WoS Query Partitioner: A Tool to Retrieve Very Large Numbers of Items From the *Web of Science* Using Different Source-Based Partitioning Approaches

Sergio Alonso
Software Engineering Department, University of Granada, Granada, Spain. E-mail: zerjioi@ugr.es

Francisco Javier Cabrerizo
Department of Software Engineering and Computer Systems, Distance Learning University, Spain. E-mail: cabrerizo@issi.uned.es

Enrique Herrera-Viedma and Francisco Herrera
Computer Science and Artificial Intelligence Department, University of Granada, Granada, Spain. E-mail: {viedma, herrera}@decsai.ugr.es

Thomson Reuters' *Web of Science* (*WoS*) is undoubtedly a great tool for scientiometrics purposes. It allows one to retrieve and compute different measures such as the total number of papers that satisfy a particular condition; however, it also is well known that this tool imposes several different restrictions that make obtaining certain results difficult. One of those constraints is that the tool does not offer the total count of documents in a dataset if it is larger than 100,000 items. In this article, we propose and analyze different approaches that involve partitioning the search space (using the *Source* field) to retrieve item counts for very large datasets from the *WoS*. The proposed techniques improve previous approaches: They do not need any extra information about the retrieved dataset (thus allowing completely automatic procedures to retrieve the results), they are designed to avoid many of the restrictions imposed by the *WoS*, and they can be easily applied to almost any query. Finally, a description of *WoS Query Partitioner*, a freely available and online interactive tool that implements those techniques, is presented.

Introduction

Thomson Reuters' *Web of Science* (*WoS*) (E-Resource-1, 2010) is one of the online resources that allows the retrieval and computation of different bibliometric measures and indicators (Archambault, Campbell, Gingras, & Larivire, 2009) such as the total numbers of papers that satisfy a particular condition or even more complex indices such as the h index (Alonso, Cabrerizo, Herrera-Viedma, & Herrera, 2009; Egghe, 2010; Meho & Rogers, 2008). However, this tool imposes several restrictions to its users:

- One of the most well-known restrictions is that obtained datasets with more than 100,000 items cannot be fully retrieved, and even more inconvenient, the total number of items in the dataset is hidden (marked as >100,000) in the results count field. Queries that deal with the scientific production of countries or research institutions and universities (Vieira & Gomes, 2009) may be affected by this restriction.
- The maximum number of terms separated by Boolean operators that can be used in a query is 50. Although this limit may seem quite high, in certain situations where queries are chained it may be reached, thus impeding certain searches.
- The Search History (i.e., the list of all the queries created during a session) has a limit of 100 queries. Altough this seems to be a quite generosus limit, it could be reached in some complicated queries series.

These restrictions make difficult to obtain accurate results for apparently easy queries. For example, to retrieve the number of items in the *WoS* for a particular country and year, a simple query can be designed in the form: *PY = year AND CU = country*, where *PY* and *CU* stand for *Publication Year* and *Country*, respectively. However, for certain countries with high production rates, the total count of such a query is greater than 100,000 and the result thus cannot be directly computed with such a simple query (Arencibia-Jorge, Leydesdorff, Chinchilla-Rodríguez, Rousseau, & Paris, 2009; Jacsó, 2009; Zhou & Leydesdorff, 2006).

In fact, in the specialized literature, we can find different approaches to overcome this kind of problem. For example,

Received February 4, 2010; revised March 16, 2010; accepted March 16, 2010

 DOI: 10.1002/asi.21360

in a recent contribution by Arencibia-Jorge et al. (2009), this issue is tackled for the particular case of the production of the United States of America and the year 2007. The query *PY = 2007 AND CU = USA* produces more than 100,000 results, and the authors thus propose a particular way to obtain an accurate result. Their main idea is to stablish a partition mainly based in the *Source* field (SO) to obtain partial results (with <100,000 results) and then to sum them up to obtain the final result (Arencibia-Jorge et al., 2009). However, this approach presents an additional difficulty: The *Source* field may offer results not only for the name of a particular source but also for the possible series in which the source may be included. This leads to the repetition of some of the results in some executed subqueries. To overcome this problem, the authors created an additional statement that finds the number of overlapping elements which is substracted from the total count to obtain the real result for the original "simple" query. In Table 1, the used queries are presented along with their results and the final total number of papers ($n = 500{,}885$). Note that the results differ from those in the study by Arencibia et al. because of updates in the *WoS* database.

This approach certainly solves the aforementioned problem, but it presents several shortcomings:

- If the results count for the overlapping query (Table 1, Query 8) is greater than 100,000 elements, the procedure cannot be applied. Note that for a query with approximately 500,000 results, around 23,000 results appeared multiple times in the subqueries. Although it is difficult to extrapolate, in a query with a several-times-larger results count, there may be too many overlapping elements.
- The overlapping query groups all previous subqueries in pairs; thus, that query grows in size quite fast. In fact, the number of terms of that query is $nTerms = nQueries \times (nQueries - 1)$. With eight or more subqueries (Any query with >700,000 results requires at least eight subqueries.), we would face the restriction of a maximum of 50 terms per query. However, and to be fair, a quite large overlapping query could be easily split into several overlapping subqueries and finally joined together. It seems that the *WoS* does a kind of simplification of queries; thus, in particular cases, a large overlapping query can be used, but it cannot be assured when this will happen as it depends on the original query.
- The overlapping query is effective because an item can be retrieved just by the name of the source *or* the name of the series of the source. However, if the policy of the *WoS* changes and, for example, it returns items also based on the name of a subseries (i.e., a result could be retrieved in three different subqueries), the overlapping query has to be adapted, and it would be probably much more complex. Although currently this fact does not apply, we cannot assure that this strategy will still work in future versions of the *WoS*.
- Although the main partition of the search space is made by restricting the *Source* field, the authors had to use additional information (Table 1, Queries 6 and 7) because there are more than 100,000 results for sources that begin with the letter *J* (Note that there are many publications whose titles begin with the word *Journal*.) Although using additional information about the problem is legitimate, doing so implies that the method cannot be extrapolated directly to any query, as no extra information might be available. For example, for the query *TS = cancer* (i.e., the number of papers that mention cancer in the whole *WoS* database), there is no easy extra information that helps us to avoid the more than 100,000 results for sources that begin with the letter *J*.
- Although the total number of queries that the authors use to compute the final solution is quite low (just eight from the minimum six queries that would be necessary for a results count of >500,000), it is clear that the design of those statements requires much effort and trials to simplify them. This work has probably been done manually as there is nothing that suggests the employment of any automatic system to generate them. This implies that a quite large amount of time has been spent in obtaining the queries. A more structured algorithm could allow the generation of those queries in an automated manner, avoiding the time consumed in this task.

In this contribution, we present two more general and automatic source-based partitioning approaches that use a simple mechanism to avoid the use of the overlapping query proposed by Arencibia-Jorge et al. (2009). Thus, both help to obtain very large results counts for almost any query from the *WoS* with little effort and in an efficient manner:

- The first approach consists of a *divide-and-conquer* strategy, which recursively splits a large query into two or more subqueries by restricting their results counts by imposing conditions on the *source* field. This approach is designed to obtain the results with as few queries as possible, but cannot be always applied:
  (a) It may come up with the 50 terms per query restriction for some large result sets, and
  (b) it presents a theoretical limit of a maximum 10-million results for a single query.
- The second approach also is a divide-and-conquer strategy in which a collapsing mechanism is applied to simplify the subqueries. In this case, the number of subqueries may be higher than the first approach (i.e., it is slightly more inefficient), but it allows the execution of almost any query (i.e., it will not usually suffer the 50-terms nor the the 10-million results restrictions), thus being a more reliable strategy.

Both approaches can be incorporated in automatic retrieval systems, thus minimizing the efforts of the user. In fact, we have developed the *WoS Query Partitioner*, a freely available and online interactive tool which implements both approaches. This tool has been used to test the partitioning approaches with the previously presented query *PY = 2007 AND CU = USA*, obtaining the same final result, and also to obtain the number of items of more complicated queries such as *TS = cancer or PY = 2007* (i.e., the number of papers published in 2007 indexed in the *WoS* database).

To do so, the article is structured as follows. We provide an improvement over the Arencibia-Jorge et al. (2009) approach by means of the use of the NOT operator. Then, the source-based partitioning approaches that use a divide-and-conquer strategy and some of the previous ideas to automatically retrieve the results count for a particular query are presented. Next, we briefly describe the WoS Query Partitioner software,

DOI: 10.1002/asi

TABLE 1. Sequence of queries to obtain the total number of papers from the USA in 2007 according to Arencibia-Jorge et al. (2009).

	Query	Items	∑
#1	PY=2007 AND CU=USA AND (SO=A* OR SO=B*)	92,166	92,166
#2	PY=2007 AND CU=USA AND (SO=C* OR SO=D* OR SO=E* OR SO=F* OR SO=G*)	92,785	184,951
#3	PY=2007 AND CU=USA AND (SO=H* OR SO=I* OR SO=K* OR SO=L* OR SO=M*)	83,859	268,810
#4	PY=2007 AND CU=USA AND (SO=N* OR SO=O* OR SO=P* OR SO=Q* OR SO=R*)	85,187	353,997
#5	PY=2007 AND CU=USA AND (SO=S* OR SO=T* OR SO=U* OR SO=V* OR SO=W* OR SO=X* OR SO=Y* OR SO=Z* OR SO=1* OR SO=2* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9*)	60,122	414,119
#6	PY=2007 AND CU=USA AND SO=J* AND AD=CA	17,089	431,208
#7	PY=2007 AND CU=USA AND SO=J* NOT AD=CA	93,194	524,402
#8	(1 AND 2) OR (1 AND 3) OR (1 AND 4) OR (1 AND 5) OR (1 AND 6) OR (1 AND 7) OR (2 AND 3) OR (2 AND 4) OR (2 AND 5) OR (2 AND 6) OR (2 AND 7) OR (3 AND 4) OR (3 AND 5) OR (3 AND 6) OR (3 AND 7) OR (4 AND 5) OR (4 AND 6) OR (4 AND 7) OR (5 AND 6) OR (5 AND 7) OR (6 AND 7)	23,517 (Overlapping)	500,885 (∑#1. . .#7–#8)

TABLE 2. Sequence of queries to obtain the total number of papers from the USA in 2007 avoiding the overlapping query by using the NOT operator.

	Query	Items	∑
#1	PY=2007 AND CU=USA AND (SO=A* OR SO=B*)	92,166	92,166
#2	PY=2007 AND CU=USA AND (SO=C* OR SO=D* OR SO=E* OR SO=F* OR SO=G*) NOT 1	91,987	184,153
#3	PY=2007 AND CU=USA AND (SO=H* OR SO=I* OR SO=K* OR SO=L* OR SO=M*) NOT 1 NOT 2	80,952	265,105
#4	PY=2007 AND CU=USA AND (SO=N* OR SO=O* OR SO=P* OR SO=Q* OR SO=R*) NOT 1 NOT 2 NOT 3	78,963	344,068
#5	PY=2007 AND CU=USA AND (SO=S* OR SO=T* OR SO=U* OR SO=V* OR SO=W* OR SO=X* OR SO=Y* OR SO=Z* OR SO=1* OR SO=2* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9*) NOT 1 NOT 2 NOT 3 NOT 4	46,775	390,843
#6	PY=2007 AND CU=USA AND SO=J* AND AD=CA NOT 1 NOT 2 NOT 3 NOT 4 NOT 5	17,029	407,872
#7	PY=2007 AND CU=USA AND SO=J* NOT AD=CA NOT 1 NOT 2 NOT 3 NOT 4 NOT 5 NOT 6	93,013	500,885

and some conclusions are discussed. Finally, the Appendix shows the results of two larger query tests that were carried out with the WoS Query Partitioner tool.

Improving the Existing Methodology by the Use of the NOT Operator

As shown in the previous section, it is possible to determine the total number of elements in the *WoS* for a query such as *PY = 2007 AND CU = USA* by creating a partition of the search space using the *Source* field and some additional information for the particular case of the source that begin with with the letter *J*. To do so, an overlapping detection query can be used to detect when particular results may have been double-counted (because the *Source* field may return results from both the name of the publication and its possible series).

However, there is a simpler way to avoid the repetition of elements in the subqueries that would allow avoiding the overlapping query and even would allow simplifying the complexity of the proposed queries. The main idea of this improvement is to avoid in each subsequent query the repetition of the results in the previous queries. To do so, we can use the NOT operator to restrict the results to only the not previously serlected items; that is, Subquery 2 should restrict its results to the ones that have not been selected in Subquery 1 with the restriction NOT 1, Subquery 3 should restrict its results to the ones not selected in Queries 1 and 2 with the restriction NOT 1 NOT 2, and so on. Thus, the queries presented in the previous section can be rewritten as shown in Table 2, obtaining the exact same final result, but avoiding the overlapping statement (and its problems).

Note that this simple mechanism easily allows simplifying even more the queries needed to obtain the final results count. For example, with a reordering and simplification, the same result can be obtained with the queries shown in Table 3. In this case, Subquery 6 retrieves all the publications from the USA and published in 2007 that have not been previously found with Subqueries 1 to 5. This kind of "wild card" subquery is interesting because allows the simplification of queries and assures that every paper that complies with the condition is retrieved. For example, in the original Arencibias-Jorge et al. (2009) formulation, sources that begin with the number 0 are not considered (Nowadays, there are no sources that begin with 0.) However, if some publications that start with 0 are added in a *WoS* update, their proposal will not count them. On the contrary, this last

DOI: 10.1002/asi

TABLE 3. Simplified sequence of queries to obtain the total number of papers from the USA in 2007.

	Query	Items	$\sum$
#1	PY=2007 AND CU=USA AND (SO=A* OR SO=B*)	92,166	92,166
#2	PY=2007 AND CU=USA AND (SO=C* OR SO=D* OR SO=E* OR SO=F* OR SO=G*) NOT #1	91,987	184,153
#3	PY=2007 AND CU=USA AND (SO=H* OR SO=I* OR SO=K* OR SO=L* OR SO=M*) NOT #1 NOT #2	80,952	265,105
#4	PY=2007 AND CU=USA AND (SO=N* OR SO=O* OR SO=P* OR SO=Q* OR SO=R*) NOT #1 NOT #2 NOT #3	78,963	344,068
#5	PY=2007 AND CU=USA AND SO=J* NOT AD=CA NOT #1 NOT #2 NOT #3 NOT #4	93,100	437,168
#6	PY=2007 AND CU=USA NOT #1 NOT #2 NOT #3 NOT #4 NOT #5	63,717	500,885

```
 0. PendingQueries = OriginalQuery
    TotalResults = 0
    NotPart = {}
 1. WHILE PendingQueries NOT EMPTY
 2.   CurrentQuery = FIRST QUERY OF PendingQueries
 3.   APPEND NotPart TO CurrentQuery
 4.   NumberResults = EXECUTE CurrentQuery IN WoS
 5.   IF NumberResults > 100000 THEN
 6.     EXPAND CurrentQuery
 7.     NewSubqueries = SPLIT CurrentSubquery
 8.     ADD NewSubqueries TO PendingQueries
 9.   ELSE
10.     TotalResults = TotalResults + NumberResults
11.     APPEND CurrentQuery TO NotPart
12.   END IF
13.   REMOVE CurrentQuery FROM PendingQueries
14. END WHILE
15. WildCard = OriginalQuery + NotPart
16. NumberResults = EXECUTE WildCard IN WoS
17. FinalResult = TotalResults + NumberResults
```

FIG. 1. Pseudocode for the first source-based divide-and-conquer partitioning strategy.

proposal, with the addition of this wild card subquery, would count them.

Source-Based Divide-and-Conquer Partitioning Strategies

In this section, we present two different source-based partitioning approaches to obtain results counts from the *WoS* for queries larger than 100,000. Both of them do not use any other external source of information about the query, but act in a recursive way by splitting the query into different subqueries until their results counts are smaller than 100,000. Then, all partial results are summed to obtain the final results counts (a classical divide-and-conquer strategy). The first strategy is a more efficient approach (as it usually needs less subqueries to be executed), but presents some reliability issues. The second one has been adapted to overcome those reliability issues, thus achieving the final result for any given query.

Both approaches need to define at least two mechanisms to split a query: The first approach is used to expand the query using the *Source* field, and the second approach is used to really split the query into two subqueries. Next, we will describe those mechanisms and provide some examples of the use of both approaches using some real-world queries with large results counts.

Initial Description of the Strategies

Apart from the aforementioned expansion and splitting mechanisms, which will be described in the following section, the first strategy also uses the idea of restricting the results found in previous queries by incorporating negation operators (doscuused earlier) to avoid the overlapping of elements in the different subqueries and to assure that the retrieval process acquires all desired results via the use of a wild card subquery.

From now on, assume that the queries and subqueries we are using have the following form:

ORIGINAL QUERY AND SO_PART NOT_PART

where the SO_PART has a set of terms in the form (SO = Prefix1* OR ... OR SO = PrefixN*) and the NOT_PART is in the form NOT Sentence1 ... NOT SentenceM. Note that at the beginning of the process, both SO and NOT parts are empty. A pseudocode description of this approach is shown in Figure 1.

Expansion Mechanism

To be able to split a particular query into subqueries with smaller results counts, it is neccesary to expand the original

DOI: 10.1002/asi

```
 0. Terms = list of terms in the SO part of the query to split
 1. Expand = true
 2. WHILE (Expand)
 3.   Expand = false       // Exit if no expansions are done
 4.   SORT Terms           // According to the sources with its prefix
 5.   FOR EACH Term IN Terms
 6.     ExpandedTerms = GET PREFIXES THAT BEGIN WITH PREFIX OF Term
 7.      IF (SIZE OF ExpandedTerms + SIZE OF Terms - 1) < 41 THEN
 8.        REMOVE Term FROM Terms
 9.        ADD ExpandedTerms TO Terms
10.        Expand = true   // We have expanded. Try another time
11.        BREAK           // Exit the FOR loop
12.      END IF
13.   END FOR
14. END WHILE
```

FIG. 2. Pseudocode for the expansion mechanism.

query. In our proposal, this expansion will be done using the *Source* field. For example, if a query *ORIGINAL QUERY* has a results count of more than 100,000 elements, it could be expanded to an equivalent query (with the same number of results) in the form ORIGINAL AND (SO = A* OR SO = B* OR ... OR SO = 8* OR SO = 9*). The only factor that we have to take into account is that the prefixes for the SO fields are all that can be in the *WoS* database. At this point, those prefixes are the letters A to Z and the numbers 1 to 9. Once this expansion is done, the query is easily splitable just by maintaining the original query part and redistributing the SO terms into several new subqueries (discussed later).

There will be certain moments in which the expansion does not need to incorporate all possible letters and numbers. For example, if in a more deeper recursion level the query ORIGINAL QUERY AND (SO = X*) has more than 100,000 results, it could be expanded to just ORIGINAL QUERY AND (SO = X−* OR SO = XE* OR SO = XI* OR SO = XU*), as there are no sources that begin with any other combinations of an initial X and other symbols. Indeed, to avoid the expansion of many nodes that would not retrieve any result (because there are no sources with particular prefix combinations), an almost complete list of sources in the *WoS* (16,328 in total) have been downloaded from the Thomson Reuters Master Journal List (E-Resource-2, 2010). From this list, a file of possible minimum, different, and unique prefixes has been created which allows identification of each one of the sources. To this list, some extra prefixes (basically numerical prefixes) have been added to cover some sources that are not included in the Thomson Reuters Master Journal List (e.g., many conference proceedings' titles start with a number). Note that some of the sources of documents in the *WoS* may not be reflected in that list; however, the use of a final wild card subquery at the end of the retrieval process will take care of the few results that might not be covered there.

The described expansion can be repeated as many times as we want, thus increasing the granularity of the SO part of the query. Obviously, we are not interested in expanding every possible prefix because the size of the query would grow to more than 16,000 terms (which would not be allowed by the *WoS* to be executed); however, note that the granularity of the SO part is large enough to try to find partitions with a similar number of results being this number as large as possible (but not surpassing the 100,000 limit).

To find a particular balance with the granularity of the SO part of the query, two policies have been fixed:

- The maximum number of terms in the expanded SO part will be 40. This allows us to split the query into two subqueries of a maximum of 20 terms (to avoid reaching the 50-terms restriction) in the SO part.
- The expansion mechanism first tries to expand the term whose number of sources that have that particular prefix is larger. This heuristic is introduced because if a given prefix identifies a larger amount of sources than other, it will probably produce a larger results count and, thus, dividing it will probably obtain subqueries with larger and more balanced numbers of queries (which will reduce the number of needed divisions). To do so, all terms in the SO part are sorted from having more sources with that particular prefix to the one having less sources with that prefix, and the first expansion is done to the first term. For example, if we have the query ORIGINAL QUERY AND (SO = J* AND SO = X*) and we want to expand it, the first term to be expanded is SO = J* because there are more than 1,000 sources that begin with the prefix J and only 10 sources that begin with X; thus, the part that most likely is contributing more results to the query (increasing the number of results >100,000) is the first one.

A pseudocode description of the expansion mechanism is presented in Figure 2, where the GET PREFIXES THAT BEGIN WITH PREFIX OF Term function just returns all the possible larger (but with the same length) prefixes from the list of prefixes that begin with the prefix of the term. For example, if the term is SO = X*, it would return the set of terms SO = X-*, SO = XE*, SO = XI*, SO = XU*.

Split Mechanism

The split mechanism of this first approach creates two different subqueries from an expanded query. To do so, it just

DOI: 10.1002/asi

```
 0. Terms = list of terms in the SO part of the query to split
 1. Q1 = NEW QUERY  (COPY ORIGINAL QUERY)
 2. Q2 = NEW QUERY  (COPY ORIGINAL QUERY)
 3. Count = 0
 4. FOR EACH Term IN Terms
 5.   IF (Count % 2) == 0 THEN
 6.     ADD Term TO Q1
 7.   ELSE
 8.     ADD Term TO Q2
 9.   END IF
10. END FOR
```

FIG. 3. Pseudocode for the split mechanism (% is the usual modulus operation).

redistributes the elements in the SO part of the query in two different sets. As the terms in the SO part are ordered according to the number of sources with that particular prefix, the split mechanism will alternatively put each term in one of the two new subqueries to try to obtain a balanced distribution of the elements. For example, the following query ORIGINAL QUERY AND (SO = J* OR SO = A* OR SO = Y* OR SO = X*) would be split in the following two subqueries: ORIGINAL QUERY AND (SO = J* OR SO = Y*) and ORIGINAL QUERY AND (SO = A* OR SO = X*). A pseudocode description of the split mechanism is presented in Figure 3.

Examples of Application and Discussion (PY = 2007 AND CU = USA)

Several tests have been done to this first divide-and-conquer approach. In fact, we used the WoS Query Partitioner application (described earlier), which interactively produces the queries and subqueries that have to be executed through the *WoS* interface. This application only requires as inputs the query from which we want to obtain the number of results and the number of results that each particular subquery introduced in the *WoS* interface produced. Thus, the task of the user is merely copying and pasting queries from the application to the *WoS* Web interface and copying and pasting the number of results from the *WoS* interface to the WoS Query Partitioner.

One of the first tests that was to try the query presented earlier, PY = 2007 AND CU = USA. The application only needed the 14 queries presented in Table 4 to obtain the exact final result. Although this number is clearly larger than the sets of queries from the previous sections, note that no extra knowledge about the problem has been introduced and that no previous queries have been tested (which is clearly not the case from the manual approaches presented earlier). In addition, the generated queries are clearly more complicated ones, but as they are not inteded to be used manually (i.e., just copied and pasted into the *WoS* interface), this is not really an issue.

In addition to the raw data, in Figure 4, the actual partition made with the subqueries is depicted. The area of each rectangle is proportional to the number of results obtained for that particular query. A green reference square has been included, with an area equivalent to 100,000 results. Thus, all leaf subqueries should have an area smaller than that of the square.

In addition, the total number of results for the much more complex query *TS = cancer* is 908155. Thirty-two different queries had to be executed in the *WoS* interface (which required about 5 min of copying and pasting). The actual queries and a graphical description for this partition can be found at http://sci2s.ugr.es/software/WoSQP/ For this particular example, note that the wild card subquery found a few results that were not retrieved by the 31 previous queries.

As has been shown, this approach is able to retrieve results counts for large datasets. In addition, it obtains the results in an automatic way, without needing any special knowledge about the problem, which allows the use of this method with almost any query. However, the presented approach does have some limitations:

- For queries with really large results counts, more than 100 subqueries might be needed. This imposes a theoretical limit of a maximum 10-million results for a single query; however, as partial results counts are usually smaller than 100,000, this limit will usually be reached earlier.
- The complexity of the generated subqueries increases as new NOT operators are included. This fact could lead to quite large queries (i.e., with >50 terms) that the *WoS* would not be able to process. In fact, this limit was reached when trying to obtain the results count for the query PY = 2007.

Thus, to solve these two problems and to allow obtaining results for queries with larger datasets, a second approach has been developed and is described next.

Source-Based Divide-and-Conquer Partitioning Strategy With Collapsing Mechanism

The second approach tries to solve the problems of the previous approach to be able to obtain results counts from almost any query from the *WoS*. The basic scheme of this approach also is based in a recursive partitioning by means of the *Source* field. However, in this case, we have adapted the expansion and split mechanisms to avoid mixing prefixes with different lengths in the SO part of the queries. This change allows

DOI: 10.1002/asi

TABLE 4. Queries produced by the first source-based partitioning approach for the query PY=2007 AND CU=USA.

	Query	Items	∑
#1	PY=2007 AND CU=USA	>100,000	
#2	PY=2007 AND CU=USA AND (SO=J* OR SO=C* OR SO=I* OR SO=S* OR SO=E* OR SO=N* OR SO=F* OR SO=H* OR SO=O* OR SO=Z* OR SO=W* OR SO=U* OR SO=Y* OR SO=X* OR SO=3* OR SO=5* OR SO=7* OR SO=9*)	>100,000	
#3	PY=2007 AND CU=USA AND (SO=JOURNAL * OR SO=I* OR SO=E* OR SO=F* OR SO=O* OR SO=W* OR SO=JA* OR SO=JU* OR SO=JE* OR SO=JI* OR SO=JOURNALS* OR SO=JN* OR SO=JOI* OR SO=3* OR SO=7* OR SO=JM* OR SO=JS* OR SO=JOG* OR SO=JOK*)	>100,000	
#4	PY=2007 AND CU=USA AND (SO=JOURNAL O* OR SO=E* OR SO=O* OR SO=JA* OR SO=JOURNAL D* OR SO=JE* OR SO=JOURNALS* OR SO=JOI* OR SO=3* OR SO=JM* OR SO=JOG* OR SO=JOURNAL I*)	>100,000	
#5	PY=2007 AND CU=USA AND (SO=E* OR SO=O* OR SO=JOURNAL OF A* OR SO=JOURNAL OF M* OR SO=JOURNAL OF S* OR SO=JOURNAL OF B* OR SO=JOURNAL OF F* OR SO=JA* OR SO=JOURNAL OF R* OR SO=JOURNAL OF L* OR SO=JOURNAL OF D* OR SO=JOURNAL D* OR SO=JOURNAL OF K* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* cumulated OR SO=JOURNALS* OR SO=JOURNAL OF X* OR SO=JM* OR SO=JOURNAL I*)	69,279	69,279
#6	PY=2007 AND CU=USA AND (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF P* OR SO=JOURNAL OF E* OR SO=JOURNAL OF N* OR SO=JOURNAL OF I* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF O* OR SO=JOURNAL OF V* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF U* OR SO=JE* OR SO=JOURNAL OF Y* OR SO=JOI* OR SO=3* OR SO=JOG*) NOT #5	69,807	139,086
#7	PY=2007 AND CU=USA AND (SO=I* OR SO=F* OR SO=W* OR SO=JOURNAL F* OR SO=JU* OR SO=JI* OR SO=JN* OR SO=JOURNAL A* OR SO=7* OR SO=JS* OR SO=JOK* OR SO=JOURNAL N*) NOT #5 NOT #6	54,345	193,431
#8	PY=2007 AND CU=USA AND (SO=C* OR SO=S* OR SO=N* OR SO=H* OR SO=Z* OR SO=U* OR SO=Y* OR SO=X* OR SO=JC* OR SO=JOURNALI* OR SO=JB* OR SO=JOA* OR SO=JOR* OR SO=5* OR SO=9* OR SO=JP* OR SO=JOE* OR SO=JOH* OR SO=JOM*) NOT #5 NOT #6 NOT #7	95,187	288,618
#9	PY=2007 AND CU=USA AND (SO=A* OR SO=P* OR SO=B* OR SO=M* OR SO=R* OR SO=T* OR SO=G* OR SO=D* OR SO=L* OR SO=V* OR SO=K* OR SO=Q* OR SO=2* OR SO=1* OR SO=4* OR SO=6* OR SO=8* OR SO=0*) NOT #5 NOT #6 NOT #7 NOT #8	>100,000	
#10	PY=2007 AND CU=USA AND (SO=A* OR SO=B* OR SO=R* OR SO=G* OR SO=L* OR SO=K* OR SO=2* OR SO=4* OR SO=8*) NOT #5 NOT #6 NOT #7 NOT #8	>100,000	
#11	PY=2007 AND CU=USA AND (SO=B* OR SO=G* OR SO=AC* OR SO=AR* OR SO=K* OR SO=AU* OR SO=AP* OR SO=AG* OR SO=AT* OR SO=AB* OR SO=AV* OR SO=AA* OR SO=AK* OR SO=A * OR SO=8*) NOT #5 NOT #6 NOT #7 NOT #8	52,522	341,140
#12	PY=2007 AND CU=USA AND (SO=R* OR SO=AN* OR SO=L* OR SO=AM* OR SO=AD* OR SO=AS* OR SO=AL* OR SO=AF* OR SO=AQ* OR SO=AI* OR SO=2* OR SO=AE* OR SO=AJ* OR SO=4* OR SO=AX*) NOT #5 NOT #6 NOT #7 NOT #8 NOT #11	75,274	416,414
#13	PY=2007 AND CU=USA AND (SO=P* OR SO=M* OR SO=T* OR SO=D* OR SO=V* OR SO=Q* OR SO=1* OR SO=6* OR SO=0*) NOT #5 NOT #6 NOT #7 NOT #8 NOT #11 NOT #12	84,471	500,885
#14	PY=2007 AND CU=USA NOT #5 NOT #6 NOT #7 NOT #8 NOT #11 NOT #12 NOT #13	0	500,885

us to incorporate a collapsing mechanism that minimizes the complexity of the next subqueries (avoiding the 50-terms limit) by collapsing certain branches of the search space. Moreover, the collapsing mechanism allows removing queries from the search history, which solves the maximum limit of 100 subqueries of the approach presented earlier.

In the following, we will describe the modified expansion and split mechanisms, the new collapsing mechanism, and some examples and discussion about the application of this new approach.

New expansion mechanism. The expansion mechanism for this approach is simpler than the previous one. In this case, we will expand the query only if there is just one term in the SO part (or none, just for the case of the first query). In doing so, we assure that all the terms in the SO part will have the same prefix.

This expansion mechanism has a small disadvantage: As there will be less terms in the SO part of the query, we will not be able to try to balance both subqueries when divided by the split mechanism; thus, more subqueries usually will be generated for the original query. In fact, as the examples will show, there will be some queries with a larger number of generated sub-queries when using this new expansion mechanism. This is not an issue because the queries could be executed in an automatic way via a specific retrieval application.

New split mechanism. As in the previous approach, this split mechanism will try to balance the number of results between the two new subqueries based on the numbers of the sources that begin with each particular prefix in the SO part. To avoid the NOT parts of the queries growing too much, it is interesting to try to maintain to the left of the search tree the SO terms which presumably will produce more results (because the distribution of journals that begin with particular perfixes is quite skewed).

Thus, this split mechanism will sort the SO terms in the SO part of the query and include on the first subquery the first

DOI: 10.1002/asi

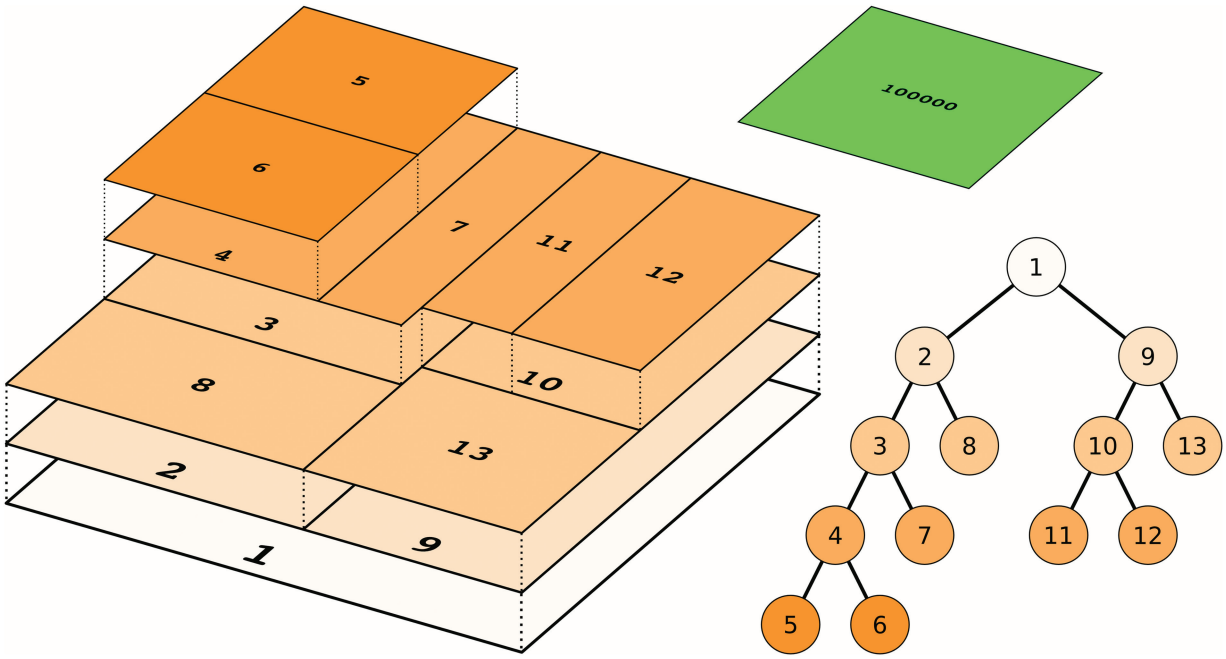

FIG. 4. Actual partition for the queries in Table 4.

```
 0. Terms = list of terms in the SO part of the query to split
    TotalNumberSources = TOTAL NUMER OF SOURCES IN Terms
 1. Q1 = NEW QUERY  (COPY ORIGINAL QUERY)
 2. Q2 = NEW QUERY  (COPY ORIGINAL QUERY)
 3. FirstTerm = FIRST TERM OF Terms
 4. ADD FirstTerm to Q1
 5. CumulatedSources = NUMBER OF SOURCES FOR FirstTerm
 6. REMOVE FirstTerm FROM Terms
 7. FOR EACH Term IN Terms
 8.   SourcesCount = NUMBER OF SOURCES FOR for Term
 9.   IF CumulatedTerms + SourcesCount < TotalNumberSources / 2 THEN
10.     ADD Term TO Q1
11.     CumulatedSources = CumulatedSources + SourcesCount
12.   ELSE
13.     ADD Term TO Q2
14.     CumulatedSources = INFINITY  // Avoid adding any more to Q1
15.   END IF
16. END FOR
```

FIG. 5. Pseudocode for the new split mechanism.

terms which have a total number of sources as close to the sum of the sources of the rest of terms. For example, if the query has the following SO part: SO = J* OR SO = A* OR SO = Y* OR SO = X*, and the number of sources that begin with J is 1,000, with A is 500, with Y is 25, and with X is 10, then the SO part of the first subquery would include only the SO = J* term, and the SO part of the second subquery would include the rest: SO = A* OR SO = Y* OR SO = X*. A pseudocode description of this new split mechanism is presented in Figure 5.

Collapsing mechanism. This new collapsing mechanism has a quite important role in this second approach. In fact, this collapsing mechanism is the one that allows overcoming the 50-terms restriction by simplifying the NOT parts of the subqueries and avoiding the 100 queries in the search-history problem because it makes every subquery independent from the previous ones: The NOT parts will not just be applied to previous queries but will be composed directly by adding the SO terms that were previously executed in the search tree.

The basic idea of the collapsing mechanism is that once all the child subqueries of a particular query have been executed, there is no need to add both subqueries to the NOT part of the next queries to be executed but they could be replaced for the more general parent query. For example, suppose that a query with a SO part has been split into subqueries with

DOI: 10.1002/asi

the following SO parts: SO = X−*, SO = XE*, SO = XI*, SO = XU*. According to the first approach presented in previous sections, the next queries to be executed should have a NOT part in the form of NOT SO = X−* NOT SO = XE* NOT SO = XI* NOT SO = XU*. However, it is quite obvious that a NOT part in the form NOT SO = X* would be equivalent but much more simple. Hence, we would avoid the subsequent queries growing excessively.

Only one important clarification in this mechanism needs to be introduced. As previously mentioned, the list of sources downloaded from the *WoS* might not be complete (and, in fact, is not complete because it includes only journals), which could lead to situations where some results are not retrieved. To get this around the collapsing mechanism will add an extra wild card subquery to assure that no results are omitted from that particular branch in the search space. Continuing with the previous example, once the four subqueries for the X prefix have been completed, an additional wild card query in the form SO = X* NOT (SO = X−* OR SO = XE* OR SO = XI* OR SO = XU*) would be added to the pending queries list. From the examples in the next section, it can be seen that these queries usually retrieve a quite low number of results (In many of them, no new result is obtained.)

Examples and discussion (PY = 2007 AND CU = USA). To test this new approach, we first used the same query as in the previous section: PY = 2007 AND CU = USA. As expected, this approach produced a higher amount of queries to obtain the same result. Concretely, it produced 29 different queries (see Table 5). In addition, Figure 6 shows the partition of the search space for this set of queries. Note that some of the queries drawn have three child subqueries. The third of those subqueries corresponds to a wild card subquery produced by the collapsing mechanism. From Figure 6, we also can appreciate from the particular split mechanism that we have included in this approach how the search is skewed toward the left (We first try to solve the subqueries with SO terms with more sources for their prefix.) Note also how many of the subqueries have been simplified.

In addition, a much larger query has been tested with this second approach. The query PY = 2007 has been resolved to have a results count of 1,786,644. To obtain this result, this second approach has produced 67 different queries that were executed in less than 15 min in the *WoS* Web interface. Note that those queries have been generated automatically, without any external source of extra information. Obtaining that result without any previous information about the problem in a manual manner would probably have required much more effort and many more trial-and-error tests. Raw data for this particular query along with its partition figure are included in the Appendix.

Although this approach solves the problems presented in the previous one (thus making this new approach more reliable), it is more inefficient in the way it splits the search space by producing a higher number of queries. However, because it is able to solve much more complex queries, it is probably more suitable as a general source-based partitioning approach. Note that the number of queries issued by this second approach could lower if the source list that we use was complete; however, obtaining such an up-to-date list is indeed a difficult task that should be solved in the future.

Finally, there is a small limitation of source-based partition approaches for the *WoS* (not only for the two presented methods but for any other possible one). For some extremely large results counts, there is a limit which cannot be surpassed with a pure source-based partitioning approach: If for a particular source there exist more than 100,000 results indexed in the *WoS*, it is clear that no additional source partitioning would be possible. However, these situations will not usually happen, as there are only a few of the more than 16,000 sources that do indeed offer more than 100,000 results for themselves (e.g., the journal *Nature* has more than 100,000 results indexed in the *WoS*). In those cases, a special partitioning scheme based on another field should be used.

WoS Query Partitioner Tool

In this section, we present a brief description of the WoS Query Partitioner tool. This application implements both source-based divide-and-conquer partitioning approaches presented in previous sections. It has been programmed as a Java Applet and thus can be executed from almost any modern browser and operating system.

The application (along with some instructions and a description) is hosted in http://sci2s.ugr.es/software/WoSQP/ and can be freely accessed and used.

The WoS Query Partitioner not only implements those two approaches but also produces some graphs and tables similar to the ones presented in this article to help researchers incorporate them in their experimentation.

The use of the WoS Query Partitioner is quite simple and can be summarized in three steps:

1. The user must insert the WoS query to split in the Query field and press the Begin Query Partitioning button. For example, a user could input the query PY = 2007 AND CU = USA (Figure 7).
2. A new window will open asking to execute a particular query to the *WoS* interface and requesting to input the number of results obtained in the blank field. To help the copy-and-paste mechanism into the *WoS* Web page, the query to execute is automatically inserted into the system clipboard. If the results count of the executed query is greater than 100,000 (>100000), the field must be left blank. The Next Iteration button should then be pressed (Figure 8).
3. Step 2 will be repeated several times until a final results count is obtained. Once this count is shown (Figure 9), the graphs and table for the query will be generated.

Note that although it has not been done, it is rather simple to extend this (or any similar) application to make it fully automatic (i.e., allowing it to make the queries directly to the *WoS*).

DOI: 10.1002/asi

TABLE 5. Queries produced by the second source-based partitioning approach for the query PY=2007 AND CU=USA.

	Query	Items	∑
#1	PY=2007 AND CU=USA	>100,000	
#2	PY=2007 AND CU=USA AND (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	>100,000	
#3	PY=2007 AND CU=USA AND (SO=J*)	>100,000	
#4	PY=2007 AND CU=USA AND (SO=JO*)	>100,000	
#5	PY=2007 AND CU=USA AND (SO=JOU*)	>100,000	
#6	PY=2007 AND CU=USA AND (SO=JOURNAL *)	>100,000	
#7	PY=2007 AND CU=USA AND (SO=JOURNAL O*)	>100,000	
#8	PY=2007 AND CU=USA AND (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*)	48,632	48,632
#9	PY=2007 AND CU=USA AND (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N* OR SO=JOURNAL OF B* OR SO=JOURNAL OF I* OR SO=JOURNAL OF F* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF R* OR SO=JOURNAL OF O* OR SO=JOURNAL OF L* OR SO=JOURNAL OF V* OR SO=JOURNAL OF D* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF K* OR SO=JOURNAL OF U* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* OR SO=JOURNAL OF Y* OR SO=JOURNAL OF X*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*)	59,039	107,671
#10	PY=2007 AND CU=USA AND (SO=JOURNAL O*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*) NOT (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N* OR SO=JOURNAL OF B* OR SO=JOURNAL OF I* OR SO=JOURNAL OF F* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF R* OR SO=JOURNAL OF O* OR SO=JOURNAL OF L* OR SO=JOURNAL OF V* OR SO=JOURNAL OF D* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF K* OR SO=JOURNAL OF U* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* OR SO=JOURNAL OF Y* OR SO=JOURNAL OF X*)	1	107,672
#11	PY=2007 AND CU=USA AND (SO=JOURNAL F* OR SO=JOURNAL D* OR SO=JOURNAL A* OR SO=JOURNAL I* OR SO=JOURNAL N*) NOT (SO=JOURNAL O*)	345	108,017
#12	PY=2007 AND CU=USA AND (SO=JOURNAL *) NOT (SO=JOURNAL O*) NOT (SO=JOURNAL F* OR SO=JOURNAL D* OR SO=JOURNAL A* OR SO=JOURNAL I* OR SO=JOURNAL N*)	0	108,017
#13	PY=2007 AND CU=USA AND (SO=JOURNALI* OR SO=JOURNALS*) NOT (SO=JOURNAL *)	311	108,328
#14	PY=2007 AND CU=USA AND (SO=JOU*) NOT (SO=JOURNAL *) NOT (SO=JOURNALI* OR SO=JOURNALS*)	0	108,328
#15	PY=2007 AND CU=USA AND (SO=JOA* OR SO=JOI* OR SO=JOR* OR SO=JOE* OR SO=JOG* OR SO=JOH* OR SO=JOK* OR SO=JOM*) NOT (SO=JOU*)	178	108,506
#16	PY=2007 AND CU=USA AND (SO=JO*) NOT (SO=JOU*) NOT (SO=JOA* OR SO=JOI* OR SO=JOR* OR SO=JOE* OR SO=JOG* OR SO=JOH* OR SO=JOK* OR SO=JOM*)	4	108,510
#17	PY=2007 AND CU=USA AND (SO=JA* OR SO=JU* OR SO=JE* OR SO=JC* OR SO=JI* OR SO=JB* OR SO=JN* OR SO=JM* OR SO=JP* OR SO=JS*) NOT (SO=JO*)	1773	110,283
#18	PY=2007 AND CU=USA AND (SO=J*) NOT (SO=JO*) NOT (SO=JA* OR SO=JU* OR SO=JE* OR SO=JC* OR SO=JI* OR SO=JB* OR SO=JN* OR SO=JM* OR SO=JP* OR SO=JS*)	0	110,283
#19	PY=2007 AND CU=USA AND (SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=J*)	>100,000	
#20	PY=2007 AND CU=USA AND (SO=A*) NOT (SO=J*)	67,025	177,308
#21	PY=2007 AND CU=USA AND (SO=C* OR SO=P* OR SO=I*) NOT (SO=J*) NOT (SO=A*)	>100,000	
#22	PY=2007 AND CU=USA AND (SO=C*) NOT (SO=J*) NOT (SO=A*)	41,214	218,522
#23	PY=2007 AND CU=USA AND (SO=P* OR SO=I*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*)	82,682	301,204
#24	PY=2007 AND CU=USA AND (SO=B* OR SO=S* OR SO=M* OR SO=E* OR SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	>100,000	
#25	PY=2007 AND CU=USA AND (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	86,175	387,379
#26	PY=2007 AND CU=USA AND (SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*)	>100,000	
#27	PY=2007 AND CU=USA AND (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*)	48,809	436,188
#28	PY=2007 AND CU=USA AND (SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*)	64,697	500,885
#29	PY=2007 AND CU=USA NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E* OR SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*)	0	500,885

DOI: 10.1002/asi

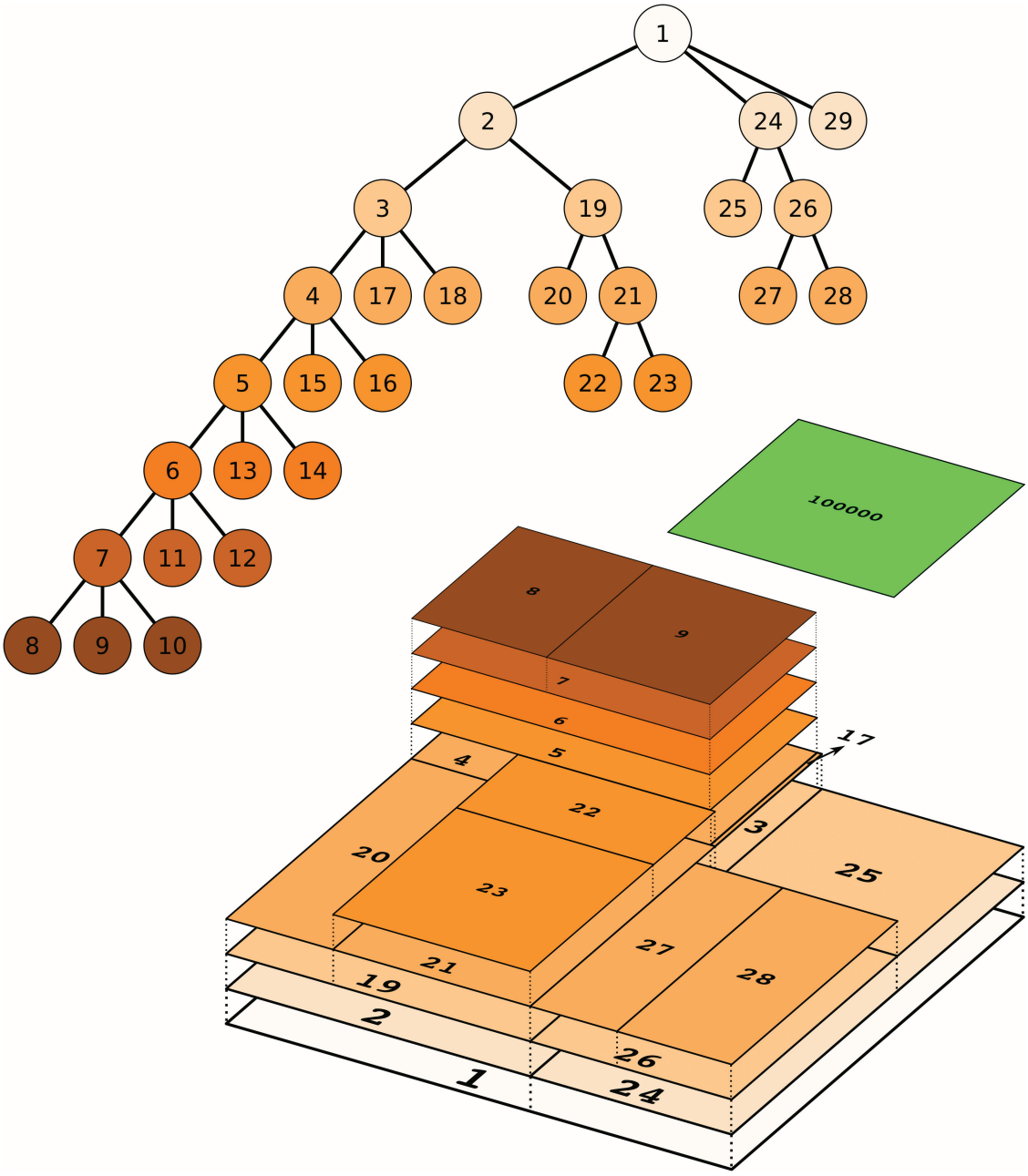

FIG. 6. Actual partition for the queries in Table 5.

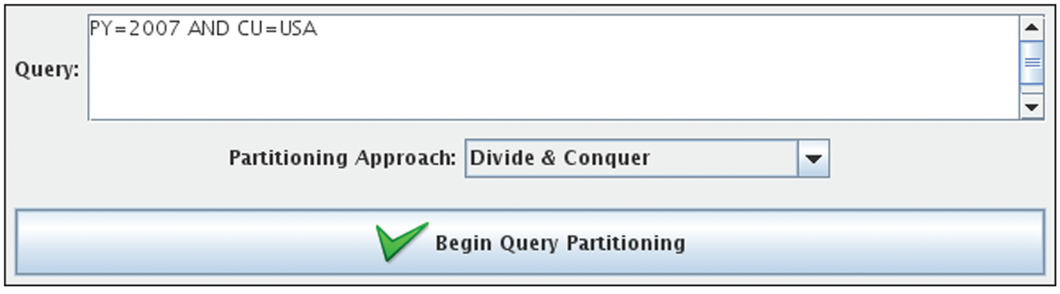

FIG. 7. Initial screen of the WoS Query Partitioner.

DOI: 10.1002/asi

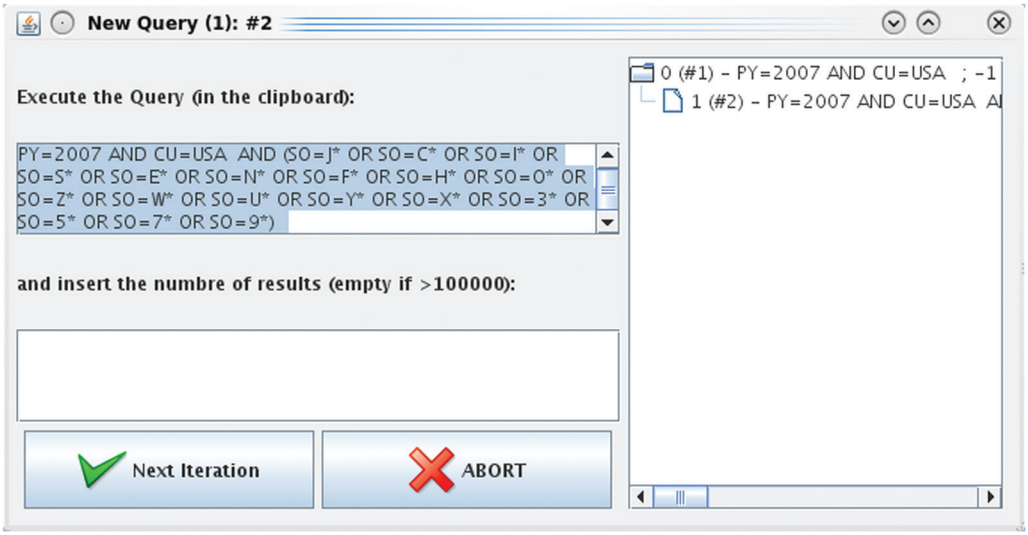

FIG. 8. Window with the subquery to execute.

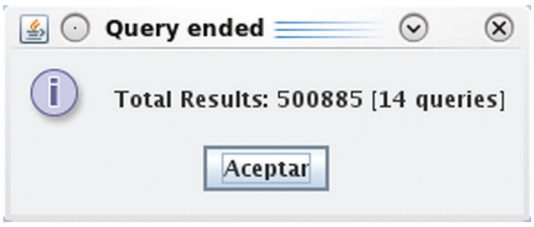

FIG. 9. Final results count is shown by the WoS Query Partitioner.

Conclusions

In this contribution, we have presented two partitioning approaches based on the *source* field for the *WoS* that allow the user to avoid the >100,000 restriction. Both work in a divide-and-conquer fashion by splitting the search space in a recursive way. They are designed to work without needing any extra information about the query being executed and could indeed be incorporated into automatic retrieval applications.

The first approach was designed trying to minimize the number of queries that have to be executed to obatin a solution; however, this approach may encounter other limitations present in the *WoS* interface (e.g., the >50 terms-per-query restriction). To overcome this issue, a second, more reliable approach has been deployed. Although this second approach produces a higher number of queries, it does not suffer from the restrictions of the first approach and is able to obtain accurate results for more complex queries.

Both approaches have been implemented into the WoS Query Partitioner application and tested with a well-known query that has been shown to produce a set of more than 500,000 results. The application is freely accessible and can be used in any future research. In addition, more complex queries with results counts of about 1 and 2 million also have been succesfully tested.

Future research and development might include discovering new heuristics to split the queries, which may offer better efficiency (i.e., less queries to be produced to solve the problem), developing more complex partitioning schemes not only based on the *source* field but with a combination of other fields, and incorporating these approaches into automatic retrieval tools.

Acknowledgments

This article has been developed with financing of the Andalucian Excellence Project (TIC05299), and FEDER funds in the PETRI project (PET2007–0460) and in the FUZZYLING project (TIN2007–61079).

References

Alonso, S., Cabrerizo, F., Herrera-Viedma, E., & Herrera, F. (2009). h-index: A review focused in its variants, computation and standardization for different scientific fields. Journal of Informetrics, 3(4), 273–289.

Archambault, E., Campbell, D., Gingras, Y., & Lariviare, V. (2009). Comparing bibliometric statistics obtained from the web of science and scopus. Journal of the American Society for Information Science and Technology, 60(7), 1320–1326.

Arencibia-Jorge, R., Leydesdorff, L., Chinchilla-Rodríguez, Z., Rousseau, R., & Paris, S. (2009). Retrieval of very large numbers of items in the web of science: An exercise to develop accurate search strategies. El Profesional de la Información, 18(5), 529–533.

E-Resource-1. (2010). Web of science. Retrieved April 28, 2010, from http://wokinfo.com/products_tools/multidisciplinary/webofscience/

E-Resource-2. (2010). Thomson reuters master journal list. Retrieved April 28, 2010, from http://science.thomsonreuters.com/mjl/

Egghe, L. (2010). The hirsch index and related impact measures. Annual Review of Information Science and Technology, 44.

Jacsó, P. (2009). Errors of omission and their implications for computing scientometric measures in evaluating the publishing productivity and impact of countries. Online Information Review, 33, 376–385.

Meho, L., & Rogers, Y. (2008). Citation counting, citation ranking, and h-index of human–computer interaction researchers: A comparison of scopus and web of science. Journal of the American Society for Information Science and Technology, 59(11), 1711–1726.

Vieira, E., & Gomes, J. (2009). A comparison of scopus and web of science for a typical university. Scientometrics, 81(2), 587–600.

Zhou, P., & Leydesdorff, L. (2006). The emergence of China as a leading nation in science. Research Policy, 35, 83–104.

DOI: 10.1002/asi

Appendix

Application of the Second Source-Based Partition Approach to PY = 2007

In Table A1, we present the list of 67 queries that had to be executed to obtain the total number of results for the query PY = 2007 using the second source-based divide-and-conquer partition approach with collapsing mechanism. In Figure A1, the actual partition of those 67 queries is presented. Note that the results counts for some queries are so small that they are not actually visible in the figure.

TABLE A1. Queries produced by the second source-based divide-and-conquer partitioning approach with collapsing mechanism for the query PY = 2007.

	Query	Items	∑
#1	PY=2007	>100,000	
#2	PY=2007 AND (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	>100,000	
#3	PY=2007 AND (SO=J*)	>100,000	
#4	PY=2007 AND (SO=JO*)	>100,000	
#5	PY=2007 AND (SO=JOU*)	>100,000	
#6	PY=2007 AND (SO=JOURNAL *)	>100,000	
#7	PY=2007 AND (SO=JOURNAL O*)	>100,000	
#8	PY=2007 AND (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*)	>100,000	
#9	PY=2007 AND (SO=JOURNAL OF T*)	36,593	36,593
#10	PY=2007 AND (SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*) NOT (SO=JOURNAL OF T*)	97,750	134,343
#11	PY=2007 AND (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N* OR SO=JOURNAL OF B* OR SO=JOURNAL OF I* OR SO=JOURNAL OF F* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF R* OR SO=JOURNAL OF O* OR SO=JOURNAL OF L* OR SO=JOURNAL OF V* OR SO=JOURNAL OF D* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF K* OR SO=JOURNAL OF U* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* OR SO=JOURNAL OF Y* OR SO=JOURNAL OF X*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*)	>100,000	
#12	PY=2007 AND (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*)	71,004	205,347
#13	PY=2007 AND (SO=JOURNAL OF B* OR SO=JOURNAL OF I* OR SO=JOURNAL OF F* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF R* OR SO=JOURNAL OF O* OR SO=JOURNAL OF L* OR SO=JOURNAL OF V* OR SO=JOURNAL OF D* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF K* OR SO=JOURNAL OF U* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* OR SO=JOURNAL OF Y* OR SO=JOURNAL OF X*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*) NOT (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N*)	90,840	296,187
#14	PY=2007 AND (SO=JOURNAL O*) NOT (SO=JOURNAL OF T* OR SO=JOURNAL OF C* OR SO=JOURNAL OF A* OR SO=JOURNAL OF P*) NOT (SO=JOURNAL OF M* OR SO=JOURNAL OF E* OR SO=JOURNAL OF S* OR SO=JOURNAL OF N* OR SO=JOURNAL OF B* OR SO=JOURNAL OF I* OR SO=JOURNAL OF F* OR SO=JOURNAL OF H* OR SO=JOURNAL OF G* OR SO=JOURNAL OF R* OR SO=JOURNAL OF O* OR SO=JOURNAL OF L* OR SO=JOURNAL OF V* OR SO=JOURNAL OF D* OR SO=JOURNAL OF W* OR SO=JOURNAL OF J* OR SO=JOURNAL OF K* OR SO=JOURNAL OF U* OR SO=JOURNAL OF Z* OR SO=JOURNAL OF Q* OR SO=JOURNAL OF Y* OR SO=JOURNAL OF X*)	6	296,193
#15	PY=2007 AND (SO=JOURNAL F* OR SO=JOURNAL D* OR SO=JOURNAL A* OR SO=JOURNAL I* OR SO=JOURNAL N*) NOT (SO=JOURNAL O*)	1,532	297,725
#16	PY=2007 AND (SO=JOURNAL *) NOT (SO=JOURNAL O*) NOT (SO=JOURNAL F* OR SO=JOURNAL D* OR SO=JOURNAL A* OR SO=JOURNAL I* OR SO=JOURNAL N*)	0	297,725
#17	PY=2007 AND (SO=JOURNALI* OR SO=JOURNALS*) NOT (SO=JOURNAL *)	442	298,167
#18	PY=2007 AND (SO=JOU*) NOT (SO=JOURNAL *) NOT (SO=JOURNALI* OR SO=JOURNALS*)	0	298,167
#19	PY=2007 AND (SO=JOA* OR SO=JOI* OR SO=JOR* OR SO=JOE* OR SO=JOG* OR SO=JOH* OR SO=JOK* OR SO=JOM*) NOT (SO=JOU*)	603	298,770
#20	PY=2007 AND (SO=JO*) NOT (SO=JOU*) NOT (SO=JOA* OR SO=JOI* OR SO=JOR* OR SO=JOE* OR SO=JOG* OR SO=JOH* OR SO=JOK* OR SO=JOM*)	12	298,782

(*Continued*)

DOI: 10.1002/asi

TABLE A1. (*Continued*)

	Query	Items	∑
#21	PY=2007 AND (SO=JA* OR SO=JU* OR SO=JE* OR SO=JC* OR SO=JI* OR SO=JB* OR SO=JN* OR SO=JM* OR SO=JP* OR SO=JS*) NOT (SO=JO*)	7,189	305,971
#22	PY=2007 AND (SO=J*) NOT (SO=JO*) NOT (SO=JA* OR SO=JU* OR SO=JE* OR SO=JC* OR SO=JI* OR SO=JB* OR SO=JN* OR SO=JM* OR SO=JP* OR SO=JS*)	0	305,971
#23	PY=2007 AND (SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=J*)	>100,000	
#24	PY=2007 AND (SO=A*) NOT (SO=J*)	>100,000	
#25	PY=2007 AND (SO=AN* OR SO=AC* OR SO=AR*) NOT (SO=J*)	88,910	394,881
#26	PY=2007 AND (SO=AM* OR SO=AD* OR SO=AU* OR SO=AS* OR SO=AP* OR SO=AL* OR SO=AG* OR SO=AF* OR SO=AT* OR SO=AQ* OR SO=AB* OR SO=AI* OR SO=AV* OR SO=AA* OR SO=AE* OR SO=AK* OR SO=AJ* OR SO=A * OR SO=AX*) NOT (SO=J*) NOT (SO=AN* OR SO=AC* OR SO=AR*)	>100,000	
#27	PY=2007 AND (SO=AM* OR SO=AD* OR SO=AU*) NOT (SO=J*) NOT (SO=AN* OR SO=AC* OR SO=AR*)	63,672	458,553
#28	PY=2007 AND (SO=AS* OR SO=AP* OR SO=AL* OR SO=AG* OR SO=AF* OR SO=AT* OR SO=AQ* OR SO=AB* OR SO=AI* OR SO=AV* OR SO=AA* OR SO=AE* OR SO=AK* OR SO=AJ* OR SO=A * OR SO=AX*) NOT (SO=J*) NOT (SO=AN* OR SO=AC* OR SO=AR*) NOT (SO=AM* OR SO=AD* OR SO=AU*)	67,228	525,781
#29	PY=2007 AND (SO=A*) NOT (SO=J*) NOT (SO=AN* OR SO=AC* OR SO=AR*) NOT (SO=AM* OR SO=AD* OR SO=AU* OR SO=AS* OR SO=AP* OR SO=AL* OR SO=AG* OR SO=AF* OR SO=AT* OR SO=AQ* OR SO=AB* OR SO=AI* OR SO=AV* OR SO=AA* OR SO=AE* OR SO=AK* OR SO=AJ* OR SO=A * OR SO=AX*)	0	525,781
#30	PY=2007 AND (SO=C* OR SO=P* OR SO=I*) NOT (SO=J*) NOT (SO=A*)	>100,000	
#31	PY=2007 AND (SO=C*) NOT (SO=J*) NOT (SO=A*)	>100,000	
#32	PY=2007 AND (SO=CO* OR SO=CA*) NOT (SO=J*) NOT (SO=A*)	56,200	581,981
#33	PY=2007 AND (SO=CH* OR SO=CU* OR SO=CL* OR SO=CE* OR SO=CR* OR SO=CI* OR SO=CY* OR SO=CZ* OR SO=CM* OR SO=CN* OR SO=CC* OR SO=CF* OR SO=CB* OR SO=CS* OR SO=CT* OR SO=CW*) NOT (SO=J*) NOT (SO=A*) NOT (SO=CO* OR SO=CA*)	91,530	673,511
#34	PY=2007 AND (SO=C*) NOT (SO=J*) NOT (SO=A*) NOT (SO=CO* OR SO=CA*) NOT (SO=CH* OR SO=CU* OR SO=CL* OR SO=CE* OR SO=CR* OR SO=CI* OR SO=CY* OR SO=CZ* OR SO=CM* OR SO=CN* OR SO=CC* OR SO=CF* OR SO=CB* OR SO=CS* OR SO=CT* OR SO=CW*)	95	673,606
#35	PY=2007 AND (SO=P* OR SO=I*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*)	>100,000	
#36	PY=2007 AND (SO=P*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*)	>100,000	
#37	PY=2007 AND (SO=PR* OR SO=PH*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*)	>100,000	
#38	PY=2007 AND (SO=PR*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*)	75,351	748,957
#39	PY=2007 AND (SO=PH*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=PR*)	44,317	793,274
#40	PY=2007 AND (SO=PA* OR SO=PO* OR SO=PE* OR SO=PS* OR SO=PL* OR SO=PU* OR SO=PI* OR SO=PF* OR SO=PM* OR SO=PT* OR SO=PY* OR SO=P * OR SO=PC* OR SO=PN* OR SO=PP*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=PR* OR SO=PH*)	61,177	854,451
#41	PY=2007 AND (SO=P*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=PR* OR SO=PH*) NOT (SO=PA* OR SO=PO* OR SO=PE* OR SO=PS* OR SO=PL* OR SO=PU* OR SO=PI* OR SO=PF* OR SO=PM* OR SO=PT* OR SO=PY* OR SO=P * OR SO=PC* OR SO=PN* OR SO=PP*)	37	854,488
#42	PY=2007 AND (SO=I*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=P*)	>100,000	
#43	PY=2007 AND (SO=IN*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=P*)	77,288	931,776
#44	PY=2007 AND (SO=IE* OR SO=IR* OR SO=IS* OR SO=IM* OR SO=IZ* OR SO=IC* OR SO=IT* OR SO=ID* OR SO=IL* OR SO=IB* OR SO=IO* OR SO=IG* OR SO=II* OR SO=IU* OR SO=IA* OR SO=IF* OR SO=IH* OR SO=IK* OR SO=IP*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=P*) NOT (SO=IN*)	70204	1,001,980
#45	PY=2007 AND (SO=I*) NOT (SO=J*) NOT (SO=A*) NOT (SO=C*) NOT (SO=P*) NOT (SO=IN*) NOT (SO=IE* OR SO=IR* OR SO=IS* OR SO=IM* OR SO=IZ* OR SO=IC* OR SO=IT* OR SO=ID* OR SO=IL* OR SO=IB* OR SO=IO* OR SO=IG* OR SO=II* OR SO=IU* OR SO=IA* OR SO=IF* OR SO=IH* OR SO=IK* OR SO=IP*)	55	1,002,035
#46	PY=2007 AND (SO=B* OR SO=S* OR SO=M* OR SO=E* OR SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	>100,000	
#47	PY=2007 AND (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	>100,000	

(*Continued*)

DOI: 10.1002/asi

TABLE A1. (*Continued*)

	Query	Items	∑
#48	PY=2007 AND (SO=B*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*)	81,487	1,083,522
#49	PY=2007 AND (SO=S* OR SO=M* OR SO=E*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B*)	>100,000	
#50	PY=2007 AND (SO=S*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B*)	75,041	1,158,563
#51	PY=2007 AND (SO=M* OR SO=E*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B*) NOT (SO=S*)	>100,000	
#52	PY=2007 AND (SO=M*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B*) NOT (SO=S*)	76,826	1,235,389
#53	PY=2007 AND (SO=E*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B*) NOT (SO=S*) NOT (SO=M*)	92,762	1,328,151
#54	PY=2007 AND (SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*)	>100,000	
#55	PY=2007 AND (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*)	>100,000	
#56	PY=2007 AND (SO=R*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*)	41,987	1,370,138
#57	PY=2007 AND (SO=N* OR SO=T* OR SO=F*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R*)	>100,000	
#58	PY=2007 AND (SO=N*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R*)	69,994	1,440,132
#59	PY=2007 AND (SO=T* OR SO=F*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R*) NOT (SO=N*)	86,270	1,526,402
#60	PY=2007 AND (SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*)	>100,000	
#61	PY=2007 AND (SO=G* OR SO=H* OR SO=D*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*)	>100,000	
#62	PY=2007 AND (SO=G*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*)	33,426	1,559,828
#63	PY=2007 AND (SO=H* OR SO=D*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=G*)	71,420	1,631,248
#64	PY=2007 AND (SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=G* OR SO=H* OR SO=D*)	>100,000	
#65	PY=2007 AND (SO=O* OR SO=L*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=G* OR SO=H* OR SO=D*)	60,670	1,691,918
#66	PY=2007 AND (SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*) NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E*) NOT (SO=R* OR SO=N* OR SO=T* OR SO=F*) NOT (SO=G* OR SO=H* OR SO=D*) NOT (SO=O* OR SO=L*)	94,726	1,786,644
#67	PY=2007 NOT (SO=J* OR SO=A* OR SO=C* OR SO=P* OR SO=I*) NOT (SO=B* OR SO=S* OR SO=M* OR SO=E* OR SO=R* OR SO=N* OR SO=T* OR SO=F* OR SO=G* OR SO=H* OR SO=D* OR SO=O* OR SO=L* OR SO=Z* OR SO=V* OR SO=W* OR SO=K* OR SO=U* OR SO=Q* OR SO=Y* OR SO=2* OR SO=X* OR SO=1* OR SO=3* OR SO=4* OR SO=5* OR SO=6* OR SO=7* OR SO=8* OR SO=9* OR SO=0*)	0	1,786,644

DOI: 10.1002/asi

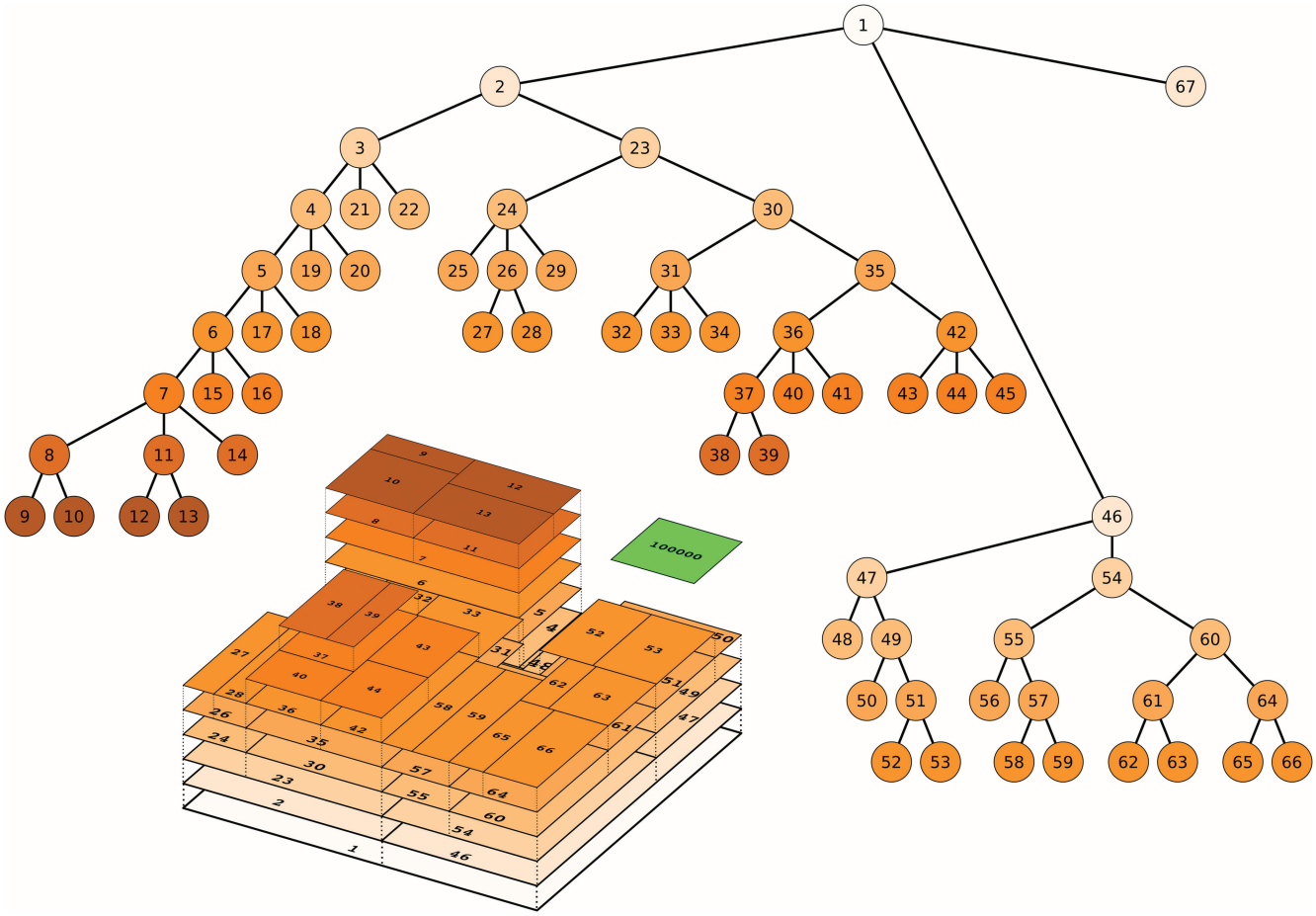

FIG. A1. Actual partition for the query PY = 2007 (Table A1).

DOI: 10.1002/asi