Willkommen zum Informix Newsletter

Liebe Leserinnen und Leser,

der Sommer ist da und die Urlaubszeit beginnt. Unser Team verbringt inzwischen mehr Zeit am See als in klimatisierten Serverräumen.
Um einen Gedanken eines Lesers (*) wieder aufzugreifen:
"Man kann den Newsletter ausdrucken, laminieren und dann sicher zum Baden mitnehmen".
Um Sie nicht zu sehr vom Baden und Grillen abzulenken, haben wir in dieser Ausgabe eher "leichte Kost" verarbeitet.
Unter dem Motto "... und das gibt es auch noch" widmen wir uns Funktionen, die eher unbekannt, aber doch nützlich sind.
Zudem freuen wir uns, wieder einmal einen Gastbeitrag vorstellen zu können.

Viel Spass mit dem aktuellen Newsletter !
Ihr TechTeam

(* Danke Dirk !)

Inhaltsverzeichnis

TechTipp: INFORMIX im UNIX Cluster

Wird Informix in einem Unix Cluster installiert, so reicht es nicht, dass nur die INFORMIX Ressource Group ($INFORMIXDIR) und ggf. das HomeVerzeichnis vom Benutzer "informix" mit dem Cluster Schwenk auf den anderen Server gemountet wird.
Den License Manager von Informix benötigt im Betriebssystem Packages aus dem Verzeichnis gskit. Fehlen diese, läuft die Instanz als Developer Edition. Im praktischen Fall bedeutet dies, dass z.B. eine Version 14.10.FC13EE nach dem Schwenk der Ressourcen sich auf dem zweiten Server als 14.10.FC13DE meldet und damit auch nur die erlaubten Optionen der Developer Edition annimmt.

Abhilfe: Nach der Installation muss die Informix Installation einmalig auf den zweiten Server geschwenkt werden. Auf diesem ist dann als User "root" der Befehl **"installgskit"** auszuführen, der sich im Verzeichnis $INFORMIXDIR/gskit befindet.

Hintergrund:
"installgskit" ruft je nach Betriebssystem Installationsbefehle auf, die Packages aus dem GSKIT im Betriebssystem installieren:

AIX:
```
/usr/sbin/installp -acgqw -d $gskitdir $gskitxdir
```
HP-UX:
```
/usr/sbin/swinstall -x enforce_scripts=false -s
        $gskitdir/$gskitxdir $gskitxdir
```
HP-UX(ia):
```
/usr/sbin/swinstall -x enforce_scripts=false -s
        $gskitdir/$gskitinstallfile  $gskitinstallfile
```
Linux Debian:
```
dpkg -i $cryptdebfile
```
Linux Redhat/Suse:
```
rpm -Uv --nodeps --ignorearch --ignoresize --nofiledigest --nodigest $cryptrpmfile
```
Solaris:
```
/usr/sbin/pkgadd -a $gskitinstallfile/admin -n -d $gskitdir $gskitinstallfile
```
Mac:
```
/usr/sbin/installer -pkg $gskitdir/$gskitinstallfile -target
```

In der Dokumentation steht, dass diese Befehle wahlweise als User "informix" oder User "root" ausgeführt werden können. Da es sich um Installationen im Betriebssystem handelt, ist unsere Empfehlung dies als User "root" durchzuführen.

TechTipp: Funktion SPACE() - Leerzeichen

Über die Funktion space() kann eine Anzahl zwischen 1 und 255 Leerzeichen ausgegeben werden.
Beispiel:

```
select  '_'||space(0)||'_',
        '_'||nvl(space(0),'')||'_',
        '_'||space(1)||'_',
        '_'||space(2)||'_',
        '_'||space(3)||'_',
        '_'||space(4)||'_',
        '_'||space(5)||'_',
        '_'||space(6)||'_',
        '_'||space(-1)||'_'
from systables
where tabid = 1;
```

Ergebnis:

```
(constant)
(constant)  __
(constant)  _ _
(constant)  _  _
(constant)  _   _
(constant)  _    _
(constant)  _     _
(constant)  _      _
(constant)
```

Gültige Werte sind Ganze Zahlen zwischen 1 bis 255 (*).
Wird eine Zahl kleiner als 1 eingegeben (0, -1,..) so wird "null" zurückgeliefert.

(*) Die obere Grenze, die in der Dokumentation angegeben ist, scheint in aktuellen Versionen weit höher zu liegen. Unsere Tests haben ergeben, dass die Funktion Werte bis "space(32755)" problemlos annimmt und die Leerzeichen ausgibt.

Praktisch ist diese Funktion, wenn in Abhängigkeit errechneter Werte eine Anzahl an Leerzeichen ausgegeben werden soll.
Das Folgende Beispiel macht dies deutlich.

```
select (year(timestamp)
        ||'-'
        ||lpad(month(timestamp),2,'0'))::char(7) as month,
        replace(space(round(sum(production)/10000,0)),' ','X')
        ||'> '
        ||space(100-round(sum(production)/10000,0))
        ||round(sum(production)/1000,2)::varchar(30) as prod
from sonne
where year(timestamp) = 2023
group by 1
into temp tmp_sun_001 with no log;
```

```
select month||':'||prod||' kWh' as sonne
from tmp_sun_001
order by 1
```

```
sonne  2023-01:XXXXXXXXXXXXXXXXXX>                                                                                       176,38 kWh
sonne  2023-02:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                                                 400,95 kWh
sonne  2023-03:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                                               417,71 kWh
sonne  2023-04:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                              590,79 kWh
sonne  2023-05:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                    685,63 kWh
sonne  2023-06:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                   859,85 kWh
sonne  2023-07:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                    689,34 kWh
sonne  2023-08:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                            605,57 kWh
sonne  2023-09:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                   697,61 kWh
sonne  2023-10:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>                                                          470,44 kWh
sonne  2023-11:XXXXXXXXXXXXXXXXX>                                                                                        167,25 kWh
sonne  2023-12:XXXXXXXXXXXXXXXXXX>                                                                                       182,68 kWh
```

TechTipp: Funktion REVERSE() - Zeichenkette umdrehen

Die Ausgabe einer Zeichenkette in umgekehrter Reihenfolge ist nun nicht das, was man im Alltag braucht.
Beispiel:

```
select "Gerd Kaluzinski" as name,
reverse("Gerd Kaluzinski") as eman
from systables where tabid = 1

name            eman
Gerd Kaluzinski iksnizulaK dreG
```

Allerdings gibt es Fälle, bei denen die umgekehrte Reihenfolge das Leben leichter macht. Ein Beispiel hierfür sind Tabellen aus BaaN, bei denen die letzten 3 Stellen die Firma beinhalten.

```
select reverse(substr(reverse(tabname),1,3)) as company,
     count(*) as anzahl_tables
from check_tabnames
where company_id = 41
and checktime = '2026-03-01 03:42:00'
and dbsname = 'baan4'
and tabname like 'tt%'
and reverse(substr(reverse(tabname),1,3)) between '000' and '999'
group by 1
order by 1

company            anzahl_tables
000                    458
103                   2505
201                   2838
202                   2280
203                   2355
242                   2617
301                   2520
302                   2774
401                   2721
```

TechTipp: Funktion NVL() - Nullvalue Handling

Mit der Funktion NVL(val1, val2) wird getestet, ob das erste Argument (val1) der Funktion den Wert "null" angenommen hat.
Ist dies nicht der Fall, so wird "val1" zurückgegeben. Ist "val1" null, so ist der Rückgabewert "val2".
Beide Argumente müssen kompatible Datentypen sein, die mittels implizitem Cast angeglichen werden können.

Für die folgenden Beispiele wurde eine Testtabelle aufgebaut:

```
create table nvl_test (
nr       int,
test     int,
valx     char(10),
alter    char(10)
);

insert into nvl_test values (1,   1,   'x1',   'Y1');
insert into nvl_test values (2,null,   'x2',   'Y2');
insert into nvl_test values (3,   3,  null,    'Y3');
insert into nvl_test values (4,null,   'x4',   null);
insert into nvl_test values (5,   5,   null,   null);
insert into nvl_test values (6,   6,'Test6','Test6');
```

```
select nr, test, valx, alter, nvl(valx,alter) as nvl_valxx
from nvl_test
order by 1;

         nr        test valx       alter  nvl_valxx
          1           1 x1         Y1         x1
          2             x2         Y2         x2
          3           3            Y3         Y3    <- valx is null, daher Y3
          4             x4                    x4
          5           5                            <- valx is null, alter auch
          6           6 Test6      Test6      Test6
```

Schreibweise als CASE Funktion:

```
select nr, test, valx, alter,
     case
          when valx is null then alter else valx
          end as nvl_valxx
from nvl_test order by 1;

         nr        test valx       alter  nvl_valxx
          1           1 x1         Y1         x1
          2             x2         Y2         x2
          3           3            Y3         Y3
          4             x4                    x4
          5           5
          6           6 Test6      Test6      Test6
```

TechTipp: Funktion NVL2() - Nullvalue Handling

Während die Funktion NVL() noch recht bekannt ist, wird die Funktion NVL2(val1,val2,val3) weit weniger oft genutzt.
Bei NVL2() erfolgt die Abfrage auf "is null" auf das Argument "val1".
Ist dieses Argument nicht "null", dann wird "val2" ausgegeben, andernfalls wird "val3" zurückgegeben.
Die Argumente val2 und val3 müssen kompatible Datentypen sein, die mittels implizitem Cast angeglichen werden können.

Beispiel:

```
select nr, test, valx, alter, nvl2(test,valx,alter) as nvl2_test
from nvl_test order by 1;

         nr        test valx       alter  nvl2_test

          1           1 x1         Y1         x1
          2             x2         Y2         Y2    <- test is null, daher alter
          3           3            Y3
          4             x4                          <- test is null, alter auch
          5           5
          6           6 Test6      Test6      Test6
```

Schreibweise als CASE Funktion:

```
select nr, test, valx, alter,
     case
          when test is not null then valx
          else alter
     end as nvl2_test
from nvl_test order by 1;

         nr        test valx       alter  nvl2_test

          1           1 x1         Y1         x1
          2             x2         Y2         Y2
          3           3            Y3
          4             x4
          5           5
          6           6 Test6      Test6      Test6
```

TechTipp: Funktion NULLIF() - Vergleich von Werten

Noch unbekannter ist wohl die Funktion NULLIF(val1,val2).
Sind die Argumente val1 und val2 gleich, so wird "null" zurückgegeben, andernfalls val1.
Beide Argumente müssen kompatible Datentypen sein, die mittels implizitem Cast angeglichen werden können.

Beispiel:

```
select nr, test, valx, alternate, nullif(valx,alternate) as nullif_test from
nvl_test order by 1;

         nr        test valx       alternate  nullif_test

          1           1 x1         Y1         x1
          2             x2         Y2         x2
          3           3            Y3
          4             x4                    x4
          5           5
          6           6 Test6      Test6
```

Schreibweise als CASE Funktion:

```
select nr, test, valx, alternate,
     case
          when valx = alternate then null
          else valx
     end as nullif_test
from nvl_test order by 1;

         nr        test valx       alternate  nullif_test

          1           1 x1         Y1         x1
          2             x2         Y2         x2
          3           3            Y3
          4             x4                    x4
          5           5
          6           6 Test6      Test6
```

TechTipp: Funktion LAST_DAY() / NEXT_DAY()

Bei INFORMIX existieren viele Funktionen mit denen ein Datum berechnet werden kann. Einige Namen sind irreführend, denn "next_day()" liefert nicht den nächsten Tag, sondern das Datum des nächsten gesuchten Wochentags.
Im Dezember 2015 hatten wir die Funktion NEXT_DAY() vorgestellt, mit der die Frage beantwortet werden kann, wann endlich wieder Freitag ist:

```
SELECT
     NEXT_DAY(TODAY, 'FRI') AS endlich_freitag_am
FROM systables WHERE tabid = 1;

endlich_freitag_am
31.07.2026
```

Eine denkbare Anwendung ist z.B. die Eintragung des Liefertermins, wenn in der Region regelmässig an einem bestimmten Wochentag die Spedition ausliefert.

Die Funktion "last_day()" gibt nicht das "Ende der Welt" (oder das Datum für Out-of-Support) an, sondern liefert das Datum des letzten Tages im Monat.
Der Erste eines Monats ist leicht anzugeben. Beim Letzten Tag fällt es schon schwerer. Im SQL könnte man hier den ersten Tag im Folgemonat nutzen und "-1 units day" abziehen, um auf dasselbe Ergebnis zu kommen.
Als Argument muss ein Argument vom Datentyp "date" oder "datetime" angegeben werden:

```
SELECT TODAY AS Heute, LAST_DAY(TODAY) AS Monatsende,
   LAST_DAY(TODAY) - TODAY AS noch_Tage_im_Monat
   FROM systables WHERE tabid = 1;

heute      monatsende  noch_tage_im_monat
18.03.2026 31.03.2026           13
```

TechTipp: Funktion QUARTER()

Die wohl wichtigste Funktion für ein amerikanisches, börsennotiertes Unternehmen ist QUARTER(). Alle Angebote, Umsätze werden an Quartalsgrenzen ausgerichtet. So verwundert es nicht, dass auch das Quartal als Funktion hinterlegt ist:

Beispiel Abfrage vom 18.03.2026:

```
select today::datetime year to day as datum, QUARTER(today) as quartal
     from systables where tabid = 1
UNION
select today +  42 units day, QUARTER(today + 42 units day)
     from systables where tabid = 1
UNION
select today + 123 units day, QUARTER(today + 123 units day)
     from systables where tabid = 1
UNION
select today + 142 units day, QUARTER(today + 142 units day)
     from systables where tabid = 1
order by 1

Ergebnis:
datum          quartal

2026-09-02           3
2026-10-14           4
2027-01-03           1
2027-01-22           1
```

TechTipp: Funktion TO_YMINTERVAL / TO_DSINTERVAL

Der Datentyp INTERVAL kann nützlich sein bei Berechnungen, die Datum und Uhrzeit beinhalten. Dieser Datentyp hält jedoch auch einige Fallstricke bereit.
Beachtet werden muss, dass der Datentyp INTERVAL entweder Jahr bis Monat, oder Tag bis Sekunde (bzw. Fraction) umfassen kann. Diese beiden Bereiche sind nicht miteinander kombinierbar.
Um die Angabe von Intervallen zu vereinfachen wurden die Funktionen TO_YMINTERVAL und TO_DSINTERVAL (und deren Synonyme NUMTOYMINTERVAL bzw. NUMTODSINTERVAL) zur Verfügung gestellt.

Hier einige Optionen, wie ein Interval eingegeben werden kann:

```
select
     TO_YMINTERVAL('01-07') as x_01_07,
     TO_YMINTERVAL(0.79,'YEAR') as x_0_79_YEAR,
     NUMTOYMINTERVAL(1,'YEAR') as x_1_YEAR
from systables
where tabid = 1
```

Ergebnis:

```
x_01_07       x_0_79_year   x_1_year
1-07          0-09          1-00
```

Die Numerischen Werte werden in Jahre und Monate im Interval aufgeteilt.
Verbliebene Tage, Stunden, Minuten werden nicht ausgegeben.

Im Gegensatz dazu wird bei einem Intervall, das mit Tag beginnt, kein Monat oder Jahr mit ausgegeben:

```
select
     TO_DSINTERVAL('42 13:23:07'   ) as x_42_13_23_07,
     TO_DSINTERVAL(410,      'HOUR')  as x_410_HOUR,
     TO_DSINTERVAL(410.5,    'HOUR')  as x_410_5_HOUR,
     TO_DSINTERVAL(24042,  'MINUTE') as x_24042_MINUTE,
     TO_DSINTERVAL(1447777,'SECOND') as x_144777_SECOND
from systables where tabid = 1
```

Das Ergebnis:

```
x_42_13_23_07      x_410_hour    x_410_5_hour  x_24042_minute   x_144777_second
42 13:23:07        17 02:00:00   17 02:30:00   16 16:42:00      16 18:09:37

select
     NUMTODSINTERVAL(409,'HOUR') as x_409_HOUR
from systables where tabid = 1

x_409_hour
17 01:00:00
```

TechTipp: ROLE / DEFAULT ROLE / CURRENT ROLE

Das Berechtigungskonzept bei Informix basiert auf Einzelberechtigungen der Benutzer (User) und Gruppenberechtigungen (Rollen).

Wird eine Tabelle neu erstellt, so besitzt der Ersteller "Owner" alle Rechte auf der Tabelle. Falls die Umgebungsvariable NODEFDAC ("No Default Distribute Access" siehe Newsletter April 2007) nicht
gesetzt ist, werden alle Rechte auch an die Gruppe "public" vergeben. Die Rechte der Gruppe "public" besitzt jeder Benutzer, der sich mit der Datenbank verbindet.

Üblicherweise wird NODEFDAC gesetzt, oder es werden nach dem Erstellen der Tabellen die Rechte von "public" entzogen, um die Basis für ein sicheres System zu schaffen.

Nun könnte den Benutzern die Rechte SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, ... auf den einzelnen Tabellen erteilt werden. Da meist viele unterschiedliche Benutzer mit der Datenbank arbeiten, wäre dies ein sehr hoher Verwaltungsaufwand.

Daher wird mit Rollen (ROLEs) gearbeitet. Eine Role stellt eine Gruppe dar, die genau wie Einzelnutzer Rechte auf Objekten bekommen kann.

Mittels "create role <role_name>" wird eine Rolle erstellt.
Beispiel:

```
create role if not exists read_user;
grant select on tab1 to read_user;
grant select on tab2 to read_user;

create role if not exists write_user;
create role if not exists 'Write_User';
grant insert,update,delete on tab1 to write_user;
grant select,insert,update,delete on tab1 to 'Write_User';
```

Hinweis:
Rollen können Case-Sensitiv angelegt werden, wenn der Name in Quotes angegeben wird.

Allein durch das Erstellen der Rollen bekommt noch kein Benutzer mehr oder weniger Rechte. Einem Benutzer muss das Recht gegeben werden, sich in eine Rolle einzufügen:

```
grant read_user to kalu;
grant 'Write_User' to kalu;
grant write_user to kalu;
```

Auch wenn ein Benutzer das Recht erhalten hat einer Gruppe beizutreten, hat er damit noch keine erweiterten Rechte. Er kann jedoch mittels "set role" die Rechte einer Rolle annehmen:

```
set role read_user;
```

Ab diesem Zeitpunkt hat der Benutzer nicht mehr die Rechte, die ihm zugeordnet wurden, sondern ausschliesslich die Rechte, die der Gruppe vergeben wurden.

Um Benutzern mit dem Connect zur Datenbank die Rechte einer Gruppe zu geben, kann eine Default Role vergeben werden, die dann für den betroffenen Benutzer automatisch vergeben wird:

```
grant default role read_user to kalu;
```

Um eine Rolle zu verlassen, ohne explizit in eine andere Rolle zu wechseln, kann zur Default Role gewechselt werden:

```
set role default;
```

Im Gegensatz zu anderen Datenbanksystemen werden Berechtigungen nicht additiv vergeben. Mit dem Wechsel in eine Rolle verliert der Benutzer alle bisherigen Rechte in der Datenbank, die er auf Grund individuell vergebener Rechte, oder aus einer vorherigen Rolle hatte. Einem Benutzer kann immer nur genau eine aktive Rolle zugeordnet sein.

Sowohl die Default Role, als auch die aktuell aktive Role können mittels SQL abgefragt werden:

```
select DEFAULT_ROLE, CURRENT_ROLE from systables where tabid = 1;

(expression)                    (expression)
read_user                       Write_User
```

TechTipp: Tabellen mit CRCOLS finden

Will man Tabellen, die keinen Primary Key und keinen Unique Index besitzen, in die CDR Replikation aufnehmen, so muss ein eindeutiger Schlüssel mit Hilfe der CRCOLS Schattenspalte erzeugt werden. Diese besteht aus CDRSERVER und CDRTIME.
Diese Spalten sind nicht in der Tabelle SYSCOLUMNS zu sehen, und werden bei einem "select *" auch nicht mit ausgegeben. Lediglich im "dbschema -ss" (-ss = system specific information) ist der Hinweis zu finden, dass eine Tabelle mit diesen Spalten angelegt wurde:

```
create table ...
  (
    ...
) with crcols ...
```

Um mittels SQL herauszufinden, ob die Schattenspalten zu CRCOLS definiert wurden, müssen die Flags der Tabelle abgefragt werden.

Codes for classifying permanent tables:

```
ROWID          1 - Has rowid column defined
UNDER          2 - Table created under a supertable
VIEWREMOTE     4 - View is based on a remote table
CDR            8 - Has CDRCOLS defined
RAW            16 - (Informix) RAW table
EXTERNAL       32- External table
AUDIT          64 - Audit table attribute - FGA
AQT            128 - View is an AQT for DWA offloading
VIRTAQT        256 - View is a virtual AQT
```

Somit suchen wir nach Flag 8:

```
select trim(tabname) as table_with_crcol
from systables
where bitand(flags,8) = 8;
```

Die Inhalte der spalten können nur bei direkter Abfrage mit ausgegeben werden:

```
select *, CDRSERVER, CDRTIME from ...
```

TechTipp: Funktion hex()

Die Funktion hex() liefert zu einem Integer bzw. Bigint Wert den zugehörigen Hexadezimal Wert ls Zeichenfolge aus zwei Zeichen [0-9A-F] je Byte.
Diese Funktion wird oft bei der Analyse von Flags innerhalb der System Tabellen genutzt. Im obigen Beispiel um CRCOLS zu finden, war diese Funktion hilfreich.
Für die weitere hierbei genutzte Funktion bitand() verweisen wir Sie auf eine der kommenden Ausgabe des Newsletters, in der wir bitand(), bitor(), bitxor(), bitandnot() und bitnot() anhand von Beispielen vorstellen wollen.

TechTipp: SPL hex2int()

Bei der Suche nach einer Funktion, mit der man über die Flags der Tabelle systables die Tabellen mit CDCOLS ermitteln kann, sind wir auf das Problem gestossen, dass es zwar eine Funktion hex() gibt, aber kein unhex() bzw. hex2int().
Da uns diese Funktion in der Vergangenheit schon mehrfach gefehlt hat, war es an der Zeit, diese zu erstellen:

```
create procedure if not exists hex2int (in_hex varchar(255))
returning bigint;

define pos, len, z int;
define x char(1);
define ret bigint;

let pos = 1;
let ret = 0;
let z = 0;

let len = length(in_hex);

while pos <= len
     let ret = ret * 16;
        let x = substr(in_hex, pos, 1);
         if x not in
('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','X','a','b','c'
,'d','e','f','x')
     then raise exception -746, 42, 'Schöne Scheisse - Keine Hexe !!!';
     end if;
     if   x = '0' then let z = 0;
     elif x = '1' then let z = 1;
     elif x = '2' then let z = 2;
     elif x = '3' then let z = 3;
     elif x = '4' then let z = 4;
     elif x = '5' then let z = 5;
     elif x = '6' then let z = 6;
     elif x = '7' then let z = 7;
```

```
     elif x = '8' then let z = 8;
     elif x = '9' then let z = 9;
     elif x = 'A' then let z = 10;
     elif x = 'B' then let z = 11;
     elif x = 'C' then let z = 12;
     elif x = 'D' then let z = 13;
     elif x = 'E' then let z = 14;
     elif x = 'F' then let z = 15;
     elif x = 'a' then let z = 10;
     elif x = 'b' then let z = 11;
     elif x = 'c' then let z = 12;
     elif x = 'd' then let z = 13;
     elif x = 'e' then let z = 14;
     elif x = 'f' then let z = 15;
     end if
        let pos = pos + 1;
     if x = 'x' then continue while; end if;
     if x = 'X' then continue while; end if;
        let ret = ret + z;
--   return ret with resume;
end while;
return ret;
end procedure;
```

Tests:

```
Function call:                              Result:
execute procedure hex2int('1');             1
execute procedure hex2int('7');             7
execute procedure hex2int('A');             10
execute procedure hex2int('F');             15
execute procedure hex2int('1A');            26
execute procedure hex2int('A1');            161
execute procedure hex2int('4008');          16392
execute procedure hex2int('0x004008');      16392
execute procedure hex2int('0xIBM');    746: Schöne Scheisse - keine Hexe !!!
```

TechTipp: SPL int2bin()

Wo wir gerade dabei sind, fehlende Funktionen zu erstellen, darf die Umwandlung von Integer Werten in die binäre Darstellung natürlich nicht fehlen:

```
create procedure if not exists int2bin(x_in int)
returning varchar(20)
define ret varchar(20);
define v_int int;
define x int;

if x_in < 0
     then raise exception -746, 42, 'Nur positive Werte erlaubt !';
end if;

let v_int = x_in;
let ret='';
while v_int >= 0
     let x = mod(v_int,2);
     let v_int = v_int/2;
     let ret=x||ret;
     if v_int = 0 then exit while; end if;
end while;
return ret;
end procedure;
```

Tests:

```
Function call:                      Result:
execute function int2bin(1);        1
execute function int2bin(7);        111
execute function int2bin(8);        1000
execute function int2bin(15);       1111
execute function int2bin(16);       10000
execute function int2bin(-42);      746: Nur positive Werte erlaubt !
```

Bei der Recherche im Internet sind wir auf eine Funktion gestossen, die auch mit negativen Werten arbeiten kann (und die Besonderheiten von INFORMIX berücksichtigt). Zur Vollständigkeit hier dieses Fundstück:

```
CREATE procedure "informix".int2bin_signed (p_val INT)
RETURNING VARCHAR(32);

    DEFINE v_bin VARCHAR(32);
    DEFINE v_rem BIGINT;
    DEFINE v_curr BIGINT;
    DEFINE i INT;

    LET v_bin = '';

    -- If negative, map to 32-bit Two's Complement unsigned equivalent
    IF p_val < 0 THEN
        -- 4294967296 is 2^32
        LET v_curr = p_val + 4294967296;
```

```
      ELSE LET v_curr = p_val;
    END IF;

    -- Explicitly handle zero
    IF v_curr = 0 THEN RETURN '00000000000000000000000000000000'; END IF;

    -- Loop exactly 32 times to populate a fixed 32-bit integer register
    FOR i = 1 TO 32
        LET v_rem = MOD(v_curr, 2);
        LET v_bin = v_rem || v_bin;
        LET v_curr = TRUNC(v_curr / 2);
    END FOR;
    RETURN v_bin;
END procedure;
```

TechTipp: SPL bin2int()

Nachdem wir nun Integer in Binary umwandeln können, fehlt uns nur noch die "Gegenrichtung":

```
create procedure if not exists bin2int(x_in varchar(128))
returning int
define ret int;
define v_val varchar(128);
define t char(1);
define x int;
define i int;

on exception
     raise exception -746, 42, 'Nur binäre Zahlen erlaubt !';
end exception;

let v_val = x_in;
let ret=0;
let i=0;
while length(v_val) > 0
     let t = substr(v_val,-1,1);
     if t not in ('0','1')
          then raise exception -746, 42, 'Nur binäre Zahlen erlaubt !';
     end if;
     if t = '1' then let ret = ret + pow(2,i); end if;
     let v_val = substr(v_val,1,length(v_val)-1);
     let i = i+1;
end while;
return ret;
end procedure;
```

Tests:

```
Function call:                        Result:
execute function bin2int('0000');     0
execute function bin2int('001');      1
execute function bin2int('0101');     5
execute function bin2int('001111');   15
execute function bin2int('Hallo');    746: Nur binäre Zahlen erlaubt !
```

TechTipp: KAIO mit Large Disk Block Size

Mit der Aktivierung von KAIO kann eine schnellere Methode aktiviert werden, um Daten von Platte in den Memory und zurück zu übertragen. Früher mussten dafür Character based Rawdevices eingerichtet werden. Seit der Einführung des Parameters DIRECT_IO kann diese Option auch im Filesystem genutzt werden.
Lange Zeit war dieses Feature jedoch auf Platten mit Blockgrössen bis zu 1024 beschränkt. Wollte man KAIO z.B. mit Platten einer Blockgrösse von 4096 aktivieren, so wurde KAIO deaktiviert und im online.log standen Meldungen der Art:

Performance Advisory: Currently IBM Informix Dynamic Server **cannot**
support KAIO on the device containing
'/DBS/rootdbs'
Its block size (4096) is larger than 1024.
Results: **Direct and concurrent I/O are disabled for this chunk**.
Action: Standard AIO (possibly buffered) will be used for I/O to this chunk.

Mit Version 14.10.FC13 wurde diese Beschränkung aufgehoben:

https://www.ibm.com/mysupport/s/defect/aCI3p000000bnIw/dt208980?language=en_US
Enhancement added to support 4K pages in Informix Server 14.10.xC13.

Allerdings gibt es eine weitere Beschränkung:
Der ROOTDBS sowie die DBSpaces für das Logical Log und Physical Log können nur in System Page Size erstellt werden.
Beim "onspaces -c -P" gibt es den Parameter der Pagesize nicht, und beim Versuch Logical Logs in einem DBSpace zu erstellen, der eine grössere Page Size hat, wird folgende Fehlermeldung zurückgegeben:

```
"log files cannot be created on dbspaces of big page"
```

Bei der Initialisierung des ROOTDBS auf einem Plattensystem mit einer Sector Size die grösser ist als die System Page Size kommt folgende Meldung:

Performance Advisory: Currently IBM Informix Dynamic Server cannot
support KAIO on the device containing
'/DBS/rootdbs'
Its Sector Size (4096) and the page size (2048) are incompatible.

TechTipp: Environments - BAR_SKIP_DUPS_CHECK

Der Aufruf "onsmsync -g <xx>" räumt alte Sicherungen auf, die älter sind als xx erfolgreiche Generationen der Sicherung. Die Sicherungen werden in der Datenbank sysutils und in der Datei $INFORMIXDIR/etc/ixbar.<servernum> protokolliert.
Brechen Sicherungen ab z.B. wegen "Disk full" oder Problemen am Storage Manager, so kann es vorkommen, dass die Informationen in der Datei ixbar.<servernum> und der Tabelle bar_action in der Datenbank sysutils nicht übereinstimmen.
Ist dies der Fall, so brechen weitere Aufrufe von onsmsync mit dem Returncode 116 ab.
Abhilfe bringt hier, die Umgebungsvariable BAR_SKIP_DUPS_CHECK auf 1 zu setzen.
Dadurch wird die Überprüfung übersprungen und onsmsync kann wieder erfolgreich aufgerufen werden.
Sind die Einträge, die das Problem verursacht haben, expired, dann sind auch Aufrufe von onsmsync ohne diesen Parameter wieder erfolgreich.

Gastbeitrag: Random ohne Random Package

Im Newsletter 2026-Q2 haben wir die Random Funktion aus dem Random Package vorgestellt. Diese bedingt, dass die Datenbank mit Logging betrieben wird.

Roland Wintgen hat uns eine Version der Random Funktion zukommen lassen, die ohne Logging und ohne das Random Package auskommt.

Vielen Dank an den Einsender:
Roland Wintgen
DBA, Genero/4GL Developer
EVG Martens GmbH & Co. KG
Moenchengladbach

Sollten auch Sie Beiträge haben, die für die Informix Gemeinschaft von Interesse sind, dann senden sie uns diese, damit wir diese als Gastbeitrag verbreiten können.

Hier der Quelltext der Procedure, die mit Fehlerbehandlung und Debugging erstellt wurde.

```
create procedure if not exists rand(_min integer default 0, _max integer default
0, _DEBUG boolean default "f")
returns float;
     define global p_seed integer default 0;
     define p_high integer;
     define p_low integer;
     define p_test integer;
```

```
     define sessionid integer;
     define sql_error integer;
     define isam_error integer;
     define error_string varchar(255);
on exception
     set sql_error, isam_error, error_string
     select dbinfo("sessionid")
          into sessionid
          from systables
          where systables.tabname = "systables";
     set debug file to "/tmp/" || sessionid || ".debug" with append;
     trace "SPL: rand (" || current year to second || ")";
     trace "User: " || user;
     trace "sql_error = " || sql_error;
     trace "isam_error = " || isam_error;
     trace "error_string = " || error_string;
     trace "_min = " || _min;
     trace "_max = " || _max;
     trace "p_seed = " || p_seed;
end exception with resume;

if _DEBUG then set debug file to "/tmp/rand.debug" with append; trace on; set
explain on; end if;

if (p_seed = 0) then call srand(0); -- initialize pseudo random number generator
end if;
let p_high = p_seed / (2147483647 / 48271);
let p_low = mod(p_seed, 2147483647 / 48271);
let p_test = 48271 * p_low - mod(2147483647, 48271) * p_high;
if (p_test > 0) then let p_seed = p_test; else let p_seed = p_test + 2147483647;
end if;
if ((_min < _max) and (_max > 0))
     then return ((((p_seed / 2147483647) * ((_max - _min) + 1)) + _min)::inte-
ger);
     else return (p_seed / 2147483647);
end if;

end procedure
;

create procedure if not exists srand(_seed integer default 0, _DEBUG boolean
default "f");
     define global p_seed integer default 0;
     define p_current integer;   -- number of seconds since 1970-01-01 00:00:00
     define p_sessionid integer; -- session id
     define p_uid integer;       -- user id
     define p_pid integer;       -- process id
     define p_tid integer;       -- thread id
     define p_uptime integer;    -- number of seconds since server instance
startup
     define p_blkused integer;   -- number of blocks of used memory
     define p_usercpu integer;   -- total number of seconds of CPU user time
     define p_isamtot integer;   -- kalu added
     define sessionid integer;
     define sql_error integer;
     define isam_error integer;
     define error_string varchar(255);
```

```
on exception
     set sql_error, isam_error, error_string
     select dbinfo("sessionid")
          into sessionid
          from systables
          where systables.tabname = "systables";
     set debug file to "/tmp/" || sessionid || ".debug" with append;
     trace "SPL: srand (" || current year to second || ")";
     trace "User: " || user;
     trace "sql_error = " || sql_error;
     trace "isam_error = " || isam_error;
     trace "error_string = " || error_string;
     trace "_seed = " || _seed;
     trace "p_seed = " || p_seed;
end exception with resume;

if _DEBUG then set debug file to "/tmp/srand.debug" with append; trace on; set
explain on; end if;

if ((_seed = 0) or (p_seed = 0))
     then let p_current = nvl(dbinfo("utc_current"), 0);
     let p_sessionid = nvl(dbinfo("sessionid"), 0);
     let p_uid = nvl((
          select sysmaster:syssessions.uid
          from sysmaster:syssessions
          where sysmaster:syssessions.sid = dbinfo("sessionid")), 0);
     let p_pid = nvl((
          select sysmaster:syssessions.pid
          from sysmaster:syssessions
          where sysmaster:syssessions.sid = dbinfo("sessionid")), 0);

     let p_tid = nvl((
          select max(sysmaster:sysrstcb.tid)
          from sysmaster:sysrstcb
          where sysmaster:sysrstcb.sid = dbinfo("sessionid")), 0);

     let p_uptime = nvl((
          select sysmaster:sysshmvals.sh_curtime -
               sysmaster:sysshmvals.sh_boottime
          from sysmaster:sysshmvals), 0);

     let p_isamtot = nvl((
          select mod(sysmaster:sysprofile.value, 2147483647)
          from sysmaster:sysprofile
          where sysmaster:sysprofile.name = "isamtot"), 0);

     let p_blkused = nvl((
          select mod(sum(sysmaster:sysseglst.seg_blkused), 2147483647)
          from sysmaster:sysseglst), 0);

     let p_usercpu = nvl((
          select mod(round(sum(sysmaster:sysvpprof.usercpu), 0), 2147483647)
          from sysmaster:sysvpprof), 0);

     if (p_current > 0) then
```

```
         if (p_sessionid > 0)
              then let p_seed = mod(p_current * p_sessionid, 2147483647);
         end if;
         if (p_uid > 0) then let p_seed = mod(mod(p_current * p_uid,
2147483647) * p_seed, 2147483647); end if;
         if (p_pid > 0) then let p_seed = mod(mod(p_current * p_pid,
2147483647) * p_seed, 2147483647); end if;
         if (p_tid > 0) then let p_seed = mod(mod(p_current * p_tid,
2147483647) * p_seed, 2147483647); end if;
         if (p_uptime > 0) then let p_seed = mod(mod(p_current * p_uptime,
2147483647) * p_seed, 2147483647); end if;
         if (p_isamtot > 0) then let p_seed = mod(mod(p_current * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_current * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_current * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_sessionid > 0) then
         if (p_uid > 0) then let p_seed = mod(mod(p_sessionid * p_uid,
2147483647) * p_seed, 2147483647); end if;
         if (p_pid > 0) then let p_seed = mod(mod(p_sessionid * p_pid,
2147483647) * p_seed, 2147483647); end if;
         if (p_tid > 0) then let p_seed = mod(mod(p_sessionid * p_tid,
2147483647) * p_seed, 2147483647); end if;
         if (p_uptime > 0) then let p_seed = mod(mod(p_sessionid * p_uptime,
2147483647) * p_seed, 2147483647); end if;
         if (p_isamtot > 0) then let p_seed = mod(mod(p_sessionid * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_sessionid * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_sessionid * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_uid > 0) then
         if (p_pid > 0) then let p_seed = mod(mod(p_uid * p_pid, 2147483647) *
p_seed, 2147483647); end if;
         if (p_tid > 0) then let p_seed = mod(mod(p_uid * p_tid, 2147483647) *
p_seed, 2147483647); end if;
         if (p_uptime > 0) then let p_seed = mod(mod(p_uid * p_uptime,
2147483647) * p_seed, 2147483647); end if;
         if (p_isamtot > 0) then let p_seed = mod(mod(p_uid * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_uid * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_uid * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_pid > 0) then
         if (p_tid > 0) then let p_seed = mod(mod(p_pid * p_tid, 2147483647) *
p_seed, 2147483647); end if;
         if (p_uptime > 0) then let p_seed = mod(mod(p_pid * p_uptime,
2147483647) * p_seed, 2147483647); end if;
```

```
         if (p_isamtot > 0) then let p_seed = mod(mod(p_pid * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_pid * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_pid * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_tid > 0) then
         if (p_uptime > 0) then let p_seed = mod(mod(p_tid * p_uptime,
2147483647) * p_seed, 2147483647); end if;
         if (p_isamtot > 0) then let p_seed = mod(mod(p_tid * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_tid * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_tid * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_uptime > 0) then
         if (p_isamtot > 0) then let p_seed = mod(mod(p_uptime * p_isamtot,
2147483647) * p_seed, 2147483647); end if;
         if (p_blkused > 0) then let p_seed = mod(mod(p_uptime * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_uptime * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if (p_isamtot > 0) then
         if (p_blkused > 0) then let p_seed = mod(mod(p_isamtot * p_blkused,
2147483647) * p_seed, 2147483647); end if;
         if (p_usercpu > 0) then let p_seed = mod(mod(p_isamtot * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;

     if ((p_blkused > 0) and (p_usercpu > 0)) then
         if (p_usercpu > 0) then let p_seed = mod(mod(p_blkused * p_usercpu,
2147483647) * p_seed, 2147483647); end if;
     end if;
else let p_seed = _seed;
end if;

end procedure
;
```

Der Aufruf für 5 Würfel als Beispiel:

```
select   rand(1,6,'f')::int as w1, rand(1,6,'f')::int as w2,
         rand(1,6,'f')::int as w3, rand(1,6,'f')::int as w4,
         rand(1,6,'f')::int as w5
from systables where tabid = 1
;
```

Ergebnis:

```
         w1          w2          w3          w4          w5
          5           1           2           4           6
```

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

Manuel Müller **Delivery Consultant**
IBM Expert Labs
manuel.mueller@de.ibm.com

Gerd Kaluzinski **Delivery Consultant**
IBM Expert Labs
gerd.kaluzinski@de.ibm.com

Gastbeitrag:
Roland Wintgen **EVG Martens GmbH & Co. KG**

Sowie eine Kooperation mit dem IIUG Insider
Gary Ben-Israel IIUG Insider Editor

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: Seerose im Teich des Redaktionsgartens (Gerd Kaluzinski)