IBM

System i

データベース・パフォーマンスおよび Query 最適化

バージョン 6 リリース 1

IBM

System i

データベース・パフォーマンスおよび Query 最適化

バージョン 6 リリース 1

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、 379 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i5/OS のバージョン 6、リリース 1、モディフィケーション 0 (製品番号 5761–SS1) に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

IBM 発行のマニュアルに関する情報のページ

http://www.ibm.com/jp/manuals/

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典 :	System i Database Performance and Query Optimization Version 6 Release 1
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 2008.2

目次

データベース・パフォーマンスおよび Query 最適化

データベース・パフォーマンスの調整の目標は、照会の応答時間を最小限にし、ネットワーク・トラフィック、ディスク入出力、および CPU 時間を最小限にしてシステムのリソースを最大限に活用することです。この目的を達成するには、データの論理および物理構造を理解し、システム上で使用されるアプリケーションを理解し、データベースの使用に多くの矛盾があるとどのようにデータベース・パフォーマンスに影響を与えるかを必ず理解する必要があります。

パフォーマンスの問題を避ける最善の方法は、パフォーマンスの問題が、進行中の開発作業の一部であることを認識することです。有効なパフォーマンスの向上の多くは、データベース開発サイクルの開始時の注意深い設計によって実現されます。パフォーマンスを効果的に最適化するには、幅広い状況で最もパフォーマンスの向上が期待される領域を識別し、それらの領域についての分析に焦点を当てる必要があります。

この資料の中の例の多くは、 SQL または OPNQRYF 照会インターフェースによって作成された照会を示しています。特定の例にインターフェースが選択されていても、特に明示されていない場合、その照会インターフェースのみが機能することを示しているわけではありません。それは、使用可能な照会インターフェースの一例に過ぎません。ほとんどの例は、必要に応じてどの照会インターフェースにも簡単に書き直すことができます。

注: リーガル情報については、 377 ページの『コードに関するライセンス情報および特記事項』をお読みください。

V6R1 の新機能

以下に示す情報は、このリリースで追加または更新されたものです。

- 25 ページの『一時特殊ソート・リスト』
- 28 ページの『一時値リスト』
- 114 ページの『プラン・キャッシュを使用したパフォーマンスの最適化』
- QAQQINI 更新
- 183 ページの『使用カウントのリセット』
- 172 ページの『派生キー索引』
- 192 ページの『派生索引の使用』
- 55 ページの『全外部結合』
- 新規データベース・モニター・フォーマット

新機能と変更点の確認方法

技術的な変更点を明示するため、本書では以下を使用しています。

- » イメージにより、新規、または変更された情報の開始点を示します。
- « イメージにより、新規または変更された情報の終了点を示します。

本リリースの新機能または変更点についての情報を検索するには、「ユーザーへのメモ (Memo to users)」を参照してください。

データベース・パフォーマンスおよび Query 最適化用の PDF ファイル

この情報の PDF を表示し、印刷するにはこれを使用します。

本書の PDF 版を表示またはダウンロードするには、データベース・パフォーマンスおよび Query 最適化を選択します。

その他の情報

また、次の任意の PDF を表示または印刷できます。

- Preparing for and Tuning the SQL Query Engine on DB2® for i5/OS®
- SQL Performance Diagnosis on IBM® DB2 Universal Database™ for iSeries™

.

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ブラウザーで PDF を右マウス・ボタン・クリックする (上部のリンクを右マウス・ボタン・クリック)。
2. PDF をローカルに保管するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「**保存**」をクリックする。

Adobe® Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Reader がシステムにインストールされていることが必要です。無料のコピーを、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) からダウンロードできます。

照会エンジンの概説

DB2 for i5/OS は照会を処理するための 2 つの照会エンジン、Classic Query Engine (CQE) と SQL Query Engine (SQE) を備えています。

CQE は、非 SQL インターフェースである OPNQRYF、 Query/400、および QQQQry API からの照会を処理します。 ODBC、JDBC、CLI、QUERY マネージャー、 Net.Data®、 RUNSQLSTM、および組み込みまたは対話式 SQL などの SQL ベースのインターフェースは、 SQE によって処理されます。使いやすくするために、照会を CQE または SQE のどちらで処理するかについての経路指定の決定は、広範囲に渡って行われ、システムによって制御されます。要求するユーザーまたはアプリケーション・プログラムは、この動作を制御することも影響を及ぼすこともできません。しかし、エンジンについて、および照会がどのパスをとるかを決定するプロセスについてさらに理解することによって、照会のパフォーマンスについてさらに理解することができます。

SQE 内で、いくつかのコンポーネントが作成され、その他の既存のコンポーネントが更新されました。さらに、CQE の下ではサポートされていない SQE で、新しいデータ・アクセス方式が可能です。

関連情報

組み込み SQL プログラミング

SQL プログラミング

Query (QQQQRY) API

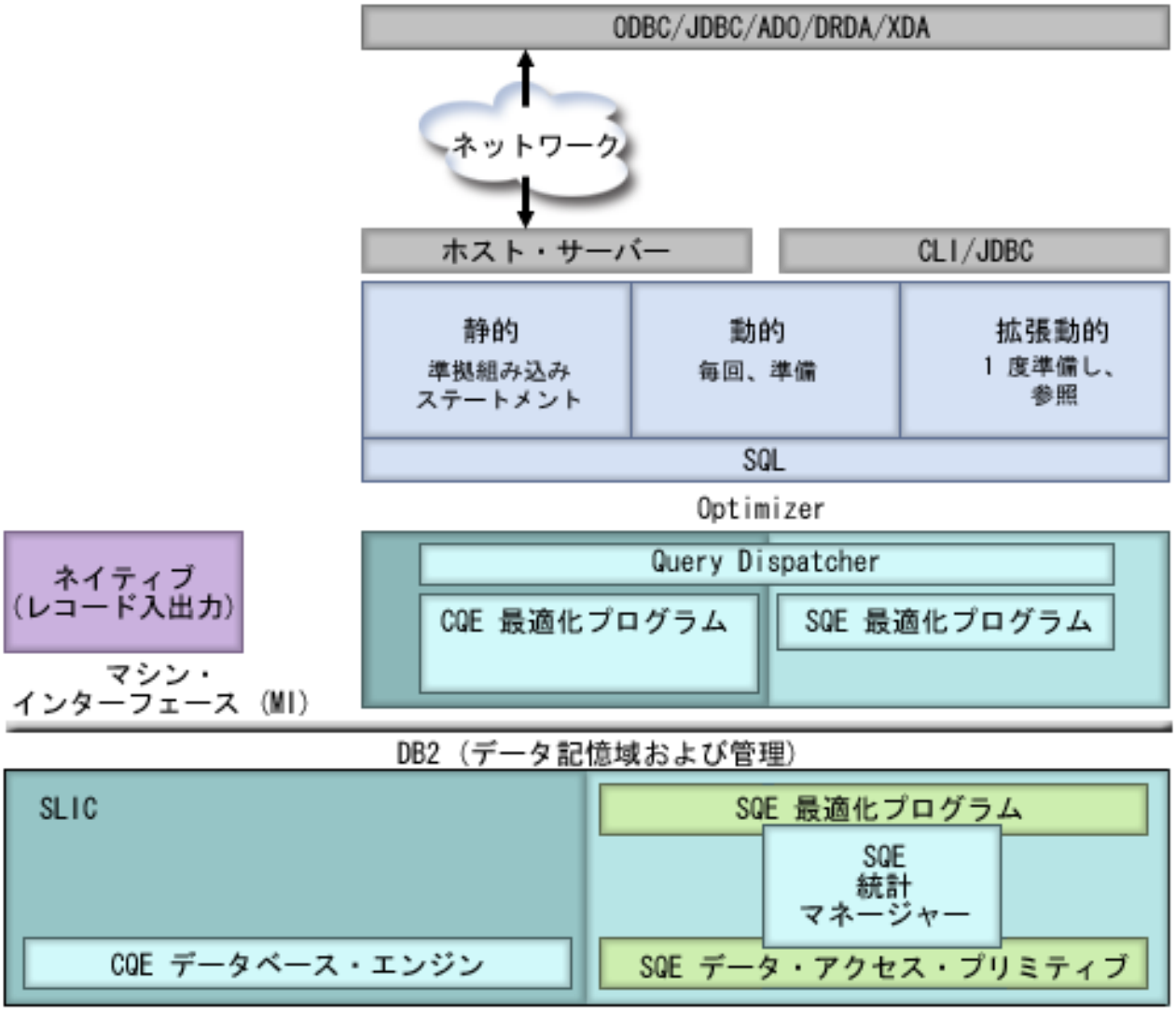
QUERY ファイル・オープン (OPNQRYF) コマンド

SQL ステートメントの実行 (RUNSQLSTM) コマンド

SQE および CQE エンジン

DB2 for i5/OS における照会管理機能および処理のインプリメンテーションを十分に理解するには、CQE と SQE との間で、照会がどのようにインプリメントされたかについて参照することが重要です。

下記の図には、DB2 for i5/OS のアーキテクチャーの概要が表示されています。この図には、CQE と SQE の間の概要説明、照会処理が照会ディスパッチャーによってどのように管理されるか、および各 SQE コンポーネントがフィットする場所が示されています。各 SQE コンポーネントの機能の区切りは、非常にはっきりしています。設計志向の、この責任の区切りによって、IBM は必要なときに個々の SQE コンポーネントにさらに簡単に機能拡張を行うことができます。 SQE 最適化プログラムのコンポーネントのほとんどが MI より下にインプリメントされることに注意してください。これは、パフォーマンス効率が拡張されることを意味します。



Query dispatcher

dispatcher の機能は、照会の属性に応じて、照会の要求を CQE か SQE のどちらかに経路指定します。すべての照会はこのディスパッチャーによって処理され、バイパスすることはできません。

現行では、Query dispatcher は SQL ステートメントが以下のいずれかを参照または含んでいることを検出した場合に、ステートメントを CQE に経路指定します。

- INSERT WITH VALUES ステートメントまたは副選択ステートメントを使用した INSERT のターゲット
- 論理ファイル
- 読み取りトリガーを使用するテーブル
- 1000 を超えるデータ・スペースを持つ読み取り専用照会、または 256 を超えるデータ・スペースを持つ更新可能照会
- DB2 マルチシステム・テーブル
- 非 SQL 照会、たとえば QQQQry API、Query/400、または OPNQRYF

dispatcher には、最初 SQE に経路指定されていた SQL 照会の経路を CQE に再指定する組み込み機能も備わっています。 QAQQINI オプション IGNORE_DERIVED_INDEX が明示的にパラメーター値 *YES に設定された場合、最適化プログラムが SELECT/OMIT DDS キーワードを指定した、その上に作成された論理ファイルを持つテーブル・オブジェクトを処理するときには、照会は通常 SQE から CQE に戻されます。

新しい機能が今後追加されていくにつれて、ディスパッチャーが SQE に経路指定する照会は多くなり、CQE に経路指定する照会は少なくなります。

関連資料

71 ページの『MQT でサポートされる機能』
MQT にはほとんどどんな照会でも入れることができますが、最適化プログラムでは、MQT とユーザー指定照会のマッチング時に照会機能のうちの一部しかサポートされていません。 SQE 最適化プログラムでは、ユーザー指定照会と MQT 照会の両方がサポートされていなければなりません。

統計マネージャー

CQE では、統計の検索は最適化プログラムの機能です。テーブルについての情報を得る必要がある場合、最適化プログラムはテーブル記述を参照して行数とテーブル・サイズを取得します。索引が使用できる場合、最適化プログラムはテーブル内のデータに関する情報を抽出する場合もあります。 SQE では、統計の収集および管理は統計マネージャーと呼ばれる別個のコンポーネントが処理します。統計マネージャーは、CQE とまったく同じ統計的なソースの効力を高めますが、さらにより多くのソースおよび機能を追加します。

統計マネージャーは実際には照会を実行または最適化しません。むしろ、統計マネージャーは、メタデータへのアクセス、および照会を最適化するのに必要な他の情報へのアクセスを制御します。統計マネージャーはこの情報を使用して、Query 最適化プログラムが提出する質問に応答します。統計マネージャーは常時、最適化プログラムに応答を提供します。実際の既存の統計情報に基づく応答を提供できない場合には、事前定義された応答を提供するよう設計されます。

一般に統計マネージャーは、以下の情報を収集および追跡します。

値のカーディナリティー
テーブルの単一列または複数列における、特定値の固有または別個の出現数のことです。

選択　この情報は、ヒストグラムとしても知られていますが、指定された任意の選択述部または組み合わされた述部によって、いくつの行が選択されるかを示しています。抽出方式を使用して、テーブルの指定された列における値の選択および分配を記述します。

頻度　列の値の頻度の限界値 *nn* と、各値の頻度です。この情報は、統計的抽出方式を使用することによって得られます。組み込まれたアルゴリズムはデータがスキューする可能性を除去します。たとえば、統計値に影響を及ぼす可能性のある NULL 値およびデフォルト値は計算に入れられません。

メタデータ情報
これには、テーブルの行の総数、テーブルを越えて存在する索引、およびどの索引が特定の照会をインプリメントするのに役立つかが含まれています。

IO 操作の推定値
これは、テーブルまたは識別された索引を処理するのに必要とされる IO 操作の量の推定値です。

統計マネージャーはハイブリッド方式を使用することにより、データベース統計を管理します。この情報の大部分は既存の索引から得ることができます。必要な統計を既存の索引から収集できない場合、統計情報はテーブルの個々の列で構成され、テーブルの一部として内部に格納されます。デフォルトでは、この情報はシステムによって自動的に収集されますが、統計の収集を手動で制御することができます。ただし、索引の場合とは異なり、統計はテーブル内のデータが変更するときに、即時に保守されることはありません。

関連資料

163 ページの『統計マネージャーによる統計の収集』
先に言及したように、統計の収集は統計マネージャーと呼ばれる別個のコンポーネントが処理します。統計情報を Query 最適化プログラムが使用して、照会の最適アクセス・プランを判別することができます。 Query 最適化プログラムのアクセス・プランの選択はテーブルにある統計情報に基づいているため、この情報が現行のものとなっていることが重要です。

プラン・キャッシュ

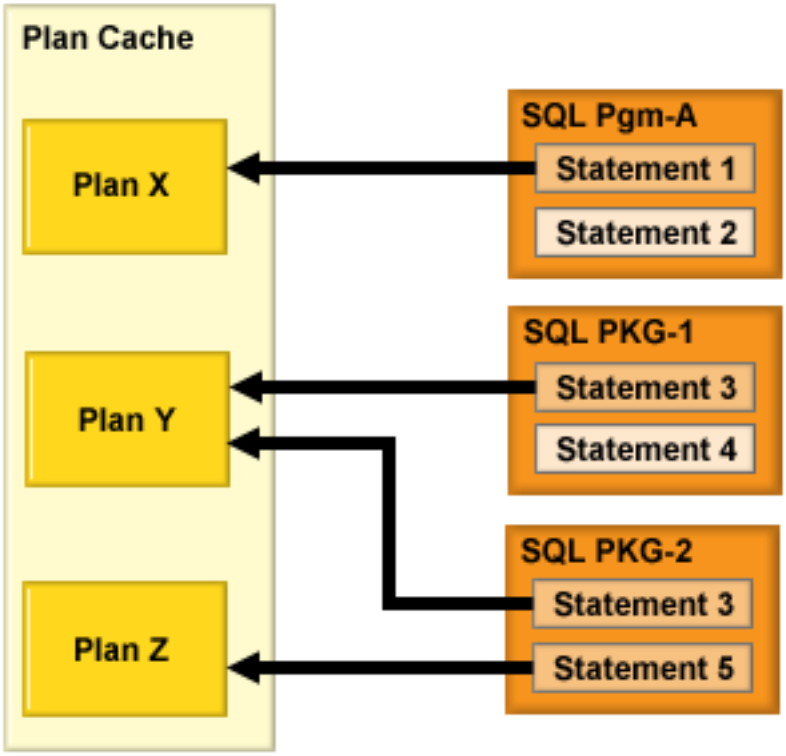
プラン・キャッシュは、SQE によって最適化された照会用のアクセス・プランを含むリポジトリーです。

CQE によって生成されたアクセス・プランはプラン・キャッシュに格納されません。その代わりに、SQL パッケージ、システム全体のステートメント・キャッシュ、およびジョブのキャッシュに格納されます。プラン・キャッシュの目的は、以下を行うことです。

- 同じ照会が再実行されるときに照会アクセス・プランの再利用を簡単にする
- 将来 Query 最適化で使用するときのためにランタイム情報を保管する
- 分析およびチューニング用にパフォーマンス情報を提供する

アクセス・プランは一度作成されると、照会がどこで発生するかにかかわらず、すべてのユーザーおよびすべての照会によって使用可能です。さらに、アクセス・プランが調整されるとき、たとえば索引が作成されるとき、すべての照会がこの更新されたアクセス・プランから恩恵を受けることができます。これによって、照会を再度最適化する必要がなくなり、その結果、効率が高まります。

下記の図は、プラン・キャッシュに格納された照会アクセス・プランの再使用の概念を示しています。



上に示されているように、プラン・キャッシュは、照会の要件を満たす有効なアクセス・プランが存在するかどうかを判別するために、照会が実行されるごとに問い合わせを受けます。有効なアクセス・プランが見つかった場合には、照会をインプリメントするために使用されます。見つからなかった場合、今後の使用のために新しいアクセス・プランがプラン・キャッシュに作成され、格納されます。プラン・キャッシュは新しい照会アクセス・プランの作成時にそれによって自動的に更新されるか、または新しい統計または索引が使用できる状態になった時点で (次回の照会実行時に) 既存のプランに対して更新されます。プラン・キャッシュはまた、照会の実行時にデータベースによって、ランタイム情報で自動更新されます。プラン・キャッシュは全体サイズが 512 メガバイト (MB) で作成されます。プラン・キャッシュのそれぞれの項目には、元の照会、最適化された照会アクセス・プラン、および照会の実行中に収集された累積ランタイム情報が入ります。さらに、照会ランタイム・オブジェクトのいくつかのインスタンスがプラン・キャッシュ項目とともに保管されます。これらのランタイム・オブジェクトは、照会の実行に使用される実際の実行可能ファイルおよび一時ストレージ・コンテナー (ハッシュ・テーブル、ソート、一時索引など)です。

プラン・キャッシュが指定されたサイズを超過する場合、バックグラウンド・タスクはプラン・キャッシュからプランを除去するよう自動的にスケジュールされています。アクセス・プランは、アクセス・プランの経過日数、使用頻度、および照会の実行によって消費された累積リソース (CPU/IO) に基づいて削除されます。プラン・キャッシュに格納されるアクセス・プランの総数は、実行されている SQL ステートメントの複雑さに大いに依存しています。あるテスト環境では、プラン・キャッシュに 10,000 から 20,000 の固有のアクセス・プランが一般的に格納されていました。プラン・キャッシュは、システムの初期プログラム・ロード (IPL) が実行されるときに、消去されます。

複数のアクセス・プランを単一の SQL ステートメント用に使用することができます。 SQL ステートメント自体がプラン・キャッシュの 1 次ハッシュ・キーですが、異なる環境設定によって異なるアクセス・プランをプラン・キャッシュに格納することができます。これらの環境設定には、たとえば次のものが含まれます。

- 同じ照会に対する異なる SMP 程度の設定
- 照会テーブル用に指定された異なるライブラリー・リスト
- 現行のプールの使用可能メモリーをジョブが共用することに関する異なる設定
- 異なる ALWCPYDTA 設定
- 選択で使用されるホスト変数値の変更に基づく異なる選択度 (WHERE 文節)

現行では、プラン・キャッシュは最大 3 つの異なるアクセス・プランを同じ SQL ステートメント用に保守できます。新しいアクセス・プランが同じ SQL ステートメント用に作成される場合、古いアクセス・プランは、新規アクセス・プランのためのスペースを空けるために破棄されます。しかし、既存のアクセス・プランを無効にできる、ある条件があります。これらには、例えば次のものが含まれます。

- QAQQINI テーブルまたは「SQL スクリプトの実行」に REOPTIMIZE_ACCESS_PLAN(*YES) または (*FORCE) を指定する
- アクセス・プランが参照するテーブルを削除または再作成する
- アクセス・プランによって使用される索引を削除する

関連資料

214 ページの『ALWCPYDTA パラメーターの使用によるデータベース・パフォーマンスの向上』
ある種の複雑な照会では、索引を使用または作成する代わりに分類またはハッシュ方式を使用して照会の実行を行うと、パフォーマンスを向上できる場合があります。

135 ページの『照会の属性の変更』
照会属性の変更 (CHGQRYA) CL コマンドにより、あるいは System i™ ナビゲーター照会属性の変更インターフェースを使用して、一定のジョブの間に実行する照会の属性の各種タイプを変更することができます。

114 ページの『プラン・キャッシュを使用したパフォーマンスの最適化』
プラン・キャッシュには、データベース中で実行される SQE 照会に関する豊富な情報が含まれます。その内容は、System i ナビゲーター GUI インターフェースによって表示できます。プラン・キャッシュの特定の部分も変更できます。

DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式

データ・アクセス方式は、照会を処理し、データにアクセスするために使用されます。

通常、照会エンジンには、以下に示す 2 種類の未加工の素材があります。これらを使用して、照会要求を満たすことができます。

- 照会するデータを含むデータベース・オブジェクト
- データを取り出して使用可能な情報に変換するための実行可能な指示または操作

照会のソース素材として使用できる永続データベース・オブジェクトとしては、テーブルと索引 (2 進基数索引およびコード化ベクトル索引) の 2 つのタイプしか実際にはありません。加えて、この照会エンジンは一時オブジェクトまたはデータ構造を作成して、アクセス・プランの実行中に暫定の結果や参照を保持する必要がある場合もあります。 DB2 Symmetric Multiprocessing フィーチャーを使用すれば、最適化プログ

ラムは並列処理を含む付加的なデータ検索方式を使用することができます。最終的に、最適化プログラムはこうしたオブジェクトを取り扱うのに特定の方式を使用します。

永続オブジェクトおよびアクセス方式

照会エンジンが使用するデータベース・オブジェクトおよびアクセス方式は、永続オブジェクトおよび一時オブジェクトを取り扱うのに使用する 3 つの基本的なタイプの操作 (作成、走査、プローブ) に分類できます。

以下の表には、各オブジェクトとそのオブジェクトに対して実行できるアクセス方式がリストされています。 表に示されているシンボルは、Visual Explain で使用されるアイコンです。

表 1. 永続オブジェクトのデータ・アクセス方式

永続オブジェクト	走査操作	プローブ操作
テーブル	テーブル走査	テーブル・プローブ
基数索引	基数索引走査	基数索引プローブ
コード化ベクトル索引	コード化ベクトル索引記号テーブル走査	コード化ベクトル索引プローブ

テーブル

SQL テーブルまたは物理ファイルは、照会の基本オブジェクトです。照会の結果セットを作成するのに使用されるデータのソースを表します。 ユーザーによって作成され、FROM 文節 (または OPNQRYF FILE パラメーター) で指定されます。

最適化プログラムは、照会を満たすためにデータをテーブルから抽出する最も効率的な方法を判別します。これには、テーブルの走査･テーブルのプローブ、またはデータを抽出するための索引の使用が含まれます。

Visual Explain アイコン:

テーブル走査:

テーブル走査は、テーブルに対して実行できる最も簡単でシンプルな操作です。テーブル内の行が照会で指定された選択基準を満たすかどうかを判別するために、すべての行を順次処理します。 この操作を実行すると、テーブルの入出力スループットが最大になります。

テーブル走査操作は、処理のために主記憶域に可能な限り多くの行を移動するので、大規模な入出力が必要となります。 さらに、テーブル走査操作は行を主記憶域にページングするのを決して待機しないようにするため、非同期的にデータをプリフェッチします。しかし、テーブル走査には、照会を満たすためにすべての行を処理しなければならないという欠点もあります。 入出力を同期的に実行する必要がなければ、走査操作自体はとても効率的です。

表 2. テーブル走査属性

データ・アクセス方式	**テーブル走査**
説明	テーブルからすべての行を読み取り、テーブル内の各行に選択基準を適用します。 テーブル内の行処理順序については特に保証はありませんが、通常は順次処理されます。
利点	• ページが順次走査されるので、非同期的な行のプリフェッチによって、ページ入出力操作が最小化されます。 • データを効率的にフェッチするため、大規模入出力が必要となります。
考慮事項	• 照会の選択に関係なく、テーブル内のすべての行が調べられます。 • 削除済みとしてマークを付けられた行が、全く選択されない場合でも引き続き記憶域にページングされます。テーブルを再編成すると、削除済み行を除去できます。
使用される可能性が高い場合	• テーブルから多くの行が戻ってくることが予想される場合 • 走査のために必要な大規模入出力数の方が、テーブルをプローブするのに必要な小規模入出力数よりも少ない場合

SQL ステートメントの例	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01'AND 'E01'`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4329 — ファイル EMPLOYEE に到着順アクセスが使用された`
	• PRTSQLINF:
	`SQL4010 — テーブル 1 に対するテーブル走査アクセス。`

SMP 並列の使用可能化	はい
別名	テーブル走査、プリロード
Visual Explain アイコン	

関連概念

50 ページの『ネストされたループ結合の実施』

DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

テーブル・プローブ:

テーブル・プローブ操作は、テーブルから行番号に基づいて特定の行を取り出すのに使用します。行番号は、テーブルの行番号を生成する他の操作によって、テーブル・プローブ・アクセス方式に提供されます。

これには、索引操作、一時行番号リストまたはビットマップが含まれます。テーブル・プローブの処理は通常ランダムです。必要な行を取り出すだけですので入出力は小規模で、無関係な行を移動しようとすること

はありません。これは、照会を満たすのに必要な行だけを処理すればよいので、すべての行を処理する必要のある走査方式よりも、小規模な結果セットの処理の場合にとても効率的です。しかし、行番号の順序は事前に分からないので、データを主記憶域に移動するためにプリフェッチはほとんど実行できません。これにより、このアクセス方式に関連付けられている入出力のほとんどは、同期的に実行されることになります。

表 3. テーブル・プローブ属性

データ・アクセス方式	**テーブル・プローブ**
説明	特定の行番号に基づいて、テーブルから単一行を読み取ります。テーブルに対してランダム入出力が実行され、行が抽出されます。
利点	• 必要のない行を記憶域にページングしないようにするため、必要とされる入出力は小規模です。 • テーブル・プローブで処理するための行番号を生成する、任意のアクセス方式と共に使用できます。
考慮事項	同期的なランダム入出力のため、大量の行が選択されるとこのプローブは十分に実行できなくなる可能性があります。
使用される可能性が高い場合	• (索引または一時行番号リストのいずれかからの) 行番号が使用されていますが、照会をさらに処理するために基礎となっているテーブル行からのデータが必要となる場合 • 値の残った選択または推定を処理する場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (LastName)`
	
	`SELECT * FROM Employee`
	`  WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`  AND LastName IN ('Smith', 'Jones', 'Peterson')`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	テーブル・プローブの使用を示す特定のメッセージはありません。この例のメッセージは、テーブル・プローブ操作を実行するのに使用される行番号を生成するデータ・アクセス方式の使用を示しています。 • 最適化プログラム・デバッグ: CPI4328 — 照会によってファイル X1 のアクセス・パスが使用された。 • PRTSQLINF: SQL4008 — テーブル 1 に索引 X1 が使用された。 SQL4011 — テーブル 1 に索引スキャン・キー行位置 (プローブ) が使用されました。
SMP 並列の使用可能化	はい
別名	テーブル・プローブ、プリロード
Visual Explain アイコン	

基数索引

SQL 索引 (またはキー順アクセス・パス) は永続オブジェクトで、走査操作またはプローブ操作用に最適化プログラムによってテーブルで作成され、データの順序付けされた表示を提供するのに使用されます。

テーブル内の行は、オブジェクトの作成時に指定されたキー列に基づいて、索引内で順序付けされます。最適化プログラムがキー列を照会にマッチングさせると、任意の選択、順序付け、グループ化、または結合要件を満たすのに役立つ基数索引を使用できるようになります。

一般に、索引操作の使用には、索引キーを見つけることができない照会を満たすのに必要な任意の列へのアクセスを提供するために、テーブル・プローブ操作も含まれます。テーブルに対する照会要求を満たすのに必要なすべての列が索引のキーとして見つかる場合には、テーブル・プローブは必要ではなく、その照会では索引アクセスのみが使用されます。 テーブル・プローブを使用しないことによって、照会のコストをかなり節約することができる場合があります。 通常テーブル・プローブに関連する入出力は、コストのかかる同期のランダム入出力です。

Visual Explain アイコン:

基数索引走査:

基数索引走査操作は、テーブルから行をキー順に取り出すのに使用します。テーブル走査と同様に索引内の行すべてが順次処理されますが、結果の行番号はキー列に基づいて順序付けされます。

順序付けされた行は、照会要求 (順序付けまたはグループ化) の一部を満たすために最適化プログラムによって使用されます。さらに、テーブルのすべての行ではなく索引キーに対して選択を実行し、より速いスループットを提供するのにも使用できます。 この索引に関連する入出力には索引キーだけが含まれるので、索引に対する 1 回の入出力で、多くの列を持つ 1 つのテーブルの行を記憶域にページングするよりも、多くの行をページングすることが一般に可能です。

表 4. 基数索引走査属性

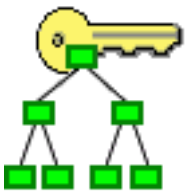
データ・アクセス方式	**基数索引走査**
説明	索引に関連付けられたすべてのキーを順次走査して処理します。すべての選択が、テーブル行の前に索引のすべてのキー値に適用されます。
利点	• 任意の選択にマッチングする索引項目のみが、継続して処理されます。 • 索引キー値からすべてのデータを抽出できるので、テーブル・プローブの必要性がありません。 • 索引のキーに基づいて、行を順序通りに戻します。
考慮事項	照会を満たすのに必要な残りの列を抽出するため、通常テーブル・プローブを実行することが必要となります。テーブル・プローブにはランダム入出力が関連するので、多くの行数が選択されると十分に実行できなくなる可能性があります。
使用される可能性が高い場合	• 索引から戻されるよう依頼されている、または予想されるのがわずかな行である場合 • 照会に対して行の順序付けが必要な場合 (たとえば、順序付けまたはグループ化の場合) • 選択列が、索引の主要なキー列とマッチングしない場合

表 4. 基数索引走査属性 (続き)

データ・アクセス方式	基数索引走査

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (LastName, WorkDept)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`ORDER BY LastName`
	`OPTIMIZE FOR 30 ROWS`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`

SMP 並列の使用可能化	はい
別名	索引走査 索引走査、プリロード 索引走査、区別 索引走査区別、プリロード 索引走査、キー選択
Visual Explain アイコン	

関連資料

214 ページの『ALWCPYDTA パラメーターの使用によるデータベース・パフォーマンスの向上』
ある種の複雑な照会では、索引を使用または作成する代わりに分類またはハッシュ方式を使用して照会の実行を行うと、パフォーマンスを向上できる場合があります。

基数索引プローブ:

基数索引プローブ操作は、テーブルから行をキー順に取り出すのに使用します。基数索引プローブと基数索引走査の主な相違点は、取り出されている行をサブセット化するプローブ操作では、戻される行が最初に識別される必要があるという点です。

最適化プログラムは、索引の主要なキーと、一部の選択またはすべての選択に使用する列とがマッチングするか試行します。 その後、直接索引のキー値でプローブするのに使用可能な一連の範囲に、その選択を再書き込みします。 一連の範囲からのキーだけが主記憶域にページングされます。 その結果、プローブ操作によって生成された行番号は、索引キーに対する任意の残りの選択で、またはテーブル・プローブ操作でさらに処理できます。これにより、選択を満たす索引の行のみにとても迅速にアクセスできます。

基数索引プローブの主な機能は索引キーに対して迅速な選択方法を提供することですが、照会の他の部分 (順序付けまたはグループ化) を満たすために行の順序付けも最適化プログラムは使用できます。この索引

に関連する入出力は、選択にマッチングする索引行だけに対して行われ、プローブ選択にマッチングしない行では無関係な処理は行われません。照会の結果セットの一部ではない行に対する入出力は行われないため、これはこの操作の主な利点の 1 つです。

表 5. 基数索引プローブ属性

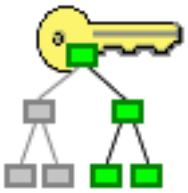
データ・アクセス方式	**基数索引プローブ**
説明	索引は、一連の範囲に書き込まれた選択基準に基づいて、迅速にプローブされます。 選択基準を満たすキーだけが、テーブルの行番号を生成するのに使用されます。
利点	• 任意の選択にマッチングする索引項目のみが、継続して処理されます。 • 選択行に対して非常に迅速にアクセスできるようになります。 • 索引キー値からすべてのデータを抽出できるので、テーブル・プローブの必要性がありません。 • 索引のキーに基づいて、行を順序通りに戻します。
考慮事項	照会を満たすのに必要な残りの列を抽出するため、通常テーブル・プローブを実行することが必要となります。テーブル・プローブにはランダム入出力が関連するので、多くの行数が選択されると十分に実行できなくなる可能性があります。
使用される可能性が高い場合	• 索引から戻されるよう依頼されている、または予想されるのがわずかな行である場合 • 照会に対して行の順序付けが必要な場合 (たとえば、順序付けまたはグループ化の場合) • 選択列が、索引の主要なキー列とマッチングする場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (LastName, WorkDept)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`AND LastName IN ('Smith', 'Jones', 'Peterson')`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`
	`SQL4011 -- テーブル 1 に索引スキャン・キー行位置が使用された。`

SMP 並列の使用可能化	はい
別名	索引プローブ 索引プローブ、プリロード 索引プローブ、区別 索引プローブ区別、プリロード 索引プローブ、キー位置決め 索引走査、キー行位置決め

表 5. 基数索引プローブ属性 (続き)

データ・アクセス方式	基数索引プローブ
Visual Explain アイコン	

次の例は、最適化プログラムが基数索引プローブ・アクセス方式を選択する可能性のある照会を示しています。

```
CREATE INDEX X1 ON Employee (LastName, WorkDept)

SELECT * FROM Employee
WHERE WorkDept BETWEEN 'A01' AND 'E01'
AND LastName IN ('Smith', 'Jones', 'Peterson')
OPTIMIZE FOR ALL ROWS
```

この例では、最適化プログラムは索引 X1 を使用して、 LastName 列および WorkDept 列の両方を作成する選択にマッチングする最初の索引項目を配置 (プローブ) します。 この選択は、索引 X1 から使用される主要なキー列すべてとマッチングする、一連の範囲に再書き込みされます。 その後、プローブは主要なキーすべての複合の連結値を基にします。この再書き込みされた SQL の疑似 SQL は、次のようになります。

```
SELECT * FROM X1
WHERE X1.LeadingKeys BETWEEN 'JonesA01' AND 'JonesE01'
      OR X1.LeadingKeys BETWEEN 'PetersonA01' AND 'PetersonE01'
      OR X1.LeadingKeys BETWEEN 'SmithA01' AND 'SmithE01'
```

プローブ操作を満たすキー項目すべては、索引に関連付けられたテーブル (たとえば、Employee) の行番号を生成するのに使用されます。行番号はテーブル・プローブ操作によって使用され、テーブルでランダム入出力を実行して照会の結果を生成します。 この処理は、索引プローブ操作を満たすすべての行が処理されるまで続行されます。この例では、すべての処理された索引項目、および取り出された行が索引プローブ基準を満たしたことに注意してください。追加の選択が加えられ、索引プローブ操作でそれを実行できない場合 (索引の主要なキー列の一部ではない列に対する選択など)、最適化プログラムはプローブされた値の範囲で索引走査操作を実行します。これにより、テーブル・プローブ操作の前に引き続き選択を実行できます。

関連概念

50 ページの『ネストされたループ結合の実施』
DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

関連資料

214 ページの『ALWCPYDTA パラメーターの使用によるデータベース・パフォーマンスの向上』
ある種の複雑な照会では、索引を使用または作成する代わりに分類またはハッシュ方式を使用して照会の実行を行うと、パフォーマンスを向上できる場合があります。

コード化ベクトル索引

コード化ベクトル索引は、特殊キー値にコードを割り当てた後、これらの値をベクトルで表すことによって、テーブルへのアクセスを提供する永続オブジェクトです。

ベクトルのサイズは、基礎となっているテーブルの行数とマッチングします。各ベクトル項目は、同じ位置にあるテーブル行番号を表します。特殊キー値を表すために生成されるコードの長さは、表す必要のある特殊値の数値に応じて、 1、2、または 4 バイトとなります。コード化ベクトル索引 (EVI) は、サイズが小さく比較的単純なため、大規模なデータをとても効率的に処理するのに使用できます。

コード化ベクトル索引はテーブルに保管されている値を表すのに使用されますが、テーブルへのアクセスを直接取得するためにこの索引を使用することはできません。代わりにコード化ベクトル索引は、一時行番号リストまたは一時行番号ビットマップのいずれかを生成するためにのみ使用できます。こうした一時オブジェクトはテーブル・プローブと共に使用して、照会が処理する必要のあるテーブル内の行を指定することができます。 (基数索引と比較して) コード化ベクトル索引と関連付けられたテーブル・プローブの主な相違点は、テーブルと関連付けられたページングが非同期であるということです。それでこの入出力は、選択済み行のグループ化の利点を生かすためにいっそう効率的にスケジュールすることができるようになりました。行が選択されていないテーブルの大部分の箇所は、スキップオーバーできます。

Visual Explain アイコン:

コード化ベクトル索引記号テーブル走査:

コード化ベクトル索引記号テーブル走査操作は、索引の記号テーブルの部分から項目を取り出すために使用されます。

記号テーブルのすべての項目 (シンボル) は順次走査されます。ただし、結果として生成される項目の順序は特に保証はありません。記号テーブルは、照会要求の GROUP BY または DISTINCT の部分を満たすために最適化プログラムによって使用されます。選択はすべて、記号テーブルのすべての項目に適用されます。項目はすべて、索引のベクトルの部分または EVI が作成される関連テーブルのレコードにアクセスすることなく、記号テーブルの部分から直接取り出されます。

表 6. コード化ベクトル索引記号テーブル走査属性

データ・アクセス方式	コード化ベクトル索引記号テーブル走査
説明	索引に関連付けられたすべての記号テーブル項目を順次走査して処理します。選択はすべて、記号テーブルのすべての項目に適用されます。選択された項目は、ベクトルまたは関連テーブルにアクセスすることなく、直接取り出されます。
利点	• 事前に要約された結果を容易に使用できます。 • 記号テーブルの固有値のみを処理し、テーブル・レコードの処理を回避します。 • 索引固有キー値からすべてのデータを抽出するので、テーブル・プローブまたはベクトル走査の必要性がありません。

表 6. コード化ベクトル索引記号テーブル走査属性 (続き)

データ・アクセス方式	コード化ベクトル索引記号テーブル走査
考慮事項	結果として生成されるグループの数が、基礎表のレコードの数と比べて比較的小さい場合の照会のグループ化で、パフォーマンスの劇的な向上が見られます。記号テーブルが非常に大きいなど、多くのグループが関係する場合、特に記号テーブルの大部分がオーバーフロー域に入れられた場合に、実行パフォーマンスが低下する可能性があります。
使用される可能性が高い場合	• 単一の表から GROUP BY、DISTINCT、COUNT、または COUNT DISTINCT を求め、参照される列がキー定義にある場合。 • 基礎表のレコードの数と比べて、キー定義の列の固有値の数値が小さい場合。 • 照会内に選択 (Where 文節) が存在しない場合、または選択がそれほど結果セットを減らさない場合。

SQL ステートメントの例	`CREATE ENCODED VECTOR INDEX EVI1 ON Sales (Region)`
	
	例 1
	`SELECT Region, count(*)`
	`FROM Sales`
	`GROUP BY Region`
	`OPTIMIZE FOR ALL ROWS`
	
	例 2
	`SELECT DISTINCT Region`
	`FROM Sales`
	`OPTIMIZE FOR ALL ROWS`
	
	例 3
	`SELECT COUNT(DISTINCT Region)`
	`FROM Sales`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4328 -- QUERY によってファイル EVI1 のアクセス・パスが使用された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 EVI1 が使用されました。`

SMP 並列の使用可能化	いいえ。索引作成中に「グループ化」が既に実行されたので、通常は重要ではありません。
別名	コード化ベクトル索引テーブル走査、プリロード
Visual Explain アイコン	

コード化ベクトル索引プローブ:

コード化ベクトル索引 (EVI) は、一連の範囲に書き込まれた選択基準に基づいて、迅速にプローブされます。一時行番号リストまたはビットマップのいずれかを作成します。

表 7. コード化ベクトル索引プローブ属性

データ・アクセス方式	コード化ベクトル索引プローブ
説明	コード化ベクトル索引 (EVI) は、一連の範囲に書き込まれた選択基準に基づいて、迅速にプローブされます。一時行番号リストまたはビットマップのいずれかを作成します。
利点	• 任意の選択にマッチングする索引項目のみが、継続して処理されます。 • 選択行に対して非常に迅速にアクセスできるようになります。 • 昇順で行番号を戻すので、テーブル・プローブが行のプリフェッチをより積極的に操作できるようになります。
考慮事項	通常 EVI は単一キーに対して作成します。列の識別が容易でオーバーフロー・パーセンテージが高ければ高いほど、コード化ベクトル索引の利点が減ります。EVI では、テーブル・プローブが EVI プローブ操作の結果に対して実行される必要があります。
使用される可能性が高い場合	• 選択列が、索引の主要なキー列とマッチングする場合 • コード化ベクトル索引が存在し、テーブルに対する入出力の削減により EVI のプローブおよび一時行番号リストをすべて取り込むことによる余分のコストを払う価値がある場合

SQL ステートメントの例	`CREATE ENCODED VECTOR INDEX EVI1 ON`
	`  Employee (WorkDept)`
	`CREATE ENCODED VECTOR INDEX EVI2 ON`
	`  Employee (Salary)`
	`CREATE ENCODED VECTOR INDEX EVI3 ON`
	`  Employee (Job)`
	
	`SELECT *`
	`FROM Employee`
	`WHERE WorkDept = 'E01' AND Job = 'CLERK'`
	`AND Salary = 5000`
	`OPTIMIZE FOR 99999 ROWS`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4329 -- ファイル EMPLOYEE に到着順アクセスが使用された。`
	`CPI4338 -- 3 アクセス・パスがファイル EMPLOYEE のビットマップ処理に使用された。`
	• PRTSQLINF:
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI1 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI2 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI3 が使用された。`

SMP 並列の使用可能化	はい
別名	コード化ベクトル索引プローブ、プリロード
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムは、この照会が使用する各コード化ベクトル索引用の一時行番号ビットマップを作成することを選択します。各ビットマップは、この索引のキー列での選択にマッチングす

る行のみを識別します。それで、こうした一時行番号ビットマップは相互にマージされて、各索引から選択された行の論理積を判別します。この論理積は、最終一時行番号ビットマップを形成するのに使用されます。このビットマップは、選択された行のテーブルに対する入出力ページングをスケジュールするのに役立ちます。

3 つのすべての列で 1 つの索引が存在する場合には、最適化プログラムは 2 進基数ツリー索引を使用して索引プローブを実行するよう選択する場合があります。 たいていの場合、戻される行数と、各プランに関連付けられた入出力の予想されるコストによって、この選択が決定されます。 ほとんど行が戻されない場合には、最適化プログラムは 2 進基数ツリー索引を使用して、テーブルに対してランダム入出力を実行するよう選択する場合があります。しかし数行以上選択されると、テーブルに対してより効率的にスケジュールされた入出力による節約のために、最適化プログラムはコード化ベクトル索引を使用することになります。

一時オブジェクトおよびアクセス方式

一時オブジェクトは、照会を処理するために最適化プログラムによって作成されます。通常、こうした一時オブジェクトは内部オブジェクトであり、ユーザーがアクセスすることはできません。

表 8. 一時オブジェクトのデータ・アクセス方式

一時的に作成されるオブジェクト	走査操作	プローブ操作
一時ハッシュ・テーブル	ハッシュ・テーブル走査	ハッシュ・テーブル・プローブ
一時ソート・リスト	ソート・リスト走査	ソート・リスト・プローブ
一時特殊ソート・リスト	ソート・リスト走査	N/A
一時リスト	リスト走査	N/A
一時値リスト	値リスト走査	N/A
一時行番号リスト	行番号リスト走査	行番号リスト・プローブ
一時ビットマップ	ビットマップ走査	ビットマップ・プローブ
一時索引	一時索引走査	一時索引プローブ
一時バッファー	バッファー走査	N/A
待ち行列	N/A	N/A

一時ハッシュ・テーブル

一時ハッシュ・テーブルは、最適化プログラムが列または列セットに基づいて行を照合できるようにする一時オブジェクトです。 最適化プログラムはこのハッシュ・テーブルを走査またはプローブし、照会に対する別個の操作に対応します。

一時ハッシュ・テーブルは、移植後に、行を迅速かつ簡単に取り出せるように編成された効率的なデータ構造です。これは、主にハッシュ・テーブルが主記憶域内に常駐しているためであり、そのために一時オブジェクトに対して、走査またはプローブのいずれかに関連付けられた入出力を避けることができます。 最適化プログラムは、作成時にキーとして使用される、列の固有の組み合わせの数 (たとえば、基数) に基づいて、ハッシュ・テーブルの最適なサイズを判別します。

さらに、ハッシュ・テーブルは、テーブル・プローブ操作によるランダム入出力ではなく、必要なすべての列を取り込んで、その後の処理を行うことができます。 しかし、計算されたサイズがこの照会で使用できるメモリー・プール記憶域を超える場合に、最適化プログラムはハッシュ・テーブルに選択的に列を含める機能を備えています。 そのような場合、テーブル・プローブ操作には、選択された行が処理される前に、欠落している列をハッシュ・テーブルから再収集することが要求されます。

また最適化プログラムには、ハッシュ・テーブルに特殊値を取り込む機能も備わっています。照会にグループ化または区別処理が含まれる場合、同じキー値を持つ行すべてが一時オブジェクトに保管される必要はありません。それらは引き続き照合されますが、ハッシュ・テーブル自体が取り込まれる際に区別処理が実行されます。 これにより、グループ化または区別操作を完了するために、結果に対して簡単な走査を実行できます。

一時ハッシュ・テーブルは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

ハッシュ・テーブル走査:

ハッシュ・テーブル走査操作の際、一時ハッシュ・テーブル全体が走査され、ハッシュ・テーブルに含まれるすべての項目が処理されます。

データ値が相互に照合されることが必要ですが、データの順序付けは不必要な場合に、最適化プログラムはハッシュ・テーブル走査を考慮します。 ハッシュ・テーブル走査の使用により、最適化プログラムは一時ハッシュ・テーブルの作成時に結合なしの選択の利点を生かすプランを生成できます。ハッシュ・テーブル走査を使用する別の利点は、一時ハッシュ・テーブルのデータ構造によって、ハッシュ・テーブル内のテーブル・データが作成後も通常は主記憶域に常駐し、以降のハッシュ・テーブル走査操作でのページングを減らすことができることです。

表 9. ハッシュ・テーブル走査属性

データ・アクセス方式	**ハッシュ・テーブル走査**
説明	一時ハッシュ・テーブル内のすべての項目を読み取ります。ハッシュ・テーブルは区別処理を実行して、重複しているものを除去したり、一時ハッシュ・テーブルの利点を生かして同じ値を持つすべての行を相互に照合したりします。
利点	• 通常ならデータを照合するために索引を使用するような、一般に長時間実行が必要な照会に関連付けられた、テーブルへのランダム入出力を減らします。 • ハッシュ・テーブルを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	通常、区別処理またはグループ処理に用いられます。処理時にハッシュ・テーブル全体が記憶域にない場合には、ほとんど実行できません。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 区別またはグループ化のために、列 (または複数列) に基づいてデータが照合される必要のある場合

SQL ステートメントの例	`SELECT COUNT(*), FirstNme FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`GROUP BY FirstNme`

表 9. ハッシュ・テーブル走査属性 (続き)

データ・アクセス方式	ハッシュ・テーブル走査

使用を示すメッセージ	メッセージによってハッシュ走査を示す方法が幾つかあります。この例でのメッセージは、ハッシュ走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4329 -- ファイル EMPLOYEE に到着順アクセスが使用された。`
	• PRTSQLINF:
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4029 -- グループ化の処理にハッシュ・アルゴリズムが使用された。`

SMP 並列の使用可能化	はい
別名	ハッシュ走査、プリロード ハッシュ・テーブル走査区別 ハッシュ・テーブル走査区別、プリロード
Visual Explain アイコン	

ハッシュ・テーブル・プローブ:

ハッシュ・テーブル・プローブ操作は、プローブ探索操作に基づいて一時ハッシュ・テーブルから行を取り出すのに使用します。

最初に、最適化プログラムは照会で指定された結合基準によって、一時ハッシュ・テーブルのキーを識別します。これにより、ハッシュ・テーブル・プローブが実行されると、一時ハッシュ・テーブルへのプローブに使用される値は、選択で指定された結合元基準から抽出されます。 こうした値は、同じハッシュ・アルゴリズムで送信され、任意の行にマッチングする (等しい) 値があるかどうかを判別するため、一時ハッシュ・テーブルに取り込まれます。その後マッチングするすべての結合行は戻されて、照会でさらに処理されます。

表 10. ハッシュ・テーブル・プローブ属性

データ・アクセス方式	ハッシュ・テーブル・プローブ
説明	一時ハッシュ・テーブルは、結合基準に基づいて迅速にプローブされます。
利点	• プローブ基準にマッチングする選択済み行に対して非常に迅速にアクセスできるようになります。 • データを照合するために索引を使用する、長く実行する必要のある照会に通常関連付けられた、テーブルへのランダム入出力を減らします。 • ハッシュ・テーブルを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	通常、等号結合基準を処理するのに使用されます。処理時にハッシュ・テーブル全体が記憶域にない場合には、ほとんど実行できません。

表 10. ハッシュ・テーブル・プローブ属性 (続き)

データ・アクセス方式	ハッシュ・テーブル・プローブ
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 結合処理の列 (または複数列) に基づいて、データが照合される必要のある場合 • 結合基準が、等号 (=) 演算子を使用して指定された場合

SQL ステートメントの例	`SELET * FROM Employee XXX, Department YYY`
	`WHERE XXX.WorkDept = YYY.DeptNbr`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	メッセージによってハッシュ・プローブを示す方法が幾つかあります。この例でのメッセージは、ハッシュ・プローブが使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4327 -- ファイル EMPLOYEE が結合位置 1 で処理された。`
	`CPI4327 -- ファイル DEPARTMENT が結合位置 2 で処理された。`
	• PRTSQLINF:
	`SQL4007 -- テーブル 1 の結合位置 1 に対する QUERYの実行。`
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4007 -- テーブル 2 の結合位置 2 に対する QUERYの実行。`
	`SQL4010 -- テーブル 2 に対するテーブル・スキャン。`

SMP 並列の使用可能化	はい
別名	ハッシュ・テーブル・プローブ、プリロード ハッシュ・テーブル・プローブ区別 ハッシュ・テーブル・プローブ区別、プリロード
Visual Explain アイコン	

通常ハッシュ・テーブル・プローブ・アクセス方式は、結合の 2 次テーブルの実施を決定する際に考慮されます。 ハッシュ・テーブルは、基礎となるテーブルの等号選択または結合基準とマッチングするキー列によって作成されます。ハッシュ・テーブル・プローブを使用すると、最適化プログラムは最も効率的な実施を選択して、結合基準にかかわりなく、基礎となるテーブルから行を選択することができます。基礎となるテーブルを 1 度通過する際に、テーブル走査の実行または既存の索引の使用を選択し、ハッシュ・テーブルの取り込みに必要な行を選択できます。

ハッシュ・テーブルは作成されると、その大部分は主記憶域に常駐するので、ハッシュ・プローブに関連付けられる入出力を最小限に抑えることができます。加えて、ハッシュ・テーブルに基礎となるテーブルから必要な任意の列が取り込まれると、このテーブルの処理を終了するのにさらにテーブル・プローブを行う必要はないので、これも入出力の節約になります。

関連概念

50 ページの『ネストされたループ結合の実施』
DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

一時ソート・リスト

一時ソート・リストは、最適化プログラムが列または列セットに基づいて行の順序付けをできるようにする一時オブジェクトです。 最適化プログラムはこのソート・リストを走査またはプローブし、照会に対する別個の操作に対応します。

一時ソート・リストは、移植後に、行を迅速かつ簡単に取り出せるように編成されたデータ構造です。移植の際、行はこの一時オブジェクトにコピーされてから、ソートを実行するためにこの一時オブジェクトへの 2 回目の通過が行われます。 この一時オブジェクトの作成を最適化するため、ソートの処理時に実行されるデータ移動は最小に抑えられます。一般に、一時ソート・リストをプローブすることは、一時ハッシュ・テーブルをプローブするほどは効率的ではありません。

さらに、ソート・リストは、テーブル・プローブ操作によるランダム入出力ではなく、必要なすべての列を取り込んで、その後の処理を行うことができます。 しかし、計算されたサイズがこの照会で使用できるメモリー・プール記憶域を超える場合に、最適化プログラムはソート・リストに選択的に列を含める機能を備えています。 そのような場合、テーブル・プローブ操作には、選択済み行が処理される前に、欠落している列をソート・リストから再収集することが要求されます。

一時ソート・リストは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

ソート・リスト走査:

ソート・リスト走査操作の際、一時ソート・リスト全体が走査され、ソート・リストに含まれるすべての項目が処理されます。

一般にソート・リスト走査は、最適化プログラムがデータ値を順序付けすることが必要なプランを考慮する場合に検討されます。ソート・リスト走査の使用により、最適化プログラムは一時ソート・リストの作成時に結合なしの選択の利点を生かすプランを生成できます。ソート・リスト走査を使用する別の利点は、一時ソート・リストのデータ構造によって、ソート・リスト内のテーブル・データが作成後も通常は主記憶域に常駐し、以降のソート・リスト走査操作でのページングを減らすことができることです。

表 11. ソート・リスト走査属性

データ・アクセス方式	**ソート・リスト走査**
説明	一時ソート・リスト内のすべての項目を読み取ります。ソート・リストは区別処理を実行して重複値を除去するか、一時ソート・リストの利点を生かしてすべての行を順序付けします。

表 11. ソート・リスト走査属性 (続き)

データ・アクセス方式	ソート・リスト走査
利点	• データを順序付けるために索引を使用する、長く実行する必要のある照会に通常関連付けられた、テーブルへのランダム入出力を減らします。 • ソート・リストを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	通常、順序付けを処理するため、または区別処理のために使用されます。移植および処理時にソート・リスト全体が記憶域にない場合には、ほとんど実行できません。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 順序付けまたは区別処理の列 (または複数列) に基づいて、データが順序付けされる必要のある場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (LastName, WorkDept)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`ORDER BY FirstNme`
	`OPTIMZE FOR ALL ROWS`

使用を示すメッセージ	メッセージによってソート・リスト走査を示す方法が幾通りかあります。この例でのメッセージは、ソート・リスト走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	`CPI4325 -- 照会プログラム用に一時結果ファイルが作成された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`
	`SQL4002 -- 再使用可能な ODP ソートが使用された。`

SMP 並列の使用可能化	いいえ
別名	ソート・リスト走査、プリロード ソート・リスト走査区別 ソート・リスト走査区別、プリロード
Visual Explain アイコン	

ソート・リスト・プローブ:

ソート・リスト・プローブ操作は、プローブ探索操作に基づいて一時ソート・リストから行を取り出すのに使用します。

最初に、最適化プログラムは照会で指定された結合基準によって、一時ソート・リストのキーを識別します。これにより、ソート・リスト・プローブが実行されると、一時ソート・リストへのプローブに使用される値は、選択で指定された結合元基準から抽出されます。 こうした値は、任意の行にマッチングする値があるかどうかを判別するために、ソート・リスト内に配置されます。 その後マッチングするすべての結合行は戻されて、照会でさらに処理されます。

表 12. ソート・リスト・プローブ属性

データ・アクセス方式	**ソート・リスト・プローブ**
説明	一時ソート・リストは、結合基準に基づいて迅速にプローブされます。
利点	• プローブ基準にマッチングする選択済み行に対して非常に迅速にアクセスできるようになります。 • 通常、長く実行する照会に関連付けられた表 (それ以外の場合はデータの照合に索引を使用) に対する、ランダム入出力を減らします。 • ソート・リストを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	通常、非等号結合基準を処理するのに使用されます。移植および処理時にソート・リスト全体が記憶域にない場合には、ほとんど実行できません。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 結合処理の列 (または複数列) に基づいて、データが照合される必要のある場合 • 結合基準が、不等号演算子を使用して指定された場合

SQL ステートメントの例	`SELECT * FROM Employee XXX, Department YYY`
	`WHERE XXX.WorkDept > YYY.DeptNbr`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	メッセージによってソート・リスト・プローブを示す方法が幾通りかあります。この例でのメッセージは、ソート・リスト・プローブ走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4327 -- ファイル EMPLOYEE が結合位置 1 で処理された。`
	`CPI4327 -- ファイル DEPARTMENT が結合位置 2 で処理された。`
	• PRTSQLINF:
	`SQL4007 -- テーブル 1 の結合位置 1 に対する QUERYの実行。`
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4007 -- テーブル 2 の結合位置 2 に対する QUERYの実行。`
	`SQL4010 -- テーブル 2 に対するテーブル・スキャン。`

SMP 並列の使用可能化	はい
別名	ソート・リスト・プローブ、プリロード ソート・リスト・プローブ区別 ソート・リスト・プローブ区別、プリロード

表 12. ソート・リスト・プローブ属性 (続き)

データ・アクセス方式	ソート・リスト・プローブ
Visual Explain アイコン	

通常ソート・リスト・プローブ・アクセス方式は、結合の 2 次テーブルの実施を決定する際に考慮されます。 ソート・リストは、基礎となるテーブルの不等号結合基準とマッチングするキー列によって作成されます。ソート・リスト・プローブを使用すると、最適化プログラムは最も効率的な実施を選択して、結合基準にかかわりなく、基礎となるテーブルから行を選択することができます。基礎となるテーブルに 1 度通過する際に、テーブル走査の実行または既存の索引の使用を選択し、ソート・リストの取り込みに必要な行を選択できます。

ソート・リストは作成されると、その一時オブジェクトの大部分は主記憶域に常駐するので、ソート・リストに関連する入出力を最小限に抑えることができます。加えて、テーブルから必要な任意の列がソート・リストに取り込まれると、このテーブルの処理を終了するのにさらにテーブル・プローブを行う必要はないので、これも入出力の節約になります。

関連概念

50 ページの『ネストされたループ結合の実施』

DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

一時特殊ソート・リスト

一時特殊ソート・リストは、一時ハッシュ・テーブルと一時ソート・リストのフィーチャーを結合します。

ハッシュ・テーブルと同様に、この一時特殊ソート・リストを使用することによって、最適化プログラムは、列または列セットに基づいて行を照合できます。ソート・リストと同様に、この一時特殊ソート・リストを使用すると、最適化プログラムは行を順序付けすることもできます。

一時特殊ソート・リストには、取り込み中に集約行への効率的なアクセスのためにセットアップされたハッシュ・テーブル・データ構造が含まれています。さらに、ハッシュ・テーブル・データ構造を通して 2 進ツリー・データ構造が維持され、データが順次アクセスできます。データ構造のソートされた側面によって、GROUP BY ROLLUP を含む SQL ステートメントの超集約行を効率的に計算することができます。

一時ソート集約ハッシュ・テーブルは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

ソート・リスト走査:

ソート・リスト走査の際、一時特殊ソート・リスト全体が走査され、一時に含まれるすべての項目が処理されます。

最適化プログラムは、データ値を集約し、順序付ける必要がある際にソート・リスト走査を使用します。ソート・リスト走査の使用により、最適化プログラムは一時特殊ソート・リストの作成時に結合なしの選択の利点を生かすプランを生成できます。ソート・リスト走査を使用する別の利点は、一時特殊ソート・リストのデータ構造によって、一時内のテーブル・データが作成後も通常はメイン・メモリーに常駐するようになり、以降のソート・リスト走査でのページングを減らすことができることです。

表 13. ソート・リスト走査属性

データ・アクセス方式	**ソート・リスト走査**
説明	一時特殊ソート・リスト内のすべての項目を読み取ります。
利点	• ROLLUP 超集約行を効率的に計算できます。 • 通常ならデータを照合するために索引を使用するような、一般に長時間実行が必要な照会に関連付けられた、テーブルへのランダム入出力を減らします。 • 特殊ソート・リストを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	一般に、GROUP BY ROLLUP 処理用に使用されます。処理時に一時オブジェクト全体がメモリーにない場合には、ほとんど実行できません。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • GROUP BY ROLLUP が SQL ステートメントで指定されている場合
使用を示すメッセージ	N/A
SMP 並列の使用可能化	はい
別名	N/A
Visual Explain アイコン	

一時リスト

一時リストは、最適化プログラムが照会の中間結果を保管できるようにする一時オブジェクトです。このリストは、照会の操作を簡単にするために使用されるソートされていないデータ構造です。このリストにはキーがないので、リスト内の行は順次走査操作によってのみ検索が可能です。

さまざまな理由で一時リストを使用できますが、それには、ビューまたは派生テーブル、対称型マルチプロセッシング (SMP) があまりにも複雑である場合、または照会の一部分が何度も処理されるのを防ぐ場合などが含まれます。

一時リストは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

リスト走査:

リスト走査操作は、照会の一部分が何度も処理されますが、識別できるキー列がない場合に使用します。こうした場合、照会の一部分は一度だけ処理され、その結果は一時リストに保管されます。その後このリストでは、この一時オブジェクトに含まれる選択または処理を満たす行に対してのみ走査を実行できます。

表 14. リスト走査属性

データ・アクセス方式	**リスト走査**
説明	この一時リスト内のすべての行を順次走査して処理します。
利点	• 一時リストおよびリスト走査は、操作の反復を最小限に抑えたり、最適化プログラムの論理フローを単純化したりするために最適化プログラムが使用できます。 • リストを生成する前に選択を実行して、一時オブジェクトの行番号をサブセット化できます。
考慮事項	通常、要求を満たすのにキー列が必要ではない場合に、照会の一部分が何度も処理されないようにするために使用されます。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 対称型マルチプロセッシングが照会に使用される場合

SQL ステートメントの例	`SELECT * FROM Employee XXX, Department YYY`
	`WHERE XXX.LastName IN ('Smith', 'Jones', 'Peterson')`
	`AND YYY.DeptNo BETWEEN 'A01' AND 'E01'`
	`OPTIMIZE FOR ALL ROWS`

表 14. リスト走査属性 (続き)

データ・アクセス方式	リスト走査

使用を示すメッセージ	メッセージによってリスト走査を示す方法が幾通りかあります。この例でのメッセージは、リスト走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4325 -- 照会プログラム用に一時結果ファイルが作成された。`
	`CPI4327 -- ファイル EMPLOYEE が結合位置 1 で処理された。`
	`CPI4327 -- ファイル DEPARTMENT が結合位置 2 で処理された。`
	• PRTSQLINF:
	`SQL4007 -- テーブル 1 の結合位置 1 に対する QUERYの実行。`
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4007 -- テーブル 2 の結合位置 2 に対する QUERYの実行。`
	`SQL4001 -- 一時結果が作成されます。`
	`SQL4010 -- テーブル 2 に対するテーブル・スキャン。`

SMP 並列の使用可能化	はい
別名	リスト走査、プリロード
Visual Explain アイコン	

最適化プログラムは、前述の例を使用して DEPARTMENT テーブルから選択された行を保管するために、一時リストを作成することを選択します。結合基準がないので、 2 つのテーブル間でカルテシアン積結合が実行されます。 この結合において、結合の可能性をそれぞれ確かめるために DEPARTMENT テーブルのすべての行を走査しないようにするには、 DEPARTMENT テーブルに対する選択を 1 度実行し、その結果を一時リストに保管します。 その後一時リストは、カルテシアン積結合を走査します。

一時値リスト

一時値リストは、ユーザーが SELECT ステートメントまたは CREATE VIEW ステートメントの VALUES 文節で指定したデータの行を、最適化プログラムが保管できるようにする一時オブジェクトです。

このリストは、照会の操作を簡単にするために使用されるソートされていないデータ構造です。このリストにはキーがないので、リスト内の行は順次走査操作によってのみ検索が可能です。

一時値リストは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

値リスト走査:

値リスト走査操作の際、一時値リスト全体が走査され、データのすべての行が処理されます。

表 15. 値リスト走査属性

データ・アクセス方式	**値リスト走査**
説明	この一時値リスト内のデータのすべての行を順次走査して処理します。
利点	一時値リストおよび値リスト走査は、最適化プログラムの論理フローを単純化するために最適化プログラムが使用できます。
使用される可能性が高い場合	VALUES 文節が SQL 全選択の FROM 文節で指定されている場合

SQL ステートメントの例	`SELECT EMPNO, 'empproject'`
	`FROM EMPPROJACT`
	`WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')`
	`UNION。VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')`

使用を示すメッセージ	メッセージによって値リスト走査を示す方法が複数あります。この例でのメッセージは、値リスト走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4329 -- ファイル *VALUES に到着順アクセスが使用された。`
	• PRTSQLINF:
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`

SMP 並列の使用可能化	はい
別名	値リスト、プリロード
Visual Explain アイコン	

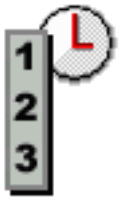
一時行番号リスト

一時行番号リストは、最適化プログラムが行アドレス (行番号) に基づいて行の順序付けをできるようにする一時オブジェクトです。 最適化プログラムはこの行番号リストを走査またはプローブし、照会に対する別個の操作に対応します。

一時行番号リストは、行を迅速および効率的に取り出せるように編成されたデータ構造です。この一時行番号リストには、行に関連付けられた行番号のみが含まれます。一時行番号リストにはテーブル・データはないため、テーブル・プローブ操作は通常この一時番号リストに関連付けられて、基礎となるテーブル・データを取り出します。 行番号がソートされるので、テーブル・プローブ操作に関連するランダム入出力を効率的に実行できます。データベース・マネージャーはプリフェッチを実行するか、または論理を予測して、複数行が隣接ページにあるかどうかを判別します。隣接ページにある場合には、そうした行を主記憶域により効率的に移動するには、テーブル・プローブは大規模な入出力を必要とします。

一時行番号リストは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:



行番号リスト走査:

行番号リスト走査操作の際、一時行番号リスト全体が走査され、その行番号リスト内に含まれるすべての行アドレスが処理されます。最適化プログラムが、コード化ベクトル索引が含まれるプランを考慮する際か、前処理またはテーブル・プローブ操作に関連付けられた行番号のソートによって、索引プローブ操作または走査操作に関連するランダム入出力のコストが削減できるかどうかを考慮する際に、通常行番号リスト走査が検討されます。

行番号リスト走査を使用すると、最適化プログラムは複数の索引の利点を生かせるプランを生成して照会の他の部分とマッチングさせることができます。

行番号リスト走査を使用する別の利点は、一時行番号リストのデータ構造により必ず行番号がソートされるということです。ソートされた行番号は、テーブルでのページングが同じページのデータに 2 度と戻ることはない、テーブル・データの行番号レイアウトをかなり正確に反映しています。これにより、照会に対する増大する入出力を節約できます。

行番号リスト走査は、ビットマップ走査操作と同じです。 2 つの操作の唯一の違いは、行番号リスト走査は行アドレスのリストに対して実行されるのに対し、ビットマップ走査は行アドレスを表すビットマップに対して実行されます。

表 16. 行番号リスト走査

データ・アクセス方式	**行番号リスト走査**
説明	一時行番号リスト内のすべての行番号を順次走査して処理します。 ソートされた行番号は他の一時行番号リストにマージするか、テーブル・プローブ操作への入力として使用できます。
利点	• 一時行番号リストには、アドレスだけが含まれておりデータは含まれていません。ですから、この一時行番号リストは記憶域で効率的に走査できます。 • 一時オブジェクトに含まれる行番号はソートされて、基礎となるテーブルにアクセスするための効率的な入出力処理が提供されます。 • 行番号リストが生成されるときに選択が実行されて、一時オブジェクトの行番号をサブセット化できます。
考慮事項	行番号リストにはテーブル内の選択された行のアドレスだけが含まれるので、テーブル行をフェッチするには、別のテーブル・プローブ操作を実行する必要があります。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 行番号をソートするコストが、テーブル・プローブ操作の際に実行できる入出力を効率的にすることによって調整される場合 • 同一テーブルでの複数索引を、選択行の数を最小化するために結合する必要がある場合

表 16. 行番号リスト走査 (続き)

データ・アクセス方式	行番号リスト走査

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (WorkDept)`
	`CREATE ENCODED VECTOR INDEX EVI2 ON`
	`Employee (Salary)`
	`CREATE ENCODED VECTOR INDEX EVI3 ON`
	`Employee (Job)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept = 'E01' AND Job = 'CLERK'`
	`AND Salary = 5000`
	`OPTIMIZE FOR 99999 ROWS`

使用を示すメッセージ	メッセージによって行番号リスト走査を示す方法が幾通りかあります。この例でのメッセージは、行番号リスト走査が使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4329 -- ファイル EMPLOYEE に到着順アクセスが使用された。`
	`CPI4338 -- 3 アクセス・パスがファイル EMPLOYEE のビットマップ処理に使用された。`
	• PRTSQLINF:
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 X1 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI2 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI3 が使用された。`

SMP 並列の使用可能化	はい
別名	行番号リスト走査、プリロード
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムは、この照会が使用する各索引用の一時行番号リストを作成しました。この照会は、基数索引と 2 つのコード化ベクトル索引とを組み合わせて使用し、行番号リストを作成します。各索引用の一時行番号リストは走査され、一時行番号リストすべてによって示される行の論理積を表す最終複合行番号リストにマージされます。その後テーブル・プローブ操作がこの最終行番号リストを使用して、選択された行、および照会結果を処理する必要のある行を判別します。

行番号リスト・プローブ:

行番号リスト・プローブ操作は、一時行番号リストの選択行に対して、別の操作によって生成された行番号をテストするのに使用されます。この行番号は、テーブルの行番号を作成する任意の操作によって生成できます。その後行番号は、一時行番号リストにプローブするのに使用され、一時行番号リストを生成するのに使用される選択と、この行番号がマッチングするかどうかを判別します。

行番号リスト・プローブ操作を使用すると、索引が提供する任意の順序付けの利点を生かせるプランを最適化プログラムは生成できますが、テーブル・プローブ操作を行う前に別の選択を実行するには、引き続き行番号リストを使用します。

行番号リスト・プローブは、ビットマップ・プローブ操作と同じです。 2 つの操作の唯一の違いは、行番号リスト・プローブは行アドレスのリストに対して実行されるのに対し、ビットマップ・プローブは行アドレスを表すビットマップに対して実行されます。

表 17. 行番号リスト・プローブ

データ・アクセス方式	**行番号リスト・プローブ**
説明	一時行番号リストは、別の操作によって生成される行番号に基づいて、迅速にプローブされます。
利点	• 一時行番号リストには、アドレスだけが含まれておりデータは含まれていません。ですから、この一時番号リストは記憶域で効率的にプローブできます。 • この行番号リストで示されている行番号はソートされ、効率的な探索処理を行い基礎となるテーブルをテストします。 • 行番号リストが生成されるときに選択が実行されて、一時オブジェクトの選択行の番号をサブセット化できます。
考慮事項	行番号リストにはテーブル内の選択された行のアドレスだけが含まれるので、テーブル行をフェッチするには、別のテーブル・プローブ操作を実行する必要があります。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 実行されるテーブル・プローブ操作数を減らすことによって、行番号リストの作成およびプローブのコストが調整される場合 • 同一テーブルでの複数索引を、選択行の数を最小化するために結合する必要がある場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (WorkDept)`
	`CREATE ENCODED VECTOR INDEX EVI2 ON`
	`  Employee (Salary)`
	`CREATE ENCODED VECTOR INDEX EVI3 ON`
	`  Employee (Job)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept = 'E01' AND Job = 'CLERK'`
	`AND Salary = 5000`
	`ORDER BY WorkDept`

使用を示すメッセージ	メッセージによって行番号リスト・プローブを示す方法が幾通りかあります。この例でのメッセージは、行番号リスト・プローブが使用されたことを SQL Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	`CPI4338 -- 2 アクセス・パスがファイル EMPLOYEE のビットマップ処理に使用された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`
	`SQL4011 -- テーブル 1 に索引スキャン・キー行位置が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI2 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI3 が使用された。`

SMP 並列の使用可能化	はい
別名	行番号リスト・プローブ、プリロード

表 17. 行番号リスト・プローブ (続き)

データ・アクセス方式	行番号リスト・プローブ
Visual Explain アイコン	

最適化プログラムは、前述の例を使用して各コード化ベクトル索引用の一時行番号リストを作成しました。加えて、基数索引 X1 に対して索引プローブ操作が実行され、順序付け要件を満たしました。 ORDER BY 文節は、結果行が WorkDept 列によって順序付けされるよう要求するので、選択行を処理するのに一時行番号リストを走査することはできなくなりました。 しかし、一時行番号リストは、順序付けを満たすのに使用される、索引 X1 から抽出される行アドレスを用いてプローブできます。索引プローブ操作から抽出される行アドレスを使用して一時行番号リストをプローブすると、索引 X1 のキーの順序付けが保持され、さらに行番号リスト内の選択済み行に対してその行をテストできます。

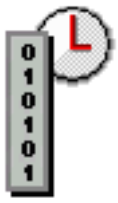
一時ビットマップ

一時ビットマップは、最適化プログラムが行アドレス (行番号) に基づいて行の順序付けをできるようにする一時オブジェクトです。 最適化プログラムはこのビットマップを走査またはプローブし、照会に対する別個の操作に対応します。

一時ビットマップは、テーブルのすべての行番号を表すのにビットマップを使用するデータ構造です。各行は別個のビットで表されるので、テーブル内のすべての行をかなり圧縮した形で示すことができます。この一時ビットマップによって行が選択されると、選択された行に対応するビットマップ内のビットがオンに設定されます。 一時ビットマップを取り込んだ後は、迅速かつ効率的に検索するため、すべての選択行をソートして取り出すことができます。この一時ビットマップは、選択された行に関連付けられた行番号のみを表します。一時ビットマップにはテーブル・データはないため、テーブル・プローブ操作は通常この一時ビットマップに関連付けられて、基礎となるテーブル・データを取り出します。 このビットマップは定義によってソートされるため、テーブル・プローブ操作に関連するランダム入出力をより効率的に実行できます。データベース・マネージャーはプリフェッチを実行するか、または論理を予測して、複数行が隣接ページにあるかどうかを判別します。隣接ページにある場合には、そうした行を主記憶域により効率的に移動するには、テーブル・プローブは大規模な入出力を必要とします。

一時ビットマップは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:



ビットマップ走査:

ビットマップ走査操作の際、一時ビットマップ全体が走査され、ビットマップに含まれるすべての行アドレスが処理されます。最適化プログラムが、コード化ベクトル索引が含まれるプランを考慮する際か、前処理またはテーブル・プローブ操作に関連付けられた行番号のソートによって、索引プローブ操作または走査操作に関連付けられたランダム入出力のコストが削減できるかどうかを考慮する際に、通常ビットマップ走査が検討されます。

ビットマップ走査を使用すると、最適化プログラムは複数の索引の利点を生かせるプランを生成して照会の他の部分とマッチングさせることができます。

ビットマップ走査を使用する別の利点は、一時ビットマップのデータ構造により必ず行番号がソートということです。ソートされた行番号は、テーブルでのページングが同じページのデータに 2 度と戻ることはない、テーブル・データの行番号レイアウトをかなり正確に反映しています。これにより、照会に対する増大する入出力を節約できます。

ビットマップ走査は、行番号リスト走査操作と同じです。 2 つの操作の唯一の違いは、行番号リスト走査は行アドレスのリストに対して実行されるのに対し、ビットマップ走査は行アドレスを表すビットマップに対して実行されます。

表 18. ビットマップ走査属性

データ・アクセス方式	**ビットマップ走査属性**
説明	一時ビットマップ内のすべての行番号を順次走査して処理します。 ソートされた行番号は他の一時ビットマップにマージするか、テーブル・プローブ操作への入力として使用できます。
利点	• 一時ビットマップには行のアドレスに対する参照だけが含まれ、データに対する参照は含まれていません。ですから、この一時番号リストは記憶域で効率的に走査できます。 • 一時オブジェクトで示される行番号はソートされて、基礎となるテーブルにアクセスするための効率的な入出力処理が提供されます。 • ビットマップが生成されるときに選択が実行されて、一時オブジェクトの選択行の番号をサブセット化できます。
考慮事項	ビットマップにはテーブル内の選択された行のアドレスだけが含まれるので、テーブル行をフェッチするには、別のテーブル・プローブ操作を実行する必要があります。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 行番号をソートするコストが、テーブル・プローブ操作の際に実行できる入出力を効率的にすることによって調整される場合 • 同一テーブルでの複数索引を、選択行の数を最小化するために結合する必要がある場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (WorkDept)`
	`CREATE ENCODED VECTOR INDEX EVI2 ON`
	`  Employee (Salary)`
	`CREATE ENCODED VECTOR INDEX EVI3 ON`
	`  Employee (Job)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept = 'E01' AND Job = 'CLERK'`
	`AND Salary = 5000`
	`OPTIMIZE FOR 99999 ROWS`

表 18. ビットマップ走査属性 (続き)

データ・アクセス方式	ビットマップ走査属性

使用を示すメッセージ	メッセージによってビットマップ走査を示す方法が幾通りかあります。この例でのメッセージは、ビットマップ走査が使用されたことを Classic Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4329 -- ファイル EMPLOYEE に到着順アクセスが使用された。`
	`CPI4338 -- 3 アクセス・パスがファイル EMPLOYEE のビットマップ処理に使用された。`
	• PRTSQLINF:
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 X1 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI2 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI3 が使用された。`

SMP 並列の使用可能化	はい
別名	ビットマップ走査、プリロード 行番号ビットマップ走査 行番号ビットマップ走査、プリロード スキップ順次走査
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムは、この照会が使用する各索引用の一時ビットマップを作成しました。この照会は、基数索引と 2 つのコード化ベクトル索引とを組み合わせて使用し、行番号リストを作成します。各索引用の一時ビットマップは走査され、一時ビットマップすべてによって示される行の論理積を表す最終複合ビットマップにマージされます。 その後テーブル・プローブ操作がこの最終ビットマップを使用して、選択された行、および照会結果を処理する必要のある行を判別します。

ビットマップ・プローブ:

ビットマップ・プローブ操作は、一時ビットマップの選択行に対して、別の操作によって生成された行番号をテストするのに使用されます。この行番号は、テーブルの行番号を作成する任意の操作によって生成できます。その後行番号は、一時ビットマップにプローブされるのに使用され、一時ビットマップを生成するのに使用される選択と、この行番号がマッチングするかどうかを判別します。

ビットマップ・プローブ操作を使用すると、索引が提供する任意の順序付けの利点を生かせるプランを最適化プログラムは生成できますが、テーブル・プローブ操作を行う前に別の選択を実行するには、このビットマップが使用されます。

ビットマップ・プローブは、行番号リスト・プローブ操作と同じです。 2 つの操作の唯一の違いは、行番号リスト・プローブは行アドレスのリストに対して実行されるのに対し、ビットマップ・プローブは行アドレスを表すビットマップに対して実行されます。

表 19. ビットマップ・プローブ属性

データ・アクセス方式	ビットマップ・プローブ属性
説明	一時ビットマップは、別の操作によって生成される行番号に基づいて、迅速にプローブされます。
利点	• 一時ビットマップには行のアドレスに対する参照だけが含まれ、データは含まれていません。そのため、この一時番号リストは記憶域で効率的にプローブできます。 • このビットマップで示されている行番号はソートされ、効率的な探索処理を行い基礎となるテーブルをテストします。 • ビットマップが生成されるときに選択が実行されて、一時オブジェクトの選択行の番号をサブセット化できます。
考慮事項	ビットマップにはテーブル内の選択された行のアドレスだけが含まれるので、テーブル行をフェッチするには、別のテーブル・プローブ操作を実行する必要があります。
使用される可能性が高い場合	• 一時結果の使用が、照会環境パラメーター (ALWCPYDTA) で許可されている場合 • 実行されるテーブル・プローブ操作数を減らすことによって、ビットマップの作成およびプローブのコストが調整される場合 • 同一テーブルでの複数索引を、選択行の数を最小化するために結合する必要がある場合

SQL ステートメントの例	`CREATE INDEX X1 ON Employee (WorkDept)`
	`CREATE ENCODED VECTOR INDEX EVI2 ON`
	`Employee (Salary)`
	`CREATE ENCODED VECTOR INDEX EVI3 ON`
	`Employee (Job)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept = 'E01' AND Job = 'CLERK'`
	`AND Salary = 5000`
	`ORDER BY WorkDept`

使用を示すメッセージ	メッセージによってビットマップ・プローブを示す方法が幾通りかあります。この例でのメッセージは、ビットマップ・プローブが使用されたことを Classic Query Engine が表す方法を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	`CPI4338 -- 2 アクセス・パスがファイル EMPLOYEE のビットマップ処理に使用された。`
	• PRTSQLINF:
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`
	`SQL4011 -- テーブル 1 に索引スキャン・キー行位置が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI2 が使用された。`
	`SQL4032 -- テーブル 1 のビットマップ処理に索引 EVI3 が使用された。`

SMP 並列の使用可能化	はい
別名	ビットマップ・プローブ、プリロード 行番号ビットマップ・プローブ 行番号ビットマップ・プローブ、プリロード

表 19. ビットマップ・プローブ属性 (続き)

データ・アクセス方式	ビットマップ・プローブ属性
Visual Explain アイコン	

最適化プログラムは、前述の例を使用して各コード化ベクトル索引用の一時ビットマップを作成しました。加えて、基数索引 X1 に対して索引プローブ操作が実行され、順序付け要件を満たしました。 ORDER BY 文節は、結果行が WorkDept 列によって順序付けされるよう要求するので、選択行を処理するのに一時ビットマップを走査することはできなくなりました。しかし一時ビットマップは、順序付けを満たすのに使用される、索引 X1 から抽出される行アドレスを用いてプローブできます。索引プローブ操作から抽出される行アドレスを使用して一時ビットマップをプローブすると、索引 X1のキーの順序付けを保持し、さらにこのビットマップ内の選択済み行に対してその行をテストできます。

一時索引

一時索引は、最適化プログラムが特定の照会に対して基数索引を作成して使用できるようにする一時オブジェクトです。 一時索引は、ユーザーが CREATE INDEX SQL ステートメントまたは 論理ファイル作成 (CRTLF) CL コマンドを使用して作成する基数索引と同じ属性および利点すべてを有しています。

さらに、一時索引は、特定の照会要求を満たすために最適化プログラムが使用するように最適化されます。これには、一時索引の作成後にその使用速度を上げるための、論理ページ・サイズの設定、および作成に対する任意の選択の適用が含まれます。

一時索引は様々な照会要求を満たすのに使用できます。

- 順序付け
- グループ化/区別
- 結合
- レコード選択

一般に、一時索引は他の一時オブジェクトよりも作成するのによりコストがかかります。索引で使用される行をフェッチするためにテーブル操作を実行するか、行を作成するために 1 つまたは複数の索引に対して索引走査またはプローブを実行すると、一時索引に取り込まれます。 最適化プログラムは、索引作成で行を作成するのに使用する方式を判別する際、すべての方式について考慮します。この処理は、最適化プログラムが使用する他の一時オブジェクトのコスト計算および選択に類似しています。

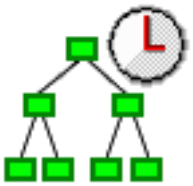
一時索引を一時オブジェクトの他のフォームと比べた場合の大きな利点は、基礎となるテーブルに変更があった場合、一時オブジェクトの中で一時索引のみが維持される点です。テーブルに対して挿入または更新が実行されるという点においては、一時索引は基数索引と全く同じです。こうした変更は、通常の索引保守処理によって一時索引に即時に反映されます。

一時索引の SQE 使用方法は、SQE では再利用が許可されているという点で、CQE の使用方法と異なります。 SQE 最適化プログラムにより作成され、使用された一時索引への参照は、システムのプラン・キャッシュの中に保持されています。一時索引は同じ照会の他のインスタンスや、別のジョブで実行している同じ照会の他のインスタンスにより再使用される場合に備え、保管されます。また、同じ一時索引を活用可能な別の照会によって再利用されるという潜在的な可能性のためにも保管されています。デフォルトで、SQE

一時索引は、最後に照会プランを参照したプラン・キャッシュの項目が除去されるまで存続します。この動作は、CACHE_RESULTS QAQQINI 値を設定することで制御できます。この INI 値のデフォルト設定では、最適化プログラムが再利用目的で一時索引を保持できるようになっています。 INI の値を「*JOB」に変更することで、一時索引をプラン・キャッシュに保存しないようにすることができます。ハードウェアの終了により索引も失われます。「*JOB」オプションを使用することで、SQE 最適化プログラムは、CQE 最適化プログラムの動作に類似した方法で、一時索引を使用するようになります。つまり、存続自体は短くなりますが、アクティブな照会が一時索引を使用している限り、共有されます。この動作は、再利用のために残っている一時索引の維持が、コスト増大につながる恐れがある場合に有用です。

一時索引は内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:



一時索引走査:

一時索引走査操作は、永続基数索引で実行される索引走査操作と同じです。さらにこの操作は、テーブルから行をキー順に取り出すのに使用できますが、一時索引オブジェクトが最初に作成される必要があります。索引内の行すべてが順次処理されますが、結果の行番号はキー列に基づいて順序付けされます。

順序付けされた行は、照会要求 (順序付けまたはグループ化) の一部を満たすために最適化プログラムによって使用されます。

表 20. 一時索引走査属性

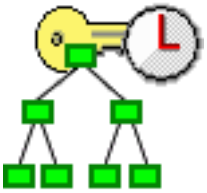
データ・アクセス方式	**一時索引走査**
説明	一時索引に関連付けられたすべてのキーを順次走査して処理します。
利点	• 索引キー値からすべてのデータを抽出できるので、テーブル・プローブの必要性がありません。 • 索引のキーに基づいて、行を順序通りに戻します。
考慮事項	照会を満たすのに必要な残りの列を抽出するため、通常テーブル・プローブを実行することが必要となります。テーブル・プローブにはランダム入出力が関連するので、多くの行数が選択されると十分に実行できなくなる可能性があります。
使用される可能性が高い場合	• 照会に対して行の順序付けが必要な場合 (たとえば、順序付けまたはグループ化の場合) • 選択列が、索引の主要なキー列とマッチングしない場合 • この照会を実施するための他の代替方式に対して、一時索引の作成に関連するオーバーヘッド・コストを調整できる場合

SQL ステートメントの例	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`ORDER BY LastName`
	`OPTIMIZE FOR ALL ROWS`

表 20. 一時索引走査属性 (続き)

データ・アクセス方式	**一時索引走査**

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4321 -- ファイル EMPLOYEE のアクセス・パスが作成された。`
	• PRTSQLINF:
	`SQL4009 -- Index created for table 1.`

SMP 並列の使用可能化	はい
別名	索引走査 索引走査、プリロード 索引走査、区別 索引走査区別、プリロード 索引走査、キー選択
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムは、LastName 列に基づいて行を順序付けするために、一時索引を作成することを選択します。 その後、この照会の ORDER BY 文節を満たすために一時索引走査が実行される場合があります。

最適化プログラムは、WorkDept 列に対する選択が属する最良の位置を判別します。一時索引そのものが作成される場合に実行されるか、一時索引走査の一部として実行できます。 選択を一時索引作成に追加すると、この照会のオープン・データ・パス (ODP) を再使用できなくなる可能性があります。選択が実行される方法を判別する際に、この ODP 再使用が考慮されます。

一時索引プローブ:

一時索引プローブ操作は、永続基数索引で実行される索引プローブ操作と同じです。主な機能は、一時索引の索引キーに対して迅速なアクセスを提供することですが、テーブルから行をキー順に取り出すためにも使用できます。

最適化プログラムは、この一時索引を使用して、照会要求の結合部分を満たします。

表 21. 一時索引プローブ属性

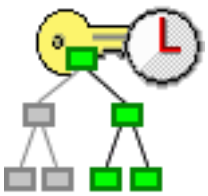
データ・アクセス方式	**一時索引プローブ**
説明	索引は、一連の範囲に書き込まれた選択基準に基づいて、迅速にプローブされます。 選択基準を満たすキーだけが、テーブルの行番号を生成するのに使用されます。

表 21. 一時索引プローブ属性 (続き)

データ・アクセス方式	一時索引プローブ
利点	• 任意の選択にマッチングする索引項目のみが、継続して処理されます。選択行に対して非常に迅速にアクセスできるようになります。 • 索引キー値からすべてのデータを抽出できるので、テーブル・プローブの必要性がありません。 • 索引のキーに基づいて、行を順序通りに戻します。
考慮事項	照会を満たすのに必要な残りの列を抽出するため、通常テーブル・プローブを実行することが必要となります。テーブル・プローブにはランダム入出力が関連するので、多くの行数が選択されると十分に実行できなくなる可能性があります。
使用される可能性が高い場合	• 照会で必要とされる行をプローブする能力 (たとえば、結合) がある場合 • 選択列が、索引の主要なキー列とマッチングしない場合 • この照会を実施するための他の代替方式に対して、一時索引の作成に関連するオーバーヘッド・コストを調整できる場合

SQL ステートメントの例	`SELET * FROM Employee XXX, Department YYY`
	`WHERE XXX.WorkDept = YYY.DeptNo`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	メッセージによって一時索引プローブを示す方法が幾通りかあります。この例でのメッセージは、一時索引プローブが使用されたことを Classic Query Engine が表す 1 つの例を示しています。
	• 最適化プログラム・デバッグ:
	`CPI4321 -- ファイル DEPARTMENT のアクセス・パスが作成された。`
	`CPI4327 -- ファイル EMPLOYEE が結合位置 1 で処理された。`
	`CPI4326 -- ファイル DEPARTMENT が結合位置 2 で処理された。`
	• PRTSQLINF:
	`SQL4007 -- テーブル 1 の結合位置 1 に対する QUERYの実行。`
	`SQL4010 -- テーブル 1 に対するテーブル・スキャン。`
	`SQL4007 -- テーブル 2 の結合位置 2 に対する QUERYの実行。`
	`SQL4009 -- テーブル 2 の索引が作成されました。`

SMP 並列の使用可能化	はい
別名	索引プローブ 索引プローブ、プリロード 索引プローブ、区別 索引プローブ区別、プリロード 索引プローブ、キー選択
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムは DEPARTMENT テーブルに対する結合要件を満たすため、DeptNo 列で一時索引を作成することを選択します。その後一時索引プローブは一時索引に対して実行され、 2 つのテーブル間で結合基準を処理しました。 この場合には、一時索引の作成の間に DEPARTMENT テーブルに対して適用される可能性のある追加の選択はありませんでした。

一時バッファー

一時バッファーは、並列処理のような、操作を容易にするのを支援する一時オブジェクトです。照会の中間行を保管するのに使用される、ソートされていないデータ構造です。 一時バッファーと一時リストの主な違いは、一時バッファーは、結果を処理するために完全に取り込む必要がないという点です。

一時バッファーは、照会の並列と非並列部分の間のシリアライゼーション・ポイントの役目を果たします。バッファーから行をフェッチする操作は並列で実行できますが、バッファーを取り込むのに使用されるこの操作は並列で実行できません。 索引走査操作および索引プローブ操作は SQL Query Engine で使用可能な SMP 並列とは見なされないので、一時バッファーがこのエンジンで必要とされます。こうした索引操作を並列で実行する Classic Query Engine とは異なり、 SQL Query Engine は必要な作業を索引操作で細分化せずに、並列処理の利点を最大限に生かします。バッファーは、索引操作へのアクセスをシリアライズ化することにより、照会を並列処理で処理可能にするのに使用されます。同時に、照会内の残りの作業を並列に処理できるようにします。

一時バッファーは内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

バッファー走査:

バッファー走査操作は、DB2 Symmetric Multiprocessing を使用して照会を処理しますが、照会の一部が並列処理で処理できない場合に使用されます。バッファー走査は、照会の並列可能な部分と非並列部分での行のアクセスを制御するゲートウェイとしての役割を果たします。

複数のスレッドを使用してバッファーから選択行をフェッチでき、それにより照会は残りの処理を並列で実行できるようになります。しかし、バッファーは非並列で取り込まれます。

バッファー走査操作は、一時リスト・オブジェクトで実行されるリスト走査操作と同じです。 走査操作の開始前に、バッファーを完全に取り込む必要はないことが主な違いです。一時リストは、行をフェッチする前にリストを完全に取り込みます。

表 22. バッファー走査属性

データ・アクセス方式	**バッファー走査**
説明	この一時バッファー内のすべての行を順次走査して処理します。 SMP 並列処理を使用可能にして、照会の非並列部分で実行します。

表 22. バッファー走査属性 (続き)

データ・アクセス方式	バッファー走査
利点	• 一時バッファーは、非並列の照会の一部で並列処理を使用可能にするのに使用できます。 • 行のフェッチを開始するのに、一時バッファーを完全に取り込む必要はありません。
考慮事項	通常、要求を満たすのにキー列が必要ではない場合に、照会の一部分が何度も処理されないようにするために使用されます。
使用される可能性が高い場合	• DB2 Symmetric Multiprocessing の利点を最大限に生かすために、照会を試行する場合 • 照会の一部を並列で実行できない場合 (たとえば、索引走査または索引プローブ)

SQL ステートメントの例	`CHGQRYA DEGREE(*OPTIMIZE)`
	`CREATE INDEX X1 ON`
	`  Employee (LastName, WorkDept)`
	
	`SELECT * FROM Employee`
	`WHERE WorkDept BETWEEN 'A01' AND 'E01'`
	`AND LastName IN ('Smith', 'Jones', 'Peterson')`
	`OPTIMIZE FOR ALL ROWS`

使用を示すメッセージ	• 最適化プログラム・デバッグ:
	`CPI4328 -- Query によってファイル X1 のアクセス・パスが使用された。`
	`CPI4330 -- 8 個のタスクがファイル EMPLOYEE の並行索引走査に使用された。`
	• PRTSQLINF:
	`SQL4027 -- アクセス・プランが、システムにインストールされている DB2 SMP によって保管さ`
	`SQL4008 -- テーブル 1 に索引 X1 が使用された。`
	`SQL4011 -- テーブル 1 に索引スキャン・キー行位置が使用された。`
	`SQL4030 -- テーブル 1 の並行スキャンに 8 タスクが指定された。`

SMP 並列の使用可能化	はい
別名	適用されない
Visual Explain アイコン	

前述の例を使用すると、最適化プログラムはテーブルに対して索引プローブ操作を実行するため、既存の索引 X1 を使用することを選択します。この照会の残りの処理 (例えばテーブル・プローブ操作) 速度を上げるには、DB2 Symmetric Multiprocessing を使用して、テーブルでのランダム・プローブを実行します。索引プローブ操作は SQL Query Engine で有効な SMP 並列処理ではないため、照会のこの部分は一時バッファーに置かれ、選択された索引項目へのアクセスを制御します。

待ち行列

待ち行列は、最適化プログラムが再帰的照会の再帰を、それに必要なデータ値を待ち行列に置くことによりフィードするための一時オブジェクトです。このデータには通常、再帰的結合述部で使用される値、および再帰的処理中に累積または操作されるその他の再帰的データが含まれます。

待ち行列では 2 種類の操作が可能です。

- 待ち行列への挿入: データを待ち行列に入れる
- 待ち行列からの除去: データを待ち行列から除去する

待ち行列には再帰のフィードに必要なデータ、または再帰的処理によって直接変更されるデータのみが含まれるため、このデータ構造は効率的です。また、そのサイズは最適化プログラムによって管理されます。

最適化プログラムによって作成される他の一時オブジェクトとは異なり、基礎となる照会ノード・ツリーによる待ち行列へのデータの取り込みは同時に行われるのではありません。待ち行列は実際、再帰をフィードする値のリアルタイムの一時保持領域です。この点に関して、待ち行列は ALWCPYDTA(*NO) が指定された場合に照会の実行を防止しないので、一時的なものとは見なされません。データは依然として照会を出入りでき、同時に、追加の結合行の取得で使用するために待ち行列に再帰的な値が挿入されるからです。

待ち行列は内部データ構造であり、データベース・マネージャーだけが作成できます。

Visual Explain アイコン:

待ち行列への挿入:

待ち行列への挿入操作中、項目は、再帰的結合述部または再帰的処理の一部として扱われるデータによって使用されるキー値を含む待ち行列に入れられます。最適化プログラムは常に、Union All の真上の照会ノードで必須の再帰的データを収集するために、待ち行列への挿入操作を提供します。

表 23. 待ち行列への挿入属性

データ・アクセス方式	**待ち行列への挿入**
説明	追加の再帰が必要な待ち行列に項目を配置します
利点	• 再帰のためのソースとして必要です。再帰的処理に必要な値のみを待ち行列に入れます。各項目のライフ・スパンは、それが待ち行列から除去されるまでの短期間です。 • 待ち行列の各項目は、同じ rcte/view から再帰的な複数の反復全選択をシードします。
使用される可能性が高い場合	再帰的照会で必要なアクセス方式

表 23. 待ち行列への挿入属性 (続き)

データ・アクセス方式	待ち行列への挿入

SQL ステートメントの例	`WITH RPL (PART, SUBPART, QUANTITY) AS`
	`( SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY`
	`FROM PARTLIST ROOT`
	`WHERE ROOT.PART = '01'`
	`UNION ALL`
	`SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY`
	`FROM RPL PARENT, PARTLIST CHILD`
	`WHERE PARENT.SUBPART = CHILD.PART`
	`)`
	`SELECT DISTINCT PART, SUBPART, QUANTITY`
	`FROM RPL`

使用を示すメッセージ	待ち行列への挿入の使用を示す明示的なメッセージはありません
SMP 並列の使用可能化	はい
別名	適用されない
Visual Explain アイコン	

親、子の関係を反映するデータが循環し、無限の再帰ループを引き起こす可能性がある場合、再帰的照会の定義で CYCLE オプションを使用してください。 CYCLE は、特定の関連 (祖先チェーン) 行のセットに関して、すでにアクセスされた再帰的キー値が再び待ち行列に入れられないようにします。

指定された親子階層配列の再帰の結果を戻すには、再帰的照会の定義で SEARCH オプションを使用してください。検索の選択項目は、「深さ優先」または「幅優先」です。「深さ優先」は、それぞれの直接の子の子孫すべてが戻されてから次の子が戻されることを意味します。「幅優先」は、それぞれの子が戻されてからそれらの子が戻されることを意味します。 SEARCH では、関係キー (列は親子関係および深さまたは幅の検索タイプを構成する) の指定だけでなく、指定された順序を完全に実施するために、提供された順序列のメイン照会の ORDER BY 文節も必要とします。

待ち行列からの除去:

待ち行列からの除去操作中、待ち行列から項目が取り出され、再帰的参照によって指定されるそれらの値は再帰的結合プロセスにフィードバックされます。

最適化プログラム常に、指定する照会中の再帰的共通テーブル式または再帰的ビューの各参照ごとに、対応する操作の待ち行列への挿入と待ち行列からの除去の対を提供します。待ち行列から取り出す項目がなくなると、再帰は終了します。

表 24. 待ち行列からの除去属性

データ・アクセス方式	待ち行列からの除去
説明	待ち行列から項目を除去し、再帰結合および再帰的処理を介して操作されるその他のデータ値をフィードする再帰的結合述部の少なくとも 1 つの側を提供します。待ち行列からの除去は常に、制約付きの内部結合の左側で、結合の右側はターゲットの子行になります。
利点	• 再帰的な値に対して非常に迅速にアクセスできるようになります。 • 再帰的データ値に対してローカル述部の事後選択ができるようになります。

表 24. 待ち行列からの除去属性 (続き)

データ・アクセス方式	待ち行列からの除去
使用される可能性が高い場合	• 再帰的照会で必要なアクセス方式 • 待ち行列から除去される単一の値は、同じ rcte/view を参照する複数の反復全選択の再帰をフィードできます

SQL ステートメントの例	`WITH RPL (PART, SUBPART, QUANTITY) AS`
	`( SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY`
	`FROM PARTLIST ROOT`
	`WHERE ROOT.PART = '01'`
	`UNION ALL`
	`SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY`
	`FROM RPL PARENT, PARTLIST CHILD`
	`WHERE PARENT.SUBPART = CHILD.PART`
	`)`
	`SELECT DISTINCT PART, SUBPART, QUANTITY`
	`FROM RPL`

使用を示すメッセージ	待ち行列からの除去の使用を示す明示的なメッセージはありません
SMP 並列の使用可能化	はい
別名	適用されない
Visual Explain アイコン	

並列で処理されるオブジェクト

DB2 Symmetric Multiprocessing フィーチャーを使用すれば、最適化プログラムは並列処理を含む付加的なデータ検索方式を使用することができます。対称型マルチプロセッシング (SMP) は、単一システム上に設けられた並列処理の形式です。このシステムでは、メモリーとディスク・リソースを共用する複数 (CPU および入出力) プロセッサーが、同時に単一の終了結果を得るために処理を行います。

この並列処理とは、データベース・マネージャーが 1 回の照会で同時に複数 (あるいはすべて) のシステム・プロセッサーを作動させることができるということを意味します。 CPU 制約の照会のパフォーマンスは、マルチプロセッサー・システムのこのフィーチャーを使用して、複数のプロセッサー間のプロセッサー・ロードを分散することによってかなり向上します。

前記の表は、DB2 Symmetric Multiprocessing フィーチャーの利点を生かすために有効なデータ・アクセス方式を示しています。しかし、注意すべき重要な事柄は、SQL Query Engine と Classic Query Engine では並列の実装方法が異なるということです。

処理要件

並列処理では、SMP 並列処理が以下のいずれかの方法によって使用可能になっていなければなりません。

- システム値 QQRYDEGREE
- 照会オプション・ファイル
- 照会属性の変更 (CHGQRYA) コマンドの DEGREE パラメーター
- SQL SET CURRENT DEGREE ステートメント

並列処理が可能になると、一連のデータベース・システム・タスクまたはスレッドは、データベース・マネージャーが使用する際のシステム始動時に作成されます。 データベース・マネージャーはそのタスクを使用して、種々のディスク装置からデータを処理および検索します。これらのタスクは複数のプロセッサーで

同時に実行することができるので、照会の経過時間を軽減することができます。並列照会の入出力と CPU 処理は、このタスクが行いますが、使用する入出力と CPU リソースのアカウンティングはアプリケーション・ジョブに転送されます。アプリケーションのこのタイプの入出力と CPU リソースの要約は、活動ジョブの処理 (WRKACTJOB) コマンドで正確に継続して表示することができます。

このジョブを共用記憶域プールで *CALC ページング・オプションを指定して実行する必要があります。このことによって、アクティブ・メモリーはより効果的に使用できます。

関連概念

50 ページの『ネストされたループ結合の実施』
DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

関連資料

135 ページの『照会の属性の変更』
照会属性の変更 (CHGQRYA) CL コマンドにより、あるいは System i ナビゲーター照会属性の変更インターフェースを使用して、一定のジョブの間に実行する照会の属性の各種タイプを変更することができます。

関連情報

SET CURRENT DEGREE statement

Query と索引の並列処理のシステム値

パフォーマンスの自動調整

活動ジョブ処理 (WRKACTJOB) コマンド

照会属性の変更 (CHGQRYA) コマンド

データの自動分散

DB2 for i5/OS は、データが割り振られている補助記憶域プール (ASP) で使用できるディスク装置に自動的にデータを分散します。これにより、データはユーザーの介入なしに分散されます。

分散させることにより、データベース・マネージャーは並列で種々のディスク装置上の行のブロックを容易に処理することができます。 DB2 for i5/OS は ASP 内のディスク装置間のデータを分散しますが、データのエクステント (連続した一連のデータ) の割り振りが公平に分散されないこともあります。これは、装置上のスペースの割り振りが均等でない場合、または新しい装置が ASP に追加されている場合に発生します。テーブル・データ・スペースの割り振りは、テーブルの保管、削除、その後の復元によって再び分散される場合があります。

すべてのディスク装置にデータを均等に分散して維持すると、照会処理でより良いスループットをもたらすことができます。最適化プログラムは別のプランに置換するコストを計算する際に、ディスク装置の数およびデータをそうした装置に拡散する方法を考慮に入れます。

照会の処理: 概説

この Query 最適化プログラムの概説では、システム・リソースをより効率的に実行し、使用する照会を設計するための指針を説明しています。

この概説では、Query 最適化プログラムで最適化される照会について説明するとともに、SQL、OPNQRYF、API (QQQQRY)、ODBC、および Query AS/400 用照会などのインターフェースにも触れます。照会の結果は、この指針を適用するかどうかに関係なく正確です。

注: この概説で取り上げる内容は複雑です。概念をより良く理解するために、この情報を読みながら System i 製品を実際に試してみるとよいでしょう。

DB2 for i5/OS が照会を処理する方法が分かれば、この概説で説明している指針のパフォーマンスへの影響も比較的分かりやすくなります。 DB2 for i5/OS 照会処理には次の 2 つの主なコンポーネントがあります。

- システムがデータにアクセスする方法。

  これらの方式は、ディスクからデータを検索するために使用するアルゴリズムです。これらの方式には、索引の使用法および行選択技法があります。さらに、DB2 Symmetric Multiprocessing オペレーティング・システム・フィーチャーでは並列アクセス方式が利用できます。
- Query 最適化プログラム。

  Query 最適化プログラムは、照会を実現するために使用できる有効な技法を識別し、最も効率のよい技法を選択します。

Query 最適化プログラムが照会をより効果的にする方法

SELECT などのデータ操作ステートメントは、ユーザーがどのデータを必要としているかを指定するだけで、そのデータを取り出す方法は指定しません。データへのこのパスは、最適化プログラムによって選択され、アクセス・プランに保管されます。 Query 最適化プログラムがこのタスクを実行するために使用する技法を理解しておく必要があります。

最適化プログラムは、DB2 for i5/OS の重要な部分で、次のことを行います。

- データベースのパフォーマンスに影響する主要な決定を行います。
- 照会の実施に利用できる技法を識別します。
- 最適な技法を選択します。

一般的な Query 最適化のヒント

照会をできるだけ高速に実行するのに役立つヒントのいくつかを次に示します。

- 左端のキー列がユーザーの選択述部に一致する索引を作成して、最適化プログラムへの選択値 (キー範囲見積もり) の提供を容易にします。
- 結合照会の場合、ユーザーの結合列に一致する索引を作成して、最適化プログラムが一致している行の平均数を決定しやすいようにします。
- 照会に必要な列だけを指定することによって、余分なマッピングを最小にします。たとえば、SELECT * と指定する代わりに、SQL SELECT ステートメントで、照会に必要な列だけを指定します。また、列を更新する必要がない場合には、FOR FETCH ONLY を指定しなければなりません。
- ユーザーの照会でテーブル走査アクセス方式を頻繁に使用する場合、物理ファイル・メンバーの再編成 (RGZPFM) コマンドを使用して削除済み行をテーブルから取り除くか、または 物理ファイルの変更 (CHGPF) REUSEDLT (*YES) コマンドを使用して削除済み行を再使用します。

次のオプションの使用を検討してください。

- パフォーマンスの向上を図るために、ALWCPYDTA(*OPTIMIZE) を指定して、Query 最適化プログラムがデータの一時コピーを作成できるようにします。System i Access用 ODBC ドライバーおよび照会管理

ドライバーは常にこのモードを使用します。 ALWCPYDTA(*YES) が指定される場合、Query 最適化プログラムはデータのコピーなしで照会を実施しようとしますが、必要な場合にはコピーを作成することもできます。 ALWCPYDTA(*NO) が指定されると、データのコピーは許可されません。 Query 最適化プログラムが一時コピーを使用しないプランを検出できない場合は、照会は実行できません。

- SQL では、CLOSQLCSR(*ENDJOB) または CLOSQLCSR(*ENDACTGRP) を使用して、オープン・データ・パスが今後の呼び出しに対してオープンのままとなるようにします。
- DLYPRP(*YES) を指定して、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで SQL ステートメントの妥当性検査を遅らせます。このオプションによって、余分な妥当性検査が除かれるため、パフォーマンスが向上します。
- ALWBLK(*ALLREAD) を使用して、読み取り専用カーソルに対して行のブロック化を可能にします。

関連情報

物理ファイル・メンバーの再編成 (RGZPFM) コマンド

物理ファイル変更 (CHGPF) コマンド

アクセス・プランの妥当性検査

アクセス・プランは、各照会要求を満足させるために必要な動作を記述する制御構造です。これには、データに関する情報とその抽出方法が含まれます。どの照会についても、最適化を行うたびに、Query 最適化プログラムは要求されたデータへのアクセス方式に関する最適化プランを作成します。

パフォーマンス向上のため、いったんアクセス・プランが作成されたら、将来照会が実行される場合にそのアクセス・プランを使用できるよう、保管されます (以下の例外を参照)。しかし、最適化プログラムには動的再計画機能があります。これは、以前作成 (および保管) されたプランが検出されても、より良いプランが可能であると最適化プログラムが判断した場合にプランが再作成される場合があることを意味します。これによって、保管されたプランを利用しつつ、最大限の柔軟性を発揮することができます。

- 動的 SQL の場合、アクセス・プランは準備の際、またはオープン時に作成されます。しかし最適化では、最適なプランの決定にホスト変数値が使用されます。そのため、準備時に作成されたプランは、照会の初回オープン時に再作成される場合があります (ホスト変数値が存在する場合)。
- 組み込み静的 SQL を含む i5/OS プログラムの場合、アクセス・プランはコンパイル時にまず作成されます。ここでも、最適化では最適なプランの決定にホスト変数値が使用されるため、コンパイル時のプランは照会の初回オープン時に再作成される場合があります。
- 照会ファイルのオープン (OPNQRYF) の場合、アクセス・プランは作成されますが、保管されません。OPNQRYF コマンドを実行するたびに、新規のアクセス・プランが作成されます。
- Query/400 の場合、アクセス・プランが照会定義オブジェクトの一部として保管されます。

プランが保管される上記のすべての場合に (静的 SQL を含む)、照会は時間とともに実行されるので引き続き動的再計画を適用することができます。

このアクセス・プランは、照会がオープンされたときに妥当性検査されます。妥当性検査には、以下の事柄が含まれます。

- アクセス・プランと同じテーブルが照会で参照されていることを検証します。たとえば、テーブルが削除されて再作成されなかったこと、または *LIBL を使用して解決されたテーブルが変更されていないことを検証します。
- 照会を実施するのに使用される索引がまだ存在していることを検証します。
- テーブル・サイズ選択または述部選択に大きな変化がないことを確かめます。
- QAQQINI オプションが変更されていないことを検証します。

単一テーブル最適化

実行時に、最適化プログラムは、データベースの現行の状態に基づき*実施コスト* を計算して、照会のための最適なアクセス方式を選択します。最適化プログラムは、決定を下す際に入出力コストと CPU コストの 2 つのコストを使用します。最適化プログラムの目的は、入出力および CPU コストの両方を最小化することです。

各テーブルへのアクセスの最適化

最適化プログラムは、各テーブルのデータにアクセスする最適な方式を選ぶために、一般的な一連の指針を使用します。最適化プログラムが行うことは次のとおりです。

- 選択文節の各述部についてデフォルトのフィルター係数を決定します。
- 選択述部が索引の左端のキーに一致するときに、キー範囲見積もりを実行して、または使用可能な場合に列統計を使用して、述部の正しいフィルター係数を決定します。
- 索引を必要としない場合に、テーブル走査処理のコストを決定します。
- 索引を必要とする場合に、テーブルに関して索引を作成するコストを決定します。この索引は、テーブル走査を実行するか、または索引から索引を作成することによって、作成されます。
- 適切な場合に、分類ルーチンまたはハッシュ方式を使用するコストを決定します。
- 索引プローブまたは索引走査を使用して、既存の索引を使用するコストを決定します。
  - 索引を順序付けします。SQE の場合、通常、アクセスする項目が最も少ない索引が最初に調べられるというように、索引は順序付けされます。 CQE の場合、通常は作成された日付が新しいものから古いものへと索引は順序付けされます。
  - 使用可能な各索引では、最適化プログラムは以下の事柄を行います。
    - 索引が選択基準を満たしているかどうかを判断します。
    - 索引プローブまたは索引走査と、使用可能なテーブル・プローブを実行するのに必要となる入出力の数および CPU コストを見積もり、索引を使用するコストを決定します。
    - この索引の使用コストと前回のコスト (現行の最低) を比較します。
    - 安い方を選択します。
    - 最適化プログラムが別の索引を探さなくなるまで、最適な索引の検索を続行します。

    SQE の場合、索引は順序付けされているため、最適の索引が最初に調べられ、索引が以前に選択した最適の索引よりもコストがかかると検索を終了します。

    CQE の場合、*時間制限* は、最適化プログラムが実施の選択に費やす時間を制限します。これは、これまでに費やした時間数と検出された現時点での最低の実施コストに基づいています。この目的は、最適化プログラムが照会の最適化に時間がかかりすぎて、実際に照会を実行する時間よりも長くならないようにすることです。動的 SQL 照会は、最適化プログラム時間制約事項に従います。静的 SQL 照会の最適化時間には制約はありません。 OPNQRYF の場合、OPTALLAP(*YES) を指定すると、最適化時間は限定されません。小さなテーブルの場合、Query 最適化プログラムは照会の最適化にほとんど時間を費やしません。大きなテーブルの場合、Query 最適化プログラムはより多くの索引を検討します。一般的に、最適化プログラムは、最適化時間を使い果たす前に (1 つの結合の各テーブルごとに) 5 つか 6 つの索引を検討します。このため、最適化プログラムは普通、大きなテーブルに対する照会を分析する場合には、より長い時間を費やします。
- 一時ビットマップを使用するコストを決定します。
  - ビットマップに使用できる索引を順序付けします。通常、最少の項目を選択する索引が最初に調べられます。

– ビットマップにこの索引を使用するコスト、およびこのビットマップと以前に生成されたビットマップをマージするコストを決定します。
– このビットマップ・プランのコストが以前のビットマップ・プランのコストより低い場合には、ビットマップ・プランの検索を続行します。

- テーブルのデータにアクセスする可能な方法を調べてから、調べたすべてのプランから最善のプランを最適化プログラムは選択します。

結合の最適化

結合操作は、優れたパフォーマンスを達成するために特に注意を必要とする複雑な機能です。このセクションでは、DB2 for i5/OS の結合照会の実施方法、および Query 最適化プログラムが行う最適化選択方法について説明します。また、パフォーマンスの問題の回避または解決に役立つ設計に関するヒントおよび技法についても説明します。

ネストされたループ結合の実施

DB2 for i5/OS は、**ネストされたループ**結合方式を提供します。この方式では、結合の中のテーブルの処理が順序付けられます。この順序は、**結合順序**と呼ばれます。最後の結合順序の中の最初のテーブルは、**1 次テーブル**と呼ばれます。他のテーブルは、**2 次テーブル**と呼ばれます。各結合テーブルの位置は、**ダイヤル**と呼ばれます。

ネストされたループは、2 次テーブル上の索引、ハッシュ・テーブル、あるいは 2 次テーブル上のテーブル走査 (到着順) のいずれかを使用して実施されます。一般に、結合は索引またはハッシュ・テーブルのどちらかを使用して実施されます。

索引でネストされたループ結合の実施

結合中に、DB2 for i5/OS は次のことを行います。

1. 1 次テーブルにローカルな述部によって選択された最初の 1 次テーブル行にアクセスします。
2. 1 次テーブルの中の結合列からキー値を作成します。
3. 以下の事柄は、最初の 2 次テーブルへのアクセスに依存しています。
   - 2 次テーブルにアクセスするのに索引を使用している場合、基数索引プローブを使用して、キーが 2 次テーブルの結合条件またはローカル行選択列に一致する索引を使用する最初の 2 次テーブルの結合条件を満たす最初の行を見つけます。
   - 可能な場合にビットマップ選択を適用します。

     各 2 次ダイヤルからの結合条件を満足するすべての行は、索引を使用して検出されます。行はランダム順序で 2 次テーブルから検索されます。このランダム・ディスク入出力時間は、多くの場合、照会の処理時間の大部分を占めます。指定された 2 次ダイヤルは、その前の 2 次ダイヤルのそれぞれについて結合条件を満足する 1 次ダイヤルおよび前の 2 次ダイヤルから選択された各行につき 1 回ずつ検索されるので、後続のダイヤルに対して大量の検索が行われる可能性があります。後続のダイヤルの選択処理で効率が損なわれるようなことがあると、照会の処理時間がかなり延長される可能性があります。結合照会についてのパフォーマンスを考慮すると、結合照会の実行時間を数時間から数分に減らすことができる理由はここにあります。

     効果的な索引が見つからない場合には、一時索引が作成される可能性があります。一部の結合照会は、すべての結合キーに関して索引が存在する場合でも 2 次ダイヤル上に一時索引を作成します。比較的長く稼働する照会の 2 次ダイヤルにとって効率は非常に重要なので、 Query 最適化プログラムは、そのダイヤルにローカル行選択を渡す項目だけが含まれる一時索引の作成を選択することがあ

ります。この行選択の前処理を行うことにより、データベース・マネージャーは、行をダイヤルに突き合わせるたびに行う代わりに、1 回の受け渡しで行選択を処理することができます。

- 2 次テーブルにアクセスするのにハッシュ・テーブル・プローブを使用する場合、最初のプローブのテーブルについてのローカル選択で抜粋した行のすべてを含むハッシュ一時結果テーブルを作成します。ハッシュ・テーブルの構造は同じ結合値の行で、同じハッシュ・テーブルの区画に (クラスター化されて) ロードされたものです。任意の与えられた結合値についての行の位置は、結合値にハッシュ関数を適用することによって検出することができます。

  ハッシュ・テーブル・プローブを使用したネスト・ループ結合には、索引プローブを使用したネスト・ループ結合に比べて幾つかの利点があります。

  – ハッシュ一時結果テーブルの構造は索引の構造よりも単純なので、ハッシュ・テーブルを構築し、プローブするために CPU 処理をそれほど必要としません。
  – ハッシュ結果テーブルの行には照会で要求したデータのすべてが入っているので、ハッシュ・テーブルをプローブするときにランダム入出力でテーブルのデータ・スペースをアクセスする必要がありません。
  – 結合値がクラスター化されている場合には、与えられた結合値の一致行は通常 1 回の入出力要求でアクセスすることができます。
  – ハッシュ一時結果テーブルは SMP 並列処理を使用して構築することができます。
  – 索引と違い、ハッシュ・テーブルの項目は、基礎テーブル内の列の変更を反映する更新を行いません。ハッシュ・テーブルの存在によって、システムでの他の更新ジョブの処理コストが影響を受けることはありません。
- 2 次テーブルにアクセスするのにソート済みリスト・プローブを使用する場合、最初のプローブのテーブルについてのローカル選択で抜粋した行のすべてを含むソート済みリストを作成します。ソート済みリスト・テーブルの構造は、同じ結合値の行で、リストに相互にソートされたものです。任意の与えられた結合値についての行の位置は、結合値を使用してプローブして検出することができます。
- 2 次テーブルにアクセスするのにテーブル走査を使用する場合、 2 次側を走査して、最初の 2 次テーブルで結合条件を満たす最初の行を見つけます。その際、テーブル走査を使用して、2 次テーブルの結合条件またはローカル行選択列に一致する行を見つけます。2 次テーブルがユーザー定義のテーブル関数の場合、結合はテーブル走査によって実施されることもあります。

4. 最初の 2 次ダイヤルにローカルな残りの選択があればそれを適用することにより、行が選択されたかどうかを判別します。

   2 次ダイヤル行を選択しない場合は、結合条件を満たす次の行が検出されます。ステップの 1 から 4 は、結合条件と残りの選択の両方を満足する行がすべての 2 次テーブルから選択されるまで、繰り返されます。
5. 結果の結合行が返されます。
6. もう一度最後の 2 次テーブルを処理して、そのダイヤルの中で結合条件を満足する次の行を検出します。

   この処理の間、結合条件を満足する行をそれ以上選択できないとき、処理は論理上の直前のダイヤルをバックアップし、その結合条件を満足する次の行を読み取ろうとします。
7. 1 次テーブルからのすべての選択行が処理されると、処理は終了します。

ネストされたループ結合の特徴は、次のとおりです。

- 順序付けとグループ化が指定されており、すべての列が単一テーブルにあり、そのテーブルが 1 次とするのに適格である場合、最適化プログラムは 1 次としてのそのテーブルとの結合のコスト、および索引のグループ化および順序付けを実行するコストを見積もります。
- 順序付けおよびグループ化が 2 つ以上のテーブルに指定された場合、または一時が許可される場合、DB2 for i5/OS は照会の処理を次の 2 つの部分に分割します。
  1. 順序付け処理またはグループ化処理を除外して結合選択を実行し、結果の行を一時作業テーブルに書き込みます。これにより、最適化プログラムは結合照会の任意のテーブルを 1 次テーブルの候補と見なすことができます。
  2. 次に、一時作業テーブルのデータに関して順序付け処理またはグループ化処理を行います。

ハッシュ結合を使用できない照会

ハッシュ結合は次の照会には使用できません。

- ハッシュ結合は、物理ファイル、あるいは読み取りトリガーがあるテーブルが関係する照会には使用できません。
- SQL ROLLBACK HOLD ステートメントまたは ROLLBACK CL コマンドの結果として、カーソル位置の復元要求。 *NONE 以外のコミットメント制御レベルを使用している SQL アプリケーションの場合は、プリコンパイラー・パラメーター ALWBLK の値として指定された *ALLREAD が必要です。
- ハッシュ結合は、結合条件が等号演算子以外の結合照会のテーブルには使用できません。
- CQE は、以下のいずれかが照会に含まれる場合にはハッシュ結合をサポートしません。
  - 副照会。ただし、照会内のすべての副照会を内部結合に変換できる場合を除く。
  - UNION または UNION ALL
  - 左辺外部結合あるいは例外結合の実行。
  - DDS で作成された結合論理ファイルの使用。

関連概念

45 ページの『並列で処理されるオブジェクト』
DB2 Symmetric Multiprocessing フィーチャーを使用すれば、最適化プログラムは並列処理を含む付加的なデータ検索方式を使用することができます。対称型マルチプロセッシング (SMP) は、単一システム上に設けられた並列処理の形式です。このシステムでは、メモリーとディスク・リソースを共用する複数 (CPU および入出力) プロセッサーが、同時に単一の終了結果を得るために処理を行います。

関連資料

8 ページの『テーブル走査』
テーブル走査は、テーブルに対して実行できる最も簡単でシンプルな操作です。テーブル内の行が照会で指定された選択基準を満たすかどうかを判別するために、すべての行を順次処理します。 この操作を実行すると、テーブルの入出力スループットが最大になります。

23 ページの『ソート・リスト・プローブ』
ソート・リスト・プローブ操作は、プローブ探索操作に基づいて一時ソート・リストから行を取り出すのに使用します。

20 ページの『ハッシュ・テーブル・プローブ』
ハッシュ・テーブル・プローブ操作は、プローブ探索操作に基づいて一時ハッシュ・テーブルから行を取り出すのに使用します。

12 ページの『基数索引プローブ』
基数索引プローブ操作は、テーブルから行をキー順に取り出すのに使用します。基数索引プローブと基数索引走査の主な相違点は、取り出されている行をサブセット化するプローブ操作では、戻される行が最初に識別される必要があるという点です。

結合最適化アルゴリズム

Query 最適化プログラムは、結合列、結合演算子、ローカル行選択、ダイヤル実施、索引の使用法、および結合照会のダイヤル順序付けを決定する必要があります。

結合列と結合演算子は、次の状況によって異なります。

- 照会の結合列指定
- 結合順序
- 結合列と他の行選択との対話

ダイヤルに組み込まれていない結合指定は、後続のダイヤルで処理できるようになるまで据え置かれるか、またはこのダイヤルに内部結合が行われる場合は、行選択として処理されます。

指定のダイヤルでは、そのダイヤルの結合列として使用できる唯一の結合指定は、*前の* ダイヤルに結合されている結合指定に限られます。たとえば、2 番目のダイヤルについては、結合条件を満足するために使用できる唯一の結合指定は、 1 次ダイヤルで列を参照する結合指定です。同様に、3 番目のダイヤルは、1 次ダイヤルと 2 番目のダイヤルの列を参照する結合指定しか使用できません。以下も同様です。後続のダイヤルを参照する結合指定は、参照されるダイヤルが処理されるまで据え置かれます。

注: OPNQRYF の場合、結合演算子の 1 つのタイプだけが、左辺外部結合または例外結合に許されています。つまり、すべての結合条件の結合演算子は同じでなければなりません。

2 次ダイヤルにアクセスするための既存の索引を探している場合、Query 最適化プログラムは索引の左端のキー列を調べます。指定したダイヤルと索引の場合は、左端のキー列を使用する結合指定を使用することができます。たとえば、次のとおりです。

```
DECLARE BROWSE2 CURSOR FOR
 SELECT * FROM EMPLOYEE, EMP_ACT
  WHERE EMPLOYEE.EMPNO = EMP_ACT.EMPNO
    AND EMPLOYEE.HIREDATE = EMP_ACT.EMSTDATE
 OPTIMIZE FOR 99999 ROWS
```

EMPNO、(PROJNO、) および EMSTDATE の各キー列の EMP_ACT 上の索引では、結合操作は EMPNO 列でしか行われません。結合を実行した後で、EMSTDATE 列により索引走査キー選択が行われます。

Query 最適化プログラムは、2 次ダイヤルにとって最適な索引の使用を選ぶときに、ローカル行選択を使用することもあります。上記の例でローカル述部を次のように表したとすると、

```
DECLARE BROWSE2 CURSOR FOR
 SELECT * FROM EMPLOYEE, EMP_ACT
  WHERE EMPLOYEE.EMPNO = EMP_ACT.EMPNO
    AND EMPLOYEE.HIREDATE = EMP_ACT.EMSTDATE
    AND EMP_ACT.PROJNO = '123456'
 OPTIMIZE FOR 99999 ROWS
```

EMPNO、PROJNO、および EMSTDATE の各列の索引は、3 つのすべてのキー列に対して結合と選択を 1 つの操作に組み合わせることによって最大限に活用されます。

一時索引を作成する場合、左端のキー列は、そのダイヤル位置で使用可能な結合列です。そのダイヤルのすべてのローカル行選択は、項目を選択して一時索引に取り込むときに処理されます。一時索引は、選択/除外キー付き論理ファイル用に作成された索引に似ています。上記の例の一時索引は、EMPNO および EMSTDATE のキー列を持ちます。

Query 最適化プログラムは、アクセス・パスの使用量を決定するときに、結合とローカル行選択の組み合わせをしようとするため、既存の索引の使用によって、一時索引と同じ利点のほとんどすべてを享受すること

ができます。上記の例では、既存の索引を使用するか、一時索引を作成するかのいずれかの使用法が用いられます。一時索引は、索引の作成中に適用される PROJNO のローカル行選択を指定して構築されます。この一時索引には、EMPNO および EMSTDATE のキー列が含まれます (結合選択との突き合わせのため)。代わりに、既存の索引が EMPNO、PROJNO、EMSTDATE (または PROJNO、 EMP_ACT、 EMSTDATE または EMSTDATE、PROJNO、EMP_ACT または ...) のキー列を指定して使用された場合、ローカル行選択が結合選択と**同時に**適用されることになります (一時索引の作成時の場合のような結合選択の前や、索引の最初のキー列が結合列と突き合わされる場合のみのような結合選択の後ではありません)。

既存の索引を使用する実施の方が、迅速なパフォーマンスが得られることになります。なぜなら、結合と選択処理は一時索引の作成によるオーバーヘッドなしで併用されるからです。しかし、既存の索引を使用することは、ローカル選択が 1 回だけではなく何回も実行されるため、一時索引に比べて入出力処理がやや遅くなる可能性があります。一般的に、既存の索引を、結合列と等号選択を左端のキーに使用している列との組み合わせに対するキー列で利用できるようにすると便利です。

結合順序の最適化

結合論理ファイルのどれかを参照する場合、結合順序は固定されています。同様に、OPNQRYF JORDER(*FILE) パラメーターが指定されている場合、または照会オプション・ファイルの (QAQQINI) FORCE_JOIN_ORDER パラメーターが *YES の場合には、結合順序は固定されます。

それ以外の場合、次の結合順序付けアルゴリズムを使用して、テーブルの順序が決定されます。

1. 1 次ダイヤルの候補として、それぞれのテーブルごとにアクセス方式を決定する。
2. ローカル行選択に基づいて各テーブルに返される行数の見積もりを行う。

   行の順序付けまたは GROUP BY 処理を指定している結合照会を 1 つのステップで処理している場合は、順序付け列またはグループ化列を指定しているテーブルを 1 次テーブルとする。
3. 1 次テーブルおよび最初の 2 次テーブルとして返された候補テーブルの結合の組み合わせごとに、アクセス方式、コストおよび予想行数を判別する。

   4 つのテーブル結合について予想される内部結合順序の組み合わせは、次のとおりです。

   ```
   1-2    2-1    1-3    3-1    1-4    4-1   2-3    3-2    2-4    4-2    3-4    4-3
   ```
4. 最も低い結合コストか選択された行の数の組み合わせ、またはその両方を選択する。
5. 前の 2 次テーブルに結合された残りの各テーブルについてコスト、アクセス方式、および予想行数を判別する。
6. 各テーブルについて、そのテーブルに関して最も低いコストのアクセス方式を選択する。
7. 最も低い結合コストか選択された行の数の 2 次テーブル、またはその両方を選択する。
8. 最も低いコストの結合順序を判別するまで、ステップ 4 から 7 を繰り返す。

注: 32 をダイヤルした後は、最適化プログラムが異なる方式を使用して、ファイルの結合順序を判別します。この場合のコストは、最も低くなるとは限りません。

照会に左辺または右辺外部結合あるいは右辺例外結合が含まれる場合、結合順序は固定されません。ただし、ON 文節のすべての FROM 列は、左辺または右辺外部結合あるいは例外結合の前のダイヤルから発生するはずです。たとえば、次のとおりです。

```
FROM A INNER JOIN B ON A.C1=B.C1
LEFT OUTER JOIN C ON B. C2=C.C2
```

この照会で使用可能な結合順序の組み合わせは以下のとおりです。

1–2–3、2–1–3、または 2–3–1

右辺外部または右辺外部例外結合は、反対のファイルで、左辺外部および左辺例外としてインプリメントされます。たとえば、次のとおりです。

```
FROM A RIGHT OUTER JOIN B ON A.C1=B.C1
```

これは、B LEFT OUTER JOIN A ON B.C1=A.C1 としてインプリメントされます。許可される結合順序は 2-1 だけです。

結合論理ファイルが参照されるか、または結合順序が指定したテーブル順序に強制されると、 Query 最適化プログラムは、指定した順序ですべてのダイヤルをループして、最低コストのアクセス方式を判別します。

関連情報

QUERY ファイル・オープン (OPNQRYF) コマンド

照会属性の変更 (CHGQRYA) コマンド

全外部結合

全外部結合は、SQE 最適化プログラムによってサポートされています。内部、左外部、または左例外のサポートされた結合タイプを反映するために、右外部結合または右例外結合が最適化プログラムによって書き直されるように、全外部結合も書き直されます。

A FULL OUTER JOIN B の全外部結合は、(A LEFT OUTER JOIN B) UNION ALL (B LEFT EXCEPTION JOIN A) と等価です。以下の例は、再書き込みについて説明しています。

```
  SELECT EMPNO, LASTNAME, DEPTNAME
   FROM CORPDATA.EMPLOYEE XXX
   FULL OUTER JOIN CORPDATA.DEPARTMENT YYY
   ON XXX.WORKDEPT = YYY.DEPTNO
```

この照会は、次のように書き直されます。

```
 SELECT EMPNO, LASTNAME, DEPTNAME
  FROM CORPDATA.EMPLOYEE XXX
  LEFT OUTER JOIN CORPDATA.DEPARTMENT YYY
  ON XXX.WORKDEPT = YYY.DEPTNO
 UNION ALL
 SELECT EMPNO, LASTNAME, DEPTNAME
  FROM CORPDATA.DEPARTMENT YYY
  LEFT EXCEPTION JOIN CORPDATA.EMPLOYEE XXX
  ON XXX.WORKDEPT = YYY.DEPTNO
```

複数の FULL OUTER JOIN 要求による照会 (例えば、A FULL OUTER JOIN B FULL OUTER JOIN C) は、迅速にこの複雑な再書き込み状態になります。これを次の例で説明します。照会がライブ・データ・モードで実行されない場合、最適化プログラムは、中間結果を一時データ・オブジェクトにカプセル化することによって、最適化中および実行時の両方でパフォーマンスを容易にすることを選択する場合があります。この一時データ・オブジェクトは、一度、最適化し、再書き込みのスキャンされたサイドおよび厳密に調べられたサイドの両方に差し込むことができます。これらの共有された一時データ・オブジェクトにより、要求を満たすために特定のテーブルを何度もパススルーする必要がなくなります。この例では、(A FULL OUTER JOIN B) の結果が、C との FULL OUTER 結合中にカプセル化の候補になります。

```
A FULL OUTER JOIN B  FULL OUTER JOIN C
```

この照会は、次のように書き直されます。

```
 ((A LEFT OUTER JOIN B) UNION ALL (B LEFT EXCEPTION JOIN A)) LEFT OUTER JOIN C )
 UNION ALL
 (C LEFT EXCEPTION JOIN ((A LEFT OUTER JOIN B) UNION ALL (B LEFT EXCEPTION JOIN A))
```

FULL OUTER は、全外部結合要求の両サイドが、結果の応答セットに NULL 値を生成し、照会の WHERE 文節で提供されたローカル選択の結果、FULL OUTER が LEFT OUTER または INNER JOIN に適切にダウングレードされることを意味します。 FULL OUTER 動作が必要であり、一部のローカル選択を適用したい場合は、FULL OUTER JOIN の ON 文節でローカル選択を指定するか、共通テーブル式を使用する必要があります。たとえば、次のとおりです。

```
WITH TEMPEMP AS (SELECT * FROM CORPDATA.EMPLOYEE XXX WHERE SALARY > 10000)
SELECT EMPNO, LASTNAME, DEPTNAME
  FROM TEMPEMP XXX
  FULL OUTER JOIN CORPDATA.DEPARTMENT YYY
  ON XXX.WORKDEPT = YYY.DEPTNO
```

結合 2 次ダイヤル用コスト見積もりおよび索引選択

Query 最適化プログラムが各種の可能なアクセスの選択を比較する際に、数値コスト値を各候補に割り当てて、その値を使用して処理時間が最も少なくて済む実施を判別します。このコスト計算値は CPU 時間と入出力時間の組み合わせです。

54 ページの『結合順序の最適化』のステップ 3 とステップ 5 では、Query 最適化プログラムは、指定したダイヤルの組み合わせに対してアクセス方式を選択して、コストを見積もる必要があります。行われた選択は、プローブを用いるプランを使用する必要がある点を除けば、行選択のそれと似ています。

コスト計算値は次の想定に基づきます。

- テーブル・ページと索引ページは、補助記憶装置から取り出す必要があります。たとえば、Query 最適化プログラムは、オブジェクト・アクセス設定 (SETOBJACC) CL コマンドを出せばテーブル全体をアクティブ・メモリーにロードできることを認識していません。このコマンドを使用すれば、照会のパフォーマンスを著しく向上できますが、Query 最適化プログラムは、テーブルの記憶域常駐状態を利用するために照会の実施を変更することはしません。
- 照会は、システム上で実行されている唯一のプロセスです。他のプロセスが同じリソースを使用することによって発生するシステムの CPU 使用率または入出力待ち状態の余地はありません。 CPU 関連コストは、照会を実行しているシステムの相対処理速度に従って見積もられます。
- 列の中の値は、テーブルに均一に分散されています。たとえば、テーブル内の行の 10% が同じ値を持っている場合、テーブル内の 10 行目ごとにその値が入っていると想定されます。
- 列の中の値は、キー定義が (A,B) の使用可能な索引がある場合を除き、行の中のその他の列の値と独立しています。マルチ・キー・フィールド索引を使用すると、最適化プログラムは列間の値の関連について検出できるようになります。たとえば、A という名前の列が、テーブルの中の行の 50% に 1 の値を持っている場合、しかも B という名前の列が行の 50% に 2 の値を持っている場合、A = 1 で B = 2 の行を選択する照会は、テーブルの中の行の 25% を選択することが予想されます。

2 次ダイヤルの結合コスト計算の主な係数は、以前のすべてのダイヤルで選択した行数と、以前のダイヤルから選択したそれぞれの行に一致する平均の行数です。これらの係数は両方とも、指定したダイヤルの一致行数を見積もることによって算出することができます。

結合演算子が等号でない場合には、予想一致行数は、次のデフォルトのフィルター係数に基づいて求められます。

- より小さい、より大きい、より小か等しい、またはより大か等しい場合は 33%
- 等しくない場合は 90%
- BETWEEN 範囲の場合は、25% (OPNQRYF %RANGE)
- 各 IN リスト値の場合は、10% (OPNQRYF %VALUES)

たとえば結合演算子が、より小さいであった場合には予想一致行数は .33 * です (ダイヤル内の行数)。現行ダイヤルに結合指定が有効になっていない場合は、カルテシアン積が演算子であると想定されます。カルテシアン積の場合、一致行数は、ローカル行選択を索引に適用できない限りダイヤル内のすべての行になります。

結合演算子が等号の場合、予想行数は、指定した値の重複行の平均数になります。

関連情報

オブジェクト・アクセスのセット (SETOBJACC) コマンド

推移的閉包によって生成される述部

結合照会の場合、Query 最適化プログラムは、追加の選択を生成するために何か特殊な処理を行う場合があります。照会に属する述部セットが、余分な述部を論理的に推論する場合、Query 最適化プログラムは追加の述部を生成します。その目的は、結合の最適化を行っている間に詳しい情報を提供するためです。

以下の例を参照してください。

```
SELECT * FROM EMPLOYEE, EMP_ACT
  WHERE EMPLOYEE.EMPNO = EMP_ACT.EMPNO
    AND EMPLOYEE.EMPNO = '000010'
```

最適化プログラムは、照会が次のようになるように変更します。

```
SELECT * FROM EMPLOYEE, EMP_ACT
  WHERE EMPLOYEE.EMPNO = EMP_ACT.EMPNO
    AND EMPLOYEE.EMPNO = '000010'    AND EMP_ACT.EMPNO  = '000010'
```

次の規則は、どの述部をその他の結合ダイヤルに追加するかを決定します。

- 影響されるダイヤルは、等号の結合演算子を備えていなければなりません。
- 述部は**分離可能**です。これは、この述部からの偽条件が行を除外することを意味します。
- 述部のオペランドの 1 つは、等号結合列で、もう 1 つは定数またはホスト変数です。
- 述部演算子は、LIKE ではありません (OPNQRYF %WLDCRD、または *CT)。
- 述部は OR により他の述部に接続されていません。

Query 最適化プログラムは、新規の述部を生成します。述部がすでに WHERE 文節 (OPNQRYF QRYSLT パラメーター) にあるかは関係ありません。

一部の述部は冗長です。これは、照会の中の他の述部に関する以前の評価が、述部が提供する結果をすでに判別しているときに起こります。冗長な述部はユーザーが指定するか、または述部操作時に Query 最適化プログラムにより生成されます。 =、>、>=、<、<=、または BETWEEN (OPNQRYF *EQ、 *GT、 *GE、 *LT、 *LE、または %RANGE) の述部演算子を持つ冗長な述部は、単一述部に組み込まれ、ほとんどの選択的な範囲に反映されます。

ルック・アヘッド述部生成 (LPG)

ルック・アヘッド述部生成 (LPG) と呼ばれる推移的閉包の特別なタイプは、結合でコストがかかる場合があります。この場合、最適化プログラムは照会の結果を大きなファクト・テーブルに事前に適用して、結合のランダム入出力コストを最小化しようとします。一般に LPG は、星形結合照会と呼ばれる照会のクラスで使用されますが、任意の結合照会でおそらく使用可能です。

次の照会を参照してください。

```
SELECT * FROM EMPLOYEE,EMP_ACT
 WHERE EMPLOYEE.EMPNO = EMP_ACT.EMPNO
 AND EMPLOYEE.EMPNO ='000010'
```

最適化プログラムは、照会が次のようになるように内部で変更することに決めるかもしれません。

```
WITH HT AS (SELECT *
  FROM EMPLOYEE
  WHERE EMPLOYEE.EMPNO='000010')

SELECT *
 FROM HT, EMP_ACT
 WHERE HT.EMPNO = EMP_ACT.EMPNO
 AND EMP_ACT.EMPNO IN (SELECT DISTINCT EMPNO
    FROM HT)
```

最適化プログラムは、「副照会」の結果を一時ハッシュ・テーブルに配置します。副照会のハッシュ・テーブルは、EMP_ACT (ファクト) テーブルに対して 2 つのメソッドのうちのいずれかで適用できます。

- ハッシュ・テーブルの特殊値が検索されます。それぞれの特殊値ごとに、その値に対して戻されるレコードを判別するために EMP_ACT に対する索引が厳密に調べられます。その後、それらのレコード ID は通常、保管され、ソートされます (予期されるレコード ID の総数によってはソートが省略される場合もあります)。 ID が決定されたら、従来のネストされたループ結合処理よりもずっと効率的な方法で EMP_ACT レコードのそれらのサブセットにアクセスできるようになります。
- EMP_ACT を走査できます。それぞれのレコードごとに、レコードがすべて EMPLOYEE と結合するかどうかを見るためにハッシュ・テーブルが厳密に調べられます。これにより、従来のネストされたループ結合処理より効率的なレコード・リジェクト・メソッドを使用してより効率的に EMP_ACT にアクセスできるようになります。

注: LPG 処理は SQL Query Engine における通常の処理の一部です。 Classic Query Engine は最初のメソッドのみを考慮し、EVI による問題の索引が必要となり、 STAR_JOIN オプションおよび FORCE_JOIN_ORDER QAQQINI オプションの使用も必要となります。

関連資料

136 ページの『照会オプション・ファイル QAQQINI による照会の動的な制御』
照会オプション・ファイル QAQQINI サポートには、照会属性の変更 (CHGQRYA) コマンドおよび QAQQINI ファイルを通して照会が実行される環境を動的に変更または上書きする機能があります。照会オプション・ファイル QAQQINI を使用して、データベース・マネージャーによって使用される属性を設定します。

2 つ以上のテーブルからデータを選択する場合のパフォーマンスの向上のためのヒント

以下に示すのは、CQE にのみ、さらに、複数のテーブルにアクセスする選択ステートメントに対し、特に適用される提案です。 2 つ以上のテーブルを結合する際に、結合列に関する余分な情報を追加したい場合があります。 CQE 最適化プログラムは、2 つの列間で推移的閉包述部を生成しません。結合を要求するとき追加情報を最適化プログラムに与えると、結合を行う最善の方法を判別できます。この追加情報は余分のように見えますが、最適化プログラムにとって役に立つ情報です。

考慮の対象にしている選択ステートメントが複数のテーブルをアクセスする場合は、 171 ページの『索引方針の作成』で示したすべての推奨事項があてはまります。たとえば、次のようなコーディングをする代わりに、

```
EXEC SQL
  DECLARE EMPACTDATA CURSOR FOR
  SELECT LASTNAME, DEPTNAME, PROJNO, ACTNO
       FROM CORPDATA.DEPARTMENT, CORPDATA.EMPLOYEE,
                                 CORPDATA.EMP_ACT
      WHERE DEPARTMENT.MGRNO = EMPLOYEE.EMPNO
            AND EMPLOYEE.EMPNO = EMP_ACT.EMPNO
END-EXEC.
```

このように、最適化プログラムにもう少しデータおよびコードを与えます。

```
EXEC SQL
  DECLARE EMPACTDATA CURSOR FOR
  SELECT LASTNAME, DEPTNAME, PROJNO, ACTNO
       FROM CORPDATA.DEPARTMENT, CORPDATA.EMPLOYEE,
                                 CORPDATA.EMP_ACT
       WHERE DEPARTMENT.MGRNO = EMPLOYEE.EMPNO
             AND EMPLOYEE.EMPNO = EMP_ACT.EMPNO
             AND DEPARTMENT.MGRNO = EMP_ACT.EMPNO
END-EXEC.
```

照会の複数の結合タイプ

複数の結合タイプ (内部、左辺外部、右辺外部、左辺例外、および右辺例外) は、JOIN 構文を使用して照会の中で指定できますが、DB2 for i5/OS は、全体の照会について、内部、左辺外部、あるいは左辺例外結合タイプのうちの、1 つの結合タイプしかサポートできません。このため、最適化プログラムは、照会についての全体の結合タイプをどれにすべきかを決定し、正しいセマンティクスをアーカイブするためにファイルを再配列する必要があります。

注: このセクションは、SQE または OPNQRYF には適用されません。

最適化プログラムは、各ダイヤルおよび全体の照会についての結合タイプを判別するために指定できる行選択とともに結合基準を評価します。この情報が知られると、最適化プログラムはテーブルの相対行番号を使用して追加の選択を生成し、照会内で発生することのある異なるタイプの結合をシミュレートします。

左辺外部結合または例外結合で一致しない行についてヌル値が返されるので、WHERE 文節で指定されることのある追加の結合基準を含めて、そのダイヤル用に指定された分離可能な選択は、(選択が IS NULL 述部についてでない限り) 一致しない行をすべて除去させることになります。これによって、IS NULL 述部が指定される場合は、そのダイヤルについての結合タイプが内部結合 (または例外結合) に変更されることになります。

次の例では、左辺外部結合が EMPLOYEE および DEPARTMENT のテーブルの間で指定されます。WHERE 文節では、DEPARTMENT テーブルにも適用される 2 つの選択述部があります。

```
SELECT EMPNO, LASTNAME, DEPTNAME, PROJNO
 FROM CORPDATA.EMPLOYEE XXX LEFT OUTER JOIN CORPDATA.DEPARTMENT YYY
        ON XXX.WORKDEPT = YYY.DEPTNO
      LEFT OUTER JOIN CORPDATA.PROJECT ZZZ
        ON XXX.EMPNO = ZZZ.RESPEMP
 WHERE XXX.EMPNO = YYY.MGRNO AND
        YYY.DEPTNO IN ('A00', 'D01', 'D11', 'D21', 'E11')
```

最初の選択述部である XXX.EMPNO = YYY.MGRNO は、結合基準に追加され、「内部結合」結合条件として評価される追加の結合条件です。 2 番目の選択述部は、一致しない行があるとそれを除去する分離可能な選択述部です。これらの選択述部のどちらも、DEPARTMENT テーブルについての結合タイプを左辺外部結合から内部結合に変更させることになります。

EMPLOYEE および DEPARTMENT テーブルの間の結合は内部結合に変更されたとはいえ、全体の照会は、PROJECT テーブルについての結合条件を満足するためにまだ左辺外部結合を残す必要があります。

注: 複数の結合タイプを指定するときは、それらが一致しない行についての照会への追加選択によってサポートされているので、注意を払う必要があります。これは、結果として得られる結合基準を満足する行数が、その個別のダイヤルの結合タイプに基づいて不一致行を抜粋または除外する選択が適用される前に、きわめて大きくなることがあることを意味します。

結合照会パフォーマンスの問題の原因

上記で説明した最適化アルゴリズムは、ほとんどの結合照会のパフォーマンスに貢献しますが、いくつかの照会のパフォーマンスを低下させる可能性があります。

パフォーマンスの低下は、次の場合に起こります。

- これから作成される結合列に重複値統計の平均数を提供する索引を使用できない場合。
- 索引または列統計が選択列に存在しないために、Query 最適化プログラムが、ローカル選択をテーブルに適用するときに選択する行数を見積もるためにデフォルトのフィルター係数を使用する場合。

  選択列に索引を作成すると、Query 最適化プログラムはキー範囲の見積もりを使用することにより、より正確なフィルター操作の見積もりを行うことができます。
- 結合列用に選択した特定値が、テーブル内の結合列の値すべてに関する重複値の平均数よりもはるかに多い一致行数を示している (たとえば、データが均一に分散されていない) 場合。

結合照会のパフォーマンスを改善するためのヒント

パフォーマンスのよくない結合照会があったり、結合照会を使用する新規のアプリケーションを作成しようとしている場合は、これらのヒントが役立つことになります。

表 25. 結合照会を使用するアプリケーションを作成するためのチェックリスト

チェック内容	効用
データベース設計のチェック。結合列のすべて、行選択列のすべて、またはその両方のすべてについて、使用できる索引があることを確認してください。最適化プログラムは、この処理を支援するために索引推奨を随所に提供します。 System i ナビゲーター -「データベース」の下の索引アドバイザー、Visual Explain の下の推奨情報、またはデータベース・モニターの 3020 レコード内の推奨情報のいずれかを使用してください。	これにより、Query 最適化プログラムは、重複値の平均数を判別できるので、効率の良いアクセス方式を選択できるようになります。多数の照会が照会を実施するために既存の索引を使用できるようになり、一時索引またはハッシュ・テーブルの作成コストがかからないようにすることができます。
照会をチェックして、複合述部の一部を他のダイヤルに追加して、最適化プログラムの各ダイヤルの選択性に関して、よりよい方法がないかどうかを調べてください。	Query 最適化プログラムは、OR で接続している述部または分離できない述部、または LIKE の述部演算子に述部を追加しないので、これらの述部を追加して照会を修正するとよいでしょう。
ALWCPYDTA(*OPTIMIZE) または ALWCPYDTA(*YES) を指定してください。	照会が一時索引またはハッシュ・テーブルを作成している場合で、最適化プログラムが既存の索引またはハッシュ・テーブルだけを使用した方が処理時間が短縮されると思われる場合は、ALWCPYDTA(*YES) を指定してください。 照会が一時索引またはハッシュ・テーブルを作成しておらず、一時索引を作成した方が処理時間が短縮されると思われる場合は、ALWCPYDTA(*OPTIMIZE) を指定してください。 代替として、OPTIMIZE FOR n ROWS を指定して、最適化プログラムにアプリケーションが作成されるすべての行を読み取ろうとしていることを通知します。それには、n に大きな数を設定してください。照会が終了する前に n を小さな数に設定することもできます。

表 25. 結合照会を使用するアプリケーションを作成するためのチェックリスト (続き)

チェック内容	効用
OPNQRYF の場合、OPTIMIZE(*FIRSTIO) または OPTIMIZE(*ALLIO) を指定します。	OPTIMIZE(*FIRSTIO) または OPTIMIZE(*ALLIO) オプションを指定し、アプリケーションに正確に反映させます。最適化プログラムが照会を最適化して、行の最初のブロックを最も効率的に検索するようにしたい場合には、*FIRSTIO を使用します。これにより、既存のオブジェクトを使用して最適化プログラムの方向が偏ることになります。応答セット全体の検索時間を最適化したい場合は、*ALLIO を使用してください。そうすると、最適化プログラムは一時索引またはハッシュ・テーブルなどの一時オブジェクトを作成して、入出力を最小に抑えることができます。

星形結合照会	1 つのテーブルがすべての 2 次テーブルに連続して結合されている結合は、**星形結合 (star join)** と呼ばれます。すべての 2 次結合述部に特定のテーブルへの列参照が含まれている星形結合の場合には、そのテーブルを結合位置 1 に置くことによってパフォーマンス上有利になる可能性があります。例 A では、すべてのテーブルがテーブル EMPLOYEE に結合されています。 Query 最適化プログラムは、自由に結合順序を決定することができます。 SQE の場合、最適化プログラムはルック・アヘッド述部生成を使用して最適の結合順序付けを決定します。CQE の場合には、FORCE_JOIN_ORDER パラメーターが *YES である照会オプション・ファイル (QAQQINI) を使用して、EMPLOYEE を強制的に変更する必要があります。この例の結合タイプは、戻されるデフォルト値がない結合 (つまり内部結合) であることに注意してください。このテーブルを先頭位置に強制的に置く理由は、ランダム入出力処理を避けるためです。 EMPLOYEE が結合位置 1 にない場合、EMPLOYEE の行は、結合処理中、繰り返し検査されることになります。EMPLOYEE がある程度の大きさの場合、かなりのランダム入出力処理が行われることになり、結果的にパフォーマンスの低下をもたらします。 EMPLOYEE を先頭位置に強制的に置くことによって、ランダム入出力処理を最小にすることができます。
	
	例 A: 星形結合照会
	`DECLARE C1 CURSOR FOR`
	`  SELECT * FROM DEPARTMENT, EMP_ACT, EMPLOYEE,`
	`  PROJECT`
	`  WHERE DEPARTMENT.DEPTNO=EMPLOYEE.WORKDEPT`
	`  AND EMP_ACT.EMPNO=EMPLOYEE.EMPNO`
	`  AND EMPLOYEE.WORKDEPT=PROJECT.DEPTNO`
	
	例 B: FORCE_JOIN_ORDER によって強制された順序による星形結合照会
	`  DECLARE C1 CURSOR FOR`
	`  SELECT * FROM EMPLOYEE, DEPARTMENT, EMP_ACT,`
	`  PROJECT`
	`  WHERE DEPARTMENT.DEPTNO=EMPLOYEE.WORKDEPT`
	`  AND EMP_ACT.EMPNO=EMPLOYEE.EMPNO`
	`  AND EMPLOYEE.WORKDEPT=PROJECT.DEPTNO`

ALWCPYDTA(*OPTIMIZE) を指定して、Query 最適化プログラムが分類ルーチンを使用できるようにしてください。	順序付けが指定してあり、しかもすべてのキー列が単一ダイヤルからきたものである場合は、Query 最適化プログラムは、これにより、可能なすべての結合順序を検討することができます。
結合述部を指定して、あるテーブルのすべての行が他のテーブルの各行と結合されないようにすることができます。	この結果、結合の多分岐が削減されることになり、パフォーマンスが向上します。すべての 2 次テーブルには、「結合先」列としてその列の 1 つを参照する、最小限 1 つの結合述部が必要です。

DISTINCT の最適化

DISTINCT は、ある特定の値を他の値と比較するために使用されます。

SQL で特殊値を戻す照会を作成する方法は 2 つあります。 1 つの方法では次のように DISTINCT キーワードを使用します。

```
SELECT DISTINCT COL1, COL2
 FROM TABLE1
```

2 つ目の方法では次のように GROUP BY を使用します。

```
SELECT COL1, COL2
 FROM TABLE1
 GROUP BY COL1, COL2
```

DISTINCT を含み、SQE を使って実行される照会はすべて、GROUP BY を使用する照会に書き換えられます。この書き換えにより、照会は DISTINCT を使用して、最適化プログラムで使用可能な多くのグループ化技法を利用できるようになります。

DISTINCT からグループ化へのインプリメンテーション

以下は、DISTINCT を使用した照会の例です。

```
SELECT DISTINCT COL1, COL2
 FROM T1
 WHERE COL2 > 5 AND COL3 = 2
```

最適化プログラムは、これを次のような照会に書き換えます。

```
SELECT COL1, COL2
 FROM T1
 WHERE COL2 > 5 AND COL3 = 2
 GROUP BY COL1, COL2
```

DISTINCT の除去

ファイル全体に DISTINCT を含む照会の集約 (グループ化または選択ではない) では、DISTINCT を除去することができます。たとえば、DISTINCT を持つ次の照会をご覧ください。

```
SELECT DISTINCT COUNT(C1), SUM(C1)
 FROM TABLE1
```

最適化プログラムはこの照会を次のように書き換えます。

```
SELECT COUNT(C1), SUM(C1)
 FROM TABLE1
```

DISTINCT フィールドと GROUP BY フィールドが同一の場合、DISTINCT は除去可能です。 DISTINCT フィールドが GROUP BY フィールドのサブセットである場合 (そして集約が存在しない場合)、DISTINCT は除去可能です。

最適化のグループ化

DB2 for i5/OS には、最適化プログラムがグループ化に直面したときに使用するための手法があります。Query 最適化プログラムは、その手法を選択して照会を最適化します。

グループ化ハッシュの実施

この技法は標準ハッシュ・アクセス方式を使用して、選択されたテーブル行のグループ化または要約を行います。抜粋された各行には、行で指定されたグループ値がハッシュ関数を通して実行されます。計算されたハッシュ値とグループ値はグループ値を示しているハッシュ・テーブルの中の項目をより早く検出するために使用されます。

現行のグループ値に常にハッシュ・テーブル内の行がある場合には、ハッシュ・テーブル項目は要求されたグループ化する列の演算子 (SUM または COUNT のような) に基づいて、現行のテーブル行値を用いて取り出され、要約 (更新) されます。ハッシュ・テーブル項目が現行のグループ値で見つからない場合には、新しい項目がハッシュ・テーブルに挿入され、現行のグループ値で初期設定されます。

しかし、この方法でグループ化の結果を取り出すためには、まず最初にハッシュ・テーブルを作成し構築する必要があるため、他のグループ化の方法よりも必要となる時間が増加することがあります。いったんハッシュ・テーブルが完全に構築されると、データベース管理者はこのテーブルを使用してグループ化の結果を返し始めます。どの結果を返す場合もその前に、データベース・マネージャーはハッシュ・テーブルの中の要約項目に対して、指定されたグループ化選択基準あるいは順序付けを処理しなければなりません。

グループ化ハッシュ方式が最も効果的な場合

グループ化ハッシュ方式は、統合率が高いほど最も有効です。**統合率**とは、計算されたグループ化の結果に対して選択されたテーブル行の比率です。データベース・テーブル行ごとにそれ自身の固有なグループ化の値がある場合には、ハッシュ・テーブルは非常に大きくなります。入れ代わる際に、これはハッシュ・アクセス方式を遅くします。

最適化プログラムは、指定されたグループ化する列の固有な値の数 (すなわち、データベース・テーブルのグループの予期される数) を最初に判断して統合率を見積もります。最適化プログラムは、次にテーブル内の合計行数および指定された選択基準を調べ、この調査結果を使用して統合率を見積もります。

グループ化する列についての索引は、最適化プログラムが率をより正確に見積もるのに役立ちます。索引はキー列についての重複値の平均数を統計の中に持っているので、正確度を向上させます。

最適化プログラムは、予期されるグループの見積数を使用して、ハッシュ・テーブルの区画数の計算もします。すなわち、ハッシュ・アクセス方式はハッシュ・テーブルが均整がとれていればさらに有効です。ハッシュ・テーブルの区画の数は、ハッシュ・テーブルとこの分配の均一性にわたってどのように項目が分散されるかに直接影響します。

ハッシュ関数はグループ化する列が、整数 (2 進数) データ・タイプの例外を除いた非数値データ・タイプを持つ列から成る場合には、より速く実行できます。さらに、可変長やヌル列属性に関係のない列をグループ化する値の列に指定することによって、ハッシュ関数はさらに有効に実行することができます。

索引のグループ化の実施

索引経由のグループ化は、順序付けグループ化、事前要約処理という 2 つの基本的な方法で実施できます。

順序付けグループ化

この方法は、基数索引走査アクセス方式または基数索引プローブ・アクセス方式を使用してグループ化を実行します。索引は、隣接する左端のキー列としてグループ化する列がすべて含まれることが必要となります。データベース・マネージャーは索引を介して個々のグループをアクセスし、要求された集計機能を実行します。

定義によって、索引にはグループ化されるキー値のすべてが常に一緒にあるので、最初のグループ化の結果はハッシュ方式よりも短時間で返されます。これは、ハッシュ方式が一時結果を要求するためです。この方法は、アプリケーションがグループ結果のすべての取り出しを必要としない場合、あるいはグループ化列に一致する索引が常に存在している場合に効果的です。

グループ化が索引を使用して実行され、グループ化列を満たす永続索引がまだ存在しないときに、一時索引が作成されます。照会内に指定されたグループ化列は索引のキー列として使用されます。

事前要約処理

この SQE 専用のインプリメンテーションは、コード化ベクトル索引を使用して、すでに索引の記号テーブルにある要約情報を抽出します。 EVI の記号テーブルの部分には、キーの固有値と、その固有値を持つテーブル・レコードの数値のカウントが含まれ、基本的に、索引キーの列のグループ化はすでに実行されています。照会が単一のテーブルを参照し、単純な集約を実行する場合、グループ化の結果に迅速にアクセスするために EVI を使用することができます。たとえば、次の照会を検討してください。

```
SELECT COUNT(*), col1
 FROM t1
 GROUP BY col1
```

t1 に対してcol1 をキーとする EVI が存在する場合、EVI 記号テーブル内の事前計算済みグループ化応答にアクセスするよう、最適化プログラムによって照会が作成し直されることがあります。これにより、テーブルのレコードの数値が大きく、結果のグループの数値が小さい場合 (テーブルのサイズに対して)、照会は劇的に向上します。参照列が EVI のキー定義にあれば、このメソッドは選択 (WHERE 文節) でも可能です。たとえば、次の照会を検討してください。

```
SELECT COUNT(*), col1
 FROM t1
 WHERE col1 > 100
 GROUP BY col1
```

EVI を利用するために、最適化プログラムはこの照会を再作成することができます。この事前要約処理は DISTINCT 処理、GROUP BY、および列関数 COUNT に対して機能します。照会で参照される表の列もすべて EVI のキー定義になければなりません。それで、たとえば次の照会を行い、EVI を使用できるとします。

```
SELECT DISTINCT col1
 FROM t1
```

しかし、この照会では以下は不可です。

```
SELECT DISTINCT col1
 FROM t1
 WHERE col2 > 1
```

この照会が EVI を使用できないのは、それが、EVI のキー定義にない、表の col2 を参照するからです。EVI キーで複数の列、たとえば col1 と col2 が定義されている場合、キーの一番左にある列を使用しなければならないということにもご注意ください。たとえば、(col1, col2) というキー定義を持つ EVI が存在し、照会が col2 のみを参照する場合、EVI が使用されることはまずありません。

グループ化列を除去することによるグループ化の最適化

すべてのグループ化列は、グループ化の列のリストから除去できるかを判別するために評価されます。等号演算子が指定された分離可能選択述部を持つこれらのグループ化列のみが、検討されます。これにより列が単一の値だけに一致でき、固有のグループの限定に役立たないことを保証します。

この処理が行われるのは、最適化プログラムが一時索引やハッシュ・テーブルに対して、照会を実行するためにより多くの索引を検討し、キー列として追加されている列数を減少させることができるようにするためです。

次の例は、最適化プログラムがグループ化の列を除去できる可能性のある照会を示しています。

```
DECLARE DEPTEMP CURSOR FOR
  SELECT EMPNO, LASTNAME, WORKDEPT
    FROM CORPDATA.EMPLOYEE
    WHERE EMPNO = '000190'
    GROUP BY EMPNO, LASTNAME, WORKDEPT
```

OPNQRYF の例:

```
OPNQRYF FILE(EMPLOYEE) FORMAT(FORMAT1)
   QRYSLT('EMPNO *EQ ''000190''')
   GRPFLD(EMPNO LASTNAME WORKDEPT)
```

この例では、選択述部が EMPNO = '000190' なので、最適化プログラムはグループ化列のリストから EMPNO を取り除くことができます。 LASTNAME および WORKDEPT のみをキー列として指定された索引は照会の実行に考慮され、一時索引またはハッシュが要求された場合に、EMPNO が使用されないことになります。

注: EMPNO をグループ化する列のリストから取り除くことができますが、グループ化する列を 3 つすべてもつ永続索引が存在する場合には、最適化プログラムはその索引を使用する選択を取り続けます。

付加グループ化列を追加することによるグループ化の最適化

グループ化列の除去の場合に用いた同じ論理を使用して照会に付加グループ化列を追加することもできます。これは、索引がグループ化の実行に使用できるか判別しようとする場合にのみ実行されます。

次の例は、最適化プログラムが付加グループ化列を追加できる可能性のある照会を示しています。

```
CREATE INDEX X1 ON EMPLOYEE
      (LASTNAME, EMPNO, WORKDEPT)

DECLARE DEPTEMP CURSOR FOR
  SELECT LASTNAME, WORKDEPT
    FROM CORPDATA.EMPLOYEE
    WHERE EMPNO = '000190'
    GROUP BY LASTNAME, WORKDEPT
```

この照会要求については、最適化プログラムがこの照会に X1 が考慮されたときに、EMPNO を追加のグループ化列として追加することができます。

索引スキップ・キー処理の使用によるグループ化の最適化

既存の索引を使用したキー順による使用法のアルゴリズムを使用してグループ化を行う場合、索引スキップ・キー処理を使用することができます。これは、各グループのすべてのレコードではなく、各グループの非常に少数のレコードを処理する配列されたグループ化の特殊なバージョンです。

索引スキップ・キー処理アルゴリズムは、次のとおりです。

1. 索引を使用してグループに位置決めをし、
2. そのグループの選択基準に一致する最初の行を検出し、最初の非ヌルの MIN 値または MAX 値がそのグループに指定されていた場合、
3. そのグループをユーザーに戻し
4. 次のグループに「スキップ」して、処理を繰り返す。

このアルゴリズムでは、あるグループに関してすべての索引キー値を処理しないこともあるため、パフォーマンスが向上します。

索引スキップ・キー処理は、次のように使用することができます。

- キー順グループ化の使用法を用いた単一テーブル照会については、次の場合に使用できます。
  - 照会に列関数が 1 つも含まれないか、または
  - 照会に単一の MIN または MAX 列関数だけがあり、MIN または MAX のオペランドがグループ化列の後の索引内の次のキー列である場合。照会内にはその他のグループ化関数を含めることはできません。 MIN 関数の場合、キー列は、昇順キーでなければなりません。MAX 関数の場合、キー列は、降順キーでなければなりません。照会がテーブル全体のグループ化の場合には、MIN または MAX のオペランドは、先頭キー列でなければなりません。

    例 1 で SQL を使用する場合、次のようにします。

    ```
    CREATE INDEX IX1 ON EMPLOYEE (SALARY DESC)

    DECLARE C1 CURSOR FOR
       SELECT MAX(SALARY) FROM EMPLOYEE;
    ```

    Query 最適化プログラムは、索引 IX1 の使用を選択します。 SLIC 実行時コードは、SALARY の最初の非ヌル値を検出するまで走査します。 SALARY が非ヌルであると想定した場合、実行時コードは、先頭索引キーに位置決めし、そのキー値を SALARY の MAX として戻します。これ以降、索引キーの処理は行われません。

    例 2 で SQL を使用する場合、次のようにします。

    ```
    CREATE INDEX IX2 ON EMPLOYEE (WORKDEPT, JOB, SALARY)

    DECLARE C1 CURSOR FOR
     SELECT WORKDEPT, MIN(SALARY)
      FROM EMPLOYEE
      WHERE JOB='CLERK'
      GROUP BY WORKDEPT
    ```

    Query 最適化プログラムは、索引 IX2 の使用を選択します。 データベース・マネージャーは、JOB が 'CLERK' に等しい DEPT の最初のグループに位置決めします。このコードはこの後 JOB が 'CLERK' に等しい次の DEPT グループにスキップします。
- 結合照会の場合:
  - すべてのグループ化列は、1 つの単一テーブルからのものでなければなりません。
  - 各ダイヤルについて、ダイヤルを参照する MIN または MAX 列関数オペランドは、最大 1 つであり、照会の中に、他の列関数を含めることはできません。
  - MIN または MAX 関数オペランドが、グループ化列と同じダイヤルからの場合には、単一テーブル照会と同じ規則を使用することになります。
  - MIN または MAX 関数オペランドが別のダイヤルからの場合には、そのダイヤルの結合列は、グループ化列の 1 つと結合する必要があり、そのダイヤル用の索引には、結合列とその後に MIN または MAX オペランドが続いて入っている必要があります。

    例 1 で SQL を使用する場合、次のようにします。

    ```
    CREATE INDEX IX1 ON DEPARTMENT(DEPTNAME)

    CREATE INDEX IX2 ON EMPLOYEE(WORKDEPT, SALARY)

    DECLARE C1 CURSOR FOR
    ```

```
SELECT DEPARTMENT.DEPTNO, MIN(SALARY)
   FROM DEPARTMENT, EMPLOYEE
   WHERE DEPARTMENT.DEPTNO=EMPLOYEE.WORKDEPT
   GROUP BY DEPARTMENT.DEPTNO;
```

読み取りトリガーを除去することによるグループ化の最適化

読み取りトリガーがある物理ファイルまたはテーブルが関係した照会の場合には、トリガーごとのグループが、処理ごとのグループより前に一時ファイルを必要とするため、これらの照会はスローダウンします。

注: TRGTIME (*AFTER) および TRGEVENT (*READ) と共にテーブルで 物理ファイル・トリガー追加 (ADDPFTRG) コマンドが使用されると、読み取りトリガーが追加されます。

照会の速度は速くなり、読み取りトリガーは除去されます (RMVPFTRG TRGTIME (*AFTER) TRGEVENT (*READ))。

関連情報

物理ファイル・トリガーの追加 (ADDPFTRG) コマンド

セット最適化のグループ化

最適化プログラムは、照会で指定された個別のグループ化セットに対して前述のすべてのグループ化の最適化を利用します。

すべてのグループ化セットを実装するために複数の一時結果セットが必要な場合、1 つのパススルー・データを使用してこれらをすべて取り込むことができます。異なるタイプの一時結果セットが各種のグループ化セットの実装に使用された場合でも、これは当てはまります。ソート済みの異なるリストと呼ばれる新しい一時結果タイプがあります。これは、特に ROLLUP 実装用に使用されます。この一時結果セットは、集約行 (つまり、ROLLUP 文節にリストされるすべての式を含むグループ化セット) を計算するために使用されます。ハッシュ・グループ化は、現行のグループ化値を迅速に検出するために内部的に使用されます。一時結果セットの項目もソートされ、これによって、集約結果は、追加の一時結果セットを作成することなく、ロールアップ結果セットの超集約行の計算に使用することができます。また、ROLLUP は、一時結果セットを作成することなく、ロールアップの列上の基数索引を使用して実装することもできます。

最適化プログラムは、最大 1 つの一時結果セットで使用することで所定の ROLLUP のすべてのグループ化セットを計算できるので、最適化プログラムによりグループ化セット照会のロールアップ・パターンを検索する方が有利です。 Dbop は、個別のグループ化セットのリストで ROLLUP パターンを検出しようと試みます。例えば、次の GROUP BY 文節

```
GROUP BY GROUPING SETS ((A, B, C), (B, D), (A, B), (A), ())
```

は、次のように書き直されます。

```
GROUP BY GROUPING SETS ((ROLLUP(A, B, C)), (B, D))
```

これによって、4 つではなく、最大で 2 つの一時結果セットを使用して照会を実装することができます。

CUBE 文節を含む照会は、ROLLUP' およびグループ化セットの共用体に分類されます。たとえば、次のとおりです。

```
CUBE(A, B, C)
```

は、次と等価です。

```
(ROLLUP(A, B, C)), (ROLLUP'(B, C)), (ROLLUP'(C, A))
```

ROLLUP' の表記は、結果セットの総合計行を含まない ROLLUP 操作の内部表記です。したがって、ROLLUP'(B, C) は、GROUP BY GROUPING SETS ((B,C), (B)) と等価です。この CUBE の書き直しによ

って、照会が書き直されていなかったときに必要となる場合がある 8 つの一時結果セットではなく、最大で 3 つの一時結果セットとともに照会を実装することができます。

順序付けの最適化

このセクションでは、DB2 for i5/OS がどのようにして順序付けの方法を実施するかについて述べるとともに、 Query 最適化プログラムが行う最適化選択方法について説明します。 Query 最適化プログラムは、索引順序付けまたは分類のいずれかを使用して、順序付けを実行することができます。

ソート順序付けの実行

ソート・アルゴリズムは、行をソート・スペースに読み込んで、指定された順序付けキーに基づいて行をソートします。その後、行は順序付けソート・スペースからユーザーに戻されます。

索引順序付けの実行

索引順序付けの実行は、隣接する左端のキー列として順序付けする列がすべて含まれる索引を必要とします。データベース・マネージャーは、索引を介して索引順に個々の行をアクセスします。この結果、行が要求元に戻されることになります。

この方法は、アプリケーションが順序付けの結果のすべての取り出しを必要としない場合、あるいは順序付け列に一致する索引が既に存在している場合に効果的です。順序付けが索引を使用して実行され、順序付け列を満たす永続索引がまだ存在しないときに、一時索引が作成されます。照会内に指定された順序付け列は索引のキー列として使用されます。

順序付け列を除去することによる順序付けの最適化

すべての順序付け列は、順序付けの列のリストから除去できるかを判別するために評価されます。 等号演算子が指定された分離可能選択述部を持つこれらの順序付け列のみが、検討されます。 これにより列が単一の値だけに一致でき、順序付けの限定に役立たないことを保証します。

この処理が行われるのは、最適化プログラムが一時索引に対して、照会を実行する際のより多くの索引を検討し、キー列として追加されている列数を減少させることができるようにするためです。 次の SQL 例は、最適化プログラムが順序付けの列を除去できる可能性のある照会を示しています。

```
DECLARE DEPTEMP CURSOR FOR
  SELECT EMPNO, LASTNAME, WORKDEPT
  FROM CORPDATA.EMPLOYEE
  WHERE EMPNO = '000190'
  ORDER BY EMPNO, LASTNAME, WORKDEPT
```

付加順序付け列を追加することによる順序付けの最適化

順序付け列の除去の場合に適用されるのと同じロジックが、照会に付加グループ化列を追加するためにも使用されることがあります。これは、索引が順序付けの実行に使用できるか判別しようとする場合にのみ実行されます。

次の例は、最適化プログラムが付加順序付け列を追加できる可能性のある照会を示しています。

```
CREATE INDEX X1 ON EMPLOYEE (LASTNAME, EMPNO, WORKDEPT)

DECLARE DEPTEMP CURSOR FOR
  SELECT LASTNAME, WORKDEPT
  FROM CORPDATA.EMPLOYEE
  WHERE EMPNO = '000190'
  ORDER BY LASTNAME, WORKDEPT
```

この照会要求については、最適化プログラムがこの照会に X1 が考慮されたときに、EMPNO を追加の順序付け列として追加することができます。

ビューの実行

ビュー、派生テーブル (ネストされたテーブルの式または NTE)、および共通テーブル式 (CTE) は、2 つのメソッドのいずれかを使用して Query 最適化プログラムにより実装されます。

これらのメソッドは次のとおりです。

- 最適化プログラムは、照会選択ステートメントを、ビューの選択ステートメントと結合します。
- 最適化プログラムは、ビューの結果を一時テーブルに置いた後、照会内のビュー参照を一時テーブルで置き換えます。

ビューの複合の実行

ビューの複合の実行は、照会選択ステートメントを、ビューの選択ステートメントと結合して、新規の照会を生成します。すると、新規の結合した選択ステートメント照会は、基礎となる基本テーブルに対して直接実行されます。

この単一の複合ステートメントは、ビューを含む照会の場合に適した実行です。これは、データを一度だけ渡せばよいからです。

以下の例を参照してください。

```
CREATE VIEW  D21EMPL AS
  SELECT * FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT='D21'
```

SQL の使用:

```
  SELECT LASTNAME, FIRSTNME, SALARY
  FROM D21EMPL
  WHERE JOB='CLERK'
```

Query 最適化プログラムは、次の例で示したような新規の照会を生成します。

```
SELECT LASTNAME, FIRSTNME, SALARY
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT='D21' AND JOB='CLERK'
```

照会に含まれるのは、ユーザーの照会で選択された列、照会内で参照される基本テーブル、およびビューとユーザーの照会の両方からの選択です。

注: Query 最適化プログラムが生成する新規の複合照会は、ユーザーには見えません。ユーザーおよびデータベース・パフォーマンス・ツールに見えるのは、ビューに対する元の照会のみです。

ビューの実体化の実行

ビューの実体化の実行は、ビューの照会を実行し、その結果を一時結果に置きます。すると、ユーザーの照会内のビュー参照は、一時に置き換えられ、照会が一時結果に対して実行されます。

ビューの実体化は、ビューの複合を作成できないときにいつでも実行されます。 SQE の場合、ビューの実体化はオプションになることに、注意してください。次のタイプの照会は、ビューの実体化を必要とします。

- ビューの最外部の選択にグループ化が含まれ、照会にグループ化が含まれ、 HAVING または選択リスト内のビューで、列関数から派生した列に参照する場合。

- 照会が結合で、ビューの最外部の選択にグループ化または DISTINCT が含まれる場合。
- ビューの最外部の選択に DISTINCT が含まれ、照会に UNION、グループ化、または DISTINCT が含まれ、次のいずれかがあてはまる場合。
  - 照会だけに共用重み NLSS テーブルがある。
  - ビューだけに共用重み NLSS テーブルがある。
  - 照会とビューの両方に、共用重み NLSS テーブルがあるが、テーブルが異なっている。
- 照会に列関数が含まれ、ビューの最外部の選択に DISTINCT が含まれる場合。
- ビューにアクセス・プランが含まれない場合。これは、ビューがビューを参照し、上記のいずれかの理由でビューの複合が作成できないときに生じることがあります。これはネストされたテーブル式および共通テーブル式には適用しません。
- 共通テーブル式 (CTE) は照会の FROM 文節で複数回参照され、CTE の SELECT 文節は MODIFIES または EXTERNAL ACTION UDF を参照します。

一時結果テーブルが作成されると、 ALWCPYDTA(*OPTIMIZE) で許可されるアクセス方式を使用して、照会を実行できます。これらの方式には、ハッシュ・グループ化、ハッシュ結合、およびビットマップがあります。

以下の例を参照してください。

```
CREATE VIEW AVGSALVW AS
  SELECT WORKDEPT, AVG(SALARY) AS AVGSAL
  FROM CORPDATA.EMPLOYEE
  GROUP BY WORKDEPT
```

SQL 例:

```
  SELECT D.DEPTNAME, A.AVGSAL
  FROM CORPDATA.DEPARTMENT D, AVGSALVW A
  WHERE D.DEPTNO=A.WORKDEPT
```

この場合、結合照会がグループ化ビューを参照するので、ビューの複合は作成できません。 AVGSALVW の結果が一時結果テーブル (*QUERY0001) に置かれます。ビュー参照 AVGSALVW が一時結果テーブルに置き換えられます。それから、新規照会が実行されます。次のような照会が生成されます。

```
SELECT D.DEPTNAME, A.AVGSAL
  FROM CORPDATA.DEPARTMENT D, *QUERY0001 A
  WHERE D.DEPTNO=A.WORKDEPT
```

注: Query 最適化プログラムが生成する新規の照会は、ユーザーには見えません。ユーザーおよびデータベース・パフォーマンス・ツールに見えるのは、ビューに対する元の照会のみです。

可能な場合にはいつでも、副照会述部を除く、照会から分離できる選択が、ビューの実体化処理に追加されます。これにより、より小さな一時結果テーブルが生成され、ビューの実体化の際に既存の索引を使用することができます。このことは、複数の参照が同一のビューまたは照会内の共通テーブル式を参照する場合には、実行されません。次は、分離できる選択をビューの実体化に追加する例です。

```
SELECT D.DEPTNAME,A.AVGSAL
  FROM CORPDATA.DEPARTMENT D, AVGSALVW A
  WHERE D.DEPTNO=A.WORKDEPT AND
  A.WORKDEPT LIKE 'D%' AND AVGSAL>10000
```

照会から分離できる選択がビューに追加される結果、一時結果テーブルを生成する新規照会が作成されます。

```
SELECT WORKDEPT, AVG(SALARY) AS AVGSAL
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT LIKE 'D%'
  GROUP BY WORKDEPT
  HAVING AVG(SALARY)>10000
```

マテリアライズ照会表最適化

マテリアライズ照会表 (MQT) (自動サマリー表またはマテリアライズ・ビューとも呼ばれる) を使用すると、照会のパフォーマンスが向上します。

それは、マテリアライズ照会表に照会結果を事前に計算して格納することによります。データベース・エンジンは、ユーザーの指定する照会でそれを毎回再計算する代わりに、それら事前に計算された結果を使用することができます。 Query 最適化プログラムは、適用できる MQT がないかどうかを調べ、特定の MQT を使用するほうが高速になると判断した場合には、それを使用した照会をインプリメントできます。

マテリアライズ照会表は、SQL の CREATE TABLE ステートメントを使用して作成されます。あるいは、ALTER TABLE ステートメントを使用して既存のテーブルをマテリアライズ照会表に変換することもできます。 MQT 中に格納されている結果を再計算するには、REFRESH TABLE ステートメントを使用します。ユーザーの保守する MQT の場合、INSERT、UPDATE、および DELETE の各ステートメントによってその保守を行えます。

関連情報

Create Table ステートメント

MQT でサポートされる機能

MQT にはほとんどどんな照会でも入れることができますが、最適化プログラムでは、MQT とユーザー指定照会のマッチング時に照会機能のうちの一部しかサポートされていません。 SQE 最適化プログラムでは、ユーザー指定照会と MQT 照会の両方がサポートされていなければなりません。

MQT マッチング・アルゴリズムでサポートされる MQT 照会の機能には、次のものがあります。

- 単一テーブルおよび結合照会
- WHERE 文節
- GROUP BY、およびオプションの HAVING 文節
- ORDER BY
- FETCH FIRST n ROWS
- ビュー、共通テーブル式、およびネストされたテーブル式
- UNION
- パーティション化テーブル

MQT マッチング・アルゴリズムには、以下の限定されたサポートがあります。

- スカラー副選択
- ユーザー定義関数 (UDF) およびユーザー定義テーブル関数
- 再帰的共通表式 (RCTE)
- 以下のスカラー関数:
  - ATAN2
  - DAYNAME
  - DBPARTITIONNAME

- DECRYPT_BIT
- DECRYPT_BINARY
- DECRYPT_CHAR
- DECRYPT_DB
- DIFFERENCE
- DLVALUE
- DLURLPATH
- DLURLPATHONLY
- DLURLSEVER
- DLURLSCHEME
- DLURLCOMPLETE
- ENCRYPT_AES
- ENCRYPT_RC2
- ENCRYPT_TDES
- GENERATE_UNIQUE
- GETHINT
- IDENTITY_VAL_LOCAL
- INSERT
- MONTHNAME
- MONTHS_BETWEEN
- NEXT_DAY
- RAND
- RAISE_ERROR
- REPEAT
- REPLACE
- ROUND_TIMESTAMP
- SOUNDEX
- TIMESTAMP_FORMAT
- TIMESTAMPDIFF
- TRUNC_TIMESTAMP
- VARCHAR_FORMAT
- WEEK_ISO

MQT には、列への参照と列関数だけを含めるようお勧めします。多くの環境において、照会に定数が含まれている場合、それらの定数はパラメーター・マーカーに変換されます。これにより ODP の再利用度はずっと高くなります。 MQT マッチング・アルゴリズムは、MQT の定数と、照会のパラメーター・マークまたはホスト変数値との突き合わせを試みます。しかし、一部の複雑な場合には、このサポートは限定されたものとなり、MQT と照会とのマッチングが行われない場合があります。

関連概念

4 ページの『Query dispatcher』
dispatcher の機能は、照会の属性に応じて、照会の要求を CQE か SQE のどちらかに経路指定します。すべての照会はこのディスパッチャーによって処理され、バイパスすることはできません。

関連資料

75 ページの『MQT マッチング・アルゴリズムに関する詳細』
MQT マッチング・アルゴリズムの一般的な動作は、次のとおりです。

照会最適化時の MQT の使用

MQT を使用する前に、環境属性を検討する必要があります。

最適化において MQT の使用を考慮する際には、環境属性が以下のようになっていることが必要です。

- 照会で ALWCPYDTA(*OPTMIZE) または INSENSITIVE カーソルが指定されていること。
- 照会が SENSITIVE カーソルでないこと。
- MQT で置き換えるテーブルが、その照会で UPDATE または DELETE の可能なものではないこと。
- 現在 MQT の ENABLE QUERY OPTIMIZATION 属性がアクティブであること
- MATERIALIZED_QUERY_TABLE_USAGE QAQQINI オプションが *ALL または *USER に設定され、MQT の使用が有効になっていること。 MATERIALIZED_QUERY_TABLE_USAGE のデフォルト設定値では、MQT の使用が無効です。
- MQT に対する最後の REFRESH TABLE のタイム・スタンプが MATERIALIZED_QUERY_TABLE_REFRESH_AGE QAQQINI オプションで指定されている期間内であること、または *ANY が指定されていること (その場合、最後の REFRESH TABLE に関係なく MQT が考慮されます)。 MATERIALIZED_QUERY_TABLE_REFRESH_AGE のデフォルト設定値では、MQT の使用が無効です。
- 照会が SQE で実行可能であること。
- QAQQINI オプションのうち、IGNORE_LIKE_REDUNDANT_SHIFTS、NORMALIZE_DATA、および VARIABLE_LENGTH_OPTIMIZATION が一致していること。それらのオプションは、マテリアライズ照会表の CREATE 時に格納されるものであり、照会実行時に指定されるオプションと一致していなければなりません。
- MQT のコミット・レベルが照会のコミット・レベル以上であること。 MQT のコミット・レベルは、MQT 照会の中で WITH 文節を使用して指定されます。あるいは、それが指定されていない場合、デフォルト値として MQT 作成時における MQT 実行のコミット・レベルになります。

MQT の例

以下に MQT の使用の例を示します。

例 1

最初の例は、職種が DESIGNER である従業員の情報を戻す照会です。元の照会は、次のようなものです。

```
Q1: SELECT D.deptname, D.location, E.firstnme, E.lastname, E.salary+E.comm+E.bonus as total_sal
       FROM Department D, Employee E
       WHERE D.deptno=E.workdept
       AND E.job = 'DESIGNER'
```

この照会を使用するテーブル MQT1 を作成します。

```
CREATE TABLE MQT1
         AS (SELECT D.deptname, D.location, E.firstnme, E.lastname, E.salary, E.comm, E.bonus, E.job
       FROM Department D, Employee E
```

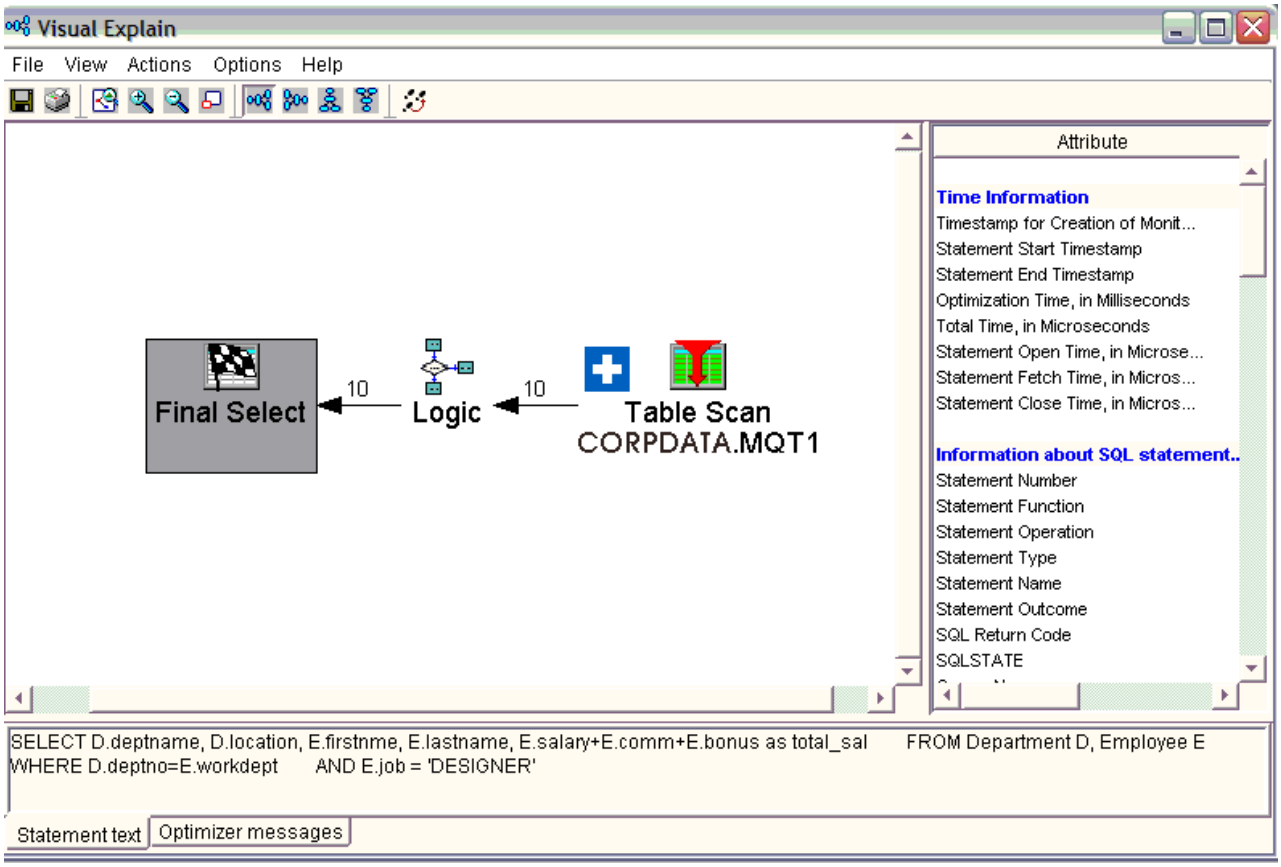
```
      WHERE D.deptno=E.workdept)
 DATA INITIALLY IMMEDIATE  REFRESH DEFERRED
  ENABLE QUERY OPTIMIZATION
  MAINTAINED BY USER
```

指定されたテーブルを MQT で置き換えた結果は、以下に示す新しい照会になります。

```
SELECT M.deptname, M.location, M.firstnme, M.lastname, M.salary+M.comm+M.bonus as total_sal
       FROM MQT1 M
       WHERE M.job = 'DESIGNER'
```

この照会では、MQT によりユーザーの照会の一部のマッチングが実行されます。テーブル DEPARTMENT と EMPLOYEE の代わりに、MQT が FROM 文節に指定されています。 MQT 照会によって実行されない残りの選択操作 (M.job= 'DESIGNER') を実行して余分の行を削除し、結果式 M.salary+M.comm+M.bonus が計算されます。付加的な選択操作を実行するためには、MQT の選択リストに JOB が含まれていなければならないことに注意してください。

MQT を使用する照会の Visual Explain 図:



例 2

'NY' にあるすべての部署の給料 (salary) の合計を求めます。元の照会は、次のようなものです。

```
SELECT D.deptname, sum(E.salary)
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept AND D.location =  'NY'
GROUP BY D.deptname
```

この照会を使用するテーブル MQT2 を作成します。

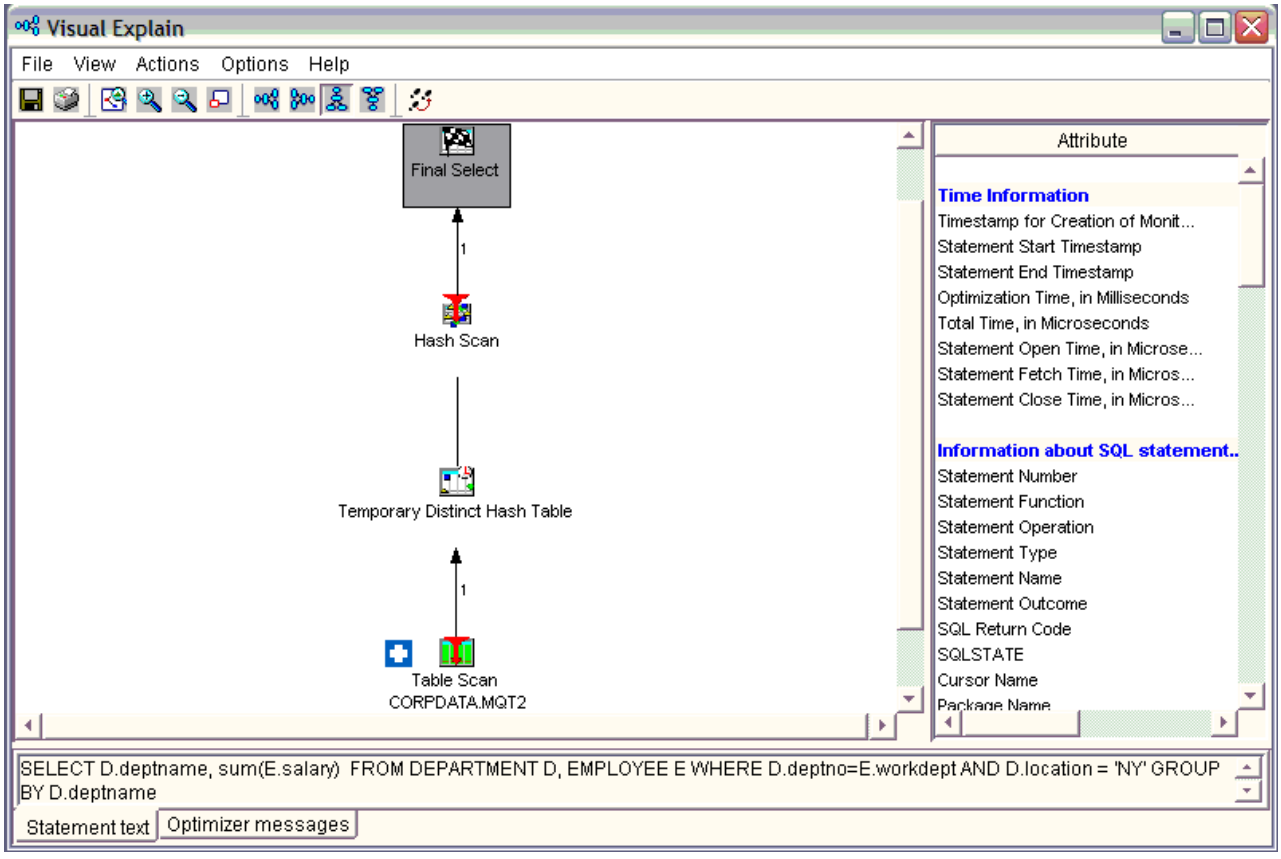
```
CREATE TABLE MQT2
        AS (SELECT D.deptname, D.location, sum(E.salary) as sum_sal
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept
GROUP BY D.Deptname, D.location)
DATA INITIALLY IMMEDIATE  REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION
 MAINTAINED BY USER
```

指定されたテーブルを MQT で置き換えた結果は、以下に示す新しい照会になります。

```
SELECT M.deptname, sum(M.sum_sal)
FROM  MQT2 M
WHERE  M.location = 'NY'
 GROUP BY M.deptname
```

MQT では元の照会より多くのグループが生成されることがあるため、正しい結果を戻すためには、最終結果の照会で再グループ化を実行し、結果の SUM を求めることが必要です。また、新しい照会の一部として M.location='NY' の選択操作が含まれていなければなりません。

MQT を使用する照会の Visual Explain 図:



MQT マッチング・アルゴリズムに関する詳細

MQT マッチング・アルゴリズムの一般的な動作は、次のとおりです。

照会で指定されたテーブルと MQT を調べます。 MQT と照会が同じテーブルを指定する場合、MQT が使用される可能性があり、マッチングは継続します。照会で参照されていないテーブルが MQT で参照されているなら、それが参照保全制約の親テーブルかどうかを判別するためにその参照されていないテーブル

は調べられます。外部キーが NULL 可能ではなく、1 次キーまたは外部キーの等号述部を使って 2 つのテーブルが結合されている場合、MQT を引き続き使用できる可能性があります。

例 3

MQT に含まれるテーブルが照会よりも少ない場合:

```
SELECT D.deptname, p.projname, sum(E.salary)
 FROM DEPARTMENT D, EMPLOYEE E, EMPPROJACT EP,  PROJECT P
 WHERE D.deptno=E.workdept AND E.Empno=ep.empno
  AND ep.projno=p.projno
 GROUP BY D.DEPTNAME, p.projname
```

上記の照会に基づく MQT を作成します。

```
CREATE TABLE MQT3
        AS (SELECT D.deptname,  sum(E.salary) as sum_sal, e.workdept, e.empno
  FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.deptno=E.workdept
 GROUP BY D.Deptname, e.workdept, e.empno)
 DATA INITIALLY IMMEDIATE  REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION
 MAINTAINED BY USER
```

書き直された照会は、次のようになります。

```
SELECT M.deptname, p.projname, SUM(M.sum_sal)
 FROM MQT3 M, EMPPROJACT EP,  PROJECT P
 WHERE M.Empno=ep.empno AND ep.projno=p.projno
 GROUP BY M.deptname, p.projname
```

MQT で指定されている述部は、すべて、照会の中でも指定されていなければなりません。照会にそれ以外の述部が含まれていても問題はありません。しかし、MQT で指定されている述部は、照会内の述部と正確に一致していなければなりません。照会の中で指定されているが MQT では指定されていない述部があるなら、それは MQT の列から派生可能なものでなければなりません。前出の例 1 を参照してください。

例 4

'NY' にあるすべての部署の給料 (salary) の合計を設定します。

```
SELECT D.deptname, sum(E.salary)
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept AND D.location =  ?
GROUP BY D.Deptname
```

上記の照会に基づく MQT を作成します。

```
CREATE TABLE MQT4
        AS (SELECT D.deptname, D.location, sum(E.salary) as sum_sal
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept AND D.location =  'NY'
GROUP BY D.deptnamet, D.location)
DATA INITIALLY IMMEDIATE  REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION
 MAINTAINED BY USER
```

この例では、定数 'NY' がパラメーター・マーカーで置き換えられ、MQT のデータ設定時に location='NY' のローカル選択が MQT に適用されています。 MQT マッチング・アルゴリズムはパラメーター・マーカーを、述部 D.Location=? の定数 'NY' と突き合わせます。これによって、パラメーター・マーカーの値が MQT の定数と同じであることが確認され、MQT を使用できることになります。

MQT マッチング・アルゴリズムは MQT と照会の間の述部が正確に同じではない場合にも突き合わせを試みます。例えば、MQT に SALARY > 50000 という述部があり、照会に SALARY > 70000 という述部がある場合、MQT には照会の実行に必要な行が入ります。 MQT は照会で使用されますが、述部 SALARY > 70000 は照会の選択として残されるため、SALARY が MQT の列でなければなりません。

例 5

```
SELECT D.deptname, sum(E.salary)
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept AND D.location =  'NY'
GROUP BY D.deptname
```

上記の照会に基づく MQT を作成します。

```
CREATE TABLE MQT5
        AS (SELECT D.deptname, E.salary
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptno=E.workdept)
DATA INITIALLY IMMEDIATE  REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION
 MAINTAINED BY USER
```

この例では、D.Location が MQT の列でないため、ユーザー照会のローカル選択述部 location='NY' を判別できません。したがって、この MQT は使用できません。

MQT にグループ化処理が含まれている場合、照会はグループ化照会でなければなりません。最も単純なケースは、MQT と照会とで、指定されているグループ化の列と列関数が同じ場合です。 MQT で指定されているグループ化列リストが照会のグループ化列を包含している場合には、再グループ化と呼ばれるステップを実行するよう照会を書き直すことができます。その場合、MQT のグループは、照会に必要なグループに再編成されます。再グループ化が必要な場合、列関数を再計算する必要があります。サポートされる再グループ化式は、以下の表に示されています。

再グループ化の新しい式/集約規則は、次のとおりです。

表 26. MQT の式/集約規則

照会	MQT	最終的な照会
COUNT(*)	COUNT(*) as cnt	SUM(cnt)
COUNT(*)	COUNT(C2) as cnt2 (c2 は NULL 可能ではない)	SUM(cnt2)
COUNT(c1)	COUNT(c1) as cnt	SUM(cnt)
COUNT(C1) (C1 は NULL 可能ではない)	COUNT(C2) as cnt2 (C2 は NULL 可能ではない)	SUM(cnt2)
COUNT(distinct C1)	C1 as group_c1 (C1 はグループ化列)	COUNT(group_C1)
COUNT(distinct C1)	C1 はグループ化列ではない	MQT は使用不可
COUNT(C2) (C2 は MQT に含まれないテーブルの列)	COUNT(*) as cnt	cnt*COUNT(C2)
COUNT(distinct C2) (C2 は MQT に含まれないテーブルの列)	適用されない	COUNT(distinct C2)
SUM(C1)	SUM(C1) as sm	SUM(sm)
SUM(C1)	C1 as group_c1, COUNT(*) as cnt (C1 はグループ化列)	SUM(group_c1 * cnt)
SUM(C2) (C2 は MQT に含まれないテーブルの列)	COUNT(*) as cnt	cnt*SUM(C2)

表 26. MQT の式/集約規則 (続き)

照会	MQT	最終的な照会
SUM(distinct C1)	C1 as group_c1 (C1 はグループ化列)	SUM(group_C1)
SUM(distinct C1)	C1 はグループ化列ではない	MQT は使用不可
SUM(distinct C2) (C2 は MQT に含まれないテーブルの列)	適用されない	SUM(distinct C2)
MAX(C1)	MAX(C1) as mx	MAX(mx)
MAX(C1)	C1 as group_C1 (C1 はグループ化列)	MAX(group_c1)
MAX(C2) (C2 は MQT に含まれないテーブルの列)	適用されない	MAX(C2)
MIN(C1)	MIN(C1) as mn	MIN(mn)
MIN(C1)	C1 as group_C1 (C1 はグループ化列)	MIN(group_c1)
MIN(C2) (C2 は MQT に含まれないテーブルの列)	適用されない	MIN(C2)

AVG、STDDEV、STDDEV_SAMP、VARIANCE_SAMP および VAR_POP は、COUNT と SUM の組み合わせを使用して計算されます。 AVG、STDDEV、または VAR_POP が MQT に含まれていて、再グループ化においてそれらの関数の再計算が必要になる場合、MQT は使用できません。 MQT では COUNT、SUM、MIN、および MAX だけを使用することをお勧めします。照会に AVG、STDDEV、または VAR_POP が含まれている場合、COUNT と SUM を使用して再計算できます。

MQT の中で FETCH FIRST N ROWS 文節が指定されている場合、照会の中でも FETCH FIRST N ROWS 文節が指定されていなければならず、MQT で指定されている行数は照会で指定されている行数以上でなければなりません。 MQT に FETCH FIRST N ROWS 文節を含めることは勧められていません。

REFRESH TABLE を実行する場合、MQT に対して ORDER BY 文節を使用することによって、MQT 内のデータを順序付けることができます。それは MQT マッチングにおいては無視され、照会に ORDER BY 文節が含まれている場合には、それが書き直し後の照会の一部になります。

関連資料

71 ページの『MQT でサポートされる機能』
MQT にはほとんどどんな照会でも入れることができますが、最適化プログラムでは、MQT とユーザー指定照会のマッチング時に照会機能のうちの一部しかサポートされていません。 SQE 最適化プログラムでは、ユーザー指定照会と MQT 照会の両方がサポートされていなければなりません。

不要な MQT の判別

照会の最適化で使用されている MQT の判別は簡単に行えます。しかし、System i ナビゲーターおよび i5/OS の機能の結果として、すべての MQT を検索したり、MQT の使用に関する統計を検索したりすることが容易になりました。

この機能により、照会での MQT 使用状況の統計が生成され、パフォーマンス調整のために利用できます。 System i ナビゲーターによりこの機能にアクセスするには、**「データベース」** → **「スキーマ」** → **「テーブル」**とナビゲートします。テーブルを右クリックしてから、**「マテリアライズ照会表の表示 (Show Materialized Query Tables)」**を選択します。「テーブル」または「ビュー」フォルダーを右クリックし、**「マテリアライズ照会表の表示 (Show Materialized Query Tables)」**を選択することによって、MQT 使用情報を表示することもできます。このアクションは、すべてのテーブルまたはそのスキーマのビューに関して作成された MQT の使用情報を表示します。

注: 統計データは、アプリケーション・プログラミング・インターフェース (API) を使用して表示することもできます。

MQT の既存属性すべてに加えて、2 つのフィールドが不要な MQT の判別に役立ちます。

これらのフィールドは、次のとおりです。

照会の最後の使用
照会でユーザー指定のテーブルを置換するために最適化プログラムによって MQT が最後に使用された時点のタイム・スタンプ。

照会使用回数
照会でユーザー指定のテーブルを置換するために最適化プログラムによって MQT が使用されたインスタンスの数。

これらのフィールドは、状況に応じて、あるいはシステムで現在実行中のアクションに応じてカウントを開始したり停止したりします。保管およびリストアの手順を実行して既存の MQT に対して MQT を上書きでリストアする場合、統計カウンターはリセットされません。システム上に存在しない MQT をリストアする場合、統計情報はリセットされます。

関連情報

Retrieve member description (QUSRMBRD) command

MQT 照会の推奨事項の要約

MQT 照会を使用の際は、以下の推奨事項に従ってください。

- MQT には、ローカル選択や定数を含めないようにしてください。それらを使用すると、Query 最適化プログラムで MQT を使用できるユーザー指定照会の数が制限されてしまいます。
- MQT のグループ化では、グループ化関数として SUM、COUNT、MIN、および MAX だけを使用してください。ユーザー指定照会の中では、Query 最適化プログラムにより AVG、STDDEV、および VAR_POP の再計算が可能です。
- MQT の中で FETCH FIRST N ROWS を指定すると、Query 最適化プログラムが MQT を使用できるユーザー指定照会の数が制限されるため、勧められていません。
- MQT 作成時に DATA INITIALLY DEFERRED が指定されている場合には、DISABLE QUERY OPTIMIZATION 文節を指定することによって、MQT のデータ設定時まで Query 最適化プログラムがその MQT を使用しないようにすることを考慮してください。 MQT のデータが設定されて使用可能な状態になった時点で、ENABLE QUERY OPTIMIZATION 文節を指定した ALTER TABLE ステートメントを使用することにより、Query 最適化プログラムが MQT を使用できるようになります。

MQT テーブルも、MQT 以外のテーブルと同じように最適化する必要があります。必要に応じて、MQT のうち選択、結合、およびグループ化に使用される列について、索引を作成してください。 MQT テーブルについての列統計は収集されます。

データベース・モニターには、最適化において考慮される MQT のリストが表示されます。その情報はレコード 3030 に含まれています。 QAQQINI ファイルを通じて MQT が有効にされていて、照会に含まれるテーブルのうち 1 つ以上について MQT が存在しているなら、その照会についてのレコード 3030 が存在します。 MQT ごとに、それが使用されこと、または使用されなかった場合にはなぜ使用されなかったかを示す理由コードが示されています。

再帰的照会の最適化

一部のアプリケーションおよびデータはその性質上、本来再帰的です。このようなアプリケーションの一例として挙げられるのは、部品表システム、予約システム、旅行プランナー・システム、またはネットワーク計画システムなど、1 つの結果行のデータが、他の 1 つまたは複数の行のデータと、本質的に関係 (親と子の関係) があるものです。これらのシステムに実装される再帰の種類は、SQL のストアード・プロシージャーと一時結果表を用いて実行できますが、この階層データへのアクセスを促すために再帰的照会を使用することで、アプリケーションをより効率的に実行できるようになります。

再帰的照会は、再帰的共通表式 (RCTE) または再帰的ビューを定義することにより、実装することができます。

再帰的照会の例

再帰的照会は、再帰をシードする初期化全選択、および FROM 文節内に直接自己参照を含む反復全選択を持つ Union All によって定義される照会です。

再帰的照会の定義に指定できるものに関しては追加の制約事項があり、これらは「SQL プログラミング」の中にあります。主な制約事項としては、関数実行の前にすべての修飾レコードの実体化を必要とする照会機能 (グループ化、集約、および DISTINCT など) が、反復全選択自体の中では使用できず、再帰を完了するためにメインの照会で要求しなければならないことがあります。

以下は、出発都市と到着都市に関する情報が含まれる flights という名前テーブルに対する再帰的照会の例です。照会は 2 つの指定都市 (ニューヨークとシカゴ) から再帰によって使用可能なすべてのフライト目的地を戻し、その最終目的地に到着するための接続便の数と合計のコストを戻します。

この例では再帰的処理を使用してランニング・コストや接続便の数などの情報の集計も行うため、実際に 4 つの値が待ち行列項目に置かれます。これらの値は、以下の通りです。

- 出発都市 (シカゴまたはニューヨークのいずれか) (これは再帰の開始から固定されたままとなるため)
- 後続の結合で使用される到着都市
- 増分する接続便の数
- 各目的地に到着するためのコストの累計

通常、待ち行列項目に必要なデータは全レコードと比べて少なくてすみますが (場合によってははるかに少なくてすむ)、この例にはこれは当てはまりません。

```
CREATE TABLE flights
 (
  departure CHAR (10) NOT NULL WITH DEFAULT,
  arrival CHAR (10) NOT NULL WITH DEFAULT,
  carrier CHAR (15) NOT NULL WITH DEFAULT,
  flight_num CHAR (5) NOT NULL WITH DEFAULT,
  ticket INT NOT NULL WITH DEFAULT)

WITH destinations (departure, arrival, connects, cost ) AS
(
   SELECT  f.departure,f.arrival, 0, ticket
  FROM flights f
  WHERE f.departure = 'Chicago' OR
      f.departure = 'New York'
  UNION ALL
    SELECT
      r.departure, b.arrival, r.connects + 1,
      r.cost + b.ticket
       FROM  destinations r, flights b
```

```
   WHERE r.arrival = b.departure
)
SELECT DISTINCT departure, arrival, connects, cost
 FROM destinations
```

以下は、上記の照会の初期化全選択です。これは再帰的処理を開始する行をシードします。これは、シカゴまたはニューヨークからの直接のフライトである最初の目的地 (到着都市) を提供します。

```
SELECT  f.departure,f.arrival, 0, ticket
 FROM flights f
 WHERE f.departure='Chicago' OR
       f.departure='New York'
```

以下は、上記の照会の反復全選択です。この FROM 文節には、宛先再帰的共通表式への 1 つの参照が含まれており、同じフライト・テーブルへの詳細な再帰的結合を入手します。親行の到着値 (最初はニューヨークまたはシカゴからの直接のフライト) は後続の子行の出発値と結合されます。再帰的結合述部での正しい親/子関係を識別しておくことは重要です。そうしないと、無限再帰が発生する可能性があります。再帰を制限するために、それ以外のローカル述部を使用することもできます。たとえば、最大で 3 つの接続フライトに制限したい場合、累積接続便数 r.connects<=3 を使用するローカル述部を指定できます。

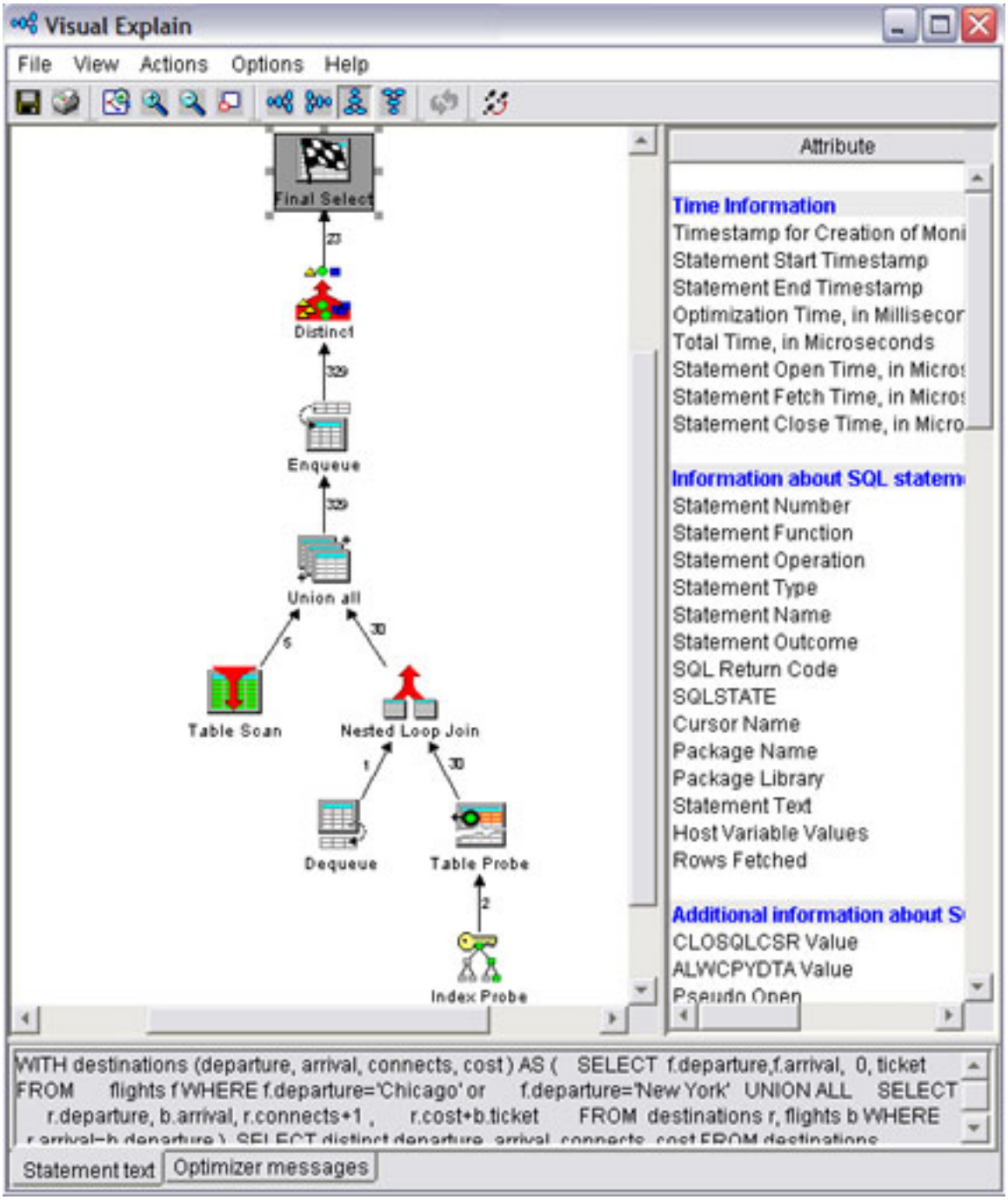
```
SELECT
  r.departure, b.arrival, r.connects + 1 ,
  r.cost + b.ticket
  FROM  destinations r, flights b
 WHERE r.arrival=b.departure
```

メイン照会は、再帰的共通表式またはビューを参照する照会です。これは、グループ化、順序付け、および DISTINCT などの要求が指定されるメイン照会にあります。

```
SELECT DISTINCT departure, arrival, connects, cost
 FROM destinations
```

インプリメンテーションに関する考慮事項

再帰のためのソースをインプリメントするために、待ち行列と呼ばれる一時データ・オブジェクトが新たに提供されています。行は初期化全選択または反復全選択のいずれかの要件を満たし、UNION ALL を介して引き出されるので、継続する再帰処理をフィードするために必要な値がキャプチャーされ、待ち行列の項目に置かれます (待ち行列への挿入操作)。その後、照会の実行時に、待ち行列のデータ・ソースは共通表式またはビューの再帰参照に取って代わります。反復全選択処理は、待ち行列の項目が使い果たされるか、または取り出し N 行の制限が満たされると終了します。再帰的待ち行列は再帰的処理をフィードし、一時データを保持するため、それらの待ち行列項目の待ち行列からの除去と全選択テーブルの残りの間の結合は常に制限付きの結合となり、待ち行列は左側に来ます。



複数の初期化全選択および反復全選択

再帰的照会定義で指定された初期化全選択および反復全選択を複数使用すると、再帰的処理をフィードするために、多数のデータ・ソースと個別の選択要件を使用することができます。

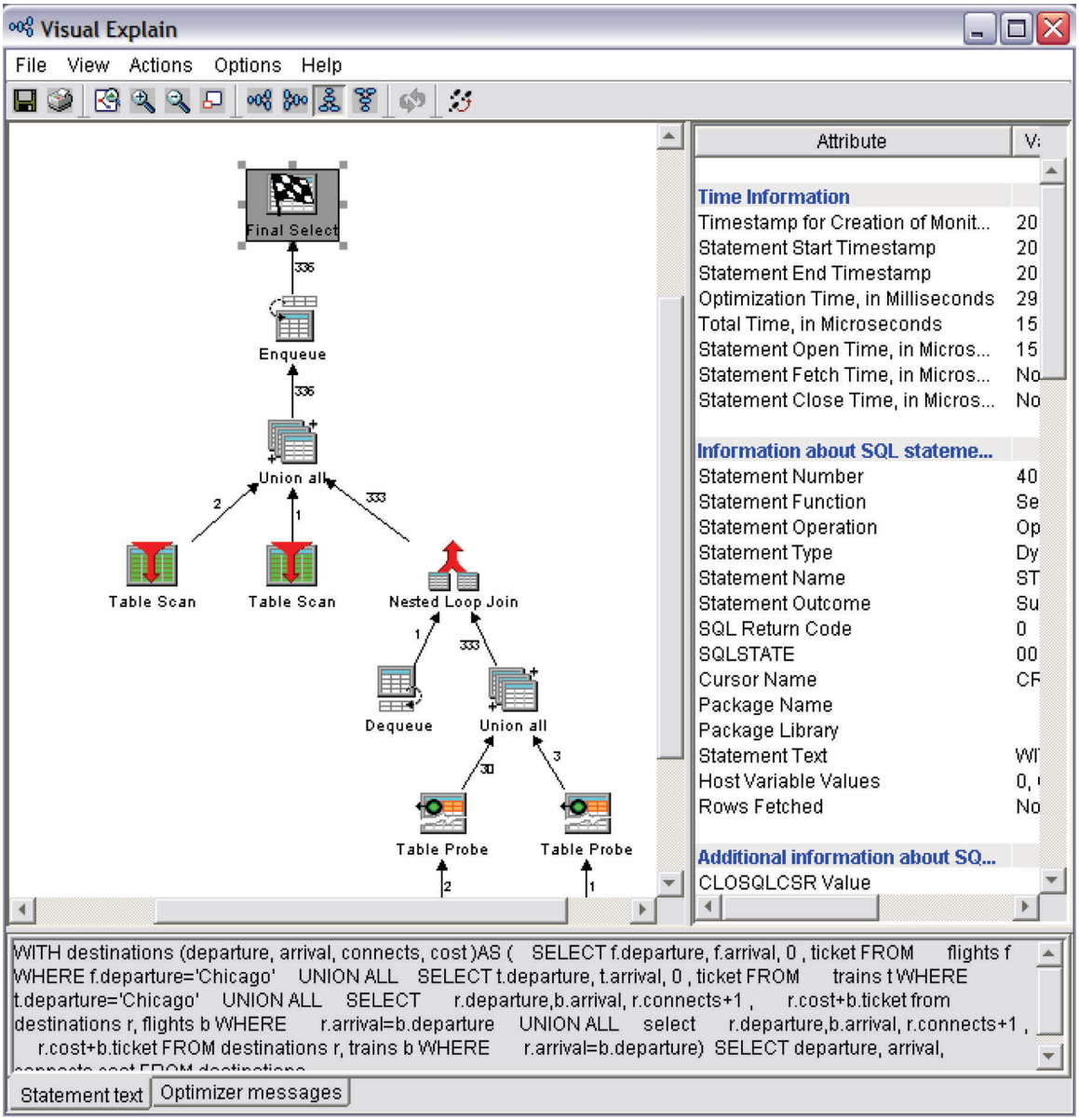
たとえば、以下の照会では、シカゴからアクセス可能な最終目的地を、空の旅と列車の旅の両方で使用できます。

```
WITH destinations (departure, arrival, connects, cost ) AS
(
   SELECT f.departure, f.arrival, 0 , ticket
  FROM flights f
  WHERE f.departure='Chicago'
```

```
   UNION ALL
   SELECT t.departure, t.arrival, 0 , ticket
  FROM trains t
  WHERE t.departure='Chicago'
   UNION ALL
   SELECT
  r.departure,b.arrival, r.connects + 1 ,
  r.cost + b.ticket
  FROM destinations r, flights b
  WHERE r.arrival=b.departure
   UNION ALL
  SELECT
   r.departure,b.arrival, r.connects+1 ,
   r.cost+b.ticket
   FROM destinations r, trains b
   WHERE r.arrival=b.departure)

SELECT departure, arrival, connects,cost
   FROM destinations;
```

RCTE/View から来る行はすべて再帰的処理の一部であり、フィードバックする必要があるため、共通表式を複数の全選択が参照する場合、再帰的初期化全選択以外をまずすべて処理し、その後 1 つの待ち行列を使用してそれらの同じ行とそれ以外のすべての行結果を同じように反復全選択にフィードするように照会が最適化プログラムによって書き換えられます。 RCTE/view の定義内で初期化全選択と反復全選択をどのように配列するとしても、まず初期化全選択が実行され、反復全選択は待ち行列の内容への同等のアクセスを共用します。



述部のプッシュ

通常、照会を非再帰的共通表式または共通表ビューと共に処理する場合、主照会に指定されたローカル述部はプッシュダウンされ、マテリアライズの必要なレコードが減少します。ただし、主照会のローカル述部を照会の定義済み再帰部分 (Union All を介して) にプッシュすると、再帰のプロセス自体が大幅に変更される可能性があります。そのため一般的なルールとして、再帰的照会で指定された Union All は、現時点で述部の境界線とし、述部がこの線を越えてプッシュダウン、またはプッシュアップされることはありません。

以下に再帰への述部のプッシュにより、どのように再帰結果が制限され、照会の意図が変わるかを例示します。

最終目的地である「Dallas」を含めることなく、「Chicago」からアクセス可能なすべての行き先を検索することが照会の趣旨だとすると、「arrival<>'Dallas'」述部を再帰的照会にプッシュすることにより、「Dallas」が最終目的地ではなく「Dallas」が中間の停止地点であるような最終目的地が出力されず、意図した結果の出力は変わってしまいます。

```
WITH destinations (departure, arrival, connects, cost ) AS
(
   SELECT f.departure,f.arrival, 0, ticket
  FROM flights f
  WHERE f.departure='Chicago'
  UNION ALL
    SELECT
   r.departure, b.arrival, r.connects + 1 ,
   r.cost + b.ticket
   FROM  destinations r, flights b
   WHERE r.arrival=b.departure
)
SELECT departure, arrival, connects, cost
  FROM destinations
  WHERE arrival != 'Dallas'
```

逆に、以下に示すのは、すべての再帰結果に適用されたローカル述部が、再帰的定義のボディでプットされる述部の好例です。これは RCTE/ビューからマテリアライズされた行数を大幅に削減できる可能性があるためです。照会要求をさらに改良し、 r.connects <=3 のローカル述部に、反復全選択、RCTE 定義を指定します。

```
WITH destinations (departure, arrival, connects, cost ) AS
(
   SELECT f.departure,f.arrival, 0, ticket
  FROM flights f
  WHERE f.departure='Chicago' OR
      f.departure='New York'
  UNION ALL
    SELECT
   r.departure, b.arrival, r.connects + 1 ,
   r.cost + b.ticket
   FROM  destinations r, flights b
   WHERE r.arrival=b.departure
)
SELECT departure, arrival, connects, cost
  FROM destinations
  WHERE r.connects<=3
```

再帰的照会では、ローカル述部が再帰定義にプッシュされた場合には、再帰的結果が誤って導き出されたり、ローカル述部が再帰を制限する場合でも不必要な行がマテリアライズされてから拒否されたりする場合があるため、ローカル述部の配置が鍵になります。

SEARCH の指定に関する考慮事項

階層データ、再帰的データを処理する一部のアプリケーションでは、データが処理される方法 (深さ、幅など) について、要件を設けている場合があります。

再帰的結合キー値をトラッキングするために、待ち行列メカニズム (First In First Out) を使用することで、幅優先順序による結果の検索を暗黙指定したことになります。幅優先とは、親行に対する直接の子をすべて検索し、その後に同じ行の孫を検索することを意味します。ただし、これは実装上の特徴であり、必ずしもこの方法が取られるわけではありません。アプリケーションではデータの取得方法を確定することもできます。深さ優先順位で階層データを検索するアプリケーションもあります。深さ優先は、直接の子の行ごとに下層すべてを検索し、その後に次の子の行の下層が検索される方式です。

SQL アーキテクチャーでは、SEARCH DEPTH FIRST、または BREADTH FIRST キーワードを使用することにより、アプリケーションによる結果のデータ検索方法を指定できます。再帰的結合値のネーミング、SET順序列の特定、外部 ORDER BY 文節での順序列の指定と共に、このオプションが指定されると、結果は深さ優先順序、または幅優先順序にて出力されます。最終的に、これは関係に基づいたソートであり、値に基づいたソートではありません。

前述の深さ優先順序の出力例を示します。

```
WITH destinations (departure, arrival, connects, cost ) AS
(
   SELECT f.departure, f.arrival, 0 , ticket
    FROM flights f
    WHERE f.departure='Chicago' OR f.departure='New York'
   UNION ALL
    SELECT
   r.departure,b.arrival, r.connects+1 ,
   r.cost+b.ticket
   FROM  destinations r, flights b
   WHERE r.arrival=b.departure)

SEARCH DEPTH FIRST BY arrival SET depth_sequence

 SELECT *
  FROM destinations
  ORDER BY depth_sequence
```

メインの照会で ORDER BY 文節が指定されていない場合、オプションの順序付けは無視されます。正しいソートを促すため、再帰の待ち行列項目にプットする追加情報があります。BREADTH FIRST では再帰レベル番号、および直接の祖先の結合値になります。そのため、兄弟行は一緒にソートされます。深さ優先検索は、比較的データ集約型だと言えます。DEPTH FIRST では照会エンジンは、現在行に至る結合値の系統全体を表示すること、およびその情報を待ち行列項目にプットすることが必要になります。また、これらのソート値は外部データ・ソースが基になっているわけではないため、ソートの実装は常に、一時的なソート・リスト (索引使用不可) によって行われます。

情報の順序付けを処理するために CPU とメモリーの付加的なオーバーヘッドがあるため、深さ優先方式、あるいは幅優先方式でデータをマテリアライズするための要件を備えていない場合は、SEARCH オプションを使用しないでください。

CYCLE の指定に関する考慮事項

再帰的照会で使用されるテーブルのデータがその性質上循環的となり得ることを理解しておくのは、無限ループを防ぐために重要です。

SQL アーキテクチャーではオプションで、循環データの検査を行うことができ、反復する循環をその時点で中止します。この追加の検査は CYCLE オプションの使用により行えます。 CYCLE 要求には正確な結合再帰値と、循環標識を指定する必要があります。循環標識はオプションで、メインの照会への出力が可能で、誤った循環データを判別し、訂正する助けとして使用できます。

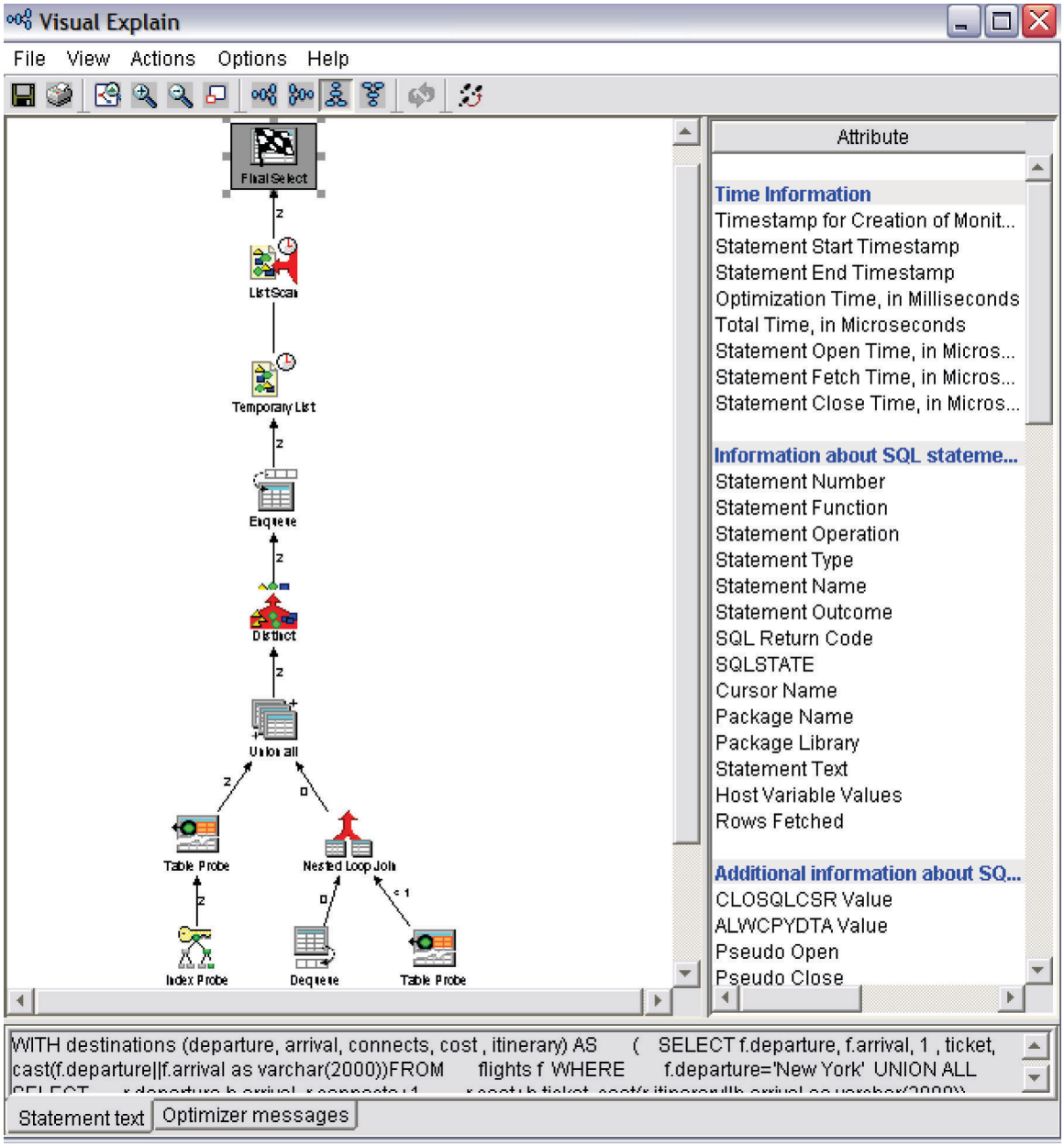
```
WITH destinations (departure, arrival, connects, cost , itinerary) AS
      (
   SELECT f.departure, f.arrival, 1 , ticket, CAST(f.departure||f.arrival AS VARCHAR(2000))
   FROM flights f
   WHERE f.departure='New York'
 UNION ALL
      SELECT r.departure,b.arrival, r.connects+1 ,
      r.cost+b.ticket, cast(r.itinerary||b.arrival AS varchar(2000))
   FROM destinations r, flights b
   WHERE r.arrival = b.departure)
CYCLE arrival SET cyclic TO '1' DEFAULT '0' USING Cycle_Path

SELECT departure, arrival, itinerary, cyclic
  FROM destinations
```

循環が反復であると判別されると、その行の循環順序の出力は停止されます。しかし、「反復」値がないか検査するには、照会エンジンは、反復している結合値を探すために、現在行にまで続く結合値のすべての先祖を表す必要があります。この先祖履歴は、それぞれの再帰循環によって付加され、待ち行列項目のフィー

ルドに置かれる情報です。これをインプリメントするには、照会エンジンは、値が以前に現れたかどうかを判別するために、累積されている先祖に対して固定長の迅速な走査を行えるよう、先祖チェーンに関する再帰値の圧縮表記を使用します。この圧縮表記は、照会ツリー内の特殊ノードの使用によって決定されます。

データが循環的かどうかが分からない場合、または CYCLE オプションの使用目的が特に修正または検査の目的で循環を見つけることである場合は、CYCLE オプションを使用しないでください。特定の行をマテリアライズする前に、反復する循環を管理したり、その有無を検査したりするために、CPU およびメモリーの付加的なオーバーヘッドが発生します。



SMP および 再帰的照会

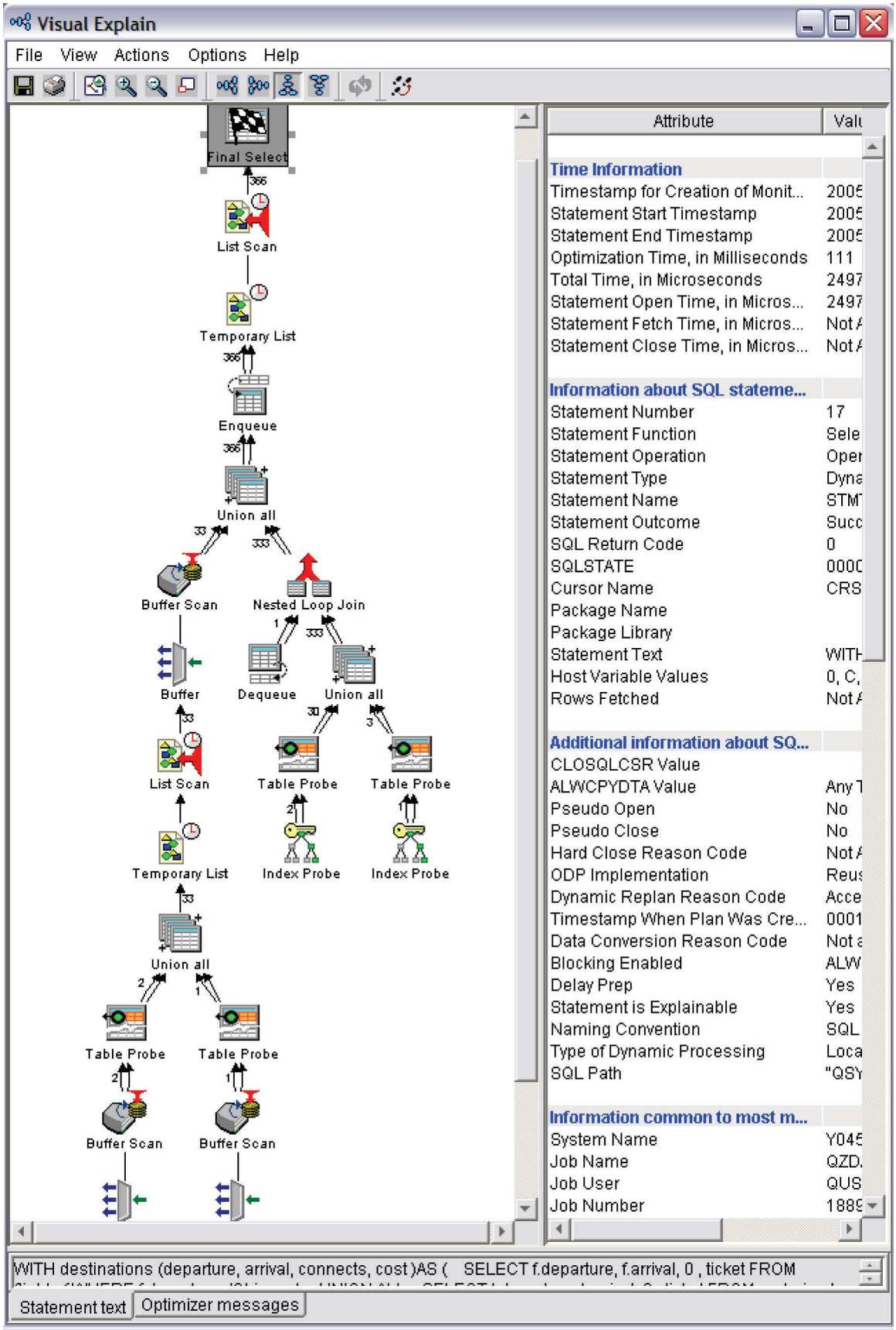
再帰的照会は、システム上の他の照会が、対称型マルチプロセッシング (SMP) から得る利益と同等の利益を SMP から得ることができます。

しかし、再帰的照会と並列処理には固有の要件がいくつかあります。再帰的照会の初期化全選択 (再帰の初期値となる全選択) は再帰プロセスを循環する、最終結果の一部のみを生成する傾向があるため、Query 最適化プログラムに、複数のスレッドを並行で実行し、待ち行列オブジェクト自体を送る固有のオブジェクトを持たせることは望ましくありません。この場合、一部のスレッドに作業が偏り、他のスレッドでは直ちに作業がなくなる原因になります。これを実行する最良の方法は、すべてのスレッドに同じ待ち行列を共有させ、待機中のスレッドが要求を待ち行列から除去するのと同様に、スレッドが新しい再帰的キー値を待ち行列に入れられるようにする方法です。共有待ち行列は、より多くの結果を提供できるスレッドがなくなるまで、すべてのスレッドを活用して、待ち行列項目全体を減少させるようにします。ただし、同じ待ち行列を複数のスレッドで共有することは、スレッドが早期に終了してしまわないよう、QUERY ランタイムで管理する必要が生じます。初期シード値のバッファリングが必要になる可能性があります。以下の照会では、再帰をシードする全選択が 2 つあり、バッファーが提供されています。そのため、照会が処理を続行するために十分な再帰的値をシードする前に、待ち行列除去状態に陥る、および終了するスレッドはありません。

以下の Visual Explain ダイアグラムでは、以下の照会を CHGQRYA DEGREE(*NBRTASKS 4) と共に実行するためのプランを説明しています。ここでは、バッファーに入れられた複数の初期化全選択の結果と、複数のスレッド (複数の矢印にて表示) による要求ノードを待ち行列に入れる処理、および待ち行列から除去する処理の様子を説明しています。すべての SMP 照会と同様、複数のスレッド (この場合は 4 つ) が、それぞれの結果を Temporary List オブジェクトにプットしています。このオブジェクトは、主照会の出力先になっています。

```
cl:chgqrya degree(*nbrtasks 4);

WITH destinations (departure, arrival, connects, cost )AS
(
   SELECT f.departure, f.arrival, 0 , ticket
   FROM flights f WHERE f.departure='Chicago'
   UNION ALL
   SELECT t.departure, t.arrival, 0 , ticket
   FROM trains t WHERE t.departure='Chicago'
   UNION ALL
   SELECT
      r.departure,b.arrival, r.connects+1 ,
      r.cost+b.ticket
    FROM destinations r, flights b
    WHERE r.arrival=b.departure
   UNION ALL
    SELECT
      r.departure,b.arrival, r.connects+1 ,
      r.cost+b.ticket
    FROM destinations r, trains b
    WHERE r.arrival=b.departure)
SELECT departure, arrival, connects,cost
  FROM destinations;
```


Visual Explain
File View Actions Options Help
Final Select
List Scan
Temporary List
Enqueue
Union all
Buffer Scan
Nested Loop Join
Buffer
Dequeue
Union all
List Scan
Table Probe
Table Probe
Temporary List
Index Probe
Index Probe
Union all
Table Probe
Table Probe
Buffer Scan
Buffer Scan
Attribute
Time Information
Timestamp for Creation of Monit...
Statement Start Timestamp
Statement End Timestamp
Optimization Time, in Milliseconds
Total Time, in Microseconds
Statement Open Time, in Micros...
Statement Fetch Time, in Micros...
Statement Close Time, in Micros...
Information about SQL stateme...
Statement Number
Statement Function
Statement Operation
Statement Type
Statement Name
Statement Outcome
SQL Return Code
SQLSTATE
Cursor Name
Package Name
Package Library
Statement Text
Host Variable Values
Rows Fetched
Additional information about SQ...
CLOSQLCSR Value
ALWCPYDTA Value
Pseudo Open
Pseudo Close
Hard Close Reason Code
ODP Implementation
Dynamic Replan Reason Code
Timestamp When Plan Was Cre...
Data Conversion Reason Code
Blocking Enabled
Delay Prep
Statement is Explainable
Naming Convention
Type of Dynamic Processing
SQL Path
Information common to most m...
System Name
Job Name
Job User
Job Number
WITH destinations (departure, arrival, connects, cost)AS (SELECT f.departure, f.arrival, 0 , ticket FROM
Statement text
Optimizer messages

Query 最適化ツールを使用した照会パフォーマンスの最適化

Query 最適化は反復プロセスです。照会に関するパフォーマンス情報を収集し、照会の処理を制御することができます。

データベース・ヘルス・センターを使用した情報の表示

データベースに関する情報をキャプチャーするには、データベース・ヘルス・センターを使用します。オブジェクトの総数、データベース内で選択されたオブジェクトのサイズ限界、選択されたオブジェクトの設計限界、環境上の制約、および活動レベルを表示できます。

ヘルス・センターを開始するには、以下のステップに従います。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開します。
3. 処理したいデータベースを右クリックして、「**ヘルス・センター**」を選択します。

「**変更**」をクリックしてフィルター情報を入力することにより、プリファレンスを変更することができます。情報を更新するには、「**最新表示**」をクリックします。

ヘルス・センター・ヒストリーを保管するには、以下を実行してください。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開します。
3. 処理したいデータベースを右クリックして、「**ヘルス・センター**」を選択します。
4. 「ヘルス・センター」ダイアログで、保管する領域を選択します。例えば、現在の概要を保管する場合は、「概要」タブの「**保管**」を選択します。サイズの制限および設計上の制限は保管されません。
5. 情報を保管するスキーマおよびテーブルを指定します。選択したテーブルの内容を表示するには、「**内容の表示**」をクリックします。存在していないテーブルに情報を保管するように選択した場合は、システムがそのテーブルを作成してくれます。

データベース・モニターの開始 (STRDBMON) を使用した照会のモニター

データベース・モニターの開始 (STRDBMON) を使用すると、リアルタイムに照会に関する情報が収集され、その情報が出力テーブルに保管されます。この情報は、ユーザーのシステムまたは照会が所定のとおりに実行されているかどうか、あるいは微調整が必要かどうかを判別するのに役立てることができます。データベース・モニターを使用すると、かなりの CPU およびディスク装置オーバーヘッドが使用中に生成される可能性があります。

パフォーマンス情報は、特定の照会について、システム上の各照会について、またはシステム上の照会のグループについて収集することができます。ジョブが複数のモニターでモニターされる場合、各モニターは、異なる出力テーブルに対して行のログ記録を行います。出力データベース・テーブル内の行は、各行固有の識別番号によって識別することができます。

| データベース・モニターの開始 (STRDBMON) コマンドを使用してモニターを開始すると、モニターは、
| System i ナビゲーターとともに自動的に登録され、System i ナビゲーター・モニター・リストに表示され
| ます。

収集できる統計の種類

データベース・モニターからは、 Query 最適化プログラム・デバッグ・メッセージ (デバッグの開始 (STRDBG)) および SQL 情報の印刷 (PRTSQLINF) コマンドで提供されるものと同じ情報が提供されます。以下は、データベース・モニターによって収集される追加情報のサンプルです。

- システム名およびジョブ名
- SQL ステートメントおよび副選択番号
- 開始と終了のタイム・スタンプ
- 処理時間の見積もり
- 照会されるテーブル内の合計行数
- 選択された行数
- 選択された行数の見積もり
- 結合された行数の見積もり
- 推奨索引のキー列
- 最適化時間の合計
- 結合タイプと方式
- ODP 実施

パフォーマンス統計の使用法

これらのパフォーマンス統計を使用して、各種報告書を生成することができます。たとえば、次のような照会を示す報告書を組み込むことができます。

- 大量のシステム・リソースを使用するもの。
- 実行するのにきわめて長い時間を要する照会。
- 照会管理プログラムの時間制限のため実行されなかった照会。
- 実行中の一時索引の作成。
- 実行時に照会分類を使用する照会。
- Query 最適化プログラムによって示されたキーを含むキー順論理ファイルを作成すると、パフォーマンスを向上させる可能性のある照会。

注: 終了要求によって取り消された照会は、一般的にはパフォーマンス統計の完全なセットを生成しません。ただし、実行時間またはマルチステップ照会情報を除くと、照会がどのように最適化されたかについてのすべての情報は入っています。

関連情報

デバッグの開始 (STRDBG) コマンド

SQL 情報印刷 (PRTSQLINF) コマンド

データベース・モニターの開始 (STRDBMON) コマンド

データベース・モニターの開始 (STRDBMON) コマンド

データベース・モニターの開始 (STRDBMON) コマンドは、指定されたジョブ、システム上のすべてのジョブ、または、選択した一連のジョブに対し、データベース・パフォーマンス統計の収集を開始します。この統計は、ユーザー指定のデータベース・テーブルとメンバーに入れられます。テーブルまたはメンバーが

存在しない場合、QSYS ライブラリーの QAQQDBMN テーブルに基づき、テーブルまたはメンバーが作成されます。テーブルおよびメンバーが存在する場合は、指定されたテーブルのレコード形式が同じであることを保証するため検査されます。

STRDBMON コマンドで開始された各モニターには、個々のモニターを一意的に識別するためのモニター ID が生成されます。このモニター ID は、ENDDBMON コマンドで終了するモニターを一意的に識別する場合にも用いられます。このモニター ID は、STRDBMON コマンドが実行されるたびに生成される情報メッセージ CPI436A で返されます。モニター ID は、QQQ3018 データベース・モニター・レコードの列 QQC101 にもあります。

概して、モニターには 2 種類あります。プライベート・モニターとは、1 つの特定のジョブ (または現在のジョブ) に限定したモニターを意味します。特定のジョブで 1 度に開始できるモニターは 1 つだけです。例えば、同じジョブ内で、STRDBMON JOB(*) の後にもう 1 つ STRDBMON JOB(*) を指定することはできません。パブリック・モニターは、複数のジョブ間でデータを収集するモニターです。 1 度に最大で 10 個のパブリック・モニターをアクティブにすることができます。たとえば、指定したパブリック・モニターの最大数が 10 個を超えていなければ、STRDBMON JOB(*ALL) の後にもう 1 つ STRDBMON JOB(*ALL) を指定できます。どのような特定ジョブでも、同時にパブリック・モニターを 10 個、プライベート・モニターを 1 個、アクティブにすることができます。

複数のモニターに同じ出力ファイルが指定された場合には、各ジョブで、データベース統計レコードをコピーしたデータが 1 つだけ、指定された出力ファイルに書き込まれます。たとえば、STRDBMON OUTFILE(LIB/TABLE1) JOB(*) と STRDBMON OUTFILE(LIB/TABLE1) JOB(*ALL) では同じ出力ファイルが、宛先に指定されています。現行ジョブの場合、プライベート・モニター用として 1 つ、パブリック・モニター用として 1 つのように、データベース統計レコードのコピーを 2 つ受け取ることはありません。ユーザーが受け取るデータベース統計レコードのコピーは 1 つだけです。

すべてのジョブでモニター (パブリック・モニター) が開始されている状態では、ジョブ待ち行列で待機中のジョブ、またはモニター中に開始されたジョブは、モニター対象データに含まれます。モニターが特定のジョブ (プライベート・モニター) で開始されている場合、コマンドを出した時点で、そのジョブがシステム内でアクティブでなければなりません。システム内の各ジョブは、プライベート・モニター 1 個、および最大 10 個のパブリック・モニターによって、並行してモニターすることができます。

STRDBMON コマンドにより、ジョブで実行されている照会の特定のセット、またはサブセットの統計レコードを収集できます。このフィルタリングは、ジョブ名、ユーザー・プロファイル、照会されているテーブルの名前、照会の見積もり実行時間、TCP/IP IP アドレスに対して、またはこれらのフィルターを組み合わせて実行することができます。 STRDBMON フィルターは任意のモニター用にキャプチャーする統計レコードの数を最小限に抑えるために役立つはずです。

例 1: 共通モニターを開始する

```
  STRDBMON   OUTFILE(QGPL/FILE1)  OUTMBR(MEMBER1 *ADD)
   JOB(*ALL)  FRCRCD(10))
```

このコマンドは、システム上のジョブすべてに対するデータベース・モニタリングを開始します。パフォーマンスの統計が、QGPL ライブラリーにあるファイル名 FILE1 の、メンバー名 MEMBER1 に追加されます。 10 件のレコードが蓄積された段階で、ファイルに書き込まれます。

例 2: 専用モニターを開始する

```
  STRDBMON   OUTFILE(*LIBL/FILE3)  OUTMBR(MEMBER2)
   JOB(134543/QPGMR/DSP01)  FRCRCD(20)
```

| このコマンドは、ジョブ番号 134543 に対し、データベース・モニタリングを開始します。ジョブ名は
| DSP01 で、ユーザー QPGMR により開始されました。パフォーマンスの統計が、ファイル名 FILE3 のメ
| ンバー名 MEMBER2 に追加されます。 20 件のレコードが蓄積された段階で、ファイルに書き込まれま
| す。

例 3: 独立 ASP にあるライブラリーのファイルに対し、専用モニターを開始する

```
STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(134543/QPGMR/DSP01)
```

| このコマンドは、ジョブ番号 134543 に対し、データベース・モニタリングを開始します。ジョブ名は
| DSP01 で、ユーザー QPGMR により開始されました。パフォーマンスの統計が、LIB41 ライブラリーにあ
| るファイル名 DBMONFILE のメンバー名 DBMONFILE (OUTMBR が指定されていないため) に追加され
| ます。このライブラリーは複数の独立補助記憶域プール (ASP) に存在する可能性があります。オリジネー
| ターのジョブのネーム・スペースにあるライブラリーが常に使用されます。

例 4: 「QZDA」で開始するすべてのジョブで共通モニターを開始する

```
  STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL/*ALL/QZDA*)
```

| このコマンドはジョブ名が「QZDA」で開始するジョブすべてに対するデータベース・モニターを開始しま
| す。パフォーマンスの統計 (モニター・レコード) が、LIB41 ライブラリーにあるファイル DBMONFILE
| のメンバー DBMONFILE (OUTMBR が指定されていないため) に追加されます。このライブラリーは複数
| の独立補助記憶域プール (ASP) に存在する可能性があります。オリジネーターのジョブのネーム・スペー
| スにあるライブラリーが常に使用されます。

例 5: 共通モニターを開始し、実行に 10 秒以上時間がかかる SQL ステートメントをフィルタリングする

```
  STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL)  RUNTHLD(10)
```

| このコマンドは、ジョブすべてに対するデータベース・モニタリングを開始します。ただし、それらの
| SQL ステートメントに対してモニター・レコードが作成されるのは、見積もり実行時間が 10 秒、または
| 10 秒を超える場合に限られます。

例 6: 共通モニターを開始し、200 メガバイトを超す見積一時ストレージを持つ SQL ステートメントをフィルタリングする

```
  STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL)  STGTHLD(200)
```

| このコマンドは、ジョブすべてに対するデータベース・モニタリングを開始します。ただし、それらの
| SQL ステートメントに対してモニター・レコードが作成されるのは、見積一時ストレージが 200 メガバイ
| ト、またはそれを超える場合に限られます。

例 7: 専用モニターを開始し、特定のファイルをフィルタリングする

```
  STRDBMON   OUTFILE(LIB41/DBMONFILE) JOB(*)  FTRFILE(LIB41/TABLE1)
```

| このコマンドは、現行ジョブに対するデータベース・モニタリングを開始します。モニター・レコードは、
| ファイル LIB41/TABLE1 を使用する SQL ステートメントに対してのみ作成されます。

例 8: 現行ユーザーに対して専用モニターを開始する

```
  STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*)  FTRUSER(*CURRENT)
```

| このコマンドは、現行ジョブに対するデータベース・モニタリングを開始します。モニター・レコードは、
| 現行ユーザーによって実行される SQL ステートメントに対してのみ作成されます。

例 9: 「QZDA」で開始するジョブに対する共通モニターを開始し、実行時間とファイルをフィルタリングする

```
STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL/*ALL/QZDA*)
 RUNTHLD(10)  FTRUSER(DEVLPR1)  FTRFILE(LIB41/TTT*)
```

このコマンドはジョブ名が「QZDA」で開始するジョブすべてに対するデータベース・モニタリングを開始します。モニター・レコードは、以下の条件すべてを満たす SQL ステートメントに対し、作成されます。

- Query 最適化プログラムにより算出された見積もり実行時間が 10 秒、または 10 秒を超える。
- ユーザー「DEVLPR1」により実行された。
- ファイル名の先頭が「TTT」で、ライブラリー LIB41 にあるファイルを使用する。

例 10: 共通モニターを開始し、IP アドレス ″9.10.111.77″ を持つ SQL ステートメントをフィルタリングする

```
STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL)
 FTRINTNETA("9.10.111.77")
```

このコマンドは、ジョブすべてに対するデータベース・モニタリングを開始します。モニター・レコードは、クライアント IP バージョン 4 アドレス ″9.10.111.77″ を使用中の TCP/IP データベース・サーバー・ジョブに対してのみ作成されます。

例 11: 共通モニターを開始し、ポート番号 8471 の SQL ステートメントをフィルタリングする

```
STRDBMON   OUTFILE(LIB41/DBMONFILE)  JOB(*ALL)  FTRLCLPORT(8471)
```

このコマンドは、ジョブすべてに対するデータベース・モニタリングを開始します。モニター・レコードは、ローカル・ポート番号 8471 を使用中の TCP/IP データベース・サーバー・ジョブに対してのみ作成されます。

例 12: 照会管理プログラムからのフィードバックに基づいて共通モニターを開始する

```
CHGSYSVAL QQRYTIMLMT(200)
 STRDBMON   OUTFILE(LIB41/DBMONFILE) JOB(*ALL) FTRQRYGOVR(*COND)
```

このコマンドは、照会管理プログラムへの応答に基づいて、推定実行時間が 200 秒を超えることが予期されるすべてのジョブに対してデータベース・モニターを開始します。この例では、照会が取り消されたか、照会管理出口プログラムによって、戻りコード 2 が戻された場合にのみ、データが収集されます。照会は、照会メッセージ CPA4259 (これは、照会が照会管理プログラムの限度を超えたために発行される) へのユーザー応答によって取り消すか、または、登録された照会管理出口プログラム内部のプログラム・ロジックによって取り消すことができます。

関連情報

データベース・モニターの開始 (STRDBMON) コマンド

データベース・モニターの終了 (ENDDBMON) コマンド

データベース・モニターの終了 (ENDDBMON) コマンドは、指定されたジョブ、システム上のすべてのジョブ、または選択されたジョブのセット (たとえば、汎用ジョブ名) のデータベース・パフォーマンス統計の収集を終了します。

モニターを終了するために、ジョブまたはモニター ID、あるいはその両方を指定できます。 JOB パラメーターのみが指定される場合、全く同じ JOB パラメーターを使って開始されたモニターは終了されます (指定された JOB と一致するモニターが 1 つしかない場合)。指定された JOB と一致する複数のモニターがアクティブの場合、MONID パラメーターの使用によって終了されるモニターをユーザーは一意的に識別

します。 MONID パラメーターのみが指定されると、指定された MONID は、現行ジョブのモニターのモニター ID、およびアクティブになっているすべての共通モニター (複数のジョブでオープンされているモニター) のモニター ID と比較されます。指定された MONID と一致するモニターは終了されます。

モニター ID は通知メッセージ CPI436A の中で戻されます。このメッセージは、STRDBMON コマンドが実行されるたびに生成されます。必要な場合は、メッセージ CPI436A について joblog を調べ、システム生成のモニター ID を探してください。モニター ID は、QQQ3018 データベース・モニター・レコードの列 QQC101 にもあります。

制約事項

- データベース・モニターの開始 (STRDBMON) コマンドで特定のジョブ名およびジョブ番号、または JOB(*) が指定された場合、ENDDBMON コマンドで同じジョブ名およびジョブ番号、または JOB(*) の指定によってのみ、モニターを終了できます。
- データベース・モニターの開始 (STRDBMON) で JOB(*ALL) が指定された場合、ENDDBMON JOB(*ALL) の指定によってのみモニターを終了できます。 ENDDBMON JOB(*) の指定でモニターを終了することはできません。

すべてのジョブに対するモニターが終了すると、システム上のすべてのジョブは、データベース・モニターの出力テーブルをクローズするよう起動されます。しかし、ENDDBMON コマンドは、モニターされたジョブのすべてがその最終統計レコードをログに書き込む前に、完了することができます。オブジェクト・ロックの処理 (WRKOBJLCK) コマンドを使用して、すべてのモニター・ジョブでデータベース・モニター出力テーブルに保持していたロックがなくなっていることを判別してから、そのモニターが終了したと見なします。

例 1: 特定ジョブのモニター終了

```
ENDDBMON   JOB(*)
```

このコマンドは、現行ジョブのデータベースのモニターを終了します。

例 2: すべてのジョブのモニター終了

```
ENDDBMON   JOB(*ALL)
```

このコマンドは、システム上のすべてのジョブでオープンされているモニターを終了します。 JOB(*ALL) が指定された複数のモニターがアクティブの場合、終了する特定の共通モニターを一意的に識別するために、MONID パラメーターも指定する必要があります。

例 3: MONID パラメーターによる個々の共通モニターのモニター終了

```
ENDDBMON   JOB(*ALL) MONID(061601001)
```

このコマンドは、JOB(*ALL) によって開始され、モニター ID 061601001 を持つモニターを終了します。JOB(*ALL) によって開始されたモニターは複数あるため、JOB(*ALL) によって開始されたモニターのどれを終了するかを一意的に識別するためにモニター ID を指定する必要があります。

例 4 MONID パラメーターによる個々の共通モニターのモニター終了

```
ENDDBMON   MONID(061601001)
```

このコマンドは、上述の例と同じ機能を果たします。このコマンドは、JOB(*ALL) または JOB(*) によって開始され、モニター ID 061601001 を持つモニターを終了します。

例 5: すべての JOB(*ALL) モニターのモニター終了

```
ENDDBMON   JOB(*ALL/*ALL/*ALL) MONID(*ALL)
```

このコマンドは、複数のジョブでアクティブになっているすべてのモニターを終了します。これは特定ジョブまたは現行ジョブでオープンされているモニターは終了しません。

例 6: 一般ジョブのモニター終了

```
ENDDBMON   JOB(QZDA*)
```

このコマンドは、JOB(QZDA*) によって開始されたモニターを終了します。 JOB(QZDA*) が指定された複数のモニターがアクティブの場合、終了する個々のモニターを一意的に識別するために、MONID パラメーターも指定する必要があります。

例 7: 一般ジョブによる個々のモニターのモニター終了

```
ENDDBMON   JOB(QZDA*) MONID(061601001)
```

このコマンドは、JOB(QZDA*) によって開始され、モニター ID 061601001 を持つモニターを終了します。 JOB(QZDA*) によって開始されたモニターは複数あるため、終了する JOB(QZDA*) モニターを一意的に識別するためにモニター ID を指定する必要があります。

例 8: 一般ジョブのグループのモニター終了

```
ENDDBMON   JOB(QZDA*) MONID(*ALL)
```

このコマンドは、JOB(QZDA*) によって開始されたすべてのモニターを終了します。

関連情報

データベース・モニターの終了 (ENDDBMON) コマンド

データベース・モニター・パフォーマンス行

データベース・テーブル内の行は、その行識別番号によって一意的に識別されます。ファイル・ベースのモニター (データベース・モニターの開始 (STRDBMON)) 内の情報は、『データベース・モニター: 形式』で定義される論理形式のセットに基づいて書き込まれます。これらのビューは、デバッグ・メッセージおよび SQL 情報の印刷 (PRSQLINF) メッセージに密接に関係します。

『データベース・モニター: 形式』のセクションでは、それぞれのビューにどの物理列が使用されるか、およびどのような情報が入るかについても説明しています。ビューを使用すると、モニターから抽出できる情報を識別することができます。これらの行は、複数の異なるビューで定義されます。これらのビューは、システムと一緒には出荷されないので、必要な場合には、ユーザーが作成しなければなりません。ビューは SQL DDL を使って作成できます。列記述の説明は、それぞれの図の後の表にあります。

データベース・モニターの例

System i ナビゲーター・インターフェースは、データベース・モニターを使ってパフォーマンス・モニター・データを収集し、分析するための強力なツールを提供します。しかし、データベース・モニター・ファイルを独自に分析することもできます。

ユーザーが SQL ステートメントを使用するアプリケーション・プログラムを所有していて、これらの照会を分析して、パフォーマンスの調整を行いたいとします。パフォーマンス分析の最初のステップは、データの収集です。次の例では、データベース・モニターの開始 (STRDBMON) および データベース・モニターの終了 (ENDDBMON) コマンドを使用してデータを収集し分析する方法を示します。パフォーマンス・データは、ユーザーの現行ジョブ内で実行中のアプリケーションから LIB/PERFDATA に収集されます。次の順序でパフォーマンス・データを収集し、その分析の準備を行います。

1. STRDBMON FILE(LIB/PERFDATA) TYPE(*DETAIL)。このテーブルがまだ存在していない場合、コマンドは、QSYS/QAQQDBMN にあるスケルトン・テーブルからこのファイルを作成します。
2. ユーザーのアプリケーションを実行します。
3. ENDDBMON
4. SQL DDL を使って LIB/PERFDATA に対するビューを作成します。ビューの作成は、必須ではありません。すべての情報は、STRDBMON コマンドで指定した基本テーブルに常駐します。ビューはデータを表示するための簡単な方法を提供するにすぎません。

これでデータ分析の準備が整いました。次の例では、このデータの使用法についてのアイデアのいくつかを提示します。物理ファイルおよび論理ビュー形式について厳密に学習し、収集するすべてのデータを理解して、ユーザーのアプリケーションに最適の情報を提供する照会を作成できるようになる必要があります。

関連情報

データベース・モニターの開始 (STRDBMON) コマンド

データベース・モニターの終了 (ENDDBMON) コマンド

データベース・モニター・パフォーマンス分析の例 1:

ユーザーの SQL アプリケーション内のどの照会をテーブル走査を使用して実施するかを決定します。完全な情報は、2 つのビュー、 QQQ1000 と QQQ3000 を結合することによって得られます。前者には、 SQL ステートメントについての情報が入っており、後者にはテーブル走査を行う照会についてのデータが入っています。

次の SQL 照会を使用することができます。

```
| SELECT (B.End_Timestamp - B.Start_Timestamp) AS TOT_TIME, A.System_Table_Schema, A.System_Table_Name,
|    A.Index_Advised, A.Table_Total_Rows, C.Number_Rows_Returned, A.Estimated_Rows_Selected,
|    B.Statement_Text_Long
|    FROM LIB.QQQ3000 A, LIB.QQQ1000 B, LIB.QQQ3019 C
|    WHERE A.Join_Column = B.Join_Column
|    AND A.Join_Column = C.Join_Column
```

この照会の出力例を以下の表に示します。この例のかぎは、結合基準です。

```
 WHERE A.Join_Column = B.Join_Column
   AND A.Join_Column = C.Join_Column
```

多くの照会についてのデータがテーブル LIB/PERFDATA 内に複数の行として入っています。ある単一照会についてのデータがこのテーブル内の 10 以上の行に入っていることは、珍しいことではありません。論理ビューを定義してからそれらのビューを 1 つに結合することによって、ユーザーは、1 つの照会または一群の照会のすべてのデータを 1 つにまとめることができます。列 QQJFLD は、あるジョブ内のすべての照会を一意的に識別します。列 QQUCNT は、照会レベルで固有です。論理ビューの文脈で参照される場合、これら 2 つの組み合わせによって、照会の方式が照会ステートメント情報に連結されます。

表 27. テーブル走査を実施した SQL 照会の出力

ライブラリー名	テーブル名	合計行数	推奨索引	戻された行数	TOT_TIME	ステートメント・テキスト
LIB1	TBL1	20000	Y	10	6.2	SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'A'
LIB1	TBL2	100	N	100	0.9	SELECT * FROM LIB1/TBL2
LIB1	TBL1	20000	Y	32	7.1	SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'B' AND FLD2 > 9000

照会が SQL を使用しない場合、SQL 情報行 (QQQ1000) は作成されません。 このため、LIB/PERFDATA 内のどの行がどの照会に関係しているかを判別するのがより困難になります。 SQL の使用時、行 QQQ1000 には、対応する照会のモニター行に一致する実際の SQL ステートメント・テキストが入っています。 SQL によってのみ、このステートメント・テキストを取り込むことができます。 OPNQRYF コマンドを使用して実行する照会の場合、OPNID パラメーターが取り込まれるので、これを使用して行を照会に結び付けることができます。 OPNID は、行 QQQ3014 の列 Open_Id に入っています。

データベース・モニター・パフォーマンス分析の例 2:

どの SQL アプリケーションでテーブル走査が実施されたかを示した前の例と同じように、次の例ではテーブル走査を実施するすべての照会を示します。

```
| SELECT (D.End_Timestamp - D.Start_Timestamp) AS TOT_TIME, A.System_Table_Schema, A.System_Table_Name,
|      A.Table_Total_Rows, A.Index_Advised,
|      B.Open_Id, B.Open_Time,
|      C.Clock_Time_to_Return_All_Rows, C.Number_Rows_Returned,
|      D.Result_Rows, D.Statement_Text_Long
|    FROM LIB.QQQ3000 A INNER JOIN LIB.QQQ3014 B
|      ON (A.Join_Column = B.Join_Column
|      LEFT OUTER JOIN LIB.QQQ3019 C
|      ON (A.Join_Column = C.Join_Column)
|      LEFT OUTER JOIN LIB.QQQ1000 D
|      ON (A.Join_Column = D.Join_Column)
```

この例で、テーブル走査を実行したすべての照会の出力を以下の表に示します。

注: テーブル QQQ1000 から選択された列は、照会が SQL を使用して実行されなかった場合には、NULL のデフォルト値を戻します。この例では、文字データ用のデフォルト値がブランクであり、数値データ用のデフォルト値がアスタリスク (*) であると想定しています。

表 28. テーブル走査を実施したすべての照会の出力

ライブラリー名	テーブル名	合計行数	推奨索引	照会 OPNID	ODP オープン時間	クロック時間	戻されたレコード数	戻された行数	TOT_TIME	ステートメント・テキスト
LIB1	TBL1	20000	Y		1.1	4.7	10	10	6.2	SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'A'
LIB1	TBL2	100	N		0.1	0.7	100	100	0.9	SELECT * FROM LIB1/TBL2
LIB1	TBL1	20000	Y		2.6	4.4	32	32	7.1	SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'A' AND FLD2 > 9000
LIB1	TBL4	4000	N	QRY04	1.2	4.2	724	*	*	*

SQL ステートメント・テキストが不要な場合、テーブル QQQ1000 への結合は不要です。 QQQ3014 および QQQ3019 行のデータから合計時間と選択された行数を判別することができます。

データベース・モニター・パフォーマンス分析の例 3:

次のステップでは、テーブル走査データについてさらに進んで分析を行います。直前の例には、推奨索引という表題の列がありました。 この列の 'Y' (yes) は、索引を使用してデータにアクセスした方が照会のパフォーマンスが向上する可能性があるという Query 最適化プログラムからのヒントです。 索引が推奨されている照会の場合、照会によって選択される行数がテーブル内の行の合計数に比べて少ないことに注目して

ください。 これは、テーブル走査が最適ではないことがあるというもう 1 つの徴候です。 最後に、実行時間が長い照会は、パフォーマンス・チューニングによってパフォーマンス向上の可能性が高いことを強調しています。

次の論理ステップでは、索引を推奨した最適化プログラムのヒントについて考察します。この目的のために、次の照会を使用することにします。

```
| SELECT A.System_Table_Schema, A.System_Table_Name,
|      A.Index_Advised, A.Index_Advised_Columns,
|      A.Index_Advised_Columns_Count, B.Open_Id,
|      C.Statement_Text_Long
|    FROM LIB.QQQ3000 A INNER JOIN LIB.QQQ3014 B
|      ON (A.Join_Column = B.Join_Column)
|      LEFT OUTER JOIN LIB.QQQ1000 C
|      ON (A.Join_Column = C.Join_Column)
|    WHERE A.Index_Advised = 'Y'
```

最初の例に若干の変更が加えられています。 まず、選択する列が変更されています。最も重要なことは、Query 最適化プログラムによって推奨された索引を作成するときに使用可能なキー列のリストが入っている列 Index_Advised_Columns を選択することです。 2 番目に、照会選択は、最適化プログラムが索引を作成 (A.Index_Advised = 'Y') するよう推奨しているそれらのテーブル走査照会への出力を制限します。以下の表に結果がどのようになるかを示します。

表 29. 推奨キー列を含む出力

ライブラリー名	テーブル名	推奨索引	推奨キー列	推奨 1 次キー	照会 OPNID	ステートメント・テキスト
LIB1	**TBL1**	**Y**	**FLD1**	**1**		SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'A'
LIB1	**TBL1**	**Y**	**FLD1, FLD2**	**1**		SELECT * FROM LIB1/TBL1 WHERE FLD1 = 'B' AND FLD2 > 9000
LIB1	**TBL4**	**Y**	**FLD1, FLD4**	**1**	**QRY04**	

ここで、最適化プログラムが推奨するように永続索引を作成することが意味があるかどうかを決定する必要があります。この例では、3 つの照会がそれぞれ FLD1 の 1 次キー列すなわち、左端のキー列を使用するので、LIB1/TBL1 について 1 つの索引を作成すると、これらの照会すべてに十分に答えます。 キー列 FLD1、FLD2 を指定して、LIB1/TBL1 について 1 つの索引を作成することによって、2 番目の照会のパフォーマンスのさらなる向上の可能性があります。 推奨された索引を作成するかどうかを決定する際には、これらの照会の実行頻度およびテーブルについての追加の索引の保守のオーバーヘッドを考慮する必要があります。

FLD1、FLD2 について永続索引を作成する場合、次のステップの順序は、次のようになります。

1. パフォーマンス・モニターを再度開始します。
2. アプリケーションを再実行します。
3. パフォーマンス・モニターを終了します。
4. データを再評価します。

3 つの推奨索引照会がもはやテーブル走査を実施することはなくなりそうです。

追加のデータベース・モニターの例:

パフォーマンス・モニター統計から情報を抽出する方法その他のアイデアおよび例を次に示します。これらの例では、データが LIB/PERFDATA に収集され、文書化されたビューが作成済みであると想定しています。

1. 何個の照会が動的再計画を実施中ですか ?

```
SELECT COUNT(*)
  FROM   LIB.QQQ1000
  WHERE  Dynamic_Replan_Reason_Code <> 'NA'
```

2. ステートメント・テキストの内容および動的再計画の理由は何ですか ?

```
SELECT Dynamic_Replan_Reason_Code, Statement_Text_Long
  FROM   LIB.QQQ1000
  WHERE  Dynamic_Replan_Reason_Code <> 'NA'
```

   注: 動的再計画の理由コードの定義については、列 Dynamic_Replan_Reason_Code の説明を参照する必要があります。

3. LIB1/TBL1 について何個の索引が作成済みですか ?

```
SELECT COUNT(*)
  FROM   LIB.QQQ3002
  WHERE  System_Table_Schema = 'LIB1'
    AND  System_Table_Name = 'TBL1'
```

4. LIB1/TBL1 について作成されるすべての索引に使用されるキー列はどれですか、また、関連する SQL ステートメント・テキストは何ですか ?

```
SELECT A.System_Table_Schema, A.System_Table_Name,
     A.Index_Advised_Columns, B.Statement_Text_Long
   FROM LIB.QQQ3002 A, LIB.QQQ1000 B
   WHERE A.Join_Column = B.Join_Column
     AND A.System_Table_Schema = 'LIB1'
     AND A.System_Table_Name = 'TBL1'
```

   注: この照会は、SQL を使用して実行された照会からのキー列のみを示します。

5. LIB1/TBL1 について作成されるすべての索引に使用されるキー列はどれですか、また、関連した SQL ステートメント・テキストまたは照会のオープン ID は何でしたか ?

```
SELECT A.System_Table_Schema, A.System_Table_Name, A.Index_Advised_Columns,
     B.Open_Id, C.Statement_Text_Long
   FROM LIB.QQQ3002 A INNER JOIN LIB.QQQ3014 B
     ON (A.Join_Column = B.Join_Column)
   LEFT OUTER JOIN LIB.QQQ1000 C
     ON (A.Join_Column = C.Join_Column)
   WHERE A.System_Table_Schema LIKE '%'
     AND A.System_Table_Name = '%'
```

   注: この照会は、システム上のすべての照会からのキー列を示します。

6. どのタイプの SQL ステートメントが実行中ですか ? どちらが最も頻繁に実行されますか ?

```
SELECT CASE Statement_Function
     WHEN 'O' THEN 'Other'
     WHEN 'S' THEN 'Select'
     WHEN 'L' THEN 'DDL'
     WHEN 'I' THEN 'Insert'
     WHEN 'U' THEN 'Update'
   ELSE 'Unknown'
   END, COUNT(*)
   FROM LIB.QQQ1000
   GROUP BY Statement_Function
   ORDER BY 2 DESC
```

7. どの SQL 照会が最も時間を消費しますか ? どのユーザーがこれらの照会を実行中ですか ?

```
SELECT (End_Timestamp - Start_Timestamp), Job_User,
     Current_User_Profile, Statement_Text_Long
   FROM LIB.QQQ1000
   ORDER BY 1 DESC
```

8. どの照会が最も時間を消費しますか ?

```
SELECT (A.Open_Time + B.Clock_Time_to_Return_All_Rows),
     A.Open_Id, C.Statement_Text_Long
   FROM LIB.QQQ3014 A LEFT OUTER JOIN LIB.QQQ3019 B
     ON (A.Join_Column = B.Join_Column)
   LEFT OUTER JOIN LIB.QQQ1000 C
     ON (A.Join_Column = C.Join_Column)
   ORDER BY 1 DESC
```

注: この例は、詳細なデータが収集されたことを前提としています (STRDBMON TYPE(*DETAIL))。

9. 論理的に一緒にしてグループにした各 SQL 照会のデータを使用する、すべての SQL 照会用のデータを示します。

```
SELECT A.*
   FROM LIB.PERFDATA A, LIB.QQQ1000 B
   WHERE A.QQJFLD = B.Join_Column
```

注: これは、関心のあるデータを読みやすい形式に形式設定する報告書内で使用されます。たとえば、すべての理由コード列は、報告書によって理由コードの定義を印刷するように拡張することができます (すなわち、物理列 QQRCOD = 'T1' は、照会されたテーブルについての索引が存在しないため、テーブル走査が実行されたことを意味します)。

10. 順序付けでキーの長さが 2000 バイトを超えているかまたは 120 を超えるキー列があったために、一時テーブルを使用して実施された照会がいくつありますか ?

```
SELECT COUNT(*)
   FROM LIB.QQQ3004
   WHERE Reason_Code = 'F6'
```

11. どの SQL 照会が再使用不可 ODP を使用して実施されましたか ?

```
SELECT B.Statement_Text_Long
   FROM LIB.QQQ3010 A, LIB.QQQ1000 B
   WHERE A.Join_Column = B.Join_Column
     AND A.ODP_Implementation = 'N';
```

12. 照会管理プログラムによって停止されたすべての照会の見積時間は、どれだけですか ?

```
SELECT Estimated_Processing_Time, Open_Id
   FROM LIB.QQQ3014
   WHERE Stopped_By_Query_Governor = 'Y'
```

注: この例は、詳細なデータが収集されたことを前提としています (STRDBMON TYPE(*DETAIL))。

13. 見積時間が実際の時間を超えているのは、どの照会ですか ?

```
SELECT A.Estimated_Processing_Time,
     (A.Open_Time + B.Clock_Time_to_Return_All_Rows),
     A.Open_Id, C.Statement_Text_Long
   FROM LIB.QQQ3014 A LEFT OUTER JOIN LIB.QQQ3019 B
     ON (A.Join_Column = B.Join_Column)
   LEFT OUTER JOIN LIB.QQQ1000 C
     ON (A.Join_Column = C.Join_Column)
   WHERE A.Estimated_Processing_Time/1000 >
     (A.Open_Time + B.Clock_Time_to_Return_All_Rows)
```

注: この例は、詳細なデータが収集されたことを前提としています (STRDBMON TYPE(*DETAIL))。

14. UNION を実行する照会があり、それに対して PTF を適用する場合。 UNION を実行する照会がある場合には、PTF を適用してください。照会のどれかがこの関数を実行しますか ?

```
SELECT COUNT(*)
  FROM   QQQ3014
  WHERE  Has_Union = 'Y'
```

注: 結果が 0 よりも大きい場合、PTF を適用する必要があります。

15. 読者がシステム管理者であり、次のリリースへのアップグレードを計画しています。 2 つのリリースからのデータを比較したいと思っています。
    - 現行リリース上のアプリケーションからデータを収集し、このデータを LIB/CUR_DATA に保管します。
    - 次のリリースに移行します。
    - 新規リリース上のアプリケーションからデータを収集し、このデータを別のテーブル LIB/NEW_DATA に保管します。
    - 結果を比較するプログラムを作成します。データの相関を行うため 2 つのテーブル内の行間でステートメント・テキストの比較が必要になります。

System i ナビゲーターを使用した詳細モニターの使用

詳細モニターは、System i ナビゲーターのインターフェースから扱えます。詳細 SQL パフォーマンス・モニターは、STRDBMON データベース・モニターの System i ナビゲーター・バージョンで、ネイティブ・インターフェースにあります。

このモニターを開始するには、System i ナビゲーター・ツリーのデータベース部分の下にある「SQL パフォーマンス・モニター」を右クリックし、「**新規**」 → 「**モニター**」を選択します。このモニターは詳細なデータをリアルタイムでハード・ディスクに保管し、その結果を分析するために、一時停止したり終了する必要はありません。また、モニターによって収集されたデータに基づいて、Visual Explain の実行を選択することもできます。このモニターはデータをリアルタイムで保管するので、システムのパフォーマンスに影響を与えることがあります。

詳細モニターの開始

System i ナビゲーター・インターフェースから、詳細モニターを開始できます。

このモニターを開始するには、System i ナビゲーター・ツリーのデータベース部分の下にある「SQL パフォーマンス・モニター」を右クリックし、「**新規**」 → 「**SQL パフォーマンス・モニター**」を選択します。「モニター・ウィザード」で、「**詳細**」を選択します。

詳細モニターを作成する場合は、キャプチャーする情報をフィルタリングすることができます。

照会の推定最短実行時間
: 指定した時間を超える照会を含めるには、このオプションを選択します。数値を選択し、時間の単位を指定します。

ジョブ名
: 特定のジョブ名によりフィルタリングする場合、選択します。フィールドにジョブ名を指定します。 ID 全体を指定するか、ワイルドカードを使用できます。たとえば、「QZDAS*」と指定すると、名前が「QZDAS」から開始するジョブ全件を検索します。

ジョブ・ユーザー
: ジョブ・ユーザーによりフィルタリングする場合、選択します。フィールドにユーザー ID を指定します。ID 全体を指定するか、ワイルドカードを使用できます。たとえば、「QUSER*」と指定すると、名前の先頭が「QUSER」であるユーザー ID 全件を検索します。

現行ユーザー
ジョブの現行ユーザーによりフィルタリングする場合、選択します。フィールドにユーザー ID を指定します。ID 全体を指定するか、ワイルドカードを使用できます。たとえば、「QSYS*」と指定すると、名前が「QSYS」から開始するユーザー全件を検索します。

インターネット・アドレス
インターネットへのアクセスによりフィルタリングする場合、選択します。フォーマットは xxx.xxx.xxx.xxx になります。例: 5.5.199.199。

表にアクセスする照会のみ
一定の表を使用する照会のみによりフィルタリングする場合、選択します。「**参照**」をクリックして、含める表を選択します。リストから表を除去するには、表を選択して「**除去**」を選択します。最大で 10 個の表名を指定できます。

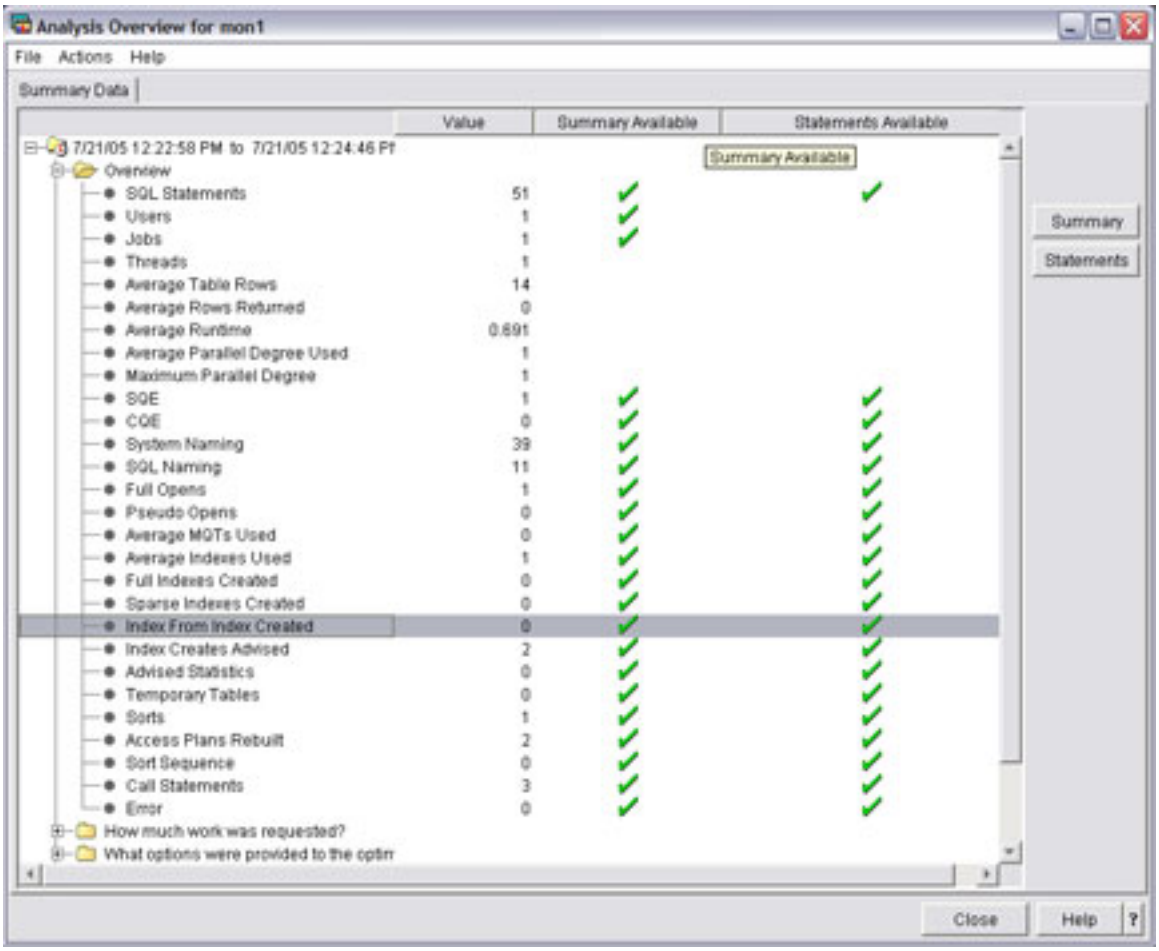
モニターするアクティビティー
ユーザー生成の照会のモニター出力、またはユーザー生成の照会とシステム生成の照会の両方のモニター出力を収集するために選択します。

モニターするジョブを選択するか、すべてのジョブをモニターするよう選択できます。システム上で、モニターのインスタンスを複数、同時に実行することができます。すべてのジョブをモニターする場合は、最大 10 個の詳細モニターを作成できます。すべてのジョブの情報を収集する場合、モニターは先に開始済みのジョブ、または、モニターが作成された後に開始された新規ジョブの情報を収集します。「**選択されたジョブ**」リストでジョブを選択、削除することにより、このリストを編集できます。

詳細モニター・データの分析

SQL パフォーマンス・モニターには、モニター・データを分析するのに使用できる定義済みの報告書が幾つか準備されています。

こうした報告書を表示するには、モニターを右クリックしてから「**分析 (Analyze)**」を選択します。この情報を表示するのにモニターを終了させる必要はありません。



「分析の概要」ダイアログで、概説情報を表示するか、または以下のカテゴリーのいずれかを選択することができます。

- どのくらいの処理が要求されましたか?
- どのオプションが最適化プログラムに供給されましたか?
- 最適化プログラムはどのインプリメンテーションを使用しましたか?
- どのタイプの SQL ステートメントが要求されましたか?
- 各種情報
- I/O 情報

「アクション」メニューから、以下の要約事前定義レポートのいずれかを選択できます。

ユーザー要約

各ユーザーごとに 1 行の要約情報が含まれます。個々の行がそのユーザーに対するすべての SQL 活動を要約しています。

ジョブ要約

各ジョブごとに 1 行の情報が含まれます。個々の行がそのジョブに対するすべての SQL 活動を要約しています。この情報は、システムのどのジョブが SQL の中で最高負荷のユーザーであるか、およびどれがパフォーマンス調整の候補となるかを示すために使用されます。これによって、ユーザーは、システム全体をモニターすることなく、個別のジョブに対して詳細パフォーマンス・モニターを開始し、より詳細な情報を得ることが必要になることがあります。

操作要約

SQL 操作の各タイプごとに 1 行の要約情報が含まれます。個々の行がそのタイプの SQL 操作に対するすべての SQL 活動を要約しています。この情報は、使用された SQL ステートメントのタイプの高水準の徴候を提供します。たとえば、アプリケーションが主として読み取り専用であるか、あるいは多くの更新、削除、または挿入活動があるかなどです。この情報を使用して、特定パフォーマンス調整手法を試みることができます。たとえば、多くの挿入活動が行われる場合には、ブロック化因数を大きくする OVRDBF コマンドの使用または QDBENCWT API の使用が適切と考えられます。

プログラム要約

SQL 操作を実行した各プログラムごとに 1 行の情報が含まれます。個々の行がそのプログラムに対するすべての SQL 活動を要約しています。この情報は、どのプログラムが最大で最もコストのかかる SQL ステートメントを使用しているかを識別するのに使用することができます。これらのプログラムは、パフォーマンス調整の潜在的な候補になります。プログラム名が使用可能となるのは、その SQL ステートメントがコンパイル済みプログラムに組み込まれている場合だけであることに注意してください。 ODBC、JDBC、または OLE DB を介して出された SQL ステートメントは、それがプロシージャー、関数、またはトリガーに起因するものでないかぎり、プログラム名はブランクになります。

加えて、「要約」列の下に緑色のチェックが表示される場合、その行を選択し、「**要約**」をクリックして、その行タイプに関する情報を表示することができます。要約報告書の詳細については、「**ヘルプ**」をクリックしてください。ステートメントによって編成される情報を表示するには、「**ステートメント**」をクリックしてください。

モニター・データの比較

System i ナビゲーターを使用して、2 つ以上のモニターのデータ・セットを比較することができます。

System i ナビゲーターは、2 つのタイプの比較を備えています。最初のタイプは、モニターまたはスナップショットのおおまかな比較機能を提供する単純比較です。 2 番目のタイプは詳細比較です。単純比較は、詳細比較が役立つかどうか判別できるようにモニターまたはスナップショットに関する十分なデータを提供します。

単純比較を起動するには、「**System i ナビゲーター**」 → 「**システム名**」 → 「**SQL パフォーマンス・モニター**」に移動します。 1 つ以上のモニターを右クリックして、「**比較**」を選択します。

詳細比較を起動するには、「詳細比較 (Detailed Comparison)」タブを選択します。

「詳細比較 (Detailed Comparison)」ダイアログで、比較するデータ・セットに関する情報を指定できます。

名前　比較したいモニターの名前。

スキーマ・マスク

比較を無視したい任意の名前を選択します。たとえば、以下のシナリオを考慮してください。テスト・スキーマで実行中で、最適化されているアプリケーションがあります。ここで、それを実動スキーマに移動し、そこでどのように実行されるかを比較したいとします。比較におけるステートメントは、テスト・スキーマのステートメントが ″TEST″ を使用し、実動スキーマのステートメントが ″PROD″ を使用すること以外は同一です。 2 つのモニターのステートメントが同一に表示されるように、スキーマ・マスクを使用して、最初のモニターの ″TEST″ と 2 番目のモニターの ″PROD″ を無視することができます。

次の時間より長く実行したステートメント

比較の対象となるステートメントの最小実行時間。

最小差のパーセント
ステートメントを等しいとみなすかどうかを判別する、比較されている 2 つのステートメントのキー属性における最小差。たとえば、最小差のパーセントに 25% を選択する場合、そのキー属性に 25% 以上の差がある一致ステートメントのみが戻されます。

「**比較**」をクリックすると、一致するステートメントを求めて両方のモニターが走査されます。一致が検出されると、各インプリメンテーションのキー属性の比較について横並びで表示されます。

「比較出力 (Comparison output)」ダイアログで、「**ステートメントの表示**」をクリックすることにより、モニターに含まれているステートメントを表示します。ステートメントを選択し、「**Visual Explain**」をクリックして Visual Explain を実行することもできます。

一致が検出されると、各インプリメンテーションのキー属性の比較について横並びで表示されます。

モニターでのステートメントの表示

詳細モニターに含まれている SQL ステートメントを表示することができます。

「SQL パフォーマンス・モニター」ウィンドウの任意の詳細モニターを右クリックし、「**ステートメントの表示**」を選択します。

フィルター・オプションは、関心のある特定の分野にフォーカスを当てる方法を提供します。

最長の実行に対する最短実行時間
少なくとも 1 つの長い個別照会インスタンスの実行時間を持つ照会にフィルターします。

この日時以降に実行された照会
最近実行された照会にフィルターします。

これらのオブジェクトを使用または参照する照会
項目を、指定されたテーブルまたは索引 (いずれも複数可) を参照または使用する項目に限定する方法を提供します。

特定のタイプを含む SQL ステートメント
SQL テキストそのものに対するワイルドカード検索機能を提供します。これは特定のタイプの照会を見つけるのに便利です。たとえば、FETCH FIRST 文節を持つ照会は、'fetch' を指定することによって見つかります。検索では使いやすさのために大/小文字を区別しません。たとえば、ストリング 'FETCH' は、検索ストリング 'fetch' と同じ項目を検出します。

フィルター・オプションは複数指定できます。複数のフィルターを指定する場合、各フィルターの候補項目は個別に計算され、すべての候補リストに存在する項目のみが表示されます。それでたとえば、「**最長の実行に対する最短実行時間**」オプションと「**この日時以降に実行された照会**」オプションを指定した場合、指定された日時以降に実行される、最短実行時間を持つ項目が表示されます。

関連資料

107 ページの『Query 最適化プログラムの索引アドバイザー』
Query 最適化プログラムは、照会内の行選択を分析して、デフォルト値に基づいて、永続索引の作成がパフォーマンスを向上するかどうかを判別します。永続索引が有利になる可能性があると最適化プログラムが判別した場合には、示された索引の作成に必要なキー列を戻します。

モニターのインポート

System i ナビゲーターを使用することにより、その他のいくつかのインターフェースを使って収集されたモニター・データをインポートすることができます。

データベース・モニターの開始 (STRDBMON) コマンドを使用して作成されたモニターは、System i ナビゲーターによって自動的に登録され、System i ナビゲーターによって表示されるモニターのリストに含まれます。

モニター・データをインポートするには、「**SQL パフォーマンス・モニター**」を右クリックして、「**インポート**」を選択します。モニターをインポートしたら、データを分析できます。

Query 最適化プログラムの索引アドバイザー

Query 最適化プログラムは、照会内の行選択を分析して、デフォルト値に基づいて、永続索引の作成がパフォーマンスを向上するかどうかを判別します。永続索引が有利になる可能性があると最適化プログラムが判別した場合には、示された索引の作成に必要なキー列を戻します。

最適化プログラムは、1 つの追加 2 次キー列に加えて、1 次キー列の任意の組み合わせについて基数索引プローブを実行することができます。したがって、最初の 2 次キー列が最も慎重に選択された 2 次キー列であることが重要になります。最適化プログラムは、残りの 2 次キー列を指定して、基数索引走査を使用することになります。基数索引走査は基数索引プローブほどは迅速ではないものの、これによって、選択されるキーの数を減らすことができます。したがって、十分に選択された 2 次キー列を組み込む必要があります。

2 次キー列に実際にどれを選択するかを決定し、索引の作成時にこれらのキー列を組み込むかどうかを決定するのは、ユーザー自身です。索引の構築時には、1 次キー列を左端のキー列にし、その後にユーザーが選択した任意の 2 次キー列が続くようにする必要があり、また、これらには、選択順位による優先順位を付ける必要があります。

注: 推奨された索引を作成し、照会を再度実行した後は、Query 最適化プログラムがこの推奨された索引を使用しないことを選択することもできます。 CQE 最適化プログラムが索引を提案するときは、選択基準のみを考慮し、結合、順序付け、およびグループ化の基準は含まれません。 SQE 最適化プログラムが索引の提案を行う際には、選択、結合、順序付け、およびグループ化の基準が含まれます。

索引アドバイザー情報へのアクセスはさまざまな方法で行えます。たとえば、次のような方法があります。

- System i ナビゲーターの索引アドバイザー・インターフェース
- SQL パフォーマンス・モニターの Show ステートメント
- Visual Explain インターフェース
- データベース・モニター・ビュー 3020 の照会 - 推奨索引

関連資料

113 ページの『Visual Explain から使用可能な情報の概説』
以下の多くのタイプの情報を表示するために Visual Explain を使用できます。

289 ページの『データベース・モニター・ビュー 3020 - 推奨索引 (SQE)』
データベース・モニター QQQ3020 の SQL 論理ビュー形式を表示します。

106 ページの『モニターでのステートメントの表示』
詳細モニターに含まれている SQL ステートメントを表示することができます。

索引アドバイザー情報の表示

System i ナビゲーターを使用して、最適化プログラムから索引アドバイザー情報を表示できます。

System i ナビゲーターは、QSYS2/SYSIXADV システム・テーブルで見つかった情報を表示します。

索引アドバイザー情報を表示するには、次のステップを実行します。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開します。
3. 作業対象になるデータベースを右クリックし、「**索引アドバイザー**」 → 「**索引アドバイザー**」を選択します。

スキーマ・オブジェクトまたはテーブル・オブジェクトを右クリックすることで、特定のスキーマ、または表の索引アドバイザー情報を検索することも可能です。

情報を 1 度表示すると、索引を作成するためにリストから選択したり、リストから助言された索引を除去したり、または、リスト全体を消去したりすることができます。さらに索引を右クリックして「SQL の表示」を選択し、索引作成ステートメントを指定して「SQL スクリプトの実行」セッションを起動することもできます。

データベース・マネージャーの推奨索引のシステム・テーブル:

このトピックでは、推奨索引のシステム・テーブルについて説明します。

表 30. SYSIXADV システム・テーブル

列名	システム列名	データ・タイプ	説明
TABLE_NAME	TBNAME	VARCHAR(258)	索引が推奨されるテーブル
TABLE_SCHEMA	DBNAME	CHAR(10)	テーブルを含むスキーマ
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	索引が推奨されるシステム・テーブル名
PARTITION_NAME	TBMEMBER	CHAR(10)	索引のパーティションの詳細
KEY_COLUMNS_ADVISED	KEYSADV	VARCHAR(16000)	推奨索引の列名
LEADING_COLUMN_KEYS	LEADKEYS	VARCHAR(16000)	先頭に来る、順序に依存しないキー。再配列され、推奨される索引を満たす COLUMNS_ADVISED フィールドの先頭のキー。
INDEX_TYPE	INDEX_TYPE	CHAR(14)	基数 (デフォルト) または EVI
LAST_ADVISED	LASTADV	TIMESTAMP	この行の最終更新時刻
TIMES_ADVISED	TIMESADV	BIGTINT	この索引が推奨された回数
ESTIMATED_CREATION_TIME	ESTTIME	INT	索引作成のための見積もり秒数
REASON_ADVISED	REASON	CHAR(2)	索引が推奨されたされた理由 (コード化)
LOGICAL_PAGE_SIZE	PAGESIZE	INT	索引の推奨ページ・サイズ
MOST_EXPENSIVE_QUERY	QUERYCOST	INT	照会の実行時間 (秒)
AVERAGE_QUERY_ESTIMATE	QUERYEST	INT	照会の平均実行時間 (秒)
TABLE_SIZE	TABLE_SIZE	BIGINT	索引推奨時のテーブル内の行数
NLSS_TABLE_NAME	NLSSNAME	CHAR(10)	索引で使用する NLSS テーブル
NLSS_TABLE_SCHEMA	NLSSDBNAME	CHAR(10)	NLSS テーブルのスキーマの名前
MTI_USED	MTIUSED	BIGINT	最適化プログラムによってこの特定の保守済み一時索引 (MTI) が使用された回数。最適化プログラムは、永続索引が作成されると、一致する MTI の使用を停止します。

表 30. SYSIXADV システム・テーブル (続き)

列名	システム列名	データ・タイプ	説明
MTI_CREATED	MTICREATED	INTEGER	最適化プログラムによってこの特定の保守済み一時索引 (MTI) が作成された回数。MTI は、システム IPL を通して持続するわけではありません。
LAST_MTI_USED	LASTMTIUSE	TIMESTAMP	この特定の保守済み一時索引 (MTI) が、照会のパフォーマンス向上のために最適化プログラムによって最後に使用された時刻を表すタイム・スタンプ。「MTI の最終使用 (MTI Last Used)」フィールドはブランクの場合があります。これはこの推奨に完全に一致する MTI が、この索引推奨を生成した照会によって使用されたことがないことを示しています。
AVERAGE_QUERY_ESTIMATE_MICRO	QRYMICRO	BIGINT	索引推奨を駆動した照会の平均実行時間 (マイクロ秒単位)
EVI_DISTINCT_VALUES	EVIVALS	INTEGER	推奨された EVI 索引を作成する際に使用する推奨値。CREATE INDEX SQL ステートメントの WITH n DISTINCT VALUES 文節内では、この値は n。

索引アドバイザー列

「索引アドバイザー」ウィンドウで使用される列を表示します。

表 31. 「索引アドバイザー」ウィンドウで使用される列

列名	説明
索引が推奨されたテーブル (Table for Which Index was Advised)	最適化プログラムは、このテーブルに永続索引を作成することを推奨しています。これは、テーブルの長い名前です。テーブルが照会されたものの、照会のパフォーマンス向上に使用できる既存の永続索引がなかったために、推奨が生成されました。
スキーマ	テーブルを含むスキーマまたはライブラリー
短縮名	索引が推奨されるシステム・テーブル名
区画	索引のパーティションの詳細。次の値が指定可能です。 • <blank> (全パーティション用 (For all partitions) を意味します) • 各パーティション用 (For Each Partition) • パーティションの具体的な名前
推奨キー (Keys Advised)	推奨索引の列名。列名の順序は重要です。先頭に来る、順序に依存しないキー情報が、配列は変更可能であることを示していない限り、名前は、CREATE INDEX SQL ステートメントと同じ順序でリストされる必要があります。
先頭に来る、順序に依存しないキー (Leading Keys Order Independent)	先頭に来る、順序に依存しないキー。再配列され、推奨される索引を満たす COLUMNS_ADVISED フィールドの先頭のキー。
推奨索引タイプ (Index Type Advised)	基数 (デフォルト) または EVI
照会の使用に対する最後の推奨 (Last Advised for Query Use)	この索引が照会用に最後に推奨された時刻を表すタイム・スタンプ。

表 31. 「索引アドバイザー」ウィンドウで使用される列 (続き)

列名	説明
照会の使用に対する推奨回数 (Times Advised for Query Use)	この索引が推奨された累積の回数。このカウントは、一致する永続索引が作成されると、増加を停止する必要があります。推奨の行は、ユーザーが除去するまでこのテーブルにとどまります。
推定索引作成時間 (Estimated Index Creation Time)	この索引の作成に必要な推定時間。
推奨理由 (Reason advised)	索引が推奨された理由。次の値が指定可能です。 行選択。 順序付け/グループ化 行選択と順序付け/グループ化
推奨論理ページ・サイズ (Logical Page Size Advised) (KB)	この索引の作成時に CREATE INDEX SQL ステートメントの PAGESIZE キーワードに使用される推奨ページ・サイズ。
コスト最高の照会推定 (Most Expensive Query Estimate)	この索引推奨を生成した実行時間が最長の照会の実行時間 (秒)。
照会推定の平均 (microseconds) (Average of Query Estimates (マイクロ秒))	この索引推奨を生成したすべての照会の平均実行時間 (秒)。
推奨時のテーブルの行 (Rows in Table when Advised)	この索引が最後に推奨された時のテーブル内の行数。
推奨 NLSS テーブル (NLSS Table Advised)	索引推奨を生成した照会によって使用中のソート・シーケンス・テーブル。ソート・シーケンスの詳細の参照先:
推奨 NLSS スキーマ (NLSS Schema Advised)	ソート・シーケンス・テーブルのスキーマ。
使用された MTI (MTI Used)	最適化プログラムによってこの特定の保守済み一時索引 (MTI) が使用された回数。
作成された MTI (MTI Created)	最適化プログラムによってこの特定の保守済み一時索引 (MTI) が作成された回数。 MTI は、システム IPL を通して持続するわけではありません。
MTI の最終使用 (MTI Last Used)	この特定の保守済み一時索引 (MTI) が、照会のパフォーマンス向上のために最適化プログラムによって最後に使用された時刻を表すタイム・スタンプ。「MTI の最終使用 (MTI Last Used)」フィールドはブランクの場合があります。これはこの推奨に完全に一致する MTI が、この索引推奨を生成した照会によって使用されたことがないことを示しています。
EVI 独自の値 (EVI Distinct Values)	推奨された EVI 索引を作成する際に使用する推奨値。CREATE INDEX SQL ステートメントの WITH n DISTINCT VALUES 文節内では、この値は n。

データベース・モニター・ビュー 3020 の照会 - 推奨索引

索引アドバイザーの情報は、データベース・モニター・ビュー 3020 - 推奨索引 (SQE) 内に見つけることができます。

このアドバイザーの情報は、列 QQIDXA、QQIDXK および QQIDXD に保管されます。 QQIDXA 列に 'Y' の値が入っていると、最適化プログラムは、列 QQIDXD に示すキー列を使用して索引を作成するよう知らせます。この索引を作成する意図は、照会のパフォーマンスを向上することです。

列 QQIDXD に入っているキー列のリストには、最適化プログラムによって提示された 1 次キー列および 2 次キー列がリストされています。 1 次キー列は、対応する照会選択に基づいて選択されるキーの数を顕著に削減する列です。 2 次キー列は、選択されるキーの数を顕著に削減することもあり、または削減しないこともある列です。

列 QQIDXK には、列 QQIDXD にリストされている推奨された 1 次キー列の数が入っています。これらは、左端に推奨されたキー列です。残りのキー列は、 2 次キー列と見なされ、照会に基づいて予期される選択順にリストされます。たとえば、QQIDXK に 4 の値が入っていて、QQIDXD が 7 つのキー列を指定しているとします。この場合、 QQIDXK に指定された先頭の 4 つのキー列が 1 次キー列になります。残りの 3 つのキー列は、推奨された 2 次キー列です。

推奨索引の圧縮

何回にもわたって、索引アドバイザーは、同じテーブルに対して複数の異なる索引を推奨します。これらの推奨された索引を照会に最も合った索引に圧縮することができます。

1. 「**System i ナビゲーター**」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開します。
3. 処理するデータベースを右クリックして、「**索引アドバイザー**」->「**推奨索引の圧縮 (Condense Advised Indexes)**」を選択します。

Visual Explain を使用した照会の実行の表示

Visual Explain ツールを System i ナビゲーターと共に使用して、SQL ステートメントの実装をグラフィックに表示する照会グラフを作成することができます。このツールは、静的および動的 SQL ステートメントの両方についての情報を参照するために使用できます。 Visual Explain は、
SELECT、INSERT、UPDATE、および DELETE のタイプの SQL ステートメントをサポートします。

照会は、実行の際に生じる別個の操作を表す一連のアイコンによるグラフを使用して表示されます。このグラフは、メイン・ウィンドウに表示されます。このペインの下部に、このグラフの基になる SQL ステートメントが示されます。Visual Explain を「SQL スクリプトの実行」から開始する場合、「**最適化プログラム・メッセージ (Optimizer messages)**」タブをクリックすると、最適化プログラムが発行したデバッグ・メッセージを表示できます。右側のペインに、照会属性が表示されます。

Visual Explain は詳細 SQL パフォーマンス・モニターに保管された照会の実装を、視覚的に表示するために使用できます。しかし、これは記憶域常駐モニターの結果のテーブルは処理しません。

Visual Explain の開始

Visual Explain ツールを起動するには、2 つの方法があります。最初の方法は、System i ナビゲーターを使用するもので、これが一般的です。 2 番目は、Visual Explain (QQQVEXPL) API を使用する方法です。

Visual Explain は System i Navigatorの次の任意のウィンドウから開始できます。

- 「**SQL スクリプトの実行**」ウィンドウに SQL ステートメントを入力する。ステートメントを選択して、コンテキスト・メニューから「**Explain**」 を選択する。または **Visual Explain** のメニューから、「**実行および Explain**」を選択する。
- 使用可能な SQL パフォーマンス・モニターのリストを展開する。詳細 SQL パフォーマンス・モニターを右マウス・ボタンでクリックし、「**ステートメントを表示**」オプションを選択します。フィルタリング情報を選択し、「ステートメントのリスト」ウィンドウでステートメントを選択します。「**Visual Explain の実行**」をクリックします。また、「SQL スクリプトの実行」からも SQL パフォーマンス・モニターを開始できます。「**モニター**」メニューから、「**SQL パフォーマンス・モニターの開始 (Start SQL Performance monitor)**」を選択します。

- 「**データベース**」を右クリックして、「**ジョブの SQL 詳細 (SQL Details for Jobs)**」を選択し、ジョブの SQL 詳細機能を開始します。リストからジョブを選択して、「**SQL ステートメント**」をクリックします。下のペインにその SQL が表示されると、「**Visual Explain の実行**」をクリックして Visual Explain を開始できます。
- SQL プラン・キャッシュを右クリックして、「**ステートメントを表示**」を選択します。フィルタリング情報を選択し、「ステートメントのリスト」ウィンドウでステートメントを選択します。「**Visual Explain の実行**」をクリックします。
- 使用可能な SQL プラン・キャッシュ・スナップショットのリストを展開します。スナップショットを右クリックし、「**ステートメントを表示**」を選択します。フィルタリング情報を選択し、「ステートメントのリスト」ウィンドウでステートメントを選択します。「**Visual Explain の実行**」をクリックします。
- SQL プラン・キャッシュ・イベント・モニターのリストを展開します。ステートメントを選択し、「**Visual Explain の実行**」をクリックします。

「SQL スクリプト」の実行から「Visual Explain」を実行する場合、オプションが 3 つあります。

Visual Explain のみ
このオプションでは、実際に照会を実行せずに、照会を Explain します。表示されるデータは、Query 最適化プログラムの見積もりを示します。

注: System i ナビゲーターの「SQL スクリプトを実行」から、Visual Explain の「Explain」オプションのみを使用する場合、一部の照会ではエラー・コード 93 を受け取ります。それは、Visual Explain で表示するには複雑過ぎることを示しています。「実行および Explain」オプションを選択すると、これを回避できます。

実行および Explain
「実行および Explain」を選択すると、ダイアグラムが表示される前に、システムにより照会が実行されます。このオプションを選択した場合、かなりの時間がかかる場合がありますが、より完全で正確な情報が表示されます。

実行中の Explain
実行に長くかかる照会の場合、照会の実行中に Visual Explain を開始するよう選択できます。Visual Explain ダイアグラムを最新表示することで、照会の進捗を表示できます。

加えて、System i ナビゲーターを使用しても作成されなかったデータベース・モニター・テーブルに関して、System i ナビゲーターによって Explain できます。最初に、System i ナビゲーターにデータベース・モニター・テーブルをインポートしなければなりません。こうするには、SQL パフォーマンス・モニターを右クリックし、「**インポート (Import)**」オプションを選択します。パフォーマンス・モニターに、名前 (System i ナビゲーター内で認識される名前) と、データベース・モニターのテーブルの修飾名を指定します。モニターのタイプには、必ず「詳細 (Detailed)」を選択してください。「詳細 (Detailed)」は、ファイル・ベースのモニター (STRDBMON) を表し、「要約 (Summary)」は、記憶域常駐モニター (Visual Explain ではサポートされていない) を表します。モニターがインポートされたら、手順に従って、System i ナビゲーター内から Visual Explain を開始します。

Visual Explain 情報を SQL パフォーマンス・モニターとして保管できます。これは、照会を「SQL スクリプトの実行」から開始し、後で比較するためにその情報を保管する場合に役に立ちます。「**ファイル**」メニューから、「**パフォーマンス・モニターとして保管 (Save as Performance monitor)**」を選択します。

関連情報

Visual Explain (QQQVEXPL) API

Visual Explain から使用可能な情報の概説

以下の多くのタイプの情報を表示するために Visual Explain を使用できます。

情報には、以下が含まれます。

- 照会グラフでのそれぞれの操作 (アイコン) の情報
- 費用のかかるアイコンの強調表示
- 統計アドバイザーおよび索引アドバイザー
- 照会述部実施
- グラフの基本情報および詳細情報

照会グラフでのそれぞれの操作 (アイコン) の情報

前述のように、グラフ内のアイコンは照会の実施の際に生じる操作を表しています。 アイコン同士を接続している矢印によって、操作の順序が示されています。操作を処理するのに並列処理が使用されている場合、矢印は 2 つになります。 時折、最適化プログラムは 1 つの照会内の別個の操作でハッシュ・テーブルを「共用」し、その照会のライン同士が交差してしまうことがあります。

アイコンを選択すると、その操作に関する情報を表示できます。情報は、右側のペインの「**属性**」テーブルに表示されます。環境に関する情報を表示するには、アイコンをクリックして、「**アクション**」メニューから「**照会環境の表示 (Display query environment)**」を選択します。最後に、アイコンを右クリックして「**ヘルプ**」すると、このアイコンに関する情報をさらに表示できます。

費用のかかるアイコンの強調表示

Visual Explain を使用して照会内の問題点 (費用のかかるアイコン) を強調表示できます。 Visual Explain では、処理時間または行数を基にした 2 つのタイプの強調表示される費用のかかるアイコンがあります。「**表示**」メニューから、「**費用のかかるアイコンの強調表示 (Highlight expensive icons)**」を選択すると、アイコンを強調表示できます。

統計アドバイザーおよび索引アドバイザー

照会の実施中、最適化プログラムは統計を作成または更新する必要があるかどうか、また索引によって照会をより速く実行できるかどうかを判別できます。 こうした推奨は、Visual Explain から統計アドバイザーおよび統計アドバイザーを使用すると表示できます。「**アクション**」メニューから「**アドバイザー**」を選択すると、アドバイザーを開始できます。さらに、直接アドバイザーから統計の収集を開始したり、索引を作成したりすることも可能です。

照会述部実施

Visual Explain を使用すると、照会述部の実施を表示できます。 述部実施は、アイコンの隣の青色の正符号 (+) で表されます。 アイコンを右クリックして「**展開 (Expand)**」を選択すると、この表示を展開できます。別のウィンドウにオープンすることもできます。この操作に関する属性を表示するには、アイコンをクリックします。表示を縮小するには、ウィンドウ内で右クリックして、「**縮小 (Collapse)**」を選択してください。この機能は、V5R3 以降のシステムでしか使用できません。

また、最適化プログラムはルック・アヘッド述部生成を使用して、結合のランダム入出力コストを最小化できます。 この方法で述部を強調表示には、「**表示**」メニューから「**LPG の強調表示 (Highlight LPG)**」を選択します。

グラフの基本情報および完全情報

また Visual Explain では、基本および完全という 2 つの異なるビューによって情報が提供されます。基本ビューには、SQL ステートメントの実行を理解するのに必要なアイコンのみが表示され、照会実施のメイン・フローを理解するのに絶対不可欠というわけではない予備操作または中間操作の一部は表示されません。完全ビューには、実行ツリーのフローを詳細に描写したより多くのアイコンが表示される可能性があります。「**オプション**」メニューから「**グラフ詳細 (Graph Detail)**」を選択し、「**基本**」または「**完全**」のどちらかを選択して、グラフ詳細を変更できます。デフォルトのビューは「**基本**」です。**完全**ビューの詳細をすべて表示するには、「グラフ詳細 (Graph Detail)」を「**完全**」に変更し、Visual Explain を閉じてから、照会をもう一度実行する必要があることに注意してください。「グラフ詳細 (Graph Detail)」の設定値は、ずっと持続します。

Visual Explain および使用可能な別のオプションについては、 Visual Explain のオンライン・ヘルプを参照してください。

Visual Explain ダイアグラムの最新表示

実行時間が長い照会の場合は、照会を完了する前に、Visual Explain のグラフを、実行時統計情報で最新表示することができます。さらに最新表示によって、画面の右側に表示されたアイコンの属性セクションにある該当する情報も更新されます。「**最新表示**」オプションを使用するには、「SQL スクリプト実行」ウィンドウから「**実行中に Explain**」を選択する必要があります。

ダイアグラムを最新表示するには、「**表示**」メニューから「**最新表示**」を選択します。あるいは、ツールバーの「**最新表示**」ボタンをクリックします。

関連資料

107 ページの『Query 最適化プログラムの索引アドバイザー』
Query 最適化プログラムは、照会内の行選択を分析して、デフォルト値に基づいて、永続索引の作成がパフォーマンスを向上するかどうかを判別します。永続索引が有利になる可能性があると最適化プログラムが判別した場合には、示された索引の作成に必要なキー列を戻します。

プラン・キャッシュを使用したパフォーマンスの最適化

プラン・キャッシュには、データベース中で実行される SQE 照会に関する豊富な情報が含まれます。その内容は、System i ナビゲーター GUI インターフェースによって表示できます。プラン・キャッシュの特定の部分も変更できます。

さらに、ユーザーがプログラマチックにプラン・キャッシュを処理できるプロシージャーも用意されています。これらのプロシージャーは、SQL CALL ステートメントを使用して呼び出すことができます。

プラン・キャッシュのインターフェースは、システム上のデータベース照会操作にウィンドウを提供します。プラン・キャッシュのインターフェースは、「**System i ナビゲーター**」 → 「**システム名**」 → 「**データベース**」の下にあります。

SQL プラン・キャッシュ・フォルダーの中には、2 つのフォルダー、「SQL プラン・キャッシュ・スナップショット」と「SQL プラン・キャッシュ・イベント・モニター」が含まれています。

「SQL プラン・キャッシュ・スナップショット」フォルダーをクリックすると、それまでに収集されたすべてのスナップショットのリストが表示されます。スナップショットは、「新規スナップショット」が要求された際にプラン・キャッシュから生成されるデータベース・モニター・ファイルです。これは、SQL パフォーマンス・モニターのリストとまったく同じように扱うことができます。スナップショットには、従来の SQL パフォーマンス・モニターと同じ分析機能が存在します。

| 「SQL プラン・キャッシュ・イベント・モニター」をクリックすると、定義済みのすべてのイベントのリストが表示されます。プラン・キャッシュ・イベント・モニターは、定義されている場合、キャッシュから除去される際にプランからデータベース・モニター情報を生成します。リストには、現在、アクティブなイベント、および完了したイベントが含まれます。スナップショットと同様に、イベント・モニターはデータベース・モニター・ファイルです。その結果、SQL パフォーマンス・モニターおよびスナップショットに使用できる分析機能と同じ機能が、イベント・ファイルでも使用できます。

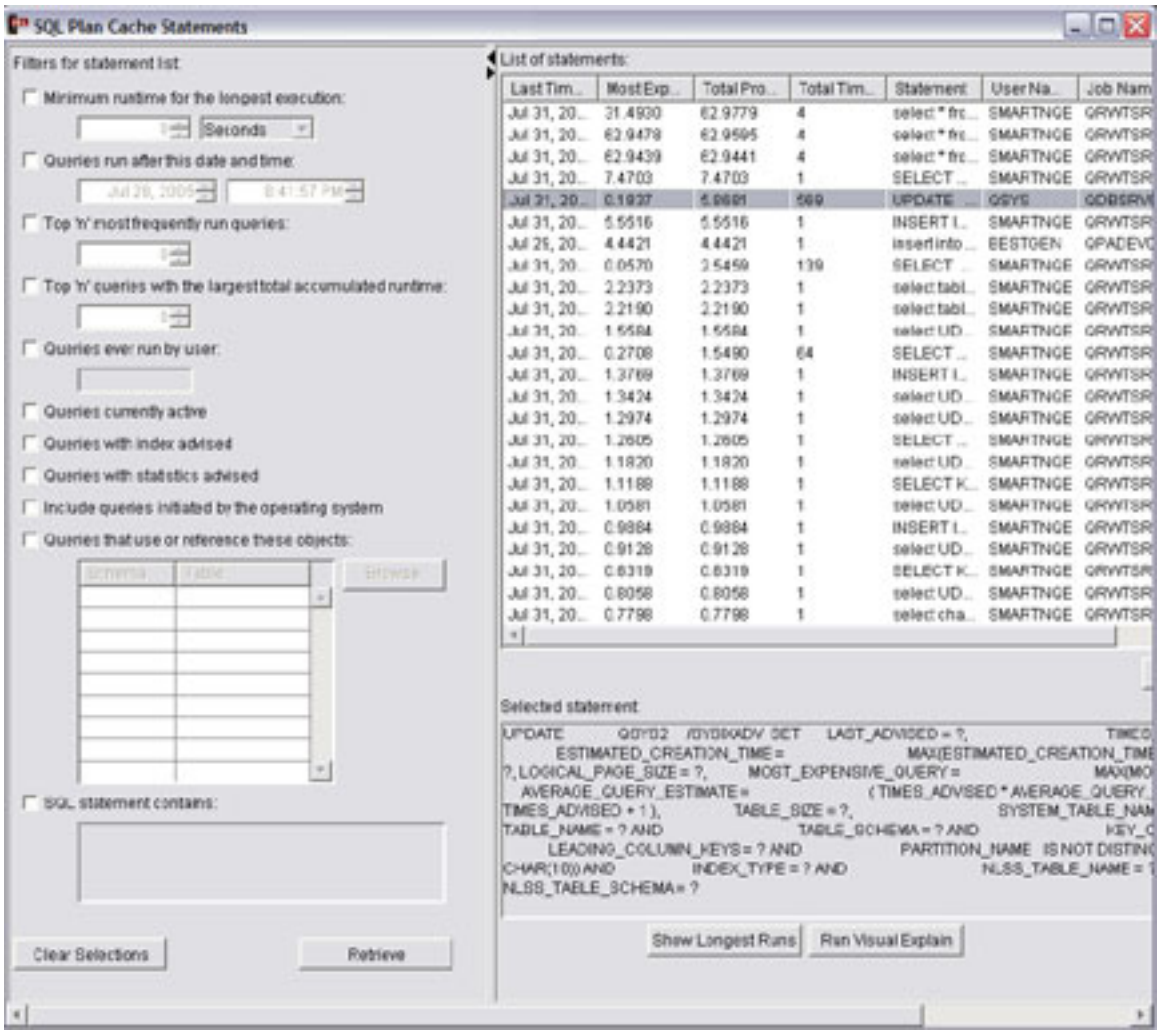
| プラン・キャッシュはアクティブに変わるキャッシュです。したがって、これにはタイムリーな情報が入っているということを覚えておくことが大切です。長期間にわたる情報に関心がある場合は、イベント・モニターを定義して、時間の経過とともにキャッシュから除去されるあらゆるプランに関して、必ず情報がキャプチャーされるようにする必要があります。あるいは、傾向および使用率の高い時間をキャプチャーするために、プラン・キャッシュの定期的なスナップショットの実行方式の実施を検討することもできます。プラン・キャッシュに関するこのセクションの後半で IBM が提供する、呼び出し可能 SQL プロシージャーの説明を参照してください。

| **関連概念**

| 5 ページの『プラン・キャッシュ』
| プラン・キャッシュは、SQE によって最適化された照会用のアクセス・プランを含むリポジトリーです。

| SQL プラン・キャッシュ - ステートメントを表示

| 「SQL プラン・キャッシュ」アイコンを右クリックすることにより、データベースの現行プラン・キャッシュの表示を切り替えるための一連のオプションが表示されます。「**SQL プラン・キャッシュ**」 → 「**ステートメントを表示**」オプションを選択すると、フィルター機能のある画面が表示されます。この画面は、システム上の現行プラン・キャッシュの直接ビューを提供します。



画面にデータを表示するには、適用または最新表示アクションを実行 (プッシュ) する必要があります。表示される情報には、SQL 照会テキスト、照会の最終実行時刻、照会の中で最もコストの高い単一のインスタンスの実行、照会によって消費された合計処理時間、照会の合計実行回数、およびプラン項目を最初に作成したユーザーおよびジョブに関する情報が含まれます。さらに、平均実行時間、平均結果セット・サイズ、および平均一時ストレージ使用量を含む、実行あたりの複数の平均も含まれます。異常な実行を無視する平均である調整済みの平均処理時間があります。また、表示では、以前の照会の実行の結果をデータベース・エンジンが再利用し、その結果、照会全体の再実行を回避できるようになった回数 (ある場合) も示されます。最終的に、プランの数値 ID およびプラン・スコアも表示されます。

画面には、関心のある特定の基準をより迅速に隔離するためにユーザーが使用できるフィルター・オプションがあります。フィルターの指定は必須ではありませんが (デフォルトでは指定されない)、フィルターを追加すると結果を表示するための時間が短くなります。最も合計処理時間のかかるものが先頭に表示されるよう、戻される照会のリストが配列されます。配列したいリストの列見出しをクリックすることにより、結果を再配列することができます。もう一度クリックすると、配列が昇順から降順に切り替わります。 1 つ以上のプランを強調表示して右クリックすることによって、複数の可能なアクションを表示したメニューが表示されます。

プランに対して Visual Explain を実行することによって、照会プランの視覚的な描写を表示し、より詳細なパフォーマンス分析を提供することができます。このアクションを実行するには、1 つのプランのみが強調表示できます。

| 「**最長実行の表示**」は、その照会の最も長い実行インスタンス 10 個までの詳細を表示します。最長実行リスト内では、項目を右クリックして視覚的に説明するか、スナップショットに保管するか、除去することができます。これは、Visual Explain の特定の実行に関する情報を収集する場合に有用です。さらに、古い、または余分な実行を除去して、将来の実行を取り込む余地を作る方法も提供します。このアクションを実行するには、1 つのプランのみが強調表示できます。また、除去されたすべての実行は、リストから除去された実行を引き続き含むプランに対して、リストに表示されている内容、合計時間、実行の総数などにのみ影響を及ぼします。

| 「**アクティブ・ジョブ数の表示**」は、現在、プランを使用中のシステム上のジョブのリストを表示します。

| 「**ユーザー・ヒストリーを表示 (Show User History)**」は、そのプランを実行したすべてのユーザー ID、およびこのプランが最後に実行された時刻のリストを表示します。

| 「**SQL ステートメントの処理 (Work with SQL Statement)**」は、SQL ステートメントを表示したスクリプト・ウィンドウを表示します。これは、ステートメントを直接、処理し、調整する場合、または独自のウィンドウでステートメントのみを表示する場合に有用です。このアクションを実行するには、1 つのプランのみが強調表示できます。

| 「**新規に保管 (Save to New)**」を使用すれば、選択した項目のスナップショットを作成できます。

| 「**プラン**」アクションの下で、プランに対する特定の変更オプションを実行できます。「プラン・スコアを変更 (Change Plan Score)」を使用すれば、プランのスコアを特定の値に設定できます。プラン・スコアは、プランをキャッシュから除去すべき時期を決定するために使用されます。下位のスコア・プランは、上位のスコア・プランの前に除去されます。スコア・プランを高く設定することによって、プランは、より長い期間、キャッシュ内にとどまります。プラン・スコアを低い値に設定すると、プランは、設定しない場合よりも早く整理されます。削除を使用すれば、プランを即時にキャッシュから除去できます。通常の環境では、プランの属性を変更する必要はないはずです。通常のデータベース処理は、プランのエージングおよび整理を適切に実行します。これらの変更オプションは、大半の場合、分の分析および一般的な関心のためにツールとして提供されます。

| 表示ステートメント上の各プラン項目に与えられるユーザーおよびジョブ名の情報は、最初のプランの作成の際のユーザーおよびジョブ (完全な最適化が実行された時点のユーザー) であることに注意してください。これは必ずしも最後に照会を実行したユーザーと同じであるとは限りません。ただし、「最長実行」画面では、特定ユーザーおよびその個別の実行のジョブを表示します。

| フィルター・オプションは、関心のある特定の分野にフォーカスを当てる方法を提供します。

| **最長の実行に対する最短実行時間**
| 少なくとも 1 つの長い個別照会インスタンスの実行時間を持つ照会にフィルターします。

| **この日時以降に実行された照会**
| 最近実行された照会にフィルターします。

| **最も頻繁に実行した上位「n」個の照会**
| 最も頻繁に実行された照会を見つけます。

| **合計累積実行時間が最長の上位「n」個の照会**
| 上位リソース消費者を表示します。フィルターが指定されない場合、これはデフォルトで示される最初の n 項目と等しくなります。 n に値を指定することで、最初の項目の画面の取得パフォーマンスは向上します。ただし、表示される合計の項目 n に限定されます。

| **これまでに次のユーザーが実行したステートメント (Statements the following user has ever run)**
| 特定のユーザーが実行した照会のリストを表示する方法を提供します。このフィルターが指定され

る場合、結果の項目に表示されるユーザーおよびジョブ名の情報は引き続き、キャッシュ項目の開始元を反映します。これは必ずしも、フィルターで指定されたユーザーと同じであるとは限りません。

現在アクティブのステートメント (Statements that are currently active)
現在実行中または疑似クローズ・モードにある照会と関連したキャッシュ項目のリストを表示します。ユーザー・フィルターと同様、結果の項目に表示されるユーザーおよびジョブ名の情報は引き続き、キャッシュ項目の開始元を反映します。これは必ずしも、現在照会を実行しているユーザーと同じであるとは限りません (複数のユーザーが照会を実行している場合もあります)。

注: ジョブの現行 SQL (データベース・アイコンを右クリック) は、特定のジョブのアクティブな照会を表示するための代替手段です。

索引が推奨されたステートメント (Statements for which an index has been advised)
パフォーマンスを向上させるために最適化プログラムによって索引が推奨されている照会にリストを制限します。

統計が推奨されたステートメント (Statements for which statistics have been advised)
まだ収集されていない統計が最適化プログラムで役に立ったかもしれない照会 (それが収集された場合) にリストを制限します。最適化プログラムはこれらの統計をバックグラウンドで自動的に収集するため、通常このオプションは、何らかの理由で統計の収集を手動で制御したい場合でない限り、それほど重要ではありません。

オペレーティング・システムによって開始された照会の組み込み
要求を処理するために裏側でデータベースによって自動的に開始された「非表示」照会をリストに組み込みます。デフォルトで、リストにはユーザーによって開始された照会のみが組み込まれます。

次のオブジェクトを参照するステートメント (Statements that reference the following objects)
項目を、指定されたテーブルまたは索引 (いずれも複数可) を参照または使用する項目に限定する方法を提供します。

特定のタイプを含む SQL ステートメント
SQL テキストそのものに対するワイルドカード検索機能を提供します。これは特定のタイプの照会を見つけるのに便利です。たとえば、FETCH FIRST 文節を持つ照会は、'fetch' を指定することによって見つかります。検索では使いやすさのために大/小文字を区別しません。たとえば、ストリング 'FETCH' は、検索ストリング 'fetch' と同じ項目を検出します。

フィルター・オプションは複数指定できます。複数のフィルターを指定する場合、各フィルターの候補項目は個別に計算され、すべての候補リストに存在する項目のみが表示されます。それで例えば、「**最も頻繁に実行した上位 n 個の照会**」オプションと「**これまでに次のユーザーが実行したステートメント (Statements the following user has ever run)**」オプションを指定した場合、指定されたユーザーによってある時点で偶然実行された項目のうち、実行頻度がキャッシュ内で最高の項目が表示されます。必ずしも、ユーザーによって最も頻繁に実行された照会が表示されるわけではありません (それらの照会がキャッシュ内全体で偶然、最も頻繁に実行された照会ではない場合)。

SQL プラン・キャッシュ列

「SQL プラン・キャッシュ・ステートメント (SQL plan cache statements)」ウィンドウで使用される列を表示します。

表 32. 「SQL プラン・キャッシュ・ステートメント (SQL plan cache statements)」ウィンドウで使用される列

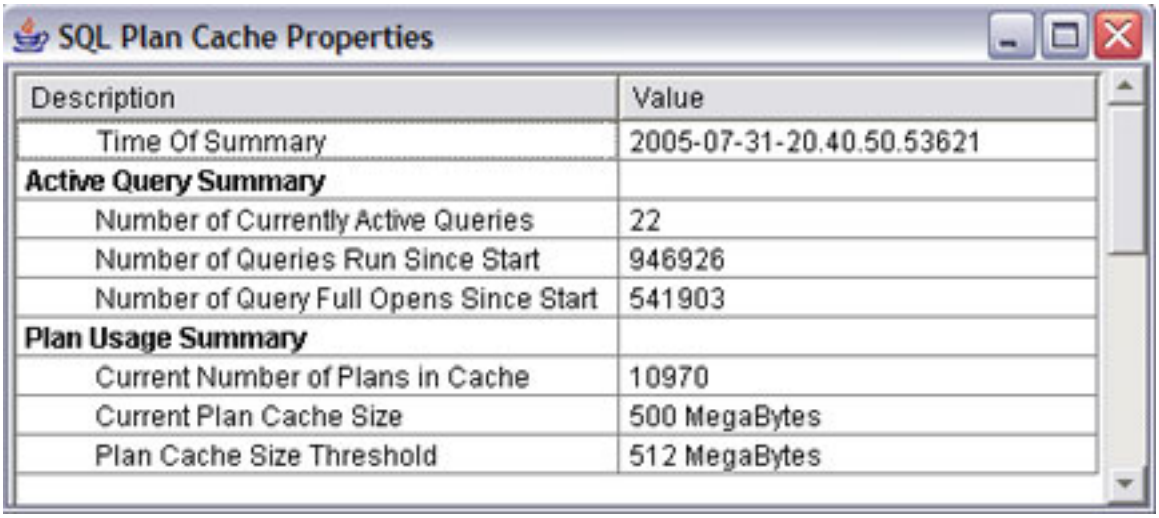
列名	説明
最後の実行時刻 (Last Time Run)	このステートメントが最後に実行された時刻を表示します。
コスト最高の時間 (秒) (Most Expensive Time (sec))	このステートメントの実行のうち最も長いものの時間。
処理時間の合計 (秒) (Total Processing Time (sec))	このステートメントのすべての実行が処理に要する合計時間 (秒単位)。
合計実行時間 (Total Times Run)	このステートメントが実行された回数の総数。
平均処理時間 (秒) (Average Processing Time (sec))	このステートメントが処理に要した実行あたりの平均時間 (秒単位)。
ステートメント	ステートメント・テキスト。
プラン作成ユーザー名 (Plan Creation User Name)	プランを作成したユーザー ID の名前。
ジョブ名	プランを作成したジョブの名前。
ジョブ・ユーザー	プランを作成したジョブを所有するユーザー ID の名前。
ジョブ番号	プランを作成したジョブのジョブ番号。
調整平均処理時間 (秒) (Adjusted Average Processing Time (sec))	平均の計算から異常な実行を除いた、このステートメントが処理に要した実行あたりの平均時間 (秒単位)。これによって、ステートメントの正常な状態に対して極めて例外的な単一の (またはわずかな) 実行を無視することによって、ステートメントの現実的な平均が得られます。
平均結果セット行 (Average Result Set Rows)	このステートメントが実行された際に戻される結果セット行の平均の行数。
使用された平均一時記憶域 (Average Temp Storage Used (MB))	このステートメントが実行された際に使用される一時記憶域の平均。
プラン・スコア (Plan Score)	キャッシュ内の他のプランとの相対関係における、このプランの格付け。他のプランに対して格付けがより高いプランは、より長い期間、キャッシュ内にとどまります。他のプランに対して格付けがより低いプランは、他のプランよりも早く、キャッシュから除去されます。
プラン ID (Plan Identifier)	プランの固有の数値 ID。
キャッシュに入れられた使用済み結果の合計 (Total Cached Results Used)	後のステートメントの実行で再使用された、ステートメントの前の実行からの結果セットの回数。
最適化時間 (秒) (Optimization Time (sec))	このステートメントの最適化に要した時間。
システムの名前	システムの名前。
リレーショナル・データベースの名前	リレーショナル・データベースの名前
QQJFLD	内部使用フィールド。

SQL プラン・キャッシュ・プロパティー

「SQL プラン・キャッシュ」 → 「プロパティー」 オプションは、キャッシュに関する高水準の情報、たとえばキャッシュ・サイズ、プランの数、発生した完全オープンおよび疑似オープンの数などを表示します。

この情報は、データベース・アクティビティー全体を表示するために使用できます。長時間トラッキングを行うと、1 日の中の、および 1 週間の中のデータベース使用率のピークとくぼみをより良く理解するのに役立つ傾向が提供されます。

プロパティーを右クリックし、「値を編集 (Edit Value)」を選択することによって、プラン・キャッシュの「プラン・キャッシュ・サイズしきい値 (Plan Cache Size Threshold)」プロパティーを編集することができます。通常の環境では、このプロパティー値で十分であり、変更は不要です。変更された場合、まず変更によるパフォーマンスの結果を慎重に評価する必要があります。この値は、IPL 時にデフォルトにリストアされることに注意してください。

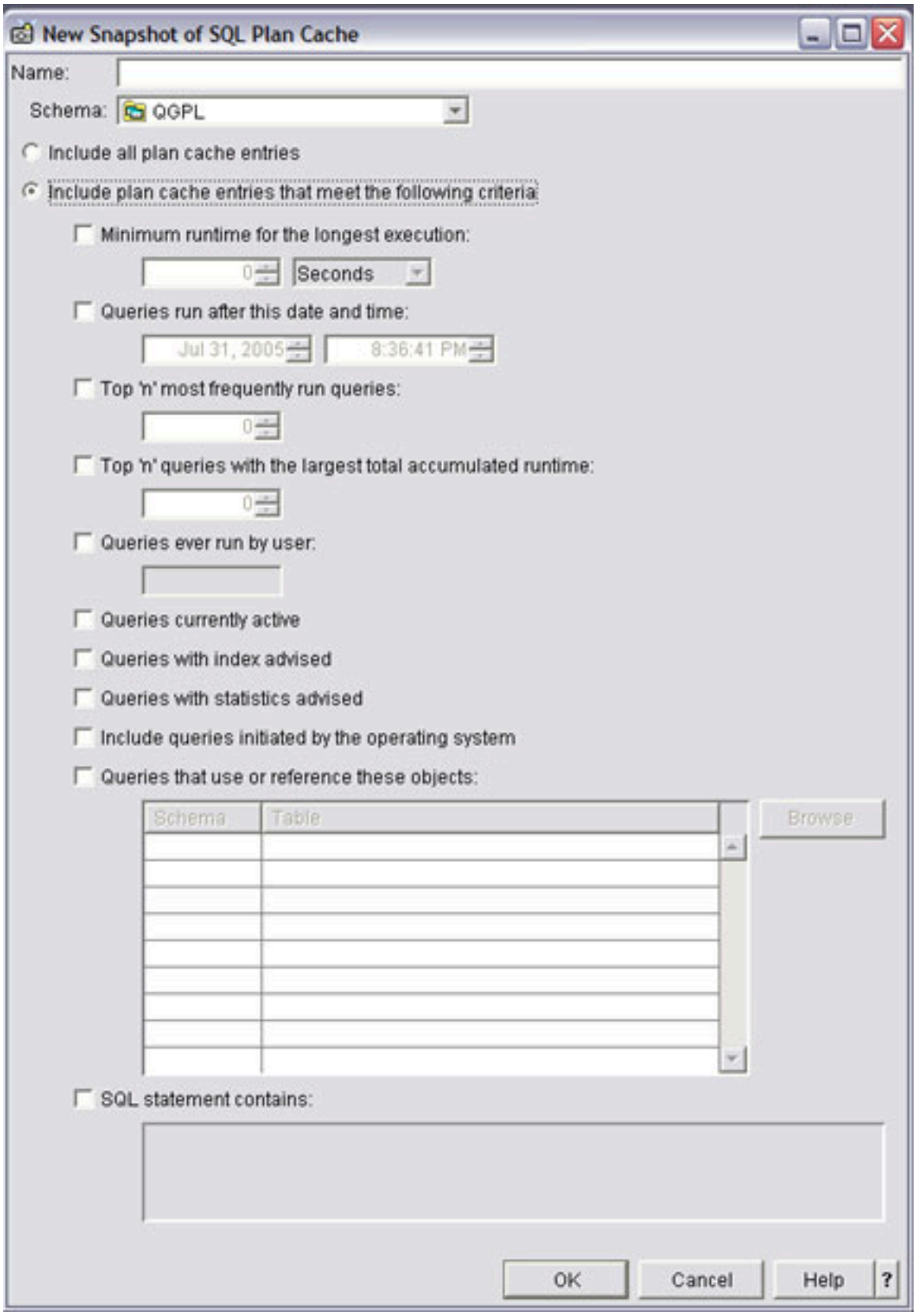


プロパティーを右クリックし、「**値を編集 (Edit Value)**」を選択することによって、プラン・キャッシュの一部のプロパティーを編集することができる場合があります。

SQL プラン・キャッシュ・スナップショット

「**新規**」 → 「**スナップショット**」 オプションで、プラン・キャッシュからスナップショットを作成できます。

「ステートメントを表示」の下のスナップショット・オプションとは異なり、これを使用すると最初に照会を表示せずにスナップショットを作成することができます。



「ステートメントを表示」画面のものと同じフィルター・オプションが提供されます。

| ストアード・プロシージャー qsys2.dump_plan_cache は、プラン・キャッシュからデータベース・モニター・ファイル出力 (スナップショット) を作成する方法として最も単純でプログラマチックな方法を提供します。 dump_plan_cache プロシージャーは、結果として生成されるデータベース・モニター・ファイルを識別するために、2 つのパラメーター、ライブラリー名、およびファイル名を取ります。ファイルが存在しない場合は作成されます。たとえば、ライブラリー QGPL のデータベース・パフォーマンス・モニター・ファイルにプラン・キャッシュをダンプするには、次のようにします。

```
CALL qsys2.dump_plan_cache('QGPL','SNAPSHOT1');
```

SQL プラン・キャッシュ・イベント・モニター

SQL プラン・キャッシュ・イベント・モニターは、プラン・キャッシュの変更を記録します。

System i ナビゲーター・インターフェースを介して、またはプロシージャーを直接呼び出して、SQL プラン・キャッシュ・イベント・モニターにアクセスできます。

SQL プラン・キャッシュ・イベントは、プランのモニター・レコードがプラン・キャッシュから除去される際にこれを取り込みます。イベント・モニターは、プランがキャッシュから除去されても、キャッシュで使用できる可能性があるすべてのパフォーマンス情報が必ず取り込まれるようにする際に有用です。イベント・モニター出力をプラン・キャッシュ・スナップショットと結合することによって、イベントの開始時から、スナップショットが取られるまで、キャッシュの複合ビューが表示されます。

イベント・モニターを使用すれば、表示ステートメント用に記述されたのと同じフィルター操作オプションを使用でき、最も頻繁に実行される Top 'n' ステートメントおよび累積実行時間が最大の Top 'n' ステートメントを例外として、スナップショットを作成します。ステートメントがキャッシュから除去されると、他のプランとの比較には参加しなくなるので、これらの 2 つの 'Top n' フィルターは、これらの整理されたプランに対して意味がありません。

SQL ストアード・プロシージャーによる SQL プラン・キャッシュへのアクセス

System i ナビゲーターは、プラン・キャッシュへのビジュアル・インターフェースを備えています。ただし、プラン・キャッシュは、同時に SQL CALL ステートメントを通じて呼び出すことができるストアード・プロシージャーを介してアクセスすることもできます。

次のプロシージャーを使用すれば、プログラムに基づいてプラン・キャッシュにアクセスでき、例えば、プラン・キャッシュ・キャプチャーのスケジューリングまたは、イベント・モニターの事前開始用に使用することができます。

qsys2.dump_plan_cache('lib','file')

このプロシージャーは、キャッシュの内容のスナップショット (データベース・モニター・ファイル) を作成します。また、結果として生成されるデータベース・モニター・ファイルを識別するために、2 つのパラメーター、ライブラリー名、およびファイル名を取ります。ファイルが存在しない場合は作成されます。ファイル名は、10 文字に制限されています。

例えば、ライブラリー QGPL の「SNAPSHOT1」と呼ばれるデータベース・パフォーマンス・モニター・ファイルにプラン・キャッシュをダンプするには、次のようにします。

```
  CALL qsys2.dump_plan_cache('QGPL','SNAPSHOT1');
```

qsys2.start_plan_cache_event_monitor('lib','file')

このプロシージャーは、イベント・モニターを開始して、プランがキャッシュから除去される際にプランをインターセプトし、指定されたデータベース・モニター・ファイルにパフォーマンス情報を生成します。ま

| た、結果として生成されるデータベース・モニター・ファイルを識別するために、2 つのパラメーター、ライブラリー名、およびファイル名を取ります。ファイルが存在しない場合は作成されます。当初、ファイルは、開始レコード ID 3018 (列 QQRID = 3018) によって、作成され、取り込まれます。制御は呼び出し元に戻されますが、イベント・モニターはアクティブなままです。ライブラリー QTEMP は許可されません。ファイル名は、10 文字に制限されています。イベント・モニターは次のうちの 1 つが発生するまでアクティブなままです。イベント・モニターがイベント・モニター終了プロシージャー呼び出しの 1 つによって終了する、System i ナビゲーター・インターフェースを使用して終了する、システムの IPL (初期プログラム・ロード) が発生する、または、指定されたデータベース・モニター・ファイルが削除される、あるいは、そうでなければ使用できなくなる。

| 例えば、イベント・モニターを開始して、ライブラリー QGPL の「PRUNEDP1」と呼ばれるデータベース・パフォーマンス・モニター・ファイルにプラン情報を配置するには、次のようにします。

```
| CALL qsys2.start_plan_cache_event_monitor('QGPL','PRUNEDP1');
```

qsys2.start_plan_cache_event_monitor('lib','file',monitorID)

| このプロシージャーは、イベント・モニターを開始して、プランがキャッシュから除去される際にプランを取り込み、データベース・モニター・ファイルにパフォーマンス情報を生成します。ライブラリー名、ファイル名、およびモニター ID の 3 つのパラメーターを取ります。ライブラリー名、およびファイル名は、結果として生成されるデータベース・モニター・ファイルを識別します。ファイルが存在しない場合は作成されます。当初、ファイルは、開始レコード ID 3018 によって、作成され、取り込まれます。モニター ID は、開始されたイベント・モニターの 10 文字の ID を含めるために、データベースによって設定された CHAR(10) 出力パラメーターです。制御はプロシージャー呼び出し元に戻されますが、イベント・モニターはアクティブなままです。ライブラリー QTEMP は許可されません。ファイル名は、10 文字に制限されています。イベント・モニターは次のうちの 1 つが発生するまでアクティブなままです。イベント・モニターがイベント・モニター終了プロシージャー呼び出しの 1 つによって終了する、System i ナビゲーター・インターフェースを使用して終了する、システムの IPL (初期プログラム・ロード) が発生する、または、指定されたデータベース・モニター・ファイルが削除される、あるいは、そうでなければ使用できなくなる。

| 例えば、イベント・モニターを開始して、ライブラリー QGPL の「PRUNEDPLANS1」と呼ばれるデータベース・パフォーマンス・モニター・ファイルにプラン情報を配置し、後で使用するためにモニター ID をホスト変数 HVmonid に取り込むには、次のようにします。

```
| CALL qsys2.start_plan_cache_event_monitor('QGPL','PRUNEDP1', :HVmonid);
```

qsys2.end_all_plan_cache_event_monitors()

| このプロシージャーは、GUI または start_plan_cache_event_monitor プロシージャーを通して開始されたすべてのアクティブ・プラン・キャッシュ・イベント・モニターを終了するために使用できます。パラメーターは取りません。

```
| CALL qsys2.end_all_plan_cache_event_monitors();
```

qsys2.end_plan_cache_event_monitor('monID')

| このプロシージャーは、所定のモニター ID 値によって識別される特定のイベント・モニターを終了するために使用できます。このプロシージャーは、start_plan_plan_event_monitor と連動して特定のイベント・モニターを終了します。

| Example:

```
| CALL qsys2.end_plan_cache_event_monitor('PLANC00001');
```

qsys2.change_plan_cache_size(sizeinMeg)

このプロシージャーは、プラン・キャッシュのサイズを変更するために使用できます。整数パラメーターは、プラン・キャッシュが設定される必要があるサイズをメガバイト単位で指定します。このサイズは、以降の IPL (初期プログラム・ロード) でデータベース・デフォルトにリセットされます。指定された値がゼロの場合は、プラン・キャッシュはデフォルト値にリセットされます。

例:

```
CALL qsys2.change_plan_cache_size(512);
```

qsys2.dump_plan_cache_properties('lib','file')

このプロシージャーは、キャッシュのプロパティーを含むファイルを作成します。また、結果として生成されるプロパティー・ファイルを識別するために、2 つのパラメーター、ライブラリー名、およびファイル名を取ります。ファイルが存在しない場合は作成されます。ファイル名は、10 文字に制限されています。ファイルの定義は、アーカイブ・ファイル qsys2/qdboppcgen に一致します。

例えば、ライブラリー QGPL の「PCPROP1」と呼ばれるファイルにプラン・キャッシュ・プロパティーをダンプするには、次のようにします。

```
CALL qsys2.dump_plan_cache_properties('QGPL','PCPROP1');
```

記憶域常駐データベース・モニターを使用した照会のモニター

記憶域常駐のデータベース・モニターは、データベース・パフォーマンスをモニターするためのもう 1 つの方法を提供するツールです。このツールは、SQL パフォーマンス・モニターのモニターのみを意図するもので、プログラマーやパフォーマンス分析者の役に立ちます。モニターは、API のセットの手助けを得て、データベース・モニター情報を取り、記憶域内でユーザーのためにそれらを管理します。この記憶域常駐のモニターは、結果のテーブル・サイズだけでなく、CPU のオーバーヘッドを低減します。

データベース・モニターの開始 (STRDBMON) は、パフォーマンス情報の収集時にシステム・リソースを拘束します。このオーバーヘッドの主因として、パフォーマンス情報が収集されるときにそれが直接データベース・テーブルに書き込まれることが考えられます。メモリー・ベースの収集モードは、パフォーマンスの結果をメモリー内で収集、管理することによって消費されるシステム・リソースを低減します。これによって、モニターは、システム全体のパフォーマンスへの (または個々の SQL ステートメントのパフォーマンスへの) 影響を最小にしてデータベースのパフォーマンス統計を収集することができます。

モニターは、STRDBMON モニターとほとんど同じ情報を収集しますが、パフォーマンス統計を記憶域内に保持します。ある程度の明細を犠牲にして、情報が同一の SQL ステートメント用に要約されるので、収集した情報量が削減されます。この目的は、統計をできるだけ迅速に記憶域に取り込むことにありますが、一方、データの操作や変換はパフォーマンス・データが分析用の結果テーブルにダンプされるまで遅らせます。

記憶域常駐のモニターが、STRDBMON モニターに取って代わるわけではありません。モニターでは明細が失われて SQL ステートメントの十分な分析ができなくなる状況が生じます。このような場合には、今までどおり STRDBMON モニターを使用する必要があります。

記憶域常駐のモニターは、パフォーマンス情報を一連の行様式に結合し、蓄積して記憶域内でこのデータを管理します。これは、固有の SQL ステートメントごとに、そのステートメントの実行のたびに情報が累積され、明細情報が収集されるのは、最も費用のかかるステートメントの実行に対してだけであることを意味します。

各 SQL ステートメントは、モニターによって下記に従って識別されます。
- ステートメント名
- パッケージ (またはプログラム)
- 準備済みステートメントが入っているスキーマ
- 使用するカーソル名

純粋な動的ステートメントの場合、ステートメント・テキストは、別のスペースに保持され、そのステートメントの識別は、ポインターを介して内部的に操作されます。

このシステムは、各 SQL 操作のテーブルへの書き込みのオーバーヘッドをかなり減らす一方で、記憶域内の統計を保持するために、詳細をいくらか犠牲にします。できるだけ速く統計を記憶域に収集することが目的であれば、後ほどデータをテーブルにダンプするときにデータ操作またはデータ変換のための時間を取っておいてください。

記憶域常駐のモニターは、パフォーマンス情報を新規の行様式に結合し蓄積することにより、記憶域内でこのデータを管理します。したがって、固有の SQL ステートメントごとに、そのステートメントの実行のたびに情報が累積され、システムは最も費用のかかるステートメントの実行に対しては詳細情報を収集するだけです。

各 SQL ステートメントは、モニターによりステートメント名、準備済みステートメントを含むパッケージ (またはプログラム) およびスキーマ、および使用されるカーソル名で識別されます。純粋な動的ステートメントの場合、
- ステートメント・テキストは別個のスペースに保持されます。
- ステートメント ID は、ポインターを介して内部的に処理されます。

記憶域常駐のモニターの API サポート

API のセットは、記憶域常駐のモニターのサポートを使用可能にします。 API は、次の活動のそれぞれをサポートします。
- 新規モニターの開始
- 統計をテーブルにダンプ
- モニター・データを記憶域からクリア
- モニター状況を照会
- 新規モニターの終了

新規モニターを開始すると、システムがモニターする各ジョブのローカル・アドレス・スペースに情報が格納されます。ステートメントが完了するたびに、システムは、ローカル・ジョブ・スペースから共通システム・スペースに情報を移動します。この共通システム・スペースに入りきらないほど多くのステートメントが実行される場合、システムは、最近実行されていないステートメントを除去します。

関連情報

Start SQL Database Monitor (QQQSSDBM) API

Dump SQL Database Monitor (QQQDSDBM) API

Clear SQL Database Monitor Statistics (QQQCSDBM) API

Query SQL Database Monitor (QQQQSDBM) API

End SQL Database Monitor (QQQESDBM) API

記憶域常駐のデータベース・モニターの外部 API の説明

記憶域常駐のデータベース・モニターは、一組の API によって制御されます。

表 33. 外部 API の説明

API	説明
SQL データベース・モニターの開始 (QQQSSDBM)	SQL モニターを開始する API
SQL データベース・モニター統計の消去 (QQQCSDBM)	SQL モニター記憶域をクリアする API
SQL データベース・モニターのダンプ (QQQDSDBM)	SQL モニターの内容をテーブルにダンプする API
SQL データベース・モニターの終了 (QQQESDBM) API	SQL モニターを終了する API
SQL データベース・モニターの照会 (QQQQSDBM)	データベース・モニターの状況を照会する API

記憶域常駐のデータベース・モニターの外部テーブルの説明

記憶域常駐のデータベース・モニターは、STRDBMON モニター・プログラムが使用する複数の論理ファイルを持つ単一のテーブルを使用する代わりに、独自のテーブルのセットを使用します。記憶域常駐のデータベース・モニター・テーブルは、STRDBMON モニターの推奨論理ファイルに密接に対応します。

表 34. 外部テーブルの説明

モニター・テーブル	説明
QAQQQRYI	照会 (SQL) 情報
QAQQTEXT	SQL ステートメント・テキスト
QAQQ3000	テーブル走査
QAQQ3001	使用された索引
QAQQ3002	作成された索引
QAQQ3003	分類
QAQQ3004	一時テーブル
QAQQ3007	最適化プログラムのタイムアウト/考慮された全索引
QAQQ3008	副照会
QAQQ3010	ホスト変数値
QAQQ3030	考慮されるマテリアライズ照会表

SQL 照会のサンプル

STRDBMON モニターと同様、すべてのモニター・データが格納されているテーブルからの情報の抽出は、ユーザーが行います。これは、ユーザーの選択する照会インターフェースを通して行うことができます。

SQL モニター用のサポート付きの System i Navigatorを使用している場合には、グラフィカル・ユーザー・インターフェースを介して結果を直接分析することができます。任意のテーブルから情報を抽出するために使用可能で、変更可能な多数の出荷済み照会があります。これらの照会のリストについては、DB2 for i5/OS Web サイトの、 Common queries on analysis of DB Performance Monitor data にアクセスしてください。

記憶域常駐データベース・モニターの行識別

結合キー列 QQKEY は、複数のテーブルを 1 つに結合するのを簡単にします。この列は、データベース・モニターで使用していた、結合フィールド (QQJFLD) および固有照会カウンター (QQCNT) を置き換えます。 この結合キー列には、この照会に関するすべての情報をテーブルのそれぞれから受け取ることを可能にする固有 ID が入っています。

この結合キー列は、照会の個々のステップについての特定の情報を識別するために引き続き必要なすべての明細列を置き換えるものではありません。 照会定義テンプレート (QDT) 番号または副選択番号で、各明細ステップについての情報が識別されます。 これらの列を使用して、次の照会処理の各ステップにどの行が属するかを識別します。

- QQQDTN - 照会定義テンプレート番号
- QQQDTL - 照会定義テンプレート副選択番号 (副照会)
- QQMATN - マテリアライズした照会定義テンプレート番号 (ビュー)
- QQMATL - マテリアライズした照会定義テンプレート副選択番号 (ビュー/副照会)
- QQMATULVL - マテリアライズした照会定義テンプレート共用体番号 (ビュー/共用体)

モニターした照会に副照会、共用体、またはビュー操作が含まれるときにこれらの列を使用します。すべての照会タイプは、複数の QDT を生成して元の照会要求を満たすことができます。システムは、各 QDT が元のこの照会 (QQKEY) に属しているとして識別することを引き続き可能にする一方で、これらの列を使用して各 QDT に関する情報を分離します。

System i ナビゲーターを使用した要約モニターの使用

要約モニターは、System i ナビゲーターのインターフェースから扱えます。要約モニターはネイティブ・インターフェースにあり、記憶域常駐データベース・モニター (DBMon) を作成します。

名前の通り、このモニターはメモリーに常駐し、収集されたデータの要約だけを保存します。モニターが一時停止または終了すると、このデータはハード・ディスクに書き込まれ、分析可能になります。このモニターはその情報をメモリーに保管するので、システムに与えるパフォーマンスの影響は最小限にとどまります。しかし詳細の一部は失われます。

要約モニターの開始

System i ナビゲーター・インターフェースから、要約モニターを開始できます。

このモニターを開始するには、System i ナビゲーター・ツリーのデータベース部分の下にある「SQL パフォーマンス・モニター」を右クリックし、「**新規**」 → 「**SQL パフォーマンス・モニター**」を選択します。「モニター・ウィザード」で、「**要約**」を選択します。

要約モニターを作成する場合、一定の種類の情報が常に収集されています。この情報には、要約情報、SQL ステートメント情報、およびホスト変数情報が含まれます。また、次のタイプの情報を収集するよう選択できます。

表スキャンおよび到着順
: モニターされたジョブの表スキャン・データに関する情報を含めるために選択します。大きいテーブルのテーブル走査には、時間がかかることがあります。 SQL ステートメントが長時間実行される場合には、それは、パフォーマンス改善のために索引が必要であることを示している可能性があります。

使用される索引
: 索引がモニターされたジョブに使用された方法に関する情報を含めるために選択します。この情報

を使用して、照会のパフォーマンス改善のために、いずれかの永続索引が使用されたかどうかを迅速に知ることができます。通常、最善の照会パフォーマンスを達成するためには、永続索引が必要になります。この情報を使用して、モニターされたステートメント中で永続索引が使用された頻度を判別することができます。テーブルに対する挿入、更新、および削除のパフォーマンスを改善するためには、まったく使用されないか、めったにしか使用されない索引を除去する必要があります。索引をドロップする前に、索引が Query 最適化プログラムにより統計上のソースとして使用されているか判別してください。

索引の作成

モニターされたジョブの索引の作成に関する情報を含めるために選択します。結合の実行、両方向スクロール・カーソルのサポート、ORDER BY または GROUP BY の実施などいくつかの理由から、一時索引の作成が必要となることがあります。作成される索引には、照会に適合する行のキーしか入れることができません (このような索引は疎索引として知られています) 。多くの場合に、この索引作成は完全に正常であり、照会を実行する最も効率的な方法です。しかしながら、行数が多い場合、あるいは同じ索引が反復作成される場合には、この照会のパフォーマンス改善のために、永続索引を作成することができます。これは、索引がアドバイスされたどうかとは関係のない事実です。

データ・ソート

モニターされたジョブが実行するデータ・ソートに関する情報を含めるために選択します。 SQL ステートメント中の大規模結果セットのソートには、時間がかかる可能性があります。ある場合には、索引を作成することによって、ソートの必要性がなくなることがあります。

一時ファイルの使用

モニターされたジョブにより作成された一時ファイルに関する情報を含めるために選択します。 SQL ステートメントによっては、一時結果が必要になる場合があります。一時結果に挿入される結果セットが大きい場合には、一時結果が必要な理由の調査が必要になる可能性があります。ある場合には、SQL ステートメントを変更して、一時結果が必要ないようにすることができます。たとえば、カーソルが INSENSITIVE 属性を持っている場合には、一時結果が作成されます。キーワード INSENSITIVE を除去すると、通常は一時結果の必要性がなくなりますが、アプリケーションは、データベース・テーブルで変更が起こると、変更内容を参照することになります。

考慮された索引

モニターされたジョブに対して考慮された索引に関する情報を含めるために選択します。この情報は、ある索引が照会で使用されているかを判別する際に役立ちます。ある索引が考慮されているが、使用されていない場合、その索引を書き直すか、ドロップする必要がある場合があります。索引をドロップする前に、索引が Query 最適化プログラムにより統計上のソースとして使用されているか判別してください。

副選択処理

副選択処理に関する情報を含めるために選択します。この情報は、複合 SQL ステートメントの最もコストの高い副照会を示します。

モニターするジョブを選択するか、すべてのジョブをモニターするよう選択できます。システム上で、モニターのインスタンスを複数、同時に実行することができます。要約モニターの場合、すべてのジョブをモニターできるモニター・インスタンスは 1 つだけ持つことができます。また、同一ジョブに対して、2 つのモニターを持つことはできません。すべてのジョブの情報を収集する場合、モニターは先に開始済みのジョブ、または、モニターが作成された後に開始された新規ジョブの情報を収集します。「**選択されたジョブ**」リストでジョブを選択、削除することにより、このリストを編集できます。

関連資料

182 ページの『不要な索引の判別』
照会の最適化で使用されている索引の判別は簡単に行えます。

要約モニター情報の分析

データがモニターで収集された後、その分析が可能となります。

要約モニターの情報の分析は、右側のペインの要約モニターを右クリックし、「**分析**」を選択することによって行えます。データを分析するには、要約モニターを終了または休止させる必要があります。

以下は、定義済み報告書から入手できる情報の概説です。

一般要約 (General Summary)
すべての SQL 活動を要約した情報が入っています。この情報は、使用された SQL ステートメントの性質について高水準の徴候を提供します。たとえば、SQL がアプリケーションで使用されている頻度、その SQL ステートメントが主として短時間実行であるか、あるいは長時間実行であるか、戻される結果の数が少ないか、多いかなどです。

ジョブ要約
各ジョブごとに 1 行の情報が含まれます。個々の行がそのジョブに対するすべての SQL 活動を要約しています。この情報は、システムのどのジョブが SQL の中で最高負荷のユーザーであるか、およびどれがパフォーマンス調整の候補となるかを示すために使用されます。これによって、ユーザーは、システム全体をモニターすることなく、個別のジョブに対して詳細パフォーマンス・モニターを開始し、より詳細な情報を得ることが必要になることがあります。

操作要約
SQL 操作の各タイプごとに 1 行の要約情報が含まれます。個々の行がそのタイプの SQL 操作に対するすべての SQL 活動を要約しています。この情報は、使用された SQL ステートメントのタイプの高水準の徴候を提供します。たとえば、アプリケーションが主として読み取り専用であるか、あるいは多くの更新、削除、または挿入活動があるかなどです。この情報を使用して、特定パフォーマンス調整手法を試みることができます。たとえば、多くの挿入活動が行われる場合には、ブロック化因数を大きくする OVRDBF コマンドの使用または QDBENCWT API の使用が適切と考えられます。

プログラム要約
SQL 操作を実行した各プログラムごとに 1 行の情報が含まれます。個々の行がそのプログラムに対するすべての SQL 活動を要約しています。この情報は、どのプログラムが最大で最もコストのかかる SQL ステートメントを使用しているかを識別するのに使用することができます。これらのプログラムは、パフォーマンス調整の潜在的な候補になります。プログラム名が使用可能となるのは、その SQL ステートメントがコンパイル済みプログラムに組み込まれている場合だけであることに注意してください。 ODBC、JDBC、または OLE DB を介して出された SQL ステートメントは、それがプロシージャー、関数、またはトリガーに起因するものでないかぎり、プログラム名はブランクになります。

さらに、別の「詳細結果」を選択できます。

基本ステートメント情報
この情報は、各 SQL ステートメントに関する基本情報を提供します。最もコストの高い SQL ステートメントがリストに最初に表示されるため、一見しただけでどのステートメント (ある場合) が長時間実行されたかが分かります。

アクセス・プラン再作成情報
アクセス計画の再作成を必要とした各 SQL ステートメントごとに 1 行の情報が含まれます。場合

によっては、新規索引の作成や除去、PTF の適用など、いくつかの理由の 1 つから再最適化が必要になります。しかし、アクセス・プランの再作成が多すぎるということは、問題を示している可能性があります。

最適化プログラム情報

SQL ステートメント内の各副選択ごとに 1 行の最適化情報が含まれます。この情報は、データ操作 (選択、オープン、更新など) を含む SQL ステートメントに関する基本最適化プログラム情報を提供します。最もコストの高い SQL ステートメントが、リストの最初に表示されます。

索引作成情報

索引の作成を必要とした各 SQL ステートメントごとに 1 行の情報が含まれます。結合の実行、両方向スクロール・カーソルのサポート、ORDER BY または GROUP BY の実施などいくつかの理由から、一時索引の作成が必要となることがあります。作成される索引には、照会に適合する行のキーしか入れることができません (このような索引は疎索引として知られています) 。多くの場合に、この索引作成は完全に正常であり、照会を実行する最も効率的な方法です。しかしながら、行数が多い場合、あるいは同じ索引が反復作成される場合には、この照会のパフォーマンス改善のために、永続索引を作成することができます。これは、索引がアドバイスされたどうかとは関係のない事実です。

使用索引情報

SQL ステートメントが使用した各永続索引ごとに 1 行の情報が含まれます。これを使用して、照会のパフォーマンス改善のために、いずれかの永続索引が使用されたかどうかを迅速に知ることができます。通常、最善の照会パフォーマンスを達成するためには、永続索引が必要になります。この情報を使用して、モニターされたステートメント中で永続索引が使用された頻度を判別することができます。テーブルに対する挿入、更新、および削除のパフォーマンスを改善するためには、まったく使用されないか、めったにしか使用されない索引を除去する必要があります。索引を除去する前には、その索引の記述情報中の最終使用日を調べることが必要になる可能性があります。

オープン情報

各 SQL ステートメントに対する個々のオープン活動ごとに 1 行の情報が含まれます。ジョブの特定のステートメントで最初にオープンが実行されると (あるいはオープンのつど)、そのオープンはフル・オープンになります。フル・オープンは、オープン・データ・パス (ODP) を作成しますが、これは行の取り出し、更新、削除、または挿入に使用されます。通常、1 つの ODP について多くの取り出し、更新、削除、または挿入操作が実行されるので、続く各入出力操作で同じ処理を実行する必要がないように、ODP 作成時には、できるだけ多くの SQL ステートメントの処理が実行されます。ジョブ中に SQL ステートメントが再び実行される場合に ODP を再使用できるように、 ODP はクローズ時にキャッシュに入れられます。こうしたオープンは、疑似オープンと呼ばれ、フル・オープンよりかなりコストを抑えることができます。ジョブのキャッシュに入れる ODP の数、およびキャッシュに入れる前に 1 つのステートメントについて同じ ODP の作成必要回数を制御することができます。

テーブル走査

レコードを到着順に処理する必要のある各副選択ごとに 1 行の情報が含まれます。大きいテーブルのテーブル走査には、時間がかかることがあります。 SQL ステートメントが長時間実行される場合には、それは、パフォーマンス改善のために索引が必要であることを示している可能性があります。

ソート情報

SQL ステートメントが実行した各ソートごとに 1 行の情報が含まれます。 SQL ステートメント中の大規模結果セットのソートには、時間がかかる可能性があります。ある場合には、索引を作成することによって、ソートの必要性がなくなることがあります。

一時ファイル情報
一時的な結果を必要とした各 SQL ステートメントごとに 1 行の情報が含まれます。 SQL ステートメントによっては、一時結果が必要になる場合があります。一時結果に挿入される結果セットが大きい場合には、一時結果が必要な理由の調査が必要になる可能性があります。ある場合には、SQL ステートメントを変更して、一時結果が必要ないようにすることができます。たとえば、カーソルが INSENSITIVE 属性を持っている場合には、一時結果が作成されます。キーワード INSENSITIVE を除去すると、通常は一時結果の必要性がなくなりますが、アプリケーションは、データベース・テーブルで変更が起こると、変更内容を参照することになります。

データ変換情報
データ変換が必要となった各 SQL ステートメントごとに 1 行の情報が含まれます。たとえば、結果列は INTEGER 属性を持っているが、結果を戻す変数が DECIMAL の場合には、このデータを整数から小数に変換しなければなりません。 1 回のデータ変換操作には、時間はかかりませんが、千回も百万回も繰り返されると加算されます。場合によっては、属性の 1 つを変更するのは簡単なタスクであり、それによってより速い直接マップを実行することができます。 正確に対応するデータ・タイプが得られないために、この変換が必要になる場合もあります。

副照会情報
1 行の副照会を含んでいます。この情報は、複合 SQL ステートメントの最もコストの高い副照会を示します。

最後に、複合ビューを選択できます。

要約データ
リソースおよびモニターされたジョブに関するその他の一般情報が含まれます。

ステートメント・テキスト
モニターされたジョブが呼び出す SQL テキストが含まれます。

テーブル走査
モニターされたジョブのテーブル走査データが含まれます。

データ・ソート
モニターされたジョブが実行するデータ・ソートの詳細が含まれます。

ホスト変数の使用
モニターされたジョブが使用するホスト変数の値が含まれます。

考慮される最適化プログラム・タイムアウト/アクセス・パス
モニターされたジョブがタイムアウトになる場合、その詳細が含まれます。

使用される索引
モニターされたジョブによる索引の使い方の詳細が含まれます。

索引の作成
モニターされたジョブによる索引の作成の詳細が含まれます。

副選択処理
SQL ステートメント内の各副選択に関する情報が含まれます。

一時ファイルの使用
モニターされたジョブが作成した一時ファイルの詳細が含まれます。

モニターのインポート

System i ナビゲーターを使用することにより、その他のいくつかのインターフェースを使って収集されたモニター・データをインポートすることができます。

データベース・モニターの開始 (STRDBMON) コマンドを使用して作成されたモニターは、System i ナビゲーターによって自動的に登録され、System i ナビゲーターによって表示されるモニターのリストに含まれます。

モニター・データをインポートするには、「**SQL パフォーマンス・モニター**」を右クリックして、「**インポート**」を選択します。モニターをインポートしたら、データを分析できます。

SQL アプリケーションのパフォーマンスの検査

SQL アプリケーションのパフォーマンスの検査は、コマンドを使用して行うことができます。

パフォーマンスの検証に役立つコマンドは、以下のとおりです。

ジョブの表示 (DSPJOB)

ジョブの表示 (DSPJOB) コマンドをパラメーター OPTION(*OPNF) を指定して使用して、ジョブで実行中のアプリケーションが使用する索引およびテーブルを表示することができます。

DSPJOB にパラメーター OPTION(*JOBLCK) を使用して、オブジェクトおよび行ロック競合を分析することができます。 ロックされているオブジェクトと行およびロックを保持しているジョブの名前が表示されます。

DSPJOB コマンドで OPTION(*CMTCTL) パラメーターを指定して、プログラムが実行中の分離レベル、トランザクションの実行中にロックされる行数、および保留状態の DDL 機能を表示します。 表示される分離レベルはデフォルトの分離レベルです。 SQL プログラムで実際に使用する分離レベルは、CRTSQLxxx コマンドの COMMIT パラメーターで指定します。

SQL 情報印刷 (PRTSQLINF)

SQL 情報の印刷 (PRTSQLINF) コマンドは、プログラム、SQL パッケージ、またはサービス・プログラムに組み込まれている SQL ステートメントに関する情報を印刷するものです。このような情報としては、SQL ステートメント、ステートメント実行時に使用するアクセス・プラン、オブジェクトのソース・メンバーをプリコンパイルするのに使用するコマンド・パラメーターのリストなどがあります。

データベース・モニターの開始 (STRDBMON)

データベース・モニターの開始 (STRDBMON) コマンドを使用して、実行するすべての SQL ステートメントに関する情報をファイルに取り込むことができます。

照会属性の変更 (CHGQRYA)

照会属性の変更 (CHGQRYA) コマンドを使用すると、 Query 最適化プログラムの照会属性を変更することができます。このコマンドで変更できる属性には、予測照会管理プログラム、並列処理、および照会オプションがあります。

デバッグの開始 (STRDBG)

デバッグ・モードにジョブを入れるには、デバッグの開始 (STRDBG) コマンドを使用できます。オプションで、20 のプログラム、20 のクラス・ファイル、および 20 のサービス・プログラムをデバッグ・モードに追加できます。また、デバッグ・セッションの特定の属性を指定します。たとえば、実動ライブラリーにあるデータベース・ファイルを、デバッグ・モードで更新可能にするかどうかを指定できます。

関連情報

ジョブの表示 (DSPJOB) コマンド

SQL 情報印刷 (PRTSQLINF) コマンド

データベース・モニターの開始 (STRDBMON) コマンド

照会属性の変更 (CHGQRYA) コマンド

デバッグの開始 (STRDBG) コマンド

ジョブ・ログの Query 最適化プログラム・デバッグ・メッセージの調査

Query 最適化プログラム・デバッグ・メッセージは、照会の実施についてジョブ・ログに通知メッセージを出します。 これらのメッセージは、Query 最適化プログラムでの処理中に何が起こったかを説明します。

たとえば、次のことを知ることができます。

- 索引が使用されるまたは使用されない理由
- 一時結果が必要な理由
- 結合およびブロック化が使用されるかどうか
- 最適化プログラムで推奨している索引のタイプはどれか
- ジョブの照会の状況
- 使用される索引
- カーソルの状況

最適化プログラムは、SQL、呼び出しレベル・インターフェース、ODBC、OPNQRYF、および SQL 照会管理プログラムを含む、それが最適化するすべての照会についてのメッセージを自動的に記録します。

STRDBG コマンドを使用したデバッグ・メッセージの表示:

STRDBG コマンドは、ジョブをデバッグ・モードに入れます。また、デバッグ・セッションの特定の属性を指定します。たとえば、実動スキーマにあるデータベース・ファイルを、デバッグ・モードで更新可能にするかどうかを指定できます。たとえば、次のコマンドを使用してください。

```
STRDBG PGM(Schema/program) UPDPROD(*YES)
```

STRDBG はジョブ・ログに、実行するすべての SQL ステートメントに関する情報を入れます。

QAQQINI コマンドを使用したデバッグ・メッセージの表示:

さらに、照会属性の変更 (CHGQRYA) コマンドの QRYOPTLIB パラメーターを QAQQINI テーブルが入っているユーザー・スキーマに設定できます。 QAQQINI テーブルのパラメーターを MESSAGES_DEBUG に設定し、その値を *YES に設定します。 このオプションは、Query 最適化情報をジョブ・ログに入れます。 QAQQINI テーブルに加えた変更はすぐに有効になり、このテーブルを使用するすべてのユーザーおよび照会に影響します。 MESSAGES_DEBUG パラメーターを変更すると、この QAQQINI テーブルを使用するすべての照会は、それぞれのジョブ・ログにデバッグ・メッセージを書き込みます。 コマンド入力パネルで F10 を押すと、メッセージ・テキストが表示されます。 2 次レベル・テキストを見るには、F1 (ヘルプ) を押します。 2 次レベル・テキストには、照会パフォーマンス向上のためのヒントが見受けられることがあります。

「SQL スクリプトの実行」でのデバッグ・メッセージの表示:

「SQL スクリプトの実行」にデバッグ・メッセージを表示するには、「**オプション**」メニューから、「**ジョブ・ログへのデバッグ・メッセージの組み込み**」を選択します。その後、「**表示**」メニューから「**ジョブ・ログ**」を選択します。詳細なメッセージを表示するには、メッセージをダブルクリックします。

Visual Explain でのデバッグ・メッセージの表示:

Visual Explain では、デバッグ・メッセージが常に表示されます。 オンまたはオフにする必要はありません。 このウィンドウの下部にデバッグ・メッセージが表示されます。メッセージをダブルクリックすると、詳細なメッセージを表示できます。

組み込み SQL ステートメントに関する情報収集

SQL 情報の印刷 (PRTSQLINF) コマンドは、プログラム、SQL パッケージ (そのオブジェクトは、通常、遠隔照会用のアクセス・プランを保管するために使用される) 、またはサービス・プログラム内の組み込み SQL ステートメントに関する情報を戻します。この情報は、スプール・ファイルに保管されます。

PRTSQLINF では、下記についての情報を提供します。

- 実行中の SQL ステートメント
- 実行中に使用されるアクセス・プランのタイプ。これには、照会の実施方法、使用される索引、結合順序、分類を行うかどうか、データベース走査を使用するかどうか、および索引が作成されるかどうかについての情報が含まれます。
- オブジェクトのソース・メンバーをプリコンパイルするのに使用するコマンド・パラメーターのリスト
- 外部プロシージャーまたはユーザー定義関数を作成するのに使用する CREATE PROCEDURE および CREATE FUNCTION ステートメント・テキスト。

この出力は、デバッグ・メッセージから入手できる情報と類似しています。 しかし、照会デバッグ・メッセージが実行時に作用する一方、 PRTSQLINF は過去にさかのぼって作用します。また、照会管理プログラムの照会メッセージ CPA4259 の 2 次レベル・テキストからこの情報を参照することもできます。

幾通りかの方法で PRTSQLINF を実行できます。第 1 に、保管されたアクセス・プランに対して PRTSQLINF を実行できます。このことは、このコマンドを使用する前に、照会を実行するか、または少なくとも照会を準備 (SQL の PREPARE ステートメントを使用して) しなければならないことを意味します。照会を実行するのが最適です。その理由は、PREPARE の結果として作成される索引は、比較的散在していて最初の実行の後で変更になることがよくあるためです。 PRTSQLINF の保管済みアクセス・プランへの要件から、このコマンドを OPNQRYF と一緒には使用できません。

また System i ナビゲーターから、関数、ストアード・プロシージャー、トリガー、SQL パッケージ、およびプログラムに対して PRTSQLINF を実行できます。この機能は、SQL の Explain と呼ばれています。PRTSQLINF 情報を表示するには、オブジェクトを右マウス・ボタン・クリックして「**SQL の Explain**」を選択します。

関連情報

SQL 情報印刷 (PRTSQLINF) コマンド

Query 最適化ツール: 比較表

この表を使用して、それぞれのツールが照会に関して報告できる情報、プロセスの中で特定のツールが照会を分析できるタイミング、および照会の向上のためにそれぞれのツールが実行できるタスクについて学ぶことができます。

PRTSQLINF	STRDBG または CHGQRYA	ファイル・ベースのモニター (STRDBMON)	メモリー・ベースのモニター	Visual Explain
照会を実行しなくても使用可能 (アクセス・プランが作成された後)	照会の実行時にのみ使用可能	照会の実行時にのみ使用可能	照会の実行時にのみ使用可能	照会の Explain にのみ使用可能
実行されたかどうかに関係なく、SQL プログラム内のすべての照会の場合に表示されます。	実行された照会の場合にのみ表示されます。	実行された照会の場合にのみ表示されます。	実行された照会の場合にのみ表示されます。	Explain された照会の場合にのみ表示されます。
ホスト変数実施についての情報	ホスト変数の実施についての限定情報	ホスト変数、実施、および値についてのすべての情報	ホスト変数、実施、および値についてのすべての情報	ホスト変数、実施、および値についてのすべての情報
プログラム、パッケージ、またはサービス・プログラムを持つ SQL ユーザーに対してのみ使用可能	すべての照会ユーザー (OPNQRYF、SQL、Query/400) に対して使用可能	すべての照会ユーザー (OPNQRYF、 SQL、Query/400) に対して使用可能	SQL インターフェースに対してのみ使用可能	System i ナビゲーター・データベースおよび API インターフェースから使用可能
メッセージはスプール・ファイルに印刷されます。	メッセージはジョブ・ログに表示されます。	パフォーマンス行はデータベース・テーブルに書き込まれます。	パフォーマンス情報は記憶域内に収集されてからデータベース・テーブルに書き込まれます。	System i ナビゲーターでは情報がビジュアルに表示されます。
メッセージを副照会または共用体を持つ照会に結合するのがより簡単	メッセージを副照会または共用体を持つ照会に結合するのが困難	すべての照会、副照会、およびマテリアライズ・ビューを一意的に識別します。	繰り返される照会要求が要約されます。	照会および関連情報のインプリメンテーションを表示するのが簡単

照会の属性の変更

照会属性の変更 (CHGQRYA) CL コマンドにより、あるいは System i ナビゲーター照会属性の変更インターフェースを使用して、一定のジョブの間に実行する照会の属性の各種タイプを変更することができます。

関連概念

5 ページの『プラン・キャッシュ』
プラン・キャッシュは、SQE によって最適化された照会用のアクセス・プランを含むリポジトリーです。

45 ページの『並列で処理されるオブジェクト』
DB2 Symmetric Multiprocessing フィーチャーを使用すれば、最適化プログラムは並列処理を含む付加的なデータ検索方式を使用することができます。対称型マルチプロセッシング (SMP) は、単一システム上に設けられた並列処理の形式です。このシステムでは、メモリーとディスク・リソースを共用する複数 (CPU および入出力) プロセッサーが、同時に単一の終了結果を得るために処理を行います。

関連情報

照会属性の変更 (CHGQRYA) コマンド

照会オプション・ファイル QAQQINI による照会の動的な制御

照会オプション・ファイル QAQQINI サポートには、照会属性の変更 (CHGQRYA) コマンドおよび QAQQINI ファイルを通して照会が実行される環境を動的に変更または上書きする機能があります。照会オプション・ファイル QAQQINI を使用して、データベース・マネージャーによって使用される属性を設定します。

実行する照会ごとに、CHGQRYA CL コマンドの QRYOPTLIB パラメーターに指定したスキーマ内の QAQQINI ファイルから照会オプション値が検索され、照会を最適化して実施するのに使用されます。

QAQQINI ファイルを通して変更可能な環境属性には、次のものがあります。

- APPLY_REMOTE
- ASYNC_JOB_USAGE
- COMMITMENT_CONTROL_LOCK_LIMIT
- FORCE_JOIN_ORDER
- IGNORE_DERIVED_INDEX
- IGNORE_LIKE_REDUNDANT_SHIFTS
- LOB_LOCATOR_THRESHOLD
- MATERIALIZED_QUERY_TABLE_REFRESH_AGE
- MATERIALIZED_QUERY_TABLE _USAGE
- MESSAGES_DEBUG
- NORMALIZE_DATA
- OPEN_CURSOR_CLOSE_COUNT
- OPEN_CURSOR_THRESHOLD
- OPTIMIZE_STATISTIC_LIMITATION
- OPTIMIZATION_GOAL
- PARALLEL_DEGREE
- PARAMETER_MARKER_CONVERSION
- QUERY_TIME_LIMIT
- REOPTIMIZE_ACCESS_PLAN
- SQLSTANDARDS_MIXED_CONSTANT
- SQL_DECFLOAT_WARNINGS
- SQL_FAST_DELETE_ROW_COUNT
- SQL_FLAGGER
- SQL_PSEUDO_CLOSE
- SQL_STMT_COMPRESS_MAX
- SQL_STMT_REUSE
- SQL_SUPPRESS_WARNINGS
- SQL_TRANSLATE_ASCII_TO_JOB
- STAR_JOIN
- STORAGE_LIMIT
- SYSTEM_SQL_STATEMENT_CACHE

- UDF_TIME_OUT
- VARIABLE_LENGTH_OPTIMIZATION

関連資料

57 ページの『ルック・アヘッド述部生成 (LPG)』
ルック・アヘッド述部生成 (LPG) と呼ばれる推移的閉包の特別なタイプは、結合でコストがかかる場合があります。この場合、最適化プログラムは照会の結果を大きなファクト・テーブルに事前に適用して、結合のランダム入出力コストを最小化しようとします。一般に LPG は、星形結合照会と呼ばれる照会のクラスで使用されますが、任意の結合照会でおそらく使用可能です。

QAQQINI ファイルの指定:

照会属性の変更 (CHGQRYA) コマンドを、照会オプション・ファイル QAQQINI が現在入っている、または入る予定のスキーマがどれであるかを指定するパラメーター QRYOPTLIB (照会オプション・ライブラリー) を指定して使用します。

照会オプション・ファイルは、各照会の QRYOPTLIB パラメーターに指定したスキーマから検索され、そのジョブまたはユーザー・セッションの間あるいは QRYOPTLIB パラメーターが 照会属性の変更 (CHGQRYA) コマンドによって変更されるまでは引き続き有効となります。

照会属性の変更 (CHGQRYA) コマンドが発行されない場合、あるいは発行されるものの QRYOPTLIB パラメーターが指定されない場合、スキーマ QUSRSYS から QAQQINI ファイルが探索されます。照会に対してオプション・ファイルが検出されない場合、どの属性も変更されません。システムの出荷時には、QUSRSYS に INI ファイルが入っていないため、INI ファイルがないというメッセージを受け取ることがあります。このメッセージは、エラーではなく、すべてのデフォルト値を含む QAQQINI ファイルが使用中であることを示します。ジョブの QRYOPTLIB パラメーターの初期値は QUSRSYS です。

関連情報

照会属性の変更 (CHGQRYA) コマンド

QAQQINI 照会オプション・ファイルの作成:

各システムは、スキーマ QSYS 内に QAQQINI テンプレート・ファイルを付けて出荷されます。 QSYS 内の QAQQINI ファイルは、すべてのユーザー指定 QAQQINI ファイルの作成時にテンプレートとして使用されます。

ユーザー独自の QAQQINI ファイルを作成するには、オブジェクト複製 (CRTDUPOBJ) コマンドを使用して QAQQINI ファイルのコピーを QUERY 属性変更 (CHGQRYA) QRYOPTLIB パラメーターに指定するスキーマに作成します。ファイル名は、QAQQINI のままでなければなりません。たとえば、次のとおりです。

```
CRTDUPOBJ OBJ(QAQQINI)
          FROMLIB(QSYS)
          OBJTYPE(*FILE)
          TOLIB(MYLIB)
          DATA(*YES)
```

システムで提供するトリガーが QSYS 内の QAQQINI ファイルに付加されるため、 QAQQINI ファイルをコピーする唯一の手段は、必然的に CRTDUPOBJ CL コマンドになります。 CPYF などの他の手段が使用されると、トリガーが破壊され、オプション・ファイルを検索できない、またはオプション・ファイルを更新できないというエラーが出されます。

QAQQINI ファイルにトリガー・プログラムが添付されているため、以下の CPI321A 通知メッセージは、ファイルを作成するために CRTDUPOBJ CL を使用する際に、ジョブ・ログに 6 回表示されます。これはエラーではありません。これは、単なる通知メッセージです。

CPI321A 情報メッセージ・ライブラリー &1 のトリガー QSYS_TRIG_&1_QAQQINI_00000&N がライブラリー &1 のファイル QAQQINI に追加された。アンパーサンド変数 (&1, &N) は、ライブラリー名または数値変数のいずれかを含む置き換え変数です。

注: QSYS 内のファイル QAQQINI は、変更しないことを推奨します。これはオリジナルのテンプレートで、使用するために、QUSRSYS またはユーザー指定ライブラリーに複写されます。

関連情報

照会属性の変更 (CHGQRYA) コマンド

複製オブジェクト作成 (CRTDUPOBJ) コマンド

QAQQINI ファイル・オーバーライド・サポート:

QAQQINI 照会オプション・ファイルの処理が煩雑な場合は、qsys2.override_qaqqini() プロシージャーの使用を検討してください。 QAQQINI *FILE オブジェクトを直接、作成、管理、および使用する代わりに、このプロシージャーを呼び出し、ユーザー指定のオプションおよび値を使用して、INI ファイルの一時バージョンを処理することができます。サポートは、QTEMP ライブラリーに依存するため、どのような変更でも、プロシージャーを呼び出すジョブのみに影響を及ぼします。

CHGQRYA コマンドには、*JOBCTL 権限が必要です。この新規プロシージャーは、ユーザーに CHGQRYA の代替を提供するので、*JOBCTL は、多少の例外を除き、すべてのオプションに対して実行されます。次のオプションは、すべてのユーザーが一時変更することができます。

- SQL_FLAGGER
- OPEN_CURSOR_THRESHOLD
- OPEN_CURSOR_CLOSE_COUNT
- PARAMETER_MARKER_CONVERSION
- SYSTEM_SQL_STATEMENT_CACHE

使用例

ステップ 1: QAQQINI オーバーライドを確立します (QTEMP.QAQQINI は、ジョブ内で使用中の現行 QAQQINI に基づいて作成されます)

```
--
call qsys2.override_qaqqini(1, '', '');
```

ステップ 2: -- オーバーライドする QAQQINI パラメーターおよび値を選択します。 (繰り返し呼び出すことができます)

```
-- Avoid UDF timeouts
call qsys2.override_qaqqini(2, 'UDF_TIME_OUT', '*MAX');

-- Force full opens of cursors
call qsys2.override_qaqqini(2, 'OPEN_CURSOR_THRESHOLD', '-1');

-- Avoid using the System Wide Statement Cache
call qsys2.override_qaqqini(2, 'SYSTEM_SQL_STATEMENT_CACHE', '*NO');

-- Force any saved access plans to be rebuilt
call qsys2.override_qaqqini(2, 'REBUILD_ACCESS_PLAN', '*YES');
```

セットアップが完了したので、テストを実行します。

ステップ 3: -- オーバーライド値を破棄し、以前の QAQQINI ファイルの使用を再開します (このステップはオプションです)

```
--
call qsys2.override_qaqqini(3, '', '');
```

QAQQINI 照会オプション・ファイルの形式:

QAQQINI ファイルはスキーマ QSYS として出荷されます。この形式は事前定義されており、行のデフォルト値で事前に設定されています。

照会オプション・ファイル:

```
A                                       UNIQUE
A                   R QAQQINI           TEXT('Query options + file')
A                     QQPARM      256A  VARLEN(10) +
                                        TEXT('Query+
                                              option parameter') +
                                        COLHDG('Parameter')
A                     QQVAL       256A  VARLEN(10) +
                                        TEXT('Query option +
                                              parameter value') +
                                        COLHDG('Parameter Value')
A                     QQTEXT      1000G VARLEN(100) +
                                        TEXT('Query +
                                              option text') +
                                        ALWNULL +
                                        COLHDG('Query Option' +
                                               'Text') +
                                        CCSID(13488) +
                                        DFT(*NULL)
A                   K QQPARM
```

照会オプション・ファイル内でのオプション設定:

QAQQINI ファイル照会オプションは、INSERT、UPDATE、または DELETE の各 SQL ステートメントを使用して変更することができます。

次の例の場合、QAQQINI ファイルがすでにライブラリー MyLib 内に作成されています。
MyLib/QAQQINI 内の既存の行を更新するには、UPDATE SQL ステートメントを使用します。 この例では、MESSAGES_DEBUG = *YES と設定しているので、Query 最適化プログラムが最適化プログラムのデバッグ・メッセージを次のように印刷することができます。

```
UPDATE MyLib/QAQQINI SET QQVAL='*YES'
WHERE QQPARM='MESSAGES_DEBUG'
```

MyLib/QAQQINI 内の既存の行を削除するには、DELETE SQL ステートメントを使用します。この例では、次のようにして、QUERY_TIME_LIMIT 行をQAQQINI ファイルから除去します。

```
DELETE FROM MyLib/QAQQINI
WHERE QQPARM='QUERY_TIME_LIMIT'
```

新規行を MyLib/QAQQINI に挿入するには、INSERT SQL ステートメントを使用します。 この例は、*NOMAX の値を持つ QUERY_TIME_LIMIT 行を QAQQINI ファイルに次のように追加します。

```
 INSERT INTO MyLib/QAQQINI
VALUES('QUERY_TIME_LIMIT','*NOMAX','New time limit set by DBAdmin')
```

QAQQINI 照会オプション・ファイル権限の要件:

QAQQINI は *PUBLIC *USE 権限付きで出荷されます。これによって、ユーザーは、照会オプション・ファイルを表示することができますが、変更することはできません。 QAQQINI ファイルの値を変更するとシステムで実行されているすべての照会に影響が及ぶため、システム管理者またはデータベース管理者だけが QAQQINI 照会オプション・ファイルに対する *CHANGE 権限を持つようにするべきです。

照会属性の変更 (CHGQRYA) CL コマンドの QRYOPTLIB パラメーターで指定するライブラリーに存在する、照会オプション・ファイルは、常に Query 最適化プログラムによって使用されます。これは、ユーザーが照会オプション・ライブラリーおよびファイルへの権限を持っていない場合でも当てはまります。これによって、システム管理者にセキュリティーのメカニズムが追加されることになります。

QAQQINI ファイルがライブラリー QUSRSYS にあるときは、照会オプションはシステム上のすべての照会ユーザーに影響します。だれかが照会オプションの挿入、削除、または更新を実行するのを防ぐために、システム管理者はファイルに対する *PUBLIC からの更新権限を削除する必要があります。これによって、ユーザーはこのファイル内のデータを変更できなくなります。

QAQQINI ファイルがユーザー・ライブラリーにあり、このライブラリーが 照会属性の変更 (CHGQRYA) コマンドの QRYOPTLIB パラメーターに指定されているときは、照会オプションはそのユーザーのジョブに対して実行されるすべての照会に影響します。照会オプションが特定のライブラリーから検索されないようにするには、システム管理者は 照会属性の変更 (CHGQRYA) CL コマンドへの権限を取り消すことができます。

QAQQINI ファイル・システム提供のトリガー:

照会オプション・ファイル QAQQINI は、このファイルに対して行われたすべての変更を処理するため、システム提供トリガーを使用します。トリガーをファイル QAQQINI から取り除いたり、これに追加したりすることはできません。

QAQQINI ファイルの更新 (INSERT、DELETE、または UPDATE 操作) でエラーが発生した場合、次の SQL0443 診断メッセージが出されます。

トリガー・プログラムまたは外部ルーチンがエラーを検出した。

QAQQINI 照会オプション:

QAQQINI ファイルのパラメーターにはさまざまなオプションを使用できます。

次の表に、QAQQINI コマンドで指定できる照会オプションの要約を示します。

表 35. QAQQINI コマンドに指定される照会オプション

パラメーター	値	説明
ALLOW_TEMPORARY_INDEXES このオプションを使用すれば、ユーザーは、一時索引が最適化プログラムによって考慮されるべきかどうかを示すことができます。一時索引が許可されない場合は、この照会を実装するためのコストに関係なく、他の任意の実行可能な計画が選択されます。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*YES	一時索引の考慮を許可します。
	*ONLY_REQUIRED	このアクセス・プランではどの一時索引の考慮も許可しません。一時索引の作成を回避するために、コストに関係なく他のインプリメンテーションを選択します。実行可能なプランが見つからない場合にのみ、一時索引は許可されます。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
APPLY_REMOTE 分散ファイルに関係するデータベース照会のために、CHGQRYA 照会属性がこのジョブに関連するリモート・システム上のジョブに適用されるかどうかを指定します。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*NO	ジョブの CHGQRYA 属性は、リモート・ジョブには適用されません。リモート・ジョブは、リモート・システムでリモート・ジョブに関連する属性を使用します。
	*YES	ジョブの照会属性は、分散テーブルが関与するデータベース照会に処理に使用されるリモート・ジョブに適用されます。 *SYSVAL が指定されている属性の場合、リモート・システム上のシステム値は、そのリモート・ジョブ用に使用されます。 CHGQRYA がこのジョブで使用されていた場合、このオプションは、リモート・ジョブが CHGQRYAコマンドを使用する権限を持っていることを要求します。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
ASYNC_JOB_USAGE ジョブ内のデータベース照会の処理に役立つように非同期 (一時書き出しプログラム) ジョブを使用できる環境を指定します。指定された使用オプションは、照会を完了する際に役立つように非同期ジョブ (並列して実行中の) が使用できるデータベース照会のタイプを決定します。非同期ジョブは、データベース照会を実行中のジョブから照会要求を処理するシステム上の別個のジョブです。要求ごとに非同期ジョブは、要求を処理し、結果を一時ファイルに書き出します。この中間一時ファイルは、次にデータベース照会を完了させるためにメイン・ジョブによって使用されます。 非同期ジョブを使用する利点は、メイン・ジョブがデータベース照会の別のステップを処理するのと同時に (並列して) 要求を処理できる点にあります。非同期ジョブを使用する欠点は、メイン・ジョブと同じ方法で処理できない状態が発生する可能性がある点です。例えば、非同期ジョブは、取り消すことになる照会メッセージを受け取る場合がありますが、メイン・ジョブではそのメッセージを無視して続行することを選択できます。 非同期ジョブを実行できるデータベース照会には、次の 2 つの異なるタイプがあります。 1. 分散照会。これらは、分散ファイルが関係するデータベース照会です。分散ファイルは、i5/OS 用のシステム・フィーチャー DB2 マルチシステムを通して提供されます。 2. ローカル照会。データベース照会を実行中のシステムにローカルなファイルだけが関与するデータベース照会があります。	*DEFAULT	デフォルト値は、*LOCAL に設定されます。
	*LOCAL	非同期ジョブは、データベース照会を実行中のシステムにローカルなテーブルだけが関与するデータベース照会に使用することができます。さらに、分散テーブルが関与する照会の場合、このオプションを使用すると、必要な通信方式を非同期にすることができます。これを使用すると、分散テーブルの照会に関与する各システムは、照会の分散テーブルに関与する部分を他のシステムと同時に (並列に) 実行することができます。
	*DIST	非同期ジョブは、分散テーブルが関与するデータベース照会に使用することができます。
	*ANY	非同期ジョブは、すべてのデータベース照会に使用することができます。
	*NONE	非同期ジョブは、データベース照会処理には使用できません。なおその上に、分散テーブルが関与する照会のすべての処理は、同期的に発生します。したがって、システム間での並列処理は、起こりません。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
CACHE_RESULTS 一時結果が関係する照会 (例えば、ソート、ハッシュなど) の場合、結果セットが大きくない限り、照会の次の実行で結果を再使用することを期待して、データベースが、照会の疑似クローズ / 疑似オープンを通して結果を保管することがしばしばあります。さらに、V5R3 からは、別のジョブが後で結果を再使用することを想定して、ジョブが処理を完了した後もデータベースはこれらの一時結果を保管するようになりました。 データベースはこれらの結果のキャッシングを自動的に制御し、記憶域の使用量が大きくなるため、キャッシュ結果を除去します。ただし、データベースが使用する一時記憶域は、従来のリリースより大幅に大きくなる可能性があります。このオプションによって、ユーザーはキャッシュ結果処理を制御できるようになります。	*DEFAULT	デフォルト値は *SYSTEM と同じです。
	*SYSTEM	データベース・マネージャーは照会結果セットをキャッシュに入れることができます。そのジョブまたは任意のジョブ (照会の ODP が削除された場合) によるその後の照会の実行は、キャッシュに入れられた結果セットの再使用を考慮に入れます。
	*JOB	照会が再使用可能 ODP を使用する限り、データベース・マネージャーは 1 つのジョブの実行から次の実行まで、照会結果セットをキャッシュに入れることができます。再使用可能 ODP が削除されると、キャッシュに入れられた結果セットは破棄されます。この値は V5R2 の処理に似ています。
	*NONE	データベースはどの照会結果もキャッシュにいれません。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
COMMITMENT_CONTROL_ LOCK_LIMIT *DEFAULT または、新しい値の設定後に開始されたコミット・トランザクションに対してロックできるレコードの最大数を示す整数値を指定します。 トランザクションに複数のジャーナルが関係する場合、COMMITMENT_CONTROL _LOCK_LIMIT は、全体としてトランザクションではなく、各ジャーナルに適用されます。例えば、ファイル F1 から F5 までがジャーナル J1 に、そしてファイル F6 から F10 が J2 にジャーナル処理され、COMMITMENT_CONTROL _LOCK_LIMIT が 100,000 に設定されると、ファイル F1 から F5 までに対しては 100,000 個のレコード・ロック、ファイル F6 から F10 までに対しては 100,000 個より多いレコード・ロックを獲得することができます。 COMMITMENT_CONTROL _LOCK_LIMIT に対して指定された値は、既にコミットメント制御を開始済みのジョブで実行中のトランザクションに影響を及ぼしません。値を有効にするには、コミットメント制御を開始する前に変更する必要があります。	*DEFAULT	*DEFAULT は 500,000,000 と同等です。
	整数値	新しい値の設定後に開始されたコミット・トランザクションにロックできるレコードの最大数。有効な整数値は 1 から 500,000,000 です。
FORCE_JOIN_ORDER 照会で指定された順序でファイルの結合が発生する Query 最適化プログラムに指定します。	*DEFAULT	デフォルトは *NO に設定されます。
	*NO	最適化プログラムによる結合テーブルの再順序付けを許可します。
	*SQL	SQL JOIN 構文を使用する照会の結合順序を強制するだけです。これは、V4R4M0 より前の最適化プログラムの動作に似ています。
	*PRIMARY nnn	数値 nnn (nnn はオプションで、デフォルトで 1 になる) によってリストされるファイルの結合位置を、結合の基本の位置 (またはダイヤル) に強制するだけです。それから最適化プログラムは、すべての残りのファイルの結合順序をコストに基づいて判別します。
	*YES	Query 最適化プログラムが、その最適化プロセスの一部として結合テーブルの再順序付けを行うことを許可しません。結合は、照会内で指定されたテーブルの順序で行われます。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
IGNORE_DERIVED_INDEX 選択除外索引が照会のテーブル上に存在しても、SQE は照会を処理できます。 SQE は、サポートされない派生索引を無視します。 **注:** このオプションは、WHERE 検索条件を指定した SQL 索引には適用されません。これらの索引の存在は、SQE が照会を処理する機能に影響を及ぼしません。	*DEFAULT	デフォルト値は *YES と同じです。
	*YES	SQE 最適化プログラムが、サポートされない派生索引を無視して照会を処理できるようになります。その結果、派生索引 (複数も可) の存在を考慮することなく、照会プランが作成されます。無視されるサポートされない索引タイプは、次のとおりです。 • 基準が選択または省略され、DYNSLT キーワードが省略されて定義された、キー処理されている論理ファイル
	*NO	サポートされない派生索引を無視しないでください。サポートされない派生索引が存在する場合、CQE が照会を処理するようにします。
IGNORE_LIKE_ REDUNDANT_SHIFTS SQL LIKE 述部あるいは OPNQRYF コマンドの %WLDCRD 組み込み関数を処理するとき、DBCS 混用オペランドの冗長シフト文字を無視するかどうかを指定します。	*DEFAULT	デフォルト値は、*OPTIMIZE に設定されます。
	*ALWAYS	SQL LIKE 述部あるいは OPNQRYF コマンドの %WLDCRD 組み込み関数を処理するとき、DBCS 混用オペランドの冗長シフト文字が無視されます。このオプションを指定すると、Query 最適化プログラムは、DBCS 混用、DBCS 択一、あるいは DBCS 専用オペランドを含む SQL LIKE 述部あるいは OPNQRYF %WLDCRD 述部で、索引を使ってキー行位置決めを実行できなくなることに注意してください。
	*OPTIMIZE	SQL LIKE 述部あるいは OPNQRYF コマンドの %WLDCRD 組み込み関数を処理するとき、それらの述部のキー行位置決めに索引が使用されるかどうかにより、DBCS 混用オペランドの冗長シフト文字が無視されるかどうかが左右されます。このオプションを指定すると、Query 最適化プログラムは、DBCS 混用、DBCS 択一、あるいは DBCS 専用オペランドを含む SQL LIKE 述部あるいは OPNQRYF %WLDCRD 述部で、キー行位置決めを考慮できるようになるということに注意してください。
LIMIT_PREDICATE_ OPTIMIZATION 索引の最適化を実行する際に、Query 最適化プログラムが単純な分離可能な述部 (OIF) のみを使用するように指定します。 OIF は、追加の評価のないレコードを除去できる述部です。 OIF として分類できない述部は、最適化プログラムによって無視され、キーなし選択述部として評価される必要があります。 A=10 および (A=>10 AND B=9) は、OIF です。 A=10 OR B=9 は、OIF ではありません。 注: *YES の場合、索引の最適化が損なわれるか制限されます。	*DEFAULT	索引の最適化を行うときに、単純な分離可能な述部 (OIF) ではない述部を除去しません。 *NO と同じです。
	*NO	索引の最適化を行うときに、単純な分離可能な述部 (OIF) ではない述部を除去しません。
	*YES	索引の最適化を行うときに、単純な分離可能な述部 (OIF) ではない述部を除去します。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
LOB_LOCATOR_THRESHOLD *DEFAULT または整数値 (ジョブ内に存在する適格な LOB ロケーターを解放するしきい値) を指定します。	*DEFAULT	デフォルト値は、0 に設定されます。これは、ロケーターを解放するためデータベースが何もアクションを取らないことを意味します。
	整数値	値が 0 の場合には、データベースはロケーターを解放するために何もアクションを取りません。値が 1 から 250,000 の場合には、FETCH 要求に対して、データベースはジョブの SQL 現行 LOB ロケーター・カウントをしきい値と比較します。ロケーター・カウントがしきい値以上の場合には、ホスト・サーバーが作成した、検索されたロケーターをデータベースは解放します。このオプションは、すべてのホスト・サーバー・ジョブ (QZDASOINIT) に適用され、他のジョブには影響を与えません。
MATERIALIZED_QUERY_TABLE_REFRESH_AGE このパラメーターは、照会最適化および実行時のマテリアライズ照会表の使用法を制御する機能を提供します。	*DEFAULT	デフォルト値は、0 に設定されます。
	0	マテリアライズ照会表は使用できません。
	*ANY	MATERIALIZED_QUERY_TABLE_USAGE INI パラメーターの指すテーブルがあれば、それらはすべて使用可能です。
	timestamp_duration	MATERIALIZED_ QUERY_TABLE_USAGE INI オプションの指すテーブルのうち、指定するタイム・スタンプ期間内に REFRESH TABLE が実行されたことのあるものだけを使用できます。
MATERIALIZED_QUERY_TABLE_USAGE このパラメーターは、最後に REFRESH TABLE ステートメントが実行された時刻に基づいて、使用に適格なマテリアライズ照会表を検査する機能を提供します。	*DEFAULT	デフォルト値は、*NONE に設定されます。
	*NONE	照会の最適化およびインプリメンテーションで、マテリアライズ照会表を使用することはできません。
	*ALL	ユーザーの保守するマテリアライズ照会表を使用できます。
	*USER	ユーザーの保守するマテリアライズ照会表を使用できます。
MESSAGES_DEBUG 通常はジョブがデバッグ・モードの場合に発行される Query 最適化プログラム・デバッグ・メッセージを、ジョブ・ログに表示するかどうかを指定します。	*DEFAULT	デフォルトは *NO に設定されます。
	*NO	表示されるデバッグ・メッセージはありません。
	*YES	STRDBG に対して生成されるすべてのデバッグ・メッセージを実行します。
NORMALIZE_DATA Unicode 定数、ホスト変数、パラメーター・マーカー、およびストリングを結合する式の正規化を実行するかどうかを指定します。	*DEFAULT	デフォルトは *NO に設定されます。
	*NO	ユニコード定数、ホスト変数、パラメーター・マーカー、およびストリングを結合する式は正規化されません。
	*YES	ユニコード定数、ホスト変数、パラメーター・マーカー、およびストリングを結合する式は正規化されます

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
OPEN_CURSOR_CLOSE_COUNT *DEFAULT または整数値 (しきい値に達した際に完全にクローズされるカーソルの数) を指定します。	*DEFAULT	*DEFAULT は 0 と同等です。詳細については『整数値』を参照してください。
	整数値	OPEN_CURSOR_CLOSE_COUNT は OPEN_CURSOR_THRESHOLD と共に使用され、ジョブ内のオープン・カーソルの数を管理します。オープン・カーソル (オープン・カーソルと疑似クローズされたカーソルを含む) の数が、OPEN_CURSOR_THRESHOLD で指定された値に達すると、疑似クローズされたカーソルがハード・クローズ (完全にクローズ) されます。使用された時期の古いカーソルから順にクローズされます。この値は、クローズするカーソルの数を決定します。このパラメーターの有効な値は 1 から 65536 です。このパラメーターの値は、OPEN_CURSOR_THREHOLD パラメーターの数値以下でなければなりません。 OPEN_CURSOR_THRESHOLD が *DEFAULT である場合には、この値は無視されます。 OPEN_CURSOR_THRESHOLD が指定されており、この値が *DEFAULT の場合、クローズされるカーソルの数は、OPEN_CURSOR_THRESHOLD に 10 % を乗算し、次の整数値に切り上げた値に等しくなります。
OPEN_CURSOR_THRESHOLD *DEFAULT または整数値 (疑似クローズされたカーソルの完全クローズを開始するしきい値) を指定します。	*DEFAULT	*DEFAULT は 0 と同等です。詳細については『整数値』を参照してください。
	整数値	OPEN_CURSOR_THRESHOLD は OPEN_CURSOR_CLOSE_COUNT と共に使用され、ジョブ内のオープン・カーソルの数を管理します。オープン・カーソル (オープン・カーソルと疑似クローズされたカーソルを含む) の数がこのしきい値に達すると、疑似クローズされたカーソルがハード・クローズ (完全にクローズ) されます。使用された時期の古いカーソルから順にクローズされます。クローズされるカーソルの数は、OPEN_CURSOR_CLOSE_COUNT により決定されます。ユーザーが入力できるこのパラメーターの有効な値は 1 から 65536 です。値 0 (デフォルト値) を指定すると、しきい値はなくなり、ジョブ内のオープン・カーソルの数に基づいてハード・クローズが強制されることはなくなります。
OPTIMIZATION_GOAL Query 最適化プログラムがコスト計算を決定する際に使用すべき目標を指定します。	*DEFAULT	最適化の目標は、インターフェース (ODBC、SQL プリコンパイラー・オプション、OPTIMIZE FOR nnn ROWS 文節) によって決定されます。
	*FIRSTIO	すべての照会は、出力の最初のページをできるだけ早く戻すという目標で最適化されます。この目標は、出力データの最初のページを見ただけでたいていは照会を取り消す可能性が高いユーザーが、出力を制御する場合に適しています。 OPTIMIZE FOR nnn ROWS 文節でコード化された照会は、文節によって指定される目標を実行します。
	*ALLIO	すべての照会は、最短の経過時間で全体の照会の実行を完了するという目標で最適化されます。これは、ファイルまたはレポートに照会の出力を書き込む場合や、インターフェースが出力データを待ち行列に入れる場合に良い選択です。 OPTIMIZE FOR nnn ROWS 文節でコード化された照会は、文節によって指定される目標を実行します。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
OPTIMIZE_STATISTIC_LIMITATION Query 最適化プログラムの統計収集フェーズに対する制限を指定します。 照会最適化で最も時間を要する側面の 1 つが、照会済みのテーブルに関連する索引からの統計収集です。一般に、照会に関係するテーブルのサイズが大きいほど、統計の収集フェーズにより長い時間を要します。 このオプションは、最適化のこのフェーズ中に消費されるリソースの量を制限する機能を提供します。統計収集に消費されるリソースが大きいほど、最適化計画はより正確 (最適) となることに注意してください。	*DEFAULT	索引統計の収集に要した総量は、Query 最適化プログラムによって判別されます。
	*NO	索引統計は、Query 最適化プログラムによっては収集されません。デフォルト統計が最適化に使用されます。 (このオプションは控えめに使用してください。)
	*PERCENTAGE 整数値	統計の収集中に探索される索引の最大のパーセントを指定します。有効な値は 1 から 99 です。
	*MAX_NUMBER_OF_RECORDS_ALLOWED 整数値	最大テーブル・サイズを統計の収集で使用できる行数で指定します。指定値よりも多くの行を指定したテーブルの場合、最適化プログラムは、統計の収集を行わず、デフォルト値を使用します。

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
PARALLEL_DEGREE ジョブ内でデータベース照会、データベース・ファイル・キー順アクセス・パスの作成、再作成、および保守を実行中に使用できる並列処理オプションを指定します。指定された並列処理オプションは、許可される並列処理のタイプを決定します。並列処理には以下の 2 つのタイプがあります。 1. 入出力 (I/O) 並列処理。入出力並列処理では、データベース・マネージャーは、照会ごとに複数のタスクを使用して、入出力処理を実行します。中央処理装置 (CPU) 処理は引き続き、順次に実行されます。 2. 対称型マルチプロセッシング (SMP)。 SMP は、CPU および入出力処理の両方を、並列して照会を実行するタスクに割り当てます。実際の CPU 並列処理には、複数のプロセッサーを持つシステムが必要です。 SMP は、システム・フィーチャーの DB2 Symmetric Multiprocessing (i5/OS 用) がインストール済みの場合にのみ使用されます。 SMP 並列処理の使用は、レコードが戻される順序に影響を及ぼす可能性があります。	*DEFAULT	デフォルト値は、*SYSVAL に設定されます。
	*SYSVAL	使用する処理オプションがシステム値、QQRYDEGREE の現行値に設定されます。
	*IO	データベース Query 最適化プログラムが照会に入出力並列処理の使用を選択するときに、使用できるタスク数。 SMP 並列処理は許可されません。
	*OPTIMIZE	Query 最適化プログラムは、照会の処理またはデータベース・ファイル・キー順アクセス・パス構築、再構築、または保守のために入出力並列処理または SMP 並列処理について、任意の個数のタスクを使用することを選択できます。 SMP 並列処理は、システム・フィーチャーの DB2 Symmetric Multiprocessing (i5/OS 用) がインストール済みの場合にのみ、使用されます。並列処理の使用および使用されるタスク数が決定される要因は、システムで使用できるプロセッサー数、ジョブが実行されるプール内でこのジョブが使用できるアクティブ・メモリーの共用量、および照会またはデータベース・ファイル・キー順アクセス・パス構築または再構築に予期される経過時間が CPU 処理または入出力リソースにより制限されるかどうかについてです。 Query 最適化プログラムは、プール内の記憶域のこのジョブの持ち分に基づいて、経過時間が最小となる設定を選択します。
	*OPTIMIZE xxx	このオプションは *OPTIMIZE と非常によく似ています。値 xxx は、1 から 200 の間の値から、整数のパーセンテージ値を指定できることを示しています。 Query 最適化プログラムは *OPTIMIZE について実行されるのと同じ処理を使用して、照会の並列度を決定します。いったん決定されたら、最適化プログラムは照会で使用される実際の並列度を、指定されたパーセンテージ分調整します。これによりユーザーは、使用される並行度をある程度、*NUMBER_OF_TASKS の下に特定の並列度を指定せずにオーバーライドできます。
	*MAX	Query 最適化プログラムは、照会を処理するために入出力並列処理または SMP 並列処理の使用を選択します。 SMP 並列処理は、システム・フィーチャーの DB2 Symmetric Multiprocessing (i5/OS 用) がインストール済みの場合にのみ使用されます。 Query 最適化プログラムによる選択は、*OPTIMIZE パラメーター値による選択と類似していますが、Query 最適化プログラムは、プールの中のアクティブ・メモリーのすべてが照会の処理またはデータベース・ファイル・キー順アクセス・パス構築、再構築、または保守に使用できることを前提にしているという違いがあります。
	*NONE	データベース照会処理またはデータベース・テーブル索引構築、再構築、または保守に並列処理は許可されません。
	*NUMBER_OF_TASKS nn	単一の照会で使用できるタスクの最大回数を指示します。この値、またはテーブルに関連付けられたディスク・アームの数で、タスクの回数は打ち切られます。
	*MAX xxx	このオプションは *MAX と非常によく似ています。値 xxx は、1 から 200 の間の値から、整数のパーセンテージ値を指定できることを示しています。 Query 最適化プログラムは *MAX について実行されるのと同じ処理を使用して、照会の並列度を決定します。いったん決定されたら、最適化プログラムは照会で使用される実際の並列度を、指定されたパーセンテージ分調整します。これによりユーザーは、使用される並行度をある程度、*NUMBER_OF_TASKS の下に特定の並列度を指定せずにオーバーライドできます。

表 35. QAQQINI コマンドに指定される照会オプション (続き)

パラメーター	値	説明
PARAMETER_MARKER_CONVERSION 動的 SQL 照会の場合、リテラルがパラメーター・マーカーとしてデータベースによってインプリメントされることを許可するかどうかを指定します。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*NO	定数は、パラメーター・マーカーとして設定することはできません。
	*YES	パラメーター・マーカーとして定数を設定することができます。
QUERY_TIME_LIMIT 照会で処理が必要な見積もり経過秒数に基づいて、開始が許可されるデータベース照会の制限時間を指定します。	*DEFAULT	デフォルト値は、*SYSVAL に設定されます。
	*SYSVAL	このジョブの照会時間制限は、システム値、QQRYTIMLMT から取得されます。
	*NOMAX	見積経過秒数の最大数はありません。
	整数値	照会を実行するために必要な経過時間の見積秒数に対してチェックされる最大値を指定します。見積経過秒数がこの値よりも大きい場合、照会は開始されません。有効な値は、0 から 2,147,352,578 の範囲です。
REOPTIMIZE_ACCESS_PLAN 保管済みのアクセス・プランによる照会の場合は、このオプションは、Query 最適化プログラムが照会を再び最適化するように指定します。 照会は、HLL プログラムの関連記憶域に保管された保管済みアクセス・プランを持つことができます。または最適化プログラム自体によって管理されるプラン・キャッシュに保管することもできます。 注: *NO を指定すれば、照会は、引き続き、再び妥当性検査されます。 この指定が必要な理由として、以下が考えられます。 • 照会されたファイルは削除されて再作成された。 • 照会が、作成時のシステムとは異なるシステムにリストアされた。 • OVRDBF コマンドが使用された。	*DEFAULT	デフォルト値は、*NO に設定されます。
	*NO	既存の照会への再度の最適化は強制しません。ただし、最適化プログラムが最適化が必要であると判断すれば、照会では再び最適化が実行されます。
	*YES	既存の照会への再度の最適化を強制します。
	*FORCE	既存の照会への再度の最適化を強制します。
	*ONLY_REQUIRED	実体のない理由で、プランが再び最適化されないようにします。このような場合、既存のプランは依然として有効な作業可能プランであるため、その使用を継続します。これは、再度の最適化を実行した場合のパフォーマンス上の利点をすべて受けられるわけではないことを意味する場合もあります。実体のない理由には、ファイル・サイズの変更、新しい索引などが含まれます。実体のある理由には、既存のアクセス・プランによって使用されている索引の削除、照会ファイルの削除および再作成などが含まれます。
SQLSTANDARDS_MIXED_CONSTANT SQL 照会の場合、このパラメーターは、IGC 定数を常に IGC-OPEN として処理することが可能かどうかを指定します。注: *NO が指定された場合、DB2 for i5/OS は、他の DB2 プラットフォームとは互換性がありません。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*YES	SQL IGC 定数は、IGC 混用定数として扱われます。
	*NO	IGC 定数のデータがシフトアウトと DBCS データとシフトインだけの場合、その定数は IGC 専用として扱われます。それ以外の場合は IGC 混用として扱われます。

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
SQL_DECFLOAT_WARNINGS このパラメーターは、0 による除算、オーバーフロー、アンダーフロー、無効なオペランド、不正確な結果、または正常以下の数値が関係する SQL DECFLOAT 計算および変換に対して、警告を受ける機能を提供します。	*DEFAULT	デフォルト値は、*NO に設定されます。
	*YES	0 による除算、オーバーフロー、アンダーフロー、無効なオペランド、不正確な結果、または正常以下の数値が関係する DECFLOAT 計算および変換は、呼び出し元に警告を戻します。0 による除算、オーバーフロー、アンダーフロー、または無効なオペランドが関係する DECFLOAT 計算および変換は、呼び出し元にエラーまたはマッピング・エラーを戻します。
	*NO	不正確な結果または正常以下の数値の場合、警告もエラーも戻されません。
SQL_FLAGGER 非標準ステートメントにフラグを立てるかどうかを指定します。このパラメーターを指定すれば、SQL ステートメントにフラグを立て、ISO/IEC 9075-2003 規格のコア・レベルに準拠しているかどうかを検証することができます。	*DEFAULT	デフォルト値は、‘ ‘ に設定されます。これは、フラガーが未設定のオプションであり、これによって SQL 構文解析プログラムが WSQR_DYN_FLG および WSQR_ANS_FLG フラグの値に基づき、構文解析プログラム内で使用するエラー・フラギングのタイプを決定することを示しています。
	*NONE	SQL ステートメントが標準構文に準拠しているかどうかは確認されません。
	*STD	SQL ステートメントが標準構文に準拠しているかどうかは確認されません。
	*DB2	SQL ステートメントがクロスプラットフォーム DB2 SQL 参照に準拠しているかどうかが確認されます。
SQL_FAST_DELETE_ ROW_COUNT この値は、WHERE 文節なしで DELETE FROM テーブル名 SQL ステートメントを処理する際に使用されます。このオプションを使用すれば、ユーザーは、データベース・マネージャーによって削除がインプリメントされる方法を制御できます。	*DEFAULT	デフォルト値は、0 に設定されます。 値が 0 であるということは、従来の削除の代わりに高速削除を使用するかどうかを決定する際にデータベース・マネージャーが考慮する行の数を選択します。デフォルト値を使用する場合、データベース・マネージャーは行数として 1000 行使用するでしょう。これは、値 1000 の INI オプションを使用しても、オプションに 0 を使用するときと比べて操作上の差は生じないことを意味します。
	*NONE	この値は、行に対して高速削除を試行しないようデータベース・マネージャーに強制します。
	*OPTIMIZE	この値は *DEFAULT の使用と同じです。
	整数値	このオプションに値を指定すると、ユーザーは DELETE の動作を調整することができます。高速削除を試行するには、DELETE ステートメントのターゲット・テーブルが、オプションで指定された行の数と一致するかまたは超過していなければなりません。高速削除は個々の行をジャーナルに書き込みません。有効な値は、1 から 999,999,999,999,999 までの範囲です。

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
SQL_PSEUDO_CLOSE V6R1 より前: SQL カーソル・オープン処理は、ジョブのライブラリー・リストに QSQPSCLS1 という名のデータ域があるかどうかをチェックします。データ域が見つかった場合、再利用できるすべてのカーソルは、再使用の候補としてマークを付けられ、アプリケーションによってクローズされる 2 度目ではなく初回に疑似クローズされます。このデータ域がなければ、デフォルトでは、カーソルは 2 度目のクローズまで再利用できる状態になりません。 初回の疑似クローズによって、再使用されない可能性のある一部のカーソルはオープンされたままになります。これによって、アプリケーションに必要な補助および主記憶域が増加する可能性があります。これは、WRKSYSSTS コマンドを使用することによって、モニターできます。使用される補助記憶域の量については、「システム ASP 使用 %」を参照してください。主記憶域の量については、WRKSYSSTS 表示上の障害比率を検査します。 データ域の書式および内容は重要ではありません。データ域は、コマンド: DLTDTAARA DTAARA(QGPL/QSQPSCLS1) を使用して削除することができます。 各ジョブの最初の SQL オープン操作中に、データ域の存在がチェックされます。チェックは一度のみ実行され、処理モードはジョブの存続期間の間、同じままです。ライブラリー・リストは検索用に使用されるので、これらのジョブのライブラリー・リストのみに含まれるライブラリーにデータ域を作成することによって、変更を特定のジョブに分離することができます。	*DEFAULT	デフォルトの動作は、QSQPSCLS1 *DTAARA が存在するかどうかによって異なります。 ジョブ内の最初の OPEN で QSQPSCLS1 *DTAARA が見つかった場合は、SQL カーソルは、再使用の候補としてマークを付けられ、最初のクローズで疑似クローズされます。 ジョブ内の最初の OPEN で QSQPSCLS1 *DTAARA が見つからなかった場合は、SQL カーソルは、再使用の候補としてマークを付けられ、2 番目のクローズで疑似クローズされます。
	整数値	ゼロより大きい値。カーソル・クローズ操作の結果、カーソルを疑似クローズすることによって、カーソルに再使用の候補としてマークが付けられる場合を示しています。このオプションの値から 1 を引いた値は、カーソルが疑似クローズの候補としてマークを付けられるまでにハード・クローズされる回数を示します。有効な値は 1 から 65535 までです。
SQL_STMT_COMPRESS_MAX ユーザーは、圧縮最大設定値を設定することができます。これは、ステートメントがパッケージに準備される際に使用されます。	*DEFAULT	デフォルト値は 2 に設定されています。これは、ステートメントが実行されずに 2 度圧縮された後、いずれかのステートメントと関連付けられているアクセス・プランが除去されることを示しています。
	整数値	整数値は、アクセス・プランが除去されてパッケージに追加のスペースが作成される前にステートメントが圧縮される回数を表します。 SQL ステートメントを実行すると、そのステートメントのカウントは 0 にリセットされます。有効な整数値は 1 から 255 です。

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
SQL_STMT_REUSE ステートメントが SQL 拡張動的パッケージに保管されるまでにステートメントを再使用することの必要な回数を指定します。ステートメントが再使用された回数が、指定された INI オプションより少ない場合、ステートメントの一時コピーが照会に使用され、ステートメントを準備中の他のすべてのジョブも完全な準備を実行します。	*DEFAULT	デフォルト値は 0 です。ステートメントはステートメントを最初に準備する際に保管されます。
	整数値	この整数値は、ステートメントを SQL パッケージに保管する前に、ステートメントを再使用することの必要な回数を表します。有効な整数値は 1 から 255 までです。
SQL_SUPPRESS_WARNINGS SQL ステートメントの場合、このパラメーターは SQL 警告を抑制する機能を提供します。	*DEFAULT	デフォルト値は、*NO に設定されます。
	*YES	ステートメントの実行後、SQLCA の SQLCODE を調べます。 SQLCODE が + 30 の場合、SQLCA を変更し、呼び出し元に警告が戻らないようにします。 SQLCODE を 0、SQLSTATE を '00000'、SQLWARN を ' ' に設定します。
	*NO	SQL 警告が呼び出し元に戻るように指定します。
SQL_TRANSLATE_ASCII_ TO_JOB DRDA® を使用して、アプリケーション・サーバー (AS) として iSeries に接続する (この場合、アプリケーション・リクエスター (AR) マシンは ASCII ベースのプラットフォーム) 際に、このパラメーターは、AS 上の SQL ステートメント・テキストをジョブの CCSID に変換する機能を提供します。	*DEFAULT	デフォルト値は、*NO に設定されます。
	*YES	ASCII SQL ステートメント・テキストを i5/OS ジョブの CCSID に変換します。
	*NO	ASCII SQL ステートメント・テキストを ASCII CCSID と関連した EBCIDIC CCSID に変換します。
STAR_JOIN (注を参照してください) ハッシュ結合用の拡張最適化は、ハッシュ・テーブルおよび値の特殊なリストの両方がデータから構成される場所を照会します。この値の特殊なリストは、次にハッシュ結合の 1 次テーブルに対する選択に追加されます。 これによって、これらの (外部キー) 列上に作成される、あらゆる EVI 索引は、結合値をマッチングさせる前にテーブルに対するビットマップ選択を実行するために使用できます。 このオプションを使用しても、必ず最適化プログラムによって選択されるとは限りません。最適化プログラムがハッシュ結合を使用して照会をインプリメントすることを既に決定している場合にのみ、この手法の使用が許可されます。	*DEFAULT	デフォルト値は、*NO に設定されます。
	*NO	EVI 星形結合最適化サポートは使用可能になりません。
	*COST	Query 最適化が EVI 星形結合サポートの使用を考慮する (コストを見積もる) ことができるようにします。 個別リスト選択を使用するかどうかの決定は、それを使用することにどれほどの利点があるかに基づき、最適化プログラムが下します。

表 35. *QAQQINI* コマンドに指定される照会オプション (続き)

パラメーター	値	説明
STORAGE_LIMIT データベース照会用の一時記憶の限界を指定します。照会が指定された量よりも多い記憶域を使用することが予期される場合は、照会の実行は許可されません。指定される値はメガバイト単位です。	*DEFAULT	デフォルト値は、*NOMAX に設定されます。
	*NOMAX	ストレージの問題により照会の実行を停止しません。
	整数値	照会が使用できる一時ストレージの最大メガバイト数。この値は、Query 最適化プログラムによって計算される、照会の実行に必要な一時ストレージの見積もり値に対して検査されます。一時ストレージの見積もり値がこの値よりも大きい場合、照会は開始されません。有効な値は、0 から 2147352578 の範囲です。
SYSTEM_SQL_STATEMENT_CACHE SQL 照会の場合、このパラメーターは、システム・ワイド SQL ステートメント・キャッシュを無効にする機能を提供します。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*YES	SQL 作成要求が処理されるとき、システム規模の SQL ステートメント・キャッシュを調べます。一致するステートメントがすでにキャッシュに存在する場合は、その作成の結果を使用します。これにより、アプリケーションの実行する作成のパフォーマンスが向上する可能性があります。
	*NO	SQL 作成要求が処理されるとき、システム規模の SQL ステートメント・キャッシュを調べません。
UDF_TIME_OUT (注を参照してください) ユーザー定義関数 (UDF) が処理を完了するのをデータベースが待機する時間量を秒単位で指定します。	*DEFAULT	待機時間量は、データベースによって決定されます。デフォルトは、30 秒です。
	*MAX	データベースが UDF の終了を待機する最大時間量。
	整数値	データベースが UDF の終了を待機すべき秒数を指定します。指定した値がデータベースの最大待ち時間を超えている場合、最大待ち時間がデータベースによって使用されます。最小値は 1 で、最大値はシステム定義になります。
VARIABLE_LENGTH_OPTIMIZATION 可変長列の積極的な最適化手法が許可されます。	*DEFAULT	デフォルト値は、*YES に設定されます。
	*YES	索引専用アクセスを含め、可変長列の積極的な最適化を有効にします。さらに、可変長列 (複数も可) に対する等号述部が存在する場合、定数値による置換も許可されます。この置換の副次作用は、可変長列 (複数も可) に戻されるデータの長さが、オリジナル・データの末尾ブランクと一致しない可能性がある点です。結果として、アプリケーションは、オリジナル・データの代わりに置換された値を受け取ることができ、関数呼び出しは、オリジナルのストリング値ではなく、置換された値に関して機能する場合があります。
	*NO	可変長列の積極的な最適化は許可されません。

注: Classic Query Engine での環境のみを変更します。

予測照会管理プログラムによるリソース制限の設定

DB2 for i5/OS 予測照会管理プログラムは、照会の見積もった実行時間 (経過実行時間) または見積一時ストレージを超過した場合、照会の開始を停止させることができます。管理プログラムは、照会の実行中ではなく照会を実行する*前に* 作動します。管理プログラムは、システムのどの対話式ジョブまたはバッチ・ジョブにおいても使用できます。このプログラムは、すべての DB2 for i5/OS 照会インターフェースで使用でき、SQL 照会での使用に限定されることはありません。

照会が開始される前に管理プログラムがその予測と停止を行えることは、次の理由から重要です。

- 長時間照会の操作および何らかの結果を得る前に照会を異常終了することは、システム・リソースの浪費です。

- 照会内部の CQE 操作の中には、要求終了 (ENDRQS) CL コマンドで中断することができないものがあります。一時索引の作成または GROUP BY 文節を含まない列関数を使用する照会が、これらのタイプの照会の 2 つの例です。 これらの操作がユーザーが容認する待ち時間よりも長い場合には、これらの操作を開始しないことが重要です。

DB2 for i5/OS における管理プログラムは、以下の 2 つの測定値に基づきます。

- 照会の見積実行時間。
- 照会の見積一時ストレージ使用量。

照会の見積実行時間または一時ストレージの使用量がユーザーの定義した制限を超える場合は、照会の開始を停止することができます。

管理プログラムが使用する時間制限 (秒単位) を定義するには、次のいずれか 1 つを行います。

- 照会属性の変更 (CHGQRYA) CL コマンドで照会時間制限 (QRYTIMLMT) パラメーターを使用します。これは、Query 最適化プログラムが時間制限の検出を試みる最初の場所です。
- 照会オプション・ファイルに照会時間制限オプションを設定します。これは、Query 最適化プログラムが時間制限をする 2 番目の場所です。
- QQRYTIMLMT システム値を設定します。これは、各ジョブが 照会属性の変更 (CHGQRYA) CL コマンドで値 *SYSVAL を使用できるようにします。さらに、照会オプション・ファイルを *DEFAULT に設定します。これは、Query 最適化プログラムが時間制限をする 3 番目の場所です。

管理プログラムが使用する一時ストレージ制限 (メガバイト単位) を定義するには、以下を行います。

- 照会属性の変更 (CHGQRYA) CL コマンドで照会ストレージ制限 (QRYSTGLMT) パラメーターを使用します。これは、Query 最適化プログラムが制限をする最初の場所です。
- 照会オプション・ファイルに照会ストレージ制限オプション STORAGE_LIMIT を設定します。これは、Query 最適化プログラムが時間制限をする 2 番目の場所です。

最適化プログラムによって生成される時間および一時ストレージの値は、*単に* 見積もりにすぎないことに留意する必要があります。実際の照会の実行時間は、見積もりを上回ることも下回ることもあります。照会されるデータに関する十分な情報を最適化プログラムが持たない場合、見積もりは実際に使用されるリソースと大いに異なる場合があります。そのような場合、不正確な見積もりに合わせて、制限を人為的に調整する必要があるかもしれません。

システム全体に対して時間制限を設定する場合、通常、任意の照会を実行するための最大許容時間に時間制限を設定することが最善です。時間制限の設定が小さすぎると、完了できない照会が出てきて、結果的にアプリケーションを正常に終了できないというリスクを負うことになります。照会要求を内部的に実行するために照会コンポーネントを使用している関数が多数あります。これらの要求もユーザーが定義した時間制限と比較されます。

予測実行時間およびストレージについて照会メッセージ CPA4259 をチェックすることができます。照会が取り消された場合でも、やはりデバッグ・メッセージがジョブ・ログに書き込まれます。

見積実行時間および一時ストレージ制限が指定の制限を超過したときに呼び出される照会管理プログラムの出口プログラムを追加することもできます。

関連情報

Query Governor Exit Program

要求終了 (ENDRQS) コマンド

照会属性の変更 (CHGQRYA) コマンド

照会管理プログラムの使用:

リソース管理プログラムは、Query 最適化プログラムと一緒に作動します。

ユーザーが照会を実行する要求をシステムに発行すると、次のことが実行されます。

1. 最適化プログラムが照会アクセス・プランを作成します。

   評価の一環として、最適化プログラムは照会の実行時間の予測または見積もりを行います。これにより、照会を行うためにデータへのアクセスおよび検索を行う最適な方法を決定することができます。さらに、見積プロセスの一部として、最適化プログラムは照会の見積一時ストレージ使用量の計算も行います。
2. 見積実行時間および見積一時ストレージは、ジョブまたはユーザー・セッションに対して現在有効となっているユーザー定義の照会制限と比較されます。
3. 照会の見積もりが指定の制限以下の場合、照会管理プログラムによって照会が中断されることなく実行できるので、ユーザーに対してメッセージは出されません。
4. 照会制限を超える場合は、照会メッセージ CPA4259 がユーザーに対して出されます。メッセージには見積もりと指定の制限が含まれています。 1 つの制限のみ超過している必要があることにご注意ください。 1 つの制限のみが超過したことに気付くかもしれません。また、制限がユーザーによって明示的に指定されなかった場合、その制限に対して長精度整数の値が示されます。

   注: ユーザーがこのメッセージに対して応答しなくても済むように、このメッセージにデフォルト応答を設定することができます。これにより、この照会要求は、*常に* 終了します。
5. デフォルトのメッセージ応答が使用されない場合は、次のいずれか 1 つを選択して実行します。
   - 実際に実行する前に照会要求を終了する。
   - たとえ関連した管理プログラムの制限を見積値が超えても照会を継続して実行する。

現行のジョブ以外のジョブに関するリソース制限の設定

現行ジョブ以外のジョブに対していずれかまたは両方のリソース制限を設定することができます。これを行うには、照会属性の変更 (CHGQRYA) コマンドの JOB パラメーターを使用して、検索する照会オプション・ファイル・ライブラリー (QRYOPTLIB) を指定するか、またはそのジョブ用の特定の QRYTIMLMT、または QRYSTGLMT あるいはその両方を指定します。

システム・リソースの平衡化のためのリソース制限の使用

ソース・ジョブが 照会属性の変更 (CHGQRYA) コマンドを実行後、管理プログラムのターゲット・ジョブへの効果は、ソース・ジョブには依存しません。照会リソース制限は、ジョブまたはユーザー・セッションを実行している間、またはリソース制限を 照会属性の変更 (CHGQRYA) コマンドで変更するまで引き続き有効となります。プログラム制御によって、実行しているアプリケーション機能、時刻、または利用できるシステム・リソースによって、ユーザーに与えられる制限が異なります。これにより、システム・リソースを一時的な照会要件とバランスをとろうとするときにかなりの柔軟性がもたらされます。

照会管理プログラムによる照会の取り消し:

設定された制限よりも照会がより多くのリソースを使用することが予測される場合、管理プログラムが照会メッセージ CPA4259 を出します。

次のいずれかの方法で、メッセージに応答することができます。

- 照会を取り消すには、C と入力します。SQL 実行時間コードにエスケープ・メッセージ CPF427F が出されます。 SQL は、SQLCODE -666 を返します。

| • 超過したこの制限を無視して、照会が最後まで行われるようにするには、I と入力します。

照会管理プログラムの照会メッセージに対するデフォルト応答の制御:

システム管理者は、ジョブの変更 (CHGJOB) CL コマンドを使用することによって、対話式ユーザーがデータベース照会の照会メッセージを無視するオプションをもつようにするかどうかを制御することができます。

行われる変更には以下が含まれます。

- *DFT の値を ジョブの変更 (CHGJOB) CL コマンドの INQMSGRPY パラメーターに指定すると、対話式ユーザーに対して照会メッセージが表示されず、照会が直ちに取り消されます。
- *RQD の値を ジョブの変更 (CHGJOB) CL コマンドの INQMSGRPY パラメーターに指定すると、対話式ユーザーに対して照会が表示され、ユーザーはこの照会に対して応答しなければなりません。
- *SYSRPYL の値を ジョブの変更 (CHGJOB) CL コマンドの INQMSGRPY パラメーターに指定すると、対話式ユーザーに対して照会を表示するかどうか、ならびに応答を必要とするかどうかを決めるためにシステム応答リストが使用されます。システム応答リスト項目を使用して、ユーザー・プロファイル名、ユーザー ID 名、または処理名に基づき、さまざまなデフォルト応答をカスタマイズすることができます。照会メッセージ CPA4259 のために、メッセージ・データの中で完全修飾ジョブ名を利用することができます。これにより、キーワード CMPDTA を使用して、処理またはユーザー・プロファイルに適用されるシステム応答リスト項目を選択することができます。ユーザー・プロファイル名は長さ 10 文字で、51 桁から始まります。処理名は長さ 10 文字で、27 桁から始まります。
- 以下の例は、デフォルト応答の C にユーザー・プロファイルが「QPGMR」のジョブに関する要求を取り消される、応答リスト・エレメントを加えます。

```
ADDRPYLE  SEQNBR(56) MSGID(CPA4259) CMPDTA(QPGMR      51) RPY(C)
```

以下の例は、デフォルト応答の C にプロセス名が「QPADEV0011」のジョブに関する要求を取り消される、応答リスト・エレメントを加えます。

```
ADDRPYLE  SEQNBR(57) MSGID(CPA4259) CMPDTA(QPADEV0011 27) RPY(C)
```

関連情報

ジョブ変更 (CHGJOB) コマンド

照会管理プログラムによるパフォーマンスのテスト:

照会管理プログラムを使用して、照会のパフォーマンスをテストすることができます。

照会管理プログラムで照会のパフォーマンスをテストするには、以下を実行します。

1. 照会属性の変更 (CHGQRYA) コマンドを使用して、または INI ファイル内で、照会の時間制限をゼロに設定します (QRYTIMLMT(0))。これは、照会実行の見積時間が照会の時間制限を超えることを知らせる照会メッセージを、管理プログラムから出させます。
2. 照会メッセージでメッセージ・ヘルプのプロンプトを出し、SQL 情報印刷 (PRTSQLINF) コマンドを実行して見つけるものと同じ情報を見つけることができます。

照会管理プログラムを使用すると、照会を何回も反復して実行しなくても済むようパフォーマンスを最適化することができます。

さらに、照会を取り消した場合は、Query 最適化プログラムがアクセス・プランを評価して、最適化プログラム・デバッグ・メッセージをジョブ・ログに送ります。これは、ジョブがデバッグ・モードになっていなくても 行われます。これにより、ジョブ・ログ内の最適化プログラム・チューニング・メッセージを検討して、最適な照会パフォーマンスを確保するためにチューニングがさらに必要かどうか調べることができま

| す。これによって、さまざまな属性、索引、および構文またはそのすべてを指定することによる照会のさまざまな組み合わせを試行して、実際には照会が完了まで実行しなくても、最適化プログラムで、どのパフォーマンスがより優れているかの判別に役立てることができます。データの実際の照会は事実上まったく行われないので、システム・リソースの節約になります。照会されるテーブルに多数の行が含まれる場合は、これはシステム・リソースの大きな節約になります。

| すべての照会要求が実行される前に停止されることになるため、パフォーマンス・テスト用にこの技法を使用する場合には十分に注意してください。 このことは、単一照会ステップに収めることができない CQE 照会の場合には、特に重要です。この種の照会の場合、分離した複数の照会要求を出して、最終結果を戻す前にこれらの結果を累積します。これらの中間ステップの 1 つで照会を停止すると、その中間ステップに関するパフォーマンス情報だけが戻され、照会全体については戻されません。

| **関連情報**

| SQL 情報印刷 (PRTSQLINF) コマンド

| 照会属性の変更 (CHGQRYA) コマンド

| **照会時間制限の設定の例:**

| 照会オプション・ファイル QAQQINI を使用する、現行ジョブまたはユーザー・セッションに照会時間制限を設定するには、パラメーターに QUERY_TIME_LIMIT が設定された QAQQINI ファイルが入っているユーザー・ライブラリーに対する照会属性の変更 (CHGQRYA) コマンドに QRYOPTLIB パラメーターを指定します。

| 照会時間制限を 45 秒に設定するには、次の 照会属性の変更 (CHGQRYA) コマンドを使用できます。

```
| CHGQRYA   JOB(*) QRYTIMLMT(45)
```

| これにより、照会時間制限が 45 秒に設定されます。 見積実行時間が 45 秒以下の短い照会を実行する場合、その照会は中断することなく実行されます。指定された照会時間制限は、ジョブまたはユーザー・セッションを実行している間、または時間制限を 照会属性の変更 (CHGQRYA) コマンドで変更するまで引き続き有効となります。

| Query 最適化プログラムが照会の実行時間を 135 秒と見積もったと想定してください。 見積もった 135 秒の実行時間が照会時間制限の 45 秒を超えていることを記述するメッセージがユーザーに送られます。

| 現行ジョブ以外のジョブに関する照会時間制限の設定または変更を行うには、JOB パラメーターを使用して 照会属性の変更 (CHGQRYA) コマンドを実行します。 照会時間制限 45 秒をジョブ 123456/USERNAME/JOBNAME について設定するには、以下の 照会属性の変更 (CHGQRYA) コマンドを使用します。

```
| CHGQRYA   JOB(123456/USERNAME/JOBNAME) QRYTIMLMT(45)
```

| これは照会時間制限をジョブ 123456/USERNAME/JOBNAME について 45 秒に設定します。ジョブ 123456/USERNAME/JOBNAME が 45 秒以下の見積実行時間で照会を実行すると、照会は中断なしに実行されます。照会の見積実行時間が 45 秒より長い場合は (たとえば、50 秒)、見積実行時間 50 秒は照会時間制限 45 秒を超える旨のメッセージがユーザーに送られます。ジョブ 123456/USERNAME/JOBNAME の継続中、またはジョブ 123456/USERNAME/JOBNAME の時間制限が 照会属性の変更 (CHGQRYA) コマンドによって変更されるまで引き続き有効となります。

| QQRYTIMLMT システム値への照会時間制限を設定または変更するには、次のように 照会属性の変更 (CHGQRYA) コマンドを使用します。

```
| CHGQRYA QRYTIMLMT(*SYSVAL)
```

| この QQRYTIMLMT システム値は、ジョブまたはユーザー・セッションの間中、または時間制限が 照会属性の変更 (CHGQRYA) コマンドで変更されるまで使用されます。 これは、照会属性の変更 (CHGQRYA) コマンドの場合のデフォルトの動作です。

| **注:** 照会の時間制限も、INI ファイルで設定するか、システム値変更 (CHGSYSVAL) コマンドを使用して 設定できます。

| **関連情報**

| 照会属性の変更 (CHGQRYA) コマンド

| システム値変更 (CHGSYSVAL) コマンド

| **照会管理プログラムによる一時記憶使用量のテスト:**

| 予測記憶管理プログラムにより、データベース照会用の一時記憶の限界が指定されます。照会管理プログラムを使用することで、ハッシュ・テーブル、ソート、一時索引などの一時的なオブジェクトが、照会の実行に使用されているかテストできます。

| 一時的オブジェクトの使用状況をテストするには、次のようにします。

| • 照会属性の変更 (CHGQRYA) コマンドを使用して、または INI ファイル内で、照会の記憶限界をゼロに設定します (QRYSTGLMT(0))。これにより、一時オブジェクトの見積もりサイズに関係なく、照会に一時オブジェクトが使用されるときはいつでも、管理プログラムから照会メッセージが送信されるようになります。

| • 照会メッセージでメッセージ・ヘルプのプロンプトを出し、SQL 情報印刷 (PRTSQLINF) コマンドを実行して見つけるものと同じ情報を見つけることができます。これにより、関係する一時オブジェクトを、ユーザーが確認できるようになります。

| **関連情報**

| SQL 情報印刷 (PRTSQLINF) コマンド

| 照会属性の変更 (CHGQRYA) コマンド

| **照会の一時記憶限界の設定の例:**

| 一時記憶限界は、QAQQINI ファイルまたは、照会属性の変更 (CHGQRYA) コマンドで指定できます。

| 照会オプション・ファイル QAQQINI を使用して、ジョブに照会の一時記憶限界を設定するには、パラメーター STORAGE_LIMIT の有効な値セットと共に、QAQQINI ファイルが入っているユーザー・ライブラリーに対し、照会属性の変更 (CHGQRYA) コマンドで QRYOPTLIB パラメーターを指定します。

| 照会属性の変更 (CHGQRYA) コマンド自体に照会の一時記憶限界を設定するには、QRYSTGLMT パラメーターに有効な値を指定してください。

| 値が 照会属性の変更 (CHGQRYA) コマンドの QRYSTGLMT パラメーターと、QRYOPTLIB パラメーターに指定された QAQQINI ファイルの両方に指定されている場合、QRYSTGLMT の値が使用されます。

| 現行ジョブで一時記憶限界に 100 MB と指定するには、次に示す照会属性の変更 (CHGQRYA) コマンドを使用できます。

```
| CHGQRYA JOB(*) QRYSTGLMT(100)
```

| 見積もり一時記憶使用量が 100 MB 以下の照会を実行する場合は、その照会は中断することなく実行されます。見積もりが 100 MB を超える場合、データベースにより CPA4259 照会メッセージが送信されま

| す。現行ジョブ以外のジョブに関する照会時間制限の設定または変更を行うには、JOB パラメーターを使
| 用して CHGQRYA コマンドを実行します。 ジョブ 123456/USERNAME/JOBNAME に同じ限界を設定す
| るには、以下の CHGQRYA コマンドを使用します。

```
| CHGQRYA JOB(123456/USERNAME/JOBNAME) QRYSTGLMT(100)
```

| これでジョブ 123456/USERNAME/JOBNAME に対し、照会の一時記憶限界が 100 MB に設定されます。

| **注:** 照会の時間制限とは異なり、一時記憶限界にシステム値はありません。デフォルトの動作では、一時記
| 憶使用量に関係なく、すべての照会が実行されます。照会の一時記憶限界は、INI ファイルまたは、照
| 会属性の変更 (CHGQRYA) コマンドで指定できます。

| **関連情報**

| 照会属性の変更 (CHGQRYA) コマンド

照会の並列処理の制御

使用可能な並列処理には以下の 2 つのタイプがあります。 1 つは、無料で使用できる並列入出力です。もう 1 つは購入可能なフィーチャー、DB2 Symmetric Multiprocessing です。並列処理をオン/オフにすることができます。

システムあるいは与えられたジョブについては並列処理が使用できますが、ジョブの中で実行する個々の照会は、実際には並列方式を使用していない場合があります。これは機能的な制限があるためか、または最適化プログラムが速く処理できると判断して、並列でない方式のほうを選択した場合です。

並列アクセス方式で処理される照会は、主記憶域、CPU、およびディスク・リソースを積極的に使用するので、並列処理を用いる照会の数は限定して管理する必要があります。

照会のシステム規模の並列処理の制御:

QQRYDEGREE システム値は、システムの並列処理を制御するために使用できます。

システム値の現行値は、以下の CL コマンドを使用して表示または変更できます。

- WRKSYSVAL - システム値処理
- CHGSYSVAL - システム値変更
- DSPSYSVAL - システム値表示
- RTVSYSVAL - システム値検索

QQRYDEGREE の特殊値は、デフォルトがシステムの全ジョブについて並列処理を許可するかどうかを管理します。次の値が指定可能です。

***NONE**
データベース照会処理に並列処理は許可されません。

***IO**
照会で入出力並列処理が許可されます。

***OPTIMIZE**
Query 最適化プログラムは、照会を処理するために入出力並列処理または SMP 並列処理について、任意の number-of-tasks (タスク数) を使用することを選択できます。 SMP 並列処理は、DB2 Symmetric Multiprocessing フィーチャーがインストール済みの場合にのみ、使用されます。 Query 最適化プログラムは並列処理の使用を選択し、ジョブがプール内のメモリーを共用することに基づいて経過時間を最小化します。

***MAX**
Query 最適化プログラムは、照会を処理するために入出力並列処理または SMP 並列処理の使用を選択できます。 SMP 並列処理は、DB2 Symmetric Multiprocessing フィーチャーがインストール済みの場合にのみ、使用することができます。 Query 最適化プログラムによる選択は、*OPTIMIZE パラメーター値による選択と類似していますが、最適化プログラムは、プールの中のアクティブ・メモリーのすべてが照会の処理に使用できることを前提にしているという違いがあります。

QQRYDEGREE システム値のデフォルト値は *NONE なので、並列照会処理をシステムでのジョブ実行のデフォルトにする場合には、この値を変更する必要があります。

このシステム値の変更は、DEGREE 照会属性が *SYSVAL のシステムで実行予定の、あるいは現在実行中の全ジョブに影響を与えます。しかし、すでに開始済みの照会、あるいは再使用可能 ODP を使用した照会は、影響を受けません。

照会のジョブ・レベルの並列処理の制御:

照会属性の変更 (CHGQRYA) コマンドの DEGREE パラメーターを使用して、または QAQQINI ファイル内で、あるいは SET_CURRENT_DEGREE SQL ステートメントを使用して、ジョブ・レベルで照会並列処理を制御することもできます。

照会属性の変更 (CHGQRYA) コマンドの使用

許可される並列処理オプション、および任意で、ジョブ内でデータベース照会を実行中に使用できるタスク数を指定できます。 DEGREE 照会属性の現行値を表示する対話式ジョブに、照会属性の変更 (CHGQRYA) コマンドで指定できます。

DEGREE 照会属性の変更は、すでに開始済みの照会あるいは再使用可能 ODP を使用した照会には影響しません。

DEGREE キーワードのパラメーター値は次のとおりです。

***SAME**
並列度照会属性は、変更しません。

***NONE**
データベース照会処理に並列処理は許可されません。

***IO**
データベース Query 最適化プログラムが照会に入出力並列処理の使用を選択するときに、使用できるタスク数。 SMP 並列処理は許可されません。

***OPTIMIZE**
Query 最適化プログラムは、照会を処理するために入出力並列処理または SMP 並列処理について、任意の number-of-tasks (タスク数) を使用することを選択できます。 SMP 並列処理は、DB2 Symmetric Multiprocessing フィーチャーがインストール済みの場合にのみ、使用することができます。並列処理の使用および使用されるタスク数が決定される要因は、システムで使用できるプロセッサー数、ジョブが実行されるプール内でジョブが使用できるアクティブ・メモリーの共用量、および照会に予期される経過時間が CPU 処理または入出力リソースにより制限されるかどうかについてです。 Query 最適化プログラムはジョブがプール内の記憶域を共用することに基づいて経過時間の最小化を実施します。

***MAX**
Query 最適化プログラムは、照会を処理するために入出力並列処理または SMP 並列処理の使用を選択できます。 SMP 並列処理は、DB2 Symmetric Multiprocessing フィーチャーがインストール済みの場合にのみ、使用することができます。 Query 最適化プログラムが行う選択は、*OPTIMIZE パラメータ

一値が行う選択と類似していますが、 Query 最適化プログラムは、プールの中のアクティブ記憶域のすべてが照会の処理に使用できることを前提にしているという違いがあります。

***NBRTASKS** *number-of-tasks*
Query 最適化プログラムが照会を処理するために、 SMP 並列処理の使用を選択するときに使用されるタスク数を指定します。入出力並列処理も許可されます。 SMP 並列処理は、DB2 Symmetric Multiprocessing フィーチャーがインストール済みの場合にのみ、使用することができます。

使用するタスクの数が、システムで使用できるプロセッサー数より少ないと、特定の照会の実行で同時に使用されるプロセッサー数が制限されます。タスク数を多くすると、照会はシステムで照会実行に使用できる全プロセッサーを使用することができます。タスク数が多すぎると、アクティブ・メモリーのオーバー・コミットメントと全タスクを管理するオーバーヘッド・コストにより、パフォーマンスが落ちます。

***SYSVAL**
使用される処理オプションを QQRYDEGREE システム値の現行値に設定する必要があることを指定します。

ジョブのための DEGREE 属性の初期値は *SYSVAL です。

SET CURRENT DEGREE SQL ステートメントの使用

SET CURRENT DEGREE SQL ステートメントを使用して、CURRENT_DEGREE 特殊レジスターの値を変更することができます。 CURRENT_DEGREE 特殊レジスターに指定可能な値は次のとおりです。

1 並列処理は許可されません。

2 から 32767
使用される並列処理の度合いを指定します。

ANY
データベース・マネージャーは、入出力並列処理または SMP 並列処理について、任意の number-of-tasks (タスク数) を使用することを指定できます。並列処理の使用および使用されるタスク数が決定される要因は、システムで使用できるプロセッサー数、このジョブが実行されるプール内でジョブが使用できるアクティブ・メモリーの共用量、および操作に予期される経過時間が CPU 処理または入出力リソースにより制限されるかどうかについてです。データベース・マネージャーはジョブがプール内の記憶域を共用することに基づいて経過時間の最小化を実施します。

NONE
並列処理は許可されません。

MAX
データベース・マネージャーは、入出力並列処理または SMP 並列処理について、任意の number-of-tasks (タスク数) を使用することができます。 MAX は ANY と類似していますが、プール内のすべてのアクティブ・メモリーを使用できることをデータベース・マネージャーが前提とするという点で異なります。

IO データベース・マネージャーが照会に入出力並列処理の使用を選択するときに、使用できるタスク数。 SMP は許可されません。

値は SET CURRENT DEGREE ステートメントを呼び出すことによって変更できます。

CURRENT DEGREE の初期値は、CHGQRYA CL コマンド、現行照会オプション・ファイル (QAQQINI) の PARALLEL_DEGREE パラメーター、または QQRYDEGREE システム値から現時点でどれだけ影響を受けているかによって決まります。

関連情報

Set Current Degree statement

照会属性の変更 (CHGQRYA) コマンド

統計マネージャーによる統計の収集

先に言及したように、統計の収集は統計マネージャーと呼ばれる別個のコンポーネントが処理します。統計情報を Query 最適化プログラムが使用して、照会の最適アクセス・プランを判別することができます。Query 最適化プログラムのアクセス・プランの選択はテーブルにある統計情報に基づいているため、この情報が現行のものとなっていることが重要です。

多くのプラットフォーム上では、統計収集は手動のプロセスであり、データベース管理者がその責任を持ちます。 System i 製品では、データベース統計収集プロセスは自動的に実行され、統計を手動で更新する必要はほとんどありません。

統計マネージャーは実際には照会を実行または最適化しません。統計マネージャーは、メタデータへのアクセス、および照会を最適化するのに必要な他の情報へのアクセスを制御します。統計マネージャーはこの情報を使用して、Query 最適化プログラムが提出する質問に応答します。応答は、テーブル・ヘッダー情報から、既存の索引から、または単一列統計から取り出すことができます。

統計マネージャーは、必ず最適化プログラムから質問に応答する必要があります。応答を備えるのに使用できる最適の方法を使用します。例えば、単一列統計を使用するかもしれませんし、索引に対してキー範囲見積もりを実行する可能性もあります。応答と共に、サイジング・アルゴリズムにいっそう裁量を持たせるために最適化プログラムが使用できる信頼度を、統計マネージャーは最適化プログラムに戻します。グループ要求用に見積もるグループの数に関して統計マネージャーの信頼度が低い場合、最適化プログラムは割り振られる一時ハッシュ・テーブルのサイズを増加させることもあります。

関連概念

4 ページの『統計マネージャー』

CQE では、統計の検索は最適化プログラムの機能です。テーブルについての情報を得る必要がある場合、最適化プログラムはテーブル記述を参照して行数とテーブル・サイズを取得します。索引が使用できる場合、最適化プログラムはテーブル内のデータに関する情報を抽出する場合もあります。 SQE では、統計の収集および管理は統計マネージャーと呼ばれる別個のコンポーネントが処理します。統計マネージャーは、CQE とまったく同じ統計的なソースの効力を高めますが、さらにより多くのソースおよび機能を追加します。

自動統計収集

統計マネージャーが最適化プログラムへの応答を準備すると、(列統計または索引は使用できなかったため) デフォルトのフィルター係数を使用して生成される応答のトラックを維持します。この情報は、列の統計収集要求を自動的に生成するためアクセス・プランがプラン・キャッシュに書き込まれる際に使用されます。システム・リソースがそれを許可する場合、統計マネージャーは最適化プログラムへのデフォルトの応答を避け、現行の照会で直接使用するための統計集計をリアルタイムで生成します。

これが行われない場合は、システム・リソースの使用可能時に、要求された列統計はバックグラウンドで収集されます。そのように、照会が次に実行される際、欠落していた列統計が統計マネージャーで使用できるようになり、最適化プログラムにその時点でのより正確な情報を提供できるようになります。さらに統計を作成すれば、それだけ最適化プログラムがアクセス・プランをより良く実行できるようになります。

照会が実行前または実行中に取り消される場合、実行によって生成されたアクセス・プランがプラン・キャッシュに書き込まれるようになるまでは、列統計への要求は引き続き処理されます。

統計収集の際に 1 つのテーブルをパススルーする回数を最小化するには、統計マネージャーは同一テーブルに対する複数の要求をまとめてグループ化します。例えば、2 つの照会がテーブル T1 に対して実行されるとします。最初の照会は列 C1 に選択基準があり、2 番目の照会は列 C2 にあります。このテーブルで使用できる統計がない場合、統計マネージャーはこれらの両方の列を列統計の有効な候補と識別します。統計マネージャーが要求を検討する場合には、同じテーブルに対する複数の要求を調べ、それらをまとめて 1 つの要求にグループ化します。これにより、テーブル T1 を 1 回だけパススルーするように 2 つの列統計を作成できます。

注目すべき事柄の 1 つは、デフォルトのフィルター係数を使用して最適化プログラムから統計マネージャーが質問に応答する必要のある場合には、列統計は通常自動的に作成されるということです。しかし、応答を生成するのに用いる場合のある索引が使用可能なときには、列統計は自動的には生成されません。最適化プログラムから質問に応答するために列統計を使用する方が、索引データを使用するよりも一般に効率的なので、このシナリオでは、列統計の使用によって最適化時間が恩恵を受けます。ですから照会パフォーマンスが悪くなったように思える場合には、照会内に関連のある複数の列を持つ複数の索引がないか検証することもできます。そのような索引がある場合には、こうした列の列統計を手動で生成してみてください。

前述のように、システム・リソースが使用可能になると統計収集が生じます。システム上で永続的にアクティブで、処理用にすべてのスペア CPU サイクルを使用することが予想される優先順位の低いジョブをスケジュールしている場合には、統計収集は決してアクティブにはなりません。

自動統計の最新表示

基礎になっているテーブル・データが変更されると、列統計は保守されません。統計マネージャーは、列統計が依然として有効か、またはもはや列を正確に表していないか (失効) を判別します。

この妥当性検査は、以下のいずれかが生じるごとに実行されます。

- アクセス・プランを作成するのに列統計が使用された照会に対してフル・オープンが生じる
- 全く新しい照会が最適化されたためか、または既存のプランが再最適化されたため、新規プランがプラン・キャッシュに追加される

統計を妥当性検査するには、統計マネージャーは以下の事柄が適用されるかどうかをチェックします。

- テーブル内の行数が、テーブル行カウント合計の 15% を超えて変更された
- 変更されたテーブル内の行数が、テーブル行カウント合計の 15% を超えた

統計が失効したと判別されると、統計マネージャーは引き続き失効した列統計を使用して最適化プログラムからの質問に応答しますが、同時にプラン・キャッシュで列統計は失効としてマークを付けられ、統計を最新表示するための要求が生成されます。

統計要求の表示

System i ナビゲーターを使用することにより、または統計 API を使用することにより、現行の統計要求を表示できます。

System i ナビゲーターで要求を表示するには、「**データベース**」を右クリックして「**統計要求 (Statistic Requests)**」を選択します。このウィンドウには、保留された、またはアクティブなすべてのユーザー要求統計収集の他に、システム要求統計収集として考慮された (候補として) もの、アクティブなもの、失敗したものすべてが表示されます。要求の状況を変更し、要求を順序付けして、即時に処理するか、要求を取り消すことができます。

関連資料

168 ページの『統計マネージャー API』
API を使用して、System i Navigatorの統計関数を実装することができます。

索引および列統計の比較

類似する機能を実行しても、索引と列統計では異なります。

統計マネージャーに情報を提供するために、統計を使用するか、索引を使用するかを決定しようとする場合、次の相違点について銘記してください。

索引と列統計との主要な相違点の 1 つは、索引は基礎となっているテーブルに対して変更が生じると更新される永続オブジェクトですが、列統計は永続オブジェクトではありません。 データが絶えず変更されていると、統計マネージャーは失効した列統計を基にする必要がある場合もあります。 しかし、テーブルに変更が加えられるごとに索引を保守すると、テーブルに生じた変更をグループ化して失効した列統計を最新表示するよりも、システム・リソースをより多く消費する可能性があります。

別の相違点は、新しい索引または列統計が存在する場合の最適化プログラムへの影響です。新しい索引が使用できるようになると、最適化プログラムはそうした索引を実装用と見なします。候補の場合には、最適化プログラムは照会を再最適化し、より最適な実装を見つけようとします。しかし、列統計にはこれは当てはまりません。新しい列統計または最新表示された列統計が使用できるようになると、統計マネージャーは即時に問い合わせます。統計を最新表示する前に指定された応答と、その後の応答がかなり異なる場合にのみ、再最適化が発生します。つまり、アクセス・プランを再最適化することなく、最新表示された統計を使用できるということです。

述部の選択を判別しようとする場合、次の順序で統計マネージャーは列統計および索引を応答のリソースと見なします。

1. AND 演算された述部または OR 演算された述部が複数列を参照する場合、複数列のキー処理された索引を使用しようとする
2. 述部内のすべての列を含む完全な索引が存在しない場合には、使用可能な索引の組み合わせを見つけようとする
3. 単一列質問の場合には、使用可能な列統計を使用する
4. 列統計から生成された質問に 2% 未満の選択が表示される場合、この質問を検査するために索引が使用される

列統計にアクセスすると、索引からそうした応答を取得しようとするよりも早く質問に応答できます。

列統計は SQE によってのみ使用できます。 CQE の場合、統計すべては索引から検索されます。

最後に、列統計は Query 最適化に対してのみ使用できます。索引は最適化と実際の実施の両方に使用できますが、列統計は照会の実際の実施には使用できません。

バックグラウンド統計収集のモニター

システム値 QDBFSTCCOL は、バックグラウンドでの作成を許可する統計について制御します。

以下のリストには、可能な値が提供されています。

***ALL**
すべての統計が、バックグラウンドで収集できます。これは、デフォルト設定です。

***NONE**
バックグラウンドで統計を作成することは全くできません。しかし、これにより即時にユーザー要求統計が収集できなくなることはありません。

***USER**
ユーザー要求統計だけが、バックグラウンドで収集できます。

***SYSTEM**
システム要求統計だけが、バックグラウンドで収集できます。

システム値を *ALL または *SYSTEM 以外の値に切り替えると、統計マネージャーは統計要求をプラン・キャッシュに引き続き入れます。システム値を例えば *ALL に再び切り替えると、バックグラウンド処理によってプラン・キャッシュ全体が分析され、任意の列統計要求が検索されます。さらに、このバックグラウンド・タスクは、プラン・キャッシュ内のプランで使用されている列統計を識別し、こうした列統計が失効していないかどうかも判別します。その後、新しい列統計に対する要求、および失効した列統計のリフレッシュ要求が実行されます。

システムによって開始されるか、ユーザーによってバックグラウンドに実行依頼されるすべてのバックグラウンド統計収集は、システム・ジョブ QDBFSTCCOL によって実行されます (ユーザーによって開始された即時要求は、ユーザー・ジョブで実行されます)。このジョブは、複数のスレッドを使用して統計を作成します。スレッドの数は、システムにあるプロセッサーの数によって決まります。それから、各スレッドは要求待ち行列に関連付けられます。

誰が要求を実行依頼したか、および実行のために見積もられる収集の長さに応じて、要求待ち行列には 4 つのタイプがあります。各スレッドに割り当てられるデフォルト優先順位によって、スレッドが属する待ち行列を決定できます。

- 優先順位 90 — 短ユーザー要求
- 優先順位 93 — 長ユーザー要求
- 優先順位 96 — 短システム要求
- 優先順位 99 — 長システム要求

バックグラウンド統計収集は、可能な限り多くの並列処理を使用しようとします。この並列処理は、システムにインストールされている SMP フィーチャーとは無関係です。しかし、SMP がシステムにインストールされていて、列統計を要求するジョブが並列処理できるように設定されている場合には、即時統計収集に対してのみ並列処理が許可されます。

関連情報
バックグラウンドでのデータベース統計収集許可 (QDBFSTCCOL) システム値

列統計の複製の際の CRTDUPOBJ および CPYF の比較

列統計は オブジェクト複製 (CRTDUPOBJ) または ファイル・コピー (CPYF) コマンドを使って複製できます。

ファイル・コピー (CPYF) コマンドを使用すると、統計は新規テーブルにはコピーされません。このコマンドの使用後、すぐに統計を必要とする場合には、System i ナビゲーターまたは統計 API を使用して、統計を手動で生成する必要があります。統計がすぐに必要ではない場合には、照会が列に最初に作用してからシステムが自動的に列統計の作成を実行することもできます。

オブジェクト複製 (CRTDUPOBJ) コマンドで DATA(*YES) を指定すると、統計がコピーされます。ファイル・コピー (CPYF) コマンドの使用後に、このコマンドを代わりに使用して統計を自動的に作成できます。

関連情報
複製オブジェクト作成 (CRTDUPOBJ) コマンド

ファイル・コピー (CPYF) コマンド

存在する列統計の判別

幾通りかの方法で、存在する列統計を判別できます。

最初の方法は、System i ナビゲーターを使用して統計を表示します。テーブルまたは別名を右クリックして、「**統計データ**」を選択します。別の方法は、ユーザー定義テーブル関数を作成して、 SQL ステートメントまたはストアード・プロシージャーからその関数を呼び出します。

統計の手動での収集および最新表示

System i ナビゲーターまたは統計 API を使用して、統計を手動で収集して最新表示できます。

System i ナビゲーターを使用して統計を収集するには、テーブルまたは別名を右クリックして「統計データ」を選択します。「統計データ」ダイアログで、「新規」をクリックします。その後、統計を収集する列を選択します。列を選択すると、統計を即時に収集したり、バックグラウンドで収集したりできます。

System i ナビゲーターを使用して統計を最新表示するには、テーブルまたは別名を右クリックして「**統計データ**」を選択します。「**更新**」をクリックします。最新表示する統計を選択します。統計を即時に収集したり、バックグラウンドで収集したりできます。

列統計の手動管理 (作成、除去、最新表示など) に関する、有益で推奨できる可能性のあるシナリオが幾つかあります。

高可用性 (HA) ソリューション

ジャーナル項目を使用してデータを 2 次システムに複製する高可用性ソリューションの設計を考慮する場合、列統計情報がジャーナルされないことを理解するのは重要です。つまりバックアップ・システムでは、そのシステムを使用して最初に開始する際に列統計は全く使用できないということです。 それが原因となって起こる可能性のある「ウォームアップ」影響を防ぐには、手動で実動システムで生成された列統計を伝搬し、それらをバックアップ・システム上に再作成することもできます。

ISV (Independent Solution Provider) 準備

ISV は、列統計を作成するために自動的に統計収集されるのを待つのではなく、アプリケーションで頻繁に使用される列統計が既に含まれるソリューションを配送することもできます。これを行うには、開発システム上のアプリケーションをしばらく実行し、自動的に作成された列統計を調べます。その後、初期データ・ロードが実行されてから、カスタマー・システムで実行されるべきアプリケーションの一部として発送されるスクリプト・ファイルを生成できます。

ビジネス・インテリジェンス環境

大規模なビジネス・インテリジェンス環境では、大規模なデータ・ロードおよび更新操作が夜間に行われるのはとても一般的です。統計マネージャーが列統計に作用して、その後に最新表示された場合にのみ、列統計には失効のマークが付けられるので、データのロード後に手動で最新表示することを考慮することができます。

システム値を QDBFSTCCOL から *NONE に切り替え、その後に *ALL に戻すと、簡単にそうできます。これにより、すべての失効した列統計は最新表示され、以前にシステムが要求したものの、まだ使用できる状態でない列統計すべての収集が開始されます。この処理はプラン・キャッシュに保管されているアクセス・プランに基づいているので、IPL がプラン・キャッシュをクリアしてしまうことのないよう、QDBFSTCCOL の切り替えの前にシステム初期プログラム・ロード (IPL) を実行しないでください。

このプロシージャーは、テーブルを削除 (ドロップ) しないで、データのロード処理の際に再作成する場合にのみ作用することに注意してください。テーブルを削除すると、このテーブルを参照するプラン・キャッシュ内のアクセス・プランは削除されます。そのテーブル上の列統計に関する情報も失われます。この環境における処理は、ご使用のテーブルにデータを追加するか、テーブルを削除するのではなくテーブルをクリアするかのどちらかになります。

大規模データの更新

基数をかなり変更する、新規の値の範囲を追加する、またはデータ値の分布を変更する、列統計使用可能なテーブル内の行を更新すると、照会を新規データに対して初めて実行する際にその照会のパフォーマンスに影響を与える可能性があります。 これは、そうした照会を初めて実行する際に、最適化プログラムが失効した列統計を使用してアクセス・プランを決定するために生じる場合があります。その場合、列統計を最新表示するという要求が開始されます。

データにこうした種類の更新を実行していることが分かっているなら、システム値 QDBFSTCCOL から *NONE に切り替えて、その後 *ALL または *SYSTEM に戻すこともできます。このようにすると、プラン・キャッシュが分析されます。この分析には、アクセス・プランの生成に使用された列統計の検索、および失効の分析、失効した統計の更新の要求が含まれています。

こうしてテーブルに対して大規模な更新またはデータのロード、および照会の実行を同時に実行する場合、列統計の自動収集はデータの 15% が変更されるたびに最新表示を試行します。このようにすると、データの更新またはロードの処理をまだ行っているので、処理が冗長的になる可能性があります。その場合には、質問中のテーブルに対する自動統計収集をブロックして、データ更新またはロードが終了した後に再び非ブロックにすることもできます。別の方法は、データを更新またはロードして、その更新またはロードが完了してから再び元に戻す前に、システム全体で自動統計収集をオフにします。

バックアップおよび回復

バックアップおよび回復方針について考慮する際、列統計の作成はジャーナルされないことに注意してください。保管操作が行われる際に存在する列統計は、テーブルの一部として保管され、テーブルに復元されます。保管後に作成された列統計は失われ、ジャーナル項目の適用などの技法を使用して再作成することはできません。 次の保管操作との間隔が長めで、現行状況への環境の復元のためにジャーナルに大いに依存している場合には、最後に行った保管操作の後に生成された列統計を記録することを考慮してください。

関連情報

バックグラウンドでのデータベース統計収集許可 (QDBFSTCCOL) システム値

統計マネージャー API

API を使用して、System i Navigatorの統計関数を実装することができます。

- 統計収集要求の取り消し (QDBSTCRS、QdbstCancelRequestedStatistics) は、要求は出されたもののまだ完了していない、あるいは正常に完了しなかった統計収集を即座に取り消します。
- 統計収集の削除 (QDBSTDS、QdbstDeleteStatistics) は、完了した既存の統計収集を即時に削除します。
- 統計収集要求のリスト表示 (QDBSTLRS、QdbstListRequestedStatistics) は、バックグラウンド統計収集が要求されたもののまだ完了していない、すべての列、および列とファイル・メンバーの組み合わせをリストします。
- 統計収集の詳細表示 (QDBSTLDS、QdbstListDetailStatistics) は、単一の統計収集の付加的な統計データをリストします。
- 統計収集のリスト表示 (QDBSTLS、QdbstListStatistics) は、有効な統計を持つ、所定のファイル・メンバーのすべての列および列の組み合わせをリストします。

- 統計収集の要求 (QDBSTRS、QdbstRequestStatistics) は、特定のファイル・メンバーの所定の列セットの 1 つ以上の統計収集を要求することを可能にします。
- 統計収集の更新 (QDBSTUS、QdbstUpdateStatistics) は、属性を更新し、既存の単一統計収集のデータを最新表示することを可能にします。

関連資料

164 ページの『統計要求の表示』
System i ナビゲーターを使用することにより、または統計 API を使用することにより、現行の統計要求を表示できます。

マテリアライズ照会表列の表示

System i ナビゲーターを使用して、別のテーブルに関連付けられたマテリアライズ照会表を表示できます。

マテリアライズ照会表を表示するには、次のステップを実行します。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開して、処理するデータベースを展開します。
3. 「**スキーマ**」を展開して、処理するスキーマを展開します。
4. テーブルを右クリックしてから、「**マテリアライズ照会表の表示 (Show Materialized Query Tables)**」を選択します。

表 36. 「マテリアライズ照会表の表示 (Show materialized query table)」ウィンドウで使用される列

列名	説明
SQL 名	マテリアライズ照会表の SQL 名
スキーマ	マテリアライズ照会表を含むスキーマまたはライブラリー
区画	索引のパーティションの詳細。次の値が指定可能です。 • <blank> (全パーティション用 (For all partitions) を意味します) • 各パーティション用 (For Each Partition) • パーティションの具体的な名前
所有者	マテリアライズ照会表の所有者のユーザー ID。
短縮名	マテリアライズ照会表のシステム・テーブルの名前
有効	マテリアライズ照会表が有効かどうか。次の値が指定可能です。 • はい • いいえ マテリアライズ照会表が有効でない場合、QUERY 最適化に使用できません。ただし、直接、照会することができます。
作成日付	マテリアライズ表が作成された時点のタイム・スタンプ。
最終の最新表示日付 (Last Refresh Date)	最後にマテリアライズ照会表が最新表示された時点のタイム・スタンプ。
照会の最後の使用	照会でユーザー指定のテーブルを置換するために最適化プログラムによってマテリアライズ照会表が最後に使用された時点のタイム・スタンプ。
照会統計の最後の使用 (Last Query Statistics Use)	アクセス方式を決定するために統計マネージャーによってマテリアライズ照会表が最後に使用された時点のタイム・スタンプ。

表 36. 「マテリアライズ照会表の表示 (Show materialized query table)」ウィンドウで使用される列 (続き)

列名	説明
照会使用回数	照会でユーザー指定のテーブルを置換するために最適化プログラムによってマテリアライズ照会表が使用されたインスタンスの回数。
照会統計使用回数 (Query Statistics Use Count)	アクセス方式を決定するために統計マネージャーによってマテリアライズ照会表が使用されたインスタンスの回数。
最終使用日付	マテリアライズ照会表が最後に使用された時点のタイム・スタンプ。
使用日数カウント	マテリアライズ照会表が使用された日数。
使用日数がリセットされた日	使用日数が最後に 0 に設定された年および日付。
現在の行数	現時点でこのマテリアライズ照会表に含まれる行数の合計。
現在のサイズ	マテリアライズ照会表の現在のサイズ。
最終変更	マテリアライズ照会表が最後に変更された時点のタイム・スタンプ。
保守	マテリアライズ照会表の保守。次の値が指定可能です。 • ユーザー • システム
初期データ	初期データが即時に挿入されるか、据え置かれるか。次の値が指定可能です。 • 据え置き • 即時
最新表示モード	マテリアライズ照会表の最新表示モード。マテリアライズ照会表は、テーブルが変更されたか、後に据え置かれるたびに最新表示することができます。
分離レベル	マテリアライズ照会表の分離レベル。
ソート順序	各国語サポート (NLS) の代替文字ソート・シーケンス。
言語 ID	オブジェクトの言語コード。
SQL ステートメント	テーブルを取り込むために使用される SQL ステートメント。
テキスト	マテリアライズ照会表のテキスト記述。

チェック・ペンディング制約列の管理

システムによってチェック・ペンディング状態に置かれた制約を表示および変更することができます。チェック・ペンディングとは、参照制約の場合は親と外部キーとの間に不一致が存在し、チェック制約の場合は、列の値とチェック制約定義の間に不一致が存在する状態を意味します。

チェック・ペンディング状態に置かれた制約を表示するには、以下のステップを実行します。

1. システム名および「**データベース**」を展開します。使用するデータベースを右クリックして、「**チェック・ペンディング制約を管理 (Manage check pending constraints)**」を選択します。
2. このインターフェースから、制約の定義、および制約規則に違反している行を表示できます。作業対象になる制約を選択し、「**ファイル**」メニューから「**チェック・ペンディング制約を編集 (Edit Check Pending Constraint)**」を選択します。
3. 違反している行を変更または削除することができます。

表 37. 「チェック・ペンディング制約 (Check pending constraints)」ウィンドウで使用される列

列名	説明
チェック・ペンディングの制約の名前 (Name of Constraint in Check Pending)	チェック・ペンディング状態にある制約の名前を表示します。
スキーマ	チェック・ペンディング状態にある制約を含むスキーマ。
タイプ	チェック・ペンディングの状態にある制約のタイプを表示します。次の値が指定可能です。 チェック制約 外部キー制約
テーブル名	チェック・ペンディング状態にある制約に関連付けられたテーブルの名前。
有効	制約が有効かどうかを表示します。入出力 (I/O) 操作が実行できるようになる前に、制約を無効にするか、関係をチェック・ペンディング状態の対象外にする必要があります。

索引方針の作成

DB2 for i5/OS にはテーブル・アクセスのために基本的な 2 つの方法が用意されています。それは、テーブル走査と索引による検索です。テーブル行の 20% 未満が選択されるとき、索引による検索は通常、テーブル走査より効果的です。

2 種類の永続的索引があります。それは、1988 年以来使用可能な 2 進基数ツリー索引と 1998 年に V4R2 で使用可能になったコード化ベクトル索引 (EVI) です。どちらのタイプの索引も、ある種類の照会のパフォーマンスの向上に役立ちます。

2 進基数索引

基数索引は、アクセス時間を最小化するだけでなく多数のキー値が効果的に格納できるようにする、マルチレベルでハイブリッドなツリー構造です。キー圧縮アルゴリズムは、このプロセスを支援します。ツリーの最下位レベルには、リーフ・ノード (キー値と関連した基本テーブルにある行のアドレスを含む) が含まれています。キー値は、少数の単純な 2 進検索テストを備えたリーフ・ノードへ素早くナビゲートするために使用されます。

2 進基数ツリー構造は、特定の行を最少量の処理で検索することができるので、少数の行を検索するのに非常に適しています。例えば、「1 人の顧客の未払いのオーダーを検索する」というような典型的な OLTP 要求について、カスタマー番号列に対して 2 進基数索引を使用すると、高速なパフォーマンスが得られます。カスタマー番号列に対して作成された索引は、データベースが必要とする行に的をしぼり、実行する入出力を最小限に抑えることができるので、このタイプの照会に対して完璧な索引と考えられます。

ただし、ある状態では、常時同じレベルが予測できるとは限りません。ユーザーは、次第に詳細なデータに随時アクセスしたいと思うようになります。たとえば、毎週報告書を実行して販売データを調べ、報告書で検出された特定の問題部分と関連した詳しい情報を「掘り下げる」場合があります。このシナリオでは、エンド・ユーザーに代わって前もってすべての照会を書き込むことはできません。どの照会が実行されるかを知らずに、完璧な索引を構築することは不可能です。

関連情報

SQL Create Index ステートメント

派生キー索引

SQL CREATE INDEX ステートメントを使用して、SQL 式を使用する派生キー索引を作成できます。

従来、索引は、ベースとなるテーブル上の索引のキーに列名のみを指定できました。このサポートでは、索引は、列名の代わりに、組み込み関数、ユーザー定義関数、または他の有効な式を使用できる式を持つことができます。さらに、SQL CREATE INDEX ステートメントを使用して、WHERE 条件を使用する疎索引を作成できます。

例

```
CREATE UNIQUE INDEX MYLIB/X1 ON MYLIB/T1 (CONCAT(COL1,COL2) AS MYCOL1
CREATE INDEX MYLIB/X2 on MYLIB/T2  (UPPER(COL2))
```

3 つの入力パラメーターを受け入れ、INT を戻す MYUDF が作成されました。

```
CREATE UNIQUE INDEX MYLIB/X3
     ON MYLIB/T3 (MYUDF3(COL1,COL2, 10) AS MYCOL3)
CREATE ENCODED VECTOR INDEX MYLIB/X4
     ON MYLIB/T4 (MYUDF4(COL1,COL2, COL3) AS MYCOL4)
```

例えば、テーブル T1 には、COL1、COL2 および COL3 が含まれています。以下の項目が有効です。

```
CREATE UNIQUE INDEX X1
     ON T1 (COL1, MYUDF(COL1,COL2, 10) AS COL2)
```

次の例は、COL2 が索引キーで 2 回使用されているため無効です。

```
CREATE UNIQUE INDEX X1
     ON T1 (COL1, COL2, MYUDF(COL1,COL2, 10) AS COL2)
```

派生索引に関する制約事項およびその他の情報については、Create Index ステートメントを参照してください。

関連資料

192 ページの『派生索引の使用』

SQL 索引は、キーが式として指定された場所に作成できます。これは、「派生キー」という名前で呼ばれることもあります。

関連情報

SQL Create Index ステートメント

索引作成時に PAGESIZE を指定する

物理ファイル作成 (CRTPF) コマンドまたは 論理ファイル作成 (CRTLF) コマンド、あるいは SQL CREATE INDEX ステートメントを使ってキー付きファイルまたは索引を作成する際、PAGESIZE パラメーターを使用して、アクセス・パスの作成時にシステムによって使用されるアクセス・パスの論理ページ・サイズを指定することができます。

この論理ページ・サイズは、ページ不在のための補助記憶域からジョブの記憶域プールに移動することができるアクセス・パスのバイト量です。

ページ・サイズがキー (複数可) の全長に基づいてシステムによって決定されるために、まれな事情を除いて、このパラメーターにはデフォルトの *KEYLEN を使用することを検討すべきです。非常に選択的な照会 (たとえば個別のキー検索など) によってアクセス・パスが使用される場合、通常、ページ・サイズが小さければ小さいほど効率的になります。また、照会によって選択されているキーが共にアクセス・パスでグ

ループ化され、多くのレコードが選択されている場合、またはアクセス・パスが走査されている場合には、通常、ページ・サイズが大きければ大きいほど効率的になります。

関連情報

論理ファイル作成 (CRTLF) コマンド

物理ファイル作成 (CRTPF) コマンド

SQL Create Index ステートメント

一般的な索引保守

索引の作成および使用時には、保守が原因で入出力の速度が低下する可能性があります。したがって、追加の索引を作成して使用する場合の保守コストを考慮する必要があります。 MAINT(*IMMED) を使用する基数索引では、行を挿入、更新、または削除する場合に保守が実行されます。

索引の保守を削減するには、以下のことを考慮してください。

- 複合 (複数の列) キー索引を作成し、索引を複数の異なる状況で使用できるようにすることにより、特定のテーブルに対する索引の数を最小限に抑える。
- バッチの挿入、更新、および削除時に索引を除去する
- 並行的な作成、つまり、 SMP を使用して、並行して一度に 1 つずつ索引を作成するか、または複数のバッチ・ジョブを使用して複数の索引を同時に作成する
- SMP を使用して並行して索引を保守する

索引作成のゴールは、保守のオーバーヘッドを制限するために索引の数の適切なバランスを保ちつつ、統計および実施の選択を提供することによって照会パフォーマンスを向上させることです。

コード化ベクトル索引

コード化ベクトル索引 (EVI) は、Query 最適化プログラムおよびデータベース・エンジンによって使用される索引オブジェクトであり、決定サポートおよび照会レポート環境での高速データ・アクセスを提供します。

EVI は、既存の索引オブジェクト (2 進基数ツリー構造 - 論理ファイルまたは SQL 索引) の補足的な代替機能であり、ビットマップ索引付けのバリエーションです。 EVI 索引は、サイズが小さく比較的単純なため、並列処理も可能なテーブルの走査がさらに高速になります。

EVI は、以下の 2 つのコンポーネントで保管されるデータ構造です。

- 記号テーブルには、テーブルで示されるそれぞれの特殊キー値についての統計および記述情報が含まれます。それぞれの特殊キーには、固有コードが割り当てられ、サイズは 1、2、または 4 バイトのいずれかになります。
- ベクトルは、テーブルにある行と同じ順序の位置にリストされるコードの配列です。ベクトルには、テーブルの実際の行へのポインターは含まれません。

EVI の利点

- 記憶域が比較的小さくて済む
- 基数に比べ、ビルド時間が軽減できる場合がある。特にキーに定義された列の固有値が、比較的小さい場合に軽減できる。
- Query 最適化プログラムにさらに正確な統計を提供する
- 照会の一部のグループ化タイプで、パフォーマンスが大幅に向上する
- 意思決定支援環境のパフォーマンス指標が向上する。

EVI の欠点

- 順序付けでは使用できない
- グループ化の使用が特殊化される
- 結合の使用が常にハッシュ・テーブル処理との連携で行われる
- その他いくつか保守上の特異性がある

関連情報

SQL Create Index ステートメント

EVI の機能

EVI は、コスト計算やインプリメンテーションのためにさまざまな方法で機能します。

コスト計算を行う場合、最適化プログラムは、照会についてのメタデータ情報を収集する記号テーブルを使用します。

インプリメンテーションを行う場合、最適化プログラムは以下のいずれかの方法で EVI を使用できます。

- **選択 (WHERE 文節)**

  最適化プログラムが、EVI を使用して照会を処理することを決定した場合には、データベース・エンジンはベクトルを使用して、テーブルで各行につき 1 ビットを含む (ビットは選択されたそれぞれの行でオンになる) 動的ビットマップ(または選択行 ID のリスト) を作成します。ビットマップ索引と同様に、これらの中間動的ビットマップ (またはリスト) は AND および OR を使用して、随時照会の条件を満たすことができます。

  たとえば、あるユーザーが、一定の期間について特定の地域の販売データを参照しようとする場合、データベースの領域列と四半期列に対して EVI を定義することができます。照会が実行されると、データベース・エンジンは 2 つの EVI を使用して動的ビットマップを作成し、それから、このビットマップを AND で結合し、両方の選択基準に適する行のみを含むビットマップを生成します。この AND の機能により、システムが読み取って検査しなければならない行の数は大幅に削減されます。動的ビットマップは、照会が実行中である時にのみ存在します。照会が完了すると、動的ビットマップは除去されます。

- **グループ化または DISTINCT**

  EVI 内の記号テーブルには、キー定義で指定された列の特殊値、および基本テーブルの各特殊値を持つレコードの番号のカウントが含まれます。実際に記号テーブルには、そのキーの列のグループ化の結果が含まれます。したがって、そのキーの列のグループ化または DISTINCT を伴う照会は、照会の結果を判別するために直接記号テーブルを使用する手法の潜在的な候補となります。記号テーブルにはキー値とそれに関連したカウントのみが含まれます。したがって、列関数 COUNT を伴う照会はこの手法に適していますが、別の列の列関数 MIN または MAX を伴う照会は適しません (min および max の値は記号テーブルに保管されないからです)。

EVI を作成するタイミング

EVI の作成を考慮すべきインスタンスがいくつかあります。

以下のいずれかに当てはまる場合、コード化ベクトル索引について考慮する必要があります。

- 統計をライブで収集したい
- 現在、照会に対し、完全テーブル走査が選択されている

- 照会の選択度が 20%-70% で、動的ビットマップと共にスキップ順次アクセスを使用することで、走査の高速化が期待できる
- 星形スキーマ結合照会に、星形スキーマ結合の使用が想定されている
- 列に対し、グループ化、または特殊照会が指定されている場合、その列にはわずかな特殊値があり、(列関数がすべてに対して指定されている場合) COUNT 列関数のみが使用される

コード化ベクトル索引は、以下のものを使用して作成されます。

- 予期される特殊値の数が少ない、単一キー列
- 揮発性の低いキー列 (頻繁に変更されることがない)
- WITH n DISTINCT VALUES 文節を使用して予期される特殊値の最大数
- 星形スキーマ・モデル用の外部キー列に対する単一キー

EVI の保守

EVI の保守には固有の問題があります。以下の表では、EVI の保守の進行方法、および EVI の効果を最大にする条件、および EVI の効果が最小となる条件を、 EVI 保守の特性に基づいて説明します。

表 38. EVI の保守についての考慮事項

	条件	特性
最も効果が高い  最も効果が低い	既存の特殊キー値を挿入する場合	• 最小オーバーヘッド • 記号テーブルのキー値が検索され、統計が更新される • 既存のバイト・コードを使用して、新しい行にベクトル・エレメントが追加される
	新規の 特殊キー値を、バイト・コードの範囲内で整列させて挿入する場合	• 最小オーバーヘッド • 記号テーブルのキー値が追加され、バイト・コードが割り当てられ、統計が割り当てられる • 新しいバイト・コードを使用して、新しい行にベクトル・エレメントが追加される
	新規の特殊キー値を、バイト・コードの範囲内で順序に関係なく挿入する場合	• オーバーフロー域しきい値内に入っている場合には、最小オーバーヘッド • 記号テーブルのキー値がオーバーフロー域に追加され、バイト・コードが割り当てられ、統計が割り当てられる • 新しいバイト・コードを使用して、新しい行にベクトル・エレメントが追加される • オーバーフロー域しきい値に達している場合には、かなりのオーバーヘッド • アクセス・パスは妥当性検査済み - 利用不能 • EVI は更新され、オーバーフロー域キーが取り込まれ、新しいバイト・コードが割り当てられる (記号テーブルおよびベクトル・エレメントは更新される)
	新規の特殊キー値を、バイト・コードの範囲を超えて挿入する場合	• かなりのオーバーヘッド • アクセス・パスは無効 - 利用不能 • EVI は更新され、次のバイト・コード・サイズが使用され、新しいバイト・コードが割り当てられる (記号テーブルおよびベクトル・エレメントは更新される)

EVI の使用に関する推奨事項

コード化ベクトル索引は、決定サポートおよび照会報告環境で高速データ・アクセスを提供する強力なツールですが、EVI を効果的に使用するには、以下の指針に従って EVI をインプリメントする必要があります。

以下に基づいて EVI を作成します:

- 読み取り専用テーブル、または最小の INSERT、UPDATE、DELETE アクティビティーがあるテーブル。
- WHERE 文節で使用されるキー列 - SQL 要求のローカル選択述部。
- 比較的小規模な特殊値のセットがある、単一キー列。
- 比較的小規模な特殊値のセットがある、複数のキー列。
- 静的または比較的静的な特殊値のセットがあるキー列。
- 多くの重複がある非固有キー列。

予期される最大バイト・コード・サイズで EVI を以下のように作成します:

- CREATE ENCODED VECTOR INDEX ステートメントで ″WITH n DISTINCT VALUES″ 文節を使用します。
- 不確実な場合には、65,535 より大きい数を使用して 4 バイト・コードを作成し、こうしてバイト・コード・サイズを切り替えることによって EVI の保守オーバーヘッドを避けます。

データをロードする場合:

- EVI を除去し、データをロードし、EVI を作成します。
- EVI バイト・コード・サイズは、テーブルで見つかる実際の特殊キー値の数に基づいて自動的に割り当てられます。
- 記号テーブルには、すべてのキー値が順序どおりに入っており、オーバーフロー域にはキーは入っていません。

SMP および並列索引作成および保守の考慮

対称マルチプロセッシング (SMP) は、並列で索引を作成および保守するために役立つツールです。 i5/OS のオプションの SMP 機能を使用する結果、索引の作成時間が短縮され、並列で索引を保守する際の入出力速度も高速化します。 *OPTIMIZE あるいは *MAX のいずれかの SMP 次数値を使用すると、索引の作成および保守に、付加的な複数タスクおよび付加的なシステム・リソースが使用されます。次数値 *MAX では、索引作成時のリニア・スケーラビリティーを予期します。例えば、4 つのプロセッサー・システムで索引を作成すると、1 つのプロセッサー・システムの場合より 4 倍も速く処理できます。

オーバーフロー域での値の検査

ファイル記述の表示 (DSPFD) コマンド (または System i ナビゲーター - データベース) を使用して、オーバーフロー域にいくつの値があるかを検査することもできます。 DSPFD コマンドが発行されたら、オーバーフロー域のパラメーターで、オーバーフロー域にある特殊キーの値の初期の数および実際の数についての詳細を検査してください。

CHGLF を使用して索引のアクセス・パスを再作成する

再構築強制アクセス・パスの属性を *YES* に設定して (FRCRBDAP(*YES))、論理ファイルの変更 (CHGLF) コマンドを使用します。このコマンドは、索引の除去および再作成と同じことを実行しますが、ユーザーがこの索引の構成を把握している必要はありません。このコマンドは、オリジナルの索引定義を使用できない

アプリケーション、あるいはアクセス・パスの最新表示に特に効果的です。

関連情報

SQL Create Index ステートメント

論理ファイル変更 (CHGLF) コマンド

ファイル記述表示 (DSPFD) コマンド

2 進基数索引とコード化ベクトル索引の比較

DB2 for i5/OS は索引を強力なツールに変えます。

以下の表は、2 進基数索引とコード化ベクトル索引の間のいくつかの相違を要約しています。

表 39. 基数索引と evi 索引の比較

	2 進基数索引	**コード化ベクトル索引**
基本データ構造	幅広い、フラット・ツリー	記号テーブルおよびベクトル
作成のためのインターフェース	コマンド、SQL、System i ナビゲーター	SQL、System i ナビゲーター
並列して作成できる	はい	はい
並列して保守できる	はい	はい
統計に使用される	はい	はい
選択に使用される	はい	はい、動的ビットマップまたは RRN リスト経由で
結合に使用される	はい	はい (ハッシュ・テーブルとの結合)
グループ化に使用される	はい	はい
順序付けに使用される	はい	いいえ
固有の参照保全制約を実行するのに使用される	はい	いいえ

索引および最適化プログラム

最適化プログラムは最適化に基づいたコストを使用するので、データベースの行または列について最適化プログラムがさらに情報を与えられれば与えられるほど、最適化プログラムは照会に対して最良の可能なアクセス・プラン (コスト最低/速度最高) を作成できます。索引からの情報を使用して、最適化プログラムは要求 (ローカル選択、結合、グループ化、および順序付け) を処理する方法についてより良い選択をすることができます。

CQE 最適化プログラムは、タイムアウトでない場合またはタイムアウトになるまで、テーブルに対して構築されたほとんどの (すべてではないにしても) 索引を調べようとします。しかし、SQE 最適化プログラムは統計マネージャーによって戻された索引だけを考慮します。統計マネージャーが判別する索引のみを含むこれらの索引は、「where」文節述部を基にしたローカル選択を実行するのに有用です。したがって、SQE 最適化プログラムはタイムアウトになりません。

最適化プログラムの主なゴールは、関心がない、または要求を満たすのに必要でない行をすばやく、効果的に除去するインプリメンテーションを選択することです。通常、Query 最適化は関心のある行を検索しようとするものと考えられます。適切な索引付け方針を確立すると、このタスクでの最適化プログラムおよびデータベース・エンジンに役立ちます。

索引が使用されない場合のインスタンス

DB2 for i5/OS は、特定のインスタンスでは索引を使用しません。

- 更新を予定している列の場合。たとえば、SQL を使用するときにユーザーのプログラムに次のようなコードが含まれているとします。

```
EXEC SQL
 DECLARE DEPTEMP CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
      WHERE (WORKDEPT = 'D11' OR
               WORKDEPT = 'D21') AND
              EMPNO = '000190'
      FOR UPDATE OF EMPNO, WORKDEPT
END-EXEC.
```

  OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((CORPDATA/EMPLOYEE)) OPTION(*ALL)
   QRYSLT('(WORKDEPT *EQ ''D11'' *OR WORKDEPT *EQ ''D21'')
   *AND EMPNO *EQ ''000190''')
```

  従業員の部門を更新するつもりがない場合でも、システムは WORKDEPT のキーで索引を使用することができません。

  索引内で使用されたすべての更新可能な列が照会内でも等号演算子を持つ分離可能な選択述部として使用される場合、システムは索引を使用できます。上記の例で、システムはキー EMPNO を指定して索引を使用することになります。

  FOR UPDATE OF の列リストでユーザーが更新を望んでいる列名 (*WORKDEPT*) のみを指定すれば、システムはさらに効率よく稼働することができます。したがって、FOR UPDATE OF の列リストには、更新を望んでいない列名を指定しないでください。

  動的 SQL のために更新可能なカーソルがある場合、または FOR UPDATE 文節が指定されず、プログラムが UPDATE ステートメントを含む場合、全列を更新することができます。

- 同じ行内の他の列と比較される列の場合。たとえば、SQL を使用する場合は、プログラムに以下を組み込むことができます。

```
EXEC SQL
 DECLARE DEPTDATA CURSOR FOR
    SELECT WORKDEPT, DEPTNAME
      FROM CORPDATA.EMPLOYEE
      WHERE WORKDEPT = ADMRDEPT
END-EXEC.
```

  OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE (EMPLOYEE) FORMAT(FORMAT1)
   QRYSLT('WORKDEPT *EQ ADMRDEPT')
```

  WORKDEPT に索引があり、 *ADMRDEPT* にも別の索引がありますが、DB2 for i5/OS はどちらも使用しません。この場合、テーブルのすべての行を走査する必要があるので、索引の持つ利点は活用できません。

テーブルの索引の表示

System i ナビゲーターを使用して、テーブル上に作成された索引を表示することができます。

テーブルの索引を表示するには、次のステップを実行します。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開して、処理するデータベースを展開します。
3. 「**スキーマ**」を展開して、処理するスキーマを展開します。
4. テーブルを右クリックしてから、「**Show Indexes (索引の表示)**」を選択します。

「Show index (索引の表示)」ウィンドウには、以下の列が含まれます。

表 40. 「Show index (索引の表示)」ウィンドウで使用される列

列名	説明
SQL 名	索引の SQL 名
タイプ	表示される索引のタイプ。次の値が指定可能です。 • キー順物理ファイル • キー順論理ファイル • 基本キー制約 • ユニーク・キー制約 • 外部キー制約 • 索引
スキーマ	索引またはアクセス・パスを含むスキーマまたはライブラリー
所有者	この索引またはアクセス・パスの所有者のユーザー ID
短縮名	索引またはアクセス・パスのシステム・テーブル名。
テキスト	索引またはアクセス・パスのテキスト記述
索引パーティション	索引のパーティションの詳細。次の値が指定可能です。 • <blank> 全パーティション用 (For all partitions) • 各パーティション用 (For Each Partition) • パーティションの具体的な名前
有効	アクセス・パスまたは索引が有効かどうか。指定可能な値は「はい」または「いいえ」です。
作成日付	索引作成時のタイム・スタンプ。
最終構築	アクセス・パスまたは索引が最後に再ビルドされた時刻。
照会の最後の使用	アクセス・パスが最適化プログラムによって最後に使用された際のタイム・スタンプ。
照会統計の最後の使用 (Last Query Statistics Use)	アクセス・パスが最後に統計で使用された際のタイム・スタンプ。
照会使用回数	アクセス・パスが照会で使用された回数
照会統計使用回数 (Query Statistics Use Count)	アクセス・パスが統計で使用された回数
最終使用日付	アクセス・パスまたは索引が最後に使用された際のタイム・スタンプ。
使用日数カウント	索引が使用された日数。
使用日数がリセットされた日	使用日数が最後に 0 に設定された年および日付。
キー列の数 (Number of Key Columns)	アクセス・パスまたは索引用に定義されたキー列の数。
キー列 (Key Columns)	アクセス・パスまたは索引用に定義されたキー列。
現行キー値 (Current Key Values)	現行キー値の数値。
現在のサイズ	アクセス・パスまたは索引のサイズ。

表 40. 「Show index (索引の表示)」ウィンドウで使用される列 (続き)

列名	説明
現行の割り振りページ (Current Allocated Pages)	アクセス・パスまたは索引用に割り振られた現行のページ数。
論理ページ・サイズ (Logical Page Size)	アクセス・パスまたは索引の論理ページ・サイズ用に使用されるバイト数。索引の論理ページ・サイズが大きいほど、照会処理中に走査する際、一般により効率的です。索引の論理ページ・サイズが小さいほど、一般に単純な索引プローブおよび個別キー探索をより効率的に実行できます。アクセス・パスまたは索引がエンコード・ベクトルである場合、値 0 が戻されます。
重複キーの順序 (Duplicate Key Order)	アクセス・パスまたは索引が重複キー値を処理する方法。次の値が指定可能です。 • 固有 - すべての値が固有です。 • 非 NULL の場合に固有 (Unique where not null) - NULL が指定されていない限り、すべての値が固有です。
最大キー長	アクセス・パスまたは索引用の最大キー長。
固有部分キー値 (Unique Partial Key Values)	キー・フィールド 1 から 4 までの固有部分キーの数値。アクセス・パスがエンコード・ベクトルである場合、この数値は全キーの異なる数値を表します。
オーバーフロー値	このエンコード・ベクトル索引のオーバーフロー値の数値。
キー・コード・サイズ	エンコード・ベクトル索引の異なるキー値ごとに割り当てられたコードの長さ。
疎 (Sparse)	疎と見なされる索引です。疎索引には、照会に適合する行のキーのみが入ります。次の値が指定可能です。 • はい • いいえ
派生キー (Derived Key)	派生されたと見なされる索引です。派生キーは、基本列の操作の結果であるキーです。次の値が指定可能です。 • はい • いいえ
分割	指定された列を使用してテーブル用に定義されたデータ・パーティションごとに索引パーティションを作成するかどうか。次の値が指定可能です。 • はい • いいえ
最大サイズ	アクセス・パスまたは索引の最大サイズ。
ソート順序	各国語サポート (NLS) の代替文字ソート・シーケンス。
言語 ID	オブジェクトの言語コード。
再ビルドの推定時間	アクセス・パスまたは索引の再ビルドに必要な推定時間。
保留	アクセス・パスまたは索引の再ビルドが保留されます。次の値が指定可能です。 • はい • いいえ

表 40. 「Show index (索引の表示)」ウィンドウで使用される列 (続き)

列名	説明
保守	キー・フィールドを持つオブジェクトまたは結合論理ファイルに対して使用されるアクセス・パス保守のタイプ。次の値が指定可能です。 • 待機しない • 遅延 • 再作成
遅延保守キー	アクセス・パスまたは索引の遅延保守キーの数。
リカバリー	アクセス・パスに対する損傷が認識された際にアクセス・パスを即時に再作成するかどうか。次の値が指定可能です。 • IPL 後 • IPL 時 • 次のオープン
索引の論理読み取り数	最後の IPL 以後のアクセス・パスまたは索引の論理読み取り操作の数。
索引の物理読み取り数	最後の IPL 以後のアクセス・パスまたは索引の物理読み取り操作の数。

不要な索引の判別

照会の最適化で使用されている索引の判別は簡単に行えます。

V5R3 より前のリリースでは、不要な索引を判別することが困難でした。最終使用日付を使用するという方法も考えられますが、それが更新されるのはネイティブ・データベース・アプリケーション (たとえば RPG アプリケーションなど) を使用して論理ファイルが開かれた時点だけなので、信頼できるものではありません。さらに、物理ファイルについてのすべての索引を検出することは困難でした。索引は、キー付き物理ファイル、キー付き論理ファイル、結合論理ファイル、SQL 索引、主キーまたはユニーク制約、または参照制約の一部として作成されます。しかし、System i ナビゲーターおよび i5/OS の機能の結果として、すべての索引を検出したり、索引の使用に関する統計データを調べたりすることが容易になりました。この機能により、索引の使用に関する統計データおよび照会での索引使用状況の情報が生成され、パフォーマンス調整のために利用できるようになりました。

System i ナビゲーターによりこの機能にアクセスするには、「**データベース**」 → 「**スキーマ**」 → 「**テーブル**」とナビゲートします。テーブルを右クリックしてから、「**Show Indexes (索引の表示)**」を選択します。

「**テーブル**」または「**索引**」を右クリックするか、「Show Indexes (索引の表示)」を選択して、スキーマのすべての索引を表示することができます。

注: 統計は、メンバー記述の検索 (QUSRMBRD) API を使用して表示することもできます。

「**Show Indexes (索引の表示)**」ウィンドウで使用できる特定のフィールドは、不要な索引を判別する際に役立つ場合があります。それらのフィールドは、次のとおりです。

照会の最後の使用
照会のデータを検索するために索引が最後に使用された時点のタイム・スタンプ。

照会統計の最後の使用
統計情報を提供するために索引が最後に使用された時点のタイム・スタンプ。

照会使用回数
照会で索引が使用されたインスタンスの数。

Query Statistics Use (照会統計使用)
統計情報のために索引が使用されたインスタンスの数。

最終使用日付
この索引が最後に使用された世紀および日付。

使用日数カウント
索引が使用された日数。索引に最終使用日付がない場合、カウントは 0 になります。

使用日数がリセットされた日
使用日数が最後にリセットされた日付。日数のリセットは、オブジェクト記述変更 (CHGOBJD) コマンドの使用によって行えます。

これらのフィールドは、状況に応じて、あるいはシステムで現在実行中のアクションに応じてカウントを開始したり停止したりします。カウンターに影響するものとしては、以下のものがあります。

- SQE および CQE 照会エンジンは、どちらのカウンターも増分します。そのため、どの照会インターフェースを使用する場合も統計フィールドが更新されます。
- 保管およびリストアの手順を実行して既存の索引に対して索引を上書きでリストアする場合、統計カウンターはリセットされません。システム上に存在しない索引をリストアする場合、統計情報はリセットされます。

関連資料

127 ページの『要約モニターの開始』
System i ナビゲーター・インターフェースから、要約モニターを開始できます。

関連情報

Retrieve Member Description (QUSRMBRD) API

オブジェクト記述変更 (CHGOBJD) コマンド

使用カウントのリセット

テーブルの使用カウントをリセットすることによって、索引付けストラテジーへの変更が、そのテーブル上に作成された索引および制約にどのように影響を及ぼすかを決定することができます。例えば、新しいストラテジーによって索引が決して使用されなくなった場合、その索引を削除できる場合があります。テーブル上の使用カウントをリセットすれば、そのオブジェクト上に作成されたすべての索引および制約が影響を受けます。

注: キー順物理ファイルの使用カウントまたは「索引の表示 (Show Indexes)」ウィンドウの制約をリセットすると、そのファイルまたはテーブルのすべての制約およびキー順アクセスに影響を及ぼすカウントがリセットされます。

「テーブル」フォルダー、「索引」フォルダーの特定の索引、および「索引の表示 (Show Indexes)」ダイアログのすべての項目を右クリックして、「**使用カウントをリセット (Reset Usage Counts)**」を選択することによって、索引の使用カウントをリセットすることができます。

索引再ビルドの管理

索引の再ビルドの管理は、System i ナビゲーターを使って行えます。再作成するアクセス・パスのリストを表示して、アクセス・パスの再作成を保留するか、再作成の優先順位を変更することができます。

再ビルドのためのアクセス・パスを表示するには、以下のステップに従ってください。

1. 「System i ナビゲーター」ウィンドウで、使用するシステムを展開します。
2. 「**データベース**」を展開します。
3. 処理したいデータベースを右クリックして、「**索引の再ビルドを管理**」を選択します。

再ビルドのためのアクセス・パスのダイアログには、以下の列が含まれます。

表 41. 「索引の再ビルドを管理」ウィンドウで使用される列

列名	**説明**
再ビルドされる索引の名前 (Name of Index to Rebuild)	再ビルドされるアクセス・パスのロング・ネーム
スキーマ	索引が配置されるスキーマの名前
タイプ	表示される索引のタイプ。次の値が指定可能です。 キー順物理ファイル キー順論理ファイル 基本キー ユニーク・キー 外部キー 索引
状況	再ビルドの状況を表示します。次の値が指定可能です。 1-99 – *再ビルド優先順位* 実行中 – *再ビルド中* 保留 – *再ビルドの保留*
再ビルド優先順位	このアクセス・パスの再ビルドが実行される優先順位を表示します。シーケンス番号とも呼ばれます。次の値が指定可能です。 1-99: 再ビルドの順序 保留 オープン
再ビルドの理由	このアクセス・パスを再ビルドする必要がある理由を表示します。次の値が指定可能です。 索引の作成またはビルド (Create or build index) IPL 実行時エラー ファイルまたは索引共用の変更 (Change file or index sharing) その他 不要 データの終わりを変更 復元 テーブルの変更 (Alter table) テーブルの変更 (Change table) ファイルの変更 再編成 制約の使用可能化 (Enable a constraint) テーブル回復の変更 (Alter table recovery) ファイル回復の変更 (Change file recovery) 索引の共有 (Index shared) 実行時エラー 制約の検証 (Verify constraint) メンバーの変換 (Convert member) 回復の復元 (Restore recovery)

表 41. 「索引の再ビルドを管理」ウィンドウで使用される列 (続き)

列名	説明
再ビルドの理由サブタイプ (Rebuild Reason Subtype)	このアクセス・パスを再ビルドする必要があるサブタイプの理由を表示します。次の値が指定可能です。 予期しないエラー 障害時に使用中の索引 (Index in use during failure) 更新、削除、または挿入時の予期しないエラー (Unexpected error during update, delete, or insert) 遅延保守オーバーフローまたはキャッチアップ・エラー その他 イベントなし データの終わりを変更 遅延保守の不一致 (Delayed maintenance mismatch) 論理ページ・サイズの不一致 (Logical page size mismatch) 部分索引の復元 (Partial index restore) 索引の変換 (Index conversion) 索引が保管されておらず、復元されていない (Index not saved and restored) 区画化の不一致 (Partitioning mismatch) 区画化の変更 (Partitioning change) 索引またはキー属性の変更 (Index or key attributes change) オリジナル索引が無効 (Original index invalid) 索引属性の変更 (Index attributes change) 索引の再ビルドの強制 (Force rebuild of index) 索引が復元されていない (Index not restored) 非同期再ビルドが要求された (Asynchronous rebuilds requested) ジョブの異常終了 (Job ended abnormally) テーブルの変更 (Alter table) 制約の変更 (Change constraint) 無効な索引または属性の変更 (Index invalid or attributes change) 無効な固有索引の検出 (Invalid unique index found) 無効な制約索引の検出 (Invalid constraint index found) 索引の変換が必要 (Index conversion required) サブタイプがない場合は、このフィールドは 0 を表示します。

表 41. 「索引の再ビルドを管理」ウィンドウで使用される列 (続き)

列名	説明
無効化の理由 (Invalidation Reason)	このアクセス・パスが無効化された理由を表示します。次の値が指定可能です。 ユーザー要求 (User requested) (詳細については「無効化の理由タイプ」を参照してください。) 索引の作成またはビルド (Create or build Index) ロード (詳細については「無効化の理由タイプ」を参照してください。) 初期プログラム・ロード (IPL) 実行時エラー 変更 索引のビルドに失敗したジャーナル (Journal failed to build the index) 実行時に索引を修復可能としてマーク済み (Marked index as mendable during runtime) IPL 時に索引を修復可能としてマーク済み (Marked index as mendable during IPL) データの終わりを変更

表 41. 「索引の再ビルドを管理」ウィンドウで使用される列 (続き)

列名	説明
無効化の理由タイプ	このアクセス・パスが無効化された理由タイプを表示します。 ユーザー要求に可能な理由タイプは次のとおりです。 REORG のために無効 (Invalid because of REORG) コピーである (It is a copy) ファイルの変更 新しいメンバーの変換 (Converting new member) *FRCRBDAP への変更 (Change to *FRCRBDAP) *UNIQUE への変更 (Change to *UNIQUE) *REBLD への変更 (Change to *REBLD) LOAD に可能な理由タイプは次のとおりです。 索引は無効化用にマーク付けされたが、システムは無効化が実際に発生する前に破損された (The index was marked for invalidation but the system crashed before the invalidation could actually occur) 索引はロード時にオーバーレイされたデータ・スペース・ヘッダーに関連付けされたため、無効化された (The index was associated with the overlaid data space header during a load, therefore it was invalidated) 索引は IMPI 形式だった。(Index was in IMPI format.) ヘッダーは変換されたが、現在は RISC 形式で再ビルドするために無効化されている (The header was converted and now it is invalidated to be rebuilt in RISC format) RISC 索引は V5R1 形式に変換された (The RISC index was converted to V5R1 format) 索引は部分ロードのために無効化された (Index invalidated due to partial load) 索引は遅延保守の不一致のために無効化された (Index invalidated due to a delayed maintenance mismatch) 索引は埋め込みキーの不一致のために無効化された (Index invalidated due to a pad key mismatch) 索引は重要なフィールドのビットマップ修正のために無効化された (Index invalidated due to a significant fields bitmap fix) 索引は論理ページ・サイズの不一致のために無効化された (Index invalidated due to a logical page mismatch) 索引が復元されなかった。(Index was not restored.)ファイルは ACCPTH(*NO) を指定して保管されたか、ファイルの保管時に索引が存在しなかった。(File may have been saved with ACCPTH(*NO) or index did not exist when file was saved.) 索引が復元されなかった。(Index was not restored.)ファイルは ACCPTH(*NO) を指定して保管されたか、ファイルの保管時に索引が存在しなかった。(File may have been saved with ACCPTH(*NO) or index did not exist when file was saved.) ファイルが SAVACT(*SYSDFN) を指定して矛盾する状態で保管されたため、索引は再ビルドされた。(Index was rebuilt because file was saved in an inconsistent state with SAVACT(*SYSDFN).) 他の無効化コードの場合は、このフィールドは 0 を表示します。
再ビルドの推定時間	アクセス・パスの再ビルドにかかる推定合計時間。

表 41. 「索引の再ビルドを管理」ウィンドウで使用される列 (続き)

列名	説明
再ビルド開始時刻	再ビルドが開始された時刻。
経過した再ビルド時間	アクセス・パスの再ビルドの開始以来経過した合計時間
固有	アクセス・パスの行が固有かどうかを示します。次の値が指定可能です。 はい いいえ
照会の最後の使用	アクセス・パスが最後に使用された際のタイム・スタンプ。
照会統計の最後の使用 (Last Query Statistics Use)	アクセス・パスが最後に統計で使用された際のタイム・スタンプ。
照会使用回数	アクセス・パスが照会で使用された回数
照会統計使用回数 (Query Statistics Use Count)	アクセス・パスが統計で使用された回数
区画	索引のパーティションの詳細。次の値が指定可能です。 • <blank> (全パーティション用 (For all partitions) を意味します) • 各パーティション用 (For Each Partition) • パーティションの具体的な名前
所有者	このアクセス・パスの所有者のユーザー ID。
並列度	索引の再ビルドに使用されるプロセッサーの数。
短縮名	再ビルドされる索引を所有するファイルのシステム名。
テキスト	索引を所有するファイルのテキスト記述。

アクセス・パスの再作成の編集 (EDTRBDAP) コマンドを使ってアクセス・パスの再ビルドを管理することもできます。

関連情報

アクセス・パスの再作成

アクセス・パスの再作成の編集 (EDTRBDAP) コマンド

索引付けの方針

索引作成には 2 つのアプローチがあります。それは事前型のアプローチと事後型のアプローチです。名前が暗黙指定しているように、事前型の索引作成には、どの列が選択、結合、グループ化、および順序付けに最も頻繁に使用され、その後これらの列に対して索引が構築されるかを予想することが含まれます。事後型のアプローチでは、索引は最適化プログラムのフィードバック、照会のインプリメンテーション計画、およびシステム・パフォーマンス測定値を基にして作成されます。

任意の特定の照会ではなく、データベース・モデルおよびアプリケーションを基にした索引を最初に構築することは有用です。まず始めに、以下の基準を基にした基本索引の設計について考慮します。

- データベース・モデルを基にした基本および外部キー列
- 共通して使用される、自動車の作成とモデルのように従属した列を含むローカル選択列
- 共通して使用される、基本または外部キー列とみなされない結合列
- 共通して使用されるグループ化列

関連情報

DB2 for i5/OS の索引付けおよび統計方針

チューニングの事後型アプローチ

事後型チューニングを実行するには、索引のない提案されたアプリケーションのプロトタイプを構築し、いくつかの照会の実行を開始するか、または索引の初期設定を構築し、何が使用されるようになり、何が使用されないかを知るためにアプリケーションの実行を開始します。さらに小さいデータベースを使用する場合であっても、時間のかかる照会の実行が明らかに速くなります。

期待しているところまで実行していない既存のアプリケーションを理解し調整しようとするときにも、事後型のチューニング・メソッドが使用されます。 適切なデバッグおよびモニター・ツール (次のセクションで説明されている) を使用して、次のような 3 つのことを基本的に伝えるデータベースのフィードバック・メッセージが表示できます。

- 最適化プログラムがローカル選択に推奨する任意の索引
- 照会に使用される任意の一時索引
- 最適化プログラムが照会を実行するために選択するインプリメンテーション・メソッド

データベース・エンジンが、結合を実行するためまたは永続テーブルに対してグループ化および選択を実行するために一時索引を構築している場合、永続索引は一時索引の作成を除去しようとして、同じ列に対して構築されるはずです。ある場合には、一時索引は一時テーブルに対して構築されるので、それらのテーブルには永続索引を構築することはできません。前のセクションでリストされた最適化ツールを使用して、一時索引の作成、一時索引が作成された理由、および一時索引のキー列を注記することができます。

チューニングの事前型アプローチ

一般的に、照会で最も選択頻度の高い列のために索引を作成し、最も選択頻度の低い列のために統計を作成します。索引を作成することによって、最適化プログラムは、特定の列が選択的であることを確認して、照会をインプリメントするために索引を使用することができます。

完璧な基数索引では、列の順序が重要です。実際、最適化プログラムがデータ検索のために列を順序付けするかどうかで差が生じます。一般規則として、以下の方法で索引内で列の順序付けをします。

- 最初に等号述部。すなわち、「=」演算子を使用する任意の述部は最も高速で行の範囲を狭める可能性があり、したがって索引内で最初になるはずです。
- すべての述部に等号演算子がある場合、列を以下のように順序付ける。
  - 選択述部 + 結合述部
  - 結合述部 + 選択述部
  - 選択述部 + 列によるグループ
  - 選択述部 + 列による順序

上記のガイドラインに加えて、一般的に最も選択頻度の高いキー列は索引内で最初に置かれるはずです。

次の SQL ステートメントを考えてください。

```
SELECT b.col1, b.col2, a.col1
   FROM table1 a, table2 b
   WHERE b.col1='some_value' AND
     b.col2=some_number AND
     a.join_col=b.join_col
   GROUP BY b.col1, b.col2, a.col1
   ORDER BY b.col1
```

このように照会を使用して、事前型の索引作成のプロセスを始めることができます。基本的な規則は次のとおりです。

- 最も大きい、または最も共通して使用される照会の基数索引をカスタム・ビルドする。上記の照会を使用した例は次のとおりです。

  ```
  radix index over join column(s) - a.join_col and b.join_col
  radix index over most commonly used local  selection column(s) - b.col2
  ```

- 特別な OLAP (online analytical processing) 環境、またはさらに頻繁に使用されない照会の場合、照会で使用されているローカル選択列に対して単一キーの EVI を構築する。上記の照会を使用した例は次のとおりです。

  ```
  EVI over non-unique local selection columns - b.col1 and b.col2
  ```

効果的な索引のコーディング

以下に示すのは、DB2 for i5/OS が、使用できる索引の利点を活用することを可能にするコードを設計する場合に役立つ注意事項です。

数字変換の回避

列の値とホスト変数 (または定数値) とを比較する場合には、同じデータ・タイプと属性を指定するようにしてください。 DB2 for i5/OS は、ホスト変数または定数値の精度のほうが列の精度よりも高い場合には、指定された列について索引を使用しない場合があります。比較される 2 つの項目のデータ・タイプが異なっていると、DB2 for i5/OS はそのどちらか一方の値を変換する必要があり、結果が不正確なものになることがあります (計算機の精度に制約があるため)。

列と定数とが比較されるという問題を回避するには、次のように使用します。

- 同じデータ・タイプ
- 適用可能な場合には、同じ位取り
- 適用可能な場合には、同じ精度

たとえば、EDUCLVL がハーフワードの整数値 (SMALLINT) であるとすれば、 SQL を使用する場合、次のように指定します。

```
... WHERE EDUCLVL < 11 AND
                  EDUCLVL >= 2
```

次のような指定は避けてください。

```
... WHERE EDUCLVL < 1.1E1 AND
                  EDUCLVL > 1.3
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
... QRYSLT('EDUCLVL *LT 11 *AND ENUCLVL *GE 2')
```

次のような指定は避けてください。

```
... QRYSLT('EDUCLVL *LT 1.1E1 *AND EDUCLVL *GT 1.3')
```

| EDUCLVL 列について、索引が作成された場合には、最適化プログラムは、2 番目の例では索引を使用しない場合があります。その理由は、定数の精度が列の精度よりも高いためです。最適化プログラムは、定数を列の精度に変換しようとします。最初の例では、最適化プログラムは、精度が等しいので索引を使用しようとします。

算術式の回避

行選択述部の列と比較するオペランドとして算術式を絶対に指定しないでください。最適化プログラムでは、算術式と比較される列についての索引は使用しません。これによって列の索引が使用不可になることは

ないものの、索引での見積もりや、おそらく索引走査のキー位置決めの使用は妨げられることになります。失われる主な機能は、照会の最適化に役立つ統計を使用および抽出する機能です。

たとえば、SQL を使用する場合は次のように指定します。

```
... WHERE SALARY > 16500
```

次のような指定は避けてください。

```
... WHERE SALARY > 15000*1.1
```

文字ストリングの埋め込みの回避

固定長文字ストリングの列値とホスト変数または定数値とを比較するときは、同じデータ長を使用するようにしてください。 DB2 for i5/OS は、定数値またはホスト変数が列より長いと、索引を使用しない場合があります。

たとえば、EMPNO が CHAR(6) で、DEPTNO が CHAR(3) であるとすれば、たとえば、SQL を使用する場合は次のように指定します。

```
... WHERE EMPNO > '000300' AND
                  DEPTNO < 'E20'
```

次のような指定は避けてください。

```
... WHERE EMPNO > '000300 ' AND
                  DEPTNO < 'E20 '
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
... QRYSLT('EMPNO *GT "000300" *AND DEPTNO *LT "E20"')
```

次のような指定は避けてください。

```
... QRYSLT('EMPNO *GT "000300" *AND DEPTNO *LT "E20"')
```

% または _ で始まる LIKE パターンの使用の回避

パーセント記号 (%) および下線 (_) は、LIKE (OPNQRYF %WLDCRD) 述部のパターンの中では、選択したい行の列の値に類似する文字ストリングを指定します。文字ストリングの途中または終わりにこれらの文字を使用すると、索引の利点を活用することができます。

たとえば、SQL を使用する場合は次のように指定します。

```
... WHERE LASTNAME LIKE 'J%SON%'
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
... QRYSLT('LASTNAME *EQ %WLDCRD(''J*SON*'')')
```

ただし、文字ストリングの先頭で使用する場合、索引走査のキー位置決めを使用して走査する行数を限定するために、 DB2 for i5/OS は LASTNAME 列で定義されている索引を使用できなくなります。ただし、索引走査のキー選択は許可されます。たとえば、以下の照会では、索引走査のキー選択は使用できますが、索引走査のキー位置決めは使用できません。

SQL の場合:

```
... WHERE LASTNAME LIKE '%SON'
```

OPNQRYF の場合:

```
... QRYSLT('LASTNAME *EQ %WLDCRD(''*SON'')')
```

理想的には、% を付けるパターンを避ければ、述部でキーの処理の実行時のパフォーマンスを最良のものにすることができます。可能であれば、検索の部分ストリングを使って、索引走査のキー位置決めを使用できるかどうかを試してください。

たとえば、″Smithers″ という名前を探している場合に、″S%″ とだけタイプすると、この照会は ″S″ で始まるすべての名前を戻します。 ″Smi%″ が含まれる名前すべてを戻すよう、照会を調整する必要があります。部分ストリングの使用を強制することにより、長期的により良いパフォーマンスを得られる場合があります。

派生索引の使用

SQL 索引は、キーが式として指定された場所に作成できます。これは、「派生キー」という名前で呼ばれることもあります。

例えば、次のコマンドを参照してください。

```
CREATE INDEX TOTALIX ON EMPLOYEE(SALARY+BONUS+COMM AS TOTAL)
```

この例では、給与の合計が 50000 を超す、すべての従業員が戻されます。

```
SELECT * FROM EMPLOYEE
 WHERE SALARY+BONUS+COMM > 50000
 ORDER BY SALARY+BONUS+COMM
```

最適化プログラムは、索引プローブとともに索引 TOTALIX を使用して、WHERE 選択および配列基準を満たします。

派生キー索引の使用およびマッチングに関する特殊な考慮事項には、以下の項目が含まれます。

- 索引キーの定数には、照会のホスト変数に一致するものはありません。これには、データベース・マネージャーによって実行される暗黙的パラメーター・マーカー変換が含まれます。

  ```
  CREATE INDEX D_IDX1 ON EMPLOYEE (SALARY/12 AS MONTHLY)
  ```

  この例では、月給が 3000 を超す、すべての従業員が戻されます。

  ```
  long months = 12;
  EXEC SQL SELECT * FROM EMPLOYEE WHERE SALARY/:months > 3000
  ```

  ただし、この場合、最適化プログラムは索引を使用しません。これは、照会のホスト変数値月を索引の定数 12 に一致させるためのサポートがないためです。

  値 *NO を指定して QAQQINI オプション PARAMTER_MARKER_CONVERSION を使用すると、定数がパラメーター・マーカーに変換されることを防ぐことができ、派生索引キーのマッチングを向上させることができます。ただし、この QAQQINI 設定の使用はパフォーマンスに影響を及ぼすため、使用には注意が必要です。
- 一般に、索引の式は、照会の式に一致する必要があります。

  ```
  .... WHERE SALARY+COMM+BONUS > 50000
  ```

  この場合、 WHERE SALARY+COMM+BONUS は、索引キー SALARY+BONUS+COMM とは異なり、一致しません。
- 派生索引キーは可能な限り単純なまま保持することをお勧めします。一致する照会式および索引キー式が複雑になるほど、索引が使用される可能性は低くなります。
- CQE 最適化プログラムでは、派生キー索引の一致にするサポートが非常に制限されています。

| • WHERE 文節を指定する SQL 索引は、照会インプリメンテーションまたはコスト計算には使用されません。

関連資料

172 ページの『派生キー索引』

SQL CREATE INDEX ステートメントを使用して、SQL 式を使用する派生キー索引を作成できます。

関連情報

SQL Create Index ステートメント

分類順序と一緒に索引を使用する

以下のセクションでは、分類順序テーブルと一緒に索引を使用する方法について説明します。

選択、結合、またはグループ化で索引および分類順序を使用する

既存の索引を使用する前に、DB2 for i5/OS は列の属性 (選択列、結合列、またはグループ化列) が既存の索引内のキー列の属性と一致することを確認します。分類順序テーブルは、比較する必要がある追加の属性です。

照会に関連する分類順序テーブル (SRTSEQ パラメーターおよび LANGID パラメーターによって指定される) は、既存の索引を作成するために使用された分類順序テーブルと一致しなければなりません。 DB2 for i5/OS は分類順序テーブルを比較し、一致しない場合は、既存の索引を使用することはできません。

ただし、これには例外があります。照会に関連する分類順序テーブルが固有分類順序テーブル (*HEX を含む) の場合は、 DB2 for i5/OS は以下の演算子および述部を使用する選択列、結合列、またはグループ化列に対しては分類順序テーブルが指定されていないかのように機能します。

- 等号 (=) 演算子
- 不等号 (^= または <>) 演算子
- LIKE 述部 (OPNQRYF %WLDCRD および *CT)
- IN 述部 (OPNQRYF %VALUES)

キー列が列に一致していて、以下のいずれかの場合は、 DB2 for i5/OS は任意の既存の索引を自由に使用することができます。

- 索引に分類順序テーブルが含まれていない場合。または、
- 索引に固有分類順序テーブルが含まれている場合。

注:

1. テーブルは、照会に関連した固有分類順序テーブルに一致する必要はありません。
2. ビットマップ処理では、複数の索引を 1 つのテーブルについて使用する場合に、特別な考慮事項があります。複数の索引内に共通のキー列があり、そのフィールドが照会選択でも参照される場合、これらの索引は、同じ分類順序テーブルを使用するか、分類順序テーブルを使用しないようにする必要があります。

順序付けで索引および分類順序を使用する

順序付けの要求に応えるために、最適化プログラムが分類の実行を選択しない場合は、索引に関連する分類順序テーブルは、照会に関連する分類順序テーブルと一致しなければなりません。

分類を使用する場合は、変換は分類中に行われます。分類によって分類順序の要件が処理されるので、これによって、 DB2 for i5/OS は選択基準に合致する任意の既存の索引を使用することができます。

索引の例

以下の索引の例は、有効な索引を作成する上で役に立ちます。

例を示すために、3 つの索引を作成したとします。

索引 HEXIX は分類順序として *HEX を使用して作成したとします。

```
CREATE INDEX HEXIX ON STAFF (JOB)
```

索引 UNQIX は固有分類順序を使用して作成したとします。

```
CREATE INDEX UNQIX ON STAFF (JOB)
```

索引 SHRIX は同順位分類順序を使用して作成したとします。

```
CREATE INDEX SHRIX ON STAFF (JOB)
```

索引例: 分類順序テーブルを使用しない「等しい」選択

分類順序テーブルを使用しない「等しい」選択 (SRTSEQ(*HEX))。

```
SELECT * FROM STAFF
  WHERE JOB = 'MGR'
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   SRTSEQ(*HEX)
```

システムは、索引 HEXIX または索引 UNQIX を使用することができます。

索引例: 固有分類順序テーブルを使用した「等しい」選択

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した「等しい」選択。

```
SELECT * FROM STAFF
 WHERE JOB = 'MGR'
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは、索引 HEXIX または索引 UNQIX を使用することができます。

索引例: 同順位分類順序テーブルを使用した「等しい」選択

同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した「等しい」選択。

```
SELECT * FROM STAFF
 WHERE JOB = 'MGR'
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
QRYSLT('JOB *EQ ''MGR''')
SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは索引 SHRIX しか使用できません。

索引例: 固有分類順序テーブルを使用した「より大」選択

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した「より大」選択。

```
SELECT * FROM STAFF
 WHERE JOB > 'MGR'
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *GT ''MGR''')
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは索引 UNQIX しか使用できません。

索引例: 固有分類順序テーブルを使用した結合選択

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した結合選択。

```
SELECT * FROM STAFF S1, STAFF S2
 WHERE S1.JOB = S2.JOB
```

または JOIN 構文を使用する同じ照会。

```
SELECT *
 FROM STAFF S1 INNER JOIN STAFF S2
       ON S1.JOB = S2.JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE(STAFF STAFF)
   FORMAT(FORMAT1)
   JFLD((1/JOB 2/JOB *EQ))
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは、どの照会についても、索引 HEXIX または索引 UNQIX を使用することができます。

索引例: 同順位分類順序テーブルを使用した結合選択

同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した結合選択。

```
SELECT * FROM STAFF S1, STAFF S2
 WHERE S1.JOB = S2.JOB
```

または JOIN 構文を使用する同じ照会。

```
SELECT *
 FROM STAFF S1 INNER JOIN STAFF S2
       ON S1.JOB = S2.JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE(STAFF STAFF) FORMAT(FORMAT1)
   JFLD((1/JOB 2/JOB *EQ))
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは、どの照会についても、索引 SHRIX だけを使用できます。

索引例: 分類順序テーブルを使用しない順序付け

分類順序テーブル (SRTSEQ(*HEX)) を使用しない順序付け。

```
SELECT * FROM STAFF
 WHERE JOB = 'MGR'   ORDER BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   KEYFLD(JOB)
   SRTSEQ(*HEX)
```

システムは索引 HEXIX しか使用できません。

索引例: 固有分類順序テーブルを使用した順序付け

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した順序付け。

```
  SELECT * FROM STAFF
   WHERE JOB = 'MGR'   ORDER BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   KEYFLD(JOB) SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは索引 UNQIX しか使用できません。

索引例: 同順位分類順序テーブルを使用した順序付け

同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した順序付け。

```
  SELECT * FROM STAFF
   WHERE JOB = 'MGR'   ORDER BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   KEYFLD(JOB) SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは索引 SHRIX しか使用できません。

索引例: ALWCPYDTA(*OPTIMIZE) および固有分類順序テーブルを使用した順序付け

ALWCPYDTA(*OPTIMIZE) および固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した順序付け。

```
  SELECT * FROM STAFF
   WHERE JOB = 'MGR'   ORDER BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF))
   QRYSLT('JOB *EQ ''MGR''')
   KEYFLD(JOB)
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
   ALWCPYDTA(*OPTIMIZE)
```

システムは、選択に対して、索引 HEXIX または索引 UNQIX を使用することができます。順序付けは、*LANGIDUNQ 分類順序テーブルを使用して、分類時に実行されます。

索引例: 分類順序テーブルを使用しないグループ化

分類順序テーブル (SRTSEQ(*HEX)) を使用しないグループ化。

```
  SELECT JOB FROM STAFF
   GROUP BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT2)
   GRPFLD((JOB))
   SRTSEQ(*HEX)
```

システムは、索引 HEXIX または索引 UNQIX を使用することができます。

索引例: 固有分類順序テーブルを使用したグループ化

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用したグループ化。

```
SELECT JOB FROM STAFF
 GROUP BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT2)
   GRPFLD((JOB))
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは、索引 HEXIX または索引 UNQIX を使用することができます。

索引例: 同順位分類順序テーブルを使用したグループ化

同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用したグループ化。

```
SELECT JOB FROM STAFF
 GROUP BY JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT2)
   GRPFLD((JOB))
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは索引 SHRIX しか使用できません。

以下の例では、JOB 列と SALARY 列に関してさらに 3 つの索引が作成されています。 CREATE INDEX ステートメントの例を先に示します。

索引 HEXIX2 は、分類順序として *HEX を使用して作成されています。

```
CREATE INDEX HEXIX2 ON STAFF (JOB, SALARY)
```

索引 UNQIX2 は固有分類順序を使用して作成されています。

```
CREATE INDEX UNQIX2 ON STAFF (JOB, SALARY)
```

索引 SHRIX2 は、同順位分類順序を使用して作成されています。

```
CREATE INDEX SHRIX2 ON STAFF (JOB, SALARY)
```

索引例: 固有分類順序テーブルを使用した、同じ列に関する順序付けとグループ化

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した、同じ列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY JOB, SALARY
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(JOB SALARY)
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

システムは、UNQIX2 を使用してグループ化要件と順序付け要件の両方を満たすことができます。索引 UNQIX2 が存在しなかった場合、システムは、 *LANGIDUNQ の分類順序テーブルを使用して索引を作成します。

索引例: ALWCPYDTA(*OPTIMIZE) および固有分類順序テーブルを使用した、同じ列に関する順序付けとグループ化

ALWCPYDTA(*OPTIMIZE) および固有の重みを持つ分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した、同じ列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY JOB, SALARY
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(JOB SALARY)
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
   ALWCPYDTA(*OPTIMIZE)
```

システムは、UNQIX2 を使用してグループ化要件と順序付け要件の両方を満たすことができます。索引 UNQIX2 が存在しない場合には、システムは以下のいずれかの処置を行います。

- *LANGIDUNQ の分類順序テーブルを使用して索引を作成するか、または
- グループ化要求に応えるために索引 HEXIX2 を使用し、順序付け要求に応えるために分類を実行します。

索引例: 同順位分類順序テーブルを使用した、同じ列に関する順序付けとグループ化

同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した、同じ列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY JOB, SALARY
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(JOB SALARY)
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは、SHRIX2 を使用してグループ化要件と順序付け要件の両方を満たすことができます。索引 SHRIX2 が存在しない場合、システムは、 *LANGIDSHR の分類順序テーブルを使用して索引を作成します。

索引例: ALWCPYDTA(*OPTIMIZE) および同順位分類順序テーブルを使用した、同じ列に関する順序付けとグループ化

ALWCPYDTA(*OPTIMIZE) および同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した、同じ列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY JOB, SALARY
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(JOB SALARY)
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
   ALWCPYDTA(*OPTIMIZE)
```

システムは、SHRIX2 を使用してグループ化要件と順序付け要件の両方を満たすことができます。索引 SHRIX2 が存在しない場合、システムは、 *LANGIDSHR の分類順序テーブルを使用して索引を作成します。

索引例: 固有分類順序テーブルを使用した、異なる列に関する順序付けとグループ化

固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した、異なる列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY SALARY, JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(SALARY JOB)
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

システムは、索引 HEXIX2 または索引 UNQIX2 を使用して、グループ化要件を満たすことができます。グループ化の結果を含む一時結果が作成されます。次に、順序付け要求に応えるために、*LANGIDUNQ 分類順序テーブルを使用して、一時結果に関する一時索引が作成されます。

索引例: ALWCPYDTA(*OPTIMIZE) および固有分類順序テーブルを使用した、異なる列に関する順序付けとグループ化

ALWCPYDTA(*OPTIMIZE) および固有分類順序テーブル (SRTSEQ(*LANGIDUNQ) LANGID(ENU)) を使用した、異なる列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY SALARY, JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(SALARY JOB)
   SRTSEQ(*LANGIDUNQ) LANGID(ENU)
   ALWCPYDTA(*OPTIMIZE)
```

システムは、索引 HEXIX2 または索引 UNQIX2 を使用して、グループ化要件を満たすことができます。順序付け要求に応えるために、分類が実行されます。

索引例: ALWCPYDTA(*OPTIMIZE) および同順位分類順序テーブルを使用した、異なる列に関する順序付けとグループ化

ALWCPYDTA(*OPTIMIZE) および同順位分類順序テーブル (SRTSEQ(*LANGIDSHR) LANGID(ENU)) を使用した、異なる列に関する順序付けとグループ化。

```
SELECT JOB, SALARY FROM STAFF
 GROUP BY JOB, SALARY
 ORDER BY SALARY, JOB
```

OPNQRYF コマンドを使用する場合、次のように指定します。

```
OPNQRYF FILE((STAFF)) FORMAT(FORMAT3)
   GRPFLD(JOB SALARY)
   KEYFLD(SALARY JOB)
   SRTSEQ(*LANGIDSHR) LANGID(ENU)
   ALWCPYDTA(*OPTIMIZE)
```

システムは、索引 SHRIX2 を使用してグループ化要件を満たすことができます。順序付け要求に応えるために、分類が実行されます。

データベース・パフォーマンスに関するアプリケーション設計のヒント

データベース・パフォーマンスを最大限向上させるために SQL アプリケーションの設計時に適用できる設計上のヒントを説明します。

ライブ・データの使用

ライブ・データ とは、データベース・マネージャーがデータのコピーを作成せずにデータを取り出すときに使用するアクセスのタイプのことを言います。このタイプのアクセスを使用すると、プログラムに返されるデータは、常にデータベースの中のデータの現行値を反映します。プログラマーは、データベース・マネージャーがデータのコピーを使用するか、データを直接取り出すかを制御できます。このためには、プリコンパイラー・コマンドまたは SQL の開始 (STRSQL) コマンドでデータのコピー可能 (ALWCPYDTA) パラメーターを指定します。

ALWCPYDTA(*NO) を指定すると、データベース・マネージャーは常にライブ・データを使用します。ほとんどの場合、ライブ・データへのアクセスの強制は、パフォーマンスに悪影響をもたらします。なぜならこれは、最適化プログラムが照会のインプリメントに使用するかもしれない可能な計画の選択を大幅に制限するからです。このため、これはほとんどの場合回避されるべきです。しかし、単純な照会を伴う特殊な場合に、ライブ・データ・アクセスでは、取り出すデータを再表示するためにカーソルをクローズしてから再びオープンする必要がないため、これをパフォーマンス上の利点として使用することができます。画面上にリストを作り出すアプリケーションが、この利点を示す一例です。リストの要素を一度に 20 個しか表示できない画面の場合、アプリケーション・プログラマーは、最初の 20 個の要素が表示された後で次の 20 行の表示を要求することができます。 i5/OS オペレーティング・システム以外のオペレーティング・システム用に設計された代表的な SQL アプリケーションでは、次のような構造になっています。

```
EXEC SQL
    DECLARE C1 CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
     ORDER BY EMPNO
END-EXEC.

EXEC SQL
    OPEN C1
END-EXEC.

*    PERFORM FETCH-C1-PARA  20 TIMES.

     MOVE EMPNO to LAST-EMPNO.

EXEC SQL
    CLOSE C1
END-EXEC.

*    Show the display and wait for the user to indicate that
*    the next 20 rows should be displayed.

EXEC SQL
    DECLARE C2 CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
    WHERE EMPNO > :LAST-EMPNO
    ORDER BY EMPNO
END-EXEC.
```

```
EXEC SQL
    OPEN C2
END-EXEC.

*    PERFORM FETCH-C21-PARA  20 TIMES.

*    Show the display with these 20 rows of data.

EXEC SQL
    CLOSE C2
END-EXEC.
```

上の例では、リストを継続して現行データを取り出すために、もう 1 つのカーソルを再びオープンする必要があります。そのため、追加の ODP の作成が必要になることがあり、システムでの処理時間が増加することになります。この例に代えて、プログラマーは ALWCPYDTA(*NO) を指定して、以下の SQL ステートメントでアプリケーションを設計することができます。

```
EXEC SQL
    DECLARE C1 CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
     ORDER BY EMPNO
END-EXEC.

EXEC SQL
    OPEN C1
END-EXEC.

*    Display the screen with these 20 rows of data.

*    PERFORM FETCH-C1-PARA  20 TIMES.

*    Show the display and wait for the user to indicate that
*    the next 20 rows should be displayed.

*    PERFORM FETCH-C1-PARA  20 TIMES.

EXEC SQL
    CLOSE C1
END-EXEC.
```

上の例では、複数行用 FETCH ステートメントで FOR 20 ROWS 文節を使用して、1 回の操作で 20 行を取り出すことにより、照会のパフォーマンスが向上します。

関連情報

SQL 対話式セッションの開始 (STRSQL) コマンド

オープン操作の回数の削減

SQL データ操作言語ステートメントでは、データへのオープン・データ・パス (ODP) を作成するためにデータベースのオープン操作が必要になります。オープン・データ・パスは、テーブルに対する入出力操作すべてを行うためのパスです。ある意味では、これは SQL アプリケーションとテーブルを結び付けるものです。プログラムでのオープン操作の回数は、パフォーマンスに大きく影響します。

データベースのオープン操作は次のステートメントで行われます。

- OPEN ステートメント
- SELECT INTO ステートメント
- VALUES 文節を伴う INSERT ステートメント
- WHERE 条件を伴う UPDATE ステートメント

- WHERE CURRENT OF カーソル文節および演算子または関数を参照する SET 文節を伴う UPDATE ステートメント
- 式を含む SET ステートメント
- 式を含む VALUES INTO ステートメント
- WHERE 条件を伴う DELETE ステートメント

選択ステートメントを伴う INSERT ステートメントには、オープン操作が 2 回必要です。また、副照会の形式によっては、副選択 1 回ごとに 1 回のオープンが必要になる場合があります。

オープンの回数を最小限にするために、DB2 for i5/OS は次の場合を除いて、オープン・データ・パス (ODP) をオープンのままにしておき、ステートメントの再実行の際に再度使用します。

- ODP がホスト変数を用いてサブセットの一時索引を作成している場合。 i5/OS データベース・サポートは、 SQL ステートメントに指定された行選択基準に該当する行のみの項目からなる一時索引を作成することがあります。行の選択でホスト変数が使用されている場合、一時索引には、そのホスト変数内の他の異なる値に対応する項目が含まれていません。
- ホスト変数値に順序付けを指定している場合。
- ODP のオープン後に SQL ステートメントの実行に影響を与えるデータベース・ファイルの一時変更 (OVRDBF) または一時変更の削除 (DLTOVR) の CL コマンドを出した場合。 ODP は DB2 for i5/OS によってオープンされます。

  注: 参照しているテーブルの名前に影響を与える一時変更のみが、そのプログラム呼び出しで ODP のクローズを引き起こします。
- 結合が複合結合で、結合の中間ステップを入れるための一時テーブルが必要な場合。
- 複合分類が関係する場合。この場合には一時ファイルが必須になり、再使用可能でないこともあります。
- 最後のオープン以後に行われたライブラリー・リストの変更により、システム命名モードの非修飾の参照によって選択されたテーブルが変更される場合。
- ハッシュ結合を使って CQE 最適化プログラムによって結合が実施された場合。

組み込み静的 SQL の場合、DB2 for i5/OS が再使用するのは、同じステートメントでオープンした ODP に限ります。プログラムで同一のステートメントが後でコーディングされていても、他のステートメントで同じ ODP が再使用されるわけではありません。プログラム内で同じステートメントを何回も実行しなければならない場合には、そのステートメントをサブルーチンの形で 1 回コーディングしておき、そのステートメントを実行するたびにそのサブルーチンを呼び出してください。

DB2 for i5/OS によりオープンされた ODP がクローズされるのは、次のいずれかが発生した時点です。

- CLOSE、INSERT、UPDATE、DELETE、または SELECT INTO の各ステートメントが完了し、ODP が再利用できない一時結果またはサブセットの一時索引を必要とする場合。
- リソースの再使用 (RCLRSC) コマンドが出される場合。呼び出しスタック上の最初の COBOL プログラムが終了したとき、または COBOL プログラムが STOP RUN COBOL ステートメントを出したときに、リソースの再使用 (RCLRSC) が出されます。リソースの再使用 (RCLRSC) は、CLOSQLCSR(*ENDJOB) を使用してプリコンパイルされたプログラム用に作成された ODP をクローズしません。非デフォルト活動化グループと リソースの再使用 (RCLRSC) との対話については、次の資料を参照してください。
  - WebSphere® Development Studio: ILE C/C++ Programmer's Guide
  - WebSphere Development Studio: ILE COBOL プログラマーの手引き

– WebSphere Development Studio: ILE RPG プログラマーの手引き

- 呼び出しスタック上の、SQL ステートメントを含む最後のプログラムが終了したとき。ただし、CLOSQLCSR(*ENDJOB) を使用してプリコンパイルされたプログラム用、または CLOSQLCSR (*ENDACTGRP) を使用してプリコンパイルされたモジュール用に作成された ODP を除きます。
- CONNECT (タイプ 1) ステートメントが活性化グループのアプリケーション・サーバーを変更したとき、その活性化グループ用に作成されたすべての ODP はクローズされます。
- DISCONNECT ステートメントがアプリケーション・サーバーへの接続を終了すると、そのアプリケーション・サーバー用のすべての ODP はクローズされます。
- 解放された接続を正常な COMMIT によって終了すると、そのアプリケーション・サーバー用のすべての ODP はクローズされます。
- オプション・ファイルの照会 (QAQQINI) パラメーターによって指定されるオープン・カーソルのしきい値 OPEN_CURSOR_THRESHOLD に達した時。
- SQL LOCK TABLE または CL ALCOBJ OBJ((ファイル名 *FILE *EXCL)) CONFLICT(*RQSRLS) コマンドは、指定されたテーブルと関連付けられているすべての疑似クローズ・カーソルをクローズします。
- アプリケーションがクローズを要求した際に DB2 for i5/OS によってオープンされたままになったオープン・データ・パスは、ALCOBJ CL コマンドを使用して、特定のファイルに対して強制的にクローズさせることができます。アプリケーションがカーソルのクローズを要求しなかった場合、これによって ODP が強制的にクローズされることはありません。このコマンドの構文は、ALCOBJ OBJ((library/file *FILE *EXCL)) CONFLICT(*RQSRLS) です。

システムが ODP をオープンのままにするかどうかを制御するには、以下のようにします。

- SQL ステートメントを出したプログラムが常に呼び出しスタックに残っているようにアプリケーションを設計します。
- CLOSQLCSR(*ENDJOB) パラメーターまたは CLOSQLCSR(*ENDACTGRP) パラメーターを使用します。
- 照会オプション・ファイル (QAQQINI) の OPEN_CURSOR_THRESHOLD および OPEN_CURSOR_CLOSE_COUNT パラメーターを指定することにより。

SET 文節の式に演算子または関数が入っている場合には、各 UPDATE WHERE CURRENT OF の最初の実行の時点で、システムはオープン操作を行います。このオープンは、ホスト言語コードでその関数または演算をコーディングすることにより避けることができます。

たとえば、次の UPDATE が実行されると、システムはオープン操作を行います。

```
EXEC SQL
 FETCH EMPT INTO :SALARY
END-EXEC.

EXEC SQL
 UPDATE CORPDATA.EMPLOYEE
   SET SALARY = :SALARY + 1000
   WHERE CURRENT OF EMPT
END-EXEC.
```

この代わりとして、次のようにコーディングすると、オープンは避けられます。

```
EXEC SQL
 FETCH EMPT INTO :SALARY
END EXEC.

ADD 1000 TO SALARY.
```

```
EXEC SQL
 UPDATE CORPDATA.EMPLOYEE
   SET SALARY = :SALARY
   WHERE CURRENT OF EMPT
END-EXEC.
```

SQL ステートメントの結果として全オープンになるかどうかは、様々な方法で判別することができます。好ましいのは、データベース・モニターを使用する方法や、デバッグがアクティブの時に出されたメッセージを見る方法です。さらに、CL コマンドのジョブのトレース (TRCJOB) またはジャーナルの表示 (DSPJRN) を使用することもできます。

関連情報

資源再利用 (RCLRSC) コマンド

ジョブのトレース (TRCJOB) コマンド

ジャーナルの表示 (DSPJRN) コマンド

RPG

COBOL

C および C++

カーソル位置の保持

カーソル位置を保持することによりパフォーマンスを向上させることができます。

非 ILE プログラム呼び出しの場合のカーソル位置の保存

非 ILE プログラム呼び出しの場合、SQL カーソル・クローズ (CLOSQLCSR) パラメーターを使用すると、以下の有効範囲を指定することができます。

- カーソル
- 準備されたステートメント
- ロック

CLOSQLCSR パラメーターを正しく使用すると、必要な SQL OPEN、PREPARE、および LOCK ステートメントの数を減らすことができます。さらに、複数回のプログラム呼び出しを通じてカーソル位置を保存できるので、アプリケーションを単純化することができます。

***ENDPGM**

これは、非 ILE プリコンパイラーのデフォルトです。このオプションを選択すると、カーソルは、それをオープンしたプログラムが呼び出しスタック上にある間だけ、オープンのままであり、アクセス可能です。プログラムが終了すると、SQL カーソルは使用できなくなります。準備されたステートメントもプログラムが終了すると消滅します。しかし、ロックは、呼び出しスタック上の最後の SQL プログラムが完了するまで残っています。

***ENDSQL**

このオプションを選択すると、プログラムによって作成された SQL カーソルと準備されたステートメントは、呼び出しスタック上の最後の SQL プログラムが完了するまでオープンのままです。これらは、同じプログラムを別に呼び出すことによる以外は、他のプログラムが使用することはできません。ロックは、呼び出しスタック内の最後の SQL プログラムが完了するまで残っています。

***ENDJOB**

このオプションを選択すると、ジョブが持続する間、SQL カーソル、準備されたステートメント、およびロックを活動状態にしておくことができます。スタック上の最後の SQL プログラムが完了

したとき、 *ENDJOB プログラムによって作成された SQL リソースがあれば、そのリソースは活動状態のままになっています。ロックは効力をもったままです。 CLOSE、COMMIT、または ROLLBACK ステートメントによって明示的にクローズされていない SQL カーソルはオープンのままです。準備されたステートメントは、同じプログラムの次回の呼び出し以降でも引き続いて使用できます。

関連資料

212 ページの『プリコンパイル・オプションの使用によるデータベース・パフォーマンスの向上』
パフォーマンスが向上するように SQL プログラムを作成するためのプリコンパイル・オプションがいくつか用意されています。これらのオプションを使用すると、アプリケーションの働きに影響することがあるので、これらはあくまでもオプションです。そのために、これらのパラメーターのデフォルト値は、以前のリリースからアプリケーションを正しくマイグレーションできるような値になっています。ただし、他のオプションを指定してもパフォーマンスを向上することができます。

複数の ILE プログラム呼び出しを通じてのカーソル位置の保存

ILE プログラム呼び出しの場合、SQL カーソル・クローズ (CLOSQLCSR) パラメーターを使用すると、次の有効範囲を指定することができます。

- カーソル
- 準備されたステートメント
- ロック

CLOSQLCSR パラメーターを正しく使用すると、必要な SQL OPEN、PREPARE、および LOCK ステートメントの数を減らすことができます。さらに、複数回のプログラム呼び出しを通じてカーソル位置を保存できるので、アプリケーションを単純化することができます。

***ENDACTGRP**

これは、ILE プリコンパイラーのデフォルトです。このオプションを選択すると、SQL カーソルおよび準備されたステートメントは、プログラムがその下で実行中である活性化グループが終了するまでオープンしたままです。これらは、同じプログラムを別に呼び出すことによる以外は、他のプログラムが使用することはできません。ロックは活性化グループが終了するまで残っています。

***ENDMOD**

このオプションを選択すると、カーソルはそれをオープンしたモジュールが活動状態である間だけ、オープンのままであり、アクセス可能です。モジュールが終了すると、SQL カーソルは使用できなくなります。準備されたステートメントもモジュールが終了すると消滅します。しかし、ロックは、呼び出しスタック内の最後の SQL プログラムが完了するまで残っています。

すべてのプログラム呼び出しに関するカーソル位置の保存の一般規則

CLOSQLCSR(*ENDPGM) または CLOSQLCSR(*ENDMOD) のいずれかでコンパイルしたプログラムを使用している場合は、データにアクセスするために、プログラムまたはモジュールを呼び出すたびにカーソルをオープンする必要があります。 SQL プログラムまたはモジュールが何回も呼び出される予定になっていて、しかも再使用可能な ODP の利点を利用したい場合には、プログラムまたはモジュールを終了する前にカーソルを明示的にクローズする必要があります。

CLOSQLCSR パラメーターを使用して、*ENDSQL、*ENDJOB、または *ENDACTGRP を指定すると、呼び出しのたびに OPEN ステートメントと CLOSE ステートメントを実行する必要がなくなります。実行するステートメントが少なくなる上、プログラムまたはモジュールへの複数の呼び出しの間でカーソル位置を保存することができます。

以下の SQL ステートメントの例は、 CLOSQLCSR パラメーターの使用による利点を理解するのに役立ちます。

```
EXEC SQL
 DECLARE DEPTDATA CURSOR FOR
   SELECT EMPNO, LASTNAME
     FROM CORPDATA.EMPLOYEE
     WHERE WORKDEPT = :DEPTNUM
END-EXEC.

EXEC SQL
 OPEN DEPTDATA
END-EXEC.

EXEC SQL
 FETCH DEPTDATA INTO :EMPNUM, :LNAME
END-EXEC.

EXEC SQL
 CLOSE DEPTDATA
END-EXEC.
```

このプログラムが別の SQL プログラムから複数回呼び出される場合には、このプログラムは再使用可能な ODP を使用することができます。これは、このプログラムに対する呼び出しから次の呼び出しまでの間、SQL が活動状態のままである限り、 OPEN ステートメントではデータベース・オープンの操作の必要がないことを意味します。しかし、各 OPEN ステートメントの後、カーソルは最初の結果の行に位置付けられたままであり、 FETCH ステートメントにより常に最初の行へ戻ります。

次の例では、CLOSE ステートメントが取り除かれています。

```
EXEC SQL
 DECLARE DEPTDATA CURSOR FOR
   SELECT EMPNO, LASTNAME
     FROM CORPDATA.EMPLOYEE
     WHERE WORKDEPT = :DEPTNUM
END-EXEC.

  IF CURSOR-CLOSED IS = TRUE THEN
EXEC SQL
 OPEN DEPTDATA
END-EXEC.

EXEC SQL
 FETCH DEPTDATA INTO :EMPNUM, :LNAME
END-EXEC.
```

このプログラムが *ENDJOB オプションまたは *ENDACTGRP オプションを指定してプリコンパイルされていて、活性化グループが活動状態のままの場合には、カーソル位置は維持管理されます。カーソル位置は、以下の場合にも維持管理されます。

- プログラムが *ENDSQL オプションを指定してプリコンパイルされている場合。
- SQL がプログラムの複数の呼び出しの間で活動状態のままである場合。

この方式によると、プログラムの呼び出しのたびにカーソル位置にある次の行が取り出されることになります。後続のデータ要求では、OPEN ステートメントは不要であり、実際には、SQLCODE -502 で失敗します。このエラーは無視することができますが、 OPEN をスキップするコードを追加することもできます。そのためには、最初に FETCH ステートメントを使用し、 FETCH 操作が正常に実行されなかった場合にのみ OPEN ステートメントを実行するようにします。

この方法は、準備されたステートメントにも適用されます。最初に EXECUTE を試み、それが正常に実行されなかった場合に PREPARE を実行するように、プログラムを組むことができます。その結果、正しい

CLOSQLCSR オプションが選択されていると仮定すれば、 PREPARE はそのプログラムへの最初の呼び出しでのみ必要となります。もちろん、そのプログラムへの呼び出しと呼び出しとの間でステートメントの変更がある場合には、いずれの場合にも、PREPARE を実行する必要があります。

主プログラムも最初の呼び出しでのみ特別なパラメーターを送ることによって上記の制御を行うことができます。この特別なパラメーター値は、これが最初の呼び出しであるためサブプログラムは OPEN、PREPARE、および LOCK を実行しなければならないことを指示するものです。

注: COBOL プログラムを使用する場合には、 STOP RUN ステートメントを使用しないようにしてください。呼び出しスタック上にある最初の COBOL プログラムが終了するか、 STOP RUN ステートメントが実行されると、リソース再利用 (RCLRSC) 操作が実行されます。この操作により SQL カーソルはクローズされます。 *ENDSQL オプションは希望通りには作動しません。

データベース・パフォーマンス向上のためのプログラミング方法

照会のコーディングを変更することによって、そのパフォーマンスを向上させることができます。

OPTIMIZE 文節の使用

あるカーソルに関する結果のテーブル全体を検索することを予定していないアプリケーションでは、OPTIMIZE 文節を使用すると、パフォーマンスを向上させることができます。 Query 最適化プログラムは、OPTIMIZE 文節に指定した値を使用して行のサブセットを検索するための見積コストを変更します。

次の照会では 1000 行が戻されるものとします。

```
EXEC SQL
    DECLARE C1 CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
      WHERE WORKDEPT = 'A00'
    ORDER BY LASTNAME
    OPTIMIZE FOR 100 ROWS
END EXEC.
```

注: 上記の OPTIMIZE 文節で使用できる値は、1 から 9999999 または ALL です。

この場合、最適化プログラムは次のようにしてコストを計算します。

最適化率 = n 行分の最適化の値/応答セット内の見積行数

一時的に作成した索引を使用する場合のコスト:

```
                応答セットの行を検索するためのコスト
            +   索引を作成するためのコスト
            +   一時索引を使用して該当行をもう一度検索
                    するためのコスト * 最適化率
```

SORT を使用する場合のコスト:

```
                応答セットの行を検索するためのコスト
            +   SORT 入力処理のコスト
            +   SORT 出力処理のコスト * 最適化率
```

既存の索引を使用する場合のコスト:

```
                応答セットの行を検索するためのコスト    * 最適化率
```

上記の例では、索引を分類する場合または索引を作成する場合の見積コストは最適化率で調整されません。このため、最適化プログラムは最適化と前処理の要件のバランスをとることができます。最適数が結果のテ

ーブル内の行数より大きい場合には、コスト見積もりに対して調整は行われません。照会に OPTIMIZE 文節の指定がない場合には、ステートメント・タイプ、指定された ALWCPYDTA の値、または出力装置に基づくデフォルト値が使用されます。

ステートメント・タイプ	ALWCPYDTA(*OPTIMIZE)	ALWCPYDTA(*YES または *NO)
DECLARE CURSOR	結果のテーブル内の行数	結果のテーブル内の行数または 3%
組み込み選択	2	2
表示装置への INTERACTIVE 選択出力	結果のテーブル内の行数または 3%	結果のテーブル内の行数または 3%
プリンターまたはデータベース・テーブルへの INTERACTIVE 選択出力	結果のテーブル内の行数	結果のテーブル内の行数

OPTIMIZE 文節は次の場合に照会の最適化に影響を及ぼします。

- 既存の索引を使用する場合 (小さい数の指定によるもの)
- 索引の作成を可能にする場合または応答セット内に可能な行数として大きい数を指定することにより分類あるいはハッシュを実行する場合

関連情報

選択ステートメント

FETCH FOR n ROWS の使用

FETCH ステートメントを何回も連続して実行するアプリケーションは、FETCH FOR n ROWS を使用すれば、パフォーマンスを改善することができます。この文節を使用すると、1 回の FETCH で、テーブルから複数行のデータを検索し、それをホスト構造配列または行記憶域に入れることができます。

FETCH ステートメントを使用している SQL アプリケーションは、FOR n ROWS 文節がなくても、複数行を取り出すための複数行用 FETCH ステートメントを使用することによって改善することができます。FETCH でホスト構造配列または行記憶域に記入した後、アプリケーションは、その配列または記憶域内のデータでループして、それぞれの行を処理することができます。 SQL 実行時間が呼び出されるのは一度だけで、しかもすべてのデータが同時にアプリケーション・プログラムに返されているので、ステートメントの実行速度は向上します。

SQL 実行時間がテーブルから検索する行を、データベース・マネージャーがブロック化できるようにアプリケーション・プログラムを変更することもできます。

次の表では、プログラムは 100 行分の FETCH を行ってその行をアプリケーションに組み入れることを試みています。ただし、テーブルでブロック化が実行可能なときに SQL 実行時間の呼び出しの回数とデータベース・マネージャーの呼び出しの回数が異なっているので注意してください。

表 42. FETCH ステートメントを使用する呼び出しの回数

	ブロック化を使用しないデータベース・マネージャー	ブロック化を使用するデータベース・マネージャー
単一行用 FETCH ステートメント	SQL の呼び出しは 100 回、データベースの呼び出しは 100 回	SQL の呼び出しは 100 回、データベースの呼び出しは 1 回
複数行用 FETCH ステートメント	SQL 実行時間の呼び出しは 1 回、データベースの呼び出しは 100 回	SQL 実行時間の呼び出しは 1 回、データベースの呼び出しは 1 回

関連情報

FETCH FOR n ROWS 使用時に SQL ブロック化のパフォーマンスを向上させる

FETCH FOR n ROWS を使用する際は、以下の点についてパフォーマンス上の特別な考慮が必要です。

以下の場合には、SQL のブロック化のパフォーマンスを向上させることができます。

- ホスト構造配列内の属性情報または行記憶域に関連する記述子が、検索する列の属性と一致している場合。
- アプリケーションが、1 回の複数行用 FETCH 呼び出しで、できるだけ多くの行を検索する場合。複数行用の FETCH 要求のブロック化因数は、システムのページ・サイズや OVRDBF コマンドの SEQONLY パラメーターでは制御されません。この因数は、複数行用 FETCH 要求で要求した行数によって制御されます。
- プログラム内の同一のカーソルに対して単一行用 FETCH 要求と複数行用 FETCH 要求を同時に使用しない場合。 1 つのカーソルに対して出された 1 つの FETCH が複数行用 FETCH として扱われると、そのカーソルに対するすべての FETCH が複数行用 FETCH として処理されます。その場合、各単一行用 FETCH 要求は、 1 行分の複数行用 FETCH として扱われることになります。
- PRIOR、CURRENT、および RELATIVE の各スクロール・オプションが複数行用 FETCH ステートメントと一緒に使用されない場合。アプリケーションがカーソルを任意に移動できるようにするには、データベース・マネージャーは、そのアプリケーションと同じカーソル位置を維持していなければなりません。したがって、SQL 実行時間は上記のオプションを指定して 1 つのスクロール可能カーソルに対して出されたすべての FETCH 要求を複数行 FETCH 要求として処理します。

INSERT n ROWS の使用

INSERT ステートメントを何回も連続して実行するアプリケーションは、INSERT n ROWS を使用すれば、改善することができます。この文節を使用すると、ホスト構造配列から 1 行または複数行のデータを取り出してそれを目的テーブルに挿入することができます。この配列は、構造の各要素が目的テーブルの各列に対応しているような構造の配列になっていなければなりません。

INSERT...VALUES ステートメント (n ROWS 文節を指定していないもの) でループを行う SQL アプリケーションは、 INSERT n ROWS ステートメントを使用して複数行をテーブルに挿入すれば、改善することができます。アプリケーションがループした結果、ホスト配列が行で満たされた後、 INSERT n ROWS ステートメントを 1 回実行すると、その配列全体をテーブルに挿入することができます。 SQL 実行時間が呼び出されるのは一度だけで、しかもすべてのデータが同時に目的テーブルに挿入されているので、ステートメントの実行速度は向上します。

次の表では、プログラムは 100 行分の INSERT を行ってその行をテーブルに挿入することを試みています。ただし、この表では、ブロック化が実行可能なときに SQL 実行時間の呼び出しの回数とデータベース・マネージャーの呼び出しの回数が異なっているので注意してください。

表 43. INSERT ステートメントを使用する呼び出しの回数

	ブロック化を使用しないデータベース・マネージャー	ブロック化を使用するデータベース・マネージャー
単一行用 INSERT ステートメント	SQL 実行時間の呼び出しは 100 回、データベースの呼び出しは 100 回	SQL 実行時間の呼び出しは 100 回、データベースの呼び出しは 1 回
複数行用 INSERT ステートメント	SQL 実行時間の呼び出しは 1 回、データベースの呼び出しは 100 回	SQL 実行時間の呼び出しは 1 回、データベースの呼び出しは 1 回

関連情報

データベース・マネージャーのブロック化の制御

パフォーマンスの向上を図るために、SQL 実行時間は、可能な限りデータベース・マネージャーから一度に 1 ブロックずつ行を取り出して挿入しようとします。

ブロック化は、必要ならば、SQL ステートメントを含んでいるアプリケーション・プログラムを呼び出す前に CL コマンドのデータベース・ファイル一時変更 (OVRDBF) で SEQONLY パラメーターを使用することにより制御することができます。あるいは CRTSQLxxx コマンドで ALWBLK パラメーターを指定することもできます。

データベース・マネージャーは、次の場合にはブロック化を許しません。

- カーソルが更新可能または削除可能の場合。
- 行とフィードバック情報の長さの合計が 32767 より大きい場合。フィードバック情報の最小サイズは 11 バイトです。フィードバック・サイズは、カーソルによって使用される索引のキー列のバイト数だけ、およびヌルが許されるキー列がある場合は、その数だけ増加します。
- COMMIT(*CS) が指定され、ALWBLK(*ALLREAD) が指定されていない場合。
- COMMIT(*ALL) が指定され、次が当てはまる場合。
  - SELECT INTO ステートメントまたはブロック化 FETCH ステートメントが使用されていない場合。
  - 照会が列関数を使用していなかったり、GROUP BY 列を指定していない場合。
  - 一時結果テーブルを作成する必要がない場合。
- COMMIT(*CHG) が指定され、ALWBLK(*ALLREAD) が指定されていない場合。
- カーソルに少なくとも 1 つの副照会が含まれており、最も外側の副選択により副照会の相関参照が提供されたか、または、最も外側の副選択が、副照会の述部演算子 (相互参照として扱われる) IN、= ANY、または < > ALL を伴う副照会を処理し、かつこの副照会が分離不能な場合。

次の場合には、SQL 実行時間は自動的にデータベース・マネージャーを使用して行をブロック化します。

- INSERT

  INSERT ステートメントに選択ステートメントが含まれている場合には、挿入される行はブロック化され、ブロックがいっぱいになるまでは目的のテーブルには挿入されません。ブロック化挿入の場合には、SQL 実行時間は自動的にブロック化を行います。

  注: VALUES 文節を伴う INSERT を指定した場合には、SQL 実行時間は、プログラムが終了するまでは、実際には、挿入を行うために使用する内部カーソルをクローズしません。したがって、同じ INSERT ステートメントが再度実行される場合は、完全なオープンは必要ないので、アプリケーションの実行速度がかなり向上します。

- OPEN

  次のすべての条件に該当する場合には、行が検索されるとき OPEN ステートメントに従ってブロック化が行われます。

  - カーソルが FETCH ステートメントだけに使用されている。
  - EXECUTE または EXECUTE IMMEDIATE ステートメントがプログラムにないか、あるいは ALWBLK(*ALLREAD) が指定されていたか、あるいはカーソルが FOR FETCH ONLY 文節を使用して宣言されている。

– COMMIT(*CHG) と ALWBLK(*ALLREAD) が指定されていて、 COMMIT(*CS) と ALWBLK(*ALLREAD) が指定されているか、あるいは COMMIT(*NONE) が指定されている。

関連資料

212 ページの『プリコンパイル・オプションの使用によるデータベース・パフォーマンスの向上』
パフォーマンスが向上するように SQL プログラムを作成するためのプリコンパイル・オプションがいくつか用意されています。これらのオプションを使用すると、アプリケーションの働きに影響することがあるので、これらはあくまでもオプションです。そのために、これらのパラメーターのデフォルト値は、以前のリリースからアプリケーションを正しくマイグレーションできるような値になっています。ただし、他のオプションを指定してもパフォーマンスを向上することができます。

関連情報

データベース・ファイルの一時変更 (OVRDBF) コマンド

SELECT ステートメントで選択される列数の最適化

SELECT ステートメントの選択リストに指定する列の数に応じて、データベース・マネージャーは基礎となるテーブルからデータを検索し、そのデータをアプリケーション・プログラムのホスト変数に対応づけます。指定する列の数をできる限り少なくすると、処理装置リソースの使用率を低下させることができます。

SELECT * をコーディングするのは便利な方法ですが、パフォーマンスの観点からは、アプリケーションで実際に必要とする列を明示的にコーディングする方がはるかに効率的です。このことが特に重要な意味をもつのは、索引専用アクセスが要求された場合、あるいはすべての列が分類操作の対象とされる場合 (SELECT DISTINCT および SELECT UNION の場合に生じる状況) です。

照会での列の数を最小にし、索引を使用してすべてのデータに対する要求を完全に満たす可能性を増すことができるため、索引専用アクセスについて考慮する場合にもこのことは重要です。

関連情報

選択ステートメント

SQL PREPARE ステートメントに伴う冗長妥当性検査の除去

SQL の PREPARE ステートメントが実行されたとき行われる処理は、プリコンパイル処理のときに行われる処理とほとんど同じです。

準備されるステートメントには、次の処理が行われます。

- 構文の検査。
- オブジェクトの使用が有効であることを確かめる妥当性検査。
- アクセス・プランの作成。

この場合も、ステートメントが実行またはオープンされると、データベース・マネージャーはアクセス・プランがまだ有効であるかどうかの妥当性検査を再度行います。このオープン処理時の妥当性検査の多くは、PREPARE の処理時に行われた妥当性検査と重複しています。 DLYPRP(*YES) パラメーターは、このプログラムの中の PREPARE ステートメントが動的ステートメントの妥当性を完全に検査するかどうかを指定します。妥当性検査は、動的ステートメントのオープンまたは実行時に行われます。このパラメーターを使用すると、重複する妥当性検査が省かれるので、PREPARE SQL ステートメントを使用するプログラムのパフォーマンスが大幅に向上します。このプリコンパイル・オプションを指定したプログラムでは、OPEN ステートメントまたは EXECUTE ステートメントが実行された後、 SQLCODE と SQLSTATE を調べてステートメントが有効であるかどうか確かめる必要があります。 PREPARE ステートメントで

INTO 文節が使用されている場合や、 OPEN がステートメントに対して出される前に DESCRIBE ステートメントが動的ステートメントを使用している場合は、 DLYPRP(*YES) を使用しても、パフォーマンスの向上は得られません。

関連資料

『プリコンパイル・オプションの使用によるデータベース・パフォーマンスの向上』
パフォーマンスが向上するように SQL プログラムを作成するためのプリコンパイル・オプションがいくつか用意されています。これらのオプションを使用すると、アプリケーションの働きに影響することがあるので、これらはあくまでもオプションです。そのために、これらのパラメーターのデフォルト値は、以前のリリースからアプリケーションを正しくマイグレーションできるような値になっています。ただし、他のオプションを指定してもパフォーマンスを向上することができます。

関連情報

PREPARE ステートメント

REFRESH(*FORWARD) により対話式に表示されるデータのページ送り

大きなテーブルでは、テーブルから最新のデータを直接かつ動的に検索する SQL の開始 (STRSQL) コマンドの REFRESH(*ALWAYS) パラメーターが原因で、通常、ページ送りのパフォーマンスが低下します。ページ送りのパフォーマンスは、REFRESH(*FORWARD) を指定して向上させることができます。

REFRESH(*FORWARD) を用いて対話式にデータを表示しているときに、ユーザーが画面のページ送りを行うと、選択ステートメントの結果が一時テーブルにコピーされます。テーブルを共用している他のユーザーは、このように選択ステートメントの結果が表示されている間にも、行に変更を加えることができます。この場合、行のページ戻しまたはページ送りによって、すでに表示された行を再表示すると、表示されるのは更新後のテーブル内の行ではなく、一時テーブルの行です。

再表示オプションは、「セッション・サービス」画面から変更することができます。

関連情報

SQL の開始 (STRSQL) コマンド

DB2 for i5/OS のパフォーマンスに関する一般的な情報

アプリケーションのコーディングでは、パフォーマンスの最適化に役立ついくつかの汎用的なヒントがあります。

長いオブジェクト名の使用によるデータベース・パフォーマンスの向上

長いオブジェクト名は、 SQL ステートメントで使用されているときに内部的にシステム・オブジェクト名に変換されます。この変換は、何らかの影響をパフォーマンスに与える場合があります。

長いオブジェクト名がライブラリー名で修飾されている場合、プリコンパイル時に短い名前への変換が起こります。この場合、ステートメントを実行しているときのパフォーマンスへの影響はありません。ただし、長いオブジェクト名が修飾されていないと、変換は実行時に行われ、わずかながらパフォーマンスに影響を及ぼします。

プリコンパイル・オプションの使用によるデータベース・パフォーマンスの向上

パフォーマンスが向上するように SQL プログラムを作成するためのプリコンパイル・オプションがいくつか用意されています。これらのオプションを使用すると、アプリケーションの働きに影響することがあるの

で、これらはあくまでもオプションです。そのために、これらのパラメーターのデフォルト値は、以前のリリースからアプリケーションを正しくマイグレーションできるような値になっています。ただし、他のオプションを指定してもパフォーマンスを向上することができます。

以下の表は、これらのプリコンパイル・オプションとそれぞれがパフォーマンスに及ぼす影響を示しています。

これらのオプションの一部は、ほとんどのアプリケーションで使用できます。 CRTDUPOBJ コマンドを使用すると、SQL CRTSQLxxx コマンドのコピーを作ることができ、 CHGCMDDFT コマンドを使用すると、プリコンパイル・パラメーターの最適値をカスタマイズすることができます。 DSPPGM、DSPSRVPGM、 DSPMOD、または PRTSQLINF の各コマンドを使用すると、既存のプログラム・オブジェクトで使用されているプリコンパイル・オプションを表示することができます。

プリコンパイル・オプション	最適値	改善点	考慮事項
ALWCPYDTA	*OPTIMIZE (デフォルト)	順序付け基準またはグループ化基準が選択基準と矛盾するような照会	照会をオープンするときデータのコピーを作ることができる。
ALWBLK	*ALLREAD (デフォルト)	追加の読み取り専用カーソルがブロック化を使用する。	ROLLBACK HOLD は読み取り専用カーソルの位置を変更できない。位置指定の更新または削除の動的処理は失敗するおそれがある。
CLOSQLCSR	*ENDJOB、*ENDSQL、または *ENDACTGRP	カーソル位置をプログラムの呼び出しから呼び出しの間保存できる。	プログラムの呼び出しが終了したとき SQL カーソルの暗黙のクローズが行われない。
DLYPRP	*YES	SQL の PREPARE ステートメントを使用するプログラムは実行速度が速くなる。	準備されたステートメント全体の妥当性検査は、ステートメントが実行またはオープンされるまで据え置かれる。
TGTRLS	*CURRENT (デフォルト)	プリコンパイラーは、現行リリースで強化されたパフォーマンスを利用するコードを生成できる。	プログラム・オブジェクトは旧リリースのシステムで使用できない。

関連資料

214 ページの『ALWCPYDTA パラメーターの使用によるデータベース・パフォーマンスの向上』
ある種の複雑な照会では、索引を使用または作成する代わりに分類またはハッシュ方式を使用して照会の実行を行うと、パフォーマンスを向上できる場合があります。

210 ページの『データベース・マネージャーのブロック化の制御』
パフォーマンスの向上を図るために、SQL 実行時間は、可能な限りデータベース・マネージャーから一度に 1 ブロックずつ行を取り出して挿入しようとします。

204 ページの『非 ILE プログラム呼び出しの場合のカーソル位置の保存』
非 ILE プログラム呼び出しの場合、SQL カーソル・クローズ (CLOSQLCSR) パラメーターを使用すると、以下の有効範囲を指定することができます。

211 ページの『SQL PREPARE ステートメントに伴う冗長妥当性検査の除去』
SQL の PREPARE ステートメントが実行されたとき行われる処理は、プリコンパイル処理のときに行われる処理とほとんど同じです。

ALWCPYDTA パラメーターの使用によるデータベース・パフォーマンスの向上

ある種の複雑な照会では、索引を使用または作成する代わりに分類またはハッシュ方式を使用して照会の実行を行うと、パフォーマンスを向上できる場合があります。

分類またはハッシュを使用すると、データベース・マネージャーは順序付けおよびグループ化処理から行選択を分離することができます。このパラメーターを使用すると、ビットマップ処理も部分的に制御できます。この分離により、選択に最も効率のよい索引を使用することが可能になります。一例として、次の SQL ステートメントを考えてください。

```
EXEC SQL
    DECLARE C1 CURSOR FOR
    SELECT EMPNO, LASTNAME, WORKDEPT
      FROM CORPDATA.EMPLOYEE
      WHERE WORKDEPT = 'A00'
     ORDER BY LASTNAME
END-EXEC.
```

OPNQRYF コマンドを使用すると、上記の SQL ステートメントを次のように書き込むことができます。

```
OPNQRYF FILE(CORPDATA/EMPLOYEE)
        FORMAT(FORMAT1)
        QRYSLT(WORKDEPT *EQ ''A00'')
        KEYFLD(LASTNAME)
```

上記の例で、ALWCPYDTA(*NO) または ALWCPYDTA(*YES) を指定すると、データベース・マネージャーは、そのような索引が存在すれば、LASTNAME という名前の列を持つ最初の索引から索引を作成することを試みます。この索引を使用してテーブル内の行が走査され、WHERE 条件に合致する行のみが選択されます。

ALWCPYDTA(*OPTIMIZE) の指定があるときは、データベース・マネージャーは WORKDEPT の最初の索引列を持つ索引を使用します。次にデータベース・マネージャーは WHERE 条件に合致するすべての行のコピーを作成します。最後に、データベース・マネージャーは、コピーされた行を LASTNAME の値によって分類します。この行選択の処理は、使用する索引によって選択する行が直ちに見つかるため、はるかに効率がよくなります。

ALWCPYDTA(*OPTIMIZE) を指定すると、照会の処理の合計時間が最適化されます。しかし、結果のテーブルの最初の行を返す前にデータのコピーを作成する必要があるため、最初の行が取り出される時間は増加することがあります。この応答時間の初期変更は、対話式画面を表示するアプリケーションまたは照会の最初の数行だけを取り出すアプリケーションの場合に重要になることがあります。 OPTIMIZE 文節を使用すると、分類を回避するように DB2 for i5/OS Query 最適化プログラムに影響を及ぼすことができます。

結合順序が ORDER BY 仕様に関係なく最適化されるために、結合操作が関係する照会の場合にも、ALWCPYDTA(*OPTIMIZE) の使用による利点が得られます。

関連概念

5 ページの『プラン・キャッシュ』
プラン・キャッシュは、SQE によって最適化された照会用のアクセス・プランを含むリポジトリーです。

関連資料

212 ページの『プリコンパイル・オプションの使用によるデータベース・パフォーマンスの向上』
パフォーマンスが向上するように SQL プログラムを作成するためのプリコンパイル・オプションがいくつか用意されています。これらのオプションを使用すると、アプリケーションの働きに影響することがあるので、これらはあくまでもオプションです。そのために、これらのパラメーターのデフォルト値は、以前のリリースからアプリケーションを正しくマイグレーションできるような値になっています。ただし、他のオプションを指定してもパフォーマンスを向上することができます。

11 ページの『基数索引走査』
基数索引走査操作は、テーブルから行をキー順に取り出すのに使用します。テーブル走査と同様に索引内の行すべてが順次処理されますが、結果の行番号はキー列に基づいて順序付けされます。

12 ページの『基数索引プローブ』
基数索引プローブ操作は、テーブルから行をキー順に取り出すのに使用します。基数索引プローブと基数索引走査の主な相違点は、取り出されている行をサブセット化するプローブ操作では、戻される行が最初に識別される必要があるという点です。

データベースにおける VARCHAR および VARGRAPHIC データ・タイプの使用上のヒント

可変長列 (VARCHAR または VARGRAPHIC) のサポートを使用すると、 1 つのテーブルに可変長の列としていくつでも列を定義することができます。 VARCHAR または VARGRAPHIC サポートを使用すると、通常、テーブルのサイズを小さくすることができます。

可変長列のデータは、内部では 2 つの区域に保管されます。すなわち、固定長または ALLOCATE 域とオーバーフロー域です。デフォルト値を指定すると、割り振られた長さは少なくとも値と同じ大きさになります。次の点を考慮すると、記憶域の最良の使い方を決定する際に役立ちます。

可変長データをもつテーブルを定義するときは、ALLOCATE 域の幅を決定する必要があります。その幅は、主要目的によって異なります。

- **スペースの節約:** ALLOCATE(0) を使用します。
- **パフォーマンス:** ALLOCATE 域の幅は、その列の値の少なくとも 90% から 95% を収容できるだけの大きさでなければなりません。

スペースの節約とパフォーマンスとのバランスをとることもできます。次の例は電子電話帳の例で、以下のデータを使用しています。

- 8600 人分の名前を姓、名、およびミドル・ネームで示したもの。
- Last、First、および Middle の各列は可変長です。
- 最も短い姓は 2 文字で構成され、最も長い姓は 22 文字で構成されます。

この例では、可変長の列を使用するとどの程度スペースを節約できるかを示しています。固定長の列で構成したテーブルは、使用するスペースが最も大きくなります。 ALLOCATE 域のサイズを慎重に計算してテーブルを作成すると、使用するディスク・スペースは少なくてすみます。また、ALLOCATE 域を使用せずに (すなわち、すべてのデータをオーバーフロー域に保管して) テーブルを定義すると、使用するディスク・スペースは最小になります。

サポートの種類	姓列の最大値/ ALLOCATE 域	名列の最大値/ ALLOCATE 域	ミドルネーム列の最大値/ ALLOCATE 域	物理ファイル全体のサイズ	オーバーフロー・スペースの行数
固定長	22	22	22	567 K	0
可変長	40/10	40/10	40/7	408 K	73

サポートの種類	姓列の最大値/ ALLOCATE 域	名列の最大値/ ALLOCATE 域	ミドルネーム列の最大値/ ALLOCATE 域	物理ファイル全体のサイズ	オーバーフロー・スペースの行数
可変長のデフォルト	40/0	40/0	40/0	373 K	8600

多くのアプリケーションでは、パフォーマンスを考慮しなければなりません。デフォルトの ALLOCATE(0) を使用すると、ディスク装置のトラフィックが 2 倍になります。 ALLOCATE(0) では、行の固定長部分の読み取りとオーバーフロー・スペースの読み取りをそれぞれ 1 回ずつ、合わせて 2 回の読み取りが必要です。 ALLOCATE を慎重に指定して可変長を使用すると、オーバーフローの発生と必要なスペースは最小になり、パフォーマンスは最高になります。テーブルのサイズは、固定長を使用した場合より 28% 小さくなっています。 1% 分の行はオーバーフロー域に入っているので、2 回の読み取りを要するアクセスは最低限になります。可変長を使用すると、固定長を使用した場合と同程度のパフォーマンスを発揮します。

ALLOCATE キーワードを使用してテーブルを作成する場合は、次のようにコーディングします。

```
CREATE TABLE PHONEDIR
        (LAST    VARCHAR(40) ALLOCATE(10),
         FIRST   VARCHAR(40) ALLOCATE(10),
         MIDDLE  VARCHAR(40) ALLOCATE(7))
```

ホスト変数を使用して可変長の列の挿入または更新を行う場合は、そのホスト変数は可変長にしなければなりません。ブランクは固定長ホスト変数から切り捨てられないので、固定長ホスト変数を使用した場合、あふれてオーバーフロー・スペースに入れられることになる行数が多くなる可能性があります。この場合、テーブルのサイズも大きくなります。

次の例では、固定長ホスト変数を使用してテーブルに 1 行挿入します。

```
01  LAST-NAME PIC X(40).
   ...
    MOVE "SMITH" TO LAST-NAME.
    EXEC SQL
      INSERT INTO PHONEDIR
       VALUES(:LAST-NAME, :FIRST-NAME, :MIDDLE-NAME, :PHONE)
    END-EXEC.
```

この場合、ホスト変数 LAST-NAME は、可変長ではありません。文字ストリング“SMITH”の後に 35 個のブランクが付いたものが、 VARCHAR 列 LAST に挿入されます。この値は、10 という割り振りサイズより大きくなっています。後続の 35 個のブランクのうち、30 個はオーバーフロー域に入ります。

次の例では、可変長ホスト変数を使用してテーブルに 1 行挿入します。

```
01  VLAST-NAME.
    49 LAST-NAME-LEN PIC S9(4) BINARY.
    49 LAST-NAME-DATA PIC X(40).
   ...
    MOVE "SMITH" TO LAST-NAME-DATA.
    MOVE 5 TO LAST-NAME-LEN.
    EXEC SQL
      INSERT INTO PHONEDIR
     VALUES(:VLAST-NAME, :VFIRST-NAME, :VMIDDLE-NAME, :PHONE)
    END-EXEC.
```

この場合のホスト変数 VLAST-NAME は可変長です。データの実際の長さは 5 にセットされます。この値は、割り振られた長さより小さくなっています。したがって、この値は列の固定長部分に入れることができます。

可変長の列を含むテーブルに対して 物理ファイル・メンバーの再編成 (RGZPFM) コマンドを実行すると、パフォーマンスを向上させることができます。オーバーフロー域の使用されていない部分の断片は、物理ファイル・メンバーの再編成 (RGZPFM) コマンドによって圧縮させることができます。このため、オーバーフローした行の読み取り時間は短縮され、参照対象の位置の限定がさらに図られ、逐次バッチ処理のために最適の順序が得られます。

可変長列については、適切な最大長を指定してください。指定した長さが長すぎると、処理アクセス・グループ (PAG) が増大します。 PAG が大きくなると、パフォーマンスが低下します。また、最大長が大きいと、SEQONLY(*YES) を指定しても効率が悪くなります。可変長の列が 2000 バイトを超えると、その列はキー列としては使用できません。

同一テーブルでの LOB および VARCHAR の使用

VARCHAR 列と同一の方式で割り振られた LOB 列を保管します。オーバーフロー記憶域に格納された列を参照している場合、現行の領域にある列すべてがメモリーにページインされます。オーバーフロー域のより小さな VARCHAR 列は、必要以上に LOB 列のページングを強制する可能性があります。たとえば、アプリケーションで取得された VARCHAR(256) 列には、同じ行にある 2 つの 5MB BLOB 列がページインされるという副次作用があります。これを回避するため、ALLOCATE キーワードを使用して、オーバーフロー域に格納する列を LOB 列に限定することが可能です。

関連情報

物理ファイル・メンバーの再編成 (RGZPFM) コマンド

物理ファイルの再編成

組み込み SQL プログラミング

データベース・モニター: 形式

このセクションでは、データベース・モニター SQL テーブルおよびビューを作成するための形式について説明します。

データベース・モニター SQL テーブル形式

システムとともに出荷される QSYS/QAQQDBMN パフォーマンス統計テーブルを作成するために使用される形式を表示します。

```
 CREATE TABLE QSYS.QAQQDBMN (
QQRID DECIMAL(15, 0) NOT NULL DEFAULT 0 ,
QQTIME TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ,
QQJFLD CHAR(46) CCSID 65535 NOT NULL DEFAULT '' ,
QQRDBN CHAR(18) CCSID 37 NOT NULL DEFAULT '' ,
QQSYS CHAR(8) CCSID 37 NOT NULL DEFAULT '' ,
QQJOB CHAR(10) CCSID 37 NOT NULL DEFAULT '' ,
QQUSER CHAR(10) CCSID 37 NOT NULL DEFAULT '' ,
QQJNUM CHAR(6) CCSID 37 NOT NULL DEFAULT '' ,
QQUCNT DECIMAL(15, 0) DEFAULT NULL ,
QQUDEF VARCHAR(100) CCSID 37 DEFAULT NULL ,
QQSTN DECIMAL(15, 0) DEFAULT NULL ,
QQQDTN DECIMAL(15, 0) DEFAULT NULL ,
QQQDTL DECIMAL(15, 0) DEFAULT NULL ,
QQMATN DECIMAL(15, 0) DEFAULT NULL ,
QQMATL DECIMAL(15, 0) DEFAULT NULL ,
QQTLN CHAR(10) CCSID 37 DEFAULT NULL ,
QQTFN CHAR(10) CCSID 37 DEFAULT NULL ,
QQTMN CHAR(10) CCSID 37 DEFAULT NULL ,
QQPTLN CHAR(10) CCSID 37 DEFAULT NULL ,
QQPTFN CHAR(10) CCSID 37 DEFAULT NULL ,
QQPTMN CHAR(10) CCSID 37 DEFAULT NULL ,
```

```
QQILNM CHAR(10) CCSID 37 DEFAULT NULL ,
QQIFNM CHAR(10) CCSID 37 DEFAULT NULL ,
QQIMNM CHAR(10) CCSID 37 DEFAULT NULL ,
QQNTNM CHAR(10) CCSID 37 DEFAULT NULL ,
QQNLNM CHAR(10) CCSID 37 DEFAULT NULL ,
QQSTIM TIMESTAMP DEFAULT NULL ,
QQETIM TIMESTAMP DEFAULT NULL ,
QQKP CHAR(1) CCSID 37 DEFAULT NULL ,
QQKS CHAR(1) CCSID 37 DEFAULT NULL ,
QQTOTR DECIMAL(15, 0) DEFAULT NULL ,
QQTMPR DECIMAL(15, 0) DEFAULT NULL ,
QQJNP DECIMAL(15, 0) DEFAULT NULL ,
QQEPT DECIMAL(15, 0) DEFAULT NULL ,
QQDSS CHAR(1) CCSID 37 DEFAULT NULL ,
QQIDXA CHAR(1) CCSID 37 DEFAULT NULL ,
QQORDG CHAR(1) CCSID 37 DEFAULT NULL ,
QQGRPG CHAR(1) CCSID 37 DEFAULT NULL ,
QQJNG CHAR(1) CCSID 37 DEFAULT NULL ,
QQUNIN CHAR(1) CCSID 37 DEFAULT NULL ,
QQSUBQ CHAR(1) CCSID 37 DEFAULT NULL ,
QQHSTV CHAR(1) CCSID 37 DEFAULT NULL ,
QQRCDS CHAR(1) CCSID 37 DEFAULT NULL ,
QQRCOD CHAR(2) CCSID 37 DEFAULT NULL ,
QQRSS DECIMAL(15, 0) DEFAULT NULL ,
QQREST DECIMAL(15, 0) DEFAULT NULL ,
QQRIDX DECIMAL(15, 0) DEFAULT NULL ,
QQFKEY DECIMAL(15, 0) DEFAULT NULL ,
QQKSEL DECIMAL(15, 0) DEFAULT NULL ,
QQAJN DECIMAL(15, 0) DEFAULT NULL ,
QQIDXD VARCHAR(1000) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QQC11 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC12 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC13 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC14 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC15 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC16 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC18 CHAR(1) CCSID 37 DEFAULT NULL ,
QQC21 CHAR(2) CCSID 37 DEFAULT NULL ,
QQC22 CHAR(2) CCSID 37 DEFAULT NULL ,
QQC23 CHAR(2) CCSID 37 DEFAULT NULL ,
QQI1 DECIMAL(15, 0) DEFAULT NULL ,
QQI2 DECIMAL(15, 0) DEFAULT NULL ,
QQI3 DECIMAL(15, 0) DEFAULT NULL ,
QQI4 DECIMAL(15, 0) DEFAULT NULL ,
QQI5 DECIMAL(15, 0) DEFAULT NULL ,
QQI6 DECIMAL(15, 0) DEFAULT NULL ,
QQI7 DECIMAL(15, 0) DEFAULT NULL ,
QQI8 DECIMAL(15, 0) DEFAULT NULL ,
QQI9 DECIMAL(15, 0) DEFAULT NULL ,
QQIA DECIMAL(15, 0) DEFAULT NULL ,
QQF1 DECIMAL(15, 0) DEFAULT NULL ,
QQF2 DECIMAL(15, 0) DEFAULT NULL ,
QQF3 DECIMAL(15, 0) DEFAULT NULL ,
QQC61 CHAR(6) CCSID 37 DEFAULT NULL ,
QQC81 CHAR(8) CCSID 37 DEFAULT NULL ,
QQC82 CHAR(8) CCSID 37 DEFAULT NULL ,
QQC83 CHAR(8) CCSID 37 DEFAULT NULL ,
QQC84 CHAR(8) CCSID 37 DEFAULT NULL ,
QQC101 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC102 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC103 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC104 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC105 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC106 CHAR(10) CCSID 37 DEFAULT NULL ,
QQC181 VARCHAR(128) ALLOCATE(18) CCSID 37 DEFAULT NULL ,
QQC182 VARCHAR(128) ALLOCATE(18) CCSID 37 DEFAULT NULL ,
QQC183 VARCHAR(128) ALLOCATE(15) CCSID 37 DEFAULT NULL ,
```

```
QQC301 VARCHAR(30) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QQC302 VARCHAR(30) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QQC303 VARCHAR(30) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QQ1000 VARCHAR(1000) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QQTIM1 TIMESTAMP DEFAULT NULL ,
QQTIM2 TIMESTAMP DEFAULT NULL ,
QVQTBL VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVQLIB VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVPTBL VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVPLIB VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVINAM VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVILIB VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVQTBLI CHAR(1) CCSID 37 DEFAULT NULL ,
QVPTBLI CHAR(1) CCSID 37 DEFAULT NULL ,
QVINAMI CHAR(1) CCSID 37 DEFAULT NULL ,
QVBNDY CHAR(1) CCSID 37 DEFAULT NULL ,
QVJFANO CHAR(1) CCSID 37 DEFAULT NULL ,
QVPARPF CHAR(1) CCSID 37 DEFAULT NULL ,
QVPARPL CHAR(1) CCSID 37 DEFAULT NULL ,
QVC11 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC12 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC13 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC14 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC15 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC16 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC17 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC18 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC19 CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1A CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1B CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1C CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1D CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1E CHAR(1) CCSID 37 DEFAULT NULL ,
QVC1F CHAR(1) CCSID 37 DEFAULT NULL ,
QWC11 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC12 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC13 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC14 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC15 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC16 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC17 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC18 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC19 CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1A CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1B CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1C CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1D CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1E CHAR(1) CCSID 37 DEFAULT NULL ,
QWC1F CHAR(1) CCSID 37 DEFAULT NULL ,
QVC21 CHAR(2) CCSID 37 DEFAULT NULL ,
QVC22 CHAR(2) CCSID 37 DEFAULT NULL ,
QVC23 CHAR(2) CCSID 37 DEFAULT NULL ,
QVC24 CHAR(2) CCSID 37 DEFAULT NULL ,
QVCTIM DECIMAL(15, 0) DEFAULT NULL ,
QVPARD DECIMAL(15, 0) DEFAULT NULL ,
QVPARU DECIMAL(15, 0) DEFAULT NULL ,
QVPARRC DECIMAL(15, 0) DEFAULT NULL ,
QVRCNT DECIMAL(15, 0) DEFAULT NULL ,
QVFILES DECIMAL(15, 0) DEFAULT NULL ,
QVP151 DECIMAL(15, 0) DEFAULT NULL ,
QVP152 DECIMAL(15, 0) DEFAULT NULL ,
QVP153 DECIMAL(15, 0) DEFAULT NULL ,
QVP154 DECIMAL(15, 0) DEFAULT NULL ,
QVP155 DECIMAL(15, 0) DEFAULT NULL ,
QVP156 DECIMAL(15, 0) DEFAULT NULL ,
QVP157 DECIMAL(15, 0) DEFAULT NULL ,
QVP158 DECIMAL(15, 0) DEFAULT NULL ,
```

```
QVP159 DECIMAL(15, 0) DEFAULT NULL ,
QVP15A DECIMAL(15, 0) DEFAULT NULL ,
QVP15B DECIMAL(15, 0) DEFAULT NULL ,
QVP15C DECIMAL(15, 0) DEFAULT NULL ,
QVP15D DECIMAL(15, 0) DEFAULT NULL ,
QVP15E DECIMAL(15, 0) DEFAULT NULL ,
QVP15F DECIMAL(15, 0) DEFAULT NULL ,
QVC41 CHAR(4) CCSID 37 DEFAULT NULL ,
QVC42 CHAR(4) CCSID 37 DEFAULT NULL ,
QVC43 CHAR(4) CCSID 37 DEFAULT NULL ,
QVC44 CHAR(4) CCSID 37 DEFAULT NULL ,
QVC81 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC82 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC83 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC84 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC85 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC86 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC87 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC88 CHAR(8) CCSID 37 DEFAULT NULL ,
QVC101 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC102 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC103 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC104 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC105 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC106 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC107 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC108 CHAR(10) CCSID 37 DEFAULT NULL ,
QVC1281 VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVC1282 VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVC1283 VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVC1284 VARCHAR(128) ALLOCATE(10) CCSID 37 DEFAULT NULL ,
QVC3001 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3002 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3003 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3004 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3005 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3006 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3007 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC3008 VARCHAR(300) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC5001 VARCHAR(500) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC5002 VARCHAR(500) ALLOCATE(32) CCSID 37 DEFAULT NULL ,
QVC1000 VARCHAR(1000) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QWC1000 VARCHAR(1000) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QQINT01 INTEGER DEFAULT NULL ,
QQINT02 INTEGER DEFAULT NULL ,
QQINT03 INTEGER DEFAULT NULL ,
QQINT04 INTEGER DEFAULT NULL ,
QQSMINT1 SMALLINT DEFAULT NULL ,
QQSMINT2 SMALLINT DEFAULT NULL ,
QQSMINT3 SMALLINT DEFAULT NULL ,
QQSMINT4 SMALLINT DEFAULT NULL ,
QQSMINT5 SMALLINT DEFAULT NULL ,
QQSMINT6 SMALLINT DEFAULT NULL ,
QQ1000L CLOB(2147483647) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QFC11 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC12 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC13 CHAR(1) CCSID 37 DEFAULT NULL ,
QQCLOB2 CLOB(2147483647) ALLOCATE(48) CCSID 37 DEFAULT NULL ,
QFC14 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC15 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC16 CHAR(1) CCSID 37 DEFAULT NULL ,
QQCLOB3 CLOB(2147483647) CCSID 37 DEFAULT NULL ,
QFC17 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC18 CHAR(1) CCSID 37 DEFAULT NULL ,
QFC19 CHAR(1) CCSID 37 DEFAULT NULL ,
QQDBCLOB1 DBCLOB(1073741823) ALLOCATE(24) CCSID 1200 DEFAULT NULL ,
QFC1A CHAR(1) CCSID 37 DEFAULT NULL ,
```

```
 QFC1B CHAR(1) CCSID 37 DEFAULT NULL ,
 QFC1C CHAR(1) CCSID 37 DEFAULT NULL ,
 QQDBCLOB2 DBCLOB(1073741823) CCSID 1200 DEFAULT NULL ,
 QFC1D CHAR(1) CCSID 37 DEFAULT NULL ,
 QFC1E CHAR(1) CCSID 37 DEFAULT NULL ,
 QFC1F CHAR(1) CCSID 37 DEFAULT NULL ,
 QQBLOB1 BLOB(2147483647) DEFAULT NULL ,
 QXC11 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC12 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC13 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC14 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC15 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC16 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC17 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC18 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC19 CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC1A CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC1B CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC1C CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC1D CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC1E CHAR(1) CCSID 37 DEFAULT NULL ,
 QXC21 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC22 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC23 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC24 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC25 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC26 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC27 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC28 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC29 CHAR(2) CCSID 37 DEFAULT NULL ,
 QXC41 CHAR(4) CCSID 37 DEFAULT NULL ,
 QXC42 CHAR(4) CCSID 37 DEFAULT NULL ,
 QXC43 CHAR(4) CCSID 65535 DEFAULT NULL ,
 QXC44 CHAR(4) CCSID 37 DEFAULT NULL ,
 QQINT05 INTEGER DEFAULT NULL ,
 QQINT06 INTEGER DEFAULT NULL ,
 QQINT07 INTEGER DEFAULT NULL ,
 QQINT08 INTEGER DEFAULT NULL ,
 QQINT09 INTEGER DEFAULT NULL ,
 QQINT0A INTEGER DEFAULT NULL ,
 QQINT0B INTEGER DEFAULT NULL ,
 QQINT0C INTEGER DEFAULT NULL ,
 QQINT0D INTEGER DEFAULT NULL ,
 QQINT0E INTEGER DEFAULT NULL ,
 QQINT0F INTEGER DEFAULT NULL ,
 QQSMINT7 SMALLINT DEFAULT NULL ,
 QQSMINT8 SMALLINT DEFAULT NULL ,
 QQSMINT9 SMALLINT DEFAULT NULL ,
 QQSMINTA SMALLINT DEFAULT NULL ,
 QQSMINTB SMALLINT DEFAULT NULL ,
 QQSMINTC SMALLINT DEFAULT NULL ,
 QQSMINTD SMALLINT DEFAULT NULL ,
 QQSMINTE SMALLINT DEFAULT NULL ,
 QQSMINTF SMALLINT DEFAULT NULL )

 RCDFMT QQQDBMN    ;
  RENAME QSYS/QQQDBMN TO SYSTEM NAME QAQQDBMN;

  LABEL ON TABLE QSYS/QAQQDBMN
 IS 'Database Monitor Physical File' ;

  LABEL ON COLUMN QSYS.QAQQDBMN
(QQRID   IS 'Record              ID' ,
 QQTIME  IS 'Created             Time' ,
 QQJFLD  IS 'Join                Column' ,
 QQRDBN  IS 'Relational          Database            Name' ,
 QQSYS   IS 'System              Name' ,
```

```
QQJOB   IS 'Job                Name' ,
QQUSER  IS 'Job                User' ,
QQJNUM  IS 'Job                Number' ,
QQUCNT  IS 'Unique             Counter' ,
QQUDEF  IS 'User               Defined            Column' ,
QQSTN   IS 'Statement          Number' ,
QQQDTN  IS 'Subselect          Number' ,
QQQDTL  IS 'Subselect          Nested             Level' ,
QQMATN  IS 'Subselect          Number of          Materialized View' ,
QQMATL  IS 'Subselect          Level of           Materialized View' ,
QQTLN   IS 'Library of         Table              Queried' ,
QQTFN   IS 'Name of            Table              Queried' ,
QQTMN   IS 'Member of          Table              Queried' ,
QQPTLN  IS 'Library of         Base               Table' ,
QQPTFN  IS 'Name of            Base               Table' ,
QQPTMN  IS 'Member of          Base               Table' ,
QQILNM  IS 'Library of         Index              Used' ,
QQIFNM  IS 'Name of            Index              Used' ,
QQIMNM  IS 'Member of          Index              Used' ,
QQNTNM  IS 'NLSS               Table' ,
QQNLNM  IS 'NLSS               Library' ,
QQSTIM  IS 'Start              Time' ,
QQETIM  IS 'End                Time' ,
QQKP     IS 'Key              Time' ,
QQETIM  IS 'End                Time' ,
QQKP     IS 'Key               Positioning' ,
QQKS    IS 'Key                Selection' ,
QQTOTR  IS 'Total              Rows' ,
QQTMPR  IS 'Number             of Rows            in Temporary' ,
QQJNP   IS 'Join               Position' ,
QQEPT   IS 'Estimated          Processing         Time' ,
QQDSS   IS 'Data               Space              Selection' ,
QQIDXA  IS 'Index              Advised' ,
QQORDG  IS 'Ordering' ,
QQGRPG  IS 'Grouping' ,
QQJNG   IS 'Join' ,
QQUNIN  IS 'Union' ,
QQSUBQ  IS 'Subquery' ,
QQHSTV  IS 'Host               Variables' ,
QQRCDS  IS 'Row                Selection' ,
QQRCOD  IS 'Reason             Code' ,
QQRSS   IS 'Number             of Rows            Selected' ,
QQREST  IS 'Estimated          Number of          Rows Selected' ,
QQRIDX  IS 'Number of          Entries in         Index Created' ,
QQFKEY  IS 'Estimated          Entries for        Key Positioning' ,
QQKSEL  IS 'Estimated          Entries for        Key Selection' ,
QQAJN   IS 'Estimated          Number of          Joined Rows' ,
QQIDXD  IS 'Advised            Key                Columns' ,
QQI9    IS 'Thread             Identifier' ,
QVQTBL  IS 'Queried            Table              Long Name' ,
QVQLIB  IS 'Queried            Library            Long Name' ,
QVPTBL  IS 'Base               Table              Long Name' ,
QVPLIB  IS 'Base               Library            Long Name' ,
QVINAM  IS 'Index Used         Long Name' ,
QVILIB  IS 'Index Used         Library            Name' ,
QVQTBLI IS 'Table              Long               Required' ,
QVPTBLI IS 'Base               Long               Required' ,
QVINAMI IS 'Index              Long               Required' ,
QVBNDY  IS 'I/O or CPU         Bound' ,
QVJFANO IS 'Join               Fan                Out' ,
QVPARPF IS 'Parallel           Pre-Fetch' ,
QVPARPL IS 'Parallel           Pre-Load' ,
QVCTIM  IS 'Estimated          Cumulative         Time' ,
QVPARD  IS 'Parallel           Degree             Requested' ,
QVPARU  IS 'Parallel           Degree             Used' ,
QVPARRC IS 'Parallel           Limited            Reason Code' ,
QVRCNT  IS 'Refresh            Count' ,
```

```
 QVFILES IS 'Number of           Tables              Joined' ) ;

LABEL ON COLUMN QSYS.QAQQDBMN
(QQRID   TEXT IS 'Record ID' ,
 QQTIME  TEXT IS 'Time record was created' ,
 QQJFLD  TEXT IS 'Join Column' ,
 QQRDBN  TEXT IS 'Relational Database Name' ,
 QQSYS   TEXT IS 'System Name' ,
 QQJOB   TEXT IS 'Job Name' ,
 QQUSER  TEXT IS 'Job User' ,
 QQJNUM  TEXT IS 'Job Number' ,
 QQUCNT  TEXT IS 'Unique Counter' ,
 QQUDEF  TEXT IS 'User Defined Column' ,
 QQSTN   TEXT IS 'Statement Number' ,
 QQQDTN  TEXT IS 'Subselect Number' ,
 QQQDTL  TEXT IS 'Subselect Nested Level' ,
 QQMATN  TEXT IS 'Subselect Number of Materialized View' ,
 QQMATL  TEXT IS 'Subselect Level of Materialized View' ,
 QQTLN   TEXT IS 'Library of Table Queried' ,
 QQTFN   TEXT IS 'Name of Table Queried' ,
 QQTMN   TEXT IS 'Member of Table Queried' ,
 QQPTLN  TEXT IS 'Base Table Library' ,
 QQPTFN  TEXT IS 'Base Table' ,
 QQPTMN  TEXT IS 'Base Table Member' ,
 QQILNM  TEXT IS 'Library of Index Used' ,
 QQIFNM  TEXT IS 'Name of Index Used' ,
 QQIMNM  TEXT IS 'Member of Index Used' ,
 QQNTNM  TEXT IS 'NLSS Table' ,
 QQNLNM  TEXT IS 'NLSS Library' ,
 QQSTIM  TEXT IS 'Start timestamp' ,
 QQETIM  TEXT IS 'End timestamp' ,
 QQKP    TEXT IS 'Key positioning' ,
 QQKS    TEXT IS 'Key selection' ,
 QQTOTR  TEXT IS 'Total row in table' ,
 QQTMPR  TEXT IS 'Number of rows in temporary' ,
 QQJNP   TEXT IS 'Join Position' ,
 QQEPT   TEXT IS 'Estimated processing time' ,
 QQDSS   TEXT IS 'Data Space Selection' ,
 QQIDXA  TEXT IS 'Index advised' ,
 QQORDG  TEXT IS 'Ordering' ,
 QQGRPG  TEXT IS 'Grouping' ,
 QQJNG   TEXT IS 'Join' ,
 QQUNIN  TEXT IS 'Union' ,
 QQSUBQ  TEXT IS 'Subquery' ,
 QQHSTV  TEXT IS 'Host Variables' ,
 QQRCDS  TEXT IS 'Row Selection' ,
 QQRCOD  TEXT IS 'Reason Code' ,
 QQRSS   TEXT IS 'Number of rows selected or sorted' ,
 QQREST  TEXT IS 'Estimated number of rows selected' ,
 QQRIDX  TEXT IS 'Number of entries in index created' ,
 QQFKEY  TEXT IS 'Estimated keys for key positioning' ,
 QQKSEL  TEXT IS 'Estimated keys for key selection' ,
 QQAJN   TEXT IS 'Estimated number of joined rows' ,
 QQIDXD  TEXT IS 'Key columns for the index advised' ,
 QQI9    TEXT IS 'Thread Identifier' ,
 QVQTBL  TEXT IS 'Queried Table, Long Name' ,
 QVQLIB  TEXT IS 'Queried Library, Long Name' ,
 QVPTBL  TEXT IS 'Base Table, Long Name' ,
 QVPLIB  TEXT IS 'Base Library, Long Name' ,
 QVINAM  TEXT IS 'Index Used, Long Name' ,
 QVILIB  TEXT IS 'Index Used, Libary Name' ,
 QVQTBLI TEXT IS 'Table Long Required' ,
 QVPTBLI TEXT IS 'Base Long Required' ,
 QVINAMI TEXT IS 'Index Long Required' ,
 QVBNDY  TEXT IS 'I/O or CPU Bound' ,
 QVJFANO TEXT IS 'Join Fan out' ,
 QVPARPF TEXT IS 'Parallel Pre-Fetch' ,
```

```
QVPARPL TEXT IS 'Parallel Pre-Load' ,
QVCTIM  TEXT IS 'Cumulative Time' ,
QVPARD  TEXT IS 'Parallel Degree, Requested' ,
QVPARU  TEXT IS 'Parallel Degree, Used' ,
QVPARRC TEXT IS 'Parallel Limited, Reason Code' ,
QVRCNT  TEXT IS 'Refresh Count' ,
QVFILES TEXT IS 'Number of, Tables Joined' ) ;
```

オプションのデータベース・モニター SQL ビュー形式

次の例は、示された SQL で作成できるさまざまなオプションの SQL ビュー形式を示しています。列記述の説明は、それぞれの例の後にあります。これらのビューはシステムに付属していないため、独自に作成する必要があります。これらのビューはオプションであり、モニター・データの分析には必須ではありません。

行識別番号 (QQRID) 5000 以上の行はすべて、内部データベースの使用のためのものです。

データベース・モニター・ビュー 1000 - SQL 情報

データベース・モニター QQQ1000 の SQL 論理ビュー形式を表示します。

```
| Create View QQQ1000 as
|   (SELECT QQRID as Row_ID,
|           QQTIME as Time_Created,
|           QQJFLD as Join_Column,
|           QQRDBN as Relational_Database_Name,
|           QQSYS as System_Name,
|           QQJOB as Job_Name,
|           QQUSER as Job_User,
|           QQJNUM as Job_Number,
|           QQI9 as Thread_ID,
|           QQUCNT as Unique_Count,
|           QQI5 as Unique_Refresh_Counter,
|           QQUDEF as User_Defined,
|           QQSTN as Statement_Number,
|           QQC11 as Statement_Function,
|           QQC21 as Statement_Operation,
|           QQC12 as Statement_Type,
|           QQC13 as Parse_Required,
|           QQC103 as Package_Name,
|           QQC104 as Package_Library,
|           QQC181 as Cursor_Name,
|           QQC182 as Statement_Name,
|           QQSTIM as Start_Timestamp,
|           QQ1000 as Statement_Text,
|           QQC14 as Statement_Outcome,
|           QQI2 as Result_Rows,
|           QQC22 as Dynamic_Replan_Reason_Code,
|           QQC16 as Data_Conversion_Reason_Code,
|           QQI4 as Total_Time_Milliseconds,
|           QQI3 as Rows_Fetched,
|           QQETIM as End_Timestamp,
|           QQI6 as Total_Time_Microseconds,
|           QQI7 as SQL_Statement_Length,
|           QQI1 as Insert_Unique_Count,
|           QQI8 as SQLCode,
|           QQC81 as SQLState,
|           QVC101 as Close_Cursor_Mode,
|           QVC11 as Allow_Copy_Data_Value,
|           QVC12 as PseudoOpen,
|           QVC13 as PseudoClose,
|           QVC14 as ODP_Implementation,
|           QVC21 as Dynamic_Replan_SubCode,
|           QVC41 as Commitment_Control_Level,
|           QWC1B as Skip_Locked_Data,
```

```
|            QVC15 as Blocking_Type,
|            QVC16 as Delay_Prepare,
|            QVC1C as Explainable,
|            QVC17 as Naming_Convention,
|            QVC18 as Dynamic_Processing_Type,
|            QVC19 as LOB_Data_Optimized,
|            QVC1A as Program_User_Profile_Used,
|            QVC1B as Dynamic_User_Profile_Used,
|            QVC1281 as Default_Collection,
|            QVC1282 as Procedure_Name,
|            QVC1283 as Procedure_Library,
|            QQCLOB2 as SQL_Path,
|            QVC1284 as Current_Schema,
|            QQC18 as Binding_Type,
|            QQC61 as Cursor_Type,
|            QVC1D as Statement_Originator,
|            QQC15 as Hard_Close_Reason_Code,
|            QQC23 as Hard_Close_Subcode,
|            QVC42 as Date_Format,
|            QWC11 as Date_Separator,
|            QVC43 as Time_Format,
|            QWC12 as Time_Separator,
|            QWC13 as Decimal_Point,
|            QVC104 as Sort_Sequence_Table,
|            QVC105 as Sort_Sequence_Library,
|            QVC44 as Language_ID,
|            QVC23 as Country_ID,
|            QQIA as First_N_Rows_Value,
|            QQF1 as Optimize_For_N_Rows_Value,
|            QVC22 as SQL_Access_Plan_Reason_Code,
|            QVC24 as Access_Plan_Not_Saved_Reason_Code,
|            QVC81 as Transaction_Context_ID,
|            QVP152 as Activation_Group_Mark,
|            QVP153 as Open_Cursor_Threshold,
|            QVP154 as Open_Cursor_Close_Count,
|            QVP155 as Commitment_Control_Lock_Limit,
|            QWC15 as Allow_SQL_Mixed_Constants,
|            QWC16 as Suppress_SQL_Warnings,
|            QWC17 as Translate_ASCII,
|            QWC18 as System_Wide_Statement_Cache,
|            QVP159 as LOB_Locator_Threshold,
|            QVP156 as Max_Decimal_Precision,
|            QVP157 as Max_Decimal_Scale,
|            QVP158 as Min_Decimal_Divide_Scale ,
|            QWC19 as Unicode_Normalization,
|            QQ1000L as Statement_Text_Long,
|            QVP15B as Old_Access_Plan_Length,
|            QVP15C as New_Access_Plan_Length,
|            QVP151 as Fast_Delete_Count,
|            QQF2 as Statement_Max_Compression,
|            QVC102 as Current_User_Profile,
|            QVC1E as Expression_Evaluator_Used,
|            QVP15A as Host_Server_Delta,
|            QQC301 as NTS_Lock_Space_Id,
|            QQC183 as IP_Address,
|            QFC11 as IP_Type,
|            QQSMINT2 as IP_Port_Number,
|            QVC3004 as NTS_Transaction_Id,
|            QQSMINT3 as NTS_Format_Id_Length,
|            QQSMINT4 as NTS_Transatction_ID_SubLength,
|            QVRCNT as Unique_Refresh_Counter2,
|            QVP15F as Times_Run,
|            QVP15E as FullOpens,
|            QVC1F as Proc_In_Cache,
|            QWC1A as Combined_Operation,
|            QVC3001 as Client_Applname,
|            QVC3002 as Client_Userid,
```

```
|            QVC3003 as Client_Wrkstnname,
|            QVC3005 as Client_Acctng,
|            QVC3006 as Client_Progamid,
|            QVC5001 as Interface_Information,
|            QVC82 as Open_Options,
|            QWC1D as Extended_Indicators,
|            QWC1C as DECFLOAT_Rounding_Mode,
|            QWC1E as SQL_DECFLOAT_Warnings,
|            QVP15D as Worst_Time_Micro,
|            QQINT05 as SQUNQCT
|     FROM   DbMonLib/DbMonTable
|     WHERE  QQRID=1000)
```

表 44. QQQ1000 - SQL 情報

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
Unique_Refresh_Counter	QQI5	固有最新表示カウンター
User_Defined	QQUDEF	ユーザー定義列
Statement_Number	QQSTN	ステートメント番号 (ステートメントごとに固有)
Statement_Function	QQC11	ステートメント関数 • S - 選択 • U - 更新 • I - 挿入 • D - 削除 • L - データ定義言語 • O - その他

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Statement_Operation	QQC21	ステートメント操作 • AD - 記述子の割り振り • AF - 変更関数 • AL - テーブル更新 • AP - プロシージャーの変更 • AQ - 順序の変更 • CA - 呼び出し • CC - コレクションの作成 • CD - タイプの作成 • CF - 関数の作成 • CG - トリガーの作成 • CI - 索引の作成 • CL - クローズ • CM - コミット • CN - 接続 • CO - 注記 • CP - プロシージャーの作成 • CQ - 順序の作成 • CS - 別名/同義語の作成 • CT - テーブルの作成 • CV - ビューの作成 • DA - 記述子の割り振り解除 • DE - 記述 • DI - 切断 • DL - 削除 • DM - パラメーター・マーカーの記述 • DP - プロシージャーの宣言 • DR - 除去 • DT - 記述テーブル • EI - 即時実行 • EX - 実行 • FE - 取り出し • FL - 空きロケーター • GR - 権限付与 • GS - 記述子の取得 • HC - ハード・クローズ • HL - ロケーターの保持

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Statement_Operation (続き)	QQC21	• IN - 挿入 • JR - 再使用されるサーバー・ジョブ • LK - ロック • LO - ラベル付け • MT - テキストをさらに表示 (V5R4 では使用すべきでない) • OP - オープン • PD - 作成および記述 • PR - 準備 • QF - OPNQRYF コマンド • QM - Query/400 STRQMQRY コマンド • QQ - QQQQRY() API • QR - RUNQRY コマンド • RB - 保管ポイントへのロールバック • RE - 解放 • RF - テーブルの最新表示 • RG - 再シグナル • RO - ロールバック • RS - 保管ポイントの解放 • RT - テーブルの名前変更 • RV - 取り消し • SA - 保管ポイント • SC - 接続の設定 • SD - 記述子の設定 • SE - 暗号化パスワードの設定 • SN - セッション・ユーザーの設定 • SI - 選択 • SO - CURRENT DEGREE (現行影響度) の設定 • SP - パスの設定 • SR - 結果セットの設定 • SS - 現行スキーマの設定 • ST - トランザクションの設定 • SV - 変数の設定 • UP - 更新 • VI - 値 • X0 - 不明ステートメント • X1 - 不明ステートメント • X2 - DRDA (AS) 不明ステートメント • X3 - 不明ステートメント • X9 - 内部エラー • XA - X/Open API • ZD - ホスト・サーバー専用アクティビティー

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Statement_Type	QQC12	ステートメント・タイプ • D - 動的ステートメント • S - 静的ステートメント
Parse_Required	QQC13	構文解析が必須 (Y/N)
Package_Name	QQC103	パッケージの名前または現行 SQL ステートメントが入っているプログラムの名前
Package_Library	QQC104	パッケージが入っているライブラリーの名前
Cursor_Name	QQC181	該当する場合、この SQL ステートメントに対応するカーソルの名前
Statement_Name	QQC182	該当する場合、SQL ステートメントのステートメント名
Start_Timestamp	QQSTIM	このステートメントに入った時刻
Statement_Text	QQ1000	ステートメント・テキストの最初の 1000 バイト
Statement_Outcome	QQC14	ステートメントの結果 • S - 正常 • U - 異常
Result_Rows	QQI2	戻された結果行数。次の SQL 操作に対してのみ設定され、それ以外のすべての場合は 0。 • IN - 挿入 • UP - 更新 • DL - 削除

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Dynamic_Replan_Reason_Code	QQC22	動的再計画 (アクセス・プランの再作成) • NA - 再計画はありません。 • NR - 新規リリース用に SQL QDT が再構築されました。 • A1 - テーブルまたはメンバーが、アクセス・プランが最後に構築された時に参照されたものと同一ではありません。これらが異なる理由として、以下が考えられます。 – オブジェクトが削除されて、再作成された。 – オブジェクトが保管されて、復元された。 – ライブラリー・リストが変更された。 – オブジェクトの名前が変更された。 – オブジェクトが移動された。 – オブジェクトが別のオブジェクトに上書きされた。 – これは、照会を含んでいるオブジェクトが復元された後の、この照会の最初の実行です。 • A2 - 再使用可能オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用不可 ODP を使用することを選択しました。 • A3 - 再使用不可オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用可能 ODP を使用することを選択しました。 • A4 - 最後にアクセス・プランが構築されてから、変更されたテーブル・メンバー内の行数が 10% を超えました。 • A5 - 照会内のテーブルの 1 つについて新規索引が存在します。 • A6 - このアクセス・プランに使用した索引は、もはや存在しないか、またはもはや有効ではありません。 • A7 - i5/OS 照会プログラムでは、システム・プログラミング変更のため、アクセス・プランを再作成する必要があります。 • A8 - 現行ジョブの CCSID が、最後にアクセス・プランを作成したジョブの CCSID と異なっています。 • A9 - 現行ジョブでの下記の値の 1 つ以上が、このアクセス・プランを最後に作成したジョブでの値と異なっています。 – 日付形式 – 日付区切り記号 – 時刻形式 – 時刻区切り記号

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Dynamic_Replan_Reason_Code (続き)	QQC22	• AA - 指定した分類順序テーブルが、このアクセス・プランを作成した時に使用した分類順序テーブルと異なっています。 • AB - 記憶域プールが変更されたか、または CHGQRYA コマンドの DEGREE パラメーターが変更されました。 • AC - システム機能 DB2 多重システムがインストールされたか、または除去されました。 • AD - 等級 (degree) 照会属性の値が変更されました。 • AE - ビューが、高水準言語によってオープンされたか、またはビューがマテリアライズされています。 • AF - ユーザー定義タイプまたはユーザー定義関数がアクセス・プランで参照されたのと同一のオブジェクトではありません。あるいは、SQL パスがアクセス・プランが構築されたときのものと同一ではありません。 • B0 - 指定したオプションが、照会オプション・ファイルの結果として変更されました。 • B1 - 現行ジョブで異なるコミットメント制御レベルで、アクセス・プランが生成されました。 • B2 - 以前のアクセス・プランと異なる静的カーソル応答セット・サイズで、アクセス・プランが生成されました。 • B3 - これが、作成後の最初の照会の実行であるので、照会が再最適化されました。つまり、実際の実パラメーター・マーカー値を使用した最初の実行です。 • B4 - 参照制約または検査制約が変更されたので、照会は再最適化されました。 • B5 - マテリアライズ照会表が変更されたので、照会は再最適化されました。
Data_Conversion_Reason_Code	QQC16	データ変換 • N - いいえ。 • 0 - 該当しません。 • 1 - 長さが一致しません。 • 2 - 数値タイプが一致しません。 • 3 - C ホスト変数が NUL で終了しています。 • 4 - ホスト変数または列が可変長で、その他は可変長ではありません。 • 5 - ホスト変数または列が可変長ではなく、その他は可変長です。 • 6 - ホスト変数または列が可変長で、その他は可変長ではありません。 • 7 - CCSID 変換。 • 8 - DRDA および NULL 可能、可変長、部分行の内容、派生式、または十分なホスト変数が指定されていないブロック化取り出し。 • 9 - 挿入のターゲット・テーブルが SQL テーブルではありません。

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Data_Conversion_Reason_Code (続き)		• 10 - ホスト変数が、取り出される TIME または TIMESTAMP 値を保持するには短すぎます。 • 11 - ホスト変数は DATE、TIME、または TIMESTAMP であり、取り出される値は文字ストリングです。 • 12 - 多すぎるホスト変数が指定され、レコードがブロック化されています。 • 13 - DRDA がブロック化された FETCH に使用され、INTO 文節に指定されたホスト変数の数は、選択リストにある結果値の数より少ないです。 • 14 - LOB ロケーターが使用され、コミットメント制御レベルは *ALL ではありません。
Total_Time_Milliseconds	QQI4	このステートメントの合計時間 (世界標準時、ミリ秒単位)。取り出しの場合、これには、カーソルのこの OPEN に対するすべての取り出しが含まれます。 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成された場合、この時間は、この照会のすべての実行に対する集約時間を表します。この場合、この時間は、実行の合計回数、COALESCE(QVP15F,1) によって除算され、照会の所定の実行に対する平均時間を決定することができます。
Rows_Fetched	QQI3	カーソルに対して取り出された合計行数 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成された場合、このカウントは、この照会のすべての実行に対する集約カウントを表します。この場合、このカウントは、実行の合計回数、COALESCE(QVP15F,1) によって除算され、照会の所定の実行に対して取り出される平均行を決定することができます。
End_Timestamp	QQETIM	SQL 要求が完了した時刻
Total_Time_Microseconds	QQI6	このステートメントの合計時間 (世界標準時)、マイクロ秒単位。取り出しの場合、これには、カーソルのこの OPEN に対するすべての取り出しが含まれます。 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成された場合、この時間は、この照会のすべての実行に対する集約時間を表します。この場合、この時間は、実行の合計回数、COALESCE(QVP15F,1) によって除算され、照会の所定の実行に対する平均時間を決定することができます。
SQL_Statement_Length	QQI7	SQL ステートメントの長さ
Insert_Unique_Count	QQI1	INSERT に関連した QDT の固有照会カウント。 QQUCNT には、ステートメントの WHERE 文節に関連した QDT の固有照会カウントが含まれます。
SQLCode	QQI8	SQL 戻りコード
SQLState	QQC81	SQLSTATE

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Close_Cursor_Mode	QVC101	クローズ・カーソル。次の値が指定可能です。 • *ENDJOB - SQL カーソルは、ジョブの終了時にクローズされます。 • *ENDMOD - SQL カーソルは、モジュールの終了時にクローズされます。 • *ENDPGM - SQL カーソルは、プログラムの終了時にクローズされます。 • *ENDSQL - SQL カーソルは、呼び出しスタック上の最初の SQL プログラムの終了時にクローズされます。 • *ENDACTGRP - SQL カーソルは、活動化グループの終了時にクローズされます。
Allow_Copy_Data_Value	QVC11	ALWCPYDTA 設定 (Y/N/O) • Y - データのコピーを使用できる。 • N - データのコピーを使用できない。 • O - パフォーマンス・データのコピーを使用するかは最適化プログラムが決定する。
PseudoOpen	QVC12	オープンをトリガーできる SQL 操作の疑似オープン (Y/N)。 • OP - オープン • IN - 挿入 • UP - 更新 • DL - 削除 • SI - 選択 • SV - 設定 • VI - 値 すべての操作について、これをブランクにすることができます。
PseudoClose	QVC13	クローズをトリガーできる SQL 操作の疑似クローズ (Y/N)。 • CL - クローズ • IN - 挿入 • UP - 更新 • DL - 削除 • SI - 選択 • SV - 設定 • VI - 値 すべての操作について、これをブランクにすることができます。
ODP_Implementation	QVC14	ODP 実施 • R - 再使用可能 ODP • N - 再使用不可 ODP • ' ' - 列は使用しません
Dynamic_Replan_SubCode	QVC21	動的再計画、サブタイプ理由コード

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Commitment_Control_Level	QVC41	コミットメント制御レベル。次の値が指定可能です。 • CS - カーソル固定 • CSKL - カーソル固定。排他ロックの保持。 • NC - コミットなし • RR - 反復可能読み取り • RREL - 反復可能読み取り。排他ロックの保持。 • RS - 読み取り固定 • RSEL - 読み取り固定。排他ロックの保持。 • UR - 非コミット読み取り
Skip_Locked_Data	QWC1B	SQL SKIP LOCKED DATA 文節が指定されたかどうか、およびその処理方法を指定します。 • Y - SKIP LOCKED DATA 文節が指定され、他のトランザクションによって保持される非互換ロックのある行はスキップされます。 • N - SKIP LOCKED DATA 文節は指定されませんでした。 • I - SKIP LOCKED DATA 文節が指定されましたが、分離レベルまたは照会実装により、この文節は無視されました。
Blocking_Type	QVC15	ブロック化のタイプ。次の値が指定可能です。 • S - 単一行、ALWBLK(*READ) • F - 1 行強制、ALWBLK(*NONE) • L - 限定ブロック、ALWBLK(*ALLREAD)
Delay_Prepare	QVC16	遅延準備 (Y/N)
Explainable	QVC1C	SQL ステートメントの説明可能性 (Y/N)
Naming_Convention	QVC17	命名規則。 次の値が指定可能です。 • N - システム命名規則 • S - SQL 命名規則
Dynamic_Processing_Type	QVC18	動的処理のタイプ。 • E - 拡張動的 • S - システム全体キャッシュ • L - ローカルの準備済みステートメント
LOB_Data_Optimized	QVC19	LOB データ・タイプの最適化 (Y/N)
Program_User_Profile_Used	QVC1A	コンパイル済みプログラムの実行時に使用するユーザー・プロファイル。次の値が指定可能です。 • N = ユーザー・プロファイルを命名規則によって判別する。 *SQL の場合は USRPRF(*OWNER) が使用されます。 *SYS の場合は USRPRF(*USER) が使用されます。 • U = USRPRF(*USER) を使用する。 • O = USRPRF(*OWNER) を使用する。
Dynamic_User_Profile_Used	QVC1B	動的 SQL ステートメントで使用されるユーザー・プロファイル。 • U = USRPRF(*USER) を使用する。 • O = USRPRF(*OWNER) を使用する。
Default_Collection	QVC1281	デフォルト・コレクションの名前。
Procedure_Name	QVC1282	SQL への CALL 上のプロシージャー名
Procedure_Library	QVC1283	SQL への CALL 上のプロシージャー・ライブラリー
SQL_Path	QQCLOB2	静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパス。

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Current_Schema	QVC1284	SQL 現行スキーマ
Binding_Type	QQC18	バインディング・タイプ • C - 列方向バインディング • R - 行方向バインディング
Cursor_Type	QQC61	カーソル・タイプ • NSA -スクロール不可能、ASENSITIVE、前方のみ • NSI -スクロール不可能、INSENSITIVE、前方のみ • NSS -スクロール不可能、SENSITIVE、前方のみ • SCA - スクロール可能、ASENSITIVE • SCI - スクロール可能、INSENSITIVE • SCS - スクロール可能、SENSITIVE
Statement_Originator	QVC1D	SQL ステートメント発信元 • U - ユーザー • S - システム
Hard_Close_Reason_Code	QQC15	SQL カーソルのハード・クローズの理由。次の理由が考えられます。 • 1 - 内部エラー • 2 - 排他ロック • 3 - 対話式 SQL の再使用制限 • 4 - ホスト変数の再使用制限 • 5 - 一時結果の制限 • 6 - カーソルの制限 • 7 - カーソルのハード・クローズが要求された • 8 - 内部エラー • 9 - カーソルのしきい値 • A - 最新表示エラー • B - カーソルの再使用エラー • C - DRDA AS のカーソルのクローズ • D - WITH HOLD のない DRDA AR • E - 反復可能読み取り • F - 競合または QSQPRCED のしきい値のロック - ライブラリー • G - ロックの競合または QSQPRCED のしきい値 - ファイル • H - 即時アクセス・プラン・スペースの実行 • I - QSQCSRTH ダミー・カーソルのしきい値 • J - ファイルのオーバーライド変更 • K - プログラム呼び出し変更 • L - ファイル・オープンのオプション変更 • M - ステートメントの再使用制限 • N - 内部エラー • O - ライブラリー・リストの変更 • P - 出口処理 • Q - SET SESSION USER ステートメント
Hard_Close_Subcode	QQC23	SQL カーソルのハード・クローズの理由サブコード

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Date_Format	QVC42	日付形式。次の値が指定可能です。 • ISO • USA • EUR • JIS • JUL • MDY • DMY • YMD
Date_Separator	QWC11	日付区切り記号。 次の値が指定可能です。 • "/" • "." • "," • "-" • " "
Time_Format	QVC43	時刻形式。 次の値が指定可能です。 • ISO • USA • EUR • JIS • HMS
Time_Separator	QWC12	時刻区切り記号。次の値が指定可能です。 • ":" • "." • "," • " "
Decimal_Point	QWC13	小数点。 次の値が指定可能です。 • "." • ","
Sort_Sequence_Table	QVC104	分類順序テーブル
Sort_Sequence_Library	QVC105	分類順序ライブラリー
Language_ID	QVC44	言語 ID
Country_ID	QVC23	国別 ID
First_N_Rows_Value	QQIA	FIRST n ROWS 文節で指定された値
Optimize_For_N_Rows _Value	QQF1	OPTIMIZE FOR n ROWS 文節で指定された値

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
SQL_Access_Plan_Reason_Code	QVC22	SQL アクセス・プラン再作成の理由コード。次の理由が考えられます。 • A1 - テーブルまたはメンバーが、アクセス・プランが最後に構築された時に参照されたものと同一ではありません。これらが異なる理由として、以下が考えられます。 – オブジェクトが削除されて、再作成された。 – オブジェクトが保管されて、復元された。 – ライブラリー・リストが変更された。 – オブジェクトの名前が変更された。 – オブジェクトが移動された。 – オブジェクトが別のオブジェクトに上書きされた。 – これは、照会を含んでいるオブジェクトが復元された後の、この照会の最初の実行です。 • A2 - 再使用可能オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用不可 ODP を使用することを選択しました。 • A3 - 再使用不可オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用可能 ODP を使用することを選択しました。 • A4 - 最後にアクセス・プランが構築されてから、変更されたテーブル内の行数が 10% を超えました。 • A5 - 照会内のテーブルの 1 つについて新規索引が存在します。 • A6 - このアクセス・プランに使用した索引は、もはや存在しないか、またはもはや有効ではありません。

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
SQL_Access_Plan_Reason_Code (続き)		• A7 - i5/OS 照会プログラムでは、システム・プログラミング変更のため、アクセス・プランを再作成する必要があります。 • A8 - 現行ジョブの CCSID が、最後にアクセス・プランを作成したジョブの CCSID と異なっています。 • A9 - 現行ジョブでの下記の値の 1 つ以上が、このアクセス・プランを最後に作成したジョブでの値と異なっています。 – 日付形式 – 日付区切り記号 – 時刻形式 – 時刻区切り記号 • AA - 指定した分類順序テーブルが、このアクセス・プランを作成した時に使用した分類順序テーブルと異なっています。 • AB - 記憶域プールが変更されたか、または CHGQRYA コマンドの DEGREE パラメーターが変更されました。 • AC - システム機能 DB2 多重システムがインストールされたか、または除去されました。 • AD - 等級 (degree) 照会属性の値が変更されました。 • AE - ビューが、高水準言語によってオープンされたか、またはビューがマテリアライズされています。 • AF - ユーザー定義タイプまたはユーザー定義関数がアクセス・プランで参照されたのと同一のオブジェクトではありません。あるいは、SQL パスがアクセス・プランが構築されたときのものと同一ではありません。 • B0 - 指定したオプションが、照会オプション・ファイルの結果として変更されました。 • B1 - 現行ジョブで異なるコミットメント制御レベルで、アクセス・プランが生成されました。 • B2 - 以前のアクセス・プランと異なる静的カーソル応答セット・サイズで、アクセス・プランが生成されました。 • B3 - これが、作成後の最初の照会の実行であるので、照会が再最適化されました。つまり、実際の実パラメーター・マーカー値を使用した最初の実行です。 • B4 - 参照制約または検査制約が変更されたので、照会は再最適化されました。 • B5 - マテリアライズ照会表が変更されたので、照会は再最適化されました。

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Access_Plan_Not_Saved_Reason_Code	QVC24	アクセス・プランが保管されない理由コード。次の理由が考えられます。 • A1 - プログラムまたはパッケージの関連スペースで、LSUP ロックの取得に失敗しました。 • A2 - プログラムの関連スペースの最初のバイトで、即時 LEAR スペース・ロケーション・ロックの取得に失敗しました。 • A3 - プログラムの関連スペースの最初のバイトで、即時 LENR スペース・ロケーション・ロックの取得に失敗しました。 • A5 - プログラムの ILE 関連スペースの最初のバイトで、即時 LEAR スペース・ロケーション・ロックの取得に失敗しました。 • A6 - ILE プログラムのスペースを拡張しようとしてエラーが発生しました。 • A7 - プログラムに空きがありません。 • A8 - プログラム関連スペースに空きがありません。 • A9 - プログラム関連スペースに空きがありません。 • AA - 保管する必要はありません。他のジョブですでに保管されています。 • AB - Query 最適化プログラムが QDT をロックできません。 • B1 - プログラム関連スペースの終了時に保管されました。 • B2 - プログラム関連スペースの終了時に保管されました。 • B3 - 適所に保管されました。 • B4 - 適所に保管されました。 • B5 - プログラム関連スペースの終了時に保管されました。 • B6 - 適所に保管されました。 • B7 - プログラム関連スペースの終了時に保管されました。 • B8 - プログラム関連スペースの終了時に保管されました。
Transaction_Context_ID	QVC81	トランザクション・コンテキスト ID。
Activation_Group_Mark	QVP152	活動化グループ・マーク
Open_Cursor_Threshold	QVP153	カーソルのしきい値をオープン
Open_Cursor_Close_Count	QVP154	オープン・カーソルのクローズ・カウント
Commitment_Control_Lock_Limit	QVP155	コミットメント制御ロック限界
Allow_SQL_Mixed_Constants	QWC15	SQL 混合定数の使用 (Y/N)
Suppress_SQL_Warnings	QWC16	SQL 警告メッセージの抑制 (Y/N)
Translate_ASCII	QWC17	ASCII をジョブに変換 (Y/N)
System_Wide_Statement_Cache	QWC18	システム規模の SQL ステートメント・キャッシュの使用 (Y/N)
LOB_Locator_Threshold	QVP159	LOB ロケーターのしきい値
Max_Decimal_Precision	QVP156	最大の 10 進数精度 (63/31)
Max_Decimal_Scale	QVP157	最大の 10 進数の位取り
Min_Decimal_Divide_Scale	QVP158	最小の 10 進数除算の位取り
Unicode_Normalization	QWC19	要求されたユニコード・データ正規化 (Y/N)
Statement_Text_Long	QQ1000L	ステートメント・テキスト全体
Old_Access_Plan_Length	QVP15B	古いアクセス・プランの長さ
New_Access_Plan_Length	QVP15C	新しいアクセス・プランの長さ

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Fast_Delete_Count	QVP151	SQL FAST DELETE のカウント。次の値が指定可能です。 • 0 = *OPTIMIZE または *DEFAULT • 1-999,999,999,999 = ユーザー指定値 • 'FFFFFFFFFFFFFFFF'x = *NONE
Statement_Max_Compression	QQF2	SQL ステートメント最大圧縮。次の値が指定可能です。 • 1 - *DEFAULT • 1 - ユーザー指定照会 • 2 - 全照会、全ユーザー、全システム • 3 - システム生成の内部照会
Current_User_Profile	QVC102	現行ユーザー・プロファイル名
Expression_Evaluator_Used	QVC1E	式エバリュエーターの使用 (Y/N)
Host_Server_Delta	QVP15A	ホスト・サーバー以外で消費された時間
NTS_Lock_Space_Id	QQC301	NTS ロック・スペース ID
IP_Address	QQC183	IP アドレス
IP_Type	QFC11	IP アドレス・タイプ • '0' = クライアント IP アドレスなし • '1' = IPV4 形式 • '2' = IPV6 形式 データベース・サーバー・ジョブに対してのみ適用可能です。
IP_Port_Number	QQSMINT2	IP ポート番号
NTS_Transaction_Id	QVC3004	NTS トランザクション ID
NTS_Format_Id_Length	QQSMINT3	NTS フォーマット ID の長さ
NTS_Transaction_ID_SubLength	QQSMINT4	NTS トランザクション ID のサブの長さ
Unique_Refresh_Counter2	QVRCNT	固有最新表示カウンター
Times_Run	QVP15F	このステートメントが実行された回数。 NULL の場合、ステートメントは 1 度実行されたことを意味します。 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成された場合、この値は設定できます。ただし、照会が決して完了しない、またはスナップショットの作成時に実行されていた場合は、NULL の可能性があります。プラン・キャッシュ・スナップショットを使用しない場合は、値は NULL です。
Full_Opens	QVP15E	フル・オープンとして処理された実行回数。NULL の場合は、最新表示カウント (qvrcnt) を使用して、オープンがフルであった (0) か、疑似オープン (>0) であったかを判別するのが妥当です。 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成された場合、この値は設定できます。ただし、照会が決して完了しない、またはスナップショットの作成時に実行されていた場合は、NULL の可能性があります。プラン・キャッシュ・スナップショットを使用しない場合は、値は NULL です。
Proc_In_Cache	QVC1F	プロシージャー定義が内部キャッシュ内で検出されました。 (Y/N) CALL ステートメントに対してのみ適用可能です。
Combined_Operation	QWC1A	ステートメントは、別のステートメント用の処理と連携して実行されました。 (Y/N) OPEN、FETCH および CLOSE ステートメントに対してのみ適用可能です。
Client_Applname	QVC3001	クライアント特殊レジスター - アプリケーション名
Client_Userid	QVC3002	クライアント特殊レジスター - ユーザー ID

表 44. QQQ1000 - SQL 情報 (続き)

ビュー列名	テーブル列名	説明
Client_Wrkstnname	QVC3003	クライアント特殊レジスター - ワークステーションの名前
Client_Acctng	QVC3005	クライアント特殊レジスター - 会計情報ストリング
Client_Programid	QVC3006	クライアント特殊レジスター - プログラムの名前
Interface_Information	QVC5001	CLIENT 特殊レジスター情報の一部。 3 つのタイプの情報がコロンで区切られて、この char500 列に保管されます。 • 最初の部分、インターフェース名、varchar(127); • 2 番目の部分、インターフェース・レベル、varchar(63); • 3 番目の部分、インターフェース・タイプ、varchar(63)
Open_Options	QVC82	オープン・オプションは、カーソルの実際の機能を表す、次の文字の組み合わせとして表示されます。文字値は、左寄せされ、右側にブランクが埋め込まれます。例えば、「RU」は、カーソルが読み取りおよび更新の両方が可能であることを示します。 • R - 読み取り可能 • W - 書き込み可能 • U - 更新可能 • D - 削除可能
Extended_Indicators	QWC1D	拡張標識を使用するために UPDATE または INSERT ステートメントが有効でした (Y/N)。
DECFLOAT_Rounding_Mode	QWC1C	DECFLOAT 計算および変換用に使用する丸めモード。 • 'E' = ROUND_HALF_EVEN • 'C' = ROUND_CEILING • 'D' = ROUND_DOWN • 'F' = ROUND_FLOOR • 'G' = ROUND_HALF_DOWN • 'H' = ROUND_HALF_UP • 'U' = ROUND_UP
SQL_DECFLOAT_Warnings	QWC1E	0 による除算、オーバーフロー、アンダーフロー、無効なオペランド、不正確な結果、または正常以下の数値が関係する DECFLOAT 計算および変換は、警告を戻します (Y/N)。
Worst_Time_Micro	QVP15D	NULL でない場合、これは、この照会の単一の実行のうち最も遅いものの時間です。 注: SQL プラン・キャッシュ・スナップショットの使用中にモニター・ファイルが作成される場合、この時間は、照会の単一の実行のうち最も長いものの実行時間を表します。この値が NULL の場合、最長の実行情報は、使用できません。その場合、QQI6 が次善の応答である場合があります。そのフィールドの適切な使用については、QQI6 の資料を参照してください。
SQUNQCT	QQINT05	ODP はないが、ホスト変数に入れて渡されるステートメントを一意的に識別するために使用される固有カウント。 QQUCNT が 0 であり、ステートメントがホスト変数に渡される場合、この値は非ゼロです。一例として CALL ステートメントが挙げられます。

データベース・モニター・ビュー 3000 - テーブル走査

データベース・モニター QQQ3000 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3000 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
```

```
QQJFLD as Join_Column,
QQRDBN as Relational_Database_Name,
QQSYS as System_Name,
QQJOB as Job_Name,
QQUSER as Job_User,
QQJNUM as Job_Number,
QQI9 as Thread_ID,
QQUCNT as Unique_Count,
QQUDEF as User_Defined,
QQQDTN as Unique_SubSelect_Number,
QQQDTL as SubSelect_Nested_Level,
QQMATN as Materialized_View_Subselect_Number,
QQMATL as Materialized_View_Nested_Level,
QVP15E as Materialized_View_Union_Level,
QVP15A as Decomposed_Subselect_Number,
QVP15B as Total_Number_Decomposed_SubSelects,
QVP15C as Decomposed_SubSelect_Reason_Code,
QVP15D as Starting_Decomposed_SubSelect,
QQTLN as System_Table_Schema,
QQTFN as System_Table_Name,
QQTMN as Member_Name,
QQPTLN as System_Base_Table_Schema,
QQPTFN as System_Base_Table_Name,
QQPTMN as Base_Member_Name,
QQTOTR as Table_Total_Rows,
QQREST as Estimated_Rows_Selected,
QQAJN as Estimated_Join_Rows,
QQEPT as Estimated_Processing_Time,
QQJNP as Join_Position,
QQI1 as DataSpace_Number,
QQC21 as Join_Method,
QQC22 as Join_Type,
QQC23 as Join_Operator,
QQI2 as Index_Advised_Columns_Count,
QQDSS as DataSpace_Selection,
QQIDXA as Index_Advised,
QQRCOD as Reason_Code,
QQIDXD as Index_Advised_Columns,
QVQTBL as Table_Name,
QVQLIB as Table_Schema,
QVPTBL as Base_Table_Name,
QVPLIB as Base_Table_Schema,
QVBNDY as Bound,
QVRCNT as Unique_Refresh_Counter,
QVJFANO as Join_Fanout,
QVFILES as Join_Table_Count,
QVPARPF as Parallel_Prefetch,
QVPARPL as Parallel_PreLoad,
QVPARD as Parallel_Degree_Requested,
QVPARU as Parallel_Degree_Used,
QVPARRC as Parallel_Degree_Reason_Code,
QVCTIM as Estimated_Cumulative_Time,
QQC11 as Skip_Sequential_Table_Scan,
QQI3 as Table_Size,
QVC3001 as DataSpace_Selection_Columns,
QQC14 as Derived_Column_Selection,
QVC3002 as Derived_Column_Selection_Columns,
QQC18 as Read_Trigger,
QVP157 as UDTF_Cardinality,
QVC1281 as UDTF_Specific_Name,
QVC1282 as UDTF_Specific_Schema,
QVP154 as Pool_Size,
QVP155 as Pool_Id,
QQC13 as MQT_Replacement,
```

```
        QQINT03 as Estimated_Storage

FROM    UserLib/DBMONTABLE
WHERE   QQRID=3000)
```

表 45. QQQ3000 - テーブル走査

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
Table_Total_Rows	QQTOTR	テーブル内の合計行数
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI1	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合

表 45. *QQQ3000 - テーブル走査 (続き)*

ビュー列名	テーブル列名	説明
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Index_Advised_Columns_Count	QQI2	索引走査のキー位置決めを使用する推奨列の数
DataSpace_Selection	QQDSS	データ・スペース選択 • Y - はい • N - いいえ
Index_Advised	QQIDXA	推奨索引 • Y - はい • N - いいえ
Reason_Code	QQRCOD	理由コード • T1 - 索引がない。 • T2 - 索引があるが、どれも使用できない。 • T3 - 最適化プログラムが使用可能な索引についてテーブル走査を選択した。
Index_Advised_Columns	QQIDXD	推奨索引の列
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Bound	QVBNDY	入出力制約または CPU 制約。 次の値が指定可能です。 • I - 入出力制約 • C - CPU 制約
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。

表 45. *QQQ3000 - テーブル走査 (続き)*

ビュー列名	テーブル列名	説明
Join_Table_Count	QVFILES	結合されたテーブルの数
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (Y/N)
Parallel_Degree_Requested	QVPARD	要求された並列度
Parallel_Degree_Used	QVPARU	使用された並列度
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Skip_Sequential_Table_Scan	QQC11	順次テーブル走査をスキップ (Y/N)
Table_Size	QQI3	照会されたテーブルのサイズ
DataSpace_Selection_Columns	QVC3001	データ・スペース選択で使用された列
Derived_Column_Selection	QQC14	派生列選択 (Y/N)
Derived_Column_Selection_Columns	QVC3002	派生列選択で使用された列
Read_Trigger	QQC18	読み取りトリガー (Y/N)
UDTF_Cardinality	QVP157	ユーザー定義テーブル関数基数
UDTF_Specific_Name	QVC1281	ユーザー定義テーブル関数特定名
UDTF_Specific_Schema	QVC1282	ユーザー定義テーブル関数特定スキーマ
Pool_Size	QVP154	プール・サイズ
Pool_Id	QVP155	プール ID
MQT_Replacement	QQC13	マテリアライズ照会表による照会表の置換 (Y/N)
Estimated_Storage	QQINT03	一時索引の作成に使用される一時ストレージの見積量 (メガバイト単位)。

データベース・モニター・ビュー 3001 - 使用された索引

データベース・モニター QQQ3001 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3001 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQTLN as System_Table_Schema,
          QQTFN as System_Table_Name,
          QQTMN as Member_Name,
          QQPTLN as System_Base_Table_Schema,
```

```
QQPTFN as System_Base_Table_Name,
QQPTMN as Base_Member_Name,
QQILNM as System_Index_Schema,
QQIFNM as System_Index_Name,
QQIMNM as Index_Member_Name,
QQTOTR as Table_Total_Rows,
QQREST as Estimated_Rows_Selected,
QQFKEY as Index_Probe_Keys,
QQKSEL as Index_Scan_Keys,
QQAJN as Estimated_Join_Rows,
QQEPT as Estimated_Processing_Time,
QQJNP as Join_Position,
QQI1 as DataSpace_Number,
QQC21 as Join_Method,
QQC22 as Join_Type,
QQC23 as Join_Operator,
QQI2 as Index_Advised_Probe_Count,
QQKP as Index_Probe_Used,
QQI3 as Index_Probe_Column_Count,
QQKS as Index_Scan_Used,
QQDSS as DataSpace_Selection,
QQIDXA as Index_Advised,
QQRCOD as Reason_Code,
QQIDXD as Index_Advised_Columns,
QQC11 as Constraint,
QQ1000 as Constraint_Name,
QVQTBL as Table_Name,
QVQLIB as Table_Schema,
QVPTBL as Base_Table_Name,
QVPLIB as Base_Table_Schema,
QVINAM as Index_Name,
QVILIB as Index_Schema,
QVBNDY as Bound,
QVRCNT as Unique_Refresh_Counter,
QVJFANO as Join_Fanout,
QVFILES as Join_Table_Count,
QVPARPF as Parallel_Prefetch,
QVPARPL as Parallel_Preload,
QVPARD as Parallel_Degree_Requested,
QVPARU as Parallel_Degree_Used,
QVPARRC as Parallel_Degree_Reason_Code,
QVCTIM as Estimated_Cumulative_Time,
QVc14 as Index_Only_Access,
QQc12 as Index_Fits_In_Memory,
QQC15 as Index_Type,
QVC12 as Index_Usage,
QQI4 as Index_Entries,
QQI5 as Unique_Keys,
QQI6 as Percent_Overflow,
QQI7 as Vector_Size,
QQI8 as Index_Size,
QQIA as Index_Page_Size,
QVP154 as Pool_Size,
QVP155 as Pool_Id,
QVP156 as Table_Size,
QQC16 as Skip_Sequential_Table_Scan,
QVC13 as Tertiary_Indexes_Exist,
QVC3001 as DataSpace_Selection_COlumns,
QQC14 as Derived_Column_Selection,
QVC3002 as Derived_Column_Selection_Columns,
QVC3003 as Table_Columns_For_Index_Probe,
QVC3004 as Table_Columns_For_Index_Scan,
QVC3005 as Join_Selection_Columns,
QVC3006 as Ordering_Columns,
QVC3007 as Grouping_Columns,
QQC18 as Read_Trigger,
QVP157 as UDTF_Cardinality,
```

```
        QVC1281 as UDTF_Specific_Name,
        QVC1282 as UDTF_Specific_Schema,
        QQC13 as MQT_Replacement
FROM    UserLib/DBMONTable
WHERE   QQRID=3001)
```

表 46. QQQ3001 - 使用された索引

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
System_Index_Schema	QQILNM	アクセスに使用された索引のスキーマ名
System_Index_Name	QQIFNM	アクセスで使用された索引の名前
Index_Member_Name	QQIMNM	アクセスで使用された索引のメンバー名
Table_Total_Rows	QQTOTR	基礎テーブルの合計行数
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Index_Probe_Keys	QQFKEY	索引走査のキー位置決めにより選択された列
Index_Scan_Keys	QQKSEL	索引走査のキー選択により選択された列
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)

表 46. QQQ3001 - 使用された索引 (続き)

ビュー列名	テーブル列名	説明
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI1	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Index_Advised_Probe_Count	QQI2	索引走査のキー位置決めを使用する推奨キー列の数
Index_Probe_Used	QQKP	索引走査のキー位置決め • Y - はい • N - いいえ
Index_Probe_Column_Count	QQI3	使用する索引に対して索引走査のキー位置決めを使用する列の数
Index_Scan_Used	QQKS	索引走査のキー選択 • Y - はい • N - いいえ
DataSpace_Selection	QQDSS	データ・スペース選択 • Y - はい • N - いいえ
Index_Advised	QQIDXA	推奨索引 • Y - はい • N - いいえ
Reason_Code	QQRCOD	理由コード • I1 - 行選択 • I2 - 順序付け/グループ化 • I3 - 行選択と順序付け/グループ化 • I4 - ネスト・ループ結合 • I5 - ビットマップ処理を使用した行選択

表 46. QQQ3001 - 使用された索引 (続き)

ビュー列名	テーブル列名	説明
Index_Advised_Columns	QQIDXD	推奨索引の列
Constraint	QQC11	索引は制約である (Y/N)
Constraint_Name	QQ1000	制約名
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Index_Name	QVINAM	使用された索引 (または制約)、長い名前
Index_Schema	QVILIB	使用された索引のライブラリー、長い名前
Bound	QVBNDY	入出力制約または CPU 制約。 次の値が指定可能です。 • I - 入出力制約 • C - CPU 制約
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_Preload	QVPARPL	並列プリロード (Y/N)
Parallel_Degree_Requested	QVPARD	要求された並列度
Parallel_Degree_Used	QVPARU	使用された並列度
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Index_Only_Access	QVC14	索引専用アクセス (Y/N)
Index_Fits_In_Memory	QQC12	メモリー内での索引当てはめ (Y/N)
Index_Type	QQC15	索引タイプ。 次の値が指定可能です。 • B - 2 進基数索引 • C - 制約 (2 進基数) • E - コード化ベクトル索引 (EVI) • X - 照会で作成された一時索引
Index_Usage	QVC12	索引の使用法。次の値が指定可能です。 • P - 1 次索引 • T - 3 次索引 (AND/OR)
Index_Entries	QQI4	索引項目の数
Unique_Keys	QQI5	固有キー値の数

表 46. QQQ3001 - 使用された索引 (続き)

ビュー列名	テーブル列名	説明
Percent_Overflow	QQI6	パーセント・オーバーフロー
Vector_Size	QQI7	ベクトル・サイズ
Index_Size	QQI8	索引サイズ
Index_Page_Size	QQIA	索引ページ・サイズ
Pool_Size	QVP154	プール・サイズ
Pool_Id	QVP155	プール ID
Table_Size	QVP156	テーブル・サイズ
Skip_Sequential_Table_Scan	QQC16	順次テーブル走査をスキップ (Y/N)
Tertiary_Indexes_Exist	QVC13	3 次索引の存在 (Y/N)
DataSpace_Selection_Columns	QVC3001	データ・スペース選択で使用された列
Derived_Column_Selection	QQC14	派生列選択 (Y/N)
Derived_Column_Selection_Columns	QVC3002	派生列選択で使用された列
Table_Column_For_Index_Probe	QVC3003	索引走査のキー位置決めで使用された列
Table_Column_For_Index_Scan	QVC3004	索引走査のキー選択で使用された列
Join_Selection_Columns	QVC3005	結合選択で使用された列
Ordering_Columns	QVC3006	順序付けで使用された列
Grouping_Columns	QVC3007	グループ化で使用された列
Read_Trigger	QQC18	読み取りトリガー (Y/N)
UDTF_Cardinality	QVP157	ユーザー定義テーブル関数基数
UDTF_Specific_Name	QVC1281	ユーザー定義テーブル関数特定名
UDTF_Specific_Schema	QVC1282	ユーザー定義テーブル関数特定スキーマ
MQT_Replacement	QQC13	マテリアライズ照会表による照会表の置換 (Y/N)

データベース・モニター・ビュー 3002 - 作成された索引

データベース・モニター QQQ3002 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3002 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQTLN as System_Table_Schema,
```

```
QQTFN as System_Table_Name,
QQTMN as Member_Name,
QQPTLN as System_Base_Table_Schema,
QQPTFN as System_Base_Table_Name,
QQPTMN as Base_Member_Name,
QQILNM as System_Index_Schema,
QQIFNM as System_Index_Name,
QQIMNM as Index_Member_Name,
QQNTNM as NLSS_Table,
QQNLNM as NLSS_Library,
QQSTIM as Start_Timestamp,
QQETIM as End_Timestamp,
QQTOTR as Table_Total_Rows,
QQRIDX as Created_Index_Entries,
QQREST as Estimated_Rows_Selected,
QQFKEY as Index_Probe_Keys,
QQKSEL as Index_Scan_Keys,
QQAJN as Estimated_Join_Rows,
QQEPT as Estimated_Processing_Time,
QQJNP as Join_Position,
QQI1 as DataSpace_Number,
QQC21 as Join_Method,
QQC22 as Join_Type,
QQC23 as Join_Operator,
QQI2 as Index_Advised_Probe_Count,
QQKP as Index_Probe_Used,
QQI3 as Index_Probe_Column_Count,
QQKS as Index_Scan_Used,
QQDSS as DataSpace_Selection,
QQIDXA as Index_Advised,
QQRCOD as Reason_Code,
QQIDXD as Index_Advised_Columns,
QQ1000 as Created_Index_Columns,
QVQTBL as Table_Name,
QVQLIB as Table_Schema,
QVPTBL as Base_Table_Name,
QVPLIB as Base_Table_Schema,
QVINAM as Index_Name,
QVILIB as Index_Schema,
QVBNDY as Bound,
QVRCNT as Unique_Refresh_Counter,
QVJFANO as Join_Fanout,
QVFILES as Join_Table_Count,
QVPARPF as Parallel_Prefetch,
QVPARPL as Parallel_Preload,
QVPARD as Parallel_Degree_Requested,
QVPARU as Parallel_Degree_Used,
QVPARRC as Parallel_Degree_Reason_Code,
QVCTIM as Estimated_Cumulative_Time,
QQC101 as Created_Index_Name,
QQC102 as Created_Index_Schema,
QQI4 as Created_Index_Page_Size,
QQI5 as Created_Index_Row_Size,
QQC14 as Created_Index_Used_ACS_Table,
QQC103 as Created_Index_ACS_Table,
QQC104 as Created_Index_ACS_Library,
QVC13 as Created_Index_Reusable,
QVC14 as Created_Index_Sparse,
QVC1F as Created_Index_Type,
QVP15F as Created_Index_Unique_EVI_Count,
QVC15 as Permanent_Index_Created,
QVC16 as Index_From_Index,
QVP151 as Created_Index_Parallel_Degree_Requested,
QVP152 as Created_Index_Parallel_Degree_Used,
QVP153 as Created_Index_Parallel_Degree_Reason_Code,
QVC17 as Index_Only_Access,
QVC18 as Index_Fits_In_Memory,
```

```
        QVC1B as Index_Type,
        QQI6 as Index_Entries,
        QQI7 as Unique_Keys,
        QVP158 as Percent_Overflow,
        QVP159 as Vector_Size,
        QQI8 as Index_Size,
        QVP156 as Index_Page_Size,
        QVP154 as Pool_Size,
        QVP155 as Pool_ID,
        QVP157 as Table_Size,
        QVC1C as Skip_Sequential_Table_Scan,
        QVC3001 as DataSpace_Selection_Columns,
        QVC1E as Derived_Column_Selection,
        QVC3002 as Derived_Column_Selection_Columns,
        QVC3003 as Table_Column_For_Index_Probe,
        QVC3004 as Table_Column_For_Index_Scan,
        QQC18 as Read_Trigger,
        QQC13 as MQT_Replacement,
        QQC16 as Reused_Temporary_Index
        QQINT03 as Estimated_Storage
FROM    UserLib/DBMONTable
WHERE   QQRID=3002)
```

表 47. QQQ3002 - 作成された索引

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名

表 47. QQQ3002 - 作成された索引 (続き)

ビュー列名	テーブル列名	説明
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
System_Index_Schema	QQILNM	アクセスに使用された索引のスキーマ名
System_Index_Name	QQIFNM	アクセスで使用された索引の名前
Index_Member_Name	QQIMNM	アクセスで使用された索引のメンバー名
NLSS_Table	QQNTNM	NLSS テーブル
NLSS_Library	QQNLNM	NLSS ライブラリー
Start_Timestamp	QQSTIM	開始タイム・スタンプ - 使用可能な場合
End_Timestamp	QQETIM	終了タイム・スタンプ - 使用可能な場合
Table_Total_Rows	QQTOTR	テーブル内の合計行数
Created_Index_Entries	QQRIDX	作成された索引内の項目数
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Index_Probe_Keys	QQFKEY	索引走査のキー位置決めにより選択されたキー
Index_Scan_Keys	QQKSEL	索引走査のキー選択により選択されたキー
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI1	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Index_Advised_Probe_Count	QQI2	索引走査のキー位置決めを使用する推奨キー列の数
Index_Probe_Used	QQKP	索引走査のキー位置決め • Y - はい • N - いいえ

表 47. QQQ3002 - 作成された索引 (続き)

ビュー列名	テーブル列名	説明
Index_Probe_Column_Count	QQI3	使用する索引に対して索引走査のキー位置決めを使用する列の数
Index_Scan_Used	QQKS	索引走査のキー選択 • Y - はい • N - いいえ
DataSpace_Selection	QQDSS	データ・スペース選択 • Y - はい • N - いいえ
Index_Advised	QQIDXA	推奨索引 • Y - はい • N - いいえ
Reason_Code	QQRCOD	理由コード • I1 - 行選択 • I2 - 順序付け/グループ化 • I3 - 行選択と順序付け/グループ化 • I4 - ネスト・ループ結合
Index_Advised_Columns	QQIDXD	推奨索引のキー列
Created_Index_Columns	QQ1000	作成された索引のキー列
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Index_Name	QVINAM	使用された索引 (または制約)、長い名前
Index_Schema	QVILIB	使用された索引のスキーマ、長い名前
Bound	QVBNDY	入出力制約または CPU 制約。 次の値が指定可能です。 • I - 入出力制約 • C - CPU 制約
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_Preload	QVPARPL	並列プリロード (使用された索引)

表 47. QQQ3002 - 作成された索引 (続き)

ビュー列名	テーブル列名	説明
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Created_Index_Name	QQC101	作成された索引の名前 - 使用可能な場合
Created_Index_Schema	QQC102	作成された索引のスキーマ - 使用可能な場合
Created_Index_Page_Size	QQI4	作成された索引のページ・サイズ
Created_Index_Row_Size	QQI5	作成された索引の行サイズ
Created_Index_Used_ACS_Table	QQC14	代替照合シーケンス・テーブルで使用されている作成された索引 (Y/N)
Created_Index_ACS_Table	QQC103	作成された索引の代替照合順序テーブル
Created_Index_ACS_Library	QQC104	作成された索引の代替照合順序ライブラリー
Created_Index_Reusable	QVC13	作成された索引の再利用可能性 (Y/N)
Created_Index_Sparse	QVC14	作成された索引は疎索引 (Y/N)
Created_Index_Type	QVC1F	作成された索引のタイプ。次の値が指定可能です。 • B - 2 進基数索引 • E - コード化ベクトル索引 (EVI)
Created_Index_Unique_EVI_Count	QVP15F	固有の索引値の数 (作成された索引が EVI 索引の場合)
Permanent_Index_Created	QVC15	永続索引の作成 (Y/N)
Index_From_Index	QVC16	索引からの索引 (Y/N)
Created_Index_Parallel_Degree_Requested	QVP151	要求された並列度 (作成された索引)
Created_Index_Parallel_Degree_Used	QVP152	使用された並列度 (作成された索引)
Created_Index_Parallel_Degree_Reason_Code	QVP153	並列処理が制限された理由 (作成された索引)
Index_Only_Access	QVC17	索引専用アクセス (Y/N)
Index_Fits_In_Memory	QVC18	メモリー内での索引当てはめ (Y/N)
Index_Type	QVC1B	索引タイプ。 次の値が指定可能です。 • B - 2 進基数索引 • C - 制約 (2 進基数) • E - コード化ベクトル索引 (EVI) • T - 3 次索引 (AND/OR)
Index_Entries	QQI6	索引入力の数 (使用された索引)
Unique_Keys	QQI7	固有キー値の数 (使用された索引)
Percent_Overflow	QVP158	パーセント・オーバーフロー (使用された索引)
Vector_Size	QVP159	ベクトル・サイズ (使用された索引)
Index_Size	QQI8	使用された索引のサイズ
Index_Page_Size	QVP156	索引ページ・サイズ
Pool_Size	QVP154	プール・サイズ
Pool_ID	QVP155	プール ID
Table_Size	QVP157	テーブル・サイズ

表 47. QQQ3002 - 作成された索引 (続き)

ビュー列名	テーブル列名	説明
Skip_Sequential_Table_Scan	QVC1C	順次テーブル走査をスキップ (Y/N)
DataSpace_Selection_Columns	QVC3001	データ・スペース選択で使用された列
Derived_Column_Selection	QVC1E	派生列選択 (Y/N)
Derived_Column_Selection_Columns	QVC3002	派生列選択で使用された列
Table_Columns_For_Index_Probe	QVC3003	索引走査のキー位置決めで使用された列
Table_Columns_For_Index_Scan	QVC3004	索引走査のキー選択で使用された列
Read_Trigger	QQC18	読み取りトリガー (Y/N)
MQT_Replacement	QQC13	マテリアライズ照会表による照会表の置換 (Y/N)
Reused_Temporary_Index	QQC16	再使用された一時索引 (Y/N)
Estimated_Storage	QQINT03	一時索引の作成に使用される一時ストレージの見積量 (メガバイト単位)。

データベース・モニター・ビュー 3003 - 照会分類

データベース・モニター QQQ3003 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3003 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQSTIM as Start_Timestamp,
          QQETIM as End_Timestamp,
          QQRSS as Sorted_Rows,
          QQI1 as Sort_Space_Size,
          QQI2 as Pool_Size,
          QQI3 as Pool_Id,
          QQI4 as Internal_Sort_Buffer_Length,
          QQI5 as External_Sort_Buffer_Length,
          QQRCOD as Reason_Code,
          QQI7 as Union_Reason_Subcode,
          QVBNDY as Bound,
          QVRCNT as Unique_Refresh_Counter,
          QVPARPF as Parallel_Prefetch,
          QVPARPL as Parallel_PreLoad,
          QVPARD as Parallel_Degree_Requested,
          QVPARU as Parallel_Degree_Used,
          QVPARRC as Parallel_Degree_Reason_Code,
          QQEPT as Estimated_Processing_Time,
          QVCTIM as Estimated_Cumulative_Time,
```

```
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI6 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QQINT03 as Estimated_Storage
FROM    UserLib/DBMONTable
WHERE   QQRID=3003)
```

表 48. QQQ3003 - 照会分類

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Start_Timestamp	QQSTIM	開始タイム・スタンプ - 使用可能な場合
End_Timestamp	QQETIM	終了タイム・スタンプ - 使用可能な場合
Sorted_Rows	QQRSS	選択された、またはソートされた行数の見積もり
Sort_Space_Size	QQI1	分類スペースのサイズの見積もり
Pool_Size	QQI2	プール・サイズ
Pool_Id	QQI3	プール ID
Internal_Sort_Buffer_Length	QQI4	内部分類バッファー長
External_Sort_Buffer_Length	QQI5	外部分類バッファー長

表 48. QQQ3003 - 照会分類 (続き)

ビュー列名	テーブル列名	説明
Reason_Code	QQRCOD	理由コード • F1 - 照会には、2 つ以上のテーブルからのグループ化列 (Group By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからのグループ化列が含まれていたりします。 • F2 - 照会には、2 つ以上のテーブルからの順序付け列 (Order By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからの順序付け列が含まれていたりします。 • F3 - グループ化列と順序付けフィールドには、互換性がありません。 • F4 - DISTINCT が照会に対して指定されました。 • F5 - UNION が照会に対して指定されました。 • F6 - 照会は、分類を使用して実施しなければなりませんでした。順序付けでキーの長さが 2000 バイトを超えているかまたは 120 を超えるキー列が指定されました。
Reason_Code (続き)		• F7 - Query 最適化プログラムは、照会の結果を順序付けする索引ではなくて、分類を使用することを選択します。 • F8 - 指定した列選択を実行して入出力待ち時間を最小にします。 • FC - 照会はグループ・フィールドを含み、照会で少なくとも 1 つの物理ファイルには読み取りトリガーがあります。
Union_Reason_Subcode	QQI7	共用体の理由サブコード • 51 - 照会が UNION および ORDER BY を含んでいる • 52 - 照会が UNION ALL を含んでいる
Bound	QVBNDY	入出力制約または CPU 制約。 次の値が指定可能です。 • I - 入出力制約 • C - CPU 制約
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号

表 48. *QQQ3003 - 照会分類 (続き)*

ビュー列名	テーブル列名	説明
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Estimated_Storage	QQINT03	一時索引の作成に使用される一時ストレージの見積量 (メガバイト単位)。

データベース・モニター・ビュー 3004 - 一時テーブル

データベース・モニター QQQ3004 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3004 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
```

```
        QVP15A as Decomposed_Subselect_Number,
        QVP15B as Total_Number_Decomposed_SubSelects,
        QVP15C as Decomposed_SubSelect_Reason_Code,
        QVP15D as Starting_Decomposed_SubSelect,
        QQTLN as System_Table_Schema,
        QQTFN as System_Table_Name,
        QQTMN as Member_Name,
        QQPTLN as System_Base_Table_Schema,
        QQPTFN as System_Base_Table_Name,
        QQPTMN as Base_Member_Name,
        QQSTIM as Start_Timestamp,
        QQETIM as End_Timestamp,
        QQC11 as Has_Default_Values,
        QQTMPR as Table_Rows,
        QQRCOD as Reason_Code,
        QVQTBL as Table_Name,
        QVQLIB as Table_Schema,
        QVPTBL as Base_Table_Name,
        QVPLIB as Base_Table_Schema,
        QQC101 as Temporary_Table_Name,
        QQC102 as Temporary_Table_Schema,
        QVBNDY as Bound,
        QVRCNT as Unique_Refresh_Counter,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QVPARPF as Parallel_Prefetch,
        QVPARPL as Parallel_PreLoad,
        QVPARD as Parallel_Degree_Requested,
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI6 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QQI2 as Temporary_Table_Row_Size,
        QQI3 as Temporary_Table_Size,
        QQC12 as Temporary_Query_Result,
        QQC13 as Distributed_Temporary_Table,
        QVC3001 as Distributed_Temporary_Data_Nodes,
        QQI7 as Materialized_Subqery_QDT_Level,
        QQI8 as Materialized_Union_QDT_Level,
        QQC14 as View_Contains_Union,
        QQINT03 as Estimated_Storage
FROM    UserLib/DBMONTable
WHERE   QQRID=3004)
```

表 49. QQQ3004 - 一時テーブル

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード

表 49. *QQQ3004 - 一時テーブル (続き)*

ビュー列名	テーブル列名	説明
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
Start_Timestamp	QQSTIM	開始タイム・スタンプ - 使用可能な場合
End_Timestamp	QQETIM	終了タイム・スタンプ - 使用可能な場合
Has_Default_Values	QQC11	デフォルト値が一時ファイルに表れる可能性がある • Y - はい • N - いいえ
Table_Rows	QQTMPR	一時ファイル内の見積もり行数

表 49. QQQ3004 - 一時テーブル (続き)

ビュー列名	テーブル列名	説明
Reason_Code	QQRCOD	理由コード。 次の値が指定可能です。 • F1 - 照会には、2 つ以上のテーブルからのグループ化列 (Group By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからのグループ化列が含まれていたりします。 • F2 - 照会には、2 つ以上のテーブルからの順序付け列 (Order By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからの順序付け列が含まれていたりします。 • F3 - グループ化列と順序付けフィールドには、互換性がありません。 • F4 - DISTINCT が照会に対して指定されました。 • F5 - UNION が照会に対して指定されました。 • F6 - 照会は、分類を使用して実施しなければなりませんでした。順序付けでキーの長さが 2000 バイトを超えているかまたは 120 を超えるキー列が指定されました。 • F7 - Query 最適化プログラムは、照会の結果を順序付けする索引ではなくて、分類を使用することを選択します。 • F8 - 指定した列選択を実行して入出力待ち時間を最小にします。 • F9 - Query 最適化プログラムは、索引を使ってグループ化するのではなくて、ハッシュ・アルゴリズムを使用することを選択します。 • FA - 照会には、一時テーブルを必要とする結合条件が含まれています。 • FB - Query 最適化プログラムは、照会ごとに特定の相関グループをインプリメントするために、ランタイム一時ファイルを作成します。 • FC - 照会はグループ・フィールドを含み、照会で少なくとも 1 つの物理ファイルには読み取りトリガーがあります。 • FD - Query 最適化プログラムは静的カーソル要求のためにランタイム一時ファイルを作成します。 • H1 - 照会中のテーブルが結合論理ファイルであって、その結合タイプ (JDFTVAL) が照会に指定された結合タイプと一致しない場合。 • H2 - 論理テーブルに指定された形式が複数の基礎テーブルを参照しています。 • H3 - テーブルは、複合 SQL ビューであって、その SQL ビューの結果を入れるための一時テーブルを必要としています。 • H4 - 更新可能照会の場合、副選択はこのテーブルの列で、更新中のいずれかの列と一致する列を参照します。 • H5 - 更新可能照会の場合、副選択は更新中のテーブルを基礎とした SQL ビューを参照します。 • H6 - 削除可能照会の場合、副選択は行の削除元テーブル、SQL ビュー、または行の削除元テーブルを基礎とした索引のいずれかを参照します。 • H7 - ユーザー定義テーブル関数がマテリアライズされました。
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのライブラリー、長い名前
Temporary_Table_Name	QQC101	一時テーブル名
Temporary_Table_Schema	QQC102	一時テーブル・スキーマ

表 49. QQQ3004 - 一時テーブル (続き)

ビュー列名	テーブル列名	説明
Bound	QVBNDY	入出力制約または CPU 制約。 次の値が指定可能です。 • I - 入出力制約 • C - CPU 制約
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (Y/N)
Parallel_Degree_Requested	QVPARD	要求された並列度
Parallel_Degree_Used	QVPARU	使用された並列度
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Temporary_Table_Row_Size	QQI2	一時テーブルの行サイズ (バイト単位)
Temporary_Table_Size	QQI3	一時テーブルの見積もりサイズ (バイト単位)
Temporary_Query_Result	QQC12	照会結果が含まれる一時結果テーブル (Y/N)
Distributed_Temporary_Table	QQC13	分散テーブル (Y/N)
Distributed_Temporary_Data_Nodes	QVC3001	一時テーブルのデータ・ノード
Materialized_Subqery_QDT_Level	QQI7	マテリアライズした副照会 QDT レベル

表 49. QQQ3004 - 一時テーブル (続き)

ビュー列名	テーブル列名	説明
Materialized_Union_QDT_Level	QQI8	マテリアライズした共用体 QDT レベル
View_Contains_Union	QQC14	ビューの中の共用体 (Y/N)
Estimated_Storage	QQINT03	一時索引の作成に使用される一時ストレージの見積量 (メガバイト単位)。

データベース・モニター・ビュー 3005 - ロックされたテーブル

データベース・モニター QQQ3005 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3005 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQTLN as System_Table_Schema,
          QQTFN as System_Table_Name,
          QQTMN as Member_Name,
          QQPTLN as System_Base_Table_Schema,
          QQPTFN as System_Base_Table_Name,
          QQPTMN as Base_Member_Name,
          QQC11 as Lock_Success,
          QQC12 as Unlock_Request,
          QQRCOD as Reason_Code,
          QVQTBL as Table_Name,
          QVQLIB as Table_Schema,
          QVPTBL as Base_Table_Name,
          QVPLIB as Base_Table_Schema,
          QQJNP as Join_Position,
          QQI6 as DataSpace_Number,
          QQC21 as Join_Method,
          QQC22 as Join_Type,
          QQC23 as Join_Operator,
          QVJFANO as Join_Fanout,
          QVFILES as Join_Table_Count,
          QVRCNT as Unique_Refresh_Counter
     FROM   UserLib/DBMONTable
     WHERE  QQRID=3005)
```

表 50. QQQ3005 - ロックされたテーブル

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻

表 50. QQQ3005 - ロックされたテーブル (続き)

ビュー列名	テーブル列名	説明
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
Lock_Success	QQC11	成功ロック標識 (Y/N)
Unlock_Request	QQC12	アンロック要求 (Y/N)

表 50. QQQ3005 - ロックされたテーブル (続き)

ビュー列名	テーブル列名	説明
Reason_Code	QQRCOD	理由コード • L1 - *ALL を指定した UNION またはロック保持を指定した *CS • L2 - *ALL を指定した DISTINCT またはロック保持を指定した *CS • L3 - *ALL を指定した複写キーがないか、ロック保持を指定した *CS • L4 - *ALL を指定した一時ファイルまたはロック保持を指定した *CS • L5 - *ALL を指定したシステム・テーブルまたはロック保持を指定した *CS • L6 - *ALL を指定した 2000 バイトを超える順序付けまたはロック保持を指定した *CS • L9 - 不明 • LA - *ALL を指定したユーザー定義テーブル関数またはロック保持を指定した *CS
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積

表 50. QQQ3005 - ロックされたテーブル (続き)

ビュー列名	テーブル列名	説明
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター

データベース・モニター・ビュー 3006 - アクセス・プラン再作成

データベース・モニター QQQ3006 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3006 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQRCOD as Reason_Code,
          QQC21 as SubCode,
          QVRCNT as Unique_Refresh_Counter,
          QQTIM1 as Last_Access_Plan_Rebuild_Timestamp,
          QQC11 as Reoptimization_Done,
          QVC22 as Previous_Reason_Code,
          QVC23 as Previous_SubCode
   FROM   UserLib/DBMONTable
   WHERE  QQRID=3006)
```

表 51. QQQ3006 - アクセス・プラン再作成

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前

表 51. QQQ3006 - アクセス・プラン再作成 (続き)

ビュー列名	テーブル列名	説明
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号

表 51. QQQ3006 - アクセス・プラン再作成 (続き)

ビュー列名	テーブル列名	説明
Reason_Code	QQRCOD	アクセス・プランの再作成の理由コード • A1 - テーブルまたはメンバーが、アクセス・プランが最後に構築された時に参照されたものと同一ではありません。これらが異なる理由として、以下が考えられます。 – オブジェクトが削除されて、再作成された。 – オブジェクトが保管されて、復元された。 – ライブラリー・リストが変更された。 – オブジェクトの名前が変更された。 – オブジェクトが移動された。 – オブジェクトが別のオブジェクトに上書きされた。 – これは、照会を含んでいるオブジェクトが復元された後の、この照会の最初の実行です。 • A2 - 再使用可能オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用不可 ODP を使用することを選択しました。 • A3 - 再使用不可オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用可能 ODP を使用することを選択しました。 • A4 - 最後にアクセス・プランが構築されてから、変更されたテーブル内の行数が 10% を超えました。 • A5 - 照会内のテーブルの 1 つについて新規索引が存在します。 • A6 - このアクセス・プランに使用した索引は、もはや存在しないか、またはもはや有効ではありません。 • A7 - i5/OS 照会プログラムでは、システム・プログラミング変更のため、アクセス・プランを再作成する必要があります。 • A8 - 現行ジョブの CCSID が、最後にアクセス・プランを作成したジョブの CCSID と異なっています。 • A9 - 現行ジョブでの下記の値の 1 つ以上が、このアクセス・プランを最後に作成したジョブでの値と異なっています。 – 日付形式 – 日付区切り記号 – 時刻形式 – 時刻区切り記号

表 51. QQQ3006 - アクセス・プラン再作成 (続き)

ビュー列名	テーブル列名	説明
Reason_Code (続き)	QQRCOD	• AA - 指定した分類順序テーブルが、このアクセス・プランを作成した時に使用した分類順序テーブルと異なっています。 • AB - 記憶域プールが変更されたか、または CHGQRYA コマンドの DEGREE パラメーターが変更されました。 • AC - システム機能 DB2 多重システムがインストールされたか、または除去されました。 • AD - 等級 (degree) 照会属性の値が変更されました。 • AE - ビューが、高水準言語によってオープンされたか、またはビューがマテリアライズされています。 • AF - 順序オブジェクトまたはユーザー定義タイプか関数が、アクセス・プランで参照されたのと同一のオブジェクトではありません。あるいは、アクセス・プランの生成に使用された SQL パスが現行の SQL パスと異なります。 • B0 - 指定したオプションが、照会オプション・ファイルの結果として変更されました。 • B1 - 現行ジョブで異なるコミットメント制御レベルで、アクセス・プランが生成されました。 • B2 - 以前のアクセス・プランと異なる静的カーソル応答セット・サイズで、アクセス・プランが生成されました。 • B3 - これが、作成後の最初の照会の実行であるので、照会が再最適化されました。つまり、実際の実パラメーター・マーカー値を使用した最初の実行です。 • B4 - 参照制約または検査制約が変更されたので、照会は再最適化されました。 • B5 - MQT が変更されたので、照会は最適化されました。
SubCode	QQC21	アクセス・プラン再作成の理由コードが A7 の場合、この 2 バイトの 16 進数値は、再作成が強制された特定の理由を識別します。
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Last_Access_Plan_Rebuild_Timestamp	QQTIM1	最後のアクセス・プラン再作成のタイム・スタンプ
Reoptimization_Done	QQC11	このプランで必要な最適化 • Y - はい、プランは本当に最適化されました。 • N - いいえ、REOPTIMIZE_ACCESS_PLAN パラメーター値の QAQQINI オプションのためにプランは最適化されませんでした。
Previous_Reason_Code	QVC22	直前の理由コード
Previous_SubCode	QVC23	直前の理由サブコード

データベース・モニター・ビュー 3007 - タイムアウトになった最適化プログラム

データベース・モニター QQQ3007 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3007 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQTLN as System_Table_Schema,
          QQTFN as System_Table_Name,
          QQTMN as Member_Name,
          QQPTLN as System_Base_Table_Schema,
          QQPTFN as System_Base_Table_Name,
          QQPTMN as Base_Member_Name,
          QQ1000 as Index_Names,
          QQC11 as Optimizer_Timed_Out,
          QQC301 as Reason_Codes,
          QVQTBL as Table_Name,
          QVQLIB as Table_Schema,
          QVPTBL as Base_Table_Name,
          QVPLIB as Base_Table_Schema,
          QQJNP as Join_Position,
          QQI6 as DataSpace_Number,
          QQC21 as Join_Method,
          QQC22 as Join_Type,
          QQC23 as Join_Operator,
          QVJFANO as Join_Fanout,
          QVFILES as Join_Table_Count,
          QVRCNT as Unique_Refresh_Counter,
          QQIDXNL as Index_Names_2
     FROM   UserLib/DBMONTable
     WHERE  QQRID=3007)
```

表 52. QQQ3007 - タイムアウトになった最適化プログラム

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード

表 52. QQQ3007 - タイムアウトになった最適化プログラム (続き)

ビュー列名	テーブル列名	説明
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名

表 52. *QQQ3007 - タイムアウトになった最適化プログラム (続き)*

ビュー列名	テーブル列名	説明
Index_Names	QQ1000	使用されていない索引名と理由コード。 1. アクセス・パスが有効な状態ではありませんでした。システムはアクセス・パスを無効にしました。 2. アクセス・パスが有効な状態ではありませんでした。ユーザーはアクセス・パスが再作成されるよう要求しました。 3. アクセス・パスは、一時アクセス・パス (ライブラリー QTEMP にある) であり、照会されるファイルとして指定されませんでした。 4. 最適化プログラムによって判別された、このアクセス・パスを使用するコストは、選択されたアクセス方式と関連したコストより高いものでした。 5. アクセス・パスのキーは、順序付け/グループ化基準に指定されたフィールドと一致しませんでした。分散ファイルの照会に関しては、ALWCPYDTA(*YES または *NO) が指定されているときにアクセス・パスが使用される場合、アクセス・パスのキーが順序付けフィールドと正確に一致しなければなりません。 6. アクセス・パスのキーは、結合基準に指定されたフィールドと一致しませんでした。 7. このアクセス・パスの使用は、ファイルからレコードを読み取るときの遅延を最小限に抑えません。ユーザーは、ファイルからレコードを読み取るときの遅延を最小限に抑えるよう要求しました。 8. アクセス・パスには静的選択/除外の選択基準が含まれているため、結合照会の 2 次ファイルに使用することはできません。照会の結合タイプは、2 次ファイルに対するアクセス・パスの選択/除外の使用を許可しません。 9. ファイルにはレコード ID 選択が含まれています。照会の結合タイプは、一時アクセス・パスがレコード ID 選択を処理するために作成されるよう強制します。 10. 指定されたユーザーは、照会の 10 進データ・エラーを無視します。これにより、永続アクセス・パスの使用は許可されなくなります。

表 52. QQQ3007 - タイムアウトになった最適化プログラム (続き)

ビュー列名	テーブル列名	説明
Index_Names (続き)	QQ1000	• 11. アクセス・パスには、照会の選択と互換性のない静的選択 /除外の選択基準が含まれます。 • 12. アクセス・パスには、照会の選択との互換性が判別できない、静的選択 /除外の選択基準が含まれます。互換性の処理中に、選択/除外基準か照会選択のどちらかが複雑になりすぎました。 • 13. アクセス・パスには、挿入または更新中に照会によって変更される場合のある 1 つ以上のキーが含まれます。 • 14. アクセス・パスは、他のプロセスの非コミット作業単位で削除されているか、作成されています。 • 15. アクセス・パスのキーは、順序付け/グループ化基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。 • 16. アクセス・パスのキーは、結合基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。 • 17. アクセス・パスの左端のキーは、選択基準に指定された任意のフィールドと一致しませんでした。このアクセス・パスを使用するコストが選択されたアクセス方式と関連するコストより高くなるため、キーの行の位置決めは実行できません。 • 18. アクセス・パスの左端のキーは、選択基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。このアクセス・パスを使用するコストが選択されたアクセス方式と関連するコストより高くなるため、キーの行の位置決めは実行できません。 • 19. 結合照会の 2 次ファイルが選択/除外の論理ファイルであるため、アクセス・パスを使用することはできません。結合タイプは、2 次ファイルと関連する選択/除外のアクセス・パスが使用されるか、動的な場合には、アクセス・パスがシステムによって作成されることを要求します。
Optimizer_Timed_Out	QQC11	タイムアウトになった最適化プログラム (Y/N)
Reason_Codes	QQC301	タイムアウト期限が切れた索引で使用された、固有理由コードのリスト (各索引には、対応する理由コードが関連付けられています)
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号

表 52. *QQQ3007 - タイムアウトになった最適化プログラム (続き)*

ビュー列名	テーブル列名	説明
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Index_Names_2	QQ1000L	リストが QQ1000 にフィットしない場合の索引名。それ以外の場合は、NULL に設定します。

データベース・モニター・ビュー 3008 - 副照会処理

データベース・モニター QQQ3008 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3008 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
```

```
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QQI1 as Original_QDT_Count,
          QQI2 as Merged_QDT_Count,
          QQI3 as Final_QDT_Count,
          QVRCNT as Unique_Refresh_Counter
   FROM   UserLib/DBMONTable
   WHERE  QQRID=3008)
```

表 53. QQQ3008 - 副照会処理

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Original_QDT_Count	QQI1	QDT の元の番号
Merged_QDT_Count	QQI2	QDT のマージ回数
Final_QDT_Count	QQI3	QDT の最終番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター

データベース・モニター・ビュー 3010 - ホスト変数と ODP 実施

データベース・モニター QQQ3010 の SQL 論理ビュー形式を表示します。

```
| Create View QQQ3010 as
|   (SELECT QQRID as Row_ID,
|           QQTIME as Time_Created,
|           QQJFLD as Join_Column,
|           QQRDBN as Relational_Database_Name,
|           QQSYS as System_Name,
|           QQJOB as Job_Name,
|           QQUSER as Job_User,
|           QQJNUM as Job_Number,
|           QQI9 as Thread_ID,
|           QQUCNT as Unique_Count,
|           QQI5 as Unqiue_Refresh_Counter2,
|           QQUDEF as User_Defined,
|           QQC11 as ODP_Implementation,
|           QQC12 as Host_Variable_Implementation,
```

```
|           QQ1000 as Host_Variable_Values,
|           QVRCNT as Unique_Refresh_Counter,
|           QQDBCLOB1 as DBCLOB_CCSID ,
|           QQI7 as DBCLOB_Length,
|           QQI8 as SQUNQCT
|     FROM  UserLib/DBMONTable
|     WHERE QQRID=3010)
```

| *表 54. QQQ3010 - ホスト変数と ODP 実施*

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
Unqiue_Refresh_Counter2	QQI5	固有最新表示カウンター
User_Defined	QQUDEF	ユーザー定義列
ODP_Implementation	QQC11	ODP 実施 • R - 再使用可能 ODP • N - 再使用不可 ODP • ' ' - 列は使用しません
Host_Variable_Implementation	QQC12	ホスト変数の実施 • I - インターフェースが提供した値 (ISV) • V - ホスト変数を定数として処理 (V2) • U - テーブル管理行の位置決め (UP)
Host_Variable_Values	QQ1000	ホスト変数値
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
DBCLOB_CCSID	QQDBCLOB1	DBCLOB CCSID 1200 フィールドのホスト変数値
DBCLOB_Length	QQI7	DBCLOB 列のホスト変数の長さ。
SQUNQCT	QQINT05	ODP はないが、ホスト変数に入れて渡されるステートメントを一意的に識別するために使用される固有カウント。 QQUCNT が 0 であり、ステートメントがホスト変数に渡される場合、この値は非ゼロです。一例として CALL ステートメントが挙げられます。

データベース・モニター・ビュー 3014 - 総称照会情報

データベース・モニター QQQ3014 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3014 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
```

```
QQUSER as Job_User,
QQJNUM as Job_Number,
QQI9 as Thread_ID,
QQUCNT as Unique_Count,
QQUDEF as User_Defined,
QQQDTN as Unique_SubSelect_Number,
QQQDTL as SubSelect_Nested_Level,
QQMATN as Materialized_View_Subselect_Number,
QQMATL as Materialized_View_Nested_Level,
QVP15E as Materialized_View_Union_Level,
QVP15A as Decomposed_Subselect_Number,
QVP15B as Total_Number_Decomposed_SubSelects,
QVP15C as Decomposed_SubSelect_Reason_Code,
QVP15D as Starting_Decomposed_SubSelect,
QQREST as Estimated_Rows_Selected,
QQEPT as Estimated_Processing_Time,
QQI1 as Open_Time,
QQORDG as Has_Ordering,
QQGRPG as Has_Grouping,
QQJNG as Has_Join,
QQC22 as Join_Type,
QQUNIN as Has_Union,
QQSUBQ as Has_Subquery,
QWC1F as Has_Scalar_Subselect,
QQHSTV as Has_Host_Variables,
QQRCDS as Has_Row_Selection,
QQC11 as Query_Governor_Enabled,
QQC12 as Stopped_By_Query_Governor,
QQC101 as Open_Id,
QQC102 as Query_Options_Library,
QQC103 as Query_Options_Table_Name,
QQC13 as Early_Exit,
QVRCNT as Unique_Refresh_Counter,
QQI5 as Optimizer_Time,
QQTIM1 as Access_Plan_Timestamp,
QVC11 as Ordering_Implementation,
QVC12 as Grouping_Implementation,
QVC13 as Join_Implementation,
QVC14 as Has_Distinct,
QVC15 as Is_Distributed,
QVC3001 as Distributed_Nodes,
QVC105 as NLSS_Table,
QVC106 as NLSS_Library,
QVC16 as ALWCPYDATA,
QVC21 as Access_Plan_Reason_Code,
QVC22 as Access_Plan_Reason_SubCode,
QVC3002 as Summary,
QWC16 as Last_Union_Subselect,
QVP154 as Query_PoolSize,
QVP155 as Query_PoolID,
QQI2 as Query_Time_Limit,
QVC81 as Parallel_Degree,
QQI3 as Max_Number_of_Tasks,
QVC17 as Apply_CHGQRYA_Remote,
QVC82 as Async_Job_Usage,
QVC18 as Force_Join_Order_Indicator,
QVC19 as Print_Debug_Messages,
QVC1A as Parameter_Marker_Conversion,
QQI4 as UDF_Time_Limit,
QVC1283 as Optimizer_Limitations,
QVC1E as Reoptimize_Requested,
QVC87 as Optimize_All_Indexes,
QQC14 as Has_Final_Decomposed_QDT,
QQC15 as Is_Final_Decomposed_QDT,
QQC18 as Read_Trigger,
QQC81 as Star_Join,
SUBSTR(QVC23,1,1) as Optimization_Goal,
```

```
        SUBSTR(QVC24,1,1) as VE_Diagram_Type,
        SUBSTR(QVC24,2,1) as Ignore_Like_Redunant_Shifts,
        QQC23 as Union_QDT,
        QQC21 as Unicode_Normalization,
        QVP153 as Pool_Fair_Share,
        QQC82 as Force_Join_Order_Requested,
        QVP152 as Force_Join_Order_Dataspace1,
        QQI6 as No_Parameter_Marker_Reason_Code,
        QVP151 as Hash_Join_Reason_Code,
        QQI7 as MQT_Refresh_Age,
        SUBSTR(QVC42,1,1) as MQT_Usage,
        QVC43 as SQE_NotUsed_Reason_Code,
        QVP156 as Estimated_IO_Count,
        QVP157 as Estimated_Processing_Cost,
        QVP158 as Estimated_CPU_Cost,
        QVP159 as Estimated_IO_Cost,
        SUBSTR(QVC44,1,1) as Has_Implicit_Numeric_Conversion,
        QVCTIM as Accumulated_Est_Process_Time,
        QQINT01 as Query_Gov_Storage_Limit,
        QQINT02 as Estimated_Storage,
        QQINT03 as Adjusted_Temp_Storage,
        QQINT04 as Original_Cost_Estimate,
        QQI8 as Parallel_Degree_Percentage
FROM    UserLib/DBMONTable
WHERE   QQRID=3014)
```

表 55. QQQ3014 - 総称照会情報

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Open_Time	QQI1	カーソルのオープンに費やした時間 (ミリ秒単位)

表 55. QQQ3014 - 総称照会情報 (続き)

ビュー列名	テーブル列名	説明
Has_Ordering	QQORDG	順序付け (Y/N)
Has_Grouping	QQGRPG	グループ化 (Y/N)
Has_Join	QQJNG	結合照会 (Y/N)
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Has_Union	QQUNIN	合併 (UNION) 照会 (Y/N)
Has_Subquery	QQSUBQ	副照会 (Y/N)
Has_Scalar_Subselect	QWC1F	スカラー副選択 (Y/N)
Has_Host_Variables	QQHSTV	ホスト変数 (Y/N)
Has_Row_Selection	QQRCDS	行選択 (Y/N)
Query_Governor_Enabled	QQC11	照会管理プログラムが使用可能 (Y/N)
Stopped_By_Query_Governor	QQC12	照会管理プログラムが照会を停止した (Y/N)
Open_Id	QQC101	照会オープン ID
Query_Options_Library	QQC102	照会オプション・ライブラリー名
Query_Options_Table_Name	QQC103	照会オプション・ファイル名
Early_Exit	QQC13	照会早期終了値
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Optimizer_Time	QQI5	最適化プログラムで経過した時間 (ミリ秒単位)
Access_Plan_Timestamp	QQTIM1	アクセス・プラン再作成のタイム・スタンプ、アクセス・プランが最後に再作成された時刻
Ordering_Implementation	QVC11	順序付けのインプリメンテーション。次の値が指定可能です。 • I - 索引 • S - 分類
Grouping_Implementation	QVC12	グループ化の実施。次の値が指定可能です。 • I - 索引 • H - ハッシュ・グループ
Join_Implementation	QVC13	結合の実施。 次の値が指定可能です。 • N - ネスト・ループ結合 • H - ハッシュ結合 • C - ネスト・ループとハッシュの組み合わせ
Has_Distinct	QVC14	DISTINCT 照会 (Y/N)
Is_Distributed	QVC15	分散照会 (Y/N)
Distributed_Nodes	QVC3001	分散ノード
NLSS_Table	QVC105	分類順序テーブル
NLSS_Library	QVC106	分類順序ライブラリー
ALWCPYDATA	QVC16	ALWCPYDTA 設定
Access_Plan_Reason_Code	QVC21	アクセス・プランの再作成の理由コード

表 55. QQQ3014 - 総称照会情報 (続き)

ビュー列名	テーブル列名	説明
Access_Plan_Reason_SubCode	QVC22	アクセス・プラン再作成の理由のサブコード
Summary	QVC3002	照会実施の要約。照会対象のテーブルごとに、データ・スペース数値と索引名を表示します。
Last_Union_Subselect	QWC16	合併 (UNION) の最後のパーツ (最後の QDT) (Y/N)
Query_PoolSize	QVP154	プール・サイズ
Query_PoolID	QVP155	プール ID
Query_Time_Limit	QQI2	照会時間制限
Parallel_Degree	QVC81	並列度 • *SAME - 現行の設定を使用します。 • *NONE - 並列処理を許可しません。 • *I/O - 任意の数のタスクを入出力処理で使用できます。 SMP 並列処理は許可されません。 • *OPTIMIZE - 最適化プログラムが、入出力または SMP 並列処理のいずれかで使用するタスクの数を選択します。 • *MAX - 最適化プログラムが、入出力または SMP 並列処理のいずれかを選択します。 • *SYSVAL - 現行のシステム値を使って照会を処理します。 • *ANY - *I/O と同じ意味です。 • *NBRTASKS - SMP 並列処理のタスク数は QVTASKN 列で指定します。
Max_Number_of_Tasks	QQI3	タスクの最大数
Apply_CHGQRYA_Remote	QVC17	CHGQRYA をリモートで適用 (Y/N)
Async_Job_Usage	QVC82	非同期ジョブの使用法 • *SAME - 現行の設定を使用します。 • *DIST - 非同期ジョブは、分散テーブルが関係した照会に使用することができます。 • *LOCAL - 非同期ジョブは、ローカル・テーブルのみが関係した照会に使用することができます。 • *ANY - 非同期ジョブは、すべてのデータベース照会に使用することができます。 • *NONE - 非同期ジョブは許可されていません。
Force_Join_Order_Indicator	QVC18	結合順序の強制 (Y/N)
Print_Debug_Messages	QVC19	デバッグ・メッセージの印刷 (Y/N)
Parameter_Marker_Conversion	QVC1A	パラメーター・マーカーの変換 (Y/N)
UDF_Time_Limit	QQI4	ユーザー定義関数の時間制限
Optimizer_Limitations	QVC1283	最適化プログラムの制限。 次の値が指定可能です。 • *PERCENT、およびパーセント値を記述した 2 バイト整数。 • *MAX_NUMBER_OF_RECORDS、および最大行数を記述した整数値。

表 55. QQQ3014 - 総称照会情報 (続き)

ビュー列名	テーブル列名	説明
Reoptimize_Requested		要求されたアクセス・プランの再最適化。次の値が指定可能です。 • O - 絶対必要な時にのみアクセス・プランの再最適化をします。主観的な理由で再最適化しないようにします。 • Y - はい、アクセス・プランを強制的に再最適化します。 • N - いいえ、最適化プログラムが必要であると判断するのでない限り、アクセス・プランを再最適化しません。主観的な理由で再最適化できます。
Optimize_All_Indexes		要求されたすべての索引の最適化 • *SAME - 現行の設定を使用します。 • *YES - すべての索引を検査する • *NO - 最適化プログラムにタイムアウトを許可する • *TIMEOUT - 最適化プログラムにタイムアウトを強制する
Has_Final_Decomposed_QDT	QQC14	作成された最終分析済み QDT 標識 (Y/N)
Is_Final_Decomposed_QDT	QQC15	これは最終分析済み QDT 標識です。(Y/N)
Read_Trigger	QQC18	ファイルの 1 つは読み取りトリガーを含んでいます。(Y/N)
Star_Join	QQC81	要求された星形結合最適化。 • *NO - 星形結合最適化は行われません。 • *COST - 最適化プログラムは星形結合最適化に何らかの EVI を使用できるかを判別します。 • *FORCE - 最適化プログラムは星形結合最適化に使用できるあらゆる EVI を追加します。
Optimization_Goal	QVC23	Byte 1 = 最適化ゴール。次の値が指定可能です。 • F - 最初の入出力、照会を最適化して、行で埋まっている最初の表示画面をできるだけ早く戻します。 • A - すべての入出力、 照会を最適化して、すべての行をできるだけ早く戻します。
VE_Diagram_Type	QVC24	Byte 1 = Visual Explain ダイアグラムのタイプ。次の値が指定可能です。 • D - 詳細 • B - 基本
Ignore_Like_Redunant_Shifts	QVC24	Byte 2 - LIKE の冗長シフトを無視します。次の値が指定可能です。 • O - 最適化、Query 最適化プログラムは、どの冗長シフトを無視するかを判別します。 • A - すべて、すべての冗長シフトは無視されます。
Union_QDT	QQC23	Byte 1 = この QDT はビュー内に含まれている UNION の一部です。(Y/N) Byte 2 = この QDT はビュー内に含まれている UNION の最後の副選択です。(Y/N)
Unicode_Normalization	QQC21	要求されたユニコード・データ正規化 (Y/N)

表 55. QQQ3014 - 総称照会情報 (続き)

ビュー列名	テーブル列名	説明
Pool_Fair_Share	QVP153	最適化プログラムによって決定されたプール・サイズの公平な共用
Force_Join_Order_Requested	QQC82	要求された結合順序の強制。次の値が指定可能です。 • *NO - 最適化プログラムは結合ファイルの再順序付けを許可されました。 • *YES - 最適化プログラムは最適化処理の一部として、結合ファイルの再順序付けを許可されませんでした。 • *SQL - 最適化プログラムは SQL JOIN 構文を使用する照会の結合順序を強制しただけです。 • *PRIMARY - 最適化プログラムは、結合のための 1 次ダイヤルを強制するのに必要であっただけです。
Force_Join_Order_Dataspace1	QVP152	Force_Join_Order_Indicator が *PRIMARY の場合、1 次ダイヤルを強制。
No_Parameter_Marker_Reason_Code	QQI6	パラメーター・マーカー変換が実行されなかった原因を示す理由コード。 1. 関数の引数がリテラルでなければならない 2. LOCALTIME または LOCALTIMESTAMP 3. 算術式の期間リテラル 4. WHERE 文節を伴わない UPDATE 照会 5. BLOB リテラル 6. 値を伴う UPDATE または INSERT の特殊レジスター 7. CASE の 結果式 8. GROUP BY 式 9. ESCAPE 文字 10. 二重の負の値 -(-1) 11. リテラル、パラメーター・マーカー、および NULL の混合を伴う INSERT または UPDATE 12. リテラル、およびパラメーター・マーカーの混合を伴う UPDATE 13. NULL 値および式を含む VALUES を伴う INSERT 14. 式のリストを伴う UPDATE 99. QAQQINI によって使用不可にされたパラメーター・マーカーの変換
Hash_Join_Reason_Code	QVP151	ハッシュ結合が使用されなかった理由コード。
MQT_Refresh_Age	QQI7	MATERIALIZED_QUERY_TABLE_REFRESH_AGE 期間の値。QAQQINI パラメーター値が *ANY なら、タイム・スタンプ期間は 99999999999999 になります。

表 55. QQQ3014 - 総称照会情報 (続き)

ビュー列名	テーブル列名	説明
MQT_Usage	QVC42,1,1	バイト 1 - MATERIALIZED_QUERY_TABLE_USAGE の値。次の値が指定可能です。 • N - *NONE - 照会の最適化および実装でマテリアライズ照会表は使用されません • A - *ALL - ユーザー保守。最新表示据え置き照会表を使用できます。 • U - *USER - ユーザー保守のマテリアライズ照会表だけを使用できます。
SQE_NotUsed_Reason_Code	QVC43	SQE 未使用の理由コード。次の値が指定可能です。 • LF - 照会の定義に指定された DDS 論理ファイル • DK - 照会されたテーブルで派生キー付きの索引または選択/省略が見つかりました • NF - 照会のテーブルが多すぎます • NS - SQL 照会以外、または SQL インターフェースを介して実行されない照会 • DF - 照会の分散テーブル • DK - 照会されたテーブルで派生キー付きの索引または選択/省略が見つかりました • RT - 照会されたテーブルで定義されている読み取りトリガー • PD - 照会のプログラム記述ファイル • WC - WHERE CURRENT OF パーティション・テーブル • IO - 単純 INSERT 照会 • CV - ビュー・ステートメントの作成
Estimated_IO_Count	QVP156	Estimated I/O count
Estimated_Processing_Cost	QVP157	見積処理コスト (ミリ秒単位)
Estimated_CPU_Cost	QVP158	見積 CPU コスト (ミリ秒単位)
Estimated_IO_Cost	QVP159	見積 I/O コスト (ミリ秒単位)
Has_Implicit_Numeric_Conversion	QVC44	バイト 1: 暗黙的な数値変換 (Y/N)
Accumulated_Est_Process_Time	QVCTIM	累積見積処理時間 (秒単位)。これは、すべての副選択を通じて累積された見積処理時間です。
Query_Gov_Storage_Limit	QQINT01	指定された照会管理プログラムのストレージ限界 (メガバイト単位)
Estimated_Storage	QQINT02	使用される見積一時ストレージ (メガバイト単位)。これは、オリジナルの見積もりです。
Adjusted_Temp_Storage	QQINT03	使用される調整済み一時ストレージ (調整されたメガバイト単位)。この値は、任意の一時索引および一時表の作成に要した実際の時間およびストレージを累積します。 CQE によってのみ設定されます。
Original_Cost_Estimate	QQINT04	CQE Query 最適化プログラムによって決定されたオリジナル・コスト見積もり。CQE によってのみ設定されます。
Parallel_Degree_Percentage	QQ18	Parallel_Degree *OPTIMIZE および *MAX 上で指定されるパーセント。

データベース・モニター・ビュー 3015 - 統計情報

データベース・モニター QQQ3015 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3015 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QQTLN as System_Table_Schema,
          QQTFN as System_Table_Name,
          QQTMN as Member_Name,
          QQPTLN as System_Base_Table_Schema,
          QQPTFN as System_Base_Table_Name,
          QQPTMN as Base_Member_Name,
          QVQTBL as Table_Name,
          QVQLIB as Table_Schema,
          QVPTBL as Base_Table_Name,
          QVPLIB as Base_Table_Schema,
          QQNTNM as NLSS_Table,
          QQNLNM as NLSS_Library,
          QQC11 as Statistic_Status,
          QQI2 as Statistic_Importance,
          QQ1000 as Statistic_Columns,
          QVC1000 as Statistic_ID
   FROM   UserLib/DBMONTable
   WHERE  QQRID=3015)
```

表 56. QQQ3015 - 統計情報

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)

表 56. QQQ3015 - 統計情報 (続き)

ビュー列名	テーブル列名	説明
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名
System_Base_Table_Name	QQPTFN	照会された基礎テーブルの名前
Base_Member_Name	QQPTMN	基礎テーブルのメンバー名
Table_Name	QVQTBL	照会されたテーブル、長い名前
Table_Schema	QVQLIB	照会されたテーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
NLSS_Table	QQNTNM	NLSS テーブル
NLSS_Library	QQNLNM	NLSS ライブラリー
Statistic_Status	QQC11	統計状況。次の値が指定可能です。 • 'N' - 統計はありません。 • 'S' - 失効した統計。 • ' ' - 不明。
Statistic_Importance	QQI2	この統計の重要度
Statistic_Columns	QQ1000	推奨統計の列
Statistic_ID	QVC1000	統計 ID

データベース・モニター・ビュー 3018 - STRDBMON/ENDDBMON

データベース・モニター QQQ3018 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3018 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQC11 as Monitored_Job_type,
```

```
|           QQC12 as Monitor_Command,
|           QQC301 as Monitor_Job_Information,
|           QQ1000L as STRDBMON_Command_Text,
|           QQC101 as Monitor_ID,
|           QQC102 as Version_Release_Mod,
|       QQC103 as Group_PTF
|     FROM   UserLib/DBMONTable
|     WHERE  QQRID=3018)
```

表 57. QQQ3018 - STRDBMON/ENDDBMON

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Monitored_Job_type	QQC11	監視されるジョブのタイプ • C - 現行 • J - ジョブ名 • A - すべて
Monitor_Command	QQC12	コマンド・タイプ • S - STRDBMON • E - ENDDBMON
Monitor_Job_Information	QQC301	監視されるジョブの情報 • * - 現行ジョブ • ジョブ番号/ジョブ名 • *ALL - すべてのジョブ
STRDBMON_Command_Text	QQ1000L	STRDBMON コマンド・テキスト。
Monitor_ID	QQC101	モニター ID
Version_Release_Mod	QQC102	バージョン・リリースおよびモディフィケーション・レベル
Group_PTF	QQC103	インストール済みのグループ PTF 番号およびレベル。例えば、「SF99601 3」は、Database Group リリース V6R1M0、バージョン 3 がインストールされたことを示しています。

データベース・モニター・ビュー 3019 - 検索された行

データベース・モニター QQQ3019 の SQL 論理ビュー形式を表示します。

```
| Create View QQQ3019 as
|   (SELECT QQRID as Row_ID,
|           QQTIME as Time_Created,
|           QQJFLD as Join_Column,
|           QQRDBN as Relational_Database_Name,
|           QQSYS as System_Name,
|           QQJOB as Job_Name,
|           QQUSER as Job_User,
|           QQJNUM as Job_Number,
```

```
|            QQI9 as Thread_ID,
|            QQUCNT as Unique_Count,
|            QQUDEF as User_Defined,
|            QQQDTN as Unique_SubSelect_Number,
|            QQQDTL as SubSelect_Nested_Level,
|            QQMATN as Materialized_View_Subselect_Number,
|            QQMATL as Materialized_View_Nested_Level,
|            QVP15E as Materialized_View_Union_Level,
|            QVP15A as Decomposed_Subselect_Number,
|            QVP15B as Total_Number_Decomposed_SubSelects,
|            QVP15C as Decomposed_SubSelect_Reason_Code,
|            QVP15D as Starting_Decomposed_SubSelect,
|            QQI1 as CPU_Time_to_Return_All_Rows,
|            QQI2 as Clock_Time_to_Return_All_Rows,
|            QQI3 as Number_Synchronous_Database_Reads,
|            QQI4 as Number_Synchronous_Database_Writes,
|            QQI5 as Number_Asynchronous_Database_Reads,
|            QQI6 as Number_Asynchronous_Database_Writes,
|            QVP151 as Number_Page_Faults,
|            QQI7 as Number_Rows_Returned,
|            QQI8 as Number_of_Calls_for_Returned_Rows,
|            QVP15F as Number_of_Times_Statement_was_Run,
|            QQINT03 as Temporary_Storage,
|            QQC11 as DBMON_Temp_Result_Reused,
|            QQC21 as DBMON_Temp_Reused_RC,
|            QQINT01 as DBMON_Temp_Reuse_Count,
|            QQIA as Skip_Lock_Row_Count,
|            QQINT05 as Skip_Lock_Row_Runs,
|            QQINT06 as Skip_Lock_On_Runs,
|            QQF1 as Adjusted_Average_Run_Time,
|            QVRCNT as Unique_Refresh_Counter
|     FROM   UserLib/DBMONTable
|     WHERE  QQRID=3019)
```

表 58. QQQ3019 - 検索された行

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数

表 58. QQQ3019 - 検索された行 (続き)

ビュー列名	テーブル列名	説明
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
CPU_Time_to_Return_All_Rows	QQI1	すべての行を戻すための CPU 時間 (ミリ秒単位)
Clock_Time_to_Return_All_Rows	QQI2	すべての行を戻すためのクロック時間 (ミリ秒単位)
Number_Synchronous_Database_Reads	QQI3	データベース同期読み取りの回数
Number_Synchronous_Database_Writes	QQI4	データベース同期書き込みの回数
Number_Asynchronous_Database_Reads	QQI5	データベース非同期読み取りの回数
Number_Asynchronous_Database_Writes	QQI6	データベース非同期書き込みの回数
Number_Page_Faults	QVP151	ページ不在の数
Number_Rows_Returned	QQI7	戻された行数
Number_of_Calls_for_Returned_Rows	QQI8	戻された行を検索するための呼び出し回数
Number_of_Times_Statement_was_Run	QVP15F	このステートメントが実行された回数。 NULL の場合、ステートメントは 1 度実行されたことを意味します。
Temporary_Storage	QQINT03	使用された一時記憶域の量。
DBMON_Temp_Result_Reused	QQC11	DBMON 一時結果が再利用されたかどうかを示します (Y/N)。
DBMON_Temp_Reused_RC	QQC21	DBMON 一時結果が再利用された理由コード。
DBMON_Temp_Reuse_Count	QQINT01	DBMON 一時結果が再利用された回数。
Skip_Lock_Row_Count	QQIA	スキップされたロック済みの行数。
Skip_Lock_Row_Runs	QQINT05	一部の行がスキップされた実行回数。
Skip_Lock_On_Runs	QQINT06	スキップ・ロックがアクティブだった実行回数。
Adjusted_Average_Run_Time	QQF1	標準を大きく逸脱した個別の実行を含まないように調整された、照会の平均ランタイム。単位は、マイクロ秒。
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター

データベース・モニター・ビュー 3020 - 推奨索引 (SQE)

データベース・モニター QQQ3020 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3020 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
```

```
        QVP15D as Starting_Decomposed_SubSelect,
        QQTLN as System_Table_Schema,
        QQTFN as System_Table_Name,
        QQTMN as Member_Name,
        QQPTLN as System_Base_Table_Schema,
        QQPTFN as System_Base_Table_Name,
        QQPTMN as Base_Member_Name,
        QVPLIB as Base_Table_Schema,
        QVPTBL as Base_Table_Name,
        QQTOTR as Table_Total_Rows,
        QQEPT as Estimated_Processing_Time,
        QQIDXA as Index_is_Advised,
        QQIDXD as Index_Advised_Columns_Short_List,
        QQ1000L as Index_Advised_Columns_Long_List,
        QQI1 as Number_of_Advised_Columns,
        QQI2 as Number_of_Advised_Primary_Columns,
        QQRCOD as Reason_Code,
        QVRCNT as Unique_Refresh_Counter,
        QVC1F as Type_of_Index_Advised,
        QQNTNM as NLSS_Table,
        QQNLNM as NLSS_Library
FROM    UserLib/DBMONTable
WHERE   QQRID=3020)
```

表 59. QQQ3020 - 推奨索引 (SQE)

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
System_Table_Schema	QQTLN	照会されたテーブルのスキーマ
System_Table_Name	QQTFN	照会されたテーブルの名前
Member_Name	QQTMN	照会されたテーブルのメンバー名
System_Base_Table_Schema	QQPTLN	基礎テーブルのスキーマ名

表 59. QQQ3020 - 推奨索引 (SQE) (続き)

ビュー列名	テーブル列名	説明
System_Base_Table_Name	QQPTFN	照会されたテーブルの基礎テーブル名
Base_Member_Name	QQPTMN	基礎テーブルのメンバー
Base_Table_Schema	QVPLIB	基礎テーブルのスキーマ、長い名前
Base_Table_Name	QVPTBL	基礎テーブル、長い名前
Table_Total_Rows	QQTOTR	テーブルの行数
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Index_is_Advised	QQIDXA	推奨索引 (Y/N)
Index_Advised_Columns_Short_List	QQIDXD	推奨索引の列、最初の 1000 バイト
Index_Advised_Columns_Long_List	QQ1000L	推奨索引の列
Number_of_Advised_Columns	QQI1	推奨索引の数
Number_of_Advised_Primary_Columns	QQI2	索引走査のキー位置決めを使用する推奨列の数
Reason_Code	QQRCOD	理由コード • I1 - 行選択 • I2 - 順序付け/グループ化 • I3 - 行選択と順序付け/グループ化 • I4 - ネスト・ループ結合 • I5 - ビットマップ処理を使用した行選択
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Type_of_Index_Advised	QVC1F	推奨索引のタイプ次の値が指定可能です。 • B - 基数索引 • E - コード化ベクトル索引
NLSS_Table	QQNTNM	分類順序テーブル
NLSS_Library	QQNLNM	分類順序ライブラリー

関連資料

107 ページの『Query 最適化プログラムの索引アドバイザー』

Query 最適化プログラムは、照会内の行選択を分析して、デフォルト値に基づいて、永続索引の作成がパフォーマンスを向上するかどうかを判別します。永続索引が有利になる可能性があると最適化プログラムが判別した場合には、示された索引の作成に必要なキー列を戻します。

データベース・モニター・ビュー 3021 - 作成されたビットマップ

データベース・モニター QQQ3021 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3021 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
```

```
        QQMATN as Materialized_View_Subselect_Number,
        QQMATL as Materialized_View_Nested_Level,
        QVP15E as Materialized_View_Union_Level,
        QVP15A as Decomposed_Subselect_Number,
        QVP15B as Total_Number_Decomposed_SubSelects,
        QVP15C as Decomposed_SubSelect_Reason_Code,
        QVP15D as Starting_Decomposed_SubSelect,
        QVRCNT as Unique_Refresh_Counter,
        QVPARPF as Parallel_Prefetch,
        QVPARPL as Parallel_PreLoad,
        QVPARD as Parallel_Degree_Requested,
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI6 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QQI2 as Bitmap_Size,
        QVP151 as Bitmap_Count,
        QVC3001 as Bitmap_IDs,
        QQINT03 as StorageEst
FROM    UserLib/DBMONTable
WHERE   QQRID=3021)
```

表 60. QQQ3021 - 作成されたビットマップ

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号

表 60. *QQQ3021 - 作成されたビットマップ (続き)*

ビュー列名	テーブル列名	説明
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Bitmap_Size	QQI2	ビットマップ・サイズ
Bitmap_Count	QVP151	作成されたビットマップの数
Bitmap_IDs	QVC3001	内部ビットマップ ID

表 60. QQQ3021 - 作成されたビットマップ (続き)

ビュー列名	テーブル列名	説明
StorageEst	QQINT03	一時索引の作成に使用される一時ストレージの見積量 (メガバイト単位)。

データベース・モニター・ビュー 3022 - ビットマップ・マージ

データベース・モニター QQQ3022 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3022 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QVRCNT as Unique_Refresh_Counter,
          QVPARPF as Parallel_Prefetch,
          QVPARPL as Parallel_PreLoad,
          QVPARD as Parallel_Degree_Requested,
          QVPARU as Parallel_Degree_Used,
          QVPARRC as Parallel_Degree_Reason_Code,
          QQEPT as Estimated_Processing_Time,
          QVCTIM as Estimated_Cumulative_Time,
          QQREST as Estimated_Rows_Selected,
          QQAJN as Estimated_Join_Rows,
          QQJNP as Join_Position,
          QQI6 as DataSpace_Number,
          QQC21 as Join_Method,
          QQC22 as Join_Type,
          QQC23 as Join_Operator,
          QVJFANO as Join_Fanout,
          QVFILES as Join_Table_Count,
          QQI2 as Bitmap_Size,
          QVC101 as Bitmap_ID,
          QVC3001 as Bitmaps_Merged,
          QQINT03 as StorageEst
   FROM   UserLib/DBMONTable
   WHERE  QQRID=3022)
```

表 61. QQQ3022 - ビットマップ・マージ

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前

表 61. *QQQ3022 - ビットマップ・マージ (続き)*

ビュー列名	テーブル列名	説明
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合

表 61. QQQ3022 - ビットマップ・マージ (続き)

ビュー列名	テーブル列名	説明
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Bitmap_Size	QQI2	ビットマップ・サイズ
Bitmap_ID	QVC101	内部ビットマップ ID
Bitmaps_Merged	QVC3001	組み合わされたビットマップ ID
StorageEst	QQINT03	最終ビットマップの作成に使用される一時ストレージの見積量 (メガバイト単位)。CQE によってのみ設定されます。

データベース・モニター・ビュー 3023 - 作成された一時ハッシュ・テーブル

データベース・モニター QQQ3023 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3023 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QVRCNT as Unique_Refresh_Counter,
          QVPARPF as Parallel_Prefetch,
          QVPARPL as Parallel_PreLoad,
          QVPARD as Parallel_Degree_Requested,
```

```
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI6 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QVC1F as HashTable_ReasonCode,
        QQI2 as HashTable_Entries,
        QQI3 as HashTable_Size,
        QQI4 as HashTable_Row_Size,
        QQI5 as HashTable_Key_Size,
        QQIA as HashTable_Element_Size,
        QQI7 as HashTable_PoolSize,
        QQI8 as HashTable_PoolID,
        QVC101 as HashTable_Name,
        QVC102 as HashTable_Library,
        QVC3001 as HashTable_Columns,
        QQINT03 as StorageEst
FROM    UserLib/DBMONTable
WHERE   QQRID=3023)
```

表 62. QQQ3023 - 作成された一時ハッシュ・テーブル

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)

表 62. QQQ3023 - 作成された一時ハッシュ・テーブル (続き)

ビュー列名	テーブル列名	説明
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
HashTable_ReasonCode	QVC1F	ハッシュ・テーブル理由コード • J - ハッシュ結合のために作成された • G - ハッシュ・グループ化のために作成された
HashTable_Entries	QQI2	ハッシュ・テーブル項目
HashTable_Size	QQI3	ハッシュ・テーブル・サイズ

表 62. QQQ3023 - 作成された一時ハッシュ・テーブル (続き)

ビュー列名	テーブル列名	説明
HashTable_Row_Size	QQI4	ハッシュ・テーブル行サイズ
HashTable_Key_Size	QQI5	ハッシュ・テーブル・キー・サイズ
HashTable_Element_Size	QQIA	ハッシュ・テーブル要素サイズ
HashTable_PoolSize	QQI7	ハッシュ・テーブル・プール・サイズ
HashTable_PoolID	QQI8	ハッシュ・テーブル・プール ID
HashTable_Name	QVC101	ハッシュ・テーブル内部名
HashTable_Library	QVC102	ハッシュ・テーブル・ライブラリー
HashTable_Columns	QVC3001	ハッシュ・テーブルの作成で使用された列
StorageEst	QQINT03	ハッシュ・テーブルの作成に使用される一時ストレージの見積量 (メガバイト単位)。CQE によってのみ設定されます。

データベース・モニター・ビュー 3025 - 特殊処理

データベース・モニター QQQ3025 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3025 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QVRCNT as Unique_Refresh_Counter,
          QVPARPF as Parallel_Prefetch,
          QVPARPL as Parallel_PreLoad,
          QVPARD as Parallel_Degree_Requested,
          QVPARU as Parallel_Degree_Used,
          QVPARRC as Parallel_Degree_Reason_Code,
          QQEPT as Estimated_Processing_Time,
          QVCTIM as Estimated_Cumulative_Time,
          QQREST as Estimated_Rows_Selected
   FROM   UserLib/DBMONTable
   WHERE  QQRID=3025)
```

表 63. QQQ3025 - 特殊処理

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)

表 63. QQQ3025 - 特殊処理 (続き)

ビュー列名	テーブル列名	説明
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり

データベース・モニター・ビュー 3026 - セット操作

データベース・モニター QQQ3026 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3026 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
```

```
        QVP15E as Materialized_View_Union_Level,
        QVP15A as Decomposed_Subselect_Number,
        QVP15B as Total_Number_Decomposed_SubSelects,
        QVP15C as Decomposed_SubSelect_Reason_Code,
        QVP15D as Starting_Decomposed_SubSelect,
        QVRCNT as Unique_Refresh_Counter,
        QVPARPF as Parallel_Prefetch,
        QVPARPL as Parallel_PreLoad,
        QVPARD as Parallel_Degree_Requested,
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQC11 as Union_Type,
        QVFILES as Join_Table_Count,
        QQUNIN as Has_Union,
        QWC16 as Last_Union_Subselect,
        QQC23 as Set_in_a_View,
        QQC22 as Set_Operator
FROM    UserLib/DBMONTable
WHERE   QQRID=3026)
```

表 64. QQQ3026 - セット操作

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (Y/N)
Parallel_Degree_Requested	QVPARD	要求された並列度
Parallel_Degree_Used	QVPARU	使用された並列度

表 64. QQQ3026 - セット操作 (続き)

ビュー列名	テーブル列名	説明
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Union_Type	QQC11	共用体のタイプ。次の値が指定可能です。 • A - 共用体すべて • U - 共用体
Join_Table_Count	QVFILES	照会されたテーブルの数
Has_Union	QQUNIN	共用体 (UNION) 副選択 (Y/N)
Last_Union_Subselect	QWC16	これは照会に対し最後の副選択か、唯一の副選択です。(Y/N)
Set_in_a_View	QQC23	ビュー内のセット操作 • バイト 1/2 (Y/N): この副選択はビューに含まれる照会の一部であり、その中にはセット操作が含まれます (たとえば共用体 (Union))。 • バイト 2/2 (Y/N): これはビューに含まれる照会の最後の副選択です。
Set_Operator	QQC22	セット操作のタイプ。次の値が指定可能です。 • UU - 共用体 • UA - 共用体すべて • UR - 共用体 (再帰的) • EE - 除外 • EA - 除外すべて • II - 交差 • IA - 交差すべて

データベース・モニター・ビュー 3027 - 副照会マージ

データベース・モニター QQQ3027 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3027 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
```

```
        QVP15D as Starting_Decomposed_SubSelect,
        QVRCNT as Unique_Refresh_Counter,
        QVPARPF as Parallel_Prefetch,
        QVPARPL as Parallel_PreLoad,
        QVPARD as Parallel_Degree_Requested,
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI1 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QVP151 as Subselect_Number_of_Inner_Subquery,
        QVP152 as Subselect_Level_of_Inner_Subquery,
        QVP153 as Materialized_View_Subselect_Number_of_Inner,
        QVP154 as Materialized_View_Nested_Level_of_Inner,
        QVP155 as Materialized_View_Union_Level_of_Inner,
        QQC101 as Subquery_Operator,
        QVC21 as Subquery_Type,
        QQC11 as Has_Correlated_Columns,
        QVC3001 as Correlated_Columns
FROM    UserLib/DBMONTable
WHERE   QQRID=3027)
```

表 65. QQQ3027 - 副照会マージ

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	外部副照会の副選択番号
SubSelect_Nested_Level	QQQDTL	外部副照会の副選択レベル
Materialized_View_Subselect_Number	QQMATN	外部副照会のマテリアライズ・ビュー副選択番号
Materialized_View_Nested_Level	QQMATL	外部副照会のマテリアライズ・ビュー副選択レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号

表 65. QQQ3027 - 副照会マージ (続き)

ビュー列名	テーブル列名	説明
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置 - 使用可能な場合
DataSpace_Number	QQI6	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数
Subselect_Number_of_Inner_Subquery	QVP151	内部副照会の副選択番号
Subselect_Level_of_Inner_Subquery	QVP152	内部副照会の副選択レベル
Materialized_View_Subselect_Number_of_Inner	QVP153	内部副照会のマテリアライズ・ビュー副選択番号

表 65. QQQ3027 - 副照会マージ (続き)

ビュー列名	テーブル列名	説明
Materialized_View_Nested_Level_of_Inner	QVP154	内部副照会のマテリアライズ・ビュー副選択レベル
Materialized_View_Union_Level_of_Inner	QVP155	内部副照会のマテリアライズ・ビュー共用体レベル
Subquery_Operator	QQC101	副照会の演算子。 次の値が指定可能です。 • EQ - 等しい • NE - 等しくない • LE - より小か等しい • LT - より小 • GE - より大か等しい • GT - より大 • IN • LIKE • EXISTS • NOT - IN、LIKE、または EXISTS の前に指定可能
Subquery_Type	QVC21	副照会のタイプ。次の値が指定可能です。 • SQ - 副照会 • SS - スカラー副選択 • SU - 更新の設定
Has_Correlated_Columns	QQC11	相関列の存在 (Y/N)
Correlated_Columns	QVC3001	対応する QDT 数値を持った相関列のリスト

データベース・モニター・ビュー 3028 - グループ化

データベース・モニター QQQ3028 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3028 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QVRCNT as Unique_Refresh_Counter,
          QVPARPF as Parallel_Prefetch,
          QVPARPL as Parallel_PreLoad,
          QVPARD as Parallel_Degree_Requested,
          QVPARU as Parallel_Degree_Used,
          QVPARRC as Parallel_Degree_Reason_Code,
          QQEPT as Estimated_Processing_Time,
```

```
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQAJN as Estimated_Join_Rows,
        QQJNP as Join_Position,
        QQI1 as DataSpace_Number,
        QQC21 as Join_Method,
        QQC22 as Join_Type,
        QQC23 as Join_Operator,
        QVJFANO as Join_Fanout,
        QVFILES as Join_Table_Count,
        QQC11 as GroupBy_Implementation,
        QQC101 as GroupBy_Index_Name,
        QQC102 as GroupBy_Index_Library,
        QVINAM as GroupBy_Index_Long_Name,
        QVILIB as GroupBy_Index_Long_Library,
        QQC12 as Has_Having_Selection,
        QQC13 as Having_to_Where_Selection_Conversion,
        QQI2 as Estimated_Number_of_Groups,
        QQI3 as Average_Number_Rows_per_Group,
        QVC3001 as GroupBy_Columns,
        QVC3002 as MIN_Columns,
        QVC3003 as MAX_Columns,
        QVC3004 as SUM_Columns,
        QVC3005 as COUNT_Columns,
        QVC3006 as AVG_Columns,
        QVC3007 as STDDEV_Columns,
        QVC3008 as VAR_Columns
FROM    UserLib/DBMONTable
WHERE   QQRID=3028)
```

表 66. QQQ3028 - グループ化

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号

表 66. QQQ3028 - グループ化 (続き)

ビュー列名	テーブル列名	説明
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (使用された索引)
Parallel_Degree_Requested	QVPARD	要求された並列度 (使用された索引)
Parallel_Degree_Used	QVPARU	使用された並列度 (使用された索引)
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由 (使用された索引)
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Estimated_Join_Rows	QQAJN	結合された行数の見積もり
Join_Position	QQJNP	結合位置
DataSpace_Number	QQI1	データ・スペース番号
Join_Method	QQC21	結合方式 - 使用可能な場合 • NL - ネスト・ループ • MF - 選択付きネスト・ループ • HJ - ハッシュ結合
Join_Type	QQC22	結合タイプ - 使用可能な場合 • IN - 内部結合 • PO - 左方部分的外部結合 • EX - 例外結合
Join_Operator	QQC23	結合演算子 - 使用可能な場合 • EQ - 等しい • NE - 等しくない • GT - より大 • GE - より大か等しい • LT - より小 • LE - より小か等しい • CP - カルテシアン積
Join_Fanout	QVJFANO	結合ファンアウト。 次の値が指定可能です。 • N - 通常の結合。ファンアウトが許可されており、結合ファンアウトの各突き合わせ行が戻されます。 • D - 特殊ファンアウト。結合ファンアウトが許可されているが、結合ファンアウト行は戻されません。 • U - 固有ファンアウト。結合ファンアウトは許可されていません。結合ファンアウトを実行すると、エラーが生じます。
Join_Table_Count	QVFILES	結合されたテーブルの数

表 66. QQQ3028 - グループ化 (続き)

ビュー列名	テーブル列名	説明
GroupBy_Implementation	QQC11	Group by (グループ化) の実装 • ' ' - グループ化を使用しない • I - 索引 • H - ハッシュ
GroupBy_Index_Name	QQC101	グループ化で使用された索引 (または制約)
GroupBy_Index_Library	QQC102	グループ化で使用された索引のライブラリー
GroupBy_Index_Long_Name	QVINAM	グループ化で使用された索引 (または制約)、長い名前
GroupBy_Index_Long_Library	QVILIB	グループ化で使用された索引 (または制約) のライブラリー、長い名前
Has_Having_Selection	QQC12	Having 選択の存在 (Y/N)
Having_to_Where_Selection_Conversion	QQC13	Having から Where への変換 (Y/N)
Estimated_Number_of_Groups	QQI2	グループ数の見積もり
Average_Number_Rows_per_Group	QQI3	各グループの平均行数
GroupBy_Columns	QVC3001	グループ化する列
MIN_Columns	QVC3002	MIN 列
MAX_Columns	QVC3003	MAX 列
SUM_Columns	QVC3004	SUM 列
COUNT_Columns	QVC3005	COUNT 列
AVG_Columns	QVC3006	AVG 列
STDDEV_Columns	QVC3007	STDDEV 列
VAR_Columns	QVC3008	VAR 列

データベース・モニター・ビュー 3030 - マテリアライズ照会表

データベース・モニター QQQ3030 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3030 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
          QQSYS as System_Name,
          QQJOB as Job_Name,
          QQUSER as Job_User,
          QQJNUM as Job_Number,
          QQI9 as Thread_ID,
          QQUCNT as Unique_Count,
          QQUDEF as User_Defined,
          QQQDTN as Unique_SubSelect_Number,
          QQQDTL as SubSelect_Nested_Level,
          QQMATN as Materialized_View_Subselect_Number,
          QQMATL as Materialized_View_Nested_Level,
          QVP15E as Materialized_View_Union_Level,
          QVP15A as Decomposed_Subselect_Number,
          QVP15B as Total_Number_Decomposed_SubSelects,
          QVP15C as Decomposed_SubSelect_Reason_Code,
          QVP15D as Starting_Decomposed_SubSelect,
          QVRCNT as Unique_Refresh_Counter,
```

```
        QQ1000 as Materialized_Query_Tables,
        QQC301 as MQT_Reason_Codes
FROM    UserLib/DBMONTable
WHERE   QQRID=3030)
```

表 67. QQQ3030 - マテリアライズ照会表

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Materialized_Query_Tables	QQ1000	調べられたマテリアライズ照会表、およびそれが使用された理由または使用されなかった理由: • 0 - マテリアライズ照会表が使用されました • 1 - マテリアライズ照会表を使用するためのコストとして最適化プログラムの判別した値が、選択されたインプリメンテーションに関連するコストよりも高くなっています。 • 2 - マテリアライズ照会で指定された結合には、照会との互換性がありませんでした。 • 3 - マテリアライズ照会表に、照会の中で一致しなかった述部があります。 • 4 - マテリアライズ照会表の中で指定されているグループ化には、照会で指定されているグループ化との互換性がありません。

表 67. *QQQ3030 - マテリアライズ照会表 (続き)*

ビュー列名	テーブル列名	説明
Materialized_Query_Tables (続き)		• 5 - マテリアライズ照会表の選択リストに含まれていない列が照会の中で指定されました。 • 6 - マテリアライズ照会表の照会に、Query 最適化プログラムでサポートされていない機能が含まれています。 • 7 - マテリアライズ照会表で DISABLE QUERY OPTIMIZATION 文節が指定されました。 • 8 - マテリアライズ照会表の中で指定されている順序付けには、照会で指定されている順序付けとの互換性がありません。 • 9 - マテリアライズ照会表のマッチング・アルゴリズムでサポートされていない機能が、照会に含まれています。 • 10 - この照会では、マテリアライズ照会表を使用できません。 • 11 - このマテリアライズ照会表の最新表示未実行期間が、MATERIALIZED_QUERY_TABLE_REFRESH_AGE QAQQINI オプションで指定された期間を超えました。 • 12 - マテリアライズ照会表のコミット・レベルが、照会で指定されているコミット・レベルより低くなっています。 • 13 - マテリアライズ照会表の中で指定されている DISTINCT には、照会で指定されている DISTINCT との互換性がありません。 • 14 - マテリアライズ照会表の FETCH FOR FIRST n ROWS 文節には、照会との互換性がありません。 • 15 - マテリアライズ照会表の作成に使用された QAQQINI オプションには、この照会を実行するために使用される QAQQINI オプションとの互換性がありません。 • 16 - マテリアライズ照会表は使用できません。 • 17 - マテリアライズ照会表で指定された共用体 (UNION) は、照会と互換性がありません。 • 18 - マテリアライズ照会表で指定された定数は、照会に指定されたホスト変数値と互換性がありません。
MQT_Reason_Codes	QQC301	マテリアライズ照会表によって使用される固有の理由コードのリスト (マテリアライズ照会表ごとに 1 つの理由コードが対応)
QVRCNT	QVRCNT	固有最新表示カウンター

データベース・モニター・ビュー 3031 - 再帰的な共通テーブル式

データベース・モニター QQQ3031 の SQL 論理ビュー形式を表示します。

```
Create View QQQ3031 as
  (SELECT QQRID as Row_ID,
          QQTIME as Time_Created,
          QQJFLD as Join_Column,
          QQRDBN as Relational_Database_Name,
```

```
        QQSYS as System_Name,
        QQJOB as Job_Name,
        QQUSER as Job_User,
        QQJNUM as Job_Number,
        QQI9 as Thread_ID,
        QQUCNT as Unique_Count,
        QQUDEF as User_Defined,
        QQQDTN as Unique_SubSelect_Number,
        QQQDTL as SubSelect_Nested_Level,
        QQMATN as Materialized_View_Subselect_Number,
        QQMATL as Materialized_View_Nested_Level,
        QVP15E as Materialized_View_Union_Level,
        QVP15A as Decomposed_Subselect_Number,
        QVP15B as Total_Number_Decomposed_SubSelects,
        QVP15C as Decomposed_SubSelect_Reason_Code,
        QVP15D as Starting_Decomposed_SubSelect,
        QVRCNT as Unique_Refresh_Counter,
        QVPARPF as Parallel_Prefetch,
        QVPARPL as Parallel_PreLoad,
        QVPARD as Parallel_Degree_Requested,
        QVPARU as Parallel_Degree_Used,
        QVPARRC as Parallel_Degree_Reason_Code,
        QQEPT as Estimated_Processing_Time,
        QVCTIM as Estimated_Cumulative_Time,
        QQREST as Estimated_Rows_Selected,
        QQC11 as Recursive_Query_Cycle_Check,
        QQC15 as Recursive_Query_Search_Option,
        QQI2 as Number_of_Recursive_Values
FROM    UserLib/DBMONTable
WHERE   QQRID=3031)
```

表 68. QQQ3031 - 再帰的な共通テーブル式

ビュー列名	テーブル列名	説明
Row_ID	QQRID	行 ID
Time_Created	QQTIME	行が作成された時刻
Join_Column	QQJFLD	結合列 (ジョブごとに固有)
Relational_Database_Name	QQRDBN	リレーショナル・データベースの名前
System_Name	QQSYS	システムの名前
Job_Name	QQJOB	ジョブ名
Job_User	QQUSER	ジョブ・ユーザー
Job_Number	QQJNUM	ジョブ番号
Thread_ID	QQI9	スレッド識別コード
Unique_Count	QQUCNT	固有カウント (照会ごとに固有)
User_Defined	QQUDEF	ユーザー定義列
Unique_SubSelect_Number	QQQDTN	固有の副選択番号
SubSelect_Nested_Level	QQQDTL	副選択のネスト・レベル
Materialized_View_Subselect_Number	QQMATN	マテリアライズ・ビュー副選択の番号
Materialized_View_Nested_Level	QQMATL	マテリアライズ・ビューのネスト・レベル
Materialized_View_Union_Level	QVP15E	マテリアライズ・ビューの共用体レベル
Decomposed_Subselect_Number	QVP15A	すべての分析済み副選択で固有な、分析済み照会副選択番号
Total_Number_Decomposed_SubSelects	QVP15B	分析済み副選択の合計数
Decomposed_SubSelect_Reason_Code	QVP15C	分析済み照会副選択の理由コード
Starting_Decomposed_SubSelect	QVP15D	最初の分析済み副選択の、分析済み照会副選択番号

表 68. QQQ3031 - 再帰的な共通テーブル式 (続き)

ビュー列名	テーブル列名	説明
Unique_Refresh_Counter	QVRCNT	固有最新表示カウンター
Parallel_Prefetch	QVPARPF	並列プリフェッチ (Y/N)
Parallel_PreLoad	QVPARPL	並列プリロード (Y/N)
Parallel_Degree_Requested	QVPARD	要求された並列度
Parallel_Degree_Used	QVPARU	使用された並列度
Parallel_Degree_Reason_Code	QVPARRC	並列処理が制限された理由
Estimated_Processing_Time	QQEPT	見積処理時間 (秒単位)
Estimated_Cumulative_Time	QVCTIM	見積累積時間 (秒単位)
Estimated_Rows_Selected	QQREST	選択された行数の見積もり
Recursive_Query_Cycle_Check	QQC11	CYCLE オプション: • Y - 巡回データの検査あり • N - 巡回データの検査なし
Recursive_Query_Search_Option	QQC15	SEARCH オプション: • N - 指定なし • D - 深さ優先 • B - 幅優先
Number_of_Recursive_Values	QQI2	再帰を実装するために待ち行列に入れられる値の数。CYCLE および SEARCH の各オプションで必要な値を含みます。

メモリー常駐のデータベース・モニター: DDS

以下の DDS ステートメントは、メモリー常駐のデータベース・モニター物理ファイルおよび論理ファイルの作成に使用されます。

外部テーブル記述 (QAQQQRYI) - SQL 情報の要約行

QAQQQRYI - SQL 情報の要約行を表示します。

表 69. QAQQQRYI - SQL 情報の要約行

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQJOB	ジョブ名
QQUSER	ジョブ・ユーザー
QQJNUM	ジョブ番号
QQTHID	スレッド ID
QQUDEF	ユーザー定義列
QQPLIB	プログラムまたはパッケージが入っているライブラリーの名前
QQCNAM	カーソル名
QQPNAM	パッケージの名前または現行 SQL ステートメントが入っているプログラムの名前
QQSNAM	該当する場合、SQL ステートメントのステートメント名
QQCNT	ステートメントの使用量カウント

表 69. QAQQQRYI - SQL 情報の要約行 (続き)

列名	説明
QQAVGT	平均実行時間 (ミリ秒)
QQMINT	最小実行時間 (ミリ秒)
QQMAXT	最大実行時間 (ミリ秒)
QQOPNT	最も費用のかかる実行のオープン時間 (ミリ秒)
QQFETT	最も費用のかかる実行の取り出し時間 (ミリ秒)
QQCLST	最も費用のかかる実行のクローズ時間 (ミリ秒)
QQOTHT	最も費用のかかる実行のその他の時間 (ミリ秒)
QQLTU	最後に使用された時刻ステートメント
QQMETU	使用された最も費用のかかる時間
QQAPRT	アクセス・プラン再作成時間
QQFULO	全オープンの回数
QQPSUO	疑似オープンの回数
QQTOTR	テーブル内の合計行数 (結合なし)
QQRROW	戻された結果行数
QQRROW	ステートメント関数 S - 選択U - 更新I - 挿入 D - 削除 L - データ定義言語 O - その他
QQSTOP	ステートメント操作 • AD - 記述子の割り振り • AF - 変更関数 • AL - テーブル更新 • AP - プロシージャーの変更 • AQ - 順序の変更 • CA - 呼び出し • CC - コレクションの作成 • CD - タイプの作成 • CF - 関数の作成 • CG - トリガーの作成 • CI - 索引の作成 • CL - クローズ • CM - コミット • CN - 接続 • CO - 注記 • CP - プロシージャーの作成 • CQ - 順序の作成 • CS - 別名/同義語の作成 • CT - テーブルの作成 • CV - ビューの作成 • DA - 記述子の割り振り解除 • DE - 記述 • DI - 切断 • DL - 削除

表 69. QAQQQRYI - SQL 情報の要約行 (続き)

列名	説明
QQSTOP (続く)	• DM - パラメーター・マーカーの記述 • DP - プロシージャーの宣言 • DR - 除去 • DT - 記述テーブル • EI - 即時実行 • EX - 実行 • FE - 取り出し • FL - 空きロケーター • GR - 権限付与 • GS - 記述子の取得 • HC - ハード・クローズ • HL - ロケーターの保持 • IN - 挿入 • JR - 再使用されるサーバー・ジョブ • LK - ロック • LO - ラベル付け • MT - テキストをさらに表示 (V5R4 で減価) • OP - オープン • PD - 作成および記述 • PR - 準備 • RB - 保管ポイントへのロールバック • RE - 解放 • RF - テーブルの最新表示 • RG - 再シグナル • RO - ロールバック • RS - 保管ポイントの解放 • RT - テーブルの名前変更 • RV - 取り消し • SA - 保管ポイント • SC - 接続の設定 • SD - 記述子の設定 • SE - 暗号化パスワードの設定 • SN - セッション・ユーザーの設定 • SI - 選択 • SO - CURRENT DEGREE (現行影響度) の設定 • SP - パスの設定 • SR - 結果セットの設定 • SS - 現行スキーマの設定 • ST - トランザクションの設定 • SV - 変数の設定 • UP - 更新 • VI - 値 • X0 - 不明ステートメント

表 69. QAQQQRYI - SQL 情報の要約行 (続き)

列名	説明
QQSTOP (続く)	• X1 - 不明ステートメント • X2 - DRDA (AS) 不明ステートメント • X3 - 不明ステートメント • X9 - 内部エラー • XA - X/Open API • ZD - ホスト・サーバー専用アクティビティー
QQODPI	ODP 実施 R - 再使用可能 ODP (ISV) N - 再使用不可 ODP (V2)
QQHVI	ホスト変数の実施 I - インターフェースが提供した値 (ISV) V - ホスト変数を定数として処理 (V2) U - テーブル管理行の位置決め (UP)
QQAPR	アクセス・プラン再作成 • **A1** - テーブルまたはメンバーが、アクセス・プランが最後に構築された時に参照されたものと同一ではありません。 これらが異なる理由として、以下が考えられます。 – オブジェクトが削除されて、再作成された。 – オブジェクトが保管されて、復元された。 – ライブラリー・リストが変更された。 – オブジェクトの名前が変更された。 – オブジェクトが移動された。 – オブジェクトが別のオブジェクトに上書きされた。 – これは、照会を含んでいるオブジェクトが復元された後の、この照会の最初の実行です。 • **A2** - 再使用可能オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用不可 ODP を使用することを選択しました。 • **A3** - 再使用不可オープン・データ・パス (ODP) を使用するためにアクセス・プランが構築されました。最適化プログラムは、この呼び出しに対して再使用可能 ODP を使用することを選択しました。 • **A4** - 最後にアクセス・プランが構築されてから、変更されたテーブル内の行数が 10% を超えました。 • **A5** - 照会内のテーブルの 1 つについて新規索引が存在します。 • **A6** - このアクセス・プランに使用した索引は、もはや存在しないか、またはもはや有効ではありません。 • **A7** - i5/OS 照会プログラムでは、システム・プログラミング変更のため、アクセス・プランを再作成する必要があります。 • **A8** - 現行ジョブの CCSID が、最後にアクセス・プランを作成したジョブの CCSID と異なっています。 • **A9** - 現行ジョブでの下記の値の 1 つ以上が、このアクセス・プランを最後に作成したジョブでの値と異なっています。 – 日付形式 – 日付区切り記号 – 時刻形式 – 時刻区切り記号

表 69. QAQQQRYI - SQL 情報の要約行 (続き)

列名	説明
QQAPR (続く)	• **AA** - 指定した分類順序テーブルが、このアクセス・プランを作成した時に使用した分類順序テーブルと異なっています。 • **AB** - 記憶域プールが変更されたか、または CHGQRYA コマンドの DEGREE パラメーターが変更されました。 • **AC** - システム機能 DB2 多重システムがインストールされたか、または除去されました。 • **AD** - 等級 (degree) 照会属性の値が変更されました。 • **AE** - ビューが、高水準言語によってオープンされたか、またはビューがマテリアライズされています。 • **AF** - 順序オブジェクトまたはユーザー定義タイプか関数が、アクセス・プランで参照されたのと同一のオブジェクトではありません。あるいは、アクセス・プランの生成に使用された SQL パスが現行の SQL パスと異なります。 • **B0** - 指定したオプションが、照会オプション・ファイル QAQQINI の結果として変更されました。 • **B1** - 現行ジョブで異なるコミットメント制御レベルで、アクセス・プランが生成されました。 • **B2** - 以前のアクセス・プランと異なる静的カーソル応答セット・サイズで、アクセス・プランが生成されました。 • **B3** - これが、作成後の最初の照会の実行であるので、照会が再最適化されました。つまり、実際の実パラメーター・マーカー値を使用した最初の実行です。 • **B4** - 参照制約または検査制約が変更されたので、照会は再最適化されました。 • **B5** - マテリアライズ照会表が変更されたので、照会は再最適化されました。
QQDACV	データ変換 • N - いいえ。 • 0 - 該当しません。 • 1 - 長さが一致しません。 • 2 - 数値タイプが一致しません。 • 3 - C ホスト変数が NUL で終了しています。 • 4 - ホスト変数または列が可変長で、その他は可変長ではありません。 • 5 - ホスト変数または列が可変長ではなく、その他は可変長です。 • 6 - ホスト変数または列が可変長で、その他は可変長ではありません。 • 7 - CCSID 変換。 • 8 - DRDA および NULL 可能、可変長、部分行の内容、派生式、または十分なホスト変数が指定されていないブロック化取り出し。 • 9 - 挿入のターゲット・テーブルが SQL テーブルではありません。 • 10 - ホスト変数が、取り出される TIME または TIMESTAMP 値を保持するには短すぎます。 • 11 - ホスト変数は DATE、TIME、または TIMESTAMP であり、取り出される値は文字ストリングです。 • 12 - 多すぎるホスト変数が指定され、レコードがブロック化されています。 • 13 - DRDA がブロック化された FETCH に使用され、INTO 文節に指定されたホスト変数の数は、選択リストにある結果値の数より少ないです。 • 14 - LOB ロケーターが使用され、コミットメント制御レベルは *ALL ではありません。
QQCTS	ステートメント・テーブル走査の使用量カウント
QQCIU	ステートメント索引の使用量カウント
QQCIC	ステートメント索引の作成カウント
QQCSO	ステートメント分類の使用量カウント
QQCTF	ステートメント一時テーブル・カウント

表 69. QAQQQRYI - SQL 情報の要約行 (続き)

列名	説明
QQCIA	ステートメント推奨索引カウント
QQCAPR	ステートメント・アクセス・プラン再作成カウント
QQARSS	平均結果セットのサイズ
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQC1000	予約済み

外部テーブル記述 (QAQQTEXT) - SQL ステートメントの要約行

QAQQTEXT - SQL ステートメントの要約行を表示します。

表 70. QAQQTEXT - SQL ステートメントの要約行

列名	説明
QQKEY	単一照会用の行を行 ID とリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQSTTX	ステートメント・テキスト
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3000) - 到着順

QAQQ3000 - 到着順の要約行を表示します。

表 71. QAQQ3000 - 到着順

列名	説明
QQKEY	単一照会用の行を行 ID とリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (ODT ごとに固有)
QQQDTL	QDT 副照会のネスト・レベル
QQMATN	マテリアライズ・ビューの QDT 番号

表 71. QAQQ3000 - 到着順 (続き)

列名	説明
QQMATL	マテリアライズ・ビューのネスト・レベル
QQTLN	ライブラリー
QQTFN	テーブル
QQPTLN	物理ライブラリー
QQPTFN	物理テーブル
QQTOTR	テーブル内の合計行数
QQREST	選択された行数の見積もり
QQAJN	結合された行数の見積もり
QQEPT	見積処理時間 (秒単位)
QQJNP	結合位置 - 使用可能な場合
QQJNDS	データ・スペース番号
QQJNMT	結合方式 - 使用可能な場合 NL - ネスト・ループ MF - 選択付きネスト・ループ HJ - ハッシュ結合
QQJNTY	結合タイプ - 使用可能な場合 IN - 内部結合 PO - 左方部分的外部結合 EX - 例外結合
QQJNOP	結合演算子 - 使用可能な場合 EQ - 等しい NE - 等しくない GT - より大 GE - より大か等しい LT - より小 LE - より小か等しい CP - カルテシアン積
QQDSS	データ・スペース選択 Y - はい N - いいえ
QQIDXA	推奨索引 Y - はい N - いいえ
QQRCOD	理由コード T1 - 索引がない。 T2 - 索引があるが、どれも使用できない。 T3 - 最適化プログラムが使用可能な索引についてテーブル走査を選択した。
QQLTLN	ライブラリー - 長
QQLTFN	テーブル - 長
QQLPTL	物理ライブラリー - 長

表 71. QAQQ3000 - 到着順 (続き)

列名	説明
QQLPTF	テーブル - 長
QQIDXD	推奨索引のキー列
QQC11	マテリアライズ照会表
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	索引走査のキー位置決めを使用する推奨キー列の数
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3001) - 既存の索引の使用

QAQQ3001 - 既存の索引の使用の要約行を表示します。

表 72. QQQ3001 - 既存の索引の使用

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQTLN	ライブラリー
QQTFN	テーブル
QQPTLN	物理ライブラリー
QQPTFN	物理テーブル
QQILNM	索引ライブラリー
QQIFNM	索引
QQTOTR	テーブル内の合計行数
QQREST	選択された行数の見積もり
QQFKEY	キー位置決めキーの数
QQKSEL	キー選択キーの数
QQAJN	結合位置 - 使用可能な場合
QQEPT	見積処理時間 (秒単位)
QQJNP	結合位置 - 使用可能な場合
QQJNDS	データ・スペース番号

表 72. QQQ3001 - 既存の索引の使用 (続き)

列名	説明
QQJNMT	結合方式 - 使用可能な場合 NL - ネスト・ループ MF - 選択付きネスト・ループ HJ - ハッシュ結合
QQJNTY	結合タイプ - 使用可能な場合 IN - 内部結合 PO - 左方部分的外部結合 EX - 例外結合
QQJNOP	結合演算子 - 使用可能な場合 EQ - 等しい NE - 等しくない GT - より大 GE - より大か等しい LT - より小 LE - より小か等しい CP - カルテシアン積
QQIDXK	索引走査のキー位置決めを使用する推奨キー列の数
QQKP	索引走査のキー位置決め Y - はい N - いいえ
QQKPN	キー位置決め列の数
QQKS	索引走査のキー選択 Y - はい N - いいえ
QQDSS	データ・スペース選択 Y - はい N - いいえ
QQIDXA	推奨索引 Y - はい N - いいえ
QQRCOD	理由コード I1 - 行選択 I2 - 順序付け/グループ化 I3 - 行選択と 順序付け/グループ化 I4 - ネスト・ループ結合 I5 - ビットマップ処理を使用した行選択
QQCST	制約標識 Y - はい N - いいえ

表 72. QQQ3001 - 既存の索引の使用 (続き)

列名	説明
QQCSTN	制約名
QQLTLN	ライブラリー - 長
QQLTFN	テーブル - 長
QQLPTL	物理ライブラリー - 長
QQLPTF	テーブル - 長
QQLILN	索引ライブラリー – 長
QQLIFN	索引 – 長
QQIDXD	推奨索引のキー列
QQC11	マテリアライズ照会表
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3002) - 作成された索引

QAQQ3002 - 作成された索引の要約行を表示します。

表 73. QQQ3002 - 作成された索引

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQTLN	ライブラリー
QQTFN	テーブル
QQPTLN	物理ライブラリー
QQPTFN	物理テーブル
QQILNM	索引ライブラリー
QQIFNM	索引
QQNTNM	NLSS テーブル
QQNLNM	NLSS ライブラリー
QQTOTR	テーブル内の合計行数
QQRIDX	作成された索引内の項目数
QQREST	選択された行数の見積もり

表 73. QQQ3002 - 作成された索引 (続き)

列名	説明
QQFKEY	索引走査のキー位置決めキーの数
QQKSEL	索引走査のキー選択キーの数
QQAJN	結合された行数の見積もり
QQJNP	結合位置 - 使用可能な場合
QQJNDS	データ・スペース番号
QQJNMT	結合方式 - 使用可能な場合 NL - ネスト・ループ MF - 選択付きネスト・ループ HJ - ハッシュ結合
QQJNTY	結合タイプ - 使用可能な場合 IN - 内部結合 PO - 左方部分的外部結合 EX - 例外結合
QQJNOP	結合演算子 - 使用可能な場合 EQ - 等しい NE - 等しくない GT - より大 GE - より大か等しい LT - より小 LE - より小か等しい CP - カルテシアン積
QQIDXK	索引走査のキー位置決めを使用する推奨キー列の数
QQEPT	見積処理時間 (秒単位)
QQKP	索引走査のキー位置決め Y - はい N - いいえ
QQKPN	索引走査のキー位置決め列の数
QQKS	索引走査のキー選択 Y - はい N - いいえ
QQDSS	データ・スペース選択 Y - はい N - いいえ
QQIDXA	推奨索引 Y - はい N - いいえ
QQCST	制約標識 Y - はい N - いいえ

表 73. QQQ3002 - 作成された索引 (続き)

列名	説明
QQCSTN	制約名
QQRCOD	理由コード I1 - 行選択 I2 - 順序付け/グループ化 I3 - 行選択と 順序付け/グループ化 I4 - ネスト・ループ結合 I5 - ビットマップ処理を使用した行選択
QQTTIM	索引作成時間
QQLTLN	ライブラリー - 長
QQLTFN	テーブル - 長
QQLPTL	物理ライブラリー - 長
QQLPTF	テーブル - 長
QQLILN	索引ライブラリー - 長
QQLIFN	索引 - 長
QQLNTN	NLSS テーブル - 長
QQLNLN	NLSS ライブラリー - 長
QQIDXD	推奨索引のキー列
QQCRTK	作成された索引のキー列
QQC11	マテリアライズ照会表
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3003) - 照会分類

QAQQ3003 - 照会分類の要約行を表示します。

表 74. QQQ3003 - 照会分類

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル

表 74. QQQ3003 - 照会分類 (続き)

列名	説明
QQTTIM	分類時間
QQRSS	選択または分類された行数
QQSIZ	分類スペースのサイズ
QQPSIZ	プール・サイズ
QQPID	プール ID
QQIBUF	内部分類バッファー長
QQEBUF	外部分類バッファー長
QQRCOD	理由コード **F1** - 照会には、2 つ以上のテーブルからのグループ化列 (Group By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからのグループ化列が含まれていたりします。 **F2** - 照会には、2 つ以上のテーブルからの順序付け列 (Order By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからの順序付け列が含まれていたりします。 **F3** - グループ化列と順序付けフィールドには、互換性がありません。 **F4** - DISTINCT が照会に対して指定されました。 **F5** - UNION が照会に対して指定されました。 **F6** - 照会は、分類を使用して実施しなければなりませんでした。順序付けでキーの長さが 2000 バイトを超えているかまたは 120 を超える列が指定されました。 **F7** - Query 最適化プログラムは、照会の結果を順序付けする索引ではなくて、分類を使用することを選択します。 **F8** - 指定した列選択を実行して入出力待ち時間を最小にします。 **FC** - 照会はグループ・フィールドを含み、照会で少なくとも 1 つの物理ファイルには読み取りトリガーがあります。
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3004) - 一時テーブル

QAQQ3004 - 一時テーブルの要約行を表示します。

表 75. QQQ3004 - 一時テーブル

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名

表 75. *QQQ3004 - 一時テーブル (続き)*

列名	説明
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQTLN	ライブラリー
QQTFN	テーブル
QQTTIM	一時テーブル作成時刻
QQTMPR	一時ファイル内の行数
QQRCOD	理由コード **F1** - 照会には、2 つ以上のテーブルからのグループ化列 (Group By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからのグループ化列が含まれていたりします。 **F2** - 照会には、2 つ以上のテーブルからの順序付け列 (Order By) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからの順序付け列が含まれていたりします。 **F3** - グループ化列と順序付けフィールドには、互換性がありません。 **F4** - DISTINCT が照会に対して指定されました。 **F5** - UNION が照会に対して指定されました。 **F6** - 照会は、分類を使用して実施しなければなりませんでした。順序付けでキーの長さが 2000 バイトを超えているかまたは 120 を超える列が指定されました。 **F7** - Query 最適化プログラムは、照会の結果を順序付けする索引ではなくて、分類を使用することを選択します。 **F8** - 指定した列選択を実行して入出力待ち時間を最小にします。 **F9** - Query 最適化プログラムは、アクセス・パスを使ってグループ化するのではなくて、ハッシュ・アルゴリズムを使用することを選択します。 **FA** - 照会には、一時ファイルを必要とする結合条件が含まれています。 **FB** - Query 最適化プログラムは、照会ごとに特定の相関グループをインプリメントするために、ランタイム一時ファイルを作成します。 **FC** - 照会はグループ・フィールドを含み、照会で少なくとも 1 つの物理ファイルには読み取りトリガーがあります。 **FD** - Query 最適化プログラムは静的カーソル要求のためにランタイム一時ファイルを作成します。 **H1** - 照会中のテーブルが結合論理ファイルであって、その結合タイプ (JDFTVAL) が照会に指定された結合タイプと一致しない場合。 **H2** - 論理テーブルに指定された形式が複数の基礎テーブルを参照しています。 **H3** - テーブルは、複合 SQL ビューであって、その SQL ビューの一時結果を必要としています。 **H4** - 更新可能照会の場合、副選択はこのテーブルの列で、更新中のいずれかの列と一致する列を参照します。 **H5** - 更新可能照会の場合、副選択は更新中のテーブルを基礎とした SQL ビューを参照します。 **H6** - 削除可能照会の場合、副選択は行の削除元テーブル、SQL ビュー、または行の削除元テーブルを基礎とした索引のいずれかを参照します。 **H7** - ユーザー定義テーブル関数がマテリアライズされました。

表 75. QQQ3004 - 一時テーブル (続き)

列名	説明
QQDFVL	デフォルト値が一時ファイルに表れる可能性がある Y - はい N - いいえ
QQLTLN	ライブラリー - 長
QQLTFN	テーブル - 長
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3007) - 最適化プログラム情報

QAQQ3007 - 最適化プログラム情報の要約行を表示します。

表 76. QQQ3007 - 最適化プログラム情報

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQTLN	ライブラリー
QQTFN	テーブル
QQPTLN	物理ライブラリー
QQPTFN	テーブル
QQTOUT	タイムアウトになった最適化プログラム Y - はい N - いいえ。
QQIRSN	理由コード
QQLTLN	ライブラリー - 長
QQLTFN	テーブル - 長
QQPTL	物理ライブラリー - 長
QQPTF	テーブル - 長
QQIDXN	索引名

表 76. QQQ3007 - 最適化プログラム情報 (続き)

列名	説明
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3008) - 副照会処理

QAQQ3008 - 副照会処理の要約行を表示します。

表 77. QQQ3008 - 副照会処理

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	QDT 番号 (QDT ごとに固有)
QQQDTL	RQDT 副照会ネスト・レベル・リレーショナル・データベース名
QQMATN	マテリアライズ・ビューの QDT 番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQORGQ	マテリアライズ・ビューの QDT 番号
QQMRGQ	マテリアライズ・ビューのネスト・レベル
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み
QQ1000	予約済み

外部テーブル記述 (QAQQ3010) - ホスト変数および ODP 実施

QAQQ3010 - ホスト変数および ODP 実装の要約行を表示します。

表 78. QQQ3010 - ホスト変数と ODP 実施

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻

表 78. QQQ3010 - ホスト変数と ODP 実施 (続き)

列名	説明
QQHVAR	ホスト変数値
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQI1	予約済み
QQI2	予約済み
QQC301	予約済み
QQC302	予約済み

外部テーブル記述 (QAQQ3030) - マテリアライズ照会表実装

QAQQ3030 - マテリアライズ照会表実装の要約行を表示します。

表 79. QQQ3030 - マテリアライズ照会表実装

列名	説明
QQKEY	単一照会用の行を互いにリンクするために使用した結合列 (照会ごとに固有)
QQTIME	行が作成された時刻
QQQDTN	固有の副選択番号
QQQDTL	副選択のネスト・レベル
QQMATN	マテリアライズ・ビュー副選択の番号
QQMATL	マテリアライズ・ビューのネスト・レベル
QQMRSN	マテリアライズ照会表によって使用される固有の理由コードのリスト (マテリアライズ照会表ごとに 1 つの理由コードが対応)
QQMQTS	MQT が使用されたかどうか、また使 用されなかった場合、なぜ使用されなかったかを示す理由コードと、調査された MQTのリスト。
QQC11	予約済み
QQC12	予約済み
QQC21	予約済み
QQC22	予約済み
QQCI1	予約済み
QQCI2	予約済み
QQC301	予約済み
QQC302	予約済み

Query 最適化プログラムのメッセージ解説

Query 最適化プログラムのメッセージ解説については、以下を参照してください。

Query 最適化パフォーマンス通知メッセージ

データベース・マネージャーによってジョブ・ログに入れられる通知メッセージを使用すれば、プログラム中の特定の SQL ステートメントの構造およびパフォーマンスを評価することができます。

これらのメッセージは、デバッグ・モードでの実行時に SQL プログラムまたは対話式 SQL について出されます。データベース・マネージャーは、必要に応じて以下のメッセージを出すことがあります。アンパーサンド変数 (&1, &X) は、メッセージがジョブ・ログに現れる際にはオブジェクト名または他の置換値が入れられる置き換え変数です。これらのメッセージは照会がどのように実行されたかのフィードバックとなるもので、場合によっては、照会の実行を速めるのに役立つ改善を示します。

メッセージにはメッセージ・ヘルプが含まれており、メッセージが出された原因、オブジェクト名参照、および可能なユーザー応答に関する情報が提供されます。

メッセージが送られる時点は、必ずしも関連機能が実行された時点を示すものではありません。いくつかのメッセージは照会実行の開始時に一緒に送られます。

CPI4321 - &18 &19 のアクセス・パスが作成された。	
メッセージ・テキスト:	&18 &19 のアクセス・パスが作成された。
原因テキスト:	理由コード &10 のため、ライブラリー &5 にある &18 &19 のメンバー &6 からレコードにアクセスするために一時アクセス・パスが作成されました。このプロセスに &11 分 &12 秒かかりました。アクセス・パスの作成には &15 項目が含まれています。アクセス・パスは、&16 の並列タスクを使用して作成されました。並列タスクの数がゼロの場合、並列処理が使用されなかったことを示しています。理由コードおよびその意味は次のとおりです。 1 - 指定された順序付け/グループ化基準を実行します。 2 - 指定された結合基準を実行します。 3 - 指定したレコード選択を実行して入出力待ち時間を最小にします。 アクセス・パスは、次のキー・フィールドを使用して作成されました。キー・フィールドおよび対応する順序 (ASCEND または DESCEND) が表示されます。 &17。 *MAP のキー・フィールドは、キー・フィールドが式 (派生したフィールド) であることを示します。 アクセス・パスは、ライブラリー &14 の順序テーブル &13 を使用して作成されました。 *N の順序テーブルは、アクセス・パスが順序テーブルを使用せずに作成されたことを示します。*I の順序テーブルは、テーブルがユーザーの使用できない、内部で派生したテーブルであったことを示します。 ライブラリー &5 の &18 &19 が論理ファイルである場合、アクセス・パスはライブラリー &8 の物理ファイル &7 のメンバー &9 に対して作成されます。 *QUERY または *N で始まるファイル名は、アクセス・パスが一時ファイルに対して作成されたことを示します。
リカバリー・テキスト:	この照会が頻繁に実行される場合、パフォーマンス上の理由でこの定義に類似したアクセス・パス (索引) を作成することができます。順序テーブルが *N でない場合、ライブラリー &14 の順序テーブル &13 を使用してアクセス・パスを作成します。アクセス・パスが作成される場合、Query 最適化プログラムが、照会を処理するために一時アクセス・パスを作成するよう選択できます。 *MAP がキー・フィールドのいずれか 1 つに対して戻される場合、または *I が順序テーブルに対して戻される場合、永続アクセス・パスは作成できません。永続アクセス・パスは、これらの仕様では作成できません。

CPI4322 - アクセス・パスがキー付きファイル &1 から作成された。	
メッセージ・テキスト:	アクセス・パスがキー付きファイル &1 から作成された。
原因テキスト:	理由コード &10 のため、ライブラリー &5 にあるファイル &4 のメンバー &6 からレコードにアクセスするために、ライブラリー &2 にあるキー付きファイル &1 のメンバー &3 からアクセス・パスを使用して、一時アクセス・パスが作成されました。このプロセスに &11 分 &12 秒かかりました。アクセス・パスの作成には &15 項目が含まれています。理由コードおよびその意味は次のとおりです。 1 - 指定された順序付け/グループ化基準を実行します。 2 - 指定された結合基準を実行します。 3 - 指定したレコード選択を実行して入出力待ち時間を最小にします。 アクセス・パスは、次のキー・フィールドを使用して作成されました。キー・フィールドおよび対応する順序 (ASCEND または DESCEND) が表示されます。 &17。 *MAP のキー・フィールドは、キー・フィールドが式 (派生したフィールド) であることを示します。 一時アクセス・パスは、ライブラリー &14 の順序テーブル &13 を使用して作成されました。 *N の順序テーブルは、アクセス・パスが順序テーブルを使用せずに作成されたことを示します。*I の順序テーブルは、テーブルがユーザーの使用できない、内部で派生したテーブルであったことを示します。 ライブラリー &5 のファイル &4 が論理ファイルである場合、一時アクセス・パスはライブラリー &8 の物理ファイル &7 のメンバー &9 に対して作成されます。キー付きファイルからアクセス・パスを作成することによって、一般的にパフォーマンスが向上します。
リカバリー・テキスト:	この照会が頻繁に実行される場合、パフォーマンス上の理由でこの定義に類似したアクセス・パス (索引) を作成することができます。順序テーブルが *N でない場合、ライブラリー &14 の順序テーブル &13 を使用してアクセス・パスを作成します。アクセス・パスが作成される場合、Query 最適化プログラムが、照会を処理するために一時アクセス・パスを作成するよう選択できます。 *MAP がキー・フィールドのいずれか 1 つに対して戻される場合、または *I が順序テーブルに対して戻される場合、永続アクセス・パスは作成できません。永続アクセス・パスは、これらの仕様では作成できません。 この一時アクセス・パスによって使用されたすべてのフィールドも、キー付きファイルからのアクセス・パスのキー・フィールドである場合、一時アクセス・パスは、索引専用アクセスを使用してのみ作成することができます。

CPI4323 - 照会アクセス・プランが再作成された	
メッセージ・テキスト:	照会アクセス・プランが再作成された。

CPI4323 - 照会アクセス・プランが再作成された	
原因テキスト:	アクセス・プランは、理由コード &13 のため再作成されました。理由コードおよびその意味は次のとおりです。 0 - 新規のアクセス・プランが作成されました。 1 - ファイルまたはメンバーが、アクセス・プランで参照されたのと同じオブジェクトではありません。理由として、オブジェクトが再作成されている、復元されている、または新規オブジェクトにオーバーライドされていることなどが含まれます。 2 - アクセス・プランは、再使用可能オープン・データ・パス (ODP) を使用していました。最適化プログラムは、再使用不可 ODP を使用することを選択しました。 3 - アクセス・プランは、再使用不可オープン・データ・パス (ODP) を使用していました。最適化プログラムは、再使用可能 ODP を使用することを選択しました。 4 - ライブラリー &2 にあるファイル &1 のメンバー &3 にあるレコードの数は、10% より多く変更されました。 5 - ライブラリー &5 にあるファイル &4 のメンバー &6 に対する新しいアクセス・パスが存在します。 6 - このアクセス・プランに使用した、ライブラリー &8 にあるファイル &7 のメンバー &9 に対するアクセス・パスは、もはや存在しないか、またはもはや有効ではありません。 7 - システム・プログラミング変更のため、照会アクセス・プランを再作成する必要がありました。 8 - 現行ジョブの CCSID (コード化文字セット ID) が、アクセス・プランで使用した CCSID と異なっています。 9 - 次のいずれか 1 つの値が、現行ジョブにおいて異なります: 日付形式、日付区切り記号、時刻形式、または時刻区切り記号。 10 - 指定したソート順序テーブルが変更されました。 11 - アクティブなプロセッサーの数、またはストレージ・プールのサイズかページングのオプションが変更されました。 12 - システム・フィーチャー DB2 Symmetric Multiprocessing がインストールされるか除去されました。 13 - CHGSYSVAL か CHGQRYA CL コマンドによって、またはライブラリー &16 にある照会オプション・ファイル &15 を使用して、等級 (degree) 照会属性の値が変更されました。 14 - ビューが、高水準言語のオープンによってオープンされたか、またはマテリアライズされています。 15 - 順序オブジェクトまたはユーザー定義タイプか関数が、アクセス・プランで参照されたのと同一のオブジェクトではありません。あるいは、アクセス・プランの生成に使用された SQL パスが現行の SQL パスと異なります。 16 - 照会属性が、ライブラリー &16 にある照会オプション・ファイル &15 から指定されました。 17 - 現行ジョブで異なるコミットメント制御レベルで、アクセス・プランが生成されました。 18 - 異なる静的カーソル応答セット・サイズで、アクセス・プランが生成されました。 19 - これが、作成またはコンパイル後の最初の照会の実行です。 20 およびそれ以上 -- これらの理由コードの説明については、発行される次のメッセージ (CPI4351) の 2 次レベルのメッセージ・テキストを表示します。 理由コードが 4、5、6、20、または 21 であり、理由コードの説明で指定したファイルが論理ファイルである場合、ライブラリー &11 にある物理ファイル &10 のメンバー &12 が、指定された変更が加えられたファイルです。

CPI4323 - 照会アクセス・プランが再作成された	
リカバリー・テキスト:	過度の再作成が避けられるはずであり、アプリケーション設計の問題を示す場合があります。

CPI4324 - ファイル &1 用に一時ファイルが作成された。	
メッセージ・テキスト:	ファイル &1 用に一時ファイルが作成された。
原因テキスト:	理由コード &4 のため、一時ファイルがライブラリー &2 にあるファイル &1 のメンバー &3 のために作成されました。このプロセスに &5 分 &6 秒かかりました。一時ファイルは、照会が処理されるために必要です。理由コードおよびその意味は次のとおりです。 1 - ファイルが結合論理ファイルであって、その結合タイプ (JDFTVAL) が照会に指定された結合タイプと一致しません。 2 - 論理ファイルに指定された形式が複数の物理ファイルを参照しています。 3 - ファイルは、複合 SQL ビュー、ネストされたテーブルの式、共通テーブル式、または一時ファイルを必要とするデータ変更テーブル参照です。 4 - 更新可能照会の場合、副選択はこのファイルのフィールドで、更新中のいずれかのフィールドと一致するフィールドを参照します。 5 - 更新可能照会の場合、副選択は更新中のファイルに基づく SQL ビュー &1 を参照します。 6 - 削除可能照会の場合、副選択はレコードの削除元ファイルか SQL ビュー、またはレコードの削除元ファイルに基づく論理ファイルのいずれかを参照します。 7 - ファイルは &2 にあるユーザー定義テーブル関数 &8 であり、すべてのレコードが関数から検索されました。処理時間はこの理由コードのために戻されません。 8 - ファイルは、グループ化または結合を処理するために一時ファイルを必要とするパーティション・ファイルです。
リカバリー・テキスト:	照会を変更して、作成のために一時ファイルを必要としないファイルを参照することができます。

CPI4325 - 照会プログラム用に一時結果ファイルが作成された。	
メッセージ・テキスト:	照会プログラム用に一時結果ファイルが作成された。

CPI4325 - 照会プログラム用に一時結果ファイルが作成された。	
原因テキスト:	理由コード &4 のため、照会の結果を入れるために一時結果ファイルが作成されました。このプロセスに &5 分 &6 秒かかりました。作成された一時ファイルには、&7 レコードが含まれています。理由コードおよびその意味は次のとおりです。 1 - 照会には、複数のファイルからのグループ化フィールド (GROUP BY) が含まれていたり、または再順序付けができない結合照会の 2 次ファイルからのグループ化フィールドが含まれていたりします。 2 - 照会には、複数のファイルからの順序付けフィールド (ORDER BY) が含まれていたり、または再順序付けができない結合照会の 2 次ファイルからの順序付けフィールドが含まれていたりします。 3 - グループ化フィールドと順序付けフィールドには、互換性がありません。 4 - DISTINCT が照会に対して指定されました。 5 - セット演算子 (UNION、EXCEPT、または INTERSECT) が照会に対して指定されました。 6 - 照会は、分類を使用して実装しなければなりませんでした。順序付けで 120 個を超えるキー・フィールドが指定されました。 7 - Query 最適化プログラムは、照会の結果を順序付けするアクセス・パスではなく、分類を使用することを選択しました。 8 - 指定したレコード選択を実行して入出力待ち時間を最小にします。 9 - Query 最適化プログラムは、アクセス・パスを使って照会のグループ化を実行するのではなく、ハッシュ・アルゴリズムを使用することを選択しました。 10 - 照会には、一時ファイルを必要とする結合条件が含まれています。 11 - Query 最適化プログラムは、照会ごとに特定の相関グループをインプリメントするために、ランタイム一時ファイルを作成します。 12 - 照会はグループ・フィールド (GROUP BY、MIN/MAX、COUNT など) を含み、照会内の 1 つ以上の基礎となる物理ファイルには、読み取りトリガーがあります。 13 - 照会には静的カーソルまたは SQL FETCH FIRST 文節が含まれます。
リカバリー・テキスト:	一時結果が使用された理由について詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』 を参照してください。

CPI4325 - 照会プログラム用に一時結果ファイルが作成された。	
メッセージ・テキスト:	&12 &13 が結合位置 &10 で処理された。

CPI4325 - 照会プログラム用に一時結果ファイルが作成された。	
原因テキスト:	理由コード &9 のため、ライブラリー &4 にあるファイル &3 のメンバー &5 に対するアクセス・パスが、ライブラリー &1 にあるファイル &13 のメンバー &2 にあるレコードにアクセスするために使用されました。理由コードおよびその意味は次のとおりです。 1 - 指定されたレコード選択を実行します。 2 - 指定された順序付け/グループ化基準を実行します。 3 - レコード選択および順序付け/グループ化基準。 4 - 指定された結合基準を実行します。 ライブラリー &1 にあるファイル &13 が論理ファイルである場合、ライブラリー &7 にある物理ファイル &6 のメンバー &8 は、結合位置 &10 にある実際のファイルです。 アクセス・パスの *TEMPX で始まるファイル名は、ファイル &6 に対して作成された一時アクセス・パスであることを示します。 ファイル名が *N または *QUERY で始まるファイルは、一時ファイルであることを示します。 索引専用アクセスが照会 &11 内のこのファイルに使用されました。 索引専用アクセスの処理に対する *YES の値は、この照会のこのファイルから使用されるすべてのフィールドが、ファイル &3 のアクセス・パス内で検出できることを示します。 *NO の値は、索引専用アクセスがこのアクセス・パスで実行できなかったことを示します。 索引専用アクセスは、すべてのデータがアクセス・パスから抽出でき、データ・スペースをアクティブ・メモリー内にページ付けする必要がないため、一般的にパフォーマンス上の利点があります。
リカバリー・テキスト:	一般的に、結合位置 1 でファイルが処理されるように強制するには、そのファイルからのみフィールド別に順序を指定します。 順序付けが必要な場合、複数のファイルに対して ORDER BY フィールドを指定することによって、一時ファイルの作成を強制し、最適化プログラムはすべてのファイルの結合順序を最適化できます。第 1 であるように強制されるファイルはありません。 このファイルの照会内で使用されるすべてのフィールドも、そのアクセス・パスのキー・フィールドである場合、アクセス・パスは索引専用アクセスに対してのみ考慮されます。 照会の結合順序および索引専用アクセスの最適化に関する追加のヒントは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』 を参照してください。 場合によっては、一時結果テーブルを作成すると、照会の実行速度が最速になります。一時結果テーブルにコピーする行が多数になるような他の照会の場合は、かなりの時間を要する場合があります。しかし、照会に使用できるよりも多くの時間およびリソースが費やされる場合には、照会を変更して一時結果テーブルが必要なくなるようにすることを検討してください。

CPI4326 - &12 &13 が結合位置 &10 で処理された。	
メッセージ・テキスト:	&12 &13 が結合位置 &10 で処理された。

CPI4326 - &12 &13 が結合位置 &10 で処理された。	
原因テキスト:	理由コード &9 のため、ライブラリー &4 にあるファイル &3 のメンバー &5 に対するアクセス・パスが、ライブラリー &1 にあるファイル &13 のメンバー &2 にあるレコードにアクセスするために使用されました。理由コードおよびその意味は次のとおりです。 1 - 指定されたレコード選択を実行します。 2 - 指定された順序付け/グループ化基準を実行します。 3 - レコード選択および順序付け/グループ化基準。 4 - 指定された結合基準を実行します。 ライブラリー &1 にあるファイル &13 が論理ファイルである場合、ライブラリー &7 にある物理ファイル &6 のメンバー &8 は、結合位置 &10 にある実際のファイルです。 アクセス・パスの *TEMPX で始まるファイル名は、ファイル &6 に対して作成された一時アクセス・パスであることを示します。 ファイル名が *N または *QUERY で始まるファイルは、一時ファイルであることを示します。 索引専用アクセスが照会 &11 内のこのファイルに使用されました。 索引専用アクセスの処理に対する *YES の値は、この照会のこのファイルから使用されるすべてのフィールドが、ファイル &3 のアクセス・パス内で検出できることを示します。 *NO の値は、索引専用アクセスがこのアクセス・パスで実行できなかったことを示します。 索引専用アクセスは、すべてのデータがアクセス・パスから抽出でき、データ・スペースをアクティブ・メモリー内にページ付けする必要がないため、一般的にパフォーマンス上の利点があります。
リカバリー・テキスト:	一般的に、結合位置 1 でファイルが処理されるように強制するには、そのファイルからのみフィールド別に順序を指定します。 順序付けが必要な場合、複数のファイルに対して ORDER BY フィールドを指定することによって、一時ファイルの作成を強制し、最適化プログラムはすべてのファイルの結合順序を最適化できます。第 1 であるように強制されるファイルはありません。 このファイルの照会内で使用されるすべてのフィールドも、そのアクセス・パスのキー・フィールドである場合、アクセス・パスは索引専用アクセスに対してのみ考慮されます。 照会の結合順序および索引専用アクセスの最適化に関する追加のヒントは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』 を参照してください。 場合によっては、一時結果テーブルを作成すると、照会の実行速度が最速になります。一時結果テーブルにコピーする行が多数になるような他の照会の場合は、かなりの時間を要する場合があります。しかし、照会に使用できるよりも多くの時間およびリソースが費やされる場合には、照会を変更して一時結果テーブルが必要なくなるようにすることを検討してください。

このメッセージは、テーブルのデータをアクセスするために索引が使用されるとき、指定されたテーブルの結合位置を示しています。**結合位置**とは、テーブルが結合される順序に関する用語です。

CPI4327 - ファイル &12 &13 が結合位置 &10 で処理された。	
メッセージ・テキスト:	&12 &13 が結合位置 &10 で処理された。
原因テキスト:	到着順アクセスが、ライブラリー &1 にあるファイル &13 のメンバー &2 からレコードを選択するのに使用されました。 ライブラリー &1 にあるファイル &13 が論理ファイルである場合、ライブラリー &7 にある物理ファイル &6 のメンバー &8 は、結合位置 &10 にある実際のファイルです。 ファイルの *QUERY で始まるファイル名は、一時ファイルであることを示します。
リカバリー・テキスト:	一般的に、結合位置 1 でファイルが処理されるように強制するには、そのファイルからのみフィールド別に順序を指定します。 照会の結合順序の最適化に関する追加のヒントは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』 を参照してください。

CPI4328 - 照会によってファイル &3 のアクセス・パスが使用された。	
メッセージ・テキスト:	照会によってファイル &3 のアクセス・パスが使用された。
原因テキスト:	理由コード &9 のため、ライブラリー &4 にあるファイル &3 のメンバー &5 に対するアクセス・パスが、ライブラリー &1 にある &12 &13 のメンバー &2 からレコードにアクセスするために使用されました。理由コードおよびその意味は次のとおりです。 1 - レコード選択。 2 - 順序付け/グループ化基準。 3 - レコード選択および順序付け/グループ化基準。 ライブラリー &1 にあるファイル &13 が論理ファイルである場合、ライブラリー &7 にある物理ファイル &6 のメンバー &8 は、アクセスされる実際のファイルです。 索引専用アクセスがこの照会 &11 に使用されました。 索引専用アクセスの処理に対する *YES の値は、この照会で使用されるすべてのフィールドが、ファイル &3 のアクセス・パス内で検出できることを示します。 *NO の値は、索引専用アクセスがこのアクセス・パスで実行できなかったことを示します。 索引専用アクセスは、すべてのデータがアクセス・パスから抽出でき、データ・スペースをアクティブ・メモリー内にページ付けする必要がないため、一般的にパフォーマンス上の利点があります。
リカバリー・テキスト:	このファイルの照会内で使用されるすべてのフィールドも、そのアクセス・パスのキー・フィールドである場合、アクセス・パスは索引専用アクセスに対してのみ考慮されます。 索引専用アクセスに関する追加のヒントは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI4329 - &12 &13 に到着順アクセスが使用された。	
メッセージ・テキスト:	&12 &13 に到着順アクセスが使用された。

CPI4329 - &12 &13 に到着順アクセスが使用された。	
原因テキスト:	到着順アクセスが、ライブラリー &1 にあるファイル &13 のメンバー &2 からレコードを選択するのに使用されました。 ライブラリー &1 にあるファイル &13 が論理ファイルである場合、ライブラリー &7 にある物理ファイル &6 のメンバー &8 は、そこからレコードが選択される実際のファイルです。 ファイル名が *N または *QUERY で始まるファイルは、一時ファイルであることを示します。
リカバリー・テキスト:	レコード選択が指定される場合、アクセス・パスの使用が、照会のパフォーマンスを向上する場合があります。 アクセス・パスが存在しない場合には、左端のキー・フィールドがレコード選択のフィールドの 1 つに一致するアクセス・パスを作成することができます。レコード選択のフィールドと一致するアクセス・パスのキー・フィールドが多いほど、パフォーマンスは向上します。 一般的に、既存のアクセス・パスの使用を強制するには、そのアクセス・パスの左端のキー・フィールドと一致するフィールドによって順序を指定します。 詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI432A - ファイル &1 で Query 最適化プログラムのタイムアウトが起こった。	
メッセージ・テキスト:	ファイル &1 で Query 最適化プログラムのタイムアウトが起こった。

CPI432A - ファイル &1 で Query 最適化プログラムのタイムアウトが起こった。	
原因テキスト:	ライブラリー &2 にあるファイル &1 のメンバー &3 に対して作成されたすべてのアクセス・パスを考慮する前に、Query 最適化プログラムがタイムアウトになりました。 下のリストは、最適化プログラムがタイムアウトになる前に考慮されたアクセス・パスを表示しています。ライブラリー &2 にあるファイル &1 が論理ファイルである場合、指定されたアクセス・パスはライブラリー &8 にある物理ファイル &7 のメンバー &9 に対して実際に作成されます。リストにある以下の各アクセス・パス名は、最適化プログラムがアクセス・パスをどのように見なしたかを説明する理由コードです。 &11。 理由コードおよびその意味は次のとおりです。 0 - アクセス・パスは、照会をインプリメントするのに使用されました。 1 - アクセス・パスが有効な状態ではありませんでした。システムはアクセス・パスを無効にしました。 2 - アクセス・パスが有効な状態ではありませんでした。ユーザーはアクセス・パスが再作成されるよう要求しました。 3 - アクセス・パスは、一時アクセス・パス (ライブラリー QTEMP にある) であり、照会されるファイルとして指定されませんでした。 4 - 最適化プログラムによって判別された、このアクセス・パスを使用するコストは、選択されたアクセス方式と関連したコストより高いものでした。 5 - アクセス・パスのキーは、順序付け/グループ化基準に指定されたフィールドと一致しませんでした。 6 - アクセス・パスのキーは、結合基準に指定されたフィールドと一致しませんでした。 7 - このアクセス・パスの使用は、ユーザーの要求時にファイルからレコードを読み取るときの遅延を最小限に抑えません。 8 - アクセス・パスには静的選択/除外の選択基準が含まれているため、結合照会の 2 次ファイルに使用することはできません。照会の結合タイプは、2 次ファイルに対するアクセス・パスの選択/除外の使用を許可しません。 9 - ファイル &1 にはレコード ID 選択が含まれています。照会の結合タイプは、一時アクセス・パスがレコード ID 選択を処理するために作成されるよう強制します。 10 およびそれ以上 - これらの理由コードの説明については、発行される次のメッセージ (CPI432D) の 2 次レベルのメッセージ・テキストを表示します。
リカバリー・テキスト:	アクセス・パスが最適化のために考慮されるようにするには、アクセス・パスが照会されるファイルとなるよう指定します。最適化プログラムは照会に指定されたファイルのアクセス・パスを最初に考慮します。 SQL で作成された索引は照会されませんが、削除して再作成することによって、照会の最適化中に考慮される機会を増やすことができます。 不要になったアクセス・パスは削除することができます。

CPI432B - 副選択が結合 QUERY として処理された	
メッセージ・テキスト:	副選択が結合 QUERY として処理された。
原因テキスト:	2 つ以上の SQL 副選択が Query 最適化プログラムによって共に結合され、結合照会として処理されました。結合照会として副選択を処理すると、一般にパフォーマンスを向上することになります。
リカバリー・テキスト:	ありません — 一般に、この処理方法は良好なパフォーマンスを与えるオプションです。

CPI432C - ファイル &1 のすべてのアクセス・パスが考慮された。	
メッセージ・テキスト:	ファイル &1 のすべてのアクセス・パスが考慮された。

CPI432C - ファイル &1 のすべてのアクセス・パスが考慮された。	
原因テキスト:	Query 最適化プログラムは、ライブラリー &2 にあるファイル &1 のメンバー &3 に対して作成されたすべてのアクセス・パスを考慮しました。 下のリストは、考慮されたアクセス・パスを表示しています。ライブラリー &2 にあるファイル &1 が論理ファイルである場合、指定されたアクセス・パスはライブラリー &8 にある物理ファイル &7 のメンバー &9 に対して実際に作成されます。リストにある以下の各アクセス・パス名は、最適化プログラムがアクセス・パスをどのように見なしたかを説明する理由コードです。 &11。 理由コードおよびその意味は次のとおりです。 0 - アクセス・パスは、照会をインプリメントするのに使用されました。 1 - アクセス・パスが有効な状態ではありませんでした。システムはアクセス・パスを無効にしました。 2 - アクセス・パスが有効な状態ではありませんでした。ユーザーはアクセス・パスが再作成されるよう要求しました。 3 - アクセス・パスは、一時アクセス・パス (ライブラリー QTEMP にある) であり、照会されるファイルとして指定されませんでした。 4 - 最適化プログラムによって判別された、このアクセス・パスを使用するコストは、選択されたアクセス方式と関連したコストより高いものでした。 5 - アクセス・パスのキーは、順序付け/グループ化基準に指定されたフィールドと一致しませんでした。分散ファイルの照会に関しては、ALWCPYDTA(*YES または *NO) が指定されているときにアクセス・パスが使用される場合、アクセス・パスのキーが順序付けフィールドと正確に一致しなければなりません。 6 - アクセス・パスのキーは、結合基準に指定されたフィールドと一致しませんでした。 7 - このアクセス・パスの使用は、ファイルからレコードを読み取るときの遅延を最小限に抑えません。ユーザーは、ファイルからレコードを読み取るときの遅延を最小限に抑えるよう要求しました。 8 - アクセス・パスには静的選択/除外の選択基準が含まれているため、結合照会の 2 次ファイルに使用することはできません。照会の結合タイプは、2 次ファイルに対するアクセス・パスの選択/除外の使用を許可しません。 9 - ファイル &1 にはレコード ID 選択が含まれています。照会の結合タイプは、一時アクセス・パスがレコード ID 選択を処理するために作成されるよう強制します。 10 およびそれ以上 - これらの理由コードの説明については、発行される次のメッセージ (CPI432D) の 2 次レベルのメッセージ・テキストを表示します。
リカバリー・テキスト:	不要になったアクセス・パスは削除することができます。

CPI432D - 追加のアクセス・パス理由コードが使用された。	
メッセージ・テキスト:	追加のアクセス・パス理由コードが使用された。

CPI432D - 追加のアクセス・パス理由コードが使用された。	
原因テキスト:	メッセージ CPI432A またはメッセージ CPI432C がこのメッセージの直前に出されました。メッセージ長に制限があるために、メッセージ CPI432A およびメッセージ CPI432C で使用している一部の理由コードを、これらのメッセージ中ではなく下記で説明しています。 理由コードおよびその意味は次のとおりです。 10 - 指定されたユーザーは、照会の 10 進データ・エラーを無視します。これにより、永続アクセス・パスの使用は許可されなくなります。 11 - アクセス・パスには、照会の選択と互換性のない静的選択/除外の選択基準が含まれます。 12 - アクセス・パスには、照会の選択との互換性が判別できなかった、静的選択/除外の選択基準が含まれます。互換性の処理中に、選択/除外基準か照会選択のどちらかが複雑になりすぎました。 13 - アクセス・パスには、挿入または更新中に照会によって変更される場合のある 1 つ以上のキーが含まれているため、使用できません。 14 - アクセス・パスは、他のプロセスの非コミット作業単位で削除されているか、作成されています。 15 - アクセス・パスのキーは、順序付け/グループ化基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。 16 - アクセス・パスのキーは、結合基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。 17 - アクセス・パスの左端のキーは、選択基準に指定された任意のフィールドと一致しませんでした。このアクセス・パスを使用するコストが選択されたアクセス方式と関連するコストより高くなるため、キーの行の位置決めは実行されませんでした。 18 - アクセス・パスの左端のキーは、選択基準に指定されたフィールドと一致しました。ただし、アクセス・パスに関連する順序テーブルは、照会に関連する順序テーブルと一致しませんでした。このアクセス・パスを使用するコストが選択されたアクセス方式と関連するコストより高くなるため、キーの行の位置決めは実行されませんでした。 19 - 結合照会の 2 次ファイルが選択/除外の論理ファイルであるため、アクセス・パスを使用することはできません。結合タイプは、2 次ファイルと関連する選択/除外のアクセス・パスが使用されるか、動的な場合には、アクセス・パスがシステムによって作成されることを要求します。 99 - アクセス・パスは、Query 最適化プログラムの統計情報を収集するために使用されました。
リカバリー・テキスト:	詳しくは、前のメッセージ CPI432A または CPI432C を参照してください。

メッセージ長に制限があるために、メッセージ CPI432A およびメッセージ CPI432C で使用している一部の理由コードを CPI432D のメッセージ・ヘルプで説明しています。このメッセージからメッセージ・ヘルプを使用して、メッセージ CPI432A またはメッセージ CPI432C から返された情報を解釈してください。

CPI432E - 選択フィールドが異なる属性にマップされた。	
メッセージ・テキスト:	選択フィールドが異なる属性にマップされた。

CPI432E - 選択フィールドが異なる属性にマップされた。	
原因テキスト:	フィールドを、それに関連するリテラル、ホスト変数、またはフィールド・オペランドと正しく比較できるように、次の各選択フィールドのデータ・タイプ、桁数、小数点以下の桁数、または長さが変更されました。したがって、アクセス・パスをその選択の処理に使用することはできません。これは、キー・フィールドが、フィールドの新しい属性に一致する属性を持たないからです。 &1. フィールドのデータ・タイプが比較オペランドに一致するように変更された可能性があります。数字フィールドの場合、比較オペランドの桁数または小数桁の合計がフィールドのそれを超えた可能性があります。
リカバリー・テキスト:	各比較オペランドを次のように変更することが必要な場合があるかもしれません。 1 - リテラルの場合、属性がフィールドの属性に一致するようにリテラル値を変更します。通常、属性の不一致は、意味のない先行または後続ゼロのある数値リテラルによって引き起こされます。 2 - ホスト変数の場合、ホスト変数の定義を変更してフィールドの定義に一致させるか、フィールドの定義に一致する新しいホスト変数を定義します。 3 - フィールドの場合、フィールドの 1 つの属性を変更して、もう一方の属性に一致させます。

CPI432F - ファイル &1 のアクセス・パス示唆	
メッセージ・テキスト:	ファイル &1 のアクセス・パス示唆。
原因テキスト:	パフォーマンスを向上するために、Query 最適化プログラムは推奨されているキー・フィールドで永続アクセス・パスが作成されることを示唆しています。アクセス・パスは、ライブラリー &2 にあるファイル &1 のメンバー &3 からレコードにアクセスします。 続くキー・フィールドのリストで、Query 最適化プログラムは最初の &10 キー・フィールドを 1 次キー・フィールドとして推奨しています。残りのキー・フィールドは、2 次キー・フィールドと見なされ、この照会に基づいて予期される選択順にリストされます。 1 次キー・フィールドは、対応する選択述部に基づいて選択されるキーの数を顕著に削減するフィールドです。 2 次キー・フィールドは、選択されるキーの数を顕著に削減することもあり、または削減しないこともあるフィールドです。 2 次キー・フィールドに実際にどれを選択するかを決定し、アクセス・パスの作成時にこれらのキー・フィールドを使用するかどうかを決定するのは、ユーザー自身です。 Query 最適化プログラムは、1 つの追加 2 次キー・フィールドに加えて、1 次キー・フィールドの任意の組み合わせについてキーの位置決めを行うことができます。したがって、最初の 2 次キー・フィールドが最も慎重に選択された 2 次キー・フィールドであることが重要になります。Query 最適化プログラムは、任意の残った 2 次キー・フィールドでキー選択を使用します。キー選択は、キーの位置決めほどは迅速ではないけれども、それでもこれによって、選択されるキーの数を減らすことができます。したがって、十分に選択された 2 次キー・フィールドを組み込む必要があります。アクセス・パスを作成しているとき、すべての 1 次キー・フィールドが最初に指定され、選択によって優先順位付けられる 2 次キー・フィールドが後に続くはずです。続くリストには、提案された 1 次および 2 次キー・フィールドが含まれています。 &11。 ライブラリー &2 にあるファイル &1 が論理ファイルである場合、アクセス・パスはライブラリー &8 にある物理ファイル &7 のメンバー &9 に対して作成されます。

CPI432F - ファイル &1 のアクセス・パス示唆	
リカバリー・テキスト:	この照会が頻繁に実行される場合、パフォーマンス上の理由で提案されたアクセス・パスを作成できます。 Query 最適化プログラムが、作成されたばかりのアクセス・パスを使用しないことを選択することもできます。 詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI4330 - &6 個のタスクがファイル &1 の並行 &10 走査に使用された。	
メッセージ・テキスト:	&6 個のタスクがファイル &1 の並行 &10 走査に使用された。
原因テキスト:	&6 は、ライブラリー &2 にあるファイル &1 のメンバー &3 の &10 走査に使用されたタスクの平均数です。 ライブラリー &2 にあるファイル &1 が論理ファイルである場合、ライブラリー &8 にある物理ファイル &7 のメンバー &9 は、そこからレコードが選択される実際のファイルです。 *QUERY または *N で始まるファイル名は、一時結果ファイルが使用されていることを示します。 Query 最適化プログラムは、タスクの最適数が、理由コード &4 のため限定された &5 であると計算しました。理由コードの定義は次のとおりです。 1 - *NBRTASKS パラメーター値が CHGQRYA CL コマンドの DEGREE パラメーターに指定されました。 2 - 最適化プログラムは、すべての中央演算処理装置 (CPU) を使用するタスクの数を計算しました。 3 - 最適化プログラムは、このジョブのメモリー・プールの共用で効果的に実行できるタスクの数を計算しました。 4 - 最適化プログラムは、メモリー・プール全体を使用して効果的に実行できるタスクの数を計算しました。 5 - 最適化プログラムは、ファイルのデータを含むディスク装置の数と等しくなるようタスクの数を限定しました。 ファイルのデータの割り振りがディスク装置を介して平等に分散されない場合、データベース・マネージャーはさらに使用されるタスクの数を限定する場合があります。

CPI4330 - &6 個のタスクがファイル &1 の並行 &10 走査に使用された。	
リカバリー・テキスト:	並行 &10 走査の使用を許可しないためには、照会属性 DEGREE で *NONE を指定します。 タスクの数を多くすると、パフォーマンスが向上する場合があります。最適化プログラムの理由コードに基づく次のアクションによって、最適化プログラムはさらに大きな数を計算できます。 1 - CHGQRYA CL コマンドの DEGREE パラメーターに、タスク数としてもっと大きい値を指定します。 &5 よりわずかに大きいタスクの数の値で開始します。 2 - 結果バッファーにマップされるフィールドの数を少なくすることによって、または式を除去することによって、照会を単純化します。また、理由コード 1 によって説明されているように、タスクの数を指定します。 3 - 照会属性 DEGREE に *MAX を指定します。 4 - メモリー・プールのサイズを増やします。 5 - CHGPF CL コマンドまたは SQL ALTER ステートメントを使用して、さらに多くのディスク装置を介してファイルのデータを再配布します。

CPI4331 - &6 タスクがファイルの並列索引に作成された。	
メッセージ・テキスト:	&6 タスクがファイル &1 に対する並列索引の作成に使用された。
原因テキスト:	&6 は、ライブラリー &2 にあるファイル &1 のメンバー &3 に対する索引の作成に使用されたタスクの平均数です。 ライブラリー &2 にあるファイル &1 が論理ファイルである場合、ライブラリー &8 にある物理ファイル &7 のメンバー &9 は、索引が作成されている実際のファイルです。 *QUERY または *N で始まるファイル名は、一時結果ファイルが使用されていることを示します。 Query 最適化プログラムは、タスクの最適数が、理由コード &4 のため限定された &5 であると計算しました。理由コードの定義は次のとおりです。 1 - *NBRTASKS パラメーター値が CHGQRYA CL コマンドの DEGREE パラメーターに指定されました。 2 - 最適化プログラムは、すべての中央演算処理装置 (CPU) を使用するタスクの数を計算しました。 3 - 最適化プログラムは、このジョブのメモリー・プールの共用で効果的に実行できるタスクの数を計算しました。 4 - 最適化プログラムは、メモリー・プール全体を使用して効果的に実行できるタスクの数を計算しました。 ファイルのデータの割り振りがディスク装置を介して平等に分散されないか、システムにあるディスク装置が少なすぎる場合、データベース・マネージャーは並列索引作成に使用されるタスクの数を、さらに限定する場合があります。

CPI4331 - &6 タスクがファイルの並列索引に作成された。	
リカバリー・テキスト:	並列索引作成の使用を許可しないために、照会属性 DEGREE で *NONE を指定します。 タスクの数を多くすると、パフォーマンスが向上する場合があります。理由コードに基づく次のアクションによって、最適化プログラムはさらに大きな数を計算できます。 1 - CHGQRYA CL コマンドの DEGREE パラメーターに、タスク数としてもっと大きい値を指定します。 &5 よりわずかに大きいタスクの数の値で開始して、パフォーマンスが向上したかどうか確かめてください。 2 - 結果バッファーにマップされるフィールドの数を少なくすることによって、または式を除去することによって、照会を単純化します。また、理由コード 1 によって説明されているように、CHGQRYA CL コマンドの DEGREE パラメーターにタスクの数を指定します。 3 - 照会属性 DEGREE に *MAX を指定します。 4 - メモリー・プールのサイズを増やします。

CPI4332 - &1 ホスト変数が照会で使用された。	
メッセージ・テキスト:	&1 ホスト変数が照会で使用された。
原因テキスト:	照会で使用するよう定義された &1 ホスト変数がありました。この照会のオープンのためにホスト変数に使用される値が続きます: &2。 上に表示されたホスト変数の値は、特殊値である可能性があります。特殊値の説明は次のとおりです。 - DBCS データが 16 進数形式で表示されます。 - *N は NULL の値を表します。 - *Z は長さゼロのストリングを表します。 - *L は長すぎて置換テキストに表示できない値を表します。 - *U は表示できなかった値を表します。
リカバリー・テキスト:	ありません。

CPI4333 - 結合の処理にハッシュ・アルゴリズムが使用された。	
メッセージ・テキスト:	結合の処理にハッシュ・アルゴリズムが使用された。

CPI4333 - 結合の処理にハッシュ・アルゴリズムが使用された。	
原因テキスト:	ハッシュ結合方式は一般的に、結合照会を長く実行するために使用されます。元の照会は、ハッシュ結合ステップに細分化されます。 各ハッシュ結合ステップは最適化され、別個に処理されます。各ハッシュ結合ステップのインプリメンテーションを説明するデバッグ・メッセージは、ジョブ・ログでこのメッセージに続きます。 下記のリストは、この照会で使用するファイルまたはテーブル関数の名前を示しています。項目がファイルに対するものである場合、このリストの項目の形式は、ハッシュ結合ステップの数、照会で指定されるファイル名、照会で指定されるメンバー名、ハッシュ結合ステップで実際に使用されるファイル名、およびハッシュ結合ステップで実際に使用されるメンバー名となります。項目がテーブル関数に対するものである場合、このリストの入力の形式は、ハッシュ結合ステップの数と、照会で指定される関数名となります。 同じハッシュ・ステップに複数のファイルまたは関数がリストされる場合、ハッシュ・ステップはネストされたループ結合で実行されます。
リカバリー・テキスト:	ハッシュ結合方式は、通常良い選択です。ただし、この方式の使用を許可したくない場合は、ALWCPYDTA(*YES) を指定します。

CPI4334 - 照会プログラムが再使用可能な ODP として実現された。	
メッセージ・テキスト:	照会プログラムが再使用可能な ODP として実現された。
原因テキスト:	Query 最適化プログラムはこの照会にアクセス・プランを作成したため、再使用可能なオープン・データ・パス (ODP) が作成されます。このプランを使用すると、ODP を毎回再作成せずに、このジョブに対して照会を繰り返し実行することができます。ODP はジョブに対して一度だけ作成されるため、このプランは通常パフォーマンスを向上させます。
リカバリー・テキスト:	一般に、再使用可能 ODP は、再使用不可 ODP よりも効果的に実行します。

CPI4335 - ハッシュ結合ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり	
メッセージ・テキスト:	ハッシュ結合ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり。
原因テキスト:	この結合照会は、ハッシュ結合アルゴリズムを使用して実行されます。続く最適化プログラム・デバッグ・メッセージは、ハッシュ結合ステップ &1 についての Query 最適化情報を提供します。
リカバリー・テキスト:	結合処理のハッシュ・アルゴリズムについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI4336 - グループ処理が生成された	
メッセージ・テキスト:	グループ処理が生成された。
原因テキスト:	グループ処理 (GROUP BY) が照会のステップに追加されました。グループ処理を追加すると、結果レコードの数を減らすため、後続のステップのパフォーマンスを向上させます。
リカバリー・テキスト:	詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI4337 - ハッシュ結合ステップ &1 の一時ハッシュ・テーブルが作成された。	
メッセージ・テキスト:	ハッシュ結合ステップ &1 の一時ハッシュ・テーブルが作成された。
原因テキスト:	ハッシュ結合ステップ &1 の結果を入れるために一時ハッシュ・テーブルが作成されました。このプロセスに &2 分 &3 秒かかりました。作成された一時ハッシュ・テーブルには、&4 レコードが含まれています。一時ハッシュ・テーブルの合計サイズは、1024 バイト単位で &5 です。ハッシュ・キーを定義するフィールドのリストは、次のとおりです。
リカバリー・テキスト:	結合処理のハッシュ・アルゴリズムについて詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI4338 - &1 アクセス・パスがファイル &2 のビットマップ処理に使用された。	
メッセージ・テキスト:	&1 アクセス・パスがファイル &2 のビットマップ処理に使用された。
原因テキスト:	ライブラリー &3 にあるファイル &2 のメンバー &4 からレコードにアクセスするために、ビットマップ処理が使用されました。 ビットマップ処理は、1 つ以上のアクセス・パスが、ファイルから選択されたレコードにアクセスするために使用できる方式です。ビットマップ処理を使用すると、キーの行の位置決めと同じように、ビットマップを作成するために、レコード選択が各アクセス・パスに対して適用されます。ビットマップは、選択されるファイルのレコードに関してのみマークされます。複数のアクセス・パスが使用される場合、結果ビットマップはブール論理を使用して共にマージされます。その後、結果ビットマップは、ファイルから実際に選択されたこれらのレコードへのアクセスを減らすために使用されます。 ビットマップ処理は、2 つの基本アクセス方式、到着順 (CPI4327 または CPI4329) またはキー順アクセス (CPI4326 または CPI4328) と共に使用されます。基本アクセス方式を説明するメッセージが、このメッセージの直前に出されます。 ビットマップがキー順アクセス方式と共に使用されると、ファイルから選択レコードを取り出す前に、基本アクセス・パスによって選択されるレコードの数をさらに減らすことができます。 ビットマップが到着順と共に使用されると、ファイルの順次走査はビットマップによって選択されなかったレコードをスキップすることができます。これは順次処理のスキップと呼ばれます。 下記のリストは、ビットマップ処理で使用されるアクセス・パスの名前を示しています。 &8 ライブラリー &3 にあるファイル &2 が論理ファイルである場合、ライブラリー &6 にある物理ファイル &5 のメンバー &7 は、アクセスされる実際のファイルです。
リカバリー・テキスト:	ビットマップ処理について詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

CPI433A - 照会オプション・ファイルを検索できない。	
メッセージ・テキスト:	照会オプション・ファイルを検索できない。

CPI433A - 照会オプション・ファイルを検索できない。	
原因テキスト:	理由コード &4 により、ライブラリー &1 のファイル &2 のメンバー &3 から照会オプション・ファイルを検索できません。理由コードおよびその意味は次のとおりです。 1 - ライブラリー &1 が見つかりませんでした。 2 - ライブラリー &1 のファイル &2 が見つかりませんでした。 3 - ファイルは損傷を受けました。 4 - 別のプロセスによってファイルがロックされ、照会オプションの正常な検索が妨げられました。 5 - ファイル &2 および内部照会オプション構造は同期が取れていません。 6 - オプション・ファイルの検索を試行中に予期しないエラーが発生しました。 照会オプション・ファイルは、Query 最適化プログラムによって使用され、照会をインプリメントする方法を決定します。
リカバリー・テキスト:	上の理由コードに基づいて、次のアクションの 1 つを取らない限り、デフォルトの照会オプションが使用されます。 1 - ライブラリーを作成する (CRTLIB コマンド) か、ライブラリー名を訂正して要求を再試行します。 2 - 照会オプション・ファイルを含むライブラリー名を指定するか、ライブラリー QSYS から指定されたライブラリーにファイル &2 の複写オブジェクトを作成 (CRTDUPOBJ コマンド) します。 4 - ライブラリー &1 のファイル &2 上でロックがリリースされるのを待ち、要求を再試行します。 3、5、または 6 - ライブラリー &1 の照会オプション・ファイル &2 を削除し、QSYS から複写します。問題が継続する場合は、問題を報告します (ANZPRB コマンド)。

CPI433B - 照会オプション・ファイルを更新できない。	
メッセージ・テキスト:	照会オプション・ファイルを更新できない。
原因テキスト:	理由コード &4 により、ライブラリー &1 のファイル &2 のメンバー &3 から照会オプション・ファイルの更新を試行中にエラーが発生しました。理由コードおよびその意味は次のとおりです。 1 - ライブラリー &1 が見つかりませんでした。 2 - ライブラリー &1 のファイル &2 が見つかりませんでした。 3 - パラメーター &5 が見つかりませんでした。 4 - パラメーター &5 の値 &6 が無効です。 5 - オプション・ファイルの更新を試行中に予期しないエラーが発生しました。

CPI433B - 照会オプション・ファイルを更新できない。	
リカバリー・テキスト:	上の理由コードに基づいて、次のアクションの 1 つを実行します。 1 - ライブラリーを作成する (CRTLIB コマンド) か、ライブラリー名を訂正して要求を再試行します。 2 - 照会オプション・ファイルを含むライブラリー名を指定するか、ライブラリー QSYS から指定されたライブラリーに QAQQINI の複写オブジェクトを作成 (CRTDUPOBJ コマンド) します。 3 - 有効なパラメーターを指定するか、パラメーター名を訂正して、要求を再試行します。 4 - 有効なパラメーター値を指定するか、パラメーター値を訂正して、要求を再試行 (WRKJOB コマンド) します。

CPI433C - ライブラリー &1 が見つからない。	
メッセージ・テキスト:	ライブラリー &1 が見つからない。
原因テキスト:	指定したライブラリーが存在しないか、ライブラリーの名前のスペルが誤っています。
リカバリー・テキスト:	ライブラリー名のスペルを訂正するか、既存のライブラリーの名前を指定します。その後、要求を再試行してください。

CPI433D - 照会アクセス・プランの作成に使用する照会オプション	
メッセージ・テキスト:	照会アクセス・プランの作成に使用する照会オプション。
原因テキスト:	保管されたアクセス・プランは、ライブラリー &1 にあるファイル &2 から取り出された照会オプションを使用して作成されました。
リカバリー・テキスト:	ありません。

CPI433E - ユーザー定義関数 &4 がライブラリー &1 で見つかった	
メッセージ・テキスト:	ユーザー定義関数 &4 がライブラリー &1 で見つかった。
原因テキスト:	関数 &4 がライブラリー &1 に対して解決されました。関数の具体的な名前は、&5 です。 関数が外部プログラムを使用するように定義されている場合、関連プログラムまたはサービス・プログラムは、ライブラリー &2 の &3 です。
リカバリー・テキスト:	ユーザー定義関数について詳しくは、「SQL プログラミング」のトピック・コレクションを参照してください。

CPI433F - 結合の処理に複数の結合クラスが使用されました。	
メッセージ・テキスト:	結合の処理に複数の結合クラスが使用されました。

CPI433F - 結合の処理に複数の結合クラスが使用されました。	
原因テキスト:	操作が競合している結合照会が書き込まれるか、結合照会を単一照会としてインプリメントできない場合、複数の結合クラスが使用されます。 各結合クラス・ステップは最適化され、別個に処理されます。各結合クラスのインプリメンテーションを詳述するデバッグ・メッセージが、ジョブ・ログでこのメッセージに続きます。 下記のリストは、この照会で使用するファイルのファイル名を示しています。このリストの各項目の形式は、結合クラス・ステップの数、結合クラス・ステップにある結合位置の数、照会で指定されているファイル名、照会で指定されているメンバー名、結合クラス・ステップで実際に使用されるファイル名、および結合クラス・ステップで実際に使用されるメンバー名となります。
リカバリー・テキスト:	結合クラスについて詳しくは、 50 ページの『結合の最適化』を参照してください。

CPI4340 - 結合クラス・ステップ &1 の最適化プログラム・デバッグ・メッセージは次の通りである	
メッセージ・テキスト:	結合クラス・ステップ &1 の最適化プログラム・デバッグ・メッセージは次の通りである。
原因テキスト:	この結合照会は、複数の結合クラスを使用して実行されます。続く最適化プログラム・デバッグ・メッセージは、結合クラス・ステップ &1 についての Query 最適化情報を提供します。
リカバリー・テキスト:	結合クラスについて詳しくは、 50 ページの『結合の最適化』を参照してください。

CPI4341 - 分散照会の実行中	
メッセージ・テキスト:	分散照会の実行中。
原因テキスト:	照会には、分散ファイルが含まれています。照会はノード &1 で並列して処理されました。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

CPI4342 - 照会の分散結合の実行中	
メッセージ・テキスト:	照会の分散結合の実行中。
原因テキスト:	照会には、分散ファイルに対する結合基準が含まれており、分散結合はノード &1 で並列に実行されました。 結合に含まれる各ファイルのライブラリー、ファイル、およびメンバーの名前が続きます: &2。 *QQTDF で始まるファイル名は、Query 最適化プログラムによって作成された一時分散結果ファイルであり、関連したライブラリーまたはメンバー名は含まれていないことを示します。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」を参照してください。

CPI4343 - &2 の分散照会ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり	
メッセージ・テキスト:	&2 の分散照会ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり。

CPI4343 - &2 の分散照会ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり	
原因テキスト:	分散ファイルが照会に指定され、複数のステップで照会が処理されます。続く最適化プログラム・デバッグ・メッセージは、&2 のすべてのステップの分散ステップ &1 についての Query 最適化情報を提供します。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」を参照してください。

CPI4345 - 照会プログラム用に一時分散結果ファイル &3 が作成された。	
メッセージ・テキスト:	照会プログラム用に一時分散結果ファイル &3 が作成された。
原因テキスト:	理由コード &6 のため、照会の中間結果を入れるために一時分散結果ファイル &3 が作成されました。理由コードおよびその意味は次のとおりです。 1 - ライブラリー &1 にある &7 &8 のメンバー &2 からのデータが、他のノードに指定されました。 2 - ライブラリー &1 にある &7 &8 のメンバー &2 からのデータが、すべてのノードにブロードキャストされました。 3 - 照会に分散ファイルのパーティション・キーと一致しないグループ化フィールド (GROUP BY) が含まれているか、照会にグループ化基準は含まれているがグループ化フィールドが指定されていないか、または照会に副照会が含まれています。 4 - 照会には、分散ファイルに対する結合基準が含まれており、照会は複数のステップで処理されました。 ライブラリーおよびメンバーの名前 *N は、データが照会の一時分散ファイルからのものであることを示します。 ファイル &3 がノード &9 で作成されました。 パーティション・キー &10 を使用して作成されました。 パーティション・キー *N は、一時分散結果ファイルを作成する際に、パーティション・キーが使用されなかったことを示します。

CPI4345 - 照会プログラム用に一時分散結果ファイル &3 が作成された。	
リカバリー・テキスト:	理由コードが次のとおりです。 1 - 一般に、結合フィールドが分散ファイルのパーティション・キーと一致しない場合、ファイルが指定されます。ファイルが指定されるとき、照会は複数のステップで処理され、並列で処理されます。各ステップの中間結果を入れる一時分散結果ファイルが必要になります。 2 - 一般に、結合フィールドが、結合されているファイルか結合演算子が等価演算子でないファイルのどちらかのパーティション・キーと一致しない場合、ファイルはブロードキャストされます。ファイルがブロードキャストされるとき、照会は複数のステップで、並列で処理されます。各ステップの中間結果を入れる一時分散結果ファイルが必要になります。 3 - グループ化フィールドがパーティション・キーと一致するように指定される場合、さらに高いパフォーマンスを達成できる場合があります。 4 - 照会が複数のステップで処理されるため、各ステップの中間結果を入れる一時分散結果ファイルが必要になります。どのファイルを共に結合したかを判別するには、前のメッセージ CPI4342 を参照してください。 分散ファイルの処理について詳しくは、「分散データベース・プログラミング」を参照してください。

CPI4346 - &2 の QUERY 結合ステップ &1 に対する最適化プログラム・デバッグ・メッセージが続く	
メッセージ・テキスト:	&2 の QUERY 結合ステップ &1 に対する最適化プログラム・デバッグ・メッセージが続く。
原因テキスト:	照会は複数のステップで処理されました。続く最適化プログラム・デバッグ・メッセージは、&2 のすべてのステップの結合ステップ &1 についての Query 最適化情報を提供します。
リカバリー・テキスト:	リカバリーは不要です。

CPI4347 - QUERY は複数ステップで処理されている。	
メッセージ・テキスト:	QUERY は複数ステップで処理されている。
原因テキスト:	元の照会は、複数のステップに細分されます。 各ステップは最適化され、別個に処理されます。各ステップのインプリメンテーションを説明するデバッグ・メッセージは、ジョブ・ログでこのメッセージに続きます。 下記のリストは、この照会で使用するファイルのファイル名を示しています。このリストの項目の形式は、結合ステップの数、照会で指定されるファイル名、照会で指定されるメンバー名、ステップで実際に使用されるファイル名、およびステップで実際に使用されるメンバー名となります。
リカバリー・テキスト:	リカバリーは不要です。

CPI4348 - カーソルと関連した ODP がハード・クローズされた。	
メッセージ・テキスト:	カーソルと関連した ODP がハード・クローズされた。

CPI4348 - カーソルと関連した ODP がハード・クローズされた。	
原因テキスト:	このステートメントまたはカーソルに対するオープン・データ・パス (ODP) は、理由コード &1 のため、ハード・クローズされました。理由コードおよびその意味は次のとおりです。 1 - 新しい LIKE パターンの長さがゼロで、古い LIKE パターンの長さがゼロでないか、新規 LIKE パターンの長さがゼロでなく、古い LIKE パターンの長さがゼロであるかのどちらかです。 2 - 追加のワイルドカードが、カーソルのこの呼び出しの LIKE パターンで指定されました。 3 - SQL は、カーソルが更新できないことを Query 最適化プログラムに示しました。 4 - システム・コードは、照会されているファイル上のロックを取得することができませんでした。 5 - ホスト変数の値の長さが、Query 最適化プログラムによって決定されるようなホスト変数には大きすぎます。 6 - 更新される ODP のサイズが大きすぎます。 7 - 分散照会のローカル ODP の最新表示に失敗しました。 8 - SQL は、ファースト・パスの更新コードの前に、カーソルをハード・クローズしました。
リカバリー・テキスト:	カーソルを再使用可能モードで使用するためには、カーソルをハード・クローズすることはできません。カーソルがハード・クローズされた理由を調べ、ハード・クローズが生じるのを防ぐための適切なアクションを取ります。

CPI4349 - ホスト変数の値の高速パス更新は可能ではない。	
メッセージ・テキスト:	ホスト変数の値の高速パス更新は可能ではない。

CPI4349 - ホスト変数の値の高速パス更新は可能ではない。	
原因テキスト:	このステートメントまたはカーソルに対するオープン・データ・パス (ODP) は、理由コード &1 のため、高速パス更新コードを呼び出すことができませんでした。理由コードおよびその意味は次のとおりです。 1 - 新しいホスト変数の値が非ヌルであり古いホスト変数の値はヌルであるか、または新しいホスト変数の値が長さゼロであり古いホスト変数の値の長さがゼロではありません。 2 - 新しいホスト変数の値の属性が、古いホスト変数の値の属性と同じではありません。 3 - ホスト変数の値の長さが、長すぎるか、短すぎるかのどちらかです。長さの差は、ファースト・パス更新コードでは処理できません。 4 - ホスト変数には IGC ONLY のデータ・タイプがあり、長さは偶数ではないか、または 2 バイト未満です。 5 - ホスト変数には IGC ONLY のデータ・タイプがあり、新しいホスト変数の値には偶数バイトが含まれていません。 6 - 置換文字のある変換テーブルが使用されました。 7 - ホスト変数に、DBCS データと、置換文字が必要な CCSID 変換テーブルが含まれています。 8 - ホスト変数には、十分に形式化されていない DBCS が含まれています。すなわち、シフトアウトのないシフトインか、その逆です。 9 - ホスト変数は分類順序テーブルで変換される必要があり、分類順序テーブルには置換文字が含まれます。 10 - ホスト変数には DBCS データが含まれており、置換文字を含む分類順序テーブルで変換される必要があります。 11 - ホスト変数は日付、時刻、またはタイム・スタンプのデータ・タイプであり、ホスト変数の値の長さは、長すぎるか、短すぎるかのどちらかです。
リカバリー・テキスト:	ファースト・パス更新コードが使用できなかった理由を調べ、このステートメントまたはカーソルを次に呼び出す時に高速パス更新が使用できるように、適切なアクションを取ります。

CPI434 - メンバー &3 は、指定より少ないオープン・オプションで開かれた。	
メッセージ・テキスト:	メンバー &3 は、指定より少ないオープン・オプションで開かれた。
原因テキスト:	INSTEAD OF トリガーが、オープン・オプションの一部に使用されています。ただし、トリガー・アクションが使用できない基礎となる SQL ビュー・ファイル上に追加の INSTEAD OF トリガーがあります。オープン要求は、1 つの SQL ビュー・ファイルのみから INSTEAD OF トリガーをサポートできます。メンバーは、オープン・オプション: &4 によってオープンすることができませんでした。
リカバリー・テキスト:	INSTEAD OF トリガーを追加する際に、要求されたすべてのオープン・オプションに対してトリガー・アクションを指定します。

CPI434E - 照会は、SQE を使用して実行できなかった	
メッセージ・テキスト:	照会は、SQE を使用して実行できなかった。

CPI434E - 照会は、SQE を使用して実行できなかった	
原因テキスト:	照会は、CQE (現行照会エンジン) を使用して実行されました。照会は、理由コード &1 により、SQE (SQL 照会エンジン) を使用して実行できませんでした。理由コードおよびその意味は次のとおりです。 1 -- ライブラリー &3 のソート・シーケンス・テーブル &2 は、SQE によってサポートされない ICU (International Components of Unicode) ソート・シーケンス・テーブルです。
リカバリー・テキスト:	理由コード 1 のリカバリー: SQE を使用して照会を実行するには、&4 以降のバージョンの ICU ソート・シーケンス・テーブルを指定します。

CPI4350 - マテリアライズ照会表の最適化が検討されました。	
メッセージ・テキスト:	マテリアライズ照会表の最適化が検討されました。

CPI4350 - マテリアライズ照会表の最適化が検討されました。	
原因テキスト:	Query 最適化プログラムは、この照会のマテリアライズ照会表の使用を考慮しました。 リストにある以下の各マテリアライズ照会表名は、マテリアライズ照会表が使用されなかった原因を説明する理由コードです。理由コード 0 は、マテリアライズ照会表が照会をインプリメントするのに使用されたことを示します。 理由コードおよびその意味は次のとおりです。 1 - マテリアライズ照会表を使用するためのコストとして最適化プログラムの判別した値が、選択されたインプリメンテーションに関連するコストよりも高くなっています。 2 - マテリアライズ照会で指定された結合には、照会との互換性がありませんでした。 3 - マテリアライズ照会表に、照会の中で一致しなかった述部があります。 4 - マテリアライズ照会表の中で指定されているグループ化または DISTINCT には、照会で指定されているグループ化または DISTINCT との互換性がありません。 5 - マテリアライズ照会表の選択リストに含まれていない列が照会の中で指定されました。 6 - マテリアライズ照会表の照会に、Query 最適化プログラムでサポートされていない機能が含まれています。 7 - マテリアライズ照会表で DISABLE QUERY OPTIMIZATION 文節が指定されました。 8 - マテリアライズ照会表の中で指定されている順序付けには、照会で指定されている順序付けとの互換性がありません。 9 - マテリアライズ照会表のマッチング・アルゴリズムでサポートされていない機能が、照会に含まれています。 10 - この照会では、マテリアライズ照会表を使用できません。 11 - このマテリアライズ照会表の最新表示未実行期間が、MATERIALIZED_QUERY_TABLE_REFRESH_AGE QAQQINI オプションで指定された期間を超えました。 12 - マテリアライズ照会表のコミット・レベルが、照会で指定されているコミット・レベルより低くなっています。 14 - マテリアライズ照会表の FETCH FOR FIRST n ROWS 文節には、照会との互換性がありません。 15 - マテリアライズ照会表の作成に使用された QAQQINI オプションには、この照会を実行するために使用される QAQQINI オプションとの互換性がありません。 16 - マテリアライズ照会表は使用できません。 17 - マテリアライズ照会表で指定された共用体 (UNION) は、照会と互換性がありません。 18 - マテリアライズ照会表で指定された定数は、照会に指定されたホスト変数値と互換性がありません。 19 - マテリアライズ照会表がチェック・ペンディング状況にあり、使用できません。 20 - マテリアライズ照会表で指定された UDTF は、照会の UDTF と互換性がありません。 21 - マテリアライズ照会表の中で指定されている Values 文節には、照会で指定されている Values との互換性がありません。
リカバリー・テキスト:	不要になったマテリアライズ照会表は削除できます。

CPI4351 - 照会アクセス・プランの追加理由コードが再作成された	
メッセージ・テキスト:	照会アクセス・プランの追加理由コードが再作成された。
原因テキスト:	メッセージ CPI4323 がこのメッセージの直前に出されました。メッセージ長に制限があるために、メッセージ CPI4323 で使用している一部の理由コードを、メッセージ中ではなく下記で説明しています。CPI4323 メッセージは、理由コード &13 のために再作成されました。追加理由コードおよびその意味は次のとおりです。 20 - アクセス・プランが生成されてから、ライブラリー &18 にあるファイル &17 のメンバー &19 に対する参照制約またはチェック制約が変更されました。 21 - アクセス・プランが生成されたため、ライブラリー &21 にあるファイル &20 のメンバー &22 のマテリアライズ照会表が変更されました。ファイルが *N の場合、ファイル名は選択不可です。
リカバリー・テキスト:	詳しくは、前のメッセージ CPI4323 を参照してください。

CPI436A - ジョブ &1、モニター ID &2 のデータベース・モニターが開始された	
メッセージ・テキスト:	ジョブ &1、モニター ID &2 のデータベース・モニターが開始された。
原因テキスト:	データベース・モニターがジョブ &1 に対して開始されました。このデータベース・モニターのシステム生成のモニター ID は、&2 です。 複数のモニターが同じ汎用ジョブ名を使用して開始された場合は、ENDDBMON コマンドによって終了するモニターを一意的に識別するために、モニター ID が必要です。
リカバリー・テキスト:	複数のモニターが同じ汎用ジョブ名を使用して開始された場合は、モニター ID を記憶しておきます。モニター ID は、ENDDBMON コマンドを使用してこの特定のモニターを終了する際に必要です。

Query 最適化パフォーマンス通知メッセージおよびオープン・データ・パス

以下のいくつかの SQL 実行時メッセージは、オープン・データ・パスに関するものです。

オープン・データ・パス (ODP) 定義は、カーソルがオープンされるとき、または他の SQL ステートメントが実行されるときに作成される内部オブジェクトです。 ODP はデータとの直接リンクを提供し、入出力操作を可能にします。 ODP は OPEN、INSERT、UPDATE、DELETE、および SELECT INTO の各ステートメントで使用されて、データに対してそれぞれの操作を行います。

SQL カーソルがクローズされ、SQL ステートメントがすでに実行されている場合でも、データベース・マネージャーは、多くの場合、ステートメントの次回の実行時に使用するために SQL 操作の関連 ODP を保管します。したがって、SQL の CLOSE ステートメントによって SQL カーソルがクローズされても、ODP は使用可能のままであり、カーソルの次回のオープン時に使用することができます。これによって、SQL ステートメントの実行時の処理時間と応答時間が大幅に短縮できます。

SQL ステートメントが繰り返し実行される場合に ODP を再使用できることは、パフォーマンス向上の上で重要な考慮事項となります。

SQL7910 - SQL カーソルがクローズされた	
メッセージ・テキスト:	SQL カーソルがクローズされた。
原因テキスト:	SQL カーソルがクローズされ、CLOSQLCSR(*ENDJOB) オプションによってプログラムがオープンしたもの、または CLOSQLCSR(*ENDACTGRP) オプションによってモジュールがオープンしたものを除いて、すべてのオープン・データ・パス (ODP) が削除されました。呼び出しスタックにあるすべての SQL プログラムが完了し、SQL 環境が終了しました。この処理には、カーソルのクローズ、ODP の削除、準備されたステートメントの除去、およびロック解除があります。
リカバリー・テキスト:	プログラムの完了後もカーソル、ODP、作成されたステートメント、およびロックを使用できるようにするには、CLOSQLCSR プリコンパイル・パラメーターを使用します。 -- *ENDJOB オプションによってユーザーは、ジョブが持続する間 SQL リソースをアクティブに保つことができます。 -- SQL 環境が存在する場合、*ENDSQL オプションによってユーザーは、プログラム呼び出しに対して SQL リソースをアクティブに保つことができます。アプリケーションの最初のプログラムにある SQL ステートメントを実行すると、そのアプリケーションが持続する間、SQL 環境は活動状態になっています。 -- *ENDPGM オプション (非統合言語環境® (ILE) プログラムではデフォルト) を指定すると、SQL リソースへのアクセスはすべてプログラムの同じ呼び出しによってのみ可能になります。いったん *ENDPGM プログラムが完了したなら、もう一度呼び出される場合でも、SQL リソースはアクティブではありません。 -- *ENDMOD オプションは、すべての SQL リソースを同じモジュール呼び出しによってのみアクセス可能にします。 -- *ENDACTGRP オプション (ILE モジュールではデフォルトである) を使用すると、ユーザーは、活動化グループが持続する間 SQL リソースをアクティブに保つことができます。

SQL7911 - ODP が再使用された	
メッセージ・テキスト:	ODP が再使用された。
原因テキスト:	以前に作成された ODP が再使用されました。この SQL ステートメント用に検出された再使用可能なオープン・データ・パス (ODP) があり、それが使用されました。再使用可能な ODP は、同じプログラムの呼び出しから、または前のプログラムの呼び出しからである場合があります。 ODP の再使用は、ジャーナルに OPEN 項目を生成しません。
リカバリー・テキスト:	ありません。

SQL7912 - ODP が作成された	
メッセージ・テキスト:	ODP が作成された。

SQL7912 - ODP が作成された	
原因テキスト:	オープン・データ・パス (ODP) が作成されました。再使用可能な ODP は検出できませんでした。これは、次の場合に起こります。 -- ステートメントが実行されたのが初めてである。 -- このステートメントが最後に実行された後に、RCLRSC が発行されている。 -- ステートメントの最後の実行によって、ODP が削除された。 -- これが OPEN ステートメントである場合、このカーソルの最後の CLOSE によって ODP が削除された。 -- アプリケーション・サーバー (AS) が CONNECT ステートメントによって変更された。
リカバリー・テキスト:	1 つのアプリケーションでカーソルが何度もオープンされている場合、再使用可能な ODP を使用して、毎回 ODP を作成しないようにすると効果的です。これは、INSERT、UPDATE、DELETE、および SELECT INTO ステートメントを繰り返し実行することにも適用できます。 ODPが毎回オープンされるときに作成される場合、ODP が削除される理由を判別するためにクローズ・メッセージを参照してください。

ステートメントが最初に実行されるとき、またはカーソルが処理のためにオープンされるときには、常に ODP を作成する必要があります。しかし、ステートメントを実行するたびに、またはカーソルをオープンするたびにこのメッセージが表示される場合には、 204 ページの『非 ILE プログラム呼び出しの場合のカーソル位置の保存』 で推奨されているヒントを、このアプリケーションに適用してください。

SQL7913 - ODP が削除された	
メッセージ・テキスト:	ODP が削除された。
原因テキスト:	このステートメントまたはカーソルに対するオープン・データ・パス (ODP) は、削除されました。 ODP は再使用可能ではありませんでした。これは、LIKE 文節のホスト変数を使用することによって、ホスト変数を順序付けることによって、または Query 最適化プログラムが再使用可能でない ODP で照会を遂行するよう選択したために、起こりました。
リカバリー・テキスト:	どのようにカーソルがオープンされたかを判別するために、直前の Query 最適化プログラムのメッセージを参照してください。

SQL7914 - ODP は削除されなかった	
メッセージ・テキスト:	ODP は削除されなかった。
原因テキスト:	このステートメントまたはカーソルに対するオープン・データ・パス (ODP) は、削除されませんでした。この ODP は、後のステートメントの実行で再使用することができます。これは、ジャーナルに項目を生成しません。
リカバリー・テキスト:	ありません。

SQL7915 - SQL ステートメントのアクセス・プランが作成された	
メッセージ・テキスト:	SQL ステートメントのアクセス・プランが作成された。

SQL7915 - SQL ステートメントのアクセス・プランが作成された	
原因テキスト:	SQL は実行時に、このステートメントのアクセス・プランを作成する必要がありました。これは、次の場合に起こります。 -- プログラムが異なるリリースから復元され、このステートメントが初めて実行された。 -- ステートメントに必要なすべてのファイルがプリコンパイル時に存在せず、このステートメントが初めて実行された。 -- プログラムが SQL 命名モードを使用してプリコンパイルされ、最後にプログラムが呼び出されて以来、プログラム所有者が変更されている。
リカバリー・テキスト:	これは、SQL の通常の処理です。いったんアクセス・プランが作成されると、後のステートメントの実行に使用されます。

SQL7916 - 照会でブロック化が使用された	
メッセージ・テキスト:	照会でブロック化が使用された。
原因テキスト:	この照会のインプリメンテーションで、ブロック化が使用されました。SQL は、最初の FETCH ステートメントのデータベース・マネージャーからレコードのブロックを取り出します。追加の FETCH ステートメントが呼び出し側プログラムによって発行されなければなりませんが、SQL にさらに多くのレコードを要求する必要はないため、さらに高速で実行します。
リカバリー・テキスト:	SQL は可能な時には常に、ブロック化を使用しようとします。カーソルが更新可能でなく、コミットメント制御がアクティブでない場合、ブロック化が使用される可能性があります。

SQL7917 - アクセス・プランは更新されなかった	
メッセージ・テキスト:	アクセス・プランは更新されなかった。
原因テキスト:	Query 最適化プログラムはこのステートメント用のアクセス・プランを再作成しますが、プログラムは更新できませんでした。他のジョブがプログラムを実行している可能性があります。このプログラムは、ジョブがプログラムに排他ロックを得るまで、新しいアクセス・プランで更新することはできません。他のジョブがプログラムを実行している場合、ジョブがプログラムを実行する適切な権限を持っていない場合、またはプログラムが現在保管されている場合、排他ロックを得ることはできません。照会は依然として実行しますが、アクセス・プランの再作成は、プログラムが更新されるまで続きます。
リカバリー・テキスト:	アクセス・プランが再作成された理由を判別するために、Query 最適化プログラムからの直前のメッセージを参照してください。プログラムが新規アクセス・プランで更新されることを確認するには、他のアクティブ・ジョブが使用していないときにプログラムを実行してください。

SQL7918 - 再使用可能な ODP が削除された	
メッセージ・テキスト:	再使用可能な ODP が削除された。理由コード &1。

SQL7918 - 再使用可能な ODP が削除された	
原因テキスト:	既存のオープン・データ・パス (ODP) がこのステートメント用に検出されましたが、理由コード &1 のため、再使用できませんでした。この場合、ステートメントは ODP にあるものとは別のファイルを参照するか、または別の一時変更オプションを使用します。理由コードおよびその意味は次のとおりです。 1 -- コミットメント制御の分離レベルには、互換性がありません。 2 -- ステートメントには、SQL 特殊レジスター USER、CURRENT DEBUG MODE、CURRENT DECFLOAT ROUNDING MODE、または CURRENT TIMEZONE が含まれており、これらのレジスターの 1 つの値が変更されました。 3 -- SQL 関数を位置指定するのに使用される PATH が変更されました。 4 -- ジョブのデフォルトである CCSID が変更されました。 5 -- ライブラリー・リストが変更されました。例えば、ファイルが異なるライブラリーで検出されました。テーブルが複数のライブラリーに存在するとき、これは、非修飾テーブル名のステートメントにのみ影響します。 6 -- 元の ODP のファイル、ライブラリー、またはメンバーがオーバーライドで変更されました。 7 -- OVRDBF または DLTOVR コマンドが出されました。ステートメントで参照されているファイルが、異なるファイル、ライブラリー、またはメンバーを参照しています。 8 -- OVRDBF または DLTOVR コマンドが発行され、異なる SEQONLY または WAITRCD 値など、異なるさまざまなオーバーライド・オプションとなりました。 9 -- ステートメントのオーバーライド情報が再利用できる ODP 情報と互換性があるかどうかを検査しようとして、エラーが発生しました。 10 -- Query 最適化プログラムは、ODP が再使用できないと判別しました。 11 -- クライアント・アプリケーションは、ODP を再使用しないように要求しました。
リカバリー・テキスト:	再使用可能な ODP を使用する場合、ライブラリー・リスト、オーバーライド環境、または特殊レジスターの値を変更しないでください。

SQL7919 - FETCH または挿入された SELECT にはデータ変換が必要である	
メッセージ・テキスト:	FETCH または挿入された SELECT にはデータ変換が必要である。

SQL7919 - FETCH または挿入された SELECT にはデータ変換が必要である	
原因テキスト:	ホスト変数 &2 は変換が要です。 FETCH または挿入された SELECT ステートメントのために取り出されるデータは、ホスト変数に直接に移すことはできません。ステートメントは正しく実行されました。ただし、データ変換が必要ない場合に、パフォーマンスが向上します。理由 &1 のため、ホスト変数は変換が必要です。 -- 理由 1 - ホスト変数 &2 は、取り出される値と異なる長さの文字またはグラフィック・ストリングです。 -- 理由 2 - ホスト変数 &2 は、取り出される値のタイプと異なる数値タイプです。 -- 理由 3 - ホスト変数 &2 は、NUL で終了する C 文字または C グラフィック・ストリングであり、プログラムは指定されたオプション *CNULRQD でコンパイルされ、ステートメントは複数行 FETCH です。 -- 理由 4 - ホスト変数 &2 は可変長ストリングですが、取り出される値はそうではありません。 -- 理由 5 - ホスト変数 &2 は可変長ストリングではなく、取り出される値はそうです。 -- 理由 6 - ホスト変数 &2 は、最大の長さが取り出される可変長の値の最大の長さとは異なる、可変長ストリングです。 -- 理由 7 - ホスト変数 &2 に取り出される値のマッピングで、CCSID 変換などのデータ変換が必要とされました。 -- 理由 8 - DRDA 接続が、ホスト変数 &2 に取り出される値を取得するために使用されました。取り出される値は、ヌル可能か可変長である、部分行に含まれている、または派生式です。 -- 理由 10 - ホスト変数 &2 の長さが、取り出される TIME または TIMESTAMP 値を保持するには短すぎます。 -- 理由 11 - ホスト変数 &2 は DATE、TIME、または TIMESTAMP タイプであり、取り出される値は文字ストリングです。 -- 理由 12 - 多すぎるホスト変数が指定され、レコードがブロック化されています。ホスト変数 &2 には、照会から戻された対応する列がありません。 -- 理由 13 - DRDA 接続がブロック化された FETCH に使用され、INTO 文節に指定されたホスト変数の数は、選択リストにある結果値の数より少ないです。 -- 理由 14 - LOB ロケーターが使用され、プロセスのコミットメント制御レベルは *ALL ではありません。
リカバリー・テキスト:	さらに高いパフォーマンスを得るには、対応する結果列とタイプと長さが同じホスト変数を使用してください。

SQL7939 - INSERT または UPDATE 時にデータ変換が必要である	
メッセージ・テキスト:	INSERT または UPDATE 時にデータ変換が必要である。

SQL7939 - INSERT または UPDATE 時にデータ変換が必要である	
原因テキスト:	INSERT または UPDATE 値は、列の 1 つとデータ・タイプまたは値の長さが異なるため、列に直接に移すことはできません。 INSERT または UPDATE ステートメントは正しく実行されました。ただし、データ変換が必要ない場合に、パフォーマンスが向上します。データ変換が必要な理由は &1 です。 -- 理由 1 は、INSERT または UPDATE の値が、列 &2 と異なる長さの文字またはグラフィック・ストリングであることを示します。 -- 理由 2 は、INSERT または UPDATE の値が、列 &2 のタイプと異なる数値タイプであることを示します。 -- 理由 3 は、INSERT または UPDATE の値が可変長ストリングであり、列 &2 はそうでないことを示します。 -- 理由 4 は、INSERT または UPDATE の値が可変長ストリングではなく、列 &2 はそうであることを示します。 -- 理由 5 は、INSERT または UPDATE の値が、最大の長さが列 &2 の最大の長さと異なる可変長ストリングであることを示します。 -- 理由 6 - 列 &2 への INSERT または UPDATE の値のマッピングで、CCSID 変換などのデータ変換が必要であることを示します。 -- 理由 7 は、INSERT または UPDATE の値が文字ストリングであり、列 &2 は DATE、TIME、または TIMESTAMP タイプであることを示します。 -- 理由 8 は、INSERT のターゲット・テーブルが SQL テーブルではないことを示します。
リカバリー・テキスト:	さらに良いパフォーマンスを得るには、対応する列とタイプと長さが同じ値を使用してください。

PRTSQLINF メッセージ解説

以下に、PRTSQLINF から戻されるメッセージを示します。

SQL400A - 一時分散結果ファイル &1 が作成され、結合の結果が入れられた	
メッセージ・テキスト:	一時分散結果ファイル &1 が作成され、結合の結果が入れられた。結果ファイルが出力先指定されました。
原因テキスト:	照会には、分散ファイルに対する結合基準が含まれており、分散結合は並列に実行されました。分散結合の結果を入れるために一時分散結果ファイルが作成されました。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL400B - 一時分散結果ファイル &1 が作成され、結合の結果が入れられた	
メッセージ・テキスト:	一時分散結果ファイル &1 が作成され、結合の結果が入れられた。結果ファイルがブロードキャストされました。
原因テキスト:	照会には、分散ファイルに対する結合基準が含まれており、分散結合は並列に実行されました。分散結合の結果を入れるために一時分散結果ファイルが作成されました。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL400C - &2 の Query 定義ステップ &1 の最適化プログラム・デバッグ・メッセージは次の通りである	
メッセージ・テキスト:	&2 の分散照会ステップ &1 の最適化プログラム・デバッグ・メッセージは次のとおり。
原因テキスト:	分散ファイルが照会に指定され、複数のステップで照会が処理されます。続く最適化プログラム・デバッグ・メッセージは、現行のステップについての Query 最適化情報を提供します。
リカバリー・テキスト:	分散ファイルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL400D - GROUP BY 処理が生成された	
メッセージ・テキスト:	GROUP BY 処理が生成された。
原因テキスト:	GROUP BY 処理が照会のステップに追加された。 GROUP BY 処理の追加は、結果行の数を減らすため、後続のステップのパフォーマンスを向上させます。
リカバリー・テキスト:	詳しくは、「SQL プログラミング」トピック・コレクションを参照してください。

SQL400E - 分散 SUBQUERY の処理中に一時分散結果ファイル &1 が作成された	
メッセージ・テキスト:	分散 SUBQUERY の処理中に一時分散結果ファイル &1 が作成された。
原因テキスト:	照会の中間結果を入れるために一時分散結果ファイルが作成されました。照会には、中間結果を必要とする副照会が含まれます。
リカバリー・テキスト:	一般に、照会と副照会の間で相関関係のあるフィールドが、それぞれのファイルの区分化キーと一致しない場合、照会は複数ステップで処理されなければならず、一時分散ファイルが中間結果を入れるために作成されます。分散ファイルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL4001 - 一時結果が作成されます	
メッセージ・テキスト:	一時結果が作成されます。
原因テキスト:	一時結果が作成されるようにする条件が照会に存在します。次の理由の 1 つが、一時結果の原因となる場合があります。 -- テーブルが結合論理ファイルであって、その結合タイプ (JDFTVAL) が照会に指定された結合タイプと一致しません。 -- 論理ファイルに指定された形式が複数の物理テーブルを参照します。 -- テーブルは、複合 SQL ビューであって、その SQL ビューの結果を入れるための一時テーブルを必要としています。 -- 照会には、複数のテーブルからのグループ化列 (GROUP BY) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからのグループ化列が含まれていたりします。
リカバリー・テキスト:	一時結果を避けるよう照会を変更できる場合、パフォーマンスが向上する場合があります。

SQL4002 - 再利用できる ODP ソートが使用された	
メッセージ・テキスト:	再利用できる ODP ソートが使用された。
原因テキスト:	分類が使用されるようにする条件が照会に存在します。これによって、オープン・データ・パス (ODP) は再使用可能になります。次の理由の 1 つが、ソートの原因となる場合があります。 -- 照会には、複数のテーブルからの順序付け列 (ORDER BY) が含まれていたり、または再順序付けができない結合照会の 2 次テーブルからの順序付け列が含まれていたりします。 -- グループ化列と順序付け列には、互換性がありません。 -- DISTINCT が照会に対して指定されました。 -- UNION が照会に対して指定されました。 -- 照会は、分類を使用して実装しなければなりませんでした。キーの長さが 2000 バイトを超える、120 を超える順序付け列、または外部のユーザー定義関数の参照を含む順序付け列が、順序付けで指定されました。 -- Query 最適化プログラムは、照会の結果を順序付けする索引ではなくて、分類を使用することを選択しました。
リカバリー・テキスト:	再使用可能な ODP は一般に、再使用不可な ODP と比較してパフォーマンスが向上します。

SQL4003 - UNION	
メッセージ・テキスト:	UNION、EXCEPT、または INTERSECT。
原因テキスト:	演算子 UNION、EXCEPT、または INTERSECT が照会で指定されました。このキーワード区切り文字に先行するメッセージは、UNION、EXCEPT、または INTERSECT 演算子に先行する副選択に対応します。このキーワード区切り文字に続くメッセージは、UNION、EXCEPT、または INTERSECT 演算子に続く副選択に対応します。
リカバリー・テキスト:	ありません。

SQL4004 - SUBQUERY	
メッセージ・テキスト:	SUBQUERY。
原因テキスト:	SQL ステートメントには副照会が含まれています。 SUBQUERY 区切り文字に先行するメッセージは、副照会を含む副選択に対応します。 SUBQUERY 区切り文字に続くメッセージは、副照会に対応します。
リカバリー・テキスト:	ありません。

SQL4005 - テーブル &1 の Query 最適化プログラムがタイムアウトになりました	
メッセージ・テキスト:	テーブル &1 の Query 最適化プログラムがタイムアウトになりました。

SQL4005 - テーブル &1 の Query 最適化プログラムがタイムアウトになりました	
原因テキスト:	テーブルに対して作成されたすべての索引を考慮する前に、 Query 最適化プログラムはタイムアウトになりました。これはエラー状態ではありません。 Query 最適化プログラムは、最適化の時間を最小限に抑えるためにタイムアウトになる場合があります。照会をデバッグ・モード (STRDBG) で実行すると、最適化中に考慮した索引のリストを参照できます。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	最適化のために索引が必ず考慮されるようにするには、その索引の論理ファイルを照会されるテーブルとして指定してください。最適化プログラムは、SQL 選択ステートメントで指定された論理ファイルの索引を最初に考慮します。 SQL 作成済み索引は照会できないことに注意してください。SQL 索引は、照会の最適化中に考慮されるチャンスを増やすために、削除して再作成することができます。必要のない索引の削除を考慮してください。

SQL4006 - テーブル &1 のすべての索引が検討されました	
メッセージ・テキスト:	テーブル &1 のすべての索引が検討されました。
原因テキスト:	Query 最適化プログラムは、照会を最適化するときに、テーブルに作成されたすべての索引を考慮しました。照会をデバッグ・モード (STRDBG) で実行すると、最適化中に考慮した索引のリストを参照できます。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	ありません。

SQL4007 - テーブル &2 の結合位置 &1 に対する照会実装	
メッセージ・テキスト:	テーブル &2 の結合位置 &1 に対する照会実装。
原因テキスト:	結合位置は、テーブルが結合される順序を識別します。結合位置 1 は、このテーブルが結合順序において最初 (左端) のテーブルであることを示します。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	結合順序は、照会に ORDER BY 文節を追加することによって影響されます。結合最適化、および結合順序に影響を及ぼすヒントについて詳しくは、 50 ページの『結合の最適化』を参照してください。

SQL4008 - テーブル &2 に索引 &1 が使用されました	
メッセージ・テキスト:	テーブル &2 に索引 &1 が使用されました。
原因テキスト:	次の理由のいずれかのため、テーブルから行にアクセスするために索引が使用されました。 -- 行選択。 -- 結合基準。 -- 順序付け/グループ化基準。 -- 行選択、および順序付け/グループ化基準。 テーブル番号は、照会のこのテーブルの相対位置を参照します。 照会をデバッグ・モード (STRDBG) で実行すると、索引が使用された特定の理由を判別できます。

SQL4008 - テーブル &2 に索引 &1 が使用されました	
リカバリー・テキスト:	ありません。

SQL4009 - テーブル &1 の索引が作成されました	
メッセージ・テキスト:	テーブル &1 の索引が作成されました。
原因テキスト:	次の理由のいずれかのため、テーブルから行にアクセスするために一時索引が作成されました。 -- 指定された順序付け/グループ化基準を実行します。 -- 指定された結合基準を実行します。 テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	パフォーマンスを向上するには、照会が頻繁に実行されている場合には、永続索引の作成を検討します。照会をデバッグ・モード (STRDBG) で実行すると、索引が作成された特定の理由や、索引の作成時に使用するキー列を判別できます。注: 永続索引が作成される場合、Query 最適化プログラムは、テーブルから行にアクセスするために引き続き一時索引を作成するよう選択する場合があります。

SQL401A - 分散テーブルが入っている QUERY のグループ化基準を処理中	
メッセージ・テキスト:	分散テーブルが入っている QUERY のグループ化基準を処理中。
原因テキスト:	分散テーブルを含む照会のグループ化は、1 ステップ方式か 2 ステップ方式を使用して実行されます。 1 ステップ方式が使用される場合、グループ化列 (GROUP BY) は分散テーブルの区分化キーと一致します。 2 ステップ方式が使用される場合、グループ化列は分散テーブルの区分化キーと一致しないか、照会はグループ化基準を含んでいますがグループ化列が指定されませんでした。 2 ステップ方式が使用される場合、メッセージ SQL401B は他の SQL401A メッセージが後に続いて表示されます。
リカバリー・テキスト:	分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL401B - グループ化基準の処理時に一時分散結果テーブル &1 が作成されました	
メッセージ・テキスト:	グループ化基準の処理時に一時分散結果テーブル &1 が作成されました。
原因テキスト:	照会の中間結果を入れるために一時分散結果テーブルが作成されました。照会は分散テーブルの区分化キーと一致しないグループ化列 (GROUP BY) を含むか、照会はグループ化基準を含みますがグループ化列は指定されませんでした。
リカバリー・テキスト:	分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL401C - 照会の分散結合の実行中	
メッセージ・テキスト:	照会の分散結合の実行中。
原因テキスト:	照会には分散テーブルに対する結合基準が含まれており、分散結合は並列に実行されました。どのテーブルが共に結合されるかを判別するために、続く SQL401F メッセージを参照してください。

SQL401C - 照会の分散結合の実行中	
リカバリー・テキスト:	分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL401D - テーブル &2 が出力先指定されたので、一時分散結果テーブル &1 が作成されました	
メッセージ・テキスト:	テーブル &2 が出力先指定されたので、一時分散結果テーブル &1 が作成されました。
原因テキスト:	照会の中間結果を入れるために一時分散結果テーブルが作成されました。照会の分散テーブルからのデータが、他のノードに指示されました。
リカバリー・テキスト:	一般に、結合列が分散テーブルの区分化キーと一致しないとき、テーブルは指定されます。テーブルが指定されるとき、照会は複数のステップで処理され、並列で処理されます。各ステップの中間結果を入れる一時分散結果ファイルが必要になります。分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL401E - テーブル &2 がブロードキャストされたので、一時分散結果テーブル &1 が作成されました	
メッセージ・テキスト:	テーブル &2 がブロードキャストされたので、一時分散結果テーブル &1 が作成されました。
原因テキスト:	照会の中間結果を入れるために一時分散結果テーブルが作成されました。照会の分散テーブルからのデータが、すべてのノードにブロードキャストされました。
リカバリー・テキスト:	一般に、結合列が、結合されているテーブルか結合演算子が等価演算子でないテーブルのどちらかの区分化キーと一致しないとき、テーブルはブロードキャストされます。テーブルがブロードキャストされるとき、照会は複数のステップで処理され、並列で処理されます。一時分散結果テーブルは、各ステップの中間結果を含むよう要求されます。分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL401F - テーブル &1 が分散結合に使用されました	
メッセージ・テキスト:	テーブル &1 が分散結合に使用されました。
原因テキスト:	照会には分散テーブルに対する結合基準が含まれており、分散結合は並列に実行されました。
リカバリー・テキスト:	分散テーブルの処理について詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL4010 - テーブル &1 に対するテーブル・スキャン・アクセス	
メッセージ・テキスト:	テーブル &1 に対するテーブル・スキャン・アクセス。
原因テキスト:	テーブルから行を選択するためにテーブル・スキャン・アクセスが使用されました。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	そのテーブルから高い割合で行を選択するとき、テーブル走査は一般的に効率の良いオプションです。ただし、テーブルから選択される行の割合が低い場合、索引を使用する方が照会のパフォーマンスを向上する場合があります。

SQL4011 - テーブル &1 に索引スキャン・キー行位置が使用されました	
メッセージ・テキスト:	テーブル &1 に索引スキャン・キー行位置が使用されました。

SQL4011 - テーブル &1 に索引スキャン・キー行位置が使用されました	
原因テキスト:	索引走査キー行の位置決めは、索引に対して選択を適用して、選択基準の一部またはすべてに一致するキーの範囲に直接置かれるように定義されます。索引走査キー行位置は、索引のキーのサブセットのみを処理し、そのテーブルから選択される行の割合が低い場合に効率の良いオプションとなります。 テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	索引走査キー行位置について詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4012 - テーブル &2 の索引 &1 から索引が作成されました	
メッセージ・テキスト:	テーブル &2 の索引 &1 から索引が作成されました。
原因テキスト:	次の理由のいずれかのため、照会されたテーブルの行にアクセスするために、指定された索引を使用して一時索引が作成されました。 -- 指定された順序付け/グループ化基準を実行します。 -- 指定された結合基準を実行します。 テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	索引から索引を作成することは、一般的に効率の良いオプションとなります。頻繁に実行される照会のために永続索引を作成することを考慮してください。照会をデバッグ・モード (STRDBG) で実行すると、索引の作成時に使用するキー列を判別できます。注: 永続索引が作成される場合、Query 最適化プログラムは、テーブルから行にアクセスするために引き続き一時索引を作成するよう選択する場合があります。

SQL4013 - アクセス・プランが作成されていない	
メッセージ・テキスト:	アクセス・プランが作成されていない。
原因テキスト:	この照会にはアクセス・プランは作成されませんでした。次の理由が考えられます。 -- プログラムが作成されたときに、テーブルが検出されませんでした。 -- 照会は複雑で、一時結果テーブルが必要になりました。 -- 動的 SQL が指定されました。
リカバリー・テキスト:	アクセス・プランが作成されなかった場合、考えられる原因を検討します。可能なら、問題を解決を試行します。

SQL4014 - この結合位置に &1 結合列のペアが使用されました	
メッセージ・テキスト:	この結合位置に &1 結合列のペアが使用されました。

SQL4014 - この結合位置に &1 結合列のペアが使用されました	
原因テキスト:	Query 最適化プログラムは、結合選択か行選択として結合述部を処理することを選択する場合があります。結合選択で使用される結合述部は、最後の結合順序および使用される索引によって決定されます。このメッセージは、この結合位置で結合選択として、いくつの結合列のペアが処理されるかを示します。メッセージ SQL4015 は、列が結合列のペアを構成していることについての詳細を提供します。 結合列のペアが指定されなかった場合、行選択のある索引走査キー行位置が、結合選択の代わりに使用されます。
リカバリー・テキスト:	予想より少ない結合のペアが結合位置で使用される場合、要求される結合列と一致するキーを持つ索引が存在しない可能性もあります。キーが結合述部と一致する索引の作成を試行してください。 結合列のペアが指定されなかった場合、索引走査キー行位置が使用されました。索引走査キー行位置は、通常効率の良いオプションとなります。メッセージ SQL4011 は、索引走査キー行位置についての詳しい情報を提供します。

SQL4015 - 結合元列 &1.&2、結合先列 &3.&4、結合演算子 &5、結合述部 &6	
メッセージ・テキスト:	結合元列 &1.&2、結合先列 &3.&4、結合演算子 &5、結合述部 &6。
原因テキスト:	現行の結合位置でどの結合述部がインプリメントされたかを示します。置換テキスト・パラメーターは次のとおりです。 -- &1: 結合「元テーブル」番号。テーブル番号は、照会のこのテーブルの相対位置を参照します。 -- &2: 結合「元列」名。結合列のペアの左半分を構成する結合元テーブル内の列。列名が *MAP である場合、列は式 (派生したフィールド) です。 -- &3: 結合「先テーブル」番号。テーブル番号は、照会のこのテーブルの相対位置を参照します。 -- &4. 結合「先列」名。結合列のペアの右半分を構成する結合先列内の列。列名が *MAP である場合、列は式 (派生したフィールド) です。 -- &5. 結合演算子。可能な値は EQ (等しい)、NE (等しくない)、GT (より大きい)、LT (より小)、GE (より大きいか等しい)、 LE (より小か等しい)、および CP (相互結合またはカルテシアン積) です。 -- &6. 結合述部番号。結合のペアのこのセット内にある結合述部を識別します。
リカバリー・テキスト:	結合について詳しくは、 50 ページの『結合の最適化』を参照してください。

SQL4016 - 副選択が結合 QUERY として処理された	
メッセージ・テキスト:	副選択が結合 QUERY として処理された。
原因テキスト:	Query 最適化プログラムは、結合照会で副選択の一部かすべてを実行することを選択しました。結合で副照会を実行することは、一般的に代替方式よりもパフォーマンスを向上させます。
リカバリー・テキスト:	ありません。

SQL4017 - ホスト変数が再利用できる ODP として実装された	
メッセージ・テキスト:	ホスト変数が再利用できる ODP として実装された。
原因テキスト:	Query 最適化プログラムは、照会がオープンされるときにホスト変数の値が提供されるのを許可するアクセス・プランを作成しました。この照会は、アクセス・プランの再作成を必要とすることなく、ホスト変数に異なる値を提供して実行できます。これは、アクセス・プランでホスト変数を処理する通常の方式です。このアクセス・プランから作成されるオープン・データ・パス (ODP) は、再使用可能な ODP です。
リカバリー・テキスト:	一般に、再使用可能なオープン・データ・パスは、再使用不可のオープン・データ・パスよりも効率良く実行します。

SQL4018 - ホスト変数が再利用できない ODP として実装された	
メッセージ・テキスト:	ホスト変数が再利用できない ODP として実装された。
原因テキスト:	Query 最適化プログラムは、再使用可能でないオープン・データ・パス (ODP) でホスト変数を実行しました。
リカバリー・テキスト:	これは、特定の環境では効率の良いオプションとなりますが、一般的には再使用可能な ODP が最良のパフォーマンスを提供します。

SQL4019 - ホスト変数がファイル管理行位置指定再使用可能 ODP として実装された	
メッセージ・テキスト:	ホスト変数がファイル管理行位置指定再使用可能 ODP として実装された
原因テキスト:	Query 最適化プログラムは、ファイル管理行位置を使用して、再利用できるオープン・データ・パス (ODP) でホスト変数を実装しました。
リカバリー・テキスト:	一般に、再使用可能 ODP は再使用不可 ODPより効率良く実行します。

SQL402A - 結合の処理にハッシュ・アルゴリズムが使用された	
メッセージ・テキスト:	結合の処理にハッシュ・アルゴリズムが使用された。
原因テキスト:	ハッシュ結合アルゴリズムは一般的に、長く実行される結合照会のために使用されます。元の照会は、ハッシュ結合ステップに細分化されます。各ハッシュ結合ステップは最適化され、別個に処理されます。アクセス・プランは個々のハッシュ結合ダイヤルのために保管されていないため、各ハッシュ結合ステップについてのアクセス・プランのインプリメンテーション情報は使用できません。照会が STRDBG CL コマンドを使用してデバッグ・モードで実行される場合、各ハッシュ・ダイヤルのインプリメンテーションを詳述するデバッグ・メッセージがジョブ・ログに記録されます。
リカバリー・テキスト:	ハッシュ結合方式は、通常良い選択です。ただし、この方式の使用を許可したくない場合は、ALWCPYDTA(*YES) を指定します。結合処理のハッシュ・アルゴリズムについて詳しくは、&qryopt を参照してください。

SQL402B - ハッシュ結合ステップ &2 でテーブル &1 が使用されました	
メッセージ・テキスト:	ハッシュ結合ステップ &2 でテーブル &1 が使用されました。

SQL402B - ハッシュ結合ステップ &2 でテーブル &1 が使用されました	
原因テキスト:	このメッセージは、ハッシュ結合ステップによって使用されるテーブルの番号をリストします。テーブル番号は、照会のこのテーブルの相対位置を参照します。同じハッシュ結合ステップにこれら複数のメッセージがある場合、そのステップはネストされたループ結合です。アクセス・プランは個々のハッシュ・ステップのために保管されていないため、各ハッシュ結合ステップについてのアクセス・プランのインプリメンテーション情報は使用できません。照会が STRDBG CL コマンドを使用してデバッグ・モードで実行される場合、各ハッシュ・ステップのインプリメンテーションを詳述するデバッグ・メッセージがジョブ・ログに記録されます。
リカバリー・テキスト:	ハッシュについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL402C - ハッシュ結合結果に一時テーブルが作成されました	
メッセージ・テキスト:	ハッシュ結合結果に一時テーブルが作成されました。
原因テキスト:	照会の処理が完了するために、ハッシュ結合の結果が一時テーブルに書き込まれました。照会が次の 1 つ以上を含むため、一時テーブルが必要とされました。 GROUP BY または集計機能、ORDER BY、DISTINCT、複数のテーブルからの列を含む式、複数のテーブルからの列を含む複合行選択
リカバリー・テキスト:	結合処理のハッシュ・アルゴリズムについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL402D - QUERY 属性がライブラリー &1 の QUERY オプション・ファイル &2 から一時変更された	
メッセージ・テキスト:	QUERY 属性がライブラリー &1 の QUERY オプション・ファイル &2 から一時変更された。
原因テキスト:	ありません。
リカバリー・テキスト:	ありません。

SQL4020 - 見積 QUERY 実行時間は &1 秒である	
メッセージ・テキスト:	見積 QUERY 実行時間は &1 秒である。
原因テキスト:	この照会の実行の合計見積時間 (秒単位) です。
リカバリー・テキスト:	ありません。

SQL4021 - アクセス・プランが最後に保管された &1 の &2 です	
メッセージ・テキスト:	アクセス・プランが最後に保管された &1 の &2 です。
原因テキスト:	日時は、プログラム・オブジェクトでアクセス・プランが最後に正常に更新された時を表します。
リカバリー・テキスト:	ありません。

SQL4022 - アクセス・プランは SRVQRY 属性が活動状態で保管された	
メッセージ・テキスト:	アクセス・プランは SRVQRY 属性が活動状態で保管された。

SQL4022 - アクセス・プランは SRVQRY 属性が活動状態で保管された	
原因テキスト:	SRVQRY がアクティブである間に、保管されたアクセス・プランが作成されました。アクセス・プランに保管された属性は SRVQRY の結果である場合があります。
リカバリー・テキスト:	SRVQRY の属性は永続的に保管されないため、照会は次に実行されるときに再最適化されます。

SQL4023 - 並行テーブルの事前取り出しが使用されました。	
メッセージ・テキスト:	並行テーブルの事前取り出しが使用されました。
原因テキスト:	Query 最適化プログラムは、テーブル走査に必要とされる処理時間の削減のために、並列プリフェッチ・アクセス方式の使用を選択しました。
リカバリー・テキスト:	並列プリフェッチは照会のパフォーマンスを向上させることができます。アクセス・プランが並列プリフェッチを使用するために作成されたとしても、次のことが真である場合にのみ、システムは照会を実際に実行します。 -- 照会属性 DEGREE が、アプリケーション・プロセスに *IO または *ANY のオプションで指定されました。 -- 複数の入出力ストリームによって取り出されるデータをキャッシュに入れるのに使用できる、十分な主記憶域があります。通常、5 メガバイトが最小です。共用プールのサイズを増やすと、パフォーマンスが向上する場合があります。 並列テーブル・プリフェッチについて詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4024 - 並行索引プリロード・アクセス方式が使用されました

メッセージ・テキスト:	並行索引プリロード・アクセス方式が使用されました。
原因テキスト:	Query 最適化プログラムは、この照会に必要とされる処理時間の削減のために、並列索引プリロード・アクセス方式の使用を選択しました。これは、照会がオープンされるときに、この照会によって使用される索引がアクティブ・メモリーにロードされることを意味しています。
リカバリー・テキスト:	並列索引プリロードは照会のパフォーマンスを向上させることができます。アクセス・プランが並列プリロードを使用するために作成されたとしても、次のことが真である場合にのみ、システムは実際に並列プリロードを使用します。 -- 照会属性 DEGREE が、アプリケーション・プロセスに *IO または *ANY のオプションで指定されました。 -- この照会によって使用されるすべての索引オブジェクトを、アクティブ・メモリーにロードするのに十分な主記憶域があります。通常、5 メガバイトが最小です。共用プールのサイズを増やすと、パフォーマンスが向上する場合があります。 並列テーブル・プリフェッチについて詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4025 - 並行テーブル・プリロード・アクセス方式が使用されました	
メッセージ・テキスト:	並行テーブル・プリロード・アクセス方式が使用されました。
原因テキスト:	Query 最適化プログラムは、この照会に必要とされる処理時間の削減のために、並列テーブル・プリロード・アクセス方式の使用を選択しました。これは、照会がオープンされるときに、この照会によってアクセスされるデータがアクティブ・メモリーにロードされることを意味しています。
リカバリー・テキスト:	並列テーブル・プリロードは照会のパフォーマンスを向上させることができます。アクセス・プランが並列プリロードを使用するために作成されたとしても、次のことが真である場合にのみ、システムは実際に並列プリロードを使用します。 -- 照会属性 DEGREE が、アプリケーション・プロセスに *IO または *ANY のオプションで指定されたこと。 -- ファイルのすべてのデータをアクティブ・メモリーにロードするのに使用できる、十分な主記憶域があること。通常、5 メガバイトが最小です。共用プールのサイズを増やすと、パフォーマンスが向上する場合があります。 並列テーブル・プリフェッチについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4026 - テーブル番号 &1 で索引専用アクセスが使用されました	
メッセージ・テキスト:	テーブル番号 &1 で索引専用アクセスが使用されました。
原因テキスト:	索引専用アクセスは主に、索引走査キー行位置や索引走査キー選択と共に使用されます。このアクセス方式は、全データをデータ・スペースへのランダム入出力の実行からではなく、むしろ索引から取り出します。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	索引専用アクセスについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4027 - アクセス・プランが、システムにインストールされている DB2 Symmetric Multiprocessing によって保管されました。	
メッセージ・テキスト:	アクセス・プランが、システムにインストールされている DB2 Symmetric Multiprocessing によって保管されました。
原因テキスト:	システム・フィーチャー DB2 Symmetric Multiprocessing がシステムにインストールされているにもかかわらず、保管されたアクセス・プランが作成されました。アクセス・プランはこのシステム・フィーチャーの存在によって影響を受ける場合があります。このシステム・フィーチャーがインストールされているために、照会の実装が変更されることがあります。
リカバリー・テキスト:	システム・フィーチャー DB2 Symmetric Multiprocessing がどのように照会に影響を及ぼすかについて詳しくは、160 ページの『照会の並列処理の制御』を参照してください。

SQL4028 - QUERY に分散テーブルが入っています	
メッセージ・テキスト:	QUERY に分散テーブルが入っています。

SQL4028 - QUERY に分散テーブルが入っています	
原因テキスト:	分散テーブルは、複数のステップで処理されるように照会で指定されました。照会が複数のステップで処理される場合、追加メッセージが各ステップのインプリメンテーションを詳述します。アクセス・プランは個々のステップのために保管されていないため、各ステップについてのアクセス・プランのインプリメンテーション情報は使用できません。照会が STRDBG CL コマンドを使用してデバッグ・モードで実行される場合、各ステップのインプリメンテーションを詳述するデバッグ・メッセージがジョブ・ログに記録されます。
リカバリー・テキスト:	分散テーブルがどのように照会のインプリメンテーションに影響を及ぼす可能性があるかについて詳しくは、「分散データベース・プログラミング」トピック・コレクションを参照してください。

SQL4029 - グループ化の処理にハッシュ・アルゴリズムが使用された	
メッセージ・テキスト:	グループ化の処理にハッシュ・アルゴリズムが使用された。
原因テキスト:	照会内で指定されたグループ化は、ハッシュ・アルゴリズムでインプリメントされました。
リカバリー・テキスト:	ハッシュ・アルゴリズムでグループ化をインプリメントすることは、索引が作成される必要がないため、一般的にパフォーマンス上の利点があります。ただし、この方式の使用を許可したくない場合は、単に ALWCPYDTA(*YES) を指定します。ハッシュ・アルゴリズムについて詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4030 - テーブル &2 の並行スキャンに &1 タスクが指定されました	
メッセージ・テキスト:	テーブル &2 の並行スキャンに &1 タスクが指定されました。
原因テキスト:	Query 最適化プログラムは、照会属性 DEGREE に基づいてこの照会のタスクの最適数を計算しました。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	並列テーブルまたは索引走査は、照会のパフォーマンスを向上させることができます。並列走査のための指定された数のタスクを使用するためにアクセス・プランが作成されたとしても、このジョブが実行されているプールの可用性またはディスク装置のテーブル・データの割り振りに基づいてシステムがこの数を変更する場合があります。並列走査について詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4031 - テーブル &2 上の並行索引の作成に &1 タスクが指定されました	
メッセージ・テキスト:	テーブル &2 上の並行索引の作成に &1 タスクが指定されました。
原因テキスト:	Query 最適化プログラムは、照会属性 DEGREE に基づいてこの照会のタスクの最適数を計算しました。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	並列索引の作成は照会のパフォーマンスを向上させることができます。並列索引の作成のための指定された数のタスクを使用するためにアクセス・プランが作成されたとしても、このジョブが実行されているプールの可用性またはディスク装置のテーブル・データの割り振りに基づいてシステムがこの数を変更する場合があります。並列索引の作成について詳しくは、7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4032 - テーブル &2 のビットマップ処理に索引 &1 が使用されました	
メッセージ・テキスト:	テーブル &2 のビットマップ処理に索引 &1 が使用されました。
原因テキスト:	索引は、ビットマップを作成するために照会選択と共に使用されました。次にテーブルから行にアクセスするためにビットマップが使用されました。このメッセージは、1 つのテーブルに複数表示される場合があります。これが生じる場合、ビットマップは各メッセージの各索引から作成されました。ビットマップは、ブール論理を使用して 1 つのビットマップに結合され、結果のビットマップはテーブルから行にアクセスするために使用されました。テーブル番号は、照会のこのテーブルの相対位置を参照します。
リカバリー・テキスト:	照会をデバッグ・モード (STRDBG) で実行すると、さらに特定の情報を判別できます。また、ビットマップ処理について詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4033 - &2 を使用する並行ビットマップの作成に &1 タスクが指定されました	
メッセージ・テキスト:	&2 を使用する並行ビットマップの作成に &1 タスクが指定されました。
原因テキスト:	Query 最適化プログラムは、照会属性 DEGREE に基づいてビットマップの作成に使用するタスクの最適数を計算しました。
リカバリー・テキスト:	並列索引走査を使用してビットマップを作成すると、照会のパフォーマンスを向上させることができます。指定された数のタスクを使用するためにアクセス・プランが作成されたとしても、システムはこのジョブが実行されているプールの可用性ディスク装置のまたはファイル・データの割り振りに基づいてこの数を変更する場合があります。並列走査について詳しくは、 7 ページの『DB2 for i5/OS でのデータ・アクセス: データ・アクセス・パスおよびアクセス方式』を参照してください。

SQL4034 - 結合の処理に複数の結合クラスが使用されました	
メッセージ・テキスト:	結合の処理に複数の結合クラスが使用されました。
原因テキスト:	操作が競合している結合照会が書き込まれるか、結合照会を単一照会としてインプリメントできない場合、複数の結合クラスが使用されます。各結合クラスは、一時テーブルに書き込まれた結果とともに、照会の個々のステップとして最適化され、処理されます。アクセス・プランは個々の結合クラス・ダイヤルのために保管されていないため、各結合クラスについてのアクセス・プランのインプリメンテーション情報は使用できません。照会が STRDBG CL コマンドを使用してデバッグ・モードで実行される場合、各結合ダイヤルのインプリメンテーションを詳述するデバッグ・メッセージがジョブ・ログに記録されます。
リカバリー・テキスト:	結合クラスについて詳しくは、 50 ページの『結合の最適化』を参照してください。

SQL4035 - 結合クラス &2 でテーブル &1 が使用されました	
メッセージ・テキスト:	結合クラス &2 でテーブル &1 が使用されました。

SQL4035 - 結合クラス &2 でテーブル &1 が使用されました	
原因テキスト:	このメッセージは、各結合クラスで使用されるテーブルの数をリストします。テーブル番号は、照会のこのテーブルの相対位置を参照します。同じ結合クラスにリストされているすべてのテーブルが、照会の同じステップ中に処理されます。すべての結合クラスからの結果は、照会の最終結果を戻すために共に結合されます。アクセス・プランは個々のクラスのために保管されていないため、各結合クラスについてのアクセス・プランのインプリメンテーション情報は使用できません。照会が STRDBG CL コマンドを使用してデバッグ・モードで実行される場合、各結合クラスのインプリメンテーションを詳述するデバッグ・メッセージがジョブ・ログに記録されます。
リカバリー・テキスト:	結合クラスについて詳しくは、 50 ページの『結合の最適化』を参照してください。

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、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。

IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に対する責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

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。

〒106－8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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 Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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、IBM 機械コードのご使用条件、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。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

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的創作物にも、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.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書「パフォーマンスおよび Query 最適化」には、プログラムを作成するユーザーが IBM i5/OS のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

以下は、IBM Corporation の商標です。

DB2
DRDA
i5/OS
IBM
iSeries
Language Environment
Net.Data
System i
WebSphere

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

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。

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です。

UNIX は、The Open Group の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等はそれぞれ各社の商標です。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、 IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、第三者の権利の不侵害の保証、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なしで提供されます。

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