# SZAKDOLGOZAT

*Tóth Viktória*

*Debrecen*
*2008*

Debreceni Egyetem
Informatika Kar

# Az RDF lekérdezőnyelvek

Témavezető:

Jeszenszky Péter

egyetemi tanársegéd

Készítette:

Tóth Viktória

programozó matematikus

Debrecen

2008

# Tartalomjegyzék

# 1. Bevezetés

Weben keresztül elérhető dokumentumok száma jelenleg több milliárd és exponenciális mértékben növekszik. A temérdek honlap közül megtalálni a számunkra fontos információt tartalmazót, koránt sem egyszerű.

Az Interneten különböző típusú és formájú dokumentumok találhatók, melyek egy része ember által létrehozott (HTML, word vagy excel fájlok, PDF-ek, Power Pointok, stb.), másik része pedig nem (pl. statisztikák, egy-egy webszerver havi statisztikája, látogatottsági adatok). Továbbá dokumentumok nagyon sokféle nyelven íródhatnak. Itt nem csak a természetes nyelv sokszínűségére kell gondolni, hanem a különböző programozási nyelvekre is. Ugyanakkor figyelembe kell venni, hogy a különböző dokumentumok különböző terminológiákat használnak ugyanarra a fogalomra.

Az emberek számára nem okoz nehézséget megérteni ezen dokumentumok tartalmát, hiszen felismerjük azt, amit egy fénykép ábrázol (hacsak nincsenek látási problémáink), könnyedén áthidalunk terminológiai különbségeket, félkész információkat tudunk teljessé tenni és sokszor, különböző nyelveken is beszélünk. Ellenben erre a számítógépek nem képesek.

Vegyük a következő estet: meg akarjuk találni egyik új ismerősünk honlapját. Az illetőnek csak a keresztnevére emlékszünk és tudjuk, hogy X nevű cégnél dolgozik logisztikusként, továbbá az unokatestvérünk évfolyamtársa volt. Ennyi információnak valószínűleg elegendőnek kellene lennie a weblap megtalálásához. Sajnos, a jelenlegi kulcsszavas keresők segítségével nem tudnánk felkutatni a honlapot. A probléma az, hogy az ilyen típusú adatokat nem tudjuk leírni a számítógép számára, azt pedig egyelőre végképp nem várhatjuk el tőle, hogy feldolgozza és értelmezze őket.

Az ilyen jellegű problémára a szemantikus web jelent megoldást, amelynek célja egy olyan infrastruktúra létrehozása, amely lehetővé teszi a weben lévő adatok integrálását, a közöttük levő kapcsolatok definiálását és jellemzését, illetve az adatok értelmezését.

A szemantikus világháló elgondolás két alapötleten nyugszik: egyrészt a metaadatok erőforrásokhoz való kapcsolásán, másrészt azon, hogy ezen metaadatok segítségével következtetni is kell tudni az adott dokumentum tartalmára.

A dolgozatban bemutatom a szemantikus web komponensei közül az RDF adatmodellt, amely formális keretet ad az adatok egymáshoz való kapcsolatának definiálására, és a kapcsolatok leírására. Továbbá ismertetem az RDF sémát, amivel a kapcsolatok leírására szolgáló terminológiát tudjuk definiálni. Végezetül bemutatok néhány ismertebb RDF lekérdezőnyelvet, amikkel az adatok közötti kapcsolatokat, illetve a kapcsolt adatok referenciáit tudjuk elérni, lekérdezni.

# 2. Az RDF elképzelés

A webes tartalmak feldolgozhatóságában nagy előrelépés volt az XML, amely szabványos adatcsere formátumként biztosítja az egyes alkalmazások együttműködésének szintaktikus oldalát. Az XML használatával sem lehetséges azonban az, hogy alkalmazások alkalmazásokkal kommunikáljanak anélkül, hogy előtte nem egyeztetik az átvitt információ szemantikáját.

A szemantikus világháló elmélet azt célozza meg, hogy az átvitt, tárolt információ a felek számára egyértelmű legyen, és amennyiben ez megvalósul, akkor a keresőrobot is érteni fogja bizonyos internetes erőforrások tartalmát. Ezzel együtt az információt a gépek számára feldolgozható és érthető formában kell tárolni. Ez utóbbira az RDF-elképzelés szolgálhat.

Az RDF (Recource Description Framework – Erőforrás-Leíró Keretrendszer) adatmodell alkalmas arra, hogy tetszőleges erőforrásokhoz metaadatokat kapcsoljon. Az RDF úgy definiálja az erőforrás fogalmát, hogy az bármi lehet, ami azonosítható URI-val. Akár az asztalomon lévő füzet is lehet erőforrás, feltéve, hogy társítok hozzá egy URI-t.

## 2.1 URI[1]( Universal Resource Identifier)

Az URI (egységes erőforrás-azonosító) egy rövid karaktersorozat, amelyet egy webes erőforrás azonosítására használunk.

Kétféle típusú URI létezik:

- az URL ( *Universal Resource Locator - Egységes erőforrás helymeghatározó*) : az erőforrást azzal határozza meg, hogy hogyan lehet azt elérni. pl.: *http://www.inf.unideb.hu/index.html*
- az URN ( *Universal Resource Name - Egységes erőforrás név*) : az erőforrást anélkül határozza meg, hogy bármit mondana a helyéről vagy az elérhetőségéről.pl.:*urn:isbn:0-395-36341-1*

1 Az RDF URI hivatkozások kompatibilisek a nemzetköziesített forrásazonosítóval ( Inernationalized Resource Identifier, **IRI**).

Az RDF szempontjából lényeges, hogy az URI-knak megkülönböztetjük két fajtáját: az *abszolút* és a *relatív* URI-t. Az abszolút URI-k egyértelműen azonosítják az erőforrásokat, bárhol és bármilyen körülmények között is használjuk az azonosítót. A relatív URI -knak csak egy adott környezetben van értelmük, ha rendelkezésre áll egy ún. bázis URI. A relatív URI-kat a bázis URI segítségével mindig feloldjuk egy algoritmus szerint és abszolút azonosítót kapunk.

Egy URI hivatkozás (vagy *URIref*) egy URI egy opcionális erőforrásrész-azonosítóval (*fragment identifier*-rel) a végén. Például a *http://www.example.org/index.html#section2* URI hivatkozás URI-je: *http://www.example.org/index.html*, erőforrásrész-azonosítója pedig a "#" karakterrel elválasztott s*ection2* azonosító.

## 2.2 Az RDF(Recource Description Framework) adatmodell

A metaadatok leírhatósága érdekében az RDF specifikáció definiál egy halmazelméleti alapokon nyugvó adatmodellt: az RDF adatmodellt. A modellben a következő halmazok definiáltak:

- *Erőforrások* (*Resourceses*): Összes olyan dolog halmaza, amire egy kijelentés vonatkozhat. A halmaz elemeit erőforrásoknak nevezzük, amelyeket URI-k azonosítanak.
- *Tulajdonságok* (*Properties*): Az erőforrásokhoz kapcsolható jellemzők, amelyek vagy RDF-erőforrások vagy literálok. Ontológiák definiálják a tulajdonságok jelentését; azt, hogy milyen erőforráshoz kapcsolhatók és hogy milyen értéket vehetnek fel, valamint, hogy milyen viszonyban vannak más tulajdonságokkal.
- *Kijelentések* (*Statements*): A halmaz elemei olyan hármasok vagy más néven kijelentések, amelyek alanyból (*subject*), állítmányból (*predicate*) és tárgyból (*object*) állnak. Az alany tetszőleges RDF-erőforrás, az állítmány egy tetszőleges RDF-tulajdonság és a tárgy egy tetszőleges RDF-erőforrás vagy literál, azaz egy kijelentést vagy három összetartozó RDF URI hivatkozás alkot, vagy pedig két RDF URI hivatkozás és egy literál.

Az adatmodell jelentése, hogy kijelentések halmazába tartozó hármasok igazak. Az

RDF adatmodell többféleképpen ábrázolható. Az RDF szabvány definiált három szabványos adatmodell reprezentációt: *hármasok halmaza (tripletek)*; *címkézett, irányított gráf* és az *XML-adat*. Az RDF szempontjából a gráfmodell az elsődleges fontosságú, de a feldolgozhatóság szempontjából pedig az XML reprezentáció a lényeges.

### *2.2.1 Az RDF gráfmodellje*

Az RDF az állításokat egy gráf csomópontjaival és éleivel modellezi, amelynek csomópontjainak halmazát a gráf triplettjeinek alanyai és tárgyai alkotják, és az élek halmazát az állítmányok. Az él iránya szignifikáns, és ennek mindig a tárgyra kell mutatnia.

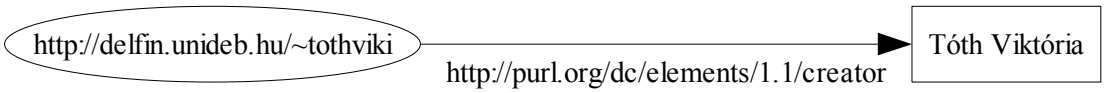
A RDF gráfok megrajzolásakor az URI-val azonosított csomópontokat ellipszissel ábrázoljuk, literálokat szögletes dobozzal reprezentáljuk.

Fontos, hogy egy RDF-gráfban csak abszolút URI-k szerepelhetnek.

Vegyük a következő kijelentést:

> *http://delfin.unideb.hu/~tothviki* erőforrásnak van egy *creator* nevű tulajdonsága, amelynek az értéke *Tóth Viktória*

Ennek megfelelő gráf:



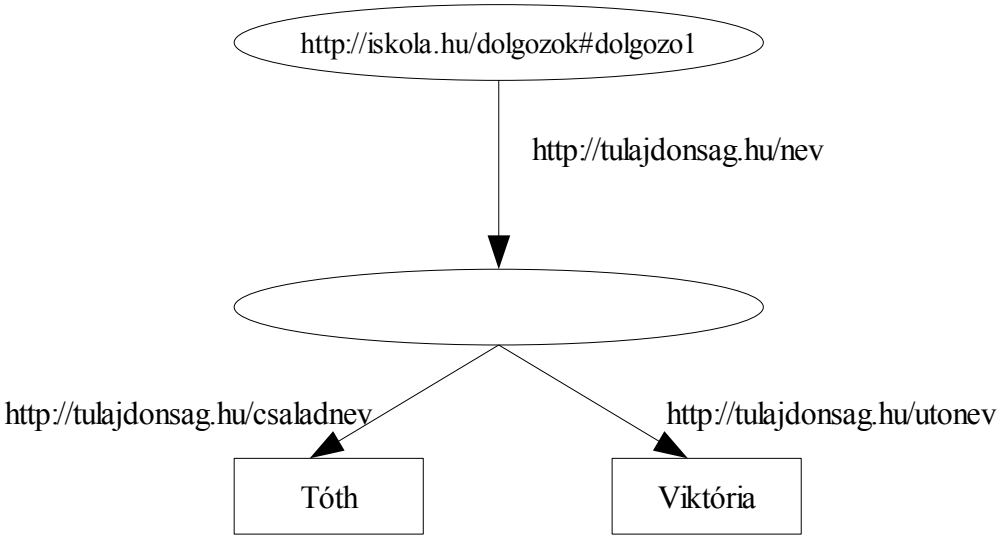

A hármasokkal történő ábrázolás során a gráfban szereplő minden kijelentést egy egyszerű alany-állítmány-tárgy hármassal írjuk le. A fenti kijelentés triplettekkel ábrázolva:

```
<http://delfin.unideb.hu/~tothviki>  <http://purl.org/dc/elements/1.1/creator>  "Tóth Viktória" .
```

### *2.2.2 Üres csomópontok*

A való világban az adatok elég összetettek. Például egy ember nevét tekinthetjük egyetlen egy ábrázolandó információnak vagy két információnak, ha külön akarjuk kezelni a család- illetve utónevet. Az RDF-ben a strukturált információ ábrázolásakor, bevezetünk egy ún. *köztes*, vagy más néven *üres* vagy *névtelen* erőforrást, amelynek jellegzetessége, hogy semmilyen módon nem tartozik hozzá URI. Az RDF-gráfban az ilyen erőforrás üres

csomópontként jelenik meg.




Az üres csomópont betölti célját a rajzon anélkül, hogy szükség volna egy URI azonosítóra, hiszen jól láthatók a kapcsolatok a gráf különböző részei között.

Amennyiben triplettek próbálnánk meg leírni a gráfot, mégis csak szükségünk lenne valamilyen azonosítóra, hogy hivatkozni tudjunk erre az üres csomópontra. Fontos, hogy a bevezetett azonosító különbözzön a gráfban szereplő URI-ktől és literáloktól. Nem kell globálisan egyedinek lennie, hiszen egyetlen egy gráfhoz tartoznak (gráf egyesítéskor újra el kell nevezni az esetleges névütközés elkerülése miatt). Az üres csomópont azonosítására *_:name* használjuk.

| prefix=isk: | névtér-URI= http://iskola.hu/dolgozok# | |
|---|---|---|
| prefix=n: | névtér-URI= http://tulajdonsag.hu/ | |
| Isk:dolgozol | n:nev | _:emberneve . |
| _: embeneve | n: csaladnev | "Tóth" . |
| _: emberneve | n: utonev | "Viktória" . |

Az üres csomópontot használhatjuk n-ér reláció leírására. Minden n-ér relációban az egyik résztvevő elemet kinevezzük a reláció alanyának és üres csomópontot készítünk, amelynek tulajdonságai a reláció többi résztvevője.

Az üres csomópontot használhatjuk URI-val nem rendelkező erőforrásról szóló kijelentés pontosabb megfogalmazására, ha az erőforrás leírható más URI-val ellátott

erőforrásokhoz való viszonya alapján. Továbbá üres csomópont alkalmazása elkerülhetővé teszi a literálok használatát olyan esetekben, amikor az nem a legjobb megoldás. Például, ha egy kép festőjére hivatkozom.

### *2.2.3 Adattípusok az RDF-ben*

A programozási nyelvek és az adatbázis-rendszerek területén elterjedt gyakorlat az, hogy adattípust kapcsolnak a literálokhoz, amelyből egyértelműen kiderül miképpen kell értelmezni az adott literált.

A típussal rendelkező literálokat tipizált literáloknak nevezzük, amelyek egy karakterláncból és egy URI hivatkozásból állnak. Az előbbi a literál lexikai megjelenését reprezentálja, az utóbbi a típusát azonosítja.

```
prefix=xsd:      névtér-URI= http://www.w3c.org/2001/XMLSchema#
<http://valami.hu/fotel>          <http://raktar.hu/rakszam>        "53232"^^xsd: integer .
```

Az RDF csak egy szabványos módot biztosít arra, hogy egy literálhoz hozzárendelhessünk egy külső típust, azaz RDF nem törődik milyen URI-t adunk meg a karaktersorozat mellé. Az RDF-forrás feldolgozó feladata, amennyiben érti az URI-t, hogy ellenőrizze a literált.

### *2.2.4 Az RDF XML-alapú szintaxisa*

Ebben a részben bemutatom a főbb XML-konstrukciók és XML-névterek használatát az RDF esetén.

#### *2.2.4.1 Alapvető XML-szintaxis*

Az RDF-gráf XML-alakja egy valódi XML-dokumentum, így egy RDF/XML-dokumentum az XML-deklarációval indít.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:isi="http://delfin.unideb.hu/~tothviki/iskola">
        . . .
</rdf:RDF>
```

Az <rdf:RDF> elemen belüli tartalmat RDF-ként kell kezelni, amit be lehet illeszteni

más, például nagyobb XML-forrásba. Az XML névtér deklarációt az <rdf:RDF> elem *xmlns* attribútumaként szokás megadni.. Például: *xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"* . Ez a deklaráció azt mondja ki, hogy minden olyan név ebben a kontextusban, amely az *rdf:* prefixet viseli, annak a névtérnek a része, amelyet az attribútum értékekén megadott *http://www.w3.org/1999/02/22-rdf-syntax-ns#* névtér-URIref azonosít.

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <isi:neve>Seregélyesi Szakképzõiskola és Kollégium</isi:neve>
</rdf:Description>
```

A *Description* elem segítségével írhatjuk le az RDF-kijelentés alanyát, amit azonosító URI-t az <rdf:Description> elem *rdf:about* attribútumaként adhatjuk meg. A kijelentés állítmányát az <isi:neve> tulajdonságelem neve, a tárgyát pedig a <isi:neve> elem tartalma határozza meg. Az RDF-kijelentés predikátumát egy URI azonosítja, jelen esetben az *URI="http://delfin.unideb.hu/~tothviki/iskola#neve"*, tehát az *isi:neve* egy minősített XML név.

Ha egy alanyról több állítást akarunk megfogalmazni, akkor érdemes a következőképpen rövidíteni az XML-alakot ( *rdf:Description* elemnek több tulajdonságeleme van):

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <isi:neve>Seregélyesi Szakképzõiskola és Kollégium</isi:neve>
        <isi:fenntarto rdf:resource="http://www.fejer.hu/"/>
</rdf:Description>
```

A fenti példa további újdonsága az *rdf:resource* attribútum, amely az erőforrás-erőforrás kapcsolatok leírására használjuk. Jelen esetben a *"http://www.seregszaksuli.hu"*-val azonosított erőforrást a *"http://www.fejer.hu"* URI-val ellátott erőforrással kapcsolja össze, azaz a kijelentés alanyát a tárggyal. Az attribútum értékét minden esetben egy URI-ként értelmezzük. Az erőforrást azonosító URI-t megadhatjuk teljes alakban vagy használhatjuk a rövidített alakot.

#### *2.2.4.2 A köztes erőforrások XML-alakja*

Üres csomópont megadását többféleképpen tehetjük meg az RDF/XML szintaxisban. Egyik módszer az *rdf:parseType* attribútum használata, amellyel megváltoztathatjuk azon

tulajdonságelem interpretációját, amihez tartozik. Ha az attribútum értéke *Resource*, akkor az attribútum elem tartalmát egy új erőforrás (üres csomópont) leírásaként kell értelmezni anélkül, hogy megadnánk egy beágyazott *rdf:Description* elemet. Az alábbi kódrészletben az *isi:cime* tulajdonságelem *rdf:parseType="Resource"* attribútuma jelzi, hogy üres csomópontot kell kreálni az *isi:cime* tulajdonság értékeként, és az *isi:iranyitoszama*, *isi:helysegneve*, *isi:utcanev*, *isi:hazszam* az új csomópont tulajdonságai. Továbbá az <isi:iranyitoszama> attribútumaként *rdf:datatype*-t szerepel, amellyel literálok típusát adhatjuk meg.

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <isi:cime rdf:parseType="Resource">
                <isi:iranyitoszama rdf:dataType="xsd:int">8111</isi:iranyitoszama>
                <isi:helysegneve>Seregélyes</isi:helysegneve>
                <isi:utcanev>Fõ</isi:utcanev>
                <isi:hazszam>278</isi:hazszam>
        </isi:cime>
</rdf:Description>
```

Másik lehetőség a köztes erőforrás megadására lokális azonosítót használata, amit a *rdf:nodeID* attribútummal adhatok meg. Előbbi példa átírva:

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <isi:cime rdf:nodeID="lokalis_azon1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="lokalis_azon1">
        <isi:iranyitoszama rdf:dataType="xsd:int">8111</isi:iranyitoszama>
        <isi:helysegneve>Seregélyes</isi:helysegneve>
        <isi:utcanev>Fõ</isi:utcanev>
        <isi:hazszam>278</isi:hazszam>
</rdf:Description>
```

Bármelyik módszer is válasszuk, az RDF-gráf ugyanaz lesz.

#### *2.2.4.3 Relatív URI-k az RDF/XML-szintaxisban*

Az RDF/XML szintaxis megengedi a relatív URI-k használatát minden olyan esetben, amikor az abszolút URI-t is használhatnánk. Ugyanis az RDF XML-alakja esetén jól definiált a bázis URI. Az alapesetben a bázis URI a dokumentum saját URI-ja, de specifikálhatunk

ettől különböző bázist, az *xml:base* attribútummal.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                    xmlns:isi="http://delfin.unideb.hu/~tothviki/iskola#"
                    xml:base="http://delfin.unideb.hu/~tothviki/iskola">
```

A példában specifikált a bázis URI, így a dokumentum elhelyezkedésétől függetlenül mindig *"http://delfin.unideb.hu/~tothviki/iskola"* URI-val történik a dokumentumban található relatív URI-k feloldása.

#### *2.2.4.4 Az rdf:ID attribútum*

Az eddigi állításokban szereplő alanynak ismert volt az URI-ja, de elképzelhető olyan kijelentés, ahol az alanynak megfelelő erőforrás azonosító nem ismert. Az *rdf:ID* attribútum pont az előbb említett problémára ad megoldást. Az *rdf:ID* attribútumot az *rdf:about* attribútum helyett kell használni az RDF-kijelentések alanyainak azonosításához. Az alanyokhoz tartozó URI-t következő módon konstruáljuk: vesszük az érvényes bázis URI-t, hozzávéve a # karaktert, végül pedig az rdf:ID értékét

(példában *http://delfin.unideb.hu/~tothviki/iskola#tanar1* a konstruált abszolút URI).

```
<rdf:No rdf:ID="tanar1">
        <rdf:type rdf:resource="#Matektanar"/>
        <isi:nev>Tóth Viktória</isi:nev>
</rdf:No>
```

Az rdf:ID-vel azonosított erőforrásra a dokumentumon (azonos *xml:base* hatáskörön) belül hivatkozhatunk, a # karakter és az *rdf:ID* érték segítségével:

```
<rdf:Description rdf:ID="igazgato">
        <isi:nev>Butola Zoltán</isi:nev>
</rdf:Description>

<rdf:Ember rdf:about="#igazgato">
        <isi:iskolaDolgozoja rdf:resource="http://www.seregszaksuli.hu"/>
</rdf:Ember>
```

Lehetőség van külső dokumentumból vagy az *xml:base* hatáskörén kívülről hivatkozni egy *rdf:ID*-vel azonosított erőforrásra (abszolút URI hivatkozást kell alkalmazni):

```
<rdf:Ember rdf:about="http://delfin.unideb.hu/~tothviki/iskola#igazgato">
        <isi:iskolaDolgozoja rdf:resource="http://www.seregszaksuli.hu"/>
</rdf:Ember>
```

Az *rdf:ID* jelentősége abban áll, hogy nagyon hasznos lehet például akkor, ha egy dokumentumon belül különböző, ámde valamilyen szempont szerint mégiscsak összetartozó dolgok RDF-leírását szeretnénk megadni (példa esetén a tanárikarhoz tartozó személyeket szeretnénk összefogni). A különbözőséget az *rdf:ID* egy speciális tulajdonsága garantálja: egy *rdf:ID*-vel elnevezett azonosító csak egyszer szerepelhet az adott bázis URI hatáskörében, amelyet az RDF elemzőknek ellenőrizniük kell.

Fontos megjegyezni, hogy nem célszerű használni az alapesetként felkínált bázis URI-t, ha a dokumentumunkat több különböző helyen is elhelyezzük a világhálón. Ugyanis más és más alany azonosítót kapunk, tehát a különböző RDF-kijelentéseket fogalmazunk meg.

#### *2.2.4.5 Példányok RDF-ben*

Az RDF lehetőséget biztosít, hogy egy erőforrásról kijelenthessük, hogy egy adott osztály példánya. Erre az *rdf:type* tulajdonság szolgál, amelynek értéke azonosítja a kérdéses osztályt.

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <rdf:type rdf:resource="http://delfin.unideb.hu/~tothviki/iskola#Szakkepzo"/>
</rdf:Description>
```

A fenti példa a kijelentés tartalma, hogy Seregélyesi Szakképzőiskola az *Szakkepzo*.

Az RDF nem biztosítja a saját osztály definiálás lehetőségét, erre az RDF sémák alkalmasak. Az RDF nyújt viszont pár beépített osztályt, amelyeket felhasználhatunk példányok létrehozására, és amelyek segítik a magasabb rendű kijelentéseket, valamint a konténerek használatát.

A példányok kezelése nagyon gyakori, így az XML-szintaxis biztosít egy egyszerűbb írásmódot. Ilyenkor az *rdf:type* tulajdonságelemet elhagyjuk és lecseréljük az *rdf:Description* elemet *rdf:type* minősített névre.

```
<rdf:Szakkepzo rdf:about="http://www.seregszaksuli.hu">
        <isi:fenntarto rdf:resource="http://www.fejer.hu/"/>
</rdf:Szakkepzos>
```

Egy erőforráshoz tetszőleges számú rdf:type tulajdonság kapcsolódhat, ilyenkor az erőforrás több osztálynak példánya is egyszerre. Az egyszerűsített írásmódot azonban csak egyikük esetében használhatjuk, a többi hagyományos módon kell reprezentálni.

```
<rdf:No rdf:ID="tanar1">
        <rdf:type rdf:resource="#Matektanar"/>
        <isi:nev>Tóth Viktória</isi:nev>
</rdf:No>
```

#### *2.2.4.6 Magasabb rendű kijelentések*

Az RDF-ben lehetséges, hogy kijelentésekről fogalmazzunk meg kijelentéseket, amelyeket *magasabb rendű kijelentéseknek* hívunk.

Például a Seregélyesi Szakképzőiskola igazgatója kijelenti, hogy Tóth Viktória az Tóparti Gimnázium és Művészeti Szakközépiskola tanára. Ennek az állításnak az RDF megfelelője olyan kijelentés, amelynek az alanya az igazgató, predikátuma az, hogy állítja, míg a tárgya az a kijelentés, hogy Tóth Viktória az Tóparti Gimnázium és Művészeti Szakközépiskola tanára. Az RDF kijelentésben a tárgynak erőforrásnak kell lennie, ezért kell egy módszer arra, hogy a kijelentést egy speciális erőforrásként modellezzük. Ennek a folyamat neve reifikáció, a speciális erőforrásé pedig reifikált kijelentés.

Egy reifikált kijelentés az rdf:Statement osztály példánya.

```
<URI1>          rdf:type          rdf:Statement .
```

A fenti állítás jelentése, hogy a URI1 azonosítójú erőforrás egy retifikált kijelentés, amelynek alanyát az rdf:subject, predikátumát az rdf:predicate és tárgyát pedig az rdf:object tulajdonságelemekkel rendelhetjük hozzá.

```
<URI1>          rdf:subject       <http://delfin.unideb.hu/~tothviki/iskola#tanar1> .
<URI1>          rdf:predicate     isi:tanara .
<URI1>          rdf:object        <http://www.seregszaksuli.hu> .
```

Az eredeti kijelentésünk:

```
<http://delfin.unideb.hu/~tothviki/iskola#igazgato>      isi:allitja          <URI1> .
```

Példa RDF/XML-ben:

```
<rdf:Statement rdf:nodeID="lokalis_azon">
        <rdf:subject rdf:resource="#tanar1"/>
        <rdf:predicate  rdf:resource="http://delfin.unideb.hu/~tothviki/iskola#tanara"/>
        <rdf:object rdf:resource="http://www.toparti-szfvar.hu"/>
</rdf:Statement>

<rdf:Description rdf:ID="igazgato">
        <isi:nev>Butola Zoltán</isi:nev>
        <isi:allitja rdf:nodeID="lokalis_azon">
</rdf:Description>
```

Fontos tény, ha egy reifikált kijelentés jelen van a RDF-gráfban nem az jelenti feltétlenül azt, hogy az általa modellezett kifejezés is megtalálható benne. Nevezetesen, ha a példát alapul véve az RDF-leírás nem az állítja, hogy Tóth Viktória tanára a Tóparti Gimnázium és Művészeti Szakközépiskolának, hanem azt, hogy a Seregélyesi Szakképzőiskola igazgatója ezt állítja.

Egy reifikált kijelentés az RDF specifikáció szerint egy konkrét RDF hármasra egy konkrét gráfban vonatkozik, de ugyanazt az alanyt, predikátumot és tárgyat tartalmazó kijelentés több gráfban előfordulhat. Sajnos az RDF specifikáció nem ad lehetőséget arra, hogy reifikált kijelentést közvetlenül összekapcsoljuk azzal az állítással, amire vonatkozik.

#### *2.2.4.7 Konténerek*

A konténerek RDF-ben olyan erőforrások, amelyek dolgoknak egy csoportját azonosítja. Az RDF-nek van egy konténer szókészlete, amely három előre definiált típust tartalmaz (néhány ezekhez kapcsolódó tulajdonsággal együtt). Ezek a következők:

- Bag (rdf:Bag): Erőforrások vagy literálok olyan csoportja, ahol egy elem többször is előfordulhat, és az elemek sorrendje nem lényeges. Egy Bag konténer leírhatja például egy számítógépterembe lévő gépek leltározási számát.
- Sequence (rdf:Seq): Egy rendezett Bag, azaz olyan erőforrások vagy literálok csoportja, amelyben duplikált tagok előfordulhatnak és a tagok sorrendje szignifikáns. Egy Seq konténerre példa: egy számítógépterembe lévő gépek leltározási száma növekvő sorrendben.
- Alternative (rdf:Alt): olyan erőforrások vagy literálok nem rendezett csoportja,

amelyben egy elem többször is szerpelhet, és az elemek bizonyos szempontból egyenértékűek és felcserélhetőek egymással. Előbbi két konténertípussal ellentétben ez nem lehet üres, legalább egyetlen egy elemet kell tartalmazni. Az Alternative konténer első elem alapértelmezett, amelytől eltekintve a konténer elemeinek a sorrendje közömbös. Példa az Alt típusú konténerre: egy könyv címének különböző nyelvű fordításai.

Fontos megérteni, hogy a RDF konténerek használatakor, a kijelentéseink nem konstruálják a konténert, mint a programozási nyelvek adatstruktúrái esetén. A kijelentések csupán leírják a konténert. Konténer erőforrásról csak azt tudjuk megmondani, hogy milyen dolgokat tartalmaz, amelyeket a tagjainak tekintünk. Tehát azt nem jelenthetjük ki, hogy a leírt dolgokon kívül nincs más tagja a konténernek.

Egy erőforrásról az *rdf:type* tulajdonság segítségével jelenthetjük ki azt, hogy az egyike a fent definiált konténertípusoknak. Egy konténer elemeit speciális RDF-tulajdonságokkal írhatjuk le, amelyeknek alanya egy konténer típusú erőforrás és tárgya pedig a konténer éppen definiált tagja. Ezen tulajdonságok formátuma *rdf:_n*, ahol *n* egy 10-es számrendszerbeli természetes szám. Az *rdf:_n* helyett használhatjuk az *rdf:li* tulajdonságot. Egy konténer típusú erőforrásnak az rdf:type és az rdf:_n tulajdonságokon kívül lehetnek egyéb tulajdonságai. Példa egy Bag típusú konténerre:

```
<rdf:Description rdf:ID="tanarikar">
        <rdf:type rdf:resource=
                "http://www.w3c.org/1999/02/22-rdf-syntax-ns#Bag"/>
        <rdf:_1 rdf:resource="#tanar1"/>
        <rdf:_2 rdf:resource="#tanar2"/>
        <isi:letszam>24</isi:letszam>
</rdf:Description>
```

Egyszerűbb írásmóddal és az rdf:li használva:

```
<rdf:Bag rdf:ID="tanarikar">
        <rdf:li rdf:resource="#tanar1"/>
        <rdf:li rdf:resource="#tanar2"/>
        <isi:letszam>24</isi:letszam>
</rdf:Bag>
```

A tanárikar erőforrásról, ami egy Bag típusú konténerről állítjuk, hogy van két tagja és egy létszám tulajdonsága. Ugyanakkor ez nem azt jelenti, hogy a tanárikar erőforrásnak nem

lehet egyéb tulajdonsága illetve más tagjai. Hiszen egy másik RDF-forrás kijelentheti újabb erőforrásokról, hogy a tanárikarnak eleme. Fontos megjegyezni, ha például előbbi tanárikarról állítjuk, hogy aktívan közreműködött a diáknap szervezésében, nem jelenti azt, hogy minden tagja a tanárikarnak ténylegesen aktívan részt vállalt szervezésben.

A konténerek egyik szokásos alkalmazása, hogy kijelentések tárgyaként szerepelnek.

```
<rdf:Description rdf:about="http://www.seregszaksuli.hu">
        <isi:dolgozoi>
                <rdf:Bag>
                        <rdf:li rdf:ID="igazgatosag"/>
                        <rdf:li rdf:ID="tanarikar"/>
                        <rdf:li rdf:ID="technikaiSzemelyzet"/>
                </rdf:Bag>
        </isi:dolgozoi>
</rdf:Description>
```

A fenti példában a konténert üres csomóponttal azonosítottuk, így lényegében „zárttá” tettük. Hiszen az URI ismeretének hiányában más forrásból nem tudunk hivatkozni a konténerre. Továbbá a tulajdonság értékének megadásának újabb módját is tartalmazza a példa; konkrétan az isi:dolgozoi tulajdonság értékét, a konténert úgy adtuk meg, hogy az <rdf:Bag> elem az <isi:dolgozoi> elem tulajdonság-elemeként szerepel. A kódrészlet másik újdonsága az, amikor egy tulajdonságelemben használjuk az rdf:ID. Ez arra használható, hogy automatikusan előállítsa annak a triplettek a tárgyiasítását, amelyet a tulajdonságelem generál.

#### *2.2.4.8 Kollekciók*

Konténerek egyik hiányossága, hogy nem adnak lehetőséget arra, hogy lezárjuk őket, azaz csak az általunk megadott elemeket tartalmazza. Az *RDF-kollekciók* ezt a hiányosságot pótolják. A kollekció valójában nem más, mint egy lista. Egy konkrét kollekció az *rdf:List* osztály példánya. A kollekció első elemét az *rdf:first* tulajdonsággal adhatjuk meg. A maradék egy újabb kollekció, amelyet az *rdf:rest* tulajdonsággal kapcsolunk hozzá az aktuális kollekcióhoz. A lista végét egy *rdf:nil* értékkel rendelkező rdf:rest tulajdonság jelzi, ahol az rdf:nil egy rdf:List típusú üres listát reprezentál.

```
<rdf:Description rdf:about="http://iskola.hu">
    <isi:humanMunkakozosseg>
        <rdf:List>
            <rdf:first>Balogh Gábor</rdf:first>
                <rdf:rest>
                    <rdf:List>
                        <rdf:first>Nagy Ilona</rdf:first>
                        <rdf:rest rdf:resource="http://.../22-rdf-syntax-ns#nil"/>
                    </rdf:List>
                </rdf:rest>
        </rdf:List>
    </isi:humanMunkakozosseg>
</rdf:Description>
```

Az RDF/XML támogat a kollekcióknak egy egyszerűbb definiálását. Az kollekciók leírhatók az *rdf:parseType* tulajdonság elemmel, amelynek az attribútuma az *Collection*. Fontos, hogy csak az RDF-erőforrásokat tartalmazó kollekció esetén használható az egyszerűbb alak.

## 2.3 Az RDF séma (RDF Vocabulary Description Language)

Az RDF szemantikus kiterjesztése az RDF szókészlet leíró nyelv, azaz RDF séma. Az RDF séma lehetőséget adnak arra, hogy saját, alkalmazásspecifikus osztályokat és tulajdonságokat definiáljunk, valamint megadjuk az osztályok és tulajdonságok egymás közötti hierarchikus viszonyát, illetve ezek jellemzőit. Az ontológia nyelvek gazdagabb szókészlet leírók, mint az RDF séma.

### *2.3.1 Osztályok és tartalmazási viszonyaik*

Az RDF sémában *osztálynak* tekintünk, minden olyan erőforrást, amelyiknek a leírásában szerepel egy olyan rdf:type tulajdonság *rdfs:Class* erőforrás értékkel. Az rdfs:Class-ra úgy tekintünk, mint az osztályok osztálya.

```
<rdfs:Description rdf:ID="Ember">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdfs:Description>
```

Egy `A` osztályról az *rdfs:subClassOf* tulajdonság segítségével jelenthetjük ki, hogy *al-*

*osztálya* egy `B` osztálynak., azaz `A` minden példánya példánya `B`-nek is. RDF-ben egy osztálynak tetszőleges számú szülőosztálya és tetszőleges számú alosztálya lehet. Ha `A` alosztálya `B`-nek és `B` alosztálya `C`-nek, akkor `A` alosztálya `C`-nek,tehát az rdfs:subClassOf tulajdonság tranzitív.

```
<rdfs:Class rdf:ID="Nőnemű">
        <rdfs:subClassOf   rdf:resource="#Ember"/>
        <rdfs:comment>Nők</rdfs:comment>
</rdfs:Class>
```

Az *rdfs:comment* használatával egy osztályhoz az ember számára érthető rövid, szöveges magyarázatot fűzünk.

Egy erőforrásról példányosítással jelenthetjük ki, hogy egy adott osztálynak példánya. Egy erőforrás több osztálynak is lehet példánya.

### *2.3.2 Tulajdonságok és tartalmazási viszonyaik*

Az RDF-tulajdonság az *rdfs:Property* osztály egy példánya. Ennek megfelelően egy erőforrásról példányosítással jelenthetjük ki, hogy ez egy tulajdonság. Az RDF-ben lehetséges az *rdfs:subPropertyOf* tulajdonsággal megadni tulajdonságok közötti hierarchikus viszonyt. Az *rdfs:subPropertyOf* tranzitív, mivel ha `A` egy altulajdonsága `B`-nek, akkor minden olyan esetben, amikor `A` fennáll két erőforrás között, az is igaz, hogy `B` is fennáll.

```
<rdf:Description rdf:ID="iskolaDolgozója">
          <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
</rdf:Description>
```

```
<rdf:Property rdf:ID="tanára">
          <rdfs:subPropertyOf   rdf:resource="#iskolaDolgozója"/>
</rdf:Property>
```

Az így definiált tulajdonságokra minősített neveken keresztül hivatkozhatunk az RDF-dokumentumokban.

### *2.3.3 Tulajdonság értelmezési tartománya(domain), értékkészlete (range)*

Az RDF sémák lehetőségeket adnak arra, hogy az RDF adatok leírásakor mely tulajdonságokat , mely osztályokkal összefüggésben kívánunk használni. Sémákkal megadhatjuk egy

tulajdonság értelmezési tartományát (rdfs:domain) és/vagy értékkészletét (rdfs:range).

Egy tulajdonság értelmezési tartománya (értékkészlete) egy osztály. Az értékkészlet megkötésnek megadhatunk egy adattípust azonosító URI-t is. Ha a tulajdonsághoz nem tartozik rdfs:domain (rdfs:range) megkötés, akkor automatikusan az rdfs:Resourse osztály lesz, amely minden létező erőforrást tartalmaz. Ez azt jelenti, hogy ha egy tulajdonságra nem adunk meg értelmezési tartománybeli megkötést, akkor a tulajdonság tetszőleges osztályra alkalmazható. Hasonlóan, ha az értékkészletet nem specifikáljuk, a tulajdonság értéke tetszőleges erőforrás lehet. Egy tulajdonsághoz tartozhat több rdfs:domain (rdfs:range) is, ekkor a tulajdonság értelmezési tartománya (értékkészlete) a megadott osztályok metszete. Például az alábbi esetben olyan erőforrásnak lehet édesanya, amely egyszerre nőnemű és szülő:

```
<rdfs:Property rdf:ID="édesanya">
        <rdfs:domain rdf:resource="#Szülő"/>
        <rdfs:domain rdf:resource="#Nőnemű"/>
</rdfs:Property>
```

Az rdfs:domain és rdfs:range segítségével ún. globális tulajdonságkorlátokat írhatunk le, azaz miután specifikáltunk a megkötéseket a tulajdonságra, azok megváltoztathatatlanok és minden esetben érvényesek az adott tulajdonságra nézve.

# 3. RDF-források feldolgozása és lekérdezése

Ebben a fejezetben megmutatom, hogy az RDF-forrásokat milyen módon tárolhatjuk, illetve milyen nyelvek segítségével dolgozhatjuk fel és kérdezhetjük le.

## 3.1 RDF - források elhelyezése

Az RDF felhasználásának egyik természetes és elvárt módja az, hogy az általa nyújtott többletinformációk megteremtsék a szemantikus világháló alapjait. Az RDF segítségével az eddigieknél intelligensebb kereséseket tudjunk végezni a weben. Ehhez viszont nyilvánvalóan szükséges, hogy a keresőrendszerek valamilyen formában egyáltalán elérjék az RDF nyelven leírt metainformációkat. Mivel a keresőrendszerek amúgy is használnak olyan keresőrobotokat, amelyek periodikusan és folyamatosan járják a webet, érdemes az RDF-alakú metainformációkat egy honlap kontextusában, a honlapba ágyazva vagy ahhoz megfelelően csatolva megadni. Ekkor a keresőrendszerek a szokásos gyűjtés során juthatnak hozzá ahhoz a háttértudáshoz, amelyet felhasználhatnak a hozzájuk intézett kérdések megválaszolása során.

Egy honlaphoz társított RDF - információ nem feltétlenül szól magáról a honlapról vagy annak valamely részéről. Például egy Ausztráliában lévő honlap RDF - leírása kijelentheti, hogy egy Magyarországon élő ember szeme kék. Ezen információmorzsák konzisztens egésszé állnak össze egy keresőrendszerben. Kiemelkedő szerepet játszanak az URI-k, mivel ezek biztosítják a különböző RDF - gráfok csatlakozását azáltal, hogy egyértelműen azonosítanak erőforrásokat.

### *3.1.1 RDF - metaadat HTML-be ágyazva*

Az előző fejezetben láttuk, hogy az RDF - leírások szabványos módon megadhatók XML-alakban. Ugyan az RDF szemantikája szempontjából a gráf-, és nem az XML-

reprezentációja lényeges, a hordozhatóság és gépi feldolgozás azonban kellő létjogosultságot ad az XML-alaknak. Így a létező XML-elemzők minden változtatás nélkül képesek beolvasni egy RDF- leírást, hiszen azok szintaktikai szempontból valójában közönséges, jól formált XML-állományok. Az XHTML elterjedése miatt, ha az RDF-adatok a honlap részeként adottak, a létező keresőrobotok a szintaxis szempontjából felkészültek a azok kezelésére.

Az RDF-adatokat egy HTML-dokumentumban több helyre is beilleszthetjük. Ideális helynek tűnik a metaadatok elhelyezésére a HTML-oldalak feje, azaz <HEAD> és </HEAD> közötti rész. A keresőgépek biztosan beolvassák az indexelendő oldalak fejét, mert a HTML által biztosított alapszintű metainformációkat itt adhatjuk meg és ezek fontosak a keresők számára. A HTML-oldal fejrészben RDF-adatok megadásakor, használjuk az RDF egyik egyszerűsített írásmódját, amely lehetővé teszi, hogy az olyan RDF predikátum-tárgy párokat, ahol a tárgy literál, XML-attribútumérték formában írjuk. Azon okból, mivel a böngésző megjeleníti az egyszerű XML-elemek tartalmát.

Például:

```
<rdf:Description rdf:about="http://.../valami.html">
        <s:van>ITT VAN</s:van>
</rdf:Description>
```

E helyett:

```
<rdf:Description rdf:about="http://.../valami.html", s:van="ITT VAN" />
```

### *3.1.2 RDF-metaadat HTML-hez kapcsolva*

Másik elképzelés előzőtől gyökeresen különbözik: a weboldalakhoz nem társítunk metaadatokat, hanem linket adunk azokra. A keresőrobotok ezt a linket is ugyanúgy követik, mint minden más linket és végül eljutnak egy valódi RDF-forráshoz.

Architekturális szempontból ez tisztább megoldás. Nem fordul elő beágyazáskor fellépő problémák, könnyebb módosítani/karbantartani a metaadatokat.

A megfelelő linket elhelyezhetjük az oldalunk fejében a <link> elem használatával, aminek feladata, hogy összekapcsolja oldalunkat külső erőforrásokkal.

Például:

```
<head>
        <title>Ez egy honlap</title>
        <link rel="meta" type="application/rdf+xml" href="metadatok.rdf">
        <link rel="meta" type="application/rdf+xml" href="metadatok.n3">
</head>
```

A fenti példából látható, hogy linkelés további előnye, hogy segítségével az RDF-leírást többféle fizikai reprezentációban csatolhatjuk az oldalunkhoz.

## 3.2 Lekérdezés RDF-forrásokon

Az RDF- lekérdezések egyik legtermészetesebb módja, hogy XML-lekérdezőket (XQuery, XPath[2]) használnánk, hiszen az RDF- dokumentumokat általában ilyen alakban tároljuk. Fontos tény, hogy RDF esetén az XML pusztán csak szintaxis. Hiszen két, formailag különböző, ámde modellezési szempontból tökéletes XML-dokumentum is reprezentálhatja ugyanazt az RDF-gráfot. Emiatt nagyon oda kell figyelni arra, hogy a kérdéseket *hogyan* tesszük fel az XML-lekérdezőnek. Ugyanis egy adott kérdés megválaszolásához több lekérdezésre is szükség lehet, és így az XML/RDF-dokumentumokra csak korlátozott mértékben használhatók az XML-lekérdezők. Inkább olyan lekérdezőnyelveket érdemes használni, amelyek RDF logikai modelljén, az irányított gráfon dolgoznak.

### *3.2.1 RDF lekérdezőnyelvek sajátosságai*

Az RDF-lekérdezők megőrizték az SQL nyelv alapötletét, a *deklarativitást*[3]. Ezen felül a SQL-kérdések általános felépítése sok RDF-es keresőnyelvben megtalálható, persze vannak olyanok is, amelyek szakítottak ezzel a filozófiával. Továbbá az RDF-lekérdezőnyelveknek egyik közös jellemzője, hogy valamilyen mértékben támogatniuk kell az RDF-ben fellelhető adattípusokat.

A lekérdezőnyelveket különböző szempont szerint vizsgáljuk:

- *kifejezőerő*: milyen bonyolult kérdéseket fogalmazhatunk meg az adott nyelven

2 Mind két nyelvre úgy tekintenek, mint egy XML útkifejezésekkel kiegészített általános célú programozási nyelvre. Az útkifejezések csomópontokat azonosítanak az XML-fában.

3 A kérés feltevője azt specifikálja, hogy *mit* szeretne kérdezni, és nem azt, hogy *hogyan*.

- *zártság*: egy kérdésre adott válasz ugyanannak az adatmodellnek az eleme, mint amire maga a kérdés vonatkozott
- *ortogonalitás*: a nyelv által felkínált operációkat mennyire használhatjuk egymástól függetlenül tetszőleges környezetben
- *biztonságos*: tetszőleges kérdésre véges választ szolgáltat a nyelv

### *3.2.2 Adatbázis-alapú RDF-lekérdezők*

Az RDF-lekérdezők egyik fajtája az RDF-forrásokra úgy tekint, mint relációs vagy XML-adatbázisokra és ennek megfelelően alakítják ki a lekérdezőnyelvet és lekérdezési stratégiát. Ilyen lekérdezőket *adatbázis-alapú RDF-lekérdezőknek* nevezzük. Közös jellemzőjük, hogy feltételezik forrás struktúrájának pontos ismeretét ugyanúgy, mint egy SQL-kérdés esetén tudni kell az adatbázisban milyen táblák, milyen oszlopokkal találhatók.

#### *3.2.2.1 RDF Query Specification (RQS)*

1998-ban keletkezett RQS lekérdezőnyelv. Az alapötlete az, hogy a feltehető kérdések RDF-konténerekből RDF-konténereket[4] állítanak elő. Az eredménykonténer tartalma *részhalmaza* a kiindulási konténernek, szélsőséges esetben az eredménykonténer üres. Egy lekérdezés eredményét használhatjuk bemenetként egy újabb kérdés esetén, így lehetővé válnak az egymásba ágyazott kérdések. Tehát az RQL nyelv *zárt*.

Egy RQS-kérdés önmaga is egy RDF-forrás és minden esetben specifikálja a forráskonténert. Erre a rdfq:From URI által azonosított RDF-tulajdonság szolgál. A tulajdonság értékeként megadott konténer lesz maga a forrás. Amennyiben mást nem adunk meg, lekérdezés visszaadja a forráskonténer összes elemét.

Példa[5]:

4 Továbbiakban konténer alatt az RDF rdf:Bag konstrukciót értjük.

5 Az RQS példák *http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html* honlapról származnak. Megjegyzem, hogy a RQS nyelv kidolgozása az RDF ajánlás előtt történt meg, így a példák nem tükrözik a mai RDF elképzelést.

```
<rdfquery>
  <rdfq:From>
    <rdf:Bag>
      <li resource="http://www.research.ibm.com/people/ashok/paper1.html"/>
      <li resource="http://www.research.ibm.com/people/ashok/paper3.html"/>
      <li resource="http://www.research.ibm.com/people/neel/paper1.html"/>
      <li resource="http://www.research.ibm.com/people/neel/paper7.html"/>
    </rdf:Bag>
  </rdfq:From>
</rdfquery>
```

Az ilyen explicit módon történő forrásmegadással szemben sokkal gyakoribb, hogy *rdfq:From* segítségével csak rámutatunk arra a konténerre, ami lekérdezni kívánt erőforrásokat tartalmazza. Például[6]:

```
<rdf:Bag bagID="papers">
      <li resource="http://www.research.ibm.com/people/ashok/paper1.html"/>
      <li resource="http://www.research.ibm.com/people/ashok/paper3.html"/>
      <li resource="http://www.research.ibm.com/people/neel/paper1.html"/>
      <li resource="http://www.research.ibm.com/people/neel/paper7.html"/>
    </rdf:Bag>
```

Így a kérdésünket következőképpen tehetjük fel:

```
<rdfquery>
  <rdfq:From eachResource="papers"/>
</rdfquery>
```

Egy RQS-kérdésben a forráskonténeren kívül megadhatunk egy RDF-tulajdonságot is. A kérdés eredménye ekkor azon erőforrások halmaza lesz, amely erőforrások a forráskonténerben megtalálhatók és kapcsolódik hozzájuk valamilyen érték vagy másik erőforrás a megadott RDF-tulajdonságon keresztül. Az RDF-tulajdonságot az *rdfq:Property* tulajdonsággal adhatjuk meg az *rdfq:Select* elemen belül:

```
<rdfq:rdfquery>
  <rdfq:From eachResource="http://www.research.ibm.com/people/neel"/>
    <rdfq:Select>
      <rdfq:Property name="ResearchPapers"/>
    </rdfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

6 Az RDF ajánlásban nincs bagID, hanem helyette rdf:nodeID-t kell használni.

Kérdésünk a megadott emberek halmazából kiválogatja azokat, akinek van legalább egy publikációja.

Az SQL -nél megszokott projekciót az RQS nyelv esetén is alkalmazhatjuk. A relációs világban ilyenkor új, ideiglenes táblázatok jönnek létre; az RDF esetében új, ideiglenes erőforrásokat hozunk létre. A lekérdezni óhajtott tulajdonságneveket a *rdfq:Select* elem *rdfq:properties* attribútumában adhatjuk meg.

```
<rdfq:rdfquery>
  <rdfq:From eachResource="http://www.research.ibm.com/people/"/>
    <rdfq:Select properties="fullname experience">
       <rdfq:Property name="WebTechnologies/>
    </rfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

A példában olyan IBM-nél dolgozó kutatókat válogatunk ki, akik Webtechnológiai projekt résztvevői, és ezen erőforrásoknak csak a *teljes neve* és *tapasztalat* éleit tartjuk meg az eredmény RDF-gráfban.

A feltételeket a *rdfq:Select* konstrukción belül az *rdfq:Condition* elem jelöli ki.

```
<rdfq:rdfquery>
  <rdfq:From eachResource="http://www.research.ibm.com/people/">
  <rdfq:Select>
    <rdfq:Condition>
      <rdfq:greaterThan>
        <rdfq:Property name="Age" />
        <rdf:Integer>50</rdf:Integer>
      </rdfq:greaterThan>
    </rdfq:Condition>
  </rdfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

A példában azokat az IBM-nél dolgozó kutatókat keressük, akik 50 évnél idősebbek. A feltétel rész helyét az *rdfq:Condition* elem jelöli ki, magát a feltételt a jelen esetben az *rdfq:greaterThan* reprezentálja.

Az RQS nyelvben lehetőség van összetett feltételek megadására logikai elem használatával. Például azokat az IBM-nél dolgozó kutatókat szeretnénk kiválogatni, akik egy millió dollárnál nagyobb költségvetésű és kevesebb, mint 10 főből álló részlegen dolgoznak.

```
<rdfq:rdfquery>
 <rdfq:From eachResource="http://www.research.ibm.com/people">
 <rdfq:Select>
   <rdfq:Condition>
     <rdfq:and>
       <rdfq:greaterThan>
         <rdfq:Property name="Budget" />
         <rdf:Integer>1000000</rdf:Integer>
       </rdfq:greaterThan>
       <rdfq:lessthan>
         <rdfq:Property name="Size" />
         <rdf:Integer>10</rdf:Integer>
       </rdfq:lessthan>
     </rdfq:and>
   </rdfq:Condition>
 </rdfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

Sok esetben szükségünk van arra, hogy tulajdonságok mentén navigáljunk az RDF-gráfban. Például a következő beágyazott kérdés esetén:

```
<rdfq:rdfquery>
 <rdfq:From eachResource="http://www.research.ibm.com/people/">
 <rdfq:Select>
   <rdfq:Property name="BelongTo" />
   <rdfq:Select>
     <rdfq:Condition>
       <rdfq:greaterThan>
         <rdfq:Property name="Budget" />
         <rdf:Integer>1000000</rdf:Integer>
       </rdfq:greaterThan>
     </rdfq:Condition>
   </rdfq:Select>
 </rdfq:Select>
</rdfq:From>
</rdfq:rdfquery>
```

A példában azokat kutatókat válogatja ki, akik egy millió dollárnál nagyobb költségvetésű részlegen dolgoznak.

Az RQS nyelv lehetőséget ad rövidített jelölés használatára ilyen fajta navigáció esetén. Az útvonalat a *rdfq:path* attribútum értékeként kell megadni. A következő példában az IBM-nél dolgozó Neelnek azon publikációit szeretnénk kiválogatni, amelyek külföldön jelentek meg.

```
<rdfq:rdfquery>
 <rdfq:From eachResource="http://www.research.ibm.com/people/neel">
 <rdfq:Select>
  <rdfq:Condition>
   <rdfq:not>
    <rdfq:equal>
     <rdfq:Property path="ResearchPapers/Conference/Venue/Country"/>
     <rdf:String>U.S.A.</rdf:String>
    </rdfq:equal>
   </rdfq:not>
  </rdfq:Condition>
 </rdfq:Select>
 </rdfq:From>
 </rdfq:rdfquery>
```

Az RQS nyelv további lehetőségekkel is rendelkezik. Az eredménykonténerek által reprezentált erőforráshalmazoknak képezhetjük unióját, metszetét, az eredményeket sorba rendezhetjük. A példában Neel publikációit rendezzük év és hónap szerint.

```
<rdfq:rdfquery>
  <rdfq:From eachResource="http://www.research.ibm.com/people/neel">
   <rdfq:Select>
     <rdfq:Property name="ResearchPapers"/>
   </rdfq:Select>
   <rdfq:Order>
     <rdf:Seq>
       <rdfq:Property path="ResearchPapers/Year"/>
       <rdfq:Property path="ResearchPapers/Month"/>
     <rdf:Seq>
   </rdfq:Order>
  </rdfq:From>
</rdfq:rdfquery>
```

Használhatunk univerzális és egzisztenciális kvantálást bonyolultabb kérdések megválaszolásában. A következő példában azon kutatókat gyűjtjük ki akinek, 1998-ban van legalább egy publikációja.

```
<rdfq:rdfquery>
  <rdfq:From eachResource="Almaden_Researchers">
   <rdfq:Select>
     <rdfq:Condition>
       <rdfq:Quantifier type="exists" var="x">
          <rdfq:Property path="ResearchPapers"/>
          <rdfq:equals>
            <rdfq:Property var-ref="x" name="Year"/>
            <rdf:Integer>1998</rdf:Integer>
          </rdfq:equals>
        </rdfq:Condition>
      </rdfq:Quantifier>
    <rdfq:Condition>
   </rdfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

Tehát a nyelv kellően *nagy kifejezőerejű*. Az RDF-konstrukciókat megfelelően támogatja és *biztonságos* is, mert a válasz mindenképpen a kiindulási konténer része.

AZ RQS nem terjedt el, mivel egyrészt a nyelv zártsága miatt bonyolultabb kérdéseket nem vagy nagyon nehezen lehet feltenni; másrészt a nyelv feltételezi, hogy a kérdező ismeri az RDF-forrás felépítését. Ez utóbbit nem lehet elvárni, hiszen metaadatok esetén sokszor az ismeretlen dolgokra vonatkoznak a kérdések.

### *3.2.3 Modell-alapú RDF-lekérdezők*

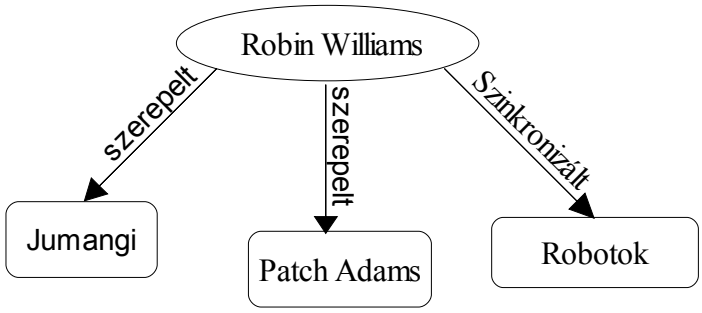
Az RDF adatmodell egy irányított gráf, amelynek élei URI-kal címkézettek, csomópontjai vagy üresek vagy literálok vagy pedig URI-val címkézettek. A lekérdezőnyelvek másik fajtája *modell-* vagy *tudásbázis-alapú* lekérdezők, amelyek közvetlenül az adatmodellen dolgoznak. Az ilyen típusú lekérdezőkre jellemző, hogy figyelembe veszik az RDF-hez és az RDF sémához társítható szemantikát is. Például, ha egy bizonyos osztály példányait keressük, kaphatunk olyan példányokat is, amelyek az adott osztály valamely alosztályának példányai. További tulajdonsága a modellalapú lekérdezőknek, hogy nem feltételezik az RDF-állításokat tároló adatbázis (*tudásbázis*) szerkezetének pontos ismeretét. A kérés feltevőjének nem kell tudni a kérdés feltételének pillanatában, hogy milyen kapcsolatok lehetségesek az egyes erőforrások között.

### *3.2.3.1 Általánosan a modell-alapú lekérdezőnyelvekről*

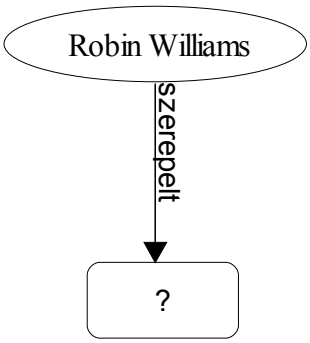
A tudásbázis-alapú lekérdezők mindegyike a részgráfillesztést és változóbehelyettesítés technikáját alkalmazza. Ebben a szemléletben a kérdések önmaguk is irányított, címkézett gráfok (*gráfminta*), amelyek azonban tartalmazhatnak változót. Ezen változók kétféleképpen lehetnek jelen a gráfban:

- *csomópontként*: ekkor a változó egy olyan RDF-erőforrást jelöl, amely ismeretlen a kérdés feltevésekor
- *élként*: ekkor a változó egy ismeretlen RDF-tulajdonságot jelöl

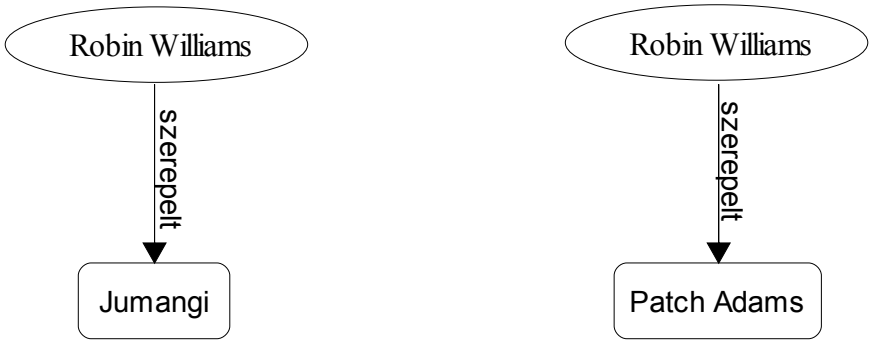
Példa egyszerű tudásbázisra:




Kérdés: Milyen filmben szerepelt a Robin Williams?




Válasz (két részgráf lesz):



A kérdéseket egy előre kialakított RDF-tudásbázisra vonatkozóan tehetjük fel. A válasz egy olyan részgráfja a tudásbázisnak, amely azonos a feltett kérdéssel valamilyen változó-helyettesítés(ek) mellett. Ugyanakkor a válasz maga is egy RDF-tudásbázis, amelyre vonatkozóan újabb kérdés tehető fel; azaz kiindulási RDF-tudásbázisból egy RDF-tudásbázist kapunk eredményül. Így modell-alapú lekérdezők *zártak*.

A legegyszerűbb gráfminta egy kétpontú, összefüggő, címkézett, irányított egyszerű gráf, ahol mind az él, mind a két végpont címkéje lehet változó[7]. Egy ilyen gráfminta megfelel egy változókat is tartalmazó RDF-hármasnak. Ilyen gráfokat hívjuk *primitív kérdésnek* vagy *részkérdésnek*. Összesen 8 formailag különböző primitív kérdés képzelhető el attól függően, hogy mely csúcspontok helyén szerepel változó, illetve változó-e az él. Észrevehető, hogy egy primitív kérdés az RDF bináris relációjának kibővítése annyiban, hogy változókat megenged ott, ahol az RDF-forrásban URI-knak kell lennie. Egy ilyen kérdésre a válasz a változók egy lehetséges behelyettesítése vagy csak egy egyszerű igen/nem a válasz, ha a kérdés nem tartalmazott változót.

Az a modell-alapú RDF-lekérdezőnyelv nem megfelelő, amelyben csak primitív kérdéseket tehetünk fel, így csupán arra kérdezhetnénk rá, hogy egy adott, részlegesen kitöltött RDF-kijelentés szerepel-e a tudásbázisban. Éppen ezért, az tudásbázis-alapú nyelvek lehetőséget adnak komplex gráfminták megadására is. Megengedik, hogy logikai "és" kapcsolatot fogalmazzunk meg két primitív kérdés között; amelynek a jelentése az, hogy mindkét gráf szerepel a tudásbázisgráfban. A részkérdések közös változókon keresztül kapcsolódhatnak egymáshoz, így *összetett kérdést* hoznak létre.

Egy összetett kérdés feldolgozásának lépései a következők:

- az első részkérdésnek megfelelő gráfot megkeressük a tudásbázisban
- találat esetén a megfelelő változókat behelyettesítjük, a kérdés feldolgozásának hátralévőrészében ezek már megváltoztathatatlanok
- rekurzívan folytatjuk a következő részkérdéssel (ha van ilyen)

Amennyiben az *n.* részkérdésnek megfelelő gráfot nem sikerül illeszteni a tudásbázishoz (nem szerepel a részgráf a tudásbázisban), visszalépés történik:

- az *n-1.* részkérdésnek megfelelő illesztésből származó változóhelyettesítéseket töröljük
- az *n-1.* részkérdésnek megfelelő gráfot megpróbáljuk másképpen illeszteni a

7 Továbbiakban egy él címkéjére élként, egy csomópont címkéjére csomópontként hivatkozunk.

tudásbázishoz

- siker esetén a megfelelő változók behelyettesítődnek és folytatjuk a feldolgozást az *n.* részkérdésnél
- kudarc esetén folytatjuk ezt a rekurziót *n-2.* részkérdéssel

Egy összetett kérdés *meghiúsult*, ha a rekurzióban visszaérünk az első részkérdéshez, és már nem tudjuk megtalálni a tudásbázisban, ahogy eddig előtte még sosem. Speciális esete ennek, ha egyszer se tudjuk illeszteni az első részkérdést a tudásbázisra.

Egy összetett kérdés sikerül, ha található olyan változóbehelyettesítés-kombináció, melyben minden, a kérdésben található változó behelyettesített, és az ilyen behelyettesítések mellett mindegyik részkérdésnek megfelelő kétpontú gráf jelen van a tudásbázisgráfjában. Siker esetén lehetőség van új válasz kérésére, ha eddigi részkérdésekhez felvesszük egy ál-részkérdést, aminek mi tudjuk szabályozni a sikerült/nem sikerült voltát. Alapesetben ez a primitív kérdés sikerült, de új megoldás kérés esetén ez meghiúsul, és az utolsó részkérdés által okozott változóbehelyettesítések törlődnek, s megpróbáljuk az utolsó részkérdést máshogy illeszteni az előbbi rekurzió szerint.

A névtelen erőforrásokat linearizáláskor el kell nevezni az esetleg fellépő ütközés elkerülése miatt. Ugyanezen okok miatt ugyanígy kell eljárnunk akkor is, ha kérdésre kapott válaszban szerepel névtelen erőforrás.

A kétpontú "és" kapcsolatban lévő részkérdések megfeleltethetők erdőnek. A legtöbb esetben az erdő komponensei fák, de általánosan tetszőleges gráfok. Azon részkérdésekből lesz összefüggő fa vagy gráf, amelyek kapcsolódnak valamilyen közös változón/változókon keresztül, és azok kerülnek külön komponensbe, amelyek nem. Tehát a komponensek nem függnek egymástól, így azokat a részkérdéseket, amelyek egy komponensbe tartoznak, mintegy új összetett kérdésként hajthatjuk végre. Ebből következik, hogy a komponensek léte nem befolyásolja a kérdések kifejezőerejét, elég arra koncentrálni, amikor egy kérdés nem erdő, hanem sima fa vagy összefüggő gráf.

#### *3.2.3.2 Az RDQL nyelv*

Az RDQL (*RDF Data Query Language*) egy modell-alapú lekérdezőnyelv, amely egy hosszú fejlődés eredményeként jött létre és a HP Jena rendszerének része.

Az RDQL nyelvnek a szintaxisa SQL-szerű, annak ellenére, hogy maga a nyelv kifejezetten nem adatbázisos szemléletű.

Egy RDQL-kérdés felépítése a következő:

SELECT Változók
FROM Dokumentumok
WHERE Minta
AND Szűrők
USING Névterek

A SELECT részben azokat a változókat kell megadnunk, amelyek érékeire kíváncsiak vagyunk a kérdés végrehajtásakor. A FROM részben kell kijelölnünk azokat az RDF-forrásokat, amelyekre a kérdésünk vonatkozik. A WHERE részben adjuk meg a gráfmintát (változókat is tartalmazhat), amely alapján a tényleges keresés történik. Az AND részben korlátokat adhatunk meg az előforduló változókra, például kiköthetjük milyen nevű embert keressen. Végül a USING részben pedig azokat a névtereket adhatjuk meg, amelyekre rövidített névvel szeretnénk hivatkozni a gráfmintában.

A példában keressük azokat az erőforrásokat, amelyeknek a John Smith a teljes nevük.[8]

```
SELECT ?x
WHERE (?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  "John Smith")
```

A változókat kérdőjellel kell kezdeni.

Válasz:

```
x
============================
<http://somewhere/JohnSmith/>
```

A második példában olyan személyeket keresünk, akiknek tudjuk a teljes nevét abban az értelemben, hogy van egy FN tulajdonság alanyaként és tárgyaként szerepelnek a függelékben megadott tudásbázisban. Az eddigi példákban egyszerű gráfmintákat használtunk, azaz minden olyan hármast kiválogattunk a tudásbázisból, amely illeszkedett a megadott mintára.

```
SELECT ?x, ?fname
WHERE (?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  ?fname)
```

8 A példák *http://jena.sourceforge.net/tutorial/RDQL/index.html* weboldalról származnak. Függelékben megtalálható vc-db-1.rdf dokumentumra vonatkozó lekérdezések szerepelnek a példákban.

A következő példában szereplő kérdés már összetett gráfmintát használ. Az adott személyek keresztneveit keressük. Az első gráfminta a neveket válogatja, a második meg minden egyes névhez megkeresi a keresztnevet.

```
SELECT ?resource  ?givenName
WHERE (?resource  <http://www.w3.org/2001/vcard-rdf/3.0#N>   ?z)
    (?z  <http://www.w3.org/2001/vcard-rdf/3.0#Given>  ?givenName)
```

A két minta a közös *z* változón keresztül kapcsolódik.

Az előbbi kérdést felírhatjuk a következő módon:

```
SELECT ?resource,  ?givenName
WHERE (?resource, vCard:N, ?z),
    (?z,  vCard:Given,   ?givenName)
USING vCard FOR <http://www.w3.org/2001/vcard-rdf/3.0#>
```

A USING elem segítségével példában szereplő prefixhez egy rövidebb nevet rendelünk, amit felhasználunk a minta megadásakor.

A következő kérdésünkben olyan személyek keresztnevét keressük, akik 24 évesek vagy idősebbek.

```
SELECT ?resource, ?familyName
WHERE (?resource  info:age  ?age)
    (?resource, vCard:N  ?y) , (?y, <vCard:Family>, ?familyName)
AND ?age >= 24
USING  info  FOR <http://somewhere/peopleInfo#>
     vCard FOR <http://www.w3.org/2001/vcard-rdf/3.0#>
```

Eddigi kérdések a kijelentések alanyára illetve tárgyára kérdeztek rá, de megfogalmazható olyan kérdés, amely egy tulajdonságra kíváncsi. Például milyen összefüggés van a John Smith literál és http://somewhere/JohnSmith URI-val rendelkező erőforrás között:

```
SELECT ?prop
WHERE (<http://somewhere/JohnSmith/> , ?prop, "John Smith")
USING  info  FOR <http://somewhere/peopleInfo#>
     vCard FOR <http://www.w3.org/2001/vcard-rdf/3.0#>
```

Sokan az RDQL hiányosságának tartják, hogy SQL szintaxist használ. Így nem különül el eléggé az adatbázis-alapú RDF-lekérdezőktől. További hibának róják fel, hogy nem RDF-alakban kell megadni a kérdéseket, sőt még a válaszok se RDF-alakban adottak.

A különböző RDF-konstrukciók, mint például a példányok, konténerek kezelése sokszor nehézkes, mivel a lekérdezendő RDF-forrást tisztán gráfként értelmezi, és alapesetben nem tulajdonít semmilyen jelentést az élek címkéjének.

#### *3.2.3.3 A SPARQL lekérdezendőnyelv*

2004 februárjában az RDF elképzelés magalkotója, a W3C, megalkotta az RDF-lekérdezésekkel foglalkozó munkacsoportot. A csoport célja, hogy megalkosson egy egységes RDF lekérdezőnyelvet, amely ötvözi a meglévő nyelvek legfontosabb tulajdonságait és a tervek szerint olyan etalonként szolgál majd, mint a relációs világban az SQL.

A munkacsoport 2004 októberében kiadta a SPQRQL elnevezésű RDF-lekérdezőnyelv első munkaváltozatát. Jelenleg a SPARQL-specifikációk W3C-javaslattá ( *Recommendation*) léptek elő. A SPARQL (kiejtése az angol "sparkle" szónak megfelelően) módszert kínál a fejlesztőknek és a végfelhasználóknak a keresési kritériumok megfogalmazásához és a találati listák feldolgozásához az információfajták széles spektruma esetében, mint pl. a személyes adatok, az emberi kapcsolatok hálója és a digitális alkotásokról (zenéről, képekről stb.) szóló metaadatok.

A SPARQL nyelv egy modell-alapú lekérdező. Lehetőség van a válaszokat nemcsak változó hozzárendelések formájában megkapni, hanem kérhetjük magát a talált részgráfot is, mint egy RDF-modellt.

Arra is lehetőségünk van, hogy a keletkezett változóbehelyettesítések alkalmazásával új RDF-gráfot konstruáljunk (CONSTRUCT); továbbá lekérhetjük egy erőforrásról annak jellemzőit (milyen élek milyen értékkel kapcsolódnak hozzá) (DESCRIBE), illetve igen/nem kérdésként eldönthetjük, hogy a megadott gráfmintánk kielégíthető-e, azaz létezik-e olyan változóbehelyettesítés, amely felhasználásával a kérdés azonos alakú lesz a tudásbázis egy részgráfjával (ASK). Tehát a SPARQL nyelv négy fajta lekérdezést különböztet meg.

A SPARQL nyelv külön gondot fordít az RDF-literálok kezelésére. Megadásuk hasonlóan történik, mint az RDQL nyelvnél, azaz "" vagy ' ' között. A literál esetén megadhatjuk annak nyelvét (@) és típusát (^^).

A változók jelölése szintén megegyezik az RDQL nyelvvel. Az üres csomópontra való hivatkozás pedig a tripletteknél bemutatott módon történik, azaz "_:a" vagy pedig "[]" jelöljük a névtelen csomópontot. Másik gráfból is hivatkozni a szeretnénk az üres csomópontunkra, akkor [:*p* „*v*”] alakban kell megadni. Az üres csomópont annak a kijelentésnek az alany lesz, amelynek állítmánya *p* és tárgya *v*.

A következőben ugyanazt azt a kérdést fogom feltenni, azaz mi a címe annak a könyvnek, amelyet a *http://example.org/book/book1* URI-val azonosítunk.[9]

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <http://example.org/book/book1> dc:title ?title }
```

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX  : <http://example.org/book/>
SELECT  $title
WHERE   { :book1  dc:title  $title }
```

```
BASE    <http://example.org/book/>
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  $title
WHERE   { <book1>  dc:title  ?title }
```

```
BASE    <http://example.org/book/>
PREFIX  dcore:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <book1> dcore:title ?title }
```

Az utolsó kettő példában előfordul a bázis URI-val történő hivatkozás. Az összetett gráfmintákat '{' '}' között kell megadni.

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE   { ?x foaf:name ?name .
          ?x foaf:mbox ?mbox }
```

A kérdéssel a tudásbázisban lévő személyek neveit és e-mail címüket akarjuk megtudni.

A SPARQL nyelvben is megadhatunk feltételeket az egyes változókra.

9 A példák a *http://www.w3.org/TR/rdf-sparql-query/* honlapról származnak.

Például:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
                FILTER (?price < 30) .
          ?       x dc:title ?title . }
```

Olyan könyveknek (erőforrások) a címét és árát keressük, amelyek nem kerülnek többe, mint 30.

A nyelv lehetőséget biztosít opcionális gráfminták megadására, mely alatt azt kell érteni, hogy keresés végrehajtásakor ne kötelezően rendeljen a változókhoz helyettesítési értéket valamennyi megoldás esetén.

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x dc:title ?title .
        OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) }
      }
```

Az előző kérdést opcionális módon fogalmazzuk meg, tehát azoknak a könyvek címére is kíváncsiak vagyunk, amelyeknek nincs meghatározva az ára. Az opcionális gráfminták számára nincs megszorítás, azaz tetszőleges számú lehet egy lekérdezésben.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?foafName ?mbox ?gname ?fname
WHERE
  {  ?x foaf:name ?foafName .
     OPTIONAL { ?x foaf:mbox ?mbox } .
     OPTIONAL {  ?x vcard:N ?vc .
                 ?vc vcard:Given ?gname .
                 OPTIONAL { ?vc vcard:Family ?fname }
               }
  }
```

A kérdéssel olyan foafName tulajdonsággal rendelkező személyeket (erőforrásokat) válogatunk ki, amelyekhez megadjuk a mbox, gname illetve fname tulajdonságot, ha az erőforrás rendelkezik vele.

Képezhetjük két gráfminta unióját. A kérdés eredményében szerepelnek azok a

részgráfok, amelyek megoldásai a valamelyik részgráfillesztésnek. A példában a kérdés olyan könyv címeket válogat ki két RDF-gráfból.

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION
           { ?book dc11:title  ?title }
         }
```

A GRAPH kulcsszóval megadhatjuk milyen gráfokat kell használni az egyes illesztésnél, pontosabban nevesített a gráfra tudok hivatkozni.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?name ?mbox ?date
WHERE
 {  ?g dc:publisher ?name ;
       dc:date ?date .
   GRAPH ?g
     { ?person foaf:name ?name ; foaf:mbox ?mbox }
 }
```

Kérdés azon személyek nevét, e-mail címeit gyűjti ki, amelyeknek megváltozott az e-mail címük.

Az alapértelmezett gráfot a FROM utasításrészekben kell megadni, amely létrejöhet több gráf uniójából is. Példa:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name
FROM    <http://example.org/foaf/aliceFoaf>
WHERE   { ?x foaf:name ?name }
```

A SELECT lekérdezés szintaxisa a következő:

```
'SELECT' 'DISTINCT'? (változó+ | '*')
('FROM' (IRI_hivatkozás | 'NAMED' IRI_hivatkozás))*
'WHERE'? '{' gráfminta '}'
('ORDER' 'BY' rendezési_feltétel+)?
('LIMIT' egész_érték)?
('OFFSET' egész_érték)?
```

Használhatók a következő módosító szavak: projekció, DISTINCT, ORDER BY, LIMIT, OFFSET.

A projekcióban megadott feltételeknek eleget tevő elemek szerepelhetnek. A DISTINCT utasítást az ismétlődések kiszűrésére használjuk. Az ORDER BY utasítás esetében többrendezési feltételt is megadhatunk, amelyek változók vagy függvényhívások. Rendezést a numerikus, string vagy dateTime típusú értékeken tudjuk értelmezni. A LIMIT kulcsszó után egy nemnegatív számot kell megadni, amellyel a megoldások számát határozzuk meg. Végezetül az OFFSET kulcsszó után megadott nemnegatív szám, a visszaadandó megoldás sorszámát adja meg.

A CONSTRUCT lekérdezéssel egy RDF-gráfot kapunk a megadott sablonból. Az eredmény gráfot egy üres gráfból állítja elő úgy, hogy lekérdezésben szereplő minta mindegyik megoldására el kell végezni az alábbiakat:[10]

- Az adott megoldásban értéket kapott változók helyettesítési értékeit be kell helyettesíteni a sablon hármasaiba, és az így nyert hármasokat hozzá kell adni az eredmény gráfhoz
- Ha a sablon egy hármasában olyan változók szerepelnek, amelyek nem kapnak értéket egy megoldásban, vagy pedig nem érvényes RDF konstrukció jön létre – amelyben például literál szerepel alany pozícióban – , akkor ez a hármas nem kerül bele az eredménybe
- A sablonban szerepelhetnek olyan hármasok, amelyben egyáltalán nincsenek változók, értelemszerűen ezek is belekerülnek az eredmény gráfba

A szintaxisa a következő:

'CONSTRUCT' '{' *sablon* '}'
('FROM' (*IRI_hivatkozás* | 'NAMED' *IRI_hivatkozás*))*
'WHERE'? '{' *gráfminta* '}'
('ORDER' 'BY' *rendezési_feltétel*+)?
('LIMIT' *egész_érték*)?
('OFFSET' *egész_érték*)?

A ORDER BY, LIMIT, OFFSET utasítás ugyanaz, mint a SELECT lekérdezésnél.

10 A definíció *http://www.inf.unideb.hu/~jeszy/download/semweb/sparql.pdf* honlapról származik.

```
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
CONSTRUCT { $book dc:title $title }
WHERE
  { $book dc:title $title }
```

Az SPARQL nyelvnek van két további lekérdezési módja: ASK és a DESCRIBE. Az előbbi lekérdezésre egy igen/nem választ kapunk attól függően, hogy gráfmintának van e megoldása. Szintaxisa:

```
'ASK'
('FROM' (IRI_hivatkozás | 'NAMED' IRI_hivatkozás))*
'WHERE'? '{' gráfminta '}'
```

Példa:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
ASK WHERE { ?book dc:creator "J.K. Rowling"}
```

A DESCRIBE lekérdezéssel a megoldásában szereplő erőforrásokat írja le egy gráffal. Szintaxisa:

```
'DESCRIBE' ((változó | IRI_hivatkozás)+ | '*')
('FROM' (IRI_hivatkozás | 'NAMED' IRI_hivatkozás))*
'WHERE'? '{' gráfminta '}'
('ORDER' 'BY' rendezési_feltétel+)?
('LIMIT' egész_érték)?
('OFFSET' egész_érték)?
```

Példa:

```
PREFIX books: <http://www.example/book/>
DESCRIBE books:book6
```

A gyakorlatban lokálisan a SPARQL beleépült adott programozási környezetekbe (pl.:Jena). A programnyelvbe való beépülés természetesen nyelvfüggő: tartalmazhat egy SPARQL értelmezőt, vagy felajánlhat egy lokális API-t a SPARQL motornak.

# 4. Összefoglalás

A szemantikus világhálót két alapötletre építették fel: az egyik az, hogy metainformációt társítsunk minden internetes erőforráshoz; és a másik az, hogy előbbiekből következtetni lehessen. Így a számítógépek számára is feldolgozhatóvá válnak az internetes erőforrások. A meteainformációkat a gépi feldolgozás érdekében XML alakban tároljuk. Az XML nem alkalmas arra, hogy az alkalmazások nélkülözhessék a kommunikáció előtti egyeztetést, az átvitel során felhasznált nyelv jelentéséről.

Ezen problémára a megoldás: az RDF. Dolgozatomban részletesen bemutattam az RDF elképzelést. Az RDF nyelv alkalmas arra, hogy tetszőleges URI-val azonosított erőforráshoz jelentést rendeljen. Az URI-k segítségével egyértelmű állításokat fogalmazhatunk meg. Az RDF nyelv alapötlete az, hogy az URI-val azonosított erőforrásokat tulajdonságok segítségével más erőforrásokhoz vagy literálokhoz köti össze, ezt a hármast nevezik RDF-kijelentésnek. A hármasokból képezhetjük az RDF-gráfot, amely a nyelv szempontjából elsődleges reprezentációnak tekintünk. A számítógépek számára RDF nyelv XML alapú reprezentációja fontos.

Ugyanakkor magában az RDF nyelv nem elegendő ahhoz, hogy következtetéseket vonhassunk le a metainformációkból. Az RDF Sémában a dolgok jelentését úgy adjuk meg, hogy formálisan specifikáljuk milyen viszonyban van más entitással. A sémák lehetőséget biztosítanak saját osztály és tulajdonság definiálására.

Végezetül az RDF feldolgozását tárgyaltam. Az első RDF lekérdezők adatbázisos szemléletben tekintettek az RDF-forrásokra, míg az újabbak már RDF adatmodellen dolgoznak. Az adatbázis-alapú RDF lekérdezők közül RQS-t, az adatmodell szemléletű lekérdezők közül pedig SPARQL és RDQL nyelvet mutattam be.

# 5. Irodalomjegyzék

[1] Szeredi Péter, Lukács Gergely, Benkő Tamás : A szemantikus világháló elmélete és gyakorlata(2005)

[2] Gottdank Tibor : Szemantikus web(2005)

[3] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[4] http://hu.wikipedia.org/wiki/URI

[5] http://www.w3.org/TR/rdf-schema/

[6] http://www.w3.org/Submission/RDQL/

[7] http://www.w3schools.com/rdf/default.asp

[8] http://www.w3.org/TR/rdf-sparql-query/

[9] http://www.w3.org/2001/11/13-RDF-Query-Rules/terms

[10] http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html#Inference

[11] http://www.inf.unideb.hu/~jeszy/download/semweb/sparql.pdf

[12] http://www.w3.org/2001/11/13-RDF-Query-Rules/

[13] http://www.infoera.hu/infoera2007/ea/szemantikusweb.ppt

[14] http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html

[15] http://jena.sourceforge.net/tutorial/RDQL/index.html

[16] http://www.w3.org/TR/rdf-sparql-query/

# 6. Függelék

*vc-db-1.rdf*

```
<?xml version='1.0' encoding='UTF-8'?>

<rdf:RDF
 xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:vCard='http://www.w3.org/2001/vcard-rdf/3.0#'
 xmlns:info='http://somewhere/peopleInfo#'
  >

 <rdf:Description rdf:about="http://somewhere/JohnSmith/">
  <vCard:FN>John Smith</vCard:FN>
  <info:age>25</info:age>
  <vCard:N rdf:parseType="Resource">
 <vCard:Family>Smith</vCard:Family>
 <vCard:Given>John</vCard:Given>
  </vCard:N>
 </rdf:Description>

 <rdf:Description rdf:about="http://somewhere/RebeccaSmith/">
  <vCard:FN>Becky Smith</vCard:FN>
  <info:age>23</info:age>
  <vCard:N rdf:parseType="Resource">
 <vCard:Family>Smith</vCard:Family>
 <vCard:Given>Rebecca</vCard:Given>
  </vCard:N>
 </rdf:Description>

 <rdf:Description rdf:about="http://somewhere/SarahJones/">
  <vCard:FN>Sarah Jones</vCard:FN>
  <vCard:N rdf:parseType="Resource">
 <vCard:Family>Jones</vCard:Family>
 <vCard:Given>Sarah</vCard:Given>
  </vCard:N>
 </rdf:Description>

 <rdf:Description rdf:about="http://somewhere/MattJones/">
  <vCard:FN>Matt Jones</vCard:FN>
  <vCard:N
 vCard:Family="Jones"
 vCard:Given="Matthew"/>
 </rdf:Description>
</rdf:RDF>
```