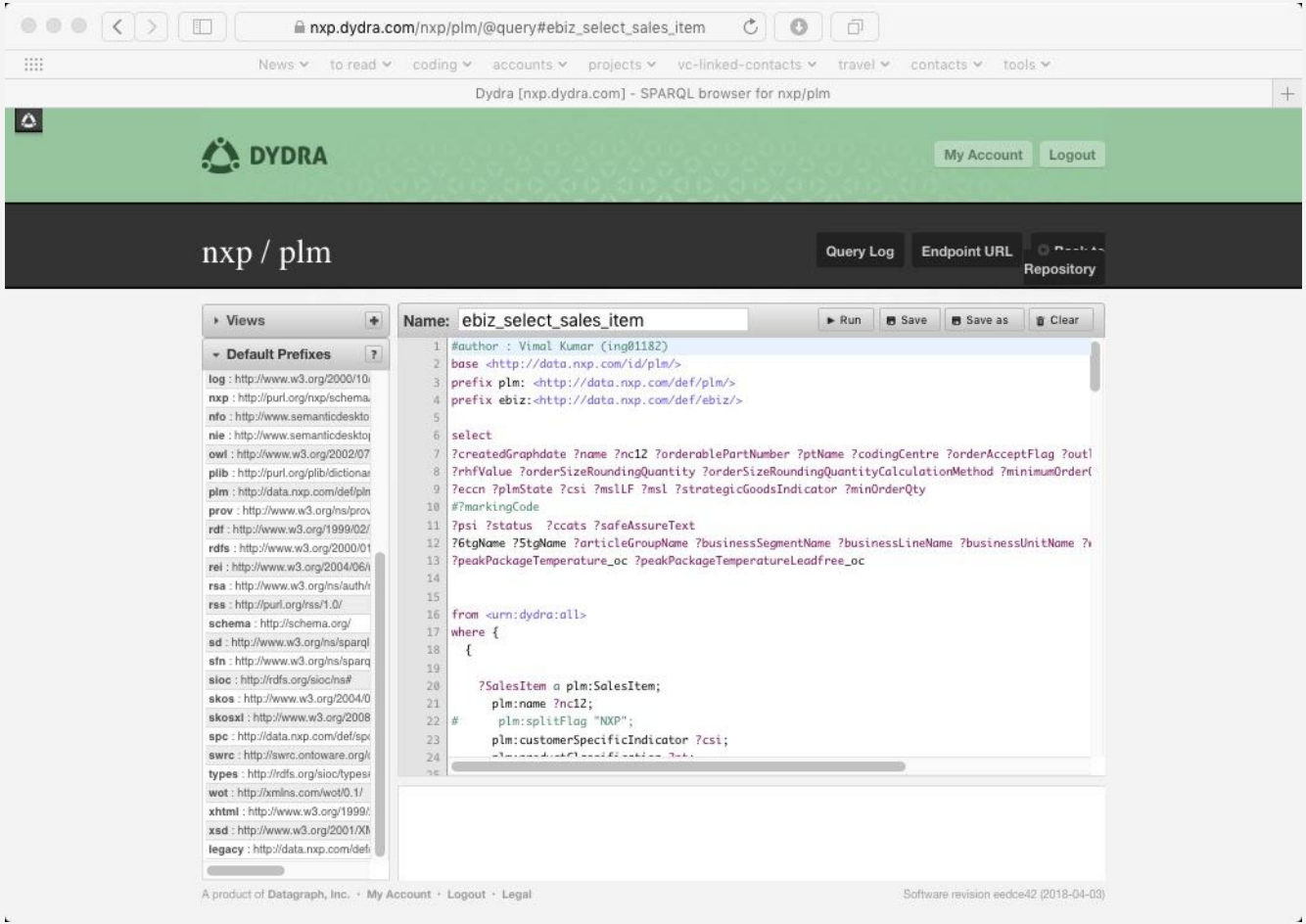
Dydra

define-declaration

http://dydra.com

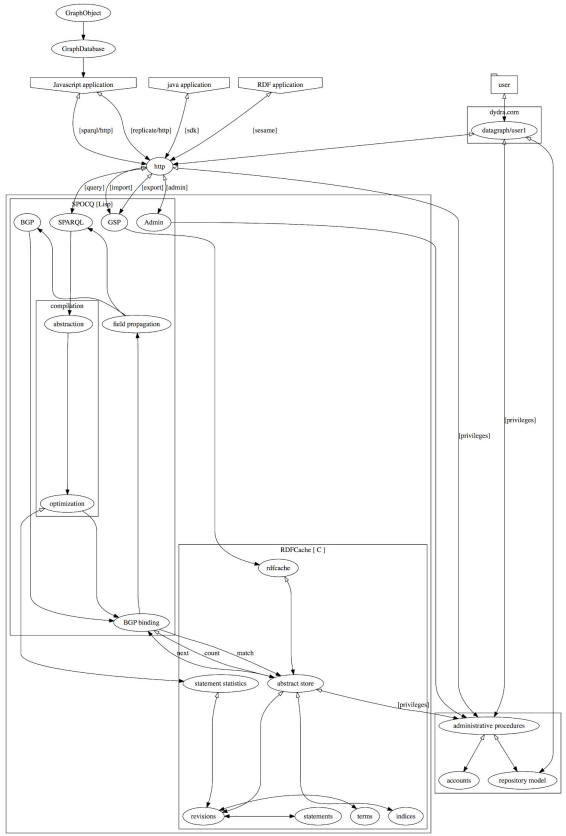
or ...
don't walk

a sparql service



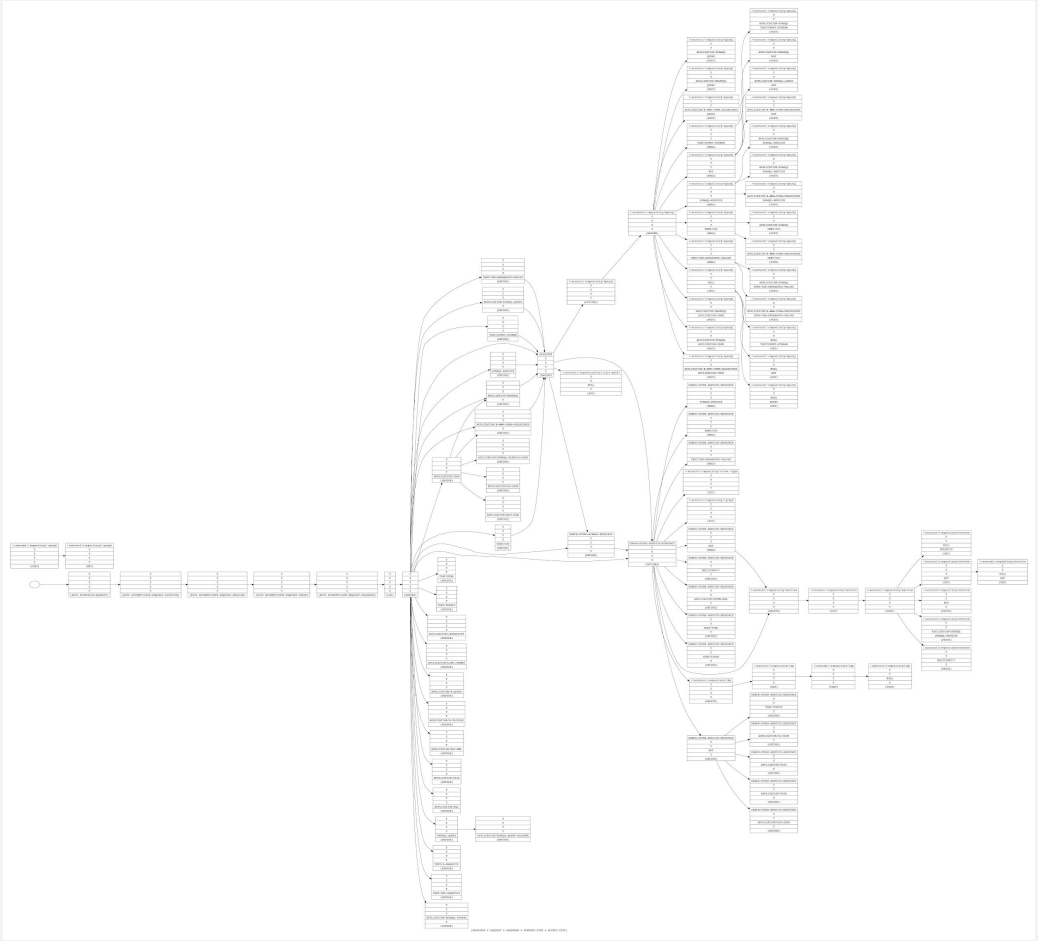
http://dydra.com

a sparql service



http://dydra.com

a sparql service



http://dydra.com

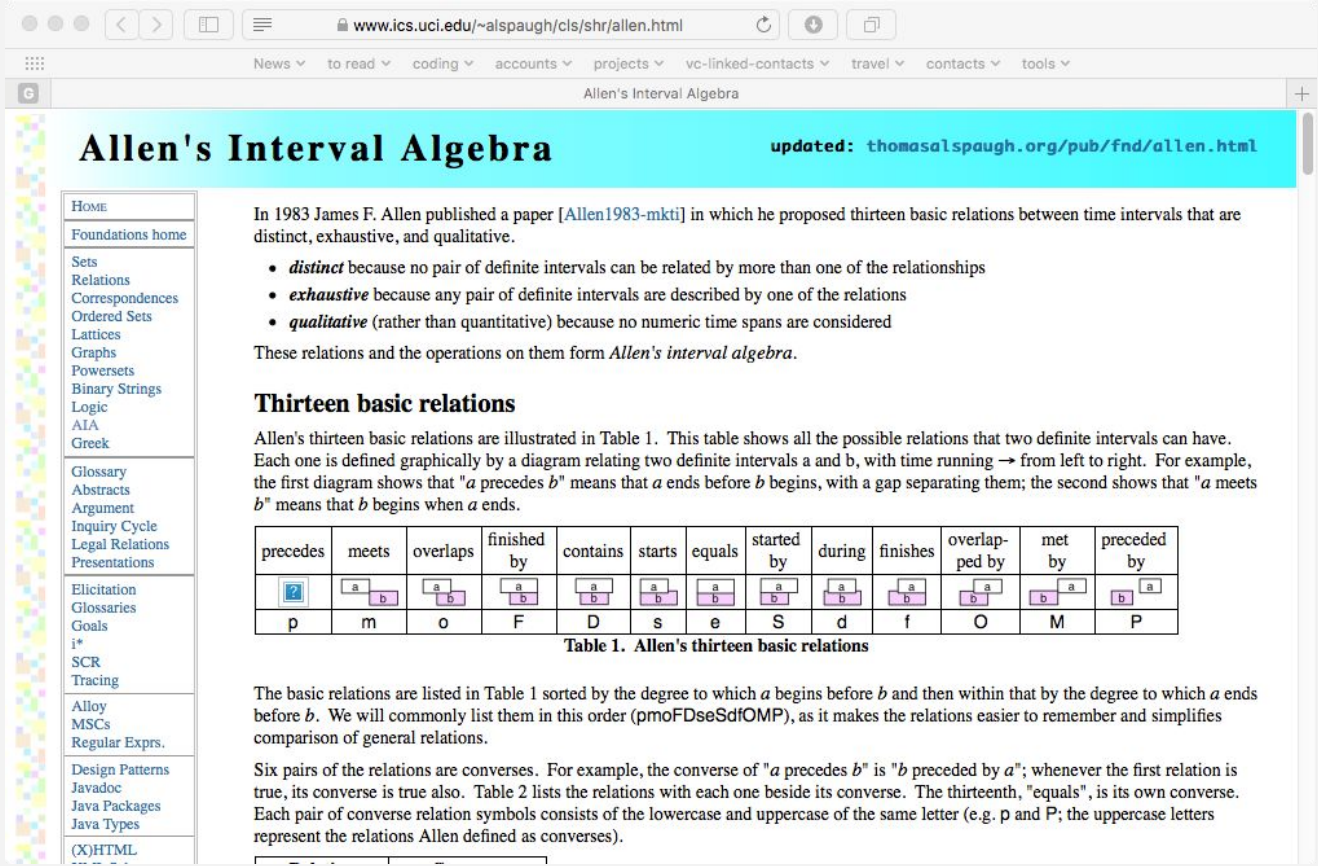
sbcl 20

simple sparql queries

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res:  <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?uri ?string
WHERE {
       ?uri rdf:type dbo:Book .
        ?uri dbo:author res:Danielle_Steel .
       OPTIONAL { ?uri rdfs:label ?string . FILTER (lang(?string) = 'en') }
}
```


https://aifb-ls3-kos.aifb.kit.edu/projects/spartiqulator/examples.htm

temporal constraints



https://www.ics.uci.edu/~alspaugh/cls/shr/allen.html

simple filtered queries

```
SELECT ?o ?s ?y
WHERE {
  { ?s <http://dbpedia.org/property/country> ?o
    FILTER ( <http://www.w3.org/2006/time#versionIncludes>(1)) }
  { ?o <http://www.w3.org/2000/01/rdf-schema#label> ?y
     FILTER ( <http://www.w3.org/2006/time#versionIncludes>(6)) }
}
```

simple filtered queries

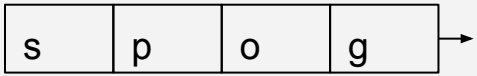
```
(select
   (join
    (filter
     (bgp
       {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y})
     (<http://www.w3.org/2006/time#versionIncludes> 6))
    (filter
     (bgp
       {?s <http://dbpedia.org/property/country> ?o})
     (<http://www.w3.org/2006/time#versionIncludes> 1)))
   (?o ?s ?y))
```

simple filtered queries

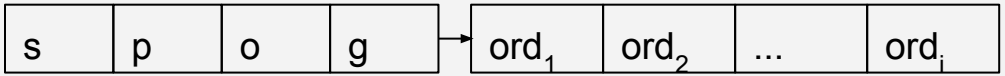
```
(PROJECT
 (JOIN
  (agp (id #:AGP-1577)
       {?s <http://dbpedia.org/property/country> ?o :DIMENSIONS (?::s #:constant1576 ?::o)}
    (filter (<http://www.w3.org/2006/time#versionIncludes> 1))
    (filter (<http://www.w3.org/2006/time#versionIncludes> 6)))
  (agp (id #:AGP-1578)
       {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y :DIMENSIONS (?::o #:constant1575 ?::y)}
    (filter (<http://www.w3.org/2006/time#versionIncludes> 1))
    (filter (<http://www.w3.org/2006/time#versionIncludes> 6))))
 '(?o ?s ?y))
```

alternative index forms

- static index entries :

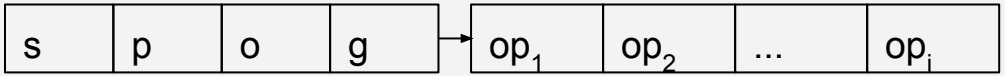


- revisioned index entries : a sequence of insertion/deletion ordinals



The adoption of timestamps made-up of temporal elements instead of (multi-temporal) simple intervals avoids the duplication of triples in the presence of a temporal pertinence with a complex shape. In fact, we store different triple versions only once with a complex timestamp rather than storing multiple copies of them with a simple timestamp as in [6, 10, 17]. The memory saving we obtain grows with the dimensionality of the time domain, but it can even be appreciated with a monodimensional time domain, when the temporal pertinence of a triple is not a convex interval.[1]

- replicated : a sequence of located insertion/deletion operations



[1]: Grandi.2010 : Grandi F. T-SPARQL: A TSQL2-like Temporal Query Language for RDF

actual sparql queries

http://dydra.com

sbcl 20

actual sparql queries

```
#author : Vimal Kumar (ing01182)
base <http://data.nxp.com/id/plm/>
prefix nxp: <http://purl.org/nxp/schema/v1/>
prefix spc: <http://data.nxp.com/def/spc/>
prefix plm: <http://data.nxp.com/def/plm/>
prefix legacy: <http://data.nxp.com/def/legacy/>
prefix plib: <http://purl.org/plib/dictionary.owl#>
prefix ebiz: <http://data.nxp.com/def/ebiz/>

SELECT distinct ?name ?magNameList #?magName
?type ?status ?state
?nxpPackageCode ?psi ?csi ?codingCenter
?createdGraphdate ?harmonizedSystemName ?finalpov ?searchableflag ?marketingDescription
?fit ?mtbfVal ?eftVal ?spcGraphDate

from <urn:dydra:all>

WHERE {

    ?bt a plm:BasicType;
     #plm:description ?description;
     #plm:splitFlag "NXP";
     plm:name ?name;
     plm:requires [plm:name ?nxpPackageCode];
     plm:productSecurityIndicator ?psi;
     plm:customerSpecificIndicator ?csi;
     plm:state ?state.

    ###################
```

actual sparql queries

```
    ##################

    optional{
    #pick no recommend from ebiz - starts
     {
        select ?name (group_concat(distinct ?mainArticleGroupName ; separator="-") as ?magNameList) (group_concat(?plmStateList ;
separator="-")as ?lastval)
       #select ?name (sample(?mainArticleGroupName) as ?magName) (group_concat(?plmStateList ; separator="-")as ?lastval)

(max(?updateDate) as ?dateEbizWeb)
        where
        {    ?basicType plm:name ?name.
             ?basicType a plm:BasicType.
             #optional{
             #?basicType plm:name ?name.
             #?basicType a plm:BasicType.
             ?productType plm:productClassification ?basicType;
             a plm:ProductType;
             plm:name ?pname.

             ?salesItem plm:productClassification ?productType ;
             a plm:SalesItem;
             plm:state ?siState;
             plm:notRecommendedForNewDesigns ?nrfnd_enovia;
             plm:name ?sname.
             optional{ ?salesItem  plm:codingCentre ?codingCentreSI.}

             # Mag Code Identification
```

actual sparql queries

```
        # Mag Code Identification

        optional{
          service <http://localhost/james/test> {
            Graph<urn:dydra:all>{
             ?salesItem ebiz:PP_LAST_UPDATE_DTE ?ebizupdateDate.
               optional{
                  ?salesItem ebiz:SALES_ITEM_NO_RECOMMEND_FLG ?nrfnd_ebiz.
               }
            }
          }
        }

                BIND (
                                                IF(?codingCentreSI IN ("CP-ATX-01", "CP-ATX-02"), ?nrfnd_ebiz, ?nrfnd_enovia)
                                                        AS ?notRecommendedState
                                                )
        BIND(
  IF(
    BOUND(?notRecommendedState),
    IF(?notRecommendedState="Yes",
      concat(?siState,"-Yes"),
      IF(?notRecommendedState="No",concat(?siState,"-No"),concat(?siState,"-BL"))
    ),
    concat(?siState,"-BL")
  )
 AS ?plmStateList
)

        bind(xsd:dateTime(IF(bound(?ebizupdateDate), ?ebizupdateDate, "1900-11-27T18:34:55Z"^^xsd:dateTime)) as ?updateDate)
```

actual sparql queries

```
        bind(xsd:dateTime(IF(bound(?ebizupdateDate), ?ebizupdateDate, "1900-11-27T18:34:55Z"^^xsd:dateTime)) as ?updateDate)
        #bind(if(bound(?nrfnd_ebiz),"NotRecommendedState",?siState) as ?plmStateList)
        #bind(if(?nrfnd_ebiz="true"^^xsd:boolean,"NotRecommendedState",?siState) as ?plmStateList)
      }
    group by ?name
}
#pick no recommend from ebiz - ends
}

 {
   ?bt a plm:BasicType;
   optional
   {
      service <http://localhost/james/test> {
          graph  <urn:dydra:all>{
             ?bt nxp:publishToWeb ?ptw.
          }
      }
   }
 }
 {
     ?bt a plm:BasicType;
     plm:name ?name;
      optional
      {
          service <http://localhost/james/test> {
             graph  <urn:dydra:all>{

                  optional{
                    ?legacyBt rdfs:label ?name.
                    ?legacyBt legacy:P_AAD290 ?fit.

                  }
```

actual sparql queries

```
                optional{
                  ?legacyBt rdfs:label ?name.
                  ?legacyBt legacy:P_AAD290 ?fit.

                }
                 optional{
                    ?legacyBt rdfs:label ?name.
                    ?legacyBt legacy:P_AAD272 ?mtbf.
                    ?mtbf plib:hasValue ?mtbfVal.
                 }
                 optional{
                    ?legacyBt rdfs:label ?name.
                    ?legacyBt legacy:P_AAD260 ?eft.
                    ?eft plib:hasValue ?eftVal.
                 }
            }
          }
      }
}

optional
{
   service <http://localhost/james/test> {
    graph  ?g2{
     #?bts  rdfs:label ?name.
       optional{
         ?bts  rdfs:label ?name.
         ?bts  spc:P_PAE079 ?marketingDescription.  }
       optional{ ?bts  rdfs:label ?name.   ?bts spc:P_AAD324 ?povEptos.
       }
       bind(IRI(?g2) as ?spcGraph)
       ?spcGraph dcterms:created ?spcGraphDate.
     }
```

actual sparql queries

```
        optional {graph ?g3 {
            ?povEptos spc:P_PAE127 ?poveptosdesc.
        }}
      }
    }

    optional {?bt plm:externalClassification ?taricCode.
            ?taricCode a plm:TaricCode;
            plm:externalClassification
                [a plm:HarmonizedSystemType; plm:name ?harmonizedSystemName].
    }


    optional {?bt plm:codingCentre ?codingCenter.}

  optional {
      ?bt plm:state "OBS".
      bind("No Longer Manufactured" as ?btState)
    }

    optional
{
    select distinct ?name (min(?povname) as ?pov) (sample(?povEnoviaDescVal) as ?povEnoviaDesc)
    where {
      {
        select * where {
          {
            select * where {
              ?subpackage a plm:Subpackage ;
                plm:packageClassification ?povEnovia .
              ?povEnovia plm:description ?povEnoviaDescVal .
              ?povEnovia plm:name ?povname.
            }
          }
```

actual sparql queries

```
        select * where {
          {
            select * where {
              ?subpackage a plm:Subpackage ;
                plm:packageClassification ?povEnovia .
              ?povEnovia plm:description ?povEnoviaDescVal .
              ?povEnovia plm:name ?povname.
            }
          }
          ?subpackage ^plm:bom+ ?SalesItem
        }
      }

      ?SalesItem a plm:SalesItem ;
        plm:productClassification [ plm:productClassification [ plm:name ?name ] ] .
    }
    group by ?name

  }
  bind(COALESCE(?poveptosdesc,?povEnoviaDesc, "") As ?finalpov).
 BIND (
COALESCE(
  if(bound(?btState),?btState,1/0),
    IF(contains(?lastval,"RFS-No")||contains(?lastval,"RFS-BL"), "Active", 1/0),
      IF(contains(?lastval,"RFS-Yes"),"Not Recommended for New Designs", 1/0),
        IF(contains(?lastval,"CQS-Yes")||contains(?lastval,"CQS-No")||contains(?lastval,"CQS-BL"), "Qualification", 1/0),

IF(contains(?lastval,"DEV-Yes")||contains(?lastval,"ASM-Yes")||contains(?lastval,"DEV-No")||contains(?lastval,"ASM-No")||contains(?lastval,"DE
V-BL")||contains(?lastval,"ASM-BL"), "Development", 1/0),
#    IF(contains(?lastval,"NotRecommendedState"), "Not Recommended for New Designs", 1/0),
          IF(contains(?lastval,"DOD-Yes")||contains(?lastval,"DOD-No")||contains(?lastval,"DOD-BL"), "End of Life", 1/0),
```

actual sparql queries

```
IF(contains(?lastval,"WIT-Yes")||contains(?lastval,"OBS-Yes")||contains(?lastval,"WIT-No")||contains(?lastval,"OBS-No")||contains(?lastval,"WIT
-BL")||contains(?lastval,"OBS-BL"), "No Longer Manufactured", 1/0),
    "NA"
  ) AS ?status)

      bind(if(contains(?magNameList,"RST") || contains(?magNameList,"RAU") || contains(?magNameList,"RAE") ||
contains(?magNameList,"RCU"),"yes", "no") as ?isMarvellData)
      # bind(if(contains(?lastval,"yes") && !contains(?lastval,"no"),"Not Recommended for New Designs", 1/0) as ?notrecommended)
      #bind (coalesce(?btState, ?rfs, ?cqs, ?asm, ?dev, ?dod, ?endoflife, "Development") as ?status)
      #bind (coalesce(?btState,?notrecommended, ?rfs, ?cqs, ?asm, ?dev, ?dod, ?endoflife, "Development") as ?status)
       bind (if((lcase(?nxpPackageCode)="nad000"
||lcase(?nxpPackageCode)="nak000"||lcase(?nxpPackageCode)="nar000"),"Demoboard","BasicType")as ?type)
       ?btGraph foaf:primaryTopic ?bt;
              dcterms:created ?createdGraphdate.
       bind(IF(bound(?ptw),?ptw,"true"^^xsd:boolean) as ?sFlag)
       bind(IF(?sFlag,"Y","N") as ?searchableflag)
       bind(xsd:dateTime(IF(bound($date), $date, "1900-11-27T18:34:55Z"^^xsd:dateTime)) as ?dateArg)
       filter(xsd:dateTime(?createdGraphdate) >= ?dateArg || xsd:dateTime(?spcGraphDate) >= ?dateArg ||?dateEbizWeb >= ?dateArg)
#       filter(xsd:dateTime(?createdGraphdate) >= ?dateArg || xsd:dateTime(?spcGraphDate) >= ?dateArg ||
xsd:dateTime(?convertedDateTime) >= ?dateArg)
       #FILTER ( bound($name) && ucase(?name)=$name )
      #filter (?magName not in ("RST","RAU","RAE","RCU") )
      filter (?isMarvellData = "no")
}
order by ?name
#limit 10
```

"complex" filtered queries

```
SELECT ?o ?s ?y
WHERE {
  { SELECT * { ?s <http://dbpedia.org/property/country> ?o .
               ?o <http://www.w3.org/2000/01/rdf-schema#label> ?y }
  }
  FILTER ( <http://www.w3.org/2006/time#versionIncludes>(1))
  FILTER ( <http://www.w3.org/2006/time#versionIncludes>(6))
}
```

simple filtered queries

```
(select
   (filter
    (select
     (bgp
       {?s <http://dbpedia.org/property/country> ?o}
       {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y})
     (?o ?s ?y))
    (exprlist (<http://www.w3.org/2006/time#versionIncludes> 1)
              (<http://www.w3.org/2006/time#versionIncludes> 6)))
   (?o ?s ?y))
```

simple filtered queries

```
(PROJECT
 (LOCALLY
   (DECLARE (REFERENCE-DIMENSIONS ?y ?s ?o))
   (LOCALLY
     (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 6)))
     (LOCALLY
       (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 1)))
       (LOCALLY
         (DECLARE (VERSION-CONSTRAINT COMMON-LISP:NIL))
         (LOCALLY
           (DECLARE (PROJECTION-DIMENSIONS ?o ?s ?y))
           (JOIN
            (agp (id #:AGP-1581)
                 {?s <http://dbpedia.org/property/country> ?o :DIMENSIONS (?::s #:constant1579 ?::o)})
            (agp (id #:AGP-1582)
                 {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y :DIMENSIONS (?::o #:constant1580
?::y)}))))))))
 '(?o ?s ?y))
```

establish constraint

```
 (define-declaration spocq.e:version-constraint (declaration &optional env)
   "Capture a temporal constraint which has been articulated in some reference to
  a pattern variable. Make it available for the BGP evaluation process to apply it to
  the statement matching process."
    (destructuring-bind (tag form) declaration
      (let ((old (when env (declaration-information 'spocq.e:version-constraint env))))
        ;; allow nil to shadow outer declarations in a sub-select
        (values :declare (cons tag (when form (cons form old)))))))

 (defmacro spocq.e:with-version-constraint (constraint &body body)
   `(locally (declare (spocq.e:version-constraint ,constraint)) ,@body))
```

```
(defun push-filter-test (test-expression field-expression)
  "Given a filter over some field, examine the particular field form
   and decide whether it is possible to push the filter down into it.
   If so, return the rewritten field form.
   If not, return NIL"
  (unless (find 'spocq.a:|exists| (expression-algebra-operators test-expression))
    (let ((test-variables (expression-variables test-expression))
          (field-variables (expression-variables field-expression))
          (dynamic-variables (first (task-dynamic-bindings *task*))))
      ;(print (list test-variables field-variables dynamic-variables))
      (unless (set-difference (set-difference test-variables field-variables) dynamic-variables)
        (cond ((version-constraint-p test-expression)
               `(spocq.e:with-version-constraint ,test-expression
                  ,field-expression))
              ((bgp-form-p field-expression)
               ;; push the just filter and the slice into the bgp.
               `(spocq.a:|bgp| (spocq.a:|filter| ,test-expression)
                         ,@(rest field-expression)))
  ;;; …
  )))))
```

apply constraint

```
 (defun push-filter-test (test-expression field-expression)
   "Given a filter over some field, examine the particular field form
    and decide whether it is possible to push the filter down into it.
    If so, return the rewritten field form.
    If not, return NIL"
   (unless (find 'spocq.a:|exists| (expression-algebra-operators test-expression))
     (let ((test-variables (expression-variables test-expression))
           (field-variables (expression-variables field-expression))
           (dynamic-variables (first (task-dynamic-bindings *task*))))
       ;(print (list test-variables field-variables dynamic-variables))
       (unless (set-difference (set-difference test-variables field-variables) dynamic-variables)
         (cond ((version-constraint-p test-expression)
                `(spocq.e:with-version-constraint ,test-expression
                   ,field-expression))
               ((bgp-form-p field-expression)
                ;; push the just filter and the slice into the bgp.
                `(spocq.a:|bgp| (spocq.a:|filter| ,test-expression)
                          ,@(rest field-expression)))
   ;;; …
   )))))
```

```
(PROJECT
 (LOCALLY
   (DECLARE (REFERENCE-DIMENSIONS ?y ?s ?o))
   (LOCALLY
     (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 6)))
     (LOCALLY
       (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 1)))
       (LOCALLY
         (DECLARE (PROJECTION-DIMENSIONS ?o ?s ?y))
         (JOIN
          (agp (id #:AGP-1581)
               {?s <http://dbpedia.org/property/country> ?o :DIMENSIONS (?::s #:constant1579 ?::o)})
          (agp (id #:AGP-1582)
               {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y :DIMENSIONS (?::o #:constant1580
?::y)})))))))
 '(?o ?s ?y))
```

''

interpret constraint

```
(PROJECT
 (LOCALLY
   (DECLARE (REFERENCE-DIMENSIONS ?y ?s ?o))
   (LOCALLY
     (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 6)))
     (LOCALLY
       (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 1)))
       (LOCALLY
         (DECLARE (PROJECTION-DIMENSIONS ?o ?s ?y))
         (JOIN
          (agp (id #:AGP-1581)
               {?s <http://dbpedia.org/property/country> ?o :DIMENSIONS (?::s #:constant1579 ?::o)})
          (agp (id #:AGP-1582)
               {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y :DIMENSIONS (?::o #:constant1580 ?::y)})))))))
'(?o ?s ?y))
```

```
(defmethod macroexpand-bgp-phase ((phase (eql :statement-combinations)) body env)

    (let* ((slice (rest (assoc 'spocq.a:|slice| body)))
           ;;;...
           (version-constraints (declaration-information 'spocq.e::version-constraint env))

`(make-agp :body (append agp-binds agp-filters patterns)
           ;;; …
           :temporal-binds ',temporal-binds
           ;;; …
           ))


(defmethod compute-lmdb-statement-lambda ((repository lmdb-repository) (body list) &key
            ;;; …
 (if version-constraints
     `(rlmdb::map-repository-statements-filtered #',lmdb-continuation transaction
                                                 ,quad-pattern #'version-filter)
     `(rlmdb:map-repository-statements #',lmdb-continuation transaction
                                       ,quad-pattern
                                       :first min-revision-ordinal
                                       :last max-revision-ordinal)))
```