ORACLE®



ORACLE®

Explain Plan 命令说明

Nancy Guo 郭颖忠 Senior Sales Consultant

免责声明

- 本讲座旨在为您提供有关如何阅读 SQL 执行计划的说明,并帮助您确定该计划是否满足您的要求。
- 本讲座并不能使您一举成为优化器专家,也无法使您 具备轻松调整 **SQL** 语句的能力!

议题

- 什么是执行计划,如何生成执行计划?
- 一个优秀的优化器计划是什么样的?
- 理解执行计划
 - 基数
 - 访问方法
 - 联接顺序
 - 联接类型
 - 分区修剪
 - 并行度
- 执行计划示例



什么是执行计划,如何生成执行计划?

什么是执行计划?

- 执行计划显示在执行一条 SQL 语句时必须执行的详细 步骤
- 这些步骤表示为一组数据库运算符,这些运算符将使用和生成行
- 这些运算符及其实施的顺序由优化器使用查询转换及 物理优化技术的组合来确定
- 执行计划通常以表格的形式显示,但它实际上为树形

什么是执行计划?

查询

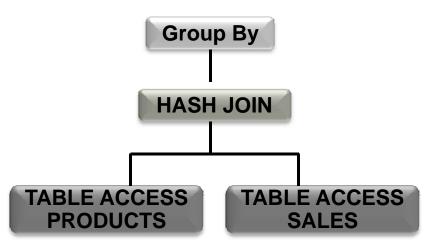
SELECT prod_category, avg(amount_sold)
FROM sales s, products p
WHERE p.prod_id = s.prod_id
GROUP BY prod_category;

执行计划的表格表示

Id Operation Name

O SELECT STATEMENT
HASH GROUP BY
HASH JOIN
TABLE ACCESS FULL PRODUCTS
PARTITION RANGE ALL
TABLE ACCESS FULL SALES

执行计划的树形表示



如何获取执行计划

可以使用两种方法查看执行计划

1.EXPLAIN PLAN 命令

• 显示一条 SQL 语句的执行计划,而不实际执行此语句

2.V\$SQL_PLAN

• 在 Oracle 9i 中引入的字典视图,它可显示已编译到游标缓存中一个游标的一条 SQL 语句的执行计划

使用 DBMS_XPLAN 包来显示执行计划

在某些情况下,使用 EXPLAIN PLAN 显示的计划可能与使用 V\$SQL_PLAN 显示的计划不同



```
示例 1 EXPLAIN PLAN 命令和 dbms_xplan.display 函数
SQL> EXPLAIN PLAN FOR
       SELECT prod_category, avg(amount_sold)
       FROM sales s, products p
       WHERE p.prod_id = s.prod_id
       GROUP BY prod_category;
 Explained
SQL> SELECT plan_table_output
  FROM table(dbms_xplan.display('plan_table',null,'basic'));
 Id Operation
                             Name
 0 SELECT STATEMENT
 1 HASH GROUP BY
 2 HASH JOIN
 3 TABLE ACCESS FULL PRODUCTS
 4 PARTITION RANGE ALL
    TABLE ACCESS FULL
                             SALES
```

如何获取执行计划

示例 2 生成并显示在会话中最后执行的 SQL 语句的执行计划 SQL>SELECT prod_category, avg(amount_sold) FROM sales s, products p WHERE p.prod_id = s.prod_id **GROUP BY prod_category**; no rows selected **SQL> SELECT plan table output** FROM table(dbms xplan.display cursor(null,null,'basic')); Name **Id Operation 0 SELECT STATEMENT** 1 HASH GROUP BY 2 HASH JOIN TABLE ACCESS FULL PRODUCTS 4 PARTITION RANGE ALL TABLE ACCESS FULL **SALES**

如何获取执行计划

示例 3 显示 V\$SQL_PLAN 中的任何其他语句的执行计划

1.直接:

SQL> SELECT plan_table_output FROM table(dbms_xplan.display_cursor('fnrtqw9c233tt',null,'basic'));

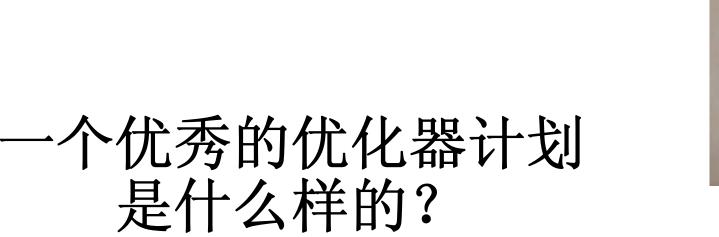
2.间接:

```
SQL> SELECT plan_table_output
FROM v$sql s, TABLE(dbms_xplan.display_cursor(s.sql_id,s.child_number, 'basic')) t
WHERE s.sql_text like 'select PROD_CATEGORY%';
```

ORACLE

DBMS_XPLAN 参数

- DBMS_XPLAN.DISPLAY 接受 3 个参数
 - 计划表的名称(默认为"PLAN_TABLE"),
 - statement_id (默认为 null)
 - 格式(默认为"TYPICAL")
- DBMS_XPLAN.DISPLAY_CURSOR 接受 3 个参数
 - SQL_ID (默认为此会话中最后一个执行的语句),
 - 子编号(默认为 0),
 - 格式 (默认为"TYPICAL")
- 格式是高度可定制的
 - Basic
 - Typical
 - All
 - 其他低级别参数可显示更多的详细信息





一个优秀的优化器计划是什么样的?

优化器有两个不同的目标

- 串行执行: 其关注的是开销
 - 开销越低越好
- 并行执行: 其关注的是性能
 - 速度越快越好

两个基本问题:

- 什么是开销?
- 什么是性能?

什么是开销?

- 优化器生成的神奇数字?
- 执行 SQL 语句所需的资源?
- 复杂计算的结果?
- 执行语句所需时间的估计?

实际定义

- 开销指的是所使用的工作单元或资源的数量
 - 优化器用 CPU、内存使用和 IO 作为工作单元
 - 开销是对执行操作时要使用的 CPU 和内存量以及磁盘 I/O 数的估计

开销是 Oracle 的一个内部量度

性能是什么?

- 完成尽可能多的查询?
- 使用最少的资源获得尽可能快的运行速度?
- 获得最佳的并发率?

实际定义

- 性能指的是对查询的最快响应时间
 - 目标是尽可能快地完成查询操作
 - 优化器不关注执行计划所需的资源



理解执行计划

SQL 执行计划

您在查看计划时能否确定以下项是否正确?

- 基数
 - 每个对象是否生成正确的行数?
- 访问方法
 - 是否以最好的方式访问数据?扫描?索引查找?
- 联接顺序
 - 是否以正确的顺序联接各表以便尽早尽多地消除数据?
- 联接类型
 - 是否使用了正确的联接类型?
- 分区修剪
 - 我执行过分区修剪吗? 是否消除了足够多的数据?
- 并行度

基数

什么是基数?

- 估算将返回的行数
- 单值谓词的基数 = 行的总数/不同值的总数
 - 例如: 共 100 行, 共 10 个不同值 => 基数 = 10 行
- 或者,如果为柱状图表示,则是行数*密度

为什么要关注?

- 它将影响所有方面! 访问方法、联接类型、联接顺序等哪些因素会导致基数出错?
- 统计信息陈旧/缺少
- 数据偏差
- 一个表有多个单列谓词
- where 子句谓词中包含函数
- 复杂表达式,其中包含来自不同表的列



Plan ha	sh value: 2011745446			1								
l Id l	Operation	Name	I Rows ∣	Bytes	Cost	(%CPU)	Time	Pstart	Pstop I	TQ	IIN-OUT	PQ Distrib
1 0 1	SELECT STATEMENT		l 1407 l	54873 I	166	(83)	00:00:02	I I	I		Ι	
I 1 I	PX COORDINATOR		1 1	ا				1 1	I			l
121	PX SEND QC (ORDER)	:TQ10003	l 1407 l	54873 I	166		00:00:02		I			I QC (ORDER)
I 3 I	VIEW		l 1407 l	54873 I	166		00:00:02		I		I PCWP	
1 4 1	SORT GROUP BY		1407	148KT			00:00:02		ļ.		I PCWP	
151	PX_RECEIVE		1407	148K I			00:00:02				I PCWP	
161	PX SEND RANGE	:TQ10002	1407	148K I			00:00:02				I P->P	RANGE
1 7 1	HASH GROUP BY		1407	148KI			00:00:02		!		I PCWP	
l* 8 I	HASH JOIN		1407	148KI	165		00:00:02		!		I PCWP	
9 1	PART JOIN FILTER CREATE	:BF0000	1408	78848 I			00:00:01	!!	!		I PCWP	
10	PX RECEIVE	T044444	1408	78848 I			00:00:01	!!	!		I PCWP	DART (UEU)
11	PX_SEND_PARTITION (KEY)	:TQ10001	1408	78848 I			00:00:01		!			PART (KEY)
* 12	HASH JOIN		1 1408	78848 I	/		00:00:01		!		I PCWP	
13	PX_RECEIVE	. TO4 AAAA	1 139 1	3614 I	2		00:00:01	!!	!		I PCWP	BROARCACT
14	PX SEND BROADCAST	:TQ10000	1 139 1	3614 I	2		00:00:01	!!	!		I P->P	BROADCAST
15	PX_BLOCK_ITERATOR	HOCON DOODE UP DEE	1 139 1	3614 I	2		00:00:01	!!	!	Q1,00		
16 17	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF		3614 I	4		00:00:01				I PCWP I PCWC	
1/ 18	PX_BLOCK_ITERATOR TABLE_ACCESS_FULL	T ACCT MACTED UD	1 1408	42240 I	4		00:00:01		4 I	Q1,01		
10 19	PX PARTITION RANGE SINGLE	T_ACCT_MASTER_HD	1408 18919	42240 I	157		00:00:01 00:00:02		64 I 14 I		I PCWP I PCWC	
l 20 l	PX PARTITION RANGE SINGLE PX PARTITION HASH JOIN-FILTER		16515 18919	960KT 960KT			00:00:02				I PCWC	l
20 * 21		T_TRAN_DETAIL_HD	18919	960KT			00:00:02		;вгоооот 56 I	01.02		
. 21 1		1_1VUM_DE1UTF_UD	1 10313	30001	137	(65)		1 33 1		QI,VZ	i FCWr	
			T T									

Predicate Information (identified by operation id):

估算返回行数的基数

使用简单的 SELECT COUNT(*) 从每个表应用任何属于该表的 WHERE 子句谓 词确定正确的基数

^{8 -} access("A"."ACCT_NUM"="C"."ACCT_NUM" AND "A"."CO_ID"="C"."CO_ID")
12 - access("A"."PCODE"="B"."PCODE")
21 - filter("C"."ASOF_YYYYMM"=200102 AND "C"."TRAN_AMT"<2000000000)

使用以下代码查看基数

SELECT /*+ gather_plan_statistics */

p.prod_name as product, sum(s.quantity_sold) as units,

FROM sales s, products p

WHERE s.prod_id =p.prod_id

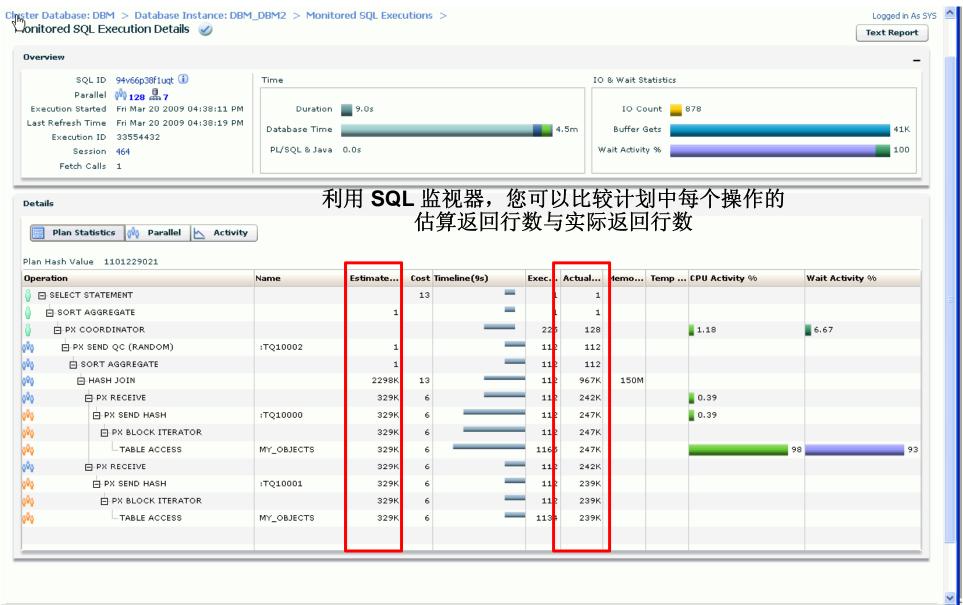
GROUP BY p.prod_name;

SELECT * FROM table (

DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'ALLSTATS LAST'));

|Plan hash value: 47|9755685 | | Starts | E-Rows | A-Rows | A-Time | Buffers | OMem I Id | Operation I Name 1Mem | Used-Mem 398 |**|**0:00:01.42 | 101 SELECT STATEMENT 398 [00:00:01.42] 101 I PX COORDINATOR PX SEND QC (RANDOM) :T010002 100:00:00.01 0 1 HASH GROUP BY 100:00:00.01 717KT 717KT 370K (0)I PX RECEIVE 100:00:00.01 :T010001 100:00:00.01 PX SEND HASH 0 [0:00:00.01] 717KT HASH GROUP BY 15289 100:00:00.01 791KT HASH JOIN 791KT 0 [0:00:00.01] 40960 | 40960 | 136864 BUFFER SORT 100:00:00.01 PX RECEIVE 766 PX SEND BROADCAST | :TQ10000 766 0 [0:00:00.01] 0 10 I 766 766 Id0:00:00.01 I 11 I TABLE ACCESS FULLI PRODUCTS 29 12 I 15289 I 0 | 0:00:00.01 | PX BLOCK ITERATOR * 13 I TABLE ACCESS FULL | SALES 0 [0:00:00.01] 15289 0.1

使用 SQL 监视器查看基数



有关解决基数问题的建议

原因	解决方法
统计信息陈旧/缺少	DBMS_STATS
数据偏差	创建一个柱状图*
一个表有多个单列谓词	使用 DBMS_STATS.CREATE_EXTENDED_STATS 创 建一个列组
在一个联接中使用多个列	使用 DBMS_STATS.CREATE_EXTENDED_STATS 创 建一个列组
包含函数的列	使用 DBMS_STATS.CREATE_EXTENDED_STATS 创 建有关包含函数的列的统计信息
复杂表达式,其中包含来自多个表 的列	使用 4 级或更高的动态抽样级别

ORACLE

中骨文

访问方法 — 获取数据

访问方法	解释
完整表扫描	读取表中所有行并过滤掉那些不符合 WHERE 子句谓词的行。用于索引、DOP 集等
按 ROWID 访问表	ROWID 指定含有所需行的数据文件和数据块以及该行在该块中的位置。当在索引或 WHERE 子句中提供 rowid 时使用
索引唯一扫描	将只返回一行。当语句中包含 UNIQUE 或 PRIMARY KEY 约束条件时使用,这些约束条件用于保证只访问一行
索引范围扫描	访问相邻索引项,可返回多个 ROWID 值。与等式一起用于非唯一索引,或与范围谓词一起用于唯一索引(<.>、between 等)
索引跳过扫描	如果前导列中只有很少的不同值,而非前导列中有许多不同的值,则跳过索引的前导部分,使用其余有用的部分
完整索引扫描	处理索引的所有叶块,但只有经过足够多的分支块才能找到第 1 个叶块。当所有需要的列都位于索引中且 order by 子句与索引结构匹配,或者排序合并联接已完成时,即可使用
快速完整索引扫描	扫描索引中的所有块,用来在所有需要的列都在索引中时代替 FTS。使用多块 IO,可以并行运行
索引联接	散列联接多个索引,这些索引一起包含有查询中引用的所有表列。 不会消除排序操作
位图索引	使用键值位图和映射函数,映射函数可将每个比特的位置转换成一个 rowid。 可以有效地合并对应于 WHERE 子句中的多个条件的索引

Plan hash value: 2011745446

l Id l	Operation	Name	I Rows I	Bytes	Cost	(%CPU)I	Time	Pstart	Pstop	TQ	IIN-OUT	l PQ Distrib
I 0 I	SELECT STATEMENT		1407	54873 I	166	(83) I	00:00:02	 	ا ا			
l 1 l	PX COORDINATOR		1	I		- 1		1	- 1		ı	
1 2 1	PX SEND QC (ORDER)	:TQ10003	1 1407	54873 I	166	(83)1	00:00:02	1	- 1	Q1,03	ß I P−>S	l QC (ORDER)
I 3 I	VIEW		1 1407	54873 I	166	(83)1	00:00:02	1	- 1	Q1,03	3 I PCWP	
1 4 1	SORT GROUP BY		1 1407	148KT	166	(83)1	00:00:02	1	- 1	Q1,03	3 I PCWP	
I 5 I	PX RECEIVE		1 1407	148KT	166	(83)1	00:00:02	1		Q1,03	3 I PCWP	
I 6 I	PX SEND RANGE	:TQ10002	1 1407	148KT	166	(83)1	00:00:02	1		Q1,02	? P->P	I RANGE
1 7 1	HASH GROUP BY		1 1407	148KT	166	(83)1	00:00:02	1	- 1	Q1,02	2 I PCWP	
I* 8 I	HASH JOIN		1 1407	148KT	165	(83)1	00:00:02	1	- 1	Q1,02	2 I PCWP	
191	PART JOIN FILTER CREATE	:BF0000	1 1408	78848 I	7	(29)1	00:00:01	1	- 1	Q1,02	? I PCWP	
I 10 I	PX RECEIVE		1 1408	78848 I	7	(29)1	00:00:01	1	- 1	Q1,02	2 I PCWP	
l 11 l	PX SEND PARTITION (KEY)	:TQ10001	1 1408	78848 I	7	(29)1	00:00:01	1	- 1	Q1,01	. I P->P	l Part (Key)
I* 12 I	HASH JOIN		1 1408	78848 I	7	(29)1	00:00:01	1	- 1	Q1,01	. I PCWP	
l 13 l	PX RECEIVE		1 139 1	3614 I	2	(0)1	00:00:01	1	- 1	Q1,01	. I PCWP	
I 14 I	PX SEND BROADCAST	:TQ10000	1 139 1	3614 I	2	(0)1	00:00:01	1	- 1	Q1,00) P->P	I BROADCAST
l 15 l	PX_BLOCK_ITERATOR		1 139 1	3614 l	2	(0)1	00:00:01	1	- 1	Q1,00) I PCWC	l
l 16 l	TABLE ACCESS FULL ←	HOCAN_PCODE_HD_REF	130	3014 +	- 2	(0)+	00:00:01	+		——Q1, 00) I PCWP	l
l 17 l	PX REHEK LIFRHILIK		1 1408	42240 I	4	(25)1	00:00:01	1 1	4 1	01,01	. I PCWC	l
l 18 l	TABLE ACCESS FULL ←	T_ACCT_HASTER_HD	1400	42240 1	4	(25) t	00,00,01	1 1	64	1,01	. I PCWP	
l 19 l	PX PARTITION RANGE SINGLE		18919	960KT	157	(85) I	00:00:02	14 1	14	01,02	2 I PCWC	
I 20 I	<u>PX_PARTITION_HASH_</u> JOIN-FILTER		18919	960KT	157	(85) I	00:00:02	1:BF00001	:BF00001	01,02	? I PCWC	
I* 21 I	TABLE ACCESS FULL ←	T_TRAN_DETAIL_HD	1 10010	960KH	157	(05) 1	00:00:02	53	50	 01,02	? I PCWP	l

Predicate Information (identified by operation id):

8 - access("A","ACCT_NUM"="C","ACCT_NUM" AND "A","CO_ID"="C","CO_ID")
12 - access("A","PCODE"="B","PCODE")

21 - filter("C", "ASOF_YYYYMM"=200102 AND "C", "TRAN_AMT"<2000000000)

查看 Operation 部分以了解对象的访问方式

如果发现使用了错误的访问方法,请检查基数、联接顺序...

访问方法示例

一个 countries 表包含 10K 行,并且有一个 country_id 主键。 您希望对以下查询使用什么计划?

Select country_id, name from countries where country_id in ('AU','FR','IE');

l Id l	Operation	 	Name	ı	Rows	1	Bytes	Cost	(%CPU)I	Time	
0 1 2 * 3	SELECT STATEMENT INLIST ITERATOR TABLE ACCESS BY INDEX INDEX UNIQUE SCAN		COUNTRIES COUNTRY_PK	i		i	136K 136K 		(0) I	00:00:01 00:00:01 00:00:01	1

Select country_id, name from countries where country_id between 'AU' and 'IE';

Id Operation	l Name	I	Rows	I	Bytes	Cost	(%CPU)I	Time	I
0 SELECT STATEMENT 1 TABLE ACCESS BY INDE * 2 INDEX RANGE SCAN	I K ROWIDI COUNTRIES I COUNTRY_PK	i		•	136KT 136KT T	2 2 1	1(0)	00:00:01 00:00:01 00:00:01	

Select country_id, name from countries where name='USA';

ΙΙ	d	l	Operation	l	Name	l	Rows	1	Bytes	1	Cost	(%CPU)	Time	l	
 *	0		SELECT STATEMENT TABLE ACCESS FUL		COUNTRIES	 	1 1	 	14 14		 2 2	2 (0)	00:00:01 00:00:01	 	ORACLE [®]
· 						·									电电文

有关解决访问的建议

问题	原因
使用表扫描, 而不是索引扫描	DOP 针对表,而不是索引或 MBRC 值
采用错误的索引	统计信息陈旧/缺少 采用了匹配最多列的索引 完整索引访问方式的开销要比索引查找后跟 表访问方式的开销低

联接类型

- 一个联接可从多个表中检索数据
- 可能的联接类型包括 嵌套循环联接 散列联接 智能化分区联接 排序合并联接 笛卡尔联接 外联接

联接类型 — 联接可从多个表中检索数据

访问方法	解释
嵌套循环联接	对于外部表中的每一行,Oracle 访问内部表中的所有行。当联接多个小型数据子集,并且有一个高效的方法(索引查找)来访问第二个表时,这非常有用
散列联接	对两个表中较小的表执行扫描,使用结果行根据内存中的联接键创建散列表。然 后扫描较大的表,对结果行的联接列执行散列操作,并用其值探测散列表以查 找匹配的行。 对于较大的表和 if equality 谓词,这非常有用
排序合并联接	包括两个步骤: 1. 排序联接操作:基于联接键对两个输入都进行排序。 2. 合并联接操作:将排序的列表合并在一起。 当两个表之间的联接条件是不相等条件时,这非常有用
笛卡尔联接	将来自一个数据源的每一行与来自另一个数据源的每一行进行联接,生成这两个 数据集的笛卡尔乘积。只有在表非常小时才适用。如果没有在查询中指定任 何联接条件,则这是唯一的选择
外联接	返回所有满足联接条件的行,并从没有 (+) 的表中返回所有这样的行:在另一个表中没有满足联接条件的行

Plan hash value: 2011745446

Id I	Operation	l Name	l Rows	l Bytes I	Cost (%CPU)∣	Time	Pstart	Pstop I	TQ	IIN-OUT	PQ Distri
0 1	SELECT STATEMENT	 	I 1407	I 54873	166	(83)1	00:00:02	 	 I		 	
1 I	PX COORDINATOR	l				- 1		1 1	- 1		1	
2 1	PX SEND QC (ORDER)	I :TQ10003	I 1407	l 54873 l	166	(83)1	00:00:02	1 1	I	Q1,03	1 P->S 1	QC (ORDE
3 I	VIEW	I	I 1407	l 54873 l	166	(83)1	00:00:02	1 1	I	Q1,03	I PCWP I	
4 I	SORT GROUP BY	I	I 1407	l 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP I	
5 I	PX RECEIVE	I	I 1407	l 148KI	166	(83) [00:00:02	1 1	- 1	Q1,03	I PCWP I	
6 I	PX SEND RANGE	l :TQ10002	I 1407	l 148KI	166	(83) I	00:00:02	1 1	I	Q1,02	P->P	RANGE
7 I	HASH GROUP BY	I	I 1407	l 148KI	166	(83) I	00:00:02	1 1	I	Q1,02	I PCWP I	
* 8 I	HASH JOIN ←		1407	140KI	165	(83) l	00:00:02	+	-	 01,02	I PCWP I	
9 I	PART JUIN FILTER CREATE	l :BF0000	l 1408	l 78848 l	7	(29)1	00:00:01	1 1	I	01,02	I PCWP I	
10 I	PX RECEIVE	I	I 1408	l 78848 l	7	(29)1	00:00:01	1 1	I	1,02	I PCWP I	
11 I	PX SEND PARTITION (KEY)	I :TQ10001	I 1408	l 78848 l	7	(29)1	00:00:01	1 1	- 1	1,01	1 P->P 1	PART (KE'
' 12 T	HASH JOIN ◀		1400	70040	7	(23) I	00:00:01	+	+	1,01	I PCWP I	
13 I	PX RECEIVE	I	l 139	l 3614 l	2	(0) I	00:00:01	1 1	I	1,01	I PCWP I	
14 I	PX SEND BROADCAST	I :TQ10000	l 139	l 3614 l	2	(0) I	00:00:01	1 1	I	1,00	1 P->P 1	BROADCAS'
15 I	PX BLOCK ITERATOR	I	l 139	l 3614 l	2	(0)1	00:00:01	1 1	I	1,00	I PCWC I	
16 I	TABLE ACCESS FULL	I HOGAN_PCODE_HD_REF	l 139	l 3614 l	2	(0)1	00:00:01	1 1	I	1,00	I PCWP I	
17 I	PX BLOCK ITERATOR	I	I 1408	l 42240 l	4	(25)1	00:00:01	1 1	4 1	1,01	I PCWC I	
18 I	TABLE ACCESS FULL	I T_ACCT_MASTER_HD	I 1408	l 42240 l	4	(25)1	00:00:01	1 1	64 I	1,01	I PCWP I	
19 I	PX PARTITION RANGE SINGLE	I	l 18919	I 960KT	157	(85)1	00:00:02	l 14 l	14 I	1,02	I PCWC I	
20 I	PX PARTITION HASH JOIN-FILTER	I	l 18919	I 960KT	157	(85)1	00:00:02	1:BF00001	:BF00001	1,02	I PCWC I	
* 21 T	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	l 18919	I 960KT	157	(85)1	00:00:02	I 53 I	56 I	1,02	I PCWP I	

Predicate Information (identified by operation id):

8 - access("A","ACCT_NUM"="C","ACCT_NUM" AND "A","CO_ID"="C","CO_ID")
12 - access("A","PCODE"="B","PCODE")

21 - filter("C", "ASOF_YYYYMM"=200102 AND "C", "TRAN_AMT"<2000000000)

查看 Operation 部分以检查是否使用了正确联接类型

如果使用了错误的联接类型,则返回并检查所编写的语句是否正确,以及估算 的基数是否正确

ORACLE

应为此查询使用什么联接类型?

SELECT e.name, e.salary, d.dept_name FROM hr.employees e, hr.departments d WHERE d.dept_name IN ('Marketing','Sales') AND e.department_id=d.department_id;

Employees 有 107 行 Departments 有 27 行

Employees 和 Departments 之间基于 dept_id 的外键关系

Name	-	Rows	I	Bytes	I	Cost(%C	PU)	Time	I
 	·	19	·	722	 	3	(0)	00:00:01	- .
1	- [1		1		- 1		-
1	- [19	-	722	1	3	(0) [00:00:01	.
DEPARTMENTS	- [2		32	1	2	(0) [00:00:01	.
EMP DEPARTMENT IX	1	10	1		Ι	0	(0) [00:00:01	.
DEX ROWID EMPLOYEES	- 1	10	1	220	1	1	(0) [00:00:01	.
V			19 19 19 19 DEPARTMENTS 2 EMP_DEPARTMENT_IX 10	19 19 19 19 LL DEPARTMENTS 2 EMP_DEPARTMENT_IX 10	19 722 19 722 19 722 19 722 10 DEPARTMENTS 2 32	19 722 19 722 19 722 19 722 DEPARTMENTS 2 32 EMP_DEPARTMENT_IX 10	19 722 3 19 722 3 19 722 3 LL DEPARTMENTS 2 32 2 N EMP_DEPARTMENT_IX 10 0	19 722 3 (0)	19 722 3 (0) 00:00:01 19 722 3 (0) 00:00:01 19 722 3 (0) 00:00:01 DEPARTMENTS 2 32 2 (0) 00:00:01 EMP_DEPARTMENT_IX 10 0 (0) 00:00:01

Predicate Information (identified by operation id):

^{3 -} filter ("D"."DEPARTMENT NAME"='Marketing' OR "D"."DEPARTMENT NAME"='Sales')

^{4 -} access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

应为此查询使用什么联接类型?

SELECT o.customer_id, l.unit_price * l.quantity FROM oe.orders o ,oe.order_items l WHERE l.order_id = o.order_id;

Orders 有 105 行 Order Items 有 665 行

Id Operation	Name	Rows Bytes Cost (%CPU)
O SELECT STATEMENT * 1 HASH JOIN 2 TABLE ACCESS FULL 3 TABLE ACCESS FULL		665 13300 8 (25) 665 13300 8 (25) 105 840 4 (25) 665 7980 4 (25)

Predicate Information (identified by operation id):

^{1 -} access("L"."ORDER_ID"="O"."ORDER_ID")

应为此查询使用什么联接类型?

SELECT o.order_id,0.order_date,e.name FROM oe.orders o , hr.employees e;

Orders 有 105 行 Employees 有 107 行

|Plan hash value: 3229651169

Id Operation	l —				_									
1 MERGE JOIN CARTESIAN 11235 120K 33 (7) 00:00:01 2 INDEX FULL SCAN ORDER_PK 105 420 1 (0) 00:00:01 3 BUFFER SORT 107 749 32 (7) 00:00:01		Id		Operation	<u> </u>	Name	1	Rows	l	Bytes	Cost	(%CPU)I	Time	
1 1 110511 1101 1 055 001111 511 51111 521 1 110 1 4 (4)1 444444		1 2 3	 	MERGE JOIN CARTESIAN INDEX FULL SCAN BUFFER SORT			 	11235 105	 	120K l 420	33 1	(7) (0) 2 (7)	00:00:01 00:00:01 00:00:01	

应为此查询使用什么联接类型?

SELECT d.department_id,e.emp_id
FROM hr.employees e FULL OUTER JOIN hr.departments d
ON e.department_id = d.department_id
ORDER BY d.department_id;

Employees 有 107 行 Departments 有 27 行

Employees 和 Departments 之间基于 dept_id 的外键关系

I	Id	ı	Operation	 -	Name	 	Rows	I	Bytes	 	Cost	(%CPU)	Time		I
I	0)	SELECT STATEMENT SORT ORDER BY	 		 	122 122	•	4758 4758	 		5 (34) 5 (34)		0:01 0:01	•
į			VIEW	•	VW_FOJ_O	į	122	i	4758		5	5 (20)	00:0	0:01	İ
* 		 - -	HASH JOIN FULL OUTER INDEX FAST FULL SCAN	•	DEPT_ID_PK	1	122 27	•	1342 108	 	2	5 (20) 2 (0)		0:01	•
I	5	5	TABLE ACCESS FULL		EMPLOYEES	I	107	I	749	I	2	(0)	00:0	0:01	I

Predicate Information (identified by operation id):

^{3 -} access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

联接顺序

在多表语句中对多个表进行联接的顺序

- 应该从可消除最多行数的表开始操作
- 将受可用访问方法的很大影响
- 一些基本规则
- 总是最先执行最多生成一行的联接
- 当使用外联接时,在谓词中,含有此外联接运算符的表必须 位于其他表之后
- 如果不能执行视图合并,在联接视图外部的表之前联接视图内部的所有表

Plan hash value: 2011745446

Id I	Operation	l Name	I Rows	Bytes	Cost	(%CPU)I	Time	Pstart	Pstop	TQ	IIN-OUT	l PQ Distrib
0 1	SELECT STATEMENT	 	I 1407	I 54873	166	1 (83) I	00:00:02					
1 I	PX COORDINATOR		I	l I		- 1		1 1			I	
2 1	PX SEND QC (ORDER)	:TQ10003	I 1407	I 54873 I	166		00:00:02					l QC (ORDER)
3 I	VIEW		I 1407	I 54873 I	166		00:00:02				I PCWP	l
4 1	SORT GROUP BY	l	I 1407	I 148KI	166	(83)1	00:00:02	1 1		~-·-	I PCWP	l
5 I	PX RECEIVE		I 1407	I 148KI	166	(83) I	00:00:02	1 1		Q1,03	I PCWP	
6 I	PX SEND RANGE	:TQ10002	I 1407	I 148KI	166	(83)1	00:00:02	1 1		Q1,02	I P−>P	I RANGE
7 I	HASH GROUP BY		I 1407	I 148KI	166	(83)1	00:00:02	1 1		Q1,02	I PCWP	
* 8 I	HASH JOIN	l	I 1407	I 148KI	165	(83)1	00:00:02	1 1		Q1,02	I PCWP	l
9 I	PART JOIN FILTER CREATE	:BF0000	I 1408	I 78848 I	7	(29)1	00:00:01	1 1		Q1,02	I PCWP	l
10 I	PX RECEIVE	1	I 1408	I 78848 I	7	(29)1	00:00:01	1 1		Q1,02	I PCWP	l
11 I	PX SEND PARTITION (KEY)	:TQ10001	I 1408	I 78848 I	- 7	(29)1	00:00:01	1 1		Q1,01	I P->P	l Part (Key)
* 12 I	HASH JOIN	l ·	I 1408	I 78848 I	- 7	(29)1	00:00:01	1 1		Q1,01	I PCWP	i
13 I	PX RECEIVE	l	I 139	I 3614 I	2	(0)1	00:00:01	1 1		Q1,01	I PCWP	l
14 I	PX SEND BROADCAST	:TQ10000	I 139	I 3614 I	2	(0)	00:00:01	1 1		Q1,00	I P->P	I BROADCAST
15 I	PX BLOCK ITERATOR	l	I 139	I 3614 I	2	(0)1	00:00:01	1 1		Q1,00	I PCWC	l
16 I	1 TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	I 4139 -	3014	2	(0)+	00:00:01	+		01,00	+ PCWP	l
17 I	PX BLOCK ITERATOR	I	I 1408	I 42240 I	4	(25)1	00:00:01	1 1 1	4	Q1,01	I POWO	l
18 I	2 TABLE ACCESS FULL	T_ACCT_MASTER_HD	I 1408	I 42240 I	4	(25)1	00:00:01	1 1 1	64	Q1,01	I PCWP	l
19 I	PX PARTITION RANGE SINGLE		I 18919	1 960KT	157	(85)1	00:00:02	I 14 I	14	Q1,02	I POWO	l
20 I	PX PARTITION HASH JOIN-FILTER	I	I 18919		157			1:BF00001	:BF0000			l
* 21 I	3 TABLE ACCESS FULL	T_TRAN_DETAIL_HD	I 18919		157		00:00:02	I 53 I	56	Q1,02	I PCWP	l

Predicate Information (identified by operation id):

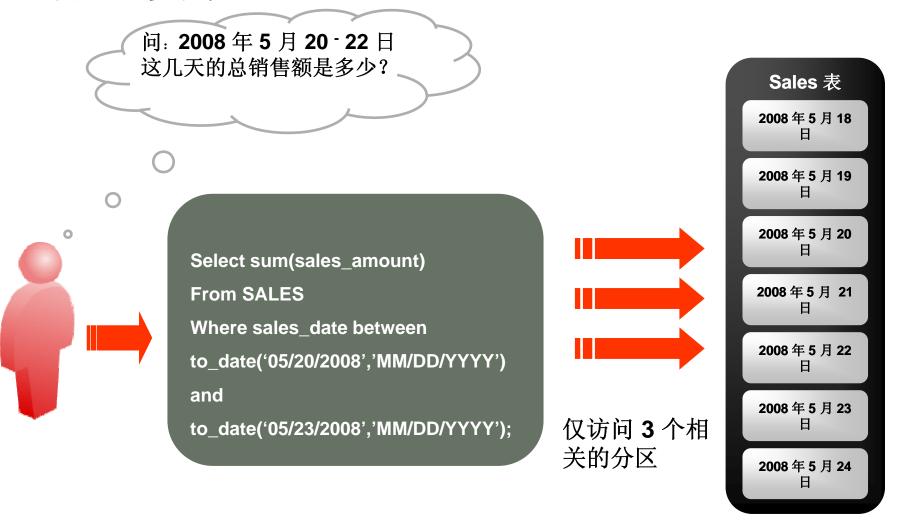
```
8 - access("A"."ACCT_NUM"="C"."ACCT_NUM" AND "A"."CO_ID"="C"."CO_ID")
12 - access("A"."PCODE"="B"."PCODE")
```

需要从可最大程度减少结果集行数 的表开始操作

如果联接顺序不正确, 请检查统计信息、基数及访问方法

^{21 -} filter("C"."ASOF_YYYYMM"=200102 AND "C"."TRAN_AMT"<20000000000)

分区修剪



分区修剪

Plan hash value: 2011745446

Id	Operation	Name	l Rows	l Bytes I	Cost ((%CPU) I	Time I	Pstart	Pstop	TQ	IIN-OUTI	PQ Distrib
0	SELECT STATEMENT	 	I 1407	 I 54873 I	166	(83) I	00:00:02 I		 		 	
1	PX COORDINATOR					I	- 1		l I		1 1	
2	PX SEND QC (ORDER)	:TQ10003	l 1407		166		00:00:02 I		l I			QC (ORDER
3	VIEW		l 1407		166		00:00:02 I		l I		I PCWP I	
4	SORT GROUP BY		l 1407	l 148KT	166		00:00:02 I		l I		I PCWP I	
5	PX RECEIVE		I 1407		166		00:00:02 I		l I		I PCWP I	
6	PX SEND RANGE	:TQ10002	l 1407	l 148KT	166	(83) I	00:00:02 I		l I		P->P	RANGE
- 7	HASH GROUP BY		l 1407	l 148KT	166		00:00:02 I		l I		I PCWP I	
8	HASH JOIN		I 1407	l 148KI	165		00:00:02 I		l I		I PCWP I	
9	PART JOIN FILTER CREATE	:BF0000	l 1408	l 78848 l	7	(29)1	00:00:01 I		l I	Q1,02	I PCWP I	
10	PX RECEIVE		l 1408	l 78848 l	7	(29)1	00:00:01		l I	Q1,02	I PCWP I	
11	PX SEND PARTITION (KEY)	:TQ10001	I 1408	l 78848 l	7	(29)1	00:00:01 I		l I			PART (KE'
12	HASH JOIN		I 1408	l 78848 l	- 7	(29)1	00:00:01		l I	Q1,01	I PCWP I	
13	PX RECEIVE		l 139	3614	2	(0) I	00:00:01		l I	Q1,01	I PCWP I	
14	PX SEND BROADCAST	:TQ10000	l 139	3614	2	(0) I	00:00:01 I		l I	Q1,00	1 P->P 1	BROADCAS*
15	PX BLOCK ITERATOR		l 139	3614	2	(0)1	00:00:01		l I	Q1,00	I PCWC I	
16	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	l 139	3614	2		00:00:01 I		l I	Q1,00	I PCWP I	
17	PX BLOCK ITERATOR		I 1408	l 42240 l	4	(25)1	00:00:01 I	1	141	Q1,01	I PCWC I	
18	TABLE ACCESS FULL	T_ACCT_MASTER_HD	I 1408	I 42240 I	4	(25)1	00:00:01 I	1	l 64 I	Q1,01	I PCWP I	
19	PX PARTITION RANGE SINGLE		l 18919	960KT	157	(85) I	00:00:02 I	14	l 14 l	Q1,02	I PCWC I	
20	PX PARTITION HASH JOIN-FILTER		l 18919	960KT	157	(85)1	00:00:02 I	BF0000	:BF00001	Q1,02	I PCWC I	
	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	l 18919		157	(85)1	00:00:02 I	53	i 56 I	01.02	I PCWP I	

21 - filter("C", "ASOF_YYYYMM"=200102 AND "C", "TRAN_AMT"<2000000000)

Pstart 和 Pstop 列出了查询操作所访问的分区

如果显示了单词"KEY",则意味着将在运行时确定所访问的分区

并行度

目标是并行执行计划的所有方面

- 确定是使用一组还是多组并行服务器进程
 - 生产者和使用者
- 确定计划中是否有任何串行运行的部分

Plan hash value: 2011745446

l Id l	Operation	l Name	l Rows	l Bytes I C	ost (%CPU)I	Time	Pstart	Pstop	TQ	I IN-OUT I	PQ Distrib
I 0	SELECT STATEMENT	 	I 1407	I 54873 I	166	(83)1	00:00:02	I I	I			
1 1	PX COORDINATOR		1	l l		- 1		1 1	I			
1 2 5	PX SEND QC (ORDER)	:TQ10003	I 1407	l 54873 l	166	(83)1	00:00:02	1 1	I	Q1,03		QC (ORDER)
1 3	VIEW	l	I 1407	l 54873 l	166	(83)1	00:00:02	1 1	I	Q1,03	I PCWP I	
1 4	SORT GROUP BY	l	I 1407	l 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP I	
I 5 I	PX RECEIVE	l	I 1407	l 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP I	
161	PX SEND RANGE	:TQ10002	I 1407	l 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,02	I P->P I	RANGE
1 7 5	HASH GROUP BY		I 1407	l 148KI	166	(83) I	00:00:02	1 1		Q1,02	I PCWP I	
I* 8 I	HASH JOIN	l	I 1407	l 148KI	165	(83)1	00:00:02	1 1	- 1	Q1,02	I PCWP I	
1 9 '	PART JOIN FILTER CREATE	l :BF0000	l 1408	l 78848 l	- 7	(29)1	00:00:01	1 1	- 1	Q1,02	I PCWP I	
I 10 I	PX RECEIVE	l	I 1408	l 78848 l	- 7	(29)1	00:00:01	1 1	- 1	Q1,02	I PCWP I	
11	PX SEND PARTITION (KEY)	:TQ10001	I 1408	l 78848 l	- 7	(29)1	00:00:01	1 1	- 1	Q1,01	P->P	PART (KEY)
I* 12 I	HASH JOIN		I 1408	l 78848 l	7	(29)1	00:00:01	1 1	- 1	Q1,01	I PCWP I	
I 13 I	PX RECEIVE	l	l 139	l 3614 l	2	(0)1	00:00:01	1 1	- 1	Q1,01	I PCWP I	
I 14 I	PX SEND BROADCAST	:TQ10000	l 139	l 3614 l	2	(0)1	00:00:01	1 1	- 1	Q1,00	I P->P I	BROADCAST
l 15 l	PX BLOCK ITERATOR	l	l 139	l 3614 l	2	(0)1	00:00:01	1 1	- 1	Q1,00	I PCWC I	
I 16 I	TABLE ACCESS FULL	I HOGAN_PCODE_HD_REF	l 139	l 3614 l	2	(0)1	00:00:01	1 1	- 1	Q1,00	I PCWP I	
l 17 l	PX BLOCK ITERATOR	l	I 1408	l 42240 l	4	(25)1	00:00:01	1 1	4 1	Q1,01	I PCWC I	
I 18 I	TABLE ACCESS FULL	I T_ACCT_MASTER_HD	I 1408	l 42240 l	4	(25)1	00:00:01	1 1	64 I	Q1,01	I PCWP I	
I 19 I	PX PARTITION RANGE SINGLE		l 18919	1 960KT	157	(85)1	00:00:02	I 14 I	14 I	Q1,02	I PCWC I	
1 20 1	PX PARTITION HASH JOIN-FILTER	l	l 18919	1 960KT	157	(85)1	00:00:02	1:BF00001	:BF00001	Q1,02	I PCWC I	
l* 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	l 18919	1 960KT	157	(85)1	00:00:02	l 53 l	56 I	Q1,02	I PCWP I	

Predicate Information (identified by operation id):

8 - access("A","ACCT_NUM"="C","ACCT_NUM" AND "A","CO_ID"="C","CO_ID")
12 - access("A","PCODE"="B","PCODE")

21 - filter("C", "ASOF_YYYYMM"=200102 AND "C", "TRAN_AMT"<2000000000)

IN-OUT 列显示出哪些步骤是并行运 行的,以及使用的是一组还是多组 并行服务器进程

如果任何行显示为以字母 "S" 开头,则表明将以串行方式运行,检查所使用的 每个表和索引的 DOP

ORACLE

甲骨文

确定在计划的扫描过程中的并行度粒度

- 数据被划分为以下粒度
 - 块范围
 - 分区
- 将为每个并行服务器分配一个或多个粒度
- 该粒度方法在 Operation 部分中的扫描内容的上一行 中指定

确定计划中扫描过程中的并行度粒度

1	Id	 	Operation	I	Name	 	Rows	 	Bytes	1 (Cost	(%CPU)I	Time	1	Pstartl	Pstop	I TQ	IIN-OUTI	PQ Distrib	Ī
	1 2 3 4		SELECT STATEMENT PX COORDINATOR PX SEND QC (RANDOM) HASH GROUP BY PX RECEIVE PX SEND HASH HASH GROUP BY PX BLOCK ITERATOR		:TQ10001 :TQ10000	 	17 17 17 17 17 17 17 17		153 153 153 153 153 153 153 85M		565 565 565 565 565	(100) (100) (100) (100) (100)	00:00:07 00:00:07 00:00:07 00:00:07 00:00:07 00:00:07 00:00:01			16	01,01 01,01 01,00 01,00		QC (RAND) HASH	
*	8		TABLE ACCESS FULL	_	SALES	I	10M		85M	П	60	(97)1	00:00:01	Ι	1 I	16	Q1,00	I PCWP I		

Predicate Information (identified by operation id):

8 - filter("CUST_ID"<=22810 AND "CUST_ID">=22300)

l I	d I	Operation	l Name	I	Rows	I	Bytes	l Cost	(%	CPU) I	Time	Pstart	Pstop	TQ	IIN-OUTI	PQ Distrib
ļ	0	SELECT STATEMENT	 [ļ	17	ļ	153	! :	2	(50) إ	00:00:01				!!!	
l	뀙	PX COORDINATOR PX SEND QC (RANDOM)	 :TQ10001		17		153	I I ;	2	(50)	00:00:01	 		Q1,01		QC (RAND)
!	3 I	HASH GROUP BY		ļ	17	•	153			(50)1	00:00:01	l I		Q1,01	I PCWP I	
l¦	4 I 5 I	PX RECEIVE PX SEND HASH	 :TQ10000		26 26	-	234 234		1 1	: :	00:00:01 00:00:01			~-/	I PCWP I	HASH
İ	6 [PX PARTITION RANGE ALL	Ι΄.	İ	26		234	Ι :	1	3	00:00:01		16		I PCWC I	
 *	7 I 8 I	TABLE ACCESS BY LOCAL INDEX ROWID INDEX RANGE SCAN	I SALES I SALES_CUST		26 26		234	: 	0		00:00:01 00:00:01		16 16		I PCWP I	

Predicate Information (identified by operation id):

8 - access("CUST_ID">=22300 AND "CUST_ID"<=22810)

访问方法及其并行执行方式

访问方法	并行化方法
完整表扫描	块迭代器
按 ROWID 访问表	分区
索引唯一扫描	分区
索引范围扫描 (降序)	分区
索引跳过扫描	分区
完整索引扫描	分区
快速完整索引扫描	块迭代器
位图索引(以星型转换方式)	块迭代器

并行分发

- 当使用生产者与使用者组时是必需的
- 生产者必须将其数据传递或分发到使用者
- 将行传递到的运算符负责确定分发
- 分发可以在本地执行,也可以在 RAC 中的其他节点上执行
- 五种常见的重新分发

并行分发

散列

- 假定其中一个表是散列分区形式
- 对联接列的值应用散列函数
- 分发到基于相应散列分区而工作的使用者

广播

- 其中一个结果集的是小型结果集
- 向所有使用者发送数据副本

范围

- 通常用于并行排序操作
- 各并行服务器基于数据范围而工作
- QC 无需进行排序,即可以正确的顺序显示并行服务器结果

• 分区键分发 — PART (KEY)

- 假定目标表已进行分区
- 目标表的分区被映射到并行服务器
- 生产者基于分区列将扫描的每行映射到使用者

循环

• 随机但均匀地在使用者中分发数据

并行分发

Plan hash value: 2011745446

l Id	Operation	l Name	l Rows	Bytes	Cost	(%CPU)I	Time	Pstart	Pstop I	TQ	I IN-OU	l PQ Distrib
1 0	I SELECT STATEMENT	 	I 1407	I 54873 I	166	(83)1	00:00:02	I I	 I			
1	I PX COORDINATOR		1	I I				1 1	I		1	1
1 2	I PX SEND QC (ORDER)	:TQ10003	I 1407	I 54873 I	166	(83)1	00:00:02	1 1	I	Q1,03		I QC (ORDER)
1 3	I VIEW		I 1407	I 54873 I	166	(83) I	00:00:02	1 1	I	Q1,03	I PCWP	I
1 4	I SORT GROUP BY		I 1407	I 148KI	166	(83) I	00:00:02	1 1	I	Q1,03	I PCWP	1
1 5	I PX RECEIVE		I 1407	I 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP	1
I 6	I PX SEND RANGE	:TQ10002	I 1407	l 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,02	I P->P	I RANGE
1 7	I HASH GROUP BY	l	I 1407	I 148KI	166	(83)1	00:00:02	1 1	- 1	Q1,02	I PCWP	1
l* 8	I HASH JOIN	l	I 1407	I 148KI	165	(83)1	00:00:02	1 1	- 1	Q1,02	I PCWP	1
1 9	I PART JOIN FILTER CREATE	I :BF0000	I 1408	I 78848 I	7	(29)1	00:00:01	1 1	- 1	Q1,02		1
1 10	I PX RECEIVE	l .	I 1408	I 78848 I	7	(29)1	00:00:01	1 1	- 1	Q1,02		1
1 11	PX SEND PARTITION (KEY)	:TQ10001	I 1408	I 78848 I	7	(29)1	00:00:01	i i	ĺ	01,01		I PART (KEY)
I* 12	I HASH JOIN	1	I 1408	I 78848 I	7	(29)1	00:00:01	i i	ĺ	01,01	I PCWP	1
1 13	PX RECEIVE		I 139	I 3614 I	2		00:00:01	i i	ĺ	Q1,01		İ
1 14	PX SEND BROADCAST	:TQ10000	I 139	I 3614 I	2	(0)	00:00:01	i i	ĺ	01.00	I P->P	I BROADCAST
1 15	I PX BLOCK ITERATOR		l 139	I 3614 I	2		00:00:01	i i	i	Q1,00	I PCWC	
1 16	I TABLE ACCESS FULL	HOGAN_PCODE_HD_REF		I 3614 I	2		00:00:01		i	01,00	I PCWP	i
1 17	PX_BLOCK_ITERATOR	 	I 1408	I 42240 I	4		00:00:01		4 1	01,01	I PCWC	İ
18	TABLE ACCESS FULL	T_ACCT_MASTER_HD	1408	l 42240 i	4		00:00:01		64 I	01,01	I PCWP	
i 19		 	18919	. 960KI	157		00:00:02	i 14 i	14 i	01.02	I PCWC	-
i 20	I PX PARTITION HASH JOIN-FILTER		I 18919	. 960KI			00:00:02			01,02		1
i* 21			I 18919			(85) i		i 53 i	56 I	01.02		
l												

Predicate Information (identified by operation id):

8 - access("A","ACCT_NUM"="C","ACCT_NUM" AND "A","CO_ID"="C","CO_ID")
12 - access("A","PCODE"="B","PCODE")
21 - filter("C","ASOF_YYYYMM"=200102 AND "C","TRAN_AMT"<2000000000)

Note

- dynamic sampling used for this statement (level=2)

显示 PQ 服务器如何在彼此 之间分发行



阅读计划示例

示例 SQL 语句和方框图

MCO ID

```
SELECT '(' || pcode || ')' || pcode desc AS PRODUCT, CNT
FROM (SELECT a.pcode, b.pcode_desc, count(a.pcode) CNT
        FROM
              t acct master hd a
        hogan pcode hd ref b
        , t tran detail hd c
        WHERE a.pcode = b.pcode
        AND a.acct num=c.acct num
        AND a.co id=c.co id
        AND c.asof yyyymm=200102
        AND c.tran amt <2000000000
        GROUP BY a.pcode, b.pcode desc
        ORDER BY a.pcode, b.pcode desc)
                             HOGAN PCODE HD RE
                                       PCOD
                                                               数 TB 大小
                   ACCT NU
                                                            1 GB 大小
  TRAN DETAIL HD
                              T ACCT MASTER HD
```

执行计划示例 (续)

1.检查所返回的行数是否大致正确

Plan ha	sh value: 2011745446		_	╛								
l Id l	Operation	Name	l Rows	Bytes	Cost ((%CPU) I	Time	Pstart	Pstop I	TQ	IIN-OUTI	PQ Distrib
0	ozzzor omiznem	 	1407	54873	166	(83) إ	00:00:02	!			·	
1 2 3 4	PX COORDINATOR PX SEND QC (ORDER) VIEW SORT GROUP BY	:TQ10003	1407 1407 1407	I 54873 I	166 166 166	(83)1	00:00:02 00:00:02 00:00:02		 	Q1,03		QC (ORDER)
5 I I 6 I I 7 I	PX RECEIVE PX SEND RANGE HASH GROUP BY	:TQ10002	1407 1407 1407 1407	l 148KI I 148KI I 148KI	166 166 166	(83) I (83) I (83) I	00:00:02 00:00:02 00:00:02			Q1,03 Q1,02 Q1,02	I PCWP I I P->P I I PCWP I	RANGE
* 8 9 10	HASH JOIN PART JOIN FILTER CREATE PX RECEIVE	:BF0000	1407 1408 1408	I 78848 I I 78848 I	165 7 7	(29) I (29) I	00:00:02 00:00:01 00:00:01			Q1,02 Q1,02	I PCWP I I PCWP I	DART (VEV)
11 * 12 13 14	PX SEND PARTITION (KEY) HASH JOIN PX RECEIVE PX SEND BROADCAST	:TQ10001 - :TQ10000	1408 1408 139 139		7 2 2	(29) I (0) I	00:00:01 00:00:01 00:00:01 00:00:01			Q1,01 Q1,01	P->P PCWP PCWP	
14 15 16 17	PX BLOCK ITERATOR TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	133 139 133 1408	3614 3614	2 2 4	(0) I (0) I	00:00:01 00:00:01 00:00:01	1 1	4	Q1,00 Q1,00		вконысног
1 18 I I 19 I I 20 I	PX PARTITION RANGE SINGLE PX PARTITION HASH JOIN-FILTER TABLE ACCESS FULL	T_ACCT_MASTER_HU	1408 18919 18919		4 157 157	(25) I (85) I	00:00:01 00:00:02 00:00:02	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	64 I 14 I	Q1,01 Q1,02	I PCWP I I PCWC I I PCWC I	
I* 21 I	TABLE ACCESS FULL	I_IRAN_DETAIL_HD	T 18919		157		00:00:02	T 53 I	56 I		I PCWP I	
Predica	te Information (identified by operation	id):			++· \\\ \(\)	LASTET	= 					
12 -	access("A","ACCT_NUM"="C","ACCT_NUM" AND access("A","PCODE"="B","PCODE")			2	- 垄奴化	占算是召	江正明(
21 -	filter("C"."ASOF_YYYYMM"=200102 AND "C".	TKHN_HMT"<2000000000	')				3. 访问方法	上是否正确	1 ?	•		
Note - du	namic sampling used for this statement (level=2)			 意味 强烈	着没有 表明这	收集到统设 不是最佳设	十信息, 十划				
											عمرا	~ `

DRACLE

执行计划示例 (续)

Plan	hach	ualua+	2011745446
ır tanı	nasn	varue:	ZV11/4044D

Id Operation	l Name	l Rows	Bytes	Cost	(%CPU)I	Time	l Pstart	Pstop	TQ	IIN-OUT	PQ Distrib
0 SELECT STATEMENT 1 PX COORDINATOR 2 PX SEND QC (ORDER) 3 VIEW	 :TQ10003	1407 1407 1407		166 166 166	(83) I	00:00:02 00:00:02 00:00:02		 			I I I QC (ORDER) I
4 SORT GROUP BY 5 PX RECEIVE 6 PX SEND RANGE 7 HASH GROUP BY	 	1407 1407 1407 1407	l 148KI l 148KI l 148KI l 148KI	166 166 166 166	(83) I (83) I (83) I (83) I	00:00:02 00:00:02 00:00:02 00:00:02	 	 	Q1,03 Q1,03 Q1,02 Q1,02	PCWP PCWP P->P PCWP	 RANGE
	:BF0000 :TQ10001	1407 1408 1408 1408	1 78848 78848 78848	165 7 7 7	(29) I (29) I	00:00:02 00:00:01 00:00:01 00:00:01	 		Q1,02 Q1,02 Q1,02 Q1,01	PCWP PCWP PCWP P->P	I I I I I PART (KEY) I
* 12 HASH JOIN	 :TQ10000	1408 139 139 139	78848 3614 3614 3614	7 2 2 2	(0) I (0) I	00:00:01 00:00:01 00:00:01 00:00:01	 	 	Q1,01 Q1,01 Q1,00 Q1,00	PCWP PCWP P->P PCWC	I I I BROADCAST I
I 16 I TABLE ACCESS FULL I 17 I PX BLOCK ITERATOR I 18 I TABLE ACCESS FULL	HOGAN_PCODE_HD_REF 	139 1408 1408	3614 42240 42240	2 4 4	(0)1 (25)1 (25)1	00:00:01 00:00:01 00:00:01	l 1 l 1	 4 64	Q1,00 Q1,01 Q1,01	I PCWP I I PCWC I I PCWP I	
19 PX PARTITION RANGE SINGLE 20 PX PARTITION HASH JOIN-FILTE 1* 21 TABLE ACCESS FULL	 RI T_TRAN_DETAIL_HD	18919 18919 18919	I 960KI I 960KI I 960KI	157 157 157	(85)1	00:00:02 00:00:02 00:00:02	l:BF0000	l:BF00001	Q1,02 Q1,02 Q1,02	I PCWC I	
Predicate Information (identified by operation	id):						1		日不己	发生分区	` <i>l</i> /2
8 - access("A"."ACCT_NUM"="C"."ACCT_NUM" AN 12 - access("A"."PCODE"="B"."PCODE") 21 - filter("C"."ASOF_YYYYMM"=200102 AND "C"						5. 是否使月	用了正确的	4 剪 均联接方法	?	火 生 开 丛	· 112
Note		L,	6. 联接顺	序是否	正确?	是否首先该	方问可消除	余最多行的	勺表?		
- dynamic sampling used for this statement	(level=2)										

ORACLE

中骨文

执行计划示例 (续)

Plan hash value: 2011745446

l Id		Operation	Name	 I	Rows	l Bytes	I C	Cost ((%CPU) I	Time	Ι	Pstartl	Pstop	TQ	IN-OUTI	PQ Distrib
! 9) <u> </u>	SELECT STATEMENT		 !	1407	54873	! !	166	(83)	00:00:02	!	 !			!	
	LI	PX COORDINATOR	*T040007	!	1407	 E4077		400	/07\L	00+00+00				04 07	D /C	oc (oppen)
1 2	2 I	PX SEND QC (ORDER)	:TQ10003	-	1407 1407	l 54873 I 54873		166 166		00:00:02 00:00:02				Q1,03 Q1,03	PCWP I	QC (ORDER) I
1 4	1 1	SORT GROUP BY		i	1407	1481 1481		166	: :	00:00:02		i		Q1,03	PCWP I	
i 9	i i	PX RECEIVE		i	1407	. 148 I 148		166	1 1	00:00:02		i		Q1,03	PCWP I	
i ē	δi	PX SEND RANGE	:TQ10002	i	1407	148		166		00:00:02		i		Q1,02	P->P I	RANGE Ì
1 7	7	HASH GROUP BY		Ī	1407	l 148l	ΚL	166		00:00:02		İ		Q1,02	PCWP I	
	3	HASH JOIN		ı	1407	148		165		00:00:02		- 1		Q1,02	PCWP I	
1 9		PART JOIN FILTER CREATE	:BF0000	!	1408	78848		7		00:00:01		!		Q1,02	PCWP I	
1 10)	PX RECEIVE	. TO4 0004	!	1408	78848		_ (00:00:01		!		Q1,02	PCWP I	DADT (UEV)
11 11	LI	PX SEND PARTITION (KEY)	:TQ10001	!	1408	78848 70040	•	_ /		00:00:01				Q1,01		PART (KEY)
12 13		HASH JOIN PX RECEIVE		-	1408 139	l 78848 l 3614	-	- (00:00:01		- !		Q1,01 Q1,01	PCWP I	
14		PX SEND BROADCAST	:TQ10000		139	3614 3614		2	1	00:00:01	-			Q1,01		BROADCAST I
15		PX BLOCK ITERATOR	*1610000	i	139	3614 3614		5		00:00:01		i		Q1,00	PCWC i	DIVOLDOUS
i 18		TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	i	139	3614		2	: :	00:00:01		i		Q1,00	PCWP I	
i 17		PX BLOCK ITERATOR		Ì	1408	42240		4		00:00:01	-	1 İ	4	Q1,01	PCWC I	
1 18	3 1	TABLE ACCESS FULL I	T_ACCT_MASTER_HD	Ι	1408	1 42240	1	4		00:00:01		1 I	64	Q1,01	PCWP I	
1 19		PX PARTITION RANGE SINGLE			18919	960		157		00:00:02		14	14	Q1,02	PCWC I	
1 20		PX PARTITION HASH JOIN-FILTER			18919	I 960I		157		00:00:02					PCWC I	
l* 21	LI	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	I	18919	I 960I	ΚI	157	(85)1	00:00:02	ı	53 I	56	Q1,02	PCWP I	ا

Predicate Information (identified by operation id):

8 - access("A"."ACCT_NUM"="C"."ACCT_NUM" AND "A"."CO_ID"="C"."CO_ID")
12 - access("A"."PCODE"="B"."PCODE")
21 - filter("C"."ASOF_YYYYMM"=200102 AND "C"."TRAN_AMT"<2000000000)

Note

- dynamic sampling used for this statement (level=2)

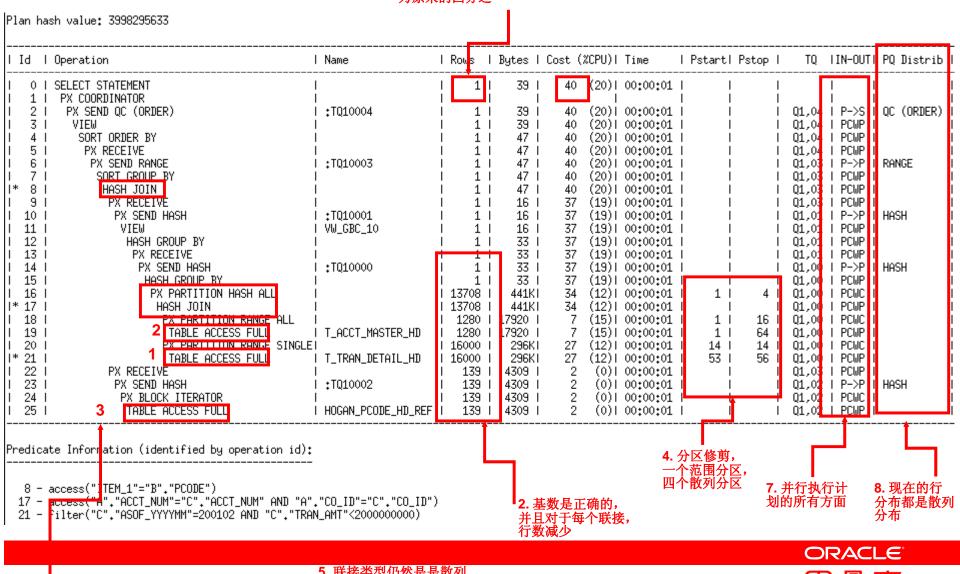
7. 检查是否并行执行计划的所有方面

8. 检查分发方法,确保不会广播大型表

ORACLE

执行计划示例(续) — 解决方案

1. 现在实际只返回了一行,开销降低 为原来的四分之一



6. 联接顺序已改变 — PWJ,将散列联 接到查找表 5. 联接类型仍然是是散列 联接,但现在是 PWJ

3. 访问方法保持不变

中骨文



查询

SELECT quantity_sold FROM sales s, customers c WHERE s.cust_id =c.cust_id;

您期望此语句的计划应是什么样的?

PLAN_TABLE_UUTPUT

Plan hash value: 2489314924

Id		I	Operation Nam	ie l	Rows		Bytes	Cost	(%CPU)I	Time		Pstart Pstop]
	 0 1 2	ĺ	SELECT STATEMENT PARTITION RANGE ALLI TABLE ACCESS FULL SAL	I I ES I	960 960 960	 	2880 2880 2880	 [[i (0) i	00:00:01 00:00:01 00:00:01	I	 1 16 1 16	

说明

联接到 customers 是多余的,因为没有选择列

存在"主键-外键"关系意味着我们可以删除表

问答

