InteGraal : Data Integration with Rules and Reasoning

Summary of year 2021-2022

Florent TORNIL

September 2022

Boreal, INRIA, LIRMM

Table of contents

Features added in 2021-2022

Work of 2021-2022

Internal Storage

- Storing atoms with a DBMS
  - Drivers SQL : Postgres, MySQL, HSQL, SQLite
    - 1 table per predicate + term encoding with integers
  - Drivers RDF/Repository
    - Remote SPARQL endpoint
    - Local in-memory triplestore
- Evaluating queries and rules
  - Default : match(atom) + backtrack
  - Delegation to DBMS
    - CQ-to-SQL
    - CQ-to-SPARQL
    - Datalog rules as INSERT queries (SQL/SPARQL)

Work of 2021-2022

Functions, built-in predicates, and negation

- Work based on Riadh's internship (M2 - 2021)
- Deployment : Elie's thesis, INRAe use-case from David Camarazo internship (M2 - 2022)
- Added to the DLGP parser and to InteGraal
- Built-in predicates and negation used as a filter
- Added to the reasoning (forward chaining only)

Example

evenLengthConcept(C) :-
concept(C), hasLabel(C, L),
fct :isEven(fct :length(L)),
not fct :isEmpty(L).

Work of 2021-2022

Mappings

- Deployment : Elie's thesis, INRAe use-case from David Camarazo internship (M2 - 2022), DFKI use-case
- Functionalities
  - Two parts : (1) view definition + (2) mapping-rules
  - Associate a native query on a datasource to a relational view
  - Materialization as well as virtualization approaches
- Mapping format
  - Use-case driven
  - JSON-based syntax
  - Aim for a simple syntax with possibilities to enrich it
  - Not finished yet

Work of 2021-2022

Rewriting and GRD

- Unification adapted to the new API + data structure improvement
- GRD bridge and then adaptation
- Rewriting (beginning of)
  - Bridge with Graal v1.3 rewriting as a first quick solution
  - Import of the main algorithm using rewriting operators
  - Import of a first rewriting operator (SRA)
  - Missing the other operators

Mediation

- Tatooine (detailed next)

Work of 2021-2022

Other

- Debugging
- Documentation
- Test

Valorization

- Demonstrations (Anabasis, CQFD, DFKI - R4Agri, INRAe - TCalis-Fair, Odalim, Open-Silex, SED)
- Used for internships (David Camarazo, Lucas) / thesis (Aziz, Elie)
- Licence (Apache 2)
- Software registration (dépôt APP) - this week

Last feature : conjunctive query evaluation over federated databases using the Tatooine mediator

Motivations

Goal : Answering a conjunctive query (on the vocabulary of the views) over a federated factbase

Default approach

- Backtrack
- Each atom is evaluated over the (virtual) federated factbase
- The federated factbase dispatches the evaluation of the atomic queries to the storage systems

Use an External Mediator

- Reuse optimizations developed for the mediator that our backtrack does not have

The Tatooine Mediator

Tatooine

- Datasource mediator and federated query evaluator
- Made by INRIA/Cedar team (and Maxime)
- Obi-wan is an example tool on how to use it

Tatooine : What does it do ?

Input

Logical query plan (defined in the next slides)

Operations

- (Optionally) Optimizes the logical plan
- Chooses an associated physical plan
- Evaluates the (physical) plan

Output

Result Tuples (each position has a column name and a type)

Federated Query Evaluation - Example

Datasources for the scenario

1. DBPedia SPARQL endpoint : we will use the **Concept** class
2. PostgreSQL database : provides metadata on the concepts
   - **Label** of a Concept
   - **Owner** of a Label

View-Definition

- $Q_1^{SPARQL} \rightsquigarrow vDbpediaConcept(Concept)$
- $Q_2^{SQL} \rightsquigarrow vPgresConcept(C, Label)$
- $Q_3^{SQL} \rightsquigarrow vPgresOwner(L, Owner)$

Federated Query Evaluation - Example

View-Definition

- $Q_1^{SPARQL} \rightsquigarrow vDbpediaConcept(Concept)$
- $Q_2^{SQL} \rightsquigarrow vPgresConcept(C, Label)$
- $Q_3^{SQL} \rightsquigarrow vPgresOwner(L, Owner)$

Mapping-Rules

- $vDbpediaConcept(Concept) \rightarrow concept(Concept)$
- $vPgresConcept(C, Label) \rightarrow hasLabel(C, Label)$
- $vPgresOwner(L, Owner) \rightarrow hasOwner(L, Owner)$

Federated Query Evaluation - Example

Mapping-Rules

- $vDbpediaConcept(Concept) \rightarrow concept(Concept)$
- $vPgresConcept(C, Label) \rightarrow hasLabel(C, Label)$
- $vPgresOwner(L, Owner) \rightarrow hasOwner(L, Owner)$

Query (on the ontology vocabulary)

```
?(Concept, Label, Owner) :-
        concept(Concept),
        hasLabel(Concept, Label),
        hasOwner(Label, Owner).
```

Federated Query Evaluation - Example

Query (on the ontology vocabulary)

```
?(Concept, Label, Owner) :-
        concept(Concept),
        hasLabel(Concept, Label),
        hasOwner(Label, Owner).
```

Rewriting into the vocabulary of the views

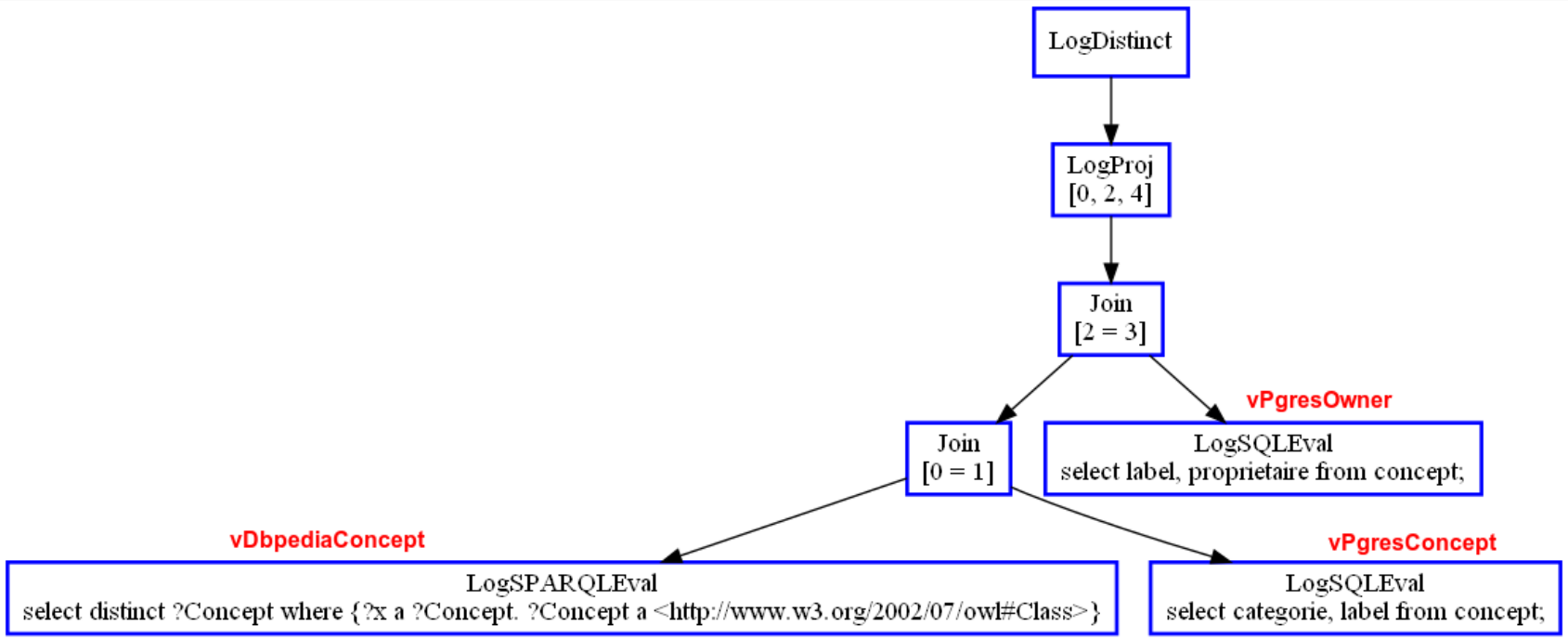
```
?(Concept, Label, Owner) :-
        vDbpediaConcept(Concept),
        vPgresConcept(Concept, Label),
        vPgresOwner(Label, Owner).
```

Maxime's algorithm for 1-step mapping rewriting added into InteGraal

Logical Plan Example

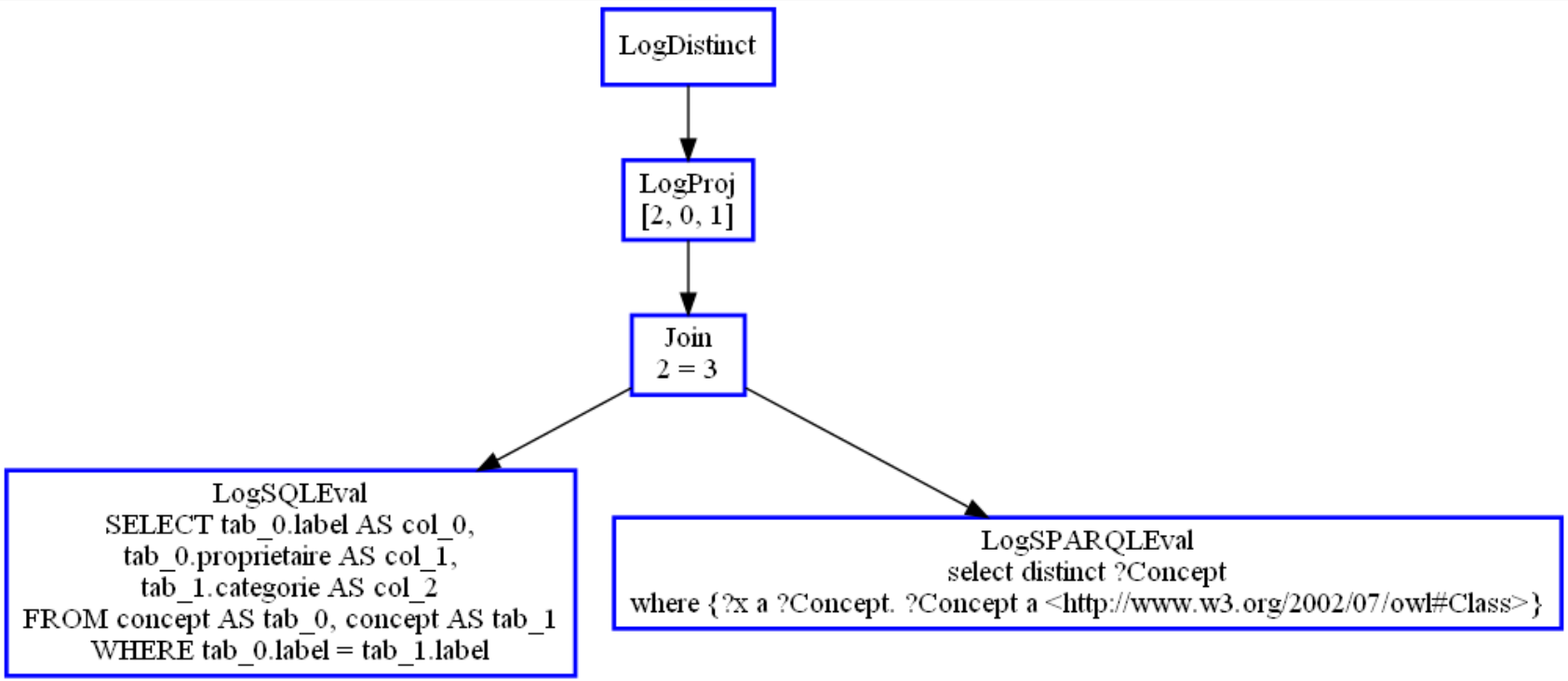
InteGraal can now build a Logical Plan from a query

```
?(Concept, Label, Owner) :-
        vDbpediaConcept(Concept),
        vPgresConcept(Concept, Label),
        vPgresOwner(Label, Owner).
```



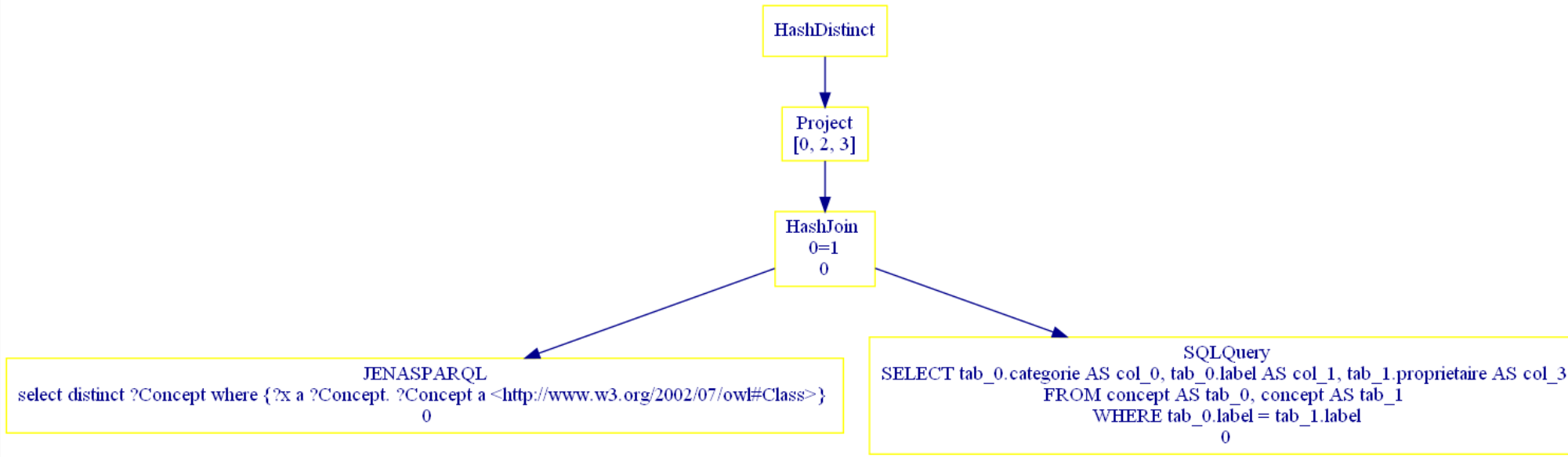
Inside Tatooine : Logical Plan Optimization

- Tatooine regroups the queries on the same datasource (whenever this is possible)
- Tatooine pushes operations such as selections and joins (whenever the underlying system allows for it)

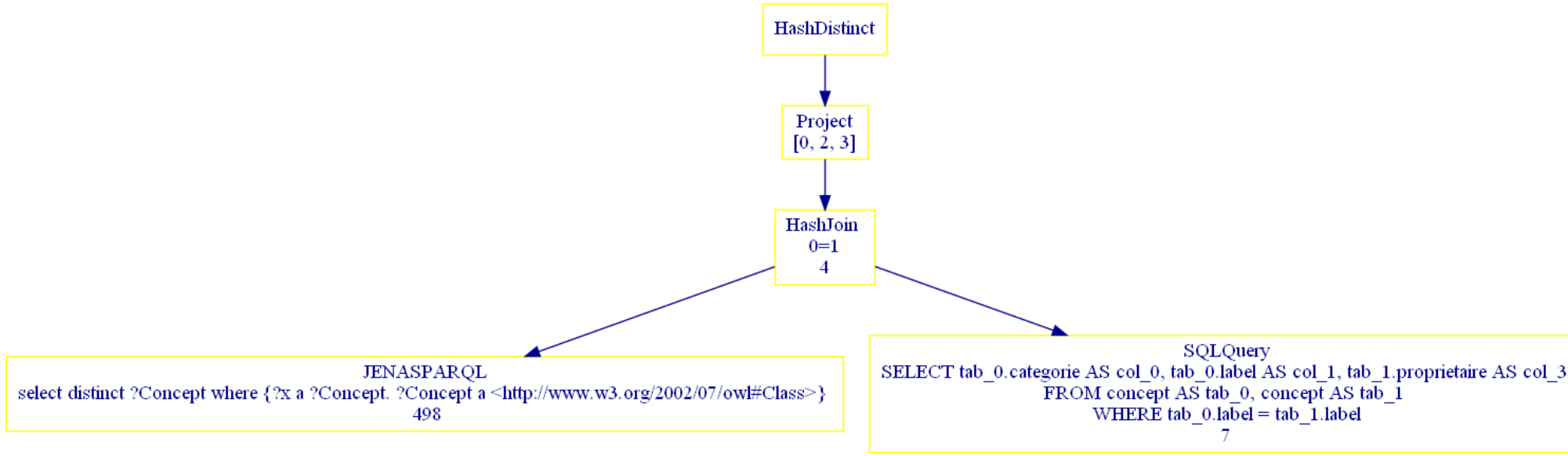


Inside Tatooine - Physical Plan

- Tatooine chooses the implementation to use for each logical operator



Inside Tatooine - Statistics on the Physical Plan Evaluation



Implementation : What Has Been Added into InteGraal

Extensions to integraal-model module with mediation

- Interfaces
  - Plan
  - PlanFactory (operator-to-plan)
- Algorithm Query-to-plan (FOQueryToPlanFactory)
  - uses a PlanFactory as parameter to adapt for the target mediator

New module integraal-tatooine

- TatooinePlan (implements Plan)
- TatooinePlanFactory (implements PlanFactory)
- TatooineQueryEvaluator (implements QueryEvaluator)

Implementation : What Has Been Added into InteGraal

Modeling Choices

- Optionality : InteGraal-Tatooine is an optional module
- Dependencies : no dependency with tatooine beside InteGraal-Tatooine module
- Genericity : the TatooineQueryEvaluator can be used anywhere instead of the backtrack (QueryEvaluator interface).
- Extensibility : to add a new mediator, you just have to implement the following interfaces :
  - Plan
  - Plan factory
  - Query Evaluator

Implementation (2) : Which datasources can be used with Tatooine

SQL Postgres, SQLite (no HSQL or MySQL)

RDF SPARQL endpoint only (no in-memory)

MongoDB

InteGraal objects for MongoDB mappings are not stable yet

InteGraal has no native MongoDB storage

Blackbox

- Tatooine only requires that tuples are returned
- Tatooine allows the system to include novel (non-optimizable) sources (eg, Web-API)
- Designed for evolutivity (adding optimizations)
- InteGraal has no real equivalent for blackboxes but this can be seen as the mapping datasource (we can only execute the query and get the results)

Features to be added in 2022-2023 - discussions for the roadmap

Road-map for 2022-2023

Objectives

- First stable release for the end of the year 2022
  - Import most important missing functionalities from Graal v1.3 (next slide)
  - Mapping format : define the syntax + parser and factory implementation
  - Documentation and test

- Define the Roadmap
  - List all the remaining tasks (next slide)
  - Prioritize the tasks
    - Important features of Graal v1.3
    - Which projects, thesis, internships ?
    - Estimate task duration

We have to define the roadmap together

Missing functionalities of Graal v1.3

- Backtrack optimisations
- Compilation
- Core computation
- DLGP export
- Kiabora usage
- Metachase
- Rewriting operators

Other tasks

- Benchmark
- Documentation and test
- Mapping format
- Refactoring of the DLGP parser and grammar extension
- Stratification
- Update Kiabora with InteGraal
- Valorization (example application, website, releases, support, tutorials)

We will define all of this in a dedicated meeting : **WHEN ?**

Next week : Tuesday ?