Løsningsforslag til Multi-Model DBs (ArangoDB) oppgaver

Bakgrunnsinfo

- Følg dokumentet for oppsett av ArangoDB : https://docs.arangodb.com/3.3/Manual/GettingStarted/Installing/

- For dokumentasjon på AQL (spørrespråket) se her: https://docs.arangodb.com/3.3/AQL/Fundamentals/

Graf oppgaver (spørringer)

Med utgangspunkt i oppgavene for Neo4J (https://www.dropbox.com/s/sfyuk2j1dff5g76/Graph%20-%20Neo4j%20-%20homework.pdf?dl=0), gjør det samme i ArangoDB:

- Opprett en ny collection «Nodes» og en edge-collection «node_edges»
- Du kan enten opprette nodene og kantene gjennom admin panelet, eller du kan lage 2 json filer med dataen i. Dersom du lager json filer må du passe på at du kun har ett json objekt per linje. I tillegg må hvert objekt ha en variabel «_key» som er unik og forteller hvilke objekt det er.
  Så student 1 blir for eks:
  {_key: «s1», studentID: "1", lastName: "Doe", firstName: "Ana", middleName: «Maria"}
  Deretter kan du følge import oppskriften i dokumentet for oppsett og import til ArangoDB

Document collection


```
{"_key": "s1", "studentID": "1", "lastName": "Doe", "firstName": "Ana", "middleName": "Maria"}
{"_key": "s2", "studentID": "2", "lastName": "Ung", "firstName": "Peter", "middleName": "John"}
{"_key": "s3", "studentID": "3", "lastName": "Doe", "firstName": "John"}
{"_key": "s4", "studentID": "4", "lastName": "Berre", "firstName": "Stine"}
{"_key": "s5", "studentID": "5", "lastName": "Travolta", "firstName": "John"}
{"_key": "c1", "courseNr": "1", "courseName": "Databases"}
{"_key": "c2", "courseNr": "2", "courseName": "Programming"}
{"_key": "c3", "courseNr": "3", "courseName": "Graphics"}
{"_key": "p1", "projectNr": "34", "projectName": "eCommerce database"}
{"_key": "p2", "projectNr": "24", "projectName": "eCommerce website"}
{"_key": "p3", "projectNr": "13", "projectName": "User interface"}
{"_key": "p4", "projectNr": "26", "projectName": "Reporting"}
{"_key": "r1", "roomName": "Pascal"}
{"_key": "r2", "roomName": "Seminar C"}
{"_key": "r3", "roomName": "Alpha"}
{"_key": "r4", "roomName": "Beta"}
```


Edge collection


```
{"_from": "c1", "_to": "r1" , "type": "takesplaceing"}
{"_from": "c1", "_to": "r3" , "type": "takesplaceing"}
{"_from": "c1", "_to": "r4" , "type": "takesplaceing"}
{"_from": "c2", "_to": "r2" , "type": "takesplaceing"}
{"_from": "s1", "_to": "c1" , "type": "enrolledin"}
{"_from": "s2", "_to": "c1" , "type": "enrolledin"}
{"_from": "s3", "_to": "c2" , "type": "enrolledin"}
{"_from": "s4", "_to": "c1" , "type": "enrolledin"}
{"_from": "s1", "_to": "p1" , "type": "workson", "hours": 1}
{"_from": "s1", "_to": "p2" , "type": "workson", "hours": 2}
{"_from": "s2", "_to": "p1" , "type": "workson", "hours": 3}
{"_from": "s2", "_to": "p2" , "type": "workson", "hours": 4}
{"_from": "s2", "_to": "p3" , "type": "workson", "hours": 1}
{"_from": "s2", "_to": "p4" , "type": "workson", "hours": 1}
{"_from": "s3", "_to": "p1" , "type": "workson", "hours": 1}
{"_from": "s3", "_to": "p2" , "type": "workson", "hours": 2}
{"_from": "s3", "_to": "p4" , "type": "workson", "hours": 3}
```

```
/*Q1 Neo4j*/
FOR course IN Neo4J
    //First find the node with coursnumber 1
    FILTER course.courseNr == "1"

    //look at edges with type takesplace in
    FOR room, edge, path IN ANY course GRAPH 'Neo4J'
        FILTER edge.type == "takesplacein"

RETURN {"CourseName": course.courseName, "room": room.roomName}


/*Q2 Neo4j*/
FOR student IN Neo4J
    //First find the node with coursenumber 1
    FILTER student.studentID == "1"

    //Find the corresponding projects, edge with type "workson"
    FOR project, edge, path IN ANY student GRAPH 'Neo4J'
        FILTER edge.type == "workson"

RETURN {"Student": student.firstName, "Project": project.projectName, "Hours": edge.hours}
```

```
/*Q3 Neo4j*/
FOR project IN Neo4J
    //First find the node with projectnumber 24
    FILTER project.projectNr == "24"

    //Find the corresponding students, edge with type "workson"
    FOR student, edge, path IN ANY project GRAPH 'Neo4J'
        FILTER edge.type == "workson"

RETURN {"Student": student.lastName, "Project": project.projectName, "Hours": edge.hours}


/*Q4 Neo4j*/
FOR project IN Neo4J
    //First find the nodes thats a project
    FILTER HAS(project, "projectNr") == true

    //Find the corresponding student, edge with type "workson"
    FOR student, edge, path IN ANY project GRAPH 'Neo4J'
        FILTER edge.type == "workson"
SORT student.lastName
LIMIT 4
RETURN {"Student": student.lastName, "Project": project.projectName, "Hours": edge.hours}
```

```
/*Q5 Neo4j*/
FOR student IN Neo4J
    //First find the nodes that are students
    FILTER HAS(student, "studentID") == true

    //Find the corresponding projects, edge with type "workson"
    LET projects = (FOR projects, edge, path IN ANY student GRAPH 'Neo4J'
        FILTER edge.type == "workson"
        return projects)

    //How many projects are there?
    LET numberProjects = LENGTH(projects)

    //We only want those that have two or more projects
    FILTER numberProjects > 1

    //Sort based on number of projects
    SORT numberProjects

RETURN {"Student": student.lastName, "Project": numberProjects}
```

```
/*Q6 Neo4j*/
FOR project IN Neo4J
    //First find the nodes that are projects
    FILTER HAS(project, "projectNr") == true

    //Find all students connected to the project
    LET studentsinProject = (FOR student, edge, path IN ANY project GRAPH 'Neo4J'
                                FILTER edge.type == "workson"
                                return student)

    //Check if students in the project has the same last name
    LET students = (
        FOR student1 IN studentsinProject
            FOR student2 IN studentsinProject
                FILTER student1.lastName == student2.lastName && student1.studentID != student2.studentID
        RETURN {"student": student1.firstName, "student": student2.firstName}
        )

//Filter so we only get projects having students with equal lastname
FILTER LENGTH(students) > 0

RETURN {"Project": project.projectName, "students": students}
```