SQL Server 海量数据查询代码优化以及建议

具体要注意的：
1.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：
select id from t where num is null
可以在 num 上设置默认值 0，确保表中 num 列没有 null 值，然后这样查询：
select id from t where num=0

2.应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。优化器将无法通过索引来确定将要命中的行数,因此需要搜索该表的所有行。

3.应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如：
select id from t where num=10 or num=20
可以这样查询：
select id from t where num=10
union all
select id from t where num=20

4.in 和 not in 也要慎用，因为 IN 会使系统无法使用索引,而只能直接搜索表中的数据。如：
select id from t where num in(1,2,3)
对于连续的数值，能用 between 就不要用 in 了：
select id from t where num between 1 and 3

5.尽量避免在索引过的字符数据中，使用非打头字母搜索。这也使得引擎无法利用索引。
见如下例子：
SELECT * FROM T1 WHERE NAME LIKE '%L%'
SELECT * FROM T1 WHERE SUBSTING(NAME,2,1)='L'
SELECT * FROM T1 WHERE NAME LIKE 'L%'
即使 NAME 字段建有索引，前两个查询依然无法利用索引完成加快操作，引擎不得不对全表所有数据逐条操作来完成任务。而第三个查询能够使用索引来加快操作。

6.必要时强制查询优化器使用某个索引，如在 where 子句中使用参数，也会导致全表扫描。因为 SQL 只有在运行时才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然 而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：
select id from t where num=@num
可以改为强制查询使用索引：

select id from t with(index(索引名)) where num=@num

7.应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。如：
SELECT * FROM T1 WHERE F1/2=100
应改为:
SELECT * FROM T1 WHERE F1=100*2
SELECT * FROM RECORD WHERE SUBSTRING(CARD_NO,1,4)='5378'
应改为:
SELECT * FROM RECORD WHERE CARD_NO LIKE '5378%'
SELECT member_number, first_name, last_name FROM members
WHERE DATEDIFF(yy,datofbirth,GETDATE()) > 21
应改为:
SELECT member_number, first_name, last_name FROM members
WHERE dateofbirth < DATEADD(yy,-21,GETDATE())
即：任何对列的操作都将导致表扫描，它包括数据库函数、计算表达式等等，查询时要尽可能将操作移至等号右边。

8.应尽量避免在 where 子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描。如：
select id from t where substring(name,1,3)='abc'--name 以 abc 开头的 id
select id from t where datediff(day,createdate,'2005-11-30')=0--'2005-11-30'生成的 id
应改为:
select id from t where name like 'abc%'
select id from t where createdate>='2005-11-30' and createdate<'2005-12-1'

9.不要在 where 子句中的"="左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

10.在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

11.很多时候用 exists 是一个好的选择：
select num from a where num in(select num from b)
用下面的语句替换：
select num from a where exists(select 1 from b where num=a.num)
SELECT SUM(T1.C1)FROM T1 WHERE(
(SELECT COUNT(*)FROM T2 WHERE T2.C2=T1.C2>0)

SELECT SUM(T1.C1) FROM T1WHERE EXISTS(
SELECT * FROM T2 WHERE T2.C2=T1.C2)
两者产生相同的结果，但是后者的效率显然要高于前者。因为后者不会产生大量锁定的表扫描或是索引扫描。
如果你想校验表里是否存在某条纪录，不要用count(*)那样效率很低，而且浪费服务器资源。可以用 EXISTS 代替。如：
IF (SELECT COUNT(*) FROM table_name WHERE column_name = 'xxx')
可以写成：
IF EXISTS (SELECT * FROM table_name WHERE column_name = 'xxx')
经常需要写一个 T_SQL 语句比较一个父结果集和子结果集，从而找到是否存在在父结果集中有而在子结果集中没有的记录，如：
SELECT a.hdr_key FROM hdr_tbl a---- tbl a 表示 tbl 用别名 a 代替
WHERE NOT EXISTS (SELECT * FROM dtl_tbl b WHERE a.hdr_key = b.hdr_key)
SELECT a.hdr_key FROM hdr_tbl a
LEFT JOIN dtl_tbl b ON a.hdr_key = b.hdr_key WHERE b.hdr_key IS NULL
SELECT hdr_key FROM hdr_tbl
WHERE hdr_key NOT IN (SELECT hdr_key FROM dtl_tbl)
三种写法都可以得到同样正确的结果，但是效率依次降低。

12.尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

13.避免频繁创建和删除临时表，以减少系统表资源的消耗。

14.临时表并不是不可使用，适当地使用它们可以使某些例程更有效，例如，当需要重复引用大型表或常用表中的某个数据集时。但是，对于一次性事件，最好使用导出表。

15.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先 create table，然后 insert。

16.如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

17.在所有的存储过程和触发器的开始处设置 SET NOCOUNT ON ，在结束时设置 SET NOCOUNT OFF 。无需在执行存储过程和触发器的每个语句后向客户端发送 DONE_IN_PROC 消息。

18.尽量避免大事务操作，提高系统并发能力。

19.尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

20. 避免使用不兼容的数据类型。例如 float 和 int、char 和 varchar、binary 和 varbinary 是不兼容的。数据类型的不兼容可能使优化器无法执行一些本来可以进行的优化操作。例如:
SELECT name FROM employee WHERE salary > 60000
在这条语句中,如 salary 字段是 money 型的,则优化器很难对其进行优化,因为 60000 是个整型数。我们应当在编程时将整型转化成为钱币型,而不要等到运行时转化。

21.充分利用连接条件，在某种情况下，两个表之间可能不只一个的连接条件，这时在 WHERE 子句中将连接条件完整的写上，有可能大大提高查询速度。
例：
SELECT SUM(A.AMOUNT) FROM ACCOUNT A,CARD B WHERE A.CARD_NO = B.CARD_NO
SELECT SUM(A.AMOUNT) FROM ACCOUNT A,CARD B WHERE A.CARD_NO = B.CARD_NO AND A.ACCOUNT_NO=B.ACCOUNT_NO
第二句将比第一句执行快得多。

22、使用视图加速查询
把表的一个子集进行排序并创建视图，有时能加速查询。它有助于避免多重排序 操作，而且在其他方面还能简化优化器的工作。例如：
SELECT cust.name，rcvbles.balance，……other columns
FROM cust，rcvbles
WHERE cust.customer_id = rcvlbes.customer_id
AND rcvblls.balance>0
AND cust.postcode>“98000”
ORDER BY cust.name
如果这个查询要被执行多次而不止一次，可以把所有未付款的客户找出来放在一个视图中，并按客户的名字进行排序：
CREATE VIEW DBO.V_CUST_RCVLBES
AS
SELECT cust.name，rcvbles.balance，……other columns
FROM cust，rcvbles
WHERE cust.customer_id = rcvlbes.customer_id
AND rcvblls.balance>0
ORDER BY cust.name
然后以下面的方式在视图中查询：
SELECT ＊ FROM V_CUST_RCVLBES
WHERE postcode>“98000”
视图中的行要比主表中的行少，而且物理顺序就是所要求的顺序，减少了磁盘 I/O，所以查

询工作量可以得到大幅减少。

23、能用 DISTINCT 的就不用 GROUP BY
SELECT OrderID FROM Details WHERE UnitPrice > 10 GROUP BY OrderID
可改为：
SELECT DISTINCT OrderID FROM Details WHERE UnitPrice > 10

24.能用 UNION ALL 就不要用 UNION
UNION ALL 不执行 SELECT DISTINCT 函数，这样就会减少很多不必要的资源

25.尽量不要用 SELECT INTO 语句。
SELECT INOT 语句会导致表锁定，阻止其他用户访问该表。
上面我们提到的是一些基本的提高查询速度的注意事项,但是在更多的情况下,往往需要反复试验比较不同的语句以得到最佳方案。最好的方法当然是测试，看实现 相同功能的 SQL 语句哪个执行时间最少，但是数据库中如果数据量很少，是比较不出来的，这时可以用查看执行计划，即：把实现相同功能的多条 SQL 语句考到 查询分析器，按 CTRL+L 看查所利用的索引，表扫描次数（这两个对性能影响最大），总体上看询成本百分比即可。