Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

im Mai fand im Hofbräukeller in München die diesjährige INFORMIX Infobahn und der 62. Workshop der Informix User Group statt. Beide Veranstaltungen waren sehr gut besucht und boten die Möglichkeit Erfahrungen auszutauschen und neue Kontakte zu knüpfen (mehr dazu unten in einer Nachlese).

Die Vielzahl an neuen Features der vorgestellten Version 12.10 bestimmt den Inhalt dieses Newsletters. Seinen Sie neugierig und testen Sie selbst, welche Features Ihnen die Arbeit erleichtern.

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

Ihr TechTeam

TechTipp: In Transaktion ? - Wie erkenne ich, ob eine TA offen ist ?

Die Frage, ob man bereits in einer offenen Transaktion steht, oder ob Transaktionen überhaupt in der entsprechenden Datenbank unterstützt werden, ist mit einer SQL-Abfrage nicht ganz leicht zu beantworten. Da sich diese Frage aber häufiger stellt, wollen wir eine Prozedur vorstellen, die diese Frage beantwortet. Dabei geht die Prozedur nach dem Prinzip „Trial&Error“ vor, versucht also eine Transaktion auf zu machen. Gelingt dies, dann wird die Transaktion sofort wieder beendet und der Wert 0 zurückgegeben, da man ja nicht innerhalb einer Transaktion war.
Scheitert der Versuch mit dem Fehler „-535 already in transaction“, so wird der Wert 1 zurückgeliefert, da man in einer offenen Transaktion steht.
Kommt hingegen der Fehler „-256 transaction not available“ zurück, so ist die Datenbank ohne Transaktionslogging erstellt und somit liefert die Prozedur den Wert -1 zurück.

Hier die Procedure als Beispiel:

```
create procedure query_ta_flag()
returning smallint
define sqlerr smallint;
define isamerr smallint;
define retflag smallint;
define txt varchar(60);

on exception set sqlerr,isamerr
        if (sqlerr = -535) then let txt = "already in transaction"; let retflag
= 1; end if;
        if (sqlerr = -256) then let txt = "transaction not available"; let
retflag = -1; end if;
        return retflag;
end exception;

let sqlerr = 0;
let isamerr = 0;
let retflag = 0;
let txt = "";

begin work;
        let txt = "not in transaction"; let retflag = 0;
commit work;
        return retflag;

end procedure;
```

Sicher gibt es eine Reihe weiterer Möglichkeiten den Transaktionsstatus abzufragen. Wenn Sie eine elegante Lösung dazu gefunden haben, schreiben Sie uns diese, damit wir sie hier den Lesern vorstellen können.

TechTipp: RANK() - Die Reihenfolge der Ausgaben einer Abfrage

Die Funktionen RANK() und DENSERANK() bieten die Möglichkeit eine Reihung der Ausgabewerte vorzunehmen. Hierbei wird eine interne Reihung der Ergebnisse vorgenommen und diese mit dem entsprechenden Rang versehen.

```
SELECT c.lname, o.order_date, SUM(i.total_price) AS umsatz,
   RANK () OVER (PARTITION BY c.lname ORDER BY SUM(i.total_price) DESC)
          AS umsatz_tag
FROM orders o, items i, customer c
WHERE o.order_num = i.order_num
    AND o.customer_num = c.customer_num
GROUP BY c.lname,o.order_date
ORDER BY c.lname, umsatz_tag
```

Das Ergebnis zeigt in der Spalte „umsatz_tag“ den Rang des Ergebnisses innerhalb des Ergebnisses zum Kunden.

lname	order_date	umsatz	umsatz_tag
Grant	23.12.2008	84.00 €	1
Grant	17.06.2008	84.00 €	1
Grant	27.12.2008	78.00 €	3
Hanlon	11.07.2008	438.00 €	1
Jaeger	07.06.2008	940.00 €	1
Jaeger	27.06.2008	450.00 €	2
Jewell	09.07.2008	584.00 €	1
Keyes	14.06.2008	450.00 €	1
Lawson	30.05.2008	448.00 €	1
Lechner	22.05.2008	959.00 €	1
Lechner	20.05.2008	250.00 €	2
Lechner	22.06.2008	143.80 €	3
Lechner	18.06.2008	99.00 €	4
Neelie	24.07.2008	232.00 €	1

Bei den Bestellungen von „Grant“ ist zu sehen, dass die ersten beiden Bestellungen mit Rang 1 gelistet werden, die nächste Bestellung dann mit Rang 3. Soll eine Reihung ohne Lücken erfolgen, so kann die Funktion DENSERANK() genutzt werden, die dann nach den beiden ersten Plätzen den 2. Platz ausgibt:

lname	order_date	umsatz	umsatz_tag
Grant	23.12.2008	84.00 €	1
Grant	17.06.2008	84.00 €	1
Grant	27.12.2008	78.00 €	2
Hanlon	11.07.2008	438.00 €	1

Als weitere Funktionen stehen **PERCENT_RANK()** und **CUME_DIST()** zur Verfügung, die den Rang als Prozent bzw. kumulierte Prozent ausgeben.
Die Funktion **NTILE (x)** teilt das Ergebnis in x nahezu gleich große Gruppen auf, zu denen Werte berechnet werden können.

TechTipp: LEAD() - Folgenden Wert einbeziehen

Die Funktion LEAD() bietet die Möglichkeit, den folgenden Wert mit in die Berechnung einzubeziehen. Diese Funktionalität lässt sich an folgendem Beispiel demonstrieren. Es soll zur Summe einer Bestellung immer auch die Differenz zur nachfolgenden Bestellsumme mit angezeigt werden:

```
SELECT c.lname, o.order_date, SUM(i.total_price) AS umsatz,
   LEAD (SUM(i.total_price),1) OVER (PARTITION BY c.lname ORDER BY
SUM(i.total_price) DESC) - SUM(i.total_price) AS diff_to_next
      FROM orders o, items i, customer c
      WHERE o.order_num = i.order_num
          AND o.customer_num = c.customer_num
       GROUP BY c.lname,o.order_date
       ORDER BY c.lname, umsatz DESC
```

Das Ergebnis sieht dann folgendermassen aus:

```
lname           order_date              umsatz        diff_to_next

Grant           23.12.2008           84.00 €           0.00 €
Grant           17.06.2008           84.00 €          -6.00 €
Grant           27.12.2008           78.00 €
Hanlon          11.07.2008          438.00 €
Jaeger          07.06.2008          940.00 €        -490.00 €
Jaeger          27.06.2008          450.00 €
Jewell          09.07.2008          584.00 €
Keyes           14.06.2008          450.00 €
Lawson          30.05.2008          448.00 €
Lechner         22.05.2008          959.00 €        -709.00 €
Lechner         20.05.2008          250.00 €        -106.20 €
Lechner         22.06.2008          143.80 €         -44.80 €
Lechner         18.06.2008           99.00 €
```

Der Offset als 2. Argument im LEAD gibt an, auf welchen der folgenden Datensätze sich die Berechnung bezieht. So kann z.B. mit Offset 2 immer der übernächste Datensatz verglichen werden.

```
… LEAD (SUM(i.total_price),2)...
```

Würde dann folgendes Ergebnis liefern:

```
...
Lechner         22.05.2008          959.00 €        -815.20 €
Lechner         20.05.2008          250.00 €        -151.00 €
Lechner         22.06.2008          143.80 €
Lechner         18.06.2008           99.00 €
...
```

TechTipp: LAG() - Vorausgehenden Wert einbeziehen

Die Funktion LAG() bietet die Möglichkeit den vorausgehenden Wert mit in die Berechnung einzubeziehen. Die Abfrage aus dem TechTipp zu LEAD() sieht dann unter Verwendung von LAG() so aus, dass nun die Differenz immer hinter dem neuen Bestellwert steht:

```
SELECT c.lname, o.order_date, SUM(i.total_price) AS umsatz,
   LAG (SUM(i.total_price),1) OVER (PARTITION BY c.lname ORDER BY
SUM(i.total_price) DESC) - SUM(i.total_price) AS diff_to_next
      FROM orders o, items i, customer c
      WHERE o.order_num = i.order_num
          AND o.customer_num = c.customer_num
       GROUP BY c.lname,o.order_date
       ORDER BY c.lname, umsatz desc
```

Ergebnis:

lname	order_date	umsatz	diff_to_next
Grant	23.12.2008	84.00 €	
Grant	17.06.2008	84.00 €	0.00 €
Grant	27.12.2008	78.00 €	6.00 €
Hanlon	11.07.2008	438.00 €	
Jaeger	07.06.2008	940.00 €	
Jaeger	27.06.2008	450.00 €	490.00 €
Jewell	09.07.2008	584.00 €	
Keyes	14.06.2008	450.00 €	
Lawson	30.05.2008	448.00 €	
Lechner	22.05.2008	959.00 €	
Lechner	20.05.2008	250.00 €	709.00 €
Lechner	22.06.2008	143.80 €	106.20 €
Lechner	18.06.2008	99.00 €	44.80 €
Neelie	24.07.2008	232.00 €	

Sowohl bei LEAD(), als auch bei LAG() lässt sich mit dem zusätzlichen Schlüsselwort „RESPECT NULLS“ bzw. „IGNORE NULLS“ die Behandlung von Null-Werten anpassen. Der Default ist „RESPECT NULLS“, wobei Null-Werte wie üblich mit ausgegeben und in die Berechnung mit einbezogen werden.

Zudem lässt sich ein Default-Wert als drittes Argument von LEAD() und LAG() angeben, der im Falle von Null-Werten dann zurückgegeben wird.

TechTipp: ROWNUMBER() - Zeilennummern ausgeben

Die Funktion ROWNUMBER() gibt zu den Ergebniszeilen eindeutige Zeilennummern aus. Nutzt man die Funktion ohne OVER(), so werden die Werte vor dem Sortieren nummeriert.

Beispiel:

```
SELECT ROWNUMBER() OVER(), c.lname, c.fname
      FROM customer c
      ORDER BY c.lname, c.fname
```

Ergebnis:

```
       (row_number) lname           fname

                  18 Albertson       Frank
                  28 Baxter          Dick
                  13 Beatty          Lana
                  26 Currie          Philip
                  14 Grant           Alfred
                   5 Hanlon          Marvin
                   1 Henry           James
                  17 Jaeger          Roy
                  10 Jewell          Fred
                  …
```

Um wirklich eine Nummerierung der Ergebniszeilen zu erhalten muss die Abfrage umgeschrieben werden in:

```
SELECT ROWNUMBER() OVER(PARTITION BY "" ORDER BY c.lname,c.fname), c.lname,
c.fname, c.state
      FROM customer c
      ORDER BY c.lname, c.fname
```

Ergebnis:

```
       (row_number) lname           fname           state

                   1 Albertson       Frank            CA
                   2 Baxter          Dick             CA
                   3 Beatty          Lana             CA
                   4 Currie          Philip           CA
                   5 Grant           Alfred           CA
                   6 Hanlon          Marvin           FL
                   7 Henry           James            MA
                   8 Jaeger          Roy              CA
                   9 Jewell          Fred             AZ
                  10 Keyes           Frances          CA
                  11 Lawson          Margaret         CA
                  12 Lechner         Anthony          CA
```

Die eigentliche Aufgabe der ROWNUMBER ist die Nummerierung innerhalb einer Gruppe. Also z.B. alle Kunden aus einem Bundesstaat nummeriert auflisten, was im folgenden Beispiel gezeigt wird:

```
SELECT ROWNUMBER() OVER(partition by c.state order by c.lname), c.lname,
c.fname, c.state
      FROM customer c
      ORDER BY c.state, 1, c.lname, c.fname
```

Ergebnis:

```
        (row_number) lname           fname           state

                   1 Jewell          Fred            AZ
                   2 Lessor          Frank           AZ
                   1 Albertson       Frank           CA
                   2 Baxter          Dick            CA
                   3 Beatty          Lana            CA
                   4 Currie          Philip          CA
                   5 Grant           Alfred          CA
                   6 Jaeger          Roy             CA
                   7 Keyes           Frances         CA
                   8 Lawson          Margaret        CA
                   9 Lechner         Anthony         CA
```

TechTipp: INTERSECT

Die Möglichkeit, eine Schnittmenge über mehrere Abfragen zu bilden macht in einigen Bereichen die SQL-Programmierung leichter. Anstatt temporäre Tabellen zu erstellen und diese dann mittels Key zu vergleichen, bekommt man so mit einem Statement das gewünschte Ergebnis.

Beispiel: Es sollen alle Bestellungen ausgegeben werden, deren Gesamtbetrag über 1000.- liegt und die mindestens einen Artikel des Herstellers „HRO“ beinhalten

```
SELECT UNIQUE  c.lname, c.fname, o.order_num, o.order_date
FROM customer c, orders o, items i
WHERE c.customer_num = o.customer_num
AND i.order_num = o.order_num
AND c.state = "CA"
GROUP by 1,2,3,4
HAVING SUM(i.total_price) > 1000
INTERSECT
SELECT UNIQUE  c.lname, c.fname, o.order_num, o.order_date
FROM customer c, orders o, items i
WHERE c.customer_num = o.customer_num
AND i.order_num = o.order_num
AND i.manu_code = "HRO"
```

Der obere Abschnitt liefert 5 Ergebnisse, der untere Abschnitt 4 Ergebnisse. Das Gesamtergebnis beinhaltet 3 Ergebniszeilen. Ausgegeben werden nur die Zeilen, die in allen Spalten mit einem Satz der zweiten Abfrage übereinstimmen.

TechTipp: EXCEPT

Die Möglichkeit, vom Ergebnis einer SQL-Abfrage die Ergebnisse einer anderen Abfrage abzuziehen besteht mit der Funktion EXCEPT.

Beispiel: Es sollten alle Bestellungen ausgegeben werden, deren Gesamtbetrag über 1000.- liegt und die keine Artikel des Herstellers HRO= enthalten

```
SELECT UNIQUE  c.lname, c.fname, o.order_num, o.order_date
FROM customer c, orders o, items i
WHERE c.customer_num = o.customer_num
AND i.order_num = o.order_num
AND c.state = "CA"
GROUP by 1,2,3,4
HAVING SUM(i.total_price) > 1000
EXCEPT
SELECT UNIQUE  c.lname, c.fname, o.order_num, o.order_date
FROM customer c, orders o, items i
WHERE c.customer_num = o.customer_num
AND i.order_num = o.order_num
AND i.manu_code = "HRO"
```

TechTipp: CASE/DECODE in ORDER BY

Die Sortierung der Ausgabe von Datenbankabfragen stellt oft eine Herausforderung dar. Waren die Anforderungen nicht mit herkömmlichen SQL-Mitteln zu bewältigen blieb oft nur die Alternative, der Abfrage eine „Sortierspalte“ hinzuzufügen oder das Ergebnis zuerst in eine temporäre Tabelle zu schreiben, auf der dann die weitere Sortierung erfolgen konnte.

Mit der Version 12.10 wird nun die Verwendung von CASE/DECODE im ORDER BY unterstützt. Dies ermöglicht, bei bekannten Ergebniswerten eine beliebige Sortierung zu erzwingen.

Das folgende Beispiel zeigt, wie bei der Abfrage der Kunden zuerst die Kunden aus ausgewählten Bundesländern sortiert nach Namen ausgegeben werden, danach diejenigen aus den restlichen Bundesländern.

Beispiel zu CASE:

```
SELECT customer_num, lname, state
FROM customer
ORDER BY CASE
                WHEN state = "NJ" THEN 1
                WHEN state = "AZ" THEN 2
                WHEN state = "CA" THEN 3
                ELSE 99
        END, lname
```

Beispiel zu DECODE:

```
SELECT customer_num, lname, state
FROM customer
ORDER BY DECODE (state, "NJ",1, "AZ",2, "CA",3, 99) , lname
```

Das Ergebnis sieht dann so aus:

```
customer_num lname            state
         122 O'Brian          NJ
         119 Shorter          NJ
         120 Jewell           AZ
         128 Lessor           AZ
         114 Albertson        CA
         118 Baxter           CA
         113 Beatty           CA
         103 Currie           CA
         115 Grant            CA
…
         117 Sipes            CA
         105 Vector           CA
         106 Watson           CA
         123 Hanlon           FL
         125 Henry            MA
         126 Neelie           CO
         124 Putnum           OK
         127 Satifer          NY
         121 Wallack          DE
```

Info: Nachlese zur INFORMIX Infobahn und zum 62. IUG Workshop

Mitte Mai fand eine kombinierte Veranstaltung der IBM und der Informix User Group Deutschland im Hofbräukeller in München statt. Die IBM startete die Veranstaltung am ersten Tag mit der Informix Infobahn, bei der der Schwerpunkt auf der neuen Informix Version 12.10 und deren Features lag. Am zweiten Tag wurde die Veranstaltung mit dem IUG-Workshop fortgesetzt.
Beide Veranstaltungen waren bis auf den letzten Platz gefüllt, was das steigende Interesse an Informix widerspiegelt.

Das gemeinsame Thema beider Veranstaltungen war Big-Data. Jeder spricht davon, aber die wenigsten fühlen sich wirklich davon betroffen. Daher lag der Schwerpunkt beim IUG-Workshop auf den praxisorientierten Herausforderungen im Umgang mit großen Datenmengen.

IBM hatte den Fokus auf dem theoretischen Ansatz, mittels Informix den Umgang mit großen Datenmengen und den aktuell vorhandenen Anforderungen in der IT zu meistern. Bestehende und neue Funktionalitäten in der neuen Version 12 unterstützen dabei die Entwicklung von Anwendungen und Prozessen auf allen Ebenen.

Beim IUG-Workshop wurden Big-Data Themen angesprochen wie:
- Hardware
- Umgang in Projekten
- Operating
- Datenschutz
- Übergreifende Prozesse und Datenhaltung
- Was bedeutet Big-Data für mich?

Den Einstieg übernahm Stephan Reimann (IBM), der die Definitionen von Big-Data noch einmal zusammengefasst hat und aufzeigte, welche Architekturen zur Verfügung stehen.

Im Anschluss zeigte Werner Wellmann (Ordix) anhand eines durchgeführten Projektes, welche hohe Performance Steigerung eine Unstellung auf Informix/IWA im Bereich SmartMetering bewirken kann und welche Herausforderungen damit in Verbindung stehen.

Auch die Hardware wurde in diesem Zusammenhang mit Big-Data näher betrachtet. Herr Thomas Nelleßen (Asus) zeigte, wie man mit Standard-Server Hardware den steigenden Leistungsanforderungen kostengünstig und zukunftsorientiert entgegentreten kann. Dazu stellte Asus eine Testserver zur Verfügung. Sandor Szabo und Andreas Breitfeld (IBM) zeigten auf diesem Server welche Leistungen mit und ohne IWA möglich sind.

Herr Dirk Paasche (HM Informatik) erklärte, wie mittels dem Stichwort 'Industrie 4.0', Geschäftsprozesse und Dienstleistungen übergreifend zusammengefasst werden können und welche Schlüsselrolle Informix dabei einnehmen kann.

Am Nachmittag gab es eine Podiumsdiskussion mit Vertretern der IBM, einem Berater und Kunden, die sich mit dem Umgang und den Herausforderung von Massendaten im täglichen Leben befassten. Es zeigte sich schnell, dass nicht nur technische Belange zu berücksichtigen sind, sondern auch rechtliche, wie z.B. Datenschutz.

Den Abschluß machte Gerd Kaluzinski (IBM), der am Beispiel eines konkreten Projektes (bei einem Kunden in Norddeutschland) anschaulich vorstellte, wie der Informix Warehouse Accelerator in einem mittelständischen Unternehmen die neue Möglichkeit schafft, die Gesamtheit der Betriebsdaten für bislang nicht durchführbare Businessanalysen und Vorhersagen zu nutzen.

Fazit: Big-Data geht alle an. Jede Firma und sogar jede Privatperson erfasst/sammelt mehr und mehr Daten. Der wachsende Aufwand der Verwaltung und die daraus resultierenden Kosten müssen laufend durch neue Technologien, Prozesse und rechtliche Gegebenheiten kompensiert werden.
Daher muss jedes Projekt im Hinblick auf Big-Data einzeln betrachtet und bewertet werden.

Die große Teilnehmerzahl beider Veranstaltungen unterstreichen das aktuell hohe Interesse und den Bedarf an Informationen zum Thema Big-Data.

Alle Präsentationen befinden sich auf den Seiten der Informix User Group:
http://www.iug.de/index.php?option=com_content&task=view&id=246&Itemid=340

Anmeldung / Abmeldung / Anmerkung

Der Newsletter wird ausschließlich an angemeldete Adressen verschickt. Die Anmeldung erfolgt, indem Sie eine E-Mail 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.bytec.de/de/software/ibm_software/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

Nachlese zur INFORMIX Infobahn und dem 62. Workshop der Informix User Group:

Knuth J.Hartlieb IT-Consulting & Databases
mail@knuth-hartlieb.de
http://www.knuth-hartlieb.de

Sowie unterstützende Teams im Hintergrund.

Fotonachweis: Gerd Kaluzinski (Schwanennachwuchs Köchlinweiher Lindau)