Journal of Computational Science & Engineering 27 (2016) 775-778

Available at http://www.asocse.org ISSN 1710-4068

Optimized Research of Batch Query for Detailed Call Records

Menglin Liu[1], jianguang Liu[2]

[1]College of Information Science and Engineering Zaozhuang University Zaozhuang, Shandong, China

[2]Zaozhuang Unicom Company, Zaozhuang, Shandong, China

Abstract: **For Telecom operators, the query for detailed call records is an important window to provide services for the outside world, which can provide the service for users or public, inspection, law and other units. For Shandong Unicom Corporation， due to the current query system of detailed call records can only achieve the query of a single number or batch numbers in a single account period, but it can't achieve the query such as batch number in a multi-account period. In order to solve this problem, this paper proposes an optimization plan of batch query for detailed call records and gives the detailed implementation process, and batch inquiry for batch numbers in the multi-billing periods is achieved. Practice has proved that this optimization has achieved the batch query for detailed call records within a multi-account period ,and the implementation process is very simple, efficient, and the query results is fairly accurate, so it has a certain value of application and academic.**

Keywords: Detailed call records; Batch query; Optimization; Stored procedure

Received 5 October 2016, Revised 23 November 2016, Accepted 25 November 2016

*Corresponding Author: Menglin Liu, 9070826@qq.com

1. Introduction

In daily life, people often interconnected via telephone or SMS. Data generated by these ways will be stored to the appropriate database in real-time. When the public security organs, procuratorate, court and other units or individuals (hereinafter referred to as users) need detailed call records about some specified phone within certain stage to handle business, each telecommunications company should extract data from these databases for them accurately and quickly, therefore, for the telecommunications companies, the query for detailed call records is an important window to provide services for the outside world. To this window, Quality of Service will directly affect the company's image. for example, In Shandong Unicom Corporation (Hereinafter referred to as the Company), information service center often need to provide detailed call records about voice and SMS of some specific Unicom mobile phone number or landline number within certain stage for the public security organs, procuratorate, court and other units or individuals. Although the current query system of detailed call records is running well during single number is queried, but it has the following problems: query efficiency is low, query results are stored cumbersome, and running process is influenced by parallel transactions. These problems have affected the service quality of Unicom seriously.

In addition, the company usually carry on investigation for the user's communication habits and call size, Bulk detailed list has become the main research material. So batch detail query is the main means of the company's traffic research, and optimized research of batch query for detailed call records is imperative.

In order to solve the question of batch query for detailed call records in multiple accounts, in this paper, on the basis of full investigation, use the SQLyog client to create the middle table and stored procedures in the call list database, according to the batch queries, call statements are generated. Then using these call statements to call stored procedure, the query results are stored in the middle of the table, then the query results are extracted from the middle table, So the batch query are finished .

2. The Status about Current Query System of Detailed Call Records

2.1. Low Query Efficiency

The function about Current query system of detailed call records is relatively simple. This system only can query a single number or batch numbers in a single account period at once and the query is not efficient. If you want to query a number of call details within a year, you will query 12 times, each query needs about three minutes, these queers will take about 3 * 12 = 36 minutes. And so on, if you want to query for batch numbers within many account periods, so consuming long can imagine.

2.2. Query Results Are Stored Cumbersome

Using the Current query system of detailed call records, the query results are stored cumbersome. For each number within each account period, each query result needs to be preserved by oneself. If you want to query detailed call records for six numbers in 12 months, you will have to produce 6 * 12 = 72 files, so many files to be stored will cause storage cumbersome, efficiency decreases and poor maintenance. And so on, if you want to query for batch numbers within many account periods, the stored query results will be great. The workload to store query results will be enormous.

2016 © The American Computational Science Society. All rights reserved.

2.3. Running Process Is Influenced by Parallel Transactions

Because the current query system of detailed call records is a small part function of ESS and the parallel transaction is more in ESS, so its speed is affected by other parallel transactions greatly. If there is a large amount of parallel transactions, the running speed of the current query system will become very slow.

3. The Optimization Plan of Batch Query for Detailed Call Records

In order to achieve batch query for batch number in the multi-billing period and to meet the fast, efficient and accurate call detail inquiry needs, the panel members have put forward the following optimization program by analyzing the problems of current query system of detailed call records:

The overall program: make full use of the historical detailed database, select and install Mysql's client: SQLyog which can running fast, create the intermediate tables and stored procedures in this database, call to these stored procedures, quickly and accurately generate and extract query results.

Firstly: According to the needs, five intermediate tables are designed using to store data: query requirements data that is user-submitted, mobile voice data, and mobile SMS data. Fixed telephone voice data and batch query generation data, which can solve the problem that query results are stored too many.

Secondly: Three stored procedures are designed using to check out the detailed call records from the database, and the data are inserted into the following tables: one table to store the mobile voice data, one table to store the mobile SMS data and one table to store fixed telephone voice data. Using Java Web technology, graphical user interface is designed and is used to Export query results, obtain the final detailed call records, which can improve the query efficiency.

thirdly: Because SQLyog runs fast, and it is affected by other systems, so take advantage of SQLyog to solve the problem on the local machine by other concurrent transactions influence, and improve the operating speed.

4. Optimization Process of Batch Query for Detailed Call Records

4.1. Create Five Intermediate Tables

One table is used to store demand. The structure of this table is following:

zz_haoma (haoma varchar2(20) , type_cx varchar2(1),

start_yymm date , ds_mounth number , cxr varchar2(20));

This is the detailed query requirements table, which is used to store the numbers of batch query, query type, start date and person's name who makes a query request.

The other one table is used to store the batch call statements.

The structure of this table is following:

zz(v_v varchar2(1000), s_t varchar2(2));

The other three tables will be used to store mobile voice data, fixed telephone voice data and mobile SMS data.

Among them:

sd_zaozhuang.ucr_zz_tg_cdr10: used to store the bulk of the query to the mobile voice detail data;

sd_zaozhuang. ucr_zza_tg_sm: used to store the bulk of the query to the detailed sms data;

sd_zaozhuang. ucr_zza_tg_cdr10_fix: used to store the bulk query to the fixed-line voice detailed data;

The three table's structure is similar.

Take sd_zaozhuang.ucr_zz_tg_cdr10 as an example, part of structure is shown in Figure 1.

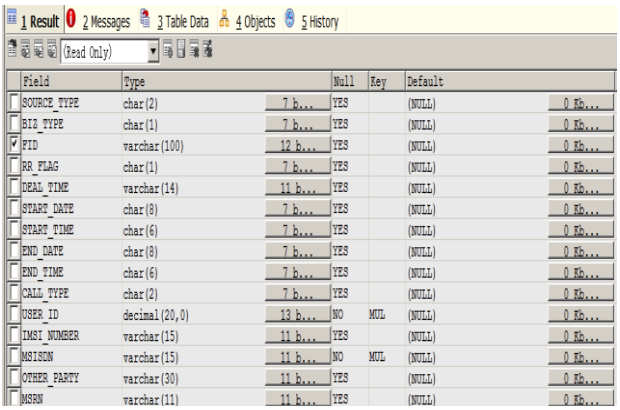

Figure 1. sd_zaozhuang.ucr_zz_tg_cdr10 structure.

4.2. Create Stored Procedure: plxd_shch

According to the table zz_haoma, the stored procedure can generate the statement of batch call. The core code is as follows:

```
Create or replace procedure plxd_shch (cx_type number,mounth_ds number, start_zq varchar2) as
  sql_min varchar 2(500);
  start_s   varchar(10):='01000000';
  end_e     varchar(10):='235959';
  v_min_sy     varchar2(400) ;
  v_yyy                    varchar2(100):='call sd_zaozhuang.cdr_insert(';
  v_yms                    varchar2(100):='call sd_zaozhuang.cdr_sm_insert(';
  v_gdyy                   varchar2(100):=  'call sd_zaozhuang.cdr_fix_insert(';
  v_gdms                   varchar2(100):= 'sd_zaozhuang.cdr_fixsm_insert(';
  v_gs                     varchar2(100):=  'call sd_zaozhuang.cdr_gs_insert(';
  v_lower number:=1; v_max    number;
  cursor cursor_1   is select * from zz_haoma;
  begin
  update          zz_haoma          set
```


2016 © The American Computational Science Society. All rights reserved.

```
type_cx=cx_type,start_yymm=to_date(start_zq,'yyyym
m'),ds_mounth=mounth_ds;
  delete   from zz;
  commit;
  for cxhm in   cursor_1   loop
  v_max                                          :=
trunc(months_between(sysdate,cxhm.start_yymm)) ;
   if cxhm.ds_mounth=2 then
  for i in v_lower ..(v_max-1) loop
  v_min_sy:=""||cxhm.haoma||""||','||""|| (to_char(
  last_day(add_months(sysdate,-i-1)),'yyyymm'))
  ||start_s ||""||','||""|| (to_char(last_day(add_months(
  sysdate,-i-1)),'yyyymm'))||substr((to_char(last_day(
  add_months(sysdate,-i-1)),'yyyymmdd')),7,8)
   || end_e||""||','||""||'0632'||""||','||""||'0'||""||');';
  case
   when   cxhm.type_cx='1' then
   sql_min   :=v_yyy||v_min_sy ;
   when   cxhm.type_cx='2' then
   sql_min   :=v_yms||v_min_sy ;
   when   cxhm.type_cx='3' then
  sql_min   :=v_gdyy||v_min_sy ;
  end case;
  insert into zz (v_v,s_t) values(sql_min,null);
  commit;
   end loop;
  end if;
```

For example: telephone number 18606420981 is stored in the table of zz_haoma, if you will generate batch detailed call statements from September 2015 to March 2016,you can call : plxd_shch (1,2,201509).Call statement is generated as follows:

```
  call sd_zaozhuang.cdr_insert('186.......1',
  '20151101000000','20151130235959','0632','0');
  call sd_zaozhuang.cdr_insert('186.......1',
  '20151201000000','20151231235959','0632','0');
  call sd_zaozhuang.cdr_insert('186.......1',
  '20160101000000','20160131235959','0632','0');
  call sd_zaozhuang.cdr_insert('186.......1',
  '20160201000000','20160229235959','0632','0');
  call sd_zaozhuang.cdr_insert('186.......1',
  '20160301000000','20160331235959','0632','0');
```

4.3. Create Three Stored Procedures

Create the following three stored procedures in the company history detail database:

sd_zaozhuang. cdr_insert: it is used to query the details of mobile voice data and insert into the intermediate table sd_zaozhuang.ucr_zz_tg_cdr10.

sd_zaozhuang.cdr_sm_insert: it is used to query the details of mobile sms data and insert into the intermediate table sd_zaozhuang. ucr_zza_tg_sm.

sd_zaozhuang.cdr_fix_insert: it is used to query the details of fixed telephone voice data and insert into the intermediate table sd_zaozhuang. ucr_zza_tg_cdr10_fix.

The three stored procedure is similar.

Take sd_zaozhuang.cdr_insert as an example, the core code is as follows:

```
delimiter $$;
drop procedure if exists `sd_zaozhuang`.`cdr_insert`$$
create definer=`xdcx_zz`@`%` procedure
  `cdr_insert`(msisdn varchar(32),starttime
  varchar(32),endtime varchar(32),areanum
  char(4),notype varchar(25))
begin
  declare   sqlstr varchar(100000);
  declare   body varchar(10240);
  declare   workstr varchar(200);
  declare   wherestr varchar(2000);
  set body="select * from zz_";
  set body=concat(body,substring(starttime,1,6),".");
  set sqlstr="insert into
    sd_zaozhuang.ucr_zz_tg_cdr10 ";
  set workstr="_tg_cdr";
  set workstr=concat(workstr,substring(starttime,5,2));
  set wherestr=" where msisdn=";
  set wherestr=concat(wherestr,"'",msisdn);
  set wherestr=concat(wherestr,"'");
  if areanum='0632' then
    set sqlstr=concat(sqlstr,body);
    set sqlstr=concat(sqlstr,"ucr_zza");
    set sqlstr=concat(sqlstr,workstr);
    set sqlstr=concat(sqlstr,wherestr);
    set sqlstr=concat(sqlstr," union all ");
    set sqlstr=concat(sqlstr,body);
    set sqlstr=concat(sqlstr,"ucr_zzb");
    set sqlstr=concat(sqlstr,workstr);
    set sqlstr=concat(sqlstr,wherestr);
    set sqlstr=concat(sqlstr," union all ");
    set sqlstr=concat(sqlstr,body);
    set sqlstr=concat(sqlstr,"ucr_zzc");
    set sqlstr=concat(sqlstr,workstr);
    set sqlstr=concat(sqlstr,wherestr);
    set sqlstr=concat(sqlstr," union all ");
    set sqlstr=concat(sqlstr,body);
    set sqlstr=concat(sqlstr,"ucr_zzd");
    set sqlstr=concat(sqlstr,workstr);
    set sqlstr=concat(sqlstr,wherestr);
  end if;
  set @sql = sqlstr;
  prepare stmt from @sql;
  execute stmt;
  end$$
  delimiter ;$$
```

4.4. Create View

Create view by sqlyog and export the detail query results from the follow tables:

sd_zaozhuang.ucr_zz_tg_cdr10
sd_zaozhuang. ucr_zza_tg_sm
sd_zaozhuang. ucr_zza_tg_cdr10_fix.

For example: we can create a view yd_select and export the detail query results from the table sd_zaozhuang.ucr_zz_tg_cdr10.

the core code is as follows:

```
delimiter $$;
```


2016 © The American Computational Science Society. All rights reserved.

```
drop view if exists `sd_zaozhuang`.`yd_select`$$
create algorithm=undefined definer=`xdcx_zz`@`%`
sql security definer view `yd_select` as ( select
( case
    when  f.biz_type=  '1' then    ' voice '
   when  f.biz_type=  '2' then   ' SMS '
    when  f.biz_type=  '3' then   'gprs'
    when  f.biz_type=  '4' then ' data service '
    when  f.biz_type=  'a' then ' easyown '
    when  f.biz_type=  'b' then ' fixed user ip service
    when    f.biz_type=    'c'  then  ' the ordering
generated monthly ticket '
 end) ' service type      ',
f.start_date ' call start date ',
f.start_time ' call start time ',
 f.call_duration ' call time seconds ',
 (case
    when  f.call_type='01'then ' the calling moc'
    when  f.call_type='02'then ' the called mtc'
    when  f.call_type='03'then ' call transfer forw'
    when  f.call_type='07'then ' SMS calling smmo'
    when  f.call_type='08'then ' SMS called smmt'
    when    f.call_type='11'then  '  fixed  telephone
calling poc'
    when    f.call_type='12'then  '  fixed  telephone
called ptc'
    when  f.call_type='20'then  ' call transfer called
split forw_mtc'
   end ) ' the type of call ',
   f.msisdn ' the number of queries ',
   f.other_party   ' the other phone number ',
   f.home_area_code ' home area number ',
   f.called_home_code ' the terminal area code ',
   (case
     when f.long_type1= '0'then ' local   call '
     when f.long_type1= '1'then ' provincial call '
     when f.long_type1= '2'then ' domestic toll call '
     when f.long_type1= '3'then ' international call '
   end    ) ' the type of call',
   f.lac1   ' location area no. 1',
   cell_id1 ' base station no. 1'   ,
   f.cfee ' basic call charge ' ,
   f.lfee ' long distance call charge ',
   f.otherfee ' other call charge '
   from   sd_zaozhuang.ucr_zz_tg_cdr10 f
   order by f.msisdn,f.start_date,f.start_time )$$
delimiter ;$$
```

4.5. Export the Detailed Query Results

We execute the SQL statement: select * from yd_select, the detailed list of the batch mobile voice is exported.

5. Optimization results and analysis

After the stage of testing and evaluation, the batch number of multi-billing period detailed query optimization is now on-line and put into use. During this period, a larger typical demand is: query 11,096 mobile phones from September 2015 to March 2016 for seven consecutive months of detailed list for today's mobile phone market research. If we use the current query system, the workload is enormous, even it is almost impossible to be completed. But we use the optimization system to query the detailed data quickly. The final results and related indicators are as shown in Table 1.

Table 1 the Optimization analysis of batch query

Optimization metric name	Optimization metric	Remarks
number of demand	11096	call details
number of accounts	7 months	for7 months
generate number of call statements	77672 items	11096*7
generate call statements time consuming	11.481 seconds	
batch calling takes time	216840 seconds	
number of queries	1927841 items	
average time-consuming	2.791 seconds / item	

From Table 1, we can see that the optimization of the batch query is realized by the combination of batch detail and multiple accounts, the query process is simple, the batch call speed is fast, the query result is accurate.

6. Conclusion

This optimization of batch inquiry for call details is a useful complement and innovation for the company's detailed inquiry system. It is the result of paying attention to full research and seeking truth from facts.

After nearly one year of using, practice has proved that this optimized batch query system provides a more efficient, more accurate and more efficient services, and can well meet query needs for the user and the company. The optimization has some practical value.

References

[1] S.Z. Dominik, P. Synak, A. Wojna, J. Wroblewsk, Fundamenta Informaticae, 127(2013):445–459.

[2] V.G. Kotukh, M.A. Miroshnik, S.N. Selevko, Telecommunications and Radio Engineering, 72(10):875-876.

[3] G.L. Li, J.H. Feng, X.F. Zhou, J.Y. Wang, The VLDB Journal, 20(2001):1-2.

[4] G.L. Li, L. Guo, J.H. Feng, F. Lin, L.Z. Zhou. Lecture Notes in Computer Science, 5071(2008):193-197.

[5] G.C, Deka. IT Professional, 16(2005):50-51.

[6] H.R. Aprianggi. International Journal of Computer Trends and Technology, 10(2004):197-198.

2016 © The American Computational Science Society. All rights reserved.