Two Years with Vitess:

How JD runs the biggest Vitess cluster in the world

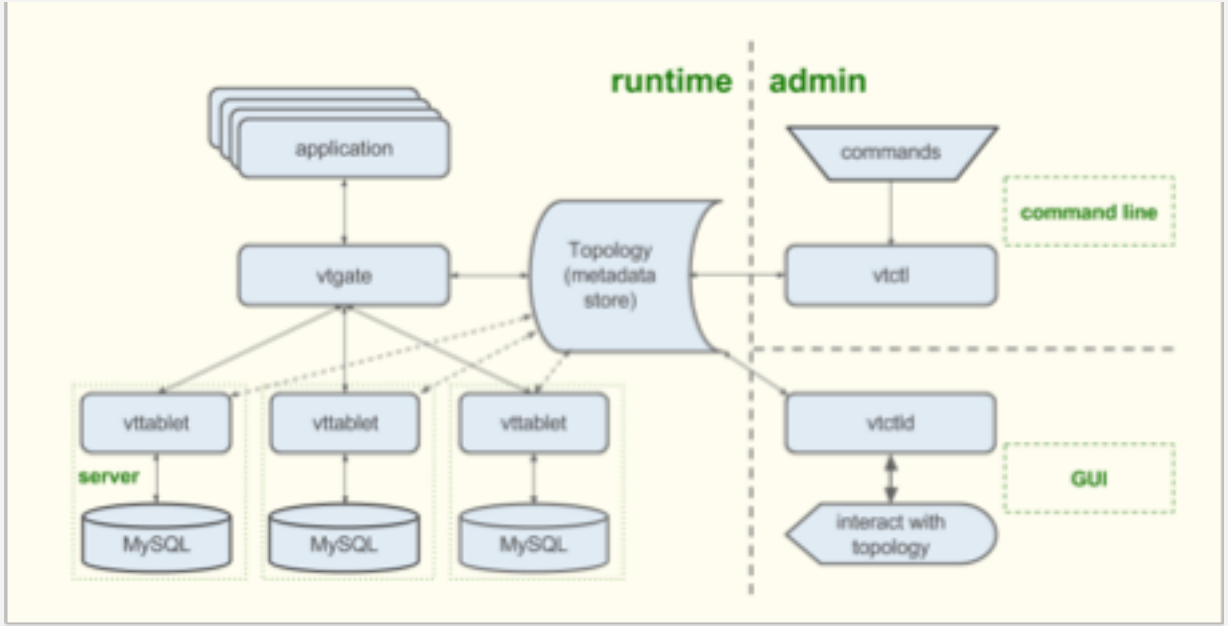
Speaker: Haihua Xu and Jinke Xie

CONTENTS

Why Vitess

What Is Vitess

Vitess is a database clustering system for horizontal scaling of MySQL



Advantages of Vitess

MySQL based

highly reliable

Resharding

scale data as needed

MySQL protocal support

easy to migrate from MySQL

Two Phase Commit

atomic commits for distributed txn

Stream query

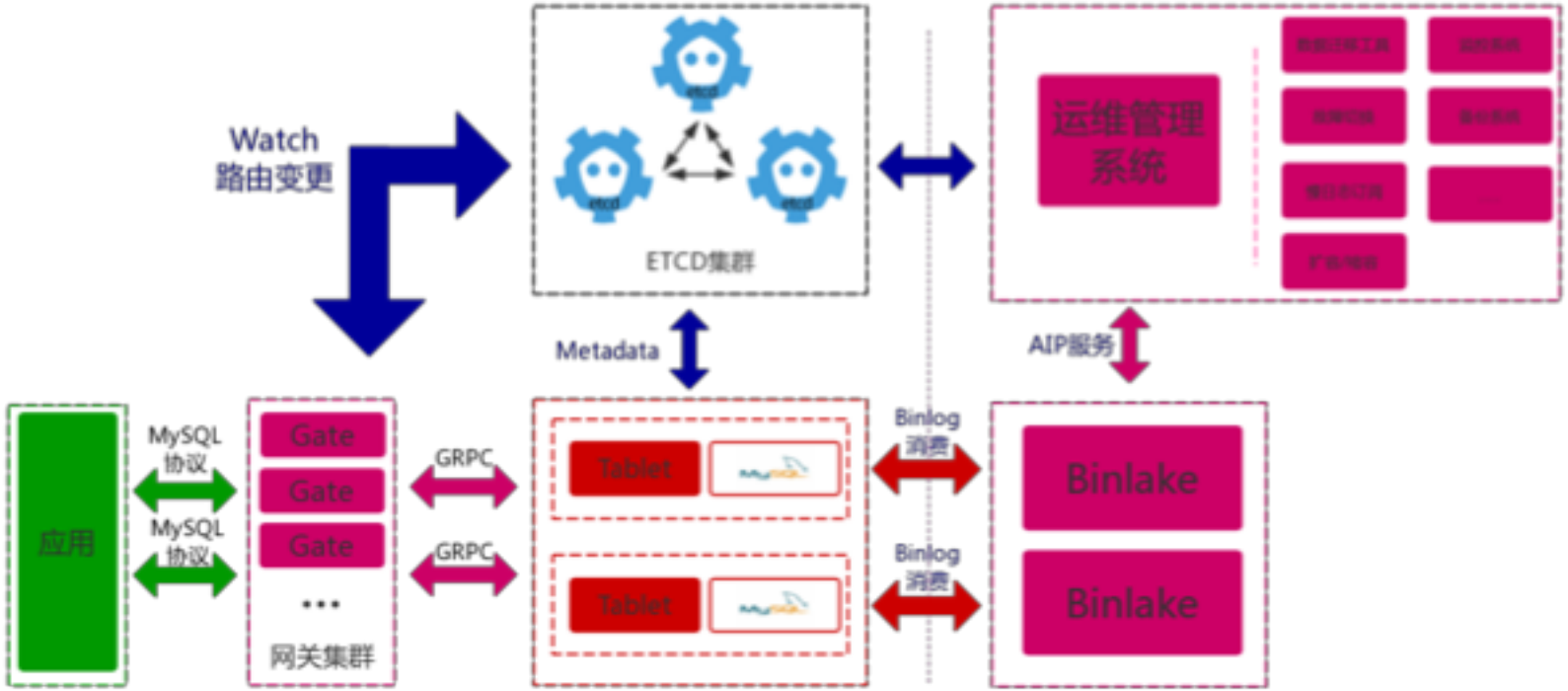
Stream data to big data platform

Global Secondary Index

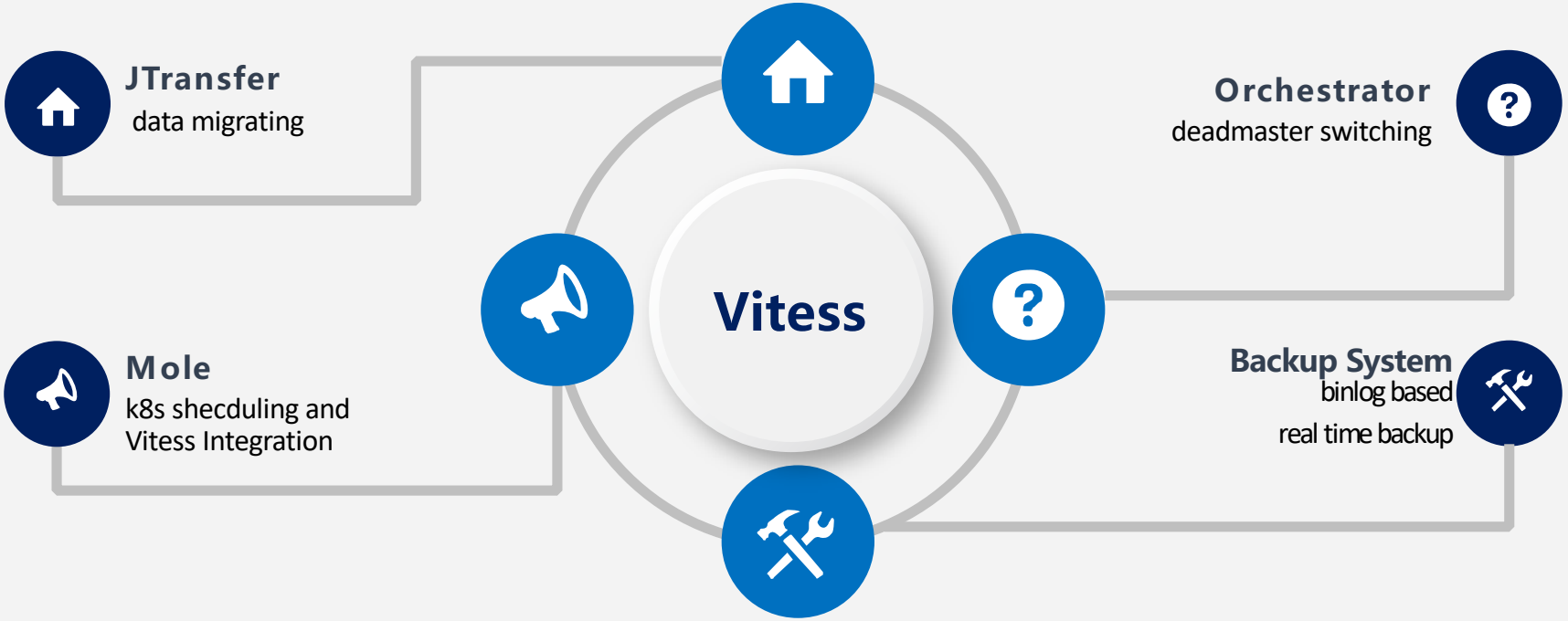
avoid reading amplification

How we run Vitess at JD

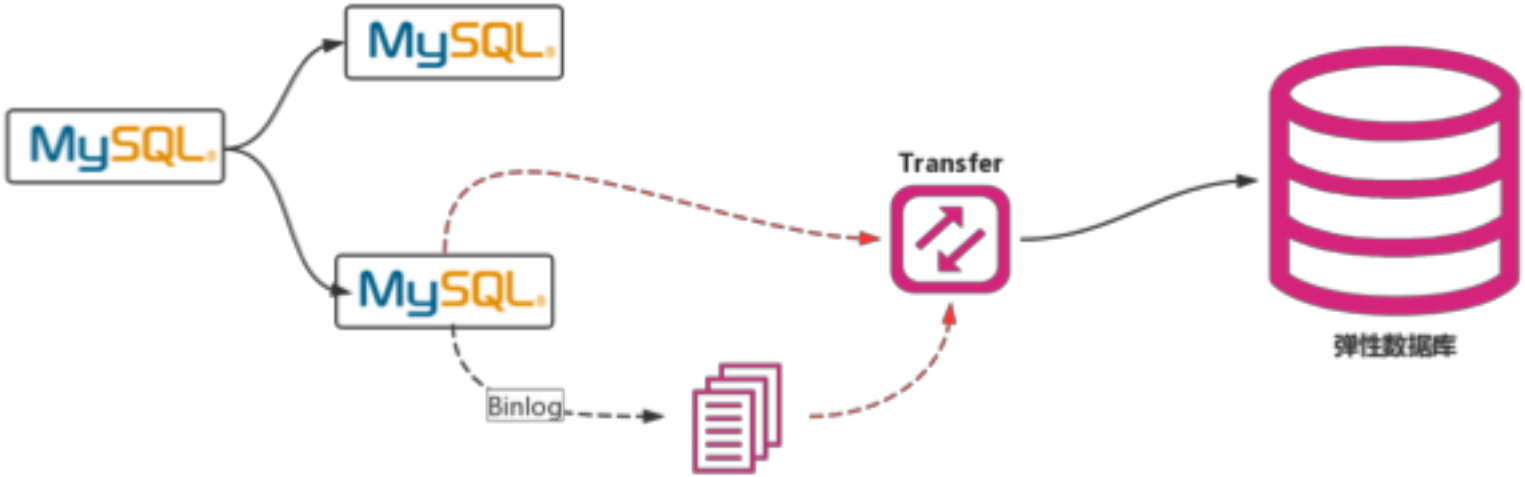
JED Architecture



Key systems

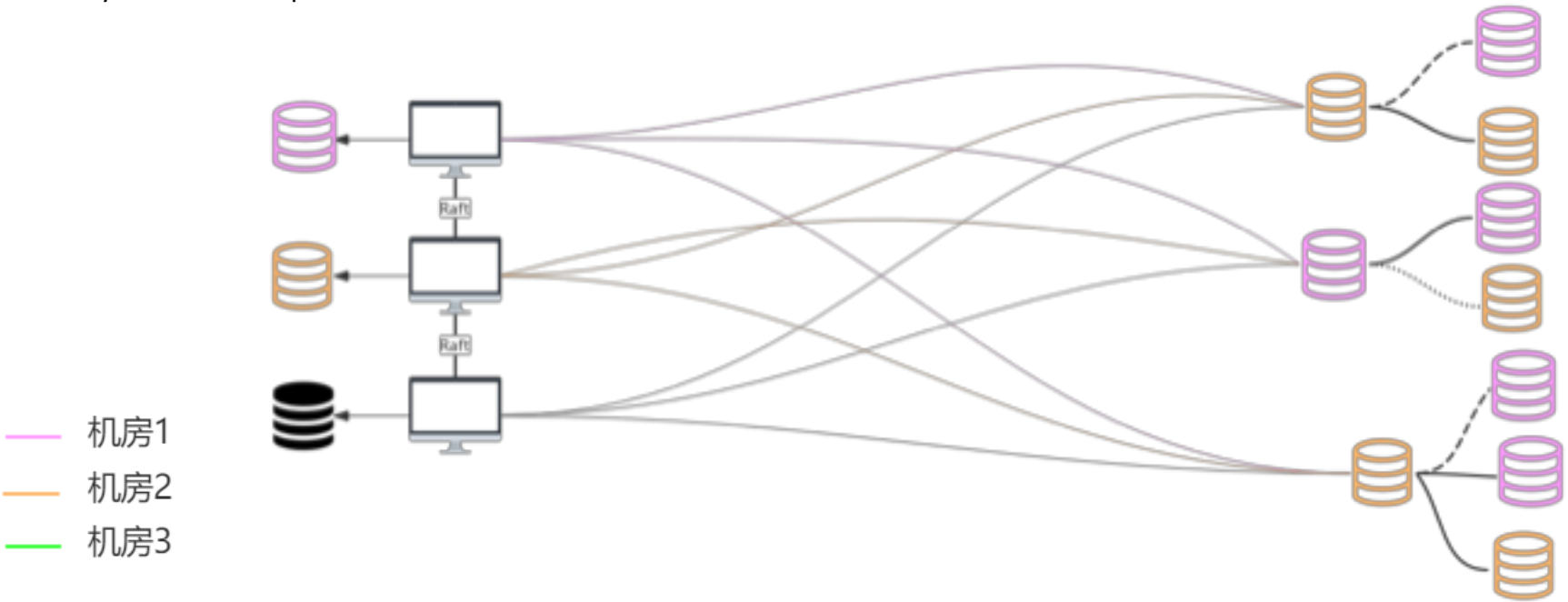

JTransfer
data migrating
Mole
k8s shecduling and
Vitess Integration
Vitess
Orchestrator
deadmaster switching
Backup System
binlog based
real time backup

How we migrate the app

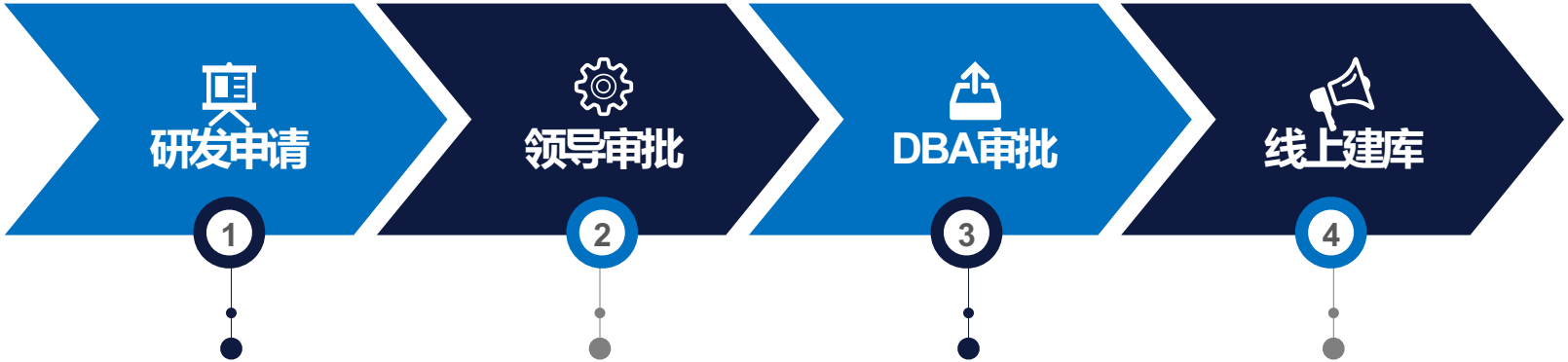


How we do failover

- One master with two or three replications
- One or two semi-synchronous replications
- One asynchronous replication



How we apply Mysql Instances



Apply

R&D colleagues apply to create a new keyspace, given shard number,resources volume

Approval

PM approval，Passing or reject

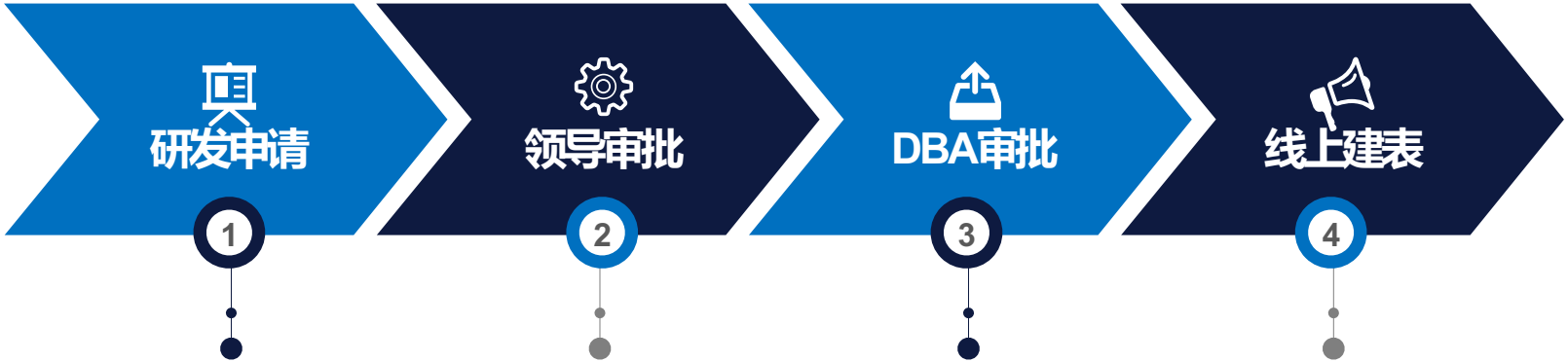
Approval

DBA approval, check whether the applyment is legal

Create

Create a new database online(Mole will schedule through k8s and create it)

How we Apply a MySQL table



Apply

R&D colleagues apply to create a new table, submit a statement in the system

Approval

PM approval，Passing or reject

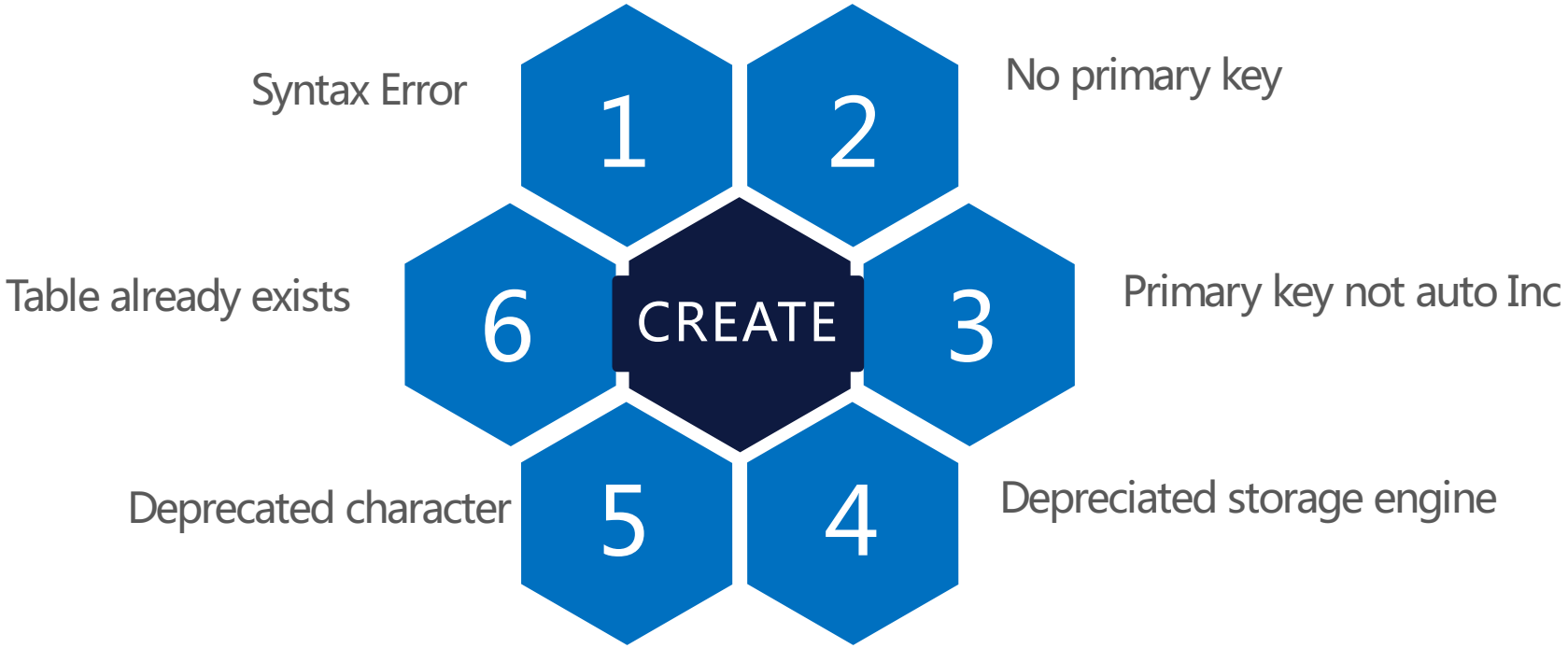
Approval

DBA approval, check whether the statement is in compliance with the specification

Create

Create a new table online(may be occur an Syntax error)

Issues may arise



Start new app vtcheck

Vtcheck will check if the table conforms to the database specification

```go
package main

import (
    "context"

    "fmt"

    "vitess.io/vitess/go/mysql"
    "vitess.io/vitess/go/vt/sqlparser"
)

func main() {
    stmt, err := sqlparser.Parse( sql: "create talble t(id int(20), name varchar(20))")
    if err != nil {
        return
    }
    ddl := stmt.(*sqlparser.DDL)
    name := ddl.Table
    cols := ddl.TableSpec.Columns
    idxs := ddl.TableSpec.Indexes

    conn, _ := mysql.Connect(context.Background(), params: nil)
    t, err := conn.ExecuteFetch(fmt.Sprintf( format: "show create table %v", name.Name.String()), maxrows: 100, wantfields: true)

    checkTable(cols, idxs, t)
}
```

Start new app DSLFormatter

Formatter convert a ast to ElasticSearch DSL

```
13  // Formatter translate sql to dsl.
14  func Formatter(buf *sqlparser.TrackedBuffer, node sqlparser.SQLNode) {
15      if buf.HasTrackError() {
16          return
17      }
18      switch node := node.(type) {
19      case *sqlparser.ComparisonExpr:
20          buf.InCmpExpr = true
21          defer func() {
22              buf.InCmpExpr = false
23          }()
24          switch node.Operator {
25          case sqlparser.EqualStr:
26              buf.Myprintf( format: `{"term" : {"%v" : %v}}`, node.Left, node.Right)
27          case sqlparser.NotEqualStr:
28              buf.Myprintf( format: `{"bool" : {"must_not" : [{"term" : {"%v" : %v}}]}}`, node.Left, node.Right)
29          case sqlparser.GreaterThanStr:
30              buf.Myprintf( format: `{"range" : {"%v" : {"gt" : %v}}}`, node.Left, node.Right)
31          case sqlparser.GreaterEqualStr:
32              buf.Myprintf( format: `{"range" : {"%v" : {"gte" : %v}}}`, node.Left, node.Right)
33          case sqlparser.LessThanStr:
34              buf.Myprintf( format: `{"range" : {"%v" : {"lt" : %v}}}`, node.Left, node.Right)
35          case sqlparser.LessEqualStr:
36              buf.Myprintf( format: `{"range" : {"%v" : {"lte" : %v}}}`, node.Left, node.Right)
37          case sqlparser.InStr:
38              buf.Myprintf( format: `{"terms" : {"%v" : [%v]}}`, node.Left, node.Right)
39          case sqlparser.NotInStr:
40              buf.Myprintf( format: `{"bool" : {"must_not" : {"terms" : {"%v" : [%v]}}}}`, node.Left, node.Right)
41          case sqlparser.LikeStr:
42              buf.Myprintf( format: `{"terms" : {"%v" : [%v]}}`, node.Left, node.Right)
43          case sqlparser.NotLikeStr:
44              buf.Myprintf( format: `{"bool" : {"must_not" : {"terms" : {"%v" : [%v]}}}}`, node.Left, node.Right)
45          default:
46              buf.SetTrackError(fmt.Errorf( format: "unsupported compare operator %v", node.Operator))
47              return
48          }
49          return
50      case sqlparser.BoolVal:
```

Start new app DSLFormatter

DSLFormatter convert a sql to ElasticSearch DSL

```
    {
        input:  `select * from t where id = 1`,
        output: `{"query" : {"term" : {"id" : 1}}  ,"from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id != 1`,
        output: `{"query" : {"bool" : {"must_not" : [{"term" : {"id" : 1}}]}}  ,"from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id > 1`,
        output: `{"query" : {"range" : {"id" : {"gt" : 1}}},  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id >= 1`,
        output: `{"query" : {"range" : {"id" : {"gte" : 1}}},  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id <= 1`,
        output: `{"query" : {"range" : {"id" : {"lte" : 1}}},  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id in (1,2,3)`,
        output: `{"query" : {"terms" : {"id" : [1, 2, 3]}},  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where name not in ('aa', 'bb', 'cc')`,
        output: `{"query" : {"bool" : {"must_not" : {"terms" : {"name" : ["aa", "bb", "cc"]}}}}  ,"from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id > 1 order by id desc`,
        output: `{"query" : {"range" : {"id" : {"gt" : 1}}},"sort":[{"id": "desc"}],  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select * from t where id > 1 order by id desc, age asc`,
        output: `{"query" : {"range" : {"id" : {"gt" : 1}}},"sort":[{"id": "desc"}, {"age": "asc"}],  "from" : 0, "size" : 10000}`,
    },
    {
        input:  `select city, city , avg(age) from space3 group by city`,
        output: `{"query" : {"bool" : {"must": [{"match_all" : {}}]}} , "aggs": {"city": {"terms": {"field": "city"},"aggs":{ "avg(age)":{"a
    },
```

Problems and Solutions

Challenges encoutered

01 **Vtgate cluster upgrading**

when we get a vtgate cluster with more than 1000 instances, how do we upgrade it?

02 Various demands

Complex Aggregate statement
Load statement
Prepare protocol

03 **Apps interaction**

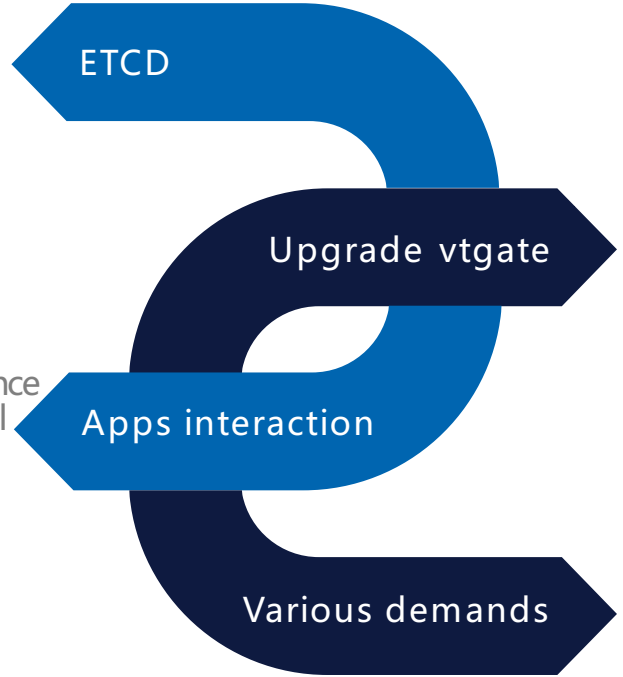
Apps wants Read and write separation
Apps wants to extract data from mysql to big data platform
0 level app wants the highest priority

04 Etcd problems

Some cell local etcd OOM weird
Local Vschema datasize growing as apps count increase

Solutions

Etcd client watcher leak fixed
Migrate local etcd big value to Redis



Gray-Release based on k8s scheduling

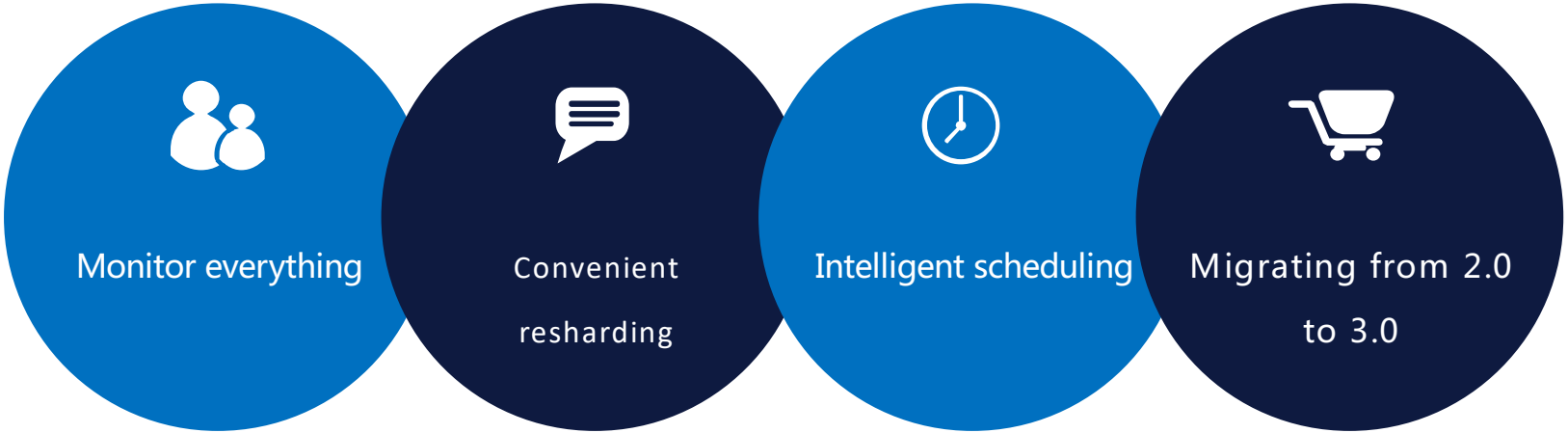
Different access role works in different instance
Exclusive Extract data Vtgate cluster each cell
Exclusive Cluster for 0 level app if necessary

Custom development base on Vitess

Future plan

Work plan for next year

Future Plan


Monitor everything
Convenient resharding
Intelligent scheduling
Migrating from 2.0 to 3.0

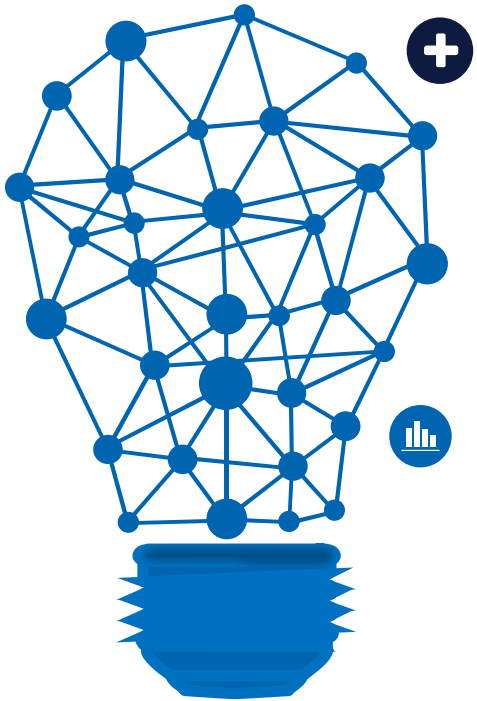
Future Plan

Monitor Everything

Including vtgate vttablet mysql etc
Build our own monitoring system if necessary

Convenient Reshard

Support a more interactive UI for DBA to reshard within minimum steps



Intelligent scheduling

Based on our Machine learning Algorithms
Dynamic expanse or reduce container resource

Migrating from 2.0 to 3.0

We have been merged 2.0 to 3.0 and test it for some time .
New apps apply to 3.0 already.
It is time to make a plan to migrate apps from 2.0 to 3.0 gradually.

THANK YOU