Release Notes

Product	VoltDB
Version	4.9.7
Release Date	February 11, 2016

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.

Note: The VoltDB Enterprise Manager is part of the VoltDB Enterprise Edition and continues to be supported for customers who are currently using it. However, due to limitations in its implementation, no further development work is being done on the Enterprise Manager and it is not recommended for new deployments. The Enterprise Manager's functionality will be replaced by new, more robust, deployment and management capabilities in the future.

1. Release V4.9.7

The following issue was resolved in this release.

1.1. Race condition involving large deletes and snapshots

In rare cases, frequent deletes can cause memory compaction that coincides with a simultaneous snapshot. In the past, the result was that the servers would stop processing transactions and fill the log file with multiple messages reporting "tuple count > 0 after streaming". This error has been corrected. The triggering condition, although rare, can still occur. However, now if compaction coincides with an automated snapshot (such as a command log snapshot), the snapshot is requeued. If compaction coincides with a manual snapshot, the snapshot is canceled and reported to the user. If this occurs, simply reinitiate the manual snapshot.

2. Release V4.9.6

The following issues are resolved in this release.

2.1. Further improvements to DR buffering

Additional changes have been made to improve memory utilization for database replication (DR) buffering in virtualized environments and snapshot processing. The setting `-XX:MaxDirectMemorySize=<size>` is no longer required.

2.2. Timeout during DR initiation

There was an issue where, if the master database was under load, the DR agent would timeout attempting to retrieve the initial DR snapshot. This issue has been resolved by extending the timeout period on the DR agent from 5 seconds to 90 seconds and modifying the master database to issue periodic ACKs even when processing snapshots for DR initiation.

2.3. Race condition stops DR master database

Under certain circumstances, a race condition in database replication (DR) was found that could result in a node on the master cluster crashing with the error "java.lang.IllegalStateException: buffer is already compiled". This issue has been resolved.

3. Release V4.9.5

The following issues are resolved in this release.

3.1. Improved DR buffering

Changes have been made to DR buffering to use Java-managed DirectByteBuffers instead of unmanaged DirectByteBuffers. This change requires setting `-XX:MaxDirectMemorySize=<size>` in the

VOLTDB_OPTS. The recommended initial setting is a size of 4 Gigabytes, but this should be tested against the actual application and hardware to best determine the setting.

4. Release V4.9.4

The following issues are resolved in this release.

4.1. Exception when inputting decimal values with a scale greater than 12

Previously, when inserting values into a VoltDB DECIMAL datatype, any values with a scale greater than 12 (that is, more than 12 decimal places) were rejected and generated an exception in the JDBC interface. This behavior has been changed and VoltDB now rounds higher precision values to the closest 12 decimal digits.

4.2. Improved performance of JDBC prepared statements and ad hoc queries with parameters

When processing ad hoc queries that use parameters and placeholders (rather than a single SQL statement as a text string), VoltDB now makes use of previously cached queries to significantly improve performance for repeated queries. This can be most notable for JDBC prepared statements that are implemented as ad hoc queries with parameters.

5. Release V4.9.3

This release contains no new features but corrects the following issues from earlier releases.

5.1. Issues related to using INSERT INTO SELECT with export tables

There was an issue in earlier releases where using an INSERT INTO SELECT statement with an export table as the target for the insert either generated a null pointer exception or did not insert the expected data into the export stream. The issue only applies to INSERT INTO SELECT as an ad hoc query or within a multi-partitioned stored procedure.

- If the target of an INSERT INTO SELECT statement in a multi-partitioned query is an export table that is *not* partitioned, the planner would throw a null pointer exception (NPE).
- If the source of the INSERT INTO SELECT statement in a multi-partitioned query (that is, the table in the SELECT subquery) *is* a partitioned table, then an insert into an export table may not insert all of the selected rows.

These issues have now been corrected.

5.2. Database failure when reporting long-running queries

There was an issue in previous versions (starting with VoltDB 4.8), where if a query runs for a significant amount of time, VoltDB attempts to log a warning. However, the warning generates an error (index out of bounds) and stops the database.

This issue is now fixed.

5.3. Command log recovery hangs if the log contains a catalog update followed by an ad hoc INSERT, UPDATE or DELETE.

There was an issue in earlier releases where command log recovery would hang if the command logs contained an application catalog update (for example, from a **voltadmin update** command) followed by a subsequent ad hoc query that writes to the database. The recovery would either stop or silently fail to process the transaction when attempting to replay the ad hoc query.

This issue is now fixed.

6. Release V4.9.2

Important

Because of the serious nature of the issue with the JSON HTTP interface identified below, any customers using VoltDB 4.9 or 4.9.1 should either upgrade to 4.9.2 or should turn off the http port (using the `<httpd enable="false">` in the deployment file).

6.1. Bug fix: Leaving VoltDB Management Center connected to the server could hang the database

In previous releases, a defect in the JSON HTTP server could cause the server to stop handling client requests if the Management Center browser window was left open for a few hours. The more Management Center sessions in use, the sooner the problem would manifest itself.

If you have a server that has experienced this issue, you may stop the server and rejoin it to the cluster. Or you can stop the entire cluster and recover the database.

The problem is resolved in this release. Because this issue impedes normal operation of the database, we recommend any customers using VoltDB 4.9 or 4.9.1 update to VoltDB 4.9.2 as soon as possible.

6.2. Bug fix: **voltdb collect** fails with a traceback

Under certain conditions, the **voltdb collect** command could fail displaying a traceback reporting an attribute error. This problem has been resolved.

6.3. Support for CSV output when exporting to HttpFS from the HTTP connector

HTTP export now supports CSV output format when using HttpFS as a destination for WebHDFS export. To export CSV data to HttpFS, add the following properties to the export configuration in the deployment.xml:

```
<property name="httpfs.enable">true</property>
<property name="type">csv</property>
```

7. Release V4.9.1

7.1. Support for HttpFS added to the HTTP export connector

HTTP export now supports HttpFS as a destination for WebHDFS export using the Avro format. CSV format will be supported in the next major release. To enable exporting to HttpFS add the following properties to the export configuration in the deployment.xml:

```
<property name="httpfs.enable">true</property>
<property name="type">avro</property>
```

7.2. Bug fixes

The following bugs were fixed in this release:

- Previously, spaces and line feeds in the <export> configuration in the deployment file could cause it not to load. This issue has been corrected.
- Previously, the Nagios script 'check_voltdb_memory' would get an error when connected to localhost (`-H localhost`). VoltDB now returns an error with instructions on how to connect.

8. Release V4.9

8.1. VoltDB Management Center

VoltDB 4.9 introduces a new management console that is available from any running VoltDB database. The VoltDB Management Center integrates the catalog report and web studio functionality, plus the monitoring functions of the VoltDB Enterprise Manager, into a single interface. Available at http://server:8080 by default, the Management Center provides real-time monitoring, a schema browser, and interactive SQL queries.

8.2. Reorganization of security roles and permissions

Security has been revamped in this release, providing new permissions organized around functional capabilities. Security still operates on the basic principles of users assigned to roles in the deployment file and roles assigned permissions in the schema. However, the generic permissions (not associated with a specific stored procedure) have been reorganized and renamed to match logical functions. So, for example, there are named permissions for SQLREAD (that is, read-only ad hoc SQL queries), SQL (all SQL queries), ALLPROC (all stored procedures, including the default stored procedures), and ADMIN (full access, including read and modify system procedures).

For backwards compatibility, the previous named permissions ADHOC and SYSPROC are still permitted and map to the new SQL and ADMIN permissions. Note, however, that this mapping increases the permissions of the old permissions slightly, giving ADHOC access to default procedures and giving SYSPROC access to all user-defined stored procedures.

This release also includes two new predefined roles, ADMINISTRATOR and USER, that are always available and can be assigned to users in the deployment file. The ADMINSTRATOR role has ADMIN permissions and the USER role has SQL and ALLPROC permissions. In other words, the ADMINISTRATOR role has full access while the USER role can run any SQL query, stored procedure, or read-only system procedure, but cannot access system procedures that modify the database (such as @Shutdown and @SnapshotRestore).

See the chapter on security in the *Using VoltDB* manual for details.

8.3. Improvements to sqlcmd

The sqlcmd utility for entering interactive SQL queries from a command line is being overhauled. The first major addition is a new command line argument, `--stop-on-error`, that lets you specify whether a batch of commands — piped to the utility from the shell prompt or executed through the FILE directive — stops on the first error or continues to process subsequent commands. The default for the new argument is `--stop-on-error=true`, which corresponds to previous behavior.

8.4. Ability to time out long read queries

It is now possible to cancel read queries that take too long based on a predefined query timeout period. The query timeout property, which is a child of `<systemsettings>` in the deployment file, lets you set a timeout limit for read queries. Any read-only queries (or batch of queries submitted using voltExecuteSQL()) that exceed the limit will be cancelled and an exception returned to the calling application. You set the time out limit in milliseconds. For example, the following deployment file fragment sets the query timeout to 3 seconds:

```
<systemsettings>
   <query timeout="3000"/>
</systemsettings>
```

By default, there is no query timeout limit. See the section on database configuration options in the *Administrators Guide* for details.

8.5. Support for multiple column DISTINCT

Previously, VoltDB supported use of the DISTINCT keyword on a single column only. You can now use DISTINCT with multiple columns.

8.6. New Log4J category for logging all ad hoc queries

A new logger has been added to the VoltDB Log4J configuration that logs any ad hoc queries to the database. The new logger, called ADHOC, is not enabled by default, but can be enabled to record the text of the query and, if security is enabled, the user who submitted it.

8.7. Support for CentOS and Red Hat 7.0

VoltDB support for CentOS and Red Hat as base operating operating systems has been extended to include version 7.0.

8.8. Deprecation Notice

Support for several older technologies, obsolete synonyms for existing commands, and features that were superseded in previous releases will be removed in the next major release of VoltDB. The deprecated features that will be removed in VoltDB v5.0 include:

- Ubuntu 10.04 as a supported operating system
- Use of a project.xml file for defining the contents of the application catalog
- Use of "group" as a synonym for "role" when defining security policies in the schema and deployment file

In addition, use of Java 7 as the JDK will be supported but deprecated starting with VoltDB 5.0. This means Java 7 will be supported for the initial releases of VoltDB V5.0. However, use of Java 8 is recommended. Support for Java 7 will be removed from VoltDB some time after Java 7 reaches end-of-llife, currently scheduled for April 2015.

8.9. Bug fixes

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

- Previously, it was not possible to start multiple server processes using the `--background` argument to the voltdb command, because the PID files conflicted. This issue has been resolved and you can use `--background` to create more than one daemon process.

9. Release V4.8

9.1. Ability to select recent logs in voltdb collect

The v**oltdb collect** command has a new argument, `--days`, that lets you selectively collect only the most recent log files. For example, the command `voltdb collect --days=2` collects logs files from the last two days. If you do not specify the `--days` argument, the command uses a default value of 14 days.

9.2. Support for Avro format when exporting to Hadoop

The HTTP connector now supports writing the export files in the Apache Avro format. See the chapter on Export in the *Using VoltDB* manual for details.

9.3. New UPSERT statement

A new SQL statement, UPSERT is now supported. UPSERT has syntax similar to the INSERT statement, except it can only be used on tables that have a primary key. If a record with the specified key value doesn't exist, the UPSERT statement performs an insert operation; if such a record does exist, it performs an update operation. See the documentation for UPSERT for details.

9.4. New CPU selector for the @Statistics system procedure

The @Statistics system procedure has a new selector, CPU, which returns information on the percent of total available CPU used by the VotlDB server process for each server in the cluster.

9.5. New SET_FIELD() function

A new function has been added that lets you modify JSON fields within a SQL query. The SET_FIELD() function returns a copy of a JSON-encoded string with the specified field replaced. For example, the following SQL statement replaces the "statebird" field in the JSON-encoded column Description when the State column contains "New Hampshire":

```
UPDATE Region
   SET Description = SET_FIELD(Description,'statebird','"purple finch"')
   WHERE State = 'New Hampshire';
```

See the description of the SET_FIELD() function in the *Using VoltDB* manual for details.

9.6. Additional functions to support legacy JDBC applications

A number of aliases, common in JDBC interfaces, have been added to provide access to existing functions. New aliases include INSERT, LTRIM, RTRIM, SUBSTR, LCASE, and UCASE. These aliases are not currently listed in the documentation, but will be added in a future update.

9.7. Bug fixes

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

- Previously, if a JDBC client lost all of its connections to the database and could not successfully reconnect, the process would create an additional network thread with each attempted reconnect but failed to recover those threads, eventually running out of threads for the process. This error has been corrected.
- The HTTP export connector has been improved for better handling of file naming conflicts or missing files on the target system. Users interested in Hadoop export are encouraged to upgrade.
- There was an issue where the compiler reported an error if the optional third parameter to the SUBSTRING() function was missing.This issue has been fixed.

10. Release V4.7

10.1. New jdbcloader utility

The new jdbcloader copies the contents of a database table from a remote database using JDBC and inserts the records into a VoltDB table. Similar to the csvloader and kafkaloader utilities, jdbcloader provides yet another way to move data from other sources into VoltDB. See the jdbcloader reference page in the *Using VoltDB* manual for details.

10.2. New export connector for Hadoop (HTTP)

VoltDB now includes an export connector that lets you export data to Hadoop via the WebHDFS API. The HTTP connector can be configured to export data to WebHDFS URLs, including the table name and other metadata as part of the URL. See the chapter on Export in the *Using VoltDB* manual for details.

10.3. Improved FIELD() functionality

The FIELD() function has been improved to provide easier access to lower-level fields within JSON strings. Originally, the second argument to the FIELD() function specified the name of a top-level field to be returned. Starting with this release, you can specify a field name path, separating the levels with periods, so you can select sub fields. For example, if the Film column contains the following the JSON-encoded string:

```
{"title":"Mark of Zorro","year":"1920",
  "cast":{"Zorro":"Douglas Fairbanks",
          "Lolita":"Marguerite De La Motte",
          "Pedro":"Noah Beery"},
  "alternateTitles:["La marca del Zorro",
                    "Das Zeichen des Zorro",
                    "Zorros Kendemaerke",
                    "Il segno di Zorro"]
}
```

You can extract the top-level field containing "1920" using `FIELD(film,'year')` and the second-level field containing "Douglas Fairbanks" using `FIELD(film,'cast.Zorro')`. You can also use array notation, so the function `FIELD(film,'alternateTitles[3]')` returns the Italian title of the film. See the description of the FIELD() function in the *Using VoltDB* manual for details.

10.4. Ability to use INSERT INTO... SELECT in multi-partitioned procedures and queries

It is now possible to use the INSERT INTO... SELECT statement in multi-partitioned procedures and queries. This means you can also use the statement in ad hoc queries, including interactive queries run through the sqlcmd utility. See the description of INSERT in the *Using VoltDB* manual for further information.

10.5. The @Statistics memory report has been extended

New information has been added to the results of the MEMORY selector for the @Statistics system procedure. A new column (PHYSICALMEMORY) has been added reporting the total physical memory for each server in the cluster. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

10.6. Bug fixes

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

- In previous releases, VoltDB issued a runtime error if you tried to cast a small (less than 64 bytes) VARCHAR or VARBINARY expression to a larger VARCHAR or VARBINARY. This issue is now fixed.
- There was a small memory leak associated with multi-partitioned stored procedures that contained declarations for both READ and WRITE queries but only executed READ queries. Although the memory leak was small, it would add up when the procedure is called repeatedly. This problem is now resolved.
- There was a planning bug where if a query in a stored procedure contained both placeholders and a SELECT... UNION statement, at runtime VoltDB threw an exception stating that the number of arguments did not match in the HSQL-BACKEND. This error has been corrected.
- There was an issue where queries with very complex WHERE clauses (for example, more than 10 predicates) could be incorrectly planned and return more records than expected. This error has been corrected.

11. Release V4.6

11.1. Support for Ubuntu 14.04

VoltDB has been tested and validated on Ubuntu 14.04. The VoltDB server software now supports the Ubuntu long-term support releases 10.04, 12.04, and 14.04.

11.2. Native snapshots create a flag file on completion

When a native format snapshot completes, either for a snapshot initiated manually with **voltadmin SAVE** or periodic snapshots created by the system automatically , a flag file is now created to indicate the snapshot is done. The flag file is created in the same directory as the snapshot using the snapshot's unique identifier as the file name and `.finished` as the file extension. You can use the flag file to identify and backup or otherwise process completed snapshots.

11.3. New SQL functions

This release contains both enhancements to existing SQL functions as well as new functions, many provided for compatibility with other SQL implementations. Changes include:.

- Support for more than two arguments to the CONCAT function.
- Additional keywords to the EXTRACT function as well as standalone timestamp functions providing similar functionality in syntax compatible with MySQL and other dialects of SQL. New functions include SECOND(). MINUTE(), HOUR(), DAY(), WEEK(). MONTH(), QUARTER(), YEAR(), DAYOFMONTH(). DAYOFWEEK(), DAYOFYEAR(), WEEKDAY(), and WEEKOFYEAR().
- The addition of COALESCE() for compatibility with MySQL. COALESCE() returns the first non-null argument or null if all arguments are null.

See the appendix on SQL functions in *Using VoltDB* for more information.

11.4. Support for INSERT INTO... SELECT

VoltDB now supports the INSERT INTO... SELECT statement. For the initial release, this statement is limited to partitioned procedures only (it cannot be used in multi-partition stored procedures or as an ad hoc statement from the **sqlcmd** prompt). Broader usage will follow in future releases. See the documentation of INSERT in *Using VoltDB* for more information.

11.5. Support in Java client API for automatic reconnection of failed connections

This release includes support in the Java client API for enabling automatic reconnection of the client. By default, if a connection to a database server is lost, it is up to the client application to detect the lost connection and reconnect as appropriate. Now you can enable automatic reconnection as part of the client configuration. When enabled, auto-reconnect periodically attempts to reconnect to servers whose connection was lost. To enable auto-reconnect, you set a property on the client configuration before creating the client object. For example:

```
ClientConfig config = new ClientConfig("", "");
config.setReconnectOnConnectionLoss(true);
client = ClientFactory.createClient(config);
```

11.6. Support for custom load procedures in the kafkaloader

The kafkaloader is a tool for bulk loading data into a VolltDB database from a Kafka message queue. By default, kafkaloader uses a custom procedure to batch multiple rows into a single insert operation. You can now specify an alternate stored procedure to use for loading the data into the table. To specify an alternate stored procedure, use the `-p` or `--procedure` flag on the command line, just as you would for the csvloader.I'm

11.7. Improved performance deleting data with low-cardinality indexes

Low-cardinality indexes — that is, indexes with very few unique values — can be problematic because finding any single row requires scanning all the rows with the same index value. Deleting records from such a table amplifies the problem because the database must not only find and delete the data but also the index entry itself.

In previous versions of VoltDB, deleting large volumes of data from a table with a low-cardinality index could take a long time. This release adds a performance optimization to dramatically improve the latency of delete operations on low-cardinality indexes. This optimization will also benefit any index with a commonly recurring value, such as the null value.

11.8. New Java property to disable DR ports

A new Java property, VOLTDB_DISABLE_DR, has been added that allows you to disable the DR port listening ports when you start the VoltDB database process. This property is intended to help protect VoltDB against port scanners. See the implementation note concerning port hardening for details.

11.9. Fix for issues with aggregate functions of small VARCHAR columns

A error was introduced in version 4.5 where use of the MIN() and MAX() functions on small VARCHAR columns could produce incorrect answers. This issue only occurred for VARCHAR columns of 63 bytes or less — that is, VARCHAR(15) or VARCHAR(63 BYTES) or smaller.

This bug is now fixed. Because the issue introduced in v4.5 involves wrong answers, anyone using v4.5 is urged to upgrade to v4.6 at their earliest possible convenience.

11.10. Fix for unused command logging segment files being created

A timing issue related to command logging in earlier versions could result in unnecessary segment files being created. Under the worst conditions, unused segment files could continue to be created, filling up the `/command_log/segments` directory and using up available disk space.

This bug has now been fixed. Note that when the database restarts (for example, when you upgrade to 4.6) any extra segment files created by an earlier version as a result of this bug issue will automatically be deleted.

12. Release V4.5

12.1. Support for Java 8

VoltDB has been tested and validated on Java 8. The VoltDB server software now supports both the Java 7 and Java 8 JDKs. The VoltDB Java client library continues to support Java 6 as well as 7 and 8.

12.2. Support for exporting data to the RabbitMQ messaging service

An export connector for the RabbitMQ messaging service has been added to the VoltDB server software. See the chapter on "Exporting Live Data" in the *Using VoltDB* manual for details.

12.3. Kafka export connector improvements

The Kafka export connector has been extended and improved to allow more control over the data being exported, including support for VARBINARY columns and new properties to control the format of the data and the value used as the Kafka partitioning key. More information is available in the export chapter of the *Using VoltDB* manual.

12.4. Further improvements to temporary table usage

This release includes further performance improvements to the query planner, reducing temptable usage for multi-partition stored procedures. These improvements focus specifically on reducing temptable usage for queries involving GROUP BY clauses.

12.5. New FORMAT_CURRENCY() function

A new SQL function, FORMAT_CURRENCY(), converts a DECIMAL value to a text string. The function takes two arguments: the DECIMAL value and an integer specifying the decimal place to which the value is rounded. For example FORMAT_CURRENCY(1234.567, 2) = "1,234.57". See the *Using VoltDB* manual for details.

12.6. New graph available in Web Studio

The graphs in VoltDB Web Studio have been enhanced, improving the accuracy of the latency graph, fixing a bug where the partition starvation graph was not available on K-safe clusters, and adding a new graph: the transaction breakdown graph. The transaction breakdown graph is a pie chart showing the weighted average of execution time for stored procedures running on the database cluster. The new graph type is selectable from the pulldown menu above the graphs in the Web Studio interface.

12.7. Large number of joins no longer need explicit join order

Previously a query joining six or more tables required the user to explicitly list the join order as an additional argument to the Java SQLStmt declaration. VoltDB now assumes that joins of six or more tables, without an explicit join order, are joined in the order the tables are listed in the query. This change allows many large joins to compile and run, even as ad hoc queries. Note, however, that the order of the tables in the query may not be the optimal join order, so explicitly defining a join order is still recommended when performance is a consideration. See the *VoltDB Performance Guide* for more information on join order.

12.8. Additional string format as input to TIMESTAMP columns

When casting between TIMESTAMP and VARCHAR datatypes, VoltDB uses the string format YYYY-MM-DD hh.mm.ss.nnnnnn to represent the TIMESTAMP value. Previously, when entering a text string as input to a TIMESTAMP value, you had to provide the complete string, including both date and time (including fractional microseconds). You can now use just the date portion of the string as input, YYYY-MM-DD, and VoltDB assumes 00.00.00.000000 as the time portion.

12.9. Bug fixes

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

- In certain edge cases, subqueries involving partitioned tables with a LIMIT or DISTINCT clause within in a multi-partitioned query that also joined the results of the subquery to another partitioned table could result in poor planning and incorrect results. This issue is now fixed.
- Previously, when using csvloader in batch mode to load data into a master database where the table being loaded has especially large rows (e.g. lots of columns or "wide" columns), database replication (DR) could fail. This problem has been resolved.
- In previous releases, if multiple queries are queued using voltQueueSQL and more than one query fails, VoltDB would report the last failure rather than the first. This is now fixed and the voltExecuteSQL method reports the first failure.
- In previous releases, if the csvloader encounters too many errors (and the maximum error limit is set to a very large number), the load process could fail with an out of memory error. This limitation has been fixed. The csvloader writes errors and frees memory on an ongoing basis to avoid using up available memory.

13. Release V4.4.1

Important

V4.4.1 is primarily a bug fix release. Anyone using V4.4 — especially those using export functionality — are strongly recommended to upgrade.

13.1. Issue with export stopping after system failure and rejoin: fixed

It was possible for pending export data to get stuck in the overflow queue if nodes fail unexpectedly. The consequences of this bug were that the pending data was not written to the export target after the node rejoined and export would eventually stop altogether after a restart. This bug has been fixed.

13.2. Issue with "transactions moving backwards": fixed

There was a problem where certain operations, such as command log recovery, could result in an error warning of "transactions moving backwards". Most instances of this error have been eliminated. However, if you encounter this error after upgrading to V4.4.1, please contact support@voltdb.com.

13.3. Null pointer exception in export to file: fixed

A race condition in the export-to-file client could result in a null pointer exception (NPE), causing export to stop. This bug has been fixed.

13.4. Recovery failed after starting from an old catalog: fixed

VoltDB recently added the ability to start the database using a catalog compiled with an older version of VoltDB. In this situation, VoltDB recompiled the catalog before starting. However, the recompiled catalog was not correctly saved as part of the command logs. As a consequence, the resulting command logs could not be recovered when the database restarted. This bug has been fixed.

13.5. Kafka export did not handle the property skipinternals correctly: fixed

A new property added to the export-to-kafka connector in 4.4 could cause the database to fail with a Java out-of-memory error. This bug has been fixed.

14. Release V4.4

14.1. Import from Kafka

VoltDB now includes the **kafkaloader** utility, which is similar to **csvloader**, in that it imports from an Apache Kafka message queue and writes the records into the specified VoltDB database table. One difference with the **kafkaloader** is that, rather than the load process being a one-time event, **kafkaloader** continues to run, monitoring the queue for additional records and acting as a persistent import function. See the *Using VoltDB* manual for details.

14.2. Change to csvloader defaults

Previously, if CSV input contained missing fields, the default behavior for the **csvloader** was to generate an error. The default behavior has been changed so **csvloader** now treats missing values as null, to be more in keeping with other applications such as spreadsheets. You can revert to the previous behavior, if you wish, by including the flag `--blank=error` when running **csvloader**.

14.3. Ability to update the application catalog or deployment separately

The **voltadmin update** command originally required both a catalog and a deployment file. You can now specify either or both on the command line. If you specify just a catalog or just a deployment file, the other is assumed to remain the same. The file extensions .jar and .xml are used to identify which type of file is being updated. For example, all of the following are valid commands:

```
$ voltadmin update mycatalog.jar mydeployment.xml
$ voltadmin update mycatalog.jar
$ voltadmin update mydeployment.xml
```

14.4. New automatic stored procedure, upsert, for inserting or updating a record

When a schema is compiled into an application catalog, several simple stored procedures are created automatically for each table, including create for all tables and select, delete, and insert for tables with a primary key. A new automatic procedure, upsert, has been added for tables with primary keys. The upsert procedure inserts a new record if one does not already exist for the specified primary key, or updates the existing record if it does. See the *Using VoltDB* manual for details.

14.5. Improved handling of temporary memory (temp tables) during query processing

VoltDB has a limit on the amount of memory, known as temp tables, that can be used when processing SQL queries. It is possible to exceed this limit when performing very complex queries or queries with large intermediate result sets. In this release, the management of temp tables has been significantly improved, allowing more complex queries and combinations of queries to operate effectively within the limit. Users who have previously used the <temptables> setting in the deployment file to increase the temp table limit may find that this is no longer necessary,

14.6. Increased support for subqueries in ad hoc and multi-partition queries

Earlier releases limited use of subqueries in ad hoc and multi-partition transactions to replicated tables only. This limitation is now removed; ad hoc and multi-partition SELECT queries can include subqueries involving both replicated and partitioned tables.

14.7. Support for project.xml files will be removed in a future release

Prior to VoltDB V3.0, a project definition file was required to compile an application catalog. With V3.0, support for compiling schema files directly was added and use of the project definition file was deprecated and removed from the documentation. However, support for project definition files has remained in the product up to the current release. However, that support will be removed in an upcoming version. Anyone still using a project definition file should migrate to current usage to avoid problems in the future.

14.8. Bug fixes

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

- Previously, a failed rejoin could result in a spurious warning message "no stream snapshot ack message" appearing every 10 minutes. No action was required, but the error was misleading. This problem has been resolved.
- In earlier versions, sqlcmd and the web studio could not process SELECT statements containing subselects properly. The parsing of more complex statements in these utilities has been improved and this problem has been resolved.

15. Release V4.3

15.1. 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.

15.2. 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.

15.3. New sample application demonstrating time-based reporting and incremental deleting of old records

This release adds a new example application, windowing, to the VoltDB kit. This new application demonstrates how to perform two common tasks for real-time, big data applications:

- First, the application performs periodic reports aggregating data over a moving time "window". For the demonstration, the time windows are 1, 5, 10, and 30 seconds. But they could as easily be the last minute, the last hour, or the last day.
- Second, the application periodically deletes outdated records using the @RunEverywhere system procedure to keep only the pertinent records online.

The sample code demonstrates techniques for performing these common tasks using VoltDB's builtin functions and capabilities to achieve maximum throughput with minimal interruption to ongoing data ingestion.

15.4. 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).

15.5. 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.

15.6. 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*])

15.7. 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.

15.8. 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.

15.9. 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.

15.10. 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.

15.11. 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.

16. Release V4.2

16.1. 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.

16.2. 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.

16.3. 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.

16.4. 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.

16.5. 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.

16.6. 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.

16.7. 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.

16.8. 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.

16.9. 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.

16.10. 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.

16.11. 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.

16.12. 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.

16.13. 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.

17. Release V4.0.2.3

17.1. "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.

17.2. 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.

18. Release V4.0.2

18.1. 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.

18.2. 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.

18.3. 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.

18.4. 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.

18.5. 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.

18.6. 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.

18.7. 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..

19. Release V4.0.1

19.1. 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.

19.2. 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.

19.3. 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()

19.4. 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;
```

19.5. 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;
```

19.6. 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*.

19.7. 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.

20. Release V4.0

20.1. 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

20.2. 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. Synchronous export in Kafka can use up all available file descriptors and crash the database.

A bug in the Apache Kafka client can result in file descriptors being allocated but not released if the producer.type attribute is set to "sync" (which is the default). The consequence is that the system eventually runs out of file descriptors and the VoltDB server process will crash.

Until this bug is fixed, use of synchronous Kafka export is not recommended. The workaround is to set the Kafka producer.type attribute to "async" using the VoltDB export properties.

4. SQL and Stored Procedures

4.1. 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.

4.2. 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. 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.

6.2. 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}".

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.

2.2. Protecting VoltDB Against Port Scanners

VoltDB uses a number of different ports for interprocess communication as well as features such as HTTP access, DR, and so on. Port scanning software often interferes with normal operation of such ports by sending bogus data to them in an attempt to identify open ports.

Over the past few releases, VoltDB has hardened its port usage to ignore unexpected or irrelevant data from port scanners. However, the ports used for Database Replication (DR) cannot be protected in this way. So, in V4.6, a Java property was introduced to allow you to disable the DR ports, for situations where port scanning cannot be avoided. To disable the DR ports, set the Java property VOLTDB_DISABLE_DR to `true` before starting the database process. For example:

```
$ export VOLTDB_OPTS="-DVOLTDB_DISABLE_DR=true"
$ voltdb create myapplication.jar \
                --deployment=deployment.xml \
                --host=voltsvr1
```

Note that, if you disable the DR ports, you cannot use the database as a master for database replication.

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 voltify 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.