Willkommen zum „IBM DB2 Newsletter“

Liebe Leserinnen und Leser,

Weihnachten steht vor der Tür – klopft eigentlich schon an-, auch wenn man das vom Wetter her eher nicht vermuten kann.
Um diese nasskalte Jahreszeit zu überbrücken, haben wir wieder interessante Artikel zum lesen und ausprobieren zusammengetragen.

Wir wünschen ein
Frohes Fest und einen guten Start ins Jahr 2016!

Für Fragen und Anregungen unsere Kontaktadresse: db2news@de.ibm.com.

Ihr TechTeam

Inhaltsverzeichnis

DB2 NL 02/2015 – Nachtrag – DB2 Backup

Im Artikel „Backup include Log ...“ haben wir darauf hin gewiesen, das ein Backup geschrottet werden kann, wenn das Retrieve der Logfiles am Ende vom Backup fehlschlägt.
Ein fleißiger Leser (Berni Schmidt) hat uns den entscheidenden Tipp gegeben, wie man das Backup erhalten kann, auch wenn Retrieve fehlschlägt.
Wenn man per db2set den Parameter

```
DB2_BCKP_INCLUDE_LOGS_WARNING=YES
```

setzt, wird das Backup nicht gelöscht, sondern ohne die Logs auf dem Datenträger erhalten. Dieses kann bei großen Datenbanken, die nur ein Sicherungsfenster am Wochenende haben sinnvoll sein.
Frei nach dem Motto, besser ein Backup ohne incl. Logs als gar keins.

db2pd zum Monitoring

Da arbeitet man Jahrlang mit DB2 und Tools und lernt immer noch dazu. Kennen Sie alle Optionen vom db2pd? Ich habe eine für mich „neue“ kennengelernt:
-repeat

```
--+-----------------------------------+--+-----------+--------->
   '- -repeat--+---------+--+-------+-'  '- -runstats-'
```

zum fortlaufenden Anzeigen der Information:

Beispiel zum Monitoring des HADR Zustandes bei einer Übernahme:

```
db2pd -db <DBNAME> -hadr -repeat
```

Richtige Ladereihenfolge mit rekursivem SQL

Neulich stand ich vor der Aufgabe, den Inhalt einer ganzen Reihe von Tabellen, in eine andere Datenbank zu laden. Alle diese Tabellen waren in einem Schema angelegt (nennen wir es „MYSCHEMA“).

Dazu standen mir Dateien zur Verfügung, die jeweils ein INGEST Kommando enthielten, mit dem eine Tabelle geladen werden konnte. Die Dateien hatten den Namen:

MYSCHEMA.*Tabellenname*.ingest

In den Dateien befanden sich INGEST Kommandos, die in etwa wie folgt aussahen:

INGEST FROM FILE *Schemaname.Tabellenname*.del FORMAT DELIMITED INSERT INTO *Schemaname.Tabellenname* (FELD1, FELD2, ...)

Zunächst habe ich eine Schleife über die SYSCAT.TABLES gemacht und die Tabellennamen des Schemas MYSCHEMA ausgelesen und die Aufrufe der Dateien damit erzeugt.

Leider hatte ich vergessen, dass es Foreign-Key Beziehungen zwischen den Tabellen gab. In den Fällen, wo ich versuchte die abhängige Tabelle (Child) vor der übergeordneten Tabelle (Parent) zu laden, wurde ich mit einer Fehlermeldung bestraft, die z.B. so aussah:

```
SQL2905I  The following error occurred issuing the SQL "INSERT" statement on
table "MYSCHEMA.CHILD" using data from line "2" of input file "CHILD.del".
SQL0530N  The insert or update value of the FOREIGN KEY "MYSCHEMA.CHILD.FK_CHILD" is not
equal to any value of the parent key of the parent table.  SQLSTATE=23503
```

Um den Konflikt zu lösen, könnte man z.B. statt INGEST lieber LOAD verwenden und die Constraints ignorieren lassen. Anschliessend sind die Tabellen in CHECK PENDING Zustand und man muss mit SET INTEGRITY die Constraints prüfen.

Eine andere Strategie ist, die Tabellen in der "richtigen" Reihenfolge zu laden. Dafür kann rekursives SQL verwendet werden. Dieses möchte ich an dieser Stelle erläutern.

In der SYSIBM.SYSFOREIGNKEYS Tabelle stehen die FK Beziehungen. Mit folgenden SQL erhält man die richtige Reichenfolge der Tabellennamen.

```
with FKINFO (PKTABLE_SCHEM, PKTABLE_NAME, FKTABLE_SCHEM, FKTABLE_NAME, LEVEL)
as (
     select PKTABLE_SCHEM, PKTABLE_NAME, FKTABLE_SCHEM, FKTABLE_NAME,
            cast(1 as integer) as level
       from  SYSIBM.SQLFOREIGNKEYS
      where trim(PKTABLE_SCHEM) like 'mySchema'
      UNION ALL
     select k.PKTABLE_SCHEM, k.PKTABLE_NAME, k.FKTABLE_SCHEM, k.FKTABLE_NAME,
            i.level + 1 as level
       from FKINFO i, SYSIBM.SQLFOREIGNKEYS k
      where i.PKTABLE_SCHEM=k.FKTABLE_SCHEM and i.PKTABLE_NAME=k.FKTABLE_NAME
)
select  t.tabschema, t.tabname, max(coalesce(LEVEL,0)) as LEVL
from FKINFO
    right outer join syscat.tables t
   on t.tabschema = FKINFO.PKTABLE_SCHEM and t.tabname=FKINFO.PKTABLE_NAME
where t.type='T' and trim(t.tabschema) like 'mySchema'
group by t.tabschema, t.tabname
order by LEVL desc
```

Kernstück ist die Commen Table Expression ("with .."), mit dem rekursivem SQL. Im ersten Teil des UNION wähle ich alle FK-Beziehungen aus und vergebe diesen den Level=1. Das entspricht dem Startpunkt einer vollständigen Induktion im Sinne eines mathematischen Beweises. Mathematisch „vorbelastete" finden sich hier schnell wieder.

Im zweiten Teil des UNION verjoine ich die Beziehungen, die ich habe, mit weiteren FK-Beziehungen, die zu jeweils nächsten Tabelle führen. Das entspricht in der vollständigen Induktion dem Schritt von Ebene N zu N+1.

Beispiel:

```
Tabelle1 --FK--> Tabelle2                          Level Tabelle1=1
Tabelle8 --FK--> Tabelle9 --FK--> Tabelle10        Level Tabelle8=2
```

Da mich nur die maximalen Längen solcher Ketten interessieren, nehme ich nur die Max(Level). Das verjoine ich mit den Einträgen der syscat.tables per outer join, um die Tabellen zu bekommen, die gar keine FK-Beziehungen haben. Diese erhalten den Level NULL - den ersetze ich durch 0.

Somit erhalte ich

- Tabellen ohne FK-Beziehungen (Level=0)
- Tabellen mit nur einer FK Beziehung (Level=1)
- Tabellen, die durch eine Kette von FK-Beziehungen erreichbar sind (Level=N, wobei N>1)

Das Sortieren nach dem Level gibt mit die Richtige Reihenfolge, um die Tabellen zu laden. Man erhält beispieldweise folgendes Ergebnis:

```
TABSCHEMA        TABNAME                                                  LEVL
---------------------------------------------------------------- ------------------
SQL0347W  The recursive common table expression "MYSCHEMA.FKINFO" may contain
an infinite loop.  SQLSTATE=01605

MY_SCHEMA              TABLE1                                             3
MY_SCHEMA              TABLE2                                             3
MY_SCHEMA              TABLE3                                             2
MY_SCHEMA              TABLE4                                             1
…
```

Man beachte: Es wird eine Warnung ausgegeben, da rekursives SQL auch zu Endlosschleifen führen kann. Diese Warnung kann ignoriert werden.

Mit der so hergestellten Reihenfolge (z.B.über entsprechenden Join), können die Aufrufe in die richtige Reihenfolge gebracht werden.

Fazit: Rekurives SQL ist ein sehr nützliches Feature, welches u.a. auch dafür verwendet

werden kann, um Reihenfolgen, die durch FK-Beziehungen erzwungen werden, herauszufinden.

HADR und TSA: backup after takeover

Im Artikel „Stolpersteine beim Einsatz von DB2 HADR mit ACR und TSA" (DB2 Newsletter – Ausgabe 01/2015, Autor Andrea Ott) wurde darauf hingewiesen, dass eine Wiederherstellung mit Incremental-Backups verschiedener Cluster-Seiten nicht möglich ist. Deswegen ist es empfehlenswert nach einem Takeover auf der neuen Primary-Seite als nächstes auf jeden Fall ein Full-Backup durchzuführen.

Nachfolgend wird eine TSA-basierende Lösung präsentiert, die einen Full-Backup nach einem Takeover automatisch erstellt.

Konzept

Um einen Backup nach einem Takeover anstoßen zu können, werden zwei Arten von TSA-Ressourcen verwendet (s. das Bild):

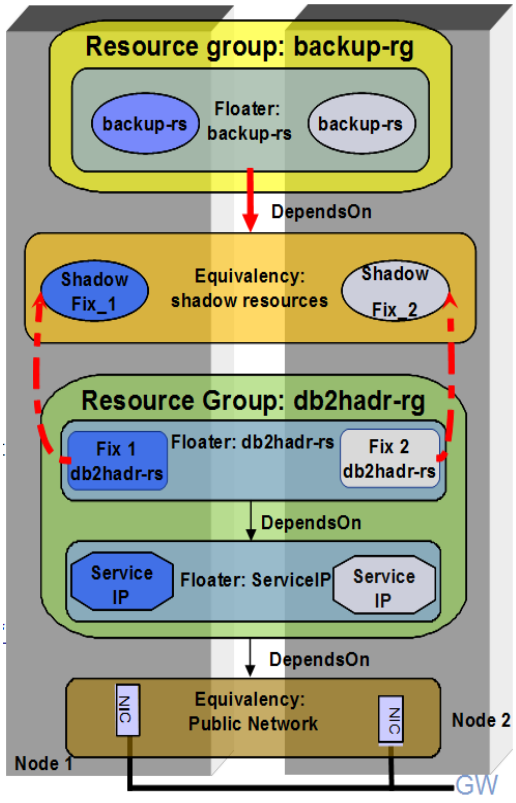
- Ressourcen, die einen Takeover erkennen (das Monitoring)
- Ressourcen, die den Backup anstoßen (der Starter).

Um festzustellen, dass ein Takeover stattgefunden hat, wäre eine **`DependsOn`**-Beziehung von einer Starter-Ressource (**`backup-rs`**), die ein Backup auslösen sollte, zur HADR-Ressource (**`db2hadr-rs`**) nützlich. Weil aber die festen HADR-Ressourcen auf den unterschiedlichen Knoten aktiv sind, braucht man Schattenressourcen als eine Zwischenschicht dazu.

Eine Schattenressource hat folgende Merkmale:

- Sie ist eine feste Ressource der Klasse **`IBM.Application`** und überwacht den Betriebsstatus (**`OpState`**) einer anderen festen Ressource.
- Sie gehört zu einer Äquivalenz, deren Attribut **`SelectFromPolicy`** auf den Wert "**`NoControl`**" gesetzt ist. Dieser Wert gibt an, dass TSA nur den Betriebsstatus der zur Äquivalenz gehörenden Ressourcen auswertet, *die Ressourcen jedoch nicht startet oder stoppt.*

Man muss Schattenressourcen verwenden, wenn man zwischen einer variablen Ressource und einer von mindestens zwei festen Ressourcen, die auf verschiedenen Knoten ausgeführt werden, eine **`DependsOn`**-Beziehung definieren möchte, denn die **`DependsOn`**-Beziehung löst sowohl ein **`StartAfter`**- als auch ein **`StopAfter`**-Verhalten aus.



Wenn man eine solche **`DependsOn`**-Beziehung konfiguriert *ohne Schattenressourcen zu definieren*, kommt kann es zu unerwünschten Automationsverhalten kommen.

Daher wird als Auslöser für die Starter-Ressource (**`backup-rs`**) eine NoControl-Equivalency aus zwei „fixed" Schattenressourcen dienen, die den Betriebsstatus (**`OpState`**) der nativen Primary- und Standby- Datenbank-Ressourcen (**`db2hadr-rs`**) überwachen.

Zwischen der Starter-Ressource (**`backup-rs`**) und der Equivalency besteht eine **`DependsOn`** Beziehung.

Implementierung

Die ganze Logik ist in 2 Skripten implementiert (s. Skriptstexte am Ende des Artikels):

- **hadrV97_monitor_shadow_primary.ksh**
- **hadrV97_backup_after_takeover.ksh**

die auf den beiden Knoten des Clusters im Verzeichnis **/usr/sbin/rsct/sapolicies/db2/** liegen, wie alle TSA Policy-Skripts für DB2 auch.

Alle nötigen TSA-Ressourcen lassen sich mithilfe von Skripten **1_mkresource_hadr_shadow_primary.sh** und **2_mkresource_backup_after_takeover.sh** anlegen. Man soll sie lediglich in dieser Reihenfolge mit den richtigen Parameter auf einem von den beiden Knoten unter den User **root** anstoßen. Einen Skript-Aufruf mit allen nötigen Parameter kann man sehen, wenn man das Skript ohne Parameter bzw. mit dem Parameter „**-help**" aufruft.

Monitoring

Das Monitoring wird im Skript **hadrV97_monitor_shadow_primary.ksh** implementiert.

Mit dem Skript wird der **OpState** der HADR-Ressource auf dem Knoten überwacht, auf dem das Skript gestartet ist. Wichtig ist es, dass es auch überwacht wird, ob das **move**-Request die HADR-Gruppe (**db2hadr-rg**) immer noch belegt.

*Erst wenn nach einem Takeover der **OpState** der neuen Primary-DB **Online** ist und KEIN **move**-Request die HADR-Gruppe mehr belegt, wird ein Backup angestoßen.*

Die zahlreichen Tests haben nämlich ergeben, dass ein **OpStatus = Online** als alleinige Bedingung nicht ausreicht, da in diesem Fall ein Backup viel zu früh gestartet wird, währenddessen die **lock**- bzw. **move**- Requests die HADR-Gruppe immer noch belegen.

In der Praxis manifestiert sich dies dadurch, dass wenn man nur die Bedingung **OpStatus = Online** berücksichtigt und in einer Konsole einen Befehl „**db2 takeover hadr on db DBNAME** " absetzt, kommt die Kommandozeile so lange nicht zurück, bis ein Backup, der dabei ausgelöst wird, abgeschlossen ist (es kann wohl ziemlich lange dauern).

Dabei sieht man im **db2diag.log** unmittelbar nach der Info-Meldung „**Backup complete**" eine Error-Meldung, die zeigt, dass kein **UnLock** während des Backups möglich war:

```
2014-10-30-18.11.05.113610+060 E1518892E432        LEVEL: Info
PID     : 9598                 TID  : 46937894021440PROC : db2sysc 0
INSTANCE: db2inst1              NODE : 000          DB   : SAMPLE1
APPHDL  : 0-126                APPID: *LOCAL.db2inst1.141030171045
AUTHID  : DB2INST1
EDUID   : 423                  EDUNAME: db2agent (SAMPLE1) 0
FUNCTION: DB2 UDB, database utilities, sqlubcka, probe:906
MESSAGE : Backup complete.

2014-10-30-18.11.19.872527+060 E1519325E442        LEVEL: Error
PID     : 9598                 TID  : 46937940158784PROC : db2sysc 0
INSTANCE: db2inst1              NODE : 000          DB   : SAMPLE1
APPHDL  : 0-130                APPID: *LOCAL.db2inst1.141030170935
AUTHID  : DB2INST1
EDUID   : 426                  EDUNAME: db2agent (SAMPLE1) 0
FUNCTION: DB2 Common, SQLHA APIs for DB2 HA Infrastructure, sqlhaUnLockHADRResource, probe:100
```

Starter

Ein Backup wird aus dem Skript **hadrV97_backup_after_takeover.ksh** gestartet.

Eine Backup-Ressourcen-Gruppe (**backup-rg**) enthält eine gewöhnliche ‚floating'-Ressource (**backup-rs**), die mit einem Skript den Backup startet, „stoppt" und überwacht. Obwohl das Stoppen hier im übertragenen Sinne gemeint ist: Es wird bloß ein leeres File gelöscht.

Diese Ressource kann man als eine Task-Ressource bezeichnen, da in ihr ein Laufzustand eines relativ kurz laufenden Backups dem Datenmodell von in der Regel lang laufenden Prozessen,

welche von der TSA gesteuert werden, angepasst wird.

Tasks im Kontext der System Automation sind einfache, wiederkehrende Aktionen wie Backups, Installation von Updates oder ändern von Konfigurationen einiger Services.

Ein Zustandsübergang der Task-Ressource wird im Skript **`hadrV97_backup_after_takeover.ksh`** auf folgende Weise erreicht:

- Modus „start": Ein Backup-Befehl bzw. Skript wird angestoßen (evtl. sollte die Skript-Variable **`cmd`** angepasst werden). Nachdem der Backup erfolgreich abgeschlossen ist, wird ein leeres Key-File **`/home/instancename/ADMIN/instancename_DBNAME.key`** angelegt. Im Fall eines Fehlers verfällt die Backup Ressource nach einer kurzen Zeit in den Zustand „**`Failed Offline`**".
- Modus „stop": Das Key-File wird gelöscht.
- Modus „status": Es wird geprüft, ob das Key-File existiert.

Operative Zustände einer Backup-Ressource

Die beobachteten Zustandswerte und Zustandsübergänge der Task-Ressource (**`backup-rs`**) können folgendermaßen interpretiert werden:

Zustand	Bedeutung
Offline	Der Backup läuft momentan nicht. Kein Key-File existiert als Ergebnis eines vorherigen Laufs.
Pending Online	Der Backup läuft. KEIN Key-File existiert.
Online	Der Backup läuft momentan nicht. Der letzte Lauf war erfolgreich. Das Key-File existiert.
Failed Offline	Der Backup läuft momentan nicht. Der letzte Lauf war NICHT erfolgreich. Das Key-File existiert NICHT.

- Zunächst ist die Task-Ressource im Zustand **`Offline`**.
- Wenn eine Online-Anfrage gegen die Ressource kommt (Start-Modus des Skripts), wird ihr Zustand auf "**`Pending Online`**" verändert.
- Wenn der Backup erfolgreich abgeschlossen ist (Skript kehrt mit einer "0" zurück), bekommt die Ressource einen neuen End-Zustand **`Online`**.
- Wenn der Backup fehlschlägt (Skript endet mit einem Return-Code ungleich "0"), wird die Ressource in den Zustand „**`Failed Offline`**"gesetzt.

`Online` bzw. „**`Failed Offline`**" ist ein End-Zustand, der das Ergebnis eines erfolgreichen bzw. gescheiterten Backups bei dieser Ressource darstellt. Der Zustand „**`Failed Offline`**" wird bis dahin bestehen, bis ein manuell ausgelöster "reset"-Befehl (**`resetrsrc`**) diese Ressource wieder in **`Offline`**-Zustand versetzt.

Deaktivierung des Feature

Es ist vorgesehen das „backup after takeover" Feature bei Bedarf abschalten zu können (Tests, Wartungsarbeiten etc.). Dies wird erreicht, wenn man ein File namens **`/home/instancename/ADMIN/instancename_DBNAME.nobackup`** auf dem zukünftigen Primary-Node ablegt, auf dem nach einem Takeover ansonsten ein Backup ausgelöst würde:

```
touch /home/instancename/ADMIN/instancename_DBNAME.nobackup
```

Unter dem **`instancename`** ist ein Instance Owner gemeint (z. B. db2inst1), **`DBNAME`** ist ein Platzhalter für einen Datenbankname.

Die Skripte dazu sind in der IBM-BOX zu finden.

Chats mit dem Labor

Eine Liste der bereits durchgeführten Chats ist hier zu finden.
Die Präsentationen der Chats können dort angeschaut und heruntergeladen werden.

Schulungen / Tagungen / Informationsveranstaltung

DB2 Regionale Usergroup

Das 10. DeDUG Meeting (DB2 LUW User Group) findet am 19. Februar 2016 in München (IBM) statt.

Agenda und Anmeldung zu diesem kostenlosen Event hier.

Nähere Details zur DeDUG auf der IDUG Webseite.

DB2 Services online zum Vorzugspreis bestellen

Kennen Sie schon unseren Webauftritt zum IBM Software Service Shop?

www.ibm.com/de/softwareserviceshop

Dort finden Sie zahlreiche Software Service Angebote,
darunter auch die beiden folgenden DB2 Services:

Datenbank Konfigurations-prüfung durch Guardium

DB2 Datenbank Health Check

Besuchen Sie uns und
erfahren Sie mehr Details
zu unseren Angeboten.

Newsletter Archiv

Wir haben ein weiteres Archiv für den DB2 Newsletter bei

Alte Ausgaben vom DB2-NL sind nun zum Nachlesen in den Archiven zu finden von:

- Bytec
- Cursor Software AG
- Drap
- ids-System GmbH
- Lis.Tec
- ORDIX

Anmeldung/Abmeldung

Sie erhalten diesen Newsletter bis zur 3ten Ausgabe ohne Anmeldung. Wenn Sie weiterhin diesen Newsletter empfangen wollen, schicken Sie Ihre Anmeldung mit dem Subject „ANMELDUNG" an db2news@de.ibm.com.

Die Autoren dieser Ausgabe

Sollten Sie Anfragen zu den Artikeln haben, können Sie sich entweder direkt an den jeweiligen Autor wenden oder stellen Ihre Frage über den DB2 NL, denn vielleicht interessiert ja die Antwort auch die anderen DB2 NL Leser.

Doreen Stein	Master Certified IT-Spezialist DB2 LUW, IBM Certified DB2 LUW 10.5, IBM SWG Services; Chief-Editor DB2NL, Dipl-Inf (TU). djs@de.ibm.com
Frank Berghofer	Artikel: „Richtige Ladereihenfolge mit rekursivem SQL“
Vlad Cohen (arcor) Konstantin Konson (IBM)	Artikel: „HADR und TSA: backup after takeover“

Reviewer und Ideenlieferanten:

Dirk Fechner	IBM SWG
Peter Schurr	IBM SWG
Frank Berghofer	IBM SWG
Gerhard Paulus	ids-system GmbH