Release Notes

Product	VoltDB
Version	5.9.2
Release Date	March 20, 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

For customers upgrading from pre-5.0 releases of VoltDB, please see the *V4.0 Upgrade Notes* for special considerations when upgrading from previous major versions. Otherwise, the process for upgrading from a previous version of VoltDB is as follows:

1. Place the database in admin mode (using **voltadmin pause**).
2. Perform a manual snapshot of the database (using **voltadmin save**).
3. Shutdown the database (using **voltadmin shutdown**).
4. Upgrade the VoltDB software.
5. Restart the database (using the **voltdb create** action).
6. If upgrading to version 5.6 or earlier, reload any Java stored procedures and the database schema (using the sqlcmd directives **load classes** and **file**). *This step is not necessary for VoltDB 5.7 and later.*
7. Restore the snapshot created in Step #2 (using **voltadmin restore**).
8. Return the database to normal operations (using **voltadmin resume**).

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that might impact their existing applications.

1. Release V5.9.2 (March 20, 2016)

1.1. Recent improvements

The following limitations in previous versions have been resolved:

- Previously, using the JDBC method getFloat() to retrieve a negative value or zero (<=0) would result in the JDBC interface throwing an exception. This issue has been resolved.
- There was an issue with database replication (DR) where, if multiple transactions generated excessively large binary logs in a short period of time (more than 2 megabytes each in under a second) DR could fail, possibly taking the database cluster with it. The symptom when this occurred was that one or more nodes would fail with a DR buffer overflow error. This issue has now been corrected.

- Previously, if the export subsystem could not write data to the export_overflow directory (for example, if the disk was full), the VoltDB server process would generate errors in the log but not stop. However, this behavior results in lost data. So, to preserve data integrity and durability, the server process now fails in this situation. In a K-safe cluster, the other nodes will continue, keeping the database online, until operators can address the system issues with the failed node and rejoin it to the cluster.

- There was a condition where if, after using database replication (DR) and one of the clusters stops and recovers, the cluster could fail with a ConcurrentModificationException exception. This condition was caused by the partitions used for DR changing while the cluster was down and the partition mapping from one cluster to the other being out of sync. This issue has been resolved.

- Another rare condition related to database replication (DR) involved certain indexes with the columns in a particular order where, if one of the columns contained a null value and the record was updated or deleted, replication would stop. This issue has been resolved.

2. Release V5.9.1 (January 26, 2016)

2.1. Improved performance for high volume cross datacenter replication (XDCR) clusters

It was possible in an XDCR environment for the replay of binary logs from one cluster to interfere with the local transactions on the other cluster, resulting in high latency. The application of binary logs has been tuned to reduce the impact on the local client workload.

2.2. Fix for error when re-enabling database replication (DR) after cluster restart

There was a condition where if, after using database replication (DR) and one of the clusters stops and recovers, the cluster could fail with a ConcurrentModificationException exception. This condition was caused by the partitions used for DR changing while the cluster was down and the partition mapping from one cluster to the other being out of sync. This issue has been resolved.

2.3. Fix for edge case where database replication (DR) could deadlock when nodes fail or rejoin

In rare cases, a master database could report a null pointer exception and stall replication if the consumer cluster encountered a change in its topology (that is, a node failed or rejoined). A common symptom before the master cluster stalled was that it reported a series of informational messages that DR was "discarding ctrl message". This issue has been resolved.

3. Release V5.9 (January 15, 2016)

3.1. Ability to specify multiple servers when defining a DR connection

When configuring database replication (DR), you can now specify multiple servers, separated by commas, in the `source` attribute of the `<connection>` tag in the deployment file. This way, DR can start even if one of the listed servers on the master cluster is currently down. For example:

```
<dr id="2">
  <connection source="server1,server2" />
</dr>
```

3.2. New --wait flag for the voltadmin pause command

There is a new flag, **--wait**, available for the **voltadmin pause** command. With the **--wait** flag, the command waits until all pending transactions are processed and all database replication (DR) and export queues are flushed before returning control to the shell prompt. This can be useful when scripting the shutdown of the database, for example. Without the **--wait** flag, the **pause** command returns immediately, without waiting for the database to actually quiesce.

3.3. Ability to run VoltDB with Transparent Huge Pages (THP) for demos and testing

Normally, VoltDB refuses to start on Linux systems if Transparent Huge Pages (THP) are enabled, because THP are known to cause issues for memory-intensive applications like VoltDB. However, if you want to run simple tests using VoltDB without reconfiguring your Linux server, you can use the **--ignore=thp** flag when starting the server. For example:

```
$ voltdb create --ignore=thp
```

This flag is not supported and should *never* be used on production systems because of the problems THP can cause.

3.4. New method provides cluster ID to stored procedures

A new method in the VoltProcedure class returns the current cluster ID. The VoltProcedure.getClusterId() method returns the cluster ID specified when configuring database replication in the deployment file. The cluster ID is the unique ID listed in the `id` attribute of the `<dr>` tag. For example, the following deployment file sets the cluster ID to the value 2:

```
<dr id="2">
```

You can combine the return value of this new method with the .getUniqueId() method to generate an ID that is unique across multiple clusters, which is especially useful when using cross datacenter replication (XDCR).

3.5. Improved DR reliability against network issues

The resilience of database replication (DR) to continue in the face of network fluctuations has been improved. DR is less likely to break as a result of a network glitch than in previous releases.

3.6. New format attribute for built-in importers

There is a new attribute when configuring import in the deployment file. The `format` attribute lets you choose between comma-separated values (`csv`) and tab-separated (`tsv`) format for the imported data. For example, the following configuration specifies tab-delimited input:

```
<import>
  <configuration type="kafka" format="tsv" enabled="true">
      . . .
```

The default for the `format` attribute is `csv`.

3.7. Support for custom user-written importers

VoltDB now supports the use of custom importers through the use of the `type="custom"` and `module="{bundle}"` attributes. For example:

```
<import>
  <configuration module="mybundle.jar" type="custom" enabled="true">
    <property name="myprop1">val1</property>
    <property name="myprop2">val2</property>
  </configuration>
</import>
```

Contact support@voltdb.com for further Information on how to write custom importers.

3.8. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- 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.
- There was an issue in earlier releases where, if an export target is unavailable (for example, a remote JDBC server is down), calling the @Statistics system procedure with the TABLE selector caused the database server processes to hang. This issue has been resolved.

4. Release V5.8.1 (December 14, 2015)

4.1. Three new SQL functions: DATEADD(), PI(), and REGEXP_POSITION()

This release adds three new SQL functions:DATEADD(). PI(), and REGEXP_POSITION(). See the *Using VoltDB* appendix of SQL functions for details.

4.2. VoltDB now supports OS X version 10.8 and later

With the current release, VoltDB drops support for OS X release 10.7.

4.3. Additional improvements

In addition to the features described above, the following limitations in previous versions have been resolved:

- In previous releases, a query involving a LEFT JOIN with an OR predicate in the join, could produce an incorrect execution plan. This condition would result in either an error or wrong answer for the query. This issue has been resolved.
- Certain database workloads could produce unusually long Java garbage collection (GC) pauses when using recent versions of VoltDB. This issue has been corrected.
- Previously, attempting to disable database replication (DR) while the initial synchronization snapshot is in process could result in the database process stalling. This issue has been resolved.

5. Release V5.8 (November 16, 2015)

5.1. Cross Datacenter Replication (XDCR)

VoltDB is proud to announce the availability of Cross Datacenter Replication (XDCR), two-way active replication across two database clusters. XDCR provides the ability to maintain separate but synchronized writable databases in two separate locations. For more information, read the updated chapter on Database Replication in the *Using VoltDB* manual and contact your VoltDB sales representative. (XDCR is a separately licensed product feature.)

5.2. Statistics improvements

The information available from the @Statistics system procedure have been extended and improved. First, a new selector, IMPORTER, has been added which provides statistics about the performance of the import function.

Second, the selector DR, has been renamed DRPRODUCER to be more descriptive and complement the existing DRCONSUMER selector. Note that despite the name change, the old name for the selector, DR, will continue to be recognized for backwards compatibility.

Finally, the value of the status column SYNCSNAPSHOTSTATE returned by the DRPRODUCER selector has been changed. If no synchronization snapshot is occurring, the status column now returns the value "NONE".

5.3. New warnings for non-deterministic use of floating point values

Floating point datatypes approximate decimal values with a high but not perfect level of precision. As a result, arithmetic with floating point values, especially aggregation of multiple values is capable of generating slightly different values if aggregated in different orders. As a result, it is possible for use of aggregate functions on floating point column values to be non-deterministic.

VoltDB now generates non-determinism warnings if read/write stored procedures include queries that use the aggregate functions AVG() or SUM() on floating point columns. For precise decimal arithmetic, especially involving aggregate functions, use of the DECIMAL datatype is recommended.

5.4. String management optimizations

Short VARCHAR and VARBINARY strings are stored inline in the database record. Longer strings are managed in a separate pool using pointers. With this release the pointers used to manage longer string values have been optimized, reducing their size from 24 bytes per string to just 8 bytes. This reduces the overall memory required for databases with long string columns. Also, for performance reasons, the memory for string pointers is managed as a pool rather than being returned to the operating system with each DELETE operation. Consequently, optimizing the pointer size reduces the overall size of the pool retained by VoltDB when large numbers of string values are deleted (for example, after a TRUNCATE TABLE statement).

5.5. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- There was an issue in previous releases where, after creating a partial index involving a TIMESTAMP column, no further DDL statements were accepted by the database. This issue was specific to TIMESTAMP columns and partial indexes. That is, where the value of the TIMESTAMP column was used in a boolean expression, such as `myTimestamp < '2001-01-01'`. This problem has been resolved.
- VoltDB now gives a meaningful error message and stops if you specify a non-existent file as the deployment file argument on the **voltdb** command line (for example, if you misspell the file name).
- Previously, you could not use the **voltadmin stop** command to stop one node on a two-node cluster with K-safety=1 and network partition detection disabled. This issue has been corrected. Note, however, although this is now possible, the use of network partition detection is always recommended for K safe clusters.
- Previously, certain long-running queries could cause a node to exit the cluster with a fatal segmentation fault (SIGSEGV). This issue has been corrected.
- In earlier releases, if a replica database started but could not connect to the master database, it was possible for the replica server to eventually run out of system file descriptors, resulting in a fatal "too many open files" error. The problem has been corrected.

6. Release V5.7.1 (October 16, 2015)

The following issue has been corrected:

6.1. Issue with database replication (DR) and update operations involving null columns

An issue exists in the initial 5.7 release, where if database replication is enabled and an update operation includes a null column value, the database cluster receiving the DR transaction log could crash while attempting to apply the DR log. This issue has been resolved.

7. Release V5.7 (October 14, 2015)

7.1. Ability to include VARBINARY columns in indexes

It is now possible to include VARBINARY columns in indexes, including the primary key for the table.

7.2. Ability to snapshot selected tables

It is now possible to create a snapshot that contains only a subset of all of the tables in the database. This feature is useful for creating backups of selected tables or to reduce the size of snapshots for applications that contain large static tables and want to save only those tables that change frequently.

To create a snapshot with a subset of the database tables, you must use either the JSON-encoded arguments to the @SnapshotSave system procedure or the `--tables` and `--skiptables` flags to the **voltadmin save** shell command. In either case, you can specify either the tables to include (`tables`) or the tables to exclude (`skiptables`) from the snapshot as a comma-separated list. For example, both of the following commands create a snapshot containing the votes and contestants tables from the voter example application:

```
$ voltadmin save --tables=votes,contestants /tmp/ incpart

$ sqlcmd
1> exec @SnapshotSave '{uripath:"file:///tmp/", "nonce":"incpart",
2>     "tables":["votes","contestants"]}';
```

Whereas the following commands create a snapshot excluding just the votes table:

```
$ voltadmin save --skiptables=votes /tmp/ excpart

$ sqlcmd
1> exec @SnapshotSave '{uripath:"file:///tmp/", "nonce":"excpart",
2>     "skiptables":["votes"]}';
```

7.3. Manually restoring a snapshot now automatically loads the associated schema if no tables exist in the database

Previously, manually restoring a snapshot (with the **voltadmin restore** command or @SnapshotRestore system procedure) only restored the data, it was up to the user to reinstate the appropriate schema before performing the restore. Now, if there is no existing schema (that is, no tables are defined in the database), the restore operation will automatically restore the schema from the snapshot before restoring the data. This makes it possible to do a save and restore around a maintenance window (such as upgrading the VoltDB software) without having to manually reload the schema.

If tables do exist in the database, the old behavior of restoring just the data persists. This allows you to perform certain schema modifications that are not possible while the database is running (such as changing the partitioning column of a table with data in it) by saving a snapshot, restarting the database, loading the new schema, and restoring just the data.

7.4. Import statistics are now included in the @Statistics initiator data

The INITIATOR selector for the @Statistics system procedure returns information about stored procedure invocations. The INITIATOR statistics now include data from the stored procedure invocations associated with the import feature as well. In the return values for the INITIATOR selector, import activity is reported with the name of the importer in the CONNECTION_HOSTNAME column and the name of the import procedure in the PROCEDURE_NAME column.

7.5. Support for IS DISTINCT FROM and IS NOT DISTINCT FROM

The WHERE clause now supports the IS DISTINCT FROM and IS NOT DISTINCT FROM boolean operators. IS DISTINCT FROM and IS NOT DISTINCT FROM are similar to the equals ("=") and not equals ("<>") operators respectively, except when evaluating null operands. Equals and not equals return false if either operand is a null. IS DISTINCT FROM and IS NOT DISTINCT FROM treat nulls as valid values. See the description of the SELECT statement in the *Using VoltDB* manual for more information.

7.6. New close method for the PHP client library

Previously, there was no way to explicitly close a VoltDB connection within the PHP client library. A close method has been added. See the examples folder in the PHP client library kit, available from the VoltDB web site, for more information.

7.7. Ability to set a query timeout for each query

Administrators can set a system-wide query timeout for read-only queries in the deployment file when the database first starts. Any read-only queries that exceed the system-wide timeout limit are canceled.

It is now possible for users to set a timeout for individual read-only queries as well. There is a new method in the Java client API, callProcedureWithTimeout, that can be used in place of callProcedure to specify a query timeout for the call. You specify the timeout as an integer number of milliseconds. Interactive users can use the `--query-timeout` flag when starting the sqlcmd command line utility to specify a query timeout for all read-only queries in that sqlcmd session.

When security is enabled, only users with ADMIN privileges are allowed to set a per-transaction timeout that is longer than the system-wide timeout. However, all users can set a shorter query timeout.

7.8. Ongoing enhancements to the VoltDB Management Center

The web-based VoltDB Management Center (VMC) is enhanced at each release, adding new capabilities as features are added to the VoltDB product. For example, with this release the Admin tab shows the configuration for the DR overflow directory, which was missing previously.

We also try to make the management center as clear and usable as possible, so occasionally we rearrange features to maximize usability. For this release the default graphs and data tables shown on the Monitor tab have been changed. By default, only the Latency, TPS, Memory and CPU graphs are shown. You can customize this display by clicking on the "Show More/Fewer Charts" button and selecting the graphs that you want displayed. Your choices are saved and reused across sessions via browser cookies. We hope you find the new defaults useful and appreciate any feedback you may have regarding these changes. Feel free to comment in our forums. Thank you.

7.9. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- Previously, when starting database replication (DR) for the first time, the DR synchronization snapshot could block client applications, especially if the client workload contained large payloads. This issue has been resolved.

- When exporting to WebHDFS using the HTTP export connector, the %d placeholder is supposed to "roll" the export files to a new URL every hour. However, this was not happening properly in earlier releases. The %d placeholder now operates as described in the export documentation for WebHDFS.

- Previously, if a rejoin operation failed due to the difference in clock time between nodes exceeding the allowable limit (100 milliseconds), any subsequent attempt to rejoin a node or generate a snapshot would fail as well. This problem has been corrected.

- There was an edge case where an index could return the wrong answer if the query selection expression compared an indexed column using a less than (<) or less than or equal to (<=) operator, the results were ordered by that column and the column contained a null value in one or more rows. This issue has been resolved.

- Under certain conditions, the JSON interface connections could fail reporting the error "JSON interface exception: no connections" to the VoltDB Management Console (VMC). At which point the VMC would stop updating or processing SQL queries until the JSON interface was reset. This issue has been resolved and the VMC can recover automatically from failed connections.

8. Release V5.6 (September 11, 2015)

8.1. Ability to put the database into read-only mode when memory or disk space runs low

It is now possible to automatically pause the database when memory or disk space exceeds a defined limit. Using the `<resourcemonitor>`, `<memorylimit>`, and `<disklimit>` tags, you can specify either a fixed size (in gigabytes) or a percentage of the total resource as a limit for the process resident set size or total disk usage. For example:

```
<systemsettings>
  <resourcemonitor>
     <memorylimit size="10"/>
     <disklimit>
        <feature name="snapshots" size="75%"/>
     </disklimit>
  </resourcemonitor>
</systemsettings>
```

See the *VoltDB Administrator's Guide* chapter on monitoring for details.

8.2. Import Beta test completed, now production ready

Beta testing of the import feature in VoltDB has been completed. This feature is now supported for production use.

8.3. Support for multiple import streams

The built-in import feature now supports the ability to import multiple streams to different tables or stored procedures. By including multiple `<configuration>` sections in the `<import>` element of the deployment file, you can have different Kafka topics from a single set of brokers, or from different brokers, imported to separate tables. For example:

```
<import>
  <configuration type="kafka" enabled="true">
    <property name="brokers">kafkasvr:9092</property>
    <property name="topics">employees</property>
    <property name="procedure">EMPLOYEE.insert</property>
  </configuration>
  <configuration type="kafka" enabled="true">
    <property name="brokers">kafkasvr:9092</property>
    <property name="topics">managers</property>
    <property name="procedure">MANAGER.insert</property>
  </configuration>
</import>
```

Also note that the `type="kafka"` attribute is now supported as a production importer. See the import/export chapter of the *Using VoltDB* manual for details.

8.4. Kafka export improvements

The Kakfa export connector has been improved to avoid issues when using synchronized export. The acks.retry.timeout property avoids the export connector hanging when the Kafka infrastructure becomes unresponsive. You can specify a timeout period in milliseconds, after which the packet is assumed lost and is resent to the Kafka broker. See the import/export chapter of the *Using VoltDB* manual for details.

8.5. Ability to verify system requirements before running VoltDB

The **voltdb check** command lets you verify that your Linux server configuration meets VoltDB requirements before starting the database. The command checks such things as system settings (swappiness, transparent huge pages, etc), software versions, and whether necessary services (such as NTP) are running or not.

8.6. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- An issue existed in previous releases where a view containing a MIN or MAX aggregate function could, under specific circumstances, produce incorrect results. For the error to occur, the view must use a non-unique index including all columns in the view's GROUP BY clause and the MIN or MAX expression, and the table must have multiple records with the qualifying minimum or maximum value, of which only a subset were previously deleted. This issue has been corrected.

- Previously, using the INSERT INTO... FROM statement to export data from a regular table into an export table could generate an array out of bounds error during the operation if the export table contained a TIMESTAMP column. As a consequence, some rows from the INSERT INTO.. SELECT statement might not be exported. This problem has been fixed.

- Previously, if an index declaration included a function that expects a FLOAT or DECIMAL value, but the argument was an integer expression, the declarative statement would generate an error indicating that the integer value could not be converted (or "cast") to the appropriate destination type. This problem has been fixed.

- There was an issue (in versions 5.1 through 5.5.1) where if database replication (DR) was started with only some tables being replicated and a transaction on the master database modified a non-DR table, then rolled back, replication would be terminated and would have to be restarted. This problem has been fixed.

9. Release V5.5.1 (August 24, 2015)

9.1. Bug Fix

In previous releases, under certain circumstances, starting VoltDB on Macintosh OS X could result in an allocation (malloc) error and segmentation fault. This problem has been fixed.

10. Release V5.5 (August 11, 2015)

10.1. VoltDB now checks for Transparent Huge Pages before starting

Transparent Huge Pages (THP) are a new feature in recent Linux releases (Ubuntu starting with 14.04 and RHEL 6.x) that can cause issues for memory-intensive application like VoltDB. Consequently, VoltDB will now check for THP and refuse to start if they are enabled. See the *VoltDB Administrator's Guide* for instructions on disabling THP.

10.2. New import feature available for testing

VoltDB is adding a new feature that allows data to be imported automatically when the database starts, in much the same way the existing export feature lets you export data. For the initial beta release, a Kafka importer is provided. You can learn more in t. **Note:** this feature is a beta release and should not be used for production systems.

10.3. Updated Kafka export connector

The Kafka export connector has been updated to support the latest Kafka release, version 0.8.2. This version of Kafka significantly changes how Kafka producers work and so the properties for configuring Kafka export have changed as well. For example, the `producer.type` property is no longer valid. Customers currently using Kafka export may need to change their deployment files when upgrading. Please read the section on Kafka Export in the *Using VoltDB* manual for details.

10.4. New Elasticsearch export connector

A new export connector is available that exports data to Elasticsearch. Elasticsearch is a full-text search service that makes it possible to search VoltDB data more flexibly and faster than using VoltDB alone. See the section on Elasticsearch export in the *Using VoltDB* manual for details.

10.5. Rack-aware partitioning

K-safety increases the availability of a VoltDB database by duplicating partitions on different servers. By default, K-safety assumes all servers are distinct. However, in reality this is not always the case, for example in virtualized or hosted environments where multiple virtual servers could be running on the same hardware or on a rack sharing a power supply.

You can now provide hints to the cluster to improve the placement of duplicate partitions. The `--placement-group` flag to the **voltdb create** or **recover** commands lets you specify a group of names, separated by periods, that specify the location of the server process. For example:

```
$ voltdb create --placement-group=row5.rack3.server12
```

On startup, the cluster compares the placement strings of each server to optimize the distribution of the partitions. See the voltdb command reference page for details.

10.6. Wireshark plugin for VoltDB

A Wireshark plugin for analyzing the network connections between VoltDB servers and clients is now available from the VoltDB github repository. The plugin is provided “as is” as a courtesy to our users. See the included README for more information.

10.7. New logarithm function

The new LN() function returns the natural logarithm of a numeric value. You can also use LOG() as a synonym for LN().

10.8. New approximate distinct count function

A new function, APPROX_COUNT_DISTINCT(), returns an approximation of the number of distinct values for a specified column expression. APPROX_COUNT_DISTINCT(*expression*) can be significantly faster and use less temporary memory than the corresponding COUNT(DISTINCT *expression*) function, especially when aggregating values across partitions.

10.9. Ability to use csvloader to update records

The csvloader and kafkaloader utilities let you load data from external sources into a VoltDB database. By default, these utilities use INSERT statements to import the data, assuming there are no duplicates. However, you can now have these utilities use UPSERT to update existing records (or create new records where corresponding records do not exist) by including the `--update` flag. To use `--update` you must specify a table name rather than an input procedure and the table must have a primary key. See the csvloader and kafkaloader reference pages for details.

10.10. Ability to disable the quote character in csvloader

Another new flag for the csvloader utility is the `--noquotechar` tag. The `--quotechar` flag lets you specify the character for quoting values, which defaults to ASCII character 34, the quotation mark. The `--noquotechar` flag disables quoting and ensures the quotation mark is interpreted as a literal string character. Only the value separator and line delimiter then have special meaning in the input stream.

10.11. Simplified procedure for using DR

The configuration and startup of database replication (DR) have been simplified. The `listen="true"` attribute is no longer needed on the `<dr>` tag in the deployment file — listening is enabled by default. Also, rather than editing the deployment file and performing a **voltadmin update** to clear the DR buffers, there is a single command, **voltadmin dr reset**, that can break replication and clear any pending logs on the master database. Finally, it is now possible to change the replica database's `<connection>` source attribute after the database has started. These improvements are reflected in the instructions for using DR in the *Using VoltDB* manual.

10.12. Improvements to the JDBC interface

A number of changes and enhancements have been made to the JDBC interface to provide additional capabilities and better conformance with other JDBC implementations. Improvements include:

- Ability to pass Null values when using the Spring JDBC template
- Resolved issues when compiling JDBC client applications using Java 6
- Ability to use JDBC .getString() method on all VoltDB datatypes
- Improved compliance for standard JDBC getInt and getLong methods
- Better handling of multiple executeBatch statements and clearing of the batch queue

10.13. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- VoltDB client applications work with Java 6 through Java 8, but previously the open source server software could not be compiled with Java 8. This has been resolved and you can now build the VoltDB Community Edition using either Java 7 or Java 8.
- Due to recent changes, the VoltDB New Relic plugin was producing inaccurate latency information. This issue has been resolved and an updated plugin is available for New Relic users.
- An issue with the snapshotconverter utility in previous releases could result in rows being dropped from the output. This issue has been resolved. In addition, pre-defined CLI commands are now provided for both snapshot utilities. See the *VoltDB Administrator's Guide* for details.
- Previously the VoltDB Management Center SQL Query tab would not display null values properly for floating-point columns where Null is the default value. This problem has been fixed.

- There was an issue where, under certain conditions, attempting to recover a promoted replica from command logs would fail, reporting the fatal error "Haven't implemented more than one DC yet". This problem is now fixed.

11. Release V5.4.1 (July 28, 2015)

11.1. Bug Fix

There were edge cases where certain SQL queries with outer joins of three or more tables could return incorrect answers. This problem has been fixed.

12. Release V5.4 (July 10, 2015)

12.1. JDBC query timeout now settable in milliseconds

By default, query timeouts, as set by the JDBC setQueryTimeout function, are measured in seconds. However, it is now possible to define the timeout in more precise increments by changing the scale of the timeout function to milliseconds. You do this by adding the millisecond scale factor to the JDBC connection string. For example, the following code changes the timeout scale and then sets a timeout of 2500 millisecond (or 2.5 seconds):

```
Connection conn = DriverManager.getConnection(
    url + "?jdbc.querytimeout.unit=milliseconds");

Statement query = conn.createStatement();
query.setQueryTimeout(2500);
```

12.2. Decimal input with more than 12 decimal places is now rounded

VoltDB supports a DECIMAL datatype with a fixed scale of 12. Previously passing a decimal value with a scale greater than 12 to VoltDB resulted in a run-time error. This behavior has changed and VoltDB now rounds values with a scale greater than 12 to the closest 12 decimal places.

12.3. Ability to use Eclipse to debug VoltDB client applications and stored procedures

It is now possible to debug VoltDB client applications and stored procedures using Eclipse. Please see the VoltDB Github wiki for instructions.

12.4. GROUP BY clause no longer required for views

The requirement that a view definition contain a GROUP BY clause has been removed. It is now possible to create views containing aggregate functions, such as COUNT(*) , MAX(), MIN(), and SUM() without a GROUP BY clause. See the description of the CREATE VIEW statement in the *Using VoltDB* manual for details.

12.5. New MOD() function

This release adds the MOD() function, which performs a modulo operation on two integer values. The first argument is the dividend and the second is the divisor and the function returns the remainder of dividing the first by the second. For example MOD(12,5) returns the value 2. See the *Using VoltDB* manual for details.

12.6. Admin Mode now supports read-only access

The behavior of Admin Mode has been changed. Previously, a database in Admin Mode rejected all requests on the client port. Now Admin Mode operates as a read-only mode for the database. All write requests on the

client port are rejected, but read-only transactions (and read-only system procedures) are allowed. As before, both read-only and read/write transactions are allowed on the admin port.

12.7. New @Statistics selector provides data on command logging

The @Statistics system procedure supports a new selector, COMMANDLOG, which returns information about the status of command logging on the database cluster. See the discussion of @Statistcs in the *Using VoltDB* manual for details.

12.8. New features in the VoltDB Management Center

The web-based management tool, VoltDB Management Center, continues to be extended and enhanced. New features of the Management Center in this release include:

1. Ability to monitor command logging through a graph and data table in the Monitor tab

2. License information about the current cluster, available from the Help menu

3. The ability to see a list of affected tables where database replication (DR) and export are configured in the Admin tab

12.9. Performance improvements for database replication (DR)

Several optimizations have been applied to database replication (DR) to improve overall performance and reliability, including compression of the binary logs. Users of V5.3 are strongly encouraged to upgrade, as the current release corrects a known issue with V5.3 related to restarting DR after a system crash or maintenance window.

12.10. Improved subquery support

Subquery support has been refined to improve reliability in a number of edge cases. Upgrading is strongly recommended for customers using subqueries in their applications.

12.11. Change in the interpretation of DECIMAL and FLOAT constants

VoltDB now interprets numeric constants containing a decimal point but not in E notation, such as "456.123", as DECIMAL values. Previously VoltDB interpreted such constants as floating point values. This change is being made for consistency with the SQL standard and other SQL implementations. To specify a floating point constant, use E notation (e.g. "4.56123e2").

12.12. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- There was an issue in previous releases when converting TIMESTAMP values from microseconds to milliseconds (for example, using the JDBC ResultSet.getDate() or getTimestamp() functions). An incorrect answer was returned if the timestamp value was prior to the UNIX epoch and contained an even number of seconds. This issue has been resolved.

- In VoltDB, the COUNT(*) function is optimized for performance. In previous versions, this optimization did not apply to COUNT(1), which is another common programming pattern. COUNT(1) now uses the same optimization as COUNT(*).

- Materialized views now make better use of existing indexes to improve performance when updating view columns containing the MIN and MAX aggregate functions.

- Under certain conditions, export would fail when exporting a table with a long name and/or many columns, because the automatically constructed signature for the export data source was too long. The symptom was an error indicating export "could not find" the export data source. This issue has been corrected.
- There was an issue where starting the sqlcmd utility with the `--query` argument and redirecting output could result in the process hanging. This issue has been corrected.

13. Release V5.3 (May 26 2015)

13.1. Additional subquery support

Previously it was possible to use subqueries in the FROM clause of a SELECT statement. Now subqueries are supported in many more situations. Scalar subqueries, which return a single value, can be used in most cases where SQL functions can be used. Non-scalar subqueries, which return multiple rows or multiple columns per row, can be used in array comparisons and in the IN and EXISTS predicates.

Subqueries are valid only in the SELECT statement. They cannot be used in other SQL statements or index definitions. Also, for the initial release of extended subquery support, only replicated tables can be used in subqueries outside of the FROM clause.

13.2. VoltDB Management Center improvements

The VoltDB Management Center contains a number of improvements and new capabilities, including:

- **DR support** — The Management Center provides new support for database replication (DR) in both the Admin and Monitor tabs. In the Admin tab you can view and modify the DR configuration of both master and replica databases. In the Monitor tab you can see performance statistics for DR, including a graph showing the replication rate on the replica and data tables on both the master and replica reporting performance and status information concerning DR.
- **Export configuration support** — The Management Center now supports editing the configuration of export streams in the Admin tab.
- **SHA-256 support** — The Management Center now uses SHA-256 hashing of passwords when authenticating to servers with security enabled.

13.3. 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.

13.4. SHA-256 support in python client

The python client library has been updated to support SHA-256 hashing of passwords when authenticating to servers with security enabled.

13.5. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- Starting in VoltDB 5.2, the JSON interface supported both SHA-1 and SHA-2 hashing of passwords. However, the same user could not use both SHA-1 and SHA-2 from different connections at the same time. This could happen accidentally, for example, when using direct JSON calls while also using the VoltDB Management Center, which uses the JSON interface to communicate with the server,. This issue has been corrected.

- Previously, setting the JDBC default connection timeout to zero was supposed to disable the timeout. However, it actually set the timeout to the default of two minutes. Setting the JDBC connection timeout to zero now works properly, setting an infinite timeout.
- Including the characters "hash" in an index name produces a hash index instead of the default tree index. Previous versions also inadvertently created hash indexes if a column identified as a primary key contained "hash" in the column name. This problem has been resolved.
- Previously, a race condition could result in the VoltDB Management Center returning a "no connections" error if it attempted to access the server during or shortly after a change to the database schema. This problem has been resolved.

14. Release V5.2.2 (May 15 2015)

14.1. Excessive memory use when overflowing queued export or DR data corrected

There was in issue in earlier releases where, if the target of export or database replication (DR) stalled, the sending cluster buffers queued data to disk. However, this did not properly free the associated memory; and so memory usage would increase. It was possible, if the service buffered data to disk for an extended period of time, that the server process could run out of memory.

This issue has been resolved and memory associated with data buffered to disk is released appropriately. Note however, that even if excessive memory usage is no longer a problem, you should always try to resolve issues with stalled downstream systems when using export or DR because buffered data could eventually exceed disk storage capacity.

14.2. New C++ client supports SHA-256 hashing

The C++ client library has been updated to support SHA-256 hashing of passwords when authenticating to servers with security enabled. By default, the client supports past and present server versions by using SHA-1 hashing. However, when connecting to VoltDB 5.2 and later servers, you can use SHA-256 hashing by specifying the hash type in the client configuration. For example:

```
voltdb::ClientConfig config("myusername",
                            "mypassword",
                            voltdb::HASH_SHA256);
voltdb::Client client = voltdb::Client::create(config);
```

Now both the Java and C++ client libraries support SHA-256 hashing. The new C++ client is available from the VoltDB client downloads page.

15. Release V5.2.1 (April 30, 2015)

15.1. Support for SHA-2 in the **voltdb mask** command

VoltDB 5.2 introduced use of SHA-2 hashing for authentication. This release brings the **voltdb mask** function up to date with the new authentication scheme. For customers using the mask function, be sure to re-hash your deployment file using the 5.2.1 **voltdb mask** command and use the newly hashed deployment file when starting the database to ensure all command utilities can authenticate properly.

16. Release V5.2 (April 29, 2015)

16.1. Ability for database replication (DR) to resume across cluster outages

Previously, database replication (DR) was able to continue despite individual node failures (in a K-safe environment). However failure of either the master or replica cluster would force a restart of DR. Beginning with

5.2, DR can resume across cluster failures when either the master or replica is recovered from command logs. See the chapter on "Database Replication" in the *Using VoltDB* manual for details.

16.2. Secure export to Hadoop using Kerberos

The HTTP export connector now supports the use of Kerberos authentication when exporting to a WebHDFS endpoint that is configured to use Kerberos. See the section on using the HTTP export connector in the *Using VoltDB* manual for details.

16.3. Support for partial indexes

VoltDB now supports partial indexes. That is, the index definition can contain a WHERE clause limiting the rows that are included in the index. For example:

```
CREATE INDEX completed_tasks
    ON tasks (task_id, startdate, enddate)
    WHERE enddate IS NOT NULL;
```

For the initial release of partial indexes, there are certain limitations on when and where such indexes and the tables associated with them can be modified. For now, you cannot use the ALTER TABLE statement to modify a table with a partial index. This limitation is expected to be relaxed in a future release.

16.4. New VoltDB Management Center features

The Management Center, VoltDB's web-based management console, continues to be extended and improved. This release contains two major new features:

- **New Idle Time graph** — The Monitor tab contains a new graph, the partition idle time graph, which shows the amount of work being done by each partition on the current server. The graph plots the percentage of time each partition is idle. 100% indicates the partition is doing no work (processing no stored procedures or ad hoc queries); 0% indicates the partition is constantly in use. The graph also includes lines for the local multi-partition coordinator and the minimum and maximum idle time for the cluster as a whole.
- **Ability to change configuration settings in the Admin tab** — The Admin tab now lets you change deployment settings that are configurable at runtime, including export properties, automated snapshots, security, and selected system settings. Click on the pencil icon next to a property to edit it. Note that if security is enabled, only users with the ADMIN permission are allowed to view and edit the Admin tab settings.

16.5. New bitwise functions

VoltDB now supports several new functions for performing bitwise operations on BIGINT values. The new functions support standard binary operands such as AND, OR, XOR, and NOT as well as bit shifting operations. See the reference pages for the BITAND(), BITNOT(), BITOR(), BITXOR(), BIT_SHIFT_LEFT(), and BIT_SHIFT_RIGHT() functions in the *Using VoltDB* manual for details.

16.6. New HEX() function

Another new function, HEX(), converts a BIGINT value into its hexadecimal representation as a string. See the reference page for the HEX() function in the *Using VoltDB* manual for details.

16.7. Support for SHA-2

VoltDB now supports SHA-2 hashing of credentials between the Java and c++ client libraries and the server. When you pass a username and password, the updated client library uses a SHA-2 hash of the credentials. On the server side, the VoltDB server accepts both SHA-1 (sent by previous versions of the client) and SHA-2. So both current and previous versions of the client libraries continue to work with the latest server release.

16.8. New voltadmin command to stop individual servers

The VoltDB command line utility, **voltadmin**, now supports the **stop** command. The **voltadmin stop** command stops the VoltDB server process on the specified node. Note that the stop command can only be used on a K-safe cluster and will not intentionally shutdown the database. That is, the command will only stop a node if there are enough nodes left for the cluster to remain viable.

16.9. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- In earlier releases, the Java client library interpreted the default call timeout value incorrectly, resulting in a much longer timeout period than the expected two minutes. This problem has been corrected. It should be noted, however, that long-running procedures that completed in previous versions, may have the client timeout the call now that the default timeout period is being properly enforced. (The procedure itself may or may not complete, but the client application will not know since it no longer listens for a response after the timeout.)

- Previously, database replication (DR) ignored the setting for the external interface on the command line (that is, the `--externalinterface` flag). This issue has been corrected and now the interface used for the replication port comes from, in order of priority, A.) the interface specified using the `--replication` command line flag, B.) the interface specified using the `--externalinterface` command line flag, or if neither of the preceding are specified, C.) all available interfaces.

17. Release V5.1.2 (April 16, 2015)

VoltDB 5.1.2 is a patch release that provides performance and stability improvements for Database Replication (DR).

17.1. Improved performance for initial DR snapshot.

When database replication starts, a snapshot is sent from the master database to the replica. In this release several I/O improvements have been made to improve the performance and reliability of the initial DR snapshot.

17.2. Improved management of DR buffers

There was an issue with how multiple buffers were grouped and managed in DR, which could result in decreased replication throughput. This issue has been resolved.

18. Release V5.1.1 (April 12, 2015)

VoltDB 5.1.1 is a patch release that fixes an issue introduced in 5.1.

18.1. Bug fix: Excessive CPU usage on idle database

Changes in VoltDB 5.1 introduced a process that, when the database was idle, would "spin" making it appear that the database was consuming significant CPU cycles. When the database was active processing queries, the CPU usage would drop to normal levels.

Although not dangerous, this behavior was misleading and is corrected by the current update.

19. Release V5.1 (March 22, 2015)

VoltDB 5.1 introduces several significant new features and enhancements. Existing customers should pay close attention to the following notes to see what if any changes they may want to make to their applications and/or operations to take advantage of the new capabilities.

19.1. New implementation of Database Replication (DR)

Database Replication (DR) lets you automatically copy updates to database tables from one database (the master) to another (the replica). Starting with VoltDB 5.1, DR has been rewritten to remove any single point of failure, improve performance, and allow new capabilities in the future. New features include:

- Significantly better performance — rather than a single replication stream, DR now occurs between multiple partitions simultaneously. Also, the new DR uses binary logs of transaction results saving the replica from having to replay the transaction.

- No single point of failure — By eliminating the DR agent the new DR not only removes a single point of failure, it simplifies DR from an operational perspective.

- More flexibility — You can now specify which tables you want to replicate rather than having to replicate the entire database. Of course, you can always choose to replicate all of the tables if you like.

For existing DR customers, the new capabilities and the elimination of the DR agent do necessitate some operational changes. Specifically, you must now:

- Identify the tables participating in DR using the DR TABLE statement in the schema.

- Configure DR in the deployment files for both the master and the replica clusters.

See the chapter on Database Replication in the Using VoltDB manual for details.

19.2. Ability to export data to multiple streams

VoltDB now allows you to export data to more than one target at a time. By assigning export tables to individual streams and then configuring each stream separately in the deployment file, you can export data to multiple targets simultaneously. For example, you might export deduped sensor data to Hadoop once it has been processed and export alerts regarding unusual events to HTTP for distribution via SMS, email, or other notification service. See the chapter on exporting live data in the Using VoltDB manual for details.

Use of multiple streams does require additional information in the schema and the deployment file. For example, the EXPORT TABLE statement now requires a TO STREAM clause so you can specify the stream to which each export table is directed. However, for backwards compatibility, the old syntax is still supported temporarily to allow customers time to migrate existing applications at their convenience.

19.3. Batch processing of interactive DDL statements

VoltDB 5.0 introduced interactive DDL, eliminating the need for a precompiled application catalog. However, large schema could take a significant time to process interactively. VoltDB 5.1 solves this problem by allowing you to batch DDL statements. If you have your DDL statements in a single file, you can use the `file --batch` directive in sqlcmd to batch process the DDL statements. For example:

```
$ sqlcmd
1> file --batch myschema.sql;
```

If you have a mix of DDL (data definition language) statements and DML (data manipulation language) and directives you can batch process only the DDL statements by enclosing them in a `file --inlinebatch` directive and the specified end marker. For example:

```
load classes myprocs.jar;
file --inlinebatch END_OF_BATCH
CREATE PROCEDURE FROM CLASS procs.AddEmp;
CREATE PROCEDURE FROM CLASS procs.ChangeDept;
PARTITION PROCEDURE AddEmp ON TABLE emp COLUMN empid;
PARTITION PROCEDURE ChangeDept ON TABLE emp COLUMN empid;
```

END_OF_BATCH

Batch processing DDL statements can speed up the processing of those statements by a factor of 10 or more, depending on the number and complexity of the statements and the size of the cluster.

19.4. New Administrative features in VoltDB Management Center

VoltDB Management Center, the web-based console for managing and monitoring a VoltDB database, now has a tab for administrative functions. On the **Admin** tab, you can pause and resume the database, save and restore snapshots, as well as review and update the database configuration. If security is enabled for the database, only users with the ADMIN permission can see and use the **Admin** tab in the Management Center.

19.5. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- Previously, if a table containing data was changed to an export table, the data was lost. VoltDB no longer allows you to issue an EXPORT TABLE statement on a table with existing data.
- In release 5.0, reconfiguring export on a running database while dropping and adding export tables could cause unpredictable export results. This issue is resolved. Export tables can be modified and export streams can be reconfigured on a running database with predictable results.
- There was an issue with the sqlcmd command line utility where it could occasionally hang during startup and never reach the input prompt. This problem has been resolved.
- The HTTP port and JSON interface are now enabled by default. Previously, this port was enabled in the default deployment file, but not if you used a custom deployment file. If you want to disable the HTTP port or JSON capability, you must explicitly disable them in the deployment file. For example:

```
<httpd enabled="false">
    <jsonapi enabled="false"/>
</httpd>
```

- There was an issue where command logs could not be recovered if the logs included a catalog update using a catalog from a previous VoltDB version. This problem is now fixed. However, recompiling your existing catalogs is recommended whenever you update your VoltDB version.
- Previously, there was an issue with @UpdateLogging, where spurious startup messages could flood the logs beginning the day after invoking the system procedure to update the Log4J configuration. This problem is now fixed.
- There was an planning error where a UNION statement within a subquery could result in a null pointer exception when the statement was compiled. This problem is now fixed.
- Another planning error resulted when compiling a SQL statement, such as SELECT, where the selection expression listed the same column twice. This problem is now fixed.

20. Release V5.0.2 (February 16, 2015)

This release contains no new features but corrects the following issues from the original 5.0 release.

20.1. 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.

20.2. 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.

20.3. Lines starting with "file" in sqlcmd incorrectly interpreted as a file directive.

In the original 5.0 release, any sqlcmd input line beginning with "file" (regardless of upper or lowercase) was interpreted as a **file** directive, even in the middle of a multi-line statement. This would happen, for example, if a CREATE TABLE statement included a column name starting with "file":

```
CREATE TABLE archive (
   ID INTEGER,
   Directory VARCHAR(128),
   Filename VARCHAR(128)
);
```

This usually resulted in several errors and the intended statement not being interpreted correctly. This issue is now fixed.

21. Release V5.0 (January 28, 2015)

21.1. Interactive DDL

The major new feature in VoltDB 5.0 is the ability to enter data definition language (DDL) statements interactively. For example, using sqlcmd on the command line or the VoltDB Management Center SQL Query interface. This makes the process of creating a database and defining the schema more flexible. As part of the support for interactive DDL, the following features have been added:

- Support for the DROP and ALTER statements for removing and modifying existing schema objects
- The ability to combine the CREATE PROCEDURE and PARTITION PROCEDURE statements into a single CREATE PROCEDURE statement with a PARTITION ON clause
- A new system procedure, @UpdateClasses, for adding and removing classes
- Two corresponding sqlcmd directives, **load classes** and **remove classes**, perform this function from the command line

Pleased note that processing DDL interactively can take longer than compiling an application catalog all at once. This is most noticeable when processing a large schema and especially on a multi-node cluster (where each change must be coordinated among the servers).

If you find entering DDL interactively too slow, it is possible to revert to precompiling the schema before starting the database. You have two choices:

- You can return to using catalogs exclusively, by setting the `schema="catalog"` attribute in the deployment file.
- You can compile the initial schema as a catalog, start the database specifying the catalog on the **voltdb create** command, but leave the deployment file unchanged. In this case, the database starts from the catalog, but you can use interactive DDL to modify the schema and stored procedures once the database is running.

Performance improvements for processing large schemas interactively are expected in upcoming releases.

21.2. Ability to "trim" rows using LIMIT PARTITION ROWS EXECUTE

The LIMIT PARTITION ROWS constraint now supports an EXECUTE clause that lets you specify a DELETE statement that is executed when the constraint value is exceeded. The EXECUTE clause gives you the ability to automatically "prune" older data when the constraint is reached. See the description of the CREATE TABLE statement in the *Using VoltDB* manual for details.

21.3. Support for HttpFS targets in Hadoop export

The HTTP connector, now supports Apache HttpFS (Hadoop HDFS over HTTP) servers as a target when exporting using the WebHDFS protocol. Set the export property `httpfs.enable` to "true" when exporting to HttpFS servers.

21.4. Addition of the ORDER BY clause to the DELETE statement

It is now possible to use the ORDER BY clause with LIMIT and/or OFFSET when performing a DELETE operation. ORDER BY allows you to more selectively remove database rows. For example, the following DELETE query removes the five oldest records, based on a timestamp column:

```
DELETE FROM events ORDER BY event_time ASC LIMIT 5;
```

Note that DELETE queries that include the ORDER BY clause must be single-partitioned and the ORDER BY clause must be deterministic. See the description of the DELETE statement in the *Using VoltDB* manual for details.

21.5. Bug fixes

In addition to the new features listed above, VoltDB V5.0 includes fixes to several known issues:

- Previously, there was an undocumented limit of 200 kilobytes to the size of the parameter list on the JSON interface. This limit has been extended to 2 megabytes.

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. Changing the deployment configuration when recovering command logs, can result in unexpected settings.

There is an issue where, if the command log contains schema changes (performed through interactive DDL statements, **voltadmin update**, or @UpdateApplicationCatalog), when the command logs are recovered, the

previous deployment file settings are used, even if an alternate deployment file is specified on the **voltdb recover** command line. Then, after recovering the database, a new schema update can result in the deployment settings specified on the command line taking affect.

Until this issue is resolved, the safest workaround to ensure the desired configuration is achieved is to perform the **voltdb recover** operation without modifying the current deployment file, then make deployment changes with the **voltadmin update** command after the database has started.

1.2. 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.)

1.3. 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. Some DR data may not be delivered if master database nodes fail and rejoin in rapid succession.

Because DR data is buffered on the master database and then delivered asynchronously to the replica, there is always the danger that data does not reach the replica if a master node stops. This situation is mitigated in a K-safe environment by all copies of a partition buffering on the master cluster. Then if a sending node goes down, another node on the master database can take over sending logs to the replica. However, if multiple nodes go down and rejoin in rapid succession, it is possible that some buffered DR data — from transactions when one or more nodes were down — could be lost when another node with the last copy of that buffer also goes down.

If this occurs and the replica recognizes that some binary logs are missing, DR stops and must be restarted.

To avoid this situation, especially when cycling through nodes for maintenance purposes, the key is to ensure that all buffered DR data is transmitted before stopping the next node in the cycle. You can do this using the @Statistics system procedure to make sure the last ACKed timestamp (using @Statititcs DR on the master cluster) is later than the timestamp when the previous node completed its rejoin operation.

3. Cross Datacenter Replication (XDCR)

3.1. Avoid replicating tables without a unique index.

Part of the replication process for XDCR is to verify that the record's starting and ending states match on both clusters, otherwise known as *conflict resolution*. To do that, XDCR must find the record first. Finding uniquely indexed records is efficient; finding non-unique records is not and can impact overall database performance.

To make you aware of possible performance impact, VoltDB issues a warning if you declare a table as a DR table and it does not have a unique index.

3.2. When starting XDCR for the first time, only one database can contain data.

You cannot start XDCR if both databases already have data in the DR tables. Only one of the two participating databases can have preexisting data when DR starts for the first time.

3.3. During the initial synchronization of existing data, the receiving database is paused.

When starting XDCR for the first time, where one database already contains data, a snapshot of that data is sent to the other database. While receiving and processing that snapshot, the receiving database is paused. That is, it is in read-only mode. Once the snapshot is completed and the two database are synchronized, the receiving database is automatically unpaused, resuming normal read/write operations.

3.4. A large number of multi-partition write transactions may interfere with the ability to restart XDCR after a cluster stops and recovers.

Normally, XDCR will automatically restart where it left off after one of the clusters stops and recovers from its command logs (using the **voltdb recover** command). However, if the workload is predominantly multi-partition write transactions, a failed cluster may not be able to restart XDCR after it recovers. In this case, XDCR must be restarted from scratch, using the content from one of the clusters as the source for synchronizing and recreating the other cluster (using the **voltdb create** command) without any content in the DR tables.

3.5. A TRUNCATE TABLE transaction will be reported as a conflict with any other write operation to the same table.

When using XDCR, if the binary log from one cluster includes a TRUNCATE TABLE statement and the other cluster performs any write operation to the same table before the binary log is processed, the TRUNCATE TABLE operation will be reported as a conflict. Note that currently DELETE operations always supercede other actions, so the TRUNCATE TABLE will be executed on both clusters.

3.6. Exceeding a LIMIT PARTITION ROWS constraint can generate multiple conflicts

It is possible to place a limit on the number of rows that any partition can hold for a specific table using the LIMIT PARTITION ROWS clause of the CREATE TABLE statement. When close to the limit, transactions on either or both clusters can exceed the limit simultaneously, resulting in a potentially large number of delete operations that then generate conflicts when the the associated binary log reaches the other cluster.

3.7. Use of the VoltProcedure.getUniqueId method is unique to a cluster, not across clusters.

VoltDB provides a way to generate a deterministically unique ID within a stored procedure using the getUniqueId method. This method guarantees uniqueness *within the current cluster*. However, the method could generate the same ID on two distinct database clusters. Consequently, when using XDCR, you should combine the return values of VoltProcedure.getUniqueId with VoltProcedure.getClusterId, which returns the current cluster's unique DR ID, to generate IDs that are unique across all clusters in your environment.

4. Export

4.1. 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.

5. Import

5.1. Data may be lost if a Kafka broker stops during import.

If, while Kafka import is enabled, the Kafka broker that VoltDB is connected to stops (for example, if the server crashes or is taken down for maintenance), some messages may be lost between Kafka and VoltDB. To ensure no data is lost, we recommend you disable VoltDB import before taking down the associated Kafka broker. You can then re-enable import after the Kafka broker comes back online.

6. SQL and Stored Procedures

6.1. Comments containing unmatched single quotes in multi-line statements can produce unexpected results.

When entering a multi-line statement at the sqlcmd prompt, if a line ends in a comment (indicated by two hyphens) and the comment contains an unmatched single quote character, the following lines of input are not interpreted correctly. Specifically, the comment is incorrectly interpreted as continuing until the next single quote character or a closing semi-colon is read. This is most likely to happen when reading in a schema file containing comments. This issue is specific to the sqlcmd utility.

A fix for this condition is planned for an upcoming point release

6.2. 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.

6.3. 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.

7. Client Interfaces

7.1. 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.

8. Enterprise Manager

Important

The VoltDB Enterprise Manager is deprecated. It is supported for existing customers but is not recommended for new deployments and will be removed in a future release. The VoltDB Management Center, which has improved and extended management and monitoring capabilities built directly into the VoltDB database server, is the recommended replacement for the Enterprise Manager. See the *Administrator's Guide* for more information on the VoltDB Management Center.

8.1. 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.

8.2. 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*
```

8.3. 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
```

8.4. 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.

8.5. The Enterprise Manager cannot start or manage a replica database for database replication.

Starting with VoltDB 5.1, database replication (DR) has changed and the VoltDB Enterprise Manager can no longer correctly configure, start or manage a replica database. The recommended method is to start the database manually and use the builtin VoltDB Management Center to manage the database by connecting to the cluster nodes directly on the HTTP port (8080 by default).

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. VoltDB Management Center

1.1. Schema updates clear the stored procedure data table in the Management Center Monitor section

Any time the database schema or stored procedures are changed, the data table showing stored procedure statistics at the bottom of the Monitor section of the VOltDB Management Center get reset. As soon as new

invocations of the stored procedures occur, the statistics table will show new values based on performance after the schema update. Until invocations occur, the procedure table is blank.

2. SQL

2.1. You cannot partition a table on a column defined as ASSUMEUNIQUE.

The ASSUMEUNIQUE attribute is designed for identifying columns in partitioned tables where the column values are known to be unique but the table is not partitioned on that column, so VoltDB cannot verify complete uniqueness across the database. Using interactive DDL, you can create a table with a column marked as ASSUMEUNIQUE, but if you try to partition the table on the ASSUMEUNIQUE column, you receive an error. The solution is to drop and add the column using the UNIQUE attribute instead of ASSUMEUNIQUE.

2.2. Adding or dropping column constraints (UNIQUE or ASSUMEUNIQUE) is not supported by the ALTER TABLE ALTER COLUMN statement.

You cannot add or remove a column constraint such as UNIQUE or ASSUMEUNIQUE using the ALTER TABLE ALTER COLUMN statement. Instead to add or remove such constraints, you must first drop then add the modified column. For example:

```
ALTER TABLE employee DROP COLUMN empID;
ALTER TABLE employee ADD COLUMN empID INTEGER UNIQUE;
```

2.3. 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.

2.4. 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.

3. Runtime

3.1. 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.

3.2. 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.

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.