

Technical Brief

HP / IBM Red Brick

Proof of Performance and Scalability(POPS)

Benchmark Study

August 2002

目次

はじめに	3
ハードウェア構成の概要	4
システム	4
格納域 (物理的 I/O コンフィグレーション)	4
格納域 (論理ボリューム)	4
データベース構成	5
スキーマのレイアウト	5
セグメント化 (データパーティショニング)	5
インデックスの定義	6
ベンチマークの方法	7
データベースのスケラビリティ	7
マルチユーザのスケラビリティ	7
データベースローディング	7
DAILY_SALES 表へのロード	8
DAILY_FORECAST 表へのロード	8
TMU パラメータの設定	8
ベンチマーク結果	9
ロード結果	9
ストレージ要件	エラー! ブックマークが定義されていません。
ディメンジョン・テーブル	11
ファクト・テーブル 2TB	11
ファクト・テーブル 4TB	11
ファクト・テーブル 8TB	11
クエリ結果	12
スループットのスケラビリティ	12
クエリ・CPU および応答時間	13
マルチユーザの	17
マルチユーザ用のチューニング・パラメータ	19
結論	20
付録 A	21
ハードウェアの詳細説明	21
OS パラメータの設定	22
ハードウェアデザイン	23
付録 B クエリ	24

はじめに

本書は、Hewlett-Packard (HP) と IBM による合同調査の結果を要約したものです。この調査は、HP GS320 システム上で動作する IBM Red Brick Warehouse の性能と拡張性を実証するものです。調査では、1000 人を上限とする様々なユーザ数で、2 TB (テラバイト)、4 TB、8 TB のデータ・ウェアハウス・サイズを測定しました。この調査の目的は、HP GS320 が大容量であることと、IBM Red Brick Warehouse がデータ・ロード性能、クエリ性能、および拡張性に優れていることを実証することです。

この調査には、Red Brick 性能および拡張性の実証 (POPS) のベンチマークを採用しました。このベンチマークが採用された理由は、このベンチマークが Red Brick ユーザ向けの一般的な小売業務の作業として代表的なものであるためです。このベンチマークは広く認識されており、社内外の多数の調査に用いられています。このベンチマークの信頼性と反復性については、両社とも強く確信しています。実際の測定には、様々なユーザ数およびデータ・ウェアハウス・サイズでのベンチマークのチューニング、データのロード、クエリの実行を含め、約 5 週間を要しました。

このレポートには、ハードウェア構成、ベンチマーク・スキーマおよびデータベース・レイアウト、ベンチマークの方法論、データベース・ロード・テストおよびマルチユーザ・テストから得られた拡張性結果に関する情報を示します。また、付録には、詳細なハードウェア構成、オペレーティング・システムのチューニング・パラメータ、各クエリに対応する SQL のリストを示します。

ハードウェア構成の概要

システム

AlphaServer GS320

CPU 数:32 (1GHz)

メモリ:128 GB

オペレーティング・システムリビジョン:Tru64 V5.1A (Rev 1885)

格納域 (物理 I/O コンフィグレーション)

ファイバ・チャンネル・スイッチ数:8 スイッチあたりの FC ポート数:16

HSG80 StorageWORKS ファイバ・チャンネル・コントローラ数:24

HSG80 あたりの持続書き込みレート:80 MB/s

システムの持続 I/O レート合計:1.9 GB/s

HSG80 あたりのコントローラ数:2

各コントローラ:256MB ミラーリング・キャッシュ装備

コントローラ・ペアあたりのディスク数:42

EMA12000 StorageWorks RAID システム

物理ディスク合計:1008

18GB ディスク数:504 (15,000 rpm)

36GB ディスク数:504 (10,000 rpm)

全ディスクの構成:RAID5 セット、ストライプ・サイズ 128KB

RAID5 セットあたりのディスク数:6

18GB ディスクの RAID5 セット数:72

36GB ディスクの RAID5 セット数:72

格納域 (論理ボリューム)

論理ボリューム:1 TB (72 の RAID5 セット間でストライピング)

論理ボリュームのストライプ・サイズ:768KB

論理ボリューム内の追加ソフトウェア冗長性 (RAID0):なし

18GB ディスクを使用してファイル・システム /k15 としてマウントされた論理ボリューム数:6

36GB ディスクを使用してファイル・システム /k10 としてマウントされた論理ボリューム数:9

(ハードウェア構成の詳細と I/O 構成のレイアウト図については、付録 A を参照)

データベース構成

スキーマのレイアウト

5つのディメンジョンと2つのファクト・テーブルで構成される、標準的な小売スキーマを使用します。

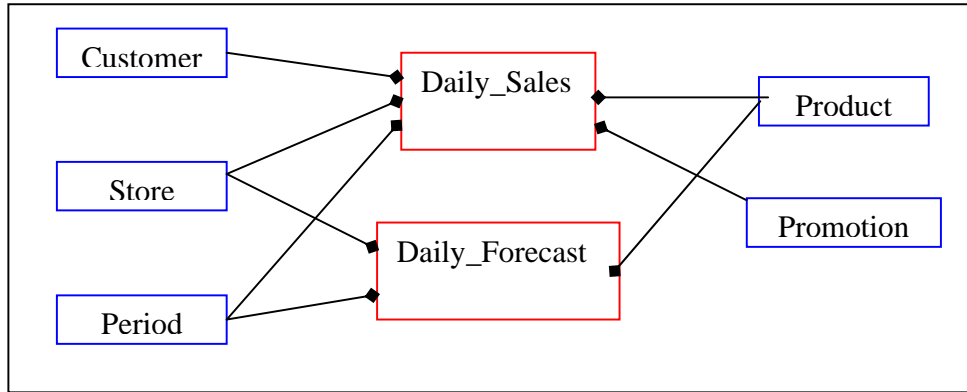


図1 POPS スキーマ

このスキーマは、主要なビジネス・ディメンジョンに関する販売業務情報を表しています。DAILY_SALES テーブルには、販売期間 / 日、販売した製品、販売した店舗、購入した顧客、販売促進活動 / セール期間が使用されたかどうかを示す販売記録が格納されます。DAILY_FORECAST テーブルには、店舗別、製品別、期間別の販売予測が格納されます。このデータベースは、一般的なユーザの利用状況をエミュレートするように編成されています。データは期間別にパーティション分割され、日々の販売記録が蓄積されるか、あるいは将来の予測が示されます。

セグメント化 (データパーティショニング)

ディメンジョン・テーブル:

ディメンジョン・テーブルはセグメント化されません。

CUSTOMER テーブルと PRODUCT テーブルにはそれぞれ、15 の物理格納ユニット (PSU) があります。(PSU はセグメント内の格納域のサブユニットであり、ディスク上のファイルに相当します。)

ファクト・テーブル:

DAILY_SALES テーブルとインデックスは、1 週間単位でセグメント化されます。DAILY_FORECAST テーブルとインデックスは、10 週間の増分でセグメント化されます。次の表に、このベンチマークのセグメント化スキームをまとめます。ファクト・テーブルはファイル・システム /k10 に格納され、すべてのインデックスとディメンジョン・テーブルは /k15 に格納されます。連続ディスク・アクセス時の最適化のため、セグメントはシーケンス順に作成され、事前に初期化されています。

DAILY_SALES	セグメント数	セグメントあたりの PSU 数	テーブルの PSU サイズ	インデックスの PSU サイズ
2TB	105	32	1.2GB	68.5MB
4TB	209	32	1.2GB	68.5MB
8TB	209	64	1.2GB	68.5MB

DAILY_SALES	セグメント数	セグメントあたりの PSU 数	テーブルの PSU サイズ	インデックスの PSU サイズ
2TB	11	32	205MB	56.3MB
4TB	21	32	205MB	56.3MB
8TB	21	64	205MB	56.3MB

インデックスの定義

次の表に、ベンチマーク用に作成されたインデックスを示します。DAILY_SALES テーブルには 2 つの Star インデックスが含まれ、どちらも全部で 5 つの外部キー制約を扱っています。DAILY-FORECAST テーブルには 1 つの Star インデックスが含まれ、すべての外部キーを扱っています。すべてのディメンジョン・テーブルには主キー・インデックスが含まれています。追加のディメンジョン・インデックスは、制約の評価に有用となる列に対して作成されます。

テーブル名	インデックスのタイプ	インデックス名	インデックス付けされる列
Customer	B-Tree Target Target Target	CUSTOMER_PK_IDX CUST_TGTIDX1 CUST_TGTIDX2 CUST_TGTIDX3	Custkey Gender Age_Level_Desc Income_Level_Desc
Period	B-Tree B-Tree Target	PERIOD_PK_IDX PER_CAL_DATE_IDX PER_WEEK_ENDING_TGT	Perkey Calendar_Date Week_Ending_Date
Product	B-Tree B-Tree Target Target	PRODUCT_PK_IDX PROD_ITEM_DESC_IDX PROD_CATEGORARY_TGT PROD_SUB_CATEGORY_TGT	Prodkey Item_Desc Category Sub_Category
Promotion	B-Tree B-Tree	PROMOTION_PK_IDX PROMO_DESC_IDX	Promokey Promodesc
Store	B-Tree B-Tree	STORE_PK_IDX STORE_STORE_NUM_IDX	Storekey Store_Number
Daily_Sales	Star	DS_STAR1	Perkey Storekey Prodkey Promokey Custkey
Daily_Sales	Star	DS_STAR2	Perkey Prodkey Storekey Promokey Custkey
Daily_Forecast	Star	STAR1_DF	Perkey Storekey Prodkey

ベンチマークの方法

データベースのスケラビリティ

この調査の主な目的は、HP GS320 システムと IBM Red Brick Warehouse の両方の性能と拡張性を実証することです。この調査では、2TB、4TB、8TB の 3 つの大容量データベース・サイズを選択しました。また、これらのデータベース・サイズごとに、100 人、200 人、500 人、1,000 人と様々なユーザ数を設定しました。このように複数の測定ポイントを選択している目的は、GS320 システムの処理と I/O 容量、および大容量データベース上で大勢のユーザをサポートする際の IBM Red Brick Warehouse の性能を実証するためです。

この調査では、次の 2 つの方法でデータベースを拡張しています。

1. 日数を増やす
2. 1 日あたりの行数を増やす

2TB のデータベースには、2 年分以上のデータ (733 日) を格納します。4TB のデータベースには (1) の方法を適用し、さらに 2 年分のデータを追加します。8TB のデータベースの場合は、年数を一定にして (2) の方法を適用し、1 日あたりの行数を 2 倍にします。

(1) の方法による拡張は、一般的なユーザの利用状況をエミュレートしたもので、販売データが期間ごとと数年にわたって蓄積されます。この場合、ユーザは長期にわたる傾向分析を行うことができます。(2) の方法による拡張は、より高い処理需要をサポートする場合の効率を実証するものです。ユーザが扱う販売量が多い場合や、販売量が増えている場合には、同じ期間でより多くの行をどれだけ効率良く処理できるかを調べることができます。

マルチユーザのスケラビリティ

この POPS ベンチマーク (リビジョン 2) では 18 件のクエリを扱います。各ユーザが 18 件すべてのクエリを実行します。ベンチマークに負荷条件を取り入れるため、同じ選択性は維持しながら、マルチユーザ・テスト用にクエリ内の制約を様々に変化させます。これにより、I/O キャッシュの確率が下がり、ディスクへのランダム I/O アクセス数が増えます。クエリ制約の対象期間は 2 年とします。4TB と 8TB のテストでは、後の 2 年も対象とするようこれらの制約を変更しており、最新情報ほど頻繁にアクセスされる実際の状況により近づけています。

マルチユーザ・テストは、ユーザ数の増分 n (100 人、200 人、500 人、1000 人) を繰り返して実施されます。テスト・スクリプトは、10 人で構成されるユーザ・グループごとにバージョンを変更します。各バージョンには、異なるクエリ制約セットが含まれています。各ユーザ・テストは、ランダム性を高め、組み合わせたクエリ制約の数を増やして、再度、堅牢性負荷テストを実施する方法がとられています。

データベースローディング

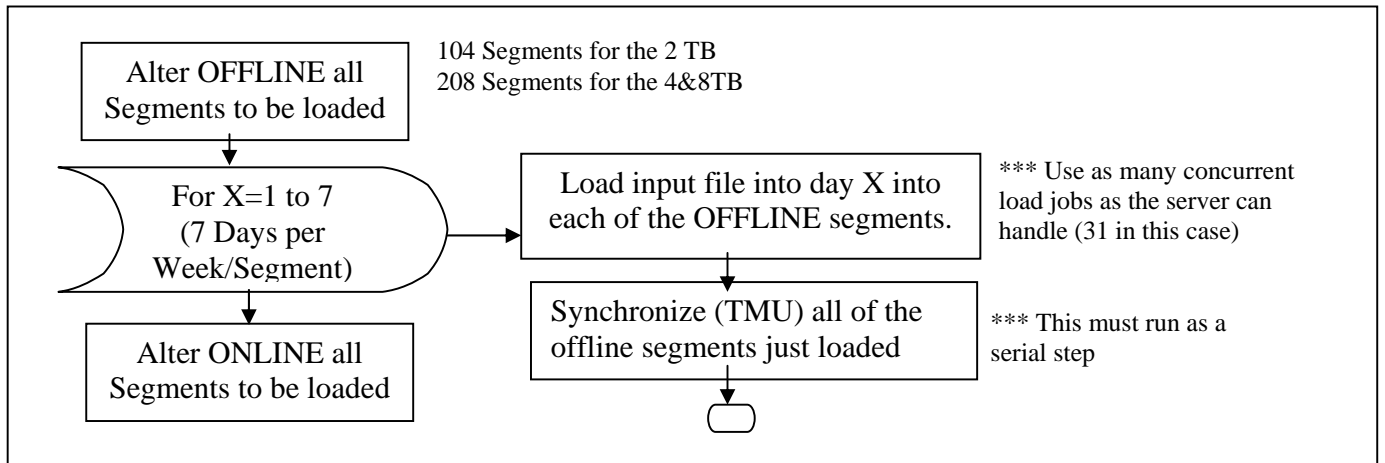
データベースのロードには、事前に生成された入力データを使用します。ファクト・テーブル用の入力データは、1 日単位で生成されています。日付はロード中に変更することができ、必要なデータベース・サイズまで拡張するために n 日間繰り返すことができます。DAILY_SALES 入力データには 11,669,320 行が含まれ、DAILY_FORECAST 入力データには 1,223,475 行が含まれています。ロード速度を最適にするため、入力データはインデックス列順に格納されます。DAILY_SALES の場合、DS_STAR1 インデックスに基づいて格納されます。

2 TB と 4 TB の場合は、入力データ・ファイルが 1 日ごとに 1 回ロードされます。8 TB の場合は、入力データ・ファイルが 1 日ごとに 2 回ロードされます (つまり、1 日あたりの行数を 2 倍にします)。

DAILY_SALES 表へのロード

DAILY_SALES テーブルの最初のデータ格納については、GS320 の容量と IBM Red Brick Warehouse の並列処理を活かして一括ロード方式を採用しています。ここでは、GS320 に装備されている 32 個の CPU と高い I/O 帯域幅を活かして、31 のオフライン・セグメントをロードしました。ここでの目的は、GS320 CPU を 100% ビジー状態に近づけることです。すべてのセグメントがオフライン・モードでロードされると、それらのセグメントが互いに同期化され、クエリ・レディ状態でオフラインに戻されます。

次の図に、オフラインでのセグメント・ロードに関連する動作の流れを示します。



DAILY_FORECAST 表へのロード

DAILY_FORECAST テーブルは、比較的小さなファクト・テーブルです。このテーブルのロードには、シリアル・ローダ (rb_tmu) を使用して小さいリソース (1 つの CPU) を割り当てます。このリソースには、平行して DAILY_SALES テーブルがロードされます。DAILY_FORECAST テーブルはサイズが小さいため先にロードが完了するので、比較的大きい DAILY_SALES テーブルによって決まる全体的なロード時間には影響しません。これについては、「ロード結果」の節を参照してください。

TMU パラメータの設定

```
TMU_BUFFERS 256
TMU_MMAP_LIMIT 12M
INDEX_TEMPSPACE_THRESHOLD 300M
OPTIMIZE ON
INDEX TEMPSPACE DIRECTORY /K15 and /K10
Load in append mode
```

ベンチマーク結果

ロード結果

IBM Red Brick Warehouse には、パス・ローダが 1 つ備わっています。ローダが入力データを読み取ると、データが内部格納形式に変換され、参照整合性チェック (RI) が実行され、インデックスが構築され、行のレコードとインデックスのノードがディスク上の PSU に書き込まれます。ロードが完了すると、Red Brick ユーザはデータベースがクエリ・レディ状態になるのを期待します。必要なステップは以上です。

POPS ベンチマークからのロード結果を以下にまとめます。報告された時間には、すべてのオフライン・セグメントをまとめて同期化するための時間と、それらのセグメントをクエリ・レディ状態でオンラインに戻すための時間が含まれます。

データ・ポイント	テーブル名	GB/時間	ロードされた行数	経過時間	行数 / 分
2TB	Daily_Sales	200.0	8,483M	10:17:38	13,735,651
4TB	Daily_Sales	229.8	17,060M	17:59:57	15,797,533
8TB	Daily_Sales	279.0	34,121M	29:39:27	19,172,922
2TB	Daily_Forecast	12.0	896M	2:56:54	5,069,571
4TB	Daily_Forecast	24.5	1,791M	2:52:34 ¹	10,369,394
8TB	Daily_Forecast	9.1	3,574M	15:23:38	3,870,577

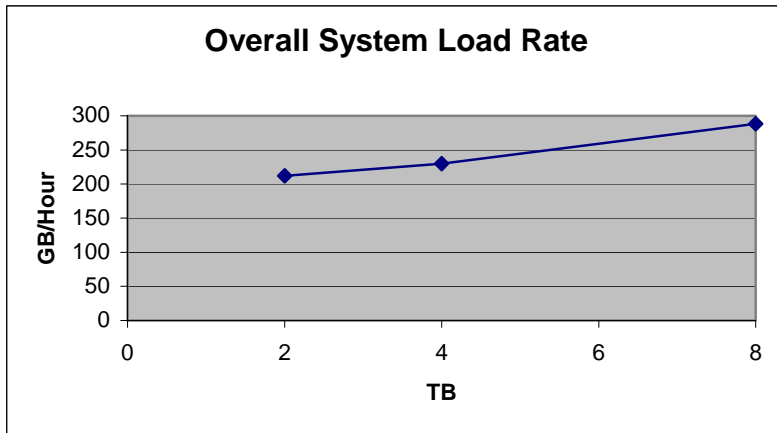
データベース・サイズを拡張したときでも、ロード・レートは増えています。DAILY_SALES テーブルのロードには、システム・リソース (31 のオフライン・セグメントの平行ロード) の大部分が提供されています。この結果から分かるように、報告されたロード・レートは非常に高く、4TB から 8TB に拡張したときに向上しています。この向上は、I/O キャッシュの利点と、Red Brick のローダが重複する入力行を利用したことを反映しています。

DAILY_FORECAST の場合、ロード・レートは期待されたほど高くはありませんでした。このロード動作には実質的に 1 つの CPU しか割り振らなかったため、DAILY_SALES への平行ロードと競合せざるを得ませんでした。

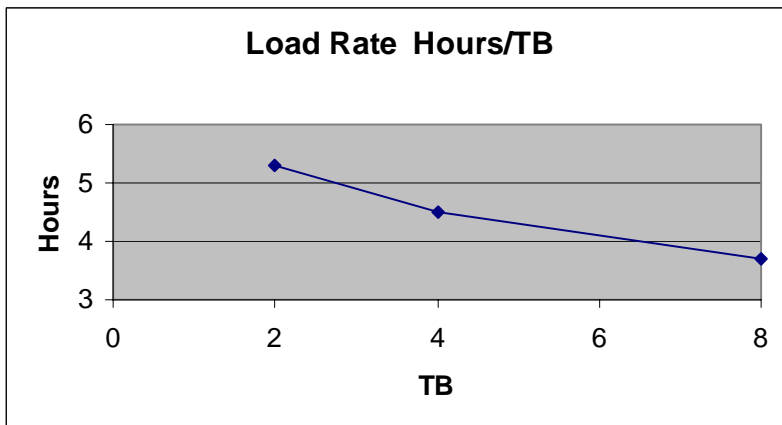
システム全体のロード・レートは、合計 2 つのファクト・テーブルの平行ロードを表しています。合計レートは、2TB のときには 212GB/時間で、8TB のときには 288.1GB まで増えています。ロードされた行数は、90 億行以上から 370 億行以上に増えています。4TB のロードに関しては、結果は切り離して考慮されます。この場合は、2 つのファクト・テーブルが独立してロードされています。リソースに対する競合がなく、並列ローダを使用したこのロードでは、DAILY_FORECAST の場合は 24 GB/時間のレート、DAILY-SALES の場合は 229GB/時間のレートが報告されています。

1. 平行して Daily_Sales テーブルをロードせずに、並列ローダを使用して Daily_Forecast のロードを自動的に実行したときの測定時間です。

次のチャートに、2TB、4TB、8TB に対するシステム全体のロード・レートを表します。これは、DAILY_SALES テーブルと DAILY_FORECAST テーブルが平行してロードされたときの、2つのテーブルのロード・レートの合計を表しています。4TB の場合は、DAILY_SALES のロード・レートしか使用されていないため、もう一方のファクト・テーブルとの平行ロードはありません。



これらの結果に関するもう1つの観点は、1TB のデータをロードしてクエリ・レディ状態にするまでの所要時間です。次のチャートに、GS320 上での IBM Red Brick Warehouse の拡張状況を示します。1TB をロードするのに必要な時間は、データベースが拡張されるに従って短くなります。実際には、2TB の 5.3 時間から 8TB の 3.7 時間まで短縮されています。



ロード時には、システムを 100% ビジー状態にすることができます。これが可能な要因は、大容量のディスク構成と、GS320 上で設定された高い I/O 帯域幅です。すべてのディスクが RAID5 用に構成されている場合でも、システムは非常に高い I/O レートを維持することが可能です。この調査では、システムに 32 個の CPU が装備されていることを活かして、実質的に 32 件ある平行ロード動作にすべての CPU を利用しています。IBM Red Brick Warehouse では、システムの大容量を活かして、非常に高いロード・レートと拡張性が実証されています。

ストレージ要件

**ディメンジョン・テーブル
(ベンチマーク全体を通して一定)**

テーブル名	行数	データ格納	インデックス格納
Customer	1,000,000	163,280K	11,872K
Store	64	16K	40K
Promotion	64	16K	16K
Product	20,000	4,592K	1,144K
Period	3,653	96K	136K
合計		168,000K	13,208K

ファクト・テーブル 2TB

テーブル名	行数	データ格納	インデックス格納
Daily_Sales	8,483,595,640	2,056,630,672K	233,238,288K
Daily_Forecast	896,807,175	35,169,024K	9,673,960K
合計		2,091,799,696K	242,912,248K

Red Brick が 2TB のロウ・データを格納するための格納領域の合計 = 2,334,893,152K
 ロウ・データからデータベース格納領域への比率 = 1:1.087

ファクト・テーブル 4TB

テーブル名	行数	データ格納	インデックス格納
Daily_Sales	17,060,545,840	4,135,892,744K	469,043,056K
Daily_Forecast	1,791,167,400	70,242,064K	19,321,472K
合計		4,206,134,808K	488,364,528K

Red Brick が 4TB のロウ・データを格納するための格納領域の合計 = 4,694,680,544K
 比率 = 1:1.093

ファクト・テーブル 8TB

テーブル名	行数	データ格納	インデックス格納
Daily_Sales	34,121,091,680	8,271,782,144K	938,052,656K
Daily_Forecast	3,574,993,950	140,196,072K	38,551,552K
合計		8,411,978,216K	976,604,208K

Red Brick が 8TB のロウ・データを格納するための格納領域の合計 = 9,388,763,632K
 比率 = 1:1.093

* daily_sales テーブルの行幅 = 236 バイト
 daily_forecast テーブルの行幅 = 36 バイト

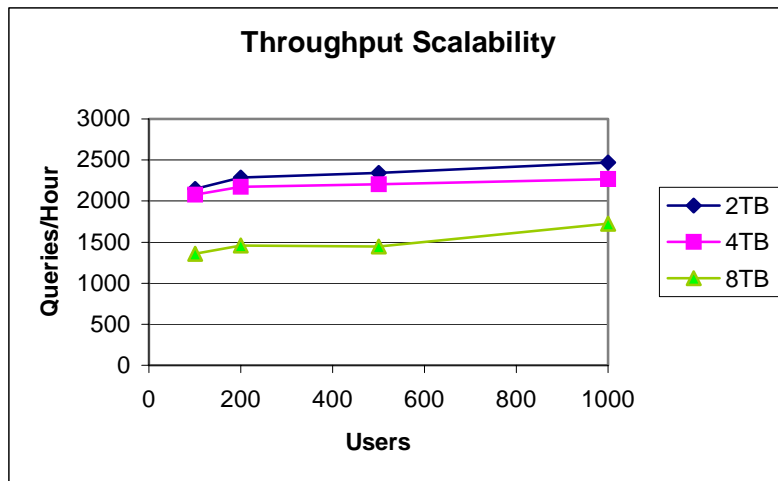
クエリ結果

直線的な拡張では、期待どおりのクエリ結果が出ました。クエリ結果では、ユーザ数を増やしても高いスループットが維持され、処理する行数を増やしたときのクエリ応答時間における直線的増加よりも良い結果を示しました。このように顕著な拡張性結果は、IBM Red Brick Warehouse と GS320 の両方が大きな負荷に十分に耐えることを示しています。IBM Red Brick Warehouse では、1,000 人もユーザで同じ CPU 時間を費やした場合でも、一貫してクエリを実行することが可能です。GS320 は、大勢のユーザと高い I/O 要求レートの条件下でも十分に対応することができます。

スループットのスケラビリティ

この調査では、2TB、4TB、8TB のデータベースのクエリ性能を、それぞれ 100 人、200 人、500 人、1,000 人のユーザ数で測定しました。平均的なクエリ CPU 時間はユーザ数に関係なく一定に維持され、同時に選択性 (つまりファクト・テーブルから選択される行数) も一定に維持されています。したがって、2TB と 4TB のクエリ CPU 時間は、一貫して似通っています (この場合、時間は 2 倍になります)。4TB と 8TB とでは、クエリ CPU 時間が 2 倍になるものと期待されましたが (この場合、各時間の行数は 2 倍になります)、場合によっては、CPU 時間の増加が 2 倍未満になることがありました。IBM Red Brick Warehouse は重複する行を識別して活用することができるため、クエリ処理時には IBM Red Brick Warehouse が最適化されます。

次のチャートに、様々な負荷条件でのスループットの拡張性を示します。このチャートは、IBM Red Brick Warehouse と HP Tru64 の両方が、負荷の多い要求をいかに効率良く処理するかと、100 人から 1,000 人のユーザ数でいかに高いスループットを維持するかを表しています。2 TB と 4 TB のスループット (クエリ数 / 時間) は、期待どおり非常に似通っています。8TB のスループットは期待どおりに拡張されています (この場合、クエリ 1 ~ 10 は 2 倍の行数を処理する必要があります)。



クエリ CPU および応答時間

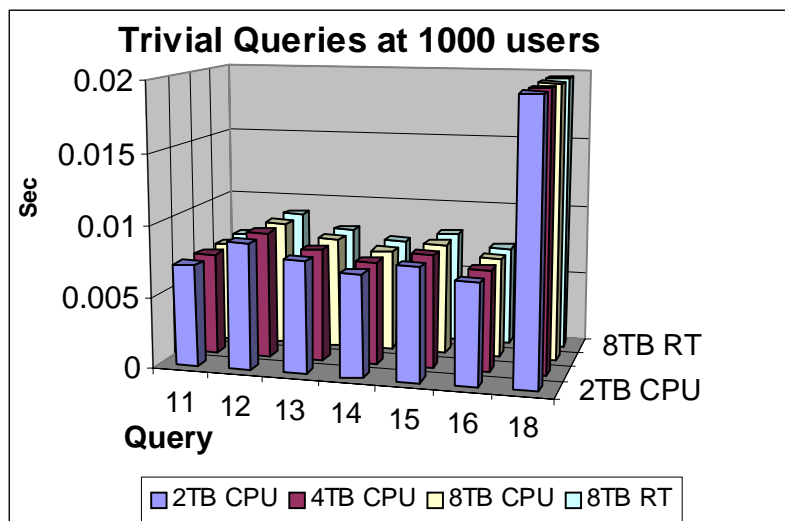
次のチャートに、データベース・サイズを拡張したときの、個々のクエリ性能の結果を示します。負荷が大きくても、クエリ CPU 時間はかなり短い状態で維持されることを示すために、ここではユーザ数を 1,000 人にしたときの結果を採用しています。クエリを完了するのに必要な平均 CPU 時間によって、クエリは次の 4 つのカテゴリに分類されます。

1. 1 秒未満、Trivial
2. 10 秒未満、Small
3. 1 分未満、Medium
4. 2 分未満、Large

クエリは、IBM Red Brick Warehouse の CPU 実行時間に基づいて分類されます。Q1 ~ Q10 は、実際には集約と整列を使用した n 方向のテーブル結合クエリです。Q9 と Q10 は、ファクト・テーブル対ファクト・テーブルの結合を実行します。Q11 ~ Q18 は、100 万行が含まれる Customer デイメンジョン・テーブルに対するクエリです。これらのクエリはほとんどが Trivial に分類されますが、13,000 以上のリスト内項目を含む Q17 だけは例外です。詳細なクエリ・リストについては、付録 B を参照してください。

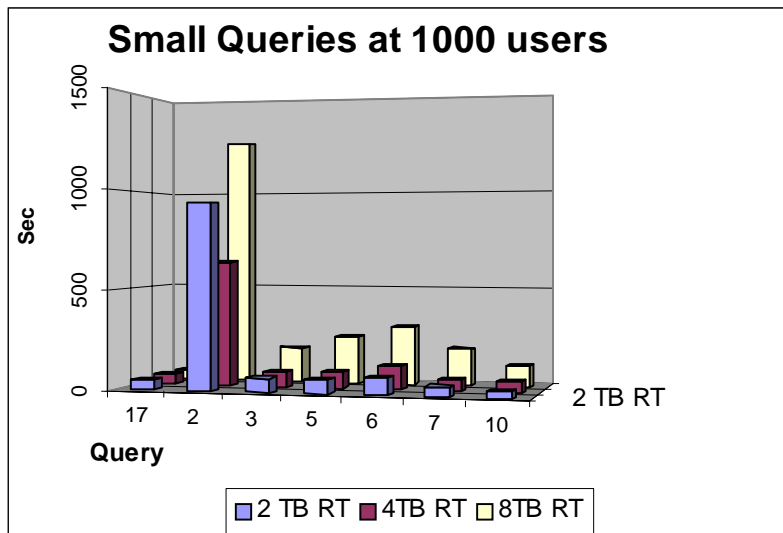
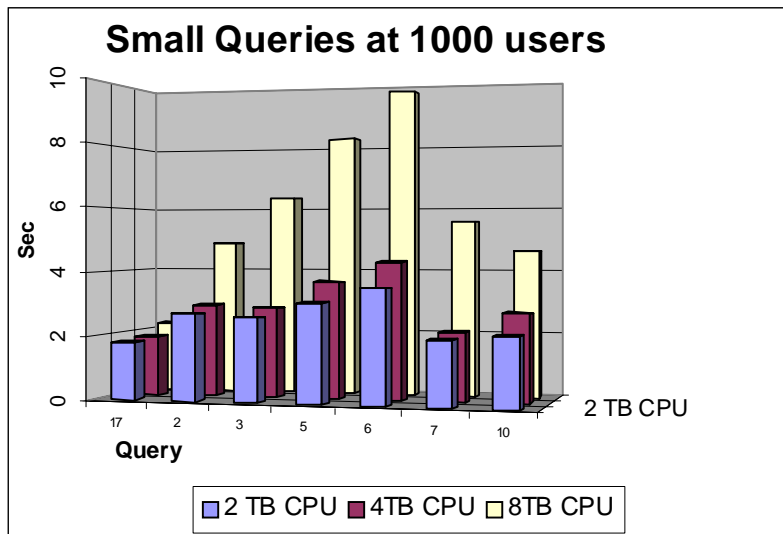
(1) Trivial クエリ

次のチャートは、2TB から 4TB、8TB へと拡張していく Trivial クエリの結果を示しています。これらのクエリは、ベンチマーク全体を通じて変化がなかった Customer デイメンジョン・テーブル上にもみ存在します。CPU とクエリ応答時間 (RT) のどちらも、データベース・サイズに関係なく変化していないことに注意してください。

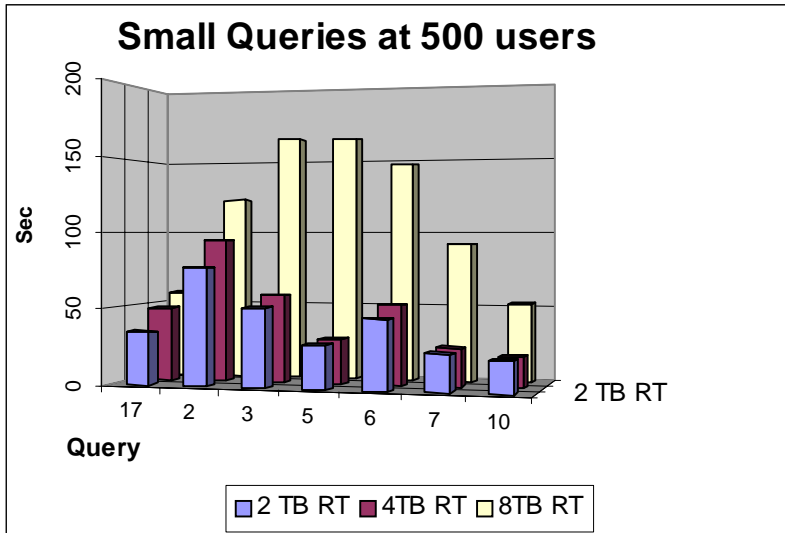


(2) Small クエリ

次のチャートは、クエリ CPU 時間がほぼ期待どおりだったことを示しています。クエリ CPU 時間は、4TB から 8TB の間でのみ増えています (この場合、処理する行数は 2 倍になります)。クエリ応答時間は妥当な動きを示しており、行数に関連してほぼ直線的な増加を示しています。クエリ CPU 時間とクエリ応答時間 (RT) には大きな違いがあることに注意してください。これらの結果は、ユーザ数を 1,000 人としたものです。応答時間は、高いシステム負荷とリソースの競合を反映しています。大きな競合が発生している間は、クエリがキューに入れられるか、またはスワップ・アウトされ (あるいはその両方が実行され)、リソースが処理を完了するまで待機します。

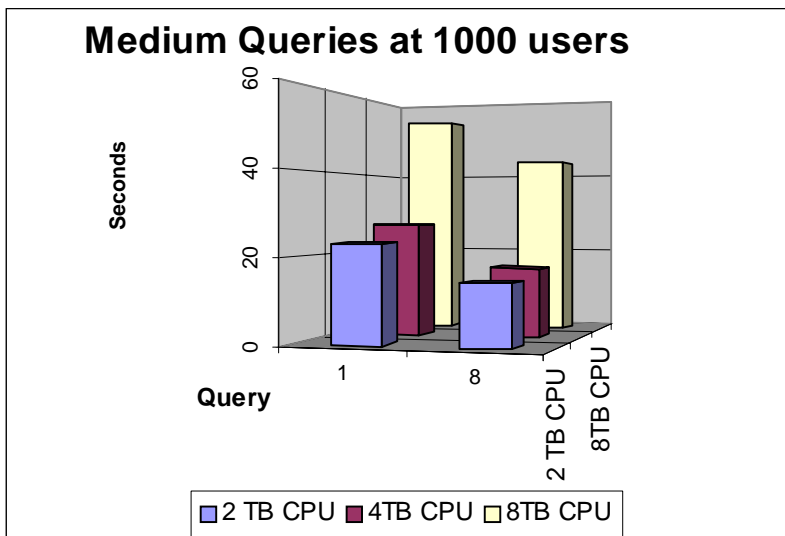


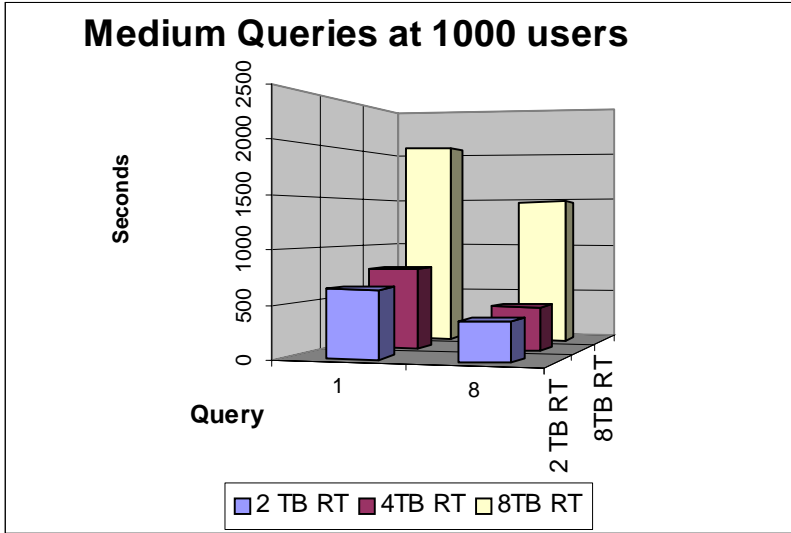
Q2 の応答時間が、他に比べて不規則な動きを示していることに注意してください。この結果は、ユーザ数 1,000 人のテストにのみ現れたものです。次のチャートは、ユーザ数を 500 人に設定したときの応答時間の結果を示しています。ここでは、Q2 の結果が期待どおりの動きを示しています。不規則な応答時間はおそらくシステム負荷によるもので、このクエリが比較的長時間スワップ・アウトされていたものと考えられます。



(3) Medium クエリ

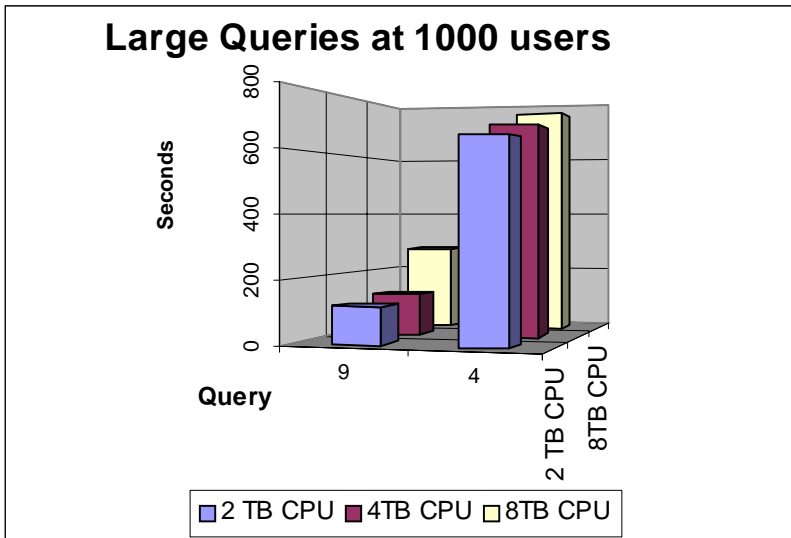
このグループのクエリは、一貫した CPU 時間と応答時間の動きを示しています。CPU 時間は、2TB から 4TB の間は比較的一定状態を保っていますが、4TB から 8TB の間は直線的に増加しています。

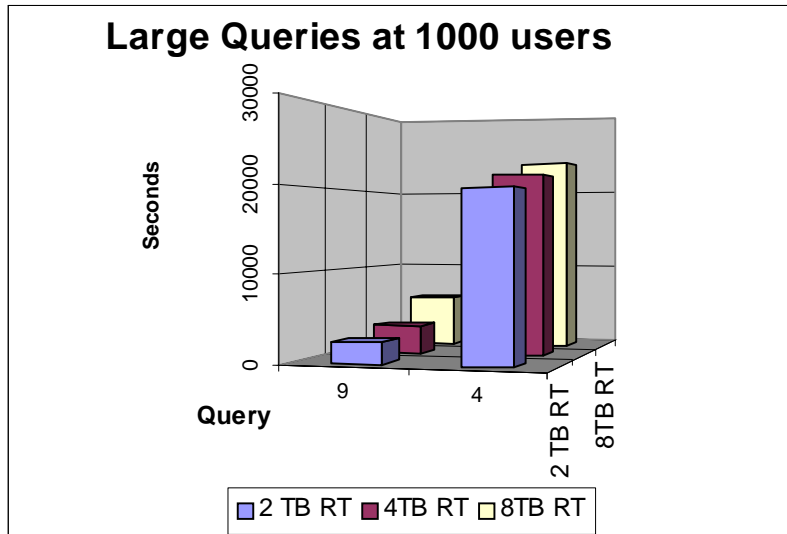




(4) Large クエリ

Q4 では、CPU 時間と応答時間の両方の増加率が、期待したよりも少ないことに注意してください。この場合は、8TB の結果が 4TB に近くなっています。



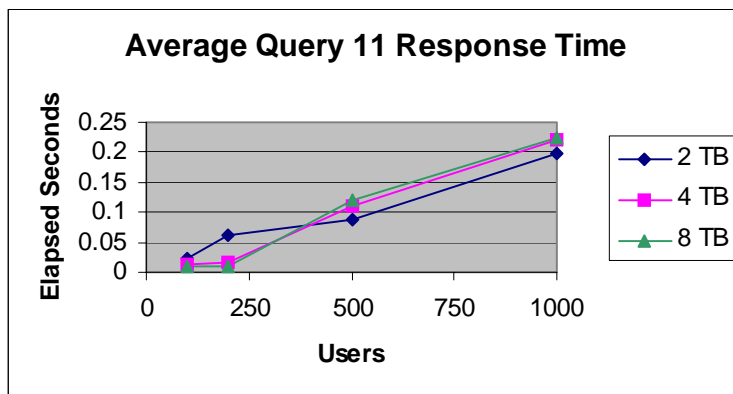


マルチユーザの拡張性

マルチユーザの結果は、非常に高い拡張性を示しています。クエリ CPU 時間は、すべてのユーザ間で比較的一定状態を保っています。クエリ応答時間については、期待どおりではあるものの、比較的興味深い結果になっています。次に、4 つのカテゴリ別に、マルチユーザの拡張性結果のサンプルを示します。

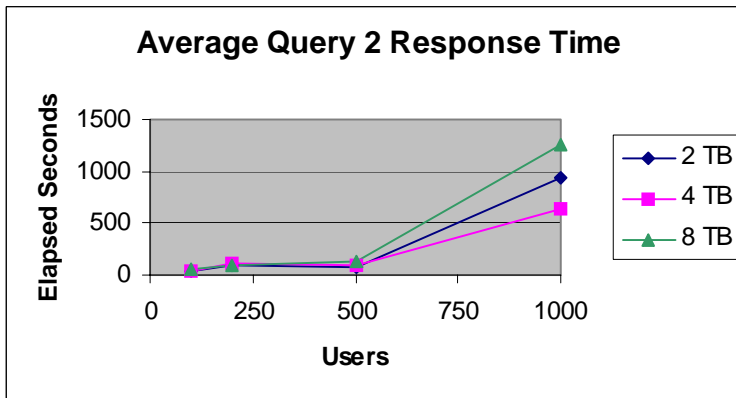
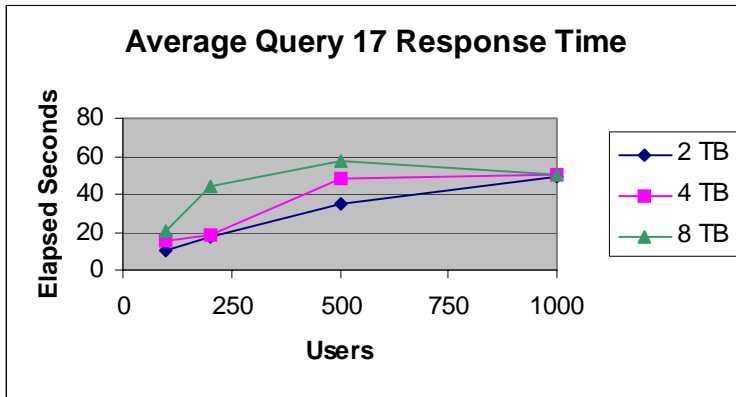
(1) Trivial クエリ

Q11 は、Trivial クエリのグループを表しています。データベース・サイズに関係なく、各ユーザ増分に対する応答時間は非常に似通っていて、ユーザの増分とともに直線的に増加しています。



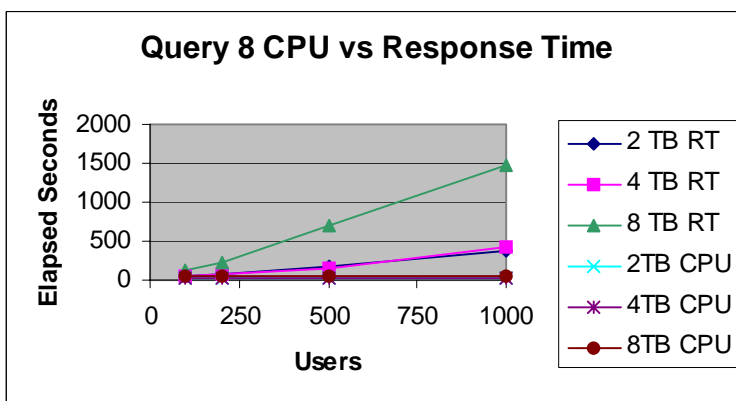
(2) Small クエリ

Small グループ・カテゴリの場合は、クエリ応答について非常に興味深い結果が出ています。Q17 では、応答時間がユーザの増分とともに増加していますが、どのケースも直線的にはなっていません。8TB には、潜在的に I/O キャッシュの利点があります (この場合、応答時間はユーザ数 1000 人で収束しています)。対照的に、8TB の Q2 の場合は、キャッシュ処理がほとんど、あるいは全くないものと予想されます。どのデータベース・サイズでも、応答時間はユーザ数 1,000 人で急激に増加しています。Q2 はおそらく、ユーザ数 1,000 人で長時間スワップ・アウトされたものと予想されます。



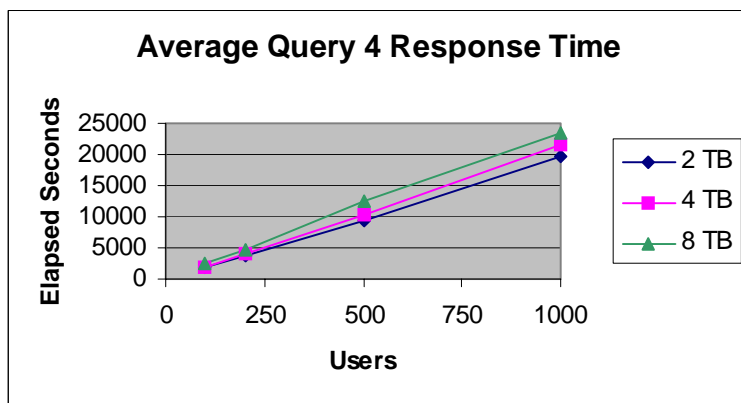
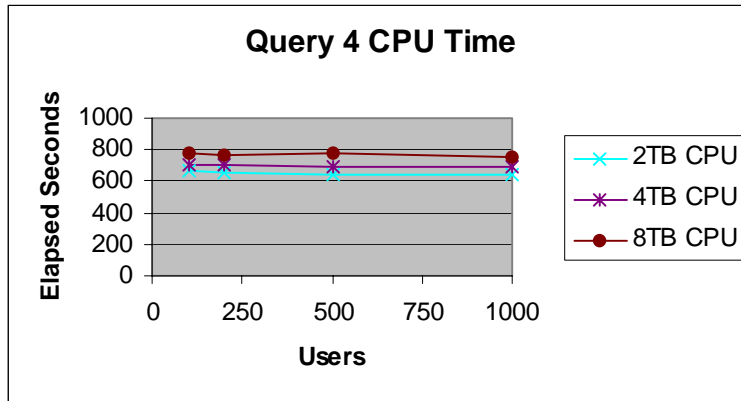
(3) Medium クエリ

Q8 は、ユーザ数とデータベース・サイズを拡張したときに、クエリ応答時間に何らかの影響を与える I/O 動作の結果を示しています。なお、CPU 時間は重要なファクタではありません。2TB と 4TB 間では、期待どおり応答時間が同じ状態を維持しています。しかしながら 8TB では、応答時間が急なレートで直線的に増加しています。



(4) Large クエリ

Q4 では、平均 CPU 時間がどのユーザ数でも同じ状態を維持しており、4TB と 8TB 間が比較的近くなっています。また、応答時間も 3 つのデータベース・サイズ間で近い状態を維持していますが、ユーザの増分とともに直線的に増加しています。応答時間が長いのは、システム負荷が大きいことと、I/O ローカルティが低いことを反映しています。



マルチユーザ用のチューニング・パラメータ

	ユーザ数 100	ユーザ数 200	ユーザ数 500	ユーザ数 1000
IBM Red Brick 6.11.xC3				
TOTAL QUERY PROCS	2,000	2,000	2,000	2,000
QUERY PROCS	8	8	4	1
(2 & 4TB)QUERY MEMORY LIMIT	100M	100M	80M	80M
(8TB)QUERY MEMORY LIMIT	80M	80M	80M	80M
ROWS_PER_....TASK	1,000	1,000	1,000	1,000
TUNE GROUP	144	144	144	144
オペレーティング・システム				
ubc_maxpercent	70	70	65	60

この調査では、IBM Red Brick の動的並列処理とメモリ割り当てを利用しています。この場合は、rbw.config ファイル内の設定可能値を変更し、サーバを再起動するだけで済みます。

結論

POPS のベンチマーク結果は、HP GS320 システムと IBM Red Brick Warehouse のどちらも優れた性能と拡張性を持っていることを実証しています。HP GS320 大容量システムは、多数の I/O 要求と多数のユーザに耐えることが可能です。IBM Red Brick Warehouse は、多数のユーザとともにマルチテラバイトのデータ・ウェアハウスをサポートしながらも、高いクエリ性能を維持することができます。

この調査では、全体的に優れた性能を実証することができました。最初の一括データ・ロード・レートは最大 288 GB/時間です。ユーザは、1 TB のデータを 4 時間以内にロードして、クエリの実行を開始する準備を整えることができます。複雑な意思決定支援クエリを使用した場合、1 時間あたり最大 2,500 の高いクエリ・スループットを持続することができます。ユーザ数を最大 1,000 人まで増やしてシステム負荷を大きくした場合でも、各データベース・サイズのスループットを同レベルに維持することが可能です。

この調査では、データベース・サイズとユーザ数の両方において、優れた拡張性を実証することができました。データベース・サイズに関連して、直線的な拡張を行うことが可能です。また、マルチユーザにおいても、優れた拡張性を実証することができました。ユーザ数最大 1,000 人の大きな負荷条件下でも、急激ではなく直線的に増加する、望ましいクエリ応答時間の動きを維持することが可能です。さらに、ユーザ数を増やしたときでも、クエリあたりの処理 (CPU) 時間を一定に維持することが可能です。

ベンチマーク結果は、調査目的を証明したものであり、なおかつ当初の期待を超えるものでした。場合によっては、直線的な性能以上のものを実証することができました。IBM Red Brick Warehouse で最適化を利用したときには、直線的なスピードアップよりもさらに優れたロード性能とクエリ性能を示しました。

付録 A

ハードウェアの詳細説明

Tru64 V5 バージョンのオペレーティング・システムを実行する AlphaServer GS320

- QBB (クワッド・ビルディング・ブロック) 数:8 (それぞれ 1Ghz CPU 4 個と Gig RAM 16 個を装備)
- PCI ドロワ数:8 (各 QBB に 1 つの PCI ドロワが接続される)
- PCI バス数:32 (PCI ドロワあたりの PCI バス数:4)
- ファイバ・チャンネル・ホスト・ベース・アダプタ数:32 (PCI バスあたりの HBA 数:1)
- 各 QBB は、すべてのファイバ・チャンネル格納への直接パスを持つ

ファイバ・チャンネル・スイッチ数:8 (スイッチあたりの FC ポート数:16)

- ファイバ・チャンネル・ポート合計:128
- ホスト/サーバ・ファイバ・チャンネル接続数:32 (FC スイッチあたりの数:4)
- 格納コントローラ FC 接続数:96 (FC スイッチあたりの数:12)

HSG80 StorageWORKS ファイバ・チャンネル・コントローラ数:24

- HSG80 コントローラ数:48 (HSG80 あたりのコントローラ数:2)
- 各コントローラは 256MB ミラーリング・キャッシュを装備
- コントローラ・ペアあたりのファイバ・チャンネル・ポート数:4
- EMA 12000 StorageWorks RAID システム
- ディスク・ドライブ数:1008 (コントローラ・ペアあたりのドライブ数:42)
 - 18GB/15K RPM ドライブ数:504 (コントローラ・ペアあたりの数:21)
 - 36GB/10K RPM ドライブ数:504 (コントローラ・ペアあたりの数:21)
- Raid5 セット数:144 (コントローラ・ペアあたりのセット数:6)
 - 各 7 台の 18GB/15K RPM ドライブを使用する Raid5 セット数:3
 - 各 7 台の 36GB/10K RPM ドライブを使用する Raid5 セット数:3
- HSG80 Raid5 のデフォルトのストライプ・サイズ:128KB
- 18GB/15K RPM ドライブの Raid5 セット合計:72
- 36GB/10K RPM ドライブの Raid5 セット合計:72

論理格納域マネージャ (ソフトウェア Raid)

- /k15 論理ボリューム内の 18GB/15K RPM Raid5 セット上の Raid0 ストライプセット数:
- Raid5 セット 24 個ごとの Raid0 ストライプセット数:2 (コントローラ・ペアあたりの数:1)
これは、論理ボリューム /k15 に対する 7.5TB の AdvFS ドメイン合計ユーザ格納域内で、1TB/ボリュームの制限があるためです。

ユームの制限があるためです。

- 論理ボリューム /k10 内の 36GB/10K RPM Raid5 セットの Raid0 ストライプセット数:9
- Raid5 セット 24 個ごとの Raid0 ストライプセット数:3 (768KB のストライプ・サイズを使用したコントローラ・ペアあたりの数:1)

これは、論理ボリューム /k10 に対する 15TB の AdvFS ドメイン合計ユーザ格納域内で、1TB/ボリュームの制限があるためです。

- すべてのソフトウェア・ストライプセットに対して 768KB のストライプ・サイズを使用

OS パラメータの設定

generic:

```
version_vendor = Compaq Computer Corporation
version_avendor = COMPAQ
version_product = Tru64 UNIX
version_banner = Compaq Tru64 UNIX
  memberid = 0
    new_vers_high = 1445664276479072064
    new_vers_low = 51969
```

vm:

```
swapdevice=/dev/disk/dsk2b,/dev/disk/dsk0c,/dev/disk/dsk1c,/dev/disk/dsk152c,/d
ev/disk/dsk153c,/dev/disk/dsk154c,/dev/disk/dsk155c,/dev/disk/dsk156c,/dev/disk/
dsk157c,/dev/disk/dsk158c,/dev/disk/dsk159c,/dev/disk/dsk160c
  vm-swap-eager=0
  ubc_maxpercent=50
```

ipc:

```
msg_mnb=65535
msg_mni=6144
msg_tql=90000
shm_max=16777216
shm_mni=6144
shm_seg=640
sem_mni=4096
sem_msl=4096
```

proc:

```
max_proc_per_user=10076
per_proc_data_size=268435456
max_per_proc_data_size=68719476736
```

vfs:

```
nlock_record=100000
```

ufs:

```
ufs_lockholdmax=2000
```

pcount:

```
Subsystem_Description = pcount device driver
Module_Config_Name = pcount
Module_Type = Dynamic
Device_Char_Major = ANY
Device_Char_Minor = 0
Device_Char_Files = pcount0
```

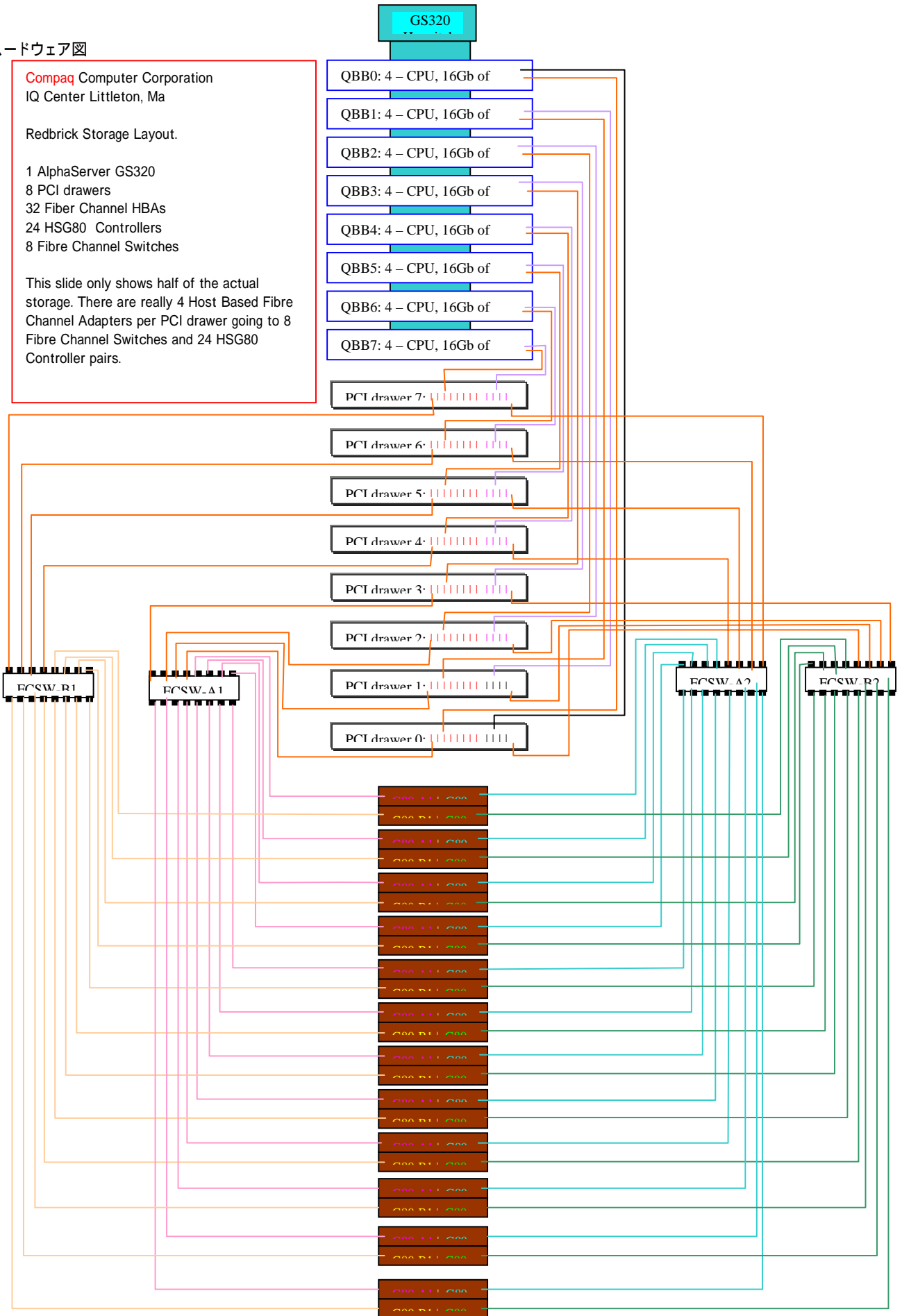
ハードウェア図

Compaq Computer Corporation
 IQ Center Littleton, Ma

Redbrick Storage Layout.

1 AlphaServer GS320
 8 PCI drawers
 32 Fiber Channel HBAs
 24 HSG80 Controllers
 8 Fibre Channel Switches

This slide only shows half of the actual storage. There are really 4 Host Based Fibre Channel Adapters per PCI drawer going to 8 Fibre Channel Switches and 24 HSG80 Controller pairs.



付録 B クエリ

以下に、テスト時に使用されたクエリ・スクリプトを示します。本書にリストされているデータ範囲は、実際に使用されたデータ範囲を反映していない場合があることに注意してください。各クエリ・バージョンに合わせて調整されたデータ範囲は、本来ならば発生するはずの「キャッシュ効果」を低減するために設定されています。

クエリ 1:

```
/* Query1 所定のカテゴリについて、販売数量、売上、費用を含むすべての項目をリスト */
```

```
SELECT ITEM_DESC,  
       SUM(QUANTITY_SOLD),  
       SUM(EXTENDED_PRICE),  
       SUM(EXTENDED_COST)  
FROM PERIOD, DAILY_SALES, PRODUCT, STORE, PROMOTION  
WHERE  
  PERIOD.PERKEY=DAILY_SALES.PERKEY AND  
  PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND  
  STORE.STOREKEY=DAILY_SALES.STOREKEY AND  
  PROMOTION.PROMOKEY=DAILY_SALES.PROMOKEY AND  
  CALENDAR_DATE BETWEEN 'Jan 01 1996' AND 'Jan 28 1996' AND  
  STORE_NUMBER='01' AND  
  PROMODESC IN ('Advertisement', 'Coupon', 'Manager's Special',  
                'Overstocked Items') AND  
  CATEGORY=42  
GROUP BY ITEM_DESC;
```

クエリ 2:

```
/* Query2 ある品目カテゴリの 4 週間にわたる週間売上  
*/
```

```
SELECT PRODUCT.UPC_NUMBER,  
       SUM(SALES_WEEK1) AS SALES_WEEK1,  
       SUM(SALES_WEEK2) AS SALES_WEEK2,  
       SUM(SALES_WEEK3) AS SALES_WEEK3,  
       SUM(SALES_WEEK4) AS SALES_WEEK4  
FROM PRODUCT,  
     (SELECT UPC_NUMBER, SUM(EXTENDED_PRICE)  
      FROM PERIOD, PRODUCT, STORE, DAILY_SALES  
      WHERE  
        DAILY_SALES.PERKEY=PERIOD.PERKEY AND  
        DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND  
        DAILY_SALES.STOREKEY=STORE.STOREKEY AND  
        PRODUCT.CATEGORY=1 AND  
        PRODUCT.SUB_CATEGORY = 50 AND  
        STORE_NUMBER='01' AND  
        PERIOD.WEEK_ENDING_DATE = '2/3/1996'  
      GROUP BY UPC_NUMBER) AS
```

```

T1(UPC_NUMBER,SALES_WEEK1),
(SELECT UPC_NUMBER,SUM(EXTENDED_PRICE)
FROM PERIOD, PRODUCT, STORE, DAILY_SALES
WHERE
DAILY_SALES.PERKEY=PERIOD.PERKEY AND
DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND
DAILY_SALES.STOREKEY=STORE.STOREKEY AND
PRODUCT.CATEGORY=1 AND
PRODUCT.SUB_CATEGORY = 50 AND
STORE_NUMBER='01' AND
PERIOD.WEEK_ENDING_DATE = '2/10/1996'
GROUP BY UPC_NUMBER) AS
T2(UPC_NUMBER,SALES_WEEK2),
(SELECT UPC_NUMBER,SUM(EXTENDED_PRICE)
FROM PERIOD, PRODUCT, STORE, DAILY_SALES
WHERE
DAILY_SALES.PERKEY=PERIOD.PERKEY AND
DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND
DAILY_SALES.STOREKEY=STORE.STOREKEY AND
PRODUCT.CATEGORY=1 AND
PRODUCT.SUB_CATEGORY = 50 AND
STORE_NUMBER='01' AND
PERIOD.WEEK_ENDING_DATE = '2/17/1996'
GROUP BY UPC_NUMBER) AS
T3(UPC_NUMBER,SALES_WEEK3),
(SELECT UPC_NUMBER,SUM(EXTENDED_PRICE)
FROM PERIOD, PRODUCT, STORE, DAILY_SALES
WHERE
DAILY_SALES.PERKEY=PERIOD.PERKEY AND
DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND
DAILY_SALES.STOREKEY=STORE.STOREKEY AND
PRODUCT.CATEGORY=1 AND
PRODUCT.SUB_CATEGORY = 50 AND
STORE_NUMBER='01' AND
PERIOD.WEEK_ENDING_DATE = '2/24/1996'
GROUP BY UPC_NUMBER) AS
T4(UPC_NUMBER,SALES_WEEK4)
WHERE
PRODUCT.UPC_NUMBER=T1.UPC_NUMBER and
PRODUCT.UPC_NUMBER=T2.UPC_NUMBER and
PRODUCT.UPC_NUMBER=T3.UPC_NUMBER and
PRODUCT.UPC_NUMBER=T4.UPC_NUMBER
GROUP BY PRODUCT.UPC_NUMBER
ORDER BY PRODUCT.UPC_NUMBER;

```

クエリ 3:

```
/* Query3 所定の日付における所定の店舗別カテゴリの店舗あたり利益。地区別の店舗あたりシェア */
```

```
SELECT T1.STORE_NUMBER,T1.CITY,T1.DISTRICT,SUM(AMOUNT) AS SUM_PROFIT,  
DEC(100*RATIOREPORT(SUM_PROFIT),5,2) AS SHARE_OF_DISTRICT  
FROM  
(SELECT STORE_NUMBER,STORE.CITY,DISTRICT, EXTENDED_PRICE-EXTENDED_COST  
FROM  
PERIOD, PRODUCT,STORE, DAILY_SALES WHERE  
PERIOD.CALENDAR_DATE='3/1/1996' AND  
PERIOD.PERKEY=DAILY_SALES.PERKEY AND  
PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND  
STORE.STOREKEY=DAILY_SALES.STOREKEY AND  
PRODUCT.CATEGORY=42 ) AS T1(STORE_NUMBER,CITY,DISTRICT,AMOUNT)  
GROUP BY DISTRICT,CITY, STORE_NUMBER  
ORDER BY DISTRICT,CITY, STORE_NUMBER DESC  
RESET BY DISTRICT;
```

クエリ 4:

```
/* Query4 1ヶ月間にわたる所定の顧客の消費額 */  
SELECT WEEK_ENDING_DATE,daily_sales.custkey,SUM(EXTENDED_PRICE) As cust_rev  
from period join daily_sales on period.perkey = daily_sales.perkey  
join customer on customer.custkey = daily_sales.custkey  
WHERE CUSTOMER.CUSTKEY in (215311,627294)  
AND CALENDAR_DATE BETWEEN '1/1/1996' AND '1/31/1996'  
GROUP BY WEEK_ENDING_DATE, daily_sales.custkey  
ORDER BY WEEK_ENDING_DATE;
```

クエリ 5:

```
/* Query5 所定の地域内カテゴリの週間利益 */  
SELECT Sub_Category_Desc,  
Sum(Case Region When 'North' Then Extended_Price-Extended_Cost Else 0  
End) As Northern_Region,  
Sum(Case Region When 'South' Then Extended_Price-Extended_Cost Else 0  
End) As Southern_Region,  
Sum(Case Region When 'East' Then Extended_Price-Extended_Cost Else 0  
End) As Eastern_Region,  
Sum(Case Region When 'West' Then Extended_Price-Extended_Cost Else 0  
End) As Western_Region,  
NORTHERN_REGION+SOUTHERN_REGION+EASTERN_REGION+WESTERN_REGION As  
ALL_REGIONS  
FROM PERIOD, PRODUCT, STORE, DAILY_SALES WHERE  
PERIOD.PERKEY=DAILY_SALES.PERKEY AND  
PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND  
STORE.STOREKEY=DAILY_SALES.STOREKEY AND  
PERIOD.CALENDAR_DATE BETWEEN '5/1/1996' AND '5/9/1996'  
AND CATEGORY=88  
GROUP BY SUB_CATEGORY_DESC  
ORDER BY SUB_CATEGORY_DESC;
```

クエリ 6:

/* Query6 所定の店舗での所定のブランドの売上。全店舗での同ブランドの売上合計 */

```
SELECT ITEM_DESC,
       EXP AS SALES,
       SUM(EXTENDED_PRICE) AS TOTAL_SALES
FROM PRODUCT, DAILY_SALES, PERIOD,
     (SELECT ITEM_DESC,SUM(EXTENDED_PRICE)
      FROM PRODUCT, DAILY_SALES, PERIOD,STORE
      WHERE WEEK_ENDING_DATE='Jun 29 1996' AND
            PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND
            PERIOD.PERKEY=DAILY_SALES.PERKEY AND
            STORE.STOREKEY=DAILY_SALES.STOREKEY AND
            STORE_NUMBER='01' AND
            ITEM_DESC LIKE 'NESTLE%'
      GROUP BY ITEM_DESC
     ) AS T1(ITEM,EXP)
WHERE ITEM_DESC LIKE 'NESTLE%' AND
      PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND
      PERIOD.PERKEY=DAILY_SALES.PERKEY AND
      T1.ITEM=ITEM_DESC AND
      WEEK_ENDING_DATE='Jun 29 1996'
GROUP BY ITEM_DESC,EXP;
```

クエリ 7:

/* Query7 所定のカテゴリまたはブランド内の各製品の売上数量および売上シェア */

```
SELECT ITEM_DESC, T1.UNITS AS UNITS, T1.UNITS_RATIO AS UNITS_RATIO,
       T1.REVENUE AS REVENUE, T1.REVENUE_RATIO AS REVENUE_RATIO FROM
     (SELECT ITEM_DESC
      ,SUM(QUANTITY_SOLD) AS UNITS
      ,RATIOTOREPORT(UNITS)*100 AS UNITS_RATIO
      ,SUM(EXTENDED_PRICE) AS REVENUE
      ,RATIOTOREPORT(REVENUE)*100 AS REVENUE_RATIO
     FROM PRODUCT, PERIOD, DAILY_SALES
      WHERE PRODUCT.PRODKEY=DAILY_SALES.PRODKEY AND
            PERIOD.PERKEY=DAILY_SALES.PERKEY AND
            ITEM_DESC LIKE 'NESTLE%' AND
            WEEK_ENDING_DATE='Jul 27 1996'
      GROUP BY ITEM_DESC)
     AS T1(ITEM_DESC, UNITS,UNITS_RATIO,REVENUE,REVENUE_RATIO)
ORDER BY ITEM_DESC
BREAK BY ITEM_DESC SUMMING 2,3,4,5;
```

クエリ 8:

/* Query8 所定のブランドの売上に基づいた店舗あたり売上。店舗、今週、先週、および先月合計別に製品をグループ化すべきかどうか */

```
SELECT STORE_NUMBER,
       SUM(CASE WHEN ((CALENDAR_DATE >= '8/8/1996')
                     AND (CALENDAR_DATE < '8/14/1996'))
           THEN EXTENDED_PRICE ELSE 0 END) AS CURR_PERIOD,
       SUM(CASE WHEN ((CALENDAR_DATE >= '8/1/1996')
                     AND (CALENDAR_DATE <= '8/7/1996'))
           THEN EXTENDED_PRICE ELSE 0 END) AS PRIOR_WEEK,
       SUM(CASE WHEN ((CALENDAR_DATE >= '7/1/1996')
                     AND (CALENDAR_DATE <= '7/28/1996'))
           THEN EXTENDED_PRICE ELSE 0 END) AS PRIOR_MONTH
FROM PERIOD,PRODUCT,DAILY_SALES,STORE
WHERE PRODUCT.PRODKEY=DAILY_SALES.PRODKEY
      AND PERIOD.PERKEY=DAILY_SALES.PERKEY
      AND STORE.STOREKEY=DAILY_SALES.STOREKEY
      AND CALENDAR_DATE BETWEEN '7/1/1996' and '8/14/1996'
      AND ITEM_DESC LIKE 'NESTLE%'
GROUP BY STORE_NUMBER
ORDER BY STORE_NUMBER;
```

クエリ 9:

/* Query9 1993年1月9日と1993年1月16日で終わる週についての、店舗番号11の実売上と予測売上 */

```
SELECT SUM(EXTENDED_PRICE) AS ACTUAL_REVENUE, SUM(EXTENDED_PRICE_FORECAST)
       AS FORECASTED_REVENUE
FROM DAILY_SALES, STORE, PERIOD, PRODUCT, DAILY_FORECAST
WHERE
DAILY_SALES.STOREKEY=STORE.STOREKEY AND
DAILY_FORECAST.STOREKEY=STORE.STOREKEY AND
DAILY_FORECAST.STOREKEY=DAILY_SALES.STOREKEY AND
DAILY_SALES.PERKEY=PERIOD.PERKEY AND
DAILY_FORECAST.PERKEY=PERIOD.PERKEY AND
DAILY_SALES.PERKEY=DAILY_FORECAST.PERKEY AND
DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND
DAILY_FORECAST.PRODKEY=PRODUCT.PRODKEY AND
DAILY_SALES.PRODKEY=DAILY_FORECAST.PRODKEY AND
STORE_NUMBER='11' AND
WEEK_ENDING_DATE IN ('9/07/1996','9/14/1996');
```

クエリ 10:

/* Query10 2週間にわたる、ある店舗での所定の公告製品カテゴリに関する実売上 / 費用と予測売上 / 費用 */

```
SELECT
    SUM(EXTENDED_PRICE) AS ACTUAL_REVENUE,
    SUM(EXTENDED_COST) AS ACTUAL_COST,
    SUM(EXTENDED_PRICE_FORECAST) AS PROJECTED_REVENUE,
    SUM(EXTENDED_COST_FORECAST) AS PROJECTED_COST
FROM STORE, PERIOD, PROMOTION, PRODUCT, DAILY_SALES, DAILY_FORECAST
WHERE
    DAILY_SALES.PERKEY=PERIOD.PERKEY AND
    DAILY_SALES.STOREKEY=STORE.STOREKEY AND
    DAILY_SALES.PRODKEY=PRODUCT.PRODKEY AND
    DAILY_SALES.PROMOKEY=PROMOTION.PROMOKEY AND

    DAILY_FORECAST.PERKEY=PERIOD.PERKEY AND
    DAILY_FORECAST.STOREKEY=STORE.STOREKEY AND
    DAILY_FORECAST.PRODKEY=PRODUCT.PRODKEY AND

    DAILY_SALES.PRODKEY=DAILY_FORECAST.PRODKEY AND
    DAILY_SALES.STOREKEY=DAILY_FORECAST.STOREKEY AND
    DAILY_SALES.PERKEY=DAILY_FORECAST.PERKEY AND

    STORE.STORE_NUMBER='11' AND
    PERIOD.WEEK_ENDING_DATE IN ('Oct 5 1996','Oct 12 1996') AND
    PROMOTION.PROMODESC='Advertisement' AND
    PRODUCT.CATEGORY IN (12, 13, 22, 42);
```

クエリ 11:

/* Query11: Customer デイメンジョンのブラウズ
現在抱えている顧客数 */

```
select count(*) from customer;
```

クエリ 12:

/* Query12: Customer デイメンジョンのブラウズ
顧客の年齢層と収入層 */

```
select min(age_level_desc), max(age_level_desc),
    min(income_level_desc), max(income_level_desc) from customer;
```

クエリ 13:

/* Query13: 所定の収入レベルを超える顧客の数 */

```
select count(*) from customer
  where income_level_desc >= '$35,000 to $44,999';
```

クエリ 14:

```
/* Query14: Customer デイメンジョンのブラウズ
   男性顧客数と女性顧客数 */
```

```
select gender, count(*) from customer
  group by gender;
```

クエリ 15:

```
/* Query15: 選択的年齢層別の顧客数 */
```

```
select age_level_desc, count(*) from customer
  where age_level_desc between '23 to 30' and '41 to 50'
  group by age_level_desc;
```

クエリ 16:

```
/* Query16: 顧客の収入層の範囲 */
```

```
select distinct income_level_desc from customer;
```

クエリ 17:

```
/* Query17: 指定された顧客キー・リストに関する個々の郵便番号 (または income_level、age_level、states)
   の数 */
```

```
select count(distinct income_level), count(distinct age_level) from customer
  where custkey in (
962369,
965423,
968285,
969555,
1208,
26262,
71425,
88610,
201783,
259096,
379152,
523366,
191918,
```

```
193523,  
 (...13000 more values...)  
340602,  
765526
```

```
);
```

クエリ 18:

/* Query18: 所定の年齢層および収入層に該当する女性顧客の数 */

```
select count(*) from customer  
where      age_level_desc = '23 to 30'  
          and INCOME_LEVEL_DESC = '$35,000 to $44,999'  
          and gender = 'F';
```