Fluree Cheat Sheet:

Contents

Key Concepts for any Request/Transaction:

- When using Postman to perform requests on the Fluree ledger the following concepts are important.

- **"@context"** – This allows the u to give a prefix to each URI. For example, giving the owl vocabulary a prefix **"owl"** will look like this in Postman:

```
{
    "@context" :
    {
        "owl" : "http://www.w3.org/2002/07/owl#"
    }
}
```

    - It is not strictly necessary to use Context to give the URI a prefix, however it increases the readability of the request a lot.

- **"ledger"** – This is a hard requirement for any transaction, whether it is a create, query or insert. The ledger specifies from which dataset the request wants to extract the data. In our use case we only used one ledger. Specifying the ledger looks like this:

```
{
    "@context" :
    {
        "owl" : "http://www.w3.org/2002/07/owl#"
    },
    "ledger" : "ledger_name"
}
```

    - Replace ledger_name with the given name of your ledger, or if you are creating a new ledger, the ledger's name.

- **"insert"** – after specifying the ledger and the context, if your goal is to insert new data into the ledger you use the insert statement:

```
{
    "@context" :
    {
        "owl" : "http://www.w3.org/2002/07/owl#"
    },
    "ledger" : "ledger_name",
    "insert" :
    {
```

```
        "@id" : "owl:Thing",
        "@type" : "owl:Class"
    }
}
```

- After "insert" follows a ":" and then "{}" to specify an "entity" being inserted into the ledger. The entity being inserted into the database is from the Owl vocabulary. Each entity always requires an **"@id"** and an **"@type"** when inserting. In this example the entity is an owl class and thus is of "@type" : "owl:Class", or without using the prefix: "@type" : "http://www.w3.org/2002/07/owl# Class".

Basic Level:

```
{
  "@context": {
    "schema": "http://schema.org/"
  },
  "ledger": "cookbook/base",
  "where": { "@id": "?s", "schema:description": "We 💙 All Blood" },
  "delete": { "@id": "?s", "schema:description": "We 💙 All Blood" },
  "insert": {
    "@id": "?s",
    "schema:description": ["We 💙 Human Blood", "We 💙 Animal Blood"]
  }
}
```

- This is what a simple Fluree query looks like.

Simple Example

Note this is a quick guide to insert an ontology in to Fluree ledger, assume each entity is already on http://example.org/ as a URI.

Simple Ontology:

- Bob is a Farmer
- Farmer is an Occupation
- Bob is a Person
- Bob Drives a Tractor
- Tractor is a Vehicle
- Vehicle is an Object
- Bob works with Tom

- Tom is a Farmer
- Tom is a Person
- Bob is 53
- Tom is 43

Creating the Ledger in Fluree:

```
{
  "@context": {
    "ex": "http://example.org/",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
  },
  "ledger": "ex_ontology",
  "insert": [
    {
     "@id"  : "ex:Person",
     "@type" : "rdfs:Class",
     "rdfs:comment" : "A person is a human being, an individual member of the species Homo sapiens."
    },
    {
      "@id":"ex:Occupation",
      "@type": "rdfs:Class",
      "rdfs:comment" : "An occupation is a job or profession."
    },
    {
      "@id":"ex:Farmer",
      "@type": "rdfs:Class",
      "rdfs:subClassOf" : {"@id" : "Occupation"},
      "rdfs:comment" : "A farmer is a person who works in agriculture, cultivating crops and/or raising animals."
    },
    {
      "@id":"ex:Object",
      "@type":"rdfs:Class",
      "rdfs:comment" : "An object is a material thing that can be seen and touched."
    },
    {
      "@id" : "Vehicle",
      "@type" : "rdfs:Class",
      "rdfs:subClassOf" : {"@id" : "Object"},
      "rdfs:comment" : "A vehicle is a means of transporting people or goods, such as a car, truck, or bicycle."
    }
  ]
}
```

- This creates an Ontology from the natural language with a foundation to insert the necessary "records" into the database.

Inserting a single record into the Fluree database:

```
{
  "@context": {
    "ex": "http://example.org/",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
  },
  "ledger": "ex_ontology",
  "insert": [
    {
      "@id": "ex:Bob",
            "@ex:name": "ex:Bob"
      "@type": "ex:Farmer",
      "rdfs:comment": "Bob the Farmer",
      "ex:worksWith": {"@id" : "ex:Tom"},
      "ex:age" : 53
    }
  ]
}
```

- This inserts "Bob who is 53 and a Farmer, Bob also works with Tom." Into the Ledger "**ex_ontology".**

Query to view all Farmers:

- The query:

```
{
  "@context": {
    "ex": "http://example.org/",
    "schema": "http://schema.org/"
  },
  "from": "ex_ontology",
  "where": {
    "@id": "?s",
    "@type": "ex:Farmer"
  },
  "select": {"?s": ["*"]}
}
```

- This shows all the "attributes" of someone who is of type **Farmer.**
    - The result:

```
-   [
-     {
-       "@type": "ex:Farmer",
-       "http://www.w3.org/2000/01/rdf-schema#comment": "Bob the Farmer",
```

```
-           "ex:age": 53,
-           "ex:married": true,
-           "ex:worksWith": {
-              "@id": "ex:Tom"
-           },
-           "@id": "ex:Bob"
-        }
-     ]
```

- Bob was the only Farmer added to the Database.

Inserting a new class in Database:

```
{
  "@context": {
    "ex": "http://example.org/",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
  },
  "ledger": "ex_ontology",
  "insert": [
    {
      "@id": "ex:Location",
      "@type": "rdfs:Class",
      "rdfs:comment": "Represents a physical or conceptual place."
    }
  ]
}
```

- Adds **"Location"** as a class.
- **Subclasses** of Location such as City or Town can now be added.
- Provided in http://example.org/", there exists an IRI named Location.

Inserting Sub class of Location:

```
-    {
-       "@context": {
-          "ex": "http://example.org/",
-          "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
-       },
-       "ledger": "ex_ontology",
-       "insert": [
-          {
-             "@id": "ex:Country",
-             "@type": "rdfs:Class",
-             "rdfs:subClassOf" : {"@id": "ex:Location"},
-             "rdfs:comment": "Represents a physical or conceptual place."
```

```
-          }
-       ]
-    }
```

- Country is added as a Subclass of Location and now we can insert data records into this Subclass.

Inserting Data to new Subclass

```
{
  "@context": {
    "ex": "http://example.org/",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
  },
  "ledger": "ex_ontology",
  "insert": [
    {
      "@id": "ex:South-Africa",
      "@type": "@ex:Country",
      "rdfs:comment": "The country South-Africa."
    }
  ]
}
```

- The JSON-LD statement above inserts a new record from the example vocabulary "ex: South-Africa".
- If the record is not in any vocabulary and exists only in this database, the "@id" does not need to include a prefix.
    - "@id" : "South-Africa" (This is not linked data.)

Creating RDF Properties in Fluree

- Bob is **married:**

```
-    {
-       "@context": {
-          "ex": "http://example.org/",
-          "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
-          "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
-       },
-       "ledger": "ex_ontology",
-       "insert": [
-          {
-             "@id": "ex:married",
-             "@type": "rdf:Property",
-             "rdfs:comment": "Whether a Person is married or not"
```

```
-         },
-         {
-            "@id" : "ex:Bob",
-            "ex:married": true
-         }
-      ]
-   }
-
```

- This updates Bob's marital status to **Married**, if Bob did not have a marital status, it is now assigned as Married.
- What is the output now if we query Bob:

```
-   [
-      {
-         "@type": "ex:Farmer",
-         "http://www.w3.org/2000/01/rdf-schema#comment": "Bob the Farmer",
-         "ex:age": 53,
-         "ex:married": true,
-         "ex:worksWith": {
-            "@id": "ex:Tom"
-         },
-         "@id": "ex:Bob"
-      }
-   ]
```