Willkommen zum „IBM DB2 Newsletter“

Liebe Leserinnen und Leser,

es ist schon lange her, seit dem letzten Newsletter. Einige Konferenzen und Workshops haben stattgefunden. Der Herbst hat die Blätter gefärbt und das letzte Viertel des Jahres ist angebrochen.

Genauso bunt und interessant, wie das von KiTa-Kindern zusammengestellte Mandala aus Naturmaterialien, sind auch diesesmal unseren Beiträge.

Wie immer wünschen wir viel Spaß beim Lesen.

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

Und wieder mal ein Aufruf in eigener Sache: Sie haben eine kleine Sammlung an Tipps & Tricks in irgendeiner Datenbank oder Datenarchiv vergraben? Dann teilen Sie Ihr Wissen mit der DB2 NL Gemeinde und schicken mir ein paar davon.

Ihr TechTeam

Inhaltsverzeichnis

Anmerkungen zur Ausgabe 201003

Nach dem letzten Newsletter habe ich erfahren, das doch einige von Ihnen den Newsletter lesen und nicht nur archivieren. Es gab zahlreiche Reaktionen und Anregungen aufgrund der

versehentlich eingebauten Fehler..

So hat sich im Abschnitt „Ersetzen von Texten" ein weiterer Fehler eingeschlichen. Erst war es nicht -e für die Inplace Aktion, sondern -i. Dann sind die Einschränkungen der globalen Ersetzung aufgefallen. Alles wird ersetzt (auch wenn Suchstring nur ein Teilstring ist), was zur Verfälschung von Tabellen oder Tablespace-Namen führen könnte. Daher wurde nachfolgender Artikel als Alternativlösung geliefert:

TechTipp: Ersetzen von Texten

Ausgehend von den Beiträgen des letzten Newsletters erhielten wir folgenden Beitrag, wie man Kommandos ausführen kann.

Häufig ist es sinnvoller, einen Textstring nicht global zu ändern, sondern diesen gezielt als Parameter zu übergeben, also SQL mit Shellvariablen unter Unix/Linux zu nutzen.

```
#--> Skript insert
    db2 connect to sample

        ZWECK=TEST
        # ZWECK=PROD


        #   db2 " create table test.mustertab (i int, c char(12), t timestamp, k int) "
        #   db2 " create table prod.mustertab (i int, c char(12), t timestamp, k int) "



        #   for (( i=1; i<=100; i-- ))   # Bash V2.0
        #   for i in {1..100}            # Bash V3.0
            for i in $(seq 1 100 )       # ksh
            do
                db2 -v " insert into "$ZWECK".MUSTERTAB  values (" $i "  \
                    , 'hallo', current timestamp, " $((100-$i)) "  ) "
            done
            db2 connect reset ;
```

Das kann man auch weitertreiben, in dem man mit SQL die Werte für eine Shellvariable ermittelt, die man dann wieder als Shellvariable im nächsten SQL-Statement nutzt.

Mit diesem Skript-Teil wird die Anzahl der Zeilen aller Katalog-Tabellen ermittelt, unabhängig von der DB2 Version:

```
db2 connect to sample

for name in $( db2 -x " select tabname from syscat.tables where tabschema ='SYSCAT'  "  ); do
   db2 " select count(*) as count, '"  $name " ' as name from syscat."$name
done
db2 connect reset
```

In der Fassung vom TechTipp: Kommando Pipe in db2 CLP hat dieses Skript die Form

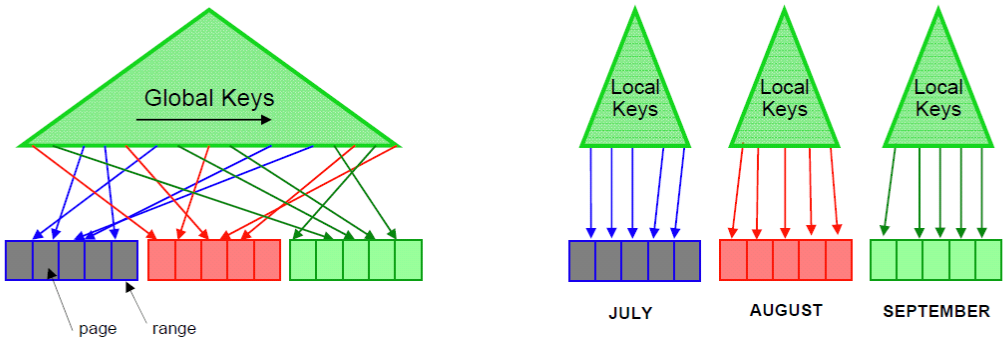
```
db2 connect to sample

for name in $( db2 -x " select tabname from syscat.tables where tabschema ='SYSCAT'  "  ); do
db2 +p -svt  <<EOF
     select count(*) as count, ' $name  ' as name from syscat.$name  ;
     select current timestamp from sysibm.sysdummy1  ;
EOF
done
db2 connect reset
```

Artikel: Partitonierte Indizes für Range partitionierte Tabellen

Seit DB2 9.7 können Indizes von Range partitionierten Tabellen (table partitioning) ebenfalls partitioniert abgelegt werden, so daß jedes Datenpartition ihren zugeordneten Index (index partition) hat.
Dies ermöglicht signifikante Verbesserung beim ATTACH/DETACH und auch die Geschwindigkeit von Backup/Restore hat sich merklich verbessert.
Nähere Informationen sind zu diesem Thema sind in der Online-Hilfe zu finden.

Weiterhin sind aber auch die nichtpartitionierten (oder auch globale) Indizes möglich, die über alle Partitionen einer Tabelle quer referenzieren.



Per Default, wenn neue Indizes auf partitionierten Tabellen angelegt werden, werden diese als partitionierte Indizes angelegt. Ausnahme dabei bilden UNIQUE Indizes, die nicht den PARTITION BY RANGE Schlüssel beinhalten.
Soll ein globaler Index angelegt werden, muß dies unter Verwendung der Schlüsselwörter NOT PARTITIONED erfolgen.

```
CREATE UNIQUE INDEX TPCD.O_OK ON TPCD.ORDERS (O_ORDERKEY ASC) NOT PARTITIONED IN
ORDERSI;
```

Partitionierte Indizes liegen in den Tablespaces welche beim CREATE TABLE Statement angegeben wurden. z.B.

```
CREATE TABLE TPCD.ORDERS  (
…
PARTITION BY RANGE("O_ORDERDATE")
(PART "PART0" STARTING(MINVALUE) IN "TABDATA0" INDEX IN "TABINDEX0",
PART "PART1" STARTING('1992-01-01') IN "TABDATA1" INDEX IN "TABINDEX1",
```

Wurde die “INDEX IN” Klausel nicht spezifiziert, dann liegen die partitionierten Indizes in dem gleichen Tablespace wie die dazugehörigen Daten.

Hier der Beispielhafte Syntax für das Erzeugen eines partitionierten Indizes O_OK

```
CREATE INDEX TPCD.O_OK_LOCAL ON TPCD.ORDERS (O_ORDERKEY ASC) PARTITIONED;
```

Das Schlüsselwort `UNIQUE` in Zusammenhang mit o.g. Index O_OK ist nicht möglich, da der Partitionierungsschlüssel O_ORDERDATE nicht im Index enthalten ist. Eine Erweiterung des Indexes mit dem Partitionierungsschlüssel ist für UNIQUE notwendig:

```
CREATE UNIQUE INDEX TPCD.O_OK ON TPCD.ORDERS (O_ORDERKEY, O_ORDERDATE)
PARTITIONED;
```

Bezugnehmend auf Größe der Indizes von partitionierten bzw. Nicht-partitionierten Index, gibt es einige Faktoren die mehr oder weniger Platz für den Index bestimmen. In einem nichtpartitionierten Index, benötig jeder Indexschlüssel 2 Byte mehr als beim gleichwertigen partitionierten Index. Jedoch ist ein partitionierter Index verteilt auf mehrere Objekte und kann daher potentiell mehr Platz benötigen.

In V9.7 kann die Größe von partitionierten und nicht-partitionierten Indizes mit der Tabellenfunktion `admin_get_index_info` wie folgt, ermittelt werden:

```
select substr(tabname,1,25) tabname,substr(indname,1,10) indname, compress_attr,
index_compressed, count(*) #ranges, sum(index_object_p_size/1024) MB
from table (admin_get_index_info ('T', 'TPCD', '')) as ti where DBPARTITIONNUM=1 group by
substr(tabname,1,20),substr(indname,1,10), compress_attr, index_compressed order by 1

TABNAME              INDNAME    COMPRESS_ATTR INDEX_COMPRESSED #RANGES     MB
-------------------- ---------- ------------- ---------------- ----------- --------------------
ORDERS               O_CK_GLOBAL N             N                  1                4908
ORDERS_L             O_CK        N             N                 84                5440
```

Performance Vergleich zwischen partitionierten und nicht-partitionierten Indizes

Anlage eines partitionierter Indizes ist um ein vielfaches schneller (> 3 mal im Test) als die Anlage des nicht-partitionierten Indizes.

Signifikante Vorteile des partitionierten Indizes ist die Möglichkeit den REORG TABLE auf nur einer Tabellenpartition durchzuführen.

```
REORG TABLE TPCD.ORDERS ON DATA PARTITION PART79
```

Wenn bekannt ist, das sich nur die letzte Tabellenpartition verändert und daher eine Reorganisation benötigt, dann muss nur diese Partition reorganisiert werden.
Müssen mehrere Partitionen REORGanisiert werden, dann kann dies parallel erfolgen, was ebenfalls die Performance verbessert.

Das Roll-out der Daten wird beim partitionierten Index ebenfalls schneller. Wenn also eine Datenpartition ausgeklinkt (DETACH) werden soll, nimmt diese Datenpartition ihre partitionierten Indizes mit und die Indizes müssen nach dem Detach nicht noch mal erzeugt werden. Weiterhin ist eine asynchrone Index Bereinigung (AIC .. asynchronous index cleanup) ist bei partitionierten Indizes nicht notwendig.
ATTACH und DETACH Operationen sind schneller (abhängig von der Datenbank und Tabellen Konfiguration und der Tabellen Constraints). Weiterhin wird weniger Logspace bei diesen Aktionen benötigt.
Ein Syntaxbeispiel für ATTACH/DETACH:

```
alter table tpcd.orders detach partition PART0 into tpcd.orders_temp;
```

(asynchronous)

```
alter table tpcd.orders attach partition PART0 starting '1/1/1992' ending
'1/31/1992' from table tpcd.orders_temp;
set integrity for tpcd.orders immediate checked;
```

Trotz allem gibt es einige vernachlässigbare Unterschiede bei INSERTs und DELETEs zwischen partitionierte/nicht-partitionierte Indizes geben.

Abfrageperformance kann sehr unterschiedlich sein bei partitionierten Indizes.
Auch der optimierte Zugriffsplan kann unterschiedlich sein. Beides ist zu erwarten, sowohl Performance Steigerung, als auch Verschlechterung. Daher ist zu empfehlen, wichtige Abfragen vor der Umstellung in Produktion zu testen.

TechTipp: Ermitteln Sortheap und Package Cache mit Views

Sie verwenden immer noch ihre bewährten Scripte aus der Zeit vor den Administrationsviews und -Funktionen?
Ich teilweise auch. Damit Ihnen aber der Umstieg auf die Views/Funktionen zu vereinfachen und werden daher immer mal wieder ein paar Abfragen auf diese in SQL-Form zu liefern.

Beginnen werden wir mit Informationen über SORTHEAP und PACKAGE CACHE.

Sort Heap

Mit der nachfolgenden Abfrage (die auch schon in V9.5 funktioniert) soll die Verwendung des Sort-Speichers überprüft werden.
Als Basis dazu wird SHEAPTHRES_SHR aus der Datenbank Konfiguration in Verhältnis zum Allokierten Shared Sort Speicher gesetzt.
Weiterhin werden die Werte für Sort Overflow und aktive Sorts ausgegeben.

Im pre-View sahen die Ergebnisse wie folgt aus:

```
>db2 get db cfg | find /i "sheap"

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(268)

>db2 get snapshot for database on sample | find /i "sort"
Total Private Sort heap allocated          = 0
Total Shared Sort heap allocated           = 4
Shared Sort heap high water mark           = 135
Post threshold sorts (shared memory)       = 0
Total sorts                                = 4
Total sort time (ms)                       = 51
Sort overflows                             = 0
Active sorts                               = 0
    Memory Pool Type                           = Shared Sort Heap
```

Aus diesen einzelnen Werten musste dann das Ergebnis zusammengesucht werden.
Mit den administrativen Views, lässt sich dies nun wie folgt ermitteln:

```
with dbcfg1 as (
  select int(value) as sheapthres_shr
    from sysibmadm.dbcfg
   where name = 'sheapthres_shr' )

  select sheapthres_shr as "Shr_sheap" ,sort_shrheap_allocated as "Shr_sort_all"
       , dec((100 * sort_shrheap_allocated)/sheapthres_shr,5,2) as " % Sheap_alloc"
       , dec((100* sort_shrheap_top)/sheapthres_shr,5,2)as " % MaxSheap_alloc"
       , sort_overflows as "SortOverflows", active_sorts as "ActSorts"
    from dbcfg1, sysibmadm.snapdb
     ;
```

Die Ausgabe dazu sieht dann wie folgt aus.

```
Shr_sheap Shr_sort_all % Sheap_alloc  % MaxSheap_alloc SortOverflows ActSorts
--------- ------------ -------------- ---------------- ------------- ----------
      268            4           1.00             1.00             0          0

  1 record(s) selected.
```

Package Cache

Die Ausnutzung des Package Caches kann nun mit folgendem SQL ermittelt werden und ersetzt dazu die bisherige grepperei aus der Datenbank-Konfiguration und dem Datenbank Snapshot.

```
with dbcfg1 as (
   select int(value) as pckcachesz
     from sysibmadm.dbcfg
    where name = 'pckcachesz' )

  select pckcachesz as "Package Cache Size" ,pkg_cache_lookups as "Lookups"
       , pkg_cache_inserts as "Inserts" ,pkg_cache_num_overflows as "Overflows"
       , 100 * pkg_cache_size_top / (pckcachesz * 4096)as " % PKG Cache alloc"
    from dbcfg1, sysibmadm.snapdb
   ;

Package Cache Size Lookups              Inserts              Overflows            % PKG Cache alloc
------------------ -------------------- -------------------- -------------------- --------------------
               320                   15                    2                    0                   21

  1 record(s) selected.
```

TechTipp: Liste aller DB2 Instanzen und Datenbanken

Oft sind auf einem Server viele Instanzen von DB2 installiert.
In Produktionsumgebungen kommt es zudem vor, dass nicht alle Instanzen zur gleichen Zeit mit einem Fixpack versorgt werden dürfen (restricted Downtimes).
Zudem sind (teilweise zum Test) auch mehrere Versionen oder Fixpack Level von DB2 installiert.

Um auf einen Blick den Überblick zu bekommen welche Datenbank in welcher Instanz aufgebaut ist, und welche Instanz auf welcher Version und mit welchem Fixpack arbeitet, kann ein Script erstellt werden, das all diese Informationen ermittelt.

Der Kern dieses Scripts ist der db2ls, der alle Installationen auflistet. Damit kann der db2ilist lokalisiert werden. Mit Hilfe der Informationen über die Instanz kommt man dann mit dem "db2 list db directory" den Datenbanken auf die Spur.

Anbei das Script, das die Übersicht verschafft:

```
#!/bin/bash
cnt=0;
for a in `db2ls | cut -f1 -d ' '`
do
   if [ $cnt -gt 1 ]
   then (
    for i in `ls $a/instance/db2ilist`
    do
      echo $i
      for j in `$i`
      do
        echo $j
        . /home/$j/sqllib/db2profile
        db2 list db directory | grep 'Database name'
      done
    done
   )
   fi;
   cnt=`expr $cnt + 1`
done
```

Die Ausgabe hat dann die Form:

```
/opt/ibm/db2/V9.5FP4/instance/db2ilist
db2luw1
 Database name                        = MX42
 Database name                        = MARION
/opt/ibm/db2/V9.5FP5/instance/db2ilist
db2luw3
 Database name                        = CARMEN
 Database name                        = CARINA
/opt/ibm/db2/V9.5FP6/instance/db2ilist
db2luw4
 Database name                        = DANIELA
/opt/ibm/db2/V9.5FP6.23042/instance/db2ilist
/opt/ibm/db2/V9.7/instance/db2ilist
db2luw2
 Database name                        = SIBYLLE
```

TechTipp: Entfernen von Blanks in Datenbankausgabe

Wer kennt das nicht, auf Command Line Ebene Abfragen zu starten und dann rennt das Ergebnis über den Bildschirm, pro Datensatz mehr als eine Zeile, mit Unmengen an Leerzeichen.
Ich habe mir dafür folgenden kleinen Alias erstellt, der erstmal die Leerzeichen entfernt:

```
alias STRIP='tr -s '\'' '\'' '\'' '\'''
```

Nun rufe ich alle Abfragen wie folgt auf:

```
db2 „select * from syscat.tables"| STRIP
```

Die Blanks sind zwar nun weg, aber nach der Überschrift sind immer noch die unübersichtlichen „-". Man könnte natürlich mit db2 -x die Kopf-/Fußzeilen von DB2 ausblenden, sieht damit aber auch nicht mehr die Spaltenüberschriften.

Daher sieht mein endgültiger STRIP-Alias wie folgt aus:

```
alias STRIP='tr -s '\'' '\'' '\'' '\'' |tr -s '\''-'\'' '\''-'\''| sed '\''/^- $/d'\'' '
```

Chats mit dem Labor

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

Schulungen / Tagungen / Informationsveranstaltung

Eine Liste der anstehenden Konferenzen ist hier zu finden.

Für Sie dabei gewesen

Viele interessante Informationen und Beiträge wurden während der DB2 Aktuell in Schweinfurt und beim DB2 LUW Workgroup Treffen der GSE in Lüneburg geliefert. In den nächsten Ausgaben werden wir immer wieder mal interessante Beiträge mit Wissenswertes aus den dort gehaltenen Beiträgen einbringen.

Newsletter Archiv

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

- Lis.Tec
- Bytec
- Drap
- Cursor Software AG

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	IT-Spezialist für DB2 LUW, IBM SWG; Chief-Editor DB2NL djs@de.ibm.com
Gerd Kaluzinski	IT-Specialist Informix Dynamic Server und DB2 UDB IBM Software Group, Information Management gerd.kaluzinski@de.ibm.com TechTipp: Liste aller DB2 Instanzen und Datenbanken
Matthias Rawohl	Finanz-Informatik, Hannover matthias.rawohl@f-i.de TechTipp: Ersetzen von Texten
Nela Krawez	IBM SWG, InfoSphere Balanced Warehouse Development lieferte die englische Grundlage für den Artikel: Partitonierte Indizes für Range partitionierte Tabellen

Reviewer und Ideenlieferanten:

Dirk Fechner	IBM SWG
Frank Berghofer	IBM SWG
Volker Fränkle	IBM SWG