NJDBUG
IT大咖说 知识共享平台

九次方大数据

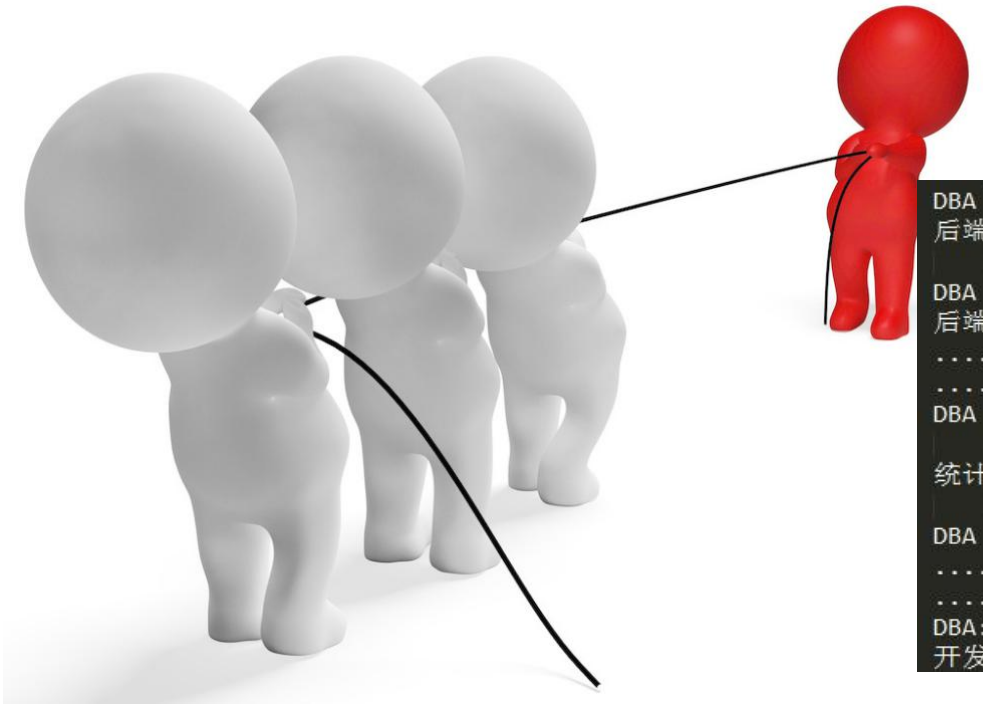
中国政府大数据资产运营商
全球大数据产业生态服务商

SQL优化及案例分析oracle11gR2

赵健

开发人员

DBA



SQL优化应该围绕业务场景来进行

```
DBA      : 你这个sql太长了，还能短些
后端开发: 业务表太多了,分段取再拼接，虽然DB这块是好了，
          服务器运行代码也一样消耗时间的，我切成几个业务大类试试
DBA      : sql是短些了，还是太慢，我加些索引试试
后端开发: 是快了不少，但是接口上看来还要等好几秒，代码维护上稍微困难些了
.....
.....
DBA      : 你这个数据同步的sql耗时不少啊，不能直接在前端请求时加上条件查询嘛，
           每次返回个20条不就很快了
统计开发: 我这个结果表关联的业务表多，还要保证准实时性，另外这个仅仅是
          查询请求时的基础表，真正查询的时候还要关联很多其他的表
DBA      : 那我给你个从库好了
.....
.....
DBA:  先这样吧
开发: 先这样吧
```

概念

优化器

优化器的目的是按照一定的判断原则来得到它认为的目标SQL在当前情形下最高效的执行路径

CBO 基于成本的优化器

Oracle里成本计算依赖于执行目标SQL所需耗费的I/O和CPU资源

RBO 基于规则的优化器

Oracle 10g开始，RBO由于明显缺陷已经不再被支持，可以使用RULE Hint继续使用

执行计划

执行目标SQL的步骤组合

执行计划顺序

执行计划关键字

Description	基数	字节
执行计划索引相关的名称		
执行计划与表连接相关的名称		
SORT JOIN/MERGE JOIN 排序合并连接		
NESTED LOOPS 嵌套循环连接		两个表做连接时依靠两层嵌套循环
HASH JOIN 哈希连接		两个表做连接是依靠哈希运算
3 NESTED LOOPS	101	20,705
1 TABLE ACCESS FULL	1	8
2 INDEX RANGE SCAN	1,517	
4 TABLE ACCESS BY INDEX ROWID	101	19,897

翻页查询

数据表记录数

第一种翻页 rownum在外层

第二种翻页 rownum在内层

```
select *
  from (select rownum as rn, t.* from ZJ_MS_DIRTYDATA_INFO t)
 where rn > 0
   and rn <= 30
```

所有行

文本 XML

Description	对象所有者	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	优化器	选项
MENT, GOAL = ALL_ROWS		1,083		51,312,753		1,081	ALL_ROWS	
	MONITOR	1,083		51,312,753	"RN"<=30...	1,081		
COUNT								
PX COORDINATOR								
PX SEND QC (RANDOM)	SYS	1,083		51,312,753		1,081		QC (RANDO
PX BLOCK ITERATOR		1,083		51,312,753		1,081		ITERATOR
TABLE ACCESS FULL	MONITOR	1,083		51,312,753		1,081		FULL

```
select *
  from (select rownum as rn, t.*
          from ZJ_MS_DIRTYDATA_INFO t
         where rownum <= 900030)
 where rn > 900000
```

优化器目标 所有行

树 HTML 文本 XML

Description	对象所有者	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	优化器	选项
SELECT STATEMENT, GOAL = ALL_ROWS		1,083		51,312,753		1,081	ALL_ROWS	
VIEW	MONITOR	1,083		51,312,753	"RN">900000	1,081		
COUNT STOPKEY					ROWNUM<=...			STOPKEY
PX COORDINATOR								
PX SEND QC (RANDOM)	SYS	1,083		51,312,753		1,081		QC (RANDO
COUNT STOPKEY					ROWNUM<=...			STOPKEY
PX BLOCK ITERATOR		1,083		51,312,753		1,081		ITERATOR
TABLE ACCESS FULL	MONITOR	1,083		51,312,753		1,081		FULL

排名函数的区别

- rownum
- row_number()
- rank()
- dense_rank()

rownum	根据查询后得到的结果自动加上去的，sql中排序筛选不会影响到rownum值
row_number()	排序字段值相同时，排名按照记录顺序递增
rank()	排序字段值相同时，排名相同，同时会和下个不同值空出排名
dense_rank()	排序字段值相同时，排名相同，同时会和下个不同值不会空出排名

```
select id,right_count,
ROW_NUMBER() OVER(Order By right_count) row_num_val  ,
RANK() OVER (Order By right_count) as rank_val,
DENSE_RANK() OVER (Order By right_count) dense_rank_val
from DG_MS_INFO
```

	ID	RIGHT_COUNT	ROW_NUM_VAL	RANK_VAL	DENSE_RANK_VAL
1	8889782	1004	1	1	1
2	8889738	1006	2	2	2
3	8887546	1060	3	3	3
4	8887511	1060	4	3	3
5	8889779	1075	5	5	4
6	8888380	1095	6	6	5
7	8884259	1146	7	7	6
8	8887663	1151	8	8	7
9	8887759	1162	9	9	8
10	8887854	1163	10	10	9

复合索引

复合索引建立

复合索引查询

复合索引将过滤和排序字段都考虑进去
复合索引的字段顺序要按照使用的频度来确定，并且区分度大的列作为前导列

在索引建立合适的前提下，最左前缀规则
查询或者过滤条件中使用到复合索引的前导列，才会走索引

```
select *
  from (s
                and policy_id = '7849'
 where rn
```

优化器目标 所有行

树 | HTML | 文本 | XML

Descript	IDX_ZJ_DIRTYDATA_A	Normal	SOURCE_TABLE_NAME, POLICY_ID, EXCUTE_TIME	CPU 耗费	过滤器谓词	IO 耗费	优化器	选项
SELEC				35,827,381		1,081	ALL_ROWS	
VIE				35,827,381	"RN">0	1,081		
(					ROWNUM<=30			STOPKEY
				35,827,381		1,081		
				35,827,381	ROWNUM<=30	1,081		ORDER BY STO...
				50,328,551		1,081		QC (RANDOM)
				50,328,551		1,081		ITERATOR
				50,328,551	"POLICY_...	1,081	ANALYZED	FULL

```
 select policy_id
from ZJ_MS_DIRTYDATA_INFO t
                where t.source_table_name = 'ali_hotel_room'

select source_table_name,policy_id,column_name
from ZJ_MS_DIRTYDATA_INFO t
                where t.source_table_name = 'ali_hotel_room'
                and t.policy_id = '7849'
```

树 | HTML | 文本 | XML

Description	对象所有者	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	优化器	选项
SELECT STATEMENT, GOAL = ALL_ROWS		108		1,695,096		108	ALL_ROWS	
VIEW	MONITOR	108		1,695,096	"RN">0	108		
COUNT STOPKEY					ROWNUM<=30			STOPKEY
VIEW	MONITOR	108		1,695,096		108		
COUNT								
TABLE ACCESS BY INDEX ...	MONITOR	108		1,695,096		108	ANALYZED	BY INDEX ROWID
INDEX RANGE SCAN	MONITOR	12	"SOUR...	389,057		12	ANALYZED	RANGE SCAN

表关联查询

一个“错误”的优化

```
select *
  from (select *
          from (select rownum as rn, t.*
                  from ZJ_MS_DIRTYDATA_INFO t,ZJ_MS_FLOW ma
                 where t.source_table_name = 'ali_hotel_room'
                   and t.policy_id = '7849'
                   and t.column_name=ma.column_name
                   and ma.is_valid=1
                 order by t.excute_time)
         where rownum <= 30)
 where rn > 0
```

优化器目标 所有行

树 | HTML | 文本 | XML

Description	对象所有者	基数	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	选项
SELECT STATEMENT, GOAL = ALL_ROWS		30	113		49,779,140		111	
VIEW	MONITOR	30	113		49,779,140	"RN">0	111	
COUNT STOPKEY						ROWNUM<=30		STOPKEY
VIEW	MONITOR	202	113		49,779,140		111	
SORT ORDER BY STOPKEY		202	113		49,779,140	ROWNUM<=30	111	ORDER BY STO..
COUNT								
HASH JOIN		202	112	"T"."COLUMN_NAME"="M...	17,825,296		111	
TABLE ACCESS FULL	MONITOR	2	3		36,127	"MA"."IS...	3	FULL
TABLE ACCESS BY INDEX ROWID	MONITOR	1,517	108		1,695,096		108	BY INDEX ROWID
INDEX RANGE SCAN	MONITOR	1,517	12	"T"."SOURCE_TABLE_NA...	389,057		12	RANGE SCAN

优化器目标 所有行

树 | HTML | 文本 | XML

Description	对象所有者	基数	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	选项
SELECT STATEMENT, GOAL = ALL_ROWS		2	112		17,676,446		111	
HASH JOIN		2	112	"from$_subqu...	17,676,446		111	
TABLE ACCESS FULL	MONITOR	1	3		36,127	"FE"."IS...	3	FULL
VIEW	MONITOR	30	108		1,695,096	"RN">0	108	
COUNT STOPKEY						ROWNUM<=30		STOPKEY
VIEW	MONITOR	1,517	108		1,695,096		108	
COUNT								
TABLE ACCESS BY INDEX ROWID	MONITOR	1,517	108		1,695,096		108	BY INDEX ROWID
INDEX RANGE SCAN	MONITOR	1,517	12	"T"."SOURCE_...	389,057		12	RANGE SCAN

查询转换

查询转换-子查询展开

```
select *
  from (select *
          from (select rownum as rn, t.*
                  from ZJ_MS_DIRTYDATA_INFO t
                 where t.source_table_name = 'ali_hotel_room'
                   and t.policy_id = '7849'
                   and t.column_name in (
                       select column_name from ZJ_MS_FLOW where is_valid=1
                   )
                 order by t.excute_time)
         where rownum <= 30)
 where rn > 0
```

Description	基数	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	选项
SELECT STATEMENT, GOAL = ALL_ROWS	30	112		65,523,542		110	
VIEW	30	112		65,523,542	"RN">0	110	
COUNT STOPKEY					ROWNUM<=30		STOPKEY
VIEW	101	112		65,523,542		110	
SORT ORDER BY STOPKEY	101	112		65,523,542	ROWNUM<=30	110	ORDER BY STOPKEY
COUNT							
NESTED LOOPS							
NESTED LOOPS	101	111		33,609,098		110	
SORT UNIQUE	1	3		36,127		3	UNIQUE
TABLE ACCESS FULL	1	3		36,127	"IS_VALID"=1	3	FULL
INDEX RANGE SCAN	1,517	11	"T"."SOURCE_TA...	382,786		11	RANGE SCAN
TABLE ACCESS BY INDEX ROWID	101	107		1,688,824	"T"."COLUMN_NAME"="COLUMN_NAME"	107	BY INDEX ROWID

树 | HTML | 文本 | XML

Description	基数	耗费	访问谓词	CPU 耗费	过滤器谓词	IO 耗费	选项
SELECT STATEMENT, GOAL = ALL_ROWS	30	153		2,267,367		153	
VIEW	30	153		2,267,367	"RN">0	153	
COUNT STOPKEY					ROWNUM<=30		STOPKEY
VIEW	101	153		2,267,367		153	
COUNT							
FILTER					EXISTS (SELECT /*+ NO_UNNEST *...		
TABLE ACCESS BY INDEX ROWID	1,517	108		1,725,444		108	BY INDEX ROWID
INDEX RANGE SCAN	1,517	12	"T"."SOURCE_TA...	389,057		12	RANGE SCAN
TABLE ACCESS FULL	1	3		36,128	"COLUMN_NAME"=:B1 AND "IS_VALID"=1	3	FULL

查询转换-视图

视图合并

谓词推入

优化器将目标SQL中视图定义的SQL语句拆开与外部查询进行基表的合并

优化器把目标SQL中视图定义的SQL当做一个独立的处理单元来执行，但是会把原本处于该视图外部查询中和该视图之间的连接条件推入到该

```
select /*+ push_pred(fa)*/ fa.* from
v_zj_dirydata fa,
 ZJ_MS_FLOW fb
 where fa.column_name(+)=fb.column_name
 and fb.column_name='chair'
```

优化器目标 所有行

树 | HTML | 文本 | XML

Description	对象名称	对象别名	过滤器谓词	基数	字节	耗费	CPU 耗费	IO 耗费
SELECT STATEMENT, GOAL = ALL_ROWS				34	22,746	11	120,379	11
NESTED LOOPS OUTER				34	22,746	11	120,379	11
TABLE ACCESS FULL	ZJ_MS_FLOW	FB@SEL$1	"FB"."COLUMN_NAME"='ch...	1	129	3	36,047	3
VIEW	V_ZJ_DIRYDATA	FA@SEL$1		34	18,360	8	84,332	8
UNION ALL PUSHED PREDICATE								
FILTER			'chair'="FB"."COLUMN_N...					
TABLE ACCESS BY INDEX ROWID	ZJ_MS_DIRTYDATA_INFO	T@SEL$2		17	11,339	4	42,166	4
INDEX RANGE SCAN	IDX_ZJ_DIRTYDATA_E	T@SEL$2		17		3	26,164	3
FILTER			'chair'="FB"."COLUMN_N...					
TABLE ACCESS BY INDEX ROWID	ZJ_MS_DIRTYDATA_INFO	T@SEL$3		17	11,339	4	42,166	4
INDEX RANGE SCAN	IDX_ZJ_DIRTYDATA_E	T@SEL$3		17		3	26,164	3

插入

- nologging
- 并行插入
- append

```
create table temp_dirtydata_info2 nologging parallel 4 as
select * from zj_ms_dirtydata_info
```

create table跳过数据缓存区，直接写入磁盘，适合海量迁移

大量的数据插入更新会触发lgwr也就是会触发把日志缓存区的数据从内存写到磁盘的REDO文件里

```
insert into temp_dirtydata_info2
select /*+ parallel(tt,4) */ * from zj_ms_dirtydata_info tt
```

插入数据的时候在表的高水位线之上直接插入数据，数据比较快

```
& 48:60  0:05  [redacted]  1001513 行被插入，耗时 5.397 秒
```

```
insert /*+ append */ into temp_dirtydata_info2
select /*+ parallel(tt,4) */ * from zj_ms_dirtydata_info tt
```

表的索引和约束在大批量的数据插入时维护的消耗也很多，待insert完成后再建索引和约束

```
& 56:1  0:01  [redacted]  1001513 行被插入，耗时 1.300 秒
```

删除

Delete

VS

trucate

水位线

```
DELETE      DML（data maintain Language)，语句每次删除一行，
            在事务日志中为所删除的每行记录一项，操作对象可以是table和view

TRUNCATE    DLL（data define language)自动提交,通过释放存储表数据所用的数据页，
            在事务日志中记录页的释放，操作对象只能是table
```

```
水位线代表的是表中的存储量，高水位线在日常的增删操作中只会上涨(数据记录删除了但是标记还在)，
所以当出现大批量delete的时候水位线不变，会造成查询的时候把删除记录的存储扫描一遍的误导

采用TRUNCATE语句删除一个表的数据的时候，会把水位线清空恢复为0
rebuild, truncate, shrink,move  等操作会降低高水位。
```

特例


```
select ma.source_table_name,
       ma.column_name as ma_column_name,
       mb.column_name
  from ZJ_MS_DIRTYDATA_INFO ma
  left join zj_ms_flow mb
    on ma.column_name = mb.column_name
   and ma.column_name = 'bed'
```

```
select ma.source_table_name,
       ma.column_name as ma_column_name,
       mb.column_name
  from ZJ_MS_DIRTYDATA_INFO ma
  left join zj_ms_flow mb
    on ma.column_name = mb.column_name
   where ma.column_name = 'bed'
```

```
select ma.source_table_name,
       ma.column_name as ma_column_name,
       mb.column_name
  from ZJ_MS_DIRTYDATA_INFO ma, zj_ms_flow mb
 where ma.column_name = case
         when ma.column_name = 'bed' then
          mb.column_name(+)
         else
          null
       end
```


贡献中国数据智慧
激活政府数据价值
构建全球数据生态