クエリプランの読み方

July, 2018

原文はこちら

MICRO FOCUS

目次

クエリプラン....3
EXPLAIN ステートメント....3
EXPLAIN 出力オプション....3
EXPLAIN 結果....4
EXPLAIN のテキスト出力....4
EXPLAIN のグラフィカル出力....6
EXPLAIN LOCAL VERBOSE の結果....10
EXPLAIN LOCAL VERBOSE のテキスト出力....11
EXPLAIN LOCAL VERBOSE のグラフィカル出力....11
EXPLAIN LOCAL VERBOSE の単純化された結合順序....12
EXPLAIN LOCAL VERBOSE のリソース情報....13
詳細情報....13

本書は、次の記事を含む、パフォーマンスチューニングに関する 3 部構成のシリーズの最初の文書です。

Part 1: Reading Query Plans（クエリプランの読み方）

Part 2: Troubleshooting Vertica Query Performance with System Tables

Part 3: Redesigning Projections for Query Optimization

クエリプラン

クエリプランは、Vertica のクエリオプティマイザが Vertica データベース上でステートメントを実行するために選択した一連の階段状のパスです。Vertica は同じ結果を得るためにさまざまな方法でクエリを実行できます。クエリのオプティマイザは、最も低い実行コストで問合せを実行するプランを作成します。Vertica ユーザーはクエリで EXPLAIN を実行して、データベースがクエリを処理する方法を分析し、Vertica がクエリを実行するために使用するパスを視覚的な表示で確認します。

EXPLAIN ステートメント

EXPLAIN ステートメントは、指定されたステートメントに対するオプティマイザの実行計画の書式付きの記述を返します。この情報を使用して、クエリを分析および調査することができます。デフォルトでは、EXPLAIN 出力はクエリプランを階層として表します。各レベルは、オプティマイザがクエリを実行するために定義する単一のデータベース処理を表します。また、EXPLAIN 出力は.DOT 言語をソースファイルに付加し、この出力をグラフィカルに表示することができます。

EXPLAIN ステートメントで以下のクエリを検討してみましょう。

```
EXPLAIN SELECT DISTINCT s.product_key, p.product_description
 FROM store.store_sales_fact s, public.product_dimension p
 WHERE s.product_key = p.product_key
 AND s.product_version = p.product_version
 AND s.store_key IN (
 SELECT store_key
 FROM store.store_dimension
 WHERE store_state = 'MA')
 ORDER BY s.product_key;
```

次のセクションでは、EXPLAIN ステートメントと出力オプションについて詳しく説明します。

EXPLAIN 出力オプション

EXPLAIN ステートメントには、次のタイプの出力があります。

- **EXPLAIN オプション**：EXPLAIN オプションは、クエリを実行せずにクエリ実行プランの詳細情報を返します。出力は、テキスト出力またはグラフィック出力として表示できます。

- **EXPLAIN LOCAL VERBOSE オプション**：EXPLAIN LOCAL VERBOSE オプションは、複雑なクエリの分析に役立つ詳細情報を返します。詳細な情報には、リソースごとのコストとメモリ使用量が含まれます。出力は、テキスト出力またはグラフィック出力として表示できます。

EXPLAIN 結果

EXPLAIN オプションは、クエリを実行せずにクエリ実行計画に関する詳細情報を戻します。クエリオプティマイザは、異なるクエリの再書込み、プロジェクションの組み合わせ、および JOIN の順序を考慮し、異なる実行計画を生成します。

最適なクエリプランを選択するために、オプティマイザは統計分析を実行します。各オペレータにコストを割り当て、最もコストのかからないプランを実行プランとして選択します。Vertica は、EXPLAIN オプションを使用して選択したクエリプランを 2 つの異なる形式で返します。

- **テキスト出力**：クエリプランのプレーンテキスト形式。クエリプランにはステップの詳細が記載されています。Verica は、下から上へのアプローチでクエリプランを実行します。
- **グラフィカル出力**：クエリプランは.DOT 言語になっています。
    - o マネージメントコンソールを使用してツリーパスを表示できます。
    - o Graphviz のような Web ビジュアライゼーションツールを使用して、グラフィカル表現を表示できます。

この表示は、プレーンテキストプランよりも詳細情報を提供し、複雑なクエリに対して読みやすさと表現力を向上させます。

EXPLAIN のテキスト出力

このドキュメントの前半に示したサンプルクエリのアクセスパス情報には、クエリのオペレータ、コスト、プロジェクション、列のマテリアライゼーション、およびパス ID が記述されています。

```
Access Path:
 +-GROUPBY HASH (SORT OUTPUT) (GLOBAL RESEGMENT GROUPS) (LOCAL RESEGMENT GROUPS) [Cost: 3M, Rows: 15M] (PATH ID:
2)
|  Group By: s.product_key, p.product_description
|  Execute on: All Nodes
| +---> JOIN HASH [Semi] [Cost: 1M, Rows: 15M] (PATH ID: 3) Inner (BROADCAST)
| |       Join Cond: (s.store_key = subQ_1.store_key)
| |       Materialize at Input: s.store_key
| |       Materialize at Output: s.product_key
| |       Execute on: All Nodes
| | +-- Outer -> JOIN HASH [Cost: 906K, Rows: 30M] (PATH ID: 4) Inner (BROADCAST)
| | |       Join Cond: (s.product_key = p.product_key) AND (s.product_version = p.product_version)
| | |       Execute on: All Nodes
| | | +-- Outer -> STORAGE ACCESS for s [Cost: 893K, Rows: 30M] (PATH ID: 5)
| | | |       Projection: store.store_sales_fact_b0
| | | |       Materialize: s.product_key, s.product_version
| | | |       Execute on: All Nodes
| | | |       Runtime Filters: (SIP2(HashJoin): s.product_key), (SIP3(HashJoin): s.product_version),
(SIP4(HashJoin): s.product_key, s.product_version), (SIP1(HashJoin): s.store_key)
| | | +-- Inner -> STORAGE ACCESS for p [Cost: 365, Rows: 60K] (PATH ID: 6)
| | | |       Projection: public.product_dimension_b0
| | | |       Materialize: p.product_key, p.product_version, p.product_description
| | | |       Execute on: All Nodes
| | +-- Inner -> SELECT [Cost: 68, Rows: 32] (PUSHED GROUPING) (PATH ID: 7)
| | |       Execute on: All Nodes
| | | +---> STORAGE ACCESS for store_dimension [Cost: 68, Rows: 32] (PATH ID: 8)
| | | |       Projection: store.store_dimension_b0
| | | |       Materialize: store_dimension.store_key
| | | |       Filter: (store_dimension.store_state = 'MA')
| | | |       Execute on: All Nodes
```

Operator

Cost

Projection

Column Materialization

Path ID

- **オペレータ**:Vertica は、クエリプランで指定されている JOIN、SORT、FILTER、LIMIT などのオペレータを、下から上へのアプローチで実行します。これらのオペレータは、下記の通りである可能性があります。

    - シングルまたはマルチスレッド
    - 次のオペレータにデータをストリーム

たとえば、オペレータの中には、SORT や GROUPBYHASH などの次のオペレータに移る前に計算を完了する必要があるものがあります。他のオペレータは、JOIN などの計算を完了する前にデータをストリームすることができます。

- **コスト**:オプティマイザは、CPU、メモリ、およびネットワークのリソースを見積るアルゴリズムを使用して、各オペレータのコストを計算します。リソース使用量の見積もりは、統計に基づいています。統計の例を次に示します。

    - テーブルの行数
    - 各列の固有値の数
    - 各列の最小値または最大値
    - 各列の値の分布のヒストグラム
    - 各列のディスク容量

定期的に統計を更新します。正確な統計情報がないと、オプティマイザはクエリのパフォーマンスに影響を与えうる準最適な計画のみ選択となります。

次のコマンドのいずれかを実行して統計を更新します。

```
=> SELECT ANALYZE_STATISTICS
=> SELECT ANALYZE_HISTOGRAM
```

詳細については、Vertica documentation の Getting Statistics and Histograms を参照してください。

- **プロジェクション**:オプティマイザは、クエリ要件を満たす可能性のあるプロジェクションの組み合わせを分析して、どのプロジェクションが計算コストが最も低いかを判断します。クエリをプランする際、オプティマイザはテーブルごとに 1 つのプロジェクションを選択します。ただし、ノードが停止している場合、オプティマイザは同じテーブルに対して 2 つのプロジェクションを選択できます。

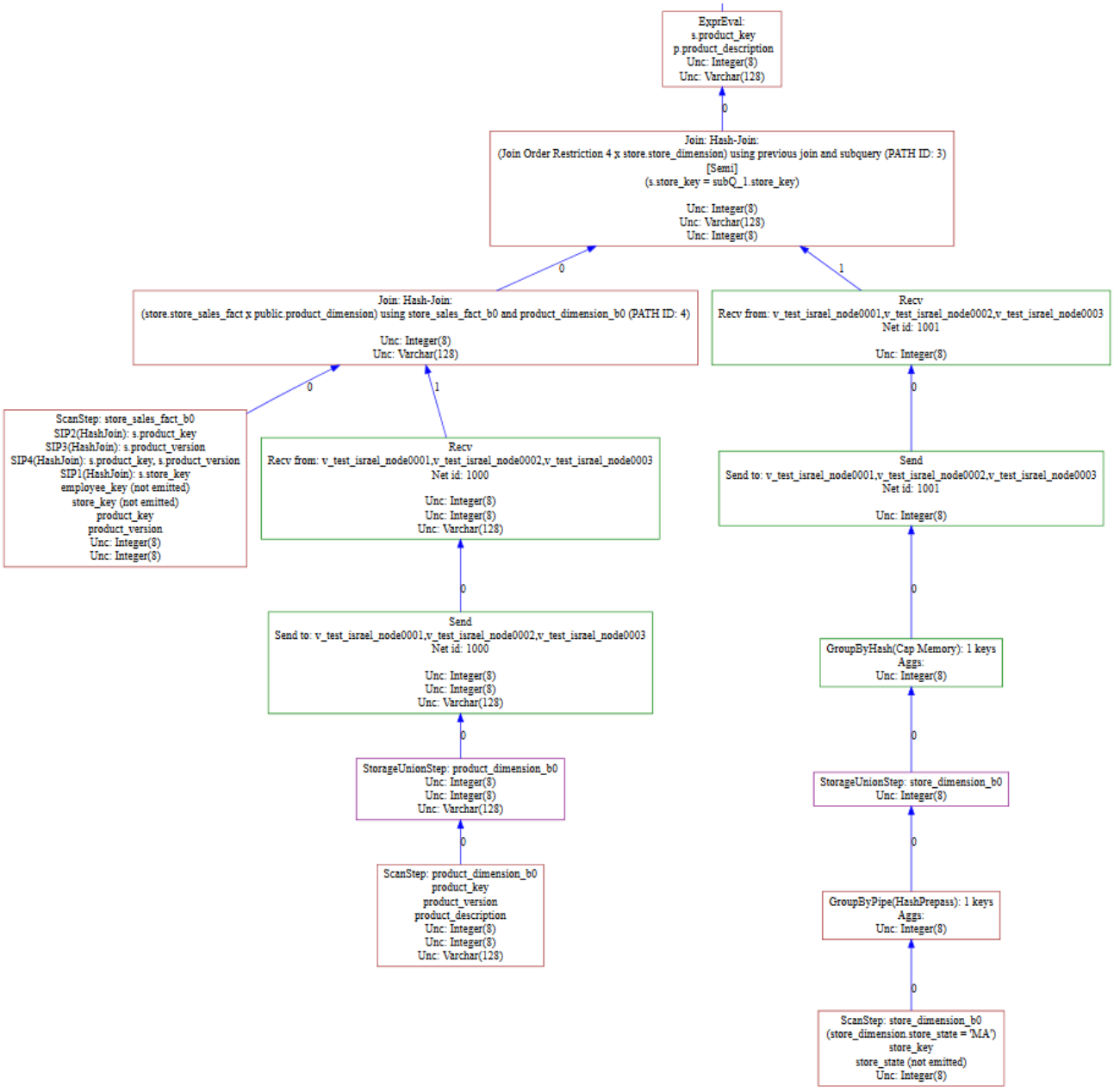
- **列のマテリアライゼーション**:列のマテリアライゼーションは、オペレータを実行するための列へのアクセスを提供します。アクセスは、列ファイルのオープン、エンコーディング、およびプロジェクション列定義に基づく非圧縮を指します。

- **パス ID**:パス ID は、オプティマイザが特定のトランザクションの各クエリ処理に割り当てる整数値です。クエリプランに複数のオペレータが含まれている場合、プロファイル情報を分析すると、パス ID によってオプティマイザはすべてのインスタンスでオペレータを識別できます。

EXPLAIN のグラフィカル出力

マネージメントコンソールを使用して、クエリプランの図を表示します。マネージメントコンソールは Vertica に同梱されています。マネージメントコンソールを使用してクエリプランを表示する方法の詳細については、Vertica documentation を参照してください。マネージメントコンソールにアクセスできない場合、オープンソースの Web ビジュアライゼーションツールを使用してグラフを表示できます。グラフィック出力は、クエリプランの詳細な情報を提供し、クエリプランの理解度を向上させます。

次の図は、グラフビジュアライゼーションツール Graphviz が、本ドキュメントの前半のサンプルクエリのグラフをどのように表示するかを示しています。次のグラフは、完全なグラフの一部です。



グラフの各オブジェクトはオペレータに対応し、ボックスはそのオペレータに関する詳細な情報を提供します。ボックスのさまざまな形と色は、データフローを視覚的に理解するのに役立ちます。

カラー	説明
緑色	

	該当のオペレータから別のオペレータにデータを渡すシングルスレッドオペレーション
茶色	マルチスレッドオペレータ。クエリが実行されるリソースプール設定で管理できるスレッドの数
紫色	データまたはストレージの統合ストリーム
青色	ディスクに書き込まれるデータターゲット

形	説明
楕円形または円形	プッシュモデル。オペレータは、その上の次のオペレータに情報を送信
長方形	プルモデル。次のオペレータは、オペレータデータバッファから情報を探索
家形	クエリが開始するルートオブジェクト

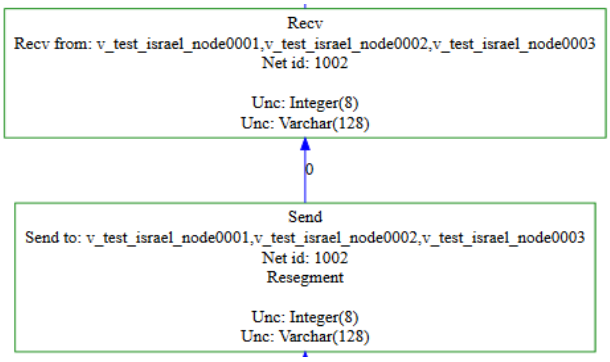
この表には、前述のグラフィカルなプランの最も重要なオペレータがリストされています。

オペレータ	説明	使用上の考慮事項
Copy	ロード中にバディープロジェクションのデータのコピーを作成	
DataTarget	ロード中にWOSからROSにデータを書き込み	
ExprEval	C1 + C2 などの式を評価	必要な列のみを選択します。
Filter	Vertica タプルを次のオペレータにフィルター	
GroupByHash	データを次のオペレータにストリーミングする前に、タプルをメモリー内のハッシュに集約	GroupByHash はすべての可能なメモリを使用します。データが使用可能なメモリを超えると、ディスクのスピルオーバーが発生します。オペレータは、次のオペレータにデータを渡す前に、すべてのオペレーションを完了する必要があります。
GroupByPipe	データを次のオペレータにストリーミングする前にソートされたタプルを集計	GroupByPipe は、GroupbyHash よりもメモリ要件が少なく、データを次のオペレータにストリームします。
Join Merge-Join	事前ソートされたタプルを結合	少ないメモリを使用します。
Join Hash-Join	JOIN オペレータの内側をメモリにロードし、ソートされていないタプルを結合	内側が大きく、データがメモリに収まらない場合、クエリは失敗します。

		内側が小さい場合、処理はマージ結合よりも高速です。
Load	ディスクからデータをロードし、入力を解析	
Merge	複数のデータストリームを 1 つのソートされたストリームにマージ	
NetworkRec NetworkSend	別のノードに送信されたデータまたは別のノードから受信されたデータに関する情報	クエリには、ネットワークオペレータの各ペア（送信および受信）のためにより多くのメモリが必要です。
ParallelMerge	ソートされたデータストリームを結合	
ParallelUnion	ソートされていないデータストリームを結合	
Root	最初のオペレータ	
Scan	フィルタを適用するためにディスクからデータを読み込み	
Sort	データストリームをソート	
StorageMerge	ソート順を維持しながらデータストレージを結合	
StorageUnion	ソート順を維持せずにデータストレージを結合	
Top-K	Top-K タプルを返す解析関数	
ValExpr	結合式を評価	

次に、サンプルクエリの 3 つのオペレータについて詳しく説明します。

- **Send もしくは Recv:** プランの途中にある送受信の緑色のセクションでは、ブロードキャストやセグメンテーションなどのネットワーク処理が表示され、誤ったプロジェクションデザインを示唆している可能性があります。



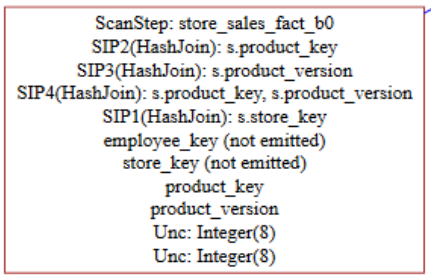
グラフでは、色が示すように、データはネットワークを介してシングルスレッドで移動します。プランの途中では、このシングルスレッディングはクエリを遅くします。これは、この処理の後のオペレータは、最初の送信処理が完了するまで実行を完了できないためです。

さらに、このネットワーク処理では、各 Send/ Recv の組み合わせにタプルを保持するネットワークバッファが必要であるため、より多くのメモリが必要です。クラスターのノードが多いほど、バッファが多く必要になります。Send /Recv オペレータは使用しないでください。そのためには、JOIN や GROUP BY を実行するときにノードでデータを利用できるように、プロジェクションを再設計します。

- **GROUPBYHASH**：GROUPBYHASH は単一スレッドの処理であり、データストリーミング処理ではありません。Vertica は、GROUPBYHASH オペレータより上のオペレータに情報を送信できるように、GROUPBY 計算全体を完了する必要があります。グラフプランでは、次のシーケンスが表示されます。

この場合、Vertica が同じパスで GROUPBYHASH 処理を実行すると、GROUPBYPIPE も実行され、ストレージコンテナから集計が開始されます。STORAGEUNIONSTEP（紫色のボックス）は、これらのオペレータの結果を組み合わせて GROUPBYHASH にデータを送信し、このパスのステートメントの集計を完了します。これらの詳細はグラフィカルなプランでのみ利用可能であり、テキストクエリプランには表示されません。

- **SIP**：グラフィカルプランは、Sidewise Information Passing（SIP）に関する情報も提供します。グラフィカルなプランでは、Vertica がタプルの派生を減らすために使用できるフィルタを SIP に表示します。JOIN オペレータを通過するタプルの数を減らすと、リソース消費が少なくなり、クエリのパフォーマンスが向上します。



EXPLAIN LOCAL VERBOSE の結果

EXPLAIN LOCAL VERBOSE は、クエリを実行するためのオプティマイザの推奨事項の詳細を検討する別の方法です。EXPLAIN LOCAL VERBOSE オプションを使用すると、複雑なクエリを確認するのに便利な情報を追加できます。EXPLAIN LOCAL VERBOSE 出力は、以下の詳細情報を提供します。

- **テキスト出力**:クエリプランのプレーンテキスト形式。クエリプランにはステップの詳細がリストされ、Vertica はボトムアップアプローチでクエリプランを実行します。このプランには、EXPLAIN 出力に表示されるのと同じオペレータとパスがあります。ただし、コスト、メモリ、各オペレータのネットワーク情報などの詳細情報も含まれています。

- **グラフィカル出力**:クエリプランは.DOT 言語になっています。

    - マネージメントコンソールを使用してツリーパスを表示できます。

    - Graphviz のような Web ビジュアライゼーションツールを使用して、グラフィカル表現を表示できます。

グラフィカル表現は、プレーンテキストプランよりも詳細を提供し、複雑なクエリの読みやすさとプレゼンテーションを向上させます。EXPLAIN LOCAL VERBOSE オプションはいくつかのグラフィカルなプランを生成します。以下はそれぞれ個別のグラフィカルなプランとなります。

- **単純化された結合順序**:単純化されたプランには、クエリを理解するために必要な最小の結合情報のみが含まれます。このプランを表示すると、複雑なステートメントを理解するのに便利です。次のセクションでは、簡単な結合順序の例を示します。

- **JOIN グラフ**:プランは、クエリプランの結合の説明で構成されます。

- **各ノードの計画**

    - イニシエーターノード

    - 実行ノード。複数のノードがある場合、ノードごとに 1 つずつ、複数のプランを作成します。

- **各ノードのリソース情報**:各ノード上でプランを実行するためのリソースに関する詳細情報。オプティマイザはリソースを計算し、実行エンジンはリソースを割り当てます。次のセクションでは、特定のノードのリソース情報を示す例を示します。

EXPLAIN ステートメントで以下のクエリを検討してみましょう。

```
EXPLAIN LOCAL VERBOSE SELECT DISTINCT s.product_key, p.product_description
FROM store.store_sales_fact s, public.product_dimension p
WHERE s.product_key = p.product_key
AND s.product_version = p.product_version
AND s.store_key IN (
SELECT store_key
FROM store.store_dimension
WHERE store_state = 'MA')
ORDER BY s.product_key;
```

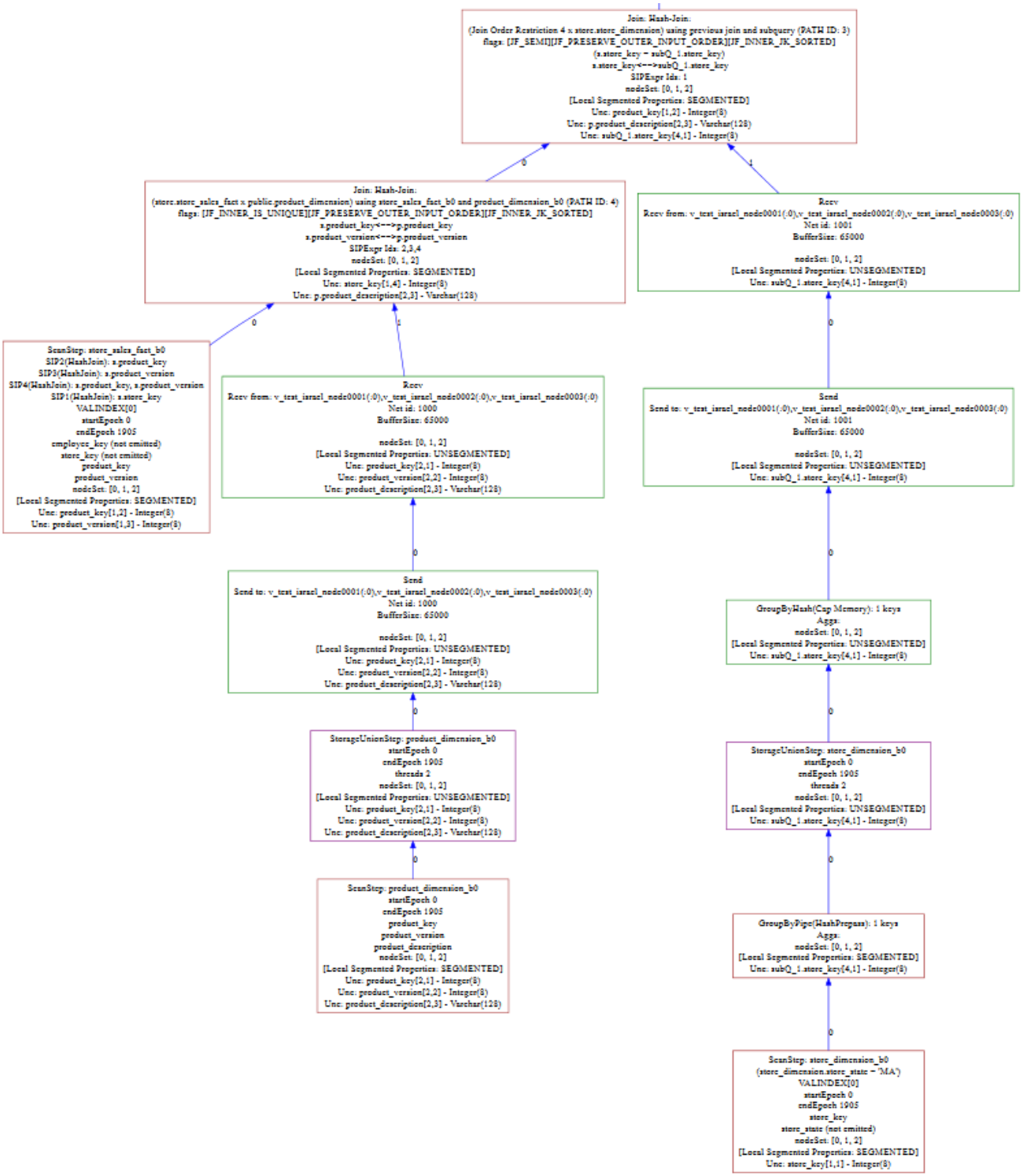
EXPLAIN LOCAL VERBOSE のテキスト出力

EXPLAIN LOCAL VERBOSE は、EXPLAIN のクエリプランと同じ情報を持つクエリプランを返します。これには、コストや異なるリソースの重みなどの追加の詳細が含まれます。次のサンプルでは、**太字**のテキストはリソースごとの追加コスト、ネットワーク、およびメモリ使用量を示しています。

```
Access Path:
 Sort Key: (store_sales_fact.product_key,
product_dimension.product_description)
  LDISTRIB_UNSEGMENTED
 +-GROUPBY HASH (SORT OUTPUT) (GLOBAL RESEGMENT GROUPS) (LOCAL RESEGMENT
GROUPS) [Cost: 2662115.000000, Rows: 15000000.000000 Disk(B):
2400000000.000000 CPU(B): 2280000000.000000 Memory(B): 4560000000.000000
Netwrk(B): 4440000000.000000 Parallelism: 3.000000] [OutRowSz (B): 144] (PATH
ID: 2)
 |  Group By: s.product_key, p.product_description
 |  Execute on: All Nodes
 |  Sort Key: (store_sales_fact.product_key,
product_dimension.product_description)
 |  LDISTRIB_SEGMENTED
 | +---> JOIN HASH [Semi] [Cost: 1387361.000000, Rows: 15000000.000000
Disk(B): 5436997632.000000 CPU(B): 4800001024.000000 Memory(B): 2304.000000
Netwrk(B): 768.000000 Parallelism: 3.000000] [OutRowSz (B): 152] (PATH ID: 3)
Inner (BROADCAST)
```

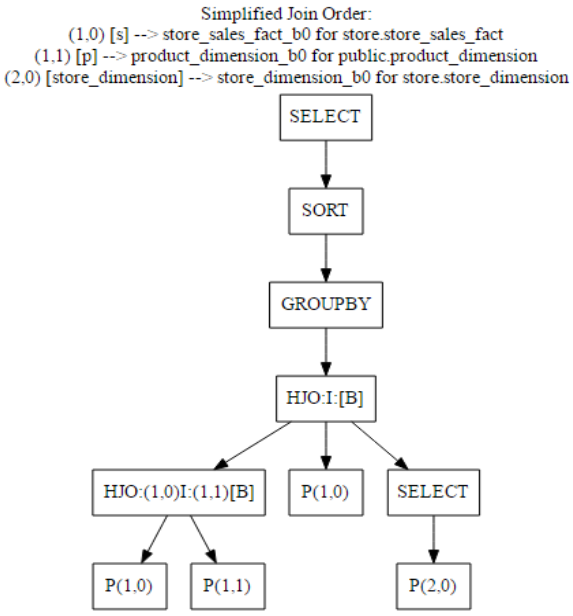
EXPLAIN LOCAL VERBOSE のグラフィカル出力

次の図は、グラフビジュアライゼーションツールである Graphviz が、このドキュメントの前半のサンプルクエリのグラフをどのように表示するかを示しています。各ボックスには、EXPLAIN ステートメントのグラフィカル出力よりも詳細な情報が表示されます。次のグラフは、完全なグラフの一部です。



EXPLAIN LOCAL VERBOSE の単純化された結合順序

Simplified Join Order（単純化された結合順序）ボックスは、オペレータ（SORT、JOIN、MERGE、および GROUPBY）に関する情報を読みやすい形式で提供します。いくつかの結合を含む複雑なクエリでは、このプランはクエリの実行を理解するのに役立ちます。



EXPLAIN LOCAL VERBOSE のリソース情報

EXPLAIN LOCAL VERBOSE オプションは、各ノードに関するリソース情報を返します。クエリプランは推定リソースを生成します。次の例は、イニシエーターノードがクエリを実行するために必要とする推定リソースを示しています。

```
Estimated resources for plan:
 ----------------------------
 Scratch Memory MB: 1474
 File Handles:      64
 Worker Threads:    38
 Blocking Threads:  22
 Externalizing Ops: 4
 Unbounded Mem Ops: 2
 Max Threads:       2
```

詳細情報

詳細情報	…See
Vertica Community Edition	https://my.vertica.com/community/
Vertica Documentation	http://my.vertica.com/docs/latest/HTML/index.htm
Big Data and Analytics Community	https://forum.vertica.com/