IEP-3: Transactional SQL

ID	IEP-3
Author	Semen Boikov Alexey Goncharuk Vladimir Ozerov Sergey Puchnin
Sponsor	Vladimir Ozerov
Created	22 Sep 2017
Status	DRAFT

Motivation

At the moment Apache Ignite's SQL engine doesn't support transactions. `SELECT` statements are executed on top of committed data without creating a snapshot. DML statements are executed as a series of batched updates using `IgniteCache.invokeAll` command. As a result it is neither possible to get consistent view of data, nor to update it with ACID semantics.

We need to implement transactions support in SQL on top new MVCC protocol.

Description

Locks

TX SQL will be implemented on top of existing snapshot-based MVCC infrastructure. Writes obtain locks on keys. Reads do not obtain locks. Writes do not block reads. Reads can be converted to blocking mode using `SELECT ... FOR UPDATE` statement.

Isolation modes

In the first iteration only **READ_COMMITTED** mode will be supported.

- Every operation first acquire global sequence number on coordinator. Both SELECT and update operations use sequence number to filter more recent updates. This way operation see only rows which existed by the time operation began.
- Update operation then acquires locks on target rows one by one. Row might have been already locked by concurrent transaction at this point. If concurrent transaction is rolled back or and lock is acquired on expected version no additional actions are required. If concurrent transaction modifies the row and commits, current transaction acquires the lock and **re-evaluates** original condition. If condition evaluates to true still, then lock is retained. Otherwise lock is released and row is ignored.
- Subsequent SELECTs see previous updates
- On TX commit client requests another sequence number which is applied to all modified rows.

All DML requests are split into two groups: with and without reduce step. If reduce step is not needed, locks are obtained on map nodes immediately. If reduce step is needed (e.g. non-collocated aggregation), then we cannot lock rows on mapper immediately, because we do not know target row set in advance. In this case filter condition should be re-evaluated as well by executing distributed query again (TBD),

Transactional protocol

Typical DML operation may modify any number of rows. it means we cannot store all modified rows on a near node. Current TX protocol must be extended, so that updates are stored on primary/backup nodes only and not transferred to near node.

Native API changes

No changes to existing API is needed. SQL statements will be enlisted into ongoing transaction if one is available. Only PESSIMISTIC /READ_COMMITTED is valid TX mode in the first iteration. Attempt to enlist SQL statement into any other TX mode will produce an exception.

Drivers support (JDBC/ODBC)

- Support for `BEGIN TRANSACTION`, `ROLLBACK` and `COMMIT` commands will be implemented on the server side. They will be mapped to appropriate methods on `IgniteTransactions` facade
- Minor changes to JDBC/ODBC drivers will be required (metadata, auto-commit, etc).

Cross-cache statements

Only MVCC-enabled transactional caches could be enlisted into transaction. An exception is throw If SQL statement use either `ATOMIC` cache or `TRANSACTIONAL` cache without MVCC support.

Deadlocks

When executing DML statements locks are typically acquired in unpredictable order, what may cause deadlocks. Typically RDBMS vendors implement deadlock detectors which rollback last statement in case deadlock is found. Deadlock detection is expensive in distributed environment as it requires coordination between nodes over networks. In the first iteration we can define per-statement lock timeout. If locks cannot be obtained in the given timeout, statement is rolled-back and appropriate exception is thrown.

Risks and Assumptions

TBD

Discussion Links

TBD

Reference Links

N/A

Tickets

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
IGNITE-9410	Add transactions support to thin clients		Aug 29, 2018	Aug 15, 2019		Aleksey Plekhanov	Vladimir Ozerov		PATCH AVAILABLE	Unresolved
IGNITE-5937	Mvcc data structure for SQL queries		Aug 04, 2017	Aug 16, 2018		Semen Boikov	Semen Boikov		CLOSED	Fixed
IGNITE-5934	Integrate mvcc support in sql query protocol		Aug 04, 2017	Aug 16, 2018		Igor Seliverstov	Semen Boikov		CLOSED	Fixed
IGNITE-5933	Integrate mvcc support in cache.getAll protocol		Aug 04, 2017	Aug 16, 2018		Semen Boikov	Semen Boikov		CLOSED	Fixed
IGNITE-7991	MVCC TX Crash recovery		Mar 19, 2018	Dec 11, 2018		Unassigned	Igor Seliverstov		CLOSED	Duplicate
IGNITE-7956	MVCC TX: cache eviction operations for key-value API		Mar 14, 2018	Mar 11, 2019		Unassigned	Alexander Paschenko		CLOSED	Won't Fix

IGNITE-7954	MVCC TX: cache load routines for key-value API		Mar 14, 2018	Mar 11, 2019	Unassigned	Alexander Paschenko		CLOSED	Won't Fix
IGNITE-7952	MVCC TX: cache clear routines for key-value API		Mar 14, 2018	Mar 11, 2019	Unassigned	Alexander Paschenko		CLOSED	Won't Fix
IGNITE-6149	Create mvcc prototype application		Aug 22, 2017	Aug 16, 2018	Semen Boikov	Semen Boikov		CLOSED	Fixed
IGNITE-10966	MVCC: Add scale factor support in Mvcc tests suites.		Jan 17, 2019	Mar 15, 2019	Unassigned	Andrew Mashenkov		CLOSED	Not A Problem
IGNITE-11228	MVCC: Implement transaction savepoints.		Feb 06, 2019	Mar 11, 2019	Unassigned	Andrew Mashenkov		CLOSED	Won't Fix
IGNITE-11229	MVCC: Implements chained transactions.		Feb 06, 2019	Mar 11, 2019	Unassigned	Andrew Mashenkov		CLOSED	Won't Fix
IGNITE-11170	MVCC: TxLog data region can be cleaned if all mvcc caches gone.		Feb 01, 2019	Mar 11, 2019	Unassigned	Andrew Mashenkov		CLOSED	Duplicate
IGNITE-11242	Update counters mismatch after loading data via DataStreamer into MVCC-enabled caches		Feb 07, 2019	Mar 11, 2019	Ivan Pavlukhin	Ivan Artukhov		CLOSED	Won't Fix
IGNITE-10871	MVCC: Support indexing SPI in mvcc mode		Jan 09, 2019	Mar 11, 2019	Unassigned	Andrew Mashenkov		CLOSED	Fixed
IGNITE-11503	MVCC: Handle partition loss policies		Mar 07, 2019	Mar 12, 2019	Unassigned	Ivan Pavlukhin		CLOSED	Duplicate
IGNITE-10738	MVCC: Enable page evictions for MVCC caches with in-memory mode		Dec 19, 2018	Mar 11, 2019	Unassigned	Roman Kondakov		CLOSED	Won't Fix
IGNITE-10583	MVCC: Assertion on txLog state update when recovering from WAL.		Dec 06, 2018	Mar 24, 2019	Andrew Mashenkov	Roman Kondakov		CLOSED	Duplicate
IGNITE-10841	Edge-chasing deadlock detection monitoring		Dec 28, 2018	Mar 11, 2019	Unassigned	Ivan Pavlukhin		CLOSED	Won't Fix
IGNITE-10788	MVCC: Get operation may hang in some cases		Dec 21, 2018	Mar 11, 2019	Andrew Mashenkov	Roman Kondakov		CLOSED	Duplicate

Showing 20 out of 359 issues