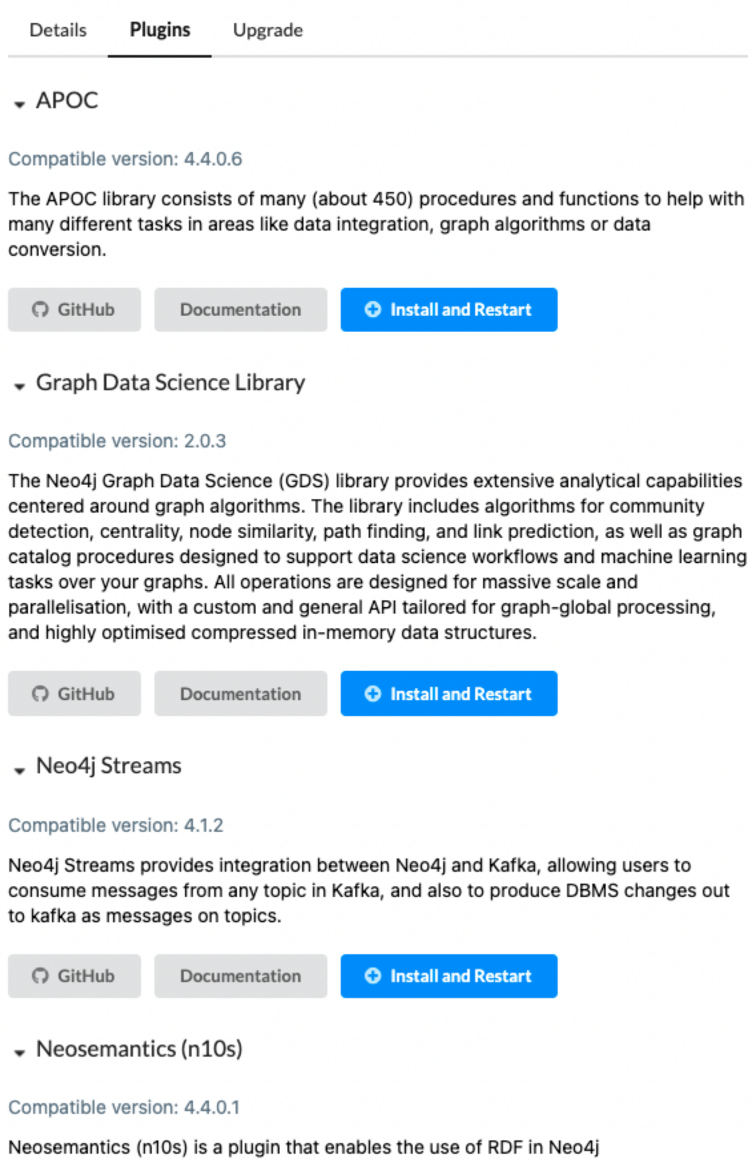
🎄

GDS 簡單介紹



APOC

Compatible version: 4.4.0.6

The APOC library consists of many (about 450) procedures and functions to help with many different tasks in areas like data integration, graph algorithms or data conversion.

Graph Data Science Library

Compatible version: 2.0.3

The Neo4j Graph Data Science (GDS) library provides extensive analytical capabilities centered around graph algorithms. The library includes algorithms for community detection, centrality, node similarity, path finding, and link prediction, as well as graph catalog procedures designed to support data science workflows and machine learning tasks over your graphs. All operations are designed for massive scale and parallelisation, with a custom and general API tailored for graph-global processing, and highly optimised compressed in-memory data structures.

Neo4j Streams

Compatible version: 4.1.2

Neo4j Streams provides integration between Neo4j and Kafka, allowing users to consume messages from any topic in Kafka, and also to produce DBMS changes out to kafka as messages on topics.

Neosemantics (n10s)

Compatible version: 4.4.0.1

Neosemantics (n10s) is a plugin that enables the use of RDF in Neo4j

若使用 desktop 可放在

- 基本結構

```
CALL gds[.<tier>].<algorithm>.<execution-mode>[.<estimate>](
  graphName: String,
  configuration: Map
)
```

- Execution mode
    - stream
        - 標準的查找模式，也是 cypher 的預設執行模式
        - 不擅長處理大數據，因此通常搭配使用 “top N-style” 的模式執行
    - stats
        - 將演算法的結果統計輸出（次數 or 百分比）
        - 不會對任何的東西做更改
        - 是 mutate 和 write 的基礎
    - mutate （變異）
        - 將統計結果寫回 projected graph （也就是被計算的 graph）
        - 可將複數的演算法寫在一起，無需再額外寫入 projected
            - 某些演算法有互相依賴的問題
            - 可直接將結果 轉換成 cypher （原本須額外的轉換 gds.util.nodeproperty）
    - write
        - 與 mutate 相似，可將統計結果直接寫進去 DB
            - projected graph 是被存在 memory 的 graph
- 使用 estimate 來估計運算資源！

> If the estimation shows that there is a very high probability of the execution going over its memory limitations, the execution is prohibited

Memory Estimation - Neo4j Graph Data Science

The graph algorithms library operates completely on the heap, which means we'll need to configure our Neo4j Server with a much larger heap size than we would for transactional

https://neo4j.com/docs/graph-data-science/current/common-usage/memory-estimation/#estimate-heap-control

The #1 Database for Connected Data

- 關於 graph catalog
  - catalog 是 將圖給提取出來放置記憶體中進行運算或研究（開發使用）
  - 延伸問題：如果有這個，還會需要radis 嗎？（開發的時候）

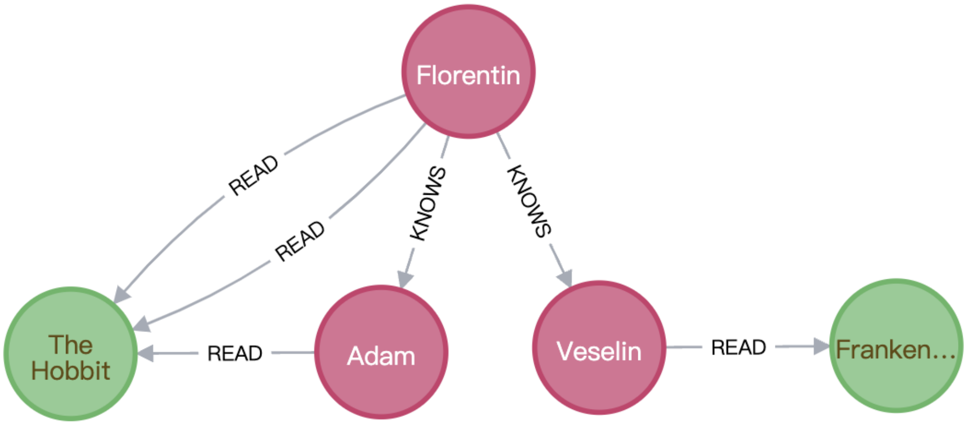
Graph Catalog - Neo4j Graph Data Science

Graph algorithms run on a graph data model which is a projection of the Neo4j property graph data model. A graph projection can be seen as a materialized view over the stored

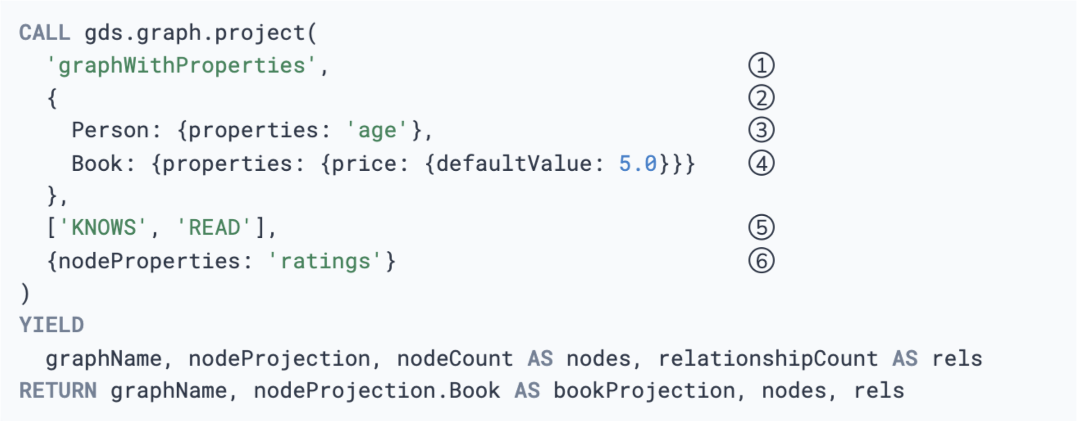
https://neo4j.com/docs/graph-data-science/current/management-ops/graph-catalog-ops/

The #1 Database for Connected Data

常用的指令可以查看上面的網頁



- 加入 graph catalog 需備三個基本元素
    - graph name
    - node
    - relation
- 加入的方法可以分為三種
    - native projection （預設）



```
CALL gds.graph.project(
  'graphWithProperties',                          ①
  {                                               ②
    Person: {properties: 'age'},                  ③
    Book: {properties: {price: {defaultValue: 5.0}}}   ④
  },
  ['KNOWS', 'READ'],                              ⑤
  {nodeProperties: 'ratings'}                     ⑥
)
YIELD
  graphName, nodeProjection, nodeCount AS nodes, relationshipCount AS rels
RETURN graphName, nodeProjection.Book AS bookProjection, nodes, rels
```

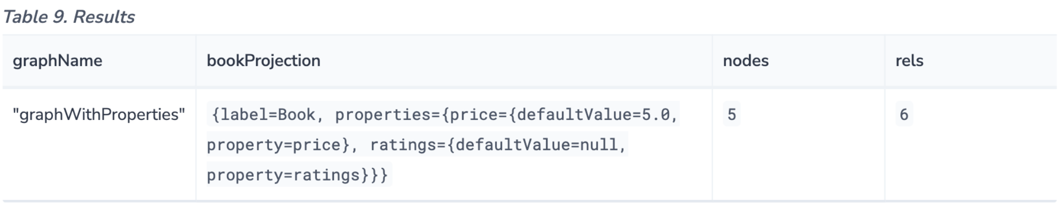
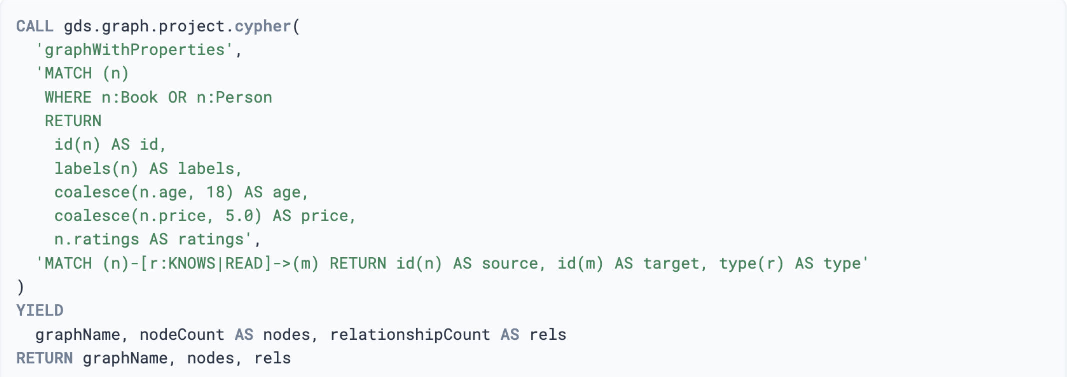


Table 9. Results

graphName	bookProjection	nodes	rels
"graphWithProperties"	`{label=Book, properties={price={defaultValue=5.0, property=price}, ratings={defaultValue=null, property=ratings}}}`	5	6

- cypher



```
CALL gds.graph.project.cypher(
  'graphWithProperties',
  'MATCH (n)
   WHERE n:Book OR n:Person
   RETURN
    id(n) AS id,
    labels(n) AS labels,
    coalesce(n.age, 18) AS age,
    coalesce(n.price, 5.0) AS price,
    n.ratings AS ratings',
  'MATCH (n)-[r:KNOWS|READ]->(m) RETURN id(n) AS source, id(m) AS target, type(r) AS type'
)
YIELD
  graphName, nodeCount AS nodes, relationshipCount AS rels
RETURN graphName, nodes, rels
```

使用 cypher 找到需要的 node 與 relations ， 並將點與關係轉換為 id 後回傳

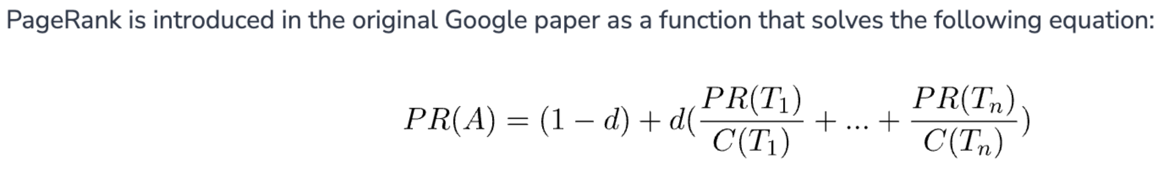
node: 須轉換 id、對應的 label，其餘的會作為 properties

relation: id、realation type，其餘的會作為 properties

- aggregation
  - 傳入指定的點
  - 還要研究一下
- project graph 是 user define 的，當你切換ㄕ

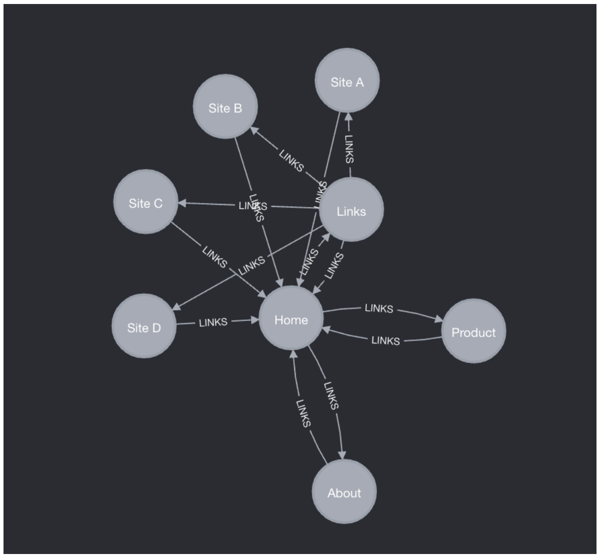
algorithms 分類

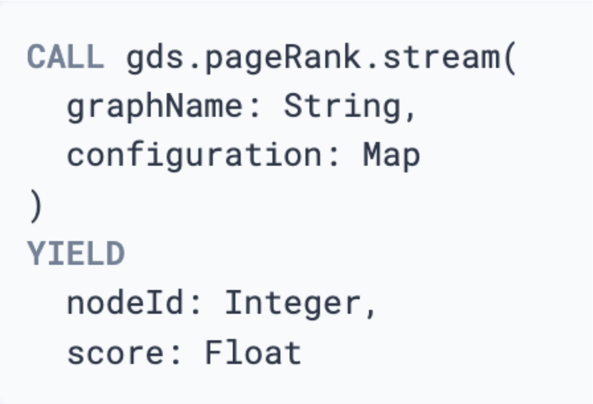
PageRank



PageRank is introduced in the original Google paper as a function that solves the following equation:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

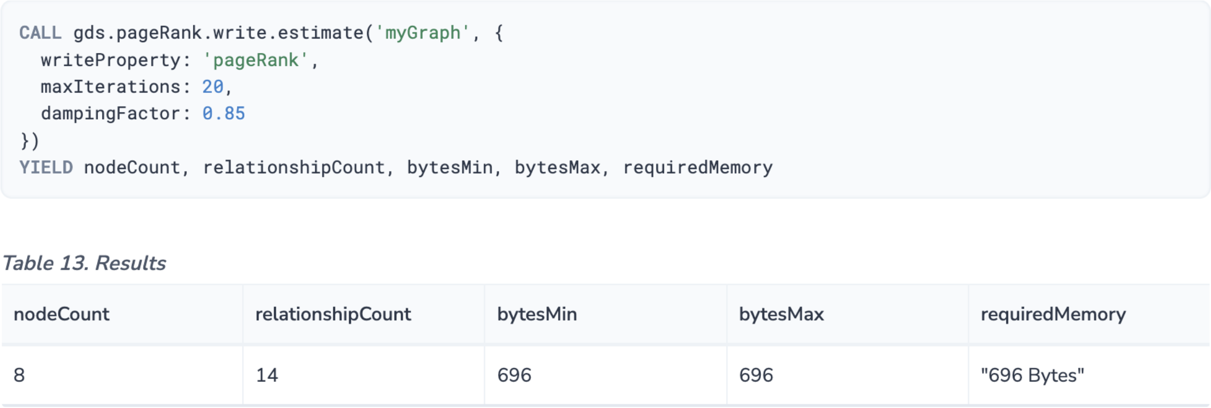




```
CALL gds.pageRank.stream(
  graphName: String,
  configuration: Map
)
YIELD
  nodeId: Integer,
  score: Float
```

```
CALL gds.graph.project(
  'myGraph',
  'Page',
  'LINKS',
  {
    relationshipProperties: 'weight'
  }
)
```

Estimate



```
CALL gds.pageRank.write.estimate('myGraph', {
  writeProperty: 'pageRank',
  maxIterations: 20,
  dampingFactor: 0.85
})
YIELD nodeCount, relationshipCount, bytesMin, bytesMax, requiredMemory
```

Table 13. Results

nodeCount	relationshipCount	bytesMin	bytesMax	requiredMemory
8	14	696	696	"696 Bytes"

Stream output

```
CALL gds.pageRank.stream('myGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC, name ASC
```

Table 14. Results

name	score
"Home"	3.215681999884452
"About"	1.0542700552146722
"Links"	1.0542700552146722
"Product"	1.0542700552146722
"Site A"	0.3278578964488539
"Site B"	0.3278578964488539
"Site C"	0.3278578964488539
"Site D"	0.3278578964488539