OrientDB®

Lukáš Dvořák, Jan Hermann, Vojtěch Kaňa, Martin Toufar

Contents

- overview
- features
- data representation
- clusters
- comparison with Neo4j

OrientDB

- open source document-graph database
- initial release in 2010
- written in Java
- cross-platform

Features

- reliable - full ACID transactions
- supports subset of SQL
- five different indexing algorithms
- up to 120,000 records per second
- multi-master replication
- easy installation
- quick import of relational databases
- schema-less, schema-full and schema-mixed mode

Multimodel database

- graph
- document
- key-value
- object

Graph database

- data is stored in edges and vertices
- two base classes
    - V - for vertices
    - E - for edges
- custom classes can be made by extending V or E

Example

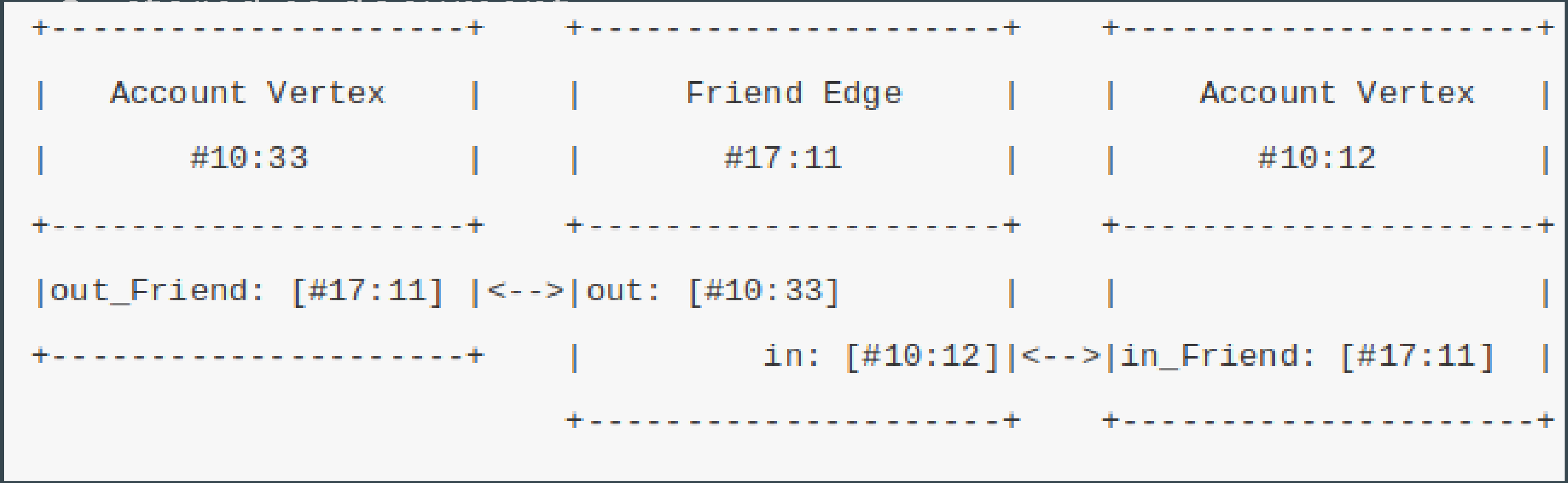
```
orientdb> CREATE CLASS Person EXTENDS V

orientdb> CREATE CLASS Student EXTENDS Person

orientdb> CREATE PROPERTY Student.name STRING
```

Edge

- connection between vertices
- bidirectional
- regular/lightweight

Regular edges

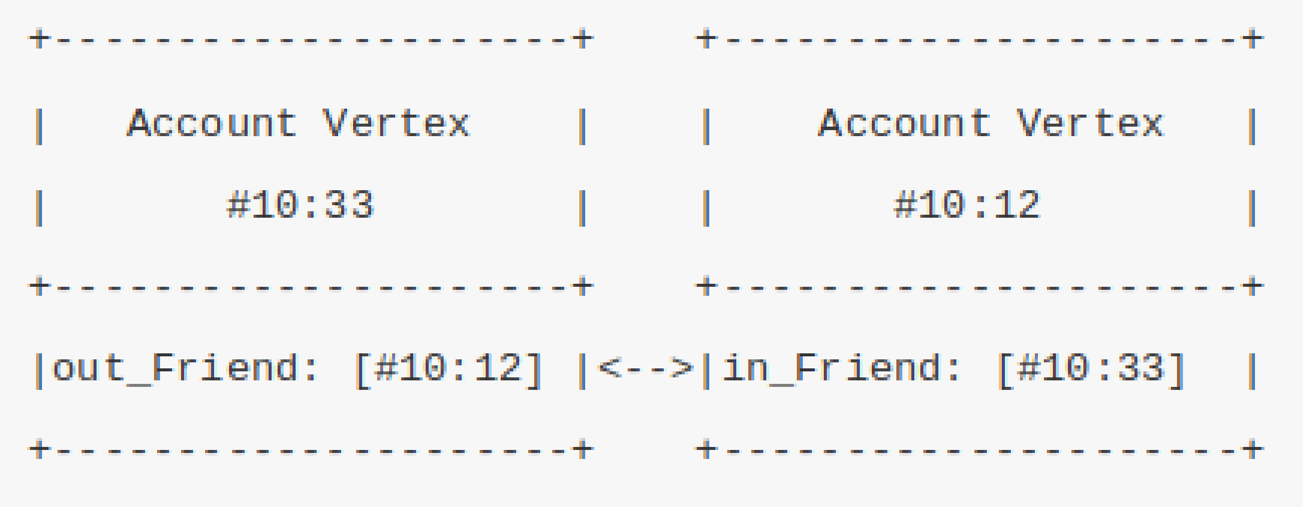
- extension of class E, can have properties
- stored as document



```
+---------------------+    +---------------------+    +---------------------+
|   Account Vertex    |    |     Friend Edge     |    |    Account Vertex   |
|       #10:33        |    |       #17:11        |    |       #10:12        |
+---------------------+    +---------------------+    +---------------------+
|out_Friend: [#17:11] |<-->|out: [#10:33]        |    |                     |
+---------------------+    |         in: [#10:12]|<-->|in_Friend: [#17:11]  |
                           +---------------------+    +---------------------+
```

Lightweight edges

- stored in vertices



```
+---------------------+    +---------------------+
|   Account Vertex    |    |    Account Vertex   |
|       #10:33        |    |       #10:12        |
+---------------------+    +---------------------+
|out_Friend: [#10:12] |<-->|in_Friend: [#10:33]  |
+---------------------+    +---------------------+
```

Lightweight edges

- advantages
    - faster creation and traversal
- disadvantages
    - cannot store properties
    - it’s difficult to query using SQL

Record ID

- #<cluster>:<position>
    - cluster identifier:
        - positive - persistent records
        - negative - temporary records
    - position - absolute position of record in cluster

Cluster

- collection of records of the same type
- similar to table in relational databases
- by default, OrientDB creates one cluster for each Class

Cluster selection

- round-robin
- balanced
- local
    - used in distributed mode
    - choses cluster stored on the current node

Advantages of using multiple clusters

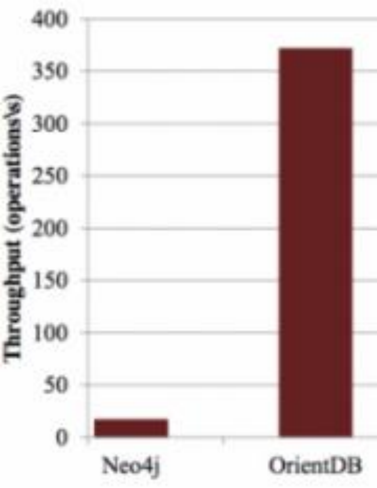
- optimization
- parallel queries
- sharding

Example

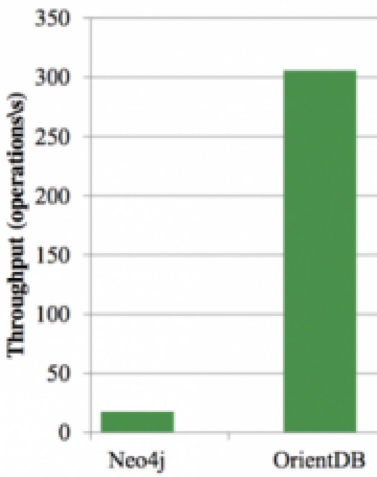
```
orientdb> CREATE  CLUSTER  <cluster-name>
```

```
orientdb> ALTER CLASS <class-name> ADDCLUSTER <cluster-name>
```

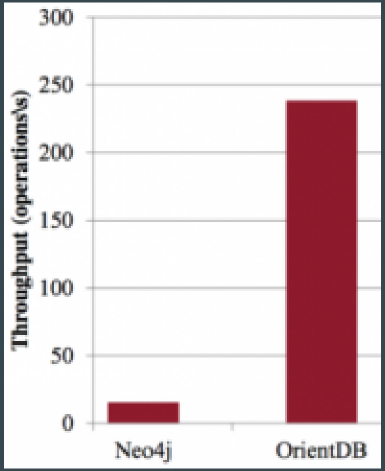
OrientDB vs Neo4j



50/50 read/update vertices



Insert new vertices



Read neighboring

Query Language

- 2 options in OrientDB - SQL-like or MATCH:
  - SELECT name, out('ACTS').title FROM Person WHERE name = 'Robin'
  - MATCH {class:Person, as:actor, where:(name:'Robin')) -ACTS_IN-> {as:movie} RETURN actor.name, movie.title
- Neo4j:
  - MATCH (actor:Person{name:'Robin'})-[:ACTS_IN]->(movie) RETURN actor.name, movie.title

DEMO

Sources

- OrientDB documentation
- https://en.wikipedia.org/wiki/OrientDB

Thanks for your attention.