

ONLINE EVOLUTION METHOD AND APPARATUS FOR EXECUTION PLAN, DEVICE, AND STORAGE MEDIUM

TECHNICAL FIELD

[0001] This specification relates to the field of database processing technologies, and in particular, to an online evolution method and apparatus for an execution plan, a device, and a storage medium.

BACKGROUND

[0002] During execution of a database statement, a specific process includes: generating an execution plan corresponding to the database statement, and executing the generated execution plan. The same database statement may have different execution plans, and there is a performance difference between different execution plans. To prevent an execution plan with good performance from being replaced by an execution plan with poor performance, in a related technology, a database plan management function is introduced into some databases to resolve a problem of plan rollback. A core idea of the database plan management function is to always execute an execution plan used in the past instead of executing a new execution plan.

[0003] However, the database plan management function of the existing database only records a new execution plan when the new execution plan appears. If a user does not actively trigger an evolution task to verify performance of the new execution plan, the database will never use the new execution plan, even if the performance of the new execution plan is better than that of the execution plan that has been executed previously.

SUMMARY

[0004] To overcome a problem in a related technology, this specification provides an online evolution method and apparatus for an execution plan, a device, and a storage medium.

[0005] According to a first aspect of embodiments of this specification, an online evolution method for an execution plan is provided. The method includes: in response to a received current

database request including a target database statement, obtaining an evolution task from a plan cache corresponding to the target database statement, where the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement; selecting a type of an execution plan actually used by the current database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with performance of each type of execution plan; and after the evolution task ends, if it is determined that the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, changing the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

[0006] According to a second aspect of embodiments of this specification, an online evolution apparatus for an execution plan is provided. The apparatus includes: an obtaining module, configured to: in response to a received current database request including a target database statement, obtain an evolution task from a plan cache corresponding to the target database statement, where the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement; a selection module, configured to select a type of an execution plan actually used by the current database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with performance of each type of execution plan; and a determining module, configured to: after the evolution task ends, if it is determined that the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, change the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

[0007] According to a third aspect of embodiments of this specification, an electronic device is provided. The electronic device includes: a processor; and a memory configured to store a

processor-executable instruction, where the processor runs the executable instruction to implement the online evolution method for an execution plan according to the first aspect.

[0008] According to a fourth aspect of one or more embodiments of this specification, a computer-readable storage medium is provided, where the computer-readable storage medium stores a computer instruction, and the instruction is executed by a processor to implement the steps of the online evolution method for an execution plan according to any one of the foregoing aspects.

[0009] The technical solutions provided in the embodiments of this specification may include the following beneficial effects: In the embodiments of this specification, a corresponding evolution task is recorded in a plan cache corresponding to a target database statement, so that when a database request including the target database statement is received, an execution plan of a baseline type or an evolution type may be triggered and executed based on the evolution task, so as to compare performance of the execution plan of the baseline type with performance of the execution plan of the evolution type, thereby implementing automatic evolution of the execution plan. Because the evolution task is processed for a real database request in an implementation process, instead of being additionally processed through simulation, no additional performance consumption is caused. In addition, a type of an execution plan is dynamically selected for a received database request including the target database statement, so as to ensure that an accumulative quantity of execution times of each type of execution plan in the implementation process of the evolution task is positively correlated with performance of each type of execution plan, so that the execution plan with better performance is executed for more times. If the execution plan of the evolution type has relatively poorer performance, impact of plan rollback caused by the execution plan of the evolution type may be weakened as much as possible. If the execution plan of the evolution type has relatively better performance, execution efficiency of the evolution task can be improved, and evolution can be accelerated, thereby further helping improve overall execution efficiency of the target database statement.

[0010] It should be understood that the foregoing general description and the following detailed description are merely examples and explanations, and should not limit this specification.

BRIEF DESCRIPTION OF DRAWINGS

[0011] The accompanying drawings, which are incorporated in and form a part of this

specification, illustrate some embodiments consistent with this specification and, together with this specification, serve to explain the principles of this specification.

[0012] FIG. 1 is a flowchart of an online evolution method for an execution plan according to an example embodiment of this specification;

5 **[0013]** FIG. 2 is a flowchart of generating an evolution task according to an example embodiment of this specification;

[0014] FIG. 3 is a flowchart of an evolution task according to an example embodiment of this specification;

[0015] FIG. 4 is a block diagram of an online evolution apparatus for an execution plan
10 according to an example embodiment of this specification; and

[0016] FIG. 5 is a schematic structural diagram of an electronic device according to an example embodiment of this specification.

DESCRIPTION OF EMBODIMENTS

[0017] Some example embodiments are described in detail here, and examples of the example
15 embodiments are presented in the accompanying drawings. When the following description relates to the accompanying drawings, unless specified otherwise, same numbers in different accompanying drawings represent same or similar elements. Implementations described in the following example embodiments do not represent all implementations consistent with this specification. On the contrary, the implementations are merely examples of apparatuses and
20 methods that are described in the appended claims in detail and consistent with some aspects of this specification.

[0018] The terms used in this specification are merely for illustrating specific embodiments, and are not intended to limit this specification. The terms "a" and "the" of singular forms used in this specification and the appended claims are also intended to include plural forms, unless
25 otherwise specified in the context clearly. It should be further understood that the term "and/or" used in this specification indicates and includes any or all possible combinations of one or more associated listed items.

[0019] It should be understood that although terms "first", "second", "third", and the like may be used in this specification to describe various types of information, the information should not

be limited by these terms. These terms are used merely to differentiate information of the same type. For example, without departing from the scope of this specification, first information may also be referred to as second information, and similarly, the second information may also be referred to as the first information. Depending on the context, for example, the word "if" used here
5 may be explained as "while", "when", or "in response to determining".

[0020] A database is an ordered set of structured information or data, and the ordered set is usually controlled by a database management system (Database Management System, DBMS). In reality, data, database management systems, and associated applications are collectively referred to as database systems, which are usually referred to as databases for short. When a user needs to
10 perform an operation on a database, for example, accessing, managing, modifying, updating, controlling, or organizing data, a database statement is usually used for implementation. For example, a database statement complying with an SQL (structured query language, Structured Query Language) standard is used in a relational database, which may also be referred to as an SQL statement.

[0021] The database statement used by the user may indicate a database table that needs to be operated, and an operator that needs to be used for the database table. The operator is used to indicate an operation to be performed for a corresponding database table. However, the database statement can only express "what the user wants" to the database, but cannot directly express, to the database, "how to obtain" the result. In addition, there are a plurality of possible connection
20 sequences between database tables during implementation of a connection operation, and there are a plurality of specific implementation algorithms for each operator. Therefore, there may be many equivalent methods for obtaining the result for one database statement, and a query optimizer in the database needs to determine a method for obtaining the result. Here, each method for obtaining the result is referred to as one "execution plan". In other words, an execution plan may refer to a
25 method used by a database to access data.

[0022] Generally, when generating a new execution plan for a database request, the database always directly executes the new execution plan. As a result, when performance of the new execution plan is poorer than performance of a previous execution plan, plan rollback is caused, affecting execution efficiency of a subsequent database statement.

[0023] Based on the foregoing problems, a database plan management function that is evolved offline is proposed for a common database. A general procedure includes the following: After the
30

database generates a new execution plan for the same database request, the new execution plan is recorded. Subsequently, the user actively initiates an evolution task. The database receives the database request in the background through simulation, executes the new execution plan and the previously used execution plan separately, and determines, through comparison, which of the two execution plans has better performance. In this offline evolution processing manner, an execution process of an evolution task causes additional resource consumption to implement simulation processing on the database request. In addition, the new execution plan is not used to execute the database request actually received by the database, but the previous execution plan is still executed. Therefore, even if the new execution plan has higher execution efficiency, the new execution plan cannot immediately take effect, thereby reducing the processing efficiency of the real database request.

[0024] In view of the foregoing description, this specification provides an online evolution method for an execution plan, so as to improve efficiency of an evolution task and implement automatic execution of an optimal plan while avoiding plan rollback, thereby improving execution efficiency of a subsequent database statement.

[0025] Some embodiments of this specification are described in detail below.

[0026] FIG. 1 is a flowchart of an online evolution method for an execution plan according to an example embodiment of this specification. As shown in FIG. 1, the method includes the following steps S101 to S103.

[0027] S101: In response to a received current database request including a target database statement, obtain an evolution task from a plan cache corresponding to the target database statement, where the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement.

[0028] A user-side device sends the database request including the database statement, and a database-side device receives and responds to the database request. In an embodiment of this specification, the database-side device may be a device used to deploy a database. After receiving a database request, the device directly executes an execution plan corresponding to a database statement locally. Alternatively, the database-side device may be specially used to connect to the

user-side device and a device on which a database is deployed. After receiving a database request from the user-side device, the database-side device sends the request to the device on which a database is deployed. Finally, the device on which a database is deployed executes an execution plan corresponding to a database statement.

5 **[0029]** The database-side device stores a plan cache corresponding to the database statement, and the plan cache may include the plan cache corresponding to the target database statement described in an embodiment of this specification. Different database statements may respectively correspond to different plan caches. The plan cache is used to record an execution plan corresponding to a corresponding database statement. The execution plan may be specifically
10 presented in a plurality of forms, for example, a generated execution plan of a baseline type, or an evolution task in an embodiment of this specification. The evolution task further relates to an execution plan of a baseline type and an execution plan of an evolution type. In fact, the plan cache may be considered as a cache for an execution plan, so that when processing the database statement in the database request, the database-side device may directly read the execution plan in the plan
15 cache for execution, without temporarily generating an execution plan.

[0030] Content stored in the plan cache is dynamically changed. When the database-side device receives the database request including the target database statement for the first time, the plan cache corresponding to the target database statement certainly does not include any execution plan. The database-side device needs to temporarily generate an execution plan for the target
20 database statement. Because the processing is performed for the first time, the execution plan of the baseline type corresponding to the target database statement certainly does not exist. Therefore, the temporarily generated execution plan is directly used as the execution plan of the baseline type corresponding to the target database statement, and the execution plan is stored in the plan cache.

[0031] The execution plan of the baseline type stored in the plan cache also complies with
25 some processing mechanisms of the "cache", such as an aging mechanism. Specifically, when the execution plan of the baseline type stored in the plan cache is not executed within preset duration, the execution plan of the baseline type is deleted from the plan cache, thereby saving cache space of the database-side device and improving cache utilization. In addition to the foregoing aging mechanism, some database changes or the like may also cause the execution plan of the baseline
30 type in the plan cache to be eliminated. For example, the changes may include creation of an index, which is not limited in this specification.

[0032] Certainly, even if the execution plan of the baseline type in the plan cache is deleted, it should also be considered that the target database statement has a corresponding execution plan of the baseline type, but in a specific execution process, the execution plan of the baseline type needs to be temporarily generated and cannot be read from the plan cache. Therefore, when the database-side device subsequently receives the database request including the target database statement, if the execution plan of the baseline type is read from the plan cache corresponding to the target database statement, the read execution plan is directly executed. If the execution plan of the baseline type does not exist in the plan cache corresponding to the target database statement, an execution plan may need to be temporarily generated, and it needs to be determined whether the temporarily generated execution plan is consistent with the execution plan of the baseline type corresponding to the target database statement. If the temporarily generated execution plan is consistent with the execution plan of the baseline type, it indicates that the temporarily generated execution plan is an execution plan of the baseline type; otherwise, it indicates that the temporarily generated execution plan is an execution plan of the evolution type. In an embodiment, when a database change such as the above-described creation of an index takes place, the execution plan of the evolution type rather than the original execution plan of the baseline type is generated for the target database statement. For the generated execution plan of the evolution type, the database-side device further generates an evolution task, so as to evolve for the execution plan of the target database statement.

[0033] In an embodiment of this specification, the evolution task is used to compare execution performance of the execution plan of the evolution type with execution performance of the execution plan of the baseline type. When the database generates a new execution plan for a database request, the generated execution plan is an execution plan of the evolution type, and the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement. However, the execution plan used before the database generates the new execution plan is an execution plan of the baseline type, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement.

[0034] In an embodiment of this specification, the generated new execution plan corresponding to the target database statement may be verified in two manners. A first manner is that based on a performance comparison result obtained by the evolution task, if the performance of the execution plan of the evolution type is better than the performance of the execution plan of

the baseline type, the performance of the execution plan of the evolution type is verified, so that the execution plan of the evolution type may be changed into an execution plan of the baseline type. A second manner is that when the target database statement does not have the corresponding execution plan of the baseline type, because there is no comparable execution plan of the baseline type, the execution plan of the evolution type may be directly considered as having been verified, and therefore may be changed into an execution plan of the baseline type. As described above, after receiving the database request including the target database statement for the first time, the database-side device considers the new execution plan as an execution plan of the baseline type based on the second case.

[0035] In addition, the execution plan of the baseline type corresponding to the target database statement may also have an elimination mechanism. For example, if unused duration of any execution plan of the baseline type corresponding to the target database statement reaches a preset duration threshold, it indicates that the any execution plan of the baseline type has low popularity (possibly due to insufficient performance or the like). To prevent storage space of the database-side device from being occupied (used to record the execution plan of the baseline type corresponding to the target database statement), the any execution plan of the baseline type may be deleted. If all execution plans of the baseline type corresponding to the target database statement are deleted, the target database statement is restored to a state in which there is no corresponding execution plan of the baseline type. It can be learned that the target database statement may have one or more execution plans of the baseline type at the same time. For example, an execution plan 1 of the baseline type originally exists, and an execution plan of the evolution type becomes an execution plan 2 of the baseline type after being verified through an evolution task. Similarly, there may be more execution plans of the baseline type. The execution plan 1 of the baseline type is used as an example. If the execution plan 1 of the baseline type is not used for a long time, the execution plan 1 of the baseline type may be deleted based on the foregoing elimination mechanism, so that the target database statement has only a commonly used execution plan of the baseline type, that is, the execution plan 2 of the baseline type. In this way, the execution plan of the baseline type is eliminated, and a transition period of a preset duration threshold is provided for the execution plan 1 of the baseline type, so that when the execution plan 1 of the baseline type is temporarily switched back for use, there is no need to perform an implementation process of an evolution task again, thereby reducing resource consumption.

[0036] After generating an evolution task corresponding to a database statement, the database-side device stores the evolution task in a plan cache corresponding to the database statement. In the foregoing embodiment, when a current database request is received, a plan cache corresponding to a target database statement already includes a corresponding evolution task, which indicates that the database-side device has previously generated an evolution task for the target database statement. The database-side device may generate an evolution task corresponding to the target database statement based on a previously received historical database request including the target database statement, and store the evolution task in a corresponding plan cache. For example, in response to the received historical database request including the target database statement, a plan cache corresponding to the target database statement is queried, and when the plan cache does not include a corresponding evolution task and a generated execution plan, a database statement execution plan corresponding to the target database statement is generated. The plan cache includes either a generated execution plan or an evolution task, and does not include both of the generated execution plan and the evolution task. If the generated database statement execution plan is an execution plan of the evolution type, the evolution task is generated and cached into the plan cache corresponding to the target database statement. In a processing procedure for the historical database request, if the plan cache corresponding to the target database statement includes an evolution task, the evolution task is directly executed without triggering generation of a new evolution task. If the plan cache corresponding to the target database statement includes a generated execution plan, the execution plan is directly executed without triggering generation of a new execution plan or evolution task. This is consistent with the foregoing process of generating an evolution task. A new execution plan is generated for the target database statement only when the plan cache corresponding to the target database statement includes neither an evolution task nor a generated execution plan. In addition, generation of an evolution task is triggered only when the new execution plan is an execution plan of an evolution type.

[0037] A person skilled in the art should understand that, in an embodiment of this specification, the current database request and the historical database request are merely relative concepts, so as to help distinguish a sequence of receiving different database requests. Specifically, the database-side device first receives the historical database request, generates an evolution task based on the historical database request, and then receives the current database request.

[0038] In an example embodiment, the current database request may alternatively not have a

corresponding historical database request, or the historical database request may not be used to generate an evolution task. For example, in response to the received current database request including the target database statement, a plan cache corresponding to the target database statement is queried, and when the plan cache does not include a corresponding evolution task and a generated execution plan, a database statement execution plan corresponding to the target database statement is generated, which is equivalent to the foregoing new execution plan or another similar concept.

[0039] In one case, there is an execution plan of a baseline type corresponding to the target database statement, that is, it indicates that the current database request is not the database request that includes the target database statement and that is received by the database-side device for the first time, and an execution plan of the baseline type stored in the plan cache may be deleted for an aging mechanism or another reason. Therefore, the foregoing database statement execution plan needs to be temporarily generated. In this case, the generated database statement execution plan may be compared with the execution plan of the baseline type. When a comparison result is that the two execution plans are consistent, it indicates that the generated database statement execution plan is an execution plan of the baseline type. In this case, the database statement execution plan is executed and cached into the plan cache corresponding to the target database statement. Hash values of the generated database statement execution plan and the execution plan of the baseline type may be compared. If the hash values are consistent, it indicates that a comparison result is that the two execution plans are consistent; otherwise, the two execution plans are inconsistent. Certainly, in addition to the comparison based on the hash value, the two execution plans may be compared in any other manner, which is not limited in this specification.

[0040] In another case, there is no execution plan of a baseline type corresponding to the target database statement, that is, it indicates that the current database request is the database request that includes the target database statement and that is received by a database-side device for the first time, or is not the database request that includes the target database statement and that is received for the first time, but the execution plan of the baseline type corresponding to the target database statement is deleted for an elimination mechanism or another reason. Therefore, the foregoing database statement execution plan needs to be temporarily generated. In this case, the generated database statement execution plan may be added to the execution plan of the baseline type, the database statement execution plan may be executed, and the database statement execution plan

may be cached into the plan cache corresponding to the target database statement.

[0041] S102: Select a type of an execution plan actually used by the current database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with performance of each type of execution plan.

[0042] As described above, the type of the execution plan includes the baseline type and the evolution type. Therefore, the execution plan selected for the current database request is either an execution plan of the evolution type or an execution plan of the baseline type. In the implementation process of the evolution task corresponding to the target database statement, each time one database request including the target database statement is received, the evolution task is triggered to be executed for one round, and an execution plan of the evolution type or an execution plan of the baseline type is selected and executed once for each round.

[0043] In an embodiment of this specification, a type of an execution plan to be selected for each round may be determined in any manner, as long as it is ensured that overall statistical data can meet a condition that the accumulative quantity of execution times of each type of execution plan is positively correlated with the performance of each type of execution plan. A manner of selecting the execution plan is not limited in this specification.

[0044] An accumulative quantity of execution times of each execution plan type is set to be positively correlated with performance of each execution plan type, so that an execution plan type with poorer execution performance has a smaller accumulative quantity of execution times, and an execution plan type with better execution performance has a larger accumulative quantity of execution times, thereby minimizing negative impact caused by an execution plan with poor execution performance, quickly verifying a better execution plan, and improving execution efficiency of an evolution task. If the execution efficiency of the execution plan of the evolution type is relatively higher, it is proved that the performance of the execution plan of the evolution type is relatively better. This embodiment of this specification is based on a technical solution of online evolution. To be specific, the type of the execution plan is dynamically selected and practically executed for each received database request including the target database statement. Therefore, when the performance of the execution plan of the evolution type is relatively better, execution efficiency of some database requests that include the target database statement and that are executed by selecting the execution plan of the evolution type can be improved. In addition, in

the implementation process of the evolution task, the execution plan of the evolution type is more likely to be selected, so that overall execution efficiency of the database request can be greatly improved. If the execution efficiency of the execution plan of the evolution type is relatively lower, it is proved that the performance of the execution plan of the baseline type is relatively better.

5 Based on a limitation that the accumulative quantity of execution times of the execution plan type is positively correlated with the performance of the execution plan type, the accumulative quantity of execution times of the execution plan of the evolution type in the overall implementation process of the evolution task is reduced, thereby minimizing negative impact caused by an execution plan with poor performance, ensuring that plan rollback is nearly not perceived, quickly verifying a
10 better execution plan when there are only a small quantity of controllable plan rollbacks, and improving execution efficiency of an evolution task.

[0045] In an example embodiment, the implementation process of the evolution task may include a plurality of stages. In an initial stage, a quantity of execution times of the execution plan of the evolution type is the same as a quantity of execution times of the execution plan of the
15 baseline type, so that the execution plan of the evolution type and the execution plan of the baseline type can be visually compared to determine execution efficiency of the two execution plans in the initial stage. The quantity of execution times of each type of execution plan may be any value. For example, each type of execution plan is executed for 10 times, or less or more times, which is not limited in this specification.

20 **[0046]** In any other stage other than the initial stage, the quantity of execution times of each type of execution plan is negatively correlated with a proportion of average execution duration of each type of execution plan in at least one stage before the any other stage. Assuming that there are a total of N stages, where $N > 1$, and is a positive integer, stages other than the initial stage include a second stage to an Nth stage. Certainly, if $N = 2$, the stages other than the initial stage
25 include only the second stage. A value of N is not limited in this specification.

[0047] For any other stage, an ith stage ($1 < i \leq N$, where i is a positive integer) is used as an example. All stages before the ith stage include a first stage (namely, the initial stage) to an (i-1)th stage. In this case, at least one stage before the ith stage is at least one stage from the first stage to the (i-1)th stage. For example, when $i = 2$, at least one stage before the second stage is the first
30 stage. For another example, when $i = 3$, at least one stage before the third stage may be the first stage, may be the second stage, or may include both the first stage and the second stage. The rest

are deduced by analogy, and are not listed one by one here. Certainly, regardless of how the at least one stage is selected, it should be ensured that the same stage is selected for each type of execution plan, so that average execution duration of corresponding calculation is comparable.

[0048] Average execution duration of each type of execution plan in at least one stage before any other stage may be a ratio of total execution duration of the type of execution plan in the at least one stage to a quantity of execution times, that is, average duration of executing the type of execution plan once in the at least one stage. A person skilled in the art should understand that, when average execution duration of a type of execution plan in at least one stage before the any other stage is longer, it indicates that execution efficiency of the type of execution plan is lower, and when the average execution duration is shorter, it indicates that the execution efficiency of the type of execution plan is higher. Therefore, a quantity of execution times of each type of execution plan is assumed to be negatively correlated with a proportion of average execution duration of each type of execution plan in at least one stage before the any other stage, so that an execution plan with relatively longer average execution duration and relatively lower execution efficiency is executed for relatively less times in the any other stage, and an execution plan with relatively shorter average execution duration and relatively higher execution efficiency is executed for relatively more times in the any other stage. This ensures that a probability of selecting and executing an execution plan with good performance continuously increases, and a probability of selecting and executing an execution plan with poor performance continuously decreases, thereby helping alleviate impact of plan rollback caused by the execution plan with poor performance, and improve execution efficiency of an evolution task.

[0049] Certainly, the evolution task may not be divided into the foregoing plurality of stages. This is equivalent to including only one stage. In this case, in the implementation process of the evolution task, execution efficiency of each type of execution plan may be dynamically accumulated, for example, average execution duration of each type of execution plan is accumulated, and each type of execution plan may be selected and executed based on the execution efficiency of each type of execution plan. This ensures that in the overall implementation process of the evolution task, an accumulative quantity of execution times of each type of execution plan is positively correlated with performance of each type of execution plan. To be specific, poorer execution performance indicates a smaller accumulative quantity of execution times, and better execution performance indicates a larger accumulative quantity of execution times.

[0050] In an example embodiment, after the execution plan selected for the current database request is completed, it is determined whether the evolution task meets a predefined evolution end condition, where the evolution end condition may include that a quantity of evolution times reaches a threshold and/or evolution duration reaches a threshold. Certainly, another evolution end condition may alternatively be used, which is not limited in this specification.

[0051] S103: After the evolution task ends, if it is determined that the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, change the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

[0052] In an example embodiment, average execution duration of the execution plan of the evolution type and average execution duration of the execution plan of the baseline type are separately determined based on an execution status of the evolution task; and either the execution plan of the evolution type or the execution plan of the baseline type, whichever has relatively shorter corresponding average execution duration, is determined as the execution plan with relatively better performance. For example, it is assumed that average execution duration of the execution plan of the evolution type for each execution time in the overall implementation process of the evolution task is $T1$, and average execution duration of the execution plan of the baseline type for each execution time in the overall implementation process of the evolution task is $T2$. Therefore, when $T1 < T2$, it indicates that the execution efficiency of the execution plan of the evolution type is higher than the execution efficiency of the execution plan of the baseline type. In this case, it may be determined that the performance of the execution plan of the evolution type is relatively better, and the performance of the execution plan of the baseline type is relatively poorer. When $T1 > T2$, it indicates that the execution efficiency of the execution plan of the baseline type is higher than the execution efficiency of the execution plan of the evolution type. In this case, it may be determined that the performance of the execution plan of the baseline type is relatively better, and the performance of the execution plan of the evolution type is relatively poorer. When $T1 = T2$, it may be determined, based on a predefined processing policy, whether to consider the execution plan of the evolution type as having passed verification.

[0053] A performance comparison result is obtained by using an evolution task. If the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, the performance of the execution plan of the evolution type

has been verified, and the execution plan of the evolution type is changed into an execution plan of the baseline type corresponding to the target database statement, and cached into a plan cache corresponding to the target database statement. If the performance of the execution plan of the baseline type is better than the performance of the execution plan of the evolution type, the execution plan of the evolution type with poorer performance is discarded, and the execution plan of the baseline type is cached into the plan cache corresponding to the target database statement. When the database-side device receives the database request including the target database statement again, the database-side device may directly execute an execution plan that is cached in the plan cache corresponding to the target database statement. Performance merits of the execution plan of the evolution type and the execution plan of the baseline type are quickly determined by using the evolution task, and an execution plan with relatively better performance is used as a subsequent execution plan corresponding to the target database statement, thereby improving overall execution efficiency of the target database statement.

[0054] In the embodiments of this specification, a corresponding evolution task is recorded in a plan cache corresponding to a target database statement, so that when a database request including the target database statement is received, an execution plan of a baseline type or an evolution type may be triggered and executed based on the evolution task, so as to compare performance of the execution plan of the baseline type with performance of the execution plan of the evolution type, thereby implementing automatic evolution of the execution plan. Because the evolution task is processed for a real database request in an implementation process, instead of being additionally processed through simulation, no additional performance consumption is caused. In addition, a type of an execution plan is dynamically selected for a received database request including the target database statement, so as to ensure that an accumulative quantity of execution times of each type of execution plan in the implementation process of the evolution task is positively correlated with performance of each type of execution plan, so that the execution plan with better performance is executed for more times. If the execution plan of the evolution type has relatively poorer performance, impact of plan rollback caused by the execution plan of the evolution type may be weakened as much as possible. If the execution plan of the evolution type has relatively better performance, execution efficiency of the evolution task can be improved, and evolution can be accelerated, thereby further helping improve overall execution efficiency of the target database statement.

[0055] When the execution plan of the baseline type is better than the execution plan of the evolution type, the execution plan of the evolution type can be eliminated in a timely manner, and impact of plan rollback caused by the execution plan with poor performance is weakened. In addition, when the execution plan of the evolution type is better than the execution plan of the baseline type, the better execution plan can be used in a timely manner, and execution efficiency of some database requests executed by selecting the execution plan of the evolution type can be improved. Finally, when there are only a small quantity of controllable plan rollbacks, a better execution plan is quickly verified, thereby improving execution efficiency of a database statement.

[0056] FIG. 2 is a flowchart of generating an evolution task according to an example embodiment of this specification. As shown in FIG. 2, the flowchart includes specific steps S201 to S210.

[0057] S201: Receive a database request. A database receives a database request that is sent by a user and that includes a target database statement.

[0058] S202: Determine whether a plan cache includes a generated execution plan or an evolution task. A plan cache corresponding to the target database statement is queried, and it is determined whether the plan cache includes a generated execution plan or a corresponding evolution task corresponding to the target database request. If no, step S203 is performed. If yes, step S204 is performed.

[0059] S203: Generate a new execution plan. If the plan cache corresponding to the target database statement does not include an available execution plan or an evolution task corresponding to the target database request, in this case, the database generates a new database statement execution plan for the database request including the target database statement.

[0060] S204: Execute the generated execution plan or the evolution task. If the plan cache corresponding to the target database statement includes the generated execution plan or the evolution task corresponding to the target database request, the corresponding generated execution plan or evolution task is directly executed.

[0061] S205: Determine whether there is an execution plan of a baseline type. It is determined whether the target database statement has the execution plan of the baseline type. If yes, step S206 is performed. If no, step S207 is performed.

[0062] S206: Determine whether the execution plan is an execution plan of the baseline type. When the target database statement has an execution plan of the baseline type, it is determined

whether a new execution plan corresponding to the target database statement is an execution plan of the baseline type. If yes, step S209 is performed. If no, step S208 is performed.

[0063] S207. Add an execution plan to the execution plan of the baseline type. When the target database statement does not have an execution plan of the baseline type, the new execution plan corresponding to the target database statement is added to the execution plan of the baseline type, and is used to determine, when subsequently receiving a database request including the target database statement again, and generating a new execution plan, whether the new execution plan is an execution plan of the baseline type.

[0064] S208: Generate an evolution task. If the new execution plan corresponding to the target database statement is not an execution plan of the baseline type, the new execution plan is determined to be an execution plan of an evolution type, data of the execution plan of the evolution type and an execution plan of the baseline type with optimal performance is saved, and the execution plan of any one of these types is executed.

[0065] S209: Execute a current execution plan. When it is determined that the new execution plan corresponding to the target database statement is an execution plan of the baseline type or is added to the execution plan of the baseline type, the current execution plan is executed, and step 210 is performed.

[0066] S210: Add an execution plan to the plan cache. The evolution task that corresponds to the target database request and that is generated in step S208 or the execution plan that corresponds to the target database request and that is generated in step S209 is added to the plan cache corresponding to the target database request.

[0067] For an implementation process of the foregoing steps, references are made to an implementation process of corresponding steps in the embodiment in FIG. 1, and details are not described here again.

[0068] FIG. 3 is a flowchart of an evolution task according to an example embodiment of this specification. The flowchart includes specific steps S301 to S311.

[0069] S301: Receive a database request. When a current database request including a target database statement is received, and a plan cache corresponding to the target database statement includes an evolution task, the evolution task is obtained.

[0070] S302: Select a type of an execution plan. An execution plan of an evolution type or an execution plan of a baseline type is dynamically selected and executed.

[0071] S303: Execute an execution plan of the selected type. The execution plan of the selected type is executed and duration of the current execution is recorded.

[0072] S304: Determine whether an evolution end condition is met. If the evolution end condition is met, step S305 is performed. If the evolution end condition is not met, step S306 is performed.

[0073] S305: Terminate an evolution task. The evolution task is terminated, and the evolution task is deleted from the plan cache corresponding to the target database statement.

[0074] S306: Continue the evolution task. Execution of the current database request including the target database statement is completed, but the evolution task is still retained in the plan cache corresponding to the target database statement, so as to be used to receive a database request including the target database statement next time.

[0075] S307: Determine whether the execution plan of the evolution type is better than the execution plan of the baseline type. If the execution plan of the evolution type is better than the execution plan of the baseline type, step S308 is performed. If the execution plan of the baseline type is better than the execution plan of the evolution type, step S310 is performed.

[0076] S308: Change the execution plan of the evolution type into an execution plan of the baseline type. The execution plan of the evolution type is changed into an execution plan of the baseline type corresponding to the target database statement.

[0077] S309: Add an execution plan to the plan cache. The execution plan of the baseline type, into which the execution plan of the evolution type is changed, corresponding to the target database statement is used as the execution plan corresponding to the target database statement and is added to the plan cache corresponding to the target database statement.

[0078] S310: Discard the execution plan of the evolution type and use the execution plan of the baseline type.

[0079] S311: Add an execution plan to the plan cache. The execution plan of the baseline type is used as the execution plan corresponding to the target database statement and is added to the plan cache corresponding to the target database statement.

[0080] For an implementation process of the foregoing steps, references are made to an implementation process of corresponding steps in the embodiment in FIG. 1, and details are not described here again.

[0081] Corresponding to some embodiments of the above-described method, this specification

further provides some embodiments of an apparatus and a terminal to which the apparatus is applied.

[0082] FIG. 4 is a block diagram of an apparatus according to an example embodiment of this specification. As shown in FIG. 4, the apparatus includes: an obtaining module 401, configured to:
5 in response to a received current database request including a target database statement, obtain an evolution task from a plan cache corresponding to the target database statement, where the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement, and the execution plan
10 of the baseline type is a verified execution plan corresponding to the target database statement; a selection module 402, configured to select a type of an execution plan actually used by the database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with performance of each type of execution plan; and a determining module 403, configured to: after the evolution task ends,
15 if it is determined that the performance of the current execution plan of the evolution type is better than the performance of the execution plan of the baseline type, change the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

[0083] For an implementation process of functions and roles of each module in the apparatus,
20 references are made to an implementation process of a corresponding step in the foregoing method, and details are not described here again.

[0084] Because some apparatus embodiments basically correspond to some method embodiments, for related parts, references are made to related descriptions in the method embodiments. The described apparatus embodiments are merely examples. The modules described
25 as separate parts may or may not be physically separate, and parts displayed as modules may or may not be physical modules, that is, may be located in one position, or may be distributed on a plurality of network modules. Some or all of the modules may be selected based on actual needs to achieve the objectives of the solutions of this specification. A person of ordinary skill in the art can understand and implement the solutions without creative efforts.

[0085] FIG. 5 is a schematic structural diagram of an electronic device according to an
30 example embodiment of this specification.

[0086] References are made to FIG. 5. At the hardware level, the electronic device includes a processor 501, an internal bus 502, a network interface 503, a memory 504, and a non-volatile memory 505, and certainly may further include other hardware needed by a service. The processor 501 reads a corresponding computer program from the non-volatile memory 505 to the memory 504 and then runs the computer program. Certainly, in addition to software implementations, this specification does not preclude other implementations, such as a logic device or a combination of software and hardware. In other words, an execution body of the following processing procedure is not limited to each logical unit, and may be hardware or a logic device.

[0087] The apparatus or unit illustrated in the foregoing embodiments may be specifically implemented by using a computer chip or an entity, or may be implemented by using a product having a specific function. A typical implementation device is a computer, and a specific form of the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

[0088] In a typical configuration, the computer includes one or more processors (CPUs), one or more input/output interfaces, one or more network interfaces, and one or more memories.

[0089] The memory may include a non-persistent memory, a random access memory (RAM), a non-volatile memory, and/or another form that are in a computer-readable medium, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer-readable medium.

[0090] The computer-readable medium includes a persistent and a non-persistent, a removable and a non-removable medium, which implement information storage by using any method or technology. Information may be a computer-readable instruction, a data structure, a program module, or other data. Examples of the computer storage medium include but are not limited to a phase change random access memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), a random access memory (RAM) of another type, a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD), or another optical storage, a cassette, a disk memory, a quantum memory, a graphene-based storage medium, another magnetic storage device, or any other non-

transmission medium. The computer storage medium may be configured to store information that can be accessed by a computing device. Based on the definition in this specification, the computer-readable medium does not include transitory media (transitory media) such as a modulated data signal and carrier.

5 **[0091]** Some specific embodiments of this specification are described above. Other implementations fall within the scope of the appended claims. In some cases, actions or steps described in the claims may be performed in a sequence different from that in some embodiments and desired results can still be achieved. In addition, processes described in the accompanying drawings do not necessarily need a specific order or a sequential order shown to achieve the desired
10 results. In some implementations, multi-tasking and parallel processing are also possible or may be advantageous.

[0092] After considering the specification and practicing the invention applied here, a person skilled in the art easily figures out other implementation solutions of this specification. This specification is intended to cover any variations, purposes, or adaptive changes of this specification.
15 These variations, purposes, or adaptive changes follow the general principles of this specification and include common knowledge or conventional technical means in the art that are not applied in this specification. This specification and some embodiments are merely considered as examples, and the actual scope and the spirit of this specification are pointed out by the claims.

[0093] It should be understood that this specification is not limited to the precise structures
20 that have been described above and shown in the accompanying drawings, and various modifications and changes may be made without departing from the scope of this specification. The scope of this specification is limited by the appended claims only.

[0094] The foregoing descriptions are merely some example embodiments of this specification, but are not intended to limit this specification. Any modification, equivalent replacement, or
25 improvement made without departing from the spirit and principle of this specification shall fall within the protection scope of this specification.

CLAIMS

1. An online evolution method for an execution plan, comprising:

in response to a received current database request comprising a target database statement, obtaining an evolution task from a plan cache corresponding to the target database statement, wherein the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target database statement, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement;

selecting a type of an execution plan actually used by the current database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with performance of each type of execution plan; and

after the evolution task ends, if it is determined that the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, changing the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

2. The method according to claim 1, wherein the method further comprises:

in response to a received historical database request comprising the target database statement, querying a plan cache corresponding to the target database statement, and when the plan cache does not comprise a corresponding evolution task and a generated execution plan, generating a database statement execution plan corresponding to the target database statement; and

if the generated database statement execution plan is an execution plan of the evolution type, generating the evolution task and caching the evolution task into the plan cache corresponding to the target database statement.

3. The method according to claim 1, wherein the method further comprises:

in response to the received current database request comprising the target database statement, querying a plan cache corresponding to the target database statement, and when the plan cache does not comprise a corresponding evolution task and a generated execution plan, generating a database statement execution plan corresponding to the target database statement; and

if the execution plan of the baseline type corresponding to the target database statement exists, comparing the generated database statement execution plan with the execution plan of the baseline type, and when a result of the comparison is that the generated database statement execution plan is consistent with the execution plan of the baseline type, executing the database statement
5 execution plan, and caching the database statement execution plan into the plan cache corresponding to the target database statement; or

if the execution plan of the baseline type corresponding to the target database statement does not exist, adding the generated database statement execution plan to the execution plan of the baseline type, executing the database statement execution plan, and caching the database statement
10 execution plan into the plan cache corresponding to the target database statement.

4. The method according to claim 1, wherein the implementation process of the evolution task comprises a plurality of stages, wherein:

in an initial stage, a quantity of execution times of the execution plan of the evolution type is the same as a quantity of execution times of the execution plan of the baseline type; and

15 in any other stage other than the initial stage, a quantity of execution times of each type of execution plan is negatively correlated with a proportion of average execution duration of each type of execution plan in at least one stage before the any other stage.

5. The method according to claim 1, wherein the method further comprises:

after the execution plan selected for the current database request is completed, determining
20 whether the evolution task meets a predefined evolution end condition, wherein the evolution end condition comprises that a quantity of evolution times reaches a threshold and/or evolution duration reaches a threshold; and

if the evolution end condition is met, terminating the evolution task.

6. The method according to claim 1, wherein the method further comprises:

25 separately determining average execution duration of the execution plan of the evolution type and average execution duration of the execution plan of the baseline type based on an execution status of the evolution task; and

determining either the execution plan of the evolution type or the execution plan of the baseline type, whichever has relatively shorter corresponding average execution duration, as the
30 execution plan with relatively better performance.

7. The method according to claim 1, wherein the method further comprises:

when unused duration of any execution plan of the baseline type corresponding to the target database statement reaches a preset duration threshold, deleting the any execution plan of the baseline type.

8. An online evolution apparatus for an execution plan, comprising:

5 an obtaining module, configured to: in response to a received current database request comprising a target database statement, obtain an evolution task from a plan cache corresponding to the target database statement, wherein the evolution task is used to compare performance of an execution plan of an evolution type with performance of an execution plan of a baseline type, the execution plan of the evolution type is an unverified execution plan corresponding to the target
10 database statement, and the execution plan of the baseline type is a verified execution plan corresponding to the target database statement;

 a selection module, configured to select a type of an execution plan actually used by the current database request, so that an accumulative quantity of execution times of each type of execution plan in an implementation process of the evolution task is positively correlated with
15 performance of each type of execution plan; and

 a determining module, configured to: after the evolution task ends, if it is determined that the performance of the execution plan of the evolution type is better than the performance of the execution plan of the baseline type, change the execution plan of the evolution type into an execution plan of the baseline type corresponding to the target database statement.

20 9. An electronic device, comprising:

 a processor; and

 a memory, configured to store a processor-executable instruction, wherein

 the processor runs the executable instruction to implement the online evolution method for an execution plan according to any one of claims 1 to 7.

25 10. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer instruction, and the instruction is executed by a processor to implement the online evolution method for an execution plan according to any one of claims 1 to 7.