

基于 Oracle 数据库海量数据的查询优化研究

张 辉,赵郁亮,徐 江,孙伟华

(公安部第三研究所 防伪事业部,上海 201204)

摘 要:系统数据库数据量大时,常常会遇到系统响应时间过长、占用系统资源过多等一系列问题。因此对海量数据查询技术的研究就很必要。文中基于 Oracle 数据库,采用各种措施和方法来提高响应速度:合理设计数据库结构、使用索引、分区、SQL 语句优化等手段优化数据库。另外,分页显示查询结果也能很大地提高查询速度。实验证明,优化后速度明显提高了很多。因此可见,文中采用的优化方法对海量数据的查询优化是正确有效的。

关键词:Oracle 数据库;海量数据;查询优化

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2012)02-0165-03

Query Optimization Research on Mass of Data Based on Oracle Database

ZHANG Hui,ZHAO Yu-liang,XU Jiang,SUN Wei-hua

(Anti-counterfeiting Technology Cause Department, The Third Research Institute of Public Security

Ministry of the P. R. C, Shanghai 201204, China)

Abstract: While database has large amount of data, the inquires will have a series of problems like system response time is too long and take up too many system resources. So the query optimization research on the mass of data based on Oracle database is necessary. It uses the various measures and methods to improve the response speed based on Oracle database such as reasonably designing the database structure, the use of the index, partition, SQL optimization method. In addition, pagination display the query results can also greatly improve query speed. Experiments show the optimization obviously improves the speed a lot.

Key words: Oracle database; the mass of data; query optimization

1 概 述

在数据库数据量比较大时,查询的响应时间过长、占用资源过多是一个很大的问题。因此对海量数据查询技术的研究^[1,2]就很有必要。

如果想要设计一个性能良好的 Oracle 数据库方案,首先要对 Oracle 数据库表结构进行合理的设计,如果数据库本身设计就不是很合理,那么对 Oracle 数据库的查询优化就会大打折扣,因此,Oracle 数据库的查询优化应该从合理设计数据库结构开始,数据库设计分为逻辑设计和物理设计两个阶段。在设计时应遵循基于第三范式的基本表进行设计,同时也要基于基本表进行扩展设计,这里不再赘述。

对系统硬件的升级也是提高查询速度的一种很重要的方法,比如更换或升级服务器的版本,但文中对查询优化技术的研究主要是针对数据库的逻辑层^[3,4]进

行研究的。数据库以 Oracle 为例,主要采取以下措施来提高查询速度:使用索引、分区、SQL 语句优化等手段对数据库的逻辑结构进行优化。

2 索引技术

索引^[5]是建立在表列上的,当某列创建了索引之后,每次进行插入、修改和删除操作之后,该表数据就会根据索引进行重新排序,以方便针对该列针对索引进行查询。因此,在创建索引的时候,要把索引列建立在经常进行查询操作而不经常进行增加、删除、修改操作的列上。创建索引时还需要注意:

(1)建立索引要付出维护的代价,建立以后却从未使用过;

(2)为了返回一条单一记录而扫描多个文件;

(3)由于存在错误的索引,导致多表的连接操作持续几个小时。

3 分区技术

Oracle 的分区^[6,7](Partitioning Option)是针对具

收稿日期:2011-06-28;修回日期:2011-10-16

基金项目:国家十一五科技支撑计划项目(2008BAF31B01)

作者简介:张 辉(1981-),女,助理研究员,硕士,研究方向为智能交通、RFID 技术、数据库研究。

有海量数据的大型表进行处理的一种技术,是将一个大表的数据从物理上分离,但从逻辑结构上仍然属于一个整体,看不出有什么变化,这样就减少了技术人员对具体数据存储位置的了解,只要从逻辑上仍将该大表看成一个整体就行。增加了可用性。

3.1 分区的方法

Oracle 分区主要有:按字段值范围分区;按字段的 HASH 函数值 HASH 分区;先按范围划分,再按 HASH 划分的复合分区;在 Oracle9i 中又增强了按字段值列表进行划分的列表(Listing)分区方法。

在本次试验中,针对车辆过车记录表中的过车时间字段进行分区,每个月的数据做一个分区表,这样按照过车时间进行查询时,会大大增加查询的速度。

本次分区表的 SPL 语句如下:

```
partition by range (TRV_ARRTIME)
(partition p_200911 values less than (to_date('2009-11-1',
YYYY-MM-DD')),
partition p_200912 values less than (to_date('2009-12-1',
YYYY-MM-DD')),
partition p_201001 values less than (to_date('2010-1-1',
YYYY-MM-DD')),
partition p_201002 values less than (to_date('2010-2-1',
YYYY-MM-DD')),
partition p_201003 values less than (to_date('2010-3-1',
YYYY-MM-DD')),
.....
partition p_201102 values less than (to_date('2011-2-1',
YYYY-MM-DD')),
partition p_201103 values less than (to_date('2011-3-1',
YYYY-MM-DD')),
partition p_201104 values less than (to_date('2011-4-1',
YYYY-MM-DD')),
partition p_201105 values less than (to_date('2011-5-1',
YYYY-MM-DD')),
partition p_201106 values less than (to_date('2011-6-1',
YYYY-MM-DD')),
partition p_max values less than (MAXVALUE));
```

3.2 分区的优点

通过对表进行分区,可以获得如下好处:

(1)增强可用性:如果表的某个分区出现故障,表在其他分区的数据仍然可用。

(2)维护方便:如果表的某个分区出现故障,需要修复数据,只修复该分区即可。

(3)均衡 I/O:可以把不同的分区映射到磁盘以平衡 I/O,改善整个系统性能。

(4)改善查询性能:对分区对象的查询可以仅搜索自己关心的分区,提高检索速度。

分区表事务操作的时候,指定了分区,系统在执行

的时候则只操作该分区的记录,提高了数据处理的速度。不要指定分区直接操作数据也是可以的。在分区表上建索引及多索引的使用和非分区表一样。此外,因为在维护分区的时候可能对分区的索引会产生一定的影响,可能需要在维护之后重建索引。

4 SQL 语句优化技术

4.1 SQL 语句优化

在数据库应用系统中编写可执行的 SQL 语句可以有多种方式实现,但用哪一条比较适合却难以确定。为了解决这一问题,有必要对 SQL 实施优化。简单地说,SQL 语句的优化^[8-10]就是将执行效率较低的 SQL 语句转换成目的相同但执行效率较高的 SQL 语句。

SQL 语句优化主要是进行查询等价转换,主要是从查询语义的角度将用户的 SQL 查询语句转换为效率更高的形式。SQL 语句优化的原则如下:

- 尽量使用索引
- 选择联合查询的次序
- 在子查询中慎重使用 IN 或者 NOT IN 语句,使用 where (NOT) exists 的效果要好的多
- 慎重使用视图的联合查询,尤其是比较复杂的视图之间的联合查询
- 避免在 SQL 里使用 PL/SQL 功能调用
- 限制对远程表的访问
- 在查询中慎用 IN 或 NOT IN 子句
- 用 UNION 替换 OR (使用于在 where 子句中包含索引列)
- 优化 GROUP BY (将不需要的记录在 GROUP BY 之前过滤掉)

Sql 语句优化的原则有很多,这里只是强调此优化技术,对具体的优化原则不再一一赘述。

4.2 分页查询

在系统进行查询功能的实现时,可使用分页查询^[11,12]来提高查询的速度。分页查询的原理就是通过 SQL 语句过滤掉大部分数据,只查询出 n (一般 n 小于等于 20) 条数据,如果需要查看后 n 条数据,可点击“下一页”按钮,查询后 n 条数据,而不是一次查询出成千上万条数据。

5 结束语

针对文中的研究内容,专门做了一个测试软件测试优化结果,测试数据将近百万条。在未做优化前,在同样的查询条件下,查询时间约为 4 到 5 秒,在经过文中所述的优化之后,查询速度缩减为几十毫秒,查询速度提高了几十倍。

表 1 是进行索引测试的测试结果,表 1 用到的表

的数据量是十几万条数据。

表1 “索引优化”实例的查询数据表

查询条件	>20 岁	>30 岁	>40 岁	>50 岁
使用索引	17.2 毫秒	13.67 毫秒	9.62 毫秒	6.75 毫秒
未用索引	17.56 毫秒	14.45 毫秒	10.43 毫秒	7.61 毫秒

图1是经过各种优化手段后输入车牌号码、起始经过时间和结束经过时间后的查询时间。图1用到的表的数据量是二十万条左右。

1354	NJB004	2009-9-17 0:00:00	苏A76022
2285	NJB004	2009-9-19 0:00:00	苏A76022
当前第1页/共10页 下一页			
运行时间:		28	毫秒

图1 车辆过车记录查询时间

文中着重从数据库层和应用层方面对查询优化技术进行了详细的讲解,如使用索引技术、分区技术、SQL语句优化技术和分页查询等等。另外,还可通过升级硬件的方法来提高查询速度。

参考文献:

- [1] Willis D, Pearce D J, Noble J. Efficient Object Querying in Java[C]//Proceedings of the European Conference on Object-Oriented Programming (ECOOP). [s. l.]:[s. n.], 2006.
- [2] Lee Hyunho, Lee Wonsuk. Query Optimization for Web BBS by Analytic Function and Function-Based Index in Oracle DBMS[C]//Proceedings of Sixth International Conference on Advanced Language Processing and Web Information Technology. SoutheastCon: IEEE, 2007.
- [3] Gibbons P B, Matias Y, Poosala V. Fast Incremental Maintenance of Approximate Histograms[J]. ACM Transactions on Database Systems, 2002, 27(3): 261-298.
- [4] Conn S S. OLTP and OLAP data integration: a review of feasible implementation methods and architectures for real time data analysis[C]//Proceedings of IEEE. SoutheastCon: [s. n.], 2005.
- [5] 袁爱梅. ORACLE 数据库性能优化研究[D]. 上海: 华东师范大学, 2007.
- [6] 周彦, 陈梅, 王翰虎, 等. 基于层次位图连接索引的数据仓库查询优化[J]. 计算机技术与发展, 2011, 21(3): 41-43.
- [7] 刘博. ORACLE 数据库性能调整与优化[D]. 大连: 大连理工大学, 2007.
- [8] 许华容. Oracle 数据查询优化方法研究[D]. 贵阳: 贵州大学, 2008.
- [9] 王君, 祝永志, 魏榕晖, 等. 基于 Oracle 分布式数据库的查询优化[J]. 计算机技术与发展, 2008, 18(1): 157-160.
- [10] 杨小艳, 尹明, 戴学丰. Oracle 数据库查询优化方法研究[J]. 计算机与现代化, 2008(4): 4-7.
- [11] 周志德. Oracle 数据库的 sql 查询优化研究[J]. 计算机与数字工程, 2010(11): 173-178.
- [12] 邓春娜, 周晓红. Oracle 数据库的查询优化方案[J]. 信息科技, 2010(5): 19-20.
- [13] (上接第164页)
- [10] 陈凤珍, 洪帆. 基于任务的访问控制(TBAC)模型[J]. 小型微型计算机系统, 2003, 24(3): 621-624.
- [11] Joshi B D, Bertino E, Latif U, et al. A generalized temporal role-based access control model[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(1): 4-23.
- [12] 陈泉冰, 王会进. 一种改进的基于任务-角色的访问控制模型[J]. 暨南大学学报(自然科学版), 2010, 31(1): 29-34.
- [13] Damiani M L, Bertino E, Perlasca P. GEO-RBAC: a spatially aware RBAC[C]//Proceedings of the 10th ACM Symposium on Access Control Models and Technologies. Stockholm, Sweden: [s. n.], 2005: 29-37.
- [14] Bertino E, Kirkpatrick M. Location-aware authentication and access control-concepts and issues[C]//Proceedings of the 23rd International Conference on Advanced Information Networking and Applications Workshops. Bradford, England: [s. n.], 2009: 10-15.
- [15] Barka E, Sandhu R. A role-based delegation model and some extensions[C]//Proceedings of the 23rd National Information Systems Security Conference. Baltimore, USA: [s. n.], 2000.
- [16] Zhang L H, Ahn G J, Chu B T. A rule-based framework for role based delegation[C]//Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies. Chantilly, VA, USA: [s. n.], 2001: 3-9.
- [17] Zhang X W, Oh S, Sandhu R. PBDM: a flexible delegation model in RBAC[C]//Proceedings of the eighth ACM symposium on access control models and technologies. Como, Italy: [s. n.], 2003: 149-157.
- [18] Crampton J, Khambhammettu H. Delegation in role-based access control[J]. International Journal of Information Security, 2007, 7(2): 123-136.
- [19] 廖俊国, 洪帆, 朱更明, 等. 基于信任度的授权委托模型[J]. 计算机学报, 2006, 29(8): 1265-1270.
- [20] 徐震, 李澜, 冯登国. 基于角色的受限委托模型[J]. 软件学报, 2005, 16(5): 970-978.
- [21] 翟征德. 基于量化角色的可控委托模型[J]. 计算机学报, 2006, 29(8): 1401-1407.
- [22] 翟征德, 冯登国, 徐震. 细粒度的基于信任度的可控委托授权模型[J]. 软件学报, 2007, 18(8): 2002-2015.
- [23] Schaad A, Moffett J, Jacob J. The role-based access control system of a european bank: a case study and discussion[C]//Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies. Chantilly, VA, USA: [s. n.], 2001: 3-9.