数据库SQL语句编写审核标准

1. 不规范的SQL语句
2. select \* from t1

这种带\*的通配符语句，造成全表扫描在高并发和大数据量情况下影响性能。

1. select id from t where num is null

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。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

1. select id from t where num=10 or num=20

应尽量避免在 where 子句中使用大量的 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

1. in 和 not in 也要慎用

select id from t where id in (1,2,3,4,5) 这种情况是可以的。

Select id from t where id in (select tid from t1)这种情况执行效率就比较低下，可以用inner join进行关联查询。

1. 很多时候用 exists 代替 in 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)

1. 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 NAME LIKE ‘L%’

即使NAME字段建有索引，第一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。而第二个查询能够使用索引来加快操作。

1. 任何情况下SELECT COUNT(\*) FROM T 是最优选择，尽量减少SELECT COUNT(\*) FROM T WHERE COL = ‘xxx’ 这种查询,杜绝SELECT COUNT(COL) FROM T WHERE COL = ‘xxx’ 的出现。(其中COL非主键)
2. 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：对字段进行函数式操作，这将导致数据库引擎放弃使用索引而进行全表扫描。，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。

10. 能用DISTINCT的就不用GROUP BY （group by 操作特别慢）

SELECT OrderID FROM Details WHERE UnitPrice > 10 GROUP BY OrderID

可改为：

SELECT DISTINCT OrderID FROM Details WHERE UnitPrice > 10

11. 能用UNION ALL就不要用UNION ，UNION ALL不执行数据排重，这样就会减少很多不必要的资源。

12.尽量避免使用游标，带有事务的语句可以封装到存储过程中执行。

13.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。

14. 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。

15. 避免频繁创建和删除临时表，以减少系统表资源的消耗。

16. 尽可能的使用 varchar 代替 char/nchar ，因为变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率要高些。