TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.7, July 2014, pp. 5693 ~ 5698
DOI: 10.11591/telkomnika.v12i7.5050

Partitioning of Oracle Application in Structured Electronic Medical Records

Chuandi Pan, Jiandong Hu*, Jian Shi
Department of Computer and Information Management, the First Affiliated Hospital of Wenzhou Medical University,
Wenzhou, 325005, Zhejiang Province, The People's Republic of China
*Corresponding author, e-mail: hjd1056@163.com

Abstract

This paper described how to use Oracle partitioning to achieve classified storage of data from structured electronic medical records. Because structured electronic medical records stored massive data, the performance, security and stability of database had been paid more and more attentions. In this paper, table with partition in structured electronic medical records system was presented to solve these problems. In addition, testing of comparing the performance of table with and without partition had been taken to prove it was much faster to retrieve partitioned tables with massive data than not partitioned tables.

Keywords*: oracle, partition tables, structured electronic medical records, VLDB*

1. Introduction

Electronic medical records is focused on outpatient and hospitalized patients (or objects), it is the integration system of clinical diagnosis and treatment of information data [1]. Structured electronic medical records system applies structured method to analyze medical documents which are input in natural language from medical information point of view, and saves those semantic structures in the way of relational structure to database [2]. Structured electronic medical records system is regarded as the future of electronic medical records system, because it has many merits. Choosing appropriate data storage is vital to function, performance and security of structured electronic medical records system, because of too many data elements, complicated structure and huge data size. Through partitioning related tables, retrieving efficiency will be improved and database will be more stable [3]. This paper describes how to use Oracle partitioning to achieve classified storage of data from structured electronic medical records.

2. Oracle Partitioning

Partitioning is provided by Oracle to support VLDB (Very Large Database).a partitioned table in Oracle contains many partitions which can reside in different table spaces. Each partition is an individual segment [4]. Partitioning is entirely transparent to applications. Retrieving data not only can be implemented by querying table with all partitions, but also can be achieved by querying certain partitions of table. There are two suggests about 'When to partition a table' from Oracle:

(1) Tables greater than 2GB should always be considered for partitioning.

(2) Tables containing historical data, in which new data is added into the newest partition. A typical example is a historical table where only the current month's data is updatable and the other 11 months are read only.

While creating partition, partition key should be chose. Oracle automatically directs insert, update, and delete operations to maps data to the appropriate partition through the use of the partition columns. Oracle provides the following partitioning methods:

(1) Range Partitioning: Range partitioning maps data to partitions based on ranges of partition key values that you establish for each partition.

Received November 4, 2013; Revised March 25, 2014; Accepted April 15, 2014

(2) List Partitioning: List partitioning enables you to explicitly control how rows map to partitions by specifying a list of discrete values for the partitioning key in the description for each partition.

(3) Hash Partitioning: Hash Partitioning maps data by hash value of partition key. You can improve I/O performance by Hash Partitioning, because Hash Partitioning maps data equally to certain partitions.

(4) Composite Partitioning: Composite Partitioning, for example Range-Hash Partitioning or Range-List Partitioning, allows you sub partition a table by create partition in an exist partition, when the table is considered to huge after partitioning or when there are other requirements.

3. Conceptual Data Model of Structured Electronic Medical Records System

Structured electronic medical records system in our hospital uses dynamic analysis of data structure. The concept data mode is shown in Figure 1.

Table named medical_record_index stores indexes of medical records. Table named medical_record_element_definition stores element definitions of medical record. Table named medical_record_data stores element data of each medical record. Table named modify_trace stores trace of modification of data of each medical record. Table named specialty_format stores medical record formats of each specialty. Table named format_definition stores definitions of every formats. Table named format_style stores styles of each format. Table named style_definition stores definition of related styles.

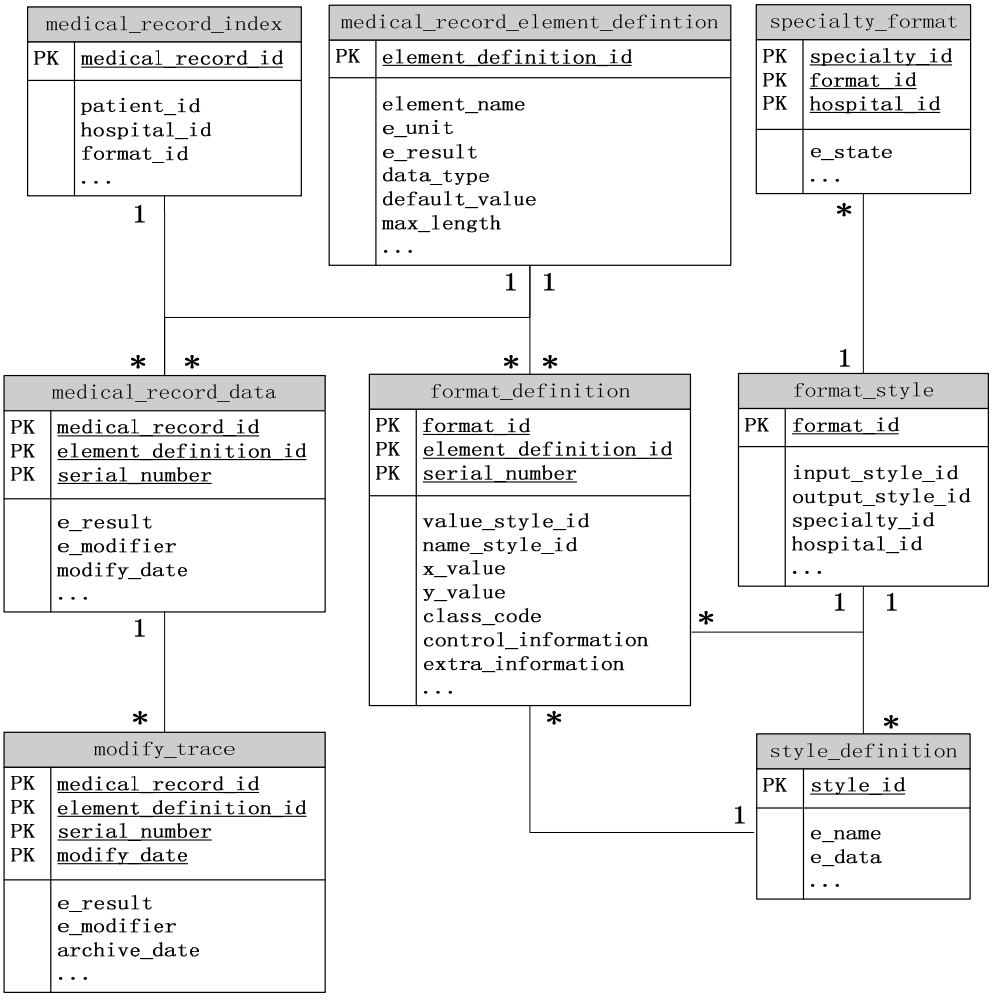

Figure 1. Concept Data Mode of Structured Electronic Medical Records System

4. Accomplishment of Using Partition to Store Data From Structured Electronic Medical Records System

We use partition table to store data of medical_record_data table and modify_trace table, because these tables are extremely huge in structured electronic medical records system based on dynamic analysis of data structure [5]. Realization of this storage method:

Firstly, we create several table spaces, like ts_emr1, ts_emr2, ts_emr3, ts_emr4, etc, which of data files are stored in different physical disks. (Certainly, these data files can be stored in the same physical disk, if I/O performance is not considered.) Statements of creating table space are as below:

create tablespace ts_emr1 datafile 'D:\oradata\ts_emr1\data01.dbf' size 100M autoextend on next 100M maxsize 30000M;

Remark: name of table space: ts_emr1. Data file: D:\oradata\ts_emr1\data01.dbf. Store value of initial: 100M. Auto extension: allow. Next extension: 100M. Store value of maximum: 30000M.

Secondly, we create anther table space for storing data of index.

create tablespace ts_index datafile 'D:\oradata\ts_index\data01.dbf' size 100M autoextend on next 100M maxsize 30000M;

There are statements of creating partition table of medical_record_data by range of medical_record_id.

```
create table medical_record_data (
medical_record_id number(10) not null,
element_definition_id number(8) not null,
serial_number number(8) not null,
e_result VARCHAR2(200),
e_modifier VARCHAR2(20),
modify_date DATE,
primary key(medical_record_id)
using index tablespace TS_INDEX )
pctfree 10 pctused 90
partition by range(medical_record_id)
(partition medical_record_data_p1 values less than(50000) tablespace ts_emr1,
partition medical_record_data_p2 values less than(100000) tablespace ts_emr2,
partition medical_record_data_p3 values less than(150000) tablespace ts_emr3,
partition medical_record_data_p4 values less than(200000) tablespace ts_emr4);
```

PCTFREE: Specify a whole number representing the percentage of space in each data block of the database object reserved for future updates to rows of the object [6]. While the percentage of space in data block reaches the value of PCTFREE, data block which is removed from FREE LIST is not allowed to insert. If the value of PCTFREE is too large, data block is not taken full advantage of. Otherwise, if the value of PCTFREE is so little, new records are forced to move to new data block, and performance will be degraded.

PCTUSED: Specify a whole number representing the minimum percentage of used space that Oracle maintains for each data block of the database object [6]. Only when used space of data block falls below PCTUSED, data block will be re-linked to available list and will be allowed to insert data. If the value of PCTUSED is too large, frequent re-using data block leads to increasing I/O consumption. If the value of PCTUSED is so little, this works to the disadvantage of re-using data block. However, little PCTUSED value helps to reduce I/O consumption and improve performance.

Choosing proper PCTFREE value and PCTUSED value sum of which must equal to pr less than 100 is vitally important. In our system, we set PCTFREE value as 10 and set PCTUSED value as 90.

Table medical_record_data_p1 stores data which medical_record_id is less than 50000 and which is stored in table space of ts_emr1. Table medical_record_data_p2 stores data which medical_record_id is equal to or larger than 50000 and less than 100000 and which is stored in table space of ts_emr2, and so on.

Enlarging table space or achieving cross disk storage can be realized by add data file to table space, for example ts_emr1, ts_emr2, ts_emr3, ts_emr4.

alter tablespace ts_emr1 add datafile 'E:\oradata\ts_emr1\data02.dbf' size 100M autoextend on next 100M maxsize 20000M;

After creating table space which will be used to store new partition, we can add a partition to partitioned table. For an instance, we add a partition which will be used to store data which medical_record_id is equal to or larger than 200000 and less than 250000 and which is stored in table space of ts_emr5 to table medical_record_data. The statement is shown as below:

alter table medical_record_data add partition medical_record_data_p5 values less than(250000) tablespace ts_emr5;

There are statements of creating partition table of modify_trace by range of archive_date.

```
create table modify_trace (
 medical_record_id number(10) not null,
 element_definition_id number(8) not null,
 e_result VARCHAR2(200),
 modifier VARCHAR2(10),
 modify_date DATE,
 archive_date DATE not null,
 primary key(medical_record_id, element_definition_id, modify_date)
 using index tablespace TS_INDEX )
 pctfree 0 pctused 99
 partition by range(archive_date)
 (partition modify_trace_p1 values less than(to_date('2014-01-01', 'yyyy-mm-dd'))
tablespace ts_emr101,
 partition modify_trace_p2 values less than(to_date('2015-01-01', 'yyyy-mm-dd'))
tablespace ts_emr102,
 partition modify_trace_p3 values less than(to_date('2016-01-01', 'yyyy-mm-dd'))
tablespace ts_emr103,
 partition modify_trace_p0 values less than (maxvalue) tablespace ts_emr199);
```

Structured electronic medical records system based on SaaS [7] can be shared to several hospitals through by storing medical record indexes of different hospitals into different list partitions. There are statements of creating list partition table of medical_record_index by list of hospital_id.

```
create table medical_record_index (
 medical_record_id number(10) not null,
 patient_id number(10) not null,
 hospital_id number(5),
 format_id number(5),
 primary key(medical_record_id)
 using index tablespace TS_INDEX )
 pctfree 20 pctused 80
 partition by list(hospital_id)
 (partition medical_record_index_p1 values(1) tablespace ts_emr201,
 partition medical_record_index_p2 values(2) tablespace ts_emr202,
 partition medical_record_index_p3 values(3) tablespace ts_emr203,
 partition medical_record_index_p0 values(default) tablespace ts_emr299);
```

5, Results and Discussion

We take table modify_trace for an example to compare the performance of not partitioned table and partitioned table to prove that table with partition will improve performance. Firstly, two tables named modify_trace_non without partition and modify_trace which is partitioned by rang of archive_date are created. There are the statements of creating modify_trace by range of archive_date.

```
create table modify_trace (
medical_record_id number(10) not null,
element_definition_id number(8) not null,
e_result VARCHAR2(200),
modifier VARCHAR2(10),
modify_date DATE,
```

```
archive_date DATE not null,
primary key(medical_record_id, element_definition_id, modify_date)
using index tablespace TS_INDEX )
pctfree 0 pctused 99
partition by range(archive_date)
(partition modify_trace_p1 values less than(to_date('2009-01-01', 'yyyy-mm-dd'))
tablespace ts_emr101,
partition modify_trace_p2 values less than(to_date('2010-01-01', 'yyyy-mm-dd'))
tablespace ts_emr102,
partition modify_trace_p3 values less than(to_date('2011-01-01', 'yyyy-mm-dd'))
tablespace ts_emr103,
partition modify_trace_p4 values less than(to_date('2012-01-01', 'yyyy-mm-dd'))
tablespace ts_emr104,
partition modify_trace_p5 values less than(to_date('2013-01-01', 'yyyy-mm-dd'))
tablespace ts_emr105,
partition modify_trace_p6 values less than (maxvalue) tablespace ts_emr106);
```

Then, after respectively inserting same massive data which contains 10000000 rows archive_date ranging from 01/01/2008 to 31/12/2013 into both tables, we execute query statements for these two tables.

```
select * from modify_trace_non where archive_date between
to_date('20100716','yyyymmdd') and to_date('20100717','yyyymmdd') order by archive_date;
select * from modify_trace where archive_date between
to_date('20100716','yyyymmdd')and to_date('20100717','yyyymmdd') order by archive_date;
```

We obtain statistics from executing these statements:

(1) The execution plan of table modify_trace_non is “TABLE ACCESS FULL”, Cost number is 17833 and Bytes are 190K, as shown in Table 1. More visible result is that it takes 3.813 seconds to retrieve 5539 rows of data.

Table 1. Execution Plan of Table modify_trace_non

ID	Operation	name	Rows	Bytes	Cost(%CPU)
0	SELECT STATEMENT		4641	190K	17833(2)
1	SORT ORBER BY		4641	190K	17833(2)
2	TABLE ACCESS FULL	MODIFY_TRACE_NON	4641	190K	17832(1)

(2) The execution plan of table modify_trace is “PARTITION RANGE SINGLE” and then “TABLE ACCESS FULL”. Because partition start and partition stop is both 3, it only need to scan partition 3 named ts_emr103. As a result, it only takes 2.985 seconds to retrieve 5539 rows much faster than the result of executing the previous statement and the Cost number and Bytes are respectively reduced to 2492 and 157K, as shown in Table2. Each partition is a small part of the whole data table, so retrieving a partition is bound to improve retrieval performance.

Table 2. Execution Plan of Table modify_trace

ID	Operation	name	Rows	Bytes	Cost(%CPU)	Pstart	Pstop
0	SELECT STATEMENT		3935	157K	2492(2)		
1	PARTITON RANGE SINGLE		3935	157K	2492(2)	3	3
2	SORT ORBER BY		3935	157K	2492(2)		
3	TABLE ACCESS FULL	MODIFY_TRACE	3935	157K	2491(2)	3	3

As a result, the Oracle range partitioning technology can optimize the huge amounts of data table query performance [8].

6, Conclusion

There are many advantages of using partitioning of Oracle to store data from structured electronic medical records system. (1) Improving performance: Operations of insert, delete,

update and query are distributed to destination partition, so performance is improved effectively. This is the most important advantage for structured electronic medical records system. (2) Convenient maintenance: We can maintain certain partition, for example, data repair. (3) I/O equilibrium: I/O equilibrium is achieved by storing different partitions to different disks. (4) Increasing usability: If certain partition of table fails, other partitions of table still work well. (5) Convenient backup and recovery: backup and recovery can be done to certain partition. (6) Partitioning is transparent to users. Partitioning has more obvious advantages to structured electronic medical records system based on SaaS.

References

[1] Di PC. Design of Electronic Medical Record System Based on Cloud Computing Technology. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(5): 4010-4017.
[2] Cai Shaohua. Design and implementation of electronic medical records. *Modern Computer*. 2006; 7: 110-112
[3] Yang Y, Ye H, Fei S. *Design of partitioned table for VLDB in the vehicle monitoring system.* Multimedia Technology (ICMT). International Conference on. IEEE. 2011;5377-5379.
[4] He Ping, Yang Shuqiang, Jia Yan. Backup and recovery of VLDB based on Oracle 10g. *Computer engineering*. 2006; 32(19): 79-81.
[5] Pan Chuandi, Zhou Xinchao, Shi Jian. Design of electronic medical record system based on dynamic analysis of data structure*. China Digital Medicine.* 2007; 2(3): 20.
[6] Fiorillo C. Oracle Database 11gR2 Performance Tuning Cookbook: Over 80 Recipes to Help Beginners Achieve Better Performance from Oracle Database Applications. Packt Publishing Ltd. 2012.
[7] Buxmann P, Hess T, Lehmann S. Software as a Service. *Wirtschaftsinformatik*. 2008; 50(6): 500-503.
[8] Wang P, Tian A, Guo C. Table Partitioning Technology Based on Massive Data. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(3): 1676-1686.