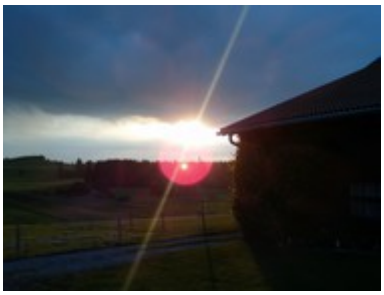
Willkommen zum „IBM DB2 Newsletter“

Liebe Leserinnen und Leser,



der Sommer ist vorbei, der Herbst mit seinen Bunten Farben ist im Jahreszeitenhaus eingezogen. DB2 V10.5 kam mit seinem Fix Pack 4 heraus.
Die IBM Insight in Las Vegas steht unmittelbar bevor. Diesmal dabei ist auch ein deutscher Beitrag zum erfolgreichen Einsatz von DB2 BLU (Session IWM-4672A).

Wir haben wieder einige Beiträge für Sie zusammengestellt und wünschen viel Spaß beim lesen und ausprobieren.

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

Ihr TechTeam

Inhaltsverzeichnis

DB2 Aktuell 2014 – Nachtrag

Die DB2 Aktuell im September gehört inzwischen zu den schönen Erinnerungen, der Kategorie „Klein aber fein". Die Anzahl der Teilnehmer und Beiträge ist überschaubar.

Beigetragen dazu haben die technischen Beiträge, die gelungene Auswahl der Lokation - das Swisshotel am Dresdner Schloss -, der Besuch des Mathematisch-Physikalischen Salons im Zwinger sowie das anschließende Festmahl im Sophienkeller (Taschenbergpalais) bei.

Neben den Vorträgen und dem aktiven Informationsaustausch gab es auch wieder die Möglichkeiten der Teilnahme an Hands-On Labs und der kostenfreien Zertifizierung.

STRING_UNITS ab DB2 10.5 FP4

Grundlagen

Bisher wurden die Längen von DB2 Datentypen, wie z.B. CHAR und VARCHAR, immer in Bytes angegeben. Das war vielen nicht bewusst, bis sie sich mit Unicode auseinandergesetzt haben.
Bei klassischen Codepages wurden alle darstellbaren Zeichen mit einem Byte gespeichert. Somit war es unerheblich ob man Zeichen oder Bytes gezählt hat, weil ein Byte einem Zeichen entsprochen hat.
Mit der Verwendung von Unicode Codepages – wie z.B. UTF-8 – können Zeichen nun in 1 bis zu 4 Bytes gespeichert werden und somit kann sich ein großer Unterschied zwischen Bytes und tatsächlichen Zeichen ergeben.
Während bei CHAR bzw. VARCHAR bisher die Länge in Bytes gezählt wurde, wurden bei GRAPHIC und VARGRAPHIC grundsätzlich zwei Bytes für ein Zeichen verwendet.

Problem

Fachliche Längendefinitionen beziehen sich meist auf Zeichen und nicht auf Bytes und die Migration einer Datenbank nach Unicode weist einige Problemstellungen auf, die nicht trivial sind. So arbeiten die klassischen Funktionen im SQL (wie z.B. SUBSTR(), LENGTH(), etc.) bisher auch auf Byte-Längen und man hat – in einer früheren DB2 Version – extra Funktionen eingeführt die auch auf Zeichenebene funktionieren. Ein Beispiel dafür ist die Funktion SUBSTRING(), die eine Angabe der Einheit erlaubt und per CODEUNITS32 auf Zeichen arbeitet, egal ob das Zeichen nun 1, 2, 3 oder 4 Bytes zur Speicherung benötigt. Dies wiederum bedeutet aber, dass Anwendungen geändert werden müssen, wenn eine Datenbank auf UTF-8 umgestellt wird, was immer zu hohen Aufwänden und Kosten führt. Auf der anderen Seite lässt sich UTF-8 heute nicht mehr wegdenken und diverse neue Funktionalitäten (z.B. BLU) bedingen sogar UTF-8 Datenbanken.

Neuerungen mit DB2 10.5.4

Mit Fixpack 4 kommen hier nun einige wichtige Änderungen bzw. Neuerungen. So kann jetzt angegeben werden, ob man die Längenangaben bei Strings wie bisher in Bytes oder lieber in Zeichen spezifizieren will.
Dies kann man auf folgenden Ebenen angeben:

- Auf Datenbank-Ebene
  STRING_UNITS ist ein neuer Datenbank-Konfigurationsparameter, der allerdings nicht online geändert werden kann.
  - `SYSTEM` (default) – Verhalten wie bisher, d.h. also Bytes als Längenangabe
  - `CODEUNITS32` – Längenangabe in Zeichen
- Auf Session-Ebene (nur in Unicode-Datenbanken)
  NLS_STRING_UNITS ist eine neue Umgebungseinstellung
  - <leer> (default) – somit gilt die Einstellung auf Datenbank-Ebene
  - `SYSTEM`
  - `CODEUNITS32`
- Auf Objekt-Ebene

Direkt bei der Spezifikation kann die Einheit mitangegeben werden
  - `CHAR (10 CODEUNITS32), VARCHAR(20 OCTETS)`

Zusätzlich zu diesen Einstellungen wurden auch neue Datentypen eingeführt, wie sie teilweise auch aus anderen Datenbanken bekannt sind. Im Detail sind das Folgende:

- Neue Datentypen:
  - `NCHAR`
  - `NVARCHAR`
  - `NCLOB`

Diese werden je nach Konfiguration in der Datenbank auf folgende Datentypen abgebildet:

- DB CFG: `NCHAR_MAPPING` (ONLINE änderbar)
  - `GRAPHIC_CU32` (default)
    - NCHAR => `GRAPHIC` mit `CODEUNITS32`
    - NVARCHAR => `VARGRAPHIC` mit `CODEUNITS32`
    - NCLOB => `DBLOB`[1] mit `CODEUNITS32`
  - `GRAPHIC_CU16`
    - NCHAR => `GRAPHIC` mit `CODEUNITS16`
    - NVARCHAR => `VARGRAPHIC` mit `CODEUNITS16`
    - NCLOB => `DBLOB` mit `CODEUNITS16`
  - `CHAR_CU32`
    - NCHAR => `CHARACTER` mit `CODEUNITS32`
    - NVARCHAR => `VARCHAR` mit `CODEUNITS32`
    - NCLOB => `CLOB` mit `CODEUNITS32`
  - `NOT APPLICABLE` – falls die Datenbank die Voraussetzung UNICODE nicht erfüllt

Die Einheiten im Detail:

- `OCTETS`
  Ist die klassische Angabe in Bytes.
- `CODEUNITS32`
  Definiert die Anzahl der Zeichen für String-Datentypen, ein Zeichen kann dabei 1-4 Bytes lang sein und entspricht somit UTF-8 Zeichen.
- `CODEUNITS16`
  Entspricht UTF-16, für alle Zeichen werden zwei Bytes verwendet (ausgenommen sog. „supplementary characters“)

Daraus ergibt sich, dass CHAR, VARCHAR, CLOB als OCTETS oder CODEUNITS32 definiert werden können und GRAPHIC, VARGRAPHIC, DBCLOB als CODEUNITS16 oder CODEUNITS32.
Einschränkungen:

- Funktioniert nur mit klassischen „row-organisierten“ Tabellen
- CODEUNITS32 geht nicht mit FOR BIT DATA

Beispiel: Verwendung unterschiedlicher Datentypen und Einheiten

```
create table t_datatype (id              int,
                         colnvarchar     nvarchar(10),
                         colnchar        nchar(10),
                         colcharbit      char(10) for bit data,
                         colcharcu32     char(10 codeunits32))
```

TABNAME	COLNAME	TYPENAME	LENGTH	TYPESTRINGUNITS	STRINGUNITSLENGTH
T_DATATYPE	ID	INTEGER	4	NULL	NULL
T_DATATYPE	COLNVARCHAR	VARCHAR	40	CODEUNITS32	10
T_DATATYPE	COLNCHAR	CHARACTER	40	CODEUNITS32	10
T_DATATYPE	COLCHARBIT	CHARACTER	10	NULL	NULL
T_DATATYPE	COLCHARCU32	CHARACTER	40	CODEUNITS32	10

[1] *DBLOB* für Unicode-Daten (bis zu 2GB)

Ein weiterer und entscheidender Vorteil ist, dass durch die Definition der Tabelle auf Zeichen- statt Byte-Länge es in Fixpack 4 möglich wird, die klassischen Funktionen (wie z.B. SUBSTR()) zu nutzen, da diese dann auch auf Zeichenebene arbeiten.
Bei Tabellen, die auf Bytes definiert werden, gelten nach wie vor die bekannten Einschränkungen und Probleme, wenn diese Unicodezeichen enthalten, die mehrere Bytes zur Speicherung benötigen.
Dadurch kann eine Migration auf Unicode maßgeblich vereinfacht und die Kosten der Anpassung von Anwendungen reduziert werden, da die Notwendigkeit entfällt die Funktionen zu ändern.

Beispiel: Klassische Funktionen bei Unicodezeichen

Fall 1: Tabellendefinition mit Angabe der Zeichen

```
create table t_datatype1 (c1 char(10 codeunits32)) organize by row;
insert into t_datatype1
       values ('ÄÜÖ'),(UX'05D0'),('äüö'),(UX'05D0' || UX'30E0');

select * from t_datatype1;
```

	C1
1	ÄÜÖ
2	א
3	äüö
4	אּ

```
select length(c1, octets) as length_in_bytes,
       substr(c1,1,1) as substr_c1
  from t_datatype1;
```

	LENGTH_IN_BYTES	SUBSTR_C1
1	6	Ä
2	2	א
3	6	ä
4	5	א

Alle Werte werden entsprechend – also mit einem Zeichen – ausgegeben und zusätzlich sieht man die Längen in Bytes.

Fall 2: Tabellendefinition mit Angabe von auf Bytes

```
create table t_datatype2 (c1 varchar(40 octets)) organize by row;
insert into t_datatype2
       values ('ÄÜÖ'),(UX'05D0'),('äüö'),(UX'05D0' || UX'30E0');

select length(c1, octets) as length_in_bytes,
       substr(c1,1,1) as substr_c1
  from t_datratype1;
```

	LENGTH_IN_BYTES	SUBSTR_C1
1	6	
2	2	
3	6	
4	5	

Trotz der identischen Abfrage bleibt das Ergebnis leer, da nur ein Byte des 2-Byte-Zeichens selektiert wurde und dies ungültig ist.
Den Beweis kann man über die Abfrage des Hex-Wertes machen:

```
select length(c1, octets) as length_in_bytes,
       substr(c1,1,1) as substr_c1,
       hex(substr(c1,1,1)) as hex
```

```
from t_datatype2;
```

	LENGTH_IN_BYTES	SUBSTR_C1	HEX
1	6		C3
2	2		D7
3	6		C3
4	5		D7

Führt man dieses Statement auf der ersten Tabelle – also t_datatype1 – aus, erhält man folgendes Ergebnis:

	LENGTH_IN_BYTES	SUBSTR_C1	HEX
1	6	Ä	C384
2	2	?	D790
3	6	ä	C3A4
4	5	?	D790

Fazit

Durch die Möglichkeit bei String-Datentypen eine Einheit anzugeben, den neuen Datentypen (NCHAR, NVARCHAR, NCLOB), sowie den neuen DB2 Konfigurationsoptionen, erhält man mit DB2 10.5 Fixpack 4 ein Paket, das die Migration nach Unicode wesentlich erleichtert. Zudem ist dies auch ein großer Fortschritt für den Data Warehouse Bereich, in dem die Daten aus unterschiedlichen Quellen verarbeitet werden, da die Anbindung von ETL-Tools durch diese Funktionalität einfacher werden dürfte. Dies alles wiederum wird zu einer Kosteneinsparung führen.

Bind nach Fixpack Upgrade auf einer Datenbank ohne PUBLIC Berechtigungen

Um den Sicherheitsanforderungen an eine Datenbank gerecht zu werden, erfolgt oftmals ein kompletter Entzug der PUBLIC Berechtigungen in einer Datenbank. Es empfiehlt sich dafür eine Ersatzrolle zu erstellen, welche exakt diesselben Berechtigungen erhält. Der Vorteil ist, dass dadurch nun nicht mehr jeder Benutzer implizit die PUBLIC Berechtigungen erhält, die auch einen Datenbankzugriff beinhalten. Einem neuen Benutzer muss nun diese Ersatzrolle explizit zugeordnet werden. Dies ist eine bewusste Entscheidung, während es mit PUBLIC ein impliziter Vorgang war.

Wenn allerdings ein Fixpack instaliert und eine Instanz aktualisiert wurde, stellt sich die Frage, wie die Standard Bind Befehle ausgeführt werden sollen. Werden diese mit „GRANT PUBLIC" ausgeführt, sind alle Korrekturen hinfällig und Packages sind wieder für PUBLIC berechtigt. Es gibt im BIND Befehl eine weitere Option „GRANT_ROLE", welche benutzt werden kann:

Aufruf: `BIND <bind_file> GRANT_ROLE <role_name>`

Zu berücksichtigen sind die üblichen Bind Dateien, bzw. Listendateien:

- `db2schema.bnd`
- `db2cli.lst`
- `db2ubind.lst`

Wird beispeilsweise eine Rolle DBPUBLI verwendet, wären folgende Kommandos passend:

1. **`db2 BIND db2schema.bnd BLOCKING ALL GRANT_ROLE DBPUBLI SQLERROR CONTINUE`**
2. **`db2 BIND @db2ubind.lst BLOCKING ALL GRANT_ROLE DBPUBLI`**
3. **`db2 BIND @db2cli.lst BLOCKING ALL GRANT_ROLE DBPUBLI`**

Es zeigt sich nun aber in Schritt 3, dass der Befehl auf die Liste `db2cli.lst` zu Fehlern der folgenden Art führt (für jedes abhängige Package), im Test führte dies bei 49 Packages zu

einer Warnung:

```
SQL0020W  The bind or precompile command parameters or parameter values in the following list were
ignored because they are not supported by the target database: "91"
```

Zwei Lösungsansätze wurden untersucht:

1. Bind mit GRANT PUBLIC - wie üblich:
   db2 BIND @db2cli.lst BLOCKING ALL GRANT PUBLIC
   Anschließend muss gegebenfalls wieder eine Korrektur der Berechtigungen von PUBLIC in eine Ersatzrolle durchgeführt werden
2. Verwendung der Option COLLECTION:
   Syntax:
   db2 BIND <bind file or list file> COLLECTION <schema> …
   Beispiel:
   db2 BIND @db2cli.lst COLLECTION DBPUBLI BLOCKING ALL GRANT_ROLE DBPUBLI

Bei Verwendung des Ansatzes 2 wird ein neues Package Schema namens DBPUBLI erstellt.

```
db2inst1:/home/db2inst1/sqllib/bnd# db2 "BIND @db2cli.lst COLLECTION DBPUBLI BLOCKING ALL
    GRANT_ROLE DBPUBLI "

LINE    MESSAGES FOR db2cli.lst
------  ------------------------------------------------------------------------
        SQL0061W  The binder is in progress.
        SQL0091N  Binding was ended with "0" errors and "0" warnings.

# db2 "select pkgschema, pkgname  , boundby from syscat.packages where pkgname = 'SYSSH200' " |
    tr -s " "
PKGSCHEMA PKGNAME BOUNDBY
------------------------------------------------------------------------------------------
DBPUBLI SYSSH200 db2inst1
NULLID SYSSH200 SYSIBM
 2 record(s) selected.
```

Im realen Betrieb zeigt sich allerdings, dass die NULLID Packages nicht gelöscht werden sollten, da diese standardmäßig von typischen Applikationen bei Verwendung von dynamischem SQL aufgerufen werden.
Im IBM Data Studio (Version 4.1) hatte dies den Minus-Effekt, dass Schemanamen im Baum nicht mehr dargestellt wurden. Das vom Data Studio verwendete Statement führte zum Fehler SQL0805N:

```
SELECT SCHEMANAME, REMARKS, OWNER FROM SYSCAT.SCHEMATA ORDER BY SCHEMANAME
DB2 SQL Error: SQLCODE=-805, SQLSTATE=51002, SQLERRMC=NULLID.SYSSH200 0X5359534C564C3031,
DRIVER=4.16.53
```

Das empfohlene Kommando für die DB2 CLI Packages per Bind-List-Datei db2cli.lst lautet daher:

```
db2 BIND @db2cli.lst BLOCKING ALL GRANT PUBLIC
```

oder

```
db2 BIND @db2cli.lst BLOCKING ALL
```

Nach dem Bind mir der Option „GRANT PUBLIC" zeigte sich, dass keine PUBLIC Berechtigungen hinzugefügt werden. Es sind daher danach keine Korrekturarbeiten erforderlich, auch wenn mit GRANT PUBLIC gearbeitet wird. Die Option GRANT kann somit auch entfallen. Es empfiehlt sich dennoch in der Praxis zu prüfen, ob dennoch PUBLIC Berechtigungen erstellt wurden. In diesem Fall sollten diese wieder entfernt werden.

Zusammenfassung:
Dies sind die empfohlenen Bind Befehle nach einem Fixpack Upgrade bei nicht existenten PUBLIC Berechtigungen, d.h. wenn eine Ersatzrolle DBPUBLI etabliert wurde:

- Bind File db2ubind.lst: Angabe von **GRANT_ROLE DBPUBLI**
  `db2 BIND @db2ubind.lst BLOCKING ALL GRANT_ROLE DBPUBLI`
- Bind File db2schema.bnd: Angabe von **GRANT_ROLE DBPUBLI**
  `db2 BIND db2schema.bnd BLOCKING ALL GRANT_ROLE DBPUBLI SQLERROR CONTINUE`

- Bind File db2cli.lst: Keine Angabe von GRANT_ROLE DBPUBLI möglich, auch ist **keine Angabe von GRANT PUBLIC erforderlich**:
  `db2 BIND @db2cli.lst BLOCKING ALL`
- Prüfung der GRANTs, und **bei Bedarf** anschließend nochmaliger Entzug der PUBLIC Berechtigungen:
  `db2 "SELECT COUNT(1) AS ct FROM syscat.packageauth WHERE GRANTEE = 'PUBLIC'"`
  Ergibt die Zählung einen Wert größer 0, müssen die PUBLIC Berechtigungen nochmals entfernt werden.

Zusatz: Scripts zum Ersetzen der PUBLIC Berechtigungen:

Methodik (für DB2 ab 9.7):

- Über die View `SYSIBMADM.PRIVILEGES` werden alle Objekttypen berücksichtigt (vor DB2 9.7 musste man über die einzelnen `SYSCAT.%AUTH` Views selektieren).
  Über die View `SYSCAT.DBAUTH` werden alle AUTHORITIES, welche Berechtigungen auf Datenbankebene behandeln, berücksichtigt.
- Es werden die passenden GRANT Befehle und REVOKE Befehle generiert.
  Einzelne Objekttypen benötigen spezielle Optionen, wie Routinen die Klausel SPECIFIC und beim REVOKE die Klausel RESTRICT. Oder es ist kein Schemanamen erforderlich.
- Die REVOKE Kommandos erfolgen gegen die Gruppe PUBLIC, die GRANT Kommandos gegen eine neue Ersatzrolle, im Beispiel DBPUBLI genannt.
- Die Ergebnisdateien sind genau zu prüfen, falls Fehler vorhanden sind, muss das dynamische Statement korrigiert und wiederholt werden. Dies ist wichtig, um zu vermeiden, dass Berechtigungen verloren gehen und nicht an die neue Rolle vergeben werden.
- Anschließend wird die neue Ersatzrolle (DBPUBLI) an die existierenden Benutzer AUTHIDs vergeben
- Die Befehle werden dynamisch generiert, die GRANT und REVOKE Befehle sollten nacheinander abgesetzt werden.

--- GRANT TO ROLE DBPUBLI (FILE gen_grants.sql)

```
---- Usage: db2 -xtf gen_grants.sql > grants.sql
----        db2 -tvf grants.sql -z grants.log
    SELECT ' GRANT ' || p.PRIVILEGE ||
    case WHEN      p.OBJECTTYPE IN ( 'TABLESPACE' )
           THEN ' OF '
           ELSE ' ON '
    END ||
    case WHEN p.OBJECTTYPE IN ( 'PROCEDURE' ,  'FUNCTION' )  THEN ' SPECIFIC ' ELSE ' ' END
           || REPLACE(REPLACE(REPLACE(REPLACE(TRIM(p.OBJECTTYPE),'MATERIALIZED QUERY',' 
    '),'GLOBAL',' '),'DB2 PACKAGE','PACKAGE') ,'VIEW',' ')  || ' ' ||
    case WHEN      p.OBJECTTYPE IN ( 'TABLESPACE' ,  'SCHEMA'  ,  'WORKLOAD' )
           THEN ' '
           ELSE TRIM(p.OBJECTSCHEMA) || '.'
    END ||
    case WHEN      p.OBJECTTYPE IN ( 'SCHEMA' )
           THEN TRIM(p.OBJECTSCHEMA)
           ELSE TRIM(p.OBJECTNAME)
    END
           || ' TO ROLE DBPUBLI '
           || ' ; '
    from SYSIBMADM.PRIVILEGES p
    where authid = 'PUBLIC'
    ;
```

--- REVOKE FROM (INTERNAL GROUP) PUBLIC (FILE gen_revokes.sql)

```
---- Usage: db2 -xtf gen_revokes.sql > revokes.sql
----        db2 -tvf revokes.sql -z revokes.log
    SELECT 'REVOKE ' || p.PRIVILEGE ||
    case WHEN      p.OBJECTTYPE IN ( 'TABLESPACE' )
           THEN ' OF '
           ELSE ' ON '
    END ||
    case WHEN p.OBJECTTYPE IN ( 'PROCEDURE' ,  'FUNCTION' )  THEN ' SPECIFIC ' ELSE ' ' END
           || REPLACE(REPLACE(REPLACE(REPLACE(TRIM(p.OBJECTTYPE),'MATERIALIZED QUERY',' 
    '),'GLOBAL',' '),'DB2 PACKAGE','PACKAGE') ,'VIEW',' ')  || ' ' ||
    case WHEN      p.OBJECTTYPE IN ( 'TABLESPACE' ,  'SCHEMA'  ,  'WORKLOAD'  )
```

```
        THEN ' '
        ELSE TRIM(p.OBJECTSCHEMA) || '.'
END ||
case WHEN      p.OBJECTTYPE IN ( 'SCHEMA' )
        THEN TRIM(p.OBJECTSCHEMA)
        ELSE TRIM(p.OBJECTNAME)
END
        || ' FROM PUBLIC' ||
case WHEN p.OBJECTTYPE IN ( 'FUNCTION' , 'PROCEDURE' )  THEN ' RESTRICT ' ELSE ' ' END
        || ' ; '
from SYSIBMADM.PRIVILEGES p
where authid = 'PUBLIC'
;
```

--- GRANT ROLE DBPUBLI ON ALL EXISTING USERS (FILE gen_authdid.sql)

```
--- Usage: db2 -xtf gen_authids.sql > authids.sql
----       db2 -tvf authids.sql -z authids.log
    WITH temp1 ( authid) AS (
    SELECT DISTINCT AUTHID from SYSIBMADM.PRIVILEGES p where authid != 'PUBLIC' AND AUTHIDTYPE =
    'U' AND authid != current user )
    SELECT
    'GRANT ROLE DBPUBLI TO USER '
    || AUTHID
    FROM temp1
    ;
```

--- GRANT ROLE DBPUBLI ON DB AUTHOROITES (FILE gen_dbauth.sql)

```
---- Usage: db2 -xtf gen_dbauth.sql > dbauth.sql
----        db2 -tvf dbauth.sql -z dbauth.log

    SELECT    'GRANT CREATETAB ON DATABASE TO ROLE DBPUBLI ;'
    FROM     SYSCAT.DBAUTH
    WHERE GRANTEE = 'PUBLIC' AND CREATETABAUTH = 'Y';

    SELECT    'GRANT BINDADD ON DATABASE TO ROLE DBPUBLI ;'
    FROM    SYSCAT.DBAUTH
    WHERE    GRANTEE = 'PUBLIC' AND BINDADDAUTH = 'Y';
    …....
    SELECT    'REVOKE CREATETAB ON DATABASE FROM PUBLIC  ;'
    FROM    SYSCAT.DBAUTH
    WHERE    GRANTEE = 'PUBLIC' AND CREATETABAUTH = 'Y';

    SELECT    'REVOKE BINDADD ON DATABASE FROM PUBLIC  ;'
    FROM    SYSCAT.DBAUTH
    WHERE    GRANTEE = 'PUBLIC' AND BINDADDAUTH = 'Y';
```

Aufrufreihenfolge

Die Skripte erfordern eine bestehende Verbindung zur Datenbank:

- db2 -xtf gen_grants.sql > grants.sql
- db2 -xtf gen_revokes.sql > revokes.sql
- db2 -xtf gen_dbauth.sql > dbauth.sql
- db2 -xtf gen_authids.sql > authids.sql
- db2 -tvf grants.sql -z grants.log
- db2 -tvf dbauth.sql -z dbauth.log
- db2 -tvf authids.sql -z authids.log
- db2 -tvf revokes.sql -z revokes.log

Beispiel für DB2 10.1 auf SuSE Linux vmware:

```
> db2 connect to z05xat04

   Database Connection Information

 Database server        = DB2/LINUXX8664 10.1.4
 SQL authorization ID   = Z05XAT04
 Local database alias   = Z05XAT04

> db2 -xtf gen_grants.sql > grants.sql
> head grants.sql
CREATE ROLE DBPUBLI ;

GRANT USAGE ON  WORKLOAD  SYSDEFAULTUSERWORKLOAD TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.MON_INTERVAL_ID TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.ROUTINE_SPECIFIC_NAME TO ROLE DBPUBLI  ;
```

```
GRANT READ ON    VARIABLE SYSIBM.ROUTINE_TYPE TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.ROUTINE_MODULE TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.ROUTINE_SCHEMA TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.PACKAGE_NAME TO ROLE DBPUBLI  ;
GRANT READ ON    VARIABLE SYSIBM.PACKAGE_SCHEMA TO ROLE DBPUBLI  ;

> db2 -xtf gen_revokes.sql > revokes.sql
> head revokes.sql
REVOKE USAGE ON  WORKLOAD  SYSDEFAULTUSERWORKLOAD FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.MON_INTERVAL_ID FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.ROUTINE_SPECIFIC_NAME FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.ROUTINE_TYPE FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.ROUTINE_MODULE FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.ROUTINE_SCHEMA FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.PACKAGE_NAME FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.PACKAGE_SCHEMA FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.PACKAGE_VERSION FROM PUBLIC  ;
REVOKE READ ON    VARIABLE SYSIBM.CLIENT_IPADDR FROM PUBLIC  ;

> db2 -xtf gen_authids.sql > authids.sql
> head authids.sql
GRANT ROLE DBPUBLI TO USER DB2USER1

GRANT CREATETAB ON DATABASE TO ROLE DBPUBLI;
GRANT BINDADD ON DATABASE TO ROLE DBPUBLI;
GRANT CONNECT ON DATABASE TO ROLE DBPUBLI;
GRANT IMPLICIT_SCHEMA ON DATABASE TO ROLE DBPUBLI;

REVOKE CREATETAB ON DATABASE FROM PUBLIC ;
REVOKE BINDADD ON DATABASE FROM PUBLIC ;
REVOKE CONNECT ON DATABASE FROM PUBLIC ;
REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC ;
```

Welche Informationen sind für DB2 Admin hilfreich?

Erste Anlaufstelle sind sicherlich die DB2-Handbücher, die in mehrsprachiger Form, neben HTML auch im PDF-Format vorliegen:

- IBM DB2 10.5 Information Center for Linux, UNIX, and Windows
- DB2 Version 10.5 for Linux, UNIX, and Windows English manuals in PDF Format

Für einen regelmäßigen Austausch über alle Themen zu IBM DB2 LUW (LINUX, Windows, UNIX) lädt die gleichnamige XING-Gruppe ein. Diese Gruppe wurde am 08.05.2008 gegründet und hat inzwischen 640 Mitglieder. Über 300 Beiträge und Kommentare helfen bei fachlichen Problemen. Sie bietet in verschiedenen Rubriken auch eine gute Gelegenheit auf Projektangebote, Produktangebote oder Veranstaltungen hinzuweisen.

Eine gute Übersichtsseite für DB2 LUW Best Practicies Beschreibung (meist auch im PDF-Format) ist hier zu finden.

Wer sich zertifizieren lassen möchte dem ist das Vorbereitungsbuch und die Vorbereitungs-Artikelserie zu DB2 10.1 fundamentals certification exam 610 und DB2 10.1 DBA for LUW certification exam 611 empfohlen.

Um sein DB2-Produkt aktuell und sicher zu halten, sollte man die empfohlenen DB2 Fix Packs im Auge behalten. Ältere Fix Packs sind hier zu finden.

Weitere Infos zu Warnmeldungen, Produktankündigungen oder Downloads finden sich zentral unter dem IBM Support Portal - DB2 for Linux, UNIX und Windows.

Weitere hilfreiche Links, um auf dem Laufenden zu bleiben sind

- der RSS-Feed zu IBM developerWorks: Information Management
- Oder auch die Sammelseiten des DB2 Channel oder Planet DB2.

Etwas allgemeiner

Allgemeinere Informationen - zu allen IBM Information Management Produkten, also nicht nur DB2 LUW - finden sich unter:

- PDF-Redbook-Anleitungen zu Spezialthemen
- *HTML/PDF-Artikel zu Spezialthemen*

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

Eine Liste der anstehenden Konferenzen ist hier zu finden.

DB2 Regionale Usergroup

Das nächste Treffen der DeDUG (Deutsche DB2 Usergroup) findet am Freitag, den 17.10.2014 in Ehningen statt.

Die Teilnahme ist kostenlos aber eine Anmeldung ist erforderlich.

Unter diesem Link findet sich auch bereits die Agenda des Meetings.

DB2 LUW Arbeitsgruppentreffen

Das diesjährige Arbeitsgruppentreffen findet am 11./12. Dezember in Hamburg im Hotel Zollenspieker Fährhaus statt.
Zu Fragen und Anmeldungsformular bitte an den Chairman Nils Kaden wenden.

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 Pientka	MATERNA GmbH, Moderator der XING Gruppe für DB2 LUW; Frank.Pientka@materna.de Artikel: Welche Informationen sind für DB2 Admin hilfreich?
Michael Tiefenbacher	Leiter Business Services & Data Management Spezialist, ids-System GmbH; m.tiefenbacher@ids-system.de Artikel: STRING_UNITS ab DB2 10.5 FP4
Peter Schurr	IT Specialist DB2 / Dipl.-Inform. Med./ IBM Certified DB2 LUW 10.5, IBM SWG Services; Peter.Schurr@de.ibm.com Artikel: Bind nach Fixpack Upgrade auf einer Datenbank ohne PUBLIC Berechtigungen

Reviewer und Ideenlieferanten:

Dirk Fechner	IBM SWG
Peter Schurr	IBM SWG
Doreen Stein	