Technologie semantyczne i sieci społecznościowe – laboratorium

Oracle Semantic Technologies

Celem ćwiczenia jest zapoznanie studentów z narzędziem Oracle Semantic Technologies oraz przedstawienie sposobów zapisywania danych semantycznych za pomocą silnika relacyjnej bazy danych i ich przetwarzania za pomocą języka SQL. W pierwszej części ćwiczenia studenci poznają sposób składowania danych semantycznych w bazie danych oraz ćwiczą definiowanie ontologii za pomocą języków RDFS i OWL. Druga część ćwiczenia obrazuje sposób definiowania nowych własności, efekt budowania bazy reguł, a także możliwość wzbogacenia istniejących danych relacyjnych za pomocą dołączonej ontologii.

1. Utwórz tabelę do fizycznego przechowywania danych semantycznych, a następnie utwórz model, który będzie stanowił logiczny kontener dla danych semantycznych. **UWAGA**: nazwa modelu musi być unikalna (wykorzystaj numer indeksu)

```
SQL> CREATE TABLE emp_rdf (
       id NUMBER,
       triple SDO_RDF_TRIPLE_S);

SQL> EXECUTE SEM_APIS.CREATE_RDF_MODEL(
      'Employees',                          -- nazwa modelu
      'emp_rdf',                            -- nazwa tabeli
      'triple');                            -- nazwa kolumny
```

2. Zapoznaj się z sygnaturą typów obiektowych **SDO_RDF_TRIPLE** i **SDO_RDF_TRIPLE_S**

```
SQL> DESCRIBE SDO_RDF_TRIPLE

SQL> DESCRIBE SDO_RDF_TRIPLE
```

3. Wstaw do bazy danych kilka przykładowych faktów

```
-- King jest menadżerem Blake'a
SQL> INSERT INTO emp_rdf VALUES (1, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/King',
  'http://semantic.cs.put.poznan.pl/emp/managerOf',
  'http://semantic.cs.put.poznan.pl/emp/Blake'));

-- Allen pracuje jako Salesman
SQL> INSERT INTO emp_rdf VALUES (2, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Allen',
  'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
  'http://semantic.cs.put.poznan.pl/emp/Salesman'));

-- Ford zarabia 3000
SQL> INSERT INTO emp_rdf VALUES (3, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Ford',
  'http://semantic.cs.put.poznan.pl/emp/salary',
  '"3000"^^xsd:decimal'));

-- departament Accounting znajduje się w Nowym Jorku
```

```
SQL> INSERT INTO emp_rdf VALUES (4, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Accounting',
  'http://semantic.cs.put.poznan.pl/emp/locatedIn',
  'http://semantic.cs.put.poznan.pl/emp/NewYork'));
```

4. Odczytaj wstawione do bazy danych informacje wykorzystując interfejs typu obiektowego **SDO_RDF_TRIPLE_S**

```
SQL> SELECT e.triple.get_subject() AS subject,
            e.triple.get_property() AS property,
            e.triple.get_object() AS object
     FROM emp_rdf e;
```

5. Wydaj przykładowe zapytanie wykorzystując język SPARQL i funkcję SEM_MATCH

```
SQL> SELECT m, n
     FROM TABLE(
       SEM_MATCH('(?m :managerOf ?n)',
       SEM_MODELS('Employees'),
       null,
       SEM_ALIASES(
         SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
       null));

SQL> SELECT m, s
     FROM TABLE(
       SEM_MATCH('(?m :salary ?s)',
       SEM_MODELS('Employees'),
       null,
       SEM_ALIASES(
         SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
       null))
     WHERE s > 2500;
```

6. Zwróć uwagę na możliwość wykorzystywania domyślnych przestrzeni nazw w funkcji SEM_MATCH:
   - **('rdf', 'http://www.w3.org/1999/02/22-rdf-syntax-ns#')**
   - **('rdfs', 'http://www.w3.org/2000/01/rdf-schema#')**
   - **('xsd', 'http://www.w3.org/2001/XMLSchema#')**

7. Usuń poprzednio stworzony zbiór reguł (ten krok nie jest wymagany w przypadku pierwszego uruchomienia) i utwórz nowy zbiór reguł. UWAGA: zbiór reguł musi mieć unikalną nazwę (posłuż się numerem indeksu)

```
SQL> BEGIN
       SEM_APIS.DROP_ENTAILMENT('owl_rix_employees');
     END;

SQL> BEGIN
       SEM_APIS.CREATE_ENTAILMENT('owl_rix_employees',
         SEM_MODELS('Employees'),
         SEM_RULEBASES('OWLPRIME'));
     END;
```

8. Dodaj informację o tym, że nazwisko Kinga to “King” oraz że Blake jest menadżerem Jonesa. Następnie, dodaj informację o tym, że bycie przełożoną(ym) jest relacją przechodnią. Aby można było powiedzieć cokolwiek na temat bycia przełożoną(ym) należy uprzednio poinformować bazę danych, że „managerOf” jest cechą o której można formułować sądy.

```
-- King ma nazwisko "King"
SQL> INSERT INTO emp_rdf VALUES (5, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/King',
  'http://semantic.cs.put.poznan.pl/emp/hasName',
  '"King"^^xsd:string')
);

-- Blake jest menadżerem Jonesa
SQL> INSERT INTO emp_rdf VALUES (6, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Blake',
  'http://semantic.cs.put.poznan.pl/emp/managerOf',
  'http://semantic.cs.put.poznan.pl/emp/Jones')
);

-- bycie przełożoną(ym) jest cechą
SQL> INSERT INTO emp_rdf VALUES (7, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/managerOf',
  'rdf:type',
  'rdf:Property')
);

SQL> INSERT INTO emp_rdf VALUES (8, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/managerOf',
  'rdf:type',
  'owl:TransitiveProperty')
);
```

9. Wyświetl wszystkich podwładnych Kinga. Sprawdź, jaki wpływ na wynik zapytania ma włączenie wnioskowania w bazie danych. Przed wydaniem zapytania wykonaj kroki z pkt. 7 aby przebudować indeks reguł.

```
SQL> SELECT n
  FROM TABLE( SEM_MATCH('(?m :managerOf ?n) (?m :hasName "King")',
    SEM_MODELS('Employees'), null,
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),null));

SQL> SELECT n
  FROM TABLE( SEM_MATCH('(?m :managerOf ?n) (?m :hasName "King")',
    SEM_MODELS('Employees'), SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),null));
```

10. Dodaj do bazy danych informacje o tym, że Blake zna Scotta oraz że relacja znajomości jest symetryczna.

```
-- Blake zna Scotta
SQL> INSERT INTO emp_rdf VALUES (9, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Blake',
  'http://semantic.cs.put.poznan.pl/emp/knows',
  'http://semantic.cs.put.poznan.pl/emp/Scott'));

-- knows jest cechą symetryczną
SQL> INSERT INTO emp_rdf VALUES (10, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/knows',
  'rdf:type',
  'owl:SymmetricProperty'));
```

11. Wyświetl pary pracowników, którzy się nawzajem znają. Przed wydaniem zapytania wykonaj kroki z pkt. 7 aby przebudować indeks reguł.

```
-- znajdź pary pracowników postaci X zna Y
SQL> SELECT m,n
  FROM TABLE( SEM_MATCH('(?m :knows ?n)',
    SEM_MODELS('Employees'),SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),null));
```

12. Dodaj do bazy danych informację o tym, że Allen zarabia 5000. Wykorzystaj do tego predykat „pay”. Zauważ, że w bazie danych występują teraz dwa różne predykaty o tym samym znaczeniu: „salary” i „pay”. Dodaj informację o tym, że są to cechy równoważne.

```
-- Allen zarabia 5000
SQL> INSERT INTO emp_rdf VALUES (11, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Allen',
  'http://semantic.cs.put.poznan.pl/emp/pay',
  '"5000"^^xsd:decimal'));

-- "pay" i "salary" to równoważne własności
SQL> INSERT INTO emp_rdf VALUES (12, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/salary',
  'owl:equivalentProperty',
  'http://semantic.cs.put.poznan.pl/emp/pay'));
```

13. Wyświetl pracowników, którzy zarabiają powyżej 2000. Przed wydaniem zapytania wykonaj kroki z pkt. 7 aby przebudować indeks reguł

```
SQL> SELECT e, s
  FROM TABLE( SEM_MATCH('(?e :salary ?s)',
    SEM_MODELS('Employees'), SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
    null, null))
  WHERE s > 2000;
```

14. Dodaj do bazy danych nową cechę „subordinateOf” która będzie odwrotnością cechy „managerOf”. Wykorzystując nowo dodaną cechę wyświetl wszystkich podwładnych Kinga. Przed wydaniem zapytania wykonaj kroki z pkt. 7 aby przebudować indeks reguł. Zauważ, że w bazie danych nie ma ani jednego jawnie wyspecyfikowanego faktu w postaci `X subordinateOf Y`.

```
-- bycie podwładną(ym) jest odwrotnością bycia przełożoną(ym)
SQL> INSERT INTO emp_rdf VALUES (13, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/subordinateOf',
  'owl:inverseOf',
  'http://semantic.cs.put.poznan.pl/emp/managerOf'));

-- przebuduj indeks reguł

SQL> SELECT m, n
  FROM TABLE(SEM_MATCH('(?m :subordinateOf ?n)(?n :hasName "King")',
    SEM_MODELS('Employees'), SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
    null, null));
```

15. Dodaj do bazy danych klasy reprezentujące pracowników i menadżerów. Zdefiniuj menadżerów jako podzbiór pracowników. Przypisz Allena do pracowników, a Kinga do menadżerów.

```
-- Employee i Manager to klasy
SQL> INSERT INTO emp_rdf VALUES (14, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Employee',
  'rdf:type',
  'rdfs:Class'));

SQL> INSERT INTO emp_rdf VALUES (15, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Manager',
  'rdf:type',
  'rdfs:Class'));

-- menadżerowie są podzbiorem pracowników
SQL> INSERT INTO emp_rdf VALUES (16, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Manager',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/Employee'));

-- King jest menadżerem a Allen pracownikiem
SQL> INSERT INTO emp_rdf VALUES (17, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/King',
  'rdf:type',
  'http://semantic.cs.put.poznan.pl/emp/Manager'));

SQL> INSERT INTO emp_rdf VALUES (18, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Allen',
  'rdf:type',
  'http://semantic.cs.put.poznan.pl/emp/Employee'));
```

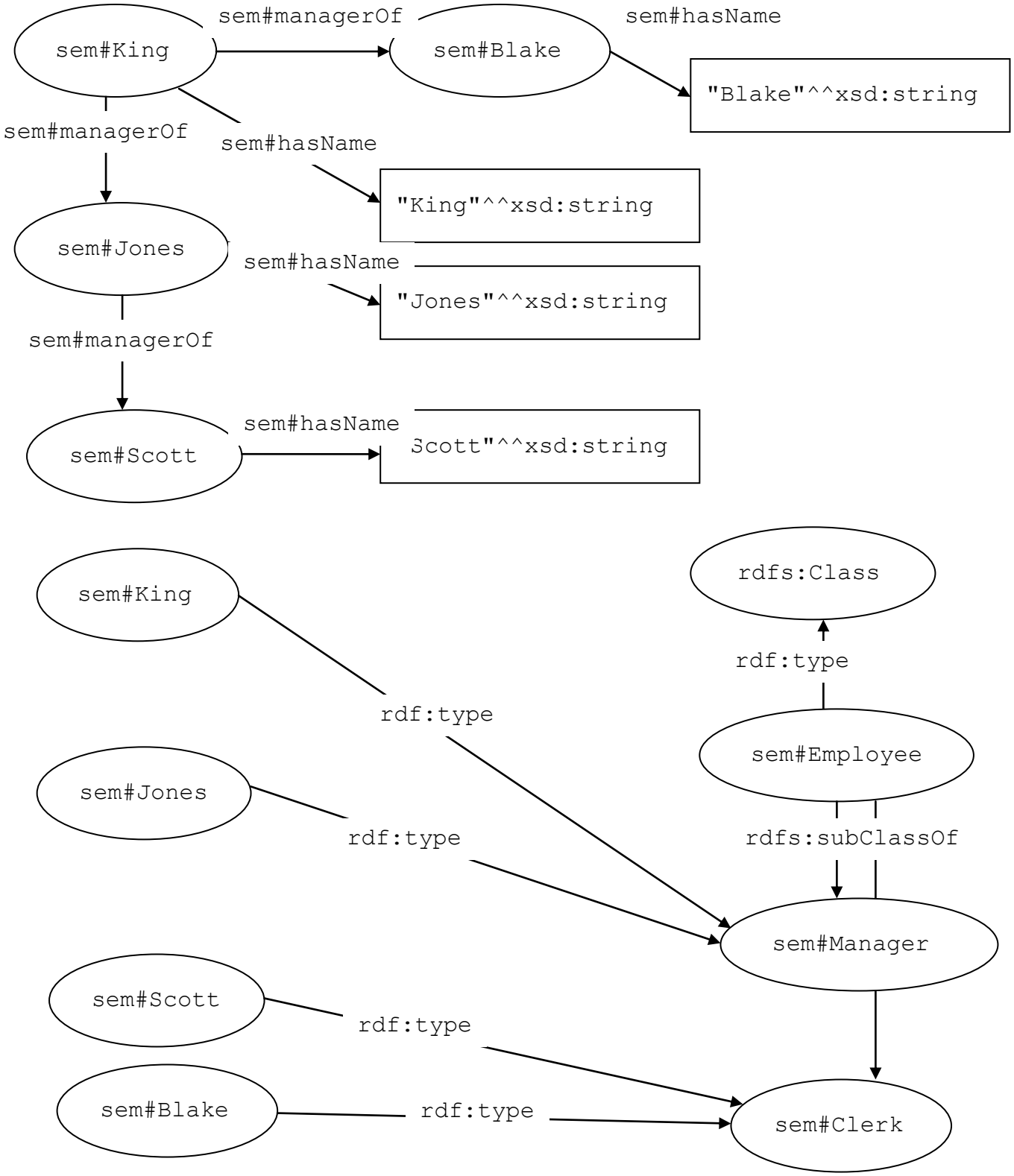
16. Wyświetl wszystkich pracowników. Przed wydaniem zapytania wykonaj kroki z pkt. 7 aby przebudować indeks reguł. Następnie wyświetl wszystkich menadżerów i porównaj uzyskane wyniki.

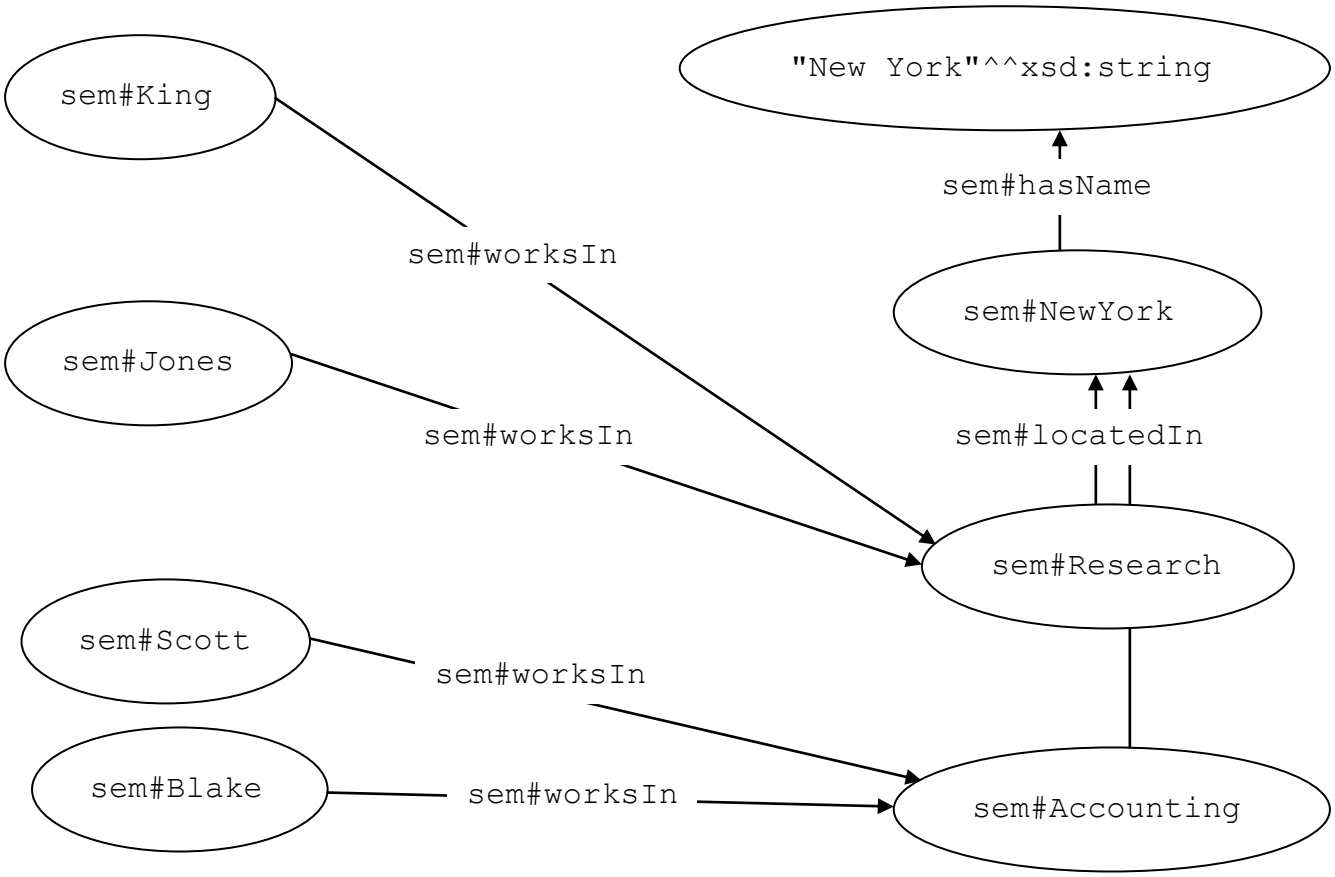
```
SQL> SELECT m
  FROM TABLE(SEM_MATCH('(?m rdf:type :Employee)',
    SEM_MODELS('Employees'), SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
    null, null));

SQL> SELECT m
  FROM TABLE(SEM_MATCH('(?m rdf:type :Manager)',
    SEM_MODELS('Employees'), SDO_RDF_RULEBASES('OWLPRIME'),
    SEM_ALIASES(
      SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
    null, null));
```

Zadanie samodzielne

Utwórz tabelę do przechowywania danych semantycznych, stwórz model, a następnie uzupełnij zbudowaną przez siebie tabelę w taki sposób, aby odpowiadała poniższej rzeczywistości.

sem# oznacza fragment URI: `http://semantic.cs.put.poznan.pl/emp#`





Uzupełnij zbudowaną w kroku (1) bazę wiedzy o następujące informacje:

- Zdefiniuj symetryczną właściwość **worksWith** i odnotuj w bazie danych fakt, że pracownicy Scott i Blake pracują razem. Wyświetl pary wszystkich pracowników pracujących razem.
- Dodaj informację o tym, że właściwość **isManagerOf** jest tranzytywna. Wyświetl wszystkich pracowników, których menedżerem jest King.
- Zdefiniuj właściwość **isHeadOf** wiążącą pracowników i departamenty. Odnotuj fakt, że pracownik King kieruje departamentem Research.
- Zdefiniuj właściwość **isChairOf** wiążącą pracowników i departamenty. Odnotuj fakt, że pracownik Blake kieruje departamentem Accounting.
- Wprowadź informację o tym, że właściwości **isHeadOf** i **isChairOf** oznaczają w rzeczywistości to samo. Następnie, dla każdego departamentu wyświetl nazwę departamentu i nazwisko pracownika kierującego departamentem.
- Dodaj właściwość **isChairedBy** jako właściwość odwrotną do właściwości **isCharOf**. Wykorzystaj nowo zdefiniowaną właściwość aby wyświetlić, dla każdego departamentu, nazwę departamentu i nazwisko pracownika kierującego departamentem.

17. Zdefiniuj ograniczenie dotyczące własności `isHeadOf`. Kierującymi departamentami mogą być jedynie osoby zatrudnione na etacie `Manager`.

```
SQL> INSERT INTO emp_rdf VALUES (100, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/isHeadOf',
  'rdfs:domain',
  'http://semantic.cs.put.poznan.pl/emp/Manager'));
```

18. Wstaw do bazy danych sprzeczne informacje o tym, że King i Blake są i jednocześnie nie są tym samym. Następnie, przebuduj indeks reguł.

```
SQL> INSERT INTO emp_rdf VALUES (101, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/King',
  'owl:sameAs',
  'http://semantic.cs.put.poznan.pl/emp/Blake'));

SQL> INSERT INTO emp_rdf VALUES (101, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/King',
  'owl:differentFrom',
  'http://semantic.cs.put.poznan.pl/emp/Blake'));

SQL> BEGIN
       SEM_APIS.DROP_ENTAILMENT('owl_rix_employees');
     END;
/

SQL> BEGIN
       SEM_APIS.CREATE_ENTAILMENT('owl_rix_employees',
         SEM_MODELS('Employees1'), SEM_RULEBASES('OWLPRIME'));
     END;
/
```

19. Wykorzystaj funkcję `VALIDATE_MODEL` z pakietu `SEM_APIS` do walidowania poprawności swojego modelu.

```
SQL> SELECT SEM_APIS.VALIDATE_MODEL( SEM_MODELS('Employees') )
     FROM dual;
```

20. Wykorzystaj procedurę VALIDATE_ENTAILMENT z pakietu SEM_APIS do walidowania poprawności swojego modelu.

```
SQL> DECLARE
       lva mdsys.rdf_longVarcharArray;
       idx int;
     BEGIN
       lva := SEM_APIS.VALIDATE_ENTAILMENT(
         SEM_MODELS('Employees'), SEM_RULEBASES('OWLPRIME')) ;

     IF (lva IS NULL) THEN
       DBMS_OUTPUT.PUT_LINE('No errors');
     ELSE
       FOR idx IN 1..lva.count LOOP
         DBMS_OUTPUT.PUT_LINE('Error := ' || lva(idx));
       END LOOP;
     END IF;
   END;
/
```

21. Utwórz własną bazę reguł

```
SQL> EXECUTE SEM_APIS.CREATE_RULEBASE('EmpRules');
```

22. Zdefiniuj własną regułę wywodzenia faktów, która pozwoli na wydedukowanie, że dwoje pracowników jest współpracownikami (jest tak, jeśli dwoje pracowników współdzieli przełożoną/przełożonego). **UWAGA**: zwróć uwagę na nazwę tabeli, w której przechowywane są reguły zdefiniowane przez użytkownika.

```
SQL> INSERT INTO mdsys.semr_EmpRules VALUES
     ('WorksTogetherWithRule',
      '(?x :managerOf ?y) (?x :managerOf ?z)', 'y != z',
      '(?y :worksTogetherWith ?z)',

SEM_ALIASES(SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')));
```

23. Przebuduj indeks reguł uwzględniając zbudowaną przez siebie bazę reguł.

```
SQL> BEGIN
       SEM_APIS.DROP_ENTAILMENT('owl_rix_employees');
     END;
/

SQL> BEGIN
       SEM_APIS.CREATE_ENTAILMENT('owl_rix_employees',
         SEM_MODELS('Employees'), SEM_RULEBASES('OWLPRIME','EmpRules'),
         null, null, 'USER_RULES=T');
     END;
/
```

24. Wyświetl wszystkich pracowników którzy posiadają tę samą przełożoną/tego samego przełożonego.

```
SQL> SELECT x,y
     FROM TABLE( SEM_MATCH('(?x :worksTogetherWith ?y)',
       SEM_MODELS('Employees'),
       SDO_RDF_RULEBASES('OWLPRIME','EmpRules'),

SEM_ALIASES(SEM_ALIAS('','http://semantic.cs.put.poznan.pl/emp/')),
       null,null));
```

25. Dodaj do bazy danych informacje o tym, że Nowy Jork i Boston leżą w regionie US-East (tj. że lokalizacje NewYork i Boston są podklasami klasy US-East) a Dallas i Chicago leżą w regionie US-West. Dodaj także informację o tym, że regiony US-East i US-West są podklasami regionu US-All.

```
-- Nowy Jork leży w regionie US-East
SQL> INSERT INTO emp_rdf VALUES (102, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/NewYork',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-East'));

-- Boston leży w regionie US-East
SQL> INSERT INTO emp_rdf VALUES (103, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Boston',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-East'));

-- Dallas leży w regionie US-West
SQL> INSERT INTO emp_rdf VALUES (104, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Dallas',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-West'));

-- Chicago leży w regionie US-West
SQL> INSERT INTO emp_rdf VALUES (105, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/Chicago',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-West'));

-- US-East jest częścią US-All
SQL> INSERT INTO emp_rdf VALUES (106, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/US-East',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-All'));

-- US-West jest częścią US-All
SQL> INSERT INTO emp_rdf VALUES (107, SDO_RDF_TRIPLE_S('Employees',
  'http://semantic.cs.put.poznan.pl/emp/US-West',
  'rdfs:subClassOf',
  'http://semantic.cs.put.poznan.pl/emp/US-All'));
```

26. Zmodyfikuj zawartość tabeli DEPT w taki sposób, aby można było dokonać połączenia wyniku wnioskowania z ontologii z zawartością tabeli: usuń z nazw miast spacje oraz doklej URI ontologii. **UWAGA**: wcześniej utwórz swoją własną kopię tabeli DEPT.

```
SQL> CREATE TABLE dept_xxxxx AS
     SELECT * FROM scott.dept;

SQL> ALTER TABLE dept_xxxxx
     MODIFY loc VARCHAR2(100);

SQL> UPDATE dept_xxxxx
     SET loc = '<http://semantic.cs.put.poznan.pl/emp/' ||
               replace(initcap(loc),' ','') || '>' ;
```

27. Ponieważ funkcja SEM_DISTANCE nie potrafi wyznaczyć odległości w przypadku wnioskowania na podstawie własnego zbioru reguł, przebuduj ponownie indeks reguł uwzględniając jedynie wbudowany zbiór reguł.

```
SQL> BEGIN
       SEM_APIS.DROP_ENTAILMENT('owl_rix_employees');
     END;
/

SQL> BEGIN
      SEM_APIS.CREATE_ENTAILMENT('owl_rix_employees',
      SEM_MODELS('Employees'), SEM_RULEBASES('OWLPRIME'));
END;
/
```

28. Wykorzystaj funkcję SEM_RELATED do wyświetlenia wszystkich pracowników zatrudnionych w regionie US-West. Wyświetl także odległość między poszczególnymi krotkami i kryterium wyszukiwania wyliczoną przez funkcję SEM_DISTANCE. Następnie, zmień kryterium wyszukiwania na US-All i zaobserwuj zmianę w wyliczonej odległości.

```
-- pracownicy zatrudnieni w regionie US-West
SQL> SELECT ename, dname, loc, SEM_DISTANCE(1)
     FROM emp NATURAL JOIN dept
     WHERE SEM_RELATED(loc,
       'rdfs:subClassOf',
       '<http://semantic.cs.put.poznan.pl/emp/US-West>',
       SEM_MODELS('Employees'), SEM_RULEBASES('OWLPRIME'), 1) = 1;
```

29. W ostatnim zadaniu wygeneruj ponownie indeks reguł, tym razem tworząc także bazę wywodów wnioskowania. Po przebudowaniu indeksu wyświetl reguły wywodzenia.

```
SQL> BEGIN
       SEM_APIS.CREATE_ENTAILMENT('owl_rix_employees',
         SEM_MODELS('Employees'), SEM_RULEBASES('OWLPRIME'),
         SEM_APIS.REACH_CLOSURE, 'SAM', 'PROOF=T');
     END;
/
SQL> SELECT link_id || ' generated by ' || explain AS proof
     FROM mdsys.semi_owl_rix_employees;
```

30. Wybierz dowolną wyświetloną regułę wywodzenia i wyświetl jej szczegóły wykorzystując poniższe zapytanie. W warunku IN umieść identyfikatory wybranych przez siebie krotek.

```
SQL> SELECT to_char(x.triple.rdf_m_id, 'FMXXXXXXXXXXXXXXXX') ||'_'||
       to_char(x.triple.rdf_s_id, 'FMXXXXXXXXXXXXXXXX') ||'_'||
       to_char(x.triple.rdf_p_id, 'FMXXXXXXXXXXXXXXXX') ||'_'||
       to_char(x.triple.rdf_c_id, 'FMXXXXXXXXXXXXXXXX'),
       x.triple.get_triple()
   FROM (
   SELECT sdo_rdf_triple_s(
            t.canon_end_node_id,
            t.model_id,
            t.start_node_id,
            t.p_value_id,
            t.end_node_id) triple
     FROM (SELECT * from mdsys.semm_employees union all
           SELECT * from mdsys.semi_owl_rix_employees
          ) t
    WHERE t.link_id IN
     ('4C_ABDFAC71018E980_B936ED76D03C50C_24253581B9CD1CD4',
      '33_3BE88D34D52C59DC_B936ED76D03C50C_24253581B9CD1CD4')
    ) x;
```

Zadanie samodzielne

1. Wykorzystaj bazę wiedzy zbudowaną w pierszej części ćwiczenia i uzupełnij ją o następujące informacje:

- Zdefiniuj ograniczenie dla właściwości **`managerOf`** i wskaż, że przełożoną(ym) może być tylko osoba zatrudniona na etacie **`Manager`**.
- Wykorzystując właściwość **`managerOf`** zdefiniuj nową właściwość **`haveSameManager`**. Utwórz potrzebne struktury (*rulebase*, *entailment*) i wyświetl pary nazwisk pracowników posiadających tego samego menedżera.

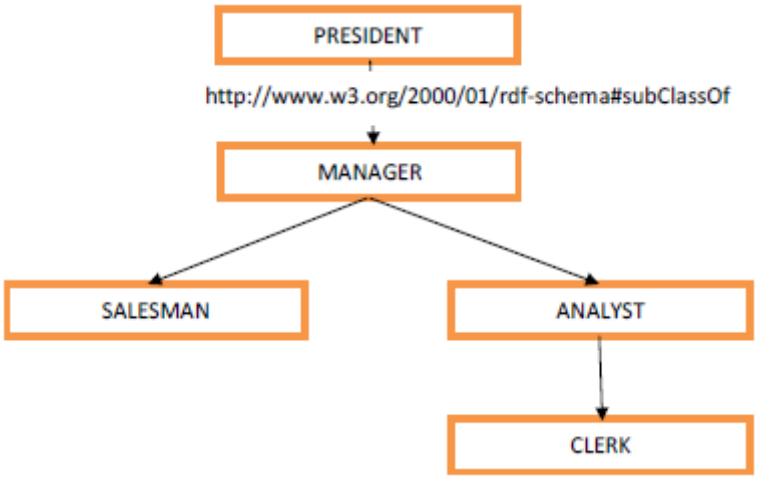
2. Wzbogać dane relacyjne poprzez umożliwienie wykorzystania wnioskowania na podstawie dodatkowych informacji semantycznych umieszczonych w ontologii.
Zdefiniuj hierarchię lokalizacji departamentów

- Nowy Jork i Boston leżą w obszarze EAST-US (rdf-schema#subClassOf)
- Dallas i Chicago leżą w obszarze WEST-US (rdf-schema#subClassOf)
- Obszary EAST-US i WEST-US należą do większego obszaru US

Następnie, napisz poniższe zapytania wykorzystując tabele EMP i DEPT

- Wyświetl nazwiska i etaty wszystkich pracowników pracujących w obszarze WEST-US
- Wyświetl sumaryczne pensje pracowników z podziałem na EAST-US i WEST-US

3. Zdefiniuj następującą hierarchię etatów



W tym przypadku bycie podklasą oznacza, że wystąpienia podklasy mogą się pojawić wszędzie tam, gdzie mogą się pojawić wystąpienia nadklasy (przykładowo, pracownik na etacie MANAGER może wykonać zadanie, które normalnie wykonują pracownicy na etatach ANALYST lub CLERK, natomiast pracownik na etacie ANALYST nie może wykonywać obowiązków pracownika na etacie SALESMAN).
Nie zapomnij o zamianie atrybutu **`Job`** w tabeli **`EMP`** na poprawny URI.

Następnie, napisz poniższe zapytania wykorzystując tabelę **`EMP`**

- Wyświetl nazwiska pracowników, którzy mogą zastąpić w pracy Adamsa.
- Dla każdego pracownika policz, ilu pracowników może go zastępować.
- Znajdź nazwiska pracowników, których może zastąpić mniej niż 3 innych pracowników.