

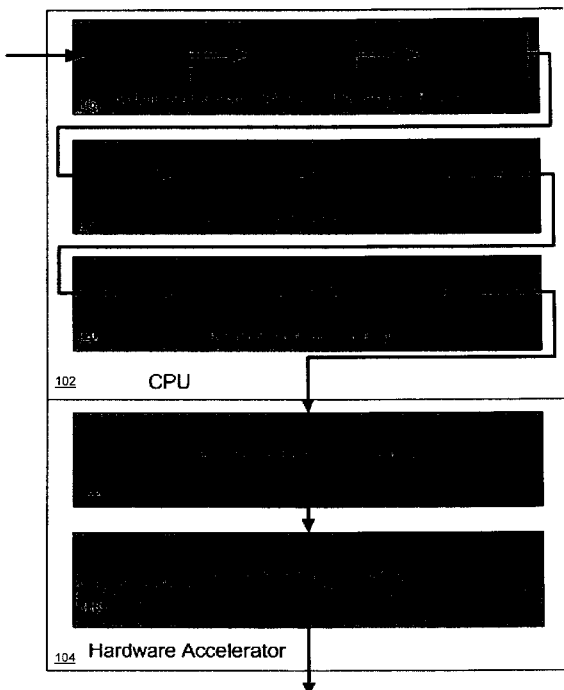


- (51) International Patent Classification:
G06F 17/30 (2006.01)
- (21) International Application Number:
PCT/MY2013/000246
- (22) International Filing Date:
10 December 2013 (10.12.2013)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
PI 2012005468 17 December 2012 (17.12.2012) MY
- (71) Applicant: **MIMOS BERHAD** [MY/MY]; Technology Park of Malaysia, Bukit Jalil, 57000 Kuala Lumpur (MY).
- (72) Inventors: **CHONG, Poh Kit**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, 57000 Kuala Lumpur (MY). **KARUPPIAH, Ettikan Kandasamy A/L**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, 57000 Kuala Lumpur (MY). **YONG, Keh Kok**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, 57000 Kuala Lumpur (MY).
- (74) Agent: **HEMINGWAY, Christopher Paul**; Marks & Clerk (Malaysia) Sdn Bhd, Lot 5B Level 5, Tower 2 Etika Twins, 11 Jalan Pinang, 50450 Kuala Lumpur (MY).
- (81) Designated States (*unless otherwise indicated, for every kind of national protection available*): 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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (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,

[Continued on next page]

(54) Title: A METHOD OF ACCELERATED SPARQL QUERY PROCESSING

Figure 1



(57) Abstract: A method of SPARQL query processing comprising the steps of parsing a SPARQL query to form one or more statement lists, expanding the statement lists into temporary tables of results, and merging the temporary tables of results at their intersections to generate output variable results.



TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG). **Published:**

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

A METHOD OF ACCELERATED SPARQL QUERY PROCESSING

Field of Invention

The invention relates to a method of accelerated SPARQL query processing.

5

Background

According to the prior art, SPARQL queries are typically translated into multiple SQL queries by a semantics engine so they can be sent to an SQL database in order to retrieve the data in RDF format. However, this can lead to highly inefficient
10 behaviour due to inefficient scheduling or collisions, in that the server can timeout if there are too many queries to process.

In addition, the prior art method is not suitable for use in massively parallel computational accelerators such as GPGPUs.

15

An aim of the invention is to provide a method of SPARQL query processing which overcomes one or more of the above issues.

Summary of Invention

In an aspect of the invention, there is provided a method of SPARQL query processing comprising the following steps:

- parsing a SPARQL query to form one or more statement lists;
- expanding the statement lists into temporary tables of results; and
- 5 merging the temporary tables of results at their intersections to generate output variable results.

In one embodiment the output variables for the SPARQL query are identified and the corresponding statements are mapped to a binary form. Typically the statements are
10 classified based on the number of variables.

In one embodiment the statements are ordered from those with the least variables to those with the most variables. Typically a previously unselected statement is selected as a start point.

15

In one embodiment the method includes a matching process where each variable in the statement that is not the final desired output variable is matched to another variable that uses the variable as an input, until the final output variable is reached wherein the statements' order is recorded as an ordered lists. Typically the matching
20 process is repeated until all combinations have been selected.

In one embodiment the ordered lists are grouped according to the final output variables. Typically the ordered lists are merged at common nodes to form trees with

each output variable as a root node. Typically the trees of different output variables are merged to obtain the relationship between the output variables

In one embodiment closest paths are found between all the output variables to
5 determine intersection positions where temporary tables need to be created and stored to enable relationship formation between the output variables at the end of the search.

In one embodiment the relationship between the output variables is defined as a
10 relationship tree wherein one or more leaf nodes are at the top of the tree opposite a root node.

In one embodiment, unprocessed leaf nodes in the relationship tree are selected and the queries therefrom are used to perform a parallel search via a computation
15 acceleration device and generate intermediate results. Typically the computation acceleration device includes a GPGPU.

In one embodiment the intermediate results from each node are recursively used as inputs to other nodes along the path from the leaf to the root until the root or an
20 intersection before the root node is reached.

In one embodiment the intermediate results from the intersection points up to the root nodes are retained and the search is repeated until all leaf nodes have been processed.

In one embodiment the intermediate result tables from the intersection points up to the root nodes are logically combined by parallel matching the inputs and outputs for all the queries.

5

In one embodiment the results matching the output variables from the tables are extracted and returned to the user.

In a further aspect of the invention there is provided a system for processing
10 SPARQL queries, comprising:

a Statement Parser for processing a SPARQL query into statements;

a Statement Table Mapper Unit for mapping the statements to a binary form;

a Statement Classifier Unit for classifying the statements so as to form linked
nodes for searching;

15 a Query Sorting Unit for sorting the statements based on the number of
variables;

a Query Flow Linking Unit for linking outputs from the statements to inputs
of other statements;

a Common Output Identifier Unit for identifying linked lists of statements
20 with common output variables;

a Common Variable Merger Unit that for forming a tree from related linked
lists;

a Parallelized Query Execution Engine for executing searches in parallel on leaf nodes within the tree; and

a Table Matching and Output Engine for matching the results of the searches, in the form of temporary tables of results, to form a final matched output.

5

Brief Description of Drawings

It will be convenient to further describe the present invention with respect to the accompanying drawings that illustrate possible arrangements of the invention. Other
10 arrangements of the invention are possible, and consequently the particularity of the accompanying drawings is not to be understood as superseding the generality of the preceding description of the invention.

Figure 1 is a schematic view of the system for SPARQL query processing according
15 to an embodiment of the invention.

Figure 2 is a schematic view of the Output Identification & Statement Classification Engine.

20 Figure 3 illustrates the Query Logics Reordering Engine: (a) in schematic view; (b) as a process flow; and (c) with an example query.

Figure 4 illustrates the Relationship Formation Engine: (a) in schematic view; (b) with an example of two ordered lists being merged in a relationship tree.

Figure 5 illustrates the Parallelized Query Execution Engine: (a) in schematic view; (b) with an example of searching at each node level being done in parallel.

5 Figure 6 is a schematic view of the Table Matching and Output Engine.

Detailed Description

10 With reference to Figure 1, there is illustrated a summary of the system for accelerating SPARQL query processing according to an embodiment of the invention.

The SPARQL query is initially processed by the CPU 102 before being parallel
15 processed by a hardware accelerator such as a GPGPU 104.

In the CPU the SPARQL query is processed by an Output Identification & Statement Classification Engine 106 comprising a Statement Parser 108 for processing a SPARQL statement to a form easily used for parallel searching, a Statement Table
20 Mapper Unit 110 for mapping the statements to a binary number, and a Statement Classifier Unit 112 for classifying the statements so as to form linked nodes for searching.

The statements are passed to a Query Logics Reordering Engine 114 comprising a Query Sorting Unit 116 for differentiating the queries based on the number of variables and a Query Flow Linking Unit 118 for linking outputs from statements to inputs of other statements.

5

The statements, now ordered into lists, are then passed to a Relationship Formation Engine 120 comprising a Common Output Identifier Unit 122 for identifying linked list of nodes with common output variables, and a Common Variable Merger Unit 124 for forming a tree from related linked lists.

10

In the GPGPU 104, there is provided a Parallelized Query Execution Engine 126 for executing searches in parallel, and a Table Matching and Output Engine 128 for matching temporary intermediate output tables to form a final matched output.

15 With reference to Figure 2, the Output Identification & Statement Classification Engine is illustrated in more detail, wherein output variables for the SPARQL query are identified 201. The corresponding SPARQL query statements are then mapped 202 to a binary form with 0 representing a pattern and 1 representing a variable and each bit corresponding to a respective subject, predicate, and object. The statements
20 are then classified 203 based on the number of variables ('1's) in them.

With regard to Figure 3a, the Query Logics Reordering Engine is illustrated in more detail, wherein a previously unselected statement is selected 301 as a start point and

the statements are ordered from those with the least variables to those with the most. Each variable in the statement that is not the final desired output variable is matched 302 to another variable that uses the variable as an input, and this is repeated 303 until the final output variable is reached wherein the statements' order is recorded as 5 a list. This process is repeated 304 until all combinations have been selected.

Figure 3b illustrates the process flow through the Query Logics Reordering Engine, starting with a check 306 for unselected statements. If found, the statement with the lowest variables is selected 308 and checked 310 for multiple outputs. If multiple 10 outputs are found the point of diversion and untraversed path is marked 312.

The statement is then checked 314 for final output variable. If present, a linked list is created 316, with the first node representing the first statement and the last node representing the final statement. If a check 318 for any diversion points being 15 untraversed is positive, return 320 to the nearest untraversed point.

If the check 314 shows no final output variable in the statement, then the statement with intermediate output variable as input is found 322 and the process loops until the final output variable is reached.

20

The procedure is repeated until all combinations have been selected

Figure 3c illustrates an example of statements being ordered and recorded as lists.

For example, the query below:

```
SELECT ?subject ?predicate ?object
5    WHERE {
    ?major rdfs:subClassOf mohl:MASCO.

    ?subMinor rdfs:subClassOf ?minor.
    ?minor rdfs:subClassOf ?subMajor.
10   ?subMajor rdfs:subClassOf ?major.
    ?subject rdf:type ?subMinor.
    ?subject mohl:hasName 'Lawyer'@en.

    ?subject ?predicate ?object.
15 }
```

becomes:

```
List1
20   ?major rdfs:subClassOf mohl:MASCO.
    ?subMajor rdfs:subClassOf ?major.
    ?minor rdfs:subClassOf ?subMajor.
    ?subMinor rdfs:subClassOf ?minor.
```

?subject rdf:type ?subMinor.

?subject ?predicate ?object.

List2

5 ?subject mohl:hasName 'Lawyer'@en.

?subject ?predicate ?object.

With respect to Figures 4a-b the Relationship Formation Engine is illustrated in more detail, wherein the ordered lists are grouped 401 according to the final output variables, the ordered lists (List1 and List2) are merged 402 at common nodes (in this case the node where the variable is '?subject') to form trees with each output variable as a root node, and the trees of different output variables are merged 403 to obtain the relationship between the output variables. The closest paths are then found 404 between all the output variables to determine intersection positions where temporary tables need to be created and stored to enable relationship formation between the output variables at the end of the search.

With regard to Figures 5a-b the Parallelized Query Execution Engine is illustrated in more detail wherein unprocessed leaf nodes in the relationship tree (the leaf nodes being at the top of the tree opposite the root) are selected 501 and the queries therefrom are used to perform a parallel search via the computation acceleration device (GPGPU). The intermediate results from each node are recursively used 502 as inputs to other nodes along the path from the leaf to the root until the root or an

intersection before the root node is reached. The parallel search result tables from the intersection points up to the root nodes are kept 503 and the process is repeated until all leaf nodes have been processed.

- 5 With regard to Figure 6 Table Matching and Output Engine is illustrated in more detail wherein the intermediate result tables from the intersection points up to the root nodes are logically combined 601 by parallel matching 602 the inputs and outputs for all the queries. Results matching the output variables from the tables are extracted and returned 603 to the CPU for display.

10

It will be appreciated by persons skilled in the art that the present invention may also include further additional modifications made to the device which does not affect the overall functioning of the device.

CLAIMS

1. A method of SPARQL query processing comprising the following steps:
parsing a SPARQL query to form one or more statement lists;
5 expanding the statement lists into temporary tables of results; and
merging the temporary tables of results at their intersections to generate
output variable results.
2. A method according to claim 1 wherein the output variables for the SPARQL
10 query are identified and the corresponding statements are mapped to a binary
form and classified based on the number of variables therein.
3. A method according to claim 2 wherein the statements are ordered from those
with the least variables to those with the most variables.
- 15 4. A method according to claim 2 or 3 including a matching process where each
variable in the statement that is not the final desired output variable is matched
to another variable that uses the variable as an input, until the final output
variable is reached wherein the statements' order is recorded as an ordered
20 lists.

5. A method according to claim 4 wherein the ordered lists are grouped according to the final output variables, and merged at common nodes to form trees with each output variable as a root node.
- 5 6. A method according to claim 5 wherein the trees of different output variables are merged to obtain the relationship between the output variables
7. A method according to claim 6 wherein the closest paths are found between all the output variables to determine intersection positions where the temporary
10 tables need to be created and stored to enable relationship formation between the output variables at the end of the search.
8. A method according to claim 7 wherein the relationship between the output variables is defined as a relationship tree in which one or more leaf nodes are
15 at the top of the tree opposite a root node.
9. A method according to claim 8 wherein unprocessed leaf nodes in the relationship tree are selected and the queries therefrom are used to perform a parallel search via a computation acceleration device to generate the temporary
20 tables of results.

10. A method according to claim 9 wherein the temporary tables from each node are recursively used as inputs to other nodes along the path from the leaf to the root until the root or an intersection before the root node is reached.
- 5 11. A method according to claim 10 wherein the temporary tables from the intersection points up to the root nodes are retained and the search is repeated until all leaf nodes have been processed.
12. A method according to claim 11 wherein the temporary tables from the
10 intersection points up to the root nodes are logically combined by parallel matching the inputs and outputs for all the queries.
13. A method according to claim 12 wherein the results matching the output variables from the temporary tables are extracted and returned to a user.
- 15
14. A system for processing SPARQL queries, comprising:
- a Statement Parser for processing a SPARQL query into statements;
 - a Statement Table Mapper Unit for mapping the statements to a binary
form;
 - 20 a Statement Classifier Unit for classifying the statements so as to form linked nodes for searching;
 - a Query Sorting Unit for sorting the statements based on the number of variables;

a Query Flow Linking Unit for linking outputs from the statements to inputs of other statements;

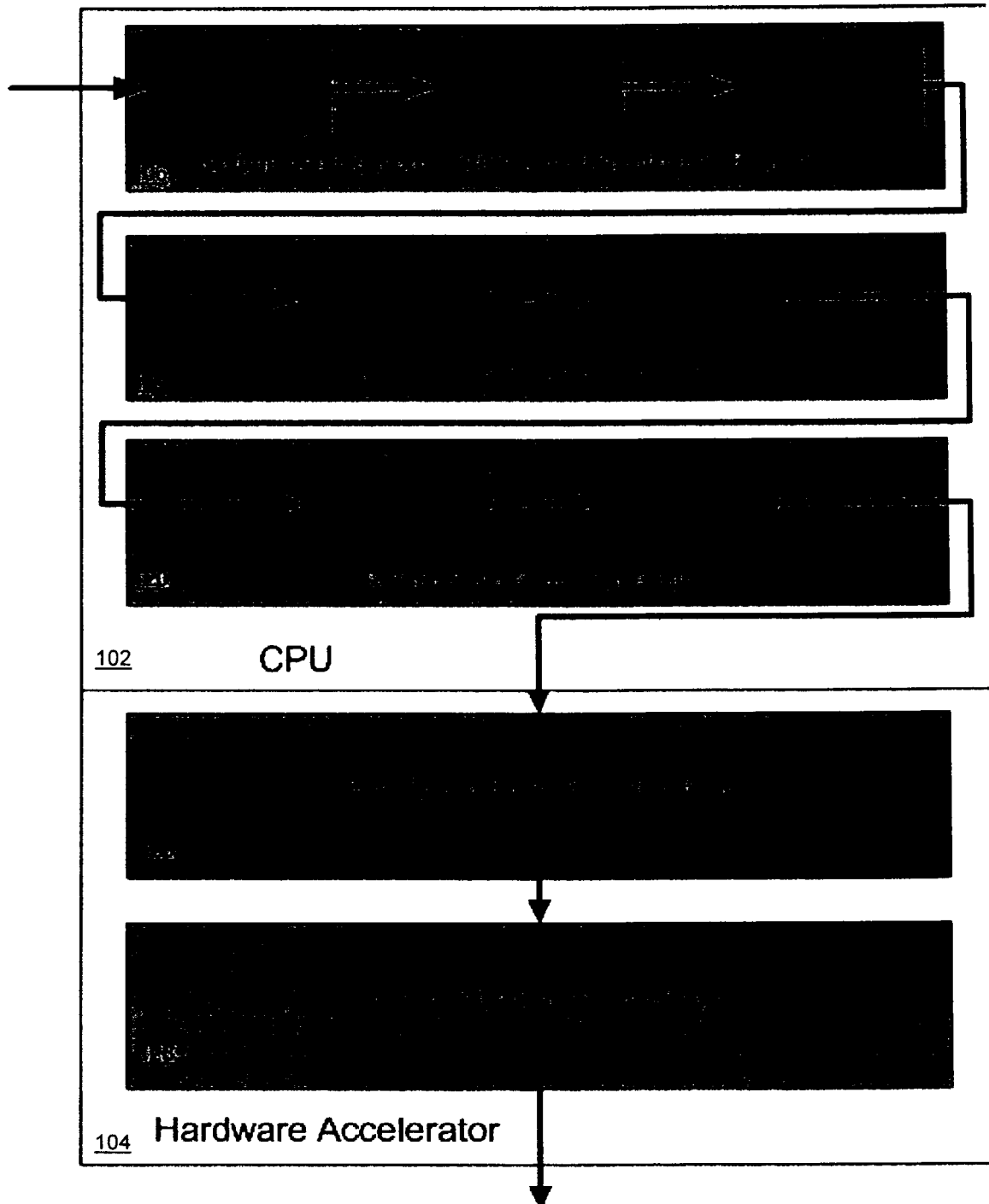
a Common Output Identifier Unit for identifying linked lists of statements with common output variables;

5 a Common Variable Merger Unit that for forming a tree from related linked lists;

a Parallelized Query Execution Engine for executing searches in parallel on leaf nodes within the tree; and

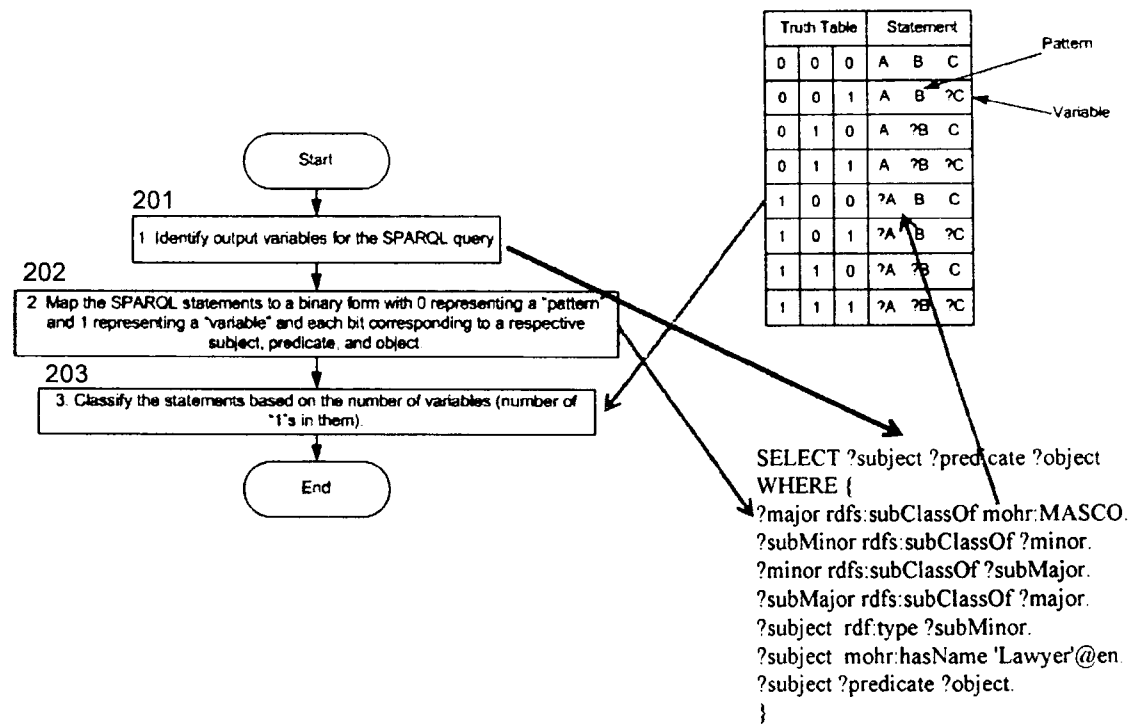
10 a Table Matching and Output Engine for matching the results of the searches, in the form of temporary tables of results, to form a final matched output.

1/10

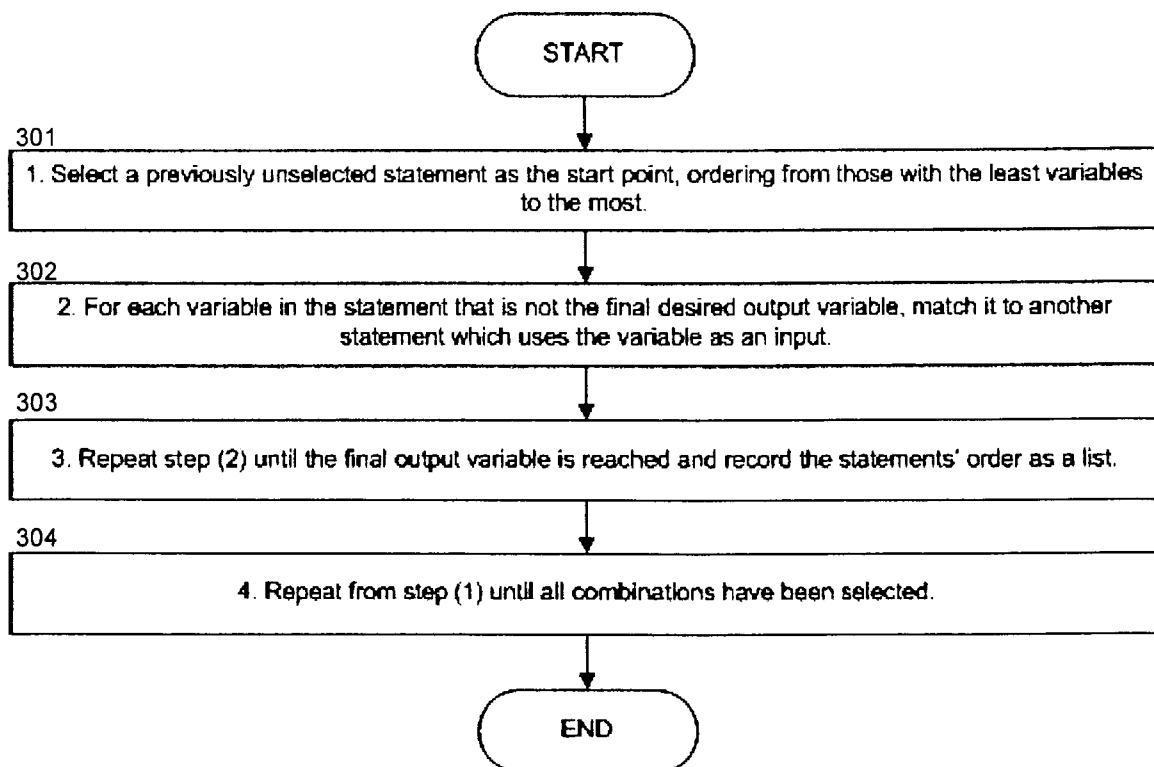
Figure 1

2/10

Figure 2

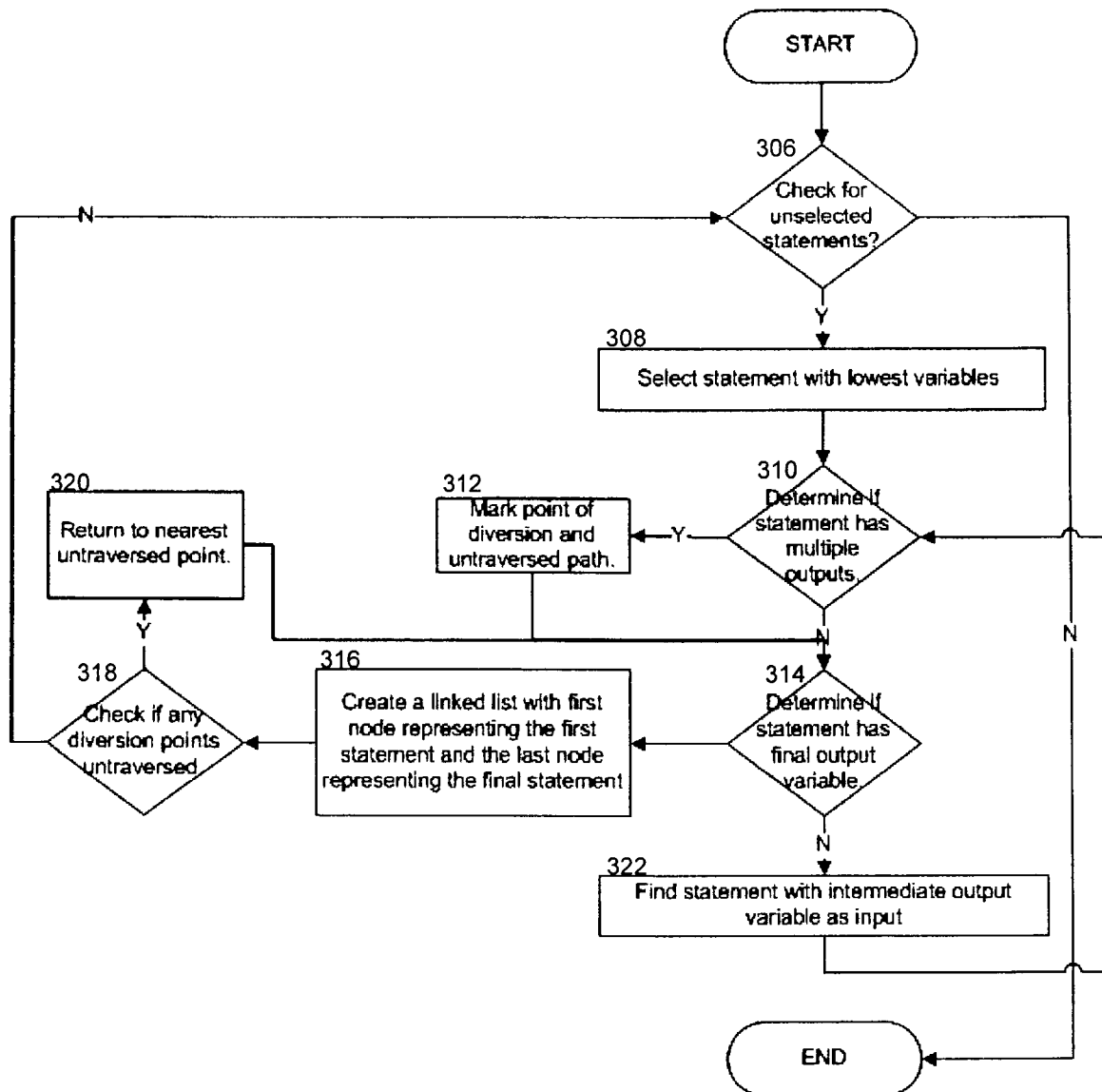


3/10

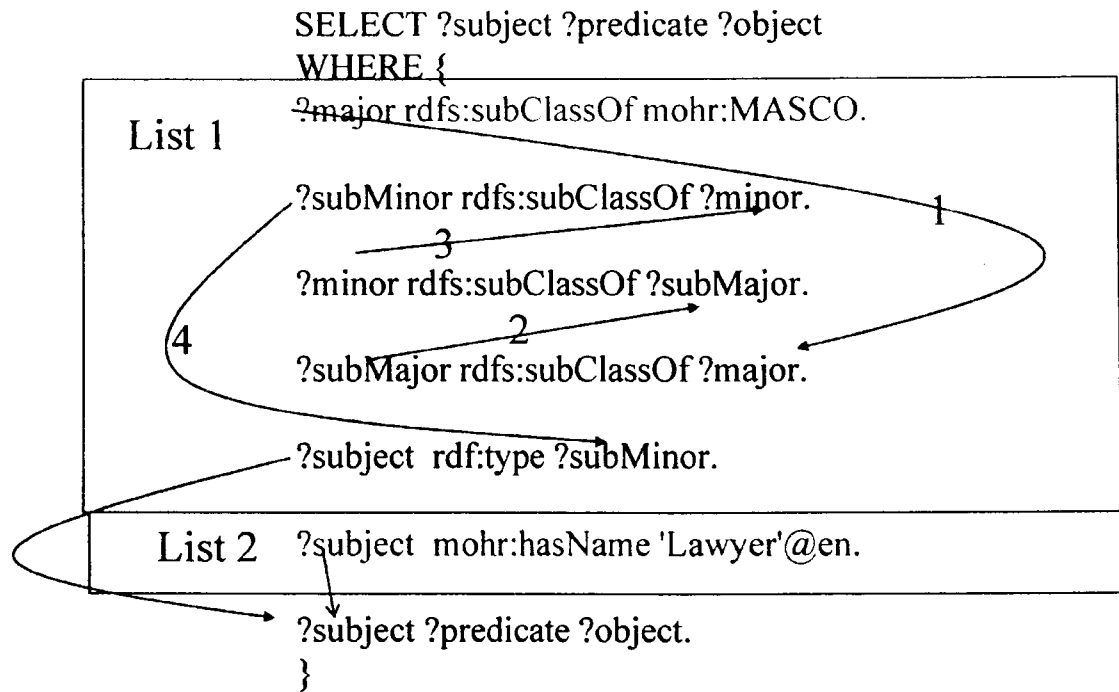
Figure 3a

4/10

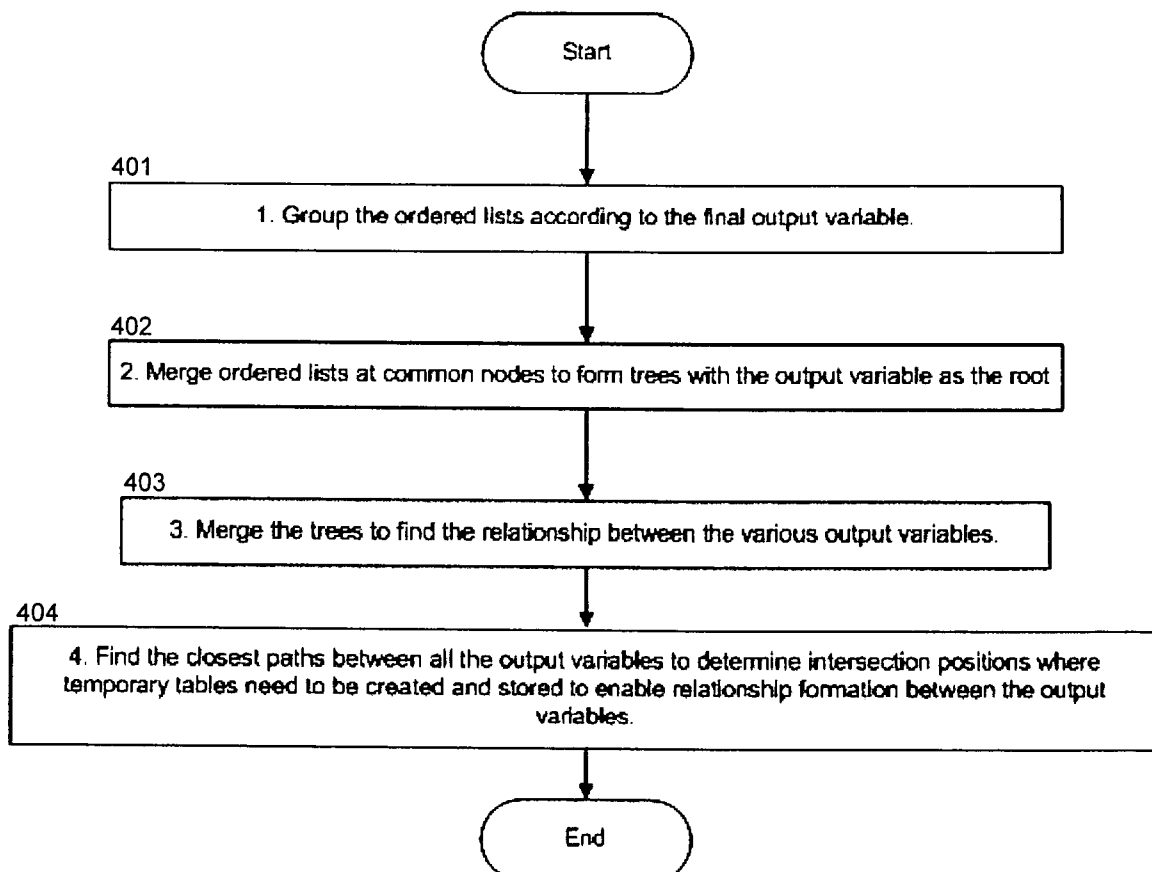
Figure 3b



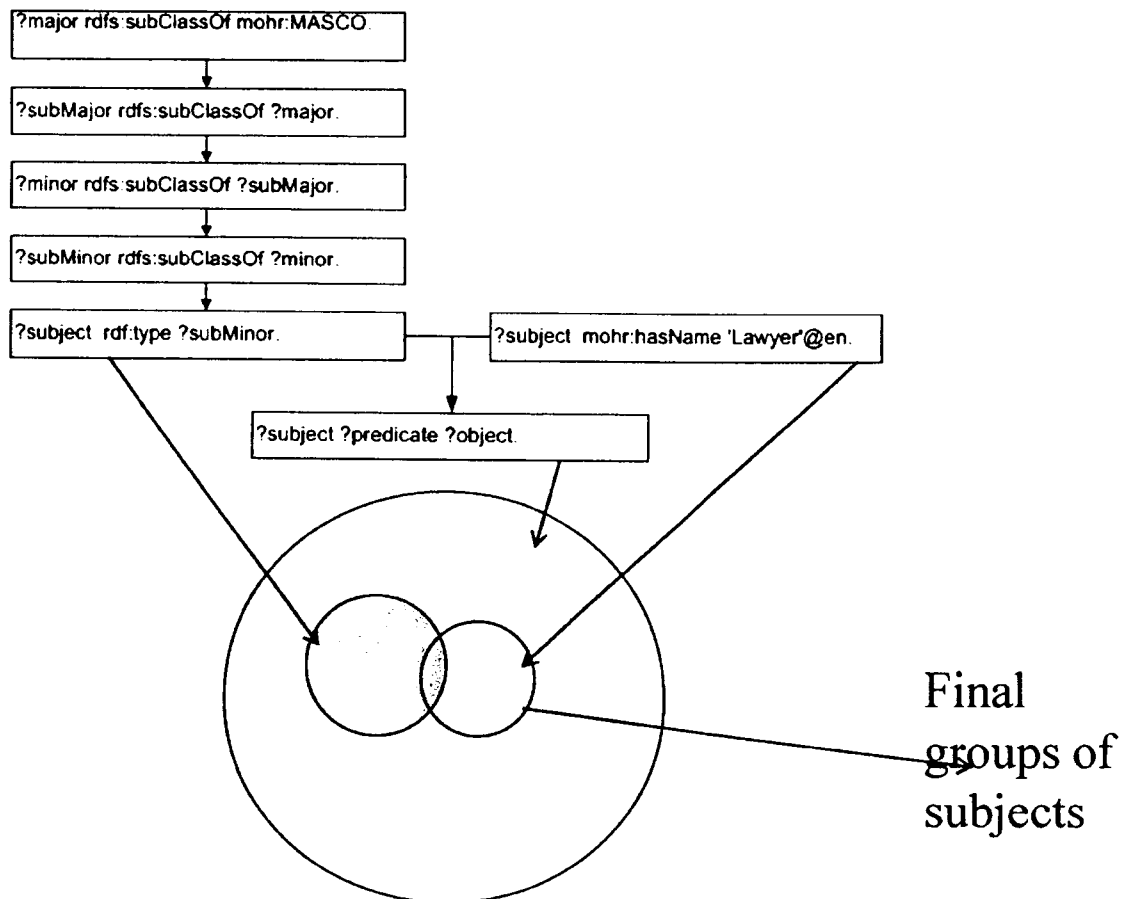
5/10

Figure 3c

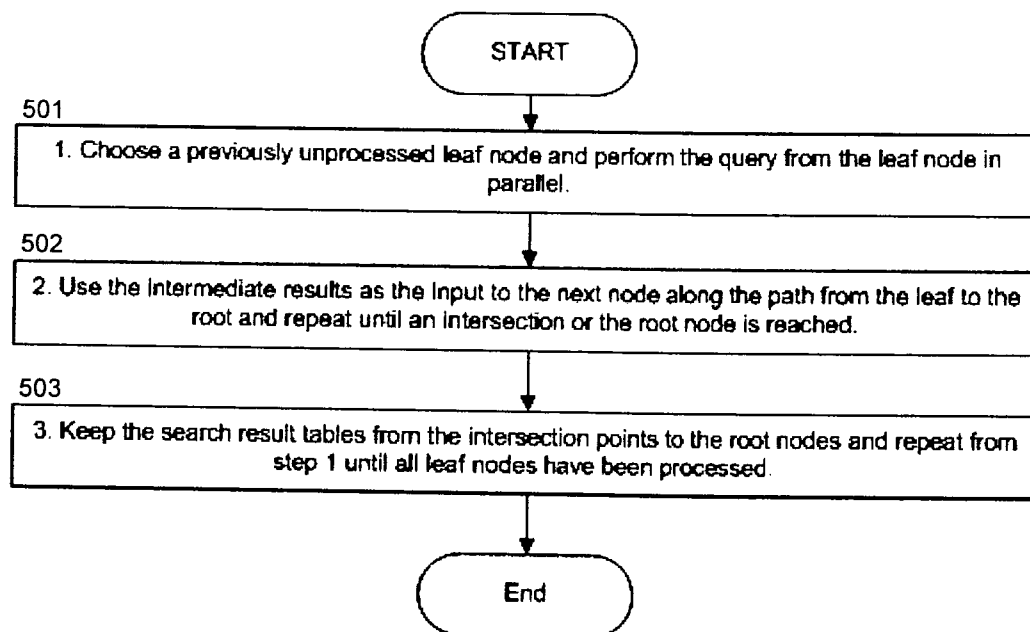
6/10

Figure 4a

7/10

Figure 4b

8/10

Figure 5a

9/10

Figure 5b

Searching at each node level is done in parallel

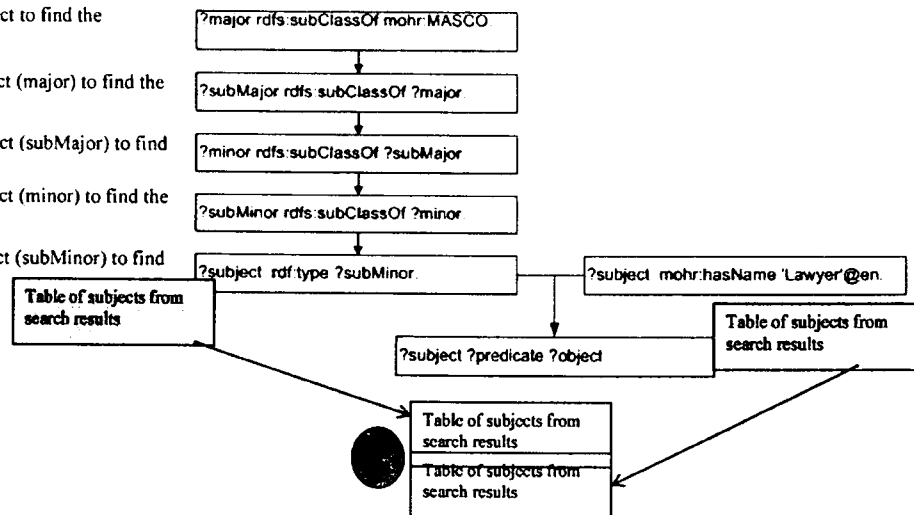
Match predicate & object to find the subject (major)

Match predicate & object (major) to find the subject (subMajor)

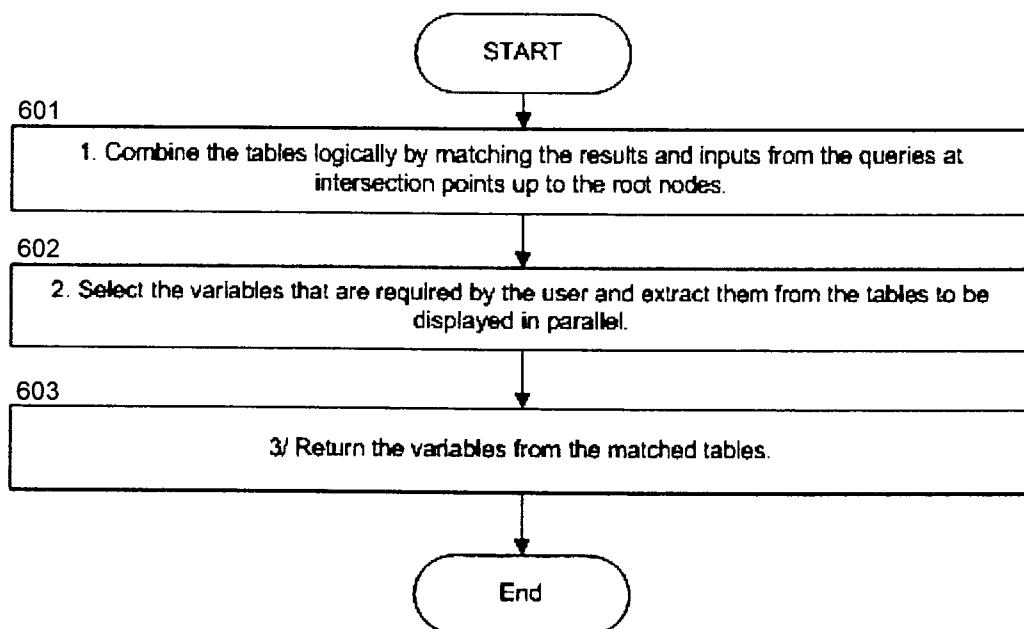
Match predicate & object (subMajor) to find the subject (minor)

Match predicate & object (minor) to find the subject (subMinor)

Match predicate & object (subMinor) to find the subject (subject)



10/10

Figure 6

INTERNATIONAL SEARCH REPORT

International application No

PCT/MY2013/000246

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30
ADD.

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

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>KATJA HOSE ET AL: "Database Foundations for Scalable RDF Processing", 23 August 2011 (2011-08-23), REASONING WEB. SEMANTIC TECHNOLOGIES FOR THE WEB OF DATA, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 202 - 249, XP019161804, ISBN: 978-3-642-23031-8 page 203, paragraph 1 - page 205, last paragraph page 218, paragraph 4 - page 219, paragraph 4 page 220, paragraph 1 - page 221, paragraph 2 page 222, paragraph 5 - last paragraph page 224, line 1</p> <p>----- -/--</p>	1-14



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

"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

23 May 2014

Date of mailing of the international search report

04/06/2014

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Herri, Edmond

INTERNATIONAL SEARCH REPORT

International application No
PCT/MY2013/000246

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JÜRG SENN: "Parallel Join Processing on Graphics Processors for the Resource Description Framework", 23RD INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS (ARCS), 23 February 2010 (2010-02-23), XP055116829, abstract -----	1-14
A	Anonymous: "General-purpose computing on graphics processing units - Wikipedia, the free encyclopedia", 29 November 2012 (2012-11-29), XP055117346, Retrieved from the Internet: URL: http://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&oldid=525538885 [retrieved on 2014-05-12] page 1, paragraph 1 -----	1-14
A	OLAF HARTIG ET AL: "The SPARQL Query Graph Model for Query Optimization", 3 June 2007 (2007-06-03), THE SEMANTIC WEB: RESEARCH AND APPLICATIONS; [LECTURE NOTES IN COMPUTER SCIENCE;;LNCS], SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 564 - 578, XP019094639, ISBN: 978-3-540-72666-1 page 565 - page 570 -----	1-14
A	"CHAPTER 19: Algorithms for Query Processing and Optimization ED - Ramez Elmasri; Shamkant B Navathe (eds)", 1 January 2011 (2011-01-01), FUNDAMENTALS OF DATABASE SYSTEMS (SIXTH EDITION), ADDISON-WESLEY, PAGE(S) 679 - 725, XP009171423, ISBN: 978-0-13-608620-8 page 710, paragraph 2 -----	1-14