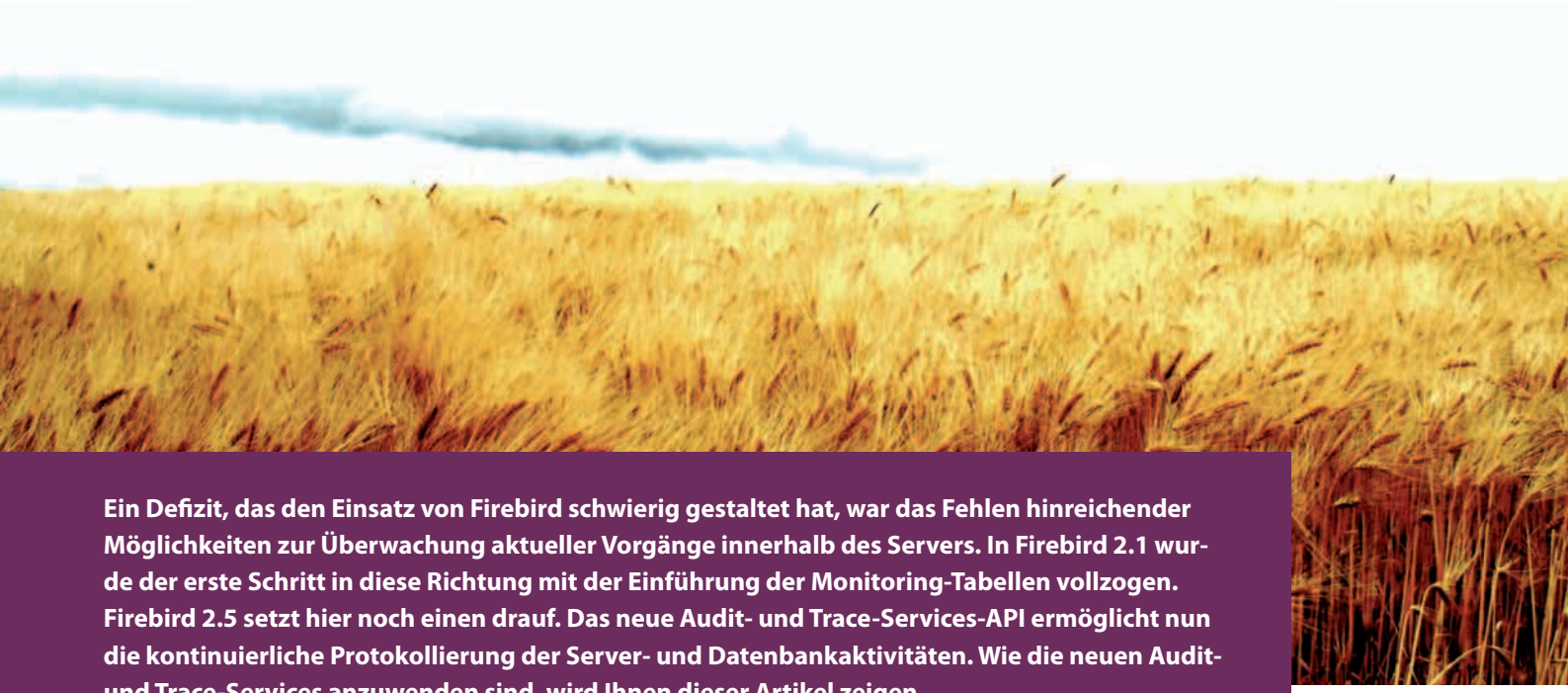
Den neuen Möglichkeiten des Monitorings auf den Zahn gefühlt

Audit- und Trace-Services in Firebird 2.5



Ein Defizit, das den Einsatz von Firebird schwierig gestaltet hat, war das Fehlen hinreichender Möglichkeiten zur Überwachung aktueller Vorgänge innerhalb des Servers. In Firebird 2.1 wurde der erste Schritt in diese Richtung mit der Einführung der Monitoring-Tabellen vollzogen. Firebird 2.5 setzt hier noch einen drauf. Das neue Audit- und Trace-Services-API ermöglicht nun die kontinuierliche Protokollierung der Server- und Datenbankaktivitäten. Wie die neuen Audit- und Trace-Services anzuwenden sind, wird Ihnen dieser Artikel zeigen.

von Thomas Steinmaurer

Mit den in 2.1 eingeführten Monitoring-Tabellen (*MON$*) besitzt man die Möglichkeit, die aktuell laufenden Aktivitäten innerhalb einer Datenbank abzufragen, z. B.:

- Informationen zu einer Datenbank wie Dateiname, Page Size, ODS-Version, Sweep-Intervall, OIT, OAT etc.
- Aktuell offene Datenbankverbindungen mit Clientinformationen (IP-Adresse, Prozessname etc.)
- Überblick über alle Transaktionen
- Überblick über alle SQL-Anweisungen
- I/O- und Datensatzstatistiken auf Datenbank-, Verbindungs-, Transaktions- und Anweisungsebene

Anweisungen können mit einem *DELETE FROM MON$STATEMENTS…* einfach beendet werden. Dies ist vor allem dann nützlich, wenn Abfragen mit schlechter Laufzeit vorzeitig beendet werden sollen. In 2.5 sind zwei neue Monitoring-Tabellen *MON$MEMORY_USAGE* und *MON$CONTEXT_VARIABLES* hinzugekommen. Des Weiteren ist in 2.5 nun auch das Beenden einer Datenbankverbindung mit allen dazugehörigen Transaktionen und Anweisungen durch ein *DELETE* auf *MON$ATTACHMENTS* möglich. Ich berichtete darüber in meinem Firebird-2.5-Artikel [1].

Continuous Monitoring

Will man nun einen zeitlichen Verlauf der Datenbankzustände für spätere Analysezwecke sicherstellen, so muss man hier im Fall der Monitoring-Tabellen selbst Hand anlegen, indem man sie periodisch abfragt und ihre Inhalte speichert. Über eine Art der „Versionierung" der Monitoring-Tabellen kann dies bewerkstelligt werden. In einem meiner Blogeinträge [2] diskutiere ich hierfür eine einfache Umsetzungsvariante. Durch das notwendige Polling dieser Tabellen erhält man allerdings nur bedingt eine kontinuierliche Aufzeichnung, da naturgemäß Lücken in der Aufzeichnung existieren, abhängig davon, wie klein die Zeitscheibe für das Polling gewählt wurde. Vor 2.5 musste man für eine kontinuierliche Aufzeichnung auf Fremdprodukte zurückgreifen, die sich im Wesentlichen in zwei Produktkategorien einordnen lassen:

- Proxy-Anwendung, die sich in den TCP/IP-Netzwerkverkehr zwischen der Clientanwendung und dem Firebird-Server einklinkt, den Netzwerkverkehr abhört und die Daten entsprechend dem Firebird-Client/Server-Kommunikationsprotokoll dekodiert
- API-Call-Interceptor-Anwendung, die API-Aufrufe von der Clientanwendung zur Firebird-Clientbibliothek abfängt

Das neue Audit- und Trace-Services-API in Firebird 2.5 ermöglicht nun out-of-the-box ein Trace auf Anweisungsebene mit den folgenden Eigenschaften:

- Kontinuierlich
- Serverseitig
- Konfigurierbar

Ein Auszug der unterstützten Anweisungen:

- Connect/Disconnect
- Start/Commit/Rollback einer Transaktion
- Prepare/Start/Beenden/Freigeben einer SQL-Anweisung
- Start/Beenden einer Stored Procedure bzw. eines Triggers

Ein Audit/Trace ersetzt allerdings nicht die Monitoring-Tabellen, da man mit diesen einen Snapshot des aktuellen Datenbankzustands erhält, wohingegen ein Audit/Trace mit einem sequenziellen Strom an protokollierten Anweisungen verglichen werden kann. Ist zum Beispiel die Fragestellung: „Anzahl der aktuell aktiven Transkationen in einer Datenbank“ über die Monitoring-Tabellen mit nur einer SELECT-Anweisung auf *MON$TRANSACTIONS* abrufbar, so kann dies mit den durch ein Audit/Trace angefallen Daten nur durch erweiterte Analysemethoden bewerkstelligt werden (wenn überhaupt). Beispielhaft nachfolgend ein paar Fragestellungen, die für ein Audit/Trace sprechen:

- Anzahl der ausgeführten Anweisungen in einer Zeitperiode
- Ausführungs-Trace einer bestehenden Anwendung (Blackbox-Debugging)
- Wird *COMMIT RETAINING* (Soft Commit) oder *COMMIT* (Hard Commit) verwendet. Hiermit kann man etwaige Performanceprobleme ableiten
- Benutzungsstatistiken für Ressourcen- und Lastplanung
- Protokollierte Anweisungen als Input für ein Security-Audit
- und mehr

Sowohl die Monitoring-Tabellen als auch der Audit/Trace-Ansatz in Firebird 2.5 ersetzen allerdings auch Trigger-basierte DML-Audit-Mechanismen nicht, da zum Beispiel im Fall von IB LogManager [3] geänderte Datenwerte in strukturierter Form (alter/neuer Feldwert) abgespeichert werden. In [4] habe ich einen trigger-basierten Ansatz vorgestellt. Es stehen somit drei Ansätze für das Monitoring mit Firebird-Bordmitteln zur Verfügung, die in Tabelle 1 gegenübergestellt werden.

Je nach Anforderung an das Monitoring hat sich in der Praxis eine Kombination aus diesen unterschiedlichen Ansätzen bewährt. Genug der einleitenden Worte und der Motivation. Sehen wir uns nun die neuen Audit- und Trace-Services näher an. Die Firebird 2.5 Release Notes widmen sich diesem sehr interessanten neuen Feature leider nur sehr spärlich. Grund genug, dass Sie dran bleiben!

System Audit vs. User Trace

Firebird 2.5 unterscheidet zwischen einem *System Audit* und einem *User Trace*. Ein System Audit wird beim Start des Firebird-Servers automatisch gestartet. Das bedeutet, dass für das Starten keine Benutzerinteraktion notwendig ist. Dafür zuständig ist ein neuer Parameter *AuditTraceConfigFile* in *firebird.conf*, der den Pfad zu einer Trace-Konfigurationsdatei angibt. Ist dieser Parameter gesetzt (und das vorangestellte # entfernt), dann lädt der Firebird-Server beim Startup diese Datei und initiiert den Start des System-Audits basierend auf den darin enthaltenen Einstellungen. Zu den einzelnen Konfigurationsparametern komme ich etwas später. Die protokollierten Anweisungen landen im Fall eines System-Audits in einer (Text-) Datei im Dateisystem am Server. Im Gegensatz dazu wird ein User Trace explizit

Ansatz	Trigger-basiert	Monitoring-Tabellen	Audit/Trace Services
Verfügbar seit Firebird-Version	1.0	2.1	2.5
Kontinuierliche Aufzeichnung	Ja	Nein	Ja
Serverseitiger Ansatz	Ja	Ja	Ja
Konfigurierbar	Ja	Nein	Ja
Monitoring von SELECT-Anweisungen	Nein	Ja	Ja
Monitoring von DELETE-, INSERT- und UPDATE-Anweisungen	Ja	Ja	Ja
Speicherung alter/neuer Feldwert bei DELETE, INSERT und UPDATE	Ja	Nein	Nein
Speicherung von Ausführungszeiten	Nein	Nein	Ja
Speicherung von Ausführungsplänen	Nein	Nein	Ja
Monitoring von datenbankweiten Operationen wie Connect, Disconnect, Transaktionsstart, Transaktionsende, Prepare Statement, Free Statement, …	Teilweise (seit 2.1 mit Datenbank-Triggern)	Teilweise	Ja

Tabelle 1: Monitoring-Ansätze für Firebird-Datenbanken

durch einen Benutzer unter Angabe einer Trace-Konfiguration gestartet. Die protokollierten Anweisungen werden in diesem Fall vom Firebird-Server nur temporär in eine Datei am Server geschrieben, bis die Clientanwendung, mit der ein User Trace gestartet wurde, die protokollierten Anweisungen über das Services-API empfängt. Ein User Trace überlebt einen Restart des Firebird-Serverprozesses nicht.

Ein Beispiel sagt mehr als 1 000 Worte

Mit dem neuen Kommandozeilentool *fbtracemgr*, zu finden im *bin*-Unterverzeichnis der Firebird-Installation, steht dem Benutzer eine rudimentäre Möglichkeit für die Verwendung der Audit- und Trace-Services zur Verfügung. Dieses neue Utility bietet die folgende Funktionalität:

- Starten eines User Trace und Lesen der Trace-Ausgabe
- Beenden einer Trace-Session
- Anhalten und Fortfahren einer Trace-Session
- Anzeige einer Liste aller gestarteten Trace-Sessions

Anhand des nachfolgenden Beispiels soll man ein Gefühl für die Verwendung von *fbtracemgr* und die Form des Trace-Outputs bekommen. Die Trace-Konfigurationsdatei beinhaltet die folgenden Zeilen:

```
<database employee.fdb>
 enabled true
 log_connections true
 log_transactions true
</database>
```

log_connections ist ein möglicher Parameter unter vielen, der im Konkreten die Protokollierung des Verbindungsaufbaus und des Verbindungsabbaus zu/von einer Datenbank *employee.fdb* aktiviert. Ein weiterer Parameter *log_transactions* aktiviert die Protokollierung von Start/Commit/Rollback-Ereignissen von Transaktionen. Wichtig ist, dass der Aufbau einer Konfigurationsdatei für ein System-Audit und ein User Trace identisch ist. *fbtracemgr* kann nun wie folgt verwendet werden, um basierend auf dieser Trace-Konfiguration ein User Trace zu starten (mein Firebird-2.5-Server läuft zu Testzwecken auf Port 3051).

```
fbtracemgr -se localhost/3051:service_mgr -user sysdba
 -password masterkey -start -name "User Trace 1" -config
                                "fbtrace.conf" > trace.out
```

Nach ein paar Clientaktivitäten entsteht beispielhaft der folgende Trace-Output:

```
Trace session ID 7 started
2009-12-29T11:57:37.4670 (2816:0148D81C) TRACE_INIT
 SESSION_7 User Trace 1
2009-12-29T11:57:37.4670 (2816:0148D81C)
                                        ATTACH_DATABASE
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
2009-12-29T11:57:37.4670 (2816:0148D81C)
                                        START_TRANSACTION
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
  (TRA_185, READ_COMMITTED | REC_VERSION | NOWAIT |
                                              READ_ONLY)
2009-12-29T11:57:37.4830 (2816:0148D81C)
                                        START_TRANSACTION
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
  (TRA_186, READ_COMMITTED | REC_VERSION | NOWAIT |
                                              READ_WRITE)
2009-12-29T11:57:37.5450 (2816:0148D81C)
                                        COMMIT_TRANSACTION
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
  (TRA_186, READ_COMMITTED | REC_VERSION | NOWAIT |
                                              READ_WRITE)
   49 ms, 1 read(s), 1 write(s), 1 fetch(es), 1 mark(s)
2009-12-29T11:57:39.3730 (2816:0148D81C)
                                        COMMIT_TRANSACTION
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
  (TRA_185, READ_COMMITTED | REC_VERSION | NOWAIT |
                                              READ_ONLY)
   0 ms
2009-12-29T11:57:39.3730 (2816:0148D81C)
                                        DETACH_DATABASE
 employee.fdb (ATT_16, SYSDBA:NONE, TCPv4:127.0.0.1)
 C:\Program Files\Upscene Productions\
                    Database Workbench 3 Pro\DBW3.exe:4568
2009-12-29T11:57:39.3730 (2816:0148D81C) TRACE_FINI
 SESSION_7 User Trace 1
```

Wie zu sehen ist, handelt es sich beim Trace-Output um Textdaten. Fett dargestellt sind die Ereignistypen, wie zum Beispiel *ATTACH_DATABASE* für einen Verbindungsaufbau. Jeder Ereignistyp besitzt einen Zeitstempel, wann das Ereignis aufgetreten ist, die Prozess-ID des Firebird-Serverprozesses (im obigen Beispiel: 2816), eine internen Trace-Nummer (im obigen Beispiel: 0148D81C) für die Identifikation der Trace-Session und die textuelle Bezeichnung des Ereignisses. Danach folgen ereignistypspezifische Detaildaten wie zum Beispiel der Datenbankname (*employee.fdb*), die Verbindungs-ID (*ATT_<nummer>*) und mehr. Die Ereignistypen können unterschiedliche Detaildaten aufweisen. So spielen zum Beispiel bei *START_TRANSACTION* der Isolation-Level der Transaktion (*READ_COMMITTED*) und weitere transaktionsspezifische Daten eine Rolle.

Dies alles ist zwar für den Menschen gut lesbar, allerdings schwer zu parsen, da man es hier leider nicht mit strukturierten Daten zu tun hat, wie man es zum Beispiel von XML gewohnt ist. Die Struktur der Daten ist hier implizit durch das mitgelieferte Firebird-*Trace-Plug-in* definiert. Ein Reverse-Engineering des Aufbaus ist nur durch Studium des originalen Trace-Plug-in-Quellcodes und durch eigene Tests möglich, da keine Dokumentation zur Struktur verfügbar ist. Da die Trace-Engine allerdings Plug-in-fähig ist, könnte man sich ein eigenes Trace-Plug-in schreiben, das strukturierten Text zurückliefert oder zum Beispiel protokollierte Anweisungen gleich in eine Trace-Datenbank einfügt. Dies setzt natürlich eine entsprechende Expertise in Bezug auf die Trace-Plug-in-Entwicklung voraus, vor allem, da auch hierfür keine Dokumentation erhältlich ist.

Firebird 2.5 wird standardmäßig mit einem Trace-Plug-in ausgeliefert, das sich im *plugins*-Unterverzeichnis der Firebird-Installation in Form einer DLL bzw. Shared-Object-Datei *fbtrace(.dll|.so)* befindet, das eben das oben gezeigte textbasierte Ausgabeformat implementiert hat. Das Firebird-Projekt hat allerdings bereits angekündigt, hier in späteren Versionen mit einem besser verarbeitbaren Format nachzubessern. Ich komme später noch auf ein neues kommerzielles Produkt *Firebird Trace Mana-*

ger zu sprechen, das das gezeigte Format verarbeiten, in eine strukturierte Form bringen und für die weitere Verarbeitung (z. B. Datenanalyse) aufbereiten kann.

Mit dem Trace- und Services-API ist natürlich auch die Auflistung aller gestarteten Audit/Trace-Sessions sowie das Anhalten, Fortführen und Stoppen möglich. *fbtracemgr* kann auch hierfür verwendet werden. Der *fbtracemgr*-Aufruf für die Auflistung aller gestarten Sessions kann wie folgt aussehen: *fbtracemgr -se*

Abschnitt	Parameter	Bedeutung	Werte
database	enabled	Aktiv/nicht aktiv	true | false
	log_filename	Pfad der Logdatei. Nur für System-Audit relevant	String
	max_log_size	Maximale Größe der Logdatei, bevor eine neue Logdatei erzeugt wird (Logrotation). Nur für System-Audit relevant	Integer
	include_filter	Nur SQL-Anweisungen protokollieren, die dem regulären Ausdruck entsprechen	String (regulärer Ausdruck)
	exclude_filter	SQL-Anweisungen explizit nicht protokollieren, die dem regulären Ausdruck entsprechen	String (regulärer Ausdruck)
	log_connections	Protokollierung von Verbindungsaufbau und Verbindungsabbau	true | false
	connection_id	Protokollierung nur für bestimmte Verbindungs-ID aktivieren (CURRENT_CONNECTION). Falls 0, dann für alle Verbindungen	Integer
	log_transactions	Protokollierung von Transaktionsanweisungen (Start, Commit, Rollback)	true | false
	log_statement_prepare	Protokollierung, wenn eine Anweisung durch ein *Prepare* in das BLR-Format übersetzt wurde	true | false
	log_statement_free	Protokollierung, wenn eine Anweisung freigegeben wurde	true | false
	log_statement_start	Protokollierung, wenn eine Anweisung gestartet wurde	true | false
	log_statement_finish	Protokollierung, wenn eine Anweisung beendet wurde	true | false
	log_procedure_start	Protokollierung, wenn eine Stored Procedure gestartet wurde	true | false
	log_procedure_finish	Protokollierung, wenn die Abarbeitung einer Stored Procedure beendet wurde	true | false
	log_trigger_start	Protokollierung, wenn ein Trigger gestartet wurde	true | false
	log_trigger_finish	Protokollierung, wenn die Abarbeitung eines Triggers beendet wurde	true | false
	log_context	Protokollierung, wenn eine benutzerdefinierte Kontextvariable mit RDB$SET_CONTEXT gesetzt wurde	true | false
	print_plan	Ausgabe des Ausführungsplans des Optimizers	true | false
	print_perf	Ausgabe von detaillierten Performancestatistiken	true | false
	log_blr_requests	Protokollierung von BLR Requests	true | false
	print_blr	Ausgabe des BLR-Outputs	true | false
	log_dyn_requests	Protokollierung von DYN Requests	true | false
	print_dyn	Ausgabe des DYN-Outputs	true | false
	time_threshold	*_finish*-Parameter werden nur dann protokolliert, wenn die Ausführungszeit der protokollierten Anweisung diesen Wert überschreitet	Integer (Millisekunden)
	max_sql_length	Maximale Länge des protokollierten SQLs. Rest wird im Trace-Output abgeschnitten	Integer (Zeichen)
	max_blr_length	Maximale Länge des ausgegebenen BLR-Outputs. Rest wird im Trace-Output abgeschnitten	Integer (Zeichen)
	max_dyn_length	Maximale Länge des ausgegebenen DYN Outputs. Rest wird im Trace-Output abgeschnitten	Integer (Zeichen)
	max_arg_length	Maximale Länge des ausgegebenen Parameternamens bei parametrisierten Anweisungen	Integer (Zeichen)
	max_arg_count	Maximale Anzahl der auszugebenden Parameter bei parametrisierten Anweisungen	Integer
services	enabled	Aktiv/nicht aktiv	true | false
	log_filename	Pfad der Logdatei. Nur für System-Audit relevant	String
	max_log_size	Maximale Größe der Logdatei, bevor eine neue Logdatei erzeugt wird (Logrotation). Nur für System-Audit relevant	Integer
	include_filter	Nur Services Events protokollieren, die dem regulären Ausdruck entsprechen	String (regulärer Ausdruck)
	exclude_filter	Services Events explizit nicht protokollieren, die dem regulären Ausdruck entsprechen	String (regulärer Ausdruck)
	log_services	Protokollierung von Service-Attach/Detach/Start-Ereignissen	true | false
	log_service_query	Protokollierung der Rückgabe von Serviceaufrufen	true | false

Tabelle 2: Trace-Konfigurationsparameter

localhost/3051:service_mgr -user sysdba -password masterkey -list. Beispielhaft mit folgender Ausgabe:

```
Session ID: 7
 name: User Trace 1
 user: SYSDBA
 date: 2009-12-22 17:15:13
 flags: active, admin, trace
```

Hier sieht man auch, aus welchen Daten sich eine Trace-Session zusammensetzt:

- *Session-ID*: Wird durch die Firebird Engine vergeben und beginnt nach einem Firebird-Serverneustart wieder bei 1
- *name*: Optionaler, nicht eindeutiger Name einer Session, der beim Start angegeben wurde
- *user*: Name des Benutzers, der die Trace-Session gestartet hat
- *date*: Zeitstempel, wann die Trace-Session gestartet wurde
- *flags*: Status (aktiv, unterbrochen), Typ (audit, trace)

Die verfügbaren Optionen/Schalter von *fbtracemgr* samt Beispielen können den Firebird 2.5 Release Notes entnommen werden.

Trace-Konfiguration

Die im vorangegangen Beispiel verwendete Konfigurationsdatei habe ich bewusst sehr einfach gehalten, um den Einsteig zu erleichtern. Es existiert allerdings eine Vielzahl an Trace-Konfigurationsparametern, die durch Kommentare in der mitgelieferten *fbtrace.conf* dokumentiert sind. Ein Blick in diese Datei lohnt sich allemal, da sich diese auch als Basis für eigene Trace-Konfigurationen anbietet. Grundsätzlich kann eine Trace-Konfiguration aus einem Default-Abschnitt

```
<database>
...
</database>

<services>
...
</services>
```

mit für die Trace-Konfiguration „globalen" Einstellungen und datenbankspezifischen Definitionen unter Angabe eines regulären Ausdrucks (konform dem neuen *SIMILAR TO*-Prädikat in Firebird 2.5) für den Datenbanknamen, wie zum Beispiel aus unserem obigen Beispiel

```
<database employee.fdb>
 ...
</database>
```

bestehen. Die Abschnitte <*database* ...> beinhalten Konfigurationsparameter für die Protokollierung auf Anweisungsebene (z. B. Prepare-Statement, Execute-Statement, ...), wohingegen der <*services*>-Abschnitt nur einmal in einer Trace-Konfiguration vorkommen darf und eine Möglichkeit anbietet, um serverweite Services-API-Aufrufe (Backup, Restore, Header-Page abrufen, ...) zu protokollieren.

Die Trace-Konfiguration wird von der Firebird-Engine beim Start einer Audit/Trace-Session von oben nach unten

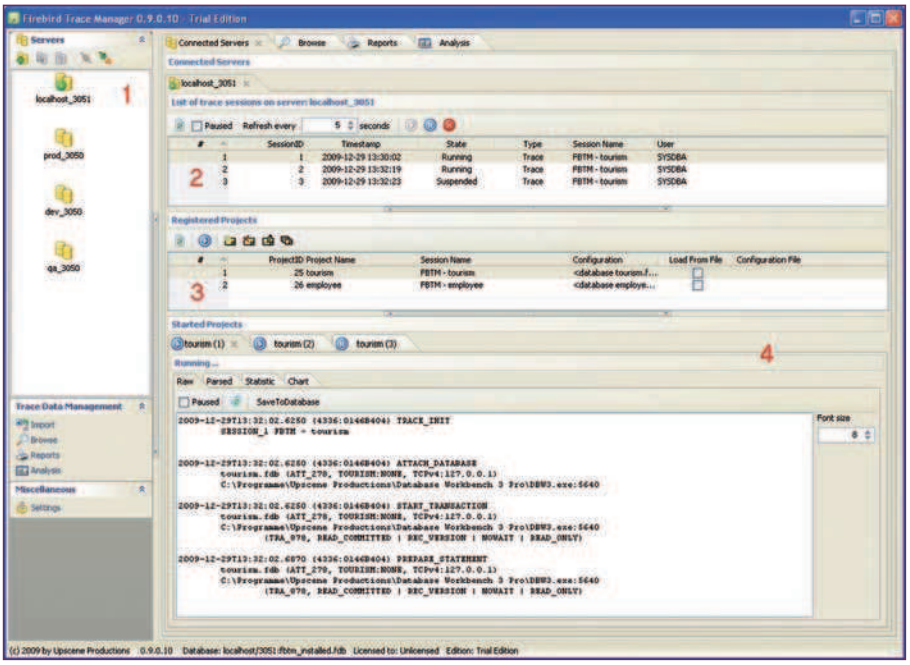


Abb. 1: Das Hauptformular des Firebird Trace Managers

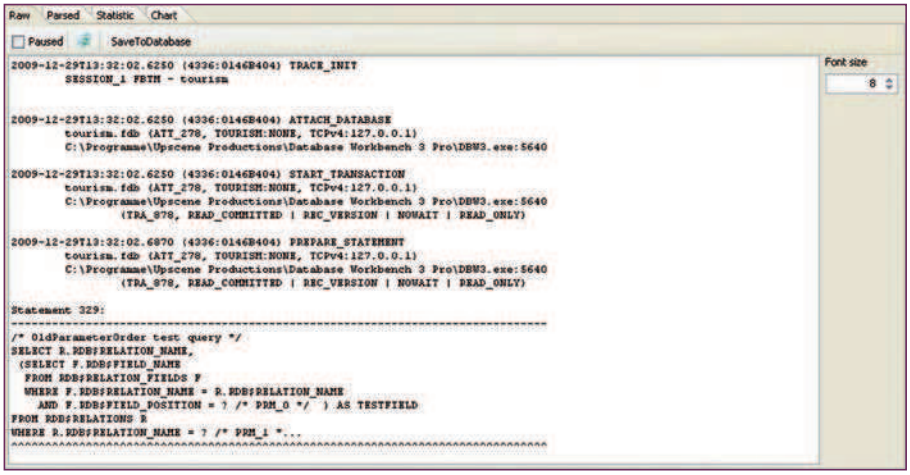


Abb. 2: Die empfangenen Daten im Raw-Format

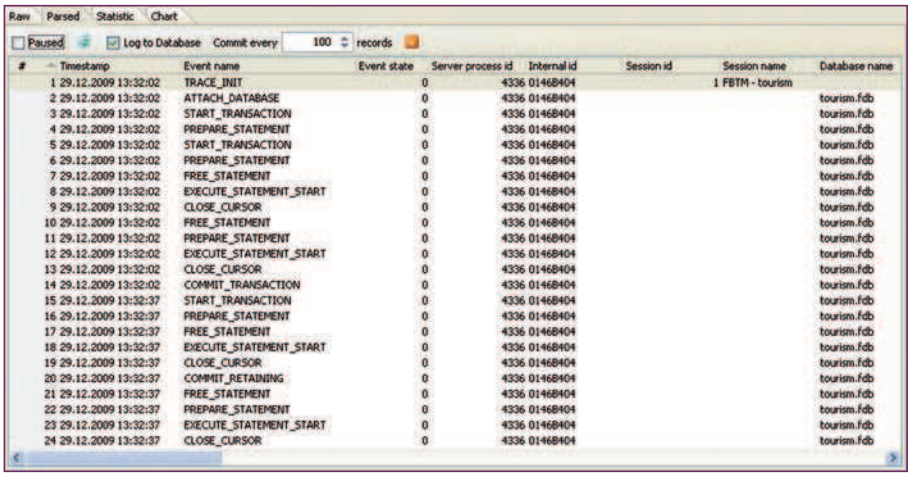


Abb. 3: Strukturierte Ansicht der Daten

abgearbeitet. Somit können globale durch datenbankspezifische Einstellungen „überschrieben" werden. Ein Beispiel:

```
<database>
 enabled true
</database>

<database employee.fdb>
 enabled false
 log_connections true
 log_transactions true
</database>

<database tourism.fdb>
 log_connections true
</database>
```

In diesem Beispiel wird für diese Trace-Konfiguration global das Audit/Trace auf Datenbankebene aktiviert, jedoch im Speziellen für *employee.fdb* deaktiviert. Für den *tourism.fdb*-Eintrag ist es nicht mehr notwendig, *enabled* auf *true* zu setzen, sofern sich an der globalen Einstellung nichts ändert. Tabelle 2 gibt einen Überblick über die möglichen Trace-Konfigurationsparameter je Abschnitt (*database*, *services*).

Security

Jeder Benutzer kann Trace-Sessions starten und verwalten. Der verwendete Benutzer bei der Anmeldung am Services-Manager entscheidet allerdings über die Rechte und die Sichtbarkeit in Bezug auf andere Trace-Sessions. Trace-Sessions, die mit dem SYSDBA-Benutzer gestartet wurden, protokollieren Aktivitäten aller Benutzer/Verbindungen (sofern der *connection_id*-Parameter in der Trace-Konfiguration nicht gesetzt wurde). SYSDBA kann auch uneingeschränkt andere Trace-Sessions anhalten, fortführen und beenden, auch System-Audit-Sessions. Startet ein Nicht-SYSDBA-Benutzer eine Trace-Session, dann bekommt diese Trace-Session nur die Ereignisse mit, die wiederum durch diesen Benutzer entstanden sind. Als Vergleichskriterium wird hier der für die Datenbankverbindung verwendete Benutzername herangezogen. Des Weiteren bekommt ein Nicht-SYSDBA-Benutzer bei der Abfrage der Liste der aktuellen Trace-Sessions nur die angezeigt, die durch diesen Benutzer ge-

startet wurden. Der Benutzer kann natürlich auch nur seine eigenen Trace-Sessions anhalten, fortführen und beenden.

Unterstützung durch Zugriffskomponenten

Die Audit- und Trace-Funktionalität kann auch in eigenen Anwendungen, vergleichbar mit z. B. Backup/Restore, über das Services-API verwendet werden. Dies kann entweder direkt über die Firebird-Clientbibliothek durch den Aufruf der entsprechenden Service-Manager-API-Aufrufe erfolgen oder, bei der Verwendung von Delphi, bieten in der Regel die Zugriffskomponenten entsprechende Services/Admin-Komponenten an, die den Umgang mit dem Services-API in der eigenen Anwendung erleichtern. Dazu müssen allerdings die von Ihnen verwendeten Komponenten diese Services-API-Erweiterungen unterstützen. Es handelt sich hierbei um die folgenden neuen Services-API-Aufrufe:

- isc_action_svc_trace_start
- isc_action_svc_trace_stop
- isc_action_svc_trace_suspend
- isc_action_svc_trace_resume
- isc_action_svc_trace_list

Bis dato sind mir nur zwei Delphi-orientierte Produkte bekannt, die diese Erweiterungen implementiert haben: IBObjects [5] in Form des bei SourceForge erhältlichen IBOAdmin Packages [6] und IBDAC [7]. Im Fall des IBOAdmin Packages allerdings noch nicht im öffentlich zugänglichen SVN-Repository, sondern in meiner lokalen Sandbox. Sollten Sie Interesse an den Erweiterungen haben, senden Sie mir einfach eine E-Mail. Die benötigten Erweiterungen im JDBC-Typ-4-Treiber „Jaybird" sind im HEAD Branch eingecheckt und müssen noch reviewed werden. Voraussichtlich wird Jaybird diese Änderungen im Release 2.2 beinhalten. Andere Produkte wie FIBPlus [8] werden vermutlich zeitnah nachziehen bzw. bieten zum Zeitpunkt der Veröffentlichung dieses Artikels bereits eine aktualisierte Version an.

Firebird Trace Manager

Das in der Firebird-Distribution enthaltene Kommandozeilentool *fbtracemgr* bietet einen nur sehr rudimentären Funktionsumfang für die Verwendung der Audit- und Trace-Services an. Abhilfe wird hier ein neues kommerzielles Produkt *Firebird Trace Manager* der Firma Upscene Productions [9] schaffen. Auf der Webseite des Herstellers gibt es zum Zeitpunkt des Verfassens des Artikels zu diesem Produkt noch keine näheren Informationen, allerdings steht eine öffentliche Vorabtestversion unter [10] zur Verfügung. Da dieses Produkt meiner Feder entstammt, erlauben Sie mir, hier etwas näher darauf einzugehen.

Abbildung 1 zeigt das Hauptformular. Im linken Bereich 1 sieht man die registrierten Server, zu denen man sich über den Firebird Services Manager verbinden kann. Im rechten Bereich sieht man im Reiter Connected Servers die aktuellen Serververbindungen. Innerhalb einer Serververbindung wird in Bereich 2 eine Liste der Trace-Sessions inklusive deren Status angezeigt. In diesem Bereich können Trace-Sessions sehr einfach temporär angehalten, wieder fortgeführt und beendet werden. Je registriertem Server kann eine beliebige Anzahl an *Projekten* erstellt werden (Bereich 3). Ein Projekt definiert im Wesentlichen die Trace-Konfiguration sowie den optionalen Namen der Trace-Session. Basierend auf einem Projekt kann nun eine Trace-Session gestartet werden. Die im Firebird Trace Manager gestarteten Trace-Sessions stehen in Bereich 4 zur Verfügung. Aufgrund der Multi-Threading-Fähigkeit des Firebird Trace Managers können mehrere Trace-Sessions verwaltet, Trace-Daten „live" empfangen und weiterverarbeitet werden. Die Trace-Daten werden im Raw-Format, wie das auch bei *fbtracemgr* der Fall ist, vom Server empfangen (Abb. 2). In einem ersten Schritt der Weiterverarbeitung wird der Raw-Output durch einen kontextbasierten Parser in eine strukturierte Form gebracht (Abb. 3) und optional in einer Firebird-Datenbank für weitere Analysezwecke gespeichert. Die weitere Verarbeitung der geparsten Daten besteht darin, diese in einer Statistik (Abb. 4) und in einem Häufigkeitsdiagramm (Abb. 5) je Ereignis zu visualisieren.

Mit der strukturierten Speicherung der Trace-Daten in der mitgelieferten Firebird-Datenbank steht nun die Tür für die weitere Verarbeitung offen. Ein *Browse-*

IMPRESSUM

Verlag:
Software & Support Verlag GmbH

Anschrift der Redaktion:
Entwickler Magazin
Software & Support Verlag GmbH
Geleitsstraße 14
D-60599 Frankfurt am Main
Tel. +49 (0)69 630089 0
Fax. +49 (0)69 630089 89
redaktion@entwickler-magazin.de
entwickler-magazin.de

Chefredakteur: Masoud Kamali
Advisory Board: Holger Flick, Rudolf Jansen, Olaf Monien
Verantwortlicher Redakteur: Robert Lippert
Redaktion: Nicole Bechtel, Thomas Wießeckel

Autoren dieser Ausgabe: René Bröcker, Bernd Fondermann, Mario Fraiß, Dmytro Gerasychuk, Markus Hasenbein, Wolfgang Henseler, Andreas Holubek, Michael Ihringer, Jörg Jackisch, Rudolf Jansen, Thomas Kalippke, Daniel Koch, Veikko Krypczyk, Claudia Matthias, Bernd Ott, Christoph Schneider, Thomas Steinmaurer, Marc André Zhou

Chefin vom Dienst/Leitung Schlussredaktion: Nicole Bechtel
Schlussredaktion: Ella Klassen, Katharina Klassen, Frauke Pesch
Leiter Grafik/Produktion: Jens Mainz
Layout, Titel: Daniela Albert, Kristin Brockmann, Pobporn Fischer, Karolina Gaspar, Melanie Hahn, Katharina Ochsenhirt, Maria Rudi, Patricia Schwesinger
CD/DVD-Erstellung: Daniel Zuzek, Özkan Peksan

Anzeigenverkauf:

Entwickler Magazin
Patrik Baumann
Software & Support Verlag GmbH
Tel. +49 (0)69 630089 20
pbaumann@entwickler-magazin.de

Open Source
Verlagsbüro Ohm-Schmidt
Osmund Schmidt
Schneckenburger Str. 22
30177 Hannover
Tel. +49 (0)511 2354164
Fax: +49 (0)1805 06033695669
E-Mail: osmund@ohm-schmidt.de

Es gilt die Anzeigenpreisliste Nr. 15

Pressevertrieb:
DPV Network GmbH
Tel.:+49(0)40 23711 0
www.dpv-network.de

Druck: PVA, Landau

Abo-Service:
Software & Support Verlag GmbH
Tel.: +49 (0)69 630089 0
Fax.: +49 (0)69 630089 89
entwickler-magazin.de/service

Abonnementpreise der Zeitschrift (inkl. Leser-CD):

Inland:	6 Ausgaben	€ 29,50
Studentenpreis:	6 Ausgaben	€ 23,60
europ. Ausland:	6 Ausgaben	€ 39,50
Stud. europ. Ausland:	6 Ausgaben	€ 33,60

Abonnementpreise der Zeitschrift (inkl. Leser-CD plus Profi-CD):

Inland:	6 Ausgaben & CD	€ 99,-
Studentenpreis:	6 Ausgaben & CD	€ 80,-
europ. Ausland:	6 Ausgaben & CD	€ 109,-
Stud. europ. Ausland:	6 Ausgaben & CD	€ 90,-

Abonnementpreise der Profi-CD:

Inland:	6 CD-ROM	€ 72,-
Studentenpreis:	6 CD-ROM	€ 62,-
europ. Ausland:	6 CD-ROM	€ 82,-
Stud. europ. Ausland:	6 CD-ROM	€ 72,-

ISSN: 1619-7941

Erscheinungsweise: zweimonatlich

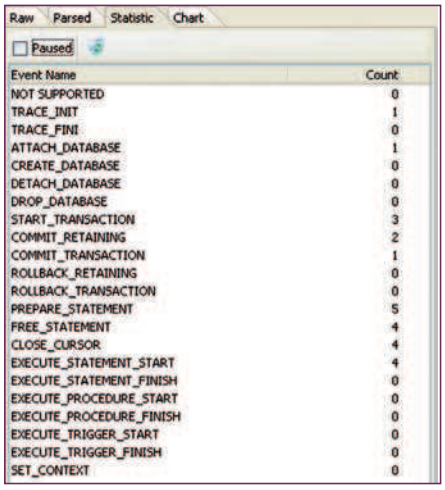


Abb. 4: Statistik aus den geparsten Daten

Modul ermöglicht den einfachen Zugriff auf die gespeicherten Trace-Daten. Ein Modul *Reporting* erlaubt, vordefinierte und benutzerdefinierte Berichte abzurufen bzw. zu erstellen. Im *Analysis*-Modul können OLAP/Pivot-ähnliche Fragestellungen sehr einfach beantwortet werden. So zum Beispiel: „Anzahl der Ereignisse nach Jahr/Quartal/Monat/Tag“ (Abb. 6). Die Analysemöglichkeiten sind durch die Ad-hoc-Fähigkeit immens. Der Benutzer kann aus einer Vielzahl an Attributen und Aggregatsfunktionen auswählen, definierte Abfragen abspeichern und zu einem späteren Zeitpunkt wieder laden.

Firebird Trace Manager wird voraussichtlich in unterschiedlichen Editionen verfügbar sein: *Lite*, *Standard* und *Enterprise*. Die Lite-Edition wird kostenlos sein und im Wesentlichen den Empfang und den Zugriff auf protokollierte Raw-Daten erlauben. Die Standard-Edition inkludiert zusätzlich zur Lite-Edition den kontextbasierten Parser, die Speicherung und den Zugriff auf die strukturierten Trace-Daten sowie das Reporting-Modul. Die Enterprise-Edition wird zusätzlich zur Standard-Edition noch das Analysemodul und ein noch in Planung befindliches *Alerting*-Modul zur automatischen Benachrichtigung beim Auftreten von bestimmten Ereignissen beinhalten. Letzteres wird allerdings in der initialen Version 1.0 noch nicht inkludiert sein. Bis zum finalen Release könnte sich das Feature-Set je Edition noch geringfügig ändern.

Fazit

Das Audit- und Trace-Services-API in Firebird 2.5 ist neben der verbesserten SMP-Unterstützung wohl die zweitwichtigste Neuerung in diesem Release. Im Zusammenspiel mit den aus 2.1 verfügbaren Monitoring-Tabellen und einem Trigger-basierten Audit-Mechanismus zur Protokollierung von datenverändernden Operationen hat man in 2.5 nun eine gute Abdeckung in Bezug auf das Monitoring von Firebird-Datenbanken erreicht.

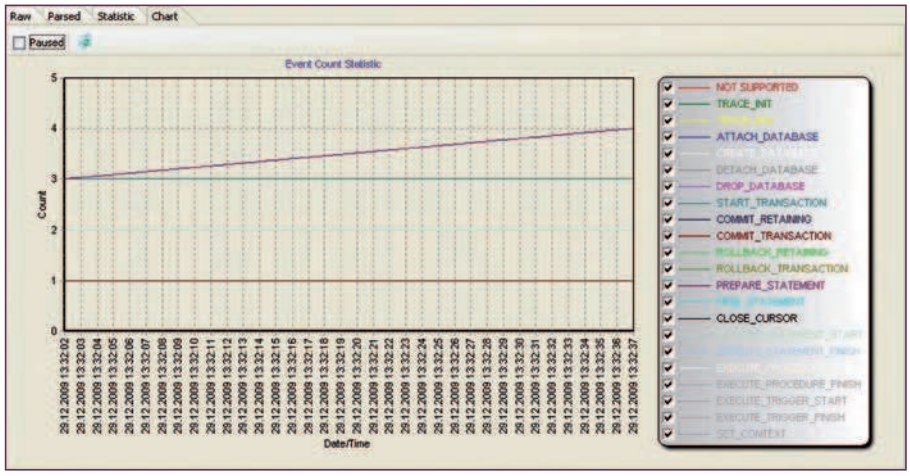


Abb. 5: Häufigkeitsdiagramm aus den geparsten Daten

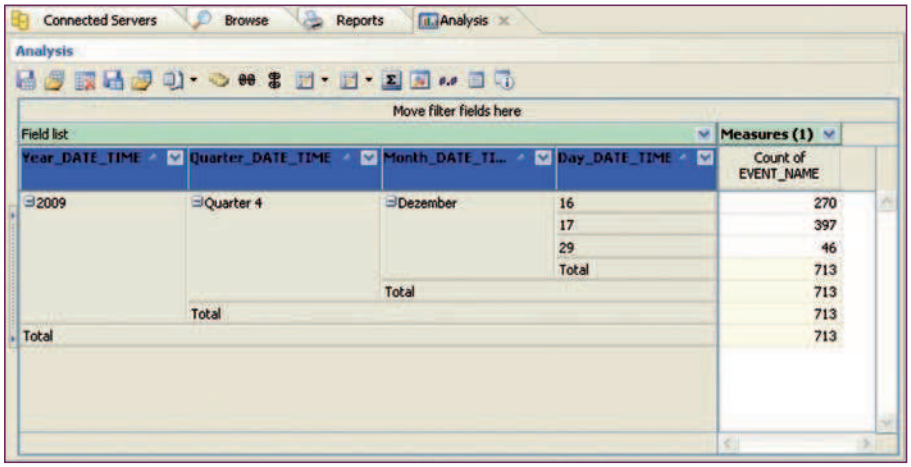


Abb. 6: Beispiel des Analysis-Moduls

Thomas Steinmaurer *ist wissenschaftlicher Mitarbeiter am Software Competence Center Hagenberg (SCCH) [11] mit Schwerpunkt Datenmanagement und Data Warehousing im industriellen Umfeld. Des Weiteren ist er für die LogManager-Produktreihe und Firebird Trace Manager bei Upscene Productions [9] verantwortlich.*

Links & Literatur

[1] Entwickler Magazin 1.2010 – Firebird 2.5

[2] Blogeintrag „Versioning monitoring tables“: http://blog.upscene.com/thomas/index.php?entry=entry080730-135908

[3] IB LogManager: http://www.upscene.com/products.audit.iblm_main.php

[4] Der Entwickler 3.2004: Serverseitiges Auditing – Protokollierungsmechanismus für InterBase 7.1 selbst gemacht

[5] IBObjects: http://www.ibobjects.com

[6] IBOAdmin Package: https://ibobjects.svn.sourceforge.net/svnroot/ibobjects/trunk/iboadmin

[7] IBDAC: http://www.devart.com/ibdac/

[8] FIBPlus: http://www.devrace.com/en/fibplus/

[9] Upscene Productions: http://www.upscene.com

[10] Vorabtestversion des Firebird Trace Managers: http://blog.upscene.com/thomas/index.php?entry=entry091214-081214

[11] Software Competence Center Hagenberg: http://www.scch.at