Modélisation (Renaud)

Conception

Donnée

Acquisition de la valeur d'une variable concernant un objet, à un temps T selon une provenance.

Ex : Le 12 octobre 2022 à 12:00, la température de la plante "plant_1" est de 30°. Cette valeur provient de l'acquisition de mesure à l'aide du capteur de température "DS18B20". - `valeur` : 30 - `date` : 12 octobre 2022 à 12:00 - `variable` : température de la plante en °Celsius - `objet` : la plante "plant_1" - `provenance` : Une acquisition de donnée faite à l'aide de capteur de température

Type de variable

La valeur (champ `value`) permet de représenter n'importe quel type de donnée : entier, décimal, réel, date, chaîne de caractère, tableau ou bien un objet. Le type va dépendre de la variable utilisée

- Plusieurs types de données/plusieurs variables :
  - variable **numérique** : température, humidité
  - variable **alphanumérique** : orientation
  - variable **multi-dimensionnelle**/objet : composantes R,G,B qui définissent la couleur

La souplesse du schéma JSON permet de représenter cette contrainte, en sachant que le type effectif dépend de la variable que l'on souhaite mesurer. Ici un entier pour représenter une température.

Provenance

Contexte d'acquisition d'une donnée selon différent niveau de granularité.

- `name`: Nom de la provenance
- `experiment`: Expérience dans laquelle la mesure a été produite

  Agents - `agents` (**Agent**) : Liste des objets/personnes ayant effectué la mesure - Un logiciel - Une personne - Un capteur

  Activité - `activity` (**Activity**): description de l'activité d'acquisition de mesure - Acquisition de température à l'aide d'un capteur de température - Un calcul via un logiciel - Mesure de la taille de plante par une personne

Data

Valeur simple

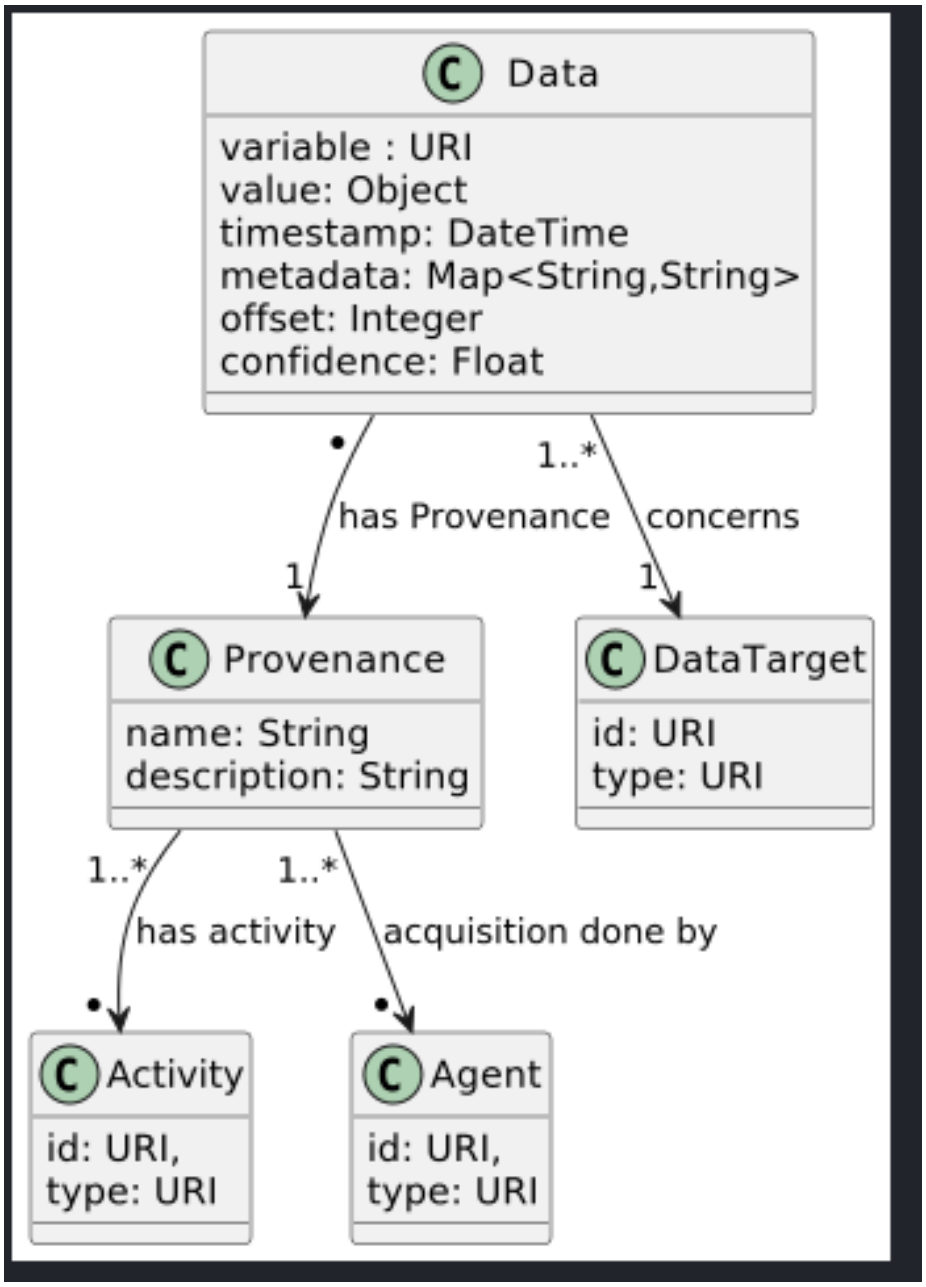

Figure 1: uml diagram data provenance

```
{
    "value": 36.5,
    "variable": "abd:temperature",
    "target": {
        "type": "vocabulary:Plant",
        "uri" :"abd:plant_1"
    },
    "date": "0221013:1306:52:+2",
    "provenance": {
        "uri" :"abd:provenance_1",
        "agents" : [
          {
              "type": "vocabulary:TemperatureSensor",
              "uri" :"abd:tempature_sensor_1"
          }
        ]
    }
}
```

Valeur multiple

```
{
    "_id":"abd:data_1",
    "value": [45,32,63,58,61,21],
    "variable":"abd:LeafSize",
    "target": {
        "type": "vocabulary:Plant",
        "uri" :"abd:plant_1"
    },
    "date": "0221013:1306:52:+2",
    "provenance": {
        "uri" :"abd:provenance_2",
        "agents" : [
          {
              "type": "vocabulary:Sensor",
              "uri" :"abd:leaf_size_sensor"
          }
        ]
    }
}
```

Valeur multidimensionelle

```
{
  "_id":"abd:data_1",
  "value": {
    "r": 214,
    "g": 58,
```

```
    "b": 21
  },
  "variable":"abd:color",
  "target": {
    "type": "vocabulary:Plant",
    "uri":"abd:plant_1"
  },
  "date": "0221013:1306:52:+2",
  "provenance": {
    "uri":"abd:provenance_3",
    "agents": [
      {
        "type": "vocabulary:ColorSensor",
        "uri":"abd:color_sensor"
      }
    ]
  }
}
```

Basic usage and CRUD (Renaud)

Connection à un serveur MongoDB (via Tunnel SSH)

Terminal

```
ssh  ssh_user@147.100.202.42 -p 22 -Nf -L 8668:localhost:27017
```

Aide

- `ssh ssh_user@147.100.202.42` : Connection SSH avec login et addresse du serveur mongodb
- `-p 22` Connection SSH sur le port 22
- `-N` Pas d'execution de commande,
- `f` Execution en fond (permet de se connecter à distance via un autre processus)
- `-L 8668:localhost:27017` Redirection du port 27017 (serveur mongodb) vers le port local 8668

Connection via Mongo Compass

Avec URl de connection

`mongodb+srv`*//atelier_big_data:P2O4ld1p2o943B@/admin*

Via formulaire (sur un server local)

```
hostname: localhost
port: 8668
```

Database et collection

Database

Show databases

```
show dbs
```

Utiliser la base de données `atelier_bid_data_shared_db_10_plant`
La collection **data** de cette base, contient une mesure par heure, de la température de 100 plantes, durant pendant 10 ans Mongo Doc

```
use atelier_bid_data_shared_db_10_plant
```

Collection

Afficher la liste des collections

```
show collections
```

Récupérer un document de la collection `data`

```
db.data.findOne()
```

Create the collection `big_data_tp_name`

```
db.createCollection("atelier_data_collection")
```

Delete the collection `atelier_data_collection`

```
db.atelier_data_collection.drop()
```

Lecture

Liens

Projection/Renvoyer seulement les champs spécifiés Requêter un sous document Requêter un tableau de sous-document

Recherche par id

Récupérer une donnée

```
db.data.findOne({});
```

Récupérer une donnée par identifiant (`uri`)

```
db.data.find({
    "uri": "opensilex:id/data/41f5454"
});
```

ou

```
db.data.find({
    "uri": {$exists:true, $eq: "opensilex:id/data/41f5454"}
})
```

Récupérer une donnée par identifiant (`_id`)

```
db.data.findOne({
    _id : ObjectId ("127x74937107dkcde3beb986")
})
```

Tri et pagination

Récupérer 1000 documents

```
db.data.find({})
    .limit(1000)
```

Récupérer les 1000 derniers documents **Ordre**: `date` (descendant)

```
db.data.find({})
    .limit(1000)
    .sort({date: -1})
```

Filtrer les champs renvoyés

Récupérer les 1000 derniers documents **Ordre**: `date` (descendant) **Projection**: `value`, `target`, ‘date

```
db.data.find(
        {},
        {value: 1, target:1, date: -1}
)
.limit(1000)
.sort({date: -1})
```

Recherche par valeur

Récupérer 1000 données acquises avec le capteur `abd:temp_sensor_1`

```
db.data
    .find({"provenance.agents.uri": "abd:tempature_sensor_1"})
    .limit(1000)
```

Récupérer 1000 données acquises avec un capteur de température `vocabulary:TemperatureSensor`

```
db.data
    .find({"provenance.agents.type": "vocabulary:TemperatureSensor"})
    .limit(1000)
```

Filtre d'égalité et ordre

Récupérer 1000 données de temperature >= 20° **Ordre**: `date` **Projection**: `value`, `target`, ‘date

```
db.data.find({
            {"value" : $gte: 20},
            {"variable": "abd:temperature"},
        },
        {"value" : 1, "target" : 1, "date": 1}
    )
    .limit(1000)
    .sort({date: -1})
```

Récupérer les données de temperature acquise ce mois-ci avec le capteur `abd:temperature` **Ordre**: `date` **Projection**: `value`, `target`, ‘date —

```
db.data
    .find({
            {"variable": "abd:temperature"},
            {"provenances.agents.uri" :"abd:tempature_sensor_1"},
            {"date": {$gte: ISODate("2023-01-01")}}
        },
        {date: -1},
        {"value" : 1, "target" : 1, "date": 1}
    )
    .limit(1000)
```

Récupérer les données de temperature de la semaine acquises sur la plante `"abd:plant_1" **Ordre**:`date`**Projection**:`value,target,date —

```
db.data
    .find({
            {"variable":"abd:temperature"},
            {"target.uri" : "abd:plant_1"},
```

```
            {"date":
                $gte: ISODate("2023-01-02"),
                $lt: ISODate("2023-01-09")
            }
        },
        {"value" : 1, "target" : 1, "date": 1}
    )
    .limit(1000)
```

> Récupérer les données du jour acquises avec un capteur de temperature

```
db.data.find({
    "provenance.agents.type" : "vocabulary:TemperatureSensor",
    "date": {
        $gte: ISODate("2012-01-02"),
        $lt: ISODate("2012-01-03")
    }
}
).limit(1000)
```

Récupérer les mesures de couleurs avec une composante rouge > 100 et bleu > 100

```
db.data
.find(
    {"variable" : "abd:color"},
    {"value.r" : {$gte:100}},
    {"value.b" : {$gte:100}},
} ).limit(1000)
```

Group by et aggregation

Aggregation Documentation Group Documentation

Récupérer la dernière temperature mesurée sur chaque plante **Ordre**: `date` **Projection**: `value`, `target`, ‘date

Récupérer le nombre de mesure effectuées sur chaque plante

```
db.data.aggregate([
    {
        $group: {
            _id: "$target.uri",
            plant_count: { $count: { } }
        }
    },
```

```
    {
        $match: {
            "target.type" : "vocabulary:Plant"
        },
    }{
        $sort: {date: -1},
    }
])
```

Récupérer la dernière temperature mesurée pour chaque capteur de temperature

#todo

Calculer la moyenne des temperatures de chaque plante sur l'année 2022

```
db.data.aggregate([
    {
        $group: {
            avg_temp_plante: {$avg: "value"},
            target.uri: 1
        }
    },
    {
        $project: {
            target.uri: 1, avg_temp_plante: 1
        }
    },
    {
        $match: {
            "target.type" : "vocabulary:Plant",
            "variable" : "abd:temperature",
            "date" : {$gte:ISODate("2022-01-01"), $lt: ISODate("2023-01-01")}
        }
    },
    {
        $sort: {date: -1},
    }
])
```

Calculer la moyenne des temperatures de chaque plante, pour chaque année, sur les 10 dernieres années

#todo

```
db.data.aggregate([
    {
        $group: {
```

```
            avg_temp_plante: {$avg: "value"},
            target.uri: 1
        }
    },
    {
        $project: {
            target.uri: 1, nb_plant_measure: 1
        }
    },
    {
        $match: {
            "target.type" : "vocabulary:Plant",
            "variable" :"abd:temperature"
        }
    },
    {
        $sort: {date: -1},
    }
])
```

Create

Add a data document

```
db.data.insert({
    variable: "dev:id/variable#variable.air_temperature",
    target: "dev:id/plant_1",
    date : "2022-18-19",
    value: 36,
})
```

Delete

Delete an element by uri

```
db.getCollection('data').deleteOne({ _id : ObjectId ("127x74937107dkcde3beb986") })
```

Delete data according search criteria

```
db.getCollection('data').find({
        variable: "dev:id/variable#variable.air_temperature")
}).drop()
```

Agrégation (Renaud)

Jointure avec une autre collection

```
db.data.aggregate([
    {
        $lookup: {
            from: "provenances",
            localField: "provenance.uri",
            foreignField: "uri",
            as: "provenance_foreign"
        }
    }
])
```

Jointure avec une autre collection (avec filtre)

```
db.data.aggregate([
    {
        $lookup: {
            from: "provenances",
            localField: "provenance.uri",
            foreignField: "uri",
            pipeline: [
                { $match: {experiment: "atelier_big_data:id/xp/1"}}
            ],
            as: "provenance_foreign"
        }
    }
])
```

Jointure avec une autre collection (avec filtre et projection)

Projection : `name,experiment`

```
db.data.aggregate([
    {
        $lookup: {
            from: "provenances",
            localField: "provenance.uri",
            foreignField: "uri",
            pipeline: [
                { $match: {experiment: "atelier_big_data:id/xp/1"}},
                { $project: {name: 1, experiment: 1}}
            ],
            as: "provenance_foreign"
        }
```

```
    }
])
```

Indexation

Basic index

- MongoDB indexes manual

  Index on field (descendant order) Example : Index data by acquisition date

```
db.data.createIndex( { date: -1 }
```

  Index on field (ascendent order) Example : Index data by variable

```
db.data.createIndex( { variable: 1 }
```

  Index a nested field Example: Index data by provenance id —

```
db.provenance.createIndex( { provenance._id: 1 }
```

  Index on nested array field Example: Index data according provenance agents type

```
db.provenance.createIndex( { provenance.agents.type: 1 }
```

  Drop an index (by index name)

```
db.data.dropIndex("variable_1_provenance_1_date_-1")
```

Compound index

- Compound indexes manual

  Add a compound index on (variable, provenance, date) fields

```
db.data.createIndex( { variable: 1, provenance: 1, date: -1 })
```

  Add an unique and compound index on (variable, provenance, date) fields

```
db.data.createIndex( { variable: 1, provenance: 1, date: -1 }, {unique: true})
```