- DEV6B -
MapReduce
&
Graphs and Cypher

Quick & Dirty Recap

Wat gaan we bespreken

- MapReduce => Lambda's!
- Neo4J => Cirkels... en pijlen!

JSON; heel veel JSON

Wat heb je nodig?

Mentale flexibiliteit

+3 uur aan tijd... yeap...

Je kunt:
C# lezen en schrijven
Je begrijpt wat classes zijn en kunnen

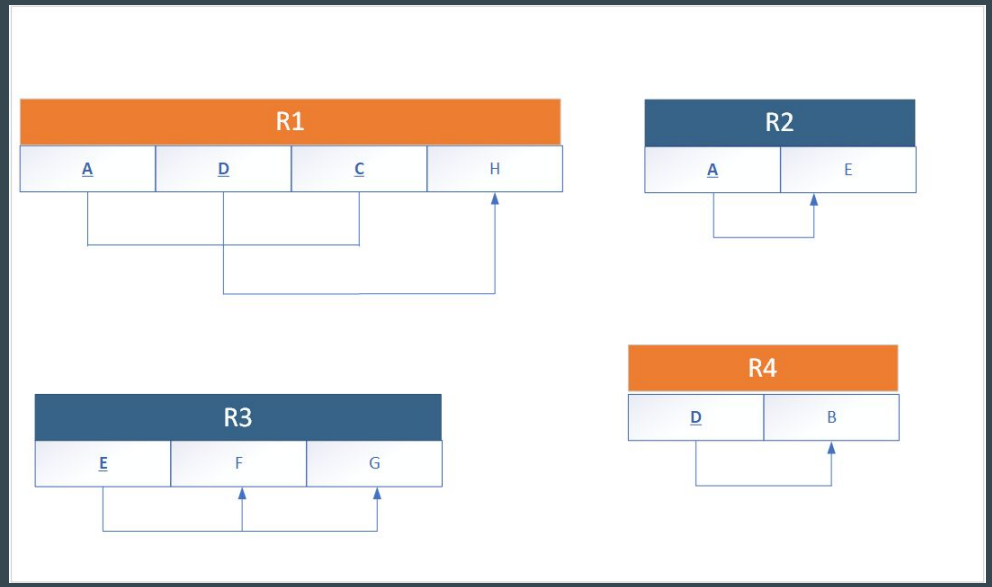
Slides & code
https://github.com/RicardoStam/Development6B

Introductie

...

Introductie

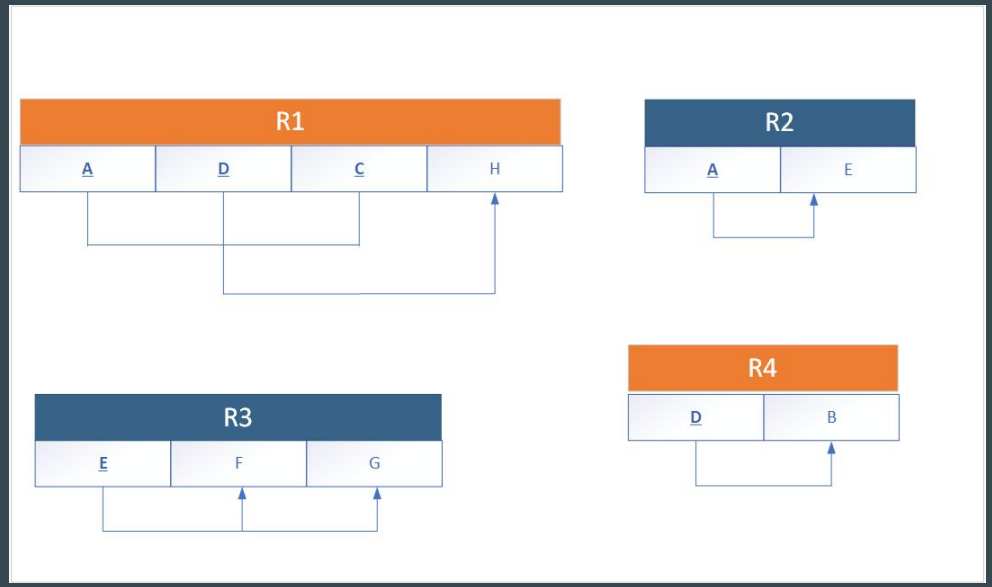
Relationele databases



Introductie

Relationele databases => SQL

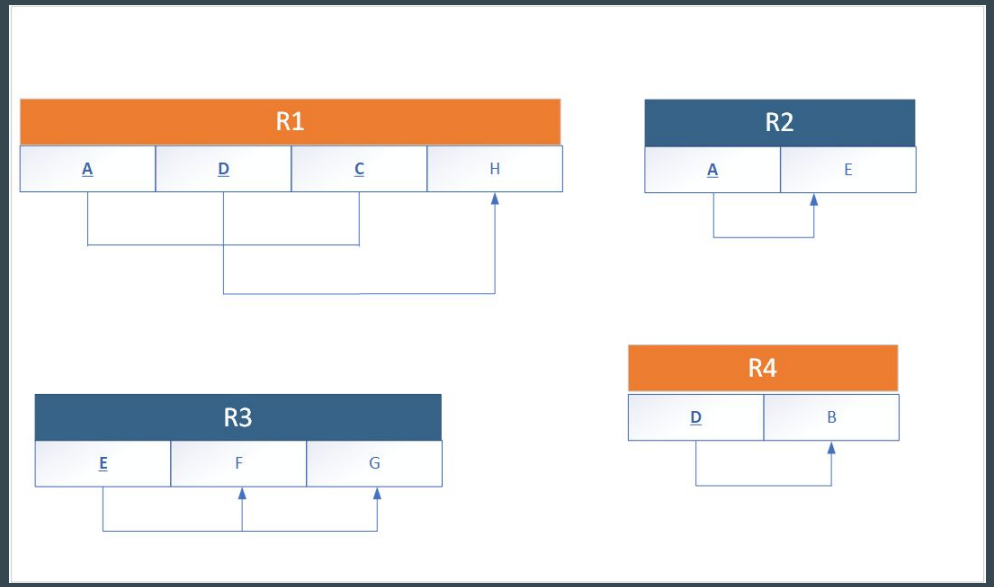
```
SELECT *
FROM R1
WHERE C > 5;
```



Introductie

Relationele databases => SQL

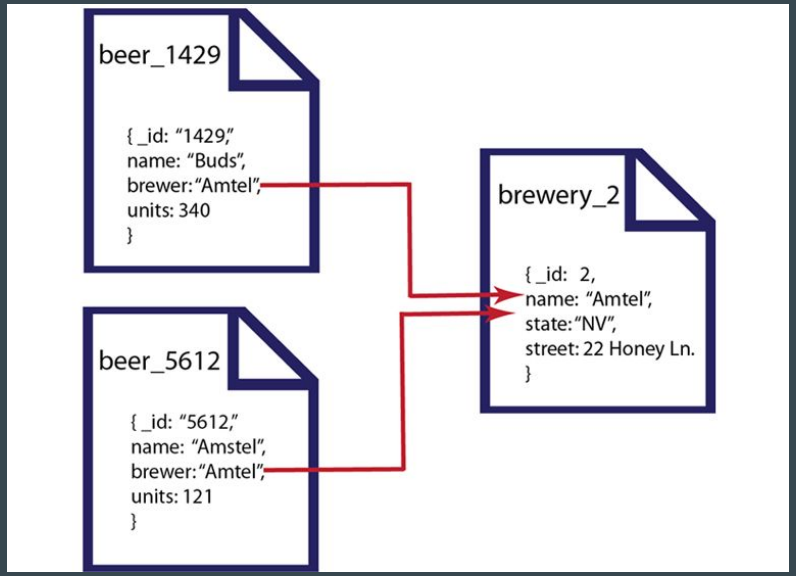
```
SELECT R1.D, R2.E
FROM R1, R2
WHERE R1.A == R2.A;
```




MapReduce

MapReduce

Document datastore



Het idee - map - reduce - join - code - opdrachten

MapReduce

Document datastore
JavaScript Object Notation a.k.a JSON

```
Var x =
{
        “Thing”: “data”,
        “Number”: 123
        “NewThing”: {
                                “InnerThing”: “thing”
                        }
}
```

Het idee - map - reduce - join - code - opdrachten

MapReduce

Belangrijk:

Var x = new {<data>,<possiblyMoreData>}

⇒ Anonymous class

MapReduce

Higher order functions!

Map (Manipulate) (Transformation) $\lambda\ A \Rightarrow B$

Reduce (Operation) $\lambda\ (B, A) \Rightarrow B$

Join (Condition) $\lambda\ (A, B) \Rightarrow Bool$

Het idee - map - reduce - join - code - opdrachten

MapReduce

Map & Reduce zijn een methode om queries uit te voeren op verzamelingen

Select == Map
Where == Reduce
Join == Reduce (+ Reduce)

Het idee - map - reduce - join - code - opdrachten

MapReduce

Map & Reduce zijn een methode om queries uit te voeren op verzamelingen

Select == Map
Where == Reduce
Join == Reduce (+ Reduce)

Join == Where waarbij een tweede tabel wordt betrokken en samengevoegd.

Het idee - map - reduce - join - code - opdrachten

MapReduce


```
IEnumerable<T2> Map<T1, T2>
(IEnumerable<T1> collection,
Func<T1, T2> transformation)
```

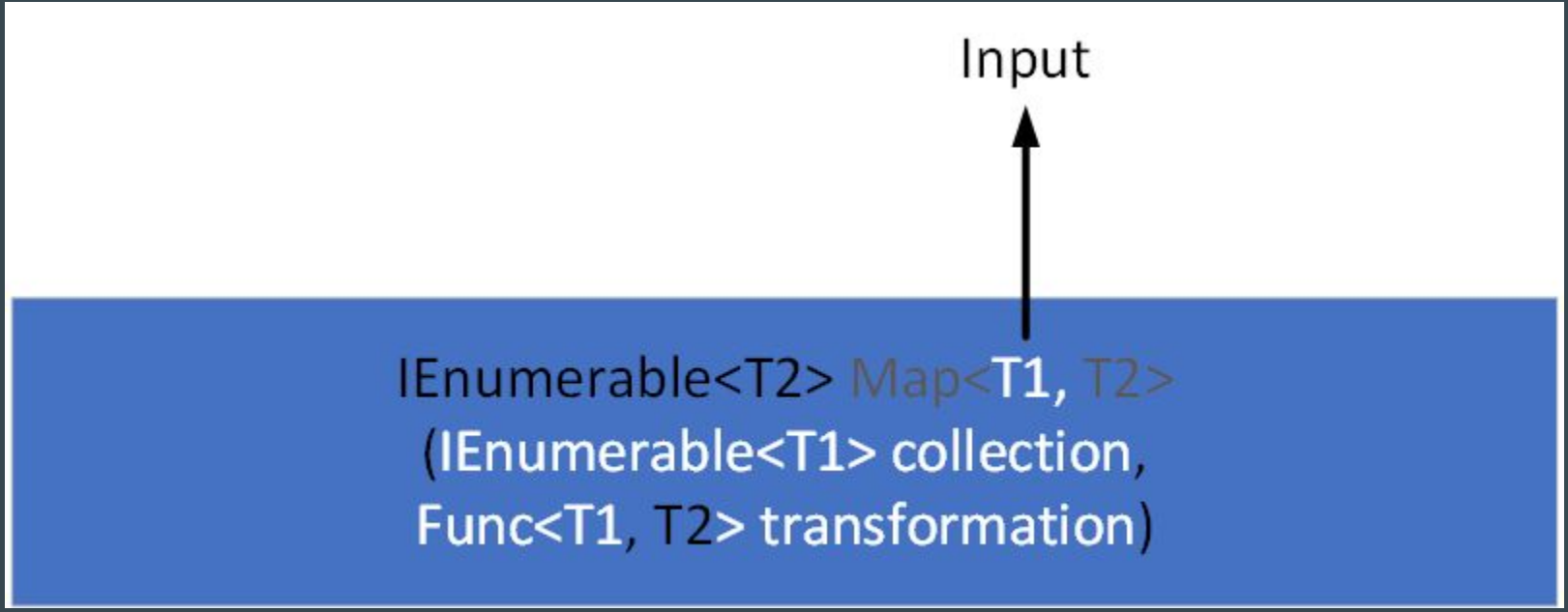
Het idee - map - reduce - join - code - opdrachten

MapReduce


```
IEnumerable<T2> Map<T1, T2>
(IEnumerable<T1> collection,
Func<T1, T2> transformation)
```

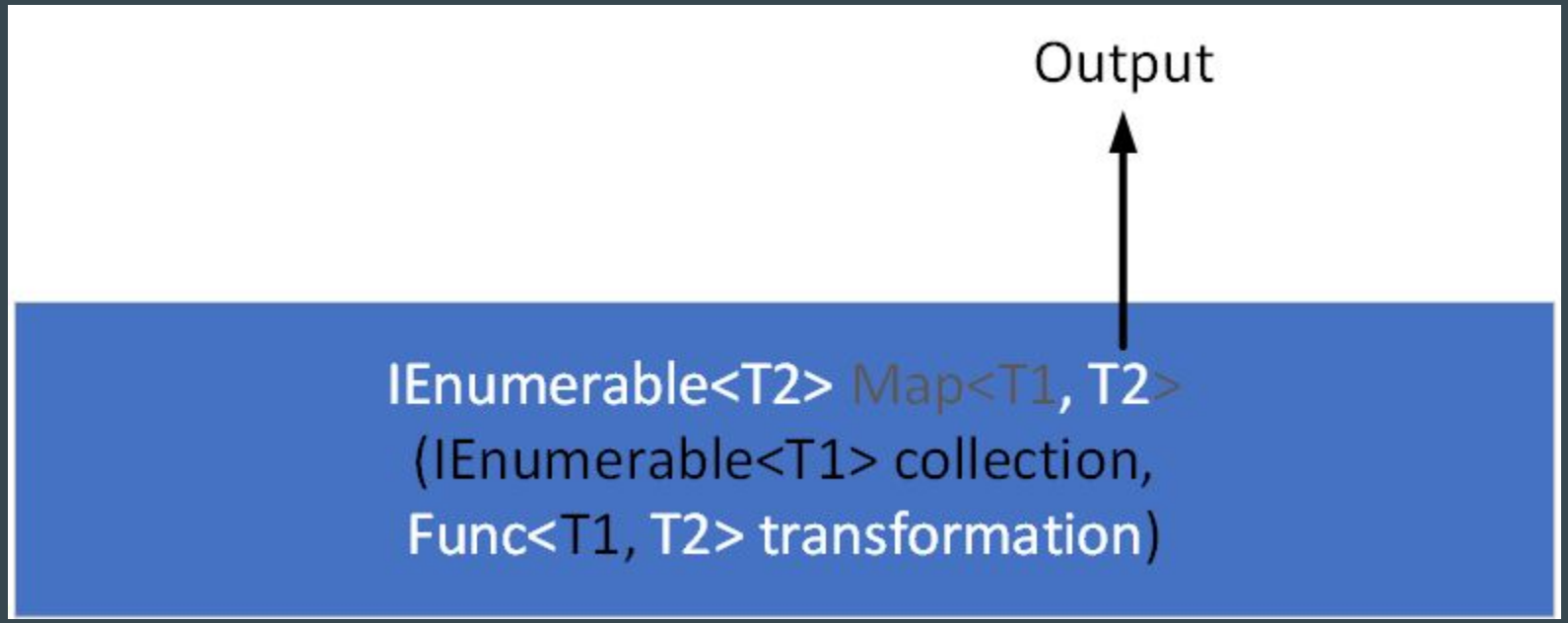
Het idee - map - reduce - join - code - opdrachten

MapReduce



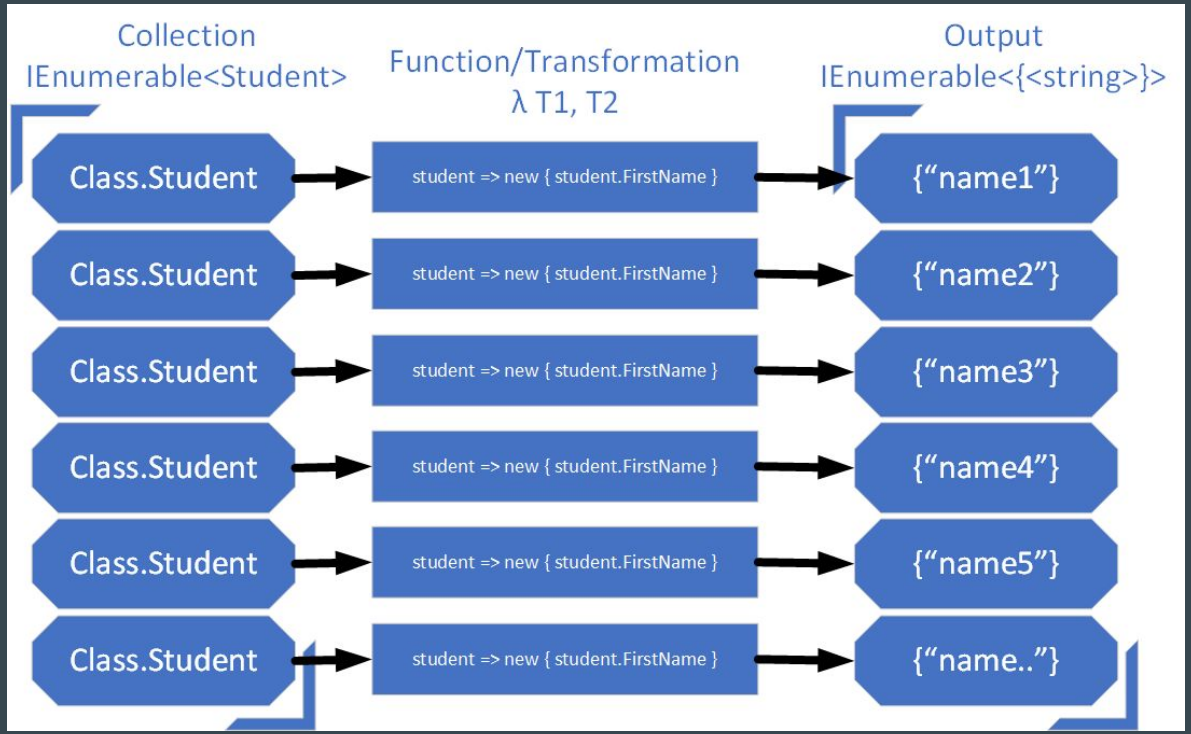
Het idee - map - reduce - join - code - opdrachten

MapReduce



Het idee - map - reduce - join - code - opdrachten

MapReduce



Het idee - map - reduce - join - code - opdrachten

MapReduce


```
public static IEnumerable<T2> Map<T1, T2>(this IEnumerable<T1> collection, Func<T1, T2> transformation)
{
    // Create an array (result) with the size of the input collection
    T2[] result = new T2[collection.Count()];
    for (int i = 0; i < collection.Count(); i++)
    {
        // Add the result to the same index after the transformation is done to
        // an element of the input collection.
        result[i] = transformation(collection.ElementAt(i));
    }
    return result;
}
```

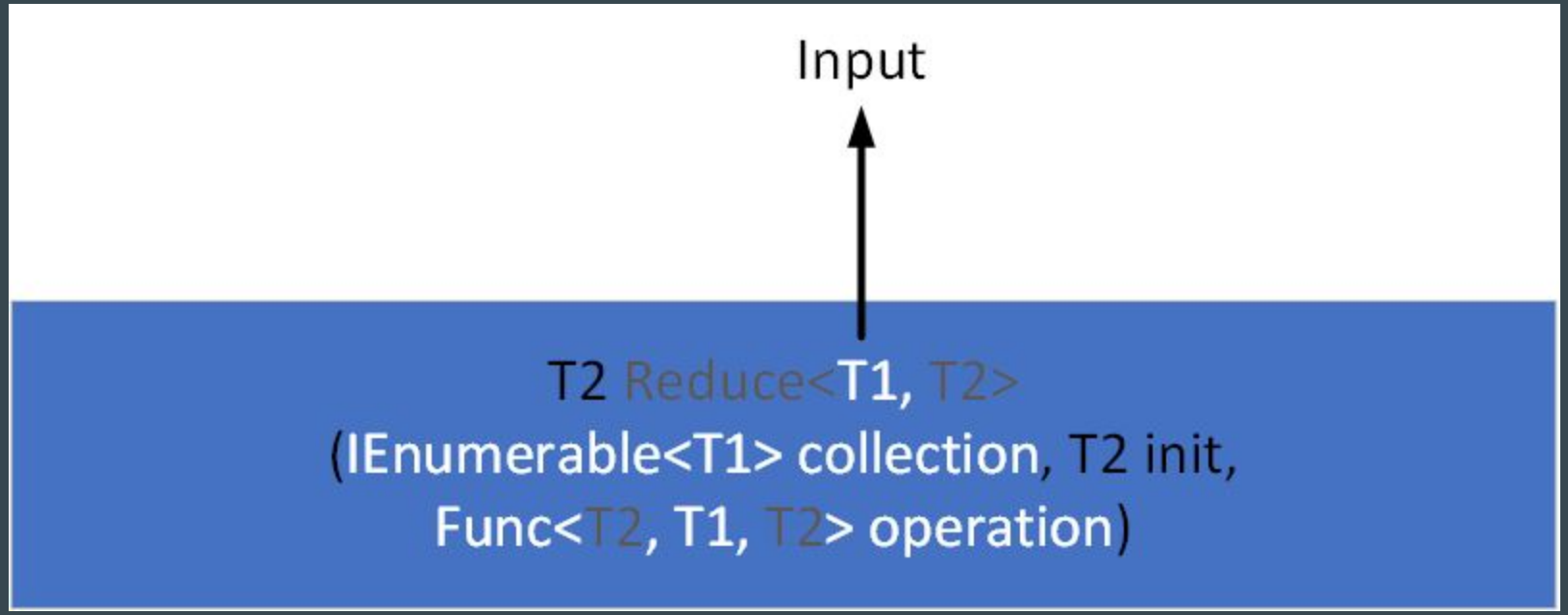
Het idee - map - reduce - join - code - opdrachten

MapReduce


```
T2 Reduce<T1, T2>
(IEnumerable<T1> collection, T2 init,
Func<T2, T1, T2> operation)
```

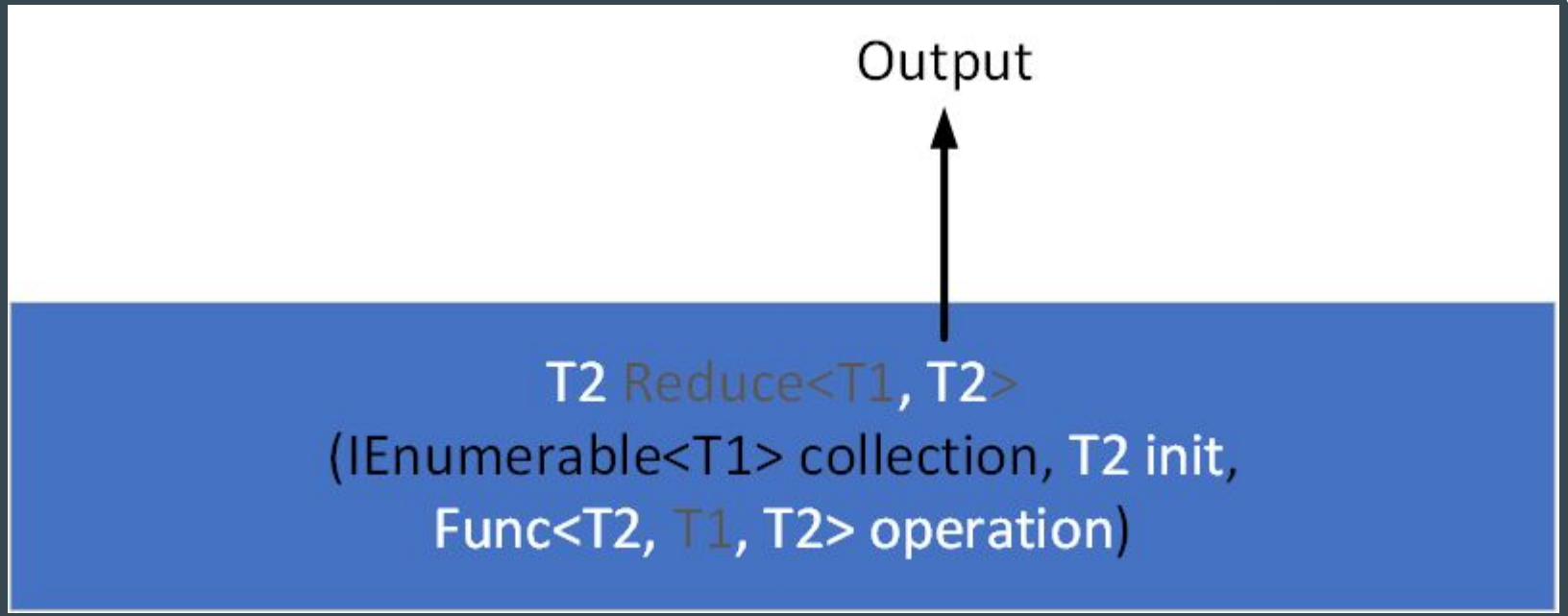
Het idee - map - reduce - join - code - opdrachten

MapReduce



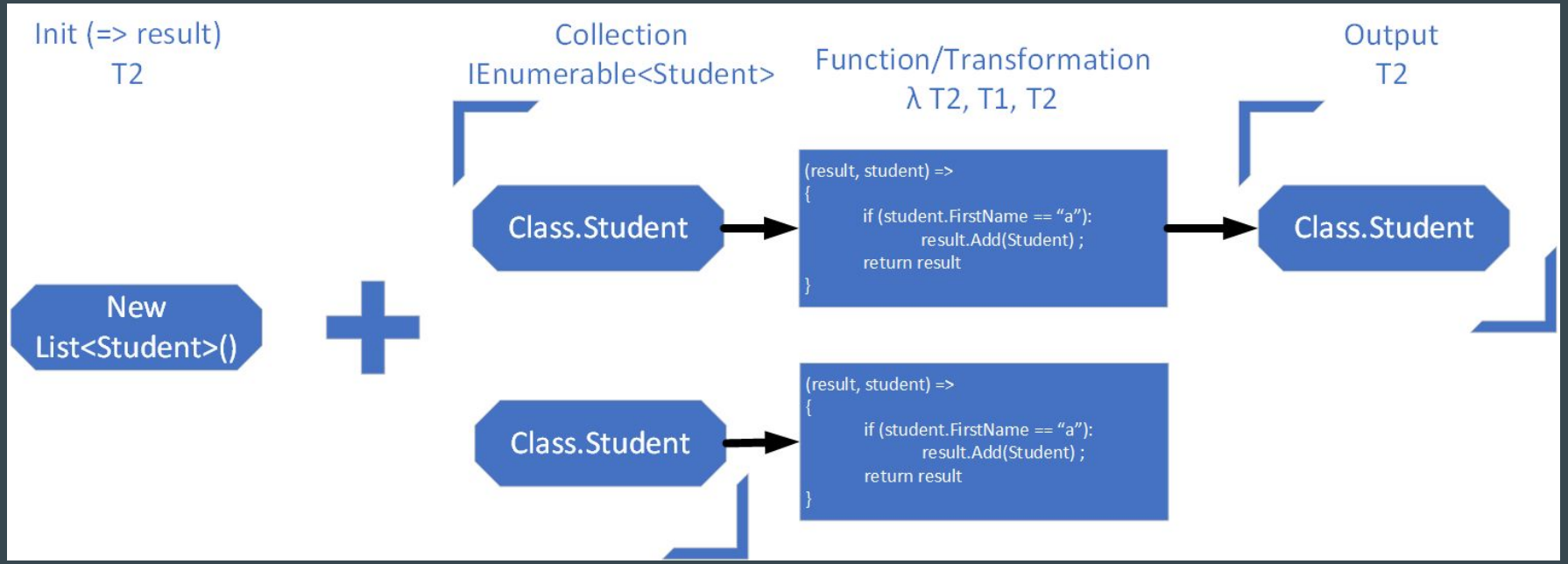
Het idee - map - **reduce** - join - code - opdrachten

MapReduce



Het idee - map - **reduce** - join - code - opdrachten

MapReduce



Het idee - map - **reduce** - join - code - opdrachten

MapReduce


```
public static T2 Reduce<T1, T2>(this IEnumerable<T1> collection, T2 init, Func<T2, T1, T2> operation)
{
    // Create a result variable; based on init.
    T2 result = init;
    for (int i = 0; i < collection.Count(); i++)
    {
        // The result equals the outcome of the opperation.
        // The result is parsed in as the first parameter.
        result = operation(result, collection.ElementAt(i));
    }
    return result;
}
```

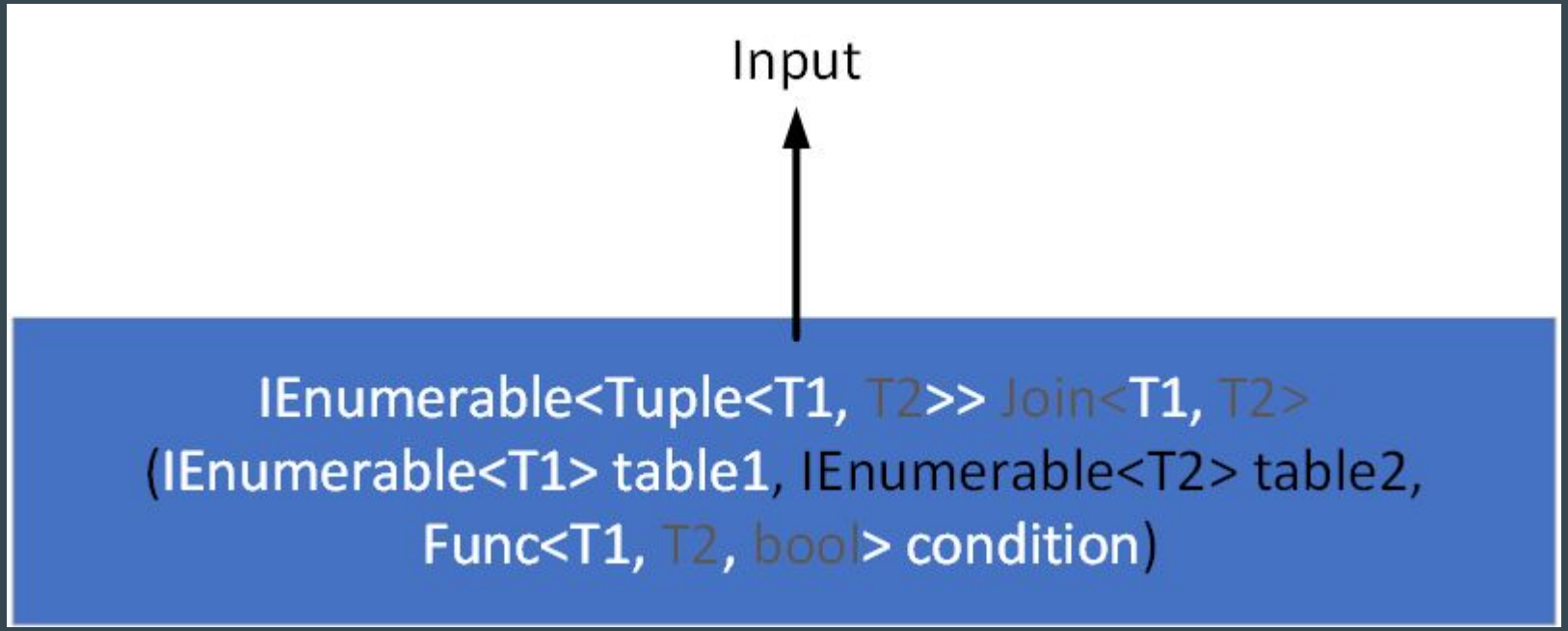
Het idee - map - **reduce** - join - code - opdrachten

MapReduce


```
IEnumerable<Tuple<T1, T2>> Join<T1, T2>
(IEnumerable<T1> table1, IEnumerable<T2> table2,
Func<T1, T2, bool> condition)
```

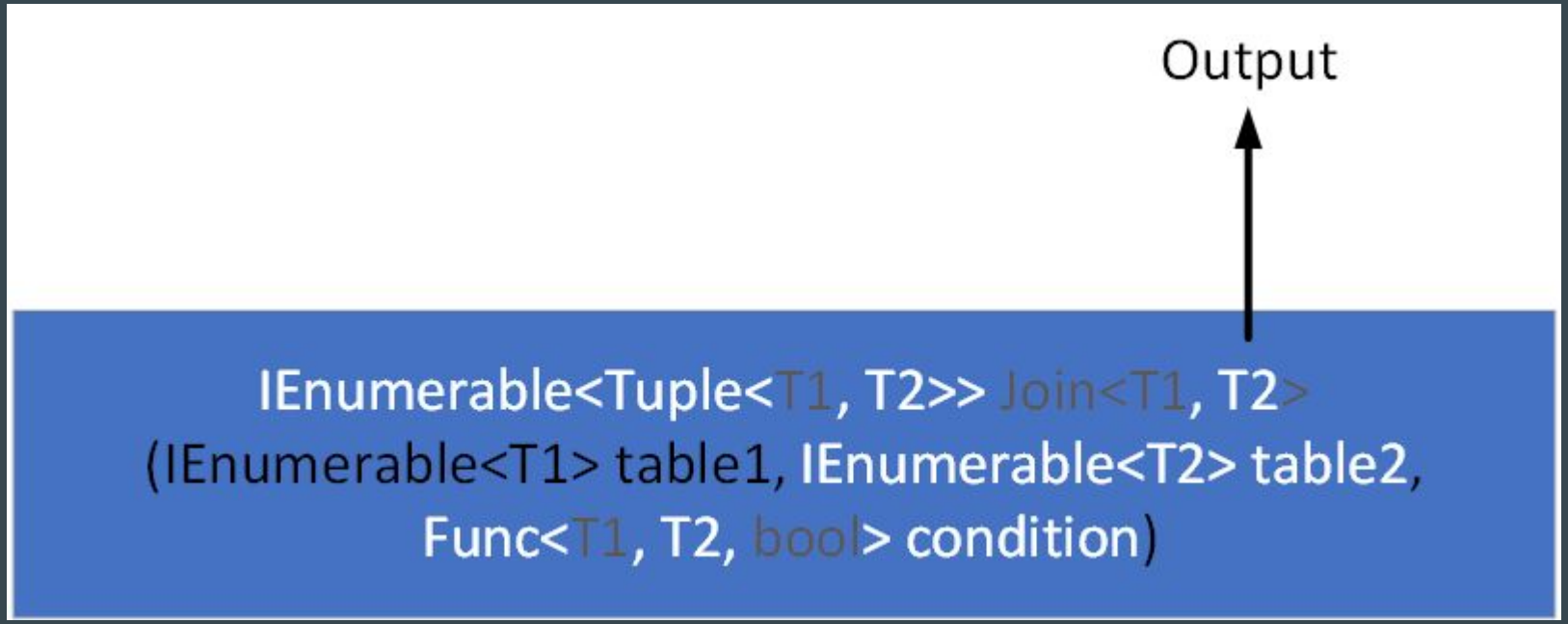
Het idee - map - reduce - join - code - opdrachten

MapReduce



Het idee - map - reduce - join - code - opdrachten

MapReduce



Het idee - map - reduce - join - code - opdrachten

MapReduce


```
public static IEnumerable<Tuple<T1, T2>> Join<T1, T2>(this IEnumerable<T1> table1, IEnumerable<T2> table2,
                                                      Func<T1, T2, bool> condition)
{
    return  Reduce
            (
            table1,
            new List<Tuple<T1, T2>>(),
            (queryResult, table1Element) =>...);
}
```

Het idee - map - reduce - join - code - opdrachten

```
public static IEnumerable<Tuple<T1, T2>> Join<T1, T2>(this IEnumerable<T1> table1, IEnumerable<T2> table2,
                                                      Func<T1, T2, bool> condition)
{
    return  Reduce
            (
            table1,
            new List<Tuple<T1, T2>>(),
            (queryResult, table1Element) =>
            {
                List<Tuple<T1, T2>> combination =
                    Reduce
                    (
                    table2,
                    new List<Tuple<T1, T2>>(),
                    (combi, table2Element) =>...
                    );
                queryResult.AddRange(combination);
                return queryResult;
            });
}
```

Het idee - map - reduce - join - code - opdrachten

MapReduce

Reduce == For loop

Join == Reduce and Reduce

Join == Reduce * Reduce (1 and 1, 1 and 0)

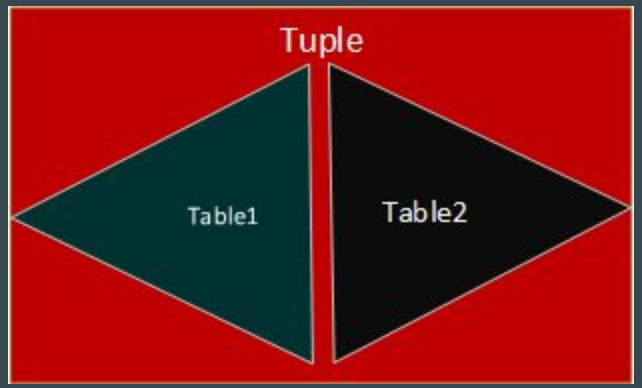
Join == For loop * For loop

```
public static IEnumerable<Tuple<T1, T2>> Join<T1, T2>(this IEnumerable<T1> table1, IEnumerable<T2> table2,
                                                      Func<T1, T2, bool> condition)
{
    return  Reduce
            (
            table1,
            new List<Tuple<T1, T2>>(),
            (queryResult, table1Element) =>
            {
                List<Tuple<T1, T2>> combination =
                    Reduce
                    (
                    table2,
                    new List<Tuple<T1, T2>>(),
                    (combi, table2Element) =>
                    {
                        Tuple<T1, T2> row = new Tuple<T1, T2>(table1Element, table2Element);
                        if (condition(table1Element, table2Element))
                        {
                            combi.Add(row);
                        }

                        return combi;
                    }
                    );
                queryResult.AddRange(combination);
                return queryResult;
            });
}
```

MapReduce

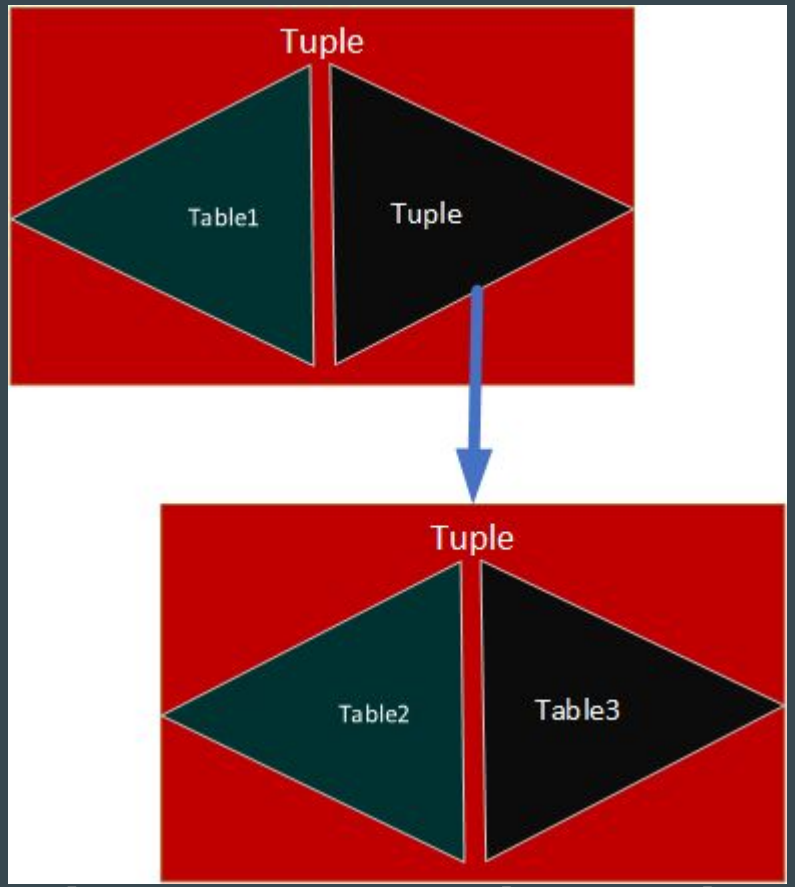
Single join (2 tabellen)



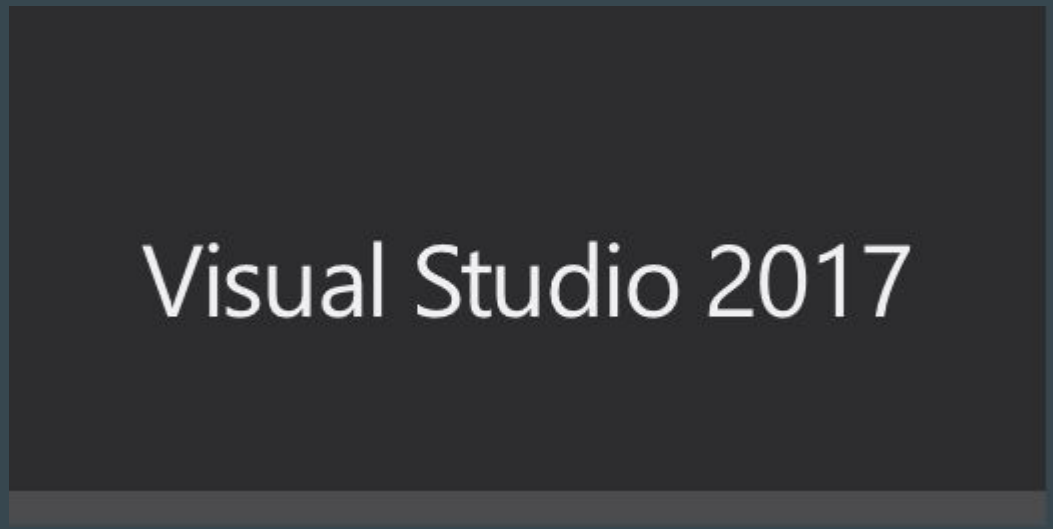
Het idee - map - reduce - join - code - opdrachten

MapReduce

Double join (3 tabellen)

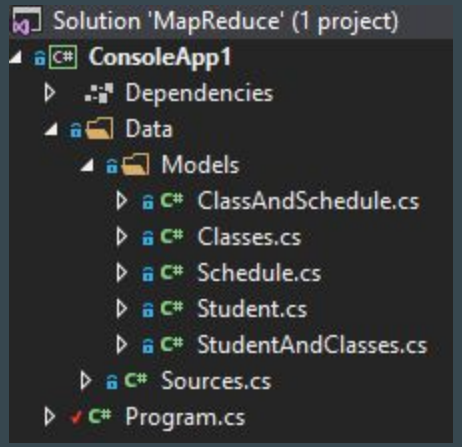


MapReduce



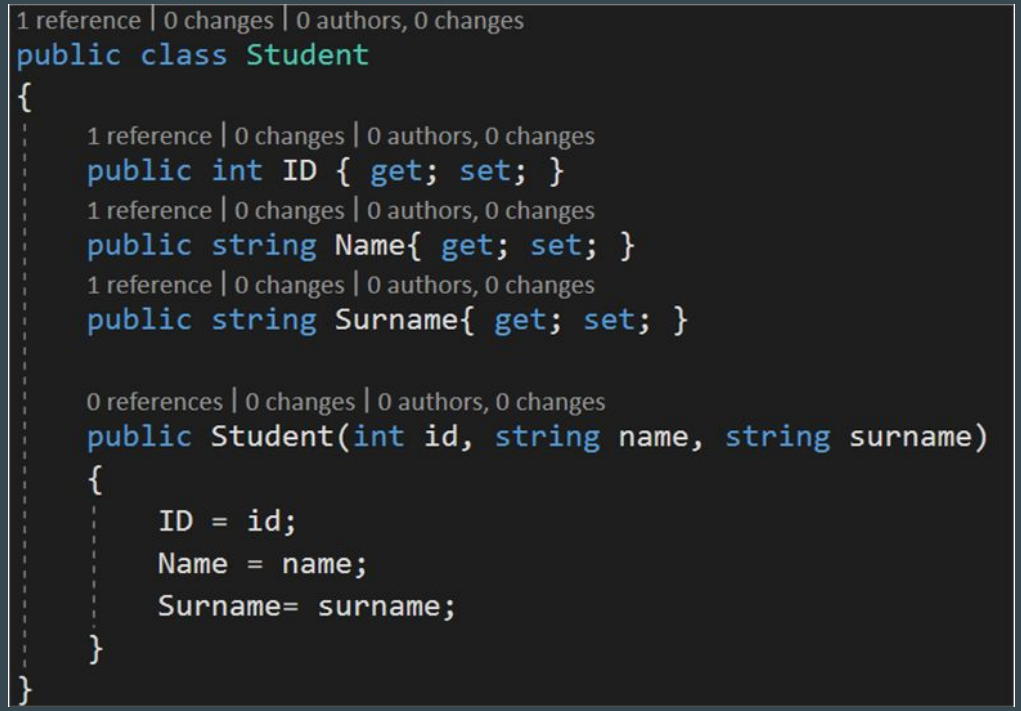
Het idee - map - reduce - join - code - opdrachten

MapReduce



Het idee - map - reduce - join - code - opdrachten

MapReduce

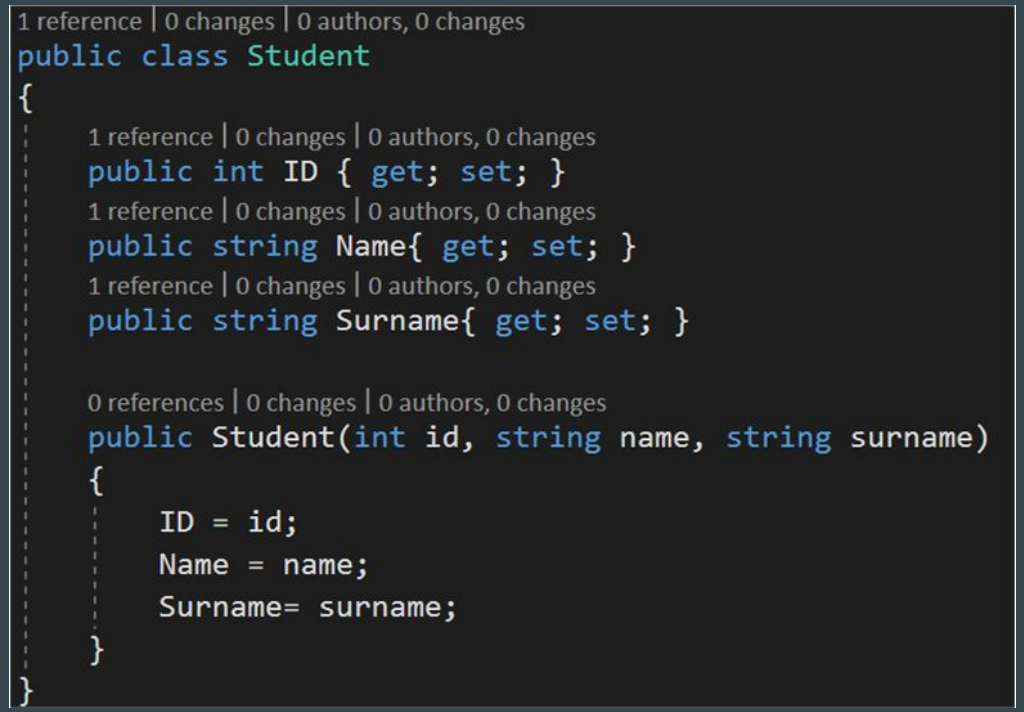


```
1 reference | 0 changes | 0 authors, 0 changes
public class Student
{
    1 reference | 0 changes | 0 authors, 0 changes
    public int ID { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Name{ get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Surname{ get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    public Student(int id, string name, string surname)
    {
        ID = id;
        Name = name;
        Surname= surname;
    }
}
```

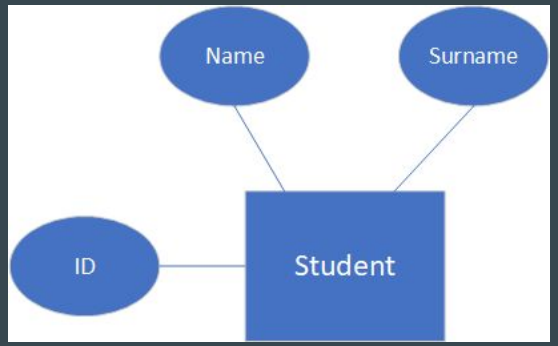
Het idee - map - reduce - join - code - opdrachten

MapReduce

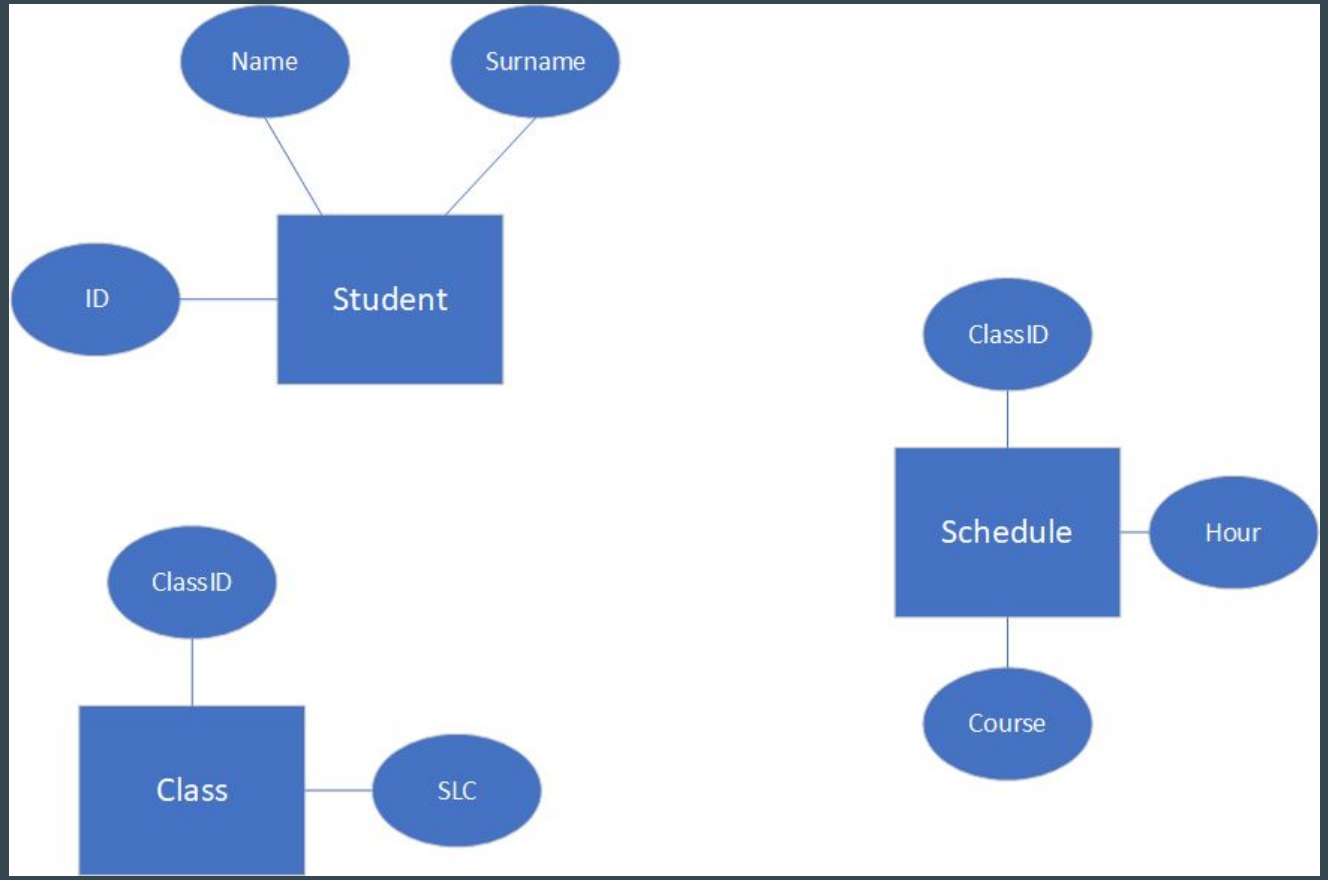


```
1 reference | 0 changes | 0 authors, 0 changes
public class Student
{
    1 reference | 0 changes | 0 authors, 0 changes
    public int ID { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Name{ get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Surname{ get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    public Student(int id, string name, string surname)
    {
        ID = id;
        Name = name;
        Surname= surname;
    }
}
```

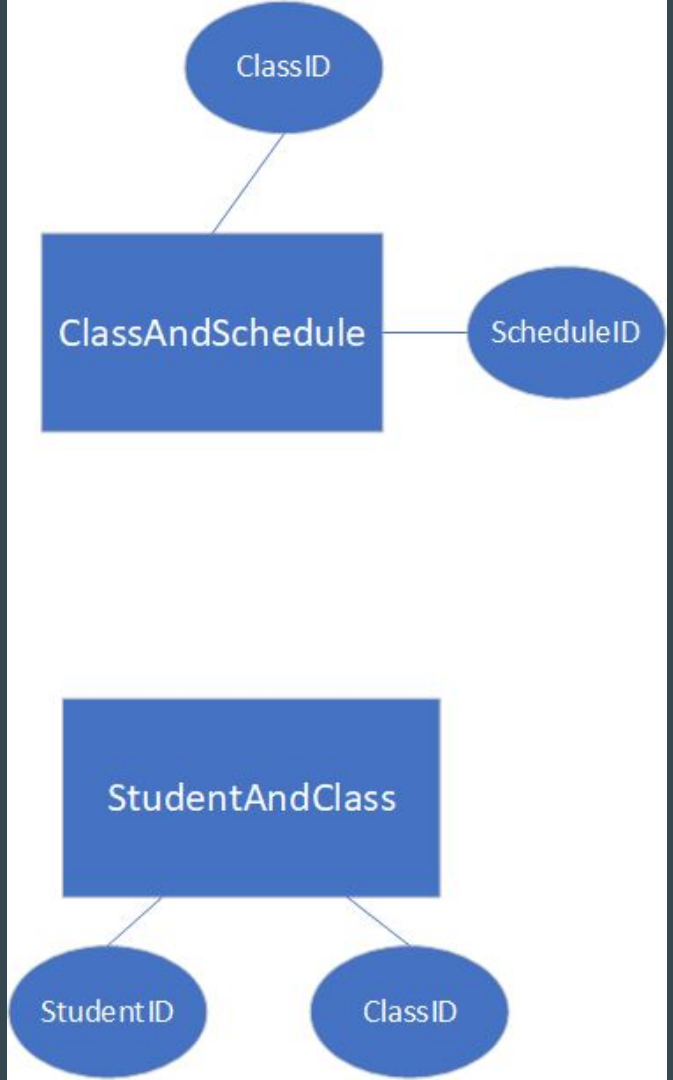


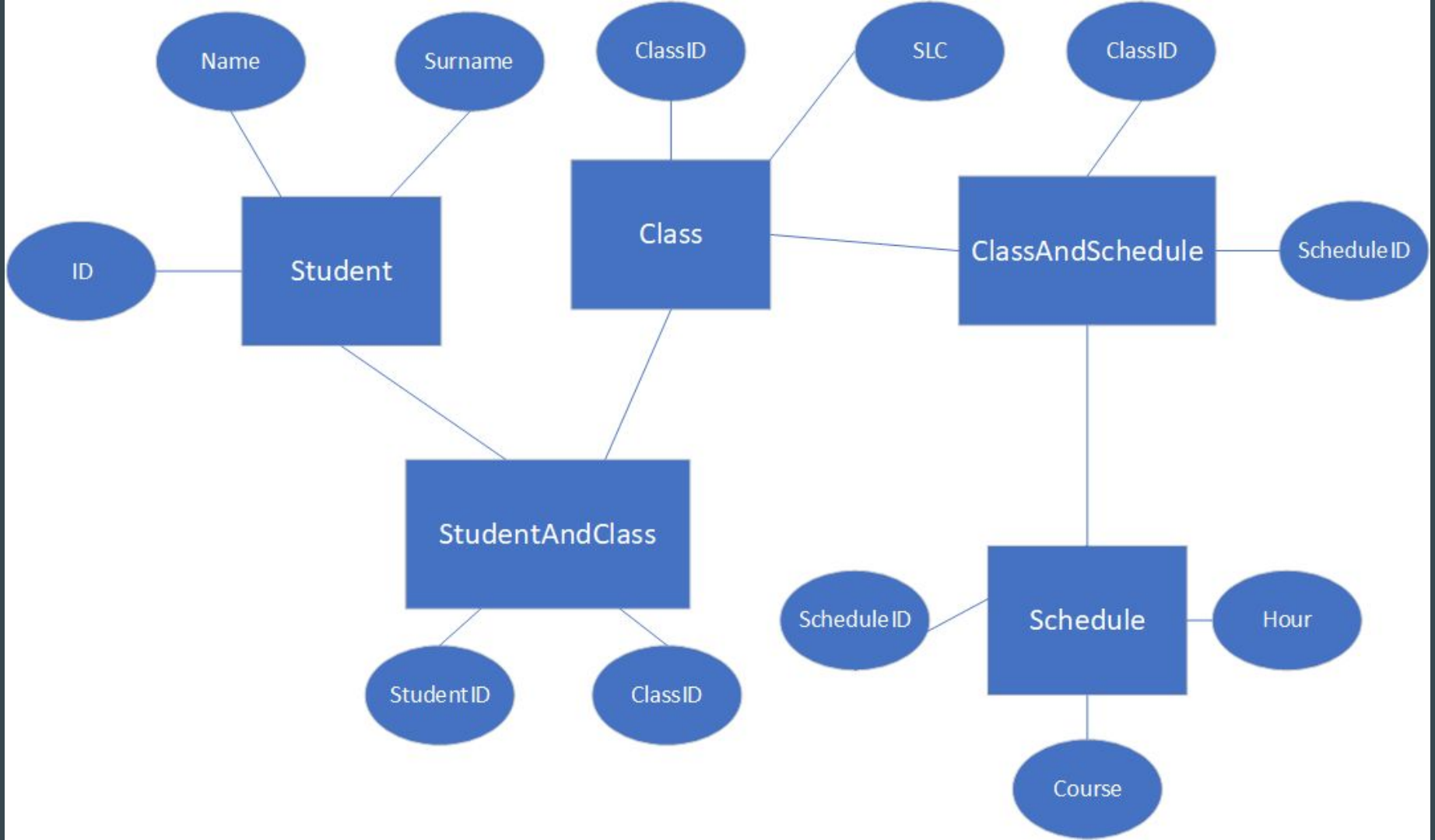
Het idee - map - reduce - join - code - opdrachten

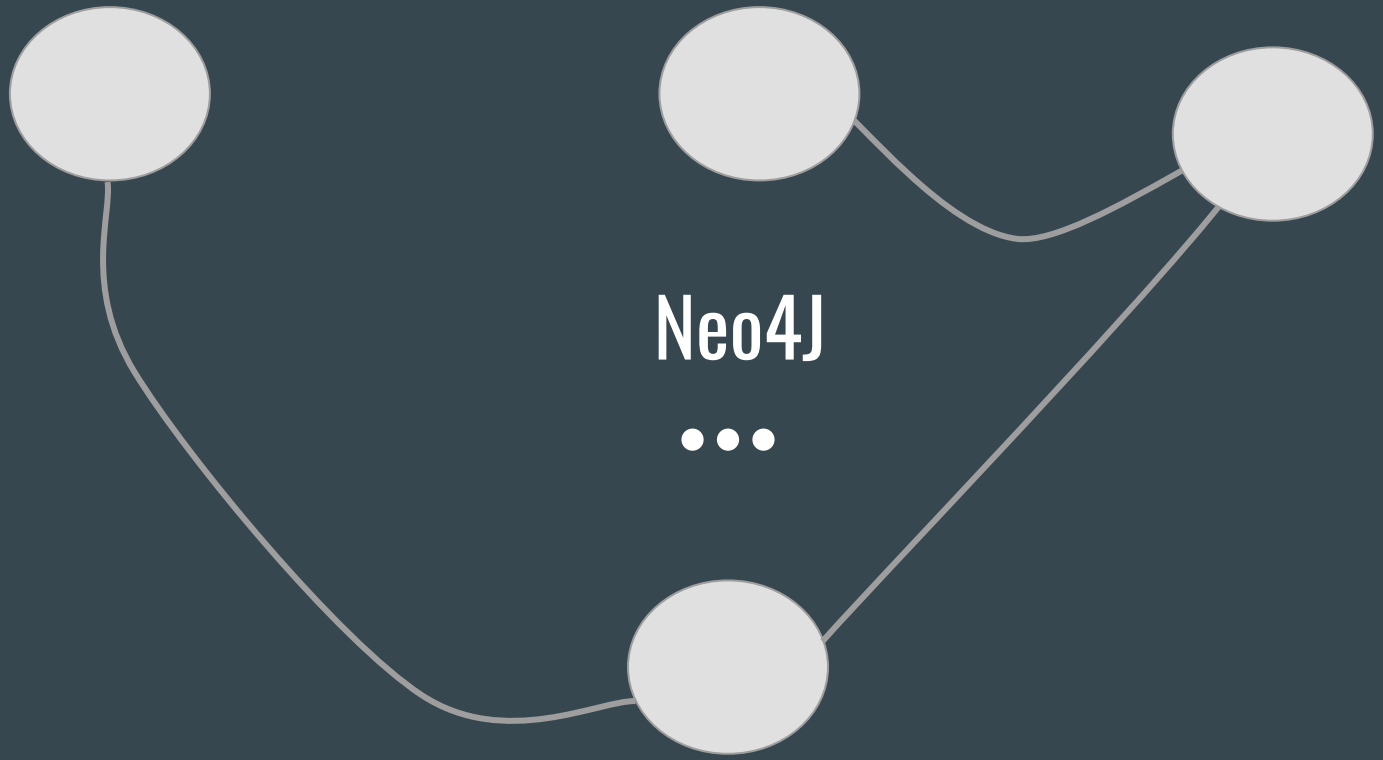


Het idee - map - reduce - join - code - opdrachten

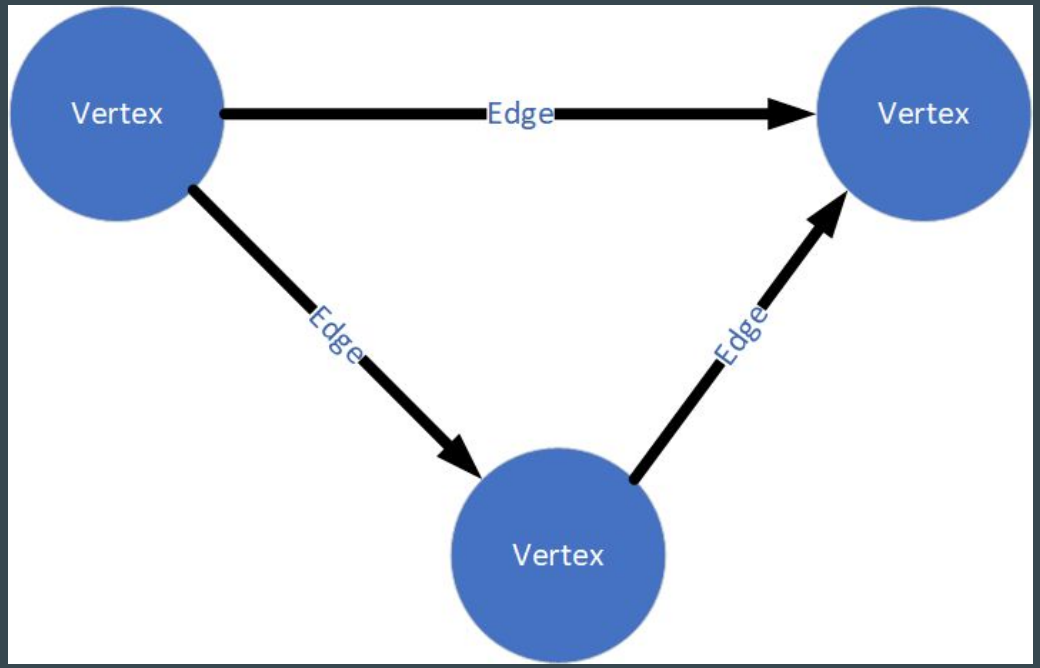
MapReduce


ClassID
ClassAndSchedule
ScheduleID
StudentAndClass
StudentID
ClassID


Name
Surname
ClassID
SLC
ClassID
ID
Student
Class
ClassAndSchedule
ScheduleID
StudentAndClass
StudentID
ClassID
ScheduleID
Schedule
Hour
Course

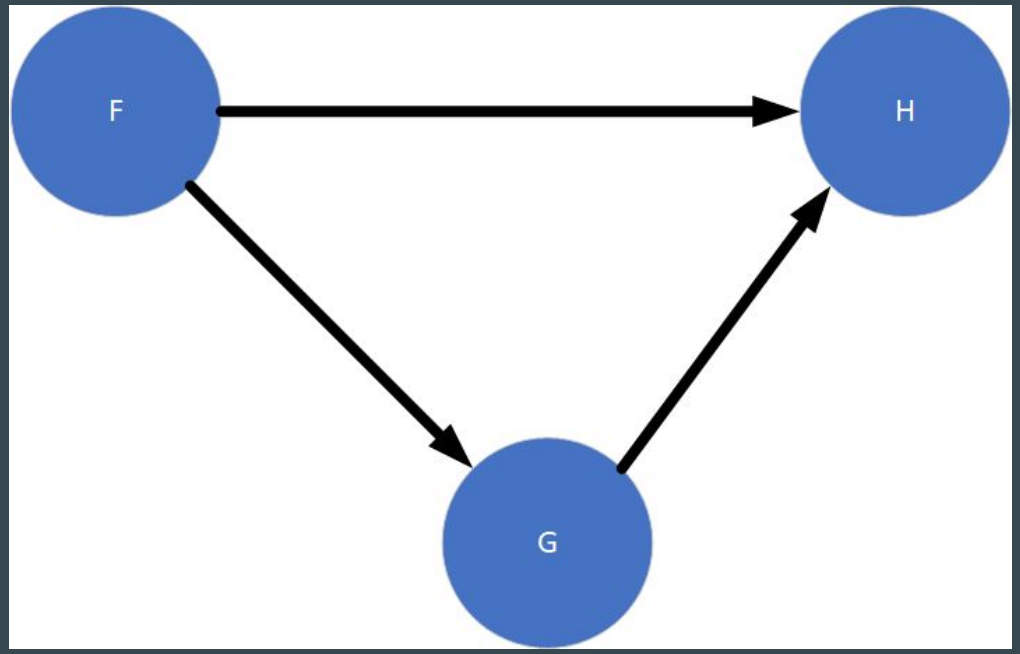

Neo4J
...

Neo4J



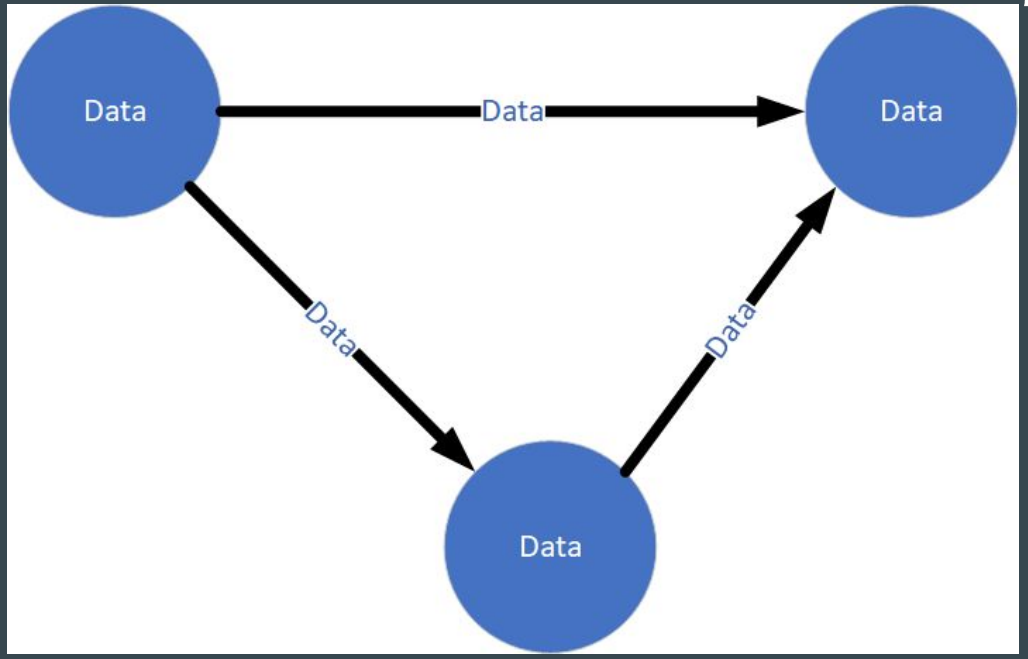
Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J



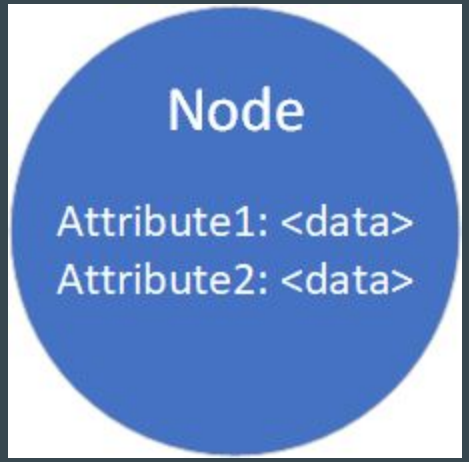
Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J



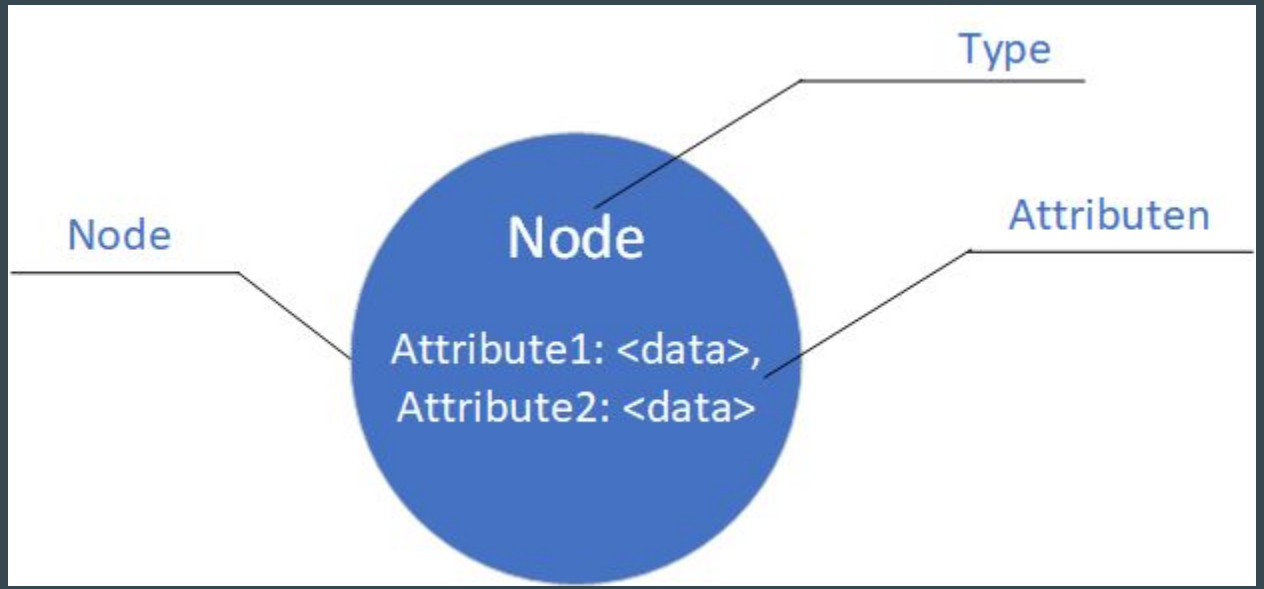
Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J



Graphs - Nodes - Relations - Cypher - Code - Opdracht

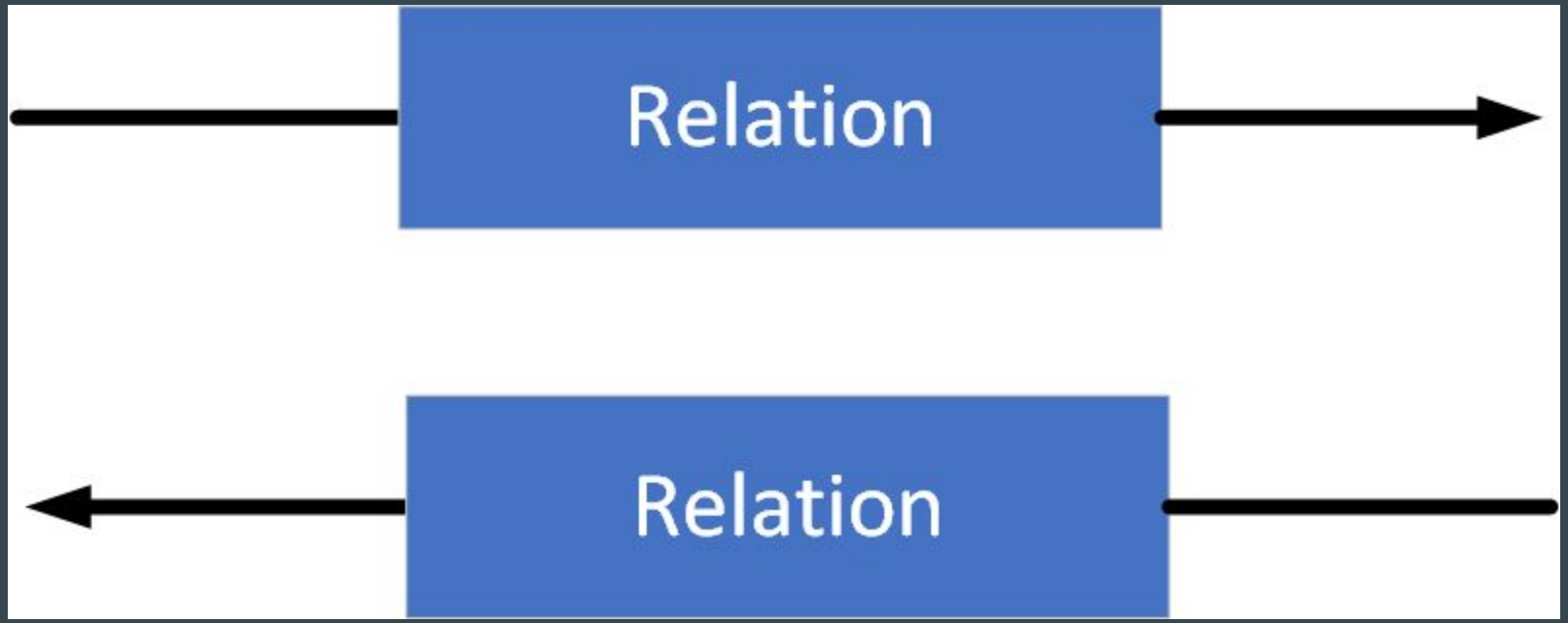
Neo4J



Graphs - **Nodes** - Relations - Cypher - Code - Opdracht

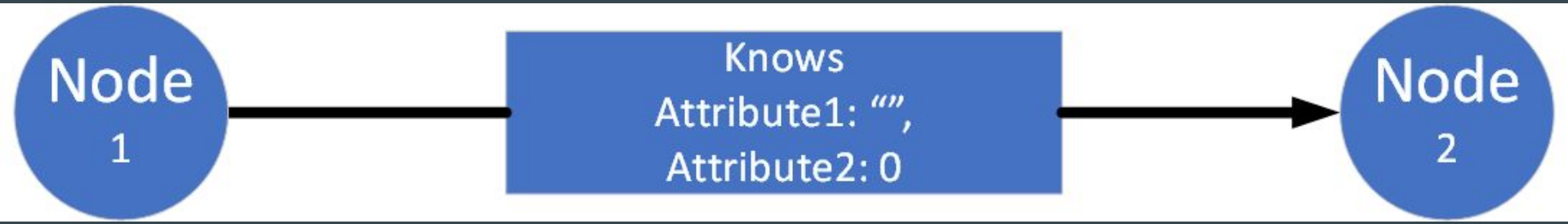

Relation

Neo4J



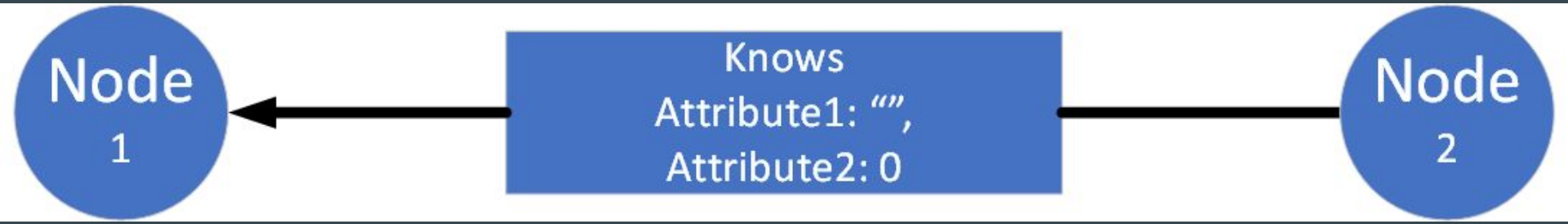
Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J



Graphs - Nodes - Relations - Cypher - Code - Opdracht

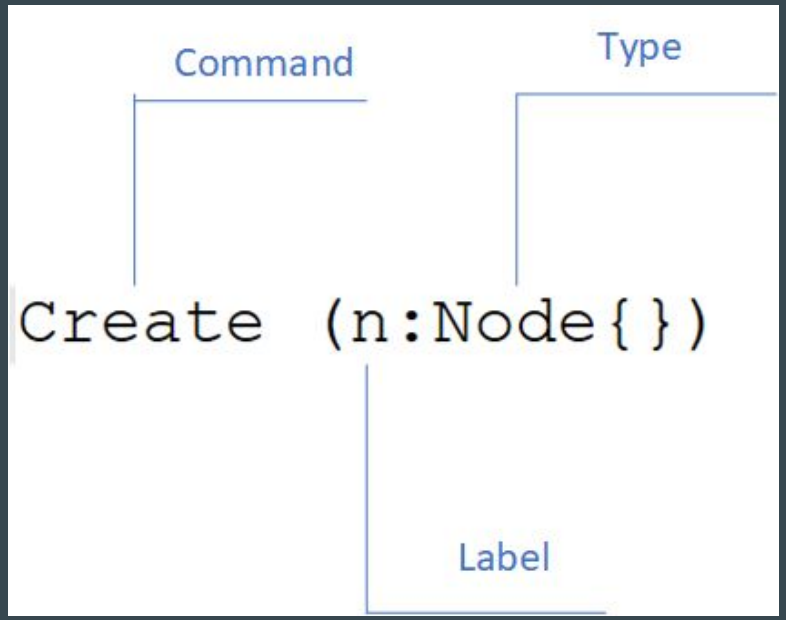
Neo4J

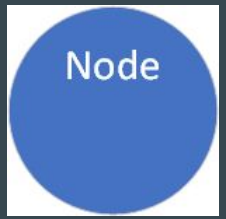


Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J

Het maken van een node:

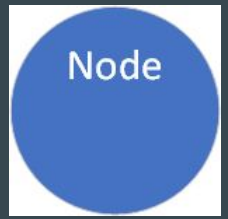




Neo4J

Het maken van een node:

```
Create (n:Node{})
```



Neo4J

Het maken van een node met data:

```
Create (n:Node{attribute1:"", attribute2: 2})
```

Neo4J

Het maken van een node met data:

```
Create(
        n:Node
            {
                attribute1:"",
                attribute2: 2
            }
    )
```

Neo4J

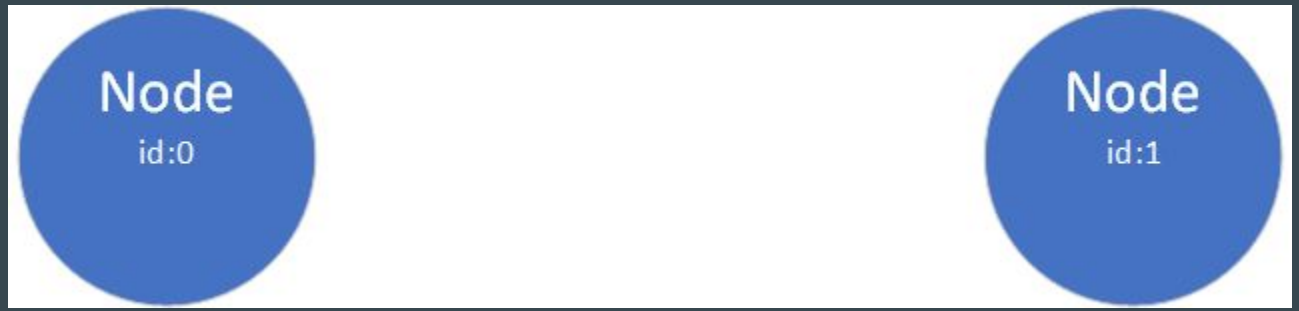
Het maken van nodes en een relatie:

```
Create (n:Node{id:0})-[r:Relation{}]->(m:Node{id:1})
```

Neo4J

Het maken van nodes en een relatie:

```
Create (n:Node{id:0})
Create (n:Node{id:1})
....
```



Neo4J

Het maken van nodes en een relatie:

```
Match (n:Node{id:0})
Match (m:Node{id:1})
Create (n)-[:Relation{}]->(m)
```

Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J

Het vinden van nodes met een relatie:

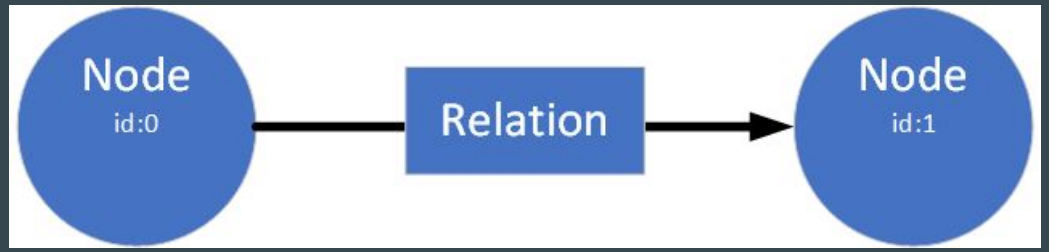
```
Match (n:Node{id:0})-[:Relation]->(m)
```

Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J

Het vinden van nodes met een relatie en returnen van een Node:

```
Match (n:Node{id:0})-[:Relation]->(m)
Return m
```

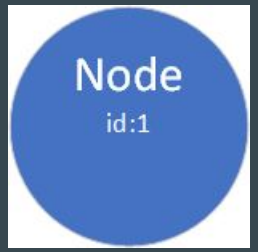


Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J

Het vinden van nodes met een relatie en returnen van een Node:

```
Match (n:Node{id:0})-[:Relation]->(m)
Return m
```



Neo4J

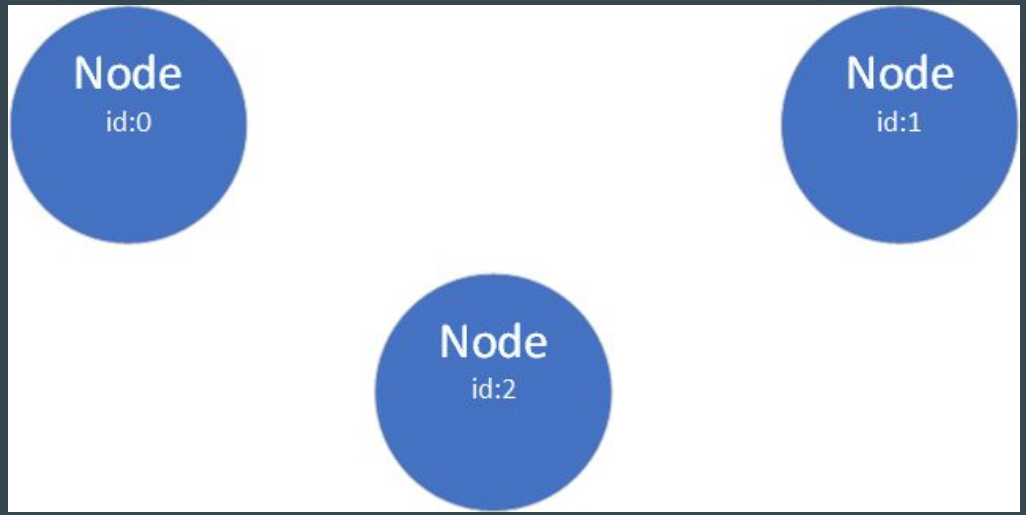
Het vinden van nodes en returnen van individuele waardes.

```
Match (n:Node{id:0})-[:Relation]->(m)
Return m.id
```

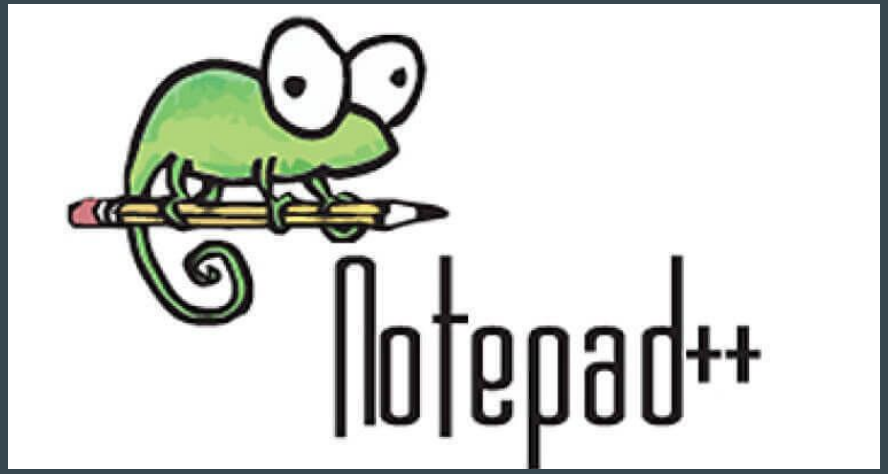
Neo4J

Het vinden van nodes en toepassen van where:

```
Match (n:Node)
Where n.id < 3
Return n
```

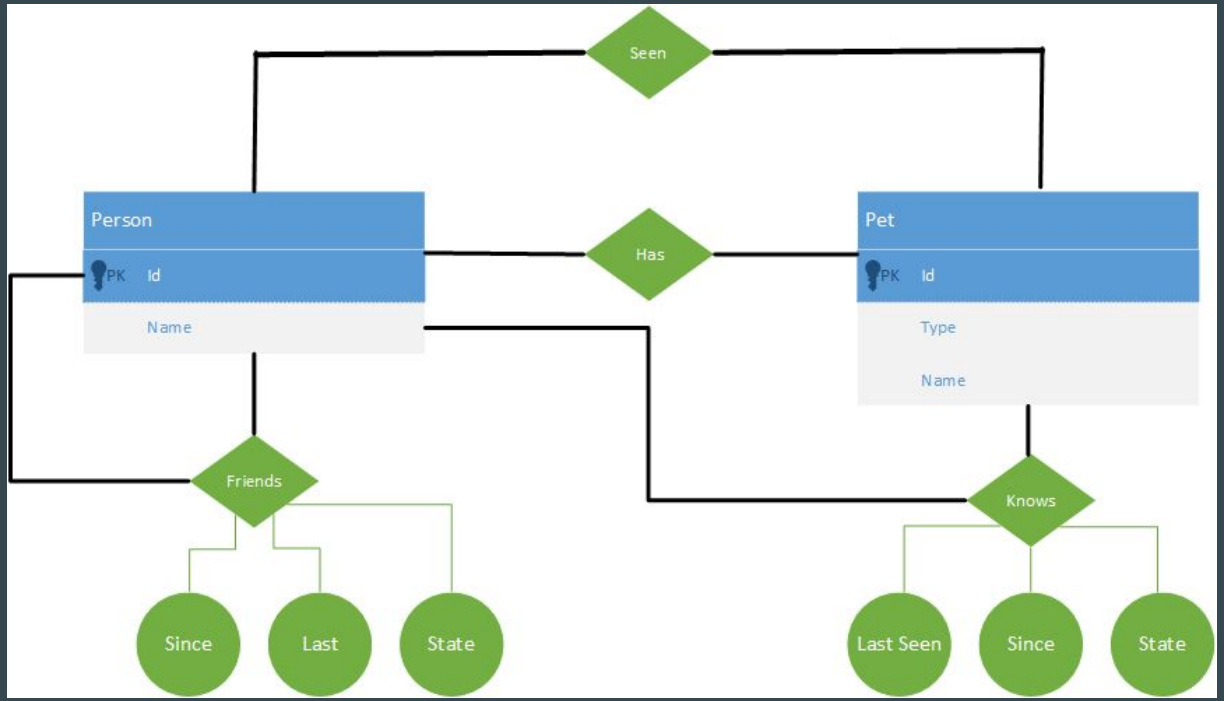


Neo4J



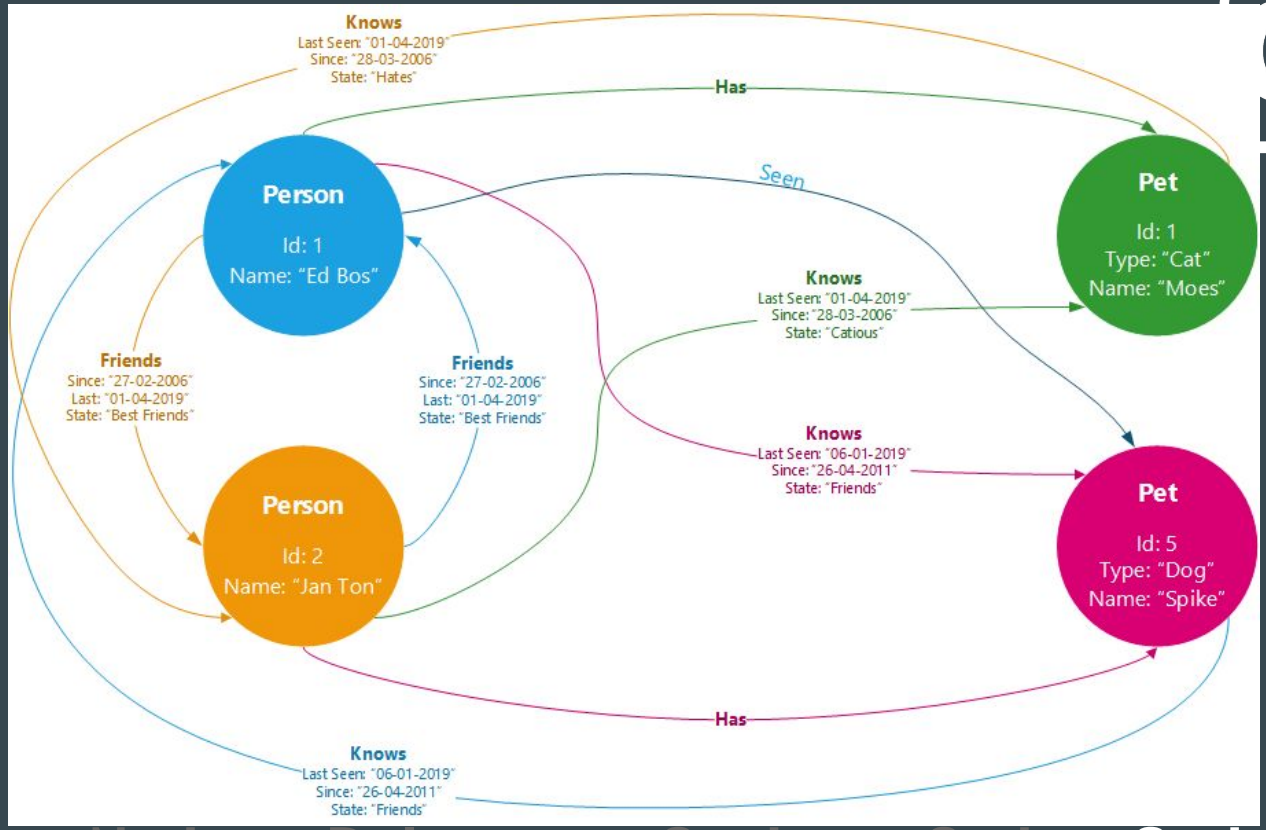
Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J



Graphs - Nodes - Relations - Cypher - Code - **Opdracht**

Neo4J



Graphs - Nodes - Relations - Cypher - Code - Opdracht

Neo4J

Neo4j Sandbox

Graphs - Nodes - Relations - Cypher - Code - Opdracht

...

Vragen?