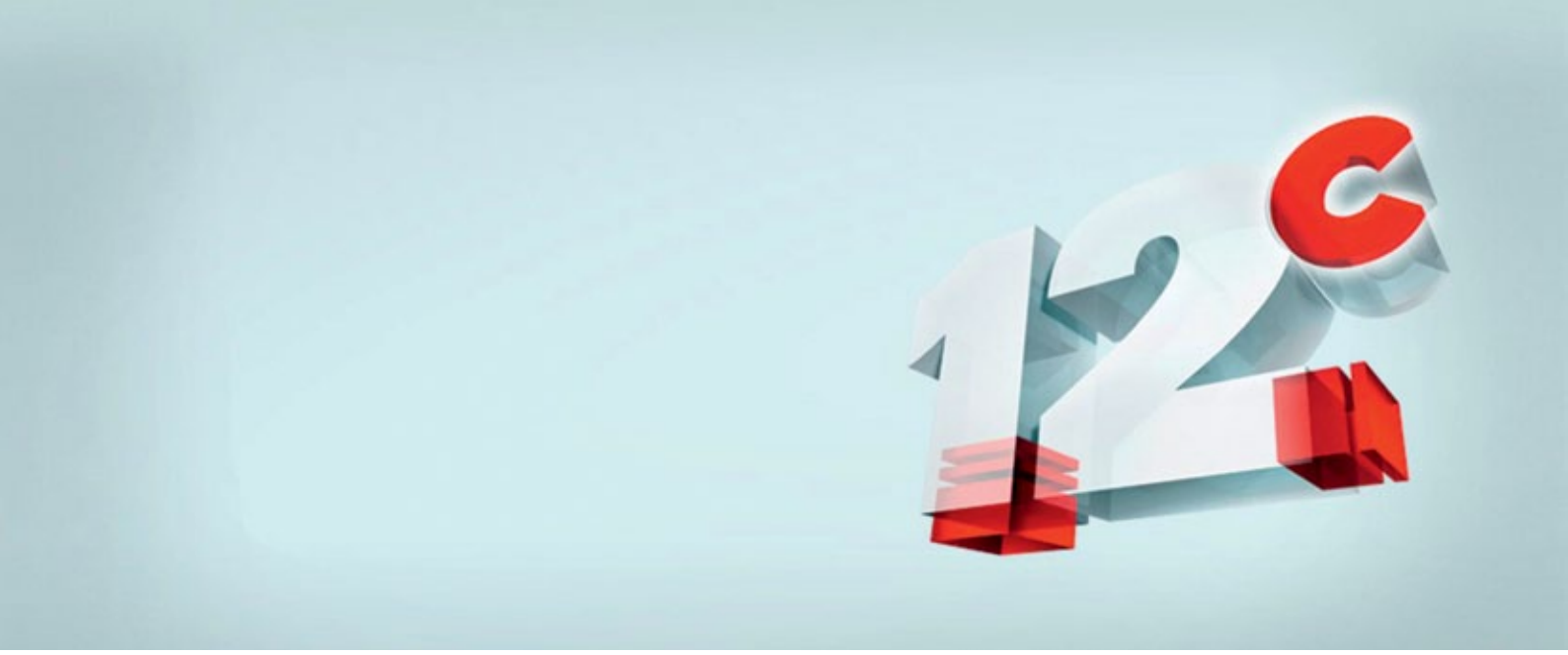

Neue Funktionen für Entwickler

Carsten Czarski, ORACLE Deutschland B.V. & Co. KG

Das neue Datenbank-Release bringt insgesamt mehr als 500 neue Funktionen mit. Auch für SQL- und PL/SQL-Entwickler sind mit SQL Pattern Matching, Identity Columns, unsichtbaren Spalten und anderen neuen Funktionen eine Menge nützlicher Dinge dabei. Dieser Artikel gibt einen Einblick in die Neuerungen.

32K VARCHAR überall

„VARCHAR2"-Spalten einer Tabelle können ab Oracle 12*c* bis zu 32.767 Bytes aufnehmen und verhalten sich damit genauso wie Variablen dieses Typs in PL/SQL. Wenn man also abschätzen kann, dass die benötigte Länge unterhalb von 32 KB bleibt, kann man sich den (komplizierteren) Einsatz von CLOBs sparen (siehe Listing 1).

Ganz ohne den DBA geht es dann aber doch nicht. Damit alles funktioniert, muss dieser den Datenbank-Parameter „MAX_STRING_SIZE" auf „EXTENDED" setzen und pro Datenbank einmal das Skript „$ORACLE_HOME/rdbms/admin/utl32k.sql" ablaufen lassen. Ausführliche Informationen dazu stehen in der Oracle Database Reference unter „MAX_STRING_SIZE". In einer Apex-Umgebung sollte danach zusätzlich das Skript „$Apex_HOME/core/collection_member_resize.sql" gestartet werden, damit das neue Limit auch in Apex Collections möglich ist.

Sequences „By Default" und Identity Columns

Der Umgang mit Sequences und das Generieren eindeutiger Werte für eine Primärschlüssel-Spalte werden in Oracle 12*c* wesentlich einfacher. So ist es nun möglich, den nächsten Wert der Sequence als Default für eine Tabellenspalte zu hinterlegen. Das bislang nötige und lästige Erstellen des Triggers gehört damit der Vergangenheit an (siehe Listing 2).

Es geht sogar noch einfacher: Die „GENERATED AS IDENTITY"-Klausel nimmt die Definition der Sequence zur Tabelle. Das explizite Erzeugen einer Sequence fällt damit weg (siehe Listing 3).

Die „GENERATED ... AS IDENTITY"-Klausel kennt noch einige Variationen. Die dargestellte Variante „GENERATED ALWAYS" nimmt immer den Wert der Sequence für die Tabellenspalte, auch wenn im „SQL INSERT" explizit ein Wert vorgegeben wurde. Alterna-

```
create table meine_tabelle(
  id         number(10),
  langertext varchar2(10000)
)
```

Listing 1

```
create sequence seq_id start with 1 increment by 1;

create table meine_tabelle(
  id      number(10)      default seq_id.nextval,
  text    varchar2(10000)
);
```

Listing 2

tiv lässt sich ein „GENERATED BY DEFAULT“ oder „BY DEFAULT ON NULL“ spezifizieren (siehe Abbildung 1).

Invisible Columns

Tabellenspalten können in Oracle 12*c* quasi unsichtbar gemacht werden – dazu dient das neue Spaltenattribut „INVISIBLE“ (siehe Listing 4).

Die Tabellenspalten (hier „HIREDATE“ und „COMM“) sind tatsächlich nicht mehr sichtbar. Invisible Columns sind dann sinnvoll, wenn es darum geht, dass andere Nutzer mit einer Spalte nicht arbeiten sollten. Das könnten Spalten sein, die allein für Partitionierung angelegt wurden, deren Inhalte nur für bestimmte PL/SQL-Pakete von Belang sind oder ähnliche Gründe. Eine Spalte unsichtbar zu machen, ist allerdings kein Schutz vor Zugriffen: Spricht man die Spalte explizit an, kann man normal damit arbeiten (siehe Listing 5).

Übrigens: Intern kennt Oracle die unsichtbaren Spalten schon lange – sobald Objekt-Typen zum Einsatz kommen, werden die Tabellen mit unsichtbaren Spalten versehen – an der Spalte „INVISIBLE“ in der Dictionary-View „USER_TAB_COLS“ lässt sich das leicht nachvollziehen.

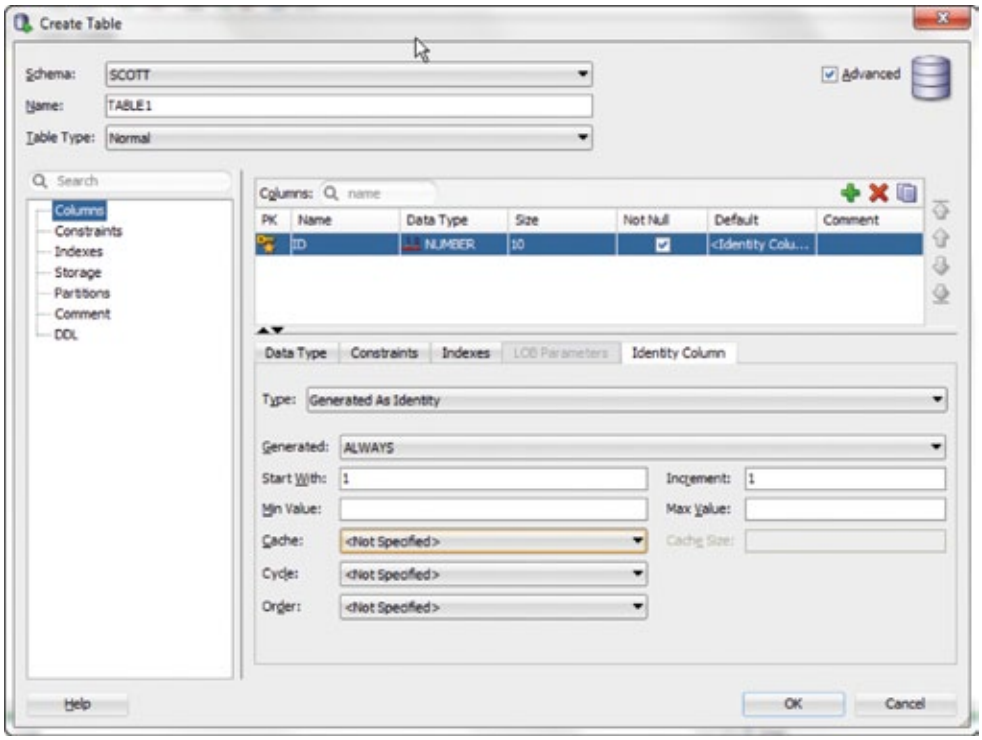

Abbildung 1: Oracle12c Identity Columns im SQL Developer

Weitere Verbesserungen in SQL und PL/SQL

Zwei neue SQL-Klauseln für Abfragen machen das Blättern in einem Bericht wesentlich einfacher: „LIMIT“ sorgt dafür, dass nur eine bestimmte Anzahl von Zeilen abgerufen wird, „OFFSET“ bewirkt, dass vorher eine bestimmte Anzahl übersprungen wird. Was man bislang aufwändig mit Unterabfragen und „ROWNUM“ gemacht hat, ist nun wesentlich einfacher und klarer. Die Abfrage in Listing 6 selektiert demnach die Zeilen aus der Tabelle „EMP“ mit dem viert-, fünft- und sechsthöchsten Gehalt.

In der WITH-Klausel einer SQL-Abfrage können nun Funktionen definiert werden; diese gelten nur für diese Abfrage – es gibt keine Einträge im Data Dictionary (siehe Listing 7).

Die „ACCESSIBLE BY“-Klausel legt fest, dass ein PL/SQL-Package, eine PL/SQL-Prozedur oder -Funktion nur von einem (oder mehreren) anderen PL/SQL-Objekt(en) – und nicht direkt – aufgerufen werden kann. Das ist besonders nützlich für „Helper“-Packages, die nicht zum direkten Aufruf und nur zur Nutzung durch andere Packages vorgesehen sind. Die neue Klausel verhindert das versehentliche (oder absichtliche) Aufrufen und dadurch eventuell entstehende Seiteneffekte (siehe Listing 8).

SQL Pattern Matching

Eine der interessantesten Neuerungen im Bereich der SQL-Funktionen ist das

```
create table meine_tabelle(
  id      number(10)
          generated always as identity start with 1 increment by 1,
  text    varchar2(10000)
);
```

Listing 3

```
SQL> alter table EMP modify (COMM invisible, HIREDATE invisible);

SQL> desc EMP
 Name                          Null?    Typ
 ----------------------------- -------- --------------------
 EMPNO                         NOT NULL NUMBER(4)
 ENAME                                  VARCHAR2(10)
 JOB                                    VARCHAR2(9)
 MGR                                    NUMBER(4)
 SAL                                    NUMBER(7,2)
 DEPTNO                                 NUMBER(2)
```

Listing 4

```
SQL> select ename, hiredate, sal, comm from emp

ENAME                HIREDATE                   SAL       COMM
-------------------- ------------------ ---------- ----------
SMITH                17.12.1980 00:00:00        800
ALLEN                20.02.1981 00:00:00       1600        300
WARD                 22.02.1981 00:00:00       1250        500
:                    :                            :          :
```

Listing 5

```
select empno, ename, sal
from emp
order by sal asc
offset 3 rows fetch first 3 rows only;

     EMPNO ENAME                          SAL
---------- -------------------- ----------
      7521 WARD                       1250
      7654 MARTIN                     1250
      7934 MILLER                     1300
```

Listing 6

```
with
  function half_sal(p_sal in number) return number is
  begin
    return p_sal/2;
  end;
select empno, sal, half_sal(sal) from emp;

         EMPNO        SAL HALF_SAL(SAL)
    ---------- ---------- -------------
          7369        800           400
          7521       1250           625
          7566       2975        1487,5
```

Listing 7

SQL Pattern Matching. Dieser Satz an SQL-Funktionen erlaubt es, Muster in Tabellendaten zu finden – und hier geht es um Muster, die sich über mehrere Tabellenzeilen hinweg ergeben. Ein Beispiel für solche Muster wären Web-Sessions, die durch Klicks in einer Log-Datei repräsentiert werden. Die Log-Datei enthält nur die einzelnen Klicks (siehe Listing 9).

Wie man sehen kann, enthält die Log-Tabelle nur Einzelaufnahmen – eben das Abrufen einer Webseite, was auf einem Klick auf einen Link basieren kann. Eine Web-Session wird durch mehrere Zeilen in dieser Tabelle repräsentiert – explizit ist die Information nicht enthalten. Man muss die Sessions aus den Daten rekonstruieren. Dass die „CLIENT_IP“ in einer Session dieselbe sein muss, ist klar. Das Ende der Session ist jedoch nicht direkt aus den Daten ableitbar – für das Beispiel wird daher angenommen, dass die Session endet, wenn die Zeitspanne bis zum nächsten Klick eine bestimmte Dauer (5 Minuten) übersteigt (natürlich vereinfacht dieses Beispiel stark). Die Zeilen, die zu einer Web-Session gehören, müssen also wie folgt gefunden werden:

- Zunächst nach „CLIENT_IP“ und „ZEITSTEMPEL“ sortieren
- Die allererste Zeile markiert den Beginn der ersten Session
- Wenn die nächste Zeile die gleiche „CLIENT_IP“ enthält und die Differenz der Zeitstempel nicht mehr als fünf Minuten beträgt, dann gehört die nächste Zeile zur gleichen, ansonsten ist sie der Beginn der nächsten Session

Man kann sich nun vorstellen, dass es nicht einfach ist, eine solche Aufgabe mit Standard-SQL zu lösen. Die neue SQL-Funktion „MATCH_RECOGNIZE“ erleichtert die Aufgabe (siehe Listing 10).

In guter alter SQL-Manier wird der Datenbank nicht mehr gesagt, wie die Aufgabe zu erledigen ist, sondern dass ein „Pattern Matching“ durchgeführt werden soll. Die nötigen Angaben erfolgen deklarativ. Als Ergebnis liefert diese SQL-Abfrage eine Übersicht über die gefundenen Sessions zurück (siehe Listing 11).

Java in der Datenbank

Es ist bekannt, dass die Oracle-Datenbank Stored Procedures und Functions nicht nur in PL/SQL ausführen kann, sondern auch in Java. Seit der Datenbank-Version 8*i* (das war im Jahr 2000) ist eine Java-Engine Teil der Datenbank. Das gilt für alle Datenbank-Editionen ab der Standard-Edition (also nicht XE). Mit Java in der Datenbank lassen sich viele Dinge erreichen, die mit PL/SQL nicht machbar sind:

- E-Mails abrufen
- Die Datenbank als FTP-Client einsetzen
- Dateisystem-Zugriffe und Betriebssystem-Kommandos ausführen

In Oracle 12*c* wurde die Java-Engine auf Java SE6 aktualisiert. Erstmalig ist es auch möglich, selbstständig ein Upgrade der Datenbank-JVM in der gleichen Datenbank durchzuführen. Wer also Java 7 braucht, findet in der Dokumentation „Java Developers‘ Guide“ eine Anleitung zum Upgrade der JVM.

```
create or replace function lowlevel return varchar2
accessible by (toplevel) is
begin
  return 'lowlevel() called.';
end lowlevel;
/

create or replace function toplevel return varchar2 is
begin
  return 'Call via function: ' || lowlevel;
end toplevel;
/

SQL> select toplevel from dual;

TOPLEVEL
-------------------------------------
Call via function: lowlevel() called.
SQL> select lowlevel from dual;

FEHLER in Zeile 1:
ORA-06552: PL/SQL: Statement ignored
ORA-06553: PLS-904: insufficient privilege to access object
LOWLEVEL
```

Listing 8

```
10.165.244.126 - - [05/May/2010:12:33:02 +0200] "GET / HTTP/1.1" 200 10003
10.165.244.126 - - [05/May/2010:12:33:02 +0200] "GET /spatialde_logo.png HTTP/1.1" 200 2094
10.165.244.126 - - [05/May/2010:12:33:02 +0200] "GET /oraclemaps.png HTTP/1.1" 200 113295
10.165.244.126 - - [05/May/2010:12:34:02 +0200] "GET / HTTP/1.1" 200 10003
:                 :                              :
```

Listing 9

```
SELECT client_ip, start_tstamp, end_tstamp, sess_clicks
FROM ohs_access_log
MATCH_RECOGNIZE (
  PARTITION BY client_ip
  ORDER BY zeitpkt asc
  MEASURES  STRT.zeitpkt AS start_tstamp,
            LAST(NXT.zeitpkt) AS end_tstamp,
            FINAL COUNT(nxt.zeitpkt) as sess_clicks
  ONE ROW PER MATCH
  AFTER MATCH SKIP PAST LAST ROW
  PATTERN (STRT NXT+)
  DEFINE
   NXT AS NXT.zeitpkt < PREV(NXT.zeitpkt) + INTERVAL '300'
SECOND
) MR
where sess_clicks > 2
order by client_ip, start_tstamp
```

Listing 10

```
10.165.115.30   26.04.11 11:06:13 26.04.11 12:42:14    166
10.165.115.9    26.04.11 08:11:30 26.04.11 08:11:36     47
:               :                                  :
```

Listing 11

Fazit

Entwickler können massiv von den neuen Funktionen in Oracle 12*c* profitieren. Viele sehen auf den ersten Blick zwar recht einfach aus, der Nutzen der „IDENTITY COLUMNS", der „LIMIT"- beziehungsweise „OFFSET"-Clauses oder des SQL Pattern Matching ist nicht zu unterschätzen: Bislang mussten solche Aufgaben mit mehr oder weniger aufwändigem Coding umgesetzt werden – das bedeutet jedoch nicht nur Mehraufwand beim Entwickeln, sondern auch bei der Wartung oder bei der Übergabe an neue Mitarbeiter im Projekt. Diese müssen den Code schließlich verstehen und ein einfaches „MATCH_RECOGNIZE" ist leichter zu verstehen als selbstgeschriebener PL/SQL-Code.

Natürlich kann dieser Artikel nur einen kleinen Ausschnitt aus den neuen Funktionen betrachten; umfassendere Informationen stehen auf der Webseite der deutschsprachigen Apex- und PL/SQL-Community oder in der Oracle-Dokumentation.

Weitere Informationen

1. Deutschsprachige Apex- und PL/SQL-Community: Oracle12c New Features: https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_TIPP?P_ID=941
2. Oracle12c Dokumentation, New Features Guide: http://docs.oracle.com/cd/E16655_01/server.121/e17906/toc.htm

Carsten Czarski
carsten.czarski@oracle.com
http://sql-plsql-de.blogspot.com