

EVENT RECORD QUERY METHOD AND APPARATUS

The present application claims priority to Chinese Patent Application No. 201510446726.2 filed on July 27, 2015 and entitled "EVENT RECORD QUERY METHOD AND APPARATUS", which is incorporated herein by reference in its entirety.

Technical Field

The present application relates to the field of database application technologies, and in particular, to an event record query method and apparatus.

Background Art

The development of computer and Internet technologies has brought unprecedented convenience to people, and these technologies have currently penetrated into all aspects of people's daily life. Typical applications include network socializing, network transaction, network entertainment, and so on. For enterprises that provide Internet services, the growth of the number and use frequency of users will inevitably lead to the growth of all kinds of related data. Therefore, higher requirements on data management technology have also been put forward.

In the running process of an application service system, each operation behavior of an external user or each operation processing within the system can be referred to as an event. In general, in addition to achieving basic functions, the system may also completely or selectively write event-related information (including occurrence time, specific content, and the like) into a database in the form of a log to meet subsequent various types of query demands, such as event list output, statistical analysis of data, even fault location, and so on. Common application scenarios include query of online transaction records, query of records of changes in the amount of a bank card, and so on.

For a lightweight event record query request that does not affect normal read and write operations of a database, the database may generally respond quickly. However, if a query request corresponds to an excessively large data amount, the whole query process will bring great pressure to the database, or in a worse case, it may even affect the normal read and write operations of the

database, thereby affecting normal running of basic functions on the application side. In addition, because it is difficult for the database to quickly respond with a query result, the application side may regard it as a timeout of data return, and as a result, the data query function is actually not available, which not only affects user experience but also may cause a user to retry repeatedly, thus further increasing the impact on the database.

Summary of the Invention

The present application provides an event record query method and apparatus for solving the problem of a query having a large data amount affecting a database. The technical solution is as follows.

- 10 The present application provides an event record query method, which includes:
- receiving an event record query request of a user and determining a query event and a query time range corresponding to the request;
 - estimating a query data amount corresponding to the query time range according to a statistical value of a historical number of occurrences of the query event of the user;
 - 15 determining a maximum single-query data amount Max currently allowable by a database;
 - splitting the query request into N query sub-requests, such that each query sub-request separately corresponds to a different time period within the query time range, and a query data amount corresponding to each time period is not greater than Max, where N is an integer not less than 1; and
 - 20 initiating a query to the database by using the N query sub-requests.

The present application further provides an event record query apparatus, which includes:

- a request receiving module configured to receive an event record query request of a user and determine a query event and a query time range corresponding to the request;
- a data amount estimation module configured to estimate a query data amount corresponding to
- 25 the query time range according to a statistical value of a historical number of occurrences of the query event of the user;
- a database performance determination module configured to determine a maximum single-

query data amount Max currently allowable by a database;

a request splitting module configured to split the query request into N query sub-requests, such that each query sub-request separately corresponds to a different time period within the query time range, and a query data amount corresponding to each time period is not greater than Max, where

5 N is an integer not less than 1; and

a query module configured to initiate a query to the database by using the N query sub-requests.

The time record query solution provided by the present application splits a query request having a large query data amount into multiple sub-requests according to the current actual query responding capability of a database, such that the data amount of a single query received by the
10 database will not exceed a bearing capacity of the database, and occurrences of instability are reduced. In another aspect, a query result may also be obtained by responding to the split sub-requests in several times, thereby avoiding a timeout of data return due to an excessively large data amount of a single query.

It should be understood that the above general description and the following detailed
15 description are merely for illustrative and explanatory purposes, but are not intended to limit the present application.

Brief Description of the Drawings

In order to describe the technical solutions in embodiments of the present application or the prior art more clearly, accompanying drawings needed for describing the embodiments or the prior
20 art will be introduced briefly in the following. It is apparent that the accompanying drawings in the following description are merely some embodiments described in the present application, and those of ordinary skill in the art may further obtain other drawings on the basis of these accompanying drawings.

FIG. 1 is a schematic flowchart of the event record query method of the present application;

25 FIG. 2 is a schematic structural diagram of the event record query system of the present application;

FIG. 3 is a second schematic flowchart of the event record query method of the present

application;

FIG. 4 is a schematic structural diagram of the event record query apparatus of the present application; and

FIG. 5 is a second schematic structural diagram of the event record query apparatus of the present application.

Detailed Description

An important performance index of a database is an allowable maximum data amount of a single query. Factors that affect this performance include CPU, memory, data structure, and other aspects. In practical applications, if the data amount corresponding to one query request exceeds the maximum value, other normal read and write operations may be affected, leading to instability of the database.

By using an online transaction platform as an example, different types of users vary greatly in the number of transactions. Some users are ordinary buyers, and the number of transaction records in one year is possibly less than 100. Some users are personal sellers according to their attributes, and the average number of transaction records per month may be several hundred or several thousand. Some users are large sellers according to their attributes, for example, some platform providers and institutions, and the average number of transaction records per month thereof may be more than ten million. Any types of users have the requirement for querying transaction records, so as to carry out reconciliation, shipping processing, and so on. When querying, a user may generally select a time span range. A query request of an ordinary buyer, even if the time span is very large (e.g. a few years), may not have a large data amount. However, a query request of a large seller user, even if the span is very small, may exceed the maximum single-query data amount allowable by a database, resulting in instability of the database.

With respect to the above problem, the solution provided by the present application is to split a query request having a large query data amount into multiple sub-requests according to the current actual query responding capability of a database, thereby reducing occurrences of instability.

In order to make those skilled in the art better understand the technical solution in the present application, the technical solution in the embodiments of the present application will be described below in detail with reference to the accompanying drawings in the embodiments of the present application. It is obvious that the embodiments to be described are only some of rather than all of the embodiments of the present application. All other embodiments derived by those of ordinary skill in the art based on the embodiments of the present application should fall within the protection scope of the present application.

FIG. 1 is a flowchart of the event record query method according to the present application. The method may include the following steps:

S101. An event record query request of a user is received and a query event and a query time range corresponding to the request are determined.

S102. A query data amount corresponding to the query time range is estimated according to a statistical value of a historical number of occurrences of the query event of the user.

S103. A maximum single-query data amount Max currently allowable by a database is determined.

S104. The query request is split into N query sub-requests, such that each query sub-request separately corresponds to a different time period within the query time range, and a query data amount corresponding to each time period is not greater than Max, where N is an integer not less than 1.

S105. A query is initiated to the database by using the N query sub-requests.

An executing entity corresponding to the aforementioned method is referred to as an event record query apparatus in the present application. In the system as shown in FIG. 2, the apparatus 100 is logically located between a user 200 and a database 300. A basic interaction process is: receiving a query request of the user, requesting data from the database, receiving the data returned by the database, and returning the data to the user. In the perspective of the physical location, the apparatus is located on the network side. Specifically, the apparatus may be located in a database server, or may be located in an application server independent of the database server. This does

not need to be limited in the present application.

In S101, the user performs a query operation on a terminal device according to his/her own requirement, and issues a query request to the network side.

During a query, the user may specify an object event type of the query. For example, in an online payment system, the user may specify that what is to be queried is a transaction record, a credit card repayment record, a remittance record, a redemption record, and so on. If the user does not specify the object event type, the query is carried out according to default settings corresponding to the currently used query function. For example, all types of event records are queried.

In addition, during a query, the user may also specify a time range of the query, i.e., specify a start time point T1 and an end time point T2 of the query. According to different specific application scenarios, this function can be accurate to different time granularities, such as day, hour, or even minute. If the user does not specify the time range, the query is carried out according to default settings corresponding to the currently used query function. For example, all records in a valid time are queried, records in the last month are queried, or the like.

After receiving the query request of the user, the event record query apparatus parses the request to determine an event type to be queried by the user as well as a time range [T1, T2] in which the query needs to be performed, for use in the subsequent steps.

After the query time range [T1, T2] is determined, a relatively direct method for determining a query data amount is: directly accessing a database to determine a data amount corresponding to the period [T1, T2]. Although such a manner only requires the database to return a statistical value of data amount, it is still necessary to interact with the database and occupy resources of the database. With respect to this problem, the present application adopts the manner of estimating a data amount of the current query according to historical behaviors of the user, thereby avoiding additional occupation of resources of the database.

In S102, after the event type to be queried by the user is determined, the historical number of occurrences of this type of event of the user further needs to be acquired, thereby estimating the

data amount corresponding to the current query.

A basis concept of the estimation is: estimating a possible data amount corresponding to $[T1, T2]$ according to historical behaviors of the user. Assuming that a statistical value of a historical number of occurrences of an event of a user within duration of a unit period for statistics T_s is C_s ,
5 an estimated value of the number of occurrences C_e of the event in the period $[T1, T2]$ is:
 $C_e = C_s * (T2 - T1) / T_s$.

In order to meet the requirements of the estimation, specific statistical values need to be recorded in advance, or these statistical values need to be acquired from the database or a backup library of the database in advance (for example, in a period in which the system is relatively idle)
10 and stored. According to actual application scenarios, statistical values that need to be recorded or acquired may be selected from several different aspects. For example:

data statistical values in different historical period ranges may be selected, for example, the last year, all valid historical times, and so on;

different statistical time granularities may be selected, for example, "day" is taken as a unit for
15 statistics, "month" is taken as a unit for statistics, and so on; and

different types of statistical values may be selected, such as an average value of the number of occurrences in a unit period for statistics, a maximum value of the number of occurrences in a unit period for statistics, and the like. Selecting a maximum value for estimation is a relatively conservative estimation algorithm, which can effectively reduce the probability of causing a
20 database stability problem.

In addition, the unit period for statistics T_s corresponding to the "maximum value" may be static, i.e., the unit period for statistics is based on the partition of natural time. For example, a maximum transaction volume is selected from transaction volumes of all natural months as a maximum value for estimation. T_s may also be dynamic, i.e., the period for statistics is not based
25 on the partition of natural time. Assuming that the duration corresponding to the period for statistics is T_s , any continuous duration T_s on a time axis can be used as the period for statistics. For example, when a sliding time window having a length of 30 days is used for statistics in all the

transaction records, a maximum transaction volume within the window in a sliding process is selected as a maximum value for estimation. Relatively speaking, it is easy to implement statistics by a static maximum value, but the static maximum value is not sensitive enough to some sudden data changes. A dynamic maximum value can effectively solve this problem, thereby further
5 reducing the probability of causing a database stability problem.

In S103, the maximum single-query data amount Max currently allowable by the database is determined by various factors such as hardware configuration (CPU, memory, and so on), software configuration (software execution efficiency, data structure, data amount of a single piece of data) and actual load of the database server. In the practical application of the solution of the present
10 application, a relatively fixed value can be set, and the value may be directly read for use. Of course, the maximum single-query data amount Max currently allowable by the database may also be estimated in real time. The specific estimation method is relatively independent of the solution of the present application, and is not described here in detail.

It should be noted that there is no particular requirement on the execution sequence of S103
15 and S101-S102. Further, in S104, the initially received query request is split based on the estimated value of the data amount of the current query obtained in S102 and the MAX value obtained in S103 to obtain a plurality of query sub-requests.

In principle, it is not required here that the total query data amount is evenly distributed to each query sub-request, provided that the estimated value of the data amount corresponding to each sub-
20 request is not greater than Max. The following algorithm can be adopted during actual splitting:

The required number of queries, N, is calculated first, where N is an integer and can be obtained by rounding up the value of C_e/Max .

Then, the time range $[T1, T2]$ is divided into N segments:

If $\Delta = 1/N * (T2 - T1)$, time ranges corresponding to the N segments are respectively:

25 $[T1 + 0 * \Delta, T1 + 1 * \Delta];$

$[T1 + 1 * \Delta, T1 + 2 * \Delta];$

$[T1 + 2 * \Delta, T1 + 3 * \Delta];$

...

$[T1+(N-1)*Delta, T2]$.

It should be noted here that a closed interval is used for the time range corresponding to each segment, so as to comply with specific query application requirements. Of course, if supported by the actual query function, a semi-closed interval may also be used for the time range corresponding to each segment. In addition, because N is obtained by rounding up, the right endpoint of the last segment is directly $T2$ instead of $T1+N*Delta$. That is to say, the time span of the last segment is less than or equal to those of other segments.

Of course, the above solution is for the illustrative purposes only, and the actual segmentation method is not limited to the one disclosed above. For example, the time span of the first segment may be made less than or equal to those of other segments, or the like. In addition, in a special but very common case, it is allowed that $N=1$, and the actual situation is equivalent to the case where the estimated value of the data amount corresponding to the initial query request does not exceed MAX . In this case, the splitting process is not required.

In S105, a query is initiated to the database by using the N query sub-requests. It is equivalent to that an original query having the time span of $[T1, T2]$ is split into N queries to request data from the database for several times, such that the data amount of a single query received by the database will not exceed a bearing capability of the database. In another aspect, a query result may also be obtained by responding to the split sub-requests in several times, thereby avoiding a timeout of data return due to an excessively large data amount of a single query.

In the above event record query method, the query request is split dynamically according to the estimated value of the query data amount and the maximum single-query data amount currently allowable by the database. In practical application, other two types of processing logic may also be adopted: In one aspect, for some users having very low event occurrence frequency, it is actually unnecessary to dynamically split (or it is unnecessary to split) the query request, and the query request may be split directly according to a predetermined rule. In another aspect, for some users having very high event occurrence frequency, the number of sub-requests may be excessively large

after the splitting, and in this case, a more suitable solution is to perform delay processing on this type of query request. The above processing logic is explained below in detail with reference to the flowchart shown in FIG. 3.

5 S201: An event record query request of a user is received, and a query event and a query time range corresponding to the request are determined.

This step is similar to S101, thus being not repeatedly described herein.

S202: A statistical value of a historical number of occurrences of the query event of the user is judged.

10 In this embodiment, three types of processing logic are provided for the query request of the user. As for which type of processing logic is triggered specifically, a judgment is made according to a statistical value of a historical number of occurrences C_s of the query event of the user. For the meaning of C_s , reference may be made to the preceding embodiment, and the description is not repeated herein.

15 To distinguish among the three types of processing logic, two thresholds C_1 and C_2 need to be set here in advance, where $C_1 < C_2$. The three types of processing logic are as follows:

If the value of C_s is between C_1 and C_2 , a branch of S203-S204 is triggered, to dynamically split the query request according to an estimated value of the query data amount and a maximum single-query data amount currently allowable by the database, and carry out the query. This branch is equivalent to S102-S105 in the previous embodiment, thus being not repeatedly described here.

20 If C_s is less than C_1 , a branch of S205-S204 is triggered. S205 differs from S203 in that the query request can be directly split into N query sub-requests according to a preset time period length, without the need to dynamically calculate a splitting solution. Then, the practical meaning is: if the statistical value C_s of the historical number of occurrences of the event of a user within duration of a unit period for statistics T_s is relatively small, it is considered that the query request
25 of the user is basically impossible to have an impact on the database, and a relatively fixed splitting manner, such as splitting according to natural months, may be directly adopted, or the request is even not split (the corresponding preset time period has an arbitrary length, $N=1$), and so on,

thereby reducing resource consumption caused by the dynamic calculation. After the splitting is completed, S205 is still performed.

If C_s is greater than C_2 , a branch of S206 is triggered, to perform delayed query processing on the query request. This is because that if C_s is too large, the amount of data to be queried by the user is of a very high order of magnitude, the splitting process alone may consume a large amount of system resources, and the excessively large number of sub-requests may also affect the stability of the database. Therefore, for this case, delayed query processing is required. Here the "delayed" is relative to the aforementioned two processing branches, i.e., after the query request is sent to the database, the database is not required to respond immediately. For example, the query request can be placed in a waiting queue and then processed when resources of the database server are idle. According to the specific situations of the database, the time of delay may be several hours or longer. In addition, the query request sent to the database here can be either a plurality of sub-requests after splitting or a complete initial query request. This does not need to be limited in the present application.

In a specific implementation of the present application, for the delayed query processing, a query may be initiated to a backup library of the database, and the data of the backup library is periodically synchronized with that of a main library (i.e., the aforementioned database). Because it is not required to immediately return the query result, the query may be initiated to the backup library after the completion of synchronization, thus ensuring accuracy of data without affecting other normal read and write operations of the main library. In practical application, many backup libraries support distributed processing. In this case, the query operation initiated to the backup library may be achieved in a Map-Reduce manner using a Hadoop cluster, thereby more efficiently solving the problem of processing massive data.

In some systems, there is a certain requirement on the return time for the query request. If the database cannot return data on time, the application side may regard this case as a timeout of data return, causing the data query function to be actually unavailable. Considering that the delayed query processing will almost inevitably lead to a timeout, in one implementation of the present

application, for a delayed query request, a special delayed response identifier may be set, to indicate that the query request does not need to be processed immediately, thereby avoiding that the application side regards this case as a timeout of data return.

Of course, from the point of view of user experience, during delayed query processing, an appropriate prompt should be further issued to the user. For example, after it is determined that the delayed query processing needs to be performed, the user is notified that: the query result cannot be returned immediately due to the excessively large query data amount. If necessary, the user may be further informed of when to download the query result, or after the query is completed, the query result is sent to the user directly in the form of a document, or the user is informed, by means of an SMS message, an instant message or other manners, of downloading the query result at a specified address.

The above describes three types of event record query processing logic. It can be understood that, in practical application, it is not required to use the three types of processing logic at the same time. Those skilled in the art may select a part of the three types of processing logic independently according to actual needs. This does not need to be limited in the present application.

Corresponding to the above method embodiment, the present application further provides an event record query apparatus. Referring to FIG. 4, the apparatus may include:

a request receiving module 210 configured to receive an event record query request of a user and determine a query event and a query time range corresponding to the request;

a data amount estimation module 220 configured to estimate a query data amount corresponding to the query time range according to a statistical value of a historical number of occurrences of the query event of the user;

a database performance determination module 230 configured to determine a maximum single-query data amount Max currently allowable by a database;

a request splitting module 240 configured to split the query request into N query sub-requests, such that each query sub-request separately corresponds to a different time period within the query time range, and a query data amount corresponding to each time period is not greater than Max,

where N is an integer not less than 1; and

a query module 250 configured to initiate a query to the database by using the N query sub-requests.

According to a specific implementation of the present application, the statistical value of the
5 historical number of occurrences may include:

an average value of the number of occurrences in a unit period for statistics,

or

a maximum value of the number of occurrences in a unit period for statistics.

According to a specific implementation of the present application, the data amount estimation
10 module 220 may be specifically configured to:

judge whether the statistical value of the historical number of occurrences of the query event
of the user is not less than C1 and not greater than C2, and if yes, further estimate the query data
amount corresponding to the query time range;

where C1 and C2 are a preset first threshold and a preset second threshold respectively.

According to a specific implementation of the present application, the request splitting module
15 240 may be further configured to:

directly split the query request into N query sub-requests according to a preset time period
length if the statistical value of the historical number of occurrences of the query event of the user
is less than C1.

Referring to FIG. 5, according to a specific implementation of the present application, the event
20 record query apparatus may further include:

a delayed query module 260 configured to perform delayed query processing on the query
request if the statistical value of the historical number of occurrences of the query event of the user
is greater than C2.

According to a specific implementation of the present application, the delayed query module
25 260 may be specifically configured to:

initiate a query to a backup library of the database;

or

set a delayed response identifier for the query request.

For details of the implementation process of the function and use of each module in the foregoing apparatus, refer to the implementation process of the corresponding steps in the foregoing method. Details are not described here again.

Through the description of the foregoing implementations, those skilled in the art can clearly understand that the present application can be implemented by software plus a necessary universal hardware platform. Based on such an understanding, the technical solution of the present application essentially, or the portion contributing to the prior art may be embodied in the form of a software product. The software product may be stored in a storage medium, such as a ROM/RAM, a magnetic disk, or an optical disc, and include several instructions that enable a computer device (which may be a personal computer, a server, or a network device) to execute the method in the embodiments or certain portions of the embodiments of the present application.

The embodiments in the specification are described progressively. Identical or similar parts of the embodiments may be obtained with reference to each other, and each embodiment emphasizes a part different from other embodiments. Especially, the apparatus embodiment is basically similar to the method embodiment, and therefore is described in a relatively simple manner. For the related parts, refer to the description of the parts in the method embodiment. The apparatus embodiment described above is merely illustrative. Modules described as separate parts may be or may not be physically separated, and functions of each module may be implemented in the same or different software and/or hardware during implementation of the solution of the present application. The objective of the solution of this embodiment may also be implemented by selecting some of or all modules according to actual requirements. Those of ordinary skill in the art can understand and implement the present application without paying any creative efforts.

The above described are merely specific embodiments of the present application. It should be noted that, for those of ordinary skill in the art, several improvements and modifications can be

made without departing from the principle of the present application, and these improvements and modifications shall also be deemed as the protection scope of the present application.

CLAIMS

1. An event record query method, comprising:

receiving an event record query request of a user and determining a query event and a query time range corresponding to the request;

5 estimating a query data amount corresponding to the query time range according to a statistical value of a historical number of occurrences of the query event of the user;

determining a maximum single-query data amount Max currently allowable by a database;

splitting the query request into N query sub-requests, such that each query sub-request separately corresponds to a different time period within the query time range, and a query data
10 amount corresponding to each time period is not greater than Max, wherein N is an integer not less than 1; and

initiating a query to the database by using the N query sub-requests.

2. The method of claim 1, wherein the statistical value of the historical number of occurrences comprises:

15 an average value of the number of occurrences in a unit period for statistics,

or

a maximum value of the number of occurrences in a unit period for statistics.

3. The method of claim 1, wherein the step of estimating a query data amount corresponding to the query time range according to a statistical value of a historical number of occurrences of the
20 query event of the user comprises:

judging whether the statistical value of the historical number of occurrences of the query event of the user is not less than C1 and not greater than C2, and if yes, further estimating the query data amount corresponding to the query time range;

wherein C1 and C2 are a preset first threshold and a preset second threshold respectively.

25 4. The method of claim 3, further comprising:

directly splitting the query request into N query sub-requests according to a preset time period length if the statistical value of the historical number of occurrences of the query event of the user

is less than C1; and

initiating a query to the database by using the N query sub-requests.

5. The method of claim 3, further comprising:

performing delayed query processing on the query request if the statistical value of the
5 historical number of occurrences of the query event of the user is greater than C2.

6. The method of claim 5, wherein the step of performing delayed query processing on the
query request comprises:

initiating a query to a backup library of the database.

7. The method of claim 5, wherein the step of performing delayed query processing on the
10 query request comprises:

setting a delayed response identifier for the query request.

8. An event record query apparatus, comprising:

a request receiving module configured to receive an event record query request of a user and
determine a query event and a query time range corresponding to the request;

15 a data amount estimation module configured to estimate a query data amount corresponding
to the query time range according to a statistical value of a historical number of occurrences of the
query event of the user;

a database performance determination module configured to determine a maximum single-
query data amount Max currently allowable by a database;

20 a request splitting module configured to split the query request into N query sub-requests,
such that each query sub-request separately corresponds to a different time period within the query
time range, and a query data amount corresponding to each time period is not greater than Max,
wherein N is an integer not less than 1; and

a query module configured to initiate a query to the database by using the N query sub-
25 requests.

9. The apparatus of claim 8, wherein the statistical value of the historical number of
occurrences comprises:

an average value of the number of occurrences in a unit period for statistics,

or

a maximum value of the number of occurrences in a unit period for statistics.

10. The apparatus of claim 8, wherein the data amount estimation module is specifically
5 configured to:

judge whether the statistical value of the historical number of occurrences of the query event
of the user is not less than C1 and not greater than C2, and if yes, further estimate the query data
amount corresponding to the query time range;

wherein C1 and C2 are a preset first threshold and a preset second threshold respectively.

11. The apparatus of claim 10, wherein the request splitting module is further configured to:
10 directly split the query request into N query sub-requests according to a preset time period
length if the statistical value of the historical number of occurrences of the query event of the user
is less than C1.

12. The apparatus of claim 10, further comprising:

15 a delayed query module configured to perform delayed query processing on the query request
if the statistical value of the historical number of occurrences of the query event of the user is
greater than C2.

13. The apparatus of claim 12, wherein the delayed query module is specifically configured
to:

20 initiate a query to a backup library of the database.

14. The apparatus of claim 12, wherein the delayed query module is specifically configured
to:

set a delayed response identifier for the query request.