IBM® DB2 Universal Database™

Visual Explain チュートリアル

バージョン 8

IBM® DB2 Universal Database™

Visual Explain チュートリアル

バージョン 8

ご注意!

本書および本書で紹介する製品をご使用になる前に、*特記事項* に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原 典:	IBM® DB2 Universal Database™ Visual Explain Tutorial Version 8
発 行:	日本アイ・ビー・エム株式会社
担 当:	ナショナル・ランゲージ・サポート

第1刷 2002.10

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

このチュートリアルについて

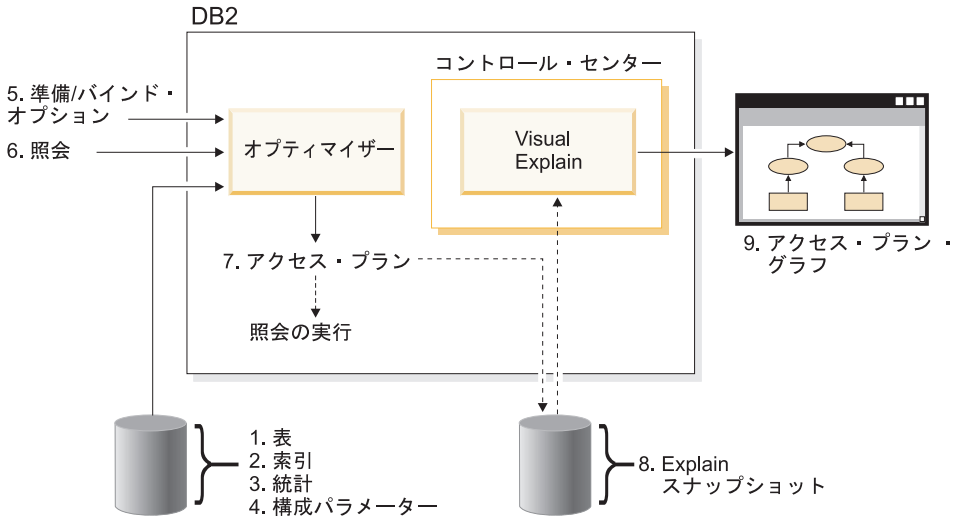
このチュートリアルでは、DB2 Visual Explain のさまざまな機能を紹介します。このチュートリアルのレッスンを完了すると、Visual Explain で、 EXPLAIN された SQL ステートメントのアクセス・プランをグラフで表示する方法を学習できます。さらに、グラフからの情報を使用して、 SQL 照会のパフォーマンスを向上させることができます。

DB2 は、オプティマイザーを使用して、提供された SQL 照会を検証して、最適なデータ・アクセス方法を決定します。このデータへのパスを、アクセス・プランと呼びます。 DB2 では、特定の SQL 照会を実行するためにオプティマイザーにより選択されたアクセス・プランを参照することができます。アクセス・プランは、Visual Explain でグラフとして表示できます。グラフは、照会で使用するデータベース・オブジェクト (たとえば、表や索引) を見える形式で表現したものです。さらに、グラフにはこれらのオブジェクトに対して実行する操作 (たとえば、スキャン操作やソート操作)、およびデータの流れも表現されます。

以下のチューニング・アクティビティーの一部または全部を実行することにより、照会でのデータ・アクセスを向上させることができます。

1. 表の設計を調整し、表データを再編成する。
2. 適切な索引を作成する。
3. **runstats** コマンドを使用して、オプティマイザーに現在の統計を提供する。
4. 適切な構成パラメーターを選択する。
5. 適切なバインド・オプションを選択する。
6. 必要なデータだけを検索するように照会を設計する。
7. アクセス・プランを処理する。
8. EXPLAIN スナップショットを作成する。
9. アクセス・プラン・グラフを使用してアクセス・プランを改善する。

これらのパフォーマンス関連のアクティビティーは、次の図に対応しています。 (破線は、Visual Explain に必要なアクションを示します。)



このチュートリアルでは、以下のレッスンを学習します。

- EXPLAIN スナップショットの作成。 EXPLAIN スナップショットは、アクセス・プラン・グラフを表示するのに必要です。
- アクセス・プラン・グラフの表示および操作。
- チューニング・アクティビティーの実行。およびそれらのアクティビティーがアクセス・プランに与える影響の判別。

  注: パフォーマンスのチューニングは、単一パーティション・データベース環境のためのレッスンと、パーティション・データベース環境のためのレッスンに分けられています。

一連のレッスンでは、DB2 が提供する SAMPLE データベースを使用します。SAMPLE データベースをまだ作成していない場合は、*管理ガイド* を参照してください。

環境固有の情報

このアイコンのマークが付けられている情報は、単一パーティション・データベース環境だけに関係するものです。

このアイコンのマークが付けられている情報は、パーティション・データベース環境だけに関係するものです。

レッスン 1. EXPLAIN スナップショットの作成

このレッスンでは、EXPLAIN スナップショットを作成します。 SQL Explain 機能は、静的または動的 SQL ステートメントをコンパイルする環境に関する情報を収集するときに使用します。収集した情報を利用すると、SQL ステートメントの構造と、予想される SQL ステートメントの実行パフォーマンスを理解することができます。 EXPLAIN スナップショットは、 SQL ステートメントが EXPLAIN されたときに収集された情報を圧縮したものです。これは EXPLAIN_STATEMENT 表にバイナリー・ラージ・オブジェクト (BLOB) として保管され、以下の情報が入っています。

- アクセス・プランの内部表記。これには、アクセス・プランの演算子と、アクセスされる表および索引が含まれます。
- オプティマイザーが使用する決定の基準。これには、データベース・オブジェクトの統計と各操作のコストの累計が含まれます。

アクセス・プラン・グラフを表示するには、 Visual Explain で、EXPLAIN スナップショットに格納された情報が必要です。

EXPLAIN 表の作成

EXPLAIN スナップショットを作成するには、以下の EXPLAIN 表がユーザーの ID に存在することを確認する必要があります。

- EXPLAIN_INSTANCE
- EXPLAIN_STATEMENT

これらが存在することを調べるには、 **DB2 list tables** コマンドを使用してください。これらの表が存在しないときは、以下の手順でこれらの表を作成する必要があります。

1. DB2 がすでに開始していない場合は、 **db2start** コマンドを実行します。
2. DB2 CLP プロンプトから、使用するデータベースに接続します。このチュートリアルでは、**connect to sample** コマンドを使用して SAMPLE データベースに接続します。
3. EXPLAIN.DDL ファイルにあるサンプル・コマンド・ファイルを使用して、EXPLAIN 表を作成します。このファイルは、sqllib¥misc ディレクトリーにあります。このコマンド・ファイルを実行するには、このディレクトリーで **db2 -tf EXPLAIN.DDL** コマンドを実行してください。このコマンド・ファイルは、接続したユーザー ID を接頭部に持つ EXPLAIN 表を作成します。このユーザー ID には、データベースに対する CREATETAB 特権か、 SYSADM または DBADM 権限がなければなりません。

EXPLAIN スナップショットの使用

Visual Explain について学習するために、4 つのサンプル・スナップショットが提供されています。以下のセクションでは、独自のスナップショットの作成について説明しますが、このチュートリアルで学習するために独自のスナップショットを作成する必要は必ずしもありません。

- 動的 SQL ステートメント用の EXPLAIN スナップショットの作成
- 静的 SQL ステートメント用の EXPLAIN スナップショットの作成

サンプル・スナップショットに対して照会を実行し、給与額が、管理者の給与の最高額の 90% を超えているすべての非管理者従業員の名前、部門、給与額をリストします。

```
SELECT S.ID,S.NAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
  O.DEPTNUMB = S.DEPT AND
  S.JOB <> 'Mgr' AND
  S.SALARY+S.COMM > ALL( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

上の照会は、以下の 2 つの部分から構成されています。

1. 副照会 (括弧で囲まれた部分) では、各管理者の給与の 90% の値が記述された行が生成されます。副照会の修飾子は ALL で定義されているため、この表の最大値だけが取得されます。
2. 主照会では、部門番号が等しく、JOB 値が 'Mgr' ではなく、給与と歩合を合計した額が副照会の戻り値より大きい ORG 表の行と STAFF 表の行が結合されます。

主照会には、以下の 3 つの述部 (比較) が含まれています。

```
1.   O.DEPTNUMB = S.DEPT
2.   S.JOB <> 'Mgr'
3.   S.SALARY+S.COMM > ALL ( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
```

これらの述部は、以下のものを表します。

1. 部門番号が同等の行で ORG 表と STAFF 表を結合するための結合述部
2. STAFF 表の JOB 列に対するローカル述部
3. 副照会の結果を使用する STAFF 表の SALARY 列と COMM 列に対するローカル述部。

サンプル・スナップショットをロードするには、以下の手順に従ってください。

1. DB2 がすでに開始していない場合は、 **db2start** コマンドを実行します。

2. EXPLAIN 表がデータベースに存在していることを確認します。 EXPLAIN 表の作成の指示に従ってください。
3. 使用するデータベースに接続します。このチュートリアルでは、SAMPLE データベースに接続します。 SAMPLE データベースに接続するには、DB2 CLP プロンプトで **connect to sample** コマンドを実行します。

   データベースが作成されていない場合は、 SAMPLE データベースのインストールに関する*管理ガイド* のセクションを参照してください。
4. 定義済みのスナップショットをインポートするには、 DB2 コマンド・ファイル VESAMPL.DDL を実行します。

   -  このファイルは、 sqllib¥samples¥ve ディレクトリーにあります。
   -  このファイルは、 sqllib¥samples¥ve¥inter ディレクトリーにあります。

   このコマンド・ファイルを実行するには、このディレクトリーで **db2 -tf vesampl.ddl** コマンドを実行します。
   - このコマンド・ファイルを実行するには、 EXPLAIN 表を作成するときと同じユーザー ID を使用する必要があります。
   - このコマンドは、定義済みスナップショットのインポートだけを実行します。表やデータの作成は行いません。 SAMPLE データベースの表およびデータに対して、後述のチューニング・アクティビティー (たとえば、CREATE INDEX および runstats) を実行します。

これで、アクセス・プラン・グラフを表示および使用するための準備ができました。

動的 SQL ステートメント用の EXPLAIN スナップショットの作成

注: このセクションに記載されている EXPLAIN スナップショットの作成に関する情報は、リファレンスとして提供されています。ただし実際には、サンプルの EXPLAIN スナップショットが提供されているため、チュートリアルでの学習を進めるのにこのタスクを完了する必要はありません。

動的 SQL ステートメントに関する EXPLAIN スナップショットを作成するには、以下の手順に従ってください。

1. DB2 がすでに開始していない場合は、 **db2start** コマンドを実行します。
2. EXPLAIN 表がデータベースに存在していることを確認します。 EXPLAIN 表の作成の指示に従ってください。
3. DB2 CLP プロンプトから、使用するデータベースに接続します。たとえば、SAMPLE データベースに接続するには、 **connect to sample** コマンドを実行します。

   SAMPLE データベースを作成するには、 SAMPLE データベースのインストールに関する*管理ガイド* のセクションを参照してください。

4. DB2 CLP プロンプトで以下のコマンドのいずれかを実行して、動的 SQL ステートメントの EXPLAIN スナップショットを作成します。
   - SQL ステートメントを実行せずに EXPLAIN スナップショットを作成するには、**set current explain snapshot=explain** コマンドを実行します。
   - EXPLAIN スナップショットを作成して SQL ステートメントを実行するには、**set current explain snapshot=yes** コマンドを実行します。

   このコマンドは、Explain 特殊レジスターを設定します。この設定を行うと、後続のすべての SQL ステートメントが影響を受けます。詳細については、*SQL リファレンス* で、現在の EXPLAIN スナップショットについてのセクションを参照してください。
5. DB2 CLP プロンプトで、SQL ステートメントを実行します。
6. スナップショットのアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウを最新表示し (Control Center で使用可能)、スナップショットをダブルクリックします。
7. オプション。スナップショット機能をオフにするには、SQL ステートメントを実行した後で、 **set current explain snapshot=no** コマンドを実行します。

静的 SQL ステートメント用の EXPLAIN スナップショットの作成

注: このセクションに記載されている EXPLAIN スナップショットの作成に関する情報は、リファレンスとして提供されています。ただし実際には、サンプルの EXPLAIN スナップショットが提供されているため、チュートリアルでの学習を進めるのにこのタスクを完了する必要はありません。

静的 SQL スナップショットに関する EXPLAIN スナップショットを作成するには、以下の手順に従ってください。

1. DB2 がすでに開始していない場合は、 **db2start** コマンドを実行します。
2. EXPLAIN 表がデータベースに存在していることを確認します。 EXPLAIN 表の作成の指示に従ってください。
3. DB2 CLP プロンプトから、使用するデータベースに接続します。たとえば、SAMPLE データベースに接続するには、 **connect to sample** コマンドを実行します。
4. アプリケーションをバインドまたは準備するときに EXPLSNAP オプションを指定して、静的 SQL ステートメント用の EXPLAIN スナップショットを作成します。たとえば、**bind** *your file* **explsnap yes** コマンドを実行します。
5. オプション。スナップショットのアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウを最新表示し (Control Center で使用可能)、スナップショットをダブルクリックします。

同等の API での EXPLSNAP オプションの使用については、*アプリケーション開発ガイド* の該当するセクションを参照してください。

次のステップ

7 ページの『レッスン 2. アクセス・プラン・グラフの表示および使用』では、アクセス・プラン・グラフを表示する方法と、グラフの内容について学習します。

レッスン 2. アクセス・プラン・グラフの表示および使用

このレッスンでは、「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウを使用して、アクセス・プラン・グラフを表示および使用します。アクセス・プラン・グラフは、 アクセス・プランをグラフ形式で表示したものです。アクセス・プラン・グラフでは、以下の情報を表示できます。

- 表 (およびそれに関連する列) と索引
- 演算子 (表スキャン、ソート、結合など)
- 表スペースおよび関数

アクセス・プラン・グラフは、以下の方法で表示することができます。

- すでに EXPLAIN されたステートメントのリストから選択する。
- パッケージに含まれている EXPLAIN 可能ステートメントのリストから選択する。
- SQL ステートメントを動的に Explain する。

ここでは、レッスン 1 でロードしたサンプルの EXPLAIN スナップショット用のアクセス・プラン・グラフを操作するため、すでに EXPLAIN されたステートメントのリストから選択します。アクセス・プラン・グラフを表示する他の方法については、 Visual Explain のヘルプを参照してください。

すでに EXPLAIN された SQL ステートメントのリストからアクセス・プラン・グラフを選択および表示する

すでに EXPLAIN された SQL ステートメントのリストからアクセス・プラン・グラフを選択して、グラフを表示するには、以下の手順に従ってください。

1. Control Center で、 SAMPLE データベースが表示されるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューから**「EXPLAIN されたステートメントのヒストリーの表示 (Show explained statements history)」**を選択します。「EXPLAIN されたステートメントのヒストリー (Explained Statement History)」ウィンドウがオープンします。
3. アクセス・プラン・グラフを表示できるのは、 EXPLAIN スナップショットが定義されているステートメントだけです。該当するステートメントは、**Explain Snapshot** 列の YES 値で識別できます。 Query Number 1 という名前の項目をダブルクリックします (**Query Number** 列が見つからない場合は、画面を右にスクロールしてください)。ステートメントの「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。

注: グラフは下から上の方向で読み取ります。照会の最初のステップがグラフの末尾にリストされ、最後のステップが先頭にリストされます。

アクセス・プラン・グラフ内の記号の読み取り

アクセス・プラン・グラフには、アクセス・プランの構造がツリー形式で表示されます。ツリーのノード は、以下のものを表しています。

- 表 (長方形)
- 索引 (ひし形)
- 演算子 (8 角形) TQUEUE 演算子 (平行四辺形)
- 表関数 (6 角形)

演算子の場合、演算子タイプの右の大括弧で囲まれた数値は、各ノードの固有 ID を表します。演算子タイプの下の数値は、累積のコストを表します。

ズーム・スライダーを使用してグラフの部分を拡大する

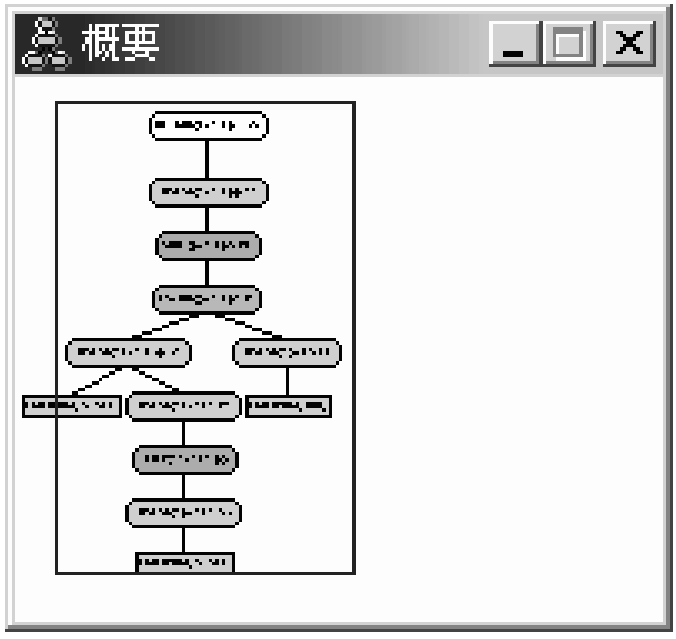
アクセス・プラン・グラフを表示すると、グラフ全体が表示されるため、 ノードが小さすぎて区別できないことがあります。

「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウでは、**ズーム・スライダー**を使用して、グラフの各部を拡大することができます。手順は以下のとおりです。

1. グラフの左側のズーム・スライダー・バー内の、小さなスクロール・ボックスの上にマウス・ポインターを移動します。
2. スライダーを左マウス・ボタンで押したままドラッグし、グラフの拡大率が適切になるように調整します。

グラフの別の部分を表示するには、スクロール・バーを使用します。

大きくて複雑なアクセス・プラン・グラフを表示するには、「グラフの概要 (Graph Overview)」ウィンドウを使用します。このウィンドウを使用することで、表示しているグラフのどの部分をみるのか、ズームインする、あるいはグラフをスクロールすることができます。ズーム・ボックス内のセクションはアクセス・プランに表示されます。



グラフをスクロールするには、「グラフの概要 (Graph Overview)」ウィンドウの強調表示域にマウス・ポインターを置き、 mouse button 1 を押したまま、アクセス・プラン・グラフの表示したい部分までマウスを移動します。

グラフ内のオブジェクトに関する詳細の取得

アクセス・プラン・グラフでは、オブジェクトに関する詳細情報にアクセスできます。以下の情報を表示できます。

- 以下のオブジェクトに関するシステム・カタログ統計。
  - 表、索引、および表関数
  - コスト、プロパティー、入力引数など、 演算子に関する情報
  - 組み込み関数またはユーザー定義関数
  - 表スペース
  - SQL ステートメントで参照される列
- 構成パラメーターおよびバインド・オプション (最適化パラメーター)。

表、索引、および表関数に関する統計の取得

グラフに含まれている単一表 (長方形)、索引 (ひし形)、または表関数 (6 角形) に関するカタログ統計を表示するには、対応するノードをダブルクリックします。選択されたオブジェクトに関する「統計 (Statistics)」ウィンドウがオープンし、スナップショットの作成時に有効だった統計や現在システム・カタログ表に存在している統計に関する情報が表示されます。

グラフに含まれている複数 の表、索引、または表関数に関するカタログ統計を表示するには、目的のノードをクリックし (これにより選択状態になります)、**「Node」-> 「Show statistics」**を選択します。選択されたオブジェクトごとに、「統計 (Statistics)」ウィンドウがオープンします。 (これらのウィンドウは積み重なっている場合があります。ウィンドウにアクセスするには、ウィンドウをドラッグ・アンド・ドロップする必要があるかもしれません。)

EXPLAIN された 列の **STATS_TIME** 項目の値が**「更新されていない統計 (Statistics not updated)」** になっている場合、オプティマイザーがアクセス・プランを作成した時点で統計は存在していませんでした。そのため、アクセス・プランを作成するのに特定の統計が必要だった場合は、デフォルトの統計が使用されました。オプティマイザーでデフォルトの統計が使用された場合は、 Explain 列に **「デフォルト (default)」** が表示されます。

グラフ内の演算子に関する詳細の取得

単一の演算子 (octagon) のカタログ統計を表示するには、演算子のノードをダブルクリックしてください。選択された演算子ごとに「演算子 (Operator)」ウィンドウがオープンし、以下の情報が表示されます。

- 推定累積コスト (I/O、CPU 命令、合計コスト)
- これまでのカーディナリティー (検索された行の推定数)
- プラン中に、アクセスまたは結合された表。
- それらの表で、すでにアクセスされた列。
- すでに適用された述部 (推定される選択性を含む)
- 各演算子の入力引数。

複数 の演算子の詳細を表示するには、目的の演算子をクリックし (これにより選択状態になります)、**「Node」->「詳細の表示 (Show Details)」**を選択します。選択されたオブジェクトごとに、「統計 (Statistics)」ウィンドウがオープンします。 (これらのウィンドウは積み重なっている場合があります。ウィンドウにアクセスするには、ウィンドウをドラッグ・アンド・ドロップする必要があるかもしれません。)

関数に関する統計の取得

組み込み関数およびユーザー定義関数に関するカタログ統計を表示するには、**「Statement」->「Show statistics」-> 「Functions」**を選択します。「Functions」ウィンドウで 1 つまたは複数の項目を選択し、**「OK」**をクリックしてください。選択された関数ごとに、「Function Statistics」ウィンドウがオープンします。

表スペースに関する統計の取得

表スペースのカタログ統計を表示するには、**「Statement」->「Show statistics」 ->「Table spaces」**を選択します。「Table Spaces」ウィンドウで 1 つまたは複数の項目を選択し、**「OK」**をクリックしてください。選択された表スペースごとに「Table Space Statistics」ウィンドウがオープンします。

SQL ステートメント内の列に関する統計の取得

SQL ステートメントで参照される列に関する統計を取得するには、以下の手順に従ってください。

1. アクセス・プラン・グラフ内の表をダブルクリックします。「表統計 (Table Statistics)」ウィンドウがオープンします。
2. **「参照列 (Referenced Columns)」**ボタンをクリックします。「Referenced Columns」ウィンドウがオープンし、表の列がリストされます。
3. 1 つ以上の列を選択して、「**OK**」をクリックします。選択された列ごとに、「Referenced Column Statistics」ウィンドウがオープンします。

構成パラメーターおよびバインド・オプションに関する情報の取得

構成パラメーターおよびバインド・オプション (最適化パラメーター) に関する情報を表示するには、「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウで **「Statement」->「Show optimization parameters」**を選択してください。「Optimization Parameters」ウィンドウがオープンし、スナップショットの作成時に有効だったパラメーター値や現在の値に関する情報が表示されます。

グラフの概観の変更

グラフの外観を変更するには、以下の手順に従ってください。

1. 「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウで、**「表示 (View)」->「設定 (Settings)」**を選択します。「Access Plan Graph Settings」ノートブックがオープンします。
2. 背景色を変更するには、「グラフ (Graph)」タブを選択します。
3. さまざまな演算子の色を変更するには、「基本 (Basic)」、「拡張 (Extend)」、「更新 (Update)」、「その他 (Miscellaneous)」タブを選択します。
4. 表の色、索引、表関数ノードを変更するには、「オペランド (Operand)」タブを選択します。
5. 演算子ノードに表示する情報のタイプ (つまり、 コスト、またはこれまでに検索された行数の推定数を表すカーディナリティー) を指定するには、「演算子 (Operator)」タブを選択します。
6. 表ノードにスキーマ名またはユーザー ID のどちらを表示するかを指定するには、「オペランド (Operand)」タブを選択します。
7. ノードを 2 次元で表示するか 3 次元で表示するかを指定するには、「ノード (Node)」タブを選択します。
8. 選択したオプションでグラフを更新し、設定を保管するには、**「適用 (Apply)」**をクリックします。

次のステップ

単一パーティション・データベース環境で操作している場合は、 13 ページの『レッスン 3. 単一パーティション・データベース環境でアクセス・プランを改善する』に進み、さまざまなチューニング・アクティビティーによりアクセス・プランがどのように変化および向上するかを学習します。

パーティション・データベース環境で操作している場合は、 14 ページの『索引または統計を使用しない照会の実行』に進み、さまざまなチューニング・アクティビティーによりアクセス・プランがどのように変化および向上するかを学習します。

レッスン 3. 単一パーティション・データベース環境でアクセス・プランを改善する

このレッスンでは、さまざまなチューニング・アクティビティーを実行したときに、アクセス・プランと、基本照会の関連ウィンドウがどのように変化するかを学習します。**runstats** コマンドを実行して、適切な索引を追加すると、どんな単純な照会でもアクセス・プランの推定合計コストが向上することを、一連の例と図により学習します。

Visual Explain で経験したとおり、他の方法でも照会をチューニングすることができます。

アクセス・プラン・グラフの処理

4 つのサンプル EXPLAIN スナップショットを使用して、チューニングがデータベース・パフォーマンスの重要な部分であることを学習します。

EXPLAIN スナップショットに関連付けられた照会には、 1 から 4 までの番号が付けられています。どの照会も同じ SQL ステートメント (レッスン 1 を参照) を使用します。

```
SELECT S.ID,S.NAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
  O.DEPTNUMB = S.DEPT AND
  S.JOB <> 'Mgr' AND
  S.SALARY+S.COMM > ALL( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

ただし、照会のそれぞれの反復では、前回の反復より多くのチューニング・アクティビティーが必要です。たとえば、Query 1 ではパフォーマンスのチューニングを実行しませんでしたが、 Query 4 では最大数のチューニングを実行しました。それぞれの照会の違いは、以下のとおりです。

Query 1
: 索引または統計を使用しない照会の実行

Query 2
: 照会で表および索引に関する現在の統計の収集

Query 3
: 照会で複数の表を結合するための列に対する索引の作成

Query 4

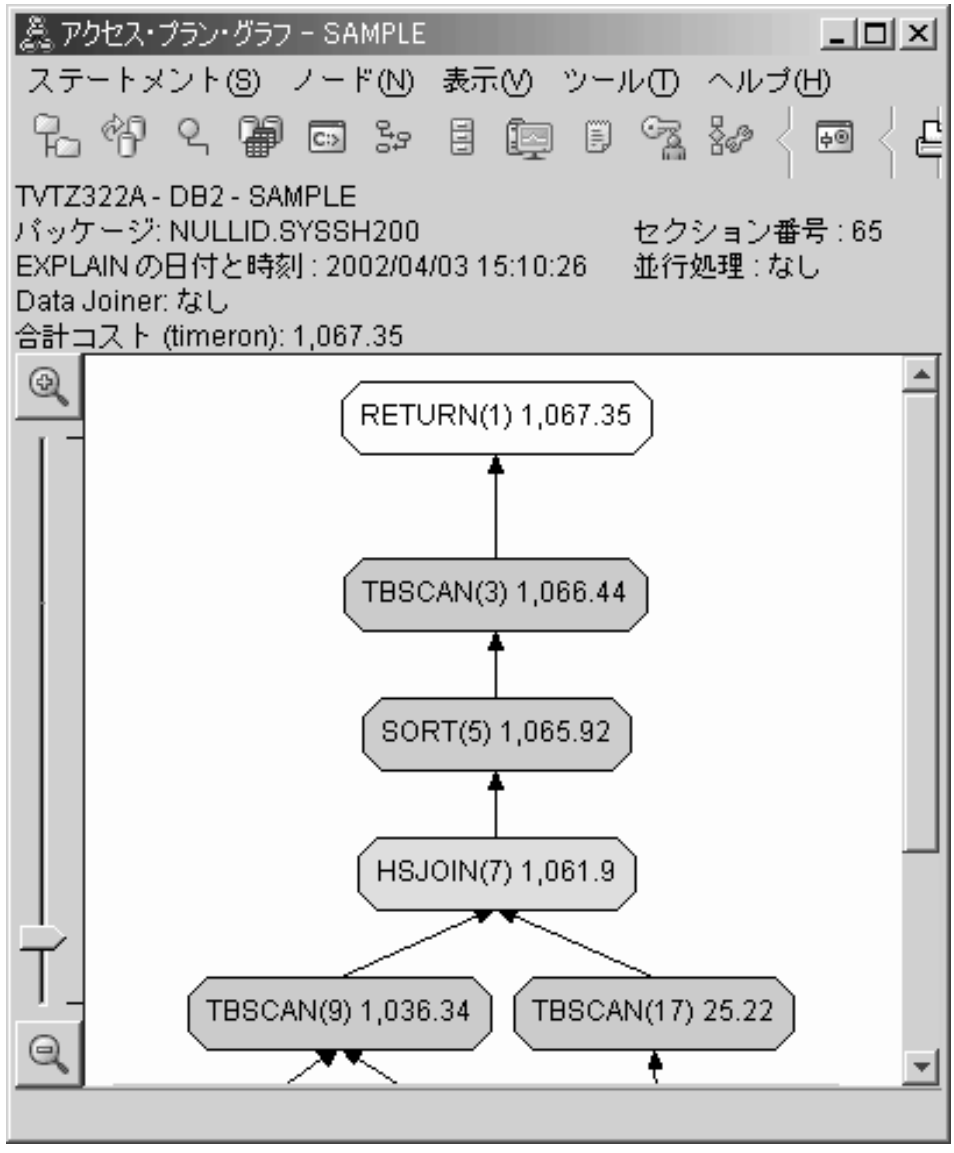
表の列に対する追加の索引の作成

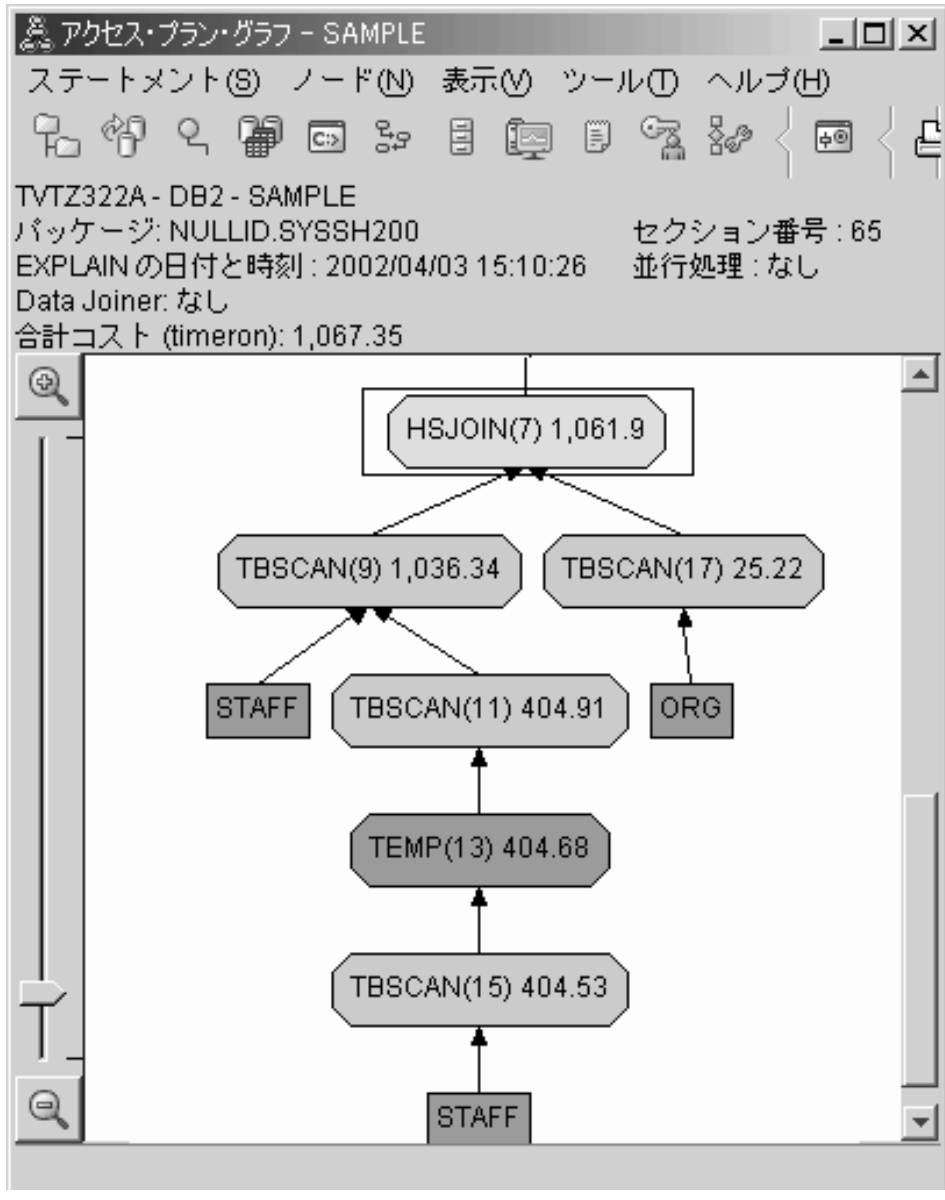
索引または統計を使用しない照会の実行

この例では、索引または統計を使用しない SQL 照会のためにアクセス・プランを作成しました。

この照会 (Query 1) のアクセス・プラン・グラフを表示するには、以下の手順に従ってください。

1. Control Center で、 SAMPLE データベースが表示されるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューから**「EXPLAIN されたステートメントのヒストリーの表示 (Show explained statements history)」**を選択します。「EXPLAIN されたステートメントのヒストリー (Explained Statement History)」ウィンドウがオープンします。
3. Query Number 1 という名前の項目をダブルクリックします (**Query Number** 列が見つからない場合は、画面を右にスクロールしてください)。ステートメントの「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。


アクセス・プラン・グラフ - SAMPLE
ステートメント(S) ノード(N) 表示(V) ツール(T) ヘルプ(H)
TVTZ322A - DB2 - SAMPLE
パッケージ: NULLID.SYSSH200
セクション番号 : 65
EXPLAIN の日付と時刻 : 2002/04/03 15:10:26
並行処理 : なし
Data Joiner: なし
合計コスト (timeron): 1,067.35
RETURN(1) 1,067.35
TBSCAN(3) 1,066.44
SORT(5) 1,065.92
HSJOIN(7) 1,061.9
TBSCAN(9) 1,036.34
TBSCAN(17) 25.22



以下の質問に答えることにより、照会を改善する方法を学習します。

1. 照会で使用する各表の最新統計が存在していますか?

   照会で参照する各表の現在の統計が存在しているかどうかを確認するには、アクセス・プラン・グラフでそれぞれのターゲット・ノードをダブルクリックしてください。対応する「表統計 (Table Statistics)」ウィンドウがオープンされ、 **Explain** の

情報列の **STATS_TIME** 行に、スナップショットの作成時に統計が収集されていないことを意味する「更新されていない統計 (Statistics not updated)」というメッセージが表示されます。

現在の統計が存在していない場合、オプティマイザーはデフォルトの統計を使用しますが、この統計は実際の統計とは異なる可能性があります。「表統計 (Table Statistics)」ウィンドウの **Explain 時の情報**列では、デフォルトの統計に「デフォルト (Default)」という ID が表示されます。

ORG 表に関する「表統計 (Table Statistics)」ウィンドウの情報によると、オプティマイザーではデフォルトの統計が使用されました (EXPLAIN された値の隣の値がそのことを表している)。スナップショットを作成したときに実際の統計は使用可能でなかったため (**STATS_TIME** 行の値がそのことを表している)、デフォルトの統計が使用されました。

2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

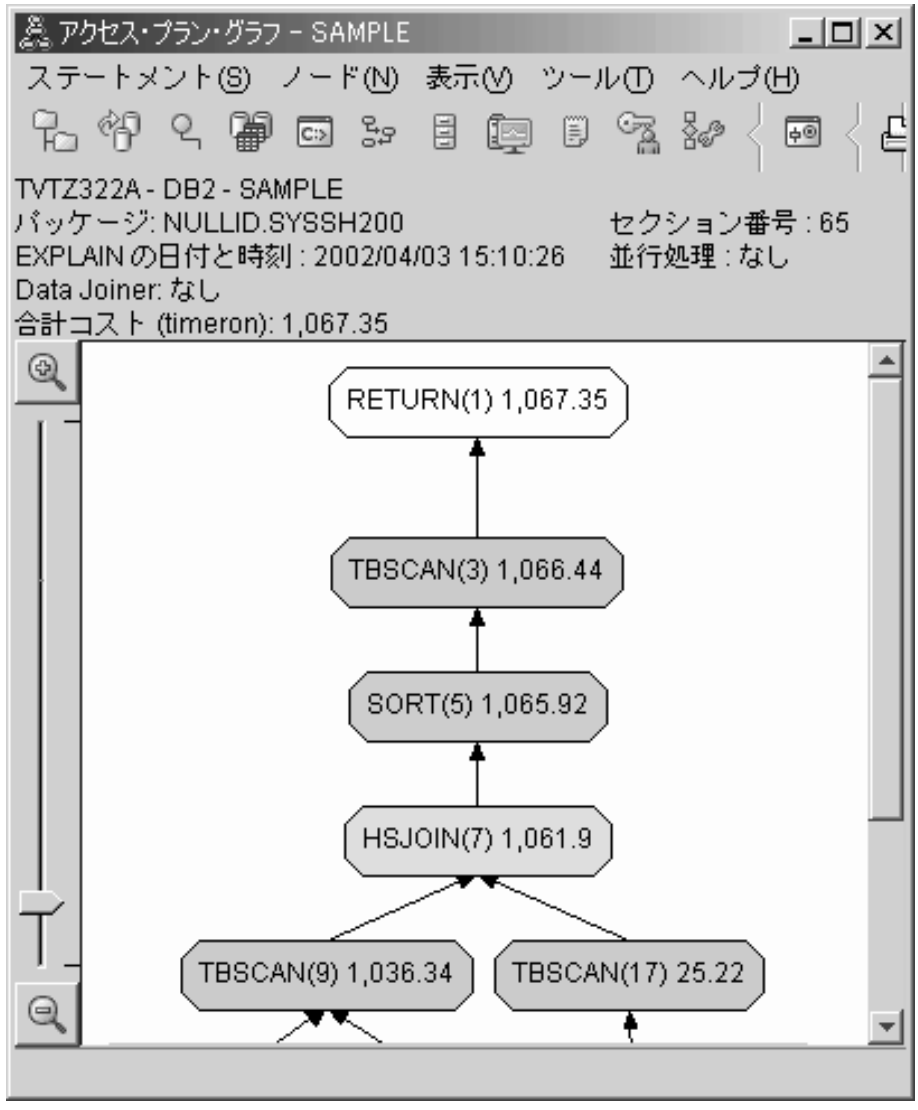
   このアクセス・プランには、索引スキャンではなく表スキャンが含まれています。表スキャンは 6 角形で表示されており、 TBSCAN というラベルが付けられています。索引スキャンが使用された場合、索引スキャンはひし形で表示され、 IXSCAN というラベルが付けられます。抽出するデータの量が少ない場合は、表スキャンを実行するより、表に対して作成された索引を使用した方がコスト的に効率的です。

3. このアクセス・プランの効果はどれほどですか?

   アクセス・プランの効果を評価するには、アクセス・プランが実際の統計に基づいていなければなりません。これまで、オプティマイザーではデフォルトの統計を使用しているため、プランの効果を評価することはできません。

   通常は、後で変更済みのアクセス・プランと比較できるように、アクセス・プランの推定合計コストを書き留めておく必要があります。なお、各ノードでリストされているコストは、照会の最初のステップからそのノードまで (ノード自体を含む) の累積値です。

   「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウでは、グラフの先頭の **RETURN (1)** に示されているとおり、合計コストは約 1,067 timeron です。推定合計コストは、ウィンドウの再上部にも表示されます。



4. 次のステップは何ですか?

   Query 2 では、**runstats** を実行した後の、基本照会用のアクセス・プランを検証します。 **runstats** コマンドを実行すると、オプティマイザーに、照会で参照するすべての表に関する現在の統計が提供されます。

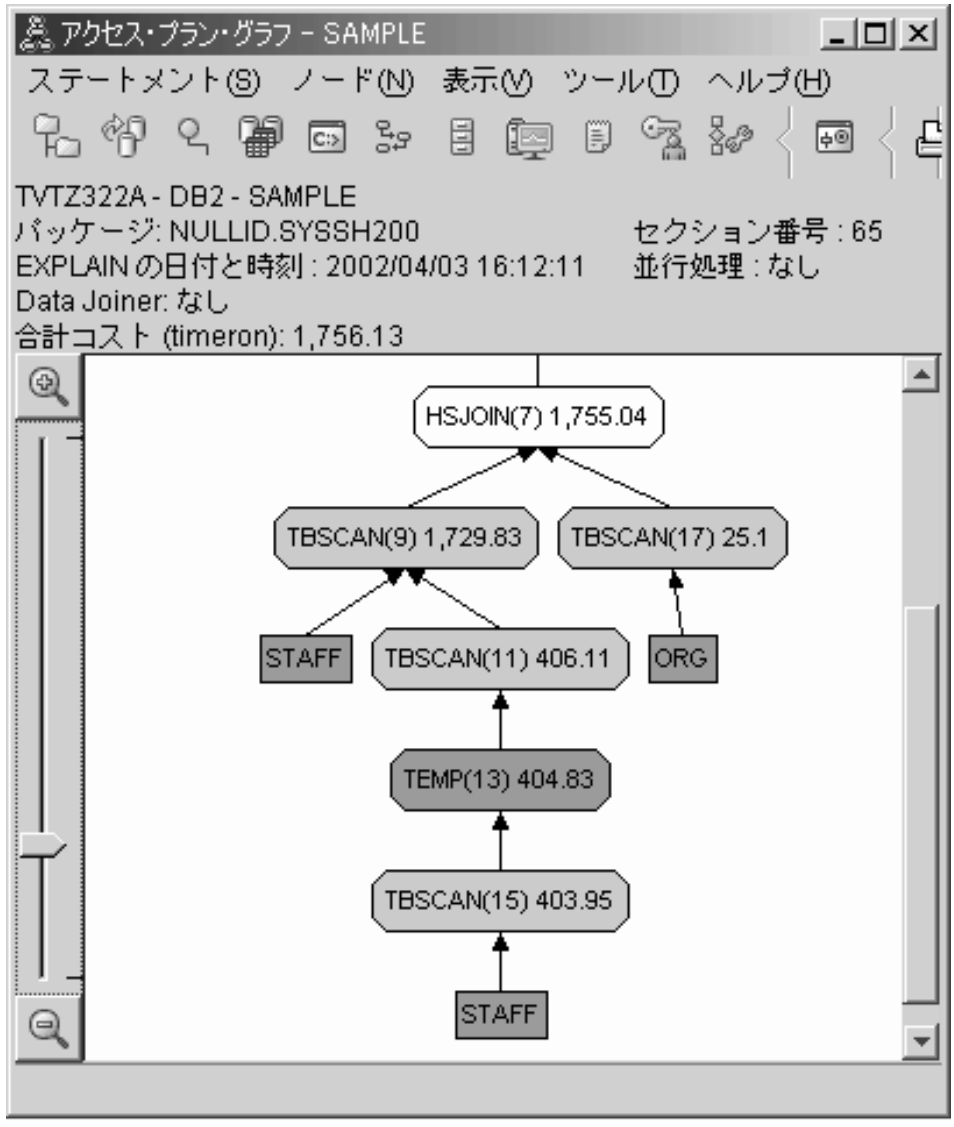
runstats を使用した表および索引の現在の統計の収集

この例では、**runstats** コマンドを実行して現在の統計を収集し、Query 1 で説明したアクセス・プランをビルドします。

強くお薦めするのは、**runstats** コマンドを使用して、表および索引に関する現在の統計を収集することです。最後に **runstats** コマンドを実行してから、大きな更新が発生した場合や新しい索引が作成されている場合は特にその必要があります。これにより、オプティマイザーは最も正確な情報に基づいて、最も効果的なアクセス・プランを決定できます。現在の統計を利用できないと、オプティマイザーは不正確なデフォルト統計に基づいて効果的でないアクセス・プランを選択してしまう可能性があります。

表の更新を行った後 で、 **runstats** を使用するようにしてください。そうしないと、オプティマイザーが表を空とみなす可能性があります。この問題は、「演算子詳細 (Operator Details)」ウィンドウのカーディナリティーが 0 である場合に明らかとなります。この場合、表更新を完了してから、 **runstats** コマンドを再実行し、関係する表の EXPLAIN スナップショットを再作成してください。

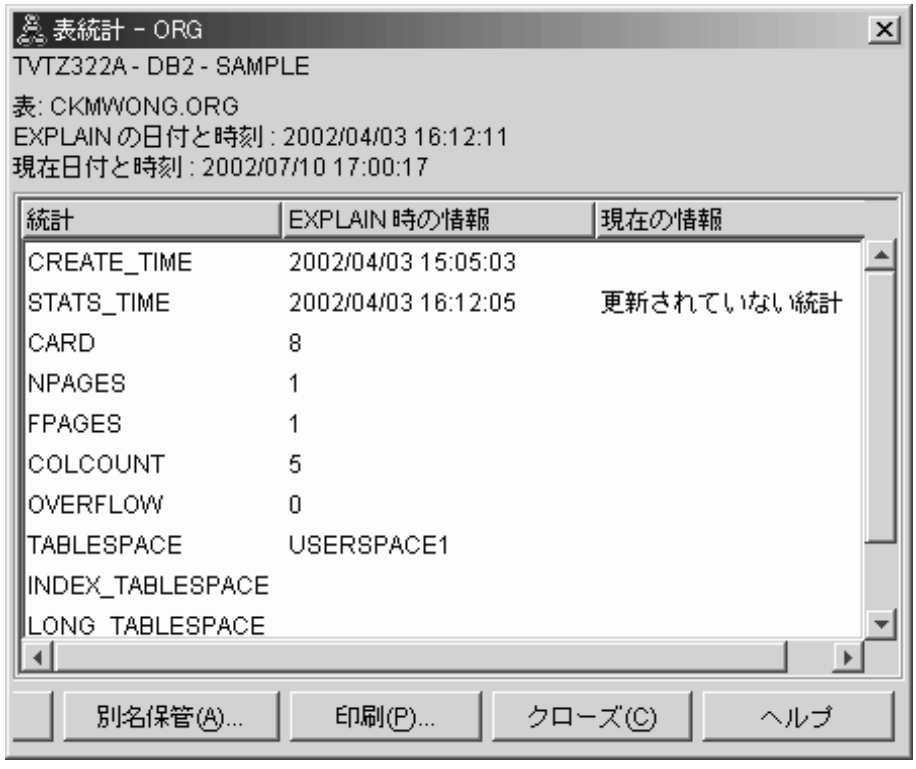
この照会 (Query 2) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウで、Query Number 2 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。



以下の質問に答えることにより、照会を改善する方法を学習します。

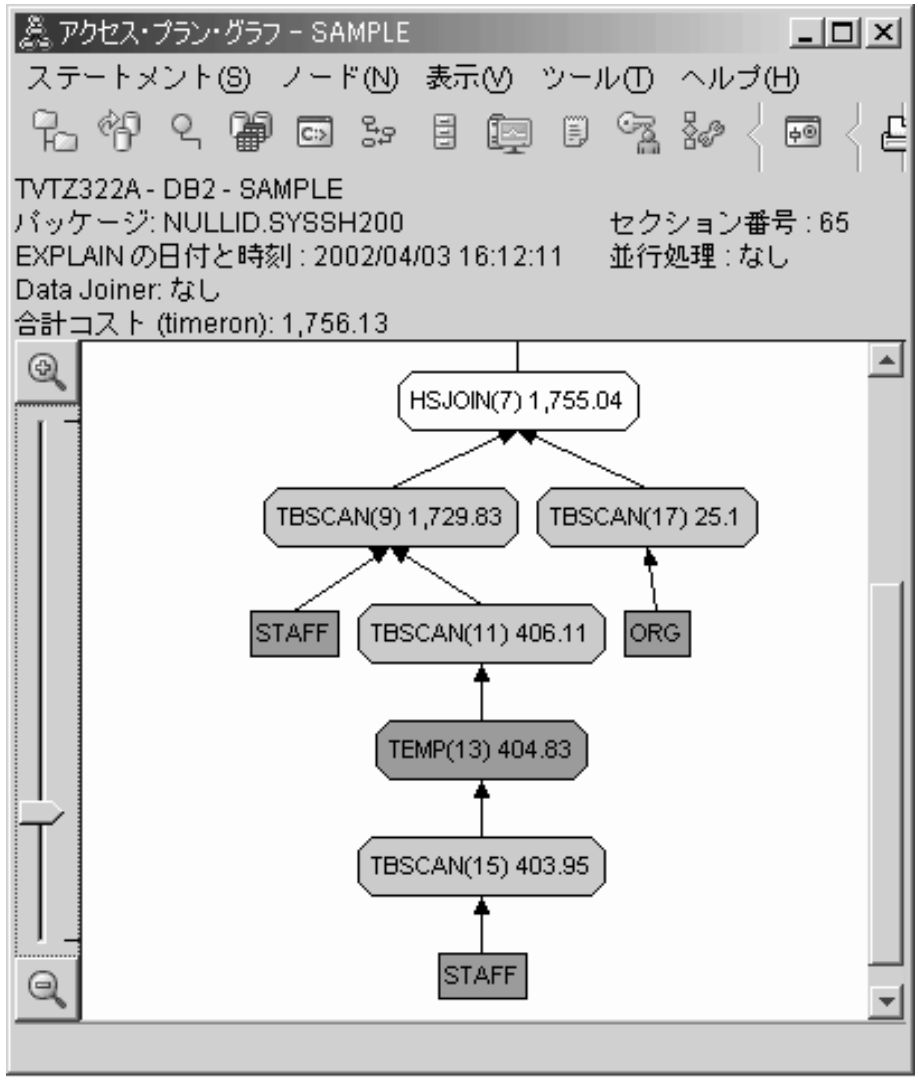
1. 照会で使用する各表の最新統計が存在していますか?

   ORG 表の「表統計 (Table Statistics)」ウィンドウは、オプティマイザーで実際の統計が使用されたことを示しています (**STATS_TIME** 値は、統計が実際に収集された時刻です)。統計が正確かどうかは、 **runstats** コマンドを実行した後に、表の内容が大幅に変更されたかどうかに依存しています。



2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   Query 1 と同様、Query 2 のアクセス・プランでは索引スキャン (IXSCAN) ではなく、表スキャン (TBSCAN) を使用します。照会で参照する列に対して索引が定義されていないため、現在の統計が存在している場合でも、索引スキャンは実行されません。照会を改善する 1 つの方法は、オプティマイザーのために、表の結合で使用する列 (つまり、結合述部に指定する列) に対して索引を定義する方法です。この例では、最初のマージ・スキャン結合 HSJOIN (7) を使用します。



HSJOIN (7) 演算子の「オペレーター詳細 (Operator Details)」ウィンドウで、**「入力引き数 (Input arguments)」**の下の**「結合述部 (Join predicates)」**セクションを確認します。この結合操作で使用する列は、**テキスト**列にリストされます。この例の場合、列の名前は DEPTNUMB および DEPT です。

3. このアクセス・プランの効果はどれほどですか?

   アクセス・プランで最新の統計を使用すると、実際の値に近い推定コスト (単位は timeron) を得ることができます。 Query 1 の推定コストはデフォルトの統計に基づいていたため、これら 2 つのアクセス・プラン・グラフの推定コストを比較して、どちらがより効果的かを判断することはできません。この場合、コストが高いか低いかは関係ありません。効果性について有効な値を得るには、実際の統計に基づいたアクセス・プランのコストを比較する必要があります。

4. 次のステップは何ですか?

   Query 3 では、DEPTNUMB 列および DEPT 列に索引を追加したときの効果を検証します。結合述部で使用される列に対して索引を追加すると、パフォーマンスが向上することがあります。

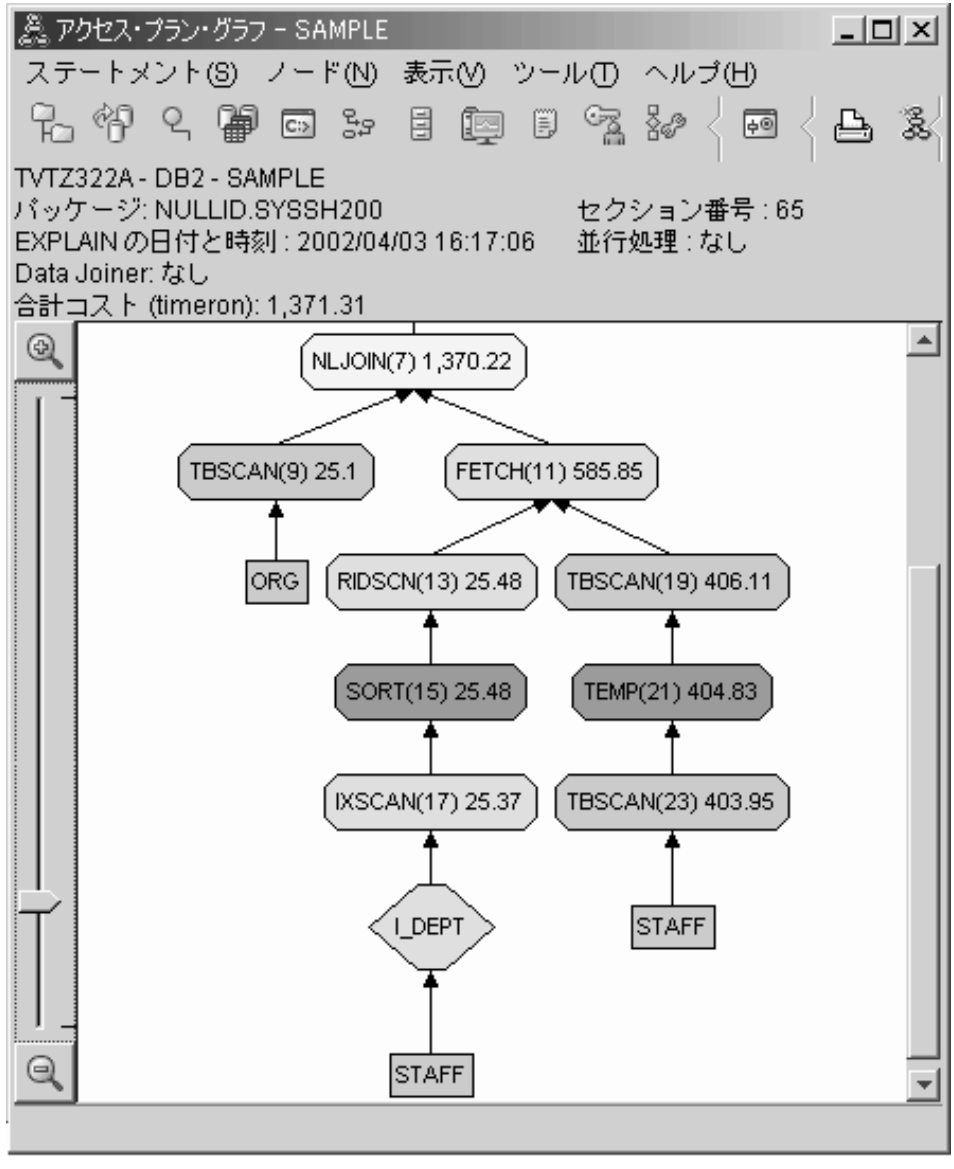
照会で複数の表を結合するための列に対する索引の作成

この例では、Query 2 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の DEPT 列と、ORG 表の DEPTNUMB 列に索引を作成します。

注: バージョン 8 では、推奨されている索引を Workload Performance ウィザードで作成することができます。

この照会 (Query 3 のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウで、Query Number 3 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。

注: DEPTNUM のために索引が作成されましたが、オプティマイザーはその索引を使用しませんでした。

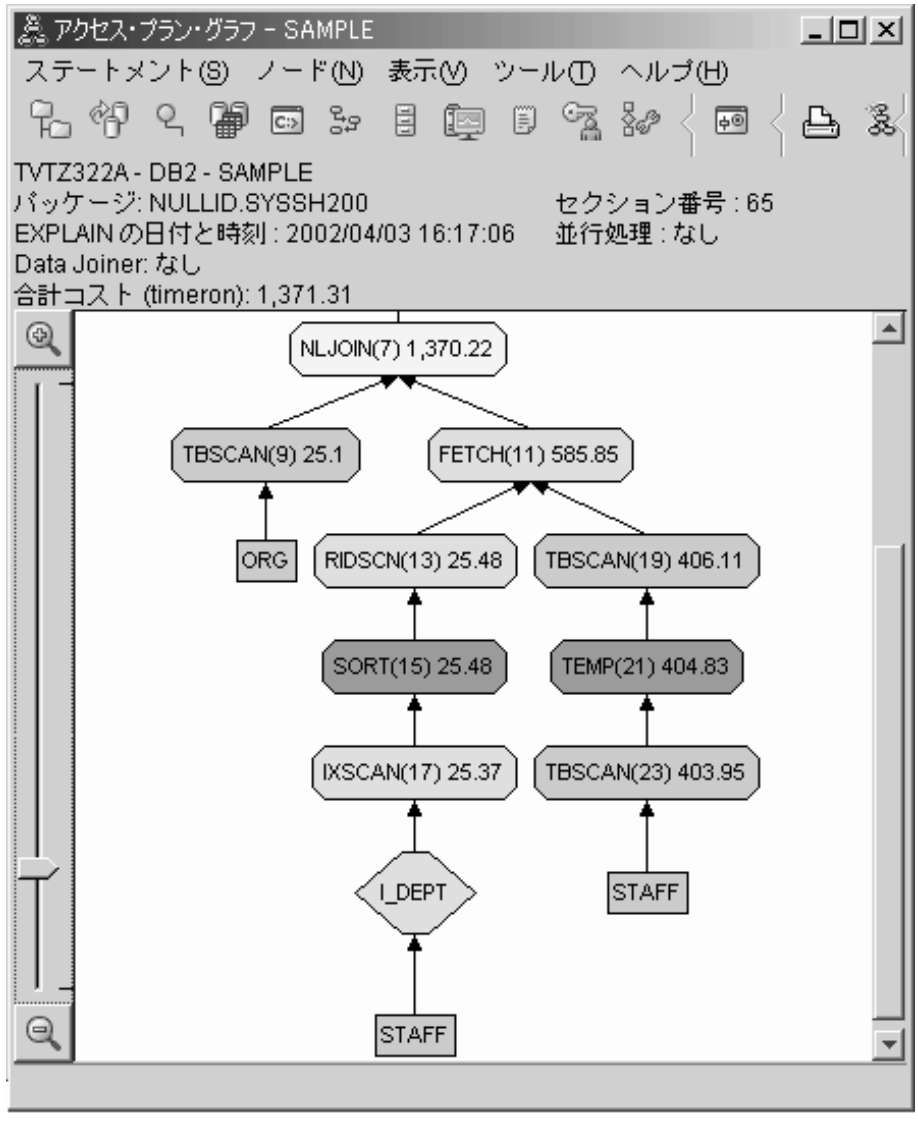


以下の質問に答えることにより、照会を改善する方法を学習します。

1. 索引の追加によりアクセス・プランはどのように変化しましたか?

   Query 2 で使用されたマージ・スキャン結合 HSJOIN (7) の代わりに、 ネスト・ループ結合 NLJOIN (7) が使用されるようになりました。ネスト・ループ結合は TEMPORARY 表を必要としないため、ネスト・ループ結合を使用すると推定コストは減少します。

   STAFF 表のすぐ上に、新しいひし形のノード **I_DEPT** が追加されました。このノードは、DEPT に作成された索引を表し、取得する行を判別するために、オプティマイザーが表スキャンではなく索引スキャンを使用したことを意味します。



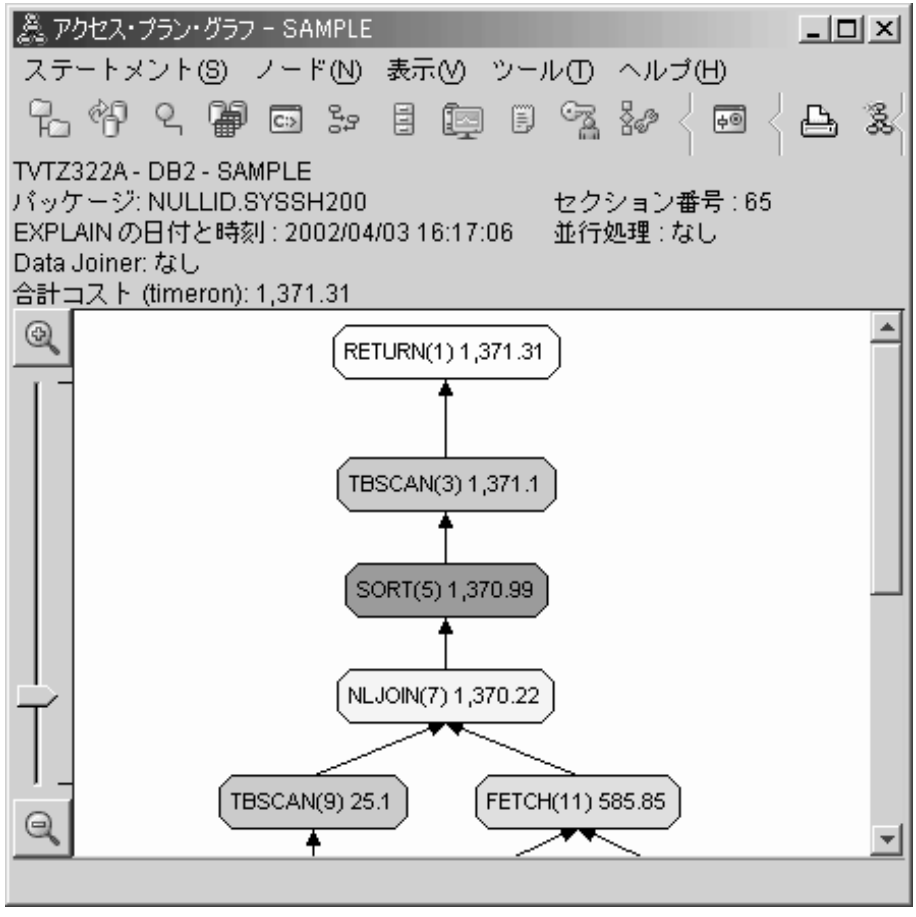
アクセス・プラン・グラフのこの部分では、DEPT 列に新しい索引 (I_DEPT) が作成されたこと、および STAFF 表にアクセスするのに IXSCAN (17) が使用されたことに注目してください。 Query 2 では、STAFF 表にアクセスするのに表スキャンを使用しました。

2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   索引を追加した結果、STAFF 表にアクセスするのに IXSCAN ノード、IXSCAN (17) を使用するようになりました。 Query 2 では、索引が定義されていなかったため表スキャンを使用しました。

FETCH ノード、FETCH (11) は、オプティマイザーが、列 DEPT を取得するために索引スキャンを使用し、 STAFF 表から追加の列を取得するためにポインターとして索引を使用することを示しています。この場合、索引スキャンとフェッチ操作を組み合わせて使用すると、以前のアクセス・プランで使用した完全表スキャンよりもコストは減少します。

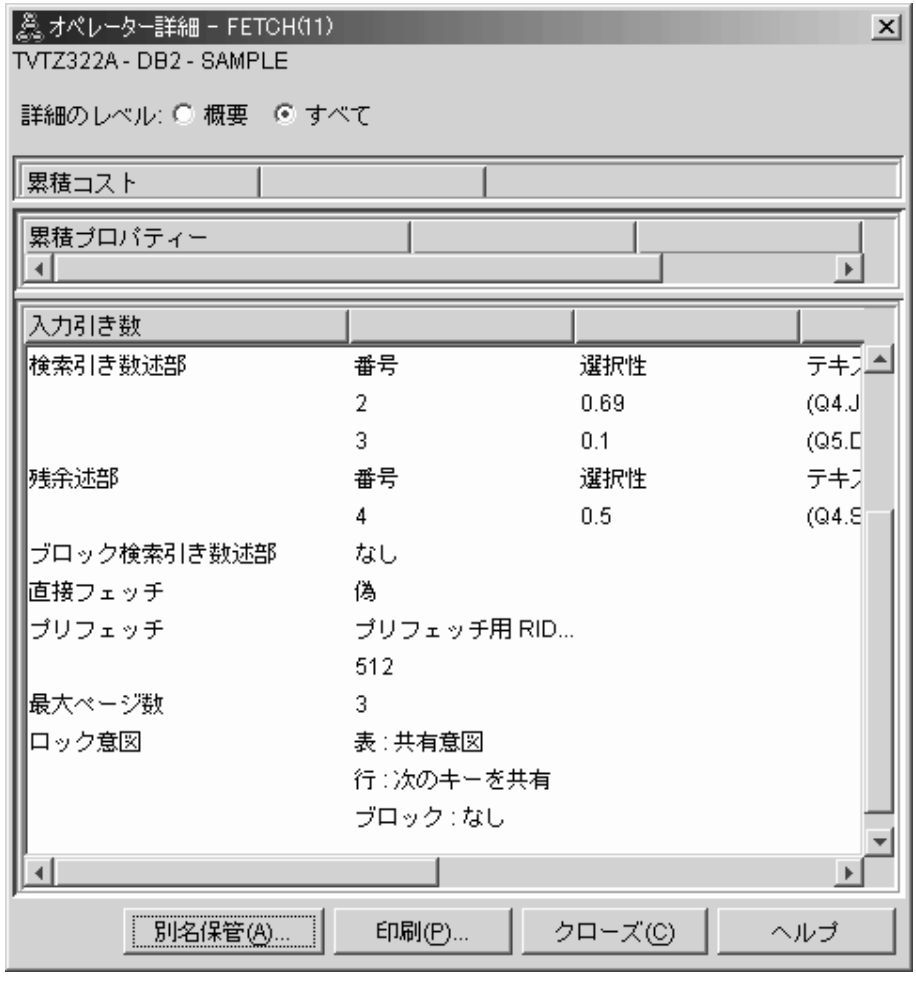
注: STAFF 表のためのノードは 2 つ含まれていますが、これは DEPT 表の索引とのリレーションシップと FETCH 操作とのリレーションシップの 2 つを示しています。



この照会のアクセス・プランは、結合述部に含まれる列に索引を作成したときの効果を示しています。索引を作成すると、 ローカル述部の適用がより高速になります。ここでは、この照会の各表に対するローカル述部を調べ、ローカル述部で参照される列に索引を追加するとアクセス・プランにどのような影響があるかを学習します。

FETCH (11) 演算子の「オペレーター詳細 (Operator Details)」ウィンドウで、**「累積プロパティー (Cumulative Properties)」**の下の列を確認します。「述部 (Predicates)」セクションに示されているとおり、このフェッチ操作の述部で使用する列は JOB です。

注: この述部の選択性は、.69 です。つまり、この述部により全体の行の 69% が後続の処理のために選択されます。



FETCH (11) 演算子の「オペレーター詳細 (Operator Details)」ウィンドウは、この操作で使用する列を示しています。 最初の行の**「入力引き数 (Input arguments)」**の下の**「検索された列 (Columns retrieved)」**の隣に、 DEPTNAME がリストされていることを確認できます。

3. このアクセス・プランの効果性はどれほどですか?

   このアクセス・プランの効率は、前の例よりも向上しました。累積コストは、Query 2 では約 1,214 timeron だったのが、 Query 3 では約 755 timeron に減少しました。

   ただし、Query 3 のアクセス・プランは、STAFF 表で定義された索引スキャン IXSCAN (17) および FETCH (11) を示しています。索引スキャンとフェッチ操作を組み合わせて使用すると、完全表スキャンよりもコストは減少しますが、各行を検索するたびに表に 1 回、索引に 1 回アクセスしなければなりません。次に、STAFF 表に対するこの二重アクセスを減らすことを試みます。

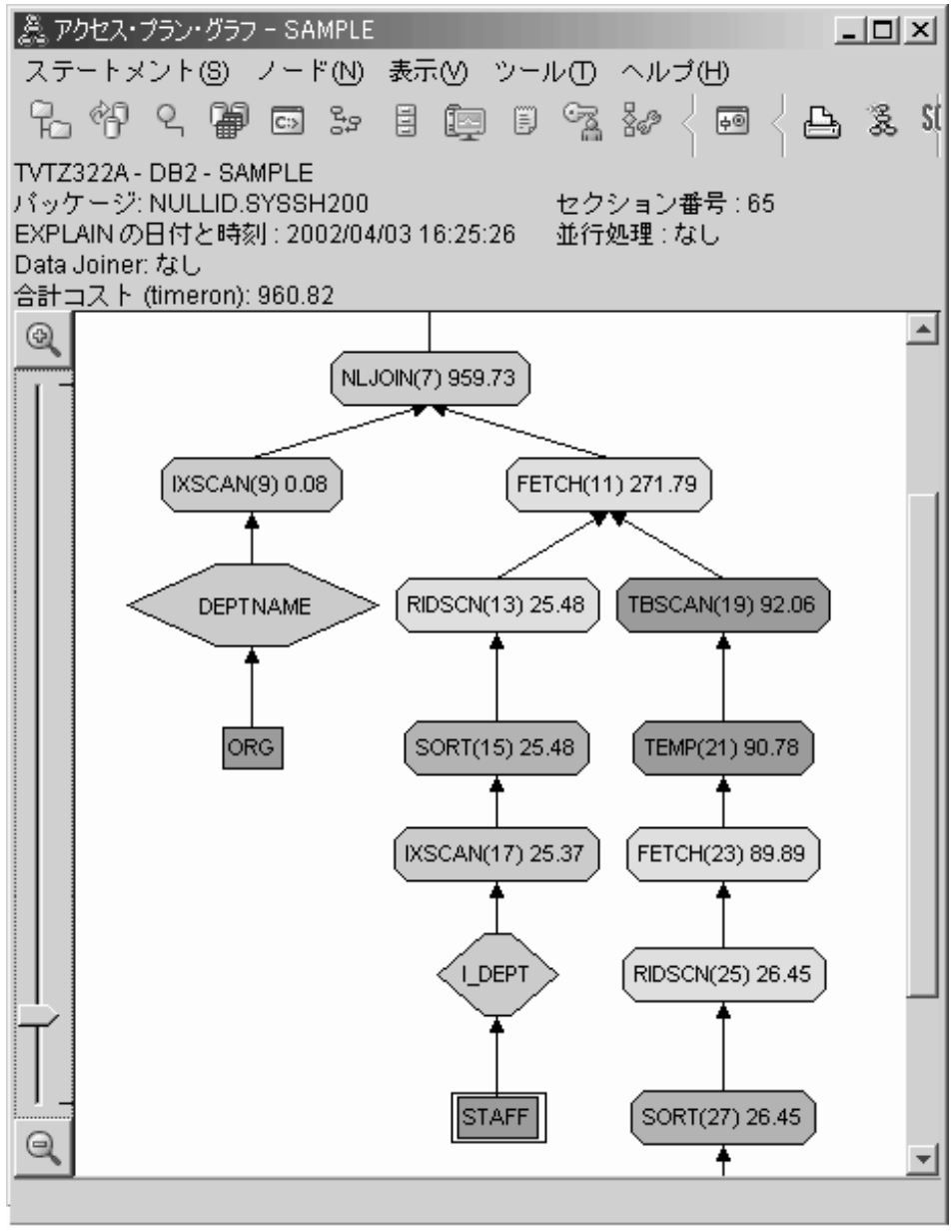
4. 次のステップは何ですか?

   Query 4 では、フェッチ操作を使用せずに、単一の索引スキャンに対するフェッチ操作および索引スキャンの回数を減らします。追加の索引を作成すると、アクセス・プランの推定コストが減少することがあります。

表の列に対する追加の索引の作成

この例では、Query 3 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の JOB 列で索引を作成し、 ORG 表に定義されている既存の索引に DEPTNAME を追加します。 (索引を追加すると、それに伴って追加のアクセスが必要になります。)

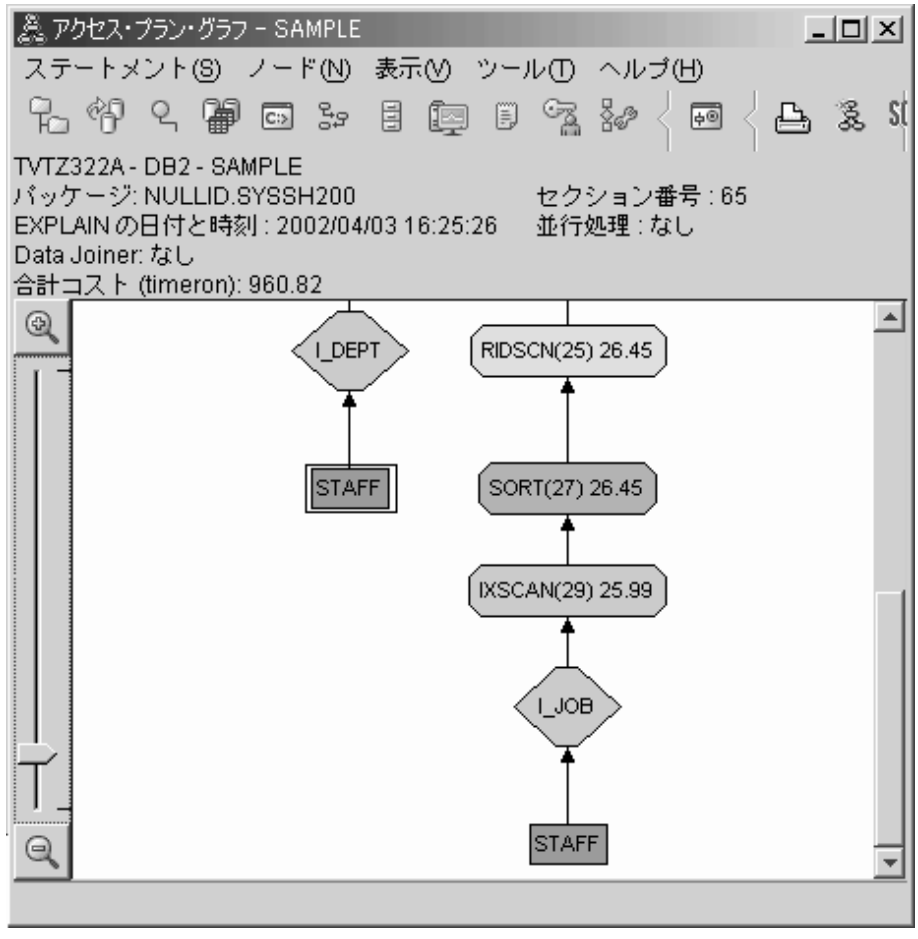
この照会 (Query 4 のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウで、Query Number 4 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。



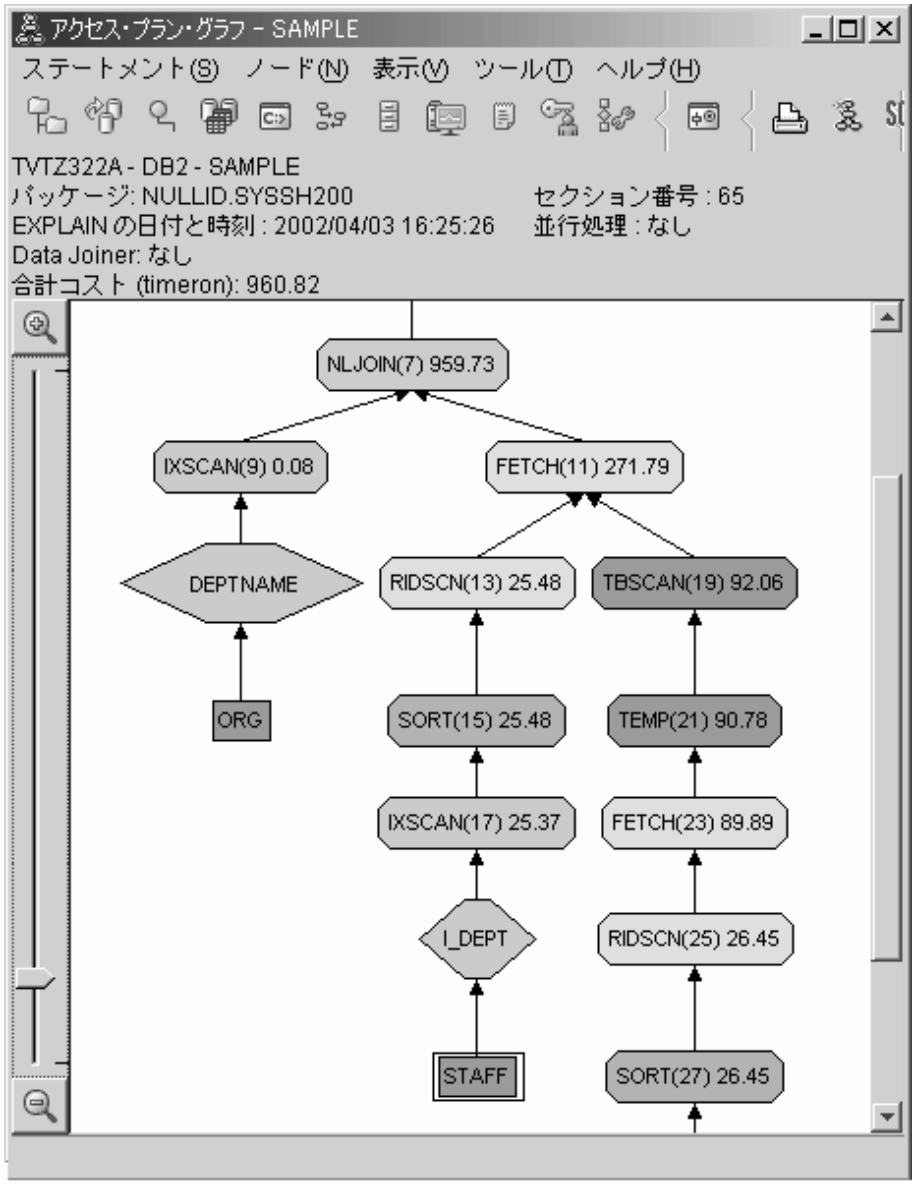
以下の質問に答えることにより、照会を改善する方法を学習します。

1. 追加の索引の作成によりアクセス・プランはどのように変化しましたか?

   オプティマイザーは、STAFF 表の JOB 列に作成された索引 (**I_JOB** というラベルが付いたひし形) を利用して、このアクセス・プランをさらに改善します。



アクセス・プラン・グラフの中間部に表示されている ORG 表で、以前は索引スキャンとフェッチ操作が使用されていたのが、現在は索引スキャンのみ IXSCAN (9) に変化したことに注目してください。 ORG 表の DEPTNAME 列に索引を追加すると、オプティマイザーでは、フェッチ操作のための余分なアクセスを省略することができます。



2. このアクセス・プランの効果はどれほどですか?

   このアクセス・プランの効率は、前の例よりも向上しました。累積コストは、Query 3 では約 1,370 timeron だったのが、 Query 4 では約 959 timeron に減少しました。

次のステップ

パフォーマンスを向上させるためのステップの詳細については、*管理ガイド* を参照してください。次に、Visual Explain に戻って、アクションの影響を評価します。

レッスン 4. パーティション・データベース環境でのアクセス・プランの改善

このレッスンでは、さまざまなチューニング・アクティビティーを実行したときに、アクセス・プランと、基本照会の関連ウィンドウがどのように変化するかを学習します。**runstats** コマンドを実行して、適切な索引を追加すると、どんな単純な照会でもアクセス・プランの推定合計コストが向上することを、一連の例と図により学習します。

Visual Explain で経験したとおり、他の方法でも照会をチューニングすることができます。

アクセス・プラン・グラフの処理

4 つのサンプル EXPLAIN スナップショットを使用して、チューニングがデータベース・パフォーマンスの重要な部分であることを学習します。

EXPLAIN スナップショットに関連付けられた照会には、 1 から 4 までの番号が付けられています。どの照会も同じ SQL ステートメント (レッスン 1 を参照) を使用します。

```
SELECT S.ID,S.NAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
  O.DEPTNUMB = S.DEPT AND
  S.JOB <> 'Mgr' AND
  S.SALARY+S.COMM > ALL( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

ただし、照会のそれぞれの反復では、前回の反復より多くのチューニング・アクティビティーが必要です。たとえば、Query 1 ではパフォーマンスのチューニングを実行しませんでしたが、 Query 4 では最大数のチューニングを実行しました。それぞれの照会の違いは、以下のとおりです。

Query 1
: 索引または統計を使用しない照会の実行

Query 2
: 照会で表および索引に関する現在の統計の収集

Query 3
: 照会で複数の表を結合するための列に対する索引の作成

Query 4

表の列に対する追加の索引の作成

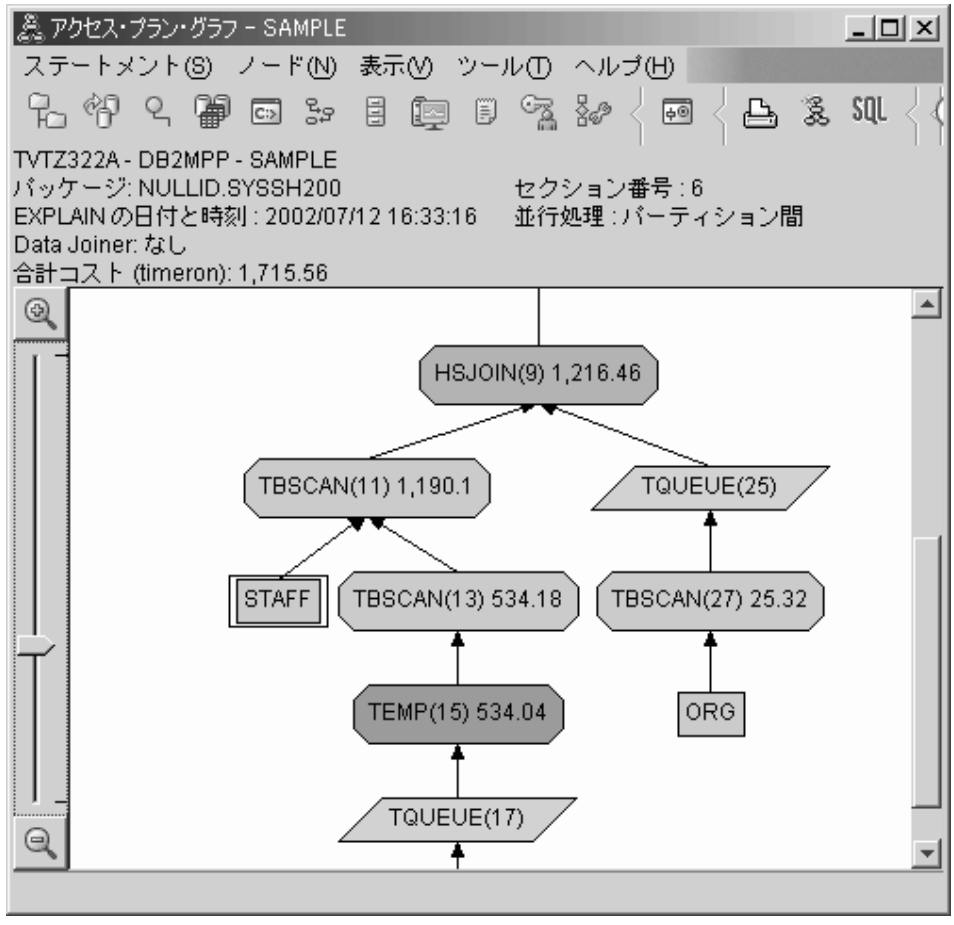
これらの例は、7 個の物理ノードが搭載されている RS/6000 SP マシンで、パーティション間並列処理を使用して生成されました。

索引または統計を使用しない照会の実行

この例では、索引または統計を使用しない SQL 照会のためにアクセス・プランを作成しました。

この照会 (Query 1) のアクセス・プラン・グラフを表示するには、以下の手順に従ってください。

1. Control Centerで、 SAMPLE データベースが表示されるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューから**「EXPLAIN されたステートメントヒストリーの表示 (Show explained statements history)」** を選択します。「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウがオープンします。
3. Query Number 1 という名前の項目をダブルクリックします (**Query Number** 列が見つからない場合は、画面を右にスクロールしてください)。ステートメントの「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。



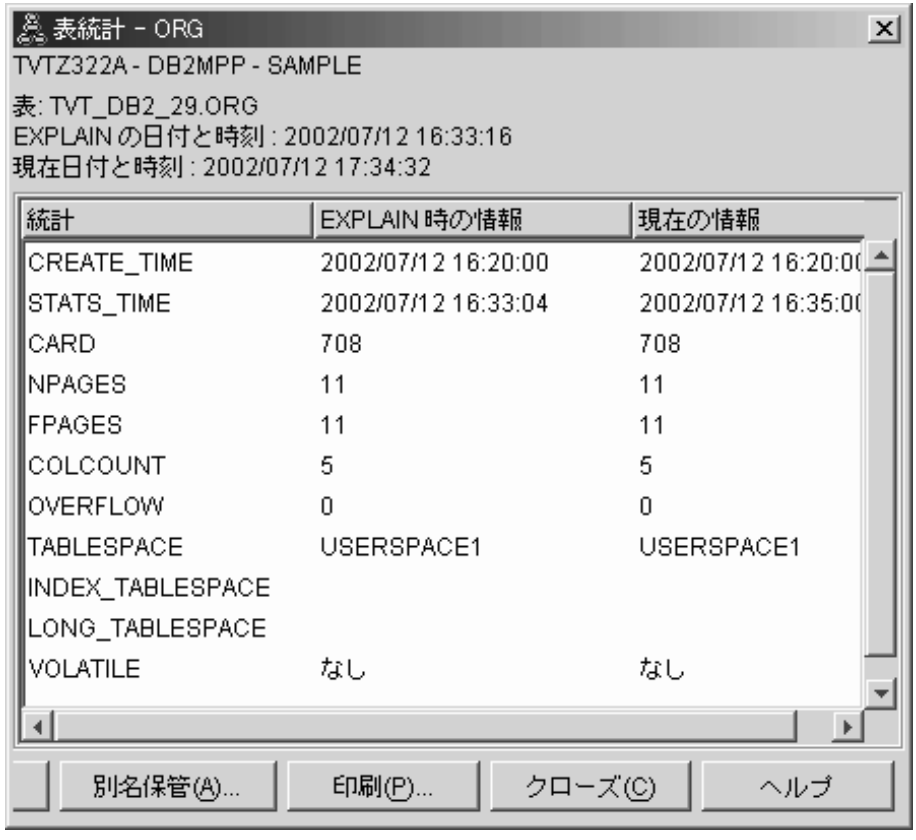
以下の質問に答えることにより、照会を改善する方法を学習します。

1. 照会で使用する各表の最新統計が存在していますか?

   照会で参照する各表の現在の統計が存在しているかどうかを確認するには、アクセス・プラン・グラフでそれぞれのターゲット・ノード をダブルクリックしてください。対応する「表統計 (Table Statistics)」ウィンドウがオープンされ、 **Explain 時の情報**列の **STATS_TIME** 行に、スナップショットの作成時に統計が収集されていないことを意味する「更新されていない統計 (Statistics not updated)」という言葉が表示されます。

   現在の統計が存在していない場合、オプティマイザーはデフォルトの統計を使用しますが、この統計は実際の統計とは異なる可能性があります。「表統計 (Table Statistics)」ウィンドウの **Explain 時の情報**列では、デフォルトの統計に「デフォルト (default)」という ID が表示されます。

ORG 表に関する「表統計 (Table Statistics)」ウィンドウの情報によると、オプティマイザーではデフォルトの統計が使用されました (EXPLAIN された値の隣の値がそのことを表している)。スナップショットを作成したときに実際の統計は使用可能でなかったため (**STATS_TIME** 行の値がそのことを表している)、デフォルトの統計が使用されました。



2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   このアクセス・プランには、索引スキャンではなく表スキャンが含まれています。表スキャンは 6 角形で表示されており、 TBSCAN というラベルが付けられています。索引スキャンが使用済みの場合、索引スキャンはひし形で表示され、 IXSCAN というラベルが付けられます。抽出するデータの量が少ない場合は、表スキャンを実行するより、表に対して作成された索引を使用した方がコスト的に効率的です。

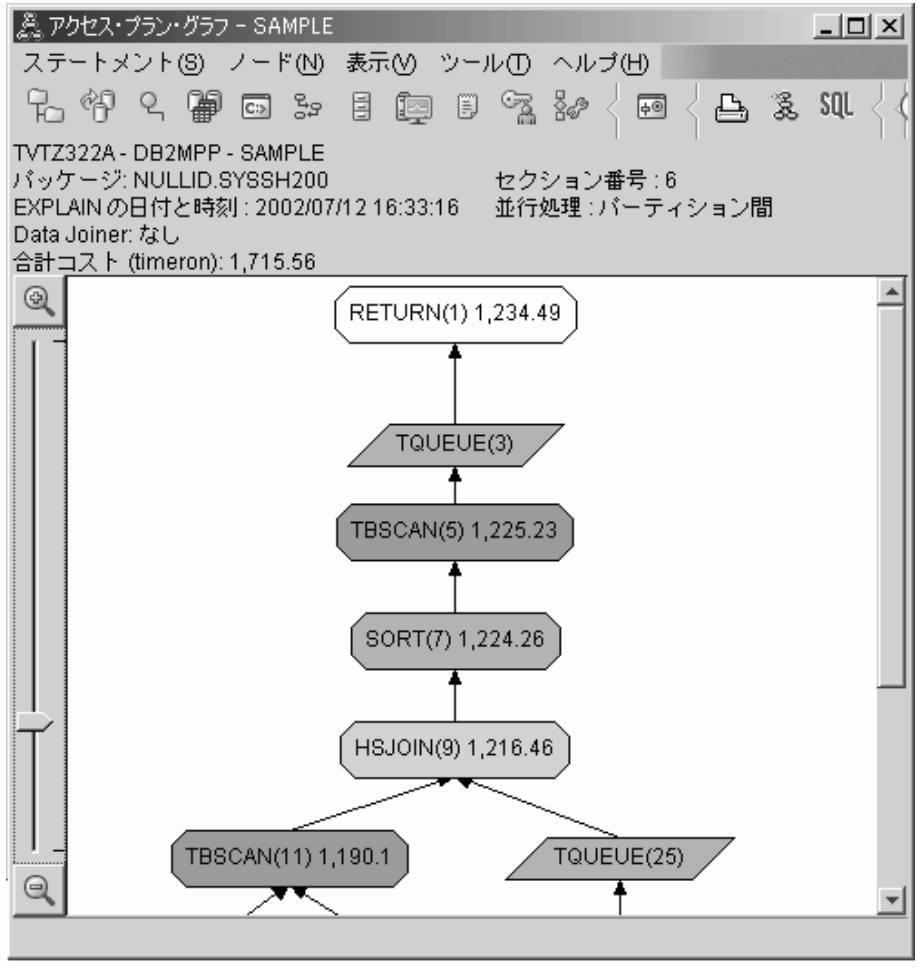
3. このアクセス・プランの効果はどれほどですか?

   アクセス・プランの効果を評価するには、アクセス・プランが実際の統計に基づいていなければなりません。これまで、オプティマイザーではデフォルトの統計を使用しているため、プランの効果を評価することはできません。

通常は、後で変更済みのアクセス・プランと比較できるように、アクセス・プランの推定合計コストを書き留めておく必要があります。なお、各ノードでリストされているコストは、照会の最初のステップからそのノードまで (ノード自体を含む) の累積値です。

注: パーティション・データベースの場合、この値は、リソースの使用量が最も多いノードの累積コストです。

「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウでは、グラフの先頭の **RETURN (1)** に示されているとおり、合計コストは約 1,234 timeron です。推定合計コストは、ウィンドウの再上部にも表示されます。



4. 次のステップは何ですか?

Query 2 では、**runstats** を実行した後の、基本照会用のアクセス・プランを検証します。 **runstats** コマンドを実行すると、照会で参照するすべての表に関する現在の統計がオプティマイザーに提供されます。

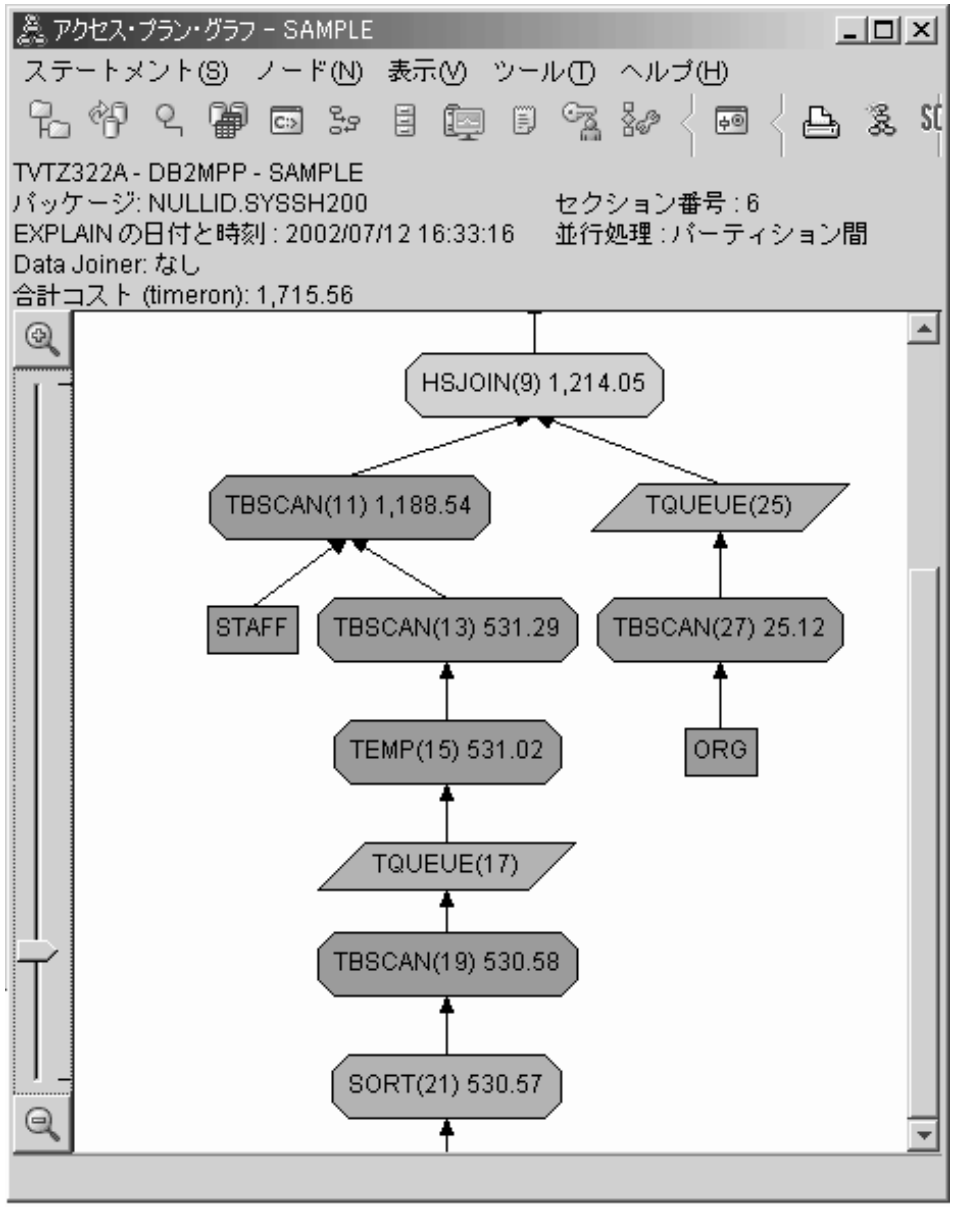
runstats を使用した表および索引の現在の統計の収集

この例では、**runstats** コマンドを実行して現在の統計を収集し、Query 1 で説明したアクセス・プランをビルドします。

強くお薦めするのは、**runstats** コマンドを使用して、表および索引に関する現在の統計を収集することです。最後に **runstats** コマンドを実行してから、大きな更新が発生した場合や新しい索引が作成されている場合は特にその必要があります。これにより、オプティマイザーは最も正確な情報に基づいて、最も効果的なアクセス・プランを決定できます。現在の統計を利用できないと、オプティマイザーは不正確なデフォルト統計に基づいて効果的でないアクセス・プランを選択してしまう可能性があります。

表の更新を行った後 で、 **runstats** を使用するようにしてください。そうしないと、オプティマイザーが表を空とみなす可能性があります。この問題は、「演算子詳細 (Operator Details)」ウィンドウのカーディナリティーが 0 である場合に明らかとなります。この場合、表更新を完了してから、 **runstats** コマンドを再実行し、関係する表の EXPLAIN スナップショットを再作成してください。

この照会 (Query 2) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statement History)」ウィンドウで、Query Number 2 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。

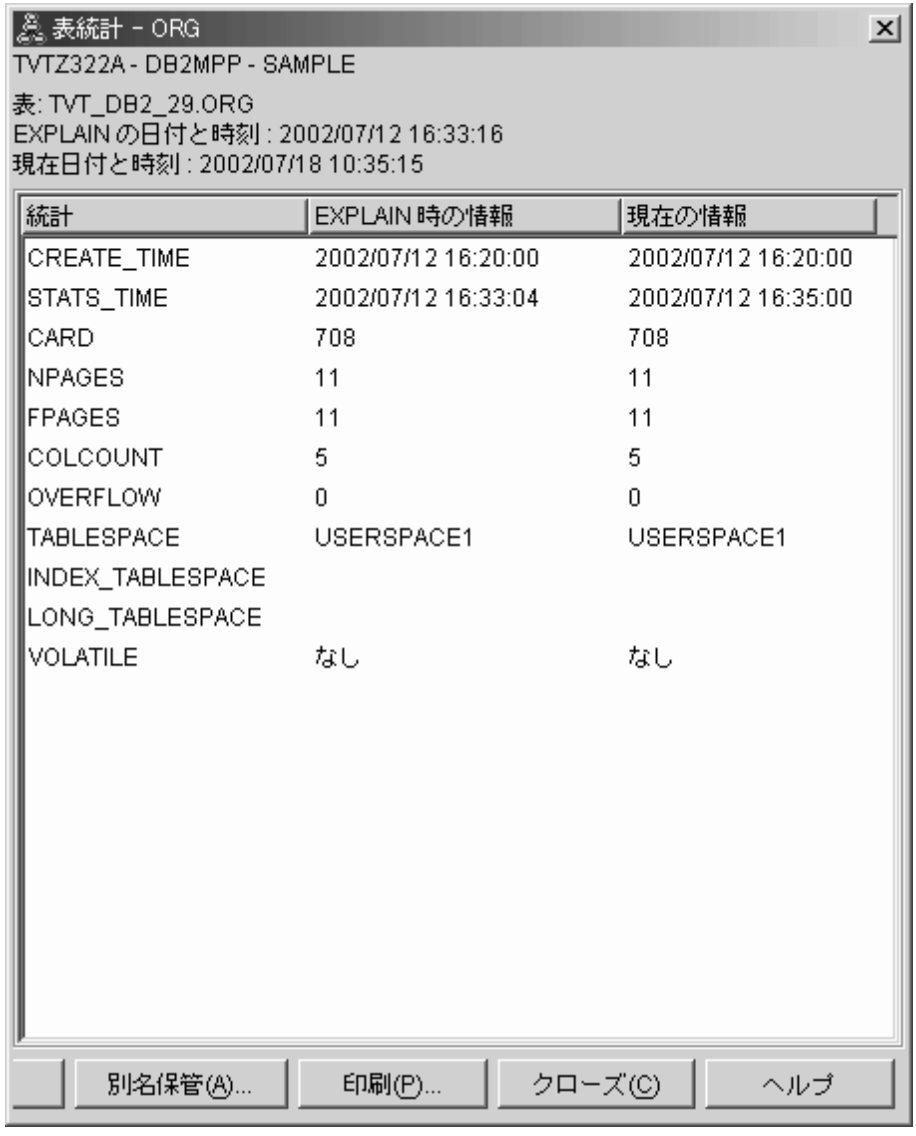


以下の質問に答えることにより、照会を改善する方法を学習します。

1. 照会で使用する各表の最新統計が存在していますか?

   ORG 表の「表統計 (Table Statistics)」ウィンドウは、オプティマイザーで実際の統計が使用されたことを示しています (**STATS_TIME** 値は、統計が実際に収集された

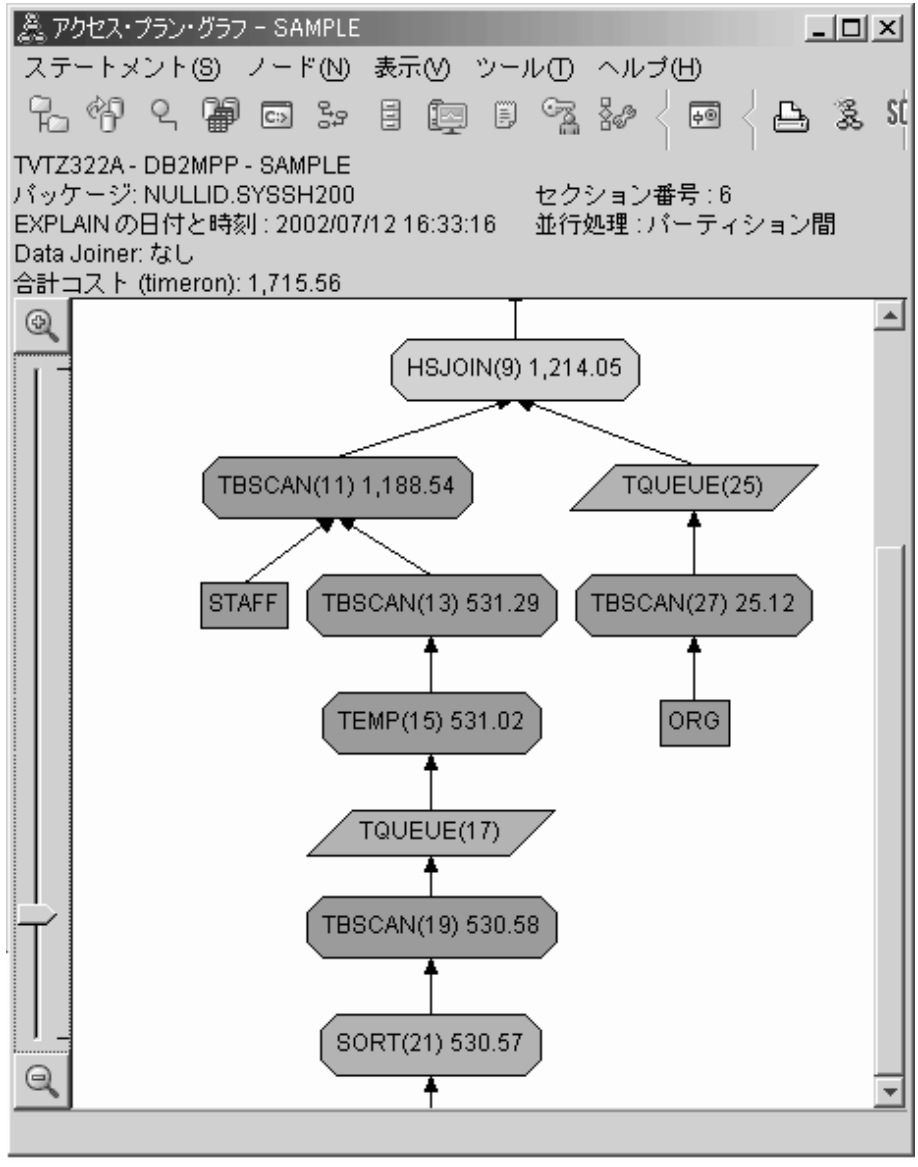
時刻です)。統計が正確かどうかは、 **runstats** コマンドを実行した後に、表の内容が大幅に変更されたかどうかに依存しています。



2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   Query 1 と同様、Query 2 のアクセス・プランでは索引スキャン (IXSCAN) ではなく、表スキャン (TBSCAN) を使用します。照会で参照する列に対して索引が定義されていないため、現在の統計が存在している場合でも、索引スキャンは実行されません。照会を改善する 1 つの方法は、オプティマイザーのために、表の結合で使用す

る列 (つまり、結合述部に指定する列) に対して索引を定義する方法です。この例では、最初のマージ・スキャン結合 HSJOIN (9) を使用します。



HSJOIN (9) 演算子の「オペレーター詳細 (Operator Details)」ウィンドウで、**「入力引数 (Input arguments)」**の下の**「結合述部 (Join predicates)」**セクションを確認します。この結合操作で使用する列は、**Text** 列にリストされます。この例の場合、列の名前は DEPTNUMB および DEPT です。

3. このアクセス・プランの効果はどれほどですか?

   アクセス・プランで最新の統計を使用すると、実際の値に近い推定コスト (単位は timeron) を得ることができます。 Query 1 の推定コストはデフォルトの統計に基づいていたため、これら 2 つのアクセス・プラン・グラフの推定コストを比較して、どちらがより効果的かを判断することはできません。この場合、コストが高いか低いかは関係ありません。効果性について有効な値を得るには、実際の統計に基づいたアクセス・プランのコストを比較する必要があります。

4. 次のステップは何ですか?

   Query 3 では、DEPTNUMB 列および DEPT 列に索引を追加したときの効果を検証します。結合述部で使用される列に対して索引を追加すると、パフォーマンスが向上することがあります。

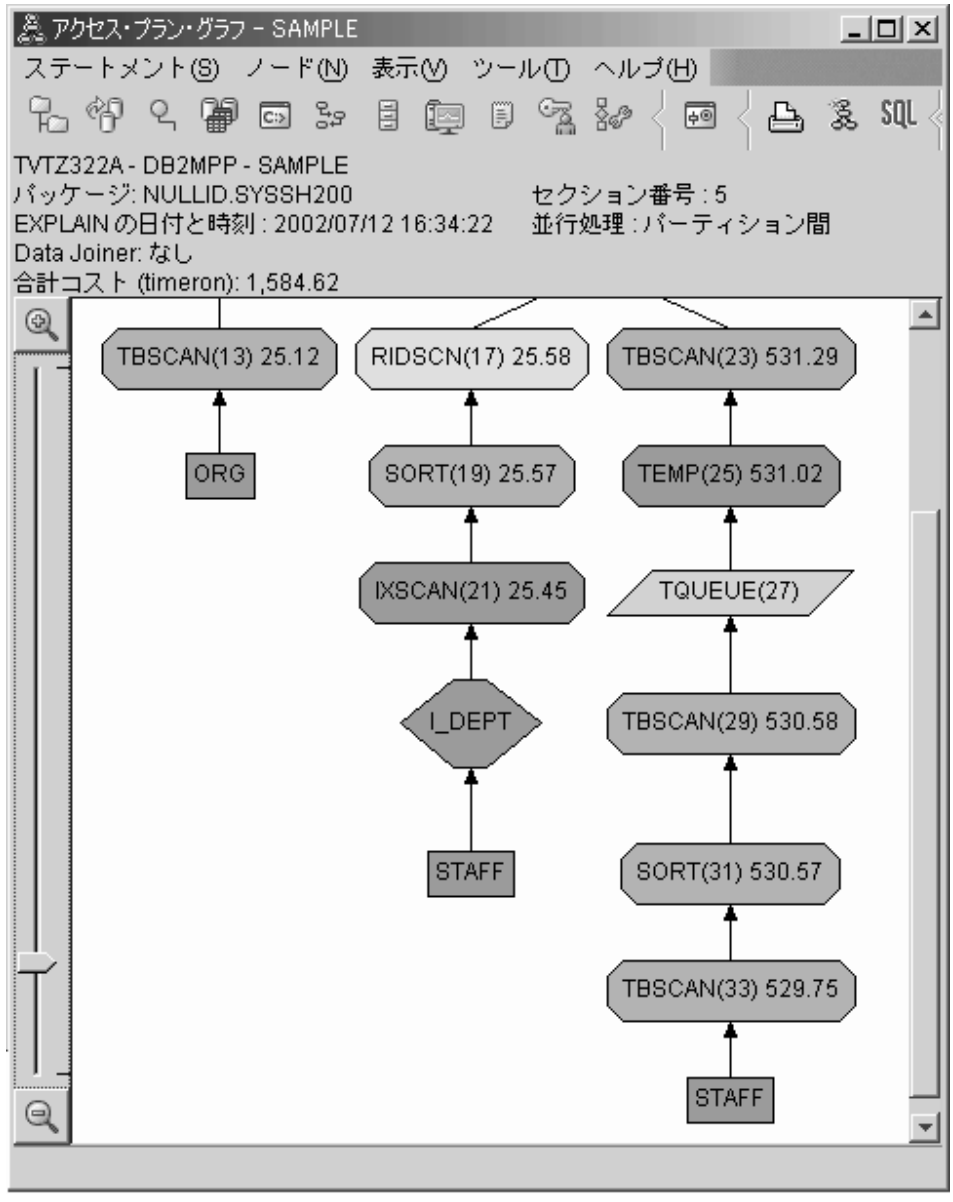
照会で複数の表を結合するための列に対する索引の作成

この例では、Query 2 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の DEPT 列と、ORG 表の DEPTNUMB 列に索引を作成します。

注: バージョン 8 では、推奨されている索引を Workload Performance ウィザードで作成することができます。

この照会 (Query 3 のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウで、Query Number 3 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。

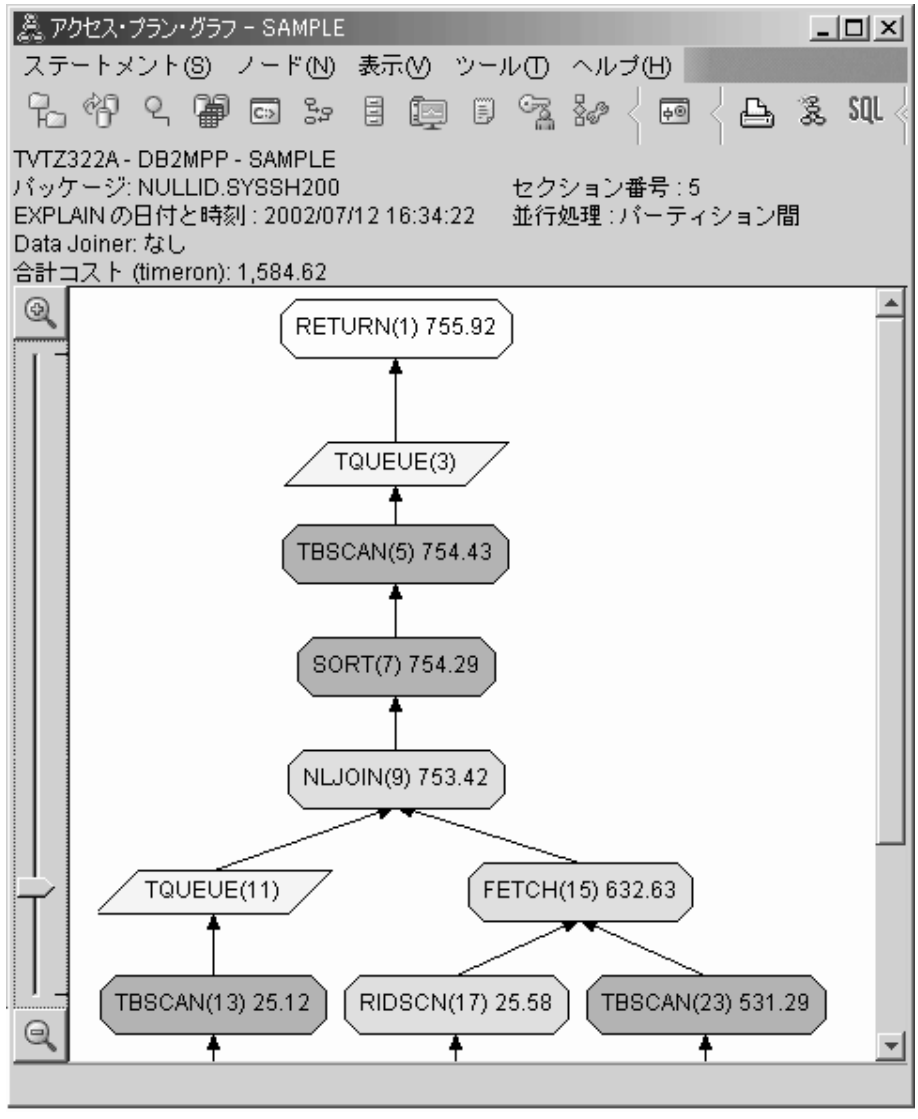
注: DEPTNUM のために索引が作成されましたが、オプティマイザーはその索引を使用しませんでした。



以下の質問に答えることにより、照会を改善する方法を学習します。

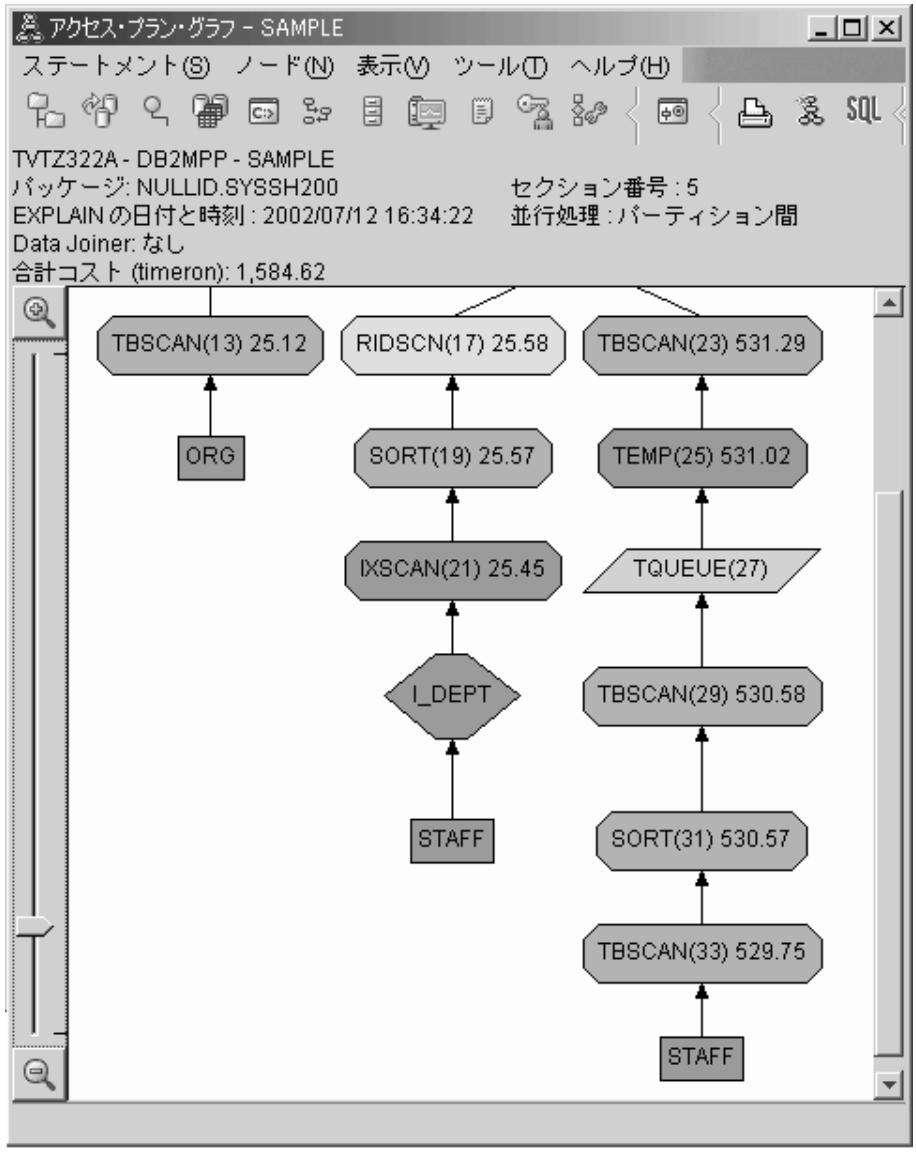
1. 索引の追加によりアクセス・プランはどのように変化しましたか?

   STAFF 表のすぐ上に、新しいひし形のノード **I_DEPT** が追加されました。このノードは、DEPT に作成された索引を表し、取得する行を判別するために、オプティマイザーが表スキャンではなく索引スキャンを使用したことを意味します。



2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   この照会のアクセス・プランは、ORG 表の DEPTNUMB 列に索引 FETCH (15) および IXSCAN (21) を作成し、STAFF 表の DEPT 列に索引を作成したときの効果を示しています。 Query 2 では、この索引が定義されていなかったため表スキャンを使用しました。



FETCH(15) 演算子の「オペレーター詳細 (Operator Details)」ウィンドウは、この操作で使用する列を示しています。

索引スキャンとフェッチ操作を組み合わせて使用すると、以前のアクセス・プランで使用した完全表スキャンよりもコストは減少します。

3. このアクセス・プランの効果はどれほどですか?

   このアクセス・プランの効率は、前の例よりも向上しました。累積コストは、Query 2 では約 1,214 timeron だったのが、 Query 3 では約 755 timeron に減少しました。

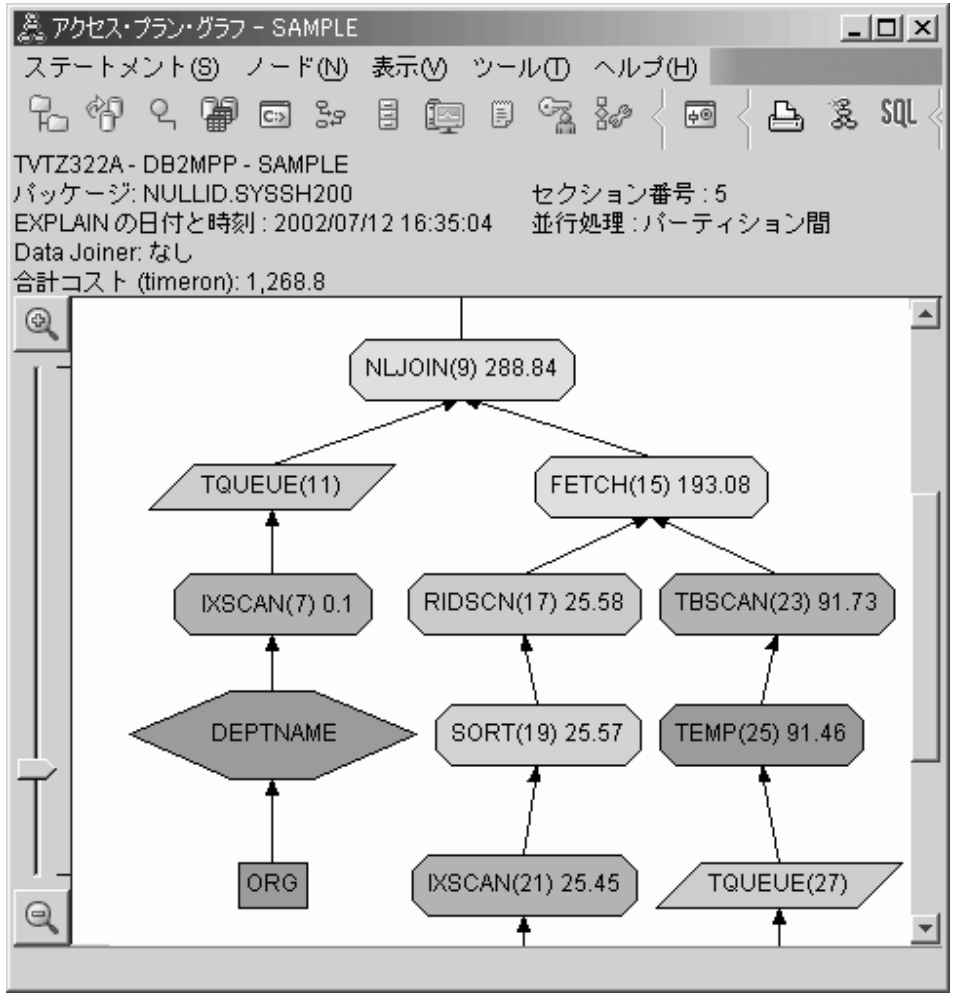
4. 次のステップは何ですか?

   Query 4 では、フェッチ操作を使用せずに、単一の索引スキャンに対するフェッチ操作および索引スキャンの回数を減らします。追加の索引を作成すると、アクセス・プランの推定コストが減少することがあります。

表の列に対する追加の索引の作成

この例では、Query 3 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の JOB 列で索引を作成し、 ORG 表に定義されている既存の索引に DEPTNAME を追加します。 (索引を追加すると、それに伴って追加のアクセスが必要になります。)

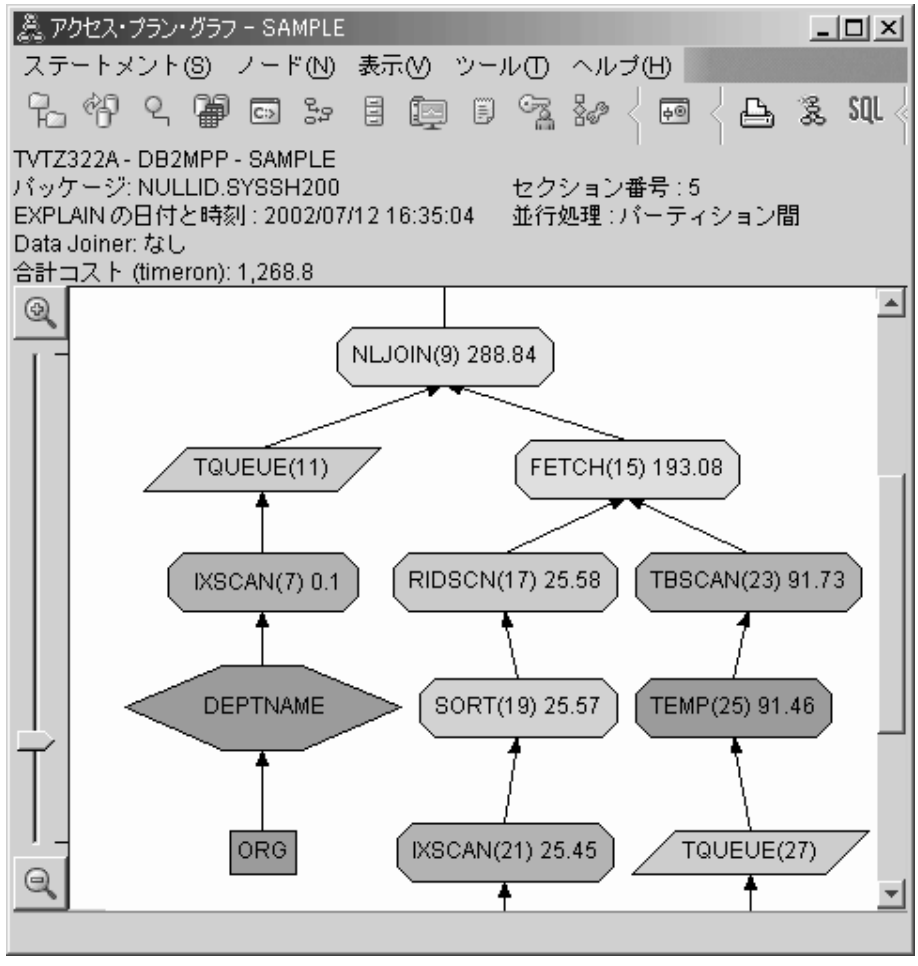
この照会 (Query 4 のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメントのヒストリー (Explained Statements History)」ウィンドウで、Query Number 4 という名前の項目をダブルクリックします。このステートメント実行に関する「アクセス・プラン・グラフ (Access Plan Graph)」ウィンドウがオープンします。



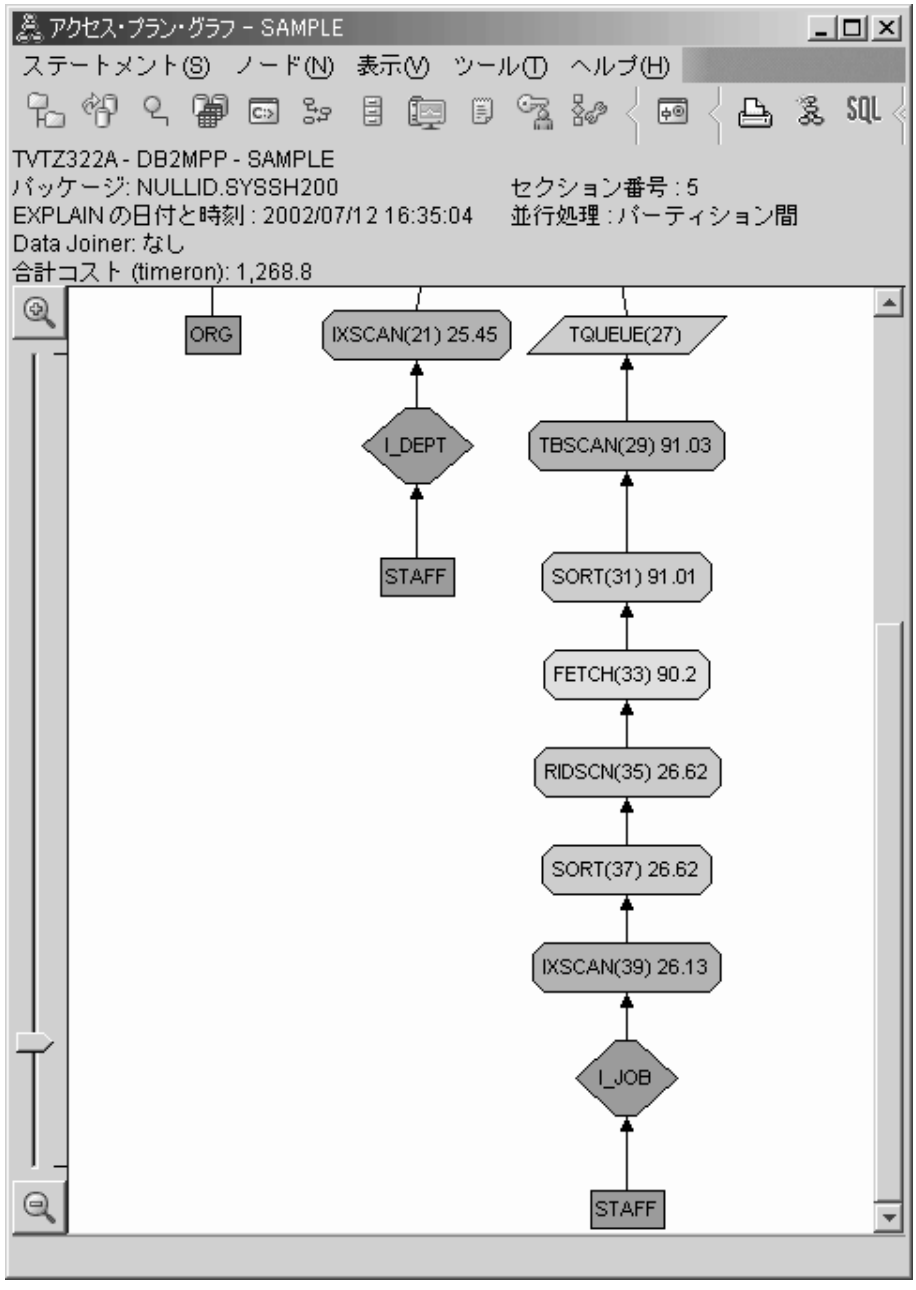
以下の質問に答えることにより、照会を改善する方法を学習します。

1. 追加の索引の作成によりアクセス・プランはどのように変化しましたか?

   アクセス・プラン・グラフの中間部に表示されている ORG 表で、以前は表スキャンが使用されていたのが、現在は索引スキャンのみ IXSCAN (7) に変化したことに注目してください。 ORG 表の DEPTNAME 列に索引を追加すると、オプティマイザーは、表スキャンが関係したアクセスを最適化することができます。



アクセス・プラン・グラフの最終部に表示されている STAFF 表で、以前は索引スキャンとフェッチ操作が使用されていたのが、現在は索引スキャンのみ IXSCAN (39) に変化したことに注目してください。 STAFF 表に JOB 索引を作成すると、オプティマイザーでは、フェッチ操作のための余分なアクセスを省略することができます。



2. このアクセス・プランの効果はどれほどですか?

   このアクセス・プランの効率は、前の例よりも向上しました。累積コストは、Query 3 では約 753 timeron だったのが、 Query 4 では約 288 timeron に減少しました。

次のステップ

パフォーマンスを向上させるためのステップの詳細については、*管理ガイド* を参照してください。次に、Visual Explain に戻って、アクションの影響を評価します。

付録 A. Visual Explain の概念

アクセス・プラン

EXPLAIN 可能な SQL ステートメントを解決するには、特定のデータが必要です。アクセス・プラン では、このデータにアクセスするための操作の順序を指定します。ここでは、選択した表、索引、または列についての統計や、操作のプロパティーを見ることができます。また、表スペースおよび関数統計といったグローバル情報、さらに最適化に関係する構成パラメーターを見ることができます。 Visual Explain では、 SQL ステートメント用のアクセス・プランを図表形式で見ることができます。

EXPLAIN 可能な SQL ステートメントがコンパイルされるたびに、オプティマイザーはアクセス・プランを生成します。このことが行われるのは、静的ステートメントの場合は準備/バインド時、動的ステートメントの場合は実行時です。

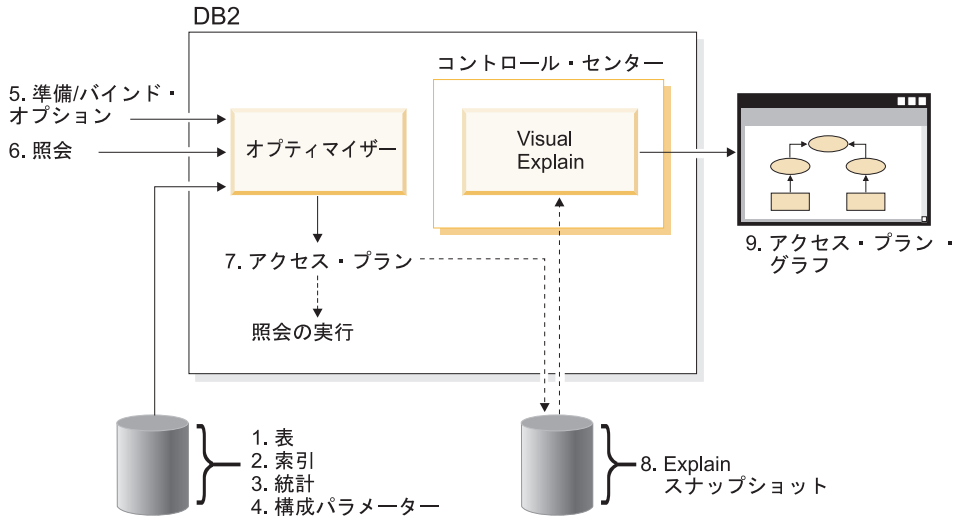
アクセス・プランは、利用できる情報に基づく *見積もり* であるということを理解しておくのは重要です。オプティマイザーは次のような情報に基づいて見積もりを行います。

- システム・カタログ表に入っている統計 (統計が最新のものでない場合は、RUNSTATS コマンドを使用してそれらを更新します)
- 構成パラメーター
- BIND オプション
- 照会最適化クラス

アクセス・プランに関連したコスト情報は、オプティマイザーにとって、照会用のリソースの使用に関する *最善の見積もり* となります。照会の実際の経過時間は、DB2 の有効範囲の外部にある要因 (たとえば、同時に実行されている他のアプリケーションの数など) によって変わってきます。実際の経過時間は、パフォーマンス・モニターを使用することによって、照会を実行している間に測定できます。

アクセス・プラン・グラフ

Visual Explain は、多くのソースからの情報を使用していかに示すようなアクセス・プラン・グラフを生成します。入ってくる情報に基づいて、 オプティマイザーは アクセス・プランを選択し、 Visual Explain はそれをアクセス・プラン・グラフ に表示します。グラフのノードは、表と索引およびそれらでの操作を示します。ノード間のリンクは、データの流れを示します。



以下のリストのタスクは、上に示されている図に対応しています。 (破線は、Visual Explain に必要なステップを示します。)

1. 表の設計を調整し、表データを再編成する。
2. 適切な索引を作成する。
3. RUNSTATS コマンドを使用して、オプティマイザーが現在の統計を生成する。
4. 適切な構成パラメーターを選択する。
5. 適切なバインド・オプションを選択する。
6. 必要なデータだけを検索するように照会を設計する。
7. アクセス・プランの作成。
8. EXPLAIN スナップショットを作成する。
9. アクセス・プラン・グラフを表示して使用する。

たとえば、Visual Explain を使用するには、表で、およびステートメントで使用される索引で **runstats** コマンドを使用して、まず現在の統計を更新します。これらの統計、構成パラメーター、BIND オプション、および照会そのものは、パッケージがバインドされるときにアクセス・プランおよび EXPLAIN スナップショットを作成するために、オプティマイザーによって使用されます。 Visual Explain は、結果の EXPLAIN スナップショットを使用して、ステートメントのアクセス・プラン・グラフを表示します。

アクセス・プラン・グラフ・ノード

ノード を表示するツリーから構成されるアクセス・プランのグラフ。これらのノードは以下のものを表します。

- 表 (長方形で表示されます)
- 索引 (ひし形で表示されます)
- 演算子 (8 角形 (8 辺) で表示されます)。 TQUEUE 演算子 (平行四辺形で表示されます)

- 表関数 (6 角形 (6 辺) で表示されます)

クラスタリング

何度も更新を行うことによりデータ・ページ上の行のロケーションが変わって、索引とデータ・ページの間に存在するクラスタリング の度合いが低下していきます。選択した索引との関連において表を再編成すると、データは再びクラスター化されます。クラスター化索引は、範囲の述部を持つ列に最も役立ちます。それにより、基本表内のデータを効率良く順次アクセスすることができるからです。似たような値が同じデータ・ページ上に置かれるので、結果としてページ取り出しが少なくて済みます。

一般に、高度なクラスタリングが可能なのは、1 つの表の中で 1 つの索引だけです。

索引のクラスタリングの度合いを調べるには、そのノードをダブルクリックして、索引統計ウィンドウを表示します。このウィンドウに、クラスター率あるいはクラスター係数の値が表示されます。この値が低い場合には、表のデータの再編成を検討してください。

詳細は、*管理ガイド* で、表データの再編成についてのセクションを参照してください。

コンテナー

コンテナー は、データの物理ストレージ・ロケーションです。それは表スペースに関連しており、ファイルか、ディレクトリーか、装置になります。

コスト

Visual Explain におけるコスト は、ステートメント (またはステートメントのエレメント) に対してアクセス・プランを実行するために必要なリソース使用量の見積合計です。コストは、CPU コスト (命令の数) と I/O (シークとページ転送の数) の組み合わせから導出されます。

コストの単位は *timeron* です。 timeron は何らかの実際の経過時間に直接等しいわけではなく、データベース・マネージャーが同じ照会に対して 2 つのプランを実行するのに必要なリソース (コスト) の相対的な概算を示しています。

アクセス・プラン・グラフの各演算子ノードに示されるコストは、アクセス・プランの実行の最初から、その特定の演算子の実行までを含む累計的なコストです。それには、システムの負荷や、データ行をユーザーに戻すコストといった要因は反映されていません。

カーソルのブロッキング

カーソルのブロッキング はオーバーヘッドを軽減するための手法の一つで、データベース・マネージャーが一回の操作で行の*ブロック* を取り出すというものです。これらの行は処理されている間、キャッシュに保管されています。このキャッシュは、アプリケーションが OPEN CURSOR 要求を出したときに割り振られ、カーソルがクローズするときに割り振り解除されます。すべての行が処理されたら、別のブロックの行が取り出されます。

PREP または **BIND** コマンドで BLOCKING オプションを使用し、以下のパラメーターを付けることにより、カーソルのブロッキングのタイプを指定します。

UNAMBIG
確定カーソルだけをブロック化します (デフォルト)。

ALL 確定カーソルと未確定カーソルの両方をブロック化します。

NO カーソルをブロック化しません。

詳細は、*管理ガイド* で、カーソルのブロッキングについてのセクションを参照してください。

データベース管理スペース (DMS) 表スペース

データベースに存在している可能性がある表スペースは、 2 種類あります。それは、データベース管理スペース (DMS) と、システム管理スペース (SMS) です。

DMS 表スペースは、データベース・マネージャーによって管理されます。その必要に合わせて設計され、調整されます。

DMS 表スペース定義には、データベースのデータが DMS 表スペース形式で保管されることになるファイル (または装置) のリストが含まれます。

記憶容量を増やすために、既存の DMS 表スペースに、事前に割り振り済みのファイル (または装置) を追加することができます。データベース・マネージャーは、その表スペースに属するすべてのコンテナー内の既存のデータのバランスを自動的に再調整します。

DMS および SMS 表スペースは、同じデータベース内に共存することができます。

動的 SQL

動的 SQL ステートメント とは、アプリケーション・プログラムの実行中にそのプログラム内で準備され、実行される SQL ステートメントです。動的 SQL では次のどちらかになります。

- CLI または CLP を使用して SQL ステートメントを対話的に発行します。

- SQL ソースをホスト言語変数に入れ、それをアプリケーション・プログラムに組み込みます。

DB2 が動的 SQL ステートメントを実行するとき、 DB2 は現行のカタログ統計と構成パラメーターに基づく アクセス・プランを作成します。このアクセス・プランは、あるアプリケーション・プログラムでのステートメントの実行と、次の実行とでは異なっていることもあります。

動的 SQL に対するもう 1 つのものとして、 静的 SQL があります。

EXPLAIN スナップショット

Visual Explain では、EXPLAIN スナップショットの内容を調べることができます。

EXPLAIN スナップショット は、 SQL ステートメントが EXPLAIN されたときに収集された情報を圧縮したものです。これは EXPLAIN_STATEMENT 表にバイナリー・ラージ・オブジェクト (BLOB) として保管され、以下の情報が入っています。

- アクセス・プランの内部表記。これには、アクセス・プランの演算子と、アクセスされる表および索引が含まれます。
- オプティマイザーが使用する決定の基準。これには、データベース・オブジェクトの統計と各操作のコストの累計が含まれます。

SQL ステートメントのアクセス・プランをグラフとして表示したい場合は、 EXPLAIN スナップショットが必要です。 EXPLAIN スナップショットが作成されるようにするには、次のようにしてください。

1. EXPLAIN スナップショットを保管するために、 EXPLAIN 表がデータベース・マネージャー内にある必要があります。これらの表の作成方法については、オンライン・ヘルプの「EXPLAIN 表を作成する」を参照してください。
2. 静的 SQL ステートメントの入ったパッケージについては、パッケージをバインドまたは prep する際に EXPLSNAP オプションを ALL または YES に設定します。パッケージに入っている EXPLAIN 可能な SQL ステートメントごとに、 EXPLAIN スナップショットを得ることができます。 **BIND** および **PREP** コマンドについては、コマンド・リファレンス を参照してください。
3. 動的 SQL ステートメントについては、それらを発行するアプリケーションをバインドする際に、 EXPLSNAP オプションを ALL に設定します。あるいは、動的 SQL ステートメントを対話的に発行する前に、CURRENT EXPLAIN SNAPSHOT 特殊レジスターを YES または EXPLAIN に設定します。詳細は、*SQL リファレンス* で、現行の EXPLAIN スナップショットについてのセクションを参照してください。

EXPLAIN 可能ステートメント

*EXPLAIN 可能*ステートメント とは、それについて EXPLAIN 操作を実行できる SQL ステートメントのことです。

EXPLAIN 可能 SQL ステートメントは次のとおりです。

- SELECT
- INSERT
- UPDATE
- DELETE
- VALUES

EXPLAIN されたステートメント

EXPLAIN されたステートメント とは、 EXPLAIN 操作が実行済みの SQL ステートメントのことです。 EXPLAIN されたステートメントは、「EXPLAIN された ステートメント・ヒストリー (Explained Statement History)」ウィンドウに表示されます。

オペランド

オペランドとは、操作が実行されるエンティティーのことです。たとえば、表や索引は各種の演算子 (TBSCAN や IXSCAN など) のオペランドです。

演算子

演算子 は、 SQL ステートメント用のアクセス・プランが実行されるときの、データに実行されるアクションか、表または索引からの出力のどちらかです。

アクセス・プラン・グラフには、以下の演算子が現れることがあります。

DELETE
表から行を削除します。

EISCAN
ユーザー定義索引を走査して、行のストリームを減らします。

FETCH
特定のレコード ID を使って、表から列を取り出します。

FILTER
1 つまたは複数の述部をデータに適用して、データをフィルターにかけます。

GRPBY
指定された列または関数の共通の値によって行をグループ化し、セット関数を評価します。

HSJOIN
2 つ以上の表が結合列上にハッシュされる、ハッシュ結合を表します。

INSERT
表に行を挿入します。

IXAND
2 つかそれ以上の索引の走査から、行 ID (RID) の論理積を取ります。

IXSCAN
任意指定の開始/終了条件によって表の索引を走査し、行の順序づけされたストリームを生成します。

MSJOIN
マージ結合 (merge join) を表しており、外部表と内部表の両方が結合述部の順番になっていなければならないことを示します。

NLJOIN
ネスト・ループ結合 (nested loop join) を表しており、外部表の各行に対して 1 回ずつ、内部表にアクセスします。

RETURN
照会からユーザーへデータが戻されることを表します。

RIDSCN
1 つかそれ以上の索引から取得される行 ID (RID) のリストを走査します。

SHIP リモート・データベース・ソースからデータを取り出します。連合システムで使用されます。

SORT 指定された列の順序により行を並べ替え、任意指定で項目の重複をなくします。

TBSCAN
データ・ページから直接、必要なデータすべてを読み取って、行を取り出します。

TEMP バックアウト読み取り (おそらく何度も行う) のために、データを TEMPORARY 表に保管します。

TQUEUE
データベース・エージェント間で表データをやりとりします。

UNION
複数の表から行のストリームを連結します。

UNIQUE
特定の列について、値が重複している行を統合します。

UPDATE
表内の行を更新します。

CMPEXP

演算子名: CMPEXP

意味: 中間結果または最終結果に必要な式の計算。

(この演算子はデバッグ・モード専用。)

DELETE

演算子名: DELETE

意味: 表からの行を削除します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、削除する行のセットを定義する他の演算子 (走査および結合など) に連結します。

パフォーマンス上の提案:

- 表からすべての行を削除する場合は、DROP TABLE ステートメントまたは **LOAD REPLACE** コマンドの使用を検討してください。

EISCAN

演算子名: EISCAN

意味: この演算子はユーザー定義索引を走査して、行のストリームを減らします。走査は、ユーザー提供の範囲生成者関数から複数の開始/停止条件を使用します。

この演算は、(述部に基づいて) 基本表にアクセスする前に、修飾する行セットの範囲を狭くするために実行されます。

パフォーマンス上の提案:

- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

FETCH

演算子名: FETCH

意味: 指定された行 ID (RID) を使用して、表から列を取り出します。

パフォーマンス上の提案:

- そのデータ・ページをアクセスする必要をなくすため、取り出された列を含むよう索引キーを展開してください。
- 取り出しに関連した索引を見つけ、そのノードをダブルクリックして、その統計ウィンドウを表示させてください。その索引のクラスタリングが高い値になっていることを確認してください。
- 取り出しに起因する入出力 (I/O) が表のページ数よりも大きい場合は、バッファー・サイズを増やしてください。
- 統計が現行のものでない場合は、RUNSTATS コマンドを使用して更新してください。

  変位値 (quantile) および頻度値の統計は、述部の選択性に関する情報を提供しており、表の走査より優先して索引の走査がいつ選ばれるかがそれによって決まります。これらの統計を更新するには、WITH DISTRIBUTION 文節を付けて、 **RUNSTATS** コマンドを表に対して使用します。

FILTER

演算子名: FILTER

意味: 述部により与えられる基準に従ってデータをフィルターに掛けるように、残りの述部を適用すること。

パフォーマンス上の提案:

- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。
- 最適化クラスが少なくとも 3 になっていることを確認してください。そうすれば、オプティマイザーは副照会ではなく結合を使用します。これが不可能な場合は、SQL 照会を手作業で書き直し、副照会を排除してください。例として、*管理ガイド* の中の SQL コンパイラーによる照会の書き直しに関するセクションをご覧ください。

GENROW

演算子名: GENROW

意味: 表、索引、または演算子からの入力を使用せずに、行からなる表を生成する組み込み関数。

GENROW は、行のデータを生成するために、オプティマイザーによって使用されます (たとえば、INSERT ステートメントや、結合にトランスフォームされる IN リストの場合)。

GENROW 関数により生成される表の概算統計を見たい場合は、そのノードをダブルクリックしてください。

GRPBY

演算子名: GRPBY

意味: 指定された列または関数の共通の値に従って行をグループ分けすること。値のグループを生成する場合、またはセット関数を評価する場合に、この演算が必要になります。

GROUP BY 列が指定されていないときにも、集合を行うときに列全体を 1 つのグループとして扱うことを指示する総計機能が SELECT リストの中に指定されているときは、GRPBY 演算子を使用できます。

パフォーマンス上の提案:

- この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、グループ化する行のセットを定義する他の演算子 (走査および結合など) に連結します。
- 単一の総計機能を持ち GROUP BY 文節のない SELECT ステートメントのパフォーマンスを改善するには、以下の方法を試してみてください。
  - MIN(C) 総計関数には、C に昇順の索引を作成します。
  - MAX(C) 総計関数には、C に降順の索引を作成します。

HSJOIN

演算子名: HSJOIN

意味: 複数の表の修飾行を、表の内容の事前の配列を行わずに直接結合できるようにするハッシュ結合。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。ハッシュ結合は、2 つの異なる表の列を等価にする結合述部は常に使用できます。結合述部はまったく同じデータ・タイプであることが必要です。 ハッシュ結合は、NLJOIN と同じように書き直された副照会から生じることもあります。

ハッシュ結合は、配列された入力表を必要としません。結合は、ハッシュ結合の内部の表を走査し、結合列の値をハッシュして参照表を生成することによって行われます。その後、外部の表を読み取り、結合列の値をハッシュし、内部表用に生成された参照表をチェックインします。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ローカル述部 (つまり、1 つの表だけを参照する述部) を使用することにより、結合する行の数を減らしてください。
- ソート・ヒープのサイズを大きくして、ハッシュ参照表をメモリーに保留できる十分な大きさにしてください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

INSERT

演算子名: INSERT

意味: 表に行を挿入します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、挿入する行のセットを定義する他の演算子 (走査および結合など) に連結します。

IXAND

演算子名: IXAND

意味: 動的ビットマップ技法を使用して、複数の索引走査の結果の論理積を取ります。潜在的な表アクセスを最小限にとどめるために、この演算子により、述部の論理積を複数の索引に適用させることができます。

この演算子は、以下の目的で実行します。
- 基本表にアクセスする前に、行セットの範囲をせばめます。
- 複数の索引に適用される述部の AND を取ります。
- スター型結合で使用し、半結合 (semijoin) の結果の AND を取ります。

パフォーマンス上の提案:
- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。
- 一般に、修飾する行の数が少ない場合は、索引の走査が最も効率的です。修飾する行の数を見積もるために、オプティマイザーは、述部に参照される列で利用できる統計を使用します。ある値が他の値より頻繁に発生するような場合は、 **runstats** コマンドに WITH DISTRIBUTION 文節を使用することにより、分散統計を要求することが重要です。一様でない分散統計を使用することにより、オプティマイザーは、頻繁に発生する値と頻繁でない値とを区別することができます。

- IXAND は単一列の索引で効果を発揮します。 IXAND の使用においては開始および停止キーが重要になります。
- スター型結合の場合、実表および関連する次元表で最も選択性の高い列にそれぞれ単一列の索引を作成します。

IXSCAN

演算子名: IXSCAN

意味: 行のストリームを減らすために索引を走査すること。走査では、任意指定の開始/停止条件を使用できます。また、索引の列を参照する、索引付き述部への適用が可能です。

この演算は、(述部に基づいて) 基本表にアクセスする前に、修飾する行セットの範囲をせばめるために実行されます。

詳細については、*管理ガイド* の中の索引走査に関するセクションをご覧ください。

パフォーマンス上の提案:

- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 2 つかそれ以上の表がアクセスされているときは、索引を介して内部表にアクセスする場合、外部表の結合列に索引を提供すると、より効率的に行えます。

  索引についての詳細なガイドラインは、Visual Explain のオンライン・ヘルプを参照してください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。
- 一般に、修飾する行の数が少ない場合は、索引の走査が最も効率的です。修飾する行の数を見積もるために、オプティマイザーは、述部に参照される列で利用できる統計を使用します。ある値が他の値より頻繁に発生するような場合は、 **runstats** コマンドに WITH DISTRIBUTION 文節を使用することにより、分散統計を要求することが重要です。一様でない分散統計を使用することにより、オプティマイザーは、頻繁に発生する値と頻繁でない値とを区別することができます。

MSJOIN

演算子名: MSJOIN

意味: 外部表と内部表の両方からの修飾行が結合述部の順番になっていなければならないマージ結合。マージ結合は、マージ・スキャン*結合* またはソート*済み*マージ*結合* とも言います。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。異なる 2 つの表からの列に等しい結合述部があるときには、常にマージ結合が可能です。このことは、書き直された副照会から生じることもあります。

たいていの場合、表は 1 回だけ走査されるので、マージ結合では結合する列において順序通りの入力を必要とします。この「順序通りの入力」は、索引またはソートされた表にアクセスすることによって得られます。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ローカル述部 (つまり、1 つの表だけを参照する述部) を使用することにより、結合する行の数を減らしてください。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

NLJOIN

演算子名: NLJOIN

意味: 外部表の各行に対して 1 回ずつ内部表をスキャン (通常は索引スキャンを使用) する、ネストされたループ結合。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。ネストされたループ結合には、結合述部が必要ではありませんが、それがあれば一般的にはパフォーマンスが上がります。

ネストされたループ結合は、次のいずれかの方法で実行します。
- 外部表のアクセスされる行のおのおのに対して、内部表全体を通して走査する。
- 外部表のアクセスされる行のおのおのに対して、内部表で索引参照を実行する。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ネストされたループ結合は、内部表の結合述部列に索引が存在していれば、効率が上がると思われます。(内部表は、NLJOIN 演算子の右に表示されます。) 内部表が IXSCAN ではなく TBSCAN になっているかどうか、確認してください。そうなっているなら、その結合列に索引を追加することを検討してください。

  重要性は劣るものの、結合をさらに効率的にするためのもう 1 つの方法は、外部表が順序通りになるように、外部表の結合列に索引を作成することです。

索引に関する詳しい指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

関連情報:

- スター型結合

PIPE

演算子名: PIPE

意味: 行に変更を加えずに、行を他の演算子に送ること。

(この演算子はデバッグ・モード専用。)

RETURN

演算子名: RETURN

意味: 照会からユーザーへデータを戻すこと。これはアクセス・プラン・グラフにおける最後の演算子であり、集計された合計値と、アクセス・プランのコストを示します。

この演算子は必須の演算を表します。

パフォーマンス上の提案:

- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。

RIDSCN

演算子名: RIDSCN

意味: 1 つまたは複数の索引から取得される行 ID (RID) のリストをスキャンします。

オプティマイザーがこの演算子を考慮するのは、以下の場合です。

- 述部が OR キーワードにより接続されているか、 IN 述部が存在する場合。 index ORing と呼ばれる技法が使用されます。これは、同じ表に対する複数の索引アクセスの結果を組み合わせるものです。
- 単一索引アクセスのために、リスト・プリフェッチを使用するのが効果的な場合。基礎行にアクセスする前に行 ID をソートしておくなら、入出力の効率が上がるからです。

RQUERY

演算子名: SHIP

意味: 連合システムで、リモート・データ・ソースからデータを取り出すために使用される演算子。 SHIP 演算子は、照会結果を取り出すためにこの演算子が SQL SELECT ステートメントをリモート・データ・ソースに送信するときに、オプティマイザーによって考慮されます。 SELECT ステートメントは、データ・ソースによってサポートされる SQL ダイアレクトを使用して生成され、データ・ソースによって許可される有効な照会を含むことができます。

パフォーマンス上の提案: *管理ガイド* を参照してください。

SORT

演算子名: SORT

意味: 表内の行を、 1 つかそれ以上の列の順序に並べ替えること。その際、オプションで、行の重複をなくします。

要求された順序づけを満足するような索引が存在していない場合、ソートは必須です。また、ソートが索引スキャンよりもコストがかからない場合にも使用されます。ソートは通常、必要な行がいったん取り出された後、最後に実行される操作です。あるいは、結合やグループ化に先立ってデータのソートを行います。

行の数が多い場合や、ソートされたデータをパイプ処理することができない場合、この操作にはコストのかかる TEMPORARY 表の生成が求められます。

ソートに関する詳細は、*管理ガイド* をご覧ください。

パフォーマンス上の提案:

- ソート列に索引を追加することを検討してください。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。
- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。
- システム TEMPORARY 表スペースのプリフェッチ・サイズが十分であること、つまり、入出力に制約されないことを確認します。 (このことをチェックするには、**「Statement」-> 「Show statistics」->「Table spaces」**の順に選択してください。)
- 頻繁に大規模なソートが必要とされる場合は、以下の構成パラメーターの値を増やすことを検討してください。

– ソート・ヒープ・サイズ (sortheap)。このパラメーターを変更するには、Control Center でデータベースを右クリックし、そのポップアップ・メニューから*構成* を選択します。そこで表示されるノートブックから、「**パフォーマンス**」タブを選択します。

– ソート・ヒープしきい値 (sheapthres)。このパラメーターを変更するには、 Control Center でデータベース・インスタンスを右クリックし、そのポップアップ・メニューから*構成* を選択します。そこで表示されるノートブックから、「**パフォーマンス**」タブを選択します。

- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

TBSCAN

演算子名: TBSCAN

意味: データ・ページから直接、必要なデータすべてを読み取って、行を取り出す表スキャン (関係スキャン) です。

オプティマイザーが索引スキャンよりもこの種のスキャンを選択するのは、次のような場合です。

- 走査される値の範囲が頻繁に発生する場合 (つまり、表の大部分をアクセスする必要がある場合)。
- 表が小さい場合。
- 索引のクラスタリングの度合いが低い場合。
- 索引が存在していない場合。

表と索引の走査に関する詳細は、*管理ガイド* をご覧ください。

パフォーマンス上の提案:

- 表が大きく、表の行の大半がアクセスされない場合は、表スキャンよりも索引スキャンの方が効率的です。この状況でオプティマイザーが索引スキャンを採用する可能性を高めるために、 選択述部のある列に索引を追加することを検討してください。

  索引に関する詳しい指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

- 索引がすでにあるが使用されていない場合、それに先行する列のおのおのに選択述部があることを確認してください。選択述部があれば、次に、その索引のクラスタリングが高い値になっていることを確認してください。 (この統計を見るには、ソートの下にある表に対して表統計ウィンドウをオープンし、その「*索引*」ボタンを選択して、索引統計ウィンドウを表示させてください。)

- 表スペースのプリフェッチ・サイズが十分であること、つまり、入出力に縛られていないことを確認してください。 (このことをチェックするには、**「Statement」-> 「Show statistics」->「Table spaces」**の順に選択してください。)

詳細については、*管理ガイド* のバッファー・プールへのデータのプリフェッチに関するセクションを参照してください。

- 統計が現行のものでない場合は、RUNSTATS コマンドを使用して更新してください。

  変位値および頻度値の統計は、述部の選択性に関する情報を提供しています。たとえば、これらの統計を使って、どんなときに、表走査より優先して索引走査を選ぶかを判別できます。これらの値を更新するには、WITH DISTRIBUTION 文節を付けて、表に対して **runstats** コマンドを使用します。

TEMP

演算子名: TEMP

意味: 他の演算子によるバックアウト読み取り (おそらく何度も行う) のために、データを TEMPORARY 表に保管する処置。その表は、SQL ステートメントが処理された後に除去されます (それ以前に除去されていない場合)。

この演算子は、副照会の評価や中間結果の保管のために必要になります。状況によっては (ステートメントが更新可能な場合など)、これは必ず必要です。

TQUEUE

演算子名: TQUEUE

意味: 複数のデータベース・エージェントが 1 つの照会を処理している場合に、1 つのデータベース・エージェントから別のデータベース・エージェントへデータを渡すのに使われる表キュー。複数データベース・エージェントは、並列処理が関係しているときに照会を処理するために使用されます。

表キューには次のような種類があります。

- **ローカル:** 単一のノード内にあるデータベース・エージェント間でデータを受け渡しするために、表キューが使用されます。ローカル表キューは、パーティション内並列処理に使用されます。
- **非ローカル:** 別々のノードにあるデータベース・エージェント間でデータを受け渡しするために、表キューが使用されます。

UNION

演算子名: UNION

意味: 複数の表から行のストリームを連結します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、連結する行のセットを定義する他の演算子 (走査および結合など) に連結します。

UNIQUE

演算子名: UNIQUE

意味: 指定された列について同じ値を持つ行の重複をなくします。

パフォーマンス上の提案:

- この演算子は、該当する列にユニーク索引が存在する場合に限り、不必要です。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

UPDATE

演算子名: UPDATE

意味: 表の行の中のデータを更新します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、更新する行のセットを定義する他の演算子 (走査および結合など) に連結します。

Optimizer

オプティマイザー は SQL コンパイラーのコンポーネントの 1 つで、データ操作言語 (DML) の SQL ステートメントに対して アクセス・プランを選択します。アクセス・プランのいろいろな代替案の実行コストをモデル化し、見積コストが最も低いものを選ぶという方法が取られます。

パッケージ

パッケージ はデータベースに保管されるオブジェクトの一種で、アプリケーション・プログラムの 1 つのソース・ファイルに関連した SQL ステートメントを処理するのに必要な情報が入っています。次のいずれかの方法で生成されます。

- **PREP** コマンドでソース・ファイルをプリコンパイルする
- プリコンパイラーにより生成されたバインド・ファイルを **BIND** コマンドでバインドする

述部

述部 とは、比較演算を暗黙指定する検索条件のエレメントです。述部は WHERE または HAVING で始まる文節に含まれます。

たとえば、次の SQL ステートメントでは

```
SELECT * FROM SAMPLE
  WHERE NAME = 'SMITH' AND
  DEPT = 895 AND YEARS > 5
```

述部は次のとおりです。NAME = 'SMITH'; DEPT = 895; AND YEARS > 5

述部は次のいずれかのカテゴリーに分けられます (効率の高い順に列挙します)。

1. 開始および終了条件は、索引スキャンをまとめ、範囲をせばめます。 (これらの条件は範囲限定述部ともいいます。)
2. 索引ページ (索引検索引き数ともいう) 述部は、索引から評価できます。それは、述部に関与する列が索引キーの一部になっているからです。
3. データ・ページ (データ検索引き数述部ともいう) 述部は、索引から評価できませんが、行がバッファー内にとどまっている間は評価できます。
4. 残りの述部は、基本表の単純なアクセス以上の入出力を必要とします。また、バッファー・ページの外へデータがコピーされた後に適用する必要があります。それらには、副照会を持つ述部が含まれ、また表とは別のファイルに保管されている LONG VARCHAR または LOB データを読み取る述部も含まれます。

述部を設計する際には、戻される行をできるだけ少なくするため、最も高い選択性を選ぶようにしてください。

以下のタイプの述部は最も効率が高く、最も一般的に使用されています。

- *単純等式結合述部* は、マージ結合に必要とされます。これは table1.column = table2.column という形のもので、 2 つの異なる表にある列を等しくして、それらの表を結合できるようにします。
- *ローカル述部* は 1 つの表だけに適用されます。

詳細については、*管理ガイド* でデータ・アクセスの概念と最適化に関するセクションを参照してください。

照会最適化クラス

*照会最適化クラス*は、照会のコンパイルのための照会再記述規則と最適化技法のセットです。

主な照会最適化クラスは次のとおりです。

1 限定された最適化。メモリーや処理リソースの制限が厳しい場合に有用です。バージョン 1 で提供された最適化とだいたい同じです。

2 低レベルの最適化。レベル 1 よりは高いレベルですが、レベル 3 以上よりは (特に非常に複雑な照会の場合に) かなりコストが低い最適化レベルです。

3 中レベルの最適化。 DB2 for MVS/ESA の照会最適化の特性に最も良く一致します。

5 通常の最適化。単純なトランザクションと複雑な照会の両方を使用する混合環境に向いています。

7 通常の最適化。複雑な動的 SQL 照会に対して照会最適化の量を減らさないという点を除けば、最適化レベル 5 と同じです。

特殊な状況に限って使用される、他の照会最適化クラスは以下のとおりです。

0 最小の最適化。最適化がほとんど、あるいは全く必要ないときだけ (つまり、良く索引付けされた表で非常に単純な照会を行う場合) に使用します。

9 最高の最適化。かなりのメモリーと処理リソースを使用します。クラス 5 で不十分な場合 (つまり、クラス 5 ではうまくいかないような非常に複雑で実行時間の長い照会の場合) だけに使用してください。

一般的に言って、静的な照会や、実行に長い時間がかかることが予想される照会には高い方の最適化を使用し、動的に発行する単純な照会や、実行回数の少ない照会には低い方の最適化を使用します。

動的 SQL ステートメントに照会最適化を設定するには、コマンド行プロセッサーに次のコマンドを入力してください。

```
SET CURRENT QUERY OPTIMIZATION = n;
```

ここで、'n' に、使用したい照会最適化クラスを指定します。

静的 SQL ステートメントに対して照会最適化を設定するには、 **BIND** または **PREP** コマンドで QUERYOPT オプションを使用します。

詳細については、*管理ガイド* で最適化クラスの調整に関するセクションを参照してください。

述部の選択性

選択性 とは、何らかの行が述部を満足する (つまり、「真」になる) 可能性のことを言います。

たとえば、1,000,000 の行がある表に対して、選択性が 0.01 (1%) の述部を作用させた場合、その述部は概算で 10,000 の行 (1,000,000 の 1%) を戻し、概算で 990,000 の行を破棄します。

選択性の高い述部 (選択性が 0.10 以下のもの) が望ましいといえます。そのような述部は、その後に作用する操作に対して少しの行だけを戻すので、照会を満足させるために必要な CPU と入出力が少なくて済みます。

例

1,000,000 の行を持つ表があって、最初の照会が 'ORDER BY' を含んでおり、追加のソート・ステップが必要であるとします。選択性が 0.01 の述部では、概算で 10,000 の行に対してソートを行う必要があります。しかし、より選択性の弱い 0.50 の述部では、概算で 500,000 の行に対してソートを行う必要があり、結果としてより多くの CPU と I/O 時間が必要になります。

スター型結合

ある実表 (大きな中央表) が 2 つ以上の次元表 (実表の列値について記述した小さな表) と結合されている場合、この結合セットをスター型結合と言います。

スター型結合は、3 つの主要部分で構成されます。
- 半結合
- 半結合の結果の索引 ANDing
- 半結合の完了

2 つ以上の結合が IXAND 演算子を持つことを示します。

半結合とは、結合の結果が内部表と外部表の列の結合ではなく、内部表の行 ID (RID) である特殊な形態の結合です。

スター型結合は半結合を使用することにより、索引 ANDing 演算子に行 ID を提供します。索引 ANDing 演算子は、さまざまな結合のフィルター効果を累積します。索引 ANDing 演算子からの出力は、索引 ORing 演算子に送られます。これにより、行 ID を配列し、索引 ANDing 演算子を必要とする結合の結果重複することになった行を除去します。実表の行は、Fetch 演算子を使用してフェッチされます。最終的には、修正された実表がすべての次元表に結合され、結合は完了します。

パフォーマンス上の提案:
- 次元表の結合ごとに実表に索引を作成してください。
- ソート・ヒープしきい値が、索引 ANDing 演算子のビット・フィルターを割り振ることができるのに十分であるかを確認してください。スター型結合の場合、12MB または 3000 4K ページ必要です。パーティション内並列処理の場合、ビット・フィルターは dbheap と同じ共有メモリー・セグメントから割り振られますが、 sortheap (およびインスタンスでの sheapthres) によってバインドされます。このため、共有メモリーは、sortheap および sheapthres によって制御され、 12MB 以上必要とされます。

- 次元表に対してフィルター述部を適用してください。統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

静的 SQL

静的 SQL ステートメントは、アプリケーション・プログラムに組み込まれています。アプリケーションが実行できるようになる前に、まず、これらすべての組み込みステートメントをコンパイルし、パッケージ にバインドする必要があります。

DB2 がこれらのステートメントをコンパイルする際、おのおのに対してアクセス・プランを作成します。これは、ステートメントがプリコンパイルおよびバインドされた時点でのカタログ統計と構成パラメーターに基づいています。

アプリケーションが実行されるときは、常にこれらのアクセス・プランが使用されます。アクセス・プランはパッケージが再びバインドされるまでは変更されません。

静的 SQL に対するもう 1 つのものとして 動的 SQL があります。

システム管理スペース (SMS) 表スペース

データベースに存在しうる表スペースは、2 種類あります。システム管理スペース (SMS) と データベース管理スペース (DMS) です。

SMS 表スペースはオペレーティング・システムによって管理され、ここでは表スペースが作成されたときに割り当てられるスペースへ、データベース・データが保管されます。表スペース定義には、このデータが保管される 1 つまたは複数のディレクトリー・パスのリストが含まれます。

ファイル・システムは、メディア・ストレージの割り振りと管理をコントロールします。

SMS および DMS 表スペースは、同じデータベース内に共存することができます。

表スペース

非常に大きなデータベースは、表スペースと呼ばれる、別個に管理されるいくつかの部分に分けるなら、管理が容易になります

表スペースにより、データのロケーションを、特定の LU またはその一部分に割り当てることができます。たとえば、表を作成するときに、その索引または長い列 (長い、またはラージ・オブジェクト (LOB) データが入っている) を、表データの残りの部分とは別の所に保管するよう指定することができます。

表スペースを 1 つか複数の物理記憶装置 (コンテナー) に広げてパフォーマンスを改善することができます。ただし、1 つの表スペース内のすべての装置またはコンテナーが似たようなパフォーマンス特性を持つようにすることが勧められています。

表スペースは 2 種類の方法で管理することができます。 システム管理スペース (SMS) として管理する方法とデータベース管理スペース (DMS) として管理する方法です。

Visual Explain

注: バージョン 6 では現在、コマンド行から Visual Explain を呼び出すことはできません。ただし、Control Center のさまざまなデータベース・オブジェクトからの呼び出しについては、これまで通り可能です。このバージョンでは、Visual Explain という名称として使用します。

Visual Explain を使用すると、EXPLAIN された SQL ステートメントの アクセス・プランをグラフで表示できます。このグラフからの情報を使用して、SQL 照会のパフォーマンスを良くすることができます。

アクセス・プラン・グラフは以下を詳細に示します。
- 表 (およびそれに関連する列) と索引
- 演算子 (たとえば、表の走査、ソート、および結合)
- 表スペースおよび関数

Visual Explain を使用してさらに以下のことを行うこともできます。
- 最適化した場合の統計の表示。これらの統計を現在のカタログ統計と比較して、パッケージを再バインドするとパフォーマンスが良くなるかどうかを判別できるようにします。
- 表のアクセスに索引が使用されたかどうかの判別。索引が使用されなかった場合、Visual Explain はどの列が索引を付けると効果的かを判別できるようにします。
- 照会についてのアクセス・プラン・グラフにさまざまなチューニング技法を行った前後を比較することにより、それらの効果の表示。
- 全体で見積もられるコストと検索する行数 (カーディナリティー) を含む、アクセス・プラン内の各演算子についての情報の取得。

付録 B. Visual Explain 演算子のアルファベット順リスト

CMPEXP

演算子名: CMPEXP

意味: 中間結果または最終結果に必要な式の計算。

(この演算子はデバッグ・モード専用。)

DELETE

演算子名: DELETE

意味: 表からの行を削除します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、削除する行のセットを定義する他の演算子 (走査および結合など) に連結します。

パフォーマンス上の提案:

- 表からすべての行を削除する場合は、DROP TABLE ステートメントまたは **LOAD REPLACE** コマンドの使用を検討してください。

EISCAN

演算子名: EISCAN

意味: この演算子はユーザー定義索引を走査して、行のストリームを減らします。走査は、ユーザー提供の範囲生成者関数から複数の開始/停止条件を使用します。

この演算は、(述部に基づいて) 基本表にアクセスする前に、修飾する行セットの範囲を狭くするために実行されます。

パフォーマンス上の提案:

- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

FETCH

演算子名: FETCH

意味: 指定された行 ID (RID) を使用して、表から列を取り出します。

パフォーマンス上の提案:

- そのデータ・ページをアクセスする必要をなくすため、取り出された列を含むよう索引キーを展開してください。
- 取り出しに関連した索引を見つけ、そのノードをダブルクリックして、その統計ウィンドウを表示させてください。その索引のクラスタリングが高い値になっていることを確認してください。
- 取り出しに起因する入出力 (I/O) が表のページ数よりも大きい場合は、バッファー・サイズを増やしてください。
- 統計が現行のものでない場合は、RUNSTATS コマンドを使用して更新してください。

  変位値 (quantile) および頻度値の統計は、述部の選択性に関する情報を提供しており、表の走査より優先して索引の走査がいつ選ばれるかがそれによって決まります。これらの統計を更新するには、WITH DISTRIBUTION 文節を付けて、 **RUNSTATS** コマンドを表に対して使用します。

FILTER

演算子名: FILTER

意味: 述部により与えられる基準に従ってデータをフィルターに掛けるように、残りの述部を適用すること。

パフォーマンス上の提案:

- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。
- 最適化クラスが少なくとも 3 になっていることを確認してください。そうすれば、オプティマイザーは副照会ではなく結合を使用します。これが不可能な場合は、SQL 照会を手作業で書き直し、副照会を排除してください。例として、*管理ガイド* の中の SQL コンパイラーによる照会の書き直しに関するセクションをご覧ください。

GENROW

演算子名: GENROW

意味: 表、索引、または演算子からの入力を使用せずに、行からなる表を生成する組み込み関数。

GENROW は、行のデータを生成するために、オプティマイザーによって使用されます (たとえば、INSERT ステートメントや、結合にトランスフォームされる IN リストの場合)。

GENROW 関数により生成される表の概算統計を見たい場合は、そのノードをダブルクリックしてください。

GRPBY

演算子名: GRPBY

意味: 指定された列または関数の共通の値に従って行をグループ分けすること。値のグループを生成する場合、またはセット関数を評価する場合に、この演算が必要になります。

GROUP BY 列が指定されていないときにも、集合を行うときに列全体を 1 つのグループとして扱うことを指示する総計機能が SELECT リストの中に指定されているときは、GRPBY 演算子を使用できます。

パフォーマンス上の提案:

- この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、グループ化する行のセットを定義する他の演算子 (走査および結合など) に連結します。
- 単一の総計機能を持ち GROUP BY 文節のない SELECT ステートメントのパフォーマンスを改善するには、以下の方法を試してみてください。
  - MIN(C) 総計関数には、C に昇順の索引を作成します。
  - MAX(C) 総計関数には、C に降順の索引を作成します。

HSJOIN

演算子名: HSJOIN

意味: 複数の表の修飾行を、表の内容の事前の配列を行わずに直接結合できるようにするハッシュ結合。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。ハッシュ結合は、2 つの異なる表の列を等価にする結合述部は常に使用できます。結合述部はまったく同じデータ・タイプであることが必要です。 ハッシュ結合は、NLJOIN と同じように書き直された副照会から生じることもあります。

ハッシュ結合は、配列された入力表を必要としません。結合は、ハッシュ結合の内部の表を走査し、結合列の値をハッシュして参照表を生成することによって行われます。その後、外部の表を読み取り、結合列の値をハッシュし、内部表用に生成された参照表をチェックインします。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ローカル述部 (つまり、1 つの表だけを参照する述部) を使用することにより、結合する行の数を減らしてください。
- ソート・ヒープのサイズを大きくして、ハッシュ参照表をメモリーに保留できる十分な大きさにしてください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

INSERT

演算子名: INSERT

意味: 表に行を挿入します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、挿入する行のセットを定義する他の演算子 (走査および結合など) に連結します。

IXAND

演算子名: IXAND

意味: 動的ビットマップ技法を使用して、複数の索引走査の結果の論理積を取ります。潜在的な表アクセスを最小限にとどめるために、この演算子により、述部の論理積を複数の索引に適用させることができます。

この演算子は、以下の目的で実行します。
- 基本表にアクセスする前に、行セットの範囲をせばめます。
- 複数の索引に適用される述部の AND を取ります。
- スター型結合で使用し、半結合 (semijoin) の結果の AND を取ります。

パフォーマンス上の提案:

- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

- 一般に、修飾する行の数が少ない場合は、索引の走査が最も効率的です。修飾する行の数を見積もるために、オプティマイザーは、述部に参照される列で利用できる統計を使用します。ある値が他の値より頻繁に発生するような場合は、 **runstats** コマンドに WITH DISTRIBUTION 文節を使用することにより、分散統計を要求することが重要です。一様でない分散統計を使用することにより、オプティマイザーは、頻繁に発生する値と頻繁でない値とを区別することができます。
- IXAND は単一列の索引で効果を発揮します。 IXAND の使用においては開始および停止キーが重要になります。
- スター型結合の場合、実表および関連する次元表で最も選択性の高い列にそれぞれ単一列の索引を作成します。

IXSCAN

演算子名: IXSCAN

意味: 行のストリームを減らすために索引を走査すること。走査では、任意指定の開始/停止条件を使用できます。また、索引の列を参照する、索引付き述部への適用が可能です。

この演算は、(述部に基づいて) 基本表にアクセスする前に、修飾する行セットの範囲をせばめるために実行されます。

詳細については、*管理ガイド* の中の索引走査に関するセクションをご覧ください。

パフォーマンス上の提案:

- データベースの更新が何度も行われると、索引のフラグメント化が進んで、必要以上に索引ページが増えてしまうことがあります。この事態を修復するには、索引をいったんドロップして再作成するか、あるいは索引の再編成を行います。
- 2 つかそれ以上の表がアクセスされているときは、索引を介して内部表にアクセスする場合、外部表の結合列に索引を提供すると、より効率的に行えます。

  索引についての詳細なガイドラインは、Visual Explain のオンライン・ヘルプを参照してください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。
- 一般に、修飾する行の数が少ない場合は、索引の走査が最も効率的です。修飾する行の数を見積もるために、オプティマイザーは、述部に参照される列で利用できる統計を使用します。ある値が他の値より頻繁に発生するような場合は、 **runstats** コマンドに WITH DISTRIBUTION 文節を使用することにより、分散統計を要求することが重要です。一様でない分散統計を使用することにより、オプティマイザーは、頻繁に発生する値と頻繁でない値とを区別することができます。

MSJOIN

演算子名: MSJOIN

意味: 外部表と内部表の両方からの修飾行が結合述部の順番になっていなければならないマージ結合。マージ結合は、マージ・スキャン*結合* またはソート*済み*マージ*結合* とも言います。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。異なる 2 つの表からの列に等しい結合述部があるときには、常にマージ結合が可能です。このことは、書き直された副照会から生じることもあります。

たいていの場合、表は 1 回だけ走査されるので、マージ結合では結合する列において順序通りの入力を必要とします。この「順序通りの入力」は、索引またはソートされた表にアクセスすることによって得られます。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ローカル述部 (つまり、1 つの表だけを参照する述部) を使用することにより、結合する行の数を減らしてください。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

NLJOIN

演算子名: NLJOIN

意味: 外部表の各行に対して 1 回ずつ内部表をスキャン (通常は索引スキャンを使用) する、ネストされたループ結合。

FROM 文節で参照される表が複数ある場合には、常に結合が必要です。ネストされたループ結合には、結合述部が必要ではありませんが、それがあれば一般的にはパフォーマンスが上がります。

ネストされたループ結合は、次のいずれかの方法で実行します。
- 外部表のアクセスされる行のおのおのに対して、内部表全体を通して走査する。
- 外部表のアクセスされる行のおのおのに対して、内部表で索引参照を実行する。

詳細については、*管理ガイド* の中の結合の概念に関するセクションをご覧ください。

パフォーマンス上の提案:

- ネストされたループ結合は、内部表の結合述部列に索引が存在していれば、効率が上がると思われます。(内部表は、NLJOIN 演算子の右に表示されます。) 内部表が IXSCAN ではなく TBSCAN になっているかどうか、確認してください。そうなっているなら、その結合列に索引を追加することを検討してください。

  重要性は劣るものの、結合をさらに効率的にするためのもう 1 つの方法は、外部表が順序通りになるように、外部表の結合列に索引を作成することです。

  索引に関する詳しい指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

関連情報:

- スター型結合

PIPE

演算子名: PIPE

意味: 行に変更を加えずに、行を他の演算子に送ること。

(この演算子はデバッグ・モード専用。)

RETURN

演算子名: RETURN

意味: 照会からユーザーへデータを戻すこと。これはアクセス・プラン・グラフにおける最後の演算子であり、集計された合計値と、アクセス・プランのコストを示します。

この演算子は必須の演算を表します。

パフォーマンス上の提案:

- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。

RIDSCN

演算子名: RIDSCN

意味: 1 つまたは複数の索引から取得される行 ID (RID) のリストをスキャンします。

オプティマイザーがこの演算子を考慮するのは、以下の場合です。

- 述部が OR キーワードにより接続されているか、 IN 述部が存在する場合。 index ORing と呼ばれる技法が使用されます。これは、同じ表に対する複数の索引アクセスの結果を組み合わせるものです。
- 単一索引アクセスのために、リスト・プリフェッチを使用するのが効果的な場合。基礎行にアクセスする前に行 ID をソートしておくなら、入出力の効率が上がるからです。

RQUERY

演算子名: SHIP

意味: 連合システムで、リモート・データ・ソースからデータを取り出すために使用される演算子。 SHIP 演算子は、照会結果を取り出すためにこの演算子が SQL SELECT ステートメントをリモート・データ・ソースに送信するときに、オプティマイザーによって考慮されます。 SELECT ステートメントは、データ・ソースによってサポートされる SQL ダイアレクトを使用して生成され、データ・ソースによって許可される有効な照会を含むことができます。

パフォーマンス上の提案: *管理ガイド* を参照してください。

SORT

演算子名: SORT

意味: 表内の行を、 1 つかそれ以上の列の順序に並べ替えること。その際、オプションで、行の重複をなくします。

要求された順序づけを満足するような索引が存在していない場合、ソートは必須です。また、ソートが索引スキャンよりもコストがかからない場合にも使用されます。ソートは通常、必要な行がいったん取り出された後、最後に実行される操作です。あるいは、結合やグループ化に先立ってデータのソートを行います。

行の数が多い場合や、ソートされたデータをパイプ処理することができない場合、この操作にはコストのかかる TEMPORARY 表の生成が求められます。

ソートに関する詳細は、*管理ガイド* をご覧ください。

パフォーマンス上の提案:

- ソート列に索引を追加することを検討してください。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

- 必要なデータだけを検索する述部を使用していることを確認してください。たとえば、述部で選択された値が必要とする戻り値の表の部分を表していることを確認します。
- システム TEMPORARY 表スペースのプリフェッチ・サイズが十分であること、つまり、入出力に制約されないことを確認します。 (このことをチェックするには、**「Statement」-> 「Show statistics」->「Table spaces」**の順に選択してください。)
- 頻繁に大規模なソートが必要とされる場合は、以下の構成パラメーターの値を増やすことを検討してください。
  - ソート・ヒープ・サイズ (sortheap)。このパラメーターを変更するには、Control Center でデータベースを右クリックし、そのポップアップ・メニューから*構成* を選択します。そこで表示されるノートブックから、「**パフォーマンス**」タブを選択します。
  - ソート・ヒープしきい値 (sheapthres)。このパラメーターを変更するには、 Control Center でデータベース・インスタンスを右クリックし、そのポップアップ・メニューから*構成* を選択します。そこで表示されるノートブックから、「**パフォーマンス**」タブを選択します。
- 統計が現行のものでない場合は、 RUNSTATS コマンドを使用して更新してください。

TBSCAN

演算子名: TBSCAN

意味: データ・ページから直接、必要なデータすべてを読み取って、行を取り出す表スキャン (関係スキャン) です。

オプティマイザーが索引スキャンよりもこの種のスキャンを選択するのは、次のような場合です。
- 走査される値の範囲が頻繁に発生する場合 (つまり、表の大部分をアクセスする必要がある場合)。
- 表が小さい場合。
- 索引のクラスタリングの度合いが低い場合。
- 索引が存在していない場合。

表と索引の走査に関する詳細は、*管理ガイド* をご覧ください。

パフォーマンス上の提案:
- 表が大きく、表の行の大半がアクセスされない場合は、表スキャンよりも索引スキャンの方が効率的です。この状況でオプティマイザーが索引スキャンを採用する可能性を高めるために、 選択述部のある列に索引を追加することを検討してください。

索引に関する詳しい指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

- 索引がすでにあるが使用されていない場合、それに先行する列のおのおのに選択述部があることを確認してください。選択述部があれば、次に、その索引のクラスタリングが高い値になっていることを確認してください。 (この統計を見るには、ソートの下にある表に対して表統計ウィンドウをオープンし、その「*索引*」ボタンを選択して、索引統計ウィンドウを表示させてください。)
- 表スペースのプリフェッチ・サイズが十分であること、つまり、入出力に縛られていないことを確認してください。 (このことをチェックするには、**「Statement」-> 「Show statistics」->「Table spaces」**の順に選択してください。)

  詳細については、*管理ガイド* のバッファー・プールへのデータのプリフェッチに関するセクションを参照してください。
- 統計が現行のものでない場合は、RUNSTATS コマンドを使用して更新してください。

  変位値および頻度値の統計は、述部の選択性に関する情報を提供しています。たとえば、これらの統計を使って、どんなときに、表走査より優先して索引走査を選ぶかを判別できます。これらの値を更新するには、WITH DISTRIBUTION 文節を付けて、表に対して **runstats** コマンドを使用します。

TEMP

演算子名: TEMP

意味: 他の演算子によるバックアウト読み取り (おそらく何度も行う) のために、データを TEMPORARY 表に保管する処置。その表は、SQL ステートメントが処理された後に除去されます (それ以前に除去されていない場合)。

この演算子は、副照会の評価や中間結果の保管のために必要になります。状況によっては (ステートメントが更新可能な場合など)、これは必ず必要です。

TQUEUE

演算子名: TQUEUE

意味: 複数のデータベース・エージェントが 1 つの照会を処理している場合に、1 つのデータベース・エージェントから別のデータベース・エージェントへデータを渡すのに使われる表キュー。複数データベース・エージェントは、並列処理が関係しているときに照会を処理するために使用されます。

表キューには次のような種類があります。

- **ローカル:** 単一のノード内にあるデータベース・エージェント間でデータを受け渡しするために、表キューが使用されます。ローカル表キューは、パーティション内並列処理に使用されます。
- **非ローカル:** 別々のノードにあるデータベース・エージェント間でデータを受け渡しするために、表キューが使用されます。

UNION

演算子名: UNION

意味: 複数の表から行のストリームを連結します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、連結する行のセットを定義する他の演算子 (走査および結合など) に連結します。

UNIQUE

演算子名: UNIQUE

意味: 指定された列について同じ値を持つ行の重複をなくします。

パフォーマンス上の提案:

- この演算子は、該当する列にユニーク索引が存在する場合に限り、不必要です。

  索引に関する指針については、Visual Explain のオンライン・ヘルプの「適切な索引を作成する」を参照してください。

UPDATE

演算子名: UPDATE

意味: 表の行の中のデータを更新します。

この演算子は必須の演算を表します。アクセス・プランのコストを改善するには、更新する行のセットを定義する他の演算子 (走査および結合など) に連結します。

付録 C. DB2 の概念

データベース

リレーショナル・データベースは、表の集合としてのデータを示します。表は、定義された列のセットおよび任意の行数から構成されます。各表のデータは論理的にリレーションシップするものであり、表の間にリレーションシップを定義することができます。データは、関係と呼ばれる数学的原則および操作 (INSERT、SELECT、 UPDATE など) に基づいて表示および操作することができます。

データベースは、データに加えてデータベース自体の構造の記述で、自己記述しています。これには、データの論理構造および物理構造を記述するシステム・カタログ表のセット、データベースに関連するパラメーター値を含む構成ファイル、および実行中のトランザクションとアーカイブできるトランザクションを記録するリカバリー・ログが含まれます。

データベースはローカルまたはリモートのいずれでも可能です。ローカル・データベースとは、使用しているワークステーション上に物理的に配置されているものですが、別のマシン上のデータベースはリモートと見なされます。

スキーマ

スキーマとは、データベース・オブジェクトのセット (表、ビュー、索引、別名など) をグループ化するために使用されるユニーク ID です。つまり、PAYROLL という名前の表を作成する場合、他の人がすでに同じ名前の表を作成していないかをデータベースで確認するのは退屈な仕事です。スキーマを使用すれば、各オブジェクトの名前はそのスキーマ内でのみユニークであればよいのです。

多くのデータベース・オブジェクトは 2 つの部分からなるオブジェクト名を持ち、最初の部分はスキーマ名、2 番目の部分はオブジェクトの名前です。オブジェクトの作成時に、オブジェクトを特定のスキーマに割り当てることができます。スキーマを指定しないとデフォルトのスキーマに割り当てられ、これは通常、オブジェクトを作成した人のユーザー ID です。たとえば、 Smith という名のユーザーは SMITH.PAYROLL という名前の表を持つことができます。

スキーマはデータベース内の 1 つのオブジェクトにもなります。これはスキーマ内に最初のオブジェクトが作成された時に作成されます。スキーマは個人が所有でき、所有者はスキーマ内のデータとオブジェクトへのアクセスをコントロールできます。

表

リレーショナル・データベースは、表の集合としてのデータを示します。表は、列と行 (一般的には、レコードと呼ばれる) に論理的に配置されたデータからなります。

それぞれの表は名前を持ち、表の中で、それぞれの列は名前を持ちます。表の行については特定の順序は維持されませんが、行の中の列の値によって決まる順序で検索することができます。 1 つの表の中のデータは論理的に関連するものです。すべてのデータベースおよび表データは、表スペースに割り当てられます。

付録 D. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032 東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。 それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。 お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。 これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのア

プリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、他社の商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Action Media、LANDesk、MMX、Pentium および ProShare は Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[ヤ行]

[ラ行]

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

DB2 Universal Database 製品に関する情報は、 www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、クライアント・ダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12