Willkommen zum „IBM DB2 Newsletter“

Liebe Leserinnen und Leser,

Der November ist bald vorbei und der Dezember steht mit seiner vorweihnachtlichen und glühweinreichen Zeit in den Startlöchern, die Weihnachtsmärkte öffnen ihre Pforten und die große Frage kommt wie jedes Jahr, was soll man denn dieses Jahr schenken. Bei der Auswahl der Geschenke können wir nicht helfen, aber sicherlich können wir mit diesem Newsletter davon ablenken.
Dafür haben wir wiedereinmal schlauen Tipps und Tricks für Sie zusamenngesucht (auch zum ausprobieren).

Viel Spaß beim Lesen.

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

Ihr TechTeam

Inhaltsverzeichnis

DB2 9.7 Fixpak 1 verfügbar

Die aktuelle Version von DB2 ist schon seit dem Sommer verfügbar. Nun gibt es auch das erste Fixpak dazu. Damit werden nun auch einige Funktionen nachgereicht. Eine wesentliche Neuigkeit in DB2 9.7 war die **Unterstützung von PL/SQL**. Bisher beherrschte DB2 nur die eigene SQL Skriptsprache SQL/PL. Diese Erweiterung ist wichtig, wenn eine Anwendung auf DB2 umgestellt werden soll, die PL/SQL verwendet. Während früher eine Migration der Skripte nötig war, kann DB2 diese nun direkt ausführen. Dabei werden beide Dialekte gleichberechtigt behandelt, sind also beide gleich performant. Um die Umstellung auf DB2 zu ermöglichen,

reicht natürlich die Unterstützung für PL/SQL nicht aus.
Mit DB2 9.7 wurden daher viele Konzepte hinzugefügt. Dies sind z.B. neue Datentypen und Funktionen oder der neue **Sperrmodus Currently Commited**. Damit ist es möglich die Parallelität von Anwendungen zu erhöhen, weil Schreibprozesse keine Leseprozesse mehr blockieren. Bei Daten, die gerade geändert werden, liest eine Anwendung einfach den alten Wert aus dem Log(buffer).
In Fixpak 1 wurden weitere Funktionen und Schnittstellen hinzugefügt, die die Kompatibilität erhöhen. Sogar **SQLplus Skripte** können mit DB2 nun **ausgeführt** werden.
Die bereits in 9.5 eingeführte **Komprimierung** von Daten wurde in 9.7 weiter **verbessert**. Sie erstreckt sich nicht mehr nur auf Daten, sondern umfasst nun auch Indexe, temporäre Daten und XML. Der Speicherplatz lässt sich also noch effizienter nutzen. Auch mit dem Problem, dass sich ein **High Water Mark** eines Tablespaces nicht **reduzieren** ließ, wurde aufgeräumt. Eine **neuer Tablespace Typ** ermöglicht das **Komprimieren der Container**.
Beim Range Partitioning sind jetzt neben globalen Indexen auch **lokale Indexe** möglich. Dies vereinfacht das Hinzufügen und Wegnehmen von Partitionen, weil keine zusätzliche Aktivität mehr notwendig ist.
Mit dem Fixpak 1 ist für HADR nun auch das **Lesen auf dem Standby System** möglich. Reports können also auf der Standby Datenbank ausgeführt werden, benötigen keine Unterbrechung der Replikation und beeinträchtigen nicht die Produktionsseite.
Wichtige **Erweiterungen** gibt es hinsichtlich **Monitoring und Performance Tuning**. DB2 ermittelt nun, wo ein Statement Zeit verbringt. Aufgeschlüsselt wird beispielweise nach Zeit für CPU, Lesen, Locks, Netzwerk, usw. Somit lassen sich Engpässe wesentlich schneller identifizieren als bisher. Auch bei den Ausführungsplänen gibt es Neuerungen. Neben den beim Explain schon bisher bekannten geschätzten Werten können nun auch aktuelle Werte im Plan eingebunden werden. Dadurch läßt sich die Schätzung von DB2 sehr leicht mit der Realität vergleichen und es es wird schneller offensichtlich, wo eine Verschätzung zum Problem wird. Außerdem kann ein für **gut befundener Plan festgeschrieben** werden (plan lock down). Dies garantiert dann die gleiche Ausführung, auch wenn sich die Systemumgebung verändert.
Dies ist nur ein kleiner Abriss der vielen Neuigkeiten in DB2 9.7. Weitere Informationen finden sie im Internet.

Viele der Themen werden wir in den kommenden DB2 Newslettern aufgreifen und näher beleuchten.

Nähere Informationen zum Inhalt von Fixpak 1 erhalten sie hier

TechTipp: Erzeugen eindeutiger Schlüssel mit CURRENT TIMESTAMP, eine gute Wahl?

In vielen Umgebungen stellt sich die Frage, wie man Zeilen eindeutig machen kann.
Man kann die mit einer Sequence erreichen, hat damit aber keine Datums- und Uhrzeitinformationen.
Der meißtgewählte Ansatz ist der, dass in einer Spalte der Current Timestamp eingefügt wird.

Beispiel:

```
create table test1
(
....
ts timestamp
);
```

Beim Einfügen wird dann einfach der CURRENT TIMESTAMP übergeben.

Beispiel:

```
insert into test1 values ( .... , CURRENT TIMESTAMP);
```

Diese Lösung ist aus mehrerlei Gründen problematisch.

Wenn mehrere Zeilen in einem Schritt eingefügt werden, so erhalten sie alle den gleichen Zeitstempel.

Beispiel:

```
create table test1 (ts timestamp);
insert into test1 select current timestamp from syscat.tables fetch first 10 rows only;
```

hat zur Folge, dass alle 10 Sätze den gleichen Timestamp erhalten:

```
db2 "select * from test1"

TS
--------------------------
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000
2009-11-05-17.49.57.671000

  10 Satz/Sätze ausgewählt.
```

Man kann dies mit einer Kombination aus Timestamp und Sequence in den Griff bekommen, es gibt aber schlauere Methoden.
Aus diesem Grund wird die Benutzung der Skalarfunktion „generate_unique()" empfohlen. Diese erzeugt nicht nur eine aufsteigende Folge von Werten je Record, sondern sogar je Spalte.

Beispiel:

```
db2 "create table test1 (ts timestamp, cu1 char(16) for bit data not null, cu2 char(16) for bit
data not null)"
DB20000I  Der Befehl SQL wurde erfolgreich ausgeführt.

db2 "insert into test1 select current timestamp, generate_unique(), generate_unique() from
syscat.tables fetch first 10 rows only"
DB20000I  Der Befehl SQL wurde erfolgreich ausgeführt.

db2 "select * from test1"

TS                         CU1                              CU2

-------------------------- -------------------------------- --------------------------------
2009-11-05-18.06.37.906000 x'20091105170638588098000000202020' x'20091105170638588092000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588124000000202020' x'20091105170638588122000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588132000000202020' x'20091105170638588130000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588140000000202020' x'20091105170638588138000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588148000000202020' x'20091105170638588146000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588156000000202020' x'20091105170638588154000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588164000000202020' x'20091105170638588162000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588172000000202020' x'20091105170638588169000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588179000000202020' x'20091105170638588177000000202020'
2009-11-05-18.06.37.906000 x'20091105170638588187000000202020' x'20091105170638588185000000202020'

  10 Satz/Sätze ausgewählt.
```

Man sieht, dass man nicht nur eine aufsteigende Folge von Werten je Zeile erhält, sondern, dass auch innerhalb einer Zeile die Spalten CU1 und CU2 unterschiedliche Werte haben.
Die Werte werden aus dem Timestamp in Universal Time UTC (nicht identisch mit dem Current Timestamp) und der Partitionsnummer, auf der die Funktion ausgeführt wird, gebildet. Das vorliegende Beispiel wurde auf einer Windowsumgebung ausgeführt, man sieht, dass die Granularität der Zeitangabe für CU1 bzw CU2 größer ist, als für den Timestamp. Die Genauigkeit des CURRENT TIMESTAMP endet nach der dritten Nachkommastelle, währenddessen die Werte der Funktion wesentlich granularer sind.

Das liegt daran, dass die Genauigkeit des CURRENT TIMESTAMP von der Systemuhr abhängt (bei Windows 7/1000 sec. per Ticks).

Man kann aus der Funktionsausgaben aber auch den Timestamp wieder extrahiern.
Beispiel:

```
select timestamp (generate_unique()) from sysibm.sysdummy1
```

Ein weitere Nachteil ist, dass die Werte für CURRENT TIMESTAMP gecached werden. Bei einem sehr großen System mit vielen gleichzeitigen Transaktionen hat das in (extrem seltenen) Fällen dazu geführt, dass der Wert für den CURRENT TIMESTAMP variiert. In diesem Extremfall konnten wir folgenden Effekt beobachten:

```
TIMESTAMP                   GENERATEUNIQUE
2009-10-19-11.20.42.662773  20091019112042663003000000202020
2009-10-19-11.20.42.662764  20091019112042675531000000202020
```

Wenn man die Reihenfolge (order by) über Timestamp bildet, so kommt der untere Datensatz vor dem oberen. Die Reihenfolge über die „generate_unique"-Spalte spiegelt die richtige Reihenfolge.

Das Lab bestätigte, dass dies durch Cachingeffekte in extremen Fällen auftreten kann und empfiehlt die Benutzung der generate_unique Funktion.

Fazit:
Wenn man mit Hilfe des aktuellen Timestamp eindeutige Schlüssel erzeugen möchte, so ist die Funktion „generate_unique ()" das Mittel der Wahl !!!

Artikel: Abschätzung der Performanceauswirkung von dynamischen SQL-Statements durch Reoptimierung

Der Vorteil beim Einsatz von dynamischem SQL mit Paramtermarkern besteht generell darin, dass ein Statement nur einmal prepariert wird und dann bei jeder weiteren Ausführung nur noch ausgeführt werden muss.
Dieser Mechanismus ist ideal, wenn für ein Statement, trotz des nur einmaligen Optimierens mit einem unbekannten Wert, der optimale Zugriffsplan für alle folgenden Ausführungen gefunden wird.
Bei trivialen Zugriffen ist dies auch meist der Fall, aber nicht unbedingt bei etwas komplexeren Abfragen. Es ist nämlich zu beachten, dass die Optimierung nicht mit den wirklichen Werten stattgefunden hat, sondern mit auf Schätzungen beruhenden Standardwerten.
DB2 stellt für diese Problematik 2 Alternativen zu Verfügung:
Reoptimization Always (Package NULLIDRA)
Reoptimization Once (Package NULLIDR1)

Bei REOPT=Always wird bei jeder Ausführung eine Ermittlung des besten Zugriffspfad durchgeführt. Dieses ist natürlich ein Overhead, der evtl. Laufzeitverbesserungen schnell zunichte machen kann.
Bei REOPT=Once wird wird die Statement Kompilation auf den Zeitpunkt der ersten Ausführung verschoben. Damit wird der Wert der ersten Eingabevariable für die Optimierung verwendet. Nachfolgende Ausführungen verwenden nun ebenfalls diesen Zugriffsplan. Diese Methode ist vorteilhaft, wenn der erste Wert für den Parametermarker repräsentativ für die folgenden Werte ist.
Das Standardverhalten ist REOPT=None.

Die Schwierigkeit für den DBA und die Entwickler besteht nun darin herauszufinden, welche Einstellung die beste ist. Dabei sind 2 Möglichkeiten zu unterscheiden:

- Optimierung kann auf Statementebene vorgenommen werden
- Optimierung muss global für die Anwendung erfolgen

Natürlich ist es vorteilhafter, wenn die Einstellung pro Statement vorgenommen werden kann,

da bei globalen Einstellungen immer einige Statements profitieren, während andere langsamer laufen werden.
Allerdings ist eine Optimierung auf Statementebene oft nicht möglich, da die Anwendung nicht geändert werden kann oder die Möglichkeit in der entsprechenden Programmiersprache gar nicht besteht. Ein Ausweg besteht dann u.U. in der Verwendung eines Optimizer Profiles. Damit kann Einfluss auf ein SQL-Statement genommen werden, ohne die Anwendung zu ändern.

Beispiel für ein Optimizer Profile:

```
<STMTPROFILE ID="REOPT example ">
    <STMTKEY>
      <![CDATA[select acct_no from customer where name = ? ]]>
    </STMTKEY>
    <OPTGUIDELINES>
      <REOPT VALUE='ALWAYS'/>
  </OPTGUIDELINES>
</STMTPROFILE>
```

Optimierung auf Statementebene

Besteht die Möglichkeit einer Optimierung auf Statementebene müssen die geeigneten Kandidaten herausgefunden werden, für die sich eine Reoptimierung lohnt. Hierzu eignet sich eine Abfrage des dynamischen Statementcaches. Dabei ist es lohnenswert sich Statements anzusehen, die eine hohe Ausführungszeit haben und oft aufgerufen werden. Wenn die durchschnittliche Ausführungszeit gering ist, spricht das eher gegen die Möglichkeit einen deutlich besseren Plan zu finden. Tendenziell sprechen sehr geringe Prepare-Zeiten für sehr einfache Abfragen mit eher wenig Optimierungspotential. Insofern ist hier die Wahrscheinlichkeit einen deutlich besseren Plan zu treffen gering, andererseits ist der durch die Reoptimierung verbundene Overhead auch nicht so groß.
Bei der Suche nach geeigneten Statements, kann beispielsweise folgende Abfrage helfen:

```
SELECT
   NUM_EXECUTIONS * AVERAGE_EXECUTION_TIME_MS / 1000.000000000 AS TOTAL_EXEC_TIME_S,
   PREP_TIME_PERCENT,
   AVERAGE_EXECUTION_TIME_MS / 1000.000000000                  AS AVG_EXEC_TIME,
   NUM_EXECUTIONS,
   PREP_TIME_MS / 1000.000000000                               AS SUM_PREP_TIME,
   SUBSTR(STMT_TEXT,1, 60)                                     AS STMT_TEXT
FROM SYSIBMADM.QUERY_PREP_COST
WHERE AVERAGE_EXECUTION_TIME_MS  > 100
ORDER BY 1 DESC FETCH FIRST 50 ROWS ONLY;
```

Um die Anzahl der durchgeführten Prepare- bzw. Kompilationen pro Statement zu ermitteln , muss ein anderer Administrative View abgefragt werden:

```
SELECT
  (TOTAL_EXEC_TIME + TOTAL_EXEC_TIME_MS / 1000000.000000 )     AS TOTAL_EXEC_TIME_S,
   NUM_COMPILATIONS,
   PREP_TIME_BEST  / 1000.000000000                            AS PREP_TIME_BEST_S,
   PREP_TIME_WORST  / 1000.000000000                           AS PREP_TIME_WORST_S,
   SUBSTR(STMT_TEXT,1, 60)                                     AS STMT_TEXT
FROM SYSIBMADM.SNAPDYN_SQL
ORDER BY 1 DESC FETCH FIRST 50 ROWS ONLY;
```

TOTAL_EXEC_TIME_S:	Aufsummierte Ausführungszeiten eines Statements in Sekunden
AVG_EXEC_TIME:	Durchschnittliche Ausführungszeit eines Statements in Sekunden
SUM_PREP_TIME:	Aufsummierte Prepare-Zeiten eines Statement in Sekunden
AVERAGE_EXECUTION_TIME_MS:	Durchschnittliche Ausführungszeit in Millisekunden
NUM_EXECUTIONS:	Anzahl der Ausführungen eines Statements
NUM_COMPILATIONS:	Anzahl der Kompilationen eines Statements
PREP_TIME_BEST_S:	Kürzeste Prepare-Zeit eines Statements in Sekunden
PREP_TIME_WORST_S:	Längste Prepare-Zeit eines Statements in Sekunden
STMT_TEXT:	Statementtext des zugehörigen SQL-Statements

Die Abfragen wurden mit einer DB2 LUW V9.7 durchgeführt. In älteren Versionen weichen die View-Definitionen leider etwas ab und müssen angepasst werden.
Zum Monitoring der Reoptimierung eignet sich das Tool db2pd, z.B.:

```
db2pd -alldbs -reopt
```

Optimierung mit globaler Einstellung für gesamte Anwendung

Oft besteht nicht die Möglichkeit die Einstellung individuell für ein Statement zu ändern, da beispielsweise kein JDBC mit (JCC Universal Treiber) verwendet wurde, oder die Anwendung einfach nicht angepasst werden soll. In diesem Fall kann nur eine der 3 Einstellungen (NONE, ONCE, ALWAYS) global eingestellt werden. Für eine CLI-Anwendung, z.B. in der db2cli.ini. Hier wird es generell so sein, dass einige Statements von einer geänderten Einstellung profitieren, während andere aufgrund des Overheads einer Recompilation Nachteile haben. Um genau festzustellen, welche Einstellung am Besten ist, müsste ein entsprechender Performance-Test mit den verschiedenen Einstellungen vorgenommen werden. Dies ist natürlich insbesondere in Produktionsumgebungen oft nicht so einfach möglich. Insofern sollte auch hier mit geeigneten Abfragen des dynamischen Statementcaches (siehe oben) eine grobe Einschätzung vorgenommen werden.

Weitere Informationen zu dem Thema finden sich unter:

- Best Practice – Writing and Tuning Queries for Optimal Performance
- Query optimization in DB2 using REOPT
- Statement Concentrator in v9.7
- Optimizer Profile

TechTipp: Date Formatierung

In letzter Zeit wurde ich immer häufiger gefragt, wie man die Ausgabe das Datum-Formats ändern kann.

Wurde die Datenbank mit Territory US und der CODEPAGE 1208 angelegt, bekommt man das wohlbekannte US-Format MM/DD/YYYY angezeigt. Dies entspricht aber nicht dem europäischen Format (DD.MM.YYYY).

Auf der Suche im Internet bin ich dann über einen developerworks-Artikel gestolpert, der mir 2 Varianten anbietet:

a) über den bind der Packages aus der db2ubind.lst

db2 bind @db2ubind.lst datetime <FORMAT> blocking all grant public

wobei für <FORMAT> folgende Werte zulässig sind (Ausgabe basieren auf o.g. Datenbank Einstellungen und dem Statement `db2 "select datum from (select current date datum from sysibm.sysdummy1) as d"`):

LOC	Locale Installation	MM-DD-YYYY	11-25-2009
ISO	ISO-Standard	YYYY-MM-DD	2009-11-25
JIS	Japanese Industrial Standard	YYYY-MM-DD	2009-11-25
DEF	Date/Time passend zu Territory	MM/DD/YYYY	11/25/2009
USA	IBM Standard für U.S.	MM/DD/YYYY	11/25/2009
EUR	IBM Standard für Europa	DD.MM.YYYY	25.11.2009

b) über SQL mittels einer selbstdefinierten Funktion

Basierend auf diesen Artikel habe ich dann folgendes SQL zusammengestellt, um das Datumsformat im gewünschten Format auszugeben.

```
substr( digits (day(datum)),9) || '.' || substr( digits (month(datum)),9)
|| '.' || rtrim(char(year(datum)))
```

```
db2 "select substr( digits (day(datum)),9) || '.' || substr( digits (month(datum))
,9) || '.' || rtrim(char(year(datum))) from (select current date datum from
sysibm.sysdummy1) as d"

1
-----------------
```

```
25.11.2009
  1 record(s) selected.
```

Hinweis: Die Verwendung von digits und rtrim ist notwendig, um folgende Nebeneffekte der Konvertierung zu umgehen:

- nachführende Blanks durch char-Funktion
- Ziffern die kleiner als 10 sind, werden ohne führende Null angezeigt
  db2 "select datum, char(day(datum)) || '.' || char(month(datum)) || '.' || char(year(datum)) from (select date('1.10.1989') datum from sysibm.sysdummy1) as d"

```
DATUM      2
---------- ----------------------------------
10/01/1989 1          .10         .1989
  1 record(s) selected.
```

Nun sind aber beide Varianten ziemlich umständlich und ich habe weitergesucht und bin dann in der DB2 z/OS Online-Hilfe fündig geworden.

```
char(datum, <FORMAT>)
```

Wobei hier das <FORMAT> wieder das Format vom o.g. bind Befehl ist.
Diese Funktion gibt es sowohl in DB2 z/OS als auch in DB2 LUW.

```
> db2 "select datum, char(datum, EUR) from (select current date datum from
sysibm.sysdummy1)"


DATUM      2
---------- ----------
11/25/2009 25.11.2009

  1 record(s) selected.
```

Das ist nun ein Statement, mit dem man ganz gut leben kann, wenn man nicht die bind-Variante verwenden möchte.
Diese Funktion ist auch in der DB2 LUW Online-Hilfe zu finden.

Literatur:

- DB2 Basic: Fun with Dates and Times
- Online-Hilfe

Schulungen / Tagungen / Informationsveranstaltung

Eine Liste der anstehenden Konferenzen ist hier zu finden.

IOD 2009

Wenn Sie Interesse an den Vorträgen und Präsentationen der Information on Demand Konferenz haben, können Sie sich für die IOD Virtual Experience registrieren. Dort sehen Sie die Höhepunkte der Keynotes und können den Aufzeichnungen von sowohl technischen als auch geschäftsorientierten Sessions folgen. Registrieren unter Softwareforum

IOD 2010 – Vorankündigung

Chats mit dem Labor

Das Auftreten des Chat mit dem Lab hat sich verändert und hat eine direkte Verbindung zum db2 channel bekommen, um vorherige Chats zu wiederholen.

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

Analog zu den DB2 Chats gibt es auch noch BI Chats mit dem Labor. Die Präsentationen und Replays der Chats können hier angeschaut und heruntergeladen werden.

Newsletter Archiv

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

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

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 Subjekt „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
Wilfried Hoge	Technical Professional Data Management, IBM Sales & Distribution, Software Sales Artikel: DB2 9.7 Fixpak 1 verfügbar
Jürgen Buck	IT-Specialist DB2 & Informix IDS Artikel: Abschätzung der Performanceauswirkung von dynamischen SQL-Statements durch Reoptimierung
Frank Berghofer	IT-Spezialist für DB2 LUW, IBM SWG TechTipp: Erzeugen eindeutiger Schlüssel mit CURRENT TIMESTAMP, eine gute Wahl?

Reviewer und Ideenlieferanten:

Nela Krawez	IBM SWG, InfoSphere Balanced Warehouse Development
Wilfried Hoge	Technical Professional Data Management, IBM Sales & Distribution, Software Sales