

IBM

@server

iSeries

DB2 マルチ・システム

バージョン 5





@server

iSeries

DB2 マルチ・システム

バージョン 5

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

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

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

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

(URL は、変更になる場合があります)

原 典： iSeries
DB2 Multisystem Version 5

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2002.8

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

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

© Copyright International Business Machines Corporation 1998, 2001, 2002. All rights reserved.

© Copyright IBM Japan 2002

目次

DB2 マルチ・システムについて	v
DB2 マルチ・システムの対象読者	v
コードについての特記事項	v
第 1 章 DB2 マルチ・システムの概要	1
DB2 マルチ・システムを使用する利点	3
DB2 マルチ・システム: 基本的な用語と概念	3
第 2 章 DB2 マルチ・システムでのノード・グループの概要	7
DB2 マルチ・システムでのノード・グループの仕組み	7
DB2 マルチ・システムでノード・グループ・コマンドを使用する前に完了しておくべきタスク	8
DB2 マルチ・システムでの CRTNODGRP コマンドを使用したノード・グループの作成	9
DB2 マルチ・システムでの DSPNODGRP コマンドを使用したノード・グループの表示	11
DB2 マルチ・システムでの CHGNODGRPA コマンドを使用したノード・グループの変更	13
DB2 マルチ・システムでの DLTNODGRP コマンドを使用したノード・グループの削除	14
第 3 章 DB2 マルチ・システムでの分散ファイルの作成	15
DB2 マルチ・システムでの分散物理ファイルの作成	15
DB2 マルチ・システムにおいて分散ファイルを作成または処理するときの制限	17
DB2 マルチ・システムでの分散ファイルの使用	18
DB2 マルチ・システムでの分散ファイルに対する CL コマンドの実行	20
CL コマンド: DB2 マルチ・システムでの分散ファイルに対して実行できないもの	20
CL コマンド: DB2 マルチ・システムでの分散ファイルのローカル部分にのみ影響を与えるもの	21
CL コマンド: DB2 マルチ・システムでの分散ファイルのすべての部分に影響を与えるもの	22
DB2 マルチ・システムでのジャーナル処理の考慮事項	23
DB2 マルチ・システムにおける分散ファイルでのファイルのコピー (CPYF) コマンドの使用	24
DB2 マルチ・システムでの区分化	25
DB2 マルチ・システムでの区分化の計画	26
DB2 マルチ・システムでの区分化キーの選択	27
DB2 マルチ・システムでのデータ分配のカスタマイズ	28

第 4 章 DB2 マルチ・システムで使用可能なスカラー関数	31
DB2 マルチ・システムでの PARTITION	31
DB2 マルチ・システムでの PARTITION の例	31
DB2 マルチ・システムでの HASH	32
DB2 マルチ・システムでの HASH の例	32
DB2 マルチ・システムでの NODENAME	33
DB2 マルチ・システムでの NODENAME の例	33
DB2 マルチ・システムでの NODENUMBER	34
DB2 マルチ・システムでの NODENUMBER の例	34
DB2 マルチ・システムでの特殊レジスタ	34
DB2 マルチ・システムでの相対レコード番号 (RRN) 関数	35
第 5 章 DB2 マルチ・システムでのパフォーマンスとスケラビリティ	37
DB2 マルチ・システムの使用	37
DB2 マルチ・システムでのパフォーマンス向上のヒント	39
データベース・システムの拡張における DB2 マルチ・システムの効用	39
ネットワークにシステムを追加する際の再分配の問題	40
第 6 章 DB2[®] マルチ・システムでのパフォーマンスのための QUERY 設計	43
DB2 マルチ・システムでの最適化の概要	44
DB2 マルチ・システムでの単一ファイル QUERY の実施と最適化	44
DB2 マルチ・システムでのレコード順序の実施と最適化	46
DB2 マルチ・システムでの UNION 文節と DISTINCT 文節の実施と最適化	47
DB2 マルチ・システムでの DSTDTA パラメーターおよび ALWCPYDTA パラメーターの処理	47
DB2 マルチ・システムでの結合の実施と最適化	48
DB2 マルチ・システムでの共通設定結合	48
DB2 マルチ・システムでの指定結合	50
DB2 マルチ・システムでの再区分化結合	50
DB2 マルチ・システムでの同報通信結合	52
DB2 マルチ・システムでの結合の最適化	52
DB2 マルチ・システムでの結合フィールドに対する区分化キー	53
DB2 マルチ・システムでのグループ化の実施と最適化	53
DB2 マルチ・システムでの 1 ステップのグループ化	54
DB2 マルチ・システムでの 2 ステップのグループ化	54
DB2 マルチ・システムでのグループ化と結合	55

DB2 マルチ・システムでの SUBQUERY サポート	56
DB2 マルチ・システムでのアクセス計画	56
DB2 マルチ・システムでの再使用可能なオープン・データ・パス (ODP)	56
DB2 マルチ・システムでの一時結果書き込み機能	57
一時書き込み機能ジョブ: DB2 マルチ・システムでの利点	59
一時書き込み機能ジョブ: DB2 マルチ・システムでの不利な点	59
DB2 マルチ・システムでの一時書き込み機能の制御	60
DB2 マルチ・システムでの最適化プログラムのメッセージ	60

DB2 マルチ・システムでの QUERY 属性の変更 (CHGQRYA) コマンドへの変更	62
DB2 マルチ・システムでの非同期ジョブ使用 (ASYN CJ) パラメーター	62
リモート適用 (APYRMT) パラメーター	63
パフォーマンスの考慮事項の要約	64

参照文献 65

索引 67

DB2 マルチ・システムについて

本書では、分散リレーショナル・データベース・ファイル、ノード・グループ、区分化など、DB2 マルチ・システムの基本概念について説明します。本書で提供する情報は、複数の iSeries サーバー間で区分化されるデータベース・ファイルの作成と使用に必要なものです。システムの構成方法、ファイルの作成方法、およびアプリケーションにおけるファイルの使用方法について説明してあります。

『DB2 マルチ・システムの対象読者』には、DB2 マルチ・システム (本書) の対象読者についての説明があります。

DB2 マルチ・システムの対象読者

本書は、大量のデータが入っているデータベースを管理するシステム管理者またはデータベース管理者を対象としています。本書の利用者は、データベースの作成方法と使用方法について十分に理解しているとともに、データベース管理とシステム管理に関してよく知っている必要があります。

DB2 マルチ・システム (本書) は、1 ページの『第 1 章 DB2 マルチ・システムの概要』からお読みください。

コードについての特記事項

本書には、プログラミングの例が含まれています。

IBM は、お客様に、このプログラムをサンプルとして使用することができる非独占的な使用権を許諾します。お客様は、このサンプル・コードから、お客様の具体的な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。

IBM は、このサンプル・コードをすべて、例として示す目的で提供しています。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。したがって、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示唆したり保証することはできません。

本書に含まれているすべてのプログラムは現存するままの状態を提供され、いかなる保証条件も適用されません。商品性および特定目的適合性に関する黙示の保証の適用も一切ありません。

第 1 章 DB2 マルチ・システムの概要

DB2 マルチ・システムは、無限ともいえるほどのスケーラビリティ・オプションをデータベースに提供する並列処理技法です。DB2 マルチ・システムを使用すると、複数の iSeries サーバー (最大 32 のサーバー) を「何も共有しない」クラスターに同時に接続することができます。(何も共有しないとは、結合されたネットワーク内の各システムが、独自のメイン・メモリーとディスク装置を所有し、管理するということです。) いったんシステムが結合されたならば、データベース・ファイルを、接続された各システム上の記憶装置に分散することができます。データベース・ファイルは、システムの集合間で区分化 (分散化) されたデータを持つことができます。また、各システムはファイル内のすべてのデータにアクセスすることができます。しかしこのファイルは、ユーザーに対して、そのシステム上にあるローカル・ファイルのように振る舞います。ユーザーから見れば、データベースは単一のデータベースのように見えるため、ユーザーは、ネットワーク内のすべてのシステム間で並列的に QUERY を実行して、ファイル内のデータにリアルタイムでアクセスすることができます。

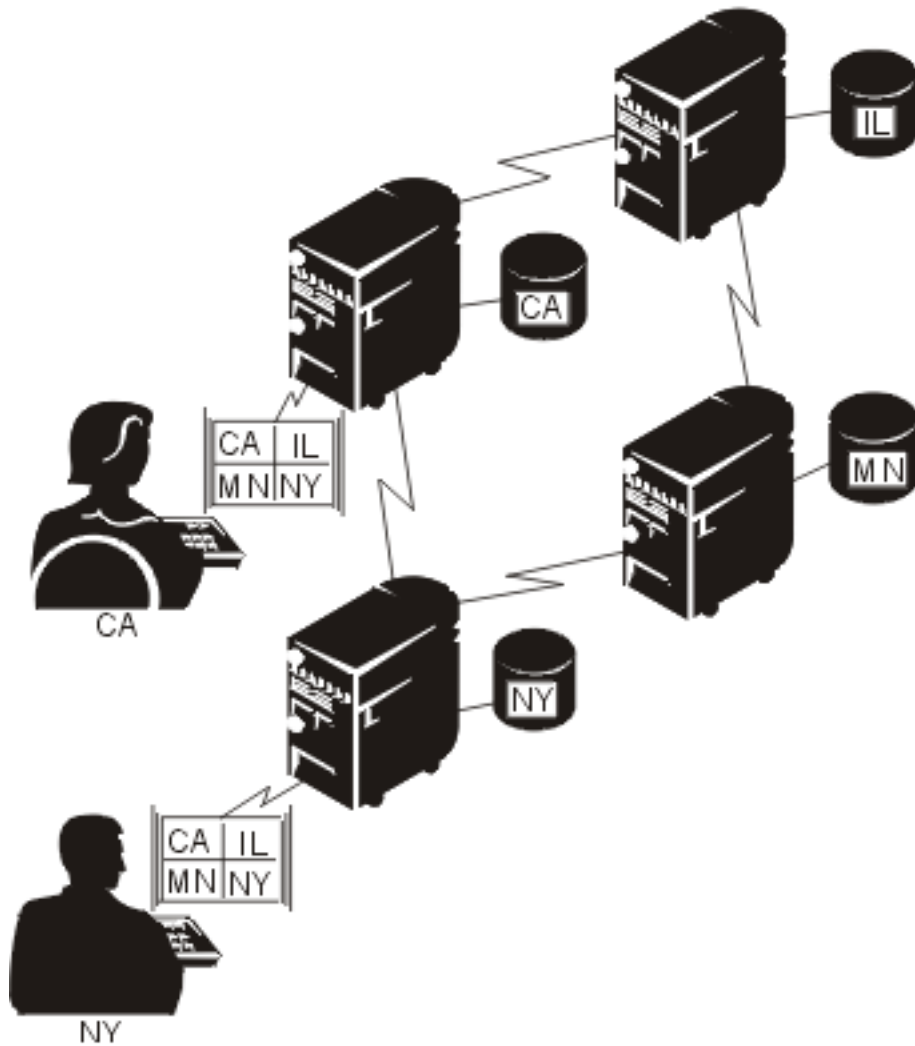


図1. 複数システムでのデータベース・ファイルの分散

この並列処理技法を使用すると、あるサーバーの使用頻度が高くて、ネットワーク内で接続されたその他のシステムのパフォーマンスが低下することはありません。大量のデータがあって、QUERY を実行する必要がある場合は、DB2 マルチ・システムを使用すると、使用可能な最も効率的な方式の 1 つによって、これらの QUERY を実行することができます。ほとんどの場合、QUERY のパフォーマンスが向上します。QUERY がローカル・ファイルに対して実行されるのではなく、いくつかのサーバー間で並列的に実行されるからです。

まだ DB2 マルチ・システムを導入していない場合は、ソフトウェアの導入 に記載されている、追加のライセンス・プログラムを導入するための手順に必要な情報を参照してください。DB2 マルチ・システムを導入するには、OS/400 オペレーティング・システムの導入可能オプションのリストにあるオプション 27 を使用してください。

さらに、以下のセクションを参照してください。

- 3 ページの『DB2 マルチ・システムを使用する利点』
- 3 ページの『DB2 マルチ・システム: 基本的な用語と概念』

DB2 マルチ・システムを使用する利点

DB2 マルチ・システムを使用すると、次のような利点があります。

- QUERY を並列的に実行する (QUERY の各部分を、異なるサーバー上で同時に実行する) ことによって、QUERY のパフォーマンスが向上する。
- すべてのサーバーがすべてのデータにアクセスできるため、データ複製の必要性が少なくなる。
- より大きなデータベース・ファイルを収容することができる。
- アプリケーションで、リモート・データの位置を考える必要がなくなる。
- ファイルを大きくする必要がある場合、さらに多くのシステム間でファイルを分配し直し、新しいシステム上でアプリケーションを変更せずに実行することができる。

DB2 マルチ・システムでは、以前使用していたものと同じ入出力 (I/O) 方式 (GET、PUT、および UPDATE) またはファイル・アクセス方式を使用することができます。追加の、あるいは異なる入出力方式やファイル・アクセス方式は必要ありません。

アプリケーションを変更する必要はありません。どのような接続方式を使用しているても、OptiConnect を使用していない限り、アプリケーションは、作成されたすべての分散ファイルで作動します。OptiConnect では、OptiConnect 制御装置記述を使用する必要があります。OptiConnect の詳細については、資料 *OptiConnect for OS/400* を参照してください。

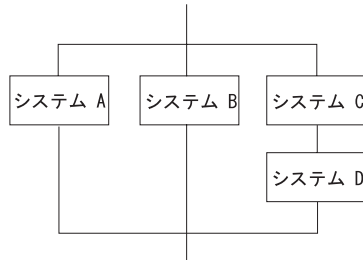
DB2 マルチ・システム: 基本的な用語と概念

分散ファイルとは、複数の iSeries サーバーに分散されたデータベース・ファイルのことをいいます。ここでは、DB2 マルチ・システムによる分散ファイルの作成および使用法を説明するときに使用される、主要概念のいくつかを説明します。各用語と概念については、この後の章で詳しく説明します。

分散ファイルの一部を含む各サーバーは、**ノード**と呼ばれます。各サーバーは、リレーショナル・データベース・ディレクトリーで定義された名前によって識別されます。

1 つまたは複数の分散ファイルを含むシステムのグループは、**ノード・グループ**と呼ばれます。ノード・グループとは、データが配布されるノードのリストが含まれているシステム・オブジェクトのことをいいます。システムは、複数のノード・グループにおける 1 つのノードの場合もあります。

ノード・グループ 1 (システム A、B、C)



ノード・グループ 2 (システム A、B、D)

RBAL3509-1

図2. ノード・グループ

ファイルは、**区分化**によってノード・グループ内のすべてのシステム間に分散されます。

区画番号は、0 から 1023 までの数字です。各区画番号は、ノード・グループ内の 1 つのノードに割り当てられます。各ノードには、多数の区画番号を割り当てることができます。ノードと区画番号の相関は、**区画マップ**に格納されます。区画マップは、ノード・グループ・オブジェクトの一部としても格納されます。区画マップは、ノード・グループを作成するときに用意することができます。そうでない場合は、システムにより省略時のマップが生成されます。

区分化ファイルを使用することによって、区画マップを定義します。区分化ファイルとは、各区画番号のノード番号を定義する物理ファイルのことをいいます。

区分化キーは、分散されているファイル内の 1 つまたは複数のフィールドからなります。区分化キーは、ノード・グループ内のどのノードが、特定の値を持つ行を物理的に含むようにするかを判別するために使用されます。この判別は、**ハッシュ**を使用して行われます。ハッシュとは、オペレーティング・システムの機能の 1 つで、レコードの区分化キーの値をとって、それを区画番号にマップするものです。その区画番号に対応するノードが、レコードの格納に使用されます。

次の例は、2 つのシステムの分散表で、区画番号とノードがどのように表示されるかを示しています。この表には、LASTNAME という区分化キーがあります。

区画番号	ノード
0	SYSA
1	SYSB
2	SYSA
3	SYSB
...	

図3. 区画マップ。区画番号 0 には SYSA が、区画番号 1 には SYSB が、区画番号 2 には SYSA が、区画番号 3 には SYSB が入ります。このパターンが繰り返されます。

区分化キーのハッシュでは、区画番号に対応する番号が判別されます。たとえば、'Andrews' という値を持つレコードは区画番号 1 にハッシュされ、'Anderson' とい

う値を持つレコードは区画番号 2 にハッシュされたとします。図 3 に示された区画マップを参照する場合、区画番号 1 のレコードは **SYSB** に格納されるのに対して、区画番号 2 のレコードは **SYSA** に格納されます。

第 2 章 DB2 マルチ・システムでのノード・グループの概要

一連の iSeries サーバーからデータベース・ファイルを見ることができるようになるためには、このファイルを置きたいシステムのグループ (ノード・グループ) をまず定義する必要があります。

ノード・グループには、2 つから 32 のノードを定義することができます。ノード・グループに定義されるノードの数によって、データベース・ファイルが作成されるシステムの数が決まります。ローカル・システムは、ノード・グループに指定されたシステムのうち 1 つでなければなりません。システムがノード・グループを作成する場合、各ノードには、1 から始まる番号が割り当てられます。

ノード・グループの処理については、次のトピックを参照してください。

- ・ 『DB2 マルチ・システムでのノード・グループの仕組み』
- ・ 8 ページの『DB2 マルチ・システムでノード・グループ・コマンドを使用する前に完了しておくべきタスク』
- ・ 9 ページの『DB2 マルチ・システムでの CRTNODGRP コマンドを使用したノード・グループの作成』
- ・ 11 ページの『DB2 マルチ・システムでの DSPNODGRP コマンドを使用したノード・グループの表示』
- ・ 13 ページの『DB2 マルチ・システムでの CHGNODGRPA コマンドを使用したノード・グループの変更』
- ・ 14 ページの『DB2 マルチ・システムでの DLTNODGRP コマンドを使用したノード・グループの削除』

DB2 マルチ・システムでのノード・グループの仕組み

ノード・グループとは、作成されたシステム上に格納されたシステム・オブジェクト (*NODGRP) のことをいいます。ノード・グループは、分散オブジェクトではありません。*NODGRP システム・オブジェクトには、データ・ファイル内のデータがどのように区分化 (分配) されるかに関する情報とともに、グループ内のシステムに関するすべての情報が含まれます。省略時の区分化では、各システム (ノード) がデータの等しい共用分を受け取ります。


区分化は、ハッシュ・アルゴリズムを使用して処理されます。ノード・グループが作成されると、0 から 1023 の範囲内の区画番号が、そのノード・グループに対応付けられます。省略時の区分化では、等しい数の区画が、ノード・グループ内の各ノードに割り当てられます。データがファイルに追加されると、区分化キー内のデータがハッシュされ、区画番号が作成されます。データのレコード全体はハッシュされません。区分化キー内のデータだけが、ハッシュ・アルゴリズムを通してハッシュされます。結果の区画番号に対応するノードは、データのレコードが物理的に存在する場所です。したがって、省略時の区分化では、十分なデータ・レコードと広範囲の値がある場合、各ノードはデータの等しい共用分を格納します。

各ノードでデータの等しい共用分を受け取りたくない場合、またはデータの特定部分を所有するシステムを指定したい場合は、ノード・グループの作成 (CRTNODGRP) コマンドに区分ファイル (PTNFILE) パラメーターを使用してカスタム区分化構造を指定するか、またはノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して区分化構造をあとで変更することによって、データの区分化の方法を変更することができます。PTNFILE パラメーターを使用すると、ノード・グループ内の各区画にノード番号を設定することができます。すなわち、PTNFILE パラメーターを使用すると、ノード・グループ内のシステムでのデータの区分化方法を調整することができます。(PTNFILE パラメーターは、9 ページの『DB2 マルチ・システムでの CRTNODGRP コマンドを使用したノード・グループの作成』の例で使用されます。) 区分化の詳細については、25 ページの『DB2 マルチ・システムでの区分化』を参照してください。

ノード・グループはシステム・オブジェクトであるため、オブジェクトの保管 (SAVOBJ) コマンドとオブジェクトの復元 (RSTOBJ) コマンドを使用して保管することができます。ノード・グループ・オブジェクトは、作成されたシステムか、またはノード・グループ内のいずれかのシステムに復元することができます。ノード・グループ・オブジェクトがノード・グループ内にないシステムに復元される場合、オブジェクトは使用できなくなります。

DB2 マルチ・システムでノード・グループ・コマンドを使用する前に完了しておくべきタスク

ノード・グループの作成 (CRTNODGRP) コマンドまたはノード・グループ・コマンドのいずれかを使用するにあたっては、使用している分散リレーショナル・データベース・ネットワークが正しく設定されていることを確認する必要があります。これが新しい分散リレーショナル・データベース・ネットワークである場合は、ネッ

トワークの確立方法について、『分散データベース・プログラミング 』を参照してください。

ネットワーク内の 1 つのシステムが、ローカル (*LOCAL) システムとして定義されていることを確認する必要があります。RDB (リレーショナル・データベース) ディレクトリー項目の処理 (WRKRDBDIRE) コマンドを使用して、項目に関する詳細を表示してください。ローカル・システムが定義されていない場合は、RDB ディレクトリー項目の追加 (ADDRDBDIRE) コマンドのリモート・ロケーション名 (RMTLOCNAME) パラメーターに *LOCAL を指定することによって、定義することができます。次に例を示します。

```
ADDRDBDIRE RDB(MP000) RMTLOCNAME(*LOCAL) TEXT ('New York')
```

New York の MP000 という名前の iSeries サーバーは、リレーショナル・データベース・ディレクトリーにローカル・システムとして定義されます。ネットワーク構成内の本システムに指定されたシステム名またはローカル・ロケーション名として iSeries サーバーに定義できるローカル・リレーショナル・データベースは 1 つだけです。これは、特にネットワークが複雑な場合、データベース名を識別して、それを分散リレーショナル・データベース内の特定のシステムに関連付けるのに役立ちます。

DB2 マルチ・システムが、定義したノード・グループ内の iSeries サーバーにファイルを正しく分配するためには、リモート・データベース (RDB) 名を、そのノード・グループ内のすべてのノード (システム) で整合性のあるものにする必要があります。

たとえば、ノード・グループに 3 つのシステムを用意する場合、各システムは少なくとも 3 つの項目を RDB ディレクトリーに持っている必要があります。各システムでは、3 つの名前がすべて同じでなければなりません。3 つのシステムのそれぞれにおいて、ローカル・システムを示す 1 つの項目が存在します。これは、*LOCAL と識別されます。他の 2 つの項目には、適切なリモート・ロケーション情報が含まれます。

DB2 マルチ・システムでの CRTNODGRP コマンドを使用したノード・グループの作成

ここでは、2 つの CL コマンドの例を使用して、ノード・グループの作成 (CRTNODGRP) コマンドによってノード・グループを作成する方法を示します。

次の例では、省略時の区分化 (システムでの等しい区分化) を使用するノード・グループを作成しています。

```
CRTNODGRP NODGRP(LIB1/GROUP1) RDB(SYSTEMA SYSTEMB SYSTEMC SYSTEMD)
          TEXT('Node group for test files')
```

この例では、コマンドは 4 つのノードを含むノード・グループを作成しています。各ノードには、RDB 項目 (ADDRDBDIRE コマンドを使用してリレーショナル・データベース・ディレクトリーに前に追加されたもの) を定義する必要があり、1 つのノードをローカル (*LOCAL) として定義する必要があることに注意してください。

区分化属性は省略時解釈として、各ノード番号に区画の 4 分の 1 が割り当てられることとなります。このノード・グループを、物理ファイルの作成 (CRTPF) コマンドの NODGRP パラメーターで使用すれば、分散ファイルを作成することができます。分散ファイルの詳細については、『第 3 章 DB2 マルチ・システムでの分散ファイルの作成』を参照してください。

注: データには違いがあるため、各システムには必ずしもデータの 4 分の 1 が区分化されるわけではありません。たとえば、区分化キーとして市外局番フィールドを選択することによって、電話番号の市外局番をもとにデータを区分化するとします。この場合、業務の 60% が市外局番 507 であるとします。市外局番 507 の区画を持つシステムは、データの 25% ではなく、60% を受け取ります。

次の例では、指定の区分化を使用するノード・グループが、ファイルの区分化 (PTNFILE) パラメーターを使用して作成されます。

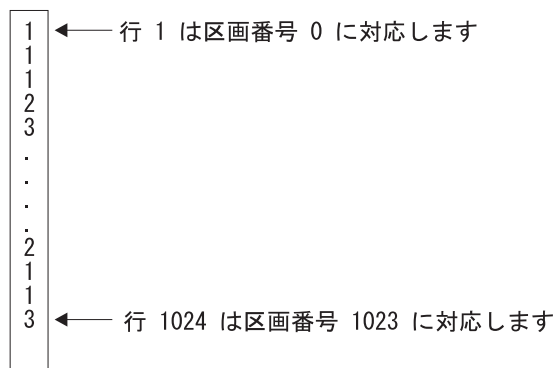
```
CRTNODGRP NODGRP(LIB1/GROUP2) RDB(SYSTEMA SYSTEMB SYSTEMC)
          PTNFILE(LIB1/PTN1)
          TEXT('Partition most of the data to SYSTEMA')
```

この例では、コマンドは 3 つのノード (SYSTEMA、SYSTEMB、および SYSTEMC) を含むノード・グループを作成しています。区分化属性は、PTN1 と呼

ばれるファイルからとられています。このファイルを設定すると、さらに多い割合のレコードを特定のシステム上に置くことができます。

この例の PTN1 は区分化ファイルです。このファイルは分散ファイルではなく、通常のローカル物理ファイルであり、カスタム区分化構造を設定するために使用することができます。区分化ファイルには、2 進数フィールドが 1 つ必要です。区分化ファイルには 1024 のレコードが必要であり、各レコードには有効なノード番号が含まれます。

2 バイトの 2 進数フィールド



RBAL3508-0

図4. 区分化ファイル PTNFILE の内容の例

ノード・グループに 3 つのノードがある場合、区分化ファイルのレコードすべてに 1、2、または 3 という番号が必要です。ノード番号は、RDB 名がノード・グループの作成 (CRTNODGRP) コマンドに指定された順序で割り当てられます。区分化ファイルにそのノード番号を含むレコードをさらに多く持たせることによって、より多い割合のデータを特定ノードに与えることができます。これは、各システムに物理的に存在するデータの量に関して、区分化をカスタマイズする方式です。特定のノードに存在する特定の値に関して区分化をカスタマイズするには、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用してください。詳細については、13 ページの『DB2 マルチ・システムでの CHGNODGRPA コマンドを使用したノード・グループの変更』を参照してください。

ノード・グループ情報は分散ファイルに格納されるため、ファイルは、ノード・グループ内の変更、またはノード・グループに含まれる RDB ディレクトリー項目の変更に対してすぐには対応しないことに注意する必要があります。ノード・グループおよび RDB ディレクトリー項目に変更を加えることはできますが、CHGPF コマンドを使用して変更されるノード・グループを指定するまで、ファイルの動作は変更されません。

他の概念として、**可視ノード**というものがあります。ノード・グループ内の可視ノードには、ファイル・オブジェクト (ファイルをいくつかのノードに分散できるようにするメカニズムの一部) が含まれますが、データは含まれません。可視ノードは、常に現行レベルのファイル・オブジェクトを保存します。可視ノードにはデータが格納されません。反対に、ノード (**データ・ノード**と呼ばれることもある) にはデータが含まれます。可視ノードをノード・グループで使用する例として、営業部


長が使用している iSeries サーバーをノード・グループの一部にします。これらの部長は、定期的には QUERY を実行しないで、特定の QUERY を実行したい場合がよくあります。それらのサーバーから、部長は各自の QUERY を実行して、リアルタイム・データにアクセスし、その QUERY の結果を受け取ることができます。したがって、データがそれらのサーバーにまったく格納されていない場合でも、それらのシステムは可視ノードであるため、部長は必要に応じていつでも QUERY を実行することができます。

可視ノードとしてノードを指定するには、PTNFILE パラメーターをノード・グループの作成 (CRTNODGRP) コマンドに使用する必要があります。区分化ファイルに特定のノード番号のレコードが含まれない場合、そのノードは可視ノードです。

DB2 マルチ・システムでの DSPNODGRP コマンドを使用したノード・グループの表示

ノード・グループの表示 (DSPNODGRP) コマンドは、ノード・グループ内のノード (システム) を表示します。また、ノード・グループの区分化構造も表示されます (区分化については、25 ページの『DB2 マルチ・システムでの区分化』で説明します)。

次の例は、GROUP1 という名前のノード・グループを、そのノード・グループに関連する区分化構造とともに表示する方法を示しています。この情報は、ワークステーションに表示されます。DSPNODGRP コマンドの詳細については、Information

Center のトピック『制御言語 』を参照してください。

```
DSPNODGRP NODGRP(LIB1/GROUP1)
```

ノード・グループ名を指定して DSPNODGRP コマンドを実行すると、ノード・グループの表示画面が示されます。この画面には、システム名 (リレーショナル・データベースの列に示されたもの) と、そのシステムに割り当てられたノード番号が示されます。これは、どのシステムがどのノード番号を持つかを定めるための直接的な方式です。

ノード・グループの表示	
ノード・グループ:	GROUP1 ライブラリー: LIB1
リレーショナル・データベース	ノード番号
SYSTEMA	1
SYSTEMB	2
SYSTEMC	3
終り	
F3=終了 F11=区画設定データ F12=取り消し F17=最上部 F18=最下部	

図5. ノード・グループの表示: データベースとノード番号の相関

各区画番号に割り当てられたノード番号を見るには、F11 (区画設定データ) をノード・グループの表示画面で使用します。次の画面は、各区画番号に割り当てられたノード番号を示しています。システムとノード番号間での (またはノード番号からシステムへの) マッピングは、DSPNODGRP コマンドを使用して簡単に実行することができます。

ノード・グループの表示	
ノード・グループ:	GROUP1 ライブラリー: LIB1
区画番号	ノード番号
0	1
1	2
2	3
3	1
4	2
5	3
6	1
7	2
8	3
9	1
10	2
11	3
12	1
13	2
14	3
続く...	
F3=終了 F11=ノード・データ F12=取り消し F17=最上部 F18=最下部	

図6. ノード・グループの表示: 区画番号とノード番号の相関

次の例は、GROUP2 というノード・グループ内のシステムのリストを、関連する区分化構造とともに印刷するものです。

```
DSPNODGRP NODGRP(LIB1/GROUP2) OUTPUT(*PRINT)
```

DB2 マルチ・システムでの CHGNODGRPA コマンドを使用したノード・グループの変更

ノード・グループ属性の変更 (CHGNODGRPA) コマンドは、ノード・グループのデータ区分化属性を変更します。ノード・グループには、1024 の区画を持つ表が含まれます。各区画にはノード番号があります。ノード番号は、ノード・グループの作成時に割り当てられ、ノード・グループの作成 (CRTNODGRP) コマンドの RDB パラメーターに指定されたりレシヨナル・データベースに対応します。有効なノード番号値と、ノード番号とレシヨナル・データベース名の相関を見るには、ノード・グループの表示 (DSPNODGRP) コマンドを使用します。

CHGNODGRPA コマンドは、指定のノード・グループを使用して作成された既存の分散ファイルに影響を与えません。変更されるノード・グループを使用するには、その変更されるノード・グループを、新しいファイルの作成時、または物理ファイルの変更 (CHGPF) コマンドに指定する必要があります。CHGNODGRPA コマンド

の詳細については、Information Center のトピック『制御言語 』を参照してください。

この最初の例には、ライブラリー LIB1 の GROUP1 というノード・グループの区分化属性を変更する方法が示されています。

```
CHGNODGRPA NODGRP(LIB1/GROUP1) PTNNBR(1019)
              NODNBR(2)
```

この例では、区画番号 1019 が指定され、1019 にハッシュされるレコードはすべて、ノード番号 2 に書き込まれます。これにより、特定の区画番号をノード・グループ内の特定のノードに指定することができます。

2 番目の例では、GROUP2 という名前のノード・グループの区分化属性が変更されています。(GROUP2 は、ライブラリー探索リスト *LIBL を使用して検出されます。) 比較データ値 (CMPDTA) パラメーターに指定された値がハッシュされて、その結果の区画番号は、既存のノード番号からノード番号 3 に変更されます。(ハッシュと区分化については、25 ページの『DB2 マルチ・システムでの区分化』に説明があります。)

```
CHGNODGRPA NODGRP(GROUP2) CMPDTA('CHICAGO')
              NODNBR(3)
```

このノード・グループを使用して作成されたファイルと、文字フィールドからなる区分化キーを持つファイルはすべて、'CHICAGO' を含むレコードをノード番号 3 の区分化キーに格納します。区分化キーに複数のフィールドを持つファイルを可能にするために、最大 300 の値をデータ比較 (CMPDTA) パラメーターに指定することができます。

CMPDTA パラメーターに値を入力する場合は、文字データの大文字小文字を区別するように注意が必要です。すなわち、'Chicago' と 'CHICAGO' では同じ区画番号になりません。数値データは、単なる数字として入力する必要があります。小数点、先行ゼロ、または後続ゼロは使用しないでください。

すべての値は、区画番号を獲得するためにハッシュされます。これらは次に、ノード番号 (NODNBR) パラメーターに指定されたノード番号に対応付けられます。完了メッセージ CPC3207 のテキストは、変更された区画番号を示します。

CHGNODGRPA コマンドを何度も、異なる多数の値に対して出すと、同じ区画番号を 2 回変更する確率が高くなることに注意してください。このことが起こると、最新の変更において指定されたノード番号が、ノード・グループに対して有効になります。

データの分散をカスタマイズする方法については、28 ページの『DB2 マルチ・システムでのデータ分配のカスタマイズ』を参照してください。

DB2 マルチ・システムでの DLTNODGRP コマンドを使用したノード・グループの削除

ノード・グループの削除 (DLTNODGRP) コマンドは、以前に作成されたノード・グループを削除します。このコマンドは、ノード・グループを使用して作成されたファイルには影響を与えません。DLTNODGRP コマンドの詳細については、

Information Center のトピック『制御言語 』を参照してください。

次の例は、GROUP1 という名前のノード・グループを削除する方法を示しています。このノード・グループによって作成されたファイルはいずれも影響を受けません。

```
DLTNODGRP NODGRP(LIB1/GROUP1)
```

ノード・グループの削除によって、そのノード・グループを使用して作成されたファイルは影響を受けませんが、ノード・グループの使用後にそれらを削除することはお勧めできません。ノード・グループを削除すると、DSPNODGRP コマンドを使用して、ノードと区分化構造を表示することができなくなるためです。ただし、ファイル記述の表示 (DSPFD) コマンドに TYPE(*NODGRP) を指定して使用すると、特定のファイルに関連するノード・グループを表示することができます。

第 3 章 DB2 マルチ・システムでの分散ファイルの作成

分散ファイルとは、複数の iSeries サーバー間にスプレッドされたデータベース・ファイルのことをいいます。分散ファイルは更新でき、SQL、QUERY ツール、および物理ファイル・メンバーの表示 (DSPPFM) コマンドなどの方法を使用してアクセスすることができます。データベース操作の場合、分散ファイルは、ほとんどの部分でローカル・ファイルのように処理されます。分散ファイルに対する CL コマンドの変化については、20 ページの『DB2 マルチ・システムでの分散ファイルに対する CL コマンドの実行』を参照してください。

データベース・ファイルを分散ファイルとして作成するには、物理ファイルの作成 (CRTPF) コマンドまたは SQL CREATE TABLE ステートメントのいずれかを使用することができます。この章では、両方の方法についてさらに詳しく説明します。CRTPF コマンドには 2 つのパラメーターがあります。これらはノード・グループ (NODGRP) と区分化キー (PTNKEY) で、ファイルを分散ファイルとして作成します。分散データベース・ファイルは、CRTPF コマンドの実行時に指定した名前とライブラリーを持つオブジェクト (PF 属性を持つ *FILE) として表示されます。

既存の非分散データベース物理ファイルを分散ファイルに変更したい場合は、物理ファイルの変更 (CHGPF) コマンドを使用して、これを行うことができます。CHGPF コマンドでは、NODGRP および PTNKEY の各パラメーターを使用して、分散ファイルに変更を加えることができます。CHGPF コマンドを使用すると、NODGRP および PTNKEY の各パラメーターの値を指定することによって、既存の分散データベース・ファイルのデータ区分化属性を変更することもできます。これらのパラメーターに値を指定すると、ノード・グループ内の区分化表に従って、データが再分配されます。

注: 分散ファイルに変更される物理ファイルに基づく論理ファイルもすべて、分散ファイルになります。大きなファイルまたは大規模ネットワークの場合、データの再分配に長い時間がかかる可能性があるため、頻繁に行わないようにしてください。

分散ファイルの処理については、次のトピックを参照してください。

- 『DB2 マルチ・システムでの分散物理ファイルの作成』
- 18 ページの『DB2 マルチ・システムでの分散ファイルの使用』
- 25 ページの『DB2 マルチ・システムでの区分化』
- 28 ページの『DB2 マルチ・システムでのデータ分配のカスタマイズ』

DB2 マルチ・システムでの分散物理ファイルの作成

ここでは、CRTPF コマンドまたは SQL CREATE TABLE ステートメントを使用して、分散物理ファイルを作成する方法を説明します。

区分化ファイルを作成したい場合は、次のパラメーターを物理ファイルの作成 (CRTPF) コマンドに指定する必要があります。

- NODGRP パラメーターのノード・グループ名

- 区分化キーとして使用される 1 つまたは複数のフィールド (PTNKEY パラメーターを使用)

区分化キーは、データの各レコードが物理的に存在する場所 (ノード) を判別します。区分化キーは、物理ファイルの作成 (CRTPF) コマンドの実行時、または SQL CREATE TABLE ステートメントの実行時に指定します。各レコードの区分化キーを構成するフィールドの値は、HASH (ハッシュ) アルゴリズムによって処理されて、レコードの所在を決めます。

単一フィールドの区分化キーがある場合は、そのフィールド内で同じ値を持つレコードすべてが、同じシステム上に存在します。

分散物理ファイルを作成したい場合は、ユーザー・プロファイルがノード・グループ内のすべてのノードに存在しなければなりません。また、ユーザー・プロファイルに、すべてのノードで分散ファイルを作成するために必要な権限がなければなりません。特定のライブラリーに分散ファイルを作成する必要がある場合は、そのライブラリーがノード・グループ内のすべてのノードに存在しなければなりません。また、ユーザー・プロファイルに、これらのライブラリーにファイルを作成するために必要な権限がなければなりません。これらの要素があてはまらない場合、ファイルは作成されません。

システムの構成方法によっては、リモート・システムで使用しているユーザー・プロファイルに影響することがあります。ユーザー・プロファイルをリモート・システムで確実に使用されるようにするには、そのシステムをセキュア・ロケーションとして構成してください。システムがセキュアなロケーションとして構成されているかどうかを判別するには、構成リストの処理 (WRKCFGL) コマンドを使用します。詳しくは、Information Center のトピック『分散データベース・プログラミング



』を参照してください。

次の例は、区分化され (NODGRP パラメーターを使用して指定され)、社員番号 (EMPNUM) フィールドに単一の区分化キーを持つ PAYROLL という名前の物理ファイルを作成する方法を示しています。

```
CRTPF FILE(PRODLIB/PAYROLL) SCRFILE(PRODLIB/DDS) SRCMBR(PAYROLL)
      NODGRP(PRODLIB/PRODGROUP) PTNKEY(EMPNUM)
```

CRTPF コマンドを実行すると、分散物理ファイルが作成されて、分散ファイルに関連するローカル・データを保持します。CRTPF コマンドは、ノード・グループに指定されたすべてのリモート・システムでも物理ファイルを作成します。

物理ファイルの所有権とすべてのシステムでの共通権限は、整合性がとれています。この整合性には、CRTPF コマンドの AUT パラメーターに指定された権限もすべて含まれています。

SQL CREATE TABLE ステートメントは、ノード・グループと区分化キーを指定するために使用することもできます。次の例では、PAYROLL という SQL 表が作成されます。この例では、IN ノード・グループ名 文節と PARTITIONING KEY 文節を使用しています。


```
CREATE TABLE PRODLIB/PAYROLL
(EMPNUM INT, EMPLNAME CHAR(12), EMPFNAME CHAR (12))
IN PRODLIB/PRODGROUP
PARTITIONING KEY (EMPNUM)
```

PARTITIONING KEY 文節が指定されていない場合、1 次キーの最初の列が定義されていれば、それが最初の区分化キーとして使用されます。1 次キーが定義されていない場合は、日付、時刻、タイム・スタンプ、または浮動小数点のデータ型を持たない表に定義された最初の列が、区分化キーとして使用されます。

ファイルが区分化されているかどうかを表示するには、ファイル記述の表示 (DSPFD) コマンドを使用します。ファイルが区分化されている場合、DSPFD コマンドはノード・グループの名前およびファイル・オブジェクトにあるノード・グループ (区画マップ全体を含む) の詳細を示し、区分化キーのフィールドをリストします。

DB2 マルチ・システムで分散ファイルを使用する際に知っておかなければならない制約事項のリストについては、『DB2 マルチ・システムにおいて分散ファイルを作成または処理するときの制限』を参照してください。

DB2 マルチ・システムにおいて分散ファイルを作成または処理するときの制限

分散ファイルを作成または処理する場合は、次の制限に注意する必要があります。

- 先変更先出し (FCFO) アクセス・パスは使用することができません。これは、複数のノード間でアクセス・パスが区分化されるためです。
- 分散ファイルが持つことができるメンバーの数は最大 1 つです。
- 分散ファイルは、一時ライブラリー (QTEMP) では認められません。
- 区分化キー内のデータの更新能力は限定されています。一般に、区分化キーを選択する場合は、値が更新されないフィールドを選択する必要があります。区分化キーの更新は、更新によってレコードが異なるノードに区分化されない限り許されます。
- 日付、時刻、タイム・スタンプ、または浮動小数点の各数字フィールドは、区分化キーには使用することができません。
- ソース物理ファイルはサポートされていません。
- 外部記述ファイルは分散ファイルでサポートされています。プログラム記述ファイルはサポートされていません。
- アクセス・パスが固有の場合、区分化キーは、固有キー・アクセス・パスの一部でなければなりません。
- 制約はサポートされています。参照制約は、親ファイルおよび外部キー・ファイルの両方のノード・グループが同じであり、区分化キーのすべてのフィールドが制約に含まれている場合にのみサポートされます。区分化キーは、制約フィールドの一部でなければなりません。また、固有の 1 次制約の場合、アクセス・パスが固有であれば、区分化キーは固有キー・アクセス・パスのサブセットでなければなりません。
- CRTPF コマンドでは、システム・パラメーターには *LCL という値 (CRTPF SYSTEM(*LCL)) が指定されていなければなりません。SYSTEM(*RMT) は許可されません。

- 論理ファイルが分散ファイルに対して作成されると、その論理ファイルも分散ファイルになります。すなわち、指定のノードにある物理ファイルの一部に対してだけローカル論理ファイルを作成することはできないということです。SQL ビューは、ビューが結合であり、基礎となる物理ファイルが同じノード・グループを持たない場合、この例外となります。この場合、ビューはローカル・システムでのみ作成されます。このビューが分配されない場合でも、そのビューを照会すると、そのビューが作成されたノードだけではなく、すべてのノードからデータが検索されます。

結合ファイルは、SQL を使用したときのみ作成できます。

DDS 作成論理ファイルには、基礎ファイルが 1 つだけ認められています。

- コード化文字セット識別コード (CCSID) と分類順序 (SRTSEQ) 表は、元のシステムから変換されます。
- 可変長レコード (VLR) 処理はサポートされていません。これは、可変長フィールドが分散ファイルでサポートされていないということではありません。この制限は、ファイルのオープン時に VLR 処理を要求する言語とアプリケーションだけに当てはまります。
- ファイル終わり遅延 (EOFDLY) 処理はサポートされていません。
- データ・ファイル・ユーティリティ (DFU) は、分散ファイルに対しては機能しません。これは、DFU が相対レコード番号処理を使用してレコードにアクセスするためです。
- 分散ファイルを、独立補助記憶域プール (IASP) に存在するライブラリーの中に作成することはできません。

DB2 マルチ・システムでの分散ファイルの使用

ファイルを作成すると、そのデータは確実に区分化されて、ファイルは並行レベルで維持されます。ファイルの作成時に自動的に起こる活動には、次のものがあります。

- ファイルで作成されるすべての索引は、すべてのノードで作成されます。
- 権限の変更はすべてのノードに送信されます。
- システムによって、ファイルの移動とそのライブラリーの名前変更が防止されません。
- ファイル自体の名前が変更されると、その新しい名前がすべてのノードに反映されます。
- オブジェクトの割り振り (ALCOBJ)、物理ファイル・メンバーの再編成 (RGZPFM)、およびジャーナル物理ファイルの開始 (STRJRNPF) などのいくつかのコマンドは、ここでファイルのすべての部分に影響を与えます。これにより、区分化ファイルを処理するときに、ローカル・ファイルという概念を維持することができます。これらの CL コマンドの詳細なリストは、22 ページの『CL コマンド: DB2 マルチ・システムでの分散ファイルのすべての部分に影響を与えるもの』を参照してください。

オブジェクト割り振り (ALCOBJ) コマンドは、ノード・グループ内のどのノードからでも出すことができます。これにより、すべての部分がロックされて、ロー

カル・ファイルを割り振るときに付与されるのと同じ保全性が確保されます。これらのアクションはすべてシステムによって処理されるため、各ノードにコマンドを入力する必要がなくなります。

物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドの場合、ジャーナル処理は各システムで開始されます。したがって、各システムは独自のジャーナルとジャーナル・レシーバーを備えている必要があります。各システムには独自のジャーナル項目があります。このジャーナル項目を使用する回復は、各システムに対して個々に実行されなければなりません。ジャーナル処理を開始、終了するコマンドは、ノード・グループ内のすべてのシステムに同時に影響を与えます。追加情報については、23 ページの『DB2 マルチ・システムでのジャーナル処理の考慮事項』を参照してください。

- オブジェクトのダンプ (DMPOBJ)、オブジェクトの保管 (SAVOBJ)、およびオブジェクト・ロックの処理 (WRKOBJLCK) などのいくつかのコマンドは、そのコマンドが出されたシステム上のファイルの部分にのみ影響を与えます。これらの CL コマンドの詳細なリストは、21 ページの『CL コマンド: DB2 マルチ・システムでの分散ファイルのローカル部分にのみ影響を与えるもの』を参照してください。

あるファイルが分散ファイルとして作成された場合、そのファイルをオープンすると、すべてのリモート・システムへの接続が作成されるとともに、実際にはファイルのローカル部分がオープンされます。作成されたファイルは、ノード・グループ内のすべてのシステムからアクセスすることができます。システムは、ファイル入出力タスク (たとえば、GETS、PUT、および UPDATES) を完了するためにどのノードとレコードを使用する必要があるかも判断します。この活動のいずれかを物理的に指示したり、指定する必要はありません。

分散リレーショナル・データベース・アーキテクチャー (DRDA[®]) の要求と分散データ管理機能 (DDM) の要求は、分散ファイルをターゲットにすることができます。DRDA または DDM を使用してリモート・システム上のデータベース・ファイルにアクセスする、以前に配布されたアプリケーションは、データベース・ファイルが分散ファイルに変更された場合でも、引き続き機能することができます。

レコードの到着順が、ローカル・データベース・ファイルと分散データベース・ファイルでは異なることに注意する必要があります。

分散ファイルはシステムにまたがって物理的に分配されるため、レコードの到着順または相対レコード番号に依存することはできません。ローカル・データベース・ファイルでは、レコードは順番に処理されます。たとえば、ローカル・データベース・ファイルにデータを挿入する場合、そのデータは、最初のレコードから最後のレコードへという順序で挿入されます。すべてのレコードが順番に挿入されます。各ノードのレコードは、レコードがローカル・ファイルに挿入される場合と同じ方法で挿入されます。

データがローカル・データベース・ファイルから読み取られると、そのデータは、最初のレコードから最後のレコードへと読み取られます。これは、分散データベース・ファイルの場合にはあてはまりません。分散データベース・ファイルでは、データベースは最初のノードの最初のレコード (最初のレコードから最後のレコードへ)、次に 2 番目のノードという順序で読み取られます。たとえば、レコード 27

に対する読み取りは、単一レコードに対する読み取りではなくなります。分散ファイルでは、ノード・グループ内の各ノードに、それぞれ別のレコード 27 が含まれています。

CL コマンドの実行について詳しくは、『DB2 マルチ・システムでの分散ファイルに対する CL コマンドの実行』を参照してください。

DB2 マルチ・システムでの分散ファイルに対する CL コマンドの実行

分散ファイルのシステム・オブジェクト・タイプは *FILE であるため、物理ファイルにアクセスする CL コマンドの多くは、分散ファイルに対して実行することができます。ただし、一部の CL コマンドは、非分散ファイルではなく分散ファイルに対して出されると、動作が変わります。次のセクションでは、分散ファイルに対する CL コマンドの動作について説明します。CL コマンドの詳細については、

Information Center のトピック『制御言語 』を参照してください。

DB2 マルチ・システムで使用できる CL コマンドの特定のタイプについて詳しくは、次のトピックを参照してください。

- 『CL コマンド: DB2 マルチ・システムでの分散ファイルに対して実行できないもの』
- 21 ページの『CL コマンド: DB2 マルチ・システムでの分散ファイルのローカル部分にのみ影響を与えるもの』
- 22 ページの『CL コマンド: DB2 マルチ・システムでの分散ファイルのすべての部分に影響を与えるもの』

CL コマンド: DB2 マルチ・システムでの分散ファイルに対して実行できないもの

次の CL コマンドまたは特定のパラメーターは、分散ファイルに対して実行することができません。

- 論理ファイル・メンバーの変更 (CHGLFM) の SHARE パラメーター
- 物理ファイル・メンバーの変更 (CHGPFM) の SHARE パラメーター
- 複写オブジェクトの作成 (CRTDUPOBJ)
- 物理ファイル・メンバーの初期化 (INZPFM)
- オブジェクトの移動 (MOV OBJ)
- データベース・ファイルの位置決め (POSDBF)
- メンバーの除去 (RMV M)
- 分散ファイルを含むライブラリーでのライブラリーの名前変更 (RNMLIB)
- メンバーの名前変更 (RNMM)
- 統合ファイル・システム (IFS) コマンド、COPY

CL コマンド: DB2 マルチ・システムでの分散ファイルのローカル部分にのみ影響を与えるもの

次の CL コマンドを実行すると、ローカル・システム上にある分散ファイルの部分にのみ影響が及びます (ローカル・ファイルとは、コマンドが実行されるシステムです)。

- ジャーナル処理済み変更適用 (APYJRNCHG)。このコマンドの追加情報については、23 ページの『DB2 マルチ・システムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクト記述の表示 (DSPOBJD)
- オブジェクトのダンプ (DMPOBJ)
- ジャーナル・アクセス・パスの終了 (ENDJRNP)
- ジャーナル処理済み変更の除去 (RMVJRNCHG)。このコマンドの追加情報については、23 ページの『DB2 マルチ・システムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクトの復元 (RSTOBJ)
- オブジェクトの保管 (SAVOBJ)
- ジャーナル・アクセス・パスの開始 (STRJRNAP)

リモート・コマンド投入 (SBMRMTCMD) コマンドを使用すると、分散ファイルに関連するすべてのリモート・システムに対して、任意の CL コマンドを出すことができます。CL コマンドをローカル・システムに出してから、分散ファイルで SBMRMTCMD コマンドを介して同じコマンドを出すと、分散ファイルのすべてのシステムに対して、CL コマンドを実行することができます。分散ファイルのすべての部分に自動的に実行される CL コマンドには、このことを行う必要はありません。詳細については、22 ページの『CL コマンド: DB2 マルチ・システムでの分散ファイルのすべての部分に影響を与えるもの』を参照してください。

ファイル記述の表示 (DSPFD) コマンドを使用すると、分散ファイルのノード・グループ情報を表示することができます。DSPFD コマンドは、ノード・グループの名前、区分化キーのフィールド、およびノード・グループの完全記述を示します。この情報を表示するには、*ALL または *NODGRP を DSPFD コマンドの TYPE パラメーターに指定する必要があります。

物理ファイル・メンバーの表示 (DSPPFM) コマンドを使用すると、分散ファイルのローカル・レコードを表示することができます。ただし、ローカル・データだけでなくリモート・データを表示したい場合は、コマンドの開始レコード (FROMRCD) パラメーターに *ALLDATA を指定する必要があります。

オブジェクトの保管 (SAVOBJ) コマンド、またはオブジェクトの復元 (RSTOBJ) コマンドを分散ファイルに使用する場合は、分散ファイルの各部分を別々に保管、復元する必要があります。ファイルの部分は、分散ファイルの一部として維持される場合、保管されたときのシステムにのみ復元することができます。必要であれば、オブジェクトの割り振り (ALLOBJ) コマンドを使用して、ファイルのすべての部分のロックを獲得して、保管処理中にファイルに更新が加えられるのを防止することができます。

システムは、次のことがあてはまる場合、ファイルの復元時に、すべての論理ファイルを自動的に配布します。

- 論理ファイルが、非分散ファイルとして保管された。
- 論理ファイルが、基礎ファイルの配布時にシステムに復元される。

ファイルの保管部分を使用して、ローカル・ファイルを作成することもできます。これには、ファイルの部分を別のライブラリー、または分散ファイルの作成時に使用されたノード・グループになかったシステムのいずれかに、復元する必要があります。分散ファイル内のすべてのレコードをローカル・ファイルに収容するには、ファイルの各部分を同じシステムに復元してから、1つの集合ファイルにレコードをコピーする必要があります。集合ファイルにレコードをコピーするには、ファイルのコピー (CPYF) コマンドを使用してください。

CL コマンド: DB2 マルチ・システムでの分散ファイルのすべての部分に影響を与えるもの

次の CL コマンドを実行すると、分散ファイルのすべての部分に影響が及びます。これらのコマンドをシステム上で実行すると、ノード・グループ内のすべてのノードで、コマンドが自動的に実行されます。

この規則によって、各システムに同じコマンドを入力しなくても、ノード・グループ間の整合性を維持することができます。権限を変更すると、ノード・グループ間でいくつかの矛盾が生じることがあります。たとえば、ユーザー ID がノード・グループ内のあるシステムから削除されると、ノード・グループ間で整合性を維持することができなくなります。

権限エラーは別々に処理されます。

次のコマンドは、分散ファイルのすべての部分に影響を与えます。

- 論理ファイル・メンバーの追加 (ADDLFM)
- 物理ファイル制約の追加 (ADDPFCST)
- 物理ファイル・メンバーの追加 (ADDPFM)
- 物理ファイル・トリガーの追加 (ADDPFTRG)
- オブジェクトの割り振り (ALCOBJ)
- 論理ファイルの変更 (CHGLF)
- オブジェクト所有者の変更 (CHGOBJOWN)
- 物理ファイルの変更 (CHGPF)
- 物理ファイル制約の変更 (CHGPFCSST)
- 物理ファイル・メンバーのクリア (CLRPFM)
- ファイルのコピー (CPYF)。このコマンドの追加情報については、24 ページの『DB2 マルチ・システムにおける分散ファイルでのファイルのコピー (CPYF) コマンドの使用』を参照してください。
- 論理ファイルの作成 (CRTLF)
- オブジェクトの割り振り解除 (DLCOBJ)
- ファイルの削除 (DLTF)

- 物理ファイル変更ジャーナルの終了 (ENDJRNPf)。このコマンドの追加情報については、『DB2 マルチ・システムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクト権限の付与 (GRTOBJAUT)
- 物理ファイル制約の除去 (RMVPCST)
- 物理ファイル・トリガーの除去 (RMVPTRG)
- オブジェクトの名前変更 (RNMOBJ)
- 物理ファイル・メンバーの再編成 (RGZPFM)
- オブジェクト権限の取り消し (RVKOBJAUT)
- 物理ファイル・ジャーナルの開始 (STRJRNPf)。このコマンドの追加情報については、『DB2 マルチ・システムでのジャーナル処理の考慮事項』を参照してください。

これらのコマンドでは、分散ファイル以外のオブジェクトが参照される場合は、ユーザーが各システムにそれらのオブジェクトを作成する必要があります。たとえば、物理ファイル・トリガーの追加 (ADDPFTRG) コマンドを使用する場合は、トリガー・プログラムがすべてのシステム上にあることを確認する必要があります。そうしないと、エラーが起こります。これと同じ概念は、物理ファイル・ジャーナルの開始 (STRJRNPf) コマンドに適用されます。ここで、ジャーナルは、すべてのシステムに存在しなければなりません。

ユーザー・プロファイルがリモート・ノードに存在しない場合に、GRTOBJAUT コマンドまたは RVKOBJAUT コマンドを出すと、プロファイルが存在するすべてのノードで権限が付与または取り消されます。プロファイルが存在しないノードでは無視されます。

DB2 マルチ・システムでのジャーナル処理の考慮事項: 物理ファイル・ジャーナルの開始 (STRJRNPf) コマンドと物理ファイル・ジャーナルの終了 (ENDJRNPf) コマンドは、他のシステムに配布されますが、実際のジャーナル処理は、各システムで独立して、各システムの独自のジャーナル・レシーバーに対して起こります。

例としては、分散ファイルのある 2 つのシステム (A および B) が挙げられます。システム A とシステム B の両方にジャーナルとレシーバーを作成する必要があります。また、ジャーナル名とライブラリーは両方のシステムで同じでなければなりません。STRJRNPf コマンドを出すと、コマンドは両方のシステムに配布されて、ジャーナル処理は両方のシステムで開始されます。ただし、システム A のジャーナル・レシーバーには、システム A に常駐するファイルの部分に対して起こる変更を示すデータだけが含まれます。システム B のジャーナル・レシーバーには、システム B に常駐するファイルの部分に対して起こる変更を示すデータだけが含まれます。

これにより、バックアップ方法だけでなく、保管および復元の方法に対しても次のような影響が及びます。

- STRJRNPf コマンドを出したら、ファイルのノード・グループ内にある各システムから、データベース・ファイルを保管する必要があります。

- 各システムに対して、標準ジャーナル管理を行う必要があります。ジャーナル・レシーバーを正しく変更、保管して、各システムのディスク・スペース使用を管理できるようにする必要があります。あるいは、システム変更ジャーナル管理機能サポートを利用することができます。

注: ジャーナルの名前は各システムで同じでなければなりません。属性は同じでなくても構いません。したがって、たとえば、システムによって異なるジャーナル・レシーバーしきい値を指定して、各システムで使用可能なディスク・スペースを反映させることができます。

- 分散データベース・ファイルの部分を回復する必要がある場合は、分散ファイルの部分が存在するシステムから、ジャーナル・レシーバーを使用するだけですみます。このジャーナル・レシーバーから、ジャーナル処理された変更を APYJRNCHG コマンドを使用して適用したり、RMVJRNCHG コマンドを使用して除去したりします。
- 1 つのシステムからジャーナル・レシーバーを使用して、他のシステムのファイル部分にジャーナル変更を適用または除去することはできません。これは、各システム上にあるファイルの各部分が、独自の固有ジャーナル識別子 (JID) を持つためです。

詳しくは、バックアップおよび回復の手引き を参照してください。

DB2 マルチ・システムにおける分散ファイルでのファイルのコピー (CPYF) コマンドの使用: ファイルのコピー (CPYF) コマンドを出すと、システムはできるだけ早く CPYF コマンドを出そうとします。指定されたコマンド・パラメーター、コピーに関与するファイル属性、およびコピーされるレコードのサイズと数はすべて、コマンドの実行速度に影響します。

分散ファイルのデータをコピーする場合、コピー・コマンドのパフォーマンスは、次のパラメーターだけを CPYF コマンドに使用することで改善することができます。すなわち、FROMFILE、TOFILE、FROMMBR、TOMBR、MBROPT、および FMTOPT(*NONE) または FMTOPT(*NOCHK) です。また、取り出しファイル (FROMFILE) と受け入れファイル (TOFILE) の各パラメーターは、ヌル可能フィールドを含むファイルを指定することはできません。一般に、コピー・コマンドの構文が簡単になるほど、最高速コピー処理が行われる確率は高くなります。分散ファイルへのコピー中に最高速コピー方式が使用されると、メッセージ CPC9203 が出されて、各ノードにコピーされるレコード数を示します。通常、このメッセージが出されないと、最高速コピーは実行されません。

分散ファイルにコピーする際には、最高速コピーが使用されるかどうかによって、以下のような違いがあることに注意してください。

- 最高速コピーの場合、レコードは各ノードごとにバッファーに入れられます。バッファーがいっぱいになると、特定のノードに送られます。レコードがいずれかのノード・バッファーに入れられた後でエラーが起これば、システムは、現在ノード・バッファー内にあるレコードすべてを正しいノードに送ろうとします。システムが特定のノードにレコードを送っているときにエラーが起これば、処理は次のノードに続けられ、システムがノード・バッファーすべてを送ろうとします。

すなわち、エラーのある特定レコードに続くレコードは、分散ファイルに書き込まれるはずですが、このアクションは、複数のノードで同時ブロック化が行われる

ために起こります。エラーのあるレコードに続くレコードを、分散ファイルに書き込みたくない場合は、CPYF コマンドに 1 以上の値を持つ ERRLVL(*NOMAX) または ERRLVL のいずれかを指定することによって、最高速コピーを使用させないようにすることができます。

最高速コピーを使用しないと、配布先ファイルのオープンが SEQONLY(*NO) オープンでない場合や、SEQONLY(*NO) オープンになるように設定されていない場合には、レコードのブロック化が行われます。

2. 最高速コピーを使用すると、メンバーのオープンが SEQONLY (*NO) に変更されたことを示すメッセージが出されます。ただし、配布先ファイルは、2 回目には、レコードのブロック化を可能にするためにオープンされます。SEQONLY(*NO) への変更に関するメッセージは無視してください。
3. 最高速コピーを使用すると、各ノードにコピーされるレコードの数を示す複数のメッセージが出されます。次に、コピーされるレコードの総数を示すメッセージが送られます。

最高速コピーを使用しないと、レコードがコピーされたことを示すメッセージの総数だけが送られます。各ノードにコピーされたレコードの数をリストするメッセージは送信されません。

次に、分散ファイルとのコピーの制限について示します。

- FROMRCD パラメーターは、分散ファイルからコピーする場合、*START または 1 という値だけを使用して指定することができます。TORCD パラメーターは、分散ファイルからコピーするときに、省略時値 *END 以外の値を使用して指定することはできません。
- MBROPT(*UPDADD) パラメーターは、分散ファイルへのコピー時には指定できません。
- COMPRESS(*NO) パラメーターは、コピー先ファイルが分散ファイルであり、コピー元ファイルがデータベース削除可能ファイルである場合、指定できません。
- コピー印刷リストの場合、指定された RCDNBR 値は、レコードが分散ファイル・レコードの場合、特定ノードのファイル内のレコードの位置を示します。同じレコード番号が、リスト上に複数回表示される場合があります。各番号は、異なるノードからのレコードを示します。

DB2 マルチ・システムでの区分化

区分化とは、ノード・グループ内でノード間にファイルを配布するプロセスのことをいいます。区分化は、ハッシュ・アルゴリズムを使用して行われます。新しいレコードが追加されると、ハッシュ・アルゴリズムが区分化キー内のデータに適用されます。ハッシュ・アルゴリズムの結果である 0 から 1023 の間の数字は、区分化マップに適用されて、レコードが存在するノードを判別します。

区分化マップは、QUERY 最適化、更新、削除、および結合にも使用されます。区分化マップをカスタマイズして、特定のキー値を特定のノードに使用することができます。

たとえば、入出力中、システムはハッシュ・アルゴリズムを区分化キー・フィールド内の値に適用します。この結果は、ファイル内に保存された区分化マップに適用されて、どのノードがレコードを保存したかを判別します。

次の例は、これらの概念が相互にどのように関連しているかを示しています。

社員番号は区分化キーであり、レコードは社員番号 56000 のデータベースに入力されます。56000 という値はハッシュ・アルゴリズムによって処理され、結果は 733 という区画番号になります。区画マップはノード・グループ・オブジェクトの一部であり、作成時に分散ファイルに格納されます。これには、区画番号 733 を示すノード番号 1 が含まれます。したがって、このコードは、ノード番号 1 を割り当てられたノード・グループ内のシステムに物理的に格納されます。区分化キー (PTNKEY パラメーター) は、区分化 (分散) ファイルを作成したときに、ユーザーによって指定されています。

区分化キー内のフィールドは、ヌル値を入れられるようにすることができます。ただし、区分化キー内にヌル値を含むレコードは、常に、区画番号 0 にハッシュされます。区分化キー内に有効な数のヌル値を持つファイルは、区画番号 0 でデータ・スキューになる可能性があります。これは、ヌル値を持つすべてのレコードが区画番号 0 にハッシュするためです。

ノード・グループ・オブジェクトと区分化分散リレーショナル・データベース・ファイルを作成した後、DSPNODGRP コマンドを使用して、区画番号とノード名の間を表示することができます。区画番号、ノード・グループ、およびシステム名を表示する方法については、11 ページの『DB2 マルチ・システムでの DSPNODGRP コマンドを使用したノード・グループの表示』を参照してください。

分散ファイルを作成する場合は、区分化キー・フィールドは物理ファイルの作成 (CRTPF) コマンドの PTNKEY パラメーターか、または SQL CREATE TABLE ステートメントの PARTITIONING KEY 文節に指定されます。データ・タイプ DATE、TIME、TIMESTAMP、および FLOAT を持つフィールドは、区分化キーでは使用できません。

区分化をインプリメントするには、次のトピックを参照してください。



- 『DB2 マルチ・システムでの区分化の計画』
- 27 ページの『DB2 マルチ・システムでの区分化キーの選択』

DB2 マルチ・システムでの区分化の計画

ほとんどの場合、区分化および区分化キーをどのように使用したいかに関する計画を立てておく必要があります。他のシステムに配置するために、データをどのように体系的に分割するか、QUERY に頻繁に結合させたいデータは何かを決める必要があります。また、選択を行う場合に有効な選択は何か、区分化キーを設定して必要なデータを獲得するための最も効率的な方法は何かを決める必要もあります。

区分化を計画する場合は、最高速システムがほとんどのデータを受け取るようにそれを設定することができます。どのシステムで対称マルチプロセッシング (SMP) 並行処理機能を利用してデータベースのパフォーマンスを改善するかを考える必要があります。QUERY 最適化プログラムがその分散アクセス計画を作成する場合、最適化プログラムは、要求ノード上のレコード数をカウントして、その数にノードの総数を乗算することに注意してください。ほとんどのレコードを SMP システムに入れることができますが、最適化プログラムは、これらの利点のいくつかを相殺する可能性があります。これは、最適化プログラムが、各ノードで等しい数のレコー

ドをその計算に使用するためです。

SMP については、『SQL プログラミング 概念 』および『データベース・プログラミング 』を参照してください。

区分化に影響を与えたい場合は、そのようにすることができます。たとえば、会社で、特定のシステムを使用して作業を完了する地域販売部門があるとした場合、区分化を使用すると、ある地域のローカル・データを、それぞれの地域の該当するシステムに格納することができます。したがって、米国北西部の社員が使用するシステムには、北西部のデータを含むようにすることができます。

区分化を設定するには、CRTPF コマンドの PTNFILE および PTNMBR の各パラメータを使用することができます。ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用すると、すでに区分化されたファイルを再分配することができます。詳細については、28 ページの『DB2 マルチ・システムでのデータ分配のカスタマイズ』を参照してください。

パフォーマンスの向上は、大きなファイル間で行われる QUERY で最高になります。トランザクション処理での使用頻度が高く、QUERY ではほとんど使用されないファイルは、区分化には適していないため、ローカル・ファイルとして残す必要があります。

結合処理では、特定のフィールドで 2 つのファイルを結合することが多い場合、そのフィールドを両方のファイルの区分化キーにする必要があります。また、フィールドが同じデータ・タイプであることも確認する必要があります。

DB2 マルチ・システムでの区分化キーの選択

システムが最も効率的な方法で区分化ファイル进行处理するためには、区分化キーを設定または使用するときに、下記の点を考慮する必要があります。

- 最も適切な区分化キーとは、多数の異なる値を持つキーであるため、区分化活動ではデータ・レコードの均等な配布が起こります。顧客番号、名字、クレーム番号、郵便番号、および電話の市外局番などが、区分化キーとして使用するのに適したカテゴリーの例としてあげられます。

性別は、男と女の 2 つの選択しかないため、区分化キーには適していない例といえます。性別では、大量のデータが、複数のノードに配布されるのではなく、単一のノードに配布されてしまいます。また、QUERY を実行する場合は、区分化キーとして性別を使用すると、システムは多数のデータ・レコードを処理しなければなりません。これは非効率的です。別のフィールドまたはデータ・フィールドを使用すると、QUERY の有効範囲を狭めて、処理をはるかに効率的にすることができます。性別に基づく区分化キーは、特定の値に基づく分配ではなく、データの均等な分配が必要な場合には不適切です。

ローカル・ファイルを分散ファイルに変更する用意をする場合は、HASH 関数を使用して、データをどのように分配できるかを考えることができます。HASH 関数は、ローカル・ファイルの各種の列に対して使用できるため、実際にファイルを分散ファイルに変更する前に、異なる区分化キーを試すことができます。たとえば、ファイルの郵便番号フィールドを使用する場合は、そのフィールドを使用して HASH 関数を実行して、各区画番号にハッシュするレコード数を考えるこ

とができます。このことは、区分化キー・フィールドを選択するうえで、またはノード・グループで区画マップを作成するうえで役立ちます。

- 頻繁に更新する必要があるフィールドは選択しないでください。区分化キー・フィールドの制約の 1 つに、更新によってレコードが異なるノードに強制されない場合にのみ、その値を更新できるというものがあります。
- 区分化キーには多数のフィールドを使用しないでください。1 つのフィールドを使用する方法が最も適しています。多数のフィールドを使用すると、システムは入出力時にさらに多くの作業を実行することになります。
- 固定長文字または整数などの簡単なデータ・タイプを区分化キーとして選択してください。この考慮事項は、ハッシュが単純なデータ・タイプの単一フィールドに行われるため、パフォーマンスに役立つ場合があります。
- 区分化キーを選択する場合は、通常実行する QUERY の結合とグループ化の基準を考慮する必要があります。たとえば、結合に関与するファイルの結合フィールドとして使用されることのないフィールドを選択すると、逆に結合のパフォーマンスに影響が及ぶ可能性があります。分散ファイルが関係する QUERY の実行については、『第 6 章 DB2® マルチ・システムでのパフォーマンスのための QUERY 設計』を参照してください。

DB2 マルチ・システムでのデータ分配のカスタマイズ

データの配置についてはシステムが担当するため、ユーザーはレコードの実際の所在を知る必要はありません。ただし、特定のレコードが特定のシステムに常に格納されるようにしたい場合は、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して、これらのレコードの所在を指定することができます。

たとえば、郵便番号 55902 のレコードすべてを、Minnesota 州 Minneapolis のシステムに常駐させたい場合を想定します。CHGNODGRPA コマンドを出す場合は、郵便番号 55902 と、Minneapolis のローカル・ノードのシステム・ノード番号を指定します。

この時点で、郵便番号 55902 はノード・グループを変更していますが、データはまだ以前と同じ場所に配布されます。CHGNODGRPA コマンドは、既存のファイルに影響を与えません。区分化ファイルが作成されると、その区分化ファイルは、そのときのノード・グループの情報のコピーを保持します。ノード・グループは、区分化ファイルに影響を与えずに変更、または削除することができます。有効にするために再分配されるレコードへの変更では、新しいノード・グループを使用して分散ファイルを再作成するか、または物理ファイルの変更 (CHGPF) コマンドを使用して、新しいあるいは更新されたノード・グループを指定することができます。

CHGPF コマンドを使用すると、次のことを行うことができます。

- すでに区分化されたファイルを再分配する。
- 区分化キーを変更する (たとえば、電話の市外局番からブランチ ID へ)。
- ローカル・ファイルを分散ファイルに変更する。
- 分散ファイルをローカル・ファイルに変更する。



注: CHGNODGRPA コマンドを使用して、すでに区分化されているファイルを再分配する必要があります。CHGNODGRPA コマンドを CHGPF コマンドとともに任意指定で使用すると、他のいずれかのタスクを実行することができます。

ローカル・ファイルを分散ファイルに変更する方法、または分散ファイルをローカル・ファイルに変更する方法については、40ページの『ネットワークにシステムを追加する際の再分配の問題』を参照してください。

第 4 章 DB2 マルチ・システムで使用可能なスカラー関数

DB2 マルチ・システムでは、分散ファイル进行处理するときを使用できる新しいスカラー関数が用意されています。これらの関数を使用すると、ファイルが分散された後のデータの場所だけでなく、ファイルにおけるデータの分散方法を判別することができます。データベース管理者は、分散ファイル进行处理する場合、これらの関数をデバッグ用ツールとして使用することができます。

これらのスカラー関数には PARTITION、HASH、NODENAME、および NODENUMBER があります。これらの関数は、SQL または QUERY ファイルのオープン (OPNQRYF) コマンドによって使用することができます。

SQL スカラー関数の構文については、『SQL 解説書 』を参照してください。OPNQRYF スカラー関数の構文については、Information Center のトピック『制御言語 』を参照してください。

- 『DB2 マルチ・システムでの PARTITION』
- 32 ページの『DB2 マルチ・システムでの HASH』
- 33 ページの『DB2 マルチ・システムでの NODENAME』
- 34 ページの『DB2 マルチ・システムでの NODENUMBER』
- 34 ページの『DB2 マルチ・システムでの特殊レジスター』

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの PARTITION

PARTITION 関数によって、分散リレーショナル・データベースの特定の行が格納されている区画番号を判別することができます。区画番号を知っていると、ノード・グループ内のどのノードにその区画番号が含まれるかを判別することができます。

PARTITION のコードの例については、『DB2 マルチ・システムでの PARTITION の例』を参照してください。

DB2 マルチ・システムでの PARTITION の例

- EMPLOYEE 表の行すべての PARTITION 番号を検出します。

注: これは、データ・スキューがあるかどうかを判別するために使用することができます。

SQL ステートメント:

```
SELECT PARTITION(CORPDATA.EMPLOYEE), LASTNAME  
FROM CORPDATA.EMPLOYEE
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
          FORMAT(FNAME)
          MAPFLD((PART1 '%PARTITION(1)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

- EMPLOYEE 表から、区画番号が 100 に等しいすべての行に社員番号 (EMPNO) を選択します。

SQL ステートメント:

```
SELECT EMPNO
FROM CORPDATA.EMPLOYEE
WHERE PARTITION(CORPDATA.EMPLOYEE) = 100
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('%PARTITION(1) *EQ 100')
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

- EMPLOYEE 表と DEPARTMENT 表を結合し、2 つの表の行が同じ区画番号を持つ結果の行すべてを選択します。

SQL ステートメント:

```
SELECT *
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE PARTITION(X)=PARTITION(Y)
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
          FORMAT(FNAME)
          JFLD((1/PART1 2/PART2 *EQ))
          MAPFLD((PART1 '%PARTITION(1)')
                 (PART2 '%PARTITION(2)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの HASH

HASH 関数は、指定の式にハッシュ関数を適用することによって、区画番号を戻します。

HASH のコードの例については、『DB2 マルチ・システムでの HASH の例』を参照してください。

DB2 マルチ・システムでの HASH の例

- 区分化キーが EMPNO と LASTNAME から構成されている場合、区画が何になるかを判別するために HASH 関数を使用します。この QUERY では、EMPLOYEE のすべての行の区画番号が戻されます。

SQL ステートメント:

```
SELECT HASH(EMPNO, LASTNAME)
FROM CORPDATA.EMPLOYEE
```


OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
          FORMAT(FNAME)
          MAPFLD((HASH '%HASH(1/EMPNO, 1/LASTNAME)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの NODENAME

NODENAME 関数によって、分散リレーショナル・データベースの特定の行が格納されているリレーショナル・データベース (RDB) の名前を判別することができます。ノード名を知っていると、その行を含むシステム名を判別することができます。これは、ある行を特定のノードに再配布する必要があるかどうかを判別する際に役立ちます。

NODENAME のコードの例については、『DB2 マルチ・システムでの NODENAME の例』を参照してください。

DB2 マルチ・システムでの NODENAME の例

- EMPLOYEE 表の行すべてのノード名と区画番号と、各行の EMPNO 列の対応する値を検出します。

SQL ステートメント:

```
SELECT NODENAME(CORPDATA.EMPLOYEE), PARTITION(CORPDATA.EMPLOYEE), EMPNO
FROM CORPDATA.EMPLOYEE
```

- EMPLOYEE 表のすべてのレコードのノード名を検出します。

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
          FORMAT(FNAME)
          MAPFLD((NODENAME '%NODENAME(1)'))
```

- EMPLOYEE 表と DEPARTMENT 表を結合し、社員番号 (EMPNO) を選択して、生成された結合に各行が含まれるときの元のノードを判別します。

SQL ステートメント:

```
SELECT EMPNO, NODENAME(X), NODENAME(Y)
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE X.DEPTNO=Y.DEPTNO
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
          FORMAT(FNAME)
          JFLD((EMPLOYEE/DEPTNO DEPARTMENT/DEPTNO *EQ))
          MAPFLD((EMPNO 'EMPLOYEE/EMPNO')
                  (NODENAME1 '%NODENAME(1)')
                  (NODENAME2 '%NODENAME(2)'))
```

- EMPLOYEE 表と DEPARTMENT 表を結合し、2 つの表の行が同じノードにある結果の行すべてを選択します。

SQL ステートメント:

```
SELECT *
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE NODENAME(X)=NODENAME(Y)
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
          FORMAT(FNAME)
          JFLD((1/NODENAME1 2/NODENAME2 *EQ))
          MAPFLD((NODENAME1 '%NODENAME(1)')
                 (NODENAME2 '%NODENAME(2)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの NODENUMBER

NODENUMBER 関数によって、分散リレーショナル・データベースの特定の行が格納されているノード番号を判別することができます。ノード番号とは、ノード・グループの作成時に、ノード・グループ内で各ノードに割り当てられる固有の番号のことをいいます。ノード番号を知っていると、その行を含むシステム名を判別することができます。これは、ある行を特定のノードに再配布する必要があるかどうかを判別する際に役立ちます。

NODENUMBER のコードの例については、『DB2 マルチ・システムでの NODENUMBER の例』を参照してください。

DB2 マルチ・システムでの NODENUMBER の例

- CORPDATA.EMPLOYEE が分散表の場合、各行のノード番号と社員名が戻されません。

SQL ステートメント:

```
SELECT NODENUMBER(CORPDATA.EMPLOYEE), LASTNAME
FROM CORPDATA.EMPLOYEE
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
          FORMAT(FNAME)
          MAPFLD((NODENAME '%NODENUMBER(1)')
                 (LNAME '1/LASTNAME'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの特殊レジスター

DB2 マルチ・システムでは、特殊レジスターのすべてのインスタンスが、リモート・ノードへ QUERY を送信する前にコーディネーター・ノードで変換されます。(コーディネーター・ノードとは、QUERY が開始されたシステムのことをいいます。) このように、すべてのノードが、整合のとれた特殊レジスター値によって QUERY を実行します。

次に、特殊レジスターに関する規則を示します。

- CURRENT SERVER は、常にコーディネーター・ノードのリレーショナル・データベース名を戻します。

- `USER` 特殊レジスターは、コーディネーター・ノード上でジョブを実行するユーザー・プロファイルを戻します。
- `CURRENT DATE`、`CURRENT TIME`、および `CURRENT TIMESTAMP` は、コーディネーター・ノードの時刻機構 (TOD) によります。
- `CURRENT TIMEZONE` は、コーディネーター・ノードのシステム値 `QUTCOFFSET` の値です。

分散ファイルのノードに格納された行の相対レコード番号を戻すことについては、『DB2 マルチ・システムでの相対レコード番号 (RRN) 関数』を参照してください。

DB2 マルチ・システムでの相対レコード番号 (RRN) 関数

RRN 関数は、分散ファイルのノードに格納された行の相対レコード番号を戻します。すなわち、RRN は分散ファイルに固有のものではありません。RRN を `NODENAME` または `NODENUMBER` のいずれかと結合すると、ファイル内の固有レコードが指定されます。

第 5 章 DB2 マルチ・システムでのパフォーマンスとスケーラビリティ

DB2 マルチ・システムでは、データベースの容量を増やして、QUERY のパフォーマンスの向上を実現し、より簡単な方式によって、リモート・データベース・アクセスを可能にすることができます。

DB2 マルチ・システムを使用すると、ユーザーおよびアプリケーションは、ローカル・システムからファイルにアクセスするだけで済みます。ローカル・ファイルではなく、分散ファイル内のデータにアクセスできるようにするために、コードを変更する必要はありません。分散リレーショナル・データベース・アーキテクチャー (DRDA) および分散データ管理機能 (DDM) などの機能では、リモート・データにアクセスするために、リモート・ファイルまたはリモート・システムに対してアクセスを明示的に指示する必要があります。DB2 マルチ・システムは、ユーザーが関知しなくてもすむようにリモート・アクセスを処理します。

DB2 マルチ・システムには、データベース拡張のための単純な拡張パスも用意されています。

DB2 マルチ・システムのパフォーマンスおよびスケーラビリティについては、さらに、次のトピックを参照してください。

- 『DB2 マルチ・システムの使用』
- 39 ページの『データベース・システムの拡張における DB2 マルチ・システムの効用』

DB2 マルチ・システムの使用

パフォーマンスの向上が、特定の QUERY にとって非常に問題になる場合があります。処理すべき大量のデータがあるのに対して、結果セットが比較的小さい照会の場合には、ファイルが分配されるシステムの数に比例してパフォーマンスが向上することがテストによりわかっています。たとえば、500 万レコードのファイルがあって、その上位 10 位までの高額所得者を照会したい場合を想定します。DB2 マルチ・システムでは、ファイルを 2 つのシステム間で等分に区分化することによって、照会に対する応答時間をほぼ 2 分の 1 にすることができます。3 つのシステムに区分化すると、応答時間は、単一のシステムで照会を実行した場合のほぼ 3 分の 1 になります。これは最適な条件の場合であり、データをノード間で移動する必要のある複雑な結合の場合にはあてはまりません。

ファイルが非常に小さいか、または主に単一レコードの読み書き処理に使用される場合、ファイルを区分化しても、パフォーマンスはほとんどあるいはまったく向上しません。それどころか、パフォーマンスが若干低下する場合があります。このような場合、照会のパフォーマンスはむしろ物理的な接続の速度によって決まります。ただしこのような場合でも、ノード・グループ内のすべてのシステムのユーザーは、データが配布されていても、使いたれた従来の“ローカル・ファイル”データベース方式を使用して、そのデータにアクセスすることができるという利点があ

ります。ユーザーは、すべての環境において、このローカル・システムの透過性と、ノード・グループ内のシステム間のデータ冗長度の除去という恩恵に浴しています。

並列機能には、他に DB2 UDB 対称マルチプロセッシングがあります。対称マルチプロセッシング (SMP) を使用すると、区分化されたファイルが処理されて、システムのいずれかがマルチプロセッサ・システムである場合、パフォーマンス向上において乗数効果が得られます。ファイルを 3 つのシステム間で区分化して、各システムが 4 ウェイのプロセッサ・システムである場合は、DB2 マルチ・システムと SMP の機能がともに作動します。前に示した 500 万レコードの例をとると、応答時間は、並列機能をまったく使用しないで照会が実行された場合のおよそ 12 分の 1 になります。ファイル・サイズと QUERY の詳細度によって、実際の向上に影響が及ぶ場合があります。

QUERY を実行すると、QUERY を実行するための大量の作業が並列して実行されます。これにより、QUERY 処理全体のパフォーマンスが向上します。システムは、QUERY を分割し、QUERY の適切な部分を適切なシステム上で処理します。これにより、処理がますます効率的に、また自動的に行われるようになります。何も指定しなくても、このような、より効率的な処理を行うことができます。

注: QUERY によっては、パフォーマンスが向上しない場合があります。特に、大量のデータを移動しなければならない場合などです。

各ノードは、そのノードに物理的に格納されたレコードだけを処理する必要があります。QUERY で、区分化キーに対する選択を指定した場合、QUERY 最適化プログラムが、1 つのノードだけを照会する必要があると判別することがあります。次の例では、郵便番号フィールドが、ORDERS ファイルの SQL ステートメント内の区分化キーです。

```
SELECT NAME, ADDRESS, BALANCE FROM PRODLIB/ORDERS WHERE ZIP='48009'
```

ステートメントが実行されると、最適化プログラムは、1 つのノードだけを照会する必要があると判別します。48009 という郵便番号を含むすべてのレコードが、同じノードに配布されることを覚えておいてください。

次の SQL ステートメントの例では、ノード・グループ内のすべての iSeries サーバーのプロセッサ機能を使用して、ステートメントを並列処理することができます。

```
SELECT ZIP, SUM(BALANCE) FROM PRODLIB/ORDERS GROUP BY ZIP
```

最適化プログラムに、適切なデータを含むシステム宛てにだけ直接入出力要求を出させると、1 つまたは複数のシステムがアクティブではない場合にでも照会を実行できるという利点も得られます。この例として、ある業務の各部門が異なるシステムにデータを保存するように区分化されたファイルがあげられます。あるシステムが使用できない場合でも、残りの部門に関連するデータに対してファイル入出力操作を実行することができます。活動状態ではない部門に対する入出力要求は失敗します。

最適化プログラムは、2 フェーズのコミット・プロトコルを使用して、データの保全性を保証します。複数のシステムがアクセスされるため、コミットメント制御を要求すると、すべてのシステムが、**保護会話**を使用します。保護会話とは、トラン

ザクシヨンの途中、または単一データベース操作の途中でシステム障害が発生した場合、その時点までに行われた変更すべてがロールバックされるということを意味します。

保護会話を使用すると、一部のコミットメント制御オプションは、パフォーマンスを高めるために、リモート・ノードで変更されます。発信待機オプションは Y に、ポート読み取り専用オプションは Y に設定されます。パフォーマンスをさらに向上させるには、コミットメント変更オプション (QTNCHGCO) API を使用して、QUERY を開始するシステム上で発信待機オプションを N に変更します。これらのコミットメント・オプションの値の効果については、Information Center のトピック『OS/400 API』を参照してください。

コミットメント制御について詳しくは、資料バックアップおよび回復の手引き を参照してください。

優れたパフォーマンスを確保する方法については、『DB2 マルチ・システムでのパフォーマンス向上のヒント』を参照してください。

DB2 マルチ・システムでのパフォーマンス向上のヒント

優れたパフォーマンスを確保する方法の 1 つに、*BUFFERED をデータベース・ファイルの一時変更 (OVRDBF) コマンドのデータ配布 (DSTDTA) パラメーターに指定する方法があります。これは、システムに対して、ファイルに即時更新を加えるという犠牲を払う可能性があっても、できるだけ早く分散ファイルからデータを検索するように指示するものです。DSTDTA(*BUFFERED) は、ファイルが読み取り専用のためにオープンされている場合のパラメーターの省略時値です。

DSTDTA パラメーターには、*CURRENT および *PROTECTED という値もあります。*CURRENT を使用すると、他のユーザーによるファイルへの更新が可能になりますが、パフォーマンスが若干低下します。ファイルを更新のためにオープンした場合は、DSTDTA(*CURRENT) が省略時値になります。*PROTECTED では、*CURRENT と同様のパフォーマンスが得られますが、*PROTECTED を使用すると、ファイルのオープン中に他のユーザーが更新を行えなくなります。

データベース・システムの拡張における DB2 マルチ・システムの効用

分散リレーショナル・データベース・ファイルを使用すると、iSeries システムの構成をより簡単に拡張することができます。DB2 マルチ・システム以前には、1 つのシステムから 2 つのシステムに移行したい場合、解決しなければならないデータベースの問題がいくつかありました。ユーザーの半分を新しいシステムに移動する場合、データの半分もその新しいシステムに移動することになります。これにより、データベース関連のアプリケーションをすべて再作成することになりますが、これは、アプリケーションがデータの所在を知っていなければならないためです。アプリケーションを作成し直したら、分散リレーショナル・データベース・アーキテクチャー (DRDA) または分散データ管理機能 (DDM) などのリモート・アクセスを使用して、複数のシステムにあるファイルにアクセスする必要があります。これを行わない場合は、なんらかのデータ複製機能を使用する必要があります。データ複製機能を使用すると、データのコピーが複数存在して、さらに多くの記憶域が使用され、システムはファイルの複数のコピーを同じレベルで保持するための作業を行うことが必要になります。

DB2 マルチ・システムでは、構成に新しいシステムを追加するためのプロセスが大幅に簡素化されています。データベース・ファイルは、複数のシステム間で区分化されます。次に、アプリケーションが新しいシステムに移動します。アプリケーションは変更されません。アプリケーションにプログラミング変更を加える必要はありません。ユーザーは、新しいシステム上で実行できるようになり、すぐに同じデータにアクセスすることができます。さらに拡張が必要になる場合は、追加のシステムを含む新しいノード・グループ間でファイルを再分配することができます。

ネットワークにシステムを追加するときの問題については、『ネットワークにシステムを追加する際の再分配の問題』を参照してください。

ネットワークにシステムを追加する際の再分配の問題

ノード・グループ間でのファイルの再分配は、非常に簡単な処理です。物理ファイルの変更 (CHGPF) コマンドを使用すると、ファイルに新しいノード・グループを指定するか、またはファイルの新しい区分化キーを指定することができます。CHGPF コマンドを使用すると、ローカル・ファイルを分散ファイルにするか、分散ファイルをローカル・ファイルにするか、または分散ファイルをノードの異なるセット間、または異なる区分化キーによって再分配することができます。

再分配のプロセスでは、ファイル内のほとんどすべてのレコードの移動が必要になる場合があることに注意する必要があります。非常に大きなファイルの場合、これは長時間のプロセスになって、その間ファイルが使用できなくなるおそれがあります。したがって、ファイルの再分配を頻繁に行ったり、適切な計画なしに行ったりはしないでください。

ローカル物理ファイルを分散ファイルに変更するには、ノード・グループ (NODGRP) と区分化キー (PTNKEY) の各パラメーターを CHGPF コマンドに指定する必要があります。このコマンドが実行されると、ファイルが変更されてノード・グループ内のノード間に分配され、さらに、PTNKEY パラメーターに指定された区分化キーを使用して、既存のデータも分配されます。

分散ファイルをローカル・ファイルに変更するには、NODGRP(*NONE) を CHGPF コマンドに指定する必要があります。これにより、ファイルのリモート部分すべてが削除され、すべてのデータがローカル・システムに返されます。

分散ファイルの区分化キーを変更するには、CHGPF コマンドの PTNKEY パラメーターに必要なフィールドを指定してください。これによって、ファイルが分配されるシステムに影響が及ぶことはありません。ハッシュ・アルゴリズムを新しい区分化キーに適用する必要があるため、データのすべてが再分配されます。

ファイルが分配されるシステムの新しい集合を指定するには、CHGPF コマンドのノード・グループ (NODGRP) パラメーターにノード・グループ名を指定してください。これにより、ファイルは、システムの新しい集合に分配されます。新しい区分化キーを PTNKEY パラメーターに指定することができます。PTNKEY パラメーターが指定されていないか、または *SAME が指定されている場合は、既存の区分化キーが使用されます。

CHGPF コマンドは、ノード・グループに新しいシステムが追加された場合、ファイルの新しい部分の作成を行います。CHGPF コマンドは、システムが新しいノード・グループにない場合、ファイルの部分的削除を行います。ノード・グループを削除

して再作成し、CRTPF コマンドを使用してファイルを再分配したい場合は、CHGPF コマンドの NODGRP パラメーターにノード・グループ名を指定する必要があります。これは、ノード・グループ名が、ファイルが最初に作成されたときに使用されたものと同じであってもです。これは、システムにそのノード・グループを調べさせて、ファイルを再分配させたいということを示します。ただし、ノード・グループを NODGRP パラメーターに指定して、システムが、ファイル内に現在格納されているノード・グループとそのノード・グループが同じであると認めた場合は、PTNKEY を同時に指定しないかぎり、再分配は起こりません。

参照制約のあるファイルでは、CHGPF コマンドを使用して親ファイルと従属ファイルを分散ファイルにしたい場合、次のことを行う必要があります。





- 参照制約を除去する。制約を除去しないと、最初に分配するファイルに制約エラーが起こります。これは、参照制約関係内の他のファイルが、まだ分散ファイルではないためです。
- CHGPF コマンドを使用して、両方のファイルを分散ファイルにする。
- 参照制約をもう一度追加する。

第 6 章 DB2[®] マルチ・システムでのパフォーマンスのための QUERY 設計

この章では、QUERY を設計するためのいくつかの指針を示して、分散ファイル使用の QUERY を実行するときに、より効率的に QUERY リソースを使用できるようにします。さらに分散ファイルを使用する QUERY が組み込まれる方法についても説明します。これにより、QUERY を調整して、分散環境においてより効率的に実行できるようにします。

分散ファイルは、SQL、QUERY ファイルのオープン (OPNQRYF) コマンド、またはシステム上のいずれかの QUERY インターフェースを使用して照会することができます。QUERY は、単一ファイル照会または結合照会のいずれかにすることができます。分散ファイルとローカル・ファイルの組み合わせを結合して使用することができます。

本章は、ユーザーが、非分散環境での QUERY の実行と最適化に詳しいことを前提としています。これらのトピックについてさらに詳しい情報が必要な場合は、次のようにしてください。

- SQL ユーザーは、『SQL 解説書 』および『SQL プログラミング 概念 』を参照してください。
- SQL ユーザーでないユーザーは、『データベース・プログラミング 』および『CL 解説書 』を参照してください。

また本章では、並行処理機能を追及して、データ移動を最小化することによって、分散 QUERY のパフォーマンスを改善する方法も示しています。

- 44 ページの『DB2 マルチ・システムでの最適化の概要』
- 44 ページの『DB2 マルチ・システムでの単一ファイル QUERY の実施と最適化』
- 46 ページの『DB2 マルチ・システムでのレコード順序の実施と最適化』
- 47 ページの『DB2 マルチ・システムでの UNION 文節と DISTINCT 文節の実施と最適化』
- 47 ページの『DB2 マルチ・システムでの DSTDTA パラメーターおよび ALWCPYDTA パラメーターの処理』
- 48 ページの『DB2 マルチ・システムでの結合の実施と最適化』
- 53 ページの『DB2 マルチ・システムでのグループ化の実施と最適化』
- 56 ページの『DB2 マルチ・システムでの SUBQUERY サポート』
- 56 ページの『DB2 マルチ・システムでのアクセス計画』
- 56 ページの『DB2 マルチ・システムでの再使用可能なオープン・データ・パス (ODP)』
- 57 ページの『DB2 マルチ・システムでの一時結果書き込み機能』

- 60 ページの『DB2 マルチ・システムでの最適化プログラムのメッセージ』
- 62 ページの『DB2 マルチ・システムでの QUERY 属性の変更 (CHGQRYA) コマンドへの変更』
- 64 ページの『パフォーマンスの考慮事項の要約』

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの最適化の概要

分散 QUERY は、分散レベルとローカル・レベルで最適化されます。

- 分散レベルでの最適化は、QUERY を最も効率的なステップに分割して、どのノードがこれらのステップを処理するかに焦点を当てています。
分散最適化プログラムは、分散 QUERY に特有のものです。分散最適化プログラムについてはこの章で説明します。
- ローカル (ステップ) レベルでの最適化は、非分散環境で起こるものと同じ最適化です。この最適化プログラムは、ご存じのものと考えられます。ローカル・レベルでの最適化については、この章では最小限の説明をします。

分散最適化の基本的な前提事項として、各データ・ノードに格納されたレコードの数がほぼ等しく、分散 QUERY のすべてのシステムが類似の構成であるということがあります。分散最適化プログラムによって行われる決定は、システムおよびコーディネーター・ノード・システムのデータ統計に基づいています。

分散 QUERY に複数のステップが必要な場合は、一時結果ファイルが使用されます。一時結果ファイルとは、特定の QUERY ステップの結果を含めるために使用されるシステム作成の一時ファイル (ライブラリー QRECOVERY に保存) のことをいいます。一時結果ファイルの内容は、次の QUERY ステップの入力として使用されます。

DB2 マルチ・システムでの単一ファイル QUERY の実施と最適化

単一分散ファイルを照会すると、すべてのノード、または照会される分散ファイルのノードのサブセットのデータが照会されます。これを行うために、QUERY が指定されたシステムであるコーディネーター・ノードは、QUERY を送信するためのファイルのノードを判別します。これらのノードは、QUERY を実行して、照会されたレコードをコーディネーター・ノードに返します。

この章の例はすべて、DEPARTMENT および EMPLOYEE という分散ファイルを使用します。これらのファイルのノード・グループは、SYSA、SYSB、および SYSC からなります。データは、部門番号で区分化されます。

次の SQL ステートメントは、分散ファイル DEPARTMENT を作成します。

```
CREATE TABLE DEPARTMENT
  (DEPTNO CHAR(3) NOT NULL,
  DEPTNAME VARCHAR(20) NOT NULL,
  MGRNO CHAR(6),
  ADMRDEPT CHAR(3) NOT NULL)
  IN NODGRP1 PARTITIONING KEY(DEPTNO)
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

次は、DEPARTMENT 表の表示です。表示のヘッディングは、ノード、レコード番号、DEPTNO、DEPTNAME、MGRNO、および ADMRDEPT です。

ノード	レコード番号	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
SYSA	1	A00	Support Services	000010	A00
SYSB	2	A01	Planning	000010	A00
SYSC	3	B00	Accounting	000050	B00
SYSA	4	B01	Programming	000050	B00

次の SQL ステートメントは、分散ファイル EMPLOYEE を作成します。

```
CREATE TABLE EMPLOYEE
  (EMPNO CHAR(6) NOT NULL,
   FIRSTNME VARCHAR(12) NOT NULL,
   LASTNAME VARCHAR(15) NOT NULL,
   WORKDEPT CHAR(3) NOT NULL,
   JOB CHAR(8),
   SALARY DECIMAL(9,2))
  IN NODGRP1 PARTITIONING KEY(WORKDEPT)
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

ノード	レコード番号	EMPNO	FIRSTNME	LASTNAME	WORK DEPT	JOB	SALARY
SYSA	1	000010	Christine	Haas	A00	Manager	41250
SYSA	2	000020	Sally	Kwan	A00	Clerk	25000
SYSB	3	000030	John	Geyer	A01	Planner	35150
SYSB	4	000040	Irving	Stern	A01	Clerk	32320
SYSC	5	000050	Michael	Thompson	B00	Manager	38440
SYSC	6	000060	Eileen	Henderson	B00	Accountant	33790
SYSA	7	000070	Jennifer	Lutz	B01	Programmer	42325
SYSA	8	000080	David	White	B01	Programmer	36450

次は、上記の定義済みの分散ファイル EMPLOYEE を使用する QUERY であり、索引 EMPIDX はフィールド SALARY に対して作成されます。この QUERY は SYSA に対して入力されます。

```
SQL:      SELECT * FROM EMPLOYEE WHERE SALARY > 40000
```

```
OPNQRYF: OPNQRYF FILE((EMPLOYEE)) QRYSLT('SALARY > 40000')
```

この場合、SYSA は、上記の QUERY を、SYSA を含む EMPLOYEE のすべてのノードに送ります。各ノードは QUERY を実行し、レコードを SYSA に返します。分散索引は、ファイル EMPLOYEE のフィールド SALARY 上に存在するため、各ノードに実行される最適化では、索引を使用するかどうかを決定します。

次の例では、QUERY は SYSA に指定されますが、QUERY は EMPLOYEE ファイルが存在するノードのサブセットに送られます。この場合、QUERY は SYSA でのみローカルに実行されます。

```
SQL:      SELECT * FROM EMPLOYEE WHERE WORKDEPT = 'A00'
```

```
OPNQRYF: OPNQRYF FILE((EMPLOYEE)) QRYSLT('WORKDEPT = 'A00')
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

分散 QUERY 最適化プログラムは、分離可能なレコード選択 `WORKDEPT = 'A00'` があることを判別します。この QUERY では区分化キー `WORKDEPT` を含みます。最適化プログラムは、'A00' という値をハッシュし、ハッシュ値に基づいて、この条件を満たすすべてのレコードがあるノードを検索します。この場合、この条件を満たすすべてのレコードは SYSA 上にあるため、QUERY はそのノードにのみ送られます。QUERY は SYSA で発生したため、QUERY は SYSA でローカルに実行されます。

次のものは、QUERY が実行されるノードの数を分割します。

- 区分化キーのすべてのフィールドは、分離可能なレコード選択でなければならない
- すべての述部は、等号 (=) 演算子を使用しなければならない
- 区分化キーのすべてのフィールドは、リテラルに比較されなければならない

注: パフォーマンス上の理由から、QUERY を特定のノードに指定するために、区分化キーに一致するレコード選択述部を指定する必要があります。スカラー関数 `NODENAME`、`PARTITION`、および `NODENUMBER` を使用するレコード選択も、QUERY を特定ノードに指示することができます。

DB2 マルチ・システムでのレコード順序の実施と最適化

QUERY に順序を指定した場合、順序の基準がその QUERY とともに送られるため、各ノードは順序付けを並行して実行することができます。最終組み合わせまたは分類がコーディネーター・ノードに対して実行されるかどうかは、指定する QUERY のタイプによって決まります。

各ノードから受け取った順序付けられたレコードの組み合わせは、最適化されています。組み合わせは、コーディネーター・ノードがレコードを受け取ると起こります。分類に対して組み合わせが持つ主なパフォーマンス上の利点は、すべてのノードからのレコードすべてを分類しなくても、レコードを返せることです。

各ノードから受け取った順序付けられたレコードの分類によって、レコードが返される前に、各ノードからのレコードすべてが読み取られ、分類されて、一時結果ファイルに書き込まれます。

組み合わせは、順序が指定されていて、UNION も最終グループ化も必要ではない場合に起こります。そうでない場合、順序 QUERY では、コーディネーター・ノードに対して、最終分類が実行されます。

データのコピー許可 (`ALWCPYDTA`) パラメーターは、分散 QUERY の各ノードが順序基準をどのように処理するかに影響を与えます。この `ALWCPYDTA` パラメーターは、QUERY ファイルのオープン (`OPNQRYF`) および SQL 開始 (`STRSQL`) の各 `CL` コマンドと、`SQLxxx` の作成 (`CRTSQLxxx`) プリコンパイラー・コマンドに指定されます。

- ALWCPYDTA(*OPTIMIZE) を使用すると、各ノードは、分類または索引のどちらかを使用して順序基準を実施するかを決めることができます。このオプションは最適です。
- OPNQRYF コマンドと QUERY API (QQQRY) では、ALWCPYDTA(*YES) または ALWCPYDTA(*NO) を使用すると、各ノードは、指定の順序フィールドと正確に一致する索引を自動的に使用します。これは、最適化プログラムがローカル・ファイルで順序を処理する方法よりも制限されています。

DB2 マルチ・システムでの UNION 文節と DISTINCT 文節の実施と最適化

結合された SELECT ステートメントが分散ファイルを参照する場合、このステートメントは分散 QUERY として処理されます。ステートメントの処理は並行して起こります。結合された各 SELECT のレコードは、結合操作を実行するために、コーディネーター・ノードに返されます。この点で、結合演算子は順次処理されます。

ORDER BY 文節が結合 QUERY とともに指定されている場合、各ノードからのレコードはすべてコーディネーター・ノードで受け取られて、レコードが返される前に分類されます。

DISTINCT 文節が分散 QUERY に指定されている場合、ORDER BY 文節を追加すると、ORDER BY 文節が指定されていない場合よりも速くレコードを返すことができます。DISTINCT に ORDER BY を指定すると、各ノードは、レコードを並列に順序付けすることができます。コーディネーター・ノードの最終組み合わせでは、各ノードから順序付けられたレコードを読み取って、それらのレコードを正しい順序で組み合わせ、最終分類を行わずに、重複するレコードを削除することができます。

DISTINCT 文節が ORDER BY 文節なしで指定されている場合、各ノードからのレコードすべてが、分類が実行されるコーディネーター・ノードに送られます。重複するレコードは、分類されたレコードが返されるときに削除されます。

DB2 マルチ・システムでの DSTDTA パラメーターおよび ALWCPYDTA パラメーターの処理

データのコピー許可 (ALWCPYDTA) パラメーターは、データベース・ファイルの一時変更 (OVRDBF) コマンドのデータの配布 (DSTDTA) パラメーターに指定された値を変更することができます。

一時変更コマンドにライブ・データ (DSTDTA(*CURRENT)) を使用するよう指定してあっても、次のいずれかがあてはまる場合には、DSTDTA は DSTDTA(*BUFFERED) に変更されます。

- 一時コピーが必要であり、ALWCPYDTA(*YES) が指定された
- パフォーマンスを向上させるために一時コピーが選択されて、ALWCPYDTA(*OPTIMIZE) が指定された

この場合、DSTDTA は DSTDTA(*BUFFERED) に変更されます。

DSTDTA(*BUFFERED) を受け入れることが不可能であり、QUERY が一時コピーを必要としない場合は、ALWCPYDTA(*YES) を指定して、DSTDTA(*CURRENT) を有効に保つ必要があります。

DB2 マルチ・システムでの結合の実施と最適化

非分散結合 QUERY でのパフォーマンスの考慮事項だけでなく、分散ファイルに關与する QUERY についてもパフォーマンスの考慮事項があります。結合は、データが区画互換の場合にのみ実行することができます。分散 QUERY 最適化プログラムは、データを区画互換にする計画を生成します。これには、ノード間でのデータの移動が關与する場合があります。

データは、両方のファイルの区分化キーのデータが、同じノード・グループを使用して同じノードにハッシュする場合、**区画互換**になります。たとえば、同じ数値を長整数フィールドまたは短整数フィールドのいずれかに格納しても、同じ値にハッシュします。

次のデータ・タイプは区画互換です。

- 長整数 (4 バイト)、短整数 (2 バイト)、パック 10 進数、およびゾーン数値
- 固定長および可変長の SBCS 文字と、DBCS オープン、DBCS 択一、または DBCS 専用
- 固定長および可変長のグラフィック

日付、時刻、タイム・スタンプ、および浮動小数点の数値データ・タイプは、区分化キーにできないため、区画互換ではありません。

分散ファイルに關与する結合は、共通設定、指定、再区分化、および同報通信の 4 つのタイプに分類されます。次のセクションでは、結合のタイプを定義して、異なる結合タイプの例を示します。

- 『DB2 マルチ・システムでの共通設定結合』
- 50 ページの『DB2 マルチ・システムでの指定結合』
- 50 ページの『DB2 マルチ・システムでの再区分化結合』
- 52 ページの『DB2 マルチ・システムでの同報通信結合』

結合で分散 QUERY 最適化プログラムがどのように動作するかについては、52 ページの『DB2 マルチ・システムでの結合の最適化』も参照してください。

DB2 マルチ・システムでの共通設定結合

共通設定結合では、結合されるファイルの対応するレコードが、同じノード上に存在します。結合されるファイルの区分化キーの値は、区画互換です。結合を実行するために、データを別のノードに移動する必要はありません。この方式は、区分化キーのすべてのフィールドが結合フィールドであり、結合演算子が = (等号) 演算子である QUERY の場合にのみ有効です。また、最初のファイルの区分化キーの n 番目のフィールド (ここで、n は区分化キーのフィールド数 1) は、2 番目のファイルの区分化キーの n 番目のフィールドに結合されなければならない。n 番目のフィールドのデータ・タイプは区画互換でなければなりません。区分化キーのすべてのフ

フィールドが結合に含まれていなければならないことに注意してください。区分化キーのフィールドを含まない追加結合述部は、共通設定結合を実行する能力に影響を与えません。

次の例では、結合述部に両方のファイルの区分化キー・フィールドが関与し、フィールドが区画互換であるため、共通設定結合を実行することができます。これは、DEPTNO および WORKDEPT の突き合わせ値が同じノード上にあることを暗黙指定します。

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
           FROM DEPARTMENT, EMPLOYEE
           WHERE DEPTNO=WORKDEPT
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
           FORMAT(JOINFMT)
           JFLD((DEPTNO WORKDEPT *EQ))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

この QUERY によって戻されるレコードは次のとおりです。

次は、QUERY 結果の表示です。この表のヘッディングは、DEPTNAME、FIRSTNAME、および LASTNAME です。

DEPTNAME	FIRSTNME	LASTNAME
Support Services	Christine	Haas
Support Services	Sally	Kwan
Planning	John	Geyer
Planning	Irving	Stern
Accounting	Michael	Thompson
Accounting	Eileen	Henderson
Programming	Jennifer	Lutz
Programming	David	White

次の例では、追加結合述部の MGRNO=EMPNO は、共通設定結合を実行するための能力に影響を与えません。これは、区分化キーがやはり結合述部に関与するためです。

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
           FROM DEPARTMENT, EMPLOYEE
           WHERE DEPTNO=WORKDEPT AND MGRNO=EMPNO
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
           FORMAT(JOINFMT)
           JFLD((DEPTNO WORKDEPT *EQ) (MGRNO EMPNO *EQ))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

この QUERY によって戻されるレコードは次のとおりです。

DEPTNAME	FIRSTNME	LASTNAME
Support Services	Christine	Haas
Accounting	Michael	Thompson

DB2 マルチ・システムでの指定結合

指定結合では、少なくとも 1 つのファイルの区分化キーが、結合フィールドとして使用されます。結合フィールドは、他のファイルの区分化キーには一致しません。ファイルのレコードは、結合フィールド値のハッシュに基づいて、2 番目のファイルの区画マップとノード・グループを使用して、2 番目のファイルのノードに指定、または送信されます。レコードが、一時分散ファイルを介して 2 番目のファイルのノードに移動すると、共通設定結合が使用されてデータを結合します。この方式は、区分化キーのすべてのフィールドが、少なくとも 1 つのファイルの結合フィールドである等結合 (equijoin) QUERY にのみ有効です。

次の QUERY では、結合フィールド (WORKDEPT) はファイル EMPLOYEE の区分化キーであり、結合フィールド (ADMRDEPT) は DEPARTMENT の区分化キーではありません。データを移動しないで結合を行うと、DEPARTMENT のレコード 2 を EMPLOYEE のレコード 1 と 2 に結合する必要があるため、これらのレコードは異なるノードに保管されるため、結果レコードは失われます。

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
           FROM DEPARTMENT, EMPLOYEE
           WHERE ADMRDEPT = WORKDEPT AND JOB = 'Manager'
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
           FORMAT(JOINFMT)
           QRYSLT('JOB *EQ 'Manager')
           JFLD((ADMRDEPT WORKDEPT *EQ))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

QUERY を実行するために必要な DEPARTMENT のレコードが読み取られ、ADMRDEPT のデータは、区分化マップとノード・グループ EMPLOYEE を使用してハッシュされます。一時ファイルが作成されて、次のようになります。

古いノード	新しいノード	DEPTNAME	ADMRDEPT (新しい区分化キー)
SYSA	SYSA	Support Services	A00
SYSB	SYSA	Planning	A00
SYSC	SYSC	Accounting	B00
SYSA	SYSC	Programming	B00

この一時表は、EMPLOYEE に結合されて、次のものを作成します。結合は、ADMRDEPT が WORKDEPT と区画互換であるため作動します。

DEPTNAME	FIRSTNME	LASTNAME
Support Services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

DB2 マルチ・システムでの再区分化結合

再区分化結合では、ファイルの区分化キーは結合フィールドとして使用されません。両方のファイルのレコードは、各ファイルの結合フィールド値をハッシュする

ことによって移動する必要があります。ファイルの区分化キー・フィールドは、いずれも結合基準に含まれないため、ファイルは、1 つまたは複数の結合フィールドを含む新しい区分化キーでハッシュすることによって、再区分化する必要があります。この方式は、等結合 (equijoin) QUERY にのみ有効です。

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
          FROM DEPARTMENT, EMPLOYEE
          WHERE MGRNO = EMPNO
```

```
OPNQRYF: OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
          FORMAT(JOINFMT)
          JFLD((MGRNO EMPNO *EQ))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

この例では、MGRNO も EMPNO も区分化キーではないため、データを再分配する必要があります。

DEPARTMENT のデータは再分配されます。

古いノード	新しいノード	DEPTNAME	MGRNO (新しい区分化キー)
SYSA	SYSB	Support Services	000010
SYSB	SYSB	Planning	000010
SYSC	SYSC	Accounting	000050
SYSA	SYSC	Programming	000050

EMPLOYEE のデータは再配布されます。

古いノード	新しいノード	FIRSTNME	LASTNAME	EMPNO (新しい区分化キー)
SYSA	SYSB	Christine	Haas	000010
SYSA	SYSC	Sally	Kwan	000020
SYSB	SYSA	John	Geyer	000030
SYSB	SYSB	Irving	Stern	000040
SYSC	SYSC	Michael	Thompson	000050
SYSC	SYSA	Eileen	Henderson	000060
SYSA	SYSB	Jennifer	Lutz	000070
SYSA	SYSC	David	White	000080

この QUERY によって戻されるレコードは次のとおりです。

DEPTNAME	FIRSTNME	LASTNAME
Support Services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

DB2 マルチ・システムでの同報通信結合

同報通信結合では、1つのファイルの選択レコードすべてが、結合が実行される前に、他のファイルのノードすべてに送信または同報通信されます。これは、非等結合 (nonequijoin) QUERY 以外のすべてに使用される結合方式です。この方式は、結合基準が、日付、時刻、タイム・スタンプ、または浮動小数点数値のいずれかのデータ・タイプを持つフィールドを使用する場合にも使用されます。

次の例では、分散 QUERY 最適化プログラムは、EMPLOYEE を同報通信するように決めています。これは、選択 JOB = 'Manager' を適用することによって、より小さいレコード・セットが同報通信されるためです。ノード・グループ内の各ノードの一時ファイルには、選択されたレコードすべてが含まれます。(レコードは各ノードで複写されます。)

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
          FROM DEPARTMENT, EMPLOYEE
          WHERE DEPTNO <> WORKDEPT AND JOB = 'Manager'
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
          FORMAT(JOINFMT)
          QRYSLT('JOB *EQ 'Manager')
          JFLD((DEPTNO WORKDEPT *NE))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

分散 QUERY 最適化プログラムは、次の 2 つの選択レコードを各ノードに送ります。

古いノード	新しいノード	FIRSTNME	LASTNAME	WORKDEPT
SYSA	SYSA, SYSB, SYSC	Christine	Haas	A00
SYSC	SYSA, SYSB, SYSC	Michael	Thompson	B00

この QUERY によって戻されるレコードは次のとおりです。

DEPTNAME	FIRSTNME	LASTNAME
Support Services	Michael	Thompson
Planning	Christine	Haas
Planning	Michael	Thompson
Accounting	Christine	Haas
Programming	Christine	Haas
Programming	Michael	Thompson

DB2 マルチ・システムでの結合の最適化

分散 QUERY 最適化プログラムは、分散ファイルを結合する計画を生成します。分散 QUERY 最適化プログラムは、ファイル・サイズ、各ファイルに選択された予期されるレコード数、および可能な分散結合のタイプを調べます。そのうえで、最適化プログラムは QUERY を複数のステップに分割します。各ステップでは、次のステップへの入力として使用される中間結果ファイルが作成されます。

最適化中、分散結合のタイプに基づいて、各結合ステップのコストが計算されます。このコストには、結合ステップに必要なデータ移動の量が部分的に反映されません。このコストは、最終分散計画を判別するために使用されます。

各ステップでは、できるだけ多くの処理が完了されます。たとえば、指定のステップに限定されたレコード選択は、そのステップ中に実行され、各ステップでは、できるだけ多くのファイルが結合されます。各結合ステップでは、複数のタイプの分散結合が行われる場合があります。共通設定結合と指定結合は、必要なファイルを最初に指定することによって、1つの共通設定結合にすることができます。指定結合と再区分化結合は、すべてのファイルを最初に指定してから、結合を実行することによって、一緒にすることができます。指定結合と再区分化結合は、実際には、結合が行われる前に1つまたは複数のファイルが指定される共通設定結合であることに注意してください。

分散ファイルをローカル・ファイルに結合すると、分散 QUERY 最適化プログラムは、分散ファイルを結合するときコストを計算する場合と同様に、コストを計算します。このコストに基づいて、分散 QUERY 最適化プログラムは次のいずれかを行います。

- すべてのローカル・ファイルを、分散ファイルのデータ・ノードに同報通信して、共通設定結合を実行する。
- すべてのローカル・ファイルおよび分散ファイルを、最大分散ファイルのデータ・ノードに同報通信して、共通設定結合を実行する。
- 分散ファイルをコーディネーター・ノードに指定して、そこで結合を実行する。

結合フィールドに対する区分化キーについては、『DB2 マルチ・システムでの結合フィールドに対する区分化キー』を参照してください。

DB2 マルチ・システムでの結合フィールドに対する区分化キー

結合のタイプに関する前述部分から、共通設定結合を除くすべての分散結合タイプにデータ移動が必要なことがわかります。データ移動の必要性を除去して、パフォーマンスを最大化するには、共通設定結合が可能であるようにすべての QUERY を作成する必要があります。すなわち、分散ファイルの区分化キーは、ファイルを結合するために使用されるフィールドに一致しなければなりません。実行頻度の高い QUERY の場合は、区分化キーを順序またはグループ化基準に一致させるよりも、結合フィールドに一致させる方が重要です。

DB2 マルチ・システムでのグループ化の実施と最適化

分散ファイルを使用する QUERY でのグループ化の実施方式は、区分化キーがグループ化基準に含まれるかどうかによって決まります。グループ化は、下記のセクションで説明するように、1ステップのグループ化、または2ステップのグループ化のいずれかを使用して実施されます。

- 54 ページの『DB2 マルチ・システムでの 1 ステップのグループ化』
- 54 ページの『DB2 マルチ・システムでの 2 ステップのグループ化』

グループ化と結合の両方の使用については、55 ページの『DB2 マルチ・システムでのグループ化と結合』を参照してください。

DB2 マルチ・システムでの 1 ステップのグループ化

区分化キーのすべてのフィールドが GROUP BY フィールドである場合、グループ化は、1 ステップのグループ化を使用して実行することができます。これは、グループのデータすべてが同じノード上にあるためです。次に、1 ステップのグループ化の例を示します。

```
SQL:      SELECT WORKDEPT, AVG(SALARY)
           FROM EMPLOYEE
           GROUP BY WORKDEPT

OPNQRYF:  OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT)
           GRPFLD(WORKDEPT)
           MAPFLD((AVGSAL '%AVG(SALARY)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

WORKDEPT は区分化キーとグループ化フィールドの両方であるため、WORKDEPT の類似の値はすべて同じノード上に存在します。たとえば、A00 の値はすべて SYSA にあり、A01 の値はすべて SYSB 上に存在します。また、B00 の値はすべて SYSC に存在し、B01 の値はすべて SYSA に存在します。このグループ化は、3 つのノードすべてで並行して実行されます。

1 ステップのグループ化を実行するには、区分化キーのすべてのフィールドがグループ化フィールドでなければなりません。追加の非区分化キー・フィールドをグループ化フィールドにすることもできます。

DB2 マルチ・システムでの 2 ステップのグループ化

区分化キーがグループ化フィールドには含まれない場合、グループ化は、2 ステップのグループ化を使用して実行することができます。これは、フィールドの類似値が同じノード上にないためです。次に、2 ステップのグループ化の例を示します。

```
SQL:      SELECT JOB, AVG(SALARY)
           FROM EMPLOYEE
           GROUP BY JOB

OPNQRYF:  OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT2)
           GRPFLD(JOB)
           MAPFLD((AVGSAL '%AVG(SALARY)'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

この例では、JOB が 'Clerk' であるグループにおいて、'Clerk' という値が、EMPLOYEE 分散ファイルの 2 つの異なるノードにあることに注意してください。グループ化は、まず 3 つのノードすべてで並行してグループ化を実行することによって実行されます。これにより、コーディネーター・ノードの一時ファイルに置かれる初期グループ化が起こります。QUERY は変更され、コーディネーター・ノードでグループ化が再び実行されて、グループ結果の最終セットが得られます。

ファイル全体のグループ化 (フィールドによるグループがない) は、常に 2 つのステップを使用して実行されます。

QUERY に、HAVING 文節または OPNQRYF コマンドのグループ選択式 (GRPSLT) パラメーターが含まれる場合、最初のグループ化ステップのすべてのグループが、コーディネーター・ノードに戻されます。HAVING 文節または GRPSLT パラメーターは、2 番目のグループ化ステップの一部として処理されます。

QUERY に DISTINCT 列 (集合) が含まれて、2 ステップのグループ化が必要な場合、最初のステップではグループ化は実行されません。かわりに、すべてのレコードがコーディネーター・ノードに戻され、すべてのグループ化が、コーディネーター・ノードにおいて 2 番目のステップの一部として実行されます。

DB2 マルチ・システムでのグループ化と結合

QUERY に結合が含まれる場合、実行可能なグループ化のタイプを決めるために使用される区分化キーは、結合を実行するために必要とされたデータの再区分化に基づいています。

次の例では、再区分化結合がグループ化の前に実行されます。これによって、MGRNO という新しい区分化キーが生成されます。MGRNO は区分化キーであるため、グループ化は 1 ステップのグループ化を使用して実行することができます。

```
SQL:      SELECT MGRNO, COUNT(*)
           FROM DEPARTMENT, EMPLOYEE
           WHERE MGRNO = EMPNO
           GROUP BY MGRNO
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT2)
           JFLD((MGRNO EMPNO *EQ))
           GRPFLD(MGRNO)
           MAPFLD((CNTMGR '%COUNT'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

次の例では、再区分化結合がグループ化の前に実行されます。これによって、EMPNO という新しい区分化キーが生成されます。EMPNO は WORKDEPT に代わる区分化キーであるため、グループ化は 1 ステップのグループ化を使用して実行することができません。


```
SQL:      SELECT WORKDEPT, COUNT(*)
           FROM DEPARTMENT, EMPLOYEE
           WHERE MGRNO = EMPNO
           GROUP BY WORKDEPT
```

```
OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT3)
           JFLD((MGRNO EMPNO *EQ))
           GRPFLD(WORKDEPT)
           MAPFLD((CNTDEPT '%COUNT'))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

DB2 マルチ・システムでの SUBQUERY サポート

分散ファイルを SUBQUERY で指定することができます。SUBQUERY は、独自の検索条件を含むことができ、これらの検索条件は逆に SUBQUERY を含むことができます。したがって、SQL ステートメントは SUBQUERY の階層を含むことができます。SUBQUERY を含む階層の要素は、それらが含む SUBQUERY よりも高いレベルにあります。SUBQUERY の使用については、

『SQL プログラミング 概念 』を参照してください。

DB2 マルチ・システムでのアクセス計画

分散ファイルを参照する QUERY に格納されたアクセス計画は、ローカル・ファイルに格納されたアクセス計画とは異なります。分散 QUERY に格納されたアクセス計画は、QUERY が複数のステップに分割される方法と QUERY の各ステップが実行されるノードに関する情報を含む分散アクセス計画です。ステップが各ノードでローカルに実施される方法についての情報は、アクセス計画には保管されません。この情報は実行時に作成されます。

DB2 マルチ・システムでの再使用可能なオープン・データ・パス (ODP)

再使用可能なオープン・データ・パス (ODP) には、分散 QUERY に関する特殊な考慮事項があります。分散 QUERY の他のほとんどと同様に、ODP には分散およびローカルの 2 つのレベルがあります。

分散 ODP は調整 ODP です。分散 ODP は、QUERY をユーザーに関連付けて、ローカル ODP を制御します。ローカル ODP は、QUERY に関与する各システムにあり、分散 ODP を介して要求をとります。

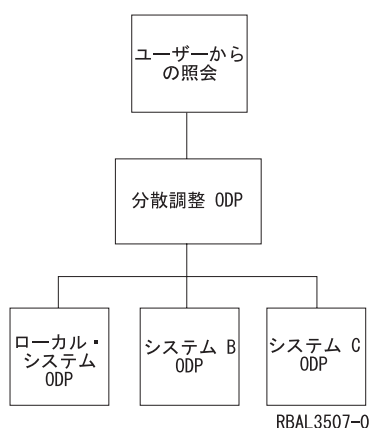


図7. オープン・データ・パス (ODP) の例

たとえば、SQL FETCH を実行するための要求が出された場合、その要求は分散 ODP に対して出されます。システムは、その要求をとって、ローカル ODP に対して適切なレコード検索を実行します。

分散 QUERY では、分散 ODP を再使用可能にして、1 つまたは複数のローカル ODP を再使用不可にすることができます。ただし、分散 QUERY ODP が再使用不可の場合は、ローカル ODP も常に再使用不可です。これは、次のことを可能にするために認められています。

- 各ローカル・システムが、そのローカル ODP をオープンする最善の方法 (再使用可能か再使用不可か) を決定することができる
- 通信などの活動資源を維持するために、ローカル ODP 方式に関係なく、分散 ODP を可能なかぎり再使用可能としてオープンする

システムは、基礎となるローカル ODP が再使用不可であっても、可能であればいつでも分散 ODP を再使用可能にしようとします。このことが起こると、システムは ODP リフレッシュを次のように処理します。

- 各ローカル ODP で反復する
- 再使用可能ローカル ODP のリフレッシュを実行する
- 『ハード』・クローズと再使用不可の ODP の再オープンを実行する
- 必要な分散 ODP 自体の残りのリフレッシュがあればリフレッシュを完了する

分散 ODP は、ローカル ODP よりも再使用可能である場合がよくあります。これは、LIKE 文節のホスト変数や、索引対索引作成を実行できるように再使用不可を選択する最適化プログラムなど、ローカル ODP を再使用不可にするものによって、分散 ODP が影響を受けないためです。ローカル ODP に影響を与えるものの一部によって、分散 ODP は再使用不可になります。これには、次のものがあります。

- 分類以外での一時ファイルの使用。これらは複数ステップの分散 QUERY と呼ばれ、最適化プログラムのデバッグ・メッセージ CPI4343 が通知されます。
- ライブラリー・リストの変更。これは、照会されるファイルに影響を与える可能性があります。
- OVRDBF の変更。これは、照会されるファイルに影響を与えます。
- 特殊レジスター USER または CURRENT TIMEZONE の値の変更。
- ジョブ CCSID の変更。
- リソースの再利用 (RCLRSC) コマンドの発行。

ローカル ODP の再使用可能性は、非分散 QUERY ODP にすでに存在する条件と同じ条件によって影響を受けます。したがって、ローカル QUERY ODP に適用されるものと同じ考慮事項が適用されます。

DB2 マルチ・システムでの一時結果書き込み機能

一時結果書き込み機能 (一時書き込み機能) は、常に活動状態にあるシステム開始ジョブです。システムでは、一時書き込み機能は QQQTEMP1 および QQQTEMP2 と呼ばれる対のジョブです。一時書き込み機能は、QUERY を実行するジョブからの要求を処理します。これらの要求は、実行すべき QUERY (QUERY ステップの) と、QUERY 結果から充てんされるシステム一時ファイルの名前から構成されます。一時書き込み機能は要求を処理して、一時ファイルを充てんします。この中間一時ファイルは、元の QUERY を完了するために要求ジョブによって使用されません。

次の例は、一時結果書き込み機能を必要とする QUERY と、QUERY を処理するために必要なステップを示しています。

```
SQL:      SELECT COUNT(*)
          FROM DEPARTMENT a, EMPLOYEE b
          WHERE a.ADMRDEPT = b.WORKDEPT
          AND b.JOB = 'Manager'

OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
          FORMAT(FMTFILE)
          MAPFLD((CNTFLD '%COUNT'))
          JFLD((1/ADMRDEPT 2/WORKDEPT))
          QRYSLT('2/JOB = 'Manager')
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

WORKDEPT は、EMPLOYEE の区分化キーですが、ADMRDEPT は DEPARTMENT の区分化キーではありません。QUERY は 2 つのステップで処理する必要がありますが、最適化プログラムは QUERY を次のステップに分割します。

```
INSERT INTO SYS_TEMP_FILE
SELECT a.DEPTNAME, a.ADMRDEPT
FROM DEPARTMENT a
```

および

```
SELECT COUNT(*) FROM SYS_TEMP_FILE x, EMPLOYEE b
WHERE x.ADMRDEPT = b.WORKDEPT AND b.JOB = 'Manager'
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

一時書き込み機能がジョブ (QUERY 属性の変更 (CHGQRYA) コマンド・オプションによって制御される) で使用できる場合、最適化プログラムは次のことを行います。

1. 一時ファイル (SYS_TEMP_FILE) をライブラリー QRECOVERY に作成します。
2. SYS_TEMP_FILE を充てんする要求を一時書き込み機能に送ります。
3. 最終 QUERY のオープンを続けて終了します (一時書き込み機能が一時ファイルを充てんする間)。
4. 最終 QUERY がオープンされたら、一時書き込み機能が一時ファイルの充てんを終了するまで待ってから、呼び出し側に制御を戻します。

このコマンドについては、62 ページの『DB2 マルチ・システムでの QUERY 属性の変更 (CHGQRYA) コマンドへの変更』を参照してください。

DB2 マルチ・システムでの一時結果書き込み機能の使用について詳しくは、下記の各セクションを参照してください。

- 59 ページの『一時書き込み機能ジョブ: DB2 マルチ・システムでの利点』
- 59 ページの『一時書き込み機能ジョブ: DB2 マルチ・システムでの不利な点』
- 60 ページの『DB2 マルチ・システムでの一時書き込み機能の制御』

一時書き込み機能ジョブ: DB2 マルチ・システムでの利点

要求の処理で一時書き込み機能ジョブを使用した場合の利点は、一時書き込み機能が、メイン・ジョブが QUERY の別のステップを処理しているのと同時に (並行して) その要求を処理できることです。

次に、一時書き込み機能を使用した場合のパフォーマンスの利点をいくつか示します。

- 一時書き込み機能は、その QUERY ステップを完了するうえで完全 SMP 並列サポートを使用することができます。これに対して、メイン・ジョブは、簡単に並列処理を行うことのない、より複雑なステップ (QUERY 最適化、索引分析など) を続けることができます。
- 分散ファイル処理には通信対話が必要であるため、通常、送受信の待機に使用されるかなりの時間を一時書き込み機能に負荷分担させて、メイン・ジョブに他の作業を実行させることができます。

一時書き込み機能ジョブ: DB2 マルチ・システムでの不利な点

一時書き込み機能には、QUERY での効用を判別するときに考慮する必要がある欠点もあります。

- 一時書き込み機能は独立したジョブです。このため、次のようなメイン・ジョブとの矛盾が生じる可能性があります。
 - メイン・ジョブが、ファイルをロックする場合があります。この場合、一時書き込み機能はファイルにアクセスできず、その QUERY ステップを完了できません。
 - メイン・ジョブは、コミットメント制御のもとで分散ファイルを作成して、作成をまだコミットしていない場合があります。この場合、一時書き込み機能はファイルにアクセスすることができません。
- 一時書き込み機能は、メイン・ジョブと同じ方法で処理できない状態に陥る場合があります。たとえば、照会メッセージが通知された場合、一時書き込み機能を取り消さなければならないことがあるのに対して、メイン・ジョブはこのメッセージを無視して処理を続けることができます。
- 一時書き込み機能は、システム上のすべてのジョブによって共用されます。いくつかのジョブが一時書き込み機能に対する要求を持つ場合、これらの要求は、書き込み機能がその処理を試みる間待機する場合があります。

注: システムは、3 つのアクティブな一時書き込み機能ジョブのペアの状態で出荷されます。

- QUERY を分析しようとする (たとえば、デバッグ・メッセージなど)、一時書き込み機能が関与する場合、処理が複雑になるおそれがあります (これは、QUERY のステップが独立したジョブとして実行されるためです)。

注: システムは、一時書き込み機能をコミットメント制御 *CS または *ALL のもとで実行される QUERY に使用することを認めません。これは、メイン・ジョブがファイル内のレコードをロックして、一時書き込み機能がこれらのレコードからロックされて、終了できないようにする可能性があるためです。

DB2 マルチ・システムでの一時書き込み機能の制御

省略時解釈では、QUERY は一時書き込み機能を使用しません。ただし、一時書き込み機能の使用は、QUERY 属性の変更 (CHGQRYA) コマンドを使用して使用可能にすることができます。

CHGQRYA コマンドの非同期ジョブの使用 (ASYN CJ) パラメーターは、一時書き込み機能の使用を制御するために使用されます。ASYN CJ パラメーターには、次のオプションがあります。

- *DIST または *ANY を使用すると、一時書き込み機能を、分散ファイルに關与する QUERY に使用することができます。
- *LOCAL または *NONE を使用すると、一時書き込み機能が分散ファイルに關与する QUERY に使用されるのを防止することができます。

DB2 マルチ・システムでの最適化プログラムのメッセージ

OS/400® 分散 QUERY 最適化プログラムを使用すると、ジョブがデバッグ・モードにあるときに、現在の QUERY 処理に関する通知メッセージを得ることができます。これらのメッセージは、分散 QUERY の処理方法を示すもので、既存の最適化プログラムのメッセージに追加されたものです。これらのメッセージは、QUERY ファイルのオープン (OPNQR YF) コマンド、DB2 UDB Query Manager and SQL Development Kit、対話式 SQL、組み込み SQL、および、iSeries サーバー高水準言語 (HLL) の場合に表示されます。すべてのメッセージは、ジョブ・ログに表示されます。ユーザーに要求されるのは、ジョブをデバッグ・モードにしておくことだけです。

分散 QUERY のパフォーマンスは、データベース・マネージャーによってジョブ・ログに入れられた通知メッセージを使用して評価することができます。データベース・マネージャーは、次の分散メッセージのいずれか、または既存の最適化プログラム・メッセージを必要に応じて送信することができます。記号変数 (&1、&X) は、次のメッセージがジョブ・ログに表示されるときのオブジェクト名または別の置換値のいずれかを含む置換変数です。

- CPI4341 分散 QUERY を実行中です。
- CPI4342 QUERY の分散結合を実行中です。
- CPI4343 分散 QUERY ステップの &2 の &1 に対する最適化プログラムのデバッグ・メッセージ。
- CPI4345 QUERY に作成された一時分散結果ファイル &4。

これらのメッセージは、分散 QUERY がどのように実行されたかに関するフィードバックを提供するとともに、場合によっては、QUERY の実行速度を高めるための手助けとしてなし得る改善点を示すこともあります。メッセージの原因とユーザーの処置は、以下の段落で示します。実際のメッセージ・ヘルプは、これよりも詳しく、各メッセージの意味と応答を判別するときに使用する必要があります。

各メッセージの詳しい説明は次のとおりです。

CPI4341

分散 QUERY を実行中です。

このメッセージは、単一分散ファイルが照会されて、複数のステップで処理されていないことを示します。このメッセージは、QUERY が実行されたファイルのノードをリスト表示します。

CPI4342

QUERY の分散結合を実行中です。

このメッセージは、分散結合が起こったことを示します。このメッセージは、結合されたファイルだけでなく、結合が実行されたノードもリスト表示します。

CPI4343

分散 QUERY ステップ &2 の &1 に対する最適化プログラムのデバッグ・メッセージ。

このメッセージは、分散 QUERY が複数のステップで処理されて、現在のステップ番号をリスト表示することを示します。このメッセージに続いて、そのステップに関するすべての最適化プログラム・メッセージが示されません。

CPI4345

QUERY に作成された一時分散結果ファイル &4。

このメッセージは、一時分散結果ファイルが作成されたことを示し、その一時ファイルがなぜ必要であったかを示す理由コードをリストします。このメッセージには、ファイルを作成するために使用された区分化キーと、一時ファイルが作成されたノードも示されます。

次の例は、分散 QUERY が処理された方法を判別するために生成された、分散最適化プログラム・メッセージを調べる方法を示しています。この例では、分散ファイル EMPLOYEE と DEPARTMENT を使用しています。

```
SQL:      SELECT A.EMPNO, B.MGRNO, C.MGRNO, D.EMPNO
          FROM EMPLOYEE A, DEPARTMENT B, DEPARTMENT C, EMPLOYEE D
          WHERE A.EMPNO=B.MGRNO
                AND B.ADMRDEPT=C.DEPTNO
                AND C.DEPTNO=D.WORKDEPT

OPNQRYF:  OPNQRYF FILE((EMPLOYEE) (DEPARTMENT) (DEPARTMENT) (EMPLOYEE))
          FORMAT(JFMT)
          JFLD((1/EMNO 2/MGRNO *EQ)
              (2/ADMRDEPT 3/DEPTNO)
              (3/DEPTNO 4/WORKDEPT))
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

分散最適化プログラム・メッセージの次のリストが生成されます。

- CPI4343 分散 QUERY ステップ 4 の 1 についての最適化プログラムのデバッグ・メッセージは、次に続いて表示されます。
 - CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0001。
ファイル B は、一時ファイル *QQTDF0001 に指定されています。
- CPI4343 分散 QUERY ステップ 4 の 2 についての最適化プログラムのデバッグ・メッセージは、次に続いて表示されます。
 - CPI4342 QUERY の分散結合を実行中です。

ファイル B、C および *QQTDF0001 が結合されました。これは、共通設定結合 (ファイル B と C の間) と指定結合 (ファイル *QQTDF0001 と) の組み合わせです。

- CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0002。
一時分散ファイル *QQTDF0002 が、ファイル B、C、および *QQTDF0001 を結合した結果を含むために作成されました。このファイルは指定されていません。
- CPI4343 分散 QUERY ステップ 4 の 3 についての最適化プログラムのデバッグ・メッセージは、次に続いて表示されます。
 - CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0003。
ファイル A は、一時ファイル *QQTDF0003 に指定されています。
- CPI4343 分散 QUERY ステップ 4 の 4 についての最適化プログラムのデバッグ・メッセージは、次に続いて表示されます。
 - CPI4342 QUERY の分散結合を実行中です。
ファイル *QQTDF0002 と *QQTDF0003 が結合されました。これは、両方のファイルが指定されてから結合が発生したため、再区分化結合でした。

パフォーマンスのために QUERY を調整するときを使用できる追加ツールには、SQL プログラムとパッケージに適用される SQL 情報の印刷 (PRTSQLINF) と、QUERY 属性の変更 (CHGQRYA) の各 CL コマンドがあります。

DB2 マルチ・システムでの QUERY 属性の変更 (CHGQRYA) コマンドへの変更

CHGQRYA コマンドには、ASYNCJ (非同期ジョブ使用) と APYRMT (リモート適用) という、分散 QUERY に適用できる 2 つのパラメーターがあります。

- 『DB2 マルチ・システムでの非同期ジョブ使用 (ASYNCJ) パラメーター』
- 63 ページの『リモート適用 (APYRMT) パラメーター』

注: 他のパラメーターとは違い、ASYNCJ と APYRMT にはシステム値がありません。省略時値以外の値が必要な場合、その値は各ジョブで変更する必要があります。

DB2 マルチ・システムでの非同期ジョブ使用 (ASYNCJ) パラメーター

ASYNCJ パラメーターを使用すると、一時書き込み機能の使用を制御することができます。ASYNCJ パラメーターには次のオプションがあります。

- *ANY — これを使用すると、一時書き込み機能ジョブを、分散ファイルを必要とするデータベース QUERY に使用することができます。
- *DIST — これを使用すると、一時書き込み機能ジョブを、分散ファイルを必要とするデータベース QUERY に使用することができます。
- *LOCAL — これを使用すると、一時書き込み機能ジョブを、ローカル・ファイルの QUERY にのみ使用することができます。このオプションは使用できませんが、一時書き込み機能をローカル QUERY 処理に使用するためのシステム・サポ

ートはありません。*LOCAL が、一時書き込み機能を分散 QUERY で使用不可にするために追加されていますが、通信を非同期で実行することもできます。

- *NONE — 一時書き込み機能をまったく使用しません。また、分散処理が実行される場合、通信は同期的に実行されます。このことは、QUERY を分析するときに非常に役立ちます。これは、リモート・システムからの QUERY デバッグ・メッセージをローカル・システムに戻すことができるからです。

次の例は、分散ファイル処理で非同期ジョブ使用を使用不可にする方法を示しています。

```
CHGQRYA ASYNCJ(*LOCAL)
```

このコマンドは、分散ファイルに關与する QUERY に非同期ジョブが使用されないようにするものです。

次の例は、非同期ジョブ使用を完全に使用不可にする方法を示しています。

```
CHGQRYA ASYNCJ(*NONE)
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

このコマンドは、すべての QUERY に非同期ジョブが使用されないようにするものです。また、分散ファイルに關与する QUERY では、リモート・システムへの通信が同期形式で実行されます。

次の例は、CHGQRYA コマンドをデバッグ開始 (STRDBG) コマンドと組み合わせて使用して、分散 QUERY を分析する方法を示しています。

```
STRDBG UPDPROD(*YES)
CHGQRYA ASYNCJ(*NONE)
STRSQL
      SELECT COUNT(*) FROM EMPLOYEE A
```

次のデバッグ・メッセージは、ジョブ・ログに入れられます。

```
Current connection is to relational database SYSA.
DDM job started.
Optimizer debug messages for distributed query step 1 of 2 follow:
Temporary distributed result file *QTDF0001 built for query.
Following messages created on target system SYSB.
Arrival sequence access was used for file EMPLOYEE.
Arrival sequence access was used for file EMPLOYEE.
Optimizer debug messages for distributed query step 2 of 2 follow:
Arrival sequence access was used for file EMPLOYEE.
ODP created.
Blocking used for query.
```

コードに関する特記事項については、v ページの『コードについての特記事項』を参照してください。

リモート適用 (APYRMT) パラメーター

APYRMT パラメーターを使用すると、他の CHGQRYA オプションを、分散 QUERY 要求の処理で使用される関連のリモート・システム・ジョブに適用するかどうかを指定することができます。

- *YES — CHGQRYA オプションをリモート・ジョブに適用します。これには、リモート・ジョブが CHGQRYA コマンドを使用するための権限が必要です。権限がないと、リモート・ジョブにエラーが通知されます。
- *NO — CHGQRYA オプションをローカルにのみ適用します。

次の例は、CHGQRYA オプションがリモートに適用されないようにするものです。

```
CHGQRYA DEGREE(*NONE) APYRMT(*NO)
```

この場合、SMP 並列処理がコーディネーター・ノードで防止されますが、リモート・システムは独自の並列度を選択することができます。

これらのパラメーターだけでなく、時間制限の QUERY (QRYTIMLMT) がどのように作動するかも注意する必要があります。パラメーターに指定された時間制限は、分散 QUERY の各ステップ (ローカルおよびリモート) に適用されます。これは、QUERY 全体には適用されません。したがって、ある QUERY ステップが時間制限に到達しても、別のステップは問題なく続けられるという場合もあります。時間制限オプションは非常に役に立ちますが、分散 QUERY では注意して使用する必要があります。

パフォーマンスの考慮事項の要約

分散ファイルを使用する QUERY を開発する場合は、次のことについて考慮する必要があります。



1. OPNQRYF コマンドと QUERY API (QQQRY) の場合、ALWCPYDTA(*OPTIMIZE) を指定すると、各ノードで索引または分類を選択して、指定の順序に対応することができます。
2. OPNQRYF コマンドおよび QUERY API (QQQRY) の場合、ALWCPYDTA(*YES) または ALWCPYDTA(*NO) を指定すると、各ノードは、強制的に、指定の順序フィールドに合致した索引を使用させられます。これは、最適化プログラムが非分散ファイルの順序を処理する場合よりも高い拘束性があります。
3. ORDER BY 文節を DISTINCT 選択に追加すると、要求システムで最終分類が不要になるため、レコードを戻す速度を上げることができます。
4. グループ化フィールドに区分化キーのフィールドすべてを含めると、通常、1 ステップのグループ化が起こります。このパフォーマンスは、2 ステップのグループ化よりも改善されます。
5. 区分化キーのすべてのフィールドを結合基準に含めると、通常、共通設定分散結合が起こります。
6. 区分化キーのすべてのフィールドを、分離可能な等しいレコード選択に含めると、通常、1 つのノードのみで処理される QUERY が起こります。
7. 次のスカラー関数のいずれかを、分離可能な等しいレコード選択に含めると、通常、QUERY は 1 つのノードのみで処理されます。
 - NODENAME
 - NODENUMBER
 - PARTITION

参考文献

ここでは、本書で引用されている資料をリストアップしています。


- バックアップおよび回復のトピックには、設定と管理に関する情報が示されています。
 - ジャーナル処理、アクセス・パス保護、およびコミットメント制御
 - ユーザー補助記憶域プール (ASP)
 - ディスク保護 (デバイス・パリティ、ミラー化、およびチェックサム)


この資料は、バックアップ・メディアおよび保管・復元操作に関するパフォーマンス情報を提供しています。また、活動状態の間の保存サポートの使用、保存と別のリリースへの復元、およびプログラミングのヒントと技法などの拡張バックアップおよび回復のトピックについても説明しています。

- Information Center のトピック『制御言語』は、iSeries サーバー制御言語 (CL) とその OS/400 コマンドについて説明しています。(非 OS/400 コマンドについては、それぞれのライセンス・プログラムの資料で説明しています。) また、iSeries サーバーのすべての CL コマンドの概要を示して、それらをコーディングするために必要な構文の規則についても説明しています。
- Information Center のトピック『データベース・プログラミング 』では、システム上でデータベース・ファイルを作成、記述、および更新する方法も含めて、iSeries サーバー・データベース編成の詳しい説明を行っています。また、OS/400 データ記述仕様 (DDS) キーワードを用いてファイルをシステムへ定義する方法についても説明しています。
- Information Center のトピック『SQL プログラミング 概念 』には、DB2 QUERY マネージャー (AS/400 版) および SQL 開発キット・ライセンス・プログラムを使用する方法を示してあります。また、データベース・ライブラリー内のデータにアクセスする方法と、組み込み SQL ステートメントを含むアプリケーション・プログラムを作成、実行、およびテストする方

法も示されています。SQL/400 ステートメントの例と、対話式 SQL 関数の説明が記載されています。SQL/400 ステートメントを COBOL/400、ILE COBOL/400、PL/I、ILE C/400、FORTRAN/400、RPG/400、ILE RPG/400、および REXX で使用する際の共通概念と規則についても説明しています。

- Information Center のトピック

『SQL 解説書 』には、構造化照会言語 /400 DB2/400 ステートメントの使用方法に関する説明と、これらのステートメントの正しい使用に関する詳細が記載されています。ステートメントの例には、構文図、パラメーター、および定義が示されています。SQL 限界のリストと SQL 連絡域 (SQLCA) および SQL 記述子域 (SQLDA) の説明も示されています。

- Information Center のトピック『分散データベース・プログラミング 』には、分散リレーショナル・データベース・アーキテクチャー (DRDA) を使用する分散リレーショナル・データベースにおいて iSeries サーバーを準備し管理する方法が記載されています。この資料では、類似のシステム環境における複数のサーバーで、分散リレーショナル・データベースを計画、設定、プログラミング、管理、および操作する方法について説明しています。

- *OptiConnect for OS/400*, SD88-5053-03 は、光ファイバー・ケーブルを使用して複数のサーバーと接続できる OptiConnect サポートについて説明しています。OptiConnect を使用すると、システム間データベースへのアクセス速度が向上するだけでなく、別のサーバーに作業を負荷分散させることができます。また、構成、導入、および操作に関する情報も示されています。

- ソフトウェアの導入, SD88-5002-06 には、オペレーティング・システムとライセンス・プログラムを導入するための下記の手順に関するプランニング情報とステップ単位の指示が記載されています。

- 初期導入

- 導入済みリリースと新しいリリースの置き換え
- 追加ライセンス・プログラムの追加
- 2 次言語の追加
- システムの 1 次言語の変更
- Information Center のトピック『OS/400 API 解説書』は、熟練したプログラマーを対象として、アプリケーション・プログラミング・インターフェース (API) を下記のような OS/400 機能に対して使用する方法を示しています。
 - 動的画面管理機能
 - ファイル (データベース・ファイル、スプール・ファイル、階層ファイル)
 - ジャーナルおよびコミットメント制御
 - メッセージ処理
 - 各国語サポート
 - ネットワーク管理
 - オブジェクト
 - 問題管理
 - 登録機能
 - セキュリティー
 - ソフトウェア・プロダクト
 - ソース・デバッグ
 - UNIX**-タイプ
 - ユーザー定義の通信
 - ユーザー・インターフェース
 - 実行管理機能

オリジナル・プログラム・モデル (OPM)、統合言語環境 (ILE)、および UNIX タイプ API も含まれます。

DB2 UDB 対称マルチプロセッシングについては、Information Center のトピック『SQL プログラミング 概念』と、『データベース・プログラミング』を参照してください。

索引

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

[ア行]

アクセス計画 56
一時書き込み機能 57
 使用上の不利な点 59
 使用上の利点 59
 制御 60
一時結果書き込み機能 57
一時結果ファイル
 定義 44
オープン・データ・パス (ODP)
 再使用可能 56
 図 56
オブジェクト記述の表示 (DSPOBJD) コマンド
 ローカル・ファイル部分にのみ影響 21
オブジェクト権限の取り消し (RVKOBJAUT) コマンド 22
オブジェクト権限の付与 (GRTOBJAUT) コマンド 22
オブジェクト所有者の変更 (CHGOBJOWN) コマンド 22
オブジェクトの移動 (MOV OBJ) コマンド
 分散ファイルに認められない 20
オブジェクトのダンプ (DMPOBJ) コマンド
 ローカル・ファイル部分にのみ影響 21
オブジェクトの名前変更 (RNMOBJ) コマンド 22
オブジェクトの復元 (RSTOBJ) コマンド
 ローカル・ファイル部分にのみ影響 21
オブジェクトの保管 (SAVOBJ) コマンド
 ローカル・ファイル部分にのみ影響 21
オブジェクトの割り振り (ALCOBJ) コマンド 22
オブジェクトの割り振り解除 (DLCOBJ) コマンド 22

[カ行]

会話、保護 37
開始レコード (FROMRCD) パラメーター 21
概要
 最適化 44
 ノード・グループ 7
書き込み機能、一時 57
書き込み機能、一時結果 57
可視ノード
 作成方法 9
 使用例 9
 定義 9
関数
 SQL
 相対レコード番号 (RRN) 35
 特殊レジスター 34
関連の印刷情報 65
キー、区分化
 定義 3
共通設定結合 48
 例 48
区画互換
 データ・タイプ 48
 定義 48
区画番号 7
 定義 3
区画番号とノード番号の相関図 11
区画マップ
 図 3
 定義 3
区分化 7
 キー 25
 計画 26
 省略時の区分化 7
 設定の考慮事項 26
 定義 3
 番号 25
 マップ 25
 SMP の利点 26
区分化キー
 結合フィールドに対する 53
 選択 27
 定義 3
 ヌル可能フィールド 25
区分化キー (PTNKEY) パラメーター 15
区分化属性、変更
 例 13
区分化ファイル
 定義 3

区分化ファイル (続き)
 ファイル定義 9
区分化ファイルの例
 図 9
組み合わせ
 レコード順序の最適化 46
グループ化
 結合 55
 最適化 53
 実施 53
グループ化、1 ステップ
 最適化 54
グループ化、2 ステップ
 最適化 54
計画、アクセス 56
結合
 共通設定 48
 例 48
 グループ化 55
 再区分化 50
 例 50
 最適化 48, 52
 実施 48
 指定 50
 例 50
 同報通信 52
 例 52
結合フィールド
 区分化キー 53
権限エラー 22
権限の変更、ノード・グループでの
 矛盾 22
コーディネーター・ノード
 定義 34
コピー (COPY) コマンド
 分散ファイルに認められない 20
コマンド、CL
 オブジェクト記述の表示 (DSPOBJD)
 ローカル・ファイル部分にのみ影響 21
 オブジェクト権限の取り消し (RVKOBJAUT) 22
 オブジェクト権限の付与 (GRTOBJAUT) 22
 オブジェクト所有者の変更 (CHGOBJOWN) 22
 オブジェクトの移動 (MOV OBJ)
 分散ファイルに認められない 20
 オブジェクトのダンプ (DMPOBJ)
 ローカル・ファイル部分にのみ影響 21

コマンド、CL (続き)

オブジェクトの名前変更
(RNMOBJ) 22

オブジェクトの復元 (RSTOBJ)
ローカル・ファイル部分にのみ影響
21

オブジェクトの保管 (SAVOBJ)
ローカル・ファイル部分にのみ影響
21

オブジェクトの割り振り
(ALCOBJ) 22

オブジェクトの割り振り解除
(DLCOBJ) 22

コピー (COPY) コマンド
分散ファイルに認められない 20

ジャーナル処理済み変更適用
(APYJRNCHG)
ローカル・ファイル部分にのみ影響
21

ジャーナル処理済み変更の除去
(RMVJRNCHG)
ローカル・ファイル部分にのみ影響
21

ジャーナル・アクセス・パスの開始
(STRJRNAP)
ローカル・ファイル部分にのみ影響
21

ジャーナル・アクセス・パスの終了
(ENDJRNAP)
ローカル・ファイル部分にのみ影響
21

データベース・ファイルの位置決め
(POSDBF)
分散ファイルに認められない 20

ノード・グループ属性の変更
(CHGNODGRPA) 7, 13

ノード・グループの削除
(DLTNODGRP) コマンド 14

ノード・グループの作成
(CRTNODGRP) 9
データの区分化の変更 7

ノード・グループの表示
(DSPNODGRP) 11

ファイル記述の表示 (DSPFD) 21

ファイルのコピー (CPYF) 22, 24

ファイルの削除 (DLTF) 22

複写オブジェクトの作成
(CRTDUPOBJ)
分散ファイルに認められない 20

物理ファイル制約の除去
(RMVPCST) 22

物理ファイル制約の追加
(ADDFCST) 22

物理ファイル制約の変更
(CHGPCST) 22

物理ファイルの作成 (CRTPF) 9, 15

コマンド、CL (続き)

物理ファイルの変更 (CHGPF) 15, 22
データ分散のカスタマイズ 28

物理ファイル変更ジャーナルの終了
(ENDJRNPF) 22

物理ファイル・ジャーナルの開始
(STRJRNPF) 22

物理ファイル・トリガーの除去
(RMVPCST) 22

物理ファイル・トリガーの追加
(ADDFCST) 22

物理ファイル・メンバーのクリア
(CLRPFM) 22

物理ファイル・メンバーの再編成
(RGZPFM) 22

物理ファイル・メンバーの初期化
(INZPFM)
分散ファイルに認められない 20

物理ファイル・メンバーの追加
(ADDFM) 22

物理ファイル・メンバーの表示
(DSPPFM) 21

物理ファイル・メンバーの変更
(CHGPCM)
分散ファイルに認められない
SHARE パラメーター 20

分散ファイルの制限 20

メンバーの除去 (RMVM)
分散ファイルに認められない 20

メンバーの名前変更 (RNMM)
分散ファイルに認められない 20

ライブラリーの名前変更 (RNMLIB)
コマンド
分散ファイルに認められない 20

リスト
分散ファイルで実行できないコマン
ド 20

分散ファイルのすべての部分に影響
を与えるコマンド 22

分散ファイルのローカル部分にのみ
影響を与えるコマンド 21

論理ファイルの作成 (CRTLF) 22

論理ファイルの変更 (CHGLF) 22

論理ファイル・メンバーの追加
(ADDFM) 22

論理ファイル・メンバーの変更
(CHGLFM)
分散ファイルに認められない
SHARE パラメーター 20

ADDFM (論理ファイル・メンバーの
追加) 22

ADDFCST (物理ファイル制約の追
加) 22

ADDFM (物理ファイル・メンバーの
追加) 22

コマンド、CL (続き)

ADDFTRG (物理ファイル・トリガー
の追加) 22

ADDRDBDIRE (RDB ディレクトリー
項目の追加) 8

ALCOBJ (オブジェクトの割り振
り) 22

APYJRNCHG (ジャーナル処理済み変
更適用)
ローカル・ファイル部分にのみ影響
21

CHGLF (論理ファイルの変更) 22

CHGLFM (論理ファイル・メンバーの
変更)
分散ファイルに認められない
SHARE パラメーター 20

CHGNODGRPA (ノード・グループ属
性の変更) 7, 13

CHGOBJOWN (オブジェクト所有者の
変更) 22

CHGPF (物理ファイルの変更) 15, 22
データ分散のカスタマイズ 28

CHGPCST (物理ファイル制約の変
更) 22

CHGPCM (物理ファイル・メンバーの
変更)
分散ファイルに認められない
SHARE パラメーター 20

CHGQRYA (QUERY 属性の変更)
変更 62

CLRPFM (物理ファイル・メンバーの
クリア) 22

COPY (コピー) コマンド
分散ファイルに認められない 20

CPYF (ファイルのコピー) 22, 24

CRTDUPOBJ (複写オブジェクトの作
成)
分散ファイルに認められない 20

CRTLF (論理ファイルの作成) 22

CRTNODGRP (ノード・グループの作
成) 9
データの区分化の変更 7

CRTPF (物理ファイルの作成) 9, 15

DLCOBJ (オブジェクトの割り振り解
除) 22

DLTF (ファイルの削除) 22

DLTNODGRP (ノード・グループの削
除) コマンド 14

DMPOBJ (オブジェクトのダンプ)
ローカル・ファイル部分にのみ影響
21

DSPFD (ファイル記述の表示) 21

DSPNODGRP (ノード・グループの表
示) 11

コマンド、CL (続き)

DSPOBJD (オブジェクト記述の表示)
ローカル・ファイル部分にのみ影響
21

DSPPFM (物理ファイル・メンバーの表示) 21

ENDJRNAP (ジャーナル・アクセス・パスの終了)
ローカル・ファイル部分にのみ影響
21

ENDJRNPF (物理ファイル変更ジャーナルの終了) 22

GRTOBJAUT (オブジェクト権限の付与) 22

INZPFM (物理ファイル・メンバーの初期化)
分散ファイルに認められない 20

MOV OBJ (オブジェクトの移動)
分散ファイルに認められない 20

OPNQRYF (QUERY ファイルのオープン) 46

POSDBF (データベース・ファイルの位置決め)
分散ファイルに認められない 20

QUERY 属性の変更 (CHGQRYA)
変更 62

QUERY ファイルのオープン (OPNQRYF) 46

RDB ディレクトリー項目の処理 (WRKRDBDIRE) コマンド 8

RDB ディレクトリー項目の追加 (ADDRDBDIRE) 8

RGZPFM (物理ファイル・メンバーの再編成) 22

RMVJRNCHG (ジャーナル処理済み変更の除去)
ローカル・ファイル部分にのみ影響
21

RMV M (メンバーの除去)
分散ファイルに認められない 20

RMV PFCST (物理ファイル制約の除去) 22

RMV PFRG (物理ファイル・トリガーの除去) 22

RNMLIB (ライブラリーの名前変更) コマンド
分散ファイルに認められない 20

RNMM (メンバーの名前変更)
分散ファイルに認められない 20

RNMOBJ (オブジェクトの名前変更) 22

RSTOBJ (オブジェクトの復元)
ローカル・ファイル部分にのみ影響
21

RVKOBJAUT (オブジェクト権限の取り消し) 22

コマンド、CL (続き)

SAVOBJ (オブジェクトの保管)
ローカル・ファイル部分にのみ影響
21

STRJRNAP (ジャーナル・アクセス・パスの開始)
ローカル・ファイル部分にのみ影響
21

STRJRNPF (物理ファイル・ジャーナルの開始) 22

WRKRDBDIRE (RDB ディレクトリー項目の処理) コマンド 8

コミットメント制御 37

[サ行]

再区分化結合 50
例 50

再使用可能なオープン・データ・パス (ODP) 56

最適化
概要 44
グループ化 53
結合 48, 52
単一ファイル QUERY 44
分散 QUERY の 44
レコードの順序 46
組み合わせ 46
分類 46
1 ステップのグループ化 54
2 ステップのグループ化 54
DISTINCT 文節 47
UNION 文節 47

最適化プログラム
メッセージ 60

再分配の問題
システムをネットワークに追加する場合 40

参考資料 65

システムのネットワークへの追加
再分配の問題 40

システム・オブジェクト
*NODGRP 7

指定結合 50
例 50

ジャーナル処理
DB2 マルチ・システムの考慮事項 23

ジャーナル処理済み変更適用 (APYJRNCHG) コマンド
ローカル・ファイル部分にのみ影響
21

ジャーナル処理済み変更の除去 (RMVJRNCHG) コマンド
ローカル・ファイル部分にのみ影響
21

ジャーナル・アクセス・パスの開始 (STRJRNAP) コマンド
ローカル・ファイル部分にのみ影響
21

ジャーナル・アクセス・パスの終了 (ENDJRNAP) コマンド
ローカル・ファイル部分にのみ影響
21

順序、レコードの最適化 46
組み合わせ 46
分類 46

処理
データのコピー許可 (ALWCPYDTA) パラメーター 47
データの配布 (DSTDTA) パラメーター 47
ALWCPYDTA (データのコピー許可) パラメーター 47
DSTDTA (データの配布) パラメーター 47

省略時の区分化 7

図 11
オープン・データ・パス (ODP) 56
区画マップ 3
区分化ファイル 9
データベースとノード番号の相関 11
データベース・ファイルの分散 1
ノード・グループ 3

スカラー関数 31
QUERY ファイルのオープン (OPNQRYF) CL コマンド
HASH 32
NODENAME 33
NODENUMBER 34
PARTITION 31

SQL
HASH 32
NODENAME 33
NODENUMBER 34
PARTITION 31

スケラビリティ 37

制御
一時書き込み機能 60

制限
分散ファイル 17
分散ファイルで実行できない CL コマンド 20
分散ファイルのすべての部分に影響を与える CL コマンド 22
分散ファイルのローカル部分にのみ影響を与える CL コマンド 21
相対レコード番号 (RRN) 関数 35

[タ行]

対称マルチプロセッシング (SMP) 37
単一ファイル QUERY
最適化 44
実施 44
追加、ネットワークへのシステムの
再分配の問題 40
データ区分化属性の変更
例 13
データのコピー許可 (ALWCPYDTA) パラ
メーター 46
処理 47
データ分散
カスタマイズ 28
物理ファイルの変更 (CHGPF) コマン
ド 28
データベースとノード番号の相関図 11
データベース・システム
DB2 マルチ・システムを使用してシス
テムを向上させる方法 39
データベース・ファイル
複数システムに分散されている図 1
分散ファイルとしての作成 15
データベース・ファイルの位置決め
(POSDBF) コマンド
分散ファイルに認められない 20
データ保全性 37
データ・タイプ
区画互換の 48
データ・ノード 9
定義
一時結果ファイル 44
可視ノード 9
区画互換 48
区画番号 3
区画マップ 3
区分化 3
区分化キー 3
区分化ファイル 3
コーディネーター・ノード 34
何も共用しないクラスター 1
ノード 3
ノード・グループ 3
ノード・グループ・オブジェクト 3
ハッシュ 3
分散ファイル 3
保護会話 37
同報通信結合 52
例 52
特殊レジスター関数
SQL の 34

[ナ行]

何も共用しないクラスター 1
定義 1
ヌル可能フィールド
区分化キーの 25
ネットワークへのシステムの追加 40
ノード
可視性
定義 9
データ 9
定義 3
表示 11
ノード・グループ
概要 7
権限の変更による矛盾 22
削除 14
例 14
作成 9
特定の区分化を使用 9
例 9
省略時の区分化 7
図 3
定義 3
表示 11
情報 21
例 11
変更 7
データ区分化属性の例 13
*NODGRP システム・オブジェクト
7
ノード・グループ (NODGRP) パラメータ
ー 9, 15
ノード・グループ属性の変更
(CHGNODGRPA) コマンド 7, 13
ノード・グループの削除 (DLTNODGRP)
コマンド
使用 14
ノード・グループの作成 (CRTNODGRP)
コマンド
使用 9
データの区分化の変更 7
ノード・グループの表示 (DSPNODGRP)
コマンド
使用 11
ノード・グループの表示画面
区画番号とノード番号の相関図 11
データベースとノード番号の相関図
11
ノード・グループ・オブジェクト
定義 3

ノード・グループ・コマンド
使用の前に完了しておくべきタスク 8

[ハ行]

ハッシュ
定義 3
ハッシュ・アルゴリズム 7
パフォーマンス 37
向上 37
例 37
対称マルチプロセッシング (SMP) 拡
張 37
QUERY 43
QUERY 処理の改善 37
QUERY の要約 64
番号
区画
定義 3
非同期ジョブ使用 (ASYN CJ) パラメータ
ー 62
ファイル
区分化
定義 3
ファイル記述の表示 (DSPFD) コマンド
21
ファイルの区分化 (PTNFILE) パラメータ
ー 7, 9
ファイルのコピー (CPYF) コマンド 22
分散ファイルでの使用 24
ファイルの削除 (DLTF) コマンド 22
複写オブジェクトの作成 (CRTDUPOBJ)
コマンド
分散ファイルに認められない 20
物理ファイル制約の除去 (RMVFCST) コ
マンド 22
物理ファイル制約の追加 (ADDPFCST) コ
マンド 22
物理ファイル制約の変更 (CHGFCST) コ
マンド 22
物理ファイルの作成 (CRTPF) コマンド
9, 15
物理ファイルの変更 (CHGPF) コマンド
15, 22
データ分散のカスタマイズ 28
物理ファイル変更ジャーナルの終了
(ENDJRNPF) コマンド 22
物理ファイル・ジャーナルの開始
(STRJRNPF) コマンド 22
物理ファイル・トリガーの除去
(RMVPTRG) コマンド 22
物理ファイル・トリガーの追加
(ADDPFTRG) コマンド 22
物理ファイル・メンバーのクリア
(CLRPFM) コマンド 22

物理ファイル・メンバーの再編成 (RGZPFM) コマンド 22

物理ファイル・メンバーの初期化 (INZPFM) コマンド
分散ファイルに認められない 20

物理ファイル・メンバーの追加 (ADDPFM) コマンド 22

物理ファイル・メンバーの表示 (DSPPFM) コマンド 21

物理ファイル・メンバーの変更 (CHGPFM) コマンド
分散ファイルに認められない SHARE パラメーター 20

分散 QUERY
一時結果ファイル 44
最適化の概要 44

分散ファイル
概要 15, 18
作成 15
制限 17
定義 3
表示
リモート・データ 21
ローカル・レコード 21
ファイルのコピー (CPYF) コマンドの使用 24

分散物理ファイル
作成
使用、物理ファイルの作成 (CRTPF) コマンドの 15
使用、SQL CREATE TABLE ステートメントの 15
例 15

分散リレーショナル・データベース・ネットワーク
設定 8
ローカル (*LOCAL) システム 8
ローカル (*LOCAL) システムの定義 8

保護会話
コミットメント制御を使用 37
定義 37

[マ行]

マップ、区画
図 3
定義 3

メッセージ
最適化プログラム 60

メンバーの除去 (RMVM) コマンド
分散ファイルに認められない 20

メンバーの名前変更 (RNMM) コマンド
分散ファイルに認められない 20

[ラ行]

ライブラリーの名前変更 (RNMLIB) コマンド
分散ファイルに認められない 20

利点
DB2 マルチ・システム 3

リモート適用 (APYRMT) パラメーター 63

リモート・データ
表示 21

リモート・ロケーション名 (RMTLOCNAME) パラメーター 8

例
可視ノードの使用 9
共通設定結合 48
共通設定結合 QUERY 48
再区分化結合 50
再区分化結合 QUERY 50
指定結合 50
指定結合 QUERY 50
データ区分化属性の変更 13
同報通信結合 52
同報通信結合 QUERY 52
特定の区分化を使用したノード・グループの作成 9
ノード・グループ
削除 14
作成 9
省略時の区分化による作成 9
表示 11
パフォーマンスの改善 37
分散物理ファイルの作成 15
CREATE TABLE ステートメント、ノード・グループを指定するために使用される 15

レコードの順序
組み合わせの最適化 46
最適化 46
実施 46
分類の最適化 46

ローカル (*LOCAL) システム
分散リレーショナル・データベース・ネットワークでの定義 8

ローカル・レコード
表示 21

論理ファイルの作成 (CRTLF) コマンド 22

論理ファイルの変更 (CHGLF) コマンド 22

論理ファイル・メンバーの追加 (ADDLFM) コマンド 22

論理ファイル・メンバーの変更 (CHGLFM) コマンド
分散ファイルに認められない SHARE パラメーター 20

[数字]

1 ステップのグループ化 54
2 ステップのグループ化 54
2 フェーズのコミット・プロトコル 37

A

ADDLFM (論理ファイル・メンバーの追加) コマンド 22

ADDPFCST (物理ファイル制約の追加) コマンド 22

ADDPFM (物理ファイル・メンバーの追加) コマンド 22

ADDPFTRG (物理ファイル・トリガーの追加) コマンド 22

ADDRDBDIRE (RDB ディレクトリー項目の追加) コマンド 8

ALCOBJ (オブジェクトの割り振り) コマンド 22

ALWCPYDTA (データのコピー許可) パラメーター 46
処理 47

APYJRNCHG (ジャーナル処理済み変更適用) コマンド
ローカル・ファイル部分にのみ影響 21

ASYN CJ (非同期ジョブ使用) パラメーター 62

C

CHGLF (論理ファイルの変更) コマンド 22

CHGLFM (論理ファイル・メンバーの変更) コマンド
分散ファイルに認められない SHARE パラメーター 20

CHGNODGRPA (ノード・グループ属性の変更) コマンド 7, 13

CHGOBJOWN (オブジェクト所有者の変更) コマンド 22

CHGPF (物理ファイルの変更) コマンド 15, 22
データ分散のカスタマイズ 28

CHGPFCSST (物理ファイル制約の変更) コマンド 22

CHGPFM (物理ファイル・メンバーの変更) コマンド
分散ファイルに認められない SHARE パラメーター 20

CHGQRYA (QUERY 属性の変更) コマンド
変更 62

CLRPFM (物理ファイル・メンバーのクリア) コマンド 22

COPY (コピー) コマンド
分散ファイルに認められない 20
CPYF (ファイルのコピー) コマンド 22
分散ファイルでの使用 24
CREATE TABLE
ノード・グループの例 15
CRTDUPOBJ (複写オブジェクトの作成)
コマンド
分散ファイルに認められない 20
CRTLF (論理ファイルの作成) コマンド
22
CRTNODGRP (ノード・グループの作成)
コマンド
使用 9
データの区分化の変更 7
CRTPF (物理ファイルの作成) コマンド
9, 15

D

DB2 マルチ・システム
概要 1
ジャーナル処理の考慮事項 23
導入 1
利点 3
DISTINCT 文節
最適化 47
実施 47
DLCOBJ (オブジェクトの割り振り解除)
コマンド 22
DLTF (ファイルの削除) コマンド 22
DLTNODGRP (ノード・グループの削除)
コマンド
使用 14
DMPOBJ (オブジェクトのダンプ) コマン
ド
ローカル・ファイル部分にのみ影響
21
DSPFD (ファイル記述の表示) コマンド
21
DSPNODGRP (ノード・グループの表示)
コマンド
使用 11
DSPOBJD (オブジェクト記述の表示) コマ
ンド
ローカル・ファイル部分にのみ影響
21
DSPPFM (物理ファイル・メンバーの表示)
コマンド 21

E

ENDJRNAP (ジャーナル・アクセス・パス
の終了) コマンド
ローカル・ファイル部分にのみ影響
21
ENDJRNPF (物理ファイル変更ジャーナル
の終了) コマンド 22

F

FROMRCD (開始レコード) パラメーター
21

G

GRTOBJAUT (オブジェクト権限の付与)
コマンド 22

H

HASH スカラー関数
QUERY ファイルのオープン
(OPNQRYF) CL コマンド 32
SQL の 32

I

INZPFM (物理ファイル・メンバーの初期
化) コマンド
分散ファイルに認められない 20

M

MOV OBJ (オブジェクトの移動) コマンド
分散ファイルに認められない 20

N

NODENAME スカラー関数
QUERY ファイルのオープン
(OPNQRYF) CL コマンド 33
SQL の 33
NODENUMBER スカラー関数
QUERY ファイルのオープン
(OPNQRYF) CL コマンド 34
SQL の 34
NODGRP (ノード・グループ) パラメータ
ー 9, 15

O

ODP (オープン・データ・パス)
再使用可能 56

ODP (オープン・データ・パス) (続き)
図 56
分散 56
ローカル 56
OPNQRYF (QUERY ファイルのオープン)
コマンド 46
スカラー関数 31
OptiConnect
接続方式 3

P

PARTITION スカラー関数
QUERY ファイルのオープン
(OPNQRYF) CL コマンド 31
SQL の 31
POSDBF (データベース・ファイルの位置
決め) コマンド
分散ファイルに認められない 20
PTNFILE (ファイルの区分化) パラメータ
ー 7, 9
PTNKEY (区分化キー) パラメーター 15

Q

QUERY
一時結果ファイル 44
共通設定結合 48
例 48
結合の最適化 48
結合の実施 48
再区分化結合 50
例 50
最適化の概要 44
指定結合 50
例 50
同報通信結合 52
例 52
パフォーマンス 43
パフォーマンスの改善 37
パフォーマンスの考慮事項の要約 64
QUERY 最適化プログラム 37
QUERY 属性の変更 (CHGQRYA) コマン
ド
変更 62
QUERY ファイルのオープン (OPNQRYF)
コマンド 46
スカラー関数 31
QUERY、単一ファイル
最適化 44
実施 44

R

RDB ディレクトリー項目の処理
(WRKRDBDIRE) コマンド 8

RDB ディレクトリー項目の追加
(ADDRDBDIRE) コマンド 8

RGZPFM (物理ファイル・メンバーの再編
成) コマンド 22

RMTLOCNAME (リモート・ロケーション
名) パラメーター 8

RMVJRNCHG (ジャーナル処理済み変更の
除去) コマンド
ローカル・ファイル部分にのみ影響
21

RMVM (メンバーの除去) コマンド
分散ファイルに認められない 20

RMVPCST (物理ファイル制約の除去) コ
マンド 22

RMVPFTRG (物理ファイル・トリガーの
除去) コマンド 22

RNMLIB (ライブラリーの名前変更) コマ
ンド
分散ファイルに認められない 20

RNMM (メンバーの名前変更) コマンド
分散ファイルに認められない 20

RNMOBJ (オブジェクトの名前変更) コマ
ンド 22

RRN (相対レコード番号) 関数 35

RSTOBJ (オブジェクトの復元) コマンド
ローカル・ファイル部分にのみ影響
21

RVKOBJAUT (オブジェクト権限の取り消
し) コマンド 22

S

SAVOBJ (オブジェクトの保管) コマンド
ローカル・ファイル部分にのみ影響
21

SMP (対称マルチプロセッシング) 37

SQL
スカラー関数 31

SQL 開始 (STRSQL) コマンド 46

STRJRNAP (ジャーナル・アクセス・パス
の開始) コマンド
ローカル・ファイル部分にのみ影響
21

STRJRNPF (物理ファイル・ジャーナルの
開始) コマンド 22

STRSQL (SQL 開始) コマンド 46

SUBQUERY
説明 56

U

UNION 文節
最適化 47
実施 47

W

WRKRDBDIRE (RDB ディレクトリー項
目の処理) コマンド 8

[特殊文字]

(APYRMT) リモート適用パラメーター
63

*NODGRP システム・オブジェクト 7



Printed in Japan