SQL优化冰与火

DTCC

2015中国数据库技术大会

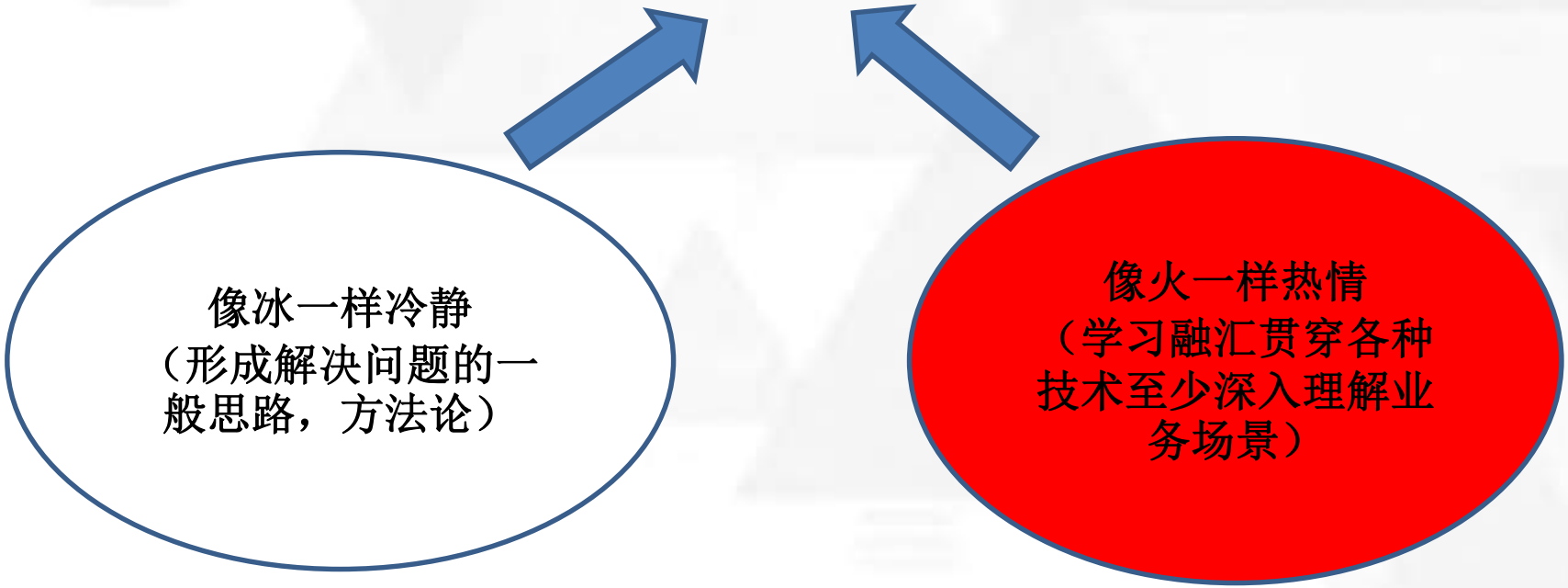
DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

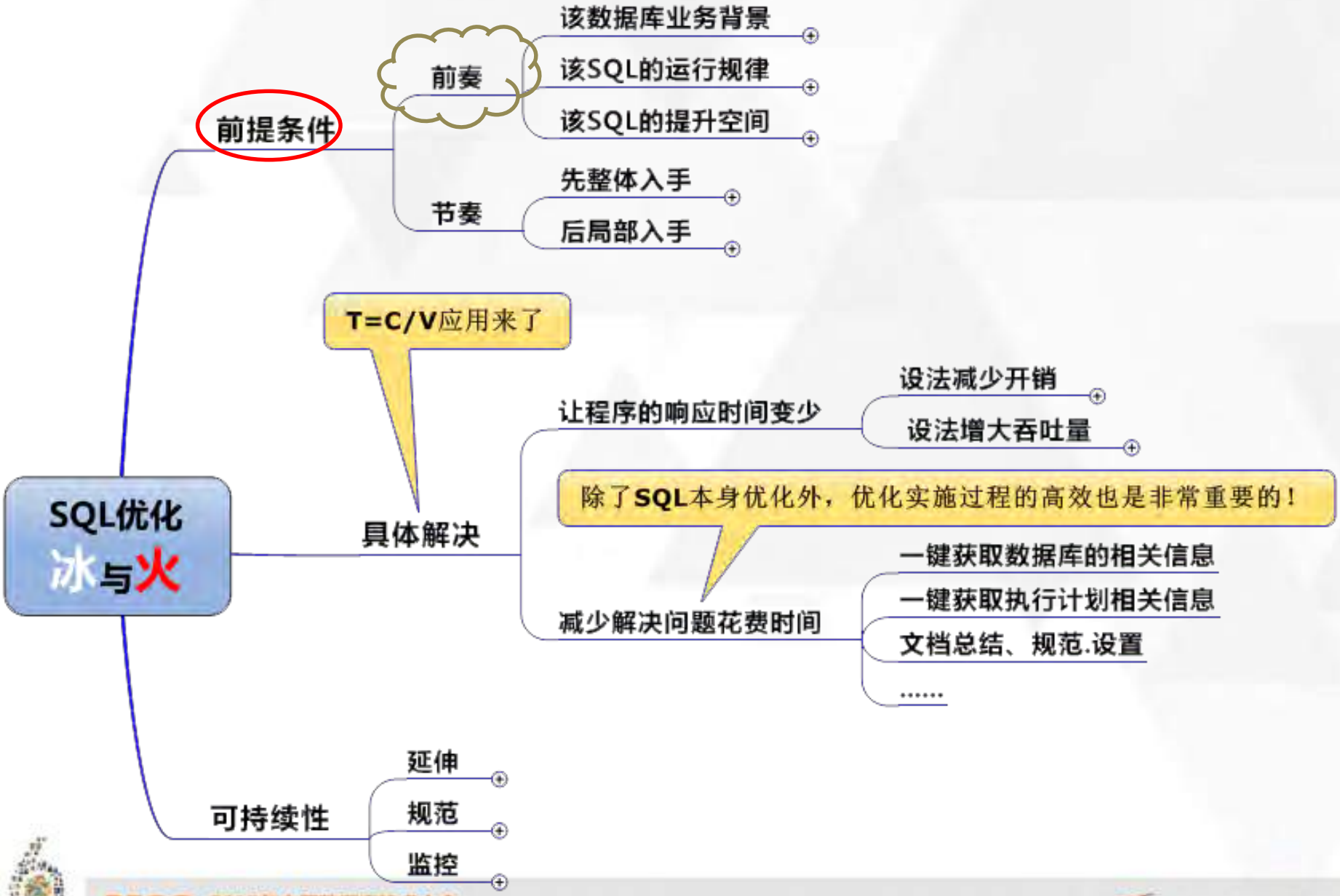
About me

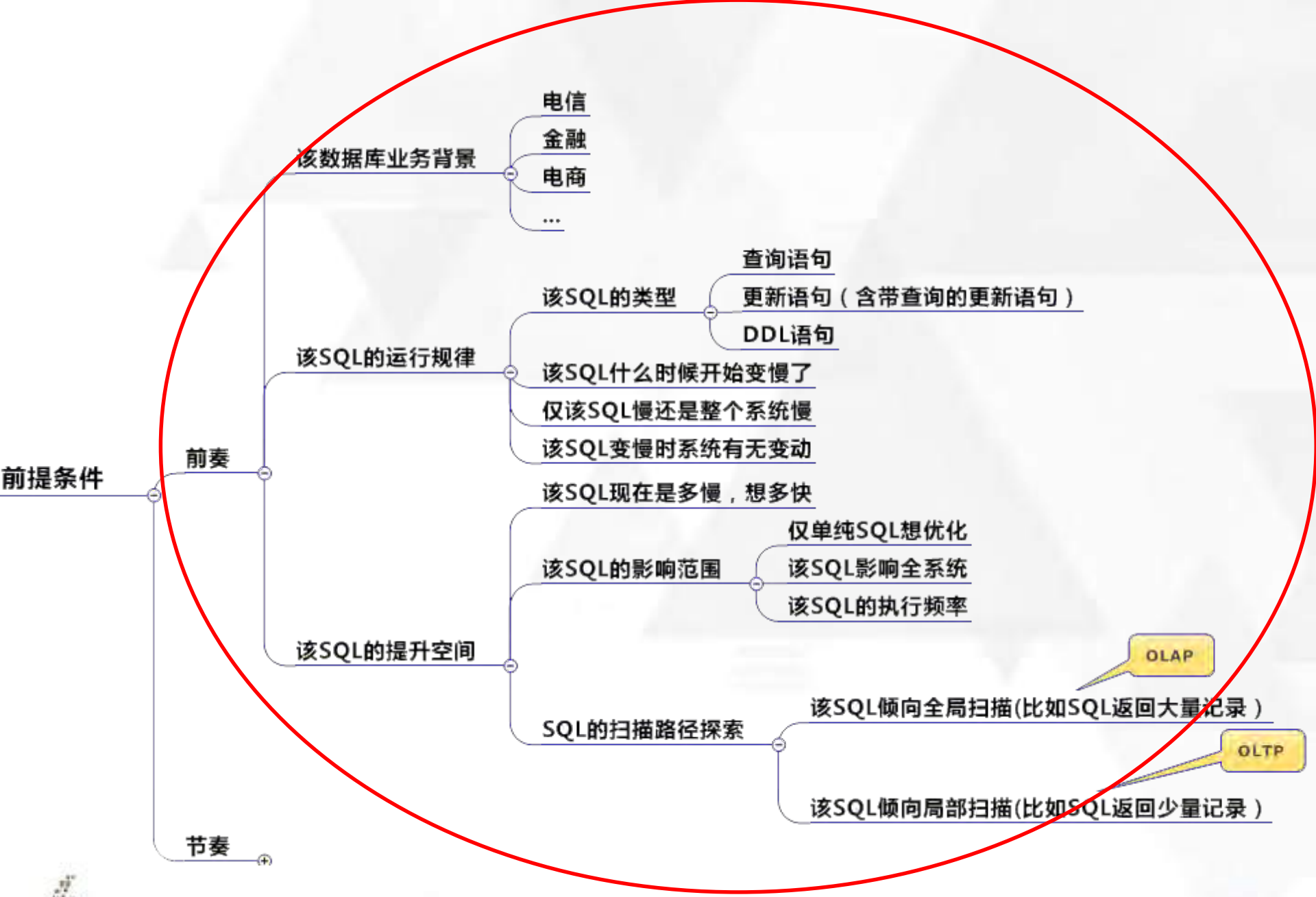
梁敬彬： 福富研究院副理事长、福富软件特级专家

SQL优化冰与火



SQL优化的前提条件之前奏




前提条件
前奏
该数据库业务背景
电信
金融
电商
...
该SQL的运行规律
该SQL的类型
查询语句
更新语句（含带查询的更新语句）
DDL语句
该SQL什么时候开始变慢了
仅该SQL慢还是整个系统慢
该SQL变慢时系统有无变动
该SQL的提升空间
该SQL现在是多慢，想多快
该SQL的影响范围
仅单纯SQL想优化
该SQL影响全系统
该SQL的执行频率
SQL的扫描路径探索
OLAP
该SQL倾向全局扫描(比如SQL返回大量记录）
OLTP
该SQL倾向局部扫描(比如SQL返回少量记录）
节奏

运行的规律是什么有多重要？

太重要了，总结就是三个字：

提升的空间有多大？

- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	53,680,378	30.3	2.35
CPU Time (ms)	47,654,130	26.2	3.66
Executions	10000000		
Buffer Gets	1000000000	100	14.18
Disk Reads	657,265	0.95	0.03
Parse Calls	0	0.00	0.00
Rows	92,571,398	133.26	
User I/O Wait Time (ms)	381,730		
Cluster Wait Time (ms)	201,812		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	1,332		
Invalidations	0		
Version Count	513		
Sharable Mem(KB)	42,185		

这有啥想法呢？

毫无PS痕迹！

提升的空间有多大？

```
select  * from t7 where id=1;
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
     142866  consistent gets
     142858  physical reads
          0  redo size
       1533  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
```

总共才返回1条，非常少，调优空间就非常大了！

```
select * from t7 where id>1;

已选择999999行。

已用时间:  00: 01: 30.25

执行计划
----------------------------------------------------------
Plan hash value: 3979761704

----------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |   797K|   401M| 58918   (1)| 00:07:48 |
|*  1 |  TABLE ACCESS FULL| T7   |   797K|   401M| 58918   (1)| 00:07:48 |
----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ID">1)

Note
-----
   - dynamic sampling used for this statement

统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
    2000000  consistent gets
    1428000  physical reads
          0  redo size
   23407731  bytes sent via SQL*Net to client
     733742  bytes received via SQL*Net from client
      66668  SQL*Net roundtrips to/from cli
          0  sorts (memory)
          0  sorts (disk)
```

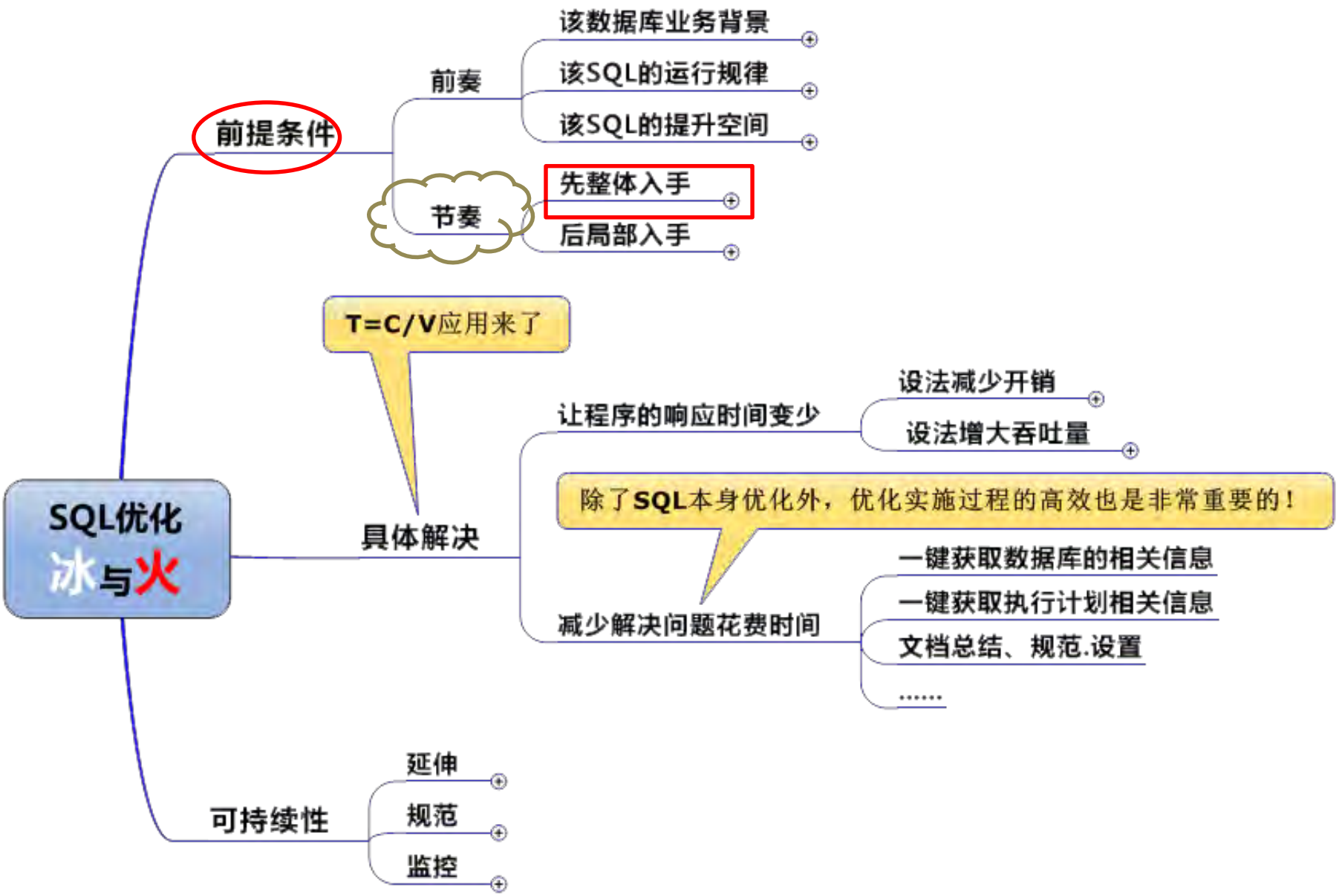
返回了近100万条，几乎所有的记录，调优空间就不大了！

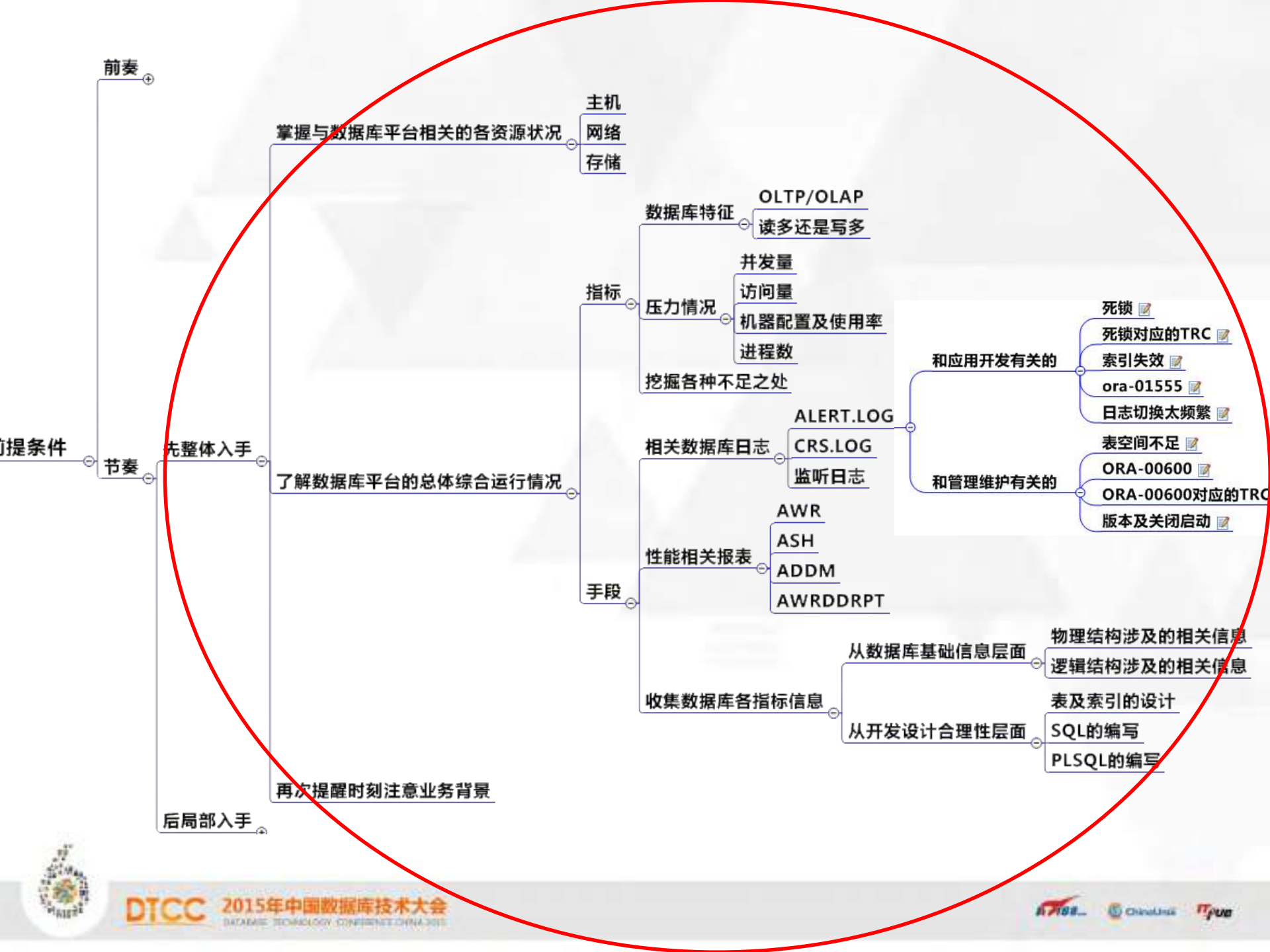
```
Plan hash value: 1909018627

-------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name          | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  | OMem | 1Mem | Used-Me
-------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |               |      1 |        |     15 |00:00:55.91 |   24688 |  15429 |      |      |
|   1 |  VIEW                          |               |      1 |  23549 |     15 |00:00:55.91 |   24688 |  15429 |      |      |
|   2 |   WINDOW SORT                  |               |      1 |  23549 |     15 |00:00:55.91 |   24688 |  15429 |  9216 |  9216 | 01
|*  3 |    HASH JOIN                   |               |      1 |  23549 |     15 |00:00:55.91 |   24688 |  15429 |  806K|  806K|
|*  4 |     TABLE ACCESS FULL          | WORKACCEPT_CFG |      1 |     24 |     24 |00:00:00.01 |      62 |      0 |      |      |
|*  5 |     HASH JOIN                  |               |      1 |  23549 |     15 |00:00:55.91 |   24626 |  15429 | 1452K|  1
|   6 |      TABLE ACCESS FULL         | GRADE         |      1 |      2 |      2 |00:00:00.01 |       2 |      0 |      |      |
|   7 |      NESTED LOOPS              |               |      1 |  23549 |     15 |00:00:55.91 |   24623 |  15429 |      |      |
|   8 |       TABLE ACCESS BY INDEX ROWID| STAFF_EVENT_HIS |      1 |  23549 |   6264 |00:00:27.53 |    5564 |   5369 |      |      |
|*  9 |        INDEX RANGE SCAN        | STAFF_ID_INDEX |      1 |  23658 |   6264 |00:00:00.05 |      16 |      15 |      |      |
|* 10 |       TABLE ACCESS BY INDEX ROWID| EVENT_Q_HIS   |   6264 |      1 |     15 |00:00:58.36 |   19059 |  10060 |      |      |
|* 11 |        INDEX UNIQUE SCAN       | PK_EVENT_Q_HIS |   6265 |      1 |   6264 |00:00:28.19 |   12795 |  9[illegible] |      |      |
-------------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("E"."EVENT_TYPE"="F"."WORK_TYPE")
   4 - filter("F"."QUEUE_TYPE"='EVENT_Q')
   5 - access("E"."GRADE"="D"."GRADE")
   9 - access("A"."STAFF_ID"=121)
  10 - filter(("B"."FLAG" IS NULL AND "B"."STATE_DATE">TRUNC(SYSDATE@!)-90))
  11 - access("A"."EVENT_ID"="B"."EVENT_ID")
```

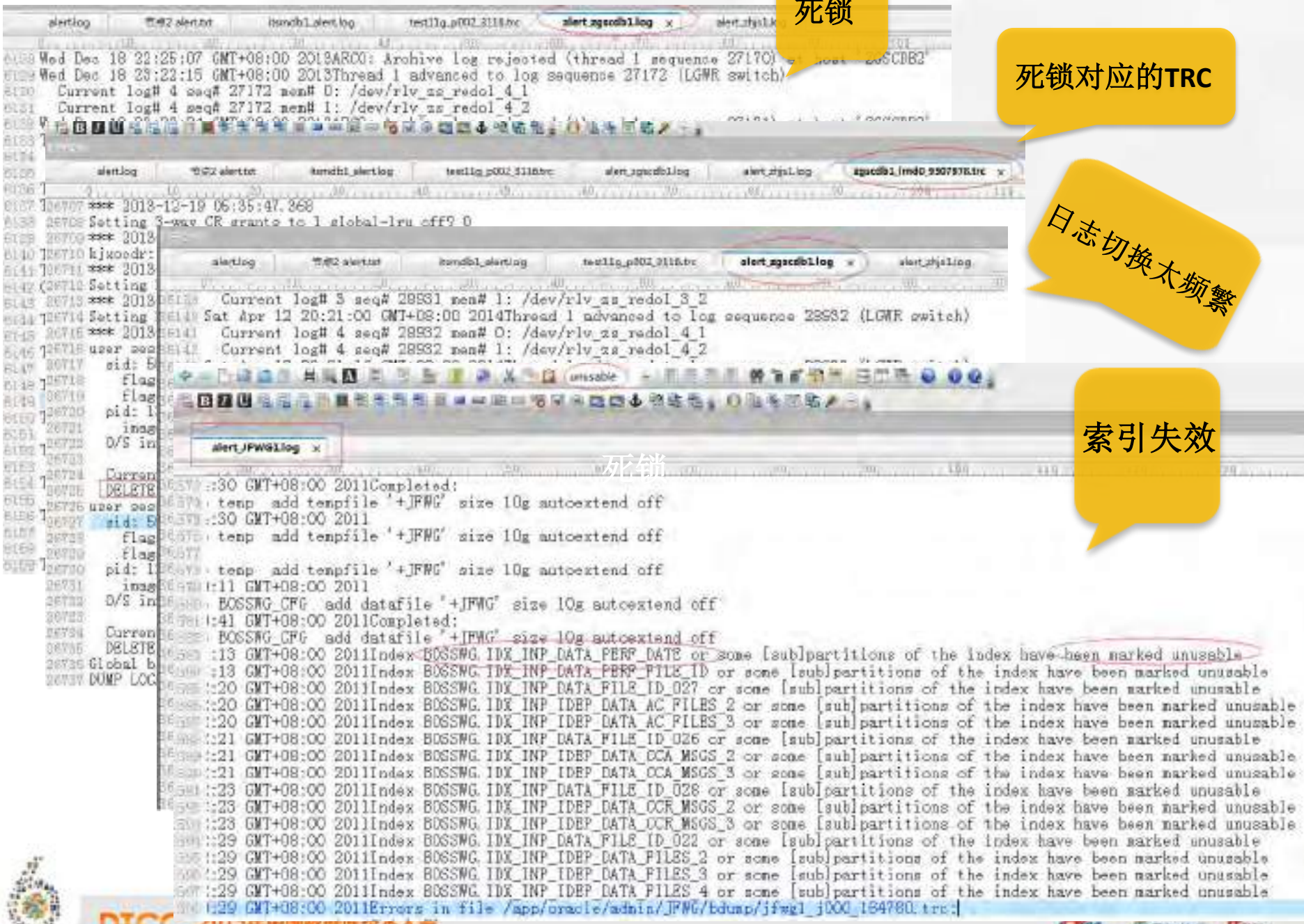
总共返回条数很少，所以调优空间很大！

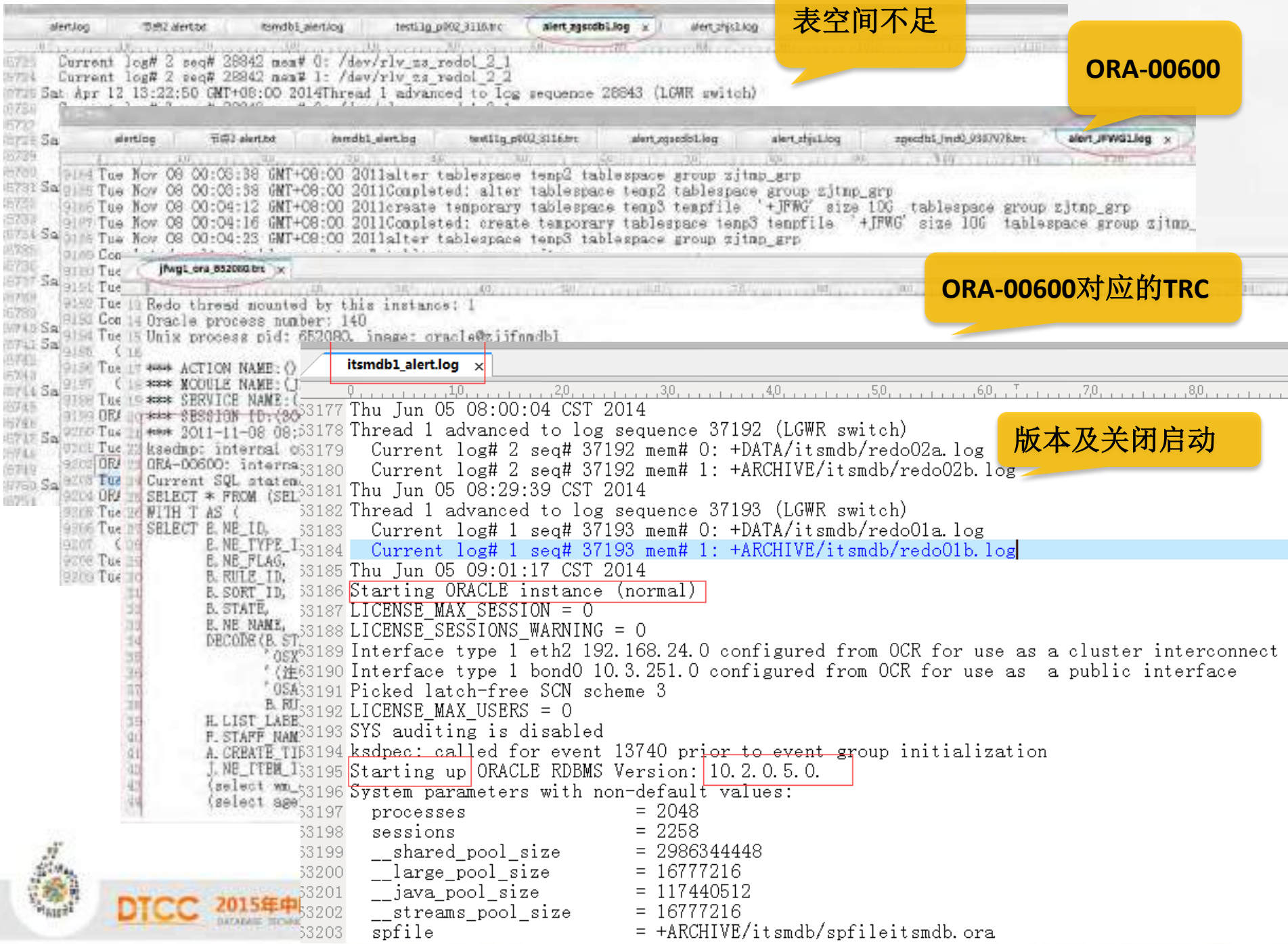
不过，这个例子其实有点特殊哦，这里还是有很多故事的，这个故事在后面慢慢和大家说。

SQL优化的前提条件之节奏（整体入手）




前奏
节奏
先整体入手
掌握与数据库平台相关的各资源状况
主机
网络
存储
了解数据库平台的总体综合运行情况
指标
数据库特征
OLTP/OLAP
读多还是写多
压力情况
并发量
访问量
机器配置及使用率
进程数
挖掘各种不足之处
手段
相关数据库日志
ALERT.LOG
CRS.LOG
监听日志
和应用开发有关的
死锁
死锁对应的TRC
索引失效
ora-01555
日志切换太频繁
和管理维护有关的
表空间不足
ORA-00600
ORA-00600对应的TRC
版本及关闭启动
性能相关报表
AWR
ASH
ADDM
AWRDDRPT
收集数据库各指标信息
从数据库基础信息层面
物理结构涉及的相关信息
逻辑结构涉及的相关信息
从开发设计合理性层面
表及索引的设计
SQL的编写
PLSQL的编写
再次提醒时刻注意业务背景
后局部入手


死锁
死锁对应的TRC
日志切换太频繁
索引失效
alert_zgscdb1.log
alert_JFWG1.log
Wed Dec 18 22:25:07 GMT+08:00 2013ARC0: Archive log rejected (thread 1 sequence 27170)
Wed Dec 18 23:23:15 GMT+08:00 2013Thread 1 advanced to log sequence 27172 (LGWR switch)
Current log# 4 seq# 27172 mem# 0: /dev/rlv_zs_redo1_4_1
Current log# 4 seq# 27172 mem# 1: /dev/rlv_zs_redo1_4_2
Current log# 5 seq# 28931 mem# 1: /dev/rlv_zs_redo1_5_2
Sat Apr 12 20:21:00 GMT+08:00 2014Thread 1 advanced to log sequence 28932 (LGWR switch)
Current log# 4 seq# 28932 mem# 0: /dev/rlv_zs_redo1_4_1
Current log# 4 seq# 28932 mem# 1: /dev/rlv_zs_redo1_4_2
:30 GMT+08:00 2011Completed:
temp add tempfile '+JFWG' size 10g autoextend off
:30 GMT+08:00 2011
temp add tempfile '+JFWG' size 10g autoextend off
temp add tempfile '+JFWG' size 10g autoextend off
:11 GMT+08:00 2011
BOSSWG_CFG add datafile '+JFWG' size 10g autoextend off
:41 GMT+08:00 2011Completed:
BOSSWG_CFG add datafile '+JFWG' size 10g autoextend off
:13 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_PERF_DATE or some [sub]partitions of the index have been marked unusable
:13 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_PERF_FILE_ID or some [sub]partitions of the index have been marked unusable
:20 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_FILE_ID_027 or some [sub]partitions of the index have been marked unusable
:20 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_AC_FILES_2 or some [sub]partitions of the index have been marked unusable
:20 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_AC_FILES_3 or some [sub]partitions of the index have been marked unusable
:21 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_FILE_ID_026 or some [sub]partitions of the index have been marked unusable
:21 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_CCA_MSGS_2 or some [sub]partitions of the index have been marked unusable
:21 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_CCA_MSGS_3 or some [sub]partitions of the index have been marked unusable
:23 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_FILE_ID_028 or some [sub]partitions of the index have been marked unusable
:23 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_CCR_MSGS_2 or some [sub]partitions of the index have been marked unusable
:23 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_CCR_MSGS_3 or some [sub]partitions of the index have been marked unusable
:29 GMT+08:00 2011Index BOSSWG.IDX_INP_DATA_FILE_ID_022 or some [sub]partitions of the index have been marked unusable
:29 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_FILES_2 or some [sub]partitions of the index have been marked unusable
:29 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_FILES_3 or some [sub]partitions of the index have been marked unusable
:29 GMT+08:00 2011Index BOSSWG.IDX_INP_IDEP_DATA_FILES_4 or some [sub]partitions of the index have been marked unusable
:29 GMT+08:00 2011Errors in file /app/oracle/admin/JFWG/bdump/jfwg1_j000_184780.trc:


表空间不足
ORA-00600
ORA-00600对应的TRC
版本及关闭启动
alert_zgscdb1.log
alert_JFWG1.log
itsmdb1_alert.log
Current log# 2 seq# 28842 mem# 0: /dev/rlv_zs_redo1_2_1
Current log# 2 seq# 28842 mem# 1: /dev/rlv_zs_redo1_2_2
Sat Apr 12 13:22:50 GMT+08:00 2014Thread 1 advanced to log sequence 28843 (LGWR switch)
Tue Nov 08 00:03:38 GMT+08:00 2011alter tablespace temp2 tablespace group zjtmp_grp
Tue Nov 08 00:03:38 GMT+08:00 2011Completed: alter tablespace temp2 tablespace group zjtmp_grp
Tue Nov 08 00:04:12 GMT+08:00 2011create temporary tablespace temp3 tempfile '+JFWG' size 10G tablespace group zjtmp_grp
Tue Nov 08 00:04:16 GMT+08:00 2011Completed: create temporary tablespace temp3 tempfile '+JFWG' size 10G tablespace group zjtmp_
Tue Nov 08 00:04:23 GMT+08:00 2011alter tablespace temp3 tablespace group zjtmp_grp
Redo thread mounted by this instance: 1
Oracle process number: 140
Unix process pid: 652080, image: oracle@zjjfnmdb1
Thu Jun 05 08:00:04 CST 2014
Thread 1 advanced to log sequence 37192 (LGWR switch)
Current log# 2 seq# 37192 mem# 0: +DATA/itsmdb/redo02a.log
Current log# 2 seq# 37192 mem# 1: +ARCHIVE/itsmdb/redo02b.log
Thu Jun 05 08:29:39 CST 2014
Thread 1 advanced to log sequence 37193 (LGWR switch)
Current log# 1 seq# 37193 mem# 0: +DATA/itsmdb/redo01a.log
Current log# 1 seq# 37193 mem# 1: +ARCHIVE/itsmdb/redo01b.log
Thu Jun 05 09:01:17 CST 2014
Starting ORACLE instance (normal)
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Interface type 1 eth2 192.168.24.0 configured from OCR for use as a cluster interconnect
Interface type 1 bond0 10.3.251.0 configured from OCR for use as a public interface
Picked latch-free SCN scheme 3
LICENSE_MAX_USERS = 0
SYS auditing is disabled
ksdpec: called for event 13740 prior to event group initialization
Starting up ORACLE RDBMS Version: 10.2.0.5.0.
System parameters with non-default values:
processes = 2048
sessions = 2258
__shared_pool_size = 2986344448
__large_pool_size = 16777216
__java_pool_size = 117440512
__streams_pool_size = 16777216
spfile = +ARCHIVE/itsmdb/spfileitsmdb.ora
DTCC 2015年中

Load Profile

	Per Second	Per Transaction
DB Time(s):	1.0	0.0
DB CPU(s):	0.9	0.0
Redo size:	3,527,300.9	520.5
Logical reads:	27,325.2	4.0
Block changes:	27,351.0	4.0
Physical reads:	1.9	0.0
Physical writes:	148.3	0.0
User calls:	1.5	0.0
Parses:	31.6	0.0
Hard parses:	0.3	0.0
W/A MB processed:	1.7	0.0
Logons:	0.5	0.0
Executes:	6,845.1	1.0
Rollbacks:	0.0	0.0
Transactions:	6,777.1	

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.2	0.4	0.00	1.31
DB CPU(s):	0.2	0.4	0.00	1.28
Redo size:	1,099,405.6	1,883,789.2		
Logical reads:	4,891.1	8,380.8		
Block changes:	9,140.0	15,661.0		
Physical reads:	0.2	0.3		
Physical writes:	0.1	0.2		
User calls:	0.2	0.3		
Parses:	10.2	17.5		
Hard parses:	0.4	0.6		
W/A MB processed:	1.1	1.9		
Logons:	0.0	0.0		
Executes:	4,503.9	7,717.3		
Rollbacks:	0.0	0.0		
Transactions:	0.6			

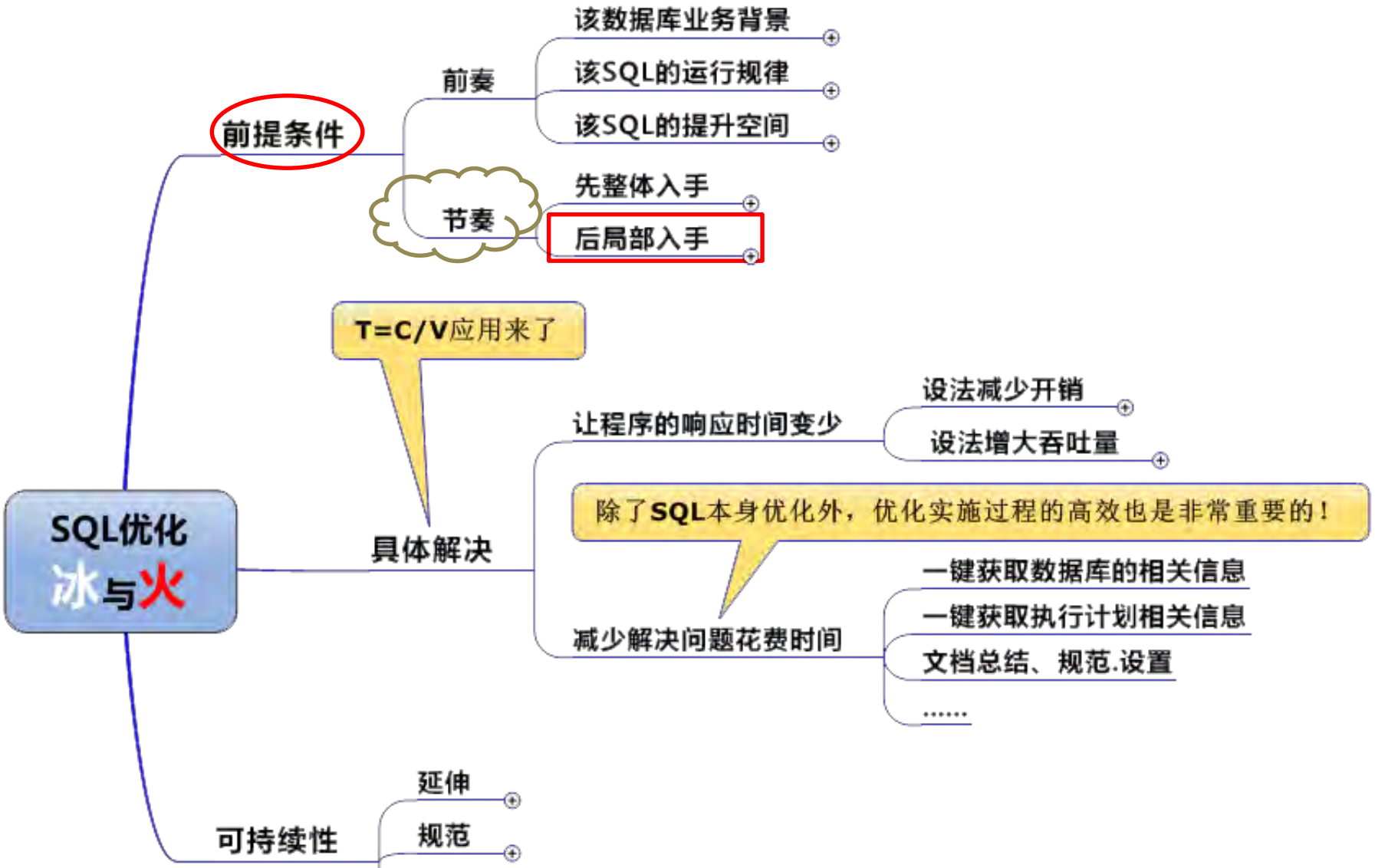
Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
enq: TX - row lock contention	3	279	93010	99.01	Application
DB CPU					
control file sequential read	141				
db file scattered read	20				
db file sequential read	31				

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
log file switch (checkpoint incomplete)	79	75	952	61.81	Configuration
DB CPU		42		34.71	
log file switch completion	39	4	100	3.20	Configuration
db file sequential read	142	0	0	0.03	User I/O
control file sequential read	141	0	0	0.03	System I/O

SQL优化的前提条件之节奏（局部入手）




前提条件
前奏
节奏
先整体入手
后局部入手
快速发现SQL问题
响应时间入手分析
快速获取SQL相关信息
各种执行计划获取
explain plan for
set autotrace on
statitstics_level=all
10046 trace
v$sql_plan
awrsqrpt.sql
SQL涉及的表的信息获取
表的大小和及表类型
对应索引大小及类型
表和索引的统计信息
表的列的类型
表的直方图
犀利识别执行计划好坏
来，看看这两个执行计划谁好
去，学会如何看执行计划好坏
其他SQL优化关联知识
高级SQL编写
执行计划解读
模型优化设计
.....

识别执行计划的好坏

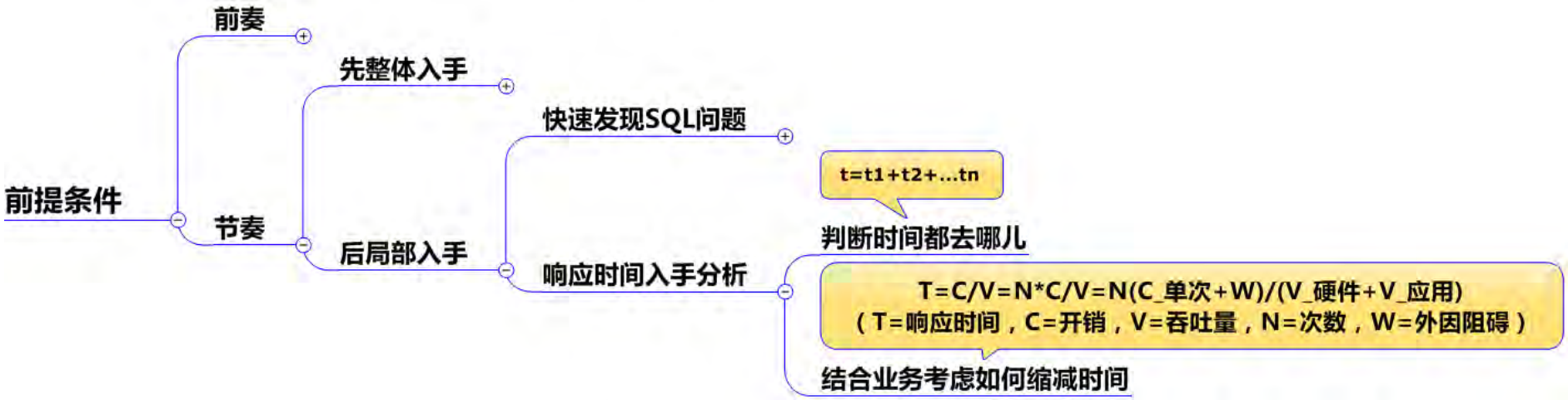
```
SELECT count(t2.col2)  FROM t1 ,t2   WHERE t1.id=t2.id and t1.col1 = 666;
```

执行计划1

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
1	SORT AGGREGATE		1	1	1	00:00:03.32	157K
2	**NESTED LOOPS**		1	32	75808	00:00:03.56	157K
3	TABLE ACCESS BY INDEX ROWID	T1	1	32	80016	00:00:01.44	1770
* 4	INDEX RANGE SCAN	T1_COL1	1	32	80016	00:00:00.32	169
5	TABLE ACCESS BY INDEX ROWID	T2	80016	1	75808	00:00:02.68	155K
* 6	INDEX UNIQUE SCAN	T2_PK	80016	1	75808	00:00:01.11	80018

执行计划2

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
1	SORT AGGREGATE		1	1	1	00:00:00.23	5174
* 2	**HASH JOIN**		1	80000	75808	00:00:01.08	5174
* 3	TABLE ACCESS FULL	T1	1	80000	80016	00:00:00.48	2644
4	TABLE ACCESS FULL	T2	1	151K	151K	00:00:00.61	2530


前提条件
前奏
节奏
先整体入手
后局部入手
快速发现SQL问题
响应时间入手分析
t=t1+t2+...tn
判断时间都去哪儿
T=C/V=N*C/V=N(C_单次+W)/(V_硬件+V_应用)
（T=响应时间，C=开销，V=吞吐量，N=次数，W=外因阻碍）
结合业务考虑如何缩减时间

$T = C / V$

注释：开销要少，吞吐量要大

$= (N * C) / V$

注释：开销要少（分解为单次开销和执行次数），吞吐量要大

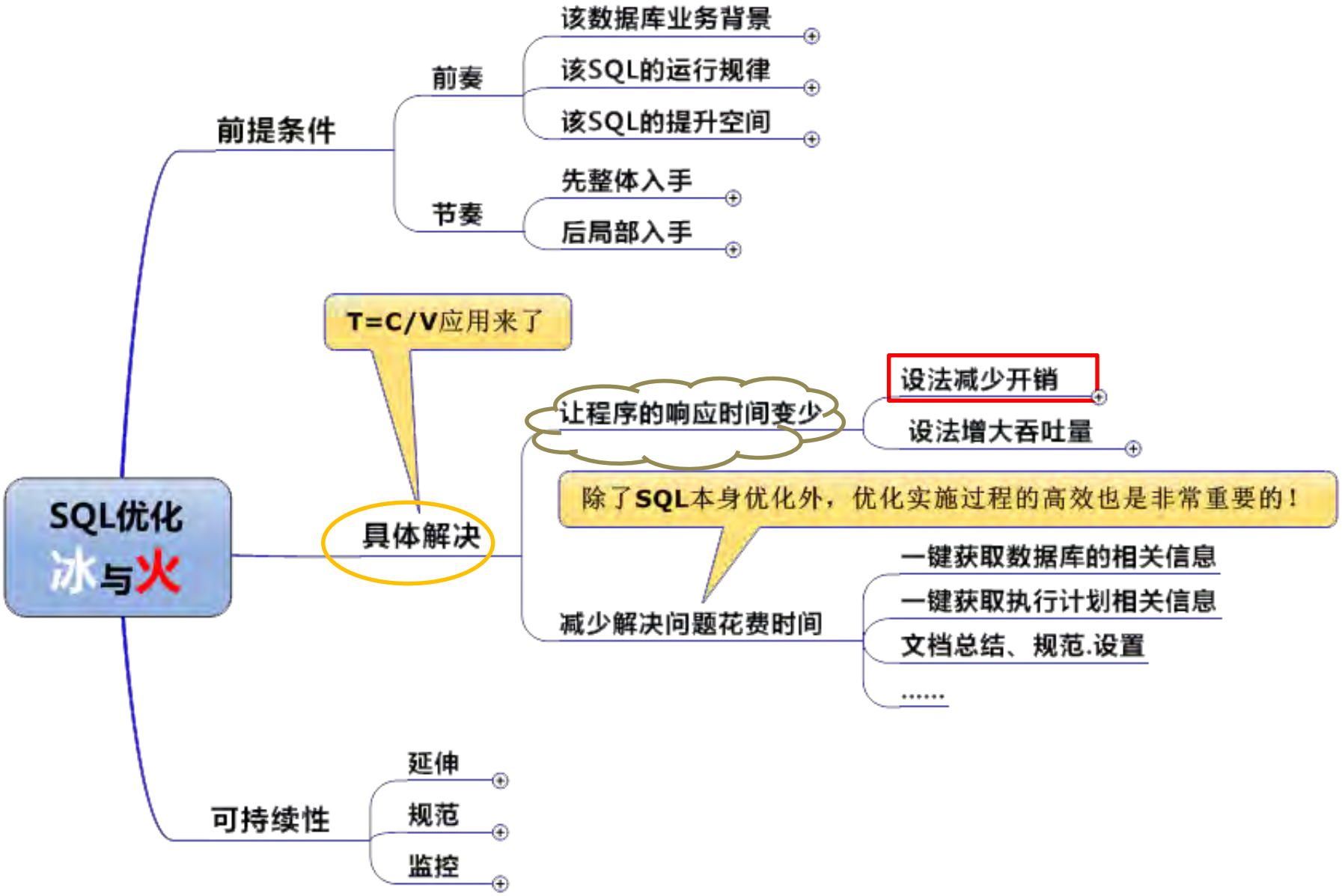
$= N * (C\_单次 + W) / (V\_硬件 + V\_应用)$

注释：你想要执行时间快快的，那你得考虑这些方案：执行次数能少则少*（单次访问路径能短就短+各种等待能没就没）/（硬件吞吐量能棒则棒+应用吞吐量能大则大）。这样你就保证了开销小吞吐量大的原则，你就赢了。

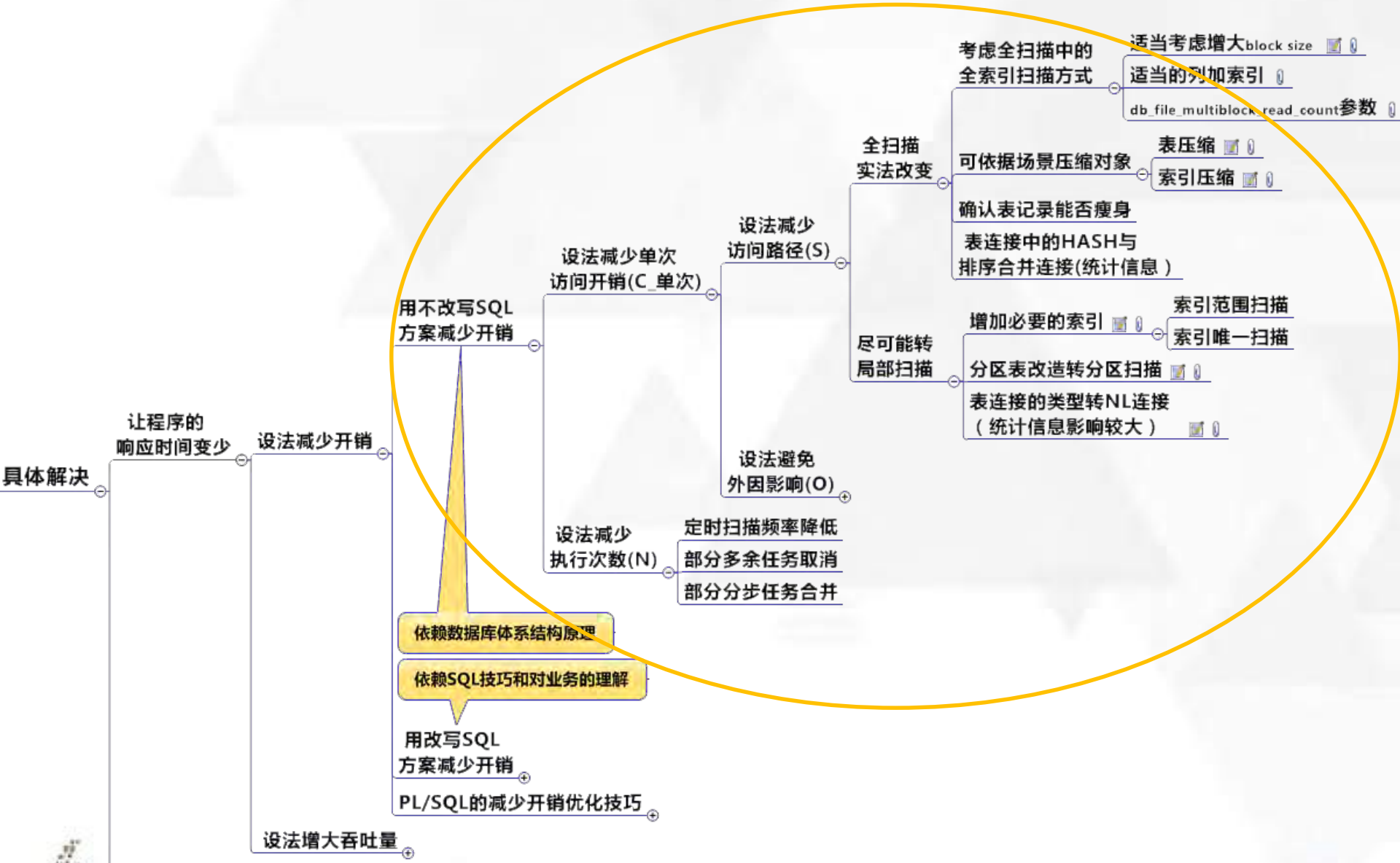
（T=响应时间，C=开销，V=吞吐量，N=次数，W=外因阻碍）

接下来，具体解决问题的思路就横空出世了！

SQL优化具体解决之响应时间分析(减少开销)



具体解决之减少开销的不改写SQL



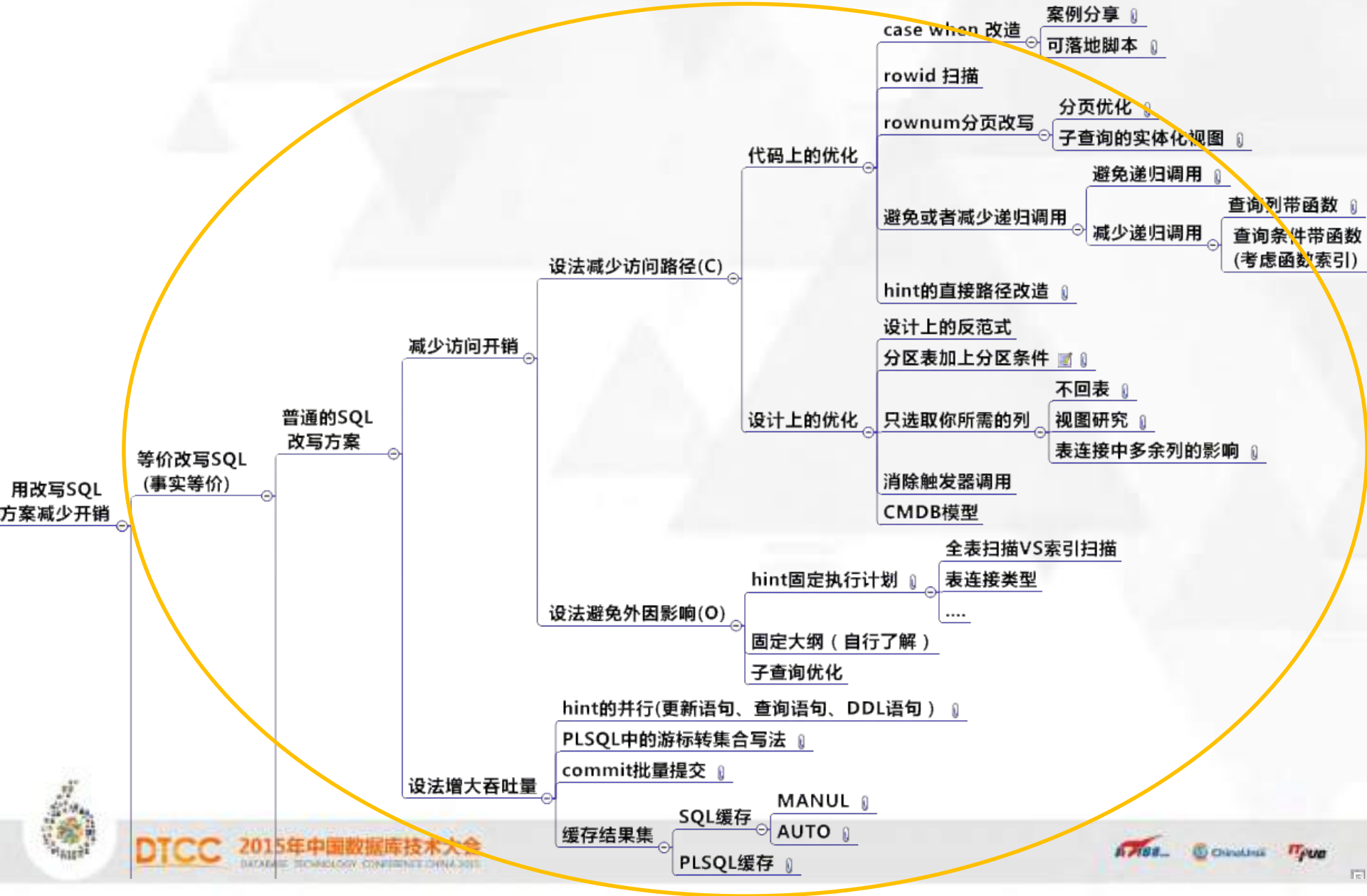
```
select count(*) from t;
执行计划
----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |   292   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE    |      |     1 |            |          |
|   2 |   TABLE ACCESS FULL| T    | 69485 |   292   (1)| 00:00:04 |
----------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1048  consistent gets
```

```
select count(*) from t;

执行计划
----------------------------------------------------------------------------
| Id  | Operation          | Name          | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |               |     1 |    46   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |               |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| PK1_OBJECT_ID | 69485 |    46   (0)| 00:00:01 |
----------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        160  consistent gets
```

```
select max(object_id) from t;
执行计划
----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |     1 |    13 |     2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                |     1 |    13 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)  | PK_OBJECT_ID   |     1 |    13 |     2   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          2  consistent gets
```

```
select * from t where object_id>2 order by object_id;
执行计划
----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                | 92407 |   18M |  1302   (1)| 00:00:16 |
|   1 |  TABLE ACCESS BY INDEX ROWID | T              | 92407 |   18M |  1302   (1)| 00:00:16 |
| * 2 |   INDEX RANGE SCAN           | IDX_T_OBJECT_ID| 92407 |       |   177   (1)| 00:00:03 |
----------------------------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
      10952  consistent gets
          0  physical reads
          0  redo size
    8115221  bytes sent via SQL*Net to client
      54029  bytes received via SQL*Net from client
       4876  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      73117  rows processed
```

具体解决之减少开销的改写SQL(普通)



举个CASE WHEN优化表访问次数的案例1

```
select t1.object_name,
       t1.object_id,
       (select count(*)
          from t2
         where temporary = 'Y'
           and t2.object_id = t1.object_id) CNT_TEMPORARY_Y,
       (select count(*)
          from t2
         where created >=sysdate-365
           and t2.object_id = t1.object_id) CNT_CREATED_NEW,
       (select sum(object_id)
          from t2
         where status <> 'VALID'
           and t2.object_id = t1.object_id) SUM_OBJID_STATUS_V,
       (select sum(object_id)
          from t2
         where generated = 'Y'
           and t2.object_id = t1.object_id)  SUM_OBJID_GENERATED_Y,
       (select sum(object_id)
          from t2
         where generated = 'N'
           and t2.object_id = t1.object_id)  SUM_OBJID_GENERATED_M,
       (select sum(object_id)
          from t2
         where generated = 'Q'
           and t2.object_id = t1.object_id)  SUM_OBJID_GENERATED_Q
  from t1
 where t1.object_id <= 50;
```

```
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT    |      |    47 |  1410 |   285   (1)| 00:00:04 |
|   1 |  SORT AGGREGATE     |      |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL | T2   |     1 |     7 |   284   (1)| 00:00:04 |
|   3 |  SORT AGGREGATE     |      |     1 |    13 |            |          |
|*  4 |   TABLE ACCESS FULL | T2   |     1 |    13 |   284   (1)| 00:00:04 |
|   5 |  SORT AGGREGATE     |      |     1 |    12 |            |          |
|*  6 |   TABLE ACCESS FULL | T2   |     1 |    13 |   284   (1)| 00:00:04 |
|   7 |  SORT AGGREGATE     |      |     1 |     7 |            |          |
|*  8 |   TABLE ACCESS FULL | T2   |     1 |     7 |   284   (1)| 00:00:04 |
|   9 |  SORT AGGREGATE     |      |     1 |     7 |            |          |
|* 10 |   TABLE ACCESS FULL | T2   |     1 |     7 |   284   (1)| 00:00:04 |
|  11 |  SORT AGGREGATE     |      |     1 |     7 |            |          |
|* 12 |   TABLE ACCESS FULL | T2   |     1 |     7 |   284   (1)| 00:00:04 |
|* 13 |  TABLE ACCESS FULL  | T1   |    47 |  1410 |   285   (1)| 00:00:04 |

Predicate Information (identified by operation id):

   2 - filter("T2"."OBJECT_ID"=:B1 AND "TEMPORARY"='Y')
   4 - filter("T2"."OBJECT_ID"=:B1 AND "CREATED">=SYSDATE@!-365)
   6 - filter("T2"."OBJECT_ID"=:B1 AND "STATUS"<>'VALID')
   8 - filter("T2"."OBJECT_ID"=:B1 AND "GENERATED"='Y')
  10 - filter("T2"."OBJECT_ID"=:B1 AND "GENERATED"='N')
  12 - filter("T2"."OBJECT_ID"=:B1 AND "GENERATED"='Q')
  13 - filter("T1"."OBJECT_ID"<=50)

统计信息

          0  recursive calls
          0  db block gets
     303904  consistent gets
          0  physical reads
          0  redo size
       2378  bytes sent via SQL*Net to client
        649  bytes received via SQL*Net from client
          5  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         49  rows processed
```

```
with w_t2 as
(select
       t2.object_id,
       count(case when t2.temporary='Y' then 1 end ) CNT_TEMPORARY_Y,
       count(case when created >=sysdate-365  then 1 end ) CNT_CREATED_NEW,
       sum(case when t2.status<>'VALID' then t2.object_id end ) SUM_OBJID_STATUS_V,
       sum(case when t2.generated = 'Y' then t2.object_id end ) SUM_OBJID_GENERATED_Y,
       sum(case when t2.generated = 'M' then t2.object_id end ) SUM_OBJID_GENERATED_M,
       sum(case when t2.generated = 'Q' then t2.object_id end ) SUM_OBJID_GENERATED_Q
 from  t2
 group by t2.object_id)
select t1.object_name,t1.object_id,w_t2.* from t1,w_t2
where t1.object_id=w_t2.object_id
and t1.object_id<=50;
```

执行计划

```
Plan hash value: 3226031139

| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT    |      |    47 |  6815 |   572   (1)| 00:00:07 |
|   1 |  HASH GROUP BY      |      |    47 |  6815 |   572   (1)| 00:00:07 |
|*  2 |   HASH JOIN         |      |    47 |  6815 |   571   (1)| 00:00:07 |
|*  3 |    TABLE ACCESS FULL| T1   |    47 |  3384 |   285   (1)| 00:00:04 |
|*  4 |    TABLE ACCESS FULL| T2   |    47 |  1128 |   285   (1)| 00:00:04 |

Predicate Information (identified by operation id):

   2 - access("T1"."OBJECT_ID"="T2"."OBJECT_ID")
   3 - filter("T1"."OBJECT_ID"<=50)
   4 - filter("T2"."OBJECT_ID"<=50)

统计信息

          0  recursive calls
          0  db block gets
       2040  consistent gets
          0  physical reads
          0  redo size
       2633  bytes sent via SQL*Net to client
        649  bytes received via SQL*Net from client
          5  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         49  rows processed
```

举个CASE WHEN优化表访问次数的案例2

```
select  substr(object_id,1,2),count(*)  from t where object_type='INDEX'  group by substr(object_id,1,2)
union
select  substr(object_id,1,3),count(*)  from t  where object_type<>'INDEX' group by substr(object_id,1,3) ;
```

```
Plan hash value: 2965795838

| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT     |      | 83440 |  1955K|       |  1130   (7)| 00:00:14 |
|   1 |  SORT UNIQUE         |      | 83440 |  1955K|  4520K|  1130   (7)| 00:00:14 |
|   2 |   UNION-ALL          |      |       |       |       |            |          |
|   3 |    HASH GROUP BY     |      |  8371 |   119K|       |   288   (3)| 00:00:04 |
|*  4 |     TABLE ACCESS FULL| T    |  8371 |   119K|       |   288   (1)| 00:00:04 |
|   5 |    HASH GROUP BY     |      | 75174 |  1612K|  4920K|   842   (2)| 00:00:11 |
|*  6 |     TABLE ACCESS FULL| T    | 75174 |  1612K|       |   284   (1)| 00:00:04 |

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter("OBJECT_TYPE"='INDEX')
   6 - filter("OBJECT_TYPE"<>'INDEX')
Note
-----
   - dynamic sampling used for this statement
统计信息
----------------------------------------------------------

          0  recursive calls
          0  db block gets
       2040  consistent gets
          0  physical reads
          0  redo size
       1704  bytes sent via SQL*Net to client
        460  bytes received via SQL*Net from client
          6  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
         73  rows processed
```

```
select substr(object_id,1,CASE WHEN object_type = 'INDEX' then 2 ELSE 3  end),count(*) from t
 group by substr(object_id,1,CASE WHEN object_type = 'INDEX' then 2 ELSE 3  end)  ;
```

```
select substr(object_id,1,CASE WHEN object_type = 'INDEX' then 2 ELSE 3  end),count(*) from t
 group by substr(object_id,1,CASE WHEN object_type = 'INDEX' then 2 ELSE 3  end)  ;
```

执行计划

```
----------------------------------------------------------
Plan hash value: 47235625

| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT   |      | 83445 |  1955K|   289   (2)| 00:00:04 |
|   1 |  HASH GROUP BY     |      | 83445 |  1955K|   289   (2)| 00:00:04 |
|   2 |   TABLE ACCESS FULL| T    | 83445 |  1955K|   285   (1)| 00:00:04 |

Note
-----
   - dynamic sampling used for this statement
```

统计信息

```
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1020  consistent gets
          0  physical reads
          0  redo size
       1859  bytes sent via SQL*Net to client
        460  bytes received via SQL*Net from client
          6  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         73  rows processed
```

举个rowid减少开销减少访问路径的例子

```
SQL> select /*+full(t)*/ * from t where object_id=2;
-------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |   207 |   291   (1)| 00:00:04 |
|*  1 |  TABLE ACCESS FULL| T    |     1 |   207 |   291   (1)| 00:00:04 |
-------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1044  consistent gets
```

```
SQL> select * from t where object_id=2;
-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                 |     1 |   207 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T               |     1 |   207 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IDX_OBJECT_ID   |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          4  consistent gets
```

最牛B的来了

```
SQL> select * from t where object_id=2 and rowid='AAAYiZAALAAAADLAAw';
-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |      |     1 |   219 |     1   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY USER ROWID| T    |     1 |   219 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          1  consistent gets
```

举个Rownum 和表连接相关的例子

```
-----略去
    select ta.tch_id, ta.flow_id
                from tache pr,
                     (select TCH_ID, c.flow_id
                        from event_q a, staff_event b, tache c
                       where (a.event_type = '2' OR a.event_type = '1')
                         and a.event_id = b.event_id
                         and (a.flag is null or a.flag = '1')
                         and b.staff_id = 1
                         and a.content_id = c.tch_id) ta
              where ta.flow_id = pr.sub_flow_id(+)
                and pr.flow_id is null
-------略去
```

```
|  31 |        NESTED LOOPS                |                |     1 |    53 |    63   (2)| 00:00:01 |
|  32 |         MERGE JOIN CARTESIAN       |                |     1 |    30 |    62   (2)| 00:00:01 |
|* 33 |          FILTER                    |                |       |       |            |          |
|* 34 |           HASH JOIN RIGHT OUTER    |                |     1 |    28 |    57   (2)| 00:00:01 |
|* 35 |            TABLE ACCESS FULL       | TACHE          |    10 |    90 |    28   (0)| 00:00:01 |
|  36 |            TABLE ACCESS FULL       | TACHE          |  7582 |   103K|    28   (0)| 00:00:01 |
|  37 |          BUFFER SORT               |                |    33 |  1045 |    34   (3)| 00:00:01 |
|* 38 |           TABLE ACCESS FULL        | STAFF_EVENT    |    33 |  1045 |     5   (0)| 00:00:01 |
|* 39 |         TABLE ACCESS BY INDEX ROWID | EVENT_Q        |     1 |    19 |     1   (0)| 00:00:01 |
|* 40 |          INDEX UNIQUE SCAN         | IDX_EVENT_Q_EVENTID |     1 |       |     0   (0)| 00:00:01 |
```

```
select ta.tch_id, ta.flow_id
            from tache pr,
                 (select TCH_ID, c.flow_id,rownum
                    from event_q a, staff_event b, tache c
                   where (a.event_type = '2' OR a.event_type = '1')
                     and a.event_id = b.event_id
                     and (a.flag is null or a.flag = '1')
                     and b.staff_id = 1
                     and a.content_id = c.tch_id) ta
          where ta.flow_id = pr.sub_flow_id(+)
            and pr.flow_id is null
```

```
|* 31 |        HASH JOIN OUTER              |                |     1 |    58 |   707   (1)| 00:00:09 |
|  32 |         NESTED LOOPS                |                |    65 |  3120 |   319   (1)| 00:00:04 |
|* 33 |          HASH JOIN                  |                |   190 |  6080 |   129   (3)| 00:00:02 |
|* 34 |           TABLE ACCESS FULL         | STAFF_EVENT    |   190 |  2280 |    16   (0)| 00:00:01 |
|* 35 |           TABLE ACCESS FULL         | EVENT_Q        |  5220 |   101K|   112   (2)| 00:00:02 |
|  36 |          TABLE ACCESS BY INDEX ROWID | TACHE          |     1 |    16 |     1   (0)| 00:00:01 |
|* 37 |           INDEX UNIQUE SCAN         | PK_TACHE       |     1 |       |     0   (0)| 00:00:01 |
|* 38 |         TABLE ACCESS FULL           | TACHE          |     1 |    11 |   388   (1)| 00:00:05 |
```

设计上的反范式（表结构+SQL改造的经典案例）

Plan hash value: 1983018627

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		15	00:00:55.91	24688	15429			
1	VIEW		1	23549	15	00:00:55.91	24688	15429			
2	WINDOW SORT		1	23549	15	00:00:55.91	24688	15429	9216	9216	8192 (0)
*3	HASH JOIN		1	23549	15	00:00:55.91	24688	15429	800K	800K	1093K (0)
*4	TABLE ACCESS FULL	WORKACCEPT_CFG	1	24	24	00:00:00.01	62	0			
*5	HASH JOIN		1	23549	15	00:00:55.91	24626	15429	1452K	1452K	893K (0)
6	TABLE ACCESS FULL	GRADE	1	3	3	00:00:00.01	3	0			
7	NESTED LOOPS		1	23549	15	00:00:55.91	24623	15429			
8	TABLE ACCESS BY INDEX ROWID	STAFF_EVENT_HIS	1	23549	6264	00:00:17.53	5564	5369			
*9	INDEX RANGE SCAN	STAFF_ID_INDEX	1	23658	6264	00:00:00.05	16	15			
*10	TABLE ACCESS BY INDEX ROWID	EVENT_Q_HIS	6264	1	15	00:00:38.36	19059	10060			
*11	INDEX UNIQUE SCAN	PK_EVENT_Q_HIS	6264	1	6264	00:00:24.15	12795	5449			

Predicate Information (identified by operation id):

```
   3 - access("B"."EVENT_TYPE"="F"."WORK_TYPE")
   4 - filter("F"."QUEUE_TYPE"='EVENT_Q')
   5 - access("B"."GRADE"="D"."GRADE")
   9 - access("A"."STAFF_ID"=121)
  10 - filter(("B"."FLAG" IS NULL AND "B"."STATE_DATE">TRUNC(SYSDATE@!)-30))
  11 - access("A"."EVENT_ID"="B"."EVENT_ID")
```

最终返回才15条，缘何要产生2万多个逻辑读，耗时55秒？这其中必有巨大的优化空间！

Plan hash value: 1983018627

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		15	00:00:55.91	24688	15429			
1	VIEW		1	23549	15	00:00:55.91	24688	15429			
2	WINDOW SORT		1	23549	15	00:00:55.91	24688	15429	9216	9216	8192 (0)
*3	HASH JOIN		1	23549	15	00:00:55.91	24688	15429	800K	800K	1093K (0)
*4	TABLE ACCESS FULL	WORKACCEPT_CFG	1	24	24	00:00:00.01	62	0			
*5	HASH JOIN		1	23549	15	00:00:55.91	24626	15429	1452K	1452K	893K (0)
6	TABLE ACCESS FULL	GRADE	1	3	3	00:00:00.01	3	0			
7	NESTED LOOPS		1	23549	15	00:00:55.91	24623	15429			
8	TABLE ACCESS BY INDEX ROWID	STAFF_EVENT_HIS	1	23549	6264	00:00:17.53	5564	5369			
*9	INDEX RANGE SCAN	STAFF_ID_INDEX	1	23658	6264	00:00:00.05	16	15			
*10	TABLE ACCESS BY INDEX ROWID	EVENT_Q_HIS	6264	1	15	00:00:38.36	19059	10060			
*11	INDEX UNIQUE SCAN	PK_EVENT_Q_HIS	6264	1	6264	00:00:24.15	12795	5449			

Predicate Information (identified by operation id):

```
   3 - access("B"."EVENT_TYPE"="F"."WORK_TYPE")
   4 - filter("F"."QUEUE_TYPE"='EVENT_Q')
   5 - access("B"."GRADE"="D"."GRADE")
   9 - access("A"."STAFF_ID"=121)
  10 - filter(("B"."FLAG" IS NULL AND "B"."STATE_DATE">TRUNC(SYSDATE@!)-30))
  11 - access("A"."EVENT_ID"="B"."EVENT_ID")
```

EVENT_Q_HIS表要索引访问6264次，难怪很慢！可是STAFF_ID=121返回的就是6264条，似乎没啥好办法。如果反过来，B.FLAG IS NULL AND B.STATE_DATE >TRUNC(SYSDATE@!)-30))的条件，返回20万，说明驱动顺序也没错。

.....接下来

Plan hash value: 1983018627

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		15	00:00:55.91	24688	15429			
1	VIEW		1	23549	15	00:00:55.91	24688	15429			
2	WINDOW SORT		1	23549	15	00:00:55.91	24688	15429	9216	9216	8192 (0)
*3	HASH JOIN		1	23549	15	00:00:55.91	24688	15429	800K	800K	1093K (0)
*4	TABLE ACCESS FULL	WORKACCEPT_CFG	1	24	24	00:00:00.01	62	0			
*5	HASH JOIN		1	23549	15	00:00:55.91	24626	15429	1452K	1452K	893K (0)
6	TABLE ACCESS FULL	GRADE	1	3	3	00:00:00.01	3	0			
7	NESTED LOOPS		1	23549	15	00:00:55.91	24623	15429			
8	TABLE ACCESS BY INDEX ROWID	STAFF_EVENT_HIS	1	23549	6264	00:00:17.53	5564	5369			
*9	INDEX RANGE SCAN	STAFF_ID_INDEX	1	23658	6264	00:00:00.05	16	15			
*10	TABLE ACCESS BY INDEX ROWID	EVENT_Q_HIS	6264	1	15	00:00:38.36	19059	10060			
*11	INDEX UNIQUE SCAN	PK_EVENT_Q_HIS	6264	1	6264	00:00:24.15	12795	5449			

Predicate Information (identified by operation id):

```
   3 - access("B"."EVENT_TYPE"="F"."WORK_TYPE")
   4 - filter("F"."QUEUE_TYPE"='EVENT_Q')
   5 - access("B"."GRADE"="D"."GRADE")
   9 - access("A"."STAFF_ID"=121)
  10 - filter(("B"."FLAG" IS NULL AND "B"."STATE_DATE">TRUNC(SYSDATE@!)-30))
  11 - access("A"."EVENT_ID"="B"."EVENT_ID")
```

奇怪，该如何优化呢？我所能想到最浪漫的事，就是这里别走NL用HASH了。不过忽然发现不对，这个NL后最终只有返回15条，没错，你没看错，真的是15条！
这里坑爹的事来了，驱动表的条件（*9的地方）和被驱动表的条件一起（*10的地方），才让条数变成15的。要是能让驱动的结果集直接就返回很少，那就好了。

表设计改造方案

1. 新增加staff_event_his表一个时间列字段，state_date;
2. 根据event_q_his和staff_event_his的关联，更新state_date字段的值；
3. 将staff_event_his和event_q_his分别改造为以时间列state_date为分区的分区表；
4. 在staff_id和state_date列建联合索引；
5. 将原代码改造为对staff_event_his增加state_date条件的写法，具体如下图：

代码改造方案（就增加一个条件）

原语句

```
--前面略去
        from staff_event_his a,
             event_q_his     b
            --grade           d,
         --    workaccept_cfg  f
       where a.event_id = b.event_id
        -- and b.grade = d.grade
         and b.flag is null
         and a.staff_id =121
         --and b.event_type = f.work_type
        -- and f.queue_type = 'EVENT_Q'
         AND b.state_date > TRUNC(SYSDATE) -30
         ) a
  where 1 = 1
  ) t  ;
```

性能产生天堑般的差别，就是因为增加了这个条件！
当然要增加这个条件并不容易，要涉及到表设计的改造，模型的变动。

改造后的语句

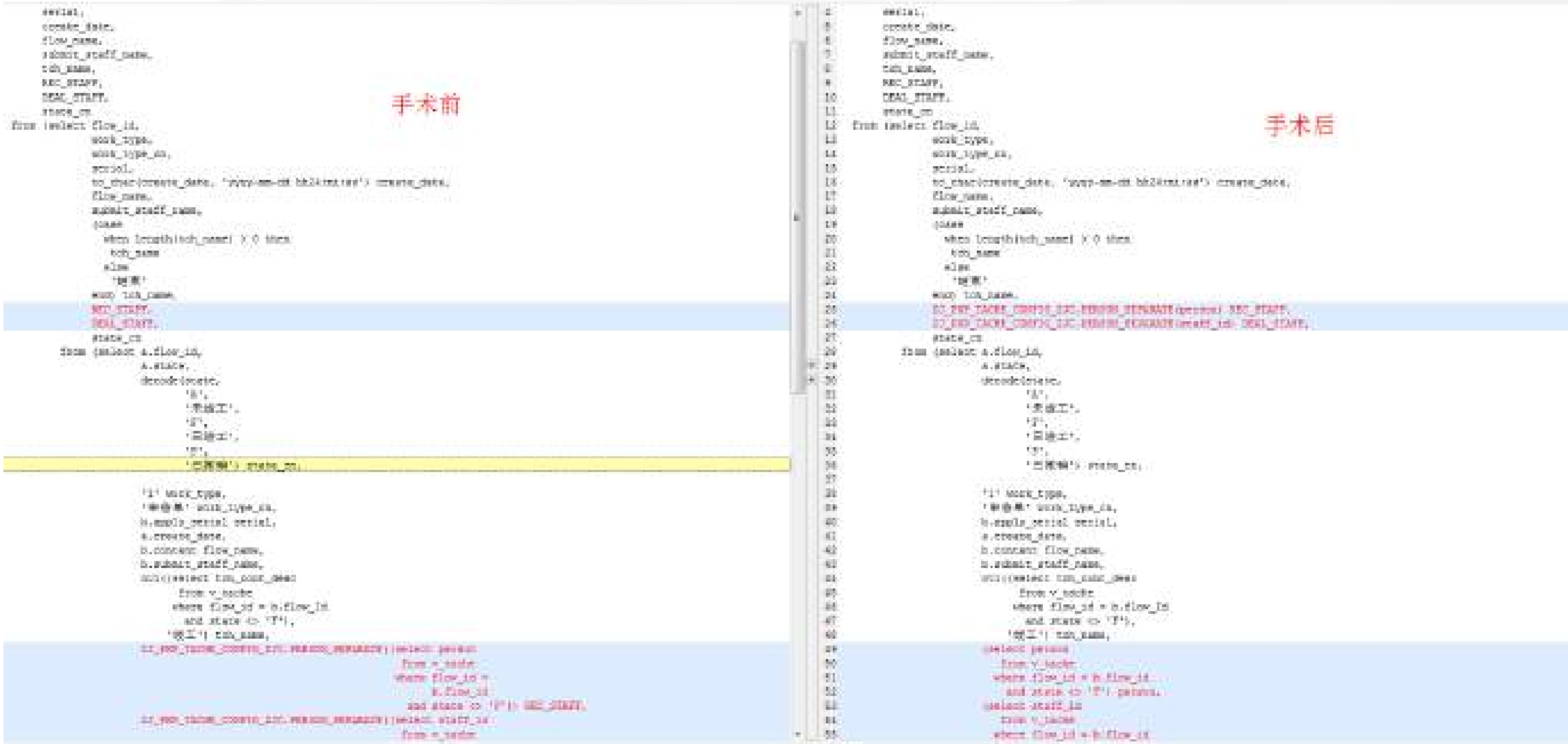
```
--前面略去
        from staff_event_his a,
             event_q_his     b
           -- grade           d,
         --    workaccept_cfg  f
       where a.event_id = b.event_id
        -- and b.grade = d.grade
         and b.flag is null
         and a.staff_id =121
         --and b.event_type = f.work_type
        -- and f.queue_type = 'EVENT_Q'
         AND b.state_date > TRUNC(SYSDATE) -30
         and a.state_date > TRUNC(SYSDATE) -30) a
  where 1 = 1
  ) t  ;
```

其他例子（减少开销就是是硬道理）1：

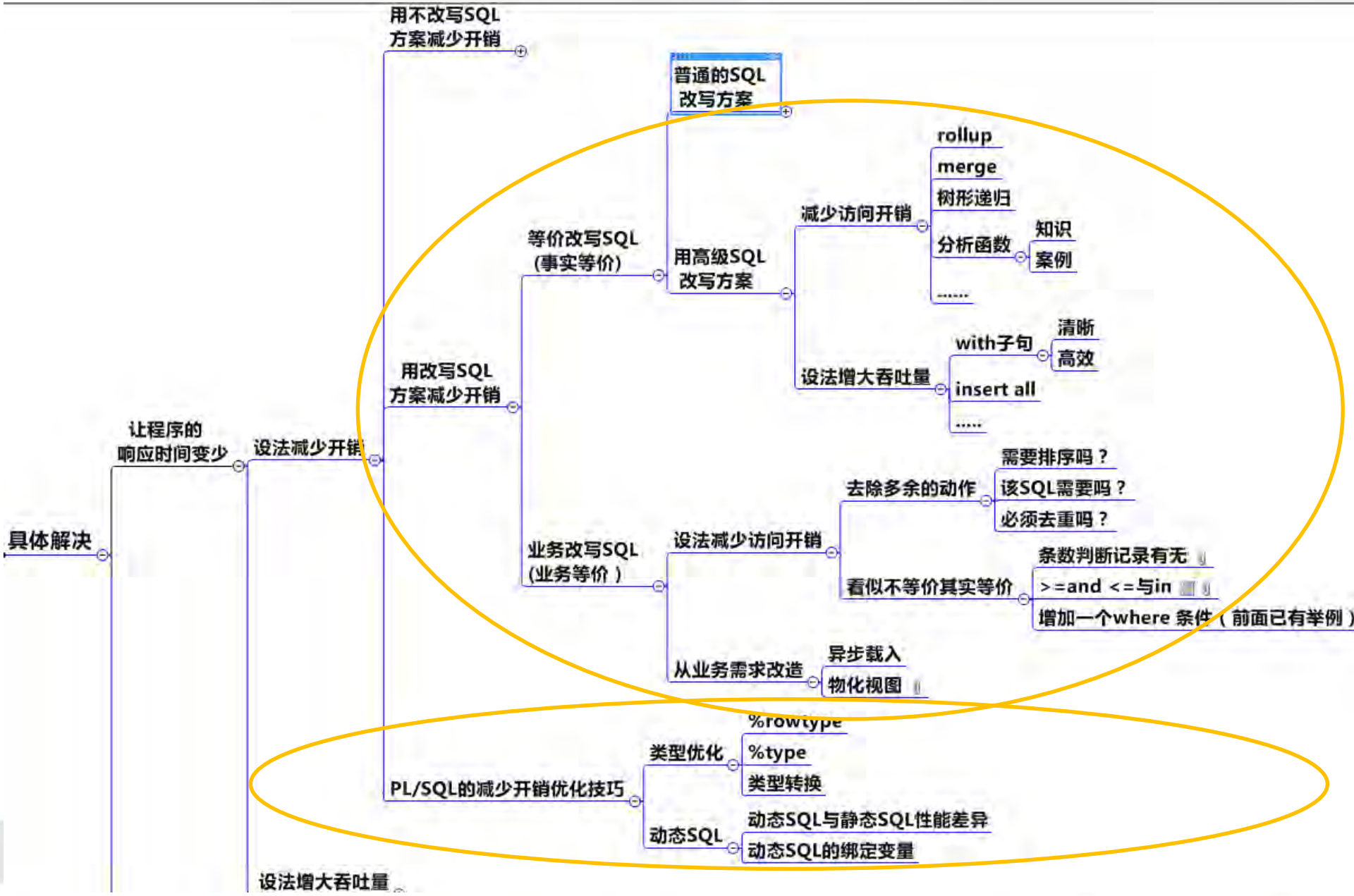
手术前

手术后

其他例子II：



具体解决之减少开销的改写SQL(高级)



随便举个ROLLUP的例子

开始说“高级“SQL了，别激动……

```
select * from (
SELECT  a.dname,b.job,SUM(b.sal) sum_sal
FROM dept a,emp b
WHERE a.deptno = b.deptno
GROUP  BY a.dname,b.job
UNION ALL
--实现了部门的小计
SELECT  a.dname,NULL, SUM(b.sal) sum_sal
FROM dept a,emp b
WHERE a.deptno = b.deptno
GROUP  BY a.dname
UNION ALL
--实现了所有部门总的合计
SELECT  NULL,NULL, SUM(b.sal) sum_sal
FROM dept a,emp b
WHERE a.deptno = b.deptno)
order by dname;
```

DNAME	JOB	SUM_SAL
ACCOUNTING	CLERK	1300
ACCOUNTING	MANAGER	2450
ACCOUNTING	PRESIDENT	5000
ACCOUNTING		8750
RESEARCH	CLERK	1900
RESEARCH	MANAGER	2975
RESEARCH	ANALYST	6000
RESEARCH		10875
SALES	CLERK	950
SALES	MANAGER	2850
SALES	SALESMAN	5600
SALES		9400
		29025

union all 合并笨办法产生的执行计划

Plan hash value: 2979078843

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		29	812	23 (22)	00:00:01
1	SORT ORDER BY		29	812	23 (22)	00:00:01
2	VIEW		29	812	22 (19)	00:00:01
3	UNION-ALL					
4	HASH GROUP BY		14	756	8 (25)	00:00:01
* 5	HASH JOIN		14	756	7 (15)	00:00:01
6	TABLE ACCESS FULL	DEPT	4	88	3 (0)	00:00:01
7	TABLE ACCESS FULL	EMP	14	448	3 (0)	00:00:01
8	HASH GROUP BY		14	672	8 (25)	00:00:01
* 9	HASH JOIN		14	672	7 (15)	00:00:01
10	TABLE ACCESS FULL	DEPT	4	88	3 (0)	00:00:01
11	TABLE ACCESS FULL	EMP	14	364	3 (0)	00:00:01
12	SORT AGGREGATE		1	39		
* 13	HASH JOIN		14	546	7 (15)	00:00:01
14	TABLE ACCESS FULL	DEPT	4	52	3 (0)	00:00:01
15	TABLE ACCESS FULL	EMP	14	364	3 (0)	00:00:01

Predicate Information (identified by operation id):

```
   5 - access("A"."DEPTNO"="B"."DEPTNO")
   9 - access("A"."DEPTNO"="B"."DEPTNO")
  13 - access("A"."DEPTNO"="B"."DEPTNO")
```

统计信息

```
          0  recursive calls
          0  db block gets
         18  consistent gets
          0  physical reads
```

```
SELECT  a.dname,b.job, SUM(b.sal) sum_sal
FROM dept a,emp b
WHERE a.deptno = b.deptno
GROUP  BY ROLLUP(a.dname,b.job);
```

DNAME	JOB	SUM_SAL
SALES	CLERK	950
SALES	MANAGER	2850
SALES	SALESMAN	5600
SALES		9400
RESEARCH	CLERK	1900
RESEARCH	ANALYST	6000
RESEARCH	MANAGER	2975
RESEARCH		10875
ACCOUNTING	CLERK	1300
ACCOUNTING	MANAGER	2450
ACCOUNTING	PRESIDENT	5000
ACCOUNTING		8750
		29025

rollup写法产生的执行计划

Plan hash value: 1037965942

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	756	8 (25)	00:00:01
1	SORT GROUP BY ROLLUP		14	756	8 (25)	00:00:01
* 2	HASH JOIN		14	756	7 (15)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	88	3 (0)	00:00:01
4	TABLE ACCESS FULL	EMP	14	448	3 (0)	00:00:01

Predicate Information (identified by operation id):

```
   2 - access("A"."DEPTNO"="B"."DEPTNO")
```

统计信息

```
          0  recursive calls
          0  db block gets
          6  consistent gets
          0  physical reads
          0  redo size
        778  bytes sent via SQL*Net to client
        416  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
         13  rows processed
```

随便举个WITH的例子

```
select ......
  from (select hopbyhop,
               svcctx_id,
               substr(cause,
                      instr(cause, 'Host = ') + 7,
                      instr(cause, 'Priority = ') - instr(cause, 'Host = ') - 11) peer,
               substr(cause,
                      instr(cause, 'Priority = ') + 11,
                      instr(cause, 'reachable = ') -
                      instr(cause, 'Priority = ') - 13) priority
          from dcc_sys_log
         where cause like '%DCC路由应答%'
           and hopbyhop in (select distinct hopbyhop from dcc_sys_log)) a,
       (select hopbyhop,
               svcctx_id,
               substr(cause,
                      instr(cause, 'Host = ') + 7,
                      instr(cause, 'Priority = ') - instr(cause, 'Host = ') - 11) peer,
               substr(cause,
                      instr(cause, 'Priority = ') + 11,
                      instr(cause, 'reachable = ') -
                      instr(cause, 'Priority = ') - 13) priority
          from dcc_sys_log
         where cause like '%DCC路由应答%'
           and hopbyhop in (select distinct hopbyhop from dcc_sys_log)) b,
       (select hopbyhop,
               svcctx_id,
               substr(cause,
                      instr(cause, 'Host = ') + 7,
                      instr(cause, 'Priority = ') - instr(cause, 'Host = ') - 11) peer,
               substr(cause,
                      instr(cause, 'Priority = ') + 11,
                      instr(cause, 'reachable = ') -
                      instr(cause, 'Priority = ') - 13) priority
          from dcc_sys_log
         where cause like '%DCC路由应答%'
           and hopbyhop in (select distinct hopbyhop from dcc_sys_log)) c
 where a.某字段 = b.某字段 and a.某字段 = c.某字段 and a.某字段 <> b.某字段 and a.某字段 <> c.某字段 and b.某字段 <> c.某字段 and ......
```

手术前

```
with t as
  (select hopbyhop,
               svcctx_id,
               substr(cause,
                      instr(cause, 'Host = ') + 7,
                      instr(cause, 'Priority = ') - instr(cause, 'Host = ') - 11) peer,
               substr(cause,
                      instr(cause, 'Priority = ') + 11,
                      instr(cause, 'reachable = ') -
                      instr(cause, 'Priority = ') - 13)

selet * from t1,t2, t
where ...t...t....t .
```

手术后

随便举个MERGE的例子

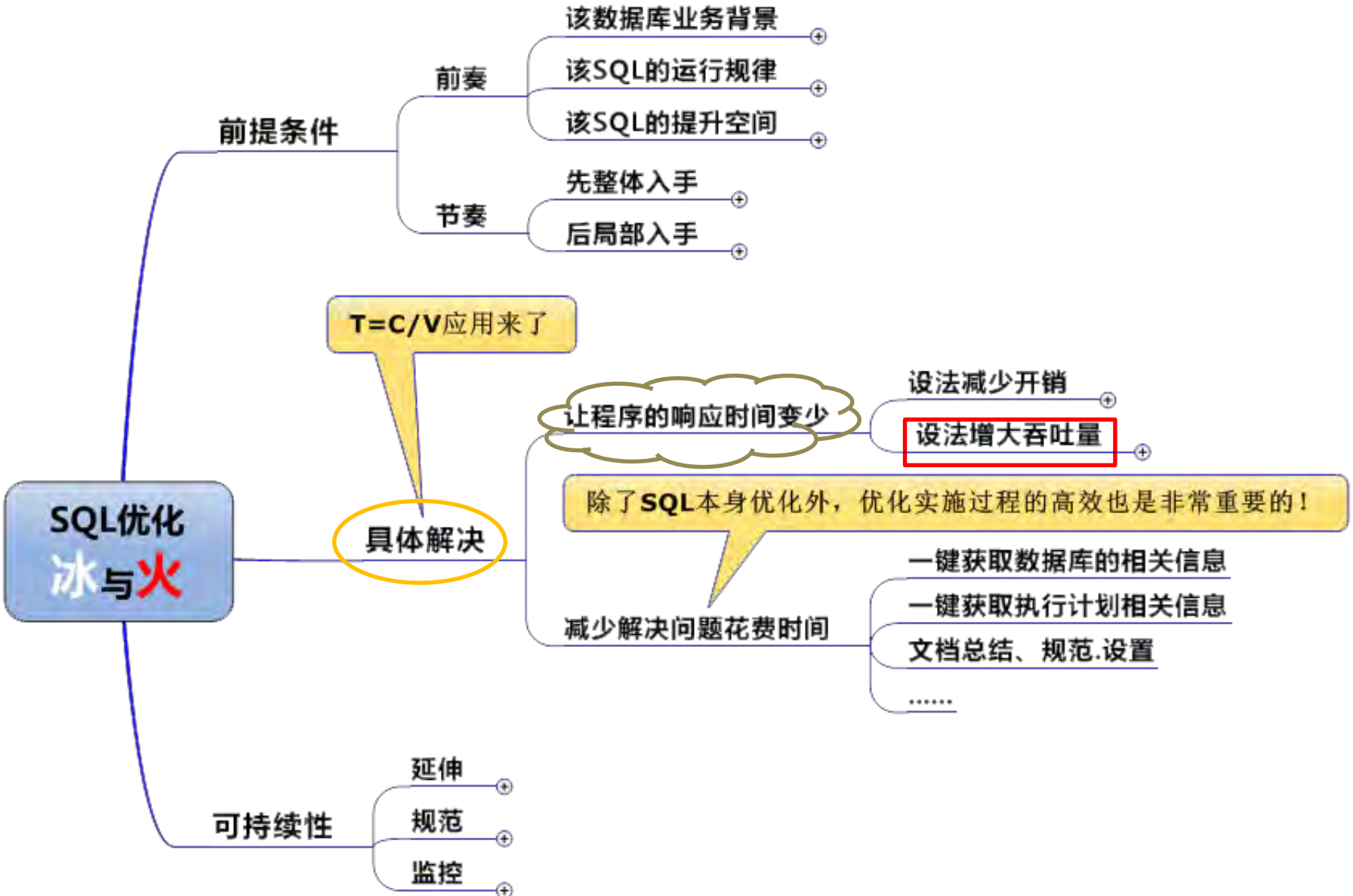
```
/*

需求为：将如下t记录的ID=1的NAME改为ID=2的NAME的值，把ID=2的NAME改为ID=1的NAME的值。

*/
drop table t;
create table t (id number,name varchar2(20));
insert into t values (1,'a');
insert into t values (2,'b');
COMMIT;
SQL> select * from t;
        ID NAME
---------- --------------------
         1 a
         2 b
```

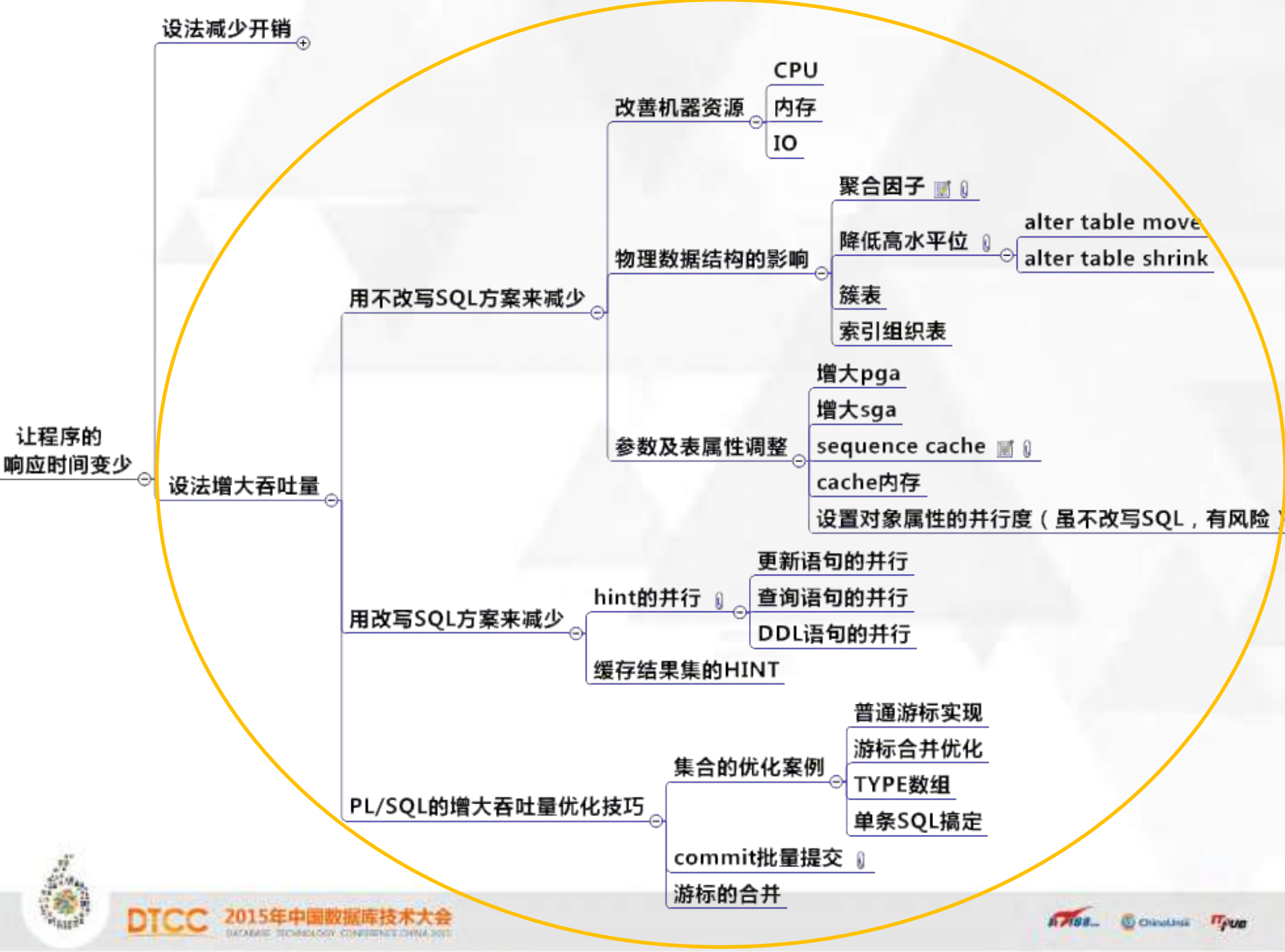
```
--如果执行如下：
UPDATE t SET NAME =(SELECT NAME FROM t WHERE ID=2) WHERE ID=1;这个时候ID=1的NAME值已改变了，就不可能用如下来更新了。
UPDATE t SET NAME =(SELECT NAME FROM t WHERE ID=1) WHERE ID=1;
```

你也可以考虑用过程来实现，不过性能肯定一般般了。

```
merge into  t using
 (
 WITH t_tmp AS
  (SELECT 1 id,(SELECT name FROM t WHERE id=2) name FROM DUAL
        UNION ALL
        SELECT 2,(SELECT name FROM t WHERE id=1) FROM DUAL
        )
  select t.id,t.rowid as rn ,t.name from t ,t_tmp
  where t.id=t_tmp.id)n
  on(t.rowid=n.rn)
   when matched then  update
   set t.name=n.name;

SQL> select * from t;
  ID NAME
----------
   1 a
   2 b
```

SQL优化具体解决之响应时间分析(增大吞吐量)




让程序的响应时间变少
设法减少开销
设法增大吞吐量
用不改写SQL方案来减少
改善机器资源
CPU
内存
IO
物理数据结构的影响
聚合因子
降低高水平位
alter table move
alter table shrink
簇表
索引组织表
参数及表属性调整
增大pga
增大sga
sequence cache
cache内存
设置对象属性的并行度（虽不改写SQL，有风险）
用改写SQL方案来减少
hint的并行
更新语句的并行
查询语句的并行
DDL语句的并行
缓存结果集的HINT
PL/SQL的增大吞吐量优化技巧
集合的优化案例
普通游标实现
游标合并优化
TYPE数组
单条SQL搞定
commit批量提交
游标的合并

太多内容了，加大吞吐量，就说说PLSQL的例子吧

需求描述

1 确定部门里的员工弄的平均工资
2. 如果员工的工资与平均工资差别在20%以上，在EMP_SAL_LOG表里新增一行，并且含偏差值
3. 如果工资在部门里最低，则在EMP_SAL_LOG表里标志出来。

写法1（最差）

游标里每选一个员工就分别
做一次平均值和最小值的判断，性能低下。

```
create or replace
 procedure report_sal_adjustment1 is
 v_avg_dept_sal emp.sal%type;
 v_min_dept_sal emp.sal%type;
 v_dname dept.dname%type;
 cursor c_emp_list is
 select empno, ename, deptno, sal, hiredate
 from emp;
begin
 for each_emp in c_emp_list loop
 select avg(sal)
 into v_avg_dept_sal
 from emp
 where deptno = each_emp.deptno;
 if abs(each_emp.sal - v_avg_dept_sal ) / v_avg_dept_sal > 0.20 then
 select dept.dname, min(emp.sal)
 into v_dname, v_min_dept_sal
 from dept, emp
 where dept.deptno = each_emp.deptno
 and emp.deptno = dept.deptno
 group by dname;
 if v_min_dept_sal = each_emp.sal then
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dname, 'Y');
 else
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dname, 'N');
 end if;
 end if;
end loop;
end report_sal_adjustment1;
/
```

写法2（略好一点）

此次是将平均与最小合并在
一起取了，虽然性能还是低下，但略有提升。

```
create or replace
 procedure report_sal_adjustment2 is
 v_avg_dept_sal emp.sal%type;
 v_min_dept_sal emp.sal%type;
 v_dname dept.dname%type;
 cursor c_emp_list is
 select empno, ename, deptno, sal, hiredate
 from emp;
begin
 for each_emp in c_emp_list loop
 select  avg(emp.sal), min(emp.sal), dept.dname
 into v_avg_dept_sal, v_min_dept_sal,v_dname
 from dept, emp
 where dept.deptno = each_emp.deptno
 and emp.deptno = dept.deptno
 group by dname;
 if abs(each_emp.sal - v_avg_dept_sal ) / v_avg_dept_sal > 0.20 then
 if v_min_dept_sal = each_emp.sal then
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dname, 'Y');
 else
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dname, 'Y');
 end if;
 end if;
 end loop;
end report_sal_adjustment2;
/
```

写法3（好多了）

把AVG和MIN值放到数组中避免反复运算

```
create or replace procedure report_sal_adjustment3 is

 type dept_sal_details is
 record (avg_dept_sal emp.sal%type,
         min_dept_sal emp.sal%type,
         dname dept.dname%type );
 type dept_sals is table of dept_sal_details
 index by binary_integer;
 v_dept_sal dept_sals;
 /*
 cursor c_emp_list is
 select empno, ename, deptno, sal, hiredate
 from emp;
 */
 cursor c_dept_salaries is
 select avg(sal) asal, min(sal) msal, dname, dept.deptno
 from dept, emp
 where emp.deptno = dept.deptno
 group by dname, dept.deptno;
begin
 for i in c_dept_salaries loop
 v_dept_sal(i.deptno).avg_dept_sal := i.asal;
 v_dept_sal(i.deptno).min_dept_sal := i.msal;
 v_dept_sal(i.deptno).dname := i.dname;
 end loop;
-- for each_emp in c_emp_list loop
 for  each_emp in (select empno, ename, deptno, sal, hiredate from emp) loop
 if abs(each_emp.sal - v_dept_sal(each_emp.deptno).avg_dept_sal ) /
 v_dept_sal(each_emp.deptno).avg_dept_sal > 0.20 then
 if v_dept_sal(each_emp.deptno).min_dept_sal = each_emp.sal then
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dept_sal(each_emp.deptno).dname, 'Y');
 else
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate, each_emp.sal,
 v_dept_sal(each_emp.deptno).dname, 'Y');
 end if;
 end if;
end loop;
end report_sal_adjustment3;
/
```

写法4（最棒）

用分析函数实现单条SQL完成

```
create or replace
 procedure report_sal_adjustment4 is
begin
 insert into emp_sal_log
 select e.empno, e.hiredate, e.sal, dept.dname,
 case when sal > avg_sal then 'Y'
 else 'N'
 end case
 from (
 select empno, hiredate, sal, deptno,
 avg(sal) over ( partition by deptno ) as avg_sal,
 min(sal) over ( partition by deptno ) as min_sal
 from emp ) e, dept
 where e.deptno = dept.deptno
 and abs(e.sal - e.avg_sal)/e.avg_sal > 0.20;
 end report_sal_adjustment4;
/
```

写法3（好多了）

把AVG和MIN值放到数组中避免反复运算

```
create or replace procedure report_sal_adjustment3 is

 type dept_sal_details is
 record (avg_dept_sal emp.sal%type,
         min_dept_sal emp.sal%type,
         dname dept.dname%type );
 type dept_sals is table of dept_sal_details
 index by binary_integer;
 v_dept_sal dept_sals;
 /*
 cursor c_emp_list is
 select empno, ename, deptno, sal, hiredate
 from emp;
 */
 cursor c_dept_salaries is
 select avg(sal) asal, min(sal) msal, dname, dept.deptno
 from dept, emp
 where emp.deptno = dept.deptno
 group by dname, dept.deptno;
begin
 for i in c_dept_salaries loop
 v_dept_sal(i.deptno).avg_dept_sal := i.asal;
 v_dept_sal(i.deptno).min_dept_sal := i.msal;
 v_dept_sal(i.deptno).dname := i.dname;
 end loop;
-- for each_emp in c_emp_list loop
 for  each_emp in (select empno, ename, deptno, sal, hiredate from emp) loop
 if abs(each_emp.sal - v_dept_sal(each_emp.deptno).avg_dept_sal ) /
 v_dept_sal(each_emp.deptno).avg_dept_sal > 0.20 then
 if v_dept_sal(each_emp.deptno).min_dept_sal = each_emp.sal then
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate,
 each_emp.sal, v_dept_sal(each_emp.deptno).dname, 'Y');
 else
 insert into emp_sal_log
 values ( each_emp.ename, each_emp.hiredate, each_emp.sal,
 v_dept_sal(each_emp.deptno).dname, 'Y');
 end if;
 end if;
end loop;
end report_sal_adjustment3;
/
```

写法4（最棒）

用分析函数实现单条SQL完成

```
create or replace
 procedure report_sal_adjustment4 is
begin
 insert into emp_sal_log
 select e.empno, e.hiredate, e.sal, dept.dname,
 case when sal > avg_sal then 'Y'
 else 'N'
 end case
 from (
 select empno, hiredate, sal, deptno,
 avg(sal) over ( partition by deptno ) as avg_sal,
 min(sal) over ( partition by deptno ) as min_sal
 from emp ) e, dept
 where e.deptno = dept.deptno
 and abs(e.sal - e.avg_sal)/e.avg_sal > 0.20;
 end report_sal_adjustment4;
/
```

未游标合并的写法

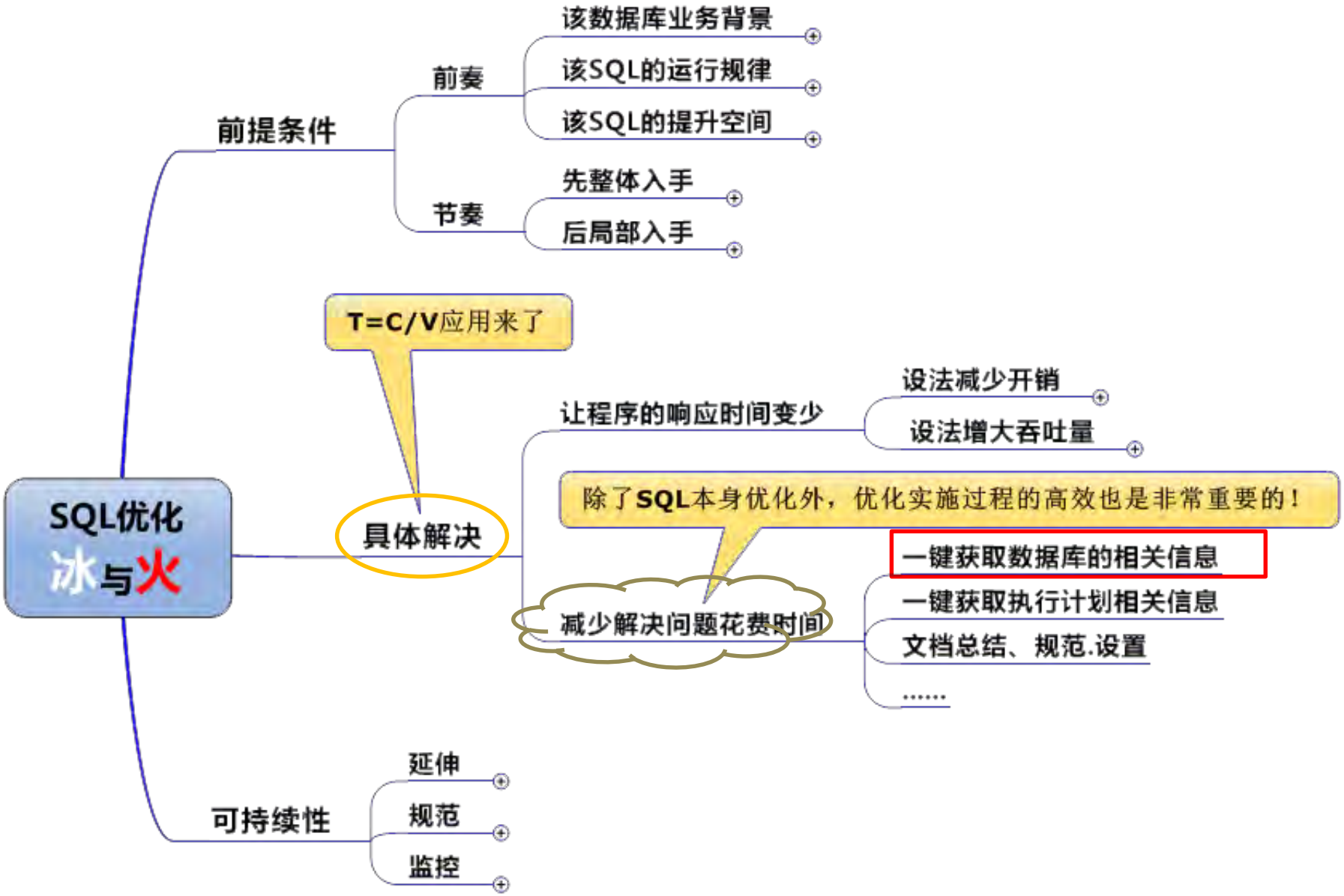
```
begin
    for i in 1 .. 100000
    loop
        for a in ( select t1.a, t1.y
                     from t1 where t1.a = i )
        loop
            for b in ( select t2.b, t2.a, t2.y
                         from t2 where t2.a = a.a )
            loop
                for c in ( select t3.c, t3.b, t3.y
                             from t3 where t3.b = b.b )
                loop
                    null;
                end loop;
            end loop;
        end loop;
    end loop;
end;
/
```

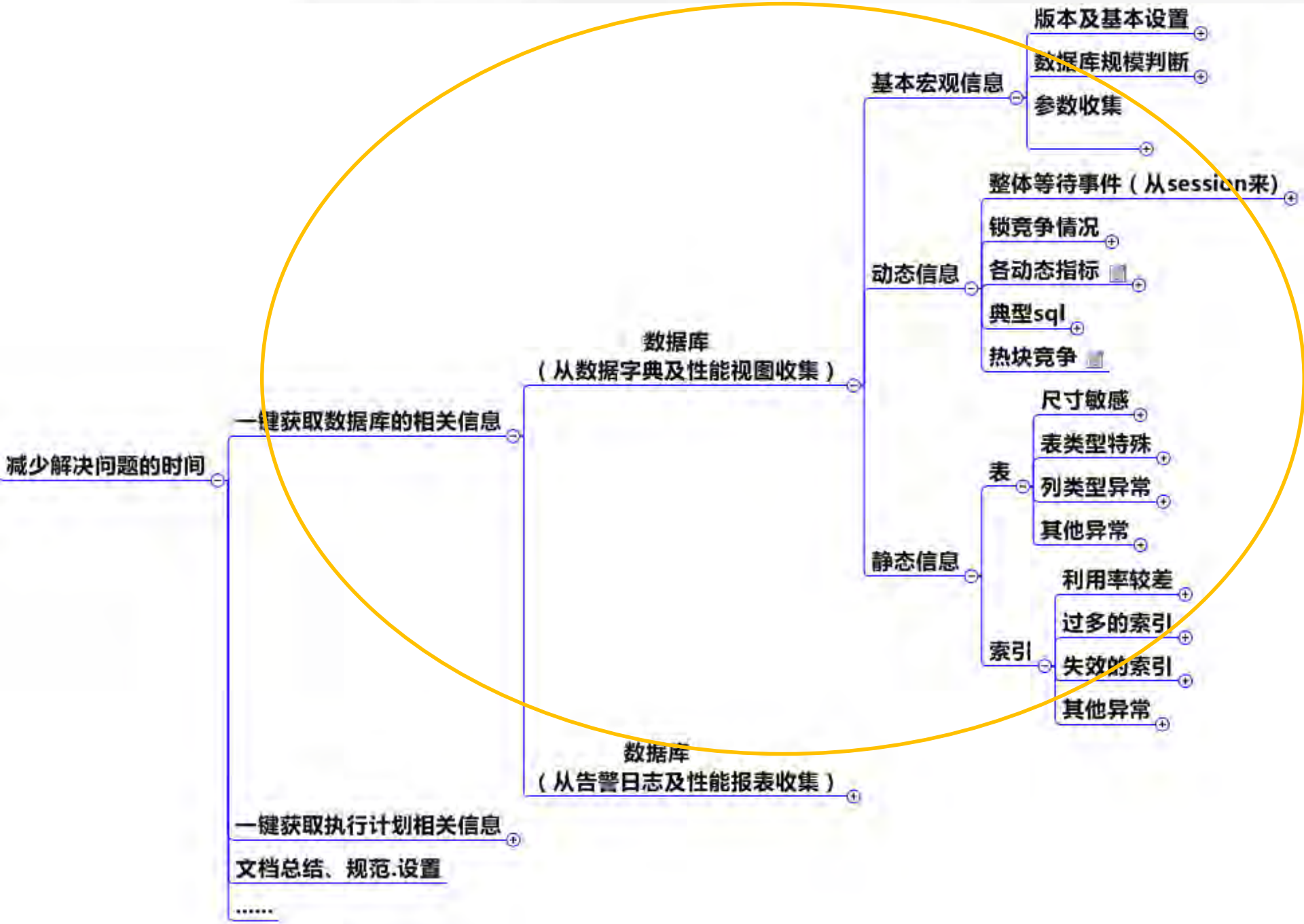
游标合并后的写法

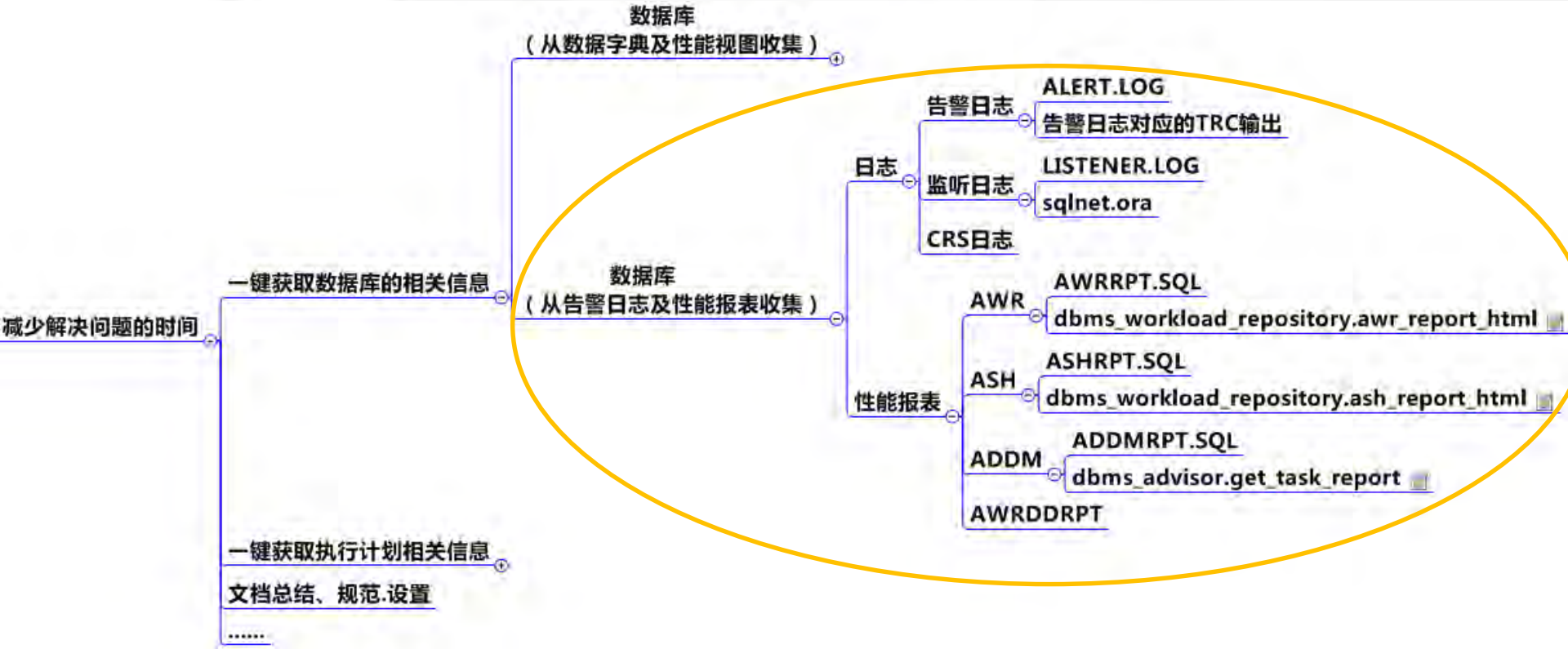
```
begin
    for i in 1 .. 100000
    loop
        for x in ( select t1.a t1a, t1.y t1y,
                          t2.b t2b, t2.a t2a, t2.y t2y,
                          t3.c t3c, t3.b t3b, t3.y t3y
                     from t1, t2, t3
                    where t1.a = i
                      and t2.a (+) = t1.a
                      and t3.b (+) = t2.b )
        loop
            null;
        end loop;
    end loop;
end;
/
```

```
Run1 ran in 141hsec
Run2 ran in 338hsec
run 1 ran in 41.72% of the time

Name                                  Run1      Run2      Diff
STAT...no work - consistent re     188,659   187,658    -1,001
STAT...consistent gets from ca     179,569   178,550    -1,019
STAT...sorts (rows)                  8,418         0    -8,418
LATCH.shared pool simulator        100,088   110,095    10,007
STAT...consistent gets from ca     400,585   448,559    47,974
STAT...consistent gets             400,585   448,559    47,974
STAT...session logical reads       400,612   448,588    47,976
LATCH.cache buffers chains         510,770   558,899    48,129
STAT...consistent gets - exami     111,016   160,009    48,993
STAT...recursive calls             100,068   160,035    59,967
STAT...calls to get snapshot s     100,050   160,026    59,976
STAT...session cursor cache hi     100,002   159,999    59,997
STAT...execute count               100,010   160,009    59,999
STAT...opened cursors cumulati     100,012   160,015    60,003
LATCH.shared pool                  100,302   160,380    60,078
STAT...buffer is not pinned co     252,230   349,227    96,997
STAT...buffer is pinned count      164,109    67,109   -97,000
STAT...session pga memory          393,216   131,072  -262,144

Run1 latches total versus runs ----difference and pct
Run1      Run2      Diff     Pct
737,174   855,465   118,291  86.17%

PL/SQL 过程已成功完成。
```

SQL优化具体解决之减少解决问题时间(数据库信息)




减少解决问题的时间
一键获取数据库的相关信息
数据库
（从数据字典及性能视图收集）
基本宏观信息
版本及基本设置
数据库规模判断
参数收集
动态信息
整体等待事件（从session来）
锁竞争情况
各动态指标
典型sql
热块竞争
静态信息
表
尺寸敏感
表类型特殊
列类型异常
其他异常
索引
利用率较差
过多的索引
失效的索引
其他异常
数据库
（从告警日志及性能报表收集）
一键获取执行计划相关信息
文档总结、规范.设置
.....


减少解决问题的时间
一键获取数据库的相关信息
一键获取执行计划相关信息
文档总结、规范.设置
.....
数据库
（从数据字典及性能视图收集）
数据库
（从告警日志及性能报表收集）
日志
告警日志
ALERT.LOG
告警日志对应的TRC输出
监听日志
LISTENER.LOG
sqlnet.ora
CRS日志
性能报表
AWR
AWRRPT.SQL
dbms_workload_repository.awr_report_html
ASH
ASHRPT.SQL
dbms_workload_repository.ash_report_html
ADDM
ADDMRPT.SQL
dbms_advisor.get_task_report
AWRDDRPT

版本

BANNER
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
PL/SQL Release 11.1.0.6.0 - Production
CORE 11.1.0.6.0 Production
TNS for 32-bit Windows: Version 11.1.0.6.0 - Production
NLSRTL Version 11.1.0.6.0 - Production

最近一次启动时间，版本，以及是否RAC

DB_NAME	INSTANCE_NAME	ARCHIVE	AWR_INTERVAL	AWR_RETENTION	FLASHBACK_ON	RAC	STARTUP_TIME	IS_ASM	RMAN_LASTCOMPL
TEST11G	testllg	STOPPED	+00000 01:00:00.0	+00008 00:00:00.0	NO	NO	17-4月 -15	NOT ASM	

30分钟内CPU或等待最长的

USERNAME	EVENT	SECONDS	SQL_ID	SID	SERIAL#	MACHINE	PROGRAM	OSUSER
SYS	db file scattered read	23	6k4r5jsnj4nqk					
SYS	Cpu + Wait For Cpu	18	6k4r5jsnj4nqk					
SYS	Cpu + Wait For Cpu	16	1ggn/zcf1dxdcln					
SYS	db file parallel read	15	5udw3J5a75qak					
SYS	db file scattered read	10	1ggn/zcf1dxdcln					

近期负载情况(根据AWR快照)

SNAP_DAT	TIME	elapse(min)	DB time(min)	REDO	redo/s	LOGICAL	logicals/s	PHYSICAL	phy/s	EXECS	execs/s	PARSE	parse/s	HARDPARSE	hardparse/s	TRANS	trans/s
15/04/17	08:08	87.15	.26	7842104	1499.94	252756	48.34	1763	.34	59817	11.44	20263	3.88	175	.03	1399	.27
15/04/17	09:08	60.13	.19	5040395	1395.84	152908	42.37	438	.12	37812	10.47	12498	3.46	85	.02	1006	.28
15/04/17	10:08	60.22	.16	4675664	1256.2	140153	38.34	183	.05	37111	10.27	11949	3.31	4	0	988	.27
15/04/17	11:03	62.63	.3	4011120	1067.37	129859	34.48	175	.05	33379	8.88	11039	2.94	24	.01	870	.23
15/04/17	12:07	63.7	.13	3619848	947	117897	30.88	193	.05	28481	7.45	9684	2.53	22	.01	917	.24
15/04/17	13:08	53.85	.29	4934412	1527.21	138034	42	88	.03	38597	3.44	10869	3.30	6	0	1041	.32
15/04/17	14:08	59.23	.19	4300724	1210.39	145899	40.83	49	.01	37230	10.47	11919	3.35	4	0	1186	.33
15/04/17	15:08	60.22	.2	5146496	1424.49	155834	42.91	51	.01	38049	10.53	12405	3.44	4	0	1256	.35
15/04/17	15:48	47.73	2.8	7188840	2516.1	1520834	533.87	776840	271.35	35367	12.31	13143	4.59	270	.09	975	.34
15/04/17	15:49	1.23	.57	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12.87	[illegible]	6.57	31	.51	[illegible]	[illegible]

OWNER	OBJECT_NAME	TABLESPACE_NAME	[illegible]
LJB	C_JNT_BAK		NO
LJB	L_XCTOC	USERS	NO
OLAPSYS	[illegible]	SYSAUX	NO
OLAPSYS	[illegible]	SYSAUX	NO
SYSMAN	[illegible]	SYSMAN	NO

表属性含COMPRESSION的

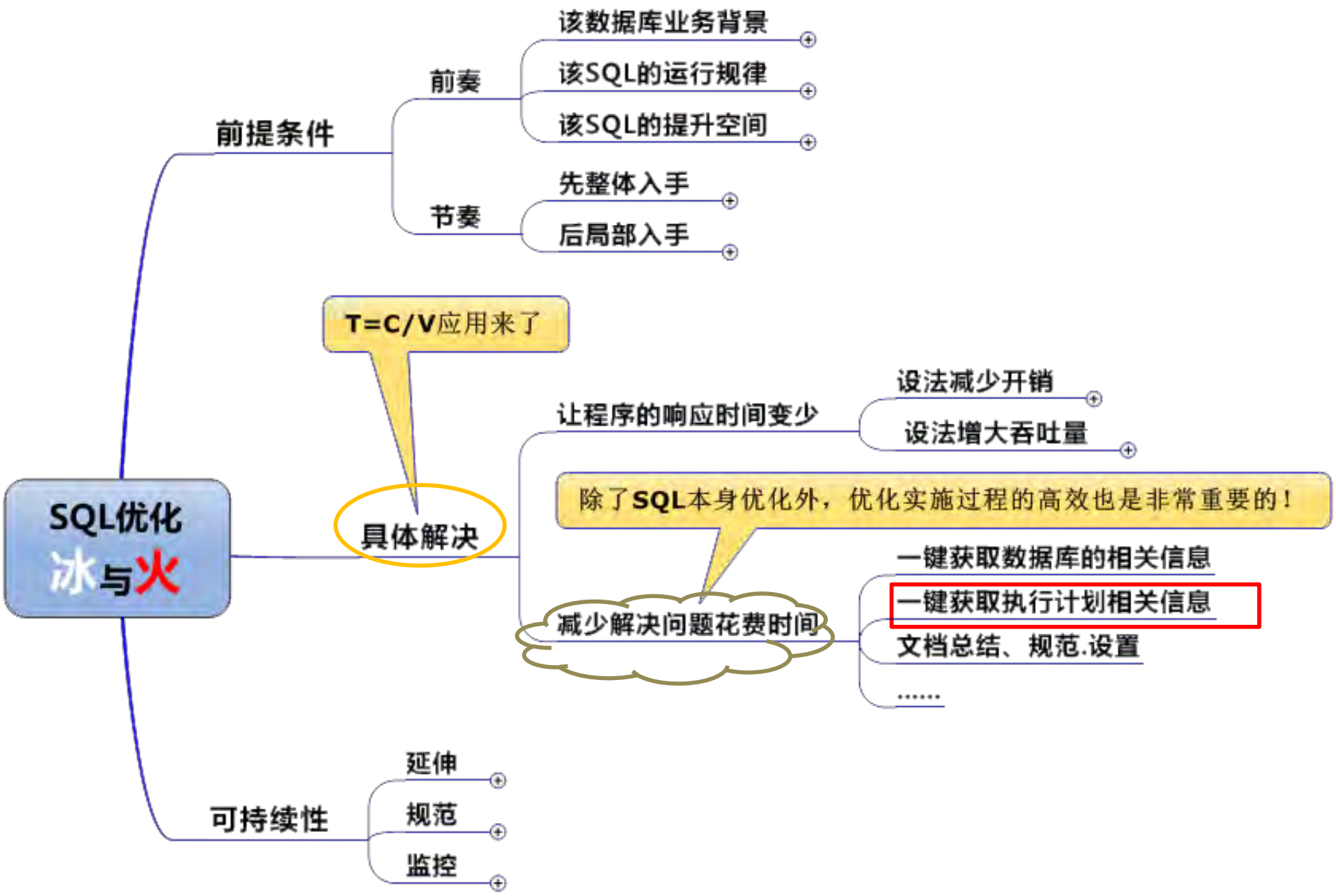
OWNER	TABLE_NAME	TABLESPACE_NAME	COMPRESS
LJB	T_COMPRESS	USERS	ENABLED

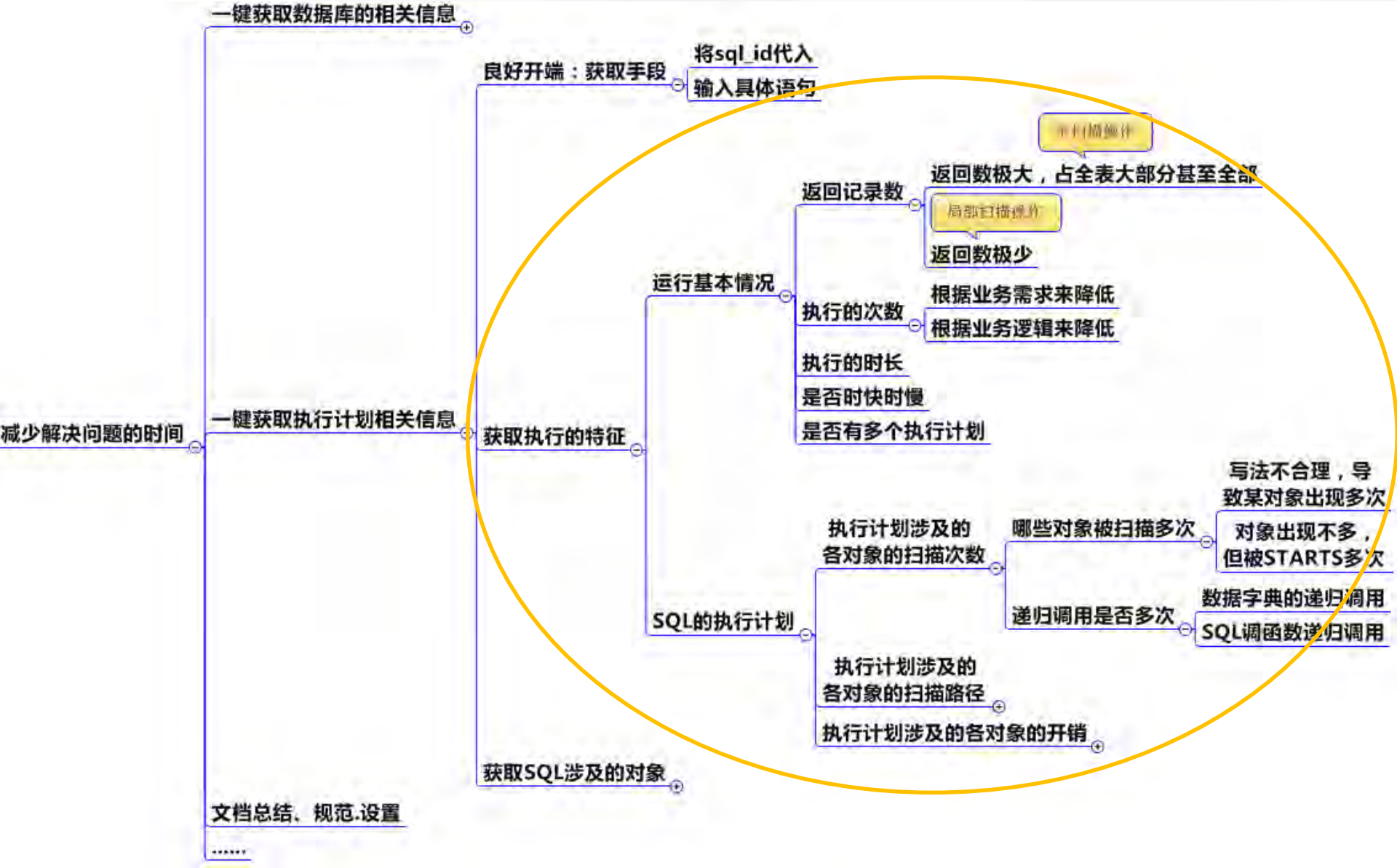
索引属性含COMPRESSION的

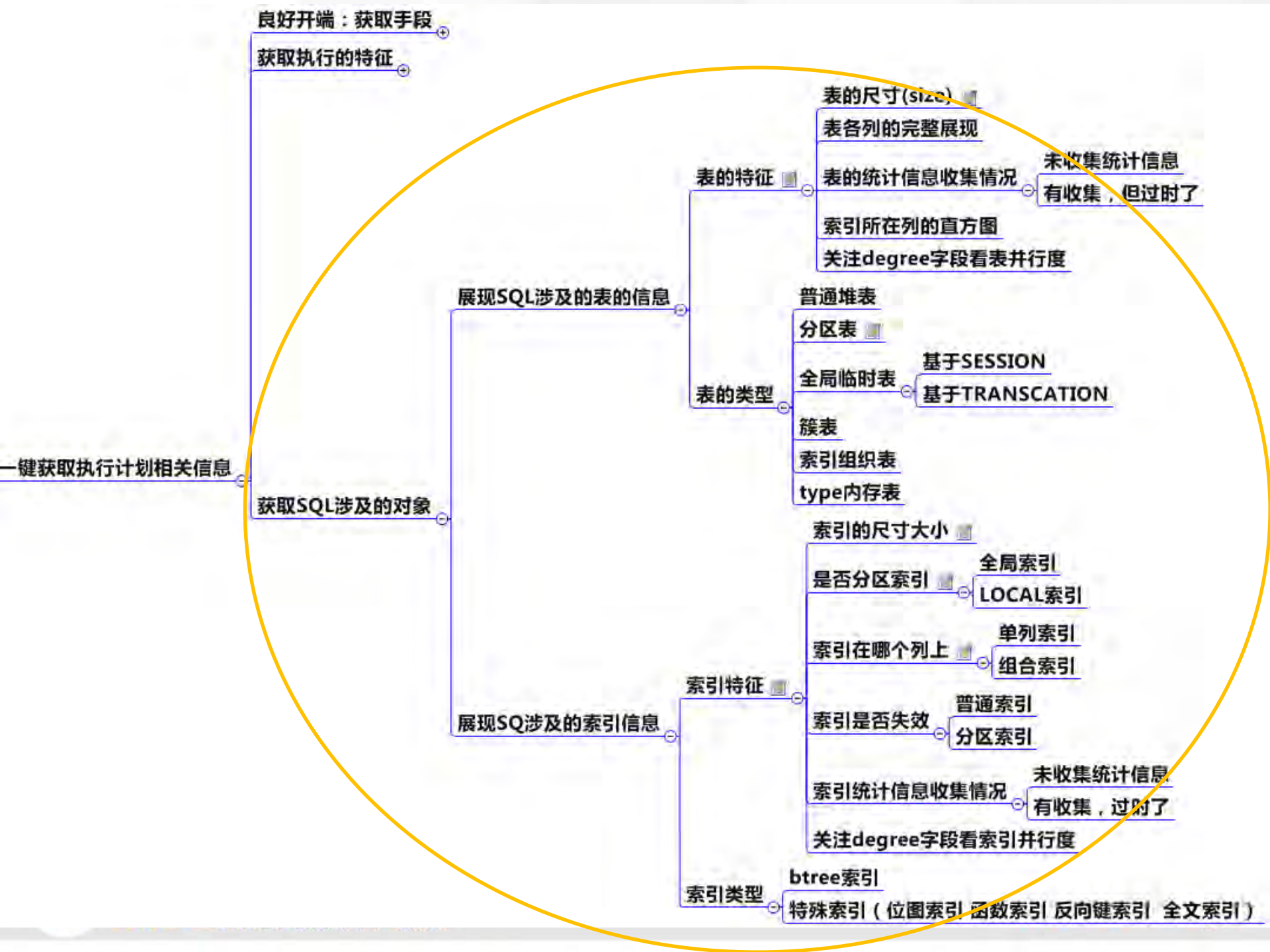
OWNER	INDEX_NAME	TABLE_NAME	COMPRESS
LJB	IDX_ID_COMPRESS	T_IDX_COMPRESS	ENABLED
SH	DRSUP_TEXT_IDXX	DRSUP_TEXT_IDXI	ENABLED

略去。。。。。。

SQL优化具体解决之减少解决问题时间(执行计划相关)

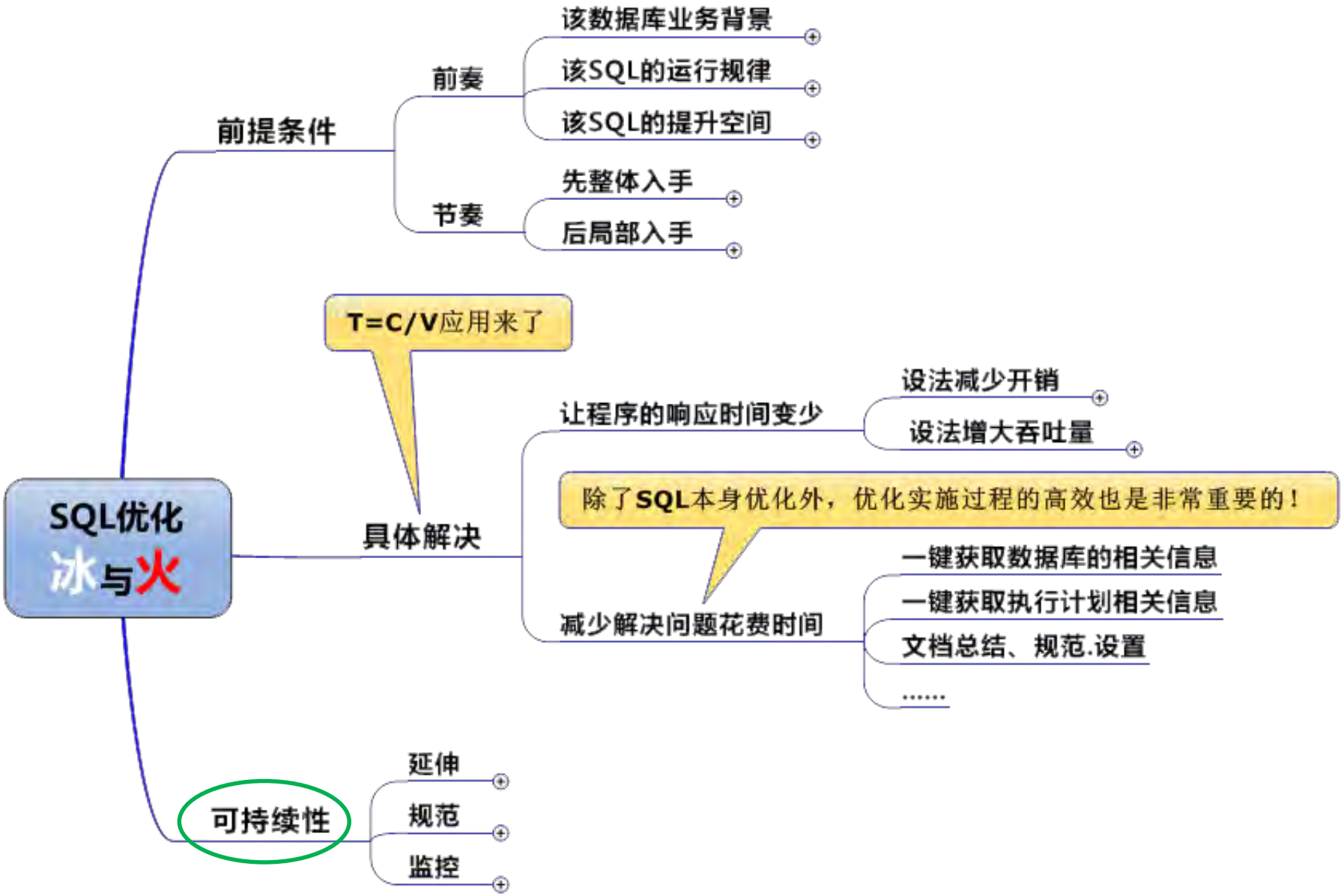


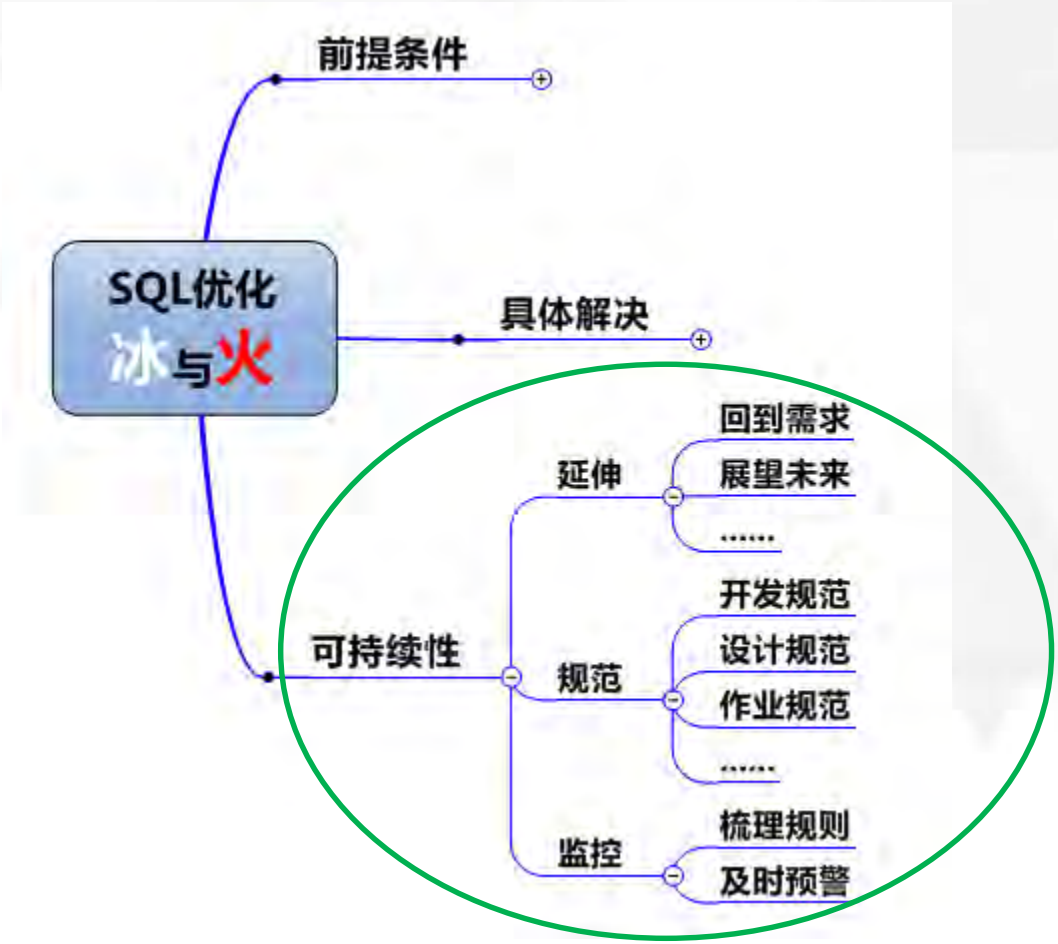

减少解决问题的时间
一键获取数据库的相关信息
一键获取执行计划相关信息
文档总结、规范.设置
.....
良好开端：获取手段
将sql_id代入
输入具体语句
获取执行的特征
运行基本情况
返回记录数
返回数极大，占全表大部分甚至全部
局部扫描很少
返回数极少
执行的次数
根据业务需求来降低
根据业务逻辑来降低
执行的时长
是否时快时慢
是否有多个执行计划
SQL的执行计划
执行计划涉及的
各对象的扫描次数
哪些对象被扫描多次
写法不合理，导
致某对象出现多次
对象出现不多，
但被STARTS多次
递归调用是否多次
数据字典的递归调用
SQL调函数递归调用
执行计划涉及的
各对象的扫描路径
执行计划涉及的各对象的开销
获取SQL涉及的对象


一键获取执行计划相关信息
良好开端：获取手段
获取执行的特征
获取SQL涉及的对象
展现SQL涉及的表的信息
表的特征
表的尺寸(size)
表各列的完整展现
表的统计信息收集情况
未收集统计信息
有收集，但过时了
索引所在列的直方图
关注degree字段看表并行度
表的类型
普通堆表
分区表
全局临时表
基于SESSION
基于TRANSCATION
簇表
索引组织表
type内存表
展现SQ涉及的索引信息
索引特征
索引的尺寸大小
是否分区索引
全局索引
LOCAL索引
索引在哪个列上
单列索引
组合索引
索引是否失效
普通索引
分区索引
索引统计信息收集情况
未收集统计信息
有收集，过时了
关注degree字段看索引并行度
索引类型
btree索引
特殊索引（位图索引 函数索引 反向键索引 全文索引）

快速获取SQL相关信息

SQL优化具体解决之可持续性




前提条件
SQL优化
冰与火
具体解决
可持续性
延伸
回到需求
展望未来
.....
规范
开发规范
设计规范
作业规范
.....
监控
梳理规则
及时预警


ChinaUnix
THANKS