Willkommen zum „IBM DB2 Newsletter“

Liebe Leserinnen und Leser,

Achtung! Der Zugang zu den Lab-Chats hat sich geändert (siehe Abschnitt „Chats mit dem Labor“)
Der Sommer ist da und die nächste DB2 Version steht in den Startlöchern. Für diese Version werden noch Tester gesucht (s.h. Abschnitt Neue DB2 Version steht an).

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

Ihr TechTeam

Chats mit dem Labor

Der letzte LabChat fand Anfang/Mitte Juli statt, zum Thema:

- Data Encryption: Protecting Your Sensitive Database Data

Eine Liste der bereits durchgeführten Chats ist zu finden unter:
http://www-306.ibm.com/software/data/db2/9/labchats.html
Die Präsentationen der Chats, können als pdf angeschaut und runtergeladen werden.

Neue DB2 Version steht an

Im August kommt die neue DB2 Version als BETA heraus. Für diese Version werden noch Unternehmen als BETA-Tester gesucht.
Sind sie interessiert und wollen an dem Programm teilnehmen, melden Sie sich entweder bei ihrem zuständigen DB2 Support oder schreiben sie uns eine email und wir werden es an die entsprechenden Kanäle weiterleiten.

Macht Table Partitioning Union-All-Views überflüssig?

Mit DB2 V9 wurde Table Partitioning als neue Methode eingeführt, um Datenbank-Tabellen zu organisieren, zusätzlich zum Database Partitioning (DPF) und zum Multi-Dimensional-Clustering (MDC). Table Partitioning – oft wird auch der Begriff „Range Partitioning“ dafür verwendet -- stellt damit eine Alternative zu den vorher für grob-granulares Partitionieren verwendeten Union-All-Views dar. Table Partitioning bietet viele der Vorteile von Union-All-Views nämlich einfaches und effizientes Entfernen und Hinzufügen von Daten und die Möglichkeit der Reduktion der bei einem Scan zu lesenden Daten schon in der Optimierungsphase. Gleichzeitig hat Table Partitioning auch noch den Vorteil gegenüber

Union-All-Views, dass die Query-Pläne wegen der geringeren Anzahl an beteiligten Tabellen deutlich weniger komplex sind. Damit stellt sich natürlich die Frage ob es damit überhaupt noch Situationen gibt, in denen Union-All-Views Vorteile gegenüber Table Partitioning bieten und deshalb bevorzugt werden sollen.

Ein wichtiger Vorteil von Union-All-Views ist die Möglichkeit die einzelnen Tabellen des Union-All-Views unterschiedlich indiziert zu haben. Weiterhin ist es natürlich möglich auch unterschiedliche MQTs (Materialized Query Tables) für die einzelnen Zweige eines Union-All-Views zu haben oder diese unterschiedlich physikalisch zu organisieren (mit unterschiedlichen MDC Definitionen).

Im folgenden werden wir uns aber auf die Möglichkeiten konzentrieren, die sich durch die unterschiedliche Indizierung von den einzelnen Tabellen eines Union-All-Views ergeben.

Beispiel:

Häufig hat man in Data Warehouses eine Tabelle `t`, die in regelmäßigen Abständen mit neuen Daten befüllt werden und aus der auch regelmäßig die ältesten Daten entfernt werden soll. Auch die Abfragen auf einer solchen Tabellen spezifizieren häufig einen Datumsbereich. Wenn `d` die Datumsspalte ist, nach der Daten geladen, entfernt und abgefragt werden, dann liegt es nahe diese Tabelle nach der Spalte `d` zu organisieren, entweder per MDC, per Table Partitioning oder eben per Union-All-View. Sollte es keine weiteren Anforderungen geben, ist normalerweise entweder ein MDC oder Table Partitioning die beste Lösung. Nehmen wir zusätzlich an, dass es in der Tabelle `t` eine weitere Spalte `c1` gibt, die häufig in Abfragen als einschränkende Bedingung vorkommt und deshalb indiziert werden soll. Nehmen wir weiterhin an, dass normalerweise täglich geladen wird und die meisten Datensätze für den aktuellen Tag sind, dass aber ein gewisser Anteil von Datensätzen als „Nachläufer" für frühere Tage kommen. Dies kann z.B. wenn man Fünf-Tage-Bereiche definiert über die drei unten angegebenen Tabellen `t1`, `t2` und `t3` realisiert werden. Jede dieser Tabellen hat eine Check-Constraint definiert. Alle Check-Constraints zusammen definieren den erlaubten Wertebereich für die Spalte `d` und alle Bedingungen definieren jeweils disjunkte Bereiche. Die Tabelle `t` wird dann durch den unten angegebenen Union-All-View realisiert. Die Indizes sind `i_t1_c1`, `i_t2_c1` und `i_t3_c1`.

```
CREATE TABLE t1 (
  d DATE,
  c1 INTEGER,
  c2 INTEGER,
  CONSTRAINT date_range
       CHECK (d >= '2008-01-01' AND d < '2008-01-05')
);

CREATE TABLE t2 (
  d DATE,
  c1 INTEGER,
  c2 INTEGER,
  CONSTRAINT date_range
       CHECK (d >= '2008-01-05' AND d < '2008-01-10')
);

CREATE TABLE t3 (
  d DATE,
  c1 INTEGER,
  c2 INTEGER,
  CONSTRAINT date_range
       CHECK (d >= '2008-01-10' AND d < '2008-01-15')
);

CREATE VIEW t AS (
  SELECT * FROM t1
  UNION ALL
  SELECT * FROM t2
  UNION ALL
```

```
  SELECT * FROM t3
);

CREATE INDEX i_t1_c1 ON t1(c1);
CREATE INDEX i_t2_c1 ON t2(c1);
CREATE INDEX i_t3_c1 ON t3(c1);
```

Wenn man nun am 12.1.2008 eine große Anzahl Rows laden möchte, wobei die meisten vom Vortag, wenige vom Vorvortag und nochmal deutlich weniger von anderen vorangegangenen Tagen sind, dann empfiehlt sich folgendes Vorgehen:

1. Zunächst wird der Index `i_t3_c1` gelöscht.
2. Dann können alle Rows mit einem Insert-Statement gegen den View `t` eingefügt werden. Dies hat zur Folge, dass die vielen Rows für den 10. und 11. Januar sehr schnell eingefügt werden, da kein Index beim Einfügen gepflegt werden muss. Die übrigen Rows werden gleichzeitig in die beiden indizierten Teiltabellen eingefügt, ohne dass eine spezielle Fallunterscheidung durchgeführt werden muss. Auch dies geht sehr schnell, da nur wenige Rows involviert sind.
3. Der Index `i_t3_c1` wird wieder gebaut.

Der Vorteil dieser Vorgehensweise ist, dass nicht täglich Ranges ab- und angehängt werden müssen, sondern dass nur jeweils ein Index gelöscht und neu angelegt werden muss, und dass keine Fallunterscheidung zwischen den verschiedenen Tagen gemacht werden muss, aber trotzdem der Performance-Vorteil einer solchen erzielt wird. Auch ist kein Merge von Indizes notwendig.

Eine andere Anwendungsmöglichkeit für Union-All-Views mit unterschiedlichen Indizes für die einzelnen Teilbereiche ergibt sich, wenn die Zugriffshäufigkeit stark variiert. Wenn man z.B. annimmt, dass in einer Tabelle Daten für zwanzig Jahre gehalten werden sollen, 99% der Zugriffe aber auf das aktuelle Jahr erfolgen sollen, dann bietet es sich an einen Union-All-View mit einer Teiltabelle für jedes Jahr zu definieren, aber nur das aktuelle Jahr zu indizieren. Abfragen müssen dann nicht explizit spezifizieren, ob sie alte oder neue Daten zugreifen wollen, Platz für Indizes wird nur für ein Zwanzigstel der Daten belegt und die durchschnittliche Abfrage-Performance sollte sehr gut sein.

TechTipp: Ermittlung Datum in Vergangenheit/Zukunft

Hier nun wieder ein Beitrag von einem DB2 Newsletter-Leser. Die Berechnung der Zeitstempel in der Vergangenheit bzw. Zukunft ist immer wieder eine notwendige Aufgabe.
Dazu gibt es verschieden Möglichkeiten:

- Timestamp/Datum aus der DB, hier für die nächste Woche
  - `db2 -x "select current timestamp + 7 days from (values 1) as h"`
    `2008-08-07-20.21.20.203000`
  - `db2 -x "values char(current date + 7 days)"`
    `2008-08-07`
- Timestamp/Datum aus der DB, für die vergangene Woche
  - `db2 -x "select current timestamp -7 days from (values 1) as h"`
    `2008-07-24-20.46.54.239000`
  - `db2 -x "values char(current date - 7 days)"`
    `2008-07-24`

…. oder auf der shell. (zumindest laeufts auf der ksh)

- Timestamp/Datum für nächsten Tag, gleiche Zeit
  - `TZ=GMT-26 date +"%y-%m-%d-%H.%M.%S.%s"`
    `08-08-01-21.24.06.1217532246`

- Timestamp/Datum für nächste Woche (dabei wurde festgestellt, das zwar der 7. Tag gefunden werden kann, jedoch die genaue Stunde konnte nicht angepeilt werden.
  - `$ TZ=GMT-167 date +"%y-%m-%d-%H.%M.%S.%s"`
    `08-08-07-18.20.35.1217532035`

- Timestamp/Datum für vorhergehenden Tag

  - `$ TZ=MEZ+26 date +"%y-%m-%d-%H.%M.%S.%s"`
    `08-07-30-17.25.37.1217532337`

  - `$ echo "$(TZ=EST26EDT date +%d-%m-%y)"`
    `30-07-08`

Hier nun noch ein Skript-Teil, mit dem der vorhergehende Tag berechnet werden kann:

```
#H-2 ---------------------------------------------------------------------------
#H-2 Funktion: lib_getPreviosDay
#H-2 Parameter: $1 .. OFFSET für die rückwärts Berechnung (Default=1 => yesterday)
#H-2 Beschreibung: Mit dieser Funktion werden Vortage (gestern ($1=1);
#H-2               vorgestern ($2=2) usw ermitteln. Für SEVENDAY ist $1=7
#H-2               Die Funktion gibt die Ausgabe in $val_getPreviousDay zurück
#H-2               wenn $2 leer ist, sonst auf stdout
#H-2 Quelle:
#H-2         Diese Funktion wurde aus dem Internet gezogen und entsprechend
#H-2         der Anforderungen angepaßt
#H-2 ---------------------------------------------------------------------------
getPreviosDay()
{
 OFFSET=${1:-1}
 case $OFFSET in
   *[!0-9]* | ???* | 3? | 29) echo "Invalid input" ; return 1;;
 esac

 case "$SHELL" in
  *"ksh"*)
     ##!/bin/ksh
     #yesterday=$(TZ=EST26EDT date +%m-%d-%y); echo $yesterday

     eval `date "+day=%d; month=%m; year=%Y"`

    # Subtract offset from day, if it goes below one use 'cal'
    # to determine the number of days in the previous month.
    day=$((day - OFFSET))
    if (( day <= 0 )) ;then
       month=$((month - 1))
       if (( month == 0 )) ;then
          year=$((year - 1))
          month=12
       fi
       set -A days `cal $month $year`
       xday=${days[$(( ${#days[*]}-1 ))]}
       day=$((xday + day))
    fi

    val_getPreviosDay=${year}$month$day
    ;;
  *)
    #C- 20070301 nachfolgendes Konstrukt arbeitet wunderbar auf aix aber nicht in linux
    (( MEZ_1 = OFFSET * 24 - 1 ))
    (( MESZ_1 = OFFSET * 24 - 2 ))
    val_getPreviosDay="`TZ=MEZ+$MEZ_1-MESZ+$MESZ_1 date +\"%Y%m%d\"`"

    #C- Die SQL-Abfrage liefert nur ein Datum zurück, Bsp. 20070104
```

```
    #C-  SEVENDAYS=`db2 -x "values substr(char(current date - 7 days),7,4) concat
substr(char(current date - 7 days),1,2) concat substr(char(current date - 7 days),4,2)"
`
    ;;
 esac
}
```

Redbooks/Whitepapers u.a.

- Im Juni wurde folgendes Redbook veröffentlicht:
  Up and Running with DB2 on Linux
  http://www.redbooks.ibm.com/abstracts/sg246899.html?Open

Erweiterte Eigenschaften zur Migration ORACLE zu DB2

Oracle to DB2 Conversion Guide for Linux, UNIX, and Windows, SG24-7048-01

In DB2 9.5 für LUW gibt es neue Eigenschaften, die es einfacher machen eine ORACLE Datenbank und deren Applikationen nach DB2 zu migrieren.

Dabei werden z.B. folgende Funktionen unterstützt, wie DECODE, OUTER JOIN, ROWNUM, sowie neue Umsetzungsalgorithmen für ARRAY, BULK COLLEC. Detailliertere Informationen sind im folgenden Artikel der developerWorks zu finden:
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0707rielau/

Schulungen/Veranstaltungen

DB2 Aktuell 2008

Wie bereits in der DB2 Newsletter Ausgabe vom Juni beschrieben, findet im September die DB2 Aktuell Veranstaltung in Potsdam statt.

23. + 24. September in Potsdam

Nähere Informationen und Anmeldemöglichkeit sind zu finden unter:
http://www-05.ibm.com/services/learning/de/ta-iris.nsf/(ExtCourseNr)/CFSYD0DE

Zertifizierungskurs

Einen Tag vor der DB2 Aktuell wird ein Zertifizierungskurs für DB2 Family Fundamentals durchgeführt.
Nähere Informationen und Anmeldung im Internet unter folgenden Adressen möglich:

http://www-05.ibm.com/services/learning/de/ta-iris.nsf/(ExtCourseNr)/CEZ1D1DE

In diesem Kurs werden die Teilnehmer auf die Zertifizierung 730 vorbereitet. Die Zertifizierung kann dann anschließend während der DB2 Aktuell durchgeführt werden.

Newsletter Archiv

Alte Ausgaben vom DB2-NL sind nun zum Nachlesen im Archiv von BYTEC zu finden:
https://www.bytec.de/de/software/ibm_software/newsletter/db2newsletter/

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 UDB, IBM SWG; djs@de.ibm.com
Andreas Weininger	IBM Sales & Distribution, Software Sales Artikel: Macht Table Partitioning Union All Views überflüssig

Reviewer und Ideenlieferanten:

Nela Krawez	IBM SWG, IT- Spezialist
Wilfried Hoge	IBM SWG, Technical Sales
Volker Fränkle	IBM SWG, IT-Spezialist
Gerold Müller	T-Systems Artikel: Ermittlung Datum in Vergangenheit/Zukunft

IBM SWG IM Services - Ihr direkter Draht zur Produktentwicklung

Der Ihnen vorliegende Newsletter wird durch "IBM Software Group Information Management Services" herausgegeben. Die Artikel werden in Zusammenarbeit mit der Produktentwicklung und dem Support erstellt und publiziert. All dies dürfte Ihnen sicherlich bekannt sein.

Aber wissen Sie, dass das Team des IBM SWG IM Services tief gehende Expertise zu folgenden Spezialthemen hat?

- Hochverfügbarkeit
- Replikation
- Performance
- Security

Diese Expertisen werden auf Tagesbasis abgerufen. Informationen zu dieser Expertise erhalten Sie über Herrn Jens Krumbiegel (krumbieg@de.ibm.com). Informieren Sie sich auch über unsere exklusiven Business-Partner Expertisen.