Release Notes

Product	VoltDB
Version	7.5
Release Date	July 28, 2017

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

The process for upgrading from the recent versions of VoltDB is as follows:

1. Shutdown the database, creating a final snapshot (using **voltadmin shutdown --save**).
2. Upgrade the VoltDB software.
3. Restart the database (using **voltdb start**).

For DR clusters, see the section on "Upgrading VoltDB Software" in the *VoltDB Administrator's Guide* for more special considerations related to DR upgrades.

Support for upgrading using **shutdown --save** was only added in V6.8. If you are upgrading from older versions of VoltDB, you will need to save and restore the snapshot manually. The procedure to do that is as follows:

1. Place the database in admin mode (using **voltadmin pause**).
2. Perform a manual snapshot of the database (using **voltadmin save --blocking**).
3. Shutdown the database (using **voltadmin shutdown**).
4. Upgrade the VoltDB software.
5. Initialize a new database root directory (using the **voltdb init --force** action).
6. Start the database in admin mode (using the **voltdb start --pause** action).
7. Restore the snapshot created in Step #2 (using **voltadmin restore**).
8. Return the database to normal operations (using **voltadmin resume**).

For customers upgrading from V6.x or earlier releases of VoltDB, please see the *V6.0 Upgrade Notes*.

For customers upgrading from V5.x or earlier releases of VoltDB, please see the *V5.0 Upgrade Notes*.

For customers upgrading from V4.x or earlier releases of VoltDB, please see the *V4.0 Upgrade Notes*.

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 V7.5 (July 28, 2017)

1.1. Support for Java boxed datatypes

Java boxed datatypes such as Short, Integer, and Long are now supported as input to stored procedures. Previously, only primitive datatypes were supported. Using boxed types, you can pass Java null values directly to VoltDB and have them interpreted correctly.

1.2. New functions to convert between radians and degrees

Two new SQL functions have been added, RADIANS() and DEGREES(), that let you convert floating-point values between radians and degrees.

1.3. Improvements to offset tracking for the kafkaloader utility and Kafka import

The kafkaloader utility and built-in Kafka importer have been modified to improve the tracking of the current offset between Kafka and VoltDB. By default, these features now use a manual commit policy to ensure the current offset in the Kafka message queue matches the records successfully inserted into the VoltDB database, rather than tracking just what messages have been successfully read from Kafka. You can, if you wish, revert to an automatic commit policy by setting the appropriate property in the command line argument or import configuration.

1.4. Support for SSL and Kafka 0.10.2

Kafka 0.10.2 supports secure socket layer (SSL) encryption. Using the 0.10.2 version of the kafkaloader utility (kafkaloader10), you can connect to SSL-secured Kafka brokers by putting the appropriate SSL properties in the Kafka configuration file specified in the `--config` argument. For example, using a local trust and key store, the properties file might look like this following. (This only works with the kafkaloader10 utility, since Kafka version 0.8.2 did not support SSL.)

```
security.protocol=SSL
ssl.truststore.location=/Home/jdoe/mytruststore.jks
ssl.truststore.password=mytrustpasswd
ssl.keystore.location=/Home/jdoe/mykeystore.jks
ssl.keystore.password=mykeypasswd
ssl.key.password=passwd
```

1.5. Ability to write Kafka messages to two or more tables

The Kafka import connector is now capable of writing messages from a single topic to multiple tables, without requiring a custom insert procedure. If you create two configurations with the same Kafka topic ID but different group IDs, you can have a single topic written to two different tables. For example, the following configuration writes records from the Kafka *employees* topic to both the *EMP* and *MGR* tables.

```
<import>
   <configuration type="kafka" enabled="true">
       <property name="topics">employees</property>
       <property name="groupid">voltdbemp</property>
       <property name="procedure">EMP.insert</property>
```

```
    </configuration>
    <configuration type="kafka" enabled="true">
        <property name="topics">employees</property>
        <property name="groupid">voltdbmgr</property>
        <property name="table">MGR.insert</property>
    </configuration>
</import>
```

1.6. New @Statistics selector QUEUE

A new selector has been added to the @Statistics system procedure. The QUEUE selector returns statistics on the number of tasks in each partition's process queue and the average and maximum time tasks were waiting in the queue. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

1.7. Improvements to INSERT INTO... SELECT to reduce temporary memory usage

Performing an INSERT INTO... SELECT can produce many records, depending on the results of the SELECT. In the past, it was easy to use up all of the available temporary memory that VoltDB allocates for transactions, resulting in the statement failing and rolling back. The execution plans for these statements have been rewritten to perform each insert inline, significantly reducing the amount of temporary memory required and increasing the number of rows that can be processed in a single statement.

1.8. More flexibility in placement of COUNT(*) in CREATE VIEW

Previously, the CREATE VIEW statement selection list had to begin with any columns listed in the GROUP BY statement, followed by COUNT(*), and then any other aggregate functions. The restriction on the placement of COUNT(*) has been removed and it can now appear anywhere in the selection list after the GROUP BY columns.

1.9. Ability to use absolute paths to identify custom import formatters

By default, when using a custom import formatter, VoltDB looks for the formatter JAR file in the `bundles` directory where VoltDB is installed. However, you can now use a full path URL (starting with "file:") to specify any arbitrary location for the JAR file. See the description of installing and invoking custom importers in the *VoltDB Guide to Performance and Customization* manual for details.

1.10. Dynamic connections to clusters for the kafkaloader and jdbcloader utilities

The kafkaloader and jdbcloader utilities have been updated to use the auto-reconnecting client library, so they can adjust to changes in the database cluster configuration dynamically.

1.11. New message logged when processes run more than 10 seconds

If a task blocks a process queue for more than 10 seconds, an informational message is written to the server log. You can adjust the threshold at which these messages are written by setting the configuration element `<procedure loginfo="{milliseconds}"/>` in the `<systemsettings>` section of the cluster configuration file. See the appendix on server configuration options in the *VoltDB Administrator's Guide* for details.

1.12. Improved partition management when cluster nodes fail

When a node fails on a K-safe cluster, management of one or more of the database partitions (known as mastership) may need to be re-assigned to the remaining nodes. In previous versions, after several failures and rejoins mastership of the partitions could become unevenly dispersed in the cluster. Reassignment has now been improved to maintain an even distribution in the cluster.

1.13. Additional improvements

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

- Previously, VoltDB did not support what is called *unary minus*. That is, where a column name or other reference has a minus sign in front of it. For example `SELECT item, in_stock, -ordered from PRODUCT`. Unary minus is now supported.

- The VoltDB Java client includes a timeout value that is user settable and breaks a hung connection after the specified time limit. However, if the server went down while the Java client was initially authenticating the connection, the timeout was not observed and the client process would hang indefinitely. This issue has been resolved and authentication requests now obey the connection timeout limit.

- Previously, there was an edge case where, if a SQL statement utilizes certain indexes, evaluates a COUNT(*) function, and attempts to compare a VARCHAR column value to a string longer than the column's maximum length, the statement would result in a runtime error. For example, under these conditions if PRODUCTCODE is defined as VARCHAR(2), the expression "WHERE PRODUCTCODE = 'ABC'" would return an error rather than false. This issue has been resolved.

- Previously, updating the stored procedure classes (using the sqlcmd LOAD CLASSES directive or the @UpdateClasses system procedure) updated all of the classes in the schema. As a result, the more classes in the schema, the longer the update would take, even if the update itself was small. VoltDB now updates only those classes affected by the change, significantly reducing the time required for the update to complete.

- Previously, specifying a schema or a schema and a JAR file of Java classes using the `--schema` and `--classes` arguments to **voltdb init** would successfully load the specified items on **start**. However, specifying just a JAR file did not result in the classes being loaded on **start**. This issue has been resolved and use of `--classes` without `--schema` now works.

- There was also a race condition in the BulkLoader, aggravated by frequent insert failures, where it could lose track of exactly how many failures had occurred. The consequences of this race condition were that the loader utilities (such as csvloader) could report an incorrect number of failures and for applications using the BulkLoader API, any callback associated with failures might not be invoked for all failure cases. This issue has been resolved.

- Previously, if a stream column was declared as NOT NULL in the CREATE STREAM statement, it was still possible to insert a null value for that column into the stream. This issue has been resolved and the NOT NULL constraint is now properly enforced at runtime.

- Normally, dropping a table is not possible if a stored procedure exists that references that table. However, in the past if a stored procedure only accessed the table by a COUNT(*) aggregate (for example, `SELECT COUNT(*) from MyTable`) the dependency was not properly recognized and the DROP TABLE could succeed. Subsequent calls to the procedure would then generate errors. This issue has been resolved and such procedures now correctly block any DROP TABLE statement for the referenced table.

- An issue was introduced in 7.4 where, if the database was configured to import data and the configuration or schema related to import was modified while the database was paused (or a node rejoined the cluster while the database was paused), the database could crash. This issue has been resolved.

2. Release V7.4 (June 19, 2017)

2.1. New DR DROP command

There is a new **voltadmin** command to safely remove a single cluster from a multi-cluster XDCR environment without disrupting the connection between the remaining clusters. The **DR DROP** command tells a running

cluster to send any queued binary logs, stop DR and inform the remaining clusters to remove it from the XDCR environment. See the description of the **voltadmin** utility in the *Using VoltDB* manual for details and an explanation of the difference between **DR DROP** and **DR RESET**.

2.2. Support for alternate character sets in the csvloader utility

The **csvloader** utility has a new option, `--charset`, that lets you specify the character set of the input file. See the description of the **csvloader** utility in the *Using VoltDB* manual for details.

2.3. New column added to the @Statistics DRPRODUCER selector output

A new column, QUEUE_GAP, has been added to the return values of the DRPRODUCER selector of the @Statistics system procedure. This column indicates if there is a gap between the last acknowledged transaction and the next transaction queued for transmission to the DR consumer. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

2.4. The sqlcmd FILE directive now supports processing multiple files in a single batch

Previously, the **sqlcmd** FILE directive let you process a single file containing SQL statements (and other **sqlcmd** directives). The directive has been extended to support a list of files, separated by spaces. If you include the `-batch` argument, all of the files are processed as a single batch. See the description of the **sqlcmd** utility in the *Using VoltDB* manual for details.

2.5. Support for converting Java BigDecimal to integer VoltDB datatypes

Java BigDecimal values are now allowable input to VoltDB integer datatypes, including TINYINT, SHORTINT, INTEGER, and BIGINT. As with any datatype conversion, if the BigDecimal value overflows the target integer type it will generate a runtime error. See the appendix on datatype compatibility in the *Using VoltDB* manual for details.

2.6. The voltdb init --schema option now ignores unsupported sqlcmd directives

The `--schema` option to the **voltdb init** command lets you load a preliminary schema when initializing the database root. However, originally the schema file could only contain SQL DML statements. So certain files that were acceptable to **sqlcmd** were not acceptable to the **voltdb init** command because the files contained **sqlcmd** directives. The **voltdb init** command now ignores these directives and successfully processes the accompanying DML statements.

2.7. Optimization of runtime schema changes

As part of an ongoing effort to optimize and reduce the impact of changes to the running database — specifically changes to the schema, stored procedures, or configuration — the internal code to implement these actions is being reorganized. One consequence of this change is that the name of internal procedures reported by the @Statistics system procedure may change. For example, previously, schema changes were reported by the @Statistics INITIATOR and PROCEDURE selectors as the UpdateApplicationCatalog procedure. Now these actions are executed by the UpdateCore procedure.

2.8. Hash indexes are now deprecated

VoltDB uses tree indexes by default. It has been possible to request a hash index by including the string "hash" in the index name. However, testing shows that in most cases, tree indexes are more effective or at least as effective as hash indexes. Consequently, hash indexes are now deprecated and will be removed at an upcoming major release.

2.9. Additional improvements

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

- There was a very rare condition where a hash collision in the planner could cause it to select the wrong SQL plan on one node. Although still extremely rare, the chances of a collision were increased with frequent schema and procedure changes. If this happened, the database would detect a hash mismatch on the transaction, shutting down the database to preserve data integrity. This issue has been resolved.

- In cross datacenter database replication (XDCR), if the schema changes on one cluster, DR interaction should stop until the other cluster(s) are updated to match. However, previously a DROP TABLE statement on a DR table was not correctly recognized as a schema change and DR was not paused. This issue has been resolved.

- There was an issue with database replication (DR), where a producer node compared the last acknowledged ID to the last queued ID to see if DR was completely drained. However, it is possible for a server that was offline temporarily to not have some of the DR logs that the consumer needs, which another unavailable server does have. However, the producer would erroneously report that the queue was successfully drained. This issue has been resolved and the clusters now takes gaps in the queue into consideration when reporting if the DR is drained or not in response to a **voltadmin pause --wait** or **shutdown --save** command. If missing logs exist but are not currently available (because a server is down), the commands will report this to the user.

- Previously, there was an issue when using the Kinesis importer and defining two configurations that read from the same Kinesis stream but using different import procedures. The symptom was that the server reported frequent class not found exceptions. This issue has been resolved.

- When using the file export connector, the `delimiters` property is supposed to let you change the delimiters for the CSV output. However, the string value was not being properly interpreted. This issue has been resolved.

  Note, however, that the documentation was also not complete in its description of how the property was meant to be interpreted. The `delimiters` property value must obey the following rules:
  - The string represents four delimiter characters: the separator, the quote character, the escape character, and the end-of-line character.
  - All four delimiters must be specified in the aforementioned order.
  - Special and non-printing characters must be encoded as HTML entities for inclusion in the XML configuration file. For example encoding "<" as ">" and new line as "
".
  - The new line character is the only white space character that can be used as the end-of-line delimiter. Space, tab, carriage return and so on cannot be used in that position.

- When a VoltDB server process is starting up, especially during a rejoin, there was a narrow window when the programming interface became available but the database replication state was not fully instantiated. It was possible for a call to the @Statistics DRROLE selector during that window to stall. This issue has been resolved.

- Previously, starting a partial cluster using the `--missing` option, then starting the cluster with a full complement of nodes, could cause export to drop some data. The problem was that the queued export data was not correctly draining on the restart. This issue has been resolved.

- An issue was introduced in VoltDB 7.2, where if the stored procedure classes are updated without any accompanying changes to the actual schema, a cache of ad hoc statements that should have been updated was not. As a result, subsequent ad hoc queries using that cache would hang and block any further ad hoc queries. This issue has been resolved.

3. Release V7.3 (May 24, 2017)

3.1. New online upgrade process for VoltDB software

There is a new process available for upgrading VoltDB software on a running cluster using XDCR without requiring a second set of cluster hardware. Upgrading the software between two XDCR clusters is the easiest

method. However, if you do not want or have the resources for a second separate cluster, the new process uses XDCR and K-safety to split an existing cluster into two and upgrade the two halves through XDCR. See the section on "Performing an Online Upgrade with Limited Hardware" in the *VoltDB Administrator's Guide* for details.

3.2. New @Statistics selector for measuring latency

There is a new selector for the @Statistics system procedure, LATENCY, which provides periodic latency statistics. The statistics are generated every five seconds and provide both maximum and various levels of percentile measurement up to 99.999%. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

3.3. Support for Kerberos when using the VoltDB command line utilities

Kerberos support has been extended to provide integration with the VoltDB command line utilities, such as sqlcmd and voltadmin. See the discussion of "Accessing the Database from the Command Line" in the Kerberos section of the *Using VoltDB* manual for details.

3.4. Beta release of kafkaloader utility support for Kafka version 0.10.2

There is a new version of the **kafkaloader** utility, named **kafkaloader10**, that works with Kafka version 0.10.2. There are enough differences between Kafka 0.8.2 and 0.10.2 that two separate utilities will be maintained for the foreseeable future.

Customers interested in using **kafkaloader10** are encouraged to try this beta release and provide us feedback. Note however, it has not yet received the rigorous testing of the default 0.8.2 version of **kafkaloader** and should not be considered production-ready quite yet. See the online help for **kafkaloader10** in the `bin` directory for more information.

3.5. Ability to choose a preferred source for a new XDCR connection

It is now possible to select one cluster in a multi-cluster XDCR configuration as the preferred host of the initial synchronizing snapshot when the current cluster joins the environment. You use the `preferred-source` attribute in the `<connection>` configuration to identify the cluster ID of the cluster that will send the snapshot. For example, the following configuration specifies that cluster ID 2 will send the snapshot when cluster ID 3 first joins the XDCR environment:

```
<dr id="3" role="xdcr" >
   <connection source="svra,svrb,svrc" preferred-source="2" />
</dr>
```

Note that `preferred-source` only affects the initial snapshot. All other DR communication occurs among all clusters within the XDCR environment.

3.6. New Import tab in the VoltDB Management Center

There is a new tab available in the VoltDB Management Center that helps you monitor the progress of import connectors. The new *Import* tab is only visible when import connectors are configured and active.

3.7. Improved performance and reduced bandwidth use in the VoltDB Management Center

The VoltDB Management Center (VMC) has been optimized to reduce its bandwidth use and thereby improve overall performance.

3.8. Ability to export table and column names in lowercase for JDBC targets

By default, VoltDB sends SQL statements through the JDBC export connector using all uppercase table and column names. However, a few databases (most notably, PostgreSQL) are case-sensitive. There is a new property for the JDBC connector that tells VoltDB to use lowercase table and column names. The `lowercase` property accepts either true or false, with false being the default. See the description of the JDBC export connector in the *Using VoltDB* manual for details.

3.9. Deprecation notice: Java client API method updateApplicationCatalog()

There is a method in the VoltDB Java client API, `updateApplicationCatalog()`, designed for internal use based on features that are no longer available in the product. This method is now deprecated and be removed from the Java client library in the next major release of VoltDB.

3.10. Additional improvements

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

- It was possible for the return values of the @Statistics PLANNER selector to overflow their INTEGER datatype. This issue has been resolved by increasing the size of the cache hits and misses columns to BIGINT.
- There was an edge case where attempting to drop a table or stream from the database could cause a null pointer exception. This issue has been resolved.
- The new online upgrade process using limited hardware allows you to upgrade from VoltDB versions starting with 7.2. However, in its initial release, the process did not support upgrading when SSL is enabled for the external ports. This issue has now been resolved. So it will be possible to perform online upgrades of SSL-enabled clusters from version 7.3 and later.
- It was possible for the kafkaloader utility to encounter an unexpected error, stopping one of its threads, without notifying the user, giving the impression the load process completed successfully. This issue has been resolved and now unexpected errors cause the utility to stop, reporting the error to the user.
- In certain circumstances, the Kafka export connector could drop a few rows between a database pause and resume. This issue has been resolved.
- With security enabled, it was possible for VoltDB to report multiple occurrences of an error indicating it failed to read a message length while attempting to authenticate and register a user. There was no real underlying problem; the errors were caused by processes connecting and disconnecting without sending any data. This issue has been resolved and the spurious messages removed.
- There was an issue where, if an XDCR cluster recovered while another XDCR cluster was also stopped, the first cluster could not properly reset replication on the downed cluster. The **voltadmin dr reset --cluster** command would not report an error, but the DR connection would not be reset. This issue has been resolved.
- VoltDB 7.1 introduced improvements to the UPDATE CLASSES statement, that significantly reduced the time needed to load new or updated classes. Unfortunately, one of these improvements introduced a new issue where updated classes with new or modified SQL statements could result in incompatibilities and cause run-time errors. This modification has been backed out. Performance of UPDATE CLASSES is still better than versions prior to 7.1. However, the new change reduces the performance increase seen in 7.1 and 7.2.
- VoltDB does not permit schema changes while a node is rejoining the cluster or a new node is being added elastically. However, there was a race condition where, if a rejoin and a schema change were initiated at approximately the same time, it was possible for both to occur simultaneously. The consequence was that the rejoining node would receive the old rather than the updated schema. Once the rejoin operation was complete, the different schema would result in a hash mismatch error, stopping the database. This rare condition has been corrected.

4. Release V7.2 (April 25, 2017)

4.1. Ability to initialize root directory with schema and procedure classes

It is now possible to add a starting schema and procedure classes when you initialize the database root directory. The new options `--schema` and `--classes` let you specify a file containing DDL commands and a JAR file of stored procedure classes, respectively. These items are loaded automatically the first time the database is started. (And subsequent starts, if command logging is not enabled.) For example, the following command pre-loads DDL commands from the file `myschema.sql` and a set of procedure classes from `myprocs.jar`:

```
$ voltdb init --config=deployment.xml \
              --schema=myschema.sql --classes=myprocs.jar
```

See the description of the voltdb init command in the *Using VoltDB* manual for details.

4.2. Enhancements to the DR reset command for managing XDCR environments

In multi-cluster XDCR environments, it is now possible to drop one cluster from the network without disrupting replication between the remaining clusters by using the **voltadmin dr reset --cluster** command. See the description of the voltadmin command in the *Using VoltDB* manual for details.

4.3. DR support for @SwapTables

Use of the @SwapTables system procedure is now supported by database replication (DR). Swapping tables on a producer cluster acts like a schema change, causing the consumer cluster to pause until you execute the same @SwapTables statement on the consumer. See the description of the @SwapTables system procedure in the *Using VoltDB* manual for details.

4.4. Support for subqueries in data manipulation language (DML) statements

Data manipulation language (DML) statements, such as INSERT, UPDATE, and DELETE, now support subqueries. See the description of the individual statements in the appendix of supported SQL statements in the *Using VoltDB* manual for details.

4.5. New voltadmin status command

There is a new command, **voltadmin status**, that reports the current status of a running cluster. Information in the output includes the number and IP addresses of the servers in the cluster and if any nodes are missing from a K-safe cluster. You can also add the `--dr` option to include information on the current state of data replication for the cluster. See the description of the voltadmin command in the *Using VoltDB* manual for details.

4.6. New voltdb collect syntax

The syntax for the **voltdb collect** command has been enhanced to be consistent with the other **voltdb** commands, such as **get**, **init**, and **start**. The **voltdb collect** command now let's you specify the location of the root directory using the `--dir` option, like other **voltdb** commands. It also has an `--output` option that lets you specify both the name and location of the output file. Note that, although deprecated, use of the old syntax will continue to be supported during the 7.x release cycle.

4.7. Improved logging of security events and configuration changes

VoltDB now provides more thorough logging of security events. Specifically, every authorization failure is logged, each user who successfully authenticates is logged, and every configuration change is logged prior to the change being made.

4.8. New documentation of custom exporters, importers, and formatters

A new chapter on writing custom exporters, importers and formatters has been added to the *VoltDB Guide to Performance and Customization*.

4.9. 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 where the BulkLoader programming interface could lose track of the correct number of processed records, resulting in a negative value for outstanding rows, which in turn caused the drain() operation to never complete. This issue has been resolved.
- Previously, the JSON API returned the wrong error status for invalid URLs and lack of permission. It now returns error codes 404 and 401, respectively.
- Previously, it was possible to define an index with an expression that could, at run-time, generate an error (for example, using a zero column value as the denominator of an expression). When offending data was inserted into the table, an error was correctly reported but the insert was still performed. Consequently, performing SELECT operations ordered by the erroneous index would not return all valid rows. VoltDB no longer accepts such index definitions on non-empty tables. As a result, some index definitions that were allowed in previous versions will not be possible in 7.2 and later.
- Performing a port scan on the VoltDB internal port (port 3021 by default) previously resulted in the socket not closing properly. This issue has been resolved.
- There was a small memory leak possible when a database replication (DR) producer cluster recovered using command logs. This issue has been resolved.
- Previously, enabling SSL caused the VoltDB Nagios scripts to fail. This issue has been resolved.
- Previously, when using cross-datacenter replication (XDCR), if a node from a cluster involved in an XDCR environment was removed from the cluster and then used as a member of *another* cluster in the same XDCR environment, DR connections for that cluster would not start properly. This issue has been resolved.
- The HTTP export connector would fail with a null pointer exception if any rows written to that connector included a VARCHAR column with a null value. This issue has been resolved.
- Previously, if an XDCR cluster performed a **voltadmin dr reset** (which reset DR connections to all other existing clusters), adding a new cluster to the XDCR environment would not work properly. The original cluster would produce binary logs for the new cluster, but not consume the new cluster's logs. This issue has been resolved.
- There was an issue where, although an index using the "IS NOT DISTINCT FROM" clause could be parsed and added to the database schema, the index definition was not properly re-generated on subsequent schema changes. The result was any further schema changes would fail with syntax errors. This issue has been resolved.
- Until now, the Kafka importer only accepted incoming data as text. It now accepts binary data as well.

5. Release V7.1 (March 21, 2017)

5.1. Support for SSL encryption on external ports

VoltDB now supports Secure Socket Layer (SSL) encryption for all externally accessible ports, not just the HTTP port. With SSL enabled, data passing through both the client and admin ports is encrypted. If encryption is enabled for the client and admin ports, you must use the `--ssl` option on the VoltDB command line tools

(such as sqlcmd, voltadmin, and voltdb) to connect to the database. The Java and C++ client libraries also now support connecting through SSL-enabled ports.

Important

To support the new options, the configuration of SSL in the deployment file has been reorganized. Customers who previously enabled SSL encryption for the HTTP port only will need to change their configuration files. For example, the old and new syntax for enabling SSL on the HTTP port is:

Old Syntax

```
<httpd>
   <https enabled="true">
      [ . . . ]
   </https>
</httpd>
```

New Syntax

```
<ssl enabled="true">
   [ . . . ]
</ssl>
```

See the section on using SSL encryption in the *VoltDB Administrator's Guide* for details on setting up SSL for a VoltDB cluster.

5.2. Ability to retrieve the schema, procedures, and configuration from a database

Three new **voltdb** commands have been added that allow you to extract the schema, stored procedures, and configuration from the local database root directory. These commands can be used whether the database is running or not. The new commands are:

voltdb get schema
voltdb get classes
voltdb get deployment

See the description of the **voltdb** command in the *Using VoltDB* manual for details.

5.3. New statistics provide performance details about individual statements within a procedure.

A new selector, PROCEDUREDETAIL, for the @Statistics system procedure provides detailed performance information about the execution of individual statements within stored procedures. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

5.4. New system procedure lets you swap two tables

A new system procedure, @SwapTables, lets you swap the contents of two tables. The two tables must have the identical schema (that is, the same column names, datatypes, and constraints) and cannot have views defined. See the description of the @SwapTables system procedure in the *Using VoltDB* manual for details.

5.5. Ability to restart a partial cluster

It is now possible to restart a K-safe cluster with fewer than the original number of nodes. For example, if one server fails on a five node cluster and you want to restart the cluster but the failed server is not ready, you can start the cluster with the remaining four nodes by using the `--missing` argument on the **voltdb start** command, like so:

```
$ voltdb start --count=5 --missing=1
```

Note that the partial cluster is not fully K-safe to its specified K-safety value until the full complement of nodes rejoin the cluster. See the description of the **voltdb** command in the *Using VoltDB* manual for details.

5.6. New SQL functions for handling internet addresses

There are four new SQL functions to convert internet addresses from text strings to binary and back again. Two functions (INET_ATON and INET_NTOA) convert IPv4 addresses and two (INET6_ATON and INET6_NTOA) convert IPv6 addresses. See the Appendix of SQL Functions in the *Using VoltDB* manual for details.

5.7. Statistics related to schema changes are no longer reset by each change

Previously, the information gathered by the @Statistics system procedure related to schema changes (from DML statements or calls to @UpdateClasses or @UpdateApplicationCatalog) was reset each time the database schema changed. Schema changes no longer reset the statistics on each invocation. Also, every schema change is recorded, rather than the 5% sampling done for other procedures.

5.8. SNMP traps now report database replication (DR) events

VoltDB can send SNMP traps for selected database events, such as node failures or when resource limits are reached. Starting with V7.1, it also reports database replication (DR) events when DR is enabled.

5.9. New index optimizations

We continue to extend support for the use of indexes to optimize the execution of SQL statements. Two new uses of indexes are when evaluating:

- Window functions
- NOT DISTINCT predicates

5.10. Enhancements to the kafkaloader utility

Three new features have been added to the kafkaloader utility:

- Support for a Kafka consumer configuration file, including the ability to set the *group.id* property. Use the `--config` argument to specify the configuration file.
- Support for custom formatters. Use the `--formatter` argument to specify custom formatter properties.
- Support for large partition counts. Use the `-k` argument to specify the number of Kafka partitions per topic if there are more than 10 partitions.

See the online help for the kafkaloader utility for details.

5.11. Improved performance when updating procedure classes

Previously, updating the stored procedure classes (using the sqlcmd LOAD CLASSES directive or the @UpdateClasses system procedure) updated all of the classes in the schema. As a result, the more classes in the schema, the longer the update would take, even if the update itself was small. VoltDB now updates only those classes affected by the change, significantly reducing the time required for the update to complete.

5.12. Managing heterogeneous clusters in Database Replication (DR)

You can create a DR environment using clusters running different versions of VoltDB. However, to support new functionality (in particular, multi-cluster XDCR) the DR protocol changed in V7.0. That major version acts as a gateway between earlier and later releases. In other words, you can run a V7.0 cluster with earlier versions, or you can run a 7.0 cluster with later versions. However, you cannot run pre-V7.0 clusters with post-V7.0 clusters.

As a consequence, if you want to perform an in-service upgrade (using different version clusters) to upgrade a DR environment using a pre-V7.0 version (say, V6.9) to V7.1 or later, you must upgrade all clusters to V7.0 first, then upgrade to your final target version. Of course, you can always use the normal single-cluster upgrade process to go directly from V6.x to V7.x if you choose.

5.13. VoltDB servers no longer support SHA-1

To ensure the security of VoltDB servers, the database no longer accepts SHA-1 for hashed data. All hashed data (such as passwords etc) must be hashed using SHA-256. **Note: this is an incompatible change.** Some earlier client libraries will not be able to connect to the V7.1 servers because they do not support the more robust SHA-256 hashing.

5.14. 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 where, if multiple VoltDB client connections were created in separate threads of a Java application, a race condition could cause the clients to attempt to use shared resources before they were properly initialized. The symptom of this problem was that the connections failed reporting an illegal state exception accessing the ReverseDNSCache. This issue has been resolved.

- Previously, VoltDB created a file (`log/volt.log`) in the user's current working directory whenever you started the **sqlcmd** utility. However, no messages were ever written to the log, so this superfluous file is no longer created.

- If a database is run without command logging but with automated snapshots, the database is not restored when the database process is stopped and restarted with the **voltdb start** command. Automated snapshots are intended as a form of backup, not persistent durability. However, previously, backups from a previous session would eventually be deleted as part of the cleanup of older backups. To avoid this, when VoltDB starts without command logs, any snapshots present in the snapshot directory are archived to a separate directory so they will not be accidentally deleted.

- If a database is not using command logging, each time you stop and restart it with the **voltdb start** command, it should start a fresh, empty database (with no durability). However, previously, if the earlier session used certain features (such as export) and left overflow data in the database root directory, the **voltdb start** command would not allow this data to be overwritten and would force the user to re-initialize the root directory. This problem has been corrected and the **voltdb start** command starts a fresh database instance, even if there is existing overflow data as long as no other durability data (such as command logs) is present.

- In earlier releases, it was possible for the database server to report a ZooKeeper CancelledKeyException error during shutdown. This error was harmless, but unnecessary. It has been removed.

- Previously, attempting to create a stream with 256 columns could crash the database. This issue has been resolved.

- If a table contained a very large VARCHAR column and the table had a view defined including either a MIN or MAX aggregate function, deleting a row with a large value in the VARCHAR column could result in a fatal error. This issue has been resolved. It is still possible for the DELETE operation to fail, but now the appropriate error is returned to the user.

- There was an edge case, where if a table had a VARCHAR or VARBINARY column that was indexed as part of a complex index and the column was also used in an MIN or MAX aggregate function of a view, updating that column could result in a fatal SIGSEGV error. This issue has been resolved.
- Previously, changing the Log4j configuration on a running database (for example, by calling @UpdateLogging) and specifying a relative path for the log file, would result in the file location being interpreted relative to the user's current working directory. This has been corrected and relative paths are now interpreted as relative to the database root directory.

6. Release V7.0.1 (March 16, 2017)

6.1. Recent improvements

The following limitations in previous versions have been resolved:

- There was an issue in the **voltadmin shutdown --save** command introduced in 6.8. If command logging was not enabled on the cluster, the final snapshot would *not* be properly restored on the subsequent **voltdb start**. This problem has now been fixed.
- There was an issue, introduced in V6.7, that caused database replication (DR) between mixed version clusters to fail when replicating from a 6.7 or later cluster to a pre-6.7 cluster. If the more recent version cluster's schema includes a view on a DR table, replication would fail to start or break when a schema change adds such a view. This issue has been resolved.
- Previously, if a cluster was configured for cross datacenter replication (XDCR), but the deployment did not specify a `<connection>` element or the `<connection>` element was explicitly disabled (with the `enabled="false"` attribute), the cluster could generate a null pointer exception. This issue has been resolved and the error is no longer generated. Note, however, that explicitly disabling the connection is not a common configuration and XDCR replication will not operate properly until the connection is enabled.
- The import infrastructure has been redesigned to reduce resource contention and optimize the way back pressure is handled. These changes should result in significantly improved import performance.
- Previously, the @Statistitcs system procedure DRPRODUCER selector could erroneously report that there was data still waiting to be drained even though the queue was empty. This issue has been resolved.

7. Release V7.0 (January 30, 2017)

7.1. Updated operating system and software requirements

The operating system and software requirements for VoltDB have been updated based on changes to the supported versions of the underlying technologies. Specifically:

- Ubuntu 12.04 is no longer supported. The supported operating system versions are CentOS 6.6, CentOS 7.0, RHEL 6.6, RHEL 7.0 and Ubuntu 14.04 and 16.04, with support for OS X 10.8 and later as a development platform.
- The VoltDB server process requires Java 8. The Java client library supports both Java 7 and 8.
- The required version for the Python client and VoltDB command line u+tilities is 2.6.

7.2. Improved configuration and reporting for database replication (DR)

As mentioned earlier, the addition of the `role` attribute to the `<dr>` element in the configuration file eliminates the need for the SET DR=ACTIVE SQL statement and the `--replica` argument to the **voltdb start** command. In addition, two columns for the local and remote cluster IDs have been added to the DRPRODUC-

ER and DRCONSUMER selectors for the @Statistics system procedure. The new columns make it easier to identify to which DR connection the current row of data applies. However, their presence also changes the order of columns from previous releases. If an application used the DRPRODUCER or DRCONSUMER selectors and retrieves column values by index instead of by name, the application will need to be modified to accommodate the new columns in the results.

7.3. VoltDB now protects against inserting invalid values into TIMESTAMP columns

Previously, VoltDB allowed any 8-byte integer as input to a TIMESTAMP column. However, the TIMESTAMP functions limit the values to those defined by the MIN_VALID_TIMESTAMP() and MAX_VALID_TIMESTAMP() functions. Starting with V7.0, VoltDB no longer accepts values outside of the valid timestamp range for TIMESTAMP columns.

Note that this is an *incompatible* change and may require changes to applications that are inadvertently inserting invalid timestamp values. If so, you can use the IS_VALID_TIMESTAMP() function to validate input values before inserting them into TIMESTAMP columns.

7.4. Additional improvements

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

- There was a race condition where occasionally, when starting XDCR, the request for a synchronizing snapshot from the consumer cluster resulted in an error on the producer cluster indicating it could not initiate the snapshot This issue has been resolved.
- An issue with elastic expansion of VoltDB clusters (that is, adding nodes to a cluster "on the fly"), was introduced in V6.9. Attempting to add a node could result in the join operation hanging. This did not happen all the time, but has now been corrected and is no longer an issue in V7.0.
- Previously, the VoltDB export subsystem was not properly releasing all memory from the Java heap during schema or configuration changes. With frequent updates to the schema or configuration, this unused memory could, potentially, accumulate until the Java heap was exhausted. This issue has been resolved.

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. 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.

2.2. Avoid bulk data operations within a single transaction when using database replication

Bulk operations, such as large deletes, inserts, or updates are possible within a single stored procedure. However, if the binary logs generated for DR are larger than 45MB, the operation will fail. To avoid this situation, it is best to break up large bulk operations into multiple, smaller transactions. A general rule of thumb is to multiply the size of the table schema by the number of affected rows. For deletes and inserts, this value should be under 45MB to avoid exceeding the DR binary log size limit. For updates, this number should be under 22.5MB (because the binary log contains both the starting and ending row values for updates).

2.3. Database replication ignores resource limits

There are a number of VoltDB features that help manage the database by constraining memory size and resource utilization. These features are extremely useful in avoiding crashes as a result of unexpected or unconstrained growth. However, these features could interfere with the normal operation of DR when passing data from one cluster to another, especially if the two clusters are different sizes. Therefore, as a general rule of thumb, DR overrides these features in favor of maintaining synchronization between the two clusters.

Specifically, DR ignores any resource monitor limits defined in the deployment file when applying binary logs on the consumer cluster. DR also ignores any partition row limits defined in the database schema when applying binary logs. This means, for example, if the replica database in passive DR has less memory or fewer unique partitions than the master, it is possible that applying binary logs of transactions that succeeded on the master could cause the replica to run out of memory. Note that these resource monitor and tables row limits *are* applied on any original transactions local to the cluster (for example, transactions on the master database in passive DR).

2.4. Different cluster sizes can require additional Java heap

Database Replication (DR) now supports replication across clusters of different sizes. However, if the replica cluster is smaller than the master cluster, it may require a significantly larger Java heap setting. Specifically, if the replica has fewer unique partitions than the master, each partition on the replica must manage the incoming binary logs from more partitions on the master, which places additional pressure on the Java heap.

A simple rule of thumb is that the worst case scenario could require an additional P * R * 20MB space in the Java heap , where P is the number of sites per host on the replica server and R is the ratio of unique partitions on the master to partitions on the replica. For example, if the master cluster is 5 nodes with 10 sites per host and a K factor of 1 (i.e. 25 unique partitions) and the replica cluster is 3 nodes with 8 sites per host and a K factor of 1 (12 unique partitions), the Java heap on the replica cluster may require approximately 320MB of additional space in the heap:

Sites-per-host * master/replace ratio * 20MB
8 * 25/12 * 20 = ~ 320MB

An alternative is to reduce the size of the DR buffers on the master cluster by setting the DR_MEM_LIMIT Java property. For example, you can reduce the DR buffer size from the default 10MB to 5MB using the VOLTDB_OPTS environment variable before starting the master cluster.

```
$ export VOLTDB_OPTS="-DDR_MEM_LIMIT=5"

$ voltdb start
```

Changing the DR buffer limit on the master from 10MB to 5MB proportionally reduces the additional heap size needed. So in the previous example, the additional heap on the replica is reduced from 320MB to 160MB.

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 --force** 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.

3.8. XDCR cannot be used with deprecated export syntax.

You cannot use cross-datacenter replication (XDCR) with the deprecated export syntax, that is the EXPORT TABLE statement. To use XDCR with export, you must use the current CREATE STREAM syntax for declaring the source streams for export targets.

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.

5.2. Kafka import can lose data if multiple nodes stop in succession.

There is an issue with the Kafka importer where, if multiple nodes in the cluster fail and restart, the importer can lose track of some of the data that was being processed when the nodes failed. Normally, these pending imports are replayed properly on restart. But if multiple nodes fail, it is possible for some in-flight imports to get lost. This issue will be addressed in an upcoming release.

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. SNMP

8.1. Enabling SNMP traps can slow down database startup.

Enabling SNMP can take up to 2 minutes to complete. This delay does not always occur and can vary in length. If SNMP is enabled when the database server starts, the delay occurs after the server logs the message "Initializing SNMP" and before it attempts to connect to the cluster. If you enable SNMP while the database is running, the delay can occur when you issue the **voltadmin update** command or modify the setting in the VoltDB Management Center Admin tab. This issue results from a Java constraint related to secure random numbers used by the SNMP library.

9. VoltDB Management Center

9.1. The VoltDB Management Center currently reports on only one DR connection.

With VoltDB V7.0, cross-datacenter replication (XDCR) supports multiple clusters in an XDCR network. However, the VoltDB Management Center currently reports on only one such connection per cluster. In the future, the Management Center will provide monitoring and statistics for all connections to the current cluster.

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"*.

If you refer to a table or column name that does not exist, VoltDB reports that the *"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.

2.5. Ambiguous column references no longer allowed.

Starting with VoltDB 6.0, ambiguous column references are no longer allowed. For example, if both the *Customer* and *Placedorder* tables have a column named *Address*, the reference to *Address* in the following SELECT statement is ambiguous:

```
SELECT OrderNumber, Address FROM Customer, Placedorder
   . . .
```

Previously, VoltDB would select the column from the leftmost table (Customer, in this case). Ambiguous column references are no longer allowed and you must use table prefixes to disambiguate identical column names. For example, specifying the column in the preceding statement as *Customer.Address*.

A corollary to this change is that a column declared in a USING clause can now be referenced using a prefix. For example, the following statement uses the prefix *Customer.Address* to disambiguate the column selection from a possibly similarly named column belonging to the *Supplier* table:

```
SELECT OrderNumber, Vendor, Customer.Address
```

```
FROM Customer, Placedorder Using (Address), Supplier
 . . .
```

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.