EnterpriseDB

The Enterprise PostgreSQL Company

Postgres Plus® Advanced Server Performance and Scalability Guide

Postgres Plus Advanced Server 9.2

November 8, 2012

Postgres Plus Advanced Server Performance Features Guide, Version 3.0
by EnterpriseDB Corporation

EnterpriseDB Corporation, 34 Crosby Drive, Suite 100, Bedford, MA 01730, USA
T +1 781-357-3390 F +1 978-589-5701 E info@enterprisedb.com www.enterprisedb.com

Table of Contents

1 Introduction

This guide describes the performance features found in Postgres Plus Advanced Server.

- Infinite Cache allows you to utilize memory on other computers connected to your network to increase the amount of memory in the shared buffer cache.

- Dynatune makes optimal use of the system resources that are available on the host machine.

- The Dynamic Runtime Instrumentation Tools Architecture (DRITA) records *wait events* that affect system performance; DRITA also offers a set of tools for inspecting those events.

- Optimizer Hints are directives that you embed in comment-like syntax immediately following the `SELECT, UPDATE`, or `DELETE` key words to influence the query optimizer.

1.1 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the proceeding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

2 Infinite Cache

Database performance is typically governed by two competing factors:

- Memory access is fast; disk access is slow.
- Memory space is scarce; disk space is abundant.

Postgres Plus Advanced Server tries very hard to minimize disk I/O by keeping frequently used data in memory. When the first server process starts, it creates an in-memory data structure known as the *buffer cache*. The buffer cache is organized as a collection of 8K (8192 byte) pages: each page in the buffer cache corresponds to a page in some table or index. The buffer cache is shared between all processes servicing a given database.

When you select a row from a table, Advanced Server reads the page that contains the row into the shared buffer cache. If there isn't enough free space in the cache, Advanced Server *evicts* some other page from the cache. If Advanced Server evicts a page that has been modified, that data is written back out to disk; otherwise, it is simply discarded. Index pages are cached in the shared buffer cache as well.

Figure 1.1 demonstrates the flow of data in a typical Advanced Server session:

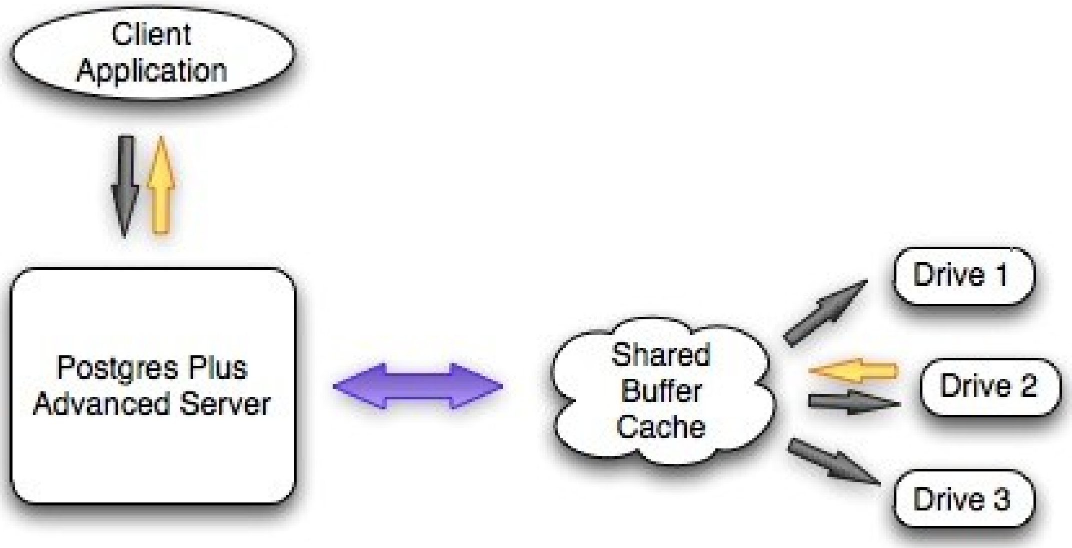


Figure 1.1 – Data Flow

A client application sends a query to the Postgres server and the server searches the shared buffer cache for the required data. If the requested data is found in the cache, the server immediately sends the data back to the client. If not, the server reads the page that holds the data into the shared buffer cache, evicting one or more pages if necessary. If the server decides to evict a page that has been modified, that page is written to disk.

As you can see, a query will execute much faster if the required data is found in the shared buffer cache.

One way to improve performance is to increase the amount of memory that you can devote to the shared buffer cache. However, most computers impose a strict limit on the amount of RAM that you can install. To help circumvent this limit, Infinite Cache lets you utilize memory from other computers connected to your network.

With Infinite Cache properly configured, Advanced Server will dedicate a portion of the memory installed on each *cache server* as a secondary memory cache. When a client application sends a query to the server, the server first searches the shared buffer cache for the required data; if the requested data is not found in the cache, the server searches for the necessary page in one of the cache servers.

Figure 1.2 shows the flow of data in an Advanced Server session with Infinite Cache:

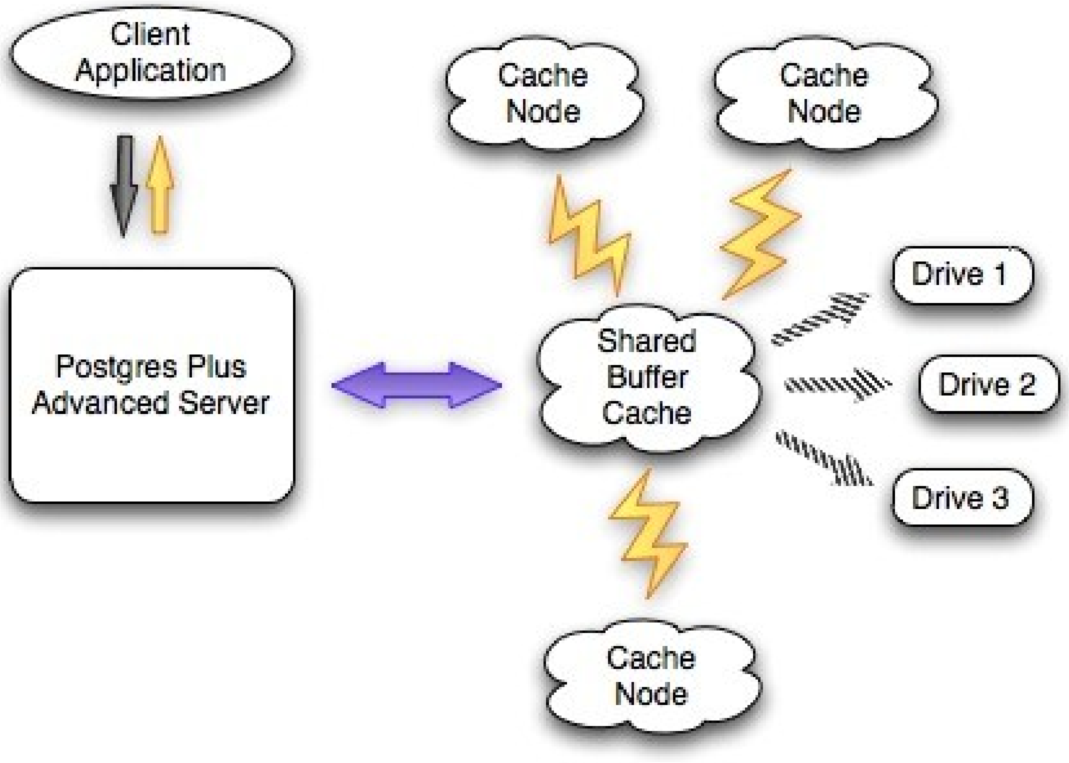


Figure 1.2 – Data flow with Infinite Cache

When a client application sends a query to the server, the server searches the shared buffer cache for the required data. If the requested data is found in the cache, the server immediately sends the data back to the client. If not, the server sends a request for the page to a specific cache server; if the cache server holds a copy of the page it sends the data back to the server and the server copies the page into the shared buffer cache. If the required page is not found in the primary cache (the shared buffer cache) or in the secondary cache (the cloud of cache servers), Advanced Server must read the page from disk. Infinite Cache improves performance by utilizing RAM from other computers on your network in order to avoid reading frequently accessed data from disk.

Updating the Cache Node Configuration

You can add or remove cache servers without restarting the database server by adding or deleting cache nodes from the list defined in the `edb_icache_servers` configuration parameter. For more information about changing the configuration parameter, see Section 2.2.2.

When you add one or more cache nodes, the server re-allocates the cache, dividing the cache evenly amongst the servers; each of the existing cache servers loses a percentage of the information that they have cached. You can calculate the percentage of the cache that remains valid with the following formula:

```
    (existing_nodes * 100) / (existing_nodes + new_nodes)
```

For example, if an Advanced Server installation with three existing cache nodes adds an additional cache node, 75% of the existing cache remains valid after the reconfiguration.

If cache nodes are removed from a server, the data that has been stored on the remaining cache nodes is preserved. If one cache server is removed from a set of five cache servers, Advanced Server preserves the 80% of the distributed cache that is stored on the four remaining cache nodes.

When you change the cache server configuration (by adding or removing cache servers), the portion of the cache configuration that is preserved is not re-written unless the cache is completely re-warmed using the `edb_icache_warm()` function or `edb_icache_warm` utility. If you do not re-warm the cache servers, new cache servers will accrue cache data as queries are performed on the server.

Infinite Cache Offers a Second Performance Advantage: Compression.

Without Infinite Cache, Advanced Server will read each page from disk as an 8K chunk; when a page resides in the shared buffer cache, it consumes 8K of RAM. With Infinite Cache, Postgres can *compress* each page before sending it to a cache server. A compressed page can take significantly less room in the secondary cache, making more space available for other data and effectively increasing the size of the cache. A compressed page consumes less network bandwidth as well, decreasing the amount of time required to retrieve a page from the secondary cache.

The fact that Infinite Cache can compress each page may make it attractive to configure a secondary cache server on the same computer that runs your Postgres server. If, for example, your computer is configured with 6GB of RAM, you may want to allocate a smaller amount (say 1GB) for the primary cache (the shared buffer cache) and a larger amount (4GB) to the secondary cache (Infinite Cache), reserving 1GB for the operating system. Since the secondary cache resides on the same computer, there is very little overhead involved in moving data between the primary and secondary cache. All data stored in the Infinite Cache is compressed so the secondary cache can hold many more pages than would fit into the (uncompressed) shared buffer cache. If you had allocated 5GB to the shared buffer cache, the cache could hold no more than 65000 pages (approximately). By assigning 4GB of memory to Infinite Cache, the cache may be able to hold 130000 pages (at 2x compression), 195000 pages (at 3x compression) or more. The compression factor that you achieve is determined by the amount of redundancy in the data itself and the `edb_icache_compression_level` parameter.

To use Infinite Cache, you must specify a list of one or more cache servers (computers on your network) and start the `edb_icache` daemon on each of those servers.

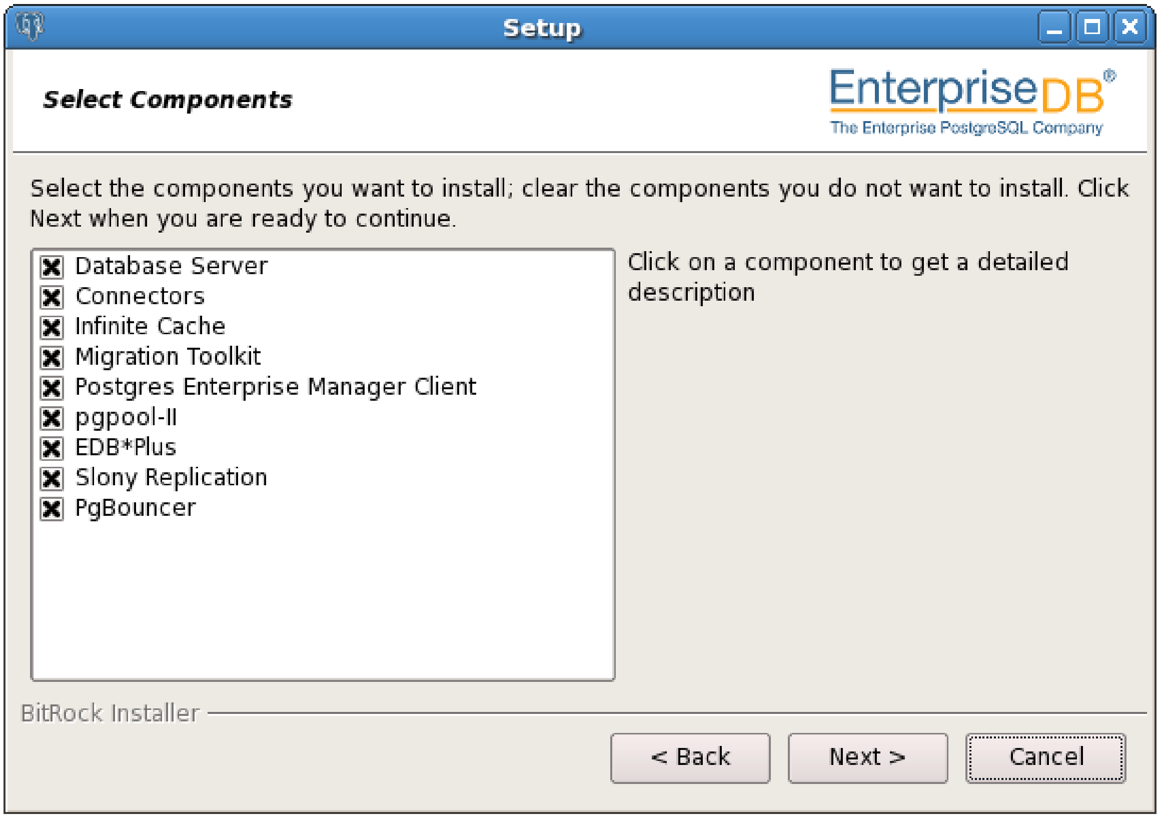
Infinite Cache is supported on Linux, HPUX and Solaris systems only.

Please Note: Infinite Cache and the `effective_io_concurrency` parameter can potentially interfere with each other. You should disable asynchronous I/O requests (by setting the value of `effective_io_concurrency` to `0` in the `postgresql.conf` file) if you enable the Infinite Cache feature.

2.1 Installing Infinite Cache

Postgres Plus Advanced Server includes Infinite Cache functionality as part of a standard installation on a Linux, HPUX or Solaris system. The Advanced Server installation wizard can optionally install only the Infinite Cache daemon on supporting cache servers without installing Advanced Server.

To install Advanced Server with Infinite Cache functionality, confirm that the box next to the `Database Server` option (located on the `Setup: Select Components` window, shown in Figure 1.3) is selected when running the installation wizard.



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

☒ Database Server
☒ Connectors
☒ Infinite Cache
☒ Migration Toolkit
☒ Postgres Enterprise Manager Client
☒ pgpool-II
☒ EDB*Plus
☒ Slony Replication
☒ PgBouncer

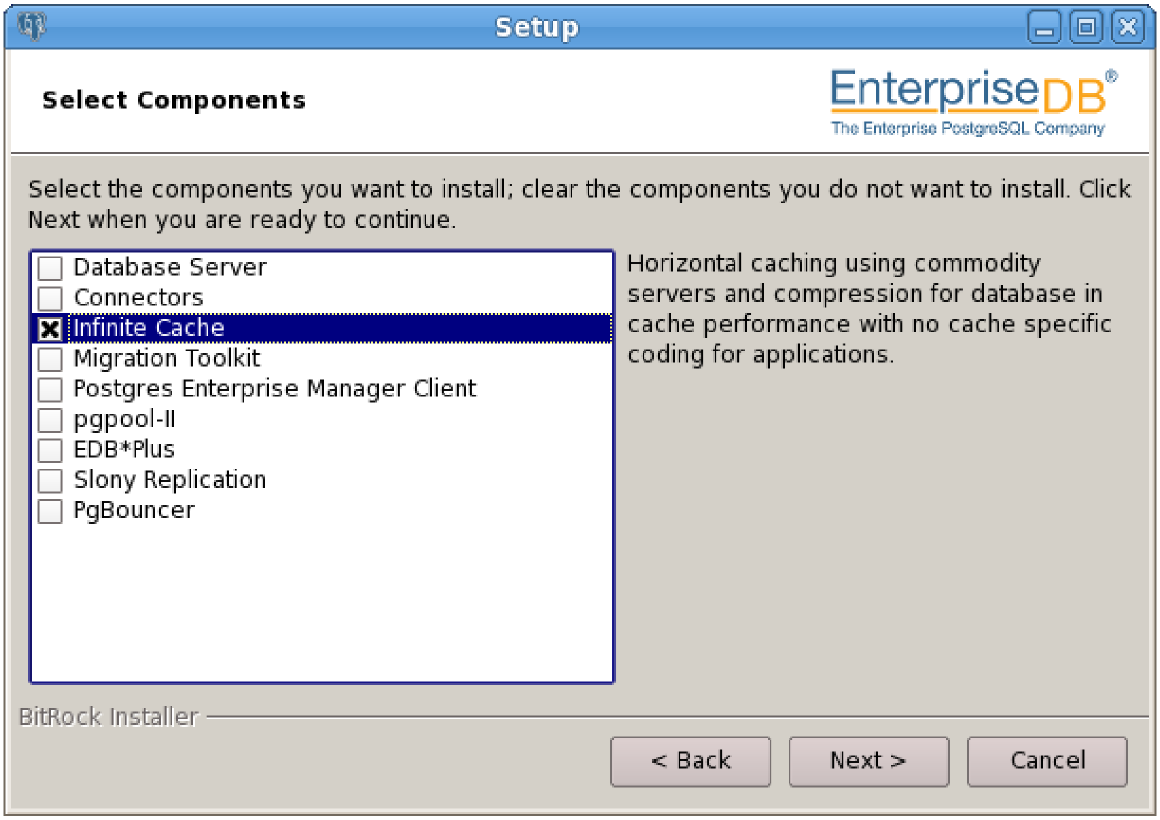
Click on a component to get a detailed description

Figure 1.3: The `Setup: Select Components` window.

Selecting the `Database Server` option installs the following Infinite Cache components:

- The `ppas-infinitecache-9.2` service script.
- The Infinite Cache configuration file (`ppas-infinitecache-9.2`).
- A command line tool that allows you to pre-load the cache servers (`edb-icache-warm`).
- The `edb_icache` libraries (code libraries required by the `edb-icache` daemon).

The installation wizard can selectively install only the Infinite Cache daemon on a cache server. To install the `edb-icache` daemon on a cache server, deploy the installation wizard on the machine hosting the cache; when the `Setup: Select Components` window opens, de-select all options except `Infinite Cache` (as shown in Figure 1.4).



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☐ Database Server
- ☐ Connectors
- ☒ Infinite Cache
- ☐ Migration Toolkit
- ☐ Postgres Enterprise Manager Client
- ☐ pgpool-II
- ☐ EDB*Plus
- ☐ Slony Replication
- ☐ PgBouncer

Horizontal caching using commodity servers and compression for database in cache performance with no cache specific coding for applications.

Figure 1.4: Installing only the `Infinite Cache Daemon`.

The `Infinite Cache Daemon` option installs the following:

- The `ppas-infinitecache-9.2` service script.
- The Infinite Cache configuration file (`ppas-infinitecache-9.2`).
- A command line tool that allows you to pre-load the cache servers (`edb-icache-warm`).
- The `edb_icache` libraries (code libraries required by the `edb-icache` daemon).

2.2 Configuring the Infinite Cache Server

Configuring Infinite Cache is a three-step process:

- Specify Infinite Cache server settings in the Infinite Cache configuration file.
- Modify the Advanced Server `postgresql.conf` file, enabling Infinite Cache, and specifying connection and compression settings.
- Start the Infinite Cache service.

2.2.1 Modifying Infinite Cache Settings

The Infinite Cache configuration file is named `ppas-infinitecache-9.2`, and contains two parameters and their associated values:

```
PORT=11211
CACHESIZE=500
```

To modify a parameter, open the `ppas-infinitecache-9.2` file (located in the `/etc` directory under your Advanced Server installation) with your editor of choice, and modify the parameter values:

`PORT`

Use the `PORT` variable to specify the port where Infinite Cache will listen for connections from Advanced Server.

`CACHESIZE`

Use the `CACHESIZE` variable to specify the size of the cache (in MB).

2.2.2 Enabling Infinite Cache

The `postgresql.conf` file includes three configuration parameters that control the behavior of Infinite Cache. The `postgresql.conf` file is read each time you start the Advanced Server database server. To modify a parameter, open the `postgresql.conf` file (located in the `$PGDATA` directory) with your editor of choice, and edit the section of the configuration file shown below:

```
# - Infinite Cache
#edb_enable_icache = off
#edb_icache_servers = '' #'host1:port1,host2,ip3:port3,ip4'
#edb_icache_compression_level = 6
```

Lines that begin with a pound sign (`#`) are treated as comments; to enable a given parameter, remove the pound sign and specify a value for the parameter. When you've updated and saved the configuration file, restart the database server for the changes to take effect.

edb_enable_icache

Use the edb_enable_icache parameter to enable or disable Infinite Cache. When edb_enable_icache is set to on, Infinite Cache is enabled; if the parameter is set to off, Infinite Cache is disabled.

If you set edb_enable_icache to on, you must also specify a list of cache servers by setting the edb_icache_servers parameter (described in the next section).

The default value of edb_enable_icache is off.

edb_icache_servers

The edb_icache_servers parameter specifies a list of one or more servers with active edb-icache daemons. edb_icache_servers is a string value that takes the form of a comma-separated list of *hostname:port* pairs. You can specify each pair in any of the following forms:

- *hostname*
- *IP-address*
- *hostname:portnumber*
- *IP-address:portnumber*

If you do not specify a port number, Infinite Cache assumes that the cache server is listening at port 11211. This configuration parameter will take effect only if edb_enable_icache is set to on. Use the edb_icache_servers parameter to specify a maximum of 128 cache nodes.

edb_icache_compression_level

The edb_icache_compression_level parameter controls the compression level that is applied to each page before storing it in the distributed Infinite Cache. This parameter must be an integer in the range 0 to 9.

- A compression level of 0 disables compression; it uses no CPU time for compression, but requires more storage space and network resources to process.

- A compression level of 9 invokes the maximum amount of compression; it increases the load on the CPU, but less data flows across the network, so network demand is reduced. Each page takes less room on the Infinite Cache server, so memory requirements are reduced.

- A compression level of 5 or 6 is a reasonable compromise between the amount of compression received and the amount of CPU time invested.

By default, edb_icache_compression_level is set to 6.

When Advanced Server reads data from disk, it typically reads the data in 8K increments. If edb_icache_compression_level is set to 0, each time Advanced Server sends an 8K page to the Infinite Cache server that page is stored (uncompressed) in 8K of cache memory. If the edb_icache_compression_level parameter is set to 9, Advanced Server applies the maximum compression possible before sending it to the Infinite Cache server, so a page that previously took 8K of cached memory might take 2K of cached memory. Exact compression numbers are difficult to predict, as they are dependent on the nature of the data on each page.

The compression level must be set by the superuser and can be changed for the current session while the server is running. The following command disables the compression mechanism for the currently active session:

```
    SET edb_icache_compression_level = 0
```

The following example shows a typical collection of Infinite Cache settings:

```
edb_enable_icache            = on
edb_icache_servers           = 'localhost,192.168.2.1:11200,192.168.2.2'
edb_icache_compression_level = 6
```

Please Note: Infinite Cache and the effective_io_concurrency parameter can potentially interfere with each other. You should disable asynchronous I/O requests (by setting the value of effective_io_concurrency to 0 in the postgresql.conf file) if you enable the Infinite Cache feature. By default, effective_io_concurrency is set to 1.

2.2.3 Controlling the Infinite Cache Server

Linux

On Linux, the Infinite Cache service script is named ppas-infinitecache-9.2. The service script resides in the /etc/init.d directory. You can control the Infinite Cache service, or check the status of the service with the following command:

```
/etc/init.d/ppas-infinitecache-9.2 action
```

Where *action* specifies:

- `start` to start the service.
- `stop` to stop the service
- `restart` to stop and then start the service.
- `status` to return the status of the service.

HP-UX

On HP-UX, the Infinite Cache service script is named `ppas-infinitecache-9.2`. The service script resides in the `/sbin/init.d` directory. You can control the Infinite Cache service, or check the status of the service with the following command:

```
      /sbin/init.d/ppas-infinitecache-9.2 action
```

Where *`action`* specifies:

- `start` to start the service.
- `stop` to stop the service
- `restart` to stop and then start the service.
- `status` to return the status of the service.

Solaris

On Solaris, the Infinite Cache service is named `ppas-infinitecache-9_2`, and resides in the `/lib/svc/method` directory. After specifying configuration options, you must manually register and start the Infinite Cache service.

On Solaris 10, enter:

```
svccfg -v import /var/svc/manifest/application/ppas-infinitecache-
9_2.xml
```

On Solaris 11, enter:

```
svccfg -v import installation_dir/installer/infinitecache/ppas-
infinitecache-9_2.xml
```

After registering and starting the Infinite Cache service, you can use the following command to check the status of the service

```
      svcs ppas-infinitecache-9_2
```

You can control the Infinite Cache service with the following command:

```
      svcadm action ppas-infinitecache-9_2
```

Where *`action`* specifies:

- `enable` to start the service.
- `disable` to stop the service
- `restart` to stop and then start the service.

2.3 Dynamically Modifying Infinite Cache Server Nodes

You can dynamically modify the Infinite Cache server nodes; to change the Infinite Cache server configuration, use the `edb_icache_servers` parameter in the `postgresql.conf` file to:

- specify additional cache information to add a server/s.

- delete server information to remove a server/s.

- specify additional server information and delete existing server information to both add and delete servers during the same reload operation.

After updating the `edb_icache_servers` parameter in the `postgresql.conf` file, you must reload the configuration parameters for the changes to take effect.

To reload the configuration parameters, navigate through the `Postgres Plus Advanced Server 9.2` menu to the `Expert Configuration` menu, and select the `Reload Configuration` option. If prompted, enter your password to reload the configuration parameters.

Alternatively, you can use the `pg_ctl reload` command to update the server's configuration parameters at the command line:

```
pg_ctl reload -D data_directory
```

Where *`data_directory`* specifies the complete path to the Advanced Server data directory.

Please Note: If Advanced Server detects a problem with the value specified for the `edb_icache_servers` parameter during a server `reload`, it will ignore changes to the parameter and use the last valid parameter value. If you are performing a server `restart`, and the parameter contains an invalid value, Advanced Server will return an error.

2.4 Controlling the edb-icache Daemons

edb-icache is a high-performance memory caching daemon that distributes and stores data in shared buffers. Advanced Server transparently interacts with edb-icache daemons to store and retrieve data.

Before starting Advanced Server, the edb-icache daemons must be running on each server node. Log into each server and start the edb-icache server (on that host) by issuing the following command:

```
    # edb-icache -u enterprisedb -d -m 1024
```

Where:

-u

-u specifies the user name

-m

-m specifies the amount of memory to be used by edb-icache. The default is 64MB.

-d

-d designates that the service should run in the background

To gracefully kill an edb-icache daemon (close any in-use files, flush buffers, and exit), execute the command:

```
    # killall -TERM edb-icache
```

If the edb-icache daemon refuses to die, you may need to use the following command:

```
    # killall -KILL edb-icache
```

2.4.1 Command Line Options

To view the command line options for the `edb-icache` daemon, use the following command from the `edb_Infinite Cache` subdirectory, located in the Advanced Server installation directory:

```
# edb-icache -h
```

The command line options are:

Parameter	Description
-p <port_number>	The TCP port number the Infinite Cache daemon is listening on. The default is 11211.
-U <UDP_number>	The UDP port number the Infinite Cache daemon is listening on. The default is 0 (off).
-s <pathname>	The Unix socket pathname the Infinite Cache daemon is listening on. If included, the server limits access to the host on which the Infinite Cache daemon is running, and disables network support for Infinite Cache.
-a <mask>	The access mask for the Unix socket, in octal form. The default value is 0700.
-l <ip_addr>	Specifies the IP address that the daemon is listening on. If an individual address is not specified, the default value is INDRR_ANY; all IP addresses assigned to the resource are available to the daemon.
-d	Run as a daemon.
-r	Maximize core file limit.
-u <username>	Assume the identity of the specified user (when run as root).
-m <numeric>	Max memory to use for items in megabytes. Default is 64 MB.
-M	Return error on memory exhausted (rather than removing items).
-c <numeric>	Max simultaneous connections. Default is 1024.
-k	Lock down all paged memory. Note that there is a limit on how much memory you may lock. Trying to allocate more than that would fail, so be sure you set the limit correctly for the user you started the daemon with (not for -u <username> user; under sh this is done with 'ulimit -S -l NUM_KB').
-v	Verbose (print errors/warnings while in event loop).
-vv	Very verbose (include client commands and responses).
-vvv	Extremely verbose (also print internal state transitions).
-h	Print the help text and exit.
-i	Print memcached and libevent licenses.
-P <file>	Save PID in <file>, only used with -d option.
-f <factor>	Chunk size growth factor. Default value is 1.25.
-n <bytes>	Minimum space allocated for key+value+flags. Default is 48.
-L	Use large memory pages (if available). Increasing the memory page size could reduce the number of transition look-aside buffer misses and improve the performance. To get large pages from the OS, Infinite Cache will allocate the total item-cache in one large chunk.
-D <char>	Use <char> as the delimiter between key prefixes and IDs. This is used for per-prefix stats reporting. The default is":" (colon). If this option is specified, stats collection is enabled automatically; if not, then it may be enabled by sending the `stats detail on` command to the server.
-t <num>	Specifies the number of threads to use. Default is 4.
-R	Maximum number of requests per event; this parameter limits the number of requests process for a given connection to prevent starvation, default is 20.
-C	Disable use of CAS (check and set).

-b	Specifies the backlog queue limit, default is 1024.
-B	Specifies the binding protocol. Possible values are `ascii`, `binary` or `auto`; default value is `auto`.
-I	Override the size of each slab page. Specifies the max item size; default 1 MB, minimum size is 1 k, maximum is 128 MB).

2.4.2 edb-icache-tool

`edb-icache-tool` provides a command line interface that queries the `edb-icache` daemon to retrieve statistical information about a specific cache node. The syntax is:

```
edb-icache-tool <host[:port]> stats
```

host specifies the address of the host that you are querying.

port specifies the port that the daemon is listening on.

edb-icache-tool retrieves the statistics described in the following table:

Statistic	Description
accepting_conns	Will this server accept new connection(s)? 1 if yes, otherwise 0.
auth_cmds	Number of authentication commands handled by this server, success or failure.
auth_errors	Number of failed authentications.
bytes	Total number of bytes in use.
bytes_read	Total number of bytes received by this server (from the network).
bytes_written	Total number of bytes sent by this server (to the network).
cas_badval	Number of keys that have been compared and swapped by this server but the comparison (original) value did not match the supplied value.
cas_hits	Number of keys that have been compared and swapped by this server and found present.
cas_misses	Number of keys that have been compared and swapped by this server and not found.
cmd_flush	Cumulative number of flush requests sent to this server.
cmd_get	Cumulative number of read requests sent to this server.
cmd_set	Cumulative number of write requests sent to this server.
conn_yields	Number of times any connection yielded to another due to hitting the edb-icache -R limit.
connection_structures	Number of connection structures allocated by the server.
curr_connections	Number of open connections.
curr_items	Number of items currently stored by the server.
decr_hits	Number of decrement requests satisfied by this server.
decr_misses	Number of decrement requests not satisfied by this server.
delete_hits	Number of delete requests satisfied by this server.
delete_misses	Number of delete requests not satisfied by this server.
evictions	Number of valid items removed from cache to free memory for new items.
get_hits	Number of read requests satisfied by this server.
get_misses	Number of read requests not satisfied by this server.
incr_hits	Number of increment requests satisfied by this server.
incr_misses	Number of increment requests not satisfied by this server.

`limit_maxbytes`	Number of bytes allocated on this server for storage.
`listen_disabled_num`	Cumulative number of times this server has hit its connection limit.
`pid`	Process ID (on cache server).
`pointer_size`	Default pointer size on host OS (usually 32 or 64).
`reclaimed`	Number of times an entry was stored using memory from an expired entry.
`rusage_user`	Accumulated user time for this process (seconds.microseconds).
`rusage_system`	Accumulated system time for this process (seconds.microseconds).
`threads`	Number of worker threads requested.
`total_time`	Number of seconds since this server's base date (usually midnight, January 1, 1970, UTC).
`total_connections`	Total number of connections opened since the server started running.
`total_items`	Total number of items stored by this server (cumulative).
`uptime`	Amount of time that server has been active.
`version`	edb-icache version.

In the following example, edb-icache-tool retrieves statistical information about an Infinite Cache server located at the address, `192.168.23.85` and listening on port `11213`:

```
    # edb-icache-tool 192.168.23.85:11213 stats

Field                        Value
accepting_conns              1
auth_cmds                    0
auth_errors                  0
bytes                        52901223
bytes_read                   188383848
bytes_written                60510385
cas_badval                   0
cas_hits                     0
cas_misses                   0
cmd_flush                    1
cmd_get                      53139
cmd_set                      229120
conn_yields                  0
connection_structures        34
curr_connections             13
curr_items                   54953
decr_hits                    0
decr_misses                  0
delete_hits                  0
delete_misses                0
evictions                    0
get_hits                     52784
get_misses                   355
incr_hits                    0
incr_misses                  0
limit_maxbytes               314572800
listen_disabled_num          0
pid                          7226
pointer_size                 32
reclaimed                    0
rusage_system                10.676667
rusage_user                  3.068191
threads                      4
time                         1320919080
total_connections            111
```

```
total_items                 229120
uptime                      7649
version                     1.4.5
```

2.5 *Warming the edb-icache Servers*

When Advanced Server starts, the primary and secondary caches are empty. When Advanced Server processes a client request, Advanced Server reads the required data from disk and stores a copy in each cache. You can improve server performance by *warming* (or pre-loading) the data into the memory cache before a client asks for it.

There are two advantages to warming the cache. Advanced Server will find data in the cache the first time it is requested by a client application, instead of waiting for it to be read from disk. Also, manually warming the cache with the data that your applications are most likely to need saves time by avoiding future random disk reads. If you don't warm the cache at startup, Postgres Plus Advanced Server performance may not reach full speed until the client applications happen to load commonly used data into the cache.

There are several ways to load pages to warm the Infinite Cache server nodes. You can:

- Use the `edb_icache_warm` utility to warm the caches from the command line.

- Use the `edb_icache_warm()` function from within edb-psql.

- Use the `edb_icache_warm()` function via scripts to warm the cache.

While it is not necessary to re-warm the cache after making changes to an existing cache configuration, re-warming the cache can improve performance by bringing the new configuration of cache servers up-to-date.

2.5.1 The edb_icache_warm() Function

The `edb_icache_warm()` function comes in two variations; the first variation warms not only the table, but any indexes associated with the table. If you use the second variation, you must make additional calls to warm any associated indexes.

The first form of the `edb_icache_warm()` function warms the given table and any associated indexes into the cache. The signature is:

```
edb_icache_warm(table_name)
```

You may specify `table_name` as a table name, OID, or `regclass` value.

```
    # edb-psql edb -c "select edb_icache_warm('accounts')"
```

When you call the first form of `edb_icache_warm()`, Advanced Server reads each page in the given table, compresses the page (if configured to do so), and then sends the compressed data to an Infinite Cache server. `edb_icache_warm()` also reads, compresses, and caches each page in each index defined for the given table.

The second form of the `edb_icache_warm()` function warms the pages that contain the specified range of bytes into the cache. The signature of the second form is:

```
edb_icache_warm(table-spec, startbyte, endbyte):
```

You must make subsequent calls to specify indexes separately when using this form of the `edb_icache_warm()` function.

```
# edb-psql edb -c "select edb_icache_warm('accounts', 1, 10000)"
```

The `edb_icache_warm()` function is typically called by a utility program (such as the `edb_icache_warm` utility) to spread the warming process among multiple processes that operate in parallel.

2.5.2 Using the edb_icache_warm Utility

You can use the `edb_icache_warm` command-line utility to load the cache servers with specified tables, allowing fast access to relevant data from the cache.

The syntax for `edb_icache_warm` is:

```
# edb_icache_warm -d database -t tablename
```

The only required parameter is *`tablename`*. *`tablename`* can be specified with or without the `-t` option. All other parameters are optional; if omitted, default values are inferred from Advanced Server environment variables.

The options for `edb_icache_warm` are:

Option	Variable	Description
-h	*hostname*	The name of the host running Advanced Server. Include this parameter if you are running Advanced Server on a remote host. The default value is PGHOST.
-p	*portname*	Port in use by Advanced Server. Default value is PGPORT.
-j	*process count*	Number of (parallel) processes used to warm the cache. The default value is 1.
-U	*username*	The Advanced Server username. Unless specified, this defaults to PGUSER.
-d	*database*	The name of database containing the tables to be warmed. Default value is PGDATABASE.
-t	*tablename*	Name of table to be warmed. The index for the table is also warmed. Required.

2.6 Retrieving Statistics from Infinite Cache

2.6.1 Using edb_icache_stats()

You can view Infinite Cache statistics by using the `edb_icache_stats()` function at the `edb-psql` command line (or any other query tool).

The `edb_icache_stats()` function returns a result set that reflects the state of an Infinite Cache node or nodes and the related usage statistics. The result set includes:

Statistic	Description
hostname	Host name (or IP address) of server
port	Port number at which edb-icache daemon is listening
state	Health of this server
write_failures	Number of write failures
bytes	Total number of bytes in use
bytes_read	Total number of bytes received by this server (from the network)
bytes_written	Total number of bytes sent by this server (to the network)
cmd_get	Cumulative number of read requests sent to this server
cmd_set	Cumulative number of write requests sent to this server
connection_structures	Number of connection structures allocated by the server
curr_connections	Number of open connections
curr_items	Number of items currently stored by the server
evictions	Number of valid items removed from cache to free memory for new items
get_hits	Number of read requests satisfied by this server
get_misses	Number of read requests not satisfied by this server
limit_maxbytes	Number of bytes allocated on this server for storage
pid	Process ID (on cache server)
pointer_size	Default pointer size on host OS (usually 32 or 64)
rusage_user	Accumulated user time for this process (seconds.microseconds)
rusage_system	Accumulated system time for this process (seconds.microseconds)
threads	Number of worker threads requested
total_time	Number of seconds since this server's base date (usually midnight, January 1, 1970, UTC)
total_connections	Total number of connections opened since the server started running
total_items	Total number of items stored by this server (cumulative)
uptime	Amount of time that server has been active
version	edb-icache version

You can use SQL queries to view Infinite Cache statistics. To view the server status of all Infinite Cache nodes:

```
SELECT hostname, port, state FROM edb_icache_stats()
```

```
 hostname       | port  | state
----------------+-------+---------
 192.168.23.85 | 11211 | UNHEALTHY
 192.168.23.85 | 11212 | ACTIVE
(2 rows)
```

Use the following command to view complete statistics (shown here using edb-psql's expanded display mode, `\x`) for a specified node:

```
SELECT * FROM edb_icache_stats() WHERE hostname = '192.168.23.85:11211'
```

```
-[RECORD 1]-----------+--------------
hostname              | 192.168.23.85
port                  | 11211
state                 | ACTIVE
write_failures        | 0
bytes                 | 225029460
bytes_read            | 225728252
bytes_written         | 192806774
cmd_get               | 23313
cmd_set               | 27088
connection_structures | 53
curr_connections      | 3
curr_items            | 27088
evictions             | 0
get_hits              | 23266
get_misses            | 47
limit_maxbytes        | 805306368
pid                   | 4240
pointer_size          | 32
rusage_user           | 0.481926
rusage_system         | 1.583759
threads               | 1
total_time            | 1242199782
total_connections     | 66
total_items           | 27088
uptime                | 714
version               | 1.2.6
```

2.6.2 edb_icache_server_list

The `edb_icache_server_list` view exposes information about the status and health of all Infinite Cache servers listed in the `edb_icache_servers` GUC. The `edb_icache_server_list` view is created using the `edb_icache stats()` API. The view exposes the following information for each server:

Statistic	Description
hostname	Host name (or IP address) of server
port	Port number at which edb-icache daemon is listening
state	Health of this server
write_failures	Number of write failures
total_memory	Number of bytes allocated to the cache on this server
memory_used	Number of bytes currently used by the cache
memory_free	Number of unused bytes remaining in the cache
hit_ratio	Percentage of cache hits

The `state` column will contain one of the following four values, reflecting the health of the given server:

Server State	Description
`Active`	The server is known to be up and running.
`Unhealthy`	An error occurred while interacting with the cache server. Postgres will attempt to re-establish the connection with the server.
`Offline`	Postgres can no longer contact the given server.
`Manual Offline`	You have taken the server offline with the edb_icache_server_enable() function.

Use the following `SELECT` statement to return the health of each node in the Infinite Cache server farm:

```
SELECT hostname, port, state FROM edb_icache_server_list
```

```
   hostname     | port  | state
----------------+-------+-------
 192.168.23.85 | 11211 | ACTIVE
 192.168.23.85 | 11212 | ACTIVE
(2 rows)
```

Use the following command to view complete details about a specific Infinite Cache node (shown here using edb-psql's `\x` expanded-view option):

```
SELECT * FROM edb_icache_server_list WHERE hostname = '192.168.23.85:11211'
```

```
-[RECORD 1]-----------+--------------
hostname               | 192.168.23.85
port                   | 11211
state                  | ACTIVE
write_failures         | 0
total_memory           | 805306368
memory_used            | 225029460
memory_free            | 580276908
hit_ratio              | 99.79
```

2.7 Retrieving Table Statistics

Advanced Server provides six system views that contain statistical information on a per-table basis. The views are:

- `pg_statio_all_tables`
- `pg_statio_sys_tables`
- `pg_statio_user_tables`
- `pg_statio_all_indexes`
- `pg_statio_sys_indexes`
- `pg_statio_user_indexes`

You can use standard SQL queries to view and compare the information stored in the views. The views contain information that will allow you to observe the effectiveness of the Advanced Server buffer cache and the icache servers.

2.7.1 pg_statio_all_tables

The pg_statio_all_tables view contains one row for each table in the database. The view contains the following information:

Column Name	Description
relid	The OID of the table.
schemaname	The name of the schema that the table resides in.
relname	The name of the table.
heap_blks_read	The number of heap blocks read.
heap_blks_hit	The number of heap blocks hit.
heap_blks_icache_hit	The number of heap blocks found on an icache server.
idx_blks_read	The number of index blocks read.
idx_blks_hit	The number of index blocks hit.
idx_blks_icache_hit	The number of index blocks found on an icache server.
toast_blks_read	The number of toast blocks read.
toast_blks_hit	The number of toast blocks hit.
toast_blks_icache_hit	The number of toast blocks found on an icache server.
tidx_blks_read	The number of index toast blocks read.
tidx_blks_hit	The number of index toast blocks hit.
tidx_blks_icache_hit	The number of index toast blocks found on an icache server.

You can execute a simple query to view performance statistics for a specific table:

```
SELECT * FROM pg_statio_all_tables WHERE relname=’jobhist’;

-[ RECORD 1 ]---------+---------
relid                 | 16402
schemaname            | public
relname               | jobhist
heap_blks_read        | 1
heap_blks_hit         | 51
heap_blks_icache_hit  | 0
idx_blks_read         | 2
idx_blks_hit          | 17
idx_blks_icache_hit   | 0
toast_blks_read       |
toast_blks_hit        |
toast_blks_icache_hit |
tidx_blks_read        |
tidx_blks_hit         |
tidx_blks_icache_hit  |
```

Or, you can view the statistics by activity level. The following example displays the statistics for the ten tables that have the greatest heap_blks_icache_hit activity:

```
SELECT * FROM pg_statio_all_tables ORDER BY heap_blks_icache_hit DESC LIMIT
10;

relid       schemaname                relname
   heap_blks_read    heap_blks_hit    heap_blks_icache_hit
   idx_blks_read     idx_blks_hit     idx_blks_icache_hit
   toast_blks_read   toast_blks_hit   toast_blks_icache_hit
   tidx_blks_read    tidx_blks_hit    tidx_blks_icache_hit
-------------------------------------------------------------------------------
```

```
16390       public                     pgbench_accounts
   264105            71150             81498
   13171             282541            18053

1259        pg_catalog                 pg_class
   22                2904              18
   14                3449              11

1249        pg_catalog                 pg_attribute
   49                1619              16
   17                2841              13

1255        pg_catalog                 pg_proc
   38                276               11
   33                682               16
   0                 0                 0
   0                 0                 0

2619        pg_catalog                 pg_statistic
   20                295               8
   4                 436               4
   0                 0                 0
   0                 0                 0

2617        pg_catalog                 pg_operator
   20                293               8
   19                791               10

2602        pg_catalog                 pg_amop
   10                721               6
   13                1154              13

2610        pg_catalog                 pg_index
   10               633                6
   8                 719               8

1247        pg_catalog                 pg_type
   17                235               5
   12                433               4

2615        pg_catalog                 pg_namespace
   4                260                4
   6                 330               4
   0                 0                 0
   0                 0                 0
 (10 rows)
```

2.7.2 pg_statio_sys_tables

The `pg_statio_sys_tables` view contains one row for each table in a system-defined schema. The statistical information included in this view is the same as for `pg_statio_all_tables`.

2.7.3 pg_statio_user_tables

The pg_statio_user_tables view contains one row for each table in a user-defined schema. The statistical information in this view is the same as for pg_statio_all_tables.

2.7.4 pg_statio_all_indexes

The pg_statio_all_indexes view contains one row for each index in the current database. The view contains the following information:

Column Name	Description
relid	The OID of the indexed table
indexrelid	The OID of the index.
schemaname	The name of the schema that the table resides in.
relname	The name of the table.
indexrelname	The name of the index
idx_blks_read	The number of index blocks read.
idx_blks_hit	The number of index blocks hit.
idx_blks_icache_hit	The number of index blocks found on an icache server.

You can execute a simple query to view performance statistics for the indexes on a specific table:

```
SELECT * FROM pg_statio_all_indexes WHERE relname=’pg_attribute’;

-[ RECORD 1 ]---------+---------
relid                 | 1249
indexrelid            | 2658
schemaname            | pg_catalog
relname               | pg_attribute
indexrelname          | pg_attribute_relid_attnam_index
idx_blks_read         | 10
idx_blks_hit          | 1200
idx_blks_icache_hit   | 0
-[ RECORD 2 ]---------+---------
relid                 | 1249
indexrelid            | 2659
schemaname            | pg_catalog
relname               | pg_attribute
indexrelname          | pg_attribute_relid_attnum_index
idx_blks_read         | 12
idx_blks_hit          | 3917
idx_blks_icache_hit   | 0
```

The result set from the query includes the statistical information for two indexes; the pg_attribute table has two indexes.

You can also view the statistics by activity level. The following example displays the statistics for the ten indexes that have the greatest idx_blks_icache_hit activity:

```
SELECT * FROM pg_statio_all_indexes ORDER BY idx_blks_icache_hit DESC LIMIT
10;

relid  indexrelid  schemaname   relname
indexrelname                 idx_blks_read  idx_blks_hit  idx_blks_icache_hit
------------------------------------------------------------------------------
16390  16401       public       pgbench_accounts
pgbench_accounts_pkey        13171          282541        18053

1249   2659        pg_catalog   pg_attribute
pg_attr_relid_attnum_index   14             2749          13

1255   2690        pg_catalog   proc
pg_proc_oid_index            16             580           12

1259   2663        pg_catalog   pg_class
pg_class_relname_nsp_index   10             2019          7

2602   2654        pg_catalog   pg_amop
pg_amop_opr_fam_index        7              453           7

2603   2655        pg_catalog   pg_amproc
pg_amproc_fam_proc_index     6              605           6

2617   2688        pg_catalog   pg_operator
pg_operator_oid_index        7              452           6

2602   2653        pg_catalog   pg_amop
pg_amop_fam_strat_index      6              701           6

2615   2684        pg_catalog   pg_namespace
pg_namespace_nspname_index   4              328           4

1262   2672        pg_catalog   pg_database
pg_database_oid_index        4              254           4
```

2.7.5 pg_statio_sys_indexes

The pg_statio_sys_indexes view contains one row for each index on the system tables. The statistical information in this view is the same as in pg_statio_all_indexes.

2.7.6 pg_statio_user_indexes

The pg_statio_user_indexes view contains one row for each index on a table that resides in a user-defined schema. The statistical information in this view is the same as in pg_statio_all_indexes.

2.8 edb_icache_server_enable()

You can use the `edb_icache_server_enable()` function to take the Infinite Cache server offline for maintenance or other planned downtime. The syntax is:

```
    void edb_icache_server_enable(host TEXT, port INTEGER, online BOOL)
```

`host` specifies the host that you want to disable. The host name may be specified by name or numeric address.

`port` specifies the port number that the Infinite Cache server is listening on.

`online` specifies the state of the Infinite Cache server. The value of *`online`* must be `true` or `false`.

To take a server offline, specify the host that you want to disable, the port number that the Infinite Cache server is listening on, and `false`. To bring the Infinite Cache server back online, specify the host name and port number, and pass a value of `true`.

The state of a server taken offline with the `edb_icache_server_enable()` function is `MANUAL OFFLINE`. Postgres Plus Advanced Server will not automatically reconnect to an Infinite Cache server that you have taken offline with `edb_icache_server_enable(..., false)`; you must bring the server back online by calling `edb_icache_server_enable(..., true)`.

2.9 Infinite Cache Log Entries

When you start Advanced Server, a message that includes Infinite Cache status, cache node count and cache node size is written to the server log. The following example shows the server log for an active Infinite Cache installation with two 750 MB cache servers:

```
** EnterpriseDB Dynamic Tuning Agent***************************************
*       System Utilization: 66 %                                          *
*       Autovacuum Naptime: 60   Seconds                                  *
*       Infinite Cache: on                                                 *
*       Infinite Cache Servers: 2                                          *
*       Infinite Cache Size: 1.500  GB                                     *
***************************************************************************
```

2.10 Allocating Memory to the Cache Servers

As mentioned earlier in this document, each computer imposes a limit on the amount of *physical* memory that you can install. However, modern operating systems typically simulate a larger *address space* so that programs can transparently access more memory than is actually installed. This "virtual memory" allows a computer to run multiple programs which may simultaneously require more memory than is physically available. For example, you may run an e-mail client, a web browser, and a database server which each require 1GB of memory on a machine that contains only 2GB of physical RAM. When the operating system runs out of physical memory, it starts swapping bits and pieces of the currently running programs to disk to make room to satisfy your current demand for memory.

This can bring your system to a grinding halt.

Since the primary goal of Infinite Cache is to improve performance by limiting disk I/O, you should avoid dedicating so much memory to Infinite Cache that the operating system must start swapping data to disk. If the operating system begins to swap to disk, you lose the benefits offered by Infinite Cache.

The overall demand for physical memory can vary throughout the day; if the server is frequently idle, you may never encounter swapping. If you have dedicated a large portion of physical memory to the cache, and system usage increases, the operating system may start swapping. To get the best performance and avoid disk swapping, dedicate a server node to Infinite Cache so other applications on that computer will not compete for physical memory.

3 Dynatune

Postgres Plus Advanced Server supports dynamic tuning of the database server to make the optimal usage of the system resources available on the host machine on which it is installed. The two parameters that control this functionality are located in the `postgresql.conf` file. These parameters are:

- `edb_dynatune`
- `edb_dynatune_profile`

3.1.1 edb_dynatune

`edb_dynatune` determines how much of the host system's resources are to be used by the database server based upon the host machine's total available resources and the intended usage of the host machine.

When Postgres Plus Advanced Server is initially installed, the `edb_dynatune` parameter is set in accordance with the selected usage of the host machine on which it was installed - i.e., development machine, mixed use machine, or dedicated server. For most purposes, there is no need for the database administrator to adjust the various configuration parameters in the `postgresql.conf` file in order to improve performance.

You can change the value of the `edb_dynatune` parameter after the initial installation of Postgres Plus Advanced Server by editing the `postgresql.conf` file. The postmaster must be restarted in order for the new configuration to take effect.

The `edb_dynatune` parameter can be set to any integer value between 0 and 100, inclusive. A value of 0, turns off the dynamic tuning feature thereby leaving the database server resource usage totally under the control of the other configuration parameters in the `postgresql.conf` file.

A low non-zero, value (e.g., 1 - 33) dedicates the least amount of the host machine's resources to the database server. This setting would be used for a development machine where many other applications are being used.

A value in the range of 34 - 66 dedicates a moderate amount of resources to the database server. This setting might be used for a dedicated application server that may have a fixed number of other applications running on the same machine as Postgres Plus Advanced Server.

The highest values (e.g., 67 - 100) dedicate most of the server's resources to the database server. This setting would be used for a host machine that is totally dedicated to running Postgres Plus Advanced Server.

Once a value of edb_dynatune is selected, database server performance can be further fine-tuned by adjusting the other configuration parameters in the postgresql.conf file. Any adjusted setting overrides the corresponding value chosen by edb_dynatune. You can change the value of a parameter by un-commenting the configuration parameter, specifying the desired value, and restarting the database server.

3.1.2 edb_dynatune_profile

The edb_dynatune_profile parameter is used to control tuning aspects based upon the expected workload profile on the database server. This parameter takes effect upon startup of the database server.

The possible values for edb_dynatune_profile are:

Value	Usage
oltp	Recommended when the database server is processing heavy online transaction processing workloads.
reporting	Recommended for database servers used for heavy data reporting.
mixed	Recommended for servers that provide a mix of transaction processing and data reporting.

4 Dynamic Runtime Instrumentation Tools Architecture (DRITA)

Note: *This information is also included in the Oracle® Compatibility Developer's Guide.*

The Dynamic Runtime Instrumentation Tools Architecture (DRITA) allows a DBA to query catalog views to determine the *wait events* that affect the performance of individual sessions or the system as a whole. DRITA records the number of times each event occurs as well as the time spent waiting; you can use this information to diagnose performance problems.

DRITA compares *snapshots* to evaluate the performance of a system. A snapshot is a saved set of system performance data at a given point in time. Each snapshot is identified by a unique ID number; you can use snapshot ID numbers with DRITA reporting functions to return system performance statistics.

DRITA consumes minimal system resources.

4.1.1 Initialization Parameters

DRITA includes a configuration parameter, `timed_statistics`, to control the collection of timing data. This is a dynamic parameter that can be set in the `postgresql.conf` file or while a session is in progress. The valid values are `TRUE` or `FALSE`; the default value is `FALSE`.

4.1.2 Setting up and Using DRITA

First, take a beginning snapshot. The beginning snapshot will be compared to a later snapshot to gauge system performance. To take a beginning snapshot:

```
SELECT * from edbsnap()
```

Then, run the workload that you would like to evaluate; when the workload has completed (or at a strategic point during the workload), take an ending snapshot:

```
SELECT * from edbsnap()
```

4.2 DRITA Functions

4.2.1 get_snaps()

The `get_snaps()` function returns a list of the current snapshots. The signature is:

```
get_snaps()
```

The following example demonstrates using the `get_snaps()` function to display a list of snapshots:

```
edb=# SELECT * FROM get_snaps();
          get_snaps
------------------------------
 1  11-FEB-10 10:41:05.668852
 2  11-FEB-10 10:42:27.26154
 3  11-FEB-10 10:45:48.999992
 4  11-FEB-10 11:01:58.345163
 5  11-FEB-10 11:05:14.092683
 6  11-FEB-10 11:06:33.151002
 7  11-FEB-10 11:11:16.405664
 8  11-FEB-10 11:13:29.458405
 9  11-FEB-10 11:23:57.595916
 10 11-FEB-10 11:29:02.214014
 11 11-FEB-10 11:31:44.244038
(11 rows)
```

The first column in the list of snapshots contains the session identifier; the DRITA functions use the session identifier to operate on a specific snapshot.

4.2.2 sys_rpt()

The `sys_rpt()` function returns system wait information. The signature is:

```
sys_rpt(beginning_id, ending_id, top_n)
```

Parameters

`beginning_id`

`beginning_id` is an integer value that represents the beginning session identifier.

`ending_id`

`ending_id` is an integer value that represents the ending session identifier .

`top_n`

`top_n` represents the number of rows to return

This example demonstrates a call to the `sys_rpt()` function:

```
edb=# SELECT * FROM sys_rpt(9, 10, 10);
```

```
                                    sys_rpt
-----------------------------------------------------------------------------
 WAIT NAME                               COUNT      WAIT TIME        % WAIT
 ---------------------------------------------------------------------------
 wal write                               21250      104.723772       36.31
 db file read                            121407     72.143274        25.01
 wal flush                               84185      51.652495        17.91
 wal file sync                           712        29.482206        10.22
 infinitecache write                     84178      15.814444        5.48
 db file write                           84177      14.447718        5.01
 infinitecache read                      672        0.098691         0.03
 db file extend                          190        0.040386         0.01
 query plan                              52         0.024400         0.01
 wal insert lock acquire                 4          0.000837         0.00
(12 rows)
```

4.2.3 sess_rpt()

The `sess_rpt()` function returns session wait information. The signature is:

```
sess_rpt(beginning_id, ending_id, top_n)
```

Parameters

beginning_id

beginning_id is an integer value that represents the beginning session identifier.

ending_id

ending_id is an integer value that represents the ending session identifier .

top_n

top_n represents the number of rows to return

The following example demonstrates a call to the `sess_rpt()` function:

```
SELECT * FROM sess_rpt(18, 19, 10);

                                               sess_rpt
-----------------------------------------------------------------------------
ID    USER       WAIT NAME             COUNT TIME(ms)    %WAIT SES   %WAIT ALL
 ----------------------------------------------------------------------------

 17373 enterprise db file read           30   0.175713   85.24       85.24
 17373 enterprise query plan             18   0.014930   7.24        7.24
 17373 enterprise wal flush              6    0.004067   1.97        1.97
 17373 enterprise wal write              1    0.004063   1.97        1.97
 17373 enterprise wal file sync          1    0.003664   1.78        1.78
 17373 enterprise infinitecache read     38   0.003076   1.49        1.49
 17373 enterprise infinitecache write    5    0.000548   0.27        0.27
 17373 enterprise db file extend         190  0.04.386   0.03        0.03
 17373 enterprise db file write          5    0.000082   0.04        0.04
 17373 enterprise wal write lock acquire 0    0.000000   0.00        0.00
 17373 enterprise bgwriter comm lock ac  0    0.000000   0.00        0.00
(13 rows)
```

4.2.4 sessid_rpt()

The `sessid_rpt()` function returns session ID information for a specified backend. The signature is:

```
sessid_rpt(beginning_id, ending_id, backend_id)
```

Parameters

beginning_id

beginning_id is an integer value that represents the beginning session identifier.

ending_id

ending_id is an integer value that represents the ending session identifier .

backend_id

backend_id is an integer value that represents the backend identifier .

The following code sample demonstrates a call to `sessid_rpt()`:

```
SELECT * FROM sessid_rpt(18, 19, 17373);

                                sessid_rpt
-----------------------------------------------------------------------------
 ID    USER       WAIT NAME            COUNT TIME(ms)  %WAIT SES   %WAIT ALL
 ----------------------------------------------------------------------------
 17373 enterprise db file read           30   0.175713  85.24       85.24
 17373 enterprise query plan             18   0.014930  7.24        7.24
 17373 enterprise wal flush              6    0.004067  1.97        1.97
 17373 enterprise wal write              1    0.004063  1.97        1.97
 17373 enterprise wal file sync          1    0.003664  1.78        1.78
 17373 enterprise infinitecache read     38   0.003076  1.49        1.49
 17373 enterprise infinitecache write    5    0.000548  0.27        0.27
 17373 enterprise db file extend         190  0.040386  0.03        0.03
 17373 enterprise db file write          5    0.000082  0.04        0.04
 17373 enterprise wal write lock acquire 0    0.000000  0.00        0.00
 17373 enterprise bgwriter comm lock ac  0    0.000000  0.00        0.00
(13 rows)
```

4.2.5 sesshist_rpt()

The `sesshist_rpt()` function returns session wait information for a specified backend. The signature is:

```
sesshist_rpt(snapshot_id, session_id)
```

Parameters

snapshot_id

snapshot_id is an integer value that identifies the snapshot.

session_id

session_id is an integer value that represents the session.

The following example demonstrates a call to the `sesshist_rpt()` function:

```
edb=# SELECT * FROM sesshist_rpt (9, 5531);
                              sesshist_rpt
-----------------------------------------------------------------------------
 ID    USER       SEQ  WAIT NAME
   ELAPSED(ms)   File  Name                 # of Blk   Sum of Blks
 -----------------------------------------------------------------------------
 5531 enterprise 1     db file read
   18546        14309  session_waits_pk     1          1
 5531 enterprise 2     infinitecache read
   125          14309  session_waits_pk     1          1
 5531 enterprise 3     db file read
   376          14304  edb$session_waits    0          1
 5531 enterprise 4     infinitecache read
   166          14304  edb$session_waits    0          1
 5531 enterprise 5     db file read
   7978         1260   pg_authid            0          1
 5531 enterprise 6     infinitecache read
   154          1260   pg_authid            0          1
 5531 enterprise 7     db file read
   628          14302  system_waits_pk      1          1
 5531 enterprise 8     infinitecache read
   463          14302  system_waits_pk      1          1
 5531 enterprise 9     db file read
   3446         14297  edb$system_waits     0          1
 5531 enterprise 10    infinitecache read
   187          14297  edb$system_waits     0          1
 5531 enterprise 11    db file read
   14750        14295  snap_pk              1          1
 5531 enterprise 12    infinitecache read
   416          14295  snap_pk              1          1
 5531 enterprise 13    db file read
   7139         14290  edb$snap             0          1
 5531 enterprise 14    infinitecache read
   158          14290  edb$snap             0          1
 5531 enterprise 15    db file read
   27287        14288  snapshot_num_seq     0          1
 5531 enterprise 16    infinitecache read
```

```
   180          14288  snapshot_num_seq     0          1
 5531 enterprise 17    query plan
   26           0      N/A                  0          0
 5531 enterprise 18    db file read
   84552        16396  pgbench_accounts     4358       1
 5531 enterprise 19    infinitecache read
   226          16396  pgbench_accounts     4358       1
 5531 enterprise 20    db file read
   334838       16401  pgbench_accounts_pke 7792       1
 5531 enterprise 21    infinitecache read
   213          16401  pgbench_accounts_pke 7792       1
 5531 enterprise 22    db file read
   52619        16396  pgbench_accounts     24829      1
 5531 enterprise 23    infinitecache read
   210          16396  pgbench_accounts     24829      1
 5531 enterprise 24    infinitecache read
   216          16401  pgbench_accounts_pke 13460      1
 5531 enterprise 25    db file read
   13925        16396  pgbench_accounts     27695      1
(27 rows)
```

4.2.6 purgesnap()

The `purgesnap()` function purges a range of snapshots from the snapshot tables. The signature is:

`purgesnap(`*`beginning_id`*`, `*`ending_id`*`)`

Parameters

beginning_id

beginning_id is an integer value that represents the beginning session identifier.

ending_id

ending_id is an integer value that represents the ending session identifier .

`purgesnap()` removes all snapshots between *beginning_id* and *ending_id* (inclusive):

```
SELECT * FROM purgesnap(6, 9);

             purgesnap
------------------------------------
 Snapshots in range 6 to 9 deleted.
(1 row)
```

A call to the `get_snaps()` function after executing the example shows that snapshots `6` through `9` have been purged from the snapshot tables:

```
edb=# SELECT * FROM get_snaps();
          get_snaps
------------------------------
 1  11-FEB-10 10:41:05.668852
 2  11-FEB-10 10:42:27.26154
 3  11-FEB-10 10:45:48.999992
 4  11-FEB-10 11:01:58.345163
 5  11-FEB-10 11:05:14.092683
 10 11-FEB-10 11:29:02.214014
 11 11-FEB-10 11:31:44.244038
(7 rows)
```

4.2.7 truncsnap()

Use the `truncsnap()` function to delete all records from the snapshot table. The signature is:

```
truncsnap()
```

For example:

```
SELECT * FROM truncsnap();

      truncsnap
----------------------
 Snapshots truncated.
(1 row)
```

A call to the `get_snaps()` function after calling the `truncsnap()` function shows that all records have been removed from the snapshot tables:

```
SELECT * FROM get_snaps();
 get_snaps
-----------
(0 rows)
```

4.3 Simulating Statspack AWR Reports

The functions described in this section return information comparable to the information contained in an Oracle Statspack/AWR (Automatic Workload Repository) report. When taking a snapshot, performance data from system catalog tables is saved into history tables. The reporting functions listed below report on the differences between two given snapshots.

- `stat_db_rpt()`
- `stat_tables_rpt()`
- `statio_tables_rpt()`
- `stat_indexes_rpt()`
- `statio_indexes_rpt()`

The reporting functions can be executed individually or you can execute all five functions by calling the `edbreport()` function.

4.3.1 edbreport()

The `edbreport()` function includes data from the other reporting functions, plus additional system information. The signature is:

```
edbreport(beginning_id, ending_id)
```

Parameters

`beginning_id`

`beginning_id` is an integer value that represents the beginning session identifier.

`ending_id`

`ending_id` is an integer value that represents the ending session identifier.

The following code sample demonstrates a call to the `edbreport()` function:

```
edb=# SELECT * FROM edbreport(9, 10);

edbreport
------------------------------------------------------------------------------
---------------------------------------------------------------------------
    EnterpriseDB Report for database edb        23-AUG-12
 Version: EnterpriseDB 9.2.0.0 on i686-pc-linux-gnu, compiled by gcc (GCC)
4.1.2 20080704 (Red Hat 4.1.2-52), 32-bit

      Begin snapshot: 9 at 23-AUG-12 13:45:07.165123
      End snapshot:   10 at 23-AUG-12 13:45:35.653036
 Size of database edb is 155 MB
      Tablespace: pg_default Size: 179 MB Owner: enterprisedb
      Tablespace: pg_global Size: 435 kB Owner: enterprisedb

 Schema: pg_toast_temp_1                 Size: 0 bytes         Owner:
enterprisedb
 Schema: public                          Size: 0 bytes         Owner:
enterprisedb
 Schema: enterprisedb                    Size: 143 MB          Owner:
enterprisedb
 Schema: pgagent                         Size: 192 kB          Owner:
enterprisedb
 Schema: dbms_job_procedure              Size: 0 bytes         Owner:
enterprisedb

                Top 10 Relations by pages

 TABLE                                         RELPAGES
 ----------------------------------------------------------------------------
------
 pgbench_accounts                               15874
 pg_proc                                        102
 edb$statio_all_indexes                         73
 edb$stat_all_indexes                           73
 pg_attribute                                   67
 pg_depend                                      58
 edb$statio_all_tables                          49
 edb$stat_all_tables                            47
 pgbench_tellers                                37
 pg_description                                 32

                Top 10 Indexes by pages

 INDEX                                         RELPAGES
 ----------------------------------------------------------------------------
------
 pgbench_accounts_pkey                          2198
 pg_depend_depender_index                       32
 pg_depend_reference_index                      31
 pg_proc_proname_args_nsp_index                 30
 pg_attribute_relid_attnam_index                23
 pg_attribute_relid_attnum_index                17
 pg_description_o_c_o_index                     15
 edb$statio_idx_pk                              11
 edb$stat_idx_pk                                11
 pg_proc_oid_index                              9

                Top 10 Relations by DML

 SCHEMA          RELATION                                    UPDATES
DELETES   INSERTS
 ----------------------------------------------------------------------------
----------------------
 enterprisedb    pgbench_accounts                            10400     0
1000000
 enterprisedb    pgbench_tellers                             10400     0
100
 enterprisedb    pgbench_branches                            10400     0
10
 enterprisedb    pgbench_history                             0         0
10400
 pgagent         pga_jobclass                                0         0
6
```

```
 pgagent         pga_exception                                 0         0
0
 pgagent         pga_job                                       0         0
0
 pgagent         pga_jobagent                                  0         0
0
 pgagent         pga_joblog                                    0         0
0
 pgagent         pga_jobstep                                   0         0
0

   DATA from pg_stat_database

 DATABASE   NUMBACKENDS  XACT COMMIT  XACT ROLLBACK   BLKS READ  BLKS HIT
BLKS ICACHE HIT      HIT RATIO  ICACHE HIT RATIO
 ---------------------------------------------------------------------------
-------------------------------------------------
 edb        0            142          0               78         10446      0
99.26      0.00

   DATA from pg_buffercache not included because pg_buffercache is not
installed

   DATA from pg_stat_all_tables ordered by seq scan

 SCHEMA               RELATION                      SEQ SCAN   REL TUP READ
IDX SCAN   IDX TUP READ INS    UPD    DEL
 ---------------------------------------------------------------------------
------------------------------------------
 pg_catalog           pg_class                      16         7162
546        319          0      1      0
 pg_catalog           pg_am                         13         13
0          0            0      0      0
 pg_catalog           pg_database                   4          16
42         42           0      0      0
 pg_catalog           pg_index                      4          660
145        149          0      0      0
 pg_catalog           pg_namespace                  4          100
49         49           0      0      0
 sys                  edb$snap                      1          9
0          0            1      0      0
 pg_catalog           pg_authid                     1          1
25         25           0      0      0
 sys                  edb$session_wait_history      0          0
0          0            50     0      0
 sys                  edb$session_waits             0          0
0          0            2      0      0
 sys                  edb$stat_all_indexes          0          0
0          0            165    0      0

   DATA from pg_stat_all_tables ordered by rel tup read

 SCHEMA               RELATION                      SEQ SCAN   REL TUP READ
IDX SCAN   IDX TUP READ INS    UPD    DEL
 ---------------------------------------------------------------------------
------------------------------------------
 pg_catalog           pg_class                      16         7162
546        319          0      1      0
 pg_catalog           pg_index                      4          660
145        149          0      0      0
 pg_catalog           pg_namespace                  4          100
49         49           0      0      0
```

```
 pg_catalog          pg_database                    4         16
42         42           0     0      0
 pg_catalog          pg_am                          13        13
0          0            0     0      0
 sys                 edb$snap                       1         9
0          0            1     0      0
 pg_catalog          pg_authid                      1         1
25         25           0     0      0
 sys                 edb$session_wait_history       0         0
0          0            50    0      0
 sys                 edb$session_waits              0         0
0          0            2     0      0
 sys                 edb$stat_all_indexes           0         0
0          0            165   0      0

   DATA from pg_statio_all_tables

 SCHEMA              RELATION             HEAP     HEAP     HEAP    IDX
IDX      IDX      TOAST   TOAST    TOAST    TIDX     TIDX     TIDX
                                          READ     HIT      ICACHE  READ
HIT      ICACHE   READ    HIT      ICACHE   READ     HIT      ICACHE
                                                            HIT
HIT                       HIT                        HIT
 ---------------------------------------------------------------------------
-----------------------------------------------------------------------
 pg_catalog          pg_class             0        539      0       0
1117     0        0       0        0        0        0        0
 enterprisedb        pgbench_accounts     48       485      0       1
778      0        0       0        0        0        0        0
 pg_catalog          pg_attribute         0        447      0       0
867      0        0       0        0        0        0        0
 enterprisedb        pgbench_branches     0        439      0       0
114      0        0       0        0        0        0        0
 enterprisedb        pgbench_tellers      0        357      0       0
112      0        0       0        0        0        0        0
 pg_catalog          pg_statistic         1        293      0       0
441      0        0       0        0        0        0        0
 pg_catalog          pg_index             0        159      0       0
171      0        0       0        0        0        0        0
 pg_catalog          pg_opclass           0        145      0       0
68       0        0       0        0        0        0        0
 pg_catalog          pg_proc              0        135      0       0
294      0        0       0        0        0        0        0
 pg_catalog          pg_type              0        103      0       0
322      0        0       0        0        0        0        0

   DATA from pg_stat_all_indexes

 SCHEMA              RELATION                  INDEX
IDX SCAN   IDX TUP READ IDX TUP FETCH
 ---------------------------------------------------------------------------
--------------------------------------------
 pg_catalog          pg_attribute
pg_attribute_relid_attnum_index    427        907          907
 pg_catalog          pg_class                  pg_class_relname_nsp_index
289        62           62
 pg_catalog          pg_class                  pg_class_oid_index
257        257          257
 pg_catalog          pg_statistic
pg_statistic_relid_att_inh_index   207        196          196
 enterprisedb        pgbench_accounts          pgbench_accounts_pkey
200        255          200
```

```
 pg_catalog           pg_cast                   pg_cast_source_target_index
199        50          50
 pg_catalog           pg_proc                   pg_proc_oid_index
116        116         116
 pg_catalog           edb_partition             edb_partition_partrelid_index
112        0           0
 pg_catalog           edb_policy                edb_policy_object_name_index
112        0           0
 enterprisedb         pgbench_branches          pgbench_branches_pkey
101        110         0

   DATA from pg_statio_all_indexes

 SCHEMA               RELATION                  INDEX
IDX BLKS READ   IDX BLKS HIT    IDX BLKS ICACHE HIT
 ---------------------------------------------------------------------------
-----------------------------------------------------------
 pg_catalog           pg_attribute
pg_attribute_relid_attnum_index     0               867             0
 enterprisedb         pgbench_accounts          pgbench_accounts_pkey
1               778             0
 pg_catalog           pg_class                  pg_class_relname_nsp_index
0               590             0
 pg_catalog           pg_class                  pg_class_oid_index
0               527             0
 pg_catalog           pg_statistic
pg_statistic_relid_att_inh_index    0               441             0
 sys                  edb$stat_all_indexes      edb$stat_idx_pk
1               332             0
 sys                  edb$statio_all_indexes    edb$statio_idx_pk
1               332             0
 pg_catalog           pg_proc                   pg_proc_oid_index
0               244             0
 sys                  edb$stat_all_tables       edb$stat_tab_pk
0               241             0
 sys                  edb$statio_all_tables     edb$statio_tab_pk
0               241             0

    System Wait Information

 WAIT NAME                                COUNT      WAIT TIME       % WAIT
 --------------------------------------------------------------------------
 query plan                               0          0.000407        100.00
 db file read                             0          0.000000        0.00

    Database Parameters from postgresql.conf

 PARAMETER                         SETTING
CONTEXT     MINVAL       MAXVAL
 ---------------------------------------------------------------------------
---------------------------------------------
 allow_system_table_mods            off
postmaster
 application_name                   psql
user
 archive_command                    (disabled)
sighup
 archive_mode                       off
postmaster
 archive_timeout                    0
sighup      0           2147483647
 array_nulls                        on
user
```

```
 authentication_timeout             60
sighup      1            600
 autovacuum                         on
sighup
 autovacuum_analyze_scale_factor    0.1
sighup      0            100
 autovacuum_analyze_threshold       50
sighup      0            2147483647
 autovacuum_freeze_max_age          200000000
postmaster  100000000    2000000000
 autovacuum_max_workers             3
postmaster  1            8388607
 autovacuum_naptime                 60
sighup      1            2147483
 autovacuum_vacuum_cost_delay       20
sighup      -1           100
 autovacuum_vacuum_cost_limit       -1
sighup      -1           10000
 autovacuum_vacuum_scale_factor     0.2
sighup      0            100
 autovacuum_vacuum_threshold        50
sighup      0            2147483647
 backslash_quote                    safe_encoding
user
 bgwriter_delay                     200
sighup      10           10000
 bgwriter_lru_maxpages              100
sighup      0            1000
 bgwriter_lru_multiplier            2
sighup      0            10
 block_size                         8192
internal    8192         8192
 bonjour                            off
postmaster
 bonjour_name
postmaster
 bytea_output                       hex
user
 check_function_bodies              on
user
 checkpoint_completion_target       0.5
sighup      0            1
 checkpoint_segments                64
sighup      1            2147483647
 checkpoint_timeout                 300
sighup      30           3600
 checkpoint_warning                 30
sighup      0            2147483647
 client_encoding                    UTF8
user
 client_min_messages                notice
user
 commit_delay                       0
user        0            100000
 commit_siblings                    5
user        0            1000
 config_file                        /opt/PostgresPlus/9.2AS/data/postgresql.
postmaster
 constraint_exclusion               partition
user
 cpu_index_tuple_cost               0.005
user        0            1.79769e+308
```

```
 cpu_operator_cost                  0.0025
user        0            1.79769e+308
 cpu_tuple_cost                     0.01
user        0            1.79769e+308
 cursor_tuple_fraction              0.1
user        0            1
 data_directory                     /opt/PostgresPlus/9.2AS/data
postmaster
 DateStyle                          Redwood, SHOW_TIME
user
 db_dialect                         redwood
user
 dbms_alert.max_alerts              100
postmaster  0            500
 dbms_pipe.total_message_buffer     30
postmaster  30           262144
 db_user_namespace                  off
sighup
 deadlock_timeout                   1000
superuser   1            2147483647
 debug_assertions                   off
user
 debug_pretty_print                 on
user
 debug_print_parse                  off
user
 debug_print_plan                   off
user
 debug_print_rewritten              off
user
 default_heap_fillfactor            100
user        10           100
 default_statistics_target          100
user        1            10000
 default_tablespace
user
 default_text_search_config         pg_catalog.english
user
 default_transaction_deferrable     off
user
 default_transaction_isolation      read committed
user
 default_transaction_read_only      off
user
 default_with_oids                  off
user
 default_with_rowids                off
user
 dynamic_library_path               $libdir
superuser
 edb_audit
sighup
 edb_audit_connect                  failed
sighup
 edb_audit_directory                edb_audit
sighup
 edb_audit_disconnect               none
sighup
 edb_audit_filename                 audit-%Y%m%d_%H%M%S
sighup
 edb_audit_rotation_day             every
sighup
```

```
 edb_audit_rotation_seconds         0
sighup      0            2147483647
 edb_audit_rotation_size            0
sighup      0            5000
 edb_audit_statement                ddl, error
sighup
 edb_connectby_order                on
user
 edb_dynatune                       66
postmaster  0            100
 edb_dynatune_profile               oltp
postmaster
 edb_enable_icache                  off
postmaster
 edb_icache_compression_level       6
superuser   0            9
 edb_icache_servers
sighup
 edb_redwood_date                   on
user
 edb_redwood_strings                on
user
 edb_stmt_level_tx                  off
user
 effective_cache_size               34277
user        1            2147483647
 effective_io_concurrency           1
user        0            1000
 enable_bitmapscan                  on
user
 enable_hashagg                     on
user
 enable_hashjoin                    on
user
 enable_hints                       on
user
 enable_indexonlyscan               on
user
 enable_indexscan                   on
user
 enable_material                    on
user
 enable_mergejoin                   on
user
 enable_nestloop                    on
user
 enable_seqscan                     on
user
 enable_sort                        on
user
 enable_tidscan                     on
user
 escape_string_warning              on
user
 event_source                       PostgreSQL
postmaster
 exit_on_error                      off
user
 external_pid_file
postmaster
 extra_float_digits                 0
user        -15          3
```

```
 from_collapse_limit                8
user        1            2147483647
 fsync                              on
sighup
 full_page_writes                   on
sighup
 geqo                               on
user
 geqo_effort                        5
user        1            10
 geqo_generations                   0
user        0            2147483647
 geqo_pool_size                     0
user        0            2147483647
 geqo_seed                          0
user        0            1
 geqo_selection_bias                2
user        1.5          2
 geqo_threshold                     12
user        2            2147483647
 gin_fuzzy_search_limit             0
user        0            2147483647
 hba_file                           /opt/PostgresPlus/9.2AS/data/pg_hba.conf
postmaster
 hot_standby                        off
postmaster
 hot_standby_feedback               off
sighup
 ident_file                         /opt/PostgresPlus/9.2AS/data/pg_ident.co
postmaster
 ignore_system_indexes              off
backend
 integer_datetimes                  on
internal
 IntervalStyle                      postgres
user
 join_collapse_limit                8
user        1            2147483647
 krb_caseins_users                  off
sighup
 krb_server_keyfile                 FILE:/home/edb/AS92/edb-postgres/inst/et
sighup
 krb_srvname                        postgres
sighup
 lc_collate                         en_US.UTF-8
internal
 lc_ctype                           en_US.UTF-8
internal
 lc_messages                        en_US.UTF-8
superuser
 lc_monetary                        en_US.UTF-8
user
 lc_numeric                         en_US.UTF-8
user
 lc_time                            en_US.UTF-8
user
 listen_addresses                   *
postmaster
 local_preload_libraries
backend
 lo_compat_privileges               off
superuser
```

```
 log_autovacuum_min_duration       -1
sighup      -1           2147483647
 log_checkpoints                   off
sighup
 log_connections                   off
backend
 log_destination                   stderr
sighup
 log_directory                     pg_log
sighup
 log_disconnections                off
backend
 log_duration                      off
superuser
 log_error_verbosity               default
superuser
 log_executor_stats                off
superuser
 log_file_mode                     0600
sighup      0            511
 log_filename                      enterprisedb-%Y-%m-%d_%H%M%S.log
sighup
 logging_collector                 on
postmaster
 log_hostname                      off
sighup
 log_line_prefix                   %t
sighup
 log_lock_waits                    off
superuser
 log_min_duration_statement        -1
superuser   -1           2147483647
 log_min_error_statement           error
superuser
 log_min_messages                  warning
superuser
 log_parser_stats                  off
superuser
 log_planner_stats                 off
superuser
 log_rotation_age                  1440
sighup      0            35791394
 log_rotation_size                 10240
sighup      0            2097151
 log_statement                     none
superuser
 log_statement_stats               off
superuser
 log_temp_files                    -1
superuser   -1           2147483647
 log_timezone                      US/Eastern
sighup
 log_truncate_on_rotation          off
sighup
 maintenance_work_mem              36871
user        1024         2097151
 max_connections                   100
postmaster  1            8388607
 max_files_per_process             1000
postmaster  25           2147483647
 max_function_args                 256
internal    256          256
```

```
 max_identifier_length               63
internal    63           63
 max_index_keys                      32
internal    32           32
 max_locks_per_transaction           64
postmaster  10           2147483647
 max_pred_locks_per_transaction      64
postmaster  10           2147483647
 max_prepared_transactions           0
postmaster  0            8388607
 max_stack_depth                     2048
superuser   100          2097151
 max_standby_archive_delay           30000
sighup      -1           2147483647
 max_standby_streaming_delay         30000
sighup      -1           2147483647
 max_wal_senders                     0
postmaster  0            8388607
 odbc_lib_path
postmaster
 optimizer_mode                      choose
user
 oracle_home
postmaster
 password_encryption                 on
user
 plpgsql.variable_conflict           error
superuser
 port                                5444
postmaster  1            65535
 post_auth_delay                     0
backend     0            2147483647
 pre_auth_delay                      0
sighup      0            60
 qreplace_function
superuser
 quote_all_identifiers               off
user
 random_page_cost                    4
user        0            1.79769e+308
 replication_timeout                 60000
sighup      0            2147483647
 restart_after_crash                 on
sighup
 search_path                         "$user",public
user
 segment_size                        131072
internal    131072       131072
 seq_page_cost                       1
user        0            1.79769e+308
 server_encoding                     UTF8
internal
 server_version                      9.2.0.0
internal
 server_version_num                  90200
internal    90200        90200
 session_replication_role            origin
superuser
 shared_buffers                      23540
postmaster  16           1073741823
 shared_preload_libraries            $libdir/dbms_pipe,$libdir/edb_gen,$libdi
postmaster
```

```
 sql_inheritance                    on
user
 ssl                                off
postmaster
 ssl_ca_file
postmaster
 ssl_cert_file                      server.crt
postmaster
 ssl_ciphers                        ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH
postmaster
 ssl_crl_file
postmaster
 ssl_key_file                       server.key
postmaster
 ssl_renegotiation_limit            524288
user        0            2097151
 standard_conforming_strings        on
user
 statement_timeout                  0
user        0            2147483647
 stats_temp_directory               pg_stat_tmp
sighup
 superuser_reserved_connections     3
postmaster  0            8388607
 synchronize_seqscans               on
user
 synchronous_commit                 on
user
 synchronous_standby_names
sighup
 syslog_facility                    local0
sighup
 syslog_ident                       postgres
sighup
 tcp_keepalives_count               0
user        0            2147483647
 tcp_keepalives_idle                0
user        0            2147483647
 tcp_keepalives_interval            0
user        0            2147483647
 temp_buffers                       1024
user        100          1073741823
 temp_file_limit                    -1
superuser   -1           2147483647
 temp_tablespaces
user
 timed_statistics                   on
user
 TimeZone                           US/Eastern
user
 timezone_abbreviations             Default
user
 trace_hints                        off
user
 trace_notify                       off
user
 trace_recovery_messages            log
sighup
 trace_sort                         off
user
 track_activities                   on
superuser
```

```
 track_activity_query_size          1024
postmaster  100          102400
 track_counts                       on
superuser
 track_functions                    none
superuser
 track_io_timing                    off
superuser
 transaction_deferrable             off
user
 transaction_isolation              read committed
user
 transaction_read_only              off
user
 transform_null_equals              off
user
 unix_socket_directory
postmaster
 unix_socket_group
postmaster
 unix_socket_permissions            0777
postmaster  0            511
 update_process_title               on
superuser
 vacuum_cost_delay                  0
user        0            100
 vacuum_cost_limit                  200
user        1            10000
 vacuum_cost_page_dirty             20
user        0            10000
 vacuum_cost_page_hit               1
user        0            10000
 vacuum_cost_page_miss              10
user        0            10000
 vacuum_defer_cleanup_age           0
sighup      0            1000000
 vacuum_freeze_min_age              50000000
user        0            1000000000
 vacuum_freeze_table_age            150000000
user        0            2000000000
 wal_block_size                     8192
internal    8192         8192
 wal_buffers                        735
postmaster  -1           2147483647
 wal_keep_segments                  0
sighup      0            2147483647
 wal_level                          minimal
postmaster
 wal_receiver_status_interval       10
sighup      0            2147483
 wal_segment_size                   2048
internal    2048         2048
 wal_sync_method                    fdatasync
sighup
 wal_writer_delay                   200
sighup      1            10000
 work_mem                           3716
user        64           2097151
 xmlbinary                          base64
user
 xmloption                          content
user
```

```
 zero_damaged_pages                    off
superuser
(412 rows)
```

4.3.2 stat_db_rpt()

The signature is:

    stat_db_rpt(*beginning_id*, *ending_id*)

Parameters

`beginning_id`

> `beginning_id` is an integer value that represents the beginning session identifier.

`ending_id`

> `ending_id` is an integer value that represents the ending session identifier .

The following example demonstrates the `stat_db_rpt()` function:

```
SELECT * FROM stat_db_rpt(9, 10);
                                stat_db_rpt
------------------------------------------------------------------------------
   DATA from pg_stat_database

 DATABASE   NUMBACKENDS  XACT COMMIT  XACT ROLLBACK   BLKS READ  BLKS HIT
        BLKS ICACHE HIT      HIT RATIO       ICACHE HIT RATIO
------------------------------------------------------------------------------
 edb         1            21            0               92928      101217
        301                  52.05         0.15
```

4.3.3 stat_tables_rpt()

The signature is:

    function_name(*beginning_id*, *ending_id*, *top_n*, *scope*)

Parameters

`beginning_id`

> `beginning_id` is an integer value that represents the beginning session identifier.

`ending_id`

ending_id is an integer value that represents the ending session identifier.

top_n

top_n represents the number of rows to return

scope

scope determines which tables the function returns statistics about. Specify SYS, USER or ALL:

- SYS indicates that the function should return information about system defined tables. A table is considered a system table if it is stored in one of the following schemas: pg_catalog, information_schema, sys, or dbo.

- USER indicates that the function should return information about user-defined tables.

- ALL specifies that the function should return information about all tables.

The following code sample demonstrates the stat_tables_rpt() function:

```
SELECT * FROM stat_tables_rpt(18, 19, 10, 'ALL');

stat_tables_rpt
------------------------------------------------------------------------------
DATA from pg_stat_all_tables ordered by seq scan

SCHEMA        RELATION
    SEQ SCAN   REL TUP READ IDX SCAN    IDX TUP READ   INS    UPD    DEL
------------------------------------------------------------------------------
pg_catalog    pg_class
    8          2952         78          65             0      0      0
pg_catalog    pg_index
    4          448          23          28             0      0      0
pg_catalog    pg_namespace
    4          76           1           1              0      0      0
pg_catalog    pg_database
    3          6            0           0              0      0      0
pg_catalog    pg_authid
    2          1            0           0              0      0      0
sys           edb$snap
    1          15           0           0              1      0      0
public        accounts
    0          0            0           0              0      0      0
public        branches
    0          0            0           0              0      0      0
sys           edb$session_wait_history
    0          0            0           0              25     0      0
sys           edb$session_waits
    0          0            0           0              10     0      0

DATA from pg_stat_all_tables ordered by rel tup read
```

```
SCHEMA       RELATION
    SEQ SCAN   REL TUP READ IDX SCAN    IDX TUP READ INS    UPD    DEL
------------------------------------------------------------------------------
pg_catalog   pg_class
    8          2952         78          65          0      0      0
pg_catalog   pg_index
    4          448          23          28          0      0      0
pg_catalog   pg_namespace
    4          76           1           1           0      0      0
sys          edb$snap
    1          15           0           0           1      0      0
pg_catalog   pg_database
    3          6            0           0           0      0      0
pg_catalog   pg_authid
    2          1            0           0           0      0      0
public       accounts
    0          0            0           0           0      0      0
public       branches
    0          0            0           0           0      0      0
sys          edb$session_wait_history
    0          0            0           0           25     0      0
sys          edb$session_waits
    0          0            0           0           10     0      0
(29 rows)
```

4.3.4 statio_tables_rpt()

The signature is:

statio_tables_rpt(*beginning_id*, *ending_id*, *top_n*, *scope*)

Parameters

beginning_id

beginning_id is an integer value that represents the beginning session identifier.

ending_id

ending_id is an integer value that represents the ending session identifier .

top_n

top_n represents the number of rows to return

scope

scope determines which tables the function returns statistics about. Specify SYS, USER or ALL:

- `SYS` indicates that the function should return information about system defined tables. A table is considered a system table if it is stored in one of the following schemas: `pg_catalog`, `information_schema`, `sys`, or `dbo`.

- `USER` indicates that the function should return information about user-defined tables.

- `ALL` specifies that the function should return information about all tables.

The following example demonstrates the `statio_tables_rpt()` function:

```
edb=# SELECT * FROM statio_tables_rpt(9, 10, 10, 'SYS');

                               statio_tables_rpt
-----------------------------------------------------------------------------
   DATA from pg_statio_all_tables

 SCHEMA      RELATION             HEAP     HEAP     HEAP     IDX      IDX
                                  READ     HIT      ICACHE   READ     HIT
                                                    HIT

             IDX      TOAST    TOAST    TOAST    TIDX     TIDX    TIDX
             ICACHE   READ     HIT      ICACHE   READ     HIT     ICACHE
             HIT                      HIT                        HIT
-----------------------------------------------------------------------------
 public      pgbench_accounts     92766    67215    288      59       32126
             9        0        0        0        0        0        0
 pg_catalog  pg_class             0        296      0        3        16
             0        0        0        0        0        0        0
 sys         edb$stat_all_indexes 8        125      0        4        233
             0        0        0        0        0        0        0
 sys         edb$statio_all_index 8        125      0        4        233
             0        0        0        0        0        0        0
 sys         edb$stat_all_tables  6        91       0        2        174
             0        0        0        0        0        0        0
 sys         edb$statio_all_table 6        91       0        2        174
             0        0        0        0        0        0        0
 pg_catalog  pg_namespace         3        72       0        0        0
             0        0        0        0        0        0        0
 sys         edb$session_wait_his 1        24       0        4        47
             0        0        0        0        0        0        0
 pg_catalog  pg_opclass           3        13       0        2        0
             0        0        0        0        0        0        0
 pg_catalog  pg_trigger           0        12       0        1        15
             0        0        0        0        0        0        0
(16 rows)
```

4.3.5 stat_indexes_rpt()

The signature is:

```
stat_indexes_rpt(beginning_id, ending_id, top_n, scope)
```

Parameters

`beginning_id`

> `beginning_id` is an integer value that represents the beginning session identifier.

`ending_id`

> `ending_id` is an integer value that represents the ending session identifier .

`top_n`

> `top_n` represents the number of rows to return

`scope`

> `scope` determines which tables the function returns statistics about. Specify `SYS`, `USER` or `ALL`:
>
> - `SYS` indicates that the function should return information about system defined tables. A table is considered a system table if it is stored in one of the following schemas: `pg_catalog`, `information_schema`, `sys`, or `dbo`.
> - `USER` indicates that the function should return information about user-defined tables.
> - `ALL` specifies that the function should return information about all tables.

The following code sample demonstrates the `stat_indexes_rpt()` function:

```
edb=# SELECT * FROM stat_indexes_rpt(9, 10, 10, 'ALL');

                            stat_indexes_rpt
------------------------------------------------------------------------------
   DATA from pg_stat_all_indexes

 SCHEMA        RELATION        INDEX
                          IDX SCAN    IDX TUP READ    IDX TUP FETCH
------------------------------------------------------------------------------
 pg_catalog    pg_cast         pg_cast_source_target_index
                          30           7               7
 pg_catalog    pg_class        pg_class_oid_index
                          15          15              15
 pg_catalog    pg_trigger      pg_trigger_tgrelid_tgname_index
                          12          12              12
 pg_catalog    pg_attribute    pg_attribute_relid_attnum_index
                          7           31              31
 pg_catalog    pg_statistic    pg_statistic_relid_att_index
                          7            0               0
```

```
pg_catalog    pg_database    pg_database_oid_index
                         5           5               5
pg_catalog    pg_proc        pg_proc_oid_index
                         5           5               5
pg_catalog    pg_operator    pg_operator_oprname_l_r_n_index
                         3           1               1
pg_catalog    pg_type        pg_type_typname_nsp_index
                         3           1               1
pg_catalog    pg_amop        pg_amop_opr_fam_index
                         2           3               3
(14 rows)
```

4.3.6 statio_indexes_rpt()

The signature is:

statio_indexes_rpt(*beginning_id*, *ending_id*, *top_n*, *scope*)

Parameters

beginning_id

beginning_id is an integer value that represents the beginning session identifier.

ending_id

ending_id is an integer value that represents the ending session identifier .

top_n

top_n represents the number of rows to return

scope

scope determines which tables the function returns statistics about. Specify SYS, USER or ALL:

- SYS indicates that the function should return information about system defined tables. A table is considered a system table if it is stored in one of the following schemas: pg_catalog, information_schema, sys, or dbo.

- USER indicates that the function should return information about user-defined tables.

- ALL specifies that the function should return information about all tables.

The following example demonstrates the `statio_indexes_rpt()` function:

```
edb=# SELECT * FROM statio_indexes_rpt(9, 10, 10, 'SYS');

                            statio_indexes_rpt
------------------------------------------------------------------------------
   DATA from pg_statio_all_indexes

 SCHEMA      RELATION        INDEX
                        IDX BLKS READ   IDX BLKS HIT    IDX BLKS ICACHE HIT
------------------------------------------------------------------------------
public               pgbench_accounts          pgbench_accounts_pkey
                        59              32126           9
 sys                  edb$stat_all_indexes      edb$stat_idx_pk
                        4               233             0
 sys                  edb$statio_all_indexes    edb$statio_idx_pk
                        4               233             0
 sys                  edb$stat_all_tables       edb$stat_tab_pk
                        2               174             0
 sys                  edb$statio_all_tables     edb$statio_tab_pk
                        2               174             0
 sys                  edb$session_wait_history  session_waits_hist_pk
                        4               47              0
 pg_catalog           pg_cast                   pg_cast_source_target_index
                        1               29              0
 pg_catalog           pg_trigger                pg_trig_tgrelid_tgname_index
                        1               15              0
 pg_catalog           pg_class                  pg_class_oid_index
                        1               14              0
 pg_catalog           pg_statistic              pg_statistic_relid_att_index
                        2               12              0
(14 rows)
```

4.4 Performance Tuning Recommendations

To use DRITA reports for performance tuning, review the top five events in a given report, looking for any event that takes a disproportionately large percentage of resources. In a streamlined system, user I/O will probably make up the largest number of waits. Waits should be evaluated in the context of CPU usage and total time; an event may not be significant if it takes 2 minutes out of a total measurement interval of 2 hours, if the rest of the time is consumed by CPU time. The component of response time (CPU "work" time or other "wait" time) that consumes the highest percentage of overall time should be evaluated.

When evaluating events, watch for:

Event type	Description
Checkpoint waits	Checkpoint waits may indicate that checkpoint parameters need to be adjusted, (`checkpoint_segments` and `checkpoint_timeout`).
WAL-related waits	WAL-related waits may indicate `wal_buffers` are under-sized.
SQL Parse waits	If the number of waits is high, try to use prepared statements.
db file random reads	If high, check that appropriate indexes and statistics exist.
db file random writes	If high, may need to decrease `bgwriter_delay`.
btree random lock acquires	May indicate indexes are being rebuilt. Schedule index builds during less active time.

Performance reviews should also include careful scrutiny of the hardware, the operating system, the network and the application SQL statements.

4.5 Event Descriptions

Event Name	Description
`add in shmem lock acquire`	Obsolete/unused
`bgwriter communication lock acquire`	The bgwriter (background writer) process has waited for the short-term lock that synchronizes messages between the bgwriter and a backend process.
`btree vacuum lock acquire`	The server has waited for the short-term lock that synchronizes access to the next available vacuum cycle ID.
`buffer free list lock acquire`	The server has waited for the short-term lock that synchronizes access to the list of free buffers (in shared memory).
`checkpoint lock acquire:`	A server process has waited for the short-term lock that prevents simultaneous checkpoints.
`checkpoint start lock acquire`	The server has waited for the short-term lock that synchronizes access to the bgwriter checkpoint schedule.
`clog control lock acquire`	The server has waited for the short-term lock that synchronizes access to the commit log.
`control file lock acquire`	The server has waited for the short-term lock that synchronizes write access to the control file (this should usually be a low number).
`db file extend`	A server process has waited for the operating system while adding a new page to the end of a file.
`db file read`	A server process has waited for the completion of a read (from disk).
`db file write`	A server process has waited for the completion of a write (to disk).
`db file sync`	A server process has waited for the operating system to flush all changes to disk.
`first buf mapping lock acquire`	The server has waited for a short-term lock that synchronizes access to the shared-buffer mapping table.
`freespace lock acquire`	The server has waited for the short-term lock that synchronizes access to the freespace map.
`Infinite Cache read`	The server has waited for an Infinite Cache read request.
`Infinite Cache write`	The server has waited for an Infinite Cache write request.
`lwlock acquire`	The server has waited for a short-term lock that has not been described elsewhere in this section.
`multi xact gen lock acquire`	The server has waited for the short-term lock that synchronizes access to the next available multi-transaction ID (when a SELECT…FOR SHARE statement executes).
`multi xact member lock acquire`	The server has waited for the short-term lock that synchronizes access to the multi-transaction member file (when a SELECT…FOR SHARE statement executes).
`multi xact offset lock acquire`	The server has waited for the short-term lock that synchronizes access to the multi-transaction offset file (when a SELECT…FOR SHARE statement executes).
`oid gen lock acquire`	The server has waited for the short-term lock that synchronizes access to the next available OID (object ID).
`query plan`	The server has computed the execution plan for a SQL statement.
`rel cache init lock acquire`	The server has waited for the short-term lock that prevents simultaneous relation-cache loads/unloads.
`shmem index lock acquire`	The server has waited for the short-term lock that synchronizes access to the shared-memory map.
`sinval lock acquire`	The server has waited for the short-term lock that synchronizes access to the cache invalidation state.
`sql parse`	The server has parsed a SQL statement.

`subtrans control lock acquire`	The server has waited for the short-term lock that synchronizes access to the subtransaction log.
`tablespace create lock acquire`	The server has waited for the short-term lock that prevents simultaneous CREATE TABLESPACE or DROP TABLESPACE commands.
`two phase state lock acquire`	The server has waited for the short-term lock that synchronizes access to the list of prepared transactions.
`wal insert lock acquire`	The server has waited for the short-term lock that synchronizes write access to the write-ahead log. A high number may indicate that WAL buffers are sized too small.
`wal write lock acquire`	The server has waited for the short-term lock that synchronizes write-ahead log flushes.
`wal file sync`	The server has waited for the write-ahead log to sync to disk (related to the wal_sync_method parameter which, by default, is 'fsync' - better performance can be gained by changing this parameter to open_sync).
`wal flush`	The server has waited for the write-ahead log to flush to disk.
`wal write`	The server has waited for a write to the write-ahead log buffer (expect this value to be high).
`xid gen lock acquire`	The server has waited for the short-term lock that synchronizes access to the next available transaction ID.

4.6 Catalog Views

The following DRITA catalog views provide access to performance information relating to system waits.

4.6.1 edb$system_waits

The edb$system_waits view summarizes the number of waits and the total wait time per session for each wait named. It also displays the average and max wait times. edb$system_waits summarizes the following information:

```
  Column    |     Type        | Modifiers |    Definition
------------+-----------------+-----------+------------------
 edb_id     | numeric         |           |identifier
 dbname     | text            |           |database name
 wait_name  | text            |           |name of the event
 wait_count | numeric         |           |number of times the event occurs
 avg_wait   | numeric(50,6)   |           |average wait time in microseconds
 max_wait   | numeric         |           |maximum wait time in microseconds
 total_wait | numeric         |           |total wait time in microseconds
 wait_name  | text            |           |name of the event
```

The following example shows the result of a SELECT statement on the edb$system_waits view:

```
select * from sys.edb$system_waits;

 edb_id | dbname |wait_name   | wait_count |avg_wait | max_wait  | totalwait
--------+--------+------------+------------+---------+-----------+----------
      1 | edb    |db fileread|        301 |0.011516 | 0.629986 | 2.742500
      1 | edb    |wal flush   |         26 |0.010364 | 0.085380 | 0.269452
      1 | edb    |wal write   |         26 |0.010355 | 0.085371 | 0.269232
      1 | edb    |query plan  |        277 |0.001367 | 0.049425 | 0.192442
      2 | edb    |wal flush   |         28 |0.040443 | 0.095150 | 0.431984
      2 | edb    |wal write   |         28 |0.040434 | 0.095093 | 0.431698
      2 | edb    |query plan  |        299 |0.001479 | 0.049425 | 0.262596
```

4.6.2 edb$session_waits

The edb$session_waits view summarizes the number of waits and the total wait time per session for each wait named and identified by backend ID. It also displays the average and max wait times. edb$session_waits summarizes the following information:

```
    Column        |      Type       | Modifiers |Definition
------------------+-----------------+-----------+----------------
 backend_id       | bigint          |           |session identifier
 wait_count       | bigint          |           |number of times the event
                                                  occurs
 avg_wait_time    | numeric         |           |average wait time in
                                                  microseconds
 max_wait_time    | numeric(50,6)   |           |maximum wait time in
                                                  microseconds
```

```
  total_wait_time | numeric(50,6) |            |total wait time in
                                                microseconds
  wait_name       | text          |            |name of the event
```

The following code sample shows the result of a SELECT statement on the edb$session_waits view:

```
SELECT * FROM sys.edb$session_waits;

 edb_id | dbname | backend_id |  wait_name     | wait_count | avg_wait_time |
max_wait_time| total_wait_time |  usename      |  current_query
--------+--------+------------+---------------+------------+---------------+-
-------------+----------------+--------------------------
      1 | edb    |      22935 | db file read  |        175 |      0.008399 |
    0.629986 |        1.469887 | enterprisedb | <IDLE>
      1 | edb    |      22988 | db file read  |        116 |      0.009556 |
    0.040627 |        1.108438 | enterprisedb | select * from edbsnap();
      1 | edb    |      22988 | wal flush     |        26  |      0.010364 |
    0.085380 |        0.269452 | enterprisedb | select * from edbsnap();
(3 rows)
```

4.6.3 edb$session_wait_history

The edb$session_wait_history view contains the last 25 wait events for each backend ID active during the session. The edb$session_wait_history view includes the following information:

```
    Column    |  Type  | Modifiers |   Definition
 ------------+--------+-----------+-------------------------
  edb_id     | numeric|           |identifier
  dbname     | text   |           |database name
  backend_id | bigint |           |session identifier
  seq        | bigint |           |number between 1 and 25
  wait_name  | text   |           |name of the event
  elapsed    | bigint |           |elapsed time in microseconds
  p1         | bigint |           |variable #1- meaning dependent on
                                              event
  p2         | bigint |           |variable #2- meaning dependent on
                                              event
  p3         | bigint |           |variable #3- meaning dependent on
                                              event
```

The following code sample shows the result of a SELECT statement on the edb$session_wait_history view:

```
SELECT * FROM sys.edb$session_wait_history;

 edb_id | dbname | backend_id | seq |    wait_name    | elapsed | p1 | p2 | p3
--------+--------+------------+-----+----------------+----------+----+----+----
      1 | edb    |      22935 |   1 | query plan     |       54 |   0 |   0 |   0
      1 | edb    |      22935 |   2 | db file read   |     1116 |2689|   8 |   1
      1 | edb    |      22935 |   3 | db file read   |      983 |1255|  32 |   1
      1 | edb    |      22935 |   4 | db file read   |    13717 |2691|  19 |   1
      1 | edb    |      22935 |   5 | query plan     |       75 |    0|   0 |   0
      1 | edb    |      22935 |   6 | db file read   |    11053 |1255|   7 |   1
      1 | edb    |      22935 |   7 | db file read   |      404 |2689|   4 |   1
 (7 rows)
```

5 DBMS_PROFILER

The DBMS_PROFILER package collects and stores performance information about the PL/pgSQL and SPL statements that are executed during a profiling session; you can review the performance information in the tables and views provided by the profiler.

DBMS_PROFILER works by recording a set of performance-related counters and timers for each line of PL/pgSQL or SPL statement that executes within a profiling session. The counters and timers are stored in a table named SYS.PLSQL_PROFILER_DATA. When you complete a profiling session, DBMS_PROFILER will write a row to the performance statistics table for each line of PL/pgSQL or SPL code that executed within the session. For example, if you execute the following function:

```
1 - CREATE OR REPLACE FUNCTION getBalance(acctNumber INTEGER)
2 - RETURN NUMBER AS
3 -     result NUMBER;
4 - BEGIN
5 -   SELECT balance INTO result FROM acct WHERE id = acctNumber;
6 -
7 -     IF (balance IS NULL) THEN
8 -      DBMS_OUTPUT.PUT_LINE('Balance is null');
9 -     END IF;
10-
11-    RETURN result;
12- END;
```

DBMS_PROFILER adds one PLSQL_PROFILER_DATA entry for each line of code within the getBalance() function (including blank lines and comments). The entry corresponding to line 4 will show that the SELECT statement executed exactly one time; and required a very small amount of time to execute. On the other hand, the entry corresponding to line 8 will show that the call to DBMS_OUTPUT.PUT_LINE executed once or not at all (depending on the value for the balance column).

Some of the lines in this function contain no executable code (for example, 6, 9, and 10) so the performance statistics for those lines will always contain *zero* values.

To start a profiling session, invoke the DBMS_PROFILER.START_PROFILER function (or procedure). Once you've invoked START_PROFILER, Advanced Server will profile every PL/pgSQL or SPL function, procedure, trigger, or anonymous block that your session executes until you either stop or pause the profiler (by calling STOP_PROFILER or PAUSE_PROFILER).

It is important to note that when you start (or resume) the profiler, the profiler will only gather performance statistics for functions/procedures/triggers that *start* after the call to START_PROFILER (or RESUME_PROFILER).

While the profiler is active, Advanced Server records a large set of timers and counters in memory; when you invoke the `STOP_PROFILER` (or `FLUSH_DATA`) function/procedure, `DBMS_PROFILER` writes those timers and counters to a set of three tables:

- `SYS.PLSQL_PROFILER_RAWDATA`
  Contains the performance counters and timers for each statement executed within the session.

- `SYS.PLSQL_PROFILER_RUNS`
  Contains a summary of each run (aggregating the information found in `PLSQL_PROFILER_RAWDATA`).

- `SYS.PLSQL_PROFILER_UNITS`
  Contains a summary of each code unit (function, procedure, trigger, or anonymous block) executed within a session.

In addition, DBMS_PROFILER defines a view, `SYS.PLSQL_PROFILER_DATA`, which contains a subset of the `PLSQL_PROFILER_RAWDATA` table (in a form that is compatible with Oracle's DBMS_PROFILER package).

Please note that a non-superuser may *gather* profiling information, but may not view that profiling information unless a superuser grants specific privileges on the profiling tables (stored in the `SYS` schema). This permits a non-privileged user to gather performance statistics without exposing information that the administrator may want to keep secret (i.e., PL/SQL code).

5.1 Querying the DBMS_PROFILER Tables and View

The following step-by-step example uses DBMS_PROFILER to retrieve performance information for procedures, functions, and triggers included in the sample data distributed with Advanced Server.

1. Open the EDB-PSQL command line, and establish a connection to the Advanced Server database. Use an `EXEC` statement to start the profiling session:

```
acctg=# EXEC dbms_profiler.start_profiler('profile list_emp');
EDB-SPL Procedure successfully completed
```

   (Note: the call to `start_profiler()` includes a comment that DBMS_PROFILER associates with the profiler session).

2. Then, use an `EXEC` statement a call to the `list_emp` procedure:

```
acctg=# EXEC list_emp;
EMPNO    ENAME
-----    -------
7369     SMITH
7499     ALLEN
7521     WARD
7566     JONES
7654     MARTIN
7698     BLAKE
7782     CLARK
7788     SCOTT
7839     KING
7844     TURNER
7876     ADAMS
7900     JAMES
7902     FORD
7934     MILLER
```

3. Stop the profiling session with a call to `dbms_profiler.stop_profiler`:

```
EDB-SPL Procedure successfully completed
acctg=# EXEC dbms_profiler.stop_profiler;
```

4. Start a new session with the `dbms_profiler.start_profiler` function (followed by a new comment):

```
EDB-SPL Procedure successfully completed
acctg=# EXEC dbms_profiler.start_profiler('profile get_dept_name
and emp_sal_trig');
EDB-SPL Procedure successfully completed
```

5. Invoke the `get_dept_name` function:

```
acctg=# SELECT emp_admin.get_dept_name(10);
 get_dept_name
---------------
 ACCOUNTING
(1 row)
```

6. Execute an `UPDATE` statement that causes a trigger to execute:

```
acctg=# UPDATE memp SET sal = 500 WHERE empno = 7902;
Updating employee 7902
..Old salary: 3000.00
..New salary: 500.00
..Raise     : -2500.00
User korry updated employee(s) on 2011-03-11
UPDATE 1
```

7. Terminate the profiling session and flush the performance information to the profiling tables:

```
acctg=# EXEC dbms_profiler.stop_profiler;
```

8. Now, query the `plsql_profiler_runs` table to view a list of the profiling sessions, arranged by `runid`:

```
EDB-SPL Procedure successfully completed
acctg=# SELECT * FROM plsql_profiler_runs;

 runid | related_run | run_owner |         run_date          |
 run_comment               | run_total_time | run_system_info | run_comment1 |
spare1
-------+-------------+-----------+---------------------------+-----------------
-----------------------+----------------+-----------------+-------------+-----
--
     1 |             | korry     | 11-MAR-11 12:19:04.439087 | profile list_emp
                      |           4211 |                 |             |

     2 |             | korry     | 11-MAR-11 12:19:53.655886 | profile
get_dept_name and emp_sal_trig |          16950 |                 |
 |
(2 rows)
```

9. Query the `plsql_profiler_units` table to view the amount of time consumed by each unit (each function, procedure, or trigger):

```
acctg=# SELECT * FROM plsql_profiler_units;
 runid | unit_number | unit_type | unit_owner |            unit_name
 | unit_timestamp | total_time | spare1 | spare2
-------+-------------+-----------+------------+---------------------------------
-+----------------+------------+--------+--------
     1 |       16895 | PROCEDURE | korry      | list_emp()
 |                |          4 |        |
     2 |       16911 | FUNCTION  | korry      | get_dept_name(p_deptno numeric)
|                |          0 |        |
     2 |       16908 |           | korry      | emp_sal_trig_memp()
|                |          2 |        |
     2 |       16906 |           | korry      | user_audit_trig_memp()
 |                |          15 |        |
(4 rows)
```

10. Query the `plsql_profiler_rawdata` table to view a list of the wait event counters and wait event times:

```
acctg=# SELECT * FROM plsql_profiler_rawdata;
 runid |                                        sourcecode
                  | func_oid | line_number | exec_count | tuples_returned |
time_
-------+----------------------------------------------------------------
----------------+----------+------------+-----------+----------------+----
--
     1 |
                  |    16895 |           0 |          0 |                0 |

     1 |     v_empno          NUMBER(4);
                 |    16895 |           1 |          0 |                0 |

     1 |     v_ename          VARCHAR2(10);
                  |    16895 |           2 |          0 |                0 |

     1 |     CURSOR emp_cur IS
                  |    16895 |           3 |          0 |                0 |

     1 |         SELECT empno, ename FROM memp ORDER BY empno;
                  |    16895 |           4 |          0 |                0 |

     1 | BEGIN
                  |    16895 |           5 |          1 |                0 |
0.0
     1 |     OPEN emp_cur;
                  |    16895 |           6 |          1 |                0 |
0.0
     1 |     DBMS_OUTPUT.PUT_LINE('EMPNO    ENAME');
                  |    16895 |           7 |          1 |                0 |
0.0
     1 |     DBMS_OUTPUT.PUT_LINE('-----    -------');
                  |    16895 |           8 |          1 |                0 |
0.0
     1 |     LOOP
                 |    16895 |           9 |         15 |                0 |
0.0
     1 |         FETCH emp_cur INTO v_empno, v_ename;
                 |    16895 |          10 |         15 |                0 |
0.0
     1 |         EXIT WHEN emp_cur%NOTFOUND;
                  |    16895 |          11 |         14 |                0 |
0.0
     1 |         DBMS_OUTPUT.PUT_LINE(v_empno || '      ' || v_ename);
                 |    16895 |          12 |          0 |                0 |

     1 |     END LOOP;
                  |    16895 |          13 |          1 |                0 |
 2.
```

```
     1 |     CLOSE emp_cur;
             |    16895 |         14 |         0 |             0 |

     1 | END
              |    16895 |          15 |          0 |             0 |

     2 | DECLARE
              |    16908 |           0 |          0 |             0 |

     2 |     sal_diff       NUMBER;
             |    16908 |          1 |         0 |             0 |

     2 | BEGIN
              |    16908 |           2 |          0 |             0 |

     2 |     IF INSERTING THEN
              |    16908 |           3 |          1 |             0 |
 5.
     2 |         DBMS_OUTPUT.PUT_LINE('Inserting employee ' || :NEW.empno);
             |    16908 |          4 |         0 |             0 |

     2 |         DBMS_OUTPUT.PUT_LINE('..New salary: ' || :NEW.sal);
              |    16908 |           5 |          0 |             0 |

     2 |     END IF;
              |    16908 |           6 |          0 |             0 |

     2 |     IF UPDATING THEN
             |    16908 |          7 |         1 |             0 |
0.0
     2 |         sal_diff := :NEW.sal - :OLD.sal;
             |    16908 |          8 |         1 |             0 |
0.0
     2 |         DBMS_OUTPUT.PUT_LINE('Updating employee ' || :OLD.empno);
              |    16908 |           9 |          1 |             0 |
0.0
     2 |         DBMS_OUTPUT.PUT_LINE('..Old salary: ' || :OLD.sal);
              |    16908 |          10 |          1 |             0 |
0.0
     2 |         DBMS_OUTPUT.PUT_LINE('..New salary: ' || :NEW.sal);
              |    16908 |          11 |          1 |             0 |
0.0
     2 |         DBMS_OUTPUT.PUT_LINE('..Raise     : ' || sal_diff);
              |    16908 |          12 |          1 |             0 |
 0.
     2 |     END IF;
              |    16908 |          13 |          0 |             0 |

     2 |     IF DELETING THEN
             |    16908 |         14 |         1 |             0 |
 3.
     2 |         DBMS_OUTPUT.PUT_LINE('Deleting employee ' || :OLD.empno);
              |    16908 |          15 |          0 |             0 |

     2 |         DBMS_OUTPUT.PUT_LINE('..Old salary: ' || :OLD.sal);
              |    16908 |          16 |          0 |             0 |

     2 |     END IF;
              |    16908 |          17 |          0 |             0 |

     2 | END
              |    16908 |          18 |          0 |             0 |
     2 |
                  |    16911 |            0 |          0 |                0 |

     2 |         v_dname         VARCHAR2(14);
                  |    16911 |            1 |          0 |                0 |

     2 |     BEGIN
                  |    16911 |            2 |          1 |                0 |
0.0
     2 |         SELECT dname INTO v_dname FROM mdept WHERE deptno = p_deptno;
                  |    16911 |            3 |          1 |                0 |
 3.
     2 |         RETURN v_dname;
                  |    16911 |            4 |          0 |                0 |

     2 |     EXCEPTION
                  |    16911 |            5 |          0 |                0 |

     2 |         WHEN NO_DATA_FOUND THEN
                  |    16911 |            6 |          0 |                0 |

     2 |             DBMS_OUTPUT.PUT_LINE('Invalid department number ' ||
p_deptno);              |    16911 |            7 |          0 |                0
|
     2 |             RETURN '';
                 |    16911 |            8 |          0 |                0 |

     2 |     END
                  |    16911 |            9 |          0 |                0 |

     2 | DECLARE
                  |    16906 |            0 |          0 |                0 |

     2 |     v_action        VARCHAR2(24);
                  |    16906 |            1 |          0 |                0 |

     2 | BEGIN
                  |    16906 |            2 |          0 |                0 |

     2 |     IF INSERTING THEN
                  |    16906 |            3 |          1 |                0 |
0.0
     2 |         v_action := ' added employee(s) on ';
                  |    16906 |            4 |          0 |                0 |

     2 |     ELSIF UPDATING THEN
                  |    16906 |            5 |          1 |                0 |
 6.
     2 |         v_action := ' updated employee(s) on ';
                  |    16906 |            6 |          1 |                0 |
 3.
     2 |     ELSIF DELETING THEN
                  |    16906 |            7 |          0 |                0 |

     2 |         v_action := ' deleted employee(s) on ';
                  |    16906 |            8 |          0 |                0 |

     2 |     END IF;
                  |    16906 |            9 |          0 |                0 |

     2 |     DBMS_OUTPUT.PUT_LINE('User ' || USER || v_action ||
TO_CHAR(SYSDATE,'YYYY-MM-DD')); |    16906 |           10 |          1 |
      0 |   0.0
     2 | END
                  |    16906 |          11 |           0 |               0 |

(57 rows)
```

11. Query the `plsql_profiler_data` view to review an Oracle-compatible subset of the information found in `plsql_profiler_rawdata` table:

```
acctg=# SELECT * FROM plsql_profiler_data;
 runid | unit_number | line# | total_occur | total_time | min_time | max_time |
spare1 | spare2 | spare3 | spare4
-------+-------------+-------+-------------+------------+----------+----------
+--------+--------+--------+--------
     1 |       16895 |     0 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |     1 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |     2 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |     3 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |     4 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |     5 |           1 |   0.000322 | 0.000322 | 0.000322 |
       |        |        |
     1 |       16895 |     6 |           1 |   0.000675 | 0.000675 | 0.000675 |
       |        |        |
     1 |       16895 |     7 |           1 |   0.000115 | 0.000115 | 0.000115 |
       |        |        |
     1 |       16895 |     8 |           1 |   0.001605 | 0.001605 | 0.001605 |
       |        |        |
     1 |       16895 |     9 |          15 |   0.000183 |    5e-06 |    5e-05 |
       |        |        |
     1 |       16895 |    10 |          15 |   0.000458 |    5e-06 | 0.000378 |
       |        |        |
     1 |       16895 |    11 |          14 |   0.000831 |  2.9e-05 | 0.000448 |
       |        |        |
     1 |       16895 |    12 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |    13 |           1 |    2.2e-05 |  2.2e-05 |  2.2e-05 |
       |        |        |
     1 |       16895 |    14 |           0 |          0 |        0 |        0 |
       |        |        |
     1 |       16895 |    15 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     0 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     1 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     2 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     3 |           1 |    5.1e-05 |  5.1e-05 |  5.1e-05 |
       |        |        |
     2 |       16908 |     4 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     5 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     6 |           0 |          0 |        0 |        0 |
       |        |        |
     2 |       16908 |     7 |           1 |   0.000846 | 0.000846 | 0.000846 |
       |        |        |
 2 |       16908 |     8 |           1 |   0.000129 | 0.000129 | 0.000129 |
   |         |         |
 2 |       16908 |     9 |           1 |   0.000239 | 0.000239 | 0.000239 |
   |         |         |
 2 |       16908 |    10 |           1 |   0.000163 | 0.000163 | 0.000163 |
   |         |         |
 2 |       16908 |    11 |           1 |   0.000158 | 0.000158 | 0.000158 |
   |         |         |
 2 |       16908 |    12 |           1 |    0.00011 |  0.00011 |  0.00011 |
   |         |         |
 2 |       16908 |    13 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16908 |    14 |           1 |    3.4e-05 |  3.4e-05 |  3.4e-05 |
   |         |         |
 2 |       16908 |    15 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16908 |    16 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16908 |    17 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16908 |    18 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     0 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     1 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     2 |           1 |   0.000328 | 0.000328 | 0.000328 |
   |         |         |
 2 |       16911 |     3 |           1 |    3.8e-05 |  3.8e-05 |  3.8e-05 |
   |         |         |
 2 |       16911 |     4 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     5 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     6 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     7 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     8 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16911 |     9 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     0 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     1 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     2 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     3 |           1 |   0.000138 | 0.000138 | 0.000138 |
   |         |         |
 2 |       16906 |     4 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     5 |           1 |    6.6e-05 |  6.6e-05 |  6.6e-05 |
   |         |         |
 2 |       16906 |     6 |           1 |    3.2e-05 |  3.2e-05 |  3.2e-05 |
   |         |         |
 2 |       16906 |     7 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     8 |           0 |          0 |        0 |        0 |
   |         |         |
 2 |       16906 |     9 |           0 |          0 |        0 |        0 |
   |         |         |
     2 |       16906 |    10 |           1 |   0.014618 | 0.014618 | 0.014618 |
       |         |         |
     2 |       16906 |    11 |           0 |          0 |        0 |        0 |
       |         |         |
(57 rows)
```

5.2 DBMS_PROFILER Functions and Procedures

The DBMS_PROFILER package collects and stores performance information about the PL/pgSQL and SPL statements that are executed during a profiling session; use the functions and procedures listed below to control the profiling tool.

Table 5-1 DBMS_PROFILER Functions/Procedures

Function/Procedure	Function or Procedure	Return Type	Description
`FLUSH_DATA`	Function and Procedure	Status Code or Exception	Flushes performance data collected in the current session without terminating the session (profiling continues).
`GET_VERSION (major OUT, minor OUT)`	Procedure	n/a	Returns the version number of this package.
`INTERNAL_VERSION_CHECK`	Function	Status Code	Confirms that the current version of the profiler will work with the current database.
`PAUSE_PROFILER`	Function and Procedure	Status Code or Exception	Pause data collection.
`RESUME_PROFILER`	Function and Procedure	Status Code or Exception	Resume data collection.
`START_PROFILER[`*`run_comment, run_comment1, run_number OUT`*`]`	Functions and Procedures	Status Code or Exception	Start data collection.
`STOP_PROFILER`	Function and Procedure	Status Code or Exception	Stop data collection and flush performance data to PLSQL_PROFILER_RAWDATA.

Return Values

The functions within the DBMS_PROFILER package return a status code to indicate success or failure; the DBMS_PROFILER procedures raise an exception only if they encounter a failure. The status codes and messages returned by the functions, and the exceptions raised by the procedures are listed in the table below.

Status Code	Message	Exception	Description
-1	`error version`	`version_mismatch`	The profiler version and the database are incompatible.
0	`success`	`n/a`	The operation completed successfully.
1	`error_param`	`profiler_error`	The operation received an incorrect parameter.
2	`error_io`	`profiler_error`	The data flush operation has failed.

5.2.1 FLUSH_DATA

The `FLUSH_DATA` procedure or function flushes the data collected in the current session without terminating the profiler session. The data is flushed to the tables listed in Section 6.3 of the Postgres Plus Advanced Server Performance Features Guide. The signature of the `FLUSH_DATA` function is:

```
DBMS_PROFILER.FLUSH_DATA
    RETURN INTEGER;
```

The signature of the FLUSH_DATA procedure is:

```
DBMS_PROFILER.FLUSH_DATA;
```

5.2.2 GET_VERSION

The GET_VERSION procedure returns the version of DBMS_PROFILER. The procedure signature is:

```
DBMS_PROFILER.GET_VERSION( major OUT INTEGER
                           minor OUT INTEGER);
```

Parameters

major

The major version number of DBMS_PROFILER.

minor

The minor version number of DBMS_PROFILER.

5.2.3 INTERNAL_VERSION_CHECK

The INTERNAL_VERSION_CHECK function confirms that the current version of DBMS_PROFILER will work with the current database. The function signature is:

```
DBMS_PROFILER.INTERNAL_VERSION_CHECK
    RETURN INTEGER;
```

5.2.4 PAUSE_PROFILER

The PAUSE_PROFILER function or procedure pauses a profiling session. The function signature is:

```
DBMS_PROFILER.PAUSE_PROFILER
    RETURN INTEGER;
```

The signature of the PAUSE_PROFILER procedure is:

```
DBMS_PROFILER.PAUSE_PROFILER;
```

5.2.5 RESUME_PROFILER

The `RESUME_PROFILER` function or procedure resumes a paused profiling session. The function signature is:

```
DBMS_PROFILER.RESUME_PROFILER
    RETURN INTEGER;
```

The signature of the `RESUME_PROFILER` procedure is:

```
DBMS_PROFILER.RESUME_PROFILER;
```

5.2.6 START_PROFILER

The `START_PROFILER` function or procedure starts a data collection session. The `START_PROFILER` function has two forms:

```
DBMS_PROFILER.START_PROFILER(
    run_comment IN TEXT := sysdate,
    run_comment1 IN TEXT := '',
    run_number OUT INTEGER)
  RETURN INTEGER;

DBMS_PROFILER.START_PROFILER(
    run_comment IN TEXT := sysdate,
    run_comment1 IN TEXT := '')
  RETURN INTEGER;
```

The `START_PROFILER` procedure has two forms:

```
DBMS_PROFILER.START_PROFILER (
    run_comment IN TEXT := sysdate,
    run_comment1 IN TEXT := '');

DBMS_PROFILER.START_PROFILER (
    run_comment IN TEXT := sysdate,
    run_comment1 IN TEXT := '',
    run_number OUT INTEGER);
```

Parameters

`run_comment`

A user-defined comment for the profiler session; the default value is `sysdate`.

`run_comment1`

An additional user-defined comment for the profiler session; the default value is `''`.

`run_number`

The session number of the profiler session.

5.2.7 STOP_PROFILER

The `STOP_PROFILER` function or procedure stops a profiling session and flushes the performance information to the DBMS_PROFILER tables and view. The `STOP_PROFILER` function signature is:

```
DBMS_PROFILER.STOP_PROFILER
  RETURN INTEGER;
```

The signature of the `START_PROFILER` procedure is:

```
DBMS_PROFILER.STOP_PROFILER;
```

5.3 DBMS_PROFILER - Reference

The Advanced Server installer creates the following tables and views that you can query to review PL/SQL performance profile information:

Table Name	Description
PLSQL_PROFILER_RUNS	Table containing information about all profiler runs, organized by runid.
PLSQL_PROFILER_UNITS	Table containing information about all profiler runs, organized by unit.
PLSQL_PROFILER_DATA	Oracle-compatible view containing performance statistics.
PLSQL_PROFILER_RAWDATA	Table containing the Oracle-compatible performance statistics *and* the extended performance statistics for DRITA counters and timers.

5.3.1 PLSQL_PROFILER_RUNS

The PLSQL_PROFILER_RUNS table contains the following columns:

Column	Data Type	Description
runid	INTEGER (NOT NULL)	Unique identifier (plsql_profiler_runnumber)
related_run	INTEGER	The runid of a related run.
run_owner	TEXT	The role that recorded the profiling session.
run_date	TIMESTAMP WITHOUT TIME ZONE	The profiling session start time.
run_comment	TEXT	User comments relevant to this run
run_total_time	BIGINT	Run time (in nanoseconds)
run_system_info	TEXT	Currently Unused
run_comment1	TEXT	Additional user comments
spare1	TEXT	Currently Unused

5.3.2 PLSQL_PROFILER_UNITS

The PLSQL_PROFILER_UNITS table contains the following columns:

Column	Data Type	Description
runid	INTEGER	Unique identifier (plsql_profiler_runnumber)
unit_number	OID	Corresponds to the OID of the row in the pg_proc table that identifies the unit.
unit_type	TEXT	PL/SQL function, procedure, trigger or anonymous block
unit_owner	TEXT	The identity of the role that owns the unit.
unit_name	TEXT	The complete signature of the unit.
unit_timestamp	TIMESTAMP WITHOUT TIME ZONE	Creation date of the unit (currently NULL).
total_time	BIGINT	Time spent within the unit (in nanoseconds)
spare1	BIGINT	Currently Unused
spare2	BIGINT	Currently Unused

5.3.3 PLSQL_PROFILER_DATA

The PLSQL_PROFILER_DATA view contains the following columns:

Column	Data Type	Description
runid	INTEGER	Unique identifier (plsql_profiler_runnumber)
unit_number	OID	Object ID of the unit that contains the current line.
line#	INTEGER	Current line number of the profiled workload.
total_occur	BIGINT	The number of times that the line was executed.
total_time	DOUBLE PRECISION	The amount of time spent executing the line.
min_time	DOUBLE PRECISION	The minimum execution time for the line.
max_time	DOUBLE PRECISION	The maximum execution time for the line.
spare1	NUMBER	Currently Unused
spare2	NUMBER	Currently Unused
spare3	NUMBER	Currently Unused
spare4	NUMBER	Currently Unused

5.3.4 PLSQL_PROFILER_RAWDATA

The PLSQL_PROFILER_RAWDATA table contains the statistical information that is found in the PLSQL_PROFILER_DATA view, as well as the performance statistics returned by the DRITA counters and timers.

Column	Data Type	Description
runid	INTEGER	The run identifier (plsql_profiler_runnumber).
sourcecode	TEXT	The individual line of profiled code.
func_oid	OID	Object ID of the unit that contains the current line.
line_number	INTEGER	Current line number of the profiled workload.
exec_count	BIGINT	The number of times that the line was executed.
time_total	DOUBLE PRECISION	The amount of time spent executing the line.
time_shortest	DOUBLE PRECISION	The minimum execution time for the line.
time_longest	DOUBLE PRECISION	The maximum execution time for the line.
tuples_returned	BIGINT	Currently Unused
num_scans	BIGINT	Currently Unused
tuples_fetched	BIGINT	Currently Unused
tuples_inserted	BIGINT	Currently Unused
tuples_updated	BIGINT	Currently Unused
tuples_deleted	BIGINT	Currently Unused
blocks_fetched	BIGINT	Currently Unused
blocks_hit	BIGINT	Currently Unused
wal_write	BIGINT	The server has waited for a write to the write-ahead log buffer (expect this value to be high).
wal_flush	BIGINT	The server has waited for the write-ahead log to flush to disk.
wal_file_sync	BIGINT	The server has waited for the write-ahead log to sync to disk (related to the wal_sync_method parameter which, by default, is 'fsync' - better performance can be gained by changing this

		parameter to open_sync).
buffer_free_list_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the list of free buffers (in shared memory).
shmem_index_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the shared-memory map.
oid_gen_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the next available OID (object ID).
xid_gen_lock_acquire	bigint	The server has waited for the short-term lock that synchronizes access to the next available transaction ID.
proc_array_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the process array
sinval_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the cache invalidation state.
freespace_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the freespace map.
wal_insert_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes write access to the write-ahead log. A high number may indicate that WAL buffers are sized too small.
wal_write_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes write-ahead log flushes.
control_file_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes write access to the control file (this should usually be a low number).
checkpoint_lock_acquire	BIGINT	A server process has waited for the short-term lock that prevents simultaneous checkpoints.
clog_control_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the commit log.
subtrans_control_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the subtransaction log.
multi_xact_gen_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the next available multi-transaction ID (when a SELECT...FOR SHARE statement executes).
multi_xact_offset_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the multi-transaction offset file (when a SELECT...FOR SHARE statement executes).
multi_xact_member_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the multi-transaction member file (when a SELECT...FOR SHARE statement executes).
rel_cache_init_lock_acquire	BIGINT	The server has waited for the short-term lock that prevents simultaneous relation-cache loads/unloads.
bgwriter_communication_lock_acquire	BIGINT	The bgwriter (background writer) process has waited for the short-term lock that synchronizes messages between the bgwriter and a backend process.
two_phase_state_lock_acqui	BIGINT	The server has waited for the short-term lock that

re		synchronizes access to the list of prepared transactions.
tablespace_create_lock_acquire	BIGINT	The server has waited for the short-term lock that prevents simultaneous CREATE TABLESPACE or DROP TABLESPACE commands.
btree_vacuum_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the next available vacuum cycle ID.
add_in_shmem_lock_acquire	BIGINT	Currently Unused
autovacuum_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the shared autovacuum state.
autovacuum_schedule_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the autovacuum schedule.
syncscan_lock_acquire	BIGINT	The server has waited for the short-term lock that coordinates synchronous scans.
icache_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to InfiniteCache state
breakpoint_lock_acquire	BIGINT	The server has waited for the short-term lock that synchronizes access to the debugger breakpoint list.
lwlock_acquire	BIGINT	The server has waited for a short-term lock that has not been described elsewhere in this section.
db_file_read	BIGINT	A server process has waited for the completion of a read (from disk).
db_file_write	BIGINT	A server process has waited for the completion of a write (to disk).
db_file_sync	BIGINT	A server process has waited for the operating system to flush all changes to disk.
db_file_extend	BIGINT	A server process has waited for the operating system while adding a new page to the end of a file.
sql_parse	BIGINT	Currently Unused
query_plan	BIGINT	The server has generated a query plan.
infinitecache_read	BIGINT	The server has waited for an Infinite Cache read request.
infinitecache_write	BIGINT	The server has waited for an Infinite Cache write request.
wal_write_time	BIGINT	The amount of time that the server has waited for a write to the write-ahead log buffer (expect this value to be high).
wal_flush_time	BIGINT	The amount of time that the server has waited for the write-ahead log to flush to disk.
wal_file_sync_time	BIGINT	The amount of time that the server has waited for the write-ahead log to sync to disk (related to the wal_sync_method parameter which, by default, is 'fsync' - better performance can be gained by changing this parameter to open_sync).
buffer_free_list_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the list of free buffers (in shared memory).
shmem_index_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to

		the shared-memory map.
oid_gen_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the next available OID (object ID).
xid_gen_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the next available transaction ID.
proc_array_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the process array.
sinval_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the cache invalidation state.
freespace_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the freespace map.
wal_insert_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes write access to the write-ahead log. A high number may indicate that WAL buffers are sized too small.
wal_write_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes write-ahead log flushes.
control_file_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes write access to the control file (this should usually be a low number).
checkpoint_lock_acquire_time	BIGINT	The amount of time that the server process has waited for the short-term lock that prevents simultaneous checkpoints.
clog_control_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the commit log.
subtrans_control_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the subtransaction log.
multi_xact_gen_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the next available multi-transaction ID (when a SELECT...FOR SHARE statement executes).
multi_xact_offset_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the multi-transaction offset file (when a SELECT...FOR SHARE statement executes).
multi_xact_member_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the multi-transaction member file (when a SELECT...FOR SHARE statement executes).
rel_cache_init_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that prevents simultaneous relation-cache loads/unloads.
bgwriter_communication_lock_acquire_time	BIGINT	The amount of time that the bgwriter (background writer) process has waited for the short-term lock that synchronizes messages

		between the bgwriter and a backend process.
two_phase_state_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the list of prepared transactions.
tablespace_create_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that prevents simultaneous CREATE TABLESPACE or DROP TABLESPACE commands.
btree_vacuum_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the next available vacuum cycle ID.
add_in_shmem_lock_acquire_time	BIGINT	Obsolete/unused
autovacuum_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the shared autovacuum state.
autovacuum_schedule_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the autovacuum schedule.
syncscan_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that coordinates synchronous scans.
icache_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to InfiniteCache state
breakpoint_lock_acquire_time	BIGINT	The amount of time that the server has waited for the short-term lock that synchronizes access to the debugger breakpoint list.
lwlock_acquire_time	BIGINT	The amount of time that the server has waited for a short-term lock that has not been described elsewhere in this section.
db_file_read_time	BIGINT	The amount of time that the server process has waited for the completion of a read (from disk).
db_file_write_time	BIGINT	The amount of time that the server process has waited for the completion of a write (to disk).
db_file_sync_time	BIGINT	The amount of time that the server process has waited for the operating system to flush all changes to disk.
db_file_extend_time	BIGINT	The amount of time that the server process has waited for the operating system while adding a new page to the end of a file.
sql_parse_time	BIGINT	The amount of time that the server has parsed a SQL statement.
query_plan_time	BIGINT	The amount of time that the server has computed the execution plan for a SQL statement.
infinitecache_read_time	BIGINT	The amount of time that the server has waited for an Infinite Cache read request.
infinitecache_write_time	BIGINT	The amount of time that the server has waited for an Infinite Cache write request.
totalwaits	BIGINT	The total number of event waits.
totalwaittime	BIGINT	The total time spent waiting for an event.

6 Index Advisor

The Index Advisor utility helps determine which columns you should index to improve performance in a given workload. Index Advisor considers B-tree (single-column or composite) index types, and does not identify other index types (GIN, GiST, Hash) that may improve performance. Index Advisor is installed with Postgres Plus Advanced Server.

Index Advisor works with Advanced Server's query planner by creating *hypothetical indexes* that the query planner uses to calculate execution costs as if such indexes were available. Index Advisor identifies the indexes by analyzing SQL queries supplied in the workload.

There are three ways to use Index Advisor to analyze SQL queries:

- Invoke the Index Advisor utility program, supplying a text file containing the SQL queries that you wish to analyze; Index Advisor will generate a text file with `CREATE INDEX` statements for the recommended indexes.

- Provide queries at the EDB-PSQL command line that you want Index Advisor to analyze.

- Access Index Advisor through the Postgres Enterprise Manager client. When accessed via the PEM client, Index Advisor works with SQL Profiler, providing indexing recommendations on code captured in SQL traces. For more information about using SQL Profiler and Index Advisor with PEM, please see Section 8.4 of the *PEM Getting Started Guide*, available from the EnterpriseDB website at:

  http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager

Index Advisor will attempt to make indexing recommendations on `INSERT`, `UPDATE`, `DELETE` and `SELECT` statements. When invoking Index Advisor, you supply the workload in the form of a set of queries (if you are providing the command in an SQL file) or an `EXPLAIN` statement (if you are specifying the SQL statement at the psql command line). Index Advisor displays the query plan and estimated execution cost for the supplied query, but does not actually execute the query.

During the analysis, Index Advisor compares the query execution costs with and without hypothetical indexes. If the execution cost using a hypothetical index is less than the execution cost without it, both plans are reported in the `EXPLAIN` statement output, metrics that quantify the improvement are calculated, and Index Advisor generates the `CREATE INDEX` statement needed to create the index.

If no hypothetical index can be found that reduces the execution cost, Index Advisor displays only the original query plan output of the `EXPLAIN` statement.

Index Advisor does not actually create indexes on the tables. Use the `CREATE INDEX` statements supplied by Index Advisor to add any recommended indexes to your tables.

A script supplied with Advanced Server creates the table in which Index Advisor stores the indexing recommendations generated by the analysis; the script also creates a function and a view of the table to simplify the retrieval and interpretation of the results.

If you choose to forego running the script, Index Advisor will log recommendations in a temporary table that is available only for the duration of the Index Advisor session.

6.1 Index Advisor Components

The Index Advisor shared library interacts with the query planner to make indexing recommendations. The Postgres Plus Advanced Server installer creates the following shared library in the `libdir` subdirectory of your Postgres Plus Advanced Server home directory:

On Linux:

```
index_advisor.so
```

On Windows:

```
index_advisor.dll
```

Please note that libraries in the `libdir` directory can only be loaded by a superuser. A database administrator can allow a non-superuser to use Index Advisor by manually copying the Index Advisor file from the `libdir` directory into the `libdir/plugins` directory (under your Advanced Server home directory). Only a trusted non-superuser should be allowed access to the plugin; this is an unsafe practice in a production environment.

The installer also creates the Index Advisor utility program and setup script:

`pg_advise_index`

> `pg_advise_index` is a utility program that reads a user-supplied input file containing SQL queries and produces a text file containing `CREATE INDEX` statements that can be used to create the indexes recommended by the Index Advisor. The `pg_advise_index` program is located in the `bin` subdirectory of the Postgres Plus Advanced Server home directory.

`index_advisor.sql`

> `index_advisor.sql` is a script that creates a permanent Index Advisor log table along with a function and view to facilitate reporting of recommendations from the log table. The script is located in the `share/contrib` subdirectory of the Postgres Plus Advanced Server directory.

The `index_advisor.sql` script creates the `index_advisor_log` table, the `show_index_recommendations()` function and the `index_recommendations` view. These database objects must be created in a schema that is accessible by, and included in the search path of the role that will invoke Index Advisor.

`index_advisor_log`

> Index Advisor logs indexing recommendations in the `index_advisor_log` table. If Index Advisor does not find the `index_advisor_log` table in the user's search path, Index Advisor will store any indexing recommendations in a temporary table of the same name. The temporary table exists only for the duration of the current session.

`show_index_recommendations()`

> `show_index_recommendations()` is a PL/pgSQL function that interprets and displays the recommendations made during a specific Index Advisor session (as identified by its backend process ID).

`index_recommendations`

> Index Advisor creates the `index_recommendations` view based on information stored in the `index_advisor_log` table during a query analysis. The view produces output in the same format as the `show_index_recommendations()` function, but contains Index Advisor recommendations for all stored sessions, while the result set returned by the `show_index_recommendations()` function are limited to a specified session.

6.2 Index Advisor Configuration

Index Advisor does not require any configuration to generate recommendations that are available only for the duration of the current session; to store the results of multiple sessions, you must create the `index_advisor_log` table (where Advanced Server will store Index Advisor recommendations). To create the `index_advisor_log` table , you must run the `index_advisor.sql` script.

When selecting a storage schema for the Index Advisor table, function and view, keep in mind that all users that invoke Index Advisor (and query the result set) must have `USAGE` privileges on the schema. The schema must be in the search path of all users that are interacting with the Index Advisor.

1. Place the selected schema at the start of your `search_path` parameter. For example, if your search path is currently:

   ```
   search_path=public, accounting
   ```
   and you want the Index Advisor objects to be created in a schema named `advisor`, use the command:
   ```
   SET search_path = advisor, public, accounting;
   ```

2. Run the `index_advisor.sql` script to create the database objects. If you are running the psql client, you can use the command:

   ```
      \i full_pathname/index_advisor.sql
   ```
   Specify the pathname to the `index_advisor.sql` script in place of *`full_pathname`*.

3. Grant privileges on the `index_advisor_log` table to all Index Advisor users; this step is not necessary if the Index Advisor user is a superuser, or the owner of these database objects.

   - Grant `SELECT` and `INSERT` privileges on the `index_advisor_log` table to allow a user to invoke Index Advisor.

   - Grant `DELETE` privileges on the `index_advisor_log` table to allow the specified user to delete the table contents.

   - Grant `SELECT` privilege on the `index_recommendations` view.

The following example demonstrates the creation of the Index Advisor database objects in a schema named `ia`, which will then be accessible to an Index Advisor user with user name *`ia_user`*:

```
$ edb-psql -d edb -U enterprisedb
edb-psql (9.2.0.0)
Type "help" for help.

edb=# CREATE SCHEMA ia;
CREATE SCHEMA
edb=# SET search_path TO ia;
SET
edb=# \i /opt/PostgresPlus/9.2AS/share/contrib/index_advisor.sql
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE FUNCTION
```

```
CREATE FUNCTION
CREATE VIEW
edb=# GRANT USAGE ON SCHEMA ia TO ia_user;
GRANT
edb=# GRANT SELECT, INSERT, DELETE ON index_advisor_log TO ia_user;
GRANT
edb=# GRANT SELECT ON index_recommendations TO ia_user;
GRANT
```

While using Index Advisor, the specified schema (`ia`) must be included in *`ia_user`*'s `search_path` parameter.

6.3 Using Index Advisor

When you invoke Index Advisor, you must supply a workload; the workload is either a query (specified at the command line), or a file that contains a set of queries (executed by the `pg_advise_index()` function). After analyzing the workload, Index Advisor will either store the result set in a temporary table, or in a permanent table. You can review the indexing recommendations generated by Index Advisor and use the `CREATE INDEX` statements generated by Index Advisor to create the recommended indexes.

Note: You should not run Index Advisor in read-only transactions.

The following examples assume that superuser `enterprisedb` is the Index Advisor user, and the Index Advisor database objects have been created in a schema in the `search_path` of superuser `enterprisedb`.

The examples in the following sections use the table created with the statement shown below:

```
CREATE TABLE t( a INT, b INT );
INSERT INTO t SELECT s, 99999 - s FROM generate_series(0,99999) AS s;
ANALYZE t;
```

The resulting table contains the following rows:

```
   a    |   b
-------+-------
     0 | 99999
     1 | 99998
     2 | 99997
     3 | 99996
       .
       .
       .
 99997 |     2
 99998 |     1
 99999 |     0
```

6.3.1 Using the pg_advise_index Utility

When invoking the `pg_advise_index` utility, you must include the name of a file that contains the queries that will be executed by `pg_advise_index`; the queries may be on the same line, or on separate lines, but each query must be terminated by a semicolon. Queries within the file should not begin with the `EXPLAIN` keyword.

The following example shows the contents of a sample `workload.sql` file:

```
SELECT * FROM t WHERE a = 500;
SELECT * FROM t WHERE b < 1000;
```

Run the `pg_advise_index` program as shown in the code sample below:

```
$ pg_advise_index -d edb -h localhost -U enterprisedb -s 100M -o advisory.sql
workload.sql
poolsize = 102400 KB
load workload from file 'workload.sql'
Analyzing queries .. done.
size = 2184 KB, benefit = 1684.720000
size = 2184 KB, benefit = 1655.520000
/* 1. t(a): size=2184 KB, benefit=1684.72 */
/* 2. t(b): size=2184 KB, benefit=1655.52 */
/* Total size = 4368KB */
```

In the code sample, the `-d`, `-h`, and `-U` options are psql connection options.

`-s`

> `-s` is an optional parameter that limits the maximum size of the indexes recommended by Index Advisor. If Index Advisor does not return a result set, `-s` may be set too low.

`-o`

> The recommended indexes are written to the file specified after the `-o` option.

The information displayed by the `pg_advise_index` program is logged in the `index_advisor_log` table. In response to the command shown in the example, Index Advisor writes the following `CREATE INDEX` statements to the `advisory.sql` output file

```
create index idx_t_1 on t (a);
create index idx_t_2 on t (b);
```

You can create the recommended indexes at the psql command line with the `CREATE INDEX` statements in the file, or create the indexes by executing the `advisory.sql` script.

```
$ edb-psql -d edb -h localhost -U enterprisedb -e -f advisory.sql
create index idx_t_1 on t (a);
CREATE INDEX
create index idx_t_2 on t (b);
CREATE INDEX
```

6.3.2 Using Index Advisor at the psql Command Line

You can use Index Advisor to analyze SQL statements entered at the edb-psql (or psql) command line; the following steps detail loading the Index Advisor plugin and using Index Advisor:

1. Connect to the server with the `edb-psql` command line utility, and load the Index Advisor plugin:

   ```
   $ edb-psql -d edb -U enterprisedb
   …
   edb=# LOAD 'index_advisor';
   LOAD
   ```

2. Use the `edb-psql` command line to invoke each SQL command that you would like Index Advisor to analyze. Index Advisor stores any recommendations for the queries in the `index_advisor_log` table. If the `index_advisor_log` table does not exist in the user's `search_path`, a temporary table is created with the same name. This temporary table exists only for the duration of the user's session.

After loading the Index Advisor plugin, Index Advisor will analyze all SQL statements and log any indexing recommendations for the duration of the session.

> If you would like Index Advisor to analyze a query (and make indexing recommendations) without actually executing the query, preface the SQL statement with the `EXPLAIN` keyword.
>
> If you do not preface the statement with the `EXPLAIN` keyword, Index Advisor will analyze the statement while the statement executes, writing the indexing recommendations to the `index_advisor_log` table for later review.

In the example that follows, the `EXPLAIN` statement displays the normal query plan, followed by the query plan of the same query, if the query were using the recommended hypothetical index:

```
edb=# EXPLAIN SELECT * FROM t WHERE a < 10000;
                                QUERY PLAN
-----------------------------------------------------------------------------
Seq Scan on t  (cost=0.00..1693.00 rows=10105 width=8)
  Filter: (a < 10000)
Result  (cost=0.00..337.10 rows=10105 width=8)
  One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
```

```
  ->  Index Scan using "<hypothetical-index>:1" on t
      (cost=0.00..337.10 rows=10105 width=8)
        Index Cond: (a < 10000)
(6 rows)


edb=# EXPLAIN SELECT * FROM t WHERE a = 100;
                                QUERY PLAN
------------------------------------------------------------------------------
Seq Scan on t  (cost=0.00..1693.00 rows=1 width=8)
  Filter: (a = 100)
Result  (cost=0.00..8.28 rows=1 width=8)
  One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
  ->  Index Scan using "<hypothetical-index>:3" on t
      (cost=0.00..8.28 rows=1 width=8)
        Index Cond: (a = 100)
(6 rows)
```

For information about reviewing the recommended queries, see Section 8.4, *Reviewing the Index Advisor Recommendations*.

After loading the Index Advisor plugin, the default value of `index_advisor.enabled` is `on`. The Index Advisor plugin must be loaded to use a `SET` or `SHOW` command to display the current value of `index_advisor.enabled`.

You can use the `index_advisor.enabled` parameter to temporarily disable Index Advisor without interrupting the psql session:

```
edb=# SET index_advisor.enabled TO off;
SET
```

To enable Index Advisor, set the parameter to `on`:

```
edb=# SET index_advisor.enabled TO on;
SET
```

6.4 Reviewing the Index Advisor Recommendations

There are several ways to review the index recommendations generated by Index Advisor. You can:

- Query the `index_advisor_log` table.
- Run the `show_index_recommendations` function.
- Query the `index_recommendations` view.

6.4.1 Using the show_index_recommendations() Function

To review the recommendations of the Index Advisor utility using the `show_index_recommendations()` function, call the function, specifying the process ID of the session:

```
SELECT show_index_recommendations( pid );
```

Where *pid* is the process ID of the current session. If you do not know the process ID of your current session, passing a value of `NULL` will also return a result set for the current session.

The following code fragment shows an example of a row in a result set:

```
edb=# SELECT show_index_recommendations(null);
                        show_index_recommendations
---------------------------------------------------------------------------
 create index idx_t_a on t(a);/* size: 2184 KB, benefit: 3040.62,
 gain: 1.39222666981456 */
(1 row)
```

In the example, `create index idx_t_a on t(a)` is the SQL statement needed to create the index suggested by Index Advisor. Each row in the result set shows:

- The command required to create the recommended index.
- The maximum estimated size of the index.
- The calculated benefit of using the index.
- The estimated gain that will result from implementing the index.

You can display the results of all Index Advisor sessions from the following view:

```
SELECT * FROM index_recommendations;
```

6.4.2 Querying the index_advisor_log Table

Index Advisor stores indexing recommendations in a table named `index_advisor_log`. Each row in the `index_advisor_log` table contains the result of a query where Index Advisor determines it can recommend a hypothetical index to reduce the execution cost of that query.

Column	Type	Description
`reloid`	`oid`	OID of the base table for the index
`relname`	`name`	Name of the base table for the index
`attrs`	`integer[]`	Recommended index columns (identified by column number)
`benefit`	`real`	Calculated benefit of the index for this query

index_size	integer	Estimated index size in disk-pages
backend_pid	integer	Process ID of the process generating this recommendation
timestamp	timestamp	Date/Time when the recommendation was generated

You can query the index_advisor_log table at the psql command line. The following example shows the index_advisor_log table entries resulting from two Index Advisor sessions. Each session contains two queries, and can be identified (in the table below) by a different backend_pid value. For each session, Index Advisor generated two index recommendations.

```
  edb=# SELECT * FROM index_advisor_log;
   reloid | relname | attrs | benefit | index_size | backend_pid |
timestamp
  --------+---------+-------+---------+------------+-------------+-----------
-----------------------
    16651 | t       | {1}   | 1684.72 |       2184 |        3442 | 22-MAR-11
16:44:32.712638 -04:00
    16651 | t       | {2}   | 1655.52 |       2184 |        3442 | 22-MAR-11
16:44:32.759436 -04:00
    16651 | t       | {1}   | 1355.9  |       2184 |        3506 | 22-MAR-11
16:48:29.317016 -04:00
    16651 | t       | {1}   | 1684.72 |       2184 |        3506 | 22-MAR-11
16:51:45.927906 -04:00
  (4 rows)
```

Index Advisor added the first two rows to the table after analyzing the following two queries executed by the pg_advise_index utility:

```
  SELECT * FROM t WHERE a = 500;
  SELECT * FROM t WHERE b < 1000;
```

The value of 3442 in column backend_pid identifies these results as coming from the session with process ID 3442.

The value of 1 in column attrs in the first row indicates that the hypothetical index is on the first column of the table (column a of table t).

The value of 2 in column attrs in the second row indicates that the hypothetical index is on the second column of the table (column b of table t).

Index Advisor added the last two rows to the table after analyzing the following two queries (executed at the psql command line):

```
   edb=# EXPLAIN SELECT * FROM t WHERE a < 10000;
                                             QUERY PLAN
   -----------------------------------------------------------------------------
--------------------
    Seq Scan on t  (cost=0.00..1693.00 rows=10105 width=8)
      Filter: (a < 10000)
    Result  (cost=0.00..337.10 rows=10105 width=8)
      One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
      ->  Index Scan using "<hypothetical-index>:1" on t  (cost=0.00..337.10
rows=10105 width=8)
            Index Cond: (a < 10000)
   (6 rows)

   edb=# EXPLAIN SELECT * FROM t WHERE a = 100;
                                        QUERY PLAN
   -------------------------------------------------------------------------
--------------
    Seq Scan on t  (cost=0.00..1693.00 rows=1 width=8)
      Filter: (a = 100)
    Result  (cost=0.00..8.28 rows=1 width=8)
      One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
      ->  Index Scan using "<hypothetical-index>:3" on t  (cost=0.00..8.28
rows=1 width=8)
            Index Cond: (a = 100)
   (6 rows)
```

The values in the benefit column of the `index_advisor_log` table are calculated using the following formula:

```
benefit = (normal execution cost) - (execution cost with hypothetical index)
```

The value of the `benefit` column for the last row of the `index_advisor_log` table (shown in the example) is calculated using the query plan for the following SQL statement:

```
EXPLAIN SELECT * FROM t WHERE a = 100;
```

The execution costs of the different execution plans are evaluated and compared:

```
benefit = (Seq Scan on t cost) - (Index Scan using <hypothetical-
index>)
```

and the benefit is added to the table:

```
benefit = 1693.00 - 8.28
benefit = 1684.72
```

You can delete rows from the `index_advisor_log` table when you no longer have the need to review the results of the queries stored in the row.

6.4.3 Querying the index_recommendations View

The `index_recommendations` view contains the calculated metrics and the `CREATE INDEX` statements to create the recommended indexes for all sessions whose results are currently in the `index_advisor_log` table. You can display the results of all stored Index Advisor sessions by querying the `index_recommendations` view as shown below:

```
SELECT * FROM index_recommendations;
```

Using the example shown in the previous section (*Querying the* `index_advisor_log` *Table*), the `index_recommendations` view displays the following:

```
 edb=# SELECT * FROM index_recommendations;
  backend_pid |                                      show_index_recommendations
 -------------+-------------------------------------------------------------
-------------------------------
          3442 | create index idx_t_a on t(a);/* size: 2184 KB, benefit:
1684.72, gain: 0.771392654586624 */
          3442 | create index idx_t_b on t(b);/* size: 2184 KB, benefit:
1655.52, gain: 0.758021539820856 */
          3506 | create index idx_t_a on t(a);/* size: 2184 KB, benefit:
3040.62, gain: 1.39222666981456 */
 (3 rows)
```

Within each session, the results of all queries that benefit from the same recommended index are combined to produce one set of metrics per recommended index, reflected in the fields named `benefit` and `gain`.

The formulas for the fields are as follows:

```
size    = MAX(index size of all queries)
benefit = SUM(benefit of each query)
gain    = SUM(benefit of each query) / MAX(index size of all queries)
```

So for example, using the following query results from the process with a `backend_pid` of `3506`:

```
  reloid | relname | attrs | benefit | index_size | backend_pid |
timestamp
 --------+---------+-------+---------+------------+-------------+-----------
----------------------
   16651 | t       | {1}   |  1355.9 |       2184 |        3506 | 22-MAR-11
16:48:29.317016 -04:00
   16651 | t       | {1}   | 1684.72 |       2184 |        3506 | 22-MAR-11
16:51:45.927906 -04:00
```

The metrics displayed from the `index_recommendations` view for `backend_pid` `3506` are:

```
  backend_pid |                                      show_index_recommendations
 -------------+-------------------------------------------------------------
-------------------------------
          3506 | create index idx_t_a on t(a);/* size: 2184 KB, benefit:
3040.62, gain: 1.39222666981456 */
```

The metrics from the view are calculated as follows:

```
benefit = (benefit from 1st query) + (benefit from 2nd
query)
benefit = 1355.9 + 1684.72
benefit = 3040.62
```

and

```
gain = ((benefit from 1st query) + (benefit from 2nd
query)) / MAX(index size of all queries)
gain = (1355.9 + 1684.72) / MAX(2184, 2184)
gain = 3040.62 / 2184
gain = 1.39223
```

The gain metric is useful when comparing the relative advantage of the different recommended indexes derived during a given session. The larger the gain value, the better the cost effectiveness derived from the index weighed against the possible disk space consumption of the index.

6.5 Limitations

Index Advisor does not consider Index Only scans; it does consider Index scans when making recommendations.

Index Advisor ignores any computations found in the `WHERE` clause. Effectively, the index field in the recommendations will not be any kind of expression; the field will be a simple column name.

Index Advisor does not consider inheritance when recommending hypothetical indexes. If a query references a parent table, Index Advisor does not make any index recommendations on child tables.

Restoration of a `pg_dump` backup file that includes the `index_advisor_log` table or any tables for which indexing recommendations were made and stored in the `index_advisor_log` table, may result in "broken links" between the `index_advisor_log` table and the restored tables referenced by rows in the `index_advisor_log` table because of changes in object identifiers (OIDs).

If it is necessary to display the recommendations made prior to the backup, you can replace the old OIDs in the `reloid` column of the `index_advisor_log` table with the new OIDs of the referenced tables using the SQL `UPDATE` statement:

```
UPDATE index_advisor_log SET reloid = new_oid WHERE reloid
= old_oid;
```

7 Other Performance Features

This chapter provides a brief summary of other performance related features of Postgres Plus Advanced Server.

7.1 SQL Profiler

Inefficient SQL code is one of, if not the leading cause of database performance problems. The challenge for database administrators and developers is locating and then optimizing this code in large, complex systems.

SQL Profiler helps you locate and optimize poorly running SQL code.

Specific features and benefits of SQL Profiler include the following:

- **On-Demand Traces.** You can capture SQL traces at any time by manually setting up your parameters and starting the trace.
- **Scheduled Traces.** For inconvenient times, you can also specify your trace parameters and schedule them to run at some later time.
- **Save Traces.** Execute your traces and save them for later review.
- **Trace Filters.** Selectively filter SQL captures by database and by user, or capture every SQL statement sent by all users against all databases.
- **Trace Output Analyzer.** A graphical table lets you quickly sort and filter queries by duration or statement, and a graphical or text based `EXPLAIN` plan lays out your query paths and joins.
- **Index Advisor Integration.** Once you have found your slow queries and optimized them, you can also let the Index Advisor recommend the creation of underlying table indices to further improve performance.

For more information about SQL Profiler and Postgres Enterprise Manager, visit the EnterpriseDB website at:

http://www.enterprisedb.com/postgres-enterprise-manager

7.2 Query Optimization Hints

The Advanced Server query planner performs the task of determining how the result set should be produced for `DELETE`, `SELECT`, and `UPDATE` SQL commands. The query planner uses cost based optimization to determine the least cost plan.

There may be cases where you want to force the query planner to either utilize (or not to utilize) a certain access method to determine the plan (for example, use a sequential scan instead of an index scan).

The *query hints* feature (also called *optimizer hints*) of Advanced Server allows you to embed these directives within the SQL command to force the query planner to use a certain access method.

Query hints are typically used with the `EXPLAIN` command so you can see the estimated costs associated with the generated plan based on the query hint you supplied.

This technique allows you to perform a more detailed cost comparison of how certain queries are expected to perform.

For more information about optimizer hints, see the Oracle Compatibility Developer's Guide, available from the EnterpriseDB website at:

http://www.enterprisedb.com/documentation

7.3 Hi-Speed Bulk Loader

Hi-speed bulk loading is provided by the Postgres Plus Advanced Server *EDB*Loader* feature. EDB*Loader is a command line utility that loads data from an input source, typically a file, into one or more tables using directives compatible with Oracle SQL*Loader.

EDB*Loader provides the following features and benefits:

- Three data loading methods providing various levels of performance – conventional path load, direct path load, and parallel direct path load
- *Conventional path load* for full insert processing, which includes all integrity checks
- *Direct path load* for faster performance by writing data directly to database pages bypassing insert processing overhead of a conventional path load
- *Parallel direct path load* for even faster performance by distributing the loading process over multiple, parallel sessions
- Oracle SQL*Loader compatible syntax for control file directives
- Input data with delimiter-separated or fixed-width fields
- Bad file for collecting rejected records
- Loading of multiple target tables
- Discard file for collecting records that do not meet the selection criteria of any target table
- Log file for recording the EDB*Loader session and any error messages
- Data loading from standard input and remote loading, particularly useful for large data sources on remote hosts

For more information about EDB*Loader, see the Oracle Compatibility Developer's Guide, available from the EnterpriseDB website at:

http://www.enterprisedb.com/documentation

7.4 Bulk Collect Fetch and Binding

SQL statements that return a result set consisting of a large number of rows may not be operating as efficiently as possible due to the constant context switching that must occur between the database server and the client in order to transfer the entire result set.

Performance can be improved in this scenario by using a collection (that is, an array) to gather the entire result set in memory, which the client can then access. This is accomplished by creating a collection and then using the SQL statement with the `BULK COLLECT` clause to gather the result set in the collection.

This performance enhancement can be applied to the following types of SQL statements:

- `SELECT INTO`
- `FETCH INTO`
- `EXECUTE IMMEDIATE INTO`
- `DELETE RETURNING INTO`
- `INSERT RETURNING INTO`
- `UPDATE RETURING INTO`

For more information, see the Oracle Compatibility Developer's Guide, available from the EnterpriseDB website at:

http://www.enterprisedb.com/documentation

7.5 Multi-Threaded Replication

The *Postgres Plus xDB Replication Server* is a comprehensive, yet easy-to-use system for replicating tables between database servers. Not only can data be replicated using Postgres Plus Advanced Server databases, but tables from Oracle, Microsoft® SQL Server®, or PostgreSQL® databases can be replicated to Advanced Server and vice versa.

xDB Replication Server provides the following benefits and features:

- Replicate Oracle, SQL Server, or PostgreSQL tables and views to Advanced Server and vice versa

- Distributed multi publication/subscription architecture
- Synchronize data across geographies
- Multi-threaded replication support
- Snapshot and continuous replication modes
- Row filtering
- Flexible replication scheduling
- Cascading replication support
- Replication history viewer
- Management via a graphical replication console or a command line driven interface

xDB Replication Server takes advantage of the system architecture of the host database server. Replication can be done in parallel mode with multiple threads on a multi-CPU or core architecture.

For more information, see the Postgres Plus xDB Replication Server User's Guide, available from the EnterpriseDB website at:

http://www.enterprisedb.com/docs/en/9.1/repguide/Contents.htm