Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

strahlender Sonnenschein im Herbst bringt uns einen besonderen Jahrgang im Wein.
Das Jahr 2011 ist aber auch für INFORMIX ein besonderer Jahrgang, da die Vorteile des Warehouse Accelerators und der Timeseries neue Kunden von INFORMIX überzeugt haben.
Diese Ausgabe des INFORMIX Newsletters wurde zum großen Teil dem Thema Client-Server gewidmet. Applikationen, die lokal am Server sehr schnell laufen, können „remote“ erheblich längere Antwortzeiten haben, wenn die Verbindung vom Client zum Server zum Engpass wird.
Aus diesem Grund zeigen wir die verschiedenen Möglichkeiten auf, wie der Durchsatz in dieser Verbindung beeinflusst werden kann.

Wie immer haben wir für Sie eine Reihe an Tipps und Tricks zusammengestellt.

Viel Spaß mit den Tipps der aktuellen Ausgabe.

Ihr TechTeam

TechTipp: SQLHOSTS

Obwohl die Datei sqlhosts, die unter $INFORMIXDIR/etc/ zu finden ist (wenn nicht $INFORMIXSQLHOSTS gesetzt wurde) sicherlich wohl bekannt ist, wollen wir diese noch einmal betrachten. Sie beinhaltet je Zeile einen Eintrag zur Kommunikation mit einem Datenbankserver.
(Einträge zu Gruppen behandeln wir im INFORMIX Newsletter November 2011)

Der prinzipielle Aufbau ist hierbei:
SERVERNAME PROTOKOLL HOST PORT OPTIONEN

Der **SERVERNAME** muss dabei mit einem, in der ONCONFIG in DBSERVERNAME oder DBSERVERALIASES eingetragenen Namen der Instanz, übereinstimmen.

Der Eintrag **PROTOKOLL** besteht aus 3 Teilen. Der erste Teil gibt an, auf welche Art vom Server verwiesen wird.
Dies können sein:

- on INFORMIX Online / IDS / Dynamic Server
- se INFORMIX Standard Engine
- dr Zugriff über das DRDA Protokoll

Danach kommt die Zugriffsart:

- soc Socket Verbindung
- tli Transport Layer Interface
- ipc Interprocess communication

Der dritte Teil definiert die Kommunikationsart:

Bei ipc können kann dies sein:

- shm Shared Memory
- str Streams
- nmp Named Pipe (nur Windows)

Bei soc und tli können dies sein:

- tcp TCP Protokoll
- ssl Secured Socket Layer

Zudem gibt es die Spezialfälle:

- onsqlmux Connection Multiplex
- onsocimc Socket Verbindung zu INFORMIX MaxConnect
- ontliimc TLI Verbindung zu INFORMIX MaxConnect

Als **HOST** kann entweder der Servername oder dessen IP-Adresse eingetragen werden. Zum Test reicht ein „ping“, ob der Server erreichbar ist.

Als **SERVICE** kann entweder der Servicename (aus /etc/services) oder direkt der Port eingetragen werden, unter dem der Datenbankserver kommunizieren soll.

Bei den **OPTIONEN** gibt es eine Reihe von Parametern, die durch Komma getrennt nacheinander aufgeführt werden. Die Reihenfolge spielt dabei keine Rolle.

Die Optionen sind:

- a Nur für INFORMIX Warehouse Accelerator (**Do not manually change !**) **(Dieser Eintrag darf NICHT manuell geändert werden !)**
- b Communication Buffer Size in Byte (siehe folgenden Artikel)
- c Gruppen Option zur Reihenfolge
- cdf Pfad für Kommunikationsdatei für ipc (Default /INFORMIXTEMP)
- csm Communication Support Module
- e Ende der Gruppe
- g Gruppen Name
- i Identifier einer Gruppe
- k Keep Alive
  - k=1 prüft die Erreichbarkeit des clinets (Default)
  - k=0 schaltet die Prüfung aus
    (sinnvoll z.B. bei langsamem Verbindungsaufbau)
- m Aktivierung von Multiplex
  (sinnvoll bei mehreren Verbindungen eines Clients zur Datenbank)
- r netrc lookup
  - r=1 aktiviert lookup in netrc (default)
  - r=0 deaktiviert lookup in netrc
- s Sicherheitseinstellungen
  - s=0 kein lookup in hosts.equiv und .rhosts
  - s=1 lookup in hosts.equiv
  - s=2 lookup in .rhosts
  - s=3 lookup in hosts.equiv und .rhosts
  - s=4 Pam Authentifizierung
  - s=6 Verbindung nur für ER/CDR und HA Verbindungen erlaubt, Client Applikationen und andere Anfragen werden abgewiesen.

TechTipp: SQLHOSTS – Wildcard im Hosts Eintrag

Statt einem Hostnamen oder einer konkreten IP-Adresse ist es möglich, in der Datei sqlhosts an 3. Stelle eine Wildcard zu setzen. Diese kann allein oder in Kombination mit Hostnamen oder der IP stehen. Sollen auch Clients diesen Eintrag nutzen, muss der Hostname oder die IP-Adresse mit angegeben werden.

Beispiel:

```
ifxibm          onsoctcp        *172.16.41.219          9088
bzw.:
ifxibm          onsoctcp        *kalu.lindau.ibm.com    9088
bzw.:
ifxibm          onsoctcp        *                       9088
```

Clients ignorieren die Wildcard und nutzen den Hostnamen bzw. die IP-Adresse, die in der Datei eingetragen ist. Der Datenbankserver weiss anhand der Wildcard, dass er Verbindungen aller gültigen IP-Adressen des Rechners annehmen darf. Damit können auch unterschiedliche IP-Adressen mit dem selben Port genutzt werden.

Mehrfache Einträge mit Wildcard zu einem Rechner sind nicht erlaubt.

TechTipp: SQLHOSTS – die Option „b=Buffer for Communication“

Die Größe der Buffer für den Datenaustausch über TCP/IP ist per Default auf 4k (4096) gesetzt. Müssen große Datenmengen zwischen Client und Server ausgetauscht werden, so können die Kommunikationsbuffer zu einem Engpass werden. Gerade wenn eine SQL-Abfrage, lokal auf dem Server ausgeführt, deutlich schneller ist als wenn sie „remote“ von einem Client aus abgesetzt wird, deutet dies darauf hin dass ein Tuning beim Datentransfer Abhilfe schaffen könnte.

Der Parameter „b=<buffersize_in_Byte>“ bietet die Möglichkeit, diesen Engpass zu beseitigen. Der Wert kann dabei auf maximal 64k erhöht werden.
Der Parameter sollte auf dem Client und dem Server identisch eingestellt werden.

Diese Einstellung gilt für alle Sessions, die diese Verbindung nutzen. Auf Client-Seite kann mittels der Umgebungsvariablen IFX_NETBUF_SIZE der Wert je Session eingestellt werden (siehe TechTipp unten).

Achtung:
Der für die Kommunikation belegte Speicher wird erst mit dem Disconnect der Session wieder freigegeben. Dies kann bei vielen Sessions und großem Datenaustausch zu erheblicher Speicherauslastung führen. Daher sollte der Parameter nur bei Bedarf und dann schrittweise erhöht werden.

TechTipp: ONCONFIG - DBSERVERNAME mit Option

Bis zur Version 11.70 konnte als Wert für den Parameter DBSERVERNAME nur der Name für den Datenbankserver eingetragen werden. Hierbei waren bis zu 128 Zeichen erlaubt, wobei die Auswahl der Zeichen auf Kleinbuchstaben, Zahlen und das Underscore beschränkt war. Zudem durften Underscore und Zahlen nicht als erstes Zeichen verwendet werden. Beim Parameter DBSERVERALIASES galten die selben Regeln, allerdings waren hier mehrere durch Komma getrennte Einträge möglich, wobei die Gesamtlänge des Eintrags 512 Zeichen nicht überschreiten durfte.

Bei den Parametern DBSERVERNAME und DBSERVERALIASES kann nunmehr die Anzahl der zu startenden Listener Threads für die Verbindung angegeben werden. Diese Option ist nur beim Socket Protokoll unterstützt. Andere Protokolle ignorieren den Parameter und nutzen den Defaultwert 1.

Beispiel:

```
DBSERVERNAME test42-3
DBSERVERALIASES test42a-2
```

Im „onstat -g ntt“ ist zu sehen, dass durch diese Einstellung statt nur je einem Listenerthread nun mehrere gestartet werden.

```
Individual thread network information (times):
  netscb thread name    sid     open     read    write address
577ebdf0 soctcplst       13 12:13:28
577e5df0 soctcplst       12 12:13:28                 172.16.41.229|9043|soctcp
577dddf0 soctcplst       10 12:13:28
577cddb8 soctcplst        7 12:13:28
577c9d40 soctcplst        6 12:13:28                 172.16.41.229|9042|soctcp
```

TechTipp: ONCONFIG - DBSERVERALIASES

Sind mehrere Netzwerkkarten für die Kommunikation zum Datenbankserver verfügbar, so können unterschiedliche Nutzergruppen auf die Netzwerkkarten verteilt werden, indem diesen jeweils ein anderer Name als $INFORMIXSERVER zugeteilt wird. Diese werden in der ONCONFIG als DBSERVERALIASES eingetragen. In der Datei sqlhosts können die unterschiedlichen Aliasnamen dann mit unterschiedlichen Hostnamen bzw. IP-Adressen und unterschiedlichen Ports eingetragen werden.
Zur Ausfallsicherheit können alternative Verbindungen auch zu Gruppen zusammengefasst werden, was in der Ausgabe November dieses Newsletters näher behandelt wird.

Bei Verwendung unterschiedlicher Aliasnamen zum selben Server ist es zudem möglich, diese mit unterschiedlichen Einstellungen bezüglich Sicherheit und Buffersize auszustatten. Dies geschieht in der jeweiligen Zeile der Datei sqlhosts mittels der Optionen in der 5. Spalte.

TechTipp: ONCONFIG - NETTYPE

Der Parameter NETTYPE bestimmt die Anzahl der Pollthreads, die in einer Instanz laufen. Zum NETTYPE gehören 4 Werte. Der erste Wert gibt an, zu welchem Protokoll die Einstellung erfolgen soll (soctcp,ipcshm,...).
Der 2. Parameter gibt die Anzahl der Pollthreads an, die für dieses Protokoll gestartet werden sollen.
Der 3. Parameter bestimmt, für wie viele Sessions je Pollthread Pages im Buffer bereitgestellt werden sollen.
Der 4. Parameter bestimmt, ob ein eigener VP je Pollthread gestartet wird (NET) oder ob die Pollthreads auf den CPU-VPs laufen sollen (CPU).
Da die CPU-VPs meist durch andere Aufgaben innerhalb der Instanz stark unter Last stehen, wird mittels der Einstellung NET dafür gesorgt, dass zusätzliche Prozesse die Aufgabe des Polling übernehmen. Übersteigt die Einstellung der Pollthreads zu CPU die Anzahl der konfigurierten CPU-VPs, so startet die Instanz selbsttätig zusätzliche Vps für die überzähligen Pollthreads.

Ein Pollthread kann durchaus bis zu 1000 Verbindungen bedienen. Um die Performance zu verbessern, sollte der Wert jedoch besser so eingestellt werden, dass nicht mehr als ca. 300 bis 400 Verbindungen je Pollthread aktiv sind.

Beispiel:

```
NETTYPE soctcp,4,50,NET

onstat -g ntt:
Individual thread network information (times):
  netscb thread name    sid     open     read    write address
577c7c08 soctcppoll       5 12:13:28
577c3ad0 soctcppoll       4 12:13:28
577bfad0 soctcppoll       3 12:13:28
577c1d98 soctcppoll       2 12:13:28
```

TechTipp: Optionen des ONSTAT (onstat -g ntu / onstat -g ntm)

Um herauszufinden, ob der Umfang der zur Verfügung stehenden Buffer für die Client-Server Kommunikation ausreicht, können onstat-Aufrufe genutzt werden:

- onstat -g ntu Network Userthread Profile Information
- onstat -g ntm Network Message Information

Zeigt der „onstat -g ntu“ im ersten Wert von „q-pvt“ (aktuelle Anzahl freier Kommunikationsbuffer) oft den Wert 0 an, so besteht Handlungsbedarf.
Ebenso ist ein hoher Wert von „q-exceeds“ in der Ausgabe von „onstat -g ntm“ ein deutliches Zeichen, dass die Anzahl oder Größe der zur Verfügung stehenden Buffer für die Kommunikation nicht ausreichend ist. Dieser Wert gibt an, wie oft die vorgehaltenen Buffer für die Kommunikation nicht ausgereicht haben.
Neben dem oben aufgeführten Option „b“ in der Datei sqlhosts gibt es über die Umgebungsvariablen „IFX_NETBUF_SIZE“ und „IFX_NETBUF_PVTPOOL_SIZE“ die Möglichkeit, diesen Engpass zu beseitigen. Mehr dazu in den folgenden Artikeln.

TechTipp: Environments - IFX_NETBUF_SIZE

Die Umgebungsvariable IFX_NETBUF_SIZE bietet die Möglichkeit, die Größe der einzelnen Kommunikationsbuffer je Session optimal einzustellen.
Er bestimmt die Größe der globalen und privaten (*) Kommunikationsbuffer je Session. Der Default liegt bei 4 KB (4,096 Byte). Gültige Werte liegen zwischen 512 Byte und 64 KB (65,536 bytes). Wird die Umgebungsvariable vor dem Start der Instanz gesetzt, so wirkt diese auf alle Sessions.
(* siehe nächsten Artikel)

TechTipp: Environments - IFX_NETBUF_PVTPOOL_SIZE

Neben dem globalen Bufferpool für die Kommunikation zwischen Client und Server kann zusätzlich ein privater Bufferpool für diesen Zweck eingerichtet werden. Dieser kann sowohl für TCP-Verbindungen über Sockets oder TLI, als auch für die lokale Verbindung über Streams genutzt werden.

Der Vorteil eines privaten Bufferpools liegt darin, dass dieser weniger CPU Ressourcen für die Zuteilung und Rückgabe der Kommunikationsbuffer benötigt, und dass der globale Bufferpool für die Kommunikation nicht mehr so stark frequentiert wird, was zur Vermeidung von Wartezuständen dient.

Die Umgebungsvariable **IFX_NETBUF_PVTPOOL_SIZE** bstimmt die Größe des privaten Bufferpools für die Kommunikation.
Der Wert wird in „Anzahl Kommunikationsbuffers“ angegeben, deren Größe vom Parameter IFX_NETBUF_SIZE bestimmt wird. Der Default Wert ist 1. Das Maximum für den für eine Session vorgehaltenen Speicher für die Kommunikation sind 10 Buffers.
Der Parameter IFX_NETBUF_PVTPOOL_SIZE steht nur unter Unix/Linux zur Verfügung.

TechTipp: SPL: SYSDBOPEN() – Flexible Beeinflussung von Sessions

Seit Informix Version 11.10 werden mit der Verbindung zur Datenbank, bzw. dem Abmelden einer Session von der Datenbank, die Prozeduren sysdbopen() und sysdbclose() ausgeführt, falls diese vorhanden sind. Es gibt die Möglichkeit, individuelle Prozeduren je Benutzer zu erstellen, indem der Benutzer als Owner der Prozedur angegeben wird. Besteht keine individuelle Prozedur sysdbopen()/sysdbclose() für einen Benutzer, so wird die allgemeine Prozedur mit dem Owner „public“ ausgeführt.

Um nicht bei jeder Änderung die Prozeduren neu erstellen zu müssen, wird in diesem Artikel eine Variante mit Einbeziehung einer Steuertabelle vorgestellt.

Das hier vorgestellte Beispiel soll als „Draft“ dienen und Sie dazu ermuntern eigene, auf den individuellen Bedarf angepasste Prozeduren zu erstellen.

In unserem Beispiel beinhaltet die Steuertabelle einen Eintrag je Benutzerkennung („who“), Spalten die den sqexplain aktivieren („expln“) und die Lock-Wait-Time steuern („wait_sec“), sowie eine Spalte für individuelle SQL-Kommandos („cmd_txt“).

Beispiel der Steuerungstabelle:

```
create table sysdbopentab (
who         varchar(32),
expln       char(1),
wait_sec    int,
cmd_txt     varchar(128)
);
-- Make sure to have only one instruction for each user
alter table sysdbopentab add constraint primary key (who)
      constraint sysdbopentab_p;
-- All users must be able to read this table
grant select on sysdbopentab to public;
```

Um die Prozedur zu testen, fügen wir für unsere Testuser Werte ein:
Die SQL Statements des Users mit dem Login "kalu" sollen mit sqexplain Protokolliert werden. Zudem soll der Lockmode auf den Wert „Wait 42“ gesetzt werden:

```
insert into sysdbopentab values ("kalu","1",42,'');
```

User "carmen" soll mit lockmode wait 23 Sekunden arbeiten:

```
insert into sysdbopentab values ("carmen","0",23,'');
```

User "marion" soll mit der Isolation „committed read last committed“ arbeiten:

```
insert into sysdbopentab values
      ("marion",null,null,'set isolation to committed read last committed');
```

Bei jedem Connect und Disconnect soll zudem noch erfasst werden, wann sich ein User zur Datenbank verbindet und wie viele Verbindungen dieser User maximal gleichzeitig offen hatte. Dazu wird eine weitere Tabelle benötigt, die diese Werte aufnehmen kann:

```
create table if not exists sysdbopenuser (
who             varchar(32),
last_conn       datetime year to second,
last_disconn    datetime year to second,
db_name         varchar(128),
num_conn        int,
num_disconn     int,
max_conn        int
);
-- Make sure to have only one counter for each user
alter table sysdbopenuser add constraint primary key (who)
      constraint sysdbopenuser_p;
-- All users must be able to read, insert and update this table
grant select,insert,update on sysdbopenuser to public;
```

Nun fehlen noch die eigentlichen Prozeduren, die beim Connect bzw. Disconnect implizit aufgerufen werden. Die Applikationen merken von diesem Aufruf nichts, so dass dieses Vorgehen unabhängig von der Art und Version des Frontends möglich ist.

Erstellen der Procedure sysdbopen() für „public“:

```
CREATE PROCEDURE public.sysdbopen()
DEFINE stmt1            varchar(128);
DEFINE stmt2            varchar(128);
DEFINE expln_file       varchar(128);
DEFINE cmd_txt_x        varchar(128);
DEFINE my_database      varchar(128);
DEFINE sess_id          int;
DEFINE who_x            varchar(32);
DEFINE expln_x          char(1);
DEFINE wait_sec_x       int;
DEFINE flag             int;
DEFINE num_conn_x       int;
DEFINE max_conn_x       int;
DEFINE act_conn_x       int;

ON EXCEPTION
   --if we insert statements that are invalid,
   --DO NOT PREVENT USERS FROM WORKING !!!
   --keep on working in case of an error within this procedure.
END EXCEPTION WITH RESUME;

--All sqexplain output should be collected in "/tmp" as username.session_id
SELECT "/tmp/"||USER||"."||DBINFO( 'sessionid' )
        INTO expln_file
FROM systables WHERE tabid = 1;

--See, if the user has already an entry in sysdbopentab:
SELECT who, expln, wait_sec, cmd_txt
   INTO who_x, expln_x, wait_sec_x, cmd_txt_x
FROM sysdbopentab
WHERE who = USER;

IF (who_x is not null)
THEN
   IF (expln_x = "1") THEN
        -- activate sqexplain for that user
                LET stmt1 = 'set explain file to "'||expln_file||'";';
                LET stmt2 = 'set explain on;' ;
                EXECUTE IMMEDIATE stmt1;
                EXECUTE IMMEDIATE stmt2;
   END IF;
   IF (wait_sec_x is not null) THEN
   -- set lock wait time for that user
                LET stmt1 = 'set lock mode to wait '||wait_sec_x||';';
                EXECUTE IMMEDIATE stmt1;
   END IF;
   IF (cmd_txt_x is not null AND cmd_txt_x != '') THEN
                -- execute further SQL statements
                LET stmt1 = cmd_txt_x||';';
                EXECUTE IMMEDIATE stmt1;
   END IF;
END IF;
```

```
-- Now let's update the user's last connection time
-- or insert the user at the first connection

-- first determine the current database--
SELECT odb_dbname INTO my_database
FROM sysmaster:sysopendb
WHERE odb_sessionid = DBINFO( 'sessionid' )
AND odb_iscurrent = 'Y';

-- Now let's see if we already know this user
SELECT count(*) INTO flag
FROM sysdbopenuser
WHERE who = USER
AND db_name = my_database;

IF flag = 0                     -- new entry
THEN BEGIN
        INSERT INTO sysdbopenuser
        VALUES (USER,CURRENT year to second,null,my_database,1,0,1);
END
ELSE BEGIN
        SELECT count(*) INTO act_conn_x
        FROM sysmaster:syssessions s, sysmaster:sysopendb o
        WHERE s.username = USER
        AND o.odb_dbname = my_database
        AND o.odb_sessionid = s.sid;

        SELECT num_conn + 1, max_conn   -- increase connections by one
                INTO num_conn_x,max_conn_x
        FROM sysdbopenuser
        WHERE who = USER;
        IF (act_conn_x > max_conn_x)            -- new maximum connections
                THEN let max_conn_x = act_conn_x;
        END IF;
        UPDATE sysdbopenuser
                SET (last_conn, num_conn, max_conn) =
                (CURRENT year to second,num_conn_x,max_conn_x)
        WHERE who = USER
        AND db_name = my_database;
END
END IF

END PROCEDURE;
```

Sobald die Procedure erstellt ist, wird diese von allen neuen Verbindungen zur Datenbank aufgerufen. Ein Neustart der Instanz ist dafür nicht notwendig.

User, die für die eine eigene Procedure sysdbopen() erstellt wurde, durchlaufen die allgemeine Procedure public.sysdbopen() nicht.

Erstellen der Procedure sysdbclose() für „public“:

```
CREATE procedure public.sysdbclose()
DEFINE cmd_txt_x        varchar(128);
DEFINE my_database      varchar(128);
DEFINE sess_id          int;
DEFINE who_x            varchar(32);
DEFINE act_conn_x       int;
DEFINE max_conn_x       int;
DEFINE num_disconn_x    int;
DEFINE flag             int;

ON EXCEPTION
        --IF we insert statements that are invalid,
        --DO NOT PREVENT USERS FROM WORKING !!!
        --keep on working in case of an error within this procedure.
END EXCEPTION with resume;

--only SET this tracing to test changes of this procedure - NOT for normal work
--SET debug file to "/tmp/sysdbclose.out";
--trace on;

-- Now let's UPDATE the user's disconnection time
-- determine my database
SELECT odb_dbname into my_database
        FROM sysmaster:sysopendb
        WHERE odb_sessionid = DBINFO( 'sessionid' )
        AND odb_iscurrent = 'Y';
-- see if the user hal already an entry in sysdbopenuser
SELECT count(*) into flag
        FROM sysdbopenuser
        WHERE who = USER
        AND db_name = my_database;
IF flag = 0
THEN BEGIN
        insert into sysdbopenuser
        values (USER,null,CURRENT year to second,my_database,1,0,1);
END
ELSE BEGIN
        -- how many sessions are active ?
        SELECT count(*)
                into act_conn_x
        FROM sysmaster:syssessions s, sysmaster:sysopendb o
        WHERE s.username = USER
        AND o.odb_dbname = my_database
        AND o.odb_sessionid = s.sid;
        -- increase the value of disconnects by one
        SELECT num_disconn + 1, max_conn
                into num_disconn_x,max_conn_x
        FROM sysdbopenuser
        WHERE who = USER;
        -- record id number of actual connections exceeds maximum recorded
        IF (act_conn_x > max_conn_x)
                THEN let max_conn_x = act_conn_x;
        END IF;
```

```
        -- update the values in sysdbopenuser
        UPDATE sysdbopenuser
                SET (last_disconn, num_disconn, max_conn) =
                (CURRENT year to second,num_disconn_x,max_conn_x)
        WHERE who = USER
        AND db_name = my_database;
END
END IF
END procedure;
```

Verbinden sich nun die User, zu denen ein Eintrag in der Tabelle sysdbopentab existiert, mit der Datenbank, so werden ihre Sessions beeinflusst, was mit dem „onstat -g sql“ zu sehen ist:

```
IBM Informix Dynamic Server Version 11.70.UC3 -- On-Line -- Up 01:42:23 ...
Sess   SQL          Current     Iso Lock       SQL   ISAM F.E.
Id     Stmt type    Database    Lvl Mode       ERR   ERR   Vers  Explain
117    -            stores      LC  Not Wait   0     0     9.24  Off     ← marion
116    -            stores      CR  Wait 23    0     0     9.24  Off     ← carmen
115    -            stores      CR  Wait 42    0     0     9.24  On      ← kalu
```

Zudem werden die Sessions in der Tabelle sysdbopenuser protokolliert:

```
select * from sysdbopenuser
who             kalu                    ← username
last_conn       2011-08-11 10:18:20     ← latest connect time
last_disconn    2011-08-10 11:26:08     ← latest disconnect time
db_name         stores                  ← current database
num_conn        12                      ← number of connects
num_disconn     10                      ← numer of disconnects
max_conn        2                       ← max. parallel connections

who             carmen
last_conn       2011-08-11 10:40:25
last_disconn    2011-08-09 15:05:29
db_name         stores
num_conn        5
num_disconn     2
max_conn        3

who             marion
last_conn       2011-08-11 10:19:04
last_disconn
db_name         stores
num_conn        1
num_disconn     0
max_conn        1
```

Den ausführlichen Artikel mit zusätzlichen „Reset-Funktionen“ für die Tabelle sysdbopenuser und allen **SQL-Statements aus diesem Beispiel zum Download finden bei IBM Developerworks** unter:
http://www.ibm.com/developerworks/data/library/techarticle/dm-1109sysdbopen/index.html

Referenzen: Böckmann Fahrzeugwerke GmbH

Das Unternehmen Böckmann Fahrzeugwerke GmbH setzt seit vielen Jahren auf INFORMIX. Mit dem Umstieg von INFORMIX 4GL auf **INFORMIX Genero** in nur 2 Wochen, konnten die Applikationen schnell auf ein aktuelles Design und neue Funktionalitäten gebracht werden. Die Kombination aus **INFORMIX Version 11.50** und **INFORMIX Genero** bietet höchste Zuverlässigkeit, beste Performance und die Möglichkeit, die Applikationen schnell veränderten Anforderungen anzupassen.

Den ausführlichen Artikel lesen Sie unter:
http://www.ibm.com/software/success/cssdb.nsf/CS/JHUN-8L95TZ?OpenDocument&Site=corp&cty=en_us

Wollen Sie auch als INFORMIX Referenz vorgestellt werden ?
Dann wenden Sie sich an unsere Redaktion, damit wir die Details abklären können:
ifxmnews@de.ibm.com
Oft ist nicht bekannt, dass hinter vielen Produkten und Dienstleistungen, die uns im Alltag begegnen, INFORMIX steckt. Daher wollen wir mit dieser Serie zeigen, dass der frühere Werbespruch „INFORMIX is everywhere“ auch auf Ihren Alltag zutrifft.

Termine: IX221 - Informix Dynamic Server 11 Database Administration

Vom 7.11.2011 bis 10.11.2011 ist der Kurs
IX221 - Informix Dynamic Server 11 Database Administration
- Managing and Optimizing Data -
in Wien geplant.

Der Kursinhalt betrachtet jeweils auch die aktuellen Änderungen und Erweiterungen in den Versionen 11.50 und 11.70 anhand von Beispielen.
Neben dem Kursinhalt, der ausführlich auf der unten aufgeführten Seite zu finden ist, haben Sie an einem Abend die Möglichkeit in einer "Fragestunde" weitere Themen rund um INFORMIX anzusprechen, die Ihnen im täglichen Betrieb begegnet sind.

Nähere Informationen zum Kurs und zur Anmeldung erhalten Sie unter:
http://www.wifiwien.at/eShop/bbDetails.aspx/Informix-Dynamic-Server-11-Database--Administration--Managing-and-Optimizing--Data---IBM-IX221DAT/@/bbnr/365201/zg/E9b1/

Der Kurs steht nicht nur den Kunden aus Österreich offen, so dass gerade die Kunden und Partner, die grenznah zu Österreich beheimatet sind, sich den Termin vormerken sollten. Wien ist immer eine Reise Wert !

Termine: 58. IUG-Workshop in Hamburg

Der 58. Workshop der Informix User Group findet am am 18.10.2011 in Hamburg statt.
Das Thema des Workshops ist diesmal:

„Applikationsentwicklung, Informix als Backend von Business-Lösungen“

Wie gewohnt findet am Vorabend ein Stammtisch statt, an dem sich die Gelegenheit bietet, in gemütlicher Atmosphäre Kontakt zu knüpfen und Erfahrungen auszutauschen. Sollten sie noch kein Mitglied der IUG sein, dann wenden Sie sich an info@iug.de. Dort erfahren Sie mehr über die Vorteile zu dieser Gemeinschaft zu gehören.

Der Stammtisch findet im
ARCOTEL Rubin Hamburg
Steindamm 63
20099 Hamburg
statt. Beginn ist 19:00 Uhr.

Der Workshop ist dann im Gebäude der
IBM Deutschland GmbH
Geschäftstelle Hamburg
Beim Strohhause 17
20097 Hamburg
Pünktlicher Beginn um 08:59 Uhr ist durch Herrn Dr. Aspiazu garantiert.

Weitere Informationen zum Workshop finden Sie im Web unter:
http://www.iug.de/index.php?option=com_content&task=view&id=207&Itemid=279

Suchen Sie nach einem Workshop in Ihrer Region ?
Kein Problem: Die IUG verteilt die drei Workshops, die jedes Jahr durchgeführt werden so, dass Norden, Mitte und Süden immer eine Veranstaltung in ihrer Nähe haben. Allerdings kann sich bei interessanten Themen auch eine Anreise quer durch die Republik lohnen, da fast immer so aktuelle Neuigkeiten zu erfahren sind, die es noch nirgends nachzulesen gibt.

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.

Neu hinzugekommen ist ein Archiv der INFORMIX Newsletters bei der International Informix User Group.

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/Informix_Newsletter/
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

Sowie unterstützende Teams im Hintergrund.

Fotonachweis: Gerd Kaluzinski (Redaktionsgarten)