There are several ways to improve the performance of the database.

1: database system part

As shown in the situation that the problem description, the database is likely to have a deadlock.First check the logs to find if there is an dead lock, if it exists, then execute dbcc traceon to enable deadlock tracking when the database instance starts. Find the tables and query where deadlocks occur. Kill the deadlock and improve the query.

Graphical user interface, text, application

Description automatically generated

Also we can use this to find lock table or lock process and also unlock it

Graphical user interface, text, application, email

Description automatically generated

Second, if there is no deadlock then create Extended Events and Execution Plan to view any potential index placement and check if there is missing index or there is useless index. In the missing index, add an index to improve the efficiency of the search. At the same time delete 0 use index or low use index to improve the efficiency of insert and update. Like this

Graphical user interface, application

Description automatically generated

Meanwhile we can also use TUNING ADVISOR to automatically locate missing indexes and Use DMV to find the content corresponding to each index and the missing index

Third, View execution time and cpu to check if we need upgrade the hardware. Like this

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

SQL query Tuning

1. Don't use select \*, because it may scan the whole table

2. Use parameter query

3. Use exists or not exists instead of in or not in

4. Try to avoid using the leading percent sign.

5. Try to avoid using the != or <> operator in the where clause, otherwise the engine will give up using the index and perform a full table scan.

6, should consider creating an index on the columns involved in where and order by.

7. Try to avoid the null value judgment of the field in the where clause, otherwise it will lead to a full table scan.

8. Avoid using or to join conditions in the where clause, otherwise it will result in a full table scan.

9. Try to avoid performing expression operations on fields in the where clause, otherwise it will lead to a full table scan.

10, try to avoid performing functional operations on the fields in the where clause, otherwise it will lead to a full table scan.

11. Use numeric fields as much as possible. If fields that only contain numerical information should not be designed as character types, this will reduce the performance of queries and connections and increase storage overhead.

12. Use varchar/nvarchar instead of char/nchar as much as possible, the storage space of variable-length fields will be smaller, which can save storage space; and for queries, searching in a relatively small field is more efficient.

13. Try to avoid using cursors, and consider rewriting cursors as much as possible when processing large amounts of data

14. Use set rowcount to achieve high-performance paging.

15. Use bulk select instead of select

16. remove correlated subqueries

17. remove useless indexes

18. remove useless groupbys