VOLTDB

APPLICATION BRIEF

DURABILITY

Durability is one of the four key ACID attributes required to ensure the accurate and reliable operation of a transactional database. Simply put, durability ensures that any transaction that succeeds is committed to the database and made persistent. Even in the event of a power failure or other unexpected occurrence, transactions that have been committed survive permanently. In reverse, any transactions that fail or are interrupted will "roll back" and not affect database integrity.

As simple as the concept of durability sounds, its impact on client applications is tremendous. Durability relieves the application developer from having to account for and handle the myriad situations that could negatively impact database consistency. Durability lets the application developer focus on workflow while the database ensures the accuracy and consistency of the data despite external forces.

Durability in VoltDB

Support for durability in VoltDB begins with its transactional model. Stored procedures are transactions in VoltDB and each stored procedure invocation succeeds or fails as a whole. You write VoltDB stored procedures as Java classes, which allow you to include both SQL queries and additional programming logic within the transaction itself.

Once the transaction succeeds and the changes are committed to the database, VoltDB provides several features that ensure the changes become durable. If the database server stops for any reason, the contents are preserved and can be restored to a consistent state. Which feature you use depends on the level of durability you require:

- **Database Snapshots** provide basic durability. Snapshots are exactly what they sound like: disk-based copies of the database contents at a moment in time. You can initiate snapshots manually. For ongoing persistence, however, it is better to enable automatic snapshots when starting the database. You configure how frequently snapshots are taken (usually in terms of minutes or hours) as part of the deployment settings. If the server fails for any reason, you can use the snapshots to recover the database to its last known state on restart.
- **Asynchronous Command Logging** provides enhanced durability by creating both snapshots and a log of all transactions between snapshots. With command logs, if the server fails and the database is then restarted, not only can VoltDB recover the last snapshot, but it can also replay all subsequent transactions in the log. Use of command logs can reduce the number of transactions lost from possibly several

minutes' worth — when using snapshots alone — to a fraction of a second's worth when using both. Only those transactions not written to the log as the server fails are not captured. The level of durability is, again, configurable when enabling asynchronous command logging.

- **Synchronous Command Logging** provides the most complete durability possible. Like asynchronous command logging, synchronous logging captures the individual transactions between snapshots. But with synchronous logging, the log is written after the transaction completes and before it is committed to the database. In other words, no transactions are committed that are not logged and no transactions are lost. The only drawback is that more advanced disk technology is required to keep up with VoltDB's high transaction throughput rate.

VoltDB provides durability options to meet any and all application requirements and budgets, from the simplest incremental snapshots to complete durability, up to the very last transaction committed. Combined with VoltDB's high availability capabilities, these features provide complete persistence and reliability for your data.

209 Burlington Road, Suite 203
Bedford, MA 01730
Phone: +1.978.528.4660
Fax: +1.978.528.0568
http://voltdb.com

VOLTDB