DTCC

第十一届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2020

架构革新 高效可控

IT168.com ChinaUnix.net ITPUB

北京国际会议中心 | 2020/12/21-12/23

个人介绍

ORACLE ACE

- 杨廷琨(yangtingkun)
  - Oracle ACE
  - ACOUG副总裁
  - ITPUB数据库管理区版主
  - 参与编写《Oracle数据库性能优化》、《Oracle DBA手记》、《Oracle DBA手记3》和《Oracle性能优化与诊断案例精选》
  - 二十年的DBA经验
  - 个人BLOG中积累了2500篇原创技术文章
  - 云和恩墨CTO

ACOUG
All China Oracle User Group
中国 Oracle 用户组

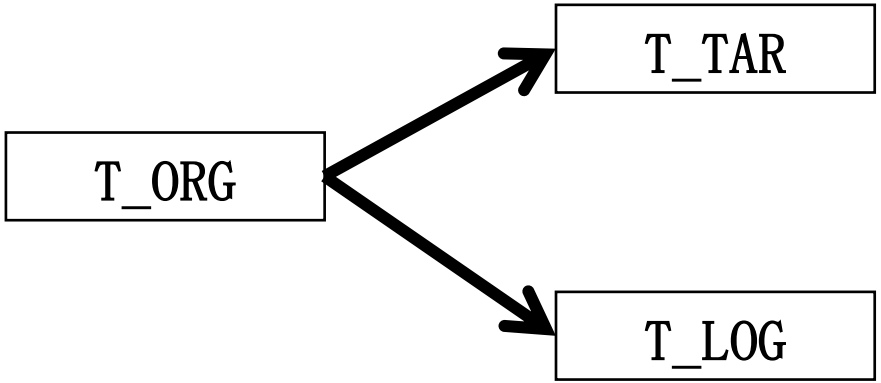
高效SQL的必要性

- SQL是世界上第二大的编程语言，超过50%的开发者使用SQL
- 80%的数据库问题是SQL引起的
- 80%的性能问题来自20%的SQL语句
- 高并发环境中，单条SQL语句可能导致整个数据库出现性能故障

- 合理运用新特性
- 数据集整体处理
- 设计SQL执行计划
- 严格过滤数据

合理运用新特性

目标：从一张表取数据插入到另一张表中，此外需要为插入的目标表做一个应用级的日志表，也就是说在插入目标表的同时，还需要将相同的数据插入到日志表中。



合理运用新特性

方案:

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER T_INSERT_TAR
AFTER INSERT ON T_TAR
FOR EACH ROW
BEGIN
INSERT INTO T_LOG (ID, NAME)
VALUES (:NEW.ID, :NEW.NAME);
END;
/
```

合理运用新特性

方案：

- CREATE TRIGGER
  - 太“重”
  - 实现与需求有差异
  - 增加后续维护成本
  - 触发器效率较低

合理运用新特性

方案:

- CREATE TRIGGER
- DOUBLE INSERT

```
INSERT INTO T_TAR SELECT * FROM T_ORG;
INSERT INTO T_LOG SELECT * FROM T_ORG;
```

合理运用新特性

方案：

- CREATE TRIGGER
- DOUBLE INSERT
  - 一致性
    - 锁
    - 串行事务
    - 临时表
    - AS OF查询
  - 原子性
    - 异常处理

合理运用新特性

方案:

- CREATE TRIGGER
- DOUBLE INSERT
- OPEN CURSOR

```
SQL> BEGIN
  2  FOR I IN (SELECT * FROM T_ORG) LOOP
  3  INSERT INTO T_TAR VALUES (I.ID, I.NAME);
  4  INSERT INTO T_LOG VALUES (I.ID, I.NAME);
  5  END LOOP;
  6  END;
  7  /

PL/SQL 过程已成功完成。
```

合理运用新特性

方案：

- CREATE TRIGGER
- DOUBLE INSERT
- OPEN CURSOR
  - 效率低

合理运用新特性

方案：

- CREATE TRIGGER
- DOUBLE INSERT
- OPEN CURSOR
- BULK INTO VARIABLE

合理运用新特性

```
SQL> DECLARE
  2  TYPE T_ID IS TABLE OF T_ORG.ID%TYPE;
  3  TYPE T_NAME IS TABLE OF T_ORG.NAME%TYPE;
  4  V_ID T_ID;
  5  V_NAME T_NAME;
  6  BEGIN
  7  SELECT ID, NAME BULK COLLECT INTO V_ID, V_NAME
  8  FROM T_ORG;
  9  FORALL I IN 1..V_ID.COUNT
 10  INSERT INTO T_TAR VALUES(V_ID(I), V_NAME(I));
 11  FORALL I IN 1..V_ID.COUNT
 12  INSERT INTO T_LOG VALUES(V_ID(I), V_NAME(I));
 13  END;
 14  /

PL/SQL 过程已成功完成。
```

合理运用新特性

方案：

- CREATE TRIGGER
- DOUBLE INSERT
- OPEN CURSOR
- BULK INTO VARIABLE
  - 复杂度高
  - 效率较低

合理运用新特性

方案:

- CREATE TRIGGER
- DOUBLE INSERT
- OPEN CURSOR
- BULK INTO VARIABLE
- INSERT ALL

```
SQL> INSERT ALL INTO T_TAR (ID, NAME)
  2  INTO T_LOG (ID, NAME)
  3  SELECT ID, NAME FROM T_ORG;
```

合理运用新特性

“新”特性：

- INSERT ALL/ANY　　插入多张表或限定条件插入
- MERGE　　合并UPDATE和INSERT语句
- WITH　　避免重复读取
- ANALYTIC FUNCTIONS　　行级计算避免自关联
- TOP-N　　分页语句
- PIVOT/UNPIVOT　　行列转换
- MODEL　　报表数组处理

- 合理运用新特性
- 数据集整体处理
- 设计SQL执行计划
- 严格过滤数据

数据集整体处理

判断表空间是否自动扩展:

```
SQL> select distinct tablespace_name, autoextensible
  2  from DBA_DATA_FILES
  3  where autoextensible = 'YES'
  4  union
  5  select distinct tablespace_name, autoextensible
  6  from DBA_DATA_FILES
  7  where autoextensible = 'NO'
  8  and tablespace_name not in
  9  (select distinct tablespace_name
 10  from DBA_DATA_FILES
 11  where autoextensible = 'YES');

TABLESPACE_NAME                AUT
------------------------------ ---
SYSAUX                         YES
SYSTEM                         YES
TEST                           NO
UNDOTBS1                       YES
USERS                          YES
```

数据集整体处理

```
SQL> select distinct tablespace_name, autoextensible
  2  from DBA_DATA_FILES
  3  where autoextensible = 'YES'
  4  union
  5  select distinct tablespace_name, autoextensible
  6  from DBA_DATA_FILES
  7  where autoextensible = 'NO'
  8  and tablespace_name not in
  9  (select distinct tablespace_name
 10  from DBA_DATA_FILES
 11  where autoextensible = 'YES');
```

- 优点：思路清晰
- 缺点：效率低效，冗余严重
  - 三次扫描DBA_DATA_FILES视图
  - 包含三次DISTINCT操作
  - 包含UNION数据集操作
  - 包含NOT IN子查询

数据集整体处理

```
SQL> SELECT TABLESPACE_NAME, MAX(AUTOEXTENSIBLE)
  2  FROM DBA_DATA_FILES
  3  GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                MAX
------------------------------ ---
SYSAUX                         YES
UNDOTBS1                       YES
USERS                          YES
TEST                           NO
SYSTEM                         YES
```

数据集整体处理

- Tom说过:
  - 如果可能你应该使用单条SQL语句来实现
  - 如果单条SQL语句无法实现，那么考虑使用PL/SQL
  - 如果PL/SQL无法实现，尝试使用Java存储过程

- 我们建议:
  - 能够用单条SQL处理的，就不要使用多条
  - 能够只扫描一次的，就不要扫描多次
  - 从整体上处理数据，避免单条处理逻辑

- 合理运用新特性
- 数据集整体处理
- 设计SQL执行计划
- 严格过滤数据

设计SQL执行计划

客户某SQL语句执行缓慢，消耗大量逻辑读：

```
SQL> EXPLAIN PLAN FOR
  2  SELECT VCM.POL_ID, TC.MOBILE, TC.FIRST_NAME PH_NAME,
  3     PKG_UTILS.F_GET_DESC('T_TITLE', DECODE(TC.GENDER,'M', '1', 'F', '2'), '211'),
  4     PKG_UTILS.F_GET_DESC('V_PRO_LIFE_3',
  5              (SELECT T.PROD_ID FROM T_CON_PROD T WHERE T.POL_ID = VCM.POL_ID AND T.MASTER_ID IS NULL), '211'),
  6     SUM(TPA.FEE_AMOUNT), VCM.POLICY_CODE, PKG_UTILS.F_GET_DESC('T_LIA_STAT',VCM.LIA_STATE, '211'),
  7     PKG_UTILS.F_GET_DESC('T_SAL_CHAN', VCM.CHANN_TYPE, '211'), VCM.CHANN_TYPE,TC.CUSTOMER_ID, TCO.ORGAN_CODE
  8  FROM V_CON_MAS VCM, T_CON_MAS TCM, T_AGENT TA, T_CUSTOMER TC,
  9     T_POH VPH,T_COM_ORG TCO, T_P_A TPA
 10  WHERE VCM.POL_ID = VPH.POL_ID
 11  AND VCM.SERVICE_AGENT = TA.AGENT_ID
 12  AND VCM.POL_ID = TCM.POL_ID
 13  AND VPH.PARTY_ID = TC.CUSTOMER_ID
 14  AND VCM.ORGAN_ID = TCO.ORGAN_ID
 15  AND VCM.POL_ID = TPA.POL_ID
 16  AND VCM.LIABILITY_STATE = 1
 17  AND TPA.FEE_TYPE IN (41, 569)
 18  AND TPA.FEE_STATUS <> 2
 19  AND (EXISTS (
 20     SELECT 1 FROM T_PO_CH T, T_CH TC WHERE T.MAS_C_ID = TC.CH_ID  AND TC.CH_ID = TPA.CH_ID AND T.SERV_ID = 3 ))
 21  AND TPA.DUE_TIME >= (TRUNC(:B1 ) + 7)
 22  AND TPA.DUE_TIME < (TRUNC(:B1 ) + 8)
 23  AND REGEXP_LIKE(TRIM(TC.MOBILE), '^\d{11}$')
 24  AND NOT EXISTS (
 25     SELECT 1 FROM T_DATA_EXT SDE, T_DATA TSD
 26     WHERE SDE.DATA_ID = TSD.DATA_ID AND SDE.RELAT_VALUE_1 = VCM.POL_ID AND TSD.SMS_ID = 12 AND SDE.RELAT_VALUE_2 = TO_CHAR(:B1 , 'yyyy-MM-dd'))
 27  GROUP BY VCM.POL_ID, TC.MOBILE, TC.FIRST_NAME, TC.GENDER, VCM.POLICY_CODE, VCM.LIABILITY_STATE, VCM.CHANNEL_TYPE, TC.CUSTOMER_ID, TCO.ORGAN_CODE;
```

设计SQL执行计划

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		356	53400	901K (1)
* 1	INDEX RANGE SCAN	IDX_CON_PROD__PRD_ID	1	12	1 (0)
2	HASH GROUP BY		356	53400	901K (1)
* 3	FILTER				
* 4	FILTER				
5	NESTED LOOPS		356	53400	84579 (2)
* 6	HASH JOIN		356	51264	84578 (2)
7	TABLE ACCESS FULL	T_C_O	143	2145	4 (0)
8	NESTED LOOPS SEMI		356	45924	84573 (2)
9	NESTED LOOPS		370	44030	84481 (2)
10	NESTED LOOPS		6067	539K	82962 (2)
11	NESTED LOOPS		6036	465K	81452 (2)
* 12	HASH JOIN		6037	430K	81451 (2)
* 13	TABLE ACCESS BY INDEX ROWID	T_P_A	4812	145K	596 (1)
* 14	INDEX RANGE SCAN	IDX_P_A__DUE	14477		10 (0)
15	VIEW	V_CON_MAS	877K	35M	80844 (2)
16	UNION-ALL				
* 17	TABLE ACCESS FULL	T_CON_MAS_LOG	877K	36M	80844 (2)
* 18	FILTER				
* 19	TABLE ACCESS FULL	T_CON_MAS	155K	6357K	21715 (1)
* 20	INDEX UNIQUE SCAN	PK_T_AGENT	1	6	1 (0)
21	TABLE ACCESS BY INDEX ROWID	T_POH	1	12	1 (0)
* 22	INDEX RANGE SCAN	IDX_POH_PO_ID	1		1 (0)
* 23	TABLE ACCESS BY INDEX ROWID	T_CUSTOMER	1	28	1 (0)
* 24	INDEX UNIQUE SCAN	PK_T_CUSTOMER	1		1 (0)
* 25	TABLE ACCESS BY INDEX ROWID	T_PO_CH	244K	2389K	1 (0)
* 26	INDEX RANGE SCAN	IDX_PO_CH__MAS_CH_ID	1		1 (0)
* 27	INDEX UNIQUE SCAN	PK_T_CON_MAS	1	6	1 (0)
28	NESTED LOOPS				
29	NESTED LOOPS		1	34	4591 (1)
30	TABLE ACCESS BY INDEX ROWID	T_DATA_EXT	1	25	4590 (1)
* 31	INDEX SKIP SCAN	IDX_DATA_EXT__RELAT_	1		4589 (1)
* 32	INDEX UNIQUE SCAN	PK_T_DATA	1		1 (0)
* 33	TABLE ACCESS BY INDEX ROWID	T_DATA	1	9	1 (0)

设计SQL执行计划

```
SQL> create table t_objects as select * from dba_objects;
表已创建。

SQL> create index ind_obj_id on t_objects(object_id);
索引已创建。

SQL> create table t_tab as select * from dba_tables;
表已创建。

SQL> create table t_ind as select * from dba_indexes;
表已创建。

SQL> create index ind_tab_name on t_tab(table_name);
索引已创建。

SQL> create index ind_ind_name on t_ind(index_name);
索引已创建。

SQL> create view v_seg as
  2  select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  3  union all
  4  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A';
视图已创建。
```

设计SQL执行计划

```
SQL> select obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj, v_seg v
  3  where obj.object_id = 12345
  4  and obj.object_name = v.table_name;

未选定行

-------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   194 |    43 |
|*  1 |  HASH JOIN                   |             |     1 |   194 |    43 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS   |     1 |   115 |     2 |
|*  3 |    INDEX RANGE SCAN          | IND_OBJ_ID  |     1 |       |     1 |
|   4 |   VIEW                       | V_SEG       |  3340 |  257K |    40 |
|   5 |    UNION-ALL                 |             |       |       |       |
|*  6 |     TABLE ACCESS FULL        | T_TAB       |  1184 | 95904 |    15 |
|*  7 |     TABLE ACCESS FULL        | T_IND       |  2156 |  178K |    25 |
-------------------------------------------------------------------------------
```

设计SQL执行计划

```
SQL> select /*+ index(t_tab ind_tab_name) */ obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj, v_seg v
  3  where obj.object_id = 12345
  4  and obj.object_name = v.table_name;

未选定行
```

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	43
* 1	HASH JOIN		1	194	43
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW	V_SEG	3340	257K	40
5	UNION-ALL				
* 6	TABLE ACCESS FULL	T_TAB	1184	95904	15
* 7	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select /*+ index(v.t_tab ind_tab_name) */ obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj, v_seg v
  3  where obj.object_id = 12345
  4  and obj.object_name = v.table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	1210
* 1	HASH JOIN		1	194	1210
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW	V_SEG	3340	257K	1207
5	UNION-ALL				
* 6	TABLE ACCESS BY INDEX ROWID	T_TAB	1184	95904	1182
7	INDEX FULL SCAN	IND_TAB_NAME	2367		11
* 8	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj,
  3  (select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A') v
  6  where obj.object_id = 12345
  7  and obj.object_name = v.table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	43
* 1	HASH JOIN		1	194	43
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	40
5	UNION-ALL				
* 6	TABLE ACCESS FULL	T_TAB	1184	95904	15
* 7	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select /*+ index(v.t_tab ind_tab_name) */ obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj,
  3  (select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A') v
  6  where obj.object_id = 12345
  7  and obj.object_name = v.table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	1210
* 1	HASH JOIN		1	194	1210
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	1207
5	UNION-ALL				
* 6	TABLE ACCESS BY INDEX ROWID	T_TAB	1184	95904	1182
7	INDEX FULL SCAN	IND_TAB_NAME	2367		11
* 8	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select /*+ index(t_tab ind_tab_name) */ obj.owner, obj.object_name, created, blocks
  2  from t_objects obj,
  3  (select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A')
  6  where obj.object_id = 12345
  7  and obj.object_name = table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	43
* 1	HASH JOIN		1	194	43
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	40
5	UNION-ALL				
* 6	TABLE ACCESS FULL	T_TAB	1184	95904	15
* 7	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> SELECT ID, OBJECT_ALIAS, DEPTH
  2  FROM V$SQL_PLAN
  3  WHERE SQL_ID IN
  4  (SELECT SQL_ID FROM V$SQL
  5  WHERE SQL_TEXT LIKE 'select /*+ index(t_tab ind_tab_name) */%')
  6  ORDER BY SQL_ID, ID;

        ID OBJECT_ALIAS                          DEPTH
---------- ------------------------------ ----------
         0                                         0
         1                                         1
         2 OBJ@SEL$1                               2
         3 OBJ@SEL$1                               3
         4 from$_subquery$_002@SEL$1               2
         5                                         3
         6 T_TAB@SEL$2                             4
         7 T_IND@SEL$3                             4
```

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   194 |    43 |
|*  1 |  HASH JOIN                   |             |     1 |   194 |    43 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS   |     1 |   115 |     2 |
|*  3 |    INDEX RANGE SCAN          | IND_OBJ_ID  |     1 |       |     1 |
|   4 |   VIEW                       |             |  3340 |   257K|    40 |
|   5 |    UNION-ALL                 |             |       |       |       |
|*  6 |     TABLE ACCESS FULL        | T_TAB       |  1184 | 95904 |    15 |
|*  7 |     TABLE ACCESS FULL        | T_IND       |  2156 |   178K|    25 |
-------------------------------------------------------------------------------
```

设计SQL执行计划

```
SQL> select /*+ index("from$_subquery$_002".t_tab ind_tab_name) */ obj.owner, obj.object_name, created, blocks
  2  from t_objects obj,
  3  (select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A')
  6  where obj.object_id = 12345
  7  and obj.object_name = table_name;

未选定行
```

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	1210
* 1	HASH JOIN		1	194	1210
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	1207
5	UNION-ALL				
* 6	TABLE ACCESS BY INDEX ROWID	T_TAB	1184	95904	1182
7	INDEX FULL SCAN	IND_TAB_NAME	2367		11
* 8	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select /*+ index(t_tab ind_tab_name) */ obj.owner, obj.object_name, created, blocks
  2  from t_objects obj,
  3  (select owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A')
  6  where obj.object_id = 12345
  7  and obj.object_name = table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	194	43
* 1	HASH JOIN		1	194	43
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	115	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	40
5	UNION-ALL				
* 6	TABLE ACCESS FULL	T_TAB	1184	95904	15
* 7	TABLE ACCESS FULL	T_IND	2156	178K	25

设计SQL执行计划

```
SQL> select /*+ index(@v_1 t_tab ind_tab_name) */ obj.owner, obj.object_name, created, blocks
  2  from t_objects obj,
  3  (select /*+ qb_name(v_1) */ owner, table_name, tablespace_name, blocks from t_tab where temporary = 'N'
  4  union all
  5  select owner, index_name, tablespace_name, num_rows from t_ind where status = 'N/A')
  6  where obj.object_id = 12345
  7  and obj.object_name = table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		1	123	1210
* 1	HASH JOIN		1	123	1210
2	TABLE ACCESS BY INDEX ROWID	T_OBJECTS	1	44	2
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1
4	VIEW		3340	257K	1207
5	UNION-ALL				
* 6	TABLE ACCESS BY INDEX ROWID	T_TAB	1184	28416	1182
7	INDEX FULL SCAN	IND_TAB_NAME	2367		11
* 8	TABLE ACCESS FULL	T_IND	2156	66836	25

设计SQL执行计划

```
SQL> select /*+ push_pred(v) */ obj.owner, obj.object_name, created, v.blocks
  2  from t_objects obj, v_seg v
  3  where obj.object_id = 12345
  4  and obj.object_name = v.table_name;
```

未选定行

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		1	78	6 (0)
1	NESTED LOOPS		1	78	6 (0)
2	TABLE ACCESS BY INDEX ROWID BATCHED	T_OBJECTS	1	44	2 (0)
* 3	INDEX RANGE SCAN	IND_OBJ_ID	1		1 (0)
4	VIEW	V_SEG	1	34	4 (0)
5	UNION ALL PUSHED PREDICATE				
* 6	TABLE ACCESS BY INDEX ROWID BATCHED	T_TAB	1	24	2 (0)
* 7	INDEX RANGE SCAN	IND_TAB_NAME	1		1 (0)
* 8	TABLE ACCESS BY INDEX ROWID BATCHED	T_IND	1	31	2 (0)
* 9	INDEX RANGE SCAN	IND_IND_NAME	1		1 (0)

设计SQL执行计划

```
SQL> alter session set "_optimizer_push_pred_cost_based"=false;

Session altered.
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		286	42900	659K (1)
* 1	INDEX RANGE SCAN	IDX_CON_PROD__PRD_ID	1	12	1 (0)
.					
.					
.					
* 12	TABLE ACCESS BY INDEX ROWID	T_P_A	4812	145K	596 (1)
* 13	INDEX RANGE SCAN	IDX_P_A__DUE	14477		10 (0)
* 14	VIEW	V_CON_MAS	1	42	1 (0)
15	UNION-ALL PARTITION				
* 16	TABLE ACCESS BY INDEX ROWID	T_CON_MAS_LOG	1	44	1 (0)
* 17	INDEX RANGE SCAN	IDX_CON_MAS_L__LAST_CM	1		1 (0)
* 18	FILTER				
* 19	TABLE ACCESS BY INDEX ROWID	T_CON_MAS	1	42	1 (0)
* 20	INDEX UNIQUE SCAN	PK_T_CON_MAS	1		1 (0)
21	NESTED LOOPS				
22	NESTED LOOPS		1	34	4591 (1)
.					
.					
.					
* 35	INDEX UNIQUE SCAN	PK_T_COM_ORG	1		1 (0)
36	TABLE ACCESS BY INDEX ROWID	T_COM_ORG	1	15	1 (0)

设计SQL执行计划

掌握HINT的重要性：

- 掌控SQL执行方式
- 理解执行计划的效率差异
- 分析定位问题的利器
- 规避bug或性能问题
- 强制数据库按照设计思路运行

- 合理运用新特性
- 数据集整体处理
- 设计SQL执行计划
- 严格过滤数据

严格过滤数据

海盗分金问题：

有五个海盗，劫掠了100两金子，需要分赃。办法是抓阄，盗亦有道。

抓到第一个阄的人，可以先提出一个分配方案，如果他的方案被一半以上的人同意，就照他的方案分金子，否则，第一个人就要被杀掉。余下的人也照此办理。

我们的问题是：如果你是第一个人，你会提出怎样的分配方案?

为了分析问题更确定，我们假定每个人都是追求自己利益极大化的人。

严格过滤数据

一个人的分配原则：独占

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A)
  5  SELECT * FROM MAX_ONE;

      MAX1
----------
       100
```

严格过滤数据

最悲催的两个人分金方案：白忙

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING (P2, P1) USE_NL(P1) */ P2.N MAX2, P1.N MAX1
  7  FROM A P1, A P2
  8  WHERE P1.N + P2.N = 100
  9  AND P1.N >= (SELECT MAX1 FROM MAX_ONE)
 10  AND ROWNUM = 1)
 11  SELECT * FROM MAX_TWO;

      MAX2       MAX1
---------- ----------
         0        100
```

严格过滤数据

三个人的分金方案：拉拢

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING (P2, P1) USE_NL(P1) */ P2.N MAX2, P1.N MAX1
  7  FROM A P1, A P2
  8  WHERE P1.N + P2.N = 100
  9  AND P1.N >= (SELECT MAX1 FROM MAX_ONE)
 10  AND ROWNUM = 1),
 11  MAX_THREE AS
 12  (SELECT /*+ LEADING(P3, P2, P1) USE_NL(P1) */ P3.N MAX3, P2.N MAX2, P1.N MAX1
 13  FROM A P1, A P2, A P3, MAX_TWO
 14  WHERE P1.N + P2.N + P3.N = 100
 15  AND SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) >=0
 16  AND ROWNUM = 1)
 17  SELECT * FROM MAX_THREE;

      MAX3       MAX2       MAX1
---------- ---------- ----------
        99          1          0
```

严格过滤数据

四个人的分金方案：排挤

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING (P2, P1) USE_NL(P1) */ P2.N MAX2, P1.N MAX1
  7  FROM A P1, A P2
  8  WHERE P1.N + P2.N = 100
  9  AND P1.N >= (SELECT MAX1 FROM MAX_ONE)
 10  AND ROWNUM = 1),
 11  MAX_THREE AS
 12  (SELECT /*+ LEADING(P3, P2, P1) USE_NL(P1) */ P3.N MAX3, P2.N MAX2, P1.N MAX1
 13  FROM A P1, A P2, A P3, MAX_TWO
 14  WHERE P1.N + P2.N + P3.N = 100
 15  AND SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) >=0
 16  AND ROWNUM = 1),
 17  MAX_FOUR AS
 18  (SELECT /*+ LEADING(P4, P3, P2, P1) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P4.N MAX4, P3.N MAX3, P2.N MAX2, P1.N MAX
 19  FROM A P1, A P2, A P3, A P4, MAX_THREE
 20  WHERE P1.N + P2.N + P3.N + P4.N = 100
 21  AND SIGN(P3.N - MAX3) + SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) > 0
 22  AND ROWNUM = 1)
 23  SELECT * FROM MAX_FOUR;

      MAX4       MAX3       MAX2       MAX1
---------- ---------- ---------- ----------
        97          0          2          1
```

严格过滤数据

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING (P2, P1) USE_NL(P1) */ P2.N MAX2, P1.N MAX1
  7  FROM A P1, A P2
  8  WHERE P1.N + P2.N = 100
  9  AND P1.N >= (SELECT MAX1 FROM MAX_ONE)
 10  AND ROWNUM = 1),
 11  MAX_THREE AS
 12  (SELECT /*+ LEADING(P3, P2, P1) USE_NL(P1) */ P3.N MAX3, P2.N MAX2, P1.N MAX1
 13  FROM A P1, A P2, A P3, MAX_TWO
 14  WHERE P1.N + P2.N + P3.N = 100
 15  AND SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) >=0
 16  AND ROWNUM = 1),
 17  MAX_FOUR AS
 18  (SELECT /*+ LEADING(P4, P3, P2, P1) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P4.N MAX4, P3.N MAX3, P2.N MAX2, P1.N MAX1
 19  FROM A P1, A P2, A P3, A P4, MAX_THREE
 20  WHERE P1.N + P2.N + P3.N + P4.N = 100
 21  AND SIGN(P3.N - MAX3) + SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) > 0
 22  AND ROWNUM = 1),
 23  FIVE AS
 24  (SELECT /*+ LEADING(P5, P4, P3, P2, P1) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P5.N N5, P4.N N4, P3.N N3, P2.N N2, P1.N N1
 25  FROM A P1, A P2, A P3, A P4, A P5, MAX_FOUR
 26  WHERE P1.N + P2.N + P3.N + P4.N + P5.N = 100
 27  AND SIGN(P4.N - MAX4) + SIGN(P3.N - MAX3) + SIGN(P2.N - MAX2) + SIGN(P1.N - MAX1) >= 0
 28  AND ROWNUM = 1)
 29  SELECT * FROM FIVE;
        N5         N4         N3         N2         N1
---------- ---------- ---------- ---------- ----------
        97          0          1          0          2

已用时间:  00: 05: 51.49
```

严格过滤数据

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N, ROWNUM - 1 NA FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING(MAX_ONE, P2, P1) USE_NL(P2) USE_NL(P1) */ DECODE(P1.NA, MAX1, P2.N - 1, P2.N) MAX2, P1.NA MAX1
  7  FROM A P1, A P2, MAX_ONE
  8  WHERE P1.NA + P2.N = 100
  9  AND P1.NA >= MAX1
 10  AND ROWNUM = 1),
 11  MAX_THREE AS
 12  (SELECT /*+ LEADING(MAX_TWO, P3, P2, P1) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P3.N MAX3, P2.NA MAX2, P1.NA MAX1
 13  FROM A P1, A P2, A P3, MAX_TWO
 14  WHERE P1.NA + P2.NA + P3.N = 100
 15  AND P3.N + P2.NA <= 100
 16  AND CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1 THEN 1 ELSE -1 END >= 0
 17  AND ROWNUM = 1),
 18  MAX_FOUR AS
 19  (SELECT /*+ LEADING(MAX_THREE, P4, P3, P2, P1) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P4.N MAX4, P3.NA MAX3, P2.NA MAX2, P1.NA MAX1
 20  FROM A P1, A P2, A P3, A P4, MAX_THREE
 21  WHERE P1.NA + P2.NA + P3.NA + P4.N = 100
 22  AND P4.N + P3.NA <= 100
 23  AND P4.N + P3.NA + P2.NA <= 100
 24  AND CASE WHEN P3.NA > MAX3 THEN 1 ELSE -1 END + CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1 THEN 1 ELSE -1 END > 0
 25  AND ROWNUM = 1),
 26  MAX_FIVE AS
 27  (SELECT /*+ LEADING(MAX_FOUR, P5, P4, P3, P2, P1) USE_NL(P5) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P5.N N5, P4.NA N4, P3.NA N3, P2.NA N2, P1.NA N1
 28  FROM A P1, A P2, A P3, A P4, A P5, MAX_FOUR
 29  WHERE P1.NA + P2.NA + P3.NA + P4.NA + P5.N = 100
 30  AND P5.N + P4.NA <= 100
 31  AND P5.N + P4.NA + P3.NA <= 100
 32  AND P5.N + P4.NA + P3.NA + P2.NA <= 100
 33  AND CASE WHEN P4.NA > MAX4 THEN 1 ELSE -1 END + CASE WHEN P3.NA > MAX3 THEN 1 ELSE -1 END + CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1
THEN 1 ELSE -1 END >= 0
 34  AND ROWNUM = 1)
 35  SELECT * FROM MAX_FIVE;

        N5         N4         N3         N2         N1
---------- ---------- ---------- ---------- ----------
        97          0          1          0          2
已用时间:  00: 00: 00.03
```

严格过滤数据

```
SQL> WITH A AS
  2  (SELECT 100 - ROWNUM + 1 N, ROWNUM - 1 NA FROM DUAL CONNECT BY ROWNUM <= 101),
  3  MAX_ONE AS
  4  (SELECT MAX(N) MAX1 FROM A),
  5  MAX_TWO AS
  6  (SELECT /*+ LEADING(MAX_ONE, P2, P1) USE_NL(P2) USE_NL(P1) */ DECODE(P1.NA, MAX1, P2.N - 1, P2.N) MAX2, P1.NA MAX1
  7  FROM A P1, A P2, MAX_ONE
  8  WHERE P1.NA + P2.N = 100
  9  AND P1.NA >= MAX1
 10  AND ROWNUM = 1),
 11  MAX_THREE AS
 12  (SELECT /*+ LEADING(MAX_TWO, P3, P2, P1) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P3.N MAX3, P2.NA MAX2, P1.NA MAX1
 13  FROM A P1, A P2, A P3, MAX_TWO
 14  WHERE P1.NA + P2.NA + P3.N = 100
 15  AND P3.N + P2.NA <= 100
 16  AND CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1 THEN 1 ELSE -1 END >= 0
 17  AND ROWNUM = 1),
 18  MAX_FOUR AS
 19  (SELECT /*+ LEADING(MAX_THREE, P4, P3, P2, P1) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P4.N MAX4, P3.NA MAX3, P2.NA MAX2, P1.NA MAX1
 20  FROM A P1, A P2, A P3, A P4, MAX_THREE
 21  WHERE P1.NA + P2.NA + P3.NA + P4.N = 100
 22  AND P4.N + P3.NA <= 100
 23  AND P4.N + P3.NA + P2.NA <= 100
 24  AND CASE WHEN P3.NA > MAX3 THEN 1 ELSE -1 END + CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1 THEN 1 ELSE -1 END > 0
 25  AND ROWNUM = 1),
 26  MAX_FIVE AS
 27  (SELECT /*+ LEADING(MAX_FOUR, P5, P4, P3, P2, P1) USE_NL(P5) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P5.N N5, P4.NA N4, P3.NA N3, P2.NA N2, P1.NA N1
 28  FROM A P1, A P2, A P3, A P4, A P5, MAX_FOUR
 29  WHERE P1.NA + P2.NA + P3.NA + P4.NA + P5.N = 100
 30  AND P5.N + P4.NA <= 100
 31  AND P5.N + P4.NA + P3.NA <= 100
 32  AND P5.N + P4.NA + P3.NA + P2.NA <= 100
 33  AND CASE WHEN P4.NA > MAX4 THEN 1 ELSE -1 END + CASE WHEN P3.NA > MAX3 THEN 1 ELSE -1 END + CASE WHEN P2.NA > MAX2 THEN 1 ELSE -1 END + CASE WHEN P1.NA > MAX1 THEN 1 ELSE -1 END >= 0
 34  AND ROWNUM = 1),
 35  FIVE AS
 36  (SELECT /*+ LEADING(MAX_FOUR, MAX_FIVE, P5, P4, P3, P2, P1) USE_NL(MAX_FIVE) USE_NL(P5) USE_NL(P4) USE_NL(P3) USE_NL(P2) USE_NL(P1) */ P5.N N5, P4.NA N4, P3.NA N3, P2.NA N2, P1.NA N1
 37  FROM A P1, A P2, A P3, A P4, A P5, MAX_FOUR, MAX_FIVE
 38  WHERE P1.NA + P2.NA + P3.NA + P4.NA + P5.N = 100
 39  AND MAX_FIVE.N5 = P5.N
 40  AND P5.N + P4.NA <= 100
 41  AND P5.N + P4.NA + P3.NA <= 100
 42  AND P5.N + P4.NA + P3.NA + P2.NA <= 100
 43  AND CASE WHEN P4.NA > MAX_FOUR.MAX4 THEN 1 ELSE -1 END
 44      + CASE WHEN P3.NA > MAX_FOUR.MAX3 THEN 1 ELSE -1 END
 45      + CASE WHEN P2.NA > MAX_FOUR.MAX2 THEN 1 ELSE -1 END
 46      + CASE WHEN P1.NA > MAX_FOUR.MAX1 THEN 1 ELSE -1 END >= 0)
 47  SELECT * FROM FIVE;

        N5         N4         N3         N2         N1
---------- ---------- ---------- ---------- ----------
        97          0          1          0          2
        97          0          1          2          0
已用时间:  00: 00: 00.03
```

严格过滤数据

```
WITH A AS
.
.
.
FROM A P1, A P2, A P3, MAX_TWO
WHERE P1.NA + P2.NA + P3.N = 100
AND P3.N + P2.NA <= 100
.
.
.
WHERE P1.NA + P2.NA + P3.NA + P4.N = 100
AND P4.N + P3.NA <= 100
AND P4.N + P3.NA + P2.NA <= 100
.
.
.
WHERE P1.NA + P2.NA + P3.NA + P4.NA + P5.N = 100
AND P5.N + P4.NA <= 100
AND P5.N + P4.NA + P3.NA <= 100
AND P5.N + P4.NA + P3.NA + P2.NA <= 100
.
.
.
```

严格过滤数据

- 第一步结果集最小原则
- 能提前限制的条件不要放到后面
- 充分利用索引的ACCESS
- 不要忽略FILTER

总结

- 处理问题从整体上考虑，避免单条操作
- 第一步结果集最小原则，合理选择驱动表
- 利用新特性、分析函数避免重复多次的扫描，减少自关联
- 在每一个步骤上尽可能过滤掉无用数据
- 多写SQL：熟能生巧
- 多思考：算法为王
- 持之以恒：优化无止境

THANKS