Interbase & Firebird, corruptions reasons

by Alexey Kovyazin and Dmitry Kuzmenko, 2005-2007

Hardware-related errors

Abnormal shutdown

Abnormal shutdowns are the leading cause of corruption. They can be caused by power loss on a computer with no UPS when a big mutant mole eats your city's power supply cable (or any other explanation your energy company might give), or the cleaning lady pulls the “wrong” cable while cleaning your office. Sometimes people just power off their computers without concern for what the machine might be engaged in. Any of these conditions can lead to corruption.

Nonetheless, you have probably observed that not every abnormal shutdown leads to corruption. The initial design of InterBase was, with some restrictions, forgiving toward such unstable environments. As you may know, earlier versions of InterBase were used in the fire control system of the MLRPS artillery platform. Every volley of MPLRS was accompanied by a strong electromagnetic pulse, causing the on-board computerto reboot every time. It was a strong requirement that the on-board database server be able to reload in seconds be robust against the potential for corruption caused by power shutdown. InterBase fitted both conditions: it started quickly and its multi-generation architecture made it capable of bypassing uncommitted or even corrupted versions of records, and retain its ability to read good records.

But time passes. In later versions advanced caching was implemented and “civilian” InterBase versions became more vulnerable in abnormal shutdowns. The best-known problem is related to Forced Writes on Windows. Forced writes is a flag, set on the database file, that determines Windows caching behavior for this file. We strongly recommend setting Forced Writes on Windows to ON, because Windows is very “lazy” about releasing its cache and could be holding days of unflushed work.

Light HDD corruption

Light corruption of a hard drive happens when just a few clusters become corrupted. Usually, the operating system warns that the file is unreadable. The effect of such corruption is that the database file has several gaps within it that are filled with zeros or, occasionally, garbage information. The gaps break the internal structure of the database, which can give rise to a wide range of possible errors.

Heavy HDD corruption4

Sometimes the hard drive can shatter into a completely unreadable pile of metal and plastic. In this case you have two options, the first of which is to try repairing the corrupted drive with a special utilitiy like R-Studio (http://www.data-recovery-software.net/)
If that doesn't help, you can enlist the aid of a hard drive recovery service. These guys can really fetch data from the Fourth Dimension. Two problems lie here: first, their fees are rather large (starting from USD$1000), meaning you have to weigh that cost up against your estimate of the value of the lost data before ordering the service. Secondly, recovered data are almost always mixed – by this, I mean that chains of clusters are arranged in ways that differ from the original database.

CD/DVD corruption

If you store databases on DVDs or CDs – for archiving purposes or as a read-only dictionary database – they may break. Usually the showing of corruption is that you can't read the database file from the DVD.

The first thing to do is to extract the corrupted file from the DVD with some tool like FixDVD!

Unfortunately, in 99% of cases, a database file extracted from a DVD is in a bad state: the extracted

file is the right size but it is filled with a mixture of database pages and garbage/zero data.

The most affordable way when extracting database files from DVD is to create an image of the entire DVD (4.7Gb) and look for database pages in this area.

Flash Drive corruption

Flash drive is rather recent technology, with some limitations on the count of read/write cycles. Early versions were incapable of sustaining even a million cycles, but the problem seems to be gone as the technology has evolved. However, I cannot recommend flash drives for everyday usage as the main data storage for a Firebird or InterBase database.

The work of a database server involves many read/write operations in random access mode. We have handled several corrupted databases living on flash drives whose users were not prudent enough to back up. Corruptions are similar to light HDD corruptions – several pieces of the database file lost.

RAM corruption

Amongst all hardware corruptions, RAM corruption is the real nightmare. In general, it's fortunate that RAM corruption becomes apparent with a BSOD (Blue Screen Of Death) or other critical events which can be easily detected by system administrator. But sometimes RAM corruption is so slight that only special tools can detect it and it shows its teeth only during intensive usage.

When other hardware corruptions occur, it's at the database page level. Whole pages are missed while other pages are intact. The problem when RAM becomes corrupted is that any bit in the database file can be intermittently changed from 0 to 1, or vice versa. This kind of corruption is recognizable only after the fact: it shows up only when some database page become sufficiently corrupted to trigger an error.

So, RAM corruption is hidden until the level of damage becomes critical. I once saw a couple of databases from a single server with corrupted RAM. The customer sent them one by one, with various corruptions, before I asked him to send me the ***interbase.log****. There, I saw multiple Wrong record length errors, Wrong page type and even several esoteric errors. We tested the RAM with the* ***memtest*** *tool and found RAM problems.*

A further problem exhibited with RAM corruption is that attempts to validate the database with gfix on the computer with the bad RAM can produce different results each time you run it. Worse, the work of *gfix -mend,* which tries to fix errors, can produce additional corruption of the database because it visits every database page and, in writing of “mended”pages, it can lay down more wrong bits.

Lack of disk space for the database

Running out of disk space is the favorite error of lazy administrators. Corruption happens when the server tries to request more page to extend the database file and discovers that there is no space available on the disk or partition.

The most dangerous situation occurs when lack of disk space is combined with a large cache and forced writes off. The operating system tries to flush a large amount of data to disk and simply fails if there is not enough room. In this case the database will be inconsistent, because loss of the cache means that all changes in page and record chains were interrupted.

When you try to repair corruption caused by lack of disk space with GFIX you may find interesting side effect: the interbase.log will be filled with a looping sequence of “page doubly allocated” errors. Gfix will never finish, the interbase.log can grow very large and disk space can be exhausted again.

Lack of disk space for temp files

If you don't have space on disk configured for InterBase or Firebird to use for storing the temporary files it creates for sorting and merge operations, the engine will use the directory specified in the system variable TEMP. If a heavy query has millions of rows to sort, the overall size of the temp files can be very large. If many queries with sorts are running, it can occupy a lot of space and

exhaustion of free space becomes a possibility. Usually such conditions are handled correctly and client that launched query receives an error message.

The funny thing about it is that, in old versions of InterBase and Firebird, the error text was "No paper in printer" due to a wrong assignment of the exception message to the Windows system error code.

But it can happen that lack of disk space for temp files leads to abnormal server termination and corruption of databases, especially with an older version of InterBase/Firebird.

Lack of disk space for interbase.log or firebird.log

If you don't watch the amount of free space on the partition where your InterBase or Firebird is installed, you can run out of free space there when the interbase.log or firebird.log file grows very large. All errors in all databases under the server are written down to the same log file so you are likely to run into the problem if you have a lot of network errors, such as:

My_server(Server) Sat Jan 02 15:14:57 2006
* INET/inet_error: read errno = 10054*

Now, you might well assume that such errors are rather rare and, with properly designed applications, should never appear. While that's true, there are a lot of cases where free space does get exhausted by the log file blowing out.

It's simple to find an example. Imagine you have light corruption of an index in a rather big database and you decide to run gfix to fix it. What happens here is that the engine marks the corrupted index as wrong, frees all its pages and logs the message "Page XXX orphan". With a large enough index, you'll get thousands of such messages in the log file that can easily eat up all available free space and lead to much heavier corruption.

Corruptions caused by maintenance carelessness

Accidentally deleted database file

Sometimes a database file can be accidentally deleted. In versions prior to Firebird 1.0 and InterBase 7.0 it was possible to delete a database file even if it was open, i.e., during active I/O operations. The server opened the database file with the flag fmShareDenyNone (0x40), so any process was able to modify and delete it.

Even with the newest versions there is no guarantee that a database file would not be deleted while the server is off or no connections are active. Usually it happens when the database has been backed up or copied to another location and it is wrongly considered that the original database can be deleted. If the backup failed or the destination file was corrupted during the file copy, we have a deleted database file situation.

The most urgent and immediate thing to do is to stop all activity on the disk where database was situated. If it is a system disk, power the computer down and remount the disk as secondary to avoid any writes to it.

Next, you need to find some software for undeleting files. For Windows and Linux there are many utilities and how-to guides for retrieving and repairing files. Some of them are listed below:

http://www.stud.tu-ilmenau.de/~mojo/undelete.html

http://recover.sourceforge.net/linux/

http://home.fnal.gov/~muzaffar/undelete/README.html

If no writes have been done over the deleted file sectors, you have a good chance to retrieve the deleted file, perhaps even unscathed and workable. Usually, though, some parts are lost and additional recovery effort is needed.

Disappeared" files on Linux/Unix/HP-UX/...

It is well-known fact that Linux uses the ***inode*** mechanism to support different file systems. One of the key features of this mechanism is the use of cache to handle file descriptors – it means that file descriptors are stored both in memory and on disk.

To InterBase and Firebird it brings an onerous side-effect. If you replace a database when users are still connected, the server will continue to work with the old file, which is wrongly assumed to be deleted. The danger here is that, when the last user detaches, the server will drop the file forever and the "new" file steps in to replace it at that point. You never know it has happened until it is too late and then, it is most likely to be discovered by furious users: "Where is my work from last week?!"

The longest period of lost data due to such «disappearing» that I have observed was 1.5 years. It was a multi-volume database on Linux and one of the 4Gb volumes was completely lost.

You may say it is a very rare circumstance but I can stake a case of beer on the fact that, right now, at least one hundred server installations have this problem. We receive at least one repair request due to this problem every two months!

Erroneous implementation limits

36.6Gb limit of table's size

Working at interesting technical support incident with one of our customers, we found what the actual maximum number of rows differs from declared limit.

In the documentation set [Operations Guide for InterBase 6, page 27, InterBase Specification] we can read the following:

"Maximum number of rows and columns per table: By design, 2^32 rows, because rows are enumerated with a 32-bit unsigned integer per table."

Of course, rows enumerated by 32-bit integer. Sadly, but actually no table ever can reach 2 billion records limit - even the table with only the one column.

The reason of such behaviour is in the algorithm of calculation of free space for new (inserted) record - the integer overflow may occur and you will see the following error message in interbase.log:

pointer page vanished from DPM_next (249)

And, the reality is, that database page size or row size doesn't affect this limit. The limit is a magic table size which is always the same (~36.6 gigabytes) and can be calculated in pages for any database page size as:

Maximum page count for one table can be calculated as
MaxDataPageCount = (MaxInt / PageSize) * 17.476

MaxInt, of course, = 2147483647. Place your database page size instead of PageSize and the result will show how much pages can be allocated for any table

For example, table can't have more than ~9 million data pages in the database with 4K page size.

From the point of view of record count, table with 2 integer columns can't grow larger than 600 million (!) records (don't forget that every record have 14 bytes header).

Confused? Multiply MaxDataPageCount by PageSize, and divide result with your biggest table average record size - you'll know when this table will exceed the limit and your database will stop working .

This limit was fixed only in Firebird 2. It still exists in InterBase 2007.

Addition*: another side of this limit was discovered recently in garbage collector thread implementation in Firebird 2.0.3: this error can appear in 36.6Gb tables during garbage collection (it's caused by combination of special conditions). We hope it will be fixed in the next sub-release 2.0.4 and in 2.1 for sure.*

BLOB maximum size restrictions

BLOBs mechanism have implementations imperfection. In earlier versions of InterBase (until 7.1) it was 512Mb, then it was fixed up to 2Gb. Currently we have information about DVD images (4.7Gb) which were stored in Firebird 2.0.3. Anyway we do not recommend to store big files (>100Mb) inside Firebird or InterBase databases until it's absolutely necessary (in this case we recommend to test the limits first).

Too many transactions in pre InterBase 6.5 versions

InterBase in pre 6.5 versions has a limit of maximum transaction numbers (depends on page size) until restore (after backup/restore transactions start again from scratch).

Critical number of transactions in pre 6.5 InterBase servers

Database page size	Critical number of transactions
1024 byte	131 596 287
2048 byte	265 814 016
4096 byte	534 249 472
8192 byte	1 071 120 384

Too many generators

For InterBase with versions less than 6.0.1.6 we have one more erroneous limit: maximum number of generators in the database (depends on page size).

Critical number of generators in early InterBase versions

Version	Page size=1024	Page size=2048	Page size=4096	Page size=8192
Pre 6	248	504	1016	2040
6.0.x	124	257	508	1020

Corruptions caused by SQL

Creating and dropping tables during intensive users' work

Well, it's the top one from the most often ways to corrupt database. Creating and dropping tables while users are writing something into database can lead to confusion between system pages (used by created/dropped tables) and user data. Usually such corruptions leads to moderate loss of data and can be recovered using manual IBSurgeon service.