Release Notes

Product	VoltDB
Version	4.3
Release Date	May 9, 2014

This document provides information about known issues and limitations to the current release of VoltDB. If you encounter any problems not listed below, please be sure to report them to support@voltdb.com. Thank you.

Upgrading From Older Versions

When upgrading from a previous version of VoltDB — especially with an existing database — there are a number of important notes that you should be aware of. Some changes to the structure and syntax of the VoltDB schema and deployment files may make old application catalogs and configuration files incompatible with newer versions.

Although incompatible changes are avoided wherever possible, some changes are necessary to add new features. It is always recommended that applications catalogs be recompiled when upgrading the VoltDB version. It is also important to note that the catalog is saved as part of snapshots and command logging. As a consequence, you must be careful to ensure an incompatible catalog is not loaded accidentally by starting a database with the **recover** action after an upgrade.

The process for upgrading VoltDB for a running database is as follows:

1. Place the database in admin mode (using **voltadmin pause** or the VoltDB Enterprise Manager).
2. Perform a manual snapshot of the database (using **voltadmin save**).
3. Shutdown the database (using **voltadmin shutdown**).
4. Upgrade VoltDB.

   If you are upgrading to a version prior to 4.2, you must also recompile your application catalog using the new version before restarting VoltDB. However, for version 4.2 and later, VoltDB automatically recompiles old catalogs when you create a new database, eliminating the need for this extra step.
5. Restart the database using the **create** option, the existing application catalog, and starting in admin mode (specified in the deployment file).
6. Restore the snapshot created in Step #2 (using **voltadmin restore**).
7. Return the database to normal operations (using **voltadmin resume**).

Again, starting with VoltDB 4.2 you no longer need to recompile your application catalog. The **voltdb create** command does this automatically for catalogs created with earlier versions.

When using the Enterprise Manager, it is also recommended that you delete the Enterprise Manager configuration files (stored by default in the `.voltdb` subfolder in the home directory of the current account) when performing an upgrade.

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that might impact their existing applications. Users of pre-V4 releases should pay special attention to the upgrade instructions for V3 users available as a separate document.

1. Release V4.3

1.1. Export to Kafka

VoltDB now includes export to Apache Kafka as a standard export client. Kafka export was added as a software preview several releases ago. You can now select and configure Kafka export in the deployment file or through the REST interface. (Kafka export is not accessible from the VoltDB Enterprise Manager at this time.) See the chapter on export in the *Using VoltDB* manual for details.

1.2. Kerberos Security

VoltDB now allows you to use Kerberos to authenticate Java clients to the VoltDB server. Kerberos security in VoltDB supports the same users, roles, and permissions defined in the deployment file and schema as with normal security, but uses the Kerberos authentication protocols to identify authorized clients to the database servers. Kerberos authentication is limited to Java clients only.

An explanation of how to implement Kerberos security in VoltDB will be added to the security chapter of the *Using VoltDB* manual shortly after version 4.3 releases.

1.3. Changes to VARCHAR

In previous versions, the length of a VARCHAR column was defined in bytes. Starting with 4.3, the length of a VARCHAR column is now declared in characters rather than bytes. This change is made for compliance with the SQL standard and for improved handling of multi-byte UTF-8 character sets.

Three major effects of this change are:

- Now a VARCHAR defined as a maximum number of characters can hold that many characters, no matter what character set they represent.
- Schemas containing short VARCHAR columns (less than 16 characters) will consume more space than in previous versions.
- Columns defined as between 16 and 63 characters that were previously stored inline will now be stored in pooled memory. This data may or may not consume more memory, depending on actual size, since strings stored in pooled memory require only the necessary pointers plus the actual memory required to store the data. More importantly, accessing these columns requires indirection that incrementally impacts performance.

The impact on strings defined as less than 16 characters results from short VARCHAR columns being stored inline as their maximum possible length. Where previously VARCHAR(15) would consume 15 bytes, now it will consume four bytes for every character, or 60 bytes. For longer VARCHARS, the strings are stored in pooled memory as their actual length, so there is no change to the memory they require.

The increased memory consumption, especially for schemas with many of short VARCHARS, could impact the ability to restore snapshots created in previous versions of the product, if memory usage is an issue.

It is possible to reproduce the previous behavior in VARCHAR declarations by including the keyword BYTES. For example VARCHAR(64 BYTES).

1.4. Simplifying the configuration and starting of clusters

Two changes have been made to simplify the process for configuring and starting VoltDB clusters.

- First, `sitesperhost` is now an optional attribute in the deployment file. If you do not specify a value for `sitesperhost`, a default of eight sites per host is used. Testing has shown this default value is effective for most systems and only needs changing for optimizing systems with very large numbers of processors.
- When starting a cluster using the VoltDB Enterprise Edition, VoltDB now searches three locations for the license file: the current working directory, the directory where the VoltDB software resides, and the user's home directory. This means that if you put your license file in your home directory, you do not need to use the `--license` flag when starting VoltDB, even when upgrading, working in multiple project directories, etc.

1.5. New SQL string functions

Several new SQL string functions have been added to simplify coding:

CHAR(*integer*)
OVERLAY(*string* PLACING *string* FROM *integer* [FOR *integer*])
REPLACE(*string*, *string* [, *string*])

1.6. System procedure to stop a single node in a cluster

The @StopNode system procedure let's you stop the VoltDB process on an individual member node of a VoltDB cluster in an orderly way. You specify the host ID of the node you want to stop as an argument to the @StopNode procedure.

You can use @StopNode to remove a node from the cluster for hardware upgrades or other maintenance, then return the node to the cluster with **voltdb rejoin**. Note that the @StopNode procedure only works if the cluster is K-safe and stopping the node will not stop the cluster itself. In other words, the cluster must remain viable after the system procedure executes. You cannot use @StopNode to shutdown the cluster.

1.7. JDBC improvements

Improvements continue to be made to the JDBC interface, focusing on reliability and extended functionality. This release fixes an issue where the connection would fail if the connection string included an inaccessible server.

1.8. SQL improvements

This version also includes a number of changes to SQL parsing to ensure correctness and proper index usage, specifically in edge cases related to complex joins and order by clauses.

1.9. csvloader improvements

The csvloader now provides additional context when reporting errors in the input file, making it easier to debug and correct the errors by identifying the specific input field that causes the error.

1.10. Bug fixes

In addition to the preceding new features and enhancements, a number of known issues have been corrected, including:

- Previously, when firehosing a server, the Web Studio interface would become unresponsive. The priority of the HTTP interface has been adjusted to avoid this condition.
- Under certain conditions, the database process on a cluster node might crash, claiming that transactions were "moving backwards". This was a rare but recurring bug which has now been fixed.

2. Release V4.2

2.1. The **voltdb create** and **voltadmin update** commands automatically recompile old catalogs

In previous releases, the server process would not start with a catalog compiled by an earlier version of VoltDB. Starting with 4.2, when you create a new database using the **voltdb create** command and an old catalog, VoltDB automatically recompiles the catalog before starting the server.

This means you can upgrade VoltDB versions without manually recompiling the catalog. The same is true when updating the catalog on a running database with the **voltadmin update** command or @UpdateApplicationCatalog system procedure. Note, however, using older catalogs does not currently work with the **voltdb recover** command or the VoltDB Enterprise Manager.

If the catalog is old enough to contain outdated or no longer valid schema syntax, VoltDB reports an error and either stops (in the case of **voltdb create**) or cancels the update (in the case of **voltadmin update**). In this case, you must update the schema source file and recompile the catalog yourself.

2.2. New LIMIT PARTITION ROWS constraint when defining tables

The VoltDB compiler now supports a new constraint, LIMIT PARTITION ROWS, that lets you limit the size of individual tables. The LIMIT PARTITION ROWS constraint is declared in the CREATE TABLE statement and limits the number of rows that can be inserted into any partition for that table. See the *Using VoltDB* manual for details.

2.3. Support for subqueries in SELECT

The SELECT statement now supports subqueries as table references in the FROM clause. For the initial release subqueries have certain constraints:

- The subquery must be enclosed in parentheses and assigned an alias.
- Ad hoc and multi-partition SELECT statements containing subqueries can operate on replicated tables only. They cannot contain references to partitioned tables.
- However, SELECT statements with subqueries in single-partition stored procedures can operate on both partitioned and replicated tables.

See the documentation of the SELECT statement in the *Using VoltDB* manual for details.

2.4. Ability to specify the network interface for individual ports

Previously, you could specify a port number for each port when starting VoltDB and a separate network interface for internal versus external ports. It is now possible to specify both the interface and the port number for individual ports when starting VoltDB. For example, the following command specifies the network interface 15.16.2.24 and the port number 21212 for the client port but the default external interface and port 31313 for the admin port.

```
$ voltdb create voter.jar --client=15.16.2.24:21212 --admin=31313
```

Note that the `--internalinterface` and `--externalinterface` flags are still available and set the default interfaces, as before. When you specify both a default interface for a collection of ports and a specific interface for an individual port, the port-specific setting overrides the default setting. It is also now possible to specify the http port (and, optionally, its interface) on the command line using the `--http` flag.

2.5. Control of elastic rebalance moved to deployment file

The interface to control the rebalance operations when nodes are added to an elastic cluster have changed from using Java system properties to elements and attributes in the deployment file. You can now adjust the length and size of rebalance operations using the attributes `duration` and `throughput` of the `<elastic>` element in the deployment file. These attributes replace the Java system properties ELASTIC_TARGET_TRANSFER_TIME_MS and ELASTIC_TARGET_THROUGHPUT. For example:

```
<deployment>
   . . .
   <systemsettings>
       <elastic duration="15" throughput="1"/>
   </systemsettings>
</deployment>
```

See the section on "Configuring How VoltDB Rebalances New Nodes" in the *Using VoltDB* for details.

2.6. Rebalance performance improvements

Previously, elastic rebalancing would occupy all partitions in the cluster. With this release, each rebalance operation only uses those partitions it needs, freeing up any remaining partitions for other database transactions. This change does not improve the performance of the rebalance itself, but can significantly reduce the impact rebalance has on ongoing client transactions.

2.7. Latency improvements during operational activities

This release contains a number of improvements to reduce latency spikes during operational activities such as snapshots, rejoin, and export. In addition, Java heap usage during export and rejoin has been reduced.

2.8. JDBC improvements

This release includes a number of improvements to the JDBC interface, including extended support for returning metadata, the SetFloat() method, and automatic conversion of numeric values to strings for VARCHAR columns.

2.9. Additional memory protection for command logging

In extreme cases, where the disks used for command logging cannot keep up with the write requests coming from VoltDB, the logging packets start to fill up memory. If this condition persisted, it could result in the server process running out of Java heap space. Command logging now includes a back pressure mechanism that will slow the processing of VoltDB transactions if the command logs begin to back up, as a way to avoid this situation.

2.10. Server and cluster shutdown improvements

Previously, the @Shutdown system procedure (and v**oltadmin shutdown** command) simply stopped the database process. The result was similar to a node failing, generating error messages and crash logs as the nodes stopped. User-requested shutdown is now handled as a synchronized event within the cluster, eliminating misleading error messages and unnecessary log files. This is the first step in an effort to provide a more orderly shutdown behavior.

In addition to cluster shutdown improvements, now when you stop a server process with CTRL-C (or, more specifically, the Unix signal SIGINT), the cluster performs a more orderly removal of that node from the cluster, rather than handling the event as an unexpected crash. In this way, removing a server from a K-safe cluster (for maintenance or replacement) is faster and less disruptive of ongoing transactions.

2.11. Better csvloader performance on clusters with large partition counts

Previously, the csvloader utility provided good performance on small and medium-sized systems. But performance would drop off on clusters with a large number of partitions. This bug has been corrected and csvloader provides scalable performance for loading partitioned tables into different size clusters.

2.12. Additional improvements

- Two new SQL string functions, UPPER() and LOWER(),
- The server uptime is now included in the result set of the @SystemInformation system procedure.
- A database sizing worksheet is included in the catalog report generated when you compile an application catalog. The worksheet is also accessible from a running server at the URL http:/*server*:8080/report.
- The latency graphs in the VoltDB Enterprise Manager are now more responsive to application behavior, displaying latency with a finer degree of granularity.
- For latency-sensitive applications, the setCallProcedureTimeout method now lets you set timeouts less than one second.

2.13. Bug fixes

In addition to the preceding new features and enhancements, a number of known issues have been corrected, including:

- Very large queries (greater than 6 Kilobytes) failed in web studio because the HTTP interface could not handle that much data in the request header. Large queries are now submitted in the body of the request rather than in the header.
- Previously it was not possible to add servers "on the fly" to an elastic cluster with no partitioned tables. This bug has been fixed and you can now add servers to a cluster, regardless of its schema.
- An issue where the JSON interface failed with a "no connections" error and could not be revived (usually when a laptop server was put to "sleep") has been corrected. The JSON interface is now self-correcting in this situation.
- Similarly, when multiple JDBC clients were accessing VoltDB and the JDBC interface lost its connection to the server, it would not reconnect and the client applications would have to restart to reconnect. The JDBC interface now reconnects without having to restart the client applications.

3. Release V4.0.2.3

3.1. "Admission control error" fixed

There was an issue in earlier releases where VoltDB could miscalculate the outstanding transactions. Two consequences of this situation were that the database server would issue an "admission control" error stating that there was a negative outstanding transaction byte count or client applications encountered connectivity issues. This problem has now been corrected.

3.2. Queued export data is maintained when cycling all servers

In previous releases, if all servers in the cluster failed and rejoined without stopping the database itself, data waiting in the export queue could be lost. This only happened if all servers in the cluster were cycled (stopped and rejoined). The cause of this problem has now been corrected.

4. Release V4.0.2

4.1. Support for running the VoltDB server process in the background

When starting the VoltDB server process from the command line (using the **voltdb create**, **add**, **recover**, or **rejoin** command) you can use the `-B` or `--background` flag to specify that the process run in the background.

4.2. Client timeout extended

When the VoltDB server does not receive a response from a client connection for a set amount of time, the server times out and closes the connection. The client timeout period has been extended from 4 to 30 seconds so connections are more resilient to network issues.

4.3. Data loaded in csvloader batch mode is compatible with command logging

In recent releases, performance of the csvloader utility was improved by introducing batch mode. However, batch mode inserts were not immediately recognized or recorded by the command logs. As a result, data loaded using csvloader batch mode did not become durable until the next snapshot occurred. This issue has been resolved and all data loaded with csvloader is now immediately durable.

4.4. The TRUNCATE TABLE statement optimized to improve performance and reduce memory usage

TRUNCATE TABLE, and its equivalent statement DELETE with no WHERE clause, have been optimized to significantly improve performance and reduce the amount of memory used during execution.

4.5. Join order is no longer case sensitive

In early releases of V4, when specifying join order for a query, the table names had to be in all uppercase. This issue has been resolved and the table names are no longer case sensitive.

4.6. Restore could fail on clusters with large numbers of partitions

There was an issue where attempting to restore a snapshot on a cluster with a large number of tables and partitions could fail, reporting an error that the "next message length" was too long. This problem has been resolved.

4.7. Support for Groovy inline stored procedures

It is now possible to declare complex stored procedures inline in the schema using the CREATE PROCEDURE AS statement and embedded Groovy code. See the *Using VoltDB* manual for details..

5. Release V4.0.1

5.1. Further testing and hardening of the new elastic functionality

A number of issues have been discovered and resolved in the elastic cluster functionality that is introduced in version 4.0. In particular, edge cases related to error conditions when nodes fail during elastic scaling have been identified and corrected.

5.2. Java client improvements

This release includes several improvements to the Java client, including:

- The client is shipped as a single JAR file with no external dependencies
- The client JAR is backwards compatible with Java 6 (although the VoltDB server now requires Java 7)
- All status information concerning failed procedure calls is now consolidated in the two ClientResponse methods getStatus() and getStatusString()

As a consequence of this last change, the method ClientResponse.getException() has been removed from the client API. Also, the causedBy property of ProcCallExceptions no longer returns an exception, All underlining exception information is returned as text by the getStatusString() method.

5.3. JDBC improvements

A number of improvements have been made to the JDBC interface as well. Similar to the Java API, the JDBC interface is provided as a single JAR file with no external dependencies. This means that if you use Guava and depended on the Guava library provided by VoltDB, you must either provide your own Guava JAR file or change the dependency to "com.google_voltpatches.common.*".

Other improvements to the JDBC interface include:

- Support for PreparedStatement.setQueryTimeout()
- Support for PreparedStatement.setString() for all VoltDB column types
- Support for DatabaseMetadata.getTypeInfo()

5.4. SQL support for CASE expressions

VoltDB now supports the CASE-WHEN-THEN-ELSE-END syntax in queries. For example:

```
SELECT Prod_name,
    CASE WHEN price > 100.00
          THEN `Expensive`
          ELSE `Cheap`
    END
FROM products ORDER BY Prod_name;
```

5.5. SQL support for HAVING with aggregate functions

VoltDB now supports the use of aggregate functions in the HAVING clause. For example:

```
SELECT game_id, count(*) FROM games
  GROUP BY game_id
  HAVING count(*) > 1;
```

5.6. Default Java heap size increased

The default Java heap size for the VoltDB server process has been increased from 1GB to 2GB. The new default more closely matches recommended settings for general purpose usage, More detailed recommendations can be found in the revised "Server Process Memory Usage" section of the *VoltDB Planning Guide*.

5.7. Recovery issues with resized clusters

In previous versions there was an issue where, if a cluster was reduced in size and then restored from snapshots, future command logs of the cluster could not be recovered. One symptom of this issue is the fatal error "No viable snapshots to restore" during recovery.

The problem only occurs if the number of unique partitions in the cluster was reduced, either by reducing the number of servers, reducing the sites per host, or increasing the K-safety factor. With this release, the issue has been corrected. The issue is resolved for any affected databases by following the instructions for upgrading in the previous section; specifically, saving a snapshot, upgrading the software to VoltDB 4.0.1 or later, then restoring the snapshot.

6. Release V4.0

6.1. New Features

VoltDB 4.0 is a major release. It consolidates and completes many features introduced in preceding releases, including elasticity and changes to the user interface to improve overall ease of use of the product. Benefits of VoltDB 4.0 include:

- Elasticity — the ability to add nodes to the database cluster "on the fly" — with support for all standard features including K-safety, command logging, and export.
- New SQL support including:
  - Improved use of indexes
  - Performance improvements for views
  - More robust support for expressions in indexes, functions, and clauses
  - Support for inner, outer, and self joins
- Improved server-based export
- A new, more consistent command line interface (CLI) for starting the database cluster

6.2. Bug fixes

In addition to the new features and enhancements listed above, VoltDB V4.0 includes fixes to a number of limitations in previous versions, including the following:

- Previously, if a node rejoined the cluster and then the cluster stopped before the node could process any transactions, the command logs could not be recovered. This issue has been resolved.
- Several issues related to comparisons of or aggregate functions involving null values, which could produce incorrect results, have been fixed.
- Previously, setting the external interface did not change the interface used by the HTTP port. The HTTP port now uses the external interface specified on the command line.
- There was an issue where a join with multiple WHERE constraints, one of which was an IN list function evaluated against one column of a multi-column index, would ignore the IN list restriction. This issue has been corrected.
- Memory management within the csvloader utility has been improved, eliminating out of memory errors that were seen in earlier releases.

Known Limitations

The following are known limitations to the current release of VoltDB. Workarounds are suggested where applicable. However, it is important to note that these limitations are considered temporary and are likely to be corrected in future releases of the product.

1. Command Logging

1.1. Command logs can only be recovered to a cluster of the same size.

To ensure complete and accurate restoration of a database, recovery using command logs can only be performed to a cluster *with the same number of unique partitions* as the cluster that created the logs. If you restart and recover to the same cluster with the same deployment options, there is no problem. But if you change the deployment options for number of nodes, sites per host, or K-safety, recovery may not be possible.

For example, if a four node cluster is running with four sites per host and a K-safety value of one, the cluster has two copies of eight unique partitions (4 X 4 / 2). If one server fails, you cannot recover the command logs from the original cluster to a new cluster made up of the remaining three nodes, because the new cluster only has six unique partitions (3 X 4 / 2). You must either replace the failed server to reinstate the original hardware configuration or otherwise change the deployment options to match the number of unique partitions. (For example, increasing the site per host to eight and K-safety to two.)

1.2. Do not use the subfolder name "segments" for the command log snapshot directory.

VoltDB reserves the subfolder "segments" under the command log directory for storing the actual command log files. Do not add, remove, or modify any files in this directory. In particular, do not set the command log snapshot directory to a subfolder "segments" of the command log directory, or else the server will hang on startup.

2. Database Replication

2.1. Node failure and rejoin on the replica during csvload operations can cause uncaught data duplication

If a node on the replica database fails while the master is loading data with the csvloader (or its associated bulk loading methods), when the node rejoins it is possible data already loaded gets reloaded during the rejoin. This can cause divergence between the master and replica databases.

To be safe until this limitation is corrected, if a node on the replica database fails while the master database is bulk loading data, you should stop the replica and the DR agent and restart replication once the bulk load is complete.

2.2. The Enterprise Manager cannot restart and recover a replica database as a master.

Using the VoltDB Enterprise Manager, if a replica database was started with command logging, then stopped (intentionally or by accident), the Enterprise Manager cannot restart the database as a normal database using the recover action to reinstate the database's previous state. The Enterprise Manager *can* restore from a snapshot.

If you want to use the Enterprise Manager to stop a replica and restart it as a normal database, the recommended procedure is:

1. Stop replication.
2. Pause the replica.
3. Use the Enterprise Manager to take a manual snapshot.
4. Stop the database.
5. Start the database, choosing "restore from snapshot" as the startup action and the manual snapshot as the source.

Note that this limitation is specific to the Enterprise Manager. Failed replica databases can be recovered manually using the command line.

3. Export

3.1. Kafka export does not support VARBINARY columns.

The Kafka export client does not support exporting VARBINARY data. If an export table containing a VARBINARY column is sent to the kafka export client, incorrect data will be exported in place of the actual column contents.

The workaround is to define the export column as VARCHAR rather than VARBINARY and BASE64 encode the binary data before inserting it into the export table.

4. SQL and Stored Procedures

4.1. SELECT DISTINCT using multiple columns or expressions is not supported.

Use of SELECT DISTINCT is supported for a single column (such as `SELECT DISTINCT Price FROM Inventory`). However, using DISTINCT with multiple columns or arithmetic expressions is not currently supported. For example, the following SELECT DISTINCT statements should not be used:

```
SELECT DISTINCT Price, Discount FROM Inventory
SELECT DISTINCT (Price - Discount) FROM Inventory
```

4.2. Do not use assertions in VoltDB stored procedures.

VoltDB currently intercepts assertions as part of its handling of stored procedures. Attempts to use assertions in stored procedures for debugging or to find programmatic errors will not work as expected.

4.3. The UPPER() and LOWER() functions currently convert ASCII characters only.

The UPPER() and LOWER() functions return a string converted to all uppercase or all lowercase letters, respectively. However, for the initial release, these functions only operate on characters in the ASCII character set. Other case-sensitive UTF-8 characters in the string are returned unchanged. Support for all case-sensitive UTF-8 characters will be included in a future release.

5. Client Interfaces

5.1. Avoid using decimal datatypes with the C++ client interface on 32-bit platforms.

There is a problem with how the math library used to build the C++ client library handles large decimal values on 32-bit operating systems. As a result, the C++ library cannot serialize and pass Decimal datatypes reliably on these systems.

Note that the C++ client interface *can* send and receive Decimal values properly on 64-bit platforms.

6. Runtime Issues

6.1. VoltDB does not currently support Java 8.

As with all required infrastructure software, VoltDB undergoes extensive testing to verify proper operation with each Java release. The initial release of Java 8 was recently made public. It has not been fully tested with VoltDB and there is at least one known issue that could affect VoltDB. For this reason, use of VoltDB on Java 8 is *not* recommended at this time.

6.2. Partially removing snapshot files from the database servers can cause recovery to fail.

To ensure proper recovery on startup, either from command logs or the last database snapshot, make sure all snapshot files — or at least complete subsets of the snapshot files — are available on the nodes of the cluster. If you delete or move snapshot files (for example, copying all snapshot files to a single node) be sure to keep all of the files for each node together. Do not selectively delete or move individual files or else the recovery may fail.

6.3. VoltDB will not start if the user does not have execute privileges to the /tmp directory.

If the HTTP port is enabled (which it is by default) but the process does not have execute privileges for the /tmp directory, VoltDB throws a fatal exception on startup. The error message indicates that the process could not load the native library for the Snappy web server.

The workaround is to either use an account that has execute permission for the /tmp directory or specify an alternate directory that the account can access by assigning the environment variable VOLTDB_OPTS = "-Djava.io.tmpdir={alternate-tmpdir}".

6.4. Under certain conditions, automated snapshots may stop occurring without notice.

Automatic snapshots (enabled through the deployment file) are intended to create periodic snapshots on a scheduled basis while the database is running. However, there are instances when the automatic snapshots stop occurring. This problem is sporadic and only happens on K-safe clusters where nodes fail and rejoin.

Once automatic snapshots stop, they will not restart until the cluster stops and restarts. The workaround is, after a node fails and rejoins, to check the logs to make sure automatic snapshots are still occurring. If not you can create a cron job to periodically take snapshots manually using the **voltadmin save** command until the next time the cluster reboots.

7. Enterprise Manager

7.1. Manual snapshots not copied to the Management Server properly.

Normally, manual snapshots (those created with the **Take a Snapshot** button) are copied to the management server. However, if automated snapshots are also being created and copied to the management server, it is possible for an automated snapshot to override the manual snapshot.

If this happens, the workaround is to turn off automated snapshots (and their copying) temporarily. To do this, uncheck the box for copying snapshots, set the frequency to zero, and click **OK**. Then re-open the Edit Snapshots dialog and take the manual snapshot. Once the snapshot is complete and copied to the management server (that is, the manual snapshot appears in the list on the dialog box), you can re-enable copying and automated snapshots.

7.2. Old versions of Enterprise Manager files are not deleted from the /tmp directory

When the Enterprise Manager starts, it unpacks files that the web server uses into a subfolder of the /tmp directory. It does not delete these files when it stops. Under normal operation, this is not a problem. However, if you upgrade to a new version of the Enterprise Edition, files for the new version become intermixed with the older files and can result in the Enterprise Manager starting databases using the wrong version of VoltDB. To avoid this situation, make sure these temporary files are deleted before starting a new version of VoltDB Enterprise Manager.

The /tmp directory is emptied every time the server reboots. So the simplest workaround is to reboot your management server after you upgrade VoltDB. Alternately, you can delete these temporary files manually by deleting the winstone subfolders in the /tmp directory:

```
$ rm -vr /tmp/winstone*
```

7.3. Enterprise Manager configuration files are not upwardly compatible.

When upgrading VoltDB Enterprise Edition, please note that the configuration files for the Enterprise Manager are not upwardly compatible. New product features may make existing database and/or deployment definitions unusable. It is always a good idea to delete existing configuration information before upgrading. You can delete the configuration files by deleting the ~/.voltdb directory. For example:

```
$ rm -vr ~/.voltdb
```

7.4. Enterprise Manager cannot start two databases on the same server.

In the past, it was possible to run two (or more) databases on a single physical server by defining two logical servers with the same IP address and making the ports for each database unique. However, as a result of internal optimizations introduced in VoltDB 2.7, this technique no longer works when using the Enterprise Manager.

We expect to correct this limitation in a future release. Note that it is still possible to start multiple databases on a single server manually using the VoltDB shell commands.

Implementation Notes

The following notes provide details concerning how certain VoltDB features operate. The behavior is not considered incorrect. However, this information can be important when using specific components of the VoltDB product.

1. SQL

1.1. Do not use UPDATE to change the value of a partitioning column

For partitioned tables, the value of the column used to partition the table determines what partition the row belongs to. If you use UPDATE to change this value and the new value belongs in a different partition, the UPDATE request will fail and the stored procedure will be rolled back.

Updating the partition column value may or may not cause the record to be repartitioned (depending on the old and new values). However, since you cannot determine if the update will succeed or fail, you should not use UPDATE to change the value of partitioning columns.

The workaround, if you must change the value of the partitioning column, is to use both a DELETE and an INSERT statement to explicitly remove and then re-insert the desired rows.

1.2. Certain SQL syntax errors result in the error message *"user lacks privilege or object not found"* when compiling the runtime catalog.

If you refer to a table or column name that does not exist, the VoltDB compiler issues the error message *"user lacks privilege or object not found"*. This can happen, for example, if you misspell a table or column name.

Another situation where this occurs is if you mistakenly use double quotation marks to enclose a string literal (such as `WHERE ColumnA="True"`). ANSI SQL requires single quotes for string literals and reserves double quotes for object names. In the preceding example, VoltDB interprets "True" as an object name, cannot resolve it, and issues the "user lacks privilege" error.

The workaround is, if you receive this error, to look for misspelled table or columns names or string literals delimited by double quotes in the offending SQL statement.

2. Runtime

2.1. File Descriptor Limits

VoltDB opens a file descriptor for every client connection to the database. In normal operation, this use of file descriptors is transparent to the user. However, if there are an inordinate number of concurrent client connections, or clients open and close many connections in rapid succession, it is possible for VoltDB to exceed the process limit on file descriptors. When this happens, new connections may be rejected or other disk-based activities (such as snapshotting) may be disrupted.

In environments where there are likely to be an extremely large number of connections, you should consider increasing the operating system's per-process limit on file descriptors.

3. Recovery

3.1. Troubleshooting Recovery Issues

There are several situations where an attempt to recover a database — either from a snapshot or command logs — may fail. For example, restoring a snapshot where a unique index has been added to a table can result in a constraint violation that causes the restore, and the database, to fail. Similarly, a command log may contain a transaction that originally succeeded but fails and raises an exception during playback.

In both of these situations, VoltDB issues a fatal error and stops the database to avoid corrupting the contents.

Although protecting you from an incomplete recovery is the appropriate default behavior, there may be cases where you want to recover as much data as possible, with full knowledge that the resulting data set does *not* match the original. VoltDB provides two techniques for performing partial recoveries in case of failure:

- Logging constraint violations during snapshot restore
- Performing command log recovery in safe mode

Logging constraint violations — There are several situations that can cause a snapshot restore to fail because of constraint violations. Rather than have the operation fail as a whole, you can request that constraint violations be logged to a file instead. This way you can review the tuples that were excluded and decide whether to ignore or replace their content manually after the restore completes.

To perform a manual restore that logs constraint violations rather than stopping when they occur, you use a special JSON form of the @SnapshotRestore system procedure. You specify the path of the log files in a JSON attribute, `duplicatePaths`. For example, the following commands perform a restore of snapshot files in the directory `/var/voltdb/snapshots/` with the unique identifier `myDB`. The restore operation logs constraint violations to the directory `/var/voltdb/logs`.

```
$ sqlcmd
1> exec @SnapshotRestore '{ "path":"/var/voltdb/snapshots/",
                            "nonce":"myDB",
                            "duplicatesPath":"/var/voltdb/logs/" }';
2> exit
```

Constraint violations are logged as needed, one file per table, to CSV files with the name `{table}-duplicates-{timestamp}.csv`.

Safe Mode Recovery — On rare occasions, recovering a database from command logs may fail. This can happen, for example, if a stored procedure introduces non-deterministic content. If a recovery fails, the specific error is known. However, there is no way for VoltDB to know the root cause or how to continue. Therefore, the recovery fails and the database stops.

When this happens, VoltDB logs the last successful transaction before the recovery failed. You can then ask VoltDB to recover up to but not including the failing transaction by performing a recovery in *safe mode*.

You request safe mode by adding the **--safemode** switch to the command line when starting the recovery operation, like so:

```
$ voltdb recover --safemode -license ~/license.xml
```

When VoltDB recovers from command logs in safe mode it enables two distinct behaviors:

- Snapshots are restored, logging any constraint violations

- Command logs are replayed up to the last valid transaction

This means that if you are recovering using an automated snapshot (rather than command logs), you can recover some data even if there are constraint violations during the snapshot restore. Also, when recovering from command logs, VoltDB will ignore constraint violations in the command log snapshot and replay all transactions that succeeded in the previous attempt.

It is important to note that to successfully use safe mode with command logs, you must perform a regular recovery operation first — and have it fail — so that VoltDB can determine the last valid transaction. Also, if the snapshot and the command logs contain both constraint violations and failed transactions, you may need to run recovery in safe mode twice to recover as much data as possible. Once to complete restoration of the snapshot, then a second time to recover the command logs up to a point before the failed transaction.

Warning

Finally, it is critically important to recognize that the techniques described above *do not* produce a complete copy of the original database or resolve the underlying problem that caused the initial recovery to fail These techniques should never be attempted without careful consideration and full knowledge and acceptance of the risks associated with partial data recovery.

4. Logging

4.1. All logging messages reported by the VoltDB server are timestamped using GMT (Greenwich Mean Time).

This is not a problem when looking at VoltDB logs separately. However, you should be aware of this distinction when integrating logging of VoltDB with logging of other system components that use the local time zone (rather than GMT). You may want to convert one or the other log streams so the time zones match.

4.2. To simplify logging, a file has been added to the distribution listing all of the VoltDB logging categories.

The file `voltdb/log4j.xml` lists all of the VoltDB-specific logging categories. It also serves as a useful logging schema. The sample applications and the VoltDB shell commands use this file to configure logging and it is recommended for new application development.

Software Previews

This release includes two new features that are currently under development. Although functional, we at VoltDB are still investigating the appropriate direction and level of completeness required for these features. We would appreciate feedback from the user community.

1.1. Migrate from MySQL to VoltDB

VoltDB includes a new utility, voltify, that helps you migrate from an existing MySQL database to VoltDB. The utility connects to a running MySQL database and creates a target schema and starter project in VoltDB to match the source database. If you are interested, you can find instructions for volity in the VoltDB github repository at the following URL:

```
http://github.com/VoltDB/voltdb/blob/master/tools/voltify-README.md
```

We encourage anyone who tries it to provide feedback in the VoltDB forums, http://forum.voltdb.com. Thank you.