セッション P11

IBM eServer J iSeries J

iHOPE
iSeries Hands-On Professional Education

SQL Indexing

特記事項

当資料で解説される項目の更に詳細な説明は、製品から提供されるマニュアル、オンライン・ヘルプ、Web上の情報を参照してください。

当資料は、2003年4月現在のIBMその他の製品情報に基づいて作成されております。この資料に含まれる情報は可能な限り正確を期しておりますが、日本アイ・ビー・エム株式会社による正式なレビューは受けておらず、当資料に記載された内容に関して日本アイ・ビー・エム株式会社および日本アイ・ビー・エム　システムズ・エンジニアリング株式会社が何ら保証をするものではありません。したがって、この情報の利用またはこれらの技法の実施はひとえに使用者の責任においてなされるものであり、当資料の内容によって受けたいかなる被害に関しても一切の保証をするものではありませんのでご了承ください。

日本アイ・ビー・エム　システムズ・エンジニアリング株式会社
システム・センター　サーバー・システム部

商標

以下の用語は、アメリカ合衆国、あるいは他国、あるいは両国でのIBM Corporationの商標です。

- AS/400
- AS/400e
- DB2
- IBM
- MQSeries
- Operating System/400
- OS/400
- SanFrancisco
- stylized
- WebSphere
- 400
- iSeries
- eServer

以下の用語は、アメリカ合衆国、あるいは他国、あるいは両国でのLotus Development社の商標です:

- Domino
- Domino.Doc
- LearningSpace
- Lotus
- QuickPlace
- Sametime

JavaとすべてのJavaをベースとする商標およびロゴは、アメリカ合衆国、他国、あるいは両国のサン・マイクロシステムズ社の商標または登録商標です。

Microsoft Windows, Windows NT, およびWindowsのロゴは、アメリカ合衆国、他国、あるいは両国のマイクロソフト社の商標です。

他の会社、製品、およびサービス名は、その会社の商標あるいはサービスマークかもしれません。

このプレゼンテーションに含まれるサードパーティーに関連する題材は、これらのサードパーティーから得られた情報に基づいています。これらの情報の正確さの確認のための、いかなる努力もなされていません。このプレゼンテーションは、いかなるサードパーティー製品またはサービスの、IBMによる推薦あるいは指示を表したり、ほのめかすものではありません。

アジェンダ

1. 索引とオプティマイザーの基礎
   - 索引の必要性
   - 索引の種類
   - 索引とオプティマイザーの動作
2. パフォーマンス・チューニングのための索引の使用
   - 一般的な手順
   - 索引の事前設計
   - 索引付けによるチューニング
   - Visual Explain使用手順の概要
3. 索引に関するその他の考慮点
4. OS/400 V5R2における機能拡張

索引とオプティマイザーの基礎

SQLにおける索引の必要性

- オプティマイザーは索引オブジェクトの統計情報を使用しアクセス・プランを作成
  - 索引オブジェクトに含まれる統計情報
    - 索引項目の数
    - 固有部分キー値の数

DSPFDコマンドによる索引のファイル記述の表示

```
アクセス・パス活動統計 . . . . . :
  アクセス・パス論理読み取り . . . . . :
  アクセス・パス物理読み取り . . . . . :
  索引サイズ . . . . . :              20193280
  アクセス・パス有効 . . . . . :      YES
  暗黙のアクセス・パスの共用 . . . . . :  NO
  アクセス・パスのジャーナル処理 . . . . :  NO
  固有部分キー値の数 . . . . . :
    キー・フィールド 1 . . . . . :          12781
    キー・フィールド 1 - 2 . . . . . :      580154
基礎になっているファイル . . . . . :      TANTO
  ライブラリー . . . . . :          SQLXX
  メンバー . . . . . :              TANTO
  論理ファイル様式 . . . . . :      TANTO
  索引項目の数 . . . . . :          670897
```

- 最適化のための統計情報として索引が必要
  - 索引の用途はSQL操作(選択、結合、グルーピング、順序付け)だけではない

IBM eServer iSeries

Notes:

OS/400はオブジェクト指向のOSであり、テーブルや索引もオブジェクトです。すべてのオブジェクトと同様、オブジェクトの構造、サイズ、および属性はテーブルおよび索引のオブジェクトに含まれています。また、オブジェクトの中にはカラム中の正確な値の数に関する統計的な情報や、これらの値のテーブル内の配置に関する情報が含まれています。DB2 UDB for iSeriesのオプティマイザーはこの情報を利用してデータの照会要求に対する最適なアクセス方法を決定します。

他のプラットフォームのようにシステム・カタログを使用するのではなく、オブジェクトそのものにこのような統計情報が含まれているところが、DB2 UDB for iSeriesの特徴です。

このため、管理者はオプティマイザーのために統計を手動でコンパイルするという必要もなく、オプティマイザーがある照会のアクセス・プランを作成する場合には、オブジェクトそのものにアクセスし、オブジェクト・サイズ、そのオブジェクトがすでに持つ同様の索引、照会するカラムなどの情報を獲得します。

他のプラットフォーム同様、システムはアクセス・プランを保管し、必要に応じて再使用することができます。また、環境が変わったときに自動的にアクセス・プランを更新するという機能も持っています。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

OS/400がサポートする索引

- 2進化基数ツリー索引(Binary Radix Tree Indexes)
  - DB2/400における従来からの索引
  - 少数行の選択に適している
  - CREATE INDEXでの省略時の索引
- EVI (コード化ベクトル索引- Encoded Vector Indexes)
  - ビットマップ索引の効率を改善した索引
  - 大量の情報検索に適している
- ビットマップ索引
  - オプティマイザーが判断して使用
  - 永久オブジェクトとしてユーザーが作成することはできない

IBM eServer iSeries

Notes:

iSeriesは上記の3種類の索引をサポートします。
このうちビットマップ索引についてはオプティマイザーがその使用を決定し、照会時に動的に作成され、テーブル走査やその他のアクセス・メソッドと併用されます。この索引をユーザーが作成し、永久オブジェクトとして存在させておくことはできません。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

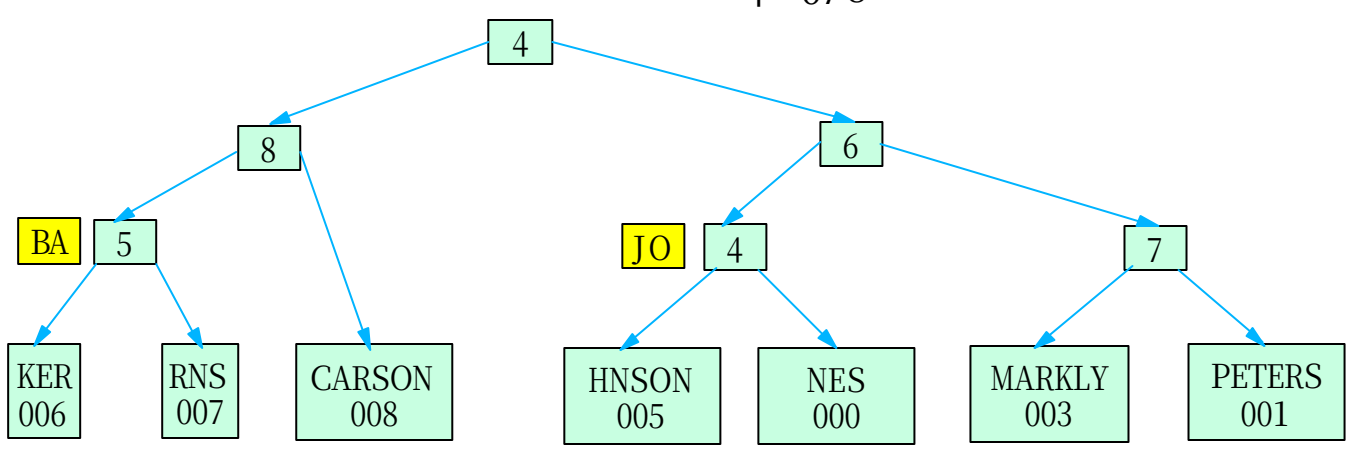
2進化基数ツリー索引(Binary Radix Tree Indexes)

索引

キー	レコード番号
BAKER	006
BARNS	007
CARSON	008
JOHNSON	005
JONES	000
MARKLY	003
PETERS	001

キー値のコード表現

EBCDIC	2進数
C2C1D2...	1100 0010 1100 0001 1101 0010 ....
C2C1D9...	1100 0010 1100 0001 1101 1001 ....
C3...	1100 0011 ...
D1D6C8...	1101 0001 1101 0110 1100 1000 ...
D1D6D5...	1101 0001 1101 0110 1101 0101 ...
D4...	1101 0100 ...
D7...	1101 0111 ...



Notes:

2進化基数ツリー索引は、他のDBと同様にiSeriesが採用している基本的な索引の構造です。
この索引はバイナリー・サーチを使用することで素早い検索を実現します。
バイナリー・サーチによる索引は、0と1での分岐を伴うツリー構造になります。ツリーにはテスト・ノードと端末ノードの2種類があり、ツリーの各テスト・ノードは数字の1ビットをテストします。ビットが0か1かによって、下位レベルの2つのノードのうちのどちらを次のノードとして選択するかが決まります。ツリーの先端から始めて、最初のノードがその数の最初のビット（左端のビット）をテストします。ツリーの2層目にはテスト・ノードが2個あり、最初のビットが0か1かによって、そのどちらかが選択されます。
上の例で、たとえばCARSONという値を検索するとします。この値をバイナリー・データによって1ビットずつテストしていきます。そうすると、左から4ビット目で分岐が発生します。ここではCARSONは0の方を選択します。すると次は左から8ビット目でまた分岐が発生します。ここではCARSONは1の方を選択することになります。するとそれは端末ノードであり、CARSONの値にたどりついたことになります。そして属性として保持している相対レコード番号を入手することができます。

EVI (コード化ベクトル索引)

EVIは 2つの部分から構成される：

記号テーブル

Key Value	Code	First Row	Last Row	Count
Arizona	1	1	80005	5000
Arkansas	2	5	99760	7300
.....				
Virginia	37	1222	30111	340
Wyoming	38	7	83000	2760

ベクトル	レコード番号
Code 1	1
Code 17	2
Code 18	3
Code 9	4
Code 2	5
Code 7	6
Code 38	7
Code 38	8
Code 1	9
…	...

- 記号テーブルは各キー値に対する情報を持つ。各キー値には固有の16進コードが割り振られる
  - コードは1，2、または4バイトで、これはキー値の総数に依存する
- 各キー値に対するビット配列以外に、索引は1つのコードの配列を持つ(ベクトル)

IBM eServer iSeries

Notes:

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個別キー値についての統計および記述情報が含まれます。それぞれの個別キーには、 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。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

EVIの動作

- 最適化プログラムは記号テーブルを使用してコストを計算
- EVIの使用が選択されるとベクトルを使用して動的ビットマップが作成される
- 照会条件に一致する行のビットはオンになる
- ANDおよびORでビットマップを結合し選択対象を絞り込む
- 照会完了時に動的ビットマップは除去される

Notes:

最適化プログラムは、照会についてのコスト情報を収集する記号テーブルを使用します。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 ビットを含む動的ビットマップを作成します。

行が、照会選択の条件を満たす場合には、ビットはオンにセットされます。行が、照会選択の条件を満たさない場合には、ビットはオフにセットされます。

ビットマップ索引と同様に、 中間動的ビットマップは 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だけ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すべき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。

EVIの使用要件

- 最適化プログラムに正確な統計情報を提供したい場合
- 完全テーブル走査を選択し、選択行がテーブルの20%-70%の場合
- 読み取り専用または更新、追加、削除が最小限のテーブルに対しての使用

Notes:

EVIは主に行の選択に使用され、与えられた述部の値の選択可能性に関して正確な統計情報をオプティマイザーに提供します。

EVIはグループ化と順序付けに対しては使用できず、結合に対しても限られた使用しかできません。結合、グループ化、順序付けが含まれる照会を実行する場合には、他の索引との組み合わせで使用されます。選択される行が比較的少ない場合には、binary radix indexのアクセスの方がパフォーマンスがよく、選択行がテーブルのおよそ20%から70%の場合には、EVIまたはbinary radix indexから作成されるビットマップを使用したスキップ順次アクセスが最善の選択肢となり得ます。また、最適化プログラムとデータベース・エンジンはデータの選択に1つ以上の索引を使用することもできます。この技法はローカル選択がANDまたはOR条件を持っていて、1つの索引が適切なキー列を含んでいない場合や 1 つの索引がすべての条件を満たさない場合に使用されます。EVIから作成されたビットマップが選択プロセスをせばめるために組み合わせて使用されます。

コード化ベクトル索引を考慮すべきなのは、 統計を収集したい場合、完全テーブル走査を選択し、照会の選択が 20%-70% であり、動的ビットマップを使用するスキップ順次アクセスにより走査の速度を上げることができる場合、あるいは、星形結合照会に星形スキーマ結合を使用することが予期される場合などです。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の制限と考慮事項

- オーバー・フロー域の保守に負荷がかかるためOLTPでの使用は推奨されない
- ORDER BY，GROUP BYでは使用できない
- 索引専用アクセスには使用できない

Notes:

新しいデータが索引付きのテーブルに挿入されるとき、DB2 UDB for iSeriesは自動的に索引を保守します。DB2 UDB for iSeriesはEVI記号を走査し、一致する値を探し出し、記号テーブルの統計情報を更新します。既存のEVIに新しい固有値が導入されるときには、以下のうちの1つが発生します。

- 新しい値がシンボル・テーブルの中の最後の固有値より論理的に後ろに位置づけられる場合、値はテーブルの一番後ろに追加される。
- 固有値が索引の順序内になる場合、値はシンボル・テーブルのオーバーフロー域に置かれ、索引が再作成またはリフレッシュされるまでそこにとどまる。たとえば、「月」のカラムに1、2、4が含まれており、そこに3が挿入されようとしたとき、この値および関連する統計情報は索引が再作成されるか、EVIがリフレッシュされるまでオーバーフロー・エリアに置かれる。

順序外に置かれる固有キー値が増加するとEVIの効率が悪くなるため、シンボル・テーブルのオーバーフロー域に置くことのできる固有キー値の数には制限があります。この上限に達すると、EVIは自動的にリフレッシュされます。?

リフレッシュの間EVIは遅延保守モードになり、オプティマイザーとデータベース・エンジンからは使用できなくなります。このためパフォーマンスが著しく悪化します。

このため、EVIは多くの更新、追加、削除が発生するOLTPの環境での使用は推奨されません。また、いくつかの新しい固有値が挿入されるときには、EVIを削除し、挿入後に再作成することが推奨されます。

オーバーフロー値をシンボル・テーブルに戻すためにEVIをリフレッシュする方法には2通りあります。

- 索引をすべて削除し、標準のSQLステートメントを使用するかオペレーションズ・ナビゲーターを使用してこれらを再作成する。
- CHGLF(論理ファイルの変更)コマンドを使用し、FRCRBDAP(アクセス・プランの再作成)パラメーターを*YESにする。これにより、索引の削除と再作成と同じ結果が得られる。

また、EVIはグループ化と順序付けに対しては使用できません。

索引内にすべての選択列が含まれる索引専用アクセスにも使用することができません。

ビットマップ処理方式

テーブル： Courses

Location	Topic
NY	DB2/400
LA	JAVA
NY	JAVA
CHI	NOTES

存在する索引：
LoclX -> 列 Location
ToplX -> 列 Topic

SELECT coursenumber FROM courses
WHERE location='NY' OR topic='JAVA'

DB2 UDB for iSeriesは結果のビットマップを使用してCoursesテーブルから最初の3行だけを読み取る

Notes:

この方式では、行をデータ・スペースから選択するために使用する１つまたは複数の索引を最適化プログラムが選択します。
各索引にはテーブル内の行ごとに1ビットを持つ一時ビットマップが作成されます。このビットマップの内、選択条件に一致した行に対するビットのみがオン(1)になります。
上の例では、索引LoclXのビットマップに対しては、‘NY’と一致する値をもつ1行目と3行目のビットだけがオンになり、ビットマップは1010となります。索引ToplXのビットマップに対しては、‘JAVA’と一致する2行目と3行目のビットがオンになり、ビットマップは0110となります。
これら複数の索引に対するビットマップから、ANDまたはORの論理演算により1つのビットマップが生成されます。
上の例では1110というビットマップが生成されます。
このビットマップを使用することで、データベースから読み取る行を最小にすることができます。
上の例ではアクセス対象は最初の3行のみに絞りこまれます。

ビットマップ方式が選択される条件

- 照会の処理時に最適化プログラムによってプリプロセスされる
- 他のアクセス方式との組み合わせで使用され、データ・アクセスに直接使用されるものではない

Notes:

この処理方式は最適化プログラムが照会の処理を開始する前に必ずプリプロセスされます。ビットマップ処理方式は、テーブル走査、索引走査キー位置決め、索引走査キー選択などの基本アクセス方式と組み合わせて使用されます。

たとえば、テーブル走査との組み合わせで使用される場合、実際にアクセスされるテーブル・スペースはビットマップでオンになっている行だけが対象となるため、データ・アクセスを削減することができます。

ビットマップ方式の考慮事項

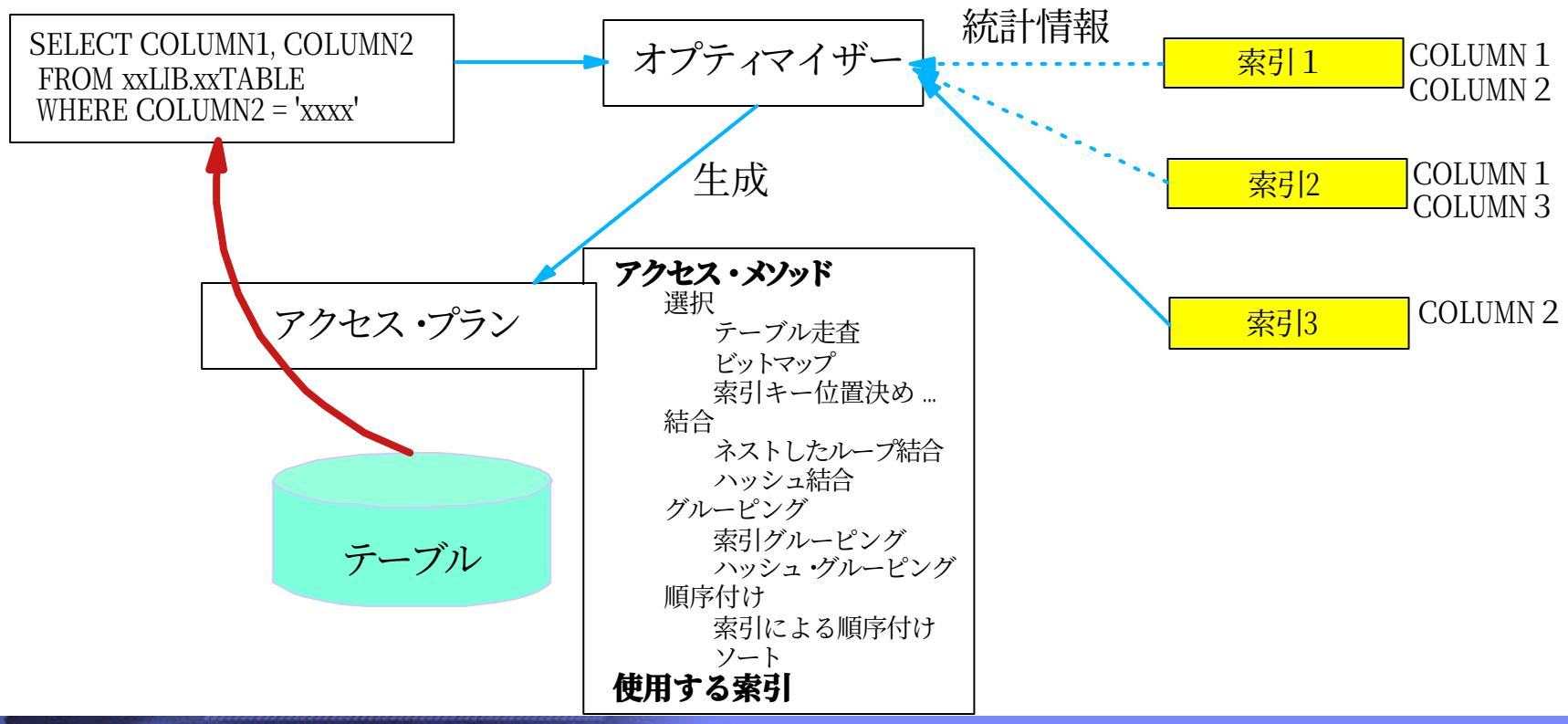
- 照会の最初の入出力要求でプリプロセスされるため最初の行のフェッチには時間がかかる
- 新規の行および更新された行は反映されない
- 挿入、更新、削除を含む照会にはビットマップは使用されない

Notes:

- ビットマップは照会の最初の入出力要求でプリプロセスされます。そのため、最初の行のフェッチには、後の行の取り出しよりも時間がかかります。
- ビットマップは一度生成されるとそれ以降は新規の行または変更された行を選択しません。
- 最適化プログラムは挿入、更新、または削除が可能な照会に対しては、ビットマップ処理を使用しません。

オプティマイザーによる索引の使用の概念

- 行と列に関する情報からコストを見積もり最適なアクセス・プランを作成



Notes:

iSeriesのオプティマイザーはコスト・ベースの最適化を行うため、行とカラムに関するより多くの情報を得ることで、オプティマイザーはその照会に対する最適な（コストの最も低い）アクセス・プランを作成することができます。索引からの情報で、オプティマイザーはその要求を処理するより良い方法を選択することができます。（ローカル選択、結合、グループ化、順序付け）

索引の使用がI/Oの低減に寄与するかどうかを判断するには、照会の結果としての行数を見積もることが重要です。たとえば、テーブル中の90%の行が照会の結果として含まれる場合は、行にアクセスする最良の方法は全テーブルのスキャンです。しかし、1%が照会結果となる場合には全テーブルのスキャンは非常に非効率であり、多くの資源を使用し、極端に遅いはずです。この場合には索引によるキー・アクセスのメソッドが最も効率的ということになります。

iSeriesサーバーにおいては、オプティマイザーは照会要求、テーブル属性、索引内の情報を参照することで照会結果の行数を見積もります。各テーブルのヘッダー情報にはテーブル内の行数が含まれていますが、オプティマイザーは索引中にある個々の値の数や、特別な値を持つ行の数を獲得するためにこのヘッダー情報を使用することはありません。

その代わり、このような情報は索引から獲得されます。Radix IndexesとEncoded Vector Indexesのどちらにも、カラム内にある個々の値の数と値の配置に関する情報が含まれています。DB2 UDB for iSeriesは、最適化の中でデータの偏りを認識できる数少ないデータベースの1つということになります。

オプティマイザーの照会行数の見積もり

- 索引中の固有値数
- 索引項目の数
- 複合キーの場合は左 4つまでのキー列で固有値数を判断

オプティマイザーの情報源として索引が必要

DSPFDコマンドによる索引のファイル記述の表示
アクセス・パス活動統計 . . . . . :
....
固有部分キー値の数 . . . . . :
キー・フィールド 1 . . . . . : **12781**
キー・フィールド 1 - 2 . . . . . : **580154**
キー・フィールド 1 - 3 . . . . . **1065842**
キー・フィールド 1−4. . . . . **6708975**
基礎になっているファイル . . . . . : CUSTOMER
.....
索引項目の数 . . . . . : **6708975**

Notes:

Radix indexesでは、オプティマイザーは索引の左端4つの連続するキーによってコスト情報を獲得します。それぞれのカラムにいくつの固有値が含まれているかを知ることに加えて、索引はカラムをまたがった基数情報を提供します。つまり、オプティマイザーは索引の左端４つのカラムを見て、テーブルにある４つのカラムにいくつの固有な順列があるかを判断することができます。この統計情報を得るために、オプティマイザーはデータベース・エンジンに対して、選択範囲と一致する行（キー）数の「キー見積もり」の実行を要求します。この見積もりプロセスは照会最適化の一部で、キーのサブセットをカウントするために索引が使用されます。

キー・カラムまたは複合キー内の固有値の数は、OS/400のコマンドであるDSPFD(ファイル記述の表示)またはオペレーションズ・ナビゲーターを使用して参照することができます。

ある場合には、オプティマイザーは最適化のために索引を使用しても、索引を使用しない照会の実行を選択することがあります。たとえば、照会によって多くの行数を獲得しようとする場合、全テーブルのスキャンをした方が高速な場合があります。iSeriesサーバーでは、独立したI/Oサブシステムや並行I/Oテクノロジー、そして巨大なメモリー・システムのために全テーブル・スキャンは比較的効率がよく、要求された行をアクセスするメソッドとして、全テーブル・スキャンが最低コストとなる場合があります。

しかし、照会要求に対して全テーブル・スキャンが最良のアクセス・メソッドであっても、オプティマイザーは索引から得られる情報をもとにその判断を行います。データに関する統計を明示的に生成するユーティリティーは存在しません。そのため、索引がデータの読み取りに使用されるかどうかとは無関係に、各テーブルにいくつかの索引を定義しておくことが重要になります。

ネストされたループ結合

- オプティマイザーによるアクセス・プランの作成
  - 個々のテーブルから獲得される行数を見積もる
  - 各テーブルに対する最適なアクセス・メソッドを選択
  - 結合順序、アクセス順序の選択肢を評価
  - 最適な結合順序、アクセス順序を選択
- 結合列に索引を作成することが重要
  - オプティマイザーへの見積もり情報提供
  - 一時索引作成による負荷回避
- 結合先の結合列とレコード選択列の双方に索引をつける
  - 結合と選択を索引のアクセスだけで行なう

IBM eServer iSeries

Notes:

DB2 UDB for iSeriesが複数のテーブルの行アクセスに内部結合を使用するアクセス・プランを作成する場合、個々のテーブルからいくつの行が獲得されるかの見積もり情報を収集します。そして使用可能な種々のメソッドのコストに基づいて、各テーブルに対する最適なアクセス・メソッドを選択します。この見積もりとコストに基づいて、オプティマイザーは可能性のある種々の結合順序、あるいは照会対象のテーブルがアクセスされる順序を検討し、テーブルを最も効率的な順序で結合するアクセス・プランが作成されます。照会が記述された順序でのみでしか結合を処理できないデータベースとは異なり、DB2 UDB for iSeriesは可能性のあるすべての選択肢を評価し、最適な（最低コストの）結合順序が使用されるように照会を書き直します。ただし、左外部結合や例外結合のような結合タイプでは、結合はSQL要求で指定された順序で結合が行われる必要があります。

結合順序を評価するために、オプティマイザーは索引からの情報を必要とします。結合順序は、どのテーブルが最も多くの行を読み取る結合の「ファン・アウト」であるかに依存します。結合のファン・アウトは単純に、与えられた結合値と一致するべき行の数として定義することができます。オプティマイザーは索引を参照して読み取られる行の数を見積もりまるため、結合カラムに索引を作成することが非常に重要になります。

結合処理の第一のメソッドは、ネストされたループ結合と呼ばれる。これは、少なくとも２つのテーブルを結合する照会に適用されます。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

ネストされたループ結合による選択

TABLE1

COL1	KEY1	KEY2
1	A	IK
2	B	GH
3	C	NJ
4	D	PO
5	E	GH
6	F	IK
7	G	PO
8	H	GH
9	I	DF
10	J	ER

選択

TABLE2

BKEY	COL2	COL3
B	2000	30
C	2500	50
E	1800	70
F	3000	70
G	6800	20
I	2700	10
J	3500	10

TABLE3

CKEY	COL4
10	GGG
20	EEE
30	SSS
40	QQQ
50	CCC
60	SSS
70	UUU
80	WWW
90	VVV

TABLE4

DKEY	COL6
DF	1
ER	3
GH	2
IK	4
NJ	7
PO	9

```
SELECT A.COL1, B.COL2, C.COL4, D.COL6
  FROM TABLE1 A, TABLE2 B, TABLE3 C
  TABLE4 D
  WHERE A.KEY1 = B.BKEY AND
  B.COL3 = C.CKEY AND
  A.KEY2 = D.DKEY
  WHERE A.COL1 IN (2, 3)
```

COL1	COL2	COL4	COL6
2	2000	SSS	2
3	2500	CCC	7

IBM eServer iSeries

Notes:

ネストされたループ結合では、行がいずれかのアクセス・メソッド(例えば テーブル・スキャン、キー行ポジショニング)を使用して1番目のテーブルまたはダイヤルから読み取られます。そして結合キー値が得られて2番目のテーブルまたはダイヤルの索引が検索され、キー値が見つかると、行がテーブルから読み取られます。次の一致するキー値が索引から読み取られ、関連する行がテーブルから読み取られます。このプロセスは索引から一致するキー値がなくなるまで続きます。データベース・エンジンは1番目のテーブルから次の行を読み取り、次のキー値に対する結合プロセスが開始されます。ネストされたループ結合は、1番目のテーブルでの選択で一致する全ての行が処理されるまで行われます。処理を可能な限り効率化するために、ネストされたループ結合の処理を理解しておく必要があります。

ネストされたループ結合では、2番目のテーブルで一致する行が多数存在する場合、多くのI/Oが発生することがあります。

ネストされたループ結合では、結合されるテーブルにradix indexが必要であり、結合に使用されるカラムに索引が存在しない場合には、このカラムに対して一時索引が作成されることになります。

この場合、照会処理のための時間が長くなり、システム資源をより多く使うことになります。

V4R5から、すべてのネストされたループ結合はローカル選択のキー行ポジションのように処理されます。結合とローカル選択の述部の両方とも結合先のダイアルに存在する場合、オプティマイザーはradix indexを検索するためにすべてのカラムを使用することができます。これにより、最小限のI/Oで照会と結合基準にマッチする行を絞り込むよう効率化を図ることが可能です。この技法は「複数キーによる行位置づけ結合」と呼ばれ、この場合ローカル選択の対象カラムと結合のためのカラムの双方がradix indexに存在していることが非常に重要です。もし結合カラムしか索引になかった場合、データベース・エンジンは結合のためにindexを検索し、ローカル選択のためにはテーブルを読み取って値のテストをしなければならなくなります。ローカル選択でデータが一致しなかった時には、索引の検索によるランダム・アクセスは無駄になってしまいます。

上の図のようなSELECT文を実行する場合、結合キーだけでなく選択条件となるカラムにも索引が存在することでテーブルのスキャンを回避し、パフォーマンスを向上させることができます。

この場合、双方のテーブルのKEYカラムに索引をつけるだけでなく、TABLE1のCOL1にも索引を付けることが効果的です。

ネストされたループ結合は、結合時に並行処理は行いません。SMPは一時索引を作成する場合には使用されることがあります。

ハッシュ結合では、2番目のテーブルにハッシュ・テーブルを作成する場合には並行処理が有効となります。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

GROUP BY とORDER BY

- GROUP BY
  - ハッシュ・グルーピングまたは索引グルーピングの 2種類
    - グループが多くの行を含む場合はハッシュ・グルーピングを選択
    - 少ない行をグルーピングする場合は索引グルーピングを選択
  - オプティマイザーは索引内の情報に基づいて固有なグループの数を見積もる
  - 索引がない場合省略時のグループ数と行数を使用
    - 非効率なアクセス・プランを選択する可能性がある
- ORDER BY
  - 索引またはソートのどちらかをオプティマイザーが選択
  - ORDER BYの列が複数テーブルにある場合はソートを使用

IBM eServer iSeries

Notes:

グルーピングと順序付けでは、オプティマイザーは選択と結合で処理される行数に基づいて使用可能な様々なメソッドのコストを見積もります。
オプティマイザーは索引内の情報に基づいて固有なグループの数を見積もります。索引がない場合、オプティマイザーはグループごとに省略時のグループ数と行数を使用します。この見積もりが正確でない場合、オプティマイザーは非効率なアクセス・プランを選択してしまうことになります。

一般的に、グループごとに多くの行をグルーピングする場合にはハッシュ・グルーピングが使用されます。ハッシュ・グルーピングでは選択した行を一時結果ファイルとして蓄え、それをグループ化しながら結果を生成します。
グループ中の行が少ない場合には、索引を経由して行をグループ化する、索引グルーピングが使用されます。

索引グルーピングには、行の処理にSMPまたは並行処理は使用されません。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

GROUP BYおよびORDER BY列のための索引

- グルーピングまたは順序付けの列を持つテーブルは多くの場合結合順の先頭になる
  - 列が単一のテーブル上にある場合
- パフォーマンスに問題がある場合
  - 別の索引を作成しその情報により最適化の修正を促す

TABLE1

COL1	COL2	COL3	KEY1
2	AA	XYZ	10
4	AB	ABC	50
5	BB	DEF	20
1	BC	XYZ	10
3	RR	DEF	30
2	AA	XYZ	20
7	EE	STU	10
1	BC	XYZ	30
2	KK	ABC	20
5	JS	XYZ	80

TABLE2

KEY2	COL4
10	1000
20	500
30	2000
40	300
50	500
60	4000
70	200
80	700
90	8000
100	900

```
SELECT A.COL1, A.COL2, SUM(B.COL4)
  FROM TABLE1 A, TABLE2 B
  WHERE A.KEY1 = B.KEY2
  AND  A.COL3 = 'XYZ'
  GROUP BY A.COL1, A.COL2
```

COL1	COL2	COL4
2	AA	1500
1	BC	3000
5	JS	700

TABLE1の索引1 | COL3 | KEY1 |

TABLE1の索引2 | COL3 | COL1 | COL2 |

TABLE2の索引 | KEY2 |

Notes:

グルーピングまたは順序付けに索引を使用することは、照会の結合順に影響します。グルーピングまたは順序付け（あるいは双方とも）のカラムが１つのテーブルにある場合にこの事象が発生します。データベース・エンジンがキー結合における1番目のテーブルから行を読み取ることができ、グループ化および（または）順序付けが自然に行われるように、このテーブルが結合順の先頭にくる傾向があるためです。

照会全体の適切なパフォーマンスにとって、これは最適なプランではない場合があります。選択と結合の統計を見ながらradix indexを作成し、選択とグルーピングまたは順序付けの総計を見ながらまた別の索引を作成することが最適化を助ける手段となります。データベース・エンジンは双方の索引を使用することができませんが、オプティマイザーに対して照会、結合、およびグルーピング属性の選択についてよい考えを与えることになります。

GROUP BYにおけるMINおよびMAXの効率化

- ローカル選択列とMIN/MAX列との複合キー索引を使用
  - ローカル選択列＋MIN列で複合キー索引を作成
    - 昇順で作成
  - ローカル選択列＋MAX列で複合キー索引を作成
    - 降順で作成
- 必ず索引の最初の行がヒットし効率が上がる

```
CREATE INDEX X2 ON EMPLOYEE
  (STATE, SALES DESC)

SELECT STATE, MAX(SALES)
  FROM EMPLOYEE
  WHERE STATE IN('ARIZONA', 'CALIFORNIA')
  GROUP BY STATE
```

EMPLOYEE

STATE	SALES	CUSTOMER
Alabama	375	Jones
Alabama	150	Smith
Alabama	110	Doe
Alaska	400	Johnson
Alaska	120	Smith
Alaska	55	Alexander
Alaska	10	Lee
Arizona	540	White
Arizona	360	Doe
Arizona	210	Brown
Arizona	80	Jacobson
Arizona	50	Milligan
Arkansas	90	Weatherby
Arkansas	25	Smith
Arkansas	5	Pippen
California	75	Lee
California	30	Wayne

IBM eServer iSeries

Notes:

オプティマイザーが選択できる別のグルーピング技法に、MINおよびMAX機能の 早期退出」機能というものがあります。この方法は複数キー列位置決めの形態の１つであり、索引を使用したデータの順序付けのメリットを享受することができます。このためには、索引で、MINまたはMAX機能で使用される列の前にローカル選択キー列を含めることが必要になります。MAXではキー ･列は降順である必要があります。MINまたはMAX機能の列の前にローカル選択列をキーとして指定することで、データベース･エンジンはローカル選択とMINまたはMAXの条件に一致する最初の複合キー値を効率的に読み取ることができます。データベース･エンジンはローカル選択で一致する次の値にポジションを移します。この「早期退出」ルーチンによりデータベース・エンジンは、MINまたはMAX値を探すためにローカル選択で一致するすべての行を読み込んで処理する必要がなくなります。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

パフォーマンス・チューニングのための索引の使用

索引作成の一般的アプローチ

- 基礎的な索引を作成する対象
  - 主キーおよび外部キーとなる列
  - 行選択述部で頻繁に使用される列
  - テーブルの結合に頻繁に使用される列
  - グループ化または順序付けで頻繁に使用される列

Notes:

特定の照会に対してではなく、データベース・モデルとアプリケーションに基づいて索引を作成することが初期の望ましいアプローチと言えます。スタート・ポイントとして、以下の尺度で基礎的な索引を設計することが推奨されます。

- 主キーおよび外部キーのカラム
- ローカル選択で共通に使用されるカラム
- 結合で共通に使用されるカラム
- 共通に使用されるグループ化のカラム

索引作成の事前アプローチ

- 索引付けの目的
  - オプティマイザーに情報を提供する
    - 固有値数、値の配分、重複値の平均など
  - アクセス・プランの選択肢を与える
- 事前アプローチの基本的規則
  - ボリュームが最大または頻繁に実行する照会に対して2進化基数索引を作成
  - OLAP、頻度の低い照会にはコード化ベクトル索引を検討

事前アプローチの際に使用するテスト用SQLの例

```
SELECT B.COL1, B.COL2, A.COL1
FROM TABLE1 A, TABLE2 B
WHERE B.COL1 = '値'          ← 選択
      B.COL2 = '値'
      A.JOIN_COL = B.JOIN_COL  ← 結合
GROUP BY B.COL1, B.COL2, A.COL1
ORDER BY B.COL1
```


IBM eServer iSeries

Notes:

事前アプローチの基本的規則：
- 最大またはもっともよく使用される照会に対するradix indexを作成する
- OLAPや使用頻度の低い照会に対しては、選択カラムに対する単一キーのEVIを作成する

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

完全な2進化基数索引のガイドライン

- 複合キー索引の中での列の順序
  - 等号の述部で使用される列を先頭にする
    - 等号 -> 不等号の順
  - 等号を持つ述部の優先順位
    - 選択述部 + 結合述部 ―――― 結合の 1次テーブルの場合
    - 結合述部 + 選択述部 ―――― 結合の 2次テーブルの場合
    - 選択述部 + GROUP BY の列
    - 選択述部 + ORDER BYの列

一般的に、選択に最も使用頻度の高い列は索引の先頭におく

Notes:

完全なradix indexではカラムの順序が重要になります。これにより、オプティマイザーがデータの読み取りに使用するかどうかが変わってきます。一般的な規則としては、カラムを以下のような方法で索引の中に順序付けます。

- 等号の術部を先頭にする。つまり ”= ”を使用する述部は、選択する行の範囲をもっとも速くせばめるため、索引の先頭におくべきである。
- すべての述部が等号のオペレーターを持つ場合、カラムを以下のように並べる
- 選択述部 + 結合述部
- 結合述部 + 選択述部
- 選択述部 + Group by カラム
- 選択述部 + Order byカラム

また一般的に、このガイドラインに加えて、最もよく使用される選択カラムは索引の先頭に置くことが推奨されます。。

Binary radix indexは選択、結合、順序付け、グループ化、一時テーブル、統計に使用されます。選択のデータ・アクセス・メソッドを評価する時は、照会のWHEREステートメント中のローカル選択と結合述部に一致するキーを持つbinary radical indexを作成する。

前に述べているとおり、複合キーに対してbinary radix indexを作成する場合には、キーの順序が重要です。通常、索引中のキーの順序はローカル選択と結合述部の順序、あるいはローカル選択とグルーピング・カラムの順序にすべきです。 等号が最初で不等号がその次)。

Binary radix indexは固定的な照会、または標準化された報告書ための照会に作成されることが推奨されます。Binary radix indexはディスク資源を消費するため、索引を作成する数はシステムの環境や照会の最適化に依存します。

完全な2進化基数索引ガイドラインの例 1

- 単一テーブルの照会

```
SELECT CUSTOMER, CUSTOMER_NUMBER, ITEM_NUMBER
FROM ITEMS
WHERE YEAR = 2000
AND QUARTER = 4
AND RETURNFLAG = "R"
AND SHIPMODE = "AIR"
ORDER BY CUSTOMER_NUMBER, ITEM_NUMBER
```

索引中の順序：

```
YEAR + QUARTER + RETURNFLAG + SHIPMODE +   ← 選択述部
CUSTOMER_NUMBER + ITEM_NUMBER              ← ORDER BY
```


IBM eServer iSeries

Notes:

例1：1テーブルの照会
この照会は、テーブルITEMSを使用し、2000年末に航空便で出荷された注文を返品する顧客を検索する例です。

```
SELECT CUSTOMER, CUSTOMER_NUMBER, ITEM_NUMBER
FROM ITEMS
WHERE YEAR = 2000
AND QUARTER = 4
AND RETURNFLAG = "R"
AND SHIPMODE = "AIR"
ORDER BY CUSTOMER_NUMBER, ITEM_NUMBER
```

照会は4つの行選択述部と2つのORDER BY列を持っています。
完全な索引のガイドラインに従って、等号の述部で使用される列を先頭におく。(YEAR, QUARTER, RETURNFLAG, SHIPMODE)
続いてORDER BYで使用するCUSTOMER_NUMBERとITEM_NUMBERをおく。

複数の等号述部列の優先順位は、他の照会を含めて頻繁に使用されるものを優先にする。

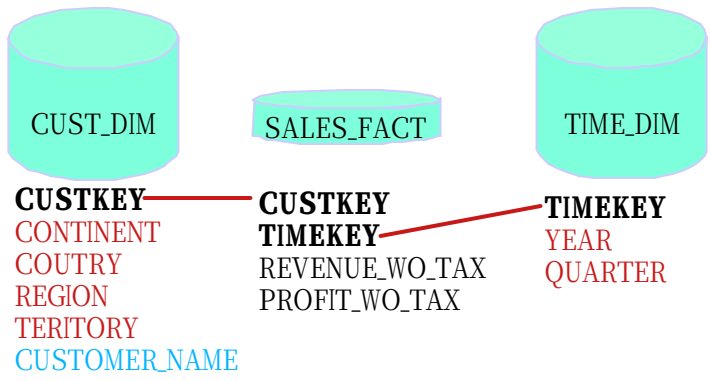
The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

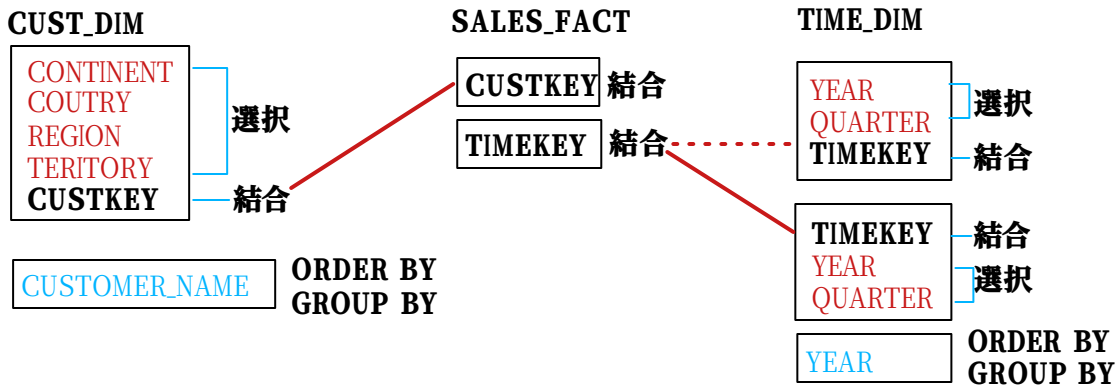
完全な2進化基数索引ガイドラインの例 2

- 3つのテーブルからの結合照会

```
SELECT T3.YEAR, T1.CUSTOMER_NAME,
SUM(T2.REVENUE_WO_TAX), SUM(T2.PROFIT_WO_TAX)
FROM CUST_DIM T1, SALES_FACT T2, TIME_DIM T3
WHERE T2.CUSTKEY=T1.CUSTKEY
AND T2.TIMEKEY = T3.TIMEKEY
AND T3.YEAR IN (2001, 2000)
AND T3.QUARTER = 1
AND T1.CONTINENT='AMERICA'
AND T1.COUNTRY='UNITED STATES'
AND T1.REGION='CENTRAL'
AND T1.TERRITORY='FIVE'
GROUP BY T3.YEAR, T1.CUSTOMER_NAME
ORDER BY T1.CUSTOMER_NAME, T3.YEAR
```



作成する索引と列の順序：



Notes:

例2：3テーブルの照会
この星型スキーマ結合の照会は、報告書の結果に含める実績テーブル中の行数を絞り込むために、範囲テーブルへの結合を行ないます。
この照会は2年間の第一四半期の売上と利益の合計を、セールス領域内の顧客ごとに検索します。

この照会には2つの結合述部と6つの選択述部があります。最初のタスクは、照会中の各テーブルに対する選択述部に着目することです。

時間軸テーブルであるTIME_DIMに対して、照会は2つのローカル選択述部を指定しています。このテーブルに対する索引はまずYEARとQUARTERを含め、その後に結合述部で指定する列TIMEKEYを含めるようにします。
顧客範囲テーブルに対しては、4つのローカル選択述部があります。これらの述部は地域的な階層によって互いに関連しており、すべての述部は等号の述部であるため、索引キーの順序はデータベース・スキーマの階層に従うべきです。
顧客範囲テーブルに対する索引は、CONTINENT、COUNTRY、REGION、TERRITORYを含み、その後に結合述部でしている列CUSTKEYを含めるようにします。

実績のテーブルSALES_FACTに対しては、WHERE句に2つの列TIMEKEYとCUSTKEYがあります。これらの列は結合述部で使用されているため、ガイドラインはTIMEKEYとCUSTKEYそれぞれに対する2つの索引を推奨します。これにより最適化プログラムはあらゆる結合順に対する統計とコストを得ることができることになります。

ガイドラインによれば、索引はGROUP BYとORDER BYで指定する列を含むべきです。
GROUP BYとORDER BYは２つの異なるテーブルの列を使用しているため、照会は２つのステップで実行されます。 選択と結合はグループ化と順序付けの前に実行されている必要がある。）
最適化プログラムとデータベース・エンジンは、グループ化と順序付けのために既存の索引を使用できないため、GROUP BYとORDER BYの列に対する索引を作成する必要があります。

完全な2進化基数索引ガイドラインの例 3

- 照会の中の不等号述部

```
SELECT T3.YEAR, T1.CUSTOMER_NAME,
SUM(T2.REVENUE_WO_TAX), SUM(T2.PROFIT_WO_TAX)
FROM CUST_DIM T1, SALES_FACT T2, TIME_DIM T3
WHERE T2.CUSTKEY=T1.CUSTKEY
AND T2.TIMEKEY = T3.TIMEKEY
AND T3.YEAR < 2001
AND T3.QUARTER = 1
AND T1.CONTINENT='AMERICA'
AND T1.COUNTRY='UNITED STATES'
AND T1.REGION='CENTRAL'
AND T1.TERRITORY='FIVE'
GROUP BY T3.YEAR, T1.CUSTOMER_NAME
ORDER BY T1.CUSTOMER_NAME, T3.YEAR
```

TIME_DIMに対する索引と列の順序：

QUARTER + TIMEKEY + YEAR

IBM eServer iSeries

Notes:

例３：照会中の不等号述部

不等号の述部は等号の述部よりも多くの結果を返す傾向があります。たとえば、ユーザーの要求が日付が開始と終了の間にある行すべてであった場合、照会は特定の日付または週に対する行よりも多くの結果を返すことになります。これは、不等号の述部は特定の値でなく値の範囲を示すためであり、最適化プログラムはアクセス・プランの作成のために異なる判断を行います。この照会は前の例を同じ報告書を要求していますが、YEARの選択述部の特定範囲は広がっています。

前の例では、TIME_DIMに対する最適な索引は、2つの選択述部の列を先に置き、結合述部の列をその後に配置するというものです。今回の例では、結合述部は 等しい」述部であるが、選択述部の１つは「より少ない」述部を使用しています。 等しい」述部がキー値への最も直接的なパスを提供するため、この照会でのTIME_DIMに対する索引はQUARTER、TIMEKEY、YEARになる。これにより、データベース・エンジンが位置付けて連続的に処理するキー値の論理範囲が定まります。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

事後的な照会のチューニング

- 必要な情報
  - 最適化プログラムが推奨する索引の情報
  - 照会に使用されている一時索引の情報
  - 最適化プログラムが選択している実行メソッド

Notes:

適切なデバッグおよびモニター・ツールを使用して以下の情報を得る必要があります。
- オプティマイザーが推奨するローカル選択のための索引
- 照会に使用される一時索引
- オプティマイザーが選択した実行メソッド

DB2 UDB for iSeriesは、永久索引を推奨するためのビルトイン・ツールであるインデックス・アドバイザーを含んでいます。ジョブログ中のインデックス・アドバイザー・メッセージは、iSeriesナビゲーターまたはOS/400のコマンドで見ることができます。または、Visual Explainを使用することも可能です。

永久テーブルに対して結合やグループ化および選択を行うために、データベース・エンジンが一時索引を作成する場合、同じカラムに永久索引を作成することで一時索引の作成を回避する必要があります。一時テーブルに一時索引が作成されることもありますが、この場合には永久索引を作成することはできません。

オプティマイザーにとって必要なことの根本は、操作対象のテーブルとカラムについてより多くの情報を提供するということです。
DB2 UDB for iSeriesの場合は、統計情報は索引を作成することでオプティマイザーに提供されるということを念頭におく必要があります。

最適化プログラムによる推奨と対応例

状況	最適化プログラムの推奨	対応
対象のテーブルに索引が存在しない。	ローカル選択に対して索引を作成	選択、結合、グループ化、順序付け対象の列に索引を作成する。
一部のローカル選択列または結合列に索引が作成されている。	ローカル選択に対して索引を作成	すべてのローカル選択列、結合列に対して索引を作成する。
すべての選択列に対して索引が作成されており、パフォーマンスは向上したがまだ要求を満たさない。	なし	索引中の列を並び替え、頻繁に選択される列、 等しい」選択列を先頭にする。結合列も考慮する。
完全な索引を作成したが最適化プログラムに選択されない。	なし	選択されているアクセス・メソッドをツールにより確認する。最適なアクセス・メソッドとして全テーブルのスキャンが選択されている可能性がある。
必要な列が含まれる索引が作成されているが、最適化プログラムによって選択されない。	同じ列を違う順序で並べて索引を作成する。	最適化プログラムが推奨する順序で列を並べた索引を作成する。
推奨された索引を作成したが、デバッグ・メッセージは照会の見積もりが実際の実行時間と乖離していることを示す。	なし	SQLステートメントにOPTIMIZE FOR ALL ROWSを追加する。
OR条件で分離された 等しくない」および/または選択述部が照会に含まれている。	最初の数列に対する2進化基数索引を作成する。	単一列の索引を各列に対して作成する。これによりAND/OR論理演算を行なうビットマップの使用を促す。

複数テーブルの結合照会のための索引

- 結合のための索引を作成
  - 結合キーに対する索引を作成する
  - 一時索引と同じ永久索引を作成する
- インデックス・アドバイザーにより推奨された索引を作成
  - 推奨された索引を作成しすべてテスト
- 以下の優先順位ですべてを結合する索引を作成
  - 等号の選択述部、結合述部、不等号の選択述部
- WHERE句に結合述部しかない場合
  - 結合列が先頭（左端）にある索引を作成

IBM eServer iSeries

Notes:

システムが永久テーブルに対して一時索引を作成している場合、ネストされたループ結合のために索引が必要である場合が多くあります。ネストされたループ結合の処理は、結合キーに対して索引を必要とします。索引はオプティマイザーによって自動的に作成されますが、ユーザーはその度に待たされることになり、この索引は照会が完了すると削除されてしまいます。同じ照会が要求されたとき、この一時索引は再度作成されることになります。
このため、この一時索引と同じ永久索引を作成する必要があります。

インデックス・アドバイザーは一時索引と異なる索引の作成を推奨する場合もあります。アドバイザーはWHERE句にあるローカル選択述部しか参照しません。たとえば、２つのテーブルを結合する照会において、1番目のテーブルは全テーブル・スキャンによってアクセスされ、2番目のテーブルは一時索引によってアクセスされるということもあります。オプティマイザーはローカル選択のための索引を推奨し、ネストされたループ結合のための一時索引の作成をメッセージとしてフィードバックします。インデックス・アドバイザーによって推奨されたカラムに索引を作成することを検討しながら、ネストされたループ結合のための一時索引と同じ索引を作成します。

このような推奨は同じ照会に対しての複数の索引を生み出す可能性があります。分析の一部として、推奨された索引をすべて作成して照会を再度実行してみる必要があります。求められた結果がまだ得られない場合、以下の優先順位ですべてのカラムを結合するradix indexの作成を検討します。

等号を持つ選択述部、結合述部、不等号を持つ選択述部

WHERE句が結合述部しか含んでいない場合は、結合されるすべてのテーブルにradix indexが存在していることを確認します。結合カラムはキーの1番目または左端のポジションに置く必要があります。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

結合順の最適化(内部結合)

- 1次テーブルと結合順の決定要因
  - ORDER BYまたはGROUP BY
    - 単一テーブル内項目に対する指定により1次テーブルとなる
  - 選択の見積もり行数
    - 選択行数の少ないものから結合
  - QAQQINIのFORCE_JOIN_ODERオプションの指定
    - *YES　　SQL文に指定された順に結合順が固定
    - *NO　　オプティマイザーが決定
    - *PRIMARY　1次テーブルのみSQL文のとおりに固定

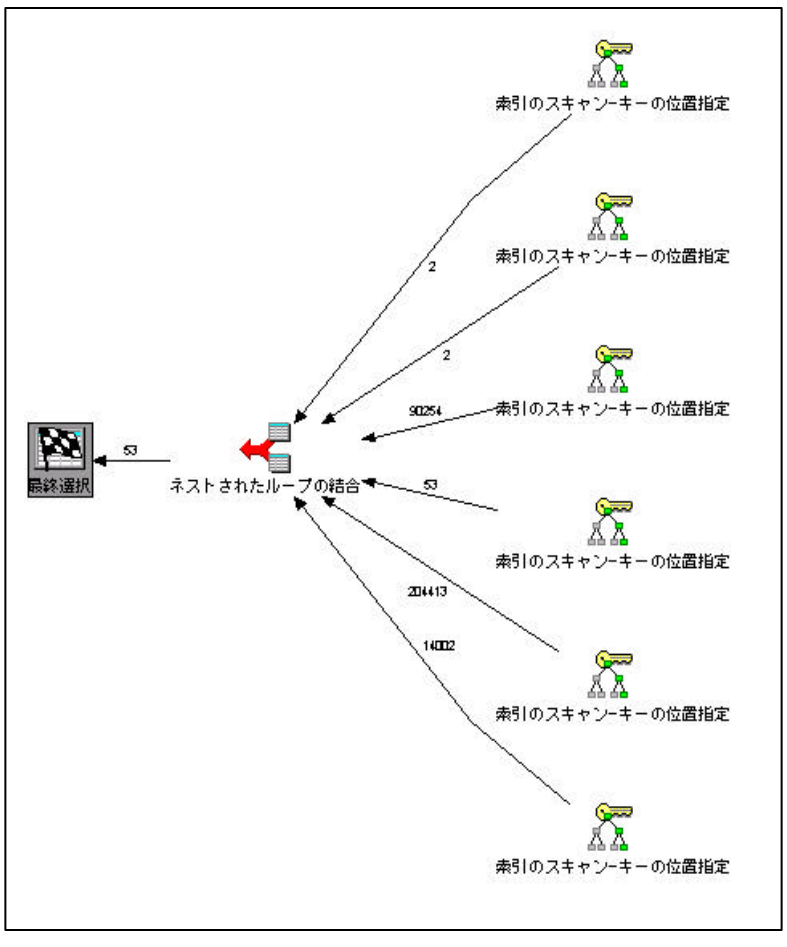
索引によるオプティマイザーへの情報提供が必要

IBM eServer iSeries

Notes:

内部結合における1次テーブルおよびその他のテーブルの結合順は以下の要素によって左右されます。

- ORDER BYまたはGROUP BYの指定
  - ORDER BYまたはGROUP BYが1つのテーブル内の列に指定されている場合には、多くの場合そのテーブルが1次テーブルになります。
- 選択の見積もり行数
  - ORDER BYまたはGROUP BYの指定がない場合、もしくはこれらが複数のテーブルにまたがる複数の項目に対して指定されている場合には、選択の見積もり行数が最も少ないテーブルが1次テーブルになる可能性が高くなります。2次テーブルの中の結合順も基本的には選択の見積もり行数を基準に決定されますが、右のVisual Explainの図のように、必ずしも見積もり行数の少ない順になるとは限りません。これは、テーブルどうしがどのテーブルとの列での結合を指定されているかに依存するためです。
- QAQQINIのFORCE_JOIN_ORDERの指定
  - CHGQRYAコマンドまたはJDBCのオプションの設定によって、照会オプションを設定することができます。
  - この中のオプションの１つとして、テーブルの結合順を左右するFORCE_JOIN_ORDERというオプションを指定することができます。
    *YES　を指定すると、結合順はSQLステートメントに記述されたテーブルの順序どおりに固定されます。
    *NO　を指定すると、結合順はオプティマイザーによって動的に決定されます。
    *PRIMARY　を指定すると、1次テーブルのみがSQLステートメント内のとおりに固定され、2次テーブル内の順序はオプティマイザーが決定します。
  - このオプションはジョブごとに設定することができますが、その実行の条件に応じて動的に設定を変えるようなしくみが必要になります。もしすべての場合においてFORCE_JOIN_ORDERを*YESにしてしまうことになると、すべてのSQLステートメントにおいて結合順が固定され、そのために良好なパフォーマンスが得られないという事態を招く可能性があります。

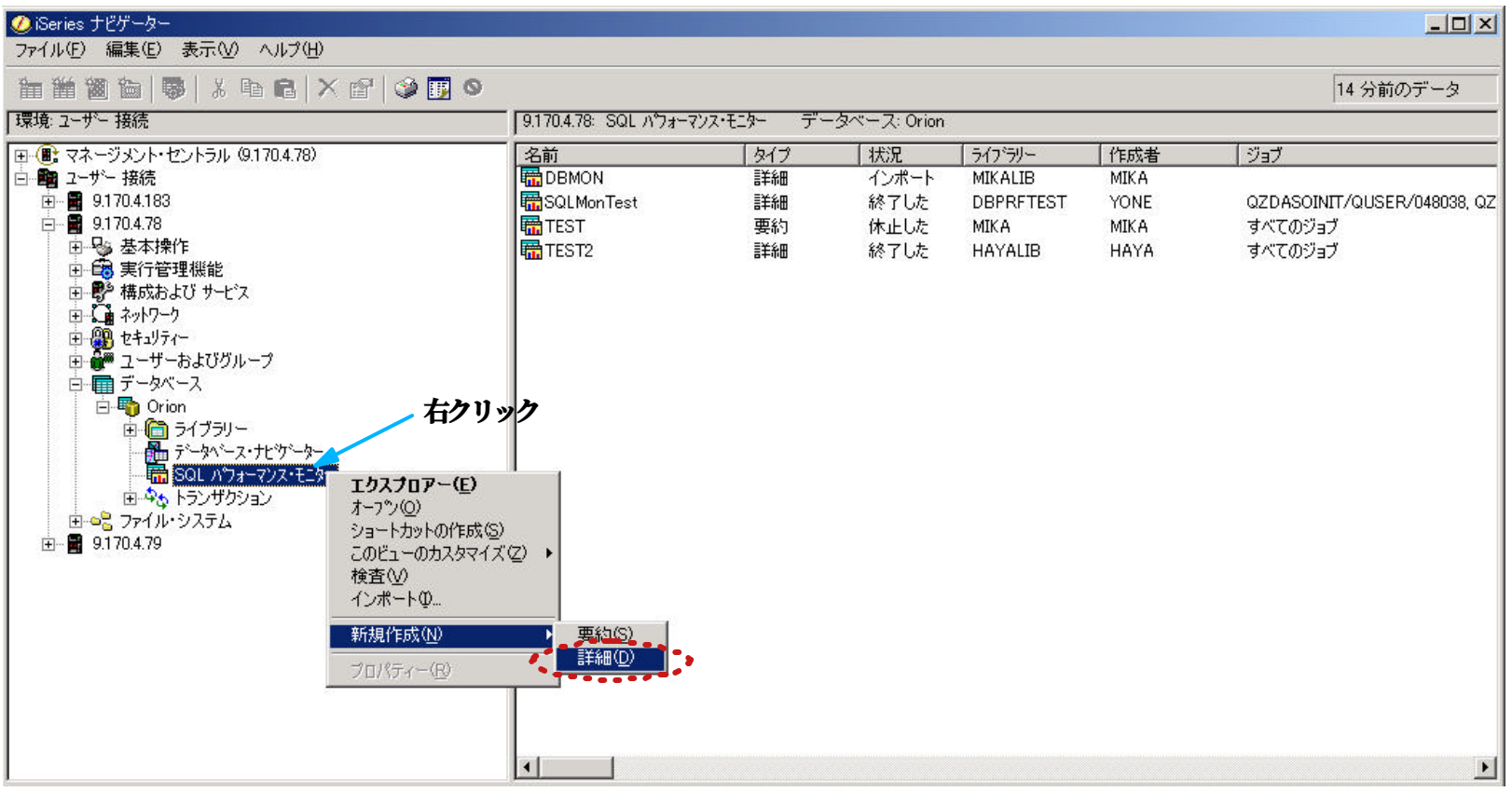


結合順の最適化 外部結合）

- 結合順はSQL文のとおりに固定
- 内部結合との混合の場合
  - 内部結合の結合順のみオプティマイザーが決定

Visual Explain使用手順の概要

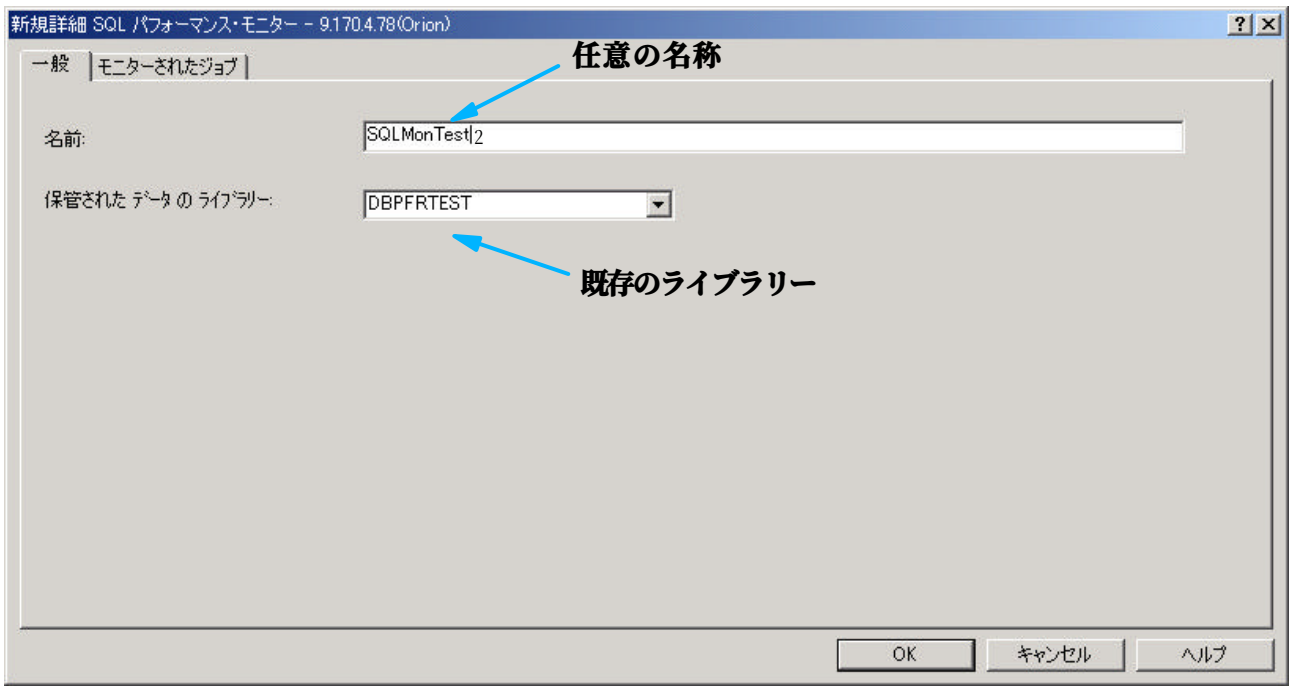
iSeriesナビゲーター
- SQLパフォーマンス・モニターの開始 -



Notes:

Visual Explainの開始方法には2通りあります。1つはiSeriesナビゲーターのSQLスクリプトの実行画面からVisual Explainを使用しながらSQLを実行する方法であり、もう１つは、収集したSQLパフォーマンス・モニター・データからVisual Explainを実行する方法です。
ここではまずSQLパフォーマンス・モニターを使用する方法から説明します。
まず、iSeriesナビゲーターを使用してSQLパフォーマンス・モニターを開始させます。
SQLジョブが稼動するiSeriesサーバーのツリー構造の中から「データベース」を選択し、その下の 「SQLパフォーマンス・モニター」を右クリックして表示されたリストから 新規作成」を選択します。SQLパフォーマンス・モニターでは要約と詳細のいずれかを選択できますが、この場合は 詳細」を選択します。

iSeriesナビゲーター
- 新しい詳細SQLパフォーマンス・モニター -

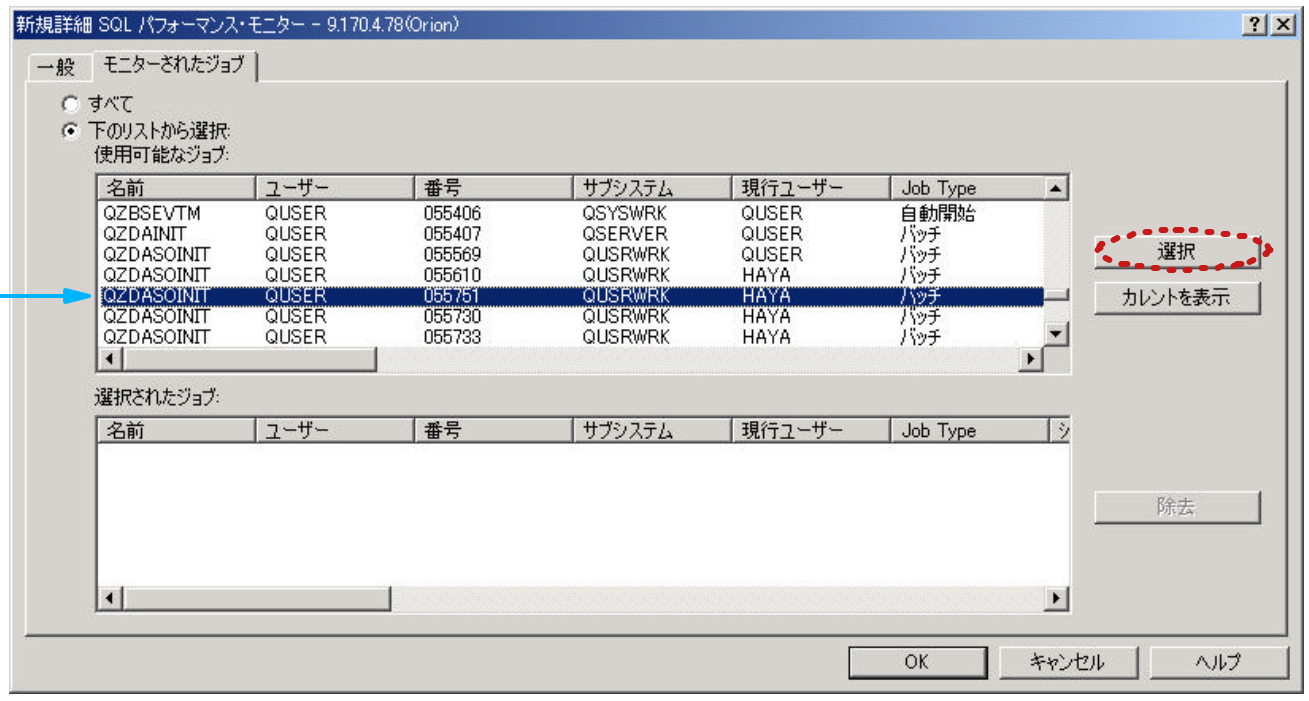


Notes:

SQLパフォーマンス・モニターに任意の名前を指定します。また、このデータが格納される場所として、iSeries上に存在するライブラリー名を指定します。

iSeriesナビゲーター
- モニター対象ジョブ -

ジョブを選択



この例は、Toolbox for JavaのJDBCドライバー使用時のDBアクセス・ジョブであるQZDASOINITを選択しています。

IBM eServer iSeries

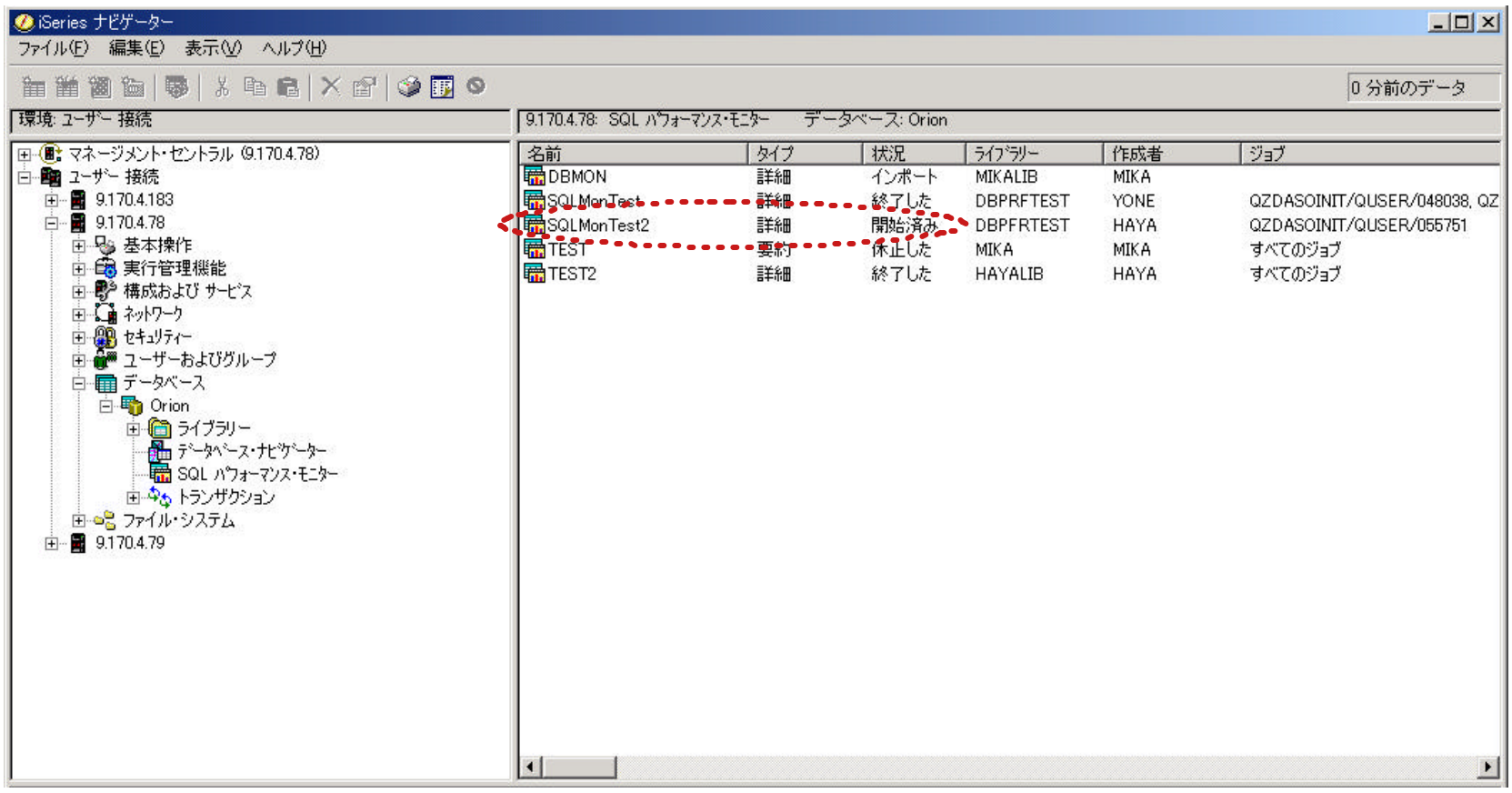
Notes:

モニターの対象とするジョブを選択します。
モニターしたいSQLをこの時点でまだ1度も実行していない場合には、iSeries上に該当するSQLジョブが発生していない可能性がありますので、この場合には「すべて」を選択します。ただしこの場合、SQLを実行するすべてのジョブに対するモニター情報が取得されますので、パフォーマンスおよびディスクの使用率には注意する必要があります。
モニターしたいジョブがすでに存在する場合は、そのジョブを選択することが可能です。Toolbox for JavaのJDBCドライバーでアクセスする場合、iSeriesサーバー上に発生するSQLジョブはQZDASOINITというジョブです。この名称を持つジョブは複数存在しています。（SQLパフォーマンス・モニターもこのジョブを使用します。）
コネクション・プールなどを使用していて、どのジョブが次のSQLで使用されるか確実でない場合には、すべてのQZDASOINITジョブをモニターするように選択します。
最後に「OK」をクリックしてモニターを開始します。

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQLパフォーマンス・モニター
- 状況 = 開始済み -



SQLパフォーマンス・モニターが開始された状態でSQLを実行することにより、データを収集します。

IBM eServer iSeries

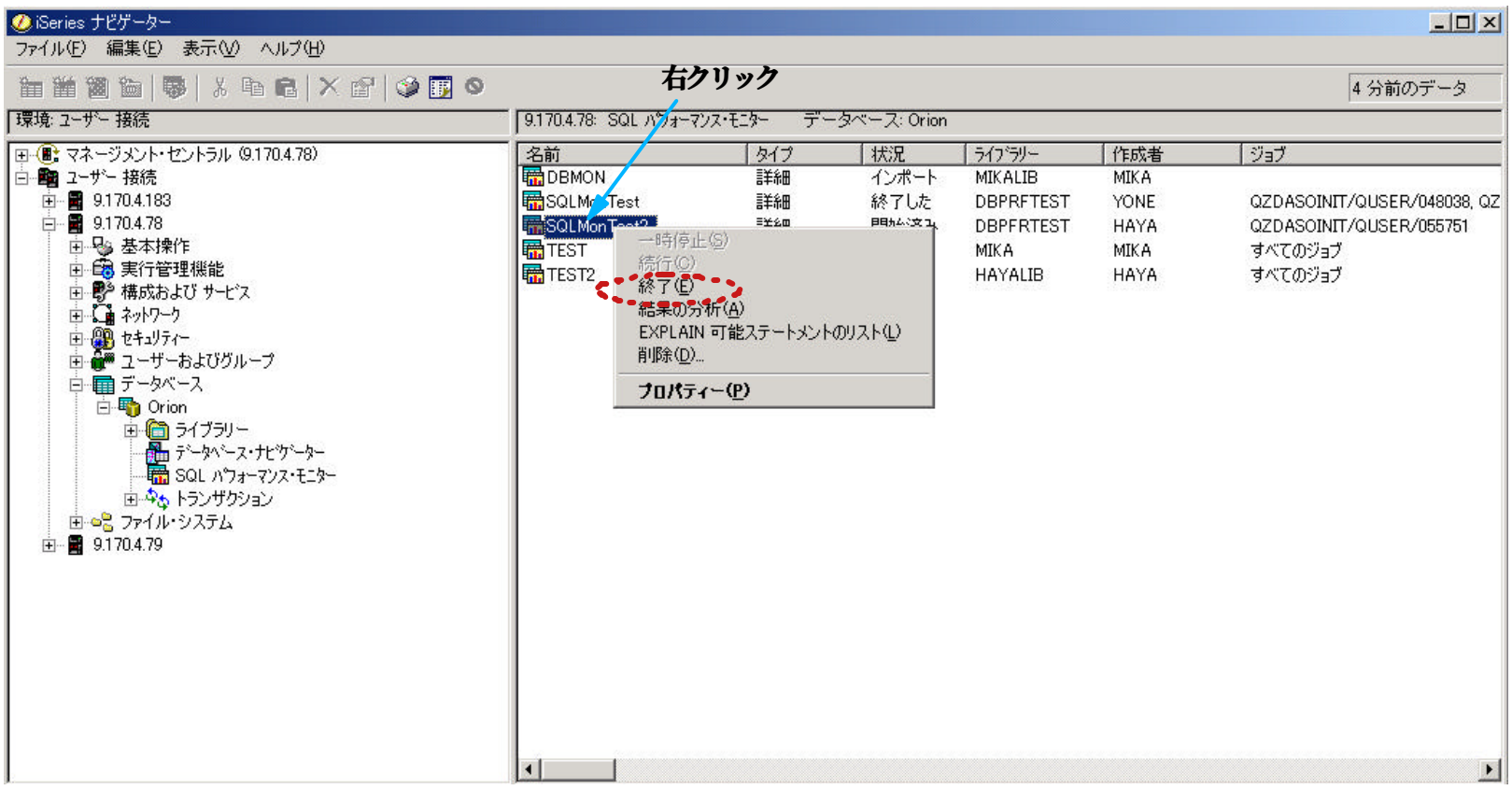
Notes:

上の画面はSQLパフォーマンス・モニターが開始された状況です。
この状況で、テストしたいSQLを実行します。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQLパフォーマンス・モニター - 終了 -



IBM eServer iSeries

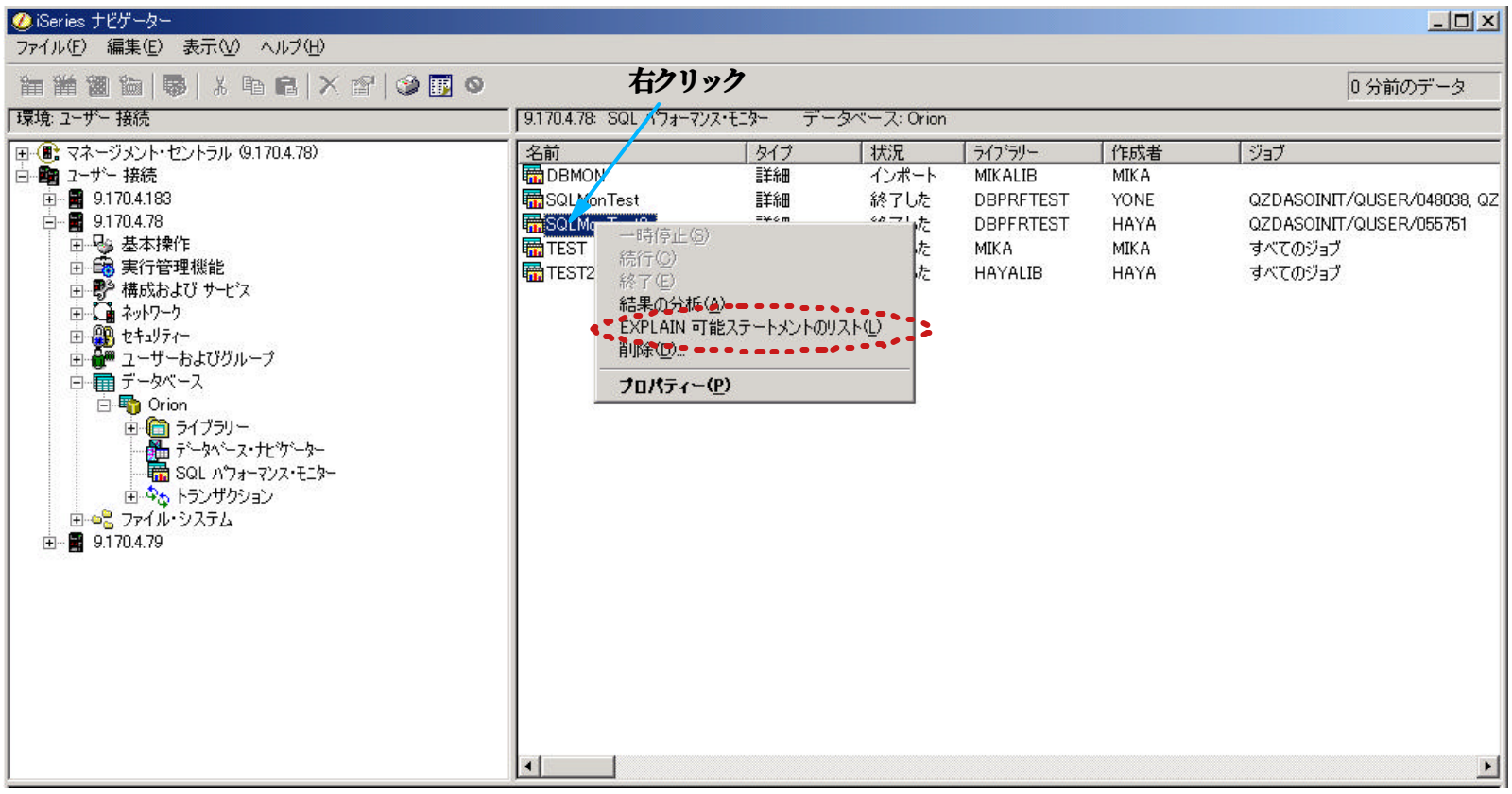
Notes:

SQLを実行し終えたら実行中のモニターを右クリックし、終了を選択してモニターを停止します。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQLパフォーマンス・モニター
- ステートメントのエクスプレーン可能リスト -

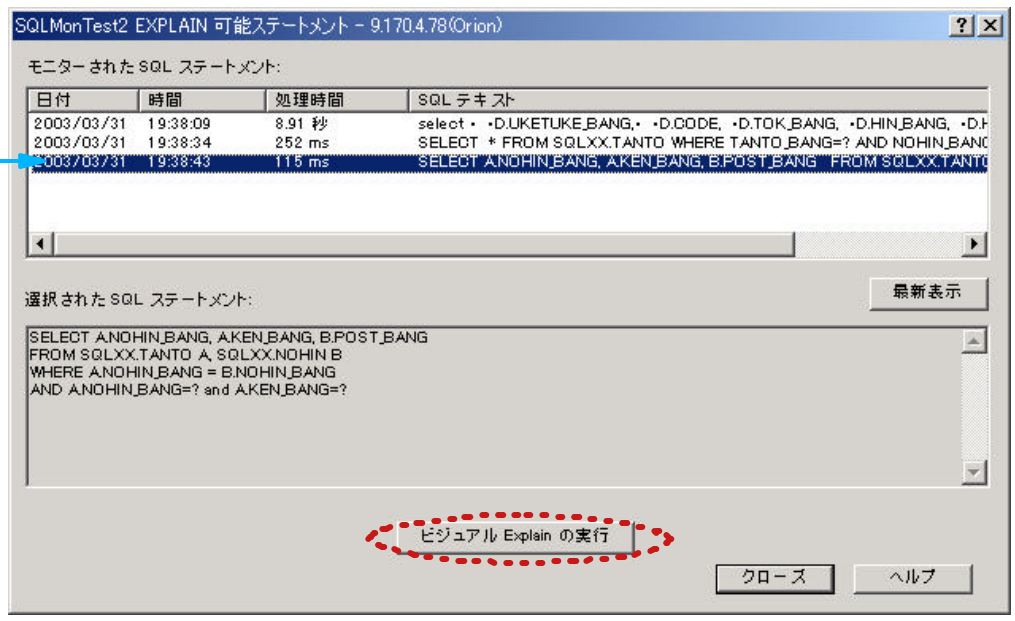


Notes:

次に、ビジュアル・エクスプレーンで表示可能なSQLステートメントを表示するための操作を行います。
取得したSQLパフォーマンス・モニターを右クリックして、「ステートメントのエクスプレーン可能リスト」を選択します。

SQLパフォーマンス・モニター
- ビジュアル・エクスプレーンの実行 -

選択



IBM eServer iSeries

Notes:

リストされたステートメントの中から該当のステートメントを選択して、「ビジュアル・エクスプレーンの実行」をクリックします。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQLパフォーマンス・モニター
- ビジュアル・エクスプレーンの結果 -

IBM eServer iSeries

Notes:

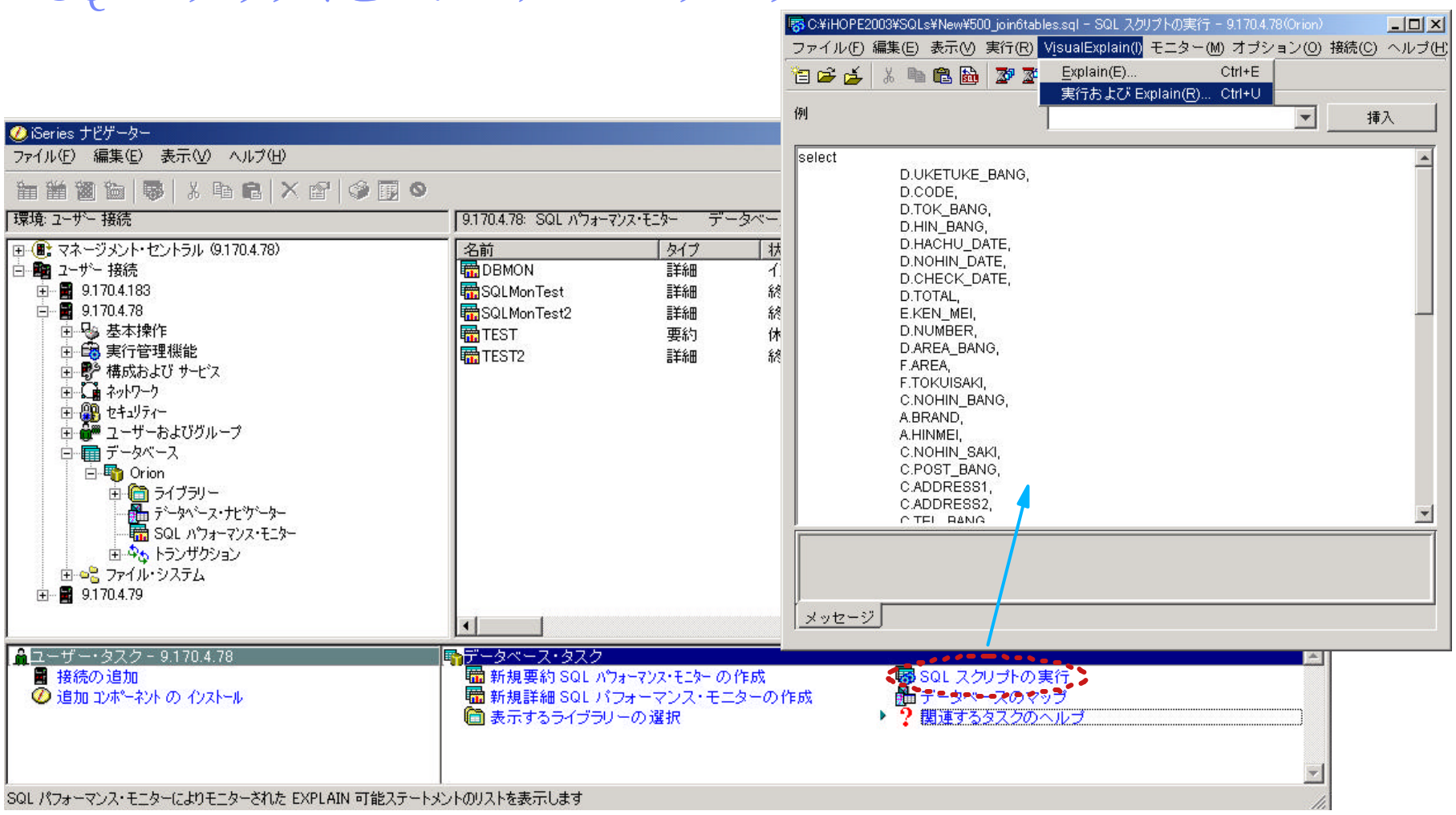
選択されたSQLに対してビジュアル・エクスプレーンが実行されると、上図のようにアクセス・プランがグラフィカルに表示されます。画面の右側には画面の左側で選択されたアイコンの詳細な属性が表示されます。

右上には、「テーブル・スキャン」のアイコンを選択したときの詳細を表示しており、この内容を参照することによって、このテーブルがどのような理由でテーブル・スキャンとして処理されたかを判断することができます。この例では「テーブル・スキャンの理由コード」として「索引が存在しません」というメッセージが表示されており、使用可能な索引が存在しないためであることがわかります。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

最適化プログラム・メッセージの表示
- SQLスクリプトとビジュアル・エクスプレーン -



IBM eServer iSeries

Notes:

アクセス・プランをビジュアルに把握するのと同時に、最適化プログラムの通知メッセージを分析することもパフォーマンス・チューニングの重要な要素となります。
最適化プログラムのメッセージはiSeriesサーバー上のジョブログ中に書き出されますが、ビジュアル・エクスプレーンを使用してこのメッセージを確認することが可能です。
ただしこの場合、SQLステートメントはiSeriesナビゲーターの「SQLスクリプトの実行」機能を使用して実行する必要があります。

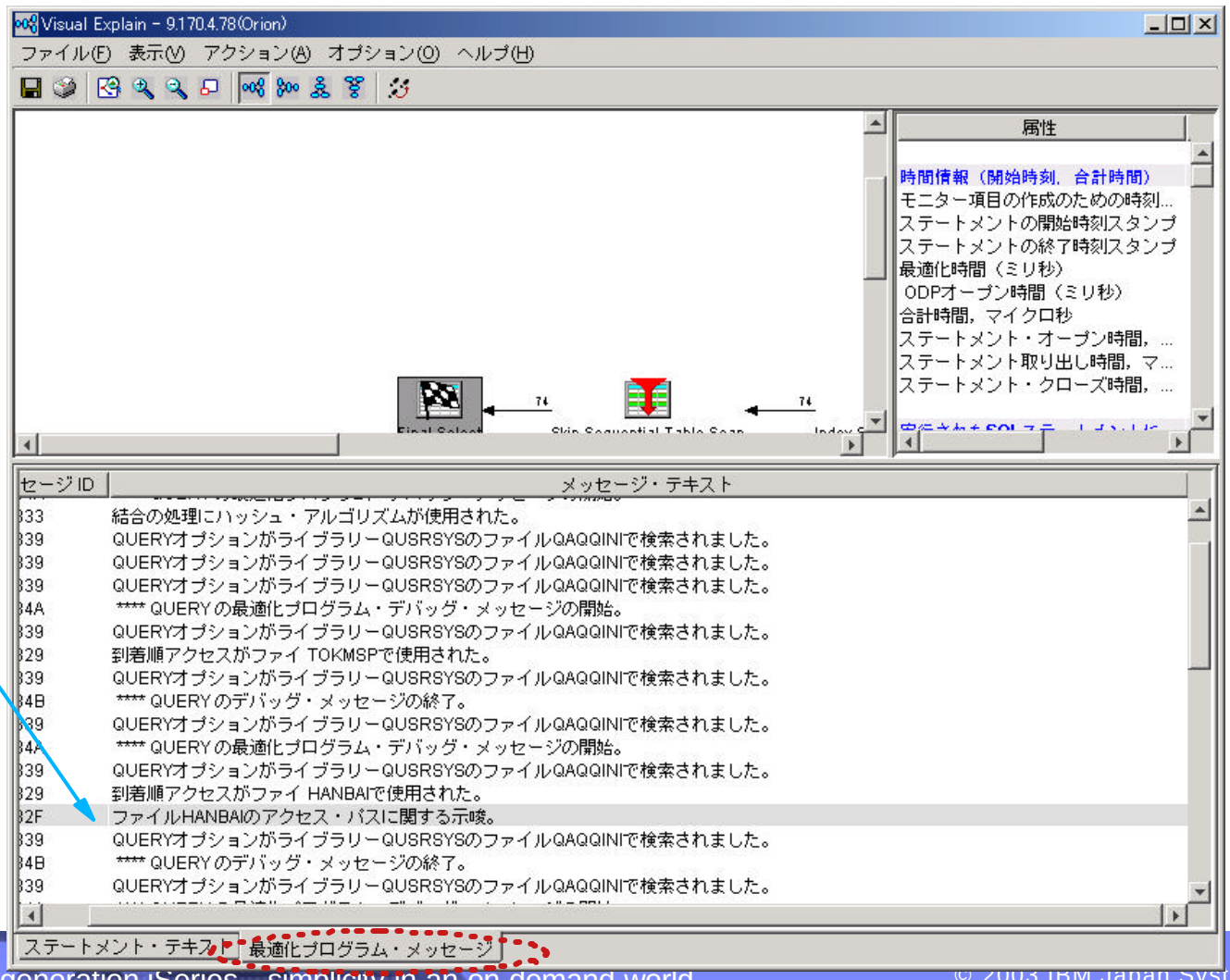
1.画面下の「SQLスクリプトの実行」を選択します。
2.表示された画面にSQLステートメントを入力します。ステートメントがファイルとして保管されている場合には、それをオープンすることも可能です。
3.SQLステートメントが入力された状態で、ツール・バーから「Visual Explain」を選択し、「エクスプレーン」または「実行およびエクスプレーン」を選択します。
「エクスプレーン」を選択した場合、SQLステートメントの結果は返されず、ビジュアル・エクスプレーンの結果だけが表示されます。 「実行およびエクスプレーン」を選択した場合はSQLステートメントの結果も表示されます。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

ビジュアル・エクスプレーン
- 最適化プログラム・メッセージ -

索引アドバイザーのメッセージ例



Notes:

表示されたビジュアル・エクスプレーンの画面の下方にある「最適化プログラムメッセージ」というタブをクリックすることにより、このジョブで出力された最適化プログラムの通知メッセージを見ることができます。
上の例では「ファイルxxxのアクセス・パスに関する示唆」というメッセージが出力されており、永久索引の作成を推奨しています。
このメッセージをダブルクリックにより展開することで、どの列に対して索引を作成するべきかを知ることができます。

最適化メッセージと処置

- ジョブログ内の通知メッセージによりSQLステートメントを評価
  - CPI xxxx
  - SQL xxxx
- メッセージの一覧および対処方法はマニュアルを参照
- 「DB2 UDB for iSeries データベース・パフォーマンスおよびQuery最適化 V5R2」 第4章

IBM eServer iSeries

Notes:

このような最適化メッセージにより、最適化プログラムの動きを把握し、必要な処置を検討することができます。
最適化プログラムのメッセージの意味の詳細と必要な処置については、マニュアル「DB2 UDB for iSeries データベース・パフォーマンスおよびQuery最適化 V5R2」 の第4章に記載がありますので、チューニングに役立てることが可能です。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

索引に関するその他の考慮点

索引付けに関するその他の考慮点

- VARCHAR列への索引付けは避ける
- 索引専用アクセスにnullデータを持つ列は使用不可
- 計算結果を条件にする検索では索引は使用されない
- 同じテーブルの列同士の比較に索引は使用されない
- 索引専用アクセスを検討 選択列がすべてキーに含まれる場合）
- 頻繁に選択する列を索引に入れ、不等号で指定する列は1つにとどめる
- 固有データの列にはUNIQUEを指定して索引を作成する

IBM eServer iSeries

Notes:

VARCHAR列への索引付けは避ける
　　このタイプの列に作成された索引は統計情報が限られており、最適化プログラムにとって利用価値が低い。
索引専用アクセスにnullデータを持つ列は使用不可
　　nullが可能な列が索引に含まれる場合は索引専用アクセスは使用できない。
計算結果を条件にする検索では索引は使用されない
　　照会が以下のような述部を持つ場合：
　　WHERE SHIPDATE > (current_date - 10) or UPPER(customer_name) = 'SMITH '
　　最適化プログラムはこのような選択に索引を使用しない。
同じテーブルの列同士の比較に索引は使用されない
　　以下のようなステートメントでSHIPDATEとORDERDATEが同じテーブルの列である場合、最適化プログラムは索引を使用しない。
　　WHERE SHIPDATE > ORDERDATE
索引専用アクセスを検討 選択列がすべてキーに含まれている場合）
　　照会で使用される列がすべて索引に含まれている場合は、最適化プログラムは索引専用アクセス(IOA)を選択することができる。
　　IOAではすべてのデータが索引上にあるため、実際のテーブルからのデータの読込は行なわれない。
頻繁に選択する列と一緒に不等号に使用する列を索引に入れる
　　最適化プログラムは不等号を値の範囲の絞込みに使用するため、不等号を使用する列を索引に1つ以上入れることによりパフォーマンス上多少の利点がある。
固有データの列にはUNIQUEを指定して索引を作成する

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

索引の保守

- 索引保守のタイミング
  - 2進化基数索引
    - 即時　(省略時の設定)
    - 遅延
    - 再作成
  - コード化ベクトル索引
    - 即時
- 最適化プログラムの動作
  - 保守オプションが再作成の索引は選択しない
  - 保守オプションが遅延の索引の選択には負荷がかかる
    - 保留されている変更の影響を見積もるため

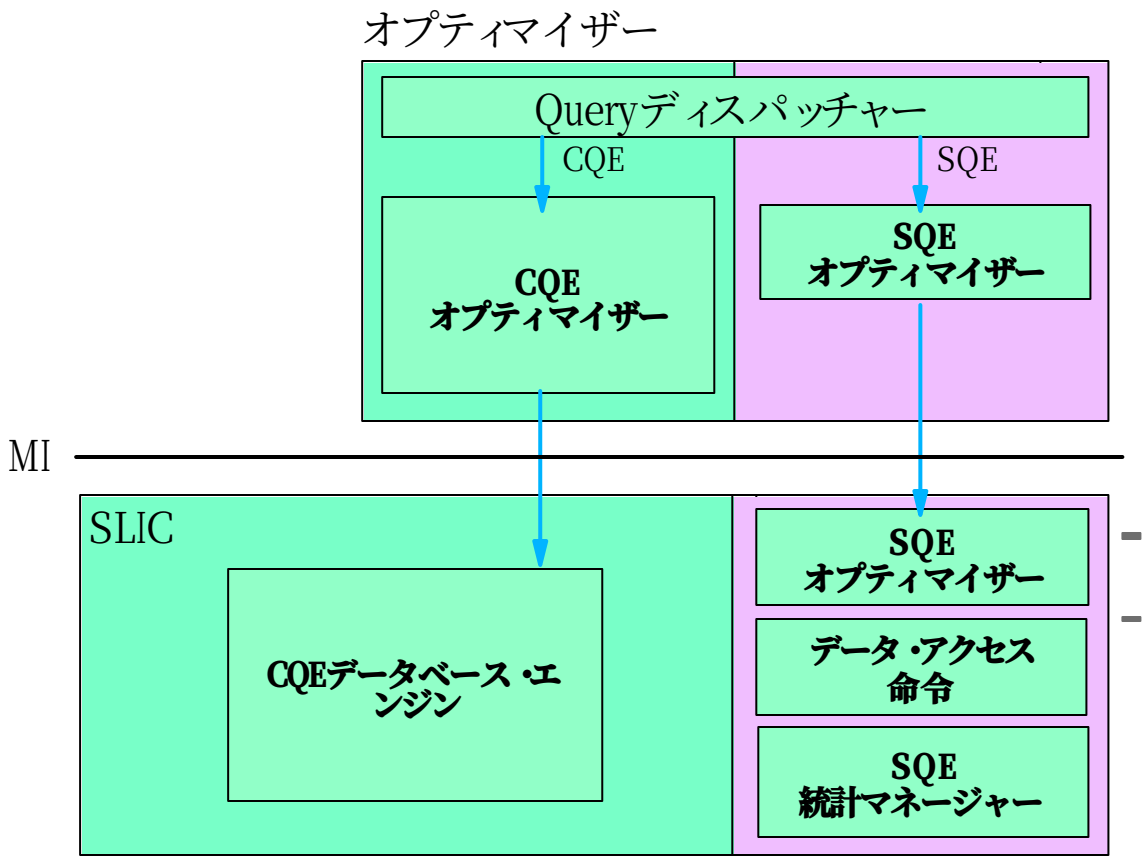
行の追加/更新と索引の保守

- 索引の数は発生しうる追加/更新数を考慮して決定
- SMPによる並行処理は索引の保守に有効
- 多くの行を更新/追加する場合には索引を削除し再作成する
  - 総数の20%を越える行の追加/更新時

IBM eServer iSeries

OS/400 V5R2における機能拡張

V5R2 データベース・アークテクチャー



- オプティマイザーとデータベース・エンジンを統合しSQL照会エンジンを構成
- 処理の多くはSLICに移行

Notes:

OS/400 V5R2より、SQEがCQEを補完します。これに伴い、いくつかの新しいコンポーネントが導入されています。

- Queryディスパッチャー
  - 要求された照会をCQEとSQEのどちらで処理するかを判断します。
- SQEオプティマイザー
  - アクセス・プランのコスト評価をより効率的にできるよう拡張された、新規のオプティマイザーです。
- 統計マネージャー
  - 新規のオプティマイザーに提供するための統計情報を管理するための機能です。
- データアクセス命令(Primitives)
  - 新たにSQL用に設計された、データアクセスのためのSLIC上の機能です。従来のDBエンジンはレコード・レベル・アクセス用に設計されたものを改造したものです。CQEにディスパッチされる照会は従来のSLIC DB機能で実行されますが、SQEにディスパッチされる照会はこの新しいDBアクセス命令で実行されます。

Queryディスパッチャー

- 以下の場合はCQEにディスパッチ
  - テーブル数 > 1
  - OR および IN 術部の使用
  - SMPの使用
  - Read以外
  - LIKE述部の使用
  - UNION
  - ビューまたは論理ファイルの使用
  - 副照会
  - 派生および共通表形式(with句の使用)
  - LOB型式の列
  - 列間のNLSS/CCSIDによるデータ変換
  - DB2マルチシステム
  - SQL以外での照会

IBM eServer iSeries

Notes:

照会が上記の条件に当てはまる場合、Queryディパッチャーはこれらを CQE にディパッチします。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQEオプティマイザー

- 新規のデータベース統計を使用してアクセス・プランのコストを評価
  - システムとテーブルに関する質問/回答形式で判断
- 一時索引への依存度が低い
  - ハッシュ・テーブル/テーブル走査/一時結果ファイルを多用

IBM eServer iSeries

Notes:

SQL照会オプティマイザーは、オブジェクト指向技術に基づく最適化機能の実現により拡張されました。このオブジェクト指向フレームワークは、新しい最適化技術を実現し、オプティマイザーの将来的な拡張を可能にするものです。

オプティマイザーの拡張機能の１つは、照会のアクセス・プランのコスト評価の拡張です。SQEによる拡張の恩恵を受ける照会においては、以前オプティマイザーが使用することのできた以上の多くの情報を、照会プランのコスト評価の段階で使用することができます。オプティマイザーは、新規に利用可能になったデータベース統計を使用して、アクセス・プランの選択のためにより正確な判断をすることができます。また、拡張されたオプティマイザーは、一時索引を作成するアクセス・プランよりも、ハッシュ・テーブルと一時結果のソートを使用して照会処理中の部分的な照会結果を保持するプランをより頻繁に選択する可能性があります。一時索引への依存度が低いため、SQEオプティマイザーは、一時索引作成のためのオーバーヘッドを軽減できる、より効率のよいプランを選択することができ、単一記憶域の恩恵を最大限に享受することができます。オプティマイザーの変更は、拡張されたデータベース・エンジンにとってより効率的なアクセス・プランを作成するようにデザインされました。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

SQEデータベース統計

- すべてのQueryは統計情報に依存
  - DB2 UDB for iSeriesは統計情報源として索引を使用
  - 他のデータベースでは手動による統計収集が必要
- SQEは混合アプローチ
  - 索引が存在しない場合に統計情報が自動的に収集される
  - 手動による統計情報の収集も可能

Notes:

過去においては、データベース統計は索引が作成されている列に対して最適化の時点で自動的に収集されていました。
SQEでは、索引を持たない列に対する統計情報も収集することができ、最適化の段階で使用することができます。列の統計には、ヒストグラム、頻繁に使用される値のリスト、および列のカーディナリティーが含まれます。

他の多くのプラットフォームでは、統計の収集はデータベース管理者の責任により手動で行われる処理であるのに対し、iSeriesサーバーでは、データベース統計の収集処理は自動的に行われます。統計情報の手動での更新も可能ですが、手動で更新しなければならないことはほとんどありません。統計情報の収集や更新が必要なときに、統計マネージャーがどの列の統計情報が必要であるかを判断します。統計情報は低優先順位のジョブとしてバックグラウンドで収集され、他のジョブへの影響を最小限にします。手動での統計情報の収集は通常のジョブの優先順位で実行されます。

システムは、実行される照会に基づいて、どの列に対する統計情報を収集するかを自動的に決定します。そのため、期待どおりのパフォーマンスが出ない照会においては、必要な統計情報が使用できるようになっているかどうかを確認する必要があります。また、実行に時間のかかる照会が一度だけ実行されるような環境では、実行の前に統計情報が使用可能であるかを確認することで効果があります。

データベース列の統計情報のプロパティー：

- 使用する記憶域は少なく、列あたり8-12KB
- 列の統計情報は、含まれるすべての列に対してデータベース・ファイルの1度のフル・スキャンによって収集される。
- 列の統計情報は、統計更新メカニズムにより定期的に更新され、その場合データベース・ファイルのフル・スキャンが必要になる。
- 列の統計情報は１つの小さなデータ構造に収められ、照会の最適化の時に少ないI/Oでメモリーにページ・インすることができる。

統計マネージャーによる照会の分析

- システム値
  - QDBFSTCCOL
    - *ALL　　ユーザーとシステムの両者から収集要求が可能
    - *NONE　　統計情報の収集不可
    - *USER　　ユーザーからのみ収集要求可能
    - *SYSTEM　システムからの自動収集のみ可能
- ジョブ
  - QDBFSTCCOL
    - 自動収集のための低優先順位のバックグラウンド・ジョブ
- iSeriesナビゲーター
  - 統計の収集をGUIで管理

IBM eServer iSeries

Notes:

V5R1以前は、重複した値や固有値の数などのDBに関する統計情報は、索引を通じてのみ収集可能でした。V5R2では、統計情報の収集はオプティマイザーから独立し、統計マネージャーの中に組み込まれました。照会をどのように実行するかを決定する際に、オプティマイザーはデータに関する統計情報を統計マネージャーに尋ねます。統計マネージャーは、索引または統計を通じて使用可能なすべての情報を検索し、オプティマイザーの質問に答えます。その回答は索引または統計から得たものであるか、それらが使用可能でない場合には省略時の値になります。
さらに、統計マネージャーはシステムによって収集されるデータの決定も行います。

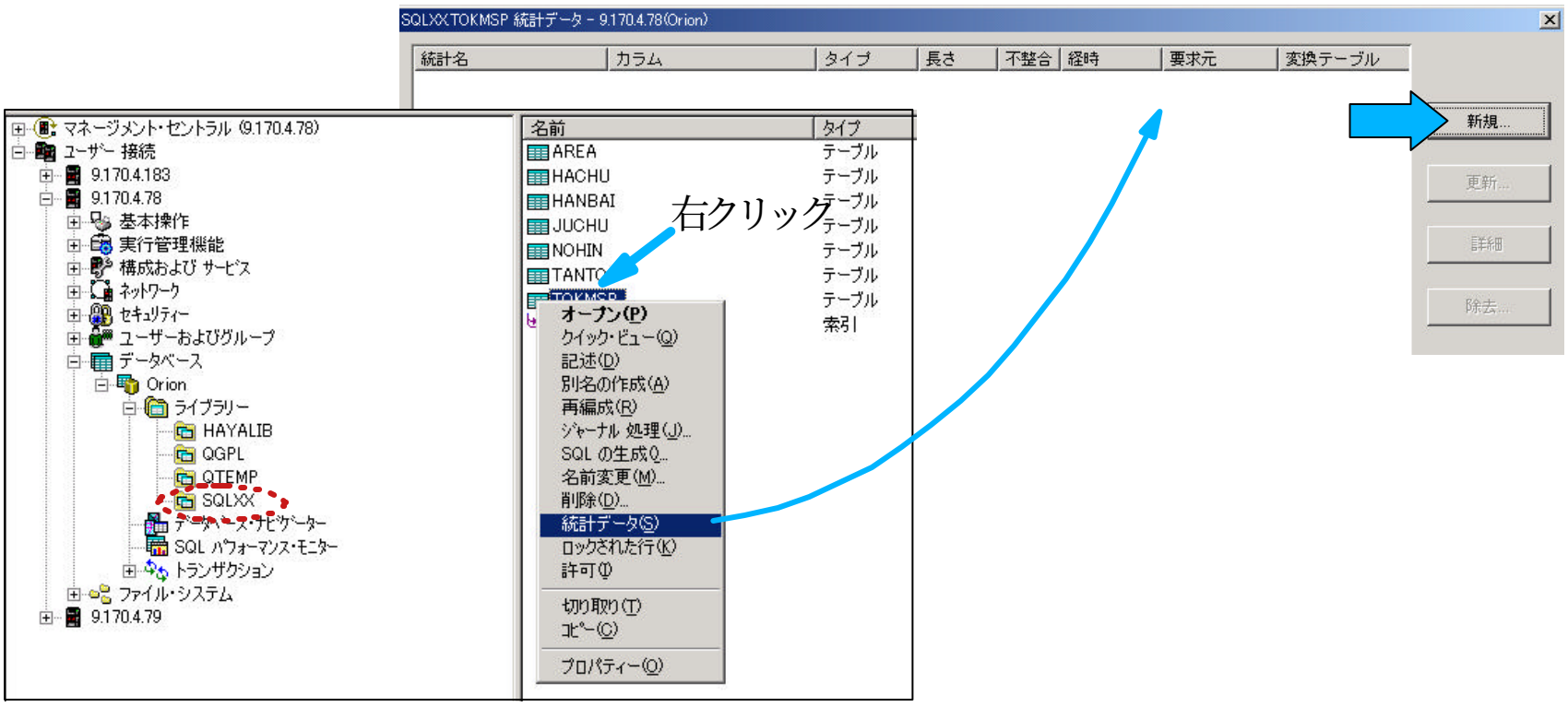
注：
この新規の統計情報はSQEによってのみ使用可能であり、CQEから使用することはできません。また、CQEにディスパッチされるSQLによって統計情報が収集されることもありません。

The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

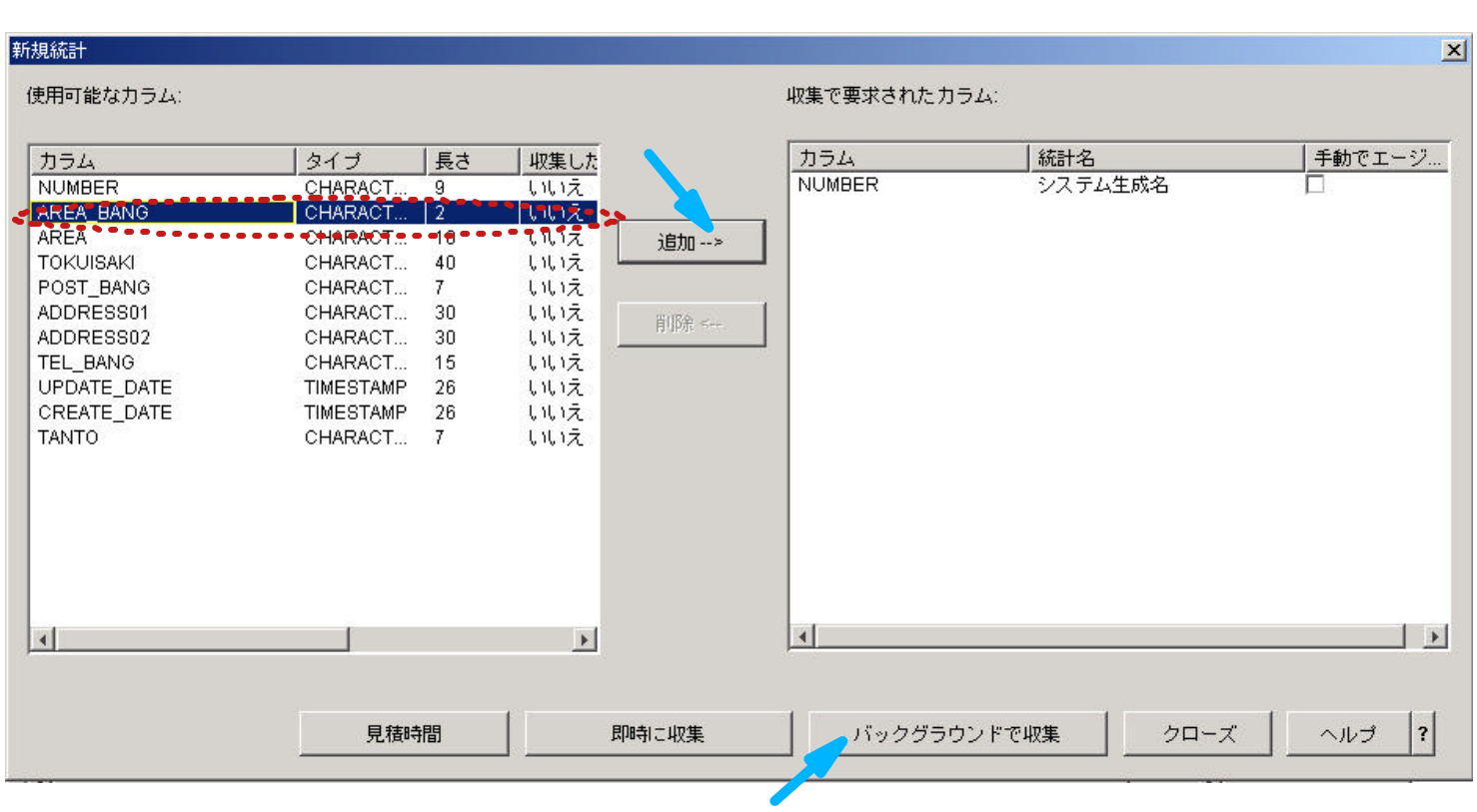
iSeriesナビゲーターによる統計情報の管理

統計の作成



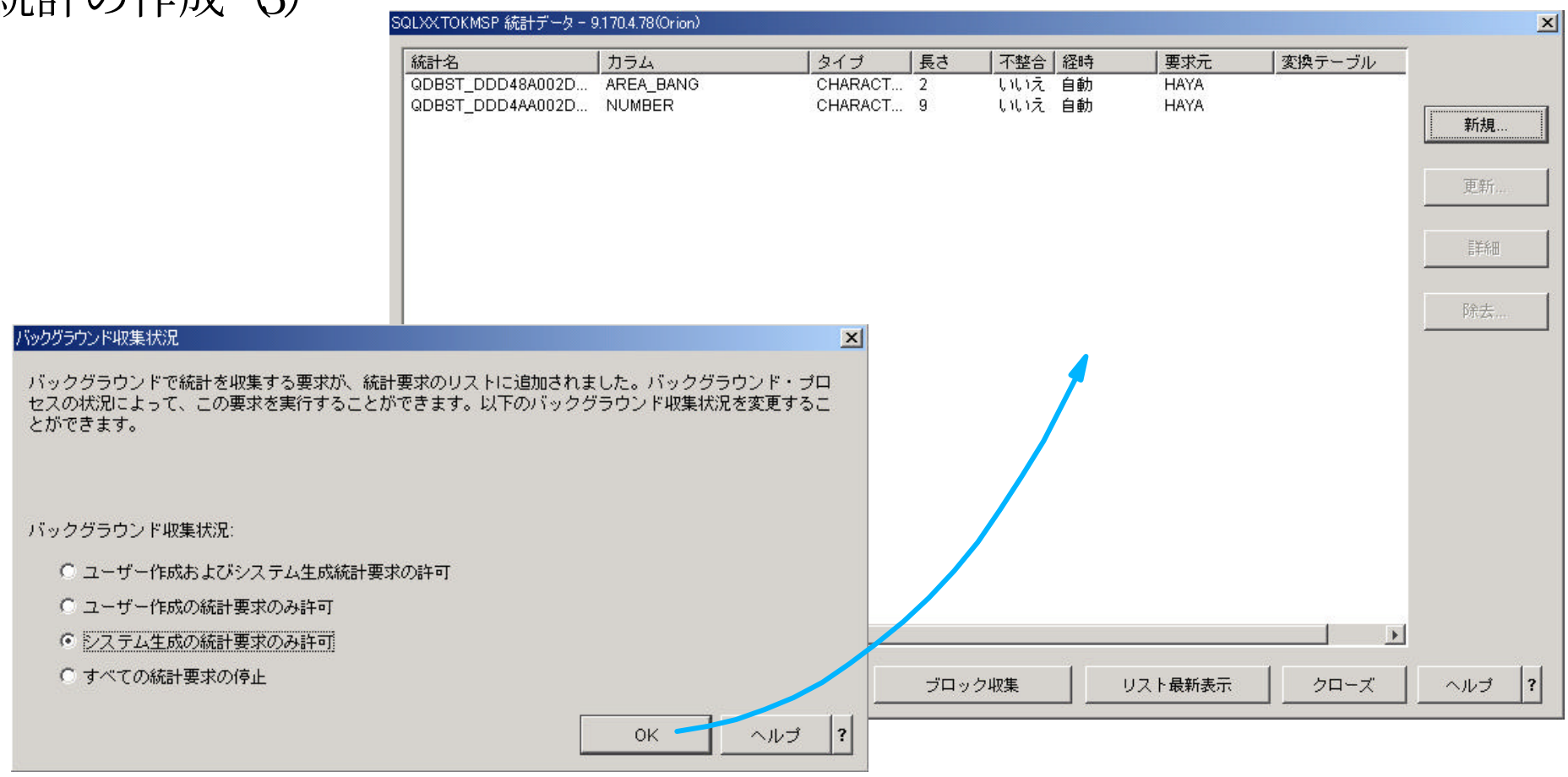
iSeriesナビゲーターによる統計情報の管理

統計の作成 (2)



iSeriesナビゲーターによる統計情報の管理

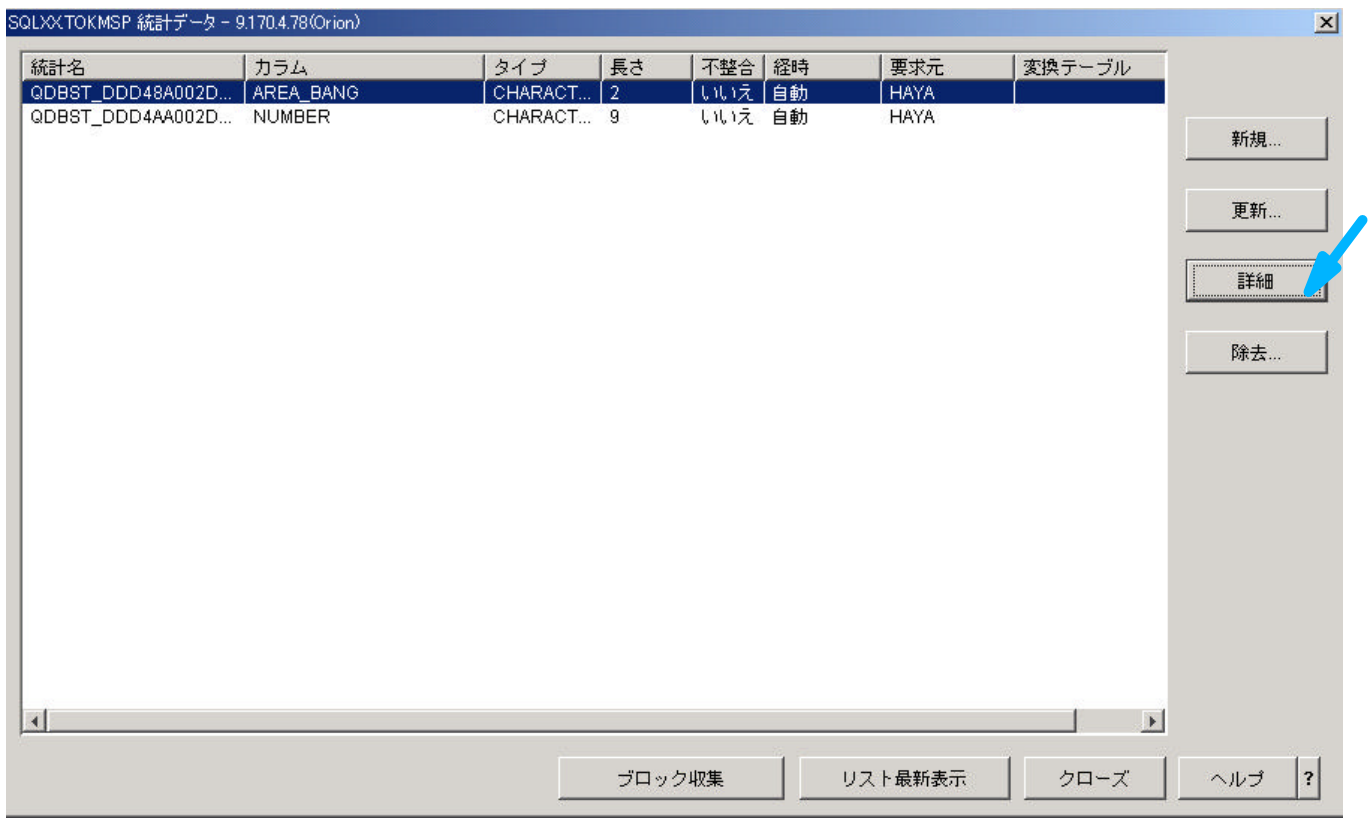
統計の作成 (3)



IBM eServer iSeries

iSeriesナビゲーターによる統計情報の管理

統計の表示

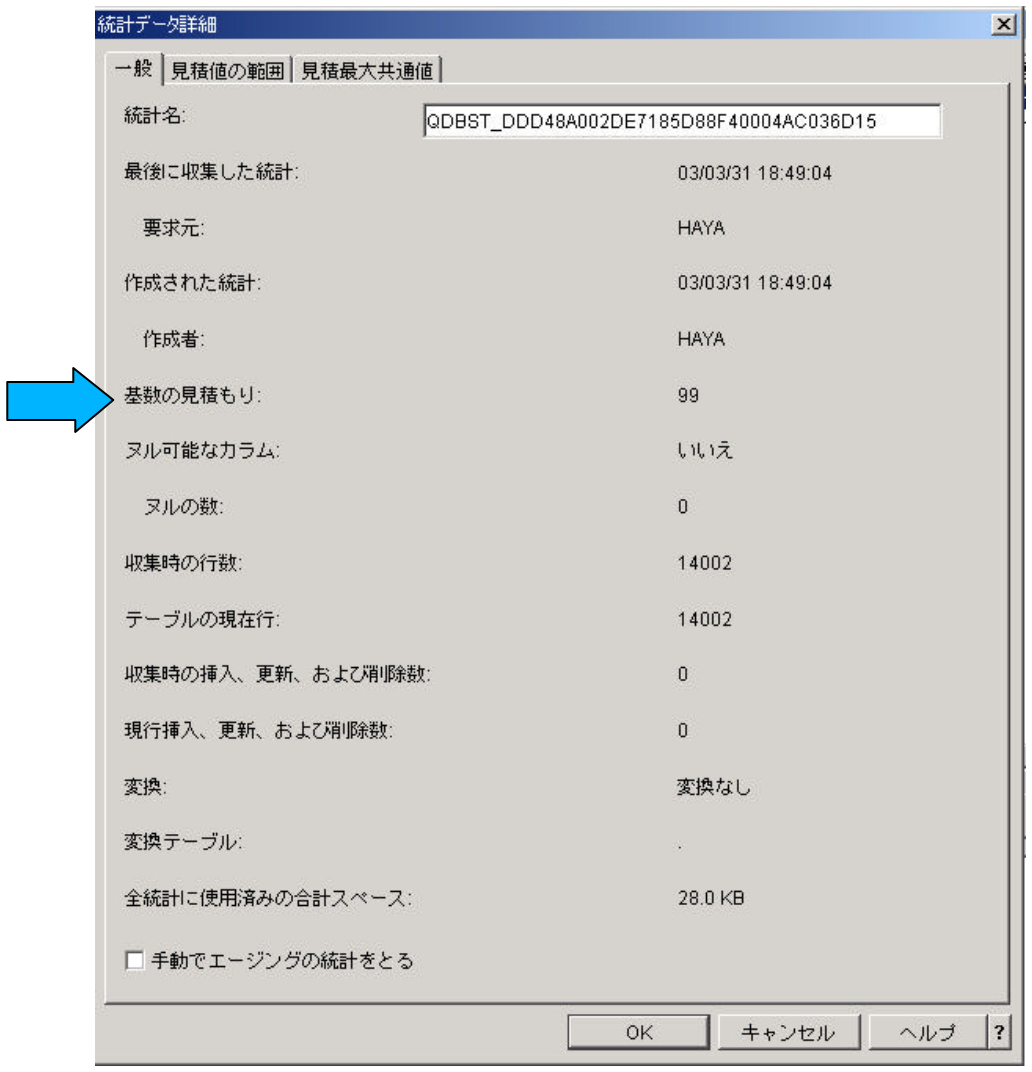


The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

iSeriesナビゲーターによる統計情報の管理

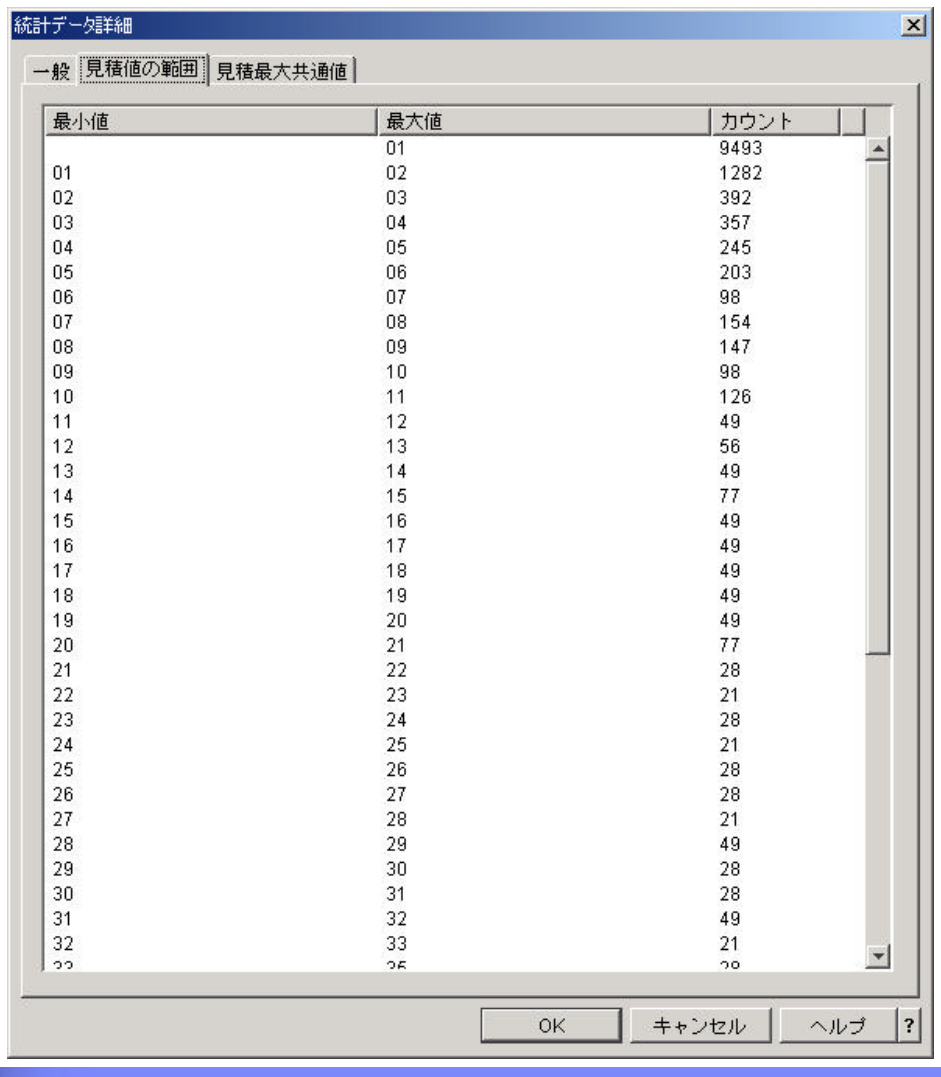
列内の固有値の数



IBM eServer iSeries

iSeriesナビゲーターによる統計情報の管理

列内の値の分布



The next generation iSeries…simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

iSeriesナビゲーターによる統計情報の管理

列内での出現頻度順の値とカウント