Provenance challenge 3 and Taverna

Paolo Missier

Information Management Group

School of Computer Science, University of Manchester, UK

Provenance Challenge 3 meeting

Amsterdam, June 10-11, 2009

Interpreting the challenge

1. Implement the challenge workflow

2. Answer the core (+optional) challenge queries

3. Produce and export OPM

4. Import and consume OPM

Interpreting the challenge

1. Implement the challenge workflow

   ➡ Expose the *same* amount of data that is visible in the Trident version of the workflow

2. Answer the core (+optional) challenge queries

3. Produce and export OPM

4. Import and consume OPM

Interpreting the challenge

1. Implement the challenge workflow
   - ➡ Expose the *same* amount of data that is visible in the Trident version of the workflow
2. Answer the core (+optional) challenge queries
   - ➡ Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
3. Produce and export OPM
4. Import and consume OPM

Interpreting the challenge

1. Implement the challenge workflow
   - ➡ Expose the *same* amount of data that is visible in the Trident version of the workflow
2. Answer the core (+optional) challenge queries
   - ➡ Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
3. Produce and export OPM
   - ➡ Export the *smallest OPM graph that contains the query answer*
     - ➡(graph for the entire run is a special “degenerate” case)
4. Import and consume OPM

Interpreting the challenge

1. Implement the challenge workflow
   - ➡ Expose the *same* amount of data that is visible in the Trident version of the workflow
2. Answer the core (+optional) challenge queries
   - ➡ Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
3. Produce and export OPM
   - ➡ Export the *smallest OPM graph that contains the query answer*
     - ➡(graph for the entire run is a special “degenerate” case)
4. Import and consume OPM
   - ➡Map an OPM graph to an instance of a TP causal graph
   - ➡Use TP queries to answer the same challenge queries as above

Our challenges

Our challenges

- Expose the *same* amount of data that is visible in the Trident version of the workflow:
  - map the control flow to a pure dataflow model

Our challenges

- Expose the *same* amount of data that is visible in the Trident version of the workflow:
  - map the control flow to a pure dataflow model
- Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
  - challenge queries gave us good new requirements

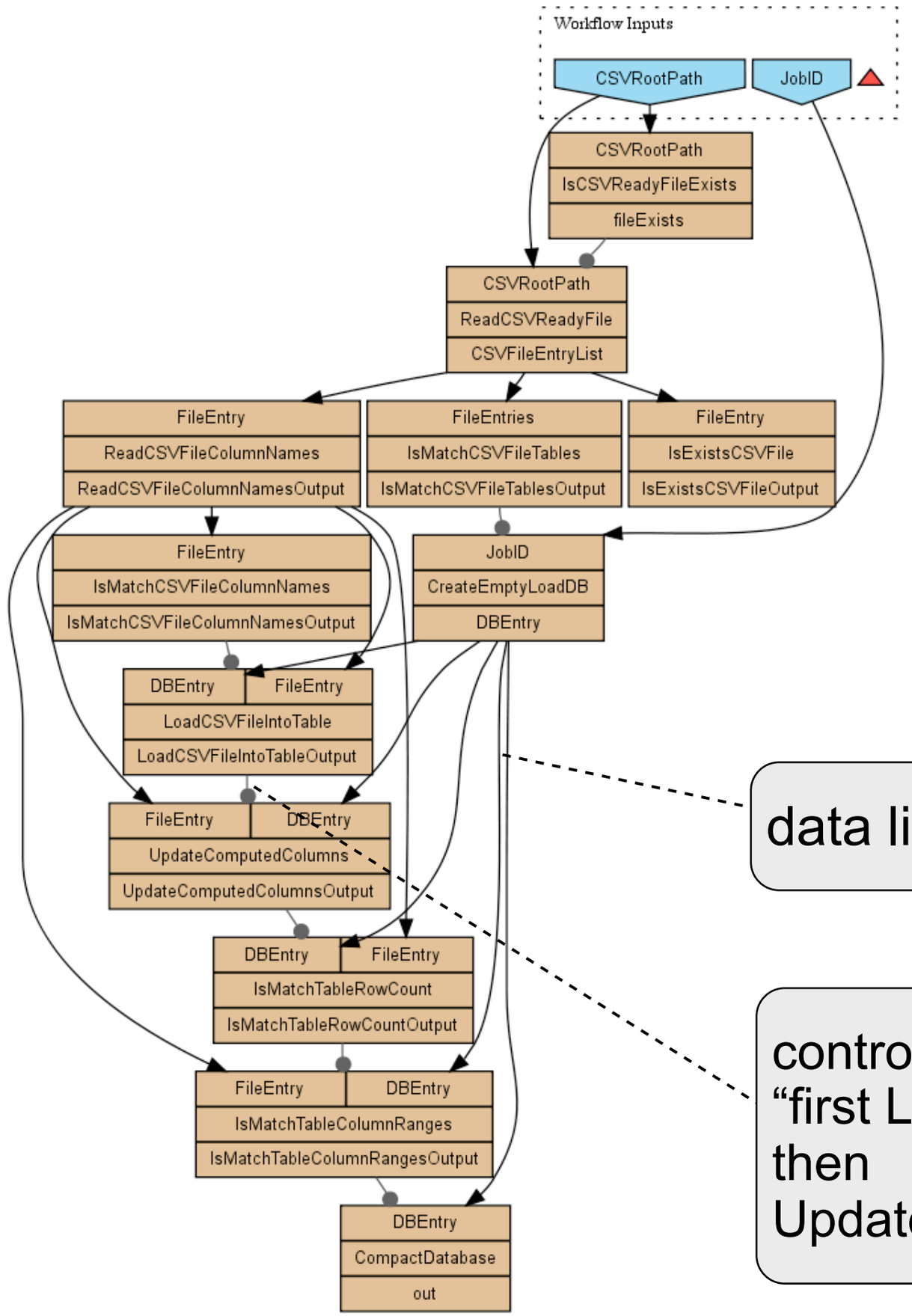
Our challenges

- Expose the *same* amount of data that is visible in the Trident version of the workflow:
  - map the control flow to a pure dataflow model
- Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
  - challenge queries gave us good new requirements
- export the *smallest OPM graph that contains the query answer*
  - OPM generation is now fully integrated into TP query answering

Our challenges

- Expose the *same* amount of data that is visible in the Trident version of the workflow:
  - map the control flow to a pure dataflow model
- Is the current Taverna Provenance (TP) query model sufficient to answer the queries?
  - challenge queries gave us good new requirements
- export the *smallest OPM graph that contains the query answer*
  - OPM generation is now fully integrated into TP query answering
- Map an OPM graph to an instance of a TP causal graph
- Use TP queries to answer the same challenge queries as above
  - TP query processing requires a representation of the *originating workflow*
  - This required generating the “minimal plausible originating dataflow” (**MPOD**) for the OPM graph

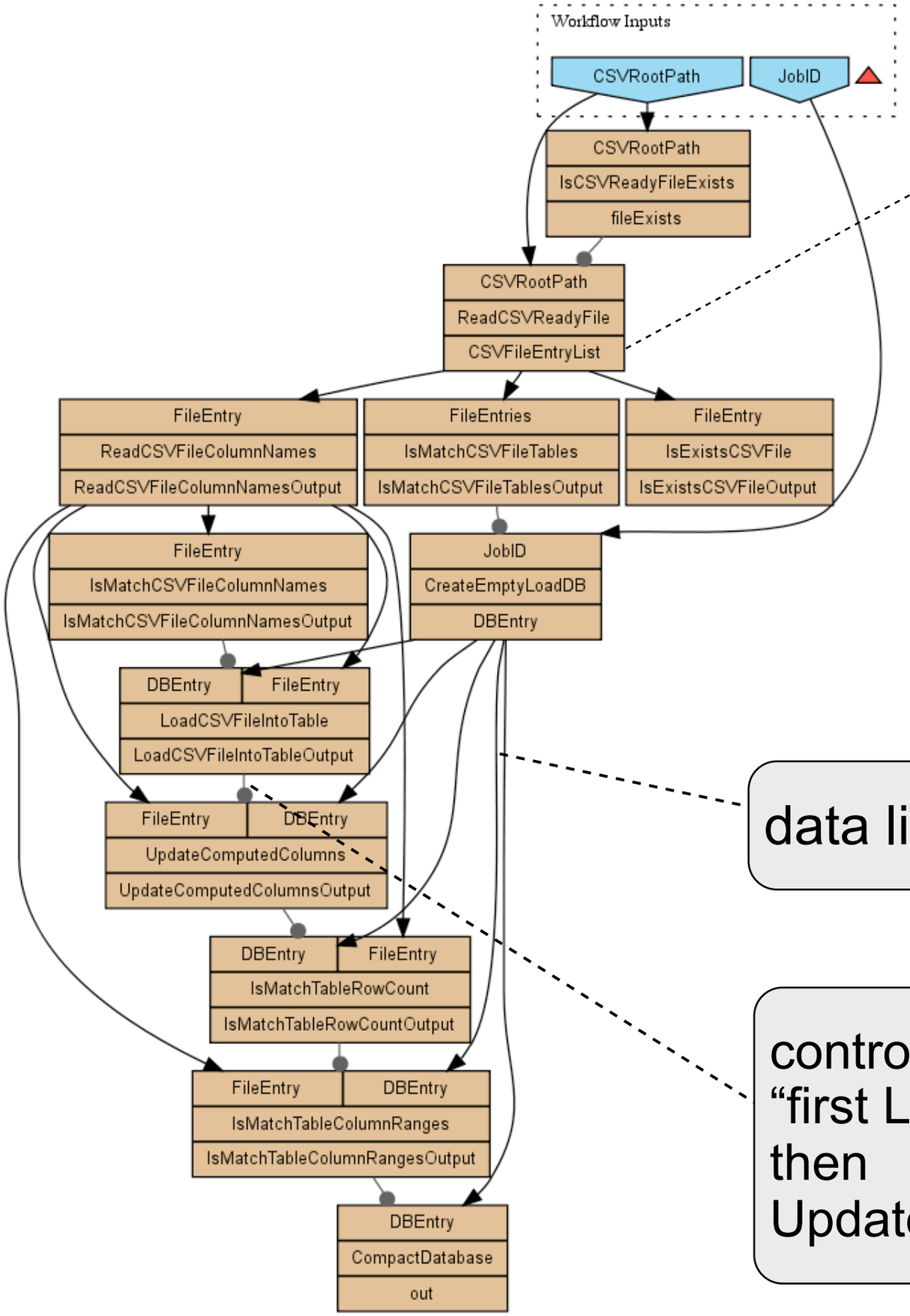
The challenge workflow as a Taverna dataflow



data links

control links:
“first LoadCSVFileIntoTable, then UpdateComputedColumns”

The challenge workflow as a Taverna dataflow

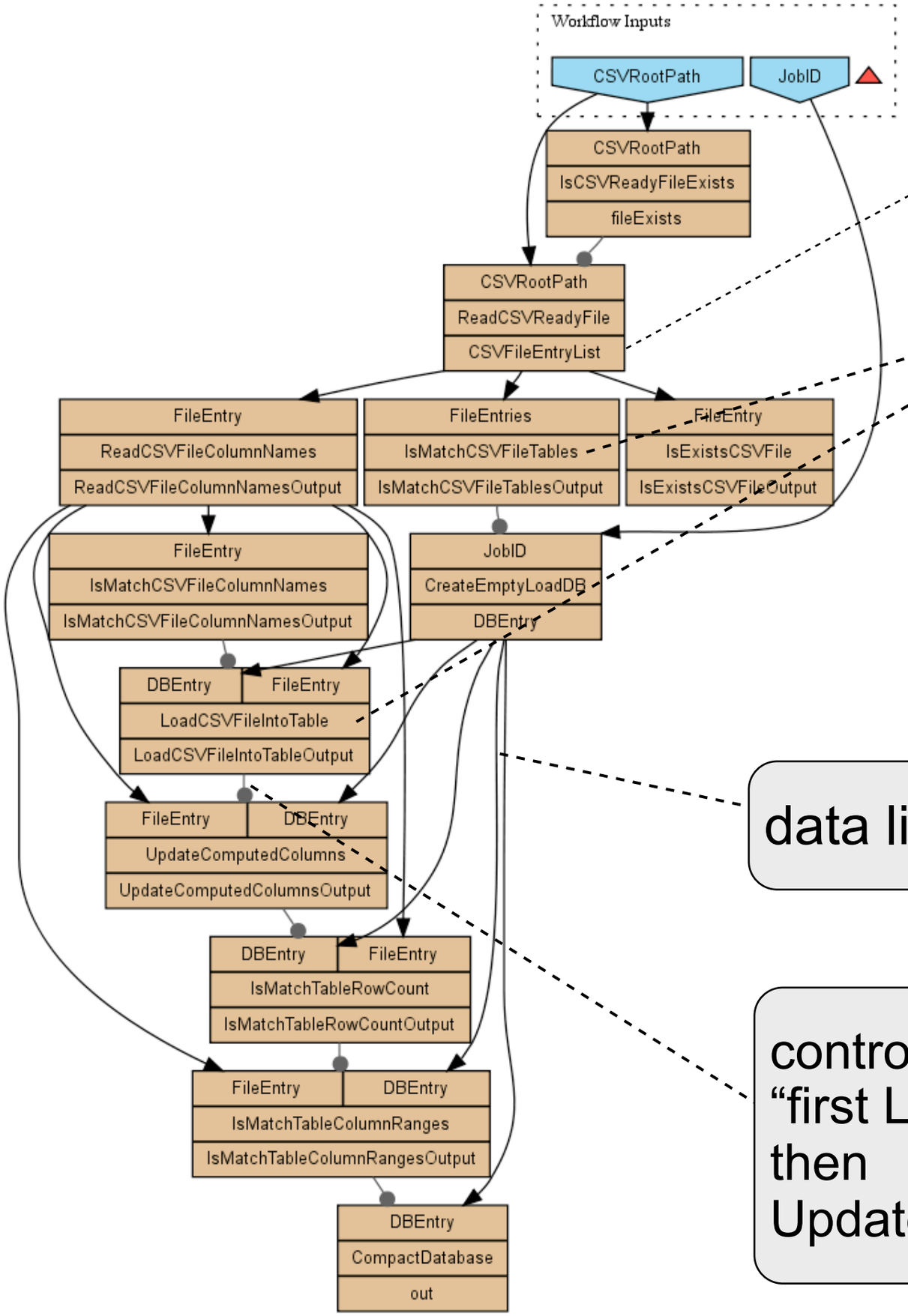


this produces a list...

data links

control links:
“first LoadCSVFileIntoTable, then UpdateComputedColumns”

The challenge workflow as a Taverna dataflow



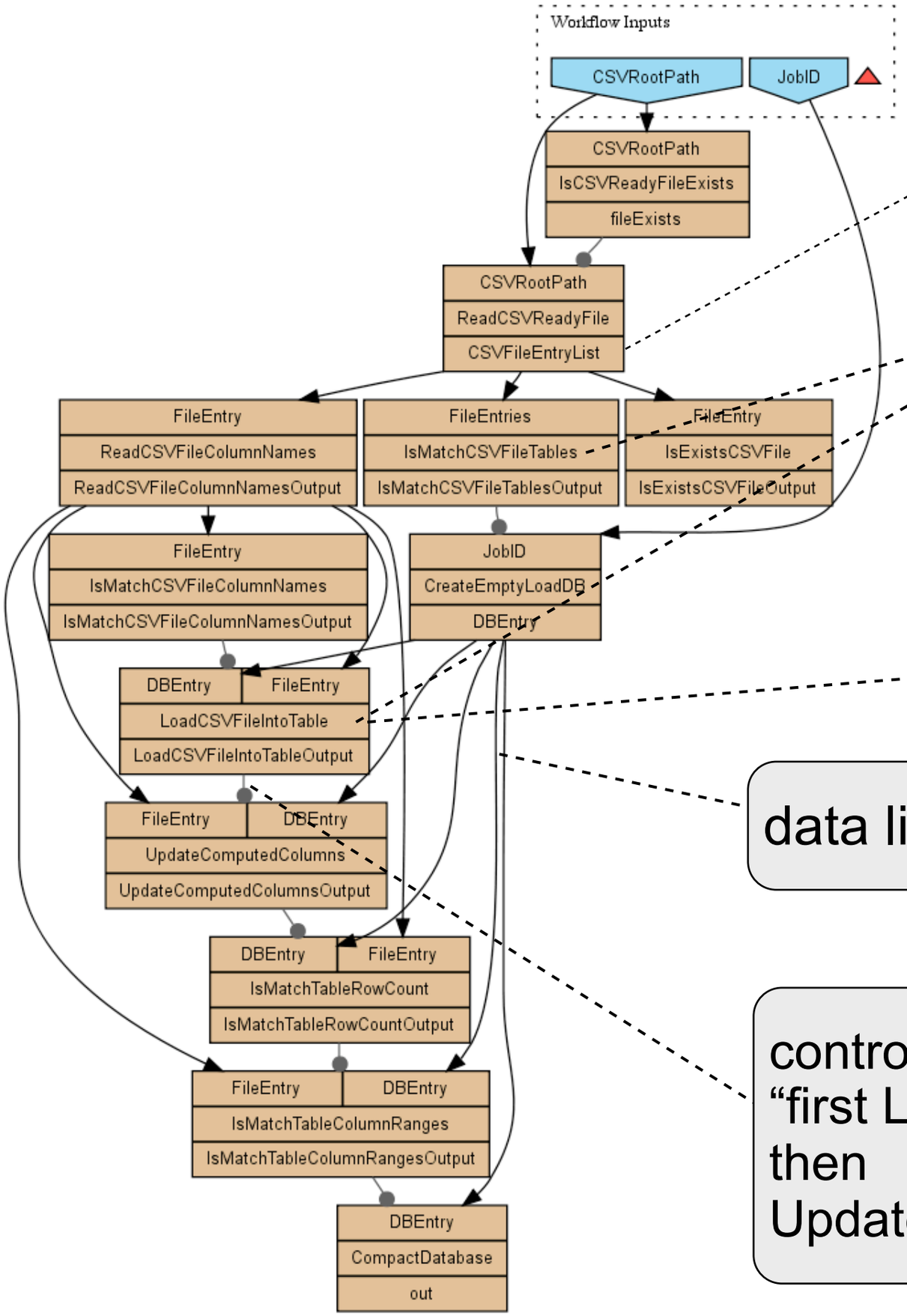
this produces a list...

...these consume one item of the list at a time....

data links

control links:
“first LoadCSVFileIntoTable, then UpdateComputedColumns”

The challenge workflow as a Taverna dataflow



this produces a list...

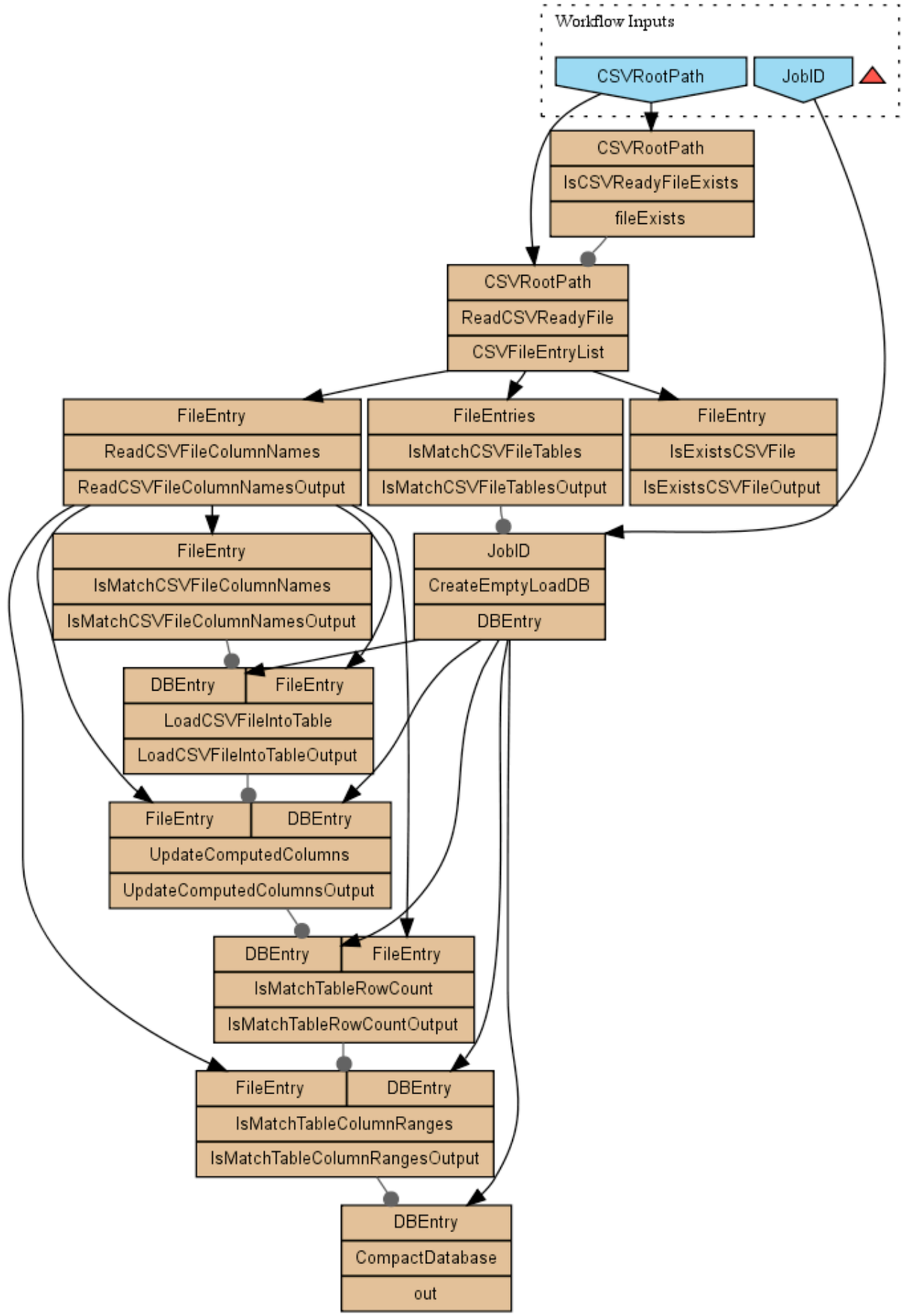
...these consume one item of the list at a time....

the resulting iteration over the lists occurs automatically

data links

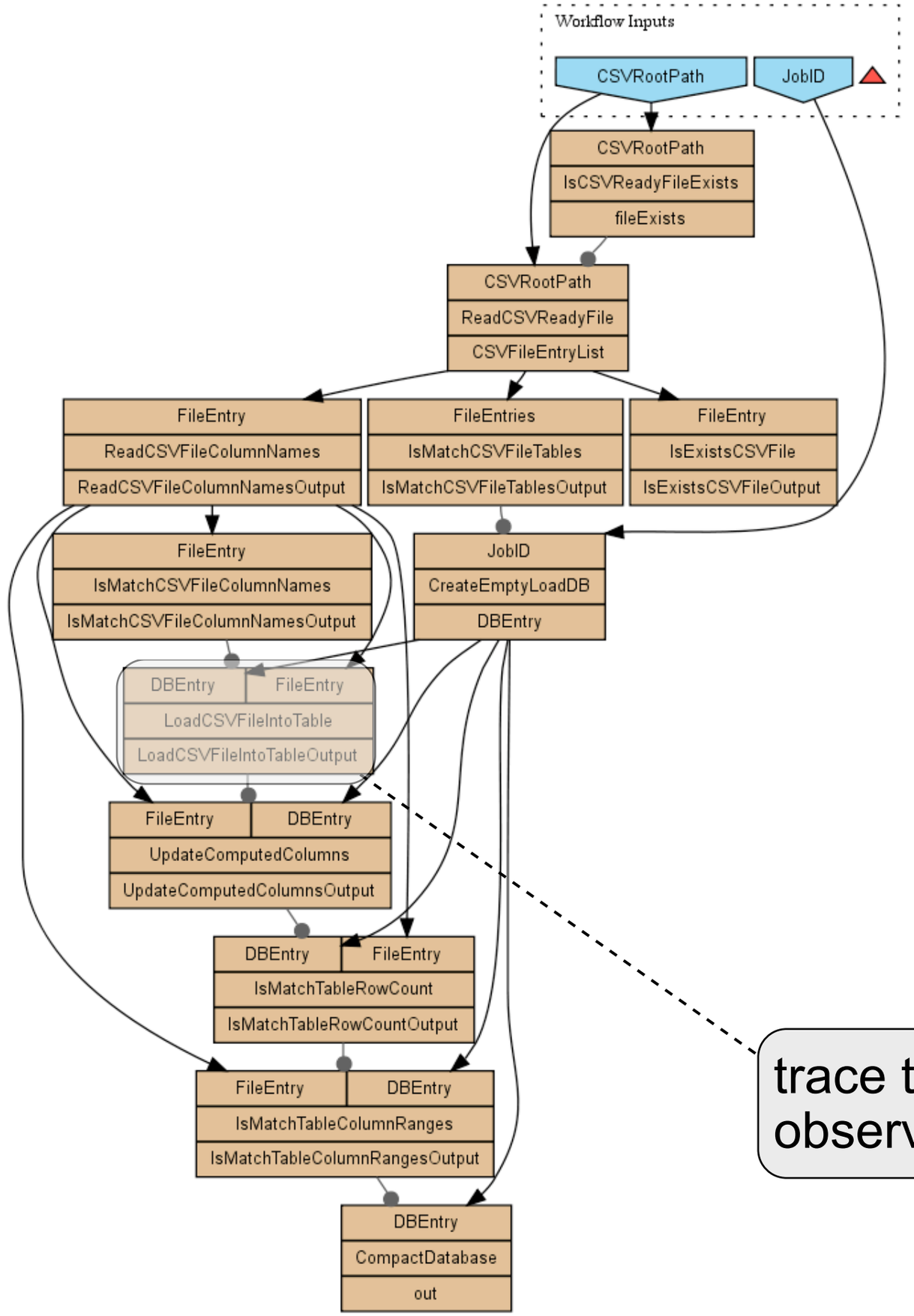
control links:
“first LoadCSVFileIntoTable, then UpdateComputedColumns”

TP query model and new requirements



- queries are purely structural, no semantics
- queries can be answered only at same level of granularity as that of the service incapsulation of the data in the workflow

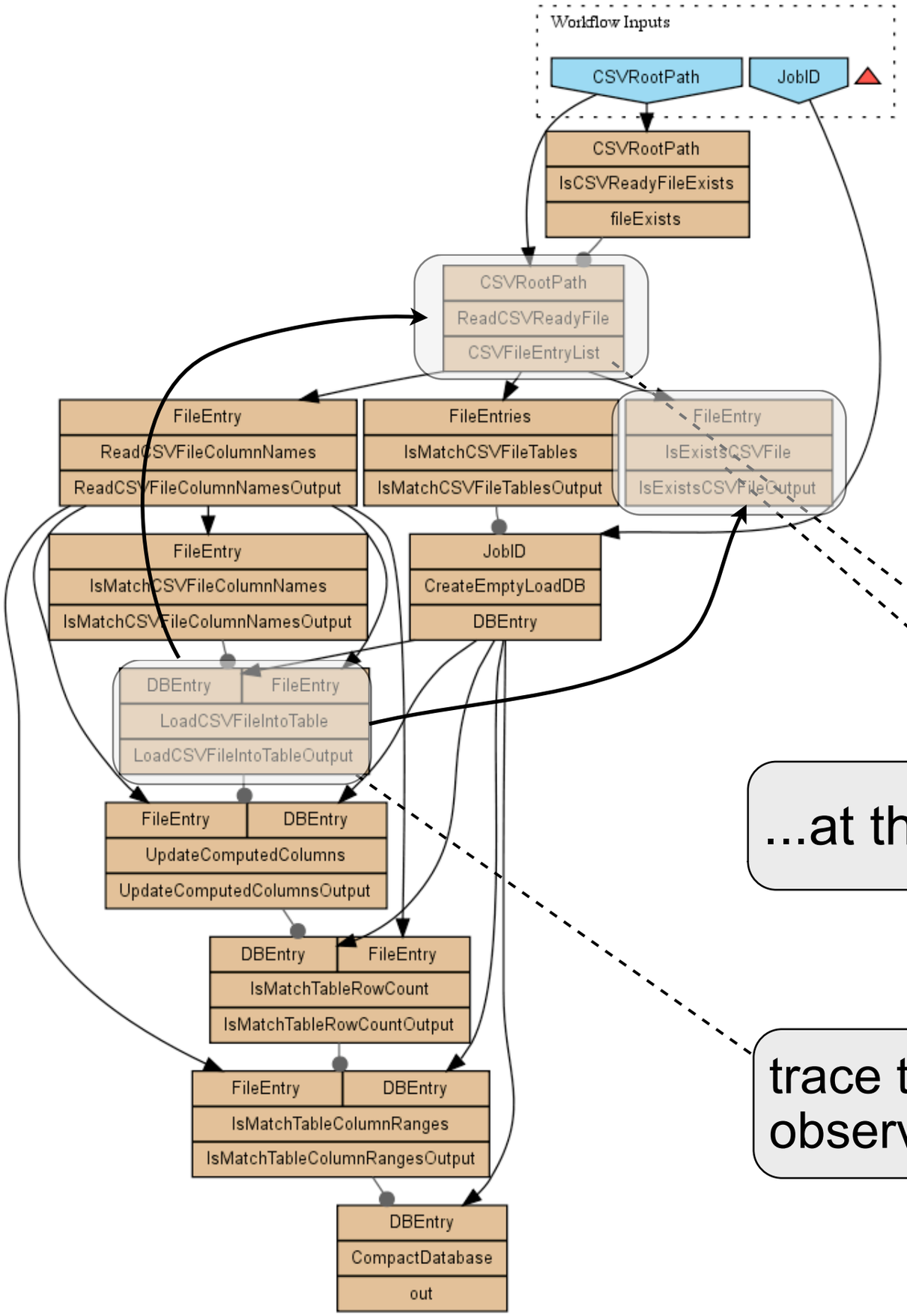
TP query model and new requirements



- queries are purely structural, no semantics
- queries can be answered only at same level of granularity as that of the service incapsulation of the data in the workflow

trace the lineage of values observed here...

TP query model and new requirements

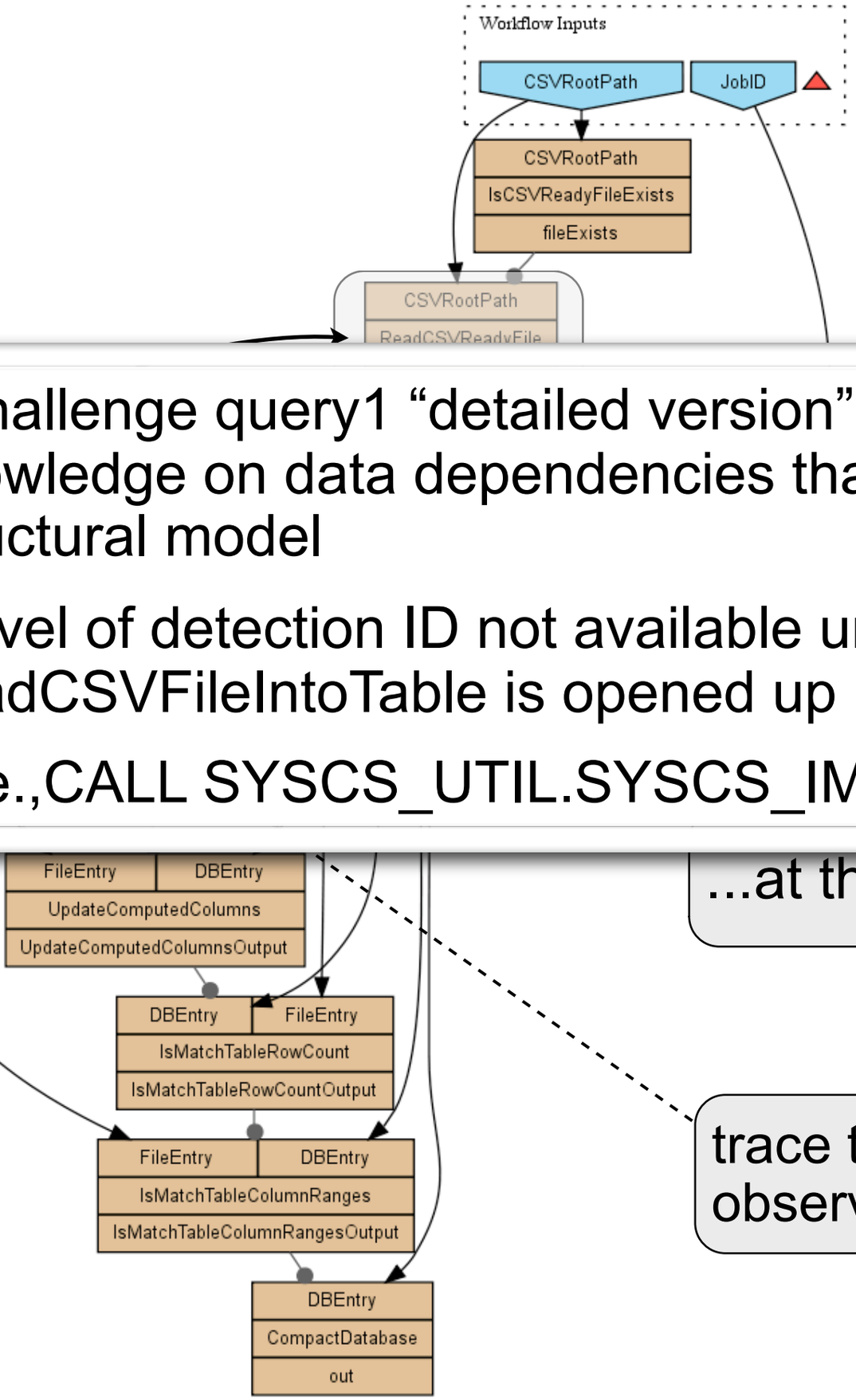


- queries are purely structural, no semantics
- queries can be answered only at same level of granularity as that of the service incapsulation of the data in the workflow

...at these points in the workflow

trace the lineage of values observed here...

TP query model and new requirements



- queries are purely structural, no semantics
- queries can be answered only ...at these points in the workflow

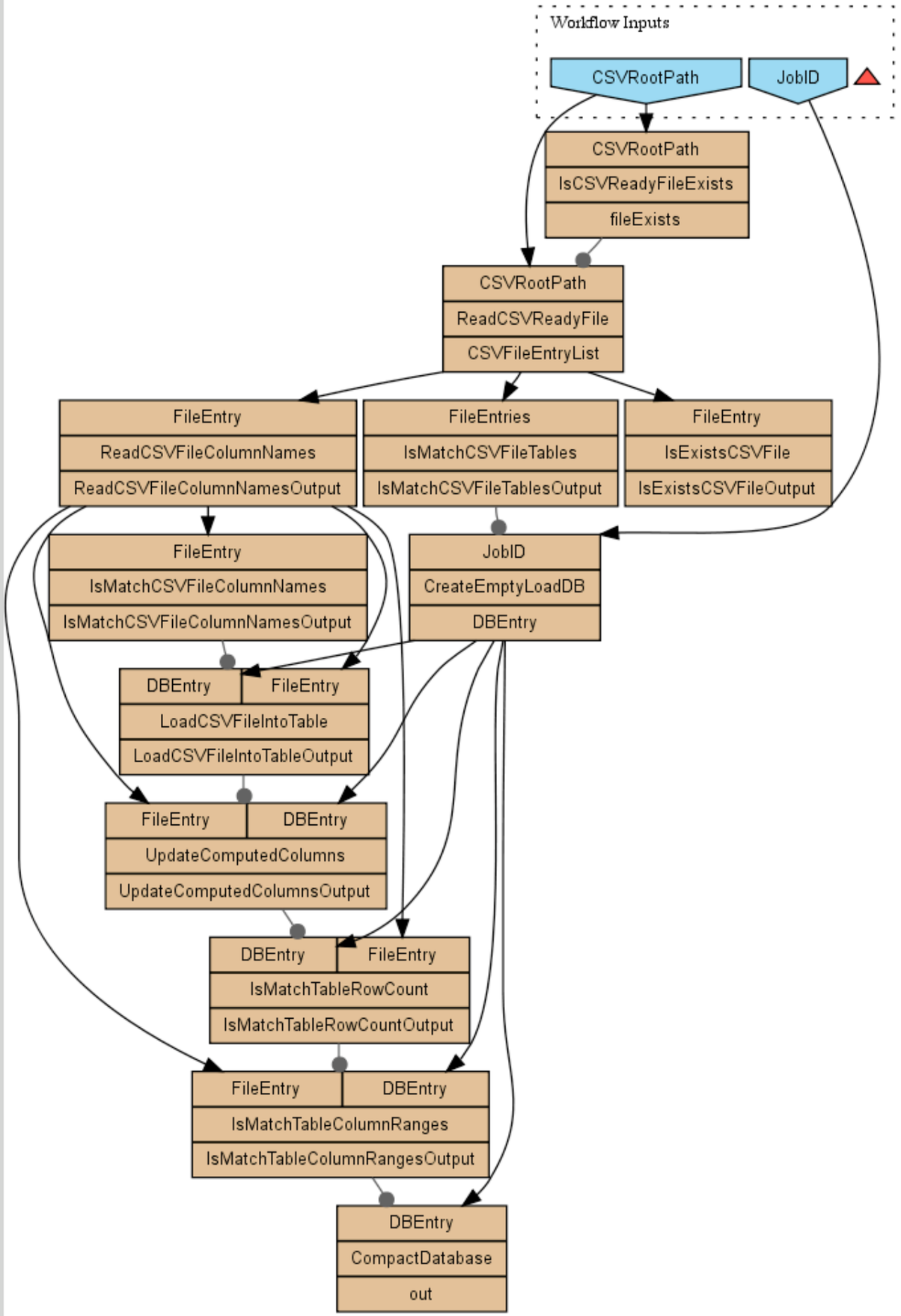
trace the lineage of values observed here...

- challenge query1 “detailed version” seems to require more knowledge on data dependencies than those obtained from this structural model

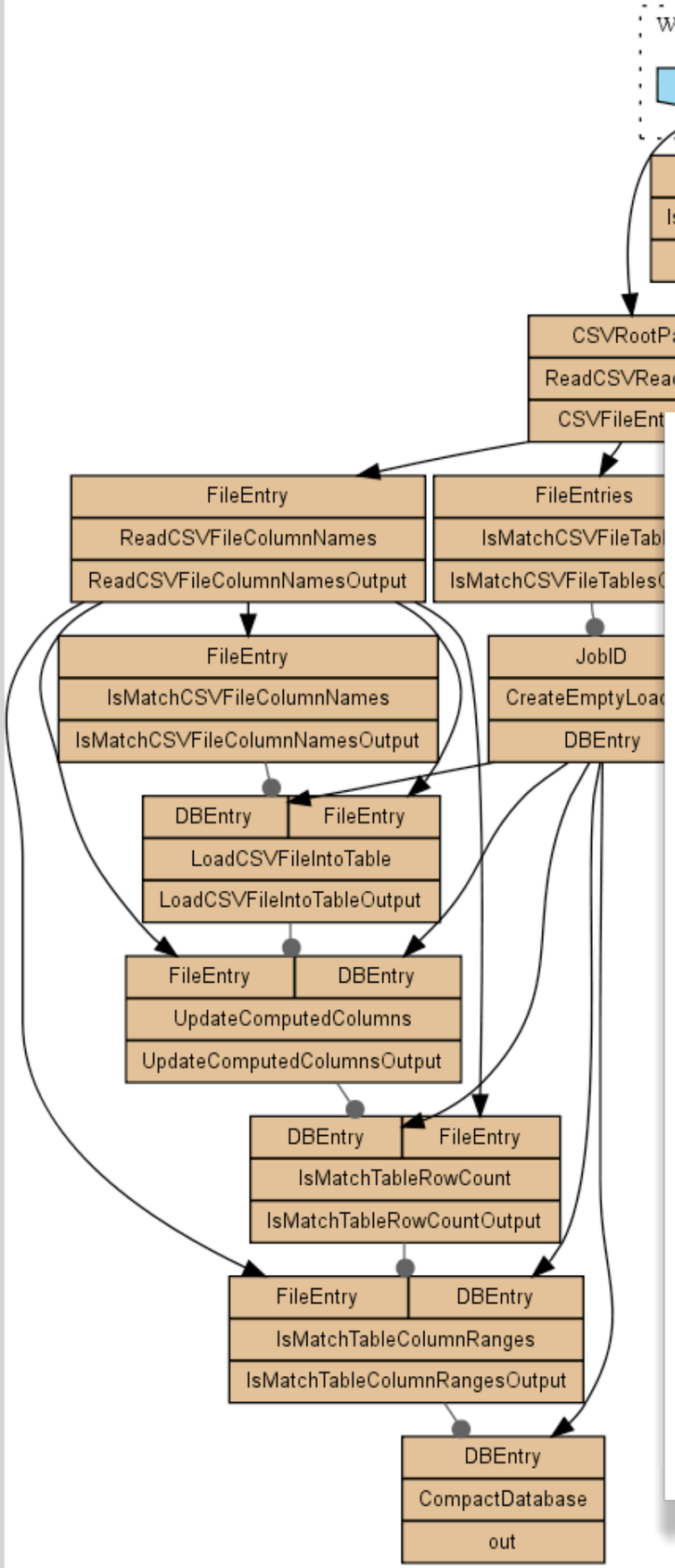
- level of detection ID not available unless query black box in LoadCSVFileIntoTable is opened up

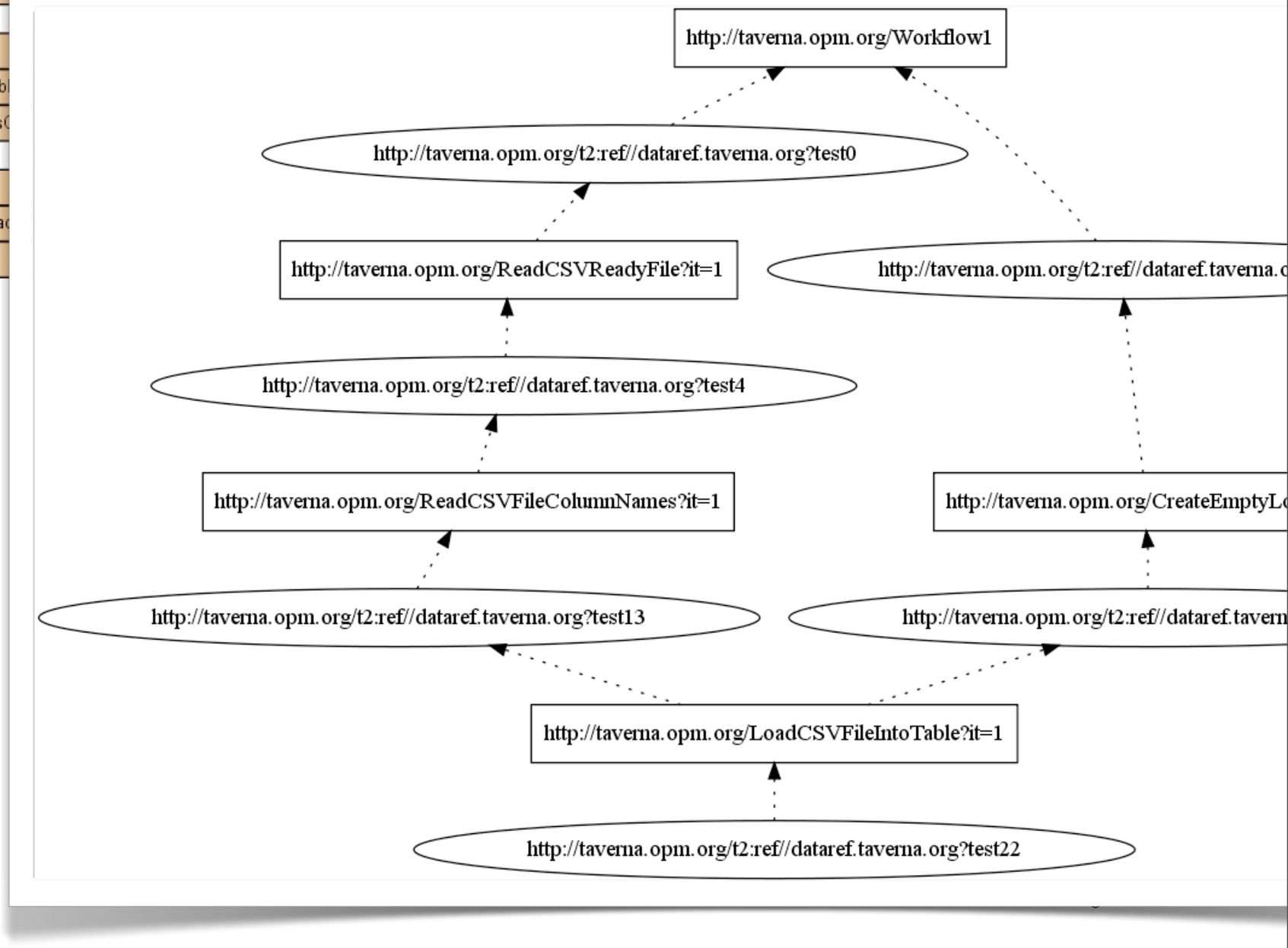
- i.e.,CALL SYSCS_UTIL.SYSCS_IMPORT_TABLE (?,?,?,?,?,?,?)

Integrated OPM generation as part of TP query



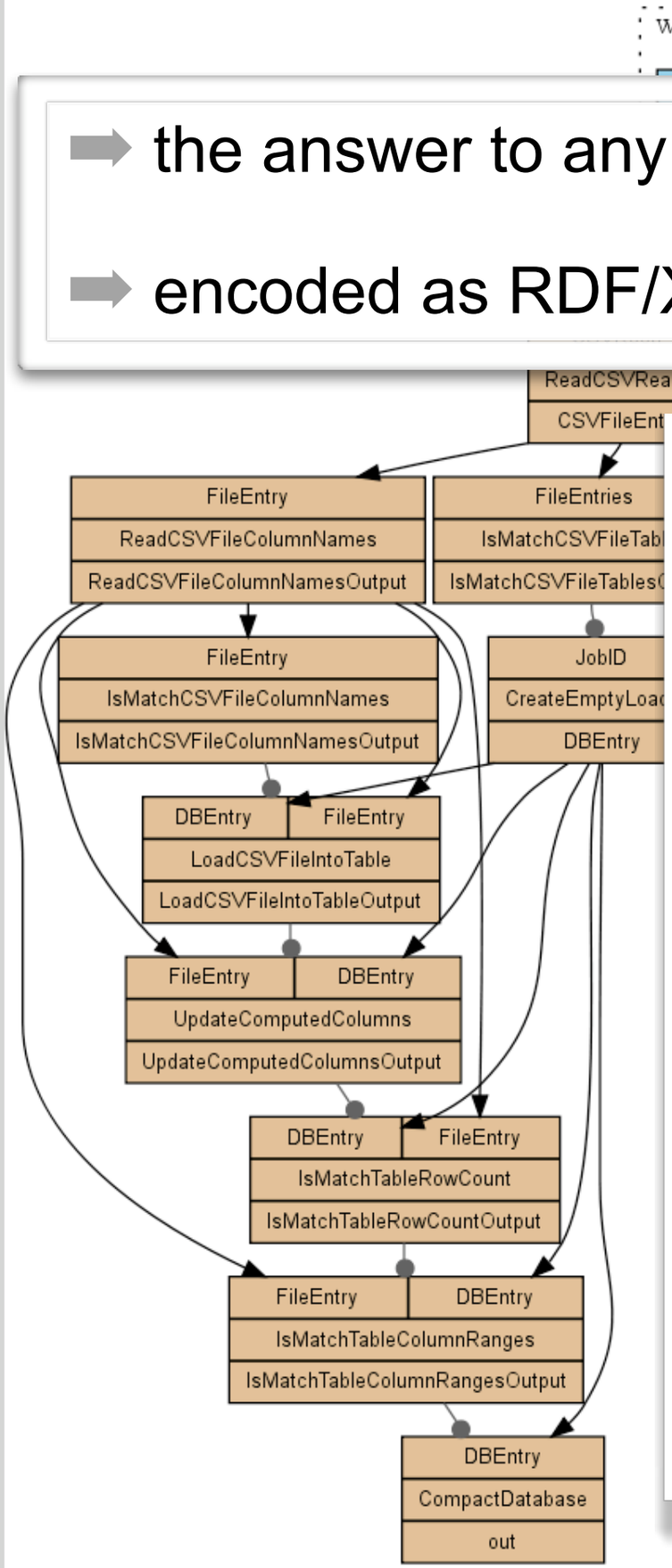
Integrated OPM generation as part of TP query

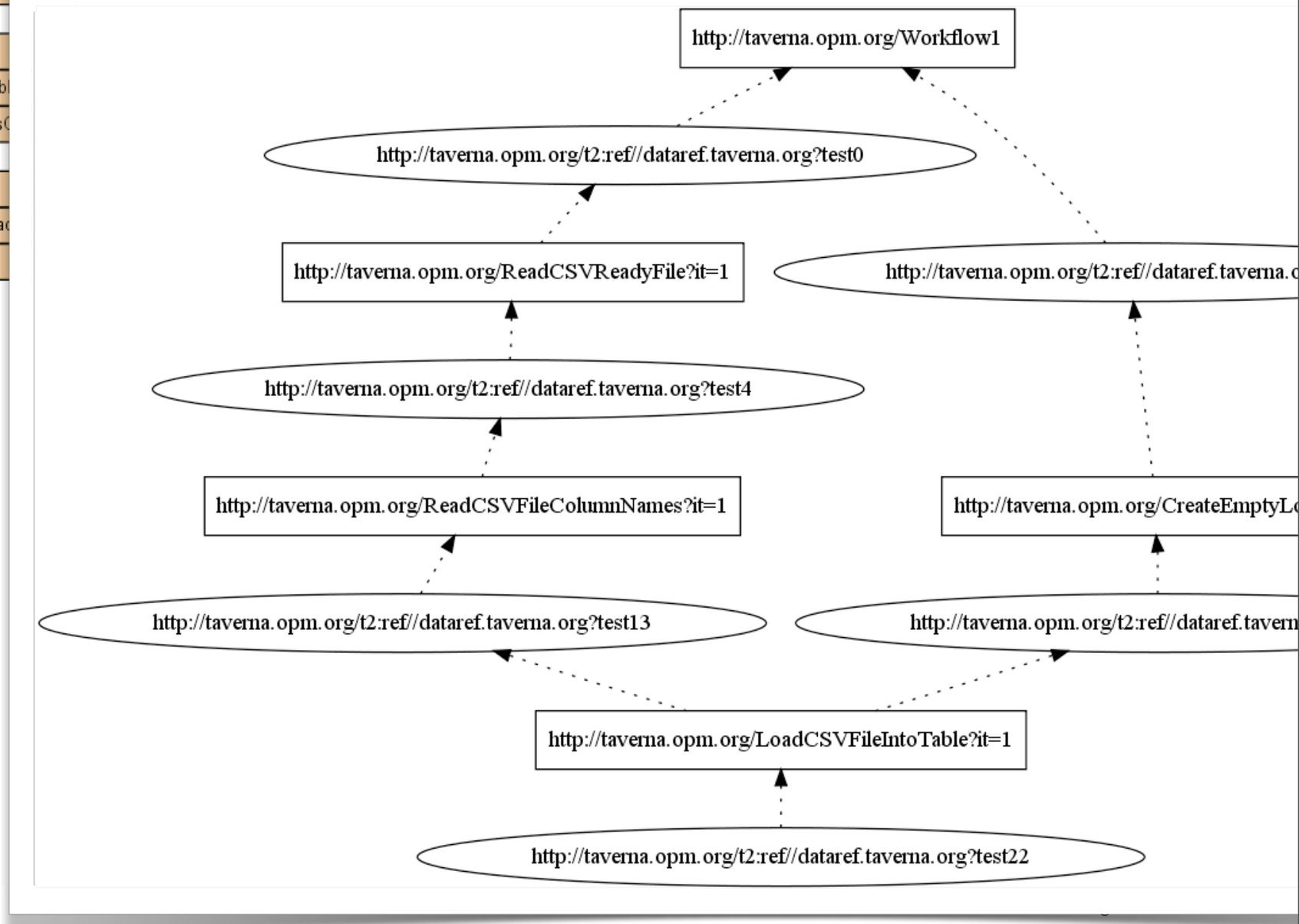




Integrated OPM generation as part of TP query

- the answer to any TP query can be viewed as an OPM graph
- encoded as RDF/XML using the Tupelo provenance API.

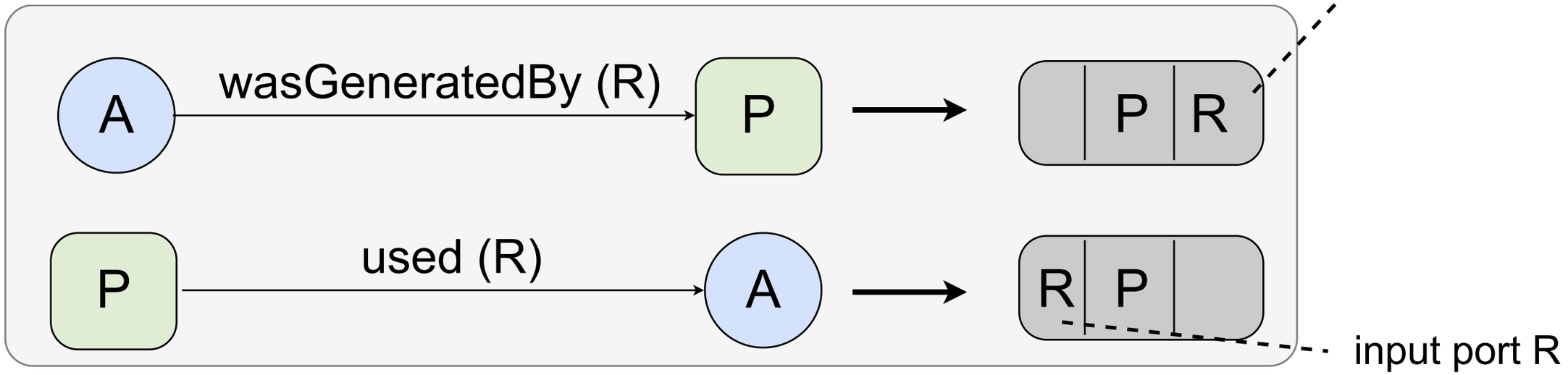




MPOD generation rules - I

MPOD = Minimal Plausible Originating Dataflow

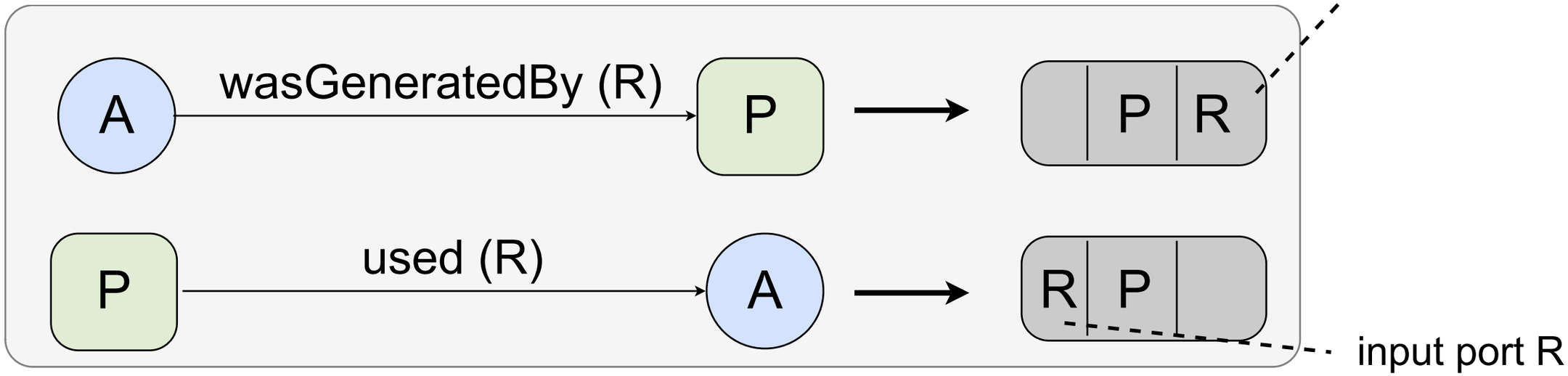
- induced from an OPM graph
- The TP query models requires a workflow structure
- this is a first approximation...subject to refinement

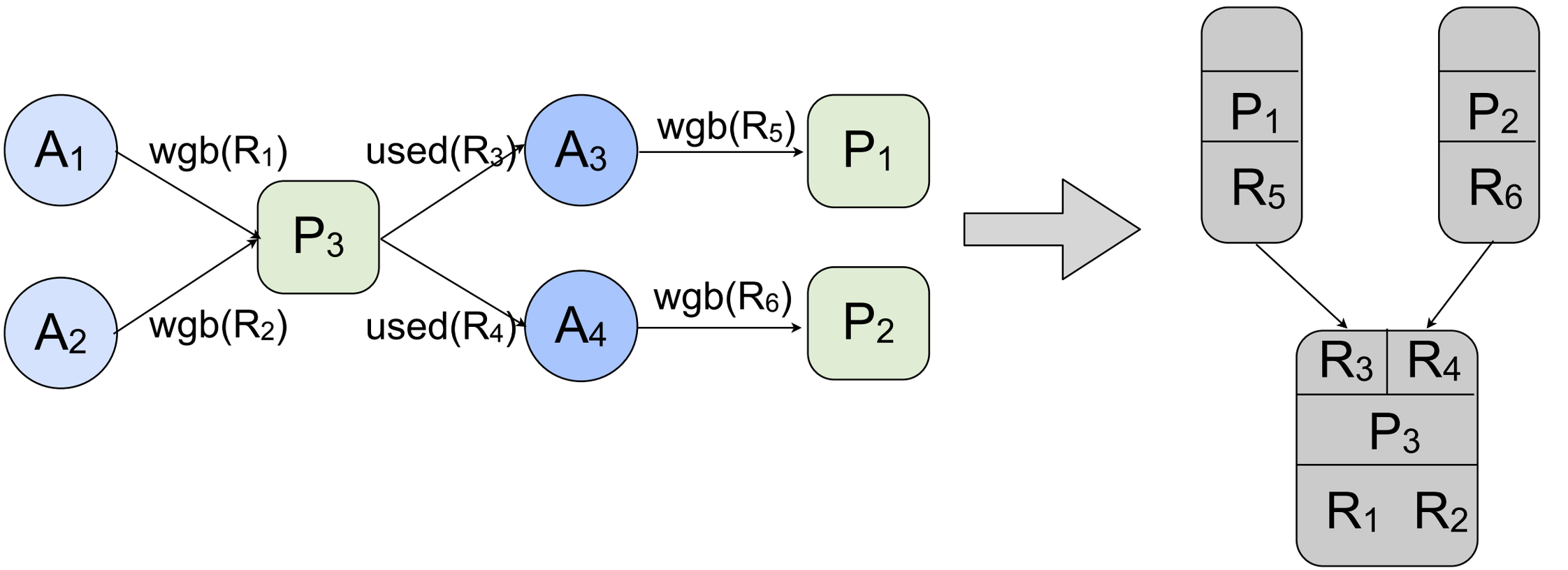


MPOD generation rules - I

MPOD = Minimal Plausible Originating Dataflow

- induced from an OPM graph
- The TP query models requires a workflow structure
- this is a first approximation...subject to refinement

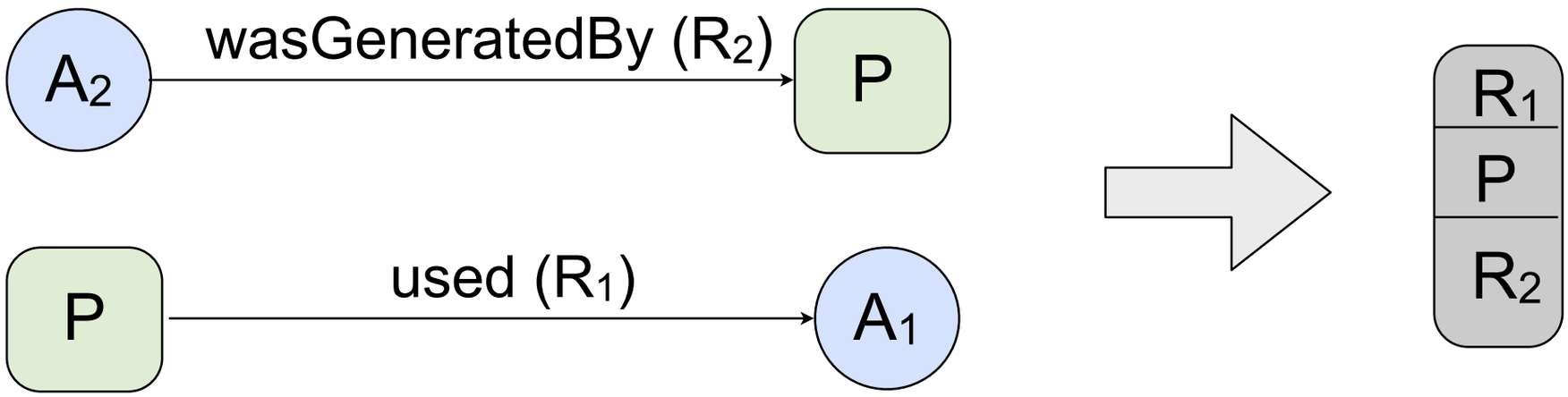




MPOD generation rules - II

Derived property:

This is usually inferred, i.e. there exist P, R_1, R_2 such that:



Note 1: if the corresponding “wgby” and “used” edges are not found, then new P, R_1, R_2 are created and added to the graph

- however, in all cases encountered so far, wasDeterminedFrom was inferred: P, R_1, R_2 appear in existing wgb(R) and used(R) edges

Note 2: wasControlledBy and wasTriggeredBy ignored for now

Note 3: a separate MPOD is created *for each account* in the OPM graph

Importing OPM for the challenge

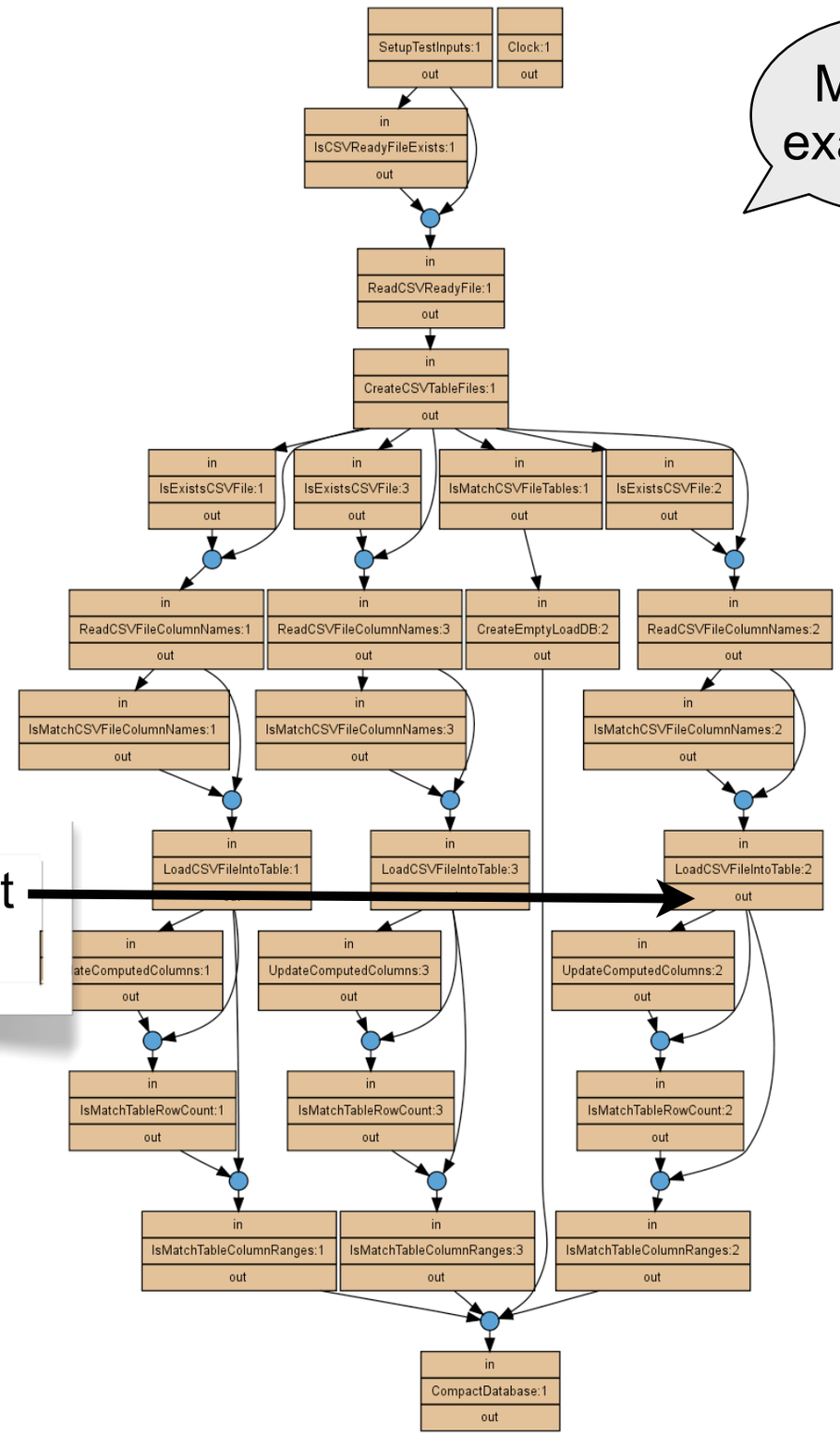
- OPM contributions successfully imported so far:
  - UC Davis
  - NCSA
  - SOTON

(links to PC3 / UoM wiki page)

- Example (UC Davis)

Which operation executions were strictly necessary for the Image table to contain a particular (non-computed) value?

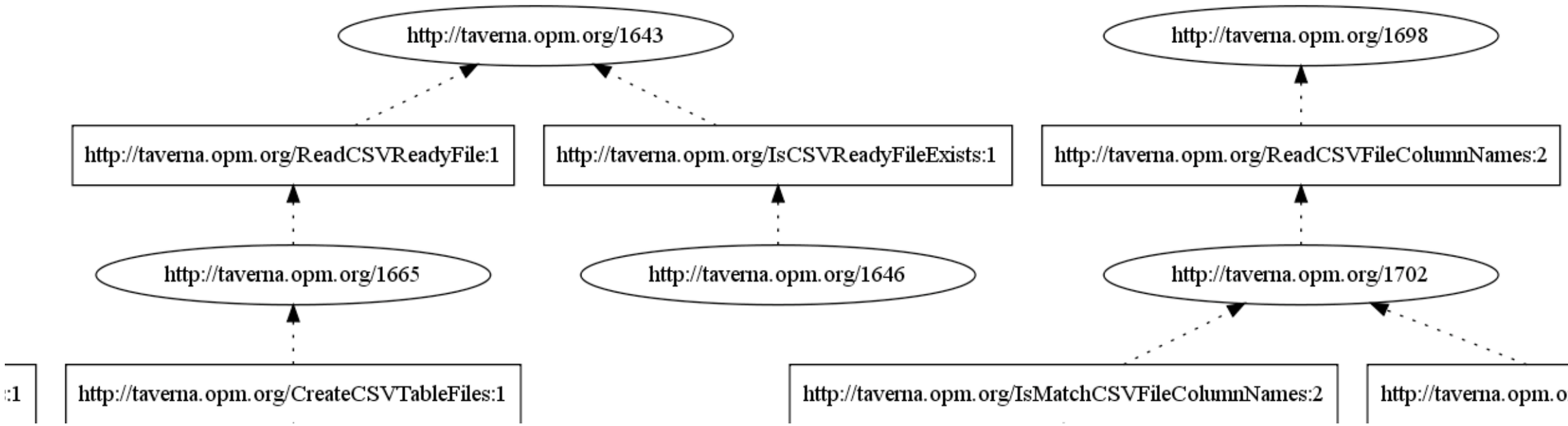
query.variables: LoadCSVFileIntoTable:2 / out
query.processors=ALL



MPOD example!

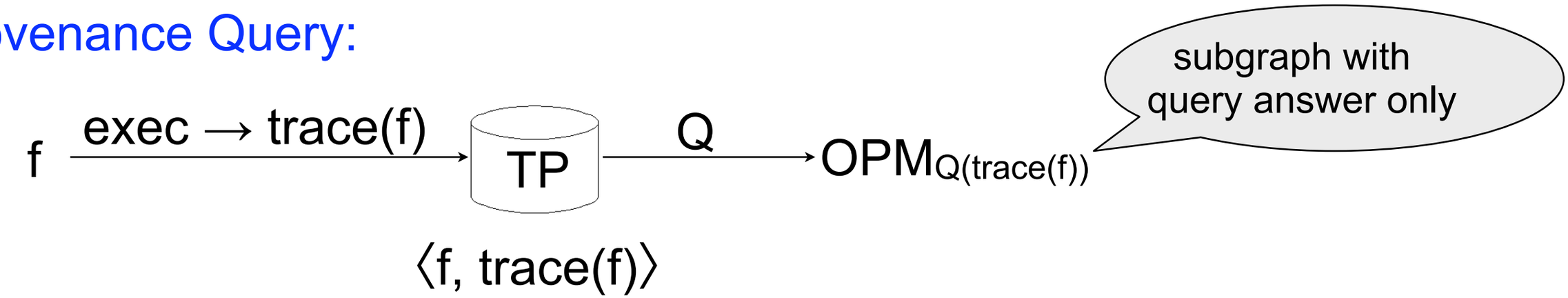
Example -- query 3 result

- Ideally, the imported graph + MPOD allow provenance queries to be submitted to the imported graph just as if it were a native TP graph
- The answer is viewed as a new OPM graph itself



Lossless mappings using OPM and MPOD

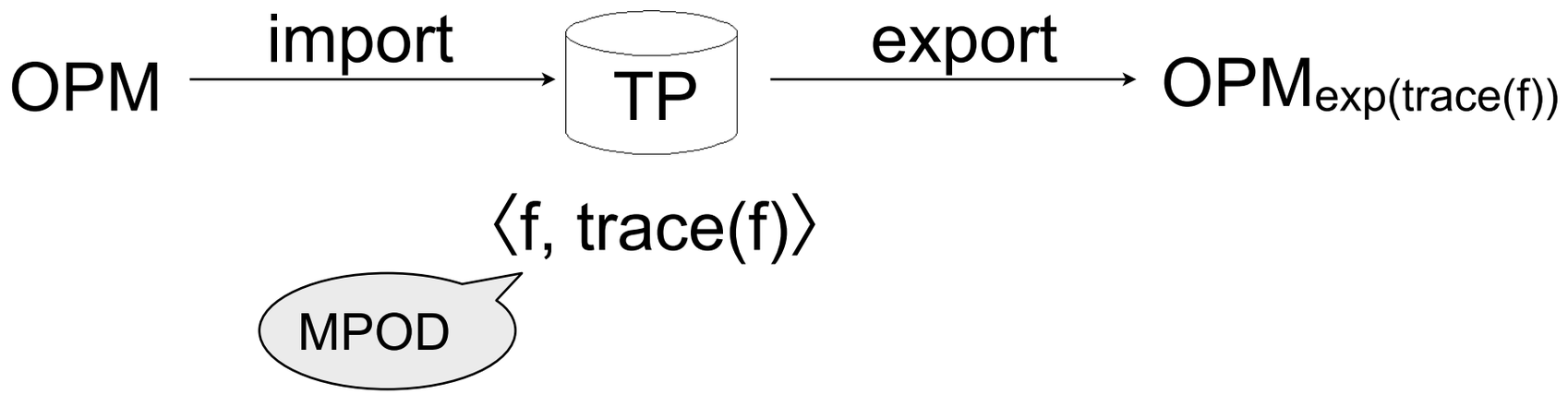
Provenance Query:



Export to OPM:

export is just a query exp(trace(f)) that returns the entire trace

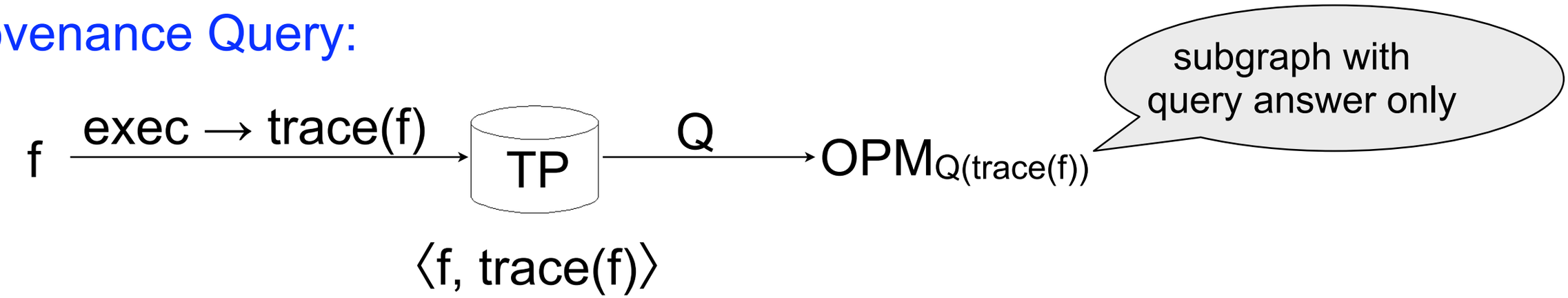
Import from OPM:



TP = Taverna Provenance model

Lossless mappings using OPM and MPOD

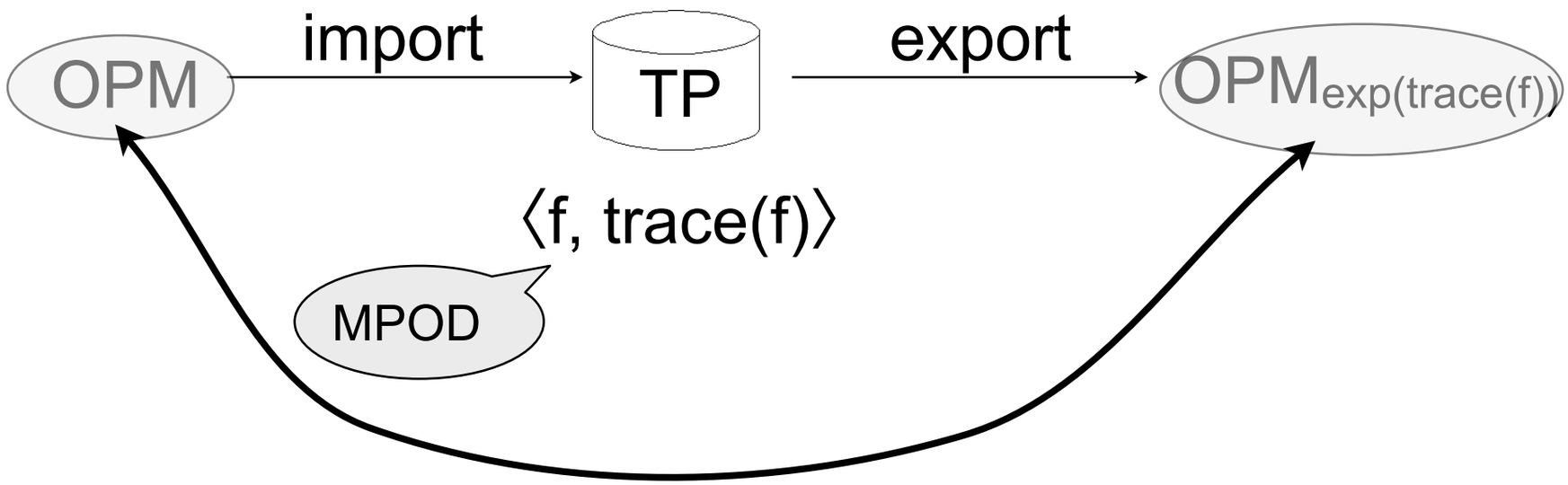
Provenance Query:



Export to OPM:

export is just a query exp(trace(f)) that returns the entire trace

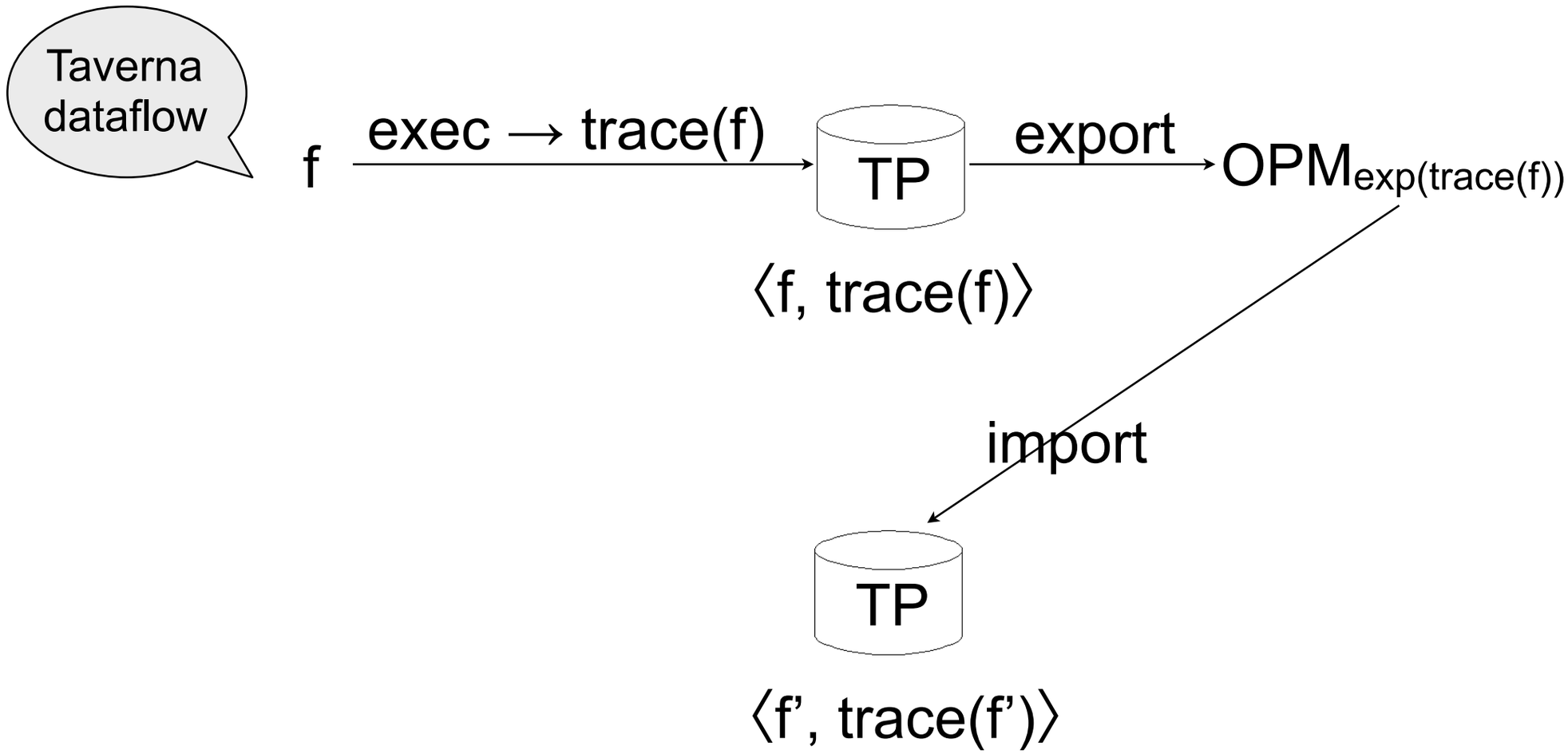
Import from OPM:



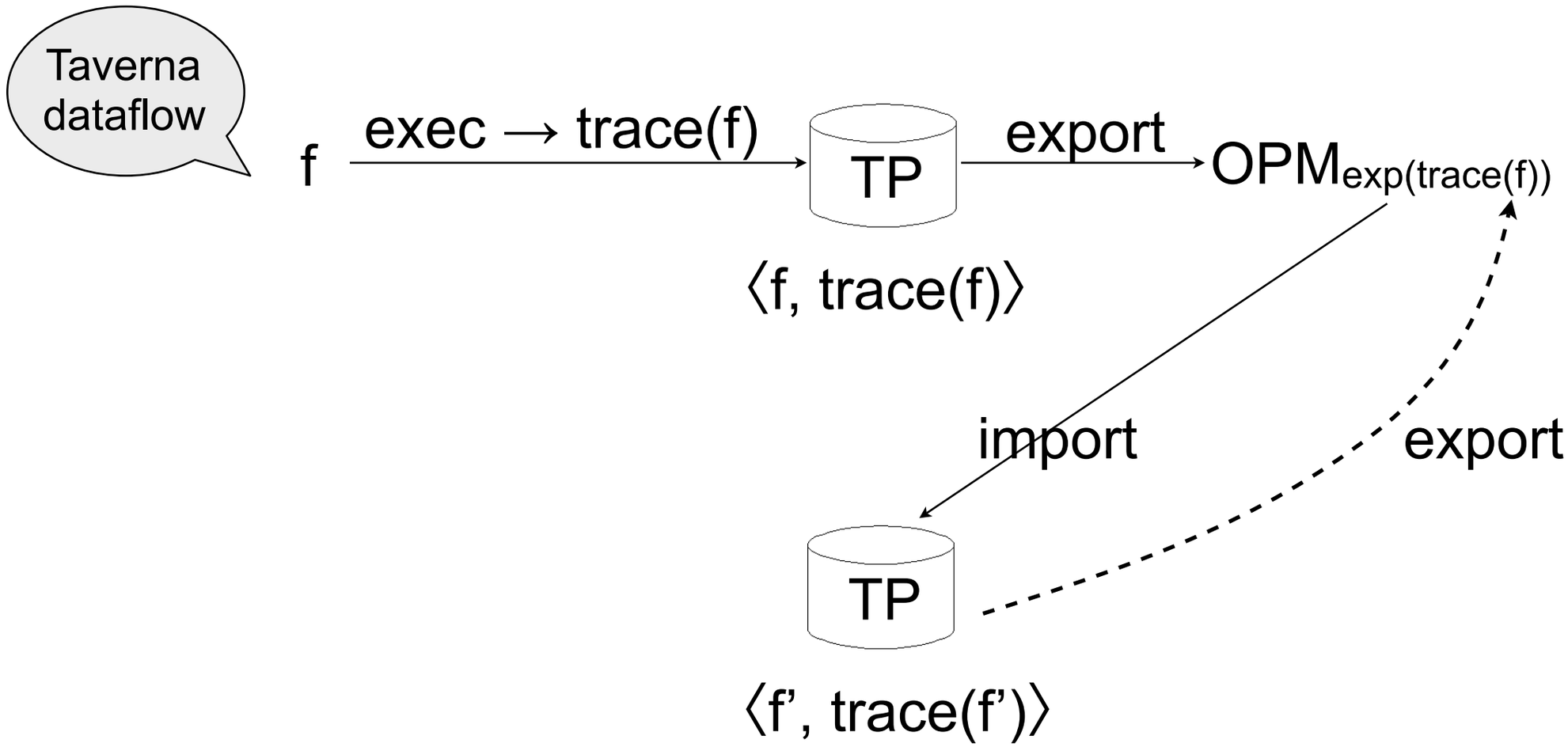
when is this transformation loss-less?

TP = Taverna Provenance model

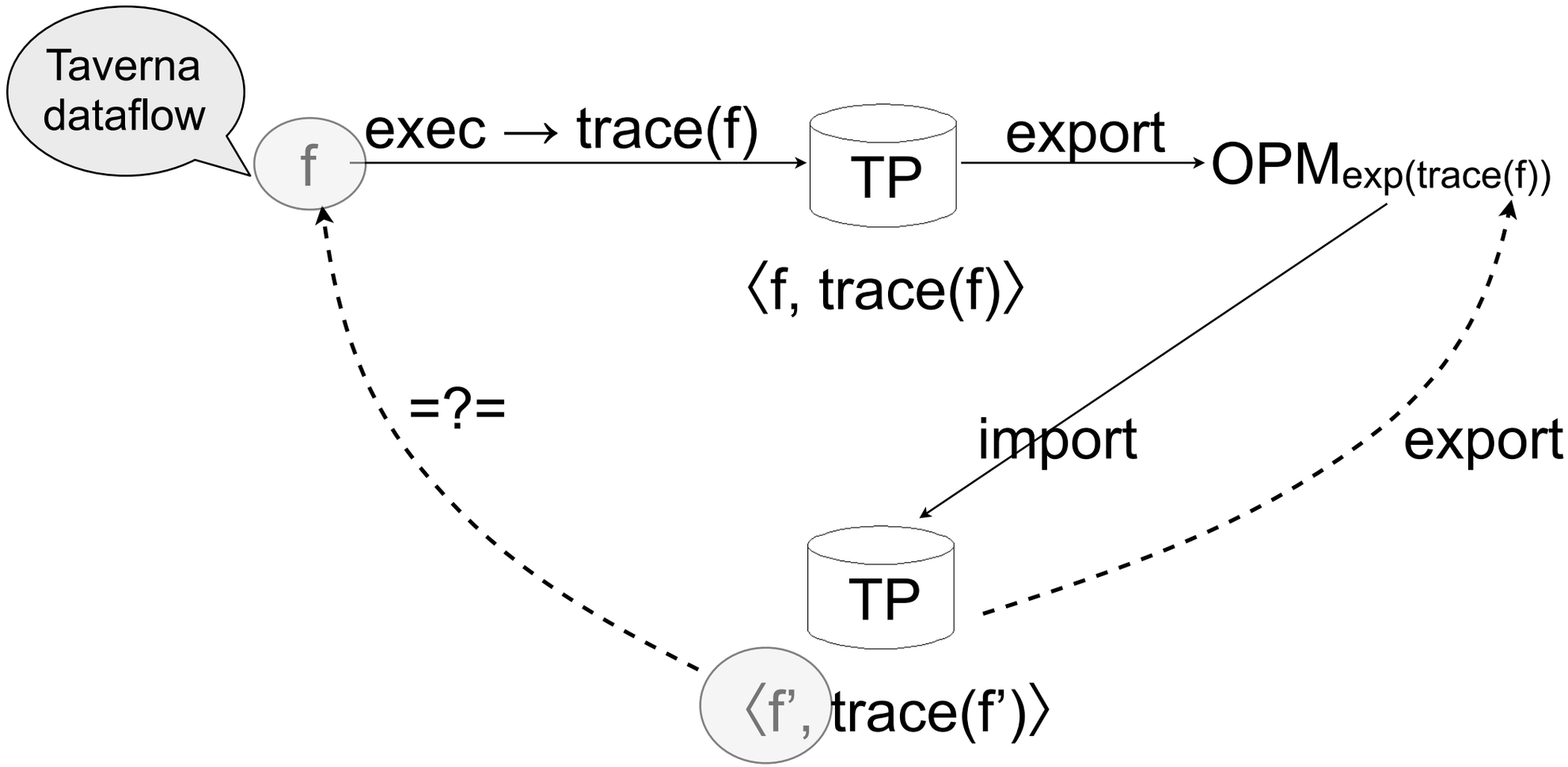
More on lossless-ness and OPM



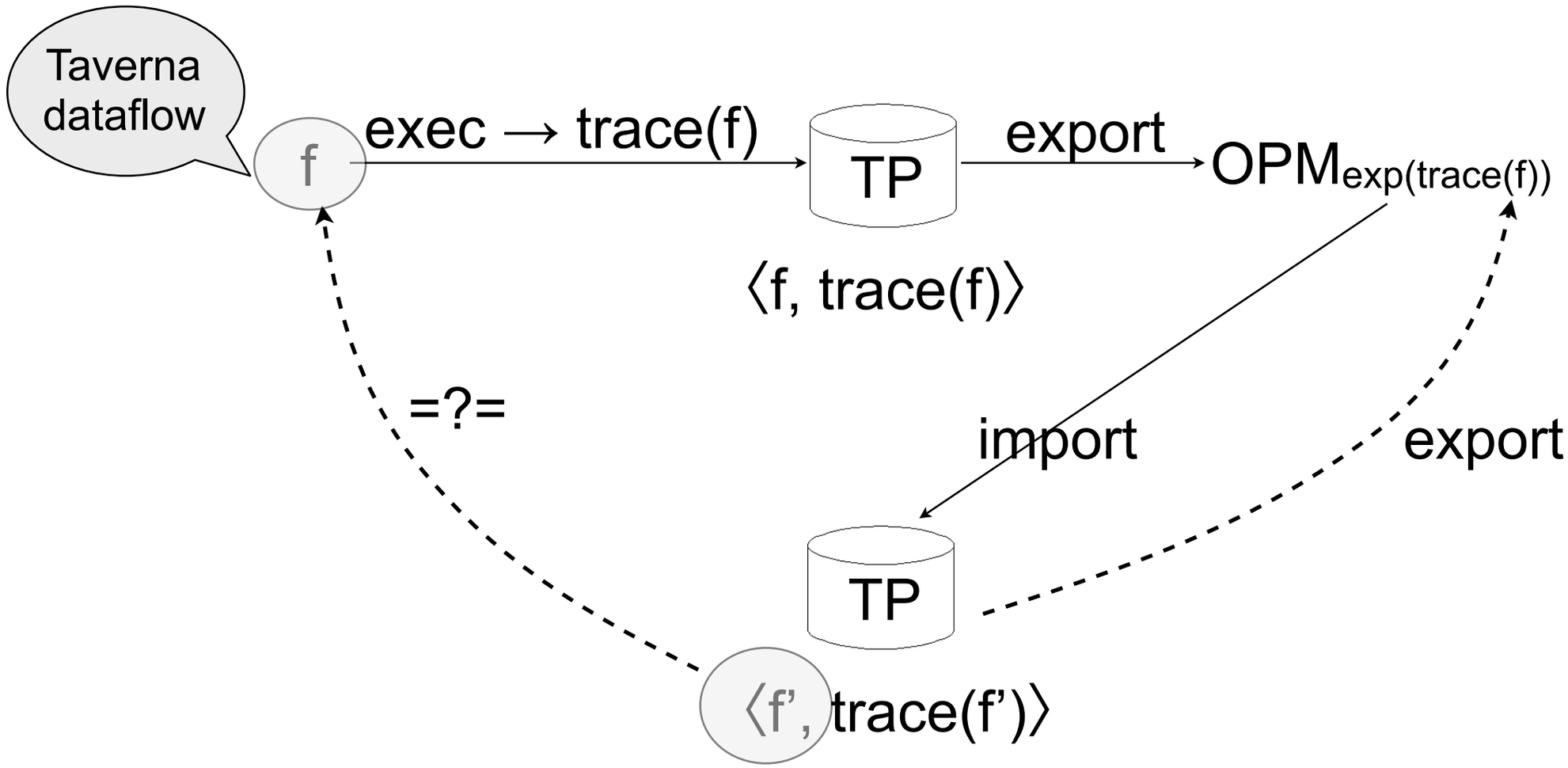
More on lossless-ness and OPM



More on lossless-ness and OPM



More on lossless-ness and OPM



This is indeed lossless when f is itself a Taverna dataflow:

export (import (export (trace(f)))) =?= export (trace(f))

(requires proof)