Optimizing performance

This section concentrates upon the performance optimization of your Firebird server. With any system there is always a limiting factor. If you remove that limiting factor, something else then in turn becomes the limiting factor. It is therefore vital to be aware of all these factors which contribute to your overall database server performance.

Operating systems

Certainly the most popular operating system today is Microsoft, although Linux is constantly improving its strong foothold in the market. With regard to Windows it is fairly irrelevant which version you use. Windows 2000 does have the advantage however, that it does not carry as much overhead as Windows XP and co. Physically it can be roughly estimated, that a Firebird server installation on Windows working with VMware, the performance is approximately 30% less than native processor use. VMware offers a number of advantages, for example that you can back up the complete VMware, complete with database, configuration etc., enabling the database to be restarted immediately with the same IP address. And VMware files are pretty well impossible to corrupt.

Performance variations are minimal when using the same hardware and the same Firebird version. Slight discrepancies in different areas may be detected, these having different advantages and disadvantages, which need to be assessed individually for individual application requirements.

The real advantage with Linux is quite simply the stability of the total system. With Windows it is possible to achieve a high level of stability, there are a number of parameters and settings that need to be accordingly configured. Linux is certainly better with regard to memory configuration, and the larger the application, the more advantages you will discover with Linux. And if you wish to run a web server alongside your Firebird server on the same machine, you should definitely consider Linux.

If however you have a classic medium-sized system with 10-20 users, you will not detect any significant differences in overall performance.

Optimal hard disk use

The optimal hard disk configuration for an efficient Firebird server is to have separate dedicated hard disks for the operating system, database and temp files. Partitions are of no advantage here, as the read/write head still has to scan the whole drive. The decisive factor with fixed disks is the read/write speed; and a large cache can also improve performance.

Raid systems are useful for large databases, and the larger the disk cache the better.

Small databases up to 2 GB can fit in the cache RAM – that can be the database cache RAM or just the Windows cache RAM.

Optimizing hardware configuration

Take into consideration the following factors when optimizing your hardware:

- Multicore CPU are useful for the Firebird Classic server, at least two cores are advisable for the Superserver - for the server itself, and another for events.
- Large cache server CPUs (Xeon, Opteron) are useful for all architectures - particularly with large databases with a high number of users.
- Server main boards are optimized for I/O speed.
- High speed RAM DDR3/DDR2.

back to top of page

Optimizing OS configuration

Firstly, remove all unnecessary tasks and services from the database server. Scrutinize anything listed in the Task Manager, when you are unsure why it's there, stop it running, and if possible deinstall the application that started it in the first place. A Windows system can run with a minimum number of processes on dedicated database server.

High performance database servers should not be used for anything else, be it file servers, mail servers (every time they do a POP grab, you're bound to register a discernible drop in database performance), or print servers and the like. No antivirus software is at all necessary, no backup/restore software that handles open file backup, especially not for the database files but also for the temp files. Even when invoking a shadow, by backing up your database files, serious degradation can be noticed in the overall server performance, particularly if you have intensive user traffic at the time.

And please do not run a 3D OpenGL screen saver; fancy screen savers also contribute to performance degradation! And if you're using Linux, run the server without the GUI to save even more memory that can be better used by your database server.

back to top of page

Firebird benchmarks tests

The IBExpertDemoDB can be used for simple server benchmark tests. By running the db1.sql it is possible to quickly determine discrepancies in performance on different hardware and OS configurations. Please refer to IBExpert Benchmarks for details of benchmarking possibilities using IBExpert tools.

Important: when benchmark testing, take into consideration the potential database size and number of users in a year's time. Testing performance on double your current database size with double the number of users will offer you the comfort factor in the near future!

Optimizing the database

1. Split complex tables into several smaller ones (Database normalization).

- For reasons of compatibility with legacy databases, it might help to add an updatable view with the name of the old table and with the same structure.

- Old source code can still use the old name for SELECT, INSERT, UPDATE or DELETE; new source code can work directly on the new smaller tables.
    - This can provide a real improvement in speed, especially in the case of very complex tables. Typically it also improves the restore speed considerably.

2. Do not use GUID for primary key fields, as these use much more space and will be slower as an INTEGER or BIGINT.

3. Do not use very long CHAR/VARCHAR fields unless they are really necessary.

4. Seldom-used columns should be stored in different tables.

5. Use indices only where necessary.

6. Compound indices should only be used on large tables.

7. If you are upgrading from an older Firebird version to the new 2.1 version, it is also important that you upgrade all your clients accordingly. The Firebird 2.1 client can communicate much more effectively with the Firebird 2.1 server, which can mean performance improvements of up to 40%!

Parameters for optimal performance

1. Database model - if your database model is weak no amount of tweaking other parameters will make any significant difference. Read the Database design and database normalization article and use IBExpert's Database Designer to optimize your database model.

2. Test SQL statements (refer to Optimizing SQL statements for further information).

3. Analyze index plans - tons of information, examples and tips can be found here: Index statistics, Index, Performance Analysis.

4. Transaction control - monitor, analyze and improve.

5. Server-side programming - let the server do the work, rather than transferring masses of data pages to the client and performing your queries there.

6. Optimizing cache - refer to Temporary files, Memory configuration and Optimizing hardware configuration for further information.

7. Hardware

8. Operating System

9. Network

back to top of page

The Firebird Optimizer and index statistics

All statistics are recalculated only when a database is restored after backing up, or when this is explicitly requested by the developer. When an index is initially created, its statistical value is 0.

Imagine the following situation: you have a database of all the inhabitants of Great Britain. You require a list of all men living in Little Bigton. How should the server process the query? The population of Great Britain is currently around 60 million. Approximately half are men. Should the server first select all men (around 30 million) and then take these results and select all those who live in Little Bigton, or should it first select all residents of Little Bigton (which let's say has a population of around 5,000) and then select all men?

The best selectivity is of course to first select all residents of Little Bigton, and then discern the number of males. The problem is that when you send the query to the server, it needs further information to help it decide how to go about executing the query. For this it uses indices, and to decide which index is the best to use first, it relies on the index selectivity.

Therefore it is extremely important, particularly with new databases where the first data sets are being entered, to regularly explicitly recompute the selectivity, so that the optimizer can recognize the most efficient indices. This is not so important with databases where little data manipulation occurs, as the selectivity will change very little.

Refer to the article below, Automating index selectivity for details on how to automate the recalculation of index selectivity in applications, and to the following articles for further information regarding indices and index statistics generally:

Index Recompute selectivity of all indices SQL Editor / Plan Analyzer SQL Editor / Performance Analysis Using the PLAN operator IBExpert Table Editor / Indices Enhancements to indexing in Firebird 2.0 Firebird for the database expert: Episode 1 - Indexes Recreating Indices 1 Recreating Indices 2

back to top of page

Automating the recalculation of index statistics

A common problem is that when an application is delivered to a customer, an “empty” database is supplied, i.e. it contains only the metadata and no customer data. As different customers enter different amounts of data, with time some may complain that their application is too slow in certain areas. This is most often due to the indices' statistics not having been calculated up to date (or not having been calculated at all!), which means that the Optimizer cannot use the indices efficiently to process queries.

If you want to have your software working at its most efficient, always use up-to-date statistic values to maximize performance (if one customer has many orders for few products all serviced by two employees and another few orders for many products, serviced by 100 employees, the index statistics and hence selectivity, will obviously develop differently). Without updating the index statistics regularly as more and more data is added you will incur performance problems (eg. all males living in Little Bigton). The command for this is:

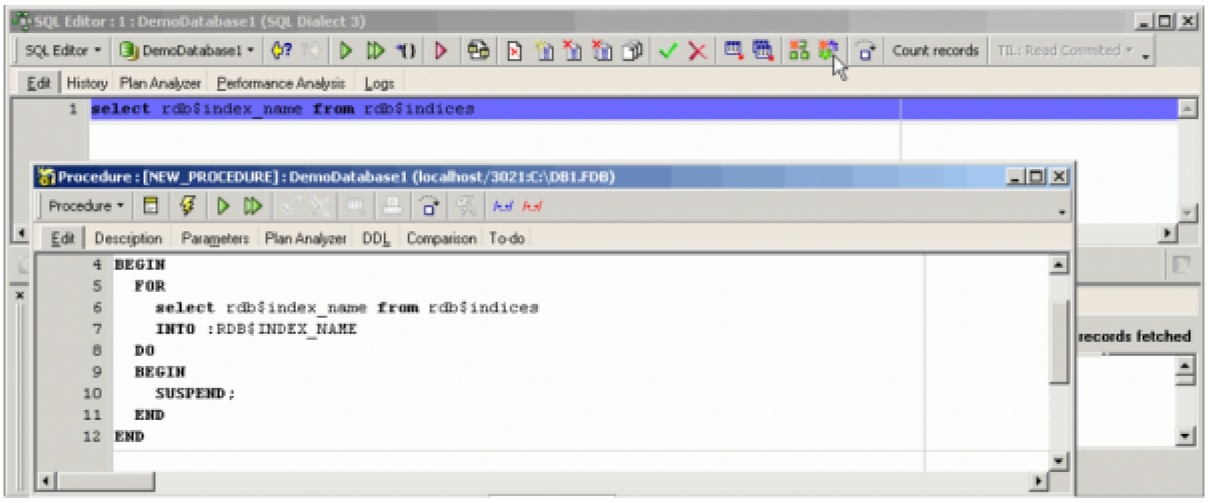
```
set statistics index
```

The index names can be found in a system table called RDB$INDICES. This table also displays the index value of each index in the RDB$STATISTICS column.

Use:

```
select rdb$index_name from rdb$indices
```

to obtain list of all index names. A procedure can then be created directly from this (refer to Create view or procedure from SELECT from for further information), selecting into Local variables.



```
select rdb$index_name from rdb$indices
```

```
BEGIN
  FOR
    select rdb$index_name from rdb$indices
    INTO :RDB$INDEX_NAME
  DO
  BEGIN
    SUSPEND;
  END
END
```

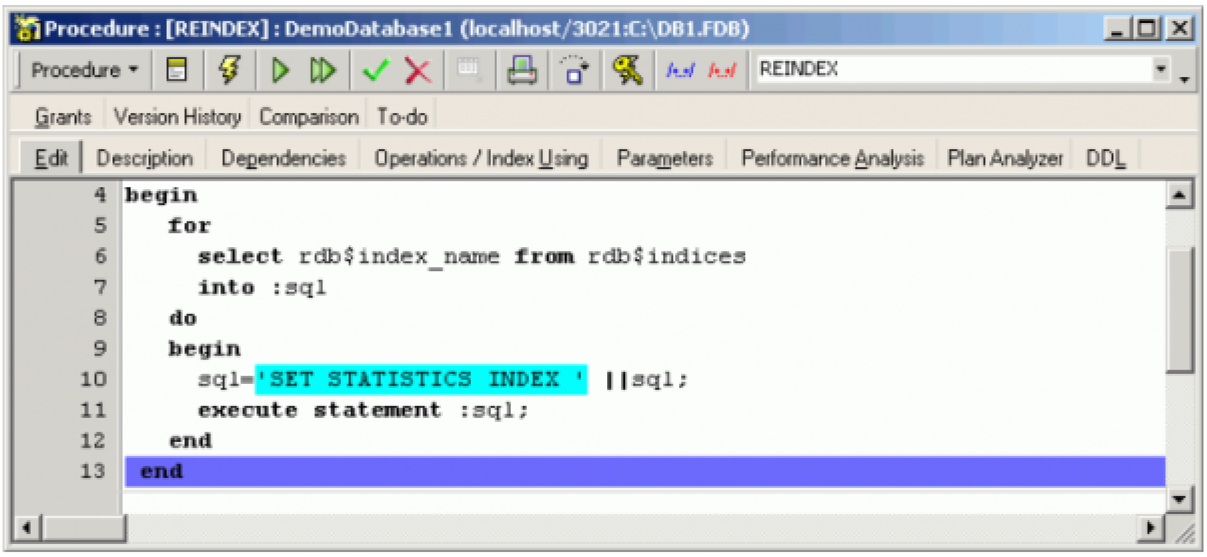
(This and the following illustration show the Procedure Editor with deactivated Lazy Mode.)

Simply rename the procedure to REINDEX, alter the variable to declare variable sql varchar(300); and also into :sql.

After the index name has been put into the variable, it should say:

```
begin
  sql='SET STATISTICS INDEX ' ||sql;
  execute statement :sql;
```

Here the SET STATISTICS INDEX statement has been combined with the sql variable. And inside a Firebird stored procedure it is possible to use this SQL statement, which is inside a variable, and execute it directly from the procedure.



```
begin
   for
     select rdb$index_name from rdb$indices
     into :sql
   do
   begin
     sql='SET STATISTICS INDEX ' ||sql;
     execute statement :sql;
   end
 end
```

To run simply type:

```
execute procedure reindex
```

You do not even need to shut down the database to recompute the selectivity of indices.

Do this regularly and the Optimizer will be able to use indices efficiently.

back to top of page

Using the IBExpert Database Statistics

The IBExpert Services menu item, Database Statistics, reveals a wealth of information about your database.

When approaching the Database Statistics analysis, it is important to know what information is available, which information is important and how to interpret and use it to solve performance problems. Please refer first to the IBExpert documentation chapter, Database Statistics, for a detailed explanation of the various statistics available and their significance.

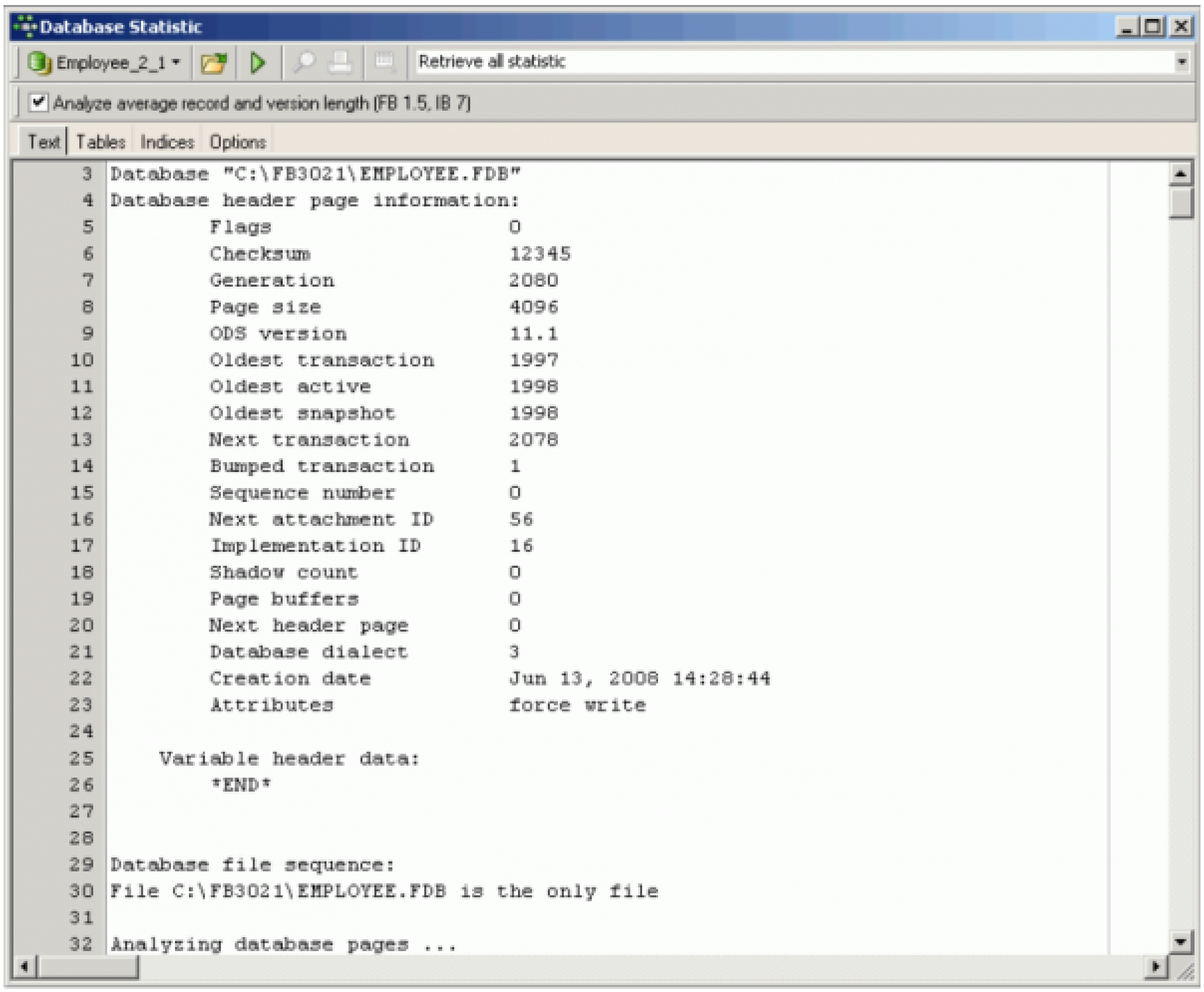
A common performance problem is that the database gradually becomes slower and slower. This is usually due to an open transaction somewhere in the database. Look at the number of record versions (total record versions). These exist because Firebird still needs to store the old data for old open transactions. This is handled internally by a transaction number.

In a production database with multiple users you will often see record versions, but if there are no old open transactions the database will delete these older record versions automatically when they are no longer needed, i.e. following a commit or rollback. The garbage collector cannot work if there are open transactions anywhere.

The oldest and newest transaction numbers can be found in the summary at the top of the log found in the Text page. The larger the difference between the *Oldest active transaction* (OAT) and the *Next transaction*, the bigger performance problems you will encounter. The Firebird server does not just administrate record versions for the database object which still has an open transaction, but for the entire database. In repeatable read mode a snapshot is made of the whole database, as soon as a transaction is started. When the transaction is completed (i.e. committed or rolled back) the garbage collector will then delete all old record versions that are no longer needed.

The log file and the Tables page show the statistics for all tables: here you can ascertain which tables have large amount of record versions being held by the server. The max number of versions means there is one record that has this amount of different versions. This indicates that there is still one active transaction in the database so that the old record versions cannot be deleted.

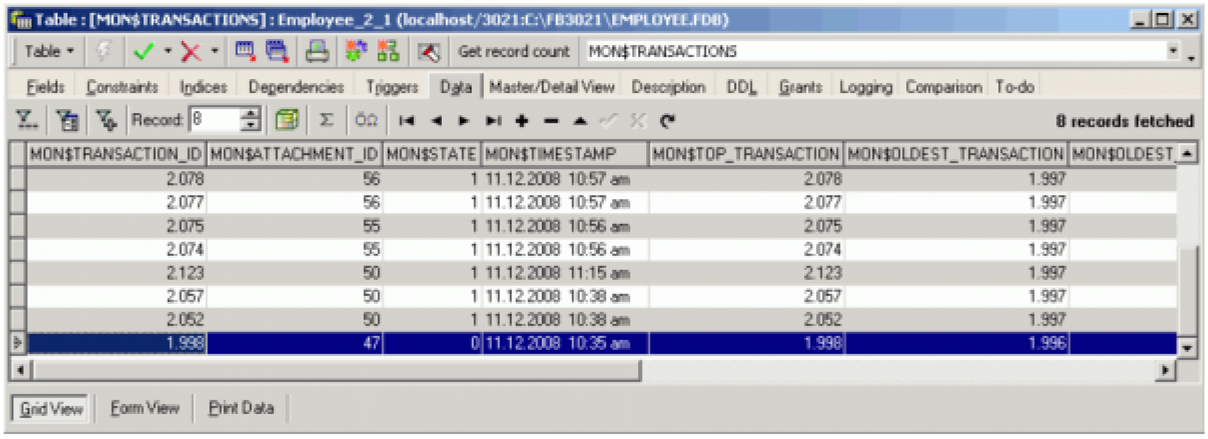
To find out what or who is causing such a problem, look at the server while the database is in use.



```
Database "C:\FB3021\EMPLOYEE.FDB"
Database header page information:
        Flags                       0
        Checksum                    12345
        Generation                  2080
        Page size                   4096
        ODS version                 11.1
        Oldest transaction          1997
        Oldest active               1998
        Oldest snapshot             1998
        Next transaction            2078
        Bumped transaction          1
        Sequence number             0
        Next attachment ID          56
        Implementation ID           16
        Shadow count                0
        Page buffers                0
        Next header page            0
        Database dialect            3
        Creation date               Jun 13, 2008 14:28:44
        Attributes                  force write

    Variable header data:
        *END*


Database file sequence:
File C:\FB3021\EMPLOYEE.FDB is the only file

Analyzing database pages ...
```

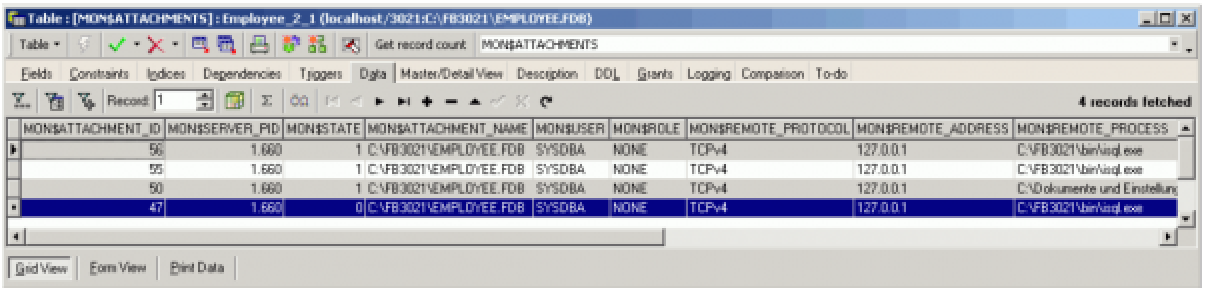
The above summary shows us that the next transaction is number 2078, and the oldest active transactions number is 1998.

If system tables are activated in the IBExpert DB Explorer (check the options using the IBExpert Database menu item, Database Registration Info / DB Explorer page), you can view and open the Firebird 2.1 MON$TRANSACTIONS table. On the Data page there is an entry in this example for transaction 1998:



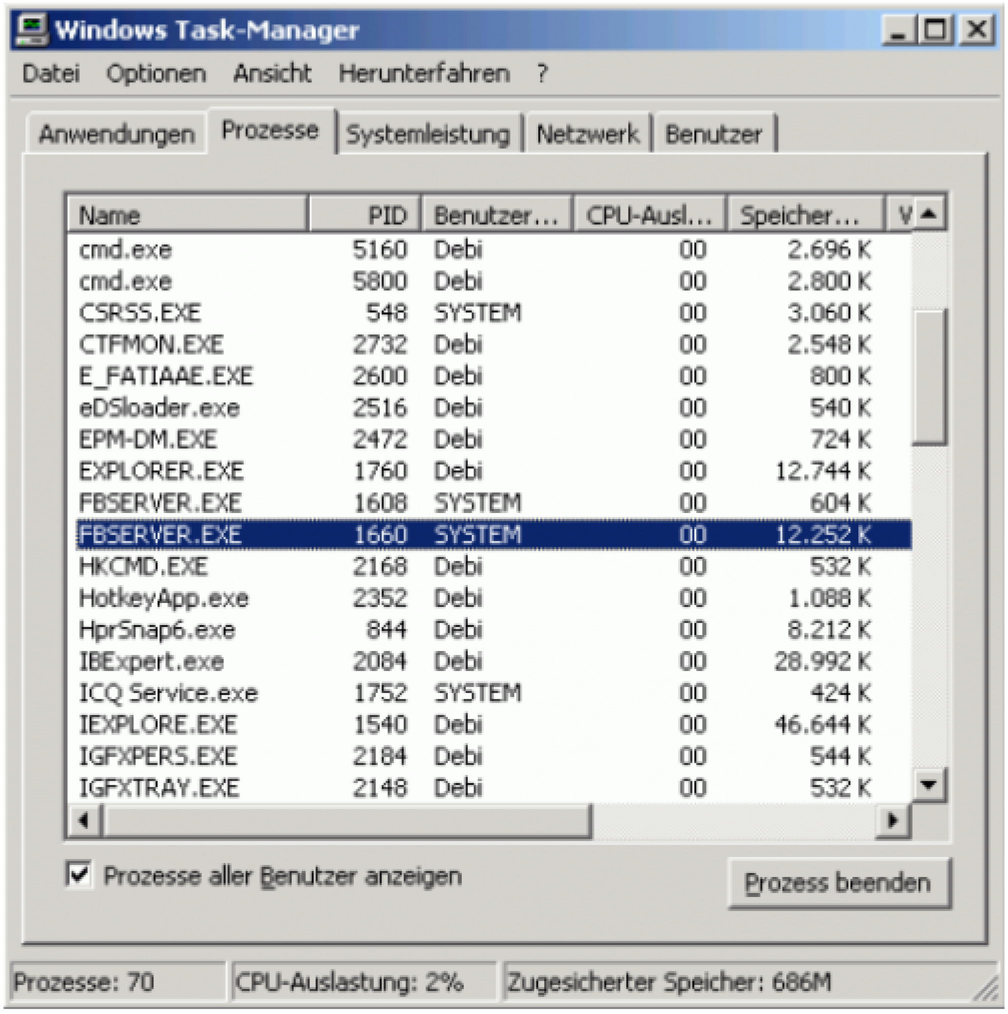
MON$TRANSACTION_ID	MON$ATTACHMENT_ID	MON$STATE	MON$TIMESTAMP	MON$TOP_TRANSACTION	MON$OLDEST_TRANSACTION	MON$OLDEST_
2.078	56	1	11.12.2008 10:57 am	2.078	1.997	
2.077	56	1	11.12.2008 10:57 am	2.077	1.997	
2.075	55	1	11.12.2008 10:56 am	2.075	1.997	
2.074	55	1	11.12.2008 10:56 am	2.074	1.997	
2.123	50	1	11.12.2008 11:15 am	2.123	1.997	
2.057	50	1	11.12.2008 10:38 am	2.057	1.997	
2.052	50	1	11.12.2008 10:38 am	2.052	1.997	
1.998	47	0	11.12.2008 10:35 am	1.998	1.996	

This transaction has an attachment ID number 47. It was started at 10:35 and has been active for over 20 minutes. A typical transaction would not be active for that length of time. More information concerning this attachment ID 47 can be found in the MON$ATTACHMENTS table:



MON$ATTACHMENT_ID	MON$SERVER_PID	MON$STATE	MON$ATTACHMENT_NAME	MON$USER	MON$ROLE	MON$REMOTE_PROTOCOL	MON$REMOTE_ADDRESS	MON$REMOTE_PROCESS
56	1.660	1	C:\FB3021\EMPLOYEE.FDB	SYSDBA	NONE	TCPv4	127.0.0.1	C:\FB3021\bin\isql.exe
55	1.660	1	C:\FB3021\EMPLOYEE.FDB	SYSDBA	NONE	TCPv4	127.0.0.1	C:\FB3021\bin\isql.exe
50	1.660	1	C:\FB3021\EMPLOYEE.FDB	SYSDBA	NONE	TCPv4	127.0.0.1	C:\Dokumente und Einstellung
47	1.660	0	C:\FB3021\EMPLOYEE.FDB	SYSDBA	NONE	TCPv4	127.0.0.1	C:\FB3021\bin\isql.exe

Here the MON$SERVER_PID is displayed. If you go to the Windows Task Manager's Processes page, you will see the process ID numbers (you may first need to select the column for display using the *View* menu item, *Select columns* ..., and check the PID (Process ID) column). You can then trace the number of the Firebird instance that is used by the server. Furthermore this table also displays the user and role name, the remote address and, if you use the new Firebird clients, you will also see the remote PID.



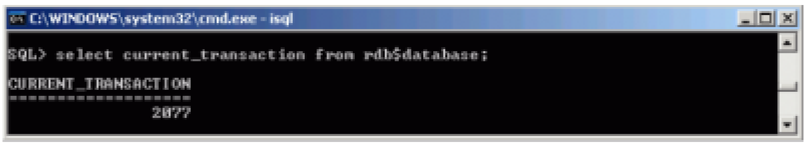
Name	PID	Benutzer...	CPU-Ausl...	Speicher...
cmd.exe	5160	Debi	00	2.696 K
cmd.exe	5800	Debi	00	2.800 K
CSRSS.EXE	548	SYSTEM	00	3.060 K
CTFMON.EXE	2732	Debi	00	2.548 K
E_FATIAAE.EXE	2600	Debi	00	800 K
eDSloader.exe	2516	Debi	00	540 K
EPM-DM.EXE	2472	Debi	00	724 K
EXPLORER.EXE	1760	Debi	00	12.744 K
FBSERVER.EXE	1608	SYSTEM	00	604 K
FBSERVER.EXE	1660	SYSTEM	00	12.252 K
HKCMD.EXE	2168	Debi	00	532 K
HotkeyApp.exe	2352	Debi	00	1.088 K
HprSnap6.exe	844	Debi	00	8.212 K
IBExpert.exe	2084	Debi	00	28.992 K
ICQ Service.exe	1752	SYSTEM	00	424 K
IEXPLORE.EXE	1540	Debi	00	46.644 K
IGFXPERS.EXE	2184	Debi	00	544 K
IGFXTRAY.EXE	2148	Debi	00	532 K

Prozesse: 70 | CPU-Auslastung: 2% | Zugesicherter Speicher: 686M

In the above example the Windows Task Manager shows me that the PID 1660 has started this transaction.

Now you only need to find out who/what is using the Firebird server with the transaction number 1998. Connect via isql or using IBExpert's SQL Editor to find out your own current transaction number using:

```
select current_transaction from rdb$database;
```


C:\WINDOWS\system32\cmd.exe - isql

```
SQL> select current_transaction from rdb$database;
CURRENT_TRANSACTION
=====================
                 2077
```

Once the initiator of the oldest transaction is found it can be committed or rolled back.

If we now go back to the MON$TRANSACTIONS table the oldest record is no longer 1998:

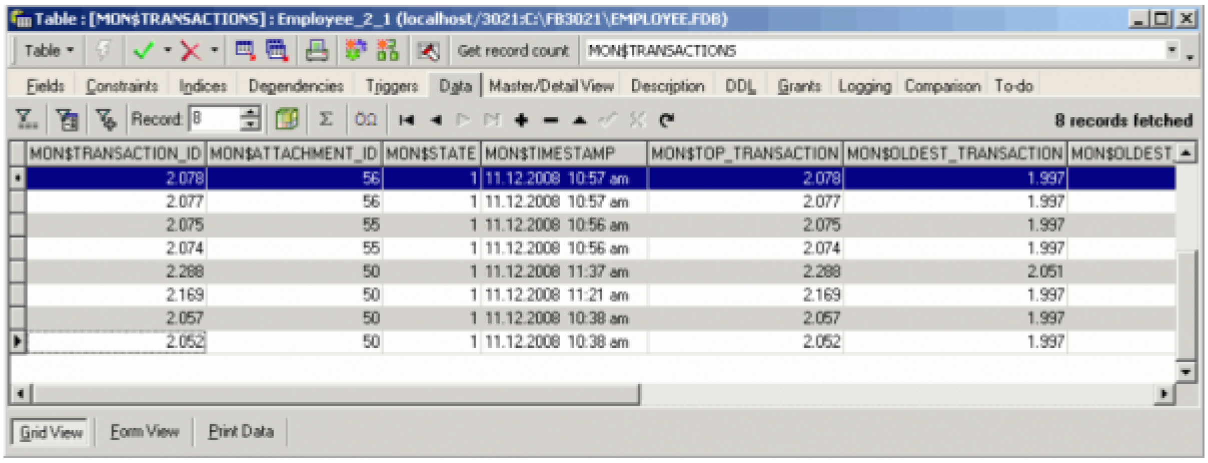
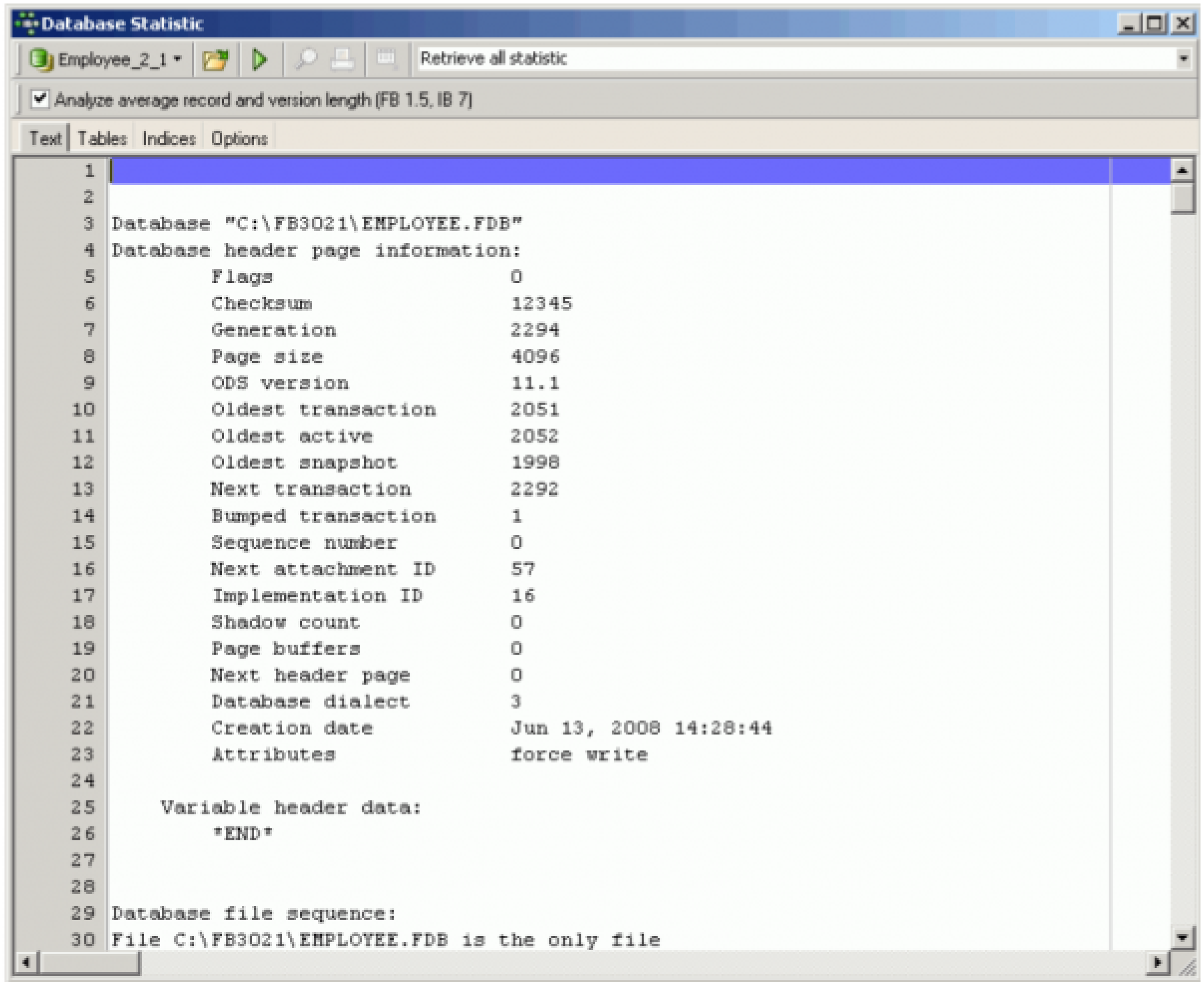


Table : [MON$TRANSACTIONS] : Employee_2_1 (localhost/3021:C:\FB3021\EMPLOYEE.FDB)

8 records fetched

MON$TRANSACTION_ID	MON$ATTACHMENT_ID	MON$STATE	MON$TIMESTAMP	MON$TOP_TRANSACTION	MON$OLDEST_TRANSACTION	MON$OLDEST_
2.078	56	1	11.12.2008 10:57 am	2.078	1.997	
2.077	56	1	11.12.2008 10:57 am	2.077	1.997	
2.075	55	1	11.12.2008 10:56 am	2.075	1.997	
2.074	55	1	11.12.2008 10:56 am	2.074	1.997	
2.288	50	1	11.12.2008 11:37 am	2.288	2.051	
2.169	50	1	11.12.2008 11:21 am	2.169	1.997	
2.057	50	1	11.12.2008 10:38 am	2.057	1.997	
2.052	50	1	11.12.2008 10:38 am	2.052	1.997	

and if we go back to the Database Statistics and run it again, we see the *Oldest active transaction* is now 2052:


Database Statistic

Employee_2_1 — Retrieve all statistic

☑ Analyze average record and version length (FB 1.5, IB 7)

```


Database "C:\FB3021\EMPLOYEE.FDB"
Database header page information:
        Flags                   0
        Checksum                12345
        Generation              2294
        Page size               4096
        ODS version             11.1
        Oldest transaction      2051
        Oldest active           2052
        Oldest snapshot         1998
        Next transaction        2292
        Bumped transaction      1
        Sequence number         0
        Next attachment ID      57
        Implementation ID       16
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        3
        Creation date           Jun 13, 2008 14:28:44
        Attributes              force write

    Variable header data:
        *END*


Database file sequence:
File C:\FB3021\EMPLOYEE.FDB is the only file
```

The *Oldest snapshot* transaction number 1998 shows where the garbage collector will start its work.

The IBExpert Database Statistics are a vital tool for solving performance problems and discerning areas for fine-tuning. They are also useful, for example, for determining the largest table, are there any empty tables, average record length (could you increase performance by splitting, for example, a large table into several smaller ones?), analyzing indices (comparing their actual selectivity with the real selectivity - do you need to recompute the selectivity of all indices?, which indices are unused or useless, analyze their depth, etc. etc.)