Christoph Böhme

Analysis of library metadata with Metafacture

DEUTSCHE NATIONAL BIBLIOTHEK

Agenda

13:00 — a short introduction to Metafacture

13:30 — warm-up exercises

14:30 — triples and counting

15:00 — exercises on counting data
(incl. 30 min coffee break at 15:30)

17:00 — joining data sets and analysing them

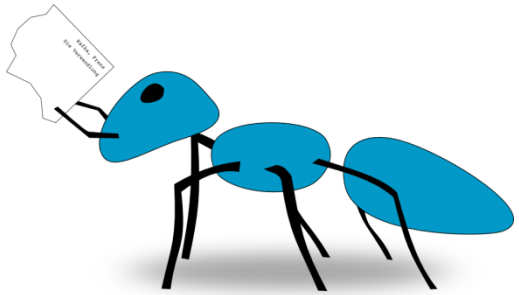
17:30 — exercises on joining data

18:50 — wrapping up

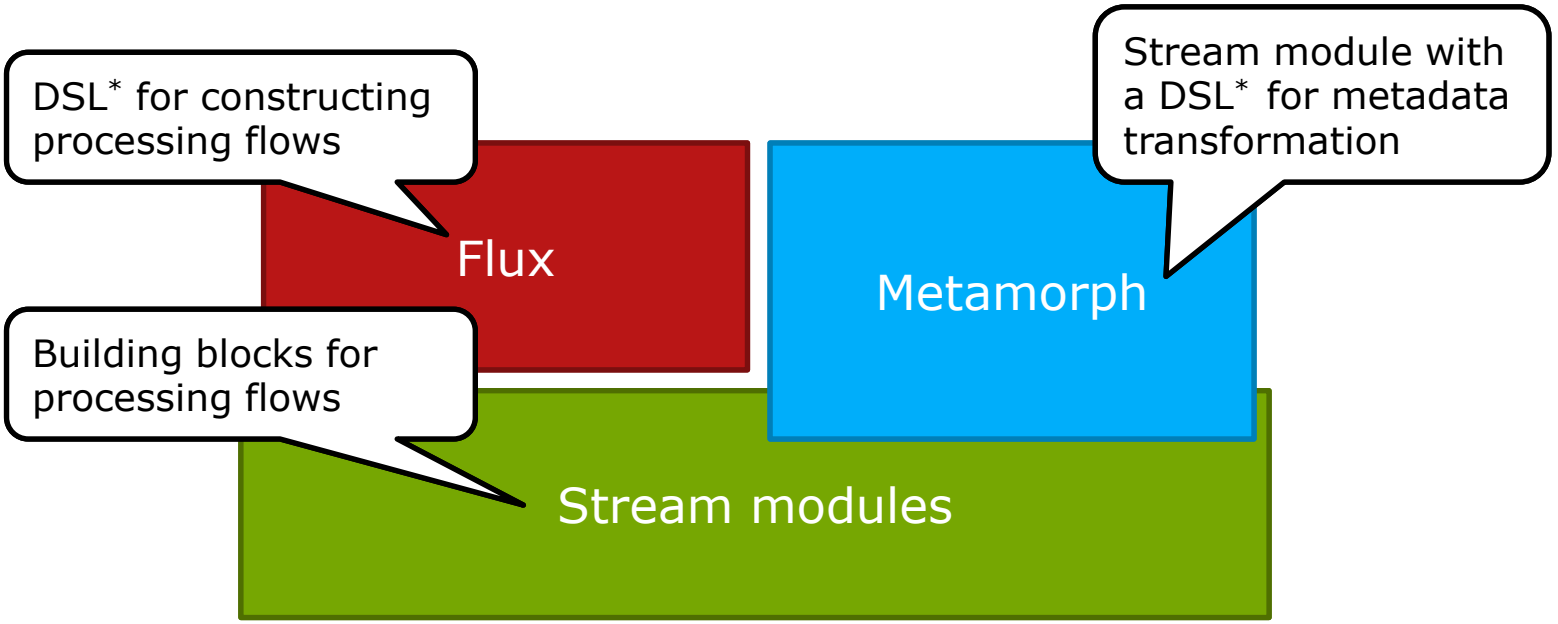
DEUTSCHE NATIONAL BIBLIOTHEK

Part 1

A short introduction to Metafacture

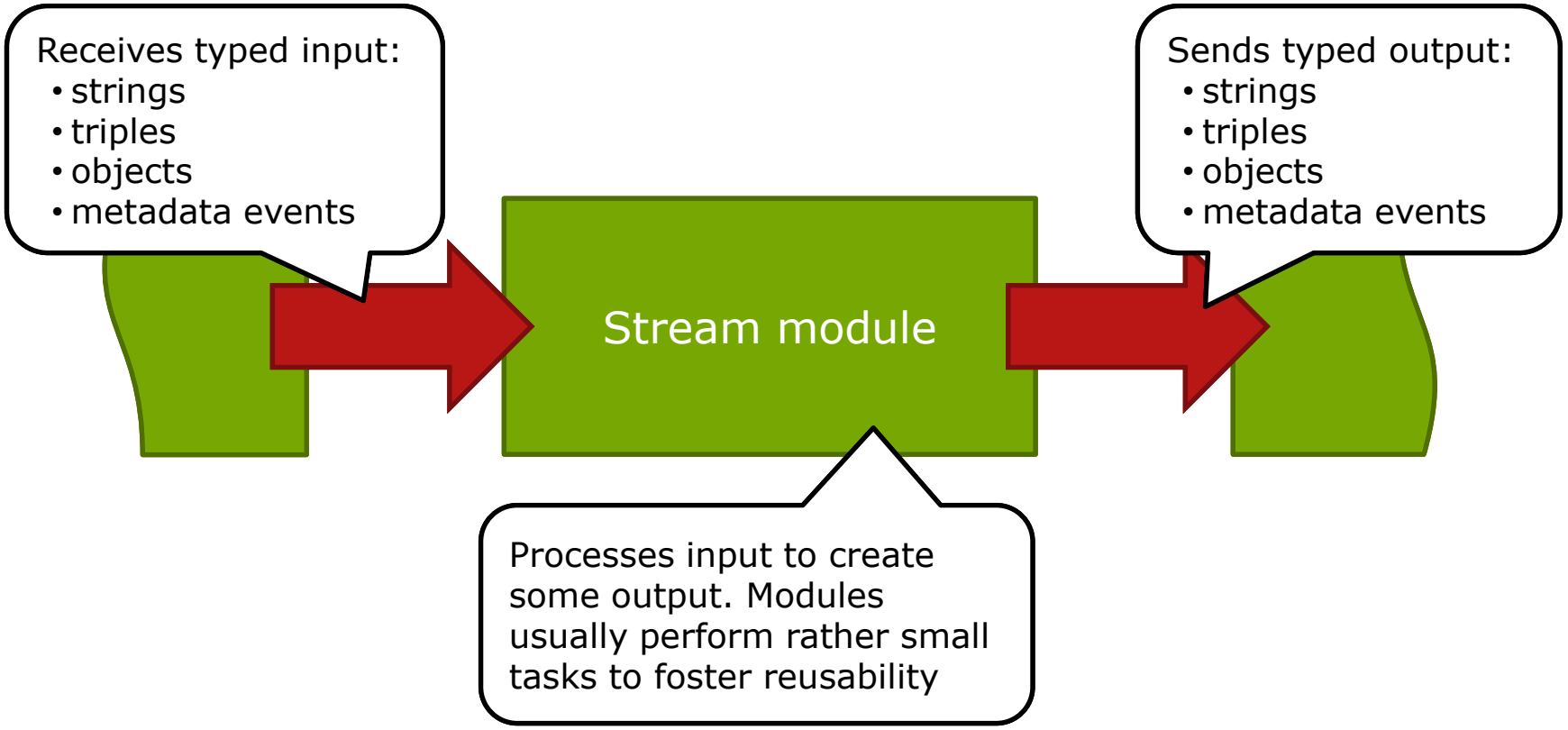


Overview of Metafacture



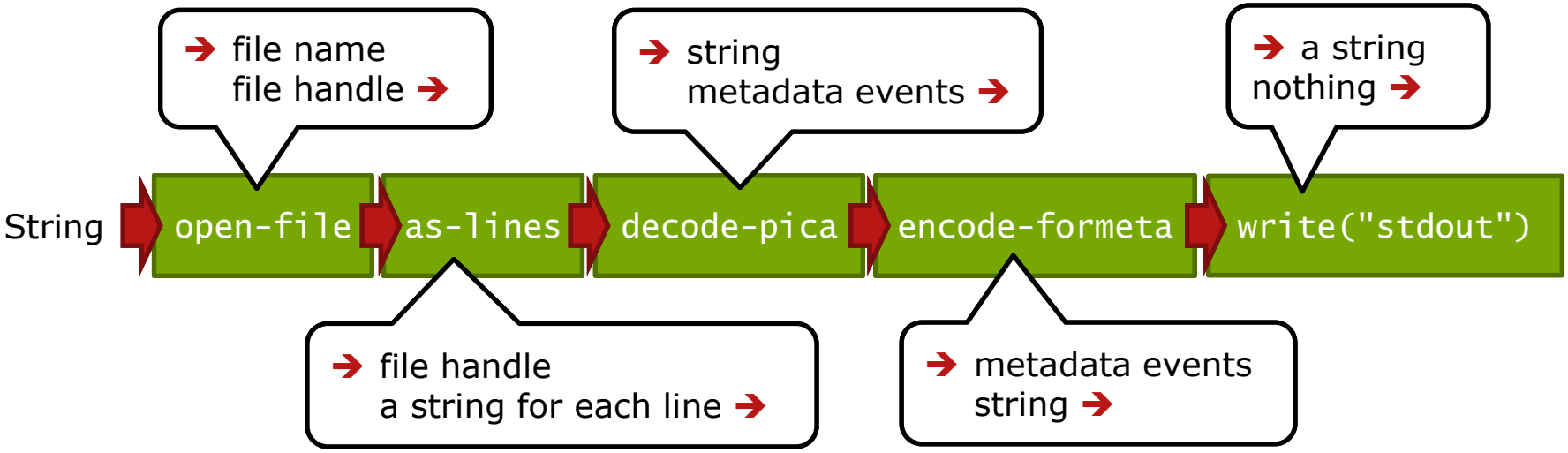
*DSL: Domain specific Language

The basic building block of Metafacture

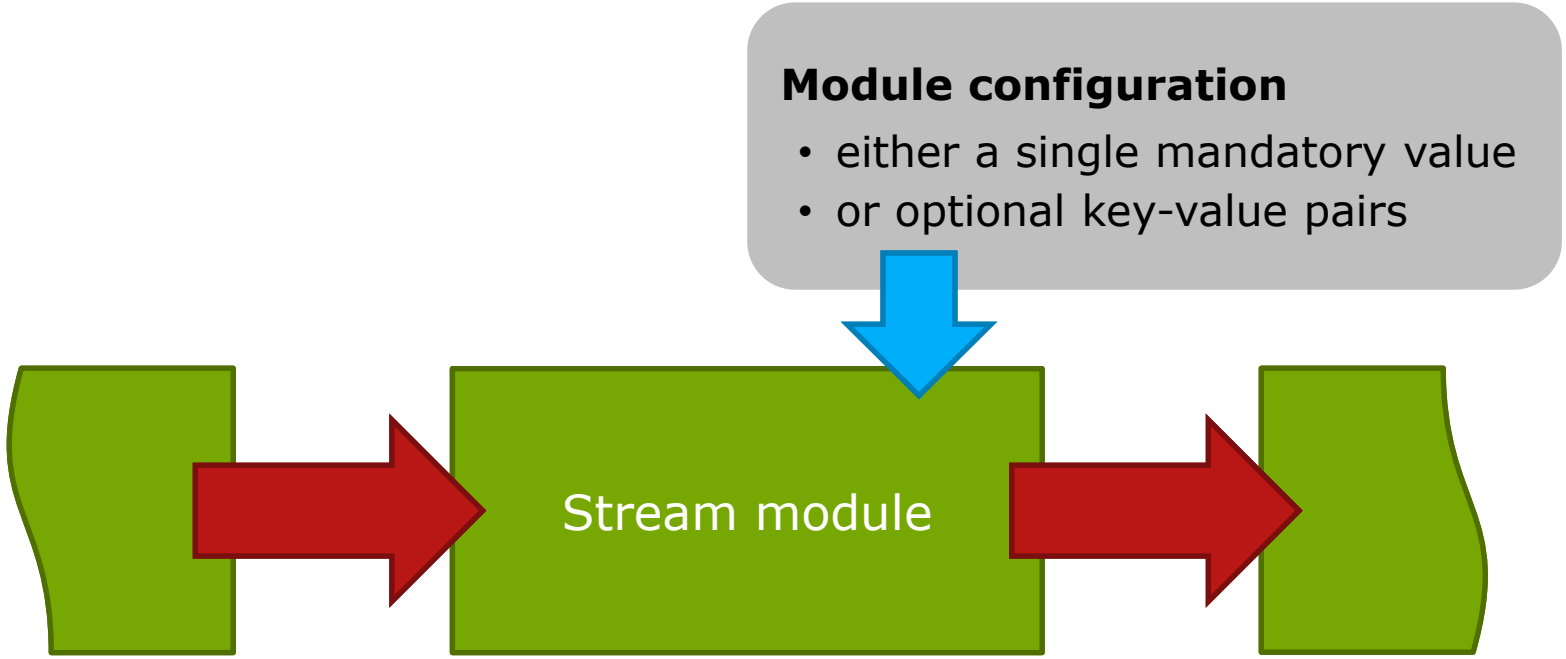


A simple processing flow

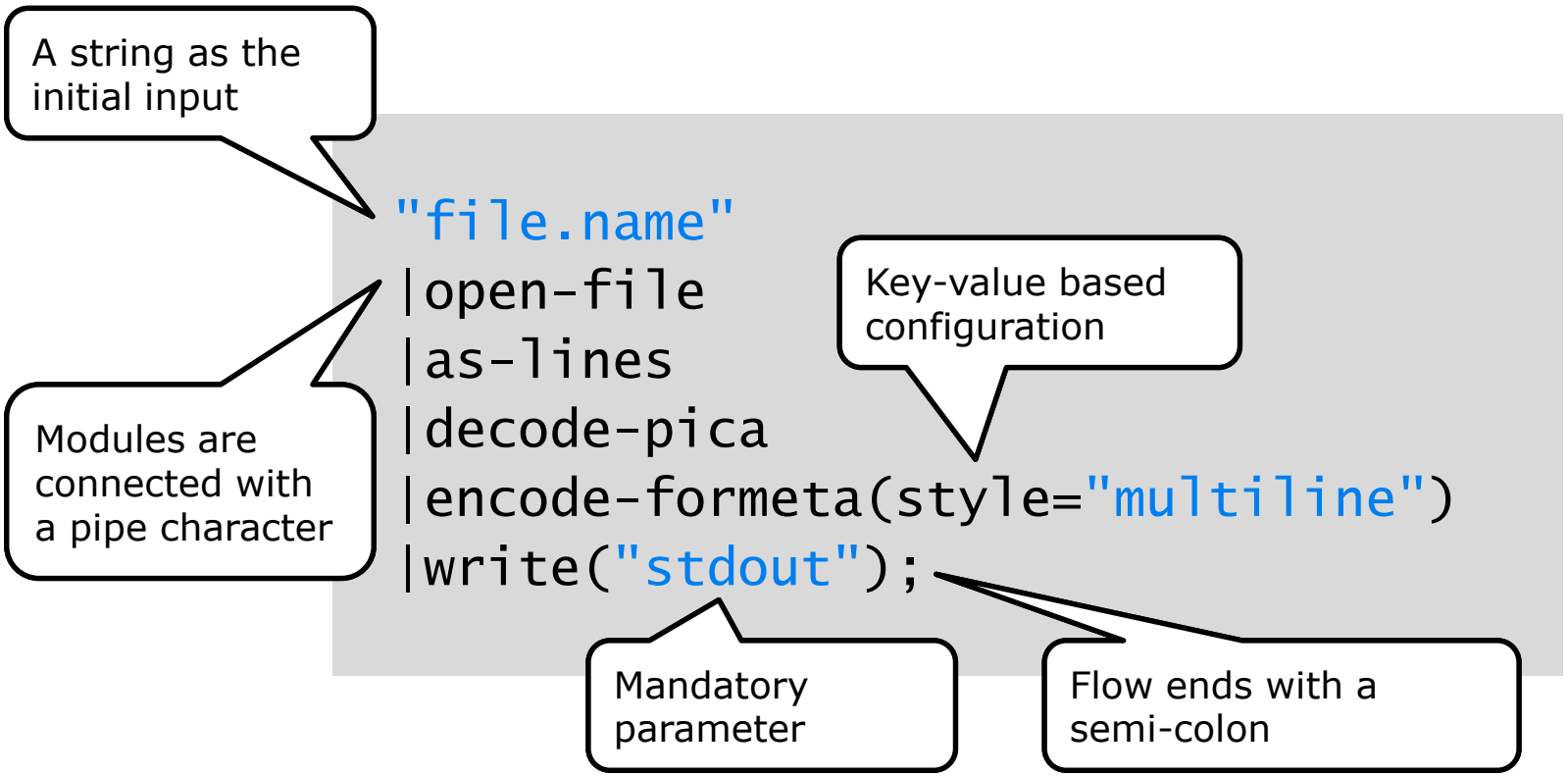
Read and print a file containing pica records:



Module configuration



Describing flows with Flux



Variables and comments in Flux

Define default values for the variables **in** and **out**

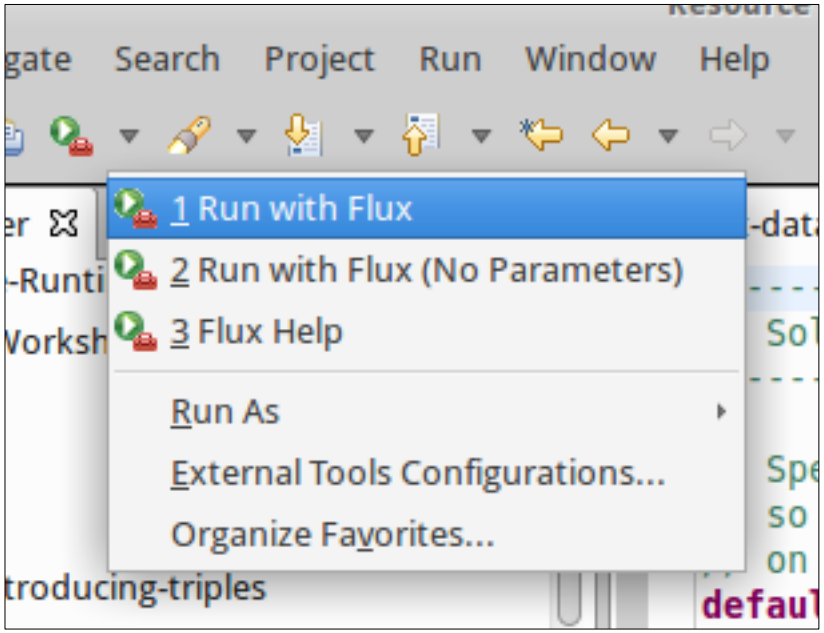
```
default in = "file.name";
default out = "stdout";

in
|open-file
// ...
|write(out);
```

Comments start with two slashes

Use variable instead of directly entering a string

Running Flux scripts



- Flux script must be selected in the IDE
- Choose “Run with Flux” to execute the selected Flux script
- “Flux Help” outputs a list of all supported modules

Representation of metadata in Meta-facture: a stream of events

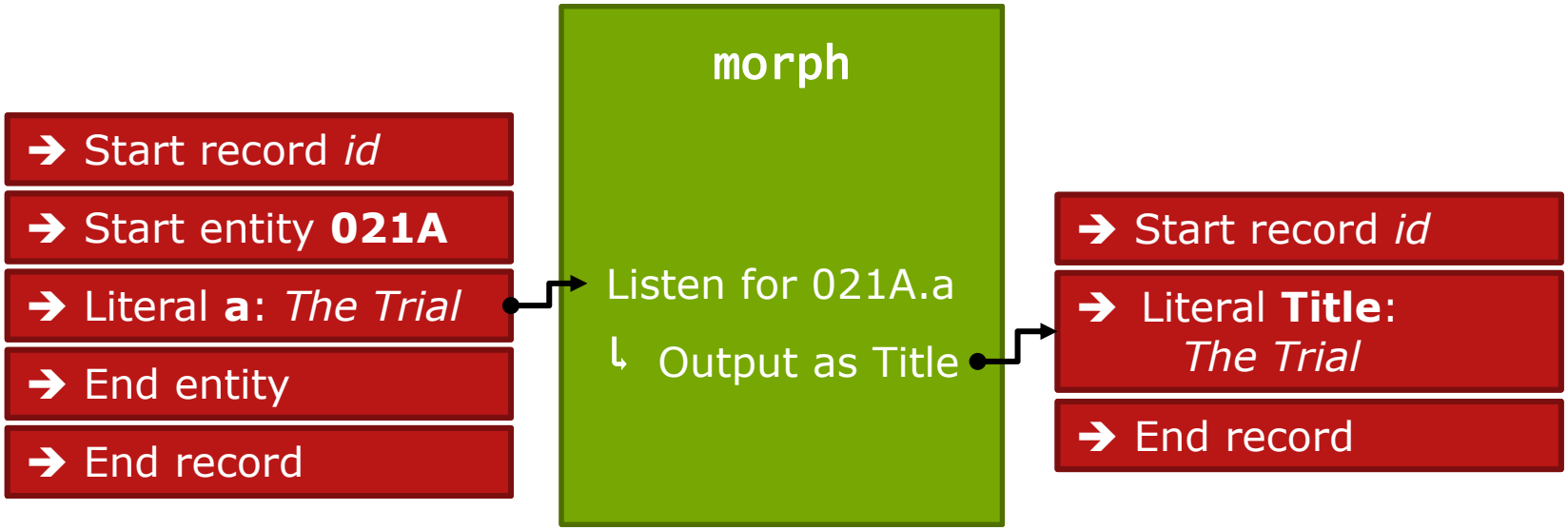
Pica record

```
003@
    $0  2809
033A
    $n  Publisher
    $p  Location
```

Sequence of metadata events

- ➔ Start record *2809*
- ➔ Start entity **003@**
- ➔ Literal **0**: *2809*
- ➔ End entity
- ➔ Start entity **033A**
- ➔ Literal **n**: *Publisher*
- ➔ Literal **p**: *Location*
- ➔ End entity
- ➔ End record

Processing metadata events with Metamorph



Metamorph: data statements

```
<?xml version="1.0" encoding="UTF-8"?>

<metamorph xmlns="http://www.culturegraph.org/metamorph"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   version="1" entityMarker=".">

      <rules>
         <data source="021A.a" name="Title" />
      </rules>

</metamorph>
```

Separator for entities and literal names

Name of the literal to listen for

Name of the literal that is output

Metamorph: modifying data

```
...

<rules>

   <data source="021A.a" name="Title">
       <regexp match="^(The) (.*)$" format="${2}, ${1}" />
   </data>

</rules>

...
```

Process the data value before outputting it. You can specify multiple functions here

Metamorph: combining data

```
...

<rules>

   <combine name="Publisher" value="${Pub}: ${Loc}">
      <data source="033A.n" name="Pub" />
      <data source="033A.p" name="Loc" />
   </combine>

</rules>

...
```

Name of the generated literal. It can include variables, too

Literal value constructed from the variables from the data statements below

The data statements do not generate output but create variables instead

DEUTSCHE NATIONAL BIBLIOTHEK

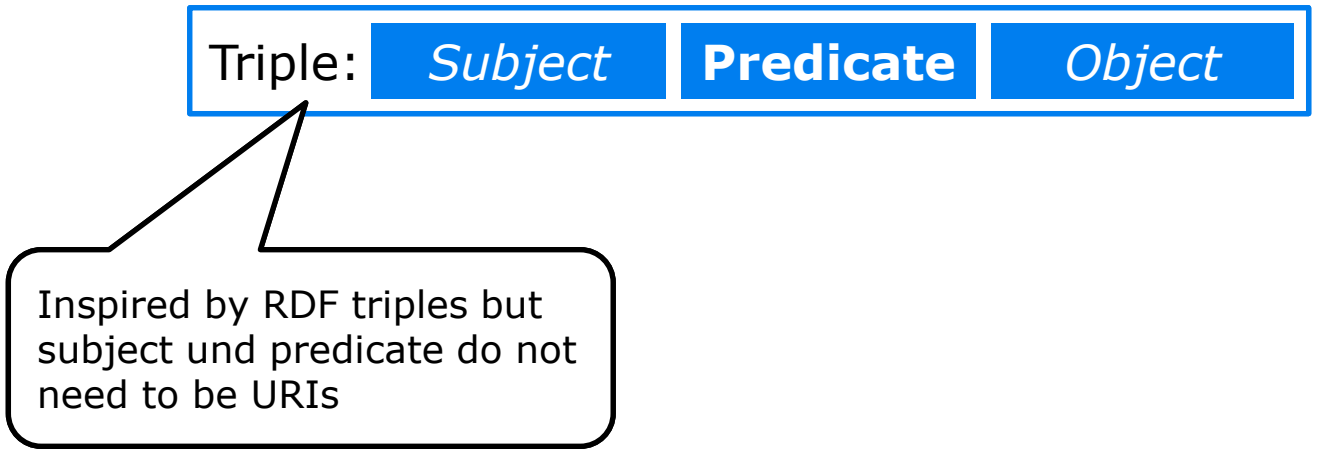
Exercises part 1
Warm-up

DEUTSCHE NATIONAL BIBLIOTHEK

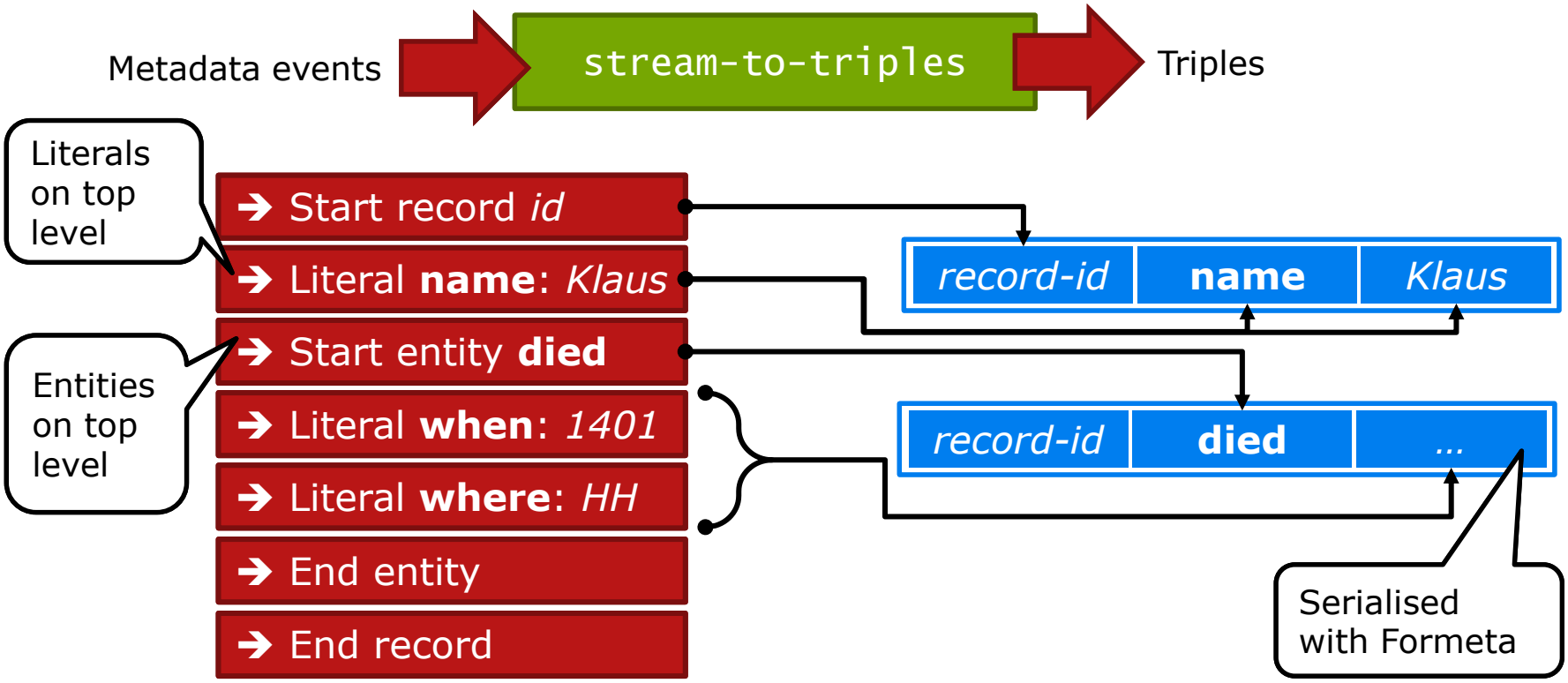
Part 2

Triples and counting

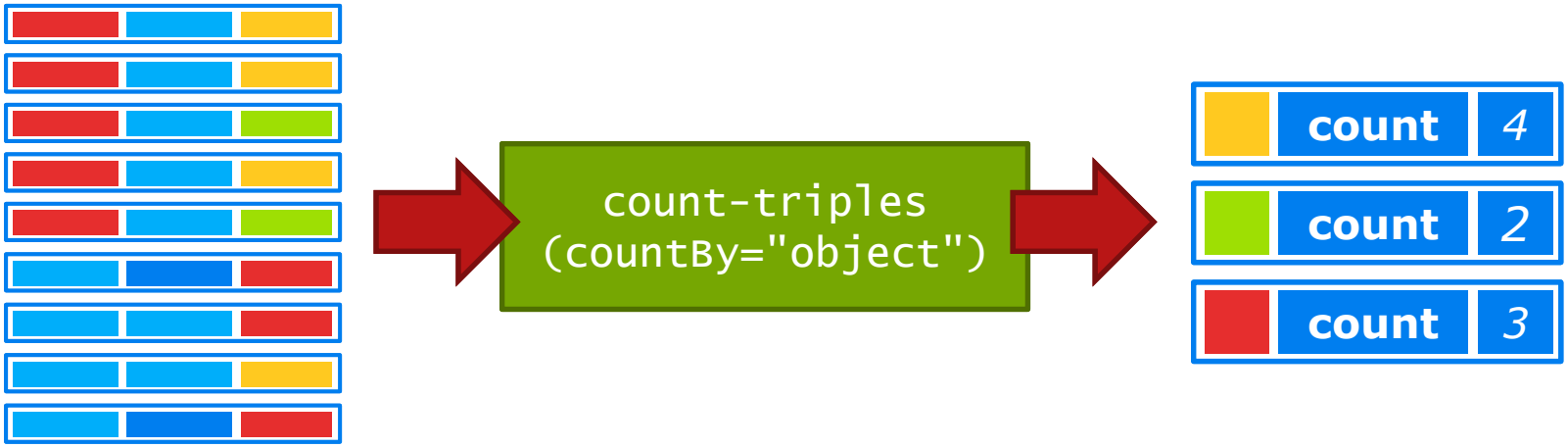
The triple


Triple:
Subject
Predicate
Object
Inspired by RDF triples but subject und predicate do not need to be URIs

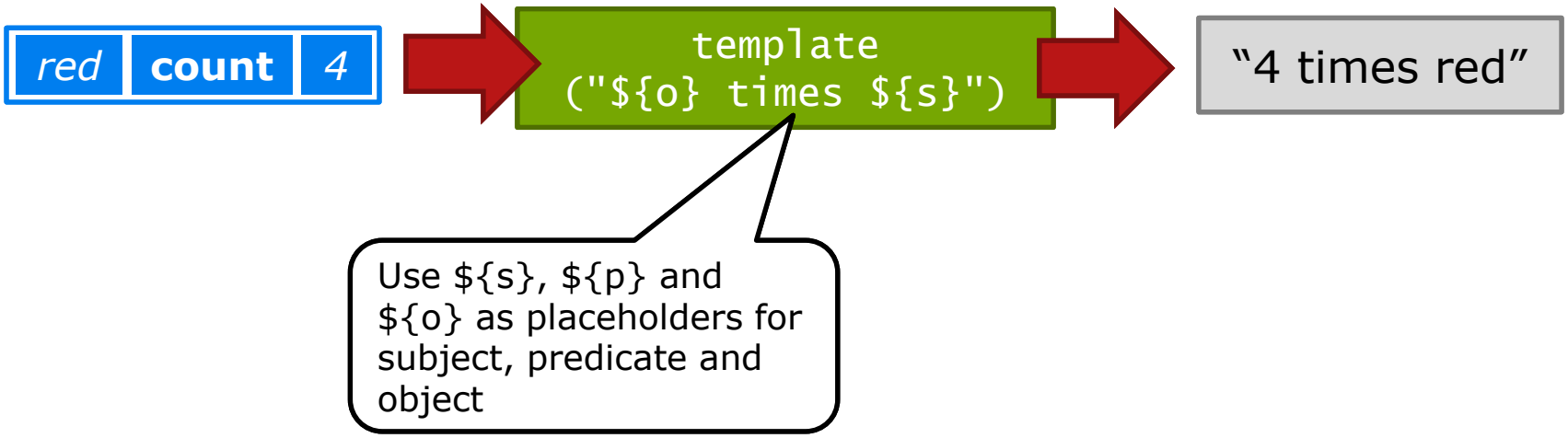
Generating triples



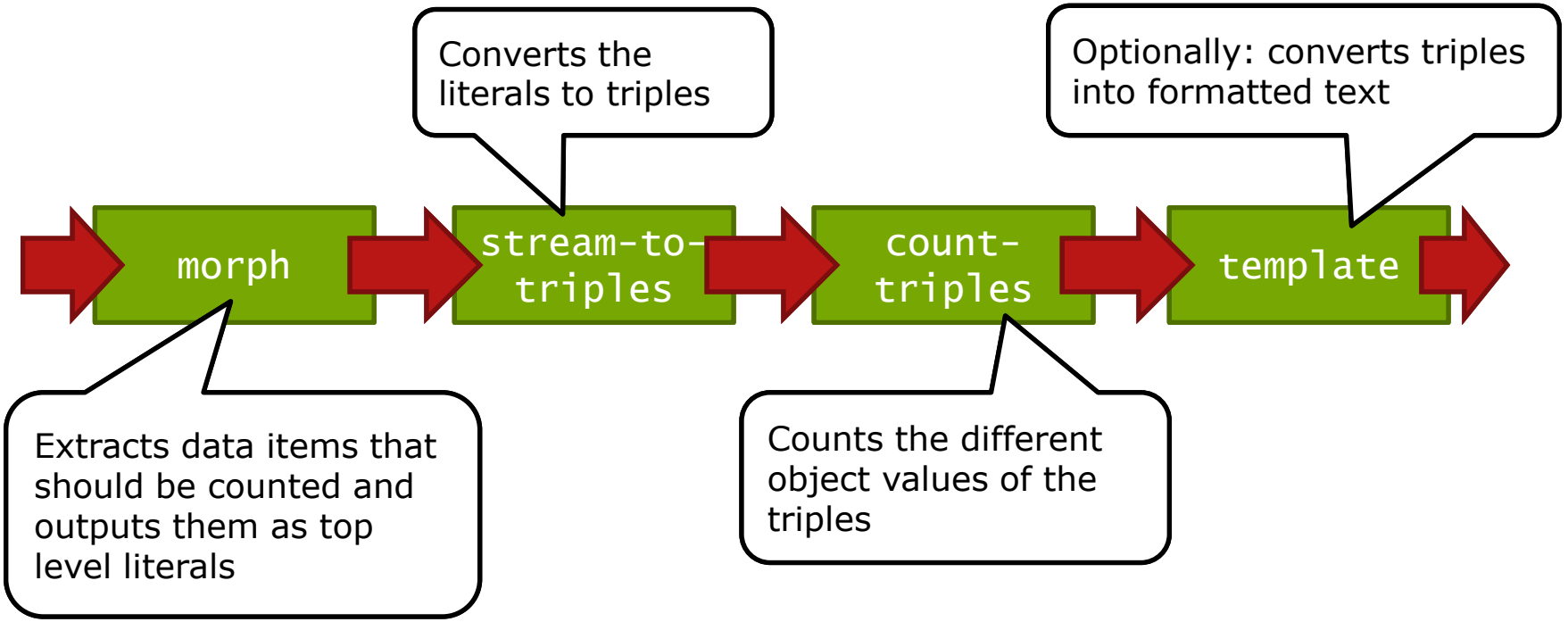
Counting triples



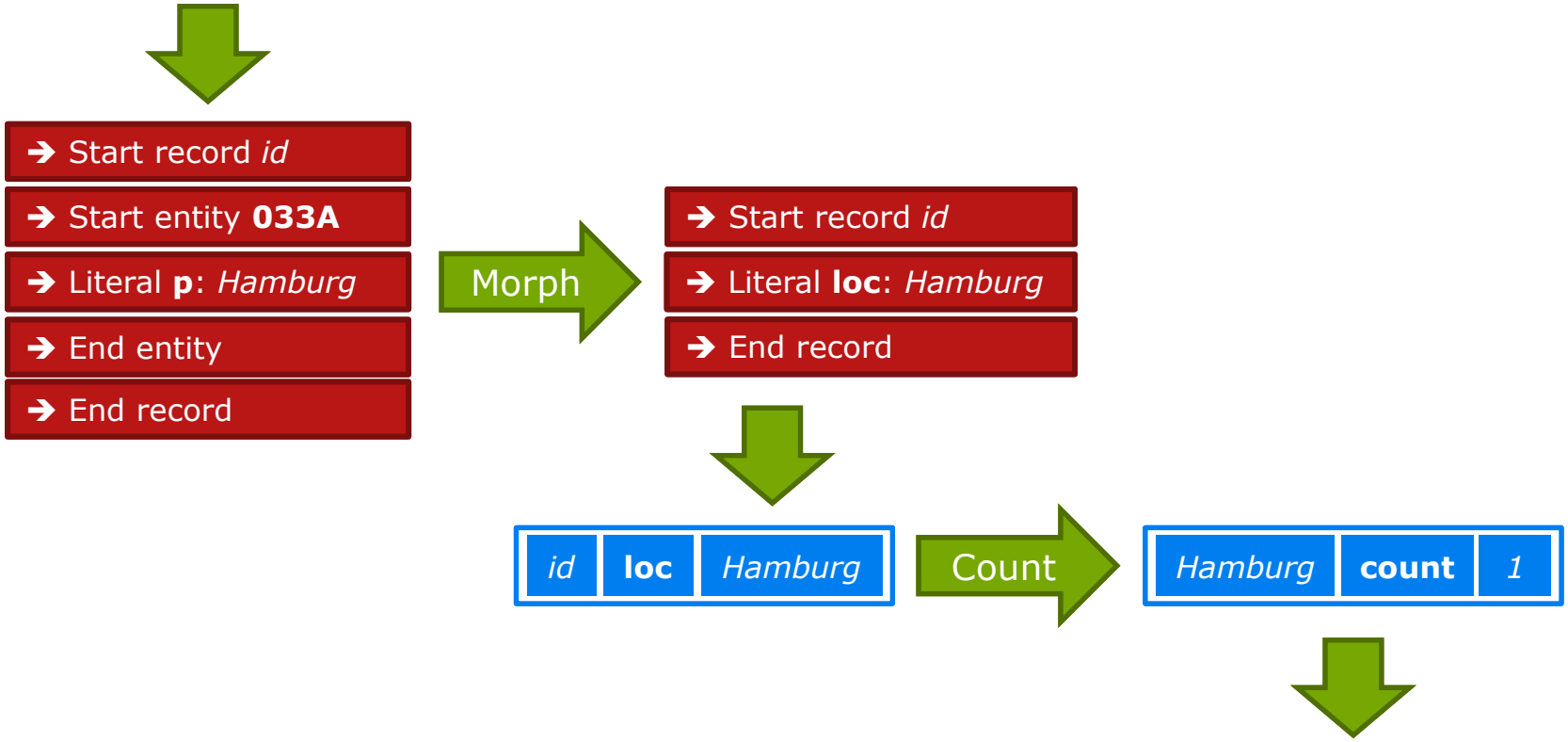
Outputting triples



Counting data values



Counting data values: flow of data



Metamorph: choosing data

```
...
<rules>

   <choose name=„Location">
      <data source="033A.p">
         <regexp match="^Ffm$" format="Frankfurt a. M." />
      </data>
      <data source="033A.p" />
   </choose>

</rules>
...
```

Only the value of the topmost data-statement that generates output is returned by the choose-statement

Metamorph: generating constant values

```
...

<rules>

   <data source="021A.a" name="Title">
       <constant value="All books have the same name" />
   </data>

</rules>

...
```

No matter what the value of literal 021A.a is, always output the defined value

DEUTSCHE NATIONAL BIBLIOTHEK

Exercises part 2

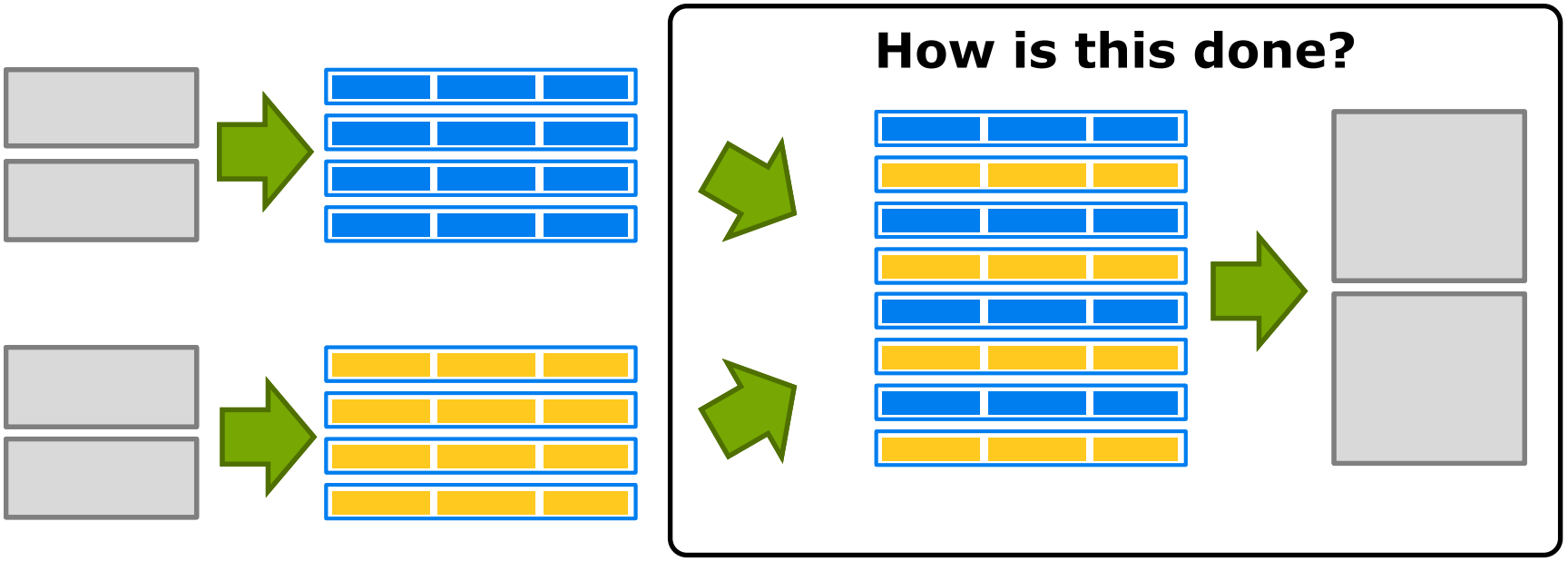
Triples and counting

DEUTSCHE NATIONAL BIBLIOTHEK

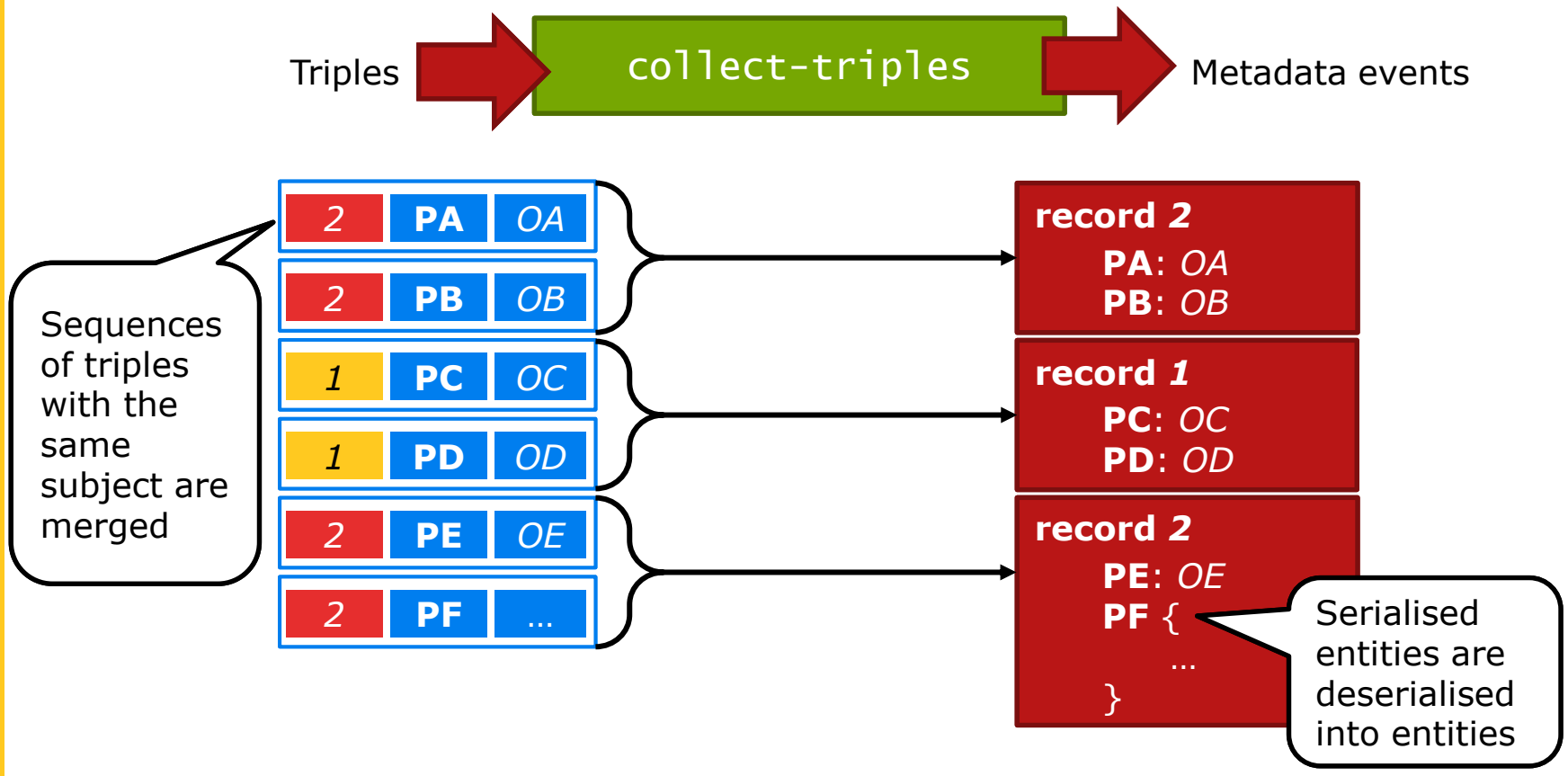
Part 3

Joining data sets and analysing them

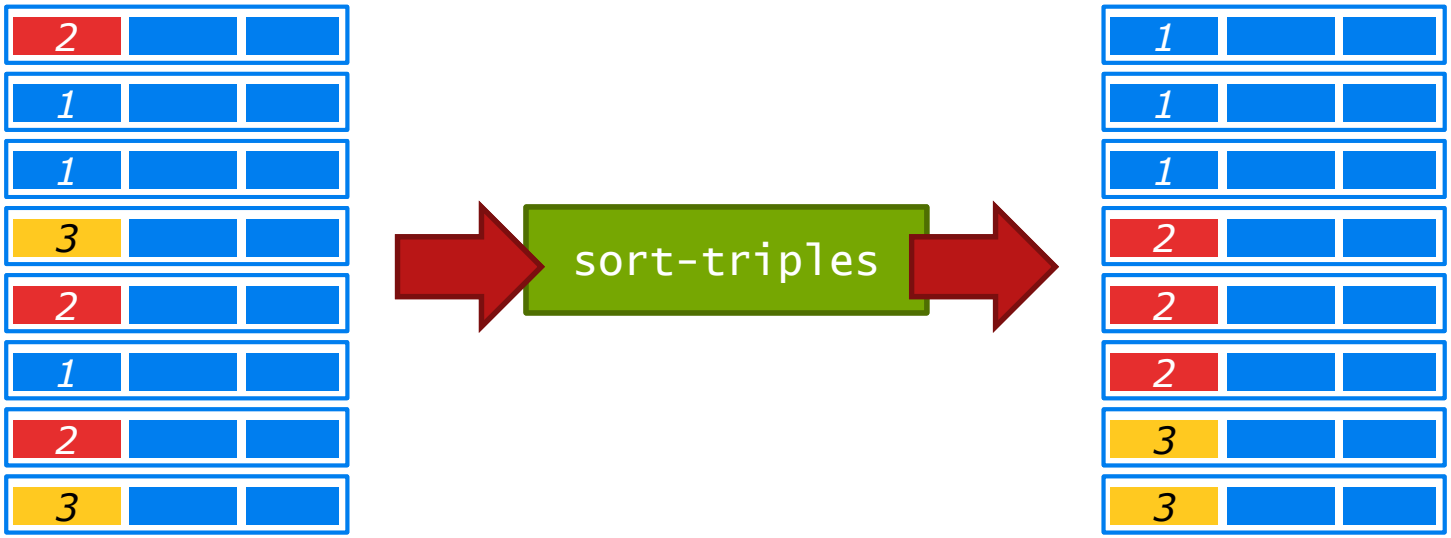
Joining streams of data



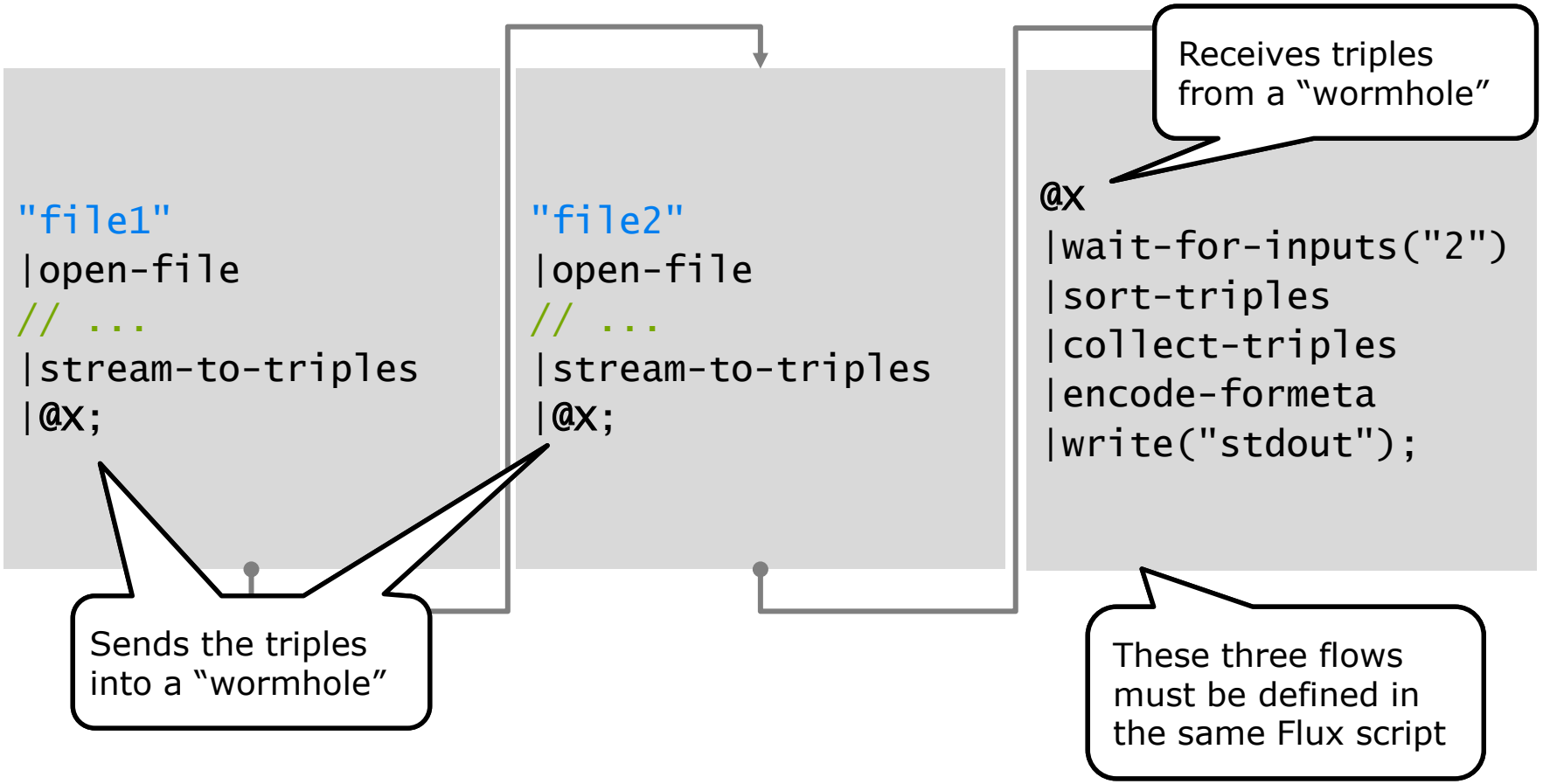
Converting triples into records



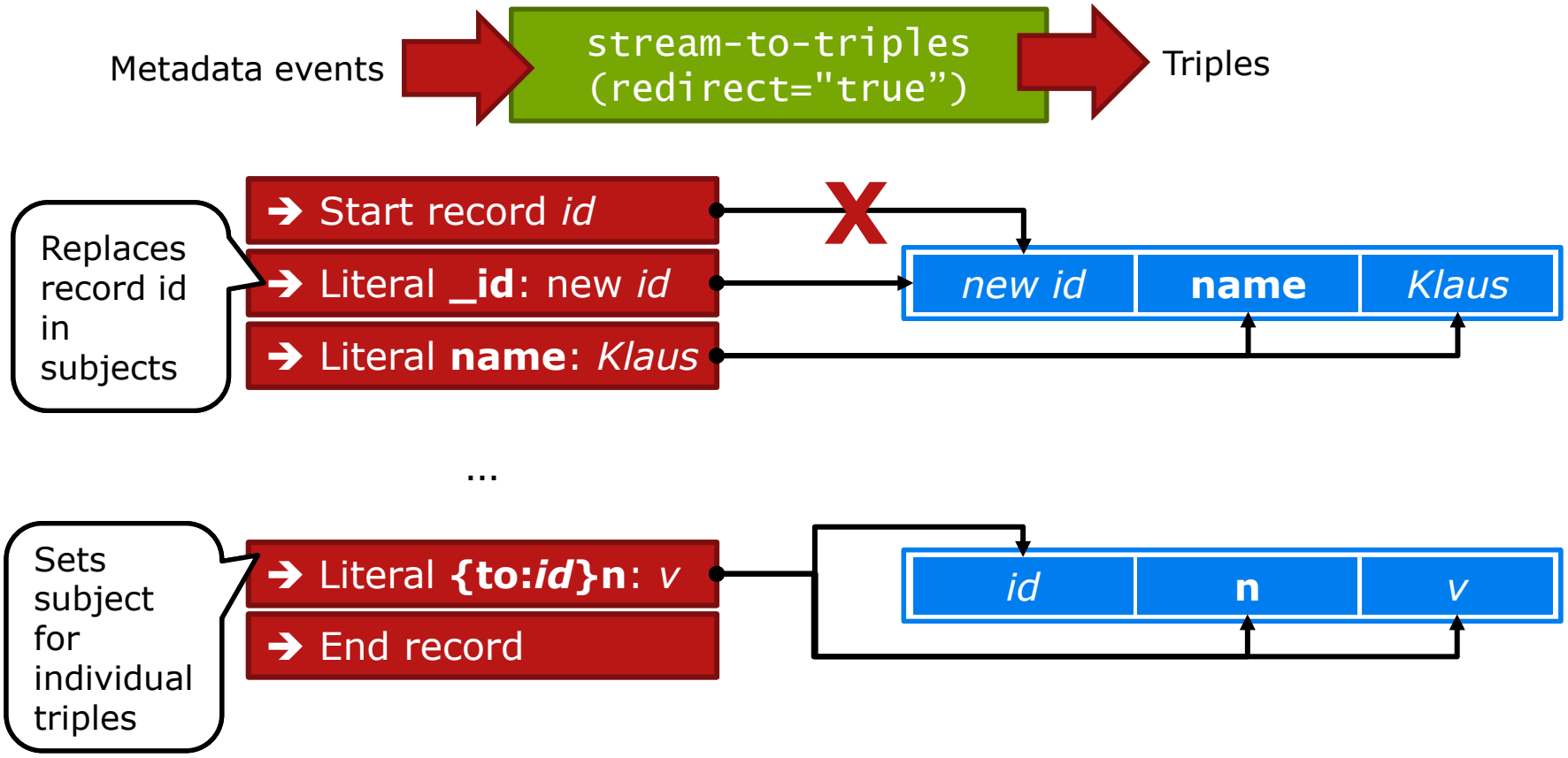
Sorting triples



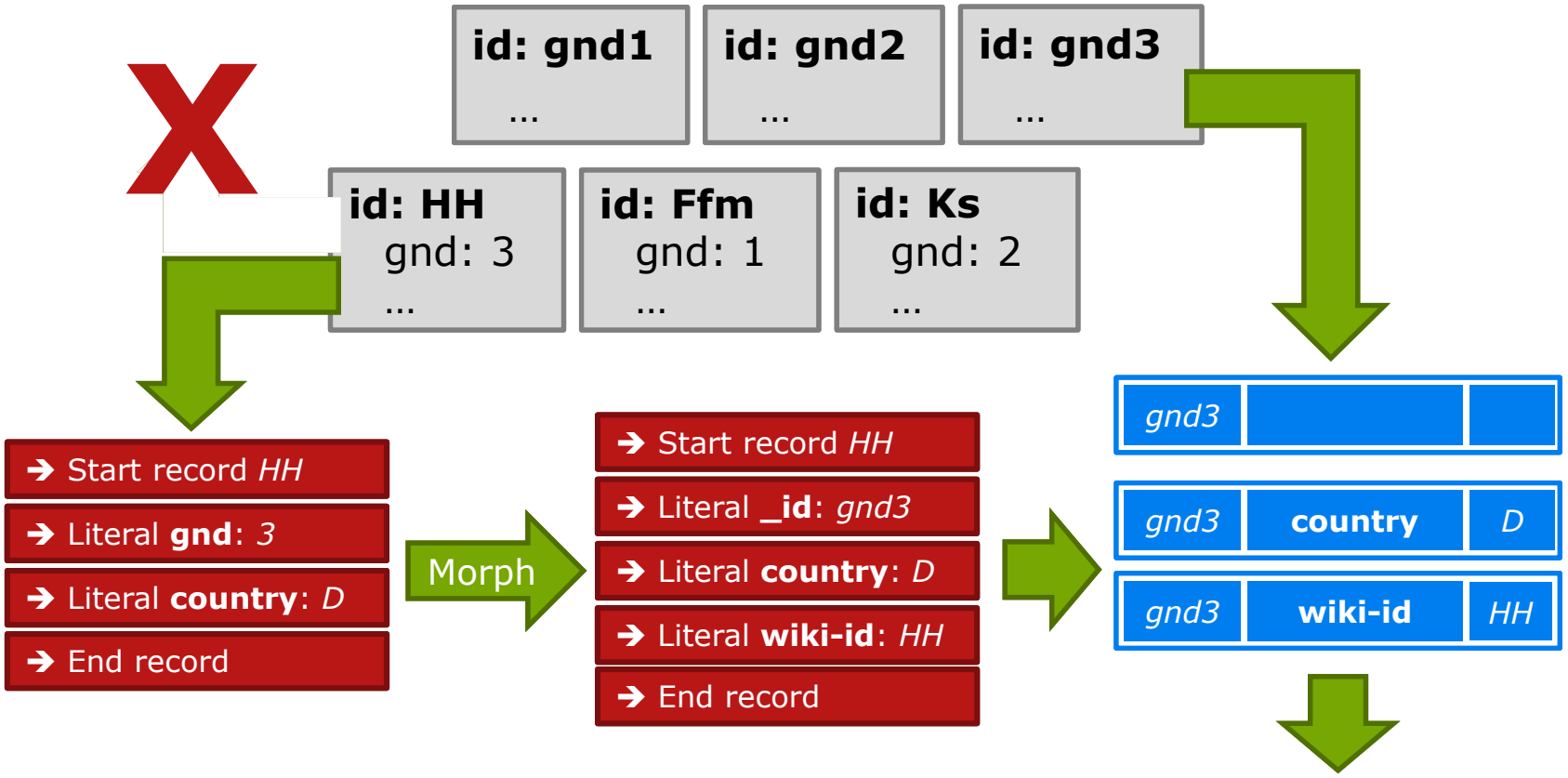
Linking streams in Flux with wormholes



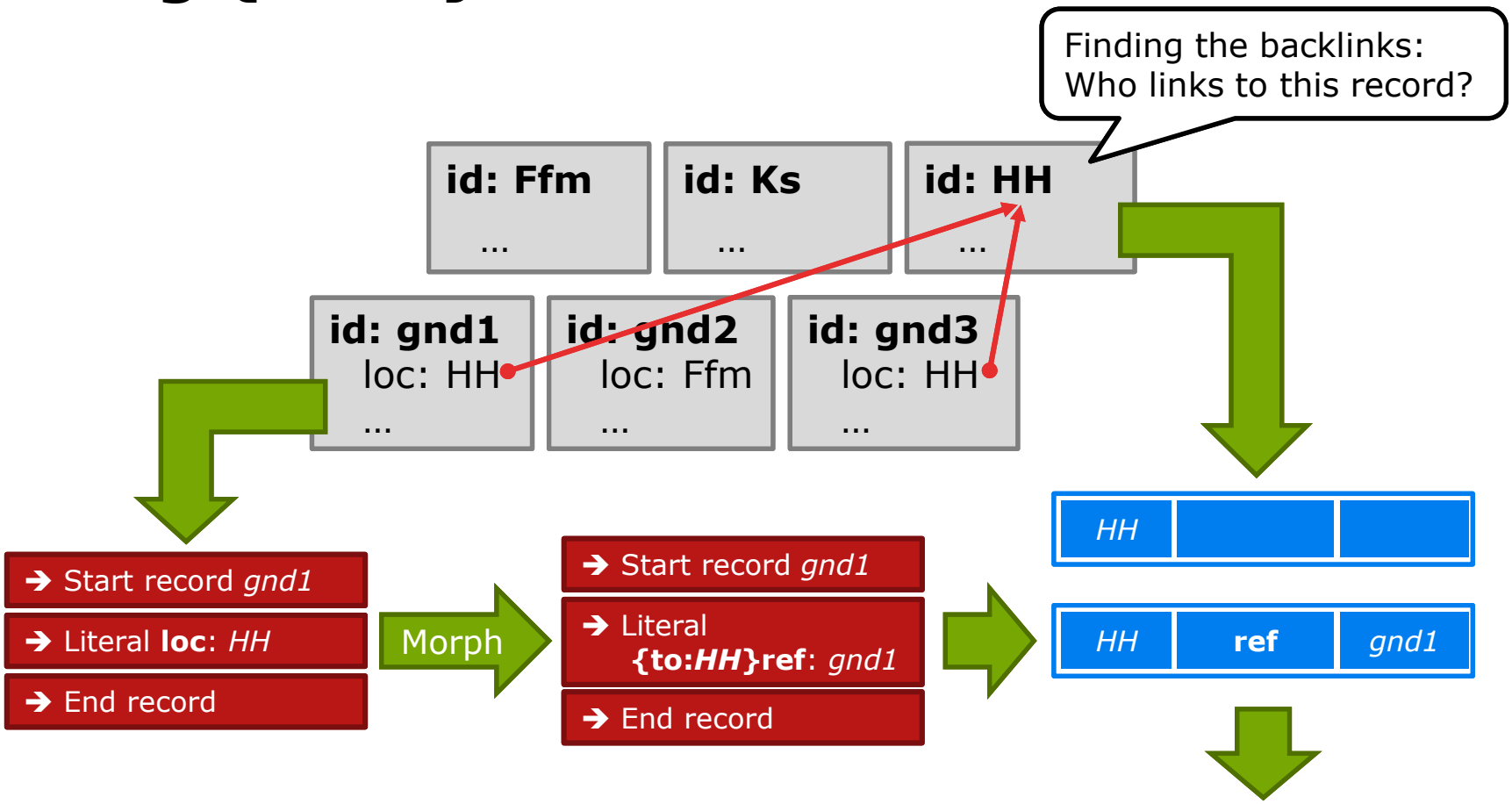
Advanced triplification: ID redirection



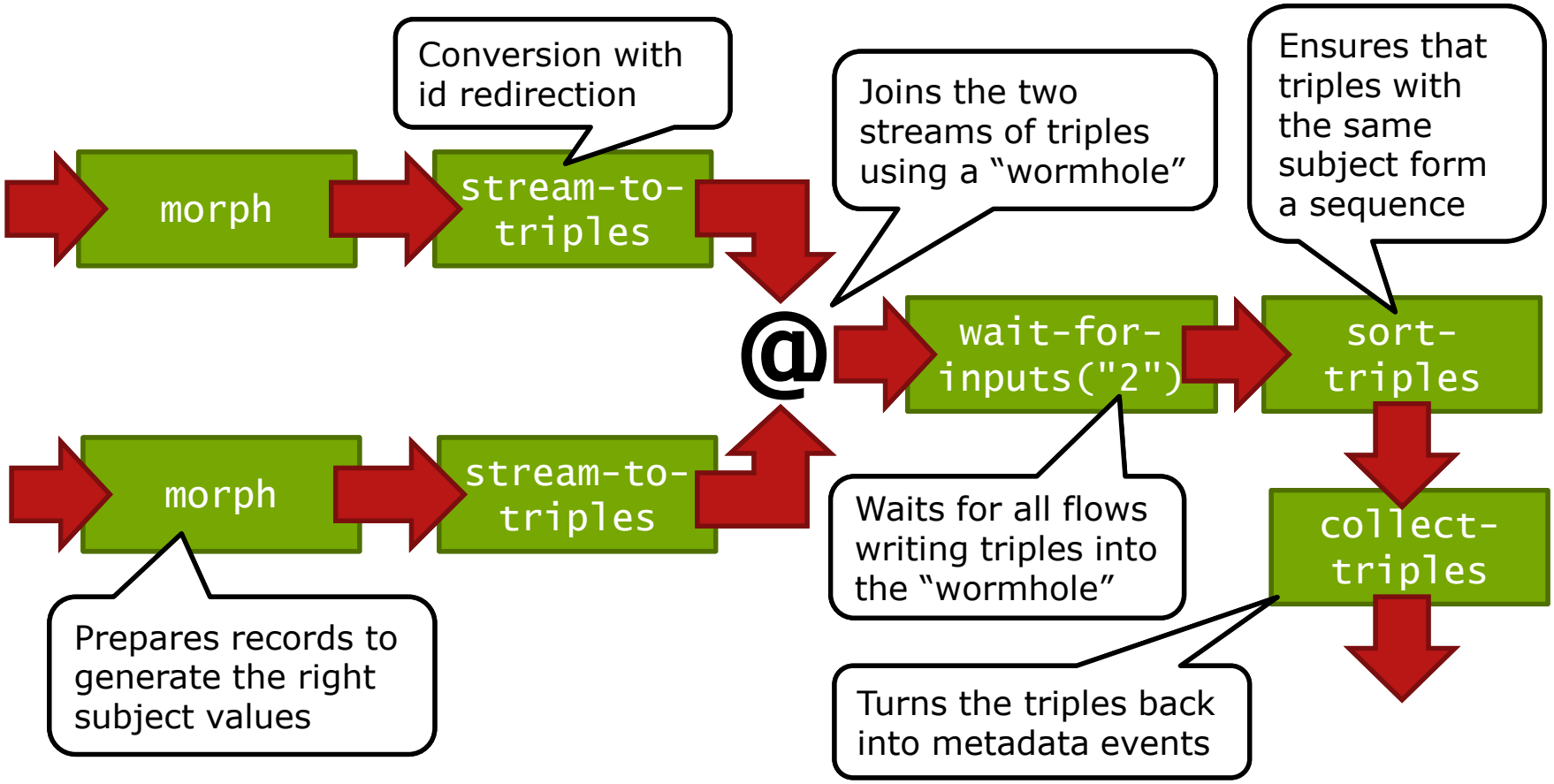
Using _id-redirection



Using {to:ID}-redirection



Putting the pieces together



Metamorph: what else?

```
...

<rules>

   <data source="_id" name="recordId" />
   <data source="033A.n" name="Publisher" />
   <data source="_else" />

</rules>

...
```

Metamorph makes the record id of the current record as `_id` available

Any literal not handled by any other data-statement is passed to this statement. It can be used to pass data through Metamorph

DEUTSCHE NATIONAL BIBLIOTHEK

Exercises part 3

Joining data sets and analysing them

Wrapping up

DEUTSCHE NATIONAL BIBLIOTHEK

What did we learn today?

- Foundations of processing metadata with Flux and Metamorph
- Exploring data sets by quantifying data values
- Joining data sets and analysing their relations
- Typical patterns for analysing data with Metafacture

  These patterns are similar to the way Hadoop operates: This makes migration from your desktop to a Hadoop cluster easy

DEUTSCHE NATIONAL BIBLIOTHEK

Metafacture

- Not only designed for data analysis but for metadata processing in general
- Software tool and library: It can easily be integrated into other applications
- Flux and Metamorph are extendable
- It is open source at http://culturegraph.github.io/

Job advert

We are looking for a software developer for our solr-based search engine infrastructure

For more information please visit:
http://www.dnb.de/stellen

DEUTSCHE NATIONAL BIBLIOTHEK

Thank you very much!

Further questions?

Contact me at c.boehme@dnb.de
or join the mailing list:
http://lists.dnb.de/mailman/listinfo/metafacture