE_SIZE    LAST_NUMBER    PARTITION_COUNT    SESSION_FLAG    KEEP_VALUE
----------    ----------    -----------    ---------------    ------------    ----------
N                     20            120                       N               N
```

And you can recognize Identity columns with `BY DEFAULT` property set:

```
CONRAD@orcl> SELECT TABLE_NAME, COLUMN_NAME, GENERATION_TYPE
  2  FROM USER_TAB_IDENTITY_COLS
  3  ORDER BY TABLE_NAME;

TABLE_NAME    COLUMN_NAME    GENERATION
----------    -----------    ----------
T             ID             BY DEFAULT
T4            ID4            ALWAYS

CONRAD@orcl>
```

So, now, I can specify my Identity column or not:

```
CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 3');

1 row created.

CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 4');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID   TNAME
----------   -----------------
        19   VALUE 4
       120   VALUE 3
       500   VALUE 1
       510   VALUE 2

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

But I can't supply a `NULL` for the Identity column:

```
CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (NULL, 'VALUE 5');
INSERT INTO T (ID, TNAME) VALUES (NULL, 'VALUE 5')
                                 *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("CONRAD"."T"."ID")

CONRAD@orcl>
```

The real problem with the possibility of inserting both system generated values and user-supplied ones, is that in this way we may have conflicts, because we may insert the same value more than once: with the auto-increment Sequence and manually. Obviously, this doesn't happen if you define the Identity column as a Primary Key, or you put a Unique constraint on it (in both cases you will have a unique constraint violation).
But you may take advantage of this capability, for example, to get back a lost value.

"By Default on NULL" clause

If we want to have the possibility of specifying a NULL for the Identity column (in the previous example we got an error), we need the **BY DEFAULT ON NULL** clause.
But - pay attention - this doesn't mean that you can insert a NULL. Absolutely not! Remember that Identity columns are - by definition - NOT NULL constrained.
So, if you specify a NULL, instead of getting the ORA-01400 error, your insert will succeed; but Oracle® will generate the Sequence NEXTVAL for you, and the NULL will be ignored.
So, for example, these two commands are equivalent:

```
INSERT INTO T (ID, TNAME) VALUES (NULL, 'VALUE X');

INSERT INTO T (TNAME) VALUES ('VALUE X');
```

Let's try this:

```
CONRAD@orcl> ALTER TABLE T
  2  MODIFY (ID NUMBER      GENERATED BY DEFAULT ON NULL AS IDENTITY);

Table altered.

CONRAD@orcl>
```

We can recognize Identity columns with BY DEFAULT ON NULL property:

```
CONRAD@orcl> SELECT COLUMN_ID, COLUMN_NAME, DATA_DEFAULT,
  2         IDENTITY_COLUMN, DEFAULT_ON_NULL
  3  FROM USER_TAB_COLUMNS
  4  WHERE TABLE_NAME = 'T'
  5  ORDER BY COLUMN_ID;

 COLUMN_ID   COLUMN_NAME   DATA_DEFAULT                          IDE   DEF
----------   -----------   ------------                          ---   ---
         1   ID            "CONRAD"."ISEQ$$_95196".nextval       YES   YES
         2   TNAME                                               NO    NO

CONRAD@orcl>
```

If we try to insert a NULL, as I previously stated, we get:

```
CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (NULL, 'VALUE 5');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID   TNAME
----------   --------------------------
        19   VALUE 4
       120   VALUE 3
       121   VALUE 5
       500   VALUE 1
       510   VALUE 2

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

Dropping a Table with an Identity column

When we drop a Table with an Identity column, the system-generated Sequence will be dropped too, but in the only case you specify the PURGE option. Otherwise it will remain until you purge the Recycle Bin.
Let's try with our Table **T** and its Sequence **ISEQ$$_95196**:

```
CONRAD@orcl> SELECT SEQUENCE_NAME
  2  FROM USER_SEQUENCES
  3  WHERE SEQUENCE_NAME = 'ISEQ$$_95196';

SEQUENCE_NAME
--------------------
ISEQ$$_95196

CONRAD@orcl> DROP TABLE T;

Table dropped.

CONRAD@orcl> SELECT SEQUENCE_NAME
  2  FROM USER_SEQUENCES
  3  WHERE SEQUENCE_NAME = 'ISEQ$$_95196';

SEQUENCE_NAME
--------------------
ISEQ$$_95196

CONRAD@orcl> PURGE RECYCLEBIN;

Recyclebin purged.

CONRAD@orcl> SELECT SEQUENCE_NAME
  2  FROM USER_SEQUENCES
  3  WHERE SEQUENCE_NAME = 'ISEQ$$_95196';

no rows selected

CONRAD@orcl>
```

Additional restrictions

The Datatype for an Identity column must be Numeric:

```
CONRAD@orcl> CREATE TABLE T
  2  (ID        VARCHAR2(11)   GENERATED ALWAYS AS IDENTITY);
(ID        VARCHAR2(11)   GENERATED ALWAYS AS IDENTITY)
                                                       *
ERROR at line 2:
ORA-30675: identity column must be a numeric type

CONRAD@orcl>
```

As I previously mentioned, we can't define more than one Identity column per Table:

```
CONRAD@orcl> CREATE TABLE T
  2  (ID1       NUMBER         GENERATED ALWAYS AS IDENTITY,
  3   ID2       NUMBER         GENERATED ALWAYS AS IDENTITY);
 ID2       NUMBER         GENERATED ALWAYS AS IDENTITY)
                                                       *
ERROR at line 3:
ORA-30669: table can have only one identity column

CONRAD@orcl>
```

As any other Sequence, all the values generated are not rollbackable. If you open a Transaction, generate some values, and then issue a ROLLBACK command, those values are lost:

```
CONRAD@orcl> CREATE TABLE T
  2  (ID        NUMBER      GENERATED ALWAYS AS IDENTITY,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 2');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID   TNAME
----------   --------------------------
         1   VALUE 1
         2   VALUE 2

CONRAD@orcl> ROLLBACK;

Rollback complete.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

ID   TNAME
----------   --------------------------
         3   VALUE 1

CONRAD@orcl> COMMIT;

Commit complete.

CONRAD@orcl>
```

If we create a Table with a CTAS (Create Table … As Select …) command from a Table containing an Identity column, the new Table will not inherit the Identity column properties, but only the NUMBER datatype and the NOT NULL constraint:

```
CONRAD@orcl> CREATE TABLE T_NEW
  2  AS
  3  SELECT ID, TNAME
  4  FROM T;

Table created.

CONRAD@orcl> SELECT DBMS_METADATA.GET_DDL('TABLE', 'T') FROM DUAL;

DBMS_METADATA.GET_DDL('TABLE','T')
--------------------------------------------------------------------------------

  CREATE TABLE "CONRAD"."T"
   (    "ID" NUMBER GENERATED ALWAYS AS IDENTITY
                    MINVALUE 1 MAXVALUE 9999999999999999999999999999
                    INCREMENT BY 1 START WITH 1 CACHE 20
                    NOORDER  NOCYCLE  NOT NULL ENABLE,
        "TNAME" VARCHAR2(128)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 4194304 NEXT 4194304 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
```

```
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "CONRAD_TAB"

CONRAD@orcl> SELECT DBMS_METADATA.GET_DDL('TABLE', 'T_NEW') FROM DUAL;

DBMS_METADATA.GET_DDL('TABLE','T_NEW')
--------------------------------------------------------------------------------

  CREATE TABLE "CONRAD"."T_NEW"
   (    "ID" NUMBER NOT NULL ENABLE,
        "TNAME" VARCHAR2(128)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 4194304 NEXT 4194304 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "CONRAD_TAB"

CONRAD@orcl> DROP TABLE T PURGE;

Table dropped.

CONRAD@orcl> DROP TABLE T_NEW PURGE;

Table dropped.

CONRAD@orcl>
```

We can't modify a Table's column from NUMBER to Identity (even if the Table is empty):

```
CONRAD@orcl> CREATE TABLE T
  2  (ID      NUMBER,
  3   TNAME   VARCHAR2(128));

Table created.

CONRAD@orcl> ALTER TABLE T
  2  MODIFY (ID      NUMBER      GENERATED AS IDENTITY);
MODIFY (ID      NUMBER      GENERATED AS IDENTITY)
        *
ERROR at line 2:
ORA-30673: column to be modified is not an identity column

CONRAD@orcl> DROP TABLE T PURGE;

Table dropped.

CONRAD@orcl>
```

Another thing to note is that a TRUNCATE TABLE command doesn't reset the Identity column starting value. If you have a little experience with Microsoft® SQL Server®, for example, you know that after such a command you'll get the Identity column reset too.

```
CONRAD@orcl> CREATE TABLE T
  2  (ID        NUMBER(1)      GENERATED AS IDENTITY,
  3   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID     TNAME
----------     --------------------
         1     VALUE 1
```

```
CONRAD@orcl> TRUNCATE TABLE T;

Table truncated.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID     TNAME
----------     --------------------
         2     VALUE 1

CONRAD@orcl> DROP TABLE T PURGE;

Table dropped.

CONRAD@orcl>
```

What about errors?

Does any error cause a loss of the next Sequence value? Obviously not… not, of course, if Oracle® detects it before the statement execution. Let's see a couple of examples:

```
CONRAD@orcl> CREATE TABLE T
  2  (ID      NUMBER   GENERATED ALWAYS AS IDENTITY,
  3   TNAME   VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 1');

1 row created.

CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 2');
INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 2')
               *
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column

CONRAD@orcl> INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 2');
INSERT INTO T (ID, TNAME) VALUES (19, 'VALUE 2')
               *
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE 2');

1 row created.

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID     TNAME
----------     --------------------
         1     VALUE 1
         2     VALUE 2

CONRAD@orcl>
```

We get two consecutive errors but we preserve the next Sequence value (2). Oracle® throws the exception before the statement execution (the Sequence value hasn't been generated yet).

But what happens if I get an error about the datatype precision?
For example, let's create an Identity column with precision "1" (the maximum allowed value will be "9") and define the Sequence with a starting value of "9" so that the second INSERT statement will fail.

```
CONRAD@orcl> DROP TABLE T PURGE;

Table dropped.

CONRAD@orcl> CREATE TABLE T
  2  (ID        NUMBER(1)      GENERATED AS IDENTITY
  3                            START WITH 9,
  4   TNAME     VARCHAR2(128));

Table created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE X');

1 row created.

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE X');
INSERT INTO T (TNAME) VALUES ('VALUE X')
*
ERROR at line 1:
ORA-01438: value larger than specified precision allowed for this column

CONRAD@orcl> INSERT INTO T (TNAME) VALUES ('VALUE X');
INSERT INTO T (TNAME) VALUES ('VALUE X')
*
ERROR at line 1:
ORA-01438: value larger than specified precision allowed for this column

CONRAD@orcl> SELECT * FROM T ORDER BY ID;

        ID     TNAME
----------     --------------------
         9     VALUE X

CONRAD@orcl> SELECT COLUMN_NAME, DATA_DEFAULT
  2  FROM USER_TAB_COLUMNS
  3  WHERE TABLE_NAME = 'T' AND IDENTITY_COLUMN = 'YES';

COLUMN_NAME     DATA_DEFAULT
-----------     ------------------------------------------------------------
ID              "CONRAD"."ISEQ$$_95329".nextval

CONRAD@orcl> SELECT "CONRAD"."ISEQ$$_95329".NEXTVAL FROM DUAL;

   NEXTVAL
----------
        12

CONRAD@orcl>
```

This time we have lost two values (10 and 11), because the Sequence values have been generated before getting the error about the precision.

```
CONRAD@orcl> DROP TABLE T PURGE;

Table dropped.

CONRAD@orcl>
```

A basic performance test for INSERT statements

It's interesting to create a little performance test for the discussed solutions. I'll use the OBJECT_NAME column from the DBA_OBJECTS view to populate the TNAME fields, and I will measure the total elapsed time. Furthermore, I will use my four Tables **T1**, **T2**, **T3** and **T4**, because they are still alive...
I will execute each of the following blocks of SQL commands for **three consecutive times**, and I will report the best elapsed time for each of the four solutions.
Both the two 12c solutions use the Sequence as a default value for the IDx column, so they almost have the same execution time.

First of all, we need to prepare our environment:

For the **first test** I will use the following version of the Trigger (I have presented two different versions, earlier):

```
CONRAD@orcl> CREATE OR REPLACE TRIGGER TRG_T1_ID1
  2  BEFORE INSERT ON T1
  3  FOR EACH ROW
  4  BEGIN
  5     SELECT SEQ_T1_ID1.NEXTVAL INTO :NEW.ID1 FROM DUAL;
  6  END;
  7  /

Trigger created.

CONRAD@orcl>
```

For the **third test** we need to recreate our Sequence SEQ_T3_ID3 that we have dropped in our earlier tests.

```
CONRAD@orcl> CREATE SEQUENCE SEQ_T3_ID3;

Sequence created.

CONRAD@orcl>
```

And now we ask SQL*Plus to show us the query elapsed times:

```
CONRAD@orcl> set timing on
```

Let's start!

1. First Pre-12c solution (Sequence + Trigger + Table)

My best execution time:

```
CONRAD@orcl> TRUNCATE TABLE T1 REUSE STORAGE;

Table truncated.

Elapsed: 00:00:00.51
CONRAD@orcl> INSERT INTO T1 (TNAME)
  2  SELECT OBJECT_NAME
  3  FROM DBA_OBJECTS;

93013 rows created.

Elapsed: 00:00:08.64
CONRAD@orcl>
```

2. Second Pre-12c solution (Sequence + Table)

My best execution time:

```
CONRAD@orcl> TRUNCATE TABLE T2 REUSE STORAGE;

Table truncated.

Elapsed: 00:00:00.31
CONRAD@orcl> INSERT INTO T2 (ID2, TNAME)
  2  SELECT SEQ_T2_ID2.NEXTVAL, OBJECT_NAME
  3  FROM DBA_OBJECTS;

93013 rows created.

Elapsed: 00:00:01.62
CONRAD@orcl>
```

3. First 12c solution (Sequence + Table)

My best execution time:

```
CONRAD@orcl> TRUNCATE TABLE T3 REUSE STORAGE;

Table truncated.

Elapsed: 00:00:00.34
CONRAD@orcl> INSERT INTO T3 (TNAME)
  2  SELECT OBJECT_NAME
  3  FROM DBA_OBJECTS;

93013 rows created.

Elapsed: 00:00:01.63
CONRAD@orcl>
```

4. Second 12c solution (Table)

My best execution time:

```
CONRAD@orcl> TRUNCATE TABLE T4 REUSE STORAGE;

Table truncated.

Elapsed: 00:00:00.12
CONRAD@orcl> INSERT INTO T4 (TNAME)
  2  SELECT OBJECT_NAME
  3  FROM DBA_OBJECTS;

93013 rows created.

Elapsed: 00:00:01.45
CONRAD@orcl>
```

We can see that there's a huge difference between the worst method (**1.** Pre-12c Trigger-based) and the best one (**4.** 12c with the Table only): from **8.64** seconds to **1.45** seconds.
The second **12c** solution has an Elapsed time which is about the **16.8%** of the pre-12c one **(~1/6)**.
If we consider the pre-12c second solution (explicit use of the Sequence NEXTVAL), yet, we have a little difference in the elapsed times. So, if you are not using the 12c release, avoid the Trigger-based approach - if possible - and make use of a Sequence only.

Appendix A

First of all, I create – connected as SYS - my test Tablespace and my test User:

```
C:\>sqlplus sys as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Tue Sep 3 17:40:35 2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SYS@orcl> CREATE TABLESPACE CONRAD_TAB
  2  DATAFILE 'C:\app\oracle\oradata\orcl\CONRAD_1.DBF'
  3              SIZE 20M AUTOEXTEND ON NEXT 10M
  4  EXTENT MANAGEMENT LOCAL
  5  SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SYS@orcl> CREATE USER CONRAD
  2  IDENTIFIED BY conrad
  3  DEFAULT TABLESPACE CONRAD_TAB
  4  QUOTA UNLIMITED ON CONRAD_TAB;

User created.

SYS@orcl>
```

Then, I need my privileges:

```
SYS@orcl> GRANT CREATE SESSION TO CONRAD;

Grant succeeded.

SYS@orcl> GRANT CREATE TABLE TO CONRAD;

Grant succeeded.

SYS@orcl> GRANT CREATE SEQUENCE TO CONRAD;

Grant succeeded.

SYS@orcl> GRANT CREATE TRIGGER TO CONRAD;

Grant succeeded.

SYS@orcl> GRANT SELECT ON DBA_OBJECTS TO CONRAD;

Grant succeeded.

SYS@orcl>
```

And finally I can connect as my test User:

```
SYS@orcl> conn conrad
Enter password:
Connected.
CONRAD@orcl>
```

Appendix B

Clean your test environment.

```
CONRAD@orcl> set timing off
```

If you want to drop the whole test environment:

```
SYS@orcl> DROP USER CONRAD CASCADE;
User dropped.
SYS@orcl> DROP TABLESPACE CONRAD_TAB INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
SYS@orcl>
```

Otherwise, if you want to drop the Objects only, but not the User and the Tablespace:

```
CONRAD@orcl> DROP TABLE T1 PURGE;
Table dropped.
CONRAD@orcl> DROP TABLE T2 PURGE;
Table dropped.
CONRAD@orcl> DROP TABLE T3 PURGE;
Table dropped.
CONRAD@orcl> DROP TABLE T4 PURGE;
Table dropped.
CONRAD@orcl> DROP SEQUENCE SEQ_T1_ID1;
Sequence dropped.
CONRAD@orcl> DROP SEQUENCE SEQ_T2_ID2;
Sequence dropped.
CONRAD@orcl> DROP SEQUENCE SEQ_T3_ID3;
Sequence dropped.
CONRAD@orcl>
```

And, finally, you can disconnect your session.

Corrado Piola
Owner and Database solutions Director at Systrategy Srl
c.piola@systrategy.it www.systrategy.it

Oracle Database 10g, 11g Administrator Certified Professional
Oracle Database 11g Performance Tuning Certified Expert
Oracle Database SQL Certified Expert, Oracle Advanced PL-SQL Developer Certified Professional