Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 3132004 A1 2022/03/29

(21) 3 132 004

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2021/09/27

(41) Mise à la disp. pub./Open to Public Insp.: 2022/03/29

(30) Priorité/Priority: 2020/09/29 (CN202011048287.7)

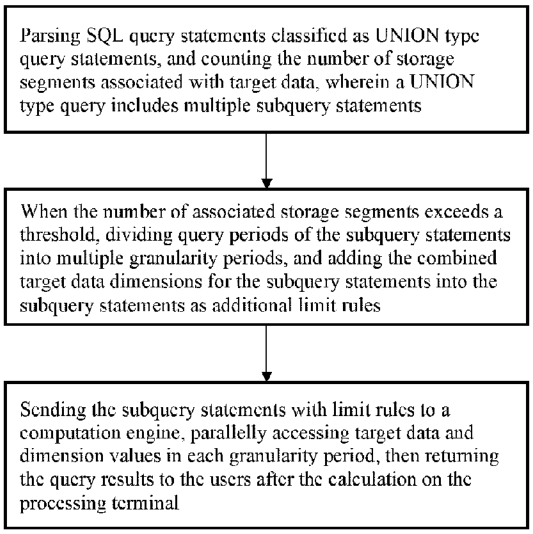
(51) Cl.Int./Int.Cl. *G06F 16/2453* (2019.01)

(71) Demandeur/Applicant:
10353744 CANADA LTD., CA

(72) Inventeurs/Inventors:
LU, YONGLIANG, CN;
ZHENG, YAOFENG, CN;
TANG, GUOQIANG, CN;
SANG, QIANG, CN;
SUN, QIAN, CN

(74) Agent: HINTON, JAMES W.

(54) Titre : METHODE ET DISPOSITIF D'OPTIMISATION D'INTERROGATION SQL
(54) Title: SQL QUERY OPTIMIZATION METHOD AND DEVICE


Parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements
When the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules
Sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal

(57) Abrégé/Abstract:
Disclosed in the present invention is a SQL query optimization method and device, related to the field of the big data technologies, to prevents initiation of circuit-breaker mechanism to improve the query efficiency. The forementioned method comprises: parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements; when the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, then adding combined target data dimensions for the subquery statements into the subquery statements as additional limit rules; and sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimensions in each granularity period, then returning the query results to users after calculation. The forementioned device is applied to achieve the described method.

ABSTRACT

Disclosed in the present invention is a SQL query optimization method and device, related to the field of the big data technologies, to prevents initiation of circuit-breaker mechanism to improve the query efficiency. The forementioned method comprises: parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements; when the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, then adding combined target data dimensions for the subquery statements into the subquery statements as additional limit rules; and sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimensions in each granularity period, then returning the query results to users after calculation. The forementioned device is applied to achieve the described method.

Date Reçue/Date Received 2021-09-27

SQL QUERY OPTIMIZATION METHOD AND DEVICE

Technical Field

[0001] The present invention is related to the field of big data technologies, in particular to a SQL query optimization method and device.

Background

[0002] With the growing number of service data volume, the intake of a traffic model into the data source in the Druid engine takes a large volume of data. During each year-on-year or month-on-month query of SQL, internal union-all statements require to search for all the data in certain SQL periods. In order to guarantee a valid query for the other tasks on the Druid engine, the Druid engine calculates the overall number of the segments in the current SQL period, and initiate the circuit-breaker mechanisms once the overall number of the segments exceeds a threshold. Although the overall number of the segments would be less than the threshold, the full-volume query will add up query time, and further reduce query efficiency.

Summary

[0003] The present invention is aiming at providing a SQL query optimization method and device, y\to prevent the initiation of circuit-breaker mechanism and improve query efficiency.

[0004] For the forementioned aim, from the first prospective, the present invention provides a SQL query optimization method, comprising:

[0005] parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements;

[0006] when the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules; and

[0007] sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal.

Date Reçue/Date Received 2021-09-27

[0008] Preferably, when the number of associated storage segments exceeds a threshold, the query periods of the subquery statements are divided into multiple granularity periods by:

[0009] accessing the query period corresponding to each described subquery statement, and dividing each query period into multiple granularity periods according to a pre-set mean granularity.

[0010] Preferably, the combined target data dimensions for the subquery statements are added into the subquery statements as additional limit rules by:

[0011] identifying the dimensions of the target data corresponding to each subquery statement; where if the dimensions of the target data corresponding to each subquery statement are identical, selecting one of the described dimension of the target data corresponding to the subquery statement and adding into each subquery statement as an additional limit rule; and

[0012] where if the dimensions of the target data corresponding to each subquery statement are not identical, combining the target data dimensions for the subquery statements and adding the combined target data into the subquery statements as an additional limit rule.

[0013] Preferably, the described UNION type query statement includes at least a primary period query statement and a secondary period query statement, and the described limit rules includes query metrices, a dimension rank, and numbers of target data queries besides the additional rule.

[0014] Preferably, the computation engine parallelly accesses the target data and dimension values in each granularity period by:

[0015] according to each subquery statement and corresponding limit rules, on the computation engine, searching for target data satisfying the described target data query amount rules from the storage segments corresponding to data sources, and extracting an associated dimension value.

[0016] Furthermore, the query result is return to users after calculation on the processing terminal by:

[0017] calculating according to the pre-set rules on the processing terminal and returning query results to users based on the dimension value of each subquery.

[0018] Preferably, the described subquery statements with limit rules are sent to the computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal, is followed by the step of:

Date Reçue/Date Received 2021-09-27

[0019] counting the hot data source and hot data dimensions from the history SQL statements, and pre-extract the dimension values of the hot data dimensions, for fast calling of the upcoming SQL query statement.

[0020] described subquery statements with limit rules are sent to the computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal, is followed by the step of:

[0021] caching each subquery statement and the target data and corresponding dimension values for each subquery statement in the database, for fast calling of upcoming identical subquery statements from the database.

[0022] Compared with the currently available techniques, the SQL query optimization method can provide the following benefits:

[0023] the SQL query optimization method provided in the present invention requires to count the storage segments corresponding to the target data before searching for the target data on a computation engine based on the SQL query statements. When the number of associated storage segments exceeds a threshold, the direct execution of a full-volume query may trigger the circuit-breaker mechanism of the computation engine. Consequently, the present invention proposes the method of dividing query periods of the subquery statements into multiple granularity periods, and prevents the problems of no query results due to different dimensions of the target data from different subquery statements. Before the query being executed, the target data dimensions for the subquery statements are combined and added into the subquery statements as additional limit rules. After performing the forementioned procedures, the final subquery statements with limit rules are sent to a computation engine, followed by parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal.

[0024] Obviously, the present invention can not only solve the problem of directly circuit-breaker mechanism of the computation engine during a full-volume query, but also standardize and yield the uniform dimension of the target data for each subquery statement, to guarantee the query result output.

[0025] From the second perspective, a SQL query optimization device is provided in the present invention used in the forementioned SQL query optimization method, comprising:

[0026] a statement parsing unit, configured to parse SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements;

[0027] a detecting and processing unit, configured to divide query periods of the subquery statements into multiple granularity periods when the number of associated storage segments exceeds a threshold, combine the target data dimensions for the subquery statements, and add the combined target data into the subquery statements as an additional limit rule; and

[0028] a searching unit, configured to send the subquery statements with limit rules to a computation engine, parallelly access target data and dimension values in each granularity period, then return the query results to the users after the calculation on the processing terminal.

[0029] Compared with the currently available techniques, the SQL query optimization device can provide the same benefits as forementioned SQL query optimization method and is not explained in detail herein.

[0030] From the third perspective, a computer readable storage medium is provided in the present invention, wherein the computer programs are stored on. When the described computer programs are executed by the processor, any of the procedures in the forementioned SQL query optimization method are performed.

[0031] Compared with the currently available techniques, the computer readable storage medium can provide the same benefits as forementioned SQL query optimization method and is not explained in detail herein.

Brief descriptions of the drawings

[0032] To clarify, the described drawings are providing further explanations for the present invention as a part of the present invention. The demonstration embodiments and descriptions are used to explain the present invention and shall not limit the present invention. In the drawings:

[0033] Fig. 1 is a portion of the flow diagram of the SQL query optimization method in the present invention.

[0034] Fig. 2 is the full flow diagram of the SQL query optimization method in the present invention.

[0035] Fig. 3 is a schematic diagram of the SQL query optimization method in the present invention.

Date Reçue/Date Received 2021-09-27

Detailed descriptions

[0036] In order to make the objective, the technical scheme, and the advantages of the present invention clearer, the present invention will be explained further in detail precisely below with references to the accompany drawings. Obviously, the embodiments described below are only a portion of embodiments of the present invention and cannot represent all possible embodiments. Based on the embodiments in the present invention, the other applications by those skilled in the art without any creative works are falling within the scope of the present invention.

[0037] Embodiment one

[0038] Referring to Fig. 1, a SQL query optimization method is provided in the present embodiment, comprising:

[0039] parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements; when the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules; and sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal.

[0040] The SQL query optimization method provided in the present invention requires to count the storage segments corresponding to the target data before searching for the target data on a computation engine based on the SQL query statements. When the number of associated storage segments exceeds a threshold, the direct execution of a full-volume query may trigger the circuit-breaker mechanism of the computation engine. Consequently, the present invention proposes the method of dividing query periods of the subquery statements into multiple granularity periods, and prevents the problems of no query results due to different dimensions of the target data from different subquery statements. Before the query being executed, the target data dimensions for the subquery statements are combined and added into the subquery statements as additional limit rules. After performing the forementioned procedures, the final subquery statements with limit rules are sent to a computation engine, followed by parallelly accessing target data and

Date Reçue/Date Received 2021-09-27

dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal.

[0041] Obviously, the present invention can not only solve the problem of directly circuit-breaker mechanism of the computation engine during a full-volume query, but also standardize and yield the uniform dimension of the target data for each subquery statement, to guarantee the query result output.

[0042] In practice, the UNION type query statements are represented as UNION ALL, including primary period query statement and a secondary period query statement. The described limit rules include query metrices, dimension rank, and numbers of target data queries besides the additional rule. The computation engine is Druid engine; the source of data is noted as datasource; the storage segments are noted as segment, and the processing terminal is the spark terminal. In particular, the numbers of target data queries in the limit rules are used to restrict the target data reading volume by the computation engine, to prevent initiating the circuit-breaker mechanisms.

[0043] In the forementioned embodiment, when the number of associated storage segments exceeds the threshold, the query periods of the subquery statements are divided into multiple granularity periods by:

[0044] accessing the query period corresponding to each described subquery statement, and dividing each query period into multiple granularity periods according to a pre-set mean granularity.

[0045] Illustratively, the SQL query statement includes inner statement and outer statement. Taking the year-to-year period query as an example, the inner statement includes the primary subquery statement and the secondary subquery statement, wherein the primary subquery statement and the secondary subquery statement imply the current period query statement and the historical period query statement, individually. The inner statement is used to acquire the target data corresponding to the primary subquery statement and the secondary subquery statement from the Druid, and the outer statement is used to access the dimensions of the target data and perform associated year-to-year calculation from the spark terminal based on the inner statement.

[0046] In practice, referring to Fig. 2, the logic plan hierarchy is obtained by parsing the SQL query statements, to extract the original data information able to be sliced, such as the target data

Date Reçue/Date Received 2021-09-27

associated storage segment query command, target data dimension query command, target data matrix query command, target data period query command, etc. Then, the number of the storage segments associated with the target data is determined to be exceeding the threshold or not. Where if the storage segments associated with the target data exceeds the threshold, the entire subquery can be sent to the Druid for a full-volume query based on the current techniques. Where if the storage segments associated with the target data does not exceed the threshold, the query period is evenly divided into multiple granularity periods for parallel queries, to prevent initiating the circuit-breaker mechanisms. For example, the minimum granularity is a day when the datasource is taken, and the current SQL query statement queries the data within one month. The one-month query period is divided into thirty granularity periods. In the following, the SQL query statement corresponding to each granularity period are processed, in order to reserve only necessary portion of the target dimension data and filter out the irrelevant derived metrices data. The post-processing SQL query data corresponding to these granularity periods are converted to json statements receivable by the druid. A request for acquiring the target data is sent to the druid. After the druid queries the target data for each granularity period, the dimension sets and corresponding dimension values of each granularity period are returned by the druid. The data sets returned from all granularity periods are combined. Then, a data set with less or equal data volume to the target data query is obtained via selection sort algorithm.

[0047] Illustratively, referring to Fig. 3 using SQL query statements to gather the outcomes of month-to-month ratio of the current month and the previous month, the target data query volume in the limit rules is set to be 500 pieces of data. If a one-month period query is executed in the druid, the large table in the druid contains approximately 200 million pieces target data. In the present embodiment, the one-month period is divided into 30 granularity periods, wherein each granularity period corresponds to approximately 1 million pieces of target data. Using the druid to parallelly access the data set associated with the granularity periods of the current month and the previous month, only 500 pieces of target data are reserved for each granularity periods according to the limit rule of 500 pieces of target data queries. Then, the target data of the same granularity periods in the current month and the previous months are combined to yield 500 pieces of target data. By scanning the 500 pieces of target data in each granularity, the final 500 query results after filtering are returned to the user. Compared to the full-volume queries in the current techniques, the present embodiment sets the limit rule of 500 target data queries, and

Date Reçue/Date Received 2021-09-27

allows the druid to access the first 500 pieces of target data based on the limit rules, to greatly reduce the computation pressure over the druid.

[0048] In the forementioned embodiment, the combined target data dimensions of multiple subquery statements are added into the subquery statements as additional limit rules by:

[0049] identifying the dimensions of the target data corresponding to each subquery statement; where if the dimensions of the target data corresponding to each subquery statement are identical, selecting one of the described dimension of the target data corresponding to the subquery statement and adding into each subquery statement as an additional limit rule; and where if the dimensions of the target data corresponding to each subquery statement are not identical, combining the target data dimensions for the subquery statements and adding the combined target data into the subquery statements as an additional limit rule.

[0050] In practice, it is necessary to prevent the problems of non-executable computation of the spark terminal according to the target data based on the primary period and the target data based on the secondary period from the druid due to mismatched target data dimension of the primary and secondary periods. For example, the target data dimension of the primary period is 100, and the target data dimension of the secondary period is 150. The target data dimensions corresponding to the primary and the secondary periods are combined as an additional rule into the subquery statement limit rules. The combined target data dimensions can cover the target data dimensions from both the primary and the secondary dimensions.

[0051] The forementioned embodiment, wherein the computation engine parallelly accesses the target data and dimension values in each granularity period, also includes:

[0052] according to each subquery statement and corresponding limit rules, in the computation engine, searching for the target data satisfying the described target data query amount rules from the storage segments corresponding to the datasources, and extracting an associated dimension value.

[0053] The forementioned embodiment, wherein the query results are returned to the users after the calculation on the processing terminal, also includes:

[0054] based on the returned dimension values of each subquery statements, returning the query results to the users according to the pre-set calculation rules by the processing terminal. For example, in terms of the case wherein the pre-set calculation rules are set for growth rate

Date Reçue/Date Received 2021-09-27

calculation, the growth rate result is obtained via dividing the dimension values of the primary by the dimension values of the primary secondary periods.

[0055] In the forementioned embodiment, wherein the subquery statements with limit rules are sent to the computation engine for parallelly accessing target data and dimension values in each granularity period, then the query results are returned to the users after the calculation on the processing terminal, the procedures also include:

[0056] counting the hot data source and hot data dimensions from the history SQL statements, and pre-extract the dimension values of the hot data dimensions, for fast calling of the upcoming SQL query statement.

[0057] In practice, by keeping counting during the execution of the SQL query statements, based on the most active datasource and the associated dimension fields, the druid statements are generated dynamically when low query volume. Then the described dimension and dimension values are pre-extracted and stored in the form of materialized views. In the upcoming operations, the materialized views can be extracted immediately once the same datasource or dimension value queries are requested. As a result, it is not required to send the requests to the druid for accessing associated dimension value sets, to achieve the fast calling.

[0058] In the forementioned embodiment, wherein the subquery statements with limit rules are sent to the computation engine for parallelly accessing target data and dimension values in each granularity period then the query results are returned to the users after the calculation on the processing terminal, the procedures also include:

[0059] caching each subquery statement and the target data and corresponding dimension values for each subquery statement in the database, for fast calling of upcoming identical subquery statements from the database.

[0060] In practice, when the query results for the present SQL query statement is obtained, each subquery statement and the target data and corresponding dimension values for each subquery statement are stored in the database such as Hbase or other cache storage. For the same upcoming queries, the previous execution plans can be modified based on the cached results without needs for re-execution. In particular, where if the present SQL statement includes the history period query results, the present SQL query statement can be split based on the query periods of the query statements. The history period query results can be directly extracted from the databased and the rest of present periods can be obtained from the druid query. For example,

Date Reçue/Date Received 2021-09-27

the dimension data for day 1 to day 20 has been stored. If the current query requests for dimension from day 1 to day 30, only the granularity periods of day 21 to day 30 are sent to the druid to greatly reduce the pressure over the druid.

[0061] After obtaining the combined target data of the current period and the history period, the subquery statements for the SQL query statements can be processed by:

[0062] 1. sorting the ‘group by’ field for the subqueries;

[0063] 2. sorting the derived metrices required to be calculated by the subqueries;

[0064] 3. combining the target data dimensions for the subquery statements and adding the combined target data into the subquery statements as an additional limit rule;

[0065] after processing each subquery statement, converting the subqueries into the json statements that are readable by the druid. With the additional limit rules, the current subquery statements request more dimensions than the limit rules. In other words, the druid does not need to scan all the data within the period, wherein no circuit-breaker mechanisms initiation or query time-out, to greatly reduce the pressure over the druid.

[0066] On the other hand, the druid returns the filtered data, the spark terminal does not need receive all the data within the period, to reduce the data processing volume on the spark terminal and the overall efficiency is greatly improved.

[0067] Embodiment two

[0068] A SQL query optimization device is provided in the present invention, comprising:

[0069] a statement parsing unit, configured to parse SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements;

[0070] a detecting and processing unit, configured to divide query periods of the subquery statements into multiple granularity periods when the number of associated storage segments exceeds a threshold, combine the target data dimensions for the subquery statements, and add the combined target data into the subquery statements as an additional limit rule; and

[0071] a searching unit, configured to send the subquery statements with limit rules to a computation engine, parallelly access target data and dimension values in each granularity period, then return the query results to the users after the calculation on the processing terminal.

Date Reçue/Date Received 2021-09-27

[0072] Compared with the currently available techniques, the SQL query optimization device can provide the same benefits as forementioned SQL query optimization method and is not explained in detail herein.

[0073] Embodiment three

[0074] A computer readable storage medium is provided in the present invention, wherein the computer programs are stored on. When the described computer programs are executed by the processor, any of the procedures in the forementioned SQL query optimization method are performed.

[0075] Compared with the currently available techniques, the computer readable storage medium can provide the same benefits as forementioned SQL query optimization method and is not explained in detail herein.

[0076] Those skilled in the art can understand and achieve the forementioned all or portions of the procedures via hardware or via programs for sending commands to the related hardware, wherein the described programs can be stored in a computer readable storage medium. The described storage medium may be read-only memory, magnetic disks, CDs, etc.

[0077] The forementioned contents of preferred embodiments of the present invention and shall not limit the applications of the present invention. Therefore, all alternations, modifications, equivalence, improvements of the present invention fall within the scope of the present invention.

Date Reçue/Date Received 2021-09-27

CLAIMS

1. A SQL query optimization method, comprises:

parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements;

when the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules; and

sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal.

2. The method of claim 1, wherein the described number of storage segments exceeds a threshold, the query periods of the subquery statements are divided into multiple granularity periods by:

accessing the query period corresponding to each described subquery statement, and dividing each query period into multiple granularity periods according to a pre-set mean granularity.

3. The method of claim 1 or claim 2, wherein the combined target data dimensions for the subquery statements are added into the subquery statements as additional limit rules by:

identifying the dimensions of the target data corresponding to each subquery statement; where if the dimensions of the target data corresponding to each subquery statement are identical, selecting one of the described dimension of the target data corresponding to the subquery statement and adding into each subquery statement as an additional limit rule; and

Date Reçue/Date Received 2021-09-27

where if the dimensions of the target data corresponding to each subquery statement are not identical, combining the target data dimensions for the subquery statements and adding the combined target data into the subquery statements as an additional limit rule.

4. The method of claim 3, wherein the described UNION type query statement includes at least a primary period query statement and a secondary period query statement, and the described limit rules includes query metrices, a dimension rank, and numbers of target data queries besides the additional rule.

5. The method of claim 4, wherein the computation engine parallelly accesses the target data and dimension values in each granularity period by:

according to each subquery statement and corresponding limit rules, on the computation engine, searching for target data satisfying the described target data query amount rules from the storage segments corresponding to data sources, and extracting an associated dimension value.

6. The method of claim 5, wherein the query result is return to the users after the calculation on the processing terminal by:

based on the dimension value of each subquery, calculating according to the pre-set rules on the processing terminal and returning query results to users.

7. The method of claim 1, wherein the described subquery statements with limit rules are sent to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal, is followed by:

counting hot data source and hot data dimensions from the history SQL statements, and pre-extract the dimension values of the hot data dimensions, for fast calling of the upcoming SQL query statement.

Date Reçue/Date Received 2021-09-27

8. The method of claim 1, wherein the described subquery statements with limit rules are sent to the computation engine, parallelly accessing target data and dimensions in each granularity period, then returning the query results to the users after the calculation on the processing terminal, is followed by:

caching each subquery statement and the target data and corresponding dimension values for each subquery statement in the database, for fast calling of upcoming identical subquery statements from the database.

9. A SQL query optimization device, comprises:

a statement parsing unit, configured to parse SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements;

a detecting and processing unit, configured to divide query periods of the subquery statements into multiple granularity periods when the number of associated storage segments exceeds a threshold, combine the target data dimensions for the subquery statements, and add the combined target data into the subquery statements as an additional limit rule; and

a searching unit, configured to send the subquery statements with limit rules to a computation engine, parallelly access target data and dimension values in each granularity period, then return the query results to the users after the calculation on the processing terminal.

10. A computer readable storage medium wherein the computer programs are stored on. When the described computer programs are executed by the processor, any of the methods in claims 1-8 are performed.

Date Reçue/Date Received 2021-09-27

Drawings

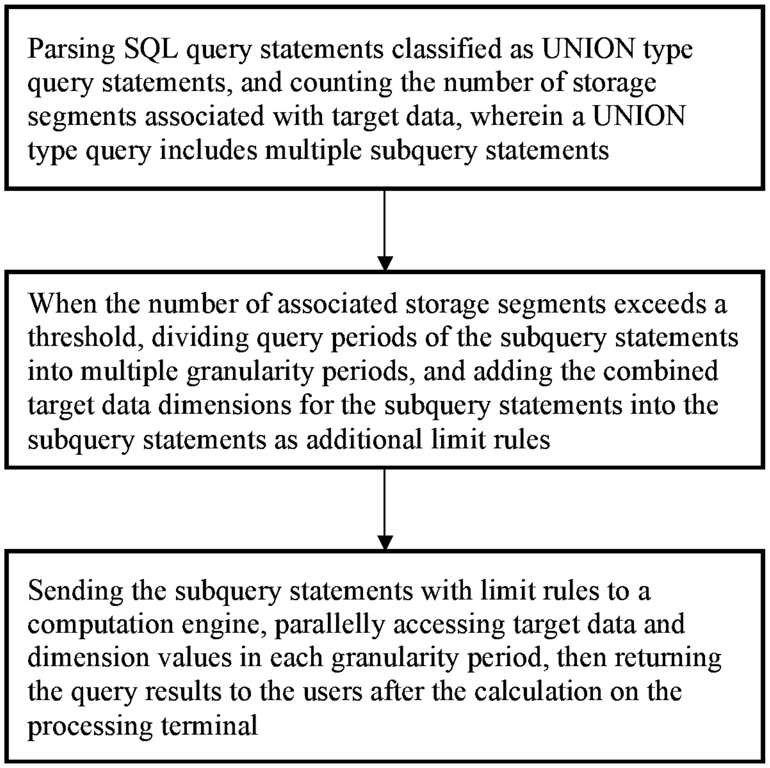

Parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements
When the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules
Sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal

Fig. 1

Date Reçue/Date Received 2021-09-27

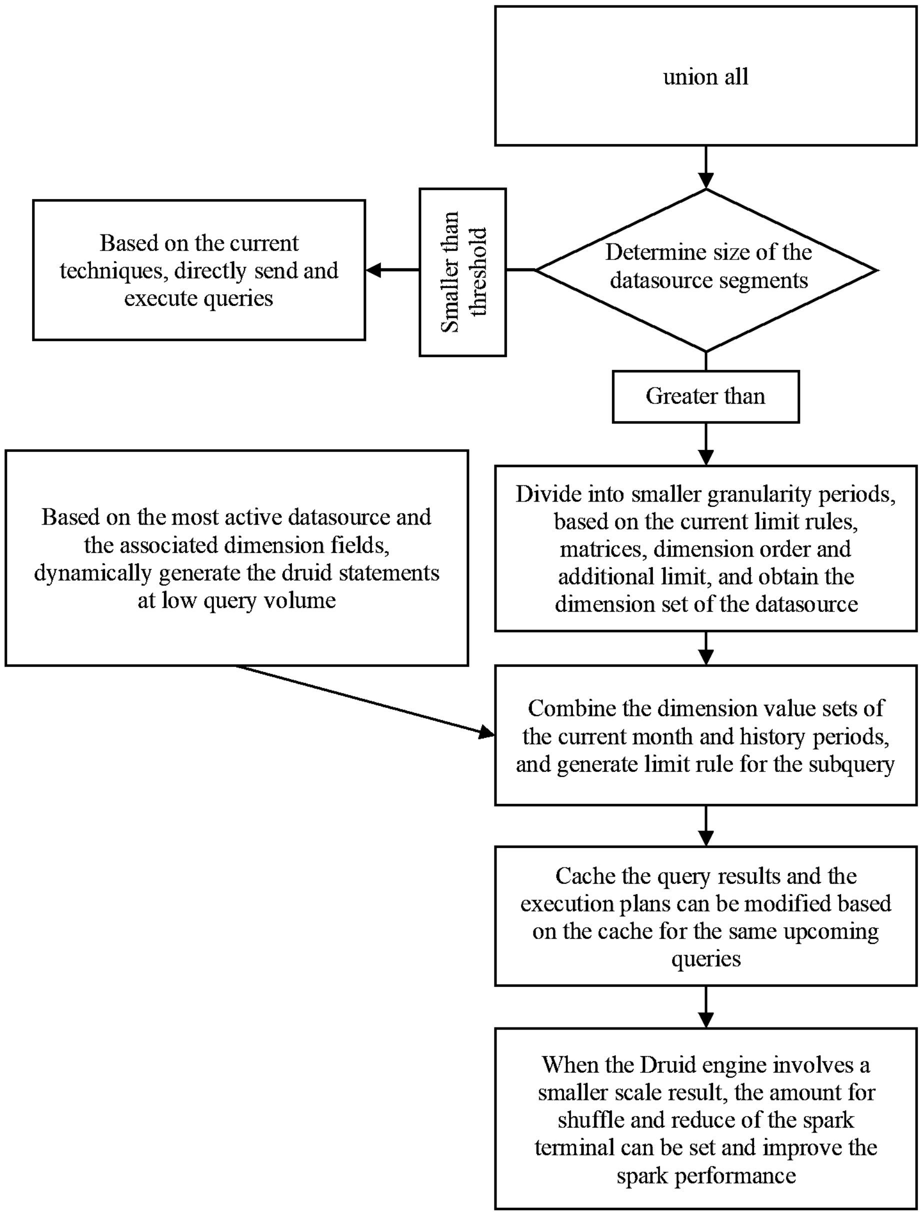

union all
Determine size of the datasource segments
Smaller than threshold
Based on the current techniques, directly send and execute queries
Greater than
Divide into smaller granularity periods, based on the current limit rules, matrices, dimension order and additional limit, and obtain the dimension set of the datasource
Based on the most active datasource and the associated dimension fields, dynamically generate the druid statements at low query volume
Combine the dimension value sets of the current month and history periods, and generate limit rule for the subquery
Cache the query results and the execution plans can be modified based on the cache for the same upcoming queries
When the Druid engine involves a smaller scale result, the amount for shuffle and reduce of the spark terminal can be set and improve the spark performance

Fig. 2

Date Reçue/Date Received 2021-09-27

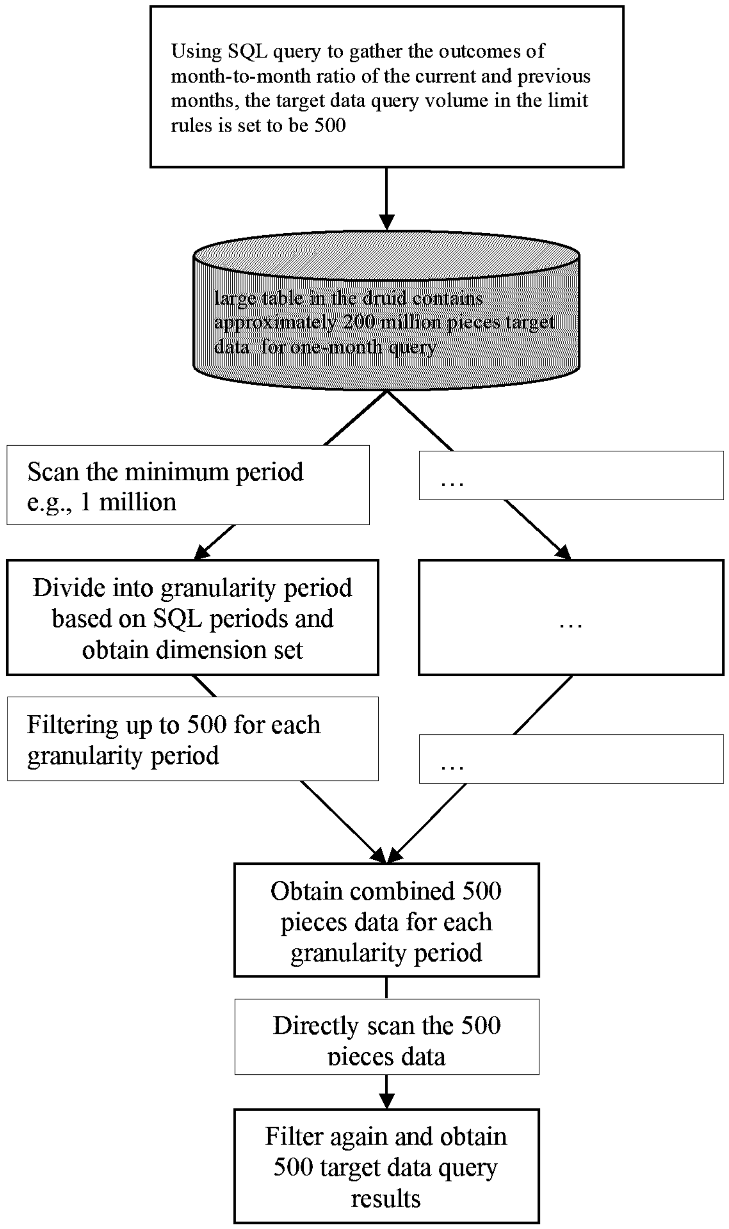
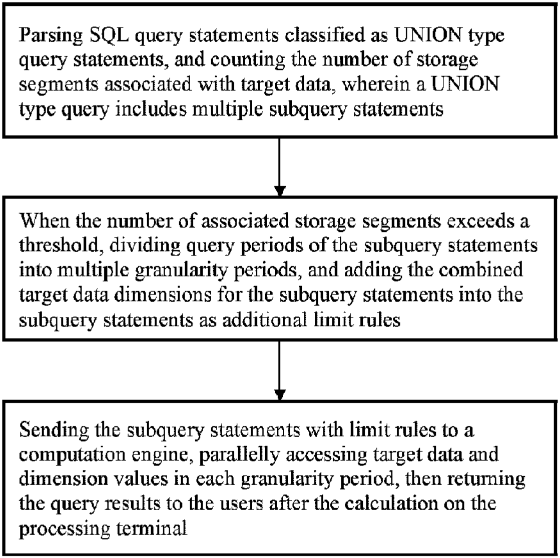

Using SQL query to gather the outcomes of month-to-month ratio of the current and previous months, the target data query volume in the limit rules is set to be 500
large table in the druid contains approximately 200 million pieces target data for one-month query
Scan the minimum period e.g., 1 million
…
Divide into granularity period based on SQL periods and obtain dimension set
…
Filtering up to 500 for each granularity period
…
Obtain combined 500 pieces data for each granularity period
Directly scan the 500 pieces data
Filter again and obtain 500 target data query results

Fig. 3

Date Reçue/Date Received 2021-09-27


Parsing SQL query statements classified as UNION type query statements, and counting the number of storage segments associated with target data, wherein a UNION type query includes multiple subquery statements
When the number of associated storage segments exceeds a threshold, dividing query periods of the subquery statements into multiple granularity periods, and adding the combined target data dimensions for the subquery statements into the subquery statements as additional limit rules
Sending the subquery statements with limit rules to a computation engine, parallelly accessing target data and dimension values in each granularity period, then returning the query results to the users after the calculation on the processing terminal