IBM DB2 Alphablox

DB2 Alphablox Cube Server 管理者用ガイド

バージョン 8.3

SD88-6489-01
(英文原典：SC18-9433-01)

IBM DB2 Alphablox

DB2 Alphablox Cube Server 管理者用ガイド

バージョン 8.3

SD88-6489-01
(英文原典：SC18-9433-01)

ご注意！

本書および本書で紹介する製品をご使用になる前に、 49 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM DB2 Alphablox for Linux, UNIX and Windows (製品番号 5724-L14) バージョン 8 リリース 3 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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)

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

原　典： SC18–9433–01
IBM DB2 Alphablox
DB2 Alphablox Cube Server
Administrator's Guide
Version 8.3

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.8

この文書では、平成明朝体™W3、平成明朝体™W7、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™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

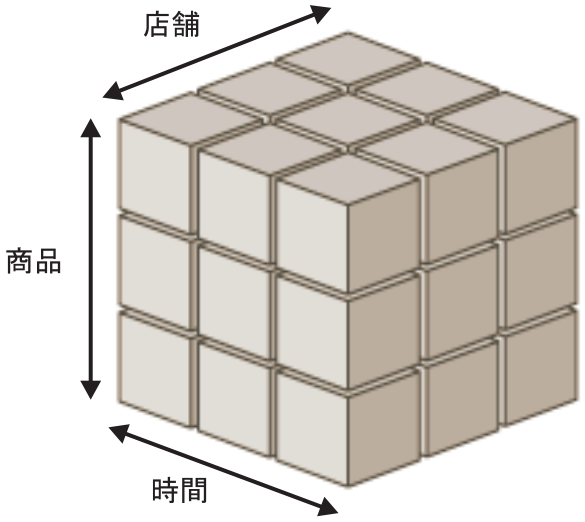
第 1 章 キューブの概念

IBM DB2 Alphablox for Linux, UNIX and Windows には DB2 Alphablox Cube Server が含まれます。 DB2 Alphablox Cube Server は、リレーショナル・データベースに格納されたデータをマルチディメンショナルで表示するビューを提供します。この章では DB2 Alphablox Cube Server についてご紹介し、どんなアプリケーションがこれを利用できるか、および使用上の要件について説明します。

概要

DB2 Alphablox Cube Server を使用すれば、管理者はリレーショナル・データベースに格納されているデータのマルチディメンション表現を作成することができます。 キューブ とは、オンライン分析プロセス (OLAP) でしばしば使われるデータ・モデルで、マルチ・ディメンションにわたって分析されるビジネス・データを表現します。 ディメンションとは、ビジネス分析で使用される概念上の軸です。たとえば、小売り企業の業績を時間、商品、および店舗の観点から分析することができます。 この企業の場合、*時間*、*商品*、*店舗* の 3 つがディメンションになります。 各ディメンションには、そのディメンションの全体的な階層を定義する 1 つまたは複数の*レベル* があります。 たとえば、*時間* ディメンションには、*年*、*四半期* (quarter)、および*月*などのレベルが含まれるでしょう。

キューブはビジネスのモデル化に使用されます。 3 ディメンション (3 次元) のキューブは幾何学的に正方体として描画できるので視覚化しやすいですが、キューブのディメンションは 1 個から *n* 個まで多数存在する場合があります。



キューブの複数のディメンションが交差する点において、分析者はメジャーを表示することができます。 メジャー (measure、測定値) とは特定の複数のディメンションの交点における数値であり、通常、計測されたビジネス指標を表します (売上高、商品の収益と費用など)。 たとえば、特定の時間における特定の商品の売上高を表示するには、キューブ内のこれら 3 つのディメンションが交差する点を調べて、メジャーを検出します。

リレーショナル・データのキューブ

照会可能な形でリレーショナル・データを保管するために、多くの組織がデータマートおよびデータウェアハウスに投資してきました。 このようなデータは通常、データが得られた元のトランザクション・システムから、照会のパフォーマンスを最適化する別のリレーショナル・データベースに移動されて、加工および変換されます。

こうして変換されたデータベースはしばしばデータウェアハウスまたはデータマートと呼ばれ、1 つまたは複数の件名に関する履歴情報を格納します。 普及しているデータマートおよびデータウェアハウスには、IBM®DB2 Universal Database™、Oracle、Microsoft® SQL Server、Sybase があります。 システムがどう呼ばれても、また、どんなリレーショナル・データベース管理システム (RDBMS) に保管されるとしても、こうしたデータベースの主な目的は、ユーザーが履歴情報を照会できるようにすることです。 このようなデータウェアハウスおよびデータマート・データベースの典型的なスキーマ設計について、詳しくは 9 ページの『第 2 章 ディメンション・スキーマの設計』を参照してください。

ディメンション・モデルを使用すれば、エンド・ユーザーはリレーショナル・データベースを簡単に照会できるようになります。 ディメンション・モデルによって、特定のビジネス手順またはビジネス領域に関連したビジネス上の調査が簡単になるためです。 ディメンション・モデルのサイズや複雑さ、およびビジネス上の要件によっては、専用の強力な OLAP サーバー (たとえば IBM DB2 OLAP Server™) が必要かもしれません。この場合、データはリレーショナル・データベースから抽出され、拡張された分析機能を提供する専用のハイスピードなキューブとして構築することができます。一方、専用の OLAP サーバーが必要とされないケースでも、DB2 Alphablox のキューブ機能を利用すれば、OLAP の強力な機能をユーザーに提供することができます。

DB2 Alphablox を使用すれば、管理者はリレーショナル・データの上に DB2 Alphablox キューブを構築できます。 つまり、基礎となる RDBMS に対する照会を使って DB2 Alphablox キューブのデータが取り込まれます。

DB2 Alphablox Cube Server の適用分野

DB2 Alphablox Cube Server を使用すれば、リレーショナル・データを OLAP キューブの形ですばやく提示できます。これにより、OLAP サーバー (たとえば IBM DB2 OLAP Server、Hyperion Essbase、または Microsoft Analysis Services) が提供するようなインテリジェント機能を部分的に提供できます。 DB2 Alphablox キューブはデータウェアハウスやデータマートに格納されたクリーン・データをそのまま利用しますが、高機能の OLAP サーバーに取って代わるものではありません。高機能の OLAP データベースを開発する時間やリソースがない場合、マルチディメンショ

ン・データ・ソースを作成するうえで Cube Server は役立ちます。 さらに、非常に大きなデータベースから比較的小さなキューブを構築して提示するうえでも、Cube Server は非常に優れています。

大規模リレーショナル・データベースからの比較的小さなキューブ・データ・セット

DB2 Alphablox Cube Server は、データ取得元のデータベースと比較して小さなデータ・セットを戻すようなキューブを構築するのに非常に適しています。基礎となるデータベースは非常に大きくなる可能性があります。 ファクト表の行数は何十億にもなるかもしれません (ファクト表の定義については、10 ページの『ファクト表』を参照してください。) DB2 Alphablox キューブはあらかじめ計算された結果を、ディスクではなく、メモリー内に保管します。 メモリーに保管されない結果はすべて、元のデータベースに残されます。 キューブはデータベースに対して SQL 照会を送ることにより、必要に応じて結果を取り出します。 その後、照会の結果はメモリーに保管され、DB2 Alphablox アプリケーションからただちにアクセスできるようになります。

プロトタイピング

DB2 Alphablox キューブはすばやく作成できるため、アプリケーションは実際のデータの値をただちに利用できます。 DB2 Alphablox キューブにアクセスする DB2 Alphablox アプリケーションを簡単に変更して、DB2® Cube Views、DB2 OLAP Server キューブ、Hyperion Essbase キューブ、または Microsoft Analysis Services キューブ内のデータにアクセスすることができます。 したがって、DB2 Alphablox キューブは開発サイクルにおいて大規模アプリケーションをプロトタイピングするための優れたプラットフォームとなります。 アプリケーションの要件によっては、DB2 Alphablox キューブの中にデータを残すだけでよい場合もあるでしょう。 あるいは、高機能の製品をスケーラブルに利用するのが適切な場合もあるでしょう。

単純なディメンションおよびメジャーを持つキューブ

DB2 Alphablox キューブの各ディメンションは、1 つの階層を持つことができます。 複数の階層を持つ複雑なディメンションを表現したい場合には、高機能の OLAP サーバー (たとえば DB2 OLAP Server、Hyperion Essbase、または Microsoft Analysis Services) を使用してください。 ただし、複雑なビジネス・シナリオのほとんどは、ディメンションごとに複数の階層を必要としません。

注: アプリケーションにおいて 1 つのディメンションに複数の階層が必要な場合、ルート・レベルが同じで階層が異なる複数のディメンションを作成することができます。

DB2 Alphablox キューブ内のメジャーは、基礎となるデータベースに対する有効な SQL 式を使って定義されます。 同じ列名を持つ複数の表が存在するというあいまいさの問題を防ぐために、SQL 式の指定にはいくつかの制約事項があります。 詳しくは、13 ページの『メジャー』を参照してください。

さまざまな RDBMS ベンダーがサポートする計算のレベルは異なりますが、ほとんどの主要な RDBMS ベンダーは非常に豊富な計算をサポートします。 SQL 式では表現できない計算をアプリケーションで使用する必要があれば、高機能の OLAP サーバーを使用することを考慮してください。

DB2 Alphablox Cube Server の利点

DB2 Alphablox Cube Server は DB2 Alphablox に含まれ、物理ディスク・ストレージをいっさい管理しないため、高機能の OLAP サーバーにつきものの多数の管理用タスクが単純化されるか、まったく必要ありません。 以下のような利点があります。

- DB2 Alphablox Cube Server が管理するディスク・スペースはありません。
- DB2 Alphablox Cube Server は DB2 Alphablox セキュリティー・モデルを使用するため、ユーザーを管理するための追加作業は必要ありません。
- DB2 Alphablox Cube Server は DB2 Alphablox に含まれるため、追加のソフトウェアをインストールする必要はありません。

DB2 Alphablox アプリケーション環境での DB2 Alphablox Cube Server

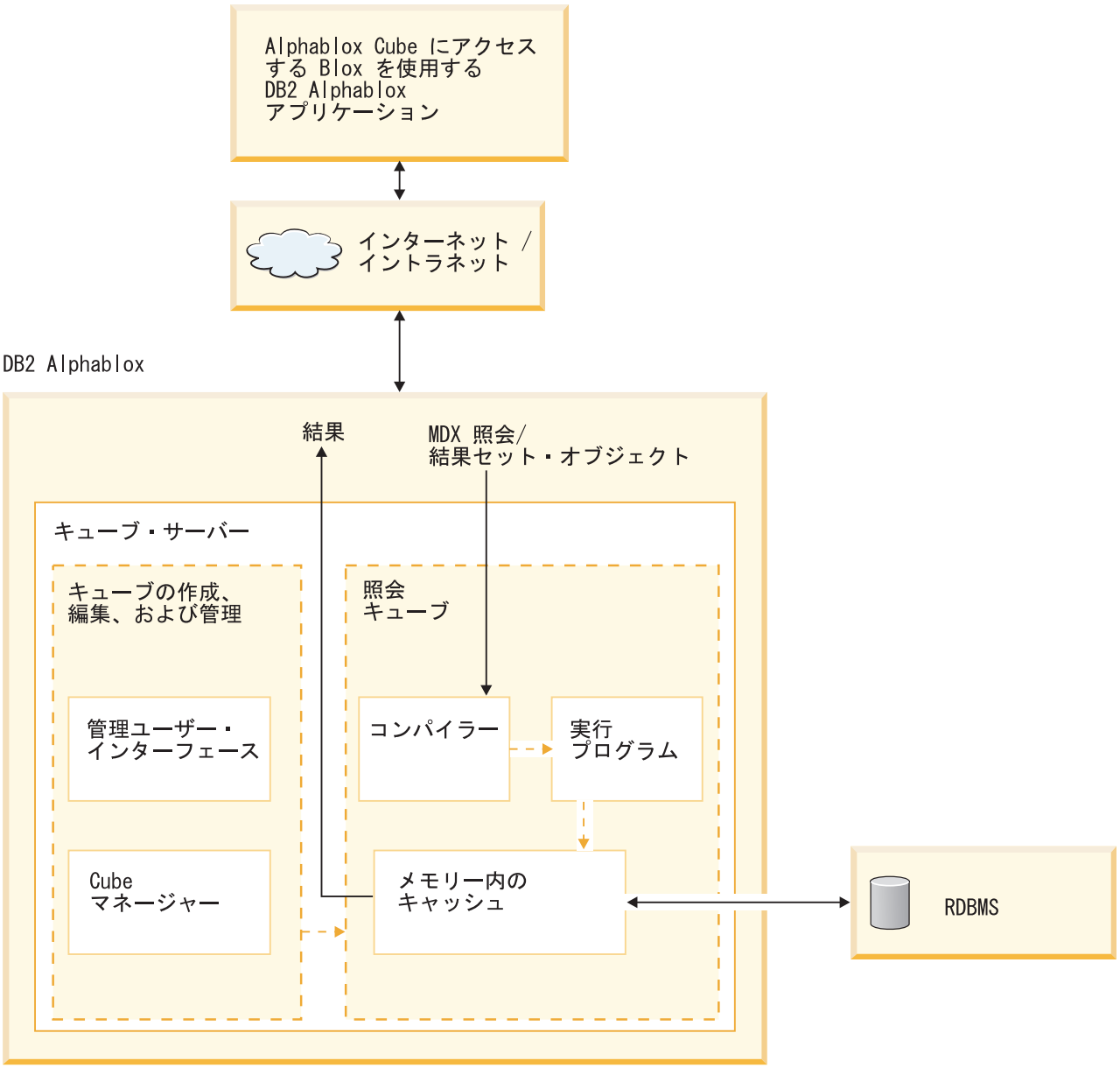
DB2 Alphablox アプリケーションにとって DB2 Alphablox キューブはデータ・ソースの 1 つにすぎません。 つまり、他のすべてのデータ・ソースに対するのと同様に、DB2 Alphablox キューブに対しても Blox が機能します。 DB2 Alphablox キューブは他のデータ・ソースと同じ Blox を使用します。 たとえば、照会およびデータ・ソース DataBlox パラメーターの値を変更するだけで、DB2 Alphablox キューブにアクセスするアプリケーションから DB2 OLAP Server キューブにアクセスするアプリケーションに変更できます。 アプリケーションのアクセス先のデータが変わるだけで、アプリケーションが機能する方法はそれまでと同じです。 他のマルチディメンションのリレーショナル・データ・ソースを操作する Blox の豊富な機能は、DB2 Alphablox キューブでも利用できます。

DB2 Alphablox Cube Server のアーキテクチャー

DB2 Alphablox Cube Server はハイパフォーマンスで拡張が容易なキューブ・エンジンであり、さまざまなキューブを照会するさまざまなユーザーをサポートします。 データウェアハウスまたはデータマート・データベースに格納されたリレーショナル・データに対する、高速かつ多元的なアクセスを可能にします。

DB2 Alphablox Cube Server のコンポーネント

DB2 Alphablox Cube Server はいくつかのコンポーネントによって構成されています。 これらの相互補完的なコンポーネントは、DB2 Alphablox キューブに対する照会を定義、管理、および実行するためのインフラストラクチャーを提供します。 以下の図に示されるように、DB2 Alphablox Cube Server のコンポーネントは DB2 Alphablox のフレームワーク内で機能します。



管理ユーザー・インターフェース

キューブ管理者は DB2 Alphablox の管理インターフェースを介して、DB2 Alphablox キューブのセットアップおよび管理タスクを実行します。 DB2 Alphablox ホーム・ページの「**管理 (Administration)**」タブの「**キューブ (Cubes)**」リンクは、キューブのセットアップと管理を行う専用ページを開きます。 ここでは、キューブのディメンション、レベル、およびメジャーを定義します。 DB2 Alphablox ユーザーが DB2 Alphablox キューブを作成、表示、または変更するには、そのユーザーは管理者グループのメンバーでなければなりません。 DB2 Alphablox キューブの管理ユーザー・インターフェースの使用方法について、詳しくは 15 ページの『第 3 章 キューブの作成と修正』および 29 ページの『第 4 章 キューブの保守』を参照してください。

Cube マネージャー

Cube マネージャーはコンポーネントの 1 つで、オブジェクトの作成、検査の実行、開始と終了など、DB2 Alphablox キューブに対する操作を実行します。 Cube マネージャーが実行するコマンドは、DB2 Alphablox コンソールでも実行できます。Cube マネージャー・コンソール・コマンドについての説明は、34 ページの『コンソール・コマンド』を参照してください。

メモリー内のキャッシュ

Cube Server は、算出された結果をメモリー内のキャッシュに保管します。こうして保管された結果は、DB2 Alphablox キューブにアクセスするすべてのユーザーによって共有されます。内部的に、それぞれのキューブはより小さな結果のセクションに分割されます。 これらの各セクションは基本的に、キューブのメモリー内キャッシュに保管されます。キューブ結果が必要とするメモリーの量、およびキューブが使用できるメモリーの量に応じて、キャッシュからいくつかの項目を削除しなければならない場合があります。メモリーを解放する必要がある場合、キャッシュ項目がキャッシュから消去されます。代わりに、基礎となるリレーショナル・データベースへの照会がキャッシュに入れられます。キャッシュにもはや保管されないデータが DB2 Alphablox キューブへの照会によって要求された場合、基礎となるデータベースからデータが検索されて、必要に応じてキャッシュから古いデータがエージングされます。 システムは、これらのキャッシュ機能をすべて自動的に実行します。

コンパイラー

DB2 Alphablox キューブに対する照会要求は、MDX 照会言語を使用します。コンパイラーは MDX 照会を解析し、要求を検証して、クライアント・アプリケーションに結果を戻すプラン (計画) を生成します。コンパイラーは各キューブに保管されたメタデータを利用して、それぞれの要求に関する最適なプランを生成します。

実行プログラム

実行プログラムは、コンパイラーが生成したプランを実行して、結果セットをキャッシュから取り出します。 生成された結果は、要求元の DataBlox、GridBlox、または PresentBlox に戻されます。

MDX から SQL 照会への変換

DB2 Alphablox アプリケーションは、MDX 照会を使ってキューブからの結果を要求します。 DB2 Alphablox Cube Server は MDX 照会を処理し、その結果として、DB2 Alphablox キューブから結果を検索するプランを生成します。次に、基礎となるリレーショナル・データベースに対する SQL 照会を実行することによって、DB2 Alphablox キューブがこれらの結果を計算します。 このような SQL 照会は、MDX 照会の発行前に実行されてキャッシュにあらかじめ保管されるか、または MDX 照会時に実行されます。キューブのメモリー内キャッシュに結果がすでに保管されている場合、その結果セットに対して SQL 照会を再び実行する必要はありません。DB2 Alphablox アプリケーションが MDX 照会を発行すると、DB2 Alphablox Cube Server はすべての必要な SQL 照会を自動的に発行します。通常、1 つの MDX 要求を満たすには多数の SQL 照会が必要です。

スキーマの要件

このセクションでは、DB2 Alphablox キューブによって参照される基礎となるデータベースの要件を説明します。 DB2 Alphablox キューブは、サポートされるいずれかれのリレーショナル・データベースを参照する必要があります。 DB2 Alphablox によってサポートされるデータベースについては、「*インストール・ガイド*」を参照してください。データベースには、*クリーン・データ* がディメンション・スキーマ の中に保管されなければなりません。

クリーン・データ

クリーン・データ とは、参照保全の規則に従うデータのことです (RDBMS によって参照保全が施行されるかどうかにかかわらず)。 加えて、クリーン・データとは、データ内の複数のフィールドの値が異なっても意味が同じある場合、それらが同じ値に変換済みであることを意味します。 たとえば、トランザクション・レベル・データ内のいくつかのレコードで第 2 四半期が *Q2* と表現され、他のいくつかのレコードでは *Quarter_2* と表現されている場合、第 2 四半期を指す固有値が 1 つだけ存在するように、これらのレコードは変換されなければなりません。

ディメンション・スキーマ

リレーショナル・データベース内のディメンション・スキーマとは、履歴照会を簡略化するクリーン・データを格納するための構造です。 通常、ディメンション・スキーマは以下のいずれかの形式です。

- 単一表
- スター・スキーマ
- スノーフレーク・スキーマ
- スター・スキーマとスノーフレーク・スキーマの組み合わせ

DB2 Alphablox キューブの基礎となるデータベースは、ファクト表をただ 1 つだけ含む必要があります。 複数のファクト表を含むスキーマはサポートされません。 DB2 Alphablox キューブ内の各ディメンションは、単一の階層を持つ必要があります。 スキーマについての詳細は、9 ページの『ディメンション・スキーマ』を参照してください。

注: データベースに複数のファクト表が含まれる場合、またはデータベースがディメンション・スキーマに従わない場合には、DB2 Alphablox キューブ用の単一ファクト表からなる “仮想” ディメンション・スキーマを作成するために、データベース内にビューを作成することができます。

第 2 章 ディメンション・スキーマの設計

DB2 Alphablox Cube Server の基礎となるデータベースには、ディメンション・スキーマが含まれなければなりません。 管理者が DB2 Alphablox キューブを正しくセットアップするためには、基礎となる RDBMS 内のデータを理解する必要があります。 この章では、ディメンション・スキーマ設計の概念を説明し、(スター・スキーマ、スノーフレーク・スキーマなどの) 用語を定義して、データベース構造とキューブ階層の関係について説明します。

ディメンション・スキーマ

データベースは 1 つまたは複数の表で構成されます。 データベース内のすべての表の間の関係を、まとめてデータベース・スキーマ といいます。 さまざまなスキーマ設計が存在しますが、履歴データを照会するために使われるデータベースは、通常、ディメンション・スキーマ設計 (中でも主にスター・スキーマまたはスノーフレーク・スキーマ) でセットアップされます。ディメンション・スキーマは多くの経緯や実用的理由から普及していますが、とくに意思決定支援リレーショナル・データベースの分野でこれほど普及しているのは、以下のような 2 つの主な利点があるためです。

- ビジネス上の質問に答える照会を作成できます。 このような照会は、通常、複数のビジネス・ディメンションを対象とする業績の測定値 (メジャー) を何らかの形で計算します。
- ほとんどの RDBMS ベンダーが採用している SQL 言語でこうした照会を作成する必要があります。

ディメンション・スキーマは、ビジネスの業績を定量化するメジャー (ファクト とも呼ばれる) と、ビジネスを記述および分類する記述的要素 (ディメンション とも呼ばれる) とを物理的に分離します。 DB2 Alphablox キューブを使用するには、基礎となるデータベースがディメンション・スキーマを使用する必要があります。 つまり、ファクト・データとディメンション・データが物理的に分離していなければなりません (少なくとも別々の列に分かれている必要があります)。 そのために、通常、スター・スキーマかスノーフレーク・スキーマ (または両者の混成) という形式が使用されます。 また、それほど数多いケースではありませんが、ディメンション・スキーマは単一表の形を取ることもあります。 この場合、ファクトとディメンションが単に表内の別個の列に格納されます。

注: データベースがディメンション・スキーマに従わない場合には、DB2 Alphablox キューブの “仮想” ディメンション・スキーマを作成するために、データベース内にビューを作成することができます。

ここでは、スター・スキーマとスノーフレーク・スキーマについて、およびこれらのスキーマがビジネス階層をどのように表現するかについて説明します。 以下のセクションがあります。

- 『スター・スキーマとスノーフレーク・スキーマ』
- 12 ページの『階層』

ディメンション・スキーマ設計の背景や詳細情報については、「*The Data Warehouse Toolkit*」(Ralph Kimball 著、John Wiley and Sons, Inc. 出版) をご覧ください。

スター・スキーマとスノーフレーク・スキーマ

スター・スキーマおよびスノーフレーク・スキーマ設計は、ファクトとディメンションを別個の表に分離するためのメカニズムです。 スノーフレーク・スキーマは、階層のさまざまなレベルをさらに別々の表に分離します。 どちらのスキーマ設計の場合も、それぞれの表が*主キー/外部キー関係* で別の表と関連しています。 主キー/外部キー関係は、リレーショナル・データベースにおいて多数の表の間の多対 1 関係を定義します。

主キー

主キー とは、表内の行を一意的に識別する値を持つ、表内の列または列のセットです。 リレーショナル・データベース設計では、主キーの一意性を実現するために、特定の主キー値を持つ行が表内で 1 つしか存在しません。

外部キー

外部キー とは、別の表の主キーと一致する値を持つ、表内の列または列のセットです。 特定の外部キー値を持つ行を追加するには、関連する表の中に、同じ主キー値を持つ行がすでに存在しなければなりません。

スター・スキーマまたはスノーフレーク・スキーマにおける表間の主キー/外部キー関係 (多対 1 関係と呼ばれることもある) は、RDBMS のさまざまな表を互いに関連させる結合 (接合) 経路を表します。 このような結合経路は、履歴データに対する照会を構築するうえで基礎となります。 多対 1 関係について、詳しくは 12 ページの『多対 1 関係』を参照してください。

ファクト表

ファクト表 とはスター・スキーマまたはスノーフレーク・スキーマにおける表で、売上高、商品コスト、収益などのビジネス業績を計測した結果であるファクト (事実) を保管します。 また、ファクト表にはディメンション表への外部キーもいくつか含まれます。 これらの外部キーは、ファクト表の各行のデータを、対応するディメンションおよびレベルに関連付けます。

ディメンション表

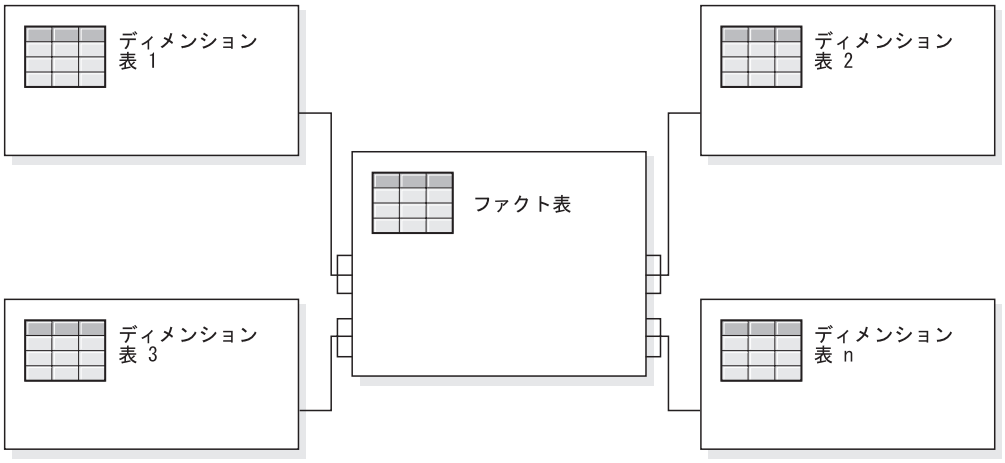
ディメンション表 とはスター・スキーマまたはスノーフレーク・スキーマにおける表で、ディメンションの特徴を記述する属性を保管します。 たとえば、時間表 (タイム・テーブル) は、年、四半期、月、日など、時間の特徴を保管します。 ファクト表の外部キーは、ディメンション表の主キーを多対 1 関係で参照します。

スター・スキーマ

以下の図は、1 つのファクト表および 4 つのディメンション表からなるスター・スキーマを示しています。 1 つのスター・スキーマには任意の数のディメンション表

を含めることができます。 表と表を連結する線の先端にある鳥の足型は、ファクト表と各ディメンション表との多対 1 関係を示しています。

スター・スキーマ



スノーフレーク・スキーマ

以下の図は 2 つのディメンションを含むスノーフレーク・スキーマを示しています。 各ディメンションには 3 つのレベルがあります。 1 つのスノーフレーク・スキーマには任意の数のディメンションを含めることができ、各ディメンションには任意の数のレベルを含めることができます。

スノーフレーク・スキーマ

ディメンションのさまざまなレベルがどのように階層を形成するかについて、詳しくは 12 ページの『階層』を参照してください。

階層

階層とは、互いに多対 1 関係を持つ複数のレベルからなるセットです。 これらのレベルのセットは集合的に 1 つのディメンションを形成します。 リレーショナル・データベースでは、1 つの階層内のさまざまなレベルを (スター・スキーマのように) 単一の表に保管したり、または (スノーフレーク・スキーマのように) 別々の表に保管することができます。

多対 1 関係

多対 1 関係とは、1 つのエンティティー (通常は列、または列セット) の中に、固有値を持つ別のエンティティー (列または列セット) を参照する値が含まれることです。 リレーショナル・データベースでは、このような多対 1 関係がしばしば外部キー/主キー関係によって実現され、通常はファクト表とディメンション表との関係、およびレベルと階層との関係を指します。 この関係は、分類やグループ分けを表すためによく使われます。 たとえば、*地区*、 *都道府県*、および*市町村* という 3 つの表からなる地理スキーマの場合、特定の地区には複数の都道府県が含まれますが、特定の都道府県が 2 つの地方に含まれることはありません。 同様に、特定の市町村はただ 1 つの都道府県に属します (ただし、同じ名前の市町村が複数の都道府県に含まれる場合は、少し異なります)。 つまり、それぞれの市町村は厳密に 1 つの都道府県に含まれますが、1 つの都道府県には多数の市町村が含まれます。これが “多対 1” の関係です。

階層が物理的にスター・スキーマまたはスノーフレーク・スキーマのどちらで表現されるとしても、階層内のさまざまな要素つまりレベルの間では、子レベルと親レベルの間に多対 1 関係が存在しなければなりません。 つまり、データはこのような関係に従う必要があります。 ディメンション・スキーマの 1 つの重要な特徴として、多対 1 関係を施行するためにクリーン・データが必要です。 さらに、この関係は、リレーショナル・データからの DB2 Alphablox キューブの作成を可能にします。

DB2 Alphablox キューブを定義するとき、階層を定義する多対 1 関係はディメンション内のレベルになります。 この情報は、管理ユーザー・インターフェースを介して入力されます。 DB2 Alphablox キューブを定義するためのメタデータの設定について、詳しくは 15 ページの『第 3 章 キューブの作成と修正』を参照してください。

平衡型階層と不平衡型階層

平衡型階層では、階層内のブランチはすべて同一レベルから派生し、各メンバーの親は、そのメンバーのすぐ上のレベルに存在します。平衡型階層では、ブランチはそれぞれ異なるレベルから派生するため、不均一な構造になります。 DB2 Alphablox Cube Server は、平衡型階層と不平衡型階層の両方をサポートします。

不平衡型階層の場合、スキップされたレベルは DB2 Alphablox Cube Server では無視され、存在しないものとして扱われます。標準のデプロイメント階層では、階層のレベル定義の関係が使用されます。つまり、階層内の各レベルは、デプロイメントにおいて 1 つのアイテムとして使用されます。標準デプロイメント階層がサポートされるのは、不平衡型階層の場合です。階層の各レベルが使用されますが、ディメンション表内の少なくとも 1 つの列がどのレベルにも必要であり、欠落したレベ

ルには NULL 値が入れられます。不平衡型であって、しかも階層の各レベル間で本来の親子関係を使用する再帰的デプロイメント階層は、DB2 Alphablox キューブではサポートされません。

不規則の階層

不規則の階層では、1 つのディメンションの少なくとも 1 つのメンバーの親メンバーが、そのメンバーのすぐ上のレベルにいるということはありません。不平衡型階層と同様に、階層のブランチはそれぞれ別々のレベルから派生してもかまいません。

DB2 Alphablox Cube Server は、不規則の階層の使用をサポートします。不規則の階層中でスキップされたレベルは無視されて、存在しないものとして扱われます。不規則の階層では、標準デプロイメント階層だけがサポートされます。階層の各レベルが使用されますが、ディメンション表内の少なくとも 1 つの列がどのレベルにも必要であり、欠落したレベルには NULL 値が入れられます。

キューブに対するリレーショナル・スキーマのマッピング

DB2 Alphablox キューブを設計および構築する管理者にとって、リレーショナル・データベースと DB2 Alphablox キューブとのマッピングを (少なくとも概要だけでも) 理解することが重要です。 このマッピングを理解すれば、DB2 Alphablox キューブの設計および作成段階での誤りを避けることができます。 基礎となるリレーショナル・データベースへの照会によってキューブにデータが入るため、キューブに対する照会結果とリレーショナル・データベースの照会結果を比較することにより、キューブの品質検査を実行できます。

ディメンション、レベル、および属性

1 つの DB2 Alphablox キューブ内には任意の数のディメンションを定義することができ、各ディメンションごとに任意の数のレベルを定義できます。 典型的なスノーフレーク・スキーマでは、各レベルが別々の表に正規化され、最も詳細なレベルがファクト表の外部キーによって参照されます。 DB2 Alphablox Cube Server はこれらの表の間の関係を使用して、キューブ内のディメンションを作成します。 DB2 Alphablox キューブを定義するときには、DB2 Alphablox キューブ定義の一部として、スキーマに関する詳細を指定する必要があります。

注: データベース内のビューを使用することによって、すべての論理表を 1 つの物理表に格納することができます。

メジャー

DB2 Alphablox キューブのメジャーは、リレーショナル・データベース内のファクト表から計算されます。 照会がメジャーを要求するとき、Cube Server は、照会で指定された各メンバーの直接の兄弟の値を計算します。 たとえば、Cube Server は、ある年の売上高メジャーをその年の 12 か月の売上高メジャーの合計として計算します。

メジャーを定義する SQL 式では、同じ列名を持つ複数の表が存在するというあいまいさの問題を防ぐために、すべての列名は、それが属する表を使って修飾されます。 このため、メジャー用の SQL 式には以下のような制約事項があります。

1. 式の中の最初のトークンは、メジャー表のいずれかの列でなければなりません。以下の式は括弧で始まるため、無効です。

   ```
   (store_sales - unit_sales) / store_cost
   ```

2. 式の残りの部分のすべての列は、厳密に 1 つの表の中に存在しなければなりません。
3. 式の中の列として、メジャー表内の外部キー列を指定することはできません。

第 3 章 キューブの作成と修正

管理者は、「**管理 (Administration)**」タブの「**キューブ (Cubes)**」セクションを使って DB2 Alphablox キューブを定義します。この章では、DB2 Alphablox キューブを作成するのに必要な手順を説明します。

キューブを作成するタスクのチェックリスト

このセクションでは、DB2 Alphablox キューブを定義するために必要なタスクのチェックリストを示し、それぞれのタスクについて簡単に説明します。タスクに関する詳細説明は、この章の後半にあります。

タスク		説明
1	基礎となるデータベースのスキーマを理解する	DB2 Alphablox キューブを定義するには、キューブ構築の基礎となるリレーショナル・データベースのスキーマを理解する必要があります。データベース・ツールを使ってデータベースをブラウズし、データベース内の表、列、主キー、および外部キーのそれぞれの名前を表示できることを確認してください。スキーマについては、9 ページの『第 2 章 ディメンション・スキーマの設計』を参照してください。
2	キューブに必要なメジャー、ディメンション、およびレベルを決定する。	スキーマを理解する以外に、DB2 Alphablox キューブの基礎となるリレーショナル・データベースのデータについても理解する必要があります。キューブ内でどんなメジャーを定義するか、これらのメジャーはデータベース内のどこに保管されているか、また、各ディメンションの階層のさまざまなレベル間の関係を理解する必要があります。
3	17 ページの『リレーショナル・データ・ソースの作成』	DB2 Alphablox キューブ作成の基礎となるリレーショナル・データベース用の DB2 Alphablox データ・ソース定義を作成します。
4	18 ページの『キューブの定義』	管理ユーザー・インターフェース「**キューブ (Cubes)**」を使用して DB2 Alphablox キューブのプロパティーを定義します。
5	19 ページの『メジャーの定義』	どんなファクトを DB2 Alphablox キューブで測定するか、およびリレーショナル・ファクト表から DB2 Alphablox キューブへの各メジャー (測定値) のマッピングを指定します。
6	20 ページの『ディメンションの定義』	DB2 Alphablox キューブ内の各ディメンション、および各ディメンションのそれぞれのレベルを指定します。 リレーショナル表と DB2 Alphablox キューブのディメンションおよびレベルとのマッピングを定義します。
7	23 ページの『Alphablox Cube Server Adapter のデータ・ソース定義の作成』	DB2 Alphablox キューブを照会するために、マルチディメンション・ドライバー **Alphablox Cube Adapter** を使って作成されるデータ・ソースを定義します。
8	24 ページの『キューブ・リソースの指定と管理』	DB2 Alphablox キューブの接続制限、更新頻度、その他の管理パラメーターを入力します。
9	26 ページの『キューブの検証』	DB2 Alphablox キューブを定義するために入力した情報が間違っていないことを確認します。

リレーショナル・データ・ソースの作成

DB2 Alphablox キューブを使用するには、基礎となるリレーショナル・データ・ソースを DB2 Alphablox データ・ソースとして定義する必要があります。それぞれの DB2 Alphablox キューブはリレーショナル・データ・ソースを参照する必要があります。データ・ソースは、ディメンション・スキーマ設計に従うリレーショナル・データベースを参照する必要があります。DB2 Alphablox キューブのためのリレーショナル・スキーマの要件については、7 ページの『スキーマの要件』を参照してください。ディメンション・スキーマについては、9 ページの『第 2 章 ディメンション・スキーマの設計』を参照してください。

リレーショナル・データベース用のデータ・ソースをすでに定義済みの場合、次のトピック (18 ページの『キューブの定義』) に進んでください。 データ・ソースについて、詳しくは 「*管理者用ガイド*」を参照してください。

リレーショナル・データベースを DB2 Alphablox データ・ソースとして指定するには、以下のステップを実行します。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**データ・ソース (Data Sources)**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。
5. 「**アダプター**」ドロップダウン・リストから、いずれかのリレーショナル・ドライバー・オプションを選択します (たとえば `IBM DB2 JDBC タイプ 4 ドライバー`)。
6. 「**データ・ソース名 (Data Source Name)**」テキスト・ボックスに、新しいデータ・ソースの名前を入力します。
7. 「**クライアント・ホスト名 (Client Host Name)**」、 「**ポート番号 (Port Number)**」、「**SID**」、および「**データベース (Database)**」フィールドに適切な情報を入力します (どのドライバーを選択したかに応じて、使用可能なフィールドが異なります)。正しい接続情報がわからない場合は、接続先のリレーショナル・データベースのデータベース管理者に問い合わせてください。
8. 「**デフォルト・ユーザー名**」と「**デフォルト・パスワード**」を入力します。これらは、リレーショナル・データベースの有効なユーザー名とパスワードでなければなりません。デフォルトのユーザー名とパスワードは、DB2 Alphablox キューブがリレーショナル・データベースにアクセスするときに常に使用されます。ここで指定するデータベース・ユーザーには、データベースに対する読み取りアクセスが必要です。

   注: データ・ソースを使って DB2 Alphablox キューブにデータを入れる場合、「**DB2 Alphablox ユーザー名とパスワードを使用 (Use DB2 Alphablox Username and Password)**」リストの値は無視されます。 DB2 Alphablox キューブへのユーザー・アクセスを制御するには、アクセス制御リスト (ACL) を使用します。ACL については、「*管理者用ガイド*」を参照してください。
9. データ・ソースを使用して DB2 Alphabloxキューブにデータを入れる場合、「**最大行数**」および「**最大列数**」テキスト・ボックスは無視されます。これら

の値を入力すると、他のアプリケーションがデータ・ソースを使用するときにこれらの値が使用されますが、DB2 Alphablox キューブはこれらのテキスト・ボックスを無視します。

10. JDBC ロギング情報を DB2 Alphablox ログ・ファイルに書き込まない場合は、「**JDBC トレースを使用可能にする (JDBC Tracing Enabled)**」ドロップダウン・リストを 「**いいえ (No)**」に設定してください。JDBC トレースを使用可能にするのは、問題が発生して、その原因をデバッグする必要がある場合のみです。
11. 「**保管**」ボタンをクリックしてデータ・ソースを保管します。

キューブの定義

DB2 Alphablox キューブの一般的なプロパティーを定義するには、以下のようにします。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。 「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. 「**Alphablox キューブ名 (Alphablox Cube Name)**」テキスト・ボックスに、DB2 Alphablox キューブの固有名を入力します。 DB2 Alphablox キューブの名前として使用できる文字は、A から Z まで、a から z まで、0 から 9 まで、下線文字 (_)、およびスペースです。
6. サーバーの再始動時に常にキューブを自動的に開始したい場合には、「**DB2 Alphablox キューブ名 (DB2 Alphablox Cube Name)**」テキスト・ボックス右側の「**使用可能 (Enabled)**」チェック・ボックスにチェック・マークを付けてください。 キューブを定義している途中で、キューブが正しく実行されないと思われる場合、または他のユーザーにまだアクセスさせたくない場合には、このチェック・ボックスのチェック・マークを外して、あとでキューブを使用可能にすることができます。
7. 「**リレーショナル・データ・ソース (Relational Data Source)**」ドロップダウン・リストを使って、17 ページの『リレーショナル・データ・ソースの作成』で定義済みのリレーショナル・データ・ソースを選択します。 DB2 Alphablox リレーショナル・データ・ソースがまだ定義済みでない場合、リストはブランクになります。
8. 「セキュリティー役割 (Security Role)」ドロップダウン・リストを使用して、(アプリケーション・サーバーまたは DB2 Alphablox で事前定義された) 役割を 1 つ選ぶことができます。 その場合、このキューブのユーザーは選択された役割に限定されます。 選択したセキュリティー役割を使用可能にするには、「使用可能 (Enabled)」チェック・ボックスがチェックされていなければなりません。
9. オプションで、IBM DB2 UDB をデータ・ソースとして使用し、このデータ・ソースに対して DB2 Cube Views キューブを使用できる場合には、「**DB2 Cube Views の設定を使用可能にする (Enable DB2 Cube Views Settings)**」オ

プションが使用可能になります。このオプションを選択すると、DB2 Cube Views の使用可能なキューブ定義を使って DB2 Alphablox キューブを指定することができます。このオプションを使用するには、以下のサブステップを実行してください。

a. 「**キューブ・モデル (Cube Model)**」リストを使用して、キューブ・モデルを 1 つ選択します。

b. 「**キューブ (Cube)**」リストを使用して、キューブを 1 つ選択します。

c. 「**ビジネス名を使用 (Use Business Names)**」または「**オブジェクト名を使用 (Use Object Names)**」ラジオ・ボタンを使用して、DB2 Alphablox キューブ内のオブジェクトの定義に使用する名前を指定します。

d. キューブ定義をインポートしてメジャーとディメンションを DB2 Alphablox キューブに事前に取り込むには、「**キューブ定義をインポート (Import Cube Definition)**」ボタンをクリックします。インポートされるキューブ定義に応じて、DB2 Alphablox Cube Server は DB2 Cube Views キューブに最もマッチする DB2 Alphablox キューブを指定するよう試みます。 インポート操作に関連した情報およびデバッグ・メッセージを示すログを確認するには、「**インポート・ログを表示 (Show Import Log)**」ボタンをクリックします。

e. この時点で、インポートされたキューブのメジャーとディメンションを (下記のように) 編集してキューブをカスタマイズできます。 あるいは、オプションで「**キューブ定義をインポート (開始時、再構築時、および編集時) (Import cube definition (on start, rebuild, and edit))**」オプションをチェックすることもできます。このオプションを選択すると、DB2 Alphablox キューブが開始され、再構築され、または編集用に開かれるたびに、DB2 Alphablox キューブは最新の DB2 Cube Views キューブ定義をロードします。

10. 「**保管**」ボタンをクリックして DB2 Alphablox キューブを保管します。

メジャーの定義

すべての DB2 Alphablox キューブでは、1 つまたは複数のメジャーを定義する必要があります。 メジャーについては、13 ページの『メジャー』の説明を参照してください。 DB2 Alphablox キューブ内のメジャーを定義するには、以下のステップを実行します。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブを選択して、「**編集**」ボタンをクリックします。 選択したキューブに対する DB2 Alphablox「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. キューブ・ナビゲーション・ツリーで、「**メジャー)**」ノードをクリックします。 メジャー・パネルが表示されます。
6. 「**メジャー・ファクト表 (Measures Fact Table)**」テキスト・ボックスに、基礎となるリレーショナル・データベースで定義されたとおりのファクト表の完全

修飾名を入力する必要があります (たとえば CVSAMPLE.SALESFACT)。 あるいは、ファクト表名を自動的に挿入するために、正しいスキーマ、カタログ、および表の組み合わせをドロップダウン・リストから選択します。

7. ファクト表を指定した後、「**新規メジャーの作成 (Create New Measure)**」ボタンをクリックして新しいメジャーを作成できます。 新しいオプションのセットが表示されます。
8. 「**名前 (Name)**」テキスト・ボックスで、″New Measure″ というテキストを新しいメジャーの名前に置き換えます。 メジャー名として使用できる文字は、A から Z まで、a から z まで、0 から 9 まで、下線文字 (_)、およびスペースです。

   この名前は DB2 Alphablox アプリケーションに送られる結果セットに表示されるため、見やすく、内容がわかりやすい名前を入力してください。 たとえば、ある店舗での売上合計を計算するメジャーの場合、*Store Sales (店舗売上)* というメジャー名にすることができます。
9. 「**式 (Expression)**」テキスト・ボックスに、有効な式を入力します。列や関数の正しい構文の入力を支援する式ビルダー・ツールを使用できます。 頻繁に使用される関数 (AVG、COUNT、MAX、MIN、および SUM) 用にショートカット・ボタンがありますが、任意の有効な関数を必要に応じて手動で入力することができます。 これらの関数は、新規メジャーを計算するために基礎となるデータベースに送られる SQL の生成に使用されます。 以下の式の例は、COGS メジャーを定義します。

   ```
   SUM(@col(CVSAMPLE.SALESFACT.COGS))
   ```
10. 「**適用**」ボタンをクリックしてメジャーをリストに追加します。
11. 必要に応じて、他の任意のメジャーを定義するためにこれらのステップを繰り返します。 メジャーを削除するには、ナビゲーション・ツリー内のメジャー・ラベルをクリックして、ツリー下部の「**選択項目を削除 (Delete Selected)**」ボタンをクリックします。
12. DB2 Alphablox キューブ定義の変更が完了したら、「**OK**」ボタンをクリックします。 または、ディメンションとレベルの定義を続けます。

    注: すでに開始済みの DB2 Alphablox キューブの内容を変更した場合、キューブを再始動または再構築するまでは、変更が有効になりません。 キューブの再始動および再構築については、29 ページの『キューブの開始、停止、および再構築』を参照してください。

ディメンションの定義

ディメンション、レベル、結合、属性、その他の DB2 Alphablox キューブ情報を定義するには、情報を入力する必要があります。

ディメンションおよびレベルについては、13 ページの『ディメンション、レベル、および属性』を参照してください。

ディメンションの作成と編集

ディメンションを作成または編集するには、以下のステップを実行します。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブを選択して、「**編集**」ボタンをクリックします。 選択したキューブに対する DB2 Alphablox「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. 左側の DB2 Alphablox キューブ・ツリーで、「**ディメンション**」ラベルをクリックします。右側のパネルに「ディメンションの作成 (Create Dimension)」ボタンが表示されます。 既存のディメンションを編集するには、ディメンション名をクリックすると既存のディメンション定義が表示されます。
6. 「**新規ディメンションの作成 (Create New Dimension)**」ボタンをクリックして新しいディメンションを作成するか、「**ディメンション**」リストからディメンションを選択して既存のディメンションを編集します。
7. 「**名前**」テキスト・ボックスで、ディメンションの名前を入力します。ディメンション名として使用できる文字は、A から Z まで、a から z まで、0 から 9 まで、下線文字 (_)、およびスペースです。
8. オプションで、「**説明**」テキスト・ボックスにディメンションに関する説明を入力します。 この説明は単なるコメント欄であり、ディメンション定義には影響を与えません。
9. 新しいディメンションに名前を付けた後、必要に応じて任意のファクト表結合およびディメンション結合を定義することができます。
10. 「**階層タイプ**」選択リストで、表示する階層のタイプを選択します。オプションには、**平衡型**、**不平衡型**、および**不規則**があります。
11. 「**デフォルト・メンバー (Default Member)**」フィールドで、ディメンションのデフォルト・メンバーを指定します。
12. 「**OK**」ボタンをクリックしてディメンションを保管します。

ファクト表結合の作成と編集

作成する各ディメンションごとに、ファクト表結合を定義する必要があります。ディメンション内にファクト表結合を作成または編集するには、以下のステップを実行します。

1. 「キューブの管理 (Cube Administration)」ダイアログを使って新規ディメンションを作成した後、新しいディメンションの下の「**ファクト表結合 (Fact Table Join)**」ノードをクリックします。
2. ファクト表結合を作成する (ファクト表結合がまだ存在しない場合) には、表示される「**新規結合の作成 (Create New Join)**」ボタンをクリックします。結合指定パネルが表示されます。 ファクト表結合がすでに存在する場合、「ファクト表結合 (Fact Table Join)」フォルダーを展開して結合をクリックします。
3. 「**式 (Expression)**」テキスト・ボックスに、ファクト表結合を指定する式を入力します。 なお、結合を定義する式の入力を支援する式ビルダーを使用することもできます。 例:

```
@col(MDSAMPLE.MARKET.STATEID) = @col(MDSAMPLE.SALESFACT.STATEID)
```

4. 「**適用**」ボタンをクリックすると、ダイアログを閉じずにこれらの設定が適用され、保管されます。 「**OK**」ボタンをクリックすると、レベル定義が保管されます。

ディメンション結合の作成と編集

作成する各ディメンションごとに、関連する表の間のディメンション結合も作成できます。 選択したディメンション内にディメンション結合を作成または編集するには、以下のステップを実行します。

1. 「キューブの管理 (Cube Administration)」ダイアログを使って新規ディメンションを作成した後、新しいディメンションの「結合 (Joins)」フォルダー下の「**ディメンション結合**」ノードをクリックします。
2. 新しいディメンション結合を作成するには、表示された「**新規結合の作成 (Create New Join)**」ボタンをクリックします。ディメンション結合ダイアログが表示されます。 既存のディメンション結合を編集するには、「ディメンション結合」フォルダーを展開して、編集したい結合を選択します。
3. 「**式 (Expression)**」テキスト・ボックスに、ディメンション結合を指定する式を入力します。 なお、結合を定義する式の入力を支援する式ビルダーを使用することもできます。 例:

   ```
   @col(MDSAMPLE.PRODUCT.FAMILYID) = @col(MDSAMPLE.FAMILY.FAMILYID)
   ```

4. 「**適用**」ボタンをクリックすると、「キューブの管理 (Cube Administration)」ダイアログを閉じずにこれらの設定が適用され、保管されます。 「**OK**」ボタンをクリックすると、変更内容を保管して「キューブの管理 (Cube Administration)」ダイアログを閉じます。

レベルの作成と編集

各ディメンションごとに、ディメンション階層内のレベルを指定できます。レベルを作成または編集するには、以下のステップを実行します。

1. 新規レベルを作成するには、そのディメンションの下の「**レベル**」ノードをクリックして、「**新規レベルの作成 (Create New Level)**」ボタンをクリックします。既存のレベルを編集するには、「レベル」フォルダーを開いて編集したいレベルを選択します。
2. 新規レベルを作成するには、「**名前**」テキスト・ボックスに名前を指定します。DB2 Alphablox キューブの名前として使用できる文字は、A から Z まで、a から z まで、0 から 9 まで、下線文字 (_)、およびスペースです。
3. 「**式 (Expression)**」テキスト・ボックスに、レベルを指定する式を入力します。なお、レベルを定義する式の入力を支援する式ビルダーを使用することもできます。 例:

   ```
   @col(MDSAMPLE.TIME.QUARTER)
   ```

4. 「**適用**」ボタンをクリックすると、「キューブの管理 (Cube Administration)」ダイアログを閉じずにこれらの設定が適用され、保管されます。 「**OK**」ボタンをクリックすると、変更内容を保管して「キューブの管理 (Cube Administration)」ダイアログを閉じます。

レベルの順序の設定

各レベルを指定した後、必要があればレベルの順序を変更することができます。

レベルの順序を設定するには、次のようにします。

1. 修正するディメンションの下の「**レベル**」ノードをクリックします。「**レベル順序の設定 (Set Level Order)**」パネルを示した新規のウィンドウが表示されます。
2. レベル順序リスト内でレベルを上下に移動するには、リスト中のレベルを選択してから、「**上に移動**」または「**下に移動**」ボタンをクリックします。
3. レベルの順序を設定し終わったら、「保管」ボタンをクリックします。

属性の作成と編集

属性は、レベルに属する追加のデータベース表列を表します。 属性は SQL 式によって指定されます。 この SQL 式は、単一のテーブル列への単純なマッピング、または複数の列や属性を組み合わせて SQL 関数を使用する複雑な式にすることができます。 属性を作成または編集するには、以下のステップを実行します。

1. 新しい属性を作成するには、ディメンションの下に表示された「**属性**」ノードをクリックして、「**新規属性の作成 (Create New Attribute)**」ボタンをクリックします。属性定義ダイアログが表示されます。 既存の属性を編集するには、編集したい属性ノードをクリックします。
2. 「**式 (Expression)**」テキスト・ボックスに、属性を指定する式を入力します。なお、属性を定義する式の入力を支援する式ビルダーを使用することもできます。 例:

   ```
   @col(FAMILY.FAMILYID)
   ```

3. 「**適用**」ボタンをクリックすると、「キューブの管理 (Cube Administration)」ダイアログを閉じずにこれらの設定が適用され、保管されます。 「**OK**」ボタンをクリックすると、変更内容を保管して「キューブの管理 (Cube Administration)」ダイアログを閉じます。

Alphablox Cube Server Adapter のデータ・ソース定義の作成

DB2 Alphablox キューブを照会するには、**Alphablox Cube Server Adapter** を使用する DB2 Alphablox データ・ソースを事前に定義する必要があります。 1 つのデータ・ソースを使用して、複数のアプリケーションから複数の DB2 Alphablox キューブにアクセスすることができます。 アクセス対象のキューブは、Alphablox アプリケーションが使用する MDX 照会の FROM 文節で判別できます。 DB2 Alphablox Cube Server Adapter データ・ソースを作成するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**データ・ソース (Data Sources)**」リンクをクリックします。
4. 「**作成**」ボタンをクリックします。
5. 「**アダプター**」ドロップダウン・リストから、「**Alphablox Cube Server Adapter**」というアダプターを選択します。

6. 「**データ・ソース名 (Data Source Name)**」テキスト・ボックスに名前を入力します。
7. オプションで、「**説明**」テキスト・ボックスに説明を入力します。
8. 「**最大行**」および「**最大列数**」テキスト・ボックスに数値を指定します。 これらの値は、このデータ・ソースを通して入力された照会の結果として戻される行数または列数を制限します。デフォルト値は 1000 です。
9. 「**保管**」ボタンをクリックしてデータ・ソースを保管します。

キューブ・リソースの指定と管理

それぞれの DB2 Alphablox キューブごとに、基礎となるデータベースからのデータをリフレッシュするスケジュールを定義できます。 また、各キューブを調整 (チューニング) するパラメーターをいくつか設定することもできます。

リフレッシュ・スケジュールの定義

DB2 Alphablox キューブで使用されるリレーショナル・データベース内の基盤のデータが変更されると、DB2 Alphablox キューブでキャッシュに入れられたすべてのデータが不整合になることがあります。データが不整合になった場合、基礎となるデータベースと比較して、DB2 Alphablox キューブから正しい答えが得られるようにするために、キューブを再構築する必要があります。 キューブを手動で再構築するには、DB2 Alphablox キューブを停止して再始動するか、コンソール・コマンド `REBUILD CUBE <cube_name>` を使用します。こうすれば、ディメンションが再構築され、メモリー内のキャッシュが空になります。 他の方法として、ディメンションは変更されないものの、新しいデータや変更されたデータがデータベースに追加された場合には、コンソール・コマンド `EMPTYCACHE <cube_name>` を使用してメモリー内のキャッシュだけを空にすることができます。

基礎となるデータベースが予測可能な間隔で定期的に更新される場合には、このデータベースを参照する DB2 Alphablox キューブを定期的に更新するようスケジュールを作成するのが適切かもしれません。 たとえば、データベースが毎晩午後 9:00 に更新される場合、毎朝午前 3:00 に DB2 Alphablox キューブを再構築するとよいかもしれません。

定期的に DB2 Alphablox キューブを自動的に再構築するよう構成するには、以下のステップを実行します。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブを選択して、「**編集**」ボタンをクリックします。 選択したキューブに対する DB2 Alphablox「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. 左側のキューブ・ナビゲーション・ツリーで、「**スケジュール**」ラベルをクリックします。 スケジューリング・パネルが表示されます。

6. 「**リフレッシュする間隔 (Refresh every)**」ボックスをチェックすると、DB2 Alphablox キューブ再構築のスケジュールを使用可能にすることができます。
7. リフレッシュ間隔を設定するために、適切なボタンをクリックして、それぞれの時間を変更します。 たとえば、毎朝 3:00 に DB2 Alphablox キューブを再構築するよう設定するには、2 番目のボタンを選択して、時刻 3:00 AM を入力します。
8. 「**保管**」ボタンをクリックして DB2 Alphablox キューブ定義を更新します。

チューニング・パラメーターの設定

それぞれの DB2 Alphablox キューブごとに、リソース管理用のチューニング・パラメーターをいくつか設定できます。 これを行うには、以下のステップを実行します。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブを選択して、「**編集**」ボタンをクリックします。 選択したキューブに対する DB2 Alphablox「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. 左側のキューブ・ナビゲーション・ツリーで、「**チューニング**」ノードをクリックしてチューニング・パネルを開きます。
6. 使用可能にしたいそれぞれのパラメーターの左側のボックスをチェックして、制限を表す数値を指定します。 以下の表は、使用できるパラメーターとその説明です。 これらのパラメーター、および他のチューニング・パラメーターについての詳細は、35 ページの『キューブのチューニング』を参照してください。

チューニング・パラメーター	説明
最大接続数	この DB2 Alphablox キューブへの同時接続の最大数。これは、すべての接続が同時に照会を実行する場合の制限です。制限に達すると、新しい接続は他の接続が空くのを待つ必要があります。
データ・ソース最大接続数	基礎となるリレーショナル・データベースへの接続の最大数。 制限に達すると、新しい接続は他のデータベース接続が空くのを待つ必要があります。この制限を使用した場合、それぞれの接続がいったん開くと、他の SQL 照会でそれを使用できるよう、(指定された制限に達するまで) ずっと開いたままになります。 この制限を使用しない場合、それぞれの照会が個別の接続を使用した後、それを閉じます。
キャッシュされる最大行数	データベースから戻されて Cube Server のメモリー内キャッシュに保管される行の最大数。 この制限は各キューブごとに別個に制御されます。 この制限に達した場合、キャッシュ内にある最も使用頻度が低い照会結果がキャッシュからエージングされて、新しい行を格納できるようになります。

7. オプションで、「**プリロード・パフォーマンス・キャッシュへの MDX 照会 (MDX Query to preload performance cache)**」テキスト・ボックスに MDX 照会を入力して、「**使用可能 (Enabled)**」チェック・ボックスをチェックすることができます。

   このテキスト・ボックスに入力した照会は、キューブが開始または再構築されるときに実行されます。 この照会は DB2 Alphablox Cube Server のメモリー内キャッシュに初期の結果セットを格納します。 これらの結果は、基礎となるデータベースから検索されたデータをキャッシュにシード (種まき) します。 DB2 Alphablox Cube Server キャッシュにすでに存在するデータだけを必要とするこれ以降のすべての DB2 Alphablox キューブ照会に対しては、キャッシュから直接応答するため、基礎となるデータベースに対する追加の SQL 照会を実行する必要がなく、応答時間が改善されます。 MDX 照会の FROM 文節で参照されるキューブ名は、「**DB2 Alphablox キューブ名 (DB2 Alphablox Cube Name)**」テキスト・ボックスですでに定義済みの名前でなければなりません。
8. 「**保管**」ボタンをクリックして DB2 Alphablox キューブ定義を更新します。

キューブの検証

DB2 Alphablox キューブの作成後、少し時間を取って、メジャー、ディメンション、およびレベルが正しく定義されたことを確認するのはよいことです。 誤りが見つかった場合、簡単に訂正できます。 DB2 Alphablox キューブを検証するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブ を選択して、「**編集**」ボタンをクリックします。 「DB2 Alphablox キューブの編集 - 一般 (Edit DB2 Alphablox Cube General)」タブを含むページが表示されます。
5. 「**リレーショナル・データ・ソース (Relational Data Source)**」テキスト・ボックスに指定されたデータ・ソースが、適切なリレーショナル・データベースを参照していることを確認します。 このとき、「**データ・ソース (Data Sources)**」管理ページでデータ・ソースの設定を確認する必要があるかもしれません。
6. DB2 Alphablox キューブを開始する前に、「**Alphablox キューブ名 (Alphablox Cube Name)**」テキスト・ボックスの横で「**使用可能**」が選択されていることを確認します。使用可能になっていない場合、キューブを開始しようとするとエラー・メッセージが表示されます。
7. 以下のようにして、メジャーが正しく作成されたことを検証します。
   a. 「**メジャー**」ノードをクリックして、リレーショナル・スキーマ内の正しい表が「**メジャー・ファクト表 (Measures Fact Table)**」テキスト・ボックスに指定されていて、名前のスペルが正しく、そして完全修飾名であることを確認します。
   b. 定義済みの各メジャーごとに、適切な集約が「**式 (Expression)**」テキスト・ボックスに指定されていることを検査します。

8. 必要なすべてのディメンションが正しく定義され、名前が正しいことを確認します。 各ディメンションごとに、以下を検査します。
   a. 必要なすべてのファクト表結合とディメンション結合をすでに追加し、式がすべて正しいことを検証します。
   b. レベルが正しく指定され、順序が正しいことを確認します。 最初のレベルは最も要約されたレベルでなければならず、それ以降の各レベルは階層内の 1 つずつ下のレベルでなければなりません。 たとえば、*時間* ディメンションの階層が *年*、*月*、*日* である場合、*年* が最初のレベル、それに続いて *月*、*日* という順序でなければなりません。
   c. 定義済みの属性 (名前と式を含む) がすべて正しいことを検証します。
9. 「**スケジュール (Schedule)**」タブをクリックして、すべての設定値が適切であることを確認します。
10. 「**チューニング (Tuning)**」タブをクリックして、すべての設定値が適切であることを確認します。

こうして DB2 Alphablox キューブ を検証した後、キューブを開始することができます。 DB2 Alphablox キューブの開始について、詳しくは 29 ページの『キューブの開始、停止、および再構築』を参照してください。

第 4 章 キューブの保守

DB2 Alphablox Cube Server には、DB2 Alphablox キューブに対する管理用タスクを実行する機能が提供されています。これらのタスクは、DB2 Alphablox 管理ユーザー・インターフェースまたはコンソールを使って実行します。 この章では、これらのタスクについて説明します。

キューブの開始、停止、および再構築

DB2 Alphablox キューブに対して最も頻繁に実行する管理用タスクは、キューブの開始、停止、および再構築です。

DB2 Alphablox キューブの開始

DB2 Alphablox キューブを照会で使用できるようにするには、キューブを開始する必要があります。 キューブを開始するには、DB2 Alphablox ホーム・ページ、またはコンソール・ウィンドウのコマンド行を使用できます。 キューブを開始すると、Cube Server は基礎となるリレーショナル・データベースへの照会を実行します。これらの照会の結果を使って、キューブのメモリー内キャッシュにディメンション・メンバーがロードされます。 いくつかの結果をあらかじめ算出してキューブのキャッシュに格納するために、キャッシュ・シード MDX 照会を DB2 Alphablox キューブの定義で指定できます。 これを指定する場合、Cube Server は起動時に MDX 照会を DB2 Alphablox キューブに対して実行し、MDX 照会から戻されたメジャー値をキャッシュに入れます。

ホーム・ページからのキューブの開始

DB2 Alphablox キューブを DB2 Alphablox ホーム・ページから開始するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。 「**一般**」ページが表示されます。
3. 「**ランタイム管理 (Runtime Management)**」セクションで、「**キューブ (Cubes)**」リンクをクリックします。
4. 「**DB2 Alphablox キューブ (DB2 Alphablox Cubes)**」リストから、開始したい DB2 Alphablox キューブを選択します。
5. DB2 Alphablox キューブの現在の状況を確認するには、「**詳細**」ボタンをクリックします。

6. 「**開始 (Start)**」ボタンをクリックします。 DB2 Alphablox キューブの開始操作が完了すると、状況フィールドに 「**実行中**」と表示されます。

コンソール・ウィンドウからのキューブの開始

コンソール・ウィンドウから DB2 Alphablox キューブを開始するには、以下を実行してください。

1. DB2 Alphablox がまだ実行中でなければ、開始します。 DB2 Alphablox の開始について、詳しくは「*管理者用ガイド*」を参照してください。
2. 「コンソール」ウィンドウで、以下のコマンドを入力します。

   ```
   start cube cube_name
   ```

   ここで、*cube_name* は開始したい DB2 Alphablox キューブの名前です。 Web ブラウザーで DB2 Alphablox 管理ページを使用している場合、「管理」->「一般」->「コンソール・セッションの開始 (Start Console Session)」をクリックしてコンソール・ウィンドウを開くこともできます。

キューブが開始しない場合のトラブルシューティング

DB2 Alphablox キューブの開始で障害が発生した場合、エラー・メッセージが表示され、それを確認すれば問題の原因を判別することができます。 問題をトラブルシューティングするとき、以下のロギング・ツールを使用すればより詳細な情報を得ることができます。

- DB2 Alphablox ログ・ファイルを検査します。
- 「コンソール」ウィンドウで以下を入力することにより、メッセージ・レベルをデバッグに引き上げます。

  ```
  report debug
  ```

- DB2 Alphablox リレーショナル・データ・ソースで JDBC トレースを使用可能にします。

これらのロギング・オプションを使用可能にする方法について、詳しくは「*管理者用ガイド*」を参照してください。

以下の表は、開始操作が失敗する典型的なシナリオをいくつか示し、問題を解決するための提案をリストしています。 問題を判別した後、その問題を修正して DB2 Alphablox キューブを再び開始してみてください。

エラー	説明
“キューブが使用可能であることを確認してください。”	キューブが使用可能であることを確認します。 コマンド行で以下を入力することによって、キューブが使用可能かどうかを確認できます。 `show cube cube_name` DB2 Alphablox キューブを使用可能にするには、「**キューブ (Cubes)**」ユーザー・インターフェースの「**一般**」タブで、「**Alphablox キューブ名 (Alphablox Cube Name)**」テキスト・ボックスの横の「**使用可能**」を選択します。

エラー	説明
基礎となるデータベースへの接続エラー。	さまざまな問題が原因で接続エラーが発生する可能性があります。よくある原因として、以下を検査してください。 • リレーショナル・データ・ソースの接続情報が正しいことを確認します。 • リレーショナル・データ・ソースのユーザー名とパスワードが有効で、非ヌルであることを確認します。 • データベースへの接続が使用可能であることを確認します。
基礎となるデータベースからの構文エラー。	リレーショナル・データベースからの構文エラーは、通常、キューブ定義のエラーを示唆しています。 たとえば、列が見つからないという構文エラーの場合、ディメンションの定義を検査して、データベースの列名および表名どおりの名前が正確に指定されていることを確認してください。

DB2 Alphablox キューブの停止

DB2 Alphablox キューブを停止すると、キューブは照会に使用できなくなり、キューブのメモリー内キャッシュのすべての項目と、すべてのディメンション・メンバーがキューブの一括表示から削除されます。

ホーム・ページからのキューブの停止

DB2 Alphablox キューブを DB2 Alphablox ホーム・ページから停止するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。 「**一般**」ページが表示されます。
3. 「**ランタイム管理 (Runtime Management)**」セクションで、「**DB2 Alphablox キューブ (DB2 Alphablox Cubes)**」リンクをクリックします。
4. 「**DB2 Alphablox キューブ (DB2 Alphablox Cubes)**」リストで、停止したい Alphablox キューブを選択します。
5. Alphablox キューブの現在の状況を確認するには、「**詳細**」ボタンをクリックします。
6. 「**停止**」ボタンをクリックします。 Alphablox キューブのシャットダウン操作が完了すると、状況フィールドに「**停止**」と表示されます。

コンソール・ウィンドウからのキューブの停止

コンソール・ウィンドウから DB2 Alphablox キューブを停止するには、以下のコマンドを入力してください。

```
stop cube cube_name
```

ここで、*cube_name* は停止したい DB2 Alphablox キューブの名前です。 Web ブラウザーで DB2 Alphablox 管理ページを使用している場合、「管理」->「一般」->「コンソール・セッションの開始 (Start Console Session)」をクリックしてコンソール・ウィンドウを開くこともできます。

注: 実行中の照会がすべて完了するまでは、DB2 Alphablox キューブは停止しません。

DB2 Alphablox キューブの再作成

基礎となるデータベースのデータ (ディメンション・データを含む) が変更された場合、DB2 Alphablox キューブを再構築または再始動する必要があります。 また、キューブ定義を変更した場合、変更内容を照会で有効にするためには、キューブを再構築または再始動する必要があります (あるいは、リフレッシュ・インターバルが構成済みの場合には次のインターバルまで待つこともできます)。

再構築の操作中には、照会でそのキューブを使用できません。 新しい照会は再構築が完了するまで待機した後で実行されます。 実行中のすべての照会が完了するまで、再構築操作は開始されません。 再構築操作にかかる時間は、ディメンションのサイズに応じて、および基礎となるデータベースからデータをディメンションに入れる照会のパフォーマンスに応じて異なります。

DB2 Alphablox キューブを再構築するには、コンソール・ウィンドウから以下のコマンドを入力します。

```
rebuild cube cubeName
```

ここで、*cubeName* は、再作成する DB2 Alphablox キューブの名前です。 Web ブラウザーで DB2 Alphablox 管理ページを使用している場合、「**管理**」 > 「**一般**」 > 「**コンソール・セッションの開始 (Start Console Session)**」をクリックしてコンソール・ウィンドウを開くこともできます。

ディメンション・データが未変更でも、ファクト・データが変更された場合 (たとえば最近の四半期の売上数がデータベースに追加された場合) には、メモリー内キャッシュの内容だけを空にすることができます。 キャッシュのすべての項目を空にして、ディメンション・メンバーをそのまま残すためには、コンソール・ウィンドウで以下のコマンドを入力します。

```
emptycache cube cubeName
```

管理ストラテジーの決定

DB2 Alphablox キューブを定義して開始した後は、以下のいずれかの場合に限って保守タスクが必要です。

- 基礎となるデータベースのデータが変更された場合。
- キューブ定義が変更された場合。

DB2 Alphablox キューブはメモリーに常駐するため、ディスク・スペースを管理する必要はありません。 メモリー管理上の考慮事項もありますが、ほとんどの場合、日常の管理タスクとはなりません。 メモリーの問題については、38 ページの『DB2 Alphablox キューブのメモリーに関する考慮事項』を参照してください。

基礎となるリレーショナル・データベースが稼働する環境について理解する必要があります。 基礎となるデータベースをどのように管理するかは、DB2 Alphablox キューブに非常に大きな影響を与えます。

データベース環境の理解

DB2 Alphablox キューブの基礎となるデータベースのデータが変更されるたびに、キューブの一部のデータが古くなる可能性があります。 DB2 Alphablox キューブは、基礎となるデータベースへの照会からデータを取得します。 照会で DB2 Alphablox キューブのデータが要求された場合、DB2 Alphablox Cube Server は結果がメモリー内キャッシュに存在するかどうかを検査します。 結果が存在する場合、アプリケーションはそれをただちに使用でき、応答時間が非常に速くなります。 このような結果は基礎となるデータベースから得られたものですが、過去のある時点で取得されたにすぎません。 データが未変更の場合には、これで問題ありません。しかし、キャッシュ項目が保管された時点から照会が結果を要求した時点までの間に、基礎となるデータベースのデータが変更された場合には、結果が古くなります。

さらに、DB2 Alphablox キューブのメンバーがデータベースで挿入、更新、または削除された場合には、DB2 Alphablox キューブの結果がディメンションの実際の状態を反映しなくなります。 DB2 Alphablox キューブへの新しい照会の結果は、基礎となるデータベースからの結果と引き続き一致する可能性もありますが、一致しない可能性もあります。 それを決定する要因は、データベース内の厳密にどんな値が変更されたか、Alphablox キューブのメモリー内キャッシュにどんなデータが格納されているか、および照会によってどんなデータが要求されるかです。

基礎となるデータベースのデータが変更された場合、DB2 Alphablox キューブが引き続き有効かつ最新であることを確実に確認する方法はないため、最も安全な処置は、キューブを再構築することです。 したがって、基礎となるデータベースのデータがいつ変更されるかを知っておくことが重要です。

たとえば、データベースが決して変更されないことがわかっている場合、一度も DB2 Alphablox キューブを再構築する必要はありません。 データベースの中で、キューブの定義対象ではない部分にのみ新しいデータが追加される場合にも、再構築の必要はないでしょう。

データベースが毎晩更新され、すべての部分が変更される可能性がある場合には、毎晩、データベースの更新が完了した後に DB2 Alphablox キューブを再構築する必要があるでしょう。 データベースの稼働環境をよく知っておけば、どんなときに DB2 Alphablox キューブのデータが無効になるか、より正確に予想することができます。

定期更新のスケジューリング

多くの場合、データウェアハウスやデータマート・データベースは既知のスケジュールに基づいて更新されます。 そのスケジュールに基づいて、DB2 Alphablox キューブの定期更新をスケジュールすることができます。 `REBUILD CUBE` または `EMPTYCACHE CUBE` コマンドを使用すれば、更新を随時実行できます。 あるいは、各 DB2 Alphablox キューブの自動的な再構築スケジュールをセットアップすることもできます。 自動スケジュールのセットアップについての詳細は、24 ページの『リフレッシュ・スケジュールの定義』を参照してください。

DB2 Alphablox キューブの更新をスケジュールするうえで、ただ 1 つの最適な方法というものはありません。 リレーショナル・データベースでどんな操作が行われているかを知っておくことが非常に重要です。 同様に、ユーザー・コミュニティーの

利用パターンとニーズを理解しておくことも重要です。キューブのサイズや基礎となるデータベースのサイズに応じて、DB2 Alphablox キューブの再構築にある程度の時間がかかる可能性があります。 通常、システムを使用するユーザーがほとんど、またはまったくいない深夜に再構築するようスケジュールするのが最適です。また、とくに再構築操作に長い時間がかかる場合には、その間はキューブが使用不可になることをユーザーに確実に知らせてください。

コンソール・コマンド

DB2 Alphablox コンソール・ウィンドウから、ほとんどのキューブ管理タスクを実行することができます。コンソールにアクセスするには、 「**管理**」タブ、「**一般**」ページ、「**コンソール・セッションの開始 (Start Console Session)**」リンクを使用するか、DB2 Alphablox を開始したときに開く DB2 Alphablox コンソール・ウィンドウを使用します。 以下の表はキューブ・コマンドをリストし、それぞれの機能を説明しています。

コマンド構文	説明
delete cube *cubeName*	キューブおよび定義全体を削除します。
disable cube *cubeName*	キューブを使用不可状態に設定します。 使用不可になったキューブは使用可能にされるまで開始できないため、DB2 Alphablox が開始しても自動的に開始されません。キューブを使用不可にする前に、あらかじめ停止する必要があります。
emptycache cube *cubeName*	キューブのメモリー内のキャッシュからすべての項目を除去します。 キャッシュが空になると、DB2 Alphablox キューブに対する照会は基礎となるデータベースから結果を取得する必要があります。 基礎となるデータベースの内容が変更された場合、データベースに保管されたデータと DB2 Alphablox キューブから取得される結果とを同じにするために、このコマンドを使用してください。 なお、EMPTYCACHE コマンドによってキューブのディメンション一括表示は再構築されないことに注意してください。 ディメンション一括表示を再構築するには、REBUILD コマンドを使用するか、キューブを停止して開始してください。
enable cube *cubeName*	キューブを使用可能状態に設定します。 キューブを開始する前に、あらかじめ使用可能にする必要があります。 使用可能になったキューブは、DB2 Alphablox の開始時に自動的に開始します。
rebuild cube *cubeName*	まず、すべてのディメンションのメンバー名とすべてのメジャーをメモリー内キャッシュから除去します。次に、基礎となるデータベースを照会して、すべてのディメンションのメンバー名を再び格納します。 キューブ定義で初期 MDX キャッシュ・シード照会が指定されている場合、その照会が実行されてキャッシュにデータが入ります。

show cube *cubeName*	キューブの現在の状況を示します。 キューブの状況は、以下のいずれかです。 • 使用不可 • 停止済み • 開始中 • 実行中 定義済みのすべての DB2 Alphablox キューブの状況を表示するには、以下のコマンドを入力します。 `show cube`
start cube <*cube_name*>	キューブを開始して、照会で使用できるようにします。 キューブが開始すると、基礎となるデータベースを照会してディメンション・メンバーにデータを入れ、MDX キャッシュ・シード照会を実行します (キューブ定義で定義されている場合)。
stop cube <*cube_name*>	実行中のキューブを停止します。 停止したキューブは照会で使用できなくなり、ディメンション・メンバーとメジャーがメモリー内キャッシュから除去されます。

キューブの修正

DB2 Alphablox キューブ定義の任意の部分を、任意の時点で変更することができます。停止済みのキューブを変更した場合、変更内容はただちに適用されます。 実行中のキューブを変更した場合、変更内容はキューブ定義にただちに保管されますが、実行中のキューブが (コンソールまたはスケジュール済みリフレッシュによって) 再構築または再始動されるまで、変更内容は適用されません。

DB2 Alphablox キューブを変更するには、キューブ作成時と同じようにして「**キューブ (Cubes)**」管理ページを使用することができます。 キューブ定義の任意の部分を更新して保管することができます。 ユーザー・インターフェースの各部分で定義を入力する方法について、詳しくは 15 ページの『第 3 章 キューブの作成と修正』を参照してください。

キューブのチューニング

DB2 Alphablox キューブをチューニング (調整) および構成するための管理コントロールが多数あります。 DB2 Alphablox キューブはメモリー内で実行され、大量のメモリーを使用する可能性があるため、メモリーに関する考慮事項をいくつか理解しておく必要があります。

チューニング・コントロール

DB2 Alphablox キューブのリソースを制御するには、このセクションで説明されるコントロールを使用してください。

接続およびキャッシュ・サイズの制限

それぞれの定義済み DB2 Alphablox キューブの接続およびキャッシュ・サイズの制限を「**キューブ (Cubes)**」ページで指定できます。 「キューブの管理 (Cube

Administration)」ダイアログを開いて、キューブ・ナビゲーション・ツリーの 「**チューニング (Tuning)** 」というラベルをクリックしてください。

最大接続数: 同時に多数のユーザーが DB2 Alphablox キューブを照会した場合、ユーザーが少ない場合に比べて、DB2 Alphablox を実行するコンピューターのマシン・リソースが早く消費される可能性があります。 ただし、複数の照会が*厳密に同時に* 実行されない限り、リソースの競合は発生しないことに注意してください。 たとえ多数のユーザーが同時に接続しても、競合が頻繁に起きるとは限りません。 システムでこの点が実際に問題になるようであれば、各 DB2 Alphablox キューブごとに許容される接続数を制限することができます。

どれほどのリソースが使用されるかは、発行される照会のタイプに完全に依存します。 ほとんどの照会はマシン・リソースを少ししか使用しませんが、長時間実行される照会はかなりのリソースを消費するかもしれません。

データ・ソースへの最大接続数: 「**データ・ソース最大接続数 (Maximum Data Source Connections)**」オプションは、基礎となるデータベースへの接続数を制限します。このボックスをチェックした場合、データベースへの “接続プール” が使用可能になります。 このモードでは、照会がデータベースに送られるたびに、接続が開いて照会が発行されます。 照会が完了した後も、後続の照会のために接続は開いたままになります。 別の照会が送信された場合、開いているアイドル状態の接続があれば、それが使用されます。 すべての接続がビジー状態であれば、指定された制限に達するまで、新しい接続が開かれます。

データベースの “接続プール” が便利なのは、データベース接続の数を制限するからです。時間がある程度経過した後、指定した数のデータベース接続が開いているかもしれませんが、指定した数を超えることは決してありません。

このボックスをチェックしないと、DB2 Alphablox Cube Server によってデータベースに照会が送られるたびに新しい接続が開き、結果が戻されるたびに接続が閉じます。他の接続の状況にかかわりなく、新しい接続が開きます。接続は共有されず、アイドル状態のままにもなりません。

どんなに小さいデータベース接続にも、それなりのコストがあります。 ほとんどの場合、応答時間の違いは気になりませんが、応答時間がはっきり異なる場合もあるかもしれません。 また、基礎となるデータベースの側で受け入れる接続数を制限する場合もあり、データベース管理者はあまりに多数の接続を使用させたくないかもしれません。 接続を開いたままにしたくない場合には、「**データ・ソース最大接続数 (Maximum Data Source Connections)**」チェック・ボックスにチェックしないでください。

キャッシュされる最大行数: 「**キャッシュされる最大行数 (Maximum Rows Cached)**」は、データベースから戻される行のうち、キャッシュに格納できる行数を制限します。 このボックスをチェックした場合、制限に達すると、最も使用頻度が低い行が除去されて、新しい照会によって戻される行のためのスペースが確保されます。 このボックスをチェックしない場合、キャッシュのサイズには制限がなく、どこまでも大きくなる可能性があります (最大で、基礎となるデータベースのデータ量と同じになります)。 場合によっては、このボックスをチェックしないとシス

テムのメモリーが不足するかもしれません。 メモリーについての詳細は、38 ページの『DB2 Alphablox キューブのメモリーに関する考慮事項』を参照してください。

より多くのデータがキャッシュに格納されるほど、DB2 Alphablox キューブに対する照会で基礎となるデータベースから結果を検索する頻度が少なくなり、照会の応答時間が速くなります。 ただし、キャッシュが大きくなりすぎると、マシンのメモリーを使い尽くして、すべてのユーザーのパフォーマンスが遅くなる可能性があります。 ご使用のシステムにとって最適のサイズを判別するには、メモリー・リソース、ユーザー・ロード、照会のロードを考慮して実際に試す必要があります。 ユーザーおよび照会のロードに応じて、利点と欠点を比較考慮して最適なキャッシュ・サイズを決定してください。

それぞれの DB2 Alphablox キューブの接続数、データ・ソース接続数、最大キャッシュ・サイズの制限を指定するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。
3. 「**キューブ (Cubes)**」リンクをクリックします。
4. キューブのリストの中から DB2 Alphablox キューブを選択して、「**編集**」ボタンをクリックします。 選択したキューブに対する DB2 Alphablox「**キューブの管理 (Cube Administration)**」ダイアログが新しい Web ページ・ウィンドウとして表示されます。
5. 「**チューニング (Tuning)**」タブをクリックします。
6. 設定したい制限のボックスをチェックして、対応する数値を入力します。
7. 「**保管**」ボタンをクリックして、DB2 Alphablox キューブ定義の制限を保管します。

キューブの最大数

多数の DB2 Alphablox キューブを定義した場合、各キューブの開始に大量のメモリーおよびマシン・リソースが使用されるのであれば、システム全体のパフォーマンスが影響を受けます。 これを制御するために、DB2 Alphablox で実行可能な DB2 Alphablox キューブの数を制限することができます。 この制限は、同時に実行できる DB2 Alphablox キューブの数を制限するだけです。 定義可能な数を制限するものではありません。

同時に実行可能な DB2 Alphablox キューブ数の制限を設定するには、以下のステップを実行してください。

1. *admin* ユーザー、または管理者グループのメンバーであるユーザーとして、DB2 Alphablox ホーム・ページにログインします。
2. 「**管理 (Administration)**」タブをクリックします。 「**一般**」ページが表示されます。
3. 「**一般プロパティー**」セクションで、「**DB2 Alphablox Cube マネージャー**」リンクをクリックします。
4. 「**キューブの最大数 (Maximum Cubes)**」というボックスをチェックして、設定したい制限の数値を入力します。

5. 「**保管**」ボタンをクリックして変更内容を保管します。

行および列の最大数

DB2 Alphablox キューブ・データ・ソースの行と列の最大数を制限することにより、大量のデータを戻すような照会を発行するアプリケーションを制限することができます。 DB2 Alphablox キューブ・データ・ソースに関するこれらの制限を設定するには、「**データ・ソース (Data Sources)**」管理ページを使用します。 このデータ・ソースは、DB2 Alphablox キューブへの MDX 照会で使用されるデータ・ソースです。

DB2 Alphablox キューブのメモリーに関する考慮事項

DB2 Alphablox Cube Server は、DB2 Alphablox の稼働環境である Java™ プロセスの一部として実行されます。 このため、Cube Server がより多くのメモリーを使用すると、Java プロセスも多くのメモリーを使用します。 DB2 Alphablox Java プロセスのメモリー制限はインストール時に設定されます。 DB2 Alphablox キューブのメモリー使用量が多いために DB2 Alphablox のメモリーが不足する場合、以下のような処置を取ることができるでしょう。

- 各キューブのメモリー内キャッシュのサイズを制限します。 詳しくは、35 ページの『接続およびキャッシュ・サイズの制限』を参照してください。
- システム内の Alphablox キューブの数を制限します。 詳しくは、37 ページの『キューブの最大数』を参照してください。
- DB2 Alphablox が稼働している Java プロセスのメモリー・ヒープ最大サイズを変更します。 詳しくは、『メモリー・ヒープ最大サイズの変更』を参照してください (下記)。
- DB2 Alphablox が稼働しているコンピューターのメモリー容量を増やします。 詳しくは、38 ページの『システムへのメモリーの追加』を参照してください。

メモリー・ヒープ最大サイズの変更

Cube Server は Java プロセスの一部として実行されます。 DB2 Alphablox でメモリー不足エラーが発生する場合には、Java プロセスのメモリー・ヒープ最大サイズを増やす必要があるかもしれません。 メモリーの要件を満たす十分に大きな値にメモリー・ヒープ最大サイズを増やしてください。 ただし、プロセスのサイズが最大値に近づいたときにオペレーティング・システムがディスクに大量にスワップするのを防ぐために、値を大きくしすぎないでください。 さらに、予期しない理由でマシンのメモリーが使用される可能性も考慮に入れてください。 たとえば、マシンのメモリーが 1024 MB で、マシン上の他のリソースが約 300 MB のメモリーを使用する場合、メモリー・ヒープの最大サイズを 600 MB に設定することができます。

ご使用のシステムでどんな値が最大値として適しているか判別するには、何度か試してみる必要があるかもしれません。 DB2 Alphablox キューブの問題がとくに発生せず、パフォーマンスが良好で、メモリー不足エラーも発生しない場合、その制限はご使用の環境に適していると言えます。

システムへのメモリーの追加

メモリーの問題が存在する場合、DB2 Alphablox が稼働するシステムにメモリーを追加するという解決策に気付かないことがよくあります。 ご使用のコンピューターにどれほどのメモリーを搭載できるか、ハードウェア・ベンダーに確認してくださ

い。 システムのメモリー使用量が、搭載された物理メモリーの限界に近づいた場合、システムは新しいメモリー要求を満たすために、メモリーをディスクにスワップします。 このとき、メモリー管理がかなり非効率的になります。

メモリーをアップグレードすると、比較的低コストでサーバーの能力を増強できる場合が少なくありません。 また、こうすることでメモリー使用の問題が改善または解決される場合が少なくありません。 システムにメモリーを追加する余地があれば、そうするよう考慮してください。

第 5 章 MDX を使用した DB2 Alphablox キューブの照会

DB2 Alphablox アプリケーションは MDX (マルチディメンション式、Multidimensional Expressions) 言語を使用して DB2 Alphablox キューブを照会します。 MDX は、Microsoft によって作成され保守されている OLE DB for OLAP 仕様の照会言語コンポーネントです。 DB2 Alphablox キューブは、MDX の構文および関数をいくつかサポートします。 このセクションでは DB2 Alphablox キューブの照会用にサポートされる MDX 構文を説明し、照会の例を示します。

サポートされる MDX 構文

MDX は、Microsoft Analysis Services などのマルチディメンショナル・データベースで使用されるマルチディメンション照会言語です。 DB2 Alphablox Cube Server は、DB2 Alphablox キューブ用の照会言語としていくつかの MDX 構文を使用します。 DB2 Alphablox キューブにアクセスする DB2 Alphablox アプリケーションでは、MDX 照会は DataBlox 照会パラメーター (または関連するメソッド) の値として使用されます。

基本構文

DB2 Alphablox キューブに対する MDX 照会の基本的な構文は、以下のとおりです。

```
SELECT {axisSpecification} ON COLUMNS,
       {axisSpecification} ON ROWS
FROM cubeName
WHERE (slicerItems)
```

ここで、

axisSpecification
: 1 つまたは複数のタプル。 タプルをリストとして入力するか、または CrossJoin 関数によって “生成する” ことができます。

cubeName
: 定義済み Alphablox キューブの名前。

slicerItems
: 照会の結果セットのフィルタリングに使用されるタプル (通常、コンマで区切られた複数メンバーからなるリスト)。 複数のスライサー・メンバーを含める場合、それぞれ異なるディメンションに属する必要があります。 また、照会で指定されるいずれかの軸においてディメンションが参照されてはなりません。

使用上の注意

各ディメンションは、照会内の単一の軸にのみ出現可能です。 複数の軸にディメンションを配置するような照会は、エラーとともに失敗します。

照会ではゼロ個以上の軸を指定できますが、通常は 2 つの軸を指定します。 COLUMNS 軸を AXIS(0)、ROWS 軸を AXIS(1) として指定することもできます。

それ以降の軸は AXIS(*n*) となります (ここで、 *n* は連続する次の整数)。 ただし、照会のデータ (GridBlox、ChartBlox、または PresentBlox) を表示する DB2 Alphablox アプリケーションは、最大で 2 つの軸を指定する照会だけを受け入れることに注意してください。 XML データ・セットを提供する照会は、任意の数の軸を受け入れることができます。

DB2 Alphablox に対する MDX のキーワードは大/小文字を区別しませんが、MDX 照会内のメンバー名を大括弧 [ ] で囲んだ場合には大/小文字が区別されます。 メンバー名を大括弧 [ ] で囲まない場合、それらはサーバーに送られる前に大文字に変換されます。 データベース内でメンバー名がすべて大文字になっていない限り、構文で大括弧を使用してください。

メンバー・セットの指定

メンバー・セット は、同じディメンションに属する 1 つまたは複数のメンバーから成ります。 メンバー名を必ずしも大括弧 [ ] で囲む必要はありませんが、そうすることをお勧めします。 メンバー名にスペースが含まれる場合には、大括弧が必要です。 以下のような指定では、メンバー名の大/小文字が区別されるため、2 つは同等ではありません。

```
[Time].[Fiscal Year]
[Time].[fiscal year]
```

メンバー名の修飾

オブジェクト構文の場合と同様に、以下のようにディメンション名および階層の親の名前を使ってメンバー名を修飾することができます。

```
[Dimension].[Level].[Member]
```

また、以下のようにして、ディメンション名およびメンバーの 1 つまたは複数の祖先を使ってメンバー名を修飾することもできます。

```
[Dimension].[Member].[Member]
```

注: 必ずメンバー名が固有になるように修飾する必要があります。

中括弧

中括弧 { } はセットを表します。 MDX 照会では、軸に配置されるセットを中括弧で囲む必要があります。 たとえば、Golden Oats (ゴールデン・オート) および Sugar Grains (粒状の砂糖) 商品からなるセットを指定する構文は、以下のようになります。

```
{[Product].[Golden Oats], [Product].[Sugar Grains]}
```

FROM:TO 構文

2 つのメンバーをコロン (:) で区切ることによって、レベル上の1 点から別の点へ拡張するメンバー・セットを指定できます (それらの点を含む)。 たとえば、*Alphabet* というディメンションに A から Z のメンバーが含まれる場合、以下の構文は {D, E, F, G, H} というセットとして評価されます。

```
{[Alphabet].[D]:[Alphabet].[H]}
```

永続的算出メンバーの作成

DB2 Alphablox キューブで永続的算出メンバーを作成するには、算出メンバーの仕様を、非データベース (フラット・ファイル) DB2 Alphablox Repository のキューブ・プロパティー・ファイルに追加します。キューブ・プロパティー・ファイルは、以下の DB2 Alphablox のインストール先に置かれている *cubeName.properties* ファイル内にあります。

db2_alphablox/repository/cubes/*cubeName*/

ただし、*db2_alphablox* は、DB2 Alphablox インストールのルート・ディレクトリーであり、*cubeName* は、キューブ定義に指定したキューブの名前です。

定義する各算出メンバーごとに、算出メンバー名、それに関連した MDX 式、および解決順序を指定する必要があります。指定する算出メンバーの前に、現存する算出メンバー数も指定する必要があります。 *cubeName.properties* に追加する必要のある仕様の構文は次のとおりです。

```
numcalculatedmembers= integerValue
calculatedmember0.calculatedmembername = mdxMember
calculatedmember0.expression = 'mdxExpression'
calculatedmember0.solveorder = solveOrderValue
calculatedmember1.calculatedmembername = mdxMember
calculatedmember1.expression = 'mdxExpression'
calculatedmember1.solveorder = integerValue
...
```

ここで、

numcalculatedmembers
: キューブ定義に指定する算出メンバー数を、整数 (*integerValue*) で指定します。算出メンバー数が 2 である場合、*integerValue* は *2* になります。

calculatedmember#
: 算出メンバーの ID を指定します。ただし # は、ゼロ・ベースの整数値を表します。上記の構文例では、calculatedmember0 は最初の指定算出メンバーであり、calculatedmember1 は 2 番目の指定算出メンバーです。

calculatedmembername
: 指定しようとしている MDX メンバーの名前 (*mdxMember*) を指定します。

calculatedmemberexpression
: 算出メンバーを定義するための MDX 式 (*mdxExpression*) を単一引用符で囲んで指定します。

solveorder
: 指定する算出メンバーを評価するときの順序を整数値 (*integerValue*) で指定します。

以下に、在庫バックログを指定する算出メンバーの例を示してあります。

```
calculatedmember0.calculatedmembername = [Measures].[Backlog]
calculatedmember0.expression =
  '(ClosingPeriod([week_end_dt],[All Time Periods].currentmember),
     [Measures]).[BACKLOG_WK])'
calculatedmember0.solveorder = 0
```

キューブ・プロパティー仕様とそれに関連した MDX 照会の例

以下の例では、DB2 SYSCAT.TABLES 表に対して実行する単純な DB2 Alphablox キューブが作成されます。 2 つのディメンション (All Schemas および All Definers)、2 つのメジャー、および表と列があります。以下は、キューブの *cubeName.properties* ファイルに出現する算出メンバー仕様の例です。

```
numcalculatedmembers=1
calculatedmember0.calculatedmembername = [Measures].[PctOfTotal]
calculatedmember0.expression =
  '([DB2Tables].[Measures].[Tables],
  [All Schemas].currentmember)/
  ([DB2Tables].[Measures].[Tables],[All Schemas] )'
calculatedmember0.solveorder = 1
```

上記の算出メンバー仕様で使用する値を決めるには、DB2 Alphablox クエリー・ビルダー・アプリケーションを使って MDX 式を指定するのが 1 つの方法です。この例では、次のようなクエリー・ビルダーを実行します。

```
WITH MEMBER [Measures].[adHocPctOfTotal] AS
 '([DB2Tables].[Measures].[Tables],[All Schemas].currentmember)/
    ([DB2Tables].[Measures].[Tables],[All Schemas] )',
   SOLVE_ORDER = 1
SELECT
 DISTINCT( {[DB2Tables].[Measures].[PctOfTotal],
  [DB2Tables].[Measures].[adHocPctOfTotal]}) ON AXIS(0),
DISTINCT( {[DB2Tables].[All Schemas].children}) ON AXIS(1)
FROM [DB2Tables]
WHERE ([DB2Tables].[All Definers])
```

上記は、これが有効な算出メンバーの定義であることを示し、算出メンバー仕様のエレメントを *cubeName.properties* ファイルに正しく指定する際の参考になります。

この算出メンバーをキューブ・プロパティー・ファイルに追加して、キューブを再始動した後、その新規の算出メンバーを MDX 照会で使用できるようになります。照会では、その算出メンバーを明示的に指定する必要があります。その算出メンバーは、格子およびチャートには表示されますが、「メンバー・フィルター」ユーザー・インターフェースや PageBlox 選択リストには表示されません。以下は、新規の算出メンバーを使用した MDX ステートメントの例です。

```
SELECT DISTINCT({[DB2Tables].[Measures].[Tables],
 [DB2Tables].[Measures].[PctOfTotal]}) ON AXIS(0),
DISTINCT({[DB2Tables].[All Schemas]}) ON AXIS(1)
FROM [DB2Tables]
WHERE ([DB2Tables].[All Definers])
```

関数

MDX 関数は、MDX 照会の可能な有効範囲を単純化および拡張するために使用されます。 以下の表には、DB2 Alphablox キューブに対する照会でサポートされる MDX 関数がリストされています。

以下にリストされる MDX 関数の構文および使用法については、以下から入手できる情報を参照してください。

- Microsoft MDX Function Reference (http://msdn.microsoft.com/library/en-us/olapdmad/agmdxfunctintro_6n5f.asp)
- Spofford, George 著「*MDX Solutions*」John Wiley & Sons 社 (New York、2001 年)

MDX 関数	構文
Ancestor	Ancestor(<Member>,<Level>) Ancestor(<Member>,<Numeric Expression>)
Ancestors	Ancestors(<Member>,<Level>) Ancestors(<Member>,<Numeric Expression>)
Ascendants	Ascendants(<Member>)
Avg	Avg(<Set>[,<Numeric Expression>])
BottomCount	BottomCount(<Set>,<Count>[,<Numeric Expression>])
BottomPercent	BottomPercent(<Set>,<Percentage>[,<Numeric Expression>]) **注:** <Numeric Expression> はここではオプションですが、MSAS では必須です。
BottomSum	BottomSum(<Set>,<Value>[,<Numeric Expression>]) **注:** <Numeric Expression> はここではオプションですが、MSAS では必須です。
Children	<Member>.Children
ClosingPeriod	ClosingPeriod(<Level>,<Member>) **注:** <Level> および <Member> はここでは必須ですが、MSAS ではオプションです。
Count	Count(<Set>[, ExcludeEmpty | IncludeEmpty]) **注:** ここでサポートされるのは、Count(<Set>[, ExcludeEmpty | IncludeEmpty]) のみです。.Count 構文はここではサポートされません。
Cousin	Cousin(<Member1>,<Member2>)
CrossJoin	Crossjoin(<Level>,<Member>)
CurrentMember	<Dimension>.CurrentMember
DefaultMember	<Dimension>.DefaultMember
Descendants	Descendants(<Member>,[<Level>[,<Desc flags>]]) **注:** ここでサポートされるのは、Descendants(<Member>,[<Level>[,<Desc flags>]]) のみです。<set> オプションを使用する Descendants() はここではサポートされません。
Distinct	Distinct(<Set>)
DrilldownLevel	DrilldownLevel(<Set>[,{<Level>|,<Index>}])
DrilldownMember	DrilldownMember(<Set1>,<Set2>[,RECURSIVE])
DrillupMember	DrillupMember(<Set1>,<Set2>)
Except	Except(<Set1>,<Set2>[,ALL])
FirstChild	<Member>.FirstChild
FirstSibling	<Member>.FirstSibling

MDX 関数	構文
Generate	Generate(<Set1>,<Set2>[,ALL]) **注:** Generate(<Set1>,<Set2>[,ALL]) がサポートされます。 Generate(<Set>,<String Expression>[,<Delimiter>]) はここではサポートされません。
Head	Head(<Set>[,<Numeric Expression])
Hierarchize	Hierarchize(<Set>[,POST])
Hierarchy	<Member>.Hierarchy <Level>.Hierarchy
Intersect	Intersect(<Set1>,<Set2>[,ALL])
Item	<Set>.Item(Index) **注:** <Set>.Item(<StringExpression>[,<String Expression>]) および <Tuple>.Item(<Index>) はここではサポートされません。
Lag	<Member>.Lag(<Numeric Expression>)
LastChild	<Member>.LastChild
LastPeriods	LastPeriods(<Index>,<Member>) **注:** <Member> はここでは必須ですが、MSAS ではオプションです。
LastSibling	<Member>.LastSibling
Lead	<Member>.Lead(<Numeric Expression>)
Level	<Member>.Level
Max	Max(<Set>[,<Numeric Expression>])
Median	Median(<Set>[,<Numeric Expression>])
Members	<Dimension>.Members <Hierarchy>.Members <Level>.Members **注:** Members(<String Expression>) はサポートされません。
Min	Min(<Set>[,<Numeric Expression>])
Name	<Dimension>.Name <Level>.Name <Member>.Name <Hierarchy>.Name
NextMember	<Member>.NextMember
OpeningPeriod	OpeningPeriod(<Level>,<Member>) **注:** <Level> および <Member> はここでは必須ですが、MSAS ではオプションです。
Order	Order(<Set>,<Numeric Expression>[,ASC|DESC|BASC|BDESC])

MDX 関数	構文
ParallelPeriod	ParallelPeriod(<Level>,<Numeric Expression>,<Member>) **注:** <Level>、<Numeric Expression>、および <Member> はここでは必須ですが、MSAS ではオプションです。
Parent	<Member>.Parent
PeriodsToDate	PeriodsToDate(<Level>,<Member>) **注:** <Level> および <Member> はここでは必須ですが、MSAS ではオプションです。
PrevMember	<Member>.PreviousMember
Properties	<Member>.Properties(<String Expression>) **注:** ここで Properties() 関数によってサポートされるのは、ユーザー定義のメンバー・プロパティーだけです。
Subset	Subset(<Set>,<Start>[,<Count>])
Sum	Sum(<Set>,<Numeric Expression>)
Tail	Tail(<Set>[,<Count>])
TopCount	TopCount(<Set>,<Count>[,<Numeric Expression>])
TopPercent	TopPercent(<Set>,<Percentage>[,<Numeric Expression>]) **注:** <Numeric Expression> はここではオプションですが、MSAS では必須です。
TopSum	TopSum(<Set>,<Value>[,<Numeric Expression>]) **注:** <Numeric Expression> はここではオプションですが、MSAS では必須です。
Union	Union(<Set1>,<Set2>[,ALL]) Union({<Set1>,<Set2>})
UniqueName	<Dimension>.UniqueName <Level>.UniqueName <Member>.UniqueName <Hierarchy>.UniqueName

注: DB2 Alphablox でのインプリメンテーションと同様に、時間ディメンションを想定する関数 (たとえば ParallelPeriod や PeriodsToDate) は、そのディメンションに関する情報を必要としない変形だけをインプリメントするため、レベル引き数の省略は DB2 Alphablox Cube Server では現在サポートされていません。

MDX 照会の例

このセクションでは、*DB2AlphabloxCube* という名前の DB2 Alphablox キューブに対する MDX 照会の例をいくつか示します。 この例の DB2 Alphablox キューブには、以下のようなディメンション、レベル、およびメジャーが含まれると想定します。

Time (時間)	Products (商品)	Measures (メジャー)
Year {1998, 1999, 2000, 2001}	Imported {Yes, No} (輸入かどうか)	{Sales, Cost, Profit} (売り上げ、コスト、利益)
Quarter {Q1, Q2, Q3, Q4} (四半期)	Product Name {A-Z} (商品名)	
Month {1-12}		

例 1

以下の照会はいくつかのメンバー (*A*、*B*、*C*、*D*、および*Z*)を *Product Name* (商品名) レベルから列軸として選択し、行軸には *Time* ディメンションに対する Children 関数を使って年のセットを生成して、WHERE 文節で *Sales* メジャーによって照会をスライスします。

```
SELECT {[Products].[Product Name].[A]:[D],
     [Products].[Product Name].[Z]} ON COLUMNS,
       {[Time].Children} ON ROWS
FROM [DB2AlphabloxCube]
WHERE ([Sales])
```

Time (時間)	A	B	C	D	Z
2001	12.5	14.25	34.95	2,503.22	
2002					
2003					
2004					179.7

例 2

以下の照会は CrossJoin 関数を使用して、商品メンバー E と F および 1999 年の 4 つの四半期を列軸に表示します。 行軸には、DB2 Alphablox キューブの 3 つのメジャーが表示されます。

```
SELECT CrossJoin({[Products].[Product Name].[E],
     [Products].[Product Name].[F]}, [Time].[1999].Children)
     ON COLUMNS,
  {[Sales], [Cost], [Profit]} ON ROWS
FROM [DB2AlphabloxCube]
```

	E				F			
Measures (メジャー)	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Sales (売り上げ)	17,700	16,80044	44	18,100	1,413.87	1413.87	5,510	1,413.87
Cost (コスト)	12,300	12,300	50	13,200	599.97	599.97	4,400	599.97
Profit (利益)	5,400	4,500	-6	4,900	813.9	813.9	1,110	813.9

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

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

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

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

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

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

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

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

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

IBM Corporation, J46A/G4, 555 Bailey Avenue, San Jose, CA 95141-1003 U.S.A.

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

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

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

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

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

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

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

商標

以下は、IBM Corporation の商標です。

IBM	DB2	DB2 OLAP Server
DB2 Universal Database	WebSphere	

Alphablox および Blox は、Alphablox Corporation の米国およびその他の国における登録商標です。

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

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

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

C

D

E

M

R

S

プログラム番号: 5724-L14

Printed in Japan

SD88-6489-01

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