Firebird Backup & Restore-Werkzeug

Norman Dunbar, Martin Köditz

Version 1.12-de, 28. Juli 2020

Inhaltsverzeichnis

Kapitel 1. Einleitung

`Gbak` ist eines der mit Firebird gelieferten Dienstprogramme zum Sichern und Wiederherstellen von Datenbanken. In Firebird 1.5 ist es das einzige mitgelieferte Dienstprogramm dieser Art, während Firebird 2.x auch das in einem anderen Dokument beschriebene Dienstprogramm `nbackup` enthält.

In diesem Handbuch werden wir folgendes diskutieren:

- Befehlszeilenoptionen für `gbak`.
- `gbak` Befehle und ihre Parameter.
- Ausführen von `gbak` im Sicherungs- oder Wiederherstellungsmodus.
- Einige Hinweise, Fallstricke und Schwächen von `gbak`.

Kapitel 2. Befehlszeilenoptionen

2.1. Allgemeine Optionen

Wenn Sie `gbak` im Sicherungs- oder Wiederherstellungsmodus ausführen, gibt es eine Reihe von Optionen, die für beide Modi gelten. Diese sind:

`-?`

Dieser Schalter zeigt die Befehlszeilenoptionen und -schalter an. Es ersetzt die alte Methode, bei der Sie einen ungültigen Schalter angeben mussten (siehe `-help` unten), um die Liste der gültigen Schalter zu sehen.

HINWEIS: Ab Firebird 2.5.

`-FE[TCH_PASSWORD] <password file name> | stdin | /dev/tty`

Dieser Schalter bewirkt, dass das Kennwort für den entsprechenden Benutzer aus einer Datei gelesen wird und nicht in der Befehlszeile angegeben wird. Der angegebene Dateiname steht nicht in Anführungszeichen und muss für den Benutzer lesbar sein, der `gbak` ausführt. Wenn der Dateiname als `stdin` angegeben ist, wird der Benutzer zur Eingabe eines Kennworts aufgefordert. Auf POSIX-Systemen führt der Dateiname `/dev/tty` auch zu einer Aufforderung zur Eingabe des Kennworts.

HINWEIS: Ab Firebird 2.5.

`-M[ETA_DATA]`

Dieser Schalter bewirkt, dass Ihre Daten ignoriert und nicht gesichert oder wiederhergestellt werden. Bei einer Sicherung werden nur die Datenbank-Metadaten gesichert. Bei dieser Verwendung einer Wiederherstellung werden keine Daten in der Speicherauszugsdatei wiederhergestellt. Dieser Schalter kann verwendet werden, wenn Datenbankklone erstellt werden, die nur die `Tabellen`, `Indizes` usw., aber keine der Daten enthalten müssen.

`-PAS[SWORD] <Kennwort>`

Liefert das Passwort für den oben angegebenen Benutzernamen. Dies muss nicht angegeben werden, wenn die Umgebungsvariable `ISC_PASSWORD` vorhanden ist und den richtigen Wert hat.

`-RO[LE] <SQL Rollenname>`

Ermöglicht die Angabe der Rolle, die vom verbindenden Benutzer verwendet werden soll. Nicht sehr praktisch und wird in der Praxis normalerweise nicht verwendet.

`-U[SER] <Benutzername>`

Ermöglicht die Angabe des Benutzernamens des SYSDBA- oder Datenbankbesitzer-Benutzers, wenn die Datenbank gesichert werden soll, oder im Falle einer Wiederherstellung (mit dem angegebenen Schalter `-c[reate]`) kann ein beliebiger gültiger Benutzername angegeben werden. Dies muss nicht angegeben werden, wenn die Umgebungsvariable `ISC_USER` vorhanden ist und einen korrekten Wert für den Benutzernamen hat.

Datenbanken können nur von SYSDBA oder dem Datenbankeigentümer gesichert werden. Eine

Wiederherstellung kann auch von SYSDBA oder dem Datenbankeigentümer durchgeführt werden. Wenn jedoch der Schalter `-c(reate)` verwendet wird, kann jeder validierte Benutzername eine neue Datenbank aus einer Sicherungsdatei erstellen.

`-V[ERIFY]`

Normalerweise arbeitet `gbak` leise, ohne dass Informationen auf das Display geschrieben werden. Dieser Schalter kehrt diese Situation um und bewirkt, dass viele Informationen angezeigt werden. Standardmäßig wird die Ausgabe auf dem Bildschirm angezeigt. Sie können die Ausgabe jedoch mit dem Schalter `-y` in eine Protokolldatei umleiten.

`-Y <Dateiname> oder der Text "suppress"`

Wird in Verbindung mit dem Schalter `-v[erify]` verwendet, um Statusmeldungen auf eine Datei oder ein Gerät anstatt auf den Bildschirm umzuleiten oder sie insgesamt zu unterdrücken.

Wenn "-y suppress" verwendet wird, werden keine Informationen auf den Bildschirm geschrieben, unabhängig davon, ob `-v[erify]` angegeben ist.

Wenn ein Dateiname angegeben wird und der Schalter `-v[erify]` angegeben wird, wird die Datei geschrieben, um Fortschritte, Fehler usw. aufzuzeichnen.

In allen bekannten Dokumentationen zu diesem Schalter wird erwähnt, dass der Text "`suppress_output`" und nicht "`suppress`" sein sollte. Dies ist ein Fehler in der Dokumentation, da der Quellcode für `gbak` zeigt, dass der Schalter "`suppress`" sein muss.

Die Protokolldatei darf nicht vorhanden sein. In diesem Fall schlägt der Sicherungs- oder Wiederherstellungsvorgang fehl:

```
tux> rm employee.log
tux> gbak -backup employee.fdb employee.fbk -y employee.log -v

tux> ls -l employee.log
-rw-r--r-- 1 firebird firebird 21610 2010-08-04 10:22 employee.log

tux> gbak -backup employee.fdb employee.fbk -y employee.log -v
gbak:cannot open status and error output file employee.log
```

`-Z`

Diese Option zeigt einige Informationen über die verwendete Version von `gbak` und optional eine Datenbank an. Führen Sie den Befehl wie folgt aus, um nur die Version von `gbak` zu erhalten:

```
tux> gbak -z

gbak:gbak version LI-V2.5.0.26074 Firebird 2.5
gbak: ERROR:requires both input and output filenames
gbak:Exiting before completion due to errors

tux> echo $?
1
```

Dies zeigt die aktuelle Version von `gbak` an und wird nach dem Anzeigen einiger Fehlermeldungen mit einem Fehlercode von 1 beendet, wie oben durch den Befehl `echo` gezeigt. Diese Methode versucht nicht, Datenbanken zu sichern, und erfordert nicht, dass ein Benutzername und ein Kennwort definiert oder angegeben werden.

Wenn Sie die Versionsnummer des Dienstprogramms `gbak` sowie Details zur Datenbank anzeigen möchten, müssen Sie einen gültigen Datenbanknamen *und* Sicherungsdateiname wie folgt angeben:

```
tux> gbak -z employee employee.fbk -user sysdba -password secret

gbak:gbak version LI-V2.1.3.18185 Firebird 2.1
gbak:    Version(s) for database employee
Firebird/linux (access method),version LI-V2.1.3.18185 Firebird 2.1
Firebird/linux (remote server),version LI-V2.1.3.18185
Firebird 2.1/tcp (tux)/P11
Firebird/linux (remote interface), version LI-V2.1.3.18185
Firebird 2.1/tcp (tux)/P11
on disk structure version 11.1

tux> echo $?
0
```

Sie werden aus dem oben Gesagten feststellen, dass ein gültiger Benutzername und ein gültiges Kennwort in der Befehlszeile oder mithilfe der Umgebungsvariablen `ISC_USER` und `ISC_PASSWORD` definiert werden müssen. Diese Version des Befehls wird mit einem Fehlercode von Null beendet.

Diese Methode zum Aufrufen von `gbak` *wird* eine Sicherung der Datenbank erzeugen. Wenn Ihre Datenbank groß ist, kann dies einige Zeit dauern, und die angegebene Sicherungsdatei wird überschrieben, wenn sie bereits vorhanden ist. Also Vorsicht.

HINWEIS: Die obige Ausgabe wurde leicht abgekürzt, damit sie der Seitenbreite für ein PDF entspricht.

`-help`

Die Hilfe ist eigentlich keine gültige Option, kann jedoch verwendet werden, um den folgenden

Informationsbildschirm als Ausgabe von "gbak" in Firebird 2.0 anzuzeigen:

```
gbak: legale Schalter sind:
        -B[ACKUP_DATABASE]  Sicherungsdatenbank in Datei
        -BU[FFERS]          überschreibt standardmäßig Seitenpuffer
        -C[REATE_DATABASE]  Datenbank aus Sicherungsdatei erstellen
        -CO[NVERT]          sichert externe Dateien als Tabellen
        -E[XPAND]           keine Datenkomprimierung
        -FA[CTOR]           Blockierungsfaktor
        -G[ARBAGE_COLLECT]  sperrt die Speicherbereinigung
        -I[NACTIVE]         deaktiviert Indizes während der Wiederherstellung
        -IG[NORE]           ignoriert schlechte Prüfsummen
        -K[ILL]             wiederherstellen, ohne Schatten zu erzeugen
        -L[IMBO]            ignoriert Transaktionen in der Schwebe
        -M[ETA_DATA]        Backup-Metadaten
        -MO[DE] <access>    "read_only" oder "read_write" Zugriff
        -N[O_VALIDITY]      stellt die Gültigkeitsbedingungen der Datenbank nicht
wieder her
        -NOD[BTRIGGERS]     führt keine Datenbank-Trigger aus
        -NT                 Nicht transportierbares Sicherungsdateiformat
        -O[NE_AT_A_TIME]    stellt jeweils eine Tabelle wieder her
        -OL[D_DESCRIPTIONS] speichert Metadatenbeschreibungen alten Stils
        -P[AGE_SIZE]        überschreibt die Standardseitengröße
        -PAS[SWORD]         Firebird-Passwort
        -R[ECREATE_DATABASE] [O[VERWRITE]] erstellen (ersetzen, wenn O[VERWRITE]
verwendet wird)
                            Datenbank aus Sicherungsdatei
        -REP[LACE_DATABASE] ersetzt Datenbank aus Sicherungsdatei
        -RO[LE]             Firebird SQL-Rolle
        -SE[RVICE]          verwendet den Service Manager
        -T[RANSPORTABLE]    transportable Backup - Daten im XDR-Format
        -USE_[ALL_SPACE]    reserviert keinen Speicherplatz für Datensatzversionen
        -USER               Firebird Benutzername
        -V[ERIFY]           meldet jede ergriffene Aktion
        -Y <Pfad>           Ausgabe der Statusmeldung umleiten / unterdrücken
        -Z                  Versionsnummer drucken
```

Die Erklärung des Schalters `-m[eta_data]` impliziert, dass er nur in einer Sicherungssituation nützlich ist. Dies ist nicht der Fall, da es auch für eine Wiederherstellung verwendet werden kann.

Ab Firebird 2.5 gibt es einen neuen Schalter `-?`, um die Liste der gültigen Optionen anzuzeigen. Die Ausgabe hat ein etwas anderes Layout und einige neue Optionen wurden hinzugefügt:

```
gbak:Gültige Schalter sind:
        -B(ACKUP_DATABASE)    Datenbank in Datei sichern
        -C(REATE_DATABASE)    Datenbank aus Sicherungsdatei erstellen
(wiederherstellen)
        -R(ECREATE_DATABASE) [O(VERWRITE)] erstellen (oder ersetzen, wenn OVERWRITE
verwendet wird)
                              Datenbank aus Sicherungsdatei (wiederherstellen)
        -REP(LACE_DATABASE)   Datenbank aus Sicherungsdatei ersetzen
(wiederherstellen)

gbak:Sicherungsoptionen sind:
        -CO(NVERT)            Sichern Sie externe Dateien als Tabellen
        -E(XPAND)             Keine Datenkompression
        -FA(CTOR)             Sperrfaktor
        -G(ARBAGE_COLLECT)    Garbage Collection sperren
        -IG(NORE)             Ignoriere fehlerhafte Checksummen
        -L(IMBO)              Ignoriere Transaktionen in Limbo-Umgebungen
        -NOD(BTRIGGERS)       Keine Datenbanktrigger ausführen
        -NT                   Nicht-transportable Sicherungsdateien
        -OL(D_DESCRIPTIONS)   Speichert Metadatenbeschreibungen im alten Stil
        -T(RANSPORTABLE)      Transportables Backup -- Daten im XDR-Format

gbak:Wiederherstellungsoptionen sind:
        -BU(FFERS)            Standardeinstellung für Seitenpuffer
        -FIX_FSS_D(ATA)       Behebt fehlerhafte UNICODE_FSS-Daten
        -FIX_FSS_M(ETADATA)   Behebt fehlerhafte UNICODE_FSS-Metadaten
        -I(NACTIVE)           Indizes während der Wiederherstellung deaktivieren
        -K(ILL)               Wiederherstellen ohne Schatten zu erzeugen
        -MO(DE) <access>      Zugriff "read_only" oder "read_write"
        -N(O_VALIDITY)        Stellt die Gültigkeitsbedingungen der Datenbank nicht
wieder her
        -O(NE_AT_A_TIME)      Stellt jeweils eine Tabelle wieder her
        -P(AGE_SIZE)          Überschreibt die Standard-Seitengröße
        -USE_(ALL_SPACE)      Reserviert keinen Platz für Datensatzversionen

gbak:Allgemeine Optionen sind:
        -FE(TCH_PASSWORD)     Holt Kennwort aus Datei
        -M(ETA_DATA)          Nur Metadaten sichern oder wiederherstellen
        -PAS(SWORD)           Firebird-Kennwort
        -RO(LE)               Firebird SQL-Rolle
        -SE(RVICE)            Service-Manager verwenden
        -USER                 Firebird-Benutzername
        -V(ERIFY)             Jede Aktion ausgeben
        -Y  <path>            Statusausgaben umleiten / unterdrücken
        -Z                    Versionsnummer ausgeben
```

Die oben gezeigten Klammern geben an, wie viel des Switch-Namens Sie verwenden müssen, um Mehrdeutigkeiten zu vermeiden. Sobald Sie das absolute Minimum angegeben haben — den Teil vor dem öffnenden '[' — können Sie so viel von dem verwenden, was folgt, wie Sie möchten. Um

beispielsweise den Schalter `-b[ackup_database]` zu verwenden, müssen Sie in der Befehlszeile mindestens `-b` angeben, aber alles zwischen `-b` und `-backup_database` wird akzeptiert.

Wenn Sie den Schalter `-help` wie diesen oder einen anderen ungültigen Schalter verwenden, wird `gbak` unter Linux und Windows mit dem Fehlercode 1 beendet.

2.2. Backup-Schalter

Wenn beim Ausführen von `gbak` der erste Dateiname ein Datenbankname oder ein Datenbankalias ist, erstellt `gbak` standardmäßig eine Sicherung der Datenbank im transportablen Format. Die Sicherungsdatei wird gemäß dem zweiten in der Befehlszeile angegebenen Dateinamen benannt.

Wenn Sie möchten, können Sie die Ausgabe an die Standardausgabe und nicht an eine Sicherungsdatei senden. In diesem Fall müssen Sie stdout als Dump-Dateinamen angeben. Dies ist nicht wirklich nützlich, es sei denn, Sie möchten den Speicherauszug durch ein Tool leiten, um ihn auf irgendeine Weise zu ändern. Sie können die Ausgabe direkt an eine `gbak`-Wiederherstellungsoperation weiterleiten, um eine Datenbank zu klonen, ohne eine Zwischen-Dump-Datei zu benötigen. Ein Beispiel wird später in diesem Handbuch gegeben.

Bei der Sicherung einer Datenbank sind zusätzlich zu den oben genannten Schaltern die folgenden Schalter von Nutzen:

`-B[ACKUP_DATABASE]`

Dieser Schalter wird immer dann verwendet, wenn Sie eine Sicherungskopie einer Datenbank erstellen möchten.

`-CO[NVERT]`

Dieser Schalter bewirkt, dass alle als `extern` definierten Tabellen so gesichert werden, als wären sie normale (nicht externe) Tabellen. Wenn diese Speicherauszugsdatei zum Wiederherstellen einer Datenbank verwendet wird, sind die Tabellen, die in der ursprünglichen Datenbank extern waren, nicht mehr extern.

`-E[XPAND]`

Normalerweise komprimiert `gbak` die Ausgabedatei. Dieser Schalter verhindert, dass diese Komprimierung stattfindet.

`-FA[CTOR] <Blockgröße>`

Wenn Sie auf einem physischen Bandgerät sichern, können Sie mit diesem Schalter den Sperrfaktor des Bandes angeben.

`-G[ARBAGE_COLLECT]`

Die Verwendung dieses Schalters verhindert, dass die Garbage Collection von Firebird stattfindet, während `gbak` ausgeführt wird. Normalerweise stellt `gbak` wie jede andere Verbindung eine Verbindung zur Datenbank her und die Speicherbereinigung wird normal ausgeführt. Die Verwendung dieses Schalters verhindert, dass die Speicherbereinigung im

Verlauf der Sicherung ausgeführt wird. Dies kann dazu beitragen, die Sicherung zu beschleunigen.

`-IG[NORE]`

Dieser Schalter bewirkt, dass `gbak` fehlerhafte Prüfsummen in der Datenbank ignoriert. Dies kann verwendet werden, um zu versuchen, eine Datenbank zu sichern, die aufgrund von Prüfsummenfehlern fehlgeschlagen ist. Es gibt jedoch keine Garantie dafür, dass die Daten verwendet werden können. Treffen Sie daher am besten andere Vorsichtsmaßnahmen, um Ihre Daten zu erhalten.

`-L[IMBO]`

Wenn Sie eine zweiphasige Transaktion (über zwei verschiedene Datenbanken) haben, die fehlgeschlagen ist, weil ein Server vor dem Festschreiben gestorben ist, aber nachdem die Änderungen vorgenommen wurden, haben Sie eine Limbo-Transaktion. Dieser Schalter zwingt die Sicherung, Daten von solchen unterbrochenen Transaktionen zu ignorieren. Es sollte nicht für normale Sicherungen verwendet werden und nur wie der Schalter `-ig[nore]` verwendet werden, um zu versuchen, einen Fehler zu beheben.

`-M[ETA_DATA]`

Siehe oben.

`-NT`

Dieser Schalter deaktiviert den Schalter `-t[ransportable]` (der standardmäßig aktiviert ist) und bewirkt, dass die Speicherauszugsdatei in plattformabhängigen Formaten erstellt wird. Wenn Sie diesen Schalter zum Erstellen eines Backups verwenden, können Sie das Backup nur auf einer ähnlichen Plattform wiederherstellen. Sie können beispielsweise keine Dump-Datei von Linux auf einen Windows-Server übertragen.

`-OL[D_DESCRIPTIONS]`

Es ist unwahrscheinlich, dass dieser Schalter verwendet wird. Es gilt als veraltet. Der Zweck besteht darin, die Sicherung unter Verwendung des Formats der Metadatenbeschreibungen älterer InterBase-Versionen zu erzwingen.

`-PAS[SWORD] <Kennwort>`

Siehe oben.

`-RO[LE] <Rollenname>`

Siehe oben.

`-SE[RVICE] <Servicename>`

Dieser Schalter bewirkt, dass `gbak` eine entfernte Datenbank über den Service Manager sichert. Dadurch wird die Sicherungsdatei auf dem Remoteserver erstellt, sodass das Pfadformat und der Dateiname auf dem Remoteserver gültig sein müssen. Der Servicename ist derzeit immer der Text `service_mgr`.

Mit dieser Option können Sie auch eine Verbindung zu einer lokal gehosteten Datenbank herstellen. In diesem Fall kann das Erstellen einer Sicherung mit dieser Option schneller ausgeführt werden als der direkte Zugriff auf die Datenbank. Weitere Informationen zum Beschleunigen von Sicherungen finden Sie im folgenden Abschnitt.

`-T[RANSPORTABLE]`

Das Standard-Dump-Dateiformat kann transportiert werden. Transportierbare Sicherungsdateien werden in einem Format geschrieben, das als XDR-Format (External Data Representation) bezeichnet wird. In diesem Format kann ein Speicherauszug, der auf einem Server eines Typs erstellt wurde, erfolgreich auf einem Server eines anderen Typs wiederhergestellt werden.

`-USER <username>`

Siehe oben.

`-V[ERIFY]`

Siehe oben.

`-Y <vollständiger Pfad der Log-Datei> oder der Text "suppress"`

Siehe oben.

2.3. Wiederherstellungs-Schalter

Wenn beim Ausführen eines `gbak`-Befehls der *erst*-Dateiname ein Datenbank-Sicherungsdateiname ist, führt `gbak` standardmäßig eine Wiederherstellung der Datenbank durch, vorausgesetzt, Sie geben eine der Optionen `-c[reate database]`, `-rep[lace_database]` oder `-r[ecreate_database]`, um Ihre Absichten klar zu machen. Die Datenbank wird unter dem Dateinamen wiederhergestellt, der als zweiter Dateiname in der Befehlszeile angegeben ist.

Wenn Sie möchten, können Sie die Speicherauszugsdaten direkt von der Standardeingabe und nicht von einer Sicherungsdatei lesen. In diesem Fall müssen Sie stdin als Namen der Speicherauszugsdatei angeben. Sie können eine `gbak`-Dump-Operation direkt an eine `gbak`-Wiederherstellungsoperation weiterleiten, um eine Datenbank zu klonen, ohne eine Zwischen-Dump-Datei zu benötigen. Ein Beispiel wird später in diesem Handbuch gegeben.

Bei der Wiederherstellung oder dem Ersetzen einer Datenbank sind zusätzlich zu den oben genannten die folgenden Schalter von Nutzen:

`-BU[FFERS] <Anzahl der Puffer>`

Dieser Schalter legt die Standardgröße des Datenbankcaches (in Anzahl der Datenbankseiten) für die wiederherzustellende Datenbank fest. Wenn eine Datenbank überschrieben wird, überschreibt diese Einstellung die vorherige Einstellung für die Cache-Größe.

`-C[REATE_DATABASE]`

Dieser Schalter bewirkt, dass eine neue Datenbank aus der Sicherungsdatei erstellt wird. Die Datenbankdatei darf nicht vorhanden sein, sonst schlägt die Wiederherstellung fehl. Entweder dieser Schalter oder `-rep[lace_database]` oder `-r[ecreate_database]` muss angegeben werden.

`-FIX_FSS_D[ATA]`

Dieser Schalter zwingt `gbak`, fehlerhafte UNICODE_FSS-Zeichendaten während einer Wiederherstellung zu korrigieren.

Dieser und der folgende Schalter sollten unter normalen Umständen nicht erforderlich sein. Wenn jedoch ein Wiederherstellungsvorgang mit einem Fehler "fehlerhafte Zeichenfolge" fehlschlägt, verweist die von `gbak` ausgegebene Nachricht den Benutzer auf einen oder beide dieser Schalter, um die fehlerhaften UNICODE_FSS-Daten oder Metadaten entsprechend zu korrigieren.

Seit Firebird 2.5.

`-FIX_FSS_M[ETADATA]`

Dieser Schalter zwingt `gbak`, fehlerhafte UNICODE_FSS-Metadaten während einer Wiederherstellung zu korrigieren.

Dieser und der vorhergehende Schalter sollten unter normalen Umständen nicht erforderlich sein. Wenn jedoch ein Wiederherstellungsvorgang mit einem Fehler "fehlerhafte Zeichenfolge" fehlschlägt, verweist die von `gbak` ausgegebene Nachricht den Benutzer auf einen oder beide dieser Schalter, um die fehlerhaften UNICODE_FSS-Daten oder Metadaten entsprechend zu korrigieren.

Seit Firebird 2.5.

`-I[NACTIVE]`

Mit diesem Schalter kann eine Datenbank wiederhergestellt werden, wenn ein vorheriger Wiederherstellungsversuch aufgrund von Indexfehlern fehlgeschlagen ist. Alle Indizes in der wiederhergestellten Datenbank sind inaktiv.

`-K[ILL]`

Dieser Schalter stellt die Datenbank wieder her, erstellt jedoch keine zuvor vorhandenen Schattendateien neu.

`-M[ETA_DATA]`

Siehe oben.

`-MO[DE] <Zugriffsmodus>`

Mit diesem Schalter kann die wiederhergestellte Datenbank beim Öffnen auf den angegebenen Zugriffsmodus eingestellt werden. Standardmäßig wird der Modus aus der Datenbank übernommen, die gesichert wurde.

`-N[O_VALIDITY]`

Dieser Schalter ähnelt dem obigen Schalter `-i[nactive]`, außer dass alle *check*-Einschränkungen

in der wiederhergestellten Datenbank deaktiviert wurden.

`-NOD[BTRIGGERS]`

Neuer Schalter von Firebird 2.1, der verhindert, dass die *Datenbank-Trigger* bei einer Wiederherstellung ausgelöst werden. Datenbank-Trigger sind eine neue Funktion ab Firebird 2.0 und unterscheiden sich von *Tabellen-Triggern*, die weiterhin ausgelöst werden.

`-O[NE_AT_A_TIME]`

Dieser Schalter bewirkt, dass bei der Wiederherstellung jeweils eine Tabelle wiederhergestellt wird. Dies kann nützlich sein, wenn eine vorherige Wiederherstellung aufgrund von Datenfehlern fehlgeschlagen ist. Normalerweise erfolgt eine Wiederherstellung in einer einzelnen Transaktion mit einem einzelnen Commit am Ende der Wiederherstellung. Wenn die Wiederherstellung aus irgendeinem Grund unterbrochen wird, ist eine leere Datenbank das Endergebnis. Bei Verwendung der Option `-o[ne_at_a_time]` wird eine Transaktion für jede Tabelle verwendet und nach Wiederherstellung jeder Tabelle festgeschrieben.

`-P[AGE_SIZE] <neue Größe der Seiten>`

Verwenden Sie diesen Schalter, um die Standardgröße der Datenbankseite zu ändern. Standardmäßig wird die Datenbank mit einer Seitengröße wiederhergestellt, die derjenigen entspricht, die beim Speichern der Datenbank verwendet wurde.

`-PAS[SWORD] <Kennwort>`

Siehe oben.

`-R[ECREATE_DATABASE] [O[VERWRITE]]`

Neu von Firebird 2.x. Erstellen Sie die benannte Datenbank aus der Sicherungsdatei (oder ersetzen Sie sie, wenn `o[verwrite]` verwendet wird). Der Datenbankdateiname sollte noch nicht vorhanden sein, da sonst ein Fehler auftritt. Dies ist nicht der Fall, wenn auch die Option `o[verwrite]` verwendet wird.

Dies ist ein neuer Schalter, der absichtlich mit `-r` abgekürzt wird, um zu verhindern, dass ahnungslose Datenbankadministratoren eine vorhandene Datenbank überschreiben, wenn sie glauben, dass `-r` mit `-restore` abgekürzt wurde. Jetzt sind besondere Anstrengungen erforderlich, um dies zu verwalten, zumal `-restore` eigentlich nie ein gültiger Schalter war. `-r` war in der Tat eine Abkürzung für `-replace_database` und tat dies, indem *zuerst* die vorhandene Datenbankdatei gelöscht und *dann* aus der Sicherung neu erstellt wurde.

Die Nutzung von `-r[ecreate_database] o[verwrite]` ist identisch mit `-rep[lace_database]`.

`-REP[LACE_DATABASE]`

Ersetzen Sie die Datenbank aus der Sicherungsdatei. Dieser Schalter wurde in früheren Versionen (zu Firebird 2.x) mit `-r` abgekürzt. Dieser Schalter wird in einer Version von Firebird später als 2.1.3 (wo er noch vorhanden ist) entfernt. Es wird empfohlen, stattdessen den Schalter `-r[ecreate_database] o[verwrite]` zu verwenden.

`-SE[RVICE] <Servicename>`

Verwenden Sie den Servicemanager für eine entfernte Datenbank, um eine entfernte Datenbank wiederherzustellen. Der Servicename ist derzeit immer der Text `service_mgr`.

HINWEIS: Mit dieser Option können Sie auch eine Verbindung zu einer lokal gehosteten Datenbank herstellen. In diesem Fall kann das Wiederherstellen einer Sicherung mit dieser Option schneller ausgeführt werden als der direkte Zugriff auf die Datenbank. Weitere Informationen zum Beschleunigen von Wiederherstellungen finden Sie im folgenden Abschnitt.

`-USE_[ALL_SPACE]`

Dieser Schalter erzwingt, dass die Wiederherstellung 100% jeder Datenbankseite verwendet und somit keinen Raum für Änderungen lässt. Wenn Sie diesen Schalter weglassen, bleibt etwas Speicherplatz für spätere Änderungen frei. Die Verwendung dieses Schalters ist wahrscheinlich nur dann von praktischem Nutzen, wenn die Datenbank im schreibgeschützten Modus erstellt und verwendet wird und keine Aktualisierungen vorhandener Daten erforderlich sind.

Sobald eine Datenbank mit dieser Option wiederhergestellt wurde, werden alle Datenbankseiten zu 100% gefüllt und es bleibt kein freier Speicherplatz für Aktualisierungen übrig. Wenn Sie diesen Schalter verwenden, setzen Sie ein Flag auf der Datenbank-Headerseite, um zu signalisieren, dass alle Seiten zu 100% gefüllt werden sollen. Dies gilt für alle neuen Seiten, die nach der Wiederherstellung erstellt wurden.

Sie können diese Einstellung mit `gfix -use full | reserve database_name` überschreiben, wobei `full` 100% jeder Seite verwendet und `reserve` Platz für nachfolgende Aktualisierungen reserviert. Weitere Informationen finden Sie im `gfix`-Handbuch.

`-USER <Benutzername>`

Siehe oben.

`-V[ERIFY]`

Siehe oben.

`-Y <Pfad der Log-Datei>`

Siehe oben.

Kapitel 3. Backup Mode

Bevor Sie andere Tools zum Erstellen einer Sicherung Ihrer Firebird-Datenbank in Betracht ziehen, stellen Sie sicher, dass Sie wissen, was die Tools tun und wie sich eine laufende Datenbank auf sie auswirkt. Wenn Sie beispielsweise *Winzip* verwenden, um eine komprimierte Kopie einer Datenbank zu erstellen, und dies tun, wenn Benutzer auf das System zugreifen, sind die Chancen für eine erfolgreiche Wiederherstellung dieser Datenbank gering. Sie müssen entweder immer die Tools `gbak` oder `nbackup` verwenden, die wissen, wie die Datenbank funktioniert, oder `gfix` verwenden, um die Datenbank vollständig herunterzufahren, bevor Sie überhaupt versuchen, die Datenbankdatei(en) zu sichern.

`gbak` erstellt eine konsistente Sicherung der Datenbank, indem eine Transaktion gestartet wird, die sich über den Sicherungszeitraum erstreckt. Wenn die Sicherung abgeschlossen ist, wird die Transaktion beendet. Dies bedeutet, dass der Sicherungsprozess ausgeführt werden kann, während Benutzer in der Datenbank arbeiten. Bei Transaktionen, die nach Beginn des Sicherungsvorgangs gestartet werden, werden jedoch keine der geänderten Daten in die Sicherungsdatei geschrieben. Die Sicherung stellt eine Kopie der gesamten Datenbank zum Zeitpunkt des Beginns der Sicherung dar.

Die Dump-Datei, die durch ein Standard-Backup mittels `gbak` erstellt wurde, ist plattformübergreifend (transportabel), sodass ein auf einem Windows-Server erstelltes Backup verwendet werden kann, um dieselbe Datenbank auf einem Linux-Server oder einer anderen von Firebird unterstützten Plattform neu zu erstellen. Dies gilt nicht für Kopien Ihrer Datenbank (während die Datenbank geschlossen wurde!), Die mit Tools wie *Winzip* usw. erstellt wurden. Diese Kopien sollten immer nur zum Wiederherstellen einer Datenbank auf derselben Plattform wie die kopierte verwendet werden.

Sichern Sie die Datenbank immer mit der Version von `gbak`, die mit dem laufenden Datenbankserver geliefert wird.

Und ein letzter Gedanke zu Sicherungen, unabhängig davon, ob die Sicherung fehlerfrei abgeschlossen wurde, mit einem Fehlercode von 0 beendet wurde und alles in Ordnung zu sein scheint. Woher wissen Sie eigentlich, dass die erstellte Sicherungsdatei verwendbar ist? Die kurze Antwort lautet: Sie tun es nicht. Wann immer Sie eine wertvolle Datenbank haben — und das sollten sie alle sein --, wird dringend empfohlen, Ihre Sicherungsdateien zu verwenden, um eine Testwiederherstellung einer Datenbank entweder auf demselben Server oder noch besser auf einem anderen Server zu erstellen. Nur so können Sie sicher sein, dass die Sicherung erfolgreich ist.

Das folgende Beispiel zeigt eine Sicherung, die auf einem Server mit dem Namen *linux* erstellt und zum Erstellen eines Klons der Datenbank auf einem anderen Linux-Server mit dem Namen *tux* verwendet wird, um sicherzustellen, dass alles in Ordnung ist. Zunächst das Backup unter *linux*:

```
linux> gbak -backup -verify -y backup.log employee employee.fbk
linux> gzip -9 employee.fbk
```

Beachten Sie, dass der obige gbak-Befehl wie folgt geschrieben werden kann, wobei der Schalter -b[ackup] weggelassen wird, da gbak standardmäßig eine Sicherung ausführt, wenn keine anderen geeigneten Schalter angegeben sind:

```
linux> gbak -verify -y backup.log employee employee.fbk
```

Dann, auf dem Server *tux*:

```
tux> scp norman@linux:employee.fbk.gz ./

Using keyboard-interactive authentication.
Password:
employee.fbk.gz           |         19 kB |  19.3 kB/s | ETA: 00:00:00 | 100%

tux> gunzip employee.fbk.gz
tux> gbak -replace -verify -y restore.log employee.fbk employee.restore.test
```

Zu diesem Zeitpunkt hat die Wiederherstellung funktioniert und die vorherige Datenbank mit dem Namen employee.restore.test überschrieben.

Der tatsächliche Speicherort der Datenbank für die Datenbank employee.restore.test wird in der Datei aliases.conf in /opt/firebird auf dem Server definiert. In diesem Test wird es in /opt/firebird/database/employee.restore.fdb aufgelöst.

Zum weiteren Nachweis der Zuverlässigkeit kann die Anwendung anhand dieses Klons der Live-Datenbank getestet werden, um sicherzustellen, dass alles in Ordnung ist.

3.1. Backup beschleunigen

Es gibt einige Tricks, mit denen Sie die Sicherung beschleunigen können. Die erste besteht darin, zu verhindern, dass die Speicherbereinigung ausgeführt wird, während die Sicherung ausgeführt wird. Durch die Speicherbereinigung werden alte Datensatzversionen gelöscht, die nicht mehr benötigt werden. Dies wird normalerweise durch einen manuellen oder automatischen Sweep oder durch einen vollständigen Tabellenscan aller betroffenen Tabellen abgedeckt. Wenn gbak auf alle Zeilen in den zu sichernden Tabellen zugreift, wird auch die Speicherbereinigung ausgelöst und kann bei einer großen Anzahl von Aktualisierungen die Sicherung verlangsamen. Verwenden Sie die Option -g[arbage_collect], um die Speicherbereinigung während der Sicherung zu verhindern.

```
tux> gbak -backup -garbage_collect employee /backups/employee.fbk
```

Die zweite Option besteht darin, die Datenbank mit der Option -se[rvice] zu sichern. Obwohl dies zur Durchführung von Remote-Sicherungen verwendet wird, kann es auch lokal verwendet werden. Mit dieser Option können Sie Ihre Backups beschleunigen. Es wird einfach vermieden, dass die Daten über das TCP-Netzwerk kopiert werden, was die Aktionen der Sicherung verlangsamen kann.

```
tux> gbak -backup -service tux:service_mgr employee /backups/employee.fbk
```

Im obigen Beispiel wird die Mitarbeiterdatenbank auf dem Tux-Server mithilfe des Service-Managers "remote" gesichert. Auf dem Tux-Server wird der Befehl natürlich ausgeführt, sodass er überhaupt nicht remote ausgeführt wird.

Sie können natürlich die Optionen `-g[arbage_collect]` und `-se[rvice]` kombinieren.

Kapitel 4. Wiederherstellungsmodus

Backups, die mit der gbak-Anwendung von einer Version von Firebird - oder InterBase - erstellt wurden, können von späteren Versionen von Firebird verwendet werden, um die Datenbank wiederherzustellen. Dies kann jedoch zu einem Upgrade der On Disc Structure (ODS) führen. In der betreffenden Datenbank wird der SQL-Dialekt niemals geändert. Wenn Sie eine InterBase-Dialekt-1-Datenbank sichern und dann die Dump-Datei verwenden, um sie beispielsweise unter Firebird 2.1 neu zu erstellen, wird der ODS auf 11.1 aktualisiert, der SQL-Dialekt bleibt jedoch 1.

Stellen Sie die Datenbank immer mit der Version von gbak wieder her, die mit dem Datenbankserver geliefert wurde, unter dem Sie die (neue) Datenbank ausführen möchten. Gbak aus Firebird 2.1 kann jedoch verwendet werden, um eine Datenbank auf einer beliebigen Version von Firebird wiederherzustellen.

Wenn Sie möchten, können Sie den SQL-Dialekt mit gfix ändern.

Unter normalen Umständen erfolgt die Wiederherstellung einer Datenbank als einzelne Transaktion. Wenn die Wiederherstellung erfolgreich ist, werden die Daten durch ein Commit am Ende dauerhaft, andernfalls ist die Datenbank am Ende leer.

Die Wiederherstellungsoption -o[ne_at_a_time] verwendet eine Transaktion für jede Tabelle. Wenn die Tabelle fehlerfrei wiederhergestellt wird, wird ein Commit ausgeführt, das diese Tabelle dauerhaft macht, unabhängig davon, was mit nachfolgenden Tabellen geschieht.

4.1. Wiederherstellen oder neu erzeugen?

Sollte eine Datenbank wiederhergestellt oder ersetzt werden? Das Wiederherstellen einer Datenbank ist der Vorgang, bei dem Sie die vorhandene Datei übernehmen und löschen, bevor Sie sie auf der Festplatte durch eine Sicherungskopie ersetzen. gbak tut dies, wenn Sie den Schalter -r[ecreate_database] o[verwrite] oder den Schalter -rep[lace_database] angeben. Was ist der Unterschied?

Wenn eine Datenbank auf der Festplatte vorhanden ist und Sie gbak auffordern, diese mit einem der beiden oben genannten Schalter wiederherzustellen, können Sie die Datenbank beschädigen, insbesondere wenn die Datenbank verwendet wird und nicht mit gfix heruntergefahren wurde. Wenn Sie die Wiederherstellung einer Datenbank nur teilweise abgeschlossen haben und einige Benutzer entscheiden, ob sie sich anmelden können, ist die Datenbank möglicherweise beschädigt.

Wenn der Wiederherstellungsprozess feststellt, dass die Speicherauszugsdatei beschädigt ist, schlägt die Wiederherstellung fehl und Ihre zuvor funktionierende Datenbank ist für immer verschwunden.

Es ist ersichtlich, dass das Wiederherstellen einer Datenbank eine schwierige Erfahrung sein kann.

Erstellen Sie aus Sicherheitsgründen die Datenbank immer unter einem neuen Namen — einem Klon — und aktualisieren Sie die Datei aliases.conf, und referenzieren Sie die Datenbank hier. Auf diese Weise verweisen Sie Ihre Benutzer unabhängig vom tatsächlichen Dateinamen auf dem Server immer über den Alias auf die Datenbank.

4.2. Fehlerhafte Zeichenfolgenfehler während der Wiederherstellung

Während eines Wiederherstellungsvorgangs, höchstwahrscheinlich beim Wiederherstellen einer Sicherung, die mit einer älteren `gbak`-Version erstellt wurde, können in der Ausgabe von `gbak` Fehlermeldungen angezeigt werden, die auf fehlerhafte Unicode-Zeichenfolgen hinweisen. Der Grund, warum diese auftreten können, ist wie von Helen Borrie erklärt:

> Der Quelltext gespeicherter Prozeduren (und verschiedener anderer Objekttypen, z. B. CHECK-Einschränkungen) wird ebenso wie der "kompilierte" BLR-Code in einem Blob gespeichert. Wenn Sie eine Datenbank wiederherstellen, wird das BLR nicht neu erstellt: Das gleiche BLR wird verwendet, bis Sie das Objekt das nächste Mal neu erstellen oder ändern.
>
> In der Vergangenheit hat die Engine in Bezug auf die Transliteration von in die Quelle und das BLR eingebetteten Zeichenfolgen nicht das Richtige getan. Wie Sie wahrscheinlich wissen, wurde in Version 2.1 und 2.5 viel Arbeit geleistet, um die internationalen Sprachprobleme anzugehen. Ein Nebeneffekt davon war, dass alles, was aus Daten und Metadaten gelesen wurde, einer Überprüfung der "Wohlgeformtheit" unterzogen wurde. Daher werfen diese zuvor gespeicherten Quell- und BLR-Objekte beim Wiederherstellen "fehlerhafte Zeichenfolgen" -Fehler aus, wenn gbak versucht, die Daten in diesen Systemtabellendatensätzen zu lesen und zu schreiben. Dieser sehr alte Fehler betrifft auch Benutzer-Blobs, wenn sie mit dem Zeichensatz NONE gespeichert wurden und der Client so konfiguriert ist, dass er einen bestimmten Zeichensatz liest, in den die gespeicherten Daten nicht transliteriert werden konnten.
>
> In Version 2.1 gab es Skripte in ../misc, die Sie ausführen konnten, um die Metadaten-Blobs zu reparieren und als Vorlage für die Reparatur ähnlicher Fehler in Blobs in Ihren Benutzerdaten zu verwenden. Die Reparaturschalter wurden dem gbak-Wiederherstellungscode in Version 2.5 hinzugefügt, um die gleichen Korrekturen an Metadaten bzw. Daten während des Wiederherstellungsvorgangs einer Datenbank für das Upgrade vorzunehmen.

4.3. Wiederherstellung beschleunigen

Die Wiederherstellung einer Datenbank aus einer Sicherung kann schneller ausgeführt werden, wenn die Option `-se[rvice]` verwendet wird. Obwohl dies für Remote-Wiederherstellungen verwendet wird, kann es auch lokal verwendet werden. Es wird einfach vermieden, dass die Daten über das TCP-Netzwerk kopiert werden, was die Aktionen der Wiederherstellung verlangsamen kann.

```
tux> gbak -replace -service tux:service_mgr /backups/employee.fbk employee
```

Im obigen Beispiel wird die Mitarbeiterdatenbank auf dem Tux-Server mithilfe des Service-Managers "remote" gesichert. Auf dem Tux-Server wird der Befehl natürlich ausgeführt, sodass er überhaupt nicht remote ausgeführt wird.

Sie können natürlich die Optionen `-g[arbage_collect]` und `-se[rvice]` kombinieren.

Kapitel 5. Sicherheit von Backups

Wie Sie oben gesehen haben, kann jeder mit einem gültigen Benutzernamen und einem gültigen Kennwort eine Datenbank-Dump-Datei mittels `gbak` wiederherstellen, vorausgesetzt, er überschreibt keine vorhandene Datenbank. Dies bedeutet, dass Ihre wertvollen Daten von schändlichen Charakteren auf ihren eigenen Servern gestohlen und verwendet werden können, um eine Kopie Ihrer Datenbank zu erstellen und beispielsweise Ihre Verkaufszahlen einsehen.

Um dies zu verhindern, wird empfohlen, Vorsichtsmaßnahmen zu treffen. Sie sollten verhindern, dass Backups versehentlich überschrieben werden, bevor sie abgelaufen sind. Einige Vorsichtsmaßnahmen, die Sie treffen können, sind:

- Stellen Sie die Dump-Datei nach Abschluss der Sicherung immer als schreibgeschützt ein. Dies verhindert, dass die Datei überschrieben wird.
- Alternativ können Sie das Datum (und die Uhrzeit?) in Ihre Sicherungsdateinamen aufnehmen.
- Bewahren Sie Backups an einem sicheren Ort auf dem Server auf. Durch das Speichern von Sicherungen an einem Ort mit eingeschränktem Zugriff wird die Wahrscheinlichkeit verringert, dass Ihre Sicherungsdateien in die Natur gelangen.
- Bewahren Sie Bandkopien Ihrer Backups sehr sicher auf. Ein verschlossener sicherer oder externer Standort mit guter Sicherheit ist ratsam. Der externe Speicherort wird auch nach einer vollständigen Katastrophe von Nutzen sein, da die Sicherungen an einem separaten Speicherort auf dem Server gespeichert werden, auf dem sie benötigt werden.
- Überführen Sie Ihr Backup auf eine Partition oder Festplatte, auf der die Verschlüsselung aktiviert ist.
- Stellen Sie sicher, dass nur autorisiertes Personal Zugriff auf Bereiche hat, in denen Backups aufbewahrt werden.
- Testen Sie Ihre Sicherungen immer, indem Sie eine Datenbank aus einer kürzlich durchgeführten Sicherung klonen.

In Firebird 2.1 ist eine zusätzliche Sicherheitsfunktion in `gbak` und allen anderen Befehlszeilenprogrammen integriert. Diese neue Funktion verbirgt das Kennwort automatisch, wenn es in der Befehlszeile mit dem Schalter `-password` angegeben wird. `Gbak` ersetzt das Passwort durch Leerzeichen — eines für jedes Zeichen im Passwort. Dies verhindert, dass andere Benutzer im System, die den Befehl `ps` ausführen und Ihre Befehlszeile und Parameter anzeigen können, ein angegebenes Kennwort anzeigen. Auf diese Weise können nicht autorisierte Benutzer das angegebene Kennwort nicht erhalten.

```
tux> gbak -b -user SYSDBA -passw secret employee /backups/employee.fbk
```

```
tux> ps efx| grep -i gba[k]
20724 ... gbak -backup -user SYSDBA -passw           employee employee.fbk
... (lots more data here)
```

Sie können oben sehen, dass das Kennwort unter Firebird 2.1 nicht angezeigt wird, da jedes

Zeichen durch ein einzelnes Leerzeichen ersetzt wird. Dies bedeutet, dass es für jemanden möglich ist, herauszufinden, wie *lang* das Passwort sein könnte, und das könnte ein Hinweis für einen dedizierten Cracker sein. Wenn Sie die Länge des erforderlichen Passworts kennen, wird dies etwas einfacher. Verwenden Sie für optimale Ergebnisse eine zufällige Anzahl von Leerzeichen zwischen `-passw` und dem tatsächlichen Passwort. Je schwieriger Sie es den bösen Menschen in Ihrem Netzwerk machen, desto besser.

Kapitel 6. Rezepte für die Sicherung und Wiederherstellung

Die folgenden Rezepte zeigen Beispiele für Sicherungs- und Wiederherstellungsaufgaben mit `gbak`. Dies sind wahrscheinlich die häufigsten Fälle, denen Sie als DBA begegnen werden. Alle Beispiele verwenden die mit Firebird gelieferte Mitarbeiter-Datenbank (`employee`), und der tatsächliche Speicherort ist in `aliases.conf` korrekt konfiguriert. Jedes der folgenden Rezepte wird unter der Annahme ausgeführt, dass den Umgebungsvariablen `ISC_USER` und `ISC_PASSWORD` geeignete Werte zugewiesen wurden.

6.1. Voraussetzungen zum Sichern und Wiederherstellen

Wenn Sie eine geöffnete und laufende Datenbank ersetzen, besteht eine gute Chance, dass Sie sie beschädigen. Um optimale Ergebnisse und eine minimale Wahrscheinlichkeit einer Beschädigung einer Datenbank zu erzielen, sollten Sie diese schließen, bevor Sie sie ersetzen. Verwenden Sie zum Schließen einer Datenbank `gfix` wie folgt:

```
tux> gfix -shut -tran 60 employee
```

Das obige Beispiel verhindert, dass eine neue Transaktion gestartet wird, wodurch verhindert wird, dass neue Abfragen ausgeführt werden oder neue Sitzungen mit der Datenbank verbunden werden. Es dauert bis zu 60 Sekunden, bis sich alle abgemeldet haben und alle aktuellen Transaktionen abgeschlossen sind, bevor die Datenbank heruntergefahren wird. Wenn Transaktionen mit langer Laufzeit bis zum Ende von 60 Sekunden noch nicht abgeschlossen sind, tritt beim Herunterfahren eine Zeitüberschreitung auf und die Datenbank bleibt geöffnet.

Nach Abschluss der Wiederherstellung der Datenbank wird die Datenbank automatisch wieder zur Verwendung geöffnet.

6.2. Ein simples Backup & Restore

In diesem Beispiel wird eine Sicherung erstellt und die ursprüngliche Datenbank mit der neuen Sicherung sofort überschrieben. Dies ist normalerweise keine gute Idee, da die erste Aktion einer Wiederherstellung darin besteht, die Datenbank zu löschen.

```
tux> # Datenbank sichern
tux> gbak -backup employee /backups/employee.fbk

tux> # Datenbank wiederherstellen
tux> gfix -shut -tran 60 employee
tux> gbak -replace /backups/employee.fbk employee
```

6.3. Nur Metadaten

Es ist möglich, `gbak` zu verwenden, um eine leere Datenbank neu zu erstellen, die nur die verschiedenen *Domains*, *Tabellen*, *Indizes* usw. der ursprünglichen Datenbank enthält, jedoch keine der Daten. Dies kann hilfreich sein, wenn Sie Ihre Anwendung in einer Testumgebung geprüft haben und das System beispielsweise in eine Produktionsumgebung migrieren möchten, aber ohne Ihre Testdaten neu beginnen möchten.

```
tux> # Nur die Metadaten sichern
tux> gfix -shut -tran 60 employee
tux> gbak -backup -meta_data employee employee.meta.fbk
```

Wenn die obige Dump-Datei auf dem Produktionsserver wiederhergestellt wird, sind nur die Metadaten vorhanden.

Es gibt eine andere Möglichkeit, eine Datenbank ohne Daten und nur mit den Metadaten zu erstellen. Stellen Sie einfach aus einem vorhandenen Speicherauszug wieder her, der die Daten enthält, und geben Sie den Schalter `-m[eta_data]` in die Wiederherstellungsbefehlszeile ein. Die Datenbank wird wiederhergestellt, aber keine der Originaldaten sind vorhanden.

```
tux> # Nur die Metadaten wiederherstellen
tux> gbak -create employee.fbk mytest.fdb -meta_data
```

Der Schalter `-m[eta_data]` kann entweder für eine Sicherung oder eine Wiederherstellung verwendet werden, um die Erstellung einer Klondatenbank (oder das Überschreiben einer vorhandenen) ohne tatsächliche Daten zu erleichtern.

6.4. Die Sicherung aufteilen

Die in einem eigenen Handbuch dokumentierte Filteranwendung `gsplit` funktioniert nicht mehr. Dieser Filter wurde mit alten Versionen von InterBase und Firebird geliefert, damit große Datenbanksicherungen auf mehrere Dateien aufgeteilt werden können, damit die Dateisystemgrenzen eingehalten werden können. Solche Beschränkungen können die Größe einer CD sein, die Beschränkung auf 2 GB für einzelne Dateigrößen auf einer DVD, wobei einige Unix-Dateisysteme eine Beschränkung auf 2 GB usw. haben.

Mit `gbak` können die Speicherauszugsdateien in verschiedene Größen (mit mindestens 2048 Byte) aufgeteilt werden, und es werden nur die benötigten Dateien erstellt.

```
tux> # Datenbanksicherung in mehreren Dateien auftrennen.
tux> gbak -backup employee /backups/emp.a.fbk 600m /backups/emp.b.fbk 600m
```

Die Größen nach jedem Dateinamen geben an, wie groß diese bestimmte Datei sein darf. Die Standardgröße ist Byte, aber Sie können ein Suffix von `k`, `m` oder `g` angeben, um Einheiten von Kilo, Mega oder Gigabyte zu verwenden.

Wenn der Speicherauszug abgeschlossen ist, bevor in einige Dateien geschrieben wird, werden diese Dateien nicht erstellt. Eine Dump-Datei wird immer nur erstellt, wenn dies erforderlich ist.

Die Größe der endgültigen Speicherauszugsdatei wird stillschweigend ignoriert, wenn die Datenbank zu groß geworden ist, um eine abgeschnittene Sicherung abzuschließen. Wenn im obigen Beispiel die Sicherung insgesamt 1500MB benötigt, wird die letzte Datei auf eine endgültige Größe von 900m anstatt der angegebenen 600m geschrieben.

Um eine solche Sicherung mit mehreren Dateien wiederherzustellen, müssen Sie alle Dateinamen im Speicherauszug und in der *richtigen Reihenfolge* angeben. Das folgende Beispiel zeigt, wie die oben genannte Mitarbeiterdatenbank aus den beiden oben gespeicherten Dateien wiederhergestellt wird:

```
tux> # Datenbank aus mehreren Dateien wiederherstellen
tux> gfix -shut -tran 60 employee
tux> gbak -replace /backups/employee.a.fbk /backups/employee.b.fbk employee
```

6.5. ODS anpassen

Normalerweise ist das verwendete ODS dasjenige, welches von der wiederherstellenden Firebird-Version verwendet wird. Die obigen Beispiele ändern also tatsächlich den ODS, wenn die Datenbank wiederhergestellt wird. Die Sicherung sollte mit dem Dienstprogramm `gbak` durchgeführt werden, das von der alten ODS-Version von InterBase oder Firebird bereitgestellt wird. Die Wiederherstellung sollte mit `gbak` aus der neueren Version von Firebird durchgeführt werden.

```
tux> setenv_firebird 2.0
Firebird environment set for version 2.0.

tux> # Aktuelle ODS-Version prüfen (als root-Benutzer!)
tux> gstat -h employee|grep ODS
        ODS version             11.0

tux> # Alte Datenbank sichern
tux> gbak -backup employee /backups/employee.2_0.fbk

tux> setenv_firebird 2.1
Firebird environment set for version 2.1.

tux> # Datenbank neu erstellen und ODS aktualisieren
tux> gfix -shut -tran 60 employee
tux> gbak -replace /backups/employee.2_0.fbk employee

tux> # Neue ODS-Version prüfen (als root-Benutzer!)
tux> gstat -h employee|grep ODS
        ODS version             11.1
```

Danach wurde die alte 2.0 Firebird-Datenbank als Firebird 2.1-Datenbank mit dem entsprechenden Upgrade auf das ODS von 11.0 auf 11.1 neu erstellt — und die alte Datenbank gelöscht.

Das Skript `setenv_firebird` wird nicht mit Firebird geliefert und setzt einfach `PATH` usw., um die richtige Version von Firebird gemäß dem angegebenen Parameter zu verwenden.

6.6. Cache-Größe anpassen

Der Standard-Datenbank-Cache wird beim Erstellen der Datenbank oder anschließend mit `gfix` erstellt. `gbak` kann eine Datenbank wiederherstellen und auch die Standard-Cache-Größe zurücksetzen. Der Prozess ist wie folgt:

```
tux> # Aktuelle Cache-Größe prüfen (als root-Benutzer!)
tux> gstat -h employee | grep -i buffer
        Page buffers            0

tux> # Datenbank wiederherstellen und Cache-Größe anpassen
tux> gfix -shut -tran 60 employee
tux> gbak -replace -buffer 200 /backups/employee.fbk employee

tux> # Neue Cache-Größe anpassen (als root-Benutz!)
tux> gstat -h employee | grep -i buffer
        Page buffers            200
```

Die Standard-Cache-Größe wird verwendet, wenn die Anzahl der Puffer Null ist, wie im ersten Beispiel oben. Mit `gbak` kann dies bei Bedarf geändert werden. `gbak` kann die Cache-Größe jedoch nicht auf Null zurücksetzen. Sie müssen dazu `gfix` verwenden.

6.7. Anpassen der Seitengröße

Ähnlich wie im obigen Beispiel zum Ändern der Standardgröße des Datenbankcaches kann die Größe der Datenbankseite auch mit `gbak` geändert werden.

```
tux> # Prüfe die aktuelle Seitengröße (als root-Benutzer!)
tux> gstat -h employee | grep -i "page size"
        Page size               4096

tux> # Stelle die Datenbank wieder her und ändere die Seitengröße.
tux> gfix -shut -tran 60 employee
tux> gbak -replace -page_size 8192 /backups/employee.fbk employee

tux> # Prfe die neue Seitengröße (als root-Benutzer!)
tux> gstat -h employee | grep -i "page size"
        Page size               8192
```

6.8. Schreibgeschützten Datenbankklon erstellen

Manchmal möchten Sie nicht, dass Ihre Berichtersteller intensive Abfragen für Ihre Produktionsdatenbank ausführen. Zu diesem Zweck können Sie ganz einfach täglich einen Klon Ihrer Produktionsdatenbank erstellen und schreibschützen. Auf diese Weise kann das Berichtsteam so viele intensive Berichte erstellen, wie es möchte, ohne dass dies negative Auswirkungen auf die Produktionsdatenbank hat, und es wird verhindert, dass versehentlich Änderungen vorgenommen werden.

Das folgende Beispiel zeigt die Produktionsmitarbeiterdatenbank, die auf dem Linux-Server *tux* ausgeführt wird und auf den Linux-Server *tuxrep* des Berichtsteams geklont wird. Zuerst auf dem Produktionsserver:

```
tux> # Sicherung der Produktions-Datenbank.
tux> gbak -backup employee /backups/employee.fbk
```

Dann auf dem *tuxrep*-Server des Berichtsteams:

```
tuxrep> # Kopiere die Dump-Datei vom tux-Server mittels Scp
tuxrep> scp fbuser@tux:/backups/employee.fbk ./
Using keyboard-interactive authentication.
Password:
employee.fbk              |         19 kB |  19.3 kB/s | ETA: 00:00:00 | 100%

tuxrep> # Mitarbeiter-DB schreibgeschützt wiederherstellen
tuxrep> gfix -shut -tran 60 employee
tuxrep> gbak -replace -mode read_only employee.fbk employee

tuxrep> # Datenbankmodus prüfen (als root-Benutzer!)
tuxrep> gstat -h employee|grep -i attributes
        Attributes              no reserve, read only
```

6.9. Datenbankklon ohne Dump-Datei erstellen

Sie können `gbak` verwenden, um einen Klon einer Datenbank auf demselben Server zu erstellen, ohne eine potenziell große Dump-Datei erstellen zu müssen. Zu diesem Zweck leiten Sie die Ausgabe einer `gbak`-Sicherung wie folgt direkt an die Eingabe einer `gbak`-Wiederherstellung weiter.

```
tux> # Klonen der Testdatenbank auf denselben Server, ohne dass eine Dump-Datei
erforderlich ist.
tux> gbak -backup emptest stdout | gbak -replace stdin emptest_2
```

Sie werden feststellen, dass der Name der Ausgabedatei für die Sicherung `stdout` und der Name der Eingabedatei für die Wiederherstellung `stdin` lautet. Durch diese Möglichkeit, die Standardausgabe eines Prozesses an die Standardeingabe eines anderen Prozesses weiterzuleiten, können Sie

vermeiden, dass eine Zwischen-Dump-Datei erstellt wird. Die obigen Befehle setzen voraus, dass geeignete Aliasnamen sowohl für emptest als auch für emptest_2 eingerichtet sind. Wenn nicht, müssen Sie den vollständigen Pfad zu den beiden Datenbanken und nicht den Alias angeben.

Die Option `-replace` beim Wiederherstellungsprozess überschreibt den angegebenen Datenbanknamen — als Alias oder als vollständigen Pfad --, falls vorhanden, und erstellt ihn neu, wenn dies nicht der Fall ist. Alternativ können Sie auch die Option `recreate overwrite` verwenden. Beide haben das gleiche Ergebnis.

Wenn Sie keine vorhandenen Datenbanken überschreiben möchten, verwenden Sie `-create`. Dadurch wird eine Datenbank nur erstellt, wenn sie noch nicht vorhanden ist, und wenn dies der Fall ist, wird sie mit einem Fehler beendet. In POSIX-kompatiblen Systemen ist der Fehlercode in `$?` In diesem Fall 1.

Weitere Beispiele für das Sichern und Wiederherstellen von Remote-Datenbanken über ssh unter Verwendung der Dateinamen stdin und stdout finden Sie unten.

6.10. Sichern und Wiederherstellen mit und ohne Schattendateien.

An Datenbanken können bei normaler Verwendung Schattendateien angehängt werden. `gbak` sichert diese glücklicherweise mit und stellt sie ebenfalls wieder her. Bei normaler Verwendung werden Schattendateien neu erstellt. Wenn Sie nur die Datenbank wiederherstellen und die Schatten ignorieren möchten, kann `gbak` dies für Sie tun, wie das folgende Beispiel zeigt.

```
tux> # Prüfe die aktuellen Schattendateien und nutze isql, da gstat fehlerhaft ist
tux> isql employee

Database:  employee
SQL> show database;
Database: employee
        Owner: SYSDBA
 Shadow 1: "/opt/firebird/shadows/employee.shd1" manual
 Shadow 2: "/opt/firebird/shadows/employee.shd2" manual
...

SQL> quit;

tux> # Datenbank ohne Schattendateien wiederherstellen
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite /backups/employee.fbk employee

tux> # Prüfe die Schattendateien erneut, nutze isql, da gstat fehlerhaft ist
tux> isql employee

Database:  employee
SQL> show database;
Database: employee
        Owner: SYSDBA
 Shadow 1: "/opt/firebird/shadows/employee.shd1" manual
 Shadow 2: "/opt/firebird/shadows/employee.shd2" manual
...

SQL> quit;


tux> # Datenbank wiederherstellen und eliminiere Schattendateien
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite -kill /backups/employee.fbk employee

tux> # Prüfe die Schattendateien erneut, nutze isql, da gstat fehlerhaft ist
tux> isql employee

Database:  employee
SQL> show database;
Database: employee
        Owner: SYSDBA
...

SQL> quit;
```

Ich benutze `isql` in den obigen Beispielen, da `gstat -h` verwirrt darüber zu sein scheint, wie viele Schatten sich in einer Datenbank befinden. Es meldet Null, wenn es zwei gibt, holt schließlich auf und meldet, dass es zwei gibt. Wenn Sie einen Schatten entfernen, meldet es, dass es jetzt drei gibt!

6.11. Remote-Sicherungen und -Wiederherstellungen

Das Dienstprogramm `gbak` von Firebird kann Backups einer entfernten Datenbank erstellen. Dazu müssen Sie eine Verbindung zum Service Manager herstellen, der auf dem Remote-Server ausgeführt wird. Dies wird normalerweise als `service_mgr` bezeichnet. Das folgende Beispiel zeigt, wie die Firebird-Mitarbeiterdatenbank auf dem Server *tuxrep* vom Server *tux* gesichert wird. Die Sicherung wird auf den Remote-Server geschrieben, dh die Sicherungsdatei wird auf dem Server *tuxrep* und nicht auf dem Server *tux* erstellt. Das verwendete Netzwerkprotokoll ist TCP.

```
tux> # Berichtsdatenbank auf Remote-Server tuxrep sichern
tux> gbak -backup -service tuxrep:service_mgr employee /backups/remote_backup.fbk
```

Die Sicherungsdatei hat denselben Eigentümer und dieselbe Gruppe wie der Firebird-Datenbankserver — zumindest auf Unix-Systemen.

Es ist auch möglich, eine entfernte Datenbank auf diese Weise wiederherzustellen, und `gbak` erlaubt dies.

```
tux> # Stelle die schreibgeschützte Berichtsdatenbank auf dem Remote-Server tuxrep
wieder her.
tux> gbak -replace -mode read_only -service tuxrep:service_mgr \
           /backups/remote_backup.fbk employee
```

Das obige Beispiel verwendet die praktische Unix-Fähigkeit, eine lange Zeile über viele kürzere zu teilen, wobei ein Schrägstrich als *letztes Zeichen* in der Zeile verwendet wird.

Wie immer wird empfohlen, sich vor dem Ersetzen einer Datenbank zu hüten, falls während der Wiederherstellung Probleme auftreten. Im obigen Beispiel wird die vorhandene Datenbank im schreibgeschützten Modus neu erstellt. Dies muss jedoch nicht immer der Fall sein.

Eine Remote-Sicherung kann auch auf dem Datenbankserver selbst ausgeführt werden! Unter Windows macht dies keinen Unterschied, aber auf Unix-Systemen reduziert diese lokal-entfernte Methode zum Sichern und Wiederherstellen den Netzwerkverkehr. Der 'Remote'-Server ist in diesem Fall nicht wirklich entfernt. Es ist nur die Methode zum Ausführen der Sicherung — herstellen einer Verbindung zum Service Manager --, die die Entfernung impliziert.

```
tux> # Sichern der Mitarbeiterdatenbank auf diesem Server, aber pseudo-remote!
tux> gbak -backup -service tux:service_mgr employee /backups/remote_backup.fbk
```

Entsprechende Wiederherstellungen können auch 'remote' ausgeführt werden:

```
tux> # Restore the employee database on this server, but pseudo-remotely!
tux> gbak -replace -service tux:service_mgr /backups/remote_backup.fbk employee
```

Das Format des Parameters, der für den `service`-Schalter verwendet wird, unterscheidet sich je nach Art des verwendeten Netzwerkprotokolls:

TCP

Bei Verwendung von TCP-Netzwerken ist das Parametertrennzeichen ein Doppelpunkt, wie in `-service server_name:service_mgr`.

Named pipes

Bei Verwendung von `Named Pipes` benötigt der Parameter zwei führende Backslashes und der Separator selbst ist ebenfalls ein Backslash, wie in `-service \\server_name\service_mgr`.

6.12. Remote-Sicherungen und Wiederherstellungen mit SSH

Wie oben gezeigt, können Sie die speziellen Dateinamen stdin und stdout verwenden, um eine Datenbank in einer separaten Datenbank auf demselben Server zu sichern und wiederherzustellen. Sie können jedoch auch dieselben Tools über eine SSH-Verbindung zu einem Remoteserver verwenden und die Sicherung einer Datenbank direkt an eine Wiederherstellung einer separaten Datenbank übergeben.

Im ersten Beispiel wird eine lokale Datenbank auf einen Remote-Server kopiert, auf dem Firebird ausgeführt wird und die Umgebung des Firebird-Benutzers so eingerichtet ist, dass sich das gbak-Tool bei der Anmeldung standardmäßig auf `$PATH` befindet.

In jedem der folgenden Beispiele wurden die Parameter `-user sysdba` und `-password whatever` in den Befehlszeilen durch {...} ersetzt. Wenn Sie diese Befehle ausführen, müssen sie für alle `gbak`-Remotebefehle angegeben werden, es sei denn, der Firebird-Benutzer der/den Remote-Datenbank(en) hat die Werte für `ISC_USER` und `ISC_PASSWORD` in `.profile` oder `.bashrc` (oder äquivalente Anmeldedateien) gesetzt. Dies ist jedoch eine *wirklich* schlechte Idee und unglaublich unsicher.

```
tux> # Klone eine Testdatenbank auf einen anderen Server, ohne dass eine Dump-Datei
erforderlich ist
tux> gbak -backup employee stdout | \
ssh firebird@tuxrep "gbak {...} -replace stdin emptest"
```

Wenn das oben Gesagte ausgeführt wird, werden Sie aufgefordert, ein Kennwort für den Remote-Firebird-Benutzer auf dem Server *tuxrep* einzugeben, vorausgesetzt, Sie haben noch kein ordnungsgemäßes SSH-Schlüsselpaar eingerichtet und sind aktiv. Der Befehl ersetzt die lokale Datenbank gemäß dem Aliasnamen `emptest`. Sie können jedoch bei Bedarf vollständige Pfadnamen

für die Datenbanken angeben. Das Folgende zeigt ein Beispiel für die Ausführung des oben genannten Vorgangs.

```
tux> # Klonen eine Testdatenbank auf einen anderen Server, ohne dass eine Dump-Datei
erforderlich ist
tux> gbak -backup employee stdout | \
ssh firebird@tuxrep "gbak {...} -replace stdin emptest"

firebird@tuxrep's password:
```

Wie Sie sehen, steht der Ausgabe nicht viel im Wege, aber Sie können eine Remoteverbindung herstellen und Folgendes überprüfen:

```
tux> isql {...} tuxrep:emptest

Database:  tuxrep:emptest

SQL> show database;

Database: tuxrep:emptest
        Owner: SYSDBA
PAGE_SIZE 4096
...
```

Das nächste Beispiel zeigt eine Remote-Datenbank, die auf ähnliche Weise in einer lokalen Datenbank gesichert wird.

```
tux> ssh firebird@tuxrep "gbak -backup {...} emptest stdout" | \
gbak -create stdin data/tuxrep_emptest.fdb

firebird@tuxrep's password:

tux> ls data

employee.fdb  tuxrep_emptest.fdb
```

Sie können sehen, dass eine neue Datenbank `tuxrep_emptest.fdb` erstellt wurde. Funktioniert sie? Die Überprüfung mit `isql` zeigt, dass dies der Fall ist.

```
tux> isql data/tuxrep_emptest.fdb

Database:  data/tuxrep_emptest.fdb

SQL> quit;
```

Das letzte Beispiel zeigt, wie eine entfernte Datenbank auf einem Server in einer entfernten

Datenbank auf einem anderen Server gesichert wird.

```
tux> ssh firebird@tuxrep "gbak -backup {...} emptest stdout" |  \
ssh firebird@tuxqa "gbak -create {...} stdin data/tuxrep_empqa.fdb"

firebird@tuxrep's password:
firebird@tuxqa's password

tux> ssh firebird@tuxqa "ls data"

employee.fdb  tuxrep_empqa.fdb
```

6.13. Externe Tools verwenden

`gbak` und `nbackup` sind die besten Tools zum Sichern und / oder Wiederherstellen von Firebird-Datenbanken. Sie wurden ausgiebig getestet und kennen die Interna der Datenbank und deren Funktionsweise. Daher ist die Wahrscheinlichkeit, dass diese Tools Ihre wertvollen Daten beschädigen, sehr gering. Einige Datenbankadministratoren verwenden jedoch weiterhin gerne externe Tools (die nicht mit Firebird geliefert werden), um aus irgendeinem Grund Backups zu erstellen.

Da externe Tools anhand des Aliasnamens nicht wissen können, wo sich eine Datenbank befindet, müssen der Skriptschreiber und / oder der Datenbankadministrator explizit den korrekten Speicherort der Datenbankdatei(en) ermitteln und diese an die externe Datei weitergeben Werkzeug. Um dies für Skriptautoren zu vereinfachen, verwendet meine eigene Installation einen Standard in meiner Datei "aliases.conf" wie folgt:

- Der Datenbankalias muss in Spalte eins beginnen.
- Vor dem Gleichheitszeichen (=) muss ein Leerzeichen stehen.
- Nach dem Gleichheitszeichen (=) muss ein Leerzeichen stehen.
- Doppelte Anführungszeichen um den Datenbankdateinamen sind nicht zulässig - dies funktioniert auch nicht für die Firebird-Dienstprogramme.
- Datenbanken sind alle Einzeldateidatenbanken.

Die letzte Regel gilt nur für meine Installation und bedeutet, dass das folgende einfache Sicherungsskript funktioniert. Wenn mehrere Dateidatenbanken verwendet würden, wäre mehr Code erforderlich, um eine Sicherung mit externen Tools durchzuführen.

```
tux> cat /opt/firebird/aliases.conf
# ------------------------------------------------------
# WARNUNG: Backup-Standards erfordern Folgendes:
#          Der Datenbankname beginnt in Spalte 1.
#          Vor dem Gleichheitszeichen steht ein einzelnes Leerzeichen.
#          Nach dem Gleichheitszeichen steht ein einzelnes Leerzeichen.
#          Der Pfad hat keine doppelten Anführungszeichen (sie funktionieren nicht!)
# ------------------------------------------------------
employee = /opt/firebird/examples/empbuild/employee.fdb
```

Im Folgenden wird die Verwendung des Dienstprogramms "gzip" auf einem Linux-Server zum Erstellen und Komprimieren einer Sicherung einer laufenden Datenbank gezeigt. Folgendes wird als Root-Benutzer ausgeführt, da `gfix` ausgeführt werden muss, um die Datenbank herunterzufahren.

```
tux> # Backup der Produktiv-Mitarbeiter-DB mit gzip
tux> gfix -shut -tran 60 employee
tux> DBFILE=`grep -i "^employee =" /opt/firebird/aliases.conf | cut -d" " -f3`
tux> gzip -9 --stdout $DBFILE > /backups/employee.fdb.gz
```

Der Wiederherstellungsprozess für diese Datenbank wäre umgekehrt. Wieder läuft das Folgende als root.

```
tux> # Wiederherstellen der Produktiv-Mitarbeiter-DB aus dem gzip-Backup
tux> gfix -shut -tran 60 employee
tux> DBFILE=`grep -i "^employee =" /opt/firebird/aliases.conf | cut -d" " -f3`
tux> gunzip --stdout /backups/employee.fdb.gz > $DBFILE

tux> # Stelle sicher, dass Firebird die Datei sehen kann.
tux> chown firebird:firebird $DBFILE
```

Kapitel 7. gbak-Fallstricke

Das Folgende ist eine kurze Liste von Fallstricken und Funnies, die ich bei meiner eigenen Verwendung von `gbak` entdeckt habe. Einige davon sind oben erwähnt, andere möglicherweise nicht. Wenn Sie sie alle hier an einem Ort sammeln, sollten Sie herausfinden können, was passiert, wenn Sie Probleme haben.

7.1. gbak-Standardmodus

Wenn Sie keinen Modusschalter wie `-b[ackup]` oder `-c[reate]` usw. angeben, führt `gbak` eine Sicherung durch, als ob der Schalter `-b[ackup]` angegeben worden wäre — vorausgesetzt, die anderen angegebenen Schalter sind für eine Sicherung korrekt.

Das Erkennen, ob Sie eine Sicherung oder eine Wiederherstellung durchführen möchten, bedeutet, dass Sie, wenn Sie den Befehlszeilenschalter `-z` verwenden, um `gbak`-Informationen anzuzeigen, eine Sicherung erstellen und die von Ihnen angegebene Sicherungsdatei überschreiben, wenn in der Befehlszeile auch Datenbankname und Name der Sicherungsdatei vorhanden sind. Dies setzt voraus, dass es für `gbak` eine Möglichkeit gibt, den Benutzernamen und das Kennwort zu bestimmen, die verwendet werden sollen — entweder als Befehlszeilenparameter oder über definierte Umgebungsvariablen.

7.2. Normaler vs. privilegierter Modus

Nur der SYSDBA oder der Eigentümer einer Datenbank kann eine Sicherung der Datenbank erstellen. Jeder authentifizierte Benutzer kann jedoch eine Datenbanksicherung mit dem Schalter `-c[reate]` wiederherstellen. Dies bedeutet, dass Sie sicherstellen müssen, dass Ihre Sicherungsdateien nicht in die falschen Hände geraten, da nichts dann Unbefugte daran hindert, Ihre Daten zu sehen, indem Sie einfach Ihre Sicherungen auf Ihrem Server wiederherstellen.

Die Datenbankwiederherstellung schlägt natürlich fehl, wenn der Benutzer, der sie ausführt, nicht der Datenbankeigentümer ist und bereits eine Datenbank mit demselben Dateinamen vorhanden ist.

7.3. Leise laufenlassen?

Der Schalter `-y suppress_output` unterdrückt jegliche Ausgaben. Ähnlich wie beim Ausführen mit `-v[erify]` nicht angegeben. Es scheint jedoch nur zu bewirken, dass die Ausgabe (gemäß der Schaltereinstellung `-v[erify]`) in eine Datei mit dem Namen `suppress_output` geschrieben wird. Dies funktioniert jedoch nur einmal, da der nächste Durchlauf von gbak mit `-y suppress_output` fehlschlägt, da die Datei `suppr_output` bereits vorhanden ist.

Es ist möglich, dass dieses Problem in Version 2 für Firebird eingeführt wurde, da sowohl die 2.0- als auch die 2.1-Version tatsächlich den Schalter `-y suppress` anstelle von `-y suppress_output` verwenden. Die Verwendung dieser (kürzeren) Option funktioniert wie beabsichtigt und die Ausgabe wird tatsächlich unterdrückt.

7.4. gbak-Protokolldatei kann nicht überschrieben werden

Wenn Sie mit dem Schalter `-y <Protokolldatei>` einen Protokolldateinamen angeben und die Datei bereits vorhanden ist, kann `gbak` sie nicht überschreiben, obwohl der Firebird-Benutzer die Datei besitzt und über Schreibberechtigungen verfügt. Sie müssen immer den Namen einer Protokolldatei angeben, die nicht vorhanden ist. Auf Linux-Systemen kann Folgendes hilfreich sein:

```
tux> # Erstelle eindeutige Log-Datei
tux> FILENAME=employee_`date "+%Y%m%d_%H%M%S"`

tux> # Datenbank herunterfahren und sichern
tux> gfix -shut -tran 60 employee
tux> gbak -backup employee /backups/${FILENAME}.fbk -y /logs/${FILENAME}.log -v
```

Dies ist insofern sehr nützlich, als es Sie daran hindert, frühere Sicherungen zu überschreiben, die möglicherweise erforderlich sind. Der Nachteil ist, dass Sie jetzt ein Reinigungssystem einführen müssen, um alte, unerwünschte Sicherungen zu beseitigen und zu verhindern, dass sich Ihr Sicherungsbereich füllt.

7.5. Verwenden von 'stdin' oder 'stdout'-Dateinamen

`gbak` erkennt die Literalzeichenfolgen 'stdin' und 'stdout' als Quell- oder Zieldateinamen. In POSIX-Systemen ist es bei Verwendung der Standardeingabe- und / oder Standardausgangskanäle nicht zulässig, Suchoperationen auf diesen Kanälen auszuführen. Wenn Sie 'stdin' oder 'stdout' als Dateinamen mit `gbak` verwenden, wird 'gbak' gezwungen, eine Verarbeitung zu verwenden, die nicht auf den Eingangs- oder Ausgangskanälen sucht, sodass sie für die Verwendung in Pipes geeignet sind — wie in den Beispielen in den Rezepten angegeben Abschnitt oben.

Diese Dateinamen scheinen zwar POSIX-Namen zu sein, sind aber definitiv keine Synonyme für `/dev/stdin` oder `/dev/stdout`. Sie sind einfach Literale, nach denen `gbak` bei der Verarbeitung seiner Parameter sucht. Versuchen Sie nicht, die Namen `/dev/stdin` oder `/dev/stdout` in einem Pipeline-Prozess zu verwenden, da dies höchstwahrscheinlich fehlschlagen wird.

Wenn Sie eine Dump-Datei mit dem Namen stdin oder stdout erstellen möchten, sollten Sie den Dateinamen als vollständigen oder relativen Pfadnamen wie `./stdin` oder `./stdout` angeben, wodurch "gbak" ausgelöst wird Behandeln Sie sie als wörtlichen Dateinamen und nicht als speziellen Dateinamen, der sich von der normalen Verarbeitung während des Dump- oder Wiederherstellungsprozesses unterscheidet.

Anhang A: Dokumentenhistorie

Der genaue Dateiversionsverlauf wird im Firebird-Dokumentations-Git-Repository aufgezeichnet; siehe https://github.com/FirebirdSQL/firebird-documentation

Revisionshistorie

1.0	10. Okt. 2009	ND	Erstellt als Kapitel im Handbuch der Befehlszeilen-Dienstprogramme.
1.1	20. Okt. 2009	ND	Weitere kleinere Updates und konvertiert in ein eigenständiges Handbuch.
1.2	24. Nov. 2009	ND	Der Wiederherstellungsoption `-o[ne_at_a_time]` wurden weitere Details hinzugefügt, um Transaktionen zu erläutern.
1.3	24. Jun. 2010	ND	Weitere Details zur Wiederherstellungsoption `-o[ne_at_a_time]`, um Transaktionen zu erläutern.
1.4	09. Aug. 2010	ND	Es wird darauf hingewiesen, dass `gbak` standardmäßig eine Sicherung oder Wiederherstellung gemäß dem ersten angegebenen Dateinamenparameter ausführt. Einige kleinere Formatierungsfehler, URLs und einige Beispiele wurden korrigiert. Außerdem wurde ein Beispiel für eine Sicherung und Wiederherstellung nur von Metadaten hinzugefügt.
1.5	31. Mär. 2011	ND	Die Option `-z` wurde aktualisiert, um anzuzeigen, dass eine Sicherung durchgeführt wird.
1.6	11. Okt. 2011	ND	Aktualisiert, um Änderungen an Firebird 2.5 abzudecken. Richtige Beschreibung des Schalters `-g[arbage_collect]`. Viele Rechtschreibfehler wurden korrigiert.
1.7	11. Jan. 2013	ND	Aktualisiert, um die Verwendung der Dateinamen stdin und stdout in Sicherungen und Wiederherstellungen zu dokumentieren, mit denen Sicherungen in Standardeingaben und Standardausgaben geschrieben oder von diesen gelesen werden können. Es wurde ein Abschnitt über die Verwendung der oben genannten Informationen zum Klonen von Datenbanken hinzugefügt, ohne dass eine Zwischen-Dump-Datei erforderlich ist. Ein zusätzlicher Abschnitt wurde hinzugefügt, um zu zeigen, wie in Verbindung mit SSH Sicherungs- und / oder Wiederherstellungsvorgänge für Datenbanken ausgeführt werden können, bei denen eine oder beide der fraglichen Datenbanken remote sind.

Revisionshistorie

1.8	14. Jan. 2013	ND	Weitere Aktualisierungen zur Dokumentation der Verwendung der Dateinamen stdin und stdout bei Sicherungen und Wiederherstellungen. Gbak Caveats wurde ein Abschnitt hinzugefügt, der detailliertere Informationen zu diesen beiden speziellen Dateinamen enthält.
1.9	11. Apr. 2013	ND	Es wurde ein Abschnitt hinzugefügt, in dem erläutert wird, wie Sie Ihre Backups beschleunigen können. Der Option `-service` wurde ein Hinweis hinzugefügt, um zu erläutern, dass die Verwendung nicht auf entfernte Datenbanken beschränkt ist. Syntaxfehler in einigen Beispielen korrigiert.
1.10	1. Mai 2013	ND	Leichte Aktualisierung des Befehlszeilenschalters `-use_[all_space]`, um zu erklären, wie es verständlicher funktioniert.
1.11	1. Mai 2013	ND	Eine Korrektur der obigen Änderung am Befehlszeilenschalter `-use_[all_space]` — dies betrifft alle nachfolgenden Seiten sowie die während der Wiederherstellung erstellten Seiten.
1.12	18. Jun. 2020	MR	Konvertierung in AsciiDoc, geringfügige Bearbeitung von Texten
1.12-de	28. Jul. 2020	MK	Deutsche Übersetzung.

Anhang B: Lizenzhinweis

Die Original-Dokumentation trägt den Titel *Firebird Backup & Restore Utility*.

Der ursprüngliche Autor der Original-Dokumentation ist: Norman Dunbar.

Mitwirkende: Norman Dunbar; Mark Rotteveel; Martin Köditz - siehe Dokumenthistorie.