Coluna do tipo auto-incremento no Oracle? Abordando o uso de seqüências no Oracle
Por Eduardo Legatti
Postado en setembro 2011

Realmente, o Oracle não possui um tipo de dado "auto-incremento" como podemos ver em alguns outros bancos de dados. Por exemplo, no caso do PostgreSQL, existe um tipo de dado **SERIAL** que, na verdade, implementa números seqüenciais através de um objeto SEQUENCE. A diferença é que o PostgreSQL permite definir esta seqüência (sequence) utilizando a cláusula DEFAULT da coluna em questão. Já no Oracle, infelizmente "ainda" o mesmo não permite definir um objeto seqüência na cláusula DEFAULT da coluna como demonstrado abaixo:

```
SCOTT> create sequence seq_teste nocache;

Seqüência criada.

SCOTT> desc minha_tabela
 Nome                                      Nulo?    Tipo
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 DESCRICAO                                 NOT NULL VARCHAR2(100)

SCOTT> alter table minha_tabela modify id default seq_teste.nextval;
alter table emp modify id default seq_emp_id.nextval
*
ERRO na linha 1:
ORA-00984: coluna não permitida aqui
```

Neste caso teremos que criar uma trigger de banco de dados para realizar esta operação. Portanto, você irá aprender sobre o objeto SEQUENCE, além de observar através de um exemplo prático como criar um campo do tipo "auto-incremento" utilizando seqüências de banco de dados.

Antes de começar a falar sobre o objeto **SEQUENCE**, veremos como gerar valores seqüenciais sem a necessidade de ter que criar uma **SEQUENCE** de banco de dados.

```
SCOTT> create table emp (
  2    id number constraint pk_emp primary key,
  3    nome varchar2(60) not null);

Tabela criada.

SCOTT> insert into emp (id,nome) select nvl(max(id),0)+1,'MARIA' from emp;

1 linha criada.

SCOTT> select * from emp;

        ID NOME
---------- ------------------------
         1 MARIA

SCOTT> insert into emp (id,nome) select nvl(max(id),0)+1,'REGINA' from emp;

1 linha criada.

SCOTT> select * from emp;

        ID NOME
---------- ------------------------------------------------------------
         1 MARIA
         2 REGINA
```

Podemos ver acima, que é simples gerar dados seqüências através de um comando INSERT, **mas este método funcionaria perfeitamente apenas para sistemas monousuários**. Se o sistema for para acesso multiusuário, aí este método não é recomendável. Bem, como eu ainda não finalizei a transação acima com (COMMIT ou ROLLBACK), o que acontecerá se eu abrir uma nova seção de banco de dados e executar o mesmo comando INSERT?

SESSÃO 2

```
SCOTT> insert into emp (id,nome) select nvl(max(id),0)+1,'EDUARDO' from emp;
[Aguardando...]
```

Podemos perceber que a sessão acima está aguardando a finalização da transação iniciada pela SESSAO 1, já que o registro na tabela EMP foi locado pela mesma. Abaixo irei retornar para a SESSAO 1 e finalizar a transação com COMMIT:

SESSÃO 1

```
SCOTT> commit;
Commit concluído.
```

Ao retornar para a SESSAO 2 mostrada abaixo, podemos perceber que houve uma violação de chave primária na tabela EMP.

SESSÃO 2

```
SCOTT> insert into emp (id,nome) select nvl(max(id),0)+1,'EDUARDO' from emp;
insert into emp (id,nome) select nvl(max(id),0)+1,'EDUARDO' from emp
*
ERRO na linha 1:
ORA-00001: restrição exclusiva (SCOTT.PK_EMP) violada
```

Isto aconteceu pelo fato de a SESSAO 2 não ter conhecimento da transação concorrente iniciada pela SESSAO 1. Isto se refere à proprie
(ISOLAMENTO) de um termo conhecido como **ACID**, na qual todos os bancos de dados relacionais devem atender:

* ATOMICIDADE - Tudo (commit) ou nada (rollback)
* CONSISTÊNCIA - Sem violação de restrições de integridade
* ISOLAMENTO - Alterações concorrentes invisíveis
* DURABILIDADE - Persistência das atualizações confirmadas

Embora diversas transações possam ser executadas de forma concorrente, o sistema gerenciador de banco de **dados deve garantir a consistência de leitura**. É aí que entram em ação os segmentos de rollback ou UNDO...

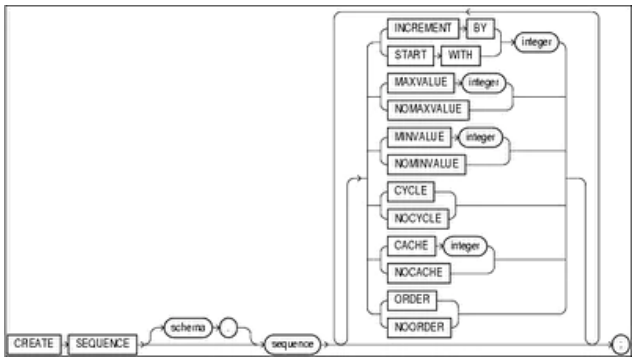
```
SCOTT> truncate table emp;

Tabela truncada.
```

Após a demonstração acima, veremos abaixo uma introdução breve sobre uso do objeto SEQUENCE disponível no banco de dados Oracle.

Então, o que é um objeto seqüência? Uma seqüência (**sequence**) é um objeto de banco de dados criado pelo usuário que pode ser compartilhado por vários usuários para gerar números inteiros exclusivos de acordo com regras especificadas no momento que a seqüência é criada. A seqüência é gerada e incrementada (ou diminuída) por uma rotina interna do Oracle. Normalmente, as seqüências são usadas para criar um valor de chave primária que deve ser exclusivo para cada linha de uma tabela.

Vale a pena salientar que os números de seqüências são armazenados e gerados de modo independente das tabelas. Portanto, o mesmo objeto seqüência pode ser usado por várias tabelas e inclusive por vários usuários de banco de dados caso necessário. Geralmente, convém atribuir à seqüência um nome de acordo com o uso a que se destina; no entanto, ela poderá ser utilizada em qualquer lugar, independente do nome.

No mais, se você só precisa da chave para garantir a singularidade não se importando em criar uma chave com nome sem muito sentido, então é perfeitamente apropriado fazê-lo se utilizando de seqüências. As seqüências são criadas com o comando CREATE **SEQUENCE** ... na qual poderemos nos utilizar de algumas cláusulas durante a sua criação.



As cláusulas mais importantes são:

start with n- Permite especificar o primeiro valor a ser gerado pela seqüência. Uma vez criada, a seqüência irá gerar o valor especificado por start with na primeira vez que a coluna virtual NEXTVAL da seqüência for referenciada. Se nenhum valor start with for especificado, então a seqüência assumirá o valor padrão 1.
increment by n - Define o número a ser incrementado cada vez que a coluna virtual NEXTVAL for referenciada. O valor padrão para esta coluna é 1, se não for explicitamente especificado. Você pode definir (n) com um valor positivo para seqüências crescentes, ou com um valor negativo para seqüências decrescentes de forma a gerar valores regressivos.
minvalue n - Define o valor mínimo que pode ser produzido pela seqüência. Se nenhum valor mínimo for especificado, o Oracle irá assumir o padrão nominvalue 1 para uma seqüência crescente e (10^27) para uma seqüência decrescente.
maxvalue n - Define o valor máximo que pode ser produzido pela seqüência. Se nenhum valor máximo for especificado, o Oracle irá assumir o padrão nomaxvalue (10^27) para uma seqüência crescente e 1 para uma seqüência decrescente.
cycle - Especifica se a seqüência continuará a gerar valores após alcançar seu valor máximo ou mínimo. Se esta cláusula não for especificada explicitamente, o Oracle irá assumir o valor padrão nocycle. Você não pode especificar cycle em conjunto com nomaxvalue ou nominvalue. Se você quiser que sua seqüência use um ciclo, você deverá especificar maxvalue para seqüências crescentes, ou minvalue para seqüências decrescentes.
cache n - Especifica quantos valores o servidor Oracle alocará previamente na memória (SGA). O uso desta cláusula permite à seqüência gerar antecipadamente um número especificado de valores, usando um cache para melhorar o desempenho. Se o cache não for especificado explicitamente, o Oracle irá assumir o padrão, que é de gerar um cache de 20 valores.
Obs: *Não use a opção CYCLE se a seqüência for utilizada para gerar valores de chave primária, a menos que você tenha um mecanismo confiável que expurgue linhas mais rapidamente que os ciclos da seqüência.*

Considere agora um exemplo de definição de seqüências. Como dito anteriormente, os números inteiros que podem ser especificados para seqüências, podem ser tanto negativos quanto positivos. O exemplo abaixo usa uma seqüência decrescente onde o número inteiro de start with é positivo, mas o número inteiro de increment by é negativo, o que na prática diz à seqüência para decrescer, ao invés de crescer. Quando o valor zero for atingido, a seqüência começará novamente a contagem.

```
SCOTT> create sequence seq_decrescente_5
    2  start with 5
    3  increment by -1
    4  maxvalue 5
    5  minvalue 0
    6  nocache
    7  cycle;

Seqüência criada.
```

Uma vez criada a seqüência, ela poderá ser utilizada usando-se como referência as pseudo-colunas **CURRVAL e NEXTVAL**. Os usuários do banco de dados podem visualizar o valor atual da seqüência usando um comando SELECT. Da mesma forma, o próximo valor da seqüência pode ser gerado com um comando SELECT. Podemos ver abaixo como a seqüência **SEQ_DECRESCENTE_5** faz um ciclo quando o valor de **minvalue** é atingido:

```
SCOTT> select seq_decrescente_5.nextval from dual;

   NEXTVAL
----------
         5

SCOTT> /

   NEXTVAL
----------
         4
```

```
SCOTT> /

   NEXTVAL
----------
         3

SCOTT> /

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

SCOTT> /

   NEXTVAL
----------
         1

SCOTT> /

   NEXTVAL
----------
         0

SCOTT> /

   NEXTVAL
----------
         5
```

Uma vez referenciada a pseudo-coluna **NEXTVAL**, o valor em **CURRVAL** é atualizado para bater com o valor de **NEXTVAL**, e o valor anterior em **CURRVAL** é descartado:

```
SCOTT> select seq_decrescente_5.currval from dual;

   CURRVAL
----------
         5

SCOTT> select seq_decrescente_5.nextval from dual;

   NEXTVAL
----------
         4

SCOTT> select seq_decrescente_5.currval from dual;

   CURRVAL
----------
         4

SCOTT> exit
Desconectado de Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production
```

Podemos perceber então que a pseudo-coluna **NEXTVAL** é usada para extrair números sucessivos de uma seqüência especificada. Será sempre necessário qualificar **NEXTVAL** com o nome da seqüência. Quando fazemos referência à **NEXTVAL**, um novo número de seqüência é gerado e o número de seqüência atual é colocado em **CURRVAL**. Vale a pena salientar que a pseudo-coluna **CURRVAL** é usada para fazer referência a um número de seqüência que a sessão do usuário atual acabou de gerar. Portanto, **NEXTVAL** deve ser usada para gerar um número de seqüência na sessão do usuário atual antes que seja feita referência a **CURRVAL**, caso contrário, o erro abaixo será emitido:

```
C:\>sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Seg Nov 3 15:39:28 2008

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Conectado a:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SCOTT> select seq_decrescente.currval from dual;
select seq_decrescente.currval from dual
*
ERRO na linha 1:
ORA-08002: a seqüência SEQ_DECRESCENTE.CURRVAL ainda não foi definido nesta sessão
```

Quer dizer que se quisermos verificar o último valor de seqüência gerado, sempre teremos que inicializar a seqüência incrementado seu valor através da paseudo-coluna NEXTVAL? **Não necessariamente. Caso tenhamos criado uma seqüência com a opção NOCACHE, poderemos verificar de forma precisa, qual foi o último valor de seqüência gerado e qual será o próximo valor a ser gerado, verificando o valor da coluna LAST_NUMBER nas views de dicionário de dados DBA/ALL/USER_SEQUENCES**. Uma vez criada, a seqüência é documentada no dicionário de dados. Já que uma seqüência é um objeto de banco de dados, poderemos identificá-la nas views de dicionário de dados DBA/ALL/USER_OBJETCS, como também ver as definições da mesma nas views DBA/ALL/USER_SEQUENCES. O exemplo abaixo demonstra a diferença de utilizar a opção **CACHE e NOCACHE**:

```
SCOTT> create sequence seq_cache;

Seqüência criada.

SCOTT> create sequence seq_nocache nocache;

Seqüência criada.



SCOTT> select seq_cache.nextval from dual;

   NEXTVAL
----------
```

```
         1

SCOTT> /

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

SCOTT> select seq_nocache.nextval from dual;

   NEXTVAL
----------
         1

SCOTT> /

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

SCOTT> select * from user_sequences;

SEQUENCE_NAME MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------- --------- ---------- ------------ - - ---------- -----------
SEQ_CACHE             1 1,0000E+27            1 N N         20          21
SEQ_NOCACHE           1 1,0000E+27            1 N N          0           3
```

Consultado a view de dicionário de dados **USER_SEQUENCES**, podemos ver acima que a seqüência SEQ_CACHE não nos mostra de forma precisa através da coluna LAST_NUMBER qual foi o último valor de seqüência gerado pela mesma, **pelo fato destes valores estarem em uma cache de memória na SGA (System Global Area)**. Na verdade, o cache é preenchido na primeira vez que fazemos referência à seqüência. Cada solicitação de próximo valor é recuperada da seqüência neste cache. Depois que o último valor é usado, a próxima solicitação da seqüência armazena outro cache de seqüências na memória.

Já a seqüência SEQ_NOCACHE nos mostra de forma precisa através da coluna LAST_NUMBER que o próximo valor de seqüência a ser gerado será 3, nos indicando que o último valor gerado foi 2. **Portanto, a coluna LAST_NUMBER exibirá o próximo número de seqüência disponível apenas se NOCACHE for especificado durante a criação de uma seqüência**.

Embora os geradores se seqüência emitam números de seqüência sem intervalos, essa ação ocorre independente de um COMMIT ou ROLLBACK. Portanto, se nós fizermos o rollback de uma instrução que contenha uma seqüência, o número será perdido. Outro evento que pode causar intervalos na seqüência é uma falha no sistema. Se a seqüência colocar valores no cache de memória, esses valores serão perdidos em caso de falha do sistema.

Abaixo será criada uma seqüência para demonstrar como poderemos utilizá-la em um comando SQL.

```
SCOTT> create sequence seq_emp_id nocache;

Seqüência criada.

SCOTT> insert into emp values (seq_emp_id.nextval,'ROBERTO');

1 linha criada.

SCOTT> insert into emp values (seq_emp_id.nextval,'LAURA');

1 linha criada.

SCOTT> insert into emp values (seq_emp_id.nextval,'GUSTAVO');

1 linha criada.

SCOTT> select * from emp;

        ID NOME
---------- --------------------------------------------------
         1 ROBERTO
         2 LAURA
         3 GUSTAVO

SCOTT> select * from user_sequences;

SEQUENCE_NAME MIN_VALUE MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------- --------- --------- ------------ - - ---------- -----------
SEQ_EMP_ID            1 1,0000E+27           1 N N          0           4
```

Após toda essa demonstração, iremos criar uma trigger de banco de dados (Before Insert) na tabela EMP de forma que a coluna ID tenha seus valores gerados de forma automática (simulando um tipo de dado auto-incremento) ao inserir dados na mesma.

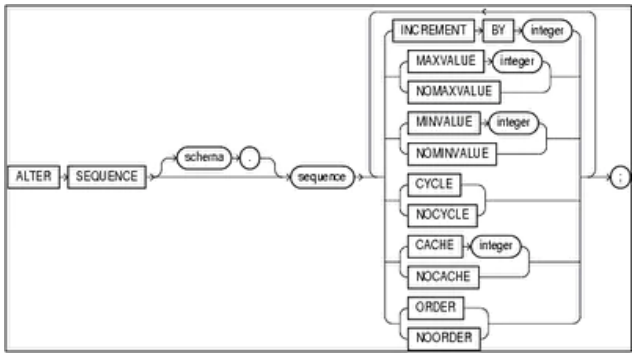
```
SCOTT> create or replace trigger gera_emp_id
  2  before insert on emp
  3  for each row
  4  begin
  5   select seq_emp_id.nextval into :new.id from dual;
  6  end;
  7  /

Gatilho criado.
```

Iremos abaixo realizar algumas inserções na tabela EMP sem referenciar a coluna ID.

```
SCOTT> insert into emp (nome) values ('RENATA');

1 linha criada.

SCOTT> insert into emp (nome) values ('CARLOS');

1 linha criada.
```

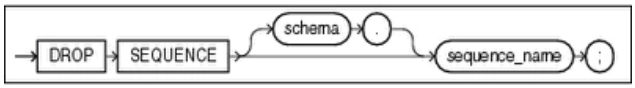
```
SCOTT> select * from emp;

        ID NOME
---------- --------------------------------------------------
         1 ROBERTO
         2 LAURA
         3 GUSTAVO
         4 RENATA
         5 CARLOS

SCOTT> select * from user_sequences;

SEQUENCE_NAME MIN_VALUE MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------- --------- --------- ------------ - - ---------- -----------
SEQ_EMP_ID           1 1,0000E+27            1 N N          0           6
```

E se quisermos alterar o valor atual da seqüência? Vale a pena salientar que esta ação somente será possível **se a seqüência for dropada e recriada**, pois não é possível alterar a propriedade **START WITH** com o comando **ALTER SEQUENCE** ...:




```
SCOTT> alter sequence seq_emp_id start with 1;
alter sequence seq_emp_id start with 1
*
ERRO na linha 1:
ORA-02283: não é possível alterar o número inicial da seqüência
```

Por fim, se quisermos dropar a seqüência, basta apenas emitirmos comando DROP SEQUENCE ... como mostrado abaixo:




```
SCOTT> drop sequence seq_emp_id;

Seqüência eliminada.
```

A propósito, para quem ainda não conhece, vale a pena salientar que o Oracle possui uma função interna chamada **SYS_GUID** (globally unique identifier) que gera e retorna valores únicos de 32 caracteres em uma representação hexadecimal. Caso alguém queira garantir alguma unicidade...

```
-- Gerando 10 valores únicos
SCOTT> select sys_guid() from dual connect by level <=10;

SYS_GUID()
--------------------------------
5AC83A4AF1E1B4CBE040EEC868FB0EE8
5AC83A4AF1E2B4CBE040EEC868FB0EE8
5AC83A4AF1E3B4CBE040EEC868FB0EE8
5AC83A4AF1E4B4CBE040EEC868FB0EE8
5AC83A4AF1E5B4CBE040EEC868FB0EE8
5AC83A4AF1E6B4CBE040EEC868FB0EE8
5AC83A4AF1E7B4CBE040EEC868FB0EE8
5AC83A4AF1E8B4CBE040EEC868FB0EE8
5AC83A4AF1E9B4CBE040EEC868FB0EE8
5AC83A4AF1EAB4CBE040EEC868FB0EE8

10 linhas selecionadas.
```

Postado por Eduardo Legatti (http://eduardolegatti.blogspot.com), Analista de Sistemas e DBA Oracle. É pós graduado em Gerência da Tecnologia da Informação, possui as certificações OCA 9i - OCP 9i/10g – OCE, e vem trabalhando como DBA Oracle desde a versão 8.0.5.

E-mail this page Printer View

Entre em Contato Conosco
Vendas: 0800-891-4433
Contatos Globais
Diretório de Suporte

Nuvem
Visão geral de Soluções em Nuvem
Software (SaaS)
Plataforma (PaaS)

Ações Principais
Faça o Download do Java
Faça o Download do Java para Desenvolvedores

Tópicos Principais
ERP, EPM (Finanças)
HCM (RH, Talento)
Marketing