Databázové systémy 2 (KIV/DB2)
RDF & SPARQL

Petr Včelák
vcelak@kiv.zcu.cz

8. dubna 2017

Obsah

1 Sémantický web

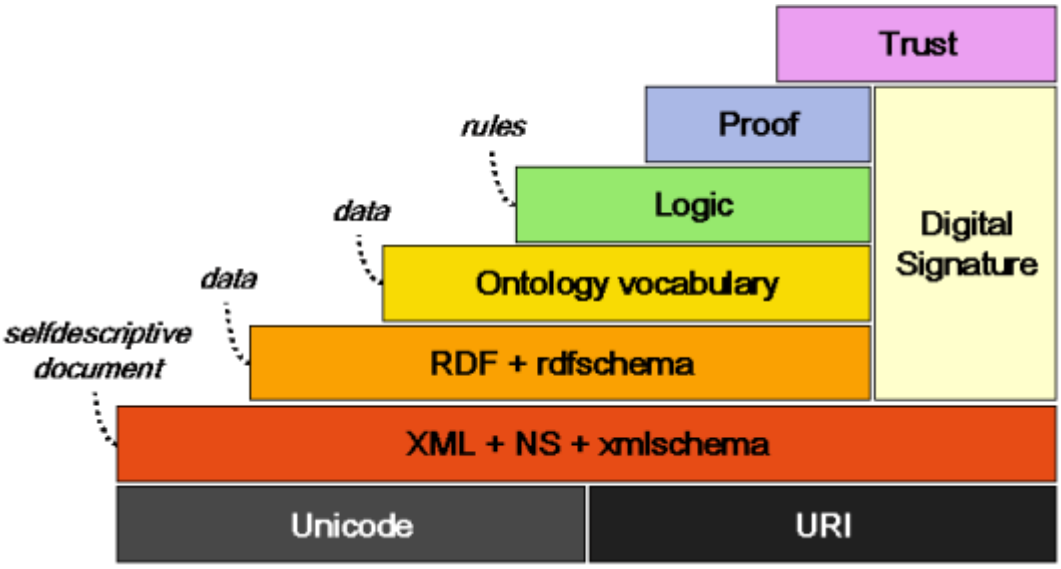
1.1 Úvod

- Semantic Web
- plány „Semantic Web Activity“:
  - informace na webu budou mít přesně daný význam,
  - informacím na webu může porozumět a zpracovávat je počítač,
  - počítače mohou integrovat a využívat informace z webu.
- popis zdrojů na webu:
  - popis informací o webové stránce (autor, datum vytvoření a změny, obsah, klíčová slova),
  - popis vlastností zboží v eshopech (cena, dostupnost), událostí,
  - popis obsahu a hodnocení,
  - popis pro vyhledávací stroje,
  - a další.
- Slyšeli jste o sémantickém webu?
  - Ne?
  - Ano, původní cíl sémantického webu lze považovat za historii.
- Přesto tato myšlenka nalezla své místo v jiných oblastech
  - Svět propojených (otevřených) dat.
  - Linked Data (LD) + Open Data
  - Linked Open Data (LOD).
  - V některých obměnách otevřenost dat nevylučuje nutnost přístupu pouze autorizovaných osob k některým zdrojům.

1.2 Vrstvy sémantického webu

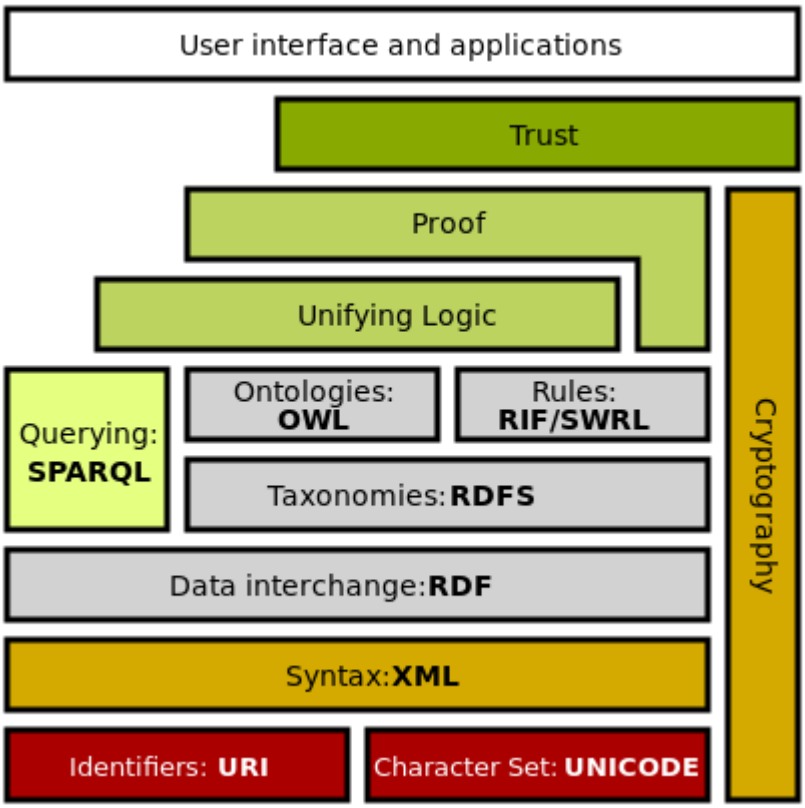
1. Unicode + Jednoznačné identifikátory (URI, IRI) – lokalizace a jméno
2. XML + jmenné prostory + XML Schema
3. RDF + RDF Schema
4. Ontologie (OWL)
5. Logika, usuzování, odvozování znalostí (tvrzení) – odvozovací pravidla (rules)
6. Důkaz a dokazatelnost (proof)
7. Důvěra (trust, digital signature, cryptography)
8. Uživatelské rozhraní a aplikace

Obrázek 1: Semantic Web Layers – 2001



Zdroj: W3C Semantic Web Activity. (2001, November 2). In *Semantic Web Kick-off*. Retrieved April 10, 2017, from https://www.w3.org/2001/12/semweb-fin/w3csw

Obrázek 2: Semantic Web Stack – 2015



Zdroj: Semantic Web Stack. (2015, September 17). In *Wikipedia, The Free Encyclopedia*. Retrieved April 10, 2017, from https://en.wikipedia.org/w/index.php?title=Semantic_Web_Stack

2 RDF

- RDF odkazuje na Resource Description Framework
  - „rámec popisu zdrojů“
- Je RDF datový model, slovník nebo rámec?
  - ano :–)
- autorem je organizace W3C.
- vytvořen pro popis zdrojů na webu.
- maximálně obecný:
  - aby mohl být čten a „pochopen“ strojem (počítačem),
  - RDF nebyl určen pro zobrazení lidmi,
  - RDF je zapsán v XML, který je označován RDF/XML,
    - XML zjednodušuje výměnu informací mezi různými typy aplikací i operačních systémů,
  - lze zapsat řadou jiných způsobů bez ztráty informace – notace
    - N-TRIPLE, TURTLE, a další včetně např. JSON
- silné stránky
  - integrace dat a informací (URI)
  - opakované použití dat a informací (jednotné identifikátory, slovníky)
  - strukturované nebo částečně strukturovaná data
  - oddělení datového modelování a syntaxe reprezentačního jazyka
  - začlenění zdrojů na webu na základě metadat popisujících jejich obsah
  - možnost klasifikace
  - možnost inference dat
  - reprezentace třídy i její instance stejným způsobem

2.1 Princip a popis RDF trojice

- RDF je založeno na atomickém prvku označovaném trojice (triple).
- Trojice popisuje vlastnost zdroje.
- Trojice se skládá ze tří částí:

  zdroj (resource, subjekt) je cokoliv co chceme popisovat a má jednoznačný identifikátor,
  vlastnost (property, predikát) je zdrojem, který má název je jednoznačně určen identifikátorem,
  hodnota (property value, objekt) je konkrétní hodnota (literál) nebo identifikátor jiného zdroje.
- Trojice tvoří vždy jedno platné tvrzení.
- Máme-li více tvrzení zapisujeme jako jako odpovídající počet trojic.
- Spojením trojic nám vzniká popis reálného světa v podobě orientovaného grafu.
  - zdroje jsou uzly
  - vlastnosti jsou hrany
- **Vše jsou trojice** (triples) resp. čtveřice (quads).
- Datové úložiště označujeme jako:
  - RDF store (RDF úložiště),
  - Triple-store (úložiště tripletů),
  - Graph database (grafová databáze), Quad-store (triplet uložen v rámci grafu).

2.2 Ilustrace převodu tvrzení do RDF

Příklad tvrzení: „Rozvrhovou akci v pondělí 9.20 vede Petr.“ lze zapsat do trojice:

zdroj = Rozvrhová akce v pondělí 9.20,

vlastnost = vede,

objekt = Petr.

Vidíme, že tam jsou i další tvrzení, které se vztahují k rozvrhové akci a blíže ji specifikují. Jak budou vypadat další tvrzení?

Další tvrzení přepsaná RDF trojicemi:

```
<zdroj>                          <vlastnost> <objekt>   .
=======================================================
<Rozvrhová akce v pondělí 9.20> <vede>      <Petr>     .
<Rozvrhová akce v pondělí 9.20> <den>       <pondělí>  .
<Rozvrhová akce v pondělí 9.20> <zacina>    <9.20>     .
<9.20>                           <hodin>     <9>        .
<9.20>                           <minut>     <20>       .
...
```

Výše uvedený zápis (bez prvních dvou řádek) se označuje N-TRIPLE a každá trojice je vždy na samostatném řádku ukončená znakem tečky. Zdroje jsou uzavřeny ve špičatých závorkách a text je v uvozovkách.

2.3 Identifikátory

- Výše uvedený zápis tvrzení byl ilustrační a v této podobě by neměl odpovídající přínos/význam.
- Zdroje musí být identifikovatelné.
- URI/IRI (Uniform Resource Identifier) je jednoznačný identifikátor
- scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
  - URN (Uniform Resource Name) – jméno zajistí identifikaci, ale nikoliv lokalizaci
    - např. ISBN urn:isbn:0-123-45568-9 odpovídá struktuře scheme:path
  - URL (Uniform Resource Locator) – identifikuje přístupovou metodu i místo v síti
    - https://zcu.cz/media/about/index.html
  - IRI (Internationalized Resource Identifier) – řetězec Unicode splňující pravidla dle RFC 3987
- Z důvodu možnosti propojení dat (v budoucnosti) na webu doporučeno použití schéma/protokolu HTTP(S).

V našem příkladě použijeme např. jmenný prostor *http://zcu.cz/rdf/* a zdroje i vlastnosti jím identifikujeme:

```
<zdroj>                <vlastnost>                 <objekt>              .
=============================================================================
<http://zcu.cz/rdf/1> <http://zcu.cz/rdf/vede>    "Petr"                .
<http://zcu.cz/rdf/1> <http://zcu.cz/rdf/den>     "pondělí"             .
<http://zcu.cz/rdf/1> <http://zcu.cz/rdf/zacina> <http://zcu.cz/rdf/2> .
<http://zcu.cz/rdf/2> <http://zcu.cz/rdf/hodin>   "9"                   .
<http://zcu.cz/rdf/2> <http://zcu.cz/rdf/minut>   "20"                  .
...
```

2.4 Datový typ a jazyk hodnoty

- Pro hodnoty lze definovat datový typ nebo jazyk.
- V N-TRIPLE a Turtle:
  - datový typ (XMLSchema – xsd)

```
"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"2012-04-18"^^<http://www.w3.org/2001/XMLSchema#date>
"2012-09-19T23:20:00+0200"^^<http://www.w3.org/2001/XMLSchema#dateTime>
```

  - jazyk (datový typ xsd:string)

```
"address"@en
"adresa"@cs
```

2.5 Slovníky a jmenné prostory

- Ilustrační příklad ukazuje popis několika tvrzení, ale rozumíme mu (pravděpodobně) pouze my.
- Pro zajištění shody a pochopení ostatními lidmi i stroji je popis (zatím) nevhodný, resp. nedostatečný.
- Vlastnosti, které jsme si zavedli jsou naše a *lokální*. Kdokoliv jiný se na ně podívá, nemusí pochopit jejich **správný význam** nebo je správně interpretovat:
  - zcu:den – den v měsíci? den v roce?
  - zcu:hodin – aktuální čas? čas události? 12/24 hod. – dopoledne nebo odpoledne?
  - zcu:minut – počet minut od/k čeho/čemu?
  - zcu:vede – vede projekt?
  - zcu:zacina – co/kde/proč začíná?

  Takto uvedené vlastnosti jsou „vytrženy z kontextu“, chybí nám kontext nebo spíše význam – **sémantika vlastností**.
- Zjednodušeně řečeno, proto vznikají **slovníky** nebo **ontologie**, které definují a současně dokumentují potřebný kontext, vztahy, doménu, obor hodnot, ...
- Vytvoříme-li si odpovídající slovník (resp. ontologii) a zveřejníme jej – data a informace může využít už i někdo další a mohou se sdílet na webu.
- **Problém**?!
  - Když si úplně každý vytvoří svůj vlastní slovník, pak bude výsledek nepoužitelný
    - data budou sdílená, ale možnosti využití a interpretace ostatními budou minimální.
- Řešení – existují základní RDF slovníky a ontologie, které přináší rámec jak nad RDF popisovat zdroje jednotným způsobem:

  RDF slovník přináší základní prvky;
  prefix: *rdf*,
  jmenný prostor: http://www.w3.org/1999/02/22-rdf-syntax-ns#
  - třídy,
  - vlastnosti,
  - hodnoty,
  - rdf:type – určení, že je popisovaný zdroj nějakého typu/třídy.

  RDF Schema (RDFS) rozšíření základního RDF slovníku;
  prefix: *rdfs*,
  jmenný prostor: http://www.w3.org/2000/01/rdf-schema#
  - umožňuje popsat pro aplikace specifické třídy a vlastnosti
  - podobnost s objektově orientovaným programováním (OOP)
    - lze vytvářet hierarchii tříd (sub-class) a vlastností (sub-property)
    - zdroje mohou být definovány jako instance tříd
  - popisuje strukturu dat

  Web Ontology Language (OWL)

  Web Ontology Language 2 (OWL2) rodina jazyků pro reprezentaci znalostí při tvorbě ontologií.
  - popisuje sémantické vztahy
  - *owl:sameAs* – pro popis, že nějaké dvě „věci“ jsou totéž
    - užitečné při spojování více schémat nebo zdrojů dat –> Linked Data
- Závěr: RDF definuje jak psát popis a OWL definuje co psát.

2.6 Používané notace RDF dat

Pro účely přehlednějšího a stručnějšího (úspornějšího) zápisu se používají prefixy i další notace.
Pro stejný příklad popisu pondělní rozvrhové akce použijeme další zápisy.

1. Přímo výše uvedený příklad zapíšeme v notaci *Turtle*:

```
<http://zcu.cz/rdf/1>
        http://zcu.cz/rdf/:vede     "Petr"     ;
        http://zcu.cz/rdf/:den      "pondělí"  ;
        http://zcu.cz/rdf/:zacina   http://zcu.cz/rdf/2      .

<http://zcu.cz/rdf/2>
        http://zcu.cz/rdf/:hodin    "9"        ;
        http://zcu.cz/rdf/:minut    "20"       .
```

   - Tvrzení patřící stejnému subjektu oddělujeme **středníkem**, více hodnot u stejné vlastnosti oddělujeme **čárkou** a za posledním tvrzením je **tečka**.

2. Ke zvolenému jmennému prostoru nadefinujeme prefix *zcu* a data ve výše uvedené notaci *Turtle* budou:

```
@prefix zcu: <http://zcu.cz/rdf/> .

<http://zcu.cz/rdf/1>
        zcu:vede    "Petr"     ;
        zcu:den     "pondělí"  ;
        zcu:zacina  zcu:2      .

<http://zcu.cz/rdf/2>
        zcu:hodin   "9"        ;
        zcu:minut   "20"       .
```

   - Ve formátu Turtle může být na začátku deklarace prefixů začínající znakem zavináče a slovem prefix, za nímž následuje vlastní prefix a za dvojtečkou úplné URI/IRI jmenného prostoru za kterým je tečka.

3. Na závěr ještě stejný zápis v provedení notace *RDF/XML*:

```
<rdf:RDF xmlns:zcu="http://zcu.cz/rdf/">
  <rdf:Description rdf:about="http://zcu.cz/rdf/1">
    <zcu:vede>Petr</zcu:vede>
    <zcu:den>pondělí</zcu:den>
    <zcu:zacina>
      <rdf:Description rdf:about="http://zcu.cz/rdf/2">
        <zcu:hodin>9</zcu:hodin>
        <zcu:minut>20</zcu:minut>
      </rdf:Description>
    </zcu:zacina>
  </rdf:Description>
</rdf:RDF>
```

   - Používá se jmenný prostor *rdf*, který je rezervovaný. Prefixy jsou definovány v kořenovém elementu *rdf:RDF*.

- Příklad v základní RDF/XML notaci včetně použitého RDF jmenného prostoru:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:zcu="http://zcu.cz/rdf/">
  <rdf:Description rdf:about="http://zcu.cz/rdf/1">
    <zcu:vede>Petr</zcu:vede>
    <zcu:den>pondělí</zcu:den>
    <zcu:zacina rdf:resource="http://zcu.cz/rdf/2" />
  </rdf:Description>
  <rdf:Description rdf:about="http://zcu.cz/rdf/2">
    <zcu:hodin>9</zcu:hodin>
    <zcu:minut>20</zcu:minut>
  </rdf:Description>
</rdf:RDF>
```

- Uvedené serializace RDF dat jsou vzájemně převoditelné – bezztrátově.
  - Obvykle záleží na aplikaci, v jaké notaci data vyžaduje.
  - Lze serializovat také do formátu JSON.

2.7 Přidání tvrzení

- Nová tvrzení/trojice stačí k původním přidat.

```
1 <zdroj>                          <vlastnost> <objekt>       .
2 ==========================================================
3 <Petr>                            <je>       <Jméno>        .
4 <Petr>                            <je>       <Osoba>        .
5 <Petr>                            <je>       <Muž>          .
6 <pondělí>                         <je>       <Den v týdnu>  .
7 ...
```

  Tyto trojice mohou pocházet např. z jiného zdroje, slovníku nebo ontologie.

- Lze se dotazovat u více zdrojů současně.
  - Distribuované prostředí webu – SPARQL Endpoint.
- Pokud zdroj používá jiné identifikátory, slovníky nebo ontologie, lze je propojit (merge) přidáním tvrzení s vlastnostmi *owl:sameAs*, *owl:differentFrom* a *owl:AllDifferent*.

2.8 Porovnání OOP s RDF

- Koncept RDF/OWL je velmi obecný ...
- Podobnost RDF a OOP přístupu.
- Dědičnost:
  - tříd,
  - vlastností,
  - vícenásobná.
- Porovnání:
  - RDF zdroj = objekt v OOP,
  - Vlastnost = instanční proměnná v OOP.
- Instance (objekt/data) je typu (rdf:type) nějaké třídy nebo tříd.

2.9 Porovnání s relační databází

- Zásadní rozdíl je ve způsobu přístupu k datům.
  - Schéma relační databáze nebude (neměl by :-)) nikdo cizí číst a zkoumat, k datům mají přístup pouze vybrané aplikace/uživatelé.
  - V RDF je veřejné schéma dat (slovník nebo ontologie).
  - Účelem RDF je zpřístupnit *data* a *informace* včetně jejich *významu* na webu ve vhodné strojově čitelné podobě.
  - V případě Linked Data jsou navíc publikovány také *vztahy* (relace) mezi informacemi.

Schéma databáze = datový model je pevně definován

- V RDF je schéma prostřednictvím slovníků a ontologií (RDF, OWL).
- V RDF neexistuje pevné schéma, může se měnit.
- V jednom RDF úložišti lze mít data s různým schématem současně.

Tabulka = entita

- RDF nebo OWL třída

Sloupec tabulky = atribut entity

- V RDF se jedná o vlastnost (property)
- V RDF nemusí vlastnost náležet ke konkrétní třídě (chybí *rdfs:domain*).
- RDF vlastnost definuje jednu nebo více domén (*rdfs:domain*)
- Existence *rdfs:domain* u RDF vlastnosti navíc znamená:
  - instance, kde je RDF vlastnost použita, bude typu (třídy), který *rdfs:domain* specifikuje.

Datový typ atributu = povolený datový typ hodnot

- V RDF není kontrola datového typu.
- Datový typ nemusí být určen.
- V RDF vlastnost může definovat více datových typů (*rdfs:range*).
  - Hodnoty stejné vlastnosti mohou mít různé datové typy.
- Existence *rdfs:range* u RDF vlastnosti navíc znamená, že všechny hodnoty, kde byl použit predikát s *rdfs:range*, bude datového typu, který definoval.

Záznam = řádek tabulky

- V RDF je to několik trojic najednou, protože jedna trojice by odpovídala hodnotě v jednom ze sloupců tabulky.
- POZOR: Z neexistence tvrzení/odpovědi nelze v RDF nic vyvozovat!
  - když dotaz na X vrátí prázdný výsledek
    - v relační databázi = X neexistuje,
    - v RDF = nevíme zda X existuje, jen nemusíme mít k dispozici data,
      - Open World Assumption (OWA).

3 Vývoj a vývojové nástroje

- RDF úložiště
  - in-memory – dle použitého frameworku
  - perzistentní
    - Apache Fuseki
    - Virtuoso Open-Source Edition od OpenLink Software
    - Oracle Semantic Web Technologies
    - a další.
- Programovací jazyky
  - Java
    - Apache Jena (https://jena.apache.org/documentation/serving_data/)
    - RDF4J (http://rdf4j.org/, dříve Sesame)
  - Python
    - RDFLib
  - a další.
- Seznam dalších nástrojů souvisejících s RDF: https://www.w3.org/RDF/
  - https://www.w3.org/2001/sw/wiki/Category:Tool
  - https://www.w3.org/2001/sw/wiki/Category:Programming_Language

4 SPARQL – dotazovací jazyk

- SPARQL 1.0 (2008)
  - Query Language
  - HTTP Protocol
  - Results – XML and JSON formats
- SPARQL 1.1 (W3C Recommendation 21 March 2013)
  - SPARQL 1.1 Overview – https://www.w3.org/TR/sparql11-overview/
  - SPARQL 1.1 Query Language https://www.w3.org/TR/2013/REC-sparql11-query-20130321/
  - Updated Query Language & HTTP Protocol
  - SPARQL 1.1 Update
  - Graph Store HTTP Protocol (RESTful přístup k RDF grafům)
  - SPARQL 1.1 Service Descriptions – popis služeb poskytovaných SPARQL endpointy
  - SPARQL 1.1 Entainments – odvozování a SPARQL
  - SPARQL 1.1 Basic Federated Query – dotazování více SPARQL endpointů současně v rámci jednoho dotazu.
- Umožňuje:
  - získat data (SELECT, DESCRIBE, CONSTRUCT)
  - objevovat data dotazováním na neznámé vztahy (ASK, DESCRIBE),
  - transformovat RDF data z jednoho schéma do jiného (CONSTRUCT).
  - provést komplexní dotazování přes více databází v jednom jednoduchém dotazu,
    - Federated SPARQL Query (*SERVICE*)
- Výběrové typy dotazu:
  - ASK
  - CONSTRUCT

  - DESCRIBE
  - SELECT

- Formát výsledku výběrových dotazů – závisí na typu dotazu:
  - ASK – vrací boolean (hodnotu true/false),
  - CONSTRUCT a DESCRIBE –vrací RDF graf,
  - SELECT – vrací tabulku – CSV/TSV, HTML, TXT, JSON, XML, ... záleží na RDF úložišti/databázi, může poskytovat i např. XLS soubory.
- Aktualizační typy dotazu
  - aktualizace grafu
    - INSERT DATA – vložení trojic (triple)
    - DELETE DATA – odstranění trojic z grafu (quad)
    - LOAD – čte data ze zadaného IRI

      LOAD [SILENT] <iri_ref> INTO GRAPH <graph_name>

    - CLEAR – odstraní všechny trojice z daného/daných grafů (IRI, DEFAULT, NAMED, ALL).
  - správa grafu
    - CREATE – vytvoření nového grafu (může-li existovat prázdný graf),
    - DROP – odstranění grafu i jeho obsahu,
    - COPY – kopíruje obsah grafu do jiného,
    - MOVE – přesune obsah grafu do jiného,
    - ADD – duplikuje obsah jednoho grafu do jiného.
- Schéma dotazu:

```
# deklarace prefixů
PREFIX foo: <http://example.com/resources/>
...

# definice datasetu/grafu/zdroje
FROM ...

# výsledek dotazu
SELECT ...

# podmínka -- vzor dotazu (query pattern)
WHERE {
    ...
}
# modifikátory dotazu
ORDER BY ...
```

- Proměnná začíná prefixem otazníku (dolaru): ?s ?p ?o.
  - Může nabývat jakékoliv hodnoty: zdroje (IRI/URI) i literálu.
- V části WHERE:
  - je vzor dotazu zapsán formou trojice/trojic,
  - část trojice může být nahrazena proměnnou,
  - nezáleží na pořadí trojic vzoru dotazu,
  - implicitně je logický AND mezi trojicemi (platí všechny současně),
  - vzor dotazu funguje jako maska/filtr, která je aplikována na RDF data (graf),
    - data, která neodpovídají masce se neberou ve výsledku dotazu v potaz,
    - jestliže nemusí trojice (ale může) existovat a chceme její hodnotu, je potřeba použít klíčové slovo a blok OPTIONAL,

  - proměnné uvedené v části WHERE jsou spojeny s hodnotou odpovídající dané části trojice (zdroj, literál) a mohou být použity v části SELECT (CONSTRUCT, DESCRIBE)
  - existuje BIND, které provede přiřazení hodnoty proměnné (explicitně zadané v dotazu např. v SELECT):

```
BIND (?hodnota*(1-?sleva) AS ?cena)
```

- V části SELECT:
  - se proměnné čárkou neoddělují,
  - vrací tabulku hodnot proměnných, které splňují část podmínky WHERE.
- Filtrování hodnot
  - FILTER pracuje s podmínkami typu boolean.
  - logické: !, &&, ||
  - matematické: +, -, *, /
  - porovnání: =, !=, <, >, IN, NOT IN, ...
  - testy RDF/SPARQL: isURI, isBlank, isLiteral, isNumeric, bound, !bound
  - SPARQL funkce:
    - str, lang, datatype
    - sameTerm, langMatches, regex, REPLACE, ...
    - podmínky: IF, COALESCE, EXISTS, NOT EXISTS
    - konstruktory URI, BNODE, STRDTm STRLANG, UUID, STRUUID,
    - řetězce: STRLEN, SUBSTR, UCASE, LCASe, STRSTARTS, STRENDS, CONTAINS, STRBEFORE, STRAFTER, CONCAT, ENCODE_FOR_URI
    - matematika: abs, round, ceil, floor, rand
    - datum a čas: now, year, month, day, hours, minutes, seconds, timezone, tz
    - hash: MD5, SHA1, SHA256, SHA384, SHA512

```
FILTER (?hodnota > 1000 && langMatches(lang(?nazev), "EN")) .
```

  - lang získá jazyk specifikovaný u textového řetězce
  - langMatches porovná s uvedeným jazykem nebo jejich výčtem

Ukázková data

```
# https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#construct
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> . # Friend-of-a-Friend

<abc> foaf:name "Alice" .
<abc> foaf:mbox <mailto:alice@example.org> .
```

Typy dotazů jsou:

SELECT všechny nebo podmnožinu proměnných z podmínkové části

```
# příklad 1 - všechny trojice
SELECT ?s ?p ?o
WHERE {
 ?s ?p ?o .
}

# příklad 1 - všechny vlastnosti a jejich hodnoty
SELECT ?p ?o
WHERE {
 <abc> ?p ?o .
}
```

CONSTRUCT vrací RDF graf sestavený dle šablony trojic

- získání pod-grafu
- vytvoření odvozených dat
- transformace dat mezi schématy nebo vytváření nových tvrzení

```
# https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#construct
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>

# příklad 1 - jen filtrování
CONSTRUCT WHERE { ?x foaf:name ?name . }

# příklad 2 - transformace schéma
CONSTRUCT {
  ?x vcard:FN ?name
}
WHERE {
  ?x foaf:name ?name .
}
```

ASK odpověď je datového typu *boolean*;

```
# https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#ask
# příklad 1 - odpověď bude 'true' nebo 'false'?
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name  "Alice" }

# příklad 2 - odpověď bude 'true' nebo 'false'?
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
ASK { ?x vcard:FN  "Alice" }
```

- hodnota *true*, pokud vzor dotazu vyhovuje nějaké odpovědi,
- jinak *false*.

DESCRIBE vrací RDF graf popisující zdroj

```
# https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#describe

# příklad 1 - známe URI
DESCRIBE <abc>

# příklad 2 - neznáme konkrétní URI
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
DESCRIBE ?x
WHERE {
  ?x foaf:name "Alice" .
}
```

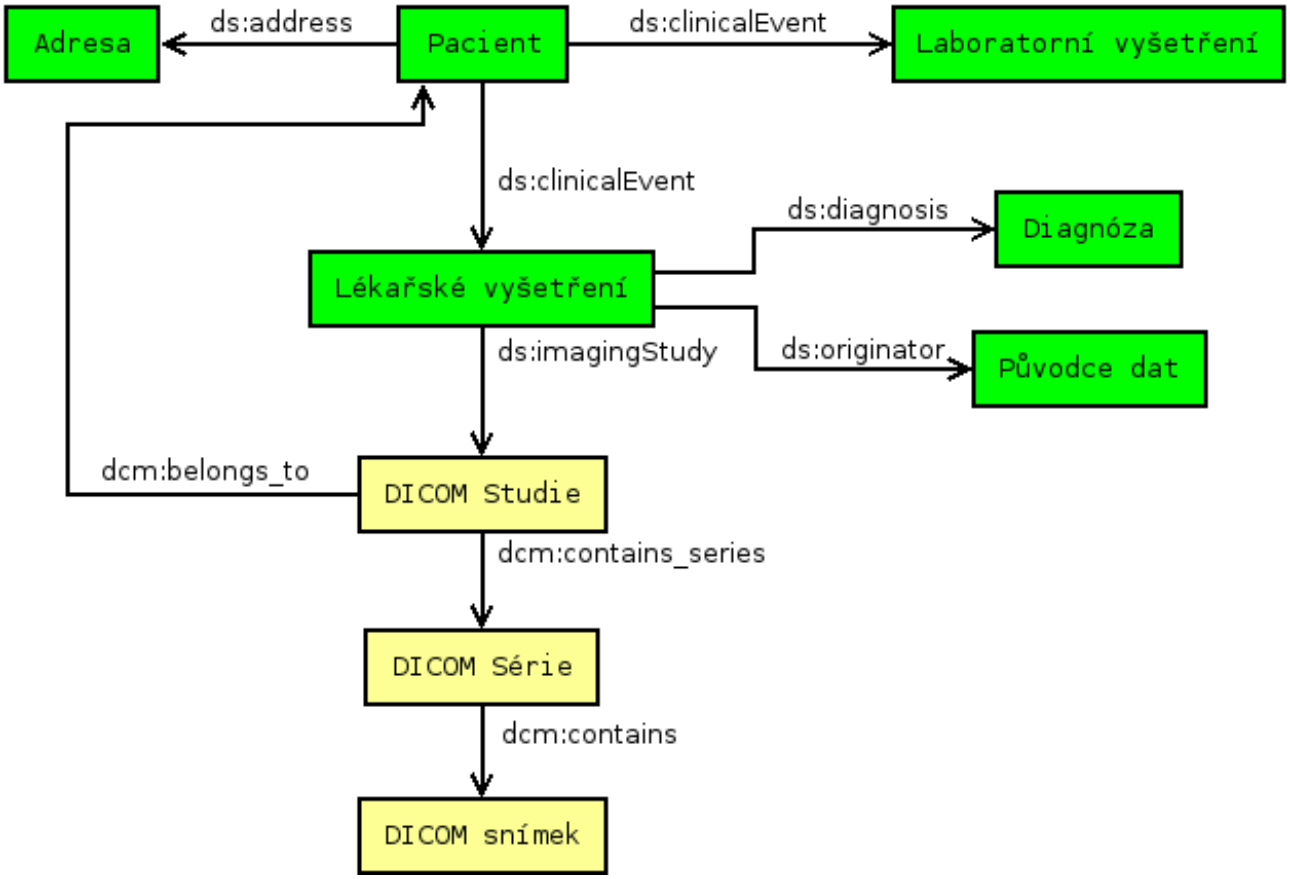
5 SPARQL – řešené příklady

5.1 Schéma dat v systému MRE

- Data vychází z datového formátu DASTA (DAtový STAndard) spravovaný Ministerstvem zdravotnictví ČR
  - http://ciselniky.dasta.mzcr.cz/,
  - obsahuje definici datového formátu DASTA (v3, v4),
  - obsahuje definici číselníků a jejich hodnot.
- V systému MRE používané ontologie jsou dokumentovány: https://mre.zcu.cz/ontology/ontologies.html
- V našem případě jsou:
  - schéma vybraných tříd a vlastností viz obrázek 3.
  - data formátu DASTA transformována do RDF.
  - schéma popisují číselníky a ontologie
  - DASTA Ontology (prefix **ds**)
    - dokumentace http://mre.zcu.cz/ontology/dasta
    - *dasta.owl* – http://mre.zcu.cz/ontology/dasta.owl
  - Číselník DASTA (prefix **dscl**)
    - zdroj číselníků DASTA, NČLP[1], ÚZIS[2]
    - dokumentace http://mre.zcu.cz/ontology/dscl
    - *dscl.owl* http://mre.kiv.zcu.cz/ontology/dscl.owl
    - data číselníku DASTA http://mre.zcu.cz/ontology/dscl.rdf.gz
  - DICOM Ontology (prefix **dcm**) – aktuálně používaná ontologie pro popis obrazových vyšetření ve formátu DICOM
    - dokumentace http://mre.kiv.zcu.cz/ontology/dicom
    - *dicom.owl* http://mre.kiv.zcu.cz/ontology/dicom.owl
    - Základem je třída Patient (*dcm:Patient*), k němuž se může vztahovat několik DICOM studií (dcm:Study).
    - Každá DICOM studie je složena z několika sérií (*dcm:Series*).
    - DICOM série obsahuje/je složena z konkrétních snímků (*dcm:CT_Image*) obrazového vyšetření, např. počítačová tomografie (CT), magnetická rezonance (MR).
    - Existuje také ontologie SEDI (SEmantic DIcom).
  - SITS Ontology – ontologie inspirovaná mezinárodním registrem *Safe Implementation of Treatments in Stroke* (SITS) pro sledování průběhu a výsledku léčby pacientů pro cévní mozkové příhodě.
    - dokumentace http://mre.kiv.zcu.cz/ontology/sits
    - *sits.owl* http://mre.kiv.zcu.cz/ontology/sits.owl
    - *nihss.owl* http://mre.kiv.zcu.cz/ontology/nihss.owl

[1] Národní číselník laboratorních položek
[2] Ústav zdravotnických informací a statistiky ČR

Obrázek 3: Schéma vybraných tříd a vlastností v systému MRE



5.2 Připravená RDF data

- patient1-medical
  - 1 pacient se jménem *Anon_666,*
  - 45 RDF trojic.
- patient6-medical
  - zahrnuje celkem 6 pacientů, včetně pacienta z patient1-medical,
  - 24 730 RDF trojic.
- patient7-medical
  - zahrnuje celkem 7 pacientů,
  - přidán jeden pacient oproti předchozímu patient6-medical,
  - 25 011 RDF trojic.
- patient7-imaging
  - data popisující obrazová vyšetření jednoho pacienta, který je součástí předchozího *patient7-medical*,
  - 33 158 RDF trojic.

Použité datové formáty (koncovka):

nt N-TRIPLE

ttl TURTLE

xml RDF/XML

5.3 Používané jmenné prostory

```
PREFIX id:    <http://mre.zcu.cz/id/>
PREFIX ds:    <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX dscl:  <http://mre.zcu.cz/ontology/dscl.owl#>
PREFIX dcm:   <http://mre.zcu.cz/ontology/dcm.owl#>
PREFIX sits:  <http://mre.zcu.cz/ontology/sits.owl#>
PREFIX nihss: <http://mre.zcu.cz/ontology/nihss.owl#>
PREFIX mre:   <http://mre.zcu.cz/ontology/mre.owl#>

PREFIX acl:   <http://www.w3.org/ns/auth/acl#>
PREFIX dc:    <http://purl.org/dc/elements/1.1/>
PREFIX nfo:   <http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
```

- Kompletní výčet a detaily viz https://mre.zcu.cz/ontology/ontologies.html.

5.4 SPARQL Server – Apache Fuseki

- Stažení: https://jena.apache.org/download/
- Rozbalíme a z rozbaleného adresáře spustíme:
  - v Linuxu:

    ```
    ./fuseki-server
    ```

  - ve Windows:

    ```
    fuseki-server.bat
    ```

  - následně v konzoli uvidíme start serveru.
- Server běží ve výchozím nastavení na portu **3030**.
- Ve webovém prohlížeči zadáme adresu:
  http://localhost:3030
- Připraveno k nahrání dat a zkoušení příkladů.

5.5 Data příkladů – pacient Anon_666

```
id:cd3f0c85b158c08a2b113464991810cf2cdfc387
  a                  ds:Patient , ds:Male ;
  ds:address         id:3840aecb9edac9f7d7c9172f2f4be82b08ab3ddf ;
  ds:clinicalEvent   id:be8d011f882326495f8d06c58f22db51d95cc7bf ;
  ds:datetimeBirth   "1938-08-13"^^xsd:date ;
  ds:lastName        "Anon_666" ;
  ds:patientID       "666" ;
  ds:sex             "M" ;
  ds:sexNCLPTPS      dscl:NCLPTPS_M ;
  ds:sexPOHLAV       dscl:POHLAV_1 ;
  dc:title           "Anon_666 (M) * 1938-08-13" .

id:3840aecb9edac9f7d7c9172f2f4be82b08ab3ddf
  a                 ds:PermanentAddress ;
  ds:addressCity    "Město 1" ;
  ds:addressZIP     "00001" ;
  dc:title          "Město 1 (00001)" .

id:be8d011f882326495f8d06c58f22db51d95cc7bf
  a                           ds:MedicalExamination ;
  ds:datetimeEvent            "2012-09-19T23:20:00+0200"^^xsd:dateTime ;
  ds:diagnosis                id:2c545600eb7a2722809d64c2753de714a5154b6b ,
                              id:2285692c932c88f8673a162ef7b5c997993da41c ;
  ds:dsclExaminationContent   dscl:LZSOZ_NL ;
  ds:dsclExaminationOrigin    dscl:LZTOZV_J ;
  ds:dsclExaminationRequest   dscl:LZTZOV_D ;
  ds:dsclExaminationState     dscl:LZSZZ_K ;
  ds:imagingStudyNumber       "00000078" ;
  ds:originator               id:44040e7024d5a4cc177bf0ed29683c2185dbd05b ;
  ds:reportText               "CT mozku:..." ;
  ds:reportTitle              "032/002 - CT mozku: s k.l. iv." ;
  dc:title                    "2012-09-19 23:20: 032/002 - CT mozku: s k.l. iv."
      .

id:2c545600eb7a2722809d64c2753de714a5154b6b
  a                 ds:ActualDiagnosis ;
  ds:datetimeEvent  "2012-04-18"^^xsd:date ;
  ds:diagCode       dscl:MKN10_5_J180 ;
  ds:diagDetail     "Bronchopneumonie NS" ;
  ds:diagOrder      1 ;
  ds:patient        id:cd3f0c85b158c08a2b113464991810cf2cdfc387 ;
  dc:title          "2012-04-18 (1) J18.0 - Bronchopneumonie NS" .

id:2285692c932c88f8673a162ef7b5c997993da41c
  a                 ds:ActualDiagnosis ;
  ds:datetimeEvent  "2012-04-18"^^xsd:date ;
  ds:diagCode       dscl:MKN10_5_I639 ;
  ds:diagDetail     "Mozkový infarkt" ;
  ds:diagOrder      2 ;
  ds:patient        id:cd3f0c85b158c08a2b113464991810cf2cdfc387 ;
  dc:title          "2012-04-18 (2) I63.9 - Mozkový infarkt" .

id:44040e7024d5a4cc177bf0ed29683c2185dbd05b
  a                  ds:OriginatorDepartment ;
  ds:departmentName  "Nemocnice na ..." ;
  dc:title           "Nemocnice na ..." .
```

5.6 Jednoduché dotazy

5.6.1 Všechny existující trojice

- Chceme-li získat všechny trojice, musíme v části WHERE mít vzor trojice se třemi proměnnými. Ty vyhovují všem trojicím. V části SELECT je uvedeme.

```
#01a
SELECT ?s ?p ?o
WHERE {
  ?s ?p ?o
}
```

  - vrátí všechny trojice, tj. tři sloupce s hodnotami dle celkového počtu trojic v datasetu.

- Omezení počtu lze provést použitím klíčového slova LIMIT

```
#01b
SELECT ?s ?p ?o
WHERE {
  ?s ?p ?o
}
LIMIT 10
```

  - vrátí 10 trojic ve třech sloupcích odpovídajících trojicím.

- Vedle LIMIT, lze použít také OFFSET.

5.6.2 Rodné číslo pacienta pro známé URI

- Známe URI zdroje (subjekt)
- Známe vlastnost s rodným číslem pacienta (ds:patientID)
- Zajímá nás hodnota rodného čísla:

```
#02
PREFIX id: <http://mre.zcu.cz/id/>

SELECT ?id
WHERE {
  id:cd3f0c85b158c08a2b113464991810cf2cdfc387 ds:patientID ?id .
}
```

  - vrátí jeden řádek s jednou hodnotou (jeden sloupec).

- V části WHERE může být i celé URI bez zkrácení prefixem.
- Hodnota vlastnosti *ds:patientID* bude svázána (bind) s proměnnou *?id* a dostupná jako výsledek v části SELECT.

5.6.3 URI pacienta pro známé rodné číslo

- Známe vlastnost s rodným číslem pacienta (ds:patientID).
- Známe hodnotu – rodné číslo.
- Zajímá nás URI pacienta (zdroje, subjekt):

```
#03
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?patient
WHERE {
  ?patient ds:patientID "666" .
}
```

  - výsledkem je jedna hodnota URI (pro data z *patient1-medical*).

5.6.4 Všechny vlastnosti a hodnoty pro zadané URI

- Známe URI.
- Zajímají nás všechny vlastnosti a jejich hodnoty:

```
#04a
PREFIX id: <http://mre.zcu.cz/id/>

SELECT ?vlastnost ?hodnota
WHERE {
  id:cd3f0c85b158c08a2b113464991810cf2cdfc387 ?vlastnost ?hodnota .
}
```

  - výsledkem je 11 dvojic (pár vlastnost a její hodnota) pro výše uvedené URI.

- V podobě grafu lze prakticky totéž získat prostřednictvím DESCRIBE:

```
#04b
PREFIX id: <http://mre.zcu.cz/id/>

DESCRIBE id:cd3f0c85b158c08a2b113464991810cf2cdfc387
```

  - vrátí RDF graf s 11 trojicemi, které mají jako subjekt uvedené URI.

5.6.5 URI všech pacientů a jejich rodných čísel

- Známe URI vlastnosti pro rodné číslo ds:patientID.
- Zajímá nás URI zdroje a hodnota vlastnosti:

```
#05
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?patient_uri ?id
WHERE {
  ?patient_uri ds:patientID ?id .
}
```

  - výsledkem je jedna hodnota URI a rodného čísla (pro data z *patient1-medical*).

5.6.6 Rodné číslo a příjmení pro zadané URI pacienta

- Můžeme použít více vzorů trojic s proměnnou.
- Pro stejné zdroje lze mít více tvrzení oddělených středníkem (viz Turtle) v části WHERE.
- Pro vypsání hodnot umístíme proměnnou v části SELECT:

```
#06a
PREFIX id: <http://mre.zcu.cz/id/>
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?prijmeni
WHERE {
  id:cd3f0c85b158c08a2b113464991810cf2cdfc387
    ds:patientID ?id ;
    ds:lastName  ?prijmeni .
}
```

- nebo použijeme hvězdičku (*), chceme-li zobrazit všechny proměnné:

```
#06b
PREFIX id: <http://mre.zcu.cz/id/>
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT *
WHERE {
  id:cd3f0c85b158c08a2b113464991810cf2cdfc387
    ds:patientID ?id ;
    ds:lastName  ?prijmeni .
}
```

  - v obou příkladech dostaneme stejný výsledek.

5.6.7 Další identifikátory a rodné číslo s příjmením pacientů

- Chceme k rodnému číslu a příjmení pacienta získat také další identifikátor.
- Ostatní nebo jiné identifikátory mají vlastnost *ds:patientID2:*

```
#07a
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?jine_id ?prijmeni
WHERE {
  ?patient ds:patientID  ?id .
  ?patient ds:patientID2 ?jine_id .
  ?patient ds:lastName   ?prijmeni .
}
```

  - jaký je výsledek dotazu? Proč?

- Správné řešení:

```
#07b
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?jine_id ?prijmeni
WHERE {
  ?patient ds:patientID ?id .
  OPTIONAL {
    ?patient ds:patientID2 ?jine_id .
  }
  ?patient ds:lastName  ?prijmeni .
}
```

  - výsledek (počet řádek) dle zvoleného příkladu: 1 pro patient1-medical, 6 pro patient6-medical, 7 pro patient7-medical.

- Význam klíčového slova OPTIONAL – ovlivní počet výsledků.

5.6.8 Rodná čísla a příjmení pacientů s rodným číslem větším než 500

- Filtrování hodnoty rodného čísla:

```
#08a
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?prijmeni
WHERE {
  ?patient ds:patientID ?id .
  ?patient ds:lastName  ?prijmeni .
  FILTER (?id > 500) .
}
```

  - všechny hodnoty rodného čísla porovná s filtrem.
  - Funguje?

- Hodnota ds:patientID je string, nikoliv číslo, proto je potřeba hodnotu přetypovat:

```
#08b
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?id ?prijmeni
WHERE {
  ?patient ds:patientID ?id .
  ?patient ds:lastName  ?prijmeni .
  FILTER (xsd:int(?id) > 500) .
}
```

5.6.9 Pacienti s datem narození v určeném období

- Ve filtru uvedeme i datový typ hodnoty:

```
#09
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?prijmeni ?narozeni
WHERE {
  ?patient ds:patientID ?id .
  ?patient ds:lastName  ?prijmeni .
  ?patient ds:datetimeBirth ?narozeni .
  FILTER (
    ?narozeni > "1900-01-01"^^xsd:date &&
    ?narozeni < "1949-12-31"^^xsd:date
  )
}
ORDER BY DESC(?datum)
```

5.6.10 Všechna rodná čísla a příjmení pacientů – seřazených dle rodného čísla

- Obdoba předchozích příkladů, jen je navíc požadováno řazení:

```
#10
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?id ?prijmeni
WHERE {
  ?patient ds:patientID ?id .
  ?patient ds:lastName  ?prijmeni .
}
ORDER BY DESC(xsd:int(?id))
```

  - vhodné použít data *patient6-medical* nebo *patient7-medical*,
  - výsledek (počet řádek) dle zvoleného příkladu: 1 pro patient1-medical, 6 pro patient6-medical, 7 pro patient7-medical
  - porovnejte různé varianty části ORDER BY – rozdíl mezi znakovým a číselným řazením:

```
ORDER BY ?id
ORDER BY ASC(?id)
ORDER BY DESC(?id)
ORDER BY DESC(xsd:int(?id))
ORDER BY ?prijmeni DESC(?id)
```

- Výsledek dotazu lze ovlivnit modifikátorem ORDER BY pro řazení.
- ASC, DESC určují směr řazení.
- V případě chybějícího datového typu se jedná obecně o string:
  - proto je řazení implicitně znakové,
  - chceme-li řadit číselně, je nutné doplnit přetypování: *xsd:int(?id)*.

5.6.11 URI všech instancí třídy aktuální diagnózy

- Aktuální diagnóza má třídu *ds:ActualDiagnosis*.
- Typ instance je určen vlastností *rdf:type*.

```
#11a
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?uri
WHERE {
  ?uri rdf:type ds:ActualDiagnosis .
}
```

- Vlastnost *rdf:type* lze zkracovat na prosté písmeno „a“.

```
#11b
PREFIX ds: <http://mre.zcu.cz/ontology/dasta.owl#>

SELECT ?uri
WHERE {
  ?uri a ds:ActualDiagnosis .
}
```

  - oba příklady vrátí vždy dvě URI (pro data z *patient1-medical*).

5.6.12 Ověření získaných znalostí

Použijte zejména data *patient7-medical* a ověřte, že vaše dotazy vrací správné výsledky pro níže uvedené příklady.

1. Příklad 12-1a – Pro všechny pacienty vypište hodnoty sloupců:
   - rodné číslo,
   - příjmení,
   - datum narození,
   - datum úmrtí,
   - pohlaví (pouze písmeno M nebo F),
   - anotace zdroje (instance) z *dc:title* a současně i *rdfs:label*,
   - numericky seřaďte vzestupně dle rodného čísla.
2. Příklad 12-1b – Upravte příklad 12-1a omezením výčtu pacientů pouze na jednoho explicitním zadáním URI.
3. Příklad 12-1c – Upravte příklad 12-1a omezením výčtu pacientů na min. dva nebo tři pacienty zadané prostřednictvím známého URI.
4. Příklad 12-2 – Upravte příklad 12-1a omezením na pacienty mužského pohlaví prostřednictvím:
   (a) filtru hodnoty (12-2a),
   (b) instance pacientů třídy *ds:Male* (12-2b).

5.7 Průchod grafu

- Doporučená data: *patient6-medical* nebo *patient7-medical*.

5.7.1 Příjmení a rodné číslo pacientů společně s datem a časem z lékařské zprávy

- Vypsat informace o pacientovi – příjmení, rodné číslo
- Vypsat datum a čas lékařské zprávy.

```
#13a
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?id ?prijmeni ?datum
WHERE {
  ?patient ds:patientID      ?id ;
           ds:lastName       ?prijmeni ;
           ds:clinicalEvent  ?event .

  ?event   ds:datetimeEvent  ?datum .
}
ORDER BY ASC(xsd:int(?id)) ?datum
```

  - vrací 1 013 výsledků.
    - Je to správně?
  - Přidání DISTINCT za SELECT omezí počet duplicitních výsledků a dostaneme 47 výsledků.
    - Je to správně?

- Kontrola:
  - zjistíme kolik existuje instancí třídy ds:MedicalExamination:

```
#13b
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?uri
WHERE {
  ?uri rdf:type ds:MedicalExamination .
}
```

  - nebo prostřednictvím agregační funkce *count()*:

```
#13c
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT (count(?uri) as ?pocet)
WHERE {
  ?uri rdf:type ds:MedicalExamination .
}
```

    - v obou případech je celkový počet 10 lékařských zpráv.

- Ve skutečnosti může být v *ds:clinicalEvent*
  - lékařská zpráva (*ds:MedicalExamination*),
  - i výsledky laboratorního vyšetření (*ds:LaboratoryReport*)
  - má-li být výstupem pouze datum a čas lékařských zpráv, musíme doplnit trojici, která specifikuje typ (třídu).

```
#13d
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?id ?prijmeni ?datum
WHERE {
  ?patient ds:patientID      ?id ;
           ds:lastName       ?prijmeni ;
           ds:clinicalEvent  ?event .

  ?event   rdf:type          ds:MedicalExamination .
  ?event   ds:datetimeEvent  ?datum .
}
ORDER BY ASC(xsd:int(?id)) ?datum
```

- vrací 10 výsledků (stejně to bude v tomto případě s DISTINCT pro uvedená data).
- Přidání trojice s určením typu třídy odstraní všechny ostatní instance *ds:LaboratoryReport*, které nás zrovna nezajímají.

5.7.2 Diagnózy pacientů

- Diagnóza je vždy uvedena u lékařské zprávy pacienta:

```
#14a
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?id ?prijmeni ?datum ?diagPopis ?diagKod
WHERE {
  ?patient   ds:patientID      ?id        ;
             ds:lastName       ?prijmeni  ;
             ds:clinicalEvent  ?event     .

  ?event     ds:diagnosis      ?diagnosis .

  ?diagnosis ds:diagDetail     ?diagPopis ;
             ds:diagCode       ?diagKod   ;
             ds:datetimeEvent  ?datum     .
}
ORDER BY ASC(xsd:int(?id)) ?datum
```

- výsledek je 43 diagnóz pro data z *patient7-medical*.
- musíme projít od pacienta *?patient* (získáme jeho rodné číslo a příjmení) na klinickou událost (*?event*)
- z klinické události *?event* projdeme k diagnóze *?diagnosis*
- z diagnózy nás zajímají číslo, datum a popis.

- využití **Property Path** ve SPARQL 1.1:

```
#14b
PREFIX ds:  <http://mre.zcu.cz/ontology/dasta.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?id ?prijmeni ?datum ?diagPopis ?diagKod
WHERE {
  ?patient ds:patientID      ?id        ;
           ds:lastName       ?prijmeni  ;
           ds:clinicalEvent/ds:diagnosis ?diagnosis .
```

```
  ?diagnosis ds:diagDetail ?diagPopis     ;
             ds:diagCode   ?diagKod       ;
             ds:datetimeEvent ?datum      .
}
ORDER BY ASC(xsd:int(?id)) ?datum
```

5.7.3 Ověření získaných znalostí

1. Je chybou, že zde není v příkladech *14a* a *14b* uvedena třída – rdf:type (*ds:MedicalExamination*)?
2. Dotazem zjistěte, kolik je v datech *patient7-medical* instancí *ds:LaboratoryReport*?
3. Doplňte do *14a* nebo *14b* řazení, dle pořadového čísla diagnózy.
4. Doplňte do *14a* nebo *14b* omezení, abychom získali pouze hlavní diagnózy (mají hodnotu 1 ve vlastnosti *ds:diagOrder*).
5. Co je potřeba v datech změnit/přidat, abychom v předchozí úloze nepotřebovali použít FILTER?
6. Podívejte se na data (na schéma výše není uvedeno) a najděte další způsob, jak jinak vypsat stejné výsledky. Vytvořte odpovídající dotaz.

5.8 Samostatné úlohy

- DISTINCT,
- FILTER
- VALUES,
- agregační funkce,
- GROUP BY,
- HAVING,
- a další

Pro následující úlohy použijte **současně** data *patient7-medical* a *patient7-imaging*.

- Můžete je nahrát:
  1. do jednoho datasetu (výchozího grafu) jako doposud
  2. nebo do dvou samostatných grafů, které si vhodně pojmenujete.
     - V dotazech musíte specifikovat, nad kterým grafem/grafy se dotazujete v části FROM.

Úlohy

1. Vyberte datum s časem všech laboratorních výsledků bez duplicit a seřazené od nejnovějších k nejstarším.
   - Řešení má být 37 výsledků.
2. Najděte a vypište všechny URI diagnóz, které jsou v datech použity (*ds:DiagCode*) a abecedně je seřaďte.
   - Řešení má být 29 výsledků.
3. Zjistěte počet DICOM studií (*dcm:Study*)?
   - Řešení má být jeden výsledek s hodnotou 1 (DICOM studií).
4. Zjistěte, jakým pacientům a které DICOM studie patří – uveďte jeho rodné číslo *ds:patientID* a číslo studie *dcm:Study_ID*.
   - Řešení má být jeden výsledek s pacientem 592 a studií 832.
5. Zjistěte počet DICOM sérií (*dcm:Series*)?
   - Řešení má být jeden výsledek s hodnotou 10 (DICOM sérií).
6. Zjistěte počet DICOM snímků (*dcm:CT_Image*)?
   - Řešení má být jeden výsledek s hodnotou 550 (DICOM snímků).
7. Zjistěte z kolika DICOM souborů (snímků *dcm:CT_Image*) se každá ze sérií skládá, a seřaďte je od největší k nejmenší? Vypište číslo série (*dcm:Series_Number*), popis (dcm:Series_Description) a počet souborů v sérii.
   - Řešení má být 7 výsledků (sérií) s počty souborů: 304, 116, 70, 31, 24, 4 a 1.
8. Zjistěte z kolika DICOM souborů (snímků *dcm:CT_Image*) se každá ze sérií skládá, a jaký objem dat na disku zabírá? Vypište číslo série (*dcm:Series_Number*), popis (dcm:Series_Description), počet souborů v sérii, celkovou velikost série v B i současně v MB. Hodnotu velikosti v MB zaokrouhlete. Série seřaďte dle velikosti v Bytech.
   - Řešení má být 7 výsledků (sérií) s velikostmi (MB): 156, 59, 35, 16, 12, 1 a 0.
9. Najděte laboratorní výsledky, jejichž hodnoty (*ds:labNumberValue*) jsou mimo stanovené referenční meze pro daného pacienta:
   - *ds:labScale4* je referenční mez dolní,
   - *ds:labScale5* je referenční mez horní,

a vypište unikátní název naměřené veličiny (*ds:labLocalName*), hodnotu, a obě referenční meze. Data budou řazena vzestupně dle názvu, dolní meze, hodnoty a horní meze.

(a) Neuvažujte datum laboratorního výsledku.

- Řešení má být 171 výsledků.

(b) Uvažujte datum laboratorního výsledku. Zobrazen bude bezprostředně za názvem. Stejně tak u řazení, nejprve název a pak dle data, následovat budou zbývající hodnoty.

- Řešení má být 216 výsledků.

(c) Zjistěte, kolikrát byly laboratorní hodnoty (název, *ds:labLocalName*) mimo stanovené meze. Zobrazit pouze ty, které se opakují (alespoň 2x) a seřadit dle počtu (sestupně) a názvů (vzestupně).

- Řešení má být 25 výsledků. (Celkem je hodnot mimo referenční meze 47.)