Zpráva k projektu

Obsah

Popis projektu

Cílem projektu je vytvoření dat restauračního informačního systému v jazyce RDF a implementace alespoň 20 netriviálních dotazů v jazyce SPARQL.

Popis entit

Informační systém obsahuje **restaurace**, které jsou vždy umístěné na strategické **lokaci** (především ve hlavních městech a významných dopravních uzlech).

Restaurace mají **otevírací dobu** a jsou spravovány **manažery**, kdy vždy právě jeden manažer řídí minimálně jednu restauraci.

Manažer má pod sebou jednotlivé **zaměstnance**, u kterých evidujeme **jméno**, **příjmení**, **plat** a jestli umí daný zaměstnanec vařit, nebo je jen číšník *(tedy všichni zaměstnanci jsou číšníci, ale někteří navíc i kuchaři)*. Zaměstnanec vždy pracuje právě v jedné restauraci, pod jedním manažerem.

V každé restauraci jsou **stoly** s daným počtem míst a typem stolu (barový, uvnitř, venkovní…) Každý stůl je právě v jedné restauraci a je obsluhován právě jedním číšníkem. Číšník může obsluhovat více stolů, ale nemusí obsahovat žádný *(je i kuchař a právě vaří, připravuje nápoje v zázemí restaurace…)*. Celkový počet míst všech stolů v restauraci určuje její kapacitu.

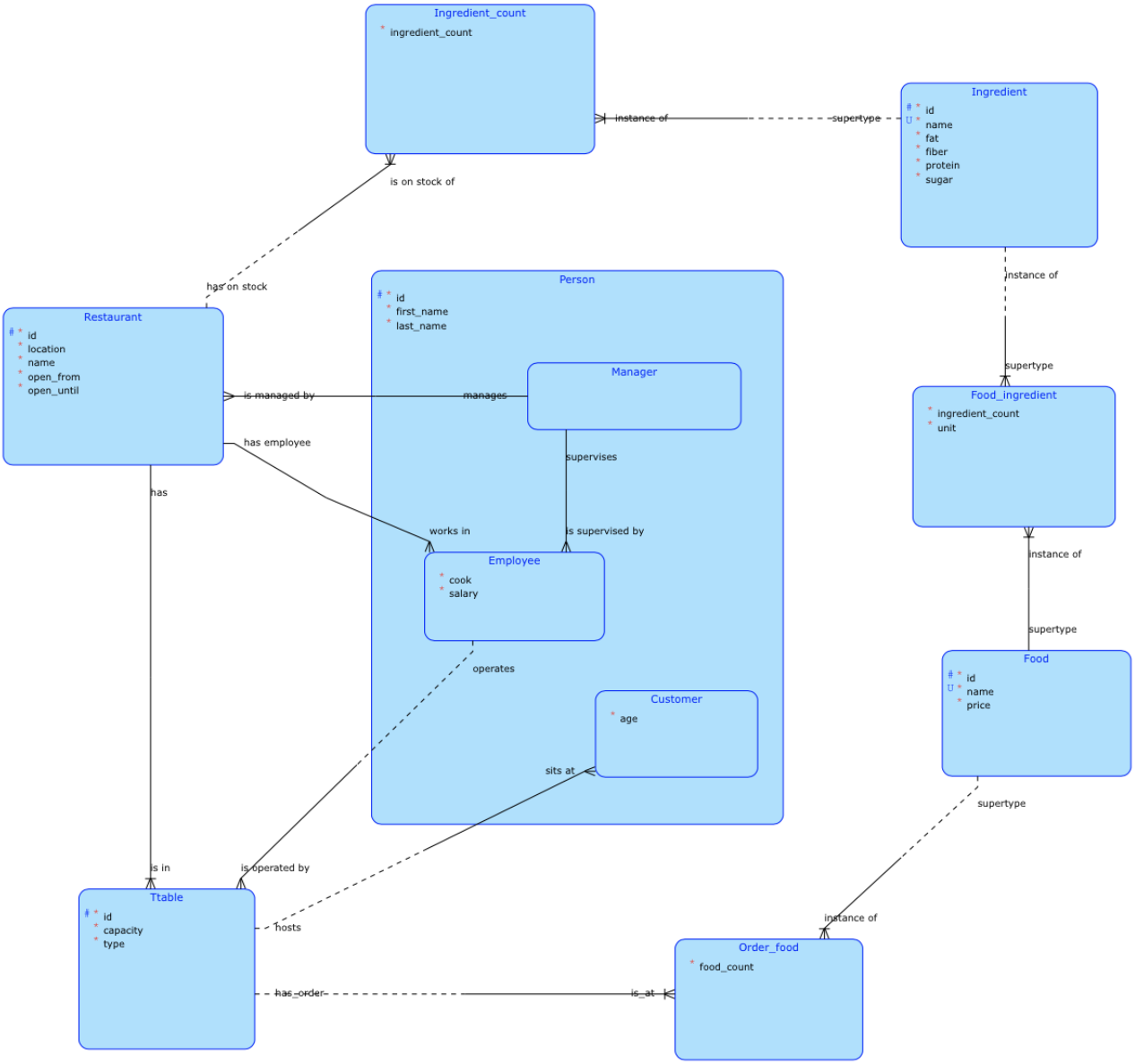
Kuchaři budou dle přísně střížených **receptů** vařit jednotlivé speciální pokrmy jako vývar s játrovým knedlíčkem, kapra s nivou nebo svítící fíkus. Není třeba evidovat, který kuchař uvěřil kolik a jakých pokrmů.

Každý **pokrm** má pevně danou **cenu** a **název**, na kterém Zdeňkovi silně záleží. Pokrm se skládá z nejméně dvou surovin, u kterých jsou silně hlídané **kalorické hodnoty** a **názvy**. Jedna surovina může být součástí více pokrmů a množství suroviny na skladu v každé restauraci je evidováno.

Do restaurací přichází **zákazníci** s uričtým **jménem**, **příjmením** a **věkem**. Zákazník nikdy není ani manažer, ani číšník. Zákazník si sedne právě k jednomu stolu, u jednoho stolů může sedět maximálně tolik lidí, jaká je jeho kapacita *(nemusí u něj však sedět nikdo)*. Zákazníci si budou prostřednictvím číšníků objednávat jednotlivé pokrmy. Není třeba evidovat který zákazník si objednal který pokrm, důležitý je pouze stůl, na který byl pokrm objednán.

Jelikož Zdeňkovi velmi záleží na čerstvosti, pokrmy jsou ze surovin připravovány až po objednávce, a není tedy třeba evidovat počet již připravených pokrmů.

Model entit



Potřebné informace

K tomu je potřeba nejprve seznámení se s Linked data, RDF datovým formátem a SPARQL jazykem.

O Linked data jsem se potřebné informace dozvěděl na přednášce předmětu BI-VWM.

Ohledně RDF datového formátu a jazyce SPARQL jsem již hledal informace dále než jen v kurzu BI-VWM, a to především v kurzu NPRG036 (KSI MFF UK), a dále na stránkách W3 (RDF) a stránkách W3 (SPARQL).

Postup řešení

Data

Při tvorbě dat a struktury dat informačního systému jsem vycházel ze semestrální práce předmětu BI-DBS, ve kterém jsem psal práci na téma "Systém restaurací Zdeňka Pohlreicha".

Nejprve jsem tedy napsal (ručně) seznam manažerů (náhodná jména a údaje), a pak jsem se inspiroval a vybral jsem pár restaurací na existujících adresách, kterým jsem vymyslel nové jméno a otevírací dobu, a přiřadil manažery.

Ručně jsem psal také seznam ingrediencí a jídel, které vycházejí z opravdových receptů či kultovních hlášek Zdeňka Pohlreicha (5 kilo česneku, kapr s nivou…)

Seznam zaměstnanců, zákazníků, stolů a objednávek jsem poté vygeneroval pseudonáhodně pomocí C++ programu, následně jsem ještě provedl drobné korekce, aby dávaly smysl.

Tady tato data `insert.sql` (napsaná v ORACLE databázi, právě kvůli výše zmíněnému předmětu BI-DBS) jsem následně pomocí IntelliJ IDE a regexů v `transform.regex` souboru přetransformoval do RDF/Turtle souboru `data.ttl`. Posledním krokem bylo převedení těchto turtle souborů do RDF/XML formátu do souboru `data.xml`, což jsem provedl pomocí nástroje https://www.easyrdf.org/converter.

Dotazy

Dotazy jsou psány v jazyce **SPARQL** jako textové soubory s příponou .sparql, dají se tedy psát i v obyčejném textovém editoru. Pro psaní dotazů jsem zvolil opět IntelliJ IDE a demoverzi pluginu RDF/SPARQL, který dokáže z nastaveného datasetu kontrolovat základní syntaxi jednotlivých dotazů.

Při psaní jednotlivých dotazů jsem se inspiroval jejich SQL variantou, kterou jsem měl k dispozici z předmětu BI-DBS, mohl jsem tak tedy srovnat správnost výsledků (a také rozdíl SQL x SPARQL, viz. poslední kapitola)

Implementace

Pro tvorbu dat bylo využito již výše zmíněné IntelliJ IDE (transformace SQL INSERT skriptu do Turtle formátu pomocí regexů) a nástroj EasyPDF (také viz. výše).

Vyhodnocování dotazů probíhá pomocí enginu Apache Jena, konkrétně pomocí příkazů `sparql`, pro zkoumání efektivity dotazů pak pomocí příkazu `qparse`, který je součástí archivu (složky bin/ a lib/, soubor log4j2.properties).

Generování zprávy k projektu probíhá pomocí nástroje AsciiDoctor PDF.

Pro vygenerování PDF je nutné nainstalovat asciidoctor-pdf `sudo gem install asciidoctor-pdf`, podtrhávání syntaxe `sudo gem install rouge` a následně provést generaci PDF pomocí `asciidoctor-pdf doc/index.adoc`.

Příklad výstupu

Obsahem této kapitoly je seznam dotazů, nejprve v přirozeném jazyce, následně v jazyce SPARQL (případně ve více formulacích). Ke každému dotazu je uveden i výpis z konzole (odpověď na dotaz).

Obsah

Dotaz 1

Zápis v přirozeném jazyce

```
Všechny ingredience potřebné k přípravě pokrmu "Zlatá česká trojkombinace"
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>
PREFIX food_ingredient: <http://wrzecond.fit.cvut.cz/food/ingredient/>
PREFIX ingredient: <http://wrzecond.fit.cvut.cz/ingredient/>

# 1. Dotaz: všechny ingredience potřebné k přípravě pokrmu "Zlatá česká trojkombinace"

SELECT ?ingredient ?ingredient_name ?count ?unit
WHERE {
    ?food food:name "Zlatá česká trojkombinace" .
    ?food_ingredient food_ingredient:makes ?food ;
                     food_ingredient:with ?ingredient ;
                     food_ingredient:count ?count ;
                     food_ingredient:unit ?unit .
    ?ingredient ingredient:name ?ingredient_name
}
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| ingredient                                | ingredient_name | count | unit |
===============================================================================
| ingredient:4                              | "česnek"        | 5     | "kg" |
| ingredient:9                              | "niva"          | 50    | "g"  |
| ingredient:17                             | "něco pod tím"  | 150   | "g"  |
-------------------------------------------------------------------------------
```

Dotaz 2

Zápis v přirozeném jazyce

```
Restaurace, jejichž manažer se nejmenuje "Vendelín"
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX manager: <http://wrzecond.fit.cvut.cz/manager/>

# 2. Restaurace, jejichž manažer se nejmenuje "Vendelín"

SELECT ?restaurant ?restaurant_id ?restaurant_name (CONCAT(?first_name, " ",
?last_name) AS ?manager_name)
WHERE {
    ?restaurant restaurant:id ?restaurant_id ;
                restaurant:name ?restaurant_name ;
                restaurant:managed_by ?manager .
    ?manager manager:first_name ?first_name ;
             manager:last_name ?last_name .
    FILTER (?first_name != "Vendelín")
}
ORDER BY ?restaurant_id
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------
| restaurant     | restaurant_id | restaurant_name    | manager_name       |
=========================================================================
| restaurant:4   | 4             | "Harmony"          | "Mikuláš Janovský" |
| restaurant:5   | 5             | "Síla chuti"       | "Mikuláš Janovský" |
| restaurant:6   | 6             | "Červené jablko"   | "Hugo Pudil"       |
| restaurant:7   | 7             | "Melodie"          | "Gustav Šťastný"   |
| restaurant:8   | 8             | "Pizzerie Verdi"   | "Roman Klusák"     |
| restaurant:9   | 9             | "Pizzerie Maestro" | "Karel Novák"      |
| restaurant:10  | 10            | "Aqua patet"       | "Bruno Opatrný"    |
| restaurant:11  | 11            | "Pizzerie Verdi"   | "Karel Novák"      |
| restaurant:12  | 12            | "Melodie"          | "Karel Novák"      |
-------------------------------------------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX manager: <http://wrzecond.fit.cvut.cz/manager/>

# 2. Restaurace, jejichž manažer se nejmenuje "Vendelín"

SELECT ?restaurant ?restaurant_id ?restaurant_name (CONCAT(?first_name, " ",
?last_name) AS ?manager_name)
WHERE {
    ?restaurant restaurant:id ?restaurant_id ;
                restaurant:name ?restaurant_name ;
                restaurant:managed_by ?manager .
    ?manager manager:first_name ?first_name ;
             manager:last_name ?last_name
    MINUS {
        ?restaurant restaurant:id ?restaurant_id ;
                    restaurant:name ?restaurant_name ;
                    restaurant:managed_by ?manager .
        ?manager manager:first_name ?first_name ;
                 manager:last_name ?last_name .
        FILTER (?first_name = "Vendelín")
    }
}
ORDER BY ?restaurant_id
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------
| restaurant     | restaurant_id | restaurant_name    | manager_name       |
=========================================================================
| restaurant:4   | 4             | "Harmony"          | "Mikuláš Janovský" |
| restaurant:5   | 5             | "Síla chuti"       | "Mikuláš Janovský" |
| restaurant:6   | 6             | "Červené jablko"   | "Hugo Pudil"       |
| restaurant:7   | 7             | "Melodie"          | "Gustav Šťastný"   |
| restaurant:8   | 8             | "Pizzerie Verdi"   | "Roman Klusák"     |
| restaurant:9   | 9             | "Pizzerie Maestro" | "Karel Novák"      |
| restaurant:10  | 10            | "Aqua patet"       | "Bruno Opatrný"    |
| restaurant:11  | 11            | "Pizzerie Verdi"   | "Karel Novák"      |
| restaurant:12  | 12            | "Melodie"          | "Karel Novák"      |
-------------------------------------------------------------------------
```

Dotaz 3

Zápis v přirozeném jazyce

```
Zákazníci (id, jméno, příjmení), kteří si objednali pouze "Filipínské kebaby".
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>

# 3. Zákazníci (id, jméno, příjmení), kteří si objednali pouze "Filipínské kebaby".

SELECT ?customer (CONCAT(?first_name, " ", ?last_name) AS ?customer_name) ?table WHERE
{
    ?customer customer:first_name ?first_name ;
              customer:last_name ?last_name ;
              customer:sits_at ?table .
    ?order order:at ?table ;
           order:contains ?food .
    ?food food:name ?food_name .
    FILTER (?food_name = "Filipínské kebaby")
    MINUS {
        ?orderx order:at ?table ; # Odečítáme pouze na základě stolu
                order:contains ?foodx .
        ?foodx food:name ?food_namex .
        FILTER (?food_namex != "Filipínské kebaby")
    }
}
ORDER BY ?customer
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| customer     | customer_name      | table                                  |
===============================================================================
| customer:15  | "Zikmund Vyskočil" | <http://wrzecond.fit.cvut.cz/table/29> |
| customer:16  | "Jan Konvalinka"   | <http://wrzecond.fit.cvut.cz/table/12> |
| customer:21  | "Josef Tříska"     | <http://wrzecond.fit.cvut.cz/table/12> |
| customer:35  | "Erik Vomáčka"     | <http://wrzecond.fit.cvut.cz/table/29> |
| customer:50  | "Zdeněk Tříska"    | <http://wrzecond.fit.cvut.cz/table/12> |
| customer:76  | "Josef Ševčík"     | <http://wrzecond.fit.cvut.cz/table/29> |
-------------------------------------------------------------------------------
```

Dotaz 4

Zápis v přirozeném jazyce

```
Stoly, na kterých byly podány objednávky obsahující všechna jídla v restauraci "Síla
chuti".
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX table_type: <http://wrzecond.fit.cvut.cz/table/type/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>

# 4. Stoly, na kterých byly podány objednávky
# obsahující všechna jídla v restauraci "Síla chuti".

SELECT ?table ?table_type (COUNT(?food) AS ?food_count) WHERE {
    ?restaurant restaurant:name ?restaurant_name .
    ?table table:in ?restaurant ;
           table:type ?type .
    ?type table_type:name ?table_type .
    ?order order:at ?table ;
           order:contains ?food
    FILTER (?restaurant_name = "Síla chuti")
    {
        SELECT (COUNT(?food) AS ?food_total) WHERE {
            ?food food:id ?food_id
        }
    }
}
GROUP BY ?table ?table_type ?food_total HAVING (?food_count = ?food_total)
ORDER BY ?table
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------
| table                                | table_type   | food_count |
===================================================================
| table:19                             | "Restaurace" | 10         |
| table:24                             | "Zahrádka"   | 10         |
-------------------------------------------------------------------
```

Dotaz 5

Zápis v přirozeném jazyce

```
Ověření předchozího dotazu: seznam jídel bez všech jídel objednaných u libovolného
stolu z dotazu 4 musí být prázdná množina.
```

Zápis v jazyce SPARQL

```
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX table_type: <http://wrzecond.fit.cvut.cz/table/type/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>

ASK WHERE { FILTER NOT EXISTS { # Reverse yes/no
    ?food food:id ?food_id
    MINUS {
        SELECT ?food_id WHERE {
            ?order order:at ?table ;
                   order:contains ?food .
            ?food food:id ?food_id .
            {
                SELECT ?table ?table_type (COUNT(?food) AS ?food_count) WHERE {
                    ?restaurant restaurant:name ?restaurant_name .
                    ?table table:in ?restaurant ;
                           table:type ?type .
                    ?type table_type:name ?table_type .
                    ?order order:at ?table ;
                           order:contains ?food
                    FILTER (?restaurant_name = "Síla chuti") {
                        SELECT (COUNT(?food) AS ?food_total) WHERE {
                            ?food food:id ?food_id
                        }
                    }
                }
                GROUP BY ?table ?table_type ?food_total HAVING (?food_count =
?food_total) LIMIT 1
            }
        }
    }
} }
```

Výstup z konzole

```
Ask => Yes
```

Dotaz 6

Zápis v přirozeném jazyce

```
Vytvoř graf "Zaměstnanec pracuje na pozici x a stará se o y objednávek"
```

Zápis v jazyce SPARQL

Pro přehlednost dokumentace se dotazuji pouze na prvních 15 výsledků.
(LIMIT by v reálné query nebyl)

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>

# 6. Vytvoř graf "Zaměstnanec pracuje na pozici x a stará se o y objednávek

CONSTRUCT {
    ?employee employee:position ?position ;
            employee:order_count ?order_count .
}
WHERE {
    SELECT ?employee ?position (COUNT(?order) AS ?order_count) WHERE {
        ?employee employee:works_in ?restaurant ;
                  employee:salary ?salary ;
                  employee:cook ?cook .
        OPTIONAL {
            ?table table:operated_by ?employee .
            ?order order:at ?table .
        }
        BIND(IF(?cook, "Kuchař a Číšník", "Číšník") AS ?position)
    }
    GROUP BY ?employee ?position
    LIMIT 15 # For documentation purposes
}
```

Výstup z konzole

```
@prefix employee: <http://wrzecond.fit.cvut.cz/employee/> .

employee:43  employee:order_count  0 ;
        employee:position     "Kuchař a Číšník" .

employee:7  employee:order_count  0 ;
        employee:position     "Číšník" .

employee:41  employee:order_count  0 ;
        employee:position     "Číšník" .

employee:5  employee:order_count  1 ;
        employee:position     "Kuchař a Číšník" .

employee:29  employee:order_count  3 ;
        employee:position     "Číšník" .

employee:13  employee:order_count  2 ;
        employee:position     "Číšník" .

employee:49  employee:order_count  0 ;
        employee:position     "Číšník" .

employee:33  employee:order_count  1 ;
        employee:position     "Číšník" .

employee:27  employee:order_count  0 ;
        employee:position     "Kuchař a Číšník" .

employee:25  employee:order_count  2 ;
        employee:position     "Číšník" .

employee:19  employee:order_count  3 ;
        employee:position     "Kuchař a Číšník" .

employee:51  employee:order_count  0 ;
        employee:position     "Kuchař a Číšník" .

employee:4  employee:order_count  0 ;
        employee:position     "Kuchař a Číšník" .

employee:9  employee:order_count  0 ;
        employee:position     "Kuchař a Číšník" .

employee:17  employee:order_count  0 ;
        employee:position     "Číšník" .
```

Dotaz 7

Zápis v přirozeném jazyce

```
Seznam neobsazených stolů ( = stolů, u kterých nikdo nesedí) v restauraci "Harmony"
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>

# 7. Seznam neobsazených stolů ( = stolů, u kterých nikdo nesedí) v restauraci
"Harmony"

SELECT ?table
WHERE {
    ?table table:in/restaurant:name "Harmony" .
    FILTER NOT EXISTS {
        ?table table:in/restaurant:name "Harmony" .
        ?customer customer:sits_at ?table .
    }
}
ORDER BY ?table
LIMIT 10
```

Výstup z konzole

```
-----------------------------------------
| table                                 |
=========================================
| table:15                              |
| table:17                              |
-----------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>

# 7. Seznam neobsazených stolů ( = u kterých nikdo nesedí) v restauraci "Harmony"

SELECT ?table
WHERE {
    ?table table:in/restaurant:name "Harmony" .
    OPTIONAL { ?customer customer:sits_at ?table }
    FILTER ( !BOUND(?customer) )
}
ORDER BY ?table
LIMIT 10
```

Výstup z konzole

```
-----------------------------------------
| table                                 |
=========================================
| table:15                              |
| table:17                              |
-----------------------------------------
```

Dotaz 8

Zápis v přirozeném jazyce

```
Všichni zaměstnanci, kteří neobsluhují žádný stůl v restauraci "Ostravská."
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>

# 8. Všichni zaměstnanci, kteří neobsluhují žádný stůl v restauraci "Ostravská."

SELECT ?employee ?name WHERE {
    ?employee employee:works_in/restaurant:name ?restaurant_name ;
              employee:first_name ?first_name ;
              employee:last_name ?last_name .
    MINUS {
        ?employee employee:works_in/restaurant:name ?restaurant_name .
        ?table table:operated_by ?employee
    }
    BIND (CONCAT(?first_name, " ", ?last_name) AS ?name)
    FILTER (?restaurant_name = "Ostravská")
}
ORDER BY ?employee
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------
| employee                                | name            |
=============================================================
| employee:3                              | "Jan Novotný"   |
| employee:4                              | "Karel Janoušek" |
| employee:7                              | "Filip Tříska"  |
-------------------------------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>

# 8. Všichni zaměstnanci, kteří neobsluhují žádný stůl v restauraci "Ostravská."

SELECT ?employee (CONCAT(?first_name, " ", ?last_name) AS ?name) WHERE {
    ?employee employee:works_in/restaurant:name "Ostravská" ;
              employee:first_name ?first_name ;
              employee:last_name ?last_name .
    OPTIONAL { ?table table:operated_by ?employee }
}
GROUP BY ?employee ?first_name ?last_name HAVING (COUNT(?table) = 0)
ORDER BY ?employee
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------
| employee                                | name              |
=============================================================
| employee:3                              | "Jan Novotný"     |
| employee:4                              | "Karel Janoušek"  |
| employee:7                              | "Filip Tříska"    |
-------------------------------------------------------------
```

Dotaz 9

Zápis v přirozeném jazyce

```
Počet surovin, které jsou potřeba k přípravě každého jídla, seřazeno sestupně.
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>
PREFIX ingredient_food: <http://wrzecond.fit.cvut.cz/food/ingredient/>

# 9. Počet surovin, které jsou potřeba k přípravě každého jídla, seřazeno sestupně.
SELECT ?food ?food_name ?ingredient_count WHERE {
    ?food food:name ?food_name .
    {
        SELECT ?food (COUNT(*) AS ?ingredient_count) WHERE {
            ?food_ingredient ingredient_food:makes ?food
        }
        GROUP BY ?food
    }
}
ORDER BY DESC(?ingredient_count)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------
| food       | food_name                                | ingredient_count |
=========================================================================
| food:6     | "Burger podle Pata a Mata"               | 5                |
| food:1     | "Vývar s nudlemi a játrovým knedlíčkem" | 4                |
| food:10    | "Steak s wasabi špenátem"                | 4                |
| food:8     | "Zapékané rozmarýnové brambory"          | 4                |
| food:3     | "Hráškový krém s opečeným chlebíčkem"    | 3                |
| food:4     | "Kapr s nivou"                           | 3                |
| food:7     | "Zlatá česká trojkombinace"              | 3                |
| food:9     | "Filipínské kebaby"                      | 3                |
| food:2     | "Hovězí vývar s celestýnskými nudlemi"   | 2                |
| food:5     | "Rumcajsovy koule"                       | 2                |
-------------------------------------------------------------------------
```

Dotaz 10

Zápis v přirozeném jazyce

```
Kontrola, že každá restaurace má právě jednoho manažera
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>

# 10. Kontrola, že každá restaurace má právě jednoho manažera
ASK WHERE { FILTER NOT EXISTS { SELECT * WHERE { # Reverse yes/no
    ?restaurant restaurant:name ?name .
    OPTIONAL { ?restaurant restaurant:managed_by ?manager }
}
GROUP BY ?restaurant ?name HAVING (COUNT(?manager) != 1) } }
```

Výstup z konzole

```
Ask => Yes
```

Dotaz 11

Zápis v přirozeném jazyce

```
Vytvoř graf popisující kolik jídel je objednáno na kterém stole
```

Zápis v jazyce SPARQL

```
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>

CONSTRUCT { ?table table:has_food_count ?sum_food }
WHERE {
    {
        SELECT ?table (SUM(?food_count) AS ?sum_food) WHERE {
            ?order order:at ?table ;
                order:count ?food_count
        } GROUP BY ?table
    }
}
```

Výstup z konzole

```
@prefix table: <http://wrzecond.fit.cvut.cz/table/> .

table:11  table:has_food_count  7 .
table:2  table:has_food_count  1 .
table:20  table:has_food_count  2 .
table:26  table:has_food_count  5 .
table:8  table:has_food_count  4 .
table:10  table:has_food_count  5 .
table:16  table:has_food_count  5 .
table:25  table:has_food_count  4 .
table:24  table:has_food_count  23 .
table:6  table:has_food_count  4 .
table:14  table:has_food_count  4 .
table:29  table:has_food_count  1 .
table:5  table:has_food_count  3 .
table:13  table:has_food_count  2 .
table:19  table:has_food_count  21 .
table:4  table:has_food_count  1 .
table:28  table:has_food_count  1 .
table:12  table:has_food_count  3 .
table:18  table:has_food_count  7 .
table:3  table:has_food_count  6 .
table:27  table:has_food_count  1 .
```

Dotaz 12

Zápis v přirozeném jazyce

```
Vytvoř graf manažer DOHLÍŽÍ NA zaměstnance (v restauracích, které spravuje)
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX manager: <http://wrzecond.fit.cvut.cz/manager/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>

# 12. Vytvoř graf manažer DOHLÍŽÍ NA zaměstnance (v restauracích, které spravuje)
CONSTRUCT {
    ?manager manager:supervises ?employee ;
             manager:in ?restaurant .
}
WHERE {
    ?restaurant restaurant:managed_by ?manager .
    ?employee employee:works_in ?restaurant .
}
```

Výstup z konzole

```
@prefix manager: <http://wrzecond.fit.cvut.cz/manager/> .
@prefix restaurant: <http://wrzecond.fit.cvut.cz/restaurant/> .
@prefix employee: <http://wrzecond.fit.cvut.cz/employee/> .

manager:2  manager:in       restaurant:5 , restaurant:4 ;
        manager:supervises  employee:30 , employee:26 , employee:25 , employee:28 ,
employee:29 , employee:27 .

manager:7  manager:in       restaurant:9 ;
        manager:supervises  employee:52 , employee:56 , employee:55 , employee:54 ,
employee:53 , employee:57 .

manager:5  manager:in       restaurant:10 ;
        manager:supervises  employee:61 , employee:59 , employee:60 , employee:58 ,
employee:62 .

manager:3  manager:in       restaurant:8 ;
        manager:supervises  employee:50 , employee:46 , employee:47 , employee:49 ,
employee:45 , employee:51 , employee:48 , employee:44 .

manager:8  manager:in       restaurant:12 , restaurant:11 ;
        manager:supervises  employee:64 , employee:63 .

manager:1  manager:in       restaurant:1 , restaurant:2 , restaurant:3 ;
        manager:supervises  employee:4 , employee:3 , employee:14 , employee:2 ,
employee:1 , employee:24 , employee:9 , employee:8 , employee:13 , employee:23 ,
employee:7 , employee:12 , employee:22 , employee:6 , employee:21 , employee:11 ,
employee:5 , employee:20 , employee:19 , employee:10 , employee:18 , employee:17 ,
employee:16 , employee:15 .

manager:6  manager:in       restaurant:6 ;
        manager:supervises  employee:31 , employee:32 , employee:39 , employee:34 ,
employee:33 , employee:35 , employee:36 , employee:38 , employee:37 .

manager:4  manager:in       restaurant:7 ;
        manager:supervises  employee:43 , employee:40 , employee:42 , employee:41 .
```

Dotaz 13

Zápis v přirozeném jazyce

```
Popiš stoly z dotazu 4 (Stoly, na kterých byly podány objednávky obsahující všechna
jídla v restauraci "Síla chuti".)
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX table_type: <http://wrzecond.fit.cvut.cz/table/type/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>

# 13. Popiš stoly z dotazu 4
DESCRIBE ?table
WHERE {
    {
        SELECT ?table ?table_type (COUNT(?food) AS ?food_count) WHERE {
            ?restaurant restaurant:name ?restaurant_name .
            ?table table:in ?restaurant ;
                   table:type ?type .
            ?type table_type:name ?table_type .
            ?order order:at ?table ;
                   order:contains ?food
            FILTER (?restaurant_name = "Síla chuti")
            {
                SELECT (COUNT(?food) AS ?food_total) WHERE {
                    ?food food:id ?food_id
                }
            }
        }
        GROUP BY ?table ?table_type ?food_total HAVING (?food_count = ?food_total)
        ORDER BY ?table
        LIMIT 10
    }
}
```

Výstup z konzole

```
@prefix restaurant: <http://wrzecond.fit.cvut.cz/restaurant/> .
@prefix employee: <http://wrzecond.fit.cvut.cz/employee/> .
@prefix table: <http://wrzecond.fit.cvut.cz/table/> .
@prefix table_type: <http://wrzecond.fit.cvut.cz/table/type/> .

table:19  table:capacity   5 ;
        table:id           19 ;
        table:in           restaurant:5 ;
        table:operated_by  employee:30 ;
        table:type         table_type:1 .

table:24  table:capacity   11 ;
        table:id           24 ;
        table:in           restaurant:5 ;
        table:operated_by  employee:30 ;
        table:type         table_type:2 .
```

Dotaz 14

Zápis v přirozeném jazyce

```
Zaměstnanci (id, jméno, příjmení), kteří pracují v restauraci "Červené jablko" a
zároveň obsluhují stůl s kapacitou 3 lidí.
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 14. Zaměstnanci (id, jméno, příjmení), kteří pracují v restauraci "Červené jablko"
# a zároveň obsluhují stůl s kapacitou 3 lidí.

SELECT ?employee (CONCAT(?first_name, " ", ?last_name) AS ?name) WHERE {
    ?employee employee:works_in/restaurant:name "Červené jablko" ;
              employee:first_name ?first_name ;
              employee:last_name ?last_name .
    ?table table:operated_by ?employee ;
           table:capacity "3"^^xsd:integer
}
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------
| employee                                | name         |
=============================================================
| employee:34                             | "Erik Novák" |
-------------------------------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 14. Zaměstnanci (id, jméno, příjmení), kteří pracují v restauraci "Červené jablko"
# a zároveň obsluhují stůl s kapacitou 3 lidí.

SELECT ?employee ?name WHERE {
    ?employee employee:works_in/restaurant:name ?restaurant_name ;
              employee:first_name ?first_name ;
              employee:last_name ?last_name .
    FILTER EXISTS {
        SELECT ?employee WHERE {
            ?table table:capacity "3"^^xsd:integer ;
                   table:operated_by ?employee
        }
    }
    BIND(CONCAT(?first_name, " ", ?last_name) AS ?name)
    FILTER (?restaurant_name = "Červené jablko")
}
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------
| employee                                  | name          |
=============================================================
| employee:34                               | "Erik Novák" |
-------------------------------------------------------------
```

Dotaz 15

Zápis v přirozeném jazyce

```
Suroviny potřebné k přípravě dvouchodového obědu složeného z "Hovězí vývar s
celestýnskými nudlemi" a "Rumcajsovy koule".
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>
PREFIX food_ingredient: <http://wrzecond.fit.cvut.cz/food/ingredient/>
PREFIX ingredient: <http://wrzecond.fit.cvut.cz/ingredient/>

# 15. Suroviny potřebné k přípravě dvouchodového obědu, složeného
# z "Hovězí vývar s celestýnskými nudlemi" a "Rumcajsovy koule".

SELECT DISTINCT ?ingredient ?ingredient_name (SUM(?ingredient_count) AS ?total_count)
?unit
WHERE {
    {
        ?food food:name "Hovězí vývar s celestýnskými nudlemi" .
        ?food_ingredient food_ingredient:makes ?food ;
                         food_ingredient:unit ?unit ;
                         food_ingredient:count ?ingredient_count ;
                         food_ingredient:with ?ingredient .
        ?ingredient ingredient:name ?ingredient_name
    }
    UNION
    {
        ?food food:name "Rumcajsovy koule" .
        ?food_ingredient food_ingredient:makes ?food ;
                         food_ingredient:unit ?unit ;
                         food_ingredient:count ?ingredient_count ;
                         food_ingredient:with ?ingredient .
        ?ingredient ingredient:name ?ingredient_name
    }
}
GROUP BY ?ingredient ?ingredient_name ?unit
ORDER BY DESC(?total_count)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------
| ingredient         | ingredient_name     | total_count | unit |
===================================================================
| ingredient:1       | "voda"              | 200         | "ml" |
| ingredient:10      | "rajčatová omáčka"  | 150         | "ml" |
| ingredient:11      | "masové koule"      | 80          | "g"  |
| ingredient:5       | "celestýnské nudle" | 80          | "g"  |
-------------------------------------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>
PREFIX food_ingredient: <http://wrzecond.fit.cvut.cz/food/ingredient/>
PREFIX ingredient: <http://wrzecond.fit.cvut.cz/ingredient/>

SELECT DISTINCT ?ingredient ?ingredient_name (SUM(?ingredient_count) AS ?total_count)
?unit
WHERE {
    ?food food:name ?food_name .
    ?food_ingredient food_ingredient:makes ?food ;
                     food_ingredient:unit ?unit ;
                     food_ingredient:count ?ingredient_count ;
                     food_ingredient:with ?ingredient .
    ?ingredient ingredient:name ?ingredient_name
    FILTER (?food_name = "Hovězí vývar s celestýnskými nudlemi" || ?food_name =
"Rumcajsovy koule")
}
GROUP BY ?ingredient ?ingredient_name ?unit
ORDER BY DESC(?total_count)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------
| ingredient         | ingredient_name     | total_count | unit |
===================================================================
| ingredient:1       | "voda"              | 200         | "ml" |
| ingredient:10      | "rajčatová omáčka"  | 150         | "ml" |
| ingredient:11      | "masové koule"      | 80          | "g"  |
| ingredient:5       | "celestýnské nudle" | 80          | "g"  |
-------------------------------------------------------------------
```

Dotaz 16

Zápis v přirozeném jazyce

```
# 16. Vytvoř graf zákazník :má_jméno jméno, :sedí_u číslo stolu, :sedí_v jméno
restaurace, :objednáno_jídel obsahující počet druhů objednaných jídel u stolu, kde
právě sedí.
```

Zápis v jazyce SPARQL

Pro přehlednost dokumentace se dotazuji pouze na prvních 10 výsledků.
(LIMIT by v reálné query nebyl)

```
# Prefix
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>

# 16. Vytvoř graf zákazník má jméno, sedí u čísla stolu (...)

CONSTRUCT {
    ?customer customer:name ?name ;
              customer:sits_at_id ?table_id ;
              customer:sits_in_name ?restaurant_name ;
              customer:unique_food_count ?food_count
}
WHERE {
    SELECT ?customer ?name ?table_id ?restaurant_name (COUNT(?order) AS ?food_count)
WHERE {
        ?customer customer:first_name ?first_name ;
                  customer:last_name ?last_name ;
                  customer:sits_at ?table ;
                  customer:sits_in/restaurant:name ?restaurant_name .
        ?table table:id ?table_id .
        OPTIONAL { ?order order:at ?table }
        BIND (CONCAT(?first_name, " ", ?last_name) AS ?name)
    }
    GROUP BY ?customer ?name ?table_id ?restaurant_name
    LIMIT 10 # Pro účely dokumentace
}
```

Výstup z konzole

```
@prefix customer: <http://wrzecond.fit.cvut.cz/customer/> .
```

```
customer:43  customer:name          "Josef Spěvák" ;
        customer:sits_at_id         52 ;
        customer:sits_in_name       "Aqua patet" ;
        customer:unique_food_count  0 .

customer:21  customer:name          "Josef Tříska" ;
        customer:sits_at_id         12 ;
        customer:sits_in_name       "Divinis" ;
        customer:unique_food_count  1 .

customer:10  customer:name          "Karel Konvalinka" ;
        customer:sits_at_id         26 ;
        customer:sits_in_name       "Síla chuti" ;
        customer:unique_food_count  2 .

customer:79  customer:name          "Josef Skřivánek" ;
        customer:sits_at_id         5 ;
        customer:sits_in_name       "Ostravská" ;
        customer:unique_food_count  1 .

customer:29  customer:name          "Hugo Janoušek" ;
        customer:sits_at_id         26 ;
        customer:sits_in_name       "Síla chuti" ;
        customer:unique_food_count  2 .

customer:50  customer:name          "Zdeněk Tříska" ;
        customer:sits_at_id         12 ;
        customer:sits_in_name       "Divinis" ;
        customer:unique_food_count  1 .

customer:22  customer:name          "Zdeněk Novotný" ;
        customer:sits_at_id         6 ;
        customer:sits_in_name       "Café Imperial" ;
        customer:unique_food_count  2 .

customer:27  customer:name          "Vojtěch Ševčík" ;
        customer:sits_at_id         46 ;
        customer:sits_in_name       "Pizzerie Maestro" ;
        customer:unique_food_count  0 .

customer:36  customer:name          "Filip Vyskočil" ;
        customer:sits_at_id         28 ;
        customer:sits_in_name       "Červené jablko" ;
        customer:unique_food_count  1 .

customer:14  customer:name          "Hugo Novák" ;
        customer:sits_at_id         10 ;
        customer:sits_in_name       "Divinis" ;
        customer:unique_food_count  2 .
```

Dotaz 17

Zápis v přirozeném jazyce

```
Najdi zákazníky z restaurace "Divinis", kteří si objednali alespoň 2 různá jídla.
```

Zápis v jazyce SPARQL

```
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>

SELECT ?customer ?name ?food_count WHERE {
    {
        SELECT ?customer ?name ?table_id ?restaurant_name (COUNT(?order) AS
?food_count) WHERE {
            ?customer customer:first_name ?first_name ;
                      customer:last_name ?last_name ;
                      customer:sits_at ?table ;
                      customer:sits_in/restaurant:name ?restaurant_name .
            ?table table:id ?table_id .
            ?order order:at ?table .
            BIND (CONCAT(?first_name, " ", ?last_name) AS ?name)
        }
        GROUP BY ?customer ?name ?table_id ?restaurant_name
    }
    FILTER (?food_count >= 2 && ?restaurant_name = "Divinis")
}
ORDER BY DESC(?food_count)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| customer                                | name              | food_count |
===============================================================================
| customer:25                             | "Vojtěch Novotný" | 3          |
| customer:1                              | "Zdeněk Vomáčka"  | 2          |
| customer:11                             | "Zdeněk Novák"    | 2          |
| customer:14                             | "Hugo Novák"      | 2          |
| customer:30                             | "Filip Vyskočil"  | 2          |
| customer:37                             | "Zdeněk Ševčík"   | 2          |
| customer:38                             | "Erik Janoušek"   | 2          |
| customer:41                             | "Patrik Janoušek" | 2          |
-------------------------------------------------------------------------------
```

Dotaz 18

Zápis v přirozeném jazyce

```
Seznam restaurací, ve kterých sedí alespoň 5 osob starších 60 let
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 18. Seznam restaurací, ve kterých sedí alespoň 5 osob starších 60 let
SELECT ?restaurant ?restaurant_name (COUNT(?customer_age) AS ?old_customers)
WHERE {
    ?table table:in ?restaurant .
    ?restaurant restaurant:name ?restaurant_name .
    ?customer customer:age ?customer_age ;
              customer:sits_at ?table .
    FILTER (?customer_age > "60"^^xsd:integer)
}

GROUP BY ?restaurant ?restaurant_name HAVING (?old_customers >= 5)
ORDER BY DESC(?old_customers)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| restaurant                          | restaurant_name  | old_customers |
===============================================================================
| restaurant:5                        | "Síla chuti"     | 9             |
| restaurant:1                        | "Ostravská"      | 5             |
| restaurant:3                        | "Divinis"        | 5             |
| restaurant:6                        | "Červené jablko" | 5             |
-------------------------------------------------------------------------------
```

Alternativní zápis v jazyce SPARQL

```
# Prefix
PREFIX customer: <http://wrzecond.fit.cvut.cz/customer/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 18. Seznam restaurací, ve kterých sedí alespoň 5 osob starších 60 let
SELECT ?restaurant ?restaurant_name (COUNT(?customer_age) AS ?old_customers)
WHERE {
    ?restaurant restaurant:name ?restaurant_name ;
                ^table:in/^customer:sits_at/customer:age ?customer_age .
    FILTER (?customer_age > "60"^^xsd:integer)
}
GROUP BY ?restaurant ?restaurant_name HAVING (?old_customers >= 5)
ORDER BY DESC(?old_customers)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| restaurant                              | restaurant_name  | old_customers |
===============================================================================
| restaurant:5                            | "Síla chuti"     | 9             |
| restaurant:1                            | "Ostravská"      | 5             |
| restaurant:3                            | "Divinis"        | 5             |
| restaurant:6                            | "Červené jablko" | 5             |
-------------------------------------------------------------------------------
```

Dotaz 19

Zápis v přirozeném jazyce

```
Stav suroviny "voda" na skladech ve všech restauracích.
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX ingredient: <http://wrzecond.fit.cvut.cz/ingredient/>
PREFIX storage_item: <http://wrzecond.fit.cvut.cz/storage/item/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>

# 19. Stav suroviny "voda" na skladech ve všech restauracích.

SELECT ?restaurant ?restaurant_name ?ingredient_name ?ingredient_count WHERE {
    ?restaurant restaurant:id ?restaurant_id ;
                restaurant:name ?restaurant_name .
    ?storage_item storage_item:in ?restaurant ;
                  storage_item:of/ingredient:name ?ingredient_name ;
                  storage_item:count ?ingredient_count .
    FILTER (?ingredient_name = "voda")
}
ORDER BY ?restaurant_id
LIMIT 30
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| restaurant     | restaurant_name    | ingredient_name | ingredient_count |
===============================================================================
| restaurant:1   | "Ostravská"        | "voda"          | 451              |
| restaurant:2   | "Café Imperial"    | "voda"          | 121              |
| restaurant:3   | "Divinis"          | "voda"          | 460              |
| restaurant:4   | "Harmony"          | "voda"          | 370              |
| restaurant:5   | "Síla chuti"       | "voda"          | 469              |
| restaurant:6   | "Červené jablko"   | "voda"          | 312              |
| restaurant:7   | "Melodie"          | "voda"          | 99               |
| restaurant:8   | "Pizzerie Verdi"   | "voda"          | 399              |
| restaurant:9   | "Pizzerie Maestro" | "voda"          | 255              |
| restaurant:10  | "Aqua patet"       | "voda"          | 365              |
| restaurant:11  | "Pizzerie Verdi"   | "voda"          | 35               |
| restaurant:12  | "Melodie"          | "voda"          | 764              |
-------------------------------------------------------------------------------
```

Dotaz 20

Zápis v přirozeném jazyce

```
Kontrola, že žádný zaměstnanec restaurace "Divinis" nemá vyšší plat než 29.000 Kč
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 20. Kontrola, že žádný zaměstnanec restaurace "Divinis" nemá vyšší plat než 29.000
Kč

ASK WHERE { FILTER NOT EXISTS { # Reverse yes/no
    ?employee employee:salary ?salary ;
              employee:works_in/restaurant:name "Divinis" .
    FILTER (?salary > "29000"^^xsd:integer)
} }
```

Výstup z konzole

```
Ask => Yes
```

Dotaz 21

Zápis v přirozeném jazyce

```
Manažeři restaurací, které mají otevřeno ve 23:15
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX manager: <http://wrzecond.fit.cvut.cz/manager/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 21. Manažeři restaurací, které mají otevřeno ve 23:15
SELECT ?restaurant (CONCAT(?first_name, " ", ?last_name) AS ?name) ?open_from
?open_until
WHERE {
    ?restaurant restaurant:open_from ?open_from ;
                restaurant:open_until ?open_until ;
                restaurant:managed_by ?manager .
    ?manager    manager:first_name ?first_name ;
                manager:last_name ?last_name .
    FILTER (?open_from <= "23:15:00"^^xsd:time && ?open_until >= "23:15:00"^^xsd:time)
}
ORDER BY ?restaurant
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| restaurant   | name               | open_from           | open_until          |
===============================================================================
| restaurant:2 | "Vendelín Bosák"   | "00:00:00"^^xsd:time | "23:59:00"^^xsd:time |
| restaurant:4 | "Mikuláš Janovský" | "11:00:00"^^xsd:time | "23:30:00"^^xsd:time |
| restaurant:6 | "Hugo Pudil"       | "09:00:00"^^xsd:time | "23:59:00"^^xsd:time |
| restaurant:8 | "Roman Klusák"     | "18:00:00"^^xsd:time | "23:59:00"^^xsd:time |
-------------------------------------------------------------------------------
```

Dotaz 22

Zápis v přirozeném jazyce

```
První tři restaurace s největší kapacitou (kapacita je určena počtem součtem kapacity
stolů v restauraci)
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 22. První tři restaurace s největší kapacitou
# (kapacita je určena počtem součtem kapacity stolů v restauraci)

SELECT ?restaurant ?restaurant_name (SUM(?capacity) AS ?restaurant_capacity) WHERE {
    ?restaurant restaurant:name ?restaurant_name .
    ?table table:in ?restaurant ;
           table:capacity ?capacity
}
GROUP BY ?restaurant ?restaurant_name
ORDER BY DESC(?restaurant_capacity)
LIMIT 3
```

Výstup z konzole

```
-------------------------------------------------------------------
| restaurant           | restaurant_name  | restaurant_capacity |
===================================================================
| restaurant:5         | "Síla chuti"     | 56                  |
| restaurant:8         | "Pizzerie Verdi" | 42                  |
| restaurant:12        | "Melodie"        | 37                  |
-------------------------------------------------------------------
```

Dotaz 23

Zápis v přirozeném jazyce

```
Počty zaměstananců restaurací, které jsou v Praze.
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>

# 23. Počet zaměstananců restaurací, které jsou v Praze.
SELECT ?restaurant ?restaurant_name ?restaurant_location (COUNT(?employee) AS
?employee_count)
WHERE {
    ?restaurant restaurant:name ?restaurant_name ;
                restaurant:location ?restaurant_location .
    ?employee employee:works_in ?restaurant .
    FILTER (CONTAINS(?restaurant_location, "Praha"))
}
GROUP BY ?restaurant ?restaurant_name ?restaurant_location
ORDER BY DESC(?employee_count)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------------------------
| restaurant     | restaurant_name | restaurant_location         | employee_count |
===============================================================================
| restaurant:2   | "Café Imperial" | "Na Poříčí 1072/15,  Praha" | 8              |
| restaurant:3   | "Divinis"       | "Týnská 21,  Praha"         | 8              |
| restaurant:10  | "Aqua patet"    | "Bechyňova 2744/8,  Praha"  | 5              |
| restaurant:4   | "Harmony"       | "Plukovníka  (...), Praha"  | 5              |
-------------------------------------------------------------------------------
```

Dotaz 24

Zápis v přirozeném jazyce

```
Zaměstnanci, kteří jsou kuchaři, ale obsluhují zároveň alespoň jeden stůl.
```

Zápis v jazyce SPARQL

```
PREFIX employee: <http://wrzecond.fit.cvut.cz/employee/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?employee ?name WHERE {
    ?employee employee:first_name ?first_name ;
              employee:last_name ?last_name ;
              employee:cook ?cook .
    ?table table:operated_by ?employee .
    BIND(CONCAT(?first_name, " ", ?last_name) AS ?name)
    FILTER (?cook = "1"^^xsd:boolean)
}
GROUP BY ?employee ?name LIMIT 30 # alespoň 1 stůl = stačí, že vazba existuje
```

Výstup z konzole

```
-------------------------------------------------------------------
| employee                                  | name               |
===================================================================
| employee:44                               | "Vojtěch Janoušek" |
| employee:19                               | "Zdeněk Ševčík"    |
| employee:57                               | "Zdeněk Vomáčka"   |
| employee:60                               | "Zikmund Vyskočil" |
| employee:30                               | "Patrik Skřivánek" |
| employee:58                               | "Filip Tříska"     |
| employee:50                               | "Karel Vomáčka"    |
| employee:9                                | "Hugo Ševčík"      |
| employee:2                                | "Karel Skřivánek"  |
| employee:53                               | "Patrik Novák"     |
| employee:26                               | "Patrik Vyskočil"  |
| employee:63                               | "Erik Skřivánek"   |
| employee:64                               | "Erik Vomáčka"     |
| employee:27                               | "Zdeněk Skřivánek" |
| employee:34                               | "Erik Novák"       |
| employee:5                                | "Jan Novotný"      |
| employee:61                               | "Zdeněk Vyskočil"  |
| employee:18                               | "Erik Vomáčka"     |
-------------------------------------------------------------------
```

Dotaz 25

Zápis v přirozeném jazyce

```
Restaurace a čísla stolů, na které bylo objednáno jídlo v hodnotě minimálně 1000 Kč.
```

Zápis v jazyce SPARQL

```
# Prefix
PREFIX restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>
PREFIX table: <http://wrzecond.fit.cvut.cz/table/>
PREFIX order: <http://wrzecond.fit.cvut.cz/order/>
PREFIX food: <http://wrzecond.fit.cvut.cz/food/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 25. Restaurace a čísla stolů, na které bylo objednáno jídlo v hodnotě minimálně 1000
Kč.
SELECT ?restaurant ?restaurant_name ?table (SUM(?count * ?food_price) AS ?order_price)
WHERE {
    ?restaurant restaurant:name ?restaurant_name .
    ?table table:in ?restaurant .
    ?order order:at ?table ;
           order:count ?count ;
           order:contains/food:price ?food_price .
}
GROUP BY ?restaurant ?restaurant_name ?table HAVING (?order_price >= 1000)
ORDER BY DESC(?order_price)
LIMIT 10
```

Výstup z konzole

```
-------------------------------------------------------------
| restaurant     | restaurant_name | table      | order_price |
=============================================================
| restaurant:5   | "Síla chuti"    | table:19   | 3628        |
| restaurant:5   | "Síla chuti"    | table:24   | 3565        |
| restaurant:3   | "Divinis"       | table:11   | 1999        |
| restaurant:4   | "Harmony"       | table:18   | 1548        |
| restaurant:4   | "Harmony"       | table:14   | 1396        |
| restaurant:4   | "Harmony"       | table:16   | 1347        |
| restaurant:5   | "Síla chuti"    | table:25   | 1249        |
| restaurant:2   | "Café Imperial" | table:8    | 1100        |
-------------------------------------------------------------
```

Experimentální sekce

Ve svém projektu jsem měl možnost si vyzkoušet **různé zápisy** SPARQL dotazů a zkoumat jejich "exekuční plán" (= přepis do SPARQL algebry, optimalizaci) pomocí nástroje `qparse`. V této sekci se nejprve zaměřím na nějaký **přehled dotazů**, a co jsem jimi zamýšlel, a u těch, kde jsem zkoušel více zápisů, provedu **detailní průzkum** a porovnání.

Přehled dotazů

1. SELECT dotazy
2. ASK dotazy
3. DESCRIBE dotazy
4. CONSTRUCT dotazy

SELECT dotazy

SELECT dotazy slouží k vyhledání informací z RDF grafu. V mém projektu se jedná o **většinu dotazů**.

SELECT dotaz se skládá ze **dvou částí**, výběru dat a následně výběru "entit", které chceme zobrazit. Při dotazu se dá využít mnoho **klíčových slov** s různými **funkcemi**, často mají také dva klíčová slova stejnou/velmi blízkou funkci.

Projekt obsahuje **18 SELECT** dotazů *(zbylých 7 jsou ostatní typy dotazů)*, z toho **6** z nich je napsáno ve **dvou formulacích**. V této sekci prozkoumám právě ty dotazy, které jsou napsány ve dvou formulacích *(pro možnost porovnání)* - dotaz 2, dotaz 7, dotaz 8, dotaz 14, dotaz 15 a dotaz 18.

Dotaz 2

Cílem dotazu je najít **restaurace, jejichž manažer se nejmenuje Vendelín**. K tomuto jsem zvolil **dva** možné přístupy - **prvním** je získat všechny kombinace restaurací a manažerů a následně pomocí `FILTER` smazat všechny Vendelíny, **druhým** získat všechny kombinace restaurací a manažerů a odečíst od nich pomocí `MINUS` ty kombinace, kde se manažer jmenuje Vendelín.

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?restaurant ?restaurant_id ?restaurant_name ?manager_name)
  (top (10 ?restaurant_id)
    (extend ((?manager_name (concat ?first_name " " ?last_name)))
      (sequence
        (filter (!= ?first_name "Vendelín")
          (bgp
            (triple ?restaurant restaurant:id ?restaurant_id)
            (triple ?restaurant restaurant:name ?restaurant_name)
            (triple ?restaurant restaurant:managed_by ?manager)
            (triple ?manager manager:first_name ?first_name)
          ))
        (bgp (triple ?manager manager:last_name ?last_name)))))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?restaurant ?restaurant_id ?restaurant_name ?manager_name)
  (top (10 ?restaurant_id)
    (extend ((?manager_name (concat ?first_name " " ?last_name)))
      (minus
        (bgp
          (triple ?restaurant restaurant:id ?restaurant_id)
          (triple ?restaurant restaurant:name ?restaurant_name)
          (triple ?restaurant restaurant:managed_by ?manager)
          (triple ?manager manager:first_name ?first_name)
          (triple ?manager manager:last_name ?last_name)
        )
        (sequence
          (assign ((?first_name "Vendelín"))
            (bgp
              (triple ?restaurant restaurant:id ?restaurant_id)
              (triple ?restaurant restaurant:name ?restaurant_name)
              (triple ?restaurant restaurant:managed_by ?manager)
              (triple ?manager manager:first_name "Vendelín")
            ))
          (bgp (triple ?manager manager:last_name ?last_name)))))))
```

Všimněme si, že zatímco v **prvním přístupu** se filtrování podle jména uplatní až úplně **na konci**, v **druhém přístupu** probíhá ještě před množinovou operací **minus**. V tomto případě však k žádnému zlepšení či zhoršení efektivity **nedochází** - vždy musíme totiž projít všechny kombinace manažerů a restaurací, následně odfiltrovat ty, kde se manažer jmenuje Vendelín *(a množinový rozdíl nám zde příliš nepomůže)*.

K **zefektivnění** by zde mohlo vést například pokud bychom **nejprve** zjistili pouze entity manažerů se jménem jiným než Vendelín, a až **následně** k nim napojili restaurace.

Dotaz 7

Cílem dotazu je najít **neobsazené stoly** v restauraci Harmony. K tomuto jsem použil dva možné přístupy - v obou nejprve získám stoly v restauraci Harmony, v **prvním** následně pomocí `FILTER NOT EXISTS` získám všechny stoly, u kterých někdo sedí *(existuje pro ně vazba customer:sits_at ?table)*. V **druhém** přístupu následně nepovinně pomocí `OPTIONAL` připojím zákazníky sedící u stolu, a odfiltruji stoly, kde zákazník existuje (je `BOUND`).

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?table)
  (top (10 ?table)
    (filter (notexists
              (bgp
                (triple ??P2 restaurant:name "Harmony")
                (triple ?table table:in ??P2)
                (triple ?customer customer:sits_at ?table)
              ))
      (bgp
        (triple ??P3 restaurant:name "Harmony")
        (triple ?table table:in ??P3)
      )))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?table)
  (top (10 ?table)
    (filter (! (bound ?customer))
      (conditional
        (bgp
          (triple ??P1 restaurant:name "Harmony")
          (triple ?table table:in ??P1)
        )
        (bgp (triple ?customer customer:sits_at ?table)))))))
```

Všimněme si, že zatímco v **prvním přístupu** získáváme všechny stoly v restauraci Harmony vlastně **dvakrát**, přičemž se jednou napojí všichni zákazníci, a pak provádíme odfiltrování zbytečných výsledků. **Druhý přístup** je efektivnější, jelikož se výběr všech stolů provede pouze jednou, a filtrování pak probíhá už **pouze na úrovni stolů**.

Dotaz 8

Cílem dotazu je najít **všechny zaměstnance**, kteří **neobsluhují žádný stůl** v restauraci Ostravská. K tomuto jsem zvolil dva přístupy: **prvním** bylo nejprve najít všechny zaměstnance a restaurace, kde pracují, a následně odečíst (MINUS) ty zaměstnance, kteří obsluhují nějaký stůl, a až finálně odfiltrovat zaměstnance z restaurace Ostravská. **Druhým** přístupem bylo nejprve najít všechny zaměstnance z restaurace Ostravská, nepovinně (`OPTIONAL`) na ně stoly, které obsluhují, a pak pomocí `GROUP BY` a `HAVING` najít ty, kde byl počet nulový.

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?employee ?name)
  (top (10 ?employee)
    (filter (= ?restaurant_name "Ostravská")
      (extend ((?name (concat ?first_name " " ?last_name)))
        (minus
          (bgp
            (triple ?employee employee:works_in ??P2)
            (triple ??P2 restaurant:name ?restaurant_name)
            (triple ?employee employee:first_name ?first_name)
            (triple ?employee employee:last_name ?last_name)
          )
          (bgp
            (triple ?employee employee:works_in ??P3)
            (triple ??P3 restaurant:name ?restaurant_name)
            (triple ?table table:operated_by ?employee)
          )))))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?employee ?name)
  (top (10 ?employee)
    (filter (= ?.0 0)
      (extend ((?name (concat ?first_name " " ?last_name)))
        (group (?employee ?first_name ?last_name) ((?.0 (count ?table)))
          (conditional
            (bgp
              (triple ??P1 restaurant:name "Ostravská")
              (triple ?employee employee:works_in ??P1)
              (triple ?employee employee:first_name ?first_name)
              (triple ?employee employee:last_name ?last_name)
            )
            (bgp (triple ?table table:operated_by ?employee)))))))))
```

Již na první pohled je patrné, že **první přístup** je značně neefektivní - zbytečně provádí množinový rozdíl, a to ještě na "cross joinu" restaurací a zaměstnanců. **Druhý přístup** je zde mnohem čistší, lepší a **efektivnější** - rovnou na začátku se zbavíme zaměstnanců mimo restauraci Ostravská a až následně na ně napojíme stoly.

Dotaz 14

Cílem dotazu je najít **zaměstanance** z restaurace "Červené jablko", kteří obsluhují stůl s kapacitou **3 lidí**. V tomto dotazu jsem se tak nezaměřoval na efektivitu různých klíčových slov, ale na efektivitu a **rozdíl** mezi `?a :b "x"` a `?a :b ?c FILTER(?c = "x")` a také rozdíl mezi tvorbě nových proměnných už v dotazu pomocí `BIND` nebo až při SELECTu pomocí `AS`.

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(slice _ 10
  (project (?employee ?name)
    (extend ((?name (concat ?first_name " " ?last_name)))
      (bgp
        (triple ??P1 restaurant:name "Červené jablko")
        (triple ?employee employee:works_in ??P1)
        (triple ?employee employee:first_name ?first_name)
        (triple ?employee employee:last_name ?last_name)
        (triple ?table table:operated_by ?employee)
        (triple ?table table:capacity 3)
      )))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(slice _ 10
  (project (?employee ?name)
    (extend ((?name (concat ?first_name " " ?last_name)))
      (sequence
        (assign ((?restaurant_name "Červené jablko"))
          (sequence
            (filter (exists
                      (project (?employee)
                        (bgp
                          (triple ?/table table:capacity 3)
                          (triple ?/table table:operated_by ?employee)
                        )))
              (bgp (triple ?employee employee:works_in ??P1)))
            (bgp (triple ??P1 restaurant:name "Červené jablko"))))
        (bgp
          (triple ?employee employee:first_name ?first_name)
          (triple ?employee employee:last_name ?last_name)
        ))))))
```

Výsledky qparse jsou **velmi zajímavé**. Zaprvé si můžeme všimnout, že klíčové slovo `FILTER EXISTS` způsobilo rozdělení dotazu na poddotazy, které se následně pomocí "sequence" - množinového sjednocení spojují. Dále si lze všimnout, že SPARQL zoptimalizovalo dotaz tak, že mezi prvním a druhým zápisem **není rozdíl** v přiřazení jména (BIND vs SELECT .. AS). V našem případě je to proto, že jméno nikde v rámci dotazu **nevyužíváme**. Rozdíl mezi zápisem `?a :b "x"` a `?a :b ?c FILTER(?c = "x")` taktéž není, SPARQL ho pokaždé "zoptimalizuje".

Dotaz 15

Cílem dotazu je najít **všechny suroviny** potřebné k přípravě **dvouchodového** obědu. K tomu jsem využil dva přístupy - **první** je získat seznam surovin pokaždé zvlášť a pak tyto seznamy spojit (pomocí UNION), **druhý** je získat kombinaci všech surovin a jídel a pak **vyfiltrovat** ty, které slouží k přípravě jednoho (||, OR) chodu.

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(top (10 (desc ?total_count))
  (distinct
    (project (?ingredient ?ingredient_name ?total_count ?unit)
      (extend ((?total_count ?.0))
        (group (?ingredient ?ingredient_name ?unit) ((?.0 (sum ?ingredient_count)))
          (union
            (bgp
              (triple ?food food:name "Hovězí vývar s celestýnskými nudlemi")
              (triple ?food_ingredient food_ingredient:makes ?food)
              (triple ?food_ingredient food_ingredient:unit ?unit)
              (triple ?food_ingredient food_ingredient:count ?ingredient_count)
              (triple ?food_ingredient food_ingredient:with ?ingredient)
              (triple ?ingredient ingredient:name ?ingredient_name)
            )
            (bgp
              (triple ?food food:name "Rumcajsovy koule")
              (triple ?food_ingredient food_ingredient:makes ?food)
              (triple ?food_ingredient food_ingredient:unit ?unit)
              (triple ?food_ingredient food_ingredient:count ?ingredient_count)
              (triple ?food_ingredient food_ingredient:with ?ingredient)
              (triple ?ingredient ingredient:name ?ingredient_name)
            ))))))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(top (10 (desc ?total_count))
  (distinct
    (project (?ingredient ?ingredient_name ?total_count ?unit)
      (extend ((?total_count ?.0))
        (group (?ingredient ?ingredient_name ?unit) ((?.0 (sum ?ingredient_count)))
          (disjunction
            (assign ((?food_name "Hovězí vývar s celestýnskými nudlemi"))
              (bgp
                (triple ?food food:name "Hovězí vývar s celestýnskými nudlemi")
                (triple ?food_ingredient food_ingredient:makes ?food)
                (triple ?food_ingredient food_ingredient:unit ?unit)
                (triple ?food_ingredient food_ingredient:count ?ingredient_count)
                (triple ?food_ingredient food_ingredient:with ?ingredient)
                (triple ?ingredient ingredient:name ?ingredient_name)
              ))
            (assign ((?food_name "Rumcajsovy koule"))
              (bgp
                (triple ?food food:name "Rumcajsovy koule")
                (triple ?food_ingredient food_ingredient:makes ?food)
                (triple ?food_ingredient food_ingredient:unit ?unit)
                (triple ?food_ingredient food_ingredient:count ?ingredient_count)
                (triple ?food_ingredient food_ingredient:with ?ingredient)
                (triple ?ingredient ingredient:name ?ingredient_name)
              )))))))))
```

Jak si můžeme všimnout, tak opravdu **jediný rozdíl** v "exekučním plánu" těchto dvou formulací je záměna "union" a "disjunction", které jsou obě **stejně efektivní**. Z hlediska přehlednosti dotazu dává větší smysl použití ORu, který má obecně "menší sílu" než UNION. *(Většina dotazů, která lze napsat pomocí OR lze napsat i pomocí UNION)*

Dotaz 18

Cílem dotazu je nalézt **restaurace, ve kterých sedí** alespoň 5 osob starších 60 let. Tento dotaz má dvě formulace, ve kterých jsem zkoumal **rozdíl** mezi efektivitou spojení **přímo** a spojení "**inverzně**". (`?a :b ?c . ?e :d ?c` vs `?a :b/^:d ?e`).

"Exekuční plán" prvního přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?restaurant ?restaurant_name ?old_customers)
  (top (10 (desc ?old_customers))
    (filter (>= ?old_customers 5)
      (extend ((?old_customers ?.0))
        (group (?restaurant ?restaurant_name) ((?.0 (count ?customer_age)))
          (sequence
            (filter (> ?customer_age 60)
              (bgp
                (triple ?table table:in ?restaurant)
                (triple ?restaurant restaurant:name ?restaurant_name)
                (triple ?customer customer:age ?customer_age)
              ))
            (bgp (triple ?customer customer:sits_at ?table)))))))))
```

"Exekuční plán" druhého přístupu je následující *(prefixy jsou vynechány)*:

```
(project (?restaurant ?restaurant_name ?old_customers)
  (top (10 (desc ?old_customers))
    (filter (>= ?old_customers 5)
      (extend ((?old_customers ?.0))
        (group (?restaurant ?restaurant_name) ((?.0 (count ?customer_age)))
          (filter (> ?customer_age 60)
            (bgp
              (triple ?restaurant restaurant:name ?restaurant_name)
              (triple ??P3 table:in ?restaurant)
              (triple ??P2 customer:sits_at ??P3)
              (triple ??P2 customer:age ?customer_age)
            ))))))))
```

V **prvním** "exekučním plánu" mě zaujalo, že se nejprve vyberou všichni zákazníci starší než 60 let a restaurace, ve kterých sedí, a až **následně** se připojí triplet zákazník sedí u stolu. Zde lze vidět, že **SPARQL optimalizátor není všemocný**, jelikož by bylo lepší nejprve pouze zjistit zákazníky starší než 60 let, až následně připojit stoly, zjistit, kde je zákazníků více než pět, a až pak připojit restaurace. U **druhého** "exekučního plánu" již není nic překvapivého, SPARQL engine vyhodnotí "inverzní vazby" a provede dotaz tak, jak bych si to představoval. Nejefektivnější je však způsob, který jsem popsal u prvního dotazu.

ASK dotazy

ASK dotazy jsou v podstatě zredukované SELECT dotazy, kde nás **nezajímají** informace o výsledcích daného dotazu, zajímá nás pouze, zda výsledek existuje, nebo ne.

ASK dotaz se tedy narozdíl od SELECT dotazu skládá pouze z části dotazovací (WHERE), žádné proměnné se zde **nevyskytují**. ASK dotazům v projektu odpovídají dotaz 5, dotaz 10 a dotaz 20.

Pokud zkusíme dotaz 5 prozkoumat pomocí příkazu **qparse**, zjistíme následující "exekuční plán":

```
(prefix ((table: <http://wrzecond.fit.cvut.cz/table/>)
         (food: <http://wrzecond.fit.cvut.cz/food/>)
         (restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>)
         (table_type: <http://wrzecond.fit.cvut.cz/table/type/>)
         (order: <http://wrzecond.fit.cvut.cz/order/>))
  (filter (notexists
            (minus
              (bgp (triple ?food food:id ?food_id))
              (project (?food_id)
                (join
                  (bgp
                    (triple ?/order order:at ?/table)
                    (triple ?/order order:contains ?/food)
                    (triple ?/food food:id ?food_id)
                  )
                  (slice _ 1
                    (project (?/table ?/table_type ?/food_count)
                      (filter (= ?/food_count ?//food_total)
                        (extend ((?/food_count ?//.0))
                          (group (?/table ?/table_type ?//food_total) ((?//.0 (count
?//food)))
                            (join
                              (sequence
                                (assign ((?//restaurant_name "Síla chuti"))
                                  (bgp (triple ?//restaurant restaurant:name "Síla
chuti")))
                                (bgp
                                  (triple ?/table table:in ?//restaurant)
                                  (triple ?/table table:type ?//type)
                                  (triple ?//type table_type:name ?/table_type)
                                  (triple ?//order order:at ?/table)
                                  (triple ?//order order:contains ?//food)
                                ))
                              (project (?//food_total)
                                (extend ((?//food_total ?///.0))
                                  (group () ((?///.0 (count ?///food)))
                                    (bgp (triple ?///food food:id
?///food_id))))))))))))))))
    (table unit)))
```

Poznámka: FILTER NOT EXISTS je využit pro inverzi významu Yes/No v ASK dotazu.

Zaprvé si můžeme všimnout, že nahoře nevidíme žádný "project" příkaz, to proto, že v dotazu **není** část s **výběrem proměnných** *(ani nedává smysl vzhledem k povaze ASK odtazu)*.

Dále mě docela překvapuje, že na vrcholu dotazu **není** žádný slice 1 (ekvivalent SQL LIMIT 1). Působí to tedy tak, že **efektivita ASK dotazu** oproti ekvivalentnímu SELECT dotazu tak může být vyšší pouze kvůli eliminaci nutnosti **bindování** proměnných.

U **dalších ASK dotazů** nedochází k žádným speciálním operacím, které by stály za zmínku v této sekci, proto je zde **nebudu rozebírat**.

Shrnutí ASK: Zkoumáním "exekučního plánu" ASK dotazů jsem zjistil, že dochází pouze k provedení "SELECT" dotazu, a to bez TOP-level projekce či slice výsledků, což mě docela **překvapilo**. Jako hlavní přínos ASK dotazů vidím **jednoduchost odpovědi** - prosté "Yes" nebo "No" (které může vést k efektivnějšímu dotazu).

DESCRIBE dotazy

DESCRIBE dotazy slouží k zjištění podrobných informací (vypsání RDF dat ve formátu Turtle) o proměnných ve výsledku určitého dotazu.

Pokud bychom tedy například v rámci naší databáze měli **vztah mezi restaurací** (wrzecond/restaurant) **a městem** například z DBpedia.org, mohli bychom pomocí dotazu DESCRIBE zjistit více informací o městě, ve kterém je restaurace z daného dotazu.

V semestrální práci je pouze jeden DESCRIBE dotaz, a to dotaz 13.

DESCRIBE dotaz se opět skládá ze **dvou částí** - nejprve provedeme (podobně jako u SELECTu) výběr dat, a následně SPARQL engine provede popis hodnot (původních dat z RDF souboru).

Pokud zkusíme dotaz 13 prozkoumat pomocí příkazu **qparse**, zjistíme následující "exekuční plán":

```
(prefix ((table: <http://wrzecond.fit.cvut.cz/table/>)
         (food: <http://wrzecond.fit.cvut.cz/food/>)
         (restaurant: <http://wrzecond.fit.cvut.cz/restaurant/>)
         (table_type: <http://wrzecond.fit.cvut.cz/table/type/>)
         (order: <http://wrzecond.fit.cvut.cz/order/>))
  (project (?table)
    (project (?table ?table_type ?food_count)
      (top (10 ?table)
        (filter (= ?food_count ?/food_total)
          (extend ((?food_count ?/.0))
            (group (?table ?table_type ?/food_total) ((?/.0 (count ?/food)))
              (join
                (sequence
                  (assign ((?/restaurant_name "Síla chuti"))
                    (bgp (triple ?/restaurant restaurant:name "Síla chuti")))
                  (bgp
                    (triple ?table table:in ?/restaurant)
                    (triple ?table table:type ?/type)
                    (triple ?/type table_type:name ?table_type)
                    (triple ?/order order:at ?table)
                    (triple ?/order order:contains ?/food)
                  ))
                (project (?/food_total)
                  (extend ((?/food_total ?//.0))
                    (group () ((?//.0 (count ?//food)))
                      (bgp (triple ?//food food:id ?//food_id)))))))))))
```

Oproti stejnému SELECT dotazu (SELECT ?table místo DESCRIBE ?table) si můžeme všimnout, že se zde neprovádí nic navíc (viz. vysvětlení dotazu 4 v kapitole Experimenty - SELECT)

Shrnutí DESCRIBE: Zkoumáním "exekučního plánu" DESCRIBE dotazů jsem zjistil, že se **nejprve** provádí SELECT (klasický výběr), následně se ze získaných výsledků vytáhnou **informace z datových souborů**.

CONSTRUCT dotazy

CONSTRUCT dotazy slouží k vytvoření RDF grafu z již existujícího. Jedná se o dotaz 6, dotaz 11, dotaz 12 a dotaz 16.

CONSTRUCT dotaz se skládá ze **dvou částí**, výběru dat a následné "konstrukci" nového grafu. V dotazu 6 tak například dochází k vytvoření nových vazeb "employee:position" a "employee:order_count".

Pokud zkusíme dotaz 6 prozkoumat pomocí příkazu **qparse**, zjistíme následující "exekuční plán":

```
(prefix ((employee: <http://wrzecond.fit.cvut.cz/employee/>)
         (table: <http://wrzecond.fit.cvut.cz/table/>)
         (order: <http://wrzecond.fit.cvut.cz/order/>))
  (slice _ 15
    (project (?employee ?position ?order_count)
      (extend ((?order_count ?.0))
        (group (?employee ?position) ((?.0 (count ?order)))
          (extend ((?position (if ?cook "Kuchař a Číšník" "Číšník")))
            (conditional
              (bgp
                (triple ?employee employee:works_in ?restaurant)
                (triple ?employee employee:salary ?salary)
                (triple ?employee employee:cook ?cook)
              )
              (bgp
                (triple ?table table:operated_by ?employee)
                (triple ?order order:at ?table)
              ))))))))
```

Zaprvé si můžeme všimnout, že oproti případnému SELECT dotazu se zde neobjevuje žádné klíčové slovo "construct" ani popis vytváření nového grafu (a vazeb employee:position...). Lze z toho tedy vyvodit, že vyhodnocení výběru dat se provádí **samostatně** a vytvoření grafu s vazbami je až následující krok.

Dále již na dotazu není nic zajímavého, provedou se dvě basic graph pattern matches, přidá se k nim proměnná **position**, seskupí se, přičemž se vytvoří proměnná **order_count** a následně se provede projekce a výběr TOP 15 výsledků *(připomínám, že v CONSTRUCTu omezení na TOP 15 provádím pouze proto, aby se do zprávy k projektu vešel výsledek (a neměl 3 A4).*

U **dalších CONSTRUCT dotazů** již dochází víceméně k podobným operacím, akorát s jinými entitami, proto to zde **nebudu rozebírat**.

Shrnutí CONSTRUCT: Zkoumáním "exekučního plánu" CONSTRUCT dotazů jsem zjistil, že se **nejprve** provádí SELECT (klasický výběr), a až **následně** je provedeno vytvoření nového grafu s vybranými vazbami.

Experimentální sekce - shrnutí

V **semestrálním projektu** z předmětu BI-VWM jsem měl možnost **experimentovat** s různými zápisy dotazů v jazyce SPARQL, zkoumat jejich **efektivitu** a zjišťovat, **jak** který (SELECT, ASK, DESCRIBE, CONSTRUCT) **pracuje**.

Také jsem se mohl aktivně **přesvědčit**, že SPARQL jakožto dotazovací jazyk nad RDF, které je součástí **sémantického webu**, nabízí mnohem lepší možnosti vyhledávání než **klasický web** a pomocí možnosti propojení jednotlivých endpointů také občas vítězí i nad **relační databází** (zde je ale nevýhodou to, že aby to propojení mohlo nastat, musí se jednotlivé endpointy, stránky, na sebe navázat).

Diskuze

Při tvorbě semestrální práce jsem využil jakési **částečné datové schéma**, a to jednak z hlediska **reálného života**, tak i z hlediska **Linked data**.

Z hlediska **reálného života** byla již při popisu projektu patrná některá omezení, která by nebyla v reálném informačním systému restaurace smysluplná. Například omezení *"Každý kuchař je zároveň i číšník"*, kdy je zřejmé, že v opravdové restauraci opravdu starší kuchařka se zástěrou v hospodě nepůjde obsluhovat hosty. Nebo *"Příprava jídel probíhá až po objednávce"*, kdy by jednak takovýto koncept neumožňoval případné reklamace a především by časově nebylo vůbec reálné takto restauraci vést a stíhat. Také chybí nějaká historie objednávek.

Z hlediska **Linked data** a **SPARQL** je zde nejpatrnější "omezení" absence napojení na zbytek linked data - v rámci systému jsou sice jednotlivé entity napojeny, svět Linked data by nám však bez problému umožňoval například napojení adres restaurací na reálná města, recepty a suroviny by mohly být napojeny do nějaké linked-data databáze receptů, lidé by mohli být napojeni do "centrálního registru osob" *(i když tady to by fungovalo nejspíše především v Číně a Severní Korei)*.

Závěr

V semestrálním projektu z předmětu BI-VWM jsem si vyzkoušel **vytvoření informačního systému restaurace**, nejprve **datově** ve formátu RDF a následně jsem si také vyzkoušel napsat pár dotazů v jazyce **SPARQL** nad tímto systémem.

Jsem velmi rád, že jsem mohl implementovat právě téma **RDF** + **SPARQL**, především proto, že jsem implementoval systém, který už jsem měl jednou napsaný v relační databázi ORACLE a dotazy v jazyce SQL. Mohl jsem se tak plně soustředit na **rozdíly** mezi jednotlivými dotazovacími jazyky a také rozdíly v datových formátech.

Koncept **Linked data, sémantického webu a SPARQL** se mi líbí - jednak proto, že dotazování pomocí SPARQL působí **logičtěji** a méně zamotaněji, než mnoho JOIN, LEFT OUTER JOIN a podobných operací nad SQL databází, ale především proto, že jednotlivé systémy **lze na sebe napojit** (jak jsem již rozebral v kapitole Diskuze - sekci Linked data).

Semestrální práce BI-VWM