Infobright DB PostgreSQL Edition

2019.2 GA USER GUIDE

Ignite
TECHNOLOGIES

This edition published December 2019.

Table of Contents

1. About Infobright DB

Infobright Overview

Thank you for choosing to install Infobright DB (IBDB) for PostgreSQL. Infobright is a column-oriented, high performance analytic engine designed for analytic applications that need fast query response across large data volumes. Infobright was designed specifically for large volume, multi-TB data analytics applications.

Infobright uses a unique and patent-pending approach to compressing, storing, and processing data that allows it to be installed and run on commodity hardware with little or no DBA intervention. Infobright requires little tuning to support ad hoc or complex business analytic queries.

Infobright is a database engine utilizing the PostgreSQL database environment. As such, Infobright uses the PostgreSQL administrative interface to reduce the learning curve for system administrators.

Infobright Enterprise Edition provides a versatile, highly-compressed database system optimized for analytic-type queries. The ratio of possible compression and the speed of data import and retrieval are optimized at the expense of some transactional features of the engine performance, like frequent data updating.

Infobright executes complex or ad hoc queries across vast amounts of data with a low cost of ownership.

Architectural Overview

The IBDB for PostgreSQL release introduces new architectural concepts. A new modularized approach aims to improve overall scalability and robustness.

IBDB combines the Infobright Engine with the PostgreSQL server implementation. Functionally the solution can be depicted as follows:

PostgreSQL provides:

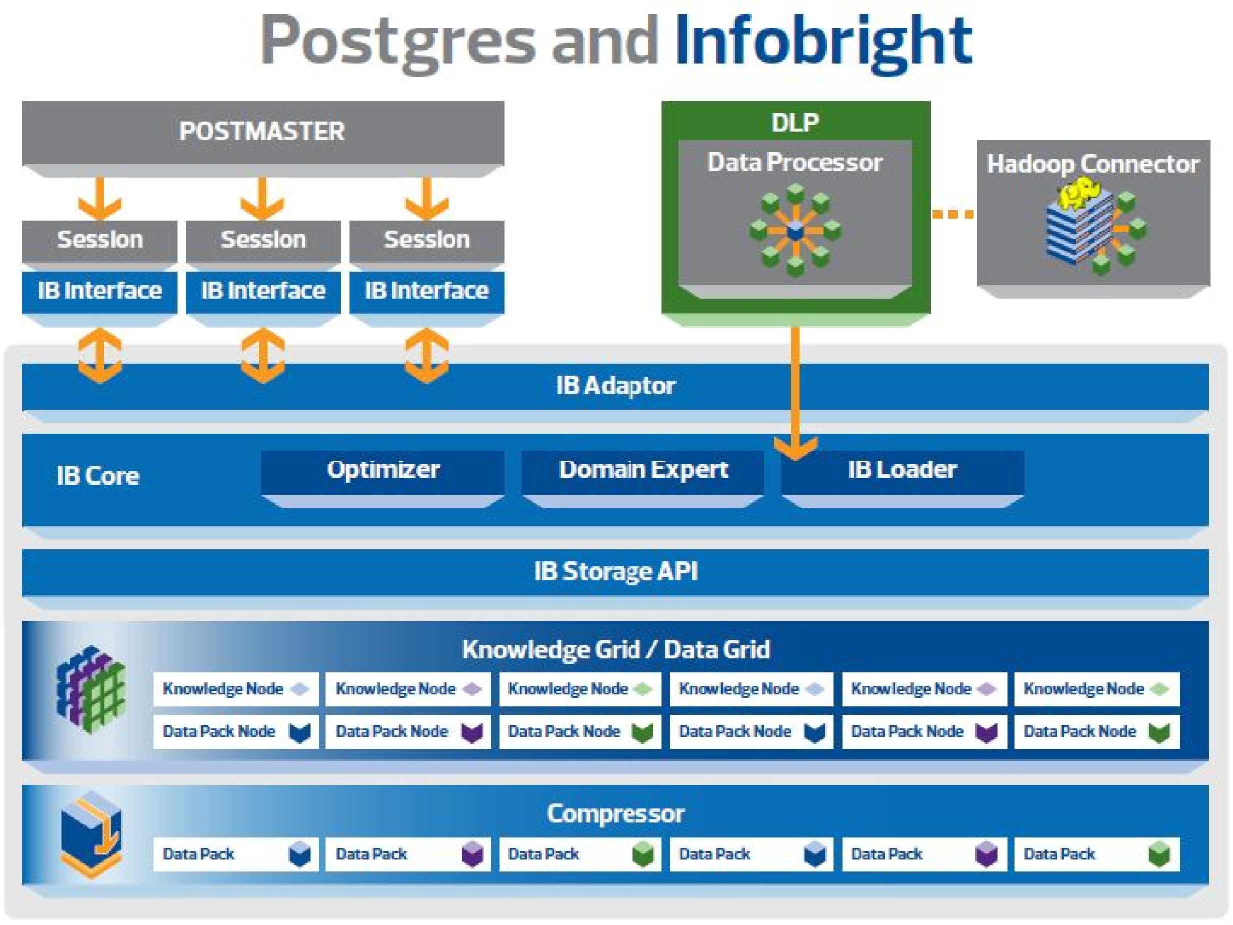
- Mature connectors, tools and resources
- Interconnectivity and certification with BI tools
- Management services and utilities

Infobright provides:

- Load function that compresses data
- Column-oriented storage engine
- Knowledge Grid metadata layer that contains information about the compressed data
- Optimizer/executor that uses the Knowledge Grid

Infobright includes its own computing engine along with the storage engine. The PostgreSQL query engine can be used with Infobright; however, since the PostgreSQL storage engine interface is row oriented, it cannot take full advantage of the column orientation or the Knowledge Grid and hence query execution via this path is reduced. Queries will be directed to the Infobright Optimizer whenever possible. Infobright ships with the full PostgreSQL binaries required.

Infobright and PostgreSQL are integrated as shown below:



2. Setting up Infobright

Technical Requirements

Before installing Infobright, review the following technical requirements.

Infobright Technical Requirements

Requirement	Description
Platforms	Windows Server 2008 and 2012 Red Hat Enterprise Linux 6.x and 7.x CentOS 6.x and 7.x Debian 6 and 8

	Novell SUSE Linux Enterprise 11
Processor Architecture	Intel 64-bit AMD 64-bit
For Personal Evaluation and/or Application Development	
CPU Speed	1.8 GHz minimum, 2.0GHz or faster dual or quad core recommended
Memory	4GB RAM minimum, 8GB recommended
For Multi-User Evaluation or Production Deployment	
CPU Speed	2.0 GHz minimum, 8 cores minimum
Memory	16GB RAM minimum 32GB recommended

Linux for Infobright

Infobright has been optimized for various flavours of Linux. While Infobright can be run "out of the box" on any supported Linux platform, there are a number of tuning opportunities to improve performance.

See "Linux Tuning Settings" on page for a list of tuning suggestions.

Infobright DB Postgres Installation

About Installation Packages

The Infobright installation packages are provided in RPM, DEB and .exe formats. For non-Windows platforms, the user installing Infobright must be the root user or a user with the necessary permissions to install files.

Note

Installation of multiple Infobright DB Postgres instances on the same OS is not supported.

Infobright DB Postgres Windows Installation

Important

The user installing Infobright DB Postgres must have **local** administrator rights. The user cannot use domain administrator rights to install Infobright DB Postgres.

Windows Installation Instructions

1. Log into the Customer Portal and download the zipped install package (e.g. infobright-iee_postgres-2019.2.0-0-win_2008_2012_64.zip) to the Windows Server machine on which you are installing Infobright.
2. Unzip to an ".exe" install package (e.g. infobright-iee-postgres-2019.2.0-0.win_64.exe).

3. Double click on the .exe file to launch the Install Wizard. Click **Next** to continue.
4. By default IBDB is installed in **C:\Program Files\Infobright Products\IEE\Postgres\2019.2.0**. To change the default location, enter the folder name in the field or click **Browse…** to select an install location. Infobright recommends using the convention **C:\Program Files\Infobright Products\IEE\<variant>\<version>** to accommodate for future modules.
5. Click **Install**. The Install Wizard completes the installation.
6. The Install Wizard automatically creates the file **infobright.cnf.sample** in the **ib_data** directory. The **infobright.cnf.sample** file contains descriptions and default values for all customizable Infobright parameters. The same will be copied into **infobright.cnf** in the **ib_data** directory by the installer. The **infobright.cnf** file is used to override the default values for all customizable Infobright parameters. The recommended way to create **infobright.cnf** is to simply create a copy of **infobright.cnf.sample**, and make any desired parameter overrides. For more information, see "Configuring Infobright (infobright.cnf file)" on page .
7. The Install Wizard automatically creates Infobright DB as a Windows Service, which allows the Infobright Server to be started and stopped automatically when you boot or shutdown Windows. If you do not want Infobright DB to start on boot, open the Services window from the Control Panel and change the Startup Type for Infobright from **Automatic** to **Manual**.
8. The Install Wizard automatically determines the optimum memory settings based on the physical memory of the system. You may change this setting by updating the value of the **ServerMainHeapSize** parameter in the **infobright.cnf** file within the **ib_data** directory.

Important
The initial memory settings assume that there are no other services consuming significant memory on the machine. If other services consume significant memory, please lower the memory settings for Infobright. For more information, please refer to "ServerMainHeapSize" on page .

9. Beginning with release 4.8.0, every instance of the Infobright Server requires that a valid license file exists. The license file is normally called **infobright.lic** and is located in the **ib_data** directory. To obtain a valid license file please contact Infobright Support. For more information, please refer to "The Infobright License File" on page .

"Silent" Installation

You may choose to embed Infobright as part of another application. In this scenario the Infobright binaries can be installed "silently"; that is, they can be installed without any Infobright specific prompts appearing during installation. The calling application must ensure sufficient information is passed to the Infobright installation process in order to carry out a silent install.

The basic silent installer can be executed by specifying the **/S** parameter. For example:

```
C:\> START /WAIT C:\infobright-iee-postgres-2019.2.0-0.win_64.exe /S
```

If the silent installation will be in a directory other than the default, the directory can be specified using the **/D** parameter. For example:

```
C:\> START /WAIT C:\infobright-iee-postgres-2019.2.0-0.win_64.exe /S
/D=C:\Program Files\Foo
```

The silent installation can be performed by a user without administrator rights by specifying the **/noadmin=yes** parameter. For example:

```
C:\> START /WAIT C:\infobright-iee-postgres-2019.2.0-0.win_64.exe /S
/noadmin=yes /D=C:\Program Files\Foo
```

Important

Infobright DB Postgres Windows edition utilizes runtime components of Visual C++ Libraries. The installation (silent or non-silent) will check if the Microsoft Visual C++ runtime components are already installed, and normally install them (if it is found they are not pre-installed).

However, when using the **/noadmin=yes** parameter, no attempt will be made to install the runtime components (as to do so would require administrator rights). Instead the installation will exit with an error if the runtime components have not been pre-installed.

Hence, when using the **/noadmin=yes** parameter, the Microsoft Visual C++ runtime components must be pre-installed. This can be accomplished by downloading and installing the Microsoft Visual C++ 2010 Redistributable Package found at http://www.microsoft.com/en-ca/download/details.aspx?id=13523.

Notes:

1. The **/S** and **/D** parameters are case-sensitive and must be uppercase.
2. Do not add any quotes in the input of the **/D** parameter.
3. The **/D** parameter must be the last parameter.
4. To check if the installation finished successfully, you can check the exit code after installation finishes by issuing the following command:

   ```
   C:\> ECHO %ERRORLEVEL%
   ```

5. The installation log is temporarily placed under **%TEMP%\infobright_install.log** where **TEMP** can be determined using the **ECHO %TEMP%** command.

Uninstalling on Windows

To uninstall Infobright DB, select **Infobright Uninstall** under the Infobright program group in the Windows Start Menu:

```
    Start/All Programs/Infobright/Infobright Uninstall
```

Note

Uninstalling Infobright will only uninstall binaries. The data directory will not be deleted.

Infobright DB Postgres Linux Installation

Linux RPM and DPKG Installation Instructions

To install Infobright on Linux using the rpm or deb package:

1. Log into the Customer Portal at https://support.infobright.com/index.php/downloads/ and download the installation package (e.g. infobright-iee_postgres-2019.2.0-0-rhel_centos_6_64.rpm).
2. Obtain root user access and run:

```
rpm -i infobright-iee_postgres-<version>-<platform>.rpm
or
dpkg -i infobright-iee_postgres-<version>-<platform>.deb
```

3. Initialize the data directory:

```
service infobright-iee-postgres initdb
```

4. In **/usr/local/infobright-products/iee/postgres/<version>/pg_data**, update **pg_hba.conf** to assign trust to local connections:

```
# "local" is for Unix domain socket connections only
local     all     all                      trust
# IPv4 local connections:
host      all      all   127.0.0.1/32       trust
# IPv6 local connections:
host       all      all   ::1/128           trust
```

5. The installation creates the file **infobright.cnf.sample** in the **ib_data** directory. The **infobright.cnf.sample** file contains descriptions and default values for all customizable Infobright parameters. However, in order to successfully start the Infobright Server, one needs to manually create the file **infobright.cnf** in the **ib_data** directory. The **infobright.cnf** file is used to override the default values for all customizable Infobright parameters. The recommended way to create **infobright.cnf** is to simply create a copy of **infobright.cnf.sample**, and make any desired parameter overrides. For more information, see “Configuring Infobright (infobright.cnf file)” on page .

6. The installation determines the optimum memory settings based on the physical memory of the system. You may change this setting by updating the value of the **ServerMainHeapSize** parameter in the **infobright.cnf** file within the **ib_data** directory.

Important
The initial memory settings assume that there are no other services consuming significant memory on the machine. If other services consume significant memory, please lower the memory settings for Infobright. For more information, please refer to "ServerMainHeapSize" on page .

7. Beginning with release 4.8.0, every instance of the Infobright Server requires that a valid license file exists. The license file is normally called **infobright.lic** and is located

in the **ib_data** directory. To obtain a valid license file please contact Infobright Support. For more information, please refer to "The Infobright License File" on page .

8. Start the Infobright Server:

   ```
   /etc/init.d/infobright-iee-postgres start
   ```

Uninstalling on Linux

To uninstall Infobright, run:

```
rpm -e infobright-iee-postgres
or
dpkg -r infobright-iee-postgres
```

Note
Uninstalling Infobright will only uninstall binaries. The data directory will not be deleted.

Infobright DB Postgres Upgrading

Infobright DB Postgres Windows Upgrade

Important
The user upgrading Infobright DB Postgres must have *local* administrator rights. The user cannot use domain administrator rights to upgrade Infobright DB Postgres.

Windows Upgrade Instructions

1. Follow the standard Infobright DB Windows installation instructions. The Install Wizard automatically detects a previous version of Infobright DB and upgrades your Infobright DB installation while preserving your data and configuration settings.
2. If upgrading from a release of IBDB earlier than 4.8.0, then the following must also be done:
   - Manually create the **infobright.cnf** file as described in "Windows Installation Instructions" on page .
   - Convert any default parameter overrides from the now obsolete **.infobright** and **brighthouse.ini** files to the **infobright.cnf** file. For information on how to do this, see "Cross-Reference Of Pre-4.8.0 Parameters To Current Parameters" on page .
   - Contact Infobright Support to obtain a valid license file as described in "Windows Installation Instructions" on page .
3. Start the Infobright Server from the Start Menu items.
4. If upgrading from a release of IBDB earlier than 5.0.1, a script needs to be run to update the Infobright database in order to accommodate some newly added functions. To accomplish this, at the Windows command prompt execute the following bat script located in the installation directory (e.g. `C:\Program Files\Infobright Products\IEE\Postgres\2019.2.0`):

```
infobright_postgres_upgrade.bat <host> <port> <user>
```

Note

In the above command, replace `<host>` with the server ip address (e.g. `192.168.20.181`), replace `<port>` with the port used to run IEE-Postgres (e.g. usually `5029`), and replace `<user>` with an existing Postgres User that has appropriate privileges to drop / replace functions (e.g. `postgres`).

When upgrading from a release of IBDB earlier than 4.8.0, the above command also updates Column Optimizer triggers and stored procedures.

Important

The upgrade process will upgrade binaries to a new release while still using existing (e.g. previous release) data directories. This is accomplished by ensuring the service used to start/stop IEE-Postgres is updated appropriately.

Infobright DB Postgres Linux Upgrade

Linux RPM and DPKG Upgrade Instructions

Note

Your configuration settings and data will not be changed during the upgrade.

To upgrade Infobright on Linux using the rpm or deb package:

1. Log into the Customer Portal and download the installation package (e.g. infobright-iee_postgres-2019.2.0-0-rhel_centos_6_64.rpm).
2. Obtain root user access and run:

   ```
   rpm -U infobright-iee_postgres-<version>-<platform>.rpm
   or
   dpkg -i infobright-iee_postgres-<version>-<platform>.deb
   ```

3. If upgrading from a release of IBDB earlier than 4.8.0, then the following must also be done:
   - Manually create the **infobright.cnf** file as described in "Linux RPM and DPKG Installation Instructions" on page .
   - Convert any default parameter overrides from the now obsolete **.infobright** and **brighthouse.ini** files to the **infobright.cnf** file. For information on how to do this, see "Cross-Reference Of Pre-4.8.0 Parameters To Current Parameters" on page .
   - Contact Infobright Support to obtain a valid license file as described in "Linux RPM and DPKG Installation Instructions" on page .
4. Start the Infobright Server:

   ```
   /etc/init.d/infobright-iee-postgres start
   ```

5. If upgrading from a release of IBDB earlier than 5.0.1, a script needs to be run to update the Infobright database in order to accommodate some newly added functions. To

accomplish this, execute the following bash script located in the installation directory (e.g. `/usr/local/infobright-products/iee/postgres/2019.2.0`):

```
infobright_postgres_upgrade.sh -h <host> -p <port> -u <user>
```

Note
In the above command, replace `<host>` with the server ip address (e.g. `192.168.20.181`), replace `<port>` with the port used to run Infobright DB Postgres (e.g. usually `5029`), and replace `<user>` with an existing Postgres User that has appropriate privileges to drop / replace functions (e.g. `postgres`).

When upgrading from a release of IBDB earlier than 4.8.0, the above command also updates Column Optimizer triggers and stored procedures.

Important
The upgrade process will upgrade binaries to a new release while still using existing (e.g. previous release) data directories. This is accomplished by ensuring that the $IBDATA and $PGDATA parameters in the file **/etc/sysconfig/infobright/postgres** contain appropriate values to point to the correct data directory.

The upgrade process will update (and uncomment) the $IBDATA and $PGDATA parameters only when the three parameters $IBBASE, $IBDATA, and $PGDATA are still commented in the file **/etc/sysconfig/infobright/postgres.** This would imply that this is the first upgrade since a fresh install and no manual changes to the desired data directory location were made.

If the $IBBASE, $IBDATA, or $PGDATA parameters are not all commented, then this implies that either a previous upgrade process has already changed the values of $IBDATA and $PGDATA, or that manual changes to the desired data directory location were made (which the upgrade process will respect).

Configuration

Configuring Infobright (infobright.cnf file)

The Infobright configuration file is called **infobright.cnf** and is where default values for Infobright parameters are located.

Important
The **infobright.cnf** file must exist in the **ib_data** subdirectory within your Infobright installation directory or the directory you specified during installation. If it does not exist at that location, the Infobright Server will fail to start.

When Infobright is initially installed, the file **infobright.cnf** will not exist. It must be created as a manual post-installation step.

The recommended way to create the initial **infobright.cnf** file is to simply create a copy of the **infobright.cnf.sample** file, which is created on an initial install (and updated on an upgrade). The **infobright.cnf.sample** file is a text file that contains the following information:

- Instructions for how to override Infobright parameter default values, including information for which parameters are product or environment specific, and which parameters should not be overridden without first getting approval from infobright.
- Detailed descriptions for all Infobright parameters, including valid values and the Infobright default setting.

Regardless of how **infobright.cnf** is created, the following rules apply:

- Each parameter must be specified on a separate line and use the form <Parameter Name> = <Parameter Override Value>. For example:
  ```
  LogLevel = D
  ```
- The contents of the file are only read at server start-up.
- If an invalid <Parameter Name> or <Parameter Override Value> is specified, the server will fail to start.
  - **Parameters are case-sensitive and must be typed exactly as shown**
  - An invalid <Parameter Name> includes parameters that are not valid for a particular product or platform (e.g. attempting to override an Infobright DB MySQL specific parameter in IEE-Postgres)
- If a parameter is not present in the file, the default value will be used.
- Blank lines and comments (lines starting with #) are ignored.

Instructions For How To Override Infobright Parameter Default Values

The following is extracted directly from the **infobright.cnf.sample** file:

```
# This file "infobright.cnf.sample" will be refreshed (e.g., with new
# parameters, changed default values) with every release. On initial Infobright
# installation, this file (or a subset of its contents) must be copied to a
# file called "infobright.cnf" that is located in the data directory. The
# "infobright.cnf" file will be read by the Infobright engine at start-up
# to override default parameter values. On Infobright upgrades, the refreshed
# "infobright.cnf.sample" file should be reviewed to determine whether any
# additional changes should be made to the "infobright.cnf" file.
#
# The format of all parameters in this file is as follows:
#
# ##### <Parameter Name> #####
# #
# # <Parameter Description> (which may cover multiple lines)
# #
# #<Parameter Name> = <Parameter Default Value>
#
# The reason for specifying the "#<Parameter Name> = <Parameter Default Value>"
# line in this file is so that the parameter default value is communicated,
# as well as the format needed for overriding the default value.
```

```
# To override the default value one should duplicate the row, remove the
# comment character "#" at the front of the row, and specify an override value.
# The resulting row should read "<Parameter Name> = <Parameter Override Value>".
#
# Some parameters may only be applicable to IEE-MySQL, IEE-Postgres or a
# specific platform (e.g., Windows). When this is the case, it will be noted in
# the first row of the parameter which will have content of the form
# "##### <Parameter Name> (<Parameter Restriction>) #####". Note that any
# specification of an override value for a product or environment other than
# that indicated by <Parameter Restriction> will result in the Infobright server
# failing to start.
#
# Certain parameters should always be kept at their default values unless prior
# written approval has been given by Infobright. When this is the case, it will
# be noted in the two rows immediately preceding the row specifying <Parameter
# Default Value> which will have content of the form "DO NOT OVERRIDE DEFAULT
# VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT. DOING SO MAY RESULT IN
# INFOBRIGHT'S INABILITY TO SUPPORT YOU."
```

Detailed Infobright Parameter Descriptions

The following is extracted directly from the **infobright.cnf.sample** file:

KNFolder

```
##### KNFolder #####
#
# Specifies the folder where the Knowledge Grid is stored.
#
# Valid values:
#     <path> - can be either an absolute path or a path relative to the database
#              folder
#
#KNFolder = BH_RSI_Repository
```

CacheFolder

```
##### CacheFolder #####
#
# Specifies the folder where intermediate database objects are stored when
# insufficient resources are available to store them in memory.
#
# If possible, the folder should be located on a fast drive. To help reduce I/O
# bottlenecks, one can consider placing the folder on a different drive than
# the data folder.
#
# The total available size requirement of the cache folder is dependent both on
# database size and the types of queries being performed. For example,
# multi-table joins and aggregations such as COUNT(DISTINCT ...) tend to require
# a larger cache.
#
# As a general rule, it is recommended that available cache free space be 500%
# of physical RAM or 50 GB (whichever is larger).
#
# Valid values:
#     <path> - can be either an absolute path or a path relative to the database
#              folder
#
#CacheFolder = cache
```

LicenseFile

```
##### LicenseFile #####
#
# Specifies the name of the file (and optionally the path) where the License
# File is stored.
#
# Valid values:
#     <file> - name of the license file located in the database folder
#     <path> - can be either an absolute path or a path relative to the database
#              folder. The path must end with the name of the license file
#
#LicenseFile = infobright.lic
```

ServerMainHeapSize

```
##### ServerMainHeapSize #####
#
# Specifies the size (in MB) of the main memory heap (referred to as MainHeap)
# in the server process.
```

```
#
# Should normally be set to between 60-80% of physical RAM.
#
# Non-standard settings may sometimes be required. Factors to consider when
# changing the default value include the following:
#  - The 60-80% guideline is based on available RAM. Memory needed by other
#    processes that may be running on the server (including data loader) should
#    not be included as "available RAM".
#  - The setting should be as large as possible but safely smaller than the
#    amount of physical memory in the machine. If performance decreases
#    because of memory swapping by the operating system (e.g., because of
#    increased memory usage due to many heavy queries run in parallel or
#    activity of other processes), try to set a lower value.
#  - As the setting approaches 100% of installed RAM, performance degradation
#    may occur due to swapping. A setting above 100% will definitely degrade
#    performance (or cause an Out Of Memory error). To get better performance,
#    install more RAM and keep default settings.
#
# Valid values:
#     n - a positive integer 'n' specifying MainHeap size (in MB)
#
# The default value specified below is only an example. The true default value
# will be set equal to 0.75 * 'amount of RAM'
#
#ServerMainHeapSize = 6000
```

LogLevel

```
##### LogLevel #####
#
# Controls the type and amount of information written to the log file.
#
# Valid values:
#     E - errors only
#     W - warnings and errors
#     N - notices (including query execution reports), warnings, errors
#     D - debug level: all kinds of messages, including more detailed query
#         execution reports
#
```

```
# Note that writing more information to the log file can affect performance.
# The effect is likely to be small for a value of 'N' but could be significant
# for a value of 'D'.
#
#LogLevel = W
```

LogRotateSize

```
##### LogRotateSize #####
#
# Specifies the maximum size (in MB) for the current log file (infobright.log).
# When the maximum size is reached, log rotation will occur.
#
# Valid values:
#     n - a positive integer 'n' indicating maximum log size (in MB)
#
#LogRotateSize = 250
```

LogRotateFiles

```
##### LogRotateFiles #####
#
# Specifies the maximum number of archived log files that will be kept as a
# result of log rotation. Every time log rotation occurs, the archived log files
# are renamed, and the oldest archived log file potentially deleted.
#
# Valid values:
#     n - a positive integer 'n' (>=2) indicating that 'n' archived log files
#         should be kept.
#
# Note that for a particular value 'n' the following log files will exist:
#     infobright.log:
#       - the current log file
#     infobright.log.1
#       - the most recently archived log file, which is kept uncompressed
#     infobright.log.2.gz ... infobright.log.n.gz
#       - the remaining archived log files, which are compressed and zipped
#
#LogRotateFiles = 9
```

ThrottleLimit

```
##### ThrottleLimit #####
#
# Controls how many SELECT queries are allowed to run concurrently.
# Note that LOADs are not affected (i.e., not throttled) by this setting.
#
# Valid values:
#      0 - no throttling of SELECT queries is done
#      n - a positive integer 'n' specifying number of allowed concurrent SELECT
#          queries
#
# Note that a secondary use of ThrottleLimit is that the parameter
# ServerMainHeapThreshold is only in effect when ThrottleLimit > 0.

#ThrottleLimit = 0
```

AllowMySQLQueryPath (IEE-MySQL only)

```
##### AllowMySqlQueryPath (IEE-MySQL only) #####
#
# Controls whether queries that cannot be executed by the Infobright Engine for
# any reason (e.g., unsupported function, use of MyISAM table, initial attempt
# to execute in Infobright Engine fails) can instead be executed by the pure
# MySQL Engine.
#
# Valid values:
#      0 or false - query cannot be executed by pure MySQL Engine
#      1 or true  - query can be executed by pure MySQL Engine
#
#AllowMySqlQueryPath = 1
```

UseMySQLImportExportDefaults (IEE-MySQL only)

```
##### UseMySqlImportExportDefaults (IEE-MySQL only) #####
#
# Controls whether MySQL Loader/Export default values (for field delimiters,
# line terminators, etc.) should also be used for Infobright Loader/Export
# default values.
```

```
#
# Valid values:
#     0 or false - do not use MySQL Loader/Export default values
#     1 or true  - use MySQL Loader/Export default values
#
#UseMySqlImportExportDefaults = 0

######################## Import/Export Settings ###############################
#                                                                             #
# Import/export settings - a group of parameters defining how data is loaded  #
# from and exported to files or pipes. The parameters are for IEE-MySQL only  #
# and are only applicable when importing to or exporting from Infobright      #
# tables.                                                                     #
#                                                                             #
###############################################################################
```

BH_DATAFORMAT (IEE-MySQL only)

```
##### BH_DATAFORMAT (IEE-MySQL only) #####
#
# Specifies the format of data in a file/pipe from which data are loaded
# or to which data are exported.
#
# Valid values:
#     txt_variable - variable length text format
#           binary - binary format
#       infobright - compressed format produced by DLP (load only)
#            mysql - format compliant with mysql import/export format
#
# Note that when loading data, values of "txt_variable", "binary", and
# "infobright" will cause the Infobright Loader to be used while a value of
# "mysql" will cause the MySQL Loader to be used.
#
# Note that this parameter is only applicable when importing to or exporting
# from Infobright tables.
#
#BH_DATAFORMAT = txt_variable
```

BH_REJECT_FILE_PATH (IEE-MySQL only)

```
##### BH_REJECT_FILE_PATH (IEE-MySQL only) #####
#
# Specifies the path to the file where rows rejected during a load are stored.
# Rejected rows are placed into the reject file in the order they are rejected.
# The original format is preserved to allow the operator to correct and rerun
# the load for only the rejected rows.
#
# Valid values:
#       NULL - no reject file will be used
#     <path> - absolute path to a file holding rows rejected during a load
#
# Note that if BH_REJECT_FILE_PATH is set to a non-NULL value, then one and
# only one of BH_ABORT_ON_COUNT or BH_ABORT_ON_THRESHOLD must also be set to a
# non-NULL value (otherwise loads will fail).
#
# Note as well that before each load, the file specified by BH_REJECT_FILE_PATH
# must not exist.
#
# Note that this parameter is only applicable when importing to Infobright
# tables.
#
#BH_REJECT_FILE_PATH = NULL
```

BH_ABORT_ON_COUNT (IEE-MySQL only)

```
##### BH_ABORT_ON_COUNT (IEE-MySQL only) #####
#
# Specifies the total number of rejected rows required to reject an entire load.
# Upon reaching this number, the server will abort and roll back the load
# transaction.
#
# Valid values:
#     NULL - abort on the first rejected row (nothing written to a reject file)
#       -1 - never abort (and write all rejected rows to a reject file)
#        0 - this is a synonym of "NULL"
#        n - a positive integer "n" indicating the number of rejected rows that
#            will cause the entire load to be aborted
```

```
#
# Note that if BH_ABORT_ON_COUNT is set to a non-NULL value, then:
#   - BH_REJECT_FILE_PATH must also be set to a non-NULL value
#   - BH_ABORT_ON_THRESHOLD must be set to NULL
#
# Note that this parameter is only applicable when importing to Infobright
# tables.
#
#BH_ABORT_ON_COUNT = NULL
```

BH_ABORT_ON_THRESHOLD (IEE-MySQL only)

```
##### BH_ABORT_ON_THRESHOLD (IEE-MySQL only) #####
#
# Specifies the ratio of rejected rows to total processed rows required to
# reject an entire load. Upon reaching this threshold, the server will abort and
# roll back the load transaction.
#
# Valid values:
#     NULL - abort on the first rejected row (nothing written to a reject file)
#        0 - this is a synonym of "NULL"
#        r - a decimal "r" between 0 and 1 indicating the ratio of rejected
#            rows to total processed rows that will cause the entire load to be
#            aborted.
#
# Note that if BH_ABORT_ON_THESHOLD is set to a non-NULL value, then:
#   - BH_REJECT_FILE_PATH must also be set to a non-NULL value
#   - BH_ABORT_ON_COUNT must be set to NULL
#
# Note that this parameter is only applicable when importing to Infobright
# tables.
#
#BH_ABORT_ON_THRESHOLD = NULL
```

BH_LOAD_ACCEPT_MISSING_COLUMNS (IEE-MySQL only)

```
##### BH_LOAD_ACCEPT_MISSING_COLUMNS (IEE-MySQL only) #####
#
# Controls whether a load can accept input data that has fewer columns than
# the destination table being loaded.
```

```
#
# Missing columns are assumed to be at the end of the table.
#  - for missing columns that allow NULL, NULL will be inserted
#  - for missing columns that do not allow NULL, the same rules as standard
#    MySQL uses for loads with missing columns will be followed
#
# Valid values:
#     0 or false - do not accept input data with fewer columns
#     1 or true  - accept input data with fewer columns
#
# Note that this parameter is only applicable to input files using the
# "infobright" data format (BH_DATAFORMAT = infobright).
#
# Note that this parameter is only applicable when importing to Infobright
# tables.
#
#BH_LOAD_ACCEPT_MISSING_COLUMNS = 0
```

BH_NULL (IEE-MySQL only)

```
##### BH_NULL (IEE-MySQL only) #####
#
# Specifies how NULLs are represented in exported data.
#
# Note that this parameter only has meaning for the "txt_variable" data format
# (BH_DATAFORMAT = txt_variable).
#
# Valid values:
#     <string> - any character string
#
# Note that when specifying a value for <string>, no wrapping quotation marks
# should be used (e.g., if one wants to set string to "na", then one should
# specify "BH_NULL = na".
#
# Note that the default value "BH_NULL =" sets the value to an empty string.
#
# Note that this parameter is only applicable when exporting from Infobright
# tables.
#
```

```
##BH_NULL =

##############################################################################
#                                                                            #
# NOTE THAT ALL REMAINING PARAMETERS REQUIRE PRIOR WRITTEN APPROVAL FROM     #
# INFOBRIGHT BEFORE OVERRIDING DEFAULT VALUES                                #
#                                                                            #
##############################################################################
```

FET

```
##### FET #####
#
# Controls whether Function Execution Times are reported in logs.
# Switching it on ('1') allows low-level profiling of slow queries, but also
# causes performance degradation.
# The FET report is generated on server shutdown, unless FETInterval is also
# set.
# In order to see FET report in logs, LogLevel must also be set to 'D' or 'N'.
#
# Valid values:
#     0 or false - FET reporting is switched off
#     1 or true  - FET reporting is switched on (will cause performance
#                  degradation)
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#FET = 0
```

FETInterval

```
##### FETInterval #####
#
# Controls how often FET reports are generated.
#
# Note that this parameter only has meaning when FET reporting is switched on
# (FET = 1)
#
# Valid values:
#     0 - indicates that FET reports should be generated only on server shutdown
#     n - a positive integer 'n' indicating the interval (in seconds) between
#         when FET reports are generated
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#FETInterval = 0
```

PrefetchThreads

```
##### PrefetchThreads #####
#
# Controls the number of threads used to service prefetch requests (parallel
# data pack decompression).
#
# Valid values:
#     n - a positive integer 'n' specifying the number of threads to use
#
# The default value specified below is only an example. The true default value
# will be set equal to the minimum(16, 0.75 * 'number of CPU cores')
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#PrefetchThreads = 8
```

PrefetchQueueLength

```
##### PrefetchQueueLength #####
#
# Controls the number of prefetch requests queued in the prefetcher. Any further
# requests are discarded.
#
# Valid values:
#     n - a positive integer 'n' specifying the maximum number of prefetch
#         requests that will be queued
#
# The default value specified below is only an example. The true default value
# will be set equal to 3 * 'value of PrefetchThreads'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
```

```
#PrefetchQueueLength = 24

####################### Parallel Threads Settings ##############################
#                                                                              #
# Parallel threads settings - a group of parameters defining maximal number of #
# threads available in one query for parallel execution of algorithms. Note    #
# that actual CPU allocation depends on a number of physical cores, data size, #
# algorithmic constraints, other active sessions, etc. Set these values to 0   #
# or 1 to disable parallel execution in particular algorithms. This            #
# functionality does not affect parallel data pack decompression (prefetching) #
# nor rough sorting of join results.                                           #
#                                                                              #
################################################################################
```

ParallelScanThreads

```
##### ParallelScanThreads #####
#
# Controls the maximum number of threads used for parallel execution of WHERE
# conditions.
#
# Valid values:
#     0 - one thread does all the work sequentially
#     1 - one thread does all the work sequentially (same definition as '0')
#     n - a positive integer 'n' specifying the maximum number of threads that
#         may be used in parallel
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ParallelScanThreads = 1024
```

ParallelJoinThreads

```
##### ParallelJoinThreads #####
#
# Controls the maximum number of threads used for parallel execution of JOINs.
#
# Valid values:
#     0 - one thread does all the work sequentially
#     1 - one thread does all the work sequentially (same definition as '0')
#     n - a positive integer 'n' specifying the maximum number of threads that
#         may be used in parallel
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ParallelJoinThreads = 1024
```

ParallelAggrThreads

```
##### ParallelAggrThreads #####
#
# Controls the maximum number of threads used for parallel execution of
# aggregations (e.g., GROUP BY, SELECT DISTINCT).
#
# Valid values:
#     0 - one thread does all the work sequentially
#     1 - one thread does all the work sequentially (same definition as '0')
#     n - a positive integer 'n' specifying the maximum number of threads that
#         may be used in parallel
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ParallelAggrThreads = 1024
```

ParallelSortThreads

```
##### ParallelSortThreads #####
#
# Controls the maximum number of threads used for parallel execution of
# ORDER BY sorting.
#
# Valid values:
#     0 - one thread does all the work sequentially
#     1 - one thread does all the work sequentially (same definition as '0')
#     n - a positive integer 'n' specifying the maximum number of threads that
#         may be used in parallel
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ParallelSortThreads = 1024
```

LoaderSaveThreads

```
##### LoaderSaveThreads #####
#
# Controls the maximum number of threads used by data loader when compressing
# and saving data.
#
# Valid values:
#     0 - the main thread does all the work sequentially
#     n - a positive integer 'n' specifying the maximum number of threads that
#         may be used in parallel in addition to the main thread
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#LoaderSaveThreads = 16
```

ParallelScanDPsAtOnce

```
##### ParallelScanDPsAtOnce #####
#
# Controls how many consecutive data packs are given for one parallel WHERE
# condition execution thread.
#
# Valid values:
#     n - a positive integer 'n' specifying number of consecutive data packs
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
# Default for Linux:
#ParallelScanDPsAtOnce = 1
# Default For Windows:
#ParallelScanDPsAtOnce = 5
```

ParallelScanDPsPerThread

```
##### ParallelScanDPsPerThread #####
#
# An internal calculation involving the value of this parameter and the total
# number of data packs being evaluated controls how many parallel WHERE
# condition threads are actually launched.
```

```
#
# Valid values:
#     n - a positive integer 'n'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ParallelScanDPsPerThread = 10
```

SyncBuffers

```
##### SyncBuffers #####
#
# Controls whether to flush disk buffers after each commit.
# Flushing of disk buffers after each commit will cause performance degradation.
#
# Valid values:
#     0 or false - disk buffers are not flushed after each commit
#     1 or true  - disk buffers are flushed after each commit (will cause
#                  performance degradation)
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#SyncBuffers = 0
```

ThrottleScheduler

```
##### ThrottleScheduler #####
#
# For throttled queries, controls whether the query to resume from the waiting
# queues is selected using a system default policy or by a FIFO scheduler.
#
# Valid values:
#     0 - select from queue using a FIFO scheduler
#     1 - select from queue using a system default policy
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ThrottleScheduler = 0
```

ServerMainHeapThreshold

```
##### ServerMainHeapThreshold #####
#
# Specifies the minimum amount of available memory (in MB) on MainHeap that
# must exist so that a query is not throttled and can start. If the minimum
# amount of memory is not available, the query is queued until an already
# running query finishes.
# Note that this parameter is effective only when ThrottleLimit > 0.
#
# Valid values:
#     n - a non-negative integer 'n' less than size of MainHeap
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ServerMainHeapThreshold = 5
```

MemoryHardLimit

```
##### MemoryHardLimit #####
#
# Controls whether memory allocation should occur from system heap when MainHeap
# is full.
#
# Valid values:
#     0 or false - Allow additional memory allocation from system heap
#     1 or true  - Do not allow additional memory allocation from system heap
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#MemoryHardLimit = 0
```

MemoryLargeTempPercentage

```
##### MemoryLargeTempPercentage #####
#
# Specifies the percentage of MainHeap reserved for large (>16 MB) memory blocks.
#
# Valid values:
#     n - a positive integer 'n' between 0 and 100
```

```
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#MemoryLargeTempPercentage = 20
```

MemoryScaleReduction

```
##### MemoryScaleReduction #####
#
# Controls how aggressive queries are in allocating memory for their own
# internal buffers. Allocating less memory may slowdown query execution
# time, but may also prevent possible system heap over usage and swapping.
#
# Valid values:
#      0 - default behaviour
#      n - a positive integer indicating a reduction of memory allocation
#          aggressiveness of 'n' levels, where each level would result in about
#          10-20% less memory being allocated (depending on size of MainHeap)
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#MemoryScaleReduction = 0
```

KNLevel

```
##### KNLevel #####
#
# Controls whether Knowledge Grid should be used in queries and updated on
# data inserts / loads.
#
# Valid values:
#      0 - disable the use and updating of the Knowledge Grid
#      1 - enable the use and updating of the Knowledge Grid
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#KNLevel = 1
```

SpliceSize

```
##### SpliceSize #####
```

```
#
# Specifies the number of DPNs kept in one memory array (called a splice).
#
# Valid values:
#     n - a positive integer 'n' that is a valid power of 2 (e.g., 2, 4, 8, 16,
#         32, 64, 128, 256, 512, 1024, etc.)
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#SpliceSize = 128
```

ConnectTimeout (IEE-Postgres only)

```
##### ConnectTimeout (IEE-Postgres only) #####
#
# Specifies the number of seconds the system will wait for OS-level TCP
# connections to be established.
#
# Valid values:
#     n - a positive integer 'n'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ConnectTimeout = 5
```

HandShakeTimeout (IEE-Postgres only)

```
##### HandShakeTimeout (IEE-Postgres only) #####
#
# Specifies the number of seconds the system will wait for each reply when
# setting up application level connections during the initialization dialog.
#
# Examples of application level connections are postgres backend to ibengine,
# and (for multi-machine only) ibengine to ibengine.
#
# Valid values:
#     n - a positive integer 'n'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
```

```
#HandShakeTimeout = 15
```

PeerCommitTimeout (IEE-Postgres Multi-Machine only)

```
##### PeerCommitTimeout (IEE-Postgres Multi-Machine only) #####
#
# Specifies the number of seconds the system will wait for confirmation from
# all active ibnodes when statements like COMMIT (also implicit auto-commit such
# as when a load finishes), DROP TABLE/COLUMN, and RENAME TABLE are executed on
# a particular ibnode. When confirmation from all ibnodes is received,
# processing is finalized. If confirmation from all ibnodes is not received
# before PeerCommitTimeout is reached, then the statement fails and is rolled
# back.
#
# Valid values:
#     n - a positive integer 'n'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#PeerCommitTimeout = 120

###################### ClusterMulticast Settings #############################
#                                                                            #
# ClusterMulticast settings - a group of parameters controlling behaviour of #
# the cluster multicast feature that is responsible for keeping the cluster in #
# a consistent state. Control packets are periodically sent and received so  #
# that each cluster node is aware which other nodes are on-line and working. #
#                                                                            #
##############################################################################
```

ClusterMulticastAddress (IEE-Postgres Multi-Machine only)

```
##### ClusterMulticastAddress (IEE-Postgres Multi-Machine only) #####
#
# The multicast ip address to which control packets are sent.
#
# Valid values:
#     <ip address> - Any valid IPv4 multicast address
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ClusterMulticastAddress = 239.0.0.2
```

ClusterMulticastPort (IEE-Postgres Multi-Machine only)

```
##### ClusterMulticastPort (IEE-Postgres Multi-Machine only) #####
#
# The port to which control packets are sent.
#
# Valid values:
#     <port> - Any valid port
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ClusterMulticastPort = 11166
```

ClusterMulticastInterval (IEE-Postgres Multi-Machine only)

```
##### ClusterMulticastInterval (IEE-Postgres Multi-Machine only) #####
#
# Specifies how often (in seconds) each node should send a control packet
#
# Valid values:
#     n - a positive integer 'n'
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ClusterMulticastInterval = 5
```

ClusterMulticastPacketsPerCheck (IEE-Postgres Multi-Machine only)

```
##### ClusterMulticastPacketsPerCheck (IEE-Postgres Multi-Machine only) #####
#
# Specifies the number of consecutive missing packets to wait before declaring
# a node disconnected.
#
# Valid values:
#     n - a positive integer 'n' (>=2)
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
```

```
#ClusterMulticastPacketsPerCheck = 3
```

ClusterMulticastFilterAddresses (IEE-Postgres Multi-Machine only)

```
##### ClusterMulticastFilterAddresses (IEE-Postgres Multi-Machine only) #####
#
# Controls whether incoming multicast packets from ip addresses outside of those
# defined in the cluster configuration are filtered or not.
#
# Valid values:
#      0 or false - do not filter ip addresses
#      1 or true  - filter ip addresses
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#ClusterMulticastFilterAddresses = 0
```

CoreDump (IEE-Postgres Windows only)

```
##### CoreDump (IEE-Postgres Windows only) #####
#
# Controls whether to generate a core dump file in the database directory when
# a server crash occurs.
#
# Valid values:
#      0 or false - do not generate a core dump file when a server crash occurs
#      1 or true  - generate a core dump file when a server crash occurs
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#CoreDump = 0
```

PushDown (IEE-MySQL only)

```
##### PushDown (IEE-MySQL only) #####
#
# Controls whether a query involving Infobright tables that is partially
# executed by pure MySQL Engine (e.g., a join with a MYISAM table) will still
# "push down" WHERE condition execution to the Infobright Engine.
#
# Valid values:
```

```
#      0 or false - push down to Infobright Engine will not occur
#      1 or true  - push down to Infobright Engine will occur
#
# DO NOT OVERRIDE DEFAULT VALUE WITHOUT PRIOR WRITTEN APPROVAL FROM INFOBRIGHT.
# DOING SO MAY RESULT IN INFOBRIGHT'S INABILITY TO SUPPORT YOU.
#PushDown = 1
```

Cross-Reference Of Pre-4.8.0 Parameters To Current Parameters

Beginning with release 4.8.0, all default value overrides for Infobright parameters are located in the **ib_data** directory file **infobright.cnf**. Prior to release 4.8.0, default value overrides were located in the now obsolete **ib_data** directory files **.infobright** and **brighthouse.ini**.

In release 4.8.0, some new parameters were added, some pre-4.8.0 parameters were removed, and some parameters were renamed.

The following tables provide a cross-reference of pre-4.8.0 parameters to current (post-4.8.0) parameters. This is especially useful when upgrading from a pre-4.8.0 release to 4.8.0 (or later) as it provides an indication on how default overrides in the **.infobright** and **brighthouse.ini** files should be converted into default overrides in the **infobright.cnf** file.

Parameters That are New in Release 4.8.0 (or later)

Current Parameter Name (in infobright.cnf)	Previous Parameter Name	Brief Description (details in infobright.cnf)
LicenseFile	n/a	Specifies path/name of newly required License file
LogLevel	n/a	Controls how much information is written to logs. Similar to obsolete ControlMessages parameter
LogRotateSize	n/a	Specifies how large the log file can be before its rotated and archived
LogRotateFiles	n/a	Specifies how many log archive files are kept
FET	n/a	Controls whether Function Execution Times are logged **Consult Infobright to update.**
FETInterval	n/a	Specifies how often FET reports are generated **Consult Infobright to update.**

Parameters That Were Previously Specified in brighthouse.ini

Current Parameter Name (in infobright.cnf)	Previous Parameter Name (in brighthouse.ini)	Brief Description (details in infobright.cnf)

KNFolder	KNFolder	Specifies folder where Knowledge Grid is stored
CacheFolder	CacheFolder	Specifies folder where temporary objects are stored
ServerMainHeapSize	ServerMainHeapSize	Specifies the size (in MB) of the main memory heap
AllowMySqlQueryPath **IEE-MySQL only**	AllowMySqlQueryPath	Controls whether queries can be redirected to MySQL Engine
UseMySqlImportExportDefa ults **IEE-MySQL only**	UseMySqlImportExportDefault s	Controls whether MySQL or Infobright Loader/Export defaults should be used
KNLevel	KNLevel	Controls whether Knowledge Grid should be used in queries and updated on inserts / loads **Consult Infobright to update.**
PushDown **IEE-MySQL only**	Pushdown	Controls how MySQL engine and Infobright Engine interact during query execution **Consult Infobright to update.**
CoreDump **IEE-Postgres Windows only**	CoreDump	Controls whether to generate a core dump when server crashes **Consult Infobright to update.**
n/a	ControlMessages	Parameter removed in 4.8.0 Note: New LogLevel parameter provides similar functionality
n/a	LoaderMainHeapSize	Parameter removed in 4.8.0
n/a	Autoconfigure	Parameter removed in 4.8.0
n/a	ServerCompressionHeapSize	Parameter removed in 4.8.0
n/a	LastPackCompression	Parameter removed in 4.8.0

Parameters That Were Previously Specified in .infobright

Current Parameter Name (in infobright.cnf)	Previous Parameter Name (in .infobright)	Brief Description (details in infobright.cnf)
ThrottleLimit	<ThrottleLimit>	Controls how many SELECT queries can run concurrently
PrefetchThreads	<Prefetch><Threads>	Controls the number of threads used for prefetch requests **Consult Infobright to update.**
PrefetchQueueLength	<Prefetch><QueueLength>	Controls the number of queued prefetch requests **Consult Infobright to update.**

ParallelScanThreads	<parallelscan><maxthreads>	Controls the maximum number of threads used for parallel execution of WHERE conditions **Consult Infobright to update.**
ParallelJoinThreads	<paralleljoin><maxthreads>	Controls the maximum number of threads used for parallel execution of JOINs **Consult Infobright to update.**
ParallelAggrThreads	<parallelaggr><maxthreads>	Controls the maximum number of threads used for parallel execution of aggregations **Consult Infobright to update.**
ParallelSortThreads	<parallelsort><maxthreads>	Controls the maximum number of threads used for parallel execution of ORDER BY sorting **Consult Infobright to update.**
LoaderSaveThreads	<LoaderSaveThreadNumber>	Controls the maximum number of threads used by data loader to compress and save data **Consult Infobright to update.**
ParallelScanDPsAtOnce	<parallelscan><noDPsAtOnce>	Controls how many data packs given to one parallel WHERE condition execution thread **Consult Infobright to update.**
ParallelScanDPsPerThread	<parallelscan><minDPsPerThread>	Controls how many parallel WHERE condition execution thread are actually launched **Consult Infobright to update.**
SyncBuffers	<sync_buffers>	Controls whether or not to flush disk buffers after every commit **Consult Infobright to update.**
ThrottleScheduler	<Throttle><Scheduler>	Controls in what order throttled queries are resumed **Consult Infobright to update.**
ServerMainHeapThreshold	<MemoryManagement><ServerMainHeapThreshold>	Specifies the minimum amount of MainHeap memory that must exist to prevent query throttling **Consult Infobright to update.**
MemoryHardLimit	<MemoryManagement><hardlimit>	Controls whether memory can be allocated from system heap when MainHeap is full **Consult Infobright to update.**
MemoryLargeTempPercentage	<MemoryManagement><largetempratio>	Specifies the percentage of MainHeap reserved for large memory blocks. Note: valid value changed from fraction (e.g. 0.2) to % (e.g. 20) **Consult Infobright to update.**

MemoryScaleReduction	<MemoryScaleReduction>	Controls how aggressive queries are in allocating memory for internal buffers **Consult Infobright to update.**
SpliceSize	<SpliceSize>	Specifies the number of DPNs kept in one memory array **Consult Infobright to update.**
n/a	<Prefetch><Depth>	Parameter removed in 4.8.0
n/a	<HugeFileDir>	Parameter removed in 4.8.0
n/a	<ClusterSize>	Parameter removed in 4.8.0
n/a	<CachingLevel>	Parameter removed in 4.8.0
n/a	<BufferingLevel>	Parameter removed in 4.8.0
n/a	<MemoryManagement><policy >	Parameter removed in 4.8.0
n/a	<MemoryManagement><release policy>	Parameter removed in 4.8.0

Parameters That Were Previously Only Available at Infobright DB MySQL Command Line

Current Parameter Name (in infobright.cnf)	Previous Parameter Name (at command line)	Brief Description (details in infobright.cnf)
BH_DATAFORMAT **Infobright DB MySQL only**	@BH_DATAFORMAT	Specifies the data format for imports and exports
BH_REJECT_FILE_PATH **Infobright DB MySQL only**	@BH_REJECT_FILE_PATH	Specifies the path/file for storing rows rejected by a load
BH_ABORT_ON_COUNT **IEE-MySQL only**	@BH_ABORT_ON_COUNT	Specifies the number of allowed rejected rows in a load
BH_ABORT_ON_ THRESHOLD **Infobright DB MySQL only**	@BH_ABORT_ON_ THRESHOLD	Specifies the fraction of allowed rejected rows in a load
BH_LOAD_ACCEPT_ MISSING_COLUMNS **IEE-MySQL only**	@BH_LOAD_ACCEPT_ MISSING_COLUMNS	Controls whether a load can accept input data with fewer rows than defined in table
BH_NULL **Infobright DB MySQL only**	@BH_NULL	Specifies how NULLs are represented in exported data
n/a	@BH_THROTTLE	Parameter removed in 4.8.0
n/a	@BH_PARALLEL_AGGR	Parameter removed in 4.8.0

n/a	@BH_IBEXPRESSIONS	Parameter removed in 4.8.0

Infobright Specific PostgreSQL Parameters

PostgreSQL parameters are managed through the **/pg_data/postgresql.conf** file. The following Infobright specific PostgreSQL parameters have been added:

Infobright Specific PostgreSQL Parameters

Parameter Name	Default Value	Description
allow_postgres_query_path	true	Controls whether queries that cannot be executed by the Infobright Server will be executed by the PostgreSQL engine
ibbindir	<installation root>/bin	Specifies where postmaster will search for ibengine binary
ibdatadir	<installation root>/ib_data	Shows the ibengine datadir absolute path
ibport	9205	Sets the TCP port the ibengine server listens on

Note

The reason the above parameters are not instead included in the infobright.cnf file (i.e. where other Infobright parameters are specified) is because the parameters are used at the ibadapter level (i.e. before a query even reaches the ibengine).

3. Using the Infobright Server

The Infobright License File

Beginning with release 4.8.0, every instance of the Infobright Server requires that a valid license file exists.

- The license file is normally called **infobright.lic** and is located in the **ib_data** directory. Both the name and location of the license file can be changed by updating the LicenseFile configuration parameter.
- Depending on your agreement with Infobright, a new license file may occasionally be required.
- Depending on your agreement with Infobright, either the same license file can simply be copied to each instance of the Infobright Server, or a unique license file for each instance of the Infobright Server will be required.
- To obtain a valid license file, please contact Infobright Support.

- Important
  Without a valid license file installed, either the Infobright Server will fail to start, or reduced functionality will be experienced.

To view the contents of the license currently in use on the Infobright Server, execute the following command:

```
postgres=# show infobright license;
```

Starting and Stopping the Infobright Server

On Windows, the Windows Install Wizard automatically creates Infobright as a Windows Service, which allows the Infobright Server to be started and stopped automatically when you boot or shutdown Windows.

- To manually start the Infobright Server, from the Windows Start Menu run:

```
Start/All Programs/Infobright/IEE Postgres Edition Start Server
```

- To manually stop the Infobright Server, from the Windows Start Menu run:

```
Start/All Programs/Infobright/IEE Postgres Edition Stop Server
```

On Linux, use the following procedures to start IBDB for PostgreSQL if it is not already started upon boot. Note that root privileges are typically required.

- To start the Infobright Server, run:

```
/etc/init.d/infobright-iee-postgres start
```

- To stop the Infobright Server, run:

```
/etc/init.d/infobright-iee-postgres stop
```

Working with the Infobright Server

This section describes the use of the provided client to interact with the Infobright Server using the PostgreSQL interface. You can also use a PostgreSQL compatible client in a similar manner.

On Windows:

- To connect to the Infobright command line interface, run:

```
Start/All Programs/Infobright/IEE Postgres Edition Command Line Client
```

On Linux:

- If you used the standard install locations, enter the following command to connect to Infobright:

```
cd /usr/local/infobright-products/postgres
./bin/psql -U postgres -d postgres
```

- If you used a different install location, modify the above command to point to the correct directory.

To enable remote connections to Infobright, refer to **pg_hba.conf** in the **pg_data** directory for authorized IP tables used for permitting or deny foreign connections.

Note

Infobright can be used with most Business Intelligence tools and any PostgreSQL GUI client. Simply point to the IP address and socket number for the Infobright Server and logon using any user credentials that have been set up. Note that compatibility with BI tools is not guaranteed with every BI engine.

Checking the Infobright Server Version

You can use the following method to check the version of the Infobright Server.

- After connecting to the Infobright administrator account, enter the following command at the PostgreSQL command prompt:

```
Postgres=# show variables like 'IBEngineRevision';


       Name        |          Value          | Description
------------------+------------------------+--------------
 IBEngineRevision | IEE_4.8.0_r32869_32871 | (Read only)
(1 row)
```

About Log Files

The Infobright Log (infobright.log)

The Infobright log file is called **infobright.log** and is located within the **ib_data** subdirectory. Uses of the Infobright log file include the following:

- Provide information to help diagnose and monitor system performance and operation
- Log events for both the Infobright Server and Infobright Adapter
- Log events for auxiliary processes such as Loader
- Provide information regarding configuration parameter settings and licensing
- Log Hardware/OS/Timezone settings and version information on start.

Note

Lines too long to be completely written to the log file will be truncated. If this occurs, the characters **[…]** will be appended to the end the row in the log file.

Infobright.log Example

The following is an example showing some of the log lines that can appear in **infobright.log**.

Note

While the below example has been extracted from an actual **infobright.log** file, it does not reflect the actual contents of the file. Many lines were removed in order to show as many different types of log lines as possible.

```
IB-IBE-00598    N       2015-05-02.06:59:26     centos66-25     0.0
mhs:8922        config:"option LogRotateFiles=9 (Def)"

IB-IBE-002da    W       2015-05-02.06:59:26     centos66-25     0.0
warning:"Cachefolder cache does not exist. Trying to create it"

IB-IBE-00743    E       2015-05-14.07:06:23     centos66-25     0.0
error:"Caught signal SIGTERM; shutting down ibengine"

IB-IBE-002c1    N       2015-05-14.07:06:23     centos66-25     0.0
control:"Infobright engine shut down"

IB-IBE-00754    D       2015-06-18.06:25:28     centos66-25     3.1
sfm:10956       uf:4    et:3178.45      nrq:0   cpu:0.0 execlog_deb:"Worker
received command 'Copy from'"

IB-IBE-00765    N       2015-06-18.06:25:28     centos66-25     3.1
sfm:10956       uf:4    et:3178.45      nrq:0   cpu:0.0 execlog:"COPY FROM
to table 'postgres/public/t_number' started"

IB-BHL-00e16    E       2015-06-18.06:25:28     centos66-25     0.0
error:"Wrong data or column definition. Row: 1, field: 1"

IB-BHL-00470    N       2015-06-18.06:25:28     centos66-25     0.0
sfm:10954       uf:4    et:0.00 nrq:0   cpu:0.0 execlog:"Loading finished"

IB-PGA-009ed    E       2015-06-18.06:25:28     centos66-25     0.0
error:"Exception caught in IBCopyFrom"

IB-IBE-00551    N       2015-06-18.06:44:39     centos66-25     3.2
sfm:10955       uf:4    mhs:8922        nos:2   nrq:1   cpu:0.0 sysm:11897
ncores:8        query_start:"select * from t_number;"

IB-IBE-004dd    D       2015-06-18.06:44:39     centos66-25     3.2
query_comp:"CHARACTER SET: iso-8859-1,  DEFAULT COLLATION: binary"

IB-IBE-00552    D       2015-06-18.06:44:39     centos66-25     3.2
sfm:10955       uf:4    et:0.00 nrq:1   cpu:0.0 execlog_deb:"Table
'postgres.public.t_number' (4); packrows: 1 (first 0 deleted); rows: 3; avg.
packrow size: 3; avg. delete ratio: 0%"

IB-IBE-00556    N       2015-06-18.06:44:39     centos66-25     3.2
sfm:10955       uf:12   et:0.00 nos:2   nrq:1   cpu:0.0 query_done:"---
Query time: 0.00 s.  Rows returned: 3.  Packs loaded: 2. ---"

IB-PGA-00bc9    D       2015-06-18.06:44:39     centos66-25     0.0
sfm:10955       uf:4    et:0.00 nrq:0   cpu:0.0 execlog_deb:"commit
transaction id 0"
```

Structure of an infobright.log line

Each line written to **infobright.log** has the following format:

```
IB-<MODNAME>-<MSGNUM><tab><LEVEL><tab><TIMESTAMP><tab><HOSTNAME><tab>
<SESSIONID><tab>{<FIELDNAME>:<fldvalue><tab>}<MESSAGEKIND>:<msgvalue>
```

where the following field substitution is performed:

- **`<MODNAME>`** - infobright module that created the log line. Can have one of the following values:
  - **`INV`** – invalid (should normally never appear)
  - **`MYS`** – mysqld
  - **`BHL`** – bhloader
  - **`ICM`** – infobright consistency manager
  - **`DAT`** – dlp
  - **`MIG`** – ibmigrator
  - **`IBE`** – ibengine
  - **`EMI`** – ibextmigrator
  - **`PGA`** – postgres ibadapter
  - **`LIC`** – license tool
- **`<MSGNUM>`** - a 5 digit hexadecimal message number used to uniquely identify the source of the log line
- **`<tab>`** - a "tab" in the log line display to separate fields
- **`<LEVEL>`** - log level associated with a particular value of **`<MESSAGEKIND>`**. Can have one of the following values:
  - **`E`** – error
  - **`W`** – warning
  - **`N`** – notice
  - **`D`** – debug
- **`<TIMESTAMP>`** - a timestamp for the log line in YYYY-MM-DD.hh:mm:ss format
- **`<HOSTNAME>`** - server host name or ip address
- **`<SESSIONID>`** - client session number
- **`<FIELDNAME>`** - depending on the value of **`<MESSAGEKIND>`,** zero or more of the following field names may occur:
  - **`sfm`** – available (free) system memory
  - **`uf`** – ib (infobright) unfreeable memory
  - **`mhs`** – ib (infobright) main heap size
  - **`et`** – execution time (seconds)
  - **`nos`** – number of sessions (including idle)
  - **`nrq`** – number of running queries (being currently executed)
  - **`cpu`** – system cpu usage, number of busy cores
  - **`sysm`** – system installed memory
  - **`ncores`** – number of cpus installed
- **`<fldvalue>`** - a value associated with a specific occurrence of **`<FIELDNAME>`**
- **`<MESSAGEKIND>`** - type of information being logged in this log line:
  - **`error`** – errors, usually causing query (or other operation) to stop
  - **`warning`** – warnings, including e.g. switching to MySQL/PG query path
  - **`control`** – general server messages: start/stop
  - **`execlog`** – query execution log (main steps and statistics).
  - **`execlog_deb`** – more detailed execution logs
  - **`querystart`** – starting a query (contain a short version of the SQL statement).
  - **`querydone`** – end of query, summarizing statistics
  - **`querycomp`** – query compilation steps
  - **`config`** – configuration parameter values, usually displayed at server startup
  - **`config_deb`** – more detailed configuration parameters

  - `license` – license information and status
  - `fet` –function execution time statistics
- `<msgvalue>` - actual message being logged in this log line

As mentioned above, depending on the value of **`<MESSAGEKIND>`**, zero or more occurrences of (different) **`<FIELDNAME>`** values may be written on a log line. The following table shows the relationship (where "Yes" indicates that a particular value of **`<FIELDNAME>`** will be written):

Relationship of <MESSAGEKIND> to <FIELDNAME>

<MESSSAGEKIND>	<FIELDNAME>								
	sfm	uf	mhs	et	nos	nrq	cpu	sysm	ncores
error									
warning									
control									
execlog	Yes	Yes		Yes		Yes	Yes		
execlog_deb	Yes	Yes		Yes		Yes	Yes		
querystart	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
querydone	Yes	Yes		Yes	Yes	Yes	Yes		
querycomp									
config			Yes						
config_deb			Yes						
license									
fet	Yes	Yes				Yes	Yes		

Other Postgres Logs

PostgreSQL specific logging can be managed through the **postgresql.conf** file.

- To enable, the **logging_collector** value in **/pg_data/postgresql.conf** must be set to ON and left uncommented. The **log_directory** and **log_filename** variables should also be uncommented.
- Log files with the format **postgresql-%Y-%m-%d_%H%M%S.log** will by default be written to the **pg_data/pg_log** directory.

Log Rotation

Rotation functionality is provided for the infobright.log by utilizing the following configuration parameters in the **infobright.cnf** file.

- **LogRotateSize** - specifies the maximum size (in MB) for the current log file (infobright.log). When the maximum size is reached, log rotation will occur.
- **LogRotateFiles** - specifies the maximum number of archived log files that will be kept as a result of log rotation.

For more information regarding these parameters, see “LogRotateSize” on page and “LogRotateFiles” on page .

As a result of log rotation, the following log files can exist in the **ib_data** directory:

- **infobright.log** – the current log file
- **infobright.log.1** – the most recently archived log file, which is kept uncompressed
- **infobright.log.2.gz … infobrigght.log.n.gz** – the remaining archived log files, which are compressed and zipped.

Every time log rotation occurs, the archived log files are renamed. For example

- **infobright.log** will be renamed **infobright.log.1**
- **infobright.log.1** will be (compressed and zipped and) renamed to **infobright.log.2.gz**
- **infobright.log.2.gz** will be renamed **infobright.log.3.gz**
- etc.

Note

Depending on the number of archived log files that exist, the oldest archived log file may be deleted when log rotation occurs.

For example, if **LogRotateFiles = 10**, then log rotation would cause **infobright.log.9.gz** to be renamed to **infobright.log.10.gz**, and any previously existing **infobright.log.10.gz** file would be deleted.

Changing the infobright.log Log Level

The amount of information written to **infobright.log** can be controlled by setting the appropriate log level. The following log levels are possible:

Log Levels

Level	Description
E	Log errors only
W	Log warnings and errors
N	Log notices (including query execution reports), warnings, and errors
D	Debug level: Log all kinds of messages, including more detailed query execution reports

Note

In general, more detail in the log may have an impact on performance; it is recommended that you find and use the setting that strikes the best balance for you in terms of performance versus log details.

The log level can either be changed statically (i.e. at startup) or dynamically. Changing the log level dynamically is useful when you want to temporarily change logging level (e.g. for trouble-shooting purposes) without requiring a restart.

Changing the Log Level at Server Start-up Using infobright.cnf

The log level is initially set at Infobright Server start-up based on the value of the "LogLevel" parameter in the **infobright.cnf** file. Any modifications made to **infobright.cnf** will take effect the next time the Infobright Server is restarted. For more information, see "LogLevel" on page .

Changing the Log Level Dynamically Using the LogLevel and ses_LogLevel Variables

At Infobright Server Start-up, the setting of the global variable **LogLevel** is initialized to the value of the "LogLevel" parameter that was specified in the **infobright.cnf** file. When a Postgres client is launched, the setting of the session variable **ses_LogLevel** is not initialized. A Postgres client will always use the session variable **ses_LogLevel** when it has a value. Otherwise it will use the global variable **LogLevel**.

The value of the session variable **ses_LogLevel** can be modified by entering a command such as the following:

```
postgres=# set ibengine ses_LogLevel = 'N';
```

- The above example changes the session setting for the current client session to "N". The complete list of valid values are: "E", "W", "N" and "D".
- Making the above change will immediately affect what is written to infobright.log for the current client session, but will not affect other client sessions that exist or are launched in the future.

Note that the value of the global variable **LogLevel** can also be modified by entering a command such as the following:

```
postgres=# set ibengine LogLevel = 'N';
```

- The above example changes the global setting to "N". The complete list of valid values are "E", "W", "N" and "D".
- Making the above change will not affect what is written to **infobright.log** for the current or any other existing client sessions that have initialized the value of **ses_LogLevel**.
- However, making the above change will immediately affect what is written to **infobright.log** for all currently existing client sessions that have not initialized the value of **ses_LogLevel**, as well as new client sessions that are launched.

FET (Function Execution Time) Logging

FET (Function Execution Time) logging is an extension to standard Infobright logging, which can be useful for trouble-shooting slow queries.

FET logging is by default disabled, as it will cause performance degradation.

Important

FET logging should not be enabled without prior written approval from Infobright. Doing so may result in Infobright's inability to support you.

Enable or Disable FET Logging at Server Start-up Using infobright.cnf

FET logging is initially enabled or disabled at Infobright Server start-up based on the values of the "FET" and "FETInterval" parameters in the **infobright.cnf** file. Any modifications made to **infobright.cnf** will take effect the next time the Infobright Server is restarted. For more information, see "FET" on page and "FETInterval" on page .

Enable or Disable FET Logging Dynamically Using the FET and FETInterval Variables

At Infobright Server Start-up, the setting of the global variables **FET** and **FETInterval** are initialized to the values of the "FET" and "FETInterval" parameters that were specified in the **infobright.cnf** file. When a Postgres client is launched, it will always use the setting of the global **FET** and **FETInterval** variables.

Note

Unlike the previously discussed **LogLevel** global variable, there is no corresponding session variable for **FET** and **FETInterval** global variables, and hence it is the global variables that are always in effect.

The values of the **FET** and **FETInterval** global variables can be modified by entering commands such as the following:

```
postgres=# set ibengine FET = '1';
postgres=# set ibengine FETInterval = '60';
```

- The above example changes the setting to "1" for global variable **FET**. The complete list of valid values are "0 or false" (FET logging off) and "1 or true" (FET logging on).
- The above example changes the setting to "60" for global variable **FETInterval**. The complete list of valid values are "0" (generate FET report only at shutdown) and "n" (generate FET report every "n" seconds).
- Making the above change will immediately affect what is written to **infobright.log** for all currently existing client sessions as well as new client sessions that are launched. This is because it affects the global setting (which has now changed from what was originally specified by "FET" and "FETInterval" at server start-up).

About Errors

Infobright reports the same errors as the standard PostgreSQL server.

About SQL Command Syntax

The syntax for Infobright SQL commands is exactly the same as the syntax for PostgreSQL commands with some minor exceptions. For more information, see SQL Statement Syntax in the PostgreSQL Reference Guide.

There are special considerations when using the following commands with Infobright. All other SQL commands can be used with Infobright as they are with the standard PostgreSQL.

Using PostgreSQL Commands with Infobright

PostgreSQL Command	More Information
CREATE TABLE, DROP TABLE	"Creating and Dropping Tables" on page
SHOW TABLE STATUS	"Viewing Table Information and Compression Statistics" on page
INSERT, UPDATE, DELETE	"Data Manipulation Statements" on page
COPY FROM/TO	"Infobright COPY FROM Syntax" on page "Infobright COPY TO Syntax" on page
SELECT	"Running Queries in Infobright" on page
VIEW	"Creating VIEWs in Infobright" on page

About SQL ISO Standards

As mentioned in the previous section, Infobright uses the same syntax as the standard PostgreSQL commands. For information about the compliance of the PostgreSQL language with ISO SQL standards, see PostgreSQL Standards Compliance.

Infobright is approaching full ISO SQL compliance. However, certain sections of the ISO SQL standard are open to interpretation and each DBMS, including Infobright, may implement these sections slightly differently. Consequently, Infobright query results may differ from those of other databases.

For example, the SQL standard does not define a default collation for string comparisons, which affects the ordering of query results. Different databases will implement different collation approaches, thus displaying inconsistent results for such things as sorts.

4. Managing Infobright Tables

About the Infobright Database Files

Infobright tables are located in the **ib_data** subdirectory in your Infobright installation directory.

Within the **ib_data** subdirectory, Infobright databases are stored in separate subdirectories. Within each database subdirectory, data files for each Infobright table are stored in separate subdirectories.

Important
Do not manually copy a data table from one database to another by copying the database files—internal table numbering errors and Knowledge Grid inconsistencies may occur. To copy a table, use import and export commands or backup the entire database directory.

The Infobright Server uses additional directories and files to store temporary data and optimization information, such as Knowledge Nodes. The following shows the **ib_data** directory, containing the Infobright directories and files, as well as one database directory called "postgres":

```
-bash-4.1$ pwd
/usr/local/infobright-products/postgres/ib_data
-bash-4.1$ ls
BH_RSI_Repository
cache
brighthouse.seq
ib_data_version
infobright.cnf
infobright.lic
infobright.log
postgres
-bash-4.1$
```

About Supported Data Types

The following PostgreSQL data types are supported in IBDB for PostgreSQL.

Numeric Types

Data Type	Minimum	Maximum
BOOLEAN	Values are either 0 or 1.	
SMALLINT	-32767	32767
INT (INTEGER)	-2147483647	2147483647
BIGINT	-9223372036854775807	9223372036854775807
REAL	-3.402823466E+38	3.402823466E+38
DOUBLE PRECISION	-1.7976931348623157E+308	1.7976931348623157E+308
Numeric(M, D) where 0 < M <= 18 and 0 <= D <= M	-(1E+M – 1) / (1E+D)	(1E+M – 1) / (1E+D)

Date and Time Types

Data Type	Minimum	Maximum	Notes

DATE	100-01-01	9999-12-31	**0000-00-00** value is illegal in PostgreSQL and will be converted to minimum (**100-01-01**) on load when using COPY FROM and the DLP
Time (without time zone)	00:00:00	24:00:00	
TIMESTAMP without time zone	100-01-01 00:00:00	9999-12-31 23:59:59	**0000-00-00 00:00:00** value is illegal in PostgreSQL and will be converted to minimum (**100-01-01 00:00:00**) on load when using COPY FROM and the DLP
TIMESTAMP with time zone	1970-01-01 00:00:00 in UTC	2038-01-01 00:59:59 in UTC	**0000-00-00 00:00:00** value is illegal in PostgreSQL and will be converted to minimum (**1970-01-01 00:00:00**) on load when using COPY FROM and the DLP
Interval	-178000000 years	178000000 years	Currently not supported

String Types

Data Type	Maximum Length
BYTEA (binary string)	0 < N <= 65536
VARCHAR(N)	Maximum length depends on character set (encoding). **0 < N * B <= 65536** where **B** is the maximum number of bytes for a single character. For example, for UTF-8 it is 4 bytes, so the maximum number of characters that can be stored in a (VAR)CHAR column is 65536 / 4 = 16384
CHAR(N)	Fixed-length. Maximum length depends on character set (encoding). **0 < N * B <= 65536** where **B** is the maximum number of bytes for a single character.

Creating and Dropping Tables

Use the standard PostgreSQL commands to create and drop tables in Infobright. For detailed syntax information, see CREATE TABLE Syntax and DROP TABLE Syntax in the PostgreSQL Reference Manual. However, it should be noted that the Infobright Server supports a non-standard and limited modifier set. See "About Column Options" on page for information on supported and unsupported options when creating columns. It should also be noted that not all PostgreSQL features are supported. For example, zero-column or inherited tables are not supported, and Infobright tables cannot be created in a transaction block.

Important
Do not manually copy a data table from one database to another by copying the database files—internal table numbering errors and Knowledge Grid inconsistencies may occur. To copy a table from one database to another, export from the source database and then import into the target database or backup the entire database directory. You can rename the entire database by renaming the folder. However, you should not copy a database folder from one active instance to another, or within the same active instance.

- Create an Infobright table with the following command:

```
postgres> create table <table_name> (<column(s)>) with
(engine=infobright);
```

- Create an Infobright temporary table with the following command:

```
postgres> create temp table ib_temp(x int) with (engine=infobright);
```

- Drop a table with the following command:

```
postgres> drop table <table_name>;
```

Note
When creating a table, one should always use the **ENGINE=** option to ensure that the correct database engine is used (e.g. **engine=infobright** for an Infobight table). If the engine is not specified an Infobright table will be created by default.

Note
Prior to release 4.8.3, when the engine was not specified, a PostgreSQL table was created by default. Now you must specify PostgreSQL explicitly by specifying **engine=postgres**.

Note
Prior to release 4.8.0, the Infobright storage engine was called "Brighthouse". Tables previously created by specifying **engine=brighthouse** will now be associated with the "Infobright" storage engine.

For backwards compatibility, Infobright tables can now be defined by either using the new syntax **engine=infobright** or the previous syntax **engine=brighthouse.**

Note
When creating a table with a **SELECT** statement, the following syntax should be used:
CREATE TABLE table_a with (engine=infobright) AS SELECT * FROM <table name> WHERE <condition> ORDER BY <column name>

Modifying Table Structures

Infobright supports common **ALTER TABLE** commands to add columns to existing tables and modify table structures, the same as you would with a PostgreSQL table. In IEE for PostgreSQL, you can add a column, drop a column, rename a table or truncate a table.

- To add a column to an existing table, enter the following command:

```
postgres> ALTER TABLE <table_name> ADD COLUMN <col_name>
<col_definition>;
```

- To add multiple columns at once, enter the following command:

```
postgres > ALTER TABLE <table_name> ADD COLUMN <col1_name>
<col1_definition>, ADD COLUMN <col2_name> <col2_definition>;
```

- To remove **the most recently added** column from an existing table, enter the following command:

```
postgres > ALTER TABLE <table_name> DROP [COLUMN] <col_name>;
```

- To rename an existing table, enter the following command:

```
postgres > ALTER TABLE <tbl_name> RENAME TO <new_tbl_name>;
```

- To truncate a table, enter the following:

```
postgres > TRUNCATE <tbl_name>;
```

About Column Options

NULL and NOT NULL

Infobright supports **NULL** and **NOT NULL** specifications for columns.

- **NULL** allows **NULL** values for the column.
- **NOT NULL** replaces the imported **NULL** values with default values such as 0 (zero) for numeric columns and an empty string (‘’) for string columns.

LOOKUP Columns

Infobright provides an additional modifier for string data type columns, called a **LOOKUP** column. The **LOOKUP** column uses an integer substitution for values. You can declare a **LOOKUP** column on a **CHAR** or **VARCHAR** column to increase its compression and performance in queries. However, to use a **LOOKUP** column, the **CHAR** or **VARCHAR** column should meet the following criteria:

- While there is no fixed upper limit for unique values in the column (cardinality), the cardinality of the column should be low. The total size of a dictionary, being the total length of all distinct values, will be loaded into RAM (for example: 1 million distinct values that are each 100-characters wide will permanently occupy 100 MB of RAM.)
- The column should contain a large number of duplicate values: the ratio of total number of records to distinct values should be greater than 10.
- Typically, a **LOOKUP** column is useful for fields like state, gender, category, and the like where the number of instances is very high but the number of unique values is very low.

To determine the ratio of records to distinct values, determine the number of distinct values using **SELECT COUNT (DISTINCT <COLUMN>) FROM…** then compare this to the number of records using a **SELECT COUNT(<COLUMN>) FROM…**

Note

Using a **LOOKUP** on a column where there are more than 10,000 distinct values will result in greatly reduced load speeds.

To declare a column as a **LOOKUP** column, use the following syntax when creating a table. For example:

```
create table test_lookup (a VARCHAR(200) LOOKUP=TRUE, b VARCHAR(200)
LOOKUP=TRUE, c INTEGER) with (engine=infobright);
```

In this example, a **LOOKUP** attribute is associated with columns a and b but column c is a standard integer column.

Note

You can only declare a column as **LOOKUP** modifier at the time of table creation. Modifying the column using **ALTER TABLE** to add or remove the **LOOKUP** modifier is not supported.

Note

Prior to release 4.8.0, **LOOKUP** columns were called **DIMENSION** columns. Columns previously defined as **DIMENSION** columns will now be associated with **LOOKUP** columns.

For backwards compatibility, **LOOKUP** columns can now be defined by either using the new syntax **lookup=true** or the previous syntax **dimension=true.**

Note

Issuing a **\d <table name>** command will display whether the **lookup** modifier has been used for any column in the table.

Optimizing Columns for INSERTs

Infobright provides an additional modifier for columns to help optimize **INSERT** operations, called a **for_insert** column. The **for_insert** modifier ensures that the most recent data pack is left uncompressed, allowing for faster **INSERTs** in the case of a large number of single **INSERTs**.

If you are expecting a large number of individual **INSERTs** or small frequent **LOADs**, you should consider setting the **for_insert** modifier on character columns and large numeric columns (e.g. 64-bit random identifiers, part numbers). Small numeric columns (e.g. color number or region id) can be decompressed and re-compressed with ease and are unlikely to gain performance benefit from the **for_insert** modifier. For columns marked as **LOOKUP**, the **for_insert** modifier may give very little benefit only. For smaller machines you may wish to leave the **for_insert** modifier off in order to maximize compression for disk space.

- To declare a column as a for_insert column, add a for_insert modifier to the column. Enter the following command:

  ```
  postgres> create table ib_tab_ins(v varchar(10) for_insert) with
  (engine = infobright);
  ```

  OR

```
postgres> create table ib_tab_dim_ins(v varchar(10) lookup = true
for_insert) with (engine = infobright);
```

Note
You can only set the **for_insert** modifier at the time of table creation. Modifying the column using **ALTER TABLE** to add or remove the **for_insert** modifier is not supported.

Issuing a d <table name> command will display whether the for_insert modifier has been used for each column.

Unsupported Indices Options

Infobright uses Knowledge Grid technology instead of standard indices and does not support explicit indices.

Viewing Table Information and Compression Statistics

You can use the standard PostgreSQL commands to obtain information about a table.

For example:

```
car_sales=# show table status;

      Name       |  Rows  | Data size (MB) | Compression ratio |    Crea
ted time     |    Updated time
-----------------+--------+----------------+-------------------+--------
-------------+---------------------
 dim_cars        | 400    | 0              | 3.360             | 2013-11
-15 06:34:50 | 2013-11-15 06:34:50
 dim_dates       | 4017   | 0              | 13.225            | 2013-11
-15 06:34:50 | 2013-11-15 06:34:50
 dim_dealers     | 1000   | 0              | 3.882             | 2013-11
-15 06:34:50 | 2013-11-15 06:34:50
 dim_msa         | 371    | 0              | 2.423             | 2013-11
-15 06:34:50 | 2013-11-15 06:34:50
 dim_sales_area  | 32765  | 1              | 4.439             | 2013-11
-15 06:34:50 | 2013-11-15 06:34:50
 fact_sales      | 997493 | 130            | 6.566             | 2013-11
-20 04:15:02 | 2013-11-15 06:34:49
 stuff           | 0      | 0              | 0.000             | 2013-11
-20 07:17:08 | 2013-11-20 07:17:08
(7 rows)
```

Note
An alternative command to the above that will show similar information is the following:

```
SHOW INFOBRIGHT TABLES STATUS;
```

You can retrieve column specific information about size and compression by specifying the table in the show table status command.

For example:

```
car_sales=# show table status fact_sales;

Column      | Data size (Bytes) | Compression ratio
------------+-------------------+-------------------
vin             | 17125000          | 1.875
make_id         | 10125000          | 9.361
dealer_id       | 10125000          | 8.119
sales_area_id   | 10125000          | 5.396
msa_id          | 10125000          | 9.481
trans_date      | 10125000          | 1620.519
dlr_trans_type  | 5353392           | 47.204
dlr_trans_amt   | 8125000           | 3.608
sales_person    | 4427550           | 4.520
```

Note

An alternative command to the above that will show similar information is the following:

```
SHOW INFOBRIGHT TABLE STATUS <table>;
```

Infobright provides specific statistics on table and column compression. The compression ratio is calculated in relation to the “natural size” of uncompressed data in the table or column. The ratio equal to **n** means that the compressed data, including statistics and technical description of a column, is **n** times smaller than its ***theoretical*** natural size.

Data Types and Natural Sizes

The following natural sizes (in bytes) are defined for various data types. Note the following:

- For all data types, if the column is not declared as **NOT NULL**, add one bit per value for **NULL** indicators.
- These data sizes take into account the typical format of data display, for example “yyyy-mm-dd” for **DATE** or decimal point for **DEC**. The size also counts the bytes that store the actual text length (**VARCHAR**).

The data type’s natural size is approximately equal to the binary import/export format.

Data Types and Natural Sizes

Data Type	Natural Size (in bytes)
CHAR(n)	n*(number of rows)
BIGINT, INT, SMALLINT, BOOL	(8 or 4 or 1 or 1)*(number of rows)
DATE	10*(number of rows)

TIME	8*(number of rows)
TIMESTAMP	19*(number of rows)
NUMERIC(x,y)	(x+1)*(number of rows)
REAL	4*(number of rows)
DOUBLE PRECISION	8*(number of rows)
VARCHAR(n), BYTEA(n)	(total number of bytes used—i.e., the total length of all strings, excluding terminating characters) + 2*(number of rows)

Comparison of Calculated Compression Ratio to Physical Size

The compression ratio calculated above will differ from the compression ratio calculated from physical sizes of files on disk. The compression ratio based on physical size will be slightly smaller, due to extra files that are generated containing statistics on the imported data, such as Knowledge Nodes. Knowledge Nodes are used to optimize query execution and are discussed further in "About the Knowledge Grid" on page .

Show Variables

Infobright supports a number of different configurable variables. These variables can be viewed using the following commands:

- To view PostgreSQL parameters, use the following command:

```
Postgres> SHOW ALL;
```

- To view Infobright parameters, use the following command:

```
Postgres> SHOW VARIABLES;
```

5. Data Manipulation Statements

Design of DML in Infobright

Infobright has been designed specifically for data warehousing applications, which are primarily load and read applications. Although Infobright supports **INSERT**, **UPDATE**, and **DELETE**, these constructs are designed for specific use cases.

UPDATE is used for updating slowly changing dimensions as frequent **UPDATEs** can result in performance degradation; the **DELETE** function is ideal for the removal or archiving of older data from tables and for the correction of invalid loads.

Infobright is not designed for OLTP type applications and its transaction model is limited. Using Infobright for an OLTP solution will result in poor performance and incremental effort will be required to enforce referential integrity.

Rules Regarding DELETE and UPDATE with Infobright

Infobright stores table row data in data files, each with a maximum size of approximately 2GB. When a row is deleted, it is marked as deleted in the data file (but no physical space is reclaimed). Whenever a **DELETE** or **UPDATE** occurs, the Infobright Compactor process will run. The Compactor will reclaim space when the following two conditions are met:

- All rows in the data file are marked as deleted
- The data file has reached its maximum size

The above logic for how / when Infobright reclaims space suggests a number of rules regarding **DELETE** and **UPDATE** that should be followed when using Infobright.

Important
Please keep these rules in mind at all times.

Monotonic / Rolling DELETEs

Monotonic / Rolling **DELETEs** are considered good. **DELETE FROM table WHERE Key<=Value** is good when **Key** is monotonic (i.e. always increasing). The previous statement should over time result in a pruning of old data from the system (assuming data is also inserted into the system based on the same monotonic **Key**).

Monotonic / Rolling **DELETEs** will result in rows marked as deleted being contiguous in the data files, eventually resulting in the conditions being met for Compactor to reclaim space.

Note
When you need to delete everything from a table, **DROP** the table; do **not DELETE FROM** (with no **WHERE** clause) the table. Similarly, do **not TRUNCATE** the table. Both **DELETE FROM** and **TRUNCATE** will only mark the rows as deleted, and will not necessarily result in reclaimed space (even when Compactor is run).

Ad Hoc DELETEs

Ad hoc **DELETEs** are detrimental and can cause an explosion in the calculated size of the data. Small volumes of ad hoc **DELETEs** can be managed but a large volume would skew many calculations such as data size.

As an example of an ad hoc **DELETE**, consider **DELETE FROM t WHERE Key=Value** where **Key** has many unique values (i.e. has a high cardinality) scattered in a random, unsorted fashion throughout table **t**.

UPDATEs

UPDATEs are detrimental for similar reasons. Remember that Infobright does not support in-place **UPDATEs**. This means that an **UPDATE** is a combination of **DELETE** and **INSERT** (remove the old data and then insert the new data). This amounts to ad hoc **DELETEs**.

INSERT

Infobright supports the **INSERT** statement. See INSERT Syntax in the PostgreSQL Reference Manual.

```
INSERT INTO table_name [ ( column_name [, ...] ) ]
    { VALUES ( { expression } [, ...] ) [, ...] | query }
```

UPDATE

Infobright supports the **UPDATE** statement. See UPDATE Syntax in the PostgreSQL Reference Manual.

```
UPDATE [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    SET { column_name = { expression } |
          ( column_name [, ...] ) = ( { expression } [, ...] ) } [, ...]
    [ FROM from_list ]
    [ WHERE condition ]
```

UPDATE can be used to maintain slowly changing dimensions, but if there are massive changes to the dimension, you might consider recreating the dimension with an ETL tool and simply dropping and reloading the dimension in the warehouse as this will improve performance.

DELETE

Infobright supports the **DELETE** statement. See DELETE Syntax in the PostgreSQL Reference Manual.

```
DELETE FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ USING using_list ]
    [ WHERE condition ]
```

Occasionally, data is incorrectly loaded to a fact table. **DELETE** can be used effectively in this case to remove the fresh incorrect data and replace it with the corrected data.

6. Character Set Support

Supported Character Sets

Infobright storage supports UTF-8 and Latin 1 character sets. This means that Infobright can store and retrieve data encoded in in multi-byte character sets.

Important
Queries that evaluate against UTF-8 character data columns will execute with less performance than an equivalent query against Latin 1 character data, due to Latin 1 support of character maps in the Knowledge Grid (see "Running Queries in Infobright" on page).

Collations and Comparisons

All PostgreSQL collations are supported by IEE; refer to the PostgreSQL support page for information on how collations behave.

Note that only **LC_COLLATE** is supported by infobright designated tables. **LC_CTYPE** does not affect query behaviour.

The default collation used is binary collation. Additional notes on how ordering is governed:

- For Infobright, character data types are case-sensitive. For example, the condition '**toronto**'='**Toronto**' is not true in Infobright. Similarly, the condition **LIKE** '**Abc%**' is not true for '**abcde**'.
- The Infobright sorting order is **A…Z a…z** (for example '**Zeta**' < '**alfa**').
- The Infobright sorting order affects **ORDER BY** results, **GROUP BY** results (which is the order of groups and their definitions—for example, '**aaa**' and '**AAA**' define different groups) and **DISTINCT** results. **WHERE** conditions may also be affected if you are expecting a different sorting order than the one used by Infobright.

Padding

Infobright treats padding differently than other DBMS engines. Infobright assumes literal comparisons of text fields, including all whitespace characters. Therefore, a string containing two spaces is different than a string containing one space or an empty (0 length) string, which is also different than the **NULL** value.

The Infobright padding definition is compatible with the SQL standard. However, most DBMS systems have defined less restricted, customizable rules regarding text comparison. For example, '**abc   ' = '**abc**' may be true in some databases but is not true in Infobright.

Note
In **CHAR** columns, trailing spaces are trimmed on **LOAD**, **INSERT**, and **UPDATE**, whereas in **VARCHAR** columns values are loaded with all spaces.

7. Importing and Exporting Data in Infobright

About Importing and Exporting Data

Infobright provides three ways to import data:

- **INSERT** statement
- Infobright DLP
- **COPY FROM** statement

INSERT is described in "Data Manipulation Statements" on page and is the slowest load approach. The DLP is the fastest load method but supports less load syntax.

Infobright DLP

- Fastest loader
- Less error handling diagnostics (only the source file row number pertaining to the error is returned)
- Strict input file formats (supports delimited text and binary formats)
- Variable Data Pack size
- Load-time clustering

Refer to the Infobright DLP User Guide for more details.

INSERT

- Supported by virtually all ETL tools
- Can be very slow depending upon the approach and commit rate

COPY FROM

- Supports capability similar to the DLP (does not support cluster on load or varying packrow size in this release)
- Primarily used for local instance only
- File-based loading
- Can work within transactions

If you are using an ETL tool, then using DLP or **COPY FROM** method with the binary format would be most efficient, although this approach may require more data preparation. For large fact tables, using DLP or **COPY FROM** method with either binary or text input is recommended.

Infobright COPY FROM Syntax

COPY FROM allows for very fast loading of file data in a single step. This is equivalent to **LOAD DATA INFILE** for those familiar with MySQL. The **COPY FROM** syntax works in a manner similar to standard PostgreSQL (**COPY FROM**); however, there are differences in the options supported. The examples and table below outline these differences.

Usage Examples

```
copy tab1 from '/tmp/data' with (format txt_variable, lines_terminated_by
e'\n', delimiter ';')
```

```
copy tab1 from '/tmp/data' with (format infobright)
```

```
copy tab1 from '/tmp/data' with (format ib_binary)
```

```
COPY table_name
    FROM { 'filename' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]
```

where ***option*** can be one of:

- format: {'txt_variable' | 'infobright' | 'ib_binary'}
- delimiter
- quote
- escape
- encoding
- lines_terminated_by
- reject_file_path
- abort_on_count
- abort_on_threshold
- accept_missing_columns

Options for COPY FROM

Option	Equivalent DLP Parameter Name	Available Values	Default Value with Infobright
format	data-format (DLP values are “txt_variable” or “binary”)	txt_variable infobright ib_binary	txt_variable
delimiter	fields-terminated-by	only a single one-byte character	\t
quote	fields-enclosed-by	only a single one-byte character	(empty)
escape	escaped-by	only a single one-byte character	(empty)
encoding	data-charset	only supported encodings by Infobright	(database encoding) *4-byte UTF-8 characters (CHAR, VARCHAR types) are replaced with question marks
lines_terminated_by	lines-terminated-by		(empty)
reject_file_path	reject-file-path		(not specified)
abort_on_count	abort-on-count		(disabled)
abort_on_threshold	abort-on-threshold	in range (0,1)	(disabled)

accept_missing_column s	true or false	false

Data Format (Mandatory)

You must set the data format option. Possible values are:

- **txt_variable** is readable text
- **ib_binary** is a native binary representation as found in "Infobright Binary Format" on page
- **infobright** is created by the DLP

Infobright Loader Reject File

By default, the **COPY FROM** command aborts on the first record that cannot be correctly parsed. However in some cases you may want the load process to continue and then later review rows that can't be loaded. You can use the Reject File functionality to accomplish this.

Reject File is disabled by default. To enable it, specify **reject_file_path**; this is the path to a file that will contain the rejected rows after load. You can set the number of records that can be rejected prior to the load being aborted and rolled back. To accomplish this, set the **abort_on_count** or **abort_on_threshold** parameter.

Usage example:

```
copy from '<path to file with data>' with (format txt_variable, …,
reject_file_path '<path to reject file>', abort_on_count 3)
```

The above command would fail and the load would be terminated if there were more than three incorrect rows in the input file. All rejected rows will be added to **<path to reject file>**.

Infobright Loader Reject File Options

Option	Description
reject_file_path	Path to the file where rejected rows are stored. Rejected rows are placed into the reject file in the order they are rejected. The original format is preserved to allow the operator to correct and rerun the load for only the rejected rows. Note: If reject_file_path is set, abort_on_count ***or*** abort_on_threshold must be set as well.
abort_on_count	Abort and rollback the load if the number of rejected rows exceeds this value. If this value is not set, the load will be rolled back to the first bad record if the load fails. A value of -1 means never abort; a value of 0 means abort on first rejected row. There is no upper limit on this value. Note: abort_on_count and abort_on_threshold are mutually exclusive.
abort_on_threshol d	Abort and rollback the load if the relative number of rejected rows to total processed rows exceeds this value (threshold test starts after one packrow row has been processed). Value must be in the range (0,1) - this is an open interval. For example:

	set @ abort_on_threshold=0.01 / 0.5 / 0.99 means that 1% / 50% / 99% of all processed lines corrupted will terminate the Infobright Loader and save the problematic rows in the reject file. Note: abort_on_count and abort_on_threshold are mutually exclusive.

Accept Missing Columns

The option **accept_missing_columns** controls whether data files in the “infobright” data format can still be successfully imported when the schema of the table being loaded has changed (by having additional columns added). Scenarios where this is useful include the following:

- Needing to reload data into tables from backup files that were originally created by DLP using an older schema.
- DLP being run on a server in which the source “file creation” process has not yet been updated to match new updates to the Infobright table schema. **Note:** Under this scenario, one would need to run DLP using an earlier pre-fetched schema (without the additional columns).

Valid values for **accept_missing_columns** are “0 or false” (do not allow accept input data with fewer columns) and “1 or true” (accept input data with fewer columns).

Note

When the DLP generated file is loaded into an Infobright table (using the **COPY FROM** command), NULLs will be loaded into the missing columns. If the column was defined as NOT NULL, the same rules as in case of ‘load’ in 'txt_variable' format will be applied. It means that NULLs will be replaced with 0 for numeric columns, with empty string for CHAR/VARCHAR, ... columns; '0' for DATE/DATETIME/TIMESTAMP columns, for TIMESTAMP NOT NULL current timestamp will be used.

Importing Files with Invalid Values

Infobright may abort a load when invalid values are found. Certain invalid values, however, can be loaded in Infobright. The following rules are used with invalid data:

- If a numeric is invalid, the value is replaced by 0.
- If a **TIME**, **DATE** or **TIMESTAMP** is invalid, the value is replaced with a minimum value for the given data type.
- If a **NULL** value is imported into a column defined as **NOT NULL** (except for **TIMESTAMP** columns), it is replaced by 0 (for numerical, date and time columns) or by an empty string (for string columns).

Options for Different Formats

Option	PostgreSQL Formats text, csv, binary	txt_variable	ib_binary	infobright
oids	As in PostgreSQL	Not supported	Not supported	Not supported

null	As in PostgreSQL	Not supported	Not supported	Not supported
header	As in PostgreSQL	Not supported	Not supported	Not supported
force_quote	As in PostgreSQL	Not supported	Not supported	Not supported
force_not_null	As in PostgreSQL	Not supported	Not supported	Not supported
delimiter	As in PostgreSQL	Supported	Not supported	Not supported
quote	As in PostgreSQL	Supported	Not supported	Not supported
escape	As in PostgreSQL	Supported	Not supported	Not supported
encoding	As in PostgreSQL	Supported	Not supported	Not supported
lines_terminated_by	Not supported	Supported	Not supported	Not supported
reject_file_path	Not supported	Supported	Supported	Not supported
abort_on_count	Not supported	Supported	Supported	Not supported
abort_on_threshold	Not supported	Supported	Supported	Not supported

Infobright COPY TO Syntax

COPY TO can be used to export Infobright table data to a file. **COPY TO** allows for fast exporting of data from a select statement. This is equivalent to **SELECT INTO OUTFILE** in MySQL. The **COPY TO** syntax works in a manner similar to PostgreSQL (**COPY TO**) but supports a different set of options. The examples and table below outline these differences.

Usage Examples

```
copy (select ...) to '/tmp/data' with (format csv, delimiter ';')
copy (select ...) to '/tmp/data' with (format txt_variable,
lines_terminated_by e'\n', delimiter ';')
copy (select ...) to '/tmp/data' with (format ib_binary)
```

```
COPY { table_name [ ( column_name [, ...] ) ] | ( query ) }
    TO { 'filename' | PROGRAM 'command' | STDOUT }
    [ [ WITH ] ( option [, ...] ) ]
```

where *option* can be one of:

- format: {'text' |'csv' |' txt_variable' | 'ib_binary'}
- delimiter
- quote

- escape
- encoding
- lines_terminated_by
- null

For Infobright tables, the following formats are supported:

- Infobright formats: **txt_variable**, **ib_binary**
- PostgreSQL formats: **text**, **csv**

Options for COPY TO

Option	text, csv	txt_variable	ib_binary
oids	As in PostgreSQL	Not supported	Not supported
null	As in PostgreSQL	Supported	Not supported
header	As in PostgreSQL	Not supported	Not supported
force_quote	As in PostgreSQL	Not supported	Not supported
force_not_null	As in PostgreSQL	Not supported	Not supported
delimiter	As in PostgreSQL	Supported	Not supported
quote	As in PostgreSQL	Supported	Not supported
escape	As in PostgreSQL	Supported	Not supported
encoding	As in PostgreSQL	Supported	Not supported
lines_terminated_by	Not supported	Supported	Not supported

Single-character Delimiter

IEE for PostgreSQL supports single character delimiters only.

About Transactions

About Transaction Behavior

While a write operation is being performed on a table, the following occurs:

- Queries to the table are executed against the state of the database before the write operation began. Once the current **LOAD** or **INSERT/UPDATE/DELETE** is complete and the operation is committed then subsequent queries execute against the new state of the database.
- Until the current write operation is committed, all subsequent write commands to the table are queued. They will wait for the write lock to be released before proceeding in the order they were received.

While a read query is being executed on a table, the following occurs:

- All subsequent queries run concurrently with the current query.

- A subsequent **LOAD** or **INSERT/UPDATE/DELETE** will run concurrently with the current queries. Further write operations are queued (as described above).

In general, Infobright uses table level locking where only one write operation (**INSERT**, **UPDATE**, **DELETE** or **LOAD**) can execute at one time.

Autocommit in IEE-Postgres

In IEE-Postgres, all DML operations are by default automatically committed. To disable "autocommit", one must explicitly "wrap" DML operations in a transaction using the **begin** and **COMMIT** (or **ROLLBACK**) commands in the following way:

```
psql> begin;
psql> DML
psql> DML
psql> COMMIT; (or ROLLBACK)
```

Failure Handling

If the Infobright Server is terminated during an export operation to a disk file, the following occurs:

- A non-empty file is saved on disk; however, the last row in the saved file is inconsistent.
- The database files are not harmed by the failed export operation. To export the data, repeat the export operation.

If Infobright tries to import data from a file created during a failed export session, the following occurs:

- No data is inserted because the input file consists of corrupted table rows. No new records are added to the database files, so no harm is done.

About Export Differences in Infobright

There are several important differences between exporting data from Infobright and exporting data from other DBMS engines.

Escape Characters

The Infobright and PostgreSQL Loaders support escape character definition and usage.

Exporting NULL Values

Infobright recognizes the following representations of **NULL** values when loading data from a text file:

```
NULL, \N, <field delimiter><field delimiter>
```

However, by default Infobright exports **NULL** values in the following representation:

```
\N
```

This can be modified using the null modifier in the **COPY TO** command.

Infobright Binary Format

With Infobright's binary format load, individual rows are not separated by any special characters. There are also no value delimiter or qualifier.

The structure of binary data files is as follows:

Data is stored contiguously: **<row_size><nulls><data_col_1>...<data_col_n>** and then the next data rows, without any line separator.

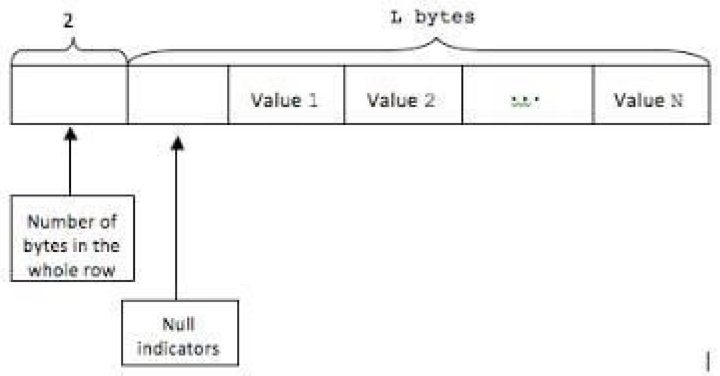
<row_size>	2-byte short integer indicating total number of bytes in this row (including all header bytes).
<nulls>	Binary map of null values, every byte reflecting to 8 consecutive columns. Bit 0 means a normal value, bit 1 means null value. The length of the <nulls> section is floor((number_of_columns+7)/8); i.e. minimal number of bytes to cover the number of columns (one bit per column).
<data_col_1>	Data itself, depending on column type. ▪ Floating point values are stored here as 8-byte values. ▪ Most numerical values (e.g. integers, dates) are stored as 4-byte integers. ▪ Fixed size texts (e.g. CHAR(n)) are stored on the fixed number of n bytes. ▪ Other text types (e.g. VARCHAR(n)) have their length stored on the first 2 bytes, followed by the text.

For example, assume we have two floating point columns. In this case, the binary file will look like the following:

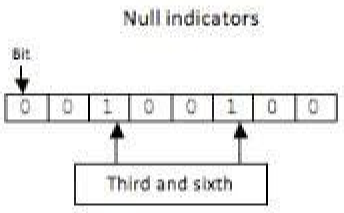
```
11, 0, 0, a1, a2, a3, a4, a5, a6, a7, a8, b1, b2, b3, b4, b5, b6, b7, b8
```

where (11, 0) is a 2-byte (HEX) representation of the record length after the first 0, the second 0 is null map (no nulls in this case), (a1a2a3a4a5a6a7a8) is an 8-byte representation of the first double and (b1b2b3b4b5b6b7b8) is an 8-byte representation of the second double. If the file contains 1000 rows it will have a length of 19000 bytes.

The following schema illustrates the format of one row in the BINARY format.



Every row starts with L (2-byte integer) which specifies the number of the following bytes of data. Null indicators are an array of bits – one bit per each column. 1 on m-th bit means that the m-th value in the row is **NULL**.



The number of columns in a record determines the numbers of bytes in **NULL** indicators. For example, for a record that contains from one to eight columns, indicator bits are stored on one byte. If a record contains from nine to 16 columns, two bytes are used and so on.

NULL indicators array is followed by N values where N is a number of columns in a row.

Formats and lengths in bytes for particular data types

Data Type	Format	Length in Bytes
SMALLINT		2
INTEGER		4
BIGINT		8
REAL	IEEE 4-byte Float	4
DOUBLE PRECISION	IEEE 8-byte Double	8
NUMERIC (N, M)	(Actual value) * 10^M	
TIME	[sign] [h] hh:mm:ss	8 - 10
DATE	4-byte integer yyyymmdd where yyyy = year - 1900	4

TIMESTAMP	yyyy-mm-dd hh:mm:ss	19
CHAR (N)	N characters	N
VARCHAR (N)	2-byte integer of value L followed by L characters	2+L
BYTEA (N)	2-byte integer of value L followed by L bytes	2+L

Note that **CHAR** is constant sized, whereas **VARCHAR** occupies only the size needed for the actual value. Integer and floating-point data are stored as a "natural" binary representation of these values (little endian).

Exporting and Importing Query Results

After exporting the results of a query to an output file, you may not be able to import the file back into the same definition of the accessed table. This is because the query may contain aggregates that will produce values beyond the boundaries of the original data types. In order to load the output file, you may need to create a new table with the appropriate data types for the values to be imported.

The following table shows the required data type conversions when using the binary format.

Data Type Conversions (Binary Format)

Operation	Column Data Type	Results Data Type
SUM	Smallint	BigInt
	Int	
	BigInt	
SUM	Float	Double
	Double	
SUM	Numeric(N, M)	Double
AVG	Smallint	Double
	Int	
	BigInt	
	Double	
	Numeric(N, M)	
COUNT	Smallint	BigInt
	Int	
	BigInt	
	Double	
	Numeric(N, M)	

8. Running Queries in Infobright

About the Knowledge Grid

The Knowledge Grid is a set of Infobright metadata used by the Infobright Server (named "Infobright") to optimize query execution. The Knowledge Grid consists of Knowledge Nodes, which are optimization data for particular tables and columns. Knowledge Nodes are stored on disk in a special directory, specified in the **infobright.cnf** configuration file. Knowledge Nodes can be lost without losing data integrity.

About Knowledge Nodes

There are three kinds of Knowledge Nodes:

Infobright Knowledge Nodes

Knowledge Node Type	Description
Histogram	Used by Infobright to enhance the speed of most queries consisting of numerical conditions (including date/time, decimal, etc.). Histograms are created automatically during data load.
Character Map	Used by Infobright to enhance the speed of most queries consisting of text conditions. Character maps are created automatically during data load.
DPN (Data Pack Nodes)	Statistical metadata that describes the content of the Data Pack. Used to assist in data access and in rough operations. DPNs are created automatically during data load.

Running Queries

Running Queries

To run queries on Infobright tables, use the following standard PostgreSQL syntax:

```
Postgres> select …;
```

The Infobright Optimizer is the primary engine used to resolve queries. While significant additions have been made to the library of supported SQL, there are cases where the query will still be executed by the PostgreSQL query engine instead of the Infobright Server. In this event, query response time tends to suffer due to the fact that the PostgreSQL engine is row-oriented and therefore cannot make use of the Knowledge Grid information, and in some cases it can be too slow to be usable. For best performance, ensure your queries (and **VIEW**s) contain only syntax supported by the Infobright Optimizer.

Terminating a Query

If you want to terminate a query executed from a client session before the query is complete, do the following:

- **SELECT** * from **pg_stat_activity**. This requires logging to be enabled in **postgresql.conf** and the **stats_command_string** must be set to **true** (see "Other Postgres Logs" on page). Use the **kill <id>** command to terminate the query.

  OR

- If you are using a command line PostgreSQL client, you can use **Ctrl+C** to terminate the query.

PostgreSQL Execution Path

Certain query types or functions are not natively supported by IEE. Queries that cannot be executed by the Infobright Server will be executed by the PostgreSQL engine; these scenarios are likely to experience performance degradation. A message will appear when queries are executed by the PostgreSQL engine.

Note
Query syntax is not implemented in Infobright and will be executed by the PostgreSQL engine.

The PostgreSQL execution path can be disabled by setting **allow_postgres_query_path** to **false** in **postgresql.conf**.

Important
When queries are executed on Infobright tables by the standard PostgreSQL engine, performance can be significantly slower than when queries are executed by the Infobright Server.

Creating VIEWs in Infobright

Infobright supports the creation of **VIEWs**. Note that the **VIEW** must contain Infobright optimized syntax or the **VIEW** will be run in the PostgreSQL query engine.

Use the following syntax to create a **VIEW**:

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW name [ ( column_name [, ...]
) ]
    AS query
```

A **VIEW** must contain unique column names. If you select two columns with the same name from separate tables, at least one must be aliased or the column list option must be used.

SELECT Syntax Supported in Infobright

The following **SELECT** syntax is supported in Infobright.

For more information, see SELECT Syntax in the PostgreSQL Reference Manual.

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ [ AS ] output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION } [ ALL | DISTINCT ] select ]
    [ ORDER BY expression [ ASC | DESC ][, ...] ]
    [ LIMIT { count } ]
    [ OFFSET start [ ROW | ROWS ] ]
```

where ***from_item*** can be one of:

- table_name [*] [[AS] alias [(column_alias [, ...])]]
- (select) [AS] alias [(column_alias [, ...])]
- with_query_name [[AS] alias [(column_alias [, ...])]]
- function_name ([argument [, ...]]) [AS] alias [(column_alias [, ...] | column_definition [, ...])]
- function_name ([argument [, ...]]) AS (column_definition [, ...])
- from_item [NATURAL] join_type from_item [ON join_condition | USING (join_column [, ...])]

Note
Recursive **WITH** queries (i.e. use of the **Recursive** modifier) is not supported.

WITH queries can only be **SELECT** statements, and can only be attached to a primary **SELECT** statement (i.e. use of **INSERT**, **MODIFY**, and **DELETE** is not supported).

Query Performance

Due to Infobright's column-oriented data organization and other Infobright-specific features, query optimization in Infobright is slightly different than in traditional DBMS approaches.

- Infobright works well with data tables containing many columns where only necessary columns are accessed by query (as opposed to **SELECT** *). The traditional approach suggests keeping records as small as possible (e.g., using schema normalization and table decomposition). However, in Infobright, only necessary columns are used in calculations. Therefore, queries with many limiting conditions on many columns of the same table are especially well optimized in Infobright.
- In traditional DBMS systems, better performance can be achieved by creating indices. In Infobright, Knowledge Nodes are used instead of indices (Knowledge Nodes are created automatically). To further enhance performance, you can try to influence the data loading procedure by keeping similar data (e.g., for similar time frames) close together. The order in which data are loaded may influence both compression ratio and query speed.

- Avoid using **OR** in queries and, if possible, use **IN** instead. In some cases **OR**s can be translated to **UNION ALL** or **IN**. For example: **...WHERE a=1 OR a=2...** could be replaced by **...WHERE a IN (1,2)...**
- Try to replace correlated subqueries with joins and independent subqueries.
- Temp tables may be used to manage intermediate steps without needing to do database cleanup.

Rough Queries

About Rough Query

Rough query provides fast ad-hoc querying without indexes or other database optimizations. Query results are processed based on Knowledge Node information only and do not involve disk access.

Rough query will never tell you something does not exist when it actually does exist. This guarantee is an important property of the rough approximation the engine uses and is why rough query is appropriate for operational type queries (namely, iterative analytics).

This also means that all queries will return an answer lying between the upper and lower bounds. Taken to its extreme, this means that a poor quality estimate of rough aggregation will return +inf, -inf. Any existential query (e.g., simple projection) will have a similar guarantee - the range of values returned is guaranteed to be within the approximation of the rough evaluation. This gives you confidence in the result and is why operational telescoping queries (getting wider and narrower) provide context for queries.

Select “roughly” allows you to instantly see the Min/Max range of the aggregate and does so by using only the in-memory Knowledge Grid meta-data structures. For example:

```
Select roughly num_of_unique_visits from fact_log
```

returns the range in which the values in the column lie—that is, they return two rows, the upper and the lower bound (for example, 10/20).

Filters (where clause) are supported for rough query. Aggregates—such as Min, Max, Sum, Count(*)—are also supported. For example:

```
Select roughly Sum(num_of_unique_visits) from fact_log
```

returns the range in which the sum of that column lies in (for example, 100/200).

Ranges for VARCHARs are not supported. Correlated sub queries are optimized using rough evaluation so they perform faster.

9. Infobright Backup and Recovery

Backup Procedure

To backup the Infobright databases, do the following:

1. Create a copy of the entire directory containing the Infobright databases (usually the **ib_data** and **pg_data** subdirectories of your Infobright installation directory).

2. If the Knowledge Grid is not located in the **ib_data** subdirectory (only possible when KNFolder parameter not at default value), then also create a copy of the directory holding the Knowledge Grid.

Note
You can take advantage of incremental backups, since only some of the database files are updated when new data is imported. Be sure to do a full backup occasionally.

Restore Procedure

To restore the Infobright databases from a backup copy, do the following:

1. Replace the entire data directories (usually the **ib_data** and **pg_data** subdirectories of your Infobright installation directory) with the backup copies.
2. If the Knowledge Grid is not located in the **ib-data** subdirectory, replace the Knowledge Grid with the backup copy.
3. Ensure that the folder permissions on the restored directories are for the same user that is used to start the IBDB Server.

Important
Do not manually modify database files or move them from one database to another. This may lead to data corruption and unpredictable results.

A. Infobright Optimizer – Supported Functions and Operators

Supported Functions

The following functions are natively supported by the IBDB for PostgreSQL engine. You may continue to use PostgreSQL functions not on this list but they will be handled by the generic PostgreSQL execution path and this may have an impact on performance.

The list of functions below is subject to change and will likely grow as more and more functions are natively supported.

Numeric Functions

Function	Comments/Alternatives
ABS(), @	absolute value
CEIL(), CEILING()	smallest integer not less than argument
DEGREES()	radians to degrees
DIV(y, x)	integer quotient of y/x **Note:** x=0 will result in an error

EXP()	exponential
FLOOR()	largest integer not greater than argument
LN()	natural logarithm
LOG()	base 10 logarithm
LOG(b, x)	logarithm to base b
MOD(y, x), y%x	modulo operation **Note:** x=0 will result in an error
PI()	"π" constant
POWER(a, b), POW(a,b)	a raised to the power of b
RADIANS()	degrees to radians
RANDOM()	random value in the range (0.0, 1.0)
ROUND()	round to nearest integer
ROUND(v, s)	round to s decimal places
SIGN()	sign of the argument (-1, 0, +1)
SQRT()	square root **Note:** An attempt to take the square root of a –'ve number will return NULL
TRUNC()	truncate toward zero
TRUNC(v, s)	truncate to s decimal places

Trigonometric Functions

Function	Comments/Alternatives
ACOS()	inverse cosine **Note:** An attempt to take the inverse cosine of an out-of-range number will return NULL
ASIN()	inverse sine **Note:** An attempt to take the inverse sine of an out-of-range number will return NULL
ATAN()	inverse tangent
ATAN2(y, x)	inverse tangent of y / x

COS()	cosine
COT()	cotangent
SIN()	sine
TAN()	tangent

String Functions

Function	Comments/Alternatives
BIT_LENGTH()	number of bits in string
CONCAT(), ||	string concatenation **Note:** Whether using CONCAT() or ||, concatenation of NULL and a value will return NULL. This is different than generic PostgreSQL, where CONCAT() and || have inconsistent behaviour when concatenating NULL and a value. **Note:** Concatenation of different data types may not always be supported, or may be supported by only one of CONCAT or ||. For example, concatenation of BIGINT and VARCHAR is supported when CONCAT() is used, but is not supported when || is used, with execution handled by the generic PostgreSQL execution path
CONCAT_WS(sep text, str [, str [, ...]])	string concatenation with separator
LEFT(str, n)	returns first n characters in the string
LENGTH(), CHAR_LENGTH(), CHARACTER_LENGTH()	number of characters in string
LOWER()	convert string to lower case
LPAD(string, length [, fill])	fill up the 'string' to length 'length' by prepending the characters 'fill'
LTRIM(string text [, characters text])	remove the longest string containing only characters from characters (a space by default) from the start of string
OCTET_LENGTH()	number of bytes in string
POSITION('substring' in 'string')	location of specified substring
REPLACE(string text, from text, to text)	replace all occurrences in string of substring "from" with substring "to"
REVERSE(str)	return reversed string

RIGHT(str, n)	returns last n characters in the string
RPAD(string, length [, fill])	fill up the ‘string’ to length ‘length’ by appending the characters ‘fill’
RTRIM(string text [, characters text])	remove the longest string containing only characters from characters (a space by default) from the end of string
SPLIT_PART(string text, delimiter text, field int)	split string on delimiter and return the given field (counting from one)
STRPOS(‘string’, ‘substring’)	location of specified substring same as position(‘substring’ in ‘string’), but note the reversed argument order
SUBSTRING (string [from int] [for int])	extract substring **Note:** substring(string from pattern) is not supported, with execution handled by the generic PostgreSQL execution path
TO_NUMBER(text1, text2)	return numeric representation of string “text1” that is formatted as specified by “text2”
TRIM([leading | trailing | both] [characters] from string)	remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string
UPPER()	convert string to upper case

Date-Time Functions

Function	Comments/Alternatives
CURRENT_DATE	return the current date
DATE ‘constant’	extract date from constant
DATE(source)	extract the date part of a date/time value
DATE_PART(field, source)	extract the specified “field” from a date/time value or interval constant **Note:** This function has equivalent functionality as EXTRACT(), supports the same “field” values (e.g. “day”, “dow”, etc), and has the same restrictions. See specific EXTRACT() variations given below for further information
EXTRACT(day FROM source);	extract days part of a date/time value or interval constant **Note:** Use of “days” instead of “day” is also supported **Note:** “time” is an invalid source for this type of extract
EXTRACT(day_hour FROM source);	extract concatenation of days and hours part of a date/time value or interval constant **Note:** For a “date” source, hours part will return as “00”

	Note: For a "time" source, days part will return as null
EXTRACT (day_microsecond FROM source)	extract concatenation of days, hours, minutes, seconds and microseconds part of a date/time value or interval constant **Note:** For a "date" source, hours, minutes, seconds and microseconds part will return as "000000000000" **Note:** For a "time" source, days part will return as null
EXTRACT(day_minute FROM source);	extract concatenation of days, hours and minutes part of a date/time value or interval constant **Note:** For a "date" source, hours and minutes part will return as "0000" **Note:** For a "time" source, days part will return as null
EXTRACT(day_second FROM source);	extract concatenation of days, hours, minutes and seconds part of a date/time value or interval constant **Note:** For a "date" source, hours, minutes and seconds part will return as "000000" **Note:** For a "time" source, days part will return as null
EXTRACT(dow FROM source);	extract day of week from a date/time value **Note:** "time" is an invalid source for this type of extract
EXTRACT(doy FROM source);	extract day of year from a date/time value **Note:** "time" is an invalid source for this type of extract
EXTRACT(epoch FROM source);	extract number of seconds since 1970-01-01 00:00:00 UTC (can be negative) from a date/time value or interval constant
EXTRACT (hour FROM source)	extract hours part of a date/time value or interval constant **Note:** Use of "hours" instead of "hour" is also supported
EXTRACT (hour_microsecond FROM source)	extract concatenation of hours, minutes, seconds and microseconds part of a date/time value or interval constant
EXTRACT (hour_minute FROM source)	extract concatenation of hours and minutes part of a date/time value or interval constant
EXTRACT (hour_second FROM source)	extract concatenation of hours, minutes and seconds part of a date/time value or interval constant
EXTRACT (microseconds FROM source)	extract concatenation of seconds and microseconds part of a date/time value or interval constant **Note:** Use of "microsecond" instead of "microseconds" is also supported
EXTRACT (milliseconds FROM source)	extract concatenation of seconds and milliseconds part of a date/time value or interval constant **Note:** Use of "millisecond" instead of "milliseconds" is also supported

EXTRACT (minute FROM source)	extract minutes part of a date/time value or interval constant **Note:** Use of “minutes” instead of “minute” is also supported
EXTRACT (minute_microsecond FROM source)	extract concatenation of minutes, seconds and microseconds part of a date/time value or interval constant
EXTRACT (minute_second FROM source)	extract concatenation of minutes and seconds part of a date/time value or interval constant
EXTRACT (month FROM source)	extract months part of a date/time value or interval constant **Note:** Use of “months” instead of “month” is also supported **Note:** “time” is an invalid source for this type of extract
EXTRACT (quarter FROM source)	extract quarter from a date/time value or interval constant **Note:** “time” is an invalid source for this type of extract
EXTRACT (second FROM source)	extract seconds part of a date/time value or interval constant **Note:** Use of “seconds” instead of “second” is also supported
EXTRACT week FROM source)	extract week from a date/time value **Note:** Use of “weeks” instead of “week” is also supported **Note:** “time” is an invalid source for this type of extract
EXTRACT (year FROM source)	extract years part of a date/time value or interval constant **Note:** Use of “years” instead of “year” is also supported **Note:** “time” is an invalid source for this type of extract
EXTRACT (year_month FROM source)	extract concatenation of years and months part of a date/time value or interval constant **Note:** “time” is an invalid source for this type of extract
NOW(), CURRENT_TIMESTAMP	return the current date and time **Note:** CURRENT_TIMESTAMP(precision) is not supported, with execution handled by the generic PostgreSQL execution path
TIME ‘constant’	extract time from constant **Note:** Extract time from column not supported
TO_DATE(text, text)	convert string to date
TO_TIMESTAMP(int)	return “timestamp with time zone” representation of an integer which is interpreted as the number of seconds since 1970-01-01 00:00:00 **Note:** Specifying a negative integer constant is not supported, with execution handled by the generic PostgreSQL execution path. Specifying a negative integer column will return NULL
TO_TIMESTAMP(text1, text2)	return “timestamp with time zone” representation of string “text1” that is formatted as specified by “text2”

	Note: Specifying a value of "text1" that is less than "1970-01-01" will result in an error.
DATE_TRUNC('field', source)	Truncate TIME or DATE to specified precision.

Control Functions / Conditional Expressions

Function	Comments/Alternatives
CASE	generic conditional expression, similar to if/else statements
COALESCE()	returns the first of its arguments that is not null
GREATEST()	select the largest value from a list of any number of expressions
LEAST()	select smallest value from a list of any number of expressions
NULLIF(value1, value2)	returns a null value if value1 equals value2; otherwise it returns value1

Cast Functions

Function	Comments/Alternatives
CAST('source' as 'datatype'), source::datatype	perform a conversion between two data types **Note:** In IBDB when a cast is generally supported (e.g. varchar to bigint) but a specific instance of the cast is invalid (e.g. varchar has value "abc def"), a warning will be generated but the cast will be successful with a special value (e.g. 0) being used to represent the casted value. This is different than generic PostgreSQL, which would instead generate an error.
CONVERT_TO(string, dest_encoding)	convert string to dest_encoding

Aggregate Functions

Function	Comments/Alternatives
AVG()	return the average value of the argument
BIT_AND()	return bitwise and

BIT_OR()	return bitwise or
COUNT(), COUNT(DISTINCT)	return a count of the number of rows returned
MAX()	return the maximum value
MIN()	return the minimum value
STDDEV_POP()	return the population standard deviation
STDDEV_SAMP(), STDDEV()	return the sample standard deviation
SUM()	return the sum
VAR_POP()	return the population variance
VAR_SAMP(), VARIANCE()	return the sample variance

Other Functions

Function	Comments/Alternatives
INET_ATON(ip string)	calculates and returns the numeric value of an IP address
INET_NTOA(integer)	calculates and returns the IP address of a numeric value
MD5(string)	calculates the MD5 hash of string, returning the result in hexadecimal

Supported Operators

Pattern Matching Operators

Function	Comments/Alternatives
ILIKE	can be used instead of LIKE to make the match case insensitive according to the active locale
'string' LIKE 'pattern'	returns true if the string is contained in the set of strings represented by pattern
'expression' IS NULL 'expression' IS NOT NULL	checks whether the specified expression is or is not null

‘expression’ IS <boolean> ‘expression’ IS NOT <boolean>	checks the boolean value of the expression. Will always return ‘true’ or ‘false’ depending on value specified by <boolean> (which can be ‘true’, ‘false’ or ‘unknown’)
~	matches regular expression, case sensitive
~*	matches regular expression, case insensitive
!~	does not match regular expression, case sensitive
!~*	does not match regular expression, case insensitive

Logical Operators

Operator	Comments/Alternatives
AND	logical intersection
OR	logical union
NOT	negation

Bitwise Operators

Operator	Comments/Alternatives
&	bitwise AND
<<	bitwise shift left
>>	bitwise shift right

Numerical Operators

Operator	Comments/Alternatives
-	minus operator / change the sign of the argument **Note:** one cannot use a double “-“ (i.e. “--“) as its regarded as a comment in IEE-Postgres, whereas its valid in IEE-MySQL.
+	addition operator
*	multiplication operator
/	division operator

Date-Time Operators

Operator	Comments/Alternatives
+	date + x (where "x" is an integer constant or column) Adds "x" days to "date", and returns a "date" datatype **Note:** If "x" is a "bigint" column, explicit casting to "int" or "smallint" must be done
+	date + interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "year(s)", "hour(s)", etc) Adds "x" units to "date", and returns a "timestamp" datatype
+	date + time Adds "time" to "date", and returns a "timestamp" datatype
+	time + x (where "x" is a constant of the form 'hh:mm:ss') Adds "x" to "time", and returns a "time" datatype
+	time + interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "hour(s)", "minute(s)", etc) Adds "x" units to "time", and returns a "time" datatype
+	timestamp + interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "year(s)", "hour(s)", etc) Adds "x" units to "timestamp", and returns a "timestamp" datatype **Note:** "timestamp with time zone" data type is also supported
-	date - x (where "x" is an integer constant or column) Subtracts "x" days from "date", and returns a "date" datatype **Note:** If "x" is a "bigint" column, explicit casting to "int" or "smallint" must be done
-	date - interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "year(s)", "hour(s)", etc) Subtracts "x" units from "date", and returns a "timestamp" datatype
-	date - time Subtracts "time" from "date", and returns a "timestamp" datatype
-	date1 – date2 (where "date1" and "date2" are either constants of the form "yyyy-mm-dd" or columns with a "date" datatype) Subtracts "date2" from "date1", and returns an "integer" datatype
-	time - interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "hour(s)", "minute(s)", etc) Subtracts "x" units from "time", and returns a "time" datatype
-	timestamp - interval 'x y' (where "x" is an integer constant, and "y" is the unit (e.g. "year(s)", "hour(s)", etc) Subtracts "x" units from "timestamp", and returns a "timestamp" datatype

Note: "timestamp with time zone" data type is also supported

Note
While many of the above date-time operators support the use of the **interval** keyword, Infobright currently only supports **interval** constants (and does not support **interval** data types).

An implication of this is that while a query such as "**select a – 4 * interval '1 month' / 3 + 2.4 * interval '3 day' / 1.1**" is supported, queries such as "**select a – b * interval '1 month' / 3 + 2.4 * interval '3 day' / 1.1**" and "**select 4 * interval '1 month' / 3 + 2.4 * interval '3 day' / 1.1**" are not supported (with execution handled by the generic PostgreSQL execution path).

B. Infobright Column Optimizer

About the Infobright Column Optimizer

The Infobright Column Optimizer improves data compression and the performance of import, queries and export. The Column Optimizer allows you to define the composition of data, particularly columns. The database then uses this information to optimize the storage of the data and to reduce query processing time.

Column Optimizer metadata is maintained in the system tables of the **sys_infobright** database and should be managed only with the use of stored procedures.

Note
Prior to release 4.8.0, Column Optimizer was known as DomainExpert.

Decomposition Rules

Decomposition rules are the main Column Optimizer objects. Each rule describes the composition structure of values of a selected column expressed in a simple language. You can create, modify and delete rules using the following stored procedures from the system database:

- create_rule(***id***, ***rule***, ***comment***)
- update_rule(***id***, ***rule***)
- change_rule_comment(***id***, ***comment***)
- delete_rule(***id***)

where:

- ***id*** is a unique identifier or name of a rule
- ***rule*** defines the structure of values

- *comment* is a free description of the rule

Decomposition rules can be applied only to columns of string types that are not **LOOKUP** columns:

- CHAR
- VARCHAR

Assigning a rule to a column of another type or to a **LOOKUP** column is ignored.

You cannot set multiple rules on the same column. If the **set_decomposition_rule** procedure is called for a column with an already assigned rule, the previous rule is replaced with the new rule.

The rule format is defined as part of the "Decomposition Rules Language" on page .

Rule Management

- Rules can be created as:

  ```
  CREATE_RULE(rule_name, rule_definition, rule_description)
  ```

  Example:

  ```
  select CREATE_RULE('Number_in_the_middle', '%s%d%s', 'very nice rule');
  ```

- Rules can be updated as:

  ```
  UPDATE_RULE(rule_name, rule_new_definition)
  ```

  Example:

  ```
  select UPDATE_RULE('Number_in_the_middle', '%s.%d.%s');
  ```

- Rules can be deleted as:

  ```
  DELETE_RULE(rule_name)
  ```

  Example:

  ```
  select DELETE_RULE('Number_in_the_middle');
  ```

  * Only rules that are not committed to any column can be deleted.

- Rule comments can be modified as:

  ```
  CHANGE_RULE_COMMENT(rule_name, rule_new_comment)
  ```

  Example:

  ```
  select CHANGE_RULE_COMMENT('Number_in_the_middle', 'That is my new
  comment to the rule');
  ```

Rule Assignment

- Rules are assigned to tables as follows:

  ```
  SET_DECOMPOSITION_RULE(database_name, schema_name, table_name,
  column_name, rule_name)
  ```

  Example:

  ```
  select SET_DECOMPOSITION_RULE('test', 'public', 't1', 'a1',
  'Number_in_the_middle');
  ```

- Rules can also be unassigned:

```
DELETE_DECOMPOSITION_RULE(database_name, schema_name, table_name,
column_name)
```

Example:

```
select DELETE_DECOMPOSITION_RULE('test', 'public', 't1', 'a1');
```

Auxiliary Functions

- The list of all rules defined for a particular table can be obtained with the following query:
  ```
  select * from SHOW_DECOMPOSITION('test', 'public', 't1');
  ```
- They can also be tested for syntactical correctness:
  ```
  IS_RULE_VALID(rule_definition)
  ```
  Example:
  ```
  select IS_RULE_VALID('%d%d');
  ```

Decomposition Rules Language

The language to define the structure of values accepts three types of primitives:

- non-negative integer number, denoted as **%d**
- arbitrary character sequence, denoted as **%s**
- literal (a sequence of characters that are to be matched exactly)

Examples:

- **%d.%d.%d.%d** decomposes an IP address (4-byte version) in four 1-byte numerical components
- **%s@%s** decomposes an email address into the user name and the domain name
- **%s://%s?%s** decomposes a simple URL with a query string into the scheme, the address and the query string

Because the percent sign (%) is a special character, to match it literally you can use a double percent sign (%%). For example, to match exactly the text **10% humidity**, the rule can be defined as **10%% humidity**. However, the percent sign only has a special meaning if it is followed by the letter **s** or **d**. Otherwise the percent sign has the literal meaning, so in the above example the unmodified text **10% humidity** is also a correct syntax of the exact rule.

There are two constraints on the rule syntax—the following ambiguous subsequences of symbols are not allowed in rules:

- **%s%s**
- **%d%d**

The matching algorithm for rules is LAZY—the algorithm moves to the next primitive in the rule as soon as possible. For example, for the text **aa.bb.cc** and the rule **%s.%s,** the first **%s** is matched to **aa** and the second **%s** is matched to **bb.cc**. However, if the most lazy approach fails, the algorithm searches back until the correct match is found or all the cases are traced. For example, for the text **aa.bb.11** and the rule **%s.%d,** the string **%s** is matched to **aa.bb** and the number **%d** is matched to **11**.

The current language is a simple, limited language that will be replaced with a much more powerful language in the future. The current language does not support the following regular expression constructs (these will be added in future releases):

- Grouping—for example, **(%s.%s).%s@(%d%s).%s**
- Type classes—for example, **[%s|%d]@%s**
- Repetition—for example, **%s{5,10}**
- Optional inclusion—for example, **(%s.)?%d.** This currently matches **(string.)?1000** whereas it might more reasonably match **string.1000** and **1000**.
- Sub-expressions
- Word boundaries
- Back-references, i.e. each group has a reference—**$1** for the match of the first group, **$2** for the match of the second group and so on

Building recursive rules using the following operations is also not yet available:

- Concatenation: **r1r2** where **r1** and **r2** are any pair of already defined rules—matches any value that is a concatenation of any pair of values, with **v1** matching **r1** and **v2** matching **r2**
- Union (alternative): **r1|r2** matches each value that matches one of **r1** and **r2**
- Closure: **r*** matches each value which is any repetition of any value matching **r**

Predefined IPv4 Rule

Besides user-defined rules, Infobright provides a built-in rule that is not expressible in the above language. This is the **IPv4** rule that is defined and added to the Column Optimizer metadata at installation. **IPv4** converts the text representation of an IP address into a single 32-bit number as used in network hardware and low-level network handling software.

If you have data with IP addresses, this allows you to compare the performance of the predefined **IPv4** with IP decompositions expressible in the language—for example, with the rule **%d.%d.%d.%d**.

Other Predefined Rules

The following predefined rules are provided with Column Optimizer.

Rule ID	Rule Content	Comments
IPv4_C	%d.%d.%d.%d	Similar to IPv4 but uses generic numeric compression.
EMAIL	%s@%s	Username/domain split of an email address.
URL	%s://%s?%s	Protocol, domain and query parameters based rule.

These rules can be improved if the user data matches more specific criteria (for example, the domain always contains a suffix such as .com). Using specific criteria may improve both the compression ratio and the response time. If you want to use more specific rules, create new rules (instead of replacing the predefined ones).

To see the current decomposition rules for a particular table, use the **show_decomposition** procedure. For example:

```
CALL sys_infobright.show_decomposition('network', 'connection');
```

If a rule is assigned to a column, you cannot change or delete the rule from the **decomposition_dictionary** system table.

Applying Rules to Data

After decomposition rules are assigned to columns, the rules are automatically applied to any new data coming to the tables containing these rules when using the following standard DML commands:

- COPY FROM/DLP
- INSERT
- UPDATE

If a rule is assigned to a column, instead of storing whole values, each value inserted into the column is decomposed into the parts matching the subsequent occurrences of **%s** and **%d** in the rule and the parts are compressed and stored in separate subcollections. Each subcollection corresponds to one occurrence of **%s** or **%d** in the rule.

A value inserted into a column with a decomposition defined does not have to match the rule. Such non-matching values are inserted into a separate subcollection. This subcollection of outliers is compressed and stored independently of other subcollections.

You can obtain the accuracy of decomposition rules by setting the **LogLevel** parameter in the **infobright.cnf** file to **N** (or **D**).

```
LogLevel = N
```

If the parameter is set on each **COMMIT** for each column, Infobright reports the number of outliers among the committed values (from **INSERTs** and **LOADs**). For example:

```
Decomposition of ./network/connection.ip left 15 outliers.
```

Infobright also reports the change in the number of outliers for the updated values (from **UPDATEs**), for example:

```
The number of outliers increased by 2 after update(-s) on
network.connection.ip
```

```
The number of outliers reduced by 3 after update(-s) on
network.connection.ip
```

Note

Applying a decomposition rule DOES NOT always result in better compression ratio and time. A decomposition rule may result in a worse compression ratio or load and slower queries. To ensure decomposition improves performance, you can compare load time, compression ratio and query time when loading the same data to a table with a decomposition rule defined and to a table without decomposition.

Modifying a Rule for an Existing Column

A rule for a column can be changed or deleted during the life of the table using the following stored procedures:

- set_decomposition_rule(database, table, column, id)
- delete_decomposition_rule(database, table, column)

The change applies only to new data. The old data remains decomposed with the previously used rules. If the rule for a column is deleted, new values are stored without decomposition.

If a value is updated to a new value with an **UPDATE** command, then for the new value Infobright uses the original rule used to decompose the old value. The currently assigned rules are not used for **UPDATEs**.

C. Linux Tuning Settings

System Settings for Red Hat Enterprise Linux and CentOS

Disable SElinux

SElinux is intended to protect Linux servers on the public internet such as web servers. It provides an extra layer of security that isn't really required for a back-end database server.

- In /etc/sysconfig/selinux add:
  ```
  SELINUX=disabled
  ```

Swappiness

Set low swappiness to avoid unnecessary paging. This only helps for machines with low levels of memory (say 4GB with 3GB allocated for Infobright).

- In /etc/rc.local add:
  ```
  echo "7" > /proc/sys/vm/swappiness
  ```

Disable Unused Processes

Run system-config-services (or edit **/etc/initd.d** directory) and leave ssh running.

Disable Zone_Reclaim_Mode

It has been found that in high data volume environments, slowdowns in loads and queries can sometimes be reduced by disabling the zone_reclaim_mode parameter.

- Test the kernel setting by writing '0' to `/proc/sys/vm/zone_reclaim_mode`
- To turn zone reclaim mode off permanently: add `vm.zone_reclaim_mode = 0` to `/etc/sysctl.conf` and run `sysctl -p` to load the new settings

File System Settings

Ensure CacheFolder is on a Fast Local Disk

See "Cache Folder" on page .

Larger Readahead

In /etc/rc.local add:

```
blockdev --setra 2048 /dev/sd<x>
```

Replace ***sd<x>*** with a proper device symbol (e.g. **sdc**); it should be the drive(s) on which the **datadir** and/or **CacheFolder** resides.

Use XFS File System for Data Directories

For XFS (may need to install kmod-xfs and xfsprogs):

```
mkfs.xfs -b size=4096  /dev/sdc1
```

In /etc/fstab add:

```
/dev/sdc1  /bha xfs     noatime 1 2
```

Note
This is for data folders only. Linux boot partition can be ext3.

noatime

Use noatime options for mounting database and cache volumes (see the next section, "Deadline Elevator", for details). Otherwise the system will update the access time for files and directories (which degrades performance).

Deadline Elevator

The default scheduler (CFQ) is 1% faster than elevator for a single user. However, in multi-user test with 4 users, elevator had 20% better performance.

- In **/etc/rc.local** add:

```
echo "deadline" > /sys/block/sd<x>/queue/scheduler
```

Replace ***sd<x>*** with a proper device symbol (e.g. sdc); it should be the drive(s) on which the **datadir** and/or **CacheFolder** resides.

Increase ulimit to Support Large Data Volume or Users

This will not change performance but may avoid errors. Ulimit determines the maximum number of files a user can have open.

Increase ulimit to unlimited or 32,768 since the default file limit is 1024. This is insufficient for large databases (lots of columns) or servers with multiple Infobright databases.

- To view current settings, run the command:

  ```
  # ulimit -a
  ```

- To set it to a new value for this running session, which takes effect immediately, run the command:

  ```
  # ulimit -n 8800
  # ulimit -n -1 // for unlimited; recommended if server isn't shared,
  reportedly doesn't work on IB03
  ```

Alternatively, if you want the changes to survive reboot, do the following:

1. Exit all shell sessions for the user you want to change limits on.
2. As root, edit the file **/etc/security/limits.conf** and add these two lines toward the end:

   ```
   user1         soft    nofile          16000
   user1         hard    nofile          20000
   ```

   These two lines change the max number of file handles (nofile) to new settings.
3. Save the file.
4. Login as user1 again. The new changes will be in effect.

Note on how to detect ulimit problem

If you are noticing crashes during multi-user use cases, check the console log for the following error:

```
what(): FileSystem Error : Bad file descriptor
psql got signal 6;
```

To fix this, increase ulimit (see the previous section, "Increase ulimit to Support Large Data Volume or Users").

D. Infobright Data Tools

Infobright Consistency Manager

Infobright provides a tool to validate Infobright's specific metadata structures. The Infobright Consistency Manager (ICM) is an external standalone application that can be run against an Infobright instance to verify and repair most Infobright data structures including the Knowledge Grid and Data Packs.

If you are seeing unexpected behavior with Infobright such as server crashes, it can help to run the Infobright Consistency Manager for information for support and to perform repairs.

Note
Currently the Infobright database must be offline in order to run the Infobright Consistency Manager.

Infobright Consistency Manager Tests

The Infobright Consistency Manager runs the following tests.

Infobright Consistency Manager Tests

Test	Description
Delete mask consistency check	Checks that the delete mask headers contain the proper sum for the delete mask body. If any inconsistency is found between the header and body, the ICM returns the list of blocks of delete mask where inconsistencies were found.
Number of objects in columns equality	Compares the stored number of objects in each column file related to the table. If any inconsistency is found in the number of objects, the ICM returns the first two columns with different object numbers.
Comparison of maximal value in **LOOKUP** dictionary versus DPN	Executes only for **LOOKUP** columns. Compares the maximal key value stored in the **LOOKUP** column dictionary and in DPNs. If the values differ, the ICM writes them to the log.
Comparison of number of objects in first-column DPN versus delete mask	Compares the metadata stored in the headers of the delete mask and DPN file related to number of objects. If any inconsistencies are found, the ICM returns both numbers. The ICM compares only the first column because there exists an independent test comparing this value between columns. If the test does not find the proper delete mask file or the proper DPN file, the ICM reports corruption.
Knowledge Grid consistency for column	Checks if the histograms report the proper value of fixed parameter. A basic test of the Knowledge Node, ensuring the file has a proper format and the type of Knowledge Node corresponds to the column.
Knowledge Grid format for column	Each Knowledge Node is stored in a separate file. This test validates that the header data of each file is in the proper format.
Test for overlapping Data Packs in data files	Checks if there are Data Packs in files that overlap each other. If this situation is discovered, the ICM returns a list of pairs of Data Packs' numbers that are overlapping.
Test of table metadata consistency	Verifies if the table's metadata is valid. Includes verification of files used to store things like table name, number of columns and its names, types, and constrains like NOT NULL. These are the files created on CREATE TABLE and modified only on ALTER TABLE.
Test of DPNs for non-binary collation	Verifies Data Pack Nodes (DPNs) for string columns, defined with non-binary collations. If errors exist, they can be repaired using the ICM -–repair option.
Test number of rows in DPN is consistent	Tests that the number of rows in DPN is consistent with actual number of rows stored in data pack.

Running the Infobright Consistency Manager

To view the run options, run ICM with the **--help** flag:

```
icm_pure --help
```

To run ICM, use the following command:

```
icm_pure --datadir=<absolute path to data directory> [parameters]
```

Note

ICM should be run by the 'postgres' user. It should not be executed by 'root' or any rebuilt knowledge nodes will be owned by root (and not editable) which will result in issues when loading any subsequent data into the 'corrected' tables.

The above command is for Linux. On Windows, the command is "icm_pure.exe".

Infobright Consistency Manager Parameters

Parameter	Description
--help	Display help message and exit.
-V [--version]	Display version information and exit.
--basedir arg	Absolute path to Infobright installation directory.
--datadir arg	Absolute path to data directory. Mandatory.
--database arg	Name of database chosen for data integrity testing. Optional. If specified, no other databases will be tested.
--table arg	Name of table chosen for data integrity testing. Optional. If specified, no other tables will be tested.
--log-file arg	Print output to log file. Optional. If not specified, the logs will be printed to the console.
-F [--full-check]	Run full set of tests (may be time-intensive). Running ICM without the full-check option will result in a quicker test; however, the "Knowledge Grid consistency for column" test will not be run.
--repair	Repair found problems.
--rebuild-kns	Rebuild Knowledge Grid. See the next section for more information.
--stop-on-error	Stop tests on first error and report.
--cleanup	In case of an error in the ICM repair procedure, this option enables ICM to manually revert the datadir to its previous state. Running ICM with the cleanup option will remove the old DPN files (containing incorrect DPNs) from the datadir and also **makes the changes performed by**

ICM impossible to undo. If the cleanup option is not used, the old DPN files will remain in the datadir.

About Rebuilding/Repairing Knowledge Nodes

Executing a rebuild of the Knowledge Nodes (using the **--rbuild-kns** option) will run the following tests:

- Test of table metadata consistency
- Test of Knowledge Grid format for column
- Test of Knowledge Grid consistency for column

The **--rbuild-kns** option will fix any issues found for the first two tests ("Test of table metadata consistency" and "Test of Knowledge Grid format for column").

You can also use the **--repair** option along with the **--full-check** option to achieve the same results as **--rbuild-kns**. Using either of these methods will rebuild any Knowledge Nodes that have been deleted.

About Cleanup Procedures

ICM creates backup files when repairing problems related to "Test of DPNs for non-binary collation". (Backup files are not created for any other tests.) These backup files can be used to revert back to the original data if the ICM encounters an error during the repair procedure. To revert to the original data, copy or rename the **TAXXXXXDPN.icm_bck** files to the **TAXXXXXDPN.ctb** files (found in the **ib_data** directory).

ibtop

The **ibtop** tool provides monitoring of Infobright database operations and system resource usage. Use **ibtop** to monitor CPU usage, physical memory usage, disk I/O, cache directory size, query concurrency, and additional insightful metrics.

Note

Starting with release 5.0.1, **ibtop** utilizes a native C-API. Therefore, previously required modules such as perl are no longer needed to run **ibtop**.

ibtop is available for all supported OS platforms.

Command Options

A description of all command options available when running **ibtop** can be found by running the **ibtop --help** command. For example:

```
# ibtop --help
ibtop will collect running IB instance's statistics information, which
includes:

* /proc/[pid]/stat for CPU/Memory usage under Linux or windows equivalent
* show status like 'IB%' (mysql) or show infobright statistics (postgres)
* show engine infobright status (mysql) or show ibengine status (postgres)
* (IMM only) additional multi-machine specific metrics

You will not see output on screen. All collection goes to file specified by
--output-file/-o or --output-dir with a name convention.

Command Options::
  -h [ --help ]                   show command usage message
  -H [ --host ] arg               host where IB instance is running on;
                                  default: 127.0.0.1
  -P [ --port ] arg               port number of IB instance; default: 5029
  -L [ --login ] arg              login user name. Please make sure it has the
                                  permission to show IB specific status /
                                  statistics. If left empty, ibtop will use
                                  postgres for IEE-PG instance, and root for
                                  IEE-MySQL instance.
  -p [ --password ] arg           password for login
  -D [ --database ] arg           database to connect. If left empty, IEE-PG
                                  will use template1, IEE-MySQL will use
                                  information_schema.
  -S [ --server-type ] arg        IB instance type; valid options are: [mysql |
                                  postgres]; default: postgres
  -o [ --output-file ] arg        output file name
  -i [ --interval ] arg           interval in seconds between each collection;
                                  can be as frequent as 1 second; default: 60
                                  seconds
  -f [ --flush-on-intervals ] arg ibtop will keep collection x intervals, and
                                  flush into output file; default:60 intervals
                                  for each flush
  -R [ --output-directory ] arg   Directory to hold output files (JSON or CSV
                                  format); default is /tmp (C:\tmp for windows
                                  OS). If output-file specified, output-file
                                  will take precedence, otherwise ibtop will
                                  use name convention of /tmp/hostname.YYYYMMDD
                                  .HHMMSS.SERVERTYPE_HOST_PORT.infobright.[gpe
                                  | csv]. File extension .gpe is in JSON
                                  format, if --enable-json-output specified.
  -q [ --skip-header ]            skip header column names when report csv
                                  format. (Does not apply to JSON format)
  -b [ --abort-on-error ]         abort collection on error. e.g. lost
                                  connection to underlying IB instance. If left
                                  empty, ibtop will keep trying indefinitely to
                                  reconnect; and all values in collection will
                                  be 0.
  -G [ --enable-json-output ]     enable JSON format output; meanwhile turn off
                                  csv output. Please refer user manual for
                                  detail JSON output format.
  -g [ --debug ]                  debug mode. show more verbose messages to
                                  help ibtop developer.
  -c [ --config-file ] arg        config file path.
  -v [ --version ]                show current ibtop version
```

Running ibtop

The **ibtop** tool can be run either remotely or locally on the same server that the Infobright database engine is running. To run **ibtop** and collect database instance metrics, enter a command such as the following:

```
$ ibtop -H 192.168.20.105 -P 5030 -L root -S postgres -i 1 -f 10 -o
/tmp/colo105.ibtop.csv
```

Based on the above command, after 10 seconds (1-second (interval) * 10 (flush-on-interval)), information will be written to the file **/tmp/colo105.ibtop.csv**.

Note

Once output is collected, it can be opened directly in any spreadsheet software (csv), json reader (json), or loaded into a database like Infobright for analysis.

By default for csv, the first line contains the column header. You can omit column header by specifying **--skip-header=yes** in the command line.

.

Use of a Configuration File (e.g. in Order to Protect a Database Password)

As an alternative to entering **ibtop** options at the command line, it is also possible to specify them using a configuration file. This may be especially useful in order to protect passwords from command line sniffing.

For example, in order to run **ibtop** using a configuration file (called **ibtop.cnf**), enter the following command:

```
$ ibtop --config-file=ibtop.cnf
```

A sample configuration file is shown below. To actually use, it would be necessary to uncomment (i.e. remove leading # character) and specify the appropriate parameter value.

```
###==== begin of ibtop.cnf =====
#host=127.0.0.1
#port=5029
#login=
#password=
#database=
#server-type=postgres
#output-file=
#interval=60
#flush-on-intervals=60
#output-directory=/tmp
#skip-header=no
#abort-on-error=no
#enable-json-output=no
###==== end of ibtop.cnf =====
```

Collecting Database Process CPU / Memory Utilization from the Operating System

ibtop is able to optionally collect database process CPU / Memory utilization information from the operating system. To enable this functionally, the following must be true:

- ibtop must be run locally
- For Linux OS's, **ibtop** must be run using the same OS user as the data base instance (i.e. **postgres**) or as a super user (e.g. **root**)
- For Windows OS's **ibtop** must be run as a super user (e.g. administrator)

Note

Under Linux, the information collection is achieved by executing the **cat /proc/[IB instance pid]/stat** command. Under Windows, the information collection is achieved by issuing the **GetProcessTimes()** and **GetProcAddress()** function calls.

Collecting Infobright Statistics

Inside the Infobright engine, a global data cache is accessible by all threads. Global data cache consists of 3 heaps:

- Main Heap
- Large Temporary Heap
- System Heap

All IB data structures are inherited from a base class called "TrackableObject", with acronym "TO". All data structures will allocate on one of 3 heaps. The type of TrackableObject could be one the following (a non-exhaustive list):

- TO_PACK: The actual compressed/decompressed data pack. By default, each pack contains 65536 elements for a column.
- TO_SPLICE: Metadata information including min/max/null/sum of each pack. Because a single DPN structure is small, the engine will store them in splices.
- TO_RSINDEX: Mirror of files from BH_RSI_Repository. It can be CMAP (Character Map) for a string column or HIST (Histogram) for a numeric column.
- TO_FTREE: Mirror for lookup dictionary file (e.g.: $datadir/table.bht/TA#####.dic)
- TO_SORTER: Temporary structure used by a query when sorting is needed.
- TO_CACHEDBUFFER+TO_INDEXTABLE: Temporary structure used by a query.
- TO_FILTER: delete mask. Please refer "delete mask" section in User Manual.
- TO_TEMPORARY: Miscellaneous temporary structure used by a query, e.g. aggregation work buffer, join buffer etc.
- others: everything else.

ibtop collects the above heap and "TO" metrics for both size (in bytes) and block count.

Summary of Information Collected by ibtop

The following tables describe all the information that is collected by **ibtop.**

ibtop Information - General

Variable Name	Note
TimeStamp	ibtop always uses UTC timestamp, e.g. 2016-04-28 13:21:46 +0000
UniqueId	Identifier if you collect multiple ibtop. It takes command line server_type + remote_ip + port

ibtop Information - DB instance CPU/Memory usage from the OS

Variable Name	Note
PID	/proc/[pid]/stat column #1 pid
NumThreads	/proc/[pid]/stat column #20 num_threads
UserCPU	/proc/[pid]/stat column #14+#16 utime (include child process). The collected value is per-second usage
SystemCPU	/proc/[pid]/stat column #15+#17 stime (include child process). The collected value is per-second usage
VmSize	/proc/[pid]/stat column #23 vmsize
VmRSS	/proc/[pid]/stat column #24 rss

ibtop Information – from "show infobright statistics" command

Variable Name	Note
IB_gdc_false_wakeup	metric to measure efficiency of internal global data cache
IB_gdc_hits	number of retrieval, which get it from memory directly, without going through disk read -> decompression process
IB_gdc_load_errors	metric to measure efficiency of internal global data cache
IB_gdc_misses	number of retrieval, which not from data cache, but go through reading from disk, then decompress
IB_gdc_pack_loads	Number of DataPack loaded and cached
IB_gdc_prefetched	metric to measure efficiency of internal global data cache
IB_gdc_read_wait_in_progres s	metric to measure efficiency of internal global data cache
IB_gdc_readwait	metric to measure efficiency of internal global data cache
IB_gdc_redecompress	obsoleted
IB_gdc_released	Objects release from global data cache
IB_mm_alloc_blocs	blocks allocated per-second
IB_mm_alloc_objs	number of objects doing allocating per-second

IB_mm_alloc_pack_size	bytes allocated by Datapack objects per-second
IB_mm_alloc_packs	blocks allocated by Datapack objects per-second
IB_mm_alloc_size	allocated memory bytes-per-second
IB_mm_alloc_temp	number of temporary blocks allocated per-second
IB_mm_alloc_temp_size	temporary allocation bytes-per-second
IB_mm_free_blocks	number of blocks deallocated per-second
IB_mm_free_pack_size	bytes-per-second of blocks deallocated by Datapack objects
IB_mm_free_packs	blocks-per-second deallocated by Datapack objects
IB_mm_free_size	bytes-per-second deallocated
IB_mm_free_temp	number of temporary blocks deallocated per-second
IB_mm_free_temp_size	bytes of temporary deallocation per-second
IB_mm_freeable	Total allocated memory that is currently in the releasable state
IB_mm_release1	Dependent on specific object release algorithm
IB_mm_release2	Dependent on specific object release algorithm
IB_mm_release3	Dependent on specific object release algorithm
IB_mm_release4	Dependent on specific object release algorithm
IB_mm_reloaded	Number of times a datapack was loaded after eviction but before falling off the history list
IB_mm_scale	Integer factor representing the magnitude of maximum buffer sizes can be allocated in a query
IB_mm_unfreeable	Total allocated memory that is currently in the un-releasable state
IB_readbytes	Read from disk, in bytes-per-second
IB_readcount	Read operation count, in count-per-second
IB_writebytes	Write to disk, in bytes-per-second
IB_writecount	Write operation count, in count-per-second

ibtop Information – from “show ibengine status” command

Variable Name	Note
System Heap Total(size)	size (in bytes) for heaps, trackable objects
Main Heap Total(size)	size (in bytes) for heaps, trackable objects
Large Temporary Heap(size)	size (in bytes) for heaps, trackable objects

TO_PACK objects(size)	size (in bytes) for heaps, trackable objects
TO_SORTER objects(size)	size (in bytes) for heaps, trackable objects
TO_CACHEDBUFFER+TO_INDEXTABL E objects(size)	size (in bytes) for heaps, trackable objects
TO_FILTER objects(size)	size (in bytes) for heaps, trackable objects
TO_RSINDEX objects(size)	size (in bytes) for heaps, trackable objects
TO_SPLICE objects(size)	size (in bytes) for heaps, trackable objects
TO_TEMPORARY objects(size)	size (in bytes) for heaps, trackable objects
TO_FTREE objects(size)	size (in bytes) for heaps, trackable objects
other objects(size)	size (in bytes) for heaps, trackable objects
System Heap Total(block)	block count for heaps, trackable objects
Main Heap Total(block)	block count for heaps, trackable objects
Large Temporary Heap(block)	block count for heaps, trackable objects
TO_PACK objects(block)	block count for heaps, trackable objects
TO_SORTER objects(block)	block count for heaps, trackable objects
TO_CACHEDBUFFER+TO_INDEXTABL E objects(block)	block count for heaps, trackable objects
TO_FILTER objects(block)	block count for heaps, trackable objects
TO_RSINDEX objects(block)	block count for heaps, trackable objects
TO_SPLICE objects(block)	block count for heaps, trackable objects
TO_TEMPORARY objects(block)	block count for heaps, trackable objects
TO_FTREE objects(block)	block count for heaps, trackable objects
other objects(block)	block count for heaps, trackable objects
cache_folder_size	CacheFolder is defined in infobright.cnf. IB instance uses it to store temporary intermediate results if there is no enough memory. This metric is the summary size (in bytes) from all files under CacheFolder.

Format of JSON Output

When **ibtop** option **enable-json-output=yes**, then the output file will be in JSON format. Each JSON output file will have two level-1 sections: meta and data.

- The meta section contains 1 variable named “interval”; this value is equal to the input variable **--interval**
- The data section contains a series of key-value collections; The first key is the timestamp. The value of this key is a nested structure of statistics. The content of this structure is similar to the following:

```
ibtop →
      instance_unique_id →
                   trend
                   tag
                   config
```

To generate **instance_unique_id**, ibtop concatenates server_type, IP, and port. This combination provides a unique identifier in the event of monitoring or comparing two distinct **ibtop** outputs.

In the nested structure for a given **instance_unique_id**, ibtop collects:

- trend: collected metrics. This is the same set of data as CSV output.
- tag: fixed element, and will always be **IBTOP@INFOBRIGHT**
- config: The Infobright instance’s configuration parameters, e.g. **ServerMainHeapSize**

An example of JSON output is the following:

```
{
    "meta": {
        "interval": 1
    },
    "data": {
        "2016-05-02 17:27:56 +0000": {
            "ibtop": {
                "postgres_127_0_0_1_5029": {
                    "trend": {
                        "gdc_false_wakeup": 0,
                        "gdc_hits": 0,
                        "gdc_load_errors": 0,
                        "gdc_misses": 0,
                        "gdc_pack_loads": 0,
                        "gdc_prefetched": 0,
                        "gdc_read_wait_in_progress": 0,
                        "gdc_readwait": 0,
                        "gdc_redecompress": 0,
                        "gdc_released": 0,
                        "mm_alloc_blocs": 0,
                        "mm_alloc_objs": 0,
                        "mm_alloc_pack_size": 0,
                        "mm_alloc_packs": 0,
                        "mm_alloc_size": 0,
                        "mm_alloc_temp": 0,
                        "mm_alloc_temp_size": 0,
                        "mm_free_blocks": 0,
                        "mm_free_pack_size": 0,
                        "mm_free_packs": 0,
                        "mm_free_size": 0,
        "mm_free_temp": 0,
        "mm_free_temp_size": 0,
        "mm_freeable": 0,
        "mm_release1": 0,
        "mm_release2": 0,
        "mm_release3": 0,
        "mm_release4": 0,
        "mm_reloaded": 0,
        "mm_scale": 5,
        "mm_unfreeable": 4,
        "readbytes": 0,
        "readcount": 0,
        "writebytes": 0,
        "writecount": 0,
        "largetemporaryheap_block": 0,
        "largetemporaryheap_size": 0,
        "mainheap_block": 12,
        "mainheap_size": 4,
        "numthreads": 0,
        "pid": 0,
        "systemheap_block": 0,
        "systemheap_size": 0,
        "systemcpu": 0,
        "cachedbuffer_indextable_block": 0,
        "cachedbuffer_indextable_size": 0,
        "filter_block": 6,
        "filter_size": 0,
        "ftree_block": 0,
        "ftree_size": 0,
        "pack_block": 4,
        "pack_size": 0,
        "rsindex_block": 0,
        "rsindex_size": 0,
        "sorter_block": 0,
        "sorter_size": 0,
        "splice_block": 1,
        "splice_size": 0,
        "temporary_block": 0,
        "temporary_size": 0,
        "usercpu": 0,
        "vmrss": 0,
        "vmsize": 0,
        "cache_folder_size": 0,
        "other_block": 1,
        "other_size": 4
    },
    "tag": {
        "add": [
            "IBTOP@INFOBRIGHT"
        ]
    },
    "config": {
        "CfgName": "postgres_127_0_0_1_5029",
        "CfgCollectionInterval": "1",
        "CacheFolder": "cache",
                                    "ConnectTimeout": "5",
                                    "FET": "0",
                                    "FETInterval": "0",
                                    "HandshakeTimeout": "15",
                                    "IBEngineRevision": "IEE_4.8.3_r35390_36166",
                                    "KNFolder": "BH_RSI_Repository",
                                    "KNLevel": "1",
                                    "LicenseFile": "infobright.lic",
                                    "LoaderSaveThreads": "16",
                                    "LogLevel": "W",
                                    "LogRotateFiles": "9",
                                    "LogRotateSize": "250",
                                    "MemoryHardLimit": "0",
                                    "MemoryLargeTempPercentage": "20",
                                    "MemoryScaleReduction": "0",
                                    "ParallelAggrThreads": "1024",
                                    "ParallelJoinThreads": "1024",
                                    "ParallelScanDPsAtOnce": "1",
                                    "ParallelScanDPsPerThread": "10",
                                    "ParallelScanThreads": "1024",
                                    "ParallelSortThreads": "1024",
                                    "PeerCommitTimeout": "120",
                                    "PrefetchQueueLength": "18",
                                    "PrefetchThreads": "6",
                                    "ServerMainHeapSize": "8834",
                                    "ServerMainHeapThreshold": "5",
                                    "SpliceSize": "128",
                                    "SyncBuffers": "0",
                                    "ThrottleLimit": "0",
                                    "ThrottleScheduler": "0",
                                    "ses_LogLevel": ""
                                }
                            }
                        }
                    },
                    "2016-05-02 17:27:57 +0000": {...},
                    "2016-05-02 17:27:58 +0000": {...},
                    "2016-05-02 17:27:59 +0000": {...}
                }
            }
```

Create Infobright Table Syntax for CSV Output

When the **ibtop** output file is in CSV format (which is the default), then one option for further analysis is to load the data into an Infobright table. Sample **create table** syntax to accomodate this is as follows:

```
create table ibtop_collection_iee (
"timestamp" varchar(32),
uniqueid varchar(64),
pid int,
numthreads int,
usercpu int,
systemcpu int,
vmsize int,
vmrss int,
ib_gdc_false_wakeup int,
ib_gdc_hits int,
ib_gdc_load_errors int,
ib_gdc_misses int,
ib_gdc_pack_loads int,
ib_gdc_prefetched int,
ib_gdc_read_wait_in_progress int,
ib_gdc_readwait int,
ib_gdc_redecompress int,
ib_gdc_released int,
ib_mm_alloc_blocs int,
ib_mm_alloc_objs int,
ib_mm_alloc_pack_size int,
ib_mm_alloc_packs int,
ib_mm_alloc_size int,
ib_mm_alloc_temp int,
ib_mm_alloc_temp_size int,
ib_mm_free_blocks int,
ib_mm_free_pack_size int,
ib_mm_free_packs int,
ib_mm_free_size int,
ib_mm_free_temp int,
ib_mm_free_temp_size int,
ib_mm_freeable int,
ib_mm_release1 int,
ib_mm_release2 int,
ib_mm_release3 int,
ib_mm_release4 int,
ib_mm_reloaded int,
ib_mm_scale int,
ib_mm_unfreeable int,
ib_readbytes int,
ib_readcount int,
ib_writebytes int,
ib_writecount int,
system_heap_total_size int,
main_heap_total_size int,
large_temporary_heap_size int,
to_pack_objects_size int,
to_sorter_objects_size int,
to_cachedbuffer_to_indextable_objects_size int,
to_filter_objects_size int,
to_rsindex_objects_size int,
to_splice_objects_size int,
to_temporary_objects_size int,
to_ftree_objects_size int,
other_objects_size int,
system_heap_total_block int,
main_heap_total_block int,
large_temporary_heap_block int,
to_pack_objects_block int,
to_sorter_objects_block int,
to_cachedbuffer_to_indextable_objects_block int,
to_filter_objects_block int,
to_rsindex_objects_block int,
to_splice_objects_block int,
to_temporary_objects_block int,
to_ftree_objects_block int,
other_objects_block int,
cache_folder_size int
) with (engine=infobright);
```

Infobright MySQL to PostgreSQL Migrator ("External Migrator")

The Infobright External Migrator allows for migration of IBDB MySQL data to IBDB for PostgreSQL. The current version of the utility works under some basic assumptions and conditions.

Assumptions

- Migrates data from version 4 (latest Infobright MySQL) to data version 5 (latest PostgreSQL version)
- Destination data directories must be created for PostgreSQL
- Migration of text types is supported under the following conditions (all conditions must be satisfied):
  - if UTF-8 is a charset in all text columns and no other charset is used
  - if binary collations used
  - if max text length from a column does not exceed 16K
- Both PostgreSQL and the Infobright Server must be offline
- Columns of time types must not contain 0 (zeros)
- Specific data types will require more space
  - After the conversion to PostgreSQL, **VARCHAR(n)** types will require more than 64KB for a single value. IBDB for MySQL using UTF-8 may have to up 3 bytes whereas IBDB for PostgreSQL uses up to 4 bytes. The maximum value for n is 16K characters.

Data Type Mappings

The following table lists the data type mappings.

MySQL to PostgreSQL Data Type Mappings

MySQL IBDB Data Type	Infobright PostgreSQL IBDB Data Type
BOOL	SMALLINT
TINYINT	SMALLINT
MEDIUMINT	INT
INT	INT
BIGINT	BIGINT

FLOAT	REAL
DOUBLE	DOUBLE PRECISION
DECIMAL(M,N)	DECIMAL(M,N)
YEAR	(To be decided)
TIME	INTERVAL HOUR TO SECOND
DATE	DATE
DATETIME	TIMESTAMP WITHOUT TIME ZONE
TIMESTAMP	TIMESTAMP WITH TIME ZONE
CHAR(N)	CHAR(N)
VARCHAR(N)	VARCHAR(N)
TINYTEXT	VARCHAR(255)
TEXT	VARCHAR(N)
BINARY(N)	BYTEA(N)
VARBINARY(N)	BYTEA(N)

Limitations and Notes

- Table migration is done by copying the data; in-place migration is not supported.
- Currently tables with decomposition rules are not supported.
- The External Migrator will convert all data to lowercase.
- The External Migrator will change all ‘0000-00-00’ **DATE** values to ‘100-01-01’.
- The External Migrator will change all ‘0000-00-00 00:00:00’ **DATETIME** and **TIMESTAMP** values to ‘100-01-01 00:00:00’ and ‘1970-01-01 00:00:00’.
- The External Migrator will apply a common character set to all columns being migrated. This is necessary because IBDB for PostgreSQL requires that all columns within a given database have the same character set.
- The External Migrator will recalculate Data Pack Nodes and Knowledge Nodes.
- There is no support for Default values within PostgreSQL, therefore this modifier will not be migrated.
- The External Migrator will rename (with a “m_” prefix) any database, schema, table, or column names that appear in the Postres parser. A list of names that will be renamed is the following:

```
'abort', 'absolute', 'access', 'action', 'add', 'admin', 'after', 'aggregate', 'all',
'also', 'alter', 'always', 'analyse', 'analyze', 'and', 'any', 'approximately',
'array', 'as', 'asc', 'assertion', 'assignment', 'asymmetric', 'at', 'attribute',
'authorization', 'backward', 'before', 'begin', 'between', 'bigint', 'binary', 'bit',
'boolean', 'both', 'by', 'cache', 'called', 'cascade', 'cascaded', 'case', 'cast',
'catalog', 'chain', 'char', 'character', 'characteristics', 'check', 'checkpoint',
'class', 'close', 'cluster', 'coalesce', 'collate', 'collation', 'column', 'comment',
'comments', 'commit', 'committed', 'concurrently', 'configuration', 'connection',
'constraint', 'constraints', 'content', 'continue', 'conversion', 'copy', 'cost',
'create', 'cross', 'csv', 'current', 'current_catalog', 'current_date',
'current_role', 'current_schema', 'current_time', 'current_timestamp', 'current_user',
'cursor', 'cycle', 'data', 'database', 'day', 'deallocate', 'dec', 'decimal',
'declare', 'default', 'defaults', 'deferrable', 'deferred', 'definer', 'delete',
'delimiter', 'delimiters', 'desc', 'dictionary', 'dimension', 'disable', 'discard',
'distinct', 'do', 'document', 'domain', 'double', 'drop', 'each', 'else', 'enable',
'encoding', 'encrypted', 'end', 'enum', 'escape', 'except', 'exclude', 'excluding',
'exclusive', 'execute', 'exists', 'explain', 'extension', 'external', 'extract',
'false', 'family', 'fetch', 'first', 'float', 'following', 'for', 'for_insert',
'force', 'foreign', 'forward', 'freeze', 'from', 'full', 'function', 'functions',
'global', 'grant', 'granted', 'greatest', 'group', 'handler', 'having', 'header',
'hold', 'hour', 'ibengine', 'identity', 'if', 'ilike', 'immediate', 'immutable',
'implicit', 'in', 'including', 'increment', 'index', 'indexes', 'inherit', 'inherits',
'initially', 'inline', 'inner', 'inout', 'input', 'insensitive', 'insert', 'instead',
'int', 'integer', 'intersect', 'interval', 'into', 'invoker', 'is', 'isnull',
'isolation', 'join', 'key', 'label', 'language', 'large', 'last', 'lc_collate',
'lc_ctype', 'leading', 'leakproof', 'least', 'left', 'level', 'like', 'limit',
'listen', 'load', 'local', 'localtime', 'localtimestamp', 'location', 'lock',
'mapping', 'match', 'maxvalue', 'minute', 'minvalue', 'mode', 'month', 'move', 'name',
'names', 'national', 'natural', 'nchar', 'next', 'no', 'none', 'not', 'nothing',
'notify', 'notnull', 'nowait', 'null', 'nullif', 'nulls', 'numeric', 'object', 'of',
'off', 'offset', 'oids', 'on', 'only', 'operator', 'option', 'options', 'or', 'order',
'out', 'outer', 'over', 'overlaps', 'overlay', 'owned', 'owner', 'parser', 'partial',
'partition', 'passing', 'password', 'placing', 'plans', 'position', 'preceding',
'precision', 'prepare', 'prepared', 'preserve', 'primary', 'prior', 'privileges',
'procedural', 'procedure', 'quote', 'range', 'read', 'real', 'reassign', 'recheck',
'recursive', 'ref', 'references', 'reindex', 'relative', 'release', 'rename',
'repeatable', 'replace', 'replica', 'reset', 'restart', 'restrict', 'returning',
'returns', 'revoke', 'right', 'role', 'rollback', 'roughly', 'row', 'rows', 'rule',
'savepoint', 'schema', 'scroll', 'search', 'second', 'security', 'select', 'sequence',
'sequences', 'serializable', 'server', 'session', 'session_user', 'set', 'setof',
'share', 'show', 'similar', 'simple', 'smallint', 'snapshot', 'some', 'stable',
'standalone', 'start', 'statement', 'statistics', 'status', 'stdin', 'stdout',
'storage', 'strict', 'strip', 'substring', 'symmetric', 'sysid', 'system', 'table',
'tables', 'tablespace', 'temp', 'template', 'temporary', 'text', 'then', 'time',
'timestamp', 'to', 'trailing', 'transaction', 'treat', 'trigger', 'trim', 'true',
'truncate', 'trusted', 'type', 'types', 'unbounded', 'uncommitted', 'unencrypted',
'union', 'unique', 'unknown', 'unlisten', 'unlogged', 'until', 'update', 'user',
'using', 'vacuum', 'valid', 'validate', 'validator', 'value', 'values', 'varchar',
'variables', 'variadic', 'varying', 'verbose', 'version', 'view', 'volatile', 'when',
'where', 'whitespace', 'window', 'with', 'without', 'work', 'wrapper', 'write', 'xml',
'xmlattributes', 'xmlconcat', 'xmlelement', 'xmlexists', 'xmlforest', 'xmlparse',
'xmlpi', 'xmlroot', 'xmlserialize', 'year', 'yes', 'zone'
```

Using the External Migrator

Run the following command:

```
./ibextmigrator options
```

Available ***options*** are:

External Migrator Options

Option	Description
-h [--help]	Print help messages
-f [--force]	Continue migration even if an error occurs
-v [--verbose]	Show more details
-b [--pg-bin] arg	PostgreSQL installation path
-u [--pg-user] arg	PostgreSQL user used to create the migration database
-s [--src-datadir] arg	Source MySQL datadir
-i [--dst-ibdatadir] arg	Destination Infobright Server datadir (**ib_data**)
-p [--dst-pgdatadir] arg	Destination PostgreSQL datadir
-d [--dst-db] arg	Destination PostgreSQL database name
-t [--tables] arg	List of tables to migrate in the form "**db1.t1 db2.t3 db2.***" where * implies migration of every table in the database. All tables will be migrated to single database and schema (public unless dst-schema is not specified). If not specified, the External Migrator will attempt to migrate whole datadir, each database to a different schema
-c [--dst-schema] arg (=public)	Name of destination schema to which tables specified with the **-t** option should be migrated Defaults to **public**
--connection-db arg (=template1)	Database that External Migrator will use to connect to PostgreSQL. Change default (template1) if your user does not have priviliges to connect to it
--force-charset-con version [=arg(=utf8)]	Specifying this option will turn off check if all data selected for migration has common charset and trigger conversion to specified charset (utf8 is default) if necessary. You can also use this option to trigger conversion of all data to specified charset
--version	Print program version number and exit

Example:

```
ibextmigrator -b c:\ib-pg\bin -d from_mysql -u infobright -s
C:\datadir_version_4_utf8_bin -p c:\empty_pgdatadir -i c:\empty_ibdatadir -t
"test.table"
```

E. InfobrightDB Postgres Major Version Upgrade Guide

Overview

With the introduction of InfobrightDB Postgres 2019.2.0, the underlying PostgreSQL version included in the release is being increased from 9.2.2 to 9.5.19. This introduces various advantages, but also makes the previous straightforward upgrade mechanism insufficient. This guide will detail the steps that a InfobrightDB user must follow to migrate an InfobrightDB Postgres 2019.1.0 or earlier database to 2019.2.0.

The migration occurs in three steps:

1. Dumping the contents of the postgres tables and the schemas of the infobright tables.
2. Replacing the installed version of InfobrightDB.
3. Restoring the data to the new database.

Please be sure to have a backup of the data folders before starting the procedure.

Dumping

We will create two dump files for each database contained in the InfobrightDB server. One will contain the schemas and data for postgres tables, and the other will contain only the schemas for infobright tables. To do this, we must first identify every database in the system and all the infobright tables in each database.

In the examples we will use the default `postgres` user for connecting to the database. Any user with read access to the required resources can be used instead.

Example:

```
$ psql -U postgres
```

List the databases

After connecting to the server, use the `\l` command to list the existing databases.

Example:

```
postgres=# \l
                            List of databases
   Name     |  Owner    | Encoding | Collate | Ctype |    Access privileges
-----------+----------+----------+---------+-------+-----------------------
 d1         | postgres | UTF8     | C       | C     |
 d2         | postgres | UTF8     | C       | C     |
 postgres  | postgres | LATIN1   | C        | C      |
 template0 | postgres | LATIN1   | C        | C      | =c/postgres          +
           |          |          |          |        | postgres=CTc/postgres
 template1 | postgres | LATIN1   | C        | C      | =c/postgres          +
           |          |          |          |        | postgres=CTc/postgres
(5 rows)
```

In this example, we have two user-created databases, `d1` and `d2`, plus the default `postgres`, `template0` and `template1` databases. The databases we are interested in are the user-created ones plus the default `postgres` database. `template0` and `template1` can be ignored.

List the tables

After listing the databases, we must connect to each in turn and list the tables, for the purpose of identifying the existing infobright tables. We use the `\c` command to connect to a database, and the `\dt` command to list the tables.

Example:

```
postgres=# \c d1
You are now connected to database "d1" as user "postgres".
d1=# \dt
                  List of relations
 Schema | Name | Type  |  Owner   |        Options
--------+------+-------+----------+---------------------
 public | t1   | table | postgres | {engine=infobright}
 public | t2   | table | postgres | {engine=infobright}
 public | t3   | table | postgres | {engine=postgres}
(3 rows)
```

In the example, the database `d1` has three tables: `t1`, `t2`, and `t3`. `t1` and `t2` are infobright tables (distinguished by the Options column), and `t3` is a postgres table. Take note of the infobright tables as we will need their names in later steps.

Repeat this step for each of the databases in the server.

Create dump files

After listing each infobright table in each database, we will use the `pg_dump` utility to create dump files. The commands for this are:

```
pg_dump --clean --create --format=c --exclude-table=$ib_tables_in_db -U $pguser
-f $dump_file $dbname
```

and

```
pg_dump --clean --create --format=c --table=$ib_tables_in_db --schema-only -U
$pguser -f $schema_file $dbname
```

In our example, the distribution looks like so:

d1: { ib_tables: [t1,t2] }

d2: { ib_tables: [t4,t5] }

This requires 4 commands (2 per database):

```
$ pg_dump --clean --create --format=c --exclude-table=t1,t2 -U postgres -f
dump-d1 d1
$ pg_dump --clean --create --format=c --table=t1,t2 --schema-only -U postgres
-f dump-d1-ib d1
$ pg_dump --clean --create --format=c --exclude-table=t4,t5 -U postgres -f
dump-d2 d2
$ pg_dump --clean --create --format=c --table=t4,t5 --schema-only -U postgres
-f dump-d2-ib d2
```

In this example, the output is the four files: `dump-d1`, `dump-d1-ib`, `dump-d2`, `dump-d2-ib`. Choose a naming convention that makes it easy to distinguish the dump files for each database, and if they contain postgres data or infobright schemas.

Replace the installed version

Replacing the version is done as normal. Uninstall the existing version (this does not delete the data directories), and then install the new version. Initialize the data directories, and copy the license file to the new ib_data directory. The `infobright.cnf` file can be copied over as is, but the `postgresql.conf` file should be reviewed and changes added manually to ensure they are compatible with the new version.

In the example below we use all default values.

```
$ sudo rpm -e infobright-iee-postgres
$ sudo rpm -i infobright-iee_postgres-2019.2.0-0-<os-version>.rpm
$ sudo service infobright-iee-postgres initdb
$ sudo cp -p
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/infobright.lic
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/
```

```
$ sudo cp -p
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/infobright.cnf
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/
$ sudo cp -p
/usr/local/infobright-products/iee/postgres/2019.1.0/pg_data/pg_hba.conf
/usr/local/infobright-products/iee/postgres/2019.2.0/pg_data/
```

Restore the database

Finally, we start the new database server and restore the data.

```
$ sudo service infobright-iee-postgres start
```

Restore postgres databases, tables, and data

We use pg_restore to recover the database state that we dumped previously. The dump file will take care of creating each database, we connect to the default postgres database to restore them. The schema files have to connect to the corresponding database to operate correctly. Please take note of this distinction in the commands below.

```
pg_restore -U $pguser --clean --if-exists --create --dbname=postgres $dump_file
pg_restore -U $pguser --clean --if-exists --dbname=$dbname $schema_file
```

These paired commands have to be run once per database.

Example:

```
$ pg_restore -U postgres --clean --if-exists --create --dbname=postgres dump-d1
$ pg_restore -U postgres --clean --if-exists --dbname=d1 dump-d1-ib
$ pg_restore -U postgres --clean --if-exists --create --dbname=postgres dump-d2
$ pg_restore -U postgres --clean --if-exists --dbname=d2 dump-d2-ib
```

Restore infobright data

We don't want to modify the database files while the server is running, so we first shut it down:

```
$ sudo service infobright-iee-postgres stop
```

Finally, we can move over the infobright table data. Depending on the size of the data, it may be more convenient to move instead of copying. If you are moving the data, be sure to have backups to prevent data loss in the event of errors. In this example, we will use $old_ib_dir to denote the original location of the infobright data directory, and $new_ib_dir to denote the new location.

First, we need to locate the table data. We do this with the following command:

```
find $old_ib_data -name *.bht
```

Each `$tablename.bht` directory holds the data for a table. There's also a directory structure for identifying the database and schema the table is under. We must then place each of these directories in the new location. Since we already created the infobright tables in the previous steps, the directory structure and other metadata already exists in the new installation.

We must clear the placeholder data and move the old data to the new location with the following commands:

```
rm -rf $new_ib_data/$dbname/$schema/$tablename.bht
mv $old_ib_data/$dbname/$schema/$tablename.bht
$new_ib_data/$dbname/$schema/$tablename.bht
```

Example:

```
$ sudo find /usr/local/infobright-products/iee/postgres/2019.1.0/ib_data -name
*.bht
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d1/public/t1.bht
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d1/public/t2.bht
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d2/public/t4.bht
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d2/public/t5.bht
$ sudo rm -rf
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d1/public/t1.bht
$ sudo rm -rf
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d1/public/t2.bht
$ sudo rm -rf
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d2/public/t4.bht
$ sudo rm -rf
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d2/public/t5.bht
$ sudo mv
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d1/public/t1.bht
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d1/public/t1.bht
$ sudo mv
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d1/public/t2.bht
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d1/public/t2.bht
$ sudo mv
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d1/public/t4.bht
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d1/public/t4.bht
```

```
$ sudo mv
/usr/local/infobright-products/iee/postgres/2019.1.0/ib_data/d2/public/t5.bht
/usr/local/infobright-products/iee/postgres/2019.2.0/ib_data/d2/public/t5.bht
```

The server can then be started again, and the previous installation's data should now be available in the new PostgreSQL 9.5.19 database.

F. Document Change Log

Document Change Log

2019.2 GA – (2019.12.19)

- Updated product version.
- Added upgrade instructions.

2018.1 GA – (2018.10.02)

- Updated product version.

5.0.6 GA – (2017.10.20)

- Updated product version.

5.0.5 GA – (2017.08.23)

- Updated the first first page.
- Changed Formatting and header styles
- Updated the 'COPYRIGHT AND DISCLAIMER' section.
- Changed 'Infobright DB PostgeSQL Windows Installation', 'Infobright DB PostgeSQL Linux Installation' and 'Configuration' sections to reflect the changes in installation of infobright.cnf file.
- Changed the name of the product to Infobright DB in various places.
- Changed "Supported Functions" section to add newly supported functions: CONCAT_WS, DATE_TRUNC(), TO_DATE().
- Changed "Supported Functions" section to add newly supported Pattern Matching Operators: ~, ~*, !~, !~.
- Changed "Supported Functions" section to add newly supported Bitwise Operators: &, <<, >>.

5.0.6 GA – version 1.0 (2017.02.21)

- Changed "Linux RPM and DPKG Installation Instructions" section to remove step "Start or stop the service" and add a last step of "Start the Infobright Server".
- Changed "Linux RPM and DPKG Upgrade Instructions" section to correct instructions on what command needs to be executed to upgrade, as well as to specify where upgrade packages can be obtained.

- Changed “Supported Functions” section to classify “TO_NUMBER as a “String” function instead of a “Date-Time” function”.
- Changed “Supported Functions” section to add newly supported “Date-Time” Extract functions: day_hour, day_microsecond, day_minute, day_second, hour_microsecond, hour_minute, hour_second, minute_microsecond, minute_second, and year_month.
- Changed “Supported Functions” section add newly supported “Date-Time” Extract plural / secondary function formats: days, hours, microsecond, millisecond, minutes, months, seconds, weeks, and years.
- Changed “Document Change Log” section so that the log is sorted by “most recent release” first.

5.0.3 GA – version 1.0 (2016.11.15)

- Changed “Technical Requirements” section to update CPU and Memory requirements
- Changed “Starting and Stopping the Infobright Server” section to remove the “Windows” and “Linux” sub-sections
- Changed “Working with the Infobright Server” section to remove the “Windows” and “Linux” subsections and reorganize contents to remove ambiguity
- Changed “About Importing and Exporting Data” section to move and update the paragraph referring to ETL tools in order to remove ambiguity
- Changed “Infobright COPY TO Syntax” section to remove reference to “binary” format, and to consistently state that “null” is a valid option

5.0.2 GA – version 1.0 (2016.08.17)

- Changed “Technical Requirements” section to indicate that Windows Server 2012 is now supported, and removed RHEL / CentOS 5.x and Windows Server 2003 as supported platforms
- Changed “Supported Functions” section to add descriptions of the newly supported DATE_PART(field,source), TO_NUMBER(text1,text2), and TO_TIMESTAMP(text1,text2) functions, as well as to change descriptions of several of the EXTRACT(field FROM source) functions to indicate that an “interval constant” is also a valid source
- Changed “Supported Operators” section to indicate that “timestamp with time zone” is also a valid data type for the “+ interval” and “- interval” operators. Also added a “note” to indicate how interval constants are supported and to clarify that the interval data type is (still) not supported
- Changed “Decomposition Rules” section to correct the list of valid string types to which decomposition rules can be applied

5.0.1 GA – version 1.0 (2016.06.14)

- Changed “About Installation Packages” section to add clarification that installation of multiple IEE-Postgres instances on the same OS is not supported. Also removed explanation of dynamic versue statically linked builds.
- Changed “IEE-Postgres Windows Installation” section to add description of the new “/noadmin=yes” silent installation parameter

- Changed "IEE-Postgres Windows Upgrade" section to add requirement to run "infobright_postgres_upgrade.bat" script and remove requirement to run "ib_update_column_optimizer.sql"script
- Changed "IEE-Postgres Linux Upgrade" section to add requirement to run "infobright_postgres_upgrade.sh" script and remove requirement to run "ib_update_column_optimizer.sql"script
- Changed "Detailed Infobright Parameter Descriptions" section by adding descriptions of five new parameters: ClusterMulticastAddress, ClusterMulticastPort, ClusterMulticastInterval, ClusterMulticastPacketsPerCheck, and ClusterMulticastFilterAddresses
- Changed "Infobright COPY FROM Syntax" section to add "Equivalent DLP Parameter Name" (of "data-format") for the "format" option. The document was previously incorrect.
- Changed "Importing Files with Invalid Values" section to indicate that "reject_file_path", "abort_on_count" and "abort_on_threshold" options are supported for "ib_binary" format. The document was previously incorrect.
- Changed "SELECT Syntax Supported in Infobright" section to indicate that "WITH queries" are now supported
- Changed "Supported Functions" section by adding descriptions of three new string functions (REPLACE, REVERSE, and SPLIT_PART), five new date-time functions (EXTRACT(dow), EXTRACT(epoch), EXTRACT(milliseconds), EXTRACT(microseconds), and EXTRACT(week), and two "other" functions (INET_ATON and INET_NTOA)
- Changed "Supported Functions" section by indicating that "time" is an invalid source for a number of date-time EXTRACT functions
- Changed "Supported Operators" section by adding a table describing a number of newly supported date-time operators. Also removed the "Note" (in Numerical Operators table) that "minus operators on date/time data types are not supported" (as some now are).
- Changed "Infobright Consistency Manager Tests" section by updating the description of the "Test of DPNs for non-binary collation" test.
- Changed "Running the Infobright Consistency Manager" section to make some cosmetic changes, and to additionally specify the command to be used on Windows.
- Added "ibtop" section (and sub-sections with detailed information) as ibtop functionality was added in this release

4.8.3 GA – version 1.1 (2016.01.27)

- Changed "IEE-Postgres Windows Installation" and "IEE-Postgres Windows Upgrade" sections by specifying that user installing/upgrading IEE-Postgres must have *local* administrator rights
- Changed "Viewing Table Information and Compression Statistics" sections by adding the commands "SHOW INFOBRIGHT TABLES STATUS" and "SHOW INFOBRIGHT TABLE STATUS <table>"
- Changed "Show Variables" section by clarifying what is displayed by the "SHOW ALL" command and adding the "SHOW VARIABLES" command
- Changed "File System Settings" subsections "Larger Readahead" and "Use XFS Files System for Data Directories" by correcting some formatting / font issues

4.8.3 GA – version 1.0 (2015.12.29)

- Changed "Detailed Infobright Parameter Descriptions" section by adding descriptions of three new parameters: HandshakeTimeout, PeerCommitTimeout, and ConnectTimeout
- Changed "Creating and Dropping Tables" section to reflect change in default Engine now being "Infobright" (previously default Engine was "Postgres")

4.8.2 GA – version 1.0 (2015.10.12)

- Changed "Technical Requirements" section to remove reference to 32-bit Windows support
- Changed "Supported Functions" and "Supported Operators" sections by adding additional comments to several functions and operators in order to clarify IBDB supported behaviour and better explain differences with generic PostgeSQL behaviour

4.8.1 GA – version 1.0 (2015.09.01)

- Changed "Infobright Overview" section to remove reference to a 50TB data limit
- Changed "Configuring Infobright (infobright.cnf file)" section to clarify that the file is only read at server start-up
- Added "Infobright Specific PostgreSQL Parameters" section in order to provide an explanation of these parameters
- Added "Infobright.log Example" and "Structure of an infobright.log line" sections in order to provide an explanation on how to read the infobright.log file
- Changed "LOOKUP Columns" section to add "Note" that it can only be defined at time of table creation
- Changed "Optimizing Columns for INSERTS" section to modify "Note" to better clarify that it can only be defined at time of table creation
- Changed "Rules Regarding DELETE and UPDATE with Infobright" section significantly by renaming section (from "Rules regarding UPDATE and DELETE with Infobright"), providing an overview on how Infobright handles deletes and the role of the Infobright Compactor, providing an explanation of "rolling deletes" in renamed and reordered sub-section "Monotonic / Rolling Deletes", by moving description of "what do do when you need to delete everything" from "Updates" sub-section to a "Note" in the "Monotonic / Rolling Deletes" section, and by adding a warning against the use of TRUNCATE
- Changed "Supported Character Sets" section to remove reference to UTF-8 specific Knowledge Grid extensions in an upcoming release
- Changed "About Transaction Behavior" section to add the "Autocommit in IEE-Postgres" sub-section in order to provide an explanation on how / when commits occur
- Changed "Supported Operators" section to add descriptions for "IS NULL / NOT NULL" and "IS <boolean> / IS NOT <boolean>"

4.8.0 GA – version 1.0 (2015.06.01)

- General changes reflecting change of "brighthouse" to "infobright" and in particular "engine=brighthouse" to "engine=infobright"
- General changes reflecting change to new configuration file infobright.cnf (which replaces .infobright and brighthouse.ini)
- Changed "Technical Requirements" section to indicate support for RHEL 7 and CentOS 7

- Changed "Windows Installation Instructions" section to add instructions for how to create the infobright.cnf file and how to obtain an Infobright license
- Changed "Uninstalling on Windows" section to add "Note" indicating that an uninstall will not delete the data directory
- Changed "Linux RPM and DPKG Installation Instructions" section to add instructions for how to create the infobright.cnf file and how to obtain an Infobright license
- Changed "Uninstalling on Linux" section to add "Note" indicating that an uninstall will not delete the data directory
- Changed "Windows Upgrade Instructions" section to add instructions for how to create the infobright.cnf file, how to convert previous parameter overrides, how to update Column Optimizer triggers and stored procedures, and how to obtain an Infobright license
- Changed "Linux RPM and DPKG Upgrade Instructions" section to add instructions for how to create the infobright.cnf file, how to convert previous parameter overrides, how to update Column Optimizer triggers and stored procedures, and how to obtain an Infobright license
- Changed "Configuration" section significantly, adding sub-sections "Configuring Infobright (infobright.cnf file)", "Instructions For How To Override Infobright Parameter Default Values", "Detailed Infobright Parameter Descriptions", and "Cross-Reference of Pre-4.8.0 Parameters To Current Parameters"
- Added the section "The Infobright License File"
- Changed "About Log Files" section significantly, adding sub-sections "The Infobright Log (infobright.log), "Other Postgres Logs", "Log Rotation", "Changing the infobright.log Log Level, and FET (Function Execution Time) Logging
- Changed "About the Infobright Database Files" section, updating the example showing contents of the ib-data directory
- Changed "Creating and Dropping Tables" section to indicate that the new syntax for creating an Infobright table is "engine=infobright" (replacing "engine=brighthouse")
- Changed the "LOOKUP Columns" section (including the name of the section) to indicate the functionality is now called "LOOKUP columns" (instead of "DIMENSION columns")
- Changed the "About Importing and Exporting Data" section to indicate that what previously was referred to as "Distributed Load Processor" is now called "DLP"
- Changed "Infobright COPY FROM Syntax" section to add the option (and sub-section) "accept_missing_columns" and remove the options "pipe-mode" and "timeout", and to consistently refer to "options" (instead of "parameters")
- Changed "Importing Files with Invalid Values" section to remove the options "pipe-mode" and "timeout"
- Changed "Infobright COPY TO Syntax" section to remove the options "pipe-mode" and "timeout"
- Changed the "About Knowledge Nodes" section to remove mention of the "Pack/Pack" Knowledge Node Type (which no longer exists)
- Added the "Rough Queries" section
- Added the "Backup Procedure" and "Restore Procedure" sections
- Changed the "Infobright Column Optimizer" section (including the name of the section) to indicate the functionality is now called "Column Optimizer" (instead of "Domain Expert")

- Changed “Ensure CacheFolder is on a Fast Local Disk” to now provide a link to “CacheFolder” section (instead of “Infobright Tuning Parameters” which was removed)
- Changed “Infobright Consistency Manager” section to reference “LOOKUP columns” (instead of “DIMENSION columns”)
- Changed “Infobright MySQL to PostgresSQL Migrator (“External Migrator”) section to remove the “Limitations and Notes” bullet referencing LOOKUP columns

4.7.1 GA – version 1.1 (2015.02.20)

- Added “Document Change Log” section
- Updated “IEE-Postgres Upgrading” section to add detailed Postgres upgrade instructions
- Updated (and changed name) of “A. Infobright Optimizer – Supported Functions and Operators” section to accurately reflect functions and operators that will be executed by the IBDB for Postgres engine.