Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

die Rheinländer kennen eine fünfte Jahreszeit, in Bayern gehen die Uhren anders, denn hier startet aktuell die fünfte Jahreszeit. Die Tipps zu den TimeSeries arbeiten aktuell zwar noch mit den herkömmlichen Kalendern, aber dies liesse sich mit einem zusätzlich erstellten Kalender „Oktoberfest“ auch anpassen. Informix startet in einen Herbst mit vielen Events. Das Bootcamp mit TimeSeries, das Genero Bootcamp und der IUG Workshop in Hamburg stehen in den kommenden Wochen an. Nutzen Sie die Gelegenheit und nehmen Sie teil !

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 (Teil 2): JSON Wire Listener Setup und Mongo Shell

Die NoSQL-Funktionalität von Informix wird durch den sogenannte JSON Wire Listener ermöglicht. Dieser hört auf einen konfigurierten IP-Port der Informix-Maschine, nimmt NoSQL-Anfragen entgegen und reicht sie über JDBC an den Informix Server weiter. Ergebnisse vom Informix Server werden auf dem selben Weg zurück kommuniziert. Als Client für NoSQL benutzen wir die Mongo Shell, mit der wir NoSQL-Anfragen stellen können, ähnlich wie 'dbaccess' für SQL-Befehle benutzt wird.

Die folgend beschriebene Vorgehensweise zur Installation, Konfiguration und Benutzung der jeweiligen Komponenten wurde auf Redhat Linux unter Benutzung von Informix Version 12.10.FC4 erprobt. Für andere Plattformen können geringfügige Anpassungen notwendig sein.

JSON Wire Listener Setup

Um die NoSQL-Funktionalität von Informix einzurichten, starten wir den JSON Wire Listener. Das Setup dafür besteht aus wenigen Schritten:

- **Umgebung** zum Starten des JSON Wire Listener:

  Zusätzlich zu den üblichen Umgebungseinstellungen für den Informix Server setzen wir folgende Umgebungsvariablen für den JSON Wire Listener:

  - **`CLASSPATH=${INFORMIXDIR}/lib/ifxjdbc.jar`**

    Da die Kommunikation mit dem Informix Server mittels JDBC stattfindet, muss der JDBC-Treiber entsprechend auffindbar sein.
    D.h. **`${INFORMIXDIR}/lib/ifxjdbc.jar`** muss im CLASSPATH enthalten sein. CLASSPATH kann natürlich weitere Pfade enthalten.

  - **`IBM_JAVA_OPTIONS=-Dcom.ibm.tools.attach.enable=no`**

    Dieser Parameter verhindert, dass irgendeine (andere) Applikation die Java Virtual Machine des JSON Wire Listeners 'kapern' kann. (Ohne diesen Parameter kann sich eine Applikation zur JVM verbinden und dann ihren eigenen Programmcode in die JVM zur Ausführung laden. So etwas ist beim JSON Wire Listener weder notwendig noch erwünscht.)

➢ **Konfigurationsdatei**:

Einige Parameter für den JSON Wire Listener geben wir in einer Konfigurationsdatei an. Ein allgemeines Beispiel für solch eine Konfigurationsdatei ist **`${INFORMIXDIR}/etc/jsonListener-example.properties`**.

Für unsere Zwecke beschränken wir uns auf ein Minimum und tragen nur die folgenden Parameter in eine Datei namens **`${INFORMIXDIR}/etc/jsonListener.properties`** ein:

```
listener.type=mongo
listener.port=26351
security.sql.passthrough=true
url=jdbc:informix-
sqli://localhost:24731/sysmaster:INFORMIXSERVER=nosql;USER=in
formix;PASSWORD=informix
```

Die Bedeutung der einzelnen Parameter:

- listener.type=mongo:
  Um unsere NoSQL-Anfragen mit der Mongo Shell abzusetzen, benutzen wir einen JSON Wire Listener vom Typ 'mongo'.
- listener.port=26351:
  Der IP-Port, auf den der JSON Wire Listener hört, um NoSQL-Anfragen entgegenzunehmen.
- security.sql.passthrough=true:
  Ermöglicht auch die Verwendung von SQL-Befehlen.
- url=jdbc:informix-sqli://**localhost**:**24731**/sysmaster:INFORMIXSERVER=**nosql**;USER=informix;PASSWORD=informix:
  Normaler JDBC-Connection String (in einer zusammenhängenden Zeile) für die Verbindung zum Informix Server mittels JDBC. Der entsprechende Eintrag in der sqlhosts-Datei des Informix Server sieht so aus:

  nosql onsoctcp **localhost** **24731**

- **Startscript**:

Da der Befehl zum Starten des JSON Wire Listener etwas länglich ist, erstellen wir ein kleines Startscript namens `json_listener_start.sh`. Wir verwenden das Java im Verzeichnis **`${INFORMIXDIR}/extend/krakatoa/jre/bin`**, welches normalerweise bei Informix mit installiert wird. Neben dem Pfad für das Java-Archive des JSON Wire Listeners selbst geben wir auch den Pfad der Konfigurationsdatei, und den Pfad der Log-Datei an. Mit dem abschliessenden '&' lassen wir den Listener als eine Art Dämon im Hintergrund laufen. Schliesslich geben wir dem JSON Wire Listener zwei Sekunden Zeit, sich zum Informix Server zu verbinden und mit dem Hören auf den IP-Port zu beginnen.

Das komplette Script:

```
#!/bin/sh

  ${INFORMIXDIR}/extend/krakatoa/jre/bin/java \
    -jar ${INFORMIXDIR}/bin/jsonListener.jar \
    -config ${INFORMIXDIR}/etc/jsonListener.properties \
    -logfile ${INFORMIXDIR}/tmp/jsonListener.log \
    -start &

  sleep 2

  exit
```

Das Startscript führen wir als Benutzer "informix" aus und erhalten als Bestätigung des korrekten Starts folgende Ausgabe:

```
$ json_listener_start.sh
  starting mongo listener on port 26351
$
```

In der Log-Datei finden wir daraufhin folgenden Eintrag:

```
$ cat ${INFORMIXDIR}/tmp/jsonListener.log
  2014-09-01 16:12:04 [JsonListener-1] INFO
  com.ibm.nosql.informix.server.mongo.MongoListener -
  JSON server listening on port: 26351, 0.0.0.0/0.0.0.0
$
```

- **Stopscript**:

Um den JSON Wire Listener ebenso einfach stoppen zu können, erstellen wir als Gegenstück zum Startscript auch ein kleines Stopscript namens **`json_listener_stop.sh`**. Die Zusammensetzung ist nun selbsterklärend:

```
#!/bin/sh

  java -jar ${INFORMIXDIR}/bin/jsonListener.jar \
    -config ${INFORMIXDIR}/etc/jsonListener.properties \
    -stop

exit
```

Führen wir das Stopscript als Benutzer "informix" aus, so erhalten wir folgende Antwort:

```
$ json_listener_stop.sh
 stop requested for localhost:26351
$
```

In der Log-Datei finden wir anschliessend folgende neue Nachrichten:

```
2014-09-01 16:26:18 [JsonListener-1] INFO
  com.ibm.nosql.informix.server.mongo.MongoListener -
  connection accepted from /127.0.0.1:39109
2014-09-01 16:26:18 [JsonListener-1-thread-1] INFO
  com.ibm.nosql.informix.server.mongo.MongoListenerMessageHandler -
  connection accepted from: 127.0.0.1:39109 ,
  sessionNumer: 1 , listenerNumber: 22
2014-09-01 16:26:18 [JsonListener-1-thread-1] INFO Session-1 -
  Shutting down Informix JSON Listener 1
2014-09-01 16:26:18 [JsonListener-1-thread-1] INFO
  com.ibm.nosql.informix.server.mongo.MongoListener -
  NOSQL_WIRE_LISTENER_SHUTDOWN_REQUEST
2014-09-01 16:26:18 [JsonListener-1-thread-1] INFO
  com.ibm.nosql.informix.server.mongo.MongoListenerMessageHandler -
  client /127.0.0.1 using thread JsonListener-1-thread-1 disconnected
2014-09-01 16:26:19 [Thread-6] INFO
  com.ibm.nosql.informix.server.mongo.MongoListener -
  Exiting Listener
```

(Nach Ausführung des Stopscripts ist der JSON Wire Listener für die weiteren Schritte natürlich wieder zu starten.)

Mongo Shell

Um NoSQL-Anfragen stellen zu können, brauchen wir einen entsprechenden Datenbank-Client, der mit Mongo-Befehlen und Daten im JSON-Format umgehen kann. Ähnlich wie das 'dbaccess'-utility für Informix und SQL-Befehle gibt es für die MongoDB Datenbank auch eine entsprechende Mongo Shell. Diese ist für unsere Zwecke gut geeignet, auch wenn wir nicht die Absicht haben, die MongoDB Datenbank selbst zu benutzen.

Das Paket der MongoDB, in dem auch die Mongo Shell mit enthalten ist, kann direkt von MongoDB für die gewünschte Plattform geladen werden:

http://www.mongodb.org/downloads.

Das Paket mit einem Dateinamen wie zum Beispiel

`mongodb-linux-x86_64-2.4.9.tgz`

packen wir in einem geeigneten Unterverzeichnis aus (z.B. durch Ausführung von

`tar xzf mongodb-linux-x86_64-2.4.9.tgz`

).

In der Verzeichnisstruktur, die dabei aufgebaut wird, befindet sich im Unterverzeichnis `bin` die Mongo Shell als Binärdatei `mongo`.Um die Mongo Shell bequem aufrufen zu können, fügen wir den absoluten Pfad des genannten `bin`-Unterverzeichnisses in die `PATH` Umgebungsvariable ein. Eventuell ist danach ein neues Einloggen erforderlich, um das neue Binary `mongo` auch von beliebiger Stelle im Dateisystem auffindbar zu haben. Als Test kann ein einfacher Aufruf

`mongo -version`

dienen, der zum Beispiel folgende Ausgabe produziert:

```
$ mongo -version
MongoDB shell version: 2.4.9
$
```

Durch Angabe von host und IP-Port des JSON Wire Listeners als Kommandozeilenparameter für die Mongo Shell kann die Verbindung zum Informix Server hergestellt werden:

```
$ mongo localhost:26351
MongoDB shell version: 2.4.9
connecting to: localhost:26351/test
Thu Jul 10 18:43:22.518 Error: couldn't connect to
  server localhost:26351 at src/mongo/shell/mongo.js:147
exception: connect failed
$
```

Die obige Ausgabe der Mongo Shell deutet darauf hin, dass wohl der JSON Wire Listener nicht wieder gestartet wurde, nachdem wir ihn zuvor gestoppt haben. In diesem Fall führen wir einfach wieder als Benutzer "informix" das Startscript **json_listener_start.sh** aus. Danach sollte ein erneuter Versuch mit der Mongo Shell besser funktionieren, und die Ausgabe der Mongo Shell einen '>'-Prompt geben, bei dem wir nun NoSQL-Anfragen absetzen können:

```
$ mongo localhost:26351
MongoDB shell version: 2.4.9
connecting to: localhost:26351/test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
>
```

Der Befehl `show dbs` zeigt uns die vorhandenen Benutzerdatenbanken an (jedoch nicht Systemdatenbanken wie "sysmaster", "sysutils", etc.):

```
> show dbs
stores_demo     (empty)
>
```

In der Informix Instanz existiert offensichtlich eine Datenbank namens "stores_demo", jedoch scheint diese leer zu sein. Dies bedeutet jedoch lediglich, dass es in der Datenbank "stores_demo" noch keine NoSQL-Collection gibt, denn normale SQL-Tabellen werden hier nicht berücksichtigt. (Wird die Beispiel-Datenbank "stores_demo" nicht angezeigt, so kann sie jederzeit mit dem Programm `dbaccessdemo` erzeugt werden. Wir werden sie im Verlauf der Artikelserie immer wieder verwenden. Die abschliessende Frage des `dbaccessdemo`-Programms, ob Beispiel-Dateien in das lokale Verzeichnis kopiert werden sollen, können wir mit "no" beantworten, da wir diese Dateien nicht benötigen.)

Auch nach dem expliziten Verbinden zur Datenbank "stores_demo" sind noch keine *`Collections`* zu sehen:

```
> use stores_demo
switched to db stores_demo
> show collections
>
```

Obwohl wir sicher sind, dass die Datenbank "stores_demo" tatsächlich schon einige SQL-Tabellen enthält, ist obige Ausgabe nicht verwunderlich, da dies ja SQL-Tabellen sind und nicht NoSQL-Collections.

Schliesslich verlassen wir die Mongo Shell durch Eingabe von CTRL-D.
Zu Beginn wurde versprochen, dass man mit Informix die Daten in normalen SQL-Tabellen zusammen mit NoSQL-Daten in Collections verarbeiten kann. Dies gilt nicht nur bei der Nutzung von SQL-Befehlen, sondern auch für NoSQL-Anfragen. Selbst wenn die SQL-Tabellen mit dem Mongo-Befehl **`show collections`** nicht angezeigt werden, so können wir nun mit dem Wissen, dass die SQL-Tabelle "classes" in der Datenbank "stores_demo" existieren sollte, versuchen, uns die Daten dieser Tabelle anzeigen zu lassen, mit der Mongo Shell und im JSON-Format.

Damit wir nicht nach jedem neuen Aufruf der Mongo Shell mit dem Kommando

```
use stores_demo
```

uns zur Datenbank "stores_demo" verbinden müssen, fügen wir beim Aufruf der Mongo Shell die Datenbank mittels vorangestelltem '/' dem Kommandozeilenparameter hinzu:

```
$ mongo localhost:26351/stores_demo
MongoDB shell version: 2.4.9
connecting to: localhost:26351/stores_demo
> db.classes.find()
{ "classid" : 1, "class" : 125, "subject" : "Chemistry" }
{ "classid" : 2, "class" : 250, "subject" : "Physics" }
{ "classid" : 3, "class" : 375, "subject" : "Mathematics" }
{ "classid" : 4, "class" : 500, "subject" : "Biology" }
>
```

In der nächsten Ausgabe des Newsletters werden wir sehen, wie eine NoSQL-Collection angelegt und mit Daten gefüllt wird, sowie Möglichkeiten, die Daten wieder anzuzeigen und zu manipulieren.

TechTipp: TimeSeries Container Limits

Die Daten des Typ TimeSeries werden in Containern abgelegt, die zuvor erstellt werden mussten. Bei der Érstellung der Container wird eine initiale Grösse (in kB) und eine Erweiterungsgrösse (in kB) angegeben, analog der Extentsize und Nextsize bei Tabellen. Beispiel mit Initial 42000 kB und Erweiterung um 13000 kB:

execute procedure TSContainerCreate
("Wetter_Day_Cont","datadbs","r_wetter_time",**42000**,**13000**);

Ein Container kann nicht unendlich wachsen. Das Limit liegt bei 16777216 Pages und die maximale Größe hängt damit von der Pagesize des DBSpaces ab, in dem der Container erstellt wurde:

- Pagesize 2-KB bedingt ein Limit von 32 GB je Container
- Pagesize 4-KB bedingt ein Limit von 64 GB je Container
- Pagesize 8-KB bedingt ein Limit von 128 GB je Container
- Pagesize 16-KB bedingt ein Limit von 256 GB je Container

Der aktuelle Füllungsstand kann mittels einer TimeSeriesFunktion ausgelesen werden:

```
execute function TSContainerUsage('Wetter_Day_Cont');
```

Ergebnis:

```
pages            slots                total
6364             1533669              21000
```

Eine weitere Funktion gibt den Füllungsgrad in Prozent aus:

```
execute function TSContainerPctUsed('Wetter_Day_Cont');
```

Ergebnis:

```
percent
30.305
```

Die Anzahl der Elemente in einem Container gibt eine andere Funktion aus:

```
execute function TSContainerNElems('Wetter_Day_Cont');
```

Ergebnis:

```
elements
1533669
```

Was all diese Funktionen nicht berücksichtigen ist, dass der Container sich bis zum Limit von 16777216 Pages erweitern kann. Wir haben daher eine Funktion für Sie erstellt, die den Füllungsgrad in Prozent gegenüber der maximalen Füllung des Containers ausgibt, und dies über alle Container der Datenbank hinweg.

Die Namen der Container finden wir in der Tabelle „**TsContainerTable**“.

Die Prozedur könnte damit wie folgt aussehen:

```
create procedure if not exists show_container_info ()
returning
     char(18) as container,
     bigint as pages,
     bigint as total_pg,
     dec(5,2) as prc_free

define cont_name        char(18);
define cont_pages       bigint;
define cont_slots       bigint;
define cont_total       bigint;
define cont_prc         dec(5,2);
define x_limit          int;

let x_limit = 16777216;  --maximale Anzahl Pages je Container

foreach select name
     into cont_name
     from TsContainerTable

     execute function TSContainerUsage(cont_name)
          into cont_pages,cont_slots,cont_total;

     let cont_prc=round((x_limit-cont_pages) / x_limit*100,2);
     return cont_name, cont_pages, cont_total, cont_prc
     with resume;

end foreach
end procedure
```

Das Ergebnis hat dann folgendes Format:

```
container           pages          total_pg       prc_free
Wetter_Day_Cont     6364           21000          99.96
Wetter_Day_Cont2    9092           21000          99.95
```

Der Wert „prc_free“ bezieht sich nicht wie bei der implizit vorhandenen Funktion auf die aktuelle Anzahl der „total_pages“, sondern auf die maximale Anzahl an Pages, die der Container erreichen könnte.

TechTipp: TimeSeries Daten verschieben

Im vorigen TechTipp haben wir beschrieben, wie man den Füllungsgrad von Containern der TimeSeries ermittelt. Stellt man nun fest, dass ein Container recht voll ist, aber in anderen Containern noch genügend Platz zur Verfügung steht, so stellt sich die Aufgabe, Daten von einem Container in einen anderen Container zu verschieben. Dies kann mittels eines Update auf die TimeSeries-Spalte der Tabelle erfolgen.

Im folgenden Beispiel wird der Inhalt der TimeSeries „wetterdaten“ der Tabelle „tageswetter“ für das KFZ-Kennzeichen „LI“ vom aktuellen Container („Wetter_Day_Cont“) in den Container „Wetter_Day_Cont2“ verschoben.

```
update tageswetter
     set wetterdaten =
         SetContainerName(wetterdaten,'Wetter_Day_Cont2')
 where kfz = 'LI';
```

Der veränderte Füllstand der Container kann mit der Auswertung aus dem vorangegangenen TechTipp evaluiert werden.

Wer nun Lust bekommen hat, selbst diese Funktionen der TimeSeries zu testen, der findet eine Anleitung zu den hier im Test genutzten TimeSeries in den Ausgaben März 2011 bis Mai 2011 des Informix Newsletters. Im Internet findet zudem Jeder sicherlich eine Wetterstatistik zu seinem eigenen Wohnort, so dass hier mit realen Daten geübt werden kann.
Die Liste der Archive, in denen die Newsletter zu finden sind, ist im Bereich „Anmeldung“ zu sehen.

TechTipp: Stored Procedure „blanks in String finden“

Immer wieder stellt sich die Aufgabe, einen String in einzelne Worte zu zerlegen. Aus diesem Grund haben wir eine Prozedur erstellt, welche die Position der Leerzeichen in einem String zurückgibt.

Das zweite Argument steuert, ob man alle Positionen der Leerzeichen sehen will (Argument 0), oder nur die Position eines bestimmtes Leerzeichens ausgegeben werden soll.

Werden keine Leerzeichen im Eingabestring gefunden, so wird -1 zurückgegeben.

Die Prozedur könnte damit folgendermassen aussehen:

```
create procedure get_blank_pos(in_str varchar(255), x_in int)
returning int as position
-- Parameter: in_str        - Der Eingabestring
-- Parameter: x_in          - 0: Alle Positionen von "blank" ausgeben
--                          - n: Die Positionen des n-ten "blank" ausgeben
define pos,cnt int;
define len_str int;
define x_search char(1);
let cnt = 0;
let pos = 1;
let len_str = length(in_str);
while (pos < (len_str))
      let x_search = substr(in_str,pos,1);
      if x_search = ' '
            then let cnt = cnt+1;
            if x_in = 0 then return pos with resume;
            else if cnt = x_in then return pos;
                  end if
            end if
      end if
      let pos = pos+1;
end while
if cnt = -1 then return -1; end if;
end procedure ;
```

Ein einfacher Test zeigt die Ergebnisse:

```
execute procedure get_blank_pos ("Informix Newsletter September 2014",0);

   position
          9
         20
         30
3 row(s) retrieved.
```

TechTipp: Optionen des ONSTAT (onstat -g dbc)

Die Informix Instanz besitzt einen Scheduler, der analog zum Cron auf Unix/Linux Aufgaben abarbeitet und Ergebnisse der ausgeführten Jobs meldet. Die Aufgaben sind entweder Tasks oder Sensoren, die in der Datenbank „sysadmin“ hinterlegt sind.
Das Flag „enabled“ bei den Tasks und Sensoren steuert, ob die Aufgaben durch den Scheduler ausgeführt werden.
Informationen über den Informix Scheduler können mittels „onstat -g dbc“ angezeigt werden. Dabei werden die aktuell laufenden Tasks, die nächsten anstehenden Tasks, die Warteschlange vor den Schedulern, sowie die Anzahl der aufgetretenen Fehler in einer Übersicht ausgegeben.

Anbei eine Beispielausgabe:

```
onstat -g dbc
IBM Informix Dynamic Server Version 12.10.FC4 -- …

Worker Thread(1)    4a464f80
======================================
Task:                4a4a1540
Task Name:           mon_iohistory (26-2681)
Task Type:           SENSOR

WORKER PROFILE
    Total Jobs Executed            36
    Sensors Executed               19
    Tasks Executed                 17
    Purge Requests                 36
    Rows Purged                    56
    Errors                          0
```

Zu sehen ist, dass der erste Worker Thread aktuell den Task „mon_iohistory“ bearbeitet. Dieser gehört zu den Sensoren, was am Typ erkennbar ist.
Der Worker Thread hat seit dem Start 36 Jobs bearbeitet, davon waren 19 Sensoren und 17 Tasks. Es wurden 36 Purge Requests ausgeführt, bei denen zumeist veraltete Ergebnisse der letzten Läufe gelöscht werden. Dabei wurden 56 Datensätze gelöscht und es traten bei der Verarbeitung keine Fehler auf.

Danach folgen die weiteren WorkerThreads.

```
Worker Thread(2)    4a4a14c0
====================================
Task:                4a4a19d8
Task Name:           mon_page_usage (44-307)
Task Type:           SENSOR

WORKER PROFILE
    Total Jobs Executed            43
    Sensors Executed               23
    Tasks Executed                 20
    Purge Requests                 43
    Rows Purged                  1729
    Errors                          0
```

Nach der Liste der Worker Threads sind ist die Statistik des Schedulers zu sehen, in der die ID des nächsten Tasks (hier 8) zu finden ist, die Wartezeit bis zur Ausführung (in Sekunden) des nächsten Tasks, die Anzahl der Einträge in der Warteschlange, sowie weitere Informationen über die Aktivitäten des Schedulers:

```
Scheduler Thread         4a464f00
====================================
Next Task                8
Next Task Waittime       1107
General Queue size       9
     full                0
     empty               9
Private Queue size       1
     full                0
     empty               1
PRIVATE QUEUE TASKS
    Total Jobs Executed            13
    Sensors Executed                0
    Tasks Executed                 13
    Purge Requests                 13
    Rows Purged                     0
    Errors                          0
```

Der Scheduler wird üblicherweise mit dem Start der Instanz aktiviert. Es besteht die Möglichkeit den Scheduler zu stoppen, indem als Benutzer „informix“ in der Datenbank „sysadmin“ folgender Befehl eingegeben wird:

execute function task ("scheduler stop");

Der Scheduler kann auf die selbe Art wieder gestartet werden mit dem Befehl:

execute function task ("scheduler start");

In Ausnahmefällen kann der automatische Start des Schedulers verhindert werden, indem eine Datei „stop“ im Verzeichnis $INFORMIXDIR/etc/sysadmin erstellt wird.
Dies kann z.B. erfolgen mittels:

touch $INFORMIXDIR/etc/sysadmin/stop

Termin: Informix Bootcamp mit TimeSeries

Auch in diesem Herbst findet wieder das beliebte Informix Bootcamp statt. Neben den Neuerungen des Informix Servers werden auch neue Features der TimeSeries vorgestellt. Veranstaltungsort ist Düsseldorf. Das Bootcamp findet von Dienstag, den 11.November bis Freitag, den 14. November statt. Nutzen Sie die Gelegenheit und melden Sie sich rechtzeitig an, bevor die Plätze vergeben sind. Der Link zur Anmeldung steht unter:
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Information%20Management/page/Informix%20Bootcamp

Termin: Informix Genero Bootcamp

Zum Thema Informix Genero laufen derzeit die Planungen für ein Bootcamp im November. Reservieren Sie sich bereits heute den Termin vom 04. bis 06. November. Als Veranstaltungsort ist Ehningen vorgesehen. Aktuell gibt es noch keinen Anmeldelink. Bei Interesse können Sie sich gerne an uns (ifmxnews@de.ibm.com) wenden, damit wir einen Platz für Sie unverbindlich reservieren.

Termin: 65. IUG-Workshop in Hamburg

Am Mittwoch, den 29. Oktober 2014 findet in Hamburg der 65. IUG Workshop statt. Das Thema ist diesmal:

Business Intelligence, Entwicklungs-Methode und Internet-Technologie

Wie gewohnt findet am Vorabend der beliebte IUG Stammtisch statt, an dem alte Verbindungen gepflegt, und neue Kontakte geknüpft werden können. Die Agenda wird in den nächsten Tagen zur Verfügung stehen.

Weitere Informationen und den Link zur Anmeldung finden Sie unter:
http://www.iug.de/index.php?option=com_content&task=view&id=296&Itemid=383

WebTipp: Data Transmission for Dynamic Databases

Passend zur Serie NoSQL ist ein Artikel im IBM Data Magazine erschienen, der die Nutzung von JSON und BSON im Zusammenhang mit relationalen Datenbanken beschreibt.
Den Artikel finden Sie unter:

http://ibmdatamag.com/2014/07/data-transmission-for-dynamic-databases/?doing_wp_cron=1406290480.7639648914337158203125

WebTipp: Hochverfügbarkeit – Ein Vergleich der Systeme

Hochverfügbarkeit wird immer mehr zum entscheidenden Argument für die Wahl einer Datenbanklösung, da viele Webanwendungen rund um die Uhr zur Verfügung stehen müssen. Ganz ohne Kosten für die Administration ist die Einrichtung dieses Service Levels nie, aber es gibt erhebliche Unterschiede bei Kosten und Aufwand, je nach gewählter Lösung. Der folgende Link zeigt einen Vergleich, der verdeutlicht, dass INFORMIX mit seiner langjährigen Erfahrung mit Replikaitonslösungen eine sehr gute Wahl ist:
http://public.dhe.ibm.com/common/ssi/ecm/en/iml14402usen/IML14402USEN.PDF

Versionsinfo: 12.10.xC4W1 ist verfügbar

Seit einigen Tagen ist die Version 12.10.xC4W1 für alle unterstützten Plattformen und Editionen verfügbar. Da es in jeder Version eine Reihe an Verbesserungen gibt, sollte immer eine der neueren Versionen eingesetzt werden.

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

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 (Martin Fuerderer – Wiesn München)