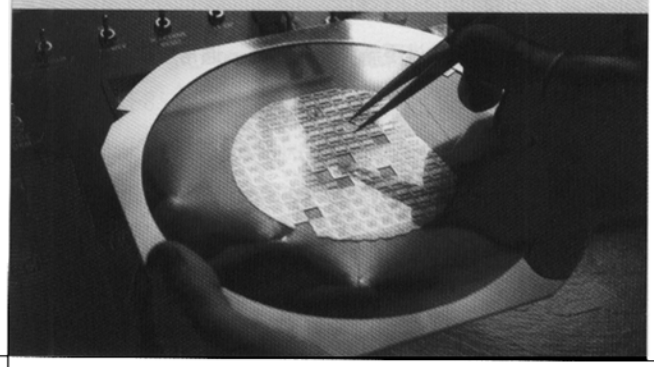
童亚凤 （绍兴市人民医院 312000）
王庆君 （绍兴市商业银行）



# 数据库的查询优化策略

## Informix-Online Dynanic Server(IDS)

**摘 要：** 本文主要阐明INFORMIX数据库系统针对不同应用从查询的优化策略、数据库系统参数性能调整、索引、库表创建方式、合理使用DBSPACE、优化方法和如何查询优化结果等方面提出了提高INFOMRIX数据库运行效率的若干策略和措施。

**关键词：** INFORMIX 数据库 优化策略

Informix－Online 动态服务器（IDS,Informix－Online Dynamic Server）作为Informix数据库产品技术的核心，以其动态可伸缩体系结构，高效的并行处理能力、共享内存技术及易管理性等特点，将硬件资源发挥得淋漓尽致。当前IDS V7正广泛地应用于我国金融、邮政、电信等行业的关键系统中，随着应用的不断深入,数据的积累,查询的复杂化,查询速度会变慢，致使响应时间过长。许多用户将其归结为硬件的原因,于是升级改造，或重新投资,数据库的调优并没有引起足够的重视。这其实是一种浪费，与国外发达国家轻硬件、重应用的思路正好相反。实践表明,数据库的不合理配置和不适当优化是其性能下降的主要因素。实施对IDS上数据库的管理维护、性能调优是系统管理员的主要工作，而能否得以良好的查询响应则集中体现了数据库的性能，因此也是调优的重点。从系统管理的角度，我们可以设置多线索、合理分配共享内存空间、建立数据库和表的分布及分片管理等来加快查询速度，但最终还要基于对数据库本身的全面理解，因为数据处于不断的变化和积累之中，并且随着应用的深入查询将日趋复杂化。下面根据INFORMIX－ONLINE的特点提出一些优化查询的建议和方法。实际应用中有很好的收效。

## 1 查询的分类及要求

针对语句中所涉及的数据库表的数目查询可分为：单表查询、多表查询、联合查询、子查询等，多表查询建立在多张表的连接之上，分嵌套循环、合并排序、哈希连接三种方式，最为复杂，也是调优的重点。目前数据库的应用分为联机事务处理(OLTP,Online Transaction Processing）和决策支持系统(DSS,Decision Support System）两大类型，它们对查询的要求不尽相同。OLTP主要涉及单张表，SQL语句简单，数据按索引读取，查询行数少，对响应的时间要求非常苛刻，常在秒级或以下，多用于在线实时业务；DSS涉及多张表之间的连接查询，SQL语句复杂，数据按物理顺序读取，查询行数多，响应时间长，多用于建立在数据仓库技术之上的复杂的数据分析。但无论何种情况，我们都希望最快的响应速度，这也是调优的最终目标。

## 2 查询的优化策略

### 2.1 充分利用查询优化器

查询优化器提供了数据查询的优化策略分析和选择方式，通过设置相关参数，优化器能够选择最佳的连接策略，并在所有的查询路径中找出一条最优路径。选择良好的路径是查询优化中至关重要的一环，一条好的路径可以扫描最少的记录，以最少的磁盘I/O得到正确的查询结果。可通过以下步骤进行。

（1）设置连接策略

通过修改配置文件 $ ONCONFIG中的OPTCOMPIND参数值来实现。

OPTCOMPIND 0：在连接中优化器只选择索引连接。

OPTCOMPIND 1：若事务处理为可重复读模式（Repeatable Read）,则选择索引策略，否则，优化器自动选择开销最低的连接策略。

OPTCOMPIND 2：优化器自动选择开销最低的连接策略。应尽量选择该参数。

（2）设置查询优化的模式。也即选择最优的查询路径，通过执

行以下SQL语句来实现，格式为：

SET OPTIMIZATION [ HIGH | LOW | FIRST_ROWS | ALL_ROWS ]

其中：

HIGH是缺省选项，表示对所有查询路径都进行检测，从中选择最优。

LOW表示采用深度优先法仅在部分路径中选择最优，即在每次连接比较中，遇到最优路径就继续深入而滤掉非最优路径，特点是优化时间短，但路径准确率低。FIRST_ROWS和ALL_ROWS是自IDS V7.3开始增加的新选项，无论对OLTP还是DSS都非常有用。传统的查询(即ALL_ROWS方式)一次将所有查询结果输出到共享内存缓冲区中，时间的消耗非常大，然而实践表明，大部分用户仅关注最初的几屏输出内容，因此FIRST_ROWS选项为我们提供了很好的选择。FIRST_ROWS指导优化器选择一条查询路径,使其只输出填满一个缓冲区的记录数，如果用户继续查询则继续执行，这样避免了不必要的输出结果和时间浪费，也使查询速度大大提高。

由此可见，优化器的丰富功能为我们提供了灵活的手段，管理员可以根据不同应用情况选择最佳的方式，既能达到最佳的查询效果，又能将由此而造成的系统开销降至最低。

### 2.2 不定期进行统计更新和数据分布

当数据库表做了大量的插入、删除操作或表的索引发生变化后，Online数据库系统表的相关信息与实际表的统计数字不一致，这对数据的完整性没有任何损害，但会影响到查询的速度。因为优化器所制定的计划和策略得以正确实施的前提是对系统表信息的精确读取，统计信息的正确性将直接影响到查询的执行效果，因此我们必须定期执行系统表信息的统计更新工作。此外还要经常做好数据分布工作，使数据的组织形式更为合理。通过数据分布，优化器可以根据有关信息确定如下内容：过滤器字段的选择率（Selectivity）、访问过滤器字段和表的策略、最佳的连接策略。一旦确定这些内容，查询的执行时间将会显著缩短。

进行统计更新和数据分布的唯一方法是运行SQL命令：Update Statistics……，其结果是：IDS 流览表和索引，一方面对统计信息加以编译，最终将编译好的信息存储到相应的系统表中，另一方面读取表中记录并对其进行排序以生成最好的组织形式。具体格式如下：

Update Statistics [ High | Medium | Low ] [ Distributions Only ] [ for table tabname [ (field1,field2……) ] ]

其中：

High对表中的所有记录进行排序以产生数据分布。

Medium随时从表中选取部分记录进行排序以产生数据分布。

Low仅执行统计更新，即仅修改系统表systables、sysindexes、syscolumns的内容，但不进行数据分布。

Distributions Only进行数据分布和部分统计更新工作，不更新系统表sysindexes 的内容。

为实现科学有效的统计更新和数据分布，通常应执行以下优化步骤：

（1）针对每张表运行

Update Statistics Medium for table tabname

Distributions Only

（2）针对每张表中带索引的第一个字段运行

Update Statistics High for table tabname (fieldname)

（3）对某些表中带复合索引的每一个字段运行

Update Statistics Low for table tabname

( fieldname1,fieldname2 … )

以上顺序非常重要，不能搞错。为方便运行，我们可以将以上命令按顺序写入到一个shell程序文件中，让操作系统在每日数据最少改动时间定时运行该程序。

### 2.3 使用 SQL 语句缓存（SSC,SQL Statement Cache）功能

SSC是Informix IDS2000 V9中增加的新功能，提供了共享的语句缓存，从而实现了快速的SQL调用。传统情况下，每条SQL语句运行前都要进行逻辑分析以判断语法正误，还要在共享内存中为各语句分配空间。实践表明，无论是OLTP还是DSS应用，大量运行的SQL语句具有相同的格式，通过SSC，重复的SQL语句可以单一在共享内存中存储及共享使用，这样不仅大大减少了大量语句的分析过程，使查询的速度明显加快，而且节省了大量共享内存空间，带来了其他应用效果的改善。SSC的使用方法如下。

（1）在IDS配置文件 $ ONCONFIG中定义

STMT_CACHE 1

或运行SQL命令：

onmode －e enable 以激活SSC功能。

（2）用户使用前还必须定义环境变量STMT_CACHE

export STMT_CACHE = 1

或运行SQL命令：

set statement cache on

经过以上设置后，所有的查询都将充分基于SSC进行高效处理。

### 2.4 Informix 其他性能参数优化

系统性能与磁盘、CPU、共享内存和网络相关。对磁盘调整的原则是降低读盘次数，极大化每次读盘数据量，数据分布均匀，防止瓶颈的发生。Online的磁盘空间应采用裸设备方式(raw device)，而不采用文件系统方式(cooked file)，前者比后者处理速度快得多，且可靠性

高。物理日志缓存空间应在30～50兆间即可，不必太大。设置cpu vp个数为cpu个数减1(若cpu个数为1，则cpu vp也为1)。共享内存一般是系统内存的1/3～1/4，一个cpu vp配4个LRU队列，n个 LRU队列配n个页刷新进程page_cleaner，调谐使高速缓冲区读命中率大于95％，写命中率大于85％，设定多张网卡可改进性能，网络碰撞率应在8％以内。系统核心参数按informix各版本提出的配置要求调整即可，若调整不对，在构造online时即可能不会成功。Informix－online数据服务器性能的调试往往在一定的经验值基础上动态反复调整、测试才能获得最终满意的结果。

### 2.5 使用临时数据库空间

当informix数据库执行一些操作时，将会产生临时文件或临时表，例如：建立索引或使用排序方式的连接、使用"order by"或"gruop by"语句、使用"select ...into temp..."语句创建临时表等，这些临时表或文件最好放在临时数据库空间中以提高系统性能。如果数据库系统频繁进行上面的操作，最好创建三个或三个以上大小相同临时数据库空间。可以通过$DBSPACETEMP 环境变量使不同的应用程序使用不同的临时dbspace，避免了所有操作对同一个disk的竞争。也可以通过$DBSPACETEMP 环境变量为一个应用程序指定多个临时 dbspace，这样应用程序生成的临时表可以分段存放到多个临时 dbspace中，充分利用了系统的并行存取机制，大大提高了系统性能。

要想在应用程序中合理使用临时dbspace，还要清楚应用生成的临时表和数据库是否带日志之间的关系。如果数据库不带日志而且DBSPACETEMP定义了临时dbspace，那么应用创建和生成的临时表将自动被写到临时dbspace空间中。如果数据库带日志，临时表也确省带日志，不会被自动创建到 DBSPACETEMP定义的临时dbspace中，而是被创建到数据库所在的普通dbspace中。此时不仅不能提高系统的性能，而且如果应用同时生成的临时表很多很大时，需要临时占用大量的数据库所在的普通dbspace空间，如果数据库所在的dbsapce剩余空间不充裕时，应用就会报错，不能正常运行下去。解决办法就是，在应用程序创建或生成临时表的SQL语句后附加上with no log，这样应用创建或生成的临时表就会被自动创建到临时dbspace中，同时提高了应用运行的效率。

### 2.6 索引策略的优化与原则

（1）两表中有关联，则关联字段必须建索引。如果select语句的where子句具有在一个表的一个单字段和另一个表的一个单字段之间的连接条件，则对记录数目更多的那个表的那个字段建立索引；如果一个表的几个字段和另一个表的几个字段的连接条件，则对记录数目更多的那个表的起作用的字段上建立复合索引。

（2）避免高重复率字段建索引。

（3）对同一表不要过多地建索引。

（4）建索引的字段的size尽量少，复合索引尽量少用。

（5）建聚族索引，减少索引文件碎片，以加快检索速度。

（6）where子句经常用到的字段做索引。

（7）先load data,后create index。

（8）数据量小的表是否建索引影响不大，一般不要对记录数小于200的小表建立索引，因为从使用索引得到的速度不能抵消在表上打开和检索索引文件所需的时间。

（9）建primary key。

### 2.7 创建库表方式优化

（1）数据库建库程序对每个数据表空间分配的优化。

在建表前将数据库每张表数据量大小作一估算，以便将表的第一个"extent"(物理上连续的页)空间分配尽量和估算值大小一致，下一个"extent"空间分配则根据表数据的增加量估计值来分配，这样可减少数据分配碎片和空间浪费，提高数据库系统的效率。

（2）引入表分割fragmentation，使数据在物理逻辑上分布均匀，有助于并行处理性能的提高。

（3）建表时对表的记录锁方式根据应用处理的不同区别对待。批量处理的表采用页锁(page)方式，实时交易的表采用行锁(row)方式。锁方式可以在建表时确定，也可以用alter tabname lock mode (row)和alter tabname lock mode(page)命令改变。值得注意的是通过dbimport、dbexport转移生成的表其默认锁方式是页级锁，对于实时交易且操作频繁的表应改为行级锁方式，使用oncheck －pt命令可查得表的锁方式状态。

（4）建库的日志方式：

No logging:不能进行事务处理。

buffered log：共享缓存满即刷新写入磁盘。

unbuffered log：当一个交易完成时即刷新写入磁盘。

ansi mode：只有日期格式差异，月日年形式，其他与unbuffered相同。

一般我们对实时处理系统日志方式采用unbuffered log，在进行大批量数据集中装卸时采用no logging。

另外，还需要应用程序的优化，在这里就不讨论了。

## 3 优化查询方法

根据应用的目的不同，查询归结为联机事物查询和决策支持查询。OLTP(联机事物)查询通常只检查一小部分数据，而DSS(决策支持系统)查询要检查大量的数据并进行复杂的处理。下面针对这两种查询给出不同的提高性能的方法。

### 3.1 在OLTP查询中提高查询性能的方法

（1）避免对ORDER BY后的列进行排序。在ORDER BY后的列上建立一个索引，让查询优化器使用该索引避免重新排序以节约时间。

（2）建立索引，避免顺序扫描。由于OLTP查询需要检索的行数量较少，顺序扫描是不合算的，最好在检索列上建立一个索引。

（3）避免合并连接，减小系统开销。在连接列上建立一个索引以避免合并连接。

（4）通过索引、数据分布和SET OPTIMIZATION改变查询途径。

### 3.2 在DSS查询中提高查询性能的方法

（1）删除索引，缩短扫描时间。由于扫描一个大表的大部分数据时，不读索引而直接扫描整个表反而更快，因此可删除索引，使查询优化器直接扫描从而提高效率。

（2）利用并行数据查询PDQ。通过配置ONLINE参数（PDQPRIORITY）启动PDQ。

（3）将OPTCOMPIND设置为2。该选项可使查询优化器从索引连接、杂凑连接和排序合并连接中选择出一种代价最小的途径。

（4）利用表分割，实行对大表的并行扫描。将DSS查询的表按照一定分布方案分段存储到不同的dbspace中。这样ONLINE在查询时，将进行并行扫描。

## 4 查看优化结果

查询优化器给用户提供了大量详尽的关于优化的信息，包括：

（1）连接过程中的开销估计。

（2）查询过程中表的使用顺序（即查询路径）。

（3）查询过程中用到的临时表。

（4）对每个表的访问类型，如：顺序扫描、索引扫描、哈希连接等，一名合格的系统管理员应熟知每一项信息所代表的含义，并进行反复的优化和输出比较方可制定出最佳的优化方案。为使系统提供以上信息，要求执行查询前先运行SQL命令：set explain on，查询完毕后再运行：set explain off，这样在用户当前工作目录下会生成一个包含以上信息sqlexplain.out文本文件。通过该文件内容，管理员可清楚地看到经过优化后的查询效果。

如果管理员想了解SSC的使用情况，可运行以下SQL命令：

onstat －g cac stmt

这时共享内存中每条SQL语句的命中情况将会详尽地显示出来，命中率越高，表明查询的效果越好，SSC得到了越充分的利用。

本文所列举的查询优化策略只是笔者工作经验的总结，实际上，数据库的优化是一个长期不懈、不断比较分析和调整的过程，因为数据在不断的变化中，应用在不断的发展中。系统管理员只有深入领会和掌握Informix动态服务器所提供的强大功能，正确观察和分析系统运行中提供的各种信息，充分结合实际应用特点，才能合理制定出良好的优化策略，实现快速、高效的数据查询和应用分析，同时也使硬件资源得到最充分的发挥。