Oracle数据库中Clob大字段的查询优化技术研究

聂红梅,赵建民
(浙江师范大学 信息科学与工程学院,浙江 金华 321004)

摘 要:在数据量达到几十万的数据库中查询Clob字段,其响应的时间是影响数据库应用的关键技术。为了提高Oracle数据库中Clob大字段的查询速度以满足用户的需求,介绍了数据库参数配置、数据表结构配置、全文索引等常用的优化技术对查询进行优化,接着介绍了笔者在开发一个应用系统中采用的建立一个优化JOB、分区建立表、分区建立表全文索引和Oracle的并行执行等优化技术来进行调整和优化。使查询Clob字段速度得到了明显的提高,最终满足了用户的需要。

关键词:Clob字段;Oracle数据库;优化索引;参数配置;全文检索
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2006)08-0097-03

Research of Optimum Query Technology on Clob Big Segment in Oracle Database

NIE Hong-mei, ZHAO Jian-min
(Dept. of Information Science & Engineering, Zhejiang Normal University, Jinhua 321004 ,China)

Abstract: The response time of querying the Clob field in a mass-data database is one of the key technologies influencing the database usage. This paper proposes some common optimum technologies, such as configuration of the database parameters, configuration of the table structures and full-text indexing used to promote the query speed of the Clob big field in the Oracle database. Ways of optimizing and adjusting the database applied in an application system development are also introduced in this paper, including creating an optimum JOB, full-text indexing of tables section by section and parallel executing the Oracle database. All these technologies make a significant improvement in the speed of querying the big Clob field and satisfy the user's needs.

Key words: Clob field; Oracle database; optimum indexing; parameter configuration; full-text search

Oracle的Clob是字符大对象,存放单字节字符数据,常用来存储大的文本项,如文档和Web页[1]。在笔者参与开发的系统中,让使用者可以发表计算机技术文档,以使大家能互相学习和交流。这些发表的计算机技术文档就保存在Oracle数据库中文章表里的Clob字段里。用户将会查询该Clob字段,找到符合自己要求的文章,即系统提供全文检索的功能给用户。每篇文章大小尺寸不同,如果按每篇20k计算,一天发表30篇,十年的文章记录数就是109500条记录,占用空间2.19G。所以,针对这种既需要全文检索,记录数据量又很大的情况,必须要对大字段查询进行优化,否则,查询速度将会很慢,不能满足用户的需求。文中将讨论如何优化大字段海量数据的查询技术,使查询速度达到令人满意的程度。

1 创建Clob字段及其全文索引

在创建Clob字段前,应对影响查询Clob数据的一些重要性能参数进行重新设置,这些配置对提高Clob的查询速度也是至关重要的。

1.1 修改Oracle的配置参数

在本系统中,含Clob大字段的表名字叫文章表(ARTICLES)。该文章表保存多年来各个用户发表的有关计算机技术文章。该表主要由文章唯一编号(ID)、文章标题(TITLE)、发表时间(PDATE)、作者(AUTHOR)和文章内容(CONTENT)等字段组成,其中,文章内容就是Clob大字段里。文章表至少有几十万条记录,也就是说Clob字段记录数至少也是几十万条。由于文章表具有大数据量的特性,跟其他表设置有所不同,为了不相互影响各自的性能,所以应将文章表和它的全文索引数据放在与其他数据不同的表空间。

存储文章表的表空间,需要改变它的一些相关参数来提高查询Clob字段的速度[2]。

(1)更改表空间的块尺寸。

在本系统中,Oracle数据库服务器是一台性能较好的低档服务器,4G内存,双CPU,RAID5磁盘陈列。由于文章表的索引数据量很大,所以决定将表空间的块尺寸(db_block_size=16k)设置大一些,这样索引的高度将大大减

收稿日期:2005-11-14
作者简介:聂红梅(1968-),女,四川达州人,讲师,硕士研究生,研究方向为计算机应用研究;赵建民,教授,硕士生导师,研究方向为计算机应用技术。

少,会提高IO效率。设置非标准的块大小,需要改变其相对应的Oralce数据库缓冲区,即DB_16k_CACHE_SIZE,因为它不会用DB_CACHE_SIZE设置的默认数据库缓冲区。命令如下:alter system set db_16k_cache_size = 200M;

(2)同时读取多块数据。

文章表里的一篇文章一般都在10k以上,它们分布在多个block上,为了让Oracle能同时读取大量的数据块以降低系统的I/O开销和CPU开销,需要设置Oracle的多块读取的特性。设置初始化参数DB_FILE_MULTIBLOCK_READ_COUNT,由于这个参数几乎不会导致系统性能的降低,所以把它设置高一些,如:DB_FILE_MULTIBLOCK_READ_COUNT = 30。为了充分发挥多块读取数据的优势,应当尽量配置自己的系统以使数据库的块尽可能都是连续的。后面创建表空间的时候,会使表空间的范围设置大一些,就是为了满足多个块是连续的。

(3)调整Oracle实例。

为了对Oracle实例进行有效的调整,需要对Oracle初始化参数进行认真的配置,因为这些重要参数将直接极大地影响整个系统(包括Clob字段查询)的性能。主要调整的参数:SGA参数、程序全局区和用户内存参数、undo参数、混合参数(如:log_buffer,open_cursors)等。

1.2 创建表空间

在Oracle9i中,需要将文章表及其索引块分配到非默认数据块大小的表空间。在创建一个表空间时,使用一个新的blocksize参数,创建了一个blocksize为16k的表空间[3]。

```
CREATE TABLESPACE "TEXT"
  DATAFILE
  'C:\ORACLE\ORADATA\NETCOP\TEXT.ORA' SIZE 2000M BLOCKSIZE 16KDEFAULT STORAGE (INITIAL 50M NEXT 50M MINEXTENTS 1 MAXEXTENTS UNLIMITED
  PCTINCREASE 0);
```

这里需要说明的是,没有采取本地管理(EXTENT MANAGEMENT LOCAL)的方式创建表空间,因为认为文章表比较特殊,这里采取人工管理会比本地管理要更好些。

为什么表空间存储参数INITIAL和NEXT会设置50M这么大呢?主要有两个原因:

(1)文章表的文章内容为Clob大字段,文章字节数多,设置一个大的范围可以使一篇文章尽量放在一个范围内存储,由于文章的数量也很多,因而不会浪费大的磁盘空间。

(2)保证尽可能多的连续数据块,使DB_FILE_MULTIBLOCK_READ_COUNT参数发挥最大作用。

1.3 创建包含Clob字段的表及索引

在作了前面的配置基础上,创建了一个文章表:

```
CREATE TABLE "ARTICLES" ("ID" NUMBER(10) NOT NULL, "TITLE" VARCHAR2(100) NOT NULL, "AUTHOR" VARCHAR2(40) NOT NULL, "PDATE" DATE DEFAULT SYSDATE NOT NULL, "CONTENT" CLOB, PRIMARY KEY("ID"))
```

创建基于Clob字段的全文索引:

```
CREATE INDEX CTXSYS.ARTICLES_CTX ON ARTICLES(CONTENT) INDEXTYPE IS CTXSYS.CONTEXT PARAMETERS ('LEXER WKSYS.WK_CHINESE_LEXER STOPLIST CTXSYS.EMPTY_STOPLIST')
```

1.4 查询Clob字段

在有几十万条记录的文章表(ARTICLES)里,执行如下查询语句:select title from ARTICLES where contains (content,'金华')>0 and pdate>=sysdate-365;查询今年来发表的含有"金华"字样的所有文章,并将标题显示出来。这时,发现查询速度非常慢,结果在20秒后才显示出来,这已经不能满足使用者的需要了。因此,必须探讨对查询的优化技术。

2 查询Clob字段数据和优化查询

2.1 优化索引

对表ARTICLES建立全文索引时,系统自动建立了多个对应的DR$表,如:DR$ ARTICLES_CTX $I、DR$ ARTICLES_CTX $K、DR$ ARTICLES_CTX $R等,事实上Oracle管理全文索引的本质就是管理上面这些表和相关索引。其中最重要的表为:DR$ ARTICLES_CTX $I。为了说明它工作的机理,下面是对它进行的跟踪过程[4]。

(1)首先插入ARTICLES表一条记录:

```
insert into ARTICLES values (1,"test","guest",sysdate,'厦门中国');
```

(2)统计DR$ ARTICLES_CTX $I的记录:

```
select count(*)from DR$ ARTICLES_CTX $I;
```

发现4条记录如表1所示。

表1 DR$ ARTICLES_CTX $I的记录

TOKEN_TEXT	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
厦门	1	1	1
门中	1	1	1
中国	1	1	1
国	1	1	1

之所以有4条记录,是因为:中文是按两个字组成一条记录,如果词组有重复,它会自动合并为一条。最后一条记录为一个中文字或英文词。这样可以大约算出索引的记录数为:中文+英文的个数。

(3)再插入ARTICLES表一条记录:

```
insert into ARTICLES values (1,"test1","guest1",sysdate,'厦门中国');
```

(4)统计DR$ ARTICLES_CTX $I的记录:

select count(*)from DR$ ARTICLES_CTX $I;

发现 8 条记录如表 2 所示。

表 2　DR$ ARTICLES_CTX $I 的记录

TOKEN_TEXT	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
厦门	1	1	1
门中	1	1	1
中国	1	1	1
国	1	1	1
厦门	2	2	1
门中	2	2	1
中国	2	2	1
国	2	2	1

如果继续增加记录,DR$ ARTICLES_CTX $I 表会越来越大。因此,应该优化索引,使重复的记录合并在一起,从而减少记录数。

(5)优化索引:

执行 ctx_ddl.optimize_index('ARTICLES_CTX','FAST');现在看看 DR$ ARTICLES_CTX $I 的记录:select count(*)from DR$ ARTICLES_CTX $I;发现 8 条记录减少到 4 条了。其记录情况如表 3 所示。

表 3　DR$ ARTICLES_CTX $I 的记录

TOKEN_TEXT	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
厦门	1	2	2
门中	1	2	2
中国	1	2	2
国	1	2	2

之所以出现上面的情况,其原因是:

文章内容的字词数(中文字+英文词)总和大约等于 DR$ ARTICLES_CTX $I 表(简称 I 表)的记录数。比如:一篇文章有 50 个中文字+10 个英文词=60,那么 I 表记录数=60,如果只按中文计算,那么占用的最大空间 4*60=240byte,因为 I 表保存的总是两个中文字(除了最后一个字)。现在大致计算实际文章表对应的 I 表记录数。假设有 20 万篇文章,一篇文章有 10000 个中文和英文,那么 I 表将有 10000*20*10000=2000000000 条记录,即 20 亿条记录,所以,即使做了上面的优化,查询还是很慢。

2.2　建立一个优化 JOB

为进一步优化,编制了一个 JOB 任务,定期优化索引,优化后的 I 表只有原来的 1/30 大小。

优化索引主要分 3 种:

(1) ctx_ddl.optimize_index('ARTICLES_CTX','FAST');它不合并空间,只合并重复记录,花费时间最短。

(2) ctx_ddl.optimize_index('ARTICLES_CTX','FULL');它要合并空间且合并重复记录,花费时间最长。

(3) ctx_ddl.optimize_index('ARTICLES_CTX','FULL',120);它要合并空间且合并重复记录,且最长时间限制在 120 分钟内,可以控制花费时间。

如果为了性能,建议长时间进行一次 FULL 优化,短时间定期进行 FAST 优化。

2.3　分区建立表、分区建立表全文索引

由于本系统对 Clob 字段的查询经常把发表时间限制在一段时间范围内,所以采取了分区建立表、分区建立表全文索引。分区的本质就是将一个表按某个/些字段分成多个区来存储,并可以分别索引。由于每天的数据增加很多,分区按时间字段(按年分割)分区。参考下面的查询:select title from ARTICLES where contains(content,'金华')>0 and pdate>=sysdate-365;如果按年来分区,那么又可以将整个数据量减少十几倍。这样做,Oracle 将在符合时间限制的各个索引表(I 表)里找记录,这样缩小了查询索引表的范围,速度会更快。

比如:有以下几个索引表:dr2002i,dr2003i,dr2004i,dr2005i。如果查询最后一年的记录,它只查询 dr2005i,前面的索引表将不查询,因而大大提高了查询速度。如果查询条件没有时间的限制或排序,查询速度比无分区索引要稍微长一点,因为所有的索引最后需要加在一起。

(1) 普通索引[5]:CREATE INDEX [schema.]index on [schema.]table(column) INDEXTYPE IS ctxsys.context LOCAL;或 CREATE INDEX [schema.]index on [schema.]table(column) INDEXTYPE IS ctxsys.context LOCAL [(PARTITION [partition] [PARAMETERS ('paramstring')] [, PARTITION [partition] [PARAMETERS ('paramstring')]])] [PARAMETERS (paramstring)] [PARALLEL n];

(2) 全文索引:CREATE INDEX index_name ON table_name (column_name) INDEXTYPE IS ctxsys.context PARAMETERS ('...')LOCAL

(3)同步索引:注意:分区名按最早的时间开始一个一个同步:ctx_ddl.sync_index('ARTICLES_CTX','50M','分区名');比如:ctx_ddl.sync_index('ARTICLES_CTX','50M','A_P1');ctx_ddl.sync_index('ARTICLES_CTX','50M','A_P2');ctx_ddl.sync_index('ARTICLES_CTX','50M','A_P3');依次类推。

2.4　并行查询处理

由于 Oracle 数据库服务器是一台双 CPU,RAID5 磁盘阵列的机器,所以可以利用 Oracle 的并行执行特性。并行查询处理允许多个服务器进程以并行方式处理某些 Oracle 语句。所以,用下面的查询语句代替以前的查询语句:

select /* + PARALLEL(ARTICLES,2) */ title from ARTICLES where contains(content,'金华')>0 and pdate>=sysdate-365;

这时系统将会使用两个查询服务器。这里的查询服务器数量是由 Oracle 初始化参数 PARALLEL_MAX_SERVERS 确定的。

经过上面的一系列查询优化,最后测试在文章(ARTICLES)表拥有几十万条记录下的 Clob 字段查询,发现查询所花费的时间大约只有 1 秒,查询速度足足提高了几十倍。

(下转第 102 页)

T0: User = User1, User2

T1: User1. o(requestUR1). i(ackRU1) = o(requestUR1). i(ackRU1). User1

T2: o(check1). User1, i(check). Router, NoSent = User1. o(check1), i(check). Router, Sent

T3: o(requestUR1). i(ackRU1). User1, i(requestUR). o(requestRS). i(ackSR). o(ackRU). Router = i(requestUR). o(requestRS). i(ackSR). o(ackRU). Router, o(requestRS1). i(ackSR1). o(ackRU1). Router, i(ackRU1). User1. o(requestUR1)

经过转换机制,得到了有 500 个状态左右的 LTS, T_i 就是 LTS 弧上的标号。

2.4 FSP 和 CHAM 的比较说明

FSP 在体系结构动态性描述方面,其简洁性显而易见,用过程来表示各个构件和连接器,用 FSP 语言模拟每个过程的行为,更重要的是,这里有一个开发成熟的工具 LTSA,使每个 FSP 规格说明到 LTS 状态图间的转化过程变得更简单易行。但 FSP 不提供体系结构静态特征描述,所以不可能用它进行完整的综合的体系结构规格说明。

CHAM 语言是通过把化学反应和抽象机的概念有机地结合来描述系统状态的变化,所以它能从系统操作的动态性方面来进行描述,使人们了解系统功能和行为。但从动态性规格说明转换到 LTS 的过程要比 FSP 方法复杂,从最后得到的全局 LTS 状态图中的状态数(FSP:256 个状态;CHAM:500 多个状态)也可以看出,从 CHAM 导出的 LTS 要比由 FSP 导出的 LTS 图复杂得多。但由 CHAM 导出的 LTS 更完整,上面的信息量更大,可能更能够全面测试整个系统。

另外,与 FSP 不同,CHAM 可以用分子的代数结构模拟系统静态的结构,因此能得到一个描述软件体系结构静态和动态特征的综合框架。同时 CHAM 是一种形式化表示方法,而形式化方法是提高软件质量的重要途径,在从高层规范到最终实现的过程中,选用适当的、以形式化方法为基础的工具进行辅助设计和验证对于提高系统安全性、可靠性是非常有帮助的。

3 相关问题讨论

FSP 和 CHAM 语言都能进行体系结构的动态描述,并各有各的特点,但不一定能从它们的描述中得到一个全局的体系结构模型,原因有两个:

(1)描述各个体系结构构件的模型都比较复杂,在最后合并模型的过程中可能引起状态爆炸的问题;

(2)由于一些未知的行为或未指定的构件,体系结构模型可能是不完整的。

所以,现在有人提出把 UML(统一建模语言)应用到软件体系结构的一致性测试中,用标准的 UML 表示法来模拟软件体系结构的行为,这在两方面是有用的:

a. 推动严格的基于软件体系结构测试方法在工业生产中的应用;

b. 利用基于 UML 的丰富的测试技巧。如在文献[6]中,作者就用了 UML 的状态图来指定行为的信息,为软件体系结构建模。

参考文献:

[1] Bertolino A, Corradini F, Inverardi P, et al. Deriving Test Plans from Architectural Descriptions[A]. Jazayeri M, Wolf A. Proceedings of ICSE 2000, International Conference on Software Engineering[C]. Limerik, Ireland: [s. n.], 2002. 220 - 229.

[2] Harrold M J. Architecture - Based Regression Testing of Evolving Systems[A]. Proceedings of the International Workshop on the Role of Software Architecture in Testing and Analysis (ROSATEA)[C]. Marsala, Sicily, Italy: [s. n.], 1998. 73 - 77.

[3] Harrold M J. Testing: A Roadmap[A]. Finkelstein A. The Future of Software Engineering[C]. New York: ACM Press, 2000. 61 - 72.

[4] Tretmans J. Testing Concurrent Systems: A Formal Approach [A]. In Proceedings of the 10th International Conference on Concurrency Theory (CONCUR '99)[C]. London, UK: Springer - Verlag, 1999. 46 - 65.

[5] Compare D, Inverardi P, Wolf A L. Uncovering Architectural Mismatch in Component Behavior[R]. Technical Report CU - CS - 828 - 97, 1997.

[6] Medvidovic N, Rosenblum D S, Redmiles D F, et al. Modeling software architectures in the Unified Modeling Language[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2002, 11(1): 13 - 15.

(上接第 99 页)

3 结束语

在低端服务器上,通过对 Oracle 数据库的初始参数配置进行调整并优化相关参数,如:块大小、多块连续读取、SGA 等。然后对数据表结构进行分区和优化索引,并优化查询 SQL 语句,使查询 Clob 字段速度得到了极为明显的提高,最终满足了用户的需要。

参考文献:

[1] 程春玲,李玲娟,周宁宁. Oracle 的大对象(LOBs)及其在企业 MIS 中的应用[J]. 微机发展,2002,12(6):28 - 29.

[2] 张燕平,王向阳,张力平. ORACLE 数据库的优化问题[J]. 微机发展,1998,8(6):46 - 47.

[3] 谈竹贤,王 毅,赵景亮. Oracle 9i PL/SQL 从入门到精通[M]. 北京:中国水利水电出版社,2002.

[4] Whalen E, Schroeter M. Oracle 性能调整和优化[M]. 高艳春,周 确,唐艳军译. 北京:人民邮电出版社,2002.

[5] Casteel J. Oracle 9i 开发指南:PL/SQL 程序设计[M]. 天宏工作室译. 北京:清华大学出版社,2004.