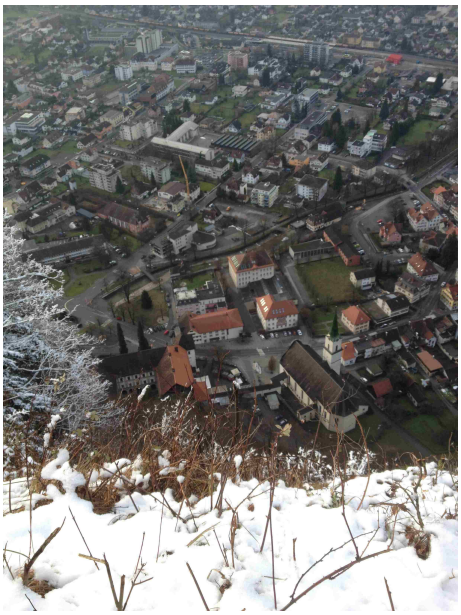
Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

in dieser Ausgabe des INFORMIX Newsletters konzentrieren wir uns auf zwei Themen:
Die Ausgaben des SQEXPLAIN und wie der Client an diese Informationen kommt, sowie die Erfassung und Ablage von Sensordaten in INFORMIX an einem Beispiel, das so sogar auf einem RaspberryPi nachgestellt werden kann.
Mit unserer Feststellung in der Ausgabe Januar, dass der Frühling schon zu spüren ist, haben wir den Winter anscheinend nochmals herausgefordert. Während im Tal Alles grün ist, sind direkt darüber die Anhöhen wieder dick mit Schnee bedeckt. Einfach ideal zum Wandern.



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

Ihr TechTeam

TechTipp: SQEXPLAIN – Performance analysieren

Der „sqexplain“ bietet die Möglichkeit, die Entscheidungen des Optimizers zu protokollieren. Dabei erhält man Informationen über die Reihenfolge, in der die Daten aus den Tabellen gelesen werden, den Weg (Index, sequentiell, …), die Einschränkung auf Fragmente oder Partitionen und die geschätzten Kosten des Zugriffs.
Zudem wird je Tabelle die geschätzte und die tatsächliche Anzahl der ermittelten Datensätze protokolliert.
Es gibt unterschiedliche Möglichkeiten den „sqexplain“ zu aktivieren. Läuft eine Session bereits, so kann die Aktivierung für alle nachfolgenden Abfragen durch den User „informix“ mittels

onmode -Y <session-id> 1 <sqxplain-file>

erfolgen, wobei die Angabe der Datei, in der die Ausgabe landen soll, optional ist.
Hat man die Möglichkeit das zu untersuchende SQL-Statement zu erweitern, so kann der Sqexplain auch mit dem Befehl

set explain on

vor Ausführeung des SQL-Statements aktiviert werden.

Die Ausgabe des Sqexplain erfolgt bei Abfragen, die lokal am Datenbankserver angesetzt werden, in die Datei „sqexplain.out“ im Aufrufverzeichnis, wenn keine andere Datei angegeben wurde. Bei Remote-Abfragen wird im $HOME des Users unter „sqexplain.<session_id>“ die Ausgabedatei erstellt.

Unter Windows ist die Ausgabe unter „%INFORMIXDIR%\sqexpln“ zu finden.
Mit dem Befehl

set explain file to “<Ausgabedatei>“

besteht die Möglichkeit, die Ausgabe des „sqexplain“ in eine frei wählbare Datei umzuleiten.
Die Ausgabedatei kann dabei direkt mit oder ohne absolutem Pfad in Hochkommata angegeben werden. Zudem besteht die Möglichkeit, den Pfad als Variable zu übergeben, die als Ergebnis einer Abfrage z.B. aus einer Parametertabelle ermittelt wird. So könnte auch die aktuelle Zeit in den Namen einfliessen.

Ist in der Datei $ONCONFIG der Parameter EXPLAIN_STAT nicht gesetzt, so kann mittels

set explain statistics

die Ausgabe um die Statistik der Anzahl der Datensätze und der gemessenen Zeiten für die Teilschritte erweitert werden (ist EXPLAIN_STAT gesetzt, so ist dies der Default).
Sollen nur die Entscheidungen des Optimizers ausgegeben werden, ohne dass die Query ausgeführt wird, so kann dies mittels

set explain on avoid_execute

erfolgen.

Beispielausgabe des sqexplain:

```
QUERY: (OPTIMIZATION TIMESTAMP: 02-16-2015 13:42:23)
------
select f.plz, p.fma, p.nam,
      case when n.prnnam is null then 'NO'
           else 'JA'
      end as bekommt_newsletter
from firma f, person p, outer informix_nl n
where f.fma = p.fma
and f.fma = n.fma
and p.nam = n.prnnam
and f.plz like "8%"
order by 1,2,3

Estimated Cost: 2716
Estimated # of Rows Returned: 808
Temporary Files Required For: Order By

  1) kalu.f: INDEX PATH
    (1) Index Name: informix.firma_plz
        Index Keys: plz   (Serial, fragments: ALL)
        Lower Index Filter: kalu.f.plz LIKE '8%'
      INDEX_NAME = firma_plz

  2) kalu.p: INDEX PATH
    (1) Index Name: informix. 108_71
        Index Keys: fma   (Serial, fragments: ALL)
        Lower Index Filter: kalu.f.fma = kalu.p.fma
      INDEX_NAME =  108_71

NESTED LOOP JOIN
  3) kalu.n: INDEX PATH
        Filters: kalu.p.nam = kalu.n.prnnam
    (1) Index Name: informix. 119_72
        Index Keys: fma   (Serial, fragments: ALL)
        Lower Index Filter: kalu.f.fma = kalu.n.fma
      INDEX_NAME =  119_72

NESTED LOOP JOIN

DB_LOCALE         =    en_US.57372
SESSION COLLATION =    de_DE.57372
```

Der Teil bis hier enthält die Entscheidungen und Schätzungen des Optimizers. Diese hängen sehr stark von der Qualität der Statistiken und den zur Verfügung stehenden Indexen ab. Die geschätzten Kosten sind nicht direkt in Zeit umrechenbar, jedoch entsprechen höhere Kosten fast durchwegs einer höheren Ausführungszeit. Üblicherweise fängt das Tuning bei den Statements mit den höchsten Kosten an. Wenn es sequentielle Zugriffe auf grosse Tabellen gibt, dann ist auch gleich ein Ansatzpunkt gefunden, wie schnell für Abhilfe gesorgt werden kann.

Der zweite Teil der Ausgabe enthält die Statistiken über die Ausführung und stellt dabei die geschätzten und tatsächlichen Anzahlen der verarbeiteten Datensätze gegenüber.

```
Query statistics:
-----------------
  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                f
  t2                p
  t3                n


  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     120         118       120        00:00.00   62

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t2     1464        5350      1464       00:00.00   6

  type     rows_prod  est_rows  time       est_cost
  ------------------------------------------------
  nljoin   1464       809       00:00.00   1055

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t3     222         1288      34047      00:00.03   1



  type     rows_prod  est_rows  time       est_cost
  ------------------------------------------------
  nljoin   1477       809       00:00.03   2470

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ------------------------------------------------------------
  sort     1477       809       1477       00:00.04   247
```

In diesem Abschnitt ist zu sehen, wie lange der Zugriff auf die Daten gedauert hat und ob die geschätzte Anzahl an Datensätzen (est_rows) mit der Anzahl der tatsächlich gelesenen Datensätze (rows_prod) übereinstimmt. Zudem finden sich hier Informationen, wie die gelesenen Daten miteinander gejoint wurden.
Je Teilschritt ist zu sehen, wie viel Zeit hierfür benötigt wurde, so dass für ein Tuning hierdurch die Tabellen identifiziert werden können, bei denen ein selektiver Index für diese Abfragen noch fehlt.

TechTipp: Clientzugriff auf die Ausgabe des SQEXPLAIN

Die im vorhergehenden Artikel beschriebene Funktion „sqexplain“ schreibt ihre Ausgaben auf den Datenbankserver. Oft erhalten die Entwickler keinen direkten Zugang auf den Datenbankserver, sondern nur den Zugriff über die SQL-Schnittstelle der Datenbank. Daher stellt sich die Aufgabe, die Ausgabe des „sqexplain“ vom Server auf den Client zu transferieren. Die Option „frag' mal den DBA, ob er uns die Dateien kopiert“ zählt dabei nicht als Lösung, da diese Personen anspruchsvollere Tätigkeiten im Programm haben. Der einfachste Option, die Ausgabedatei des „sqexplain“ zum Entwickler zu bringen ist hier sicherlich die mittels SQL. Die Funktion „filetoclob()“ liest Dateien in eine Tabelle ein, die Funktion „lotofile()“ kann binäre Daten aus einer Tabelle in eine Datei schreiben.

Das folgende Beispiel zeigt, wie eine Übertragung der Ausgabedatei erfolgen könnte:

Zuerst setzen wir die Ausgabedatei des „sqexplain“ auf einen festen Dateinamen und starten den Explain:

```
set explain file to "/tmp/kalu.explain";
set explain statistics;
```

Nun führen wie die SQL-Abfragen durch, die wir protokollieren wollen. Die Entscheidungen und Statistiken des Optimizers werden in der angegebenen Datei gespeichert.

Nun müssen wir noch eine temporäre Tabelle als Hilfstabelle erstellen, in der wir die Datei zwischenzeitlich aufnehmen können:

```
drop table if exists kalu_sqx;
create temp table kalu_sqx (
     ich     char(18),
     txt     clob
);
```

Anschliessend laden wir die Ausgabe des „sqexplain“ in die Tabelle mittels der Funktion „filetoclob()“:

```
insert into kalu_sqx values
("kalu_test",filetoclob("/tmp/kalu.explain","server"));
```

Die Informationen des „sqexplain“ können nun bereits direkt aus der Tabelle gelesen werden:

```
select * from kalu_sqx ;
```

Um die Ausgabe dauerhaft abzulegen, kann der Inhalt der Tabelle als Datei auf dem Client gespeichert werden:

```
select lotofile(txt,"/tmp/kalu.sqx","client") from kalu_sqx
where ich = "kalu_test";
```

Wichtig ist das Argument „**client**“ beim Aufruf des „lotofile()“. Dieses Argument legt fest, ob die zu speichernde Datei auf dem Server oder dem Client abgelegt werden soll.

Wird der Name wie im Beispiel nicht mit einem Ausrufezeichen abgeschlossen, so wird dem Dateinamen eine eindeutige Identifikationsnummer hinzugefügt. Im Beispiel war dies:

```
/tmp/kalu.sqx.0000000052518452
```

Mit Ausrufezeichen am Ende des Dateinamens wird der Inhalt der Tabelle exakt unter dem angegebenen Dateinamen abgelegt (hier also unter /tmp/kalu.sqx).

In einem Befehl sieht das dann so aus:

```
select lotofile(filetoclob("/tmp/kalu.explain", "server"),
                                "/tmp/kalu.sqx!", "client")
from systables where tabid = 1;
```

Nun bleibt nur noch das Problem das übertragenen Sqexplain am Server zu löschen, um für die nächste Analyse eine neue Datei zu erstellen. Hierzu schaffen wir uns eine kleine Prozedur, die das bewerkstelligt:

```
drop procedure if exists drop_file(varchar(255));
create procedure drop_file(x_in varchar(255))
define x_txt varchar(255);
--set debug file to "drop_file.debug";
--trace on;
let x_txt="rm "||trim(x_in);
system (x_txt);
end procedure;
```

und schon können wir auch das alte Explain File löschen:

```
execute procedure drop_file("/tmp/kalu.explain");
```

Als Alternative zur temporären Tabelle kam der Gedanke auf, die Ausgabe des „sqexplain“ als externe Tabelle anzusprechen, die dann vom Client abgefragt werden könnte. Da jedoch die Syntax der externen Tabellen einen Delimiter verlangt, hätte hier zuerst die Datei in jeder Zeile mit dem Delimiter versehen werden müssen, um dann anschliessend vom Client darauf zugreifen zu können.

Sollten Sie eine andere, bessere Lösung gefunden haben, dann schreiben Sie uns diese !

Wir wollen Ihnen aber unsere Versuche mit externen Tabellen nicht vorenthalten:

```
create procedure set_delimit(
     txt varchar(128), txt2 carchar(128)
)
define cmd varchar(128);
let cmd = "cat "||trim(txt)||"|sed 's/$/|/' >"||trim(txt2);
system cmd;
end procedure;
```

Die Prozedur fügt jeder Zeile aus „txt“ einen Delimiter „|“ an und speichert die Ausgabe unter „txt2“. Nun kann eine externe Tabelle erstellt werden, deren Datenfile dieses „txt2“ ist (in unserem Beispiel haben wir hierfür „/tmp/sqx.tmp“ eingesetzt).

```
create external table kalu_sqx (
     txt  char(80)
) using (
     DATAFILES ( "DISK:/tmp/sqx.tmp" )
);
select * from kalu_sqx
```

Viele Information des Sqexplain sind auch in der Tabelle syssqexplain der Datenbank sysmaster zu finden. Allerdings ist der Zugriff auf diese Daten auf den User „informix“ beschränkt und daher in den meisten Fällen auch keine Option um auf den Queryplan zuzugreifen.

TechTipp: Sensordaten mit Informix erfassen - Beispiel (Teil 1)

Im folgenden Artikel wird die Nutzung von Sensordaten in einer Informix Datenbank beschrieben. Das Beispiel kann mit kleinen Änderungen für jegliche Form von Sensordaten verwendet werden.
Im Teil 1 werden die Grundlagen für die Speicherung der Daten als TimeSeries erstellt.

Zuerst erstellen wir eine Datenbank mit Logging (hier im DBSpace „datadbs“):

```
create database sensor_db in datadbs with log
```

Um die Daten zu speichern, erstellen wir einen „Row Type“. Dieser könnte Werte vom Type Integer, Decimal, Character, … oder auch BSON enthalten. Im Beispiel nutzen wir für die Temperatur im ersten Schritt Decimal:

```
create row type sensor_t
(
     timestamp datetime year to fraction(5),
     value     decimal(15,2)
);
```

RowTypes, die Werte für eine TimeSeries speichern sollen, müssen eine Spalte „timestamp“ vom Typ „datetime year to fraction(5)“ besitzen.
Um die Daten der TimeSeries aufnehmen zu können, muss ein Container für den RowType „sensor_t“ angelegt werden. Im Beispiel trägt der Container den Namen „sensor_cont“ und liegt im „datadbs“. Der Container wird mit einer FirstExtentSize und NextExtentSize von 2 MegaByte angelegt:

```
execute procedure TSContainerCreate (
        'sensor_cont',
        'datadbs',
        'sensor_t',
        2048,
        2048);
```

Anschliessend können wir die Tabelle erstellen, die den RowType als TimeSeries beinhaltet:

```
create table sensor_ts (
        sensor_id char(40),
        sensor_type char(20),
        sensor_unit char(6),
        sensor_values timeseries(sensor_t),
        primary key (sensor_id)
) lock mode row;
```

Um die Tabelle auch mit üblichem SQL ansprechen zu können, erstellen wir zusätzlich eine Virtuelle Tabelle „sensor_data“ als „relationale Sicht“ auf die TimeSeries Tabelle:

```
execute procedure TSCreateVirtualTab (
          'sensor_data',
          'sensor_ts',
          'origin(2015-01-01 00:00:00.00000),
          calendar(ts_1min),container(sensor_cont),
          threshold(0),regular',0,'sensor_values'
);
```

Als Ursprung wird im Beispiel der 01.01.2015 angegeben. Ab diesem Zeitpunkt können Werte in der Tabelle aufgenommen werden. Als „Kalender“ wurde der Defaultkalender „ts_1min“ gewählt, der eine Granularität von einer Minute vorgibt. Weitere Parameter sind der Container, der die Daten aufnimmt, sowie die Information, dass es sich um eine reguläre Zeitreihe handelt (bei der es nur das Intervall des Kalenders gibt und keine willkürlichen Zeitmarken. Abschliessend wird die Zeitreihe angegeben, die in der Originaltabelle vorhanden ist.
Dieser Aufruf erzeugt eine virtuelle Tabelle mit folgendem Schema:

```
create table sensor_data (
     sensor_id char(40) not null ,
     sensor_type char(20),
     sensor_unit char(6),
     timestamp datetime year to fraction(5),
     value decimal(15,2)
);
```

Nun können zum Test die ersten Daten eingefügt werden:

```
insert into sensor_data values ("Sensor01", "Temp", "C",
     "2015-01-26 08:00"::datetime year to minute, 21.5);
insert into sensor_data values ("Sensor01", "Temp", "C",
     "2015-01-26 08:01"::datetime year to minute, 21.6);
insert into sensor_data values ("Sensor02", "Temp", "C",
     "2015-01-26 15:45"::datetime year to minute, 35.9);
insert into sensor_data values ("Sensor01", "Temp", "C",
     "2015-01-26 08:02"::datetime year to minute, 22.1);
insert into sensor_data values ("Sensor02", "Temp", "C",
     "2015-01-26 15:46"::datetime year to minute, 35.2);
insert into sensor_data values ("Sensor02", "Temp", "C",
     "2015-01-26 15:47"::datetime year to minute, 33.5);
insert into sensor_data values ("Sensor01", "Temp", "C",
     "2015-01-26 07:58"::datetime year to minute, 20.2);
...
```

Die eingefügten Datensätze erscheinen in der virtuellen Tabelle als zehn unterschiedliche Records. In der TimeSeries Tabelle hingegen sind es nur zwei Einträge (ein Eintrag je Sensor):

```
select count(*) from sensor_ts;
      (count(*))
               2

select count(*) from sensor_data;
      (count(*))
               10
```

TechTipp: Sensordaten mit Informix erfassen - Beispiel (Teil 2)

Der 2. Teil dieses Artikels behandelt die Abfrage der Daten mit TimeSeries Funktionen.

Ein typischer Fall ist die Abfrage der Summe der Werte in einem Zeitintervall.
In unserem Beispiel summieren wir die Werte je Stunde auf. Hierbei verwenden wir die Funktion „AggregateBy“ und den Kalender „ts_1hour“, der als Default bei der Installation bereits vorhanden ist:

```
SELECT AggregateBy(
     'SUM($value)',
     'ts_1hour',
     sensor_values,
     0,
     '2015-01-26 00:00'::datetime year to minute,
     '2015-01-31 23:59'::datetime year to minute)
FROM sensor_ts
WHERE sensor_id = "Sensor01";
```

Summierung nach Tag, Monat, Jahr ist ebenfalls als Default vorhanden. Andere Intervalle müssen zuerst als Kalender hinterlegt werden, bevor diese genutzt werden können.

Das Ergebnis der Abfrage ist:

```
(expression)  origin(2015-01-26 00:00:00.00000),
calendar(ts_1hour), container( sensor_cont), threshold(0),
regular, [NULL, NULL, NULL, NULL, NUL L, NULL, NULL, (40.90
), (65.20            ), NULL, NU LL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, N ULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, (53.20            )]
```

Zwischen den angegebenen Zeitmarken wird für jede Stunde die Summe der Werte ausgegeben. Sind für eine Stunde keine Werte vorhanden, so wird „NULL“ zurückgegeben.

Das Format der Ausgabe entspricht dem der TimeSeries und kann mittels „Transpose“ in die Form einer üblichen Tabelle umgewandelt werden:

```
SELECT sensor.timestamp::datetime year to hour as time,
     sensor.value
FROM TABLE (
     TRANSPOSE ((
          SELECT AggregateBy(
               'SUM($value)',
               'ts_1hour',
               sensor_values,
               0,
               '2015-01-26 00:00'::datetime year to minute,
               '2015-01-31 23:59'::datetime year to minute)
          FROM sensor_ts
          WHERE sensor_id = "Sensor01"
     ))::sensor_t
) AS TAB (sensor);
```

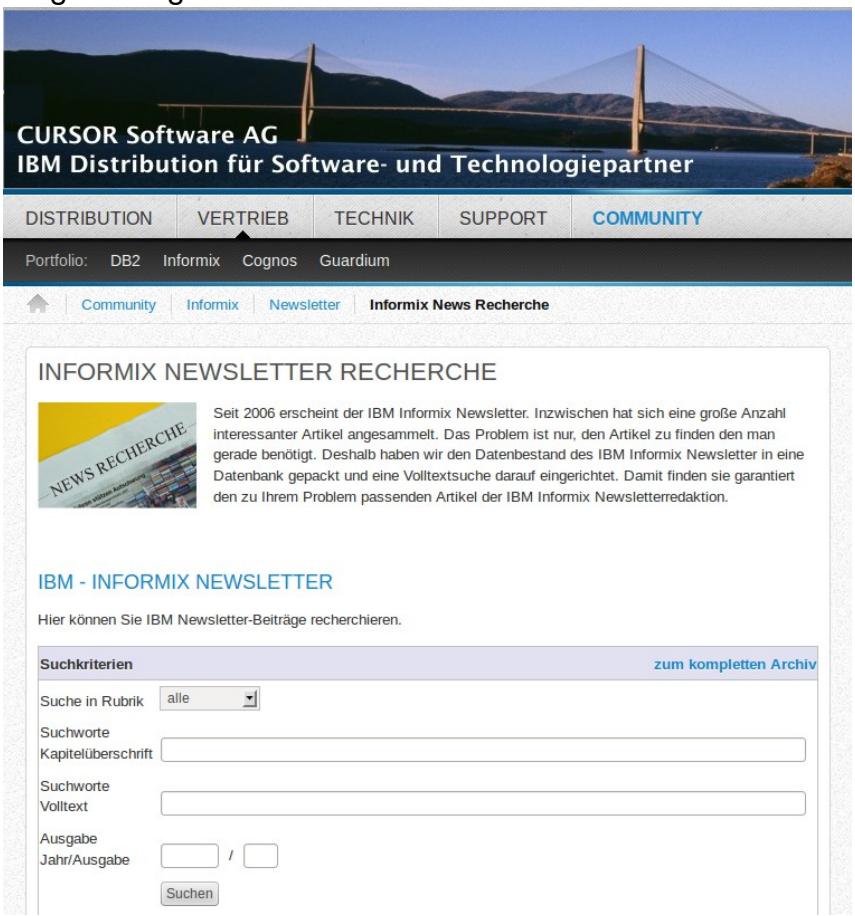
Mit dem Ergebnis:

```
time                    value
2015-01-26 07           40.90
2015-01-26 08           65.20
2015-01-27 10           53.20
```

Im Teil 3 erweitern wir die Sensordaten um den Datentyp BSON. Dies und weitere Möglichkeiten lesen Sie in der Ausgabe März des Informix Newsletters.

WebTipp: Volltextrecherche zum INFORMIX Newsletter

Nachdem der INFORMIX Newsletter nunmehr schon seit mehr als 8 Jahren monatlich Tipps verbreitet, hat die CURSOR AG einen Service eingerichtet, der die Suche in den Artikeln der bisherigen Ausgaben erleichtert.



Unter dem Link:
http://www.cursor-distribution.de/de/community/community-informix/informix-newsletter/informix-newsletter-suche
kann in allen bisher erschienen Artikeln komfortabel gesucht werden.

Hinweis: OAT-PlugIn Verteiler

Nachdem immer wieder Kunden nach den aktuellsten Versionen unserer PlugIns zum OAT fragen, erstellen wir einen Verteiler, an den bei grösseren Anpassungen oder Erweiterungen unsere PlugIns versandt werden können.
Falls Sie daran Interesse haben, dann schreiben Sie uns dies an „ifmxnews@de.ibm.com“ mit dem Stichwort „OAT-PlugIns“.

In der neuesten PlugIn Version haben wir im Bereich „ADMIN“ die Sessions nunmehr so verlinkt, dass man aus der Übersicht der kostenintensivsten Statements direkt auf den Session-View kommt. Im selben Bereich gibt es NEU einen Anwahlpunkt „DBSpace free“, der je DBSpace den Prozentsatz anzeigt, zu dem der DBSpace noch freie Pages beinhaltet. Mehr Details zu diesen Erweiterungen in der Ausgabe März 2015 des Newsletters.

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

Alexander Körner Channel Technical Sales, IMT Germany -
Informix/SOA Integration and Informix
Certified Senior IT-Specialist
IBM Certified Solutions Expert IDS, DB2, Rational & XML
akoerner@de.ibm.com +49 89 4504 1423

Sowie unterstützende Teams im Hintergrund.
Fotonachweis: Gerd Kaluzinski (Blick von der Burgruine Hohenems)