(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2022 년 7 월 7 일 (07.07.2022)

WIPO | PCT

(10) 국제공개번호
WO 2022/145503 A1

(51) 국제특허분류:
G06F 16/332 (2019.01) *G06F 9/54* (2006.01)
G06F 16/36 (2019.01) *G06F 9/451* (2018.01)

(21) 국제출원번호: PCT/KR2020/019262

(22) 국제출원일: 2020 년 12 월 31 일 (31.12.2020)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
10-2020-0187735 2020 년 12 월 30 일 (30.12.2020) KR

(71) 출원인: 한국전자기술연구원 (**KOREA ELECTRONICS TECHNOLOGY INSTITUTE**) [KR/KR]; 13509 경기도 성남시 분당구 새나리로 25, Gyeonggi-do (KR).

(72) 발명자: **수닐쿠마르 (SUNIL, Kumar)**; 17035 경기도 용인시 처인구 모현읍 백옥대로2404번길 5-14, 202호, Gyeonggi-do (KR). **이석준 (LEE, Seok Jun)**; 16821 경기도 용인시 수지구 수풍로 89, 109동 1402호, Gyeonggi-do (KR).

(74) 대리인: **남충우 (NAM, Choong Woo)**; 06226 서울시 강남구 언주로 321, 7층, Seoul (KR).

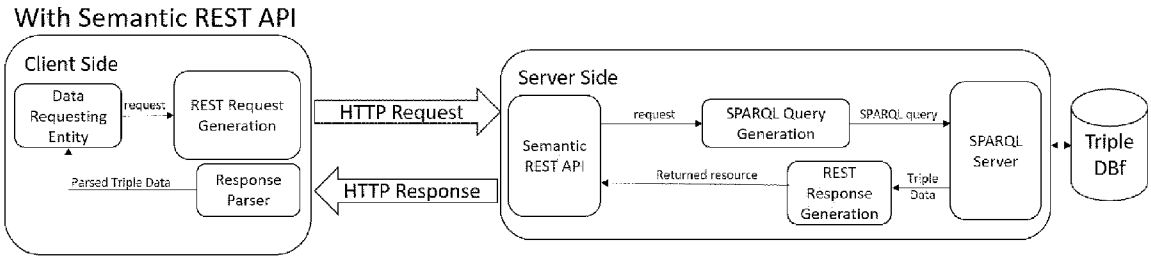
(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

공개:
— 국제조사보고서와 함께 (조약 제21조(3))

(54) Title: METHOD FOR PROVIDING SEMANTIC REST API

(54) 발명의 명칭: 시맨틱 레스트 에이피아이 제공 방법



(57) Abstract: A method for providing a semantic REST API is provided. The method for providing a semantic REST API according to an embodiment of the present invention: when receiving a semantic REST API request from a client, extracts parameters by parsing the received semantic REST API request; generates an SPARQL query by using the extracted parameters; and transmits the generated SPARQL query to a triple DB. This allows a non-semantic application to easily query semantic data.

(57) 요약서: 시맨틱 REST API 제공 방법이 제공된다. 본 발명의 실시예에 따른 본 발명의 일 실시예에 따른 시맨틱 REST API 방법은, 클라이언트로부터 시맨틱 REST API 요청을 수신하면, 수신한 시맨틱 REST API 요청을 파싱하여 파라미터들을 추출하고, 추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성하며, 생성한 SPARQL 쿼리를 트리플 DB에 전달한다. 이에 의해, 비-시맨틱 애플리케이션이 손쉽게 시맨틱 데이터를 쿼리할 수 있도록 하여 준다.

WO 2022/145503 A1

명세서

발명의 명칭: 시맨틱 레스트 에이피아이 제공 방법

기술분야

[1] 본 발명은 플랫폼 인터워킹 기술에 관한 것으로, 더욱 상세하게는 시맨틱 환경과 비-시맨틱 환경 간의 인터워킹을 위한 방법에 관한 것이다.

배경기술

[2] 시맨틱 기술은 데이터에 의미를 부여하고 데이터 간의 관계를 형성함으로써 모든 데이터들이 유기적으로 연결될 수 있게 한다.

[3] 도 1은 기존의 REST API를 적용한 시스템을 도시한 도면이다. 시맨틱 데이터를 검색할 수 있는 쿼리 언어로 SPARQL이 사용되고 있는데, 도 1에 도시된 바와 같이 복잡한 SPARQL 쿼리는 비-시맨틱 애플리케이션이 데이터를 쿼리하는 것을 어렵게 하고 있다.

[4] 이에 따라 비-시맨틱 애플리케이션이 손쉽게 시맨틱 데이터를 쿼리할 수 있도록 하기 위한 방안이 필요하다.

발명의 상세한 설명

기술적 과제

[5] 본 발명은 상기와 같은 문제점을 해결하기 위하여 안출된 것으로서, 본 발명의 목적은, 비-시맨틱 애플리케이션이 손쉽게 시맨틱 데이터를 쿼리할 수 있도록 하기 위한 방안으로, 시맨틱 REST API를 제공함에 있다.

과제 해결 수단

[6] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른, 시맨틱 REST API 방법은, 클라이언트로부터 시맨틱 REST API 요청을 수신하는 단계; 수신한 시맨틱 REST API 요청을 파싱하여, 파라미터들을 추출하는 단계; 추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성하는 단계; 생성한 SPARQL 쿼리를 트리플 DB에 전달하는 단계;를 포함한다.

[7] 시맨틱 REST API 요청은, 그래프 URI 목록을 획득하기 위한 요청, 특정 타입의 엔티티의 URI 목록을 획득하기 위한 요청, 특정 엔티티의 URI 정보를 획득하기 위한 요청, Class/Individual의 클래스 계층구조를 획득하기 위한 요청, 특정 그래프 URI의 트리플 세트를 획득하기 위한 요청, 특정 타입의 Individual의 URI 목록을 획득하기 위한 요청 중 적어도 하나를 포함할 수 있다.

[8] 파라미터들은, 검색 키워드 목록, 그래프 타입, Prefixes 형식, 엔티티 타입 목록, 엔티티 타입 목록, 엔티티 URI/IRI, 응답 형식, 클래스의 Id, 응답 제한 중 적어도 하나를 포함할 수 있다.

[9] SPARQL 쿼리를 생성 단계는, 추출한 파라미터들을 이용하여, 그래프 및 타입, SPARQL 쿼리 변수, SPARQL 순회 및 필터 조건을 SPARQL 쿼리에 수록할 수 있다.

[10] SPARQL 순회 및 필터 조건은, 검색할 데이터의 패턴을 정의하는 SPARQL WHERE 절 및 SPARQL 필터 조건을 포함할 수 있다.

[11] 본 발명의 실시예에 따른 시맨틱 REST API 방법은, 트리플 DB로부터 SPARQL 응답을 수신하는 단계; 수신한 SPARQL 응답을 파싱하여, 쿼리 변수의 값들을 추출하는 단계; 추출한 값들을 이용하여, REST 응답을 생성하는 단계; 및 생성한 REST 응답을 클라이언트에 반환하는 단계;를 더 포함할 수 있다.

[12] REST 응답 생성단계는, URI/IRI 목록 및 트리플 목록 중 적어도 하나를 추출할 수 있다.

[13] 한편, 본 발명의 다른 실시예에 따른, 시맨틱 검색 서버는, 클라이언트로부터 시맨틱 REST API 요청을 수신하는 인터페이스; 수신한 시맨틱 REST API 요청을 파싱하여 파라미터들을 추출하고, 추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성하는 SPARQL 쿼리 생성부; 및 생성한 SPARQL 쿼리를 트리플 DB에 전달하는 SPARQL 서버;를 포함한다.

발명의 효과

[14] 이상 설명한 바와 같이, 본 발명의 실시예들에 따르면, 시맨틱 REST API를 통해, 비-시맨틱 애플리케이션이 손쉽게 시맨틱 데이터를 쿼리할 수 있도록 하여 준다.

도면의 간단한 설명

[15] 도 1. 기존의 REST API를 적용한 시스템.

[16] 도 2. 본 발명의 실시예에 따른 시맨틱 REST API를 적용한 시스템.

[17] 도 3. 시맨틱 데이터 및 자원 구조 개요

[18] 도 4. 의미 론적 REST API : 기능

[19] 도 5. 엔티티 URI/IRI

[20] 도 6. 시맨틱 데이터 및 자원 구조 개요

[21] 도 7-10. 시맨틱 데이터 : 엔티티 유형 및 정보

[22] 도 11. 시맨틱 API : 단순 및 일반 형식 유형

[23] 도 12. SPARQL 쿼리 응답

[24] 도 13. 시맨틱 REST API : 쿼리 매개 변수

[25] 도 14. 그래프 목록(유형: 온톨로지)

[26] 도 15. 그래프 목록(유형 : 인스턴스)

[27] 도 16. 엔티티 목록

[28] 도 17-18. 엔티티 정보

[29] 도 19. 클래스 계층

[30] 도 20-21. 그래프 정보

[31] 도 22. Individuals 목록

발명의 실시를 위한 최선의 형태

[32] 이하에서는 도면을 참조하여 본 발명을 보다 상세하게 설명한다.

[33] 도 2는 본 발명의 실시예에 따른 시맨틱 REST API를 적용한 시스템을 도시한 도면이다. 도시된 시스템은 클라이언트(100), 시맨틱 검색 서버(200), 트리플 DB(300)를 포함하여 구성된다.

[34] 클라이언트(100)는 데이터 요청 엔티니(110)의 요청을 REST 요청 생성부(120)가 시맨틱 REST API(210)를 제공하는 시맨틱 검색 서버(200)로 전송한다.

[35] 시맨틱 검색 서버(200)의 SPARQL 쿼리 생성부(220)는 시맨틱 REST API(210)를 통해 클라이언트로부터 수신한 시맨틱 REST API 요청을 파싱하여 파라미터들을 추출하고, 추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성한다.

[36] 그러면 SPARQL 서버(230)는 SPARQL 쿼리 생성부(220)에 의해 생성된 SPARQL 쿼리를 트리플 DB(300)로 전달하여, 트리플 DB(300)로부터 SPARQL 응답을 수신한다.

[37] REST 응답 생성부(240)는 SPARQL 서버(230)가 수신한 SPARQL 응답을 파싱하여 쿼리 변수의 값들을 추출하고, 추출한 값들로 시맨틱 REST API 응답을 생성하여 클라이언트(100)의 응답 파서(130)로 반환한다.

[38] 본 발명의 실시예에서는 REST 지원 시스템에서 시맨틱 데이터를 활용할 수 있도록 시맨틱 데이터를 리소스 형식으로 표현하고, REST API에서 복잡한 SPARQL 쿼리를 어느 정도 유형화하여, 비-시맨틱 애플리케이션이 데이터를 쿼리할 수 있도록 한다.

[39] API에 전체 SPARQL 쿼리를 포함하는 대신, 리소스로 반환되는 각 쿼리 응답에 대해 정해진 형식을 갖는 GET, PUT, POST, DELETE 요청으로 통합된다.

[40] 도 3에는 시맨틱 데이터의 구조와 REST 리소스 구조를 나타내었다. 도 4는 본 발명의 실시예에서 제공하는 시맨틱 REST API의 기능(함수)들을 나열한 표이다. 도시된 바와 같이, 시맨틱 REST API 기능에는, GET Graph List, Entity List, Entity Info, Class-Hierarchy, Graph Info, Individual List가 있으며, 이들에 대한 상세한 설명은 아래와 같다.

[41] GET Graph List는 그래프 URI 목록을 획득하기 위한 기능이다. 모든 트리플 세트는 고유한 그래프 URI를 갖는 그래프에 저장되고, 온톨로지는 트리플 DB에 그래프로도 저장될 수 있다. GET Graph List로 해당 그래프 URI 목록을 획득한다.

[42] Entity List는 엔티티 타입이 Class, ObjectProperty , DataProperty , NamedIndividual ... 인 엔티티의 URI 목록을 획득하기 위한 기능이다.

[43] Entity Info는 주어진 엔티티(Class, NamedIndividual, Object Property, Data Property)의 URI 정보를 획득한다. 즉 타입과 속성들(domains, ranges, subsumption, comment, ...)에 대한 세부 정보를 제공하여 준다.

[44] Class-Hierarchy는 자식 클래스에서 부모 클래스까지, "n-level"까지 Class/Individual의 클래스 계층구조를 획득한다.

[45] Graph Info는 주어진 그래프 URI의 모든 트리플 세트를 획득한다.

[46] Individual List는 주어진 타입의 모든 Individual의 URI 목록을 획득한다.

[47]

[48] 이하에서 시맨틱 API 요청을 처리하는 과정에 대해 상세히 설명한다.

[49]

[50] 1. 요청 파라미터 파싱

[51] 각 API 요청은 자체 요청 파라미터로써, SPARQL 쿼리를 생성하는데 추가로 사용되는 조건을 추출하기 위해 파싱된다.

[52]

[53] (1) 검색 키워드 파싱

[54] 검색 키워드는 요구 데이터의 일부로써, 클라이언트가 제공한다. 이 경우 데이터는 URI/IRI로 구성되며, 데이터를 구성하는 그래프에도 URI/IRI가 식별자로 포함된다. 검색 키워드는 URI/IRI의 일부이며, (다른 검색 조건과 함께) 일치하는 경우 해당 데이터(그래프/엔티티 URI/IRI)가 후보 응답으로 고려된다.

[55]

[56] (2) URI 파싱

[57] 시맨틱 데이터의 URI/IRI는 데이터의 특정 엔티티에 대한 식별자이다. 또한, 이들은 사전(vocabulary : 시맨틱 온톨로지)인 스키마를 정의하는 데에도 사용된다.

[58] 요청과 응답을 쉽고 간결하게 만들기 위해, 시맨틱 데이터에서 엔티티 ID의 네임 스페이스를 나타내는 사전에 정의된 static prefixes가 사용된다. 도 5에 나타낸 바와 같이, 요청에서 긴 URI 대신, 네임 스페이스는 사전에 정의된 static label, 콜론 ":", Postfixes 순으로 대체된다.

[59] 도 8에 나타난 바와 같이, 이들은 모두 함께 엔티티 URI/IRI를 형성하며, 이는 API에서 유효한 URI/IRI로 식별되고 SPARQL 쿼리에 사용된다.

[60]

[61] (3) 제한, 형식 및 응답 타입 등과 같은 기타 파라미터 파싱

[62] 1) 그래프 타입 또는 ID

[63] 일반적으로 데이터베이스에는 두 가지 타입의 그래프가 있다. 하나는 사전/온톨로지인 스키마를 포함하는 그래프이고, 다른 하나는 그러한 온톨로지를 기반으로 하는 인스턴스를 포함하는 그래프이다. 요청 파라미터는 검색해야 하는 그래프 타입을 지정한다.

[64] Id(URI/IRI)의 경우, SPARQL 쿼리에서 그래프에 포함된 모든 데이터를 획득하는데 사용된다.

[65]

[66] 2) 엔티티 타입 또는 ID

[67] 엔티티는 데이터에서 엔티티를 나타내는 시맨틱 API에 의해 정의되는 용어이다. Class, Individual, ObjectProperty 또는 DatatypeProperty가 될 수 있다. 엔티티에 대한 정보는 요청에 타입 또는 ID(URI/IRI)를 지정하여 요청할 수 있다.

타입은 엔티티(URI/IRI)의 목록을 획득하기 위해 지정되는 반면, Id(URI/IRI)는 해당 엔티티의 관련 정보를 가져 오기 위해 지정된다. 도 6 내지 도 10에서 엔티티 타입 및 정보들을 예시하였다.

[68]

[69] 3) 클래스 타입

[70] 클래스(owl : Class)는 엔티티 타입(owl : NamedIndividual)이 정의되는 엔티티를 기반으로 사전/온톨로지에 정의된 특정 엔티티이다. Individuals의 특성을 정의하는 타입 정의가 있다. Individuals(URI/IRI만 해당)의 목록을 검색하기 위해 타입이 지정된다. 값(valuer)은 클래스의 Id(URI / IRI)이다.

[71]

[72] 4) Prefix/응답 형식

[73] 응답에는 심플 형식과 노멀 형식 두 가지가 있는데, 심플 형식은 값들이 직접 제공되므로 응답 크기가 작다. 반면 노멀 형식에서는 응답에서 의도된 값들에 대해 사전 정의된 특정 키가 있다. 키를 사용하면 응답이 더 커지지만 더 정교해진다. 응답 형식의 경우, 이 옵션은 API 기능 중 엔티티 정보에만 제공된다. 도 11에 심플 형식과 노멀 형식 두 가지를 예시하였다.

[74]

[75] 5) 응답 제한(Response Limit)

[76] 검색된 시맨틱 데이터가 클 수 있으므로, 더 빠른 검색 결과를 제공하기 위해, (SPARQL 쿼리에서) 결과 수를 제한하기 위해 이 파라미터가 제공된다. 제한이 지정되지 않은 경우, 디폴트 값으로 설정된다.

[77]

[78] 2. SPARQL 쿼리 생성

[79] SPARQL 쿼리는 트리플 데이터베이스(TDB)와 인터랙션하는 방법이다. 따라서 API 요청은 필요한 데이터를 검색하기 위해 SPARQL 쿼리로 변환된다.

[80]

[81] (1) 그래프 및 타입

[82] 파싱된 그래프 Id 또는 타입은 API의 SPARQL 쿼리 문자열에서, 그래프 목록 또는 지정된 그래프에 포함된 데이터를 검색하는데 사용된다. 도 13과 도 19에 예시되어 있다.

[83]

[84] (2) SPARQL 쿼리 변수

[85] SPARQL 쿼리 변수에 대해서는 상세히 후술한다.

[86]

[87] (3) SPARQL 순회 및 필터 조건

[88] 1) SPARQL WHERE 절

[89] 검색할 데이터의 패턴을 정의하는데 사용된다. SPARQL 변수와 값으로 구성된다.

[90]

[91] 2) SPARQL 필터 조건

[92] 일치시킬 특정 조건을 정의하는데 사용되며, 패턴을 사용하여 정의할 수 없다(예 : 값 비교, 산술 연산 등). 대부분의 경우, API에서 요청 파라미터에서 추출한 검색 키워드를 일치시키는데 사용된다. 검색 키워드는 SPARQL 쿼리의 필터 조건에 추가된다. 따라서 지정된 키워드를 포함하는 응답만 반환된다.

[93]

[94] (4) SPARQL 쿼리 응답 : SPARQL 쿼리는 쿼리에서 요청되는 쿼리 변수로 응답을 구성한다. SPARQL 응답에서 예상되는 쿼리 변수는 "SELECT" 문에 지정된다. "WHERE" 절의 쿼리 변수는 데이터베이스에서 검색 할 패턴을 지정힌다. "SELECT" 문에 지정된 쿼리 변수는 필요한 경우 "WHERE" 절에서도 사용할 수 있다.

[95] 반환된 응답은 일반적으로 쿼리 솔루션을 나타내는 오브젝트들로 구성되며 각 솔루션에는 쿼리 변수의 해당 값이 포함된다. 응답은 다른 직렬화를 가질 수 있으므로, 표현이 다를 수 있지만 동일한 개념을 갖는다.

[96]

[97] (5) Prefixes

[98] Prefixes는 시맨틱 API에 의해 별도로 유지된다. 응답이 생성될 때 마다, Prefixes의 각 해당 네임 스페이스가 API에 의해 저장된다. API는 각 URI/IRI를 파싱하고 응답에 관련된 네임 스페이스를 기반으로 Prefixes를 추가한다.

[99]

[100] 3. SPARQL 응답 파싱

[101] 쿼리 변수로 구성되어 캡슐화된 응답을 추출한다. SPARQL 쿼리 응답을 수신한 후, 요청된 쿼리 변수("SELECT" 문에 지정됨)의 각 값은 반복적으로 (하나씩) 순회되고, API 응답에 추가된다.

[102] 응답 순회에서, 값은 전체 URI/IRI(엔티티 ID의 네임 스페이스 + Postfixes)를 포함한다. 따라서 "#" 앞의 URI/IRI 부분은 네임 스페이스로 식별되고 추출된다. 추출된 네임 스페이스는 API의 Prefixes 저장소에서 검색되며, 여기서 네임 스페이스와 각각의 Prefixes는 저장된다.

[103] 저장소에 Prefixes가 있는 경우, API 응답에서 사용하도록 선택된다. 그렇지 않은 경우 순회에서 수집된 정확한 전체 URI/IRI가 사용된다.

[104]

[105] 4. JSON 응답 생성

[106] JSON 응답은 API가 클라이언트에 반환하는 응답이다. SPARQL 응답은 응답을 간결하게 만들기 위해 Prefixes 추가와 함께 JSON 응답으로 변환된다.

[107]

[108] (1) Prefixes 생성

[109] 응답 순회에서 값은 전체 URI/IRI(엔티티 ID의 네임 스페이스 + Postfixes)를

갖는다. 따라서 네임 스페이스가 추출되고 API 응답에서 적절한 Prefixe 레이블로 대체된다. 또한 Prefixe 레이블과 해당 네임 스페이스도 API 응답의 "prefix-list" 오브젝트 아래에 추가된다. API 응답은 API 응답에 포함된 "prefix-list" 오브젝트의 Prefixes만 포함한다.

[110]

[111] (2) Prefixes 사용 응답 생성

[112] API에서 반환 할 수있는 응답 형식에는 두 가지 타입이 있다. 형식에 관계없이 URI/IRI는 항상 Prefixes 레이블들을 포함한다(API에서 인식하는 경우, API의 Prefixes 저장소에서 사용할 수 있음).

[113]

[114] 1) URI/IRI 목록

[115] 이 형식은 API 기능 중 GET Graph List, Entity List, Class-Hierarchy, Individual List에서 사용된다. 여기서 응답은 URI/IRI 목록을 포함하는 JSON 배열이다.

[116] 이 경우 SPARQL 응답에는 쿼리 변수가 하나만 있으므로, API 응답에서 JSON 배열로 변환된다. 도 14, 도 15, 도 16, 도 19 및 도 22에 관련 예제를 제시하였다.

[117]

[118] 2) 트리플 목록

[119] 이 형식은 API 기능 중 Entity Info 및 Graph Info에서 사용된다. 여기서 응답은 JSON 오브젝트 목록을 포함하는 JSON 배열이며, 각 JSON 객체에는 하나의 트리플 데이터(subject, predicate and object)가 포함된다.

[120] Entity Info에 대해서는 전술한 바 있고, 도 11에 심플 응답 버전이 있다. 이 경우, SPARQL 응답에는 subject, predicate and object를 나타내는 s, p 및 o의 세 가지 쿼리 변수가 있다. 이러한 변수들은 함께 트리플을 정의하며, 이는 위에 정의된 형식으로 변환된다. 도 11, 도 12, 도 17, 도 18, 도 20, 도 21에 관련 예제를 제시하였다.

[121]

[122] 5. 시맨틱 REST API : 쿼리 파라미터

[123] 지금까지, 시맨틱 API 요청을 처리하는 과정에 대해 상세히 설명하였다. 도 13에는 시맨틱 REST API의 쿼리 파라미터들을 정리한 표이다. 쿼리 파라미터들에는 검색 키워드 목록(Search Keyword List), 그래프 타입(Graph Type), Prefixes 형식(Prefix format), 엔티티 타입 목록(Entity Type List), 엔티티 URI/IRI(Entity URI/IRI), 응답 형식(Response Format), ClassId(ClassId), 응답 제한(Response Limit)이 포함된다.

[124] 검색 키워드 목록은 검색할 리소스 정보에 따라, 일치하는 키워드로, 키워드를 제공할 수 있다. 그래프 목록와 엔티티 목록에 사용된다.

[125] 그래프 타입(Graph Type)은 온톨로지 타입과 인스턴스 타입의 그래프로 구분된다. 온톨로지 타입 그래프는 모든 온톨로지 모델을 저장하는, 반면 인스턴스 타입 그래프는 온톨로지 모델을 기반으로 assertions(실제 데이터)을 저장한다.

그래프 목록에 사용된다.

[126] Prefixes는 모든 온톨로지의 전체 또는 일부를 정의하는 모든 네임 스페이스의 식별자이다. 엔티티 URI/IRI의 일부로 사용되며, 읽기 쉽고 간결한 형식을 제공하기 위해 전체 URI/IRI를 단축하는데 사용된다. Prefixes 형식에는 심플 타입과 노멀 타입이 있다.

[127] 엔티티 타입 목록 : 엔티티 타입은 다음 값을 가질 수 있다 : lass, ObjectProperty, DataProperty, NamedIndividual

[128] 또한 온톨로지에서 정의된 클래스, 객체 속성 또는 데이터 속성 URI/IRI를 포함할 수도 있다. 엔티티 목록에 사용된다.

[129] 엔티티 URI/IRI는 온톨로지에서 정의된 클래스, 객체 속성 또는 데이터 속성 URI/IRI일 수 있다. 엔티티 정보에 사용된다.

[130] 응답 형식 : 응답은 심플 형식과 노멀 형식이 있다. 심플 형식은 간결한 반면 노멀 형식은 상세하며 subject, predicate and object에 따라 각 트리플을 설명한다. 엔티티 정보에 사용된다.

[131] ClassId는 클래스 타입의 엔티티이다. 클래스 계층과 Individual 목록에 사용된다.

[132] 응답 제한은 반환할 그래프, 엔터티 또는 트리플의 수를 지정하는데 사용된다. 서버 및 데이터베이스 사양에 따라 디폴트 값을 가질 수 있다.

[133]

[134] 6. 변형예

[135] 지금까지, 시맨틱 REST API 제공 방법에 대해 바람직한 실시예를 들어 상세히 설명하였다.

[136] 한편, 본 실시예에 따른 장치와 방법의 기능을 수행하게 하는 컴퓨터 프로그램을 수록한 컴퓨터로 읽을 수 있는 기록매체에도 본 발명의 기술적 사상이 적용될 수 있음은 물론이다. 또한, 본 발명의 다양한 실시예에 따른 기술적 사상은 컴퓨터로 읽을 수 있는 기록매체에 기록된 컴퓨터로 읽을 수 있는 코드 형태로 구현될 수도 있다. 컴퓨터로 읽을 수 있는 기록매체는 컴퓨터에 의해 읽을 수 있고 데이터를 저장할 수 있는 어떤 데이터 저장 장치이더라도 가능하다. 예를 들어, 컴퓨터로 읽을 수 있는 기록매체는 ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광디스크, 하드 디스크 드라이브, 등이 될 수 있음은 물론이다. 또한, 컴퓨터로 읽을 수 있는 기록매체에 저장된 컴퓨터로 읽을 수 있는 코드 또는 프로그램은 컴퓨터간에 연결된 네트워크를 통해 전송될 수도 있다.

[137] 또한, 이상에서는 본 발명의 바람직한 실시예에 대하여 도시하고 설명하였지만, 본 발명은 상술한 특정의 실시예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진자에 의해 다양한 변형실시가 가능한 것은 물론이고, 이러한 변형실시들은 본 발명의 기술적 사상이나 전망으로부터

개별적으로 이해되어져서는 안될 것이다.

청구범위

[청구항 1] 클라이언트로부터 시맨틱 REST API 요청을 수신하는 단계;
수신한 시맨틱 REST API 요청을 파싱하여, 파라미터들을 추출하는 단계;
추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성하는 단계;
생성한 SPARQL 쿼리를 트리플 DB에 전달하는 단계;를 포함하는 것을 특징으로 하는 시맨틱 REST API 방법.

[청구항 2] 청구항 1에 있어서,
시맨틱 REST API 요청은,
그래프 URI 목록을 획득하기 위한 요청, 특정 타입의 엔티티의 URI 목록을 획득하기 위한 요청, 특정 엔티티의 URI 정보를 획득하기 위한 요청, Class/Individual의 클래스 계층구조를 획득하기 위한 요청, 특정 그래프 URI의 트리플 세트를 획득하기 위한 요청, 특정 타입의 Individual의 URI 목록을 획득하기 위한 요청 중 적어도 하나를 포함하는 것을 특징으로 하는 시맨틱 REST API 방법.

[청구항 3] 청구항 1에 있어서,
파라미터들은,
검색 키워드 목록, 그래프 타입, Prefixes 형식, 엔티티 타입 목록, 엔티티 타입 목록, 엔티티 URI/IRI, 응답 형식, 클래스의 Id, 응답 제한 중 적어도 하나를 포함하는 것을 특징으로 하는 시맨틱 REST API 방법.

[청구항 4] 청구항 3에 있어서,
SPARQL 쿼리를 생성 단계는,
추출한 파라미터들을 이용하여, 그래프 및 타입, SPARQL 쿼리 변수, SPARQL 순회 및 필터 조건을 SPARQL 쿼리에 수록하는 것을 특징으로 하는 시맨틱 REST API 방법.

[청구항 5] 청구항 4에 있어서,
SPARQL 순회 및 필터 조건은,
검색할 데이터의 패턴을 정의하는 SPARQL WHERE 절 및 SPARQL 필터 조건을 포함하는 것을 특징으로 하는 시맨틱 REST API 방법.

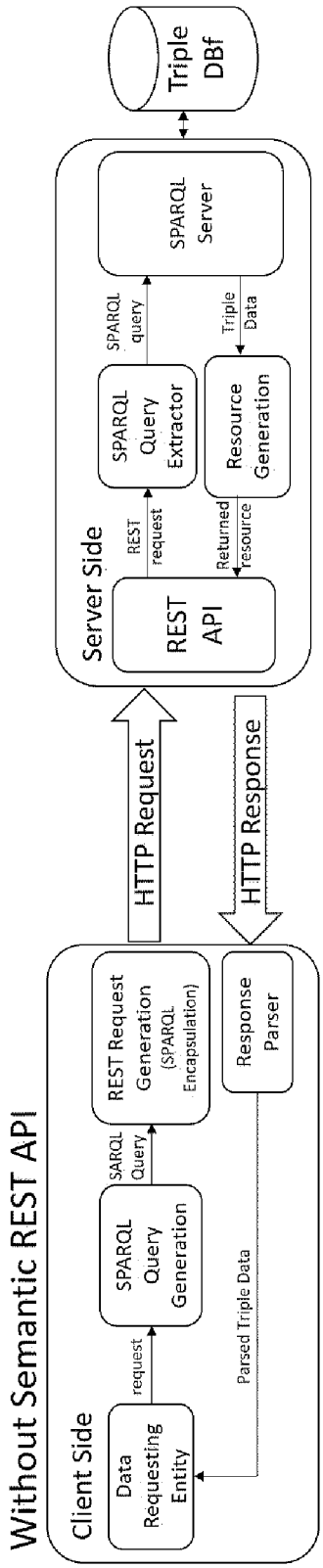
[청구항 6] 청구항 1에 있어서,
트리플 DB로부터 SPARQL 응답을 수신하는 단계;
수신한 SPARQL 응답을 파싱하여, 쿼리 변수의 값들을 추출하는 단계;
추출한 값들을 이용하여, REST 응답을 생성하는 단계; 및
생성한 REST 응답을 클라이언트에 반환하는 단계;를 포함하는 것을 특징으로 하는 시맨틱 REST API 방법.

[청구항 7] 청구항 6에 있어서,
REST 응답 생성단계는,
URI/IRI 목록 및 트리플 목록 중 적어도 하나를 추출하는 것을 특징으로

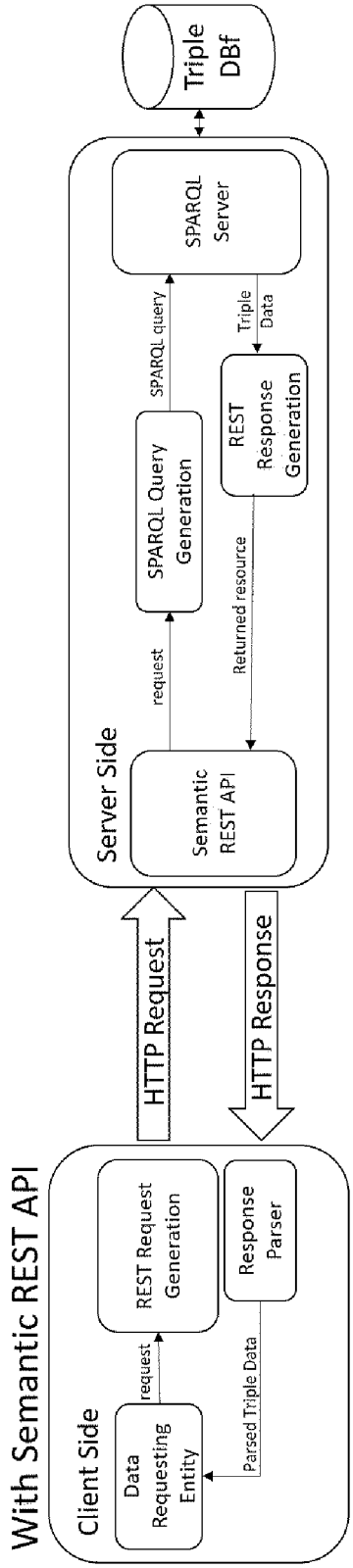
하는 시맨틱 REST API 방법.

[청구항 8] 클라이언트로부터 시맨틱 REST API 요청을 수신하는 인터페이스;
수신한 시맨틱 REST API 요청을 파싱하여 파라미터들을 추출하고,
추출한 파라미터들을 이용하여 SPARQL 쿼리를 생성하는 SPARQL 쿼리 생성부; 및
생성한 SPARQL 쿼리를 트리플 DB에 전달하는 SPARQL 서버;를
포함하는 것을 특징으로 하는 시맨틱 검색 서버.

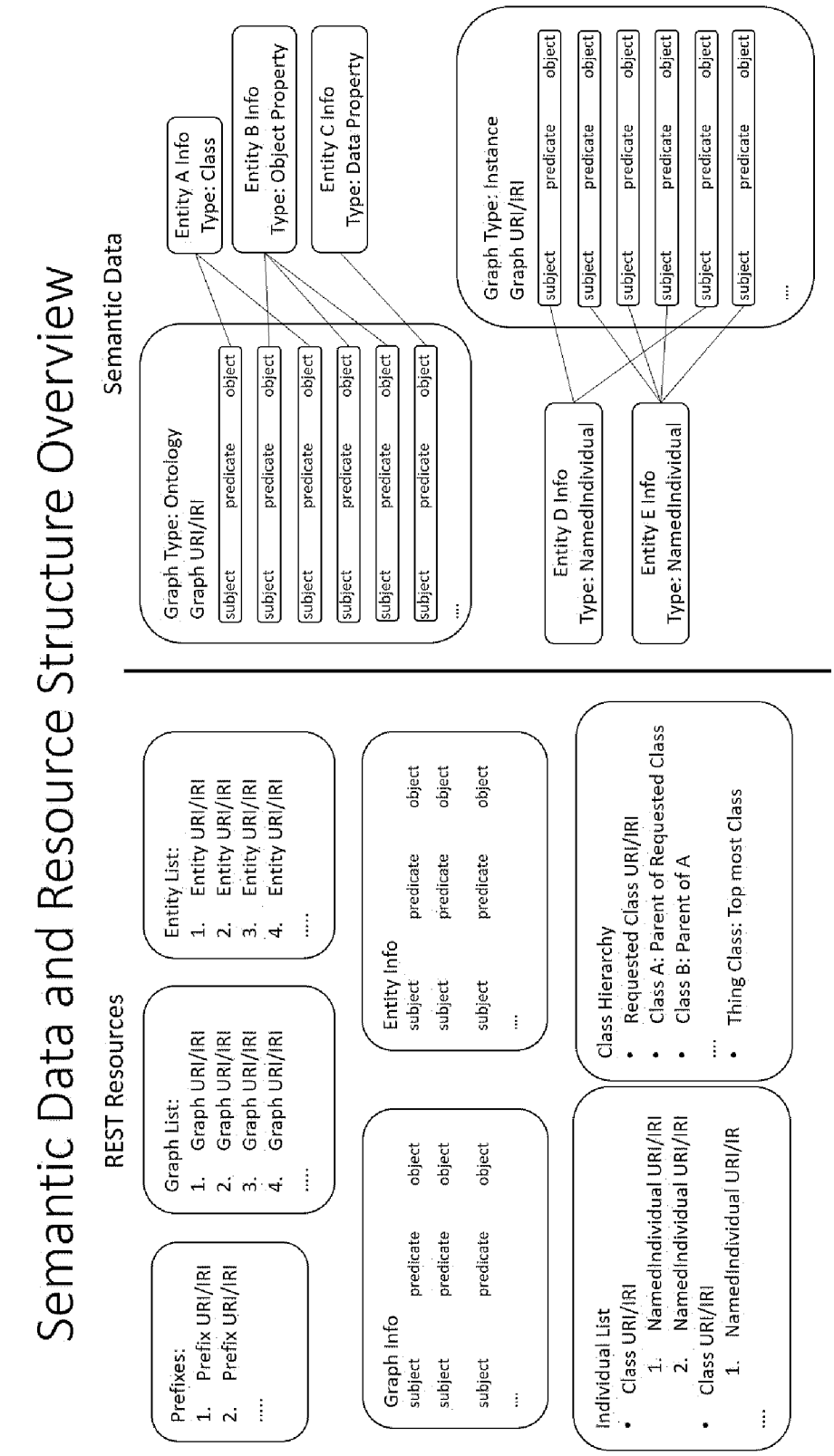
[도1]


Without Semantic REST API
Client Side
Data Requesting Entity
request
SPARQL Query Generation
SARQL Query
REST Request Generation (SPARQL Encapsulation)
Response Parser
Parsed Triple Data
HTTP Request
HTTP Response
Server Side
REST API
REST request
Returned resource
SPARQL Query Extractor
Resource Generation
SPARQL query
Triple Data
SPARQL Server
Triple DBf

[도2]


With Semantic REST API
Client Side
Data Requesting Entity
request
REST Request Generation
Parsed Triple Data
Response Parser
HTTP Request
HTTP Response
Server Side
request
Semantic REST API
SPARQL Query Generation
SPARQL query
Returned resource
REST Response Generation
Triple Data
SPARQL Server
Triple DBf

[도3]


Semantic Data and Resource Structure Overview
REST Resources
Semantic Data
Prefixes:
1. Prefix URI/IRI
2. Prefix URI/IRI
.....
Graph List:
1. Graph URI/IRI
2. Graph URI/IRI
3. Graph URI/IRI
4. Graph URI/IRI
.....
Entity List:
1. Entity URI/IRI
2. Entity URI/IRI
3. Entity URI/IRI
4. Entity URI/IRI
.....
Graph Info
subject predicate object
subject predicate object
subject predicate object
....
Entity Info
subject predicate object
subject predicate object
subject predicate object
....
Individual List
• Class URI/IRI
1. NamedIndividual URI/IRI
2. NamedIndividual URI/IRI
• Class URI/IRI
1. NamedIndividual URI/IR
....
Class Hierarchy
• Requested Class URI/IRI
• Class A: Parent of Requested Class
• Class B: Parent of A
....
• Thing Class: Top most Class
Graph Type: Ontology
Graph URI/IRI
subject predicate object
....
Entity A Info
Type: Class
Entity B Info
Type: Object Property
Entity C Info
Type: Data Property
Entity D Info
Type: NamedIndividual
Entity E Info
Type: NamedIndividual
Graph Type: Instance
Graph URI/IRI
subject predicate object
....

[도4]

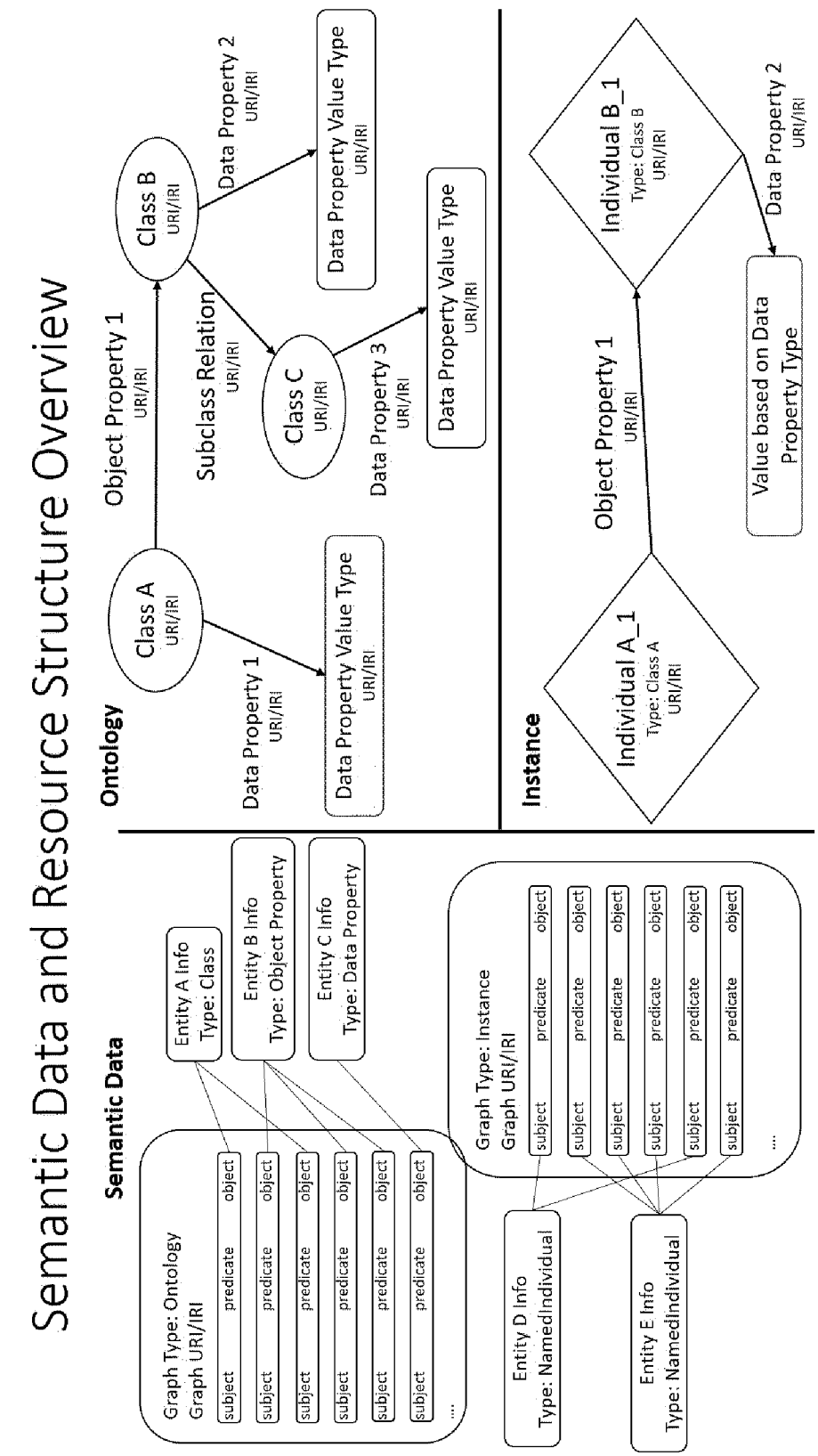
Semantic REST API: Functionalities

Functionalities		
#	Name	Description
1.	GET Graph List	Get the list of graphs URI. Every set of triples is stored in graphs, having unique graph URI. The ontologies are also stored as graphs in the TDB. This Request will get the list of those graphs URIs.
2.	Entity List	Get the list of entity URIs where an entity can be one of following type: {Class, ObjectProperty, DataProperty, NamedIndividual, ...}
3.	Entity Info	Get the info of the given entity URI {Class, NamedIndividual, Object Property, Data Property}. i.e. Provide details of its type and properties {domains, ranges, subsumption, comment, ... }
4.	Class - Hierarchy	Get the Class hierarchy of Class/Individual up to "n-level", from child class to parent class.
5.	Graph Info	Get the set of all triples of given graph URI.
6.	Individual List	Get the URI list of all the Individuals of the given type.

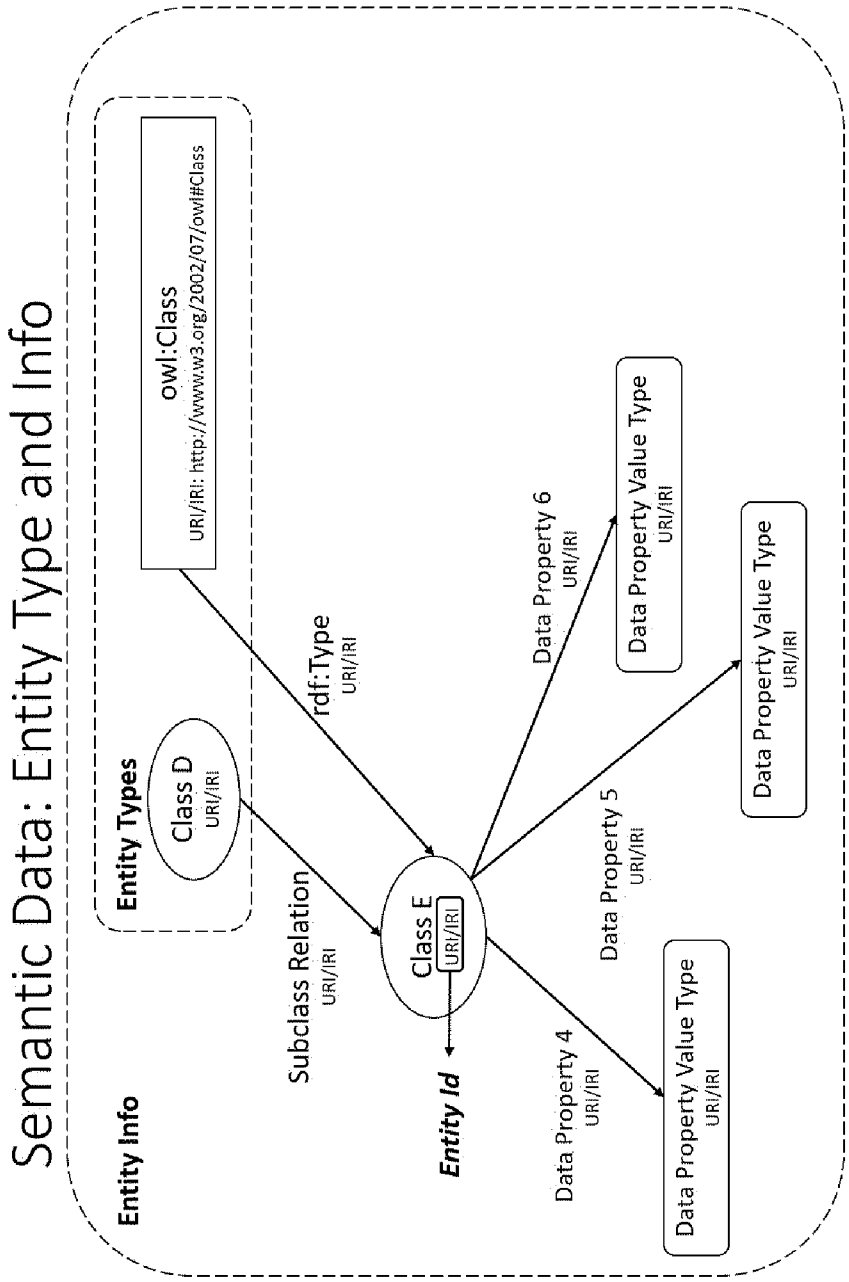
[도5]

	URI / IRI (Full Form)	URI / IRI (Full Form)
Formation	"Namespace" + "#" + "Postfix of entity Id"	"Prefix Label" + ":" + "Postfix of entity Id"
Example	http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	parking:ParkingSpot_spot_431

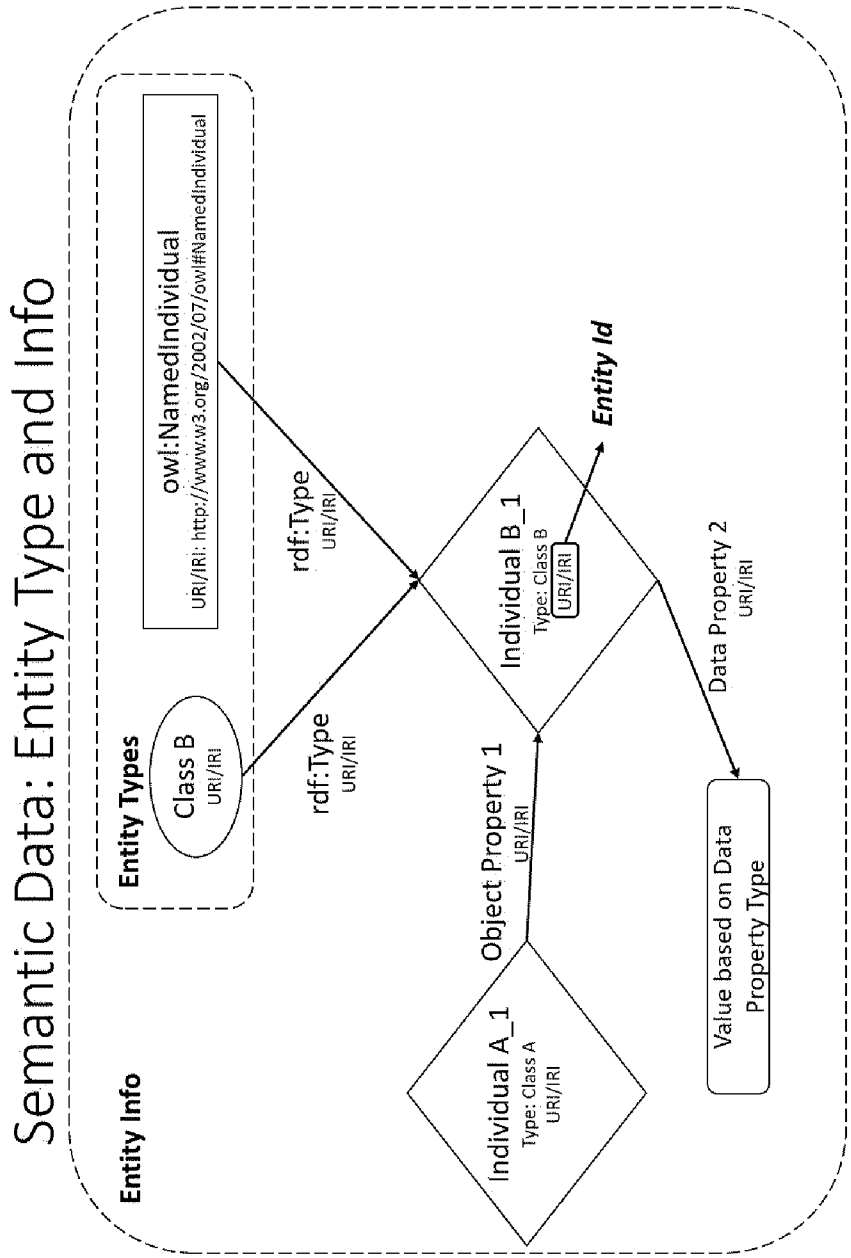
[도6]


Semantic Data and Resource Structure Overview
Ontology
Class A
URI/IRI
Object Property 1
URI/IRI
Class B
URI/IRI
Data Property 1
URI/IRI
Subclass Relation
URI/IRI
Data Property 2
URI/IRI
Class C
URI/IRI
Data Property 3
URI/IRI
Data Property Value Type
URI/IRI
Data Property Value Type
URI/IRI
Data Property Value Type
URI/IRI
Instance
Individual A_1
Type: Class A
URI/IRI
Object Property 1
URI/IRI
Individual B_1
Type: Class B
URI/IRI
Data Property 2
URI/IRI
Value based on Data Property Type
Semantic Data
Entity A Info
Type: Class
Entity B Info
Type: Object Property
Entity C Info
Type: Data Property
Graph Type: Ontology
Graph URI/IRI
subject predicate object
Graph Type: Instance
Graph URI/IRI
Entity D Info
Type: NamedIndividual
Entity E Info
Type: NamedIndividual

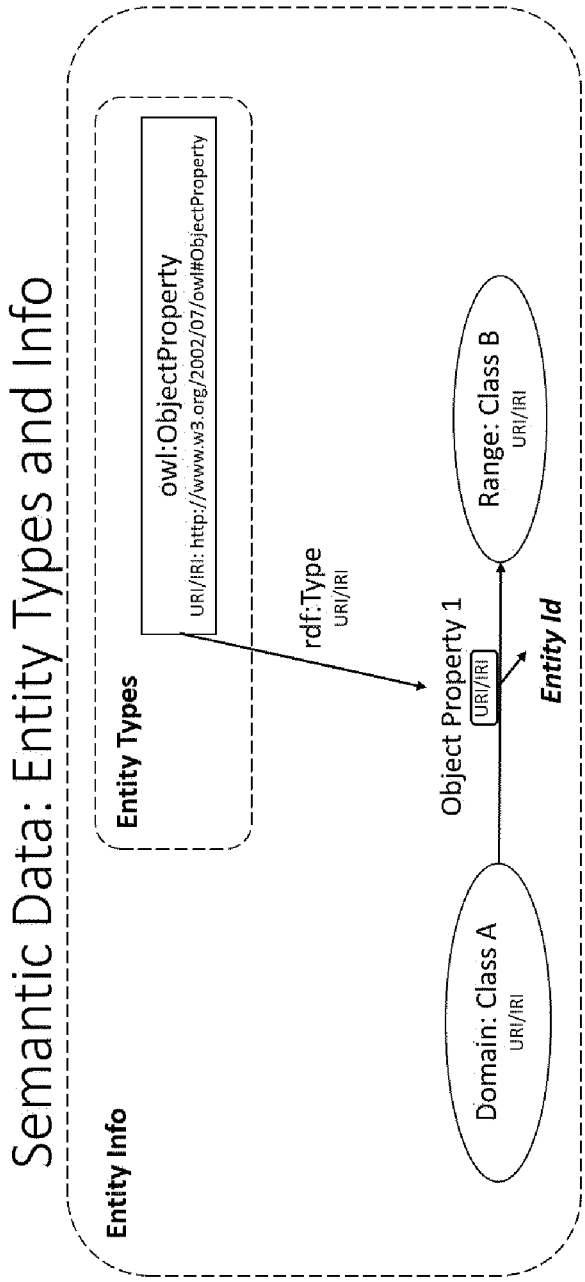
[도7]


Semantic Data: Entity Type and Info
Entity Info
Entity Types
owl:Class
URI/IRI: http://www.w3.org/2002/07/owl#Class
Class D
URI/IRI
rdf:Type
URI/IRI
Subclass Relation
URI/IRI
Class E
URI/IRI
Entity Id
Data Property 4
URI/IRI
Data Property Value Type
URI/IRI
Data Property 5
URI/IRI
Data Property 6
URI/IRI
Data Property Value Type
URI/IRI
Data Property Value Type
URI/IRI

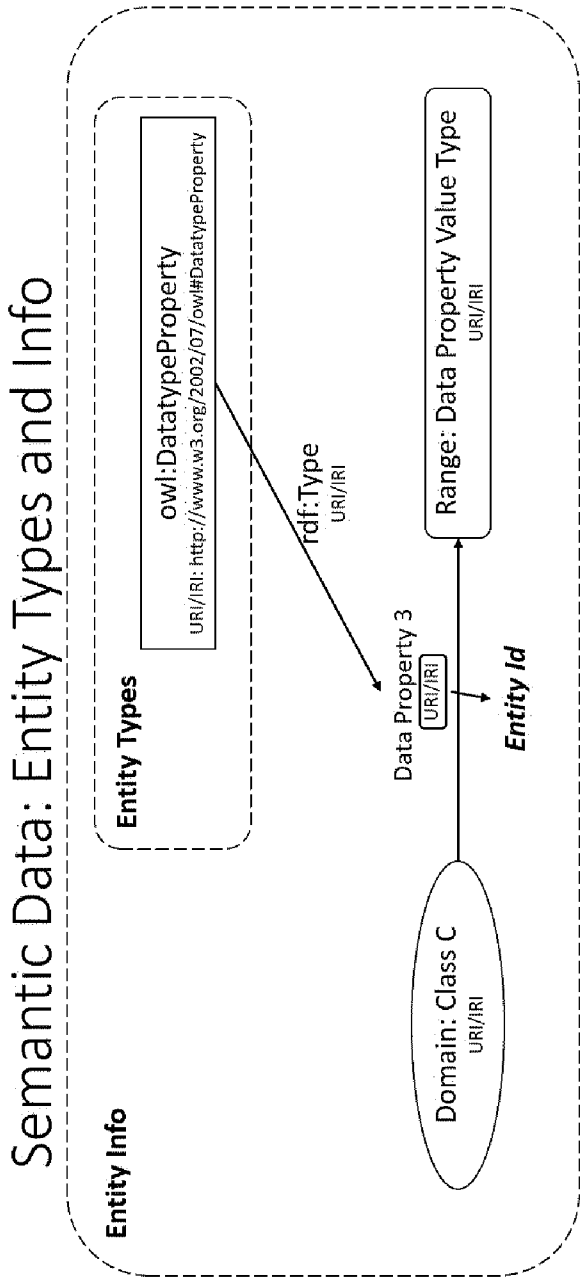
[도8]


Semantic Data: Entity Type and Info
Entity Info
Entity Types
Class B
URI/IRI
owl:NamedIndividual
URI/IRI: http://www.w3.org/2002/07/owl#NamedIndividual
rdf:Type
URI/IRI
rdf:Type
URI/IRI
Individual B_1
Type: Class B
URI/IRI
Entity Id
Object Property 1
URI/IRI
Individual A_1
Type: Class A
URI/IRI
Data Property 2
URI/IRI
Value based on Data Property Type

[도9]


Semantic Data: Entity Types and Info
Entity Info
Entity Types
owl:ObjectProperty
URI/IRI: http://www.w3.org/2002/07/owl#ObjectProperty
rdf:Type
URI/IRI
Object Property 1
URI/IRI
Entity Id
Domain: Class A
URI/IRI
Range: Class B
URI/IRI

[도10]


Semantic Data: Entity Types and Info
Entity Info
Entity Types
owl:DatatypeProperty
URI/IRI: http://www.w3.org/2002/07/owl#DatatypeProperty
rdf:Type
URI/IRI
Data Property 3
URI/IRI
Entity Id
Range: Data Property Value Type
URI/IRI
Domain: Class C
URI/IRI

[도11]

Semantic API: Simple and Normal Format Types

Entity Info Response Format: Simple

```
"entity-description" : {
    [
      { "rdf:type" : [ "owl:NamedIndividual" , "prefix1:ParkingLot" ] },
      { "prefix1:hasAddress" : [
                                "prefix1:ZoneBasedLocation_1_yt_lot_1" ,

      "prefix1:ZoneBasedLocation_2_yt_lot_1" ,
                                    "prefix1:ZoneBasedLocation_3_yt_lot_1" ,
                                    "prefix1:ZoneBasedLocation_4_yt_lot_1" ,
                                    "prefix1:ZoneBasedLocation_5_yt_lot_1"
                                    ] }
      ..........
        { "prefix1:hasDateCreated" : [ "prefix3: Instant_1_yt_lot_1" ] },
      ..........
        { "prefix1:hasLocationTag" : ["commercial"] },
      ..........
        { "prefix2:hasParkingLotProfile" : ["prefix2:ParkingLotProfile_yt_lot_1"]}
    ]

}
```

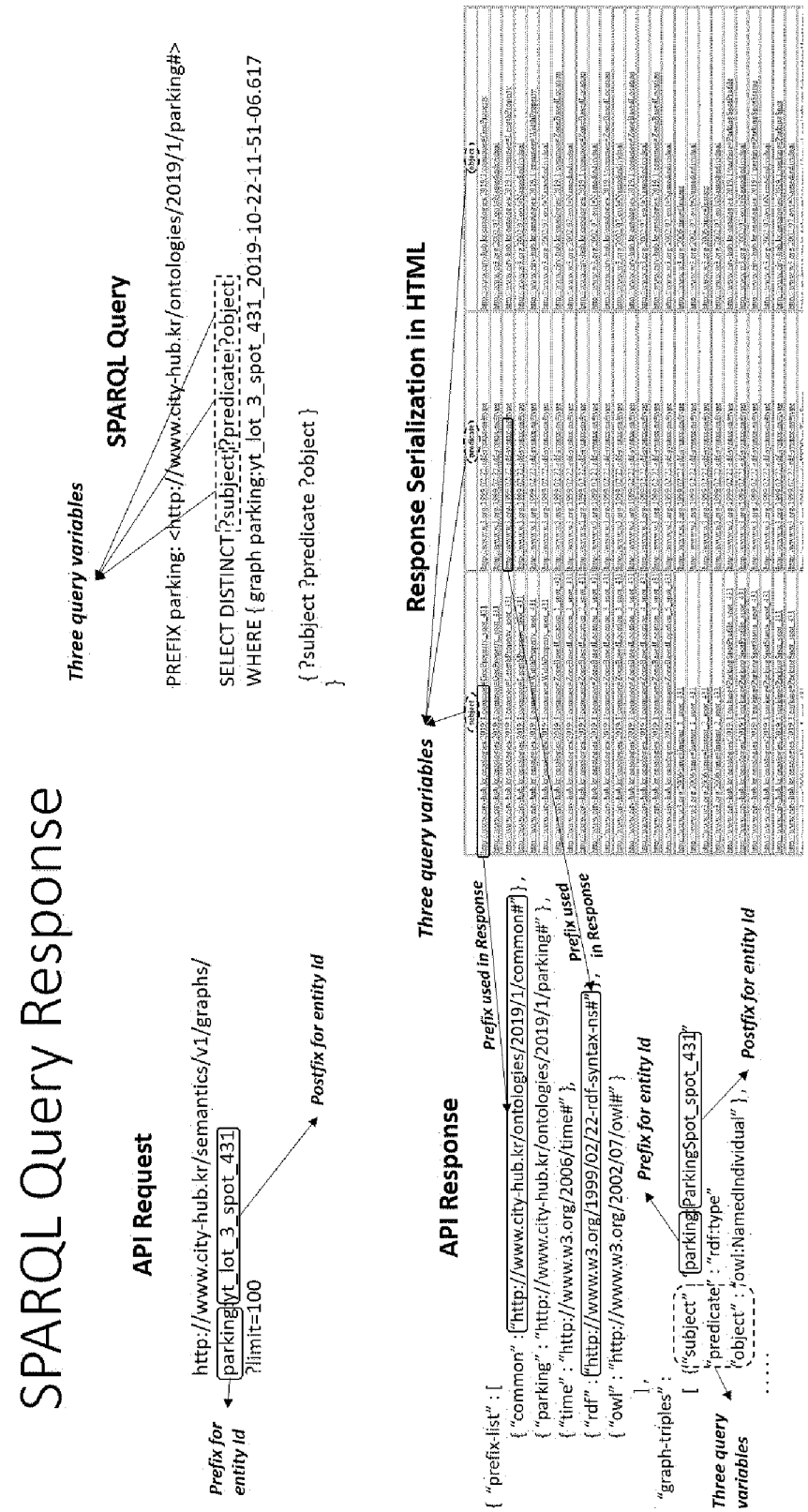
In case of Normal Format, note that the value of all "subject" keys are same i.e. "parking:ParkingLot_yt_lot_1", which is the URI/IRI of requested entity.
So in Simple Format, the subject is omitted and only values of "predicate" and "object" keys are included.

Predefined Keys

Entity Info Response Format: Normal

```
"entity-description" : {
                                { "subject" : "parking:ParkingLot_yt_lot_1",
                                  "predicate" : "rdf:type",
                                  "object" : "owl:NamedIndividual" },

                                { "subject" : "parking:ParkingLot_yt_lot_1",
                                  "predicate" : "rdf:type",
                                  "object" : "common:ParkingLot" },

                                  ..........

                                { "subject" : "parking:ParkingLot_yt_lot_1",
                                  "predicate" : "common:hasDateCreated",
                                  "object" : "time: Instant_1_yt_lot_1" },

                                  ..........

                                { "subject" : "parking:ParkingLot_yt_lot_1",
                                  "predicate" : "common:hasLocationTag",
                                  "object" : "commercial" },

                                  ..........

                                { "subject" : "parking:ParkingLot_yt_lot_1",
                                  "predicate" : "parking:hasParkingLotProfile",
                                  "object" : "parking:ParkingLotProfile_yt_lot_1" } ]

                                }
```

[도12]


SPARQL Query Response
API Request
http://www.city-hub.kr/semantics/v1/graphs/
parking:yt_lot_3_spot_431
?limit=100
Prefix for entity Id
Postfix for entity Id
Three query variables
SPARQL Query
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>
SELECT DISTINCT ?subject ?predicate ?object
WHERE { graph parking:yt_lot_3_spot_431_2019-10-22-11-51-06.617
{ ?subject ?predicate ?object }
}
Response Serialization in HTML
Three query variables
API Response
Prefix used in Response
Prefix used in Response
Prefix for entity Id
Three query variables
Postfix for entity Id

[표13]

Semantic REST API: Query Parameters

#	Name	Used In	Description
1.	Search Keyword List	• Graph List • Entity List	Keywords can be provided, based on which resource information will be searched, with matching keywords.
2.	Graph Type	Graph List	The data is organized into two types of graphs: ontology and instance. Ontology type graphs store all the ontology models whereas instance type graphs store the assertions (actual data) based on the ontology models
3.	Prefix format	All	Prefixes are identifiers of any namespace, defining whole or some part of any ontology. They are used as a part of any Entity URI/IRI as it's first part and are used to shorten the complete URI/IRI to provide readable and concise format. There are two types of prefix format: simple and normal.
4.	Entity Type List	Entity List	The entity type can have following values: Class, ObjectProperty, DataProperty, NamedIndividual. In addition, It can also involve a defined Class, object property or a data property URI/IRI in an ontology
5.	Entity URI/IRI	Entity Info	Here an entity URI/IRI can be a defined Class, object property or a data property URI/IRI in an ontology
6.	Response Format	Entity Info	The response can be organized into two types of format: simple and normal. Simple format is concise, whereas normal is detailed, describing each triple in terms of subject, predicate and object.
7.	ClassId	• Class Hierarchy • Individual List	It is an entity of type Class
7.	Response Limit	All	The number of graphs, entities, or triples to be returned. It can have a default value based on the server and database specifications.

[도14]

1. Graph List (Type: ontology)
Request (REST & SPARQL)
Request Type: GET
http://www.city-hub.kr/semantics/v1/graphs
?graphType=ontology
&keyword=parking,air-quality
&limit=10&prefixFormat=simple
SELECT DISTINCT ?graph
WHERE { GRAPH ?graph
{ ?s ?p ?o }
FILTER (!CONTAINS(STR(?graph), "#") &&
(
CONTAINS(STR(?graph), "parking") ||
CONTAINS(STR(?graph), "air-quality")
)
)
} ORDER BY ?graph
Variable for Graph
Response(API & HTML Serialization from SPARQL)
{
"prefix-list" : [
{ "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking" }
{ "air-quality" : "http://www.city-hub.kr/ontologies/2019/1/air-quality" }
],
"graphList" : [
"parking",
"air-quality"
]
}
graph
http://www.city-hub.kr/ontologies/2019/1/parking
http://www.city-hub.kr/ontologies/2019/1/air-quality

[도15]

1. Graph List (Type: instance)

Request (REST & SPARQL)

Request Type: GET

http://www.city-hub.kr/semantics/v1/graphs
?graphType=instance
&keyword=parking,air-quality
&limit=10&prefixFormat=simple

```
SELECT DISTINCT ?graph
WHERE { GRAPH ?graph
        { ?s ?p ?o }
    FILTER (    CONTAINS( STR(?graph), "#" )     &&
                (
                    CONTAINS( STR(?graph), "parking" )      ||
                    CONTAINS( STR(?graph), "air-quality" )
                )
    )
} ORDER BY ?graph
```

Response(API & HTML Serialization from SPARQL)

```
{
   "prefix-list" : [
{ "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking#" }
{ "air-quality" : "http://www.city-hub.kr/ontologies/2019/1/air-quality#" }
],
"graphList" : [
        "parking:yt_lot_1",
        "parking:yt_lot_2_spot_12",
        "parking:yt_lot_1_spot_431",
        "parking:yt_lot_2",
        "air-quality:seongnam_urn:datahub:AirQualityObserved",
        .....
            ]
}
```

graph
http://www.city-hub.kr/ontologies/2019/1/parking#yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#yt_lot_2_spot_12
http://www.city-hub.kr/ontologies/2019/1/parking#yt_lot_1_spot_431
http://www.city-hub.kr/ontologies/2019/1/parking#yt_lot_2
http://www.city-hub.kr/ontologies/2019/1/air-quality#seongnam_urn:datahub:AirQualityObserved
.....

[도16]

2. Entity List

Request (REST & SPARQL)

Request Type: GET

```
http://www.city-hub.kr/semantics/v1/entities
?keyword=parkinglot,parkingspot,longitude
&entityType=parking:ParkingLot,parking:ParkingSpot,owl:DatatypeProperty
&limit=10
```

```
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>

SELECT DISTINCT ?entity
WHERE
{
  { ?entity rdf:type parking:ParkingLot.}
  UNION
  { ?entity rdf:type parking:ParkingSpot.}
  UNION
  { ?entity rdf:type owl:DatatypeProperty.}
FILTER (
    CONTAINS( LCASE(STR(?entity)), "parkinglot" ) ||
    CONTAINS( LCASE(STR(?entity)), "parkingspot" ) ||
    CONTAINS( LCASE(STR(?entity)), "longitude" )
  )
}
LIMIT 100
```

Response(API & HTML Serialization from SPARQL)

```
{
  "prefix-list" : [
        { "common" : "http://www.city-hub.kr/ontologies/2019/1/common#" },
        { "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking#" }
  ],
  "entity-list" : [
        "parking:ParkingLot_yt_lot_1",
        "parking:ParkingLot_yt_lot_2",
        "parking:ParkingLot_yt_lot_3",
        "parking:ParkingLot_yt_lot_2_spot_122",
        "parking:ParkingLot_yt_lot_1_spot_33",
        .....
        "common:hasLongitude"
  ]
}
```

entity
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_2
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_3
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_2_spot_122
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1_spot_33
.....
http://www.city-hub.kr/ontologies/2019/1/common#hasLongitude

[도17]

3. Entity Info
Request (REST & SPARQL)
Request Type: GET
http://www.city-hub.kr/semantics/v1/entities/parking:ParkingLot_yt_lot_1
&responseType=simple
&limit=100
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>
DESCRIBE parking:ParkingLot_yt_lot_1
LIMIT 100
Response
{ "prefix-list" : [
{ "common" : "http://www.city-hub.kr/ontologies/2019/1/common" },
{ "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking" },
{ "time" : "http://www.w3.org/2006/time" },
{ "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns" },
{ "owl" : "http://www.w3.org/2002/07/owl" }
],
"entity-description" :{
[
{ "rdf:type" : ["owl:NamedIndividual" , "pakring:ParkingLot"] },
{ "common:hasAddress" : [
"common:ZoneBasedLocation_1_yt_lot_1" ,
"common:ZoneBasedLocation_2_yt_lot_1" ,
"common:ZoneBasedLocation_3_yt_lot_1" ,
"common:ZoneBasedLocation_4_yt_lot_1" ,
"common:ZoneBasedLocation_5_yt_lot_1"
] }
.....
{ "common:hasDateCreated" : ["time: Instant_1_yt_lot_1"] },
.....
{ "common:hasLocationTag" : ["commercial"] },
.....
{ "pakring:hasParkingLotProfile" : ["parking:ParkingLotProfile_yt_lot_1"] }
]
}
}

[도18]

3. Entity Info

Response(HTML Serialization from SPARQL)

http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Namedindividual
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_5_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_3_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_4_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_1_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_2_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasDateCreated	http://www.w3.org/2006/time#Instant_1_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1	http://www.city-hub.kr/ontologies/2019/1/common#hasDateModified	http://www.w3.org/2006/time#Instant_2_yt_lot_1
.....	.....	.....

[도19]

4. Class Hierarchy

Request (REST & SPARQL)

Request Type: GET

```
http://www.city-hub.kr/semantics/v1/classhierarchy/
parking:ParkingLot
&limit=100
```

```
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>

SELECT DISTINCT ?hierarchy
WHERE{
    {
      SELECT *
      WHERE{
        ?x rdfs:subClassOf+ ?hierarchy .
      }
    }
    FILTER (?x = parking:ParkingLot )
}
```

Response(API & HTML Serialization from SPARQL)

```
{   "prefix-list" : [
            { "common" :
                "http://www.city-hub.kr/ontologies/2019/1/common" } ,
            { "parking" :
                "http://www.city-hub.kr/ontologies/2019/1/parking" } ,
            ] ,
    "class-hierarchy" : [
            "parking:ParkingLot" ,
            "common:Zone" ,
            "common:System" ,
            "common:FeatureOfInterest" ,
            "owl:Thing"
            ]
    }
}
```

hierarchy
http://www.city-hub.kr/ontologies/2019/1/common#Zone
http://www.city-hub.kr/ontologies/2019/1/common#System
http://www.city-hub.kr/ontologies/2019/1/common#FeatureOfInterest

[도20]

5. Graph Info
Request (REST & SPARQL)
Request Type: GET
http://www.city-hub.kr/semantics/v1/graphs/
parking:yt_lot_3_spot_431
?limit=100
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>
SELECT DISTINCT ?subject ?predicate ?object
WHERE {
graph parking:yt_lot_3_spot_431_2019-10-22-11-51-06.617
{ ?subject ?predicate ?object }
}
Response
{ "prefix-list" : [
{ "common" : "http://www.city-hub.kr/ontologies/2019/1/common#" },
{ "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking#" },
{ "time" : "http://www.w3.org/2006/time#" },
{ "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#" },
{ "owl" : "http://www.w3.org/2002/07/owl#" }
],
"graph-triples" :
[{ "subject" : "parking:ParkingSpot_spot_431"
"predicate" : "rdf:type"
"object" : "owl:NamedIndividual" },
{ "subject" : "parking:ParkingSpot_spot_431"
"predicate" : "rdf:type"
"object" : "common:ParkingSpot" },
.....
{ "subject" : "parking:ParkingSpot_spot_431"
"predicate" : "common:hasDateCreated"
"object" : "time: Instant_1_spot_431" },
.....
{ "subject" : "parking:ParkingSpot_spot_431"
"predicate" : "parking:hasParkingSpotProfile"
"object" : "parking:ParkingSpotProfile_spot_431" }
.....
{ "subject" : "parking:ParkingSpotStatus_spot_431"
"predicate" : "parking:hasParkingStatusValue"
"object" : "occupied" }
.....]
}

[도21]

5. Graph Info

Response(HTML Serialization from SPARQL)

subject	predicate	object
http://www.city-hub.kr/ontologies/2019/1/common#GeoProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#GeoProperty
http://www.city-hub.kr/ontologies/2019/1/common#GeoProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#LengthProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#LengthProperty
http://www.city-hub.kr/ontologies/2019/1/common#LengthProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#WidthProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#WidthProperty
http://www.city-hub.kr/ontologies/2019/1/common#WidthProperty_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_1_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_1_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_2_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_2_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_3_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_3_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_4_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_4_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_5_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation
http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_5_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.w3.org/2006/time#Instant_1_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/time#Instant
http://www.w3.org/2006/time#Instant_1_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.w3.org/2006/time#Instant_2_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/time#Instant
http://www.w3.org/2006/time#Instant_2_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotProfile_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotProfile
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotProfile_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotStatus_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotStatus
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpotStatus_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
http://www.w3.org/2006/time#Instant_1_spot_431	http://www.w3.org/2006/time#inXSDDateTimeStamp	"2019-06-09T18:20:03.562+09:00"^^<http://www.w3.org/2001/XMLSchema#dateTimeStamp>
http://www.w3.org/2006/time#Instant_2_spot_431	http://www.w3.org/2006/time#inXSDDateTimeStamp	"2019-10-28T13:57:39.354+09:00"^^<http://www.w3.org/2001/XMLSchema#dateTimeStamp>
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_1_spot_431
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_2_spot_431
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_3_spot_431
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_4_spot_431
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingSpot_spot_431	http://www.city-hub.kr/ontologies/2019/1/common#hasAddress	http://www.city-hub.kr/ontologies/2019/1/common#ZoneBasedLocation_5_spot_431
.....	.....	.....

[도22]

6. Individual List

Request (REST & SPARQL)

Request Type: GET

http://www.city-hub.kr/semantics/v1/individuals
?classId=parking:ParkingLot, parking:ParkingSpot
&limit=10

```
PREFIX parking: <http://www.city-hub.kr/ontologies/2019/1/parking#>

SELECT DISTINCT ?individual
WHERE
{
  ?individual rdf:type owl:NamedIndividual .
  { ?individual rdf:type parking:ParkingLot }
  UNION
  { ?individual rdf:type parking:ParkingSpot }
}
LIMIT 20
```

Response(API & HTML Serialization from SPARQL)

```
{
  "prefix-list" : [
  { "parking" : "http://www.city-hub.kr/ontologies/2019/1/parking" }
    ],
  "named-individuals" : [
          "parking:ParkingLot_yt_lot_1",
          "parking:ParkingLot_yt_lot_2",
          "parking:ParkingLot_yt_lot_3",
          "parking:ParkingLot_yt_lot_2_spot_122",
          "parking:ParkingLot_yt_lot_1_spot_33",
          .....
    ]
}
```

individual
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_2
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_3
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_2_spot_122
http://www.city-hub.kr/ontologies/2019/1/parking#ParkingLot_yt_lot_1_spot_33
.....

INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR2020/019262

A. CLASSIFICATION OF SUBJECT MATTER

G06F 16/332(2019.01)i; **G06F 16/36**(2019.01)i; **G06F 9/54**(2006.01)i; **G06F 9/451**(2018.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 16/332(2019.01); G06F 17/30(2006.01); H04L 29/06(2006.01); H04L 29/08(2006.01); H04M 3/42(2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models: IPC as above
Japanese utility models and applications for utility models: IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS (KIPO internal) & keywords: 시맨틱(semantic), 파싱(parsing), 파라미터(parameter), 쿼리(query), 트리플(triple), SPARQL(Simple Protocol and RDF Query Language), REST(Representational State Transfer)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	MARKLOGIC CORPORATION. MarkLogic Server - Semantic Graph Developer's Guide. June 2019, pp . 1-268 [Retrieved on 03 September 2021]. Retrieved from <https://docs.marklogic.com/guide/semantics.pdf>. See pages 18, 59, 83, 92, 165, 189, 192, 208, 216 and 223.	1-8
Y	CN 107872484 B (CHINA TELECOM CORPORATION LIMITED) 25 December 2020 (2020-12-25) See paragraph [0019]; and claims 1 and 4.	1-8
A	KR 10-1927450 B1 (WISENUT, INC.) 10 December 2018 (2018-12-10) See claims 1-3; and figures 1-2.	1-8
A	US 2016-0314212 A1 (FUJITSU LIMITED) 27 October 2016 (2016-10-27) See claims 1-12.	1-8
A	CN 112003855 A (HANGZHOU WISTSKY SCIENCE AND TECHNOLOGY CO., LTD.) 27 November 2020 (2020-11-27) See claim 10.	1-8

☐ Further documents are listed in the continuation of Box C.

☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 September 2021	**29 September 2021**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **Government Complex-Daejeon Building 4, 189 Cheongsa-ro, Seo-gu, Daejeon 35208**	
Facsimile No. **+82-42-481-8578**	Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/KR2020/019262

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 107872484 B	25 December 2020	None	
KR 10-1927450 B1	10 December 2018	None	
US 2016-0314212 A1	27 October 2016	EP 3086242 A1	26 October 2016
		GB 2537670 A	26 October 2016
		JP 2016-207202 A	08 December 2016
CN 112003855 A	27 November 2020	None	

국 제 조 사 보 고 서

국제출원번호
PCT/KR2020/019262

A. 발명이 속하는 기술분류(국제특허분류(IPC))

G06F 16/332(2019.01)i; **G06F 16/36**(2019.01)i; **G06F 9/54**(2006.01)i; **G06F 9/451**(2018.01)i

B. 조사된 분야

조사된 최소문헌(국제특허분류를 기재)

G06F 16/332(2019.01); G06F 17/30(2006.01); H04L 29/06(2006.01); H04L 29/08(2006.01); H04M 3/42(2006.01)

조사된 기술분야에 속하는 최소문헌 이외의 문헌

한국등록실용신안공보 및 한국공개실용신안공보: 조사된 최소문헌란에 기재된 IPC
일본등록실용신안공보 및 일본공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

국제조사에 이용된 전산 데이터베이스(데이터베이스의 명칭 및 검색어(해당하는 경우))

eKOMPASS(특허청 내부 검색시스템) & 키워드: 시맨틱(semantic), 파싱(parsing), 파라미터(parameter), 쿼리(query), 트리플(triple), SPARQL(Simple Protocol and RDF Query Language), REST(Representational State Transfer)

C. 관련 문헌

카테고리*	인용문헌명 및 관련 구절(해당하는 경우)의 기재	관련 청구항
Y	MARKLOGIC CORPORATION, `MarkLogic Server - Semantic Graph Developer's Guide`, 2019.06, pp. 1-268 [검색일: 2021.09.03]. 출처: <https://docs.marklogic.com/guide/semantics.pdf> 페이지 18, 59, 83, 92, 165, 189, 192, 208, 216, 223	1-8
Y	CN 107872484 B (CHINA TELECOM CORPORATION LIMITED) 2020.12.25 단락 [0019]; 및 청구항 1, 4	1-8
A	KR 10-1927450 B1 (주식회사 와이즈넛) 2018.12.10 청구항 1-3; 및 도면 1-2	1-8
A	US 2016-0314212 A1 (FUJITSU LIMITED) 2016.10.27 청구항 1-12	1-8
A	CN 112003855 A (HANGZHOU WISTSKY SCIENCE AND TECHNOLOGY CO., LTD.) 2020.11.27 청구항 10	1-8

☐ 추가 문헌이 C(계속)에 기재되어 있습니다.

☑ 대응특허에 관한 별지를 참조하십시오.

* 인용된 문헌의 특별 카테고리:

"A" 특별히 관련이 없는 것으로 보이는 일반적인 기술수준을 정의한 문헌
"D" 본 국제출원에서 출원인이 인용한 문헌
"E" 국제출원일보다 빠른 출원일 또는 우선일을 가지나 국제출원일 이후에 공개된 선출원 또는 특허 문헌
"L" 우선권 주장에 의문을 제기하는 문헌 또는 다른 인용문헌의 공개일 또는 다른 특별한 이유(이유를 명시)를 밝히기 위하여 인용된 문헌
"O" 구두 개시, 사용, 전시 또는 기타 수단을 언급하고 있는 문헌
"P" 우선일 이후에 공개되었으나 국제출원일 이전에 공개된 문헌
"T" 국제출원일 또는 우선일 후에 공개된 문헌으로, 출원과 상충하지 않으며 발명의 기초가 되는 원리나 이론을 이해하기 위해 인용된 문헌
"X" 특별한 관련이 있는 문헌. 해당 문헌 하나만으로 청구된 발명의 신규성 또는 진보성이 없는 것으로 본다.
"Y" 특별한 관련이 있는 문헌. 해당 문헌이 하나 이상의 다른 문헌과 조합하는 경우로 그 조합이 당업자에게 자명한 경우 청구된 발명은 진보성이 없는 것으로 본다.
"&" 동일한 대응특허문헌에 속하는 문헌

국제조사의 실제 완료일	국제조사보고서 발송일
2021년09월28일(28.09.2021)	**2021년09월29일(29.09.2021)**

ISA/KR의 명칭 및 우편주소

대한민국 특허청
(35208) 대전광역시 서구 청사로 189, 4동 (둔산동, 정부대전청사)

팩스 번호 **+82-42-481-8578**

심사관

변성철

전화번호 **+82-42-481-8262**

국 제 조 사 보 고 서
대응특허에 관한 정보

국제출원번호
PCT/KR2020/019262

국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
CN 107872484 B	2020/12/25	없음	
KR 10-1927450 B1	2018/12/10	없음	
US 2016-0314212 A1	2016/10/27	EP 3086242 A1	2016/10/26
		GB 2537670 A	2016/10/26
		JP 2016-207202 A	2016/12/08
CN 112003855 A	2020/11/27	없음	