

Projet NoSQL – OrientDB



Table des matières

1. Choix de la technologie.....	3
2. Manuel d'installation	4
2.1. Installation avec Docker.....	4
2.2. Lancement du Studio	4
2.3. Interface.....	5
2.3.1. Onglet Schema	5
2.3.2. Onglet BD	6
2.3.3. Onglet Browse.....	6
2.3.4. Onglet Graph.....	7
3. Tutoriel : création et importation d'une BD	8
3.1. Creation d'une nouvelle BD	8
3.2. Création de classes, vertexes et d'edges	9
3.3. Importation BD existantes	13
4. Fonctionnalités avancées.....	14
4.1. Graph Editor	14
4.2. Browse Editor.....	17
5. Conclusion.....	18

1. Choix de la technologie

OrientDB est un système de gestion de base de données NoSQL, multi-modèle, opensource. Il combine les fonctionnalités d'un SGBD graphique et documentaire en un seul produit. Il est écrit en Java et peut être utilisé sur divers systèmes d'exploitation, y compris Windows, Linux et macOS. OrientDB offre des performances élevées, une scalabilité horizontale, une sécurité intégrée et une facilité d'utilisation pour les développeurs et les administrateurs de bases de données.

OrientDB est le premier SGBD NoSQL Open Source multi-modèles qui combine la puissance des graphes et la flexibilité des documents dans une base de données opérationnelle évolutive et performante.

2. Manuel d'installation

2.1. Installation avec Docker

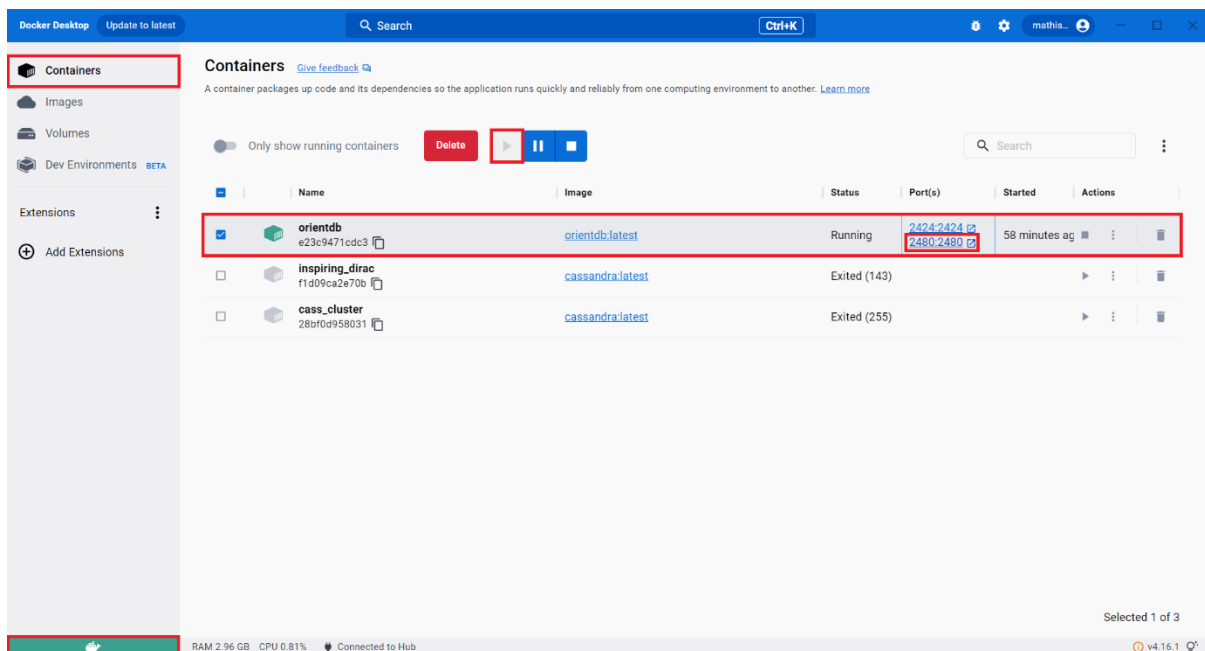
Si Docker est installé sur votre ordinateur, c'est le moyen le plus simple d'exécuter OrientDB. A partir de la ligne de commande tapez :

```
$ docker run -d --name orientdb -p 2424:2424 -p 2480:2480 -e ORIENTDB_ROOT_PASSWORD=root orientdb:latest
```

Au lieu de **root** entrer le mot de passe que vous souhaitez

2.2. Lancement du Studio

Ouvrez Docker, assurez-vous d'être bien connecté (voir icône en vert). Allez dans l'onglet Container. Vérifiez si le container orientdb est bien lancé puis cliquez sur le lien localhost:80



Une fenêtre s'ouvrira dans votre navigateur avec l'interface OrientDB. Connectez-vous alors avec votre identifiant « root » par défaut et le mot de passe que vous avez renseigné lors de l'installation.

2.3. Interface

2.3.1. Onglet Schema

Dans l'onglet Schema, se trouve toutes les tables de votre BD dans Vextex Classes

The screenshot shows the OrientDB Schema Manager interface. The 'SCHEMA' tab is selected. The 'Vertex Classes' section is highlighted with a red box. It contains a table with the following data:

Name	Color	SuperClasses	Alias	Abstract	Clusters	Default Cluster	Cluster Selection	Records	Actions
ArchaeologicalSites		V. Attractions		<input type="checkbox"/>	[113, 114, 115, 116, 117, 118, 1...	113	round-robin	55	RENAME QUERY ALL + NEW RECORD DROP
Attractions		V. Locations		<input type="checkbox"/>	[81, 82, 83, 84, 85, 86, 87, 88]	81	round-robin	436	RENAME QUERY ALL + NEW RECORD DROP
Castles		V. Attractions		<input type="checkbox"/>	[97, 98, 99, 100, 101, 102, 103, ...]	97	round-robin	127	RENAME QUERY ALL + NEW RECORD DROP
Countries		V		<input type="checkbox"/>	[33, 34, 35, 36, 37, 38, 39, 40]	33	round-robin	249	RENAME QUERY ALL + NEW RECORD DROP
Customers		V		<input type="checkbox"/>	[121, 122, 123, 124, 125, 126, ...]	121	round-robin	400	RENAME QUERY ALL + NEW RECORD DROP

Below the table, there are pagination controls showing 1, 2, 3. The 'Edge Classes' section is also visible below, showing a table with one row for 'E'.

En dessous se trouve les différentes relations entre chaque table dans Edge Classes

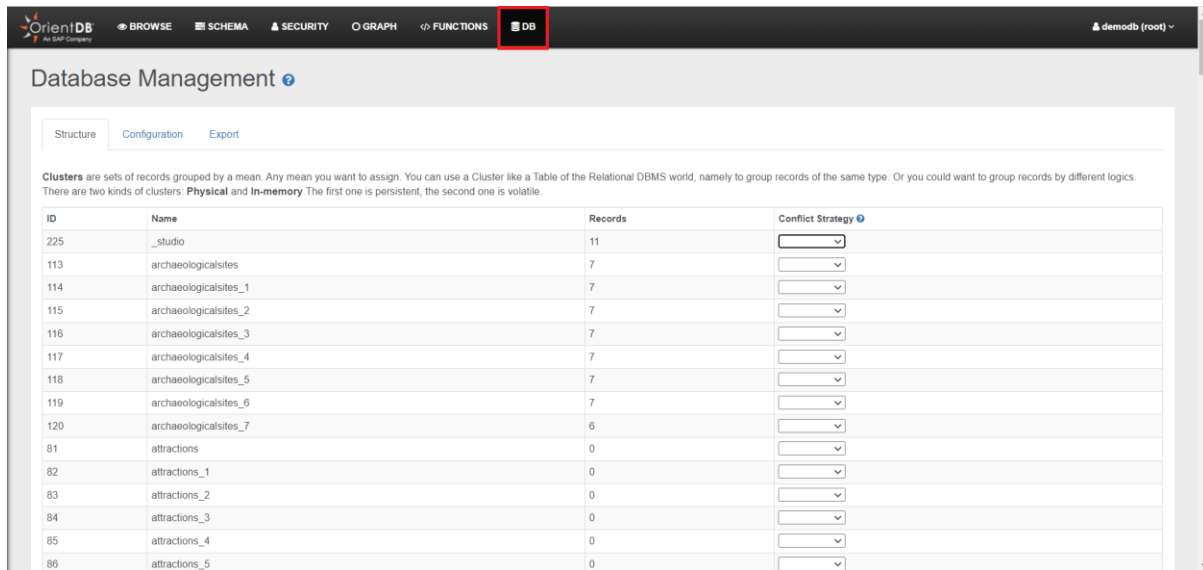
The screenshot shows the OrientDB Schema Manager interface, specifically the 'Edge Classes' section. The 'Edge Classes' table is highlighted with a red box. It contains the following data:

Name	Color	SuperClasses	Alias	Abstract	Clusters	Default Cluster	Cluster Selection	Records	Actions
E				<input type="checkbox"/>	[17, 18, 19, 20, 21, 22, 23, 24]	17	round-robin	14,872	RENAME QUERY ALL + NEW RECORD DROP
HasCustomer		E		<input type="checkbox"/>	[193, 194, 195, 196, 197, 198, ...]	193	round-robin	812	RENAME QUERY ALL + NEW RECORD DROP
HasEaten		E, HasUsedService		<input type="checkbox"/>	[169, 170, 171, 172, 173, 174, ...]	169	round-robin	2,479	RENAME QUERY ALL + NEW RECORD DROP
HasFriend		E		<input type="checkbox"/>	[217, 218, 219, 220, 221, 222, ...]	217	round-robin	1,617	RENAME QUERY ALL + NEW RECORD DROP
HasProfile		E		<input type="checkbox"/>	[185, 186, 187, 188, 189, 190, ...]	185	round-robin	400	RENAME QUERY ALL + NEW RECORD DROP

Below the table, there are pagination controls showing 1, 2, 3. The 'Generic Classes' section is also visible below, showing a table with two rows: 'DBInfo' and 'OSecurityPolicy'.

2.3.2. Onglet BD

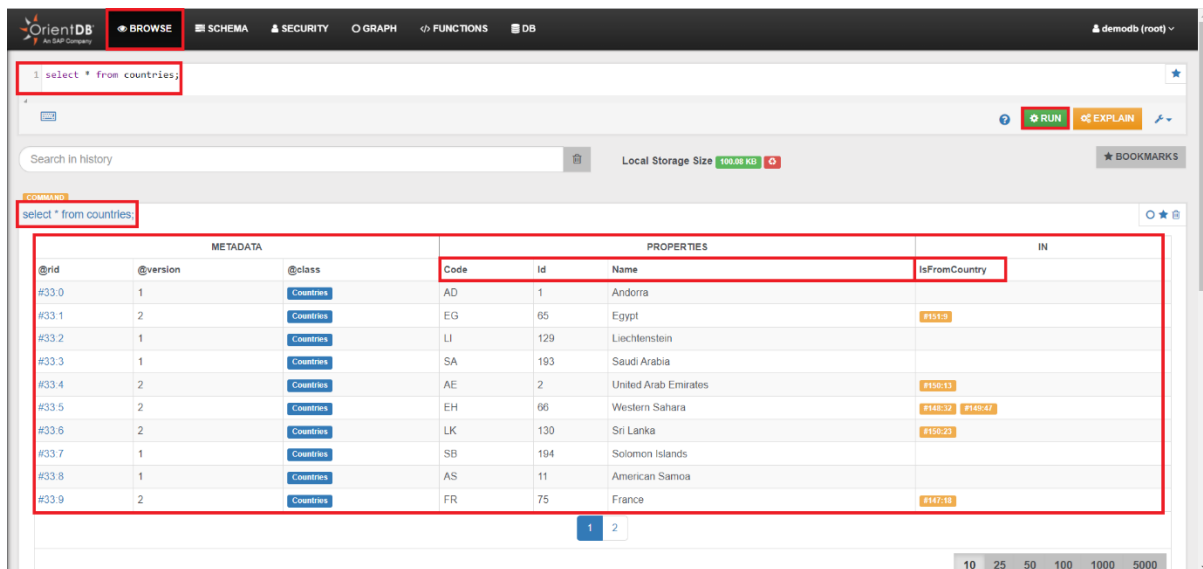
Dans l'onglet BD se trouve plus d'informations sur chaque table comme leur nombre d'enregistrement



ID	Name	Records	Conflict Strategy
225	_studio	11	
113	archaeologicalsites	7	
114	archaeologicalsites_1	7	
115	archaeologicalsites_2	7	
116	archaeologicalsites_3	7	
117	archaeologicalsites_4	7	
118	archaeologicalsites_5	7	
119	archaeologicalsites_6	7	
120	archaeologicalsites_7	6	
81	attractions	0	
82	attractions_1	0	
83	attractions_2	0	
84	attractions_3	0	
85	attractions_4	0	
86	attractions_5	0	

2.3.3. Onglet Browse

Entrez votre requête, ici on cherche à afficher tous les enregistrements de la table countries. Cliquez sur Run, pour valider la requête. En dessous s'affiche le résultat de la requête avec chaque ligne de la table ainsi que les relations « isFromCountry » associé a chaque pays. A noter que l'on peut accéder aux requêtes précédentes en scrollant.



METADATA			PROPERTIES			IN
@rid	@version	@class	Code	Id	Name	isFromCountry
#33.0	1	Countries	AD	1	Andorra	
#33.1	2	Countries	EG	65	Egypt	#131.3
#33.2	1	Countries	LI	129	Liechtenstein	
#33.3	1	Countries	SA	193	Saudi Arabia	
#33.4	2	Countries	AE	2	United Arab Emirates	#136.13
#33.5	2	Countries	EH	66	Western Sahara	#148.32 #149.47
#33.6	2	Countries	LK	130	Sri Lanka	#152.23
#33.7	1	Countries	SB	194	Solomon Islands	
#33.8	1	Countries	AS	11	American Samoa	
#33.9	2	Countries	FR	75	France	#127.61

2.3.4. Onglet Graph

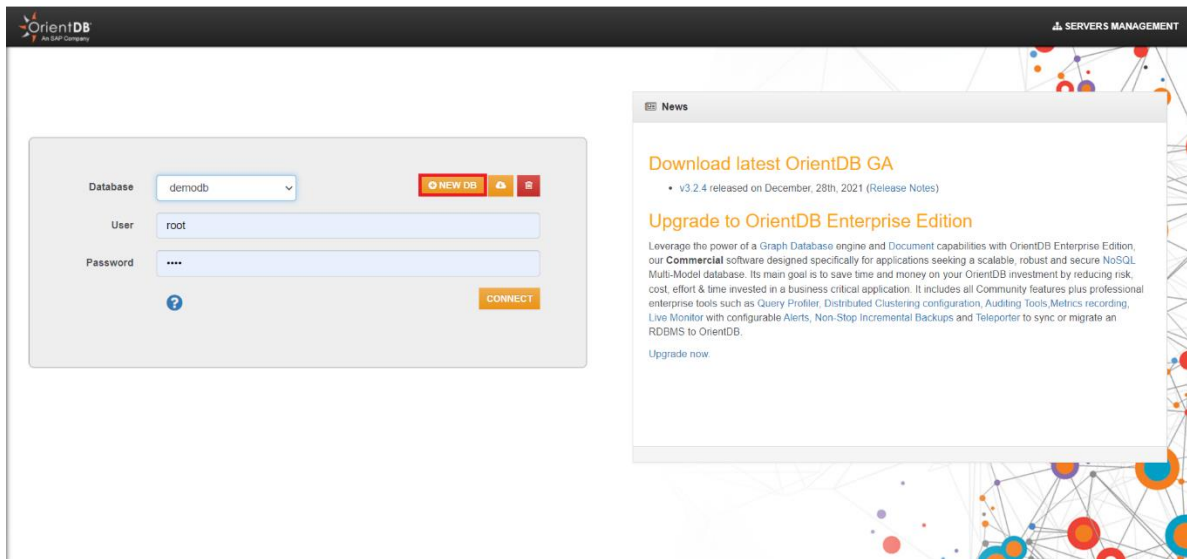
Même requête en mode graph, on peut voir chaque pays existant sous forme de cellule. En sélectionnant un pays on pourra avoir accès à l'ensemble des informations le concernant sur le volet gauche



3. Tutoriel : création et importation d'une BD

3.1. Création d'une nouvelle BD

Cliquez sur le bouton NEW DB



Entrez le nom de votre nouvelle base de données, identifiez-vous, cliquez sur CREATE DATABASE

New Database

Name

new_database

Server User

root

Server Password

....

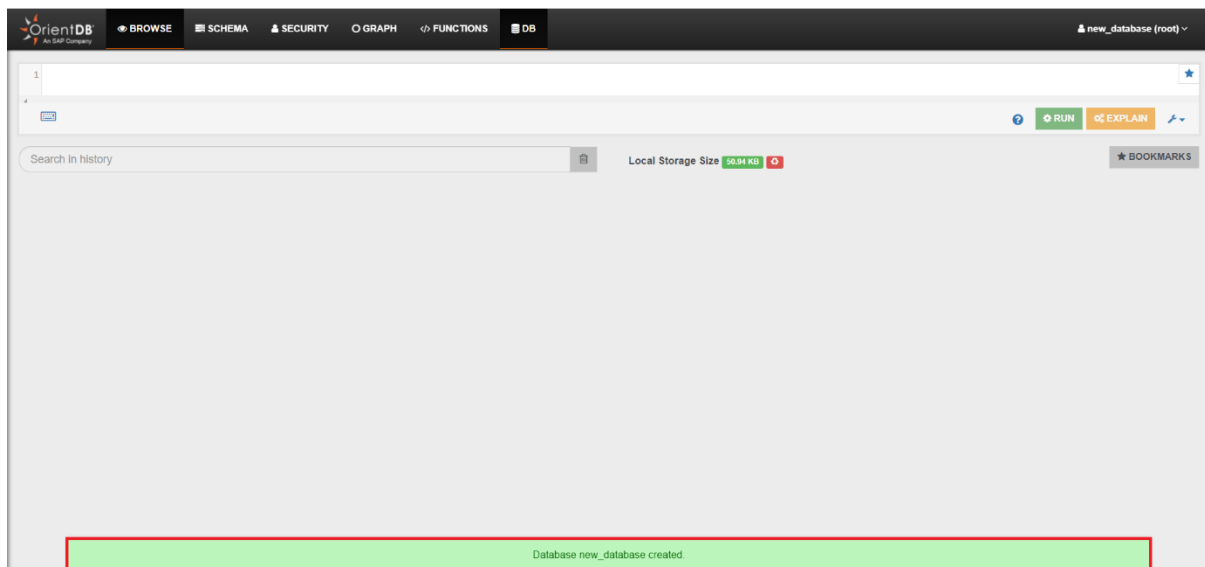
☐ Create Admin user

☐ Advanced Options

CLOSE

CREATE DATABASE

Nouvelle base de données créée ! Ici elle est encore vide.

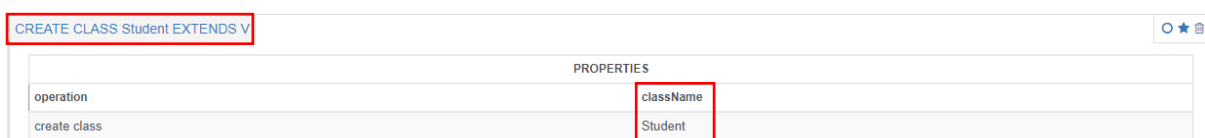


3.2. Création de classes, vertexes et d'edges

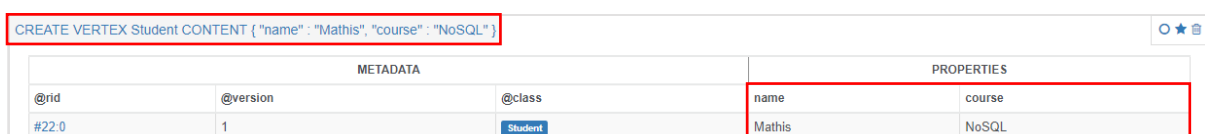
OrientDB est une base de données orientée graphes qui utilise des concepts clés de la théorie des graphes. Voici des définitions des termes clés de la base de données OrientDB :

- **Classe** : une classe est un conteneur de données dans une base de données OrientDB. Elle peut être comparée à une table dans une base de données relationnelle. Une classe contient un ensemble d'attributs qui décrivent les propriétés des objets qui y sont stockés, ainsi que des méthodes qui peuvent être utilisées pour manipuler ces objets.
- **Vertex** : un vertex (ou nœud) est un objet dans une base de données OrientDB qui représente une entité. Les vertices sont les éléments de base dans un graphe et peuvent être comparés à des enregistrements dans une table de base de données relationnelle. Les vertices peuvent avoir des attributs qui décrivent leurs propriétés et peuvent être connectés les uns aux autres via des edges.
- **Edge** : un edge (ou arête) est une relation entre deux vertices dans un graphe. Les edges représentent les connexions entre les entités et peuvent être comparés aux clés étrangères dans une table de base de données relationnelle. Les edges ont un type, qui peut être utilisé pour décrire la nature de la relation entre les vertices, ainsi que des attributs qui décrivent des propriétés de cette relation.

Création de la classe **Student**



Insertion dans Student



Création de la classe **Professor**

CREATE CLASS Professor EXTENDS V

PROPERTIES	
operation	className
create class	Professor

Insertion dans Professor

CREATE VERTEX Professor CONTENT ("name": "M.Boubchir", "course": "NoSQL")

METADATA			PROPERTIES	
@rid	@version	@class	name	course
#30:0	1	Professor	M.Boubchir	NoSQL

Création de la relation **Teach**

CREATE CLASS Teaches EXTENDS E

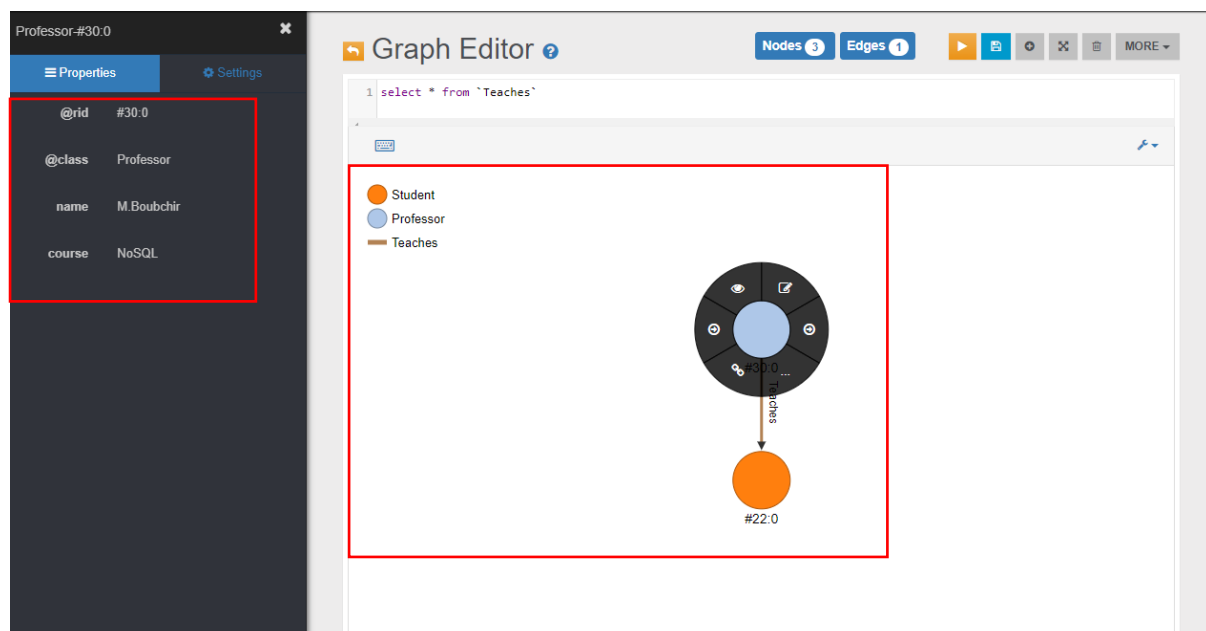
PROPERTIES	
operation	className
create class	Teaches

Assignment Teach de Professor vers Student

CREATE EDGE Teaches FROM (SELECT FROM Professor) TO (SELECT FROM Student)

METADATA			PROPERTIES	
@rid	@version	@class	out	in
#34:0	1	Teaches	#30:0	#22:0

Affichage graphique de l'ensemble des relation teaches existantes. Ici on peut voir que le professeur « M.Boubchir » enseigne à l'étudiant « Mathis »



Même requête, mais après avoir assigné de nouveaux étudiants à M.Boubchir

Professor-#30:0

Properties

- @rid: #30:0
- @class: Professor
- name: M.Boubchir
- course: NoSQL

Graph Editor

```
1 match(class:Professor, as:p, where: (name="M.Boubchir"))--(as: n)
2 return $p$teachements
```

Nodes 4 Edges 3

Legend: Professor (blue circle), Student (orange circle), Teaches (brown line)

Supprimer un edge

DELETE EDGE #37:0

PROPERTIES

count
1

Legend: Professor (blue circle), Student (orange circle), Teaches (brown line)

Supprimer un vertice

DELETE VERTEX #24:0

PROPERTIES

count
1

● Student

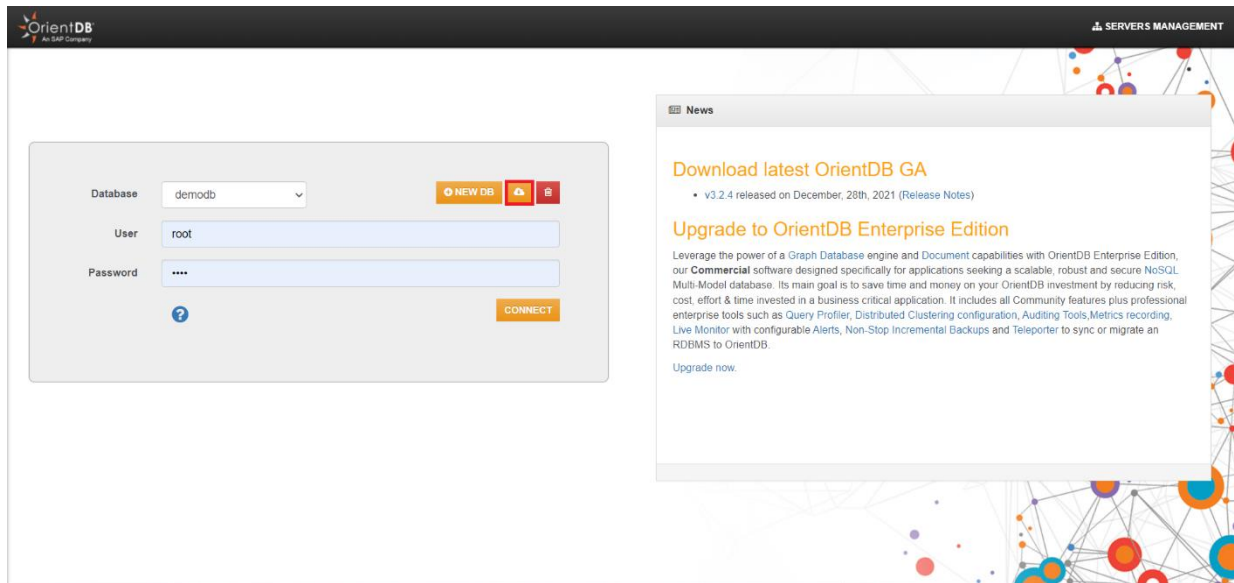
● Professor

— Teaches

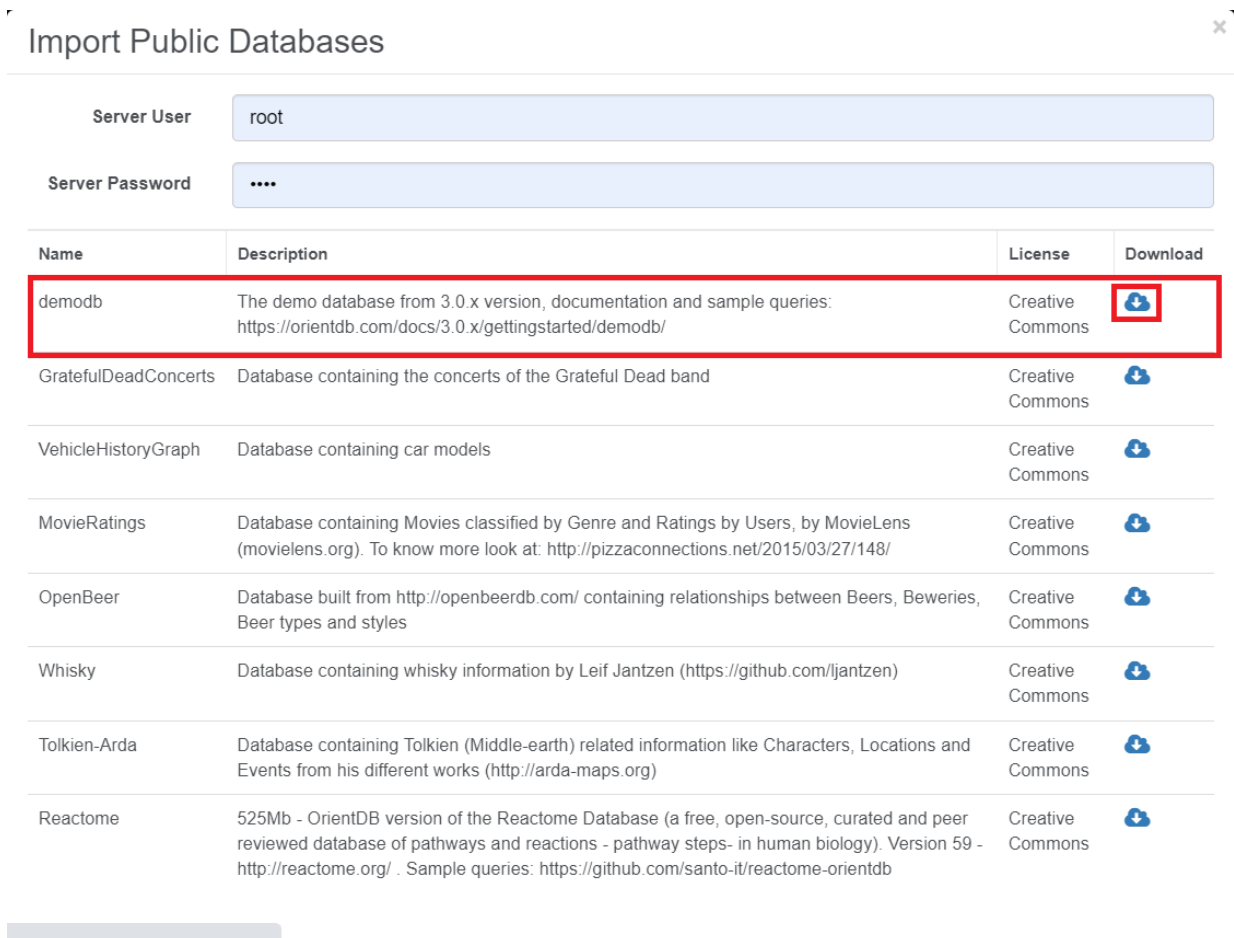
```
graph LR; P((#30:0)) -- Teaches --> S1((#22:0)); P -- Teaches --> S2((#23:0));
```

3.3. Importation BD existantes

Cliquez sur le bouton avec le nuage



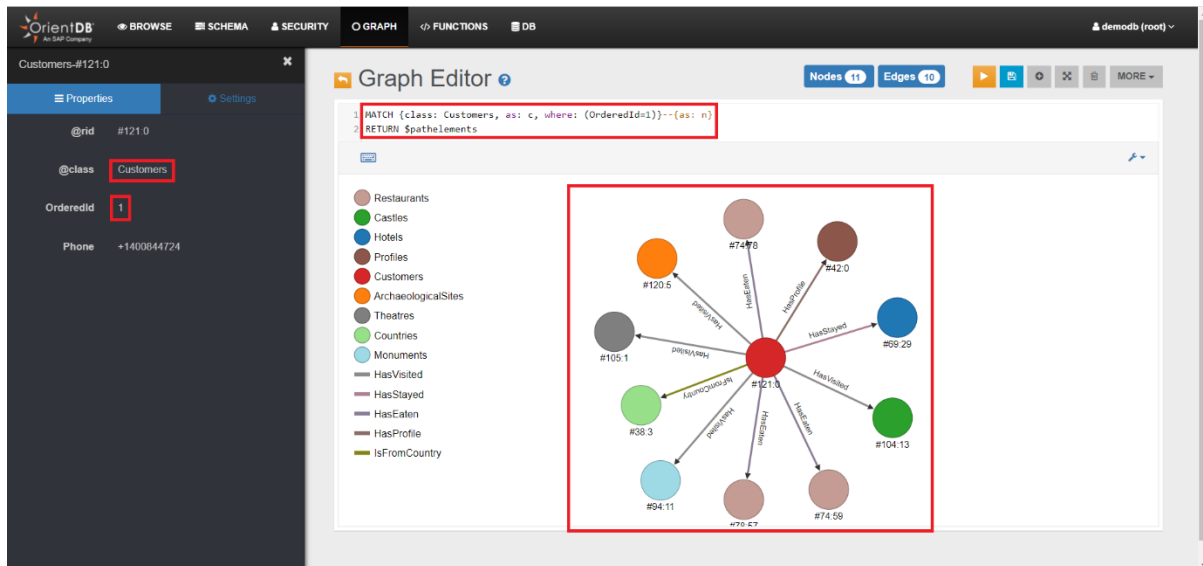
Indiquez votre pseudo + mdp puis sélectionnez la BD que vous voulez import, ici demodb



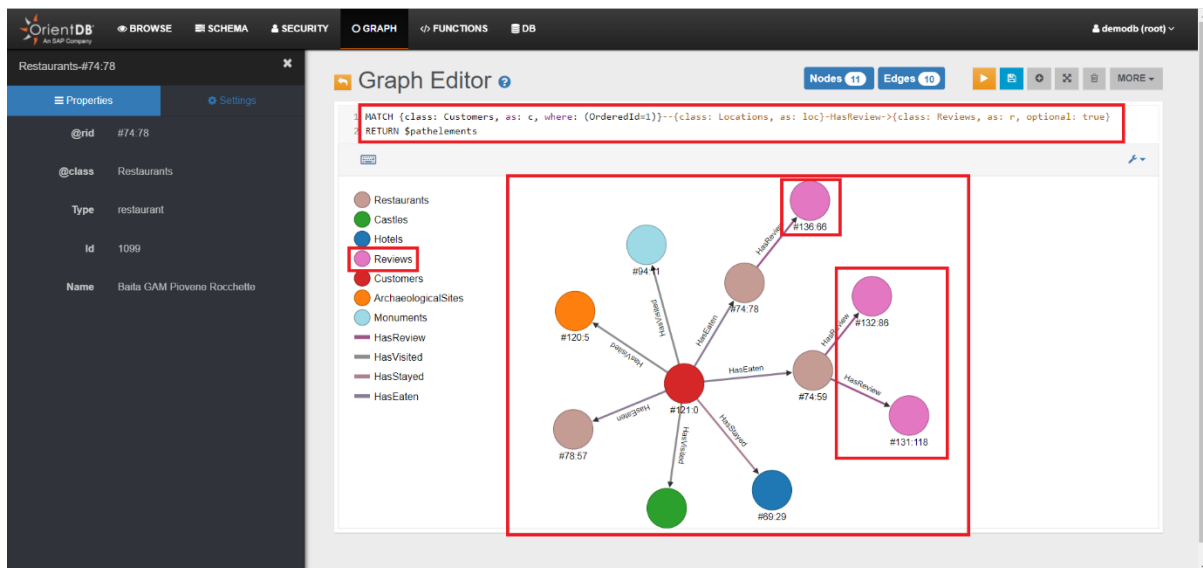
4. Fonctionnalités avancées

4.1. Graph Editor

Renvoyer tous les éléments directement connectés au premier client :



Renvoie-les commentaires/critiques des lieux dans lesquelles s'est rendu le premier client :



Renvoi toutes les commandes passées par le 2^{ème} client :

The screenshot shows the OrientDB Graph Editor interface. On the left, the 'Properties' panel for the 'Orders' class is displayed, showing the following details:

- @rid: #137:0
- @class: Orders
- Id: 1
- Amount: 536
- OrderDate: 2013-05-23 00:00:00

The main 'Graph Editor' window shows a query in the top panel:

```
1 MATCH {class: Customers, as: c, where: (OrderedId=2)}<-HasCustomer-{class: Orders, as: o}  
2 RETURN $pathElements
```

The graph visualization below the query shows three purple nodes representing 'Orders' with IDs #139:30, #143:77, and #138:54. These nodes are connected to a red node representing a 'Customer' with ID #123:6 via edges labeled 'HasCustomer'.

Retourne tous les endroits ou le premier client s'est restauré :

The screenshot shows the OrientDB Graph Editor interface. On the left, the 'Properties' panel for the 'Restaurants' class is displayed, showing the following details:

- @rid: #78:57
- @class: Restaurants
- Type: restaurant
- Id: 1738
- Name: Locanda della Luna

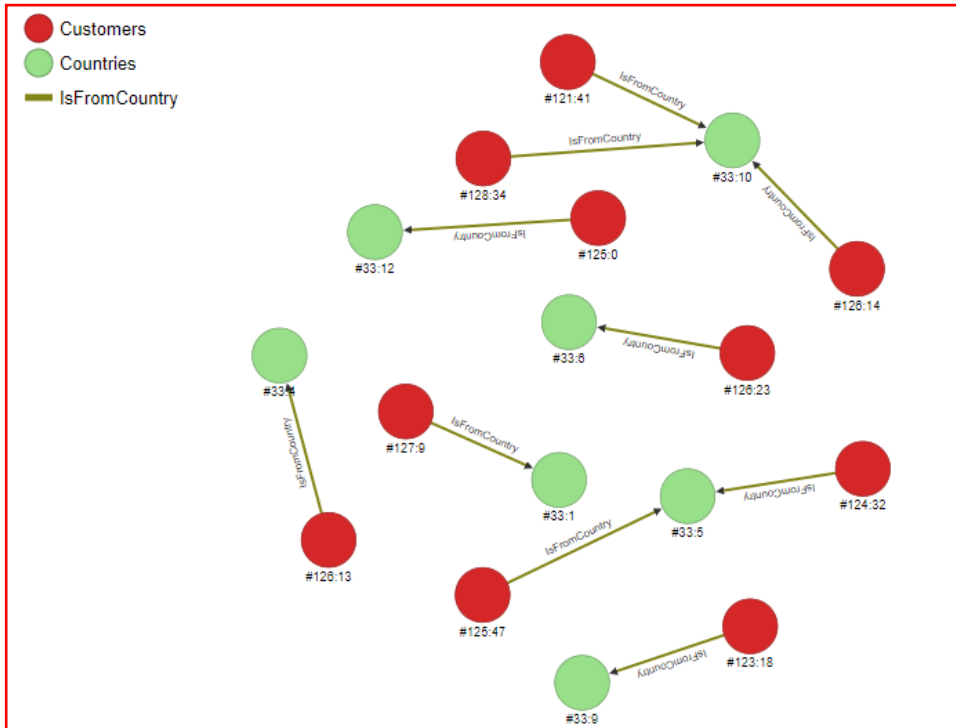
The main 'Graph Editor' window shows a query in the top panel:

```
1 MATCH (as: n)<-HasEaten-{class: Customers, as: c, where: (OrderedId=1)}  
2 RETURN $pathElements
```

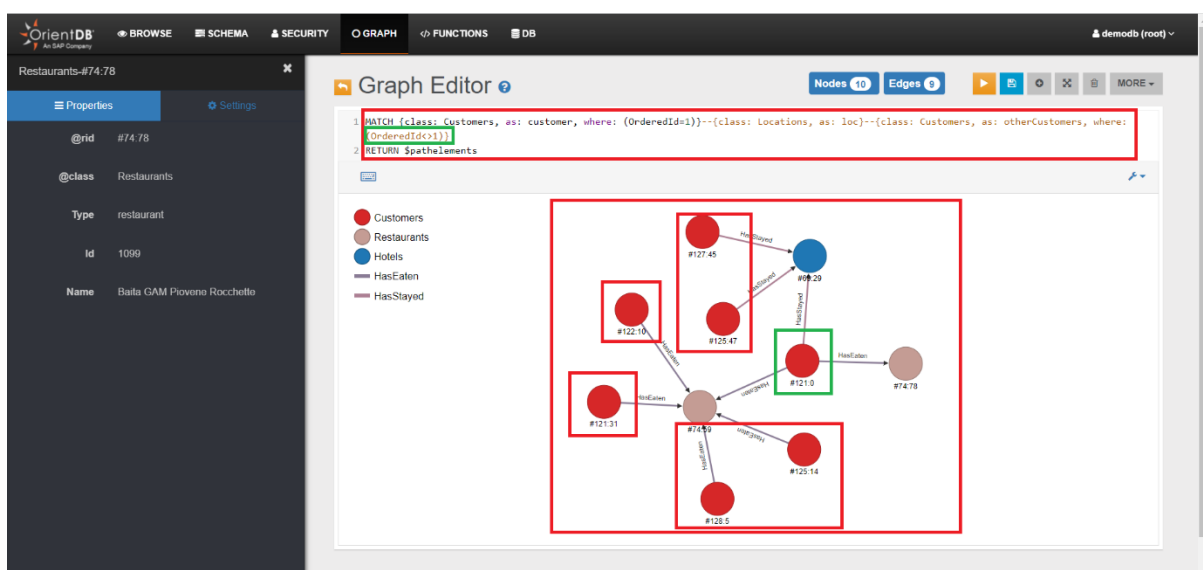
The graph visualization below the query shows a red node representing a 'Customer' with ID #121:0. This node is connected to three brown nodes representing 'Restaurants' with IDs #74:78, #78:57, and #74:59 via edges labeled 'HasEaten'.

Renvoi les pays d'origine des clients

```
1 match {class:Customers, as: customer}-IsFromCountry->{Class: Countries, as: country}
2 return $pathelements
```



Renvoi les clients ayant fréquentés les mêmes lieux que le client 1, ici le client 1 avait visité 2 restaurants (visités par 4 autres clients pour l'un et 0 pour l'autre) et 1 hôtel visité par 2 autres clients :



4.2. Browse Editor

Renvoi le montant total dépensé par le second client :

The screenshot shows the OrientDB Browse Editor interface. The query editor contains the following SQL query:

```
1 SELECT sum(Amount) as TotalAmount
2 FROM (
3   SELECT expand(in("HasCustomer"))
4   FROM Customers
5   WHERE OrderedId=2
6 )
```

The query is executed, and the result is displayed in a table with the following properties:

Properties
TotalAmount
1750

The interface also shows a search history, local storage size (137.42 KB), and a command history section with the executed query.

Retourne les 3 pays les plus représentés parmi les clients :

The screenshot shows the OrientDB Browse Editor interface. The query editor contains the following SQL query:

```
1 SELECT
2   Name as CountryName,
3   in("IsFromCountry").size() as NumberOfCustomers
4 FROM Countries
5 ORDER BY NumberOfCustomers DESC
6 LIMIT 3
```

The query is executed, and the result is displayed in a table with the following properties:

CountryName	NumberOfCustomers
Kyrgyzstan	7
Burundi	6
Madagascar	6

The interface also shows a search history, local storage size (121.85 KB), and a command history section with the executed query.

Retourne les infos de profil des clients ayant fréquentés les mêmes lieux que le client 2 triés par id :

The screenshot shows the OrientDB Browse Editor interface. The query editor contains the following SQL query:

```
1 MATCH {class: Customers, as: customer, where: (OrderedId=1)}--(class: Locations, as: loc)--(class: Customers, as: otherCustomers, where: (OrderedId<>1))-HasProfile->{class: Profiles, as: profile}
2 RETURN otherCustomers.OrderedId, profile.Name, profile.Surname, profile.Email
3 ORDER BY `otherCustomers.OrderedId` ASC
```

The query is executed, and the result is displayed in a table with the following properties:

otherCustomers.OrderedId	profile.Name	profile.Surname	profile.Email
8	Gene	Medina	gobaco@bo.co.uk
12	Amelia	Myers	olzil@jidirosag.org
26	Mayme	Perry	amle@dogi.gov
31	Keith	Clark	jekrav@bagow.co.uk
39	Glenn	Palmer	lupipi@nevilf.edu
48	Eddie	Ortega	za@ciwacil.edu
48	Eddie	Ortega	za@ciwacil.edu
51	Esther	Gibbs	ka@hi.com
76	Vernon	Murray	nuf@efpi.org
94	Henrietta	Roy	jaz@dav.io

The interface also shows a search history, local storage size (184.57 KB), and a command history section with the executed query.

5. Conclusion

- **Modèle de données** : OrientDB comme Neo4j est une base de données orientée graphes contrairement à Cassandra et MongoDB qui sont orientées document.
- **Scalabilité** : OrientDB comme Neo4j a une scalabilité plus limitée que Cassandra et MongoDB et leur scalabilité horizontale.
- **Transaction** : OrientDB et Neo4j assurent plus de fiabilité tandis que MongoDB et Cassandra assurent plus de cohérence.
- **Langage de Requête** : Chacun utilise son propre langage de requêtes (SQL orienté graphe pour OrientDB, Cypher pour Neo4j, CQL pour Cassandra et MQL pour MongoDB).
- **Communauté** : Cassandra, MongoDB et Neo4j ont des communautés et des supports importants, avec de nombreuses ressources en ligne pour aider les utilisateurs à résoudre les problèmes alors que OrientDB a une communauté bien plus limitée bien qu'il dispose d'une documentation en ligne complète et de forums de discussion actifs.