Willkommen zum Informix Newsletter

Liebe Leserinnen und Leser,

die Zeit der Weihnachtsmärkte ist gekommen. Winterlandschaften und Glühwein motivieren dazu, sich aus der wohligen Wärme zumindest zeitweise nach draussen zu begeben.

In der Küche duftet es jedes Wochenende nach einer anderen Sorte an Weihnachtsgebäck.
Mit dieser Stimmung wollen wir ihnen Lesestoff für die wenigen Tage liefern, an denen es ausserhalb der Wohnung doch eher ungemütlich ist. Dafür haben wir einen Newsletter mit allerlei interessanten Zutaten erstellt. Wir basteln uns einen Task und spielen mit Nullen und Leerstrings.
Die Ergebnisse hängen wir dann an den Weihnachtsbaum.

Viel Spass mit dem aktuellen Newsletter !
Ihr TechTeam

Inhaltsverzeichnis

TechTipp: Pending Alter Table TASK - Part 1

In der Datenbank sysadmin sind eine Reihe an Tasks definiert. So z.B. auch ein Task, der "Pending Alter Tables" erkennt, und die gefundenen Information in die Tabelle "ph_alerts" schreibt:

```
tk_name             check_for_ipa
tk_description      Find tables with outstanding in place alters
tk_type             TASK
tk_sequence         0
tk_result_table
tk_create
tk_dbs              sysadmin
tk_execute          check_for_ipa
tk_delete            30 00:00:00
tk_start_time       04:00:00
tk_stop_time
tk_frequency          7 00:00:00
tk_next_execution
tk_total_executio+  0
tk_total_time       0.00
tk_monday           t
tk_tuesday          t
tk_wednesday        t
tk_thursday         t
tk_friday           t
tk_saturday         t
tk_sunday           t
tk_attributes       4
tk_group            SERVER
tk_enable           f
tk_priority         0
```

Damit könnten wir, falls dieser Task aktiviert wurde, in der Tabelle "ph_alerts" sehen, welche Tabellen bereinigt werden müssten.

Kunden des INFORMIX HealthChecks erhalten diese Information im Rahmen des monatlichen Checks incl. der Befehle, die zur Bereinigung erforderlich sind.
Nun kam die Frage auf, ob es auch einen Task gibt, der diese Tabellen gleich automatisch bereinigt, anstatt die Information nur zu protokollieren.

Die Antwort lautet:
Nicht als Default Task, aber es ist (relativ) einfach solch einen Task zu erstellen.
Da wir zuletzt im Mai 2008 im Newsletter einen Beitrag veröffentlicht hatten, wie man Tasks erstellt, nehmen wir dies zum Anlass die Erstellung des Tasks im Detail nochmals zu vorzustellen.

TechTipp: Pending Alter Table TASK - Part 2

In der Datenbank sysmaster lassen sich die Tabellen abfragen, die bereinigt werden sollten. Aus dieser Abfrage können auch gleich die SQL-Statements generiert werden, die zur Bereinigung der "Pending Alter Table" Zustände genutzt werden können:

```
database sysmaster;
select "execute function sysadmin:task('table update_ipa paral-
lel','"||trim(n.tabname)||"','"||trim(n.dbsname)||"');"
from sysptnhdr h, systabnames n
where h.partnum = n.partnum
and n.tabname[1] != ' '
and h.pta_totpgs > 0
;
```

Ergebnis in unserer Testumgebung:

```
(expression)  execute function sysadmin:task('table update_ipa pa-
rallel','test3','stores');
(expression)  execute function sysadmin:task('table update_ipa pa-
rallel','test2','stores');
(expression)  execute function sysadmin:task('table update_ipa pa-
rallel','test4','stores');
(expression)  execute function sysadmin:task('table update_ipa pa-
rallel','test5','stores');
(expression)  execute function sysadmin:task('table update_ipa pa-
rallel','test1','stores');
```

Diesen Aufruf wollen wir nun in einem Task automatisiert laufen lassen.

Funktionen, die als Tasks in der Datenbank sysadmin aufgerufen werden, erhalten als Übergabeparameter immer die Task-ID, sowie die Task-Sequenz Nummer:

```
CREATE PROCEDURE "informix".update_ipa_kalu(task_id INTEGER,
task_seq INTEGER)
   RETURNING integer
```

Zurückgegeben wird der Returncode, welcher entweder 0 für Success, oder die SQL-Fehlernummer enthält.

Da wir ggf. nicht alle Tabellen bei einem Aufruf bereinigen wollen (um z.B. nicht den Tagesbetrieb zu stören), verwenden wir Parameter um sowohl die Anzahl der Tabellen, als auch die PDQPRIORITY, mit der der Aufruf erfolgen soll, zu steuern.

Diese Parameter tragen wir in die Tabelle ph_threshold ein.

TechTipp: Pending Alter Table TASK - Part 3

Für die Realisierung des Aufrufs aus einer Prozedur gibt es einige Optionen.
Bei den Tests mussten wie erkennen, dass nicht alle Ideen realisiert werden können.
Zur Vollständigkeit listen wir hier aber auch diese Ideen auf:

1. EXECUTE IMMEDIATE

   Dabei war der Gedanke das Statement für den Task zusammenzusetzen und dann mittels "execute immediate cmd" den Befehl auszuführen.
   Leider mussten wir recht schnell erkennen, dass es hier eine Beschränkung gibt:

```
The EXECUTE IMMEDIATE statement cannot execute the following SQL statements.
    --EXECUTE
    --EXECUTE FUNCTION
    --EXECUTE PROCEDURE
```

2. SYSTEM Command

   Der Umweg über ein, im Filesystem zusammengebautes Statement, und die anschliessende Ausführung dieses Statements über das System Command war erfolgreich, hinterliess aber den Eindruck einer "Notlösung":

```
SYSTEM 'echo "'||trim(p_cmd)||'" >/tmp/tmp.update_ipa_001.sql';
SYSTEM 'echo "'||trim(t_cmd)||'" >>/tmp/tmp.update_ipa_001.sql';
SYSTEM 'dbaccess sysadmin /tmp/tmp.update_ipa_001.sql'
```

3. Funktion admin()

Die Admin Funktion lässt sich innerhalb von Prozeduren aufrufen und kann den Task "table update_ipa parallel" ausführen.

```
EXECUTE IMMEDIATE p_cmd;
SELECT admin('table update_ipa parallel',trim(t_tabname),trim(t_dbname))
    into cmd_nr
FROM systables WHERE tabid=1;
SELECT cmd_ret_status into rc FROM command_history WHERE cmd_number = cmd_nr;
EXECUTE IMMEDIATE p0_cmd;
```

Damit fiel die Wahl für die Realisierung auf diese dritte Option.

Den Aufbau der Prozedur haben wir aus einer der Bestehenden Prozeduren in der Datenbank sysadmin übernommen, und diese an unsere Bedürfnisse angepasst:

```
CREATE PROCEDURE "informix".update_ipa_kalu(task_id INTEGER,
task_seq INTEGER)
   RETURNING integer

DEFINE rc            INTEGER;
DEFINE cmd_nr        INTEGER;
DEFINE cnt           INTEGER;
DEFINE t_id          INTEGER;
DEFINE t_cmd         CHAR(8192);
DEFINE t_pages       bigint;
DEFINE t_tabname     varchar(128);
DEFINE t_dbname      varchar(128);
DEFINE p_cmd         CHAR(200);
DEFINE p0_cmd        CHAR(200);
DEFINE param_pdq     INTEGER;
DEFINE param_cnt     INTEGER;

    LET rc      = 0;
    LET cmd_nr  = 0;
    LET cnt     = 0;
    LET p0_cmd = "SET PDQPRIORITY 0;";

-- Get the config threshold for PDQ --
    SELECT MAX(value::integer) INTO param_pdq
    FROM sysadmin:ph_threshold
    WHERE name = "IPA_PDQ" and task_name = "ipa_command";

    IF param_pdq IS NULL THEN LET param_pdq = 0;
    ELIF param_pdq < 0 THEN LET param_pdq = 0;
    ELIF param_pdq > 100 THEN LET param_pdq = 100;
    END IF

    IF param_pdq > 0 AND is_pdq_allowed() > 0
    THEN LET p_cmd = "SET PDQPRIORITY " ||param_pdq||";";
    ELSE LET p_cmd = "SET PDQPRIORITY 0;";
    END IF

-- Get the config threshold for COUNT --
    SELECT MAX(value::integer) INTO param_cnt
    FROM sysadmin:ph_threshold
    WHERE name = "IPA_CNT" and task_name = "ipa_command";

    IF param_cnt IS NULL THEN LET param_cnt = 100;
    ELIF param_cnt < 0 THEN LET param_cnt = 100;
    ELIF param_cnt > 100 THEN LET param_cnt = 1;
    END IF
```

```
-- Cleanup IPA Commands in ipa_command table --
delete from ipa_command where 1=1;

-- Insert IPA Commands into ipa_command table --
insert into ipa_command
select 0,"execute function sysadmin:task('table update_ipa paral-
lel','"||trim(n.tabname)||"','"||trim(n.dbsname)||"');", h.npused,
n.tabname, n.dbsname
from sysmaster:sysptnhdr h, sysmaster:systabnames n
where h.partnum = n.partnum
and n.tabname[1] != ' '
and h.pta_totpgs > 0
order by h.npused desc
;

FOREACH SELECT  --+ FIRST_ROWS
    ipa_cmd_id, RTRIM(ipa_cmd_exe), ipa_pages, ipa_table, ipa_db
    INTO
    t_id, t_cmd, t_pages, t_tabname, t_dbname
    FROM  ipa_command
    ORDER BY  ipa_pages DESC

   IF cnt < param_cnt THEN
    -- nur die ersten x Tabellen bereinigen --
        EXECUTE IMMEDIATE p_cmd; -- PDQ setzen
        SELECT admin('table update_ipa parallel',
            trim(t_tabname),trim(t_dbname))
        into cmd_nr
        FROM systables WHERE tabid=1;
        SELECT cmd_ret_status into rc
        FROM command_history
        WHERE cmd_number = cmd_nr;
        EXECUTE IMMEDIATE p0_cmd;
        let cnt = cnt +1;
    END IF

END FOREACH
RETURN rc;

END PROCEDURE;
```

TechTipp: Pending Alter Table TASK - Part 4

Was noch fehlt, ist eine Zwischentabelle für die auszuführenden Befehle, sowie der Eintrag des Tasks in die Tabelle ph_task. Zudem sollte ein Defaultwert für die Parameter IPA_PDQ und IPA_CNT in die Tabelle ph_threshold eingetragen werden.

```
-- Create table for IPA Commands --
CREATE TABLE if not exists informix.ipa_command (
    ipa_cmd_id              SERIAL,
    ipa_cmd_exe             LVARCHAR(8192),
    ipa_pages           BIGINT,
    ipa_table           varchar(128),
    ipa_db              varchar(128)
) LOCK MODE ROW;

-- clean up any old versions of ipa_command in ph_threshold --
delete from ph_threshold where task_name = "ipa_command";

-- insert new values in ph_threshold --
insert into ph_threshold (id, name, task_name, value, value_type,
description)
values (
    0,
    "IPA_PDQ",
    "ipa_command",
    10,
    "NUMERIC",
    "PDQ Priority for Update IPA Task"
)
;

insert into ph_threshold (id, name, task_name, value, value_type,
description)
values (
    0,
    "IPA_CNT",
    "ipa_command",
    2,
    "NUMERIC",
    "Count for tables in Update IPA Task"
)
;


-- Delete old versions of ipa_command from ph_task --
delete from ph_task where tk_name = 'ipa_command';
```

```
-- Insert task in ph_task"
insert into ph_task (tk_name, tk_description, tk_type, tk_dbs ,
tk_execute, tk_start_time, tk_stop_time, tk_frequency,
tk_monday, tk_tuesday, tk_wednesday, tk_thursday, tk_friday,
tk_saturday , tk_sunday,
tk_group, tk_enable
)
values ("ipa_command","AUTO IPA Cleanup created by KALU","TASK",
"sysadmin",
"update_ipa_kalu","02:00:00", "05:20:00", "0 01:00:00",
"t","t","t","t","t","t","t",
"PERFORMANCE ","t");
```

Hier wurde eine stündliche Ausführung zwischen 02:00 und 05:20 eingetragen, damit diese Bereinigung vor dem Tagesbetrieb läuft.

Abfrage des Tasks in sysadmin:ph_task:

```
select * from ph_task where th_name = 'ipa_command';

tk_id               73
tk_name             ipa_command
tk_description      AUTO IPA Cleanup created by KALU
tk_type             TASK
tk_sequence         18
tk_result_table
tk_create
tk_dbs              sysadmin
tk_execute          update_ipa_kalu
tk_delete             0 01:00:00
tk_start_time       02:00:00
tk_stop_time        05:20:00
tk_frequency          1 00:00:00
tk_next_execution   2025-12-04 02:00:00
tk_total_executio+  18
tk_total_time       3.044751000000
tk_monday           t
tk_tuesday          t
tk_wednesday        t
tk_thursday         t
tk_friday           t
tk_saturday         t
tk_sunday           t
tk_attributes       400
tk_group            PERFORMANCE
tk_enable           t
tk_priority         0
```

Sind diese Eintragungen erfolgt, startet der Task seine Arbeit, sobald die Start-Time erreicht ist.
Das Protokoll der Aufrufe kann abgefragt werden mittels:

```
select run_time, run_duration, run_retcode
from ph_run where run_task_id = (
    select tk_id
    from ph_task
    where tk_name = 'update_ipa_kalu'
);
```

run_time	run_duration	run_retcode
2025-12-04 02:00:03	0.09615	0
2025-12-05 02:00:03	0.036486	0
2025-12-06 02:00:04	0.299112	0
2025-12-07 02:00:03	0.10924	0
2025-12-08 02:00:05	0.065069	0
2025-12-09 02:00:02	0.012026	0
2025-12-10 02:00:01	0.241306	0

Hinweis:
Der hier vorgestellte Task versteht sich als Vorschlag, und nicht als die Musterlösung. Jegliche Verbesserung ist herzlich willkommen und sollte Eingang in das Repository der Informix User Group finden.
Bitte Rückmeldungen an info@iug.de und gerd.kaluzinski@de.ibm.com schicken.

TechTipp: group_concat

Mit Version 15.0 wurde eine neue Funktion eingeführt, die von einigen Kunden bereits nachgefragt wurde: group_concat().
Wir hatten im INFORMIX Newsletter 2021-Q2 bereits eine ähnliche, selbst entwickelte Funktion vorgestellt, die eine Art "customized listagg" zur Verfügung stellt.

Mit group_concat() lässt sich der Inhalt einer Spalte als zusammengesetzte Zeichenkette mit einem beliebig gewählten Separator ausgeben.

Beispiel:

```
select group_concat(tabname,',') as tables,
    group_concat(nrows::int,',') as rows,
    group_concat(npused::int,',') as pages
from systables where tabid > 99 and tabtype = 'T';
```

Ergebnis:

```
tables  customer,orders,manu-
fact,stock,items,state,call_type,cust_calls,catalog
        ,tab,warehouses,classes,employee
rows    28,23,9,74,67,52,5,7,74,1,4,4,1
pages   2,1,1,2,1,1,1,3,10,1,1,1,1

1 row(s) retrieved.
```

TechTipp: NULL, leer oder 0, das ist hier die Frage

Fast jede relationale Datenbank hat ein leicht abweichendes Verhalten bei der Behandlung von "null" Werten.
Betrachtet man Strings (char, varchar, lvarchar, nchar, ...), so zeigt sich, dass ein "concat" mit "null" immer "null" ergibt.
Beispiel:

```
create table null_test_c (
f1      char(10),
f2      char(10),
f3      char(10)
);
insert into null_test_c values ("Test1",null,null);
insert into null_test_c values ("Test2","Info",null);
insert into null_test_c values ("Test3",null,null);
insert into null_test_c values ("Test4","Zusatz",null);

select f1||f2
from null_test_c
order by 1;
```

Ergebnis: Die Zeilen, in denen eines der Felder "null" war, werden nicht mit ausgegeben.

```
Test2     Info
Test4     Zusatz
```

Abhilfe: Null-Value-Handling mit nvl():

```
select f1||nvl(f2 ,'')
from null_test_c
order by 1;

Test1
Test2     Info
Test3
Test4     Zusatz
```

Interessanter wird es noch, wenn es sich um Zahlen handelt.
Grundsätzlich sind alle Werte, die mit "null" kombiniert werden, immer "null", bzw. liefern kein Ergebnis. Dies gilt für die Grundrechenarten Plus, Minus, Mal, Geteilt.

Testaufbau: Tabelle mit 3 Spalten vom Typ INT;

```
insert into null_test_n values (1,null,null);
insert into null_test_n values (2,2,null);
insert into null_test_n values (3,null,1);
insert into null_test_n values (4,4,null);
insert into null_test_n values (null,null,null);
```

Nun zählen wir die Anzahl der Einträge:

```
select
        count(*), count(f1), count(f2), count(f3)
from null_test;

      (count(*))          (count)          (count)          (count)
               5                4                2                1
```

Der "count(fx)" auf eine Spalte zählt nur die Einträge, die nicht "null" sind, wohingegen der "count(*)" die Anzahl der Einträge gesamt ausgibt.

Nun betrachten wir den Durchschnittswert der Spalten:

```
select
        avg(f1), avg(f2), avg(f3)
from null_test_n;

            (avg)             (avg)             (avg)
    2.50000000000000 3.00000000000000 1.00000000000000
```

Auch hier werden für den Durchschnitt jeweils nur die Spalten genutzt, die nicht "null" sind.
Damit hat der Durchschnitt der Werte in Spalte f2 den Wert 3, da nur 2 Einträge ausgewertet werden mit "4" und "2".
Nun betrachten wir die Summen der Werte in den Spalten:

```
select
        sum(f1), sum(f2), sum(f3)
from null_test_n;

            (sum)             (sum)             (sum)
               10                 6                 1
```

Auch hier werden für die Summe jeweils nur die Spalten genutzt, die nicht "null" sind.

Nun kommen wir zur beliebtesten Ursache für falsche Berechnungen.
Wir teilen die Summe durch die Anzahl der Datensätze in der Tabelle:

```
select sum(f1)/count(*) as sum_cntx, sum(f2)/count(*) as sum_cntx,
sum(f3)/count(*) as sum_cntx
from null_test_n;

      sum_cntx       sum_cntx       sum_cntx
    2.500000000000 1.500000000000           0.25
```

Diese Werte weichen vom Durchschnitt, der avg() berechnet wurde, ab.

Um dieselben Ergebnisse wie die Funktion avg() zu erhalten, müsste die Abfrage folgendermassen umgebaut werden:

```
select
        sum(f1)/count(f1) as sum_cntf1, sum(f2)/count(f2) as
sum_cntf2, sum(f3)/count(f3) as sum_cntf3
from null_test_n;

     sum_cntf1       sum_cntf2       sum_cntf3
    2.500000000000 3.000000000000 1.000000000000
```

Wer nun noch nicht genug hat, dem empfehlen wir einmal das Speichern folgender Werte in einem CHAR(20) Feld, bzw. einem VARCHAR(20) Feld:

null	- Nullwert
""	- Leerstring
" "	- Ein Leerzeichen
" "	- Zwei Leerzeichen
" "	- Drei Leerzeichen

Unser Tipp: Nach jedem Versuch einen Glühwein, dann werden die Tests zumindest lustiger.

TechTipp: ONCONFIG - LOGREC_MAXBUFS

Der Parameter LOGREC_MAXBUFS bestimmt, mit wie vielen Buffern (zu je 256k) ein "Fast Recovery" aus den Logischen Logs stattfindet (z.B. beim Start der Instanz), bzw. falls es sich um einen Secondary Servern handelt, wie viele Buffer für das Nachfahren der Logs des Primary Servers während der Replikation zur Verfügung stehen.
Der Parameter ist in der onconfig.std nicht vorhanden und kann optional gesetzt werden, um z.B bei hohen Änderungsraten mehr Spielraum für die Replikation einzurichten, falls diese nicht als "Synchron" definiert wurde.
Die Anzahl sollte dabei nicht zu gross gewählt werden und 500 nicht übersteigen.

TechTipp: ONCONFIG - SEC_LOGREC_MAXBUFS

Dieser Parameter, der angab, wie viele 16k Buffer für das Nachführen der Loginformationen auf einem Secondary definiert sind, wurde durch den Parameter LOGREC_MUXBUFS ersetzt.

Frohes Fest

Sobald die Rentiere gefüttert sind, verabschieden wir uns mit dem grossen Schlitten in die Weihnachtsferien.

Nutzung des INFORMIX Newsletters

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit.
Die IUG hat sich dankenswerterweise dazu bereit erklärt, den INFORMIX Newsletter auf ihren Webseiten zu veröffentlichen.
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 gefundenen Tippfehler dürfen zudem behalten und nach Belieben weiterverwendet werden.
Eine Weiterverbreitung in eigenem Namen (mit Nennung der Quelle) oder eine Bereitstellung auf der eigenen HomePage ist ausdrücklich erlaubt. Alle hier veröffentlichen Scripts stehen uneingeschränkt zur weiteren Verwendung zur Verfügung.
Wir würden uns über eine Information freuen, wann und wo unsere Inhalte weiterverbreitet werden.

Die Autoren dieser Ausgabe

Andreas Legner **INFORMIX Development**
HCL Software

Martin Fuerderer **Database Development**
HCL Software

Gerd Kaluzinski **IBM Expert Labs**
Data & AI
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Herzlichen Dank an die vielen Helfer im Hintergrund.

Nicht zu vergessen der Dank an die Informix User Group, ohne die es den INFORMIX Newsletters heute nicht mehr geben würde, und die dankenswerterweise die Verteilung übernimmt.

Foto Nachweis:
Hafenweihnacht Lindau (Alter Leuchtturm) (Gerd Kaluzinski)
Schlitten am Weihnachtsmarkt Bregenz (Inja Schneider)