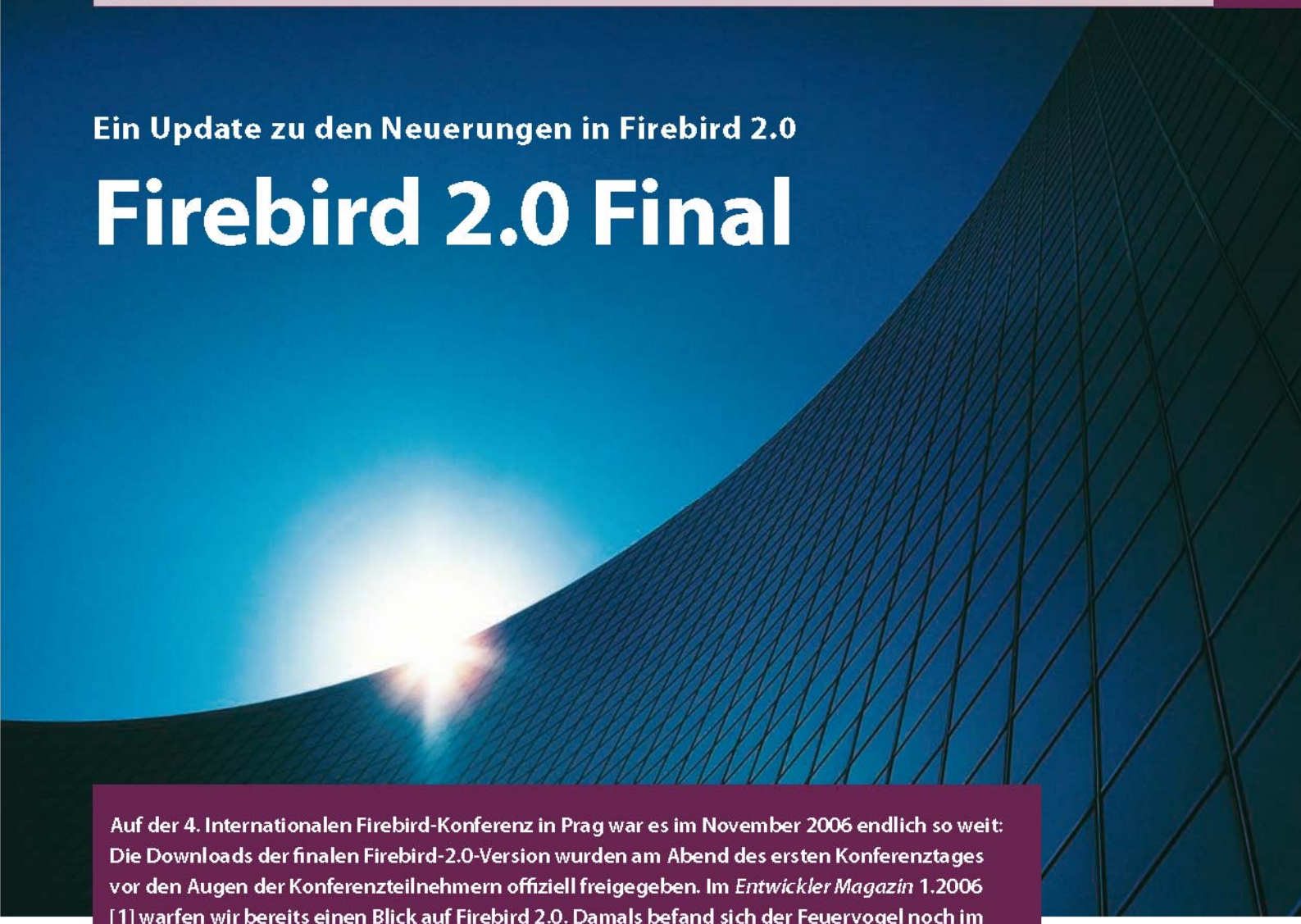


Ein Update zu den Neuerungen in Firebird 2.0

Firebird 2.0 Final

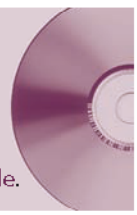
Auf der 4. Internationalen Firebird-Konferenz in Prag war es im November 2006 endlich so weit: Die Downloads der finalen Firebird-2.0-Version wurden am Abend des ersten Konferenztages vor den Augen der Konferenzteilnehmern offiziell freigegeben. Im *Entwickler Magazin* 1.2006 [1] warfen wir bereits einen Blick auf Firebird 2.0. Damals befand sich der Feuervogel noch im Alpha-Stadium. Bereits zu diesem Zeitpunkt wurde anhand der Fülle an Neuerungen jedoch deutlich, dass dieses Release ein wichtiger Meilenstein für das Firebird-Projekt sein würde.

von Thomas Steinmaurer

Im erwähnten Artikel [1] wurde ein Blick hinter die Kulissen der Entwicklung von Firebird 2.0 gewagt, als sie sich noch im Alpha-Stadium befand. Trotz des frühen Entwicklungszeitpunktes hatte man bereits ein einigermaßen stabiles DBMS in Händen, um ernsthafte Tests durchführen zu können. Dies machte Lust auf neue Alpha/Beta/RC-Releases, um mit weiteren Neuerungen zu experimentieren. Würde es nach einem bekannten Digitalkamera-Hersteller gehen, wäre der Werbeslogan „Komm spielen" für diese Testphase passend gewesen.

Quellcode

Der Quellcode zum Artikel befindet sich auf der CD und auf www.entwickler-magazin.de.



Komm spielen

Jedes Testrelease wurde von der Community dankend angenommen, heruntergeladen, installiert und intensiven Tests unterzogen. Dieser sehr breit gefächerte Field-Test, der eine nicht zu unterschätzende Eigenschaft eines akzeptierten Open-Source-Projektes ist, war ein reiner Erfolgsfaktor. Zusammen mit einer Testbench, bestehend aus einer Fülle an systematischen Tests, die jedes Release durchlaufen muss, war dies ein Garant für ein stabiles Release, das letztlich im November 2006 veröffentlicht wurde.

Bei Firebird handelt es sich allerdings nicht um ein Spielzeug, sondern um ein ausgereiftes Open-Source-DBMS, das mit den Neuerungen in Version 2.0 den kommerziellen Produkten wieder etwas näher gerückt ist. Vor allem für Klein- und Mittelbetriebe stellt es eine interessante Alternative dar, wenn geringe oder keine Lizenzkosten einen nicht unwesentlichen Wettbewerbsfaktor darstellen. Wer will schon mehr bezahlen als nötig? Auch wenn man kostenfreie Software gerne

annimmt, sollte man sich spätestens hier ins Gewissen reden (sonst tun es vielleicht noch andere), ob man nicht Teile der erzielten Einsparungen dem Firebird-Projekt zugute kommen lässt. Ein sehr guter Kanal hierfür ist die Firebird Foundation [2]. Wenn man neben den Einsparungen der Lizenzkosten außerdem den klassischen „Schuss mit der Kanone auf einen Spatz" verhindern kann, indem man für einen bestimmten Einsatzzweck kein großes kommerzielles DBMS benötigt, weil dies nicht notwendig ist, sondern ein frei verfügbares, schlankes, mit wenig Administration verbundenes Produkt einsetzt, dann ist doch alles bestens! So nebenbei erspart man sich auch die Personalkosten für einen Vollzeit-Datenbankadministrator. Aus eigener Erfahrung kann ich sagen, dass die Migration zu einem Open-Source-DBMS in letzter Zeit sehr „in Mode" gekommen ist. Die Dienstleistungsbranche nimmt es dankend an.

Internationalisierung

Die Unterstützung für unterschiedliche Zeichensätze und Sortierreihenfolgen (Collations) war in Firebird schon immer vorhanden. In Firebird 2.0 wurde dieser Bereich komplett überarbeitet. Dadurch erhält man nicht nur eine verbesserte Unicode-Unterstützung in Form des UTF-8-Zeichensatzes, als Ersatz für *UNICODE_FSS*. Auch die Konvertierung des Zeichensatzes der Datenbank mit der Client-Anwendung erfolgt nun immer über Unicode, außer bei den Zeichensätzen *NONE* bzw. *OCTSETS*. Hier erfolgt keine Konvertierung, sondern die Bytes werden nur kopiert.

Viele Entwickler werden es schätzen, dass die Funktion *UPPER()* nun auch Umlaute bei der Verwendung einer 1-Byte-Sortierreihenfolge korrekt umwandelt. Somit braucht sich der Entwickler keine Gedanken mehr darüber zu machen, dass eine Multi-Byte-Sortierreihenfolge (zum Beispiel *DE_DE* für den Zeichensatz *ISO8859_1*) auf Feldebene verwendet werden muss, um Umlaute korrekt in Großbuchstaben konvertieren zu können. Listing 1 verdeutlicht den Unterschied zwischen Firebird 1.x und Firebird 2.0 anhand eines einfachen Beispiels.

Das Feld V1 wird mit dem Zeichensatz ISO8859_1 erstellt, jedoch ohne Sortierreihenfolge. Es wird somit implizit eine binäre 1-Byte-Sortierreihenfolge verwendet. Das Feld *V2_DE_DE* verwendet denselben Zeichensatz, allerdings mit einer Multi-Byte-Sortierreihenfolge *DE_DE*, die für den verwendeten Zeichensatz ISO8859_1 definiert ist. Wird die *SELECT*-Anweisung mit Firebird 1.5 ausgeführt, so erhält man in *UPPER_V1* Kleinbuchstaben. Ein Ergebnis, das man so nicht erwartet hätte. Dieser Umstand ist vor allem für Firebird-Entwickler irritierend, die sich mit Zeichensätzen und Sortierreihenfolgen in Firebird noch nicht auseinandergesetzt haben. Mit einem Trick, der darin besteht, die *COLLATE DE_DE*-Klausel beim *UPPER* mitanzugeben, erhält man wiederum großgeschriebene Umlaute. *UPPER_V2_DE_DE* gibt das richtige Ergebnis ohne zusätzliche Maßnahmen zurück, da das Feld *V2_DE_DE* bereits mit der Sortierreihenfolge *DE_DE* angelegt wurde. In Firebird 2.0 wurde dies nun dahingehend geändert, dass ein *UPPER()* auch mit einer 1-Byte-Sortierreihenfolge korrekt funktioniert. Eine willkommene Änderung, nicht nur für Firebird-Neueinsteiger. Wo vormals noch die Verwendung einer UDF notwendig war, kann man nun eine der neuen, in der Datenbank-Engine

Listing 1

UPPER() - Firebird 1.x vs. Firebird 2.0

```
CREATE TABLE T1 (
 V1 VARCHAR(10) CHARACTER SET ISO8859_1
 , V2_DE_DE VARCHAR(10) CHARACTER SET ISO8859_
                                    1 COLLATE DE_DE
);
COMMIT;

INSERT INTO T1 (V1, V2_DE_DE) VALUES ('äöü', 'äöü');
COMMIT;

SELECT
 V1
 , UPPER(V1) UPPER_V1
 , UPPER(V1 COLLATE DE_DE) UPPER_V1_DE_DE
 , V2_DE_DE
 , UPPER(V2_DE_DE) UPPER_V2_DE_DE
FROM
 T1;

--Ergebnis in Firebird 1.5

V1    UPPER_V1 UPPER_V1_DE_DE V2_DE_DE
                                    UPPER_V2_DE_DE
------------------------------------------------------------
äöü   äöü   ÄÖÜ    äöü   ÄÖÜ


--Ergebnis in Firebird 2.0

V1    UPPER_V1 UPPER_V1_DE_DE V2_DE_DE
                                    UPPER_V2_DE_DE
------------------------------------------------------------
äöü   ÄÖÜ   ÄÖÜ    äöü   ÄÖÜ
```

News kompakt

Sollten Sie den bereits genannten Artikel zu den Neuerungen in Firebird 2.0 nicht gelesen haben, erlauben Sie mir, die wichtigsten Neuerungen kompakt zusammenzufassen:

- Neue On-Disc-Structure (ODS) in der Version 11, die unter anderem mit einschränkenden Limitierungen im Bereich der Indizes und der maximalen Tabellengröße aufräumt
- Neuerliche Verbesserung des Optimizers
- Neue Strategie für die Garbage Collection in der Super-Server-Architektur
- Eine Fülle an Spracherweiterungen wie Derived Tables, Expression Indizes, CROSS JOINs, verbesserte Typermittlung bei UNIONs und vieles mehr
- Verbindungs- bzw. transaktionsweite benutzerdefinierte Kontextvariablen
- Inkrementelles Backup während des Betriebs
- Stetige Weiterentwicklung der ODBC-, JDBC- und .NET-Treiber sowie dem frei verfügbaren Administrationswerkzeug FlameRobin [3]

Alle hier genannten Punkte sind natürlich auch in der finalen Version von Firebird 2.0 enthalten. Unter Umständen wurden dabei noch Optimierungen und Bugfixes vorgenommen. Version 2.0 hat allerdings noch viel mehr zu bieten! Dies lässt sich schon erahnen, wenn man die Release Notes öffnet und einen Schmöker von 162 Seiten vor sich hat.

integrierten Funktionen *TRIM*, *LOWER*, *BIT_LENGTH*, *CHAR_LENGTH* und *OCTET_LENGTH* verwenden.

EXECUTE BLOCK-Anweisung

PSQL (Prozedural SQL) ist die Sprache, um serverseitigen Code in Form von Stored Procedures bzw. Triggern zu entwickeln. Eine interessante Neuerung für Entwickler ist die *EXECUTE BLOCK*-Anweisung, die die Möglichkeiten von PSQL zum Client transferiert. Somit muss für das Ausführen von PSQL-Code nicht mehr notwendigerweise eine Stored Procedure erstellt werden, die vom Client aufgerufen wird. In *EXECUTE BLOCK* steht der vollständige PSQL-Sprachumfang zur Verfügung, zum Beispiel Kontrollstrukturen, Cursordeklarationen, *SUSPEND*, Aufruf von anderen Stored Procedures und vieles mehr. Die *EXECUTE BLOCK*-Syntax sehen Sie in Listing 2.

Ein einfaches Beispiel in Listing 3 befüllt eine Tabelle mit 1000 Datensätzen mit clientseitigem PSQL.

Ein weiteres *EXECUTE BLOCK*-Beispiel in Listing 4 veranschaulicht, wie über eine *FOR SELECT*-Schleife jeder zweite Datensatz an das aufrufende Programm zurückgegeben werden kann.

Listing 2

EXECUTE BLOCK-Syntax

```
EXECUTE BLOCK [ (param datatype= ?, param
                                    datatype = ?, ...) ]
[RETURNS (param datatype, param datatype, ...) ]
AS
[DECLARE VARIABLE var datatype; ...]
BEGIN
...
END
```

Listing 3

Befüllung über EXECUTE BLOCK

```
CREATE TABLE T2 (
 ZAEHLER INTEGER NOT NULL
 , CONSTRAINT PK_T2_ZAEHLER PRIMARY KEY (ZAEHLER)
);

COMMIT;

SET TERM !!;

EXECUTE BLOCK
AS
DECLARE vZaehler INTEGER;
BEGIN
 vZaehler = 1;
 WHILE (vZaehler <= 1000) DO
 BEGIN
  INSERT INTO T2 (ZAEHLER) VALUES (:vZaehler);
  vZaehler = vZaehler + 1;
 END
END
!!

SET TERM ;!!

COMMIT;
```

Erstellen von FOREIGN KEY Constraints

Eine nicht unwesentliche Verbesserung beim Anlegen eines *FOREIGN KEY* Constraints geht in den Release Notes fast unter: Wo in früheren Firebird-Versionen für das Erstellen eines *FOREIGN KEY* Constraints noch ein exklusiver Zugriff auf die Datenbank benötigt wurde, ist dies in Firebird 2.0 nicht mehr notwendig. Zu Testzwecken führen Sie das Skript in Listing 5 in einer Datenbank aus, wo andere aktive Verbindungen zur Datenbank bestehen. In Firebird 1.x kommt beim

Listing 4

Clientseitige Programmlogik über EXECUTE BLOCK

```
SET TERM !!;

EXECUTE BLOCK
 RETURNS (rZaehler INTEGER)
AS
DECLARE vMod INTEGER;
BEGIN
 vMod = 0;
 FOR SELECT ZAEHLER FROM T2 INTO :rZaehler DO
 BEGIN
  vMod = vMod + 1;
  IF (vMod = 2) THEN
  BEGIN
   SUSPEND;
   vMod = 0;
  END
 END
END
!!

SET TERM ;!!
```

COMMIT nach dem Anlegen des Constraints die Fehlermeldung „unsuccessful metadata update, object T3 is in use". In Firebird 2.0 läuft dieses Skript durch, auch wenn andere Datenbankverbindungen existieren.

Ich stimme Ihnen zu, wenn Sie sagen, dass in Produktionsdatenbanken nicht willkürlich Änderungen im Modell vorgenommen werden sollten, sondern hierfür einen Upgrade-Plan mit einem Wartungsfenster existieren sollte. Die hier diskutierte Verbesserung ist aus meiner Sicht vor allem für Entwicklungsdatenbanken sehr nützlich, wenn mehrere Entwickler auf ein und dieselbe Entwicklungsdatenbank zugreifen und dabei Modelländerungen regelmäßiger vorkommen, als das bei Produktionsdatenbanken der Fall ist.

Inkrementelles Online-Backup

Firebird 2.0 kommt mit der neuen Backup-Technologie *NBackup*, die inkrementelle Sicherungen ermöglicht. Dies ist vor allem für sehr große Datenbanken im Gigabyte-Bereich eine interessante Neuerung, da sich bei solchen Installationen die Verwendung von *gbak* sehr zeit- und ressourcenintensiv gestalten kann. Es steht sowohl das altbekannte *gbak* als auch die neue *NBackup*-Technologie für die Durchführung von Datenbanksicherungen zur Verfügung. Diese beiden Ansätze ersetzen sich nicht gegenseitig, sondern ergänzen sich vielmehr. Ein wesentlicher Unterschied besteht vor allem darin, dass *gbak* die zu sichernden Daten, auf einer logischen Ebene verarbeitet, wohingegen *NBackup* nur ganze Datenseiten und nicht einzelne Datensätze der Datenbank verarbeitet. Man spricht bei *NBackup* daher von einem Page-Level-orientierten Backup-Tool, das auch die Möglichkeit bietet, inkrementelle Sicherungen anzufertigen. Bei *gbak* geschieht eine Sicherung immer nach dem „Alles-oder-Nichts-Prinzip". Beide Tools haben eines gemein: Eine Sicherung kann während des Betriebs, das heißt, wenn aktive Verbindungen zur Datenbank bestehen, erstellt werden. Tabelle 1 gibt einen Überblick über die wesentlichen Charakteristika der beiden Sicherungsmöglichkeiten in Firebird 2.0.

Listing 5

Skript zum Erstellen eines FOREIGN KEY Constraints

```
CREATE TABLE T3 (
 T3_ID INTEGER NOT NULL
 , CONSTRAINT PK_T3_ID PRIMARY KEY (T3_ID)
);

COMMIT;

CREATE TABLE T4 (
 T4_ID INTEGER NOT NULL
 , T3_ID INTEGER NOT NULL
 , CONSTRAINT PK_T4_ID PRIMARY KEY (T4_ID)
);

COMMIT;

ALTER TABLE T4 ADD CONSTRAINT FK_T3_ID
          FOREIGN KEY (T3_ID) REFERENCES T3 (T3_ID);

COMMIT;
```

Die Syntax für die Erstellung einer Sicherung (-B) der Datenbank mit *NBackup* sieht folgendermaßen aus:

```
nbackup [-u <benutzer> -p<passwort>] -B <L>
                    <datenbankdatei> [<sicherungsdatei>]
```

Mit dem Backup-Level <L> wird die Art der Sicherung bestimmt. Der Wert 0 definiert eine vollständige Sicherung. Bei Werten größer 0 handelt es sich um eine inkrementelle Sicherung mit der Basissicherung N minus 1. Die Angabe der Sicherungsdatei ist optional. Wird der Dateiname der Sicherungsdatei nicht angegeben, generiert *NBackup* einen Dateinamen der Form *<datenbankdatei>-<L>-<zeitstempel>.nbk*. Ein Beispiel soll die Verwendung von *NBackup* verdeutlichen.

Es kann z.B. wöchentlich eine vollständige Sicherung der Datenbank *entwickler_fb2.fdb* wie folgt durchgeführt werden:

```
nbackup.exe -u sysdba -p masterkey -B 0 entwickler_fb2.
                    fdb entwickler_fb2_full_0.nbk
```

Damit wird täglich eine Level-1-Sicherung erstellt, die alle Änderungen seit der letzten vollständigen Sicherung beinhaltet. Die daraus resultierende Sicherungsdatei wird gegenüber der letzten vollständigen Sicherung verhältnismäßig kleiner ausfallen, abhängig davon, wie viele Änderungen in der Zwischenzeit in der Datenbank erfolgt sind. Die Erstellung der inkrementellen Sicherung erfolgt mit:

```
nbackup.exe -u sysdba -p masterkey -B 1 entwickler_fb2.
                    fdb entwickler_fb2_inkr_1.nbk
```

Wird die Level-1-Sicherung neu erstellt, dann handelt es sich bei der Sicherungsdatei nicht um das Inkrement der letzten Level 1, sondern der letzten Level-0-Sicherung. Somit beinhaltet die zuletzt erstellte Level-1-Sicherung immer alle Änderungen seit der letzten vollständigen Sicherung. Alle paar Stunden kann noch eine Level-2-Sicherung erstellt werden, die die Differenz zwischen der letzten Level 1 Sicherung und dem aktuellen Stand der Datenbank darstellt. Dies erfolgt beispielhaft mit:

```
nbackup.exe -u sysdba -p masterkey -B 2 entwickler_fb2.
                    fdb entwickler_fb2_inkr_2.nbk
```

Soll nun eine Datenbank aus den erstellten inkrementellen Sicherungen wieder zurückgesichert werden, ist es notwendig, dass die vollständige Kette der Sicherungsdateien (aufsteigend nach dem Backup-Level) angegeben wird. Für unser Beispiel kann die Wiederherstellung wie folgt aussehen:

```
nbackup.exe -u sysdba -p masterkey -r entwickler_fb2_neu.
    fdb entwickler_fb2_full_0.nbk entwickler_fb2_inkr_1.
                    nbk entwickler_fb2_inkr_2.nbk
```

Wird die vollständige Kette der Sicherungsdateien nicht angegeben, oder handelt es sich um die falsche Reihenfolge, dann erkennt *NBackup* dies und bricht die Wiederherstellung mit einer Fehlermeldung ab.

In Firebird 2.0 ist es auch erstmals möglich, eine Datenbankdatei in einen

> Firebird 2.0 kommt mit der neuen Backup-Technologie *NBackup*

Modus zu versetzen, ohne den Betrieb dabei zu unterbrechen, der den sicheren Zugriff für Operationen auf Dateisystemebene (z.B. Kopieren, Packen, Zugriff durch andere Sicherungssoftware) erlaubt. Dies erfolgt ebenfalls mit *NBackup* und dem –L-(Lock)Kommandozeilenschalter. Ein Szenario für die Erstellung einer ZIP-Datei einer Datenbank, die in Verwendung ist, sieht wie folgt aus:

- Sperren der Datenbankdatei mit: *nbackup.exe -u sysdba -p masterkey -L entwickler_fb2.fdb*
- Erstellen der ZIP-Datei. Währenddessen kann mit der Datenbank weiterhin gearbeitet werden. Alle Änderungen seit der Sperre werden in eine Differenzdatei geschrieben, die beim Entsperren mit der Datenbankdatei wieder verschmolzen wird
- Entsperren der Datenbankdatei mit: *nbackup.exe -u sysdba -p masterkey -N entwickler_fb2.fdb*

Sollten Sie noch mehr über *NBackup* erfahren wollen, werfen Sie bitte einen Blick in die *NBackup*-Dokumentation, die auch auf Deutsch verfügbar ist [4].

Kompatibilität

Bei einem Umstieg von Firebird 1.x auf Firebird 2.0 ist es wichtig zu wissen, dass bestimmte Kompatibilitätsprobleme auftreten können. So ist die Benutzerdatenbank von Firebird 1.x nicht direkt mit Firebird 2.0 verwendbar. Eine existierende Firebird-1.x-Benutzerdatenbank kann allerdings sehr einfach aktualisiert werden, um mit Firebird 2.0 verwendet werden zu können. Das notwendige SQL-Skript finden Sie im Unterverzeichnis *\misc\upgrade\security* ihrer Firebird-2.0-Installation. Eine neue Sicherheitsmaßnahme in Firebird 2.0 in Bezug auf die Benutzerdatenbank ist, dass keine direkten Verbindungen mehr zulässig sind. Benutzer können nur mehr mit *gsec* oder einem anderen Tool, das das Services-API verwendet, verwaltet werden. Benutzer können nun allerdings ihr eigenes Passwort ändern. Eine aus meiner Sicht willkommene Inkompatibilität (so etwas soll es auch geben) ist die Entschärfung des gefährlichen –R-Kommandozeilenschalters von *gbak* für die Wiederherstellung einer Sicherung. In Firebird 1.x stand diese Option genau genommen für „Replace“, nicht für ein ungefährliches „Restore“ und überschrieb somit eine existierende Datenbank. Dies wurde in Firebird 2.0 entschärft. Auch im Bereich von SQL existiert nun eine Stolperfalle, da in Firebird 2.0 Feldnamen mit einem Tabellenalias und nicht mehr mit dem tatsächlichen Tabellennamen verwendet werden müssen, sofern für die Tabelle ein Tabellenalias angegeben wurde. Folgende Anweisung ist unter Firebird 2.0 nicht mehr funktionsfähig: *SELECT T3.T3_ID FROM T3 T*. Dies trifft vor allem Benutzer von IBObjects [5], da IBObjects Benutzerabfragen im Hintergrund parst und teilweise in eine Form umstrukturiert, die einer für Firebird 2.0 ungültigen Abfrage entspricht. Erst mit der aktuellen Version 4.7 Build 16 ist man hier mit Firebird 2.0 auf der sicheren Seite. IBObjects-Benutzer sollten dies bei einem Umstieg berücksichtigen. Weitere Informationen zu möglichen Inkompatibilitäten finden Sie in *\doc\README.incompatibilities.txt*.

Funktionalität	gbak	NBackup
Online (während aktiver Verbindungen)	Ja	Ja
Verkleinern der Datenbank	Ja	Nein
Durchführen eines Sweeps	Ja	Nein
Ändern des Datenbankeigentümers	Ja	Nein
Transportables Backup	Ja	Nein
Inkrementelles Backup	Nein	Ja
Logisch vs. Page-Level	Logisch	Page
Unterstützung von Mehrdateiendatenbanken	Ja	Derzeit nicht
Backup von Remote-Datenbanken	Ja	Nein
Exklusive Sperre der Datenbankdatei zur sicheren Erstellung einer Kopie auf Dateisystemebene	Nein	Ja

Tabelle 1: gbak vs. NBackup

Firebird und Windows Vista

Für den Betrieb von Firebird unter Windows Vista gibt es nur einen Punkt zu berücksichtigen: Die Verbindung zu einer Datenbank über das lokale XNET-Protokoll kann unter Umständen fehlschlagen. Stellen Sie deshalb sicher, dass immer eine TCP/IP-Verbindung verwendet wird, auch wenn es sich um eine lokale Datenbank handelt. Ich würde Ihnen generell die Verwendung einer TCP/IP-Verwendung empfehlen, weil man hiermit etlichen Problemen von vornherein aus dem Weg gehen kann. So kann zum Beispiel aus einem Dienst heraus in der Regel nur über eine TCP/IP-Verbindung auf eine Datenbank zugegriffen werden. Des Weiteren ist das lokale Protokoll der Clientbibliothek von Firebird 1.x mit dem Firebird 2.0 Server (und umgekehrt) nicht mehr kompatibel. Für multi-threaded Anwendungen, die über die Firebird-Clientbibliothek auf eine Datenbank zugreifen, ist eine TCP/IP-Verbindung ebenfalls die einzige Option.

Ausblick auf Firebird 2.1

Neben einem Bugfix-Release 2.0.1 wird bereits seit geraumer Zeit an Version 2.1 gearbeitet. Einige der zu erwartenden Neuigkeiten sind: Monitoring Tables, Datenbank-Trigger (CONNECT, DISCONNECT, TRANSACTION START, TRANSACTION COMMIT, TRANSACTION ROLLBACK), Common Table Expressions, Verwendung von Domains in PSQL und Global Temporary Tables. Eine erste Alpha-Version von Firebird 2.1 soll schon bald erhältlich sein. Vielleicht beschäftigen wir uns mit diesen Neuerungen in einer der nächsten Ausgaben. Hiermit schließt sich der Kreis. Komm spielen!

Links & Literatur

[1] www.entwickler-magazin.de/zonen/magazine/psecom,id,8,online,783,p,0.html

[2] www.firebirdsql.org/index.php?op=ffoundation&id=contributions

[3] www.flamerobin.org

[4] www.firebirdsql.org/pdfmanual/de/Firebird-nbackup-de.pdf

[5] www.ibobjects.com