Course code : **CSE3009**

Course title : **No SQL Data Bases**

Module : **6**

Topic : **7**

Page Rank Algorithm

Objectives

This session will give the knowledge about

- Graph Algorithms Introduction
- Page Rank computation using Iterative Processing
- Page Rank computation using Random Walk

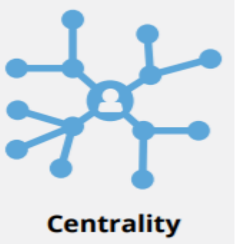
Introduction to Graph Algorithms

For transactions and operational decisions, you need real-time graph analysis to provide a local view of relationships between specific data items and take action.

To make discoveries about the overall nature of networks and model the behavior of complex systems, you need graph algorithms.

Graph algorithms provide the means to understand, model and predict complicated dynamics such as the flow of resources or information, the pathways through which contagions or network failures spread, and the influences on and resiliency of groups.

Type of Graph Algorithms

Algorithm Type	Graph Problem	Example
Pathfinding & Search	Find the optimal path or evaluate route availability and quality	• Find the quickest route to travel from A to B • Telephone call routing
Community Detection	Determine the importance of distinct nodes in the networks	• Determine social media influencers • Find likely attack targets in communication and transportation networks
 Centrality	Evaluate how a group is clustered or partitioned	• Segment customers • Find potential members of a fraud ring

Introduction to Graph Algorithms

Traversal & Pathfinding Algorithms

1. Parallel Breadth-First Search (BFS)
2. Parallel Depth-First Search (DFS)
3. Single-Source Shortest Path
4. All-Pairs Shortest Path
5. Minimum Weight Spanning Tree (MWST)

Centrality Algorithms

1. PageRank
2. Degree Centrality
3. Closeness Centrality
4. Betweenness Centrality

Introduction to Graph Algorithms

Community Detection Algorithms

1. Label Propagation
2. Strongly Connected
3. Union-Find / Connected Components / Weakly Connected
4. Louvain Modularity
5. Local Clustering Coefficient / Node Clustering Coefficient
6. Triangle-Count and Average Clustering Coefficient

Page Rank Algorithm

PageRank is an algorithm that measures the transitive, or directional, influence of nodes.

All other centrality algorithms measure the direct influence of a node, whereas PageRank considers the influence of your neighbors and their neighbors.

> For example, having a few influential friends could raise your PageRank more than just having a lot of low-influence friends.

PageRank is named after Google co-founder Larry Page, and is used to rank websites in Google’s search results. It counts the number, and quality, of links to a page which determines an estimation of how important the page is.

Page Rank - when to use

Personalized PageRank is used by Twitter to present users with recommendations of other accounts that they may wish to follow.

PageRank has been used to rank public spaces or streets, predicting traffic flow and human movement in these areas. The algorithm is run over a graph which contains intersections connected by roads, where the PageRank score reflects the tendency of people to park, or end their journey, on each street.

PageRank can be used as part of an anomaly or fraud detection system in the healthcare and insurance industries. It can help find doctors or providers that are behaving in an unusual manner, and then feed the score into a machine learning algorithm.

Page Rank – when NOT to use

There are some things to be aware of when using the PageRank algorithm:

- If there are no links from within a group of pages to outside of the group, then the group is considered a spider trap.
- Rank sink can occur when a network of pages form an infinite cycle.
- Dead-ends occur when pages have no out-links. If a page contains a link to another page which has no out-links, the link would be known as a dangling link.

Page Rank Algorithm

PageRank is defined in the original Google paper as follows:

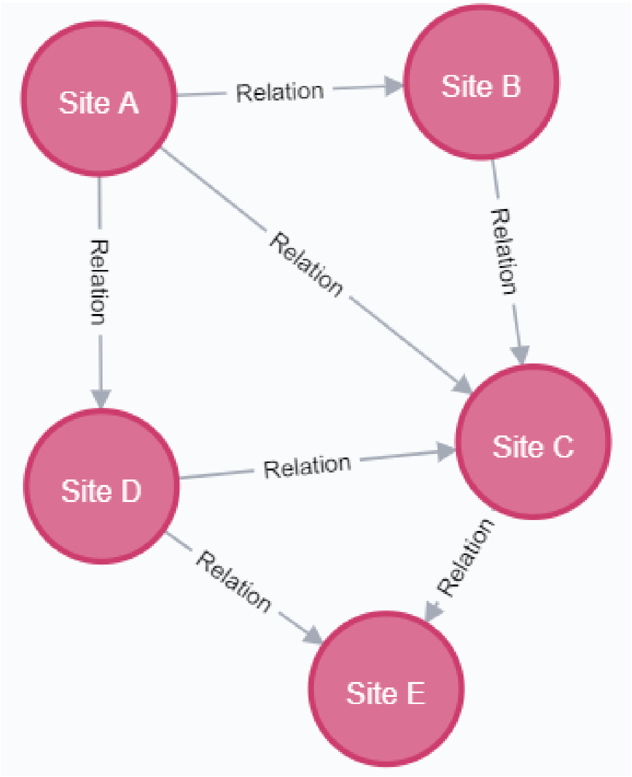
$$\text{PR(A)} = (1 - d) + d\left(\left(\frac{PR(T_1)}{C(T_1)}\right) + \cdots \ldots . + \left(\frac{PR(T_n)}{C(T_n)}\right)\right)$$

where,

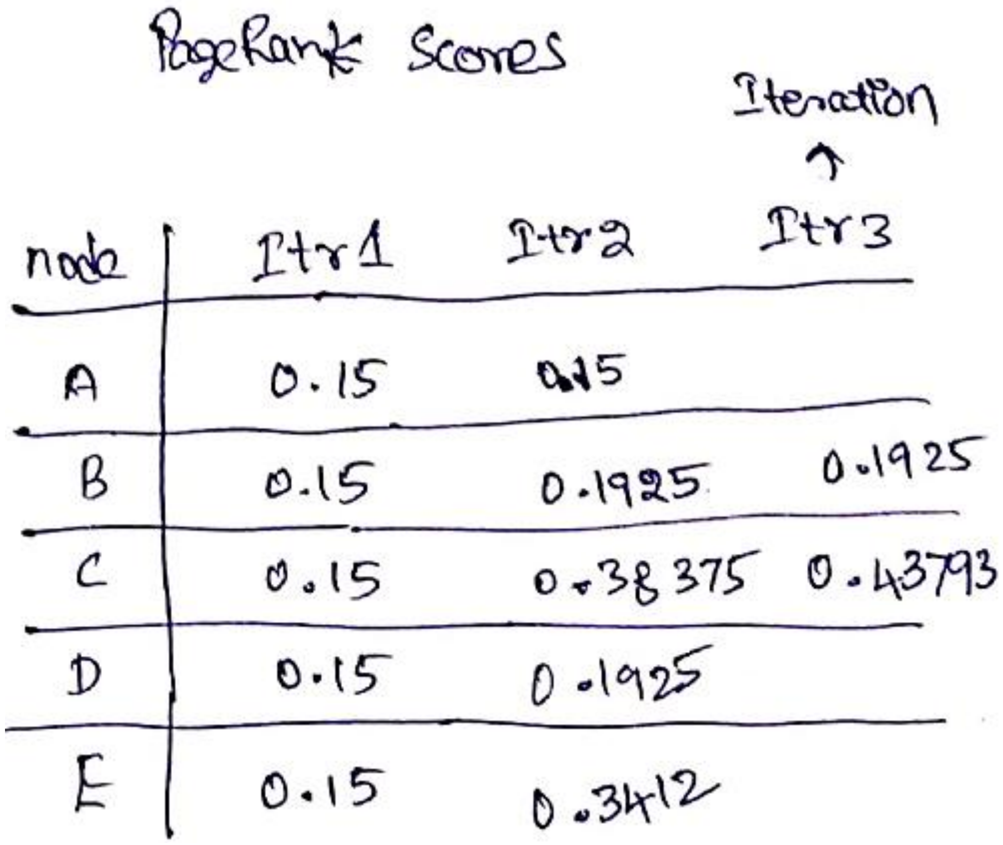
- we assume that a page A has pages T_1 to T_n which point to it (i.e., are citations).
- d is a damping factor which can be set between 0 and 1. It is usually set to 0.85.
- C(A) is defined as the number of links going out of page A.

PageRank computation : Example

Compute the page rank score for the given graph



PageRank computation : Example

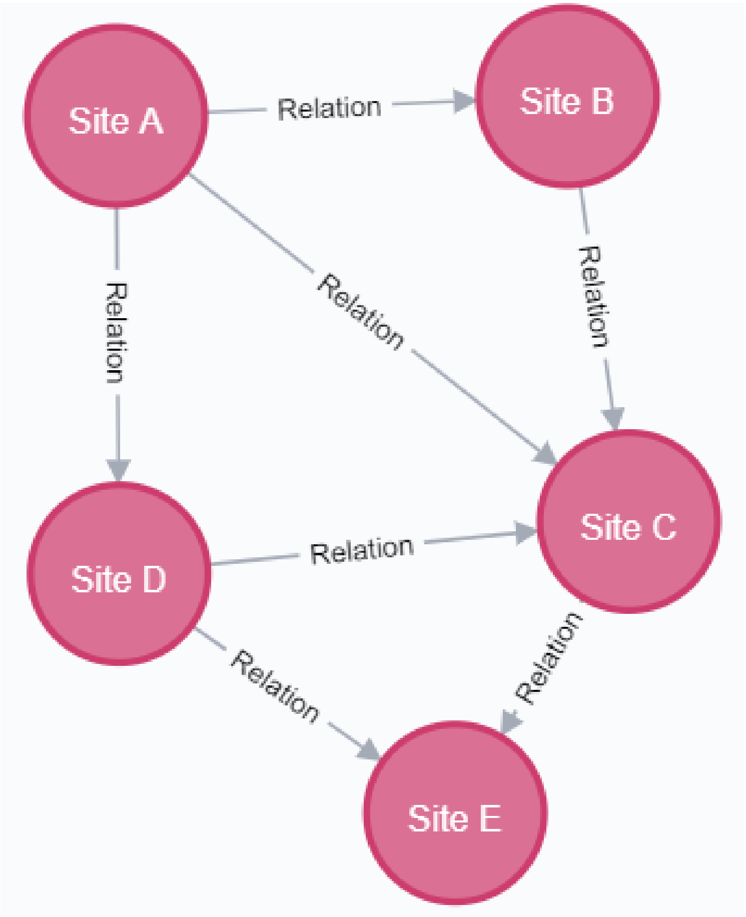


PageRank Scores

Iteration

↑

node	Itr1	Itr2	Itr3
A	0.15	0.15	
B	0.15	0.1925	0.1925
C	0.15	0.38375	0.43793
D	0.15	0.1925	
E	0.15	0.3412	



PageRank computation : Example

step1: Iteration 1

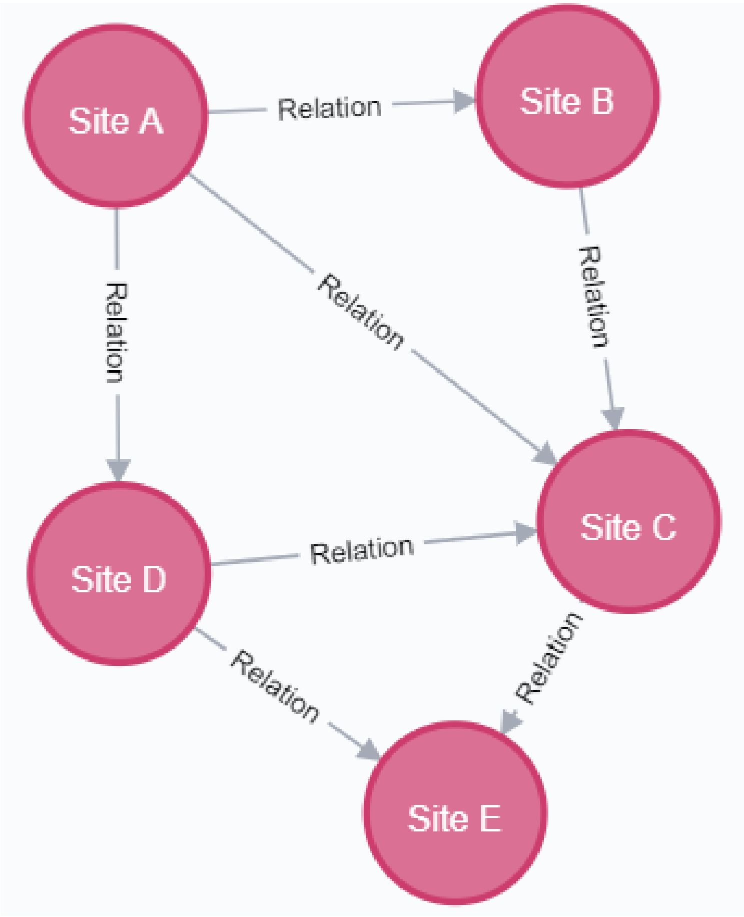
any Random node have to be choosed,

hence no need to follow any relationships.

so, the page rank score for all nodes in 1^{st} iteration is calculated as

$$PR(node) = (1-d), \text{ where } d = 0.85$$

$$= 1 - 0.85 = 0.15$$



PageRank computation : Example

Step2 : Iteration 2 (A)

$$PR(A) = (1-d) + d * \left(\frac{\overset{\text{Previous iteration}}{PR(\text{incomiding node 1})}}{\text{no. of outgoing links (incoming node 1)}} + \frac{PR(\text{incoming nod2})}{\text{no. of outgoing links (incoming node 2)}} \right)$$

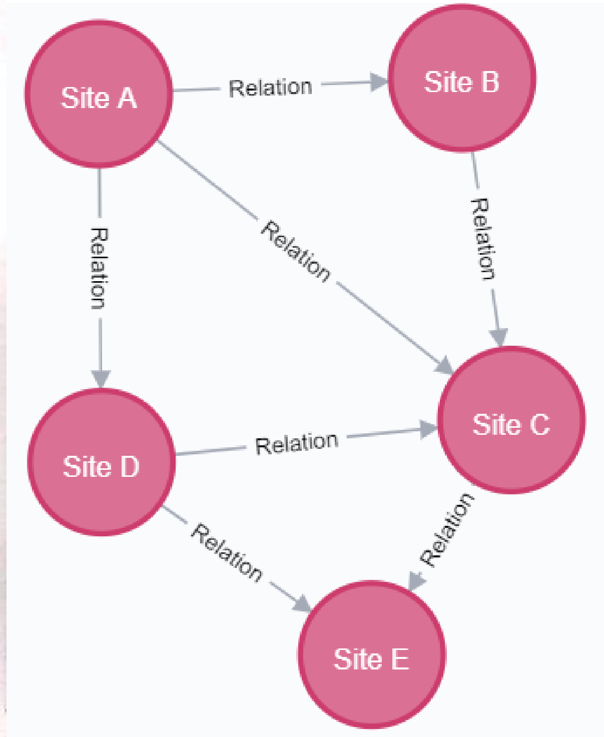
$$\therefore \; PR(A) = (1-d) + d * (\text{ no incoming nodes})$$

$$= 0.15 + d * 0 = 0.15$$

Iteration 2 (B)

$$PR(B) = (1-d) + d * \left(\frac{PR(A)}{3} \right)$$

$$= 0.15 + 0.85 * \left(\frac{0.15}{3} \right) = 0.1925$$



PageRank computation : Example

Iteration 2 (C)

$$PR(C) = (1-d) + d * \left(\frac{PR(A)}{3} + \frac{PR(B)}{1} + \frac{PR(D)}{2} \right)$$

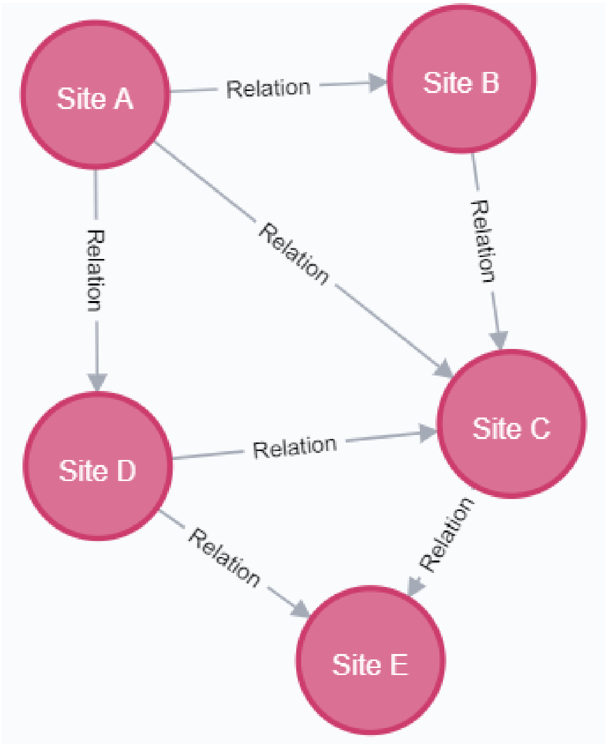
$$= 0.15 + 0.85 \left(\frac{0.15}{3} + \frac{0.15}{1} + \frac{0.15}{2} \right)$$

$$= 0.38375$$

Iteration 2 (D)

$$PR(D) = (1-d) + d * \left(\frac{PR(A)}{3} \right)$$

$$= 0.15 + 0.85 * \left(\frac{0.15}{3} \right) = 0.1925$$



PageRank computation : Example

Iteration 2 (E)

$$PR(E) = (1-d) + d * \left(\frac{PR(D)}{2} + \frac{PR(C)}{1} \right)$$

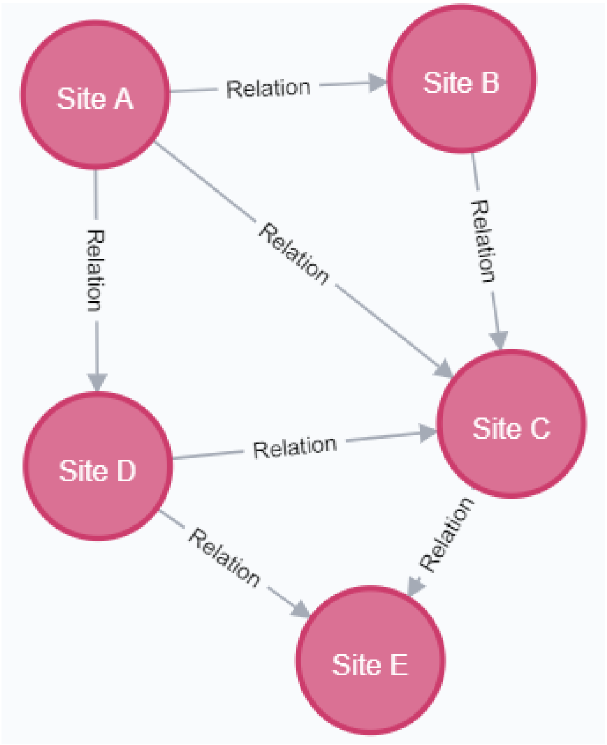
$$= 0.15 + 0.85 * \left(\frac{0.15}{2} + \frac{0.15}{1} \right)$$

$$= 0.3412$$

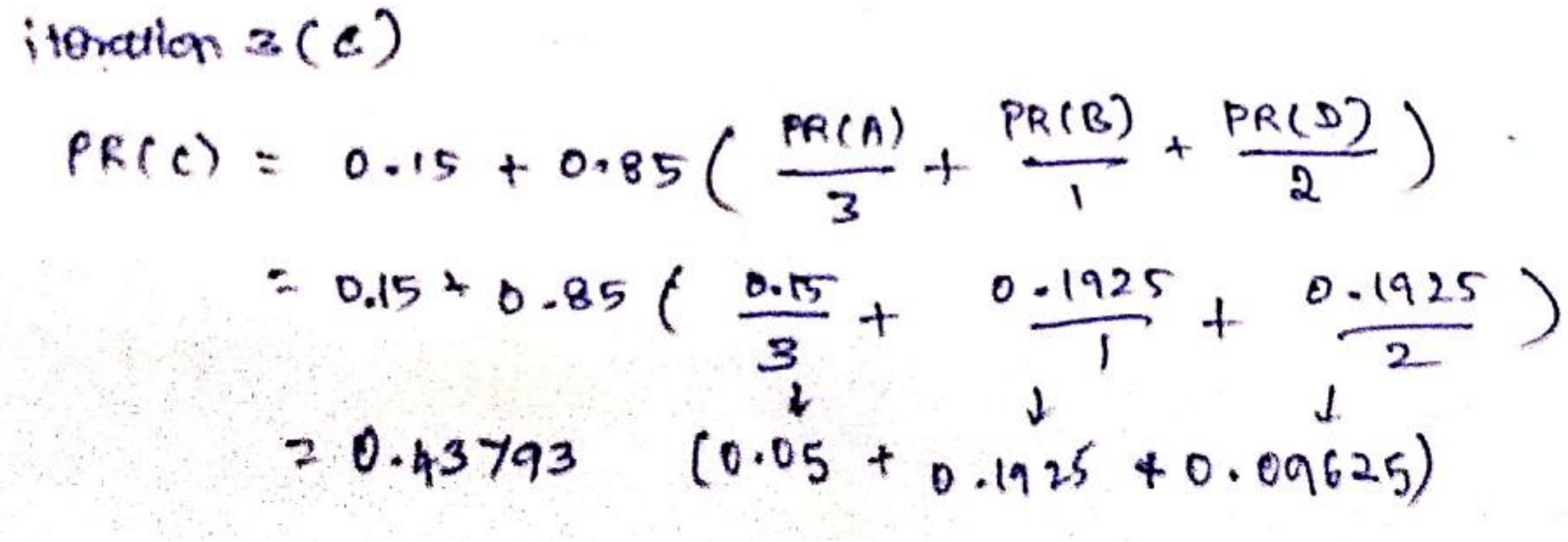
Step3 : iteration 3 (B)

$$PR(B) = (1-d) + d * \left(\frac{PR(A)}{3} \right)$$

$$= 0.15 + 0.85 * \frac{0.15}{3} = 0.1925$$



PageRank computation : Example

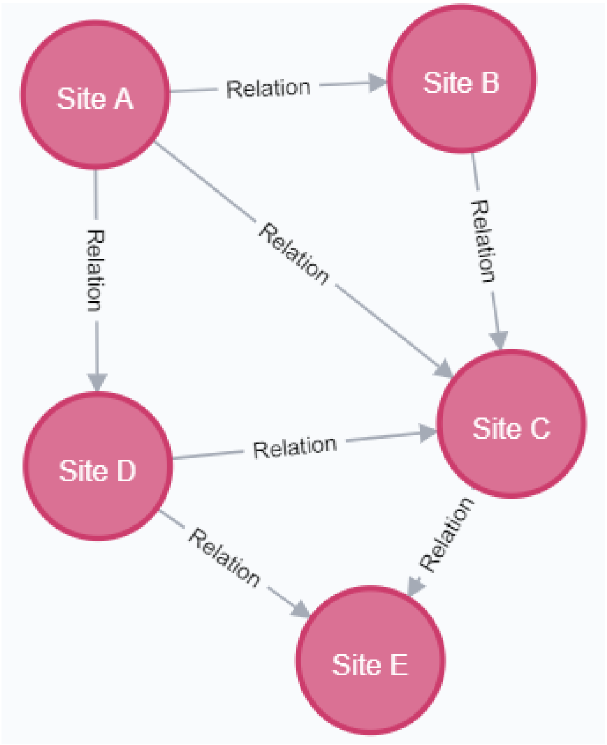


iteration 2 (c)

$$PR(C) = 0.15 + 0.85\left(\frac{PR(A)}{3} + \frac{PR(B)}{1} + \frac{PR(D)}{2}\right)$$

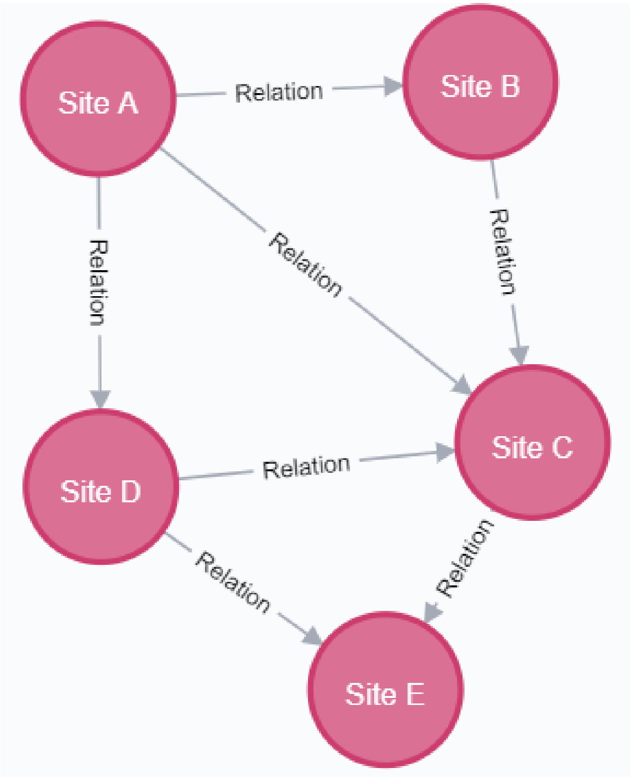
$$= 0.15 + 0.85\left(\underset{\downarrow}{\frac{0.15}{3}} + \underset{\downarrow}{\frac{0.1925}{1}} + \underset{\downarrow}{\frac{0.1925}{2}}\right)$$

$$= 0.43793 \quad (0.05 + 0.1925 + 0.09625)$$



PageRank computation in Neo4j

```
MERGE (a:Page {name:'Site A'})
MERGE (b:Page {name:'Site B'})
MERGE (c:Page {name:'Site C'})
MERGE (d:Page {name:'Site D'})
MERGE (e:Page {name:'Site E'})

MATCH (x:Page), (y:Page) WHERE x.name = "Site A"
AND y.name = "Site B"
CREATE (x)-[: Relation]->(y)
```



Return Clause

```
MATCH (x:Page), (y:Page) WHERE x.name = "Site A" AND y.name = "Site C"
CREATE (x)-[: Relation]->(y)

MATCH (x:Page), (y:Page) WHERE x.name = "Site A" AND y.name = "Site D"
CREATE (x)-[: Relation]->(y)

MATCH (x:Page), (y:Page) WHERE x.name = "Site B" AND y.name = "Site C"
CREATE (x)-[: Relation]->(y)
```

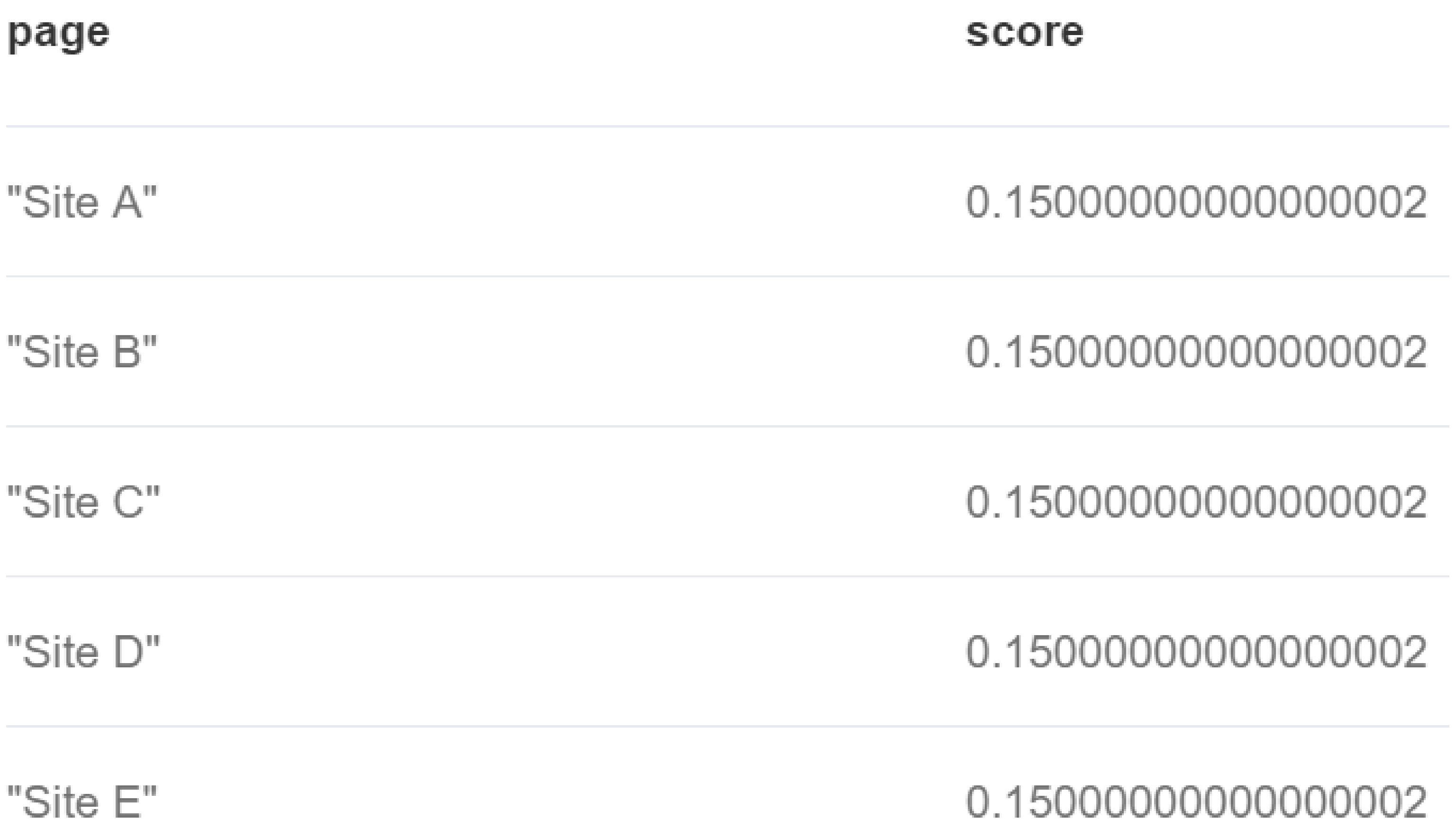
Return Clause

```
MATCH (x:Page), (y:Page) WHERE x.name = "Site C" AND y.name = "Site E"
CREATE (x)-[: Relation]->(y)

MATCH (x:Page), (y:Page) WHERE x.name = "Site D" AND y.name = "Site C"
CREATE (x)-[: Relation]->(y)

MATCH (x:Page), (y:Page) WHERE x.name = "Site D" AND y.name = "Site E"
CREATE (x)-[: Relation]->(y)
```

Neo4j command to apply PageRank Algorithm

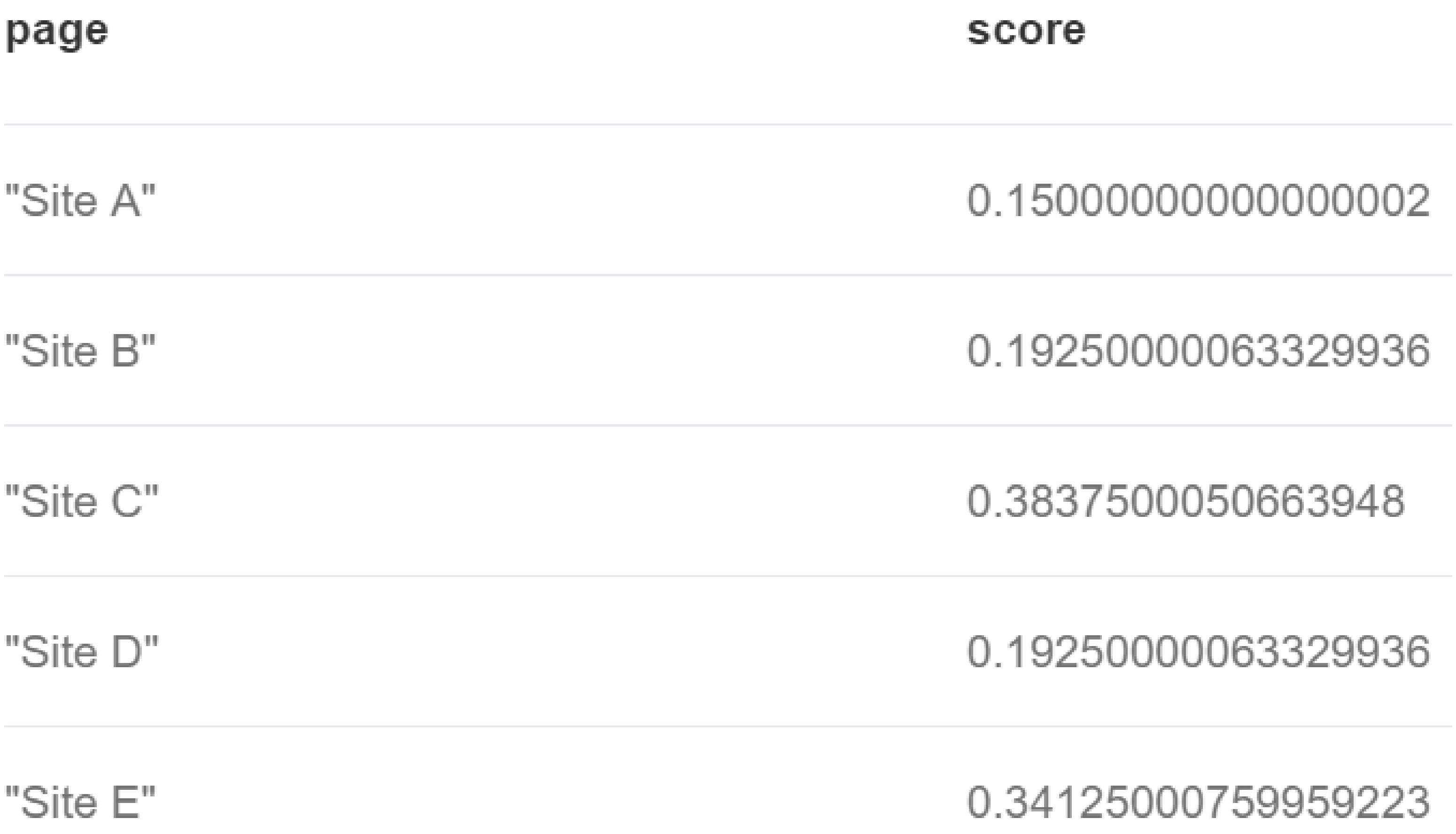
```
CALL algo.pageRank.stream('Page', 'Relation', {iterations:1,
dampingFactor:0.85})
YIELD nodeId, score

RETURN algo.asNode(nodeId).name AS page,score
ORDER BY score DESC
```

Result: 0th Iteration



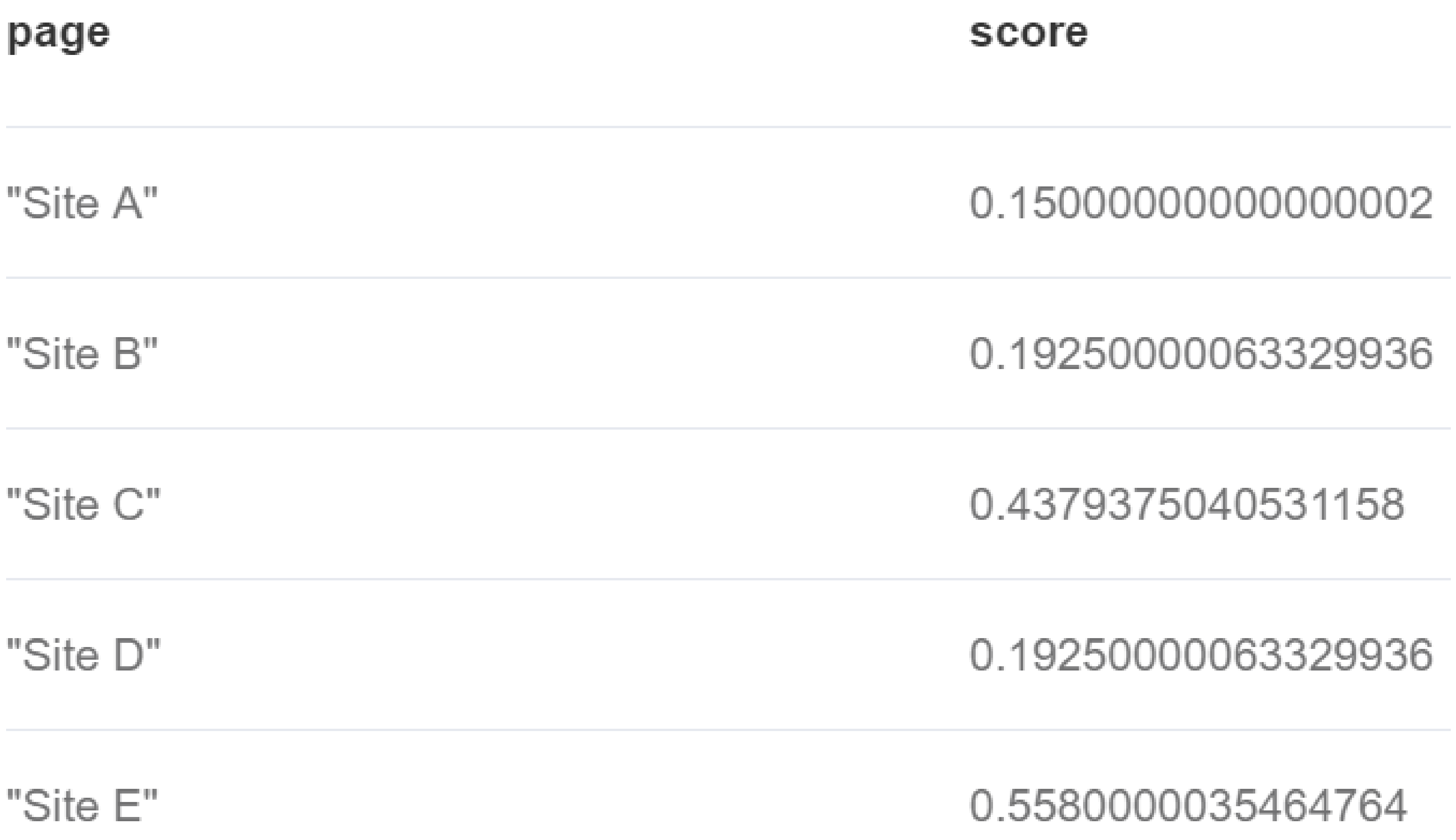
page	score
"Site A"	0.15000000000000002
"Site B"	0.15000000000000002
"Site C"	0.15000000000000002
"Site D"	0.15000000000000002
"Site E"	0.15000000000000002

Result: 1st Iteration



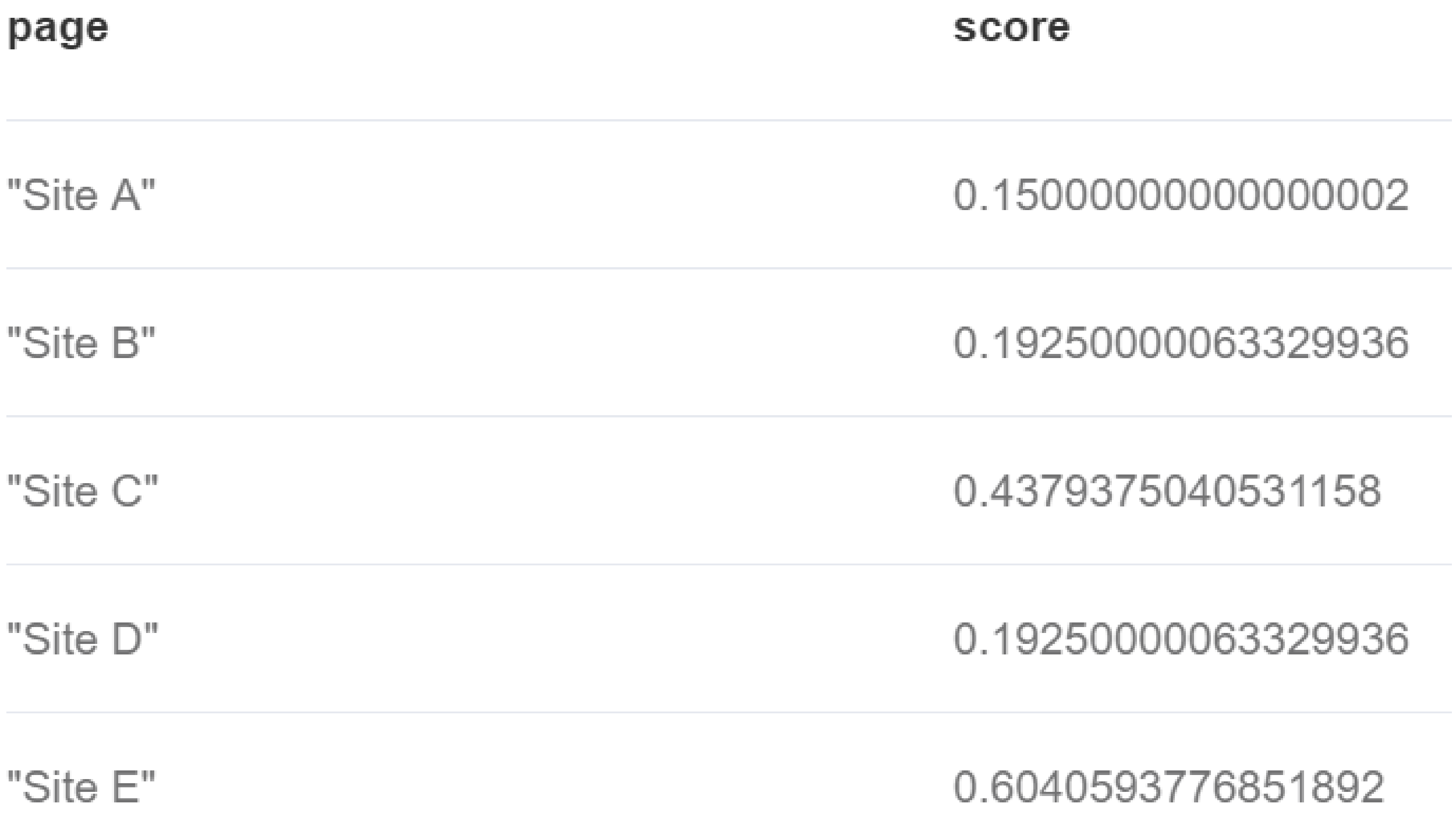
page	score
"Site A"	0.15000000000000002
"Site B"	0.19250000063329936
"Site C"	0.3837500050663948
"Site D"	0.19250000063329936
"Site E"	0.34125000759959223

Result: 2nd Iteration



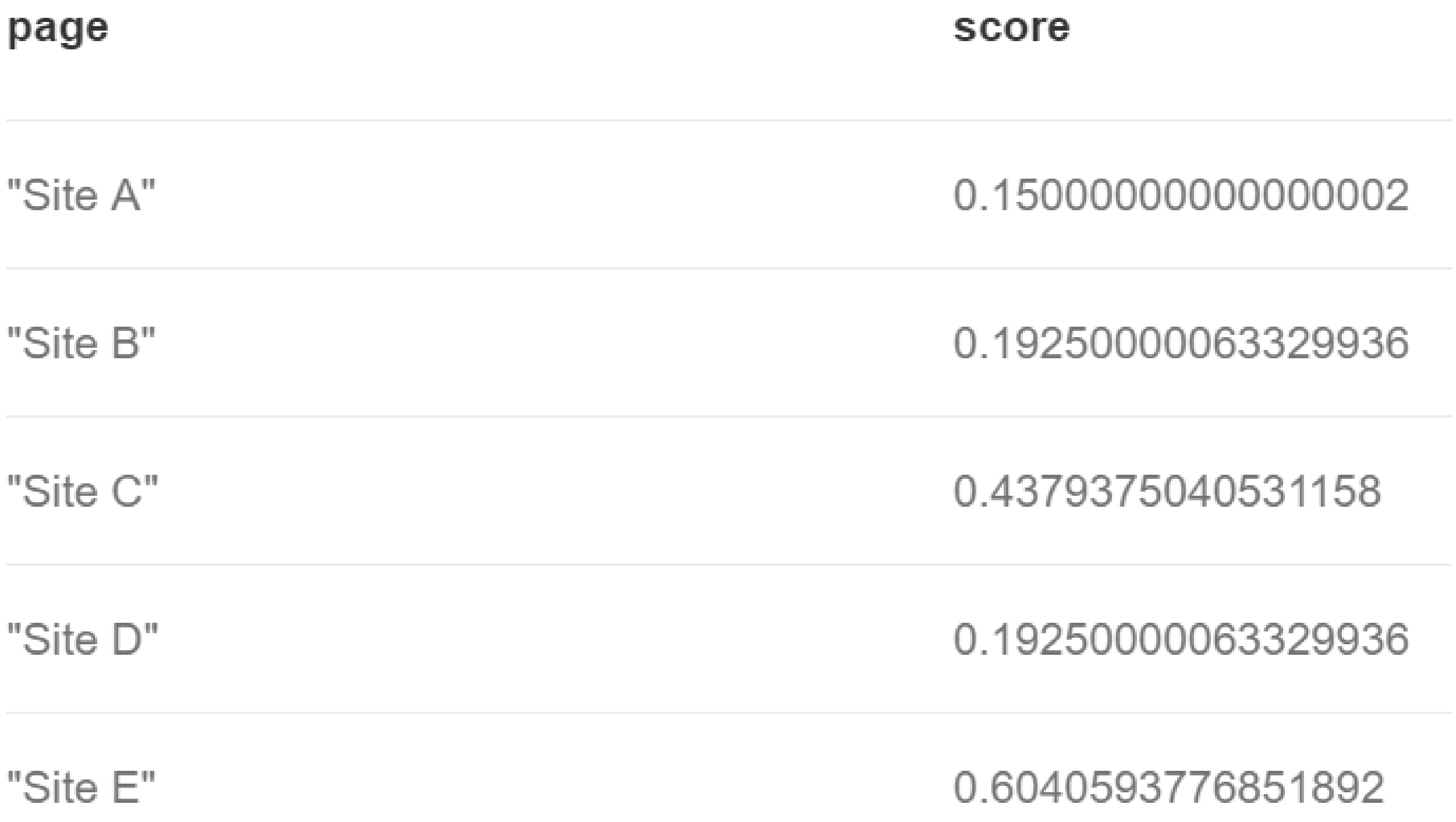
page	score
"Site A"	0.15000000000000002
"Site B"	0.19250000063329936
"Site C"	0.4379375040531158
"Site D"	0.19250000063329936
"Site E"	0.5580000035464764

Result: 3rd Iteration



page	score
"Site A"	0.15000000000000002
"Site B"	0.1925000063329936
"Site C"	0.4379375040531158
"Site D"	0.1925000063329936
"Site E"	0.6040593776851892

Result: 4th Iteration



page	score
"Site A"	0.15000000000000002
"Site B"	0.19250000063329936
"Site C"	0.4379375040531158
"Site D"	0.19250000063329936
"Site E"	0.6040593776851892

Random Walk: PageRank Algorithm

Random Walk is an algorithm that provides random paths in a graph.

A random walk means that we start at one node, choose a neighbor to navigate to at random or based on a provided probability distribution, and then do the same from that node, keeping the resulting path in a list.

The term "random walk" was first mentioned by Karl Pearson in 1905 in a letter to Nature magazine titled The Problem of the Random Walk.

When to use Random Walk

It has be shown to relate to Brownian motion and also to the movement and dispersal of animals in the study of Random walk models in biology.

It has been used to analyze ALSI index of the JSE stock exchange and show that the index followed the random walk hypothesis between years 2000 and 2011.

It can be used as part of the Walktrap and Infomap community detection algorithms. If a random walk returns a small set of nodes repeatedly, then it indicates that those set of nodes may have a community structure.

It can be used as part of the training process of machine learning model, as described in David Mack’s article Review prediction with Neo4j and TensorFlow.

When NOT to use Random Walk

Dead-ends occur when pages have no out-links. In this case, the random walk will abort and a path containing only the first first node will be returned. This problem can be avoided by passing the direction: BOTH parameter, so that the random walk will traverse relationships in both directions

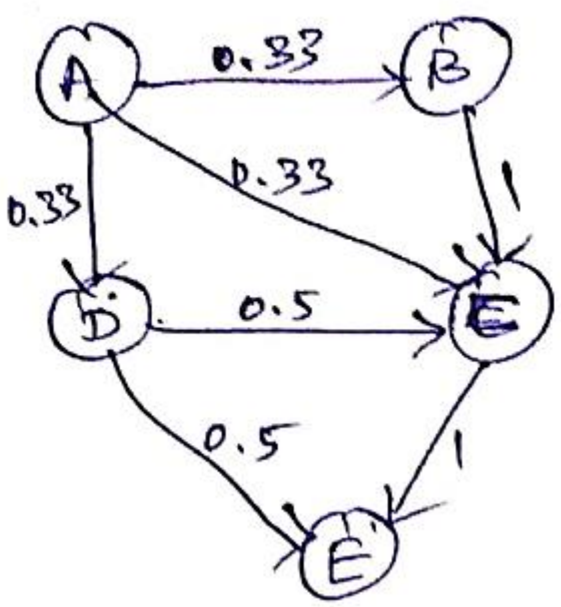
If there are no links from within a group of pages to outside of the group, then the group is considered a spider trap. Random walks starting from any of the nodes in that group will only traverse to the others in the group.

Neo4j implementation of the algorithm doesn’t allow a random walk to jump to non-neighbouring nodes.

Sinks can occur when a network of links form an infinite cycle.

Random Walk Example

The following graph finds the random walk page ranking for the example considered in iteration page rank model.



$PR(A) = 0$

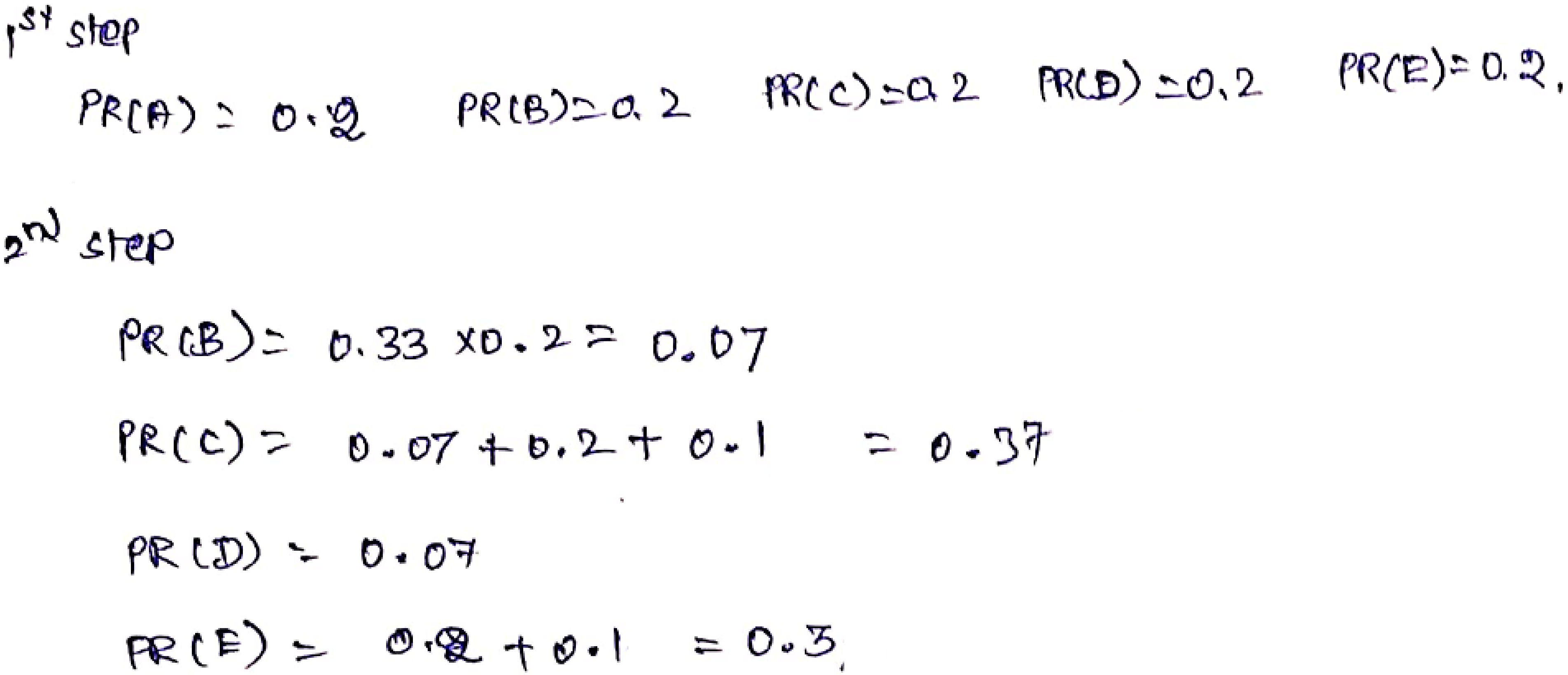
$PR(B) = 0.33\ PR(A)$

$PR(C) = 0.33\ PR(A) + PR(B) + 0.5\ PR(D)$

$PR(D) = 0.33\ PR(A)$

$PR(E) = PR(C) + 0.5\ PR(D).$

Random Walk Example



1^{st} step

PR(A) = 0.2   PR(B) = 0.2   PR(C) = 0.2   PR(D) = 0.2   PR(E) = 0.2.

2^{nd} step

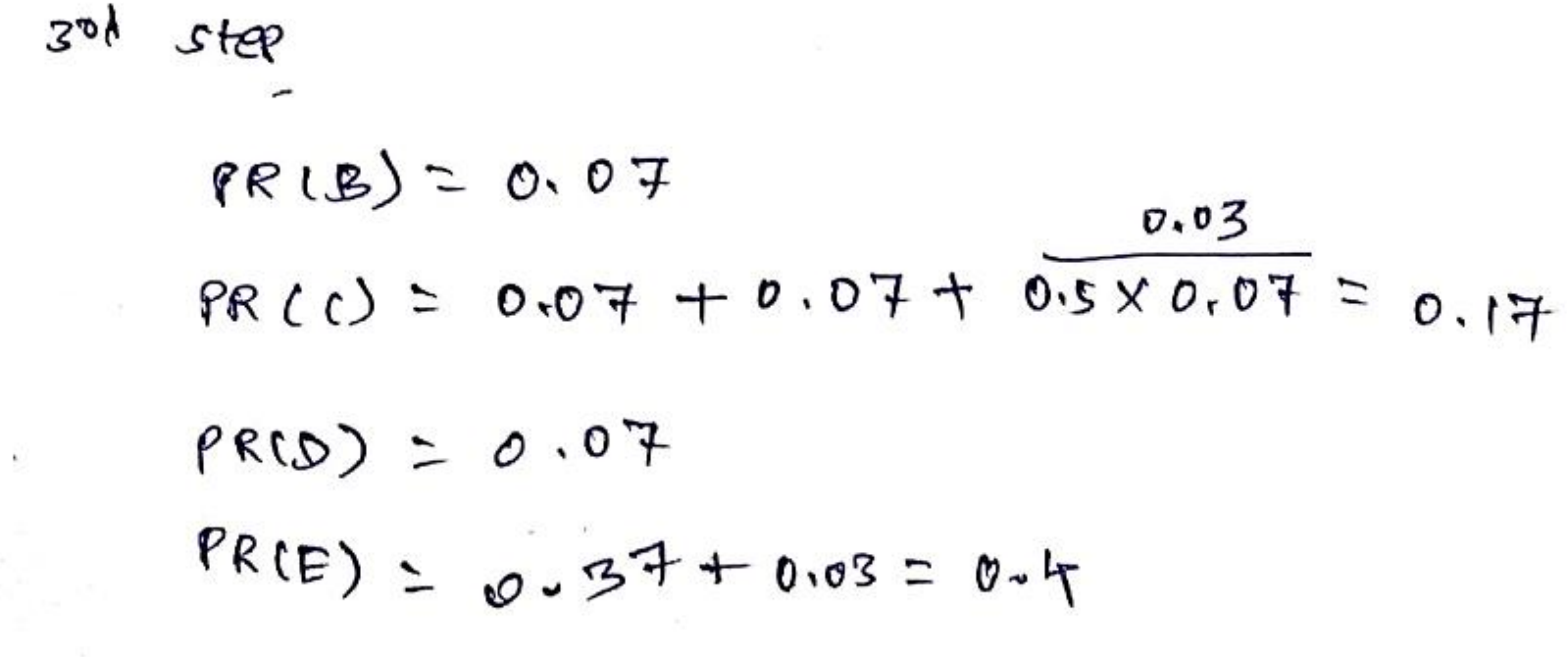
PR(B) = 0.33 x 0.2 = 0.07

PR(C) = 0.07 + 0.2 + 0.1 = 0.37

PR(D) = 0.07

PR(E) = 0.2 + 0.1 = 0.3.

Random Walk Example



3rd step

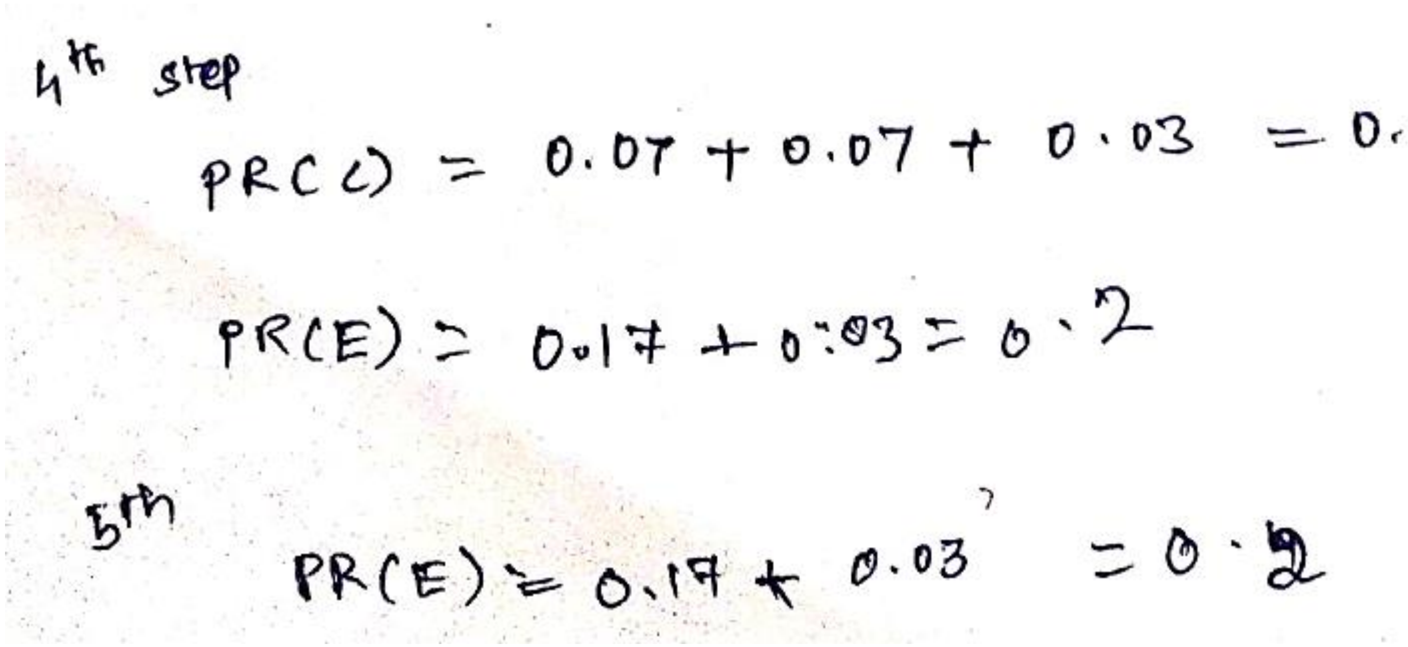
$PR(B) = 0.07$

$PR(C) = 0.07 + 0.07 + \overset{0.03}{\overline{0.5 \times 0.07}} = 0.17$

$PR(D) = 0.07$

$PR(E) = 0.37 + 0.03 = 0.4$

Random Walk Example



4th step

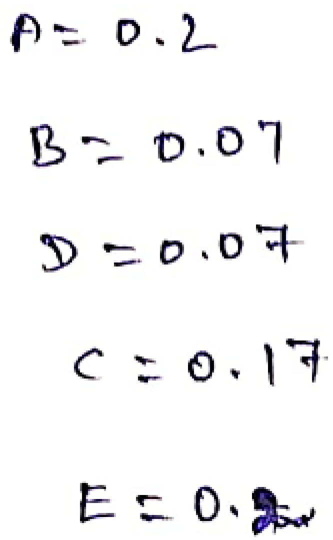
$PR(C) = 0.07 + 0.07 + 0.03 = 0.17$

$PR(E) = 0.17 + 0.03 = 0.2$

5th

$PR(E) = 0.17 + 0.03 = 0.2$

Result



$A = 0.2$

$B = 0.07$

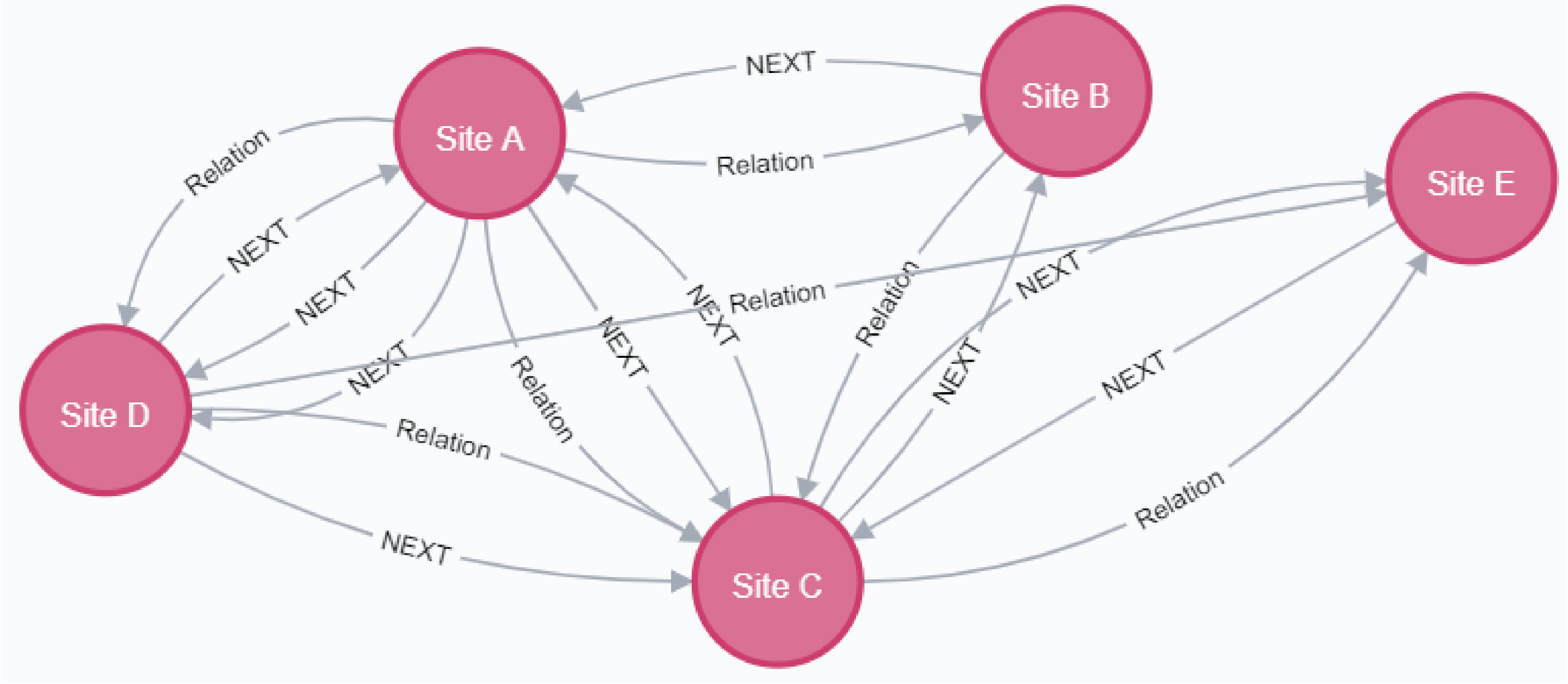
$D = 0.07$

$C = 0.17$

$E = 0.2$

Random Walk Example

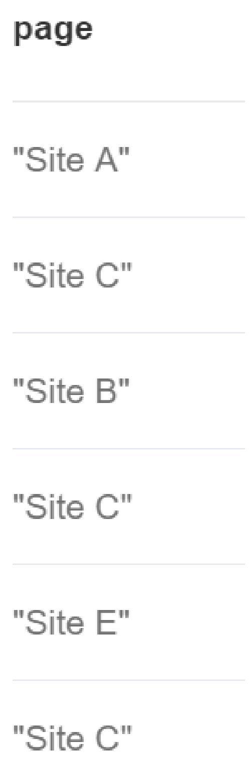
The following graph finds the random walk page ranking for the example considered in iteration page rank model.



Page Rank score using Random Walk

If node a is selected as starting, 5 number of steps and 1 walk:

```
MATCH (a:Page {name: "Site A"})
CALL algo.randomWalk.stream(id(a), 5, 1)
YIELD nodeIds

UNWIND nodeIds AS nodeId

RETURN algo.asNode(nodeId).name AS page
```

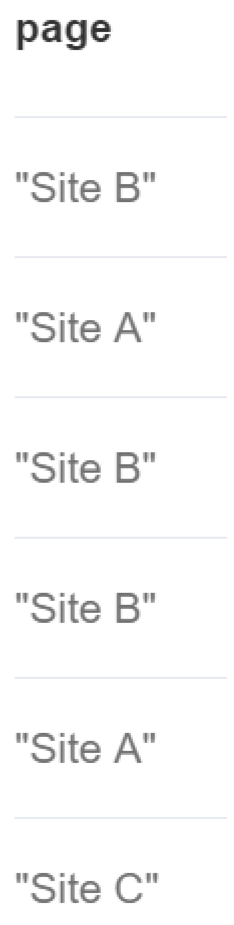


page
"Site A"
"Site C"
"Site B"
"Site C"
"Site E"
"Site C"

Page Rank score using Random Walk

If node b is selected as starting, 2 number of steps and 2 walk:

```
MATCH (b:Page {name: "Site B"})
CALL algo.randomWalk.stream(id(b), 2, 2)
YIELD nodeIds

UNWIND nodeIds AS nodeId

RETURN algo.asNode(nodeId).name AS page
```

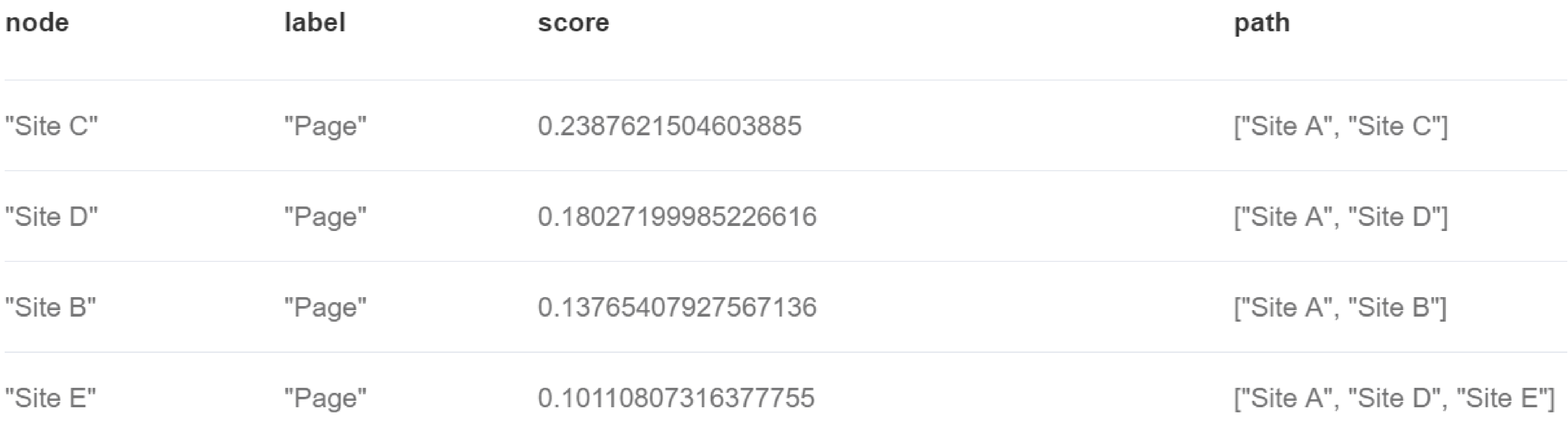


page
"Site B"
"Site A"
"Site B"
"Site B"
"Site A"
"Site C"

Page Rank Path using Random Walk

```
MATCH (a:Page {name: "Site A"})
CALL algo.pageRank.stream(null, null, {direction: "BOTH", sourceNodes: [a]})
YIELD nodeId, score
MATCH (n) WHERE id(n) = nodeId AND n <> a
RETURN coalesce(n.name, n.title) AS node,
       labels(n)[0] AS label,
       score,
       [node in nodes(shortestpath((a)-[*]-(n))) |
        coalesce(node.name, node.title)] AS path
ORDER BY score DESC
```

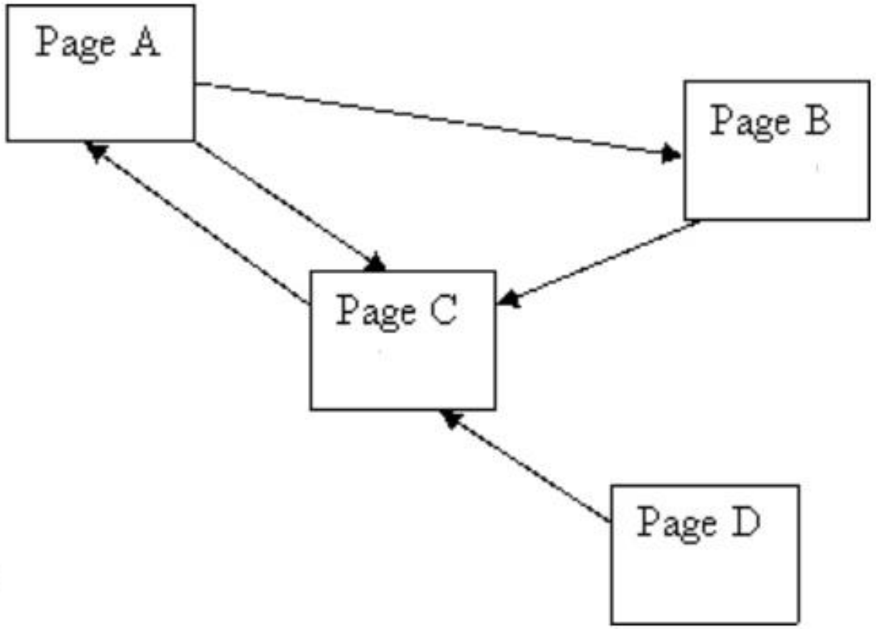
Page Rank Path Output

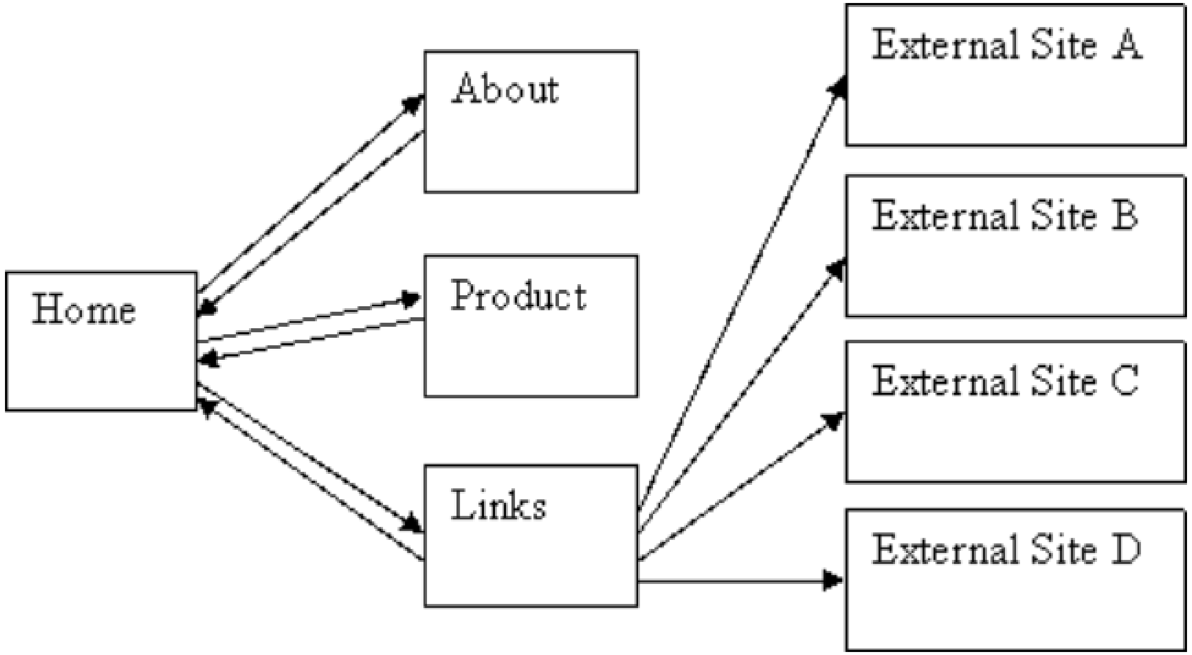


node	label	score	path
"Site C"	"Page"	0.2387621504603885	["Site A", "Site C"]
"Site D"	"Page"	0.18027199985226616	["Site A", "Site D"]
"Site B"	"Page"	0.13765407927567136	["Site A", "Site B"]
"Site E"	"Page"	0.10110807316377755	["Site A", "Site D", "Site E"]

Problems: Iteration processing based Page Rank

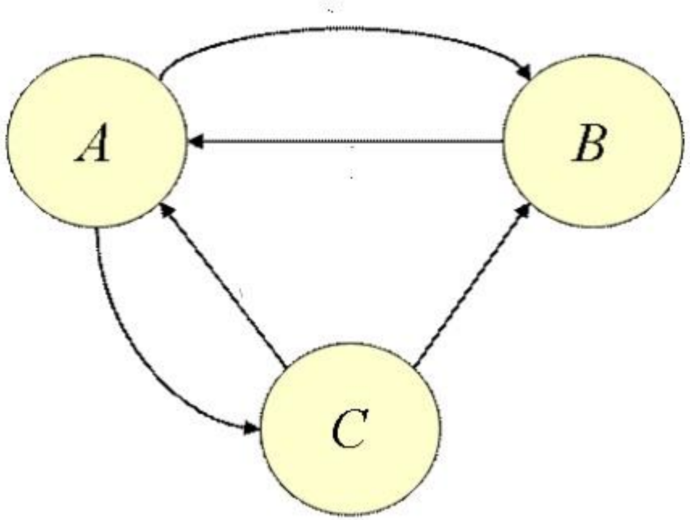
Compute the page rank for each node using iterative method and verify it with Neo4j command





Problems: Random Walk

Compute the page rank for each node using iterative method and verify it with Neo4j command



Summary

This session will give the knowledge about

- Graph Algorithms Introduction
- Page Rank computation using Iterative Processing
- Page Rank computation using Random Walk