JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Interdisziplinäre Anwendungen Raumbezogener Informationstechnik

Semantik

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Semantik

GOLF

Golf

Golf

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Golf

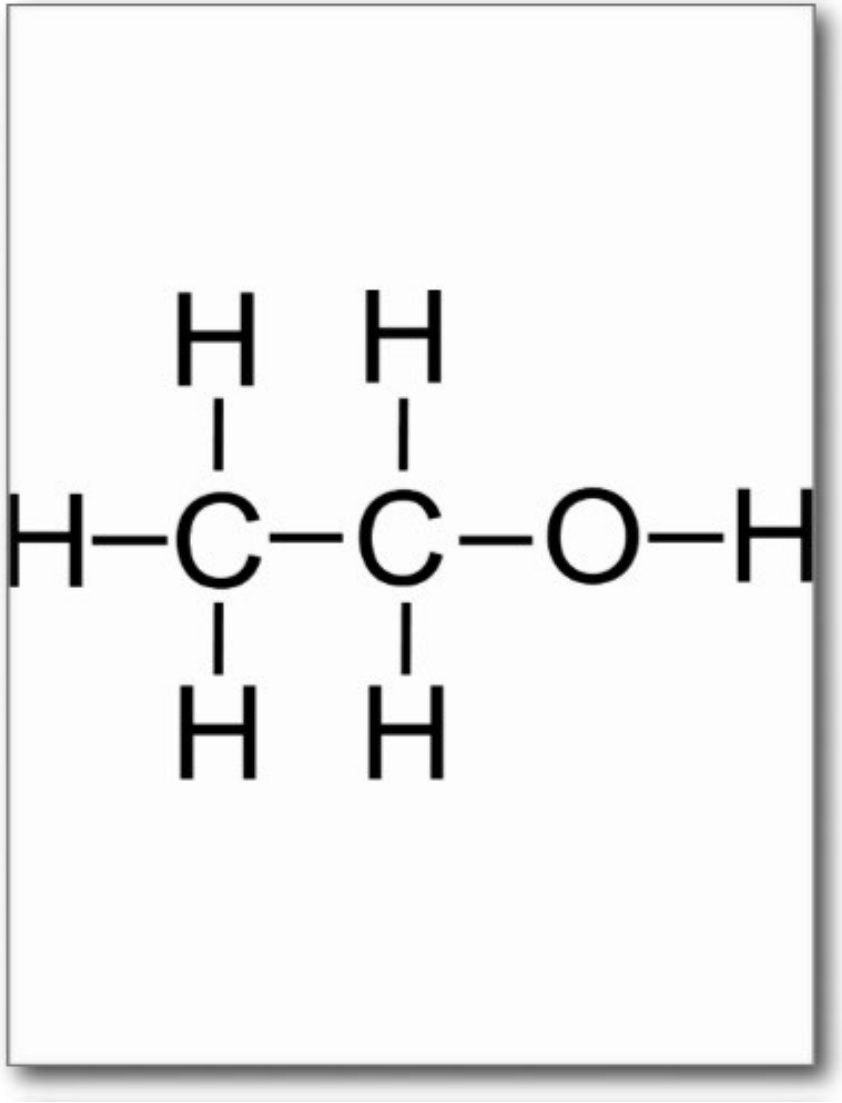
JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Semantik

ALKOHOL

Alkohol



JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Alkohol

Alkohol

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Semantik

DEUTSCHE BAHN

Deutsche Bahn

Deutsche Bahn

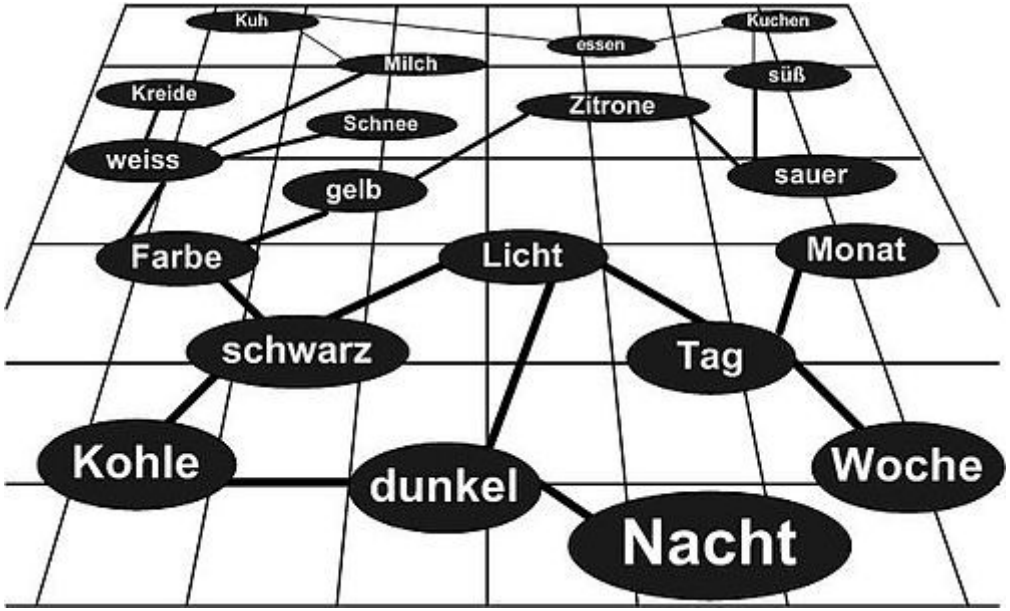
JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Deutsche Bahn

Semantik

- Menschliche Assoziationen



Triples

Triples

- **SUBJEKT** – Prädikat – **OBJEKT**

Triples

- **SUBJEKT** – Prädikat – **OBJEKT**

  Beispiel:

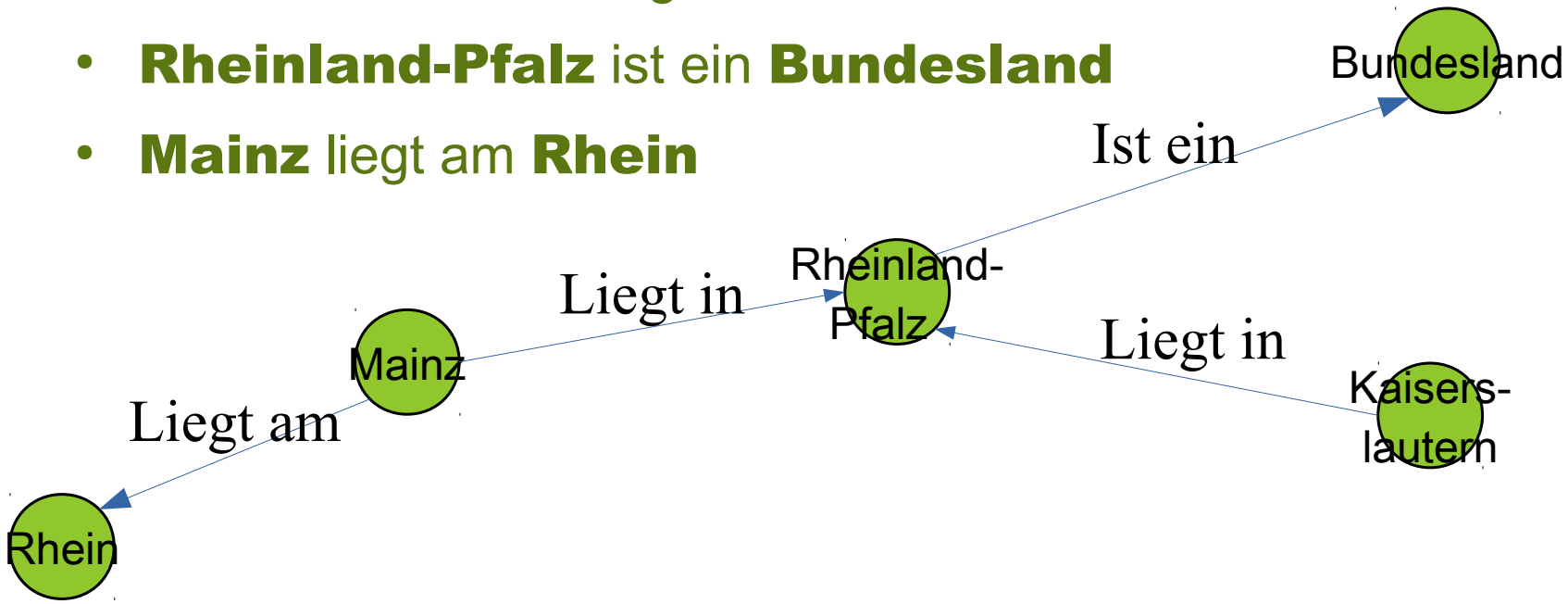
  - **Mainz** liegt in **Rheinland-Pfalz**
  - **Kaiserslautern** liegt in **Rheinland-Pfalz**
  - **Rheinland-Pfalz** ist ein **Bundesland**
  - **Mainz** liegt am **Rhein**

Triples

- **SUBJEKT** – Prädikat – **OBJEKT**

  Beispiel:

  - **Mainz** liegt in **Rheinland-Pfalz**
  - **Kaiserslautern** liegt in **Rheinland-Pfalz**
  - **Rheinland-Pfalz** ist ein **Bundesland**
  - **Mainz** liegt am **Rhein**



Triples

- PROPERTY: Verben (Prädikate) – entspricht Kantenbeschriftung
  - Beziehungen zwischen Daten
  - Festgelegt, Kontrolliert
  - Nicht beliebig erweiterbar
  - Anzahl eher gering

- RESOURCE: Substantive (Subjekte, Objekte) – entspricht Knoten
  - Eigentliche Daten
  - Beliebig erweiterbar

Noch ein Beispiel

Beispiel

- Gegeben sind folgende Informationen

Adam arbeitet bei Opel.
Berta arbeitet bei BMW.

Berta fährt Korsa.
Berta fährt Z5.

Chris arbeitet bei Audi.
Daniel arbeitet bei BMW.
Daniel arbeitet bei Audi.

Daniel fährt A3.

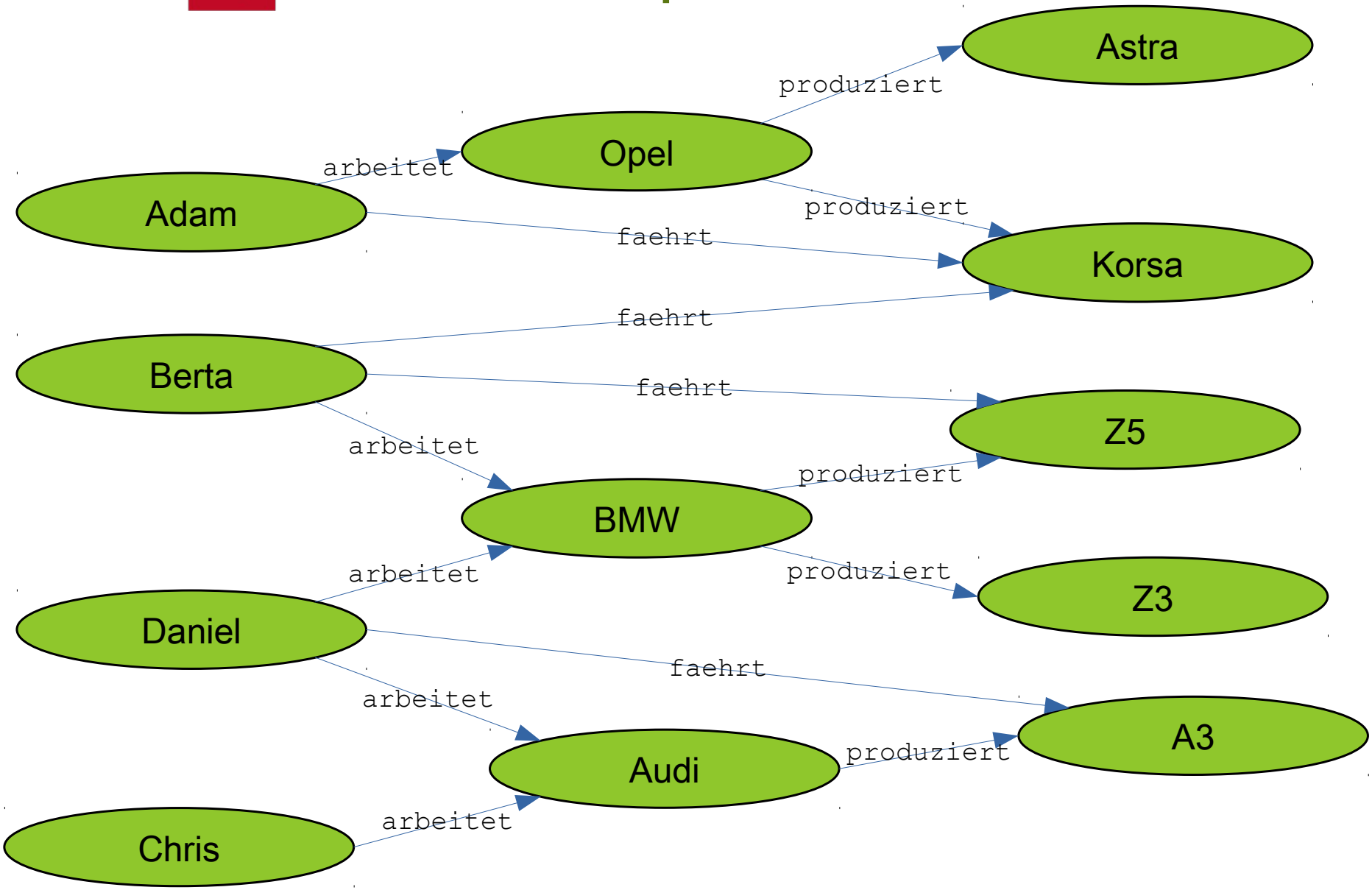
Opel produziert Korsa.
BMW produziert Z3.
Audi produziert A3.

Opel produziert Astra.
BMW produziert Z5.

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Beispiel



Beispiel


http://idarit.hs-mainz.de/Astra
http://idarit.hs-mainz.de/produziert
http://idarit.hs-mainz.de/Opel
http://idarit.hs-mainz.de/arbeitet
http://idarit.hs-mainz.de/Adam
http://idarit.hs-mainz.de/produziert
http://idarit.hs-mainz.de/faehrt
http://idarit.hs-mainz.de/Korsa
http://idarit.hs-mainz.de/faehrt
http://idarit.hs-mainz.de/Berta
http://idarit.hs-mainz.de/faehrt
http://idarit.hs-mainz.de/Z5
http://idarit.hs-mainz.de/arbeitet
http://idarit.hs-mainz.de/produziert
http://idarit.hs-mainz.de/BMW
http://idarit.hs-mainz.de/arbeitet
http://idarit.hs-mainz.de/produziert
http://idarit.hs-mainz.de/Z3
http://idarit.hs-mainz.de/Daniel
http://idarit.hs-mainz.de/faehrt
http://idarit.hs-mainz.de/arbeitet
http://idarit.hs-mainz.de/A3
http://idarit.hs-mainz.de/produziert
http://idarit.hs-mainz.de/Audi
http://idarit.hs-mainz.de/arbeitet
http://idarit.hs-mainz.de/Chris

Beispiel

- In Turtle Syntax

```
@prefix idarit: <http://idarit.hs-mainz.de/> .
idarit:Adam   idarit:arbeitet   idarit:Opel .
idarit:Adam   idarit:faehrt     idarit:Korsa .
idarit:Berta  idarit:arbeitet   idarit:BMW .
idarit:Berta  idarit:faehrt     idarit:Korsa .
idarit:Berta  idarit:faehrt     idarit:Z5 .
idarit:Chris  idarit:arbeitet   idarit:Audi .
idarit:Daniel idarit:arbeitet   idarit:BMW .
idarit:Daniel idarit:faehrt     idarit:A3 .
idarit:Opel   idarit:produziert idarit:Korsa .
idarit:Opel   idarit:produziert idarit:Astra .
idarit:BMW    idarit:produziert idarit:Z3 .
idarit:BMW    idarit:produziert idarit:Z5 .
idarit:Audi   idarit:produziert idarit:A3 .
```

Beispiel

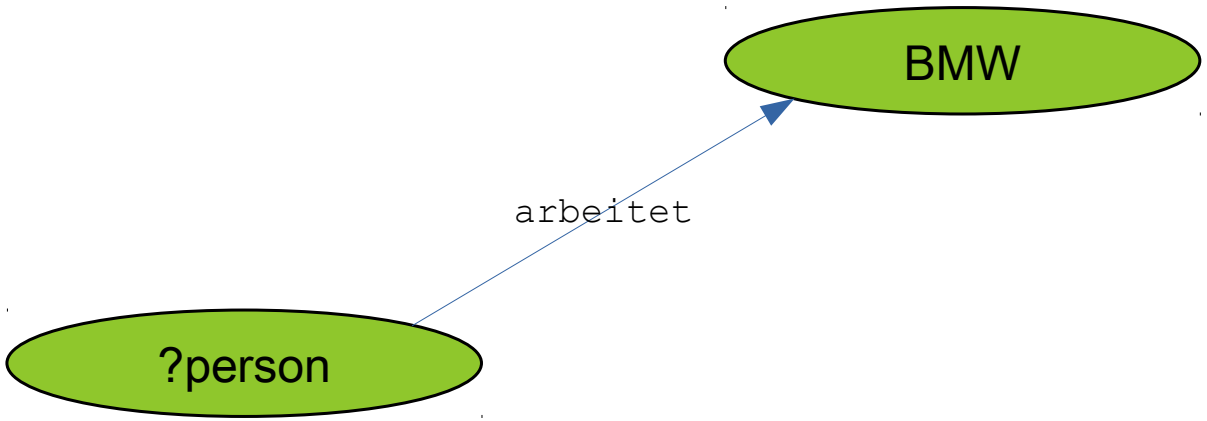
- Abfrage an die Datenbank

  Wer arbeitet bei BMW?

Beispiel

- Abfrage an die Datenbank

  Wer arbeitet bei BMW?



Beispiel

- Abfrage an die Datenbank

  Wer arbeitet bei BMW?

```
PREFIX idarit: <http://idarit.hs-mainz.de/>

SELECT ?person WHERE {

   ?person idarit:arbeitet idarit:BMW .

}
```

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U

Beispiel

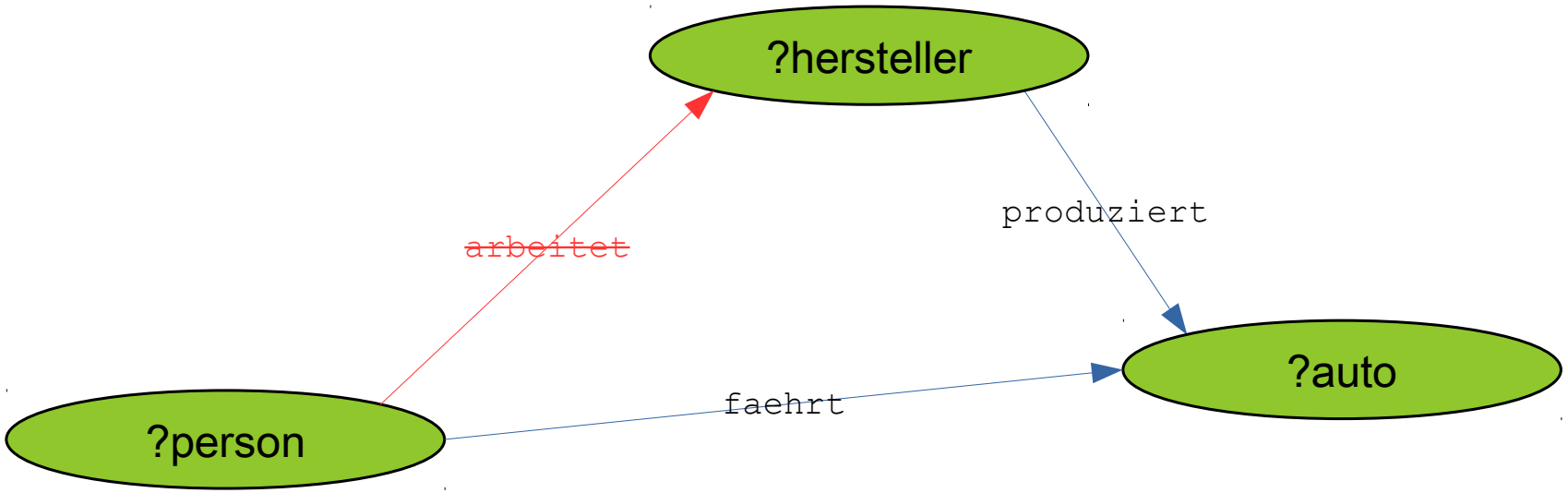
- Abfrage an die Datenbank

  Wer fährt ein Auto eines Herstellers, bei dem er nicht arbeitet?

Beispiel

- Abfrage an die Datenbank

  Wer fährt ein Auto eines Herstellers, bei dem er nicht arbeitet?



Beispiel

- Abfrage an die Datenbank

  Wer fährt ein Auto eines Herstellers, bei dem er nicht arbeitet?

```
PREFIX idarit: <http://idarit.hs-mainz.de/>

SELECT ?person WHERE {

   ?person idarit:faehrt ?auto .
   ?hersteller idarit:produziert ?auto .

   FILTER NOT EXISTS {
      ?person idarit:arbeitet ?hersteller .
   }

}
```

Ontologien

Ontologie

- Formale Beschreibung von Begrifflichkeiten
  - Welche Beziehungen zwischen welchen Ressourcen sind erlaubt?
  - Welches Wissen wird mit welchen Begriffen beschrieben?
- Wozu?
  - Interne Strukturierung des Datenbestandes
  - Austausch von Wissen zwischen verschiedenen Systemen
- Woraus besteht eine Ontologie?
  - Integritätsregeln (Zur Gewährleistung der Datenkonsistenz)
  - Inferenzregeln (Zur automatischen Generierung von neuem Wissen)

Ontologie

- Für das vorherige Beispiel
  - Es gibt Personen, Hersteller und Autos
  - Personen arbeiten bei Herstellern
  - Hersteller produzieren Autos
  - Personen fahren Autos

Ontologie

- Es gibt Personen, Hersteller und Autos

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

@prefix idarit: <http://idarit.hs-mainz.de/> .

idarit:Person      rdf:type   owl:Class .
idarit:Hersteller  rdf:type   owl:Class .
idarit:Auto        rdf:type   owl:Class .
```

Ontologie

- Personen arbeiten bei Herstellern

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

@prefix idarit: <http://idarit.hs-mainz.de/> .

idarit:arbeitet rdf:type    owl:ObjectProperty .
idarit:arbeitet rdfs:domain idarit:Person .
idarit:arbeitet rdfs:range  idarit:Hersteller .
```

Ontologie

- Hersteller produzieren Autos

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

@prefix idarit: <http://idarit.hs-mainz.de/> .

idarit:produziert rdf:type    owl:ObjectProperty .
idarit:produziert rdfs:domain idarit:Hersteller .
idarit:produziert rdfs:range  idarit:Auto .
```

Ontologie

- Hersteller produzieren Autos

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

@prefix idarit: <http://idarit.hs-mainz.de/> .

idarit:faehrt rdf:type    owl:ObjectProperty .
idarit:faehrt rdfs:domain idarit:Person .
idarit:faehrt rdfs:range  idarit:Auto .
```

Ontologie

`rdf:type`

- Verwendung

  `X rdf:type C .`

  - X ist eine Instanz der Klasse C
  - X ist eine Ressource
  - C ist eine Klasse
- Anmerkung
  - Man kann statt `rdf:type` auch nur `a` schreiben
- Beispiel
  - Adam ist eine Person.

    `idarit:Adam rdf:type idarit:Person .`

Ontologie

`owl:Class`

- Verwendung

  ```
  C rdf:type owl:Class .
  ```

  - C ist eine Klasse
- Anmerkung
  - Es gilt: `owl:Class rdf:type owl:Class .`
- Beispiele
  - Person ist eine Klasse.
    ```
    idarit:Person rdf:type owl:Class .
    ```
  - Buch ist eine Klasse.
    ```
    bib:Buch rdf:type owl:Class .
    ```

Ontologie

rdfs:Resource

- Verwendung

  ```
  X rdf:type rdfs:Resource .
  ```

  - X ist eine Ressource
- Anmerkung
  - Jede URI, die irgendwo als Subjekt oder Objekt auftaucht, ist eine Ressource, das Objekt kann auch ein Literal sein

    ```
    X P Y . =>  X rdf:type rdfs:Resource .
                [Y rdf:type rdfs:Resource .]
    ```
- Beispiele
  - Person ist eine Klasse.

    ```
    idarit:Adam rdf:type rdfs:Resource .
    ```

Ontologie

rdf:Property

- Verwendung

  ```
  P rdf:type rdf:Property .
  ```

  - P ist eine Property
- Anmerkung
  - Jede URI, die irgendwo als Prädikat auftaucht, ist eine Property, d.h.

    ```
    X P Y . => P rdf:type rdf:Property .
    ```
- Beispiele
  - Produziert ist eine Property.

    ```
    idarit:produziert rdf:type rdf:Property .
    ```

Ontologie

rdfs:domain

- Verwendung

  ```
  P rdfs:domain C .
  ```

  - Wird P als Prädikat verwendet, so muss das Subjekt eine Instanz der Klasse C sein

    ```
    X P Y . => X rdf:type C .
    ```

- Beispiele
  - Produziert muss als ausgehenden Knoten einen Hersteller haben.

    ```
    idarit:produziert rdfs:domain
    idarit:Hersteller .
    ```

Ontologie

rdfs:range

- Verwendung

  ```
  P rdfs:range C .
  ```

  - Wird P als Prädikat verwendet, so muss das Objekt eine Instanz der Klasse C sein

    ```
    X P Y . => Y rdf:type C .
    ```

- Beispiele
  - Produziert muss als eingehenden Knoten ein Auto haben.

    ```
    idarit:produziert rdfs:range idarit:Auto .
    ```

Ontologie

rdfs:Literal

- Verwendung

  ```
  P rdfs:range rdfs:Literal .
  ```

  - Wird die Property P als Prädikat verwendet, muss das Objekt ein Literal sein, also keine Ressource
- Anmerkung
  - Mit `xsd` lassen sich Datentypen genauer spezifizieren
- Beispiele
  - Die Einwohnerzahl ist ein Literal

    ```
    idarit:hatEinwohner rdfs:domain idarit:Stadt .
    idarit:hatEinwohner rdfs:range  rdfs:Literal .
    ```

Ontologie

rdfs:subClassOf

- Verwendung

  ```
  C rdfs:subClassOf D .
  ```

  - Jede Instanz der Klasse C ist auch eine Instanz der Klasse D

    ```
    X rdf:type C . => X rdf:type D .
    ```

- Beispiele
  - Jede Frau ist eine Person

    ```
    idarit:Man rdfs:subClassOf idarit:Person .
    ```

Ontologie

`rdfs:subPropertyOf`

- Verwendung

  ```
  P rdfs:subPropertyOf Q .
  ```

  - Jede Verbindung zweier Ressourcen mit der Property P ist auch eine Verbindung mit der Property Q

    ```
    X P Y . => X Q Y .
    ```

- Beispiele
  - Jeder Bruder von ist auch ein Geschwister von

    ```
    idarit:brotherOf rdfs:subPropertyOf
    idarit:siblingOf .
    ```

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Ontologie

`owl:ObjectProperty`

- Verwendung

  ```
  P rdf:type owl:ObjectProperty .
  ```

  - Wird die Property P als Prädikat verwendet, so muss das Objekt eine Ressource sein

    ```
    P rdfs:range rdfs:Ressource .
    ```

Ontologie

`owl:DatatypeProperty`

- Verwendung

  ```
  P rdf:type owl:DatatypeProperty .
  ```

  - Wird die Property P als Prädikat verwendet, so muss das Objekt ein Literal sein

    ```
    P rdfs:range rdfs:Literal .
    ```

Ontologie

`owl:TransitiveProperty`

- Verwendung

  ```
  P rdf:type owl:TransitiveProperty .
  ```

  - Properties, die dieser Klasse angehören, sind transitiv, d.h. aus zwei Verbindungen X → Y und Y → Z kann die Verbindung X → Z gefolgert werden

    ```
    X P Y . Y P Z . => X P Z .
    ```

- Beispiele
  - Jünger, Größer, Schwerer, Langsamer, …
  - Nördlich von, …
  - (Voll-)Geschwister

Ontologie

`owl:SymmetricProperty`

- Verwendung

  ```
  P rdf:type owl:SymmetricProperty .
  ```

  - Properties, die dieser Klasse angehören, sind symmetrisch, d.h. aus einer Verbindung X → Y folgt die Gegenrichtung Y → X

    ```
    X P Y . => Y P X .
    ```

- Beispiele
  - Gleich, Äquivalent
  - In der Nähe von
  - (Voll-)Geschwister

Ontologie

`owl:inverseOf`

- Verwendung

  ```
  P owl:inverseOf Q .
  ```

  - P ist die Inverse von Q, d.h. aus einer Verbindung X → Y mit P folgt die Gegenrichtung Y → X für Q

    ```
    X P Y . => Y Q X .
    ```

- Anmerkung
  - `owl:inverseOf` Ist symmetrisch

    ```
    owl:inverseOf rdf:type owl:SymmetricProperty .
    ```

- Beispiele
  - „Älter“ ist die Inverse von „Jünger“
  - „Nördlich von“ ist die Inverse von „Südlich von“

Ontologie

`owl:FunctionalProperty`

- Verwendung

  `P rdf:type owl:FunctionalProperty .`

  - Jede Ressource darf nur einmal Subjekt sein, wenn diese Property als Prädikat verwendet wird

    `X P Y . X P Z . => Y=Z`

- Beispiele
  - Geburtsort
  - Vater, Mutter

Kurze Pause

RDF Datentypen

XML Schema

- Ganze Zahlen (z.B. Integer)
  - `123` oder `"123"^^xsd:int`
- Kommazahlen (z.B. Double)
  - `1.5` oder `"1.5"^^xsd:double`
- Zeichenketten (z.B. String)
  - `"Es ist ein Text"` oder `"Es ist ein Text"^^xsd:string`
- Zeitbezogene Datentypen (z.B. DateTime)
  - `"2017-10-12T14:15:00+02:00"^^xsd:dateTime`

Räumliche Daten

- Prefixes

```
geo: <http://www.opengis.net/ont/geosparql#>
geof: <http://www.opengis.net/def/function/geosparql/>
```

- Literal (well known text)

```
"POLYGON( … )"^^geo:wktLiteral
```

Leere Knoten

Leere Knoten

- Beispiel
  - J. W. von Goethe starb am 22. März 1832 in Weimar.

Leere Knoten

- Beispiel
  - **J. W. von Goethe** starb am **22. März 1832** in **Weimar**.

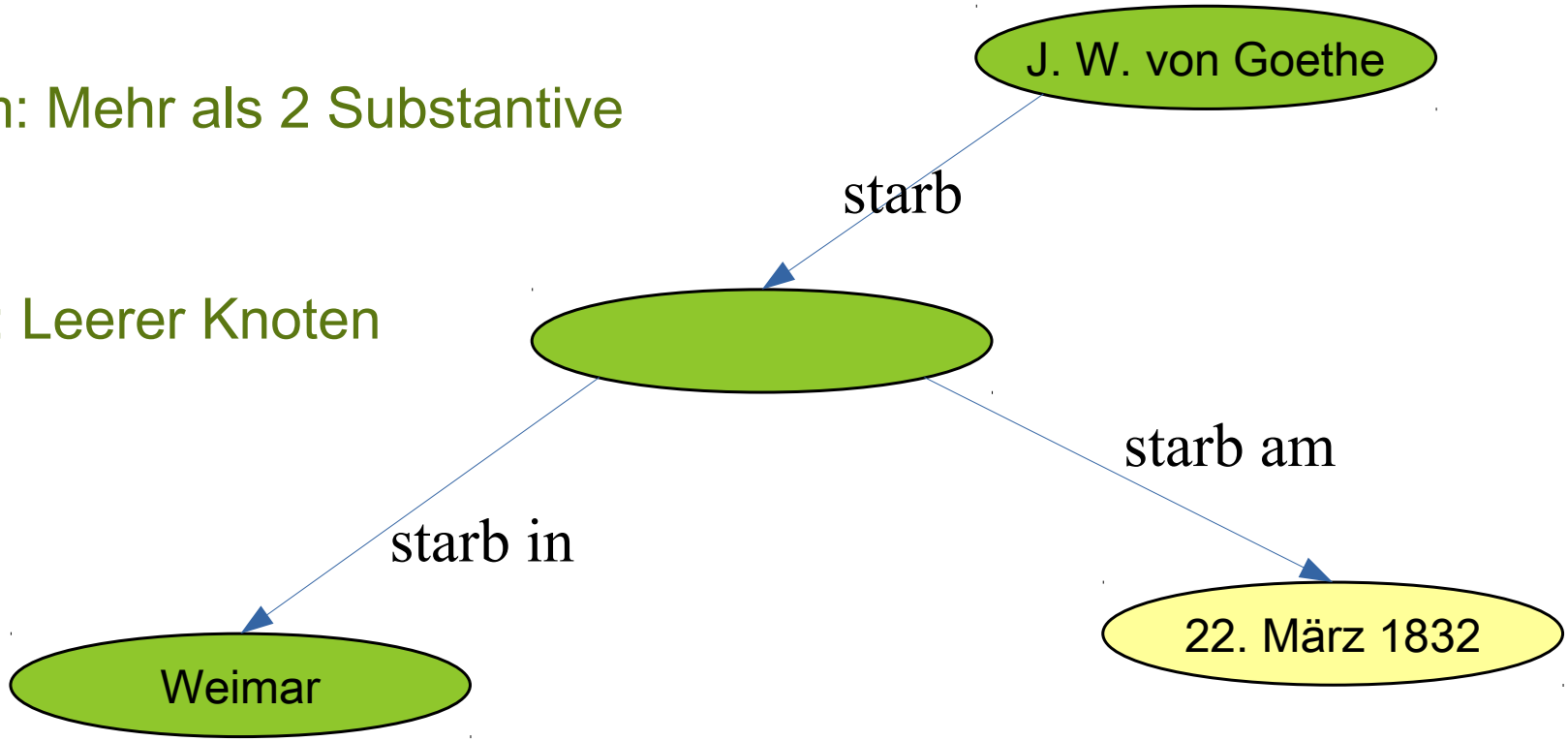
- Problem: Mehr als 2 Substantive

Leere Knoten

- Beispiel
  - **J. W. von Goethe** starb am **22. März 1832** in **Weimar**.
- Problem: Mehr als 2 Substantive
- Lösung: Leerer Knoten

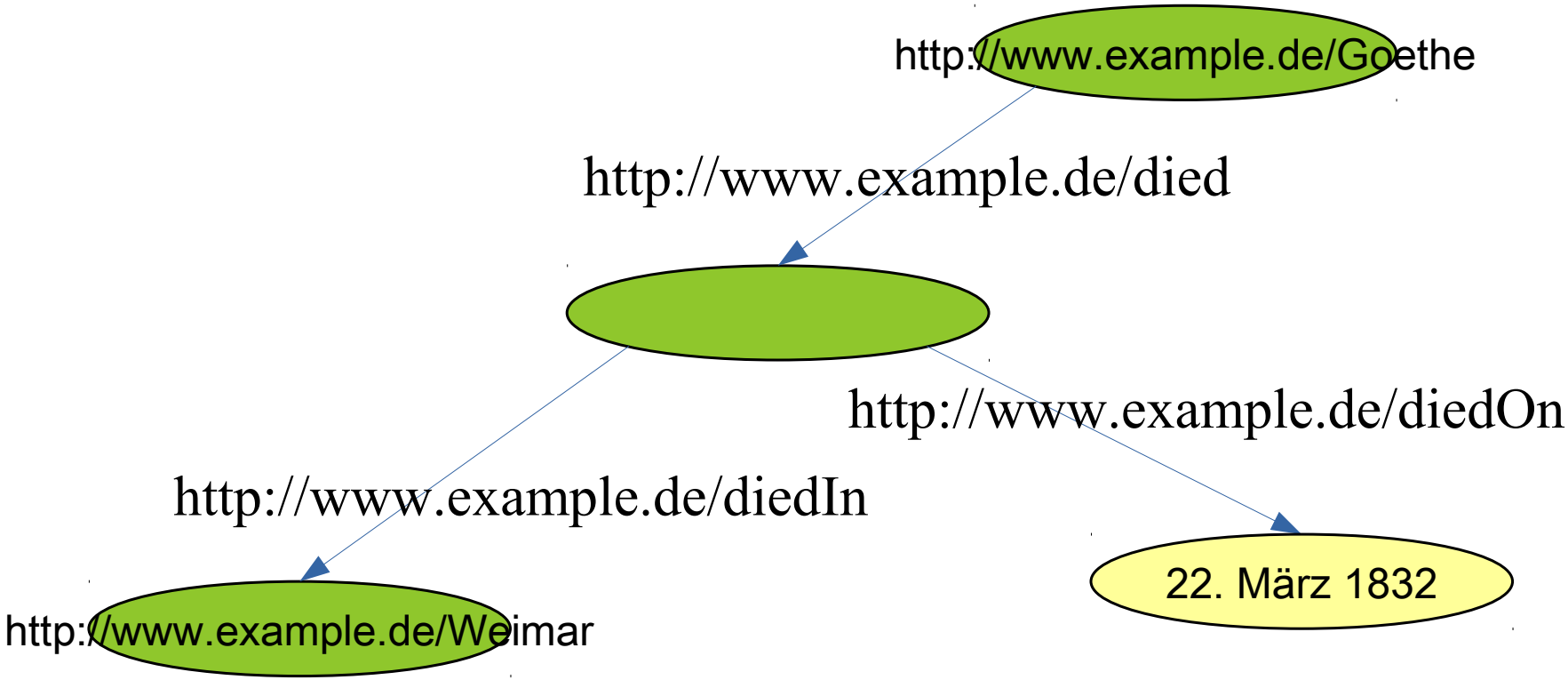
Leere Knoten

- Beispiel
  - **J. W. von Goethe** starb am **22. März 1832** in **Weimar**.
- Problem: Mehr als 2 Substantive
- Lösung: Leerer Knoten



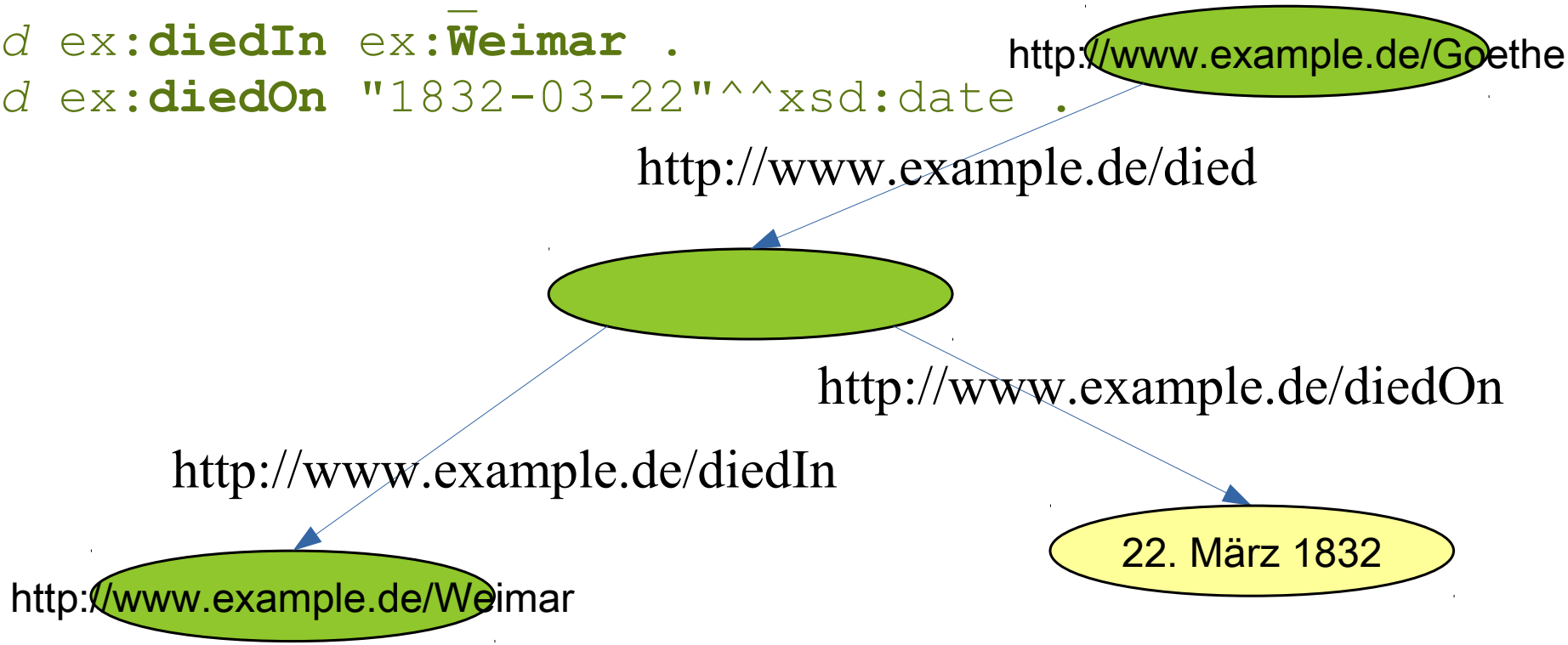
Leere Knoten

- Mit URIs



Turtle Syntax

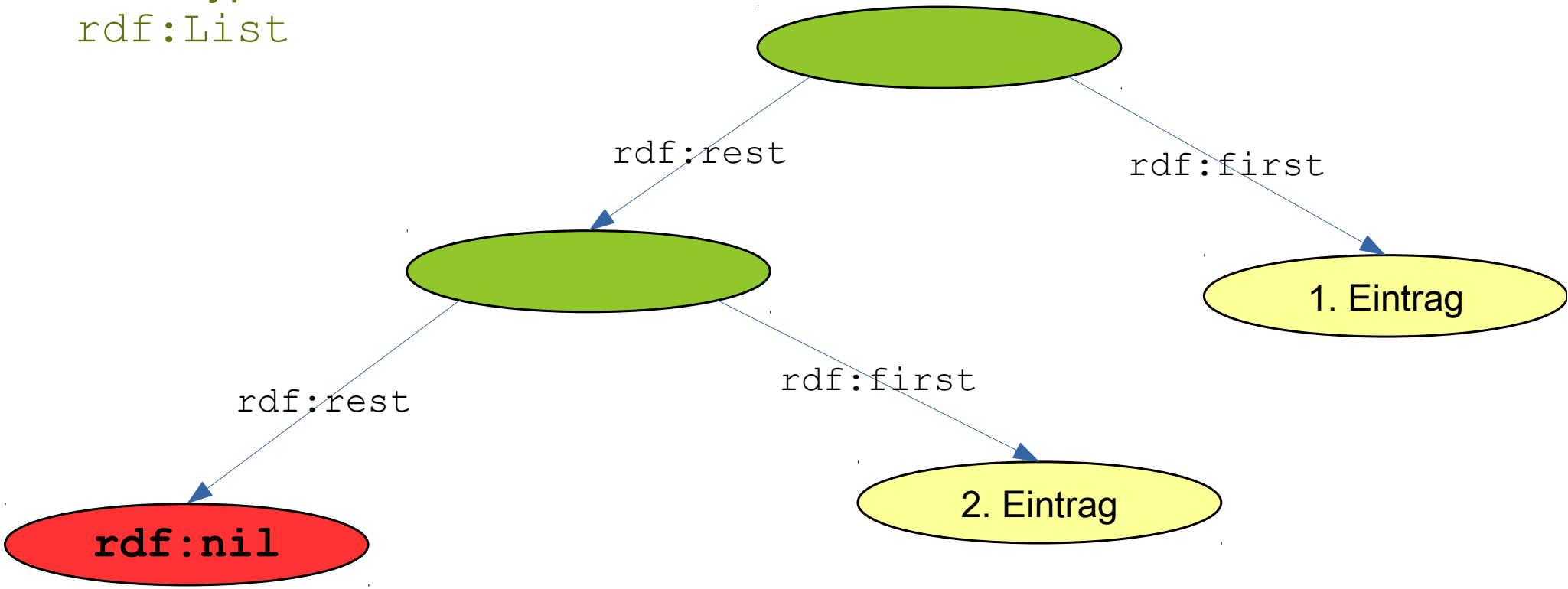
```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://www.example.de/> .

ex:Goethe ex:died _:id .
_:id ex:diedIn ex:Weimar .
_:id ex:diedOn "1832-03-22"^^xsd:date .
```



Listen

Listen

- Datentyp
  `rdf:List`



Daten im Triple Store modifizieren

Einfügen

- SPARQL Update

```
PREFIX …

INSERT {

    …

}

WHERE {

    …

}
```

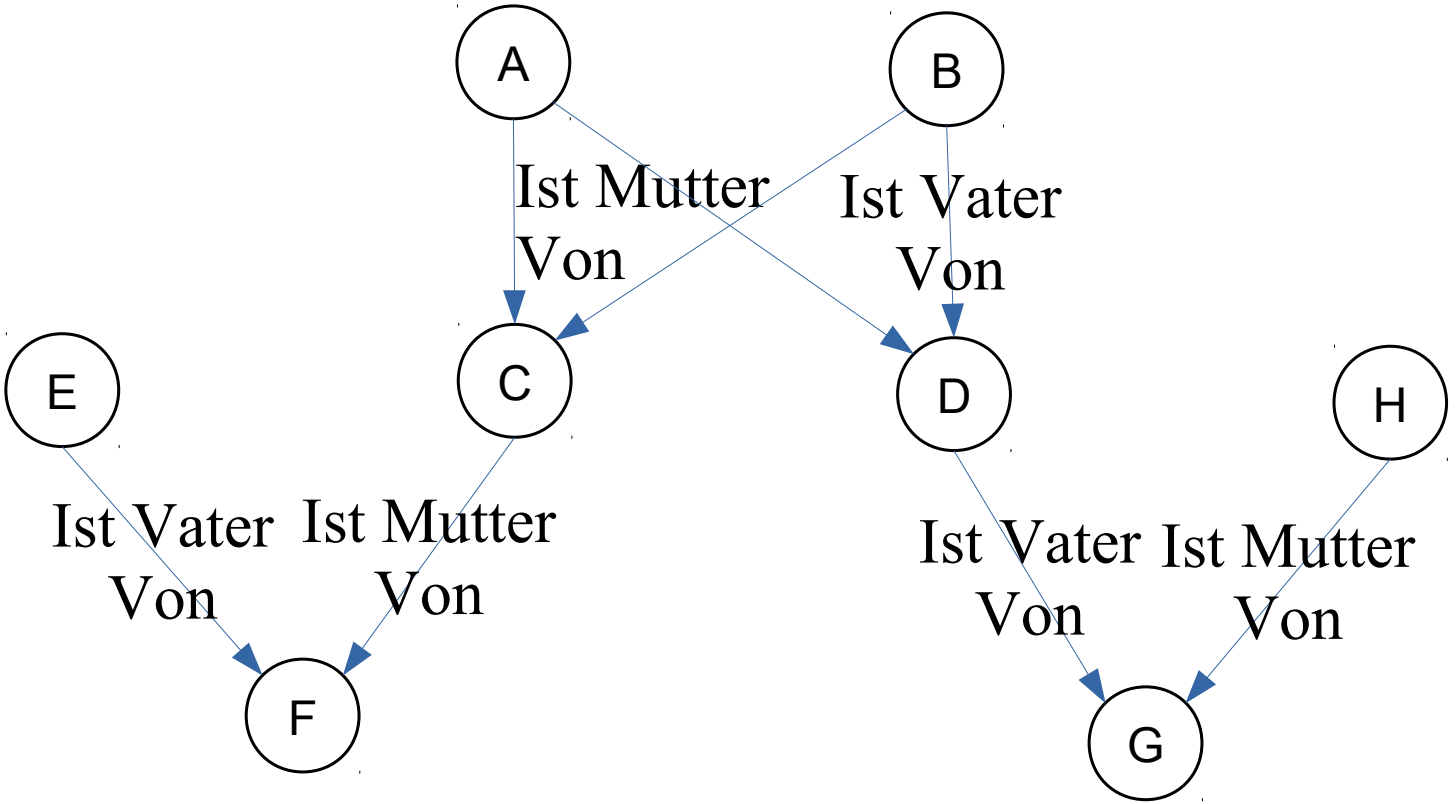
Einfügen

- Beispiel: Einzelnes Triple hinzufügen

```
INSERT {
   <subject> <predicate> <object> .
}
WHERE { }
```

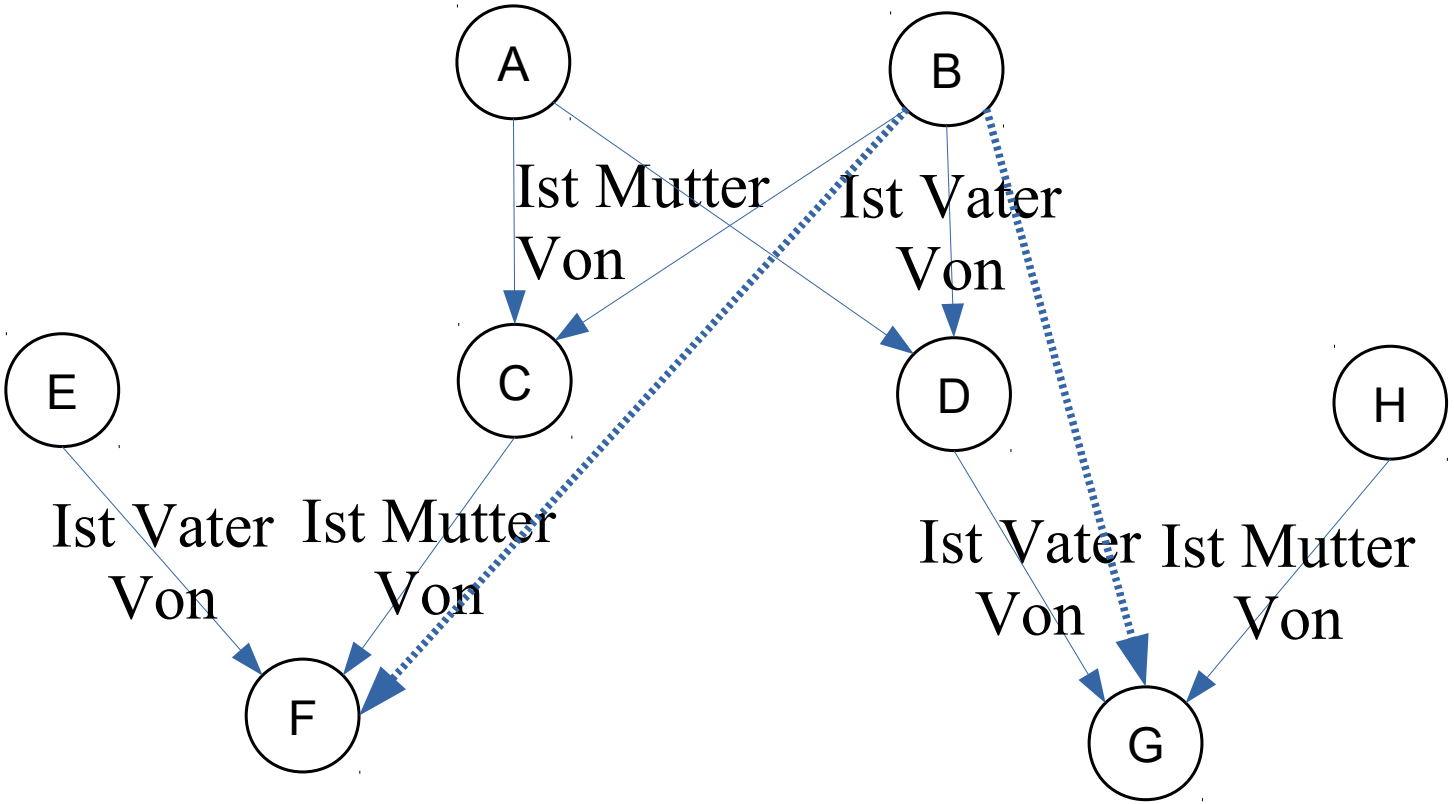
Einfügen

- Beispiel: Property Opa einfügen



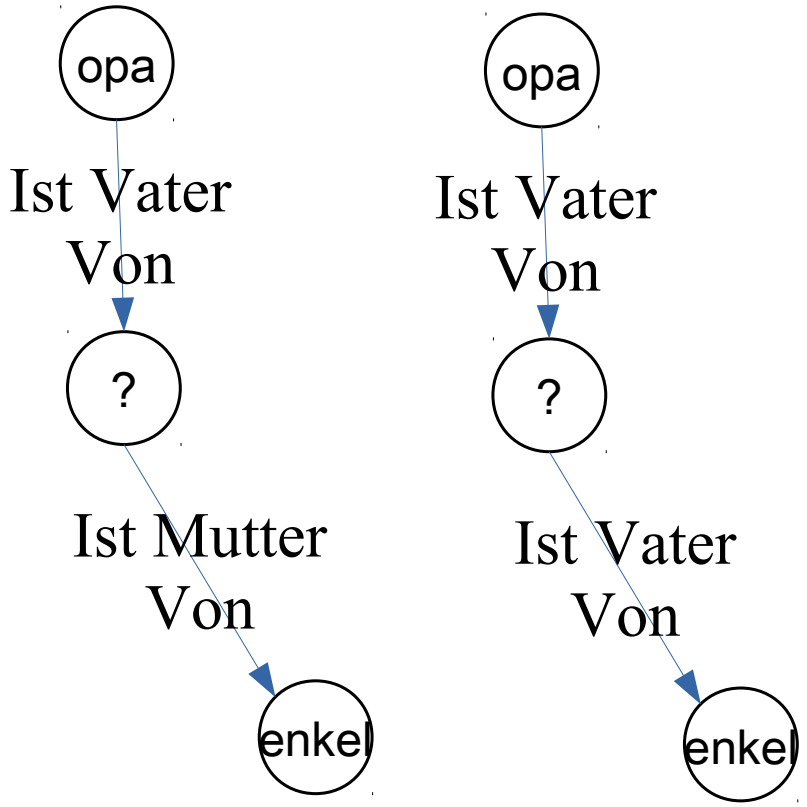
Einfügen

- Beispiel: Property Opa einfügen



Einfügen

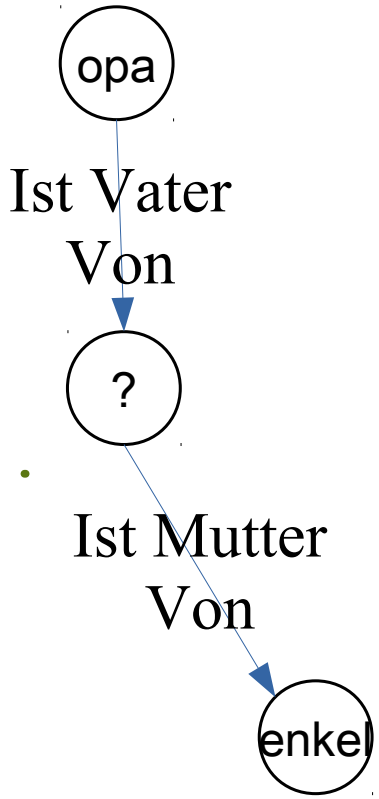
- Beispiel: Property Opa einfügen

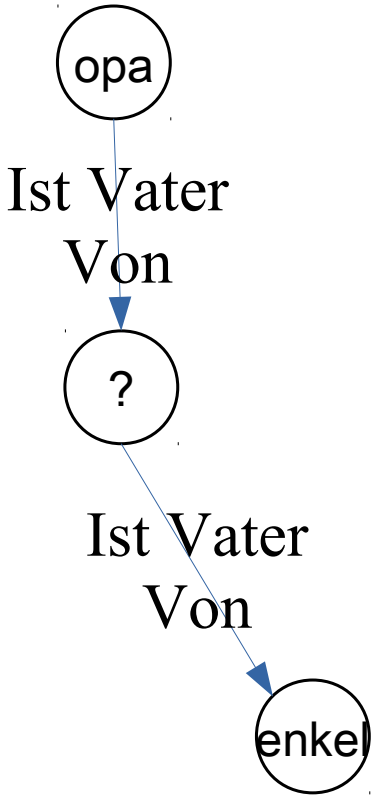


Einfügen

- Beispiel: Property Opa einfügen

```
PREFIX …

INSERT {

   ?opa idarit:Opa ?enkel .

}

WHERE {
   ?opa idarit:Vater ?x .
   { ?x idarit:Mutter ?enkel . }
   UNION
   { ?x idarit:Vater ?enkel . }
}
```





Löschen

- SPARQL Update

```
PREFIX …

DELETE {

    …

}

WHERE {

    …

}
```

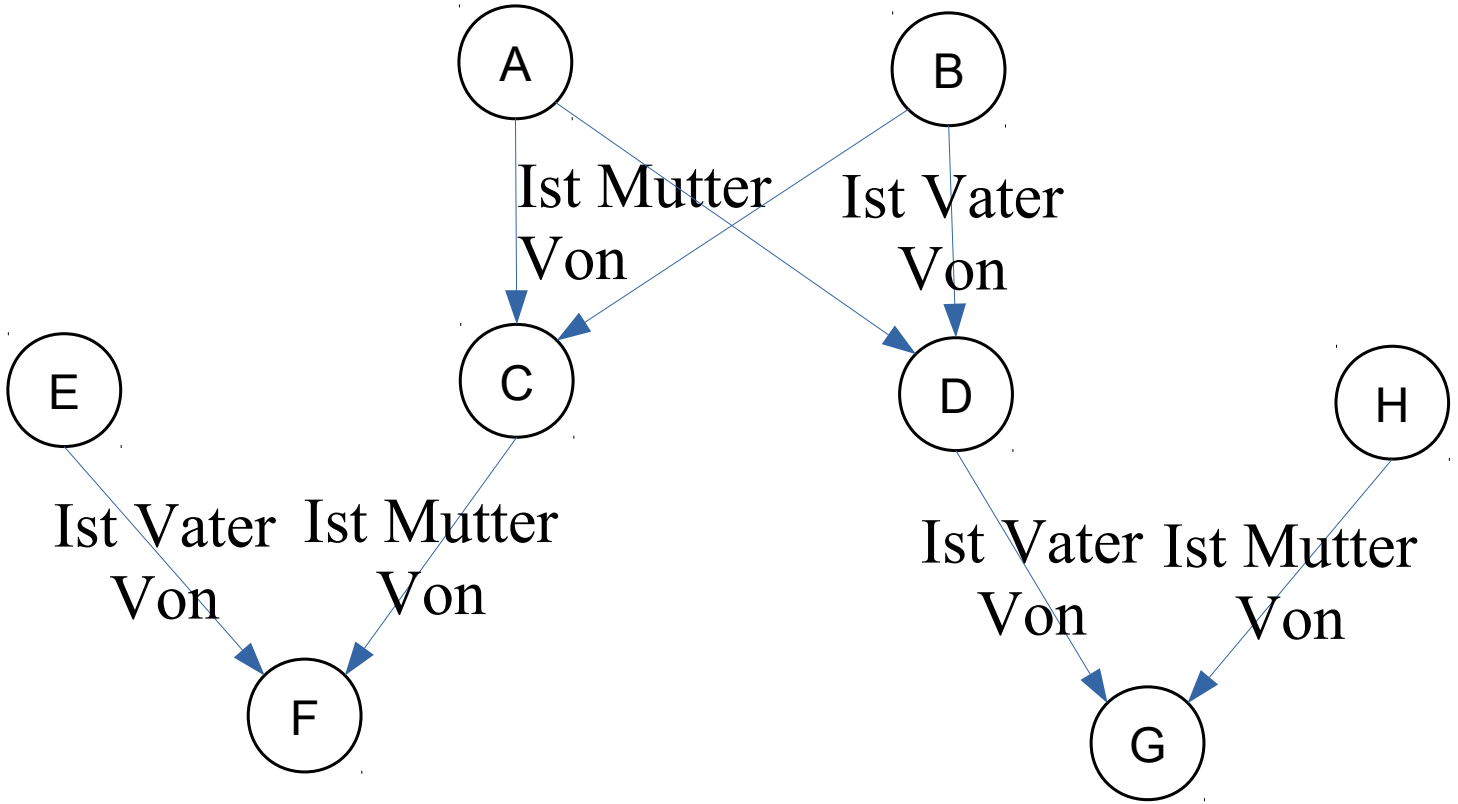
Löschen

- Beispiel: Einzelnes Triple entfernen

```
DELETE {
   <subject> <predicate> <object> .
}
WHERE { }
```

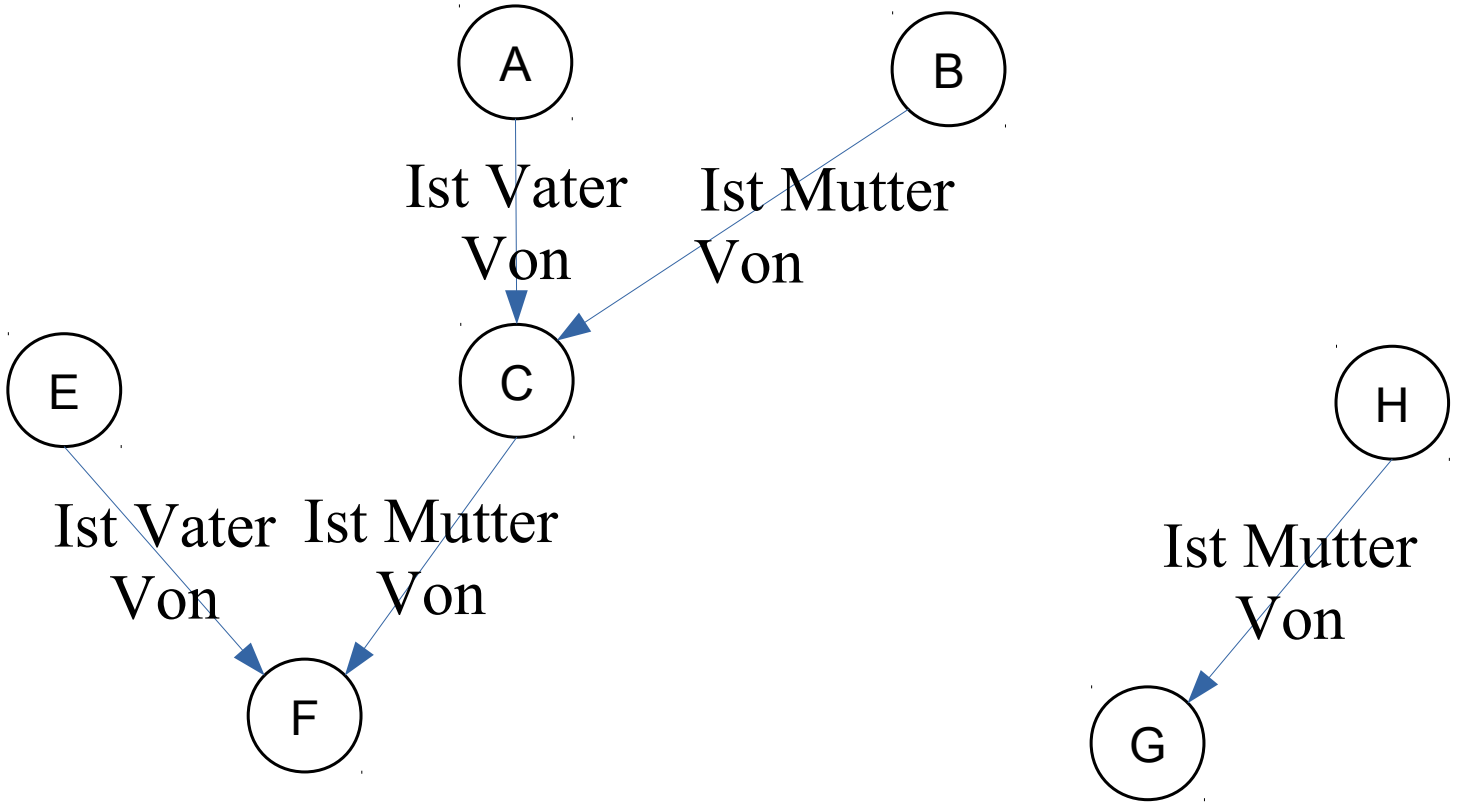
Löschen

- Beispiel: Daniel aus dem Stammbaum entfernen



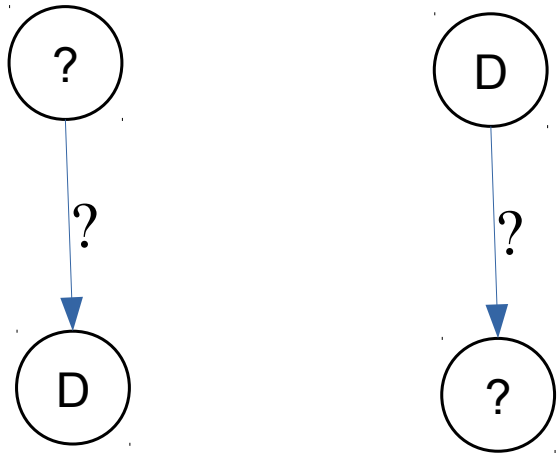
Löschen

- Beispiel: Daniel aus dem Stammbaum entfernen



Löschen

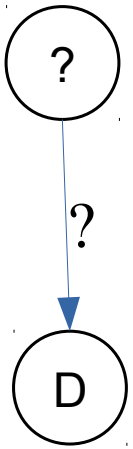
- Beispiel: Daniel aus dem Stammbaum entfernen

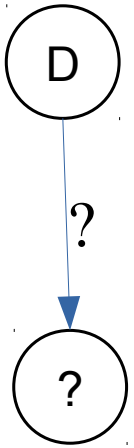


Löschen

- Beispiel: Daniel aus dem Stammbaum entfernen

```
PREFIX …

DELETE {
   ?s ?p idarit:Daniel .
}
WHERE {
   ?s ?p idarit:Daniel .
};

DELETE {
   idarit:Daniel ?p ?o .
}
WHERE {
   idarit:Daniel ?p ?o .
}
```





Vielen Dank
für die
Aufmerksamkeit