Everything You Always Wanted to Know About Your Process, But Did Not Know How To Ask

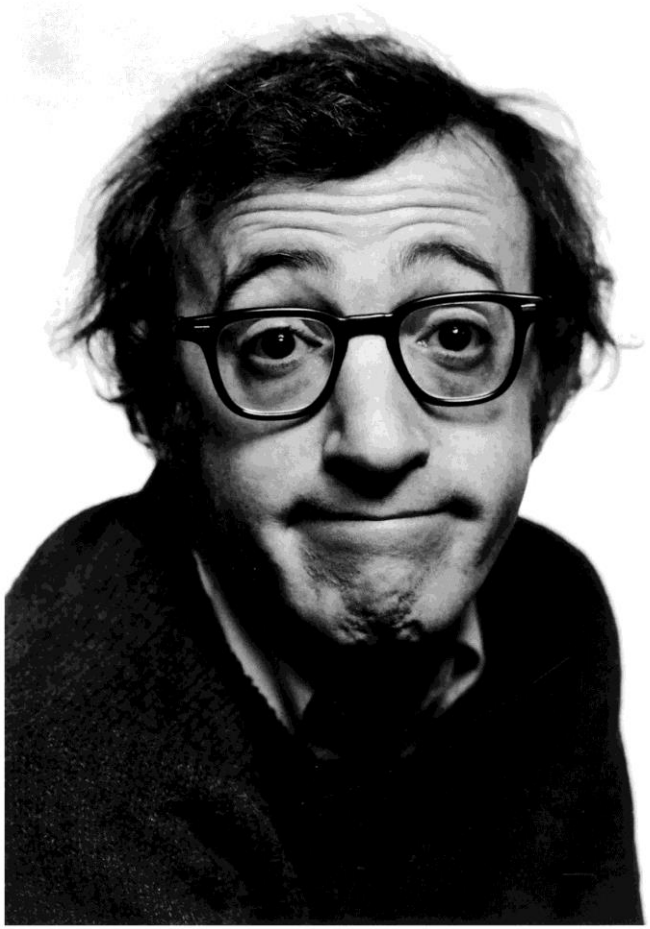
Eduardo González López de Murillas
Hajo Reijers
Wil van der Aalst

International Workshop on Process Querying 2016
September 19th, 2016
Rio de Janeiro, Brazil

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Making sense out of the big mess



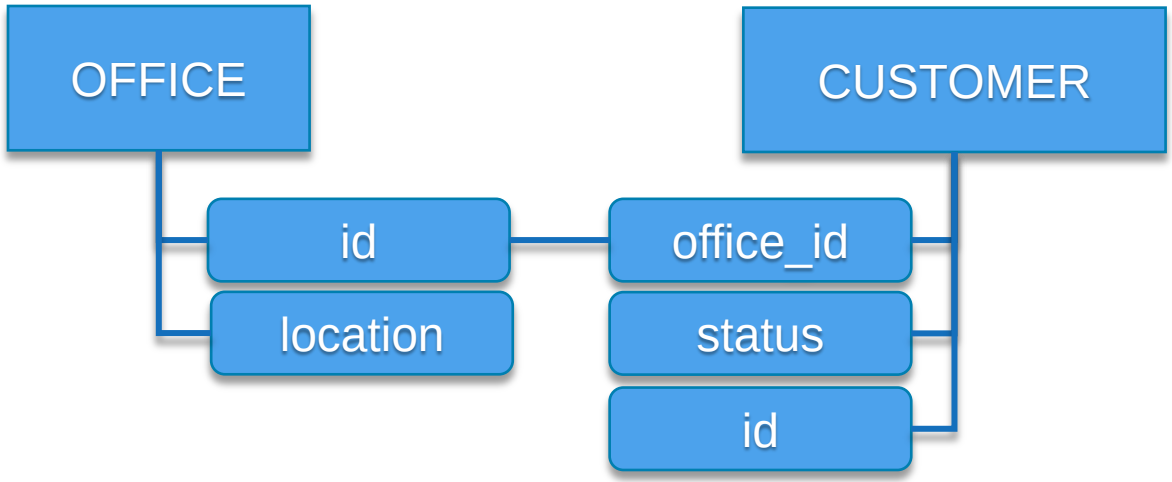
"Can we actually "know" the universe? My God, it's hard enough finding your way around in Chinatown."

We need to find the needle in the haystack

Look in the right place

Querying can help us

Simple example



Customers for which the status value has changed from Silver to Gold membership, and are linked to an office located in Madrid.

Using SQL on the underlying Meta Model

```
SELECT OBJCust.id
 FROM object as OBJCust,
    class as CCust,
    class as COff,
    version as OBJVSilver,
    version as OBJVGold,
    version as OBJVMadrid,
    attribute_value as ATVStatusSilver,
    attribute_value as ATVStatusGold,
    attribute_name as ATStatus,
    attribute_value as ATVLocation,
    attribute_name as ATLocation,
    object as OBJOff,
    class as COff,
    version as OBJVOff,
    attribute_value as ATVLocation,
    attribute_name as ATLocation,
    relation as REL
 WHERE OBJCust.class_id == CCust.id
  AND  CCust.id == ATStatus.class_id
  AND  ATStatus.name == "status"
  AND  ATVStatusGold.object_version_id == OBJVGold.id
  AND  ATVStatusSilver.object_version_id == OBJVSilver.id
  AND  ATStatus.id == ATVStatusGold.attribute_name_id
  AND  ATStatus.id == ATVStatusSilver.attribute_name_id
  AND  ATVStatusGold.value == "Gold"
  AND  ATVStatusSilver.value == "Silver"
  AND  OBJVGold.object_id == OBJCust.id
  AND  OBJVSilver.object_id == OBJCust.id
  AND  OBJVGold.start_timestamp >= OBJVSilver.end_timestamp

  AND  COff.name == "OFFICE"
  AND  OBJOff.class_id == COff.id
  AND  OBJVOff.object_id == OBJOff.id
  AND  ATLocation.name == "Location"
  AND  ATLocation.class_id == COff.id
  AND  ATVLocation.attribute_name_id == ATLocation.id
  AND  ATVLocation.object_version_id == OBJVOff.id

  AND  (REL.source_object_version_id == OBJVOff OR
REL.target_object_version_id == OBJVOff)
  AND  (REL.source_object_version_id == OBJVSilver OR
REL.source_object_version_id == OBJVGold)
  AND  (REL.target_object_version_id == OBJVSilver OR
REL.target_object_version_id == OBJVGold)
```

In DAPOQ-Lang

```
objectsOf(
        versionsRelatedTo(
                versionsOf(
                        allClasses where name == "OFFICE"
                ) where at.Location == "Madrid"
        ) where at.status changed from "Silver" to "Gold"
)
```

Customers for which the status value has changed from Silver to Gold membership, and are linked to an office located in Madrid.

Main Trends on Querying Execution Data

- **Data Provenance Oriented**
  - **Origin or source of data**

- **Business Process Oriented**
  - **Event data**
  - **Process models**

		Input data					Provenance				BP	DB			Query aspects									
Ref	Title	Event data	Model-based	Storage model	Complex event processing	Streams	Provenance-oriented	OPM-compliant	Data lineage	Depencency graphs/Relations	Business process oriented	Entities/Artifacts	Database-oriented	State-aware	Graph-based	Relevance querying	Semantic querying	Regular path queries (RPQ)	Projection queries	Temporal properties/querying	Event correlation	Multi process	Multi log	Multi data schema
[3]	Answering regular path queries on workflow provenance	✓					✓								✓			✓						
[4]	Capturing and querying workflow runtime provenance with PROV: A practical approach	✓		✓			✓			✓														
[5]	Modeling and querying scientific workflow provenance in the D-OPM	✓		✓			✓	✓							✓			✓						
[6]	Towards a scalable semantic provenance management system	✓					✓	✓	✓	✓							✓							
[7]	Towards integrating workflow and database provenance	✓		✓			✓	✓	✓	✓		✓	✓	✓										
[8]	MTCProv: A practical provenance query framework for many-task scientific computing	✓		✓			✓	✓	✓	✓														
[9]	OPQL: A first OPM-level query language for scientific workflow provenance	✓					✓	✓	✓						✓									
[10]	Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases	✓		✓			✓	✓	✓	✓														
[11]	XQuery meets Datalog: Data relevance query for workflow trustworthiness	✓					✓			✓						✓								
[12]	A model for user-oriented data provenance in pipelined scientific workflows	✓		✓			✓		✓	✓														
[13]	A knowledge driven approach towards the validation of externally acquired traceability datasets in supply chain business processes	✓							✓		✓						✓							
[14]	Process query language: A way to make workflow processes more flexible	✓		✓					✓		✓													
[15]	Workflow History Management	✓		✓							✓													
[16]	The HIT model: Workflow-aware event stream monitoring	✓	✓		✓	✓					✓													
[17]	Semantic Enabled complex Event Language for business process monitoring	✓			✓				✓		✓				✓		✓		✓	✓				
[18]	A framework supporting the analysis of process logs stored in either relational or NoSQL DBMSS	✓		✓							✓				✓									
[19]	Business impact analysis: a framework for a comprehensive analysis and optimization of business processes	✓		✓						✓	✓	✓	✓											
[20]	Model-driven event query generation for business process monitoring	✓	✓		✓						✓													
[21]	Querying process models based on the temporal relations between tasks		✓								✓									✓				
[22]	A query language for analyzing business processes execution	✓							✓	✓	✓	✓			✓		✓				✓			
[23]	Top-k projection queries for probabilistic business processes	✓	✓	✓							✓								✓					
[24]	Integration of Event Data from Heterogeneous Systems to Support Business Process Analysis	✓	✓	✓							✓										✓			
[25]	Enabling semantic complex event processing in the domain of logistics	✓			✓						✓						✓							
[26]	Optimizing complex sequence pattern extraction using caching	✓			✓						✓								✓					
[27]	Log-based understanding of business processes through temporal logic query checking	✓																		✓				
	Our approach	✓	✓	✓			✓		✓	✓	✓	✓	✓	✓		✓				✓	✓	✓	✓	✓

What is missing?

1. **Combination of data and process aspects**
2. **Support for database particularities**
3. **Integration in the analysis workflow**

Why do we need another query language?

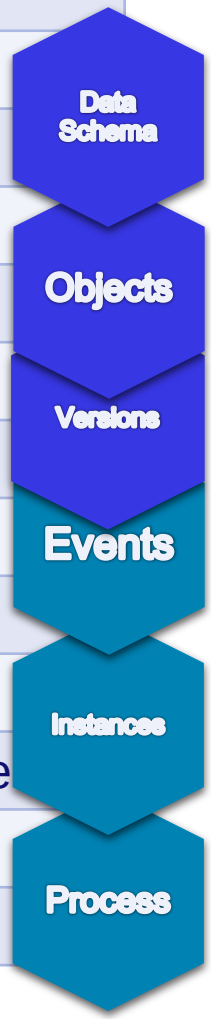
- **SQL is to Process Querying what Assembly is to programming.**

DAPOQ-Lang

- **Data Aware Process Oriented Query Language**
  - **Considers data and process perspectives**
  - **State aware**
  - **Easier to make common queries**
  - **Possible to combine with other analysis or filtering techniques (Integrated in the analysis workflow)**

DAPOQ-Lang I

Primitive elements
Datamodel
Class
Attribute
Relationship
Object
Version
Relation
Event
Case
Log
Activity Instance
Activity
Process



Process side

Data side

DAPOQ-Lang II

Functions	
datamodelsOf	Returs the set of requested elements (datamodels, classes, attributes, etc) related to the input elements. Input can be any of the primitive elements (and periods)
classesOf	
attributesOf	
relationshipsOf	
objectsOf	
versionsOf	
relationsOf	
eventsOf	
casesOf	
logsOf	
activityInstancesOf	
activitiesOf	
processesOf	

DAPOQ-Lang III

Functions	
versionsRelatedTo	Versions related to versions
periodsOf	Periods have a start and end timestamp
concurrentWith	With scopes: same case, same process, same object

Set Operations	
INTERSECTION	On sets of elements of the same type: cases with cases, versions with versions, etc.
UNION	
EXCLUDING	

Other constructs	
Foreach	On elements of a set
If Then Else	Condition checks if set is empty
Filters	On attributes of elements (versions, events, cases)

DAPOQ-Lang IV

Filters	
Common attributes	WHERE name == “CUSTOMER”
Specific attributes	WHERE at.Customer_name CONTAINS “Edu”
Versioning of values	WHERE at.Address CHANGED FROM “Madrid” to “Barcelona”
Scopes (Only for concurrency by now)	Select elements of same nature that share a common link to another type: concurrentWith(allActivities where name == “Pay”) scope case

Can we ask anything?

“I don't know the question, but sex is definitely the answer.”

We can express simple questions

allVersions() where { at.PRICE == "35" }

Or complex ones

```
def allC = allCases().where { id < 10} ;

def results = allActivities().where { false };

def eventsN = eventsOf(allActivities().where{ name.contains("Afwijzing") });

for (Object c : allC.result) {
        def ec = eventsOf(c);
        def e = eventsN.intersection(ec);
        def pe = globalPeriod(e);
        def ebef = ec.where { before(globalPeriod(it),pe) };
        results = results.union(activitiesOf(ebef));
}
return results
```

We do not need to have all the answers

- **If we get closer to the truth**

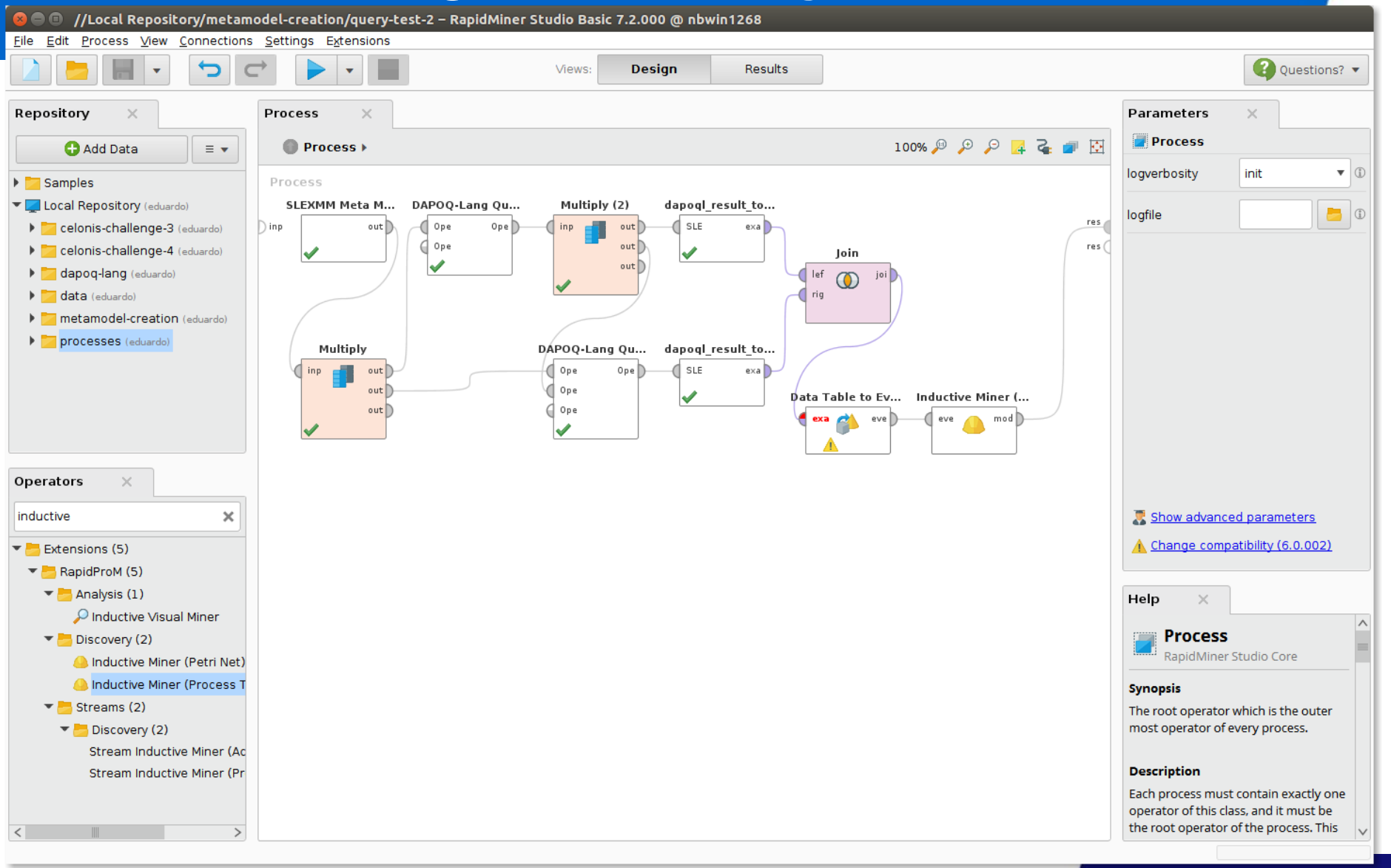
We do not need to have all the answers

- **If we get closer to the truth**
- **DAPOQ-Lang makes simple to select the relevant portion of the data**

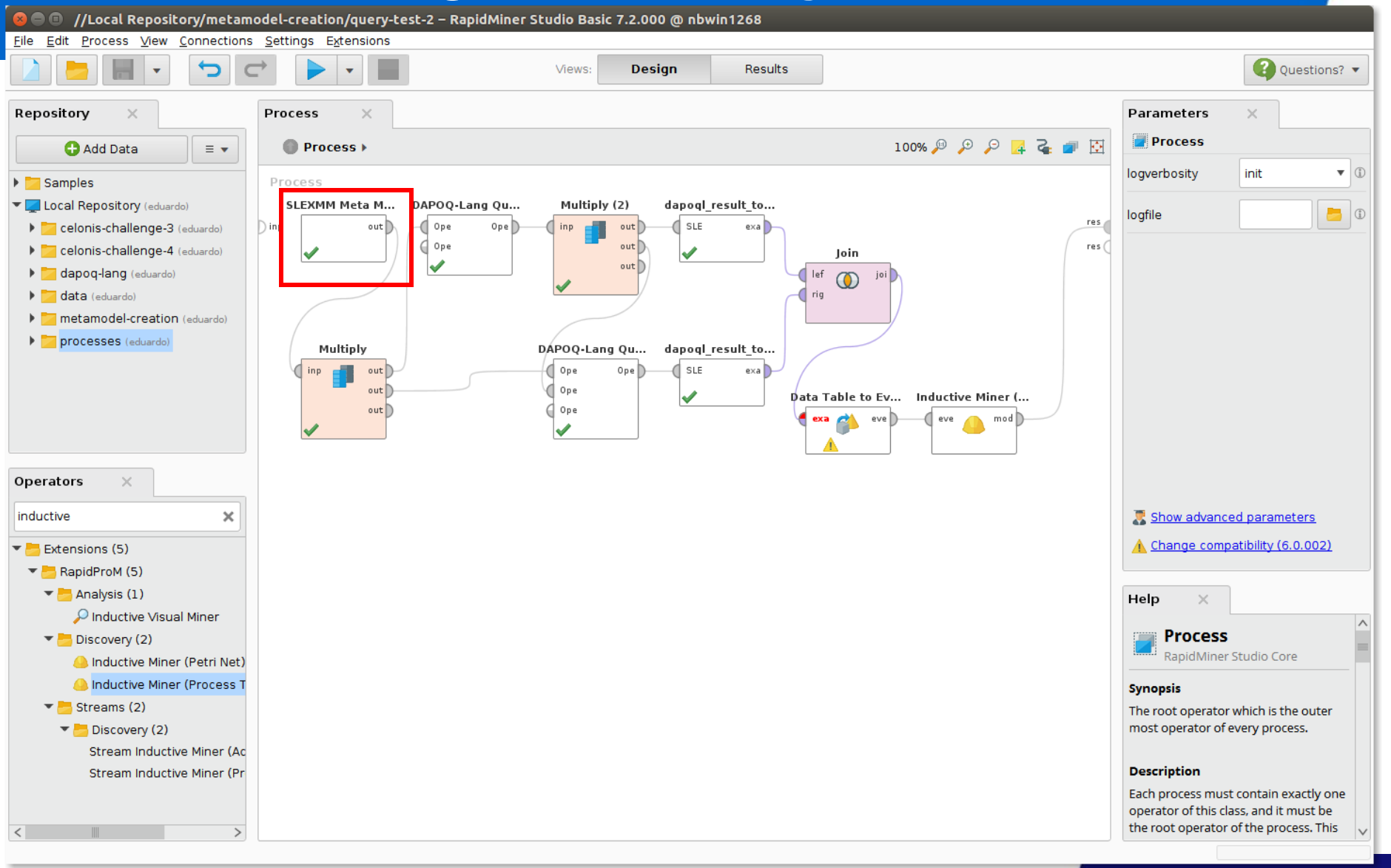
We do not need to have all the answers

- **If we get closer to the truth**
- **DAPOQ-Lang makes simple to select the relevant portion of the data**

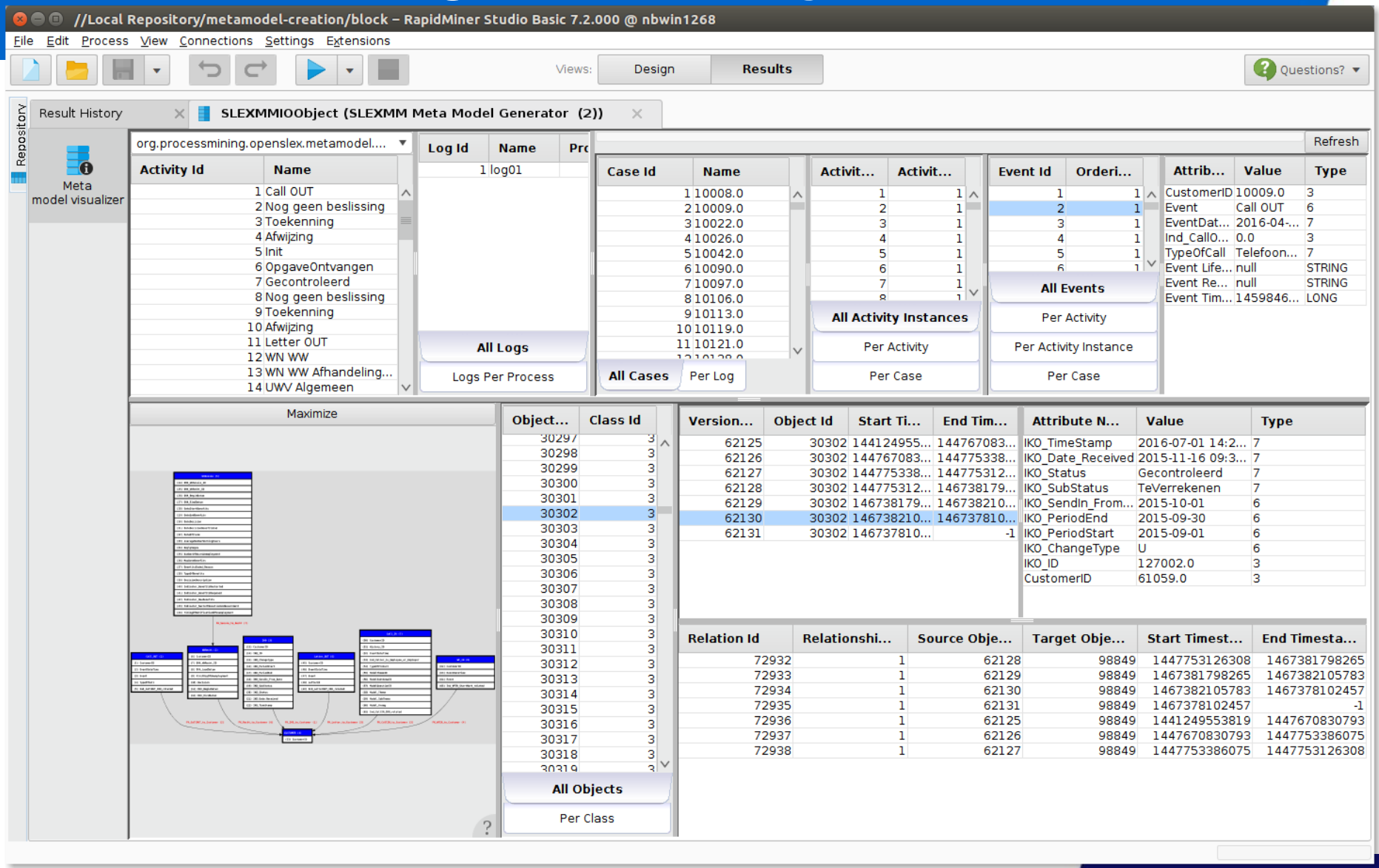
DAPOQ-Lang in the analysis workflow



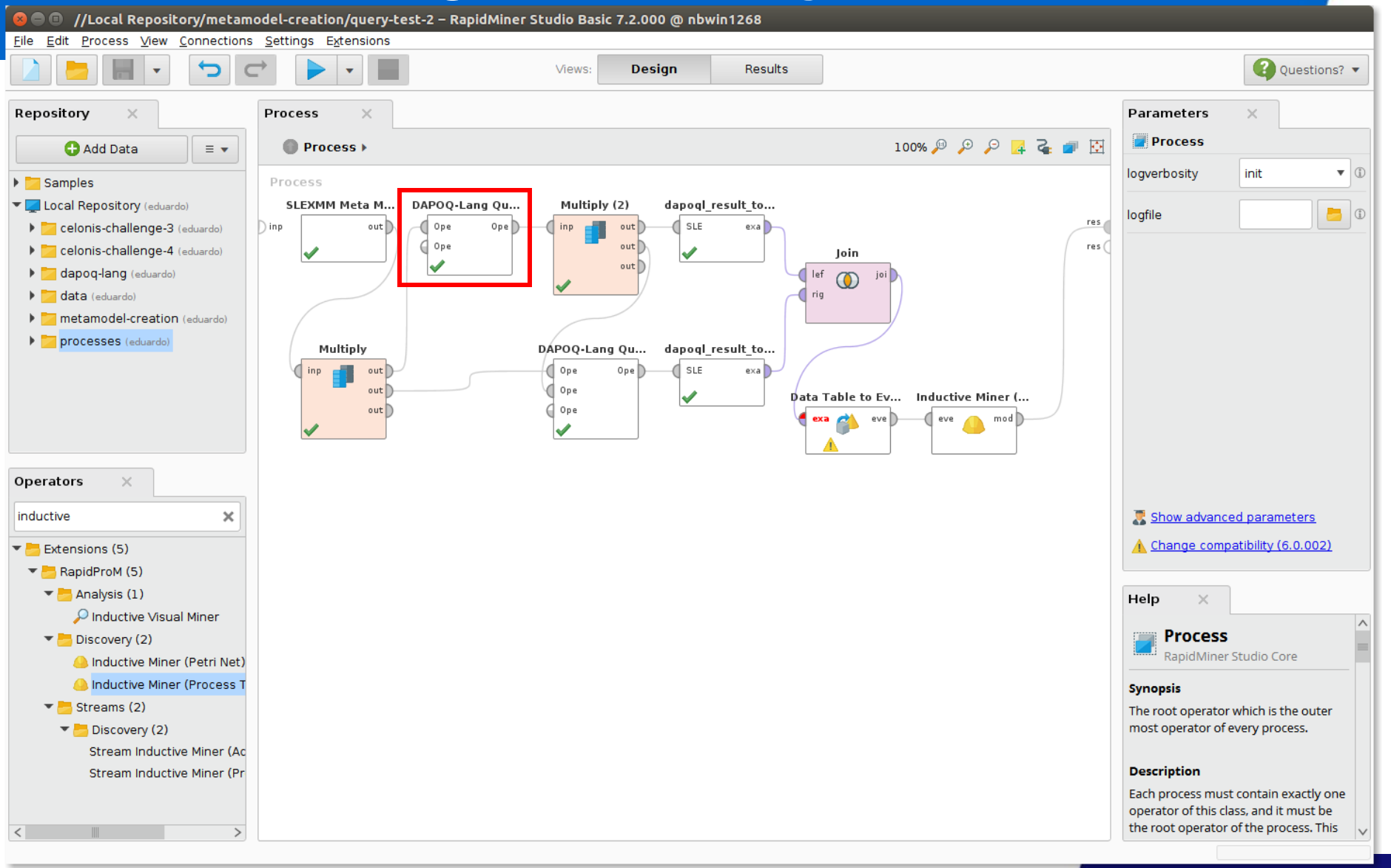
DAPOQ-Lang in the analysis workflow



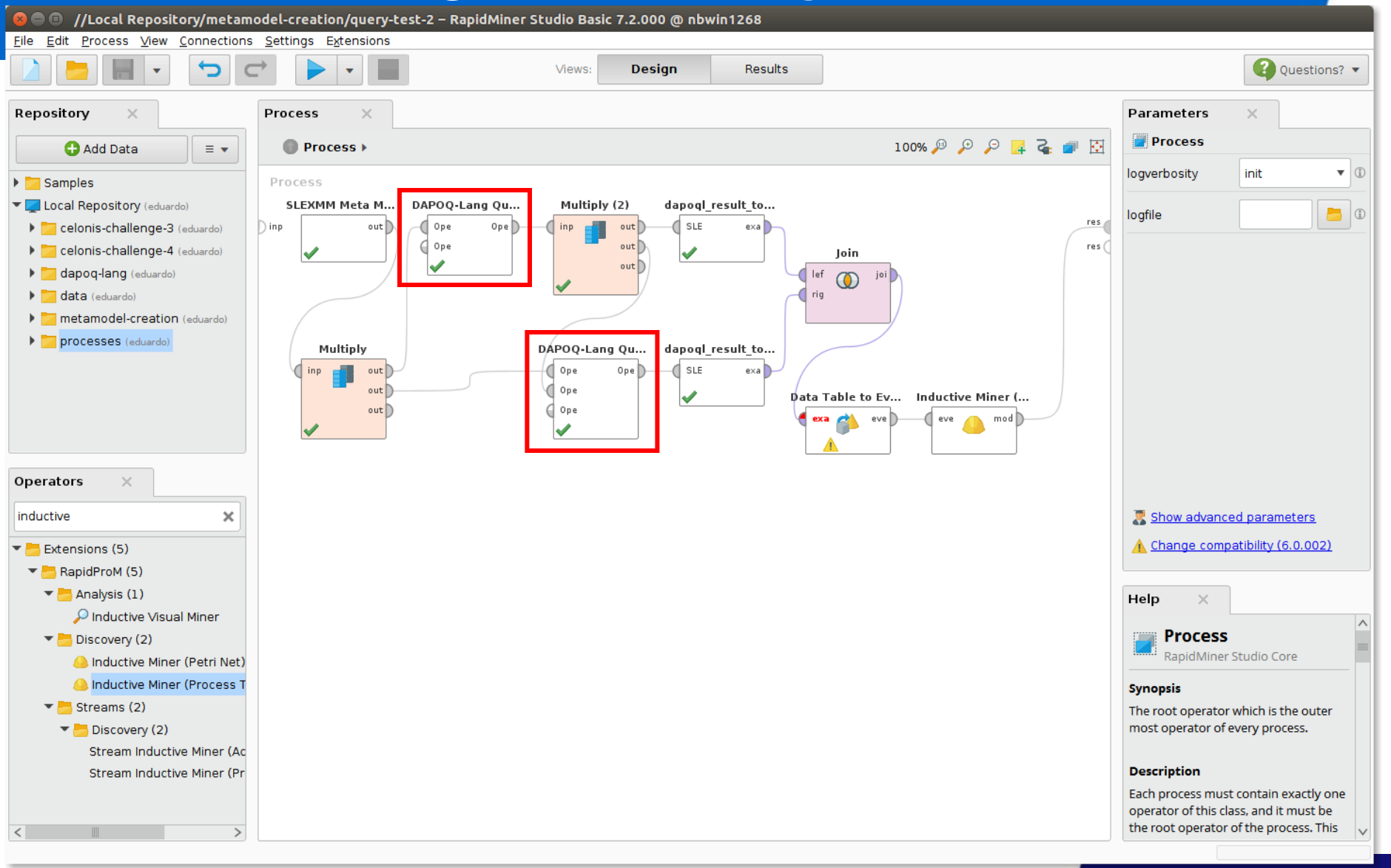
DAPOQ-Lang in the analysis workflow



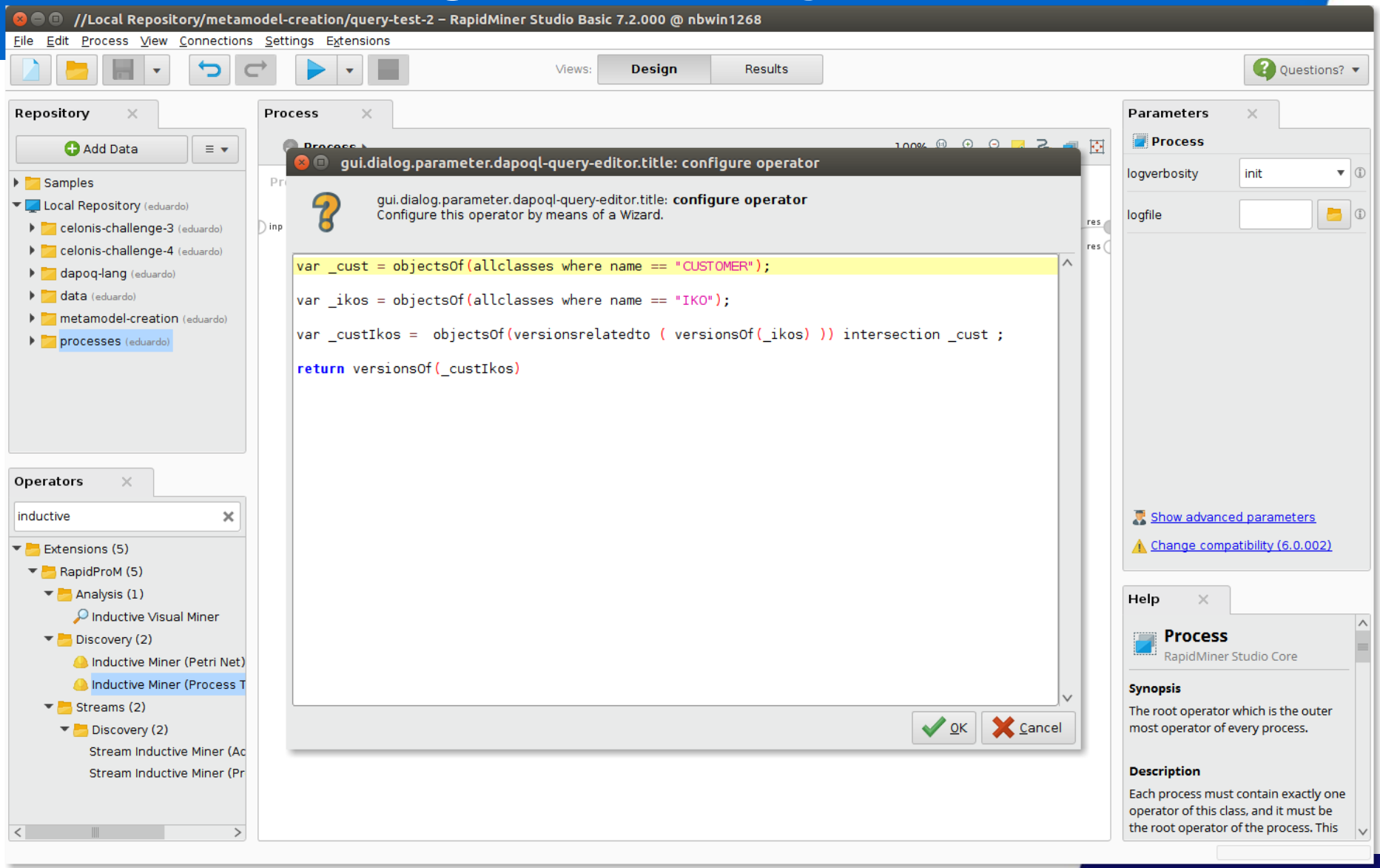
DAPOQ-Lang in the analysis workflow



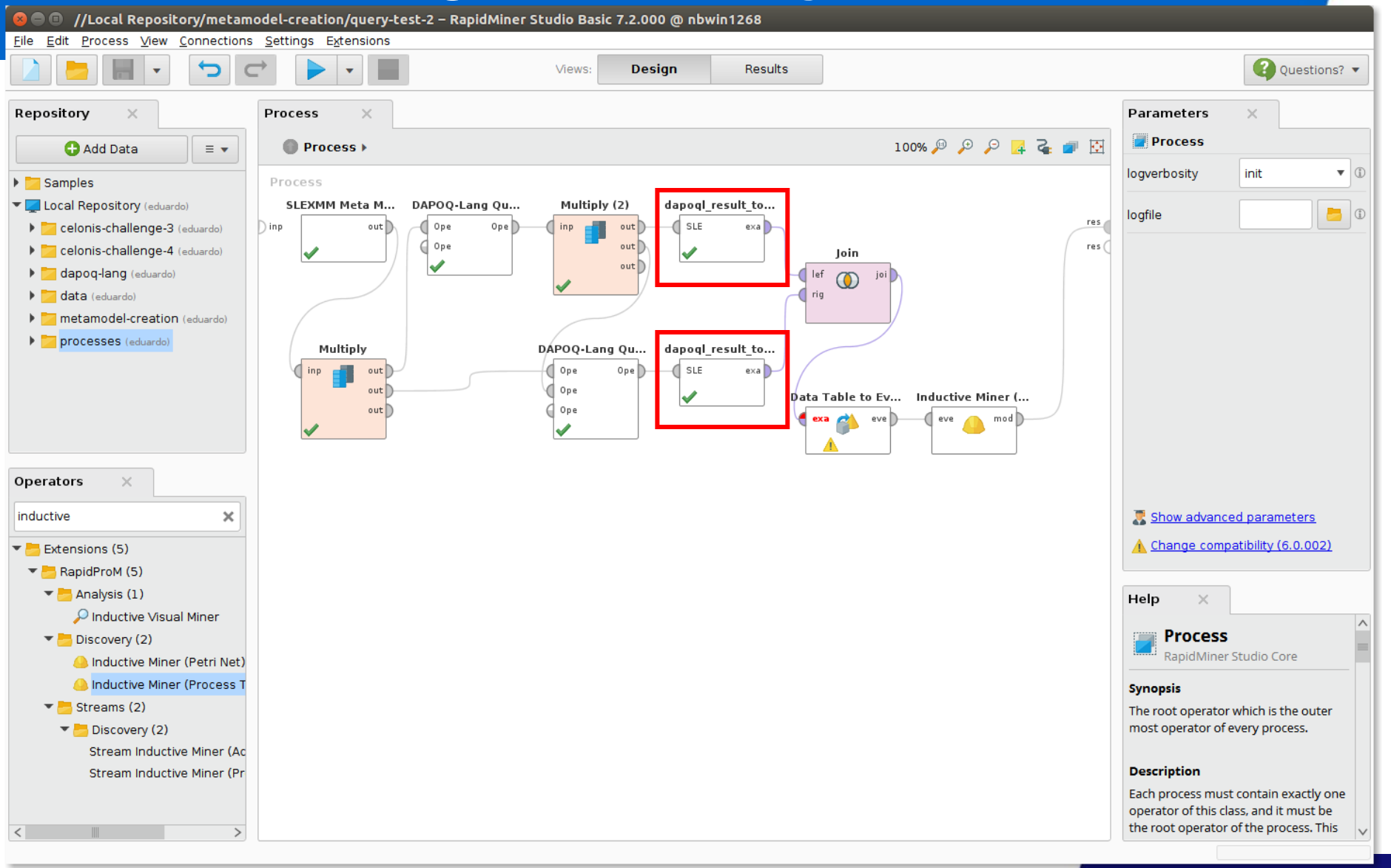
DAPOQ-Lang in the analysis workflow



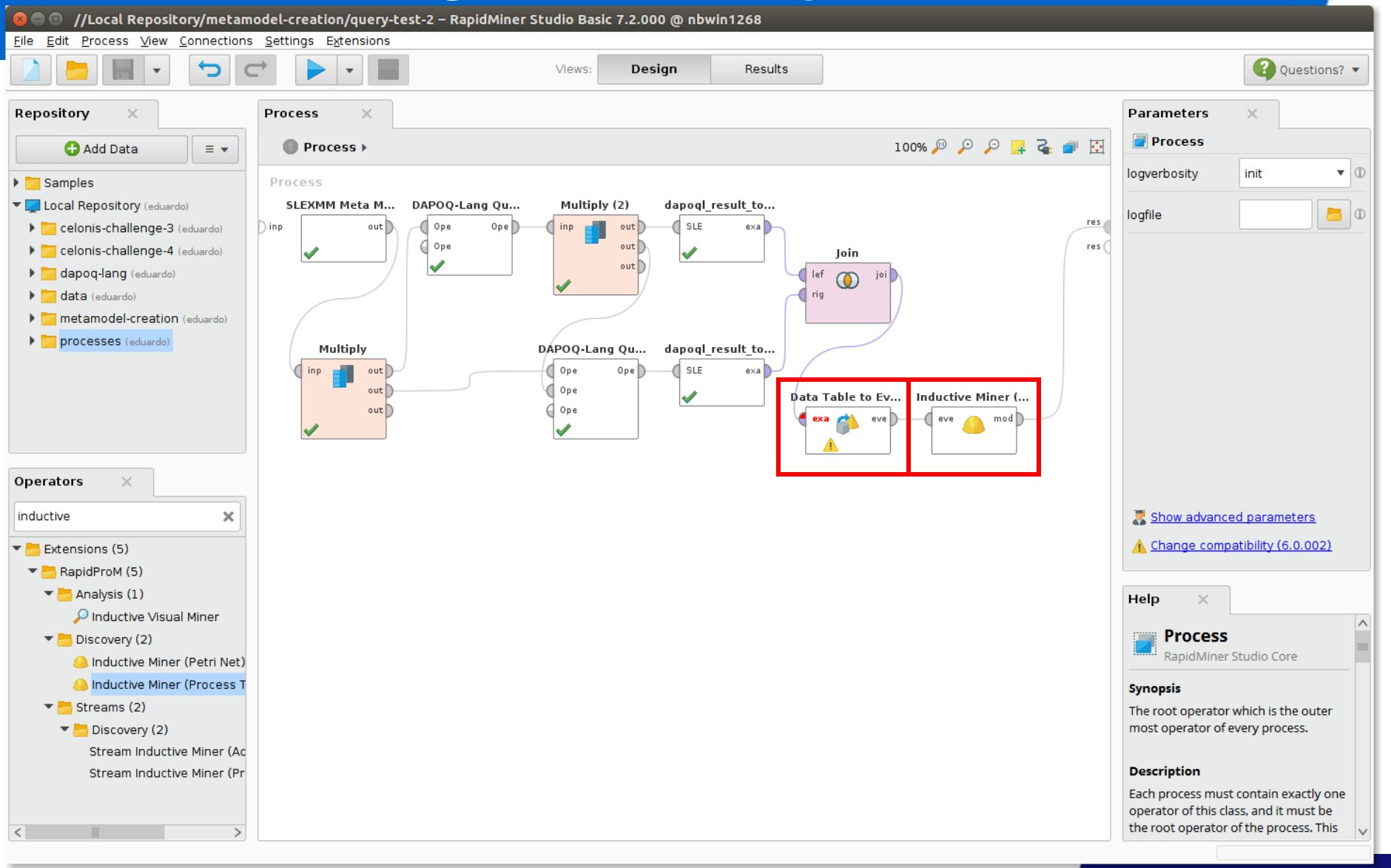
DAPOQ-Lang in the analysis workflow



DAPOQ-Lang in the analysis workflow



DAPOQ-Lang in the analysis workflow



Conclusion

- DAPOQ-Lang provides
  - Simplicity
  - Compactness
  - Suitability
- Can be used together with SQL
- Allows to focus on the analysis
- Makes easier to query and filter information
  - In the scope of Business Process Execution Data
- Integrated in a bigger tool:
  - One more element in the analysis workflow

Future work

- **The function collection is not complete**
  - **Evaluation with users to asses usefulness**
  - **Incorporate new constructs to simplify analysis**
- **Improve performance**
- **Include more operators in RapidProM to interact with other analysis tools**

Questions