

DATA QUERY METHOD AND APPARATUS, AND DATABASE SYSTEM

TECHNICAL FIELD

[0001] The present invention relates to the field of information technologies, and
5 more specifically, to a data query method and apparatus, and a database system.

BACKGROUND

[0002] A structured query language (Structured Query Language, SQL) is a
standard language of a relational database management system. The structured query
language is an advanced descriptive non-procedural language. That is, the structured
10 query language allows a user to obtain required data on a high-layer data structure,
but the structured query language does not require the user to specify a method for
obtaining the data, that is, a query plan in a general sense. The query plan may be
represented by a tree. A leaf node of the tree indicates an input table, and an
intermediate node indicates an operation. An SQL query usually has many equal-cost
15 query plans, and execution times of different query plans vary. Therefore, in a
database system, there is a query optimizer component for selecting an optimal (that is,
an execution time is the shortest) query plan.

[0003] A main function of the query optimizer is to determine an equal-cost query
plan for an original query plan and to select an optimal query plan. However, in the
20 prior art, some equal-cost query plans are omitted, and consequently query
performance is affected.

SUMMARY

[0004] Embodiments of the present invention provide a data query method and
apparatus, and a database system, so as to improve data query performance.
25 [0005] According to a first aspect, an embodiment of the present invention
provides a data query method, where the method is executed by a database system, the

database system runs on a computer device, and the method includes:

receiving a data query request;

generating an original query plan according to the data query request;

restructuring a join predicate in the original query plan;

5 determining basic information of a restructured join predicate in the original query plan;

determining, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan;

10 determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and

performing querying according to the equal-cost query plan for the original query plan.

15 **[0006]** In the embodiments of the present invention, the equal-cost query plan for the original query plan is determined according to the basic information of the predicate and the constraint condition obtained based on the join operation rule, and a relatively large quantity of equal-cost query plans for the original query plan can be obtained, so that data query performance can be improved.

20 **[0007]** In some implementations, the determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan includes:

25 determining an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and

the performing querying according to the equal-cost query plan for the original query plan includes:

30 selecting a query plan from the equal-cost query plan set to perform querying.

[0008] In some implementations, the method further includes:

obtaining a candidate query plan set according to the original query plan, where

candidate query plans for the original query plan should be obtained as much as possible herein; and preferably, a query plan enumeration algorithm may be used to enumerate all candidate query plans for the original query plan, and a typical enumeration algorithm may be used, for example, a dynamic programming algorithm used in IBM System R; and

the determining an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan includes:

determining the equal-cost query plan set from the candidate query plan set according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan.

[0009] In the embodiments of the present invention, the equal-cost query plan for the original query plan is determined from the candidate query plan set according to the basic information of the predicate and the constraint condition obtained based on the join operation rule, and all equal-cost query plans for the original query plan can be obtained, so that data query performance can be further improved.

[0010] In some implementations, the restructuring a join predicate in the original query plan includes:

transforming all join predicates of an inner join operation in the original query plan into a conjunctive normal form, and combining join predicates having a same input table; and

combining all join predicates of a non-inner join operation in the original query plan into one join predicate.

[0011] In some implementations, the basic information of the join predicate includes an input table of the join predicate, an input table of the join predicate that is included in a left subtree of a join operation in which the join predicate is located, an input table of the join predicate that is included in a right subtree of the join operation in which the join predicate is located, and a join type of the join operation in which the join predicate is located.

[0012] In some implementations, the join operation rule includes an operational-level rule and a predicate-level rule, where the operational-level rule includes an operational-level exchange rule: $e_1 \circ_{p12}^a e_2 = e_2 \circ_{p12}^a e_1$, an

operational-level combination rule: $(e_1 o_{p12}^a e_2) o_{p23}^b e_3 = e_1 o_{p12}^a (e_2 o_{p23}^b e_3)$, an operational-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12}^a e_2$, and an operational-level right-permutation rule: $e_1 o_{p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p23}^b (e_1 o_{p13}^a e_3)$, the predicate-level rule includes a predicate-level combination rule: $(e_1 o_{p12}^a e_2) o_{p \wedge p23}^b e_3 = e_1 o_{p12 \wedge p}^a (e_2 o_{p23}^b e_3)$, a predicate-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13 \wedge p}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12 \wedge p}^a e_2$, and a predicate-level right-permutation rule: $e_1 o_{p \wedge p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p \wedge p23}^b (e_1 o_{p13}^a e_3)$, e_i is a query expression, o^a and o^b are join operations, and p_{ij} is a join predicate between e_i and e_j .

[0013] In some implementations, the determining, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan includes:

traversing, for each join operation node o^B in the original query plan, each join operation node o^A in a subtree that uses o^B as a root node; and
 when o^A is in a left subtree of the subtree that uses o^B as the root node, if
 15 o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, adding a first constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level left-permutation rule and the predicate-level left-permutation rule, adding a second constraint condition to each restructured join predicate on o^B ; or
 20 when o^A is in a right subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, adding a second constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level right-permutation rule and the predicate-level right-permutation rule, adding a first constraint condition
 25 to each restructured join predicate on o^B ; where

the first constraint condition is $T(e_2) \rightarrow T(e_1) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, the second constraint condition is $T(e_1) \rightarrow T(e_2) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, $T(e_1)$ indicates an original input table included in a left subtree e_1 of a subtree that uses o^A as a root node, $T(e_2)$ indicates an original input table included in a right subtree e_2 of the subtree that
 30 uses o^A as the root node, $T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$ indicates an original input table included in a restructured join predicate $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$ on o^A , and $T_1 \rightarrow T_2$ indicates that if a first subtree that uses a join operation in which a join predicate constrained by $T_1 \rightarrow T_2$ is located as a root node includes any input table in T_1 , the first subtree includes all input tables in T_2 .

[0014] In some implementations, the determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan includes:

5 determining that a first candidate query plan P' is an equal-cost query plan for the original query plan P if P' satisfies the following conditions:

a join type of each first join operation of P' is an inner join, the first join operation is a join operation having at least two restructured join predicates, and a join type of a join operation that is in P and that is corresponding to each join predicate in the at least two restructured join predicates is an inner join;

10 for each join predicate p on each join operation $o^{b'}$ in P' ,
when $o^{b'}$ does not satisfy the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal, where $T_l(p)$ indicates an input table of p included in a left subtree of a join operation in which p is located, and $T_r(p)$ indicates an input table of p included in a right subtree of the join operation in which p is located; or

when $o^{b'}$ satisfies the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal; or $T_l(p)$ in P' and $T_r(p)$ in P are equal, and $T_r(p)$ in P' and $T_l(p)$ in P are equal; and

20 for each join predicate p on each join operation $o^{b'}$ in P' , p satisfies a constraint condition for p in P .

[0015] In some implementations, the selecting a query plan from the equal-cost query plan set to perform querying includes:

25 selecting a query plan with minimum costs from the equal-cost query plan set to perform querying.

[0016] According to a second aspect, an embodiment of the present invention provides a data query apparatus. The apparatus includes a module for executing the method according to the first aspect.

30 **[0017]** According to a third aspect, an embodiment of the present invention provides a data query apparatus. The apparatus includes a processor, a memory, and a communications interface. The processor is coupled to the memory and the communications interface. The memory is configured to store an instruction, the processor is configured to execute the instruction, and the communications interface is configured to communicate with another network element under control of the

processor. The processor executes the method according to the first aspect when the processor executes the instruction.

[0018] According to a fourth aspect, an embodiment of the present invention provides a database system. The database system includes the data query apparatus according to the second aspect or the third aspect and a database.

[0019] In some implementations, an equal-cost query plan for an original query plan is a query plan obtained from the original query plan by means of transformation by using at least one of four operational-level rules and three predicate-level rules.

[0020] In some implementations, an equal-cost query plan space includes query plans obtained from an original query plan by means of transformation by using at least one of four operational-level rules and three predicate-level rules.

BRIEF DESCRIPTION OF DRAWINGS

[0021] To describe the technical solutions in the embodiments of the present invention more clearly, the following briefly describes the accompanying drawings required for describing the embodiments of the present invention. Apparently, the accompanying drawings in the following description show merely some embodiments of the present invention, and a person of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

[0022] FIG. 1a and FIG. 1b are schematic diagrams of application scenarios according to an embodiment of the present invention;

[0023] FIG. 1c is an example of a join operation according to an embodiment of the present invention;

[0024] FIG. 2 is an example of an equal-cost query plan according to an embodiment of the present invention;

[0025] FIG. 3 is an example of an operational-level exchange rule according to an embodiment of the present invention;

[0026] FIG. 4 is an example of an operational-level combination rule according to an embodiment of the present invention;

[0027] FIG. 5 is an example of an operational-level left-permutation rule according to an embodiment of the present invention;

[0028] FIG. 6 is an example of an operational-level right-permutation rule according to an embodiment of the present invention;

[0029] FIG. 7 is an example of a predicate-level combination rule according to an embodiment of the present invention;

[0030] FIG. 8 is an example of a predicate-level left-permutation rule according to an embodiment of the present invention;

5 **[0031]** FIG. 9 is an example of a predicate-level right-permutation rule according to an embodiment of the present invention;

[0032] FIG. 10 is an example of an equal-cost query plan according to an embodiment of the present invention;

10 **[0033]** FIG. 11 is a schematic flowchart of a data query method according to an embodiment of the present invention;

[0034] FIG. 12 is an example of a query plan according to an embodiment of the present invention;

15 **[0035]** FIG. 13A and FIG. 13B are a flowchart of determining a constraint condition of a predicate in an original query plan according to an embodiment of the present invention;

[0036] FIG. 14 is a schematic diagram of adding a constraint condition according to an embodiment of the present invention;

[0037] FIG. 15 is another schematic diagram of adding a constraint condition according to an embodiment of the present invention;

20 **[0038]** FIG. 16 is an example of determining an equal-cost query plan according to an embodiment of the present invention;

[0039] FIG. 17 is a schematic block diagram of a data query apparatus according to an embodiment of the present invention;

25 **[0040]** FIG. 18 is a schematic block diagram of a database system according to an embodiment of the present invention; and

[0041] FIG. 19 is a schematic structural diagram of a data query apparatus according to an embodiment of the present invention.

DESCRIPTION OF EMBODIMENTS

30 **[0042]** The following clearly and completely describes the technical solutions in the embodiments of the present invention with reference to the accompanying drawings in the embodiments of the present invention. Apparently, the described embodiments are a part rather than all of the embodiments of the present invention.

All other embodiments obtained by a person of ordinary skill in the art based on the embodiments of the present invention without creative efforts shall fall within the protection scope of the present invention.

5 **[0043]** The technical solutions in the embodiments of the present invention may be applied to a database system. FIG. 1a and FIG. 1b are schematic diagrams of application scenarios according to an embodiment of the present invention. FIG. 1a is a database system with a single server, and FIG. 1b is a distributed database system.

10 **[0044]** As shown in FIG. 1a, the database system (that is, a server) with a single server includes a database management system and a database, and is configured to process a query request of a client and return a query result to the client. The database management system includes a management component such as a query optimizer, and is configured to manage the database system. For this scenario, a method in the embodiments of the present invention may be executed by the server, or the database management system in the server, or the query optimizer in the database management system.

15 **[0045]** As shown in FIG. 1b, the distributed database system includes a plurality of servers and a distributed management system, and is configured to process a query request of a client and return a query result to the client. Each server includes a database management system and a database, and the distributed management system includes a management component such as a query optimizer, and is configured to manage the distributed database system. For this scenario, a method in the embodiments of the present invention may be executed by the distributed database system, or the distributed management system in the distributed database system, or the query optimizer in the distributed management system.

20 **[0046]** For ease of understanding the embodiments of the present invention, the following first describes terms related to a database system.

25 **[0047]** In various embodiments of the present invention, a join operation may also be referred to as an operation or a join, and the join operation combines two or more tables in a database into one result table. A join predicate may also be referred to as a predicate, and the join predicate is used to describe a relationship of columns in two or more tables. The join operation combines tables in a database by using the join predicate.

30 **[0048]** The join operation is classified into an inner join (inner join), a left/right outer join (left/right outer join), a full outer join (full outer join), a left/right semi-join

(left/right semi-join), and a left/right anti-join (left/right anti-join).

[0049] The inner join (inner join) is the most common join operation in a database. The inner join combines columns of two tables (for example, R and S) together based on a join predicate, to generate a new result table. The operation compares each row of the table R with each row of the table S, and finds a combination that satisfies the join predicate. When the join predicate is satisfied, matched rows in R and S are combined into one row in a result table by column (side by side).

[0050] The left outer join (left outer join) is an expansion of the inner join. A difference between the left outer join and the inner join is that if the left outer join is performed for the two tables R and S, if there is no matched row in the right table S for a row in a left table (that is, the table R) to satisfy the join predicate, instead of directly discarding a value in the row of the left table in the inner join, the left outer join still returns a record, where the record retains all column values in the left table, but a value in each column corresponding to the row from the right table is null (NULL). The right outer join (right outer join) is similar to the left outer join, and the only difference is that if there is no matched row in the left table for a row in the right table, a value in each column from the left table in a record returned by the operation is null. The full outer join (full outer join) is similar to the left/right outer join, and the only difference is that regardless of the left table or the right table, if there is no matched row for a row in the table, the operation returns a record, and a column value in the table with no matched row in the record is null.

[0051] The left semi-join (left semi-join) returns only those rows, in the left table, that have matched rows in the right table (that is, satisfying the join predicate). The right semi-join (right semi-join) returns only those rows, in the right table, that have matched rows in the left table (that is, satisfying the join predicate).

[0052] The left anti-join (left anti-join) returns only those rows, in the left table, that have no matched row in the right table (that is, not satisfying the join predicate). The right anti-join (right anti-join) returns only those rows, in the right table, that have no matched row in the left table (that is, satisfying the join predicate). It can be learned from definitions of the semi-join and the anti-join that the anti-join and the semi-join are complementary sets for each other.

[0053] FIG. 1c shows examples of the foregoing various join operations. In FIG. 1c, that a join operation is performed for the table R and the table S is used as an example. The join predicate is $R.b = S.c$, and results of the various join operations are

shown in FIG. 1c.

[0054] FIG. 2 shows an example of an equal-cost query plan. As shown in FIG. 2, for an SQL query $R_1 \bowtie_{p12} R_2 \bowtie_{p23} R_3$, R_1 , R_2 , and R_3 represent input tables, $\bowtie$ represents an inner join operation, and $p12$ and $p23$ represent a join predicate between R_1 and R_2 and a join predicate between R_2 and R_3 respectively. Both two query plans in FIG. 2 are equal-cost query plans for the query.

[0055] In various embodiments of the present invention, a join operation rule includes an operational-level rule and a predicate-level rule. The operational-level rule includes an operational-level exchange rule, an operational-level combination rule, an operational-level left-permutation rule, and an operational-level right-permutation rule. The predicate-level rule includes a predicate-level combination rule, a predicate-level left-permutation rule, and a predicate-level right-permutation rule. In the following description, meaning of each symbol is shown in Table 1.

Table 1

Symbol	Explanation
e_i	Query expression
o^a, o^b	Join operation
p_{ij}	Join predicate between e_i and e_j
$A(e_i)$	All attribute sets in e_i
p_{ij} rejects null on $A(e_i)$	If all attribute values in $A(e_i)$ are null, a calculation result of p_{ij} is unknown or false

[0056] The operational-level exchange rule is $e_1 o^a_{p12} e_2 = e_2 o^a_{p12} e_1$. FIG. 3 is an example of the operational-level exchange rule. Table 2 is a join operation table of the operational-level exchange rule, and shows whether each join operation satisfies the operational-level exchange rule. + indicates that the join operation satisfies the operational-level exchange rule, and - indicates that the join operation does not satisfy the operational-level exchange rule. It should be understood that the right outer join, the right semi-join, and the right anti-join may be transformed into corresponding left join operations, and therefore only left join operations are considered in this specification.

Table 2

o^a	Inner join	Left semi-join	Left anti-join	Left outer join	Full outer join
Operational-level exchange rule	+	-	-	-	+

[0057] The operational-level combination rule is $(e_1 o_{p12}^a e_2) o_{p23}^b e_3 = e_1 o_{p12}^a (e_2 o_{p23}^b e_3)$. FIG. 4 is an example of the operational-level combination rule. Table 3 is a join operation table of the operational-level combination rule, and shows whether any two join operations satisfy the operational-level combination rule. + indicates that the two join operations satisfy the operational-level combination rule, and - indicates that the two join operations do not satisfy the operational-level combination rule. + with a superscript indicates that the combination rule is satisfied under a condition. That is, only when a condition in the superscript is satisfied, the two corresponding join operations can satisfy the combination rule.

Table 3

	Inner join	Left semi-join	Left anti-join	Left outer join	Full outer join
Inner join	+	+	+	+	-
Left semi-join	-	-	-	-	-
Left anti-join	-	-	-	-	-
Left outer join	-	-	-	$+^1$	-
Full outer join	-	-	-	$+^1$	$+^2$

$^1 p_{23}$ rejects null on $A(e_2)$ $^2 p_{12}$ and p_{23} rejects null on $A(e_2)$

[0058] The operational-level left-permutation rule is $(e_1 o_{p12}^a e_2) o_{p13}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12}^a e_2$. FIG. 5 is an example of the operational-level left-permutation rule.

Table 4 is a join operation table of the operational-level left-permutation rule, and shows whether any two join operations satisfy the operational-level left-permutation rule. + indicates that the two join operations satisfy the operational-level left-permutation rule, and - indicates that the two join operations do not satisfy the operational-level left-permutation rule. + with a superscript indicates that the left-permutation rule is satisfied under a condition. That is, only when a condition in the superscript is satisfied, the two corresponding join operations can satisfy the left-permutation rule.

Table 4

	Inner join	Left semi-join	Left anti-join	Left outer join	Full outer join
Inner join	+	+	+	+	-
Left semi-join	+	+	+	+	-
Left anti-join	+	+	+	+	-
Left outer join	+	+	+	+	+ ¹
Full outer join	-	-	-	+ ²	+ ³

¹ p₁₂ rejects null on A(e₁) ² p₁₃ rejects null on A(e₃)

³ p₁₂ and p₁₃ rejects null on A(e₁)

[0059] The operational-level right-permutation rule is $e_1 \circ_{p13}^a (e_2 \circ_{p23}^b e_3) = e_2 \circ_{p23}^b (e_1 \circ_{p13}^a e_3)$. FIG. 6 is an example of the operational-level right-permutation rule.

Table 5 is a join operation table of the operational-level right-permutation rule, and shows whether any two join operations satisfy the operational-level right-permutation rule. + indicates that the two join operations satisfy the operational-level right-permutation rule, and - indicates that the two join operations do not satisfy the operational-level right-permutation rule. + with a superscript indicates that the right-permutation rule is satisfied under a condition. That is, only when a condition in the superscript is satisfied, the two corresponding join operations can satisfy the

right-permutation rule.

Table 5

	Inner join	Left semi-join	Left anti-join	Left outer join	Full outer join
Inner join	+	-	-	-	-
Left semi-join	-	-	-	-	-
Left anti-join	-	-	-	-	-
Left outer join	-	-	-	-	-
Full outer join	-	-	-	-	+ ¹

¹ p_{13} and p_{23} rejects null on $A(e_3)$

5 **[0060]** In the foregoing operational-level rule, a join predicate in an operation and the operation are bound together, that is, the join predicate cannot move between operations. Different from the operational-level rule, in a predicate-level rule, a predicate can move between operations.

10 **[0061]** The predicate-level combination rule is $(e_1 \ o_{p12}^a \ e_2) \ o_{p \wedge p23}^b \ e_3 = e_1 \ o_{p12 \wedge p}^a \ (e_2 \ o_{p23}^b \ e_3)$. $\wedge$ indicates an AND logical operation, p indicates a join predicate except $p23$, and p is not null. FIG. 7 is an example of the predicate-level combination rule.

15 **[0062]** The predicate-level left-permutation rule is $(e_1 \ o_{p12}^a \ e_2) \ o_{p13 \wedge p}^b \ e_3 = (e_1 \ o_{p13}^b \ e_3) \ o_{p12 \wedge p}^a \ e_2$. $\wedge$ indicates an AND logical operation, p indicates a join predicate except $p13$, and p is not null. FIG. 8 is an example of the predicate-level left-permutation rule.

[0063] The predicate-level right-permutation rule is $e_1 \ o_{p \wedge p13}^a \ (e_2 \ o_{p23}^b \ e_3) = e_2 \ o_{p \wedge p23}^b \ (e_1 \ o_{p13}^a \ e_3)$. $\wedge$ indicates an AND logical operation, p indicates a join predicate except $p13$, and p is not null. FIG. 9 is an example of the predicate-level right-permutation rule.

20 **[0064]** Table 6 is a join operation table of the predicate-level rule, and shows whether any two join operations satisfy the predicate-level rule. + indicates that the

two join operations satisfy the predicate-level rule, and - indicates that the two join operations do not satisfy the predicate-level rule. Join operation tables of the foregoing three predicate-level rules are the same, that is, Table 6 represents the join operation tables of all the three predicate-level rules.

5

Table 6

	Inner join	Left semi-join	Left anti-join	Left outer join	Full outer join
Inner join	+	-	-	-	-
Left semi-join	-	-	-	-	-
Left anti-join	-	-	-	-	-
Left outer join	-	-	-	-	-
Full outer join	-	-	-	-	-

[0065] In the embodiments of the present invention, all query plans that can be obtained by means of transformation by using the foregoing operational-level rules or predicate-level rules are equal-cost query plans. That is, an equal-cost query plan space in the embodiments of the present invention is all the query plans that can be obtained by means of transformation by using at least one of the four operational-level rules and the three predicate-level rules. Whether a rule is satisfied between operations may be determined by querying a join operation table of the rule.

[0066] Compared with an equal-cost query plan space obtained by means of transformation by using only the operational-level rule, the equal-cost query plan space in the embodiments of the present invention is more complete. That is, if transformation is performed by using only the operational-level rule, not all equal-cost query plans can be obtained. For example, as shown in FIG. 10, relative to a left original query plan, a join predicate between operations is moved in two right candidate query plans, and therefore the two candidate query plans cannot be obtained by transforming the left original query plan by using the foregoing four

operational-level rules. However, the two right candidate query plans are actually equivalent to the left original query plan.

[0067] In the embodiments of the present invention, an equal-cost query plan for an original query plan is determined based on the operational-level rules and the predicate-level rules, and all equal-cost query plans for the original query plan can be obtained.

[0068] FIG. 11 shows a schematic flowchart of a data query method 1100 according to an embodiment of the present invention. The method 1100 is executed by a database system, and the database system runs on a computer device. Specifically, the method is executed by a data query apparatus running on the computer. The data query apparatus may be an apparatus in the database system, such as a query optimizer, or may be the database system.

[0069] S1110. Receive a data query request.

[0070] For example, the data query request may be a data query request input by a user by using an SQL language.

[0071] S1120. Generate an original query plan according to the data query request.

[0072] After the data query request is received, the data query request is parsed to obtain the original query plan.

[0073] Optionally, S1130. Obtain a candidate query plan set according to the original query plan.

[0074] The candidate query plan set includes all candidate query plans for the original query plan. In this step, a query plan enumeration algorithm may be used to enumerate all the candidate query plans for the original query plan. For example, a typical enumeration algorithm includes a dynamic programming algorithm used in IBM System R.

[0075] After all the candidate query plans for the original query plan are enumerated, an equal-cost query plan for the original query plan may be determined from all the candidate query plans.

[0076] This step is an optional step. That is, in this embodiment of the present invention, the equal-cost query plan or an equal-cost query plan set may be directly obtained from the original query plan by using the following step, or an equal-cost query plan set may be obtained from the candidate query plan set by using the following step.

[0077] S1140. Restructure a join predicate in the original query plan.

[0078] Specifically, all join predicates of an inner join operation in the original query plan are transformed into a conjunctive normal form, and join predicates having a same input table are combined; and all join predicates of a non-inner join operation in the original query plan are combined into one join predicate.

[0079] That is, a restructuring manner is traversing each join operation node in the original query plan, and processing is performed according to a join type of the join operation node in the following two cases:

[0080] If the join operation is an inner join, all predicates in the join operation are transformed into the conjunctive normal form (conjunctive normal form, CNF), that is, a $p_1 \wedge p_2 \wedge \dots \wedge p_n$ form, and predicates having a same input table are combined.

[0081] If the join operation is not an inner join, all predicates in the join operation are combined into one predicate.

[0082] S1150. Determine basic information of a restructured join predicate in the original query plan.

[0083] In various embodiments of the present invention, the basic information of the join predicate includes an input table of the join predicate, an input table of the join predicate that is included in a left subtree of a join operation in which the join predicate is located, an input table of the join predicate that is included in a right subtree of the join operation in which the join predicate is located, and a join type of the join operation in which the join predicate is located.

[0084] Specifically, for a predicate p , basic information of the predicate p includes:

$T(p)$: an input table included in p ;

$T_l(p)$: an input table of p that is included in a left subtree of an operation in which p is located;

$T_r(p)$: an input table of p that is included in a right subtree of the operation in which p is located; and

$O(p)$: a join type of the operation in which p is located.

[0085] For example, for a join predicate $p = \{R1.a + R2.b = R3.c\}$ of a root node shown in FIG. 12, basic information of p is as follows:

$T(p)$: $\{R1, R2, R3\}$;

$T_l(p)$: $\{R1, R2\}$;

$T_r(p)$: $\{R3\}$; and

$O(p)$: an inner join.

[0086] S1160. Determine, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan.

[0087] Specifically, for each join operation node o^B in the original query plan,

5 each join operation node o^A in a subtree that uses o^B as a root node is traversed; and

when o^A is in a left subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, a first constraint condition is added to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level left-permutation rule and the predicate-level left-permutation rule, a second constraint condition is

10 added to each restructured join predicate on o^B ;

when o^A is in a right subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, a second constraint condition is added to each restructured join

15 predicate on o^B ; and if o^A and o^B do not satisfy the operational-level right-permutation rule and the predicate-level right-permutation rule, a first constraint condition is added to each restructured join predicate on o^B ; where

the first constraint condition is $T(e_2) \rightarrow T(e_1) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, the second constraint condition is $T(e_1) \rightarrow T(e_2) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, $T(e_1)$ indicates an

20 original input table included in a left subtree e_1 of a subtree that uses o^A as a root node, $T(e_2)$ indicates an original input table included in a right subtree e_2 of the subtree that uses o^A as the root node, $T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$ indicates an original input table included in a restructured join predicate $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$ on o^A , $T_1 \rightarrow T_2$ indicates that if a first subtree that uses a join operation in which a join predicate constrained by

25 $T_1 \rightarrow T_2$ is located as a root node includes any input table in T_1 , the first subtree includes all input tables in T_2 , and $\cap$ is an intersection set symbol.

[0088] For example, FIG. 13A and FIG. 13B show a flowchart of determining a constraint condition of a predicate in an original query plan. A procedure shown in FIG. 13A and FIG. 13B include two-layer traversal. For an outer layer, each operation

30 node o^B (a restructured predicate on o^B is $p_{b1} \wedge p_{b2} \wedge \dots \wedge p_{bn}$) in the original query plan is traversed; for an inner layer, each operation node o^A (a restructured predicate on o^A is $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$) in a query subtree that uses o^B as a root node is traversed. Then, a constraint condition is added to each predicate on o^B according to a

relationship between o^A and o^B in the following two cases:

[0089] If o^A is in a left subtree of o^B , a constraint condition is added to each predicate p_{bi} on o^B according to a method shown in FIG. 14.

[0090] If o^A is in a right subtree of o^B , a constraint condition is added to each predicate p_{bi} on o^B according to a method shown in FIG. 15.

[0091] S1170. Determine an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan.

[0092] Optionally, an equal-cost query plan set for the original query plan may be determined according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan. A query plan in the equal-cost query plan set is the equal-cost query plan for the original query plan.

[0093] In this step, optionally, if the candidate query plan set for the original query plan is obtained in an enumeration manner, the equal-cost query plan set may be determined from the candidate query plan set, that is, whether each candidate query plan is an equal-cost query plan for the original query plan is determined. If there is no candidate query plan set, a candidate query plan may be obtained from the original query plan each time, and whether the candidate query plan is an equal-cost query plan for the original query plan is determined.

[0094] Specifically, it is determined that a first candidate query plan P' is an equal-cost query plan for the original query plan P if P' satisfies the following conditions:

a join type of each first join operation of P' is an inner join, the first join operation is a join operation having at least two restructured join predicates, and a join type of a join operation that is in P and that is corresponding to each join predicate in the at least two restructured join predicates is an inner join (this condition may be referred to as join type consistency);

for each join predicate p on each join operation $o^{b'}$ in P' ,
when $o^{b'}$ does not satisfy the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal, where $T_l(p)$ indicates an input table of p included in a left subtree of a join operation in which p is located, and $T_r(p)$ indicates an input table of p included in a right subtree of the join operation

in which p is located;

when o^{b_i} satisfies the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal; or $T_l(p)$ in P' and $T_r(p)$ in P are equal, and $T_r(p)$ in P' and $T_l(p)$ in P are equal (this condition may be referred to as

5 input table information consistency); and

for each join predicate p on each join operation o^{b_i} in P' , p satisfies a constraint condition for p in P (this condition may be referred to as constraint condition consistency), and if the candidate query plan set for the original query plan is obtained previously, the first candidate query plan P' herein may be any candidate
10 query plan in the candidate query plan set.

[0095] All equal-cost query plans for the original query plan can be obtained in the foregoing manner. That is, the equal-cost query plan set obtained in this embodiment of the present invention is the most complete.

[0096] S1180. Perform querying according to the equal-cost query plan for the
15 original query plan.

[0097] Optionally, a query plan may be selected from the equal-cost query plan set to perform querying. That is, the query plan is selected from the foregoing obtained equal-cost query plan set to perform querying. Specifically, a query plan with minimum costs may be selected from the equal-cost query plan set to perform
20 querying, and a query result is returned.

[0098] Because the equal-cost query plan set in this embodiment of the present invention includes all the equal-cost query plans for the original query plan, an optimal query plan can be selected in this embodiment of the present invention. Considering that different query plans differ in performance by dozens or even
25 hundreds of times, data query performance can be significantly improved by using the method in this embodiment of the present invention.

[0099] Therefore, according to the data query method in this embodiment of the present invention, the equal-cost query plan for the original query plan is determined according to the basic information of the predicate and the constraint condition
30 obtained based on the join operation rule, and all the equal-cost query plans for the original query plan can be obtained, so that data query performance can be improved.

[0100] The equal-cost query plan set in this embodiment of the present invention includes all query plans that can be obtained from the original query plan by means of transformation by using at least one of four operational-level rules and three

predicate-level rules. The method for determining the equal-cost query plan for the original query plan is given in the foregoing steps S1140 to S1170. The following describes the method for determining the equal-cost query plan for the original query plan in detail with reference to a specific example. It should be noted that this is merely intended to help a person skilled in the art better understand this embodiment of the present invention, but is not intended to limit the scope of this embodiment of the present invention.

[0101] FIG. 16 shows three query plans P, P', and P". $\Join$ represents an inner join operation, $\Joinleft$ represents a left anti-join, P is an original query plan, and P' and P" are candidate query plans. Whether the candidate query plans P' and P" are equivalent to the original query plan P may be determined in the following manner.

[0102] 1. Join predicates in the original query plan P are restructured, and basic information of restructured join predicates are determined, where P includes three join operations (that is, $\Join_{p14 \wedge p34}$, $\Joinleft_{p23}$, and $\Join_{p12}$) in total. For the join operation $\Join_{p14 \wedge p34}$, because the join operation is an inner join, predicates of the join operation need to be transformed into a conjunctive normal form (a conjunctive normal form of $p14 \wedge p34$ is $p14 \wedge p34$). Because input tables of p14 and p34 are different, p14 and p34 cannot be combined into one predicate. For a join operation $\Joinleft_{p23}$, because the join operation is not an inner join, a predicate (that is, p23) of the join operation is considered as one predicate. Similarly, a predicate of a join operation $\Join_{p12}$ is still p12 after being restructured. After the predicates are restructured, the basic information of the restructured predicates is determined. Table 7 lists the basic information of the restructured predicates in P.

Table 7

p	T(p)	T_l(p)	T_r(p)	O(p)
p14	{R ₁ , R ₄ }	{R ₁ }	{R ₄ }	Inner join
p34	{R ₃ , R ₄ }	{R ₃ }	{R ₄ }	Inner join
p23	{R ₂ , R ₃ }	{R ₂ }	{R ₃ }	Left anti-join
p12	{R ₁ , R ₂ }	{R ₁ }	{R ₂ }	Inner join

[0103] 2. Constraint conditions of the restructured join predicates are determined:

The operation $\Join_{p14 \wedge p34}$ has two descendant operations (that is, $\Join_{p12}$ and $\Join_{p23}$), and therefore whether the two descendant operations and the operation satisfy a corresponding rule needs to be determined to obtain constraint conditions of the predicates p14 and p34. The descendant operation $\Join_{p12}$ and the operation $\Join_{p14 \wedge p34}$ satisfy an operational-level combination rule and a predicate-level combination rule, and further satisfy an operational-level left-permutation rule and a predicate-level left-permutation rule, and therefore no constraint rule needs to be added. The descendant operation $\Join_{p23}$ and the operation $\Join_{p14 \wedge p34}$ do not satisfy the operational-level combination rule and the predicate-level combination rule, and therefore a rule $R_3 \rightarrow R_2$ needs to be added to p14 and p34. In addition, $\Join_{p23}$ and the operation $\Join_{p14 \wedge p34}$ do not satisfy the predicate-level left-permutation rule, but satisfy the operational-level left-permutation rule, and therefore no constraint rule needs to be further added. Similarly, constraint conditions of predicates of the operations $\Join_{p12}$ and $\Join_{p23}$ and descendant operations of the operations $\Join_{p12}$ and $\Join_{p23}$ may be obtained by determining whether the operations $\Join_{p12}$ and $\Join_{p23}$ and the descendant operations of the operations $\Join_{p12}$ and $\Join_{p23}$ satisfy a corresponding rule. Table 8 lists the constraint conditions of the restructured predicates in the original query plan P.

Table 8

p	Constraint condition
p14	$R_3 \rightarrow R_2$
p34	$R_3 \rightarrow R_2$
p23	None
p12	None

[0104] 3. An equal-cost query plan is determined according to the basic information and the constraint conditions of the predicates: The candidate query plan P' is not equivalent to the original query plan P because P' violates join type consistency (an operation for a root node of P' includes two predicates, but an operation type of the node is not an inner join, and a join type of an operation that is in P and that is corresponding to p23 is not an inner join either). The candidate query plan P'' satisfies join type consistency (an operation $\Join_{p12 \wedge p34}$ in P'' is an inner join, and

operations in the original query P that are corresponding to p12 and p34 are also inner joins), input table information consistency (input table information of a predicate in P" and input table information of the predicate in the table are consistent), and constraint condition consistency (the predicate in P" satisfies the constraint condition shown in the table), and therefore P" is equivalent to the original query plan P.

[0105] According to the method for determining the equal-cost query plan in this embodiment of the present invention, all the equal-cost query plans for the original query plan can be obtained, and data query performance can be improved.

[0106] It should be understood that sequence numbers of the foregoing processes do not mean execution sequences in various embodiments of the present invention. The execution sequences of the processes should be determined according to functions and internal logic of the processes, and should not be construed as any limitation on the implementation processes of the embodiments of the present invention.

[0107] The foregoing has described in detail the data query method according to an embodiment of the present invention. The following describes a data query apparatus and a database system according to an embodiment of the present invention. The data query apparatus and the database system can execute the data query method in the foregoing embodiment of the present invention.

[0108] FIG. 17 shows a schematic block diagram of a data query apparatus 1700 according to an embodiment of the present invention. The apparatus 1700 may be an apparatus in a database system, for example, a query optimizer. As shown in FIG. 17, the apparatus 1700 includes:

a receiving module 1710, configured to receive a data query request;

a generation module 1720, configured to generate an original query plan according to the data query request;

a restructuring module 1740, configured to restructure a join predicate in the original query plan;

a determining module 1750, configured to: determine basic information of a restructured join predicate in the original query plan; determine, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan; and determine an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the

original query plan; and

a query module 1780, configured to perform querying according to the equal-cost query plan for the original query plan.

[0109] According to the data query apparatus in this embodiment of the present invention, the equal-cost query plan for the original query plan is determined according to the basic information of the predicate and the constraint condition obtained based on the join operation rule, and all equal-cost query plans for the original query plan can be obtained, so that data query performance can be improved.

[0110] Optionally, in an embodiment of the present invention, the determining module 1750 is specifically configured to determine an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and

the query module 1780 is specifically configured to select a query plan from the equal-cost query plan set to perform querying.

[0111] Optionally, in an embodiment of the present invention, the apparatus 1700 may further include:

an obtaining module 1730, configured to obtain a candidate query plan set according to the original query plan; where

in this case, the determining module 1750 is specifically configured to determine the equal-cost query plan set from the candidate query plan set according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan.

[0112] Optionally, in an embodiment of the present invention, the restructuring module 1740 is specifically configured to:

transform all join predicates of an inner join operation in the original query plan into a conjunctive normal form, and combine join predicates having a same input table; and

combine all join predicates of a non-inner join operation in the original query plan into one join predicate.

[0113] Optionally, in an embodiment of the present invention, the basic information of the join predicate includes an input table of the join predicate, an input table of the join predicate that is included in a left subtree of a join operation in which the join predicate is located, an input table of the join predicate that is included in a

right subtree of the join operation in which the join predicate is located, and a join type of the join operation in which the join predicate is located.

[0114] Optionally, in an embodiment of the present invention, the join operation rule includes an operational-level rule and a predicate-level rule, where the operational-level rule includes an operational-level exchange rule: $e_1 o_{p12}^a e_2 = e_2 o_{p12}^a e_1$, an operational-level combination rule: $(e_1 o_{p12}^a e_2) o_{p23}^b e_3 = e_1 o_{p12}^a (e_2 o_{p23}^b e_3)$, an operational-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12}^a e_2$, and an operational-level right-permutation rule: $e_1 o_{p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p23}^b (e_1 o_{p13}^a e_3)$, the predicate-level rule includes a predicate-level combination rule: $(e_1 o_{p12}^a e_2) o_{p \wedge p23}^b e_3 = e_1 o_{p12 \wedge p}^a (e_2 o_{p23}^b e_3)$, a predicate-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13 \wedge p}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12 \wedge p}^a e_2$, and a predicate-level right-permutation rule: $e_1 o_{p \wedge p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p \wedge p23}^b (e_1 o_{p13}^a e_3)$, e_i is a query expression, o^a and o^b are join operations, and p_{ij} is a join predicate between e_i and e_j .

[0115] Optionally, in an embodiment of the present invention, the determining module 1750 is specifically configured to:

traverse, for each join operation node o^B in the original query plan, each join operation node o^A in a subtree that uses o^B as a root node; and

when o^A is in a left subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, add a first constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level left-permutation rule and the predicate-level left-permutation rule, add a second constraint condition to each restructured join predicate on o^B ;

when o^A is in a right subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, add a second constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level right-permutation rule and the predicate-level right-permutation rule, add a first constraint condition to each restructured join predicate on o^B ; where

the first constraint condition is $T(e_2) \rightarrow T(e_1) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, the second constraint condition is $T(e_1) \rightarrow T(e_2) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, $T(e_1)$ indicates an original input table included in a left subtree e_1 of a subtree that uses o^A as a root node, $T(e_2)$ indicates an original input table included in a right subtree e_2 of the subtree that

uses o^A as the root node, $T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$ indicates an original input table included in a restructured join predicate $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$ on o^A , and $T_1 \rightarrow T_2$ indicates that if a first subtree that uses a join operation in which a join predicate constrained by $T_1 \rightarrow T_2$ is located as a root node includes any input table in T_1 , the first subtree includes all input tables in T_2 .

[0116] Optionally, in an embodiment of the present invention, the determining module 1750 is specifically configured to:

determine that a first candidate query plan P' is an equal-cost query plan for the original query plan P if P' satisfies the following conditions:

a join type of each first join operation of P' is an inner join, the first join operation is a join operation having at least two restructured join predicates, and a join type of a join operation that is in P and that is corresponding to each join predicate in the at least two restructured join predicates is an inner join;

for each join predicate p on each join operation $o^{b'}$ in P' ,

when $o^{b'}$ does not satisfy the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal, where $T_l(p)$ indicates an input table of p included in a left subtree of a join operation in which p is located, and $T_r(p)$ indicates an input table of p included in a right subtree of the join operation in which p is located;

when $o^{b'}$ satisfies the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal; or $T_l(p)$ in P' and $T_r(p)$ in P are equal, and $T_r(p)$ in P' and $T_l(p)$ in P are equal; and

for each join predicate p on each join operation $o^{b'}$ in P' , p satisfies a constraint condition for p in P .

[0117] Optionally, in an embodiment of the present invention, the query module 1780 is specifically configured to:

select a query plan with minimum costs from the equal-cost query plan set to perform querying.

[0118] The data query apparatus 1700 according to this embodiment of the present invention may be corresponding to an execution body of the data query method according to the embodiments of the present invention, and the foregoing and other operations and/or functions of the modules in the apparatus 1700 are respectively used to implement corresponding procedures of the foregoing methods. For brevity, details are not described herein again.

[0119] FIG. 18 shows a schematic block diagram of a database system 1800 according to an embodiment of the present invention. As shown in FIG. 18, the database system 1800 includes the foregoing data query apparatus 1700 according to the embodiments of the present invention and a database 1810. The database system
5 1800 may execute the data query method according to the embodiments of the present invention, and performs querying in the database 1810.

[0120] FIG. 19 shows a structure of a data query apparatus according to another embodiment of the present invention. The apparatus includes at least one processor 1902 (for example, a CPU), at least one network interface 1905 or another
10 communications interface, a memory 1906, and at least one communications bus 1903 that is configured to implement connection and communication between these apparatuses. The processor 1902 is configured to execute an executable module stored in the memory 1906, such as a computer program. The memory 1906 may include a high-speed random access memory (RAM, Random Access Memory), and may
15 further include a non-volatile memory (non-volatile memory) such as at least one magnetic disk memory. A communication connection with at least one other network element is implemented by using the at least one network interface 1905 (which may be wired or wireless).

[0121] In some implementations, the memory 1906 stores a program 19061, and
20 the processor 1902 executes the program 19061 to execute the data query method according to the foregoing embodiment of the present invention.

[0122] It should be understood that, the term "and/or" in this embodiment of the present invention describes only an association relationship for describing associated objects and represents that three relationships may exist. For example, A and/or B
25 may represent the following three cases: Only A exists, both A and B exist, and only B exists. In addition, the character "/" in this specification generally indicates an "or" relationship between the associated objects.

[0123] A person of ordinary skill in the art may be aware that, in combination with the examples described in the embodiments disclosed in this specification, units
30 and algorithm steps may be implemented by electronic hardware, computer software, or a combination thereof. To clearly describe the interchangeability between the hardware and the software, the foregoing has generally described compositions and steps of each example according to functions. Whether the functions are performed by hardware or software depends on particular applications and design constraint

conditions of the technical solutions. A person skilled in the art may use different methods to implement the described functions for each particular application, but it should not be considered that the implementation goes beyond the scope of the present invention.

5 **[0124]** It may be clearly understood by a person skilled in the art that, for the purpose of convenient and brief description, for a detailed working process of the foregoing system, apparatus, and unit, reference may be made to a corresponding process in the foregoing method embodiments, and details are not described herein again.

10 **[0125]** In the several embodiments provided in this application, it should be understood that the disclosed system, apparatus, and method may be implemented in other manners. For example, the described apparatus embodiment is merely an example. For example, the unit division is merely logical function division and may be other division in actual implementation. For example, a plurality of units or
15 components may be combined or integrated into another system, or some features may be ignored or not performed. In addition, the displayed or discussed mutual couplings or direct couplings or communication connections may be implemented through some interfaces, indirect couplings or communication connections between the apparatuses or units, or electrical connections, mechanical connections, or
20 connections in other forms.

[0126] The units described as separate parts may or may not be physically separate, and parts displayed as units may or may not be physical units, may be located in one position, or may be distributed on a plurality of network units. A part or all of the units may be selected according to actual needs to achieve the objectives of
25 the solutions of the embodiments of the present invention.

[0127] In addition, functional units in the embodiments of the present invention may be integrated into one processing unit, or each of the units may exist alone physically, or two or more units are integrated into one unit. The integrated unit may be implemented in a form of hardware, or may be implemented in a form of a
30 software functional unit.

[0128] When the integrated unit is implemented in the form of a software functional unit and sold or used as an independent product, the integrated unit may be stored in a computer-readable storage medium. Based on such an understanding, the technical solutions of the present invention essentially, or the part contributing to the

prior art, or all or a part of the technical solutions may be implemented in the form of a software product. The software product is stored in a storage medium and includes several instructions for instructing a computer device (which may be a personal computer, a server, or a network device) to perform all or a part of the steps of the methods described in the embodiments of the present invention. The foregoing storage medium includes: any medium that can store program code, such as a USB flash drive, a removable hard disk, a read-only memory (ROM, Read-Only Memory), a random access memory (RAM, Random Access Memory), a magnetic disk, or an optical disc.

[0129] The foregoing descriptions are merely specific embodiments of the present invention, but are not intended to limit the protection scope of the present invention. Any modification or replacement readily figured out by a person skilled in the art within the technical scope disclosed in the present invention shall fall within the protection scope of the present invention. Therefore, the protection scope of the present invention shall be subject to the protection scope of the claims.

CLAIMS

What is claimed is:

1. A data query method, wherein the method is executed by a database system, the database system runs on a computer device, and the method comprises:

- 5 receiving a data query request;
 generating an original query plan according to the data query request;
 restructuring a join predicate in the original query plan;
 determining basic information of a restructured join predicate in the original query plan;
10 determining, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan;
 determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and
15 performing querying according to the equal-cost query plan for the original query plan.

2. The method according to claim 1, wherein the determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan comprises:

- 20 determining an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and
 the performing querying according to the equal-cost query plan for the original query plan comprises:
 selecting a query plan from the equal-cost query plan set to perform querying.

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

- obtaining a candidate query plan set according to the original query plan; and
30 the determining an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the

original query plan comprises:

determining the equal-cost query plan set from the candidate query plan set according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan.

4. The method according to any one of claims 1 to 3, wherein the restructuring a join predicate in the original query plan comprises:

transforming all join predicates of an inner join operation in the original query plan into a conjunctive normal form, and combining join predicates having a same input table; and

combining all join predicates of a non-inner join operation in the original query plan into one join predicate.

5. The method according to any one of claims 1 to 4, wherein the basic information of the join predicate comprises an input table of the join predicate, an input table of the join predicate that is comprised in a left subtree of a join operation in which the join predicate is located, an input table of the join predicate that is comprised in a right subtree of the join operation in which the join predicate is located, and a join type of the join operation in which the join predicate is located.

6. The method according to any one of claims 1 to 5, wherein the join operation rule comprises an operational-level rule and a predicate-level rule, the operational-level rule comprises an operational-level exchange rule: $e_1 o_{p12}^a e_2 = e_2 o_{p12}^a e_1$, an operational-level combination rule: $(e_1 o_{p12}^a e_2) o_{p23}^b e_3 = e_1 o_{p12}^a (e_2 o_{p23}^b e_3)$, an operational-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12}^a e_2$, and an operational-level right-permutation rule: $e_1 o_{p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p23}^b (e_1 o_{p13}^a e_3)$, the predicate-level rule comprises a predicate-level combination rule: $(e_1 o_{p12}^a e_2) o_{p \wedge p23}^b e_3 = e_1 o_{p12 \wedge p}^a (e_2 o_{p23}^b e_3)$, a predicate-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13 \wedge p}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12 \wedge p}^a e_2$, and a predicate-level right-permutation rule: $e_1 o_{p \wedge p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p \wedge p23}^b (e_1 o_{p13}^a e_3)$, e_i is a query expression, o^a and o^b are join operations, and p_{ij} is a join predicate between e_i and e_j .

7. The method according to claim 6, wherein the determining, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan comprises:

traversing, for each join operation node o^B in the original query plan, each join operation node o^A in a subtree that uses o^B as a root node; and

when o^A is in a left subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, adding a first constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level left-permutation rule and the predicate-level left-permutation rule, adding a second constraint condition to each restructured join predicate on o^B ; or

when o^A is in a right subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, adding a second constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level right-permutation rule and the predicate-level right-permutation rule, adding a first constraint condition to each restructured join predicate on o^B ; wherein

the first constraint condition is $T(e_2) \rightarrow T(e_1) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, the second constraint condition is $T(e_1) \rightarrow T(e_2) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, $T(e_1)$ indicates an original input table comprised in a left subtree e_1 of a subtree that uses o^A as a root node, $T(e_2)$ indicates an original input table comprised in a right subtree e_2 of the subtree that uses o^A as the root node, $T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$ indicates an original input table comprised in a restructured join predicate $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$ on o^A , and $T_1 \rightarrow T_2$ indicates that if a first subtree that uses a join operation in which a join predicate constrained by $T_1 \rightarrow T_2$ is located as a root node comprises any input table in T_1 , the first subtree comprises all input tables in T_2 .

8. The method according to any one of claims 1 to 7, wherein the determining an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan comprises:

determining that a first candidate query plan P' is an equal-cost query plan for the original query plan P if P' satisfies the following conditions:

a join type of each first join operation of P' is an inner join, the first join operation is a join operation having at least two restructured join predicates, and a join type of a join operation that is in P and that is corresponding to each join predicate in the at least two restructured join predicates is an inner join;

for each join predicate p on each join operation o^{b_i} in P' ,

when o^{b_i} does not satisfy the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal, wherein $T_l(p)$ indicates an input

table of p comprised in a left subtree of a join operation in which p is located, and $T_r(p)$ indicates an input table of p comprised in a right subtree of the join operation in which p is located; or

when $o^{b'}$ satisfies the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal; or $T_l(p)$ in P' and $T_r(p)$ in P are equal, and $T_r(p)$ in P' and $T_l(p)$ in P are equal; and

for each join predicate p on each join operation $o^{b'}$ in P' , p satisfies a constraint condition for p in P .

9. The method according to claim 2, wherein the selecting a query plan from the equal-cost query plan set to perform querying comprises:

selecting a query plan with minimum costs from the equal-cost query plan set to perform querying.

10. A data query apparatus, comprising:

a receiving module, configured to receive a data query request;

a generation module, configured to generate an original query plan according to the data query request;

a restructuring module, configured to restructure a join predicate in the original query plan;

a determining module, configured to: determine basic information of a restructured join predicate in the original query plan; determine, based on a join operation rule, a constraint condition of the restructured join predicate in the original query plan; and determine an equal-cost query plan for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and

a query module, configured to perform querying according to the equal-cost query plan for the original query plan.

11. The apparatus according to claim 10, wherein the determining module is specifically configured to determine an equal-cost query plan set for the original query plan according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan; and

the query module is specifically configured to select a query plan from the equal-cost query plan set to perform querying.

12. The apparatus according to claim 11, wherein the apparatus further comprises:

an obtaining module, configured to obtain a candidate query plan set according to the original query plan; wherein

5 the determining module is specifically configured to determine the equal-cost query plan set from the candidate query plan set according to the basic information of the restructured join predicate in the original query plan and the constraint condition of the restructured join predicate in the original query plan.

10 13. The apparatus according to any one of claims 10 to 12, wherein the restructuring module is specifically configured to:

transform all join predicates of an inner join operation in the original query plan into a conjunctive normal form, and combine join predicates having a same input table; and

15 combine all join predicates of a non-inner join operation in the original query plan into one join predicate.

14. The apparatus according to any one of claims 10 to 13, wherein the basic information of the join predicate comprises an input table of the join predicate, an input table of the join predicate that is comprised in a left subtree of a join operation in which the join predicate is located, an input table of the join predicate that is comprised in a right subtree of the join operation in which the join predicate is located, and a join type of the join operation in which the join predicate is located.

15. The apparatus according to any one of claims 10 to 14, wherein the join operation rule comprises an operational-level rule and a predicate-level rule, wherein the operational-level rule comprises an operational-level exchange rule: $e_1 o_{p12}^a e_2 = e_2 o_{p12}^a e_1$, an operational-level combination rule: $(e_1 o_{p12}^a e_2) o_{p23}^b e_3 = e_1 o_{p12}^a (e_2 o_{p23}^b e_3)$, an operational-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12}^a e_2$, and an operational-level right-permutation rule: $e_1 o_{p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p23}^b (e_1 o_{p13}^a e_3)$, the predicate-level rule comprises a predicate-level combination rule: $(e_1 o_{p12}^a e_2) o_{p \wedge p23}^b e_3 = e_1 o_{p12 \wedge p}^a (e_2 o_{p23}^b e_3)$, a predicate-level left-permutation rule: $(e_1 o_{p12}^a e_2) o_{p13 \wedge p}^b e_3 = (e_1 o_{p13}^b e_3) o_{p12 \wedge p}^a e_2$, and a predicate-level right-permutation rule: $e_1 o_{p \wedge p13}^a (e_2 o_{p23}^b e_3) = e_2 o_{p \wedge p23}^b (e_1 o_{p13}^a e_3)$, e_i is a query expression, o^a and o^b are join operations, and p_{ij} is a join predicate between e_i and e_j .

16. The apparatus according to claim 15, wherein the determining module is specifically configured to:

traverse, for each join operation node o^B in the original query plan, each join operation node o^A in a subtree that uses o^B as a root node; and

when o^A is in a left subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, add a first constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level left-permutation rule and the predicate-level left-permutation rule, add a second constraint condition to each restructured join predicate on o^B ; or

when o^A is in a right subtree of the subtree that uses o^B as the root node, if o^A and o^B do not satisfy the operational-level combination rule and the predicate-level combination rule, add a second constraint condition to each restructured join predicate on o^B ; and if o^A and o^B do not satisfy the operational-level right-permutation rule and the predicate-level right-permutation rule, add a first constraint condition to each restructured join predicate on o^B ; wherein

the first constraint condition is $T(e_2) \rightarrow T(e_1) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, the second constraint condition is $T(e_1) \rightarrow T(e_2) \cap T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$, $T(e_1)$ indicates an original input table comprised in a left subtree e_1 of a subtree that uses o^A as a root node, $T(e_2)$ indicates an original input table comprised in a right subtree e_2 of the subtree that uses o^A as the root node, $T(p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an})$ indicates an original input table comprised in a restructured join predicate $p_{a1} \wedge p_{a2} \wedge \dots \wedge p_{an}$ on o^A , and $T_1 \rightarrow T_2$ indicates that if a first subtree that uses a join operation in which a join predicate constrained by $T_1 \rightarrow T_2$ is located as a root node comprises any input table in T_1 , the first subtree comprises all input tables in T_2 .

17. The apparatus according to any one of claims 10 to 16, wherein the determining module is specifically configured to:

determine that a first candidate query plan P' is an equal-cost query plan for the original query plan P if P' satisfies the following conditions:

a join type of each first join operation of P' is an inner join, the first join operation is a join operation having at least two restructured join predicates, and a join type of a join operation that is in P and that is corresponding to each join predicate in the at least two restructured join predicates is an inner join;

for each join predicate p on each join operation o^{b_i} in P' ,

when o^{b_i} does not satisfy the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal, wherein $T_l(p)$ indicates an input

table of p comprised in a left subtree of a join operation in which p is located, and $T_r(p)$ indicates an input table of p comprised in a right subtree of the join operation in which p is located; or

- 5 when o^{b_i} satisfies the operational-level exchange rule, $T_l(p)$ in P' and $T_l(p)$ in P are equal, and $T_r(p)$ in P' and $T_r(p)$ in P are equal; or $T_l(p)$ in P' and $T_r(p)$ in P are equal, and $T_r(p)$ in P' and $T_l(p)$ in P are equal; and

for each join predicate p on each join operation o^{b_i} in P' , p satisfies a constraint condition for p in P .

- 10 18. The apparatus according to claim 11, wherein the query module is specifically configured to:

select a query plan with minimum costs from the equal-cost query plan set to perform querying.

- 15 19. A data query apparatus, wherein the apparatus comprises a processor and a memory, wherein the memory is configured to store an instruction, the processor is configured to execute the instruction, and the processor executes the method according to any one of claims 1 to 9 when the processor performs the instruction.

20. A database system, comprising the data query apparatus according to any one of claims 10 to 18 and a database.