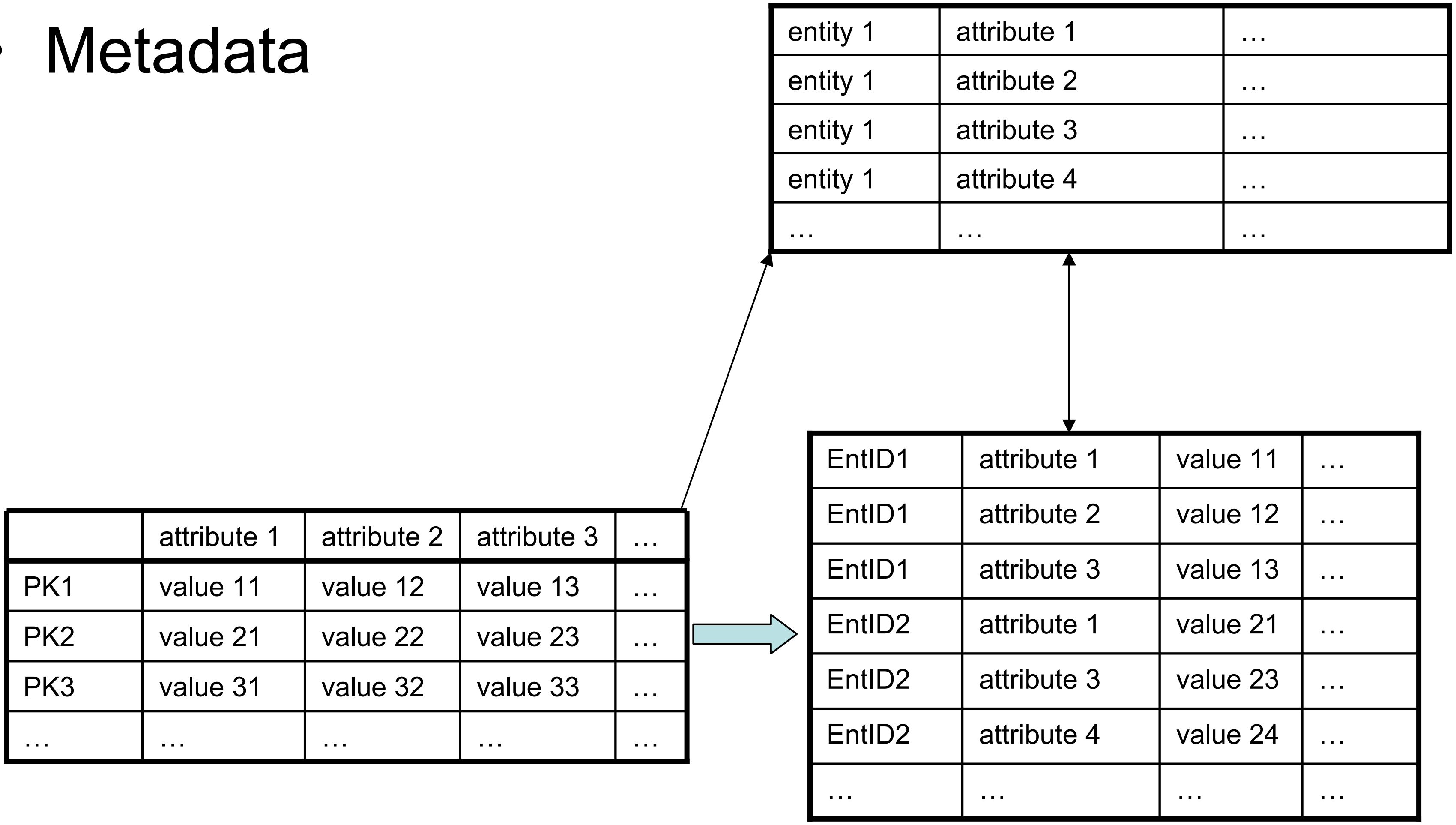
Transforming Data from DataPile Structure into RDF

Jiří Dokulil
Faculty of Mathematics and Physics
Charles University in Prague

DataPile

- Data verticalization
- Metadata



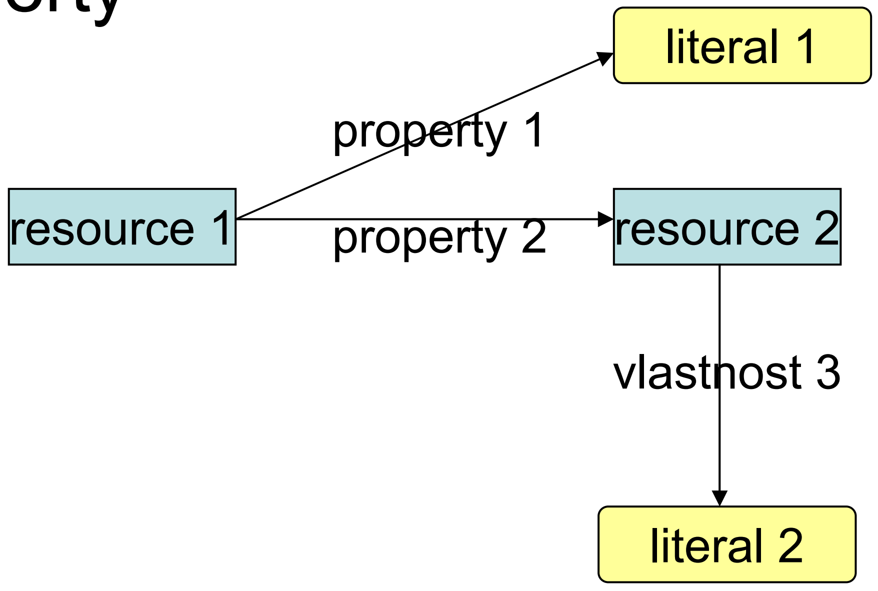
entity 1	attribute 1	…
entity 1	attribute 2	…
entity 1	attribute 3	…
entity 1	attribute 4	…
…	…	…

	attribute 1	attribute 2	attribute 3	…
PK1	value 11	value 12	value 13	…
PK2	value 21	value 22	value 23	…
PK3	value 31	value 32	value 33	…
…	…	…	…	…

EntID1	attribute 1	value 11	…
EntID1	attribute 2	value 12	…
EntID1	attribute 3	value 13	…
EntID2	attribute 1	value 21	…
EntID2	attribute 3	value 23	…
EntID2	attribute 4	value 24	…
…	…	…	…

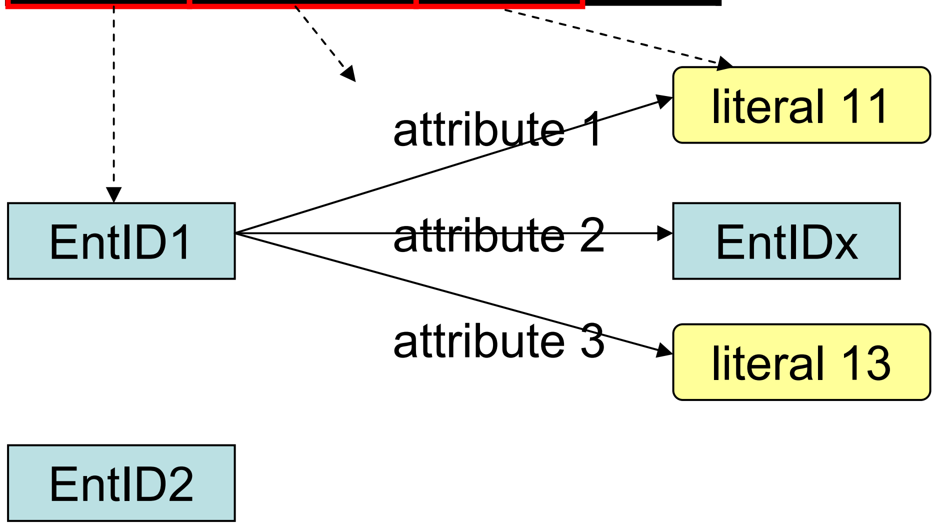
RDF

- triples
  - subject – resource being described
  - predicate – property
  - object – value of the property
- metadata
  - stored with data



DataPile → RDF

EntID1	attribute 1	value 11	…
EntID1	attribute 2	value 12	…
EntID1	attribute 3	value 13	…
EntID2	attribute 1	value 21	…
EntID2	attribute 2	value 23	…
EntID2	attribute 4	value 24	…
…	…	…	…



DataPile → RDF example

598472635	first_name	Jiří	…
598472635	last_name	Dokulil	…
598472635	place_of_birth	972324584	…
…	…	…	…

_:598472635 pile:first_name “Jiří“

_:598472635 pile:last_name “Dokulil“

_:598472635 pile:place_of_birth _:972324584

Special case: multilingual attributes

- complicated in DataPile
  - 2 entities, hard to work with
  - not only language but cases as well
- RDF has direct support for specifying language of literals
  - RFC 3066 → flexible enough to express language, case, …

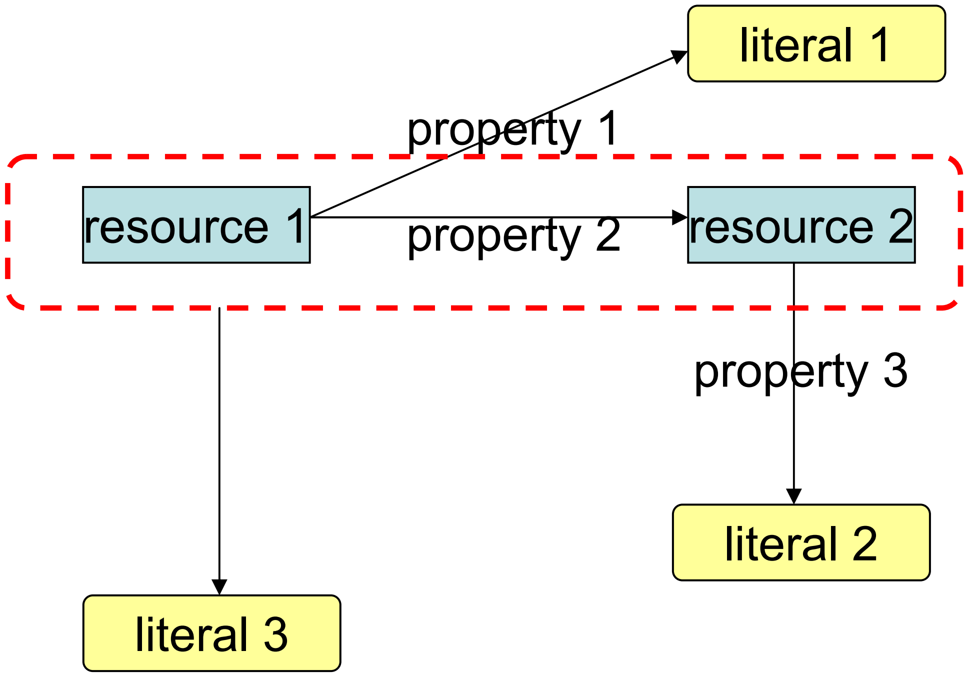
Metadata

- separated from data in the DataPile
  - tree, 2 levels
    - entities
    - attributes
  - data types
    - string, number, timestamp
    - ID – reference to other entity – typed reference
- everything is a triple in RDF
  - easy to represent the metadata tree from DataPile
  - data types taken from XSD (and simplified)
    - enough to cover types in DataPile
  - object of a triple can be anything → we need more constraints → RDFS

RDFS – RDF Schema

- predicates have to be URIs (resources) → attibutes from DataPile must be transformed to RDF
- transforming entities is easy
- entities serve as domains of predicates
- ranges of predicates can be either specific data types or instances of other entites → RDFS can provide same level of type checking as DataPile
- DataPile does not support is-a hierarchies while RDFS can define subclasses and subproperties

Reification – making statements about statements



Reification in DataPile

EntID1	attribute 1	value 11	…	valid_from 1	valid_to 1	modified_by 1	…
EntID1	attribute 2	value 12	…	valid_from 2	valid_to 2	modified_by 2	…
EntID1	attribute 3	value 13	…	valid_from 3	valid_to 3	modified_by 3	…
EntID2	attribute 1	value 21	…	valid_from 4	valid_to 4	modified_by 4	…
EntID2	attribute 3	value 23	…	valid_from 5	valid_to 5	modified_by 5	…
EntID2	attribute 4	value 24	…	valid_from 6	valid_to 6	modified_by 6	…
…	…	…	……		…	…	…

- special case of reification with fixed set of predicates

Reification in RDF

```
_:568421369754123695 mt:person__name
“John Smith” .
_:r65413 rdf:type rdf:Statement .
_:r65413 rdf:subject _:568421369754123695 .
_:r65413 rdf:predicate mt:person__name .
_:r65413 rdf:object “John Smith” .
_:r65413 mt:valid_from “20050703T15:21:49” .
_:r65413 mt:valid_to “20050821T09:35:12” .
```

Conclusion

- Basic transformation is easy
- Multilingual attributes can be expressed better in RDF
- No real reification in RDF → transforming validity period is not nice