# Alternativa datamodeller och grafdata på webben

2022-05-02

Eva Blomqvist
eva.blomqvist@liu.se

Slides delvis utvecklade av:
Robin Keskisärkkä
robin.keskisarkka@liu.se

LINKÖPINGS UNIVERSITET

# Översikt

- Alternativ till relationsdatabaser
  - Key-value store
  - Wide column store
  - Grafdatabaser
- GraphQL
- Semantic Web och länkade data
- Använda webbdata

# Alternativ till relationsdatabaser

# Begränsningar hos relationsdatabaser

- *Closed-world assumption*
- Atomära attribut
- Objekt kan inte existera utan sina attribut
- Semantiken bakom databasen är inte explicit
- Komplicerat att uppdatera databasschemat
  - Ex. ändra 1:1 till 1:N eller N:M
- Kan inte hantera **riktigt** stora datamängder
- ...

# Hur hanterar vi avsaknad av information?

- Closed-world assumption (CWA)
  - Om det saknas i databasen så antar vi att det är falskt
- Open-world assumption (OWA)
  - Om det saknas i databasen så kan det vara antingen sant eller falskt

# Exempel från Harry Potter

| Id | Födelseår | Namn | Efternamn |
|---|---|---|---|
| 1 | 1980 | Harry | Potter |
| 2 | 1980 | Hermione | Granger |
| 3 | 1979 | Ron | ? |
| 4 | ? | Draco | Malfoy |

CWA: Draco har inget födelseår. Ron har inget efternamn.
OWA: Draco har ett födelseår och Ron har ett efternamn, men deras värden är okända.

# Begränsningar hos relationsdatabaser

- Closed-world assumption
- Atomära attribut
- Objekt kan inte existera utan sina attribut
- Semantiken bakom databasen är inte explicit
- Komplicerat att uppdatera databasschemat
  - Ex. ändra 1:1 till 1:N eller N:M
- Kan inte hantera **riktigt** stora datamängder
- ...

# Alternativa modeller

- Objektorienterade databaser
- Deduktiva databaser
- NoSQL
  - Key-value stores
  - Wide column store
  - Grafdatabaser
  - ...

# Vad är NoSQL?

- NoSQL: **N**ot **o**nly **SQL**
- Data representeras inte relationellt
- Konsistenskraven är lägre än för traditionella databaser
- Fokus ligger ofta på skalbarhet
- Har vanligtvis inget schema
  - ... men ofta kan scheman i viss mån "simuleras" med hjälp av middleware

# Key-value store

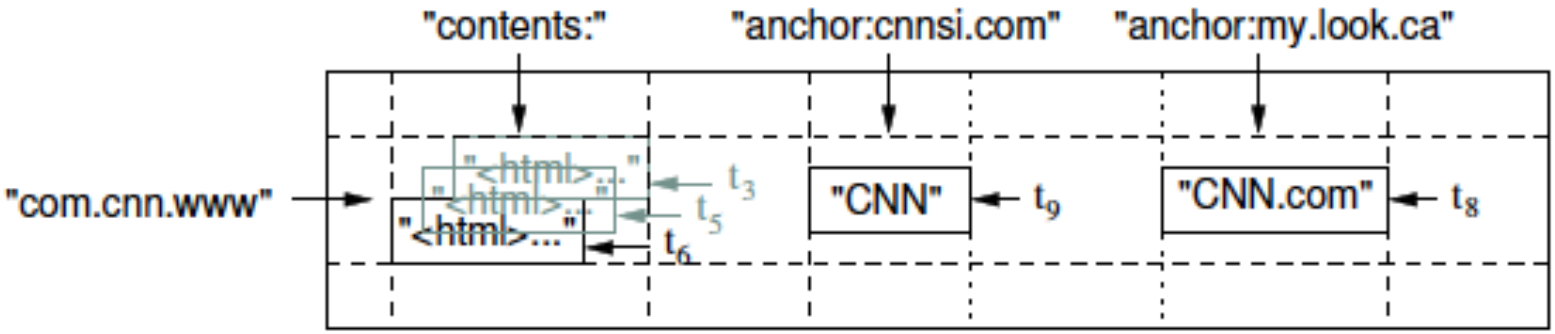
- Kan jämföras med dictionary i Python eller maps i andra språk
  - Amazon DynamoDB, Oracle NoSQL Database, Redis, MongoDB, Cassandra, ArangoDB mm.
- Fördelar
  - Kan distribueras och skalas upp relativt enkelt
  - Enkel datamodell (inget schema)
- Nackdelar
  - Datakonsistens
  - Konflikthantering
  - Inget schema!

# Wide columnstore

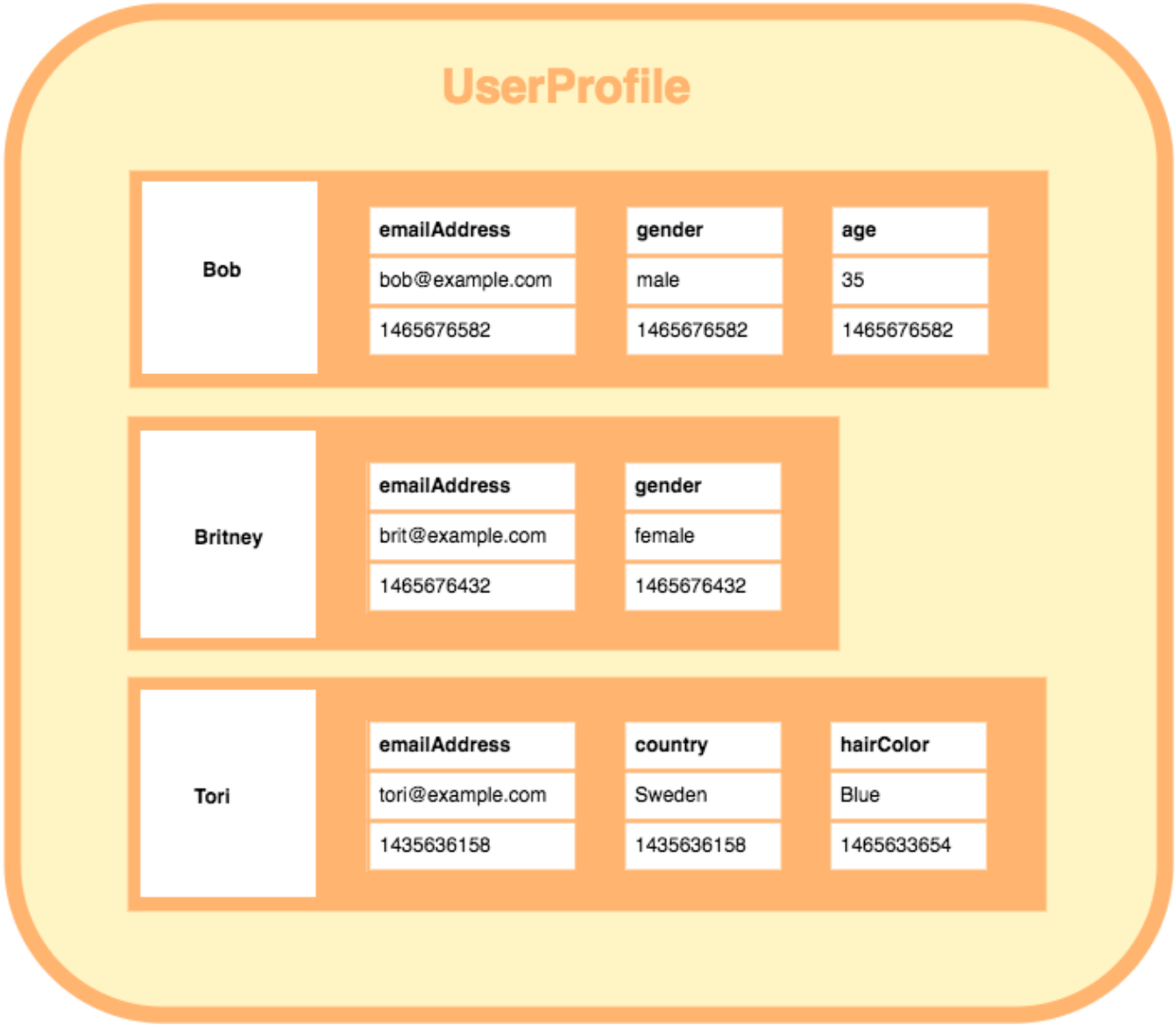
- Tvådimensionellt key-value store
- Tabellrad plus tabellkolumn utgör nyckeln för dataelement
- Fokus på stora mängder kolumner
- Exempel
  - Apache Cassandra, Amazon DynamoDB

# Exempel: Googles BigTable

- Utvecklades av Google 2004-2006




- "Rådata" från Googles crawlers (2006) ca 800TB, 1000 miljarder celler
- Distribuerat över serverkluster
  - Hierarki av tabeller för att hitta rätt data






https://database.guide/what-is-a-column-store-database/

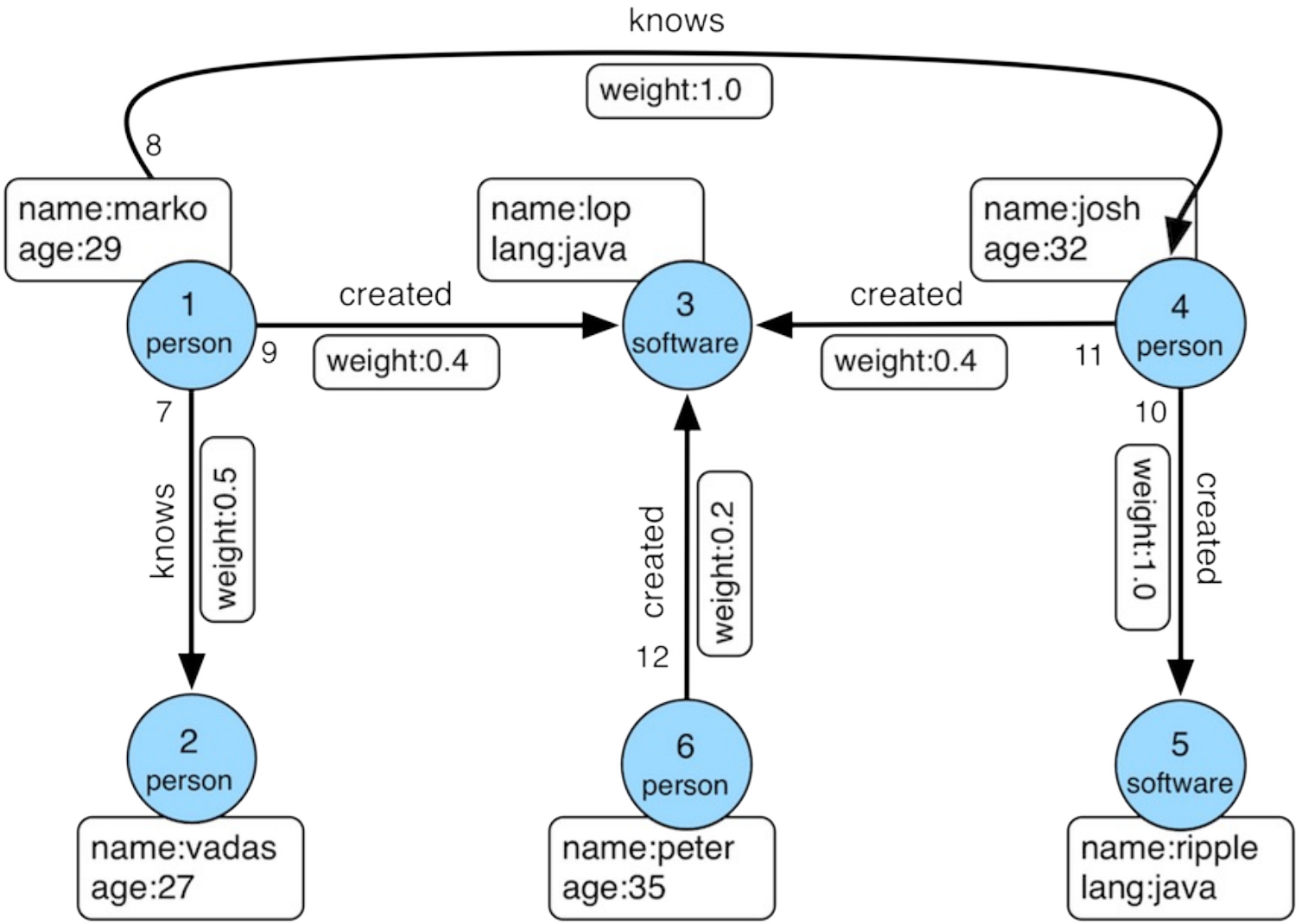
# Big Data: Apache Cassandra

*Some of the largest production deployments include* ***Apple's****, with over 75,000 nodes storing over* ***10 PB of data****,* ***Netflix*** *(2,500 nodes,* ***420 TB****, over 1 trillion requests per day), Chinese search engine* ***Easou*** *(270 nodes,* ***300 TB****, over 800 million requests per day), and* ***eBay*** *(over 100 nodes,* ***250 TB****).*

http://cassandra.apache.org/ (2020-12-16)

# Grafdatabaser

- Grafer är ofta ett naturligt sätt att beskriva data
  - Ex. sociala nätverk, webblänkar och relationer mellan dokument
- Typisk datamodell består av
  - Noder
  - Bågar (relationer)
  - Attribut på noder och bågar



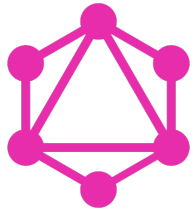



http://tinkerpop.apache.org/docs/current/reference/

# Exempel på grafdatabashanteringssystem

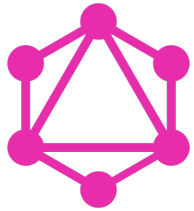
- Neo4J
- MarkLogic
- AnzoGraph DB
- ArangoDB
- OrientDB
- ... men ingen standardiserad datamodell eller gemensamt frågespråk

# GraphQL



- GraphQL är ett frågespråk för att hämta och ändra data på en server
- GraphQL kan i viss mån användas för att definiera scheman
- Kan ses som ett alternativ till REST API:er
- Hämtar endast den information som efterfrågas
  - Undviker *overfetching*
- GraphQL är inte bundet till någon specifik databas
- Mycket populärt och snabbt växande!

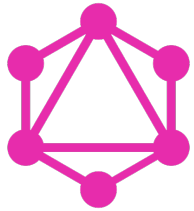
# GraphQL schema



```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}

type Post {
  title: String!
  author: Person!
}
```

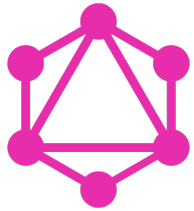
# GraphQL queries



```
query {
  allPersons {
    name
    posts {
      title
    }
  }
}
```

```
{
  "data": {
    "allPersons": [
      {
        "name": "Johnny",
        "posts": [
          {
            "title": "GraphQL is awesome"
          },
          {
            "title": "GraphQL vs. REST"
          }
        ]
      },
      # ...
    ]
  }
}
```

# GraphQL demo



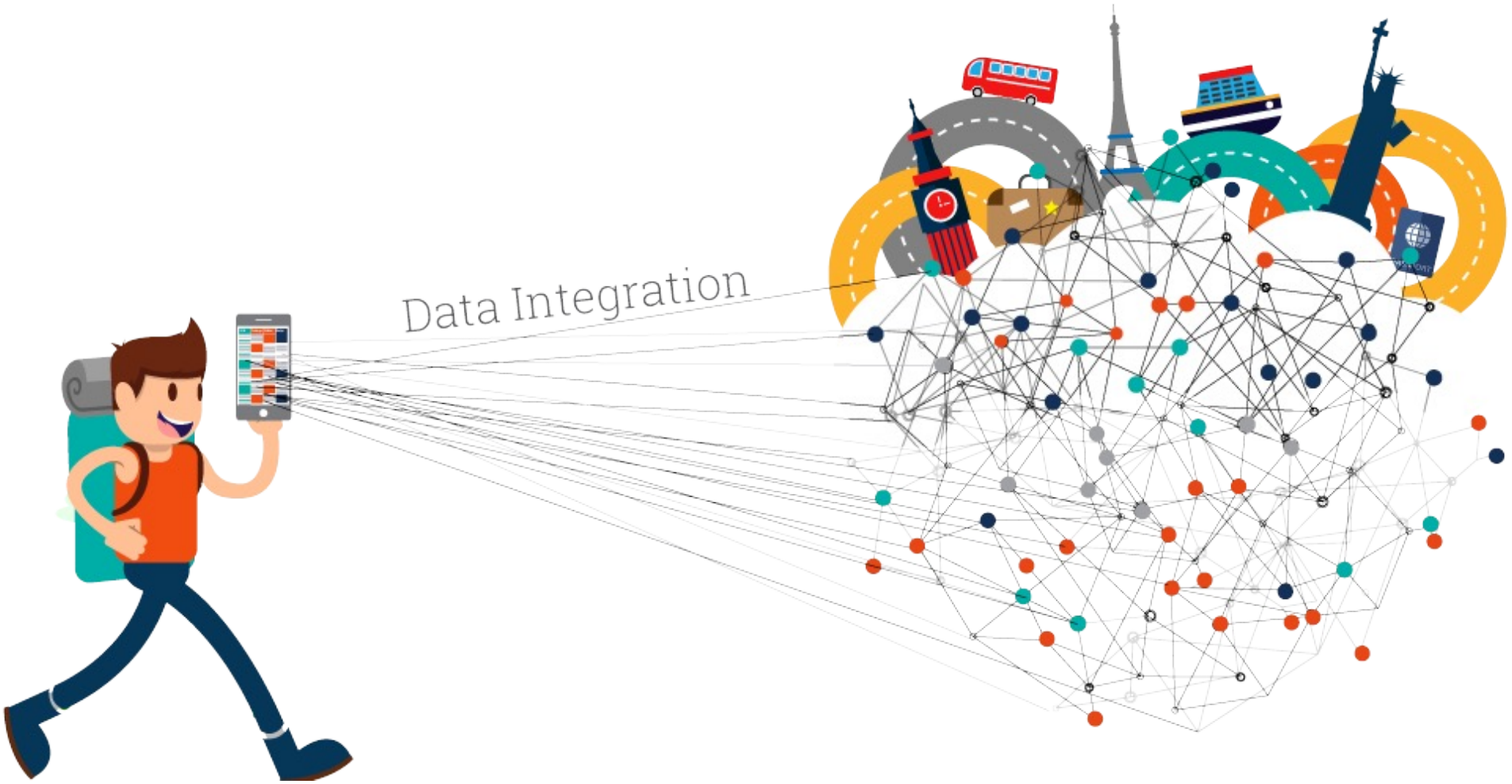
https://lucasconstantino.github.io/graphiql-online/

"Getting information off the Internet is like taking a drink from a firehose."

Mitchell Kapor

# Semantic Web

## RDF, OWL och länkade data





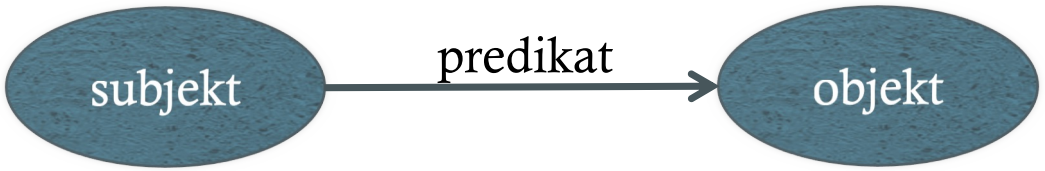

https://ontotext.com/linked-data-paths-smart-tourism-journey/

# The Semantic Web

- *Den semantiska webben*
- Adresser på webben representerar entiteter/resurser
- Länkar mellan entiteter/resurser beskriver specifika relationer
- Bildar gemensamt en semantiskt annoterad riktad graf

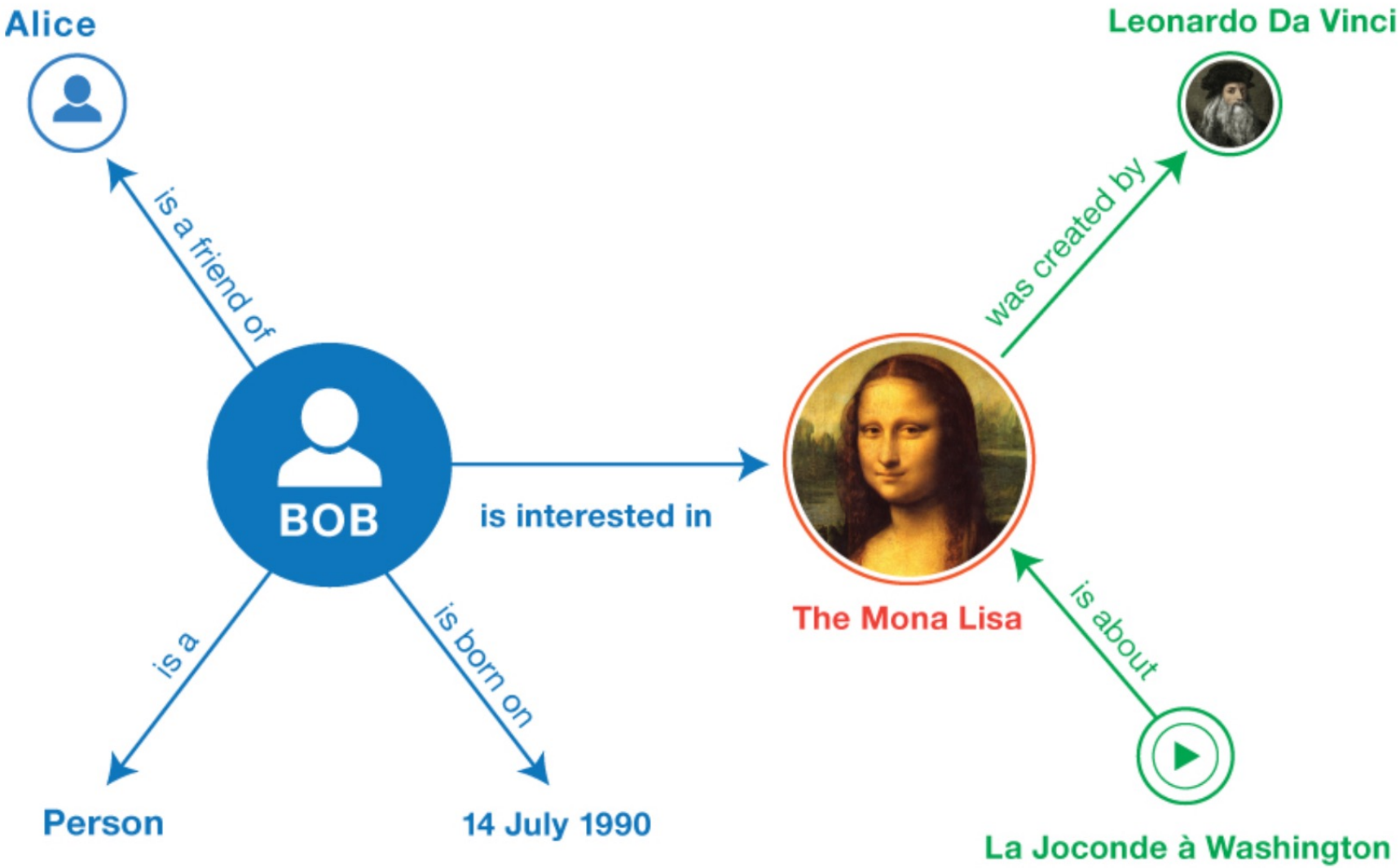
# RDF – Resource Description Framework

- Grafdatamodell för att beskriva data på webben
- En resurs är vad som helst som kan identifieras med en URI
  - Websidor, böcker, verkliga personer, ...
  - Relationerna mellan resurser är också resurser!
- Data representeras som tripplar



# RDF exempel

```
<Bob> <is a> <person> .
<Bob> <is a friend of> <Alice> .
<Bob> <is born on> <the 4th of July 1990> .
<Bob> <is interested in> <the Mona Lisa> .
<the Mona Lisa> <was created by> <Leonardo da Vinci> .
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa> .
```






https://www.w3.org/TR/rdf11-primer/

# Varför RDF?

- Ett sätt att prata om resurser på webben
- En gemensam enkel datamodell
- Ett universellt sätt att identifiera resurser (URI:er)
- ”Lågnivå”-integration av data
- Vi kan tala om att två resurser är samma sak genom att använda samma URI

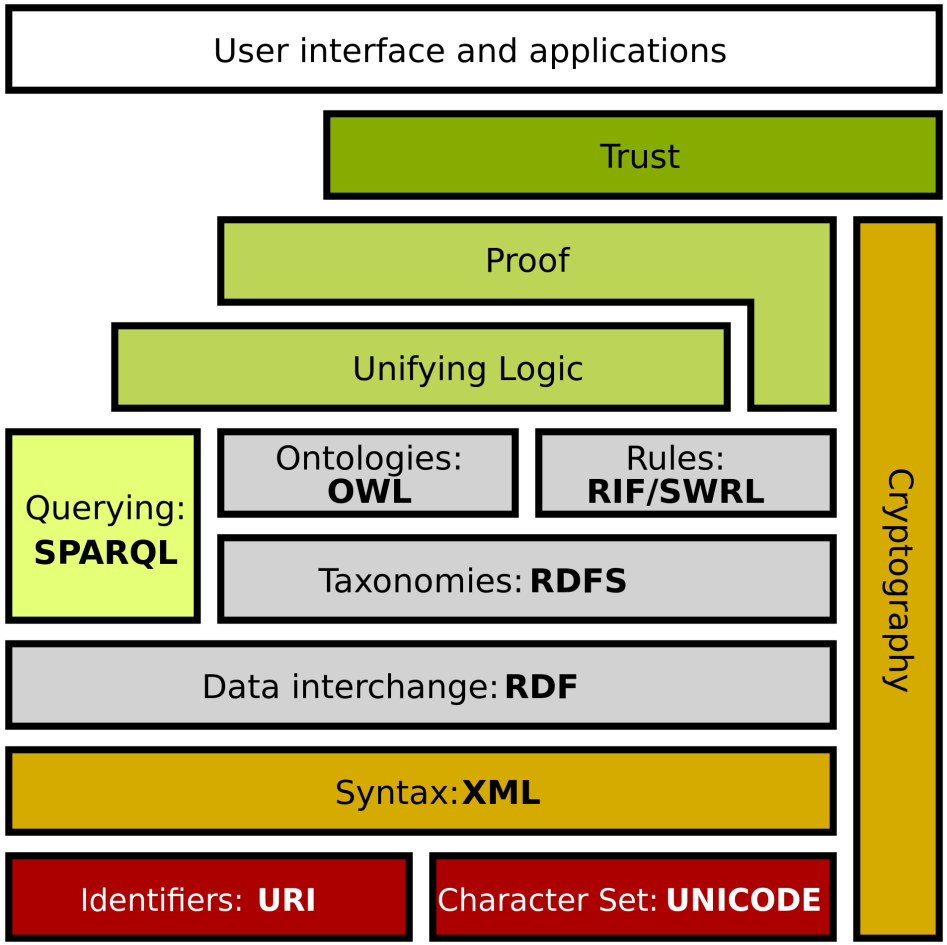
# Hur tillhandahålls RDF-data?

- Nedladdningsbara filer
  - Ladda ned och ladda in i ett eget program
- Data direkt på webben ”länkade data”
  - Slå upp URI:er och traversera grafen
    - Ex: https://dbpedia.org/page/Ada_Lovelace
  - Experimentella SPARQL-interface finns
- SPARQL-endpoints
  - Ställ SPARQL fråga mot en tjänst på webben
- Specialiserade API:er

# RDF stores

- Grafdatabaser designade för att stödja RDF
  - Exempel: Apache Jena, RDF4J, OpenLink Virtuoso, Blazegraph, Stardog, ...
- Skillnader mot andra grafdatabaser
  - RDF tillåter inte attribut på noder och bågar
  - Standardiserat frågespråk och datamodell förankrade i W3C
  - Stödjer oftast ontologier och inferens MEN uttrycksfulla modeller leder till sämre skalbarhet

# The Semantic Web Layers



# Ontologier

- Ontologier beskriver vår datamodell och vårt vokabulär
- Standardiserade ontologier
  - RDF
  - RDFS
  - OWL
  - ...

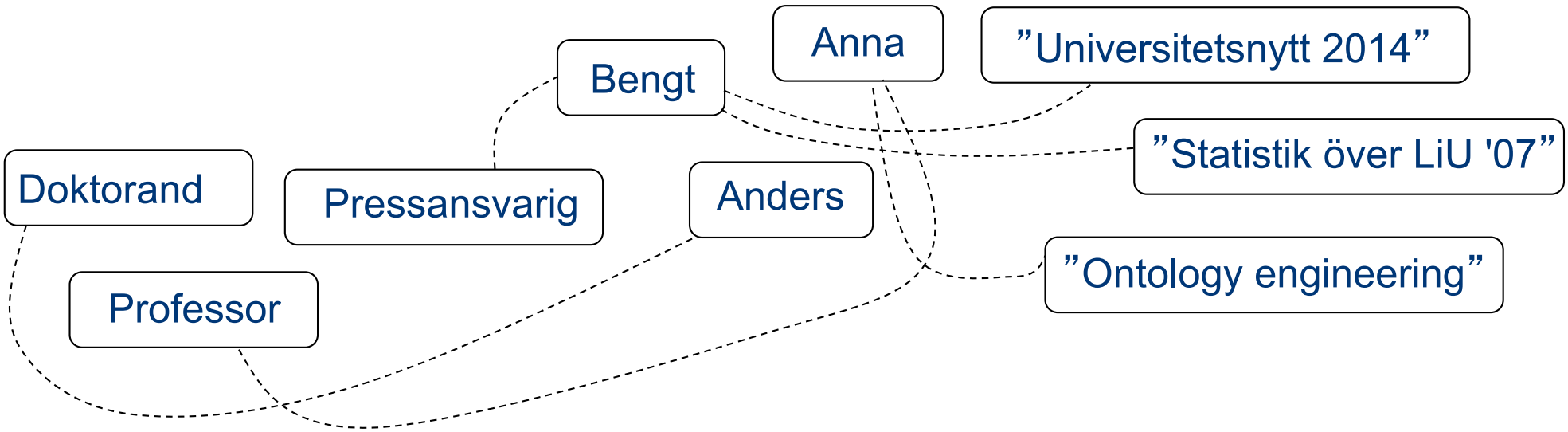
# RDF eller relationsdatabas?

- Relationsdatabaser har en mycket effektiv underliggande modell för normalstora data
- ... men semantiken är inte explicit!
- Exempel: ”Lista alla forskningsartiklar”

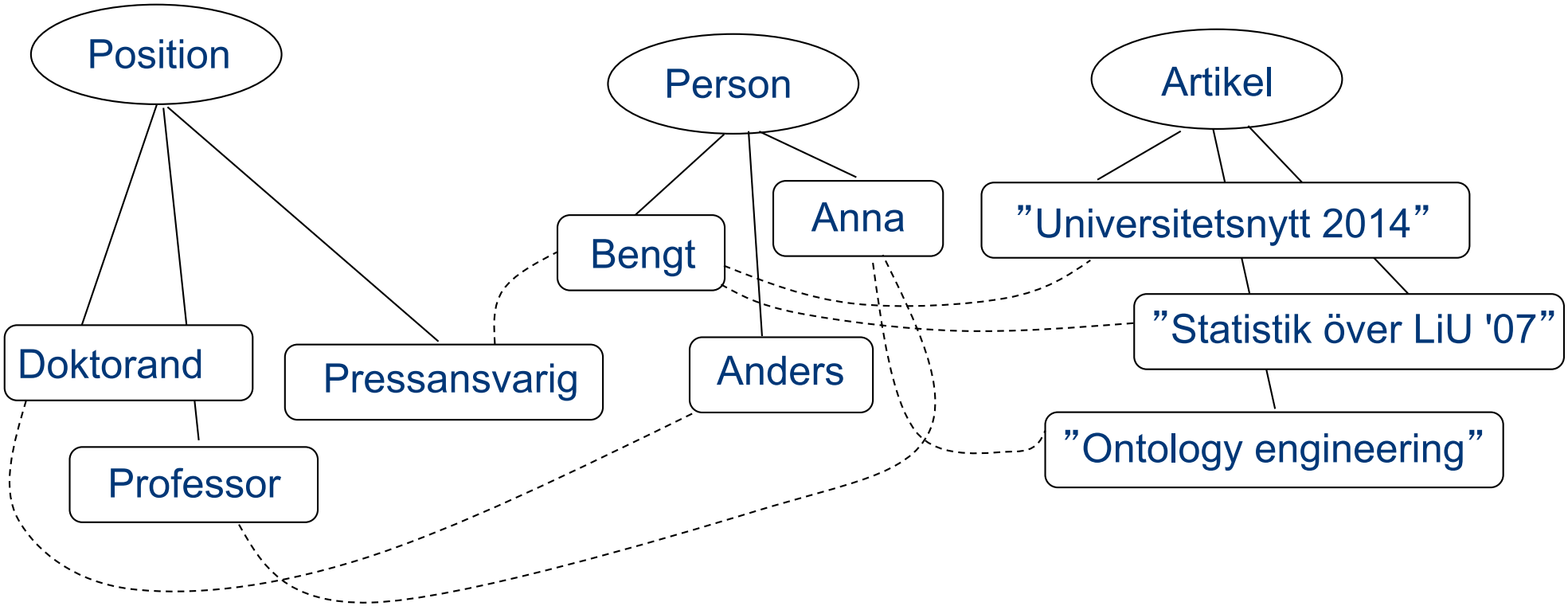
| Namn | Position |
|---|---|
| Anders | Doktorand |
| Anna | Professor |
| Bengt | Pressansvarig |

| Författare | Artikel |
|---|---|
| Bengt | ”Universitetsnytt 2014” |
| Anna | ”Ontology engineering” |
| Bengt | ”Statistik över LiU '07” |

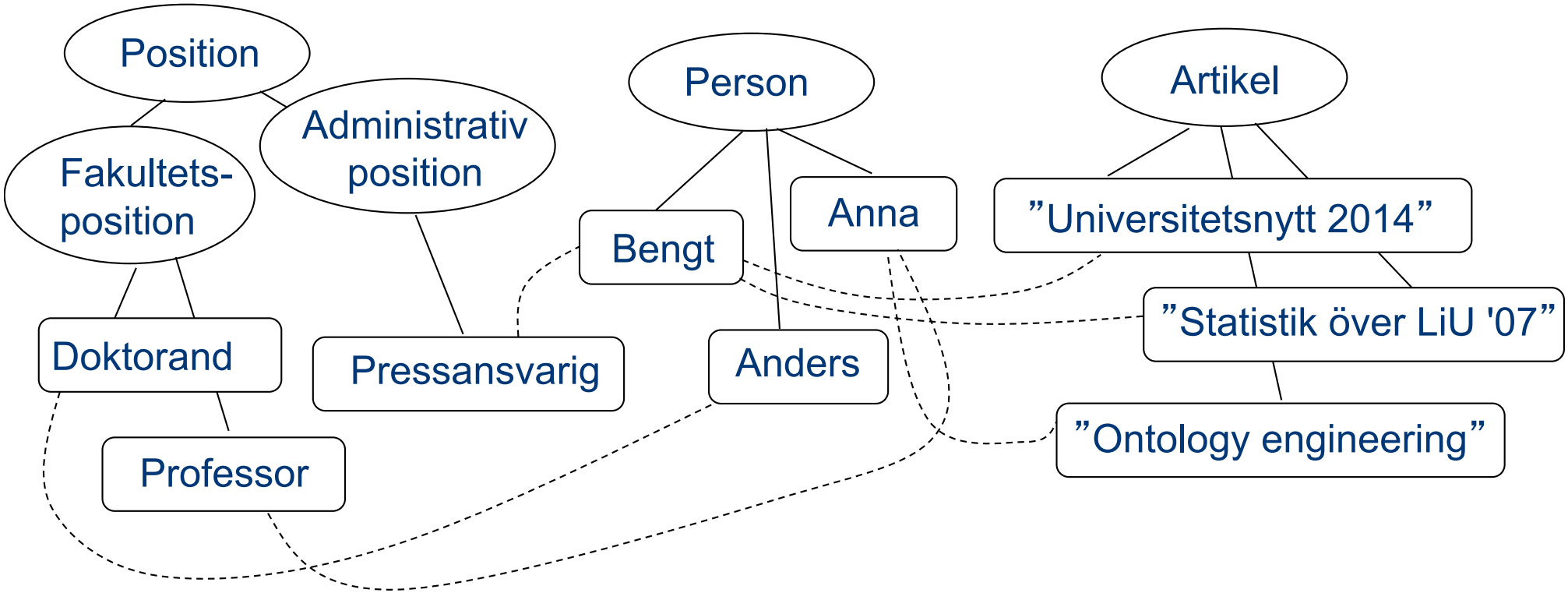
# RDF


Anna
”Universitetsnytt 2014”
Bengt
”Statistik över LiU '07”
Doktorand
Pressansvarig
Anders
”Ontology engineering”
Professor

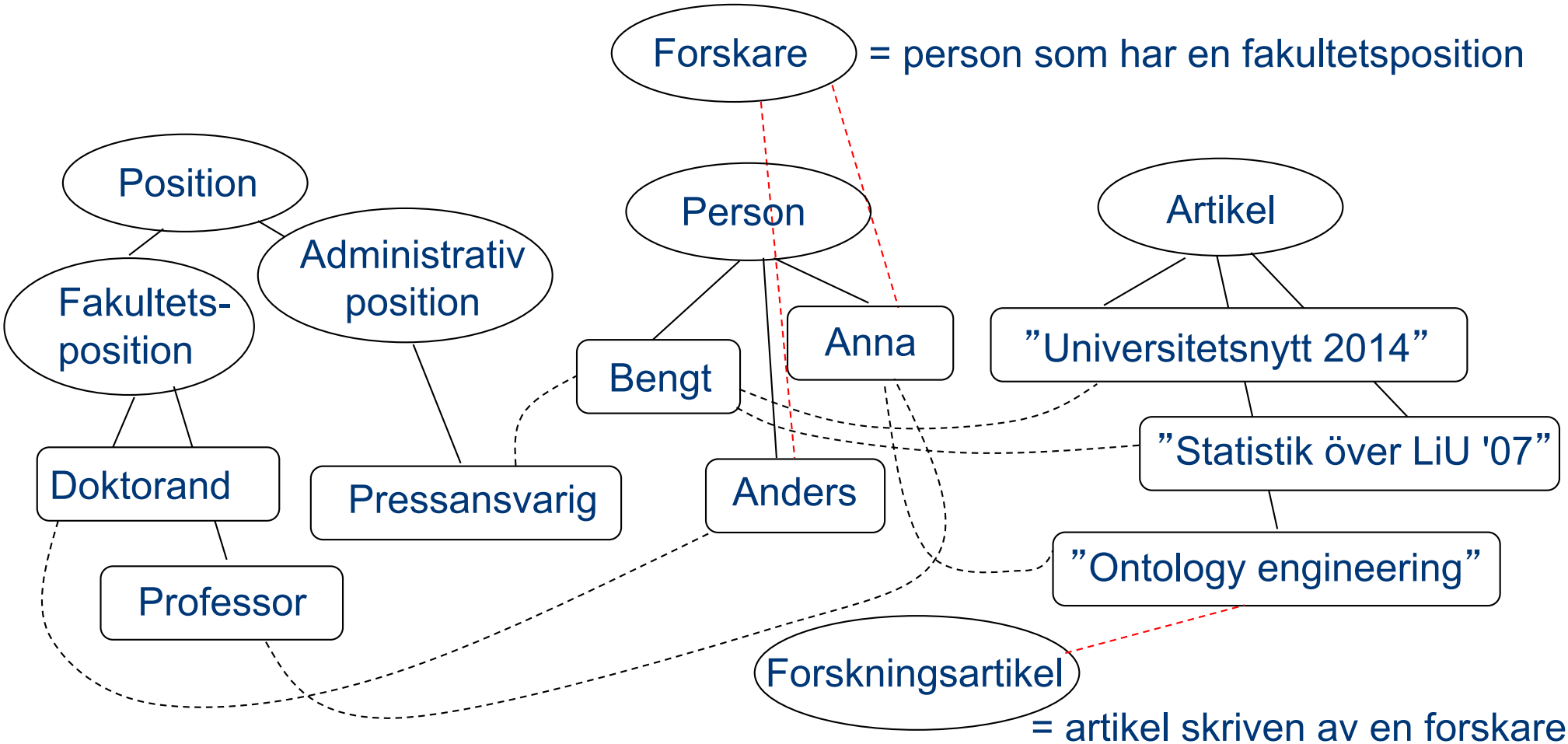
# RDF


Position
Person
Artikel
Anna
Bengt
”Universitetsnytt 2014”
”Statistik över LiU '07”
Doktorand
Pressansvarig
Anders
”Ontology engineering”
Professor

# RDF + RDF Schema (RDFS)



# RDF + RDFS + OWL


Forskare
= person som har en fakultetsposition
Position
Administrativ position
Fakultets-position
Person
Artikel
Anna
Bengt
”Universitetsnytt 2014”
”Statistik över LiU '07”
Doktorand
Pressansvarig
Anders
”Ontology engineering”
Professor
Forskningsartikel
= artikel skriven av en forskare

# Länkade data

- Data som publiceras på webben i enlighet med ett antal principer:
  1. Använd URI:er som namn på ”saker”
  2. Använd HTTP-URI:er så att man kan slå upp namnen
  3. När någon slår upp en URI, se till att returnera användbar information om URI:n
  4. Inkludera länkar till andra URI:er, så att man kan upptäcka mer
- Linking Open Data Project – LOD
  - Det första projektet som började publicera data på detta sätt
  - Nu mycket mer än ett projekt
  - Linked Open Data Cloud (https://lod-cloud.net/)

# Ett LOD exempel: DBPedia

- Strukturerad information extraherad från Wikipedia (främst ”infoboxes”)
- Utforska: skriv in en URI för en entitiet i din browser
  - ex. URI:n för Berlin (http://dbpedia.org/page/Berlin)
- Ställ frågor genom deras publika SPARQL endpoint
  - http://dbpedia.org/sparql
  - Alternativt interface: http://yasgui.org/

# Exempel 1: DBPedia

”Lista de 10 största länderna i världen med avseende på befolkningsmängd.”

| URI | Representerar |
|---|---|
| rdf:type | Predikat som pekar på resursens klass(er)/typ |
| dbo:Country | Resurs som representerar ett land (konceptet) |
| dbp:populationCensus | Predikat som pekar på en befolkningsmängd |
| rdfs:label | Textsträng som beskriver en resurs |

# SPARQL-fråga

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?country ?countryLabel ?population
WHERE {
    ?country rdf:type dbo:Country .
    ?country dbp:populationCensus ?population .
    ?country rdfs:label ?countryLabel
FILTER(lang(?countryLabel) = "en")
}
ORDER BY DESC(?population)
LIMIT 10
```

# Exempel 2: DBPedia

”Lista alla euroländer efter ländernas yta.”

| URL | Representerar |
|---|---|
| rdf:type | Predikat som pekar på resursens klass(er)/typ |
| dbo:Country | Resurs som representerar ett land (konceptet) |
| dbr:Euro | Resurs som representerar euro-valutan |
| rdfs:label | Textsträng som beskriver en resurs |
| dbo:currency | Predikat som pekar på en valuta |
| dbo:area | Predikat som pekar på en area |
| dbp:callingCode | Landsnummer |

# SPARQL-fråga

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?country ?countryLabel
WHERE {
    ?country rdf:type dbo:Country .
    ?country rdfs:label ?countryLabel .
    FILTER(lang(?countryLabel) = "en")
    ?country dbo:currency dbr:Euro .
    ?country dbo:area ?area .
    # snävare definiton av land kräver landsnummer
    ?country dbp:callingCode ?callingCode .
}
ORDER BY DESC(MAX(?area))
```

# Exempel 3: DBPedia

”Lista alla fotbollsspelare som har spelat/spelar för klubbar som har en arena med plats för fler än 40000, och som är födda i ett land med mindre än 10 miljoner invånare.”

| URL | Representerar |
|---|---|
| rdf:type | Predikat som pekar på resursens klass(er)/typ |
| dbo:Country | Resurs som representerar ett land (konceptet) |
| dbo:SoccerPlayer | Resurs som representerar fotbollsspelare (konceptet) |
| rdfs:label | Textsträng som beskriver en resurs |
| dbo:team | Predikat som pekar på ett lag |
| dbo:ground | Predikat som pekar på en stadion |
| dbp:capacity | Predikat som pekar på antal sittplatser |
| dbo:birthPlace | Predikat som pekar på födelseplats |
| dbp:populationCensus | Predikat som pekar på en befolkningsmängd |

# SPARQL-fråga

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?name ?country
WHERE {
    ?player a dbo:SoccerPlayer ;
            dbo:birthPlace ?country ;
            dbo:team ?team ;
            rdfs:label ?name .
     FILTER(lang(?name) = "en")
     ?country dbp:populationCensus ?population .
     FILTER(?population < 1000000)
     ?team dbo:ground ?ground .
     ?ground dbp:capacity ?capacity .
     # lägg till ett "rimligt" maxtak
     FILTER(?capacity > 40000 && ?capacity < 500000)
} LIMIT 1000
```

# Hur kan jag använda länkade data?

- Utnyttja ”the Web of Data” i dina system
  - Exempel: Hitta information om platser
  - Använd URI:er för att prata om resurser på webben
- Hur kommer jag åt data?
  - Nedladdning av datamängder
  - HTTP (kataloger, t.ex. https://datahub.io/)
  - SPARQL endpoints (http://sparqles.ai.wu.ac.at/availability)
- Andra API:er

# Exempel: använda LOD (via SPARQL) från R

- Använd paketet 'SPARQL'
  - https://cran.r-project.org/web/packages/SPARQL/SPARQL.pdf

- Skicka frågor till en SPARQL endpoint och hantera resultaten i R (t ex R-studio)
  - Demo…

# Summering

- Moderna storskaliga (webb-)applikationer ställer nya krav
  - Horisontell skalbarhet
  - Flexibla scheman
  - Länkar till externa data
- NoSQL är ett samlingskoncept för populära alternativ till relationsdatabaser
- GraphQL är ett växande alternativ till REST
- RDF är en datamodell specifikt utvecklad för webbdata
- Länkade öppna data gör att vi kan utnyttja webben som om det vore en stor databas

www.liu.se

LINKÖPINGS UNIVERSITET