Willkommen zum Informix Newsletter

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

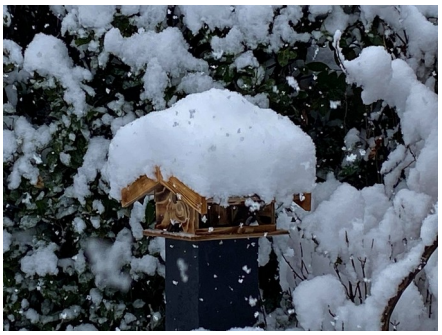
das Jahr 2021 scheint so zu beginnen wie das alte Jahr aufgehört hat. Vieles hat sich durch Corona verändert, teils zum Schlechten, teils aber auch zum Guten.

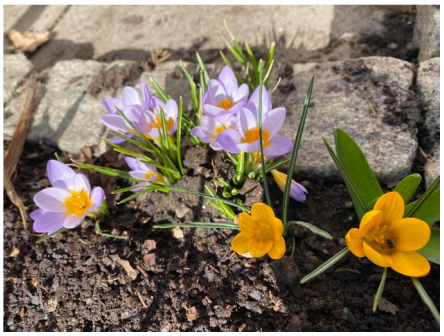
Als positiv eingestellter Mensch konzentriere ich mich auf die positiven Veränderungen und geniesse es, die Abende gemütlich zuhause zu verbringen, statt allein in einem Hotelzimmer zu sitzen. Keine langen Reisezeiten mehr, keine Wartezeiten am Flughafen, … dafür mehr Zeit für die schönen Dinge im Leben und natürlich auch ein paar Stunden mehr für den INFORMIX Newsletter.

Ein Blick in den Redaktionsgarten zeigt, dass es Frühling wird.

Bleiben Sie gesund !

Ihr TechTeam


Januar 2021


Februar 2021

TechTipp: ONCONFIG - NS_CACHE

In früheren Versionen von INFORMIX (7.x, 9,x) wurden die Werte der Name Services bei jedem neuen Connect oder jedem Remote Zugriff erneut vom Betriebssystem abgefragt. Um unnötige Aufwände für diese Abfragen auf die Dateien des Betriebssystems mittels open(),read(),close(), oder nocht schlimmer eine DNS/LDAP Abfrage über das Netzwerk zu vermeiden, wurde ein Name Service Cache eingeführt, der die gelesenen Werte für eine einstellbare Zeitdauer (in Sekunden) speichert. Der Default liegt bei 900 Sekunden (15 Minuten).
Zulässig sind ganzzahlige Werte zwischen 1 und MAXINT.
Mit dem Wert 0 kann der Cache deaktiviert werden, so dass die Informationen bei jedem Zugriff erneut im Betriebssystem abgefragt werden.

Neu in der Version 14.10 ist, dass nun auch die Datei $INFORMIXSQLHOSTS in diesem Cache mit aufgenommen wurde, der durch „sqlhosts=xxx“ angegeben wird.

Der Default für NS_CACHE in der onconfig.std ist:

NS_CACHE host=900,service=900,user=900,group=900,sqlhosts=900

Die Default Einstellung kann dazu führen, dass Änderungen im System nicht sofort, sondern erst nach bis zu 15 Minuten für die Datenbank bekannt sind.
Einen speziellen Befehl um den Cache zu leeren, gibt es nicht. Allerdings wird bei jeder Änderung der Konfiguration von NS_CACHE der bisherige Cache verworfen und neu erstellt.
Um z.B. Änderungen bei den Benutzern sofort auch in der Datenbank bekannt zu machen, kann nach dem „useradd/groupadd“ der Befehl

„onmode -wf NS_CACHE=‘…‘ “

ausgeführt werden, damit die Änderungen sofort in den Cache gelesen werden.

Beispiel für die Aufrufe zum Leeren des User- und Group-Cache:

```
onmode -wf NS_CACHE='host=900,service=900,user=0,group=0,sqlhosts=900'
```

… und dann Wiederherstellung der bisherigen Werte:

```
onmode -wf
NS_CACHE='host=900,service=900,user=900,group=900,sqlhosts=900'
```

Alle Änderungen werden im online.log protokolliert.

Anmerkung: Laut Dokumentation sucht die aktuelle Version einen Benutzer zuerst im Cache, wird dieser nicht gefunden, so wird er im Betriebssystem abgefragt und ggf. in den Cache übernommen.

TechTipp: ONCONFIG - PARTITION_NAMES

Mit Version 14.10 wurde der Parameter PARTITION_NAMES eingeführt, der viele Ausgaben des „onstat“ deutlich benutzerfreundlicher darstellt.
Zusätzlich zur Partition Nummer wird nun auch der Name des Objektes ausgegeben.

Der Parameter kann die Werte 0 und 1 (default) annehmen und ist damit bereits mit der Installation aktiv.

Die folgenden Beispiele zeigen die Vorteile dieser Erweiterung:

Anzeige der Sperren in der Instanz:

```
onstat -k

IBM Informix Dynamic Server Version 14.10.FC4W1DE -- On-Line -- Up 20:53:31 -- 181252 Kbytes
Locks
address          wtlist     owner      lklist           type     tblsnum  rowid    key#/bsiz DML
44199028         0          455948c8   0                HDR+S    100002   204         0
441992d0         0          45595a88   0                    S    100002   204         0
44199358         0          45595a88   441992d0         HDR+S    100002   201         0
441993e0         0          45591388   0                HDR+S    100002   304         0
441994f0         0          45591388   441993e0         HDR+IS   500257   0           0
442e50a8         0          455951a8   0                    S    100002   204         0
442e5350         0          45593fe8   0                    S    100002   204         0
442e53d8         0          45597528   0                    S    100002   204         0
442e7110         0          45591388   442e7c38         HDR+X    5002ca   7c405       0      U
442e7c38         0          45591388   441994f0         HDR+IX   5002ca   0           0
 10 active, 20000 total, 16384 hash buckets, 0 lock table overflows
```

Im Gegensatz zur Abfrage auf der Tabelle sysmaster:syslocks werden hier nur die Partition Nummern der Tabellen angezeigt, statt der sprechenden Namen.

Aktivieren der Partition Names:

```
onmode -wf PARTITION_NAMES=1
Value for PARTITION_NAMES (1) was saved in config file.
Value of PARTITION_NAMES has been changed to 1.
onstat -k

IBM Informix Dynamic Server Version 14.10.FC4W1DE -- On-Line -- Up 20:53:10 -- 181252 Kbytes
Locks
address          wtlist     owner      lklist           type     tblsnum  rowid    key#/bsiz DML
table_name
44199028         0          455948c8   0                HDR+S    100002   204         0
sysmaster:informix.sysdatabases
441992d0         0          45595a88   0                    S    100002   204         0
sysmaster:informix.sysdatabases
44199358         0          45595a88   441992d0         HDR+S    100002   201         0
sysmaster:informix.sysdatabases
441993e0         0          45591388   0                HDR+S    100002   304         0
sysmaster:informix.sysdatabases
441994f0         0          45591388   441993e0         HDR+IS   500257   0           0
ibm:informix.sysdistrib
442e50a8         0          455951a8   0                    S    100002   204         0
sysmaster:informix.sysdatabases
442e5350         0          45593fe8   0                    S    100002   204         0
sysmaster:informix.sysdatabases
442e53d8         0          45597528   0                    S    100002   204         0
sysmaster:informix.sysdatabases
442e7110         0          45591388   442e7c38         HDR+X    5002ca   7c405       0      U
ibm:informix.cal
442e7c38         0          45591388   441994f0         HDR+IX   5002ca   0           0
ibm:informix.cal
 10 active, 20000 total, 16384 hash buckets, 0 lock table overflows
```

Mit der Ausgabe der PARTITION_NAMES ist zu sehen, dass der Update auf der Tabelle „cal“ in der Datenbank „ibm“ stattfindet.

TechTipp: „cdr migrate server“ mit neuer Version „add_replcheck“

Wer bei INFORMIX die Enterprise Replikation (CDR) nutzt, der kennt das nützliche Feature des „Check und Repair“, bei dem die Daten auf Quelle und Ziel verglichen und ggf. angepasst werden können.
Da für den Vergleich der Datensätze ein Hashwert berechnet wird, auf dessen Basis der Vergleich abläuft, ist es sinnvoll diesen Hashwert als versteckte Spalte (Shaddow Column) in der Tabelle abzulegen.
Diese versteckte Spalte mit dem Namen „ifx_replcheck“ vom Typ BigINT kann direkt beim Erstellen der Tabelle hinzugefügt werden mittels „create table … WITH REPLCHECK“, oder nachträglich hinzugefügt werden mittels „alter table … ADD REPLCHECK“.

Die Funktion „cdr migrate server“ wurde nun ebenfalls um diese Option erweitert, indem die zusätzliche Phase „add_replcheck“ zur Verfügung steht.
Hierdurch wird in den Migrations Scripts je Tabelle der entsprechende Befehl zum „alter table“ generiert und ggf. ausgeführt.

Die versteckte Spalte „ifx_replcheck“ führt dazu, dass der Vergleich der Daten in Quelle und Ziel bei einer Replikation bedeutend schneller erfolgen kann.

TechTipp: Optionen des ONSTAT (onstat -r <interval>)

Der Befehl „onstat“ bietet eine Vielzahl an Optionen, sich Informationen aus der Informix Instanz anzeigen zu lassen.
Um die Ausgabe automatisch zu wiederholen, kann die Option „-r“ angefügt werden, die im Default alle 5 Sekunden die Ausgabe erneuert.
Dieser Default von 5 Sekunden konnte bereits bisher auf das notwendige Intervall angepasst werden, indem dem Schalter „-r“ ein Ganzzahliger Wert grösser 0 angefügt wurde. Dies wurde als Anzahl der Sekunden für die Wiederholung interpretiert.

Um auch kürzere Wiederholfrequenzen als eine Sekunde zu ermöglichen, wurde mit Version 14.10 der Wert als Dezimalzahl definiert.

Nun können die Ausgaben mit feinerer Granularität ausgegeben werden um z.B. viel mehr SQL Statements eines Benutzers zu protokollieren:

```
onstat –g sql 42 –r 0.02 > sql_42.out
```

So sind z.B. bei einer Wiederholrate von 0.02 ein „create database“ 48 unterschiedliche interne Aufrufe zu sehen.

TechTipp: Optionen des ONSTAT (Datum in der Ausgabe)

Wer die Version 14.10.xC5 oder höher installiert hat, der wird sich ggf. wundern, dass die Ausgaben des „onstat“ eine zusätzliche Zeile besitzen.
Hierbei handelt es sich um die Zeitangabe, wann diese Ausgabe des „onstat“ erstellt wurde.
Das Format ist das Date-Time-Format „YYYY-MM-DD HH:MM:SS“ und lässt sich nicht durch GL_DATETIME verändern.

Beispiel:

```
IBM Informix Dynamic Server Version 14.10.FC5 -- On-Line -- Up 2 days
23:37:42 -- 1812528 Kbytes
2021-02-23 08:23:42
```

Der Vorteil dieser Änderung liegt darin, dass bei einer späteren Analyse der genaue Zeitpunkt der Erstellung ersichtlich ist, und nicht nur die Zeit, wie lange die Instanz bereits aktiv war.
Dies spielt besonders dann eine Rolle, wenn man den aktuellen Einfluss von Wartungs Scripts wie z.B. dem „update statistics high“, oder eine Sicherung als aktuelle Einflüsse auf die Ausgaben mit berücksichtigen muss.
Aus eigener Erfahrung im Consulting weiss ich, dass es bei der Analyse einer Sammlung von Ausgaben des „onstat“ immer wieder sehr hilfreich ist zu wissen, wann genau diese Snapshots gezogen wurden.

TechTipp: SBSpace einer BLOB Spalte ermitteln

Bei der Definition einer Tabelle kann für SmartBlobs (Datentyp BLOB oder CLOB) der Speicherplatz (SBSpace) angegeben werden. Wird keine Angabe über den Speicherort gemacht, so werden die Default Werte aus der $ONCONFIG für SBSPACE und SYSSBSPACE genutzt.

Werden jedoch bei der Definition der Tabelle mehrere SBSpaces als mögliche Ablageorte angegeben, so werden die einzelnen Einträge über die SBSpaces verteilt und der genaue Ablageort ist nicht mehr direkt ersichtlich.

Die Nummer des Spaces ist indirekt in der Referenz der Spalte zu finden und kann mittels SQL Abfrage ausgegeben werden.

Hierbei muss zwischen Sytemen mit Format „little endian“ wie Linux und Windows, sowie „big endian“ Systemen (z.B. AIX, SparcSolaris, …) unterschieden werden.

Auf „big endian“ Systemen ist die Abfrage der SpaceNummer relativ einfach:

```
("0x" || substr(<sblob_column>::lvarchar,17,8))::INT
```

Für „little endian“ is die Anfrage etwas aufwendiger:

```
  ("0x"|| substr(<sblob_column>::lvarchar,23,2)
       || substr(<sblob_column>::lvarchar,21,2)
       || substr(<sblob_column>::lvarchar,19,2)
       || substr(<sblob_column>::lvarchar,17,2))::INT
```

Der Cast auf LVARCHAR wird verwendet, da dieser implizit als Default zur Verfügung steht.

Beispiel:

```
create table blob_test (
objekt_nr       int,
doc             blob
) put doc in (sbdbs,sbdbs2)
;

insert into blob_test values (42, filetoblob("/etc/passwd","server"));
insert into blob_test values (43, filetoblob("/etc/hosts","server"));
insert into blob_test values (44, filetoblob("/etc/services","server"));

select objekt_nr ,
    ("0x"||substr(doc::lvarchar,23,2)
         ||substr(doc::lvarchar,21,2)
         ||substr(doc::lvarchar,19,2)
         ||substr(doc::lvarchar,17,2))::INT as space_nr,
        (select name::char(40)
            from sysmaster:sysdbspaces
            where dbsnum =  ("0x"||substr(doc::lvarchar,23,2)
                                 ||substr(doc::lvarchar,21,2)
                                 ||substr(doc::lvarchar,19,2)
                                 ||substr(doc::lvarchar,17,2))::INT
        ) as spacename
from blob_test
where doc is not null;
```

Ergebnis:

```
  objekt_nr    space_nr spacename
         42           3 sbdbs
         43           6 sbdbs2
         44           3 sbdbs
```

Das Beispiel zeigt, dass der erste und dritte Datensatz im Bereich „sbdbs“ gespeichert wurde, der zweite Datensatz in „sbdbs2“.

Da es sich bei diesen Informationen um interne Werte handelt und diese Abfrage nicht offiziell dokumentiert ist, kann es sein dass sich die Informationen in zukünftigen Versionen ändern.

TechTipp: SMI - Sysmaster Interface - sysshmhdr

Die Datenbank „sysmaster“ bietet eine Vielzahl an Informationen zur Auswertung. Einige Tabellen darin sind in der Dokumentation beschrieben, es gibt jedoch noch viel mehr zu entdecken.
Die Tabelle sysshmhdr zeigt eine Vielzahl an Werten an, wie z.B. das Profile der Bufferpools, die Aktivitäten auf den Platten, sowie Wartezustände. In der aktuellen Version sind es 299 Einträge.

Die Tabelle hat folgenden Aufbau:

```
{ Shared Memory Header }
    create table informix.sysshmhdr                { Internal Use Only }
        (
        number          integer,        { unique identifier for element }
        name            char(32),       { name of rhead_t element       }
        value           int8            { value of rhead_t element      }
        );
```

Zu finden sind z.B. ob sich die Instanz gerade in einem Checkpoint befindet, ob ein Rollback einer Long Transaction erfolgt und wie viel Memory und Cores des Rechners genutzt werden:

```
         31 longtx                                                 0
…
         57 pf_iscommits                                       45480
         58 pf_isrollbacks                                         0
…
        122 dskreads_2K                                        10388
        123 pagreads_2K                                        62293
        124 bufreads_2K                                      2769572
        125 dskwrites_2K                                       54100
        126 pagwrites_2K                                       81117
        127 bufwrites_2K                                      457422
…
        247 os_num_processors                                      2
        249 os_memory                                     2085687296
        250 os_free_memory                                1285156864
…
        283 shm_resident_size                                9764864
        284 shm_virtual_size                               194560000
…
        293 ckpt_pending                                           0
        294 ckpt_active                                            0
```

Die Werte dieser Tabelle eignen sich daher sehr gut für die Überwachung des Systems durch externe System Monitore. Auch die Boot-Time der Instanz kann abgefragt werden:

```
select name, dbinfo('utc_to_datetime',value) as boot_time
from sysshmhdr where name in ('bttime')

name                             boot_time
bttime                           2021-02-18 08:10:49
```

Hinweis: Es handelt sich um interne Informationen, die je Version unterscheiden können.

TechTipp: Extended Utilities

Extended Utilities sind User Definded Routines (UDR), die in der Sprache Java geschrieben sind.

Die Utilities werden in die Kategorien
- Character Utilities
- Explain Utilities
- Large Object Utilities
- J/Foundation Function Utilities

unterteilt.

Im INFORMIX Newsletter Q3-2020 hatten wir bereits die Funktion UUID() aus dem Bereich der Character Utilities vorgestellt.
Nach und nach werden wir weitere Funktionen vorstellen. Zukünftige Erweiterungen in diesem Bereich finden Sie natürlich ebenfalls wieder hier im Newsletter.
Um die Funktionen nutzen zu können, müssen diese einmalig von einem Benutzer mit der Berechtigung „Extend role“ in der Datenbank erstellt werden.
Zudem muss die Nutzung für einzelne Benutzer oder allgemein (Public) freigegeben werden.

In dieser Ausgabe kümmern wir uns vornehmlich um die LOB Funktionen.
Als Basis dient folgender Versuchsaufbau:

```
create table lob_test (
nr  int,
doc_bin  blob,
doc clob
) put doc_bin in (sbdbs,sbdbs2), doc in (sbdbs,sbdbs2)
;

insert into lob_test values (42,
filetoblob("/etc/passwd","server"),filetoclob("/etc/passwd","server")
);

insert into lob_test values (43,
filetoblob("/etc/hosts","server"),filetoclob("/etc/hosts","server")
);

insert into lob_test values (44,
filetoblob("/etc/services","server"),filetoclob("/etc/services","server")
);
```

TechTipp: LOB Functions - lobSize()

Eine sehr häufig gestellte Frage ist, wie man heraus findet, wie gross gespeicherte Objekte vom Typ CLOB oder BLOB sind. Mit üblichen SQL Mitteln ist diese Frage nur sehr schwer zu beantworten.
Um die Funktion lobSize() nutzen zu können, muss diese zuerst in der Datenbank als UDR erstellt werden:

```
CREATE FUNCTION lobSize(BLOB) RETURNS BIGINT
    EXTERNAL NAME
    'com.informix.judrs.LargeObjects.lobSize(java.sql.Blob)'
    LANGUAGE JAVA;
CREATE FUNCTION lobSize(CLOB) RETURNS BIGINT
     EXTERNAL NAME
     'com.informix.judrs.LargeObjects.lobSize(java.sql.Clob)'
     LANGUAGE JAVA;
GRANT EXECUTE ON FUNCTION lobSize(BLOB) TO PUBLIC;
GRANT EXECUTE ON FUNCTION lobSize(CLOB) TO PUBLIC;
```

Nun können wir die Grösse der Objekte abfragen:

```
select nr, lobsize(doc)
from lob_test
```

Ergebnis:

```
         nr          (expression)
         42                  3255
         43                   221
         44                 19183
```

Zum Vergleich die Abfrage im Linux:

```
du -sb
    3255    /etc/passwd
    221     /etc/hosts
    19183   /etc/services
```

Es wird die Grösse in Bytes ausgegeben.
Hiermit ist es nun möglich die minimale, maximale und durchschnittliche Grösse der Objekte zu ermitteln. Ebenso der gesamte, durch die gespeicherten Objekte belegte Platz.

TechTipp: LOB Functions - concat()

Die Funktion concat() ermöglicht es, Daten vom Typ CLOB zu ergänzen. Dabei wird der angegebene String vom Typ CHAR/VARCHAR/LVARCHAR am Ende angehangen und es entsteht ein neues Objekt vom Typ CLOB.

Zuerst erstellen wir wieder die Funktion:

```
create function concat(CLOB,LVARCHAR) RETURNS CLOB
        EXTERNAL NAME
'com.informix.judrs.LargeObjects.concat(java.sql.Clob.java.lang.String)'
LANGUAGE JAVA;
grant execute on function concat(CLOB, LVARCHAR) to PUBLIC;
```

Vor der Änderung schauen wir uns an, was wir als Inhalt für „hosts“ in der Tabelle gespeichert hatten:

```
select doc from lob_test where nr = 43;

doc
127.0.0.1       localhost
127.0.1.1       kalu42

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Nun fügen wir die neue IP Adresse mittels „update“ an:

```
update lob_test set doc = concat(doc, "### added by IFX ###
192.168.1.42    kalu42_second_ip") where nr = 43;
```

Eine erneute Abfrage zeigt, dass der Inhalt des CLOB ergänzt wurde:

```
doc
127.0.0.1       localhost
127.0.1.1       kalu42

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
### added by IFX ###
192.168.1.42    kalu42_second_ip
```

Da es die Funktionen „filetoclob()“ und „lotofile()“ ermöglichen, Dateien einzulesen und auch wieder zu speichern, ist es möglich, Ergänzungen im Dateisystem über die Datenbank zu verteilen und zu pflegen.

TechTipp: LOB Functions - toClob()

Das Einfügen von Daten in Felder vom Typ CLOB ist nicht nur mittels der Funktion filetoclob() möglich, sondern kann auch mittels toClob() bewerkstelligt werden. Als Eingabe können die Datentypen CHAR/VARCHAR/LVARCHAR verwendet werden.

Wie immer müssen wir zuerst die notwendigen Funktion erstellen:

```
CREATE FUNCTION toClob(LVARCHAR) RETURNS CLOB
    EXTERNAL NAME
'com.informix.judrs.LargeObjects.toClob(java.lang.String)'
LANGUAGE JAVA;
GRANT EXECUTE ON FUNCTION toClob(LVARCHAR) TO PUBLIC;
```

Danach können wir unsere Daten in die CLOB Felder eintragen:

```
insert into lob_test (nr, doc) values (45,
toClob(
"Test Zeile 1
    und noch ne Zeile
sowie der letzte Eintrag"
));
```

Die Kontrollabfrage liefert:

```
select doc
from lob_test
where nr = 45;

doc
Test Zeile 1
    und noch ne Zeile
sowie der letzte Eintrag
```

TechTipp: LOB Functions - toString()

Zur Umwandlung von String nach CLOB gibt es natürlich auch die Gegenrichtung. Zuerst wie immer die Funktion erstellen:

```
CREATE FUNCTION toString(CLOB) RETURNS LVARCHAR
    EXTERNAL NAME
    'com.informix.judrs.LargeObjects.toString(java.sql.Clob)'
    LANGUAGE JAVA;
GRANT EXECUTE ON FUNCTION toString(CLOB) TO PUBLIC;
```

Nun kann ein CLOB in einen String umgewandelt werden, auf dem z.B. die String Funktionen UPPER(), LOWER(), INITCAP() oder SUBSTR() angewandt werden können:

```
insert into lob_test values (47, toClob("Kalu testet"));

select upper(toString(doc)) from lob_test where nr = 47;
# KALU TESTET
```

TechTipp: LOB Functions - append()

Beim Aufruf von concat() wurde aus einem bestehenden CLOB und einem String ein neues Objekt vom Type CLOB erstellt. Mittels append() wird ein bestehendes Objekt vom Type CLOB erweitert.

Wie immer muss zuerst die Funktion erstellt werden:

```
CREATE FUNCTION append(CLOB, LVARCHAR) RETURNS CLOB
    EXTERNAL NAME    'com.informix.judrs.LargeObjects.append(
        java.sql.Clob,java.lang.String
    )'
    LANGUAGE JAVA;
GRANT EXECUTE ON FUNCTION append(CLOB, LVARCHAR) TO PUBLIC;


create table lob_test (
nr  int,
doc_bin  blob,
doc  clob
) put doc_bin in (sbdbs,sbdbs2), doc in (sbdbs,sbdbs2)
;
```

Wir fügen zuerste einen Datensatz in unsere Tabelle ein:

```
insert into lob_test (nr, doc) values (48, toClob("Kalu testet"));
```

Testabfrage:

```
select doc from lob_test where nr = 48;
# doc
# Kalu testet
```

Erweitern des Dateninhalts:

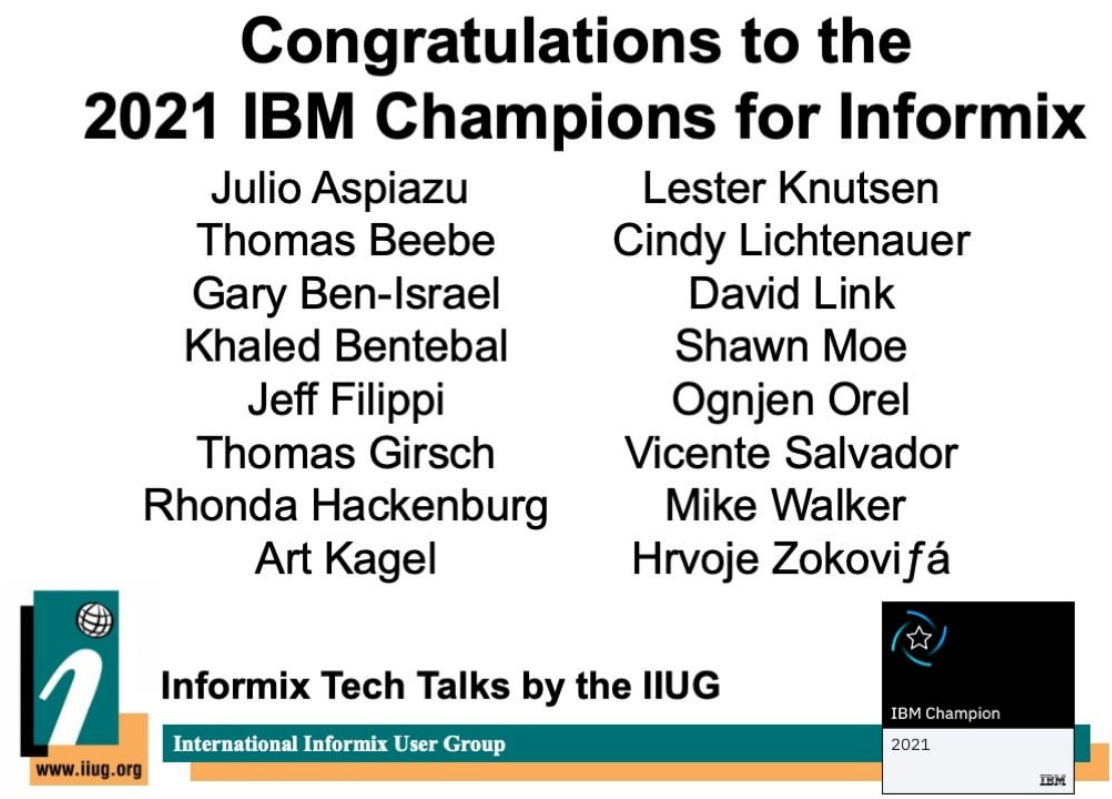
```
execute function append((SELECT doc from lob_test where nr = 48)," und
ist erfolgreich");
# (expression)
# Kalu testet und ist erfolgreich
```

Erneuter Test nach Erweiterung:

```
select doc from lob_test where nr = 48;
doc
# Kalu testet und ist erfolgreich
```

Hinweis: INFORMIX Champions 2021

Wie in jedem Jahr wurden auch dieses Jahr die INFORMIX Champions gekürt.
Wer mit INFORMIX zu tun hat, der wird viele dieser Namen bereits kennen und wissen, wie sehr sie sich für INFORMIX einsetzen. Viele wertvolle INFORMIX Tutorials und Veranstaltungen gehen auf diese Gruppe zurück.
Auch wir wollen uns den Glückwünschen und dem Dank anschliessen !



Newsletter Archive

Vielen Dank dass uns auch nach der lange Pause einige Archive wieder aufgenommen haben.
Hier eine Übersicht der Archive von denen wir wissen:

- https://iug.de/informix-newsletter
- https://www.cursor-distribution.de/de/ibm-software/informix-fuer-administratoren/informix-newsletter-ibm
- https://bereos.eu/informix-newsletter.html
- https://www.nsi.de/informix/informix-newsletter

Sie haben auch den Newsletter in einem öffentlichen Archiv abgelegt ?
Dann senden Sie uns den Link, damit wir diesen hier mit einpflegen können.

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 Web Seiten 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.
Rückmeldungen hierzu sind herzlich Willkommen !
Die gefundenen Tippfehler dürfen zudem behalten und nach Belieben weiterverwendet werden.

Eine Weiterverbreitung in eigenem Namen oder eine Bereitstellung auf der eigenen HomePage ist ausdrücklich erlaubt. Alle hier veröffentlichen Scripts stehen uneingeschränkt zur weiteren Verwendung zur Verfügung.

Die Autoren dieser Ausgabe

Gerd Kaluzinski IT-Specialist Informix, DB2, InfoSphere CDC, DataStage
IBM Analytics
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Besonderer Dank geht an Martin Fuerderer der HCL, der zuverlässig die Korrektur und Überprüfung der Inhalte und der Tippfehler übernimmt, sowie an Andreas Legner (ebenfalls HCL), der mich bei der Recherche neuer Features unterstützt.
Dank auch an die vielen Helfer im Hintergrund.

Nicht zu vergessen der Dank an die Informix User Group, ohne die es keinen Neuanfang des INFORMIX Newsletters gegeben hätte und die dankenswerter Weise die Verteilung übernimmt.

Foto Nachweis:
Redaktionsgarten Januar und Februar 2021 (Gerd Kaluzinski)
Informix Chanpions Liste – International Informix User Group Mailing