ClustrixDB DML

This page describes ClustrixDB's support for data manipulation including read-only SELECT statement.

Supported DML

ClustrixDB supports most SQL syntax including:

- SELECT, DISTINCT, [LEFT | RIGHT | OUTER] JOIN, STRAIGHT_JOIN
- UNION, HAVING, GROUP BY, LIMIT, ASC, DESC, ORDER BY, FOR UPDATE
- Subqueries, including with IN
- EXISTS, NOT EXISTS
- INSERT, INSERT... ON DUPLICATE KEY UPDATE, UPDATE, DELETE, REPLACE INTO
- CREATE, DROP, TRUNCATE, AUTO_INCREMENT
- LOAD DATA INFILE
- START TRANSACTION, COMMIT, ROLLBACK, PREPARE
- COUNT(), AVG(), STD(), SUM(), MAX(), MIN(), GROUP_CONCAT()

as well as

- Foreign Keys
- Data Types
- EXPLAIN
- Partitioned Tables (RANGE only)
- MySQL Replication
- Stored Routines
- Triggers

While we do not synchronize our feature development with MySQL releases, we strive to provide the features and functions required by our customers to successfully run production environments. We monitor both our customers' requirements and new MySQL functionality to determine our feature roadmap and use recent shipping releases of MySQL as part of an automated QA process for compatibility.

Unsupported DML

ClustrixDB does not support:

- ASC or DESC qualifier for GROUP BY; instead, for ordered output, ORDER BY must be explicit. For example:
  select a,count(*) from foo group by 1 desc; -- will give a syntax error
  select a,count(*) from foo group by 1 order by desc; -- OK
- COLLATE in SELECT
- CUBE
- EXCEPT
- IGNORE keyword as part of a DELETE statement
- INTERSECT
- LOCK TABLES statement
- NOT in conjunction with user-defined variables that reference a function, e.g. SET @a = not foo();
- ROLLUP
- SELECT INTO ... OUTFILE option. Use mysql -e "SELECT ..." > file_name instead.
- Subqueries with ALL or SOME

Caveats for SQL Support

SELECT subqueries

When using the ANY or IN options in SELECT subqueries, ClustrixDB will return a numeric value instead of a NULL value when the subquery resolves to NULL. However, in MySQL when the subquery resolves to NULL, MySQL returns a NULL value. For example, assume numeric values are inserted into the following tables t1 and t2 and then NULL is inserted into t2. When the subquery below is run in ClustrixDB, the numeric value 2 is returned instead of N ULL.

```
sql> Insert into t1 value (1), (2);
sql> Insert into t2 values (1);
sql> Insert into t2 values (NULL);

sql>  select a from t1 where a not IN (select b
from t2);
+----+
| 2  |
+----+
1 row in set (0.00 sec)
```

Inline Variable Evaluation

ClustrixDB uses a parallel evaluation model, which can result in non-deterministic results for inline variable evaluation:

```
sql>  create table foo (a int, b int);
sql>  insert into foo values (2, 0), (4, 0), (6, 0);
sql> set @rc := 1;
sql> insert into foo select @rc := @rc + 1, a from foo limit 1;
```

Depending on when the ClustrixDB planner applies the LIMIT, the value for rc may be incremented multiple times:

```
sql> select @rc;
+------+
| @rc  |
+------+
| 2    |
+------+

sql> select * from foo order
by a, b;
+------+------+
| a    | b    |
+------+------+
|    2 |    0 |
|    2 |    2 |
|    4 |    0 |
|    6 |    0 |
+------+------+
4 rows in set (0.00 sec)
```

Duplicate Handlers for SQLSTATE Code

ClustrixDB allows duplicate handlers for the same SQLSTATE code within a stored procedure block. MySQL errors out during compilation.

Locking Behavior

LOCK TABLE

ClustrixDB ignores LOCK TABLE syntax. No error is returned, because dump files generated by mysqldump include LOCK TABLE commands and ClustrixDB must be able to process such files.

However, ClustrixDB does support LOCK TABLES ... FOR UPDATE which accepts a list of relations to issue an exclusive table lock on the listed relations. With this lock, inserts by other transactions are still allowed, but UPDATE, DELETE, or SELECT FOR UPDATE will be blocked. Note that for LOCK TABLES ... FOR UPDATE to be effective, it must be issued within the context of an explicit transaction, as the table lock is held until that transaction is committed or rolled back (which will be instantly if in autocommit mode).

It is also possible to have LOCK TABLES ... FOR UPDATE also block inserts. There is a performance penalty associated with having inserts check for this lock, so this behavior is not enabled by default. This behavior is enabled on a per-table basis with ALTER TABLE table_name ENABLE INSERT_LOCK, and can be disabled with ALTER TABLE table_name DISABLE INSERT_LOCK. A table with this behavior enabled will include INSERT_LOCK at the end of SHOW CREATE TABLE output:

```
sql> show create table foo\G
*************************** 1. row ***************************
       Table: foo
Create Table: CREATE TABLE `foo` (
  `id` int(11)
) CHARACTER SET utf8 /*$ REPLICAS=2 SLICES=6 INSERT_LOCK */
```

Here is an example of the behavior, where the s1> and s2> prompts indicate two separate sessions of mysql client:

```
s1> alter table foo enable insert_lock;
Query OK, 0 rows affected (0.22 sec)

s1> begin;
Query OK, 0 rows affected (0.00 sec)

s1> lock tables foo for update;
Query OK, 0 rows affected (0.03 sec)

s2> insert into foo values (666);
[session blocks]

s1> commit;
Query OK, 0 rows affected (0.00 sec)

[session s2 resumes]
Query OK, 1 row affected (2 min 14.29 sec)
```

Temporary Tables

- ClustrixDB supports self-inserts for temporary tables, where MySQL does not.

See also:

- Administrative SQL statements
- Function and Operators
- START TRANSACTION