ORACLE®



ORACLE®

Explain Plan 命令说明 Maria Colgan

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?

- 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?

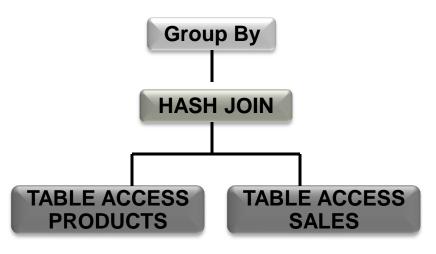
查询

```
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 Nar	
0 SELECT STATEMENT 1 HASH GROUP BY 2 HASH JOIN 3 TABLE ACCESS FULL PRO 4 PARTITION RANGE ALL 5 TABLE ACCESS FULL SALI	DUCTS ES

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

如何获取执行计划

```
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
 O SELECT STATEMENT
   HASH GROUP BY
 2 HASH JOIN
 3 TABLE ACCESS FULL PRODUCTS
 4 PARTITION RANGE ALL
      TABLE ACCESS FULL SALES
```

ORACLE

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'));
 Id Operation
                         Name
 O SELECT STATEMENT
   HASH GROUP BY
   HASH JOIN
   TABLE ACCESS FULL PRODUCTS
   PARTITION RANGE ALL
   TABLE ACCESS FULL SALES
```

一个优秀的优化器计划是什么样的?



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

基数或选择度

Plan ha	ash value: 2011745446											
Id	Operation	Name	I Rows I	Bytes	Cost	(%CPU) I	Time	Pstart	l Pstop I	TQ	I IN-OUT I	PQ Distrib
0 1 1 1 1 1 1 1 1 1	PX COORDINATOR PX SEND QC (ORDER) VIEW SORT GROUP BY PX RECEIVE PX SEND RANGE HASH GROUP BY HASH JOIN PART JOIN FILTER CREATE PX RECEIVE PX SEND PARTITION (KEY)	:TQ10003 :TQ10002 :BF0000 :TQ10001 :TQ10000 HOGAN_PCODE_HD_REF T_ACCT_MASTER_HD T_TRAN_DETAIL_HD	1407 1408 1408 1408 139 139 139 139 139 1408 1408	54873 54873 54873 148K 148K 148K 148K 148K 78848 78848 78848 78848 3614 3614 3614 42240 42240 42240 960K 960K	166 166 166 166 166 166 165 7 7 7 7 7 2 2 2 2 4 4 157 157	(83)1 (83)1 (83)1 (83)1 (83)1 (83)1 (29)1 (29)1 (29)1 (0)1 (0)1 (0)1 (25)1 (85)1 (85)1	00:00:01 00:00:02 00:00:02			01,03 01,03 01,02 01,02 01,02 01,02 01,01 01,01 01,01 01,00 01,00 01,00 01,01 01,02 01,02	PCWP PCWP PCWP PCWP PCWP PCWP PCWP PCWP	
												·

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)

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

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;

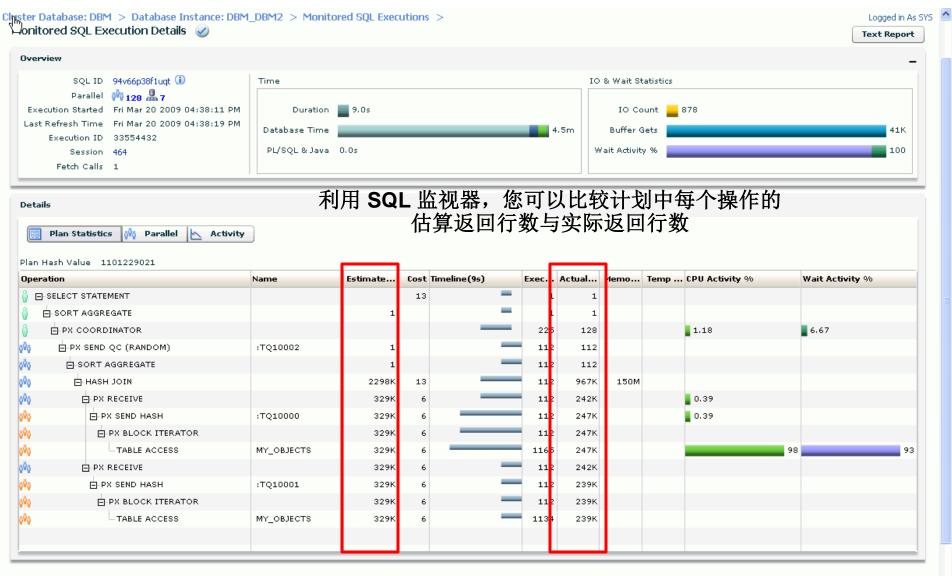
SELECT * FROM table (

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

Plan hash value: 479755685						_					
Id Operation	I Name I	Starts	I E-Rows	 	A-Rows I	A-Time	 I	Buffers	OMem I	1Mem Us	ed-Mem
0 SELECT STATEMENT		1 1 0 0 0 0 0 0 0 0	I 398 I 398 I 398 I 398 I 398 I 15289 I 766 I 766 I 766 I 15289 I 15289		398 14 0 14 0 14 0 14 0 14 0 14 766 14 0 14	0:00:01.42 0:00:01.42 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01 0:00:00.01		101 101 0 0 0 0 0 0 0 29 0	 717K	 717K 30	70K (0) S2K (0) 23K (0) S4 (0)

比较计划中每个操作的估算返回行数与实际返回行数

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

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	l Rows	Bytes I	Cost	(%CPU)I	Time	Pstart	l Pstop	I TQ	I IN-OUT	l PQ Distrib
0	SELECT STATEMENT		1407	54873	166	(83)	00:00:02	!	 	 [<u> </u>	I
1 1	PX_COORDINATOR		! !			/		1			!	
1 2 1	PX SEND QC (ORDER)	:TQ10003	1407	54873 I	166		00:00:02					I QC (ORDER)
1 3 1	VIEW		1407	54873	166		00:00:02	-		l Q1,03	–	l
1 4 1	SORT GROUP BY		l 1407 l	l 148KI			00:00:02	I		l Q1,03		l
I 5 I	PX RECEIVE I		l 1407 l	l 148KI	166		00:00:02	I		l Q1,03	I PCWP	
1 6 1	PX SEND RANGE	:TQ10002	l 1407 l	148K I	166	(83)1	00:00:02	I		l Q1,02	I P->P	I RANGE
1 7 1	HASH GROUP BY		l 1407 l	148K I	166	(83)1	00:00:02	I		l Q1,02	I PCWP	
I* 8 I	HASH JOIN		I 1407 I	148KI	165	(83)1	00:00:02	I		Q1,02	I PCWP	
1 9 1	PART JOIN FILTER CREATE I	:BF0000	I 1408 I	l 78848 l	7	(29)1	00:00:01	I		Q1,02	I PCWP	
I 10 I	PX_RECEIVE		I 1408 I	78848	7	(29)1	00:00:01	I		l Q1,02	I PCWP	
I 11 I	PX SEND PARTITION (KEY)	:TQ10001	I 1408 I	78848	7	(29)1	00:00:01	I		Q1,01	I P−>P	l Part (Key)
I* 12 I	HASH JOIN I		I 1408 I	78848	7	(29)1	00:00:01	I		Q1,01	I PCWP	
I 13 I	PX RECEIVE I		I 139 I	3614 I	2	(0)1	00:00:01	I		Q1,01	I PCWP	
I 14 I	PX SEND BROADCAST I	:TQ10000	I 139 I	3614 I	2	(0)1	00:00:01	I		Q1,00	I P->P	I BROADCAST
I 15 I	PX_BLOCK_ITERATORI		I 139 I	3614 I	2	(0)1	00:00:01	I		Q1,00	I PCWC	
I 16 I	TABLE ACCESS FULL	HOCAN POODE HD REF	130	3014	2	1777	00:00:01	+		1,00	I PCWP	
i 17 i	PV REHEK FIERRIER		1408	42240 I	4	(25)1	00:00:01	I 1	1 4	1 01,01	I PCWC	I
I 18 I	TABLE ACCESS FULL ←	T_ACCT_HASTER_HD	1408	42240	4	(00)	00.00.01	1 1	64	1,01	I PCWP	I
i 19 i	PX PARTITION RANGE SINGLE		18919	960KI	157	(85) i	00:00:02		14	1,02	I PCWC	
i 20 i	PX PARTITION HASH JOIN-FILTER		I 18919 I	960KT	157	(85) i	00:00:02					
i* 21 i	TABLE ACCESS FULL ←	T TRAN BETAIL HD	10010	960KI	457	(05)	00:00:02	53	56	01.02	I PCWP	l

Predicate Information (identified by operation id):

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...

^{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)

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');

I	d I	Operation	 	Name	1	Rows	1	Bytes I	Cost	(%CPU)I	Time	-
	0 1	SELECT STATEMENT INLIST ITERATOR	I		I	10000	I	136KI	2	2 (0)	00:00:01	
•	2 i 3 i	TABLE ACCESS BY INDEX INDEX UNIQUE SCAN		COUNTRIES COUNTRY_PK			İ	136K i	2 1		00:00:01 00:00:01	

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

Id Operation	l Name	I	Rows	1	Bytes	Cost	(%CPU)I	Time I
0 SELECT STATEMENT 1 TABLE ACCESS BY INDEX * 2 INDEX RANGE SCAN		Ì	10000	Ì	136KT	2	(0)	00:00:01 00:00:01 00:00:01

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

I	 J	Operation	1	Name	1	Rows	I	Bytes	l	Cost (%CPU)	Time	 	
		SELECT STATEMENT TABLE ACCESS FUL				1 1	 	14 14]	2 (0) 2 (0)	00:00:01 00:00:01		ORACLE

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

ORACLE

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

7 HASH GROUP RY	Id	Operation	Name	I Rows I	Bytes	l Cost	(%CPU)I	Time	Pstart	Pstop	TQ	I IN-OUT	l PQ Distrib	Ī
	0 1 1 1 1 1 1 1 1 1	SELECT STATEMENT PX COORDINATOR PX SEND QC (ORDER) VIEW SORT GROUP BY PX RECEIVE PX SEND RANGE HASH GROUP BY HASH JOIN PART JUIN FILTER CREATE PX RECEIVE PX SEND PARTITION (KEY) HASH JOIN PART JOIN PX RECEIVE PX SEND BROADCAST PX BLOCK ITERATOR TABLE ACCESS FULL	:TQ10003 :TQ10002 :BF0000 :TQ10001 :TQ10000	1407 1408 1408 1408 1408 139 139	54873 54873 54873 54873 148K 148K 148K 78848 78848 78848 78848 78848 3614 3614	166 166 166 166 166 166	(83) I (83) I (83) I (83) I (83) I (83) I (83) I (29) I (29) I (29) I (29) I (29) I (0) I (0) I	00:00:02 00:00:02 00:00:02 00:00:02 00:00:02 00:00:02 00:00:02 00:00:01 00:00:01 00:00:01 00:00:01 00:00:01 00:00:01 00:00:01		Pstop	Q1,03 Q1,03 Q1,03 Q1,03 Q1,02 Q1,02 Q1,02 Q1,02 Q1,01 W1,01 Q1,01 Q1,00 Q1,00		I QC (ORDER) - - RANGE - -	
17 PX BLOCK ITERATOR 1408 42240 4 (25) 00;00;01 1 4 1,01 PCWC 18 TABLE ACCESS FULL T_ACCT_MASTER_HD 1408 42240 4 (25) 00;00;01 1 64 1,01 PCWP 19 PX PARTITION RANGE SINGLE 18919 960K 157 (85) 00;00;02 14 14 1,02 PCWC 20 PX PARTITION HASH JOIN-FILTER 18919 960K 157 (85) 00;00;02 18F0000 1,02 PCWC *21 TABLE ACCESS FULL T_TRAN_DETAIL_HD 18919 960K 157 (85) 00;00;02 53 56 1,02 PCWP	I 19 I I 20 I	TABLE ACCESS FULL PX PARTITION RANGE SINGLE PX PARTITION HASH JOIN-FILTER		1408 18919 18919	42240 960K 960K	157	(25) I (85) I (85) I	00:00:01 00:00:02 00:00:02	1 14 :BF0000	14 BF0000	01,01 01,02 01,02	I PCWP I I PCWC I I PCWC I	 	

Predicate Information (identified by operation id):

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

^{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)

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

1	Id	I	Operation	ı	Name	I	Rows	I	Bytes	I	Cost (%C	PU)	Time	l
I	0	I	SELECT STATEMENT			ı	19	I	722	I	3	(0)	00:00:01	ı
	1	1	NESTED LOOPS	1		-								-
	2	1	NESTED LOOPS	- 1		1	19	1	722		3	(0)	00:00:01	-
*	3	1	TABLE ACCESS FULL	1	DEPARTMENTS	Ι	2	1	32	I	2	(0)	00:00:01	-
*	4	Ι	INDEX RANGE SCAN	1	EMP DEPARTMENT IX	Ι	10	1		I	0	(0)	00:00:01	-
-	5	I	TABLE ACCESS BY INI	EX ROWID	EMPLOYEES	I	10	I	220	I	1	(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

I	d	Operation	on	I	Name	I	Rows	١	Bytes	I	Cost	(%CPU)
	0 1 2	SELECT : HASH JO TABLE	OIN	 	ORDERS	 	665	İ	13300 13300 840	İ	8	(25) (25) (25)
i	3			•	ORDER_ITEMS	i	665	•		•		(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

l Id	Operation	I Name	I	Rows	I	Bytes	Cost ((%CPU)I	Time	
0 1 2 3	BUFFER SORT] 	11235 11235 105 107 107		120KI 120KI 420 I 749 I 749 I	33 33 1 32 0	(7)1 (0)1 (7)1	00:00:01 00:00:01 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	 d	 	Operation	Name	1	Rows	1	Bytes	 	Cost ((%CPU)	Time		1
 * 	0 1 2 3 4 5	•	SELECT STATEMENT SORT ORDER BY VIEW HASH JOIN FULL OUTER INDEX FAST FULL SCAN TABLE ACCESS FULL	VW_FOJ_O DEPT_ID_PK EMPLOYEES	 	122 122 122 122 122 27 107	 	4758 4758 4758 4758 1342 108 749	 			00:0 00:0	0:01 0:01 0:01 0:01	

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

l Id l	Operation	Name	l Rows	l Bytes I	Cost	(%CPU)I	Time	Pstart	Pstop	I TQ	IIN-OUTI	PQ Distrib
1 0 1	SELECT STATEMENT		I 1407	I 54873 I	166	(83)1	00:00:02	l I		<u> </u>		Į.
1	PX COORDINATOR I		1	l I				l I	١			
1 2 1	PX SEND QC (ORDER)	:TQ10003	I 1407	l 54873 l	166	(83)1	00:00:02	I I	١	Q1,03	P->S	QC (ORDER) I
1 3 1	VIEW I		I 1407	l 54873 l	166	(83)1	00:00:02	l I	I	Q1,03	I PCWP I	1
1 4 1	SORT GROUP BY		I 1407	l 148KI	166	(83) I	00:00:02	I I	I	Q1,03	I PCWP I	
1 5 1	PX RECEIVE I		I 1407	l 148KI	166	(83) I	00:00:02	I I	I	Q1,03	I PCWP I	
1 6 1	PX SEND RANGE I	:TQ10002	I 1407	l 148KI	166	(83) I	00:00:02	I I	I	Q1,02	P->P	range i
1 7 1	HASH GROUP BY		I 1407	l 148KI	166	(83)1	00:00:02	I I	I	Q1,02	I PCWP I	1
I* 8 I	HASH JOIN		I 1407	l 148KI	165	(83)1	00:00:02	I I	- 1	Q1,02	I PCWP I	1
1 9 1	PART JOIN FILTER CREATE	:BF0000	I 1408	l 78848 l	7	(29)1	00:00:01	I I	- 1	Q1,02	I PCWP I	1
I 10 I	PX RECEIVE I		I 1408	l 78848 l	7	(29)1	00:00:01	l I		Q1,02	I PCWP I	1
I 11 I	PX SEND PARTITION (KEY)	:TQ10001	I 1408	l 78848 l	7	(29)1	00:00:01	I I		Q1,01	1 P->P 1	PART (KEY) I
I* 12 I	HASH JOIN I		I 1408	l 78848 l	7	(29)1	00:00:01	I I		Q1,01	I PCWP I	1
I 13 I	PX RECEIVE I		I 139	3614	2	(0)1	00:00:01	I I	- 1	Q1,01	I PCWP I	1
I 14 I	PX SEND BROADCAST I	:TQ10000	I 139	3614	2	(0)1	00:00:01	I I	- 1	Q1,00	1 P->P 1	BROADCAST I
l 15 l	PX BLOCK ITERATOR I		I 139	3614	2			I I	- 1	Q1,00	I PCWC I	1
I 16 I	1 TABLE ACCESS FULL I	HOGAN_PCODE_HD_REF	I ←139	3014	- 2	(0)+	00:00:01	-		Q1,00	<mark>+-</mark> PCWP ∣	1
I 17 I	PX BLOCK ITERATOR I		I 1408	I 42240 I	4	(25)1	00:00:01	l 1 l	4 1	Q1,01	T POWC 1	1
I 18 I	² TABLE ACCESS FULL I	T_ACCT_MASTER_HD	I 1408	I 42240 I	4	(25)1	00:00:01	l 1 l	64 I	Q1,01	I POWP I	1
I 19 I	PX PARTITION RANGE SINGLE		I 18919	960KT	157	(85)1	00:00:02	l 14 l	14 I	Q1,02	I POWC I	1
1 20 1	PX PARTITION HASH JOIN-FILTERI		I 18919	960KT	157	(85)1	00:00:02	1:BF00001	:BF00001	Q1,02	I POWC I	1
I* 21 I	3 TABLE ACCESS FULL I	T_TRAN_DETAIL_HD	l 18919	960KT	157	(85)1	00:00:02	l 53 I	56 I	Q1,02	I PCWP I	I

Predicate Information (identified by operation id):

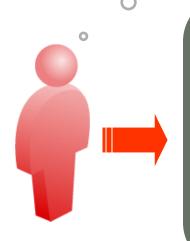
需要从可最大程度减少结果集行数 的表开始操作

如果联接顺序不正确, 请检查统计信息、基数及访问方法

^{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)

分区修剪

问: 2008 年 5 月 20 22 日 这几天的总销售额是多少?



Select sum(sales_amount)

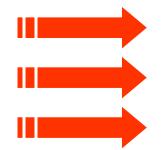
From SALES

Where sales_date between

to_date('05/20/2008','MM/DD/YYYY')

and

to_date('05/23/2008','MM/DD/YYYY');



仅访问 3 个相 关的分区 Sales 表

2008年5月18日

2008年5月19日

2008年5月20日

2008年5月21日

2008年5月22日

2008年5月23日

2008年5月24日

Plan hash value: 2011745446

l Id l	Operation	Name	l Rows	Bytes Co:	st (%CPU)	Time I	Pstart Pstop	TQ IN-OUT PQ Distrib
0	SELECT STATEMENT PX COORDINATOR		1407	54873 :	166 (83)	00:00:02		
1 2 1	PX SEND QC (ORDER)	:TQ10003	1407			00:00:02	į į	Q1,03 P->S QC (ORDER)
3 4	VIEW SORT GROUP BY		1407 1407			00:00:02 00:00:02		Q1,03 PCWP
5 1	PX RECEIVE	. 7040000	I 1407	148KI :	166 (83)I	00:00:02 I	į į	Q1,03 PCWP
6 7	PX SEND RANGE HASH GROUP BY	:TQ10002	1407 1407			00:00:02 00:00:02		Q1,02 P->P RANGE Q1,02 PCWP
I* 8 	HASH JOIN PART JOIN FILTER CREATE	:BF0000	1407 1408	l 148KI : I 78848 I		00:00:02 00:00:01		Q1,02 PCWP
i 10 i	PX RECEIVE	•	I 1408	78848 I	7 (29)1	00:00:01	i i	Q1,02 PCWP
11 * 12	PX SEND PARTITION (KEY) HASH JOIN	:TQ10001	1408 1408	78848 78848		00:00:01 00:00:01		Q1,01 P->P PART (KEY) Q1,01 PCWP
13 14	PX RECEIVE PX SEND BROADCAST	*T010000	I 139 I 139	3614 I 3614 I	2 (0)1	00:00:01 I	į	Q1,01 PCWP
l 15 i	PX BLOCK ITERATOR	:TQ10000	139	3614 I	2 (0)1	00:00:01 00:00:01	iii	Q1,00 PCWC
16 17	TABLE ACCESS FULL PX BLOCK ITERATOR	HOGAN_PCODE_HD_REF	139 1408	3614 42240		00:00:01 00:00:01	1 1 4	Q1,00 PCWP Q1,01 PCWC
18 1	TABLE ACCESS FULL	T_ACCT_MASTER_HD	1408	42240 I	4 (25)1	00:00:01 I	1 64	Q1,01 PCWP
19 20	PX PARTITION RANGE SINGLE PX PARTITION HASH JOIN-FILTER		18919 18919	960KT :	157 (85) I	00:00:02 00:00:02	14 14 BF0000 :BF0000	Q1,02 PCWC Q1,02 PCWC
I* 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	18919 	960KT :	157 (85)I 	00:00:02 I	53 l 56 l	Q1,02 PCWP

Predicate Information (identified by operation id):

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

^{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)

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	Name	l Rows	Bytes	Cost	(%CPU)I	Time	Pstart	Pstop I	TQ	I IN-OUT I	PQ Distrib
1 0 1	SELECT STATEMENT		I 1407	I 54873 I	166	(83)1	00:00:02	1 1	ı		l l	
1	PX COORDINATOR		I	I I		I		1 1	I		l l	
1 2 1	I PX SEND QC (ORDER) I	:TQ10003	I 1407	I 54873 I	166	(83) I	00:00:02	1 1	I	Q1,03	P->S	QC (ORDER)
1 3 1	VIEW		I 1407	I 54873 I	166	(83) I	00:00:02	1 1		Q1,03	I PCWP I	
1 4 1	SORT GROUP BY		I 1407	I 148KI	166	(83) I	00:00:02	1 1		Q1,03	I PCWP I	
1 5 1	PX RECEIVE		I 1407	I 148KI	166	(83)1	00:00:02	1 1		Q1,03	I PCWP I	
1 6 1	PX SEND RANGE	:TQ10002	I 1407	I 148KI	166	(83)1	00:00:02	1 1	I	Q1,02	P->P	RANGE I
1 7 1	HASH GROUP BY		I 1407	I 148KI	166	(83)1	00:00:02	1 1		Q1,02	I PCWP I	
I* 8 I	HASH JOIN		I 1407	I 148KI	165	i (83) I	00:00:02	1 1		Q1,02	I PCWP I	
1 9 1	PART JOIN FILTER CREATE	:BF0000	I 1408	I 78848 I	7	(29)1	00:00:01	i i	ĺ	01,02	I PCWP I	
I 10 I	PX RECEIVE	,	I 1408	I 78848 I	7	(29)1		i i	ĺ	01,02	I PCWP I	
l 11 l	PX SEND PARTITION (KEY)	:TQ10001	I 1408	I 78848 I	7	(29)1		i i	i	01,01	P->P	PART (KEY)
I* 12 I	HASH JOIN	* - 4	I 1408	I 78848 I	7	(29)1	00:00:01	i i	i	01,01	I PCWP I	,,,_,,
i 13 i	PX RECEIVE		I 139	I 3614 I	2	(0)		i i	i	Q1,01	I PCWP I	
i 14 i	PX SEND BROADCAST	:T010000	i 139	I 3614 I	2	(0)			i	Q1,00	P->P	BROADCAST
i 15 i	PX BLOCK ITERATOR	•	i 139	i 3614 i		(ŏ) i			i	Q1,00	i PCWC i	211211221121
i 16 i	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	i 139	. 3614 i	- 5	i (ŏ) i			i	Q1,00	PCWP I	į
i 17 i	PX BLOCK ITERATOR		i 1408	i 42240 i	4	(25) i			4 i	Q1,01	i PCWC i	
i 18 i	TABLE ACCESS FULL	T_ACCT_MASTER_HD	i 1408	i 42240 i	4				64 i	01,01	PCWP I	
i 19 i	PX PARTITION RANGE SINGLE		1 18919	. 960KI	157				14	Q1,02	PCWC i	i
i 20 i	PX PARTITION HASH JOIN-FILTER		I 18919	I 960KI						01.02	PCWC I	
i* 21 i		T_TRAN_DETAIL_HD	1 18919						56 I	Q1,02	PCWP I	

Predicate Information (identified by operation id):

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

^{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"<20000000000)

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

Id Operation																							
1 PX COORDINATOR	I	Id	I	Operation	I	Name	I	Rows]	Bytes	l Co	st	(%CPU)I	Time	I	Pstartl	Pstop	I	TQ	IIN-OUTI	PQ	Distrib	I
	١.	1 2 3 4 5 6 7	 	PX COORDINATOR PX SEND QC (RANDOM) HASH GROUP BY PX RECEIVE PX SEND HASH HASH GROUP BY PX BLOCK ITERATOR		:TQ10000	 	17 17 17 17 17 17	 	153 153 153 153 153 153 85M	 	565 565 565 565 565 60	(100) (100) (100) (100) (100) (100) (97)	00:00:07 00:00:07 00:00:07 00:00:07 00:00:07 00:00:01	 			 	Q1,01 Q1,01 Q1,00 Q1,00 Q1,00	PCWP PCWP P->P PCWP PCWC			

Predicate Information (identified by operation id):

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

l													
l I	d I	Operation	l Name	I R	Rows ∣	Bytes	l Cost	(%CPU)I	Time	Pstart Pstop	I TO) IIN-OUTI	PQ Distrib
 	0 17 2 3 4 1 5 1 6 1 7	SELECT STATEMENT PX COORDINATOR PX SEND QC (RANDOM) HASH GROUP BY PX RECEIVE PX SEND HASH PX PARTITION RANGE ALL TABLE ACCESS BY LOCAL INDEX ROWID	 :TQ10001 :TQ10000 :TQ10000 :TQ10000	 	17 17 17 17 26 26 26 26	153 153 153 153 234 234 234 234	2 2 1 1 1	(50)1 (50)1 (50)1 (0)1 (0)1	00:00:01 00:00:01 00:00:01 00:00:01 00:00:01 00:00:01 00:00:01		Q1,0 Q1,0)1 PCWP)1 PCWP)0 P->P)0 PCWC	QC (RAND)
1*	8 I	INDEX RANGE SCAN	I SALES_CUST	Ι	26 I		1 0	(0)1	00:00:01	I 1 I 16	Q1,0	O I PCWP I	1

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	l Bytes l	Cost	(%CPU)I	Time	Pstart	Pstop	TQ	I IN-OU	l PQ Distrib l
1 0	I SELECT STATEMENT	 	I 1407	I 54873 I	166	(83)1	00:00:02	I I	ļ		ļ	l l
1	I PX COORDINATOR		I	1 1				1 1	I		1	1
1 2	I PX SEND QC (ORDER)	:TQ10003	I 1407	I 54873 I	166	(83) I	00:00:02	1 1	- 1	Q1,03	I P−>S	I QC (ORDER) I
1 3	I VIEW	l	I 1407	I 54873 I	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP	1
1 4	I SORT GROUP BY	l	I 1407	I 148KT	166	(83)1	00:00:02	1 1	- 1	Q1,03	I PCWP	1
1 5	I PX RECEIVE	1	I 1407	I 148KT	166	(83)1	00:00:02	1 1	1	Q1,03	I PCWP	1 1
1 6	I PX SEND RANGE	I :TQ10002	I 1407	I 148KI	166	(83)1	00:00:02	1 1	1	01.02		I RANGE I
1 7	I HASH GROUP BY		1 1407	I 148KI	166	(83)1	00:00:02	i i	ĺ	Q1,02	I PCWP	1
i* 8		İ	1 1407	I 148KI	165	(83)		i i	i	01,02		i i
i š	PART JOIN FILTER CREATE	:BF0000	I 1408	I 78848 I	7	(29) i	00:00:01	i i	i	Q1,02		i i
i 10	PX RECEIVE	l *2. ****	i 1408	. 78848 I	ż	(29) i	00:00:01	i i	i	Q1,02		i i
i 11	PX SEND PARTITION (KEY)	: :TQ10001	I 1408	. 78848 I	ż	(29) i	00:00:01	-	i	Q1,01		i part (key) i
i* 12	HASH JOIN	l	I 1408	. 78848 I	ż	(29) i			i	Q1,01	i PCWP	
13	PX RECEIVE	i I	i 139	3614 I	,				i	Q1,01	I PCWP	i i
1 14	PX SEND BROADCAST	:TQ10000	i 139	. 3614 I	5		00:00:01	i i	i	Q1,00	i P->P	i BROADCAST i
i 15	PX BLOCK ITERATOR	+1@±~~~~	i 139	. 3614 I	5	1 1		i i	i	01,00	i PCWC	I DINORIDORIO I
16	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	i 139	3614	5	(ŏ)i			i	01.00	I PCWP	i i
1 17	PX BLOCK ITERATOR		1 1408	3014 42240	4	(25)			a i	01,00	I PCWC	i
18	TABLE ACCESS FULL	' T_ACCT_MASTER_HD	1 1408	42240 42240	4	(25)1	00:00:01		64 I		I PCWP	i i
1 19	PX PARTITION RANGE SINGLE	 HCC	I 18919	1 42240 1 I 960KI	157	(85)1	00:00:01		14 I	Q1,01	I PCWC	
		l 				(85)1				Q1,02		!
1 20 1* 21	PX PARTITION HASH JOIN-FILTER		I 18919	I 960KI			::•::•:=			Q1,02	I PCWC	!
1" 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	l 18919	I 960KT	157	(85)1	00:00:02	I 53 I	56 I	Q1,02	I PCWP	1

Predicate Information (identified by operation id):

Note

显示 PQ 服务器如何在彼此 之间分发行

^{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)

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

阅读计划示例

示例 SQL 语句和方框图

```
SELECT '(' | pcode | | ')' | pcode desc AS PRODUCT, CNT
FROM (SELECT a.pcode, b.pcode desc, count(a.pcode) CNT
       FROM
                t acct master hd
                                    а
                , 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 <200000000
       GROUP BY a.pcode, b.pcode desc
       ORDER BY a.pcode , b.pcode desc )
                           HOGAN PCODE HD REF
                                                      数 TB 大小
                                   PCOD
                 ACCT NUM
                                                      1 GB 大小
                           T_ACCT_MASTER_HD
T_TRAN_DETAIL_HD
                 CO ID
```

执行计划示例 (续)

1.检查所返回的行数是否大致正确

Plan ha	ash value: 2011745446											
l Id l	Operation	Name	l Rows	Bytes	l Cost	(XCPU) I	Time	Pstart	Pstop	TQ	I IN-OUT	l PQ Distrib
0 1 2 3 4 5 6 7 8 10 11 13 14 15 16 17 18 19	PX SEND PARTITION (KEY) HASH JOIN PX RECEIVE PX SEND BROADCAST PX BLOCK ITERATOR TABLE ACCESS FULL PX PLOCK ITERATOR TABLE ACCESS FULL PX PLOCK ITERATOR TABLE ACCESS FULL PX PLOCK ITERATOR PX PARTITION RANGE SINGLE	:TQ10003 :TQ10002 :BF0000 :TQ10001 :TQ10000 H06AN_PCOBE_HB_REF	1407 1407 1407 1407 1407 1408 1408 1408 1408 133 133 133 1408 140	7 54873 7 54873 7 54873 7 148K 7 148K 7 148K 7 148K 8 78848 8 78848 8 78848 8 78848 9 3614 9 3614 9 3614 9 3614 9 3614 9 3614 9 3614	166 166 166 165 7 7 7 7 7 2 2 2 4	(83)1 (83)1 (83)1 (83)1 (83)1 (83)1 (83)1 (29)1 (29)1 (29)1 (0)1 (0)1 (0)1 (0)1 (25)1 (85)1	00;00;02 00;00;02 00;00;02 00;00;02 00;00;02 00;00;02 00;00;02 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01 00;00;01	l 14 l	64 I 14 I	Q1,03 Q1,03 Q1,02 Q1,02 Q1,02 Q1,02 Q1,01 Q1,01 Q1,01 Q1,00 Q1,00 Q1,01 Q1,01 Q1,01 Q1,01 Q1,02	PCWP PCWP PCWP PCWP PCWP PCWP PCWP PCWP PCWP PCWP PCWC PCWC PCWC	I I I I I PART (KEY) I I I
20 * 21	PX PARTITION HASH JOIN-FILTER	T_TRAN_DETAIL_HD	18919 18919		157 157		00;00;02 00;00;02	1:BF00001 53			I PCWC I I PCWP I	
8 - 12 - 21 - Note 	ate Information (identified by operation in access("A"."ACCT_NUM"="C"."ACCT_NUM" AND access("A"."PCODE"="B"."PCODE") filter("C"."ASOF_YYYYMM"=200102 AND "C"."	 "A"."CO_ID"="C"."CO_ TRAN_AMT"<2000000000	7	- 基数化 意味 强烈	着没有	京正确? 3. 访问方法 收集到统计 不是最佳计	·信息,	育?				

执行计划示例 (续)

Plan hash value: 2011745446

l Id l	Operation	Name I	Rows	l Bytes l	Cost	(%CPU)I	Time	Pstart	Pstop	TQ	IIN-OUTI	PQ Distrib
0 1 2 3 4	SELECT STATEMENT I PX COORDINATOR I PX SEND QC (ORDER) I VIEW I SORT GROUP BY	:TQ10003	1407 1407 1407 1407	l I 54873 54873 148K		1 (83)1 (83)1 (83)1	00:00:02 00:00:02 00:00:02 00:00:02		 	Q1,03 Q1,03 Q1,03	I PCWP I	QC (ORDER) I
5 6 7 * 8	PX RECEIVE PX SEND RANGE POSH CROUP BY PASH JOIN	;TQ10002	1407 1407 1407	148K 148K 148K	166 166	(83) I (83) I	00:00:02 00:00:02 00:00:02	 	 	Q1,03 Q1,02 Q1,02 Q1,02		RANGE I
9 10 11	PART JOIN FILTER CREATE PX RECEIVE PX SENU PHR ITION (KEY)	:BF0000 : TQ10001	1408 1408 1408		7 7 7 7	(29) I (29) I	00:00:01 00:00:01 00:00:01	1 1	 	Q1,02 Q1,02 Q1,01	I PCWP I I PCWP I I P->P I	i I PART (KEY) I
* 12 13 14 15	HASH JOIN PX RECEIVE PX SEND BROADCAST PX BLOCK ITERATOR	:TQ10000	1408 139 139 139	78848 3614 3614 3614	7 2 2 2	(0) (0)	00:00:01 00:00:01 00:00:01 00:00:01		 	Q1,01 Q1,01 Q1,00 Q1,00	1 P->P 1	I BROADCAST I
16 17 18 19	1 TABLE ACCESS FULL I PX BLOCK ITERATOR I TABLE ACCESS FULL I PX PARTITION RANGE SINGLE I	HOGAN_PCODE_HD_REF		3614 42240 42240 960K	2 4 4	(0) I (25) I (25) I	00:00:01 00:00:01 00:00:01 00:00:02	1 1	4 64 14	Q1,00 Q1,01 Q1,01 Q1,02	I PCWP I I PCWC I I PCWP I	i !
i 20 i I* 21 i	🤦 PX PARTITION HASH JOIN-FILTERI	T_TRAN_DETAIL_HD I	18919 18919	I 960KI I 960KI	157	(85)1	00:00:02 00:00:02	I:BF00001		Q1,02		į Į
8 - 12 -	ate Information (identified by operation i access("A"."ACCT_NUM"="C"."ACCT_NUM" AND access("A"."PCODE"="B"."PCODE") filter("C"."ASOF_YYYYMM"=200102 AND "C"."	 "A"."CO_ID"="C"."CO_I	-				5. 是否使用	↑ 月了正确的	4. 剪	?	发生分区	修
Note 			L,	6. 联接顺	序是否	正确?	是否首先说	方问可消险	录多行的	ۇ 表?		
- dy	namic sampling used for this statement (l	evel=2)										

ORACLE

执行计划示例 (续)

Plan hash value: 2011745446

l Id	 	Operation	l Name	ı	Rows	Byte	s I	Cost	(%CPU)I	Time	l Psta	rt	Pstop	TQ	IN-OUT I	PQ Distrib
1	0	SELECT STATEMENT	 [I	1407	I 5487	'3 I	166	(83)1	00:00:02	I	ı	ļ		ļ	
ı	1 I	PX COORDINATOR	l	-		I	- 1						I		I	
ı	2-1	PX SEND QC (ORDER)	l :TQ10003	-	1407	I 5487	′3 I	166	(83) I	00:00:02				Q1,03	P->S I	QC (ORDER) I
ı	3 I	VIEW	l	-	1407	I 5487	′3 T	166	(83) I	00:00:02				Q1,03	PCWP I	١
1	4	SORT GROUP BY	l	- 1	1407	I 14	18K I	166	(83)1	00:00:02				Q1,03	PCWP I	١
1	5 I	PX RECEIVE	I		1407	I 14	18K I	166	(83)1	00:00:02				Q1,03	PCWP I	I
1	6 I	PX SEND RANGE	I :TQ10002	- 1	1407	I 14	18K I	166	(83)1	00:00:02				Q1,02	P->P I	range i
1	7 I	HASH GROUP BY			1407	I 14	18K I	166		00:00:02				Q1,02	PCWP I	I
 *	8 I	HASH JOIN	I	- 1	1407	I 14	18K I	165	(83)1	00:00:02				Q1,02	PCWP I	I
1	9 I	PART JOIN FILTER CREATE	l :BF0000	- 1	1408	I 7884	₽8 I	7	(29)1	00:00:01				Q1,02	PCWP I	I
1 1	0 1	PX RECEIVE	l	- 1	1408	I 7884	₽8 I	7	(29)1	00:00:01				Q1,02	PCWP I	I
1 1	1 I	PX SEND PARTITION (KEY)	:TQ10001		1408	I 7884	₽8 I	7	(29)1	00:00:01		-		Q1,01	P->P	PART (KEY) I
l* 1	2 1	HASH JOIN	l	- 1	1408	I 7884	₽8 I	7	(29)1	00:00:01				Q1,01	PCWP I	I
1 1	3 I	PX RECEIVE			139	I 361	.4 1	2	(0)	00:00:01		-		Q1,01	PCWP I	I
1 1	4	PX SEND BROADCAST	I :TQ10000	- 1	139	I 361	.4	2	(0)	00:00:01				Q1,00	P->P I	BROADCAST I
1 1	5 I	PX BLOCK ITERATOR	l	- 1	139	I 361	.4 1	2	(0)	00:00:01				Q1,00	PCWC I	I
1 1	6 I	TABLE ACCESS FULL	I HOGAN_PCODE_HD_REF		139	I 361	.4	2	(0)	00:00:01				Q1,00	PCWP I	١
1 1	7 I	PX BLOCK ITERATOR	l	- 1	1408	1 4224	1 O	4	(25)1	00:00:01		1 I	4 I	Q1,01	PCWC I	١
1 1	8 I	TABLE ACCESS FULL	T_ACCT_MASTER_HD		1408	1 4224	1 O	4	(25)1	00:00:01		1 I	64 I	Q1,01	PCWP I	ا
1 1	9 I	PX PARTITION RANGE SINGLE			18919	1 96	SOK L	157	(85)1	00:00:02	1 1	.4 1	14 I	Q1,02	PCWC I	ا
1 2	0 1	PX PARTITION HASH JOIN-FILTER			18919	1 96	SOK L	157	(85)1	00:00:02	1:BF00	0001:0	BF00001	Q1,02	PCWC I	ا
1* 2	1 I	TABLE ACCESS FULL	I T_TRAN_DETAIL_HD	١	18919	1 96	SOKT	157	(85)1	00:00:02	1 5	i3 T	56 I	Q1,02	PCWP I	١

Predicate Information (identified by operation id):

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

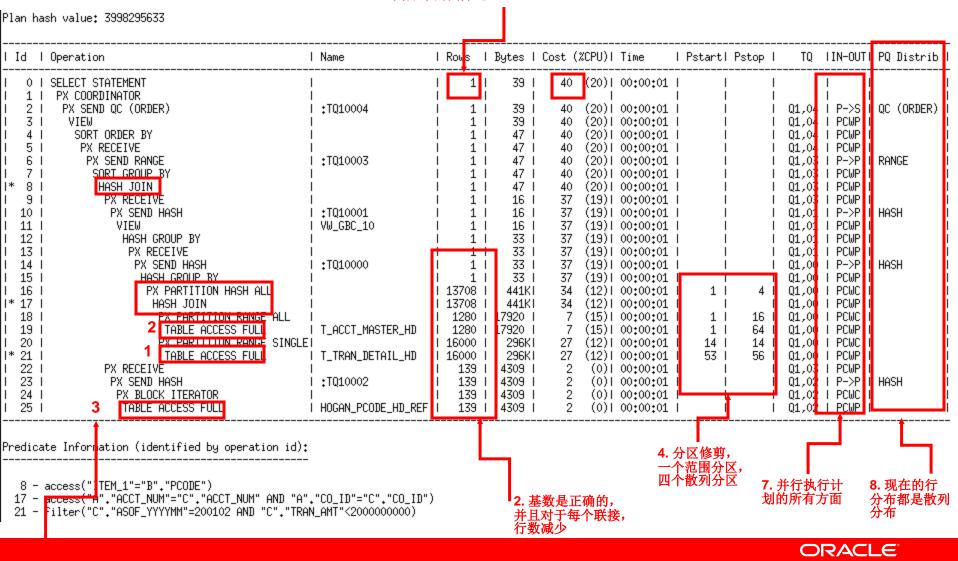
8. 检查分发方法,确保不会广播大型表



^{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)

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



确定是否获得了合适的计划

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?

PLA	ΆΝ_΄ 	TA:	BLE_OUTPUT 														
Pla	an I	ha:	sh value: 2489314924	•													
 	 Id	- <u>-</u> -	Operation	 	 Name	- <u>-</u> -	Rows	 	Bytes	 I	Cost (%	 (CPU) I	Time	 	 Pstart	Pstop	
	1	 	SELECT STATEMENT PARTITION RANGE AL TABLE ACCESS FULL	Lİ.		I	960 960 960	İ	2880 2880 2880 2880	Ī	5 5 5	-(0)1	00:00:01 00:00:01 00:00:01		1 I	16 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

