MUSE: Mapping Understanding and deSign by Example

Bogdan Alexe UC Santa Cruz
Laura Chiticariu UC Santa Cruz
Renée J. Miller U. of Toronto
Wang-Chiew Tan UC Santa Cruz

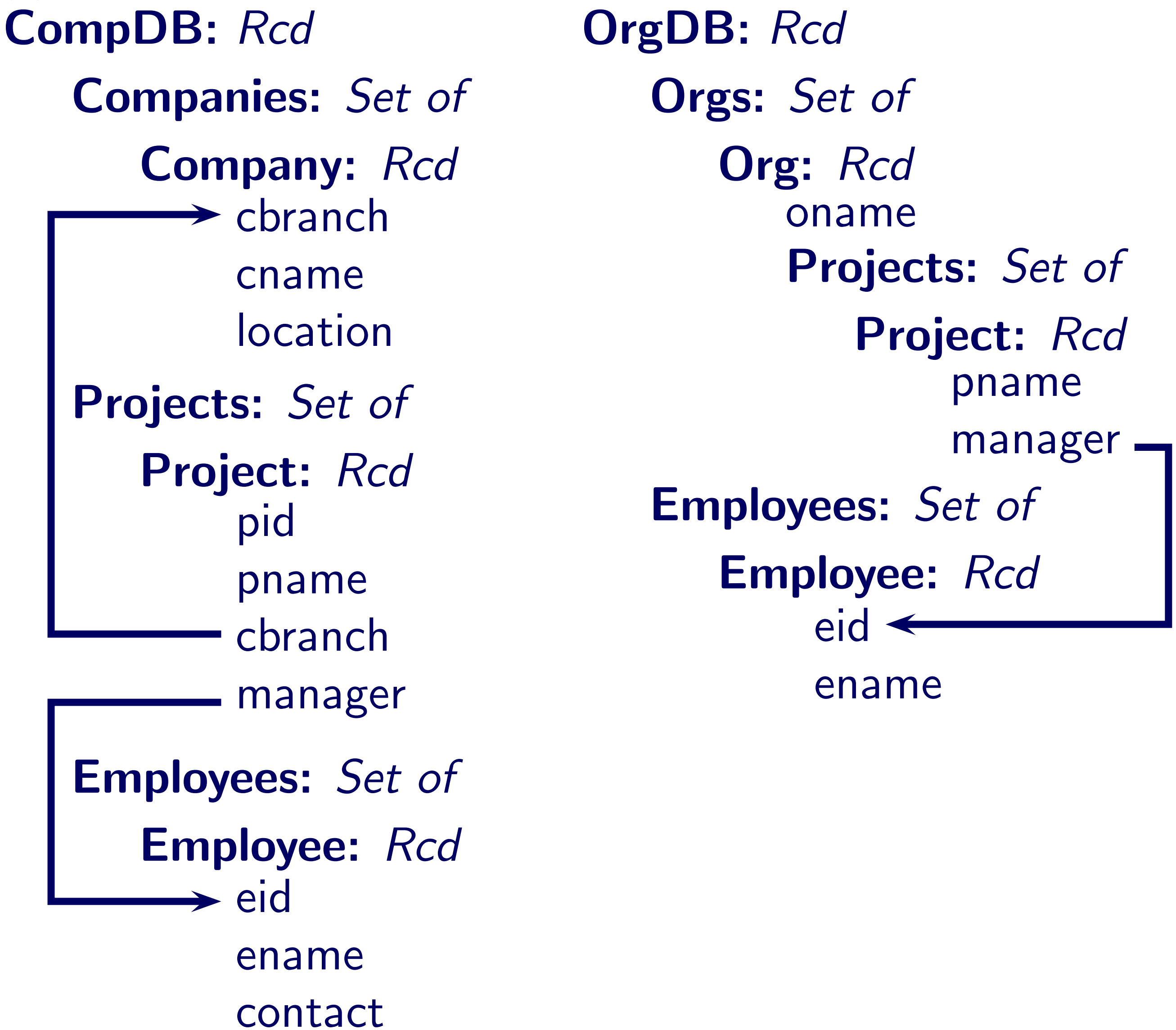
April 8, 2008

Schema Mappings

- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.

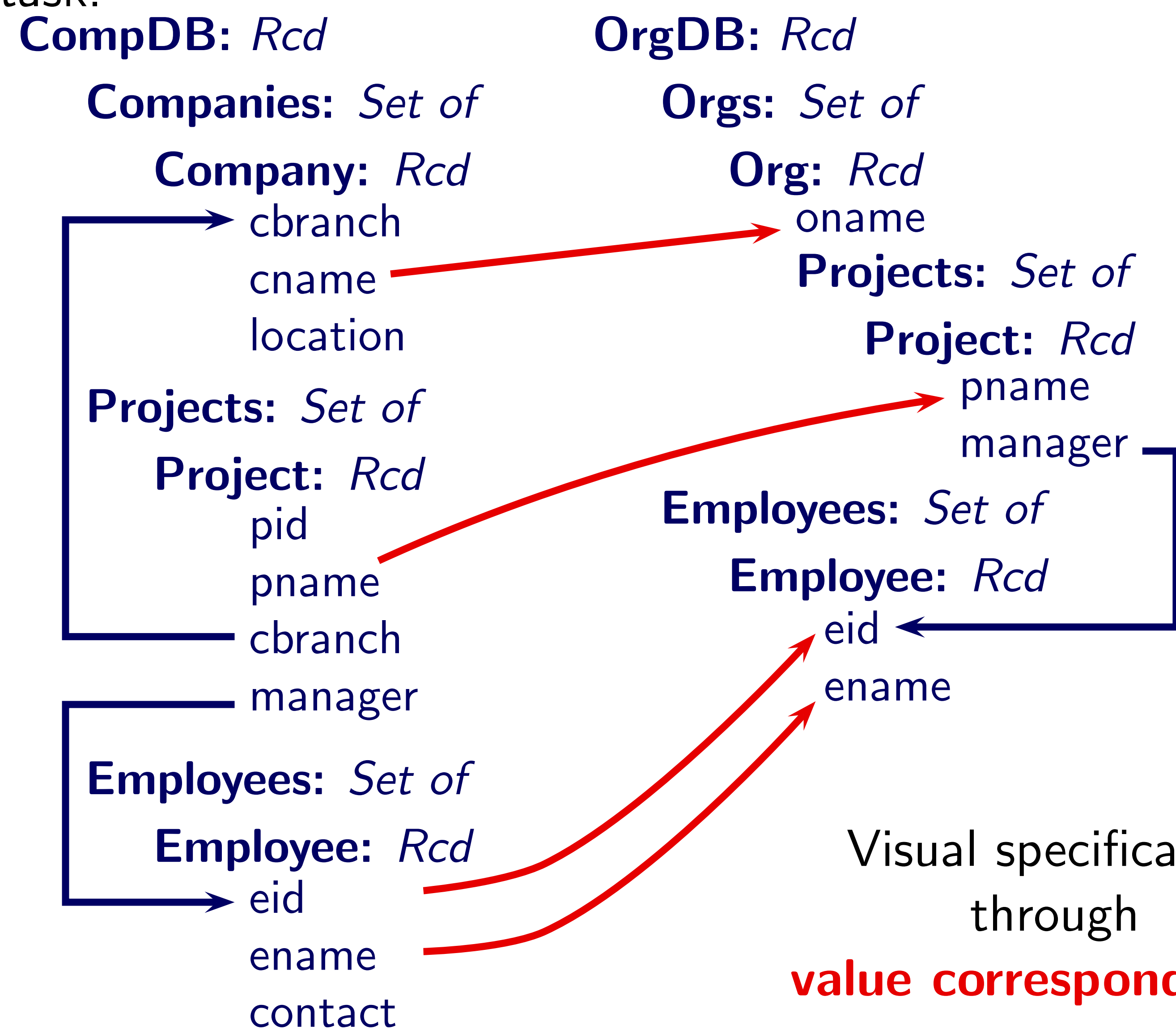
Schema Mappings

- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



Schema Mappings

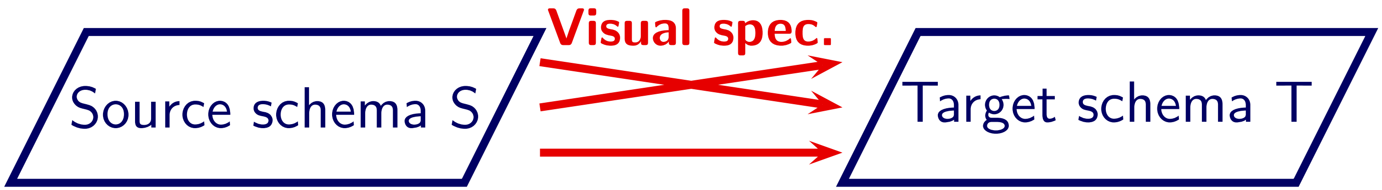
- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



Visual specification through **value correspondences**

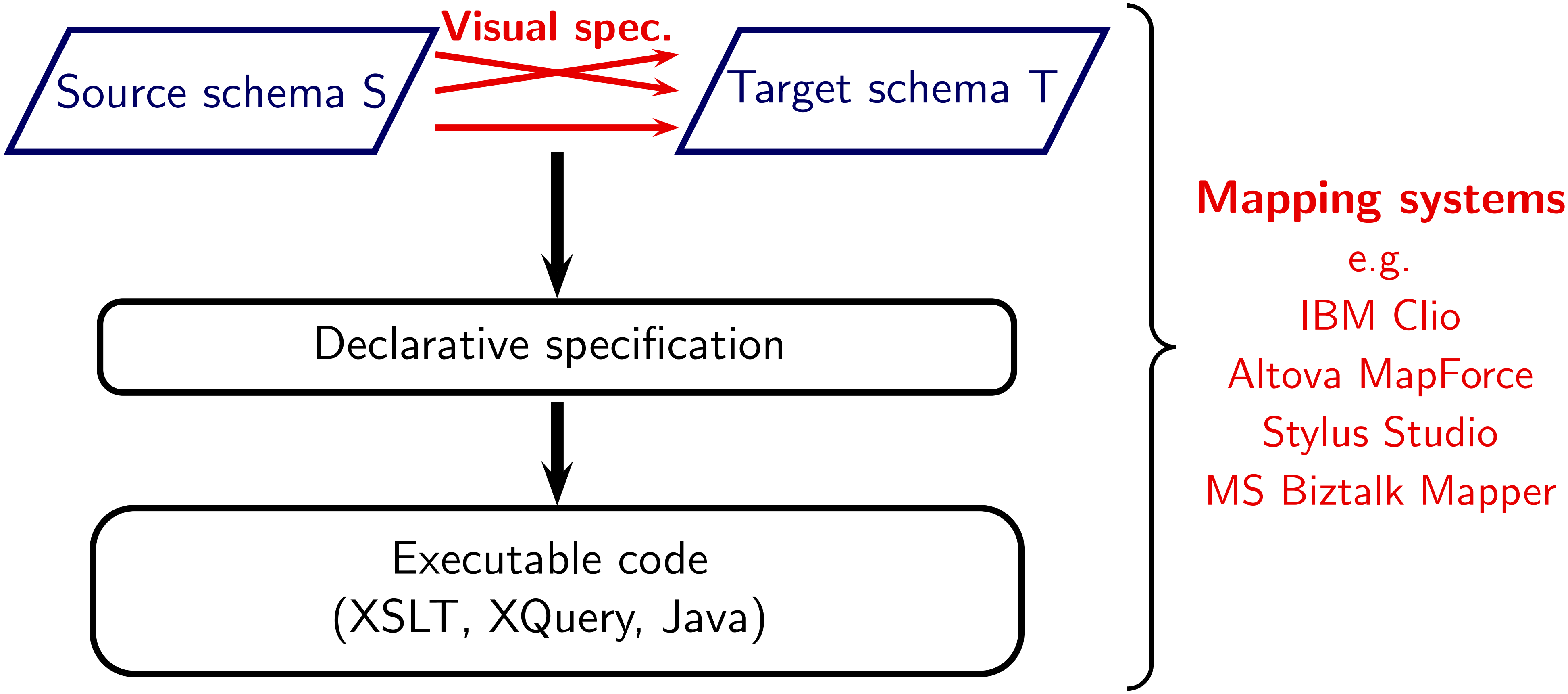
Schema Mappings

- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



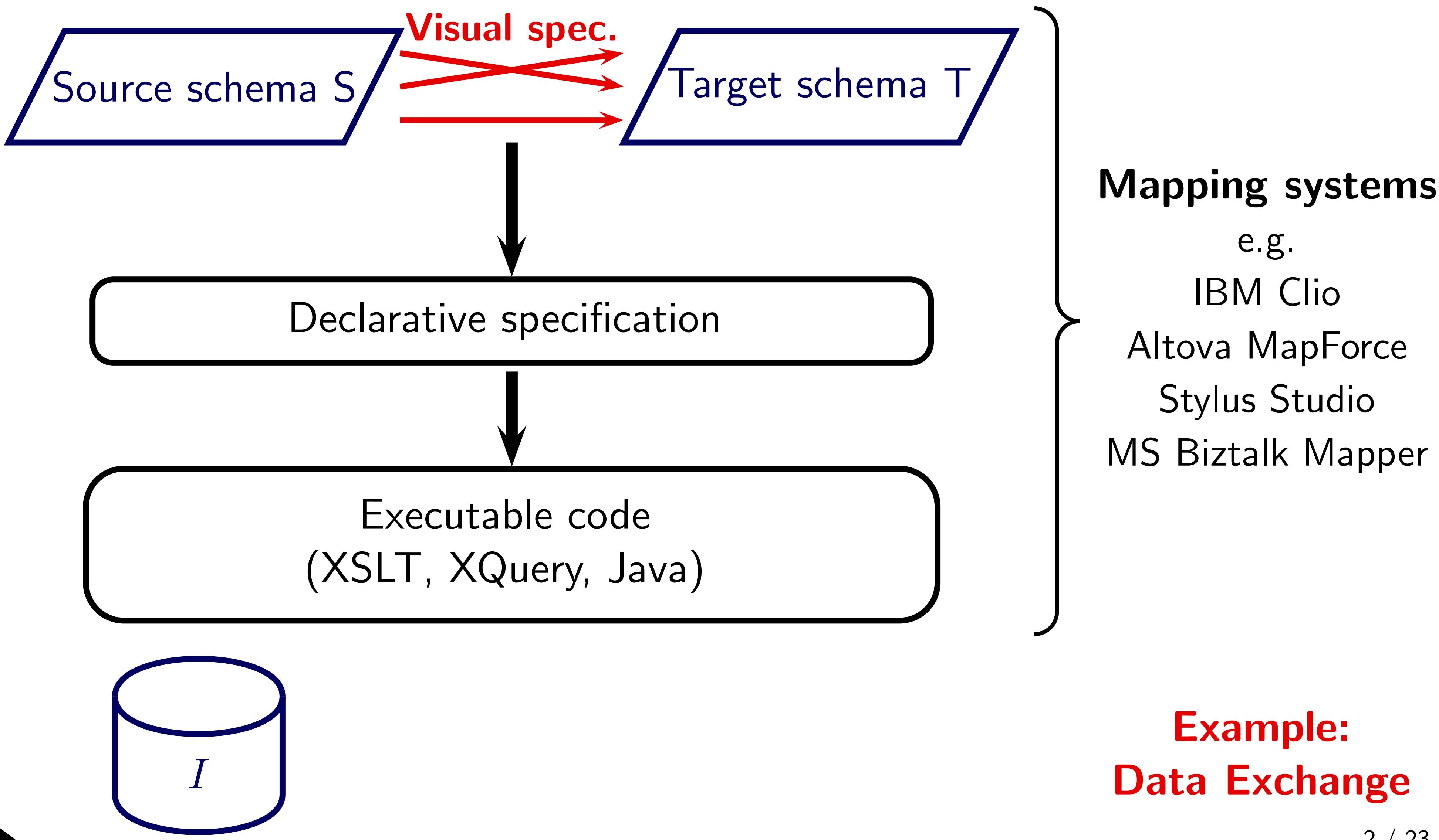
Schema Mappings

- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



Schema Mappings

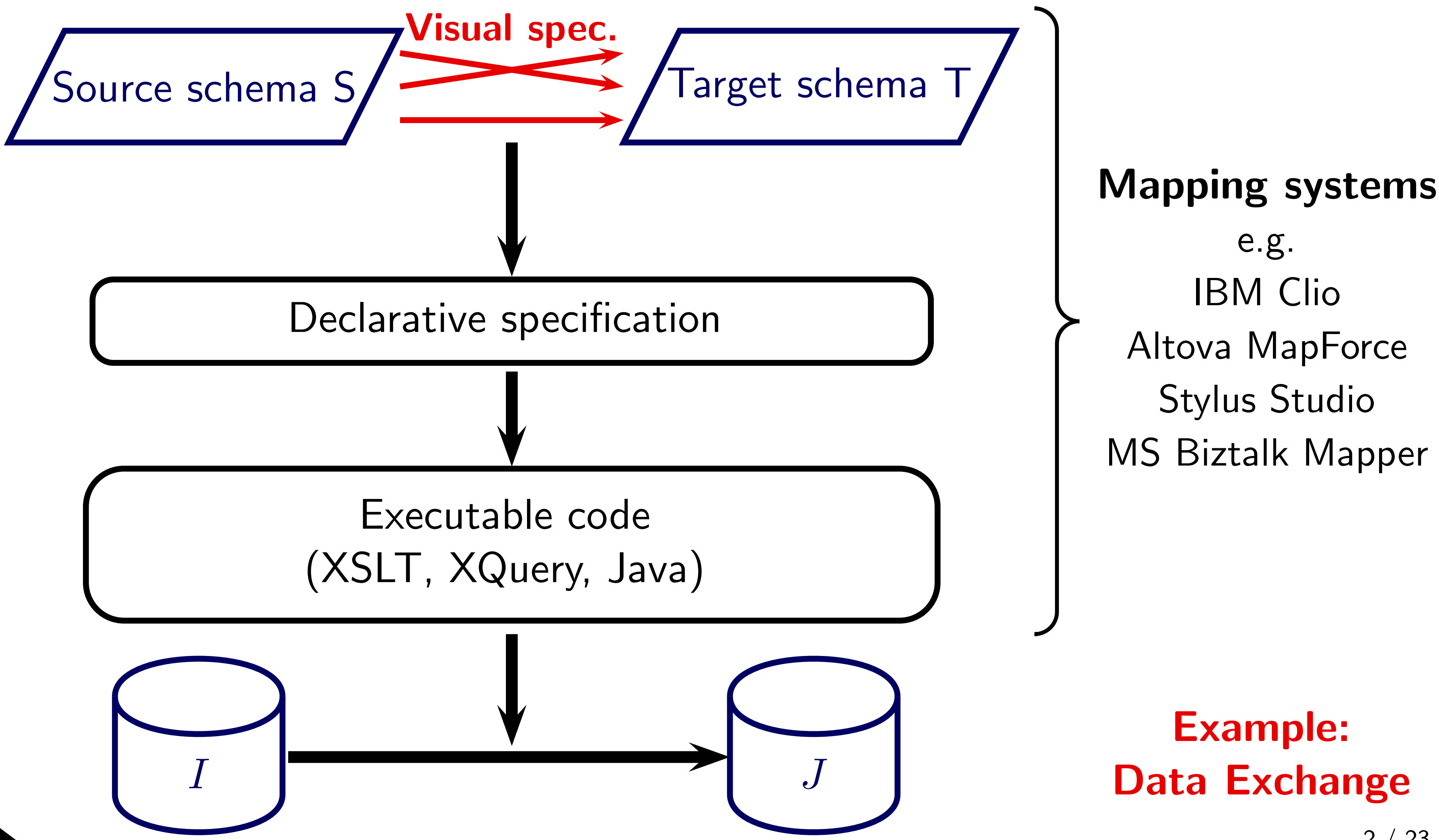
- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



Example:
Data Exchange

Schema Mappings

- One of the first steps in information integration is to specify the relationships (**schema mappings**) between schemas. This is known to be a difficult task.



Example: Data Exchange

Designing Mappings

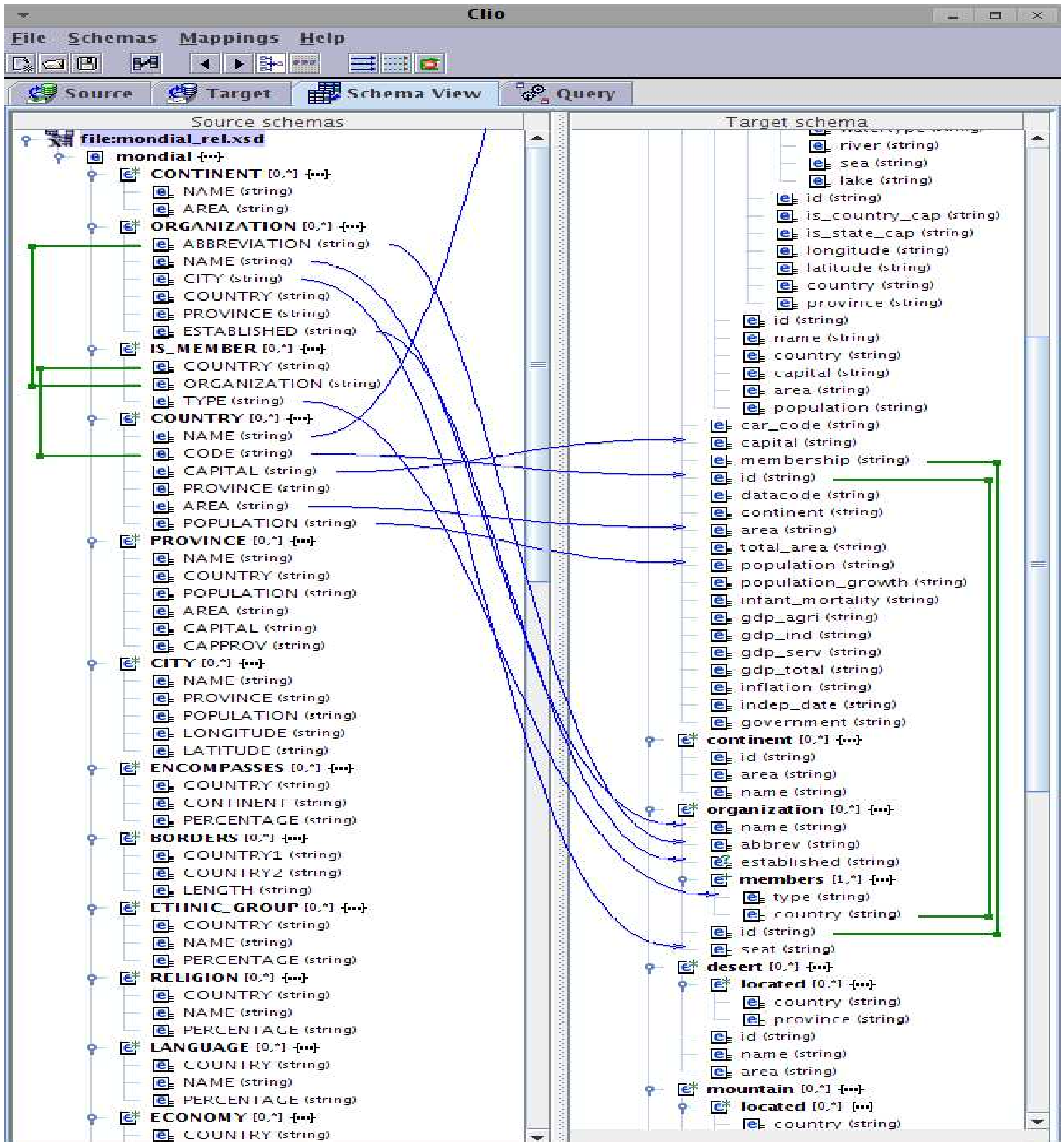
- Mapping systems can automate only part of the mapping process
  - Typically, intricate manual work is needed to perfect the specification

Designing Mappings

- Mapping systems can automate only part of the mapping process
  - Typically, intricate manual work is needed to perfect the specification
- The visual specification may be ambiguous. Mapping systems make default choices to resolve the ambiguities
  - These choices may not correspond to a designer's intentions
  - The mapping designer might refine the specification manually

Real Life Example

In real life scenarios, mappings are extremely complicated



Real Life Example

In real life scenarios, mappings are extremely complicated

```
Map 2:
          for  sm2x0 in S0.dummy_COUNTRY_4
          exists  tm2x0 in S27.dummy_country_10, tm2x1 in S27.dummy_organiza_13
             where  tm2x0.country.membership=tm2x1.organization.id,
          satisf  sm2x0.COUNTRY.AREA=tm2x0.country.area, sm2x0.COUNTRY.CAPITAL=tm2x0.country.capital,
                  sm2x0.COUNTRY.CODE=tm2x0.country.id, sm2x0.COUNTRY.NAME=tm2x0.country.name,
                  sm2x0.COUNTRY.POPULATION=tm2x0.country.population,(
          Map 3:
                  for  sm3x0 in S0.dummy_GEO_RIVE_23, sm3x1 in S0.dummy_RIVER_24,
                          sm3x2 in S0.dummy_PROVINCE_5
                     where  sm3x0.GEO_RIVER.RIVER=sm3x1.RIVER.NAME, sm3x2.PROVINCE.NAME=sm3x0.GEO_RIVER.PROVINCE,
                          sm3x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                  exists  tm3x0 in S27.dummy_river_24, tm3x1 in tm3x0.river.dummy_located_23,
                          tm3x4 in S27.dummy_country_10, tm3x5 in tm3x4.country.dummy_province_9,
                          tm3x6 in S27.dummy_organiza_13
                  where  tm3x4.country.membership=tm3x6.organization.id, tm3x5.province.id=tm3x1.located.province,
                          tm2x0.country.id=tm3x1.located.country,
                  satisf  sm2x0.COUNTRY.AREA=tm3x4.country.area, sm2x0.COUNTRY.CAPITAL=tm3x4.country.capital,
                          sm2x0.COUNTRY.CODE=tm3x4.country.id, sm2x0.COUNTRY.NAME=tm3x4.country.name,
                          sm2x0.COUNTRY.POPULATION=tm3x4.country.population, sm3x1.RIVER.LENGTH=tm3x0.river.length,
                          sm3x0.GEO_RIVER.COUNTRY=tm3x1.located.country, sm3x0.GEO_RIVER.PROVINCE=tm3x1.located.province,
                          sm3x1.RIVER.NAME=tm3x0.river.name ),(
          Map 4:
                  for  sm4x0 in S0.dummy_GEO_ISLA_25, sm4x1 in S0.dummy_ISLAND_26,
                          sm4x2 in S0.dummy_PROVINCE_5
                     where  sm4x0.GEO_ISLAND.ISLAND=sm4x1.ISLAND.NAME, sm4x2.PROVINCE.NAME=sm4x0.GEO_ISLAND.PROVINCE,
                          sm4x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                  exists  tm4x0 in S27.dummy_island_26, tm4x1 in tm4x0.island.dummy_located_25,
                          tm4x4 in S27.dummy_country_10, tm4x5 in tm4x4.country.dummy_province_9,
                          tm4x6 in S27.dummy_organiza_13
                     where  tm4x4.country.membership=tm4x6.organization.id, tm4x5.province.id=tm4x1.located.province,
                          tm2x0.country.id=tm4x1.located.country,
                  satisf  sm2x0.COUNTRY.AREA=tm4x4.country.area, sm2x0.COUNTRY.CAPITAL=tm4x4.country.capital,
                          sm2x0.COUNTRY.CODE=tm4x4.country.id, sm2x0.COUNTRY.NAME=tm4x4.country.name,
                          sm2x0.COUNTRY.POPULATION=tm4x4.country.population, sm4x1.ISLAND.AREA=tm4x0.island.area,
                          sm4x1.ISLAND.COORDINATESLAT=tm4x0.island.latitude, sm4x0.GEO_ISLAND.COUNTRY=tm4x1.located.country,
                          sm4x0.GEO_ISLAND.PROVINCE=tm4x1.located.province, sm4x1.ISLAND.COORDINATESLONG=tm4x0.island.longitude,
                          sm4x1.ISLAND.NAME=tm4x0.island.name ),(
          Map 5:
                  for  sm5x0 in S0.dummy_GEO_SEA_19, sm5x1 in S0.dummy_SEA_20,
                          sm5x2 in S0.dummy_PROVINCE_5
                     where  sm5x2.PROVINCE.NAME=sm5x0.GEO_SEA.PROVINCE, sm5x0.GEO_SEA.SEA=sm5x1.SEA.NAME,
                                  sm5x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                  exists  tm5x0 in S27.dummy_sea_19, tm5x1 in tm5x0.sea.dummy_located_18,
                          tm5x4 in S27.dummy_country_10, tm5x5 in tm5x4.country.dummy_province_9,
                          tm5x6 in S27.dummy_organiza_13
                     where  tm5x4.country.membership=tm5x6.organization.id, tm5x5.province.id=tm5x1.located.province,
                          tm2x0.country.id=tm5x1.located.country,
                  satisf  sm2x0.COUNTRY.AREA=tm5x4.country.area, sm2x0.COUNTRY.CAPITAL=tm5x4.country.capital,
                          sm2x0.COUNTRY.CODE=tm5x4.country.id, sm2x0.COUNTRY.NAME=tm5x4.country.name,
                          sm2x0.COUNTRY.POPULATION=tm5x4.country.population, sm5x1.SEA.DEPTH=tm5x0.sea.depth,
                          sm5x0.GEO_SEA.COUNTRY=tm5x1.located.country, sm5x0.GEO_SEA.PROVINCE=tm5x1.located.province,
                          sm5x1.SEA.NAME=tm5x0.sea.name ),(
```

Designing Mappings

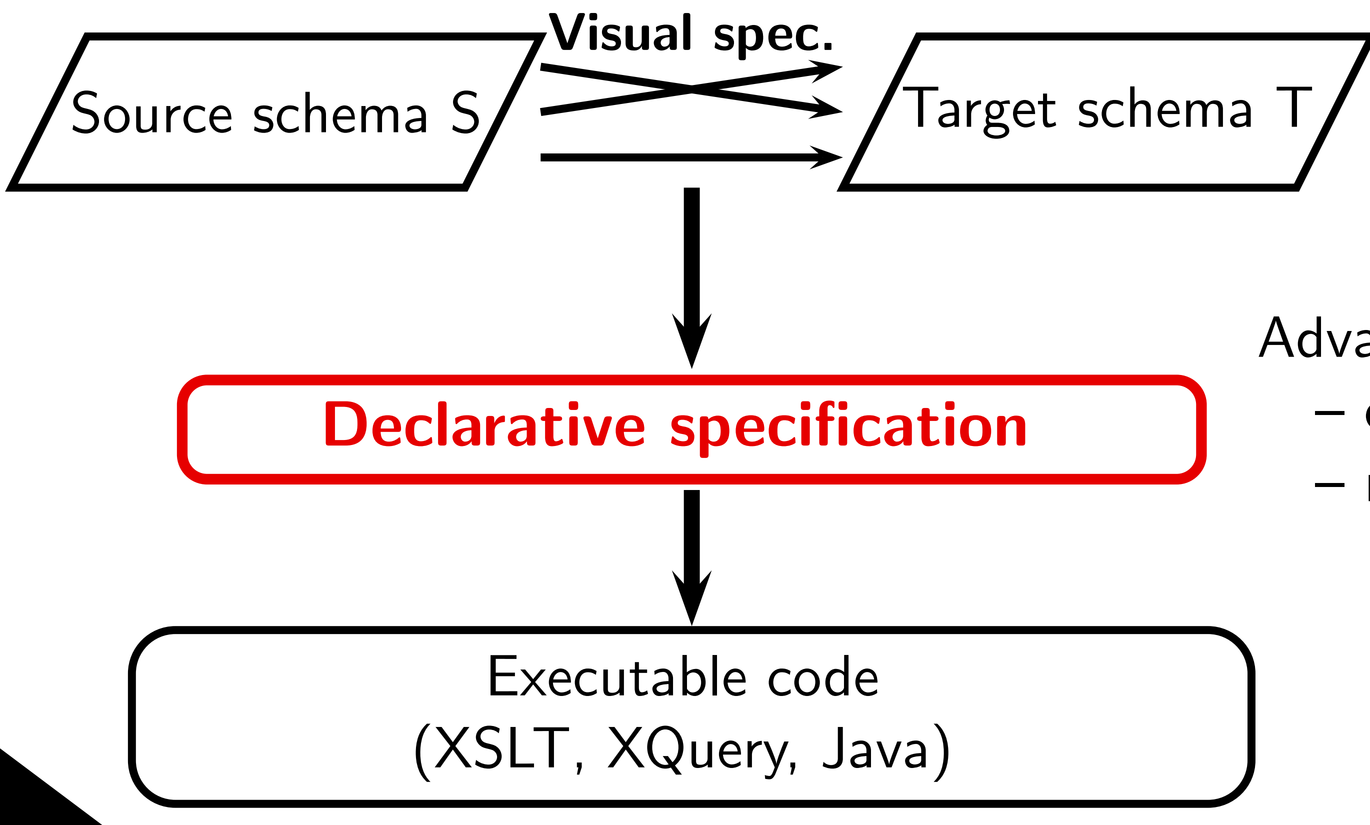
- Specifications are often impossible to understand through visual inspection
- Few tools are available to assist in understanding and designing alternative mappings

Designing Mappings

- Specifications are often impossible to understand through visual inspection
- Few tools are available to assist in understanding and designing alternative mappings
- MUSE is a tool designed towards this end

Designing Mappings

- Specifications are often impossible to understand through visual inspection
- Few tools are available to assist in understanding and designing alternative mappings
- MUSE is a tool designed towards this end
- In MUSE, we focus on declarative specifications



Advantages:

- easier to reason about
- reusable for various tasks

Our vision

- MUSE is a mapping design wizard that uses (real) data examples to help designers understand, design and refine schema mappings

- MUSE leverages familiar data examples to help understand mappings
  - real data examples are used whenever possible
  - otherwise, synthetic examples are constructed

Our vision

- MUSE is a mapping design wizard that uses (real) data examples to help designers understand, design and refine schema mappings

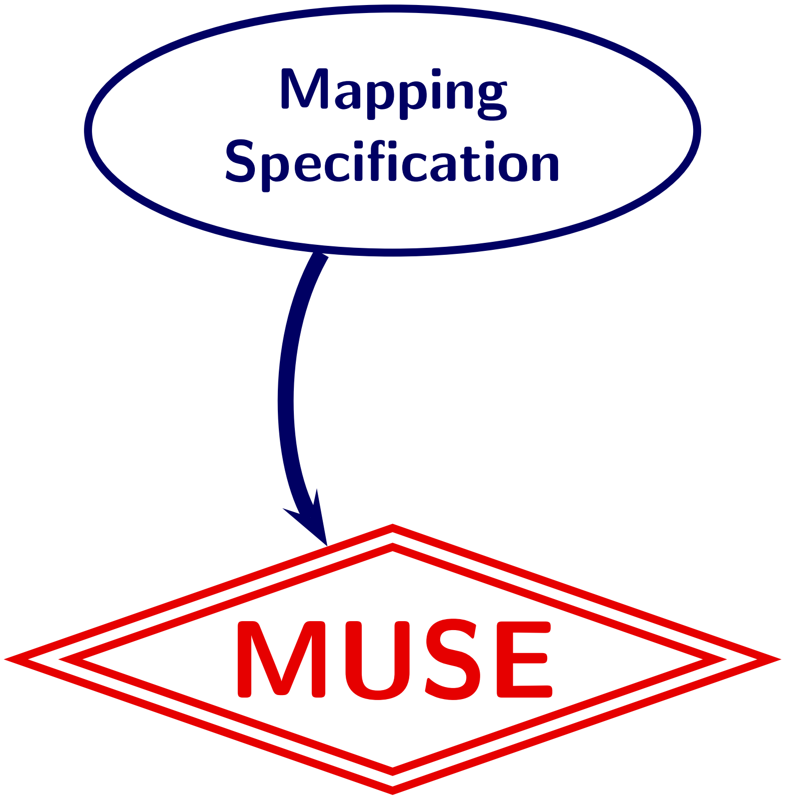
- MUSE leverages familiar data examples to help understand mappings
  - real data examples are used whenever possible
  - otherwise, synthetic examples are constructed

- Currently, MUSE has two features
  - Muse-G: design grouping semantics
  - Muse-D: disambiguate alternative mappings

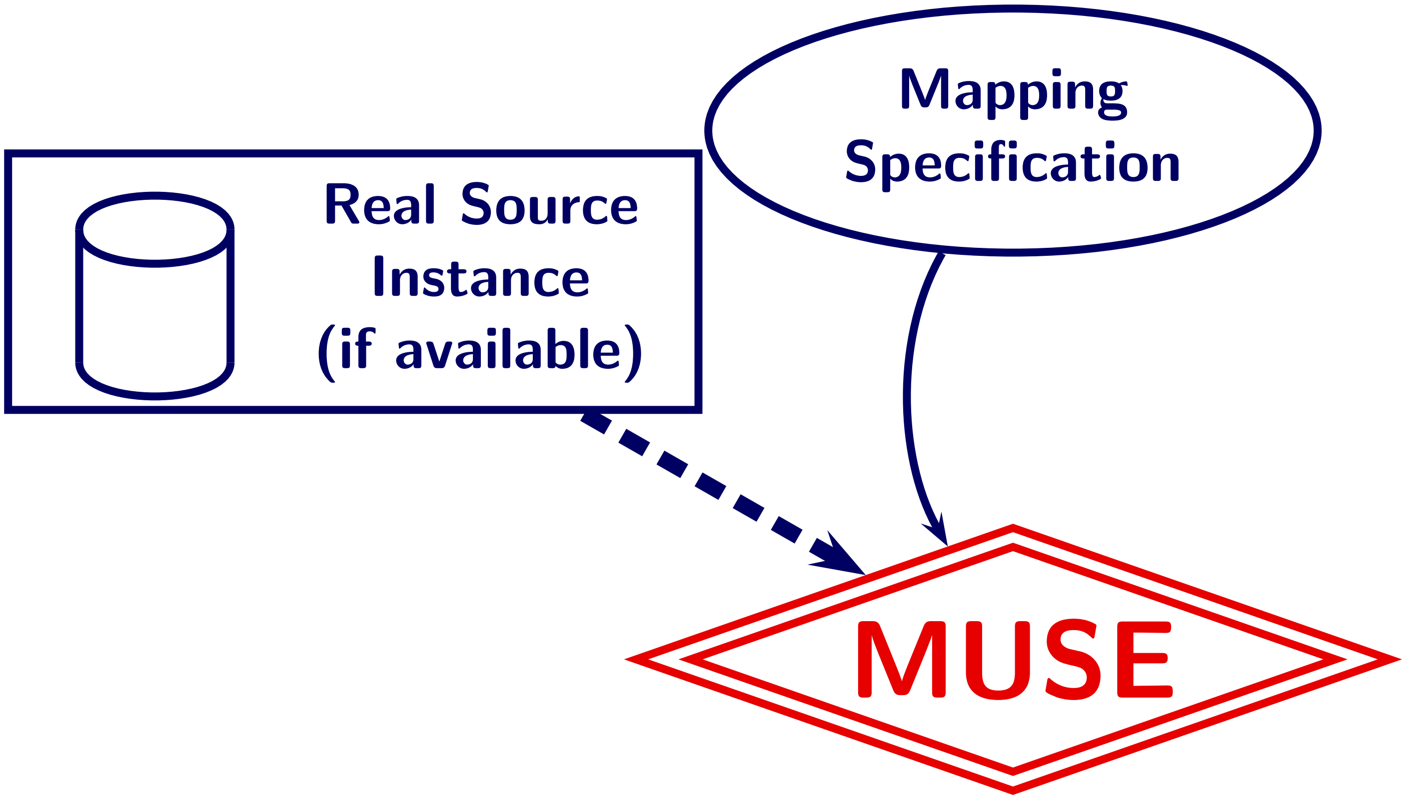
MUSE Workflow

MUSE

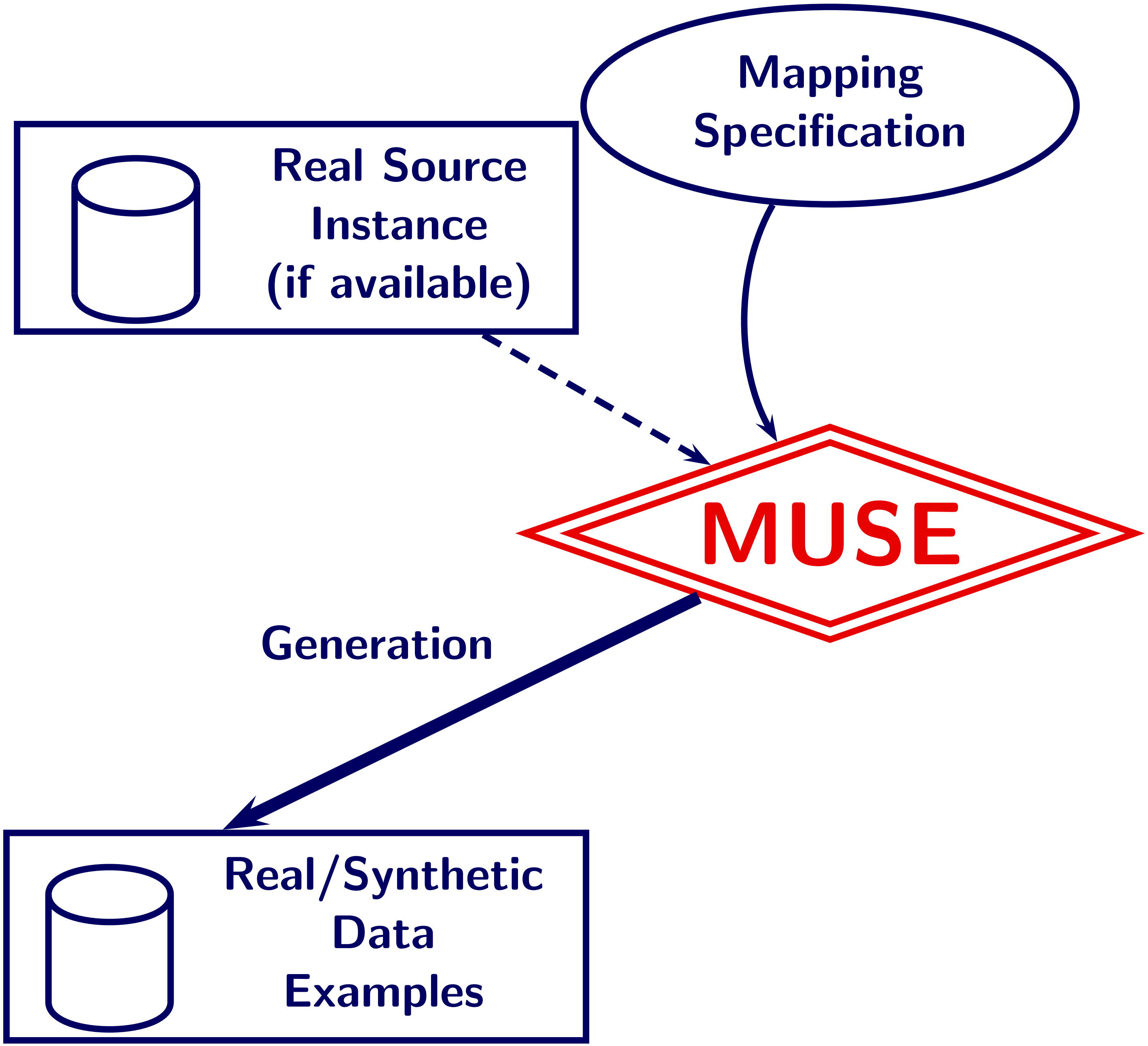
MUSE Workflow



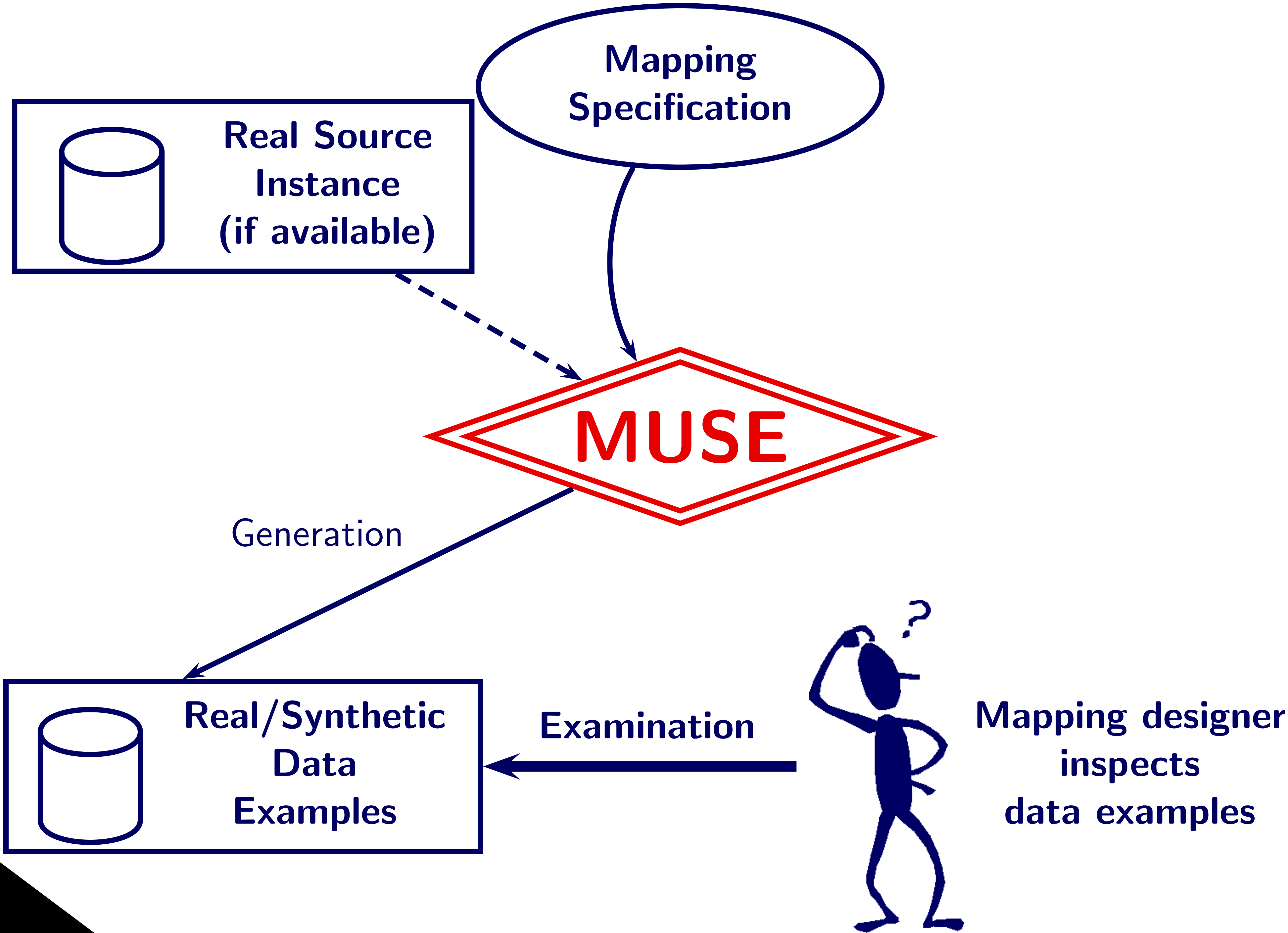
MUSE Workflow



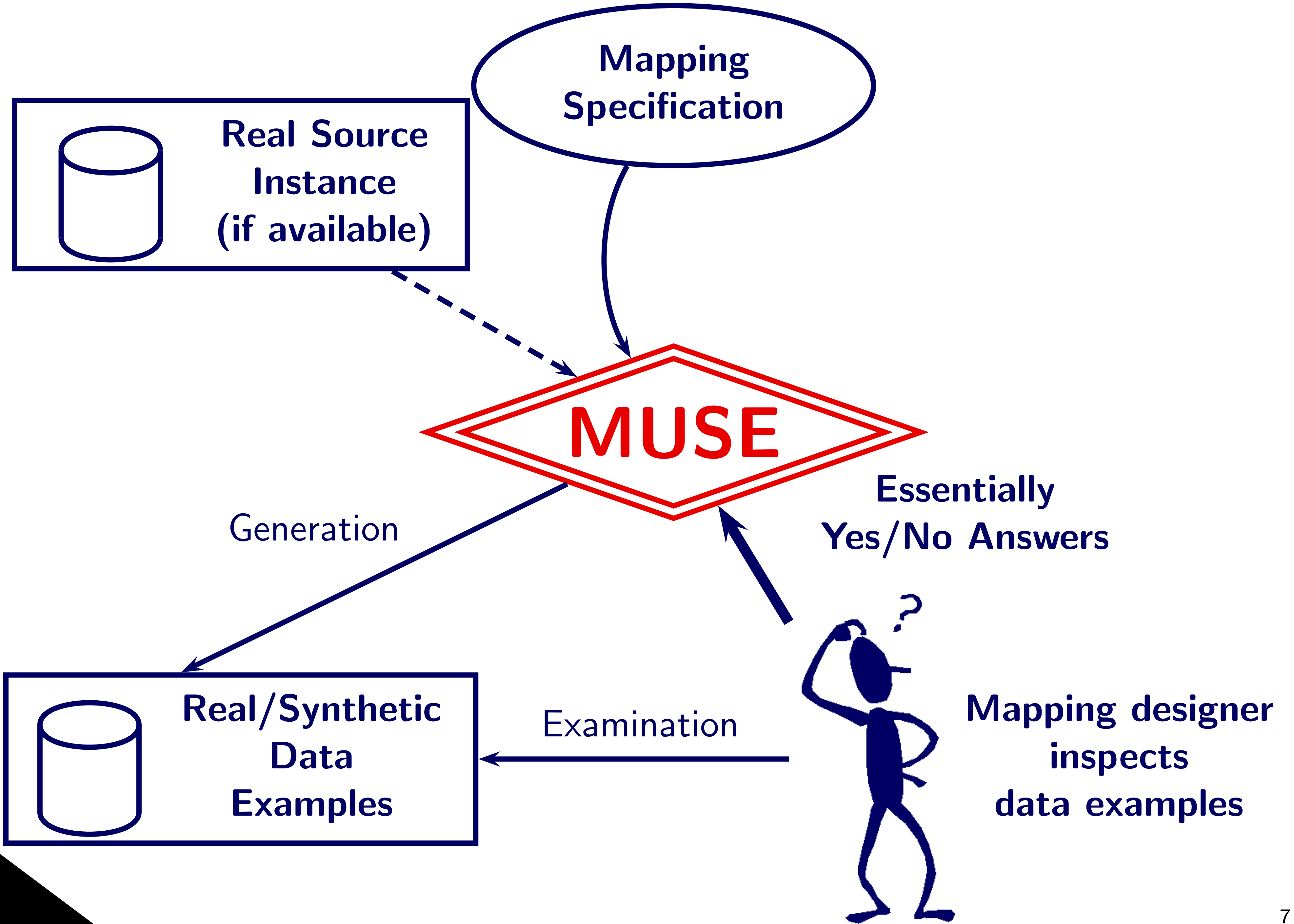
MUSE Workflow



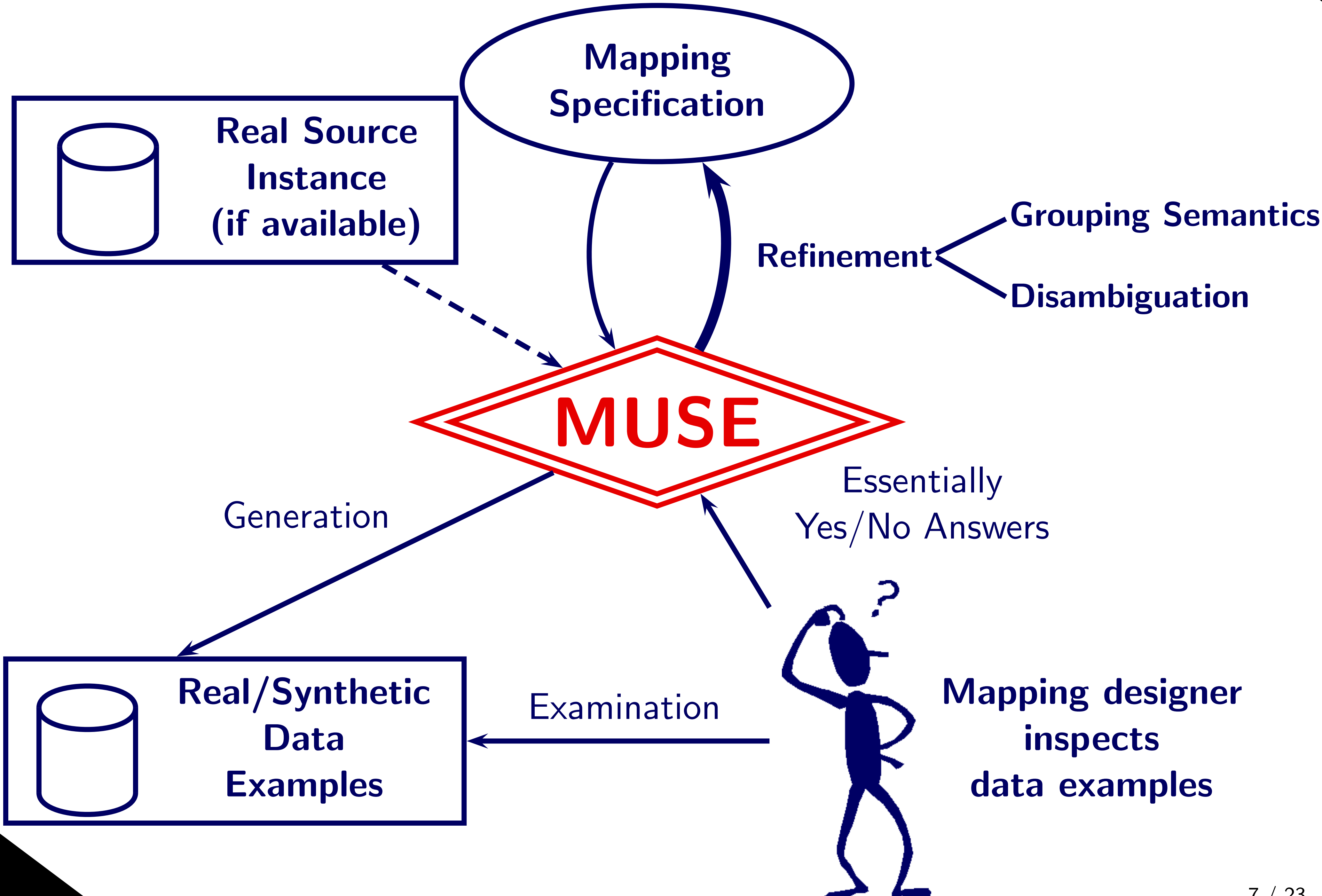
MUSE Workflow



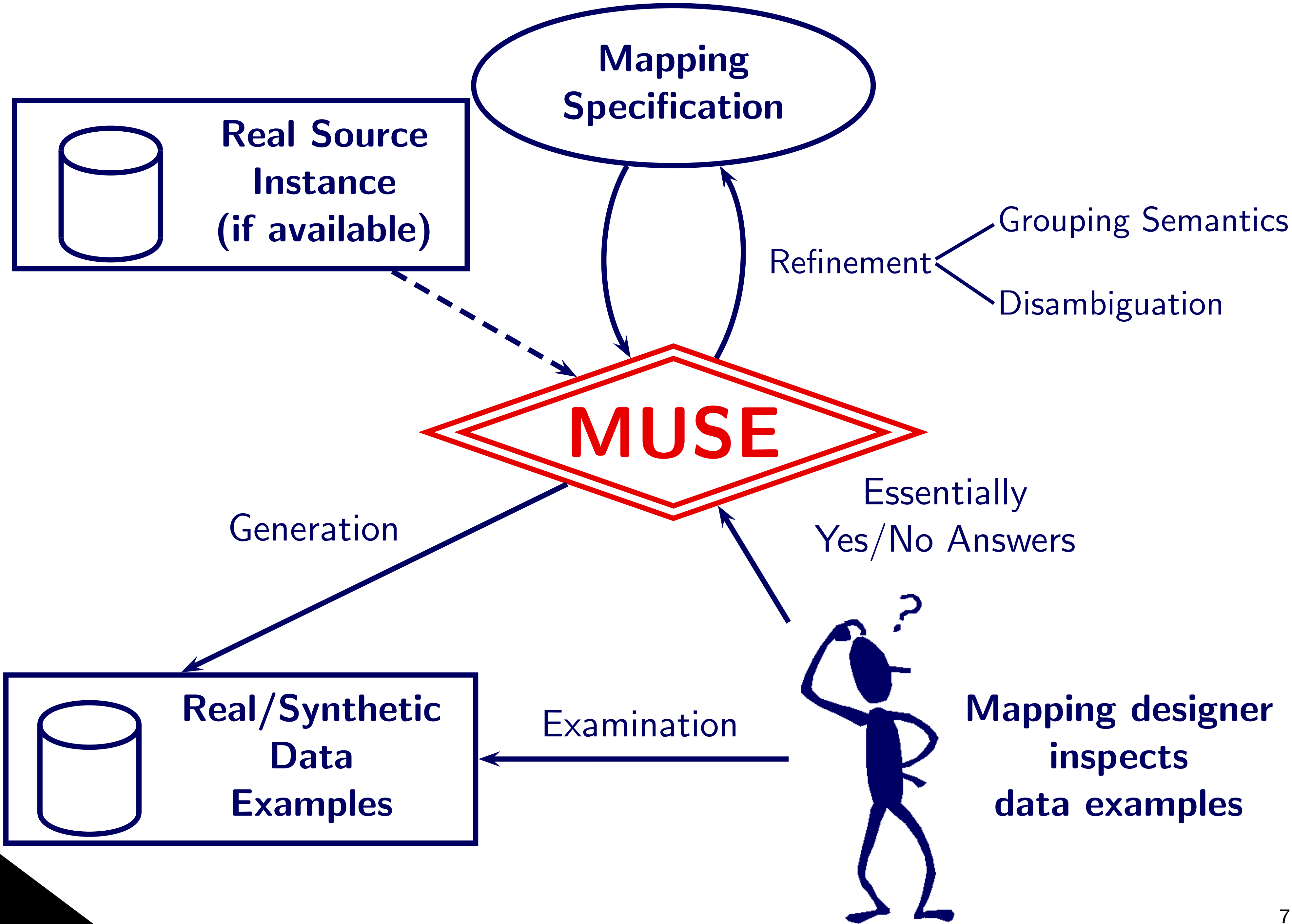
MUSE Workflow



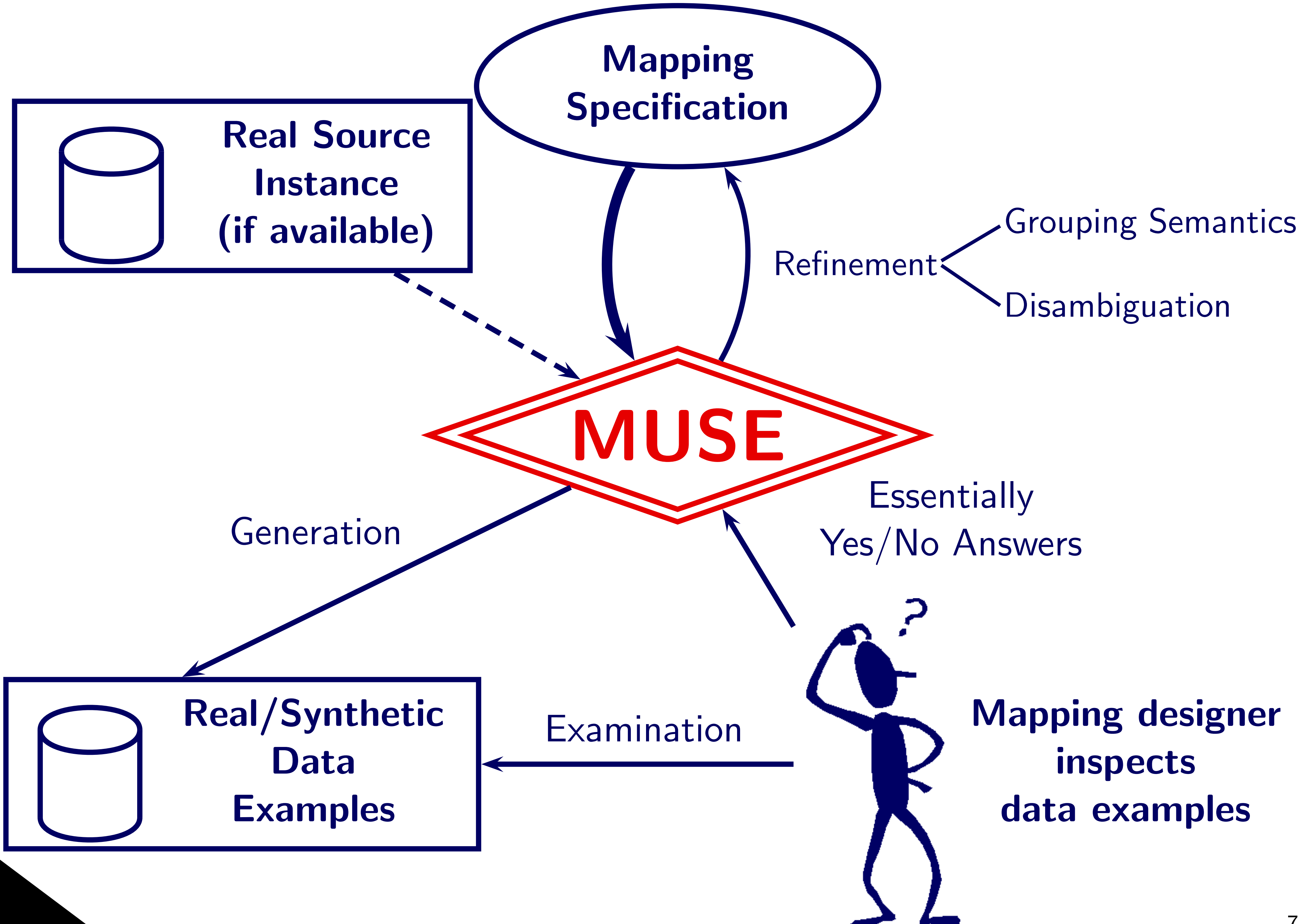
MUSE Workflow



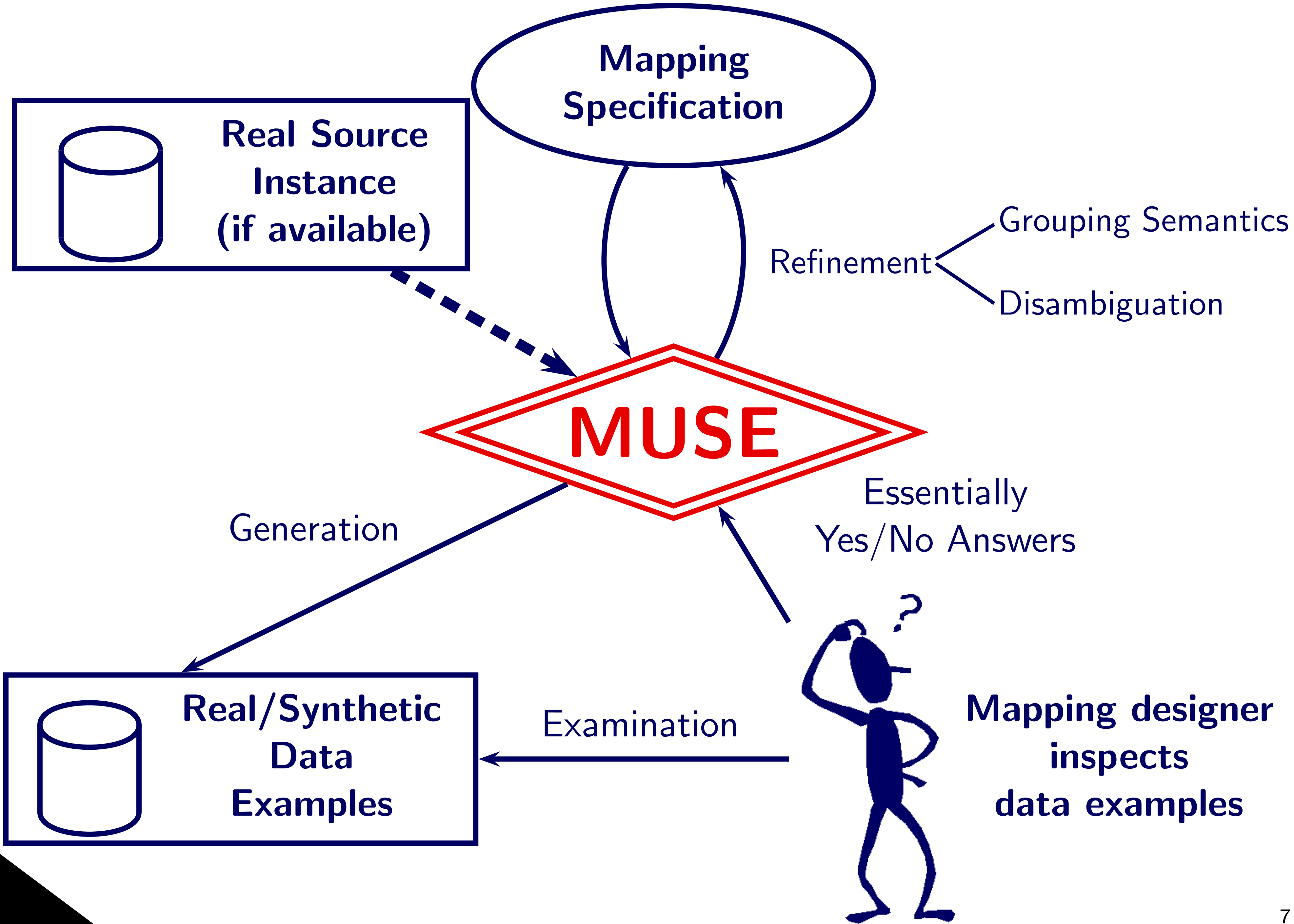
MUSE Workflow



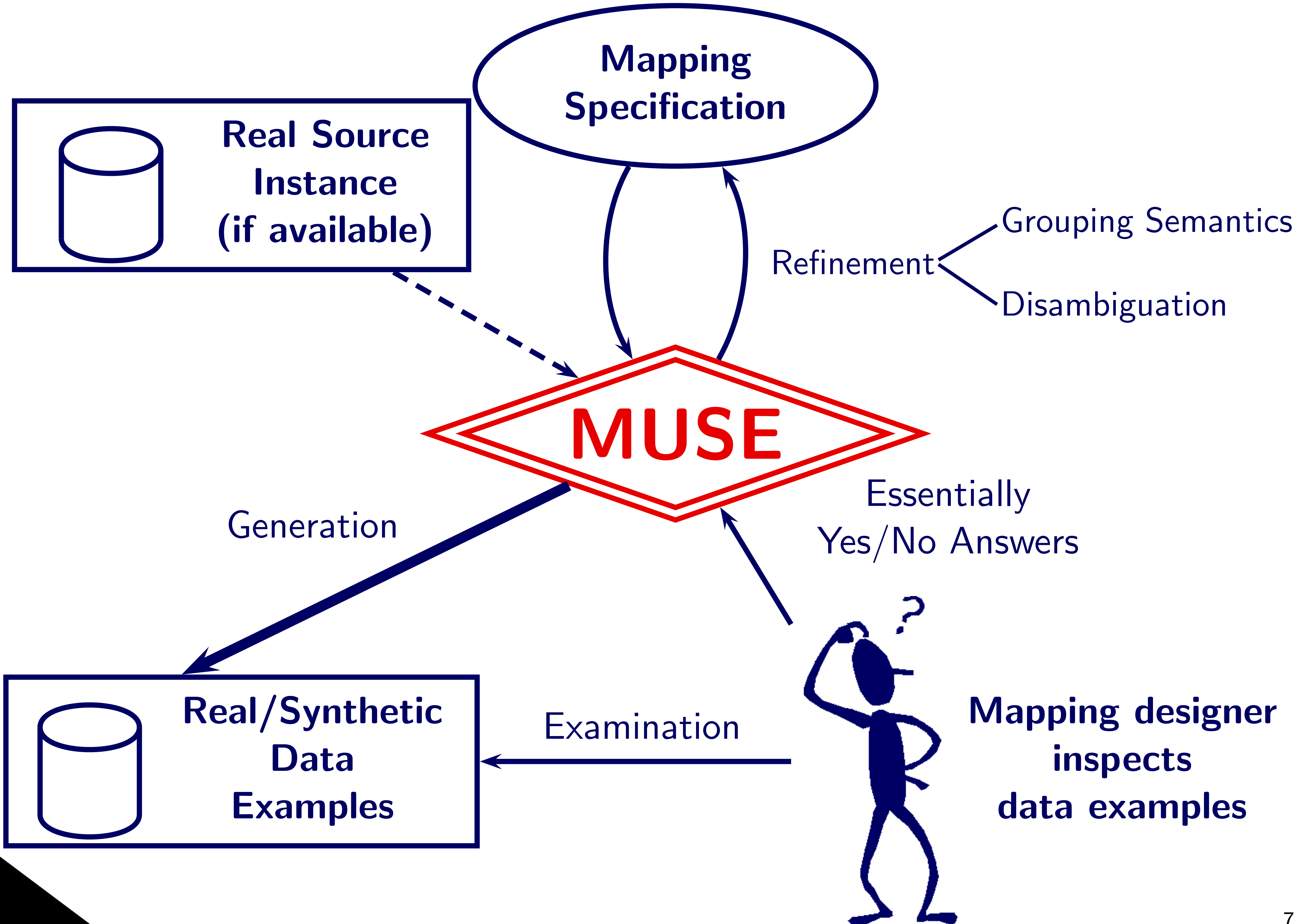
MUSE Workflow



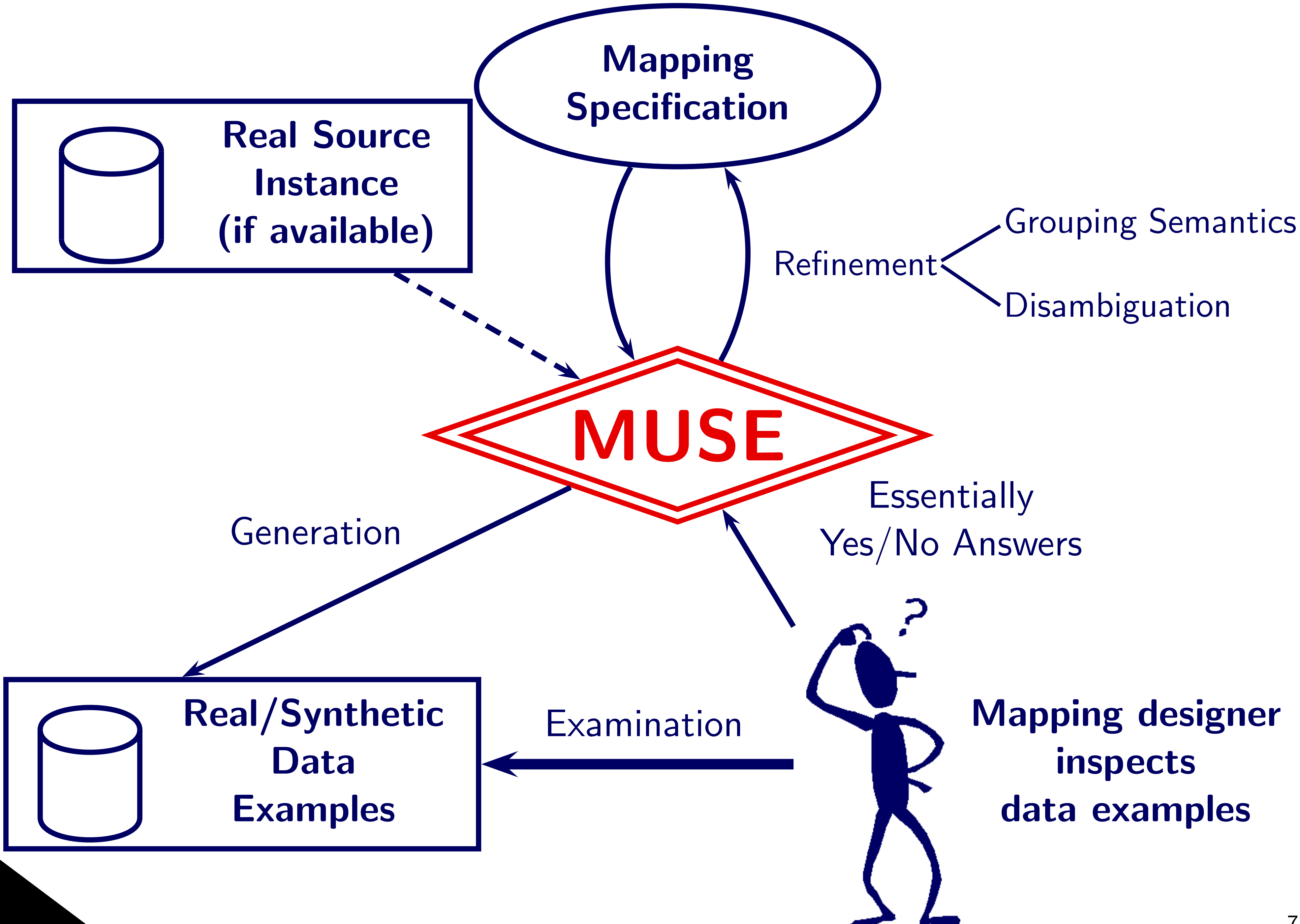
MUSE Workflow



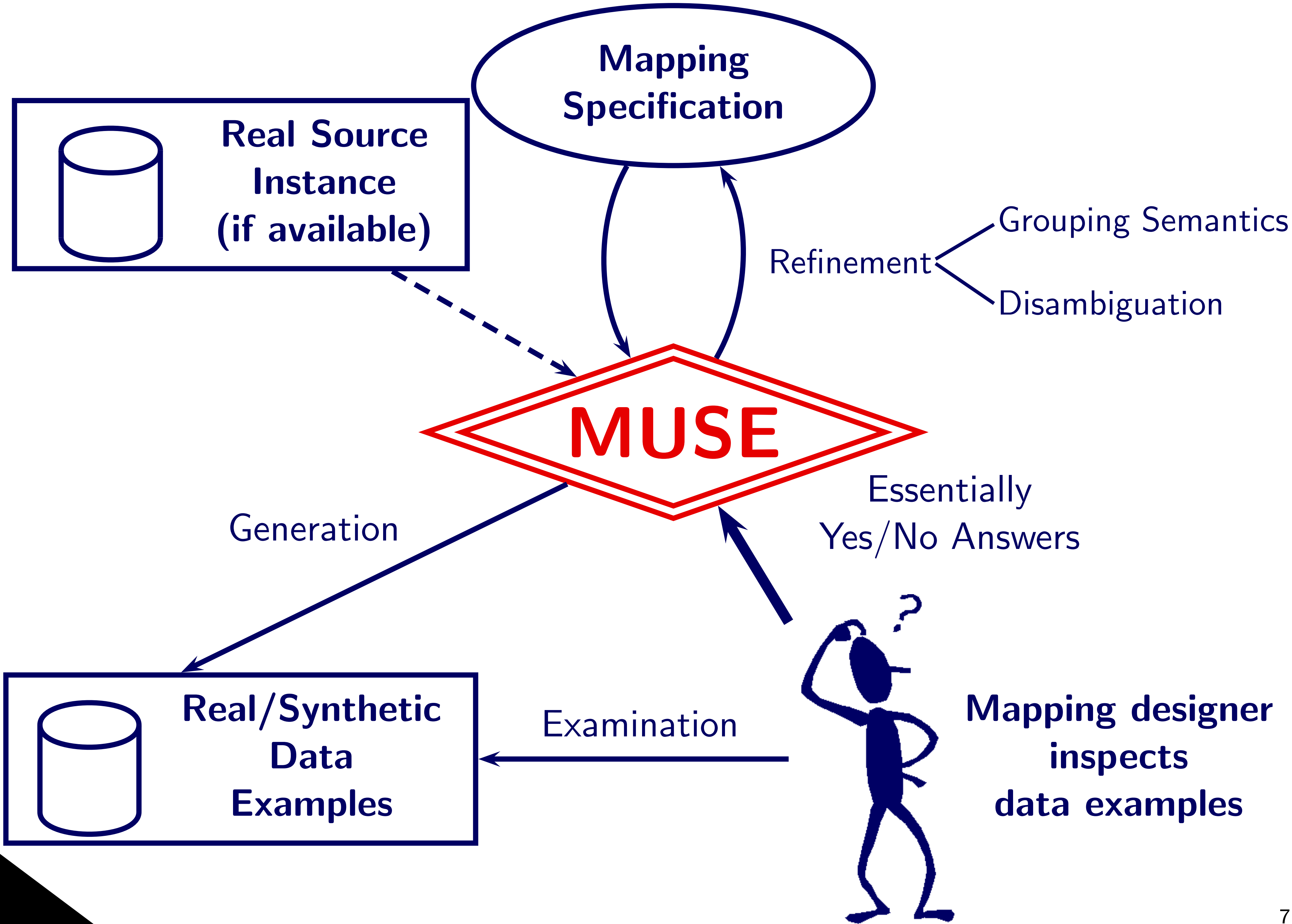
MUSE Workflow



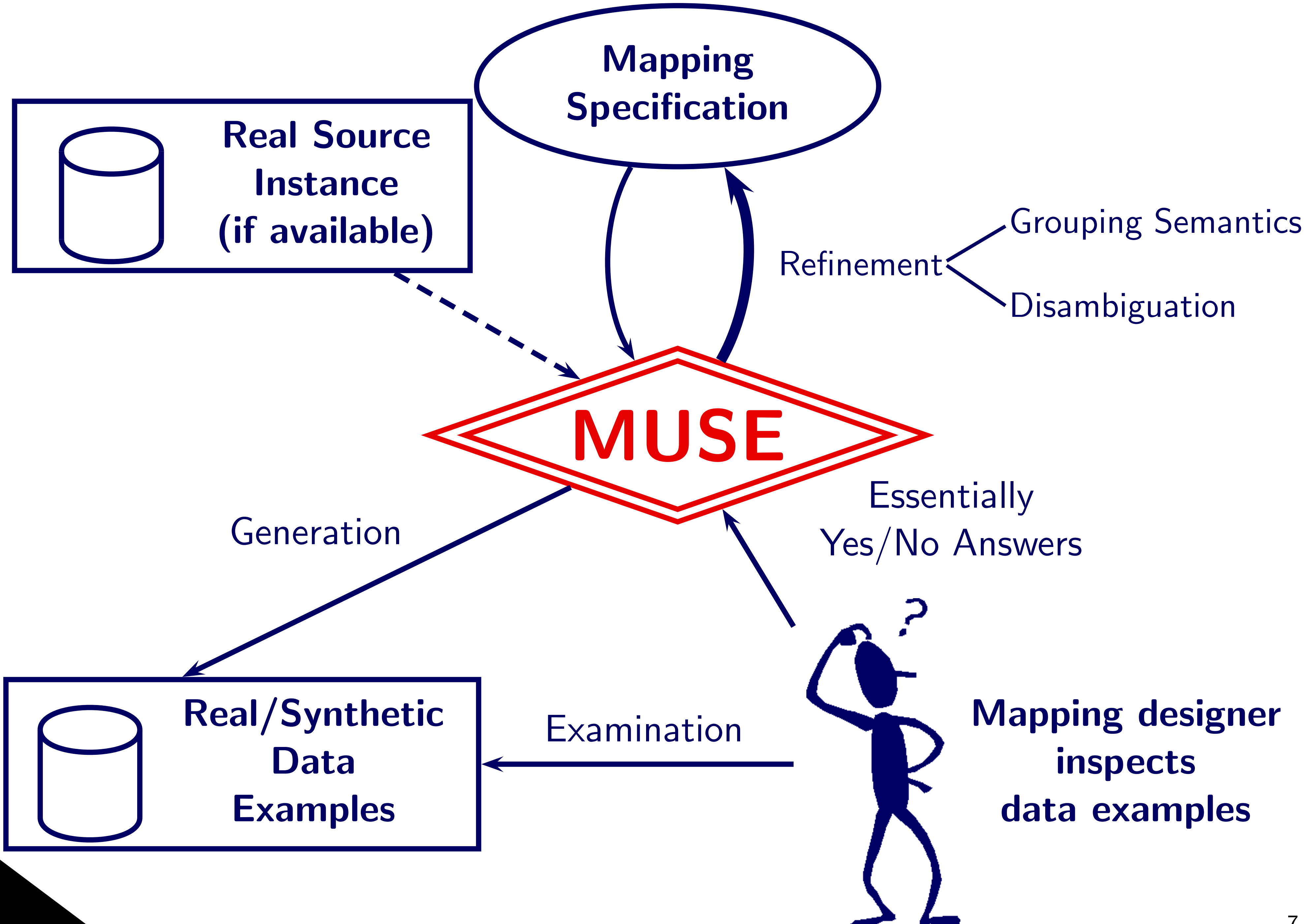
MUSE Workflow



MUSE Workflow



MUSE Workflow



Example

CompDB: *Rcd*

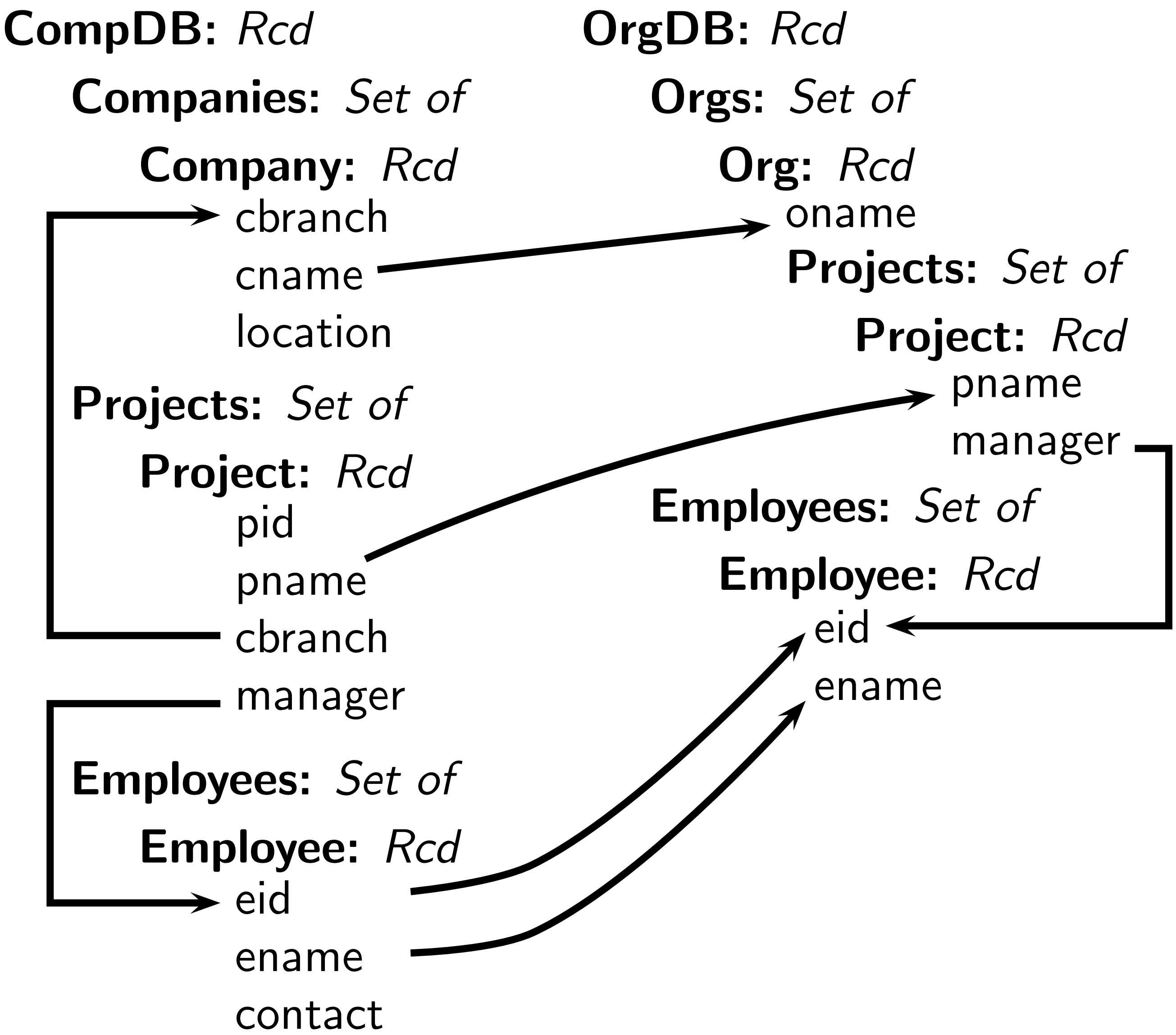
- **Companies:** *Set of*
  - **Company:** *Rcd*
    - cbranch
    - cname
    - location
- **Projects:** *Set of*
  - **Project:** *Rcd*
    - pid
    - pname
    - cbranch
    - manager
- **Employees:** *Set of*
  - **Employee:** *Rcd*
    - eid
    - ename
    - contact

Example

- **CompDB:** *Rcd*
  - **Companies:** *Set of*
    - **Company:** *Rcd*
      - cbranch
      - cname
      - location
  - **Projects:** *Set of*
    - **Project:** *Rcd*
      - pid
      - pname
      - cbranch
      - manager
  - **Employees:** *Set of*
    - **Employee:** *Rcd*
      - eid
      - ename
      - contact

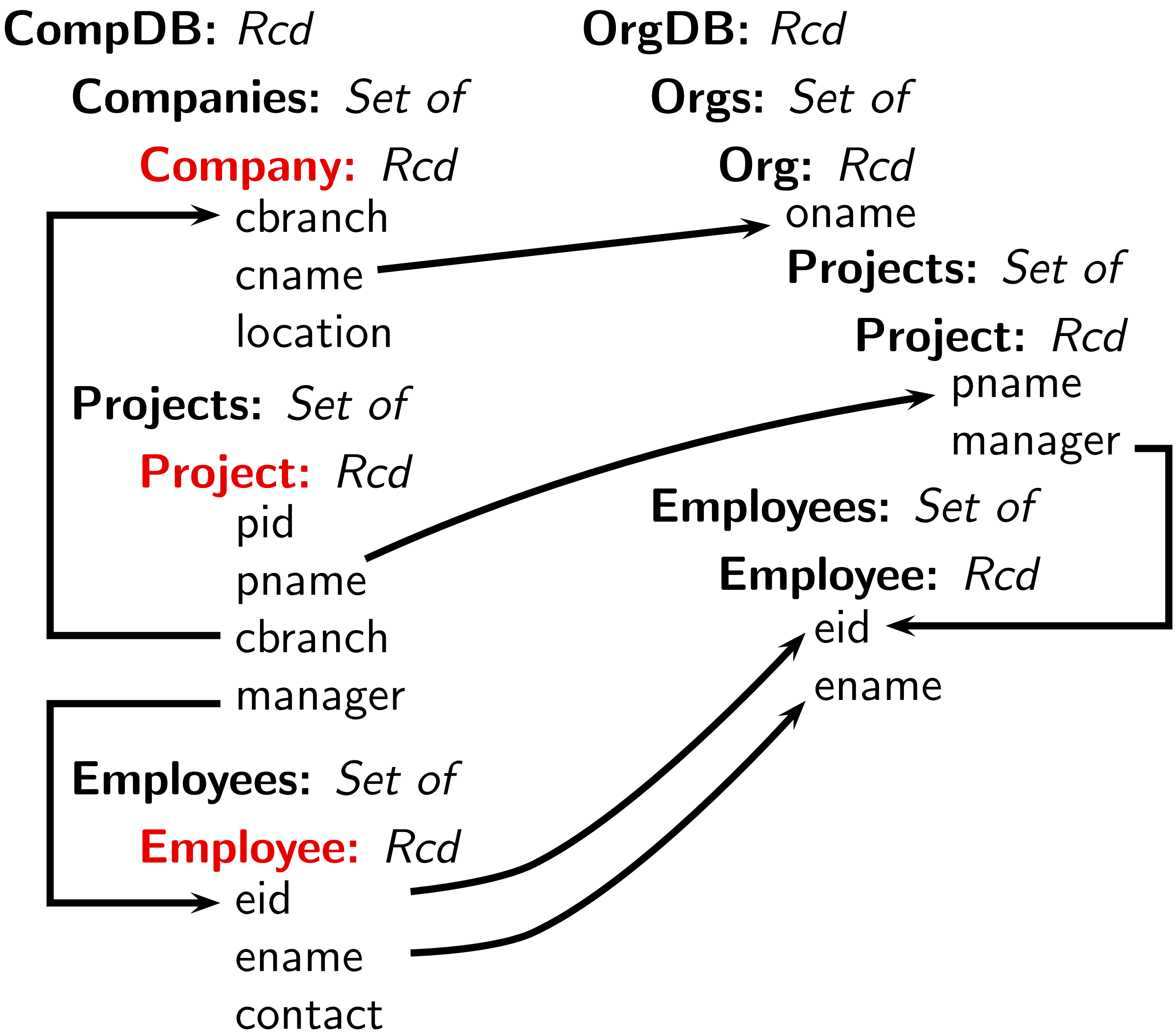
- **OrgDB:** *Rcd*
  - **Orgs:** *Set of*
    - **Org:** *Rcd*
      - oname
      - **Projects:** *Set of*
        - **Project:** *Rcd*
          - pname
          - manager
  - **Employees:** *Set of*
    - **Employee:** *Rcd*
      - eid
      - ename

Example



CompDB: Rcd
Companies: Set of
Company: Rcd
cbranch
cname
location
Projects: Set of
Project: Rcd
pid
pname
cbranch
manager
Employees: Set of
Employee: Rcd
eid
ename
contact
OrgDB: Rcd
Orgs: Set of
Org: Rcd
oname
Projects: Set of
Project: Rcd
pname
manager
Employees: Set of
Employee: Rcd
eid
ename

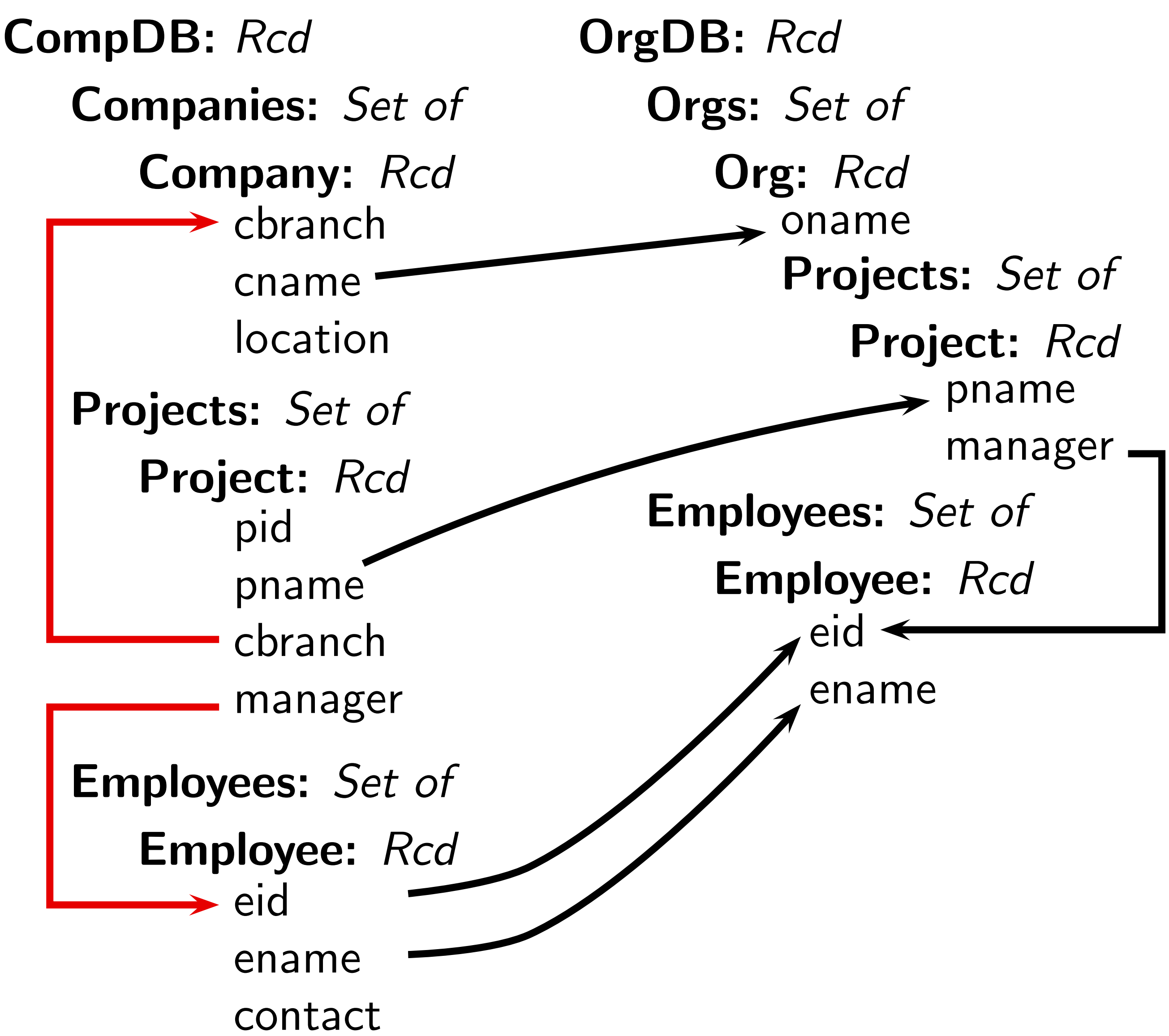
Example



Declarative Mapping

for
c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

Example



Declarative Mapping

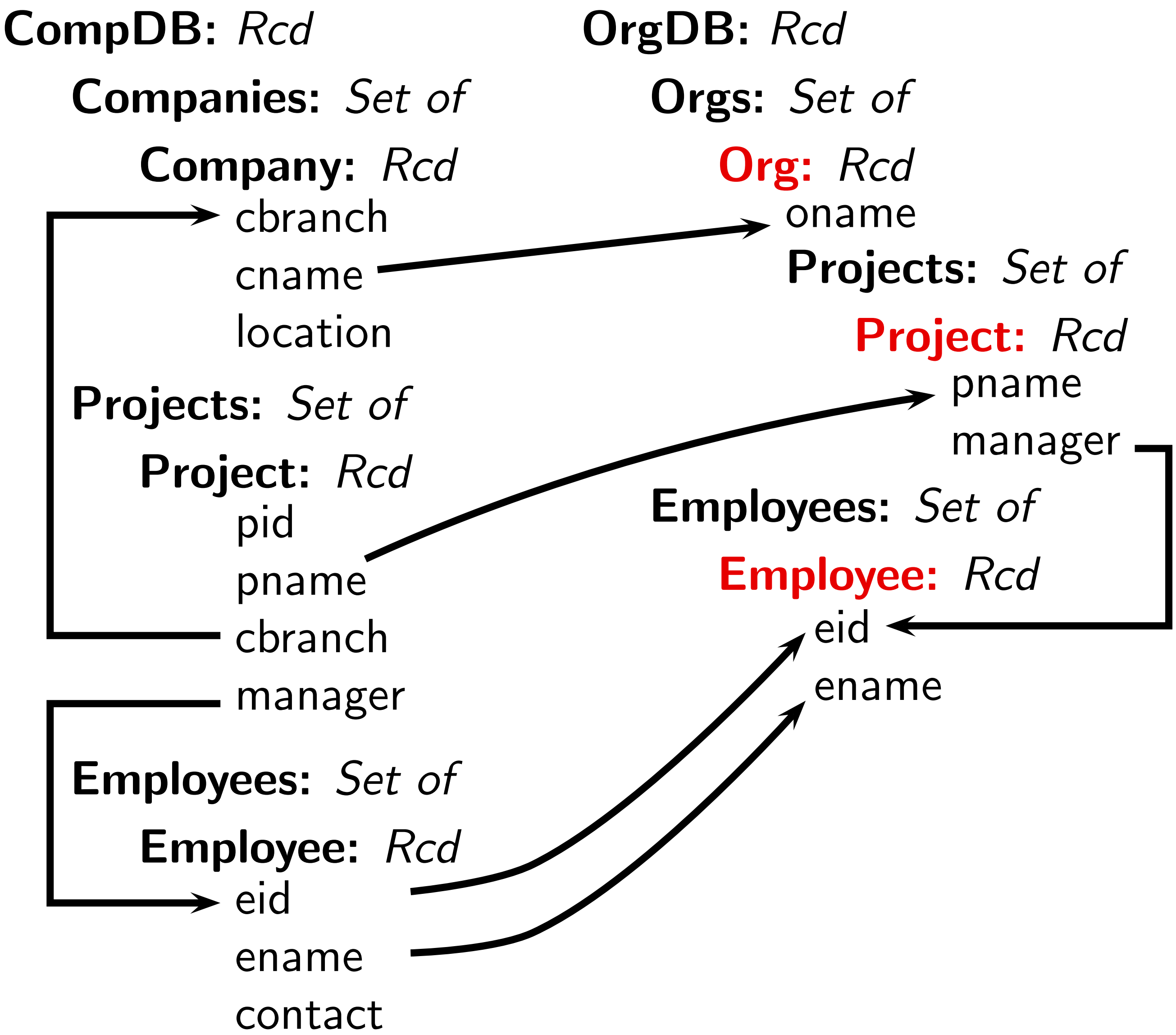
for

c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

satisfy

p.cbranch = c.cbranch
e.eid = p.manager

Example



Declarative Mapping

for

c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

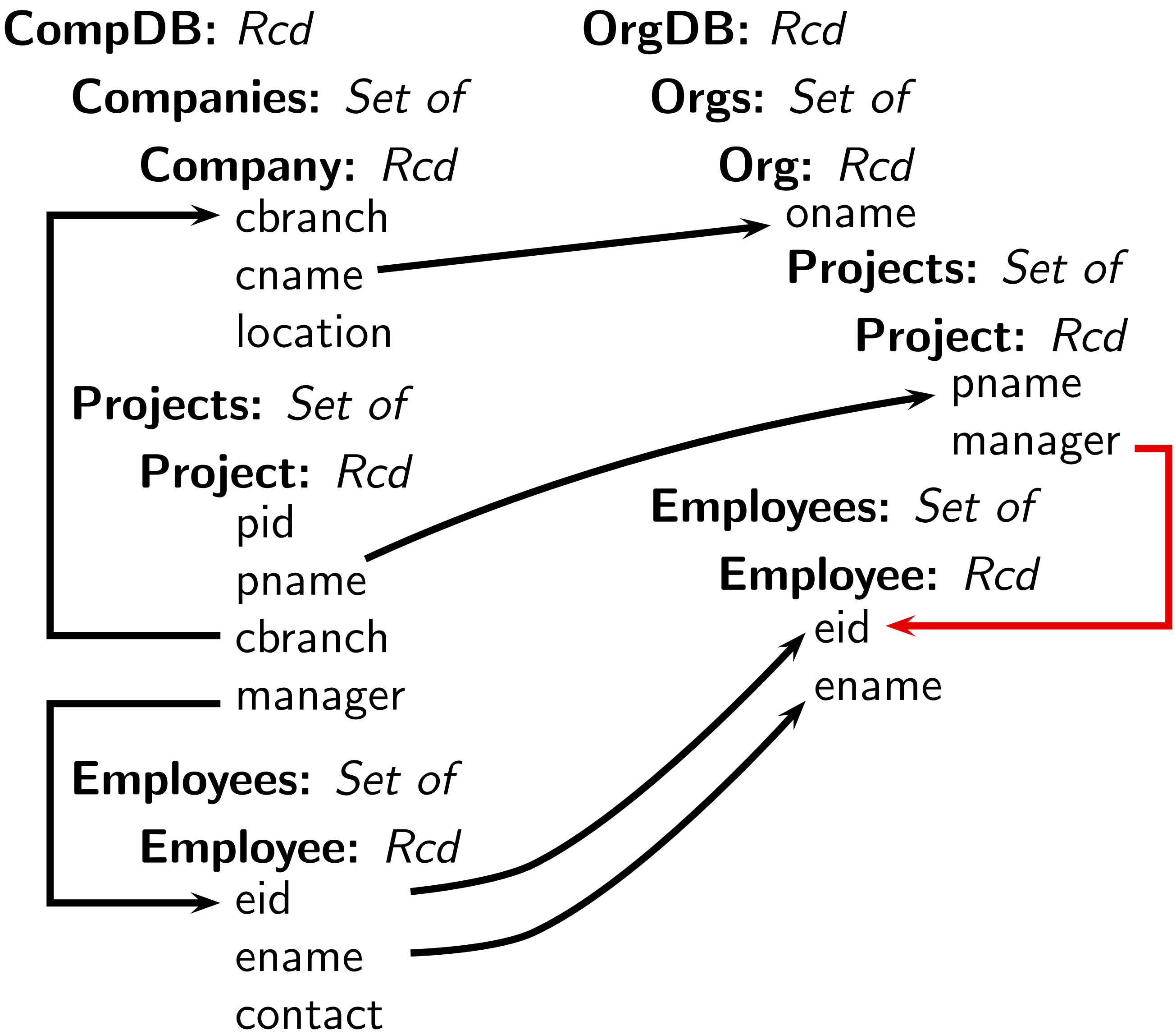
satisfy

p.cbranch = c.cbranch
e.eid = p.manager

exists

o in OrgDB.Orgs
p_1 in o.Projects
e_1 in OrgDB.Employees

Example



Declarative Mapping

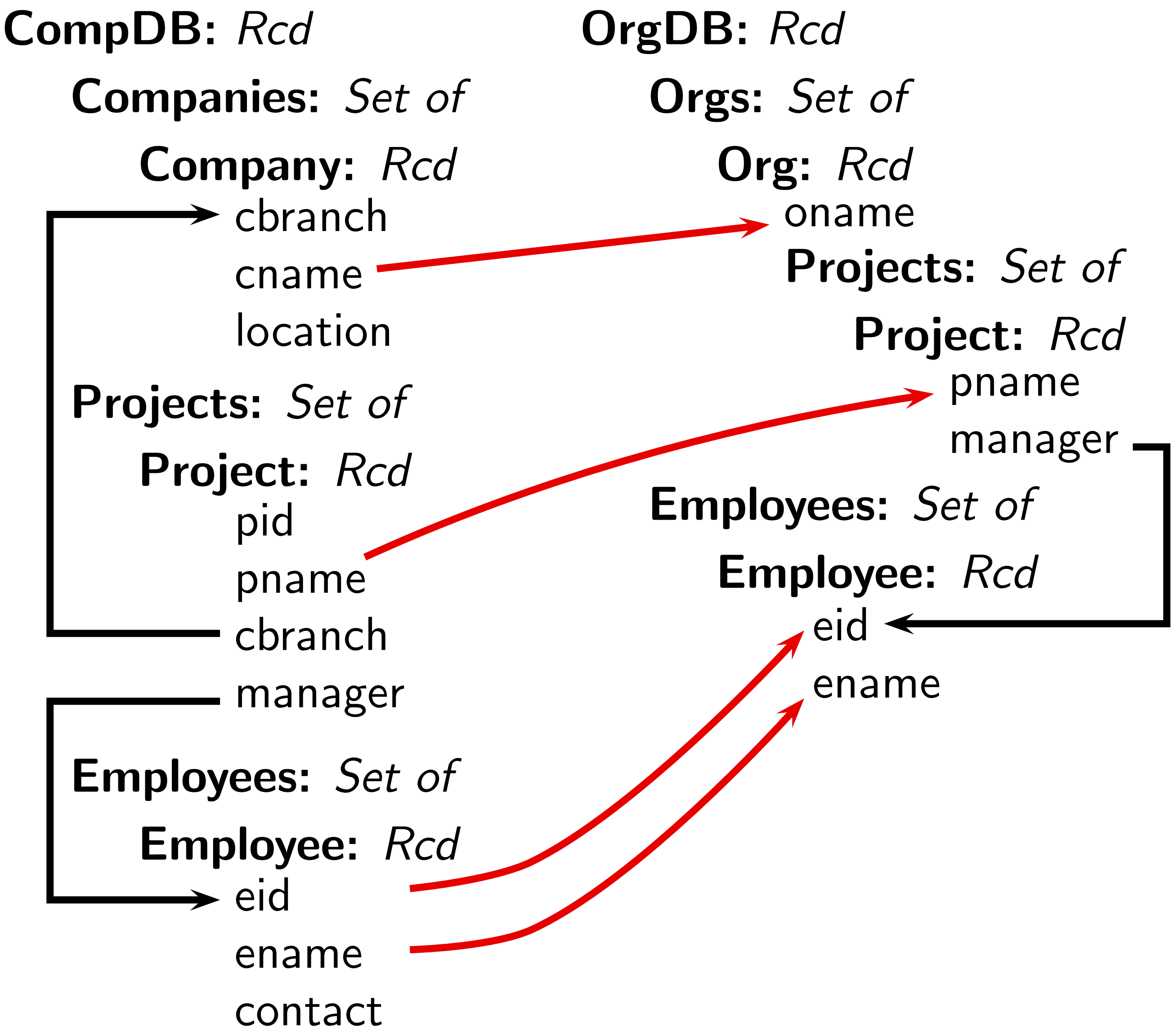
for
c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

satisfy
p.cbranch = c.cbranch
e.eid = p.manager

exists
o in OrgDB.Orgs
p_1 in o.Projects
e_1 in OrgDB.Employees

satisfy
p_1.manager = e_1.eid

Example



Declarative Mapping

for
c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

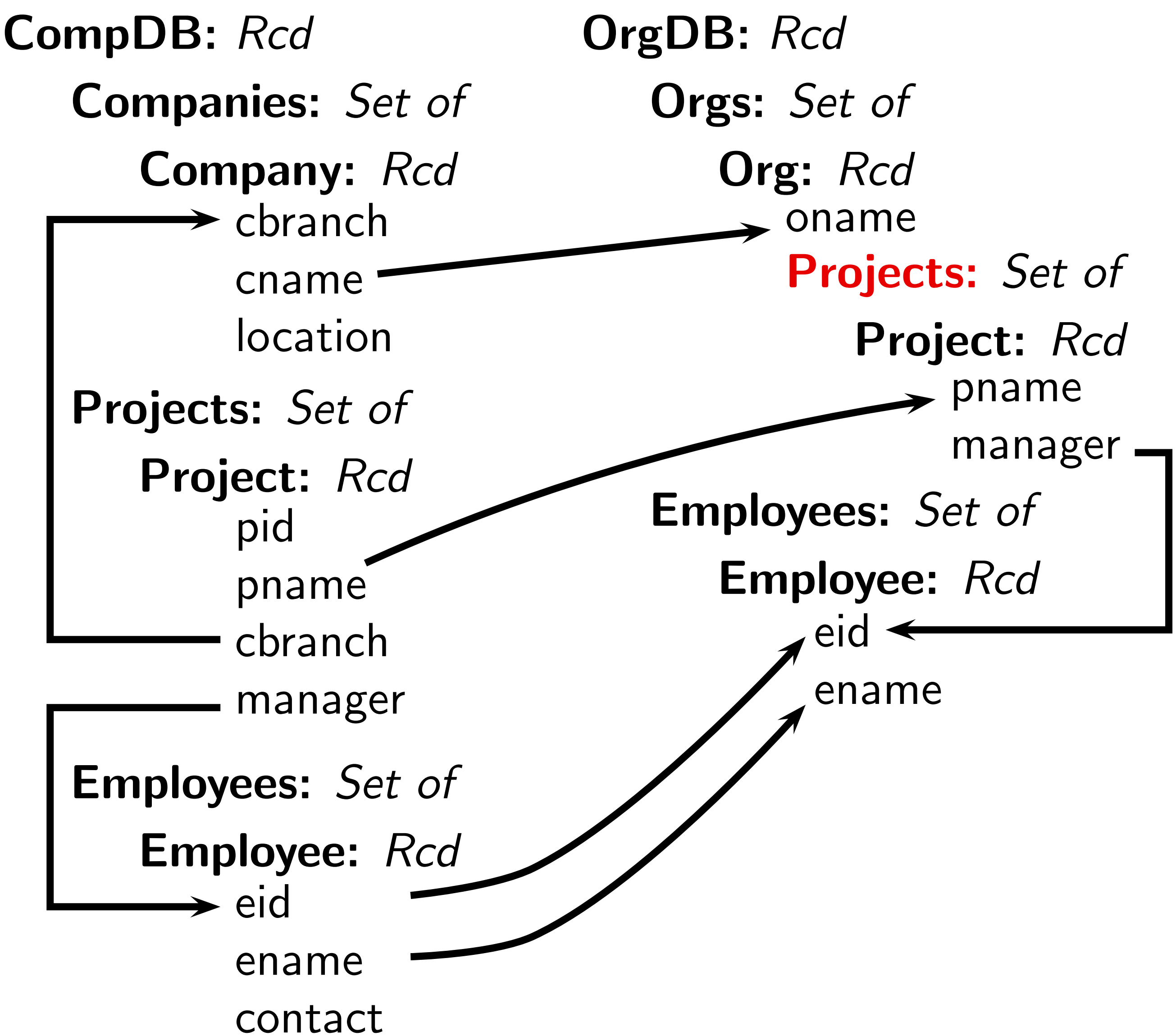
satisfy
p.cbranch = c.cbranch
e.eid = p.manager

exists
o in OrgDB.Orgs
p_1 in o.Projects
e_1 in OrgDB.Employees

satisfy
p_1.manager = e_1.eid

where
c.cname = o.oname
e.eid = e_1.eid
e.ename = e_1.ename
p.pname = p_1.pname

Example



Grouping Projects:

Example source:

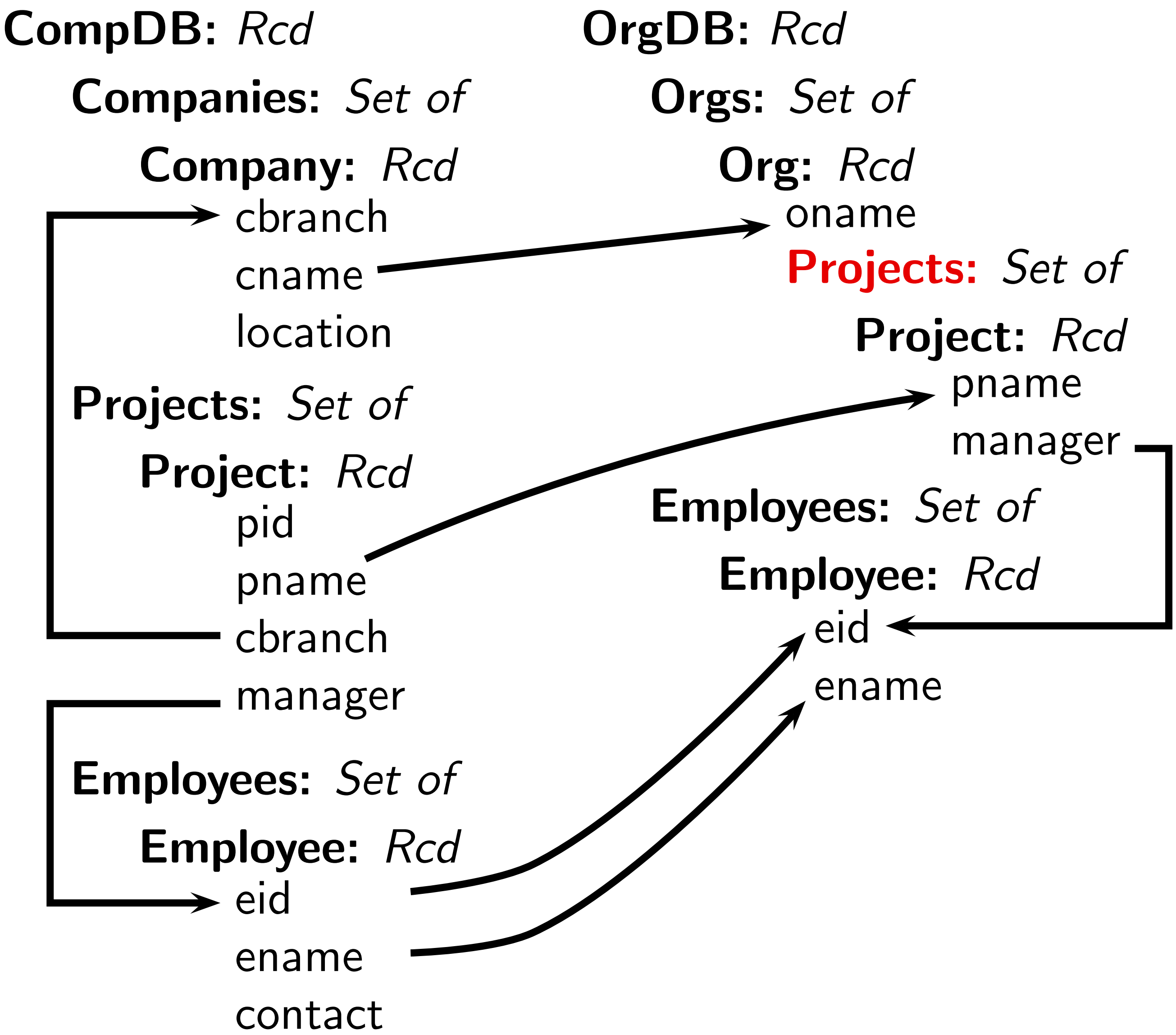
Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Example



Grouping Projects:

Example source:

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Group by **cbranch**

Orgs

Microsoft
Projects:
DB e4

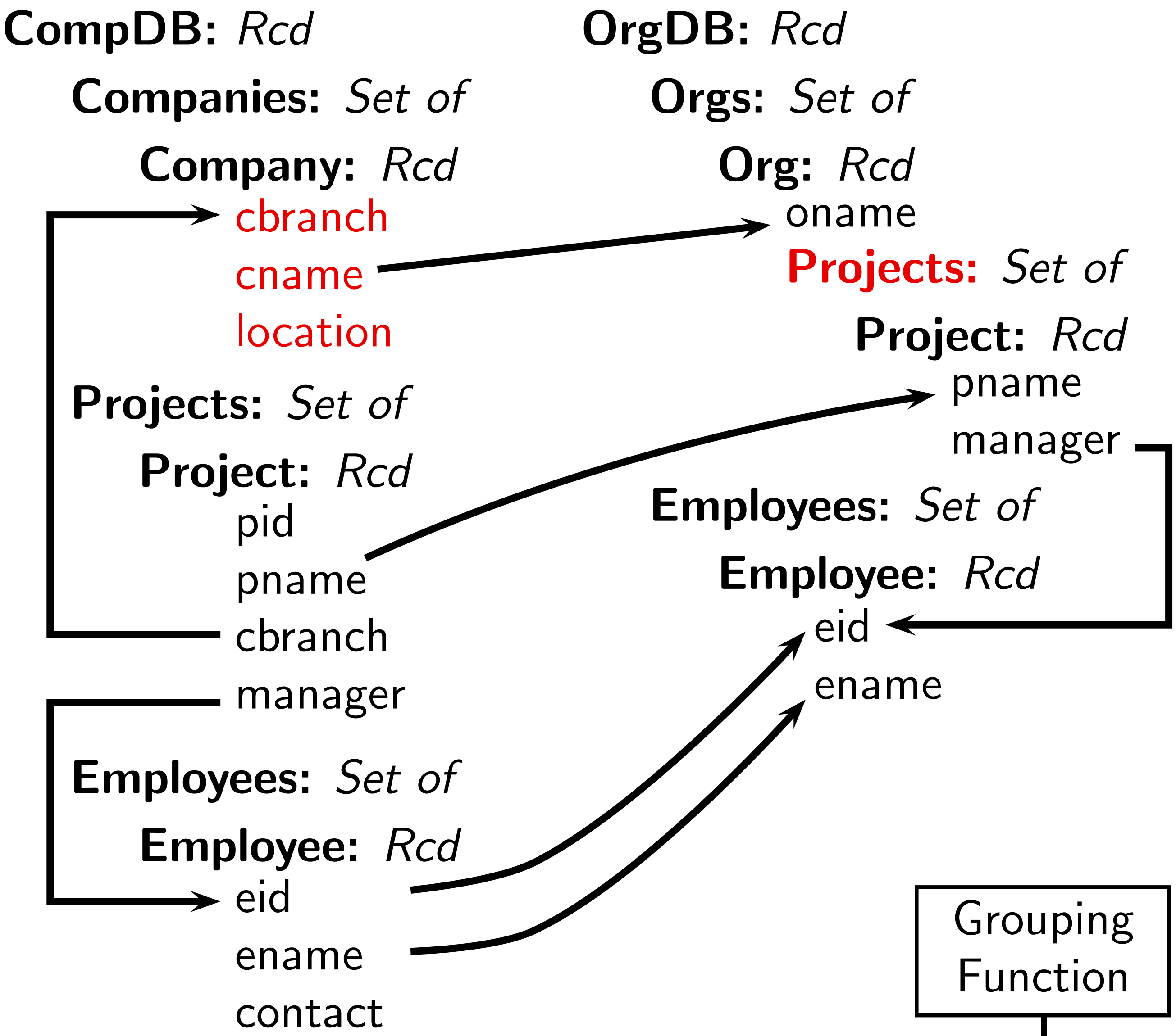
Microsoft
Projects:
Web e5

Group by **cname**

Orgs

Microsoft
Projects:
DB e4
Web e5

Example



Declarative Mapping

for

c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

satisfy

p.cbranch = c.cbranch
e.eid = p.manager

exists

o in OrgDB.Orgs
p_1 in o.Projects
e_1 in OrgDB.Employees

satisfy

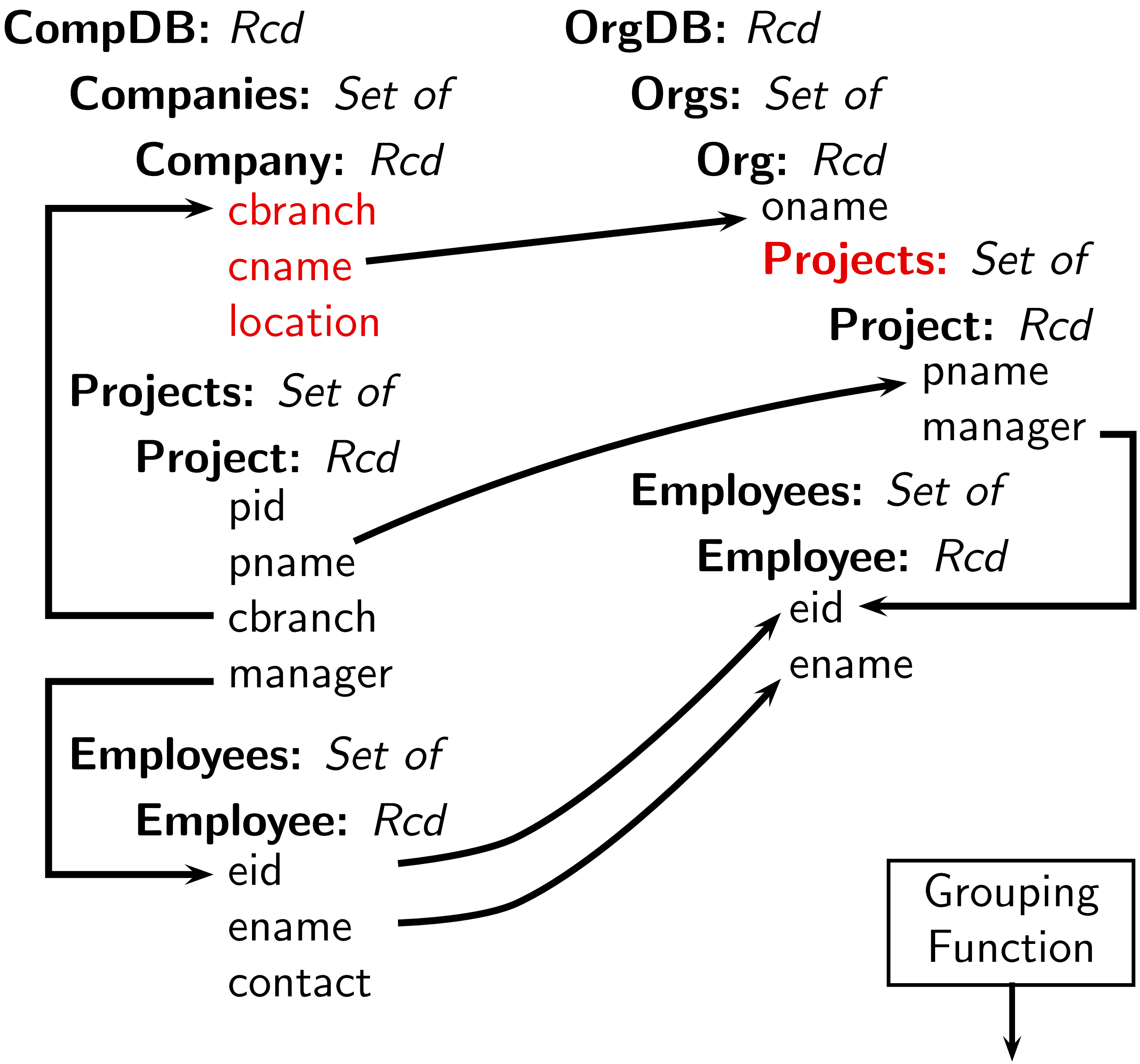
p_1.manager = e_1.eid

where

c.cname = o.oname
e.eid = e_1.eid
e.ename = e_1.ename
p.pname = p_1.pname

o.Projects = SKProjects(c.cbranch, c.cname, c.location)

Example



Declarative Mapping

for
c in CompDB.Companies
p in CompDB.Projects
e in CompDB.Employees

satisfy
p.cbranch = c.cbranch
e.eid = p.manager

exists
o in OrgDB.Orgs
p_1 in o.Projects
e_1 in OrgDB.Employees

satisfy
p_1.manager = e_1.eid

where
c.cname = o.oname
e.eid = e_1.eid
e.ename = e_1.ename
p.pname = p_1.pname

o.Projects = SKProjects(c.cbranch, c.cname, c.location)

Group by what subset of {cbranch, cname, location} ?

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Example source

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Employees

e4	John	x234
e5	Anna	x888

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Example source

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Employees

e4	John	x234
e5	Anna	x888

Target Scenario 1
group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4

Microsoft
Projects:

Web	e5

Employees

e4	John
e5	Anna

Target Scenario 2
do not group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4
Web	e5

Employees

e4	John
e5	Anna

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Example source

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Employees

e4	John	x234
e5	Anna	x888

Target Scenario 1

group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4

Microsoft
Projects:

Web	e5

Employees

e4	John
e5	Anna

Target Scenario 2

do not group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4
Web	e5

Employees

e4	John
e5	Anna

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Example source

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Employees

e4	John	x234
e5	Anna	x888

Target Scenario 1
group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4

Microsoft
Projects:

Web	e5

Employees

e4	John
e5	Anna

Target Scenario 2
do not group
by **cbranch**

Orgs

Microsoft
Projects:

DB	e4
Web	e5

Employees

e4	John
e5	Anna

Muse-G: Grouping Semantics Design

- **Goal:** infer a grouping function that has the same effect as the one intended by the designer
- Muse-G probes each possible grouping attribute: start with cbranch

Example source

Companies

Redmond	Microsoft	USA
S. Valley	Microsoft	USA

Projects

P1	DB	Redmond	e4
P2	Web	S. Valley	e5

Employees

e4	John	x234
e5	Anna	x888

Target Scenario 1
group by **cbranch**

Orgs

Microsoft
Projects: SK(Redmond,y)

DB	e4

Microsoft
Projects: SK(S. Valley,y)

Web	e5

Employees

e4	John
e5	Anna

Target Scenario 2
do not group by **cbranch**

Orgs

Microsoft
Projects: SK(y)

DB	e4
Web	e5

Employees

e4	John
e5	Anna

$y \subseteq \{ \text{Microsoft, USA} \}$

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Target Scenario 1
group by **cname**

Orgs

Microsoft
Projects:

DB	e4

Google
Projects:

Web	e6

Employees

e4	John
e6	Kat

Target Scenario 2
do not group by **cname**

Orgs

Microsoft
Projects:

DB	e4
Web	e6

Google
Projects:

DB	e4
Web	e6

Employees

e4	John
e6	Kat

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Target Scenario 1
group
by **cname**

Orgs

Microsoft
Projects:

DB	e4

Google
Projects:

Web	e6

Employees

e4	John
e6	Kat

Target Scenario 2
do not group
by **cname**

Orgs

Microsoft
Projects:

DB	e4
Web	e6

Google
Projects:

DB	e4
Web	e6

Employees

e4	John
e6	Kat

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Target Scenario 1
group
by **cname**

Orgs

Microsoft
Projects:

DB	e4

Google
Projects:

Web	e6

Employees

e4	John
e6	Kat

Target Scenario 2
do not group
by **cname**

Orgs

Microsoft
Projects:

DB	e4
Web	e6

Google
Projects:

DB	e4
Web	e6

Employees

e4	John
e6	Kat

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Target Scenario 1
group
by **cname**

Orgs

Microsoft
Projects: SK(Microsoft,y)
DB e4
Google
Projects: SK(Google,y)
Web e6

Employees
e4 John
e6 Kat

Target Scenario 2
do not group
by **cname**

Orgs

Microsoft
Projects: SK(y)
DB e4
Web e6
Google
Projects: SK(y)
DB e4
Web e6

Employees
e4 John
e6 Kat

$y \subseteq \{ \text{USA} \}$

Muse-G: Second Question

- The next probed attribute is cname

Example source

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Target Scenario 1
group
by **cname**

Orgs
Microsoft
Projects: SK(Microsoft,y)
DB e4
Google
Projects: SK(Google,y)
Web e6

Employees
e4 John
e6 Kat

Target Scenario 2
do not group
by **cname**

Orgs
Microsoft
Projects: SK(y)
DB e4
Web e6
Google
Projects: SK(y)
DB e4
Web e6

Employees
e4 John
e6 Kat

$y \subseteq \{ \text{USA} \}$

- The wizard continues to probe the remaining possible grouping attributes

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Query:

Companies(c_1,n_1,l_1) $\wedge$

Projects(p_1,pn_1,c_1,e_1) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Query:

Companies(c_1,n_1,l_1) $\wedge$

Projects(p_1,pn_1,c_1,e_1) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Companies(c_2,n_2,l_1) $\wedge$

Projects(p_2,pn_2,c_2,e_2) $\wedge$

Employees(e_2,en_2,cn_2) $\wedge$

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Query:

Companies(c_1,n_1,l_1) $\wedge$

Projects(p_1,pn_1,c_1,e_1) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Companies(c_2,n_2,l_1) $\wedge$

Projects(p_2,pn_2,c_2,e_2) $\wedge$

Employees(e_2,en_2,cn_2) $\wedge$

$n_1 \neq n_2$

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Query:

Companies(c_1,n_1,l_1) $\wedge$

Projects(p_1,pn_1,c_1,e_1) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Companies(c_2,n_2,l_1) $\wedge$

Projects(p_2,pn_2,c_2,e_2) $\wedge$

Employees(e_2,en_2,cn_2) $\wedge$

$n_1 \neq n_2$

Non-empty result →

Real Example:

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Obtaining Source Examples

Running queries over the real source instance I

Example: probing on cname

Query:

Companies(c_1,n_1,l_1) $\wedge$

Projects(p_1,pn_1,c_1,e_1) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Companies(c_2,n_2,l_1) $\wedge$

Projects(p_2,pn_2,c_2,e_2) $\wedge$

Employees(e_2,en_2,cn_2) $\wedge$

$n_1 \neq n_2$

Non-empty result → **Real Example:**

Companies

S. Valley	Microsoft	USA
Mt. View	Google	USA

Projects

P1	DB	S. Valley	e4
P4	Web	Mt. View	e6

Employees

e4	John	x234
e6	Kat	x331

Empty result → **Synthetic Example:**

Companies

c_1	n_1	l_1
c_2	n_2	l_1

Projects

p_1	pn_1	c_1	e_1
p_2	pn_2	c_2	e_2

Employees

e_1	en_1	cn_1
e_2	en_2	cn_2

Muse-G with FDs

- Considering functional dependencies in the source can reduce the number of questions posed to the designer

Muse-G with FDs

- Considering functional dependencies in the source can reduce the number of questions posed to the designer

- Two mappings M_1, M_2 have the same effect if for any source instance I, the result of exchanging I with M_1 is the "same" as the result of exchanging I with M_2

Homomorphically equivalent

Muse-G with FDs

- Considering functional dependencies in the source can reduce the number of questions posed to the designer

- Two mappings M_1, M_2 have the same effect if for any source instance I, the result of exchanging I with M_1 is the "same" as the result of exchanging I with M_2

Homomorphically equivalent

Proposition. *If a FD $A \rightarrow B$ holds, then a mapping M that groups by A has the same effect as a mapping M that groups by $A \cup C$, where $C \subseteq B$.*

Muse-G with FDs

- Considering functional dependencies in the source can reduce the number of questions posed to the designer

- Two mappings M_1, M_2 have the same effect if for any source instance I, the result of exchanging I with M_1 is the "same" as the result of exchanging I with M_2

Homomorphically equivalent

Proposition. *If a FD $A \rightarrow B$ holds, then a mapping M that groups by A has the same effect as a mapping M that groups by $A \cup C$, where $C \subseteq B$.*

- Suppose **cbranch** is a key, then we may save some questions
  - If the designer chooses Scenario 1 (including cbranch in the grouping function), probing on cname or location is no longer necessary

Muse-G: Properties

Proposition (Completeness). *If there are n possible grouping attributes for a nested set S, then the questions asked by Muse-G explore the entire space of 2^n grouping functions. Muse-G asks at most n questions to infer the desired grouping semantics for S.*

Muse-G: Properties

Proposition (Completeness)**.** *If there are n possible grouping attributes for a nested set S, then the questions asked by Muse-G explore the entire space of 2^n grouping functions. Muse-G asks at most n questions to infer the desired grouping semantics for S.*

Proposition (Small examples)**.** *At each probe, Muse-G constructs a source example of size at most twice the number of conjuncts in the* **for** *clause of the mapping.*

Muse-G: Properties

Proposition (Completeness)**.** *If there are n possible grouping attributes for a nested set S, then the questions asked by Muse-G explore the entire space of 2^n grouping functions. Muse-G asks at most n questions to infer the desired grouping semantics for S.*

Proposition (Small examples)**.** *At each probe, Muse-G constructs a source example of size at most twice the number of conjuncts in the* **for** *clause of the mapping.*

- Incremental design: group more or less starting from an existing grouping function

Muse-G: Properties

Proposition (Completeness)**.** *If there are n possible grouping attributes for a nested set S, then the questions asked by Muse-G explore the entire space of 2^n grouping functions. Muse-G asks at most n questions to infer the desired grouping semantics for S.*

Proposition (Small examples)**.** *At each probe, Muse-G constructs a source example of size at most twice the number of conjuncts in the* **for** *clause of the mapping.*

- Incremental design: group more or less starting from an existing grouping function
- Design for a specific source instance: reduce the number of questions
  - Muse-G identifies attributes whose inclusion/exclusion as arguments of grouping functions is inconsequential

Ambiguous Mappings

CompDB: *Rcd*
- **Projects:** *Set of*
  - **Project:** *Rcd*
    - pid
    - pname
    - manager
    - tech-lead
- **Employees:** *Set of*
  - **Employee:** *Rcd*
    - eid
    - ename
    - contact

Ambiguous Mappings

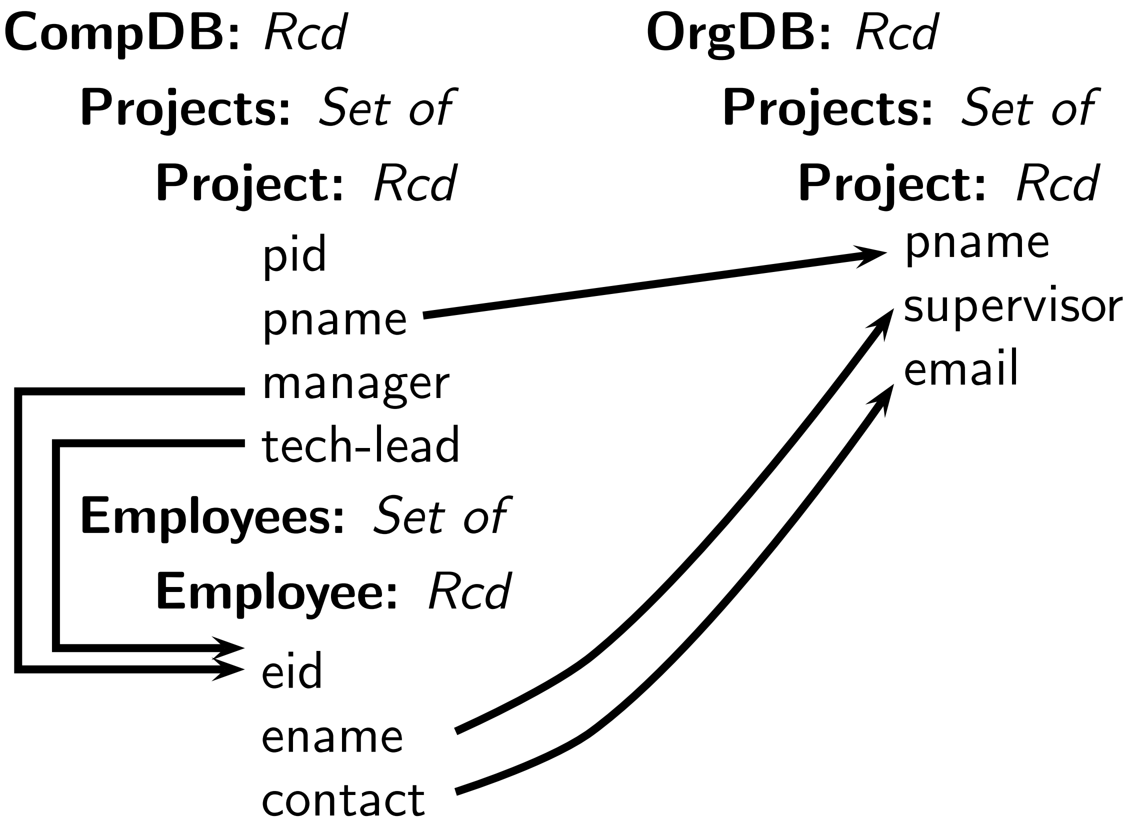
CompDB: *Rcd*

- **Projects:** *Set of*
  - **Project:** *Rcd*
    - pid
    - pname
    - manager
    - tech-lead
- **Employees:** *Set of*
  - **Employee:** *Rcd*
    - eid
    - ename
    - contact

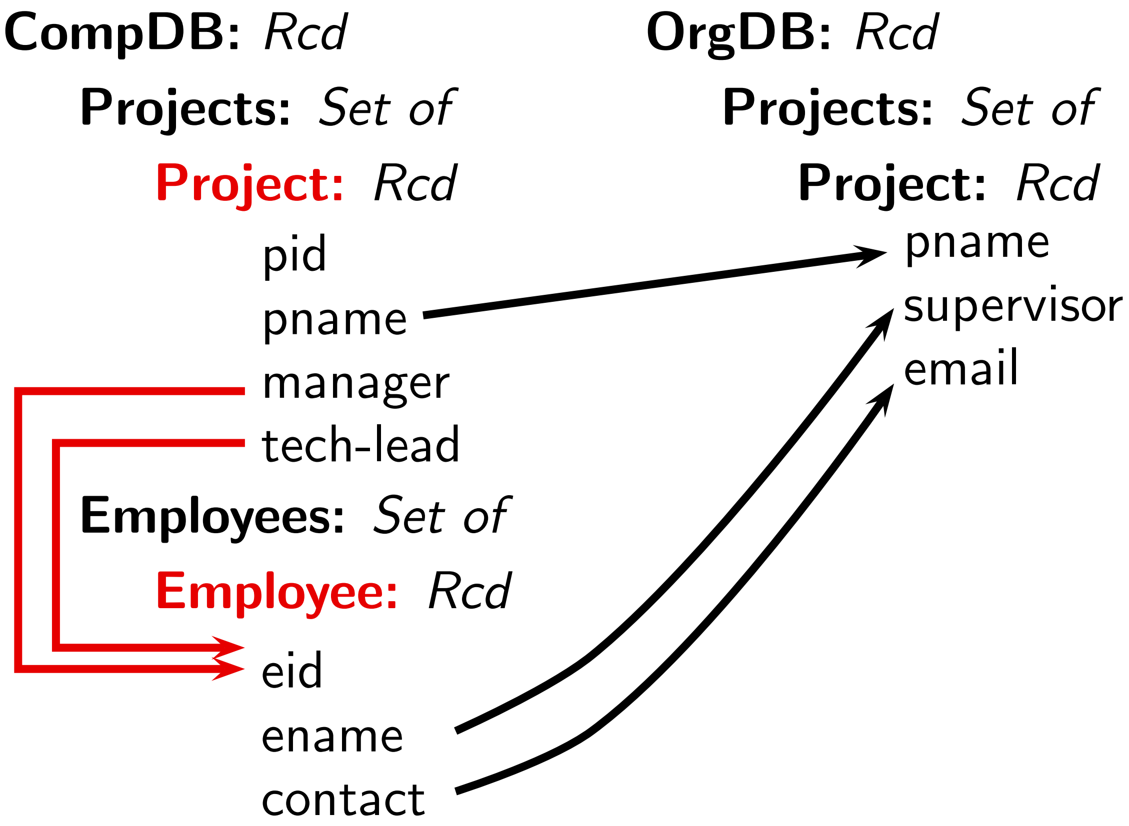
OrgDB: *Rcd*

- **Projects:** *Set of*
  - **Project:** *Rcd*
    - pname
    - supervisor
    - email

Ambiguous Mappings



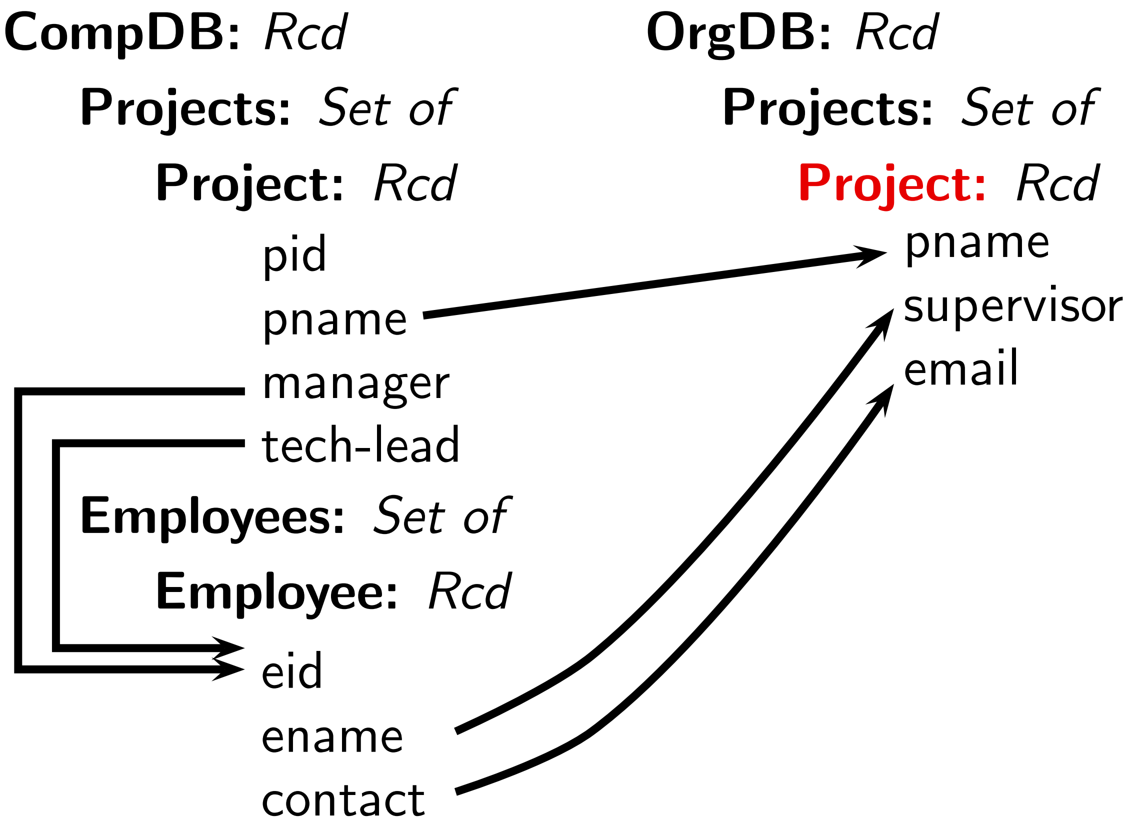
Ambiguous Mappings



for
p in CompDB.Projects
e_1 in CompDB.Employees
e_2 in CompDB.Employees

satisfy
e_1.eid = p.manager
e_2.eid = p.tech-lead

Ambiguous Mappings

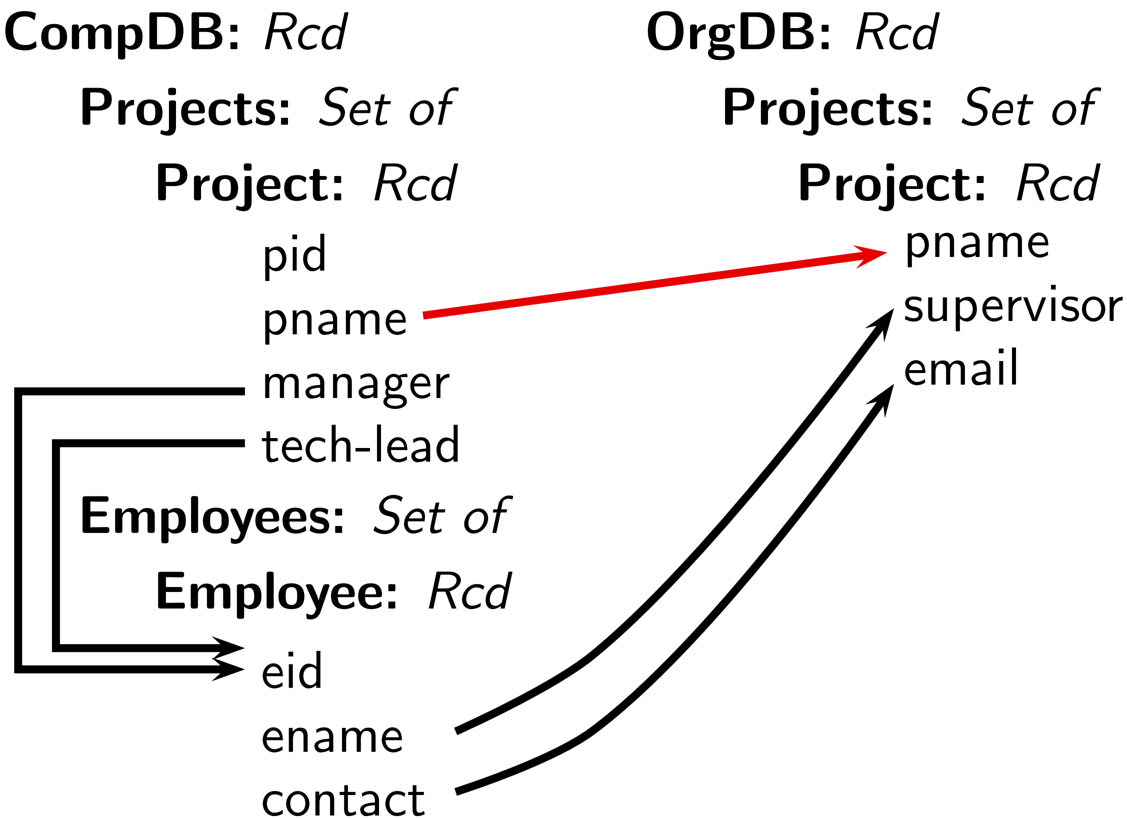


for
p in CompDB.Projects
e_1 in CompDB.Employees
e_2 in CompDB.Employees

satisfy
e_1.eid = p.manager
e_2.eid = p.tech-lead

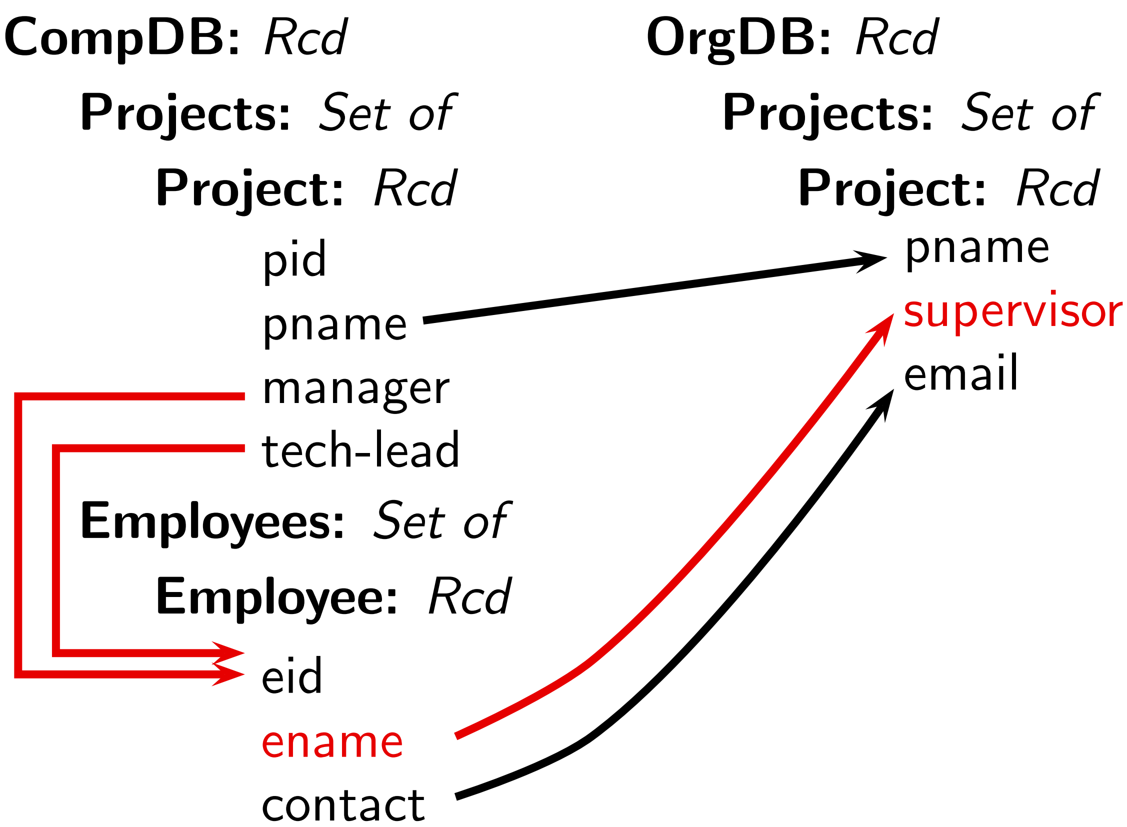
exists
p_1 in OrgDB.Projects

Ambiguous Mappings



for
  p in CompDB.Projects
  e_1 in CompDB.Employees
  e_2 in CompDB.Employees
satisfy
  e_1.eid = p.manager
  e_2.eid = p.tech-lead
exists
  p_1 in OrgDB.Projects
where
  p.pname = p_1.pname

Ambiguous Mappings



for

p in CompDB.Projects

e_1 in CompDB.Employees

e_2 in CompDB.Employees

satisfy

e_1.eid = p.manager

e_2.eid = p.tech-lead

exists

p_1 in OrgDB.Projects

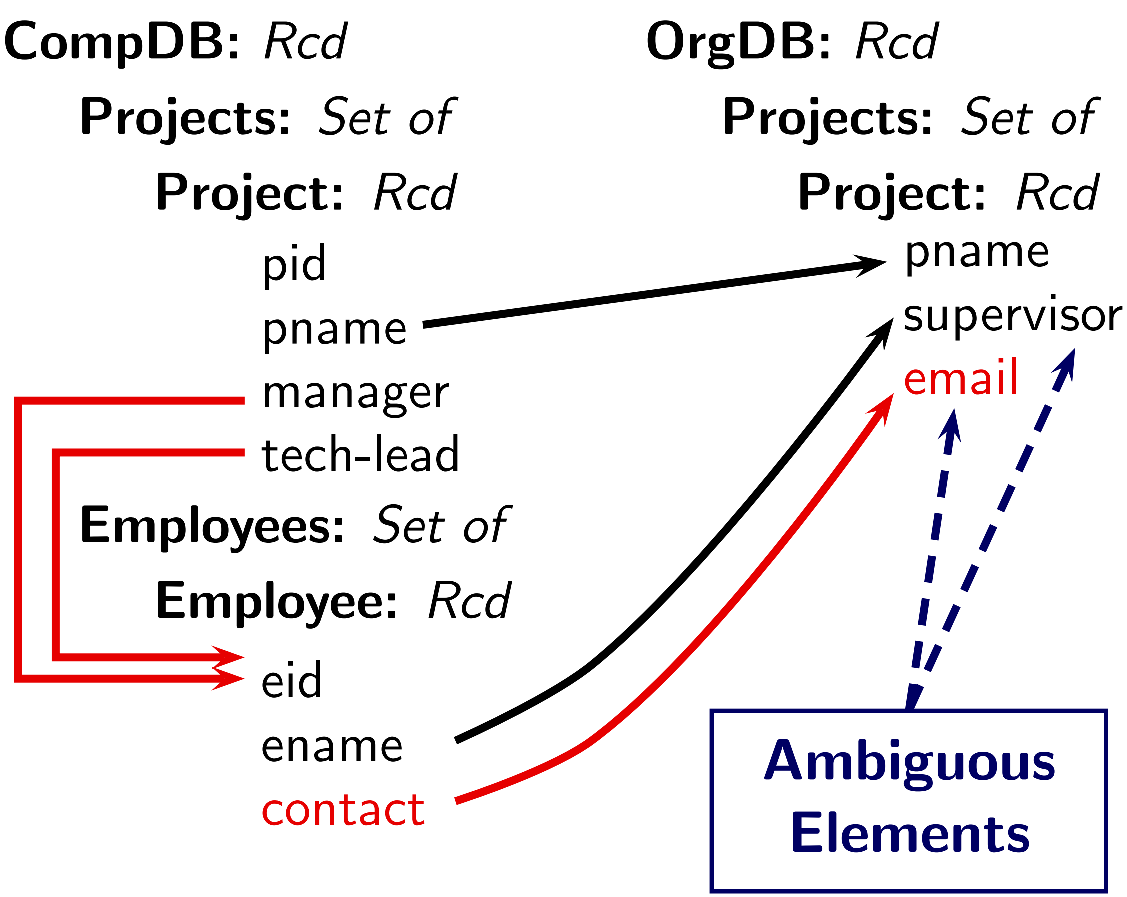
where

p.pname = p_1.pname

p_1.supervisor =

e_1.ename **or** e_2.ename

Ambiguous Mappings



for
p in CompDB.Projects
e_1 in CompDB.Employees
e_2 in CompDB.Employees
satisfy
e_1.eid = p.manager
e_2.eid = p.tech-lead
exists
p_1 in OrgDB.Projects
where
p.pname = p_1.pname
p_1.supervisor =
e_1.ename **or** e_2.ename
p_1.email =
e_1.contact **or** e_2.contact

- This mapping is ambiguous
- There are four alternative interpretations

e_1.ename	e_1.ename	e_2.ename	e_2.ename
e_1.contact	e_2.contact	e_1.contact	e_2.contact

Muse-D: Disambiguating Mappings

- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Muse-D: Disambiguating Mappings

- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm

Muse-D: Disambiguating Mappings

- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm

Orgs

Projects:

DB	John	john@ibm
	Anna	anna@ibm

Muse-D: Disambiguating Mappings

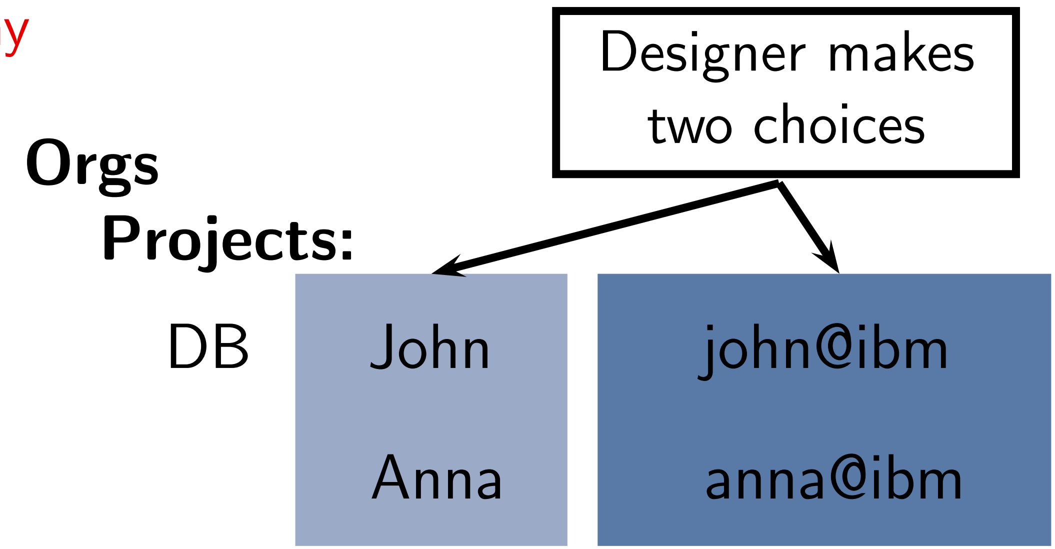
- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm



- The mapping designer makes one choice for each ambiguous element
- Each decision removes one ambiguity

Muse-D: Disambiguating Mappings

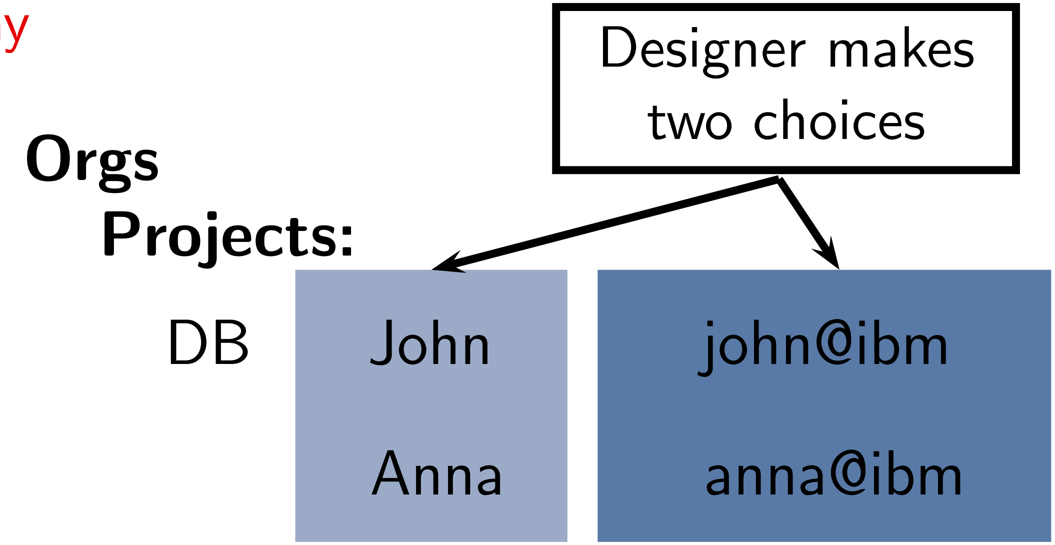
- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm



- The mapping designer makes one choice for each ambiguous element
- Each decision removes one ambiguity
  - E.g., choosing "Anna" as the supervisor and "john@ibm" as the email

p_1.supervisor $=$ e_1.ename **or** e_2.ename | p_1.email $=$ e_1.contact **or** e_2.contact

Muse-D: Disambiguating Mappings

- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

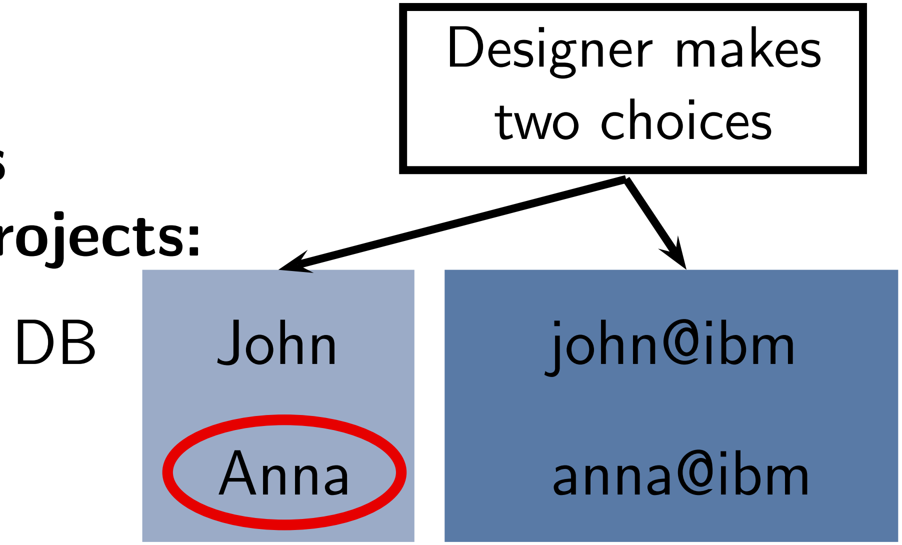
P1 DB e4 e5

Employees

e4 John john@ibm
e5 Anna anna@ibm

Orgs

Projects:



- The mapping designer makes one choice for each ambiguous element
- Each decision removes one ambiguity
  - E.g., choosing "Anna" as the supervisor and "john@ibm" as the email

p_1.supervisor = e_2.ename | p_1.email = e_1.contact **or** e_2.contact

Muse-D: Disambiguating Mappings

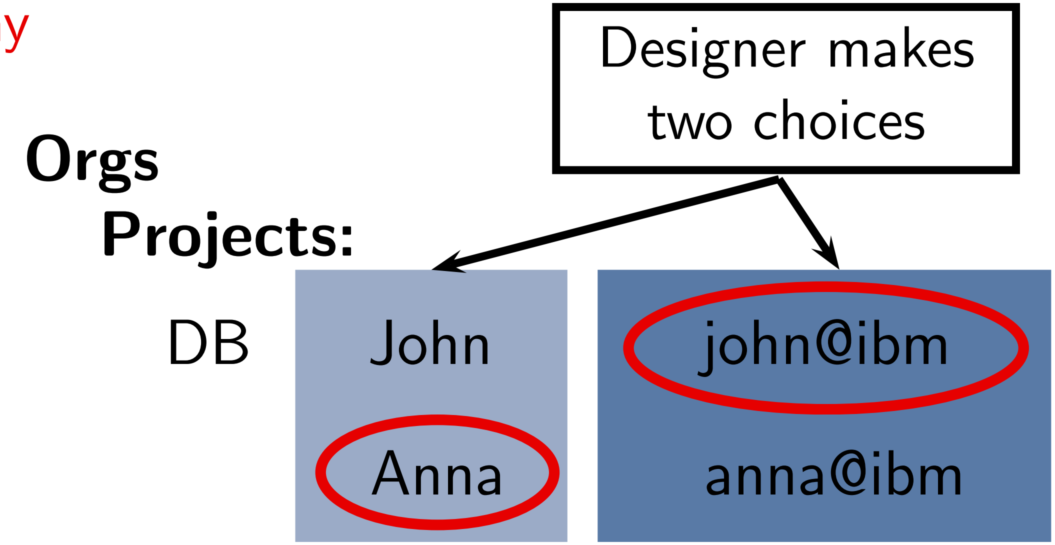
- **Key idea:** provide an example that illustrates the alternative interpretations in a compact way

Projects

P1 DB e4 e5

Employees

e4 John john@ibm
e5 Anna anna@ibm



- The mapping designer makes one choice for each ambiguous element
- Each decision removes one ambiguity
  - E.g., choosing "Anna" as the supervisor and "john@ibm" as the email

p_1.supervisor = e_2.ename | p_1.email = e_1.contact

Obtaining Source Examples

Running queries over the real source instance

Obtaining Source Examples

Running queries over the real source instance

Query:

$\text{Projects}(p_1, pn_1, e_1, e_2) \wedge$

$\text{Employees}(e_1, en_1, cn_1) \wedge$

$\text{Employees}(e_2, en_2, cn_2) \wedge$

Obtaining Source Examples

Running queries over the real source instance

Query:

Projects(p_1,pn_1,e_1,e_2) $\wedge$

Employees(e_1,en_1,cn_1) $\wedge$

Employees(e_2,en_2,cn_2) $\wedge$

$en_1 \neq en_2 \wedge cn_1 \neq cn_2$

Obtaining Source Examples

Running queries over the real source instance

Query:

$\text{Projects}(p_1, pn_1, e_1, e_2) \wedge$

$\text{Employees}(e_1, en_1, cn_1) \wedge$

$\text{Employees}(e_2, en_2, cn_2) \wedge$

$en_1 \neq en_2 \wedge cn_1 \neq cn_2$

Non-empty result →

Real Example:

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm

Obtaining Source Examples

Running queries over the real source instance

Query:

$Projects(p_1, pn_1, e_1, e_2) \wedge$

$Employees(e_1, en_1, cn_1) \wedge$

$Employees(e_2, en_2, cn_2) \wedge$

$en_1 \neq en_2 \wedge cn_1 \neq cn_2$

Non-empty result

Real Example:

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm

Empty result

Synthetic Example:

Projects

p_1	pn_1	e_1	e_2

Employees

e_1	en_1	cn_1
e_2	en_2	cn_2

Muse-D: Properties

- For each ambiguous mapping, the designer is presented with a single example

Muse-D: Properties

- For each ambiguous mapping, the designer is presented with a single example

Proposition (Completeness)**.** *The single example differentiates among all the alternative interpretations of the ambiguous mapping. The mapping designer has to make a number of choices equal to the number of ambiguous elements.*

Muse-D: Properties

- For each ambiguous mapping, the designer is presented with a single example

Proposition (Completeness)**.** *The single example differentiates among all the alternative interpretations of the ambiguous mapping. The mapping designer has to make a number of choices equal to the number of ambiguous elements.*

Proposition (Small examples)**.** *The number of tuples in the example source instance is the number of conjuncts in the* **for** *clause of the mapping.*

Experiments: Setting

Mapping Scenarios	Size of real source instance	Sets with refinable grouping	Number of mappings	Ambiguous mappings	Alternative interpretations
Mondial	1MB	8	26	7	208
DBLP	2.6MB	6	4	0	4
TPCH	10MB	4	5	1	20
Amalgam	2MB	2	14	0	14

- Mapping system: Clio
- Query evaluation: DB2 v9, Saxon-B 8.9
- MUSE implementation: Java 6

Experiments: Muse-G

Mapping scenario	Average # of grouping attributes	Number of questions (average)	% times found real example	Average time to get real example (s)	Grouping strategy
Mondial	13.1	2.6	38%	0.014	G_1
		8.5	41%	0.187	G_2
		2.9	40%	0.015	G_3
DBLP	**11**	**1.5**	**17%**	**0.450**	$\mathbf{G_1}$
		11	**11%**	**0.337**	$\mathbf{G_2}$
		1.5	**17%**	**0.454**	$\mathbf{G_3}$
TPCH	26.7	1.5	0%	0.785	G_1
		17	12%	0.893	G_2
		1.5	0%	0.782	G_3
Amalgam	14.1	2	29%	0.013	G_1
		3	52%	0.043	G_2
		3	52%	0.030	G_3

G_1: group by all possible attributes (cbranch,cname,location,pid,...)

G_2: group by all attributes exported above the set (cname)

G_3: group by all exported attributes (cname, pname, eid, ename)

Experiments: Muse-D

Mapping Scenario	Alternatives encoded	Number of questions	Size of source example (# of tuples)	Number of ambiguous values in target instance
Mondial	**208**	**7**	**3–4**	**4–5**
TPCH	16	1	9	4

Related Work

- There are many related works, we focus here on the closest one
- We were inspired by DataViewer [Yan et al. 01]

Feature \ System	**MUSE**	**DataViewer**
Examples to be analyzed	Compact representation (e.g. 7)	As many as alternative interpretations (e.g. 208)
Grouping Semantics	Yes	No
Data Models	Relational and XML	Relational
Mapping Language	Source-target mappings (GLAV)	SQL

Conclusion

- MUSE: a mapping design wizard
  - Use examples to understand, design, refine schema mappings
  - Work with **complete and small data examples** rather than complex specifications
  - Focus on two important aspects of a mapping specification
    - Grouping semantics
    - Interpretation of ambiguous mappings

Thank you