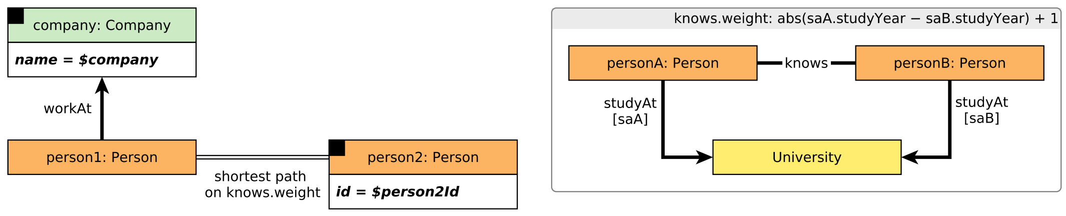
## BI / read / 20

| query | BI / read / 20 |
|---|---|
| title | Recruitment |
| pattern |   company: Company<br>name = $company<br>workAt<br>person1: Person<br>shortest path on knows.weight<br>person2: Person<br>id = $person2Id<br>knows.weight: abs(saA.studyYear − saB.studyYear) + 1<br>personA: Person<br>knows<br>personB: Person<br>studyAt [saA]<br>studyAt [saB]<br>University |
| desc. | Given a Company company and a Person person2 (who is known to be working at another Company), find a Person (person1) working the in the company who have the top-20 shortest path to person2 through people who have studied together. On this path, we only consider edges between Persons who know each other and attended the same university and set the weight of the edge to the absolute difference between the year of enrolment plus 1 (`studyAt.classYear + 1`).<br>If there are multiple Person person1 nodes with the same shortest path, return all of them. |

| | | | | |
|---|---|---|---|---|
| params | 1 | company | Long String | Companies with a similar number of employees (former or current) are selected |
| | 2 | person2Id | ID | |
| result | 1 | person1.id | ID | R |
| | 2 | totalWeight | 64-bit Integer | C |
| sort | 1 | person1.id | ↑ | |

| | |
|---|---|
| limit | 20 |
| CPs | 3.3, 7.6, 7.7, 8.4, 8.6 |
| relevance | Implementations can either pre-compute edge weights or compute them on-the-fly. To find a weighted shortest path efficiently, implementations can use e.g. a bidirectional Dijkstra algorithm. |