The OWL2Bench SPARQL queries for OWL 2 EL, OWL 2 QL, OWL 2 RL, and OWL 2 DL are listed here.

Prefix: <http://benchmark/univ-bench-owl2>

SPARQL Queries for OWL 2 EL

1. SELECT DISTINCT ?x WHERE { ?x :hasSameHomeTownWith ?y }
   Description:
   Find out all the instances who have same home town with any other instance. Query involves Reflexive (hasSameHomeTownWith) and Transitive Object Property (hasSameHomeTownWith) and performs search across universities.

2. SELECT DISTINCT ?x WHERE { ?x rdf:type :Faculty. }
   Description:
   Find out all the Faculties. Query involves the constructs ObjectSomeValuesFrom and Intersection (Faculty ≡ (Employee ∩ ∃ teachesCourse.Course))

3. SELECT DISTINCT ?x WHERE { ?x :hasCollaborationWith ?y. ?y :isMemberOf :U0ClgOfFineArts15DeptOfDrama4. }
   Description:
   Find out all the instances who has collaboration with member (Student/Faculty) of U0ClgOfFineArts15DeptOfDrama4. Query involves Object Property Hierarchy (enrollIn ⊆ isStudentOf ⊆ isMemberOf, isFacultyOf ⊆ worksFor ⊆ isMemberOf) and performs search across universities.
4. SELECT DISTINCT ?x WHERE { ?x rdf:type :Student. ?x :isStudentOf ?y. ?y :isDepartmentOf ?z . ?z :hasCollegeDiscipline :Engineering. }
   Description:
   Find out all the Engineering Students. Performs search across universities.

SPARQL Queries for OWL 2 QL

1. SELECT DISTINCT ?x WHERE { ?x :hasSameHomeTownWith ?y }
   Description:
   Find out all the instances who has same home town with any other instance. Query involves Reflexive (hasSameHomeTownWith) and Symmetric Object Property (hasSameHomeTownWith) and performs search across universities.

2. SELECT DISTINCT ?x WHERE { ?x rdf:type :Faculty. }
   Description:
   Find out all the Faculties. Query involves the constructs ObjectSomeValuesFrom and Intersection (Faculty ⊆ (Employee ∩ ∃ teachesCourse.Course))

3. SELECT DISTINCT ?x WHERE { ?x :hasCollaborationWith ?y. ?y :isMemberOf :U0ClgOfEngineering15DeptOfIndustryEngineering4. }
   Description:
   Find out all the instances who has collaboration with member (Student/Faculty) of U0ClgOfFineArts15DeptOfDrama4. Query involves Object Property Hierarchy (enrollIn ⊆ isStudentOf ⊆ isMemberOf, isFacultyOf ⊆ worksFor ⊆ isMemberOf) and performs search across universities.

4. SELECT DISTINCT ?x WHERE { ?x rdf:type :Person . :U0 :hasAlumnus ?x. }
   Description:
   Find out all the alumni of :U0. Query involves Inverse Object Property (hasAlumnus) and performs search across universities.

5. SELECT DISTINCT ?x WHERE { ?a rdf:type :Organization. ?a :isAffiliatedOrganizationOf ?y. ?a :hasDepartment ?d. ?d :hasHead ?z. ?z :teachesCourse ?c. ?x :takesCourse ?c }
   Description:
   Find out all the students who take course taught by the head of the Department. Query involves Asymmetric Object Property (isAffiliatedOrganizationOf) and InverseObjectProperty(hasHead).

6. Select distinct ?x where {?x rdf:type benchmark:Student. ?x :isStudentOf ?y ?y :isDepartmentOf ?z .?z :hasCollegeDiscipline :Engineering.}
   Description:
   Find out all the Engineering Students. Query performs search across universities.

SPARQL Queries for OWL 2 RL

1. SELECT DISTINCT ?x WHERE { ?x :hasSameHomeTownWith ?y }
   Description:
   Find out all the instances who have same home town with any other instance. Query involves Transitive Object Property (hasSameHome- TownWith) and performs search across universities.

2. SELECT DISTINCT ?x WHERE { ?x rdf:type :Faculty. }
   Description:
   Find out all the Faculties. Involves the constructs ObjectSomeValuesFrom and Intersection (Faculty is a super class of Employee and ∃ teachesCourse.Course)

3. SELECT DISTINCT ?x WHERE ?x :hasCollaborationWith ?y. ?y :isStudentOf :U0.
   Description: Find out all the instances who has collaboration with member (Student/Faculty) of U0ClgOfFineArts15DeptOfDrama4. Query involves Object Property Hierarchy (enrollIn ⊆ isStudentOf) and performs search across universities.

4. SELECT DISTINCT ?x WHERE { ?x rdf:type :Person . :U0 :hasAlumnus ?x. }
   Description:
   Find out all the alumni of :U0. Query involves Inverse Object Property(hasAlumnus) and performs search across universities.

5. SELECT DISTINCT ?x WHERE { ?a rdf:type :Organization. ?a :isAffiliatedOrganizationOf ?y. ?y :hasDean ?z. ?z :teachesCourse ?c. ?x :takesCourse ?c }
   Description:
   Find out all the students who take course taught by the head of the Department. Query involves Asymmetric Object Property (isAffiliatedOrganizationOf) and InverseObjectProperty(hasHead).

6. SELECT DISTINCT ?x WHERE {?x rdf:type :WomanCollege}
   Description:
   Find out all the instances of WomanCollege. Query involves Complement and Universal Quantification (Womancollege ⊆ (College and ∀ hasStudent.(not Man)))

7. SELECT DISTINCT ?x WHERE {?x rdf:type :LeisureStudent}
   Description:
   Find out all the instances of LeisureStudent. Involves Maximum Cardinality (Leisure student is subclass of Stduent and ≥ 1.takesCourse.Course)

8. SELECT DISTINCT ?x WHERE {?x :isHeadOf ?y}
   Description:
   Find out the head of all the Organization. isHeadOf(Inverse functional Object Property)

9. SELECT DISTINCT ?x WHERE {?x :hasHead ?y}
   Description:
   Find out all the Organizations who has head. hasHead (Functional Object Property)

10. SELECT DISTINCT ?x WHERE {?x rdf:type :Employee . ?x :isMemberOf :U0}
    Description:
    Find out all the employees of :U0. Infers Property Chain (worksFor o isSubOrganizationOf ⊆ isMemberOf)

11. Select distinct ?x where ?x rdf:type :Student. ?x :isStudentOf ?y. ?y :isDepartmentOf ?z .?z :hasCollegeDiscipline :Engineering
    Description:
    Find out all the Engineering Students. Query performs search across universities.

SPARQL Queries for OWL 2 DL

1. SELECT DISTINCT ?x WHERE { ?x :hasSameHomeTownWith ?y }
   Description:
   Find out all the instances who has same home town with any other instance. Query involves Reflexive (hasSameHomeTownWith) and Transitive Object Property (hasSameHomeTownWith) and performs search across universities.

2. SELECT DISTINCT ?x WHERE { ?x rdf:type :Faculty. }
   Description:
   Find out all the Faculties. Involves the constructs ObjectSomeValuesFrom and Intersection (Faculty equiv Employee ∩ ∃ teachesCourse.Course)

3. SELECT DISTINCT ?x WHERE {?x :hasCollaborationWith ?y. ?y :isStudentOf :U0.}
   Description:
   Find out all the instances who has collaboration with member (Student/Faculty) of U0ClgOfFineArts15DeptOfDrama4. Query nvolves Object Property Hierarchy (enrollIn ⊆ isStudentOf) and performs search across universities.

4. SELECT DISTINCT ?x WHERE { ?x rdf:type :Person . :U0 :hasAlumnus ?x. }
   Description: Find out all the alumni of :U0. Query involves Inverse Object Property(hasAlumnus) and performs search across universities.

5. SELECT DISTINCT ?x WHERE { ?a rdf:type :Organization. ?a :isAffiliatedOrganizationOf ?y. ?y :hasHead ?z. ?z :teachesCourse ?c. ?x :takesCourse ?c }
   Description:
   Find out all the students who take courses taught by the head of the Department. Query involves Asymmetric Object Property (isAffiliatedOrganizationOf) and InverseObjectProperty(hasHead).

6. SELECT DISTINCT ?x WHERE { ?x rdf:type :WomanCollege}
   Description:
   Find out all the instances of WomanCollege. Query involves Complement and Universal Quantification (Womancollege ≡ College ∩ ∀ hasStudent.(not Man))

7. SELECT DISTINCT ?x WHERE {?x rdf:type :LeisureStudent}
   Description:
   Find out all the instances of LeisureStudent. Query involves
   Qualified Maximum Cardinality (Leisure student ≡ (Student ∩ ≥ 1.takesCourse. Course))

8. SELECT DISTINCT ?x WHERE {?x :isHeadOf ?y}
   Description:
   Find out the head of all the Organization. Query involve Inverse functional Object Property (isHeadOf)

9. SELECT DISTINCT ?x WHERE {?x :hasHead ?y}
   Description:
   Find out all the Organizations who has head. Query involves Functional Object Property(hasHead)

10. SELECT DISTINCT ?x WHERE {?x rdf:type :Employee . ?x :isMemberOf :U0}
    Description:
    Find out all the employees of :U0. Infers Property Chain (worksFor o isSubOrganizationOf = isMemberOf)

11. SELECT DISTINCT ?x WHERE {?x rdf:type :UGStudent}
    Description:
    Find out all the UG Students.Query involves QualifiedExactCardinality (UGStudent ≡ (Student ∩ =1.enrollFor.UGProgram).

12. SELECT DISTINCT ?x WHERE {?x rdf:type :PeopleWithManyHobbies}
    Description:
    Find out all the instances of people with atleast 3 hobbies. Query involves Qualified Minimum Cardinality (PeopleWithManyHobbies ≡ (Person ∩ ≤ 3.likes.Interests)

13. Select distinct ?x where {?x rdf:type :Student. ?x :isStudentOf ?y. ?y :isDepartmentOf ?z .?z :hasCollegeDiscipline :Engineering}
    Description:
    Find out all the Engineering Students. Query performs search across universities.