Pivotal

从查询计划入手优化数据库

郭峰
Pivotal 中国研发中心

Agenda

- Greenplum 查询处理过程
- Greenplum 查询计划解读
- Greenplum 查询调优
- Q+A

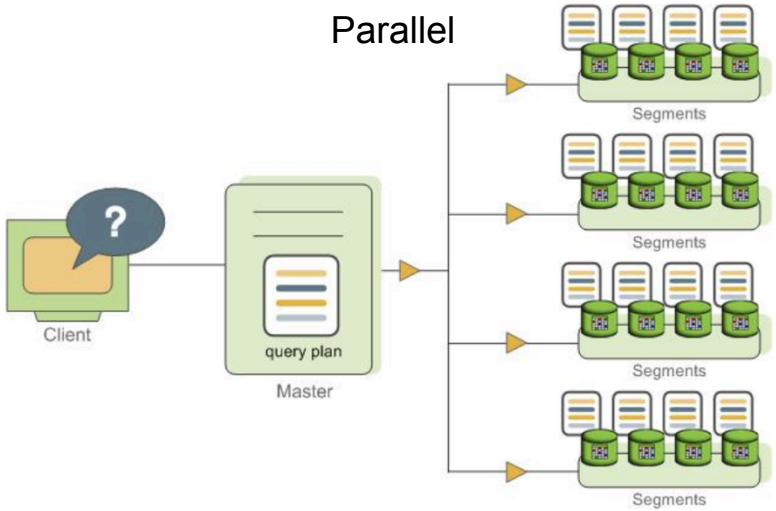
Pivotal

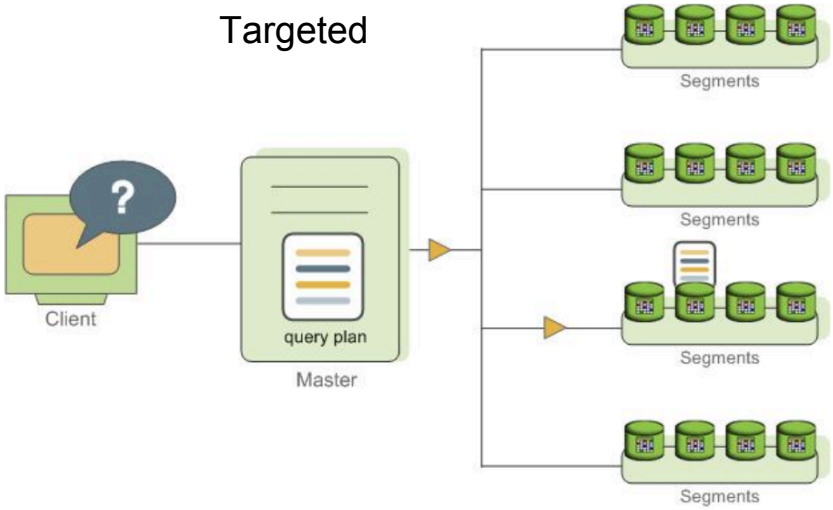
Greenplum 查询处理过程

Greenplum 查询处理过程

Greenplum 对查询的处理过程可以分成四个步骤:

1. Master 接收查询语句并生成查询计划。
2. Master 把查询计划分发到Segments.

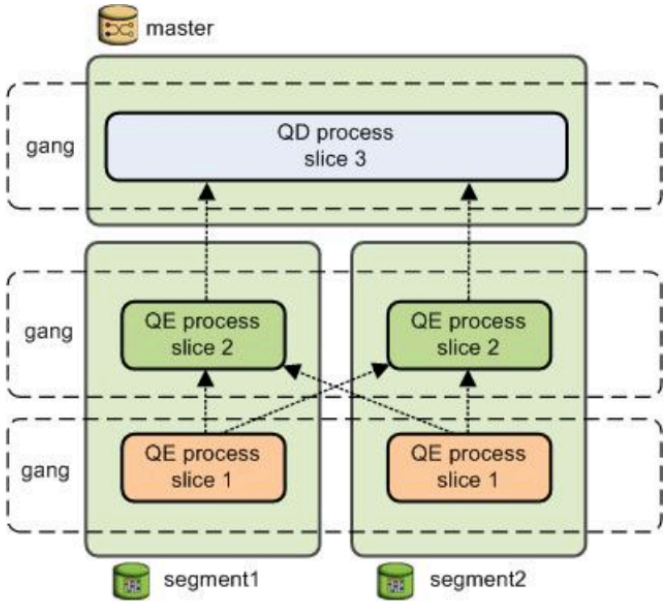




Greenplum 查询处理过程

Greenplum 对查询的处理过程可以分成四个步骤:

1. Master 接收查询语句并生成查询计划。
2. Master 把查询计划分发到Segments.
3. Segments 并发在各自本地的数据集上执行计划。
    a. Slice: A portion of the plan that segments can work on independently.
    a. Gang: Related processes that are working on the same slice of the query plan but on different segments.
1. Master 收集结果并返回给客户端。



Greenplum 查询计划解读

Pivotal

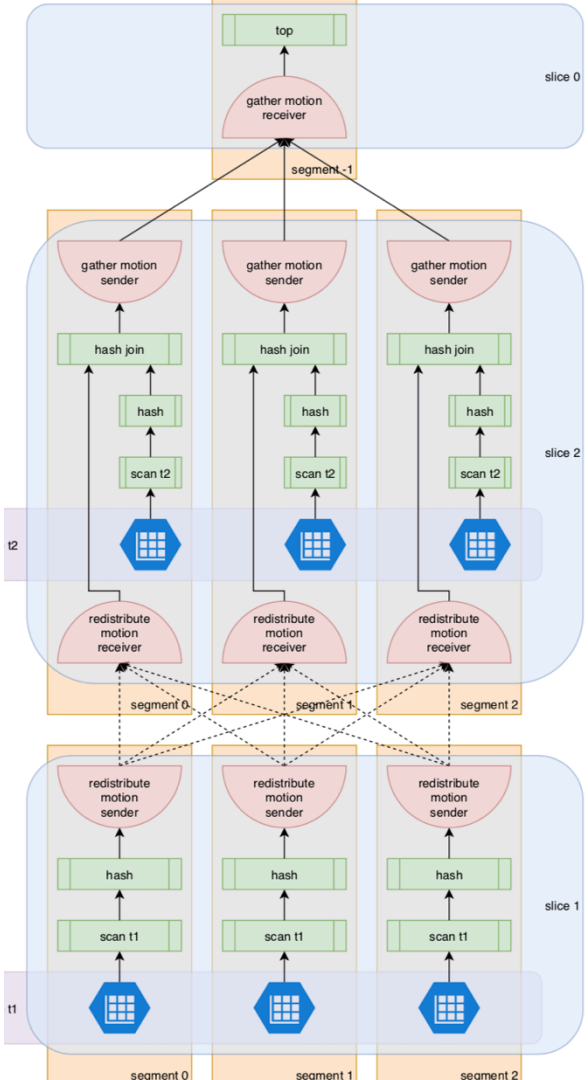
Greenplum 查询计划解读

```
gpadmin=# EXPLAIN select * from t2 join t1 on t2.c1 = t1.c2;

                                          QUERY PLAN
-----------------------------------------------------------------------------------------
----------------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=2037.25..97600.62 rows=7413210
width=16)
   ->  Hash Join  (cost=2037.25..97600.62 rows=2471070 width=16)
         Hash Cond: t1.c2 = t2.c1
         ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..2683.00
rows=28700 width=8)
               Hash Key: t1.c2
               ->  Seq Scan on t1  (cost=0.00..961.00 rows=28700 width=8)
         ->  Hash  (cost=961.00..961.00 rows=28700 width=8)
               ->  Seq Scan on t2  (cost=0.00..961.00 rows=28700 width=8)
 Optimizer status: legacy query optimizer
(9 rows)
```

6个节点，其中2个Motion节点。

```
select * from t2 join
t1 on t2.c1 = t1.c2
```



Motion: 在Segments之间传输数据

Slice: 根据Motion切割slice, motion的两边各有一个slice.

Pivotal

Greenplum 查询计划解读

每个节点含有三个估计值：cost, rows, width.

- Cost: Greenplum采用基于代价的优化器来评估执行查询的不同策略，并选择代价最低的方法。
  - 启动代价: 得到第一个tuple的代价。
  - 总代价： 处理完所有tuple的代价。
- Rows: 查询节点输出的tuple的个数 （估计值）。
- Width: 查询节点输出的tuple的长度（估计值），单位是字节。

Greenplum 查询计划解读

EXPLAIN ANALYZE

- 执行查询的总时间（以毫秒为单位）
- 内存使用量；
- 查询节点需要的工作进程个数
- 每个操作中处理最多行的Segment处理的最大行数以及Segment的序号
- 从处理最多行的Segment上获得第一行花费的时间（单位是毫秒）及从该Segment获得所有行花费的时间。

EXPLAIN ANALYZE 除了显示查询计划外还会执行查询，若需对*DML*语句使用 *EXPLAIN ANALYZE* 而不影响数据，可置 *EXPLAIN ANALYZE* 于事务中：

```
BEGIN;
EXPLAIN ANALYZE ...;
ROLLBACK;
```

```
Gather Motion 3:1  (slice2; segments: 3)  (cost=2360.80..7000.90 rows=99992 width=16)
  Rows out:  33333 rows at destination with 26 ms to first row, 93 ms to end, start offset by 0.530 ms.
  ->  Hash Join  (cost=2360.80..7000.90 rows=33331 width=16)
        Hash Cond: t1.c2 = t2.c1
        Rows out:  Avg 11111.0 rows x 3 workers.  Max 11246 rows (seg1) with 24 ms to first row, 83 ms to end, start offset by 0.821 ms.
        Executor memory:  1042K bytes avg, 1043K bytes max (seg0).
        Work_mem used:  1042K bytes avg, 1043K bytes max (seg0). Workfile: (0 spilling)
        (seg1)   Hash chain length 2.3 avg, 9 max, using 14237 of 16384 buckets.
        ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..3137.97 rows=33633 width=8)
              Hash Key: t1.c2
              Rows out:  Avg 33333.3 rows x 3 workers at destination.  Max 33468 rows (seg0) with 0.433 ms to first row, 40 ms to end, start o
ffset by 25 ms.
              ->  Seq Scan on t1  (cost=0.00..1119.99 rows=33633 width=8)
                    Rows out:  Avg 33333.3 rows x 3 workers.  Max 33348 rows (seg0) with 0.048 ms to first row, 4.813 ms to end, start offset
by 0.991 ms.
        ->  Hash  (cost=1110.91..1110.91 rows=33331 width=8)
              Rows in:  Avg 33333.3 rows x 3 workers.  Max 33348 rows (seg0) with 24 ms to end, start offset by 0.904 ms.
              ->  Seq Scan on t2  (cost=0.00..1110.91 rows=33331 width=8)
                    Rows out:  Avg 33333.3 rows x 3 workers.  Max 33348 rows (seg0) with 0.045 ms to first row, 4.892 ms to end, start offset
by 0.905 ms.
Slice statistics:
  (slice0)    Executor memory: 386K bytes.
  (slice1)    Executor memory: 212K bytes avg x 3 workers, 212K bytes max (seg0).
  (slice2)    Executor memory: 4767K bytes avg x 3 workers, 4767K bytes max (seg0).  Work_mem: 1043K bytes max.
Statement statistics:
  Memory used: 4096K bytes
Optimizer status: legacy query optimizer
Total runtime: 96.875 ms
(25 rows)
```

```
select * from t2 join t1 on
t2.c1 = t1.c2
```

Greenplum 查询计划解读

with T1 ms to first row, T2 ms to end, start offset by T3 ms

- T1: 这个查询节点上得到第一个tuple的执行时间
- T2: 这个查询节点上得到所有tuple的总执行时间
- T3: 这个查询节点执行开始时间 减去 这个查询的开始时间
  - Interconnect的建立
  - 分发查询计划
  - 创建需要的gang

Greenplum 查询计划解读

Work_mem used

- 这个查询节点执行时用到的内存大小。
- 如果分配给这个查询节点的内存量不足够用来执行操作，部分数据将溢出的文件中。

```
 Work_mem used:  68K bytes avg, 69K bytes max (seg0). Workfile: (3 spilling)
 Work_mem wanted: 1042K bytes avg, 1043K bytes max (seg0) to lessen workfile I/O affecting 3
workers.
 (seg0)   Initial batch 0:
 (seg0)     Wrote 480K bytes to inner workfile.
 (seg0)     Wrote 480K bytes to outer workfile.
 (seg0)   Initial batches 1..15:
 (seg0)     Read 611K bytes from inner workfile: 41K avg x 15 nonempty batches, 43K max.
 (seg0)     Read 613K bytes from outer workfile: 41K avg x 15 nonempty batches, 43K max.
 (seg0)   Hash chain length 2.3 avg, 10 max, using 14298 of 16384 buckets.
```

Greenplum 查询调优

Pivotal

Greenplum 查询调优

- 避免数据倾斜
- 避免使用Broadcast Motion
- 避免溢出文件
- 使用哈希聚合
- 更新统计信息
- 使用分区裁剪

避免数据倾斜

当数据在各个Segments之间分布不平衡时，就会发生数据倾斜，影响系统整体性能。

```
gpadmin=# EXPLAIN ANALYZE select * from t2 join t1 on t2.c1 = t1.c2;
                                    QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=4722.90..16391.84 rows=400529 width=16)
   Rows out:  533332 rows at destination with 60 ms to first row, 1096 ms to end, start
offset by 0.594 ms.
   ->  Hash Join  (cost=4722.90..16391.84 rows=133510 width=16)
         Hash Cond: t1.c2 = t2.c1
         Rows out:  Avg 177777.3 rows x 3 workers.  Max 444984 rows (seg1) with 262 ms to
first row, 755 ms to end, start offset by 1.020 ms.
```

ALTER TABLE SET DISTRIBUTED BY

避免使用Broadcast Motion

当对大量数据使用Broadcast Motion时，会影响查询的性能。

```
gpadmin=# select count(*) from t1;
  count
---------
 2000000
(1 row)

gpadmin=# select count(*) from t2;
  count
---------
 1000000
(1 row)
```

避免使用Broadcast Motion

```
gpadmin=# EXPLAIN select * from t2 join t1 on t2.c1 = t1.c2;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
-------------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=42883.24..116773.64 rows=875841 width=16)
   ->  Hash Join  (cost=42883.24..116773.64 rows=291947 width=16)
         Hash Cond: t1.c2 = t2.c1
         ->  Seq Scan on t1  (cost=0.00..55680.11 rows=968304 width=8)
         ->  Hash  (cost=31935.23..31935.23 rows=291947 width=8)
               ->  Broadcast Motion 3:3  (slice1; segments: 3)  (cost=0.00..31935.23
rows=291947 width=8)
                     ->  Seq Scan on t2  (cost=0.00..14418.41 rows=291947 width=8)
 Settings:  gp_segments_for_planner=1
 Optimizer status: legacy query optimizer
(9 rows)
```

Total runtime: 2826.874 ms

避免使用Broadcast Motion

```
gpadmin=# set gp_segments_for_planner to 100;
SET

gpadmin=# EXPLAIN select * from t2 join t1 on t2.c1 = t1.c2;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------
-----------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=25366.42..157355.04 rows=875841 width=16)
   ->  Hash Join  (cost=25366.42..157355.04 rows=291947 width=16)
         Hash Cond: t1.c2 = t2.c1
         ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..113778.33
rows=968304 width=8)
               Hash Key: t1.c2
               ->  Seq Scan on t1  (cost=0.00..55680.11 rows=968304 width=8)
         ->  Hash  (cost=14418.41..14418.41 rows=291947 width=8)
               ->  Seq Scan on t2  (cost=0.00..14418.41 rows=291947 width=8)
 Settings:  gp_segments_for_planner=100
 Optimizer status: legacy query optimizer
(10 rows)
```

Total runtime: 1627.801 ms

Pivotal

避免溢出文件

当分配给查询节点的内存量不足以用来执行操作时，部分数据会溢出到文件中，查询性能会受到影响。

```
gpadmin=# set statement_mem to 500;
SET

->  Hash Join  (cost=2340.64..7998.94 rows=16721 width=16)
      Hash Cond: t1.c2 = t2.c1
      Rows out:  Avg 11111.0 rows x 3 workers.  Max 11246 rows (seg1) with 27 ms to first
row, 120 ms to end, start offset by 0.913 ms.
      Executor memory:  101K bytes avg, 101K bytes max (seg0).
      Work_mem used:  101K bytes avg, 101K bytes max (seg0). Workfile: (3 spilling)
      Work_mem wanted: 1042K bytes avg, 1043K bytes max (seg0) to lessen workfile I/O
affecting 3 workers.
      (seg0)   Initial batch 0:
      (seg0)     Wrote 480K bytes to inner workfile.
      (seg0)     Wrote 480K bytes to outer workfile.
```

避免溢出文件

```
gpadmin=# set statement_mem to 4096;
SET
```

```
->  Hash Join  (cost=2340.64..7998.94 rows=16721 width=16)
      Hash Cond: t1.c2 = t2.c1
      Rows out:  Avg 11111.0 rows x 3 workers.  Max 11246 rows (seg1) with 26 ms to first
row, 85 ms to end, start offset by 0.721 ms.
      Executor memory:  1042K bytes avg, 1043K bytes max (seg0).
      Work_mem used:  1042K bytes avg, 1043K bytes max (seg0). Workfile: (0 spilling)
```

https://gpdb.docs.pivotal.io/5120/best_practices/sysconfig.html

使用哈希聚合

当需要对大量数据行进行排序和聚合时，优先使用 HashAggregate操作符。

```
gpadmin=# EXPLAIN select avg(t2.c1) from t2 join t1 on t2.c1 = t1.c2 group by t2.c1;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------
---------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=53664.12..57902.73 rows=98772 width=36)
   ->  GroupAggregate  (cost=53664.12..57902.73 rows=32924 width=36)
         Group By: t2.c1
         ->  Sort  (cost=53664.12..54665.44 rows=133510 width=4)
               Sort Key: t2.c1
               ->  Hash Join  (cost=4722.90..16391.84 rows=133510 width=4)
                     Hash Cond: t1.c2 = t2.c1
                     ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..6166.92 rows=66055
width=4)
                           Hash Key: t1.c2
                           ->  Seq Scan on t1  (cost=0.00..2203.64 rows=66055 width=4)
                     ->  Hash  (cost=2222.40..2222.40 rows=66680 width=4)
                           ->  Seq Scan on t2  (cost=0.00..2222.40 rows=66680 width=4)
 Settings:  enable_hashagg=off
 Optimizer status: legacy query optimizer
(14 rows)
```

使用哈希聚合

```
gpadmin=# set enable_hashagg to on;
SET

gpadmin=# EXPLAIN select avg(t2.c1) from t2 join t1 on t2.c1 = t1.c2 group by t2.c1;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
--------------
 Gather Motion 3:1  (slice2; segments: 3)  (cost=18394.48..19629.13 rows=98772 width=36)
   ->  HashAggregate  (cost=18394.48..19629.13 rows=32924 width=36)
         Group By: t2.c1
         ->  Hash Join  (cost=4722.90..16391.84 rows=133510 width=4)
               Hash Cond: t1.c2 = t2.c1
               ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..6166.92
rows=66055 width=4)
                     Hash Key: t1.c2
                     ->  Seq Scan on t1  (cost=0.00..2203.64 rows=66055 width=4)
               ->  Hash  (cost=2222.40..2222.40 rows=66680 width=4)
                     ->  Seq Scan on t2  (cost=0.00..2222.40 rows=66680 width=4)
 Settings:  enable_hashagg=on
 Optimizer status: legacy query optimizer
(12 rows)
```

Pivotal

更新统计信息

如果查询计划的顺序不是最优的，可以尝试更新数据库的统计信息，然后重新生成计划。

```
gpadmin=# select count(*) from t1;
  count
---------
 1001000
(1 row)

gpadmin=# select count(*) from t2;
 count
--------
 100000
(1 row)
```

更新统计信息

```
gpadmin=# EXPLAIN select * from t2 join t1 on t2.c1 = t1.c1;
                                      QUERY PLAN
-------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)  (cost=6496.01..9712.18 rows=17 width=16)
   ->  Hash Join  (cost=6496.01..9712.18 rows=6 width=16)
         Hash Cond: t2.c1 = t1.c1
         ->  Seq Scan on t2  (cost=0.00..2998.20 rows=29040 width=8)
         ->  Hash  (cost=6496.00..6496.00 rows=1 width=8)
               ->  Seq Scan on t1  (cost=0.00..6496.00 rows=1 width=8)
 Optimizer status: legacy query optimizer
(7 rows)
```

Total runtime: 437.215 ms

更新统计信息

```
gpadmin=# ANALYZE ;
ANALYZE
```

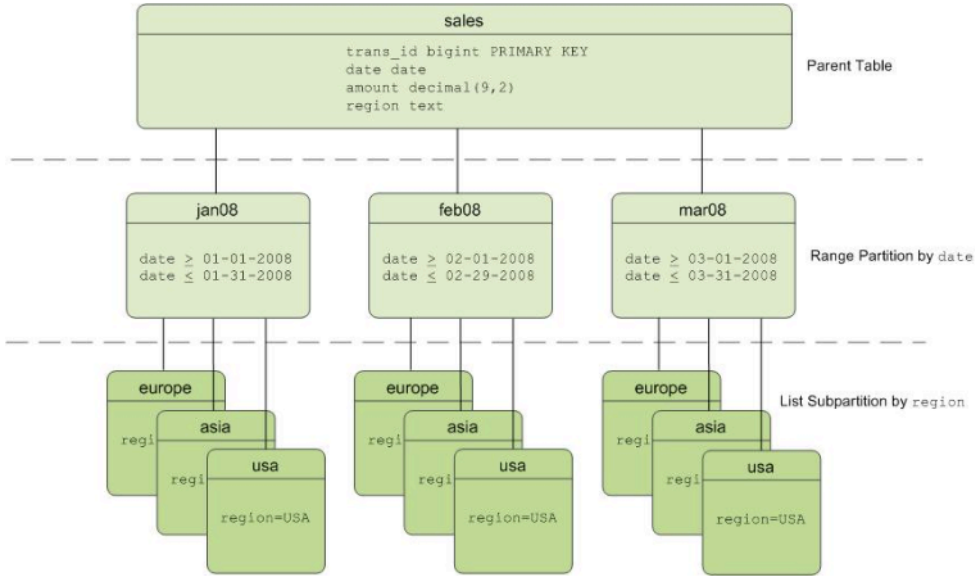
```
gpadmin=# EXPLAIN select * from t2 join t1 on t2.c1 = t1.c1;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)  (cost=5827.31..24688.70 rows=164722 width=16)
   ->  Hash Join  (cost=5827.31..24688.70 rows=54908 width=16)
         Hash Cond: t1.c1 = t2.c1
         ->  Seq Scan on t1  (cost=0.00..14961.10 rows=245504 width=8)
         ->  Hash  (cost=3771.58..3771.58 rows=54820 width=8)
               ->  Seq Scan on t2  (cost=0.00..3771.58 rows=54820 width=8)
 Optimizer status: legacy query optimizer
(7 rows)
```

Total runtime: 305.593 ms

使用分区裁剪

如果有分区表，判断分区裁剪是否有效。分区裁剪需要查询条件（WHERE 子句）必须和分区条件一样。

```
CREATE TABLE sales (trans_id int, date date, amount
decimal(9,2), region text)
DISTRIBUTED BY (trans_id)
PARTITION BY RANGE (date)
SUBPARTITION BY LIST (region)
SUBPARTITION TEMPLATE
( SUBPARTITION usa VALUES ('usa'),
  SUBPARTITION asia VALUES ('asia'),
  SUBPARTITION europe VALUES ('europe'))
  (START (date '2008-01-01') INCLUSIVE
   END (date '2009-01-01') EXCLUSIVE
   EVERY (INTERVAL '1 month'));
```



使用分区裁剪

```
gpadmin=# EXPLAIN SELECT * FROM sales WHERE date='05-07-08' AND amount=100;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
--
 Gather Motion 3:1  (slice1; segments: 3)  (cost=0.00..2073.00 rows=4 width=53)
   ->  Append  (cost=0.00..2073.00 rows=2 width=53)
         ->  Seq Scan on sales_1_prt_5_2_prt_usa sales  (cost=0.00..691.00 rows=1 width=53)
               Filter: date = '2008-05-07'::date AND amount = 100::numeric
         ->  Seq Scan on sales_1_prt_5_2_prt_asia sales  (cost=0.00..691.00 rows=1 width=53)
               Filter: date = '2008-05-07'::date AND amount = 100::numeric
         ->  Seq Scan on sales_1_prt_5_2_prt_europe sales  (cost=0.00..691.00 rows=1
width=53)
               Filter: date = '2008-05-07'::date AND amount = 100::numeric
 Optimizer status: legacy query optimizer
(9 rows)
```

使用分区裁剪

```
gpadmin=# EXPLAIN SELECT * FROM sales WHERE date='05-07-08' AND region='usa';
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)  (cost=0.00..691.00 rows=2 width=53)
   ->  Append  (cost=0.00..691.00 rows=1 width=53)
         ->  Seq Scan on sales_1_prt_5_2_prt_usa sales  (cost=0.00..691.00 rows=1 width=53)
               Filter: date = '2008-05-07'::date AND region = 'usa'::text
 Optimizer status: legacy query optimizer
(5 rows)
```

Transforming How The World Builds Software