Rebuilding Zen/PSQL Database Files

A White Paper From

For more information, see our web site at
http://www.goldstarsoftware.com

Rebuilding Zen/PSQL Database Files

Last Updated: November 2018

The Actian Zen database engine, formerly known as Actian PSQL, has a long history, starting with the Btrieve database engine first seen in the early 1980's. Even though the application programming interface (API) has remained the same over these many years, the file structure of the underlying files has needed to be changed slightly from time to time in order to support newer database features, such as larger file sizes, page compression, and other new functionality. In most cases, the engine has been modified to support the newer formats in addition to the older formats, so that users with older database files can continue to function on the older format until the newer features are actually required. Further, this has allowed applications to work on newer engines without modification and has been a great benefit for PSQL developers everywhere.

The current Actian Zen engines can actually read files in ALL file formats, going back as far as Btrieve v3! If you want to write to the database files, though, you must have a minimum of the Btrieve 6.x file format, as PSQL v9 and newer engines will simply refuse to write to those older formats due to the possibility of data corruption. Because of this flexibility, you are not required to upgrade your files when you upgrade your database engine. In fact, you can continue to run the old file format for whatever period of time you wish, until you are certain that there will be no need to ever go back to the previous database version. This makes it far superior to those OTHER databases that require a full rebuild and have an extensive recovery process if you ever need to roll back to the older version.

However, database file formats do dictate some important features and limitations:

- **Features**: Newer formats provide additional features that may improve performance of specific data structures, like VATs.
- **Stability**: Newer formats are typically more stable on the newer engines, since they are better tested.
- **Performance**: Newer formats can provide additional performance benefits over older versions. This was especially true of the upgrade from the older formats to 6.x format with the introduction of *shadow paging*, and it was seen once again in the upgrade to the 8.x format due to the *Turbo Write Accelerator*, which provides an additional performance boost on database writes.
- **Maximum File Size**: The 5.x and 6.x file formats supports at most a 4GB file size. The 7.x and 8.x formats allow files to grow to a logical 64GB in size, depending on the page size of the file. It does this through the use of 2GB extents, that have names like *filename.^01*. The 9.0 format supports a maximum file size of 128GB, and the 9.5 file format supports a maximum file size of 256GB.
- **Segmentation**: The newer 9.x file formats also allow you to store the entire data file inside of a single physical file extent on the disk, eliminating the need for the ^01, ^02, and other files.

- **Page Size**: Older engines support file page sizes between 512 and 4096. However, the 9.0 file format adds support for an 8K page, and the 9.5 format adds a 16K page (and eliminates a number of "bad" page sizes), allowing the OS reads and writes to be more efficient, too.

Of these items, there is one critical issue: Stability. Pervasive Software has determined that the 5.x file format, which is really unstable on modern caching operating systems, should not be used any longer. As such, they changed the Pervasive PSQL v9 and newer database engines to support reading of 5.x (or older) files only, and these engines do NOT allow you to write to these files. As such, you WILL need to rebuild any 5.x or older files to 6.x or newer to run on these new engines.

To take advantage of the other benefits, you may find that you must rebuild your files. Of course, before you start, you should contact your application vendor to verify that they will support you if you have the newest files, or if they have any advice about the rebuilding process. I have seen some applications that simply refuse to work if the files are not exactly as specified by the vendor! Further, if you rebuild your files to a newer format than the vendor supports and need to send them your database for support, they may have problems reading the data and be unable to assist you.

Important Note**: The rebuild process defined here is ****NOT* for corrupted database files! If you suspect that one or more of your files are damaged, which can be evidenced by Status 2, Status 54, or System Errors being reported in the Pervasive Event log, then you should use the File Recovery Process instead. Failure to observe this caveat can result in a failed rebuild, or in substantial loss of data in extreme cases.*

As indicated above, the Rebuild process can update the file format to a newer format, which can provide several benefits. However, even if you are not changing formats, the rebuild process can often be used to shrink your database files, providing a way to reclaim disk space after a mass purge (delete) of records. (Btrieve files never shrink on their own, so this is an important step if you periodically archive data.) Additionally, the process can re-write your indices to make them more compact (and thus increase efficiency). Finally, the rebuild process can re-order the data physically in your file, to store the data physically by a given index. While this is not a true *clustered index* like other database engines, it can reorder the data in the physical order by that key at least until the data starts changing again.

If you are running PSQL v12 or newer and do not need to change the file version, but only need to optimize the file structure, then you may find that the Online Database Defragmenter works better for you, as it does not require a downtime window to get the job done.

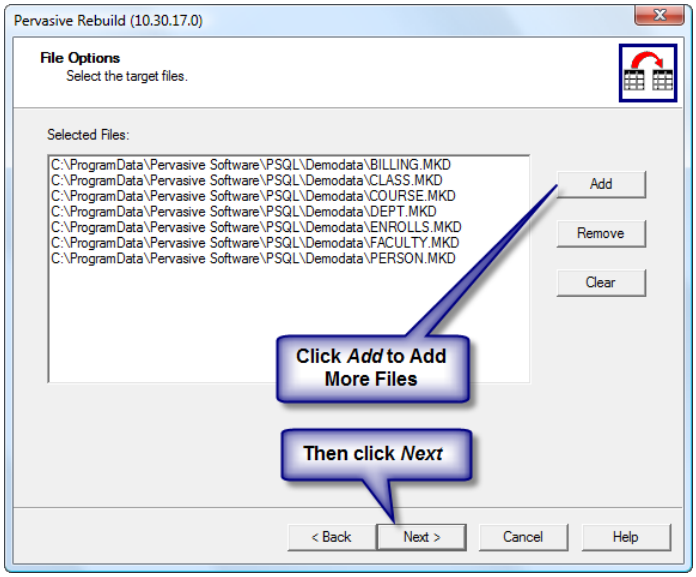
There are three primary tools for rebuilding files, the Rebuild Utility (a GUI utility shipping with all products above Pervasive.SQL V7, BREBUILD (shipping with NetWare and older DOS engines), and RBLDCLI (a command line utility that ships with Pervasive.SQL V8.5 or newer). The instructions for running all three can be found in the product manuals for your database engine. Users of Pervasive.SQL 2000i and above can look in the **Advanced Operations Guide** under the chapter entitled *Converting Pervasive.SQL Data* or *Rebuilding Data Files*.

Using the Graphical Rebuild Utility

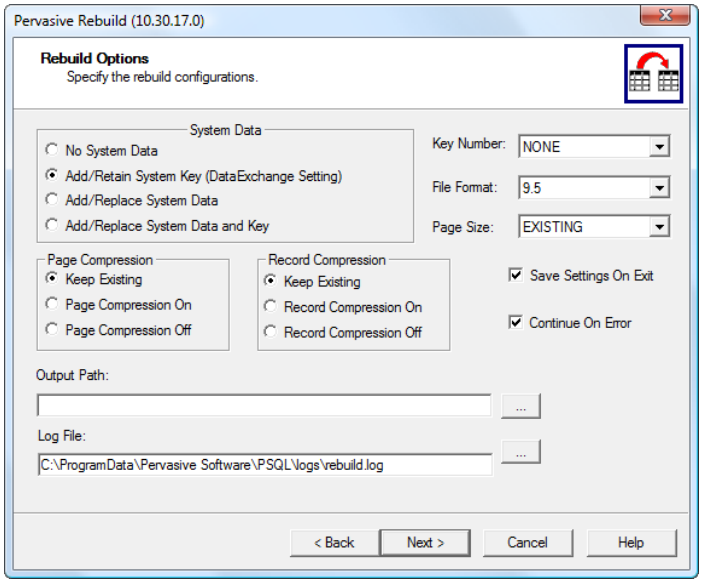
The GUI-based Rebuild Utility (also known as the Rebuild Wizard) is the easiest way to rebuild files if you are not familiar with command-line applications, because it uses the standard Windows graphical user interface to select files and options.

There are a few different versions of this tool, depending on your exact version of PSQL, so your screen shots may differ slightly. If in doubt, you can always refer to your online product manuals for current information for your release.

1. Make sure that all users are out of the database files and that all files are closed in the PSQL Monitor. If in doubt, reboot the database server! If any users are in the system, then the rebuild process will fail.
2. From the **File Options** screen, click the *Add* button to add files to the list to be processed.



3. Then, click Next to go to the Rebuild Options dialog box.



4. Verify the various options:
    - **System Data**: The *Add/Retain System Key* option must be selected if you are planning to use DataExchange, Pervasive's replication solution. Otherwise, this is optional.
    - **Page Compression**: The default is fine here, unless you want to enable or disable page compression. Note that many file recovery tools will not be able to recover page-compressed files.
    - **Record Compression**: The default is fine here, unless you want to enable or disable record compression. Record compression can save space for some tables, but at the cost of performance.
    - **Output Path**: Leave this blank to update the files in place. Set this directory if you want to rebuild them into a new location.
    - **Log File**: This is the log file for the rebuild. The default is fine, unless you want a specific log written.
    - **Key Number**: Leave to *NONE* for the best performance. Data files for some applications MAY require using a specific key. Check with your vendor for details. You can also specify a key to create a psuedo-*clustered index*.
    - **File Format**: Select the new file format in which you want the resulting files. Remember that older engines cannot read newer files, but newer engines can read all older files.
    - **Page Size**: Leave at *Existing* if you trust the original application developers, or change to 4096, 8K, or 16K for the best performance in many cases.
    - **Save Settings On Exit**: Saves these settings for the next time you run the Rebuild program.
    - **Continue on Error**: If doing one file, set to *No*. Otherwise, set this to *Yes* to have the system work on the next file if an error occurs. Be sure to

check the log file after a run to make sure that all files were converted properly.

5. Click *Next* to start the rebuilding process. Each selected database file will be rebuilt with the settings you specified.
6. Finally, check the rebuild log file for any errors. If there were no problems, allow users back into the application and enjoy the performance gains and disk savings of newly rebuilt files!

Using the Command Line Rebuild Utility

On NetWare servers, the command-line application for rebuilding files program is called BREBUILD.NLM, and it runs directly on the database server. The base command is LOAD BREBUILD <filespec>, and you will add the following options as needed.

Newer engines also include a command-line tool called RBLDCLI, that provides the same functionality from Windows servers or workstations. (Note that while rebuilding CAN be done from a workstation in a client/server configuration, we recommend always doing it at the server for best performance.)

With these command-line utilities, there is no graphical interface to select files and options. Instead, everything is done via command line options. You can see a complete list of options by just running "rbldcli" by itself from a Command Prompt window, but we'll like the more critical options for you here:

- -c: This flag continues with the next file if an error occurs. We recommend using this every time you rebuild files, unless you plan to watch it run to completion.
- -p####: If the -p option is specified by itself, the optimal page size is selected. If you specify a new page size (the number after the "p"), then the specified page size will be used. If omitted, the existing page size is used in the new file.
- -k##: Use the -k option to rebuild files on a given key.
- -f#: Specifies the file version to which you want the files rebuilt.. For example, specify –f7 to force the files into the 7.x format, -f9 to force a 9.0 format, and -f95 to force a v9.5 file format.

Note that the command line applications also take a “filespec” – an indication of the files to rebuild that can include wildcards. This means that you can do an entire directory in one simple command like this:

```
rbldcli -c -f95 *.*
```

When doing an entire folder, there is always the possibility that one or more files will fail to rebuild correctly. You’ll want to check out the text file RBLDCLI.LOG when the process completes to verify that it all went as expected.

Remember that NetWare servers have no concept of a current directory, so you need to specify the entire path from the volume, as in the following example:

```
LOAD BREBUILD –f95 -c VOL1:APPS\DATA\*.BTR
```

Be sure to check the SYS:SYSTEM\BREBUILD.LOG file when finished to verify that all files rebuilt successfully.

Optimizing the Rebuild Time

The file rebuilds **must** be done during system downtime, since all users must be out of the files for the rebuild to be successful. The most common question we get is: "How long will my rebuild take?" The answer, of course, is complicated.

While small files will rebuild very rapidly and may take only seconds to finish, larger files on slower servers will take a long time to rebuild, as will huge files on even the fastest server. There is simply no way to tell how long a process is going to take in advance, as it varies based on MANY factors, including free memory available, speed of the disk drives, size of the database cache, size of the data file, the use of compression or encryption, and more.

Remember that you do NOT need to rebuild all files at the same time, since the engine supports accessing mixed file formats freely. You can run some tests on backup copies of your files, but remember that the larger a file is, the longer the time will be, and it can be exponentially longer on some systems.

To optimize the system for rebuilding, you want to split your system memory about 50/50 between the database cache and free memory. This means that on Pervasive.SQL 2000i or lower, your database cache should be set to a reasonable limit, such that there is sufficient free memory left over afterwards. On Pervasive.SQL V8 and above, you should *also* disable the Level 2 (L2) cache by setting the Maximum Memory Size down to 0%. If you have lots of memory and a 64-bit system with PSQL v10 or newer, consider setting the L1 cache to approximately 50% of the total server memory. Remember to change any settings back when you are done, and don't forget that you must restart the engine to allow the change to take effect!

Of course, if you would rather a professional handle this task for you, please contact Goldstar Software for help!