Neo4j CQL 简单增删改查命令

1.CQL简介

CQL代表Cypher查询语言，是neo4j的查询语言。像Oracle数据库有查询语言SQL。

CQL：

它是Neo4j图形数据库的查询语言。

它是一种声明性模式匹配语言

它遵循SQL语法。

它的语法是非常简单且人性化、可读的格式。

Neo4j CQL

支持多个子句像在哪里，顺序等，以非常简单的方式编写非常复杂的查询。

支持一些功能，如字符串，Aggregation等函数，它还支持一些关系功能。

1.1Neo4j CQL命令

常用的Neo4j CQL命令/条款如下:

S.No.	CQL命令/条	用法
1。	CREATE 创建	创建节点，关系和属性
2。	MATCH 匹配	检索有关节点，关系和属性数据
3。	RETURN 返回	返回查询结果
4。	WHERE 哪里	提供条件过滤检索数据
5。	DELETE 删除	删除节点和关系
6。	REMOVE 移除	删除节点和关系的属性
7。	ORDER BY 以...排序	排序检索数据
8。	SET 组	添加或更新标签

1.2Neo4j CQL 函数

S.No.	定制列表功能	用法
1。	String 字符串	它们用于使用String字面量。
2。	Aggregation 聚合	它们用于对CQL查询结果执行一些聚合操作。
3。	Relationship 关系	他们用于获取关系的细节，如startnode，endnode等。

1.3Neo4j CQL数据类型

Neo4j CQL支持以下数据类型:

S.No.	CQL数据类型	用法
1.	boolean	用于表示布尔文字：true，false。
2.	byte	用于表示8位整数。
3.	short	用于表示16位整数。
4.	int	用于表示32位整数。
5.	long	用于表示64位整数。
6.	float	l用于表示32位浮点数。
7.	double	用于表示64位浮点数。
8.	char	用于表示16位字符。
9.	String	用于表示字符串。

2.Create创建命令：

功能：

创建没有属性的节点

使用属性创建节点

在没有属性的节点之间创建关系

使用属性创建节点之间的关系

为节点或关系创建单个或多个标签

2.1创建一个没有属性的节点

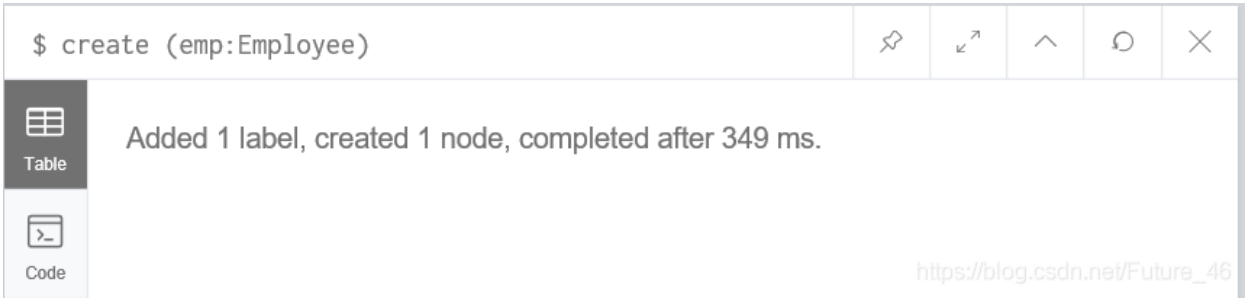
语法 ：

```
CREATE (<node-name>:<label-name>)
```

eg：CREATE (emp:Employee)

emp是一个节点名，Employee是节点emp的标签名

下图为创建成功



2.2创建具有属性的节点

语法：

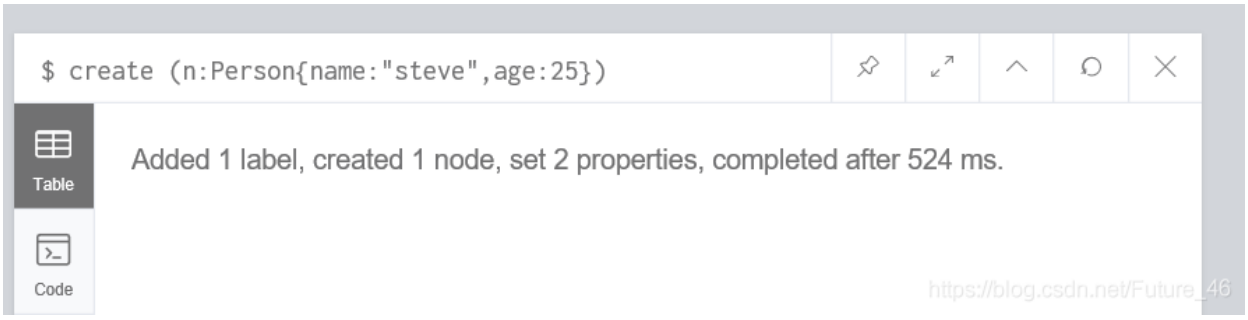
```
CREATE (
  <node-name>:<label-name>
  {
    <Property1-name>:<Property1-Value>
    ........
    <Propertyn-name>:<Propertyn-Value>
  }
)
```

eg：create (n:Person{name:“steve”,age:25})

n为节点名，Person为节点的标签名，name、age是属性名，属性的值是“steve”、25。

注意 - 要定义字符串类型属性值，我们需要使用单引号或双引号。

下图创建Person节点



2.3创建标签

Label是Neo4j数据库中的节点或关系的名称或标识符。

create 可以为节点创建一个或多个标签、为关系创建单个标签

2.3.1创建单个标签

```
CREATE (<node-name>:<label-name>)
```

eg：CREATE (google1:GooglePlusProfile)

google1是节点名，GooglePlusProfile是google1节点的标签名

2.3.2创建多个标签

CREATE (<node-name>:<label-name1>:<label-name2>.....:<label-namen>)

使用 “：”运算符来分隔节点名和标签名，分隔多个标签名
eg：CREATE (m:Movie:Cinema:Film:Picture)
上面例子为节点m创建多个标签名，Movie、Cinema、Film、Picture是节点的多个标签名称。

2.3.3创建单个关系标签

```
CREATE (<node1-name>:<label1-name>)-
    [(<relationship-name>:<relationship-label-name>)]
    ->(<node2-name>:<label2-name>)
```

eg：CREATE (p1:Profile1)-[r1:LIKES]->(p2:Profile2)
这里 p1和Profile1是节点名称和节点标签名称 ， p1是From Node
p2和Profile2 是To Node的节点名称和节点标签名称
r1是关系名称 ，LIKES是关系的标签名称

3.MATCH命令

功能：
从数据库获取有关节点和属性的数据
从数据库获取有关节点，关系和属性的数据

3.1Return命令作用：

检索节点的某些属性
检索节点的所有属性
检索节点和关联关系的某些属性
检索节点和关联关系的所有属性
单独使用MATCH和RETURN会报语法错误。正确的使用方法是合并这两个命令

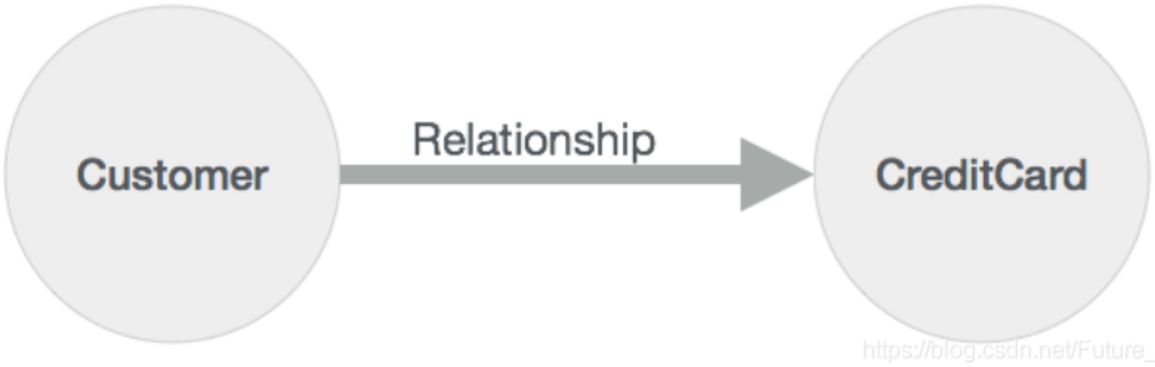
MATCH RETURN命令语法：

```
MATCH Command
RETURN Command
```

eg：MATCH (n:Employee) RETURN n.name,n.deptno,n.sal
上面n为节点名称，Employee为节点标签名，name、deptno、sal为节点的属性名
运行结果图如下：

3.2Neo4j 关系基础

属性图模型的关系是定向的Noe4j主要有两种类型。 单向关系、双向关系。



如上图所示：这里的关系是从客户->信用卡的， Neo4jCQL使用类是类箭头的标记创建两个节点之间的关系，每个关系包含两个节点：从节点、到节点。Customer是从节点，CreaditCard是到节点。
节点有两种关系：外向关系、传入关系。上图关系是到客户节点的“外向关系”，并且相同的关系是到信用卡节点的“到达关系”

4. delete删除

4.1功能简介

DELETE子句用来 删除节点和删除节点及相关节点和关系，通过使用此命令，我们可以从数据库永久删除节点及其关联的属性。

4.2删除节点子句

DELETE <node-name-list>

可以使用逗号”，“来分隔节点名，从而删除多个节点。
下面是删除Employee节点，此操作为永久删除数据

```
MATCH (e: Employee) DELETE e
```

4.3DELETE节点和关系子句语法

DELETE <node1-name>,<node2-name>,<relationship-name>

eg:
MATCH (cc: CreditCard)-[rel]-(c:Customer)
DELETE cc,c,rel
用逗号（，）来分隔节点名称和关系名称。

5.REMOVE删除

5.1功能简介

从数据库中永久删除节点或关系的属性或属性列表
删除节点或关系的标签
删除节点或关系的属性

5.2 delete和remove命令之间的区别

delete删除节点和关联关系。
remove删除标签和属性。
相似之处
这个两个命令不能单独使用
应该与match命令一起使用
REMOVE子句语法
REMOVE <property-name-list>

eg: match (n:Person) where n.name=“Mike” remove n.title
或 match (n:Person{name: ‘Mike’}) remove n.title
上面是删除节点的属性，接下来说下删除节点/关系的标签
REMOVE //节点列表名用逗号分隔
eg: MATCH (m:Movie) REMOVE m:Picture
此命令删除了m节点的Picture标签

6.SET子句

向现有节点或关系添加新属性，添加或更新属性值
SET子句语法：
SET <property-name-list> //属性列表用逗号分隔

eg: match (n:Person) where n.name=‘Mike’ set n.age=20
名字为Mike的节点，添加或修改其age值为20

增删改查

1、添加4个节点

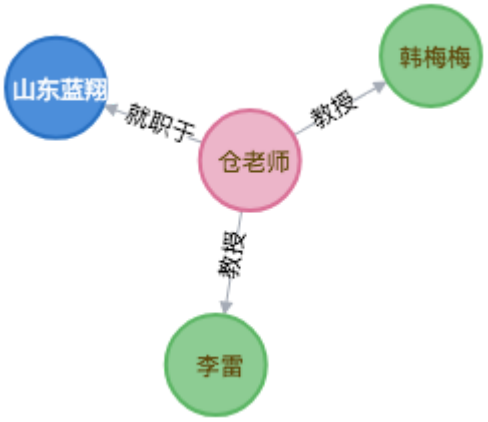
```
create(student:Student{id:1, name: "李雷"});
create(student:Student{id:2, name: "韩梅梅"});
create(teacher:Teacher{id:1, name: "仓老师"});
create(school:School{id: 1, title: "山东蓝翔"});
```

小写student 可以理解为面向对象的实体

大写Student 可以理解为面向对象的类

2、添加3个关系

```
match(s:Student{id:1}),(t:Teacher{id:1}) create (t)-[r:教授]->(s);
match(s:Student{id:2}),(t:Teacher{id:1}) create (t)-[r:教授]->(s);
match(s:School{id:1}),(t:Teacher{id:1}) create (t)-[r:就职于]->(s);
```



3、更新节点 老师改名字了

```
match(t:Teacher)
where t.id=1
set t.name="小泽老师"
```

4、删除节点 李雷被开除了

```
match(s:Student{id: 1})
detach delete s;
```

5、删除关系 小泽老师去别的班级上课了

```
match (t:Teacher)-[r:教授]->(s:Student)
where t.id=1 and s.id=2
delete r
```

6、索引操作

查看所有索引

```
:schema
```

创建索引

```
create index on:Student(name)
```

删除索引

```
drop index on:Student(name)
```

创建唯一索引

```
create constraint on (s:Teacher) assert s.name is unique
```

删除唯一索引

```
drop constraint on (s:Teacher) assert s.name is unique
```

修改关系名称

```
MATCH (p:Person)-[r:投资]->(c:Company)
CREATE (p)-[r2:Invest{name:"投资"}]->(c)
// copy properties, if necessary
SET r2 = r
WITH r
DELETE r
```

> https://stackoverflow.com/questions/22670369/neo4j-cypher-how-to-change-the-type-of-a-relationship

事务操作

```
from py2neo import Graph, Node, Relationship

g = Graph()
tx = g.begin()
a = Node("Person", name="Alice")
```

```
tx.create(a)
b = Node("Person", name="Bob")
ab = Relationship(a, "KNOWS", b)
tx.create(ab)
tx.commit()
```

批量导入

movies2.csv.

```
movieId:ID;title;year:int;:LABEL
tt0133093;'The Matrix';1999;Movie
tt0234215;'The Matrix Reloaded';2003;Movie|Sequel
tt0242653;'The Matrix Revolutions';2003;Movie|Sequel
```

actors2.csv.

```
personId:ID;name;:LABEL
keanu;'Keanu Reeves';Actor
laurence;'Laurence Fishburne';Actor
carrieanne;'Carrie-Anne Moss';Actor
```

roles2.csv.

```
:START_ID;role;:END_ID;:TYPE
keanu;'Neo';tt0133093;ACTED_IN
keanu;'Neo';tt0234215;ACTED_IN
keanu;'Neo';tt0242653;ACTED_IN
laurence;'Morpheus';tt0133093;ACTED_IN
laurence;'Morpheus';tt0234215;ACTED_IN
laurence;'Morpheus';tt0242653;ACTED_IN
carrieanne;'Trinity';tt0133093;ACTED_IN
carrieanne;'Trinity';tt0234215;ACTED_IN
carrieanne;'Trinity';tt0242653;ACTED_IN
```

The call to neo4j-admin import would look like this:

导入指令

```
$ bin/neo4j-admin import
--nodes import/movies2.csv
--nodes import/actors2.csv
--relationships import/roles2.csv
--delimiter ";"
--array-delimiter "|"
--quote "'"
```

配置外网访问

conf\neo4j.conf

#dbms.connector.http.listen_address=:7474
dbms.connector.http.listen_address=0.0.0.0:7474

#dbms.connector.bolt.listen_address=:7687
dbms.connector.bolt.listen_address=0.0.0.0:7687

配置环境变量：

#neo4j

export NEO4J_HOME=<neo4j_path>

export PATH=$PATH:$NEO4J_HOME/bin