Manipulation leicht gemacht

Eigentümermigration über den Weg der InterBase/Firebird-Systemtabellen

Haben Sie sich schon einmal darüber geärgert, dass Sie mit dem „Power-User“ SYSDBA eine Datenbank samt den darin enthaltenen Datenbankobjekten erstellt haben, obwohl sich zu einem späteren Zeitpunkt herausgestellt hat, dass ein „normaler“ Benutzer besser gewesen wäre? Ja? Dann habe ich eine gute Nachricht für Sie. In diesem Artikel wird ein Verfahren vorgestellt, wie die Migration des Datenbank- und Datenbankobjekteigentümers über die direkte Manipulation der InterBase/Firebird-Systemtabellen durchgeführt werden kann, ohne dabei die Datenbank neu erstellen und bestehende Daten von einer Quell- in eine Zieldatenbank transferieren zu müssen.

von Thomas Steinmaurer

Bei der Erstellung einer InterBase/Firebird-Datenbank muss ein Benutzer angegeben werden, der in der Sicherheitsdatenbank, abhängig von der verwendeten InterBase/Firebird-Version in *isc4.gdb*, *ibadmin.ib* oder *security.fdb* vorhanden sein muss. Dieser Benutzer wird automatisch der Eigentümer der Datenbank, ausgestattet (neben SYSDBA) mit speziellen Rechten wie zum Beispiel der Erlaubnis zum Backup oder auch zum Herunterfahren (Shutdown) einer Datenbank. Auch Datenbankobjekte wie Tabellen, Stored Procedures und Roles haben einen Eigentümer, der nicht notwendigerweise der Eigentümer der Datenbank sein muss, da sich jeder vorhandene InterBase/Firebird-Benutzer mit einem gültigen Passwort an der Datenbank anmelden und Datenbankobjekte erstellen kann. Somit wird der aktuell verbundene Benutzer, wenn dieser eine CREATE TABLE-Anweisung ausführt, der Eigentümer der Tabelle. Die Unterscheidung Datenbank- versus Datenbankobjekteigentümer ist wichtig, da eine Änderung des Datenbankeigentümers über ein Backup/Restore nicht automatisch den oder die Eigentümer der Datenbankobjekte migriert. Diese Tatsache muss auch in dem hier vorgestellten Mechanismus zur Eigentümermigration Berücksichtigung finden.

Quellcode

Den Quellcode finden Sie auf der aktuellen Profi-CD sowie unter
www.derentwickler.de

SYSDBA und die Folgen

Jede InterBase/Firebird-Installation kommt mit einem speziellen Benutzer SYSDBA, der alle nur denkbaren Rechte in einer InterBase/Firebird-Umgebung besitzt. Dieser Benutzer wird in der Regel auch beim Anlegen einer Datenbank und der Datenbankobjekte verwendet, da es schlicht und einfach bequem ist (vor allem für Neueinsteiger), SYSDBA zu verwenden, weil man sich zu Beginn nicht unmittelbar mit dem Benutzermanagement in InterBase/Firebird auseinandersetzen muss. Handelt es sich um eine Datenbank, die ausschließlich im eigenen Unternehmen eingesetzt wird, dann stellt die Verwendung von SYSDBA in der Regel kein unmittelbares Problem dar. Diese Bequemlichkeit kann sich allerdings dann negativ auswirken, wenn man keine Kontrolle über den SYSDBA-Benutzer und dessen Passwort besitzt. So zum Beispiel bei einem Server, auf dem sich aufgrund einer anderen Drittherstelleranwendung bereits eine InterBase/Firebird-Installation befindet, oder auch wenn man die Datenbank bei einem Webhoster betreiben möchte, dieser das jedoch unter der Verwendung von SYSDBA untersagt. Diese Problematik sollte man sich immer vor Augen halten, wenn man eine Datenbank erstellt, da die Erstellung und die Verwendung eines eigenen InterBase/Firebird-Benutzers einen vernachlässigbaren Mehraufwand darstellt.

Sollte man sich nun in dieser ungünstigen Lage befinden, dass man eine bestehende Datenbank, die sich vielleicht auch bereits beim Kunden im Einsatz befindet, von einem Eigentümer SYSDBA zu einem anderen Eigentümer migrieren muss, dann hat man dazu im Wesentlichen zwei Wege, die eingeschlagen werden können:

- Aus der bestehenden Datenbank ein Metadatenskript mit dem alten Eigentümer extrahieren, dieses Skript mit dem

neuen Eigentümer ausführen und danach die Daten von der Quell- in die Zieldatenbank transferieren.
- Mit einer Kombination aus Backup/Restore-Zyklen und der direkten Manipulation der Systemtabellen kann der Eigentümer der Datenbank und der darin enthaltenen Datenbankobjekte von Benutzer A nach B migriert werden, ohne die eigentlichen Daten zum Schluss transferieren zu müssen.

Ist der erste Weg von Ihrer Seite aus machbar, dann sollten Sie diesen wählen, da dieser kaum Nebenwirkungen besitzt und frei verfügbare Tools für das Erstellen und Ausführen des Metadatenskripts (Kommandozeilentool *isql*) und dem Datentransfer (z.B. IBDataPump [1]) zur Verfügung stehen. Sollten Sie den sicheren Weg nicht gehen wollen, dann steht Ihnen noch die zweite Alternative zur Verfügung. Wie dieser Weg konkret aussieht, wird Ihnen der Rest des Artikels Schritt für Schritt vermitteln. Bevor wir jedoch mit der Vorstellung des eigentlichen Verfahrens für die Eigentümermigration beginnen, noch ein kurzer Exkurs in die Welt der InterBase/Firebird-Systemtabellen.

Systemtabelle	Bedeutung
RDB$PROCEDURES	Alle Stored Procedures
RDB$RELATIONS	Alle Tabellen und Views
RDB$ROLES	Alle Rollen
RDB$USER_PRIVILEGES	Alle SQL-Privilegien

Tabelle 1: Relevante Systemtabellen

Sicherheitsprivilegstyp	Bedeutung
A	ALL
S	SELECT
D	DELETE
I	INSERT
U	UPDATE
E	EXECUTE
R	REFERENCE
M	MEMBER OF (für Rollen)

Tabelle 2: Sicherheitsprivilegstypen

Per Anhalter durch die Systemtabellen

Systemtabellen in InterBase/Firebird sind spezielle Tabellen, deren Daten die Datenbankobjekte einer Datenbank beschreiben. Man spricht hierbei auch von Metadaten, also „Daten über Daten“. Systemtabellen weisen folgende Eigenschaften auf:

- Der Name einer Systemtabelle beginnt mit *RDB$*.
- Systemtabellen sind wie jede andere Tabelle über eine SELECT-Anweisung abfragbar.
- Direkte Änderungen an Systemtabellen über DML-Anweisungen (Delete, Insert oder Update) werden entweder von Systemtriggern unterbunden oder können ohne Einwand der Engine durchgeführt werden.

Eine komplette Übersicht über alle Systemtabellen finden Sie in der InterBase/Firebird-Dokumentation. Tabelle 1 gibt Ihnen einen Überblick über die in diesem Artikel benötigten Systemtabellen.

RDB$PROCEDURES, *RDB$RELATIONS* und *RDB$ROLES* besitzen je ein Feld *RDB$OWNER_NAME*, welches den Eigentümer des Datenbankobjekts speichert. *RDB$USER_PRIVILEGES* hat ein Feld *RDB$USER*, das den Benutzer speichert, für welchen ein SQL-Privileg vergeben wurde. Das Feld *RDB$GRANTOR* in der Tabelle *RDB$USER_PRIVILEGES* speichert jedoch den Benutzer, der das SQL-Privileg vergeben hat. *RDB$GRANT_OPTION* definiert, ob ein Benutzer selbst das Recht hat, Sicherheitsprivilegien für ein Datenbankobjekt zu vergeben. Ein weiteres Feld *RDB$PRIVILEGE* gibt den Typ des SQL-Privileges an. Einen Überblick über die möglichen Wertausprägungen gibt Tabelle 2.

Ausgangssituation

Mit dem Basiswissen über die wichtigsten Systemtabellen und deren Felder machen wir uns nun Schritt für Schritt auf den Migrationsweg der „Employee“-Beispieldatenbank, die mit jeder InterBase/Firebird installiert wird. Ich werde in diesem Beispiel Firebird 1.5.2 [2] verwenden. Für InterBase sollte das hier vorgestellte Verfahren 1:1 übertragbar sein. An Tools benötigen wir die Kommandozeilentools *gsec*, *gbak* und *isql*, die mit jeder InterBase/Firebird-Installation mitgeliefert werden. Wer zum Ausführen von SQL-Anweisungen etwas Komfortableres als *isql* verwenden möchte, hat unter [3] die Qual der Wahl: Freeware und/oder kommerzielle Produkte. Es sollte für jeden etwas dabei sein. Dann benötigen wir auch noch Delphi, da wir für die Migration der SQL-Privilegien eine einfache Anwendung schreiben werden. Die Aufgabenstellung lautet nun folgendermaßen: Der Eigentümer SYSDBA der *employee.fdb*-Beispieldatenbank soll auf einen Benutzer EMPLOYEE geändert werden, wobei alle vorhandenen SQL-Privilegien erhalten bleiben müssen und kein Datentransfer notwendig werden soll. Bevor Sie nun weitermachen, sollten Sie ein Backup der zu bearbeitenden Datenbank erstellen!

Schritt 1 – Benutzer-EMPLOYEE anlegen: Zu Beginn müssen wir einen Benutzer-EMPLOYEE anlegen. Dazu verwenden wir *gsec* wie folgt:

```
C:\Programme\Firebird\Firebird_1_5\
                bin>gsec -user sysdba -password masterkey
GSEC> add EMPLOYEE -pw employee
GSEC> quit
```

Schritt 2 – Backup mit SYSDBA und Restore mit EMPLOYEE: Um den Eigentümer der Datenbank von SYSDBA auf EMPLOYEE zu ändern, ist ein Backup mit SYSDBA

```
C:\Programme\Firebird\Firebird_1_5\
        bin>gbak -b -v -user sysdba -password masterkey
          localhost:c:\employee_migr\employee.fdb
                c:\employee_migr\employee.fbk
```

und ein Restore mit EMPLOYEE notwendig. Mit *gbak* ist dies ganz schnell erledigt:

```
C:\Programme\Firebird\Firebird_1_5\
    bin>gbak -c -v -user employee -password employee c:\
          employee_migr\employee.fbk localhost:c:\
            employee_migr\employee_pre_migr.fdb
```

Wie bereits erwähnt: Dieser Schritt ändert nicht den oder die Eigentümer von Datenbankobjekten, somit sind wir noch nicht am Ziel.

der entwickler

Alle Entwickler-Ausgaben im PDF-Format sowie Quellcodes und Beispiele von 2002 bis dato!

Wenn hier keine CD klebt, haben Sie die Entwickler PROFI-CD nicht abonniert und verpassen jede Menge Extras!

■ Testversionen:

Borland StarTeam 2005 Release 2

Aktuelle Version von Borlands Lösungen für das Anforderungs-, Konfigurations- und Change-Management StarTeam. Ausgerichtet auf die neue SDO-Vision kann StarTeam 2005 nun besser zwischen Teams an mehreren Standorten oder über mehrere Repositories eingesetzt werden. Eines der Features ist Search Server, welches dem Entwickler ermöglicht, Ad-hoc-Speicherabfragen innerhalb der Tools auszuführen, um Informationen und wieder verwendbaren Code zu suchen.

■ Microsoft .NET Framework:

.NET Framework 1.1 SDK & .NET Compact Framework 1.0 SP2, Mobile Internet Toolkit 1.0

Abonnieren ist jederzeit möglich über **www.derentwickler.de**
Sie können die Profi-CD auch einzeln zum Preis von 14,50 € (inkl. Versand) unter **www.derentwickler.de/proficd/** erwerben.

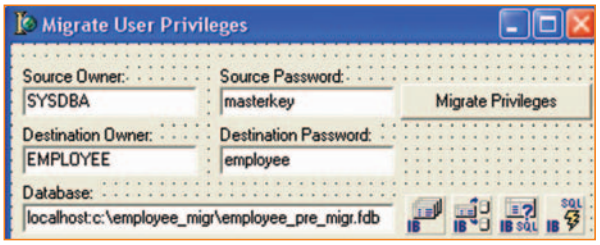

Abb. 1: Formular mit IBM-Komponenten für den Datenzugriff

Schritt 3 – Ermittlung der vorhandenen Eigentümer von Datenbankobjekten: Um zu ermitteln, welche Eigentümer von Datenbankobjekten es überhaupt gibt, verbindet man sich mit dem Eigentümer der Datenbank und führt folgende Systemtabellenabfrage aus:

```
C:\Programme\Firebird\Firebird_1_5\bin>isql -user
                    sysdba -password masterkey localhost:c:\
                      employee_migr\employee_pre_migr.fdb
Database: localhost:c:\employee_migr\
                          employee_pre_migr.fdb, User: sysdba
SQL> select rdb$owner_name from rdb$relations
CON> where rdb$system_flag = 0 or rdb$system_flag is null
CON> union
CON> select rdb$owner_name from rdb$procedures
CON> union
CON> select rdb$owner_name from rdb$roles
CON> ;

RDB$OWNER_NAME
==============================

SYSDBA

SQL> exit;
```

In unserem konkreten Beispiel mit *employee.fdb* wird hier nur ein Datensatz mit SYSDBA zurückgegeben. Diese Information ist wichtig, weil die beiden nachfolgenden Schritte mit allen vorhandenen Eigentümern durchgeführt werden müssen. Dies bedeutet:

- Diese InterBase/Firebird-Benutzer müssen angelegt werden, sofern sie noch nicht existieren.
- Man muss sich mit jedem Datenbankobjekteigentümer zur Datenbank verbinden und Schritt 4 sowie Teilaufgabe 1 von Schritt 5 ausführen.

Schritt 4 – Ändern des Eigentümers für Tabellen, Stored Procedures und Roles: Wir

Mehrere Schritte ergeben einen Weg

verbinden uns nun zur Datenbank *employee_pre_migr.fdb* unter Verwendung des alten Eigentümers SYSDBA:

```
C:\Programme\Firebird\Firebird_1_5\bin>isql -user sysdba
          -password masterkey localhost:c:\employee_migr\
                                  employee_pre_migr.fdb
Database:localhost:c:\employee_migr\employee_pre_migr.
                                  fdb,User: sysdba
SQL> update rdb$relations set rdb$owner_name =
          'EMPLOYEE' where rdb$owner_name = 'SYSDBA';
SQL> update rdb$procedures set rdb$owner_name =
          'EMPLOYEE' where rdb$owner_name = 'SYSDBA';
SQL> update rdb$roles set rdb$owner_name =
          'EMPLOYEE' where rdb$owner_name = 'SYSDBA';
SQL> commit;
SQL> exit;
```

Teilaufgabe	Realisierung
Dem neuen Eigentümer EMPLOYEE alle Rechte auf allen Datenbankobjekten einräumen	Delphi-Anwendung
Migration aller bestehenden SQL-Privilegien nach EMPLOYEE als Grantor	Delphi-Anwendung
Löschen aller SQL-Privilegien mit SYSDBA als Grantor	Delphi-Anwendung oder SQL-Anweisung
Löschen aller an SYSDBA vergebenen SQL-Privilegien	Delphi-Anwendung oder SQL-Anweisung

Tabelle 3: Teilaufgaben der Migration der SQL-Privilegien

Schritt 5 – Migration der SQL-Privilegien: Die Migration der SQL-Privilegien ist der etwas aufwendigere Teil. Dieser Schritt lässt sich nicht ausschließlich über einzelne SQL-Anweisungen realisieren, darum werden wir, wo notwendig, eine einfache Delphi-Anwendung mit der nötigen Programmlogik implementieren. Schritt 5 lässt sich in folgende Teilaufgaben unterteilen, die in Tabelle 3 dargestellt sind.

Teilaufgabe 1 und 2 werden über eine Delphi-Anwendung realisiert. Es handelt sich hierbei um ein einfaches Formular mit den notwendigsten IBX-Komponenten (*TIBDatabase, TIBTransaction, TIBQuery* und *TIBSQL*) für den Datenbankzugriff (Abb. 1). Die wichtigsten Methoden sind in Listing 1 (auf der Leser-CD) abgebildet.

Teilaufgabe 1 wird über die Methode *GrantPrivileges* realisiert. Die Datenbankverbindung erfolgt über den alten Eigentümer SYSDBA. Es wird eine Ergebnismenge von allen benutzerdefinierten Tabellen, Views und Stored Procedures durchlaufen, für jedes Datenbankobjekt werden alle bereits vorhandenen SQL-Privilegien für EMPLOYEE entfernt und erneut mit GRANT ALL hinzugefügt. So wird sichergestellt, dass EMPLOYEE alle Rechte auf eine Tabelle, View oder Stored Procedure besitzt.

Die zweite Teilaufgabe ist in der Methode *MigrateExistingPrivileges* implementiert. Die Datenbankverbindung erfolgt diesmal über den neuen Eigentümer EMPLOYEE. Es werden alle SQL-Privilegien in *RDB$USER_PRIVILEGES* selektiert, bei denen der alte Eigentümer SYSDBA als Grantor (*RDB$GRANTOR*) vorkommt. Die Ergebnismenge wird durchlaufen und, abhängig von den Werten in unterschiedlichen Feldern dieser Systemtabelle, es wird die entsprechende GRANT-Anweisung dynamisch zusammengesetzt und im Kontext des verbundenen Benutzers EMPLOYEE ausgeführt. Somit werden Duplikate der bereits vorhandenen SQL-Privilegien angelegt, allerdings mit EMPLOYEE als Grantor. Starten Sie nun die Anwendung und passen Sie den Datenbankpfad entsprechend an. Mit einem Klick auf den Button MIGRATE PRIVILEGES werden die ersten beiden Teilaufgaben abgearbeitet. Bei Teilaufgabe 3 handelt es sich um eine einzelne SQL-Anweisung:

```
C:\Programme\Firebird\Firebird_1_5\bin>isql -user
        sysdba -password masterkey localhost:c:\employee_
                                migr\employee_pre_migr.fdb
Database:localhost:c:\employee_migr\employee_pre_migr.
                                        fdb, User: sysdba
SQL> delete from rdb$user_
                        privileges where rdb$grantor = 'SYSDBA';
SQL> commit;
SQL> exit;
```

Diese kann ganz einfach wieder mit *isql* ausgeführt werden. Die Datenbankverbindung erfolgt diesmal mit dem alten Eigentümer SYSDBA!

Teilaufgabe 4 ist ebenfalls nur eine einzelne SQL-Anweisung. Diesmal ist es allerdings wichtig, dass die Datenbankverbindung mit dem neuen Eigentümer EMPLOYEE aufgebaut wird!

```
C:\Programme\Firebird\Firebird_1_5\bin>
        isql -user employee -password employee localhost:c:\
                        employee_migr\employee_pre_migr.fdb
Database:localhost:c:\employee_migr\employee_pre_migr.
                                        fdb, User: employee
SQL> delete from rdb$user_privileges where rdb$user =
                                                'SYSDBA';
SQL> commit;
SQL> exit;
```

Optionaler Schritt 6 – Anlegen einer Rolle „SYSDBA“: Es gibt einen eleganten Trick, um eine Verbindung zur Datenbank für einen bestimmten Benutzer zu unterbinden. Somit kann man sicherstellen, dass sich zum Beispiel der alte Eigentümer SYSDBA nicht zur Datenbank verbinden kann und irrtümlich Tabellen/Views, Stored Procedures oder Rollen anlegt, da hier der Eigentümer wieder SYSDBA lauten würde. Dieser Trick besteht darin, eine Rolle mit dem Namen SYSDBA anzulegen. Eine reguläre DDL-Anweisung CREATE ROLE SYSDBA wird fehlschlagen, darum werden wir die Rolle direkt über die *RDB$ROLES*-Systemtabelle mit EMPLOYEE als *RDB$OWNER_NAME* einfügen. Dieser Schritt ist optional. Sollten Sie ihn trotzdem durchführen wollen, dann ist Folgendes notwendig (die Verbindung zur Datenbank muss mit EMPLOYEE aufgebaut werden!).

```
C:\Programme\Firebird\Firebird_1_5\bin>isql -user
                employee -password employee localhost:c:\
                        employee_migr\employee_pre_migr.fdb
Database:localhost:c:\employee_migr\employee_pre_migr.
                                        fdb, User: employee
SQL> insert into rdb$roles (rdb$role_name, rdb$owner_
                        name) values ('SYSDBA', 'EMPLOYEE');
SQL> commit;
SQL> exit;
```

Schritt 7 – Backup/Restore mit EMPLOYEE: Nun sind wir beinahe am Ziel angelangt. Jetzt muss nur noch ein Backup

```
C:\Programme\Firebird\Firebird_1_5\bin>
    gbak -b -v -user employee -password employee localhost:
        c:\employee_migr\employee_pre_migr.fdb
                c:\employee_migr\employee_pre_migr.fbk
```

und Restore der Datenbank mit dem Benutzer EMPLOYEE durchgeführt werden.

```
C:\Programme\Firebird\Firebird_1_5\
    bin>gbak -c -v -user employee -password employee
    c:\employee_migr\employee_pre_migr.fbk localhost:
            c:\employee_migr\employee_final_migr.fdb
```

Schritt 8 – Abschließende Tests: Abschließend sollten Sie natürlich noch etwaige Tests mit Ihrer Anwendung durchführen. Dazu gehört der Zugriff auf Tabellen/Views und Stored Procedures, das Hinzufügen und Entfernen von Tabellen und Tabellenfeldern sowie die Vergabe und das Entfernen von SQL-Privilegien für andere Benutzer. Sollten Sie Schritt 6 durchgeführt haben, dann versuchen Sie einfach, sich mit SYSDBA an der Datenbank anzumelden.

Fazit

Man kann es nicht oft genug sagen: Finger weg von der direkten Manipulation der Systemtabellen, wenn man sich nicht im Klaren ist, welche negativen Auswirkungen solche Änderungen zur Folge haben. Man kann sich allerdings auch sehr viel Zeit ersparen, wenn einem der Umgang mit den Systemtabellen vertraut ist. Das hier vorgestellte Verfahren für die Eigentümermigration in InterBase/Firebird wurde bereits von einer Vielzahl von Benutzern in der Praxis (in Form des frei verfügbaren Tools „FBOwnerMigrator“) erfolgreich angewendet. Sollte Ihnen der hier vorgestellte Weg zu mühsam sein, dann fordern Sie einfach per E-Mail an mich (ts@iblogmanager.com) den Download-Link dieses Tools an. Warum dieses Tool nicht als Download für jedermann zur Verfügung steht? Ich denke, ich habe meine Bedenken über die direkte Manipulation von Systemtabellen bereits geäußert, darum möchte ich zumindest einen Überblick darüber haben, wer dieses Tool einsetzt.

Links & Literatur

[1] clevercomponents.com/products/datapump/ibdatapump.asp
[2] www.firebirdproject.org
[3] www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp_admin_tools

Anzeige