百倍提升 – 真实世界的SQL优化

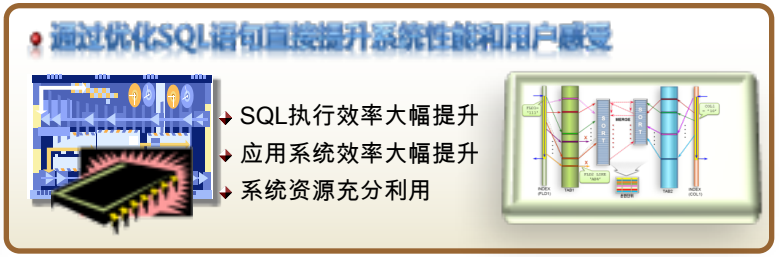
云和恩墨 侯圣文

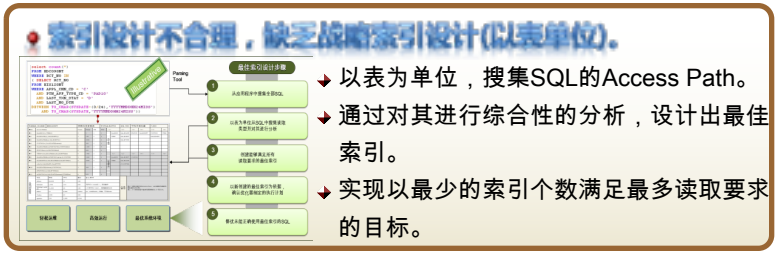
个人介绍

- 姓名：侯圣文
- 网络ID：Secooler
- 北京大学理学硕士
- Oracle ACE Director
- OCM联盟(www.ocmu.org)创始人
- BDA 大数据联盟创始人
- 恩墨学院（www.enmoedu.com）创始人
- ACOUG 核心成员
- ITPUB 论坛资深版主
- DataGuru 专家团成员
- 个人技术Blog：http://www.secooler.me
- 微博：http://weibo.com/secooler
- 电话：13910123683

SQL审核：落实规范解决SQL质量问题

- 基于前端开发的SQL审核服务 – 实现优化前置
  - SQL是一项专业的技能 – Oracle SQL Language Reference ~ 2000 页
  - 由于开发人员的技能差异、变化频繁，很多SQL隐患在开发环节被埋入系统；
  - SQL审核通过专业的工具和SQL专家服务，帮助用户守住上线关卡，实现规范落地；





疏忽 – SQL疏忽引发的性能故障

- 持续的性能优化是系统高效的保障
  - 通过趋势分析发现系统的改变；
  - 通过AWR、SQL报告定位问题；

	Snap Id	Snap Time	Sessions	Curs/Sess	Comment
Begin Snap:	18	04-12月-14 00:00:03	56	5.2	
End Snap:	26	04-12月-14 08:00:06	69	9.1	
Elapsed:		480.05 (mins)			

Load Profile

	Per Second	Per Transaction
Redo size:	33,077.28	157,241.29
Logical reads:	46,175.56	219,507.30
Block changes:	12,514.78	59,492.17
Physical reads:	1,582.20	7,521.40
Physical writes:	79.65	378.62
User calls:	41.86	199.01
Parses:	6.11	29.04
Hard parses:	0.02	0.07

疏忽– SQL疏忽引发的性能故障

- AWR快速分析锁定问题根源
  - 从逻辑读出发进行SQL筛查

```
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------
   802,758,097       15,772       50,897.7   60.4 ######## #########  827999392
UPDATE "BMS_SA_SETTLE_DTL" "A1" SET "ZXCOLUMN9" = :B1 WHERE "A1"
."SASETTLEDTLID"=:B2 AND "A1"."ZXCOLUMN9" IS NULL OR "A1"."ZXCOLUMN9"<>:B1


   358,525,496  178,705,869            2.0   27.0  4915.13 ######### 4254743636
SELECT count(*)     from edis_sasettle    where edis_sasettledtl
id = :b1

    83,661,279        6,231       13,426.6    6.3   139.72    140.72 3380638717
SELECT NULL AS table_cat,        c.owner AS table_schem,
c.table_name,        c.column_name,        c.position AS key_seq
,        c.constraint_name AS pk_name FROM all_cons_columns c, a
ll_constraints k WHERE k.constraint_type = 'P'    AND k.table_nam
e = :1    AND k.owner like :2 escape '/'    AND k.constraint_name
```

疏忽– SQL疏忽引发的性能故障

- 开发疏忽需要通过流程防范和审核
  - 通过执行计划审核可以提前发现问题

```
SQL> explain plan for
  2  UPDATE fsjzmis."BMS_SA_SETTLE_DTL" "A1"
  3  SET "ZXCOLUMN9" = '1'
  4  WHERE "A1" ."SASETTLEDTLID"=96 AND "A1"."ZXCOLUMN9" IS NULL
  5    OR "A1"."ZXCOLUMN9"<>'0';
已解释。
SQL> select * from table(dbms_xplan.display());
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
| Id  | Operation            |  Name               | Rows  | Bytes | Cost  |
--------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT     |                     |       |       |       |
|   1 |  UPDATE              | BMS_SA_SETTLE_DTL   |       |       |       |
|*  2 |   TABLE ACCESS FULL  | BMS_SA_SETTLE_DTL   |       |       |       |
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
   2 - filter("A1"."SASETTLEDTLID"=96 AND "A1"."ZXCOLUMN9" IS NULL OR "A
              1"."ZXCOLUMN9"<>'0')
```

缺少（ ）的Or展开导致了逻辑错误和性能问题

技能：SQL编写不规范引发故障案例

- 在大规模的开发团队，规范的制定、贯彻与落实，都面临着巨大的挑战。

```
select  :
        :
        :
  from uop_act1.tf_b_batch_info a
 where a.trade_time between to_date('20141101123000', 'yyyymmddhh24miss.') and
       to_date('20141101123000', 'yyyymmddhh24miss.')
   and a.trade_staff_id = 'E04Y0175'
   and a.trade_depart_id = '34b0d79'
   and a.trade_depart_id in
       (select a.depart_id
          from uop_act1.td_m_depart a, uop_act1.td_chl_kinddef b
         where a.depart_kind_code = b.chnl_kind_id
           and b.STANDARD_KIND_CODE like '2%');
```

执行计划（请关注下面红色标记部分）

ID	Operation	Name	Rows	Bytes	Cost	IO Cost	Time	访问谓词	过滤谓词
0	SELECT STATEMENT		0	0	9	0	00:00:00		
1	VIEW	VM_NWVW_2	1	547	9	8	00:00:01		
2	HASH UNIQUE		1	150	9	8	00:00:01		
3	FILTER		0	0	0	0	00:00:00		TO_DATE(:VEND_DATE,'yyyymmd...
4	NESTED LOOPS		1	150	8	8	00:00:01		
5	NESTED LOOPS		1	150	8	8	00:00:01		
6	MERGE JOIN CARTESIAN		1	138	5	5	00:00:01		
7	TABLE ACCESS BY INDEX RO...	TF_B_BATCH_INFO	1	126	2	2	00:00:01		(A.TRADE_DEPART_ID='34b0d79' ...
8	INDEX RANGE SCAN	IDX_TF_B_BATCH_ST...	1	0					
9	BUFFER SORT		21	252					
10	TABLE ACCESS FULL	TD_CHL_KINDDEF	21	252					
11	INDEX RANGE SCAN	PK_TD_M_DEPART	1	0					
12	TABLE ACCESS BY GLOBAL INDE...	TD_M_DEPART	1	12					

子查询和父查询使用了相同的表别名，导致查询歧义，结果集错误，导致笛卡尔积；

技能：SQL编写不规范引发故障案例

- 别名不规范导致的笛卡尔积消除

```
Execution Plan
----------------------------------------------------------
Plan hash value: 714801777

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                            |     1 |   143 |     9  (12)| 00:00:01 |
|   1 |  NESTED LOOPS                            |                            |     1 |   143 |     9  (12)| 00:00:01 |
|   2 |   NESTED LOOPS                           |                            |     1 |   143 |     9  (12)| 00:00:01 |
|   3 |    VIEW                                  | VW_NSO_1                   |     1 |     7 |     7   (0)| 00:00:01 |
|   4 |     HASH UNIQUE                          |                            |     1 |    24 |            |          |
|*  5 |      HASH JOIN                           |                            |     1 |    24 |     7   (0)| 00:00:01 |
|   6 |       TABLE ACCESS BY GLOBAL INDEX ROWID | TD_M_DEPART                |     1 |    12 |     4   (0)| 00:00:01 |
|*  7 |        INDEX RANGE SCAN                  | PK_TD_M_DEPART             |     1 |       |     3   (0)| 00:00:01 |
|*  8 |       TABLE ACCESS FULL                  | TD_CHL_KINDDEF             |    31 |   372 |     3   (0)| 00:00:01 |
|*  9 |     INDEX RANGE SCAN                     | IDX_TF_B_BATCH_STAFF_ID    |     1 |       |     0   (0)| 00:00:01 |
|* 10 |    TABLE ACCESS BY INDEX ROWID           | TF_B_BATCH_INFO            |     1 |   136 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

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

   5 - access("C"."DEPART_KIND_CODE"="B"."CHNL_KIND_ID")
   7 - access("C"."DEPART_ID"='34b0d79')
   8 - filter("B"."STANDARD_KIND_CODE" LIKE '2%')
   9 - access("A"."TRADE_STAFF_ID"='E04Y0175')
  10 - filter("A"."TRADE_DEPART_ID"='34b0d79' AND "A"."TRADE_TIME"=TO_DATE(' 2014-11-01 12:30:00', 'syyyy-mm-dd
             hh24:mi:ss') AND "A"."TRADE_DEPART_ID"="DEPART_ID")

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          9  consistent gets
          0  physical reads
          0  redo size
       1811  bytes sent via SQL*Net to client
        509  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

专业 - 从简单改进到专业改写

- 并行更新成为系统瓶颈

2.5小时

Execution Plan

Id	Operation
0	UPDATE STATEMENT
1	UPDATE
2	PX COORDINATOR
3	PX SEND QC (RANDOM)
4	HASH JOIN SEMI BUFFERED
5	PX RECEIVE
6	PX SEND HASH
7	PX BLOCK ITERATOR
8	TABLE ACCESS FULL
9	PX RECEIVE
10	PX SEND HASH
11	PX BLOCK ITERATOR
12	TABLE ACCESS FULL
13	TABLE ACCESS BY INDEX ROWID
14	INDEX RANGE SCAN
15	TABLE ACCESS BY INDEX ROWID
16	INDEX RANGE SCAN

Back to Plan 1(PHV: 845856064)
Back to Top

Full SQL Text

SQL Id	
66qusj70taja7	update /*+parallel(b, 8)*/ stat. serv_id=b.serv_id), plan_id=(s /*+parallel(c, 8)*/ serv_id from

Plan Statistics

- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	8,846,330	8,846,330.10	0.53
CPU Time (ms)	656,363	656,363.34	0.30
Executions	1		
Buffer Gets	52,897,314	52,897,314.00	0.91
Disk Reads	1,293,007	1,293,007.00	0.09
Parse Calls	17	17.00	0.00
Rows	0	0.00	
User I/O Wait Time (ms)	2,349,769		
Cluster Wait Time (ms)	0		
Application Wait Time (ms)	2,528		
Concurrency Wait Time (ms)	127		
Invalidations	0		
Version Count	3		
Sharable Mem(KB)	75		

专业 - 从简单改进到专业改写

- SQL的基本逻辑分析

```
update /*+parallel(b, 8)*/ stat.acc_woff_bill_dtl_4106_0852 b
   set brand    = (
                  select brand
                    from stat.STAT_USER_BAK_201205
                   where serv_id = b.serv_id
       )
     , plan_id = (
                  select plan_id
                    from stat.STAT_USER_BAK_201205
                   where serv_id = b.serv_id
       )
 where b.serv_id in(
                  select /*+parallel(c, 8)*/ serv_id
                    from stat.STAT_USER_BAK_201205 c
       )

```

专业 - 从简单改进到专业改写

```
update /*+parallel(b, 8)*/ acc_bill b
   set brand   = (select brand from USER where serv_id = b.serv_id  )
    , plan_id = (select plan_id from USER where serv_id = b.serv_id )
 where b.serv_id in( select /*+parallel(c, 8)*/ serv_id  from USER c   )
```

2.5分钟

第一次迭代

```
update acc_bill b
   set (brand, plan_id) = (
                          select brand , plan_id
                            from USER where serv_id  = b.serv_id )
 where b.serv_id in (select serv_id  from USER c )
```

1.2分钟

第二次迭代

```
update (      select a.brand brand_a
                   , a.plan_id plan_id_a
                   , b.brand, b.plan_id
                from acc_bill  b , user  a
               where b.serv_id = a.serv_id    )
   set brand     = brand_a , plan_id   = plan_id_a;
```

830 毫秒

深入 – Non Key-Preserved Table

```
SQL> update (
  2                    select a.brand brand_a
  3                         , a.plan_id plan_id_a
  4                         , b.brand
  5                         , b.plan_id
  6                      from bill b
  7                         , users a
  8                     where a.serv_id = b.serv_id
  9          )
 10      set brand      = brand_a
 11        , plan_id    = plan_id_a;
```

```
    set brand      = brand_a
        *
ERROR at line 10:
ORA-01779: cannot modify a column which maps to a non key-preserved table
```

Ora-01779错误解析：造成这个错误的原因是更新的列不是事实表的列，而是维度表的列。Oracle要确保连接后更新的内容可以写到一张表中，而这就要求**连接方式必须是1对N或者1对1的连接**。这样才能确保连接后的结果集数量和事实表一致。从而使得Oracle对连接后子查询的更新可以顺利的更新到事实表中。

```
SQL> alter table users add constraint uq_serv_id unique(serv_id);

Table altered.
```

深入 – Update的权限

- 权限需求差异

```
SQL> create user enmotech identified by enmotech;
SQL> grant create session to enmotech;
SQL> connect enmotech/enmotech
Connected.
SQL> connect eygle/eygle
SQL> create public synonym bill for eygle.bill;
SQL> create public synonym users for eygle.users;
SQL> grant select,update on bill to enmotech;

Grant succeeded.
SQL> grant select on users to enmotech;

Grant succeeded.
```

```
SQL> connect enmotech/enmotech
SQL> update (
  2              select a.brand brand_a , a.plan_id plan_id_a
  3                                     , b.brand  , b.plan_id
  4               from bill b
  5                                     , users a
  6                                     where a.serv_id = b.serv_id
  7                 )
  8              set brand     = brand_a  , plan_id    = plan_id_a;
ERROR at line 5:
ORA-01031: insufficient privileges

SQL> update bill b
  2              set (brand, plan_id) = (
  3  select brand , plan_id
  4    from users where serv_id  = b.serv_id )
  5   where b.serv_id in (select serv_id  from users c )
  6  /
13639 rows updated.
```

深入 – Update的权限

- 我们可以更彻底一点

```
SQL> connect eygle/eygle
Connected.
SQL> revoke update on bill from enmotech;
Revoke succeeded.
```

```
SQL> update (
  2                         select a.brand brand_a
  3                                , a.plan_id plan_id_a
  4                                , b.brand
  5                                , b.plan_id
  6                           from users a,
  7                                bill b
  8                          where a.serv_id = b.serv_id
  9                )
 10        set brand     = brand_a, plan_id    = plan_id_a;
                                    bill b
                                    *
ERROR at line 7:
ORA-01031: insufficient privileges
```

```
SQL> connect enmotech/enmotech
SQL> update
  2  (with tmp as (
  3                         select a.brand brand_a
  4                                , a.plan_id plan_id_a
  5                                , b.brand
  6                                , b.plan_id
  7                           from bill b
  8                                , users a
  9                          where a.serv_id = b.serv_id)
 10                          select * from tmp
 11                )
 12  set brand = brand_a,plan_id = plan_id_a;

13639 rows updated.
```

深入 – Update的权限

- 我们可以看清晰一点

```
SQL> connect enmotech/enmotech
SQL> update
  2  (with tmp as (
  3                          select a.brand brand_a
  4                                  , a.plan_id plan_id_a
  5                                  , b.brand
  6                                  , b.plan_id
  7                           from bill b
  8                                  , users a
  9                           where a.serv_id = b.serv_id)
 10                           select * from tmp
 11                )
 12  set brand = brand_a,plan_id = plan_id_a;

13639 rows updated.
```

小伙伴惊呆了，这不科学！

```
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
PL/SQL Release 11.2.0.3.0 - Production
CORE    11.2.0.3.0      Production
TNS for Linux: Version 11.2.0.3.0 - Production
NLSRTL Version 11.2.0.3.0 - Production
```

深入 – Update的权限

- 新的特性带来新的风险
  - 该漏洞影响范围非常广泛，包括在国内最常见的**11.2.0.3**，**11.2.0.4**，**12.1**等版本；
  - 该漏洞在**2014**年**7**月的**CPU**中被修正，但是如果用户未应用该**CPU**，则漏洞仍然存在；
  - 强烈建议您检查所有**Oracle**数据库，确认是否存在该安全风险；

存储 - 架构优化提升整体性能

- 32x 性能提升

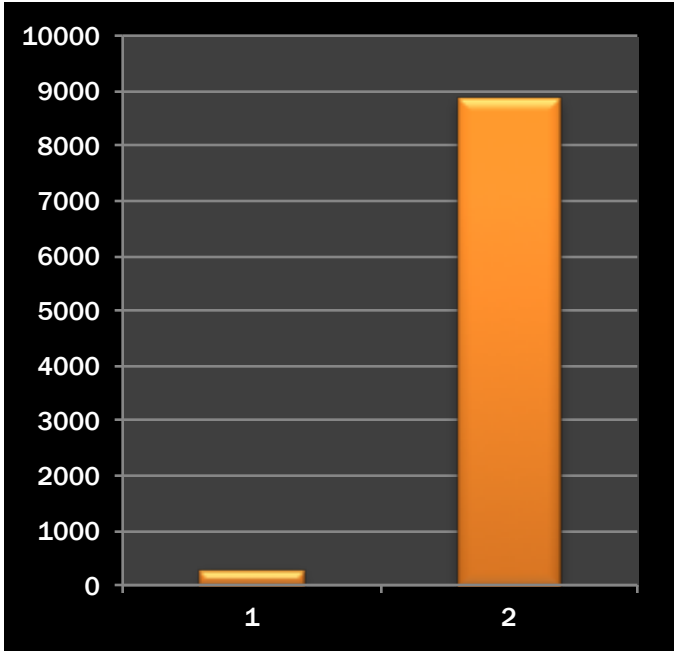
4.5 分钟

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
ZDATA	3013473919	zdata1	1	27-Jun-14 17:06	11.2.0.4.0	YES

Plan Statistics

- % Total DB Time is the Elapsed Time of the SQL statement divided into the

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	270,369	270,368.90	22.03
CPU Time (ms)	105,753	105,752.92	29.01
Executions	1		
Buffer Gets	23,205,585	23,205,585.00	36.28
Disk Reads	428,976	428,976.00	23.91
Parse Calls	9	9.00	0.19
Rows	9,743,271	9,743,271.00	
User I/O Wait Time (ms)	62,574		
Cluster Wait Time (ms)	11,758		
Application Wait Time (ms)	2,102		
Concurrency Wait Time (ms)	21,109		
Invalidations	0		
Version Count	1		
Sharable Mem(KB)	47		



技巧 – 需要熟悉Oracle的SQL特点

- SQL优化从35分钟到16秒 – 150X

```
select /*+rule*/ '没有游离库存' flag
    , e.OPCODE
    , e.GOODSNAME
    , e.GOODSTYPE
    , e.PRODAREA
    , <last_qty>
    , <now_qty>
  from (
              select distinct goodsid
               from BMS_SA_QTYSPLIT_LST
             where pf_splitflag(goodsid) = 1
      ) t     ,
pub_goods e
where t.goodsid  = e.goodsid;
```

技巧 – 需要熟悉Oracle的SQL特点

- SQL优化从35分钟到16秒 – 150X
  - Last_qty 和 Now_qty

```
select /*+rule*/sum(b.goodsqty)
  from gzmpcmis.bms_sa_doc a
     , gzmpcmis.bms_sa_dtl b
     , gzmpcmis.pub_customer c
     , gzmpcmis.zx_no_to_dept d
```

```
a.credate between
to_date('2014/10/01 00:00:00',
 'yyyy/mm/dd hh24:mi:ss')
and
to_date('2014/10/31 23:59:59',
 'yyyy/mm/dd hh24:mi:ss')
```

```
                and d.parentcompanyid              = tt.SALESDEPTID
    )
  and not exists (
              select *
                from gzmpcmis.BMS_SA_QTYSPLIT_LST tt
                   , gzmpcmis.zx_no_to_dept d
               where tt.goodsid                    = b.goodsid
                 and tt.salesdeptid is null
                 and tt.customid                   = c.customid
    )
```

```
select /*+rule*/sum(b.goodsqty)
  from gzmpcmis.bms_sa_doc a
     , gzmpcmis.bms_sa_dtl b
     , gzmpcmis.pub_customer c
```

```
a.credate between
to_date('2014/11/01 00:00:00',
'yyyy/mm/dd hh24:mi:ss')
and
to_date('2014/11/18 23:59:59',
'yyyy/mm/dd hh24:mi:ss')
```

```
                and d.parentcompanyid              = tt.SALESDEPTID
    )
  and not exists (
              select *
                from gzmpcmis.BMS_SA_QTYSPLIT_LST tt
                   , gzmpcmis.zx_no_to_dept d
               where tt.goodsid                    = b.goodsid
                 and tt.salesdeptid is null
                 and tt.customid                   = c.customid
    )
```

技巧 – 需要熟悉Oracle的SQL特点

- SQL优化从35分钟到16秒 – 150X
  - Last_qty 和 Now_qty 数据整合

```
select b.goodsqty
     , t.goodsid
     , a.credate
  from gzmpcmis.bms_sa_doc a
     , gzmpcmis.bms_sa_dtl b
```

```
a.credate between
to_date('2014/10/01 00:00:00',
'yyyy/mm/dd hh24:mi:ss') and
to_date('2014/11/18 23:59:59',
'yyyy/mm/dd hh24:mi:ss')
```

```
              select *
                from gzmpcmis.BMS_SA_QTYSPLIT_LST tt
                   , gzmpcmis.zx_no_to_dept d
               where tt.goodsid                    = b.goodsid
                 and tt.salesdeptid is not null
                 and c.customsaleareano            = d.customsaleareano
                 and d.parentcompanyid             = tt.SALESDEPTID
      )
   and not exists (
              select *
                from gzmpcmis.BMS_SA_QTYSPLIT_LST tt
                   , gzmpcmis.zx_no_to_dept d
               where tt.goodsid                    = b.goodsid
                 and tt.salesdeptid is null
                 and tt.customid                   = c.customid
      )
```

```
select '没有游离库存' flag
    , f.OPCODE
    , f.GOODSNAME
    , f.GOODSTYPE
    , f.PRODAREA
, sum(decode(to_char(e.credate, 'yyyy/mm'),
                '2014/10', e.goodsqty, null)) last_qty
, sum(decode(to_char(e.credate, 'yyyy/mm'),
                '2014/11', e.goodsqty, null)) now_qty
 from gzmpcmis.pub_goods f
   ,< last_qty + now_qty> e
   , (
                select distinct goodsid
                from gzmpcmis.BMS_SA_QTYSPLIT_LST
                where gzmpcmis.pf_splitflag(goodsid) = 1
      ) g
 where f.goodsid                = g.goodsid
   and g.goodsid                = e.goodsid(+)
 group by f.goodsid, f.OPCODE, f.GOODSNAME,f.GOODSTYPE
, f.PRODAREA
order by f.opcode;
```

技巧 – 需要熟悉Oracle的SQL特点

- SQL从35分钟到16秒 – 150X

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.10	0.09	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	2093.18	2044.35	3	160680504	0	152
total	4	2093.28	2044.44	3	160680504	0	152

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	17.20	16.83	0	756606	0	155
total	4	17.20	16.84	0	756606	0	155

架构 - 一条SQL语句引发的血案

- 单条SQL引发雪崩-导致系统不可用

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	37989	02-4□□ -14 09:00:23	836	20.2
End Snap:	37991	02-4□□ -14 09:28:50	1139	20.3
Elapsed:		28.45 (mins)		
DB Time:		3,606.99 (mins)		

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
latch: cache buffers chains	90,884	67,851	747	31.4	Concurrency
CPU time		14,451		6.7	
log file sync	202,004	13,134	65	6.1	Commit
latch: ges resource hash list	26,429	8,347	316	3.9	Other
gc buffer busy	23,770	7,143	300	3.3	Cluster

架构 - 98% LR - 单条SQL消耗

- 通过AWR报告定位Top SQL

SQL ordered by Gets

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by
- Total Buffer Gets: 2,207,946,202
- Captured SQL account for 99.5% of Total

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id
2,138,695,768	73,869	28,952.55	96.86	31576.91	267480.23	88492nrstj3xd
2,134,373,219	73,879	28,890.12	96.67	31202.27	216911.53	6zqqgm5k6nyt6
24,951,837	40,761	612.15	1.13	945.59	178971.75	gup334bprcn0d
11,729,937	24,749	473.96	0.53	353.56	111461.35	ftpaa6vnqj61j

架构 – 窥一斑而知SQL全貌

```
SELECT UNICARD_NO
FROM TF_R_UNICARD
WHERE PRESENT_TAG = '0'
AND LIMIT_DATE + 0 > SYSDATE + 90
AND UNICARD_STATE||NULL = '0'
AND UNICARD_VALCODE||NULL = :B3
AND ROWNUM <= :B2
AND RESERVED1 = :B1
AND (RESERVED2 <> '99' OR RESERVED2 IS NULL) FOR UPDATE
```

FOR UPDATE锁表？
LOCAL索引访问效率低？
ROWNUM绑定变量的值改变？
COUNT STOPKEY没有生效？

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT				476 (100)			
1	FOR UPDATE							
2	COUNT STOPKEY							
3	PARTITION HASH ALL		1	39	476 (1)	00:00:06	1	8
4	TABLE ACCESS BY LOCAL INDEX ROWID	TF_R_UNICARD	1	39	476 (1)	00:00:06	1	8
5	INDEX RANGE SCAN	IDX_TF_R_UNICARD_4	765		283 (1)	00:00:04	1	8

PARTITION HASH ALL

架构 – 数据架构设计形成逻辑炸弹

```
SQL> SELECT UNICARD_NO
  2  FROM UCR_CARD_01.TF_R_UNICARD PARTITION(P5)
  3  WHERE PRESENT_TAG = '0'
  4  AND LIMIT_DATE + 0 > SYSDATE + 90
  5  AND UNICARD_STATE||NULL = '0'
  6  AND UNICARD_VALCODE||NULL = '04'
  7  AND ROWNUM <= 1
  8  AND RESERVED1 = '0422'
  9  AND (RESERVED2 <> '99' OR RESERVED2 IS NULL) ;

UNICARD_NO
------------------------------------
XXXXXXXXXXXXXX
```

架构 - 缩减数据的访问范围

```
SQL> ALTER SESSION FORCE PARALLEL DDL;
Session altered.

SQL> CREATE INDEX IND_UNICARD_RES_VALCODE_DATE ON TF_R_UNICARD
  2  (RESERVED1, UNICARD_VALCODE||NULL, UNICARD_STATE||NULL, LIMIT_DATE + 0) PARALLEL 8 ONLINE;
Index created.

SQL> ALTER INDEX IND_UNICARD_RES_VALCODE_DATE NOPARALLEL;
Index altered.
```

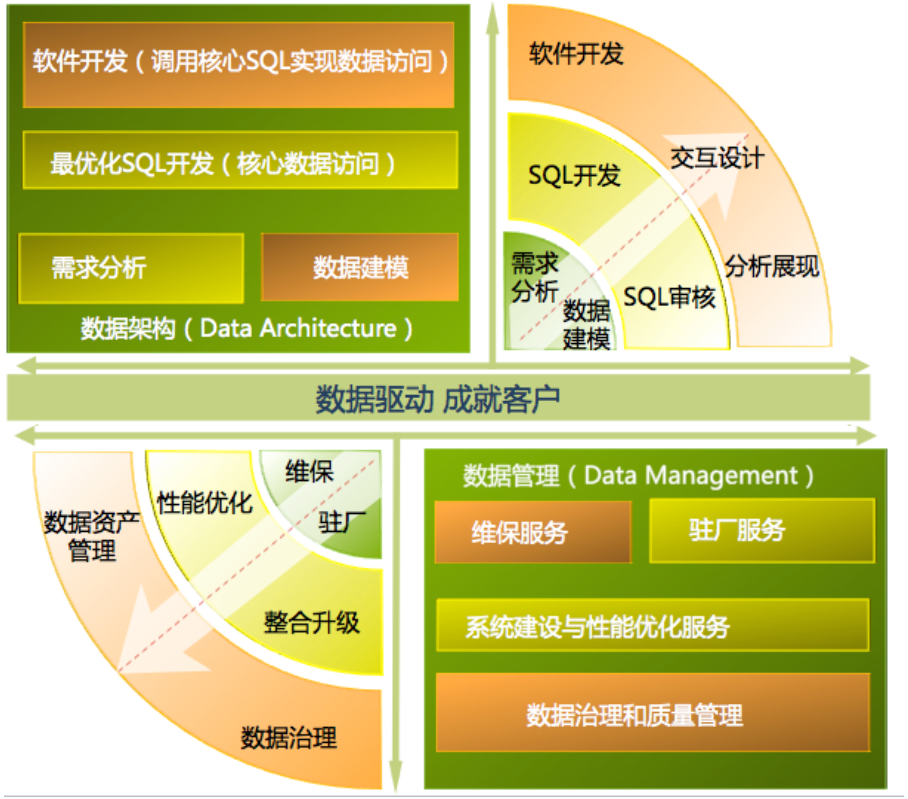
Stat Name	Statement Total	Per Execution
Elapsed Time (ms)	652,229,833	5,305.79
CPU Time (ms)	44,931,910	365.51
Executions	122,928	
Buffer Gets	3,033,188,922	24,674.52
Disk Reads	1,763,706	14.35

Stat Name	Statement Total	Per Execution
Elapsed Time (ms)	1,662,915	245.63
CPU Time (ms)	280,230	41.39
Executions	6,770	
Buffer Gets	20,870,067	3,082.73
Disk Reads	10,246	1.51

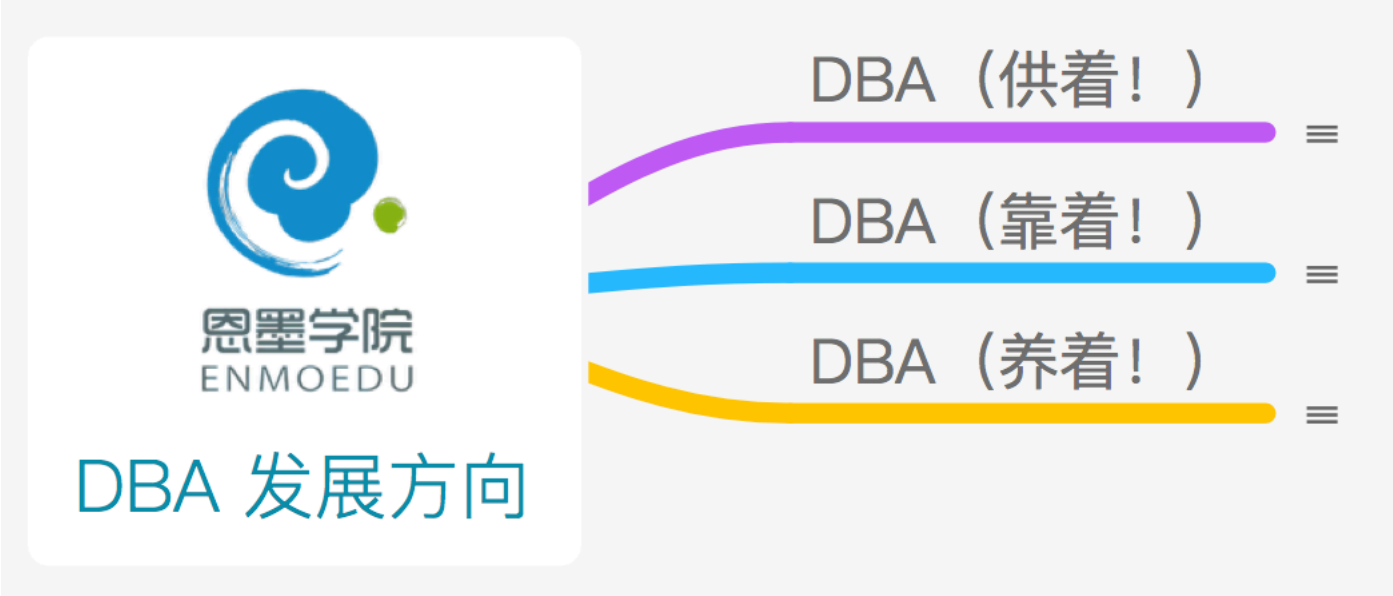
DevOps：自顶至下的融合运维

在企业系统开发建设中，最为重要的环节是业务流程分析、数据模型构建与应用质量控制。企业的数据系统应当始终以数据模型基础、以SQL质量为核心，进行高效的应用程序设计与开发。

在企业系统运维管理中，围绕数据核心，开展维保、优化以及顶层的数据治理、数据资产管理等服务。而系统优化常常可以为系统带来数百、上千倍的性能提升，改善用户体验、节约企业投资。



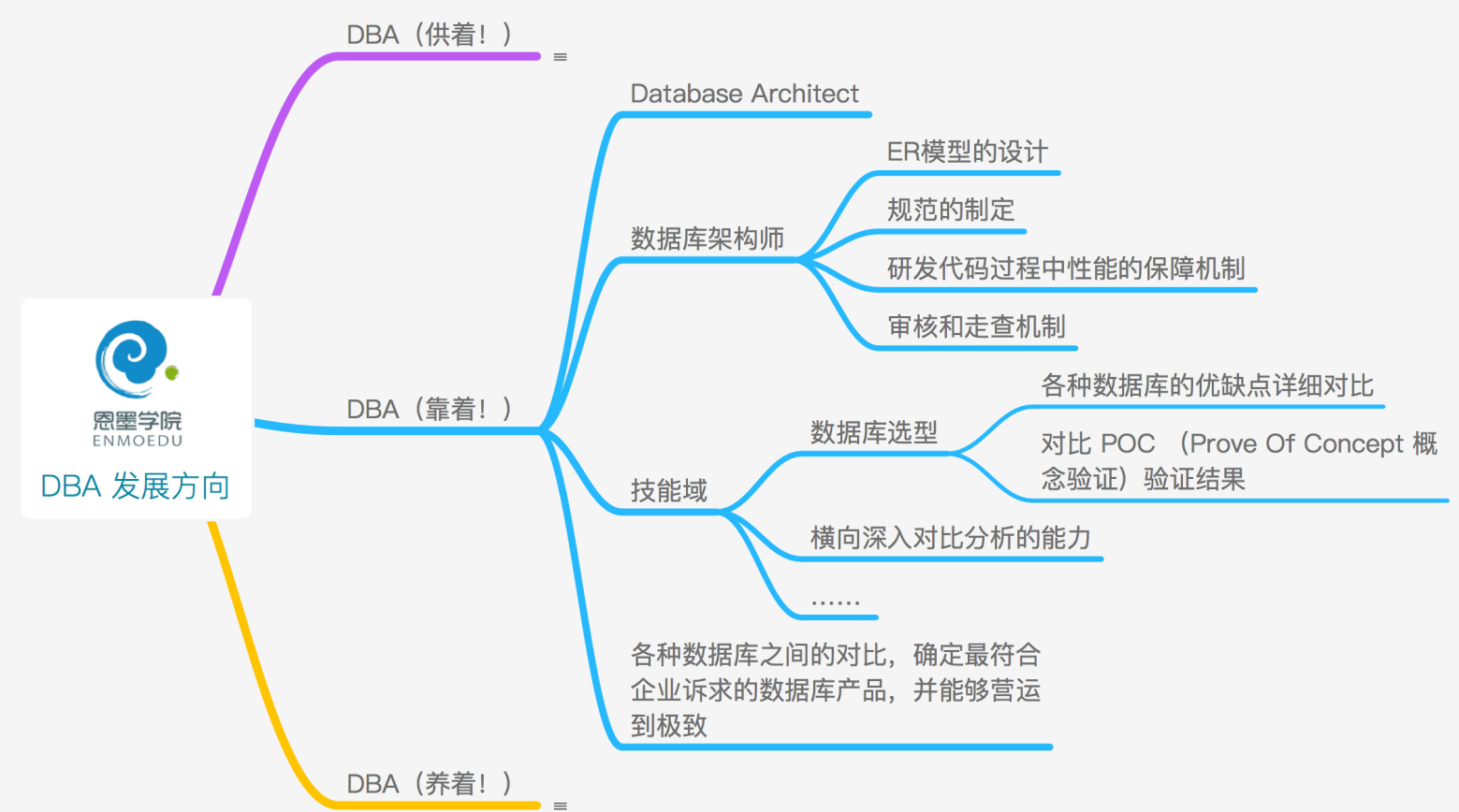
DBA 职业发展路径



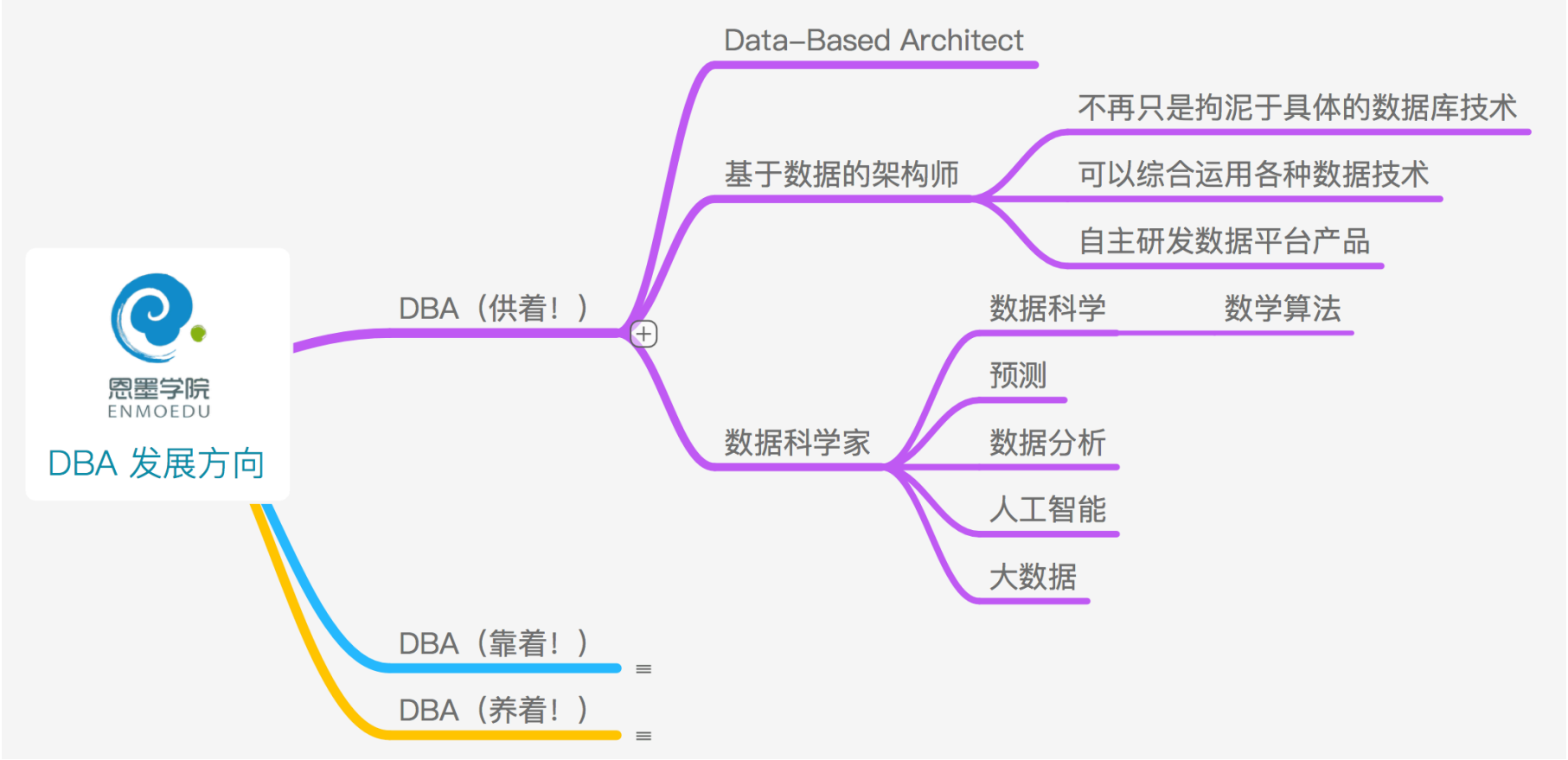
DBA 职业发展路径


恩墨学院
ENMOEDU
DBA 发展方向
DBA（供着！）
≡
DBA（靠着！）
≡
DBA（养着！）
Database Administrator
数据库管理员
类似于消防员的角色
到处救火
或者防患于未然

DBA 职业发展路径


DBA（供着！）
Database Architect
数据库架构师
ER模型的设计
规范的制定
研发代码过程中性能的保障机制
审核和走查机制
恩墨学院
ENMOEDU
DBA 发展方向
DBA（靠着！）
技能域
数据库选型
各种数据库的优缺点详细对比
对比 POC （Prove Of Concept 概念验证）验证结果
横向深入对比分析的能力
……
各种数据库之间的对比，确定最符合企业诉求的数据库产品，并能够营运到极致
DBA（养着！）

DBA 职业发展路径


恩墨学院
ENMOEDU
DBA 发展方向
DBA（供着！）
Data-Based Architect
基于数据的架构师
不再只是拘泥于具体的数据库技术
可以综合运用各种数据技术
自主研发数据平台产品
数据科学家
数据科学
数学算法
预测
数据分析
人工智能
大数据
DBA（靠着！）
DBA（养着！）

DBA 职业发展路径


DBA 发展方向
DBA（供着！）
Data-Based Architect
基于数据的架构师
不再只是拘泥于具体的数据库技术
可以综合运用各种数据技术
自主研发数据平台产品
数据科学家
数据科学
数学算法
预测
数据分析
人工智能
大数据
DBA（靠着！）
Database Architect
数据库架构师
ER模型的设计
规范的制定
研发代码过程中性能的保障机制
审核和走查机制
技能域
数据库选型
各种数据库的优缺点详细对比
对比 POC （Prove Of Concept 概念验证）验证结果
横向深入对比分析的能力
……
各种数据库之间的对比，确定最符合企业诉求的数据库产品，并能够营运到极致
DBA（养着！）
Database Administrator
数据库管理员
类似于消防员的角色
到处救火
或者防患于未然

ACOUG All China Oracle User Group 中国 Oracle 用户组 | 云和恩墨 ENMOTECH | IT大咖说 知识分享平台

坚持迎美好　　自律赢未来

一位每天陪你奔跑 10 公里的伙伴

侯圣文@secooler