加快sqlserver的运行速度 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/139/2021_2022__E5_8A_A0_E5_BF_ABsqls_c100_139685.htm ---- 人们在使用SQL时往往会陷入一个误区，即太关注于所得的结果是否正确，而忽略了不同的实现方法之间可能存在的性能差异，这种性能差异在大型的或是复杂的数据库环境中（如联机事务处理OLTP或决策支持系统DSS）中表现得尤为明显。笔者在工作实践中发现，不良的SQL往往来自于不恰当的索引设计、不充份的连接条件和不可优化的where子句。在对它们进行适当的优化后，其运行速度有了明显地提高！下面我将从这三个方面分别进行总结： ---- 为了更直观地说明问题，所有实例中的SQL运行时间均经过测试，不超过１秒的均表示为（---- 测试环境------ 主机：HP LH II---- 主频：330MHZ---- 内存：128兆---- 操作系统：Operserver5.0.4----数据库：Sybase11.0.3一、不合理的索引设计----例：表record有620000行，试看在不同的索引下，下面几个SQL的运行情况：---- 1.在date上建有一非个群集索引0select count(*) from record where date >19991201 and date 2000 (25秒)0select date,sum(amount) from record group by date(55秒)0select count(*) from record where date >19990901 and place in (BJ,SH) (27秒)---- 分析：----date上有大量的重复值，在非群集索引下，数据在物理上随机存放在数据页上，在范围查找时，必须执行一次表扫描才能找到这一范围内的全部行。---- 2.在date上的一个群集索引0select count(*) from record where date >19991201 and date 2000 （14秒）0select date,sum(amount) from record group by date（28秒）0select

count(*) from record where date >19990901 and place in (BJ,SH)（14秒）---- 分析：---- 在群集索引下，数据在物理上按顺序在数据页上，重复值也排列在一起，因而在范围查找时，可以先找到这个范围的起末点，且只在这个范围内扫描数据页，避免了大范围扫描，提高了查询速度。//from www.w3sky.com---- 3.在place，date，amount上的组合索引0select count(*) from record where date >19991201 and date 2000（26秒）0select date,sum(amount) from record group by date（27秒）0select count(*) from record where date >19990901 and place in (BJ, SH)（---- 分析：---- 这是一个不很合理的组合索引，因为它的前导列是place，第一和第二条SQL没有引用place，因此也没有利用上索引；第三个SQL使用了place，且引用的所有列都包含在组合索引中，形成了索引覆盖，所以它的速度是非常快的。---- 4.在date，place，amount上的组合索引0select count(*) from record where date >19991201 and date 2000(0select date,sum(amount) from record group by date（11秒）0select count(*) from record where date >19990901 and place in (BJ,SH)（---- 分析：---- 这是一个合理的组合索引。它将date作为前导列，使每个SQL都可以利用索引，并且在第一和第三个SQL中形成了索引覆盖，因而性能达到了最优。---- 5.总结：---- 缺省情况下建立的索引是非群集索引，但有时它并不是最佳的；合理的索引设计要建立在对各种查询的分析和预测上。一般来说：//from www.w3sky.com---- ①.有大量重复值、且经常有范围查询（between, >,=,、group by发生的列，可考虑建立群集索引；---- ②.经常同时存取多列，且每列都含有重复值可考虑建立组合索引；---- ③.组合索引要尽量使

关键查询形成索引覆盖，其前导列一定是使用最频繁的列。二、不充份的连接条件：---- 例：表card有7896行，在card_no上有一个非聚集索引，表account有191122行，在 account_no上有一个非聚集索引，试看在不同的表连接条件下，两个SQL的执行情况：0select sum(a.amount) from account a,card b where a.card_no = b.card_no（20秒）---- 将SQL改为：0select sum(a.amount) from account a,card b where a.card_no = b.card_no and a.account_no=b.account_no（---- 分析：---- 在第一个连接条件下，最佳查询方案是将account作外层表，card作内层表，利用card上的索引，其I/O次数可由以下公式估算为：---- 外层表account上的22541页 （外层表account的191122行*内层表card上对应外层表第一行所要查找的3页）=595907次I/O---- 在第二个连接条件下，最佳查询方案是将card作外层表，account作内层表，利用account上的索引，其I/O次数可由以下公式估算为：---- 外层表card上的1944页 （外层表card的7896行*内层表account上对应外层表每一行所要查找的4页）= 33528次I/O---- 可见，只有充份的连接条件，真正的最佳方案才会被执行。---- 总结：---- 1.多表操作在被实际执行前，查询优化器会根据连接条件，列出几组可能的连接方案并从中找出系统开销最小的最佳方案。连接条件要充份考虑带有索引的表、行数多的表；内外表的选择可由公式：外层表中的匹配行数*内层表中每一次查找的次数确定，乘积最小为最佳方案。---- 2.查看执行方案的方法-- 用set showplanon，打开showplan选项，就可以看到连接顺序、使用何种索引的信息；想看更详细的信息，需用sa角色执行dbcc(3604,310,302)。三、不可优化的where子句---- 1.例：下列SQL条件语句中的列

都建有恰当的索引，但执行速度却非常慢：0select * from record wheresubstring(card_no,1,4)=5378(13秒)0select * from record whereamount/300select * from record whereconvert(char(10),date,112)=19991201（10秒）---- 分析：---- where子句中对列的任何操作结果都是在SQL运行时逐列计算得到的，因此它不得不进行表搜索，而没有使用该列上面的索引；如果这些结果在查询编译时就能得到，那么就可以被SQL优化器优化，使用索引，避免表搜索，因此将SQL重写成下面这样：0select * from record where card_no like5378%（0select * from record where amount0select * from record where date= 1999/12/01（ ---- 你会发现SQL明显快起来！---- 2.例：表stuff有200000行，id_no上有非群集索引，请看下面这个SQL：0select count(*) from stuff where id_no in(0,1)（23秒）---- 分析：---- where条件中的in在逻辑上相当于or，所以语法分析器会将in (0,1)转化为id_no =0 or id_no=1来执行。我们期望它会根据每个or子句分别查找，再将结果相加，这样可以利用id_no上的索引；但实际上（根据showplan）,它却采用了"OR策略"，即先取出满足每个or子句的行，存入临时数据库的工作表中，再建立唯一索引以去掉重复行，最后从这个临时表中计算结果。因此，实际过程没有利用id_no上索引，并且完成时间还要受tempdb数据库性能的影响。---- 实践证明，表的行数越多，工作表的性能就越差，当stuff有620000行时，执行时间竟达到220秒！还不如将or子句分开：0select count(*) from stuff where id_no=00select count(*) from stuff where id_no=1---- 得到两个结果，再作一次加法合算。因为每句都使用了索引，执行时间只有3秒，在620000行下，时间也只

有4秒。或者，用更好的方法，写一个简单的存储过程：create proc count_stuff asdeclare @a intdeclare @b intdeclare @c intdeclare @d char(10)begin0select @a=count(*) from stuff where id_no=00select @b=count(*) from stuff where id_no=1end0select @c=@a @b0select @d=convert(char(10),@c)print @d---- 直接算出结果，执行时间同上面一样快！---- 总结：---- 可见，所谓优化即where子句利用了索引，不可优化即发生了表扫描或额外开销。---- 1.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。---- 2.in、or子句常会使用工作表，使索引失效；如果不产生大量重复值，可以考虑把子句拆开；拆开的子句中应该包含索引。//from www.w3sky.com---- 3.要善于使用存储过程，它使SQL变得更加灵活和高效。---- 从以上这些例子可以看出，SQL优化的实质就是在结果正确的前提下，用优化器可以识别的语句，充份利用索引，减少表扫描的I/O次数，尽量避免表搜索的发生。其实SQL的性能优化是一个复杂的过程，上述这些只是在应用层次的一种体现，深入研究还会涉及数据库层的资源配置、网络层的流量控制以及操作系统层的总体设计。 