Explore JanusGraph and its Storage Plugins

Please complete the following sub-tasks (the order is just a recommendation, you could complete them in any order):

1. Install Java 8, Maven and IntelliJ
2. Create a new Maven project (called SampleGraphApp) and develop a simple application with JanusGraph
    a. Use janusgraph-core dependency: https://mvnrepository.com/artifact/org.janusgraph/janusgraph-core/1.0.0-20231021-144322.55d7454
    b. Use janusgraph-inmemory dependency so the graph created is stored in the application memory: https://mvnrepository.com/artifact/org.janusgraph/janusgraph-inmemory/1.0.0-20231021-144322.55d7454
    c. Write code to initialize the schema for Air Routes (see the note below)
    d. Load the graph with some airroutes data. The data set could be found here: https://github.com/krlawrence/graph/blob/master/sample-data/air-routes-latest-nodes.csv https://github.com/krlawrence/graph/blob/master/sample-data/air-routes-latest-edges.csv

3. Switch to use BerkeleyDB storage backend and load the whole graph:
    a. Use janusgraph-berkeleyje dependency: https://mvnrepository.com/artifact/org.janusgraph/janusgraph-berkeleyje/1.0.0-20231021-144322.55d7454
    b. Load the whole airroutes data.
    c. Report the number of vertices/edges loaded, the time it takes to load and the storage size

4. Switch to use foundationdb storage backend (see the .tar.gz file attached).
    a. Install FoundationDB (version 6.2.30) and set it up and running
    b. Unzip the tar file and open it in IntelliJ as a maven project
    c. Build and install it locally
    d. Use its dependency in your SampleGraphApp
    e. Load the whole airroutes graph.
    f. Similarly, report the number of vertices/edges loaded, the time it takes to load and the storage size

Note: The schema for Airroutes is expressed in json format as below

```
{
 "vertexLabels":[
  {
   "name":"airport"
  },
  {
   "name":"country"
```

```
 },
 {
  "name":"continent"
 }
],
"edgeLabels":[
 {
  "name":"route",
  "directed":true,
  "multiplicity":"MULTI"
 },
 {
  "name":"contains",
  "directed":true,
  "multiplicity":"MULTI"
 }
],
"propertyKeys":[
 {
  "name":"city",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"lat",
  "dataType":"Double",
  "cardinality":"SINGLE"
 },
 {
  "name":"lon",
  "dataType":"Double",
  "cardinality":"SINGLE"
 },
 {
  "name":"dist",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"identity",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"type",
  "dataType":"String",
  "cardinality":"SINGLE"
```

```
 },
 {
  "name":"code",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"icao",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"desc",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"region",
  "dataType":"String",
  "cardinality":"SINGLE"
 },
 {
  "name":"runways",
  "dataType":"Integer",
  "cardinality":"SINGLE"
 },
 {
  "name":"longest",
  "dataType":"Integer",
  "cardinality":"SINGLE"
 },
 {
  "name":"elev",
  "dataType":"Integer",
  "cardinality":"SINGLE"
 },
 {
  "name":"country",
  "dataType":"String",
  "cardinality":"SINGLE"
 }
],
"graphIndices":{
 "compositeIndices":[
  {
   "indexName":"Idx_comidx_Vertex_identity_unique",
   "elementType":"vertex",
```

```
 "propertyKeys":[
  "identity"
 ],
 "unique":true
},
{
 "indexName":"Idx_comidx_Vertex_type_airport",
 "elementType":"vertex",
 "propertyKeys":[
  "type"
 ],
 "indexOnly":"airport",
 "unique":false
},
{
 "indexName":"Idx_comidx_Vertex_code",
 "elementType":"vertex",
 "propertyKeys":[
  "code"
 ],
 "unique":false
},
{
 "indexName":"Idx_comidx_Vertex_icao",
 "elementType":"vertex",
 "propertyKeys":[
  "icao"
 ],
 "unique":false
},
{
 "indexName":"Idx_comidx_Vertex_country",
 "elementType":"vertex",
 "propertyKeys":[
  "country"
 ],
 "unique":false
},
{
 "indexName":"Idx_comidx_Vertex_city",
 "elementType":"vertex",
 "propertyKeys":[
  "city"
 ],
 "unique":false
},
{
```

```
   "indexName":"Idx_comidx_Edge_identity",
   "elementType":"edge",
   "propertyKeys":[
    "identity"
   ],
   "unique":false
  }
 ],
 "mixedIndices":[

 ]
},
"vertexCentricIndices":[

]
}
```

Useful Links:

- Janusgraph docs: https://docs.janusgraph.org/ (to get familiar with JanusGraph)
- Tinkerpop docs: https://tinkerpop.apache.org/docs/current/reference/ (for Tinkerpop and Gremlin reference)
- Tutorial on Gremlin: https://www.kelvinlawrence.net/book/PracticalGremlin.html (the tutorial heavily uses AirRoutes graph)

Delivery:

- A report on each sub-task above (what you accomplished, whether the sub-task was fully complete or if not, what were the obstacles that you encountered).
- Initialize a new Github repository and push your project (SampleGraphApp) to it. Provide the Github repository link as part of your report. Feel free to provide inline comments or README instructions to set up and run your code.