Atitit 数据存储的数据表连接attilax总结

[1.1.](#_Toc17117) [三种物理连接运算符：嵌套循环连接、合并连接以及哈希连接 1](#_Toc17117)

[1.2. a、](#_Toc7063)[嵌套循环连接（nested loops join) 1](#_Toc7063)

[1.3. b、](#_Toc13658)[合并连接(merge join) 3](#_Toc13658)

[1.4. c、](#_Toc4078)[哈希连接(hash join)  4](#_Toc4078)

[1.5. 所以这三种算法，没有谁好谁坏，只有合适的场景应用合适的连接算法，这样才能发挥它自身的长处，而恰巧这些就是我们要掌握的技能。 6](#_Toc3469)

[1.6. 这三种连接算法我们也可以显示的指定，loop merger hash 6](#_Toc7630)

[1.7. 所有的最优计划的选择都是基于现有统计信息来评估， 7](#_Toc32282)

[1.8. 参考资料 7](#_Toc25787)

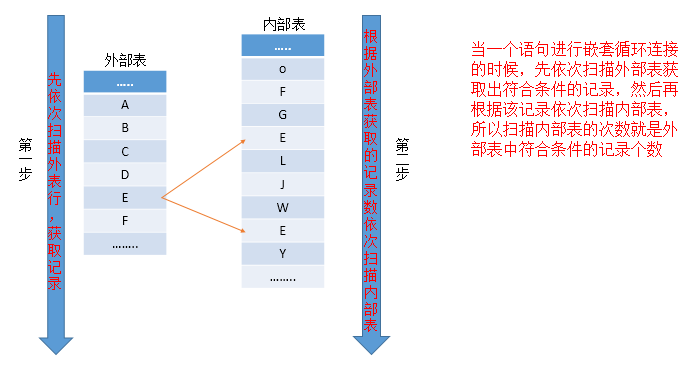
## 三种物理连接运算符：嵌套循环连接、合并连接以及哈希连接

默认支持三种物理连接运算符：嵌套循环连接、合并连接以及哈希连接。三种连接各有用途，各有特点，不同的场景会数据库会为我们选择最优的连接方式。

## a、嵌套循环连接（nested loops join)

嵌套循环连接是最简单也是最基础的连接方式。两张表通过关键字进行关联，然后通过双层循环依次进行两张表的行进行关联，然后通过关键字进行筛选。

可以参照下图进行理解分析



其实嵌套扫描是很简单的获取数据的方式，简单点就是两层循环过滤出结果值

我们可以通过如下代码加深理解

for each row R1 in the outer table

for each row R2 int the inner table

if R1 join with R2

return (R1,R2)

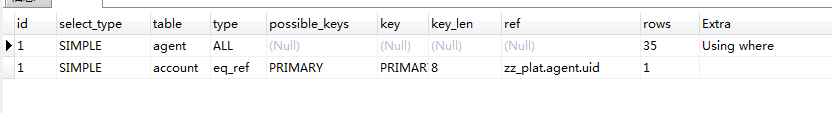
举个列子

SELECT o.OrderID

FROM Customers C JOIN Orders O

ON C.CustomerID=O.CustomerID

WHERE C.City=N'London'



这种方法的消耗就是外表和内表的乘积，其实就是我们所称呼的笛卡尔积。所以消耗的大小是随着两张表的数据量增大而增加的，尤其是内部表，因为它是多次重复扫描的，所以我们在实践中的采取的措施就是减少每个外表或者内表的行数来减少消耗。

对于这种算法还有一种提高性能的方式，因为两张表是通过关键字进行关联的，所以在查询的时候对于底层的数据获取速度直接关乎着此算法的性能，这里优化的方式尽量使用两个表关键字为索引查询，提高查询速度。

还有一点就是在嵌套循环连接中，在两张表关联的时候，对外表都是有筛选条件的，比如上面例子中【WHERE C.City=N'London'】就是对外表（Customers）的筛选，并且这里的City列在该表中存在索引，所以该语句的两个子查询都为索引查找（Index Seek）。

但是，有些情况我们的查询条件不是索引所覆盖的，这时候，在嵌套循环连接下的子运算符就变成了索引扫描（Index scan）或者RID查找。

举个例子

通过文本可以看出，该T-SQL的查询结果的获取是通过在嵌套循环运算符中，对两个表经过全表扫描之后形成的笛卡儿积进行过滤筛选的。这种方式其实不是一个最优的方式，因为我们获取的结果其实是可以先通过两个表过滤之后，再通过嵌套循环运算符获取结果，这样的话性能会好很多。

我们尝试改一下这个语句

[IMG_256](http://www.cnblogs.com/zhijianliutang/p/javascript:void(0);)

SELECT E1.EmployeeID,ECNT.CNT

FROM Employees E1 CROSS APPLY

(

SELECT COUNT(\*) CNT

FROM Employees E2

WHERE E1.HireDate<E2.HireDate

)ECNT

[IMG_257](http://www.cnblogs.com/zhijianliutang/p/javascript:void(0);)

通过上述代码查询的结果项，和上面的是一样的，只是我们根据外部表的结果对内部表进行了过滤，这样执行的时候就不需要获取全部数据项了

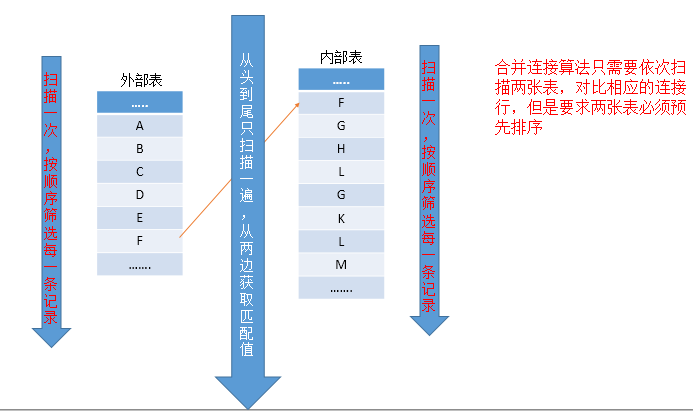
所以对嵌套循环连接连接的优化方式就是集中在这几点：对两张表数据量的减少、连接关键字上建立索引、谓词查询条件上覆盖索引最好能减少符合谓词条件的记录数。

## b、合并连接(merge join)

上面提到的嵌套循环连接方式存在着诸多的问题，尤其不适合两张表都是大表的情况下，因为它会产生N多次的全表扫描，很显然这种方式会严重的消耗资源。

鉴于上述原因，在数据库里又提供了另外一种连接方式：合并连接。记住这里没有说SQL Server所提供的，是因为此连接算法是市面所有的RDBMS所共同使用的一种连接算法。

合并连接是依次读取两张表的一行进行对比。如果两个行是相同的，则输出一个连接后的行并继续下一行的读取。如果行是不相同的，则舍弃两个输入中较少的那个并继续读取，一直到两个表中某一个表的行扫描结束，则执行完毕，所以该算法执行只会产生每张表一次扫描，并且不需要整张表扫描完就可以停止。



该算法要求按照两张表进行依次扫描对比，但是有两个前提条件：1、必须预先将两张表的对应列进行排序；2、对两张表进行合并连接的条件必须存在等值连接。

## c、哈希连接(hash join)

我们分析了上面的两种连接算法，两种算法各有特点，也各有自己的应用场景：嵌套循环连接适合于相对小的数据集连接，合并连接则应对与中型的数据集，但是又有它自己的缺点，比如要求必须有等值连接，并且需要预先排序等。

那对于大型的数据集合的连接数据库是怎么应对的呢？那就是哈希连接算法的应用场景了。

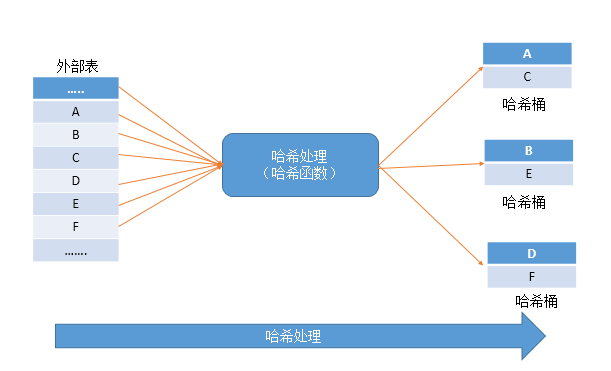
哈希连接对于大型数据集合的并行操作上都比其它方式要好很多，尤其适用于OLAP数据仓库的应用场景中。

哈希连接很多地方和合并连接类似，比如都需要至少一个等值连接，同样支持所有的外连接操作。但不同于合并连接的是，哈希连接不需要预先对输入数据集合排序，我们知道对于大表的排序操作是一个很大的消耗，所以去除排序操作，哈希操作性能无疑会提升很多。

哈希连接在执行的时候分为两个阶段：

* 构建阶段

在构建阶段，哈希连接从一个表中读入所有的行，将等值连接键的行机型哈希话处理，然后创建形成一个内存哈希表，而将原来列中行数据依次放入不同的哈希桶中。



* 探索阶段

在第一个阶段完成之后，开始进入第二个阶段探索阶段，该阶段哈希连接从第二个数据表中读入所有的行，同样也是在相同的等值连接键上进行哈希。哈希过程桶上一阶段，然后再从哈希表中探索匹配的行。

上述的过程中，在第一个阶段的构建阶段是阻塞的，也就是说在，哈希连接必须读入和处理所有的构建输入，之后才能返回行。而且这一过程是需要一块内存存储提供支持，并且利用的是哈希函数，所以相应的也会消耗CPU等。

并且上述流程过程中一般采用的是并发处理，充分利用资源，当然系统会对哈希的数量有所限制，如果数据量超大，也会发生内存溢出等问题，而对于这些问题的解决，SQL Server有它自身的处理方式。

我们可通过以下代码进行理解

[IMG_257](http://www.cnblogs.com/zhijianliutang/p/javascript:void(0);)

我们来总结一下这个算法的特点

* 和合并连接一样算法复杂度基本就是分别遍历两边的数据集各一遍
* 它不需要对数据集事先排序，也不要求上面有什么索引，通过的是哈希算法进行处理
* 基本采取并行的执行计划的方式

 但是，该算法也有它自身的缺点，因为其利用的是哈希函数，所以运行时对CPU消耗高，同样对内存也比较大，但是它可以采用并行处理的方式，所以该算法用于超大数据表的连接查询上显示出自己独有的优势。

关于哈希算法在哈希处理过程的时候对内存的占用和分配方式，是有它自己独有哈希方法，比如：左深度树、右深度树、浓密哈希连接树等，这里不做详细介绍了，只需要知道其使用方式就可以了。

Hash Join并不是一种最优的连接算法，只是它对输入不优化，因为输入数据集特别大，并且对连接符上有没有索引也没要求。其实这也是一种不得已的选择，但是该算法又有它适应的场景，尤其在OLAP的数据仓库中，在一个系统资源相对充足的环境下，该算法就得到了它发挥的场景。

当然前面所介绍的两种算法也并不是一无是处，在业务的OLTP系统库中，这两种轻量级的连接算法，以其自身的优越性也获得了认可。

## 所以这三种算法，没有谁好谁坏，只有合适的场景应用合适的连接算法，这样才能发挥它自身的长处，而恰巧这些就是我们要掌握的技能。

## 这三种连接算法我们也可以显示的指定，loop merger hash

但是一般不建议这么做，因为默认SQL Server会为我们评估最优的连接方式进行操作，当然有时候它评估不对的时候就需要我们自己指定了，方法如下：

## 所有的最优计划的选择都是基于现有统计信息来评估，

而上面关于这两列内容分布类型SQL Server是怎样知道的？这就是SQL Server的强大的统计信息在支撑了。

在SQL Server中并不是固定的语句就会形成特定的计划，并且生成的特定计划也不是总是最优的，这和数据库现有数据表中的内容分布、数据量、数据类型等诸多因素有关，而记录这些详细信息的就是统计信息。

所有的最优计划的选择都是基于现有统计信息来评估，如果我们的统计信息未及时更新，那么所评估出来最优的执行计划将不是最好的，有时候反而是最烂的。

## 参考资料

SQL Server调优系列基础篇（常用运算符总结——三种物理连接方式剖析） - 指尖流淌 - 博客园.html

作者:: 绰号:老哇的爪子claw of Eagle 偶像破坏者Iconoclast image-smasher

捕鸟王"Bird Catcher 王中之王King of Kings 虔诚者Pious 宗教信仰捍卫者 Defender of the Faith. 卡拉卡拉红斗篷 Caracalla red cloak

简称：： Emir Attilax Akbar 埃米尔 阿提拉克斯 阿克巴

全名：：Emir Attilax Akbar bin Mahmud bin attila bin Solomon Al Rapanui

埃米尔 阿提拉克斯 阿克巴 本 马哈茂德 本 阿提拉 本 所罗门  阿尔 拉帕努伊

常用名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源：attilax的专栏  http://blog.csdn.net/attilax

--Atiend