基于案例学SQL优化第2周

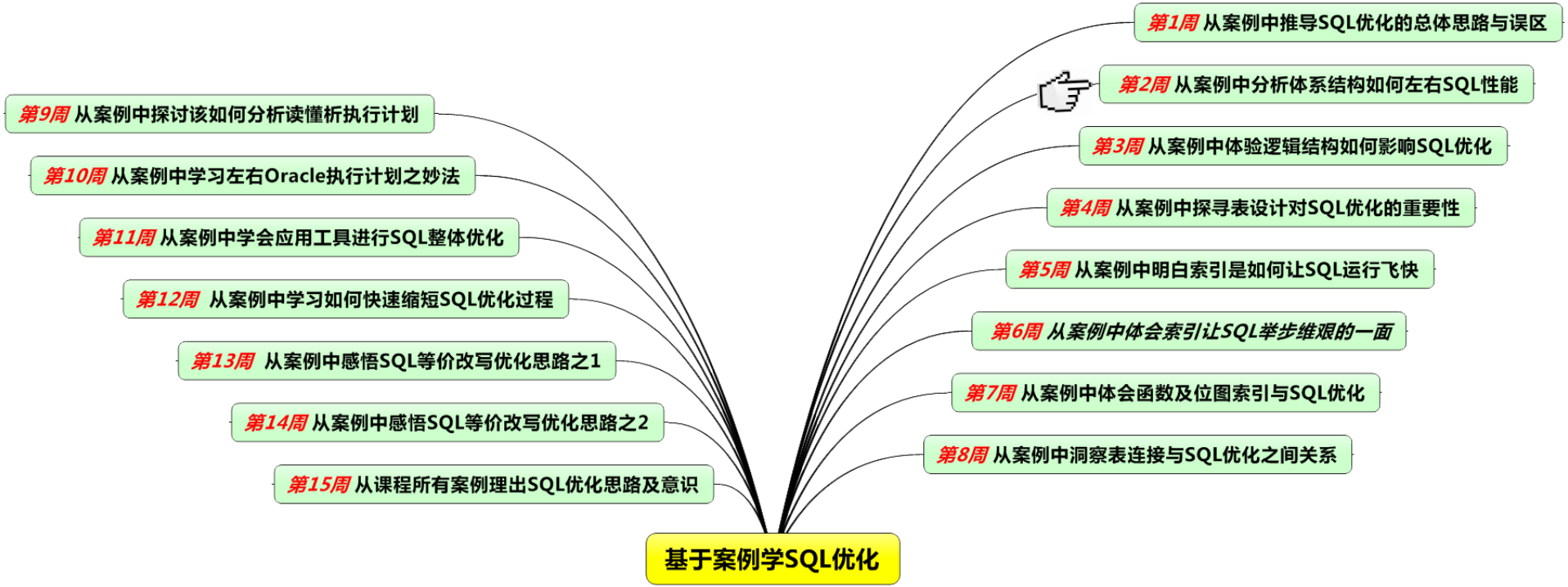
法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

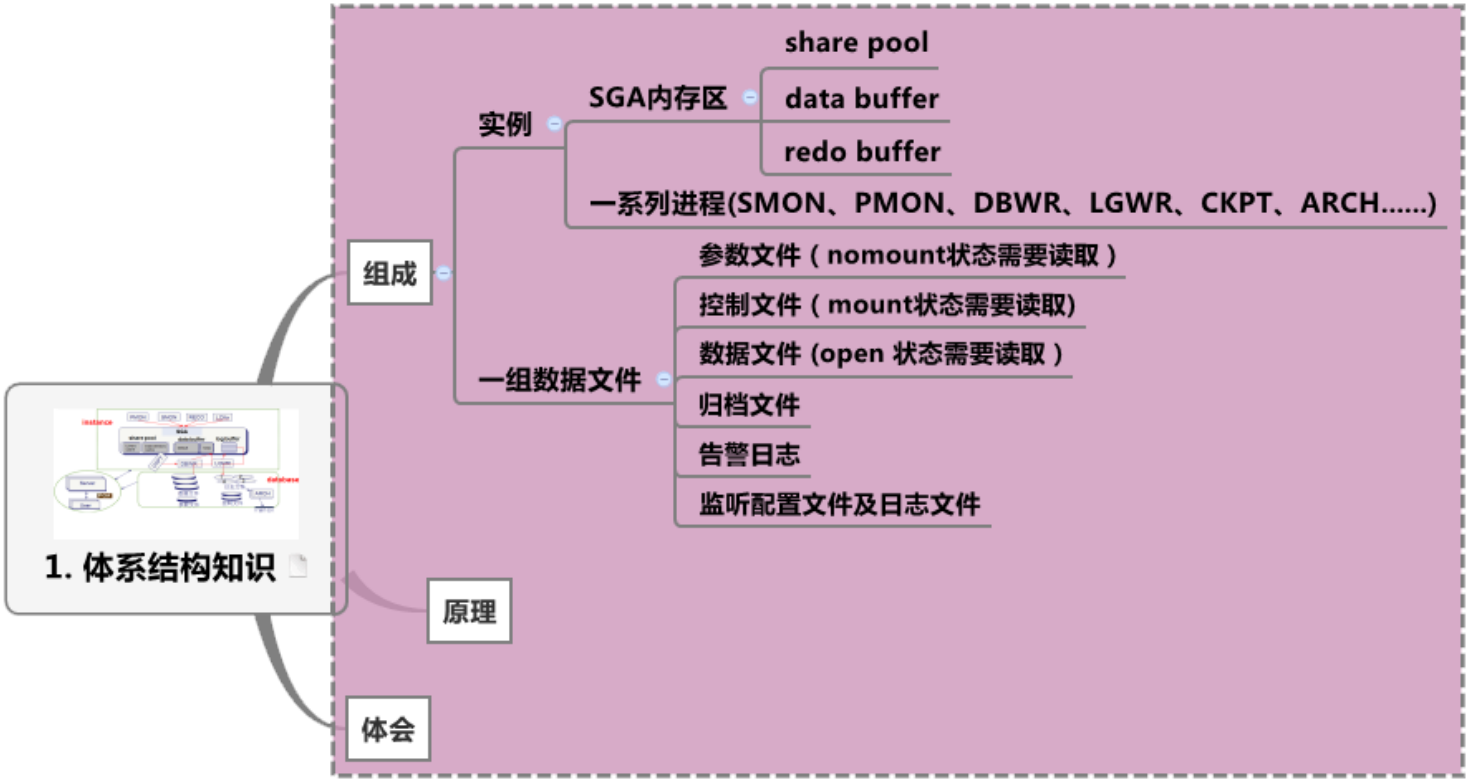
http://edu.dataguru.cn

当前课程进度

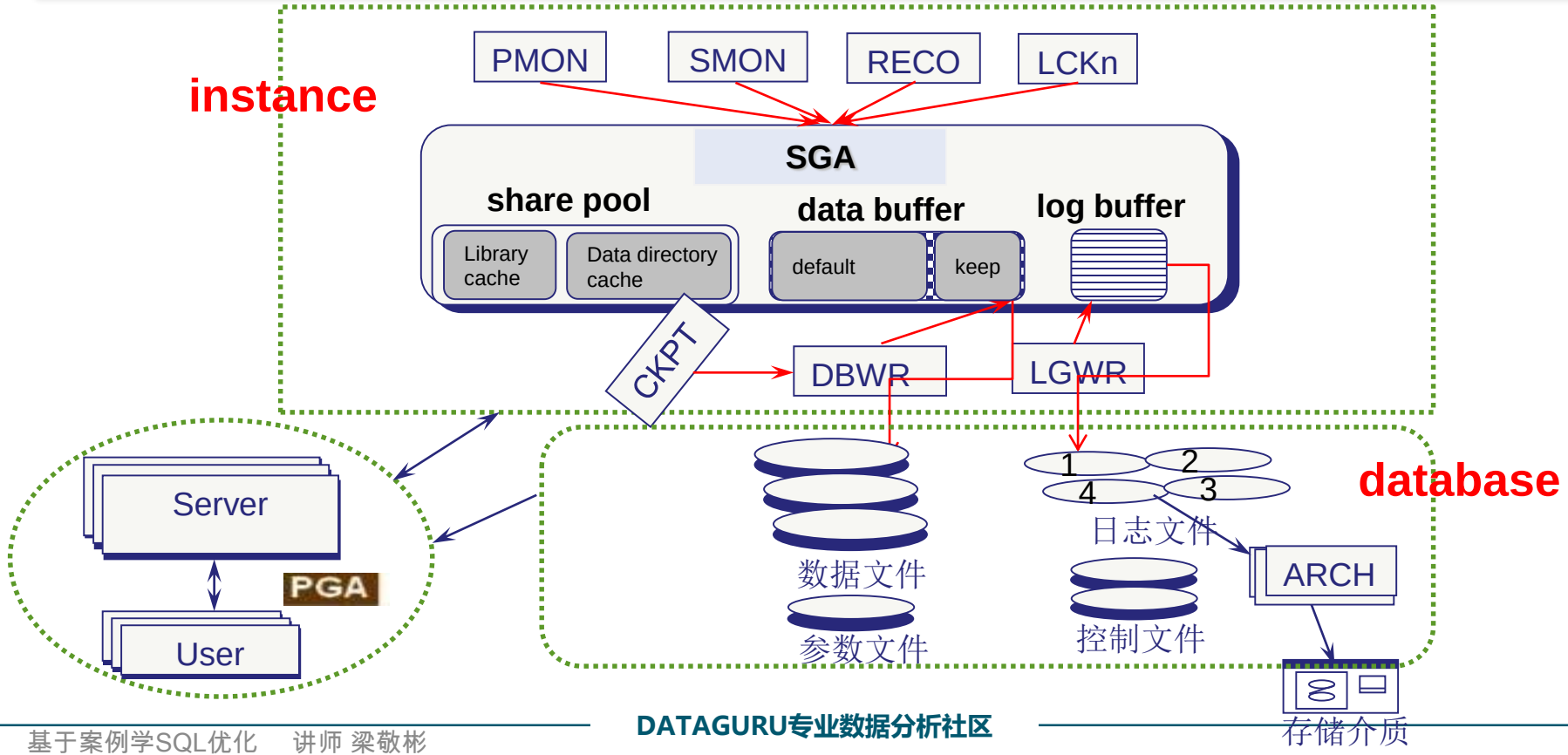

第1周 从案例中推导SQL优化的总体思路与误区
第2周 从案例中分析体系结构如何左右SQL性能
第3周 从案例中体验逻辑结构如何影响SQL优化
第4周 从案例中探寻表设计对SQL优化的重要性
第5周 从案例中明白索引是如何让SQL运行飞快
第6周 从案例中体会索引让SQL举步维艰的一面
第7周 从案例中体会函数及位图索引与SQL优化
第8周 从案例中洞察表连接与SQL优化之间关系
第9周 从案例中探讨该如何分析读懂析执行计划
第10周 从案例中学习左右Oracle执行计划之妙法
第11周 从案例中学会应用工具进行SQL整体优化
第12周 从案例中学习如何快速缩短SQL优化过程
第13周 从案例中感悟SQL等价改写优化思路之1
第14周 从案例中感悟SQL等价改写优化思路之2
第15周 从课程所有案例理出SQL优化思路及意识
基于案例学SQL优化

体系结构知识

体系结构知识


share pool
SGA内存区
data buffer
redo buffer
实例
一系列进程(SMON、PMON、DBWR、LGWR、CKPT、ARCH……)
参数文件（nomount状态需要读取）
组成
控制文件（mount状态需要读取)
数据文件 (open 状态需要读取）
一组数据文件
归档文件
告警日志
监听配置文件及日志文件
1. 体系结构知识
原理
体会

Oracle的体系结构(简化版）



体会体系结构中的SGA

1. 未启动数据库前的SGA分配情况

```
[oracle@itmapp3 ~]$ ipcs -m
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00020014 32769      gserver    666        12         10
0x00028081 131076     kfv3       666        12         6
0x00024024 1015828    nmv3       666        12         6
```

2. 启动数据库后的SGA分配情况

```
[oracle@itmapp3 ~]$ ipcs -m
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
0x66a02988 1114112    oracle     660        2485125120 14
0x00020014 32769      gserver    666        12         10
0x00028081 131076     kfv3       666        12         6
0x00024024 1015828    nmv3       666        12         6
```

3. 为啥是2485125120字节，原来如此：

```
SQL> show parameter sga
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     FALSE
sga_max_size                         big integer 2368M
sga_target                           big integer 2368M
```

注：以上是10g的分配方式，在11g中默认是内存管理分配sga（将sga和pga一同管理）

```
SQL> show parameter memory
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
hi_shared_memory_address             integer     0
memory_max_target                    big integer 4G
memory_target                        big integer 4G
shared_memory_address                integer     0
```

体会体系结构中的进程

1. 未启动数据库前的oracle进程情况

```
[oracle@itmapp3 ~]$ ps -ef |grep ora
oracle    6796     1  0 Oct28 ?        00:06:16 /home/oracle/product/10.2.0/db_1/bin/tnslsnr LISTENER -inherit
```

2. 启动数据库后的oracle本地进程

```
[oracle@itmapp3 ~]$ ps -ef |grep ora
oracle    5601     1  0 05:50 ?        00:00:00 ora_pmon_itmtest
oracle    5606     1  0 05:50 ?        00:00:00 ora_psp0_itmtest
oracle    5614     1  0 05:50 ?        00:00:00 ora_mman_itmtest
oracle    5616     1  0 05:50 ?        00:00:00 ora_dbw0_itmtest
oracle    5618     1  0 05:50 ?        00:00:00 ora_lgwr_itmtest
oracle    5620     1  0 05:50 ?        00:00:00 ora_ckpt_itmtest
oracle    5624     1  0 05:50 ?        00:00:00 ora_smon_itmtest
oracle    5626     1  0 05:50 ?        00:00:00 ora_reco_itmtest
oracle    5628     1  0 05:50 ?        00:00:00 ora_cjq0_itmtest
oracle    5630     1  0 05:50 ?        00:00:00 ora_mmon_itmtest
oracle    5632     1  0 05:50 ?        00:00:00 ora_mmnl_itmtest
oracle    5634     1  0 05:50 ?        00:00:00 ora_d000_itmtest
oracle    5636     1  0 05:50 ?        00:00:00 ora_s000_itmtest
oracle    5637  5037  0 05:50 ?        00:00:00 oracleitmtest (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
oracle    6796     1  0 Oct28 ?        00:06:17 /home/oracle/product/10.2.0/db_1/bin/tnslsnr LISTENER -inherit
```

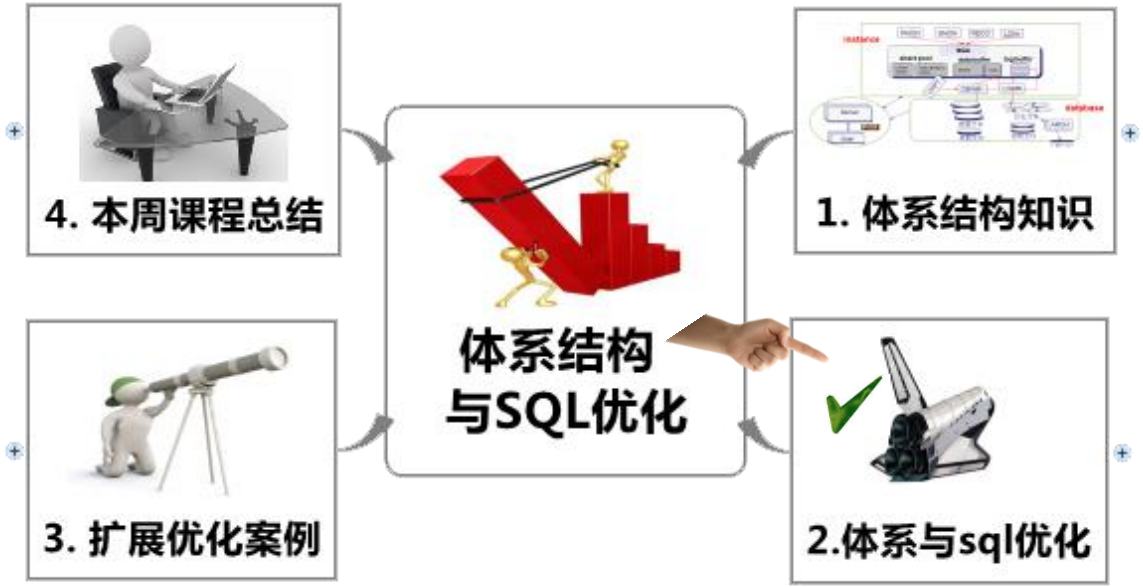
3. 接下来用户连上来的进程情况

```
oracle    6799     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    6807     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    6811     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    6820     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    6822     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    6824     1  0 05:34 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    8801     1  0 Nov04 ?        00:00:00 oracleitmtest (LOCAL=NO)
oracle    8803     1  0 Nov04 ?        00:00:00 oracleitmtest (LOCAL=NO)
```

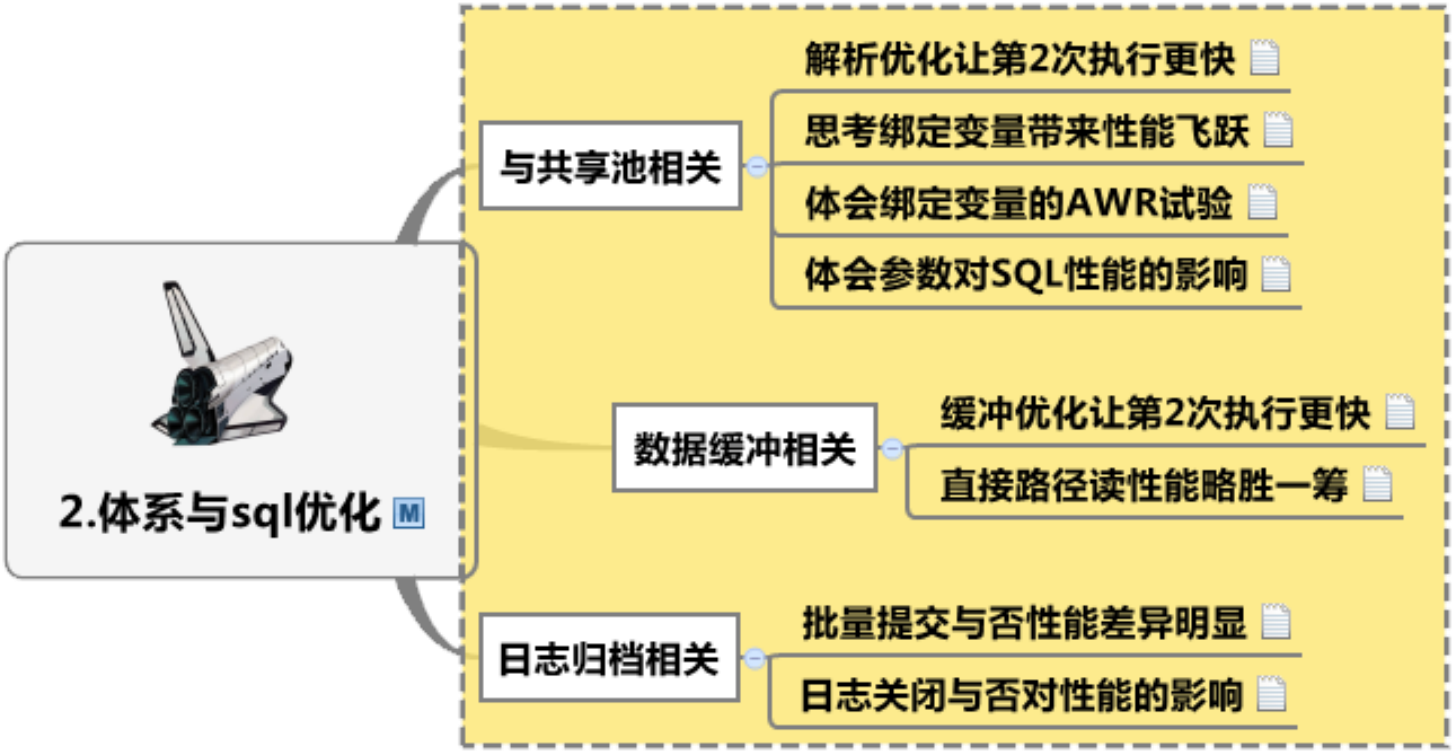
4. Oracle本地进程和外部连接进程统计

```
[oracle@itmapp3 ~]$ ps -ef |grep LOCAL=NO|wc -l
150
[oracle@itmapp3 ~]$ ps -ef |grep ora|wc -l
172
```

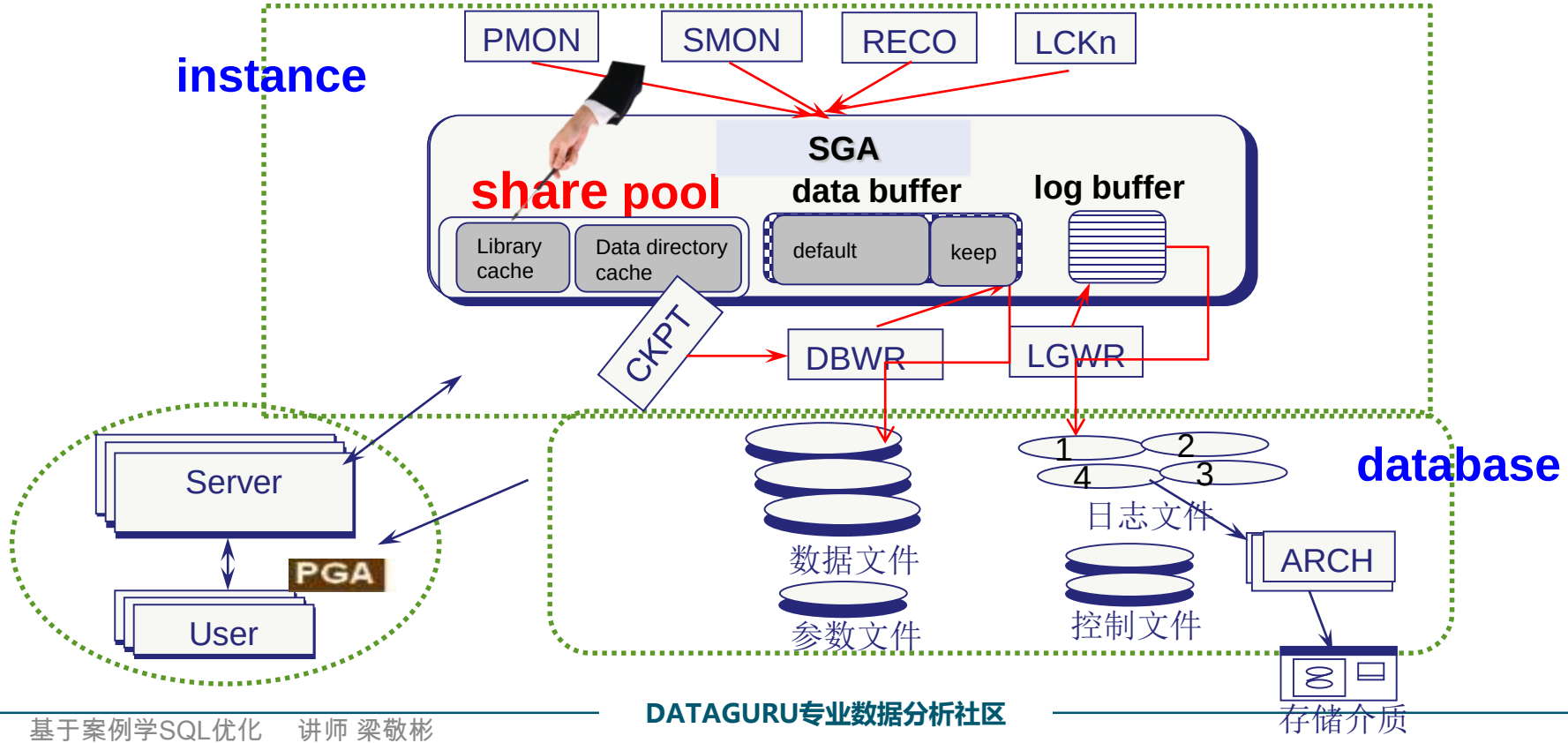
体系与SQL优化



体系与SQL优化


2.体系与sql优化
与共享池相关
解析优化让第2次执行更快
思考绑定变量带来性能飞跃
体会绑定变量的AWR试验
体会参数对SQL性能的影响
数据缓冲相关
缓冲优化让第2次执行更快
直接路径读性能略胜一筹
日志归档相关
批量提交与否性能差异明显
日志关闭与否对性能的影响

体系结构中共享池研究


instance
PMON
SMON
RECO
LCKn
SGA
share pool
data buffer
log buffer
Library cache
Data directory cache
default
keep
CKPT
DBWR
LGWR
Server
PGA
User
database
数据文件
参数文件
1
2
3
4
日志文件
ARCH
控制文件
存储介质

共享池相关的优化

解析优化让第2次执行更快

```
SQL> --第1次执行
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间:  00: 00: 00.10
执行计划
----------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 60918 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------
统计信息
----------------------------------------------------------
         28  recursive calls
          0  db block gets
       1103  consistent gets
       1038  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> --第2次执行
SQL> --该命令只是为了先不考虑2次执行物理读减少带来的效果，只考虑减少解析的的优化效
SQL> alter system flush buffer_cache;
系统已更改。
已用时间:  00: 00: 00.03
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间:  00: 00: 00.07
执行计划
----------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 60918 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1043  consistent gets
       1039  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
```

共享池相关的优化

思考绑定变量所带来的性能飞跃

```
SQL> begin
  2       for i in 1 .. 100000
  3       loop
  4            execute immediate
  5            'insert into t values ( '||i||')';
  6       end loop;
  7       commit;
  8  end;
  9  /

PL/SQL 过程已成功完成。

已用时间:  00: 00: 43.50
```

```
SQL>--使用绑定变量

SQL> begin
  2       for i in 1 .. 100000
  3       loop
  4            execute immediate
  5            'insert into t values ( :x )' using i;
  6       end loop;
  7            commit;
  8  end;
  9  /

PL/SQL 过程已成功完成。

已用时间:  00: 00: 04.77
```

共享池相关的优化

自行体会硬解析次数和执行次数

SQL | Output | Statistics

```
select t.sql_text, t.sql_id, t.executions, t.parse_calls
  from v$sql t
 where sql_text like 'insert into t values%';
```

	SQL_TEXT	SQL_ID	EXECUTIONS	PARSE_CALLS
1	insert into t values (96283)	b5qk5k0cg000f	1	1
2	insert into t values (95426)	ch0act8w48019	1	1
3	insert into t values (94549)	0v81tsyp1401h	1	1
4	insert into t values (96599)	2gn81fnjd001m	1	1
5	insert into t values (94842)	7h9y8au96n02g	1	1
6	insert into t values (96579)	b8ubtwdrac02p	1	1
7	insert into t values (92824)	aw6f4nsnt0032	1	1
8	insert into t values (92398)	1ds5nr9nfw034	1	1
9	insert into t values (95685)	1uuu6w9ykn034	1	1
10	insert into t values (96192)	g3cfj18vh8039	1	1
11	insert into t values (98214)	bsv2kq808403m	1	1
12	insert into t values (96354)	4zrqq88kw003w	1	1
13	insert into t values (98195)	gmu01syv2c04y	1	1
14	insert into t values (98652)	agvzvxvrn4056	1	1
15	insert into t values (97683)	0nvm3ta8t005f	1	1

SQL | Output | Statistics

```
select t.sql_text, t.sql_id, t.executions, t.parse_calls
  from v$sql t
 where sql_text like 'insert into t values (:x)%';
```

	SQL_TEXT	SQL_ID	EXECUTIONS	PARSE_CALLS
1	insert into t values (:x)	bbz6pdq577unj	100000	1

共享池相关的优化

体会绑定变量的AWR试验

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.9	3.2	0.00	3.17
DB CPU(s):	0.8	3.1	0.00	3.14
Redo size:	436,015.3	1,631,772.8		
Logical reads:	7,943.7	29,729.2		
Block changes:	3,625.5	13,568.4		
Physical reads:	8.0	30.0		
Physical writes:	0.3	1.2		
User calls:	0.3	1.0		
Parses:	1,823.4	6,824.0		
Hard parses:	1,790.5	6,701.0		
W/A MB processed:	0.5	1.9		
Logons:	0.1	0.2		
Executes:	1,931.9	7,229.9		
Rollbacks:	0.0	0.0		
Transactions:	0.3			

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.90	In-memory Sort %:	100.00
Library Hit %:	62.91	Soft Parse %:	1.80
Execute to Parse %:	5.61	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	98.37	% Non-Parse CPU:	25.58

共享池相关的优化

体会绑定变量的trace试验

--未使用绑定变量的

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call	count	cpu	elapsed	disk	query	current	rows
Parse	10075	3.32	3.39	0	0	0	0
Execute	10379	0.60	0.56	11	10049	30380	10006
Fetch	683	0.01	0.10	219	1565	0	1418
total	21137	3.94	4.06	230	11614	30380	11424

---使用绑定变量的

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call	count	cpu	elapsed	disk	query	current	rows
Parse	76	0.03	0.02	0	0	0	0
Execute	10379	1.41	1.45	12	52	10397	10006
Fetch	683	0.06	0.09	219	1565	0	1418
total	11138	1.51	1.58	231	1617	10397	11424

注意静态SQL自动绑定变量

```
SQL> set linesize 1000
SQL> column sql_text format a50
SQL> select t.sql_text, t.sql_id, t.executions, t.parse_calls
  2    from v$sql t
  3   where lower(sql_text) like 'insert into t values%';

SQL_TEXT                                           SQL_ID        EXECUTIONS PARSE_CALLS
-------------------------------------------------- ------------- ---------- -----------
INSERT INTO T VALUES (:B1 )                        2n9c7yuww4dx4     100000           0
```

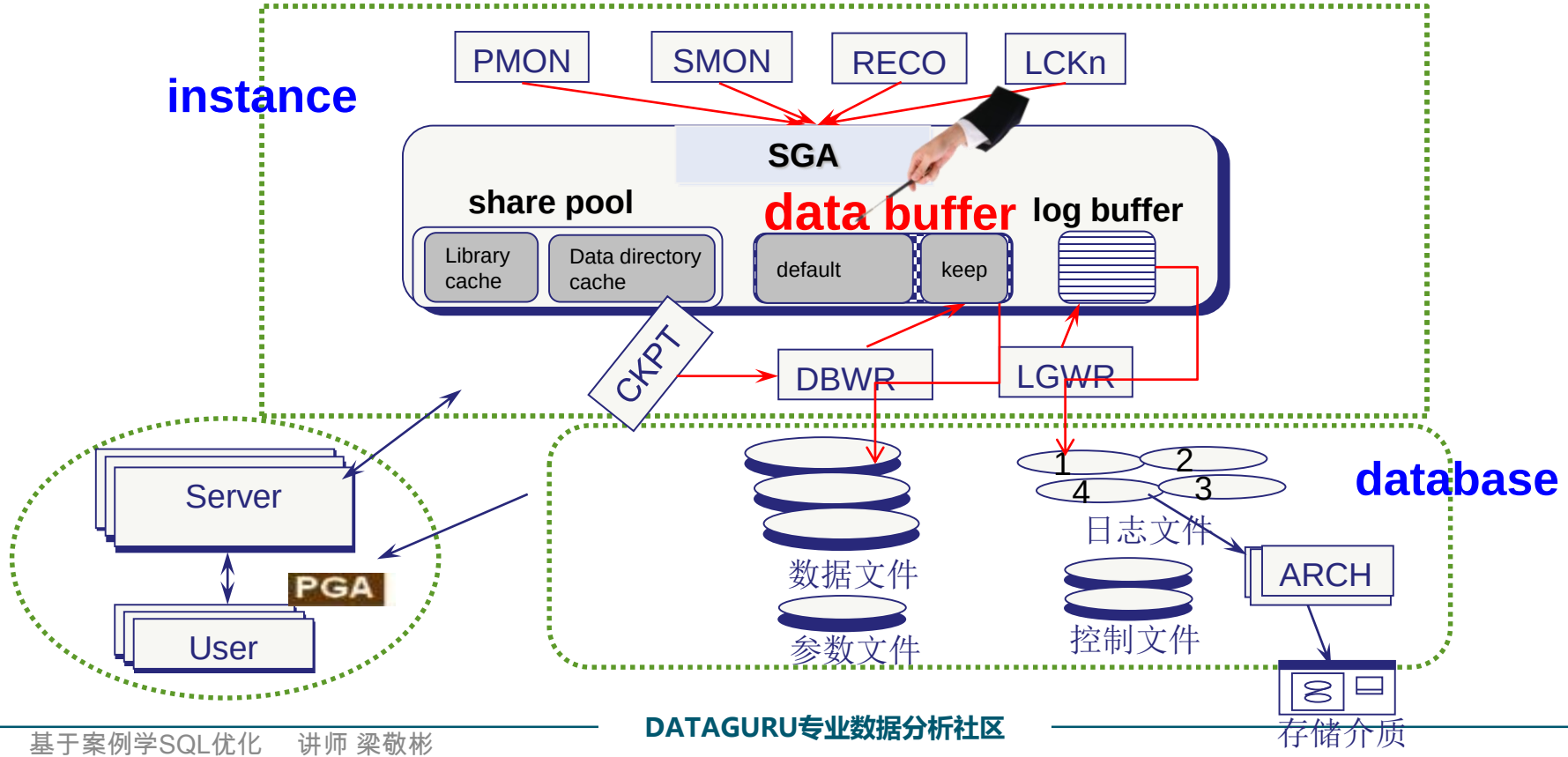
共享池相关的优化

体会参数对SQL性能的影响

```
SQL> set timing on
SQL> alter session set session_cached_cursors=0;
会话已更改。
SQL> --使用绑定变量
SQL> begin
  2       for i in 1 .. 100000
  3       loop
  4           execute immediate
  5           'insert into t values ( :x )' using i;
  6       end loop;
  7       commit;
  8   end;
  9   /
PL/SQL 过程已成功完成。
已用时间:  00: 00: 08.70
```

```
SQL> set timing on
SQL> alter session set session_cached_cursors=50;
SQL> --使用绑定变量
SQL> begin
  2       for i in 1 .. 100000
  3       loop
  4           execute immediate
  5           'insert into t values ( :x )' using i;
  6       end loop;
  7       commit;
  8   end;
  9   /
PL/SQL 过程已成功完成。
已用时间:  00: 00: 04.69
```

体系结构中的数据缓冲池研究



数据缓冲相关优化

缓冲优化让第2次执行更快

```
SQL> --第1次执行
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间:  00: 00: 00.12
执行计划
----------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 60918 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------
统计信息
----------------------------------------------------------
         28  recursive calls
          0  db block gets
       1103  consistent gets
       1038  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> --第2次执行
SQL> --该命令只是为了先不考虑解析的优化,单纯考虑第2次执行物理读减少带来的优化效应
SQL> alter system flush shared_pool;
系统已更改。
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间:  00: 00: 00.04
执行计划
----------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 60918 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------
统计信息
----------------------------------------------------------
        282  recursive calls
          0  db block gets
       1131  consistent gets
          0  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

数据缓冲相关优化

解析和缓冲优化正常一起来

```
SQL> --第1次执行
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间: 00: 00: 00.11
执行计划
----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 69684 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------------
统计信息
----------------------------------------------------------
         28  recursive calls
          0  db block gets
       1110  consistent gets
       1038  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> --第2次执行
SQL> --这里不做 shared_pool和buffer_cache的flush
SQL> select count(*) from t;
  COUNT(*)
----------
     72884
已用时间: 00: 00: 00.02
执行计划
----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   291   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 69684 |   291   (1)| 00:00:04 |
----------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1043  consistent gets
          0  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

数据缓冲相关优化

直接路径读性能略胜一筹

```
SQL> --测试普通插入
SQL> set timing on
SQL> insert into test select * from t;
已创建1166096行。
已用时间:  00: 00: 06.78
```

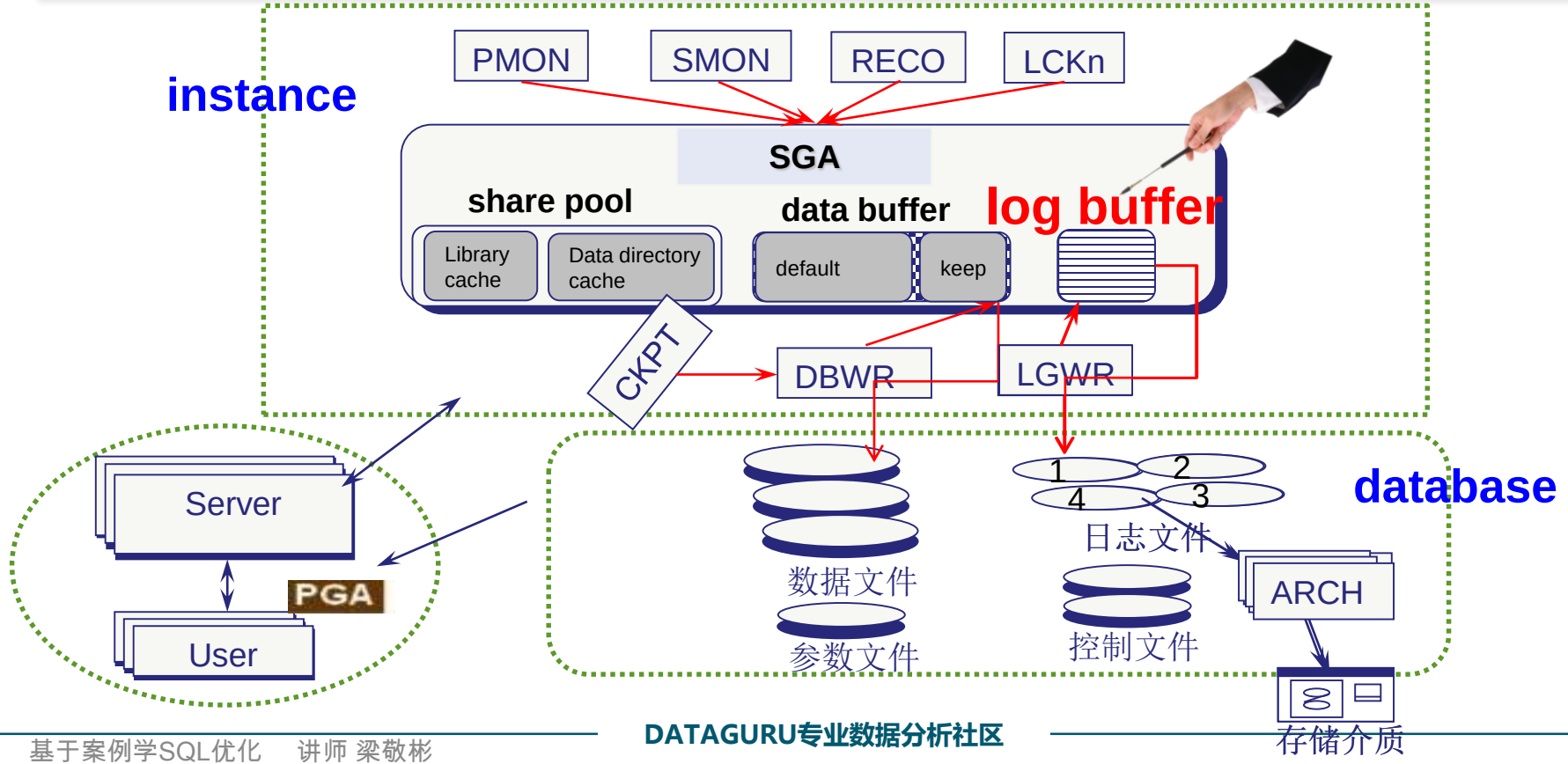
```
SQL> --测试直接路径读方式
SQL> set timing on
SQL> insert  /*+ append */ into test select * from t;
已创建1166096行。
已用时间:  00: 00: 01.24
```

能胜一筹是有原因的

```
SQL> --注意普通方式插入test表后输出的物理读（首次执行,看physical reads）
SQL> set autotrace traceonly
SQL> select count(*) from test;
------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |  4685   (1)| 00:00:57 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |  1255K|  4685   (1)| 00:00:57 |
------------------------------------------------------------------------
统计信息
----------------------------------------------------------
         29  recursive calls
          1  db block gets
      17008  consistent gets
          0  physical reads
        176  redo size
        426  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> --注意直接路径方式插入test表试验输出的物理读（首次执行,看physical reads）
SQL> set autotrace traceonly
SQL> select count(*) from test;
执行计划
------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |  4559   (1)| 00:00:55 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |  1052K|  4559   (1)| 00:00:55 |
------------------------------------------------------------------------
统计信息
----------------------------------------------------------
         28  recursive calls
          1  db block gets
      16688  consistent gets
      16604  physical reads
        168  redo size
        426  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

体系结构中的日志相关研究



日志归档相关优化

批量提交与否性能差异明显

```
SQL> drop table t purge;
表已删除。
SQL> create table t(x int);
表已创建。
SQL> set timing on
SQL> begin
  2       for i in 1 .. 100000 loop
  3          insert into t1 values (i);
  4          commit;
  5       end loop;
  6  end;
  7  /
PL/SQL 过程已成功完成。
已用时间:  00: 00: 11.21
```

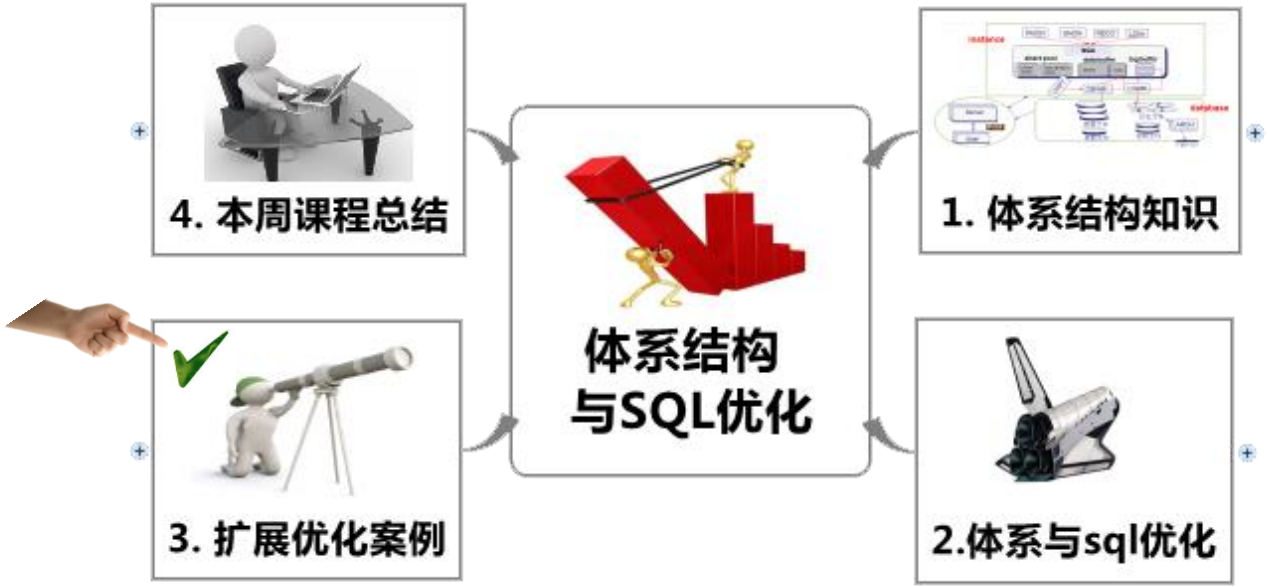
```
SQL> drop table t purge;
表已删除。
SQL> create table t(x int);
表已创建。
SQL> begin
  2       for i in 1 .. 100000  loop
  3          insert into t values (i);
  4       end loop;
  5    commit;
  6  end;
  7  /
PL/SQL 过程已成功完成。
已用时间:  00: 00: 04.26
```

日志归档相关优化

日志关闭与否对性能的影响

```
SQL> --测试直接路径读方式
SQL> drop table test;
表已删除。
SQL> create table test  as select * from dba_objects where 1=2;
表已创建。
SQL> set timing on
SQL> insert  /*+ append */ into test select * from t;
已创建4664384行。
已用时间:  00: 00: 05.01
SQL> --测试nolgging关闭日志+直接路径读方式
SQL> drop table test;
表已删除。
SQL> create table test  as select * from dba_objects where 1=2;
表已创建。
SQL> alter table test nologging;
表已更改。
SQL> set timing on
SQL> insert  /*+ append */ into test select * from t;
已创建4664384行。
已用时间:  00: 00: 04.39
```

扩展优化案例



扩展优化案例

3. 扩展优化案例

- 与共享池相关
  - 头疼，如何查硬解析问题
  - 纠结，绑定变量也有冬天
  - 经典，SQL的逻辑读变零
  - 经典，函数的逻辑读成零
- 数据缓冲相关
  - 感谢，keep让sql跑更快
  - 细致，查系统各维度规律
- 日志归档相关
  - 规律，日志切换有据可查
  - 巧妙，逮到提交过频语句

共享池的相关案例

头疼，如何查硬解析问题

```
SQL> ---执行完上述动作后，以下SQL语句可以完成未绑定变量语句的统计
SQL> set linesize 266
SQL> col  sql_text_wo_constants format a30
SQL> col  module format  a30
SQL> col  CNT format  999999
SQL> select sql_text_wo_constants, module,count(*) CNT
  2    from t_bind_sql
  3   group by sql_text_wo_constants,module
  4  having count(*) > 100
  5   order by 3 desc;

SQL_TEXT_WO_CONSTANTS          MODULE                           CNT
------------------------------ ------------------------------ -------
INSERT INTO T VALUES ( @)      SQL*Plus                        7366
```

共享池的相关案例

纠结，绑定变量也有冬天

```
SQL> SELECT count(pad) FROM t WHERE id < 990;
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |     1 |   105 |     7   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE     |      |     1 |   105 |            |          |
|*  2 |   TABLE ACCESS FULL| T    |   990 |   101K|     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------

SQL> SELECT count(pad) FROM t WHERE id < 10;
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |      |     1 |   105 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE               |      |     1 |   105 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T    |     9 |   945 |     3   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_PK |     9 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

SQL> EXECUTE :id := 990;
PL/SQL 过程已成功完成。
SQL> SELECT count(pad) FROM t WHERE id < :id;
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |      |     1 |   105 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE               |      |     1 |   105 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T    |    50 |  5250 |     3   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_PK |     9 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

SQL> EXECUTE :id := 10;
SQL> SELECT count(pad) FROM t WHERE id < :id;
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |      |     1 |   105 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE               |      |     1 |   105 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T    |    50 |  5250 |     3   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_PK |     9 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

共享池的相关案例

经典，SQL的逻辑读变零

```
SQL> ---接下来再次执行(居然发现逻辑读为0)：
SQL> set autotrace on
SQL> select /*+ result_cache */ count(*) from t;
  COUNT(*)
----------
    145762
已用时间:  00: 00: 00.01
执行计划
----------------------------------------------------------------------------------------------
| Id  | Operation            | Name                           | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                                |     1 |   589   (1)| 00:00:08 |
|   1 |  RESULT CACHE        | d827qx1jmwjc86yqynrp1kvpny     |       |            |          |
|   2 |   SORT AGGREGATE     |                                |     1 |            |          |
|   3 |    TABLE ACCESS FULL| T                              |   277K|   589   (1)| 00:00:08 |
----------------------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          0  consistent gets
          0  physical reads
          0  redo size
        425  bytes sent via SQL*Net to client
        416  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

共享池的相关案例

经典，函数的逻辑读成零

```
SQL> --看调用F_NO_RESULT_CACHE执行第2次后的结果
SQL> SELECT F_NO_RESULT_CACHE FROM DUAL;

F_NO_RESULT_CACHE
-----------------
            72883
统计信息
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1043  consistent gets
          0  physical reads
          0  redo size
        434  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> --看调用F_RESULT_CACHE执行第2次后的结果
SQL> SELECT F_RESULT_CACHE FROM DUAL;

F_RESULT_CACHE
--------------
         72883
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          0  consistent gets
          0  physical reads
          0  redo size
        431  bytes sent via SQL*Net to client
        415  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

数据缓冲相关案例

感谢，keep让sql跑更快

```
--未执行KEEP命令，通过如下查询出BUFFER_POOL列值为DEFAULT，表示未KEEP。
select BUFFER_POOL from user_tables where  TABLE_NAME='T';
BUFFER_POOL
--------------------
  DEFAULT

select BUFFER_POOL from user_indexes where INDEX_NAME='IDX_OBJECT_ID';

BUFFER_POOL
--------------------
  DEFAULT
--执行KEEP操作后，通过如下查询出BUFFER_POOL列值为KEEP,表示已经KEEP成功了
select BUFFER_POOL from user_tables where  TABLE_NAME='T';

BUFFER_POOL
--------------------
  KEEP

select BUFFER_POOL from user_indexes where INDEX_NAME='IDX_OBJECT_ID';

BUFFER_POOL
--------------------
  KEEP
```

数据缓冲相关案例

细致，查系统各维度规律

	SNAP_DATE	TIME	elapse(min)	DB time(min)	REDO	redo/s	LOGICAL	logical/s	PHYSICAL	phy/s	EXECS	execs/s	PARSE	parse/s	HARDP
1	13/11/07	18:00	64.22	0.51	105873536	27478.21	1375950	357.11	225	0.06	160181	41.57	9018	2.34	
2	13/11/07	16:56	115.77	0.04	1622588	233.6	43732	6.3	11	0	7177	1.03	4045	0.58	
3	13/11/07	15:00	60.18	0.06	5690688	1575.93	106134	29.39	5518	1.53	13346	3.7	7542	2.09	
4	13/11/07	14:00	60.17	0.65	197937568	54830.35	2338751	647.85	7149	1.98	406318	112.55	10217	2.83	
5	13/11/07	13:00	60.18	0.21	126920828	35148.39	666700	184.63	6791	1.88	14344	3.97	9617	2.66	
6	13/11/07	12:00	59.17	0.9	357269612	100639.33	1655494	466.34	6976	1.97	15255	4.3	9805	2.76	
7	13/11/07	11:00	60.17	0.57	60423804	16737.9	310342	85.97	4853	1.34	553894	153.43	9828	2.72	
8	13/11/07	08:01	60.18	0.06	2089688	578.7	59616	16.51	136	0.04	10912	3.02	6564	1.82	
9	13/11/07	07:00	60.17	0.05	2543732	704.63	68438	18.96	469	0.13	12292	3.4	7171	1.99	
10	13/11/07	06:00	34.1	0.62	3908728	1910.42	2313056	1130.53	2890	1.41	41855	20.46	8246	4.03	
11	13/11/08	22:00	60.18	1.25	4584236	1269.52	500443	138.59	9367	2.59	50851	14.08	10969	3.04	
12	13/11/08	21:00	60.17	0.06	2188264	606.17	57236	15.85	10	0	10882	3.01	6581	1.82	
13	13/11/08	20:00	59.18	0.06	2340524	659.12	59905	16.87	11	0	10844	3.05	6570	1.85	
14	13/11/08	19:01	60.9	0.09	2363340	646.78	55629	15.22	26	0.01	9964	2.73	5928	1.62	
15	13/11/08	18:00	59.18	0.06	2395280	674.54	64119	18.06	40	0.01	11448	3.22	6930	1.95	

日志归档相关案例

巧妙，逮到提交过频语句

```
SQL>  select t1.sid, t1.value, t2.name
  2     from v$sesstat t1, v$statname t2
  3    where t2.name like '%user commits%'
  4      and t1.STATISTIC# = t2.STATISTIC#
  5      and value >= 10000
  6    order by value desc;
    SID       VALUE NAME
------ ---------- -------------------------
  194     100000 user commits

SQL>   select t.SID,t.PROGRAM,t.EVENT,t.LOGON_TIME,t.WAIT_TIME,t.SECONDS_IN_WAIT,t.SQL_ID,t.PREV_SQL_ID
          from v$session t
       where sid in(194) ;

SID PROGRAM      EVENT                          LOGON_TIME    WAIT_TIME SECONDS_IN_WAIT SQL_ID    PREV_SQL_ID
---------- ------------------------------------------------------------------------------------------------
194 sqlplus.exe SQL*Net message from client   13-11月-13     0              54              ccpn5c32bmfmf

SQL>   select t.sql_id, t.sql_text, t.EXECUTIONS, t.FIRST_LOAD_TIME,t.LAST_LOAD_TIME
          from v$sql t
        where sql_id in ('ccpn5c32bmfmf');
SQL_ID          SQL_TEXT                                          EXECUTIONS FIRST_LOAD_TIME            LAST_LOAD_TIME
-------------------------------------------------------------------------------------------------------------------
ccpn5c32bmfmf   begin  for i in 1 .. 100000 loop                          1          2013-11-13/14:41:18    2013-11-13/14:46:34
                         insert into t values (i);
                       commit;
                    end loop;
                end;
```

日志归档相关案例

规律，日志切换有据可查

	DAY	H00	H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12	H13	H14	H15	H16	H17	H18	H19	H20	H21	H22	H23	TOTAL
1	11/13	0	0	0	0	0	0	1	0	1	0	0	0	0	3	9	2	25	18	1	0	5	0	0	0	65
2	11/12	0	0	0	0	1	4	1	0	7	8	0	1	17	36	0	0	1	5	1	0	1	0	1	0	84
3	11/11	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	5
4	11/10	0	0	0	0	0	0	1	0	0	0	8	0	0	0	0	0	0	1	1	0	0	0	0	0	11
5	11/09	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	0	6
6	11/08	1	0	0	0	0	1	0	0	0	2	3	0	0	1	1	0	0	0	0	0	0	0	2	0	11
7	11/07	0	0	0	0	0	1	0	0	5	0	2	8	3	5	0	0	0	3	1	0	1	0	0	1	30
8	11/06	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	4	0	1	0	2	0	11
9	11/05	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	4
10	11/04	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	4

日志归档相关案例

迷案，跟踪日志暴增故障

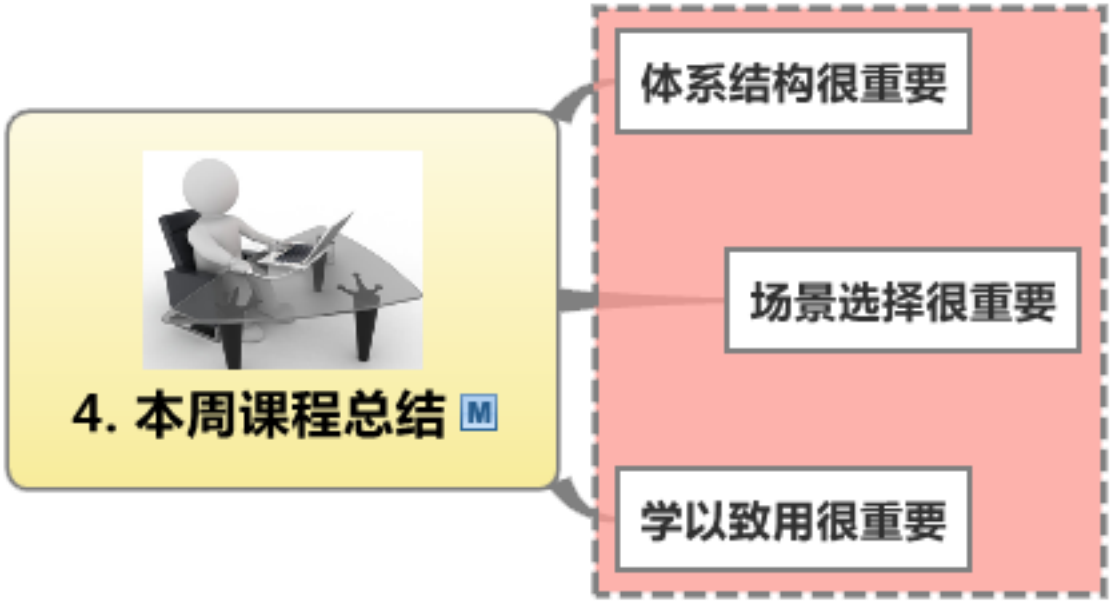
---执行了大量针对test_redo表的INSERT操作后，，我们开始根据如下方法进行跟踪，看能否找到更新语句及对应的表。

```
--执行了大量的针对test_redo表的INSERT操作后，，我们开始根据如下方法进行跟踪，看能否发现更新的是哪张表，是哪些语句。
SQL> select * from (
  2  SELECT to_char(begin_interval_time, 'YYYY_MM_DD HH24:MI') snap_time,dhsso.object_name,SUM(db_block_changes_delta)
  3    FROM dba_hist_seg_stat dhss,dba_hist_seg_stat_obj dhsso,dba_hist_snapshot  dhs
  4   WHERE dhs.snap_id = dhss. snap_id
  5     AND dhs.instance_number = dhss. instance_number AND dhss.obj# = dhsso. obj# AND dhss.dataobj# = dhsso.dataobj#
  6     AND begin_interval_time> sysdate - 60/1440
  7   GROUP BY to_char(begin_interval_time, 'YYYY_MM_DD HH24:MI'), dhsso.object_name order by 3 desc)
  8  where rownum<=3;
SNAP_TIME        OBJECT_NAME                    SUM(DB_BLOCK_CHANGES_DELTA)
---------------- ------------------------------ ---------------------------
2013_11_13 20:00 TEST_REDO                                         178272
2013_11_13 20:00 MGMT_CURRENT_METRICS_PK                              224
2013_11_13 20:00 MGMT_SYSTEM_PERF_LOG_IDX_01                          160

SQL> SELECT to_char(begin_interval_time,'YYYY_MM_DD HH24:MI'),dbms_lob.substr(sql_text,4000,1),dhss.sql_id,executions_delta,rows_processed_delta
  2    FROM dba_hist_sqlstat dhss, dba_hist_snapshot dhs, dba_hist_sqltext dhst
  3   WHERE UPPER(dhst.sql_text) LIKE '%TEST_REDO%' AND dhss.snap_id = dhs.snap_id
  4    AND dhss.instance_Number = dhs.instance_number AND dhss.sql_id = dhst.sql_id;
TO_CHAR(BEGIN_IN   DBMS_LOB.SUBSTR(SQL_TEXT,4000,1)              INSTANCE_NUMBER SQL_ID  EXECUTIONS_DELTA   ROWS_PROCESSED_DELTA
-----------------------------------------------------------------------------------------------------------------------------
2013_11_13 17:00   create table test_redo as select * from dba_objects 1      dsf2uj3pzzadq       1                72884
2013_11_13 20:00   insert into test_redo select * from test_redo       1      5w8pb7t27c85n       5              2259404
```

本周课程总结

本周课程总结



炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间