Onglet 1

**Tout savoir sur OrientDB**

# Introduction:

OrientDB = application utile pour la gestion de BDD

4 types de gestion :

* **Key / Value**
* **Column-oriented**
* **Document**
* **Graph :** Composé de Vertex/Nodes et de Edges.

Vertex = Les vertex sont des entités contenant une information. Ils ont un identifiant unique (par exemple name = “Alice”) et des edges entrants ou sortants. Type de classe : “V”

Edge = Les edges sont des liens entre 2 vertex, ont un identifiant unique et un label permettant de savoir le lien entre les vertex (entrant ou sortant). Type de classe : “E”

## Installation :

<https://orientdb.com/docs/last/fiveminute/java.html>

## Tutoriel :

<https://orientdb.org/docs/2.0/orientdb.wiki/Graph-Database-Tinkerpop.html>

# Généralité :

ATTENTION : Les commandes suivantes fonctionnent toutes sur console.bat mais pas toutes sur le localhost.

Pour lancer la BDD sur le localhost (dans orientDB/bin) : server.bat

Pour lancer la console (dans orientDB/bin) : console.bat

Pour lier le console.bat à une BDD : CONNECT remote:localhost root *my\_root\_password* puis CONNECT remote:localhost/*NameOfBDD* admin admin ou en créant directement la BDD sur le localhost

<https://orientdb.org/docs/2.0/orientdb.wiki/Console-Commands.html> pour toutes les commandes de console.bat

Classe : Permet de définir un groupe auquel les éléments d’une BDD (records) peuvent appartenir. create class *NameOfClass* pour créer une classe

Propriété : Lié à une classe (exemple Student.name => name est une propriété de student. create property *NameOfClass.NameOfProperty type* pour créer une propriété.

Toutes les informations d’une classe peuvent être obtenu par

info class *NameOfClass* .

On peut définir des obligations (alter) sur des propriétés (exemple : alter property Student.name min 3 Permet de définir que Student.name doit avoir plus de 3 caractères).

Record : Plus petite unité de stockage contenu dans une BDD de type document dans OrientDB (comme un .json). Possède un ID unique (RID) de la forme #<cluster-id>:<cluster-position> qui donne la position physique du record la BDD. Pour obtenir un record précis : load record *RID*

Cluster : Contient tous les records d’une classe. Par défaut, un cluster par classe. Possède le même identifiant que la classe auquel il est lié. Possibilité de créer plusieurs cluster pour une même classe pour affiner la gestion de la BDD (exemple : Classe = Customer, Clusters = China\_Customer + USA\_Customer)

Pour créer d’autres cluster pour une même classe : ALTER CLASS Customer ADDCLUSTER USA\_Customers

2 types de Clusters : Physique (ou local) et Mémoire. Ce dernier disparaît si le serveur de la BDD se ferme.

Pour voir tous les records d’une classe ou d’un cluster : browse class/cluster NameOfClass/Cluster

Pour chercher une ou plusieur information d’une classe : select from *NameOfClasse + options*

Options : where, like, order by, asc, dsc, group by etc…

## Relationship:

Dans OrientDB, chaque record est lié à une adresse qui correspond à son emplacement par rapport aux autres données stockées. C’est la Foreign Key, ou Primary Key. On peut l’obtenir via : SELECT *B.location* FROM *NameOfClass* A, Address B WHERE *A.parameters* = x AND A.address = B.id

Il n’existe pas de clé multiple, que des clés uniques. On peut quand même faire des recherches de clés multiples : SELECT B.location FROM *NameOfClass* A, Address B WHERE A*.parameters* = x AND B.customer = A.id => Ici on cherche les clés de tous les records d’un objet A de classe NameOfClass et de paramètre x (par exemple A.name = “Lucas”)

# Base Graphe:

Classe des vertex = V automatiquement créer par Orient, il faut donc y insérer de nouveau vertex : create vertex V set *parameters* = x

Classe des edges = E : create edge from *RID1* to *RID2*

On doit utiliser les RID des vertex pour les lier avec un edge OU en les cherchant directement : create edge *NameOfEdge* from (select from *NameOfVertex* where parameters = x) to (select from *NameOfVertex* where parameters = y)

On peut, pour plus de lisibilité de la BDD, créer des sous-classes de vertex via : create class *NameOfVertex* extends V

On pourra ainsi créer des vertex du type: create vertex *NameOfVertex* set parameters = x

On peut avoir accès au voisin d’un vertex via les options in(), out() ou both() : select in() from *NameOfVertex* where parameters = x

Mais cela ne donne uniquement les RID des voisins. Pour obtenir toutes les informations, on utilise expand( in/out/both() ) : select expand ( in() ) from *NameOfVertex* where parameters = x

On peut aussi faire des recherches plus poussées en passant par le lien d’un vertex pour avoir des informations sur ses voisins. Par exemple : select expand( both('*NameOfEdge1*').out('*NameOfEdge2*') ) from *NameOfVertex* where *parameters* = x

LightEdges : Edges sans propriétés particulières. Attention puisque pas de propriété, on ne peut pas faire de recherches sur eux

On peut créer des propriétés sur les edges pour forcer les in et out : create property NameOfEdge.in/out LINK NameOfVertex

Pour forcer OrientDB à ne pas utiliser les lightweight edges, on utilise l’option MANDATORY : alter property NameOfEdge.in/out MANDATORY=true;

On peut forcer à ce qu’il n’y ai qu’un nombre de edge fixe entre 2 classe de vertex en utilisant des index sur le edge en question: create index *NameOfIndex* on *NameofEdge*(out,in) *options.*

L’option unique par exemple limite ce nombre à 1.

ATTENTION: Si des edges sont déjà créés et que l’on ajoute un index sur ce edge, il faudra recréer les edges pour qu’il prenne en compte l’index.

Orient DB Java API

## 1.1) Forme Global

| **ORecord** | Regroupe tous les records de la BDD |
| --- | --- |
| **OBlob** | Record binaire |
| **OElement** | Record classique, de type document |

## 1.2) Création BDD

Via la commande :

OrientDB orientDB = new OrientDB("embedded:/tmp/",OrientDBConfig.defaultConfig());

orientDB.create("test",ODatabaseType.PLOCAL);

On crée une nouvelle database sur le PLOCAL, de nom “test”. Elle crée automatiquement les user “admin”, “reader” et “writer”

Pour manipuler la database, on écrira le code dans la zone suivante:

try(ODatabaseSession db = orientDB.open("test","admin","admin");) {

// Enter your code here...

}

Attention ! Plutôt que d’utiliser une seule session à chaque connexion, il est préférable de créer une “pool” de session, et on en choisira une à chaque connexion.

Implémentation :

OrientDBConfigBuilder poolCfg = OrientDBConfig.builder();

poolCfg.addConfig(OGlobalConfiguration.DB\_POOL\_MIN, 5);

poolCfg.addConfig(OGlobalConfiguration.DB\_POOL\_MAX, 10);

ODatabasePool pool = new ODatabasePool(orientDB,"test","admin","admin", poolCfg.build());

// OPEN DATABASE

try (ODatabaseSession db = pool.acquire()) {

// YOUR CODE

...

}

pool.close();

orientDB.close();

# Utilisation BDD

Transactionnal / Non-transactionnal mode : 2 types de communications différentes, l’une nécessitant un couple db.begin()/db.commit() ainsi que de element.save() alors que l’autre n’utilise que element.save().