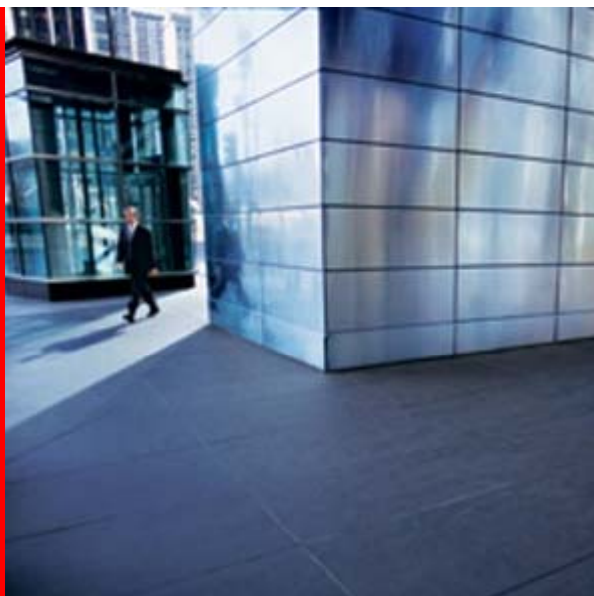


**ORACLE®**



**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?

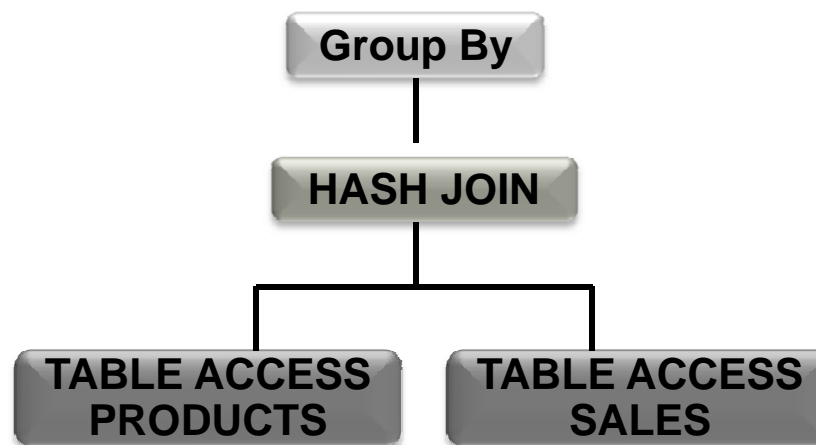
## 查询

```
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
0	SELECT STATEMENT	
1	HASH GROUP BY	
2	HASH JOIN	
3	TABLE ACCESS FULL	PRODUCTS
4	PARTITION RANGE ALL	
5	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



# 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
-----
0 SELECT STATEMENT
1 HASH GROUP BY
2 HASH JOIN
3 TABLE ACCESS FULL PRODUCTS
4 PARTITION RANGE ALL
5  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'));
```

```
-----
Id Operation                                Name
-----
0 SELECT STATEMENT
1  HASH GROUP BY
2  HASH JOIN
3  TABLE ACCESS FULL PRODUCTS
4  PARTITION RANGE ALL
5  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%';
```

注: 有关详情, 请访问 [www.optimizermagic.blogspot.com](http://www.optimizermagic.blogspot.com)



# 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？





# 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 hash value: 2011745446

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		1407	54873	166 (83)	00:00:02					
1	PX COORDINATOR		1407	54873	166 (83)	00:00:02					
2	PX SEND QC (ORDER)	:TQ10003	1407	54873	166 (83)	00:00:02			Q1,03	P->S	QC (ORDER)
3	VIEW		1407	54873	166 (83)	00:00:02			Q1,03	PCWP	
4	SORT GROUP BY		1407	148K	166 (83)	00:00:02			Q1,03	PCWP	
5	PX RECEIVE		1407	148K	166 (83)	00:00:02			Q1,03	PCWP	
6	PX SEND RANGE	:TQ10002	1407	148K	166 (83)	00:00:02			Q1,02	P->P	RANGE
7	HASH GROUP BY		1407	148K	166 (83)	00:00:02			Q1,02	PCWP	
* 8	HASH JOIN		1407	148K	165 (83)	00:00:02			Q1,02	PCWP	
9	PART JOIN FILTER CREATE	:BF0000	1408	78848	7 (29)	00:00:01			Q1,02	PCWP	
10	PX RECEIVE		1408	78848	7 (29)	00:00:01			Q1,02	PCWP	
11	PX SEND PARTITION (KEY)	:TQ10001	1408	78848	7 (29)	00:00:01			Q1,01	P->P	PART (KEY)
* 12	HASH JOIN		1408	78848	7 (29)	00:00:01			Q1,01	PCWP	
13	PX RECEIVE		139	3614	2 (0)	00:00:01			Q1,01	PCWP	
14	PX SEND BROADCAST	:TQ10000	139	3614	2 (0)	00:00:01			Q1,00	P->P	BROADCAST
15	PX BLOCK ITERATOR		139	3614	2 (0)	00:00:01			Q1,00	PCWC	
16	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	139	3614	2 (0)	00:00:01			Q1,00	PCWP	
17	PX BLOCK ITERATOR		1408	42240	4 (25)	00:00:01	1	4	Q1,01	PCWC	
18	TABLE ACCESS FULL	T_ACCT_MASTER_HD	1408	42240	4 (25)	00:00:01	1	64	Q1,01	PCWP	
19	PX PARTITION RANGE SINGLE		18919	960K	157 (85)	00:00:02	14	14	Q1,02	PCWC	
20	PX PARTITION HASH JOIN-FILTER		18919	960K	157 (85)	00:00:02	:BF0000	:BF0000	Q1,02	PCWC	
* 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	18919	960K	157 (85)	00:00:02	53	56	Q1,02	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: 429755685

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		398	00:00:01.42	101			
1	PX COORDINATOR		1		398	00:00:01.42	101			
2	PX SEND QC (RANDOM)	:TQ10002	0	398	0	00:00:00.01	0			
3	HASH GROUP BY		0	398	0	00:00:00.01	0	717K	717K	370K (0)
4	PX RECEIVE		0	398	0	00:00:00.01	0			
5	PX SEND HASH	:TQ10001	0	398	0	00:00:00.01	0			
6	HASH GROUP BY		0	398	0	00:00:00.01	0	717K	717K	362K (0)
* 7	HASH JOIN		0	15289	0	00:00:00.01	0	791K	791K	223K (0)
8	BUFFER SORT		0		0	00:00:00.01	0	40960	40960	36864 (0)
9	PX RECEIVE		0	766	0	00:00:00.01	0			
10	PX SEND BROADCAST	:TQ10000	0	766	0	00:00:00.01	0			
11	TABLE ACCESS FULL	PRODUCTS	1	766	766	00:00:00.01	29			
12	PX BLOCK ITERATOR		0	15289	0	00:00:00.01	0			
* 13	TABLE ACCESS FULL	SALES	0	15289	0	00:00:00.01	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

Cluster Database: DBM > Database Instance: DBM\_DBM2 > Monitored SQL Executions >  
Monitored SQL Execution Details

Logged in As SYS

Text Report

## Overview

SQL ID 94v66p38f1uqt  
Parallel 128 7  
Execution Started Fri Mar 20 2009 04:38:11 PM  
Last Refresh Time Fri Mar 20 2009 04:38:19 PM  
Execution ID 33554432  
Session 464  
Fetch Calls 1

## Time

Duration 9.0s  
Database Time 4.5m  
PL/SQL & Java 0.0s

## IO & Wait Statistics

IO Count 878  
Buffer Gets 41K  
Wait Activity % 100

利用 SQL 监视器，您可以比较计划中每个操作的  
估算返回行数与实际返回行数

## Details

Plan Statistics Parallel Activity

Plan Hash Value 1101229021

Operation	Name	Estimate...	Cost	Timeline(9s)	Exec...	Actual...	Temp...	CPU Activity %	Wait Activity %
SELECT STATEMENT			13		1	1			
SORT AGGREGATE		1			1	1			
PX COORDINATOR					225	128		1.18	6.67
PX SEND QC (RANDOM)	:TQ10002	1			112	112			
SORT AGGREGATE		1			112	112			
HASH JOIN		2298K	13		112	967K	150M		
PX RECEIVE		329K	6		112	242K		0.39	
PX SEND HASH	:TQ10000	329K	6		112	247K		0.39	
PX BLOCK ITERATOR		329K	6		112	247K			
TABLE ACCESS	MY_OBJECTS	329K	6		1165	247K		98	93
PX RECEIVE		329K	6		112	242K			
PX SEND HASH	:TQ10001	329K	6		112	239K			
PX BLOCK ITERATOR		329K	6		112	239K			
TABLE ACCESS	MY_OBJECTS	329K	6		1134	239K			

# 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

\*柱状图会对具有 11g 之前的绑定的语句产生令人瞩目的副作用 请谨慎使用

# 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 <b>WHERE</b> 子句谓词的行。用于索引、DOP 集等
按 ROWID 访问表	<b>ROWID</b> 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 <b>WHERE</b> 子句中提供 <b>rowid</b> 时使用
索引唯一扫描	将只返回一行。当语句中包含 <b>UNIQUE</b> 或 <b>PRIMARY KEY</b> 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 <b>ROWID</b> 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（<.>、 <b>between</b> 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 <b>order by</b> 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 <b>FTS</b> 。使用多块 <b>IO</b> ，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 <b>rowid</b> 。可以有效地合并对应于 <b>WHERE</b> 子句中的多个条件的索引



# 访问方法

Plan hash value: 2011745446

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		1407	54873	166 (83)	00:00:02					
1	PX COORDINATOR										
2	PX SEND QC (ORDER)	:TQ10003	1407	54873	166 (83)	00:00:02			Q1,03	P->S	QC (ORDER)
3	VIEW		1407	54873	166 (83)	00:00:02			Q1,03	PCWP	
4	SORT GROUP BY		1407	148K	166 (83)	00:00:02			Q1,03	PCWP	
5	PX RECEIVE		1407	148K	166 (83)	00:00:02			Q1,03	PCWP	
6	PX SEND RANGE	:TQ10002	1407	148K	166 (83)	00:00:02			Q1,02	P->P	RANGE
7	HASH GROUP BY		1407	148K	166 (83)	00:00:02			Q1,02	PCWP	
* 8	HASH JOIN		1407	148K	165 (83)	00:00:02			Q1,02	PCWP	
9	PART JOIN FILTER CREATE	:BF0000	1408	78848	7 (29)	00:00:01			Q1,02	PCWP	
10	PX RECEIVE		1408	78848	7 (29)	00:00:01			Q1,02	PCWP	
11	PX SEND PARTITION (KEY)	:TQ10001	1408	78848	7 (29)	00:00:01			Q1,01	P->P	PART (KEY)
* 12	HASH JOIN		1408	78848	7 (29)	00:00:01			Q1,01	PCWP	
13	PX RECEIVE		139	3614	2 (0)	00:00:01			Q1,01	PCWP	
14	PX SEND BROADCAST	:TQ10000	139	3614	2 (0)	00:00:01			Q1,00	P->P	BROADCAST
15	PX BLOCK ITERATOR		139	3614	2 (0)	00:00:01			Q1,00	PCWC	
16	TABLE ACCESS FULL	HOGAN_PCODE_HB_REF	139	3614	2 (0)	00:00:01			Q1,00	PCWP	
17	PX BLOCK ITERATOR		1408	42240	4 (25)	00:00:01	1	4	Q1,01	PCWC	
18	TABLE ACCESS FULL	T_ACCT_MASTER_HB	1408	42240	4 (25)	00:00:01	1	84	Q1,01	PCWP	
19	PX PARTITION RANGE SINGLE		18919	960K	157 (85)	00:00:02	14	14	Q1,02	PCWC	
20	PX PARTITION HASH JOIN-FILTER		18919	960K	157 (85)	00:00:02	:BF0000	:BF0000	Q1,02	PCWC	
* 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HB	18919	960K	157 (85)	00:00:02	53	56	Q1,02	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)

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');**

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10000	136K	2 (0)	00:00:01
1	INLIST ITERATOR					
2	TABLE ACCESS BY INDEX ROWID	COUNTRIES	10000	136K	2 (0)	00:00:01
* 3	INDEX UNIQUE SCAN	COUNTRY_PK	3		1 (0)	00:00:01

**Select country\_id, name from countries where country\_id between 'AU' and 'IE';**

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10000	136K	2 (0)	00:00:01
1	TABLE ACCESS BY INDEX ROWID	COUNTRIES	10000	136K	2 (0)	00:00:01
* 2	INDEX RANGE SCAN	COUNTRY_PK	7		1 (0)	00:00:01

**Select country\_id, name from countries where name='USA';**

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	14	2 (0)	00:00:01
* 1	TABLE ACCESS FULL	COUNTRIES	1	14	2 (0)	00:00:01

# 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

\*柱状图会对具有 11g 之前的绑定的语句产生令人瞩目的副作用 请谨慎使用



# 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

# 联接类型

Plan hash value: 2011745446

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

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

# 联接类型示例 1

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

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		19	722	3 (0)	00:00:01
1	NESTED LOOPS					
2	NESTED LOOPS		19	722	3 (0)	00:00:01
* 3	TABLE ACCESS FULL	DEPARTMENTS	2	32	2 (0)	00:00:01
* 4	INDEX RANGE SCAN	EMP_DEPARTMENT_IX	10		0 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	10	220	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")

## 联接类型示例 2

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)
0	SELECT STATEMENT		665	13300	8	(25)
* 1	HASH JOIN		665	13300	8	(25)
2	TABLE ACCESS FULL	ORDERS	105	840	4	(25)
3	TABLE ACCESS FULL	ORDER_ITEMS	665	7980	4	(25)

Predicate Information (identified by operation id):

1 - access("L"."ORDER\_ID"="O"."ORDER\_ID")



## 联接类型示例 3

应为此查询使用什么联接类型？

```
SELECT o.order_id,o.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	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		11235	120K	33 (7)	00:00:01
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
4	INDEX FAST FULL SCAN	EMP_NAME_IX	107	749	0 (0)	00:00:01

## 联接类型示例 4

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

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		122	4758	6 (34)	00:00:01
1	SORT ORDER BY		122	4758	6 (34)	00:00:01
2	VIEW	VW_FOJ_O	122	4758	5 (20)	00:00:01
* 3	HASH JOIN FULL OUTER		122	1342	5 (20)	00:00:01
4	INDEX FAST FULL SCAN	DEPT_ID_PK	27	108	2 (0)	00:00:01
5	TABLE ACCESS FULL	EMPLOYEES	107	749	2 (0)	00:00:01

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

# 连接顺序

Plan hash value: 2011745446

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

需要从可最大程度减少结果集行数的表开始操作

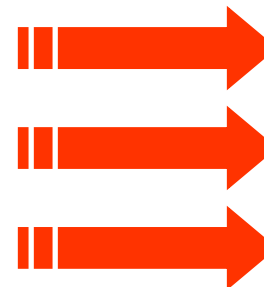
如果联接顺序不正确，请检查统计信息、基数及访问方法

# 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

# 分区修剪

Plan hash value: 2011745446

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

**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

# 并行度

Plan hash value: 2011745446

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

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





## 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	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		17	153	565 (100)	00:00:07					
1	PX COORDINATOR										
2	PX SEND QC (RANDOM)	:TQ10001	17	153	565 (100)	00:00:07			Q1,01	P->S	QC (RAND)
3	HASH GROUP BY		17	153	565 (100)	00:00:07			Q1,01	PCWP	
4	PX RECEIVE		17	153	565 (100)	00:00:07			Q1,01	PCWP	
5	PX SEND HASH	:TQ10000	17	153	565 (100)	00:00:07			Q1,00	P->P	HASH
6	HASH GROUP BY		17	153	565 (100)	00:00:07			Q1,00	PCWP	
7	PX BLOCK ITERATOR		10M	85M	60 (97)	00:00:01	1	16	Q1,00	PCWC	
* 8	TABLE ACCESS FULL	SALES	10M	85M	60 (97)	00:00:01	1	16	Q1,00	PCWP	

Predicate Information (identified by operation id):

8 - filter("CUST\_ID"<=22810 AND "CUST\_ID">=22300)

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

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

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

Note

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

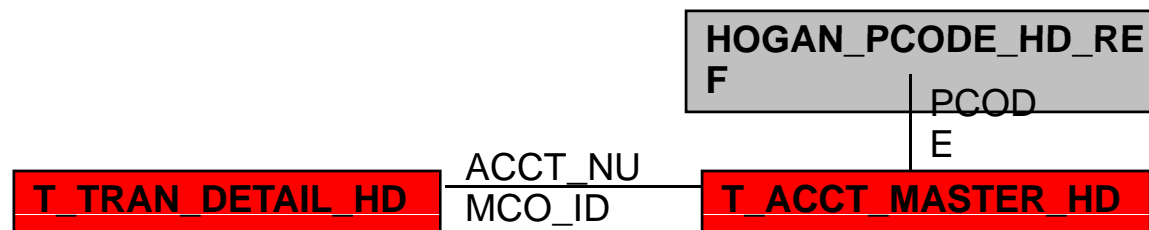
显示 PQ 服务器如何在彼此之间分发

# 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 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 )
```



 数 TB 大小

 1 GB 大小



# 执行计划示例（续）

1. 检查所返回的行数是否大致正确

Plan hash value: 2011745446

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

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)

2. 基数估算是否正确？

3. 访问方法是否正确？

意味着没有收集到统计信息，  
强烈表明这不是最佳计划

# 执行计划示例（续）

Plan hash value: 2011745446

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		1407	54873	166 (83)	00:00:02					
1	PX COORDINATOR										
2	PX SEND QC (ORDER)	:TQ10003	1407	54873	166 (83)	00:00:02			Q1,03	P->S	QC (ORDER)
3	VIEW		1407	54873	166 (83)	00:00:02			Q1,03	PCMP	
4	SORT GROUP BY		1407	148K	166 (83)	00:00:02			Q1,03	PCMP	
5	PX RECEIVE		1407	148K	166 (83)	00:00:02			Q1,03	PCMP	
6	PX SEND RANGE	:TQ10002	1407	148K	166 (83)	00:00:02			Q1,02	P->P	RANGE
7	HASH GROUP BY		1407	148K	166 (83)	00:00:02			Q1,02	PCMP	
* 8	HASH JOIN		1407	148K	166 (83)	00:00:02			Q1,02	PCMP	
9	PART JOIN FILTER CREATE	:BF0000	1408	78848	7 (29)	00:00:01			Q1,02	PCMP	
10	PX RECEIVE		1408	78848	7 (29)	00:00:01			Q1,02	PCMP	
11	PX SEND PARTITION (KEY)	:TQ10001	1408	78848	7 (29)	00:00:01			Q1,01	P->P	PART (KEY)
* 12	HASH JOIN		1408	78848	7 (29)	00:00:01			Q1,01	PCMP	
13	PX RECEIVE		139	3614	2 (0)	00:00:01			Q1,01	PCMP	
14	PX SEND BROADCAST	:TQ10000	139	3614	2 (0)	00:00:01			Q1,00	P->P	BROADCAST
15	PX BLOCK ITERATOR		139	3614	2 (0)	00:00:01			Q1,00	PCWC	
16	1 TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	139	3614	2 (0)	00:00:01			Q1,00	PCMP	
17	2 PX BLOCK ITERATOR		1408	42240	4 (25)	00:00:01	1	4	Q1,01	PCWC	
18	2 TABLE ACCESS FULL	T_ACCT_MASTER_HD	1408	42240	4 (25)	00:00:01	1	64	Q1,01	PCMP	
19	PX PARTITION RANGE SINGLE		13919	960K	157 (85)	00:00:02	14	14	Q1,02	PCWC	
20	PX PARTITION HASH JOIN-FILTER		13919	960K	157 (85)	00:00:02	:BF00001:BF00001		Q1,02	PCWC	
* 21	3 TABLE ACCESS FULL	T_TRAN_DETAIL_HD	13919	960K	157 (85)	00:00:02	53	56	Q1,02	PCMP	

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)

4. 是否已发生分区修剪?

5. 是否使用了正确的联接方法?

6. 联接顺序是否正确? 是否首先访问可消除最多行的表?

# 执行计划示例（续）

Plan hash value: 2011745446

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

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

# 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

Plan hash value: 3998295633

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		1	39	40 (20)	00:00:01					
1	PX COORDINATOR		1	39	40 (20)	00:00:01					
2	PX SEND QC (ORDER)	:TQ10004	1	39	40 (20)	00:00:01			Q1,04	P->S	QC (ORDER)
3	VIEW		1	39	40 (20)	00:00:01			Q1,04	PCWP	
4	SORT ORDER BY		1	47	40 (20)	00:00:01			Q1,04	PCWP	
5	PX RECEIVE		1	47	40 (20)	00:00:01			Q1,04	PCWP	
6	PX SEND RANGE	:TQ10003	1	47	40 (20)	00:00:01			Q1,03	P->P	RANGE
7	SORT GROUP BY		1	47	40 (20)	00:00:01			Q1,03	PCWP	
* 8	HASH JOIN		1	47	40 (20)	00:00:01			Q1,03	PCWP	
9	PX RECEIVE		1	16	37 (19)	00:00:01			Q1,03	PCWP	
10	PX SEND HASH	:TQ10001	1	16	37 (19)	00:00:01			Q1,01	P->P	HASH
11	VIEW	VW_GBC_10	1	16	37 (19)	00:00:01			Q1,01	PCWP	
12	HASH GROUP BY		1	33	37 (19)	00:00:01			Q1,01	PCWP	
13	PX RECEIVE		1	33	37 (19)	00:00:01			Q1,01	PCWP	
14	PX SEND HASH	:TQ10000	1	33	37 (19)	00:00:01			Q1,00	P->P	HASH
15	HASH GROUP BY		1	33	37 (19)	00:00:01			Q1,00	PCWP	
16	PX PARTITION HASH ALL		13708	441K	34 (12)	00:00:01	1	4	Q1,00	PCWC	
* 17	HASH JOIN		13708	441K	34 (12)	00:00:01			Q1,00	PCWP	
18	PX PARTITION RANGE ALL		1280	7920	7 (15)	00:00:01	1	16	Q1,00	PCWC	
19	TABLE ACCESS FULL	T_ACCT_MASTER_HD	1280	7920	7 (15)	00:00:01	1	64	Q1,00	PCWP	
20	PX PARTITION RANGE SINGLE		16000	296K	27 (12)	00:00:01	14	14	Q1,00	PCWC	
* 21	TABLE ACCESS FULL	T_TRAN_DETAIL_HD	16000	296K	27 (12)	00:00:01	53	56	Q1,00	PCWP	
22	PX RECEIVE		139	4309	2 (0)	00:00:01			Q1,03	PCWP	
23	PX SEND HASH	:TQ10002	139	4309	2 (0)	00:00:01			Q1,03	P->P	HASH
24	PX BLOCK ITERATOR		139	4309	2 (0)	00:00:01			Q1,03	PCWC	
25	TABLE ACCESS FULL	HOGAN_PCODE_HD_REF	139	4309	2 (0)	00:00:01			Q1,03	PCWP	

Predicate Information (identified by operation id):

8 - access("ITEM\_1"="B","PCODE")  
17 - access("A"."ACCT\_NUM"="C","ACCT\_NUM" AND "A"."CO\_ID"="C"."CO\_ID")  
21 - filter("C"."ASOF\_YYYYMM"=200102 AND "C"."TRAN\_AMT"<2000000000)

4. 分区修剪，  
一个范围分区，  
四个散列分区

7. 并行执行计划的所有方面

8. 现在的行分布都是散列分布

2. 基数是正确的，  
并且对于每个联接，  
行数减少

6. 联接顺序已改变 — PWJ，将散列联接  
接到查找表

5. 联接类型仍然是散列联接，  
但现在是 PWJ

3. 访问方法保持不变

# 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？

PLAN\_TABLE\_OUTPUT

Plan hash value: 2489314924

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		960	2880	5 (0)	00:00:01		
1	PARTITION RANGE ALL		960	2880	5 (0)	00:00:01	1	16
2	TABLE ACCESS FULL	SALES	960	2880	5 (0)	00:00:01	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

