Introduction to Neo4j

# Code and data

<https://github.com/lukedawilson/Neo4jDemo>

[\\kaizar\users\luke.wilson\TechTalk\TrayportHobbies.xlsx](file:///\\kaizar\users\luke.wilson\TechTalk\TrayportHobbies.xlsx)

# Download link

<http://www.neo4j.org/download_thanks?edition=community&release=2.0.3&platform=windows&packaging=exe&architecture=x64>

# Documentation (some of this is out-of-date)

<https://github.com/Readify/Neo4jClient/wiki>

<https://github.com/Readify/Neo4jClient/wiki/cypher>

<https://github.com/Readify/Neo4jClient/wiki/indexes>

# .NET API

<https://www.nuget.org/packages/Neo4jClient>

# Examples

## Connect to the database

var client = new GraphClient(new Uri("http://localhost:7474/db/data"));

client.Connect();

## Create an index

client.CreateIndex(

"index\_name",

new IndexConfiguration { Provider = IndexProvider.lucene, Type = IndexType.exact },

IndexFor.Node);

## Create and insert a node on the given index

var nodeRef = client.Create(

new FooNode { BarProperty = "baz value" },

null,

new[] { new IndexEntry("index\_name") { { "key", *VALUE* } } });

## Create a relationship between two nodes

public class MyRelationship :

Relationship,

IRelationshipAllowingSourceNode<FooNode>,

IRelationshipAllowingTargetNode<BarNode>

{

private const string TypeKey = "MY\_RELATIONSHIP\_TYPE";

public WroteRelationship(NodeReference targetNode, MyRelationshipData data)

: base(targetNode, data)

{ }

public override string RelationshipTypeKey

{

get { return TypeKey; }

}

}

## client.CreateRelationship(nodeRef, new MyRelationship (targetNodeRef));

## A simple query

### C#

var query = client.Cypher

.Start(new { n = Node.ByIndexLookup("index\_name", "key", "value") })

.Match("n-[r:MY\_RELATIONSHIP\_TYPE]->e")

.Return((n, e, r) => new

{

Subject = n.As<FooNode>().WhateverProperty,

Object = e.As<BarNode>().WhateverProperty,

Relationship = r.As<MyRelationshipData>().WhateverProperty

});

foreach (var result in query.Results)

Console.WriteLine(result);

### Equivalent Cypher query

**START** n**=**node**:**idx\_artists**(key** **=** "value"**)**

**MATCH** n**-[**r**:**MY\_RELATIONSHIP\_TYPE**]->**e

**RETURN** n**.**foo **AS** WhateverProperty**,** e**.**bar **AS** WhateverProperty

## Delete an index

if (client.CheckIndexExists("index\_name", IndexFor.Node))

client.DeleteIndex("index\_name", IndexFor.Node);

## Delete a node

client.Delete(new NodeReference(*ID*), DeleteMode.NodeAndRelationships);

## Delete all nodes and relationships

### C#

var query = client.Cypher.Match("(n)").OptionalMatch("(n)-[r]-()").Delete("n, r");

query.ExecuteWithoutResults();

### Cypher

**MATCH** **(**n**)**

**OPTIONAL** **MATCH** **(**n**)-[**r**]-()**

**DELETE** n**,** r

## Where clause

**MATCH** **(**n**)**

**WHERE** n**.**name **=** 'Peter' **XOR** **(**n**.**age **<** 30 **AND** n**.**name **=** "Tobias"**)** **OR** **NOT** n**.**name **=** "Tobias"

**RETURN** n

## Something a bit more interesting

### Cypher

**START** n**=**node**:**idx\_artists**(**id **=** 1**)**

**MATCH** n**-[**r1**:**COVERED**]->**e

**MATCH** e**<-[**r2**:**WROTE**]-**x

**RETURN DISTINCT** n.Name **AS** CoveringArtist, x.Name **as** Composer

### Equivalent SQL

**SELECT** **DISTINCT** coveringArtist**.**Name**,** composer**.**Name

**FROM** dbo**.**Artists coveringArtist

**JOIN** dbo**.**Covers cov **ON** coveringArtist**.**Id **=** cov**.**ArtistId

**JOIN** dbo**.**Tracks tr **ON** cov**.**TrackId **=** tr**.**Id

**JOIN** dbo**.**Compositions com **ON** tr**.**Id **=** com**.**TrackId

**JOIN** dbo**.**Artists composer **ON** com**.**ArtistId **=** composer**.**Id

**WHERE** coveringArtist**.**Id **=** 1