Replicated Database Engine in ClickHouse

Replicated database engine

- Based on Atomic engine
- Executes all DDL queries (almost) like ON CLUSTER queries
- Metadata of tables are stored in ZooKeeper
- Creation of new replicas and recovery of staled replicas
- Dynamic cluster configuration in ZooKeeper

DDL ON CLUSTER

```
<remote_servers>
    <test>
        <shard>
            <replica>
                <host>node1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>node2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>node3</host>
                <port>9000</port>
            </replica>
        </shard>
    </test>
</remote_servers>
```

DDL ON CLUSTER

```
<remote_servers>
    <test>
        <shard>
            <replica>
                <host>node1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>node2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>node3</host>
                <port>9000</port>
            </replica>
        </shard>
    </test>
</remote_servers>
```

```
node1 :) ALTER TABLE t ON CLUSTER test ADD COLUMN ...


┌host──┬status┬─error──────────────────────────────  ...
│node1│    0 │
│node2│    0 │
│node3│    0 │
└─────┴──────┴───────────────────────────────────  ...
```

DDL ON CLUSTER

```
<remote_servers>
    <test>
        <shard>
            <replica>
                <host>node1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>node2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>node3</host>
                <port>9000</port>
            </replica>
        </shard>
    </test>
</remote_servers>
```

```
node1 :) ALTER TABLE t ON CLUSTER test ADD COLUMN ...


┌host──┬status┬─error────────────────────────────── ...
│node1│    0 │
│node2│  517 │ Metadata on replica is not up to date ...
│node3│    0 │
└──────┴──────┴──────────────────────────────────── ...

There was an error on [node2:9001]: Code: 517,
e.displayText() = DB::Exception: Metadata on replica is
not up to date with common metadata in Zookeeper. Cannot
alter


node2 :) ALTER TABLE t ADD COLUMN ...
```

DDL ON CLUSTER

```
<remote_servers>
    <test>
        <shard>
            <replica>
                <host>node1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>node2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>node3</host>
                <port>9000</port>
            </replica>
        </shard>
    </test>
</remote_servers>
```

+

```
<replica>
    <host>node4</host>
    <port>9000</port>
</replica>
```

How to create a database

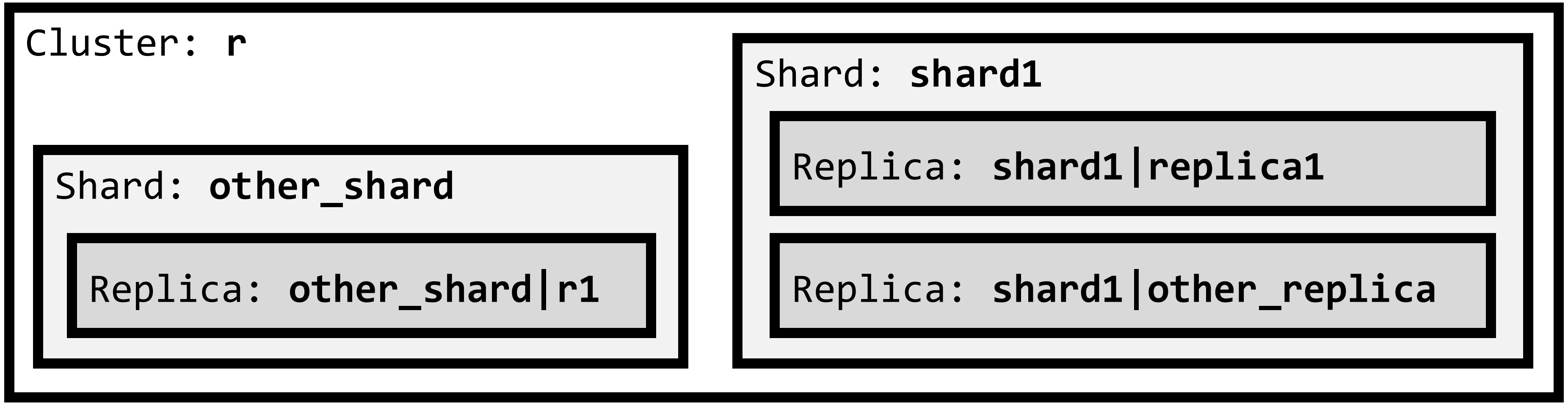
```
node1 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'replica1')
```

```
node2 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'other_replica')
```

```
node3 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', '{replica}')
```

How to create a database

```
node1 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'replica1')
node2 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'other_replica')
node3 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', '{replica}')
```



Cluster: **r**

Shard: **other_shard**

Replica: **other_shard|r1**

Shard: **shard1**

Replica: **shard1|replica1**

Replica: **shard1|other_replica**

DDL queries

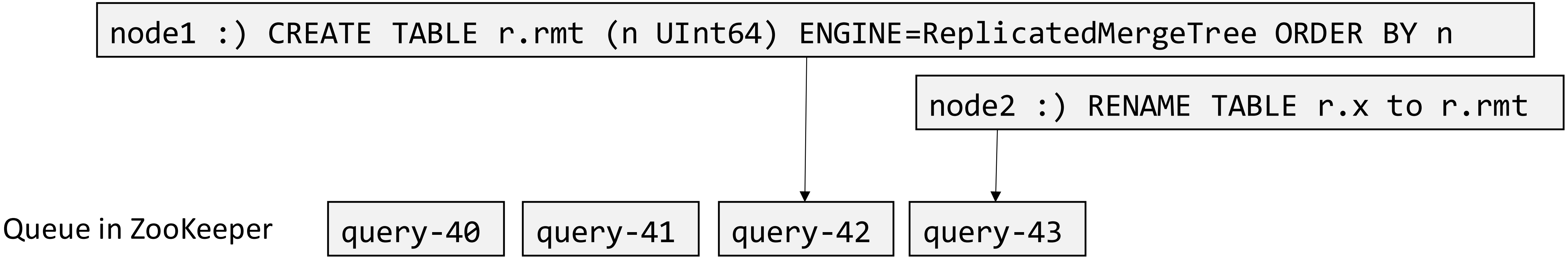
```
node1 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'replica1')
node2 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'other_replica')
node3 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', '{replica}')
```

```
node1 :) CREATE TABLE r.rmt (n UInt64) ENGINE=ReplicatedMergeTree ORDER BY n


┌─host─────────────────┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_active─┐
│ shard1|replica1      │      0 │       │                   2 │                0 │
│ shard1|other_replica │      0 │       │                   1 │                0 │
│ other_shard|r1       │      0 │       │                   0 │                0 │
└──────────────────────┴────────┴───────┴─────────────────────┴──────────────────┘
```

DDL queries

- Query acuires a sequential number in replication queue



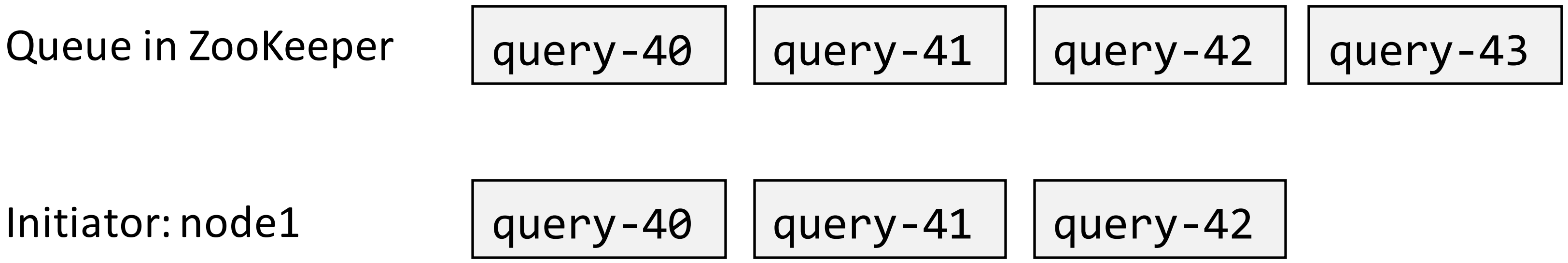
```
node1 :) CREATE TABLE r.rmt (n UInt64) ENGINE=ReplicatedMergeTree ORDER BY n
```

```
node2 :) RENAME TABLE r.x to r.rmt
```

Queue in ZooKeeper

DDL queries

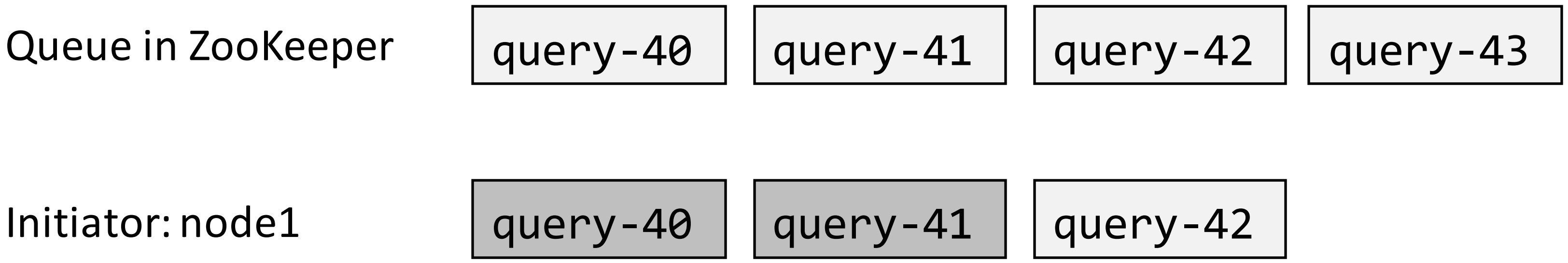
- Query acuires a sequential number in replication queue
- Initiator waits for previous queries in the queue to be executed



Queue in ZooKeeper	`query-40`	`query-41`	`query-42`	`query-43`
Initiator: node1	`query-40`	`query-41`	`query-42`	

DDL queries

- Query acuires a sequential number in replication queue
- Initiator waits for previous queries in the queue to be executed
- Initiator tries to execute the query

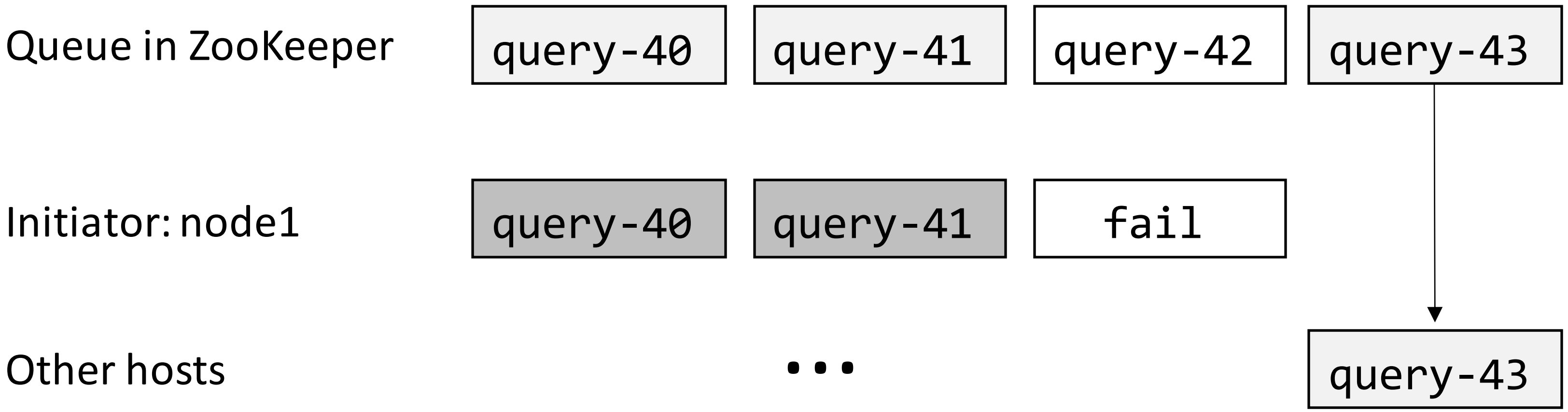


Queue in ZooKeeper: `query-40` `query-41` `query-42` `query-43`

Initiator: node1: `query-40` `query-41` `query-42`

DDL queries

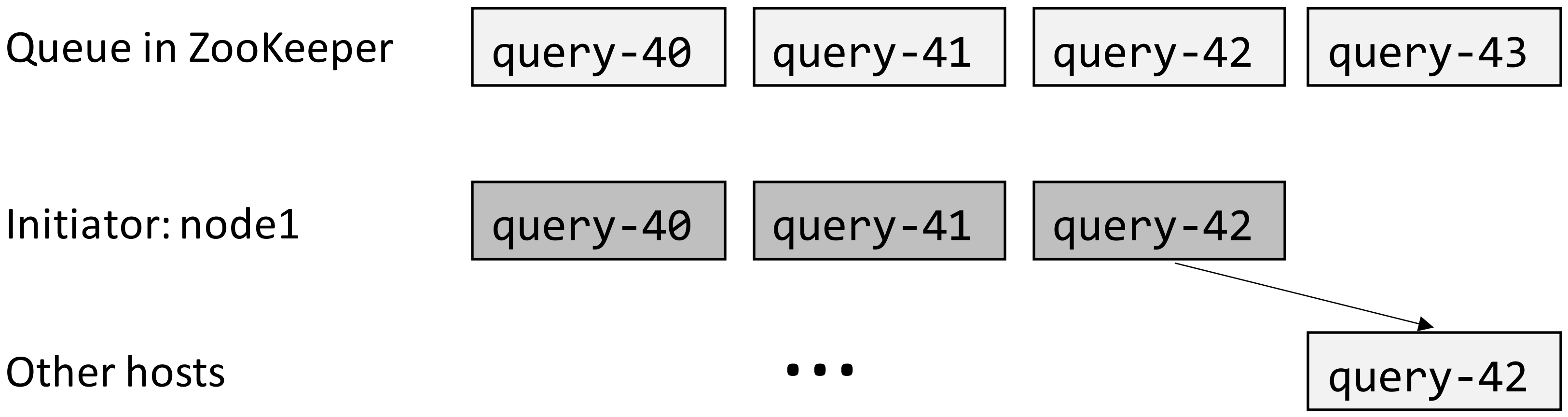
- Query acuires a sequential number in replication queue
- Initiator waits for previous queries in the queue to be executed
- Initiator tries to execute the query



Queue in ZooKeeper	query-40	query-41	query-42	query-43
Initiator: node1	query-40	query-41	fail	
Other hosts		•••		query-43

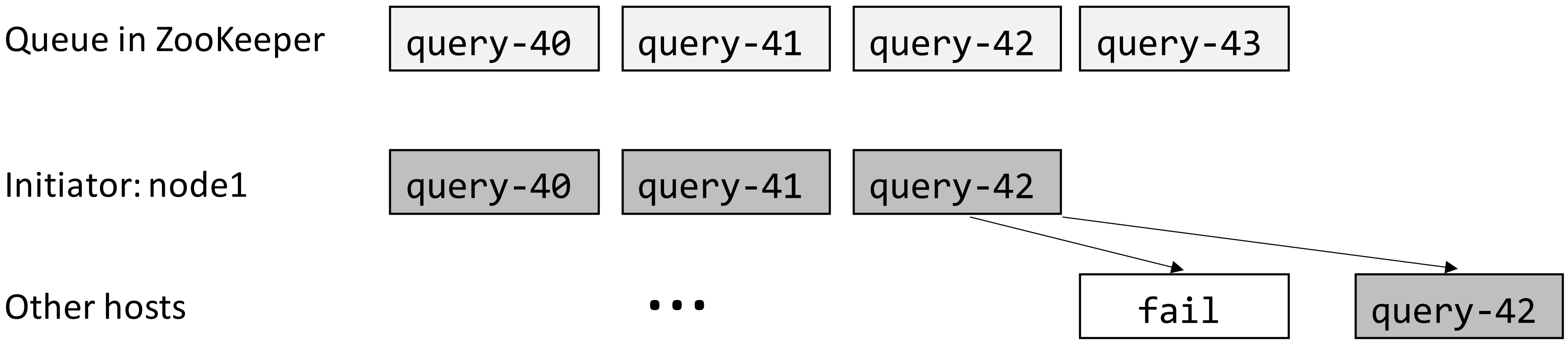
DDL queries

- Query acuires a sequential number in replication queue
- Initiator waits for previous queries in the queue to be executed
- Initiator tries to execute the query
- Other hosts execute the query



DDL queries

- Query acuires a sequential number in replication queue
- Initiator waits for previous queries in the queue to be executed
- Initiator tries to execute the query
- Other hosts execute the query
- Query will either fail on initiator or finish on all hosts (due to retries)



Cluster

```
node1 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'replica1')
node2 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'shard1', 'other_replica')
node3 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', '{replica}')
```

```
node1 :) SELECT cluster, shard_num, replica_num, host_name, host_address, port, is_local
FROM system.clusters WHERE cluster='r'


┌─cluster─┬─shard_num─┬─replica_num─┬─host_name─┬─host_address─┬─port─┬─is_local─┐
│ r       │         1 │           1 │ node3     │ 127.0.0.1    │ 9002 │        0 │
│ r       │         2 │           1 │ node2     │ 127.0.0.1    │ 9001 │        0 │
│ r       │         2 │           2 │ node1     │ 127.0.0.1    │ 9000 │        1 │
└─────────┴───────────┴─────────────┴───────────┴──────────────┴──────┴──────────┘
```

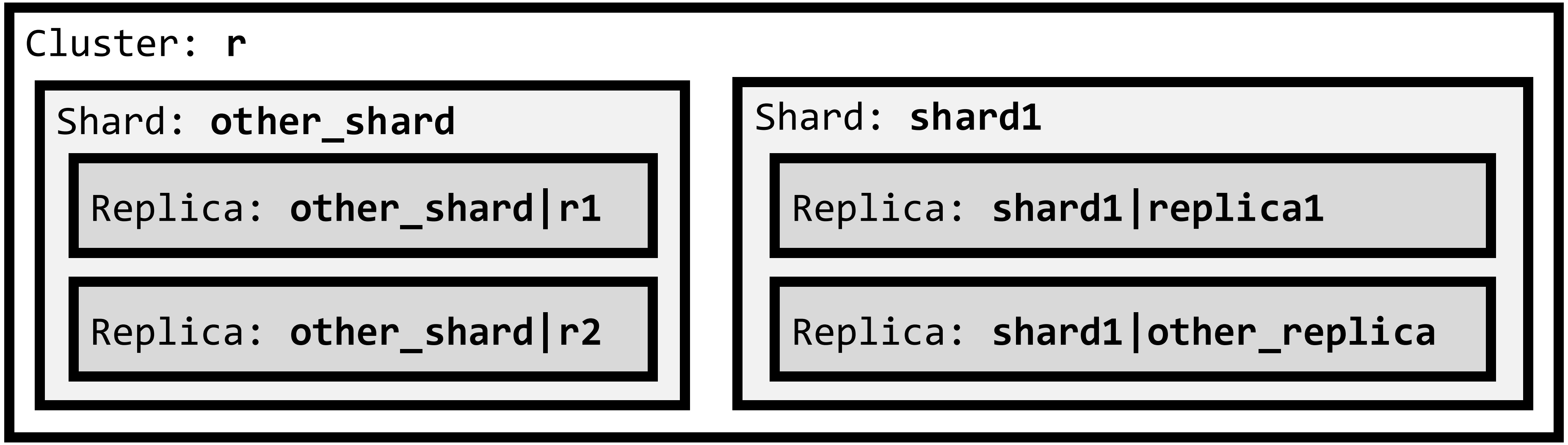
Cluster

```
node2 :) CREATE TABLE r.d (n UInt64) ENGINE=Distributed('r', 'r', 'rmt', n % 2)
node3 :) INSERT INTO r.d SELECT * FROM numbers(10)
node1 :) SELECT materialize(hostName()) AS host, groupArray(n) FROM r.d GROUP BY host

┌──host──┬─groupArray(n)─┐
│ node1 │ [1,3,5,7,9]   │
│ node3 │ [0,2,4,6,8]   │
└───────┴───────────────┘
```

How to add new replica

```
node4 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', 'r2')
```



Cluster: **r**

Shard: **other_shard**

Replica: **other_shard|r1**

Replica: **other_shard|r2**

Shard: **shard1**

Replica: **shard1|replica1**

Replica: **shard1|other_replica**

How to add new replica

```
node4 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', 'r2')

node1 :) SELECT cluster, shard_num, replica_num, host_name, host_address, port, is_local
  FROM system.clusters WHERE cluster='r'


┌─cluster─┬─shard_num─┬─replica_num─┬─host_name─┬─host_address─┬─port─┬─is_local─┐
│ r       │         1 │           1 │ node3     │ 127.0.0.1    │ 9002 │        0 │
│ r       │         1 │           2 │ node4     │ 127.0.0.1    │ 9003 │        0 │
│ r       │         2 │           1 │ node2     │ 127.0.0.1    │ 9001 │        0 │
│ r       │         2 │           2 │ node1     │ 127.0.0.1    │ 9000 │        1 │
└─────────┴───────────┴─────────────┴───────────┴──────────────┴──────┴──────────┘
```

How to add new replica

```
node4 :) CREATE DATABASE r ENGINE=Replicated('/some/path/r', 'other_shard', 'r2')
node2 :) SELECT materialize(hostName()) AS host, groupArray(n) FROM r.d GROUP BY host

┌─host──┬─groupArray(n)─┐
│ node2 │ [1,3,5,7,9]   │
│ node4 │ [0,2,4,6,8]   │
└───────┴───────────────┘
```

Recovery of staled replica

- Database compares metadata for each table
- Dictionaries and view-like tables are dropped and created again
- Non-replicated tables are moved to another database, new empty tables are created
- Names of replicated tables are updated

Recovery of staled replica

r.table1

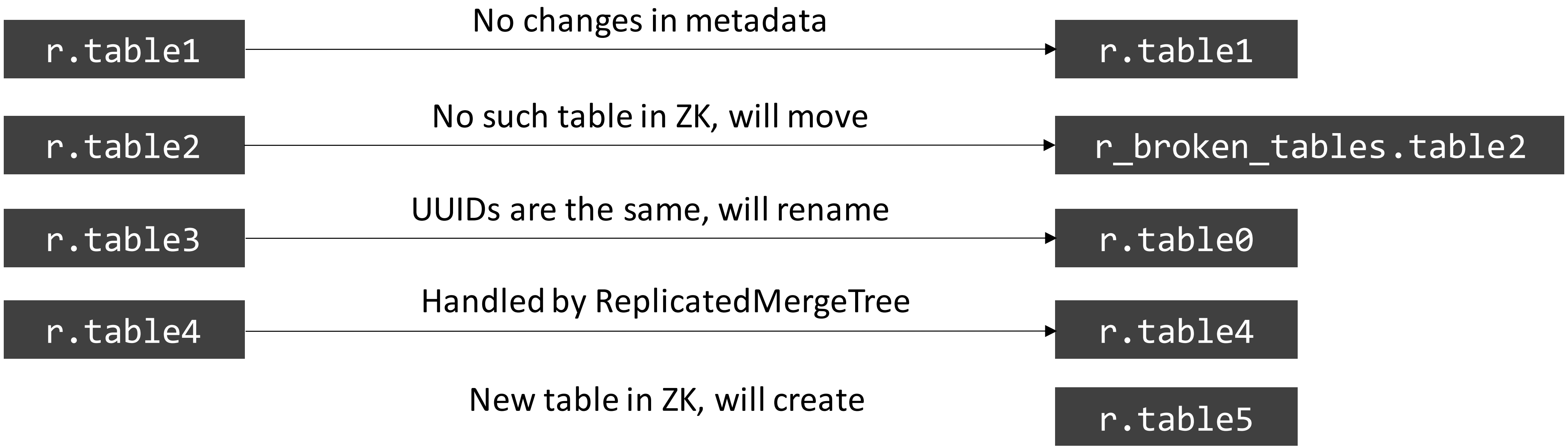
r.table2

r.table3

r.table4

```
:) DROP TABLE table2
:) RENAME TABLE table3 TO table0
:) ALTER TABLE table4 ADD COLUMN ...
:) CREATE TABLE table5 ...
```

Recovery of staled replica


No changes in metadata
r.table1
r.table1
No such table in ZK, will move
r.table2
r_broken_tables.table2
UUIDs are the same, will rename
r.table3
r.table0
Handled by ReplicatedMergeTree
r.table4
r.table4
New table in ZK, will create
r.table5

```
:) DROP TABLE table2
:) RENAME TABLE table3 TO table0
:) ALTER TABLE table4 ADD COLUMN ...
:) CREATE TABLE table5 ...
```

Questions?