Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

der Herbst zeigt sich von seiner besten Seite und die Zeit der Jahrmärkte ist gekommen. Nun beginnt auch die Zeit, um sich etwas zurückzulehnen und die Datenbankinstanz selbst für ihre Optimierung und Anpassung sorgen zu lassen. Nachdem die Chunks bereits seit einiger Zeit in der Lage sind, sich selbst zu vergrössern (extendable), ist dies nun auch für den BUFFERPOOL möglich. Mit dem Auto-Tuning weiterer Parameter passt sich das System nun immer mehr den momentanen Anforderungen an. Nutzen Sie die Zeit um sich mit den Vorteilen des Internet of Things vertraut zu machen, damit Sie vorbereitet sind, wenn diese Technologie Einzug hält.

Wie immer haben wir für Sie eine Reihe an Tipps und Tricks zusammengestellt. Viel Spaß mit den Tipps der aktuellen Ausgabe.

Ihr TechTeam

TechTipp: Informix und NoSQL (4): SQL-Zugriff auf NoSQL-Daten

In der letzten Ausgabe des Newsletters haben wir NoSQL-Daten mit den NoSQL-Befehlen der Mongo Shell manipuliert und haben dabei zum Schluss die Collection "rating" wieder entfernt. Als Vorbereitung legen wir daher in der Datenbank "stores_demo" diese Collection zuerst wieder an und füllen sie mit Daten. Hierzu verwenden wir nocheinmal die schon bekannten Befehle der Mongo Shell:

```
 $ mongo localhost:26351/stores_demo
MongoDB shell version: 2.4.9
connecting to: localhost:26351/stores_demo
>
  db.rating.insert({"catalog_num": 10017, "rating_value": 7})
  db.rating.insert({"catalog_num": 10017, "rating_value": 9})
  db.rating.insert({"catalog_num": 10019, "rating_value": 7})
  db.rating.insert({"catalog_num": 10017, "rating_value": 5})
  db.rating.insert({"catalog_num": 10017, "rating_value": 8})
  db.rating.insert({"catalog_num": 10019, "rating_value": 3})
  db.rating.insert({"catalog_num": 10017, "rating_value": 7})
  db.rating.insert({"catalog_num": 10017, "rating_value": 8})
  db.rating.insert({"catalog_num": 10017, "rating_value": 4})
  db.rating.insert({"catalog_num": 10019, "rating_value": 8})
  db.rating.insert({"catalog_num": 10019, "rating_value": 3})
  db.rating.insert({"catalog_num": 10017, "rating_value": 8})
>
> db.rating.count()
12
>
```

Die Collection "rating" enthält nun 12 Dokumente, und wir können uns mit "dbaccess" zur Datenbank "stores_demo" verbinden, um mit SQL-Befehlen auf diese Daten zuzugreifen. In einem ersten Versuch fordern wir nur den ersten Datensatz von der Tabelle "rating" mit einem einfachen SQL-SELECT Befehl an:

```
$ dbaccess stores_demo -
Database selected.
> SELECT FIRST 1 * FROM rating;

id        54007eb8db1b7464013bcea2
data      A
modcount  0
flags     0

1 row(s) retrieved.
>
```

Dieses Ergebnis entspricht wahrscheinlich nicht ganz unseren Vortellungen. In der Spalte "id" sehen wir den automatisch erzeugten Identifikations-String, den wir von den Beispielen mit NoSQL-Befehlen schon kennen. Die Daten selbst sind aber in der Spalte "data" versteckt. Um die Daten zu sehen, müssen wir auf die entsprechenden Felder der Spalte "data" zugreifen und einen Cast zum Datentyp anwenden, den wir für das jeweilige Feld erwarten. In unserem Beispiel erwarten wir für beide Felder jeweils einen Integer, so dass wir die Abfrage wie folgt formulieren können:

```
> SELECT
  data.catalog_num::INT,
  data.rating_value::INT
FROM rating;

(expression) (expression)

       10017            7
       10017            9
       10019            7
       10017            5
       10017            8
       10019            3
       10017            7
       10017            8
       10017            4
       10019            8
       10019            3
       10017            8

12 row(s) retrieved.
>
```

Das Ergebnis der Abfrage sieht nun schon recht brauchbar und wie von einer normalen SQL-Tabelle aus. Die Spaltenüberschriften sind "(expression)", was mit der expliziten Angabe eines Labels für die Spalten in der Projektionsliste leicht verbessert werden kann.

Mit dieser Abfragemethode für NoSQL Collections können wir die Daten nun auch mit denen einer regulären SQL-Tabelle kombinieren. Da wir uns unter der Katalognummer alleine nicht so gut vorstellen können, um welches Produkt es sich jeweils handelt, wollen wir im Ergebnis der Abfrage auch die Artikelbeschreibung sehen. Hierzu müssen wir - dem Schema der "stores_demo" Datenbank entsprechend - einen Join mit den SQL-Tabellen "catalog" und "stock" erstellen. Die Katalognummer ist das Bindeglied zwischen der NoSQL Collection "rating" und der SQL-Tabelle "catalog".

Da die Kurzbeschreibung der Artikel in der Spalte "description" der SQL-Tabelle "stock" enthalten ist, kombinieren wir die Tabelle "catalog" mit der Tabelle "stock" durch einen weiteren Join über die Spalten "stock_num" und "manu_code". Schliesslich wollen wir das Ergebnis der Abfrage zuerst anhand der Katalognummer und nachrangig anhand der Bewertung sortieren. Die dafür nötige Abfrage sieht dann so aus:

```
> SELECT
    c.catalog_num AS catalog_number,
    s.description AS description,
    r.data.rating_value::INT AS rating
  FROM catalog c, stock s, rating r
  WHERE r.data.catalog_num::INT = c.catalog_num
    AND c.stock_num = s.stock_num
    AND c.manu_code = s.manu_code
  ORDER BY
    catalog_number,
    rating
  ;

catalog_number description          rating

         10017 bicycle tires             4
         10017 bicycle tires             5
         10017 bicycle tires             7
         10017 bicycle tires             7
         10017 bicycle tires             8
         10017 bicycle tires             8
         10017 bicycle tires             8
         10017 bicycle tires             9
         10019 bicycle brakes            3
         10019 bicycle brakes            3
         10019 bicycle brakes            7
         10019 bicycle brakes            8

12 row(s) retrieved.

>
```

Wir sind also in der Lage, ohne grossen Aufwand die Daten der NoSQL Collection mit den Daten in den normalen SQL-Tabellen zu kombinieren. Störend ist hierbei allerdings die etwas umständliche Syntax für den Zugriff auf die Felder der Spalte "data" und der jeweils nötige Cast.

Beide Umständlichkeiten können wir eliminieren, indem wir einen entsprechenden View definieren:

```
> CREATE VIEW rating_v (
    catalog_num,
    rating_value)
  AS SELECT
     data.catalog_num::INT,
     data.rating_value::INT
  FROM rating;

View created.
>
```

Benutzen wir nun für den Zugriff auf die NoSQL-Daten diesen View "rating_v" anstatt der Tabelle "rating", dann können wir die Abfrage aus obigem Beispiel in regulärer SQL-Syntax formulieren, ohne etwas von der No-SQL Collection und ihrer Besonderheiten wissen zu müssen:

```
> SELECT
    c.catalog_num AS catalog_number,
    s.description AS description,
    r.rating_value AS rating
  FROM catalog c, stock s, rating_v r
  WHERE r.catalog_num = c.catalog_num
    AND c.stock_num = s.stock_num
    AND c.manu_code = s.manu_code
  ORDER BY
    catalog_number,
    rating
  ;

catalog_number description          rating
         10017 bicycle tires             4
         10017 bicycle tires             5
         10017 bicycle tires             7
         10017 bicycle tires             7
         10017 bicycle tires             8
         10017 bicycle tires             8
         10017 bicycle tires             8
         10017 bicycle tires             9
         10019 bicycle brakes            3
         10019 bicycle brakes            3
         10019 bicycle brakes            7
         10019 bicycle brakes            8

12 row(s) retrieved.
>
```

Mit geringem administrativem Aufwand konnten wir nicht nur NoSQL-Befehle verwenden, um NoSQL-Daten zur SQL-Datenbank "stores_demo" hinzuzufügen, sondern diese Daten auch gleich so integrieren, dass sie in Abfragen ohne besondere SQL-Syntax nahtlos mit den Daten der regulären SQL-Tabellen kombiniert werden können. Dafür war lediglich die Definition des entsprechenden Views "rating_v" vonnöten.

Allerdings bleibt der konzeptionelle Unterschied zwischen der NoSQL Collection und den SQL-Tabellen erhalten. Die SQL-Tabellen sind nach einem Schema mit einer festen Anzahl von Spalten mit einem bestimmten Datentyp definiert. Dies gilt natürlich auch für den View "rating_v", den wir ja mit SQL-Befehlen definiert haben. Die NoSQL Collection hingegen ist schemalos, d.h. wir können mittels NoSQL-Befehlen der Mongo Shell jederzeit ein neues Dokument in die Collection einfügen, welches nicht dem Schema des erstellten Views "rating_v" entspricht:

```
$ mongo localhost:26351/stores_demo
MongoDB shell version: 2.4.9
connecting to: localhost:26351/stores_demo
> db.rating.insert({"catalog_num": 10019, "rating_value": 9,
    "comment": "A very simple to use product."})
>

$ dbaccess stores_demo -

Database selected.
> SELECT * FROM rating_v
    WHERE catalog_num = 10019 ORDER BY rating_value;

catalog_num rating_value

      10019             3
      10019             3
      10019             7
      10019             8
      10019             9

5 row(s) retrieved.
>
```

Wir sehen mit der SQL-Abfrage zwar prinzipiell den neuen Datensatz ("rating_value" = 9), jedoch nicht den Kommentar, den wir mittels NoSQL-Befehl im Dokument mit eingefügt haben. Um ein solches Dokument dann in seiner Gänze mit SQL-Befehlen verarbeiten zu können, müssen wir den View "rating_v" entsprechend neu definieren:

```
> DROP VIEW rating_v;
View dropped.
```

```
> CREATE VIEW rating_v (catalog_num, rating_value, comment) AS
  SELECT
    data.catalog_num::INT,
    data.rating_value::INT,
    data.comment::LVARCHAR
  FROM rating;

View created.

> SELECT * FROM rating_v
    WHERE catalog_num = 10019 ORDER BY rating_value;

catalog_num   10019
rating_value  3
comment

catalog_num   10019
rating_value  3
comment

catalog_num   10019
rating_value  7
comment

catalog_num   10019
rating_value  8
comment

catalog_num   10019
rating_value  9
comment       A very simple to use product.

5 row(s) retrieved.
>
```

Wie wir sehen, hat die Spalte "comment" für die Datensätze den Wert NULL, wo die entsprechenden Dokumente ohne das key-value-Paar {"comment": "..."} in die Collection eingefügt wurden. Dies ist ein sinnvolles Verhalten. In der neuen Definition des Views haben wir die Spalte "comment" mit einem Cast zum Datentyp LVARCHAR angegeben, so dass auch evtl. längliche Kommentare sicher nicht abgeschnitten werden.

Zum Abschluss dieses Artikels sei noch erwähnt, dass es auch die Möglichkeit gibt, in einem SQL-SELECT das Ergebnis in einem JSON-ähnlichen Format anzeigen zu lassen. Hierzu verwenden wir für die Felder der Spalte "data" einen Cast zu "JSON":

```
> SELECT
      data.catalog_num::JSON AS catalog_num,
      data.rating_value::JSON AS rating_value
    FROM rating
    WHERE catalog_num = 10017
    ORDER BY
      catalog_num,
      rating_value
  ;

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":4.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":5.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":7.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":7.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":8.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":8.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":8.000000}

catalog_num   {"catalog_num":10017.000000}
rating_value  {"rating_value":9.000000}

12 row(s) retrieved.

>
```

Hinweis:
Alle Beispiele wurden mit Informix Version 12.10.xC4 (auf Linux x86 64-bit) erprobt.

In der nächsten Newsletterausgabe wollen wir im Gegenzug versuchen, mittels NoSQL-Befehlen der Mongo Shell auf die Daten der SQL-Tabellen zuzugreifen.

TechTipp: PHYSLOG mittels onspaces in eigenem DBSpace erstellen

Ab der Version 12.10.xC4 gibt es die Möglichkeit mittels „onspaces -c -P“ das Physical Log direkt in einen neuen DBSpace zu verschieben, der durch diesen Befehl zugleich angelegt wird.

Die Syntax entspricht dem Befehl, der beim Anlegen eines herkömmlichen DBSpaces genutzt wird. Hierbei wird nur der Paramter „d“, der üblicherweise dem „-c“ folgt, durch ein „P“ ersetzt. Implizit erstellt der Aufruf zuerst den DBSpace und verschiebt anschliessend das Physical Log in diesen neu erstellten DBSpace.
Die Syntax lautet:

```
onspaces -c -P <dbspacename> -p <Datei> -o <offset> -s <Grösse>
```

Beispiel:

```
onspaces -c -P plogdbs -p /IFX/plogdbs -o 0 -s 1000000
```

Im Messagelog sind anschliessend die Einträge zur Änderung zu sehen:

```
09:42:13  Physical log location changed to dbspace plogdbs.
09:42:13  Physical log size changed to 999894 KB.
```

Da das Physical Log ohne Downtime verschoben werden kann, ist dieser Aufruf im laufenden Betrieb möglich.

Der neu angelegte DBSpace ist im „onstat -d“ mit dem Flag „P“ als Kennung zu sehen. Der zugehörige Chunk wird automatisch als „extendable“ gekennzeichnet und die definierte Grösse wird komplett vom Physical Log ausgefüllt, so dass bei „free“ die Angabe „0 Pages“ steht:

```
Dbspaces
address  number flags     fchunk nchunks  pgsize   flags    owner    name
…
47734c20 7      0x1070001 7      1        2048     N PBA    informix plogdbs

Chunks
address     chunk/dbs  offset  size       free    bpages   flags pathname
…
478a5028    7      7   0       500000     0                PO-BED /IFX/plogdbs
```

Für den PlogDBSpace gelten folgende Einschränkungen:

- er kann nur aus einem Chunk bestehen
- er kann nur das Physical Log beinhalten
- er hat die selbe Pagesize wie der Rootdbspace
- es kann nur einen PlogDBSpace geben

TechTipp: Extendable BUFFERPOOL

Der BUFFERPOOL, der die Grösse des Speichers für das Caching der Daten und Indices angibt, war bisher durch den Parameter BUFFERPOOL in der Konfigurationsdatei $ONCONFIG bestimmt und konnte nach dem Start der Instanz nicht mehr verändert werden. Je Pagesize gibt es einen Bufferpool. Bei der Konfiguration einer Instanz mit DBSpaces unterschiedlicher Pagesize (z.B. 2k, 4k, 8k) musste man abschätzen, wie stark die jeweiligen Bereiche genutzt werden, um hier eine optimale Aufteilung der vorhandenen Resourcen zu erzielen.
Mit Version 12.10.xC4 bekommt der Parameter BUFFERPOOL zusätzliche Argumente, die eine dynamische Vergrösserung und ein limitiertes Wachstum ermöglichen.
Grundsätzlich gibt es zwei unterschiedliche Möglichkeiten der Konfiguration.
Die erste Möglichkeit benutzt die Buffers (Pages), wie dies in eingeschränktem Umfang bisher bereits möglich war. Die zweite Möglichkeit bezieht sich direkt auf die Speichergröße und benutzt die üblichen Einheiten (B, MB, GB).
Beiden Möglichkeiten gemeinsam sind die Parameter für Pagesize (**size=**) und LRUs (**lrus=...,lru_min_dirty=...,lru_max_dirty=...**).
Ebenfalls gemeinsam ist der Paramter „**extendable**“, der per Default auf 0 (disabled) steht, und der beim Wert 1 eine dynamische Anpassung des Bufferpools zulässt, sowie der Parameter „**cache_hit_ratio**“, der angibt, unter welchem Mindestwert des „Read Cache“ der Bufferpool erweitert werden soll. Der Defaultwert für „cache_hit_ratio“ ist 90.

Betrachten wir zuerst die Optionen für die zweite Möglichkeit. Hier können die Parameter „start_memory“ und „memory“ gesetzt werden. Der Default für „**start_memory**“ ist 32mb. Der Wert kann mit „auto“, oder einem Festen Wert in Byte (ohne Einheitsangabe), Kilobyte (kb), Megabyte (mb) oder Gigabyte (gb) angegeben werden.
Der Wert für „**memory**“ gibt die maximale Speichergröße an, die für diesen Bufferpool genutzt werden darf. Die Angabe kann mit konkreten Werten wie bei „start_memory“ erfolgen, oder auf „auto“ (default) gesetzt werden, wodurch die Speichergröße abhängig von der „Server Size“ (siehe folgender Artikel) gesetzt wird.

Beispiel zu Bufferpool mit der Option „memory“:

```
BUFFERPOOL size=2k,start_memory=8mb,memory=4gb,extendable=1,cache_hit_ratio=94
```

Wurde ein Bufferpool mit der Option „extendable“ erstellt, so zeigt die Ausgabe des Befehls „onstat -g buf“ die Einstellungen und ggf. die Anzahl oder Grösse der Erweiterungen in einem eigenen Abschnitt an.

onstat -g buf:

```
Buffer pool page size: 2048
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits
%cached
123        1169       692        82.23   28         47         22         0.00
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
22                  0          0          1

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
0             0             -nan          16            352Mb

                                    cache
# extends  max memory  next memory  hit ratio  last
0          4096Mb      352Mb        94         14:42:10

Bufferpool Segments
id segment      size     # buffs
0  0x47db5000   352Mb    129053

----------------------------------

Fast Cache Stats
gets       hits       %hits   puts
14         11         78.57   9
```

Wird der Bufferpool hingegen mit dem Parameter „buffers“ spezifiziert, so gibt es zum Parameter „**buffers=...**“, die Optionen

„**next_buffers=...**“ für die Größe der Erweiterungen und

„**max_extends=...**“ für die maximale Anzahl der Erweiterungen des Bufferpools.

Beispiel zu Bufferpool mit der Option „buffers“:

```
BUFFERPOOL size=2k,extendable=1,buffers=2000,max_extends=2,next_buffers=2000,...
      ...cache_hit_ratio=94,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

onstat -g buf:

```
Buffer pool page size: 2048
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits
%cached
747        1161       6590       88.66   29         42         626        95.37
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
626                 31         0          1

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
0             0             -nan          11            4Mb

                                       cache
# extends  max extends  next buffers  hit ratio   last
0          2            2000          94          14:39:05
```

```
Bufferpool Segments
id segment      size      # buffs
0  0x47db5000   4Mb       2001

---------------------------------

Fast Cache Stats
gets        hits        %hits    puts
1232        1103        89.53    1413
```

Achtung:
Alle Einträge zum Parameter BUFFERPOOL in der $ONCONFIG müssen einheitlich mit „memory“ oder „buffers“ angegeben sein. Dies gilt auch für den Eintrag mit Pagesize „default“.
Ansonsten kann die Instanz nicht starten und man erhält den Fehler:
ERROR: Cannot mix buffer arguments with memory arguments. (BUFFERPOOL)

Fällt nun der Wert für „Read Cache“ in einer Instanz, deren Bufferpool sich vergrössern kann, für längere Zeit unter den eingestellten Wert, so erweitert sich der Bufferpool in Abständen von mindestens 5 Minuten automatisch. Im Messagelog sind dann Einträge zu sehen:

Beispiel:

```
11:40:57  ** AUTO TUNING - Extending bufferpool 2K.
11:40:57  Requested shared memory segment size rounded from 4332KB to 4336KB
11:40:57  Dynamically allocated new bufferpool shared memory segment (size
4336KB)
11:40:57  Memory sizes:resident:30532 KB, virtual:50076 KB, no SHMTOTAL limit
11:40:57  Extended bufferpool 2K
        4332K memory, 2001 buffers
```

Wird die eingestellte Grenze (max_entends, bzw. memory) erreicht, so kann sich der Bufferpool trotz Bedarf nicht mehr erweitern und es wird eine Meldung in das Messagelog eingetragen:
Beispiel für das Erreichen des Limits „max_extends=2)

```
12:25:57  Performance Advisory: Unable to extend bufferpool 2K.
12:25:57   Results: Bufferpool has reached the # of extends '2' allowed.
12:25:57   Action: Increase the amount of memory the bufferpool can utilize.
```

TechTipp: ONCONFIG - AUTO_TUNE_SERVER_SIZE

Dieser Parameter ist in der Datei „onconfig.std“ nicht vertreten. Wird eine Instanz bei der Installation automatisch erstellt (Option im Installationsmenu), dann wird anhand der eingegebenen Werte aus dem Installationsdialog ein Wert für AUTO_TUNE_SERVER_SIZE ermittelt. Dieser ist dann die Grundlage für eine Reihe von Parametern. Wird der Wert nach der ersten Initialisierung des Servers verändert, so werden nur noch die Werte für den extendable Bufferpool (bei memory=auto) und die maximale Anzahl zusätzlicher logischer Logs bei Nutzung des Parameters AUTO_LLOG (siehe nächster Artikel) beeinflusst.

Default für AUTO_TUNE_SERVER_SIZE ist OFF.
Mögliche Werte sind:

- SMALL — für 1 - 100 Anwender, die mit der Informix Instanz arbeiten
- MEDIUM — für 101 – 500 Anwender
- LARGE — für 501 – 1000 Anwender
- XLARGE — für mehr als 1000 Anwender

Bei der Initialisierung während der Installation werden anhand dieses Parameters die Werte für den BUFFERPOOL, der maximale Platz für die logischen Logs, die initiale Grösse des Logdbspace, des Plogdbspace und des DatenDBSpaces, sowie die Grösse des des Physical Logs, des TempDBSpace, des SmartBlobSpace, sowie des temporären SmartBlobSpace bestimmt.

Anmerkung der Redaktion:
Laut Dokumentation wird bei „LARGE“ bis zu 33% des verfügbaren Memory für den Bufferpool verwendet, 500 MB Plattenplatz für den DBSpace der Daten und bis zu 1 GB für die Logs. Da „Grösse“ anscheinend immer relativ ist, können wir den Parameter für die erste Initialisierung nicht uneingeschränkt empfehlen. Manuelle Erstellung der DBSpaces und Logs nach Bedarf und basierend auf Erfahrung ist in jedem Fall besser.
Jedes System hat andere Anforderungen und Datenbestände, so dass hier eine pauschale Abschätzung kaum die optimale Konfiguration darstellen kann.

Für die Verwendung im Zusammenhang mit dem „extendable Bufferpool“ wird in den meisten Fällen die Einstellung „XLARGE“ die beste Wahl sein.

TechTipp: ONCONFIG - AUTO_LLOG

Dieser Parameter ist in der Datei „onconfig.std“ nicht enthalten. Der Parameter kann mittels „onmode -wf / -wm“ gesetzt, bzw. verändert werden.
Der Parameter ist im Default deaktiviert, somit werden keine logischen Logs automatisch hinzugefügt. Ist in der Konfigurationsdatei AUTO_TUNE auf 1 gesetzt, so kann dieser Parameter bei Bedarf dafür sorgen, dass zusätzliche logische Logs automatisch angelegt werden.
Syntax:

AUTO_LLOG 1,<logdbs>,<[max_size_kb]>

Der erste Parameter gibt an, ob das Feature aktiviert ist (1) oder deaktiviert (0).
Der zweite Parameter gibt den DBSpace an, in dem zusätzliche logische Logs erstellt werden sollen.
Der dritte Parameter ist optional und gibt die Maximale Grösse aller logischen Logs in kB an. Wird dieser Wert nicht gesetzt, so wird die Grenze aus AUTO_TUNE_SERVER_SIZE genutzt, die bei XLARGE 2 GB beträgt.

TechTipp: Extents der Systemtabellen

Die Überwachung der Anzahl der Extents von Tabellen und Indexe ist für den Administrator eine Routineaufgabe, die mit Hilfe der Tasks „defragment“, „repack“ und „shrink“ einfacher geworden ist. In den meisten Fällen wird bei der Erstellung des Datenbankschemas bereits eine angepasste Extentsize und Nextsize bei der Erstellung der Tabellen (und in aktuellen Versionen auch bei Indexen) von den Entwicklern mitgegeben, so dass sich der Reorganisationsaufwand in Grenzen hält.
Bei der Überwachung der Anzahl der Extents fallen immer wieder die Systemtabellen negativ auf, deren Defaulteinstellung von 8 Pages Extent Size und 8 Pages Next Size dazu führen, dass hier dutzende Extents genutzt werden müssen.
Um dies zu verhindern, sollten direkt nach dem Erstellen der Datenbank, noch vor der Erstellung der Tabellen und Indexe, die Nextsize der Systemtabellen angepasst werden.

Wird eine Datenbank mittels „dbimport“ eingespielt, so können diese Zeilen in das SQL-Script im Exportverzeichnis eingefügt werden.

Beispiel für die Erweiterung der Nextsize von Systemtabellen in einer durchschnittlichen OLTP-Datenbank:

```
alter table syscoldepend        modify next size 10000;
alter table syscolumns          modify next size 10000;
```

```
alter table sysconstraints      modify next size 10000;
alter table sysdefaults         modify next size  2000;
alter table sysdistrib          modify next size 10000;
alter table sysfragments        modify next size  1000;
alter table sysindices          modify next size  4000;
alter table sysobjstate         modify next size 10000;
alter table sysprocauth         modify next size  1000;
alter table sysprocbody         modify next size  1000;
alter table sysproccolumns      modify next size  1000;
alter table sysprocedures       modify next size  1000;
alter table sysprocplan         modify next size  1000;
alter table systabauth          modify next size  1000;
alter table systables           modify next size  2000;
alter table systrigbody         modify next size  1000;
```

WebTipp: INFORMIX entwickelt sich zur Standard-Datenbank in China

China, einer der grössten Märkte der Welt, hat sich für INFORMIX als Datenbankstandard für neue Softwareprojekte entschieden. Dieses Abkommen bedeutet, dass die Verbreitung von INFORMIX sich in den kommenden Jahren massiv ausweiten kann.
Asien und Südamerika waren traditionell immer starke Regionen bei der Verbreitung von INFORMIX. Viele weltweite Projekte wurden in diesen Ländern angestossen.
Mehr Informationen dazu finden Sie unter folgenden Links:
http://informix-myview.blogspot.co.uk/2014/11/china-is-standardizing-all-new-database.html
http://www.itsw.at/2014/11/03/informix-to-become-the-national-database-in-china/

WebTipp: The Internet of Things and Informix

Auf der Insight2014 wurde in einigen Sessions das Zusammenspiel von INFORMIX im Umfeld von Cloud, NoSQL und Warehouse beleuchtet. Einen Bericht dazu finden Sie unter:
http://www.ibmbigdatahub.com/blog/internet-things-and-informix

Termin: Industrie 4.0 und Internet of Things Webinar am 03.12.2014

In Industrie 4.0 und Internet of Things (IoT) Projekten sind Sensordaten überall und fallen in so großen Mengen an, so daß'klassische' Datenbanksysteme damit häufig überfordert sind. Die zeitreihenbasierten Sensordaten kommen sowohl strukturiert, als auch immer häufiger unstrukturiert vor (z.B. in Form von JSON Dokumenten).

Im Rahmen des Webinars - am **03.12.2014** um **9.30 Uhr** - wird in die Problematik der Sensordatenverarbeitung eingeführt und es werden die Vor- und Nachteile verschiedener Ansätze (Relational, NoSQL usw.) diskutiert. Im weiteren Verlauf des Vortrags wird eine im Markt weitgehend einmalige hybride IBM Datenbanktechnologie vorgestellt, die es erlaubt Sensordaten schnell, effizient und hochskalierbar sowohl in der Cloud (z.B. in IBMs Bluemix und in der IBM Internet of Things Foundation), als auch embedded in sogenannten 'Edge Devices' zu verarbeiten.

Der Vorteil: Sie können bequem vom Schreibtisch aus teilnehmen und sich mit anderen Teilnehmern sowie IBM Experten austauschen. Die Webinare dauern 45 Minuten.
Die IBM Software Academy finden Sie unter:http://www.ibm.com/de/events/software-academy/

Live-Webinar "**Industrie 4.0**/Internet of Things und **Sensordaten**: Kleine Ursache - Große Auswirkungen - Sind Sie bereit?" am **03.12.2014 - 9.30 - 10.15 Uhr**
https://ibm.biz/BdEfwd

Versionstipp: Version 12.10.xC4 durch 12.10.xC4W1 ersetzen

Unter besonderer Konstellation kann es dazu kommen, dass in Version 12.10.xC4 unnötig viel Speicher für die Sessions der Datenbank gebunden wird. Dieses Problem wurde mit dem Release 12.10.xC4W1 behoben, so dass ein Update auf dieses Release empfohlen wird.

Anmeldung / Abmeldung / Anmerkung

Der Newsletter wird ausschließlich an angemeldete Adressen verschickt. Die Anmeldung erfolgt, indem Sie eine Email mit dem Betreff „**ANMELDUNG**“ an **ifmxnews@de.ibm.com** senden.
Im Falle einer Abmeldung senden Sie „ABMELDUNG“ an diese Adresse.

Das Archiv der bisherigen Ausgaben finden Sie zum Beispiel unter:

http://www.iiug.org/intl/deu
http://www.iug.de/index.php?option=com_content&task=view&id=95&Itemid=149
http://www.informix-zone.com/informix-german-newsletter
http://www.drap.de/link/informix
http://www.nsi.de/informix/newsletter
http://www.cursor-distribution.de/index.php/aktuelles/informix-newsletter
http://www.listec.de/Newsletter/IBM-Informix-Newsletter/View-category.html
http://www.bereos.eu/software/informix/newsletter/

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit. Da uns weder Tippfehler noch Irrtümer fremd sind, bitten wir hier um Nachsicht falls sich bei der Recherche einmal etwas eingeschlichen hat, was nicht wie beschrieben funktioniert.

Die Autoren dieser Ausgabe

Gerd Kaluzinski IT-Specialist Informix Dynamic Server und DB2 UDB
IBM Software Group, Information Management
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Martin Fuerderer IBM Informix Entwicklung, München
IBM Software Group, Information Management
martinfu@de.ibm.com

Markus Holzbauer IBM Informix Advanced Support
IBM Software Group, Information Management Support
holzbauer@de.ibm.com

Die Versionsinfo stammt aus dem Versions-Newsletter der CURSOR Software AG
http://www.cursor-distribution.de/download/informix-vinfo

Sowie unterstützende Teams im Hintergrund.

Fotonachweis: Gerd Kaluzinski (Jahrmarkt Lindau Bodensee)