
Resetting Oracle Sequences After Loading Data With DataBee

A Net 2000 Ltd. Technical Note

Abstract

This technical note describes how to reset a sequence in the database to a value one greater than the maximum value of the contents of a specified tables column.

For more information, technical notes and white papers please visit the DataBee web site at <http://www.DataBee.com> For a full discussion the many issues which must be handled when creating referentially correct subsets please download the Test Database Generation and Management white paper at: <http://www.DataBee.com/TestDBGenWhitePaper.pdf>

Implementation Details

The DataBee software is designed to create referentially correct subsets of large Oracle databases. Unlike our white papers, which are generic discussions and do not require the use of the DataBee software, this document is a technical note. It is necessarily a non-generic discussion of the issue and the DataBee software is required to implement the technique discussed below.

Net 2000 Ltd.
Info@Net2000Ltd.com
<http://www.Net2000Ltd.com>

Resetting Oracle Sequences After Loading Data With DataBee

Once a target schema has been loaded with data, it often happens that sequences in the target schema must then be reset to a value appropriate to the newly loaded data. Usually this means the sequence NEXTVAL must be a value one greater than the maximum value currently present in a specified table column.

The primary difficulty in updating a sequence is due to two sources. Firstly, Oracle has no explicit “*Set the NEXTVAL of this sequence to this value*” type command so a little bit of trickery has to be implemented to achieve the desired effect. Secondly, the update of the sequence is a DDL command and determining the maximum value of a column in a table is a DML command. A combined DDL and DML instruction is notoriously difficult to implement.

Lets look at the first issue. Say, for example, we know the maximum value in the table column is 122 and the current sequence NEXTVAL is 100. We want to make sure the sequence has a next value of 123 so that the next time the sequence is used it returns an appropriate value. What we really want to do is issue a command like:

```
Update MYSEQUENCE Set NEXTVAL=123;
```

Unfortunately this is **NOT** possible – Oracle does not support this. We have to perform a series of steps as shown below:

```
alter sequence MYSEQUENCE increment by 23;
select MYSEQUENCE.nextval from dual;
alter sequence MYSEQUENCE increment by 1;
```

What we have to do is temporarily change the INCREMENT BY parameter to a value which will adjust the sequence to an appropriate value. In this example we know the current NEXTVAL is 100 so an increment by value of 23 will bring the sequence up to a value of 123 the next time it gets used. The first statement adjusts the INCREMENT BY value. The second statement ensures the select happens immediately and the final statement adjusts the INCREMENT BY value back down to 1 so that we do not continue to increment by 23 each time the sequence is used. Note that we can use a negative INCREMENT BY value if we need to decrease the NEXTVAL rather than increase it.

The second issue (that of mixing the DDL and DML commands) can be handled through the use of Dynamic SQL wrapped in a simple PL/SQL wrapper. The sample code below demonstrates the method.

```
drop sequence DTBSEQ;
create sequence DTBSEQ minvalue 1 maxvalue 9999999 increment by 1 start with 10;

drop table SEQTable;
create table SEQTable
(
    col1 number,
    col2 varchar2(10)
);

insert into SEQTable values (1,'111');
insert into SEQTable values (2,'111');
insert into SEQTable values (3,'111');
insert into SEQTable values (4,'111');
insert into SEQTable values (5,'111');
insert into SEQTable values (16,'111');
insert into SEQTable values (26,'111');
commit;
```

The code above creates a sequence and populates a simple table with data. Note that the start value of the sequence is 10 and the highest value in the table is 26. Ideally, in this situation, we would want the NEXTVAL of the sequence to be set to 27. The PL/SQL code below performs this function.

```
declare
    currentSeqNextVal number;
    currentTableHighVal number;
begin
    execute immediate 'select max(COL1) from SEQTable' INTO currentTableHighVal;
    execute immediate 'select DTBSEQ.nextval from dual' INTO currentSeqNextVal;
    IF (currentSeqNextVal > currentTableHighVal)
    THEN
        execute immediate 'alter sequence DTBSEQ increment by -' ||
(currentSeqNextVal - currentTableHighVal);
    ELSIF (currentSeqNextVal < currentTableHighVal)
    THEN
        execute immediate 'alter sequence DTBSEQ increment by ' ||
(currentTableHighVal - currentSeqNextVal);
    END IF;
    execute immediate 'select DTBSEQ.nextval from dual' INTO currentSeqNextVal;
    execute immediate 'alter sequence DTBSEQ increment by 1';
end;
```

The first statement of the PL/SQL block finds the maximum value from the COL1 column of the SEQTable table. Dynamic SQL using the Execute Immediate statement generates the appropriate code. The retrieved maximum value is stored in the currentTableHighVal variable. On the next line the currently active NEXTVAL of the DTBSEQ sequence is found and stored in the currentSeqNextVal variable. In order to cope with an increment or decrement situation an IF statement checks the relative values of the currentTableHighVal and currentSeqNextVal. This ensures that the DTBSEQ sequence INCREMENT BY value gets adjusted in the appropriate direction. The execution of the statement inside the IF construct sets the INCREMENT BY. Following on from that, outside the IF statement are the SELECT and statements to reset the INCREMENT BY value back down to 0.

With appropriate editing, the above anonymous PL/SQL block can be placed inside a loader set Command rule and any sequence can be adjusted to contain a NEXTVAL one larger than the maximum value of a specified column and table. Although Command rules can contain multiple PL/SQL statements it is probably better to create a new Command rule for each anonymous PL/SQL block. This makes error trapping much more intuitive.

IMPORTANT POINT: As with all loader set Command rules, be sure to use Rule Blocks to specifically define the point in the loader set execution sequence at which the Command rule will execute. This is an important point, if you do not understand the concept of Rule Blocks in a loader Set please contact the DataBee support team.