CLAIMS

1. A transaction processing method for a database system, performed by an electronic device, the method comprising:

determining a concurrent transaction of a target transaction, a read and write operation acting on a same variable being comprised between the concurrent transaction and the target transaction, and the target transaction being a transaction to be committed;

obtaining a first intersection between a read set of the target transaction and a write set of the concurrent transaction;

obtaining a second intersection between a write set of the target transaction and a read set of the concurrent transaction;

in response to at least one of the first intersection and the second intersection being a non-empty data set and the target transaction conflicting with the concurrent transaction:

determining, based on a version change count of the same variable on which the target transaction and the concurrent transaction act and a target variable list, a data anomaly type, the target variable list including the same variable on which the target transaction and the concurrent transaction act;

determining a transaction to be rolled back from the target transaction and the concurrent transaction;

rolling back the transaction; and

after rolling back the transaction, deleting a variable on which the transaction to be rolled back acts and a corresponding read and write operation from a target conflict cycle graph, wherein vertices in the target conflict cycle graph are used for representing transaction identifiers corresponding to transactions and variables on which a read and write operation corresponding to the transactions acts, and edges in the target conflict cycle graph are used for representing an association between the transactions and version changes between the variables.

2. The method according to claim 1, wherein determining the transaction to be rolled back from the target transaction and the concurrent transaction comprises:

determining, based on transaction priorities, a transaction with a lowest transaction priority of the target transaction and the concurrent transaction as the transaction to be rolled back.

3. The method according to claim 2, wherein before determining, based on the transaction priorities, the transaction with the lowest transaction priority of the target transaction and the concurrent transaction as the transaction to be rolled back, the method further comprises one of:

determining, when a processing duration of a current transaction in the target transaction and the concurrent transaction is less than a processing duration of the other transaction, the current transaction as the transaction with the lowest transaction priority, wherein the current transaction is one of the target transaction and the concurrent transaction, and the other transaction is a transaction other than the current transaction in the target transaction and the concurrent transaction;

determining, when the current transaction in the target transaction and the concurrent transaction is a read transaction, the current transaction as the transaction with the lowest transaction priority;

determining, when the current transaction in the target transaction and the concurrent transaction is a write transaction and the processing duration of the current transaction is less than a reference duration, the current transaction as the transaction with the lowest transaction priority; or

determining, when the quantity of variables on which the current transaction in the target transaction and the concurrent transaction acts is greater than the quantity of variables on which the other transaction acts, the current transaction as the transaction with the lowest transaction priority.

4. The method according to claim 1, further comprising:

obtaining a first conflict cycle graph between the target transaction and the concurrent transaction, wherein vertices in the first conflict cycle graph are used for representing transaction identifiers corresponding to transactions, and edges in the first conflict cycle graph are used for representing an association between the transactions and variables on which a read and write operation corresponding to the transactions acts;

transforming the first conflict cycle graph, to obtain a second conflict cycle graph, wherein vertices in the second conflict cycle graph are used for representing variables in a plurality of different versions, and edges in the second conflict cycle graph are used for representing version changes between the variables and variables on which a same transaction acts; and

transforming the second conflict cycle graph, to obtain the target conflict cycle graph.

5. The method according to claim 4, wherein transforming the first conflict cycle graph, to obtain the second conflict cycle graph comprises:

splitting edges of the first conflict cycle graph, to obtain the variables in the plurality of different versions; and

using the variables in the plurality of different versions as vertices, adding an edge between vertices corresponding to a same variable in adjacent versions, and adding an edge between vertices corresponding to variables on which a same transaction acts, to obtain the second conflict cycle graph.

6. The method according to claim 4, wherein transforming the second conflict cycle graph, to obtain the target conflict cycle graph comprises:

merging vertices corresponding to the variables on which the same transaction acts in the second conflict cycle graph, and determining a transaction identifier corresponding to the same transaction as a merged vertex;

adding an edge between adjacent vertices of the merged second conflict cycle graph, to obtain a third conflict cycle graph;

when two adjacent vertices of the third conflict cycle graph comprise variables in a same version, merging the two adjacent vertices, and deleting the variables in the same version in the merged vertex, to obtain the target conflict cycle graph; and

when two adjacent vertices of the third conflict cycle graph comprise a same variable in different versions, merging the two adjacent vertices, to obtain the target conflict cycle graph.

7. The method according to any one of claims 1 to 3, further comprising:

when the first intersection is the non-empty data set, automatically increasing the quantity of directed edges between the target transaction and the concurrent transaction by one and adding variables in the first intersection to the target variable list, and when the first intersection comprises variables in different versions, automatically increasing the version change count by one, the quantity of the directed edges being used for indicating whether the target transaction and the concurrent transaction conflict with each other;

when the second intersection is the non-empty data set, automatically increasing the quantity of the directed edges by one and adding variables in the second intersection to the target variable list, and when the second intersection comprises variables in different versions, automatically increasing the version change count by one; and

when a third intersection is the non-empty data set, determining the data anomaly type based on a commit state of the concurrent transaction, wherein the third intersection is an intersection between the write set of the target transaction and the write set of the concurrent transaction.

8. The method according to any one of claims 1 to 3, wherein determining, based on the version change count of the same variable on which the target transaction and the concurrent transaction act and the target variable list, the data anomaly type comprises:

determining an affiliation type based on a target read and write operation in the target transaction and the concurrent transaction acting on the same variable;

determining a sub-type in the affiliation type based on the affiliation type and the target variable list; and

determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation.

9. The method according to claim 8, wherein determining the affiliation type based on the target read and write operation in the target transaction and the concurrent transaction acting on the same variable comprises one of:

determining the affiliation type as a write anomaly type when the target read and write operation in the target transaction and the concurrent transaction meets a write anomaly condition, wherein that the write anomaly condition is met means that the target read and write operation comprises a first designated quantity of write operations acting on a same variable in different versions;

determining the affiliation type as a read anomaly type when the target read and write operation does not meet the write anomaly condition and meets a read anomaly condition, wherein that the read anomaly condition is met means that the target read and write operation comprises a read operation and a write operation acting on a same variable in a same version; or

determining the affiliation type as an intersect anomaly type when the target read and write operation meets neither of the write anomaly condition and the read anomaly condition and meets an intersect anomaly condition, wherein that the intersect anomaly condition is met means that the target read and write operation comprises a read operation and a write operation acting on a same variable in different versions.

10. The method according to claim 8, wherein determining the sub-type in the affiliation

type based on the affiliation type and the target variable list comprises one of:

determining the sub-type in the affiliation type as a single-meta data anomaly when the quantity of variables in the target variable list is a second designated quantity;

determining the sub-type in the affiliation type as a double-meta data anomaly when the quantity of variables in the target variable list is a third designated quantity; or

determining the sub-type in the affiliation type as a multi-meta data anomaly when the quantity of variables in the target variable list is a fourth designated quantity, wherein the fourth designated quantity is different from each of the second designated quantity and the third designated quantity.

11. The method according to claim 8, wherein when the affiliation type is a write anomaly type and the sub-type is a single-meta data anomaly, determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises one of:

determining, when the version change count is a second designated quantity, the data anomaly type as a lost self-update anomaly based on the write anomaly type and the single-meta data anomaly;

determining, when the version change count is a third designated quantity and the target read and write operation is a fifth designated quantity of write operations, the data anomaly type as a full write anomaly based on the write anomaly type and the single-meta data anomaly; or

determining, when the version change count is the third designated quantity and the target read and write operation is a first designated quantity of write operations and a sixth designated quantity of read operations, the data anomaly type as a lost update anomaly based on the write anomaly type and the single-meta data anomaly.

12. The method according to claim 8, wherein when the affiliation type is a write anomaly type, the sub-type is a double-meta data anomaly, and the version change count is a second designated quantity, determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises one of:

determining, when the target read and write operation meets a first condition and a second condition, the data anomaly type as a read and write skew anomaly based on the write anomaly type, the double-meta data anomaly, and the second designated quantity, wherein that the first

condition is met means that the target read and write operation comprises a write operation and a read operation acting on a first variable in different versions, and that the second condition is met means that the target read and write operation comprises a first designated quantity of write operations acting on a second variable in different versions;

determining, when the target read and write operation meets a third condition and the second condition, the data anomaly type as a rewrite skew anomaly based on the write anomaly type, the double-meta data anomaly, and the second designated quantity, wherein that the third condition is met means that the target read and write operation comprises a write operation and a read operation acting on the first variable in a same version; or

determining, when the target read and write operation meets a fourth condition and the second condition, the data anomaly type as a full write skew anomaly based on the write anomaly type, the double-meta data anomaly, and the second designated quantity, wherein that the fourth condition is met means that the target read and write operation comprises the first designated quantity of write operations acting on the first variable in different versions.

13. The method according to claim 8, wherein when the affiliation type is a read anomaly type, the sub-type is a single-meta data anomaly, and the version change count is a second designated quantity, determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises one of:

determining, when the target read and write operation comprises a first designated quantity of read operations and a sixth designated quantity of write operations acting on a same variable, the data anomaly type as a non-repeatable read anomaly based on the read anomaly type, the single-meta data anomaly, and the second designated quantity; or

determining, when the target read and write operation comprises the first designated quantity of write operations and the sixth designated quantity of read operations acting on a same variable, the data anomaly type as an intermediate read anomaly based on the read anomaly type, the single-meta data anomaly, and the second designated quantity.

14. The method according to claim 8, wherein when the affiliation type is a read anomaly type and the sub-type is a double-meta data anomaly, determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises one of:

determining, when the version change count is a seventh designated quantity, the data anomaly type as a write read skew anomaly based on the read anomaly type and the double-meta data anomaly; or

determining, when the version change count is a second designated quantity, the data anomaly type as a read skew anomaly based on the read anomaly type and the double-meta data anomaly.

15. The method according to claim 8, wherein determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises:

determining the data anomaly type as a write skew anomaly when the affiliation type is an intersect anomaly type, the sub-type is a double-meta data anomaly, and the version change count is a second designated quantity.

16. The method according to claim 8, wherein determining the data anomaly type based on at least one of the sub-type in the affiliation type, the version change count, and the target read and write operation comprises one of:

determining the data anomaly type as a stepped write anomaly when the affiliation type is a write anomaly type and the sub-type is a multi-meta data anomaly;

determining the data anomaly type as a stepped read anomaly when the affiliation type is a read anomaly type and the sub-type is the multi-meta data anomaly; or

determining the data anomaly type as a stepped intersect anomaly when the affiliation type is an intersect anomaly type and the sub-type is the multi-meta data anomaly.

17. A transaction processing apparatus, comprising:

a first determining module, configured to determine a concurrent transaction of a target transaction, a read and write operation acting on a same variable being comprised between the concurrent transaction and the target transaction, and the target transaction being a transaction to be committed;

a second determining module, configured to obtain a first intersection between a read set of the target transaction and a write set of the concurrent transaction; obtain a second intersection between a write set of the target transaction and a read set of the concurrent transaction; and in response to at least one of the first intersection and the second intersection being a non-empty data set and the target transaction conflicting with the concurrent transaction, determine, based

on a version change count of the same variable on which the target transaction and the concurrent transaction act and a target variable list, a data anomaly type, the target variable list including the same variable on which the target transaction and the concurrent transaction act;

a third determining module, configured to determine a transaction to be rolled back from the target transaction and the concurrent transaction;

a rollback module, configured to roll back the transaction; and

a deleting module, configured to delete a variable on which the transaction to be rolled back acts and a corresponding read and write operation from a target conflict cycle graph, wherein vertices in the target conflict cycle graph are used for representing transaction identifiers corresponding to transactions and variables on which a read and write operation corresponding to the transactions acts, and edges in the target conflict cycle graph are used for representing an association between the transactions and version changes between the variables.

18. An electronic device, comprising a processor and a memory, the memory being configured to store at least one computer program, and the at least one computer program being loaded and executed by the processor, to implement the transaction processing method for a database system according to any one of claims 1 to 16.

19. A computer-readable storage medium, storing at least one computer program, and the at least one computer program being loaded and executed by a processor, to implement the transaction processing method for a database system according to any one of claims 1 to 16.

20. A computer program product, comprising a computer program or instruction, the computer program or instruction, when executed by a processor, implementing the transaction processing method for a database system according to any one of claims 1 to 16.