Willkommen zum „IBM DB2 Weihnachts-Newsletter“

Liebe Leserinnen und Leser,

Weihnachten steht für mich unmittelbar bevor und bis Sie den Newsletter lesen is es auch schon wieder vorbei. Von weißer Weihnacht ist nichts zu sehen. Der Schnee kommt und geht, wie auch die Feiertage. Momentan ist mal wieder Hochwasser angesagt. Und bei Temperaturen um die 13 Grad, könnte man fast die Badehose wieder auspacken.

9 kleine Weihnachtsmänner stehen hier so rum.
Für 9 kleine Weihnachtsmänner ist das gar nicht dumm.
Weihnachten steht vor der Tür,
ein Frohes Fest - das wünschen sie Dir.

Falls Sie zwischen den Feiertagen arbeiten wollen, können und dürfen, haben wir wieder interessante Beiträge für Sie zusammengetragen.
Viel Spaß beim Ausprobieren.

Sollten Sie ein wenig Zeit haben und eine Idee für einen Beitrag für den DB2 Newsletter, dann würden wir uns freuen, wenn diese den Weg zu uns finden.

Für Fragen und Anregungen unsere Kontaktadresse: db2news@de.ibm.com.

Ihr TechTeam

Inhaltsverzeichnis

TechTipp: Transponieren

Wenn man eine Tabelle in DB2 hat und möchte sie transponieren, ist das nicht ganz einfach. Hier ein kleiner "Kniff", mit dem man das lösen kann.

Beispiel:
Angenommen man hat eine Tabelle T1 mit zwei Spalten (z.B ID und FELD).

```
select * from T1

ID FELD
---- ----
1  A
1  B
1  C
2  A
2  C
3  B

  6 record(s) selected.
```

Transponieren bedeutet, dass die Tabelle "gedreht" wird. Also pro ID eine Spalte für jede Ausprägung im Feld. Man muss allerdings wissen, wie viele Ausprägungen es geben kann.
Hier in dem Beispiel sind das drei (A, B und C).

```
ID      FELD1          FELD2   FELD3
-------------------------------------------------------------
1       A                      B              C
2       A                      -              C
3       -                      B              -
```

Die einfachste Variante diese Aufgabe zu lösen wäre:

```
select ID,
       min(case when feld='A' then 'A' else null end) feld_A,
       min(case when feld='B' then 'B' else null end) feld_B,
       min(case when feld='C' then 'C' else null end) feld_C
  from T1
 group by ID"

ID          FELD_A FELD_B FELD_C
----------- ------ ------ ------
          1 A      B      C
          2 A      -      C
          3 -      B      -

  3 record(s) selected.
```

In diesem Beispiel müsste man jedoch nicht nur wissen, wie viele Ausprägungen vorhanden sind, sondern auch noch den konkreten Inhalt.
Daher ist die Variante mit der OLAP Funktion besser. Zunächst wird mittels der COMMON TABLE EXPRESSION (CTE) eine Hilfstabelle erzeugt, die dem Feldnamen eine ROW NUMBER zuordnet.

```
    SELECT  FELD
          , ROWNUMBER() OVER(ORDER BY FELD) AS rn
    FROM (SELECT DISTINCT FELD
          FROM T1
         ) AS R
FELD RN
---- --------------------
A                       1
B                       2
C                       3
```

Anschließend wird diese Hilfstabelle mit der eigentlichen Tabelle verknüpft. Danach werden mit dem CASE-Statement die Werte den Spalten zugeordnet. Das SQL sieht dann wie folgt aus:

```
WITH Hilfstabelle AS
(
    SELECT  FELD
          , ROWNUMBER() OVER(ORDER BY FELD) AS rn
    FROM (SELECT DISTINCT FELD
          FROM T1
         ) AS R
)
SELECT a.ID
,CASE WHEN rn = 1 THEN a.FELD END f1
,CASE WHEN rn = 2 THEN a.FELD END f2
,CASE WHEN rn = 3 THEN a.FELD END f3
FROM T1 A
   , Hilfstabelle H
WHERE a.FELD = H.FELD
;
```

und liefert das Ergebnis:

```
ID F1 F2 F3
-- -- -- --
1  A  -  -
2  A  -  -
1  -  B  -
3  -  B  -
1  -  -  C
2  -  -  C
```

Das ist fast schon das gewünschte Ergebnis. Nun müssen nur die Zeilen pro ID zusammengefasst werden. Das kann z.B. mit der MIN Funktion gemacht werden. Das Ergebnis ist dann:

```
WITH Hilfstabelle AS
(
    SELECT  FELD
          , ROWNUMBER() OVER(ORDER BY FELD) AS rn
    FROM (SELECT DISTINCT FELD
          FROM T1
         ) AS R
)

SELECT a.ID
, MIN(CASE WHEN rn = 1 THEN a.FELD END) FELD1
, MIN(CASE WHEN rn = 2 THEN a.FELD END) FELD2
, MIN(CASE WHEN rn = 3 THEN a.FELD END) FELD3
FROM T1 A
   , Hilfstabelle H
WHERE a.FELD = H.FELD
GROUP BY A.ID
;

und liefert das gewünschte Ergebnis.

ID FELD1 FELD2 FELD3
-- ----- ----- ----- -----
1  A     B     C
2  A     -     C
3  -     B     -

  3 record(s) selected.
```

Anmerkung der Redaktion:
Statt des MIN/CASE Konstruktes kann auch mit MAX/DECODE gearbeitet werden.
Die Variante, bei der alle Dateninhalte bekannt sein müssen, sieht wie folgt aus:

```
select id,
       max(decode(feld,'A',feld)) as FELD1,
       max(decode(feld,'B',feld)) as FELD2,
       max(decode(feld,'C',feld)) as FELD3
  from T1
 group by ID"

ID          FELD1 FELD2 FELD3
----------- ----- ----- -----
          1 A     B     C
          2 A     -     C
          3 -     B     -
```

```
  3 record(s) selected.
```

Die Variante, die mittels der Hilfstabelle den Inhalt der Tabelle bewertet, sieht dann so aus.

```
WITH Hilfstabelle AS (
SELECT  FELD , ROWNUMBER() OVER(ORDER BY FELD) AS rn FROM (SELECT DISTINCT FELD FROM T1 ) AS R )
SELECT a.ID ,
       max(decode(rn,1,a.feld)) as FELD1,
       max(decode(rn,2,a.feld)) as FELD2,
       max(decode(rn,3,a.feld)) as FELD3
  FROM T1 A ,
       Hilfstabelle H
 WHERE a.FELD = H.FELD
 group by a.id

ID          FELD1 FELD2 FELD3
----------- ----- ----- -----
          1 A     B     C
          2 A     -     C
          3 -     B     -

  3 record(s) selected.
```

TechTipp: Tabellen-Liste und Anzahl der Datensätze

Bei der Suche nach einer bestimmten Art von Stored Procedure bin ich im SQL COOKBOOK von Graeme Birchall über folgende interessante Lösung gestolpert.

Oftmals soll eine Liste der Tabellen und die Anzahl der Datensätze pro Tabelle ausgegeben werden.

```
SCHEMA          TABNAME         #ROWS
MYSCHEMA        MYTABLE         1209
```

Bisher habe ich das so gemacht, das ich erst alle Tabellen gelesen, dann pro Tabelle den COUNT ausgeführt und dann die Liste zusammengestellt habe.
Je nach Shell ging das mit ohne ohne Zwischendatei. Das bisherige Skript (z.B. ohne Zwischendatei) sah ungefähr so aus:

```
db2 -x "select tabschema, tabname from syscat.tables where ... " | while read schema table; do
       countTbl=`db2 -x "select count(*) from $schema.$table with ur"`
       echo "$schema $table $countTbl"
done
```

Dabei geht es auch in einem SQL-Statement. Dazu wird jedoch eine FUNCTION und eine STORED PROCEDURE benötigt.
Die STORED PROCEDURE macht die eigentliche Abfrage-Arbeit. Die Funktion ist dafür da, dass die komplette Abfrage in einem SQL-Statement ablaufen kann. Nur mit der STORED PROCEDURE ginge dies nicht, da ja diese bekanntlich mit nur CALL aufgerufen werden können.

```
SELECT CHAR(tabschema,8) AS schema ,CHAR(tabname,20) AS tabname ,return_tableCount ('SELECT COUNT(*)
' || 'FROM ' || tabschema || '.' || tabname )AS #rows FROM syscat.tables WHERE tabschema not LIKE
'SYS%' ORDER BY tabschema ,tabname FOR FE
TCH ONLY WITH UR

SCHEMA   TABNAME              #ROWS
-------- -------------------- -----------
DB2NL    ANMELDUNGEN                  436
MYSCHEMA T2                             1
MYSCHEMA LISTE                          5
MYSCHEMA T1                             6

  5 record(s) selected.
```

Die dazugehörige PROCEDURE sieht wie folgt aus:

```
CREATE PROCEDURE return_tableCount (IN in_stmt VARCHAR(4000),OUT out_val INTEGER)
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
```

```
BEGIN
DECLARE c1 CURSOR FOR s1;--
PREPARE s1 FROM in_stmt;--
OPEN c1;--
FETCH c1 INTO out_val;--
CLOSE c1;--
RETURN;--
END;
```

Und nun noch die Funktion:

```
CREATE FUNCTION return_tableCount (in_stmt VARCHAR(4000))
RETURNS INTEGER
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
BEGIN ATOMIC
DECLARE out_val INTEGER;--
CALL return_tableCount(in_stmt,out_val);--
RETURN out_val;--
END;
```

TechTipp: DB2 Connect Zeiten unter Windows

Problem

Langsame Verbindung auf eine ferne Datenbank bei Verbindung von einem DB2 Windows 7 client.

Problembeschreibung

Bei Verbindungen von einem DB2 Client aus (meistens Windows 7) mit einer User ID, welche nicht auf dem Windows Client oder der Windows Domäne existiert, kommt es vermehrt zu hohen Verbindungslaufzeiten (bis zu 1-2 Minuten).

Ursache

Standardmäßig wird der DB2 Windows Client eine ID authentifizieren. Wenn diese ID jedoch auf dem Windows Client nicht existiert, werden zusätzlich die Windows Domäne und auch die vertraulichen Domains (trusted domains) durchsucht. Bis alle vertraulichen Domains durchsucht worden sind, kann dies natürlich etwas dauern. Verwendet man also eine User ID, die nur auf dem Datenbankserver bekannt ist, kommt dies natürlich häufig vor.

Lösung

Wird eine Datenbank mittels einer Server-bezogenen Authentifizierungsmethode katalogisiert, dann wird auf dem DB2 Windows Client keine Authentifizierung durchgeführt. Es wird dann direkt gegen den Datenbankserver authentifiziert.
Dies reduziert die Verbindungslaufzeiten wieder auf das übliche Maß (1-2 s).
Hier die Katalogisierungs-Kommandos für eine Server-Authentifizierung:

```
db2 catalog db <database name> at node <node name> authentication server
```

oder

```
db2 catalog db <database name> at node <node name> authentication server_encrypt
```

TechTipp: Erstellen eines EXPLAIN

Zur Laufzeit

In der DB2 Newsletter Ausgabe 01/2012 wurden die neuen Feature von DB2 9.7 bzgl. des DB2 Sections Explain vorgestellt.

Oftmals steht man als DBA vor der Aufgabe, ein einzelnes SQL bzgl. des Zugriffsplans ad hoc zu analysieren. Dieses SQL kann ein Langläufer sein, es kann zu viel CPU Ressourcen verbrauchen oder einfach nur mein Datenbanksystem zusätzlich belasten.
Ich habe mir für solche Situationen zwei kleine Tools erstellt, die mir helfen, sehr schnell einen EXPLAIN Output zu erstellen.

Im ersten Schritt suche ich die SQL's, die mich interessieren, mittels:

```
select EXECUTABLE_ID,
          QUERY_COST_ESTIMATE,
          COORD_STMT_EXEC_TIME,
          substr(STMT_TEXT,1,200) AS STMT_TEXT
  from table(MON_GET_PKG_CACHE_STMT (NULL,NULL,NULL,-1)) as t
 order by QUERY_COST_ESTIMATE desc
 fetch first 20 rows only
  with ur;
```

Wobei man selbst entscheidet, ob die ESTIMATED COST, die EXECUTING TIME oder der STATEMENT TEXT von Interesse ist. Wichtig ist nur, dass man die EXECUTABLE_ID des SQL ermittelt. Ein Output könnte so aussehen (den SQL Statementext habe ich wegen der Übersichtlichkeit weggelassen):

```
EXECUTABLE_ID                                                                    QUERY_COST_ESTIMATE  COORD_STMT_EXEC_TIME
-------------------------------------------------------------------------------- -------------------- --------------------
x'0000000100000000000000000000FDEE2B0000000000022012121715283946382 5'            61863692                    0
x'0000000100000000000000000000FE9F870000000000022012121716265095065 6'            26505150                   64
x'0000000100000000000000000000FEBC9F000000000002201212171636470257 70'            26505150                   36
x'0000000100000000000000000000FECC480000000000022012121716442820906 7'             8809406                    0
```

Im zweiten Schritt wird das Shellscript `EXPLAIN_Section.ksh` aufgerufen, mit dem Datenbank Namen und der EXECUTABLE_ID als Parameter:

```
./EXPLAIN_Section.ksh DWH 0000000100000000000000000000FDEE2B0000000000022012121715283946382 5

Output (Auszug):

+-------------------------------------------------------------------------------+
 Database: DWH
 Generate EXPLAIN for an SQL-Statement from Package Cache
 Output-File is: DWH.SQL20121217164528.expl

   Database Connection Information

 Database server        = DB2/AIX64 9.7.4
….
DB20000I  The TERMINATE command completed successfully.
DB2 Universal Database Version 9.7, 5622-044 (c) Copyright IBM Corp. 1991, 2009
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

Connecting to the Database.
Connect to Database Successful.
Output is in DWH.SQL20121217164528.expl.
Executing Connect Reset -- Connect Reset was Successful.
+-------------------------------------------------------------------------------+
```

Im Ergebnis dessen wird der Zugriffsplan in das File `DWH.SQL20121217164528.expl` geschrieben und kann anschließend analysiert werden.

Hier ist das verwendete SHELL Script:

```
#!/bin/ksh
# EXPLAIN from section
# Input:        1. Database
#               2. EXECUTABLE_ID
DB2DATABASE=$1
typeset -u DB2DATABASE
EXECUTABLE_ID=$2
EXPLAIN_OUT=$DB2DATABASE.SQL`date +%Y%m%d%H%M%S`.expl
```

```
DB2EXPLAIN=`/usr/bin/which db2exfmt`
echo ""
echo "+-------------------------------------------------------------------------------+"
echo " Database: $DB2DATABASE "
echo " Generate EXPLAIN for an SQL-Statement from Package Cache"
echo " Output-File is: $EXPLAIN_OUT"
db2 connect to $DB2DATABASE
db2 "CALL EXPLAIN_FROM_SECTION (x'$EXECUTABLE_ID', 'M',NULL,0,NULL, ?,?,?,?,?)"
db2 terminate
$DB2EXPLAIN -d $DB2DATABASE -l -no_prompt -o $EXPLAIN_OUT
echo "+-------------------------------------------------------------------------------+"
exit 0
```

Erstellen Explain für SQL aus einem SQL-File

Wenn in einem SQL Skript mehrere Statements enthalten sind, für die man ein Explain erhalten möchte, ist dies nicht so leicht mit einem Befehl möglich. Insbesondere wenn zwischen den Statements Abhängigkeiten bestehen, ist eine einzelne Ausführung von SQL's zur Erstellung des Explains nicht durchführbar. Über das folgende Skript (`make_explain.ksh`) kann jedoch das Explain für alle Statements in einer Datei ausgeführt werden:

```
#!/bin/ksh
DB=$1
db2 connect to $DB > /dev/null
zeit=`date +"%Y-%m-%d-%H.%M.%S`
zeit2=`date +"%H.%M.%S`

jahr=`date +"%Y"`

i=1
#db2 set current query optimization=7
#db2 "set current degree '1'"
db2 set current explain mode explain

db2 -vtf $2

db2 set current explain mode no
db2 set current explain snapshot no

db2 "select EXPLAIN_TIME from EXPLAIN_STATEMENT WHERE explain_level='P' and explain_time > '$zeit'"
| fgrep $jahr | while read snapzeit; do

 echo "Explain Phase $i: $snapzeit"
 db2exfmt -g TIC -d $DB -w $snapzeit -o ${1}-${i}-${zeit2}.exp -n % -s % -e % -# 0

 let i=$i+1

done
```

Der Aufruf des Skriptes erfolgt dann mit:

```
./make_explain.sh <DBNAME> <SQL-FILE-mit-Semikolon-Delimited-Statements.sql>
```

Chats mit dem Labor

Eine Liste der bereits durchgeführten Chats ist hier zu finden.
Die Präsentationen der Chats können dort angeschaut und heruntergeladen werden.

Schulungen / Tagungen / Informationsveranstaltung

Eine Liste der anstehenden Konferenzen ist hier zu finden.

Newsletter Archiv

Alte Ausgaben vom DB2-NL sind nun zum Nachlesen in den Archiven zu finden von:

- Lis.Tec
- Bytec
- Drap
- Cursor Software AG
- ids-System GmbH

Anmeldung/Abmeldung

Sie erhalten diesen Newsletter bis zur 3ten Ausgabe ohne Anmeldung. Wenn Sie weiterhin diesen Newsletter empfangen wollen, schicken Sie Ihre Anmeldung mit dem Subject „ANMELDUNG" an db2news@de.ibm.com.

Die Autoren dieser Ausgabe

Sollten Sie Anfragen zu den Artikeln haben, können Sie sich entweder direkt an den jeweiligen Autor wenden oder stellen Ihre Frage über den DB2 NL, denn vielleicht interessiert ja die Antwort auch die anderen DB2 NL Leser.

Doreen Stein	Master Certified IT-Spezialist für DB2 LUW, IBM SWG; Chief-Editor DB2NL, djs@de.ibm.com
Peter Schurr	IT Specialist, IBM SWG #TechTipp: DB2 Connect Zeiten unter Windows
Gerhard Müller	Senior IT Specialist , IBM SWG #Erstellen eines EXPLAIN zur Laufzeit
Frank Berghofer	Senior IT Specialist, IBM SWG #TechTipp: Transponieren

Reviewer und Ideenlieferanten:

Wilfried Hoge	IBM SWG #Erstellen Explain für SQL aus einem SQL-File
Dirk Fechner	IBM SWG