iSeries

iSeries DB2 OLAP によるビジネス・インテグレーション

iSeries

iSeries DB2 OLAP によるビジネス・インテグレーション

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： iSeries
Business Integration using DB2 OLAP on iSeries
Experience Report

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

目次

第 1 章 iCola の概要

iCola のシナリオは、IBM(R) カスタマー・ソリューション・テスト・チームが開発したテスト・シナリオです。カスタマー・ソリューション・テスト・チームは、業界全体を通じて情報技術設計者が使用している方法と同様の手法で、疑似顧客シナリオを設計、導入、評価、および展開します。 iCola のシナリオでは、データベースの機能性と同様、ビジネス・インテリジェンス製品もテストします。 特に、現在の iCola のシナリオでは、IBM DB2(R) OLAP Server(TM) に焦点を当てています。将来的には、iCola に新たなデータベース・ビジネス・インテリジェンスの機能性を組み込むことを計画しており、さらに多くのツールを追加する可能性もあります。この他の今後のトピックについては、本書の『将来のアイデア』のセクションで説明します。

このセクションでは、Online Analytical Processing (OLAP) の概要を示し、iCola のシナリオについて詳しく説明します。

このセクションは、次のサブセクションに分かれています。

- OLAP とは
- iCola シナリオの概要

OLAP とは

Online Analytical Processing (OLAP) を使用すると、直観的でしかも複合的な随時のビジネス上の質問を行うことができます。例えば、「特にフォーカスした製品について、南東部地域における第 3 四半期の収益性は」というような質問ができます。このような質問には、時間、地域、製品など、データに関する複数の視点が必要になります。このような視点の一つ一つをディメンション といいます。

リレーショナル・データは、2 つのディメンションを持つと考えることができますが、これは、それぞれのデータ (ファクト) が 1 つの行と 1 つの列 (ディメンション) に対応しているためです。マルチディメンション・データベースの各種ディメンションは、より高い水準の視点からデータを捉え、ビジネス・プランの中核のコンポーネントである、勘定 (Account)、時間 (Time)、製品 (Products)、および市場 (Markets) などを表します。 OLAP アプリケーションでは、このディメンションが変更されることはほとんどありません。

各ディメンションには個別にメンバー と呼ばれるコンポーネントがあります。例えば、時間 (Time) ディメンションのメンバーに相当するのが一年の四半期であり、製品 (Products) ディメンションのメンバーに相当するのが個々の製品です。時間 (Time) ディメンションで、四半期の下位に月が位置しているように、ディメンションにはメンバー間の階層構造があります。ユーザーのビジネスが成長するにつれて新製品や顧客が増えるように、時間の経過に伴ってメンバーは変化する傾向があります。

OLAP は発展を続けて情報技術の主流となり、今日のあらゆる業界におけるビジネス・インテリジェント・ソリューションの主要コンポーネントに成長しています。 大量のデータ分析に対する需要や、全社的に増加している従業員に対して、ビジネス上の意思決定のために十分な情報提供を行うという需要が拡大するにつれて、OLAP ソリューションの限界が近づいてきています。

IBM(R) DB2 OLAP Server(TM) は、マルチディメンションに対応した計画、分析、およびレポート作成アプリケーションをさまざまに構築できる OLAP 製品です。 IBM DB2 OLAP Server は、高速で直観的なマルチディメンション分析を行う分析アプリケーションを提供するため、ユーザーは理解しやすいビジネス用

語で質問を出すことができます。 OLAP Server は、マルチディメンション・データベースおよびリレーショナル・データベースのいずれかまたは両方の情報を計算、統合、検索するマルチディメンション要求を処理します。

DB2 OLAP Server は、Hyperion Solutions Corporation が開発した OLAP 技術に基づいています。 Hyperion Essbase および Hyperion Integration Server の参照情報は、製品のインターフェースまたは本書のあらゆる場所から得ることができます。

DB2 OLAP Server は、Hyperion Essbase のすべての機能を含んでいます。さらに、オプションとして、マルチディメンション・データベースをリレーショナル表のセットとして格納する機能が追加されています。選択するストレージ管理オプションの種類に関わらず、Essbase Application Manager および Essbase のコマンドを使用すると、 Essbase アプリケーションとその関連データベースが作成できます。これに加えて、70 種類を超える Essbase 対応ツールが独立ソフトウェア販売会社から提供されており、これらのソフトウェアからはマルチディメンション・データベースに容易にアクセスできます。

iCola シナリオの概要

iCola は、架空の中規模の飲料会社で、飲料の製造メーカーと顧客へのサプライヤーとの間に位置する中間業者です。 iCola は数年前に創業し、これまで販売データや顧客データを収集してきました。しかし、収集したデータの分析メカニズムが確立していないため、経営陣はこのデータを活用してビジネス上の意思決定をすることが出来ていません。 iCola の経営陣は、競合他社がマーケティングや販売上の意思決定の際にデータを利用してサービスの品質を向上させたり、データ分析に基づいて収益を拡大していることに気が付いています。

経営陣によるプランニング・セッションを経て、iCola 社はビジネス・インテリジェンスを有効活用することを決定しました。ビジネス・インテリジェンスとは、オペレーショナル・データから情報データベースを作成し、これを分析や意思決定に役立てるというビジネス手法上の概念です。データは複数のソースから 1 つのデータベース・サーバーに統合され、統合プロセスの一環として「トランスフォーム」されます。データの統合後、ツールを使用してこのデータを分析します。分析に使用するツールには、意思決定支援システム (Decision Support Systems (DSS))、経営情報システム (Executive Information Systems (EIS))、マルチディメンション分析ツール (Multidimensional Analysis Tools (MDA))、Online Analytical Processing (OLAP)、およびデータ・マイニング・ツールなどがあります。 iCola では、自社のビジネス・インテリジェンス・プロジェクトの手始めとして、データマート作成し、IBM(R)'s DB2(R) OLAP ソリューションを使用して分析を行うことを決定しました。

iCola 社は、iSeries(TM)、pSeries(R)、および xSeries(R) の各種ハードウェアを所有しています。 iSeries(R) で稼働している現行のオペレーショナル・システムとレポート作成オペレーションは、 pSeries サーバーおよび xSeries サーバーで実行されています。ビジネス・インテリジェンス・ソリューションを実装するための第 1 フェーズとして、iSeries システムでデータマートが作成されました。さらに DB2 OLAP サーバーを 2 番目の iSeries システムに配置し、DB2 OLAP Analyzer を Windows(R) xSeries サーバーにインストールしました。

図 1 と図 2 に、iCola シナリオのキー・システムとソフトウェアを示します。

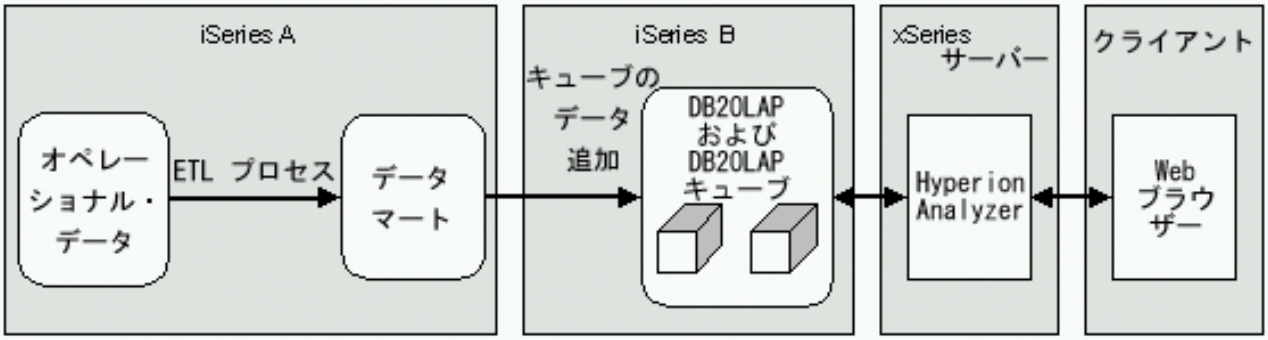

図 1: iCola シナリオの概要

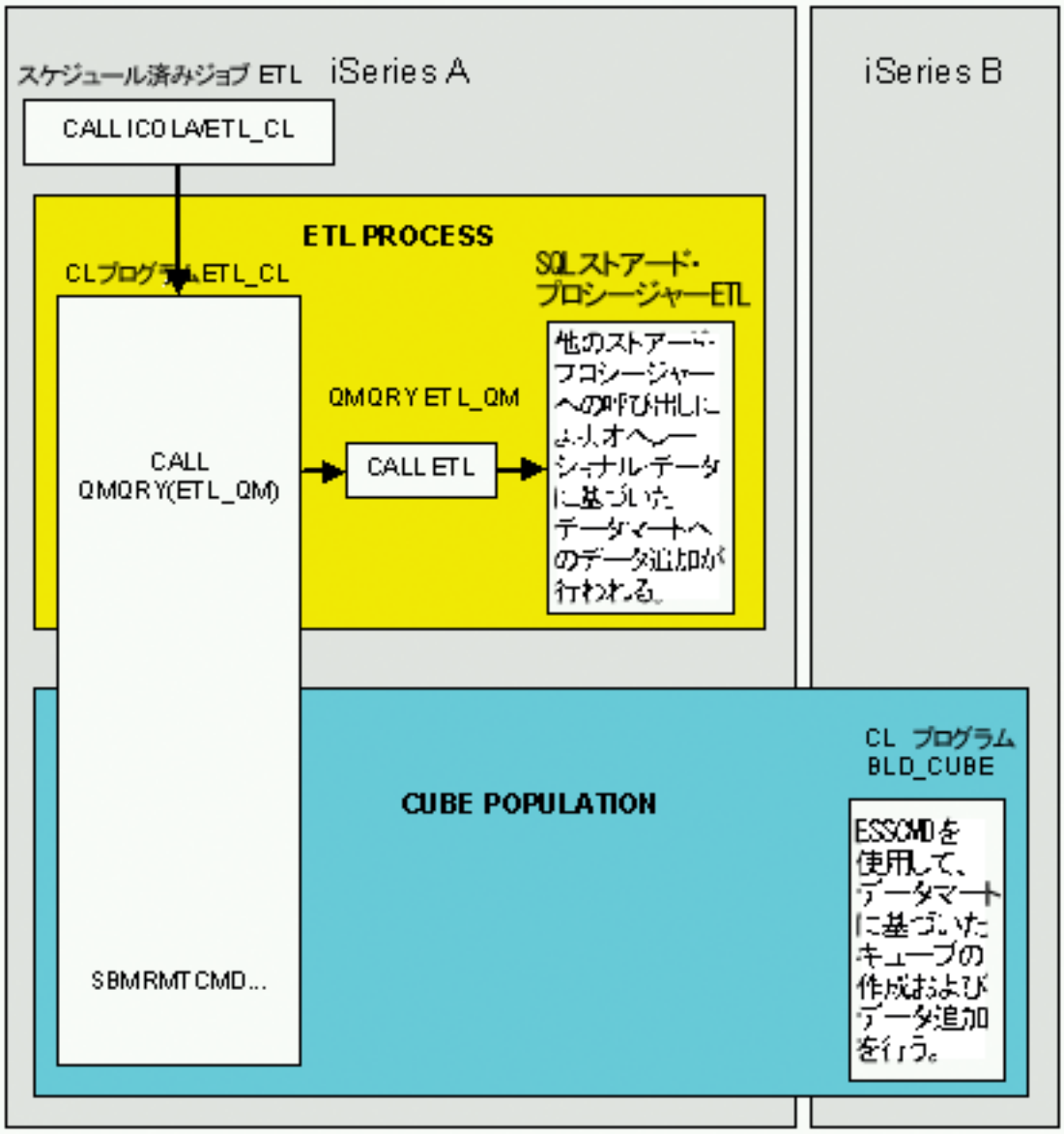

図 2: ETL プロセスとキューブのデータ追加の詳細

iSeries A には、OS/400(R) V5R2 と V5R3 のどちらを使用することもできます。しかし、iSeries B には OS/400 V5R2 を使用しなければなりません。これは、IBM DB2 OLAP Server(TM) 8.1 を実行する最新の iSeries のバージョンが OS/400 V5R2 であるため、 iSeries B は OS/400 V5R2 で実行します。

この章では、次のキー・システムおよびソフトウェアの詳細について説明します。

- データマートには、iCola シナリオのための履歴データが含まれます。
- ETL プロセスは、オペレーショナル・データをデータマートに移動させる手段です。
- 次に、データはビジネス・アナリストが使用可能なフォーマットが格納されている、マルチディメンション・キューブに追加されます。
- 最後に、iCola では DB2 OLAP Server Analyzer を使用してデータを調べ、レポートを作成します。

第 2 章 データマート

このセクションでは、iCola シナリオにおける、DB2 Universal Database(TM) for iSeries(TM) データマートのセットアップと、データウェアハウジングの概念について重点的に説明します。

日々の業務のトランザクションから収集されるデータは、オペレーショナル・データと呼ばれるものです。このデータに業務分析を行うのに有効な情報が含まれている場合があります。例えば、アナリストは、異常を見つけ出したり将来の製品販売の予想を立てるのために、年間のどの時期にどんな地域でどの製品が売れたかという情報を利用します。しかし、アナリストがオペレーショナル・データに直接アクセスすると、次のようにいくつかの問題が発生します。

- アナリストはオペレーショナル・データベースに QUERY を出すための専門知識に乏しい場合があります。例えば、情報管理システムのデータベースに QUERY を出すには、特殊なデータ操作言語を使用するアプリケーション・プログラムを実行する必要があります。一般に、オペレーショナル・データベースに QUERY を出すための技能を持つプログラマーは、データベースやそのアプリケーションを保守するフルタイムの仕事が専門であって、データ分析の専門家ではありません。
- 銀行のデータベースなど、多くのオペレーショナル・データベースにとってパフォーマンスは非常に重要です。システムはアナリストが随時出す QUERY には対処できません。
- 一般に、オペレーショナル・データは、ビジネス・アナリストが使用するのに最適なフォーマットではありません。例えば、製品、地域、時間によって要約された販売データは、アナリストにとってロー・データよりはるかに有効です。また、DB2(R) OLAP のようなツールでは、データマートの方がオペレーショナル・データよりもデータ・キューブの作成がずっと簡単にできます。
- 情報は、組織の内部にとどまらずさらに広がっていく可能性があります。アナリストは、ソースが異なるさまざまなデータではなく、集中管理されたデータにアクセスする必要があります。

データウェアハウジングによって、このような問題が解決します。データウェアハウジングは、意思決定のための理解可能情報を完全な形で、タイムリーに、また正確に管理および配信するための、プロセス、ツール、および機能を設計および実装することです。データウェアハウジングにより、情報データのストアが作成されます。このデータは、オペレーショナル・データから抽出され、ビジネス・アナリストが使用できるフォーマットに変換されます。データマートはデータウェアハウスを小規模にしたもので、iCola シナリオで使用します。データマートにはさまざまなウェアハウジング・ツールがあり、オペレーショナル・データベースからすべての販売データをコピーしたり、データを要約するための計算を行ったり、この要約データをオペレーショナル・データから個別のデータベースに書き込んだりすることができます。アナリストは、オペレーショナル・データベースのパフォーマンスに影響を及ぼさずに、個別のデータベース (データマート) に QUERY を出すことができます。

アプリケーションの概要

iCola 会社には、日常業務用のオペレーショナル・データベースと、分析用のデータマートがあります。オペレーショナル・データベースとデータマートはいずれも DB2 Universal Database(TM) for iSeries(TM) にあります。

データマートをセットアップするために必要な最も一般的なタスクは、次のとおりです。

- データマートに含める表およびフィールドを定義するためのオペレーショナル・データの探索。これらは、ビジネス・アナリストが最も興味を持つフィールドです。

- データマートのデータ用のスター・スキーマ・モデルの定義。スター・スキーマとは、ある特殊な設計になっていて、ビジネスの特徴を示す複数のディメンション表 (ディメンションには、顧客、製品、時間などがあります) および、ディメンション表のフィールドを集約したファクト表で構成されます。通常、ファクト表には、販売実績を示す数値も含まれます。

データウェアハウスおよびデータマートを作成する優れたリソースである、「IBM(R) DB2 Universal Database Business Intelligence Tutorial」は、
http://www.ibm.com/software/data/db2/db2olap/docs/V71docs/db2tu/frame3.htm#db2tussw にあります。

オペレーショナル・データベース

オペレーショナル・データベースには、iCola の日常の業務トランザクションのデータが含まれています。

このようなデータには、次のものが含まれます。

- iCola の販売する製品
- iCola が製品を販売する顧客
- iCola が製品を購入する製造業者
- iCola で採用している配送タイプ

iCola のオペレーショナル・データは、iCola がビジネス・インテリジェンスによって自社の発展を図ることを決定するはるか以前から存在していました。データマートは、オペレーショナル・データまたは業務トランザクションに影響を及ぼすことなく、このシナリオに追加されました。

データマート

通常、オペレーショナル・データは、業務用分析ツール用の最良のフォーマットにはなっていません。データにとってより優れたフォーマットは、スター・スキーマです。 iCola にとって、オペレーショナル・データとスター・スキーマとの重要な相違点は、データベース表同士の関係がより単純化されていることです。スター・スキーマは、アナリストに有用でないデータはデータマートに含まない方法でセットアップされています。分析に有用なデータは、セットアップ後に階層構造で配置されます。

アプリケーション設計のポイント

データマートの設計

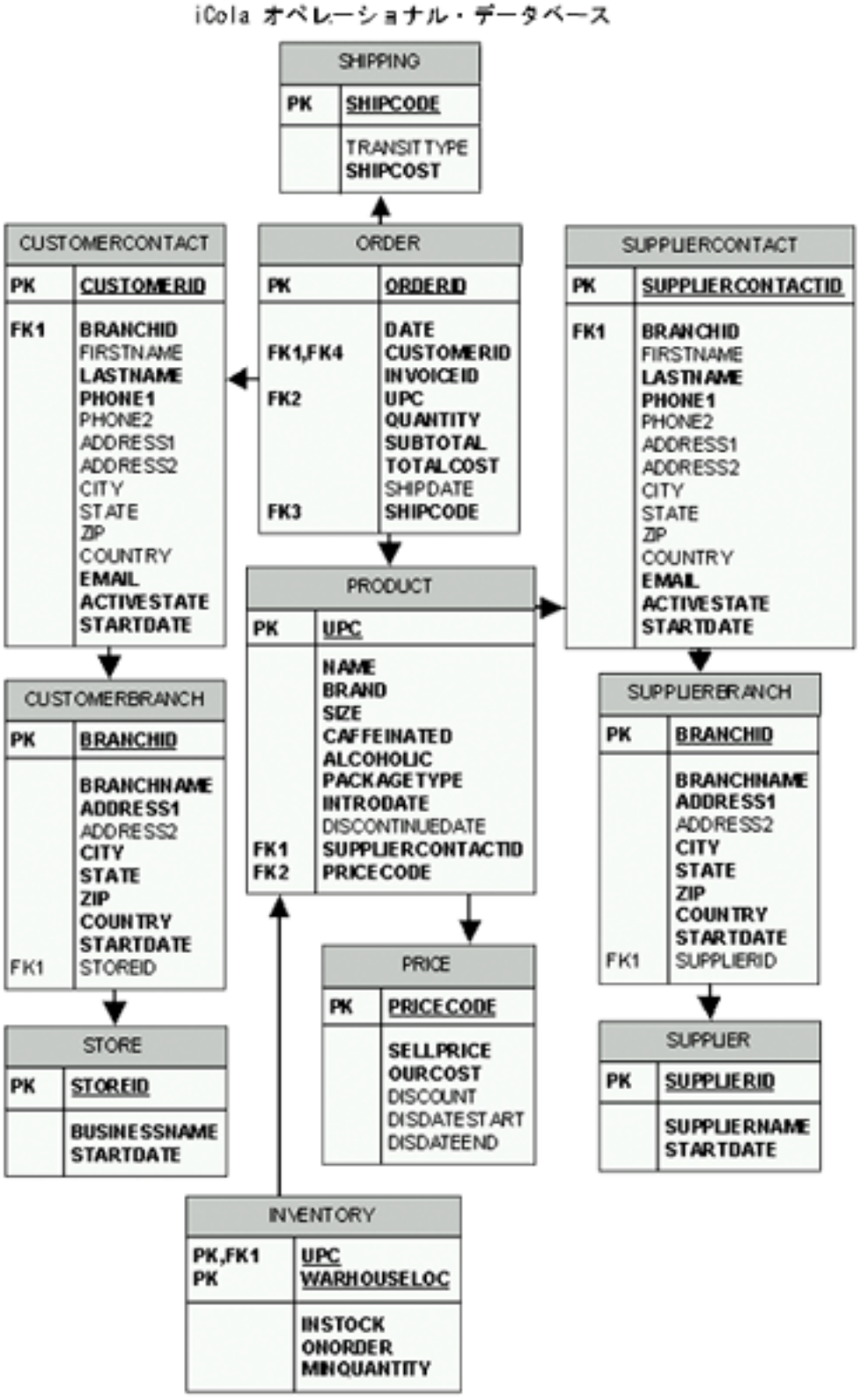

図 3 iCola オペレーショナル・データベース

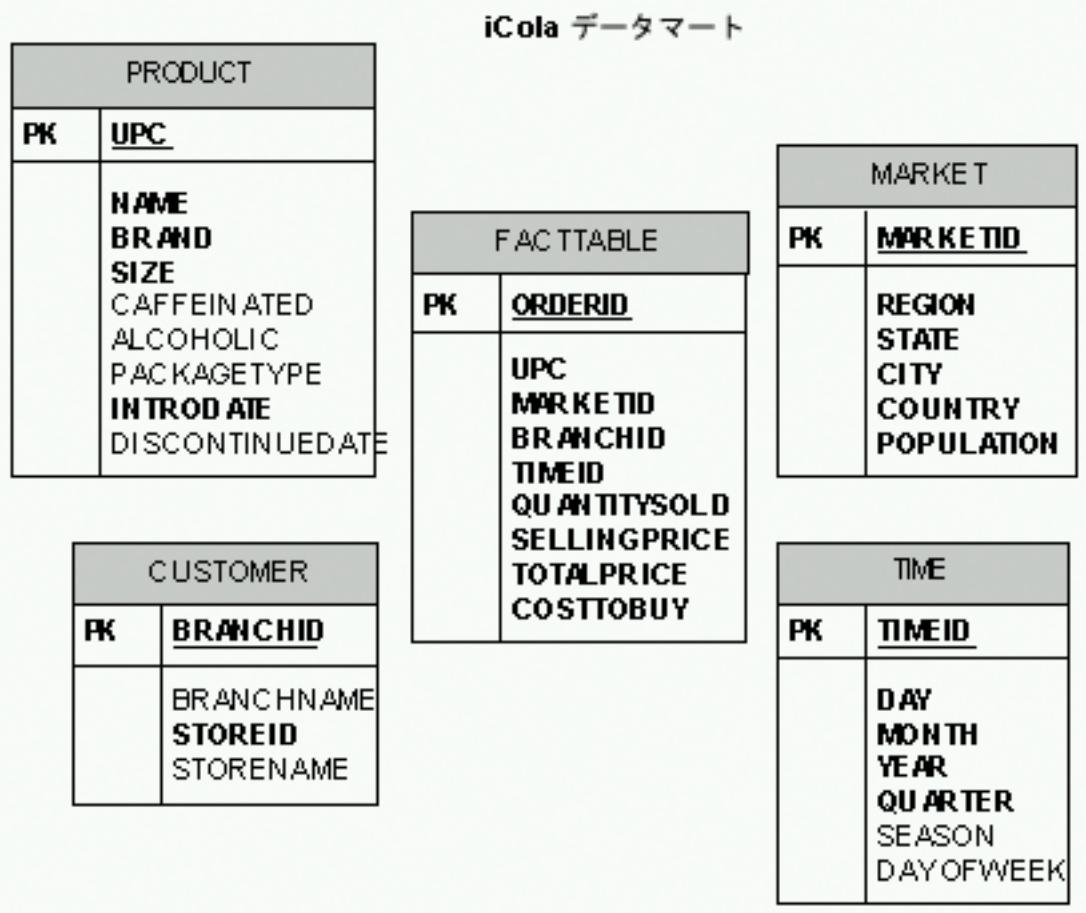

図 4 iCola データマート

データマートの設計は、オペレーショナル・データの分析、データ間の相互関係の特定、ユーザーにとって最も興味ある情報の判断から始まります。

解説書「*Designing the Star Schema*」では、尺度およびディメンションの決定の重要性について説明しています。尺度とは、測定したり追加することのできる数値のことです。尺度の例には次のようなものがあります。

- 販売データ - 発生したすべての注文
  - 販売した製品のコスト (COGS) の値
  - 売上金額
- 特定の 1 日における iCola の顧客数
- 特定の 1 日に購入された製品の平均数量

ディメンションにはこれらの尺度をどのように捉えるかに影響します。例えば、売上高に注目する場合、販売が実施された時間を示す時間 (time) ディメンションが関係します。また、販売に関する製品別または顧客別の情報も役に立ちます。

iCola のデータマート設計では次の点が考慮されました。

- 尺度

  - 販売した製品のコスト (COGS)
  - 販売実績 (製品の売上高)
- ディメンション
  - 時間 (Time)
  - 製品 (Product)
  - 市場 (Market)
  - 顧客 (Customer)

ファクト表には、データマートの各ディメンションに関する参照とともに尺度またはファクトが含まれます。ディメンションを考慮せずに尺度が有益な情報を提供することはありません。

iCola のファクト表 (FACTTABLE と呼ばれる) は、ICOLA データベースの ORDER 表を基にしています。 ORDER 表の行には尺度とディメンションの両方が含まれています。各ディメンションでは、時間、ロケーション、顧客が製品の販売に結び付けられます。

次の表は、オペレーション表 ORDER とデータマート内の FACTTABLE の相互関係を示したものです。

ORDER	**FACTTABLE**
注文ID (ORDERID)	注文ID (ORDERID)
日付 (DATE)	時間ID (TIMEID)
カスタマー ID (CUSTOMERID)	支店 ID (BRANCHID)
カスタマー ID (CUSTOMERID)	市場 ID (MARKETID)
数量 (QUANTITY)	売上高 (QUANTITYSOLD)
小計/数量 (SUBTOTAL/QUANTITY)	販売価格 (SELLINGPRICE)
小計 (SUBTOTAL)	合計価格 (TOTALPRICE)
商品コード (UPC)	商品コード (UPC)
当社コスト (OURCOST) (PRICE 表)	購入コスト (COSTTOBUY)

ディメンション表には、単一フィールドの主キーを指定する必要があります。このキーは、自動生成される増分数値で構成された ID 列である場合がよくあります。主キーを作成するのは、ディメンションの要素と FACTTABLE の行とをマッピングするために使用するからです。ディメンション表の他のフィールドには、iCola の関連データの詳細な説明が含まれています。例えば、製品 (PRODUCT) ディメンションは、ICOLA データベースの PRODUCT 表に基づいています。この表には、製品名、銘柄、サイズ、パッケージ・タイプなどの情報が含まれたフィールドがあります。データマート内のこれらのフィールドには、FACTTABLE 以外の表にリンクするコードは含まれていません。

ファクト表とディメンション表との主な相違点は、その形状にあります。

- 一般に、ファクト表には多数のレコードが含まれるが、含まれている列数は多くありません。これはファクト表に含まれるデータが尺度データまたはディメンション表に対する主キーであるためです。結果的に表の形状は細長くなります。
- 一方のディメンション表は、多数のレコードが含まれることはあまりありませんが、ディメンションに関するすべてを説明するための列が多数含まれています。結果的に表の形状は短く幅広になります。

このようなディメンションの階層により、データの「ドリルダウン」機能が可能になります。ここでは次のような QUERY が実行できます。

- 銘柄別に合計を計算する QUERY

- 各市場で購入された製品の数量を計算する QUERY
- 特定の月における各製品の合計を計算する QUERY

データマート ICOLADM には、1 つのファクト表と次の 4 つのディメンション表が含まれています。

- FACTTABLE - 市場、製品、顧客、時間に関する情報、および COGS (販売した製品のコスト) および販売データが含まれているファクト表です。この表には、主に ICOLA.ORDER 表からデータが追加されます。
- PRODUCT - 製品に関する情報が含まれているディメンション表です。この表に含まれているすべての情報は、ICOLA.PRODUCT からのものです。
- MARKET - 市場に関する情報が含まれているディメンション表です。この表は、オペレーショナル・データベースの ICOLA.CUSTOMERBRANCH 表を基にデータが追加されます。
- CUSTOMER - 顧客情報が含まれているディメンション表です。この表は、オペレーショナル・データベースの ICOLA.CUSTOMERBRANCH 表および ICOLA.STORE 表を基にデータが追加されます。
- TIME - 時間軸が含まれたディメンション表です。この表には、1 年間に相当する日付を入力する簡潔な Java(TM) プログラムによってデータが追加されます。

ICOLADM には、次のように、これ以外にもいくつかの役立つ表があります。

- REGION - 50 種類すべての都道府県の名前と対応する地域名、国名が含まれています。 (iCola が国際市場に進出する場合のために国名が含まれており、他の国の地域名と都道府県名を追加する必要があります)。ここでは MARKET 表にデータを追加するために ETL プロセスを使用します。
- ETLPROCESS - この表は直前に実施された ETL プロセスの開始と終了の時刻を示します。 ETL プロセスについては後述の説明を参照してください。

アプリケーションのセットアップ

データマート

データマート ICOLADM は、一連の SQL ステートメントを実行することで作成できます。 SQL ステートメントの詳細については、『付録 A』を参照してください。

第 3 章 ETL プロセス

オペレーショナル・データをデータマートに移動するために、管理者は、抽出 (Extract)、トランスフォーム (Transform)、ロード (Load) の ETL プロセスを使用します。一般に、この ETL プロセスは、顧客活動が少ない時間帯に (iCola シナリオでは ETL プロセスを毎日深夜 12 時に実施)、スケジュールされた間隔で実行されます。 ETL プロセスは、常にオペレーショナル・データベースから最新のデータがデータマートに追加されるようにするために欠かせない作業です。 ETL プロセスは、移動が必要なデータ量や、データをどの程度変更する必要があるかなどによって、ごく簡単なものになる場合も、非常に複雑なものになる場合があります。

ETL プロセスの詳しいステップは、次のとおりです。

1. 抽出 - 関連データをオペレーショナル・データから選択しなければなりませんが、このとき抽出したデータは、ETL プロセスをスケジュールした時間間隔に対応している必要があります。 ETL プロセスを毎日実施する場合は、通常、アプリケーションは過去 24 時間分のデータを選択します。この時点で関連データを選択しておくと、後の時間と労力を節約できます。

   また、データマートから、提供可能な量を超えるオペレーショナル・データを要求される場合もあります。次に、追加のデータが必要になった場合の iCola の ETL プロセスの例を示します。

   - iCola のデータマートは、顧客の注文に関する地域別の情報を追跡します。ミネソタ州のロチェスターにいる顧客から注文があった場合、iCola では、その顧客が中西部在住であることを把握する必要があります。データマートには、アメリカ合衆国のすべての州とその地域の REGIONS という名称の表が含まれています。 ETL プロセスの間にこの地域データは、市区町村と州 (都道府県) のデータとともに抽出され、データマートに格納されます。
2. トランスフォーム - オペレーショナル・データから選択されたデータのフォーマットが、データマートに挿入するために必要とされるフォーマットと異なる場合があります。 ETL プロセスは、必要なすべてのトランスフォーメーションに対応する必要があります。
   - iCola の ETL プロセスによるデータのトランスフォームには、次のようなものがあります。

     各注文のデータは、SQL の「DATE」フォーマットでオペレーショナル・データベースに格納されています。しかし、データマートでは、各注文の日付は、YYYYMMDD のフォーマットでなければなりません。オペレーショナル・データの日付をデータマートの日付にトランスフォームするには、次の SQL 式を使用します。

     ```
     ((YEAR(ICOLA.ORDER.DATE)*10000) + (MONTH(ICOLA.ORDER.DATE)*100) +
     DAY(ICOLA.ORDER.DATE))
     ```

   同一エンティティーを参照する有効応答を複数持つフィールドがないかどうかを検証することによって、オペレーショナル・データを「クレンジング」しなければならない場合がよくあります。例えば、ある顧客がワイオミング州の市の名前として “Mt. Horeb”, WI と入力し、別の顧客は、“Mount Horeb”, WI と入力する場合があります。 ETL プロセスでは、これら 2 つの市が同一であると判断できなければなりません。このような単純なケースでは、入力されたデータの性質が制御環境の範囲内にあるため、データのクレンジングは行われません。しかし、データのトランスフォームの方法を評価する場合は、データ・クレンジングを検討する必要があります。
3. ロード - データのトランスフォームが終了したら、データをデータマートにロードしなければなりません。この処理は SQL の INSERT ステートメントで実行できます。

ETL プロセスの実装を支援するために管理者が使用できる製品があります。このような製品の 1 つが IBM(R) DB2(R) Information Integrator です。 iCola は単純なシナリオで、iSeries(TM) で実行する ETL プロセスが必要なため、調整済みの ETL プロセスが生成されました。将来的には、iCola 社が成長した場合に IBM DB2 Information Integrator を使用して同社の ETL プロセスを実行することが検討されるかもしれません。

以降のセクションでは、iCola に関する ETL プロセスについて詳しく説明します。

アプリケーションの概要

図 5 に iCola の ETL プロセスのさまざまなコンポーネントを示します。 ETL プロセスは、iCola でスケジュールされたプロセスのほんの一部です。これに加えて、キューブのデータ追加プロセスもスケジュールされています。これについては、キューブのデータ追加に関するセクションで説明します。

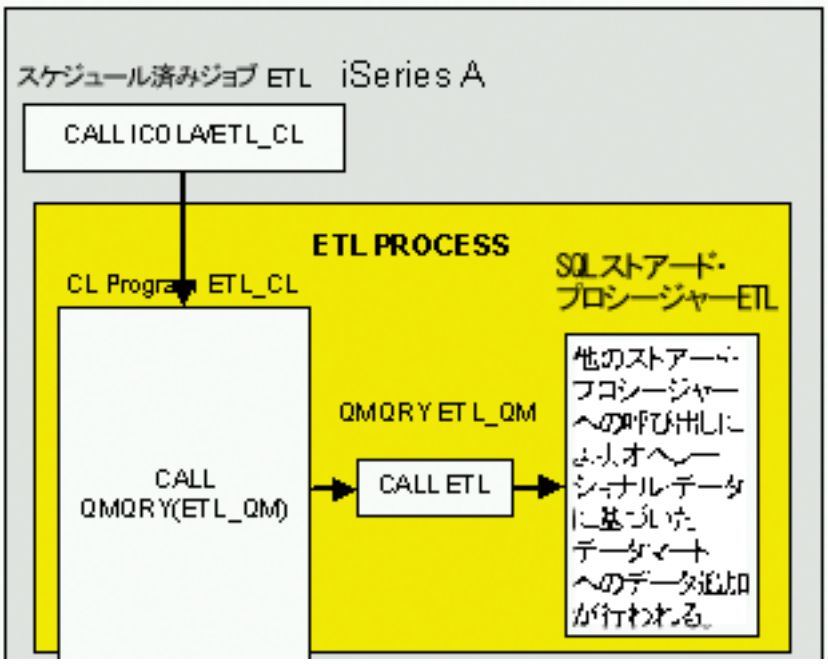

図 5: iCola の ETL プロセス (『iCola シナリオの概要』セクションの図 2 の一部であることに注意)

スケジュール済みジョブ ETL: 「ジョブ・スケジュール・エントリーの処理 (Work with Job Schedule Entries)」(WRKJOBSCDE) メニューを使用してスケジュール済みジョブを作成し、毎日深夜 12 時に ETL プロセスを自動的に開始できるようにします。スケジュール済みジョブで呼び出されるアプリケーションが ETL プログラムの ETL_CL です。

コマンド言語 (CL) プログラム ETL_CL: ETL_CL には次の 2 種類の機能があります。このプログラムの ETL の部分では、QUERY マネージャーの QUERY、ETL_QM が呼び出されます。 QUERY マネージャーの QUERY 呼び出しが完了すると、ETL_CL はキューブを含むシステム (iSeries(TM) B) に対してコマンドを実行して、キューブへのデータの追加も開始します。キューブへのデータ追加については、キューブに関するセクションを参照してください。

QUERY マネージャーの QUERY、ETL_QM: ETL_QM は SQL ストアード・プロシージャー ETL を呼び出します。 iCola で QUERY マネージャー (QM) の QUERY を使用して SQL ストアード・プロシージャーを呼び出すのは、コマンド行 (CL) プログラムからは SQL ストアード・プロシージャーを直接呼び

出せないためです。しかし、SQL ストアード・プロシージャーは QM の QUERY から呼び出すことができ、QM の QUERY は CL プログラムから呼び出すことができます。

SQL ストアード・プロシージャー ETL: ストアード・プロシージャー ETL は他の SQL ストアード・プロシージャーを呼び出します。このプロシージャーについては、『アプリケーション設計のポイント』で詳しく説明します。

- STARTETL
- CUSTOMER
- MARKET
- PRODUCT
- FACTTABLE
- ENDETL

上記の SQL ストアード・プロシージャーは、オペレーショナル・データの QUERY を実行し、データマートにデータを挿入します。

アプリケーション設計のポイント

- SQL ストアード・プロシージャーは ETL プロセスの中核です。各 SQL ストアード・プロシージャーの機能をリストすると次のようになります。
  - ETL: 他のストアード・プロシージャーを呼び出すメインのストアード・プロシージャー
  - STARTETL: ETL プロセスの開始時刻を ETLPROCESS 表の START 列に挿入します。この表はロギングに使用します。
  - ENDETL: ETL プロセスの終了時刻を ETLPROCESS 表の FINISH 列に挿入します。この表はロギングに使用します。
  - CUSTOMER: オペレーション表の CUSTOMER および CUSTOMERBRANCH の情報をデータマートの CUSTOMER 表に挿入します。
  - MARKET: オペレーション表の CUSTOMER および CUSTOMERBRANCH の地域情報をデータマートの MARKET 表に挿入します。
  - PRODUCT: オペレーション表の PRODUCT の情報をデータマートの PRODUCT 表に挿入します。
  - FACTTABLE: オペレーション表の ORDER の情報をデータマートの FACTTABLE に挿入します。

  上記のストアード・プロシージャーと、データマートを構成する各表とを比較すると、時間 (TIME) 表がないことがわかります。時間は普遍的かつ予測可能であるため、ストアード・プロシージャーを作成しなくても、単純な Java(TM) プログラムで一度に多数のエントリーを TIME 表に取り込む方法が最も簡単だからです。 TIME 表にデータを追加するためのストアード・プロシージャーの作成で困難なのは、スケジュールどおりの時刻に TIME プロシージャーを実行できなかった場合の対処法です。このような事態は、例えば、通常であればスケジュール済み ETL プロセスが実行される時刻に、システムの保存が実行されたりする場合に発生することがあります。

アプリケーションのセットアップ

iCola シナリオ用の ETL プロセスは、次のステップでセットアップされました。

1. SQL ストアード・プロシージャーの作成
2. QM QUERY の作成
3. CL プログラムの作成

4. スケジュール済みジョブの作成
5. ETL プロセスを実行するプロファイルに適切な権限があることの確認

SQL ストアード・プロシージャーの作成

各 SQL ストアード・プロシージャーを作成するために使用するコードは、付録 B に記載します。 SQL プロシージャーでは、FACTTABLE、CUSTOMER、MARKET、および PRODUCT の各プロシージャーがキーワード「EXCEPT」をプロシージャー内で使用することに注意してください。 EXCEPT キーワードは DB2 Universal Database(TM) for iSeries(TM) V5R3 の新機能です。このキーワードの機能については、図 6 を参照してください。

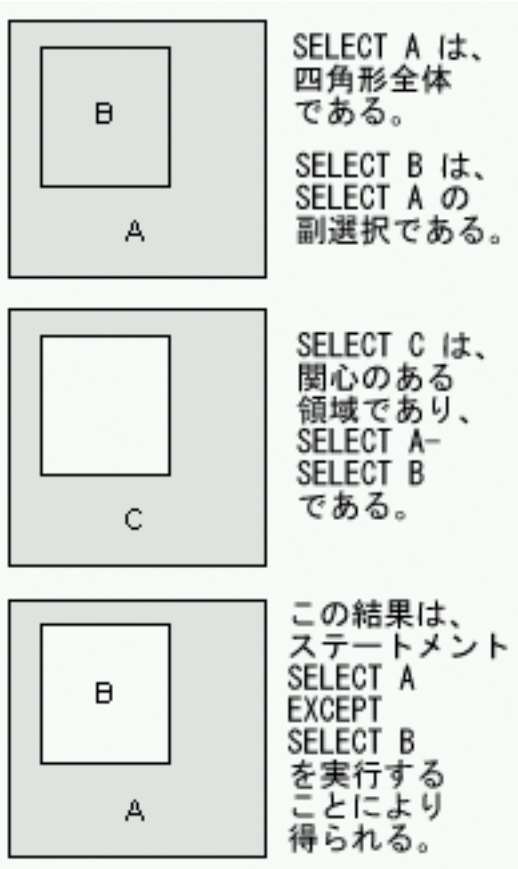

図 6: EXCEPT キーワードの機能

SELECT C の実行結果が *SELECT A - SELECT B* と同一の場合、 (*SELECT A*) EXCEPT (*SELECT B*) と実行するだけで *SELECT C* の結果を得ることができます。同様に、*SELECT B* の結果を得るには、(*SELECT A*) EXCEPT (*SELECT C*) と実行します。

EXCEPT キーワードを使用すると、このような SQL ストアード・プロシージャーを簡単に作成できます。オペレーショナル・データは広範囲に及ぶことから、最近更新されたデータ (つまり存在しないデータ) のみをデータマートに挿入する必要があります。このようなプロシージャーの作成に EXCEPT キーワードが使用できます。

SQL プロシージャーの作成で使用されるステートメントについては、『付録 B』を参照してください。付録 B には、OS/400(R) V5R2 (V5R2 では EXCEPT キーワードをサポートしないため) で実行する SQL プロシージャーの作成で使用可能なステートメントも記載しています。

QM QUERY の作成

ストアード・プロシージャーを呼び出す QM QUERY を作成するには、次のステップで行います。

1. QUERY を作成する前に、QM QUERY を作成および呼び出すプロファイルに SQL キーワードを使用する適切な権限があることを確認する。デフォルトのプロファイルには、QM 内部のみでの SELECT ステートメントの実行が許可されています。 iCola のシナリオでは、CALL ステートメントを実行する必要があります。他の SQL キーワードを使用する権限を付与するには、*SECADM 権限を持つプロファイルを使用してサインオンしてください。
2. STRQM を実行して QM インターフェースを開始する。
3. 「**QUERY マネージャーのプロファイルの処理 (Work with Query Manager Profiles)**」を実行するため、オプション 10 を選択する。
4. リストから適切なプロファイルを選択し、オプション 2 を選択してプロファイルの属性を変更する。
5. パラメーターを下にスクロールして、「**許可された SQL ステートメントを選択 (Select allowed SQL statements)**」で「Y」を選択する。
6. 次の画面に、さまざまな SQL ステートメントのキーワードが表示されます。特定のキーワードを使用する権限を付与するには、そのキーワードの隣に「1」と入力します。
7. ユーザー・プロファイルに CALL ステートメントの実行権限があることを確認したら、メインの QM の開始画面に戻り、オプション 1 を選択する。
8. 画面上部のライブラリー・パラメーターに、QUERY を作成する所要のライブラリーを入力して Enter キーを押す。入力したライブラリーに既存の QM QUERY がある場合は、それらが表示されます。
9. ライブラリーの下に表示されているのが **Query Creation Mode** パラメーターです。 CALL ステートメントを実行する QUERY を作成するには、**Query Creation Mode** を SQL にする必要があります。**Query Creation Mode** が PROMPT に設定されている場合は、F19 (shift + F7) で SQL に切り替えてください。
10. **Query Creation Mode** が SQL であることを確認したら、上部のオプション・フィールドにオプション 1 を入力して新規の QM QUERY を作成する。
11. 画面の大部分が空白の QUERY 作成画面が表示されます。この画面で CALL ステートメントを指定します。 `CALL ICOLA/ETL` と入力して CALL ステートメントを実行します。
12. QM QUERY の作成が終了したら、F3 を押して変更内容を保管し、QUERY に名前を付ける。

CL プログラムの作成

QM QUERY を呼び出すために使用する CL プログラムを作成するには、次のステップで行います。

1. まず、次のコマンドを実行して CL プログラムを格納するソース・ファイルを作成する。

   ```
   >CRTSRCPF FILE(ICOLA/QCLSRC) RCDLEN(92) TEXT('SOURCE FILE FOR CL')
   ```

2. STRPDM と入力してプログラム開発マネージャーを表示する。オプション 3 を選択してメンバーを処理します。「**メンバーの指定 (Specify Members)**」画面で、ICOLA ライブラリーの QCLSRC ファイルを入力します。
3. 「**メンバーの処理 (Work with Members)**」画面で、F6 を押して新規メンバーを作成する。メンバーにタイトル (ETL_CL) を付け、使用するソース・タイプとして CLLE を指定します。
4. CL プログラムにコードを入力するための原始ステートメント入力ユーティリティー (SEU) 画面が表示されます。 iCola のシナリオでは、次のコードを入力しました。

PGM
/* Call to QM Query */
STRQMQRY QMQRY(ICOLA/ETL_QM)
/* Populate the cube on iSeries B */
SBMRMTCMD CMD('SBMJOB CMD(CALL PGM(ICOLA/BLD_CUBE))') DDMFILE(ICOLA/TO_SD)
/* Print to the job log *:
DSPJOBLOG OUTPUT(*PRINT)
ENDPGM

5. CL プログラム・コードの入力が終了したら、F3 を押して変更を保存する。
6. 「**PDM を使用したメンバーの処理 (Work with Members using PDM)**」画面から CL プログラムをコンパイルするため、メンバーの隣にオプション 14 を入力する。プログラムが正しくコンパイルされたか判断するには、 WRKSPLF コマンドを実行してスプール・ファイルを作成して確認します。

スケジュール済みジョブの作成

スケジュール済みジョブを作成して CL プログラムを実行するには、次のステップで行います。

1. コマンド行で WRKJOBSCDE を実行してジョブのスケジュール済みエントリーを処理する。
2. F6 を押して新規のスケジュール済みジョブを作成する。 iCola のスケジュール済みジョブが作成されると、毎日深夜 12 時に実行されるスケジュール済みジョブを作成するための次のパラメーターがリストされます。
   - ジョブ名 : ETL
   - 実行するコマンド : CALL PGM(ICOLA/ETL_CL)
   - 頻度 : *WEEKLY
   - スケジュール済み日付 : *NONE
   - スケジュール済み曜日 : *MON、*TUE、*WED、*THU、*FRI、*SAT、*SUN
   - スケジュール済み時刻 : 00:00:00
   - ユーザー : DB2OLAP

権限

ETL プロセスを構成するプログラムやプロシージャーを作成する場合、 ETL プロセスを実行するプロファイルにすべての新規オブジェクトに対する適切な権限があるかどうか確認することが重要です。 iCola のシナリオでは、スケジュール済みジョブを実行するプロファイルは DB2OLAP です。作成済みのすべてのオブジェクトが同一ライブラリー (ICOLA) にあるため、次のようにコマンドを実行します。

- 次のコマンドは DB2OLAP に、所有権および iCola ライブラリーに対する全権限を付与します。
  CHGOWN OBJ('/QSYS.LIB/ICOLA.LIB') NEWOWN(DB2OLAP)
- 次のコマンドは DB2OLAP に、所有権および iCola ライブラリー内のすべてのオブジェクトに対する全権限を付与します。
  CHGOWN OBJ('/QSYS.LIB/ICOLA.LIB/*') NEWOWN(DB2OLAP)

主要な発見事項

SQL キーワード TRUNC

ETL プロセスの SQL QUERY を作成する場合、注文の合計価格情報と単価情報が必要でした。この情報を得るために、当初使用していたのは、次の公式です。

```
(ICOLA.ORDER.SUBTOTAL / ICOLA.ORDER.QUANTITY)
```

除法の結果、小数点以下に複数の桁が発生します。しかし、iCola データマートでは、小数点以下を 2 桁にする必要があります。計算後の小数点以下が 2 桁だけになるように、上記の公式ではなく、次の公式を使用します。

```
TRUNC (ICOLA.ORDER.SUBTOTAL / ICOLA.ORDER.QUANTITY , 2 )
```

TRUNC キーワードの 2 番目のパラメーターに、小数点以下の桁数 (このケースでは 2) を指定することにより、公式は指定された桁数の数値だけを残してそれ以外はすべて切り捨てます。

FACTTABLE プロシージャーで選択するデータ

もう一つの主要な発見事項は、ストアード・プロシージャー FACTTABLE に関するものです。次に、このストアード・プロシージャーを作成する SQL を示します。

```
CREATE PROCEDURE ICOLA.FACTTABLE ( )
LANGUAGE SQL
SPECIFIC ICOLA.FACTTABLE
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN INSERT INTO ICOLADM . FACTTABLE ( ORDERID , UPC , MARKETID , TIMEID,
QUANTITYSOLD , SELLINGPRICE , TOTALPRICE , COSTTOBUY , BRANCHID )
( ( SELECT ICOLA . ORDER . ORDERID , ICOLA . ORDER . UPC , ICOLADM . MARKET .
MARKETID , ( ( YEAR ( ICOLA . ORDER . DATE ) * 10000 ) + ( MONTH ( ICOLA . ORDER .
DATE ) * 100 ) + DAY ( ICOLA . ORDER . DATE ) ) , ICOLA . ORDER . QUANTITY , TRUNC (
ICOLA . ORDER . SUBTOTAL / ICOLA . ORDER . QUANTITY, 2 ) , ICOLA . ORDER . SUBTOTAL ,
ICOLA . PRICE . OURCOST , ICOLA . CUSTOMERCONTACT . BRANCHID
FROM ICOLA . ORDER , ICOLA . PRODUCT , ICOLA . PRICE , ICOLA . CUSTOMERCONTACT,
ICOLA . CUSTOMERBRANCH , ICOLADM . MARKET
WHERE ICOLA . ORDER . CUSTOMERID = ICOLA . CUSTOMERCONTACT . CUSTOMERID
AND ICOLA . CUSTOMERCONTACT . BRANCHID = ICOLA . CUSTOMERBRANCH . BRANCHID
AND ICOLA . CUSTOMERBRANCH . CITY = ICOLADM . MARKET . CITY
AND ICOLA . CUSTOMERBRANCH . STATE = ICOLADM . MARKET . STATE
AND ICOLA . ORDER . UPC = ICOLA . PRODUCT . UPC AND ICOLA . PRODUCT . PRICECODE =
ICOLA . PRICE . PRICECODE
AND ICOLA.ORDER.DATE < CURRENT DATE)
EXCEPT
( SELECT ORDERID , UPC , MARKETID , TIMEID , QUANTITYSOLD , SELLINGPRICE,
TOTALPRICE , COSTTOBUY , BRANCHID FROM ICOLADM . FACTTABLE ) ) ;
END ;
```

元々、このプロシージャーでは、where 文節に AND ICOLA.ORDER.DATE < CURRENT DATE という条件は含まれていませんでしたが、次のような状況を考慮して追加されました。

- 深夜 12 時の ETL プロセスの実行中、または、CUSTOMER プロシージャーの実行後に、新規の顧客がサイトに登録して、FACTTABLE プロシージャーの実行前に発注する場合があります。

このような状況が発生すると、この注文に関する顧客情報がデータマートに挿入されなくなります。結果的に、データ・キューブの構築時にエラーが発生します。キューブの構築ではすべてのディメンション・データを必要とするためです。この問題の解決策は AND ICOLA.ORDER.DATE < CURRENT DATE という where 文節を SQL ステートメントに追加することでした。 ETL プロセスは深夜 12 時に実行するように

スケジュールしているため、深夜 12 時から FACTTABLE プロシージャーが実行されるまでの間に出された注文だけが FACTTABLE に含まれません。深夜 12 時から FACTTABLE プロシージャーの実行までの時間はほんの数分間なので、FACTTABLE に挿入されない注文は数件だけです。この時点で挿入されなかった注文は翌日に挿入されます。

第 4 章 マルチディメンション・キューブ

iCola シナリオの中核となるのは、履歴要約データを保持するために使用するマルチディメンション・データベース・キューブです。このキューブは、IBM(R) DB2 OLAP Server で作成できます。 IBM DB2 OLAP Server(TM) を使用すると、高速で直観的に理解できるマルチディメンション分析が可能になります。

データベース・キューブは、製品、時間、勘定など、販売アナリストが関心事と考えるさまざまなディメンションで構成されています。この 3 種類のキューブを使って作成した例が、図 7 です。キューブを作成するために使用したディメンションは、データマートから抽出されたデータに基づいている必要があります。

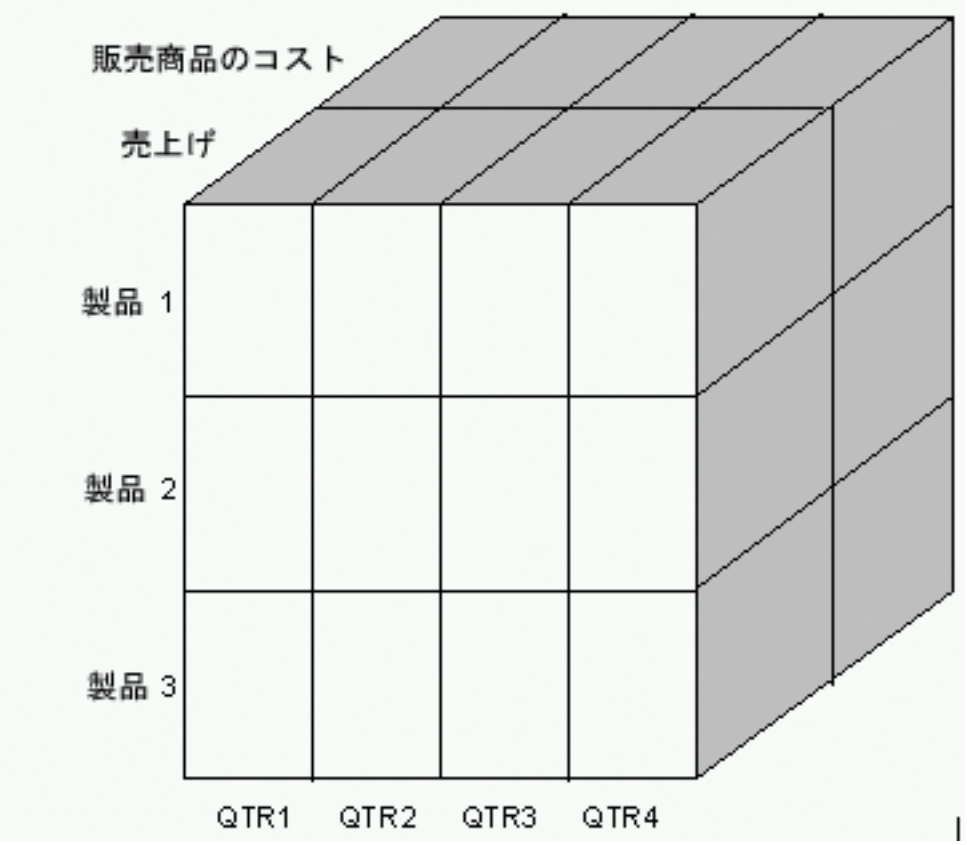

図 7: マルチディメンション・キューブ

例えば、製品、時間、勘定の各ディメンションでキューブを組み立てた場合、次のような QUERY をキューブから抽出することができます。

- 第 1 四半期に収益マージン (販売高 - 経費) が最大だった製品はどれか。
- 製品 1 の収益マージンが最低だったのはどの四半期か。
- 年間を通じて最大の初期経費を必要とした製品はどれか。

第 1 四半期に最も収益マージンが高かった製品を判定する最初の例に注目してください。このデータを表示するために調べる必要があるのは、キューブの 1 スライス (QTR1 のスライス) だけです。同様に、年間を通じた製品 1 の販売高を表示するために調べる必要があるのは、キューブの製品 1 というスライスで

す。年間を通じた全商品の販売コストを表示するには、販売した商品のコストのスライスを調べる必要があります。幸い、ユーザーはキューブのどのスライスまたはセクションにデータが格納されているか知らなくても、このような QUERY の結果を生成することができます。ユーザーは、IBM DB2 OLAP Server Analyzer 8.1 を使用して関心を持っているデータを指定するだけで、自動的に結果が生成されます。

先にリストした QUERY も、同一データが含まれているこの iCola のオペレーショナル・データベース (リレーショナル・データベース) を使用して生成できます。一方、この種の QUERY または他の QUERY 生成の複雑さは、マルチディメンション・キューブを使用すると大幅に減らすことができます。マルチディメンション・データベースには、多くの異なる視点からデータをすばやく調べることができるという強みがあります。オペレーショナル・データがデータベースのさまざまな場所に散在することはよくあり、QUERY を実行してこれらのオペレーショナル・データを収集することは不可能です。

第 1 四半期の収益をオペレーショナル・データベースから収集するには、次の計算を QUERY で実行する必要があります。

1. 第 1 四半期にどのような注文が出されたかを判別する。
2. 第 1 四半期の注文ごとに、品目の販売コストと販売量を乗算した販売データを判別する。
3. 第 1 四半期の注文ごとに、製品コストと数量を乗算した販売商品のコスト・データを判別する。
4. 第 1 四半期の注文ごとに、ステップ 2 の結果からステップ 3 の結果を減算して収益マージンを判別する。
5. 注文があったうちの販売された製品に基き、ステップ 4 の結果を加算して、各製品の収益マージンを決定する。

IBM DB2 OLAP Server を使用することで、エンド・ユーザーは、上記の QUERY や他の多くの QUERY の計算処理を意識せずに実行することができます。

アプリケーションの概要

キューブのディメンションの選択

iCola シナリオのキューブは次の 5 種類のディメンションに基づいています。

- 時間 (TIME)
- 尺度 (MEASURES) (勘定 (ACCOUNTS))
- 製品 (PRODUCT)
- 市場 (MARKET)
- ストア (STORE)

注: TIME および MEASURES が基本的なキューブのディメンションです。データ・キューブは、履歴要約データを追跡するため、TIME ディメンションが常に存在します。同様に、データ・キューブは、少なくとも 1 つの数値による数量 (販売高や売り上げた数量など) を追跡するのが普通です。この情報は、MEASURES ディメンションで追跡されます。

iCola キューブに PRODUCT、MARKET、および STORE の各ディメンションが含まれる理由は、次のような疑問に対する答えを企業が必要としているからです。

- 一年のうちの特定の時期やパッケージの種類が特定の銘柄の販売にどのような影響を及ぼすか。
- 特定の銘柄の販売高が最高と最低の地域別ストアはどこか。
- 特定の製品は大規模なチェーン・ストアより小規模ストアでよく売れるのか。

- さまざまな製品の中から地域別またはストア別に一対一の製品で販売を比較するとどのような結果になるか。

キューブのアウトラインでのディメンションの拡張

キューブのディメンションを選択したら、これらのディメンションは、各ディメンションに追加するデータに基づいてキューブのアウトラインの中で拡張しなければなりません。通常、各ディメンションには複数の特性が含まれます。例えば、MARKET ディメンションについて考えてみます。それぞれの市場のロケーションには、地域、都道府県、および市区町村があります。地域の中に都道府県が、都道府県の中に市区町村がそれぞれ含まれることから、MARKET ディメンションには次のような階層が構成されます。

- 市場 (Market) (世代 1/レベル 3)
  - 地域 (Region) (世代 2/レベル 2)
    - 都道府県 (State) (世代 3/レベル 1)
      - 市区町村 (City) (世代 4/レベル 0)

これらの階層は世代またはレベルとしても知られています。世代番号は、ディメンション内の統合レベルを参照し、ディメンションのルートからリーフに向かうにつれて数字が増えていきます。レベル番号もディメンション内のブランチを参照しますが、レベルでは、リーフのメンバーからルートに向かってカウントするため、世代とは使用する数字の順序が逆になります。

これらの世代/レベルを MARKET ディメンションで定義することにより、複数の市場に関するデータを抽出することができます。例えば、ロケーションを基準にして該当する年における製品の成功度を判断するために、キューブは市区町村、都道府県、地域別に計算できます。

キューブへのデータのロード

マルチディメンションのキューブには、定期的に更新しなければならないデータのカテゴリーが 2 種類あります。一般に、このようなカテゴリーは次のように、ディメンションの構築およびデータ・ロードとして知られています。

- ディメンションの構築 : PRODUCT、MARKET、および STORE の各ディメンションにアウトラインのデータを入力します。このデータは、iCola シナリオに新規の製品、市場、およびストアを追加するために必要となるデータです。例えば、MARKET ディメンションでは、iCola で市場ベースが拡大されたときに新しく市区町村が追加される可能性があります。
- データ・ロード : 個々の注文から抽出される次のような販売データです。
  - 購入された製品
  - その製品の購入数量
  - その製品が購入された時刻
  - その製品を購入した顧客

  このデータはキューブの計算に必要であり、これによって先に一覧した販売上の疑問に答えることができます。

これらのカテゴリーの値を収集するために、DB2 OLAP Server(TM) Application Manager でロード・ルールを作成します。このロード・ルールは、キューブにデータを追加するために IBM(R) DB2(R) OLAP Server が使用するものです。キューブにデータを追加する場合に、ディメンションの構築ルールとロード・ルールは常にデータのロード・ルールより先に実行しなければなりません。これは、キューブがデータのロード中に入力される可能性のあるすべてのディメンション・データの詳細を把握しておく必要があるためです。キ

ューブがすべての追加可能なデータ (例えば、新規ストアのデータが、ストアがディメンションの構築ルールによって追加される前にデータ・ロード・ルールで入力されるなど) を把握していないと、キューブが適切に構築されません。

データの両カテゴリーはデータマートから抽出されます。データを抽出するために、次のロード・ルールが作成されます。

- ディメンション構築ルール :
  - MRKT-RUL : MARKET 表から地理的な情報を抽出して MARKET のアウトラインを完成させる
  - PROD-RUL : PRODUCT 表から製品情報を抽出して PRODUCT のアウトラインを完成させる
  - STOR-RUL : STORE 表からストア情報を抽出して STORE のアウトラインを完成させる
- データ・ロード・ルール :
  - EXP-2004 : FACTTABLE からデータ・ロード用の経費情報を抽出する
  - SAL-2004 : FACTTABLE からデータ・ロード用の販売情報を抽出する

ロード・ルールの詳細については、『アプリケーションのセットアップ』セクションを参照してください。

データ・ロードのプロセスは、ETL プロセスを実行するために使用したものと同じ CL プログラムを使用すると自動化されます。 iSeries(TM) A で ETL プロセスが完了した後、先に一覧したロード・ルールを使用してキューブにデータを追加するプログラムが iSeries B で呼び出されます。 (iSeries A および iSeries B については、図 2 を参照してください)。

アプリケーション設計のポイント

ディメンションの考慮事項

- IBM(R) DB2 OLAP Server(TM) は、標準的なキューブのディメンションを高密度と低密度の 2 種類に分けることによりパフォーマンスを最大化し、ストレージを最小化します。低密度のディメンションとは、使用可能なデータ位置の充てん率が低い傾向にあるディメンションを指します。同様に、高密度のディメンションは、使用可能なデータ位置の充てん率が高い傾向にあるディメンションです。この区分により、IBM DB2(R) OLAP Server は、データへのマトリックス方式のアクセスによる利点を損なうことなく、円滑な分散が行われていないデータを適切に処理することができます。

  販売業務は毎日のように実施されることが多いことから、高密度ディメンションの典型的な例として、時間 (TIME) ディメンションを挙げることができます。一方、低密度ディメンションの例には、市場 (MARKET) を挙げることができます。製品はすべての市場で販売されるわけではないからです。

  iCola の高密度ディメンションは次のとおりです。
  - 時間 (TIME)
  - 尺度 (MEASURES) (勘定 (ACCOUNTS))

  iCola の低密度ディメンションは次のとおりです。
  - 製品 (PRODUCT)
  - 市場 (MARKET)
  - ストア (STORE)

アウトラインの考慮事項

- キューブを作成する際に考慮すべき重要な点として、会社の販売上の疑問点に応えるためにどの程度詳細な情報をキューブに含めなければならないか、ということがあります。時間 (TIME) ディメンション

については、データを計算するために必要な最小単位の時間の尺度を判別しなければなりません。 iCola のシナリオでは、時間の最小単位を 1 月にすれば会社の分析上のニーズが満たされるという判断が下されました。この結果、時間ディメンションの最低レベルは月 (month) になりました。

- もう一つの決定事項は、キューブにどのような数値データを含めるかということでした。販売実績などのデータがこれに相当し、通常、勘定 (ACCOUNT) ディメンションとして表されます。実行する必要のある QUERY に基づいて、会社は経費および販売データを使用することを決定しました。
- 図 8 は、製品 (PRODUCT)、市場 (MARKET)、およびストア (STORE) の各ディメンションがキューブのアウトラインに展開されている様子を示したものです。

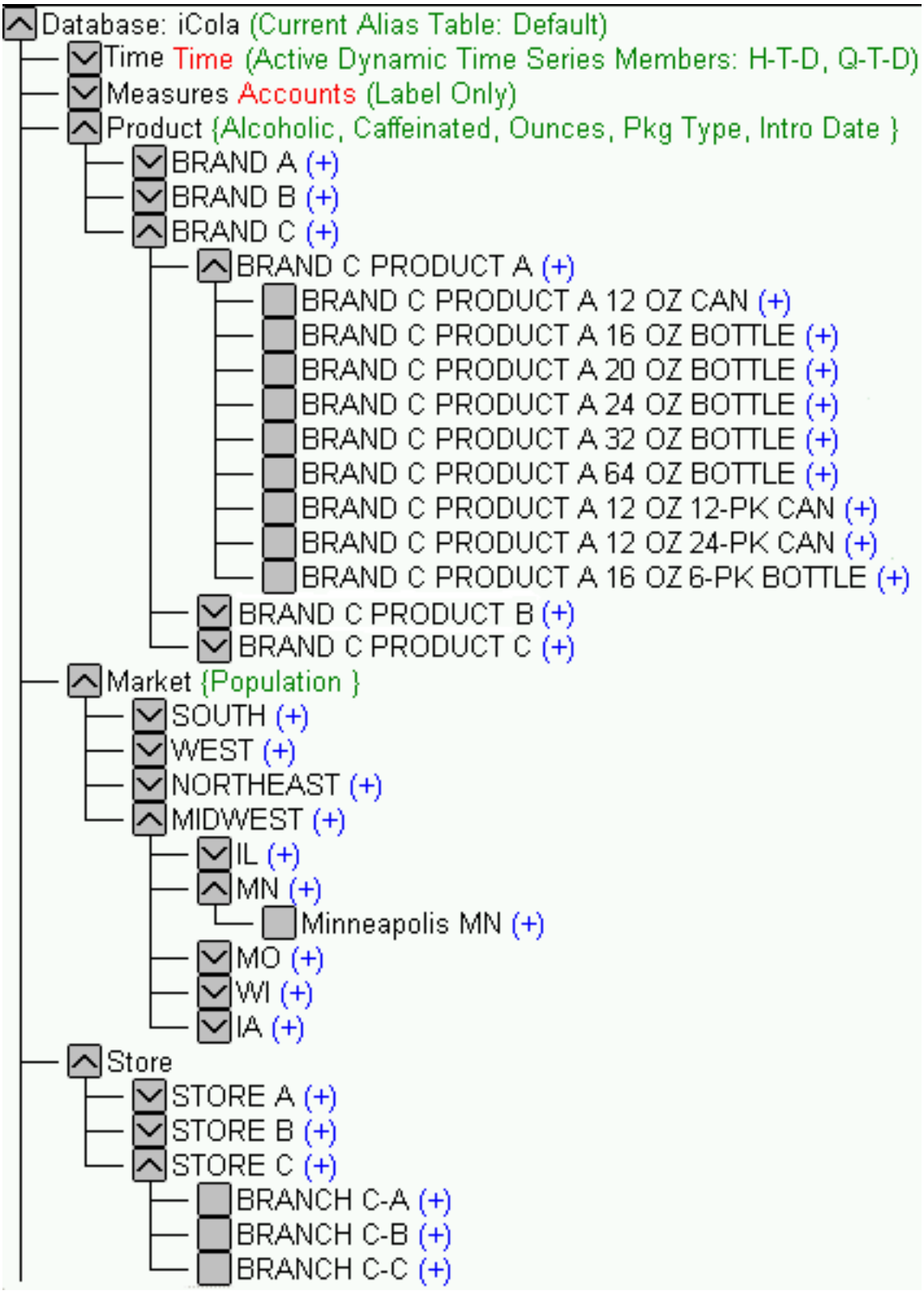

図 8: iCola の 製品 (PRODUCT)、市場 (MARKET)、ストア (STORE) ディメンション

- 各製品には、サイズ、パッケージのタイプ、(市場への) 投入日などの特徴があります。この特徴をキューブのアウトラインに反映させる最も効果的な方法は、この属性を製品 (PRODUCT) ディメンションに指定することです。属性の作成について詳しくは、「*IBM DB2 OLAP Server 8.1 Database* Administrator's Guide」を参照してください。

パフォーマンスの考慮事項

- 「*IBM DB2 OLAP Server 8.1 Database Administrator's Guide*」で説明しているパフォーマンス上のポイントは、低密度ディメンションより先に高密度ディメンションへの **QUERY** のパフォーマンスを最適化することです。また、ここでは、最も多く QUERY が出される低密度ディメンションの順序を QUERY の最少の低密度ディメンションより先にするように指摘しています。このガイドラインを iCola のシナリオに当てはめると、ディメンションの順序は次のようになります。

  1. 時間 (TIME) (高密度)
  2. 尺度 （MEASURES）(高密度)
  3. 製品 (PRODUCT) (低密度)
  4. 市場 (MARKET) (低密度)
  5. ストア (STORE) (低密度)

  しかし、ここで指摘しているもう一つのパフォーマンス上のポイントは、**計算**のパフォーマンスを最適にすることです。高密度ディメンションの順序を低密度ディメンションより前にするが、低密度ディメンションでは、メンバーの最少のディメンションから始めて最大のディメンションへという順序にします。このガイドラインを iCola のシナリオに当てはめると、ディメンションの順序は次のようになります。

  1. 時間 (TIME) (高密度)
  2. 尺度 （MEASURES）(高密度)
  3. 市場 (MARKET) (低密度)
  4. ストア (STORE) (低密度)
  5. 製品 (PRODUCT) (低密度)

  この 2 種類の見解には矛盾する部分があります。「*IBM DB2 OLAP Server 8.1 Database Administrator's Guide*」では、アウトラインの最適な順序付け手順は、高速 QUERY と高速計算でどちらがより重要かどうかで優先順位を付けることであると説明しています。 iCola では、計算は 1 日に 1 度しか実施しないのに対して、QUERY は日に何度も実施されることを主な理由に、QUERY のパフォーマンスが計算のパフォーマンスより重要度が高いと判断しています。

- もう一つのパフォーマンスの考慮事項として検討されているのは、多重キューブの使用です。 iCola のシナリオでは、1 つのキューブを数年分に相当するデータの保管に使用することがあります。しかし、iCola では、キューブが大型化するにつれ、キューブの計算や QUERY を生成する時間が長くかかるようになってきました。これを解決する方法の一つに、キューブのパーティション化がありますが、iCola でも将来、この方法を導入するかもしれません。現在のところ、年度別のキューブを使用することで対応しています。キューブに保管されている情報は月別になっているため、年ごとに異なるキューブを使うことが会社にとって現実的な解決策であると判断したためです。現在、iCola のシナリオでは、2003 年のデータと 2004 年のデータをそれぞれ保管した 2 つのキューブを使用しています。

アプリケーションのセットアップ

IBM(R) DB2 OLAP Server(TM) のインストール

iSeries(TM) および iSeries の操作に慣れているユーザーにとっては、 iSeries に IBM DB2 OLAP Server をインストールする作業は分かりやすいはずです。このインストール・ガイドでは、インストール・プロセス全体を通して説明していきます。このガイドでは、iCola シナリオで使用する iSeries 内の OLAP キューブを格納するために必要な DB2 OLAP Server インスタンスの作成手順について広範囲に説明しています。DB2(R) OLAP Server Installation Guide for iSeries は、「DB2 OLAP Server Installation Guide for iSeries」にあります。本書の第 2 章、『Before you install』および第 4 章、『Installing Essbase/400 OLAP server components』には DB2 OLAP Server をインストールするために必要となる重要な説明があります。

IBM DB2 OLAP クライアントのインストール

「DB2 OLAP Server Installation Guide」では、 Windows(R) に DB2 OLAP クライアント・アプリケーションをインストールする方法について説明しています。第 6 章、『Installing clients on Windows』では、インストールのプロセスについて説明し、第 7 章、『Connecting to Essbase/400 OLAP server』では、クライアント側およびサーバー側の各アプリケーション間でリンクを確立してクライアントを使用できるようにするための説明をします。インストール手順のガイドは、「DB2 OLAP Server Installation Guide for iSeries」にあります。

キューブの作成

マルチディメンション・キューブを作成するには、まず DB2 OLAP サーバーが始動されていることを確認してください。 ESSBASE コマンドを実行する権限をプロファイルに付与するには、次のように入力します。

- ESSBASE/GRTESSAUT USRPRF(*user_profile*)

次に、このプロファイルを使用して DB2 OLAP を開始するには次のコマンドを実行します。

- ESSBASE/STRESSSVR SERVER(*ALL) JOBD(OLAPBLDSVR/SCJOBD)

DB2 OLAP Server の始動後に、「スタート」>「プログラム」>「IBM DB2 OLAP Server 8.1」> 「アプリケーション・マネージャー」の順にクリックすると、クライアント・アプリケーションを開始できます。

IBM DB2 OLAP Server を使用する場合は、主に次の用語を理解する必要があります。

- アプリケーション: データベース、計算スクリプト、ロード・ルール、レポートが含まれているプロジェクト。
- データベース:「*IBM DB2 OLAP Server 8.1 Database Administrator's Guide*」では、データベースをキューブと称しています。

「*IBM DB2 OLAP Server 8.1 Database Administrator's Guide*」の 151 ページに、アプリケーションおよびデータベースの作成についての詳しい説明があります。

ディメンションにアウトラインを追加する手順については、このガイドの 168 ページを参照してください。アウトラインを作成する最も簡単な方法は、IBM DB2 OLAP Server に用意されているサンプル・データベースの 1 つ (Sample アプリケーションの Basic データベースなど) をコピーすることです。 iCola では、「サンプル (Sample)」 > 「基本 (Basic)」の順でアウトラインをコピーしてから、次のように変更します。

- シナリオのディメンションを削除します。この情報は不要なためです。
- 企業はストア情報を追跡する必要があるため、ストアのディメンションを追加します。
- iCola の市場は Sample の Basic 市場とは地域が異なるため、東部 (East)、中部 (Central)、南部 (South)、西部 (West) から、北東部 (NorthEast)、中西部 (Midwest)、南部 (South)、西部 (West) に変更します。

- iCola の製品は、Sample アプリケーションの製品とは異なるため、製品ディメンションの製品を削除します。
- アルコール飲料であることを明記するために属性「alcoholic」を既存属性のリストに追加します。アウトラインに属性を追加する手順については、ガイドの 249 ページを参照してください。

アウトラインではディメンションを大幅に拡張する必要はありません。アウトラインは、ディメンションに構築ロード・ルールが作成および実行されると、データが入力されます。**ロード・ルールの作成 - ディメンションの動的構築**

前述したように、3 つのディメンション構築ロード・ルールが作成されたことで、アウトラインは、新規の市場、製品、およびストアが追加されたときに最新の状態を保つことができるようになりました。ディメンションを動的に構築するロード・ルールを作成するには、次の手順で行います (詳細については、「*IBM DB2 OLAP Server TM 8.1 Database Administrator's Guide*」の第 19 章を参照してください)。

1. DB2 OLAP Server のクライアント画面でデータ・ロード・ルール・ボタン

   

   を選択して新規ルールを作成し、「**新規**」をクリックする。
2. ルール・ファイルを構築するために使用する SQL ステートメントを定義するには、「**ファイル**」> 「**SQL を開く (Open SQL)**」と選択する。
3. 適切なサーバー、アプリケーション、およびデータベースを選択し、「OK」をクリックする。
4. 適切なデータ・ソースを選択し、提供されたスペースのデータマートからディメンションの構築情報を抽出するための SQL ステートメントを入力する。「**OK**」または「**検索 (Retrieve)**」をクリックして完了します。ユーザー名およびパスワード情報が必要な場合は、DB2 OLAP の管理者ユーザー名と DB2 OLAP Server のパスワードを入力します。
5. 「**データ準備エディター (Data Prep Editor)**」が表示されます。 SQL ステートメントでデータの検索に成功すると、そのデータがエディターに表示されます。
6. ルールは、アウトラインと関連付けられていなければなりません。アウトラインとの関連付けを行うには、メインメニューから、「**オプション**」> 「**アウトラインに関連付ける (Associate Outline)**」と選択します。適切なサーバー、アプリケーション、およびデータベースを選択し、「**OK**」をクリックします。
7. 「**データ準備エディター (Data Prep Editor)**」画面の「**プロパティーの定義 (Define Properties)**」ボタン

   

   をクリックしてアウトラインのフィールド・プロパティーを完成させる。
   a. ディメンションを動的に構築するためのロード・ルールを作成する場合は、「**グローバル・プロパティー**」タブのすべての列で、次のプロパティーについて検証する。
      - 「**データ・フィールド**」ボックスにはチェック・マークを付けません。このルールはデータをロードするためのものではないため、チェック・マークを付けません。
      - 「**データのロード中はフィールドを無視する (Ignore field during dataload)**」ボックスにはチェック・マークを付けます。このルールは、ディメンションを変更するために使用されるもので、データをロードするためのものではないため、チェック・マークを付けます。

- 「**ディメンションの構築中はフィールドを無視する (Ignore field during dimension build)**」ボックスにはチェック・マークを付けません。このルールはディメンションを変更するルールであり、無視できないため、チェック・マークを付けません。
- 「**先頭/末尾の空白を切り捨てる (Drop leading/trailing whitespace)**」ボックスにはチェック・マークを付けます。これでフィールドに余分な空白があってもディメンションの構築に影響が生じることがなくなります。

b. 「**データのロード中はフィールドを無視する (Ignore field during dataload)**」ボックスにはチェック・マークを付けないため、ユーザーは「**データ・ロード・プロパティー (Data Load Properties)**」タブを選択できません。しかし、ユーザーは、すべてのフィールドをディメンションのアウトラインにマッピングするために「**ディメンション構築プロパティー (Dimension Build Properties)**」タブを選択でき、また、この選択は必須です。

- 「**フィールド・タイプ**」では、「世代 (Generation)」を選択し、適切な世代番号を「**番号 (Number)**」ボックスに入力します。
- 「**ディメンション (Dimension)**」については、「**ディメンション (Dimension)**」ドロップダウンから適切なディメンションを選択します。

属性をフィールドにマッピングするには、次のようにします。

- 「**フィールド・タイプ**」については、「属性のディメンション (Attribute Dimensions)」下にリストされている「**フィールド・タイプ**」ドロップダウン・リストから属性フィールド名を選択する。
- 「**番号 (Number)**」ボックスには、当該ディメンションの最下位レベルの世代番号を入力してください。
- 「**ディメンション (Dimension)**」では、属性が定義するディメンションを選択する。

8.

ボタンをクリックして、「**ディメンションの構築設定 (Dimension Build Settings)**」を検証する。「**ディメンションの構築設定 (Dimension Build Settings)**」タブで、「**構築方法 (Build Method)**」が「**世代参照を使用 (Use Generation References)**」になっていることを確認します。

9. ルール・ファイルを保管する。

ディメンションの構築ルールの作成方法の詳細を理解するために、iCola および MRKT-RUL のディメンションの構築ルールの作成方法を次に示します。

1. DB2 OLAP のサーバー画面で、データ・ロード・ルール・ボタン

を選択し、「新規」をクリックする。

2. 次に、ルール・ファイルの構築に使用する SQL ステートメントを定義するために、「**ファイル**」>「**SQL を開く (Open SQL)**」と選択する。
3. サーバー iSeries B (iSeries A および iSeries B については図 2 を参照)、アプリケーション iCola、およびデータベース 2004 が選択されます。

4. データ・ソースとして iSeries A を選択し、次の SQL ステートメントを入力する。 SELECT ICOLADM.MARKET.REGION, ICOLADM.MARKET.STATE, CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE)) FROM ICOLADM.MARKET

   この SQL ステートメントを使用して MARKET ディメンションの各世代/レベルの情報を抽出します。
5. 「**OK**」または「**検索 (Retrieve)**」を押すと、ユーザー名およびパスワードの入力を求めるプロンプトが表示されます。
6. 図 9 のように「**データ準備エディター (Data Prep Editor)**」が表示されます。ここで、単なる市区町村名 (city) を抽出するのではなく、市区町村と都道府県が連結されていることに注意してください。これは、すべてのディメンションのデータ要素がデータのロード時に固有でなければならないためです。例えば、Rochester, MN (ミネソタ州ロチェスター) の情報を抽出する場合、Rochester という町の名前だけを抽出すると、 DB2 OLAP Server からは、この町が Rochester, MN であるのか Rochester, NY であるのかをデータのロード時に区別できません。市区町村名と都道府県名を連結することで、市区町村 (city) フィールドは固有になります。

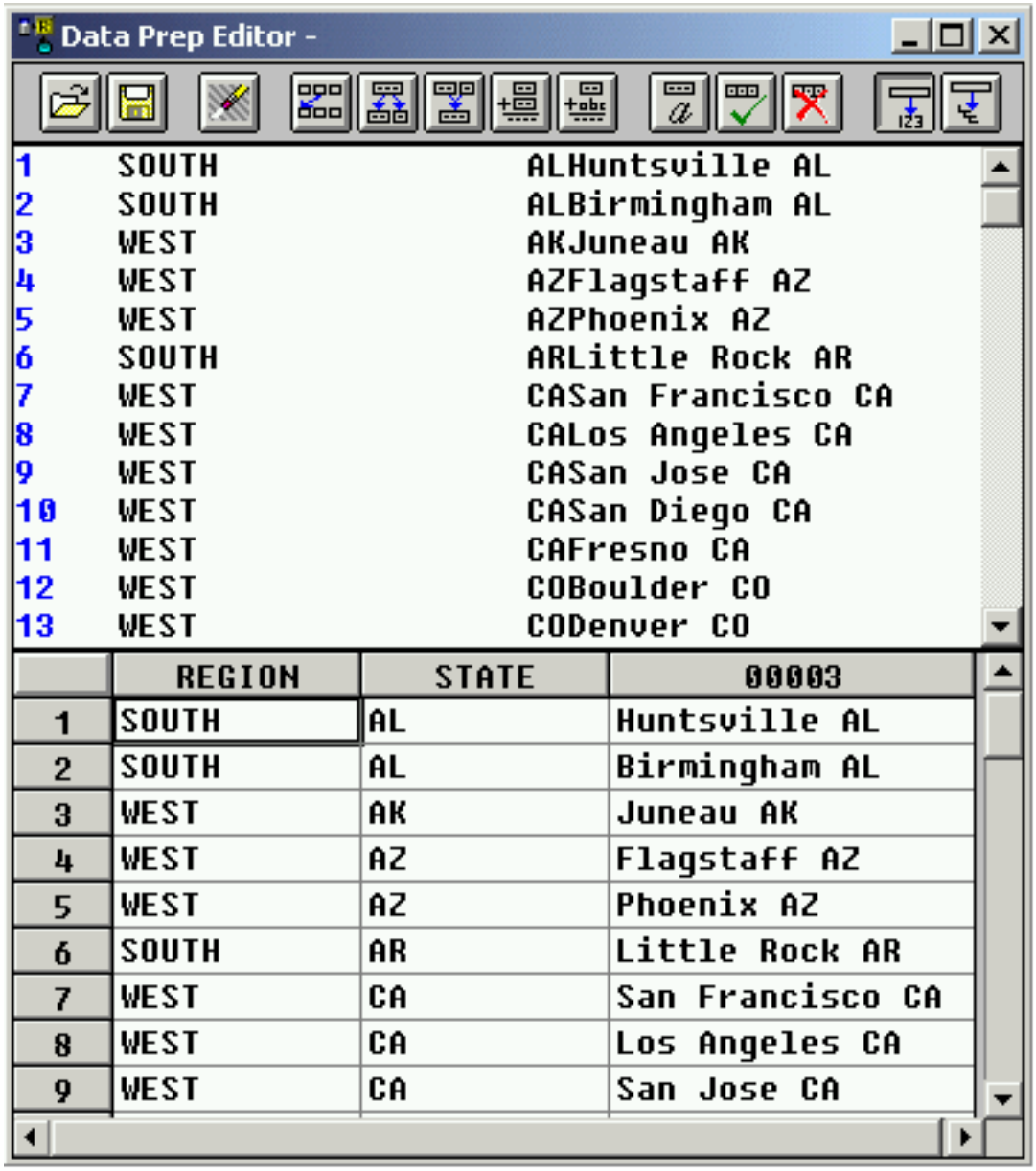

図 9: MARKET ディメンションのフィールド

7. 次に、アウトライン 2004 にルールを関連付ける。
8. フィールド・プロパティーは次のように指定する。
   - 「**グローバル・プロパティー**」タブの適切なフィールドにチェック・マークを付ける。
   - 図 10 から 12 は、「**ディメンションの構築プロパティー (Dimension Build Properties)**」タブの各世代に入力される値を示したものです。 ここで入力される値が、『アプリケーションの概要』のセクションで説明した世代情報と対応していることに注意してください。

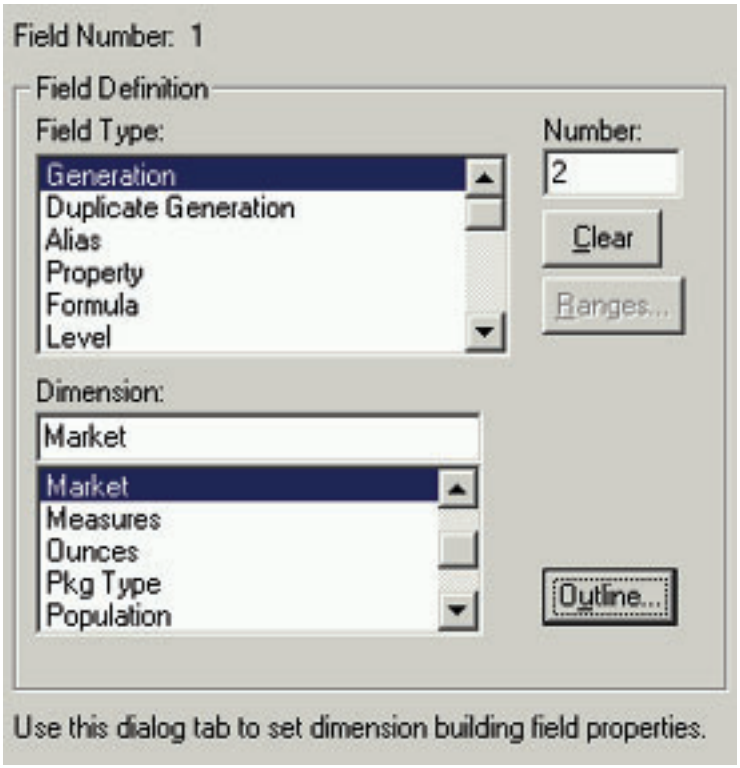

図 10: 地域フィールドの定義

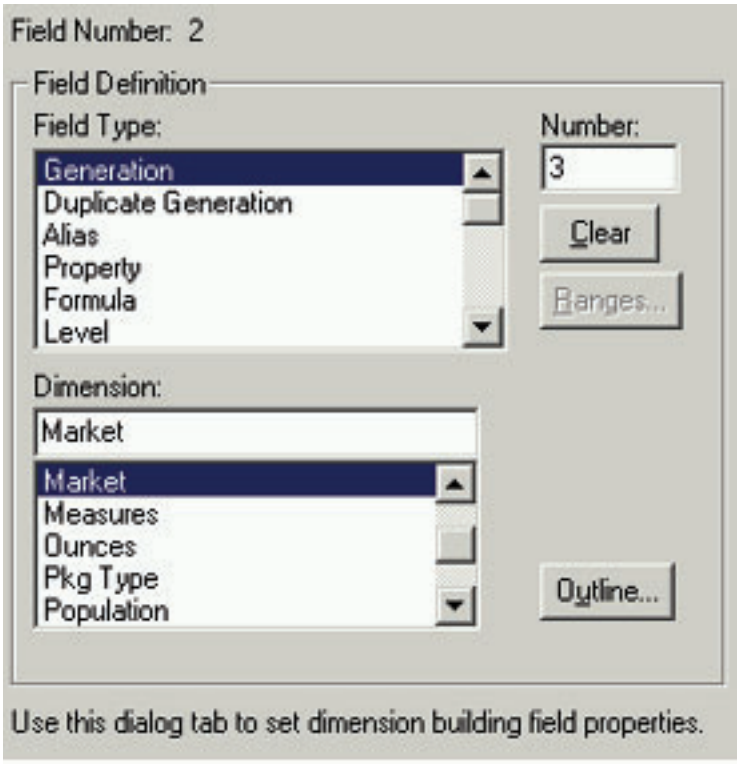

図 11: 都道府県フィールドの定義

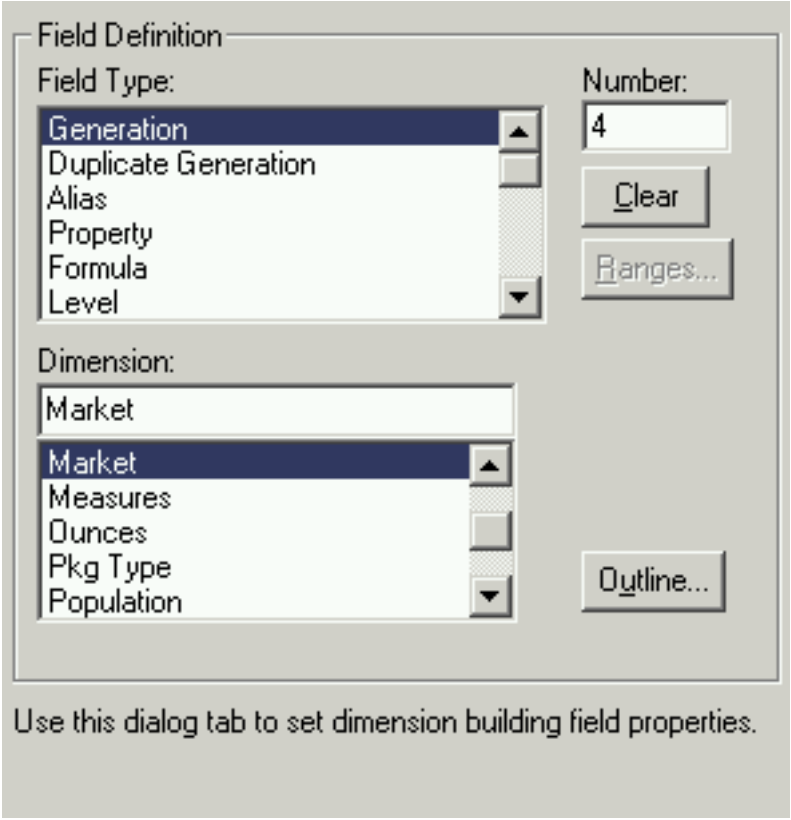

図 12: 市区町村フィールドの定義

9. 「**ディメンションの構築設定 (Dimension Build Settings)**」タブで「**世代参照を使用 (Use Generation References)**」という設定にチェック・マークを付ける。
10. ルール・ファイルが保管されます。

iCola のディメンションの構築ルールの作成で使用した SQL ステートメントについては、『付録 C』を参照してください。

ロード・ルールの作成 - データ・ロード

ディメンションとともに、キューブに保持するデータも常に現行のものでなければなりません。現行のデータをロードするためにロード・ルールを作成します。データ・ロード・ルールを作成するには、次のステップに従います (詳細については、「*IBM DB2 OLAP Server 8.1 Database Administrator's Guide*」の第 20 章を参照してください)。

1. ディメンションの動的構築に関するセクションのステップ 1 から 6 を実行してロード・ルールを作成する。

   現行のデータをロードするために使用する SQL ステートメントには、各ディメンションの最高位の世代番号 (最下位のレベル番号ともいう) が含まれます。例えば、iCola のデータ・ロード・ルールの場合、SQL ステートメントには次の内容が含まれます。

   - 販売活動のあった月
   - 数値が経費または販売実績のどちらであるか
   - 販売された製品
   - 販売が行われた市区町村

- 製品が販売されたストア

SQL ステートメントには販売の合計金額も含まれます。これは数量的なデータで、この後のステップで使用します。

データ・ロード・ルールを作成するために iCola で使用した SQL ステートメントについては、『付録 C』を参照してください。

2. アウトラインのフィールド・プロパティーを完成させる。「**データ準備エディター (Data Prep Editor)**」画面で、「**プロパティーの定義 (Define Properties)**」

ボタンをクリックする。

a. データ・ロード・ルールの作成時に、「**グローバル・プロパティー**」タブで、ロードする数量的なデータ (通常は数値) が含まれているフィールドを**除く**すべての列の以下のプロパティーを検証する。
   - 「**データ・フィールド**」ボックスにはチェック・マークを付けません。特定のフィールドにデータが含まれるわけではないため、チェック・マークを付けません。
   - 「**データのロード中はフィールドを無視する (Ignore field during dataload)**」ボックスにはチェック・マークを付けません。このルールはデータをロードするために使用するルールであり、無視できないため、チェック・マークを付けません。
   - 「**ディメンションの構築中はフィールドを無視する (Ignore field during dimension build)**」ボックスにはチェック・マークを付けます。このボックスにチェック・マークを付けるのは、このルールがデータをロードするためのものであってディメンションを構築するためのものではないためです。
   - 「**先頭/末尾の空白を切り捨てる (Drop leading/trailing whitespace)**」ボックスにはチェック・マークを付けます。これでフィールドに余分な空白があってもディメンションの構築に影響が生じることがなくなります。

   数量的なデータを含むフィールドでは、「**データ・フィールド**」ボックスにチェック・マークを付けます。その他のすべてのフィールドは直前のリストと同じにします。

b. 「ディメンションの構築中はフィールドを無視する (Ignore field during dimension build)」ボックスにはチェック・マークを付けないため、ユーザーは「**ディメンションの構築プロパティー (Dimension Build Properties)**」タブを選択できません。ただし、ユーザーは、「**データ・ロード・プロパティー (Data Load Properties)**」タブを選択し、データをロードするフィールドに名前を付けることができます。このステップは、不要な場合もあります。

3.

ボタンをクリックして、「**ディメンションの構築設定 (Dimension Build Settings)**」にチェック・マークを付けます。「**ディメンションの構築設定**」タブで、「**構築方法 (Build Method)**」が「**世代参照を使用 (Use Generation References)**」になっているかどうか確認します。

4. ルール・ファイルを保管する。

iCola のディメンションの構築ルールの作成に使用した SQL ステートメントについては、『付録 C』を参照してください。

データの消去

データをキューブにロードする前に、キューブを初期化して適切な計算ができるようにする必要があります。キューブを初期化するには、メインメニューから「データベース」>「データの消去 (Clear Data)」>「すべて」と選択します。

ディメンション構築ルールとデータ・ロード・ルールを手動で実行する

ディメンションの構築ルールとデータ・ロード・ルールの作成後、ルールを実行してデータベースを変更しなければなりません。ルールを手動で実行するには、次のステップに従います。

1. メインメニューから、「**データベース**」>「**データのロード (Load Data)**」と選択する。
2. アウトラインを構築するサーバー、アプリケーション、およびデータベースを選択する。
3. タイプとして SQL を選択する。
4. オプションでは、アウトラインを変更する場合は、「**アウトラインの変更 (Modify Outline)**」にチェック・マークを付け、「**データのロード (Load Data)**」にはチェック・マークを付けません。データをロードする場合は、「**アウトラインの変更 (Modify Outline)**」にはチェック・マークを付けず、「**データのロード (Load Data)**」にチェック・マークが付いていることを確認してください。
5. SQL ユーザーおよびパスワードでは、DB2 OLAP 管理者ユーザー名およびパスワードを入力します。
6. 「**ルール・ファイルを使用 (Use Rules File)**」ボックスにチェック・マークを付け、検索ボタンを使用して適切なルールを選択する。
7. 「OK」をクリックする。新規のエラー・ファイルを追加または上書きするかどうかを尋ねるプロンプトが表示される場合があります。必要に応じて「はい」か「いいえ」を選択してください。

データの計算

キューブへのロード後、データを計算しなければなりません。データを計算するには、「データベース」>「計算」の順に選択します。選択したサーバー、アプリケーション、データベース、計算スクリプトが適切か確認してください (計算スクリプトは、キューブにデータの計算方法を指示します。スクリプトには、計算コマンド、式、公式を含めることができます。計算スクリプトが作成されていない場合は、デフォルトのスクリプトが使用できます)。「OK」をクリックしてデータを計算します。

ディメンションの構築とデータ・ロードのプロセスを自動化する

データの消去、アウトラインの変更、データのロード、データの計算などのプロセスは毎日実行しなければならないため、これらのプロセスを自動化するための BLD_CUBE と呼ばれる CL プログラムを作成します。 iSeries 用の DB2 OLAP Server のコマンドを使用して BLD_CUBE を作成します。 BLD_CUBE は ETL プロセスの完了後に呼び出されます。

時間 (TIME) ディメンションの最小の尺度は月 (month) であることから、iCola のシナリオでは、BLD_CUBE プログラムを毎日実行します。このプログラムを 1 日ベースで実行することで、会社は販売の半月分析を行うことができます。

BLD_CUBE に含まれるコードは次のとおりです。

```
PGM
/* Log onto DB2 OLAP with the DB2 OLAP administrator ID and password */
ESSBASE/LOGINESS SVRUSER(DB2OLAP) SVRPW(PASSW0RD)
/* Choose the 2004 database under the ICOLA application*/
ESSBASE/RUNESSCMDC COMMAND('SELECT "ICOLA" "2004"')
/* Clear the database */
ESSBASE/CLRESSDB APPNAME(ICOLA) DBNAME(2004)
```

```
/* Build the dimensions using the rules files */
ESSBASE/BLDESSDIM APPNAME(ICOLA) DBNAME(2004) RULEFILE('MRKT-ALL') ERRFILE(ERROR)
SQLUSER(DB2OLAP) +
SQLPW(PASSW0RD)
ESSBASE/BLDESSDIM APPNAME(ICOLA) DBNAME(2004) RULEFILE('PROD-ALL') ERRFILE(ERROR)
SQLUSER(DB2OLAP) +
SQLPW(PASSW0RD)
ESSBASE/BLDESSDIM APPNAME(ICOLA) DBNAME(2004) RULEFILE('STOR-ALL') ERRFILE(ERROR)
SQLUSER(DB2OLAP) +
SQLPW(PASSW0RD)
/* Load the data using the rules files */
ESSBASE/IMPESSSQL APPNAME(ICOLA) DBNAME(2004) RULEFILE('SAL-2004') +
ERRFILE('/ESSBASE/APP/ICOLA/2004/ERROR.TXT') SQLUSER(DB2OLAP) SQLPW(PASSW0RD)
ESSBASE/IMPESSSQL APPNAME(ICOLA) DBNAME(2004) RULEFILE('EXP-2004') +
ERRFILE('/ESSBASE/APP/ICOLA/2004/ERROR.TXT') SQLUSER(DB2OLAP) SQLPW(PASSW0RD)
/* Calculate the data */
ESSBASE/CLCESSDFT APPNAME(ICOLA) DBNAME(2004)


ENDPGM
```

主要な発見事項

キューブのデータ追加プロセスをリモート側で呼び出す

キューブをロードするプログラムは、スケジュールされた ETL プロセスの一部です。オペレーショナル・データおよびデータマートは、iSeries(TM) A に、キューブは iSeries B にそれぞれ格納されます。 ETL プロセスが iSeries A で完了した後、プログラムが iSeries B で呼び出され、キューブにデータが追加されます。元々、iSeries B には、キューブを構築するためのスケジュールされたプログラムがあります。ただし、ETL プロセスがいつ終了したかを判定するのは難しく、データマートは成長し続けることから、 ETL プロセスを完了するための所要時間は日によって異なります。 iSeries A でいつ ETL プロセスが終了したかを判別する問題を解決するために、 iSeries A からリモートでキューブへのデータ追加を呼び出します。すると、ETL プロセスが完了し次第、キューブへのデータ追加が始まります。これによって、 ETL プロセスの途中で問題が発生しても、キューブに再びデータが追加されることはなくなります。

キューブのデータ追加プロセスを iSeries A のリモート側で呼び出すには、次のステップに従ってください。

1. iSeries A および iSeries B は、お互いが対話できる状態でなければなりません。 iCola シナリオでは、ホストのエントリーが iSeries A のホスト表に iSeries B を示すように追加されています。
2. 分散データ管理 (DDM) ファイルが、iSeries B を参照する iSeries A 上に作成されます。 DDM はリモート・ファイル処理を制御し、iSeries サーバー上で実行されるアプリケーション・プログラムから DDM をサポートする別のサーバー上に格納されているデータ・ファイルにアクセスできるようにします。同様に、DDM を持つ他のシステムもローカルの iSeries サーバーのデータベースにあるファイルにアクセスすることができます。 DDM は、複数のサーバー間でのファイル処理の分散を容易にします。

   この DDM ファイルは、iSeries A で次のコマンドを実行すると作成されます。

- CRTDDMF FILE(ICOLA/TO_SYSTM_B) RMTFILE(ICOLA/QCLSRC) RMTLOCNAME(*iSeries_B_Host_Name* *IP)

DDM ファイルの作成後、リモート・コマンドの実行 (SBMRMTCMD) によって、iSeries B のキューブのデータ追加プログラムを呼び出すことができます。

キューブのデータ追加プロセスをバッチ・ジョブとして呼び出す

元々、ETL プロセスが iSeries A で完了した後、iSeries B にキューブを作成するためのプログラムは、次のように直接呼び出されていました。

- SBMRMTCMD CMD('CALL PGM(ICOLA/BLD_CUBE)') DDMFILE(ICOLA/TO_SYSTM_B)

しかし、このプログラムを直接的に呼び出すということは、 iSeries A が、iSeries B のプログラムがキューブのデータ追加が終了するまで待機しなければならないことを意味しています。これは最適な解決策とはいえないため、このプログラム呼び出しを次のようにバッチ・ジョブとして実行します。

- SBMRMTCMD CMD('SMBJOB CMD(CALL PGM(ICOLA/BLD_CUBE))') DDMFILE(ICOLA/TO_SYSTM_B)

キューブのデータ追加プログラムをバッチ・ジョブとして実行すると、 iSeries B がキューブにデータを追加している間でも、iSeries A は自身のプログラムの処理を完了することができます。

iSeries B から iSeries A のデータマートに接続できるようにする

前述したように、iSeries A のデータマートにあるデータを使用して、iSeries B にあるキューブにデータが追加されます。 iSeries B で iSeries A のデータを抽出できるようにするために、 iSeries A を指し示すリレーショナル・データベース (RDB) のエントリーを iSeries B に追加します。リレーショナル・データベースのエントリーを追加するには、リレーショナル・データベース・ディレクトリー・エントリーの処理コマンド (WRKRDBDIRE) を使用します。

また、iSeries B から iSeries A のデータマートにアクセスできるようにするために、 iSeries A のユーザーは、データの要求を行った iSeries B のユーザーと同一のプロファイル名およびパスワードが必要になります (iCola シナリオでは、このユーザーは DB2(R) OLAP 管理者です)。 iSeries A のマッチング・プロファイルにもアクセスされるデータベース表に対する権限が必要です。データを要求している iSeries (iSeries B) のプロファイルに一致するプロファイルが iSeries A にない場合は、権限エラーが発生します。

SQL キーワードを表名として使用する

当初、データ・ロード・ルールは、次のように SQL 句を使用して作成されました。

- SELECT ..., **TIME**.MONTH, ...)
  FROM ICOLADM.FACTTABLE, ICOLADM.CUSTOMER, ICOLADM.**TIME**, ICOLADM.PRODUCT, ICOLADM.MARKET
  WHERE... AND ICOLADM.FACTTABLE.TIMEID = **TIME**.TIMEID AND ...

このステートメントを使用すると、TIME 表をポインティングするときにエラーが発生します。 V5R3 では、TIME は SQL の予約語であり、特殊な意味があります。この表をリネームせずに QUERY を実行するには、次に示すように、この表の別名を設定します。

- SELECT ..., **MYTIME**.MONTH, ...)
  FROM ICOLADM.FACTTABLE, ICOLADM.CUSTOMER, ICOLADM.**TIME AS MYTIME**, ICOLADM.PRODUCT, ICOLADM.MARKET
  WHERE... AND ICOLADM.FACTTABLE.TIMEID = **MYTIME**.TIMEID AND ...

あまり知られていない SQL キーワードを使用する

ロード・ルールを作成するときに役立つあまり知られていない SQL キーワードがいくつかあります。

- RTRIM: このキーワードは、データ・フィールドの右側の空白を切り取ります。例えば、RTRIM('Rochester ') = 'Rochester' のようなストリングになります。 RTRIM キーワードを使用すると、ストリングを連結したときに余分な空白を確実に削除できるので便利です。

  (同様に、左側の空白を切り取るには LTRIM 機能を使用します)。
- CONCAT: CONCAT キーワードは、2 つの個別のストリングを 1 つの連続したストリングに結合します。
  - 例えば、CONCAT('String 1', 'String 2') = 'String 1String 2' のようになります。

  この 2 つのストリングの間にスペースを追加するには、CONCAT('String 1, CONCAT(' ', 'String 2')) のようにします。このキーワードは、2 つのエレメントを 1 つに結合して固有の生成フィールドを作成するのに便利です。

第 5 章 Analyzer

IBM(R) DB2 OLAP Server(TM) Analyzer 8.1 は、DB2 OLAP Server アプリケーションのすべての機能を統合し、活用することを目的に設計された製品です。この製品は、Web ベースの直感的なインターフェースによって、DB2 OLAP Server で生成された OLAP キューブ内に含まれるデータを表示することができます。また、さらに重要なこととして、ユーザーが基本となる OLAP キューブ内のデータに基づいて分析レポートを作成することができます。このレポートを使用して、ビジネス上の重要な意思決定を行い、過去または現在のビジネス業務や傾向をより良いものにすることができます。

Analyzer を使用することにより、ユーザーは、複雑なマルチディメンション・データまたはリレーショナル・データを、製品の販売傾向や収益性、および他の重要なビジネス・パフォーマンス・メトリックスを反映できる優れたレポートに変換させることができます。この分析内容は、営業、マーケティング、その他の社内の各部門で共有したり管理したりできます。

Analyzer では、スプレッドシート、線グラフ、円グラフ、棒グラフ、予定表、空間グラフなどさまざまなタイプのレポートが作成できます。レポートの作成方法は、グラフィカル・コントロールのドラッグ・アンド・ドロップと同様に単純ですが、初めに一通りの全体像や設計を考えておく必要があります。 レポートの作成後には、新たに生成された任意のビューからデータをドリルアップまたはドリルダウンすることでさらに新しくレポートを作成する機能があります。その後、各レポートを Analyzer サーバーに保存して、今後の使用に備えてこのレポートの定義を保管することができます。 OLAP キューブを表示してレポートの定義に任意の時点で再びアクセスすると、レポートのデータはキューブ内の現在のデータに基づいて更新されます。オプションのフィルタリング機能を使用すると、レポートで実施した分析にさらに制約を加えることができます。

Analyzer には、Java(TM) ベースの Web クライアントと HTML ベースの Web クライアントの 2 種類のインターフェースがあります。 Java Web クライアントを使用する場合、ユーザーは、精巧なグラフィカル・インターフェースによって非常に複雑な分析レポートを作成したり保存したりできます。アプリケーションのこの部分は、レポート処理を実行する場合に、より高度な機能を使う必要があるレポートの設計者や熟練ユーザー向けの機能です。一方、HTML Web クライアントは、 業務分析レポートを作成するのではなく、これを利用するユーザーを対象にしています。経営陣やプロジェクト・マネージャーがこのインターフェースの典型的なユーザーです。開発者も、Analyzer API ツールキットによって IBM DB2 OLAP Server Analyzer 8.1 の機能を拡張することができます。このツールキットを使用すると、カスタム Web ベースの業務分析アプリケーションを作成できます。

DB2 OLAP Server Analyzer はデータ・マイニング・ツールではありません。どちらかというと、レポートの設計者からの「QUERY」や質問に基づいた業務レポートの作成に焦点を置いています。通常、レポートの設計者は、マネージャーや経営陣の要求を満たすようなレポートを作成します。データ・マイニングは他のツールを使って行うことができます。 Analyzer は、OLAP データからレポートを作成したり、このレポートから読み取れる傾向に基づいてビジネス・アクションを決定することができます。

IBM DB2 OLAP Server Analyzer 8.1 それ自体は、Hyperion Analyzer 6.1 を基にした製品であり、デフォルトでは J2EE に準拠した IBM WebSphere(R) Application Server を使用します。 IBM DB2(R) パーソナル・エディションは、Analyzer 固有の機能で必要なデフォルトのリレーショナル・リポジトリーとして使用します。 iCola シナリオの目的から、Analyzer アプリケーションに含まれる 2 つの Analyzer Administration ツールは使用しません。このアプリケーションには、1 サーバー、1 ユーザー/グループ、および 1 データベースの接続だけが必要になります。

アプリケーションの概要

iCola のシナリオでは、DB2(R) OLAP データ・キューブ内に含まれるデータの直接的な表示や分析は、主に DB2 OLAP Analyzer 製品を通じて行われます。このシナリオで使用する Analyzer サーバー・アプリケーションは、主要なフロントエンドとして DB2 OLAP データ・キューブを表示したり、キューブの再構築後に分析データが変更されているかどうか検証したりします。ビジネスの世界では、Analyzer がキー・コンポーネントとなってレポートや図表を作成し、このレポートや図表が DB2 OLAP Server(TM) をビジネス・インテリジェンス・ソリューションとして使用する企業の将来のビジネス戦略の指標となります。

Analyzer Server for Windows(R) のインストール後にアプリケーションを開始すると、Analyzer Launch Page (図13) が表示されます。このページは基本的にアプリケーションで提供される使用可能なすべてのツールへのリンクが含まれているページです。

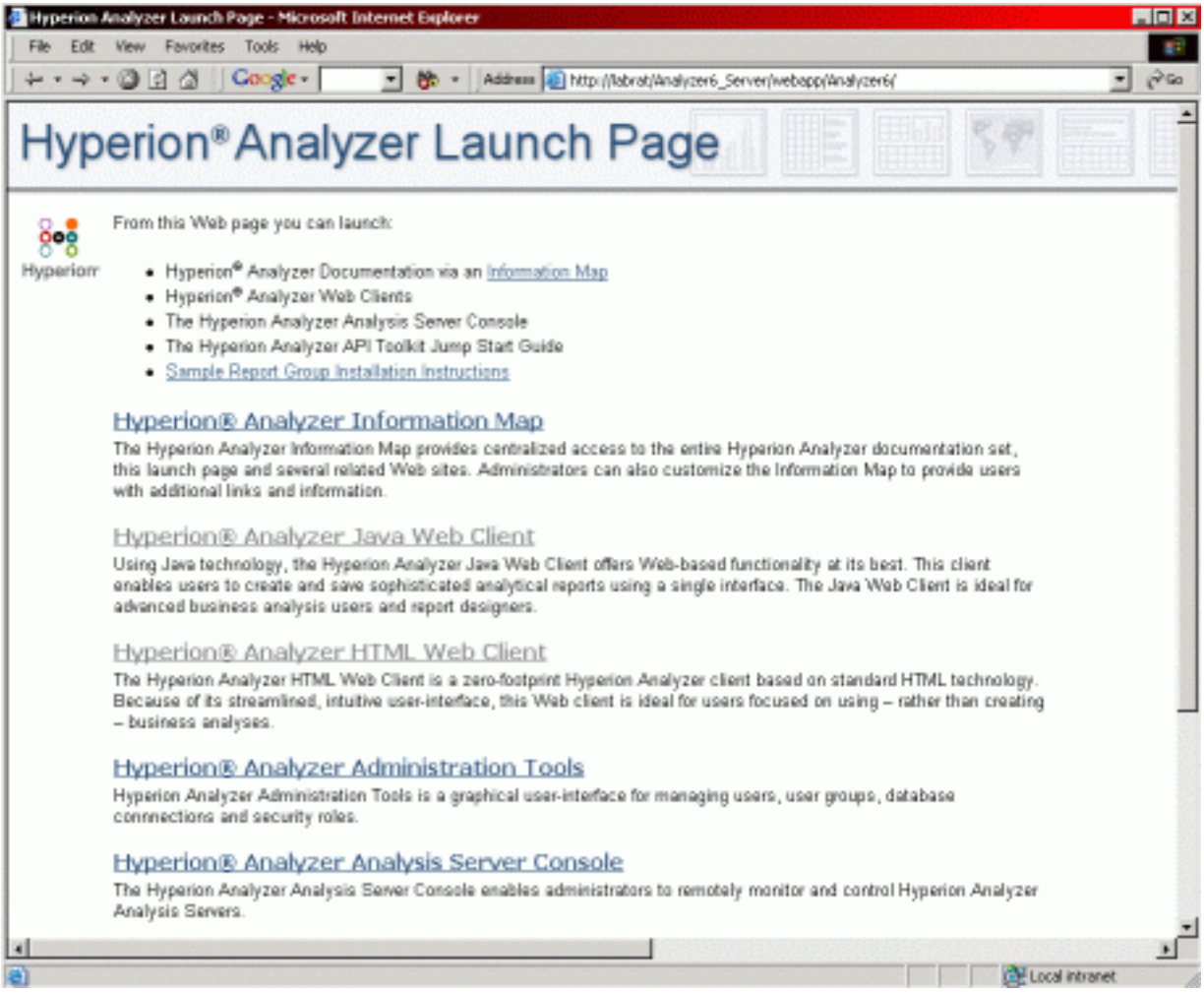

図 13: Analyzer Launch Page

iCola のシナリオでは、Java(TM) ベースおよび HTML ベースの両方の Web クライアントが使用されていますが、このシナリオではそれぞれに独自の用途があります。 Java ベースの Web クライアント (図 14) は、実施前に再検討した業務上の質問セットをベースにレポートを作成する目的で、設計フェーズで使用されます。

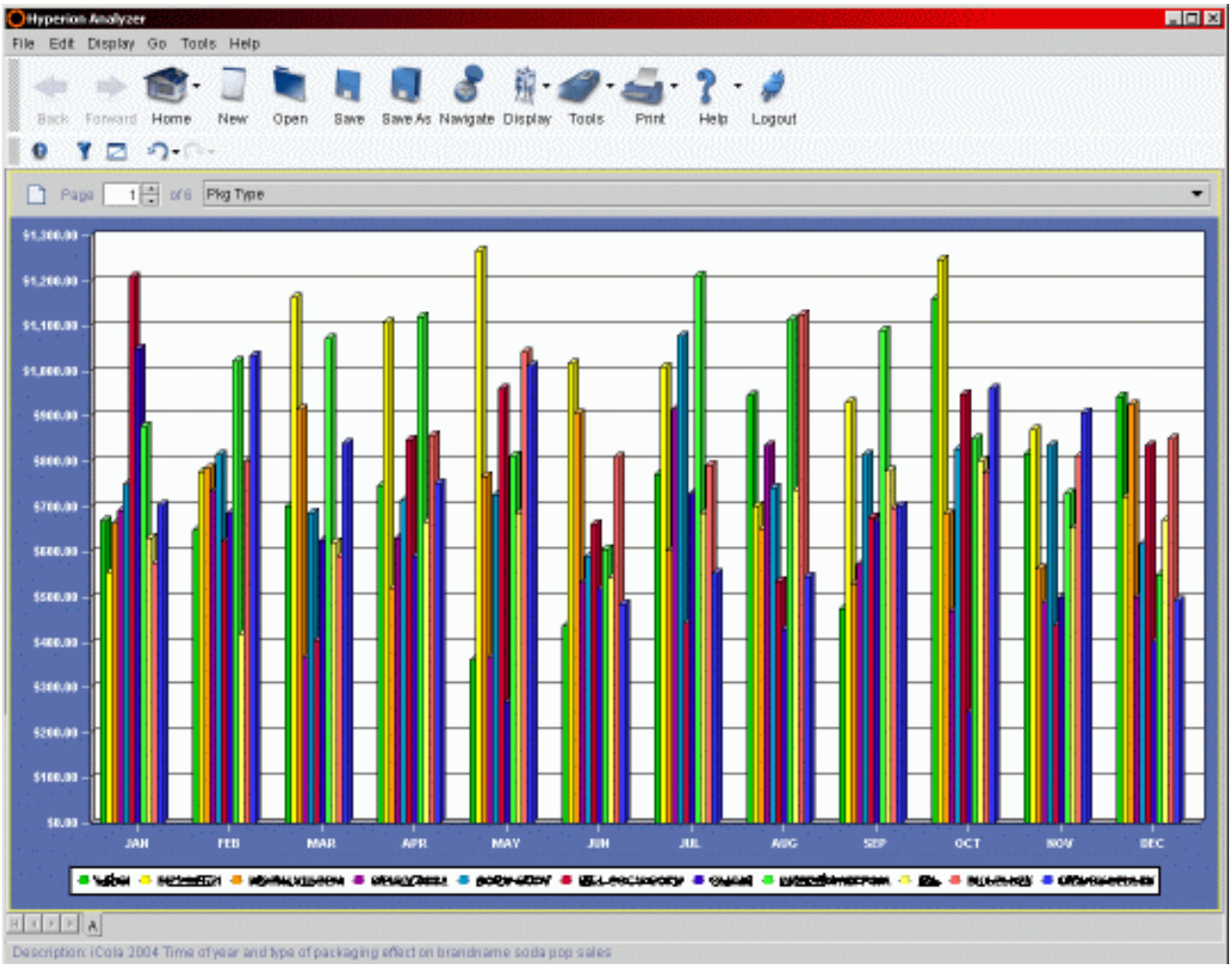

図 14: Java ベースの Web クライアント

iCola のシナリオ上の目的に応じて図表スタイルの形式を使用していますが、スプレッドシート形式のレポートや図表とスプレッドシートを組み合わせたレポートなど、そのほかにもさまざまな機能があります。iCola のシナリオに Analyzer 製品を組み込む目的は、Analyzer 製品自体を広範囲にテストすることではなく、 OLAP キューブのエクササイザーとして使用したり、Analyzer 製品の使用方法を概念的に実証したりするためです。

iCola のシナリオに含まれる Analyzer アプリケーションは、テスト環境で自動ワークロード・アプリケーションによって直接実行される 2 つのアイテムのうちの 1 つです。 HTML ベースの Web クライアント (図 15) により、ワークロード (Web インターフェース上でユーザー処置をシミュレートする自動スクリプト) に優れたインターフェースが提供され、論理的に意味のあるものになります。レポートの定義や設計に持続的に変更を加えるのは現実的とは言えないためです。その代わり、複数のデータ報告者が複数の時点において同一のビューにアクセスすることができます。またそれらの報告者は、それぞれのマネージャーや経営陣にレポートや要約を提出できます。

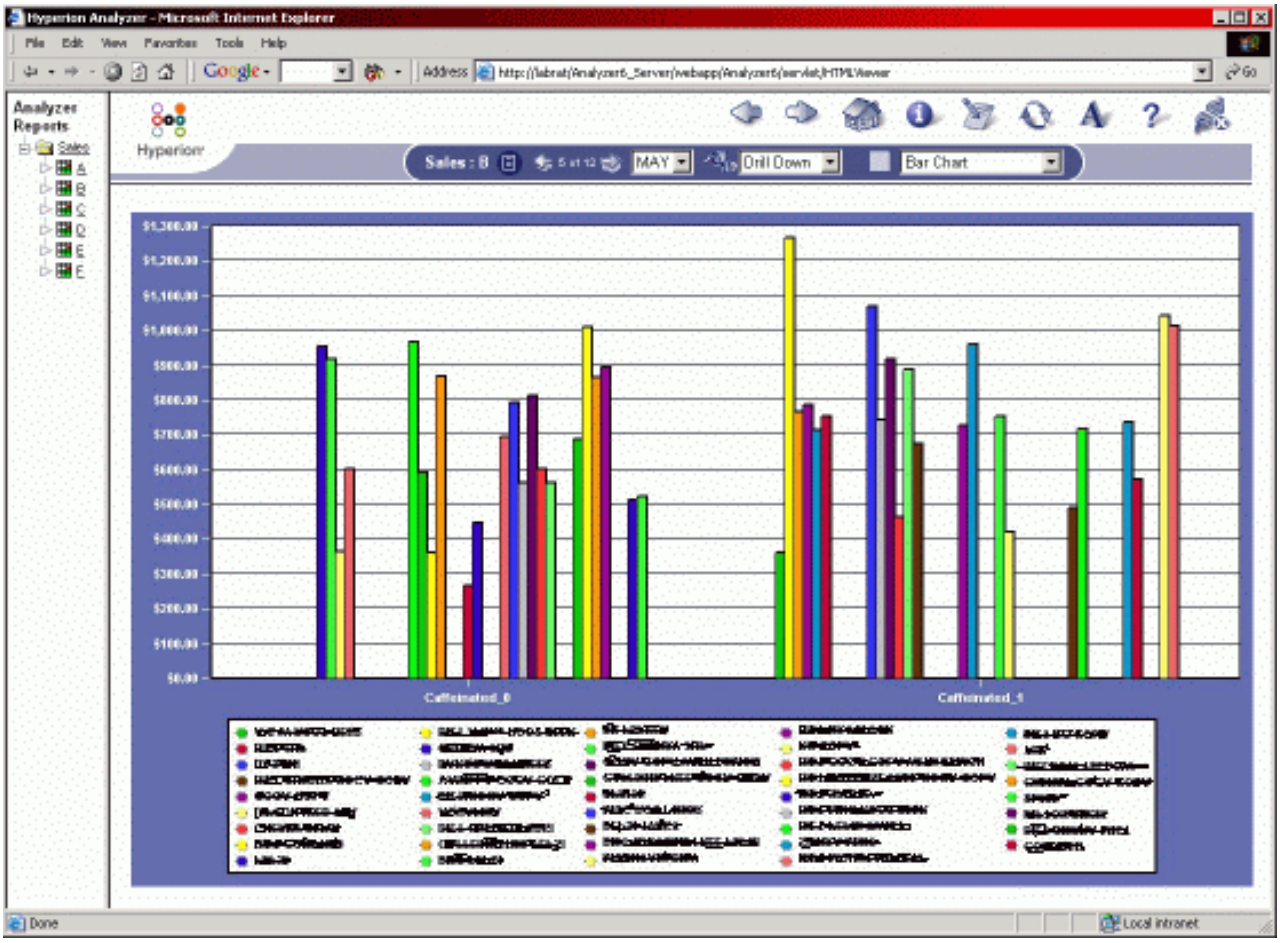

図 15: HTML ベースの Web クライアント

アプリケーション設計のポイント

シナリオのこの部分ではそれほど多くの初期設計はありません。この製品はその大部分が「すぐに使用可能」なため、構成を追加する必要がそれほど多くないからです。 IBM(R) DB2 OLAP Server(TM) Analyzer に提供されている Java(TM) API を使用すると多様なカスタマイズが可能ですが、既存の基本製品を使用するだけでこのシナリオの基本要件は満たされるため、カスタマイズは必要ありません。 Analyzer サーバー・アプリケーションのインストール後に、アプリケーションが OLAP サーバーと正しく通信できるようにするために多少の構成ステップを実行する必要があります。このステップについては、後ほど Analyzer 用のアプリケーションのセットアップのセクションで詳しく説明します。

Analyzer 自身は、このシナリオにおける特定の設計ステップを必要としないため、代わりに、設計プロセスは日常の iCola のオペレーションの一部としてのアプリケーションの使用法に焦点を置いています。Analyzer の主な目的は重要なビジネス・データのためのレポート作成メカニズムを提供することであるため、使用可能な基準に基づいてさまざまなビジネス上の疑問を提起する必要があります。この Analyzer を使用して過去および現在のビジネスの傾向を焦点に入れながら「QUERY」を編成することで、今後のビジネス・ステージにおいてより優れた戦略を展開したり活動を開始したりできます。

例えば、このような QUERY を使用して特定銘柄のソーダの注文量がどの特定の月で落ち込む傾向にあるかを調べ、その情報を基にこれらの月間にバイヤーに対して価格設定のインセンティブを設けたりすることができます。データ共用の例として、大量販売月間には、このデータを基にポテト・チップのメーカー/販

売業者などのパートナーと両社で販売を促進するためのクーポン・キャンペーンを開始することができます。 iCola 社が収集したデータは、レポート形式に集約して、実際の飲料メーカーに販売することもできます。それによってメーカー各社も、自社の製品についてデータに基づいた意思決定を行うことができるようになります。

現在、Analyzer 製品を使用して以下の質問が OLAP キューブ・データに対して実行されています。

- 人気銘柄のソーダ製品の販売高に季節とパッケージのタイプがどのように影響するか。
- 昨年の各月における販売高が最高および最低のソーダの銘柄の種類は何か。カフェイン入りとカフェインなしの種類別に提示すること。
- 当年における地域別の人気ストア、銘柄別のビールの販売高が最高および最低のストアはどこか。
- 当年において、コーラの銘柄 A と銘柄 B の二者で販売高を比較するとどのような結果になるか。
- 当年において、ビールの銘柄 C と銘柄 D の二者で販売高を比較するとどのような結果になるか。
- 今年一年間、四半期、月別に見た場合、全体的な収益高はどうなるか。

現在の設計では、選択する年に基づいてビューを動的に変更することもできるため、このような任意の QUERY を任意の過去の年のキューブ・データに対して実施してより詳しい傾向分析を行うことができます。データベース内のデータを分析するには、これ以外にも数々の有効な質問事項があり、今後、シナリオが成長、拡大していくにつれ、さらに多くの QUERY が追加されます。

アプリケーションのセットアップ

IBM(R) DB2(R) OLAP Analyzer 8.1 には、Windows(R) 版と AIX(R) 版の両方のインストール資料が用意されていますが、この資料を実際に使用してみると、その全体的な評価は、理解しにくいというものでした。インストールのプロセス中に生じた疑問のうち、資料または Web を検索しても解答が得られないものがあります。

初めに、AIX(R) を実行する IBM pSeries(R) サーバーへのアプリケーションのセットアップを行いました。しかし、サポート部門に連絡した後になって、 AIX 上の DB2 OLAP Server(TM) Analyzer から iSeries(TM) V5R2 上の DB2 OLAP Server へはアクセスできないことが判明しました。このため、次善策として、IBM xSeries(R) サーバーで動作する Windows 版の DB2 OLAP Server Analyzer を使用することにしました。

IBM xSeries 上の Windows XP Professional プラットフォームに IBM DB2 OLAP Server Analyzer 8.1 をセットアップするために必要なインストール・プロセスのハイレベルな要約は次のようになります。

1. IBM DB2 OLAP Server Analyzer 8.1 for Windows の CD-ROM を入手する。
2. 使用している xSeries サーバーが Windows オペレーティング・システムと互換性があることを確認する。 DB2 OLAP Server Analyzer の推奨ハードウェアをチェックして、xSeries サーバー上で適切に動作することを確認します。
3. xSeries に電源を入れ (まだ電源が投入されていない場合)、管理者アカウントで Windows にログインする。
4. Windows の「コントロール パネル」を開き、Java(TM) Plugin 1.3.0_02 を探す。
5. Java Plugin 1.3.0_02 がインストールされていない場合は、Analyzer の CD-ROM からインストーラーを探してこの時点でインストールする(Java_Plugin_1_3_0_02 ディレクトリーの .exe ファイルを使用)。 Java Plugin のインストールが完了するまでは次のステップに進まないでください。
6. Windows の「コントロール パネル」を再び開いて、Java Plugin 1.3.0_02 がインストールされていることを確認する。

7. Analyzer CD-ROM を使用して、システムに DB2 パーソナル・エディション・バージョン 7.2 をインストールする。このファイルは CD-ROM の DB2 ディレクトリーにあります。インストールでは、「管理者クライアント (Admin Client)」チェック・ボックスを選択し、すべてのユーザー用 ID として db2admin、パスワードには「password」を指定する以外は、すべてデフォルト・オプションを選択してください。ユーザー ID およびパスワードは、後で必要に応じて変更することができます。DB2 のインストールが完了するまでは次のステップに進まないでください。
8. xSeries をリブートし、先ほど使用した管理者用アカウントで再び Windows にログインする。
9. 現在実行中のすべての DB2 サービスを停止する。これを行うには、Windows の「コントロール パネル」を開いて、「**管理ツール**」、「**サービス**」の順に選択し、現在実行中の DB2 サービスを停止します。ウィンドウは開いたままにします。
10. DOS のプロンプト画面を開き、SQLLIB¥java12 ディレクトリー (通常は C:¥SQLLIB¥java12) で **usejdbc2.bat** ファイルを実行します。このファイルの実行が終了したら、DOS のプロンプト画面を閉じます。
11. すでに開いている「**サービス** 」ウィンドウに戻る。ステップ 9 以前に実行していたすべての DB2 サービスを開始します。
12. DB2 コントロール・センター・アプリケーションから「**DB の作成 (Create DB)**」ウィザードを使用して、 ANALYZ60 データベースを作成する。「スタート」メニューから「IBM DB2」プログラム・フォルダーに移動し、「**コントロール・センター**」を選択します。 DB2 コントロール・センター・アプリケーションで、システム名の下に移動てから、「**インスタンス**」、「**DB2**」、および「**データベース**」の順に移動します。「**データベース**」フォルダーを右クリックし、「作成」 -> 「データベース使用ウィザード (Database Using Wizard)」を選択します。データベース名および別名として ANALYZ60 と入力してから、「**次へ**」ボタンをクリックします。ウィザードの残りのステップでは、デフォルト・オプションだけを選択します (要約ページが表示されるまでは「**次へ**」ボタンをクリックし続け、最後に「**完了**」をクリック)。データベースの作成が完了するまで待機します。ステップが終了したら、DB2 コントロール・センター・アプリケーションを終了します。
13. Analyzer の CD-ROM (WebSphere35/NT ディレクトリーの CD にあります) から WebSphere(R) Application Server 3.5 をインストールする。データベース画面での選択を除いて、すべてデフォルト・オプションを選択します。データベースを選択するときに、WebSphere のインストール・プロセスで使用するデータベース・タイプとして、適切に動作するように「**InstantDB**」を選択します。要約画面が表示されるまでデフォルトを選択し続けてインストール・プロセスを完了します。インストールが完了するまでは、次のステップに進まないでください。
14. xSeries をリブートし、先ほど使用した管理者用アカウントで再び Windows にログインする。
15. WebSphere Application Server 3.5 フィックスパック #5 をインストールする。このフィックスパックは、WebSphere のアプリケーション・レベルを 3.5 から 3.5.5 に変更しますが、これは Analyzer を正しくインストールするために必要です。フィックスパック・ファイルは、Analyzer の CD-ROM のルート・ディレクトリー (X:¥was35_std_ptf_5.zip) にあります。このファイルを PC のローカル・ディレクトリーで unzip して、解凍されたセットアップ・ファイルを実行します。フィックスパックのインストーラーのプロンプトに従ってパラメーターを入力します。 IBM HTTP Server (IHS) へのアップデートを求められたら、このアプリケーションをアップデートするためのローカル・システム上の IHS 用のディレクトリー名を入力します。
16. xSeries をリブートし、先ほど使用した管理者用アカウントで再び Windows にログインする。
17. DB2 OLAP Server Analyzer CD-ROM から IBM DB2 OLAP Server Analyzer 製品をインストールする。表示されるすべてのデフォルトを選択します (xSeries 上のWebSphere および DB2 など、すでにインストールしたプログラムに適合している場合)。デフォルトと異なるのは、RDBMS のユーザー名に、デフォルトの **Analyzer** ではなく、**db2admin** を選択することだけです。

18. 最後に、URL (http://localhost/Analyzer6_Server/webapp/Analyzer6/index.html (**localhost** は、リモートからこの製品を使用するシステムのホスト名に置き換える)) に移動して、IBM DB2 OLAP Server Analyzer の使用を開始します。

主要な発見事項

IBM(R) DB2 OLAP Server(TM) Analyzer 8.1 の使用プロセス全体を通じて最も顕著なことは、アプリケーションのインストールが困難だという点です。この問題の主な原因は、Analyzer に付属するインストール手順の資料の不備にあります。本章の『アプリケーションのセットアップ』で説明する手順は、インストールのプロセス全体で非常に役立ちます。インストールが終了すると、この手順が非常に有効で、 iCola シナリオのビジネス・インテリジェンス・ニーズに適合するように高度に構造化されていることが証明されます。

Analyzer アプリケーションへのユーザー・アクセスをシミュレートする自動ワークロード・テストは iCola シナリオの残りの部分と並列的に実行されました。これは、バックエンドで OLAP キューブを実行するためであり、さらには、Analyzer ウェブ・アプリケーションを実行するためです。自動化すると、単一ユーザー・ワークロード・テストがアプリケーションに対して実行されますが、これは 1 人の実際のユーザーによって手動でアプリケーションにアクセスした場合よりパフォーマンスが低下するように見受けられました。これは、ワークロード・テストの実行中に他の要因が影響した結果である可能性があります。

Windows(R) XP Professional は、xSeries(R) ボックスにインストール済みのオペレーティング・システムですが、テストは Windows 2000 で実施されたもので、当時はパフォーマンスに問題はないように見受けられました。パフォーマンス上のもう一つの注意点として、Analyzer アプリケーションで使用した Java(TM) プロセスはリアルタイム優先順位に従って実行するように調整されており、この変更は、Windows タスク・マネージャー・ユーティリティーで実行できる点が挙げられます。この変更は、Analyzer アプリケーションのパフォーマンスを向上させるために行われ、これがシステムで実行する必要のある実際の唯一のプロセスであったため、最大の優先順位を設定しても xSeries の他のアプリケーションには影響を与えません。xSeries を多機能サーバーとして実行し、このサーバーが Analyzer の専用サーバーでない場合、実行時の優先順位を変更することはパフォーマンス上の理由から最良でない場合があります。

Analyzer を使用することで得られる重要な発見の一つは、ビジネス・インテリジェンス・データそのものを、収益性の高いエンティティーとして販売することも可能であるということです。 Analyzer を使用して収集したデータおよびレポートには、iCola 社にとって単なるビジネス・インテリジェンス・データ以上の新たな利用法があります。例えば、特定の飲料メーカーをターゲットにしたデータは、その実際の飲料メーカーのビジネス利用を目的として製品自体として販売できます。また、各ストアでは、自社で扱う製品の販売ビジネスにおけるすべての傾向を追跡できない場合もあります。このような企業に対しては、キューブを「スライス」して、知的所有権として販売することができます。この企業では自社の分析結果に基づいて競合他社との販売競争に打ち勝つためにこの知的所有権を使用することができます。このように、Analyzer は、iCola 飲料販売会社の DB2 OLAP Server が管理するデータを使用する他の企業の業務上の意思決定においても非常に重要な役割を果たします。

全体的に見ると、IBM DB2 OLAP Server Analyzer 8.1 は、予想以上にインストールが難しいものであったとしても、 iCola シナリオの目的にとって貴重な資産と言えます。製品自体は、始めは難しそうに見えるかもしれませんが、適切に理解すれば操作は難しくありません。このアプリケーションの動作に慣れてくれば、この製品を使用して、ビジネス上の意思決定者にビジネス・インテリジェンス・データを提供する方法を改善できる可能性は高いと思われます。 DB2 OLAP Server のすべてのユーザーにとって、DB2 OLAP Server Analyzer の Web インターフェースは、 DB2(R) OLAP Server を通じて収集するビジネス・インテリジェンス・データの分析を成功させる必須アイテムです。

第 6 章 将来のアイデア

戦略的な企業会計およびマーキングに関する意思決定者が、ビジネス・インテリジェンスから導き出す情報の種類に精通していくにつれ、その要求も増えていきます。ビジネス・インテリジェンス・プロジェクトの将来のフェーズでは、データを分析するためのより多くのマルチディメンション・キューブが作成されていきます。さらに、DB2 OLAP Server(TM) のデータ・マイニング機能を有効活用して、自社のビジネスにおける戦略的プランニングで意思決定者の役に立つパターンを発見する計画もあります。

高度な DB2 OLAP Server の機能を活用したり、より多くのデータを必要とする戦略的意思決定者からの要求に取り組む以外にも、 iCola のシナリオには、最新の iSeries(TM) データベース機能である、マテリアライズ照会表などを活用する計画があります。

企業が成長してデータ・ソースが多種多様になってくると、データマートを拡張してデータウェアハウスを構築する必要性が明らかになってくる可能性があります。データウェアハウスの必要性が生じると、より複雑なデータ抽出、トランスフォーメーション、およびプロセスのロードなどを支援するために使用可能なツールの分析が必要になります。さらに、単一のデータウェアハウスから、複数のサブジェクト指向のデータマートにデータを追加する場合もあります。

データからさらに多くの価値を引き出すツールについても、検討されていくことになります。データ・マイニングや意思決定を行うためのより高度なツールや技法も開発が進むと思われます。

第 7 章 付録 A

データマート ICOLADM の作成で使用する SQL ステートメント

ICOLADM ライブラリーの作成で使用する SQL ステートメントは次のとおりです。

```
CREATE SCHEMA ICOLADM ;
```

ICOLADM のファクト表およびディメンション表の作成で使用する SQL ステートメントは次のとおりです。

```
CREATE TABLE ICOLADM.CUSTOMER (
BRANCHID INTEGER NOT NULL ,
BRANCHNAME CHAR(25) CCSID 37 NOT NULL ,
STOREID INTEGER NOT NULL ,
STORENAME CHAR(25) CCSID 37 NOT NULL ,
CONSTRAINT ICOLADM.QSYS_CUSTOMER_00001 PRIMARY KEY( BRANCHID ) ) ;

CREATE TABLE ICOLADM.FACTTABLE (
ORDERID INTEGER NOT NULL ,
UPC INTEGER NOT NULL ,
MARKETID INTEGER NOT NULL ,
TIMEID INTEGER NOT NULL ,
QUANTITYSOLD FOR COLUMN QUANT00001 INTEGER NOT NULL ,
BRANCHID INTEGER NOT NULL ,
SELLINGPRICE FOR COLUMN SELLI00001 DECIMAL(9, 2) NOT NULL ,
TOTALPRICE DECIMAL(9, 2) NOT NULL ,
COSTTOBUY DECIMAL(9, 2) NOT NULL )
CONSTRAINT ICOLADM.QSYS_FACTTABLE_ORDERID_00001 PRIMARY KEY( ORDERID) ;

CREATE TABLE ICOLADM.MARKET (
MARKETID INTEGER GENERATED ALWAYS AS IDENTITY (
START WITH 1 INCREMENT BY 1
NO MINVALUE NO MAXVALUE
NO CYCLE NO ORDER
CACHE 20 ) ,
REGION CHAR(20) CCSID 37 NOT NULL ,
STATE CHAR(2) CCSID 37 NOT NULL ,
CITY CHAR(20) CCSID 37 NOT NULL ,
COUNTRY CHAR(2) CCSID 37 NOT NULL ,
POPULATION INTEGER NOT NULL ,
CONSTRAINT ICOLADM.QSYS_MARKET_00001 PRIMARY KEY( MARKETID ) ) ;

CREATE TABLE ICOLADM.PRODUCT (
UPC INTEGER NOT NULL ,
SIZE INTEGER NOT NULL ,
CAFFEINATED FOR COLUMN CAFFE00001 DECIMAL(1, 0) NOT NULL ,
```

```
ALCOHOLIC DECIMAL(1, 0) NOT NULL ,
INTRODATE DATE NOT NULL ,
DISCONTINUEDATE FOR COLUMN DISCO00001 DATE DEFAULT NULL ,
PACKAGETYPE FOR COLUMN PACKA00001 CHAR(13) CCSID 37 NOT NULL DEFAULT '' ,
NAME CHAR(80) CCSID 37 NOT NULL DEFAULT 'No default' ,
BRAND CHAR(40) CCSID 37 NOT NULL DEFAULT 'No default' ,
CONSTRAINT ICOLADM.Q_ICOLADM_PRODUCT_UPC_00002 PRIMARY KEY( UPC ) ) ;

CREATE TABLE ICOLADM.TIME (
TIMEID INTEGER NOT NULL ,
"DAY" SMALLINT NOT NULL ,
"MONTH" CHAR(3) CCSID 37 DEFAULT NULL ,
"YEAR" SMALLINT NOT NULL ,
QUARTER CHAR(4) CCSID 37 DEFAULT NULL ,
SEASON CHAR(6) CCSID 37 DEFAULT NULL ,
DAYOFWEEK CHAR(9) CCSID 37 DEFAULT NULL ,
CONSTRAINT ICOLADM.QSYS_TIME_00001 PRIMARY KEY( TIMEID ) ) ;
```

ICOLADM の補助表 ETLPROCESS (ロギングに使用) および REGIONS (MARKET 表へのデータの追加を支援する) を作成する SQL ステートメントは次のとおりです。

```
CREATE TABLE ICOLADM.ETLPROCESS (
START TIMESTAMP NOT NULL ,
FINISH TIMESTAMP DEFAULT NULL ) ;

CREATE TABLE ICOLADM.REGIONS (
STATE CHAR(2) CCSID 37 NOT NULL ,
REGION CHAR(15) CCSID 37 NOT NULL ,
COUNTRY CHAR(2) CCSID 37 NOT NULL DEFAULT 'US' ,
CONSTRAINT ICOLADM.QSYS_REGIOINS_00001 PRIMARY KEY( STATE ) ) ;
```

第 8 章 付録 B

ETL プロセスの SQL ストアード・プロシージャー

この付録には、iCola が SQL の各ストアード・プロシージャーの作成で使用するコードを記載します。

次のように、ETL プロシージャーは他のすべての SQL プロシージャーを呼び出します。

```
CREATE PROCEDURE ICOLA.ETL ( )
LANGUAGE SQL
SPECIFIC ICOLA.ETL
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
CALL ICOLA . STARTETL ;
CALL ICOLA . CUSTOMER ;
CALL ICOLA . MARKET ;
CALL ICOLA . PRODUCT ;
CALL ICOLA . FACTTABLE ;
CALL ICOLA . ENDETL ;
END;
```

STARTETL プロシージャーおよび ENDETL プロシージャーは、次のように CURRENT TIMESTAMP を記録することにより時刻をロギングします。

```
CREATE PROCEDURE ICOLA.STARTETL ( )
LANGUAGE SQL
SPECIFIC ICOLA.STARTETL
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . ETLPROCESS ( START ) VALUES ( CURRENT TIMESTAMP ) ;
END ;
```

```
CREATE PROCEDURE ICOLA.ENDETL ( )
LANGUAGE SQL
SPECIFIC ICOLA.ENDETL
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
UPDATE ICOLADM . ETLPROCESS SET FINISH = ( CURRENT TIMESTAMP )
WHERE START = ( SELECT MAX ( START ) FROM ICOLADM . ETLPROCESS ) ;
END ;
```

次のように、プロシージャー CUSTOMER、MARKET、および PRODUCT を実行すると、それぞれが対応するデータマート表に情報が挿入されます。以下のプロシージャー (EXCEPT キーワードを使用) は、OS/400 V5R3 でのみ使用できます。

```
CREATE PROCEDURE ICOLA.CUSTOMER ( )
LANGUAGE SQL
SPECIFIC ICOLA.CUSTOMER
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . CUSTOMER ( BRANCHID , BRANCHNAME , STOREID , STORENAME )
( ( SELECT ICOLA . CUSTOMERBRANCH . BRANCHID , ICOLA . CUSTOMERBRANCH .
BRANCHNAME , ICOLA . CUSTOMERBRANCH . STOREID , ICOLA . STORE . BUSINESSNAME
FROM ICOLA . CUSTOMERBRANCH , ICOLA . STORE
WHERE ICOLA . CUSTOMERBRANCH . STOREID = ICOLA . STORE . STOREID )
EXCEPT
( SELECT BRANCHID , BRANCHNAME , STOREID , STORENAME FROM ICOLADM . CUSTOMER )
) ;
END ;

CREATE PROCEDURE ICOLA.MARKET ( )
LANGUAGE SQL
SPECIFIC ICOLA.MARKET
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . MARKET ( REGION , STATE , CITY , COUNTRY , POPULATION )
( ( SELECT ICOLADM . REGIONS . REGION , ICOLA . CUSTOMERBRANCH . STATE, ICOLA .
CUSTOMERBRANCH . CITY , ICOLADM . REGIONS . COUNTRY
FROM ICOLA . CUSTOMERBRANCH INNER JOIN ICOLADM . REGIONS ON ICOLA .
CUSTOMERBRANCH . STATE = ICOLADM . REGIONS . STATE )
EXCEPT
( SELECT REGION , STATE , CITY , COUNTRY , 0 FROM ICOLADM . MARKET ) ) ;
END ;

CREATE PROCEDURE ICOLA.PRODUCT ( )
LANGUAGE SQL
SPECIFIC ICOLA.PRODUCT
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . PRODUCT ( UPC , NAME , BRAND , SIZE , CAFFEINATED ,
ALCOHOLIC , PACKAGETYPE , INTRODATE )
( ( SELECT UPC , NAME , BRAND , SIZE , CAFFEINATED , ALCOHOLIC , PACKAGETYPE,
INTRODATE
FROM ICOLA . PRODUCT )
```

```
EXCEPT
( SELECT UPC , NAME , BRAND , SIZE , CAFFEINATED , ALCOHOLIC , PACKAGETYPE,
INTRODATE FROM ICOLADM . PRODUCT ) ) ;
UPDATE ICOLADM . PRODUCT SET DISCONTINUEDATE =
( SELECT DISCONTINUEDATE FROM ICOLA . PRODUCT
WHERE ICOLADM . PRODUCT . UPC = ICOLA . PRODUCT . UPC
AND ICOLA . PRODUCT . DISCONTINUEDATE IS NOT NULL ) ;
END ;
```

以下の CUSTOMER、MARKET、および PRODUCT の各プロシージャーは、OS/400 V5R2 および V5R3 の両方で使用できます。

```
CREATE PROCEDURE ICOLA.CUSTOMER ( )
LANGUAGE SQL
SPECIFIC ICOLA.CUSTOMER
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . CUSTOMER (BRANCHID , BRANCHNAME , STOREID , STORENAME )
(SELECT ICOLA.CUSTOMERBRANCH.BRANCHID, ICOLA.CUSTOMERBRANCH.BRANCHNAME, ICOLA
.CUSTOMERBRANCH.STOREID, ICOLA.STORE. BUSINESSNAME
FROM ICOLA.CUSTOMERBRANCH, ICOLA.STORE
WHERE ICOLA.CUSTOMERBRANCH.STOREID = ICOLA.STORE.STOREID
AND ICOLA.CUSTOMERBRANCH.BRANCHID NOT IN
(SELECT ICOLADM.CUSTOMER.BRANCHID FROM ICOLADM.CUSTOMER)) ;
END ;

CREATE PROCEDURE ICOLA.MARKET ( )
LANGUAGE SQL
SPECIFIC ICOLA.MARKET
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . MARKET ( REGION , STATE , CITY , COUNTRY , POPULATION )
(SELECT DISTINCT ICOLADM.REGIONS.REGION, ICOLA.CUSTOMERBRANCH.STATE ,
ICOLA.CUSTOMERBRANCH.CITY, ICOLADM.REGIONS.COUNTRY, 0
FROM ICOLA.CUSTOMERBRANCH, ICOLADM.REGIONS, ICOLADM.MARKET
WHERE CONCAT(RTRIM(ICOLA.CUSTOMERBRANCH.CITY), CONCAT(' ',
ICOLA.CUSTOMERBRANCH.STATE)) NOT IN
(SELECT CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE))
FROM ICOLADM.MARKET)
AND ICOLA.CUSTOMERBRANCH.STATE = ICOLADM.REGIONS.STATE ) ;
END ;

CREATE PROCEDURE ICOLA.PRODUCT ( )
LANGUAGE SQL
SPECIFIC ICOLA.PRODUCT
```

```
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN
INSERT INTO ICOLADM . PRODUCT ( UPC , NAME , BRAND , SIZE , CAFFEINATED ,
ALCOHOLIC , PACKAGETYPE , INTRODATE )
(SELECT UPC, NAME, BRAND, SIZE, CAFFEINATED, ALCOHOLIC, PACKAGETYPE, INTRODATE
FROM ICOLA.PRODUCT
WHERE UPC NOT IN (SELECT UPC FROM ICOLADM.PRODUCT) ) ;
UPDATE ICOLADM . PRODUCT SET DISCONTINUEDATE =
( SELECT DISCONTINUEDATE FROM ICOLA . PRODUCT
WHERE ICOLADM . PRODUCT . UPC = ICOLA . PRODUCT . UPC
AND ICOLA . PRODUCT . DISCONTINUEDATE IS NOT NULL ) ;
END ;
```

FACTTABLE プロシージャーは、データマートの FACTTABLE に情報を挿入します。 FACTTABLE プロシージャーは、オペレーショナル・データの ORDER 表から注文情報をプルし、他のオペレーショナル・データ表を利用して詳細情報を追加します。以下の FACTTABLE プロシージャー (EXCEPT キーワードを使用) は、OS/400 V5R3 でのみ使用できます。

```
CREATE PROCEDURE ICOLA.FACTTABLE ( )
LANGUAGE SQL
SPECIFIC ICOLA.FACTTABLE
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN INSERT INTO ICOLADM . FACTTABLE ( ORDERID , UPC , MARKETID , TIMEID,
QUANTITYSOLD , SELLINGPRICE , TOTALPRICE , COSTTOBUY , BRANCHID )
( ( SELECT ICOLA . ORDER . ORDERID , ICOLA . ORDER . UPC , ICOLADM . MARKET .
MARKETID , ( ( YEAR ( ICOLA . ORDER . DATE ) * 10000 ) + ( MONTH ( ICOLA . ORDER .
DATE ) * 100 ) + DAY ( ICOLA . ORDER . DATE ) ) , ICOLA . ORDER . QUANTITY , TRUNC (
ICOLA . ORDER . SUBTOTAL / ICOLA . ORDER . QUANTITY, 2 ) , ICOLA . ORDER . SUBTOTAL ,
ICOLA . PRICE . OURCOST , ICOLA . CUSTOMERCONTACT . BRANCHID
FROM ICOLA . ORDER , ICOLA . PRODUCT , ICOLA . PRICE , ICOLA . CUSTOMERCONTACT,
ICOLA . CUSTOMERBRANCH , ICOLADM . MARKET
WHERE ICOLA . ORDER . CUSTOMERID = ICOLA . CUSTOMERCONTACT . CUSTOMERID
AND ICOLA . CUSTOMERCONTACT . BRANCHID = ICOLA . CUSTOMERBRANCH . BRANCHID
AND ICOLA . CUSTOMERBRANCH . CITY = ICOLADM . MARKET . CITY
AND ICOLA . CUSTOMERBRANCH . STATE = ICOLADM . MARKET . STATE
AND ICOLA . ORDER . UPC = ICOLA . PRODUCT . UPC AND ICOLA . PRODUCT . PRICECODE =
ICOLA . PRICE . PRICECODE
AND ICOLA.ORDER.DATE < CURRENT DATE)
EXCEPT
( SELECT ORDERID , UPC , MARKETID , TIMEID , QUANTITYSOLD , SELLINGPRICE,
TOTALPRICE , COSTTOBUY , BRANCHID FROM ICOLADM . FACTTABLE ) ) ;
END ;
```

以下の FACTTABLE プロシージャーは、OS/400 V5R2 および V5R3 の両方で使用できます。

```
CREATE PROCEDURE ICOLA.FACTTABLE ( )
LANGUAGE SQL
SPECIFIC ICOLA.FACTTABLE
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
BEGIN INSERT INTO ICOLADM . FACTTABLE ( ORDERID , UPC , MARKETID , TIMEID,
QUANTITYSOLD , SELLINGPRICE , TOTALPRICE , COSTTOBUY , BRANCHID )
(SELECT ICOLA.ORDER.ORDERID, ICOLA.ORDER.UPC , ICOLADM.MARKET.MARKETID,
((YEAR(ICOLA.ORDER.DATE) * 10000) + (MONTH(ICOLA.ORDER.DATE) * 100) +
DAY(ICOLA.ORDER.DATE)), ICOLA.ORDER.QUANTITY,
TRUNC(ICOLA.ORDER.SUBTOTAL/ICOLA.ORDER.QUANTITY, 2), ICOLA.ORDER.SUBTOTAL,
ICOLA.PRICE.OURCOST, ICOLA.CUSTOMERCONTACT.BRANCHID
FROM ICOLA.ORDER, ICOLA.PRODUCT, ICOLA.PRICE, ICOLA.CUSTOMERCONTACT,
ICOLA.CUSTOMERBRANCH, ICOLADM.MARKET
WHERE ICOLA.ORDER.CUSTOMERID = ICOLA.CUSTOMERCONTACT.CUSTOMERID
AND ICOLA.CUSTOMERCONTACT.BRANCHID = ICOLA.CUSTOMERBRANCH.BRANCHID
AND ICOLA.CUSTOMERBRANCH.CITY = ICOLADM.MARKET.CITY
AND ICOLA.CUSTOMERBRANCH.STATE = ICOLADM.MARKET.STATE
AND ICOLA.ORDER.UPC = ICOLA.PRODUCT.UPC AND ICOLA.PRODUCT.PRICECODE =
ICOLA.PRICE.PRICECODE
AND ICOLA.ORDER.DATE < CURRENT DATE
AND ICOLA.ORDER.ORDERID NOT IN (SELECT ORDERID FROM ICOLADM.FACTTABLE)
) ;
END ;
```

第 9 章 付録 C

ルール・ファイルの作成で使用する SQL ステートメント

市場 (MARKET) ディメンションにデータを追加 : MRKT-RUL

```
SELECT ICOLADM.MARKET.REGION, ICOLADM.MARKET.STATE,
CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE))
FROM ICOLADM.MARKET
```

製品 (PRODUCT) ディメンションにデータを追加 : PROD-RUL

```
SELECT ICOLADM.PRODUCT.BRAND, ICOLADM.PRODUCT.NAME,
CONCAT(RTRIM(ICOLADM.PRODUCT.NAME), CONCAT(' ', CONCAT(CAST(ICOLADM.PRODUCT.SIZE
AS CHAR(2)), CONCAT(' OZ ', RTRIM(ICOLADM.PRODUCT.PACKAGETYPE)))),
ICOLADM.PRODUCT.SIZE, ICOLADM.PRODUCT.PACKAGETYPE, ICOLADM.PRODUCT.CAFFEINATED,
ICOLADM.PRODUCT.ALCOHOLIC
FROM ICOLADM.PRODUCT
```

ストア (STORE) ディメンションにデータを追加 : STOR-RUL

```
SELECT ICOLADM.CUSTOMER.STORENAME, CONCAT(RTRIM(ICOLADM.CUSTOMER.STORENAME),
CONCAT(' ', RTRIM(ICOLADM.CUSTOMER.BRANCHNAME)))
FROM ICOLADM.CUSTOMER
```

経費データをロードする QUERY : EXP-2003

```
SELECT CONCAT(RTRIM(ICOLADM.CUSTOMER.STORENAME), CONCAT(' ',
RTRIM(ICOLADM.CUSTOMER.BRANCHNAME))), MYTIME.MONTH,
CONCAT(RTRIM(ICOLADM.PRODUCT.NAME), CONCAT(' ', CONCAT(CAST(ICOLADM.PRODUCT.SIZE
AS CHAR(2)), CONCAT(' OZ ', RTRIM(ICOLADM.PRODUCT.PACKAGETYPE)))),
CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE)), 'COGS',
SUM(ICOLADM.FACTTABLE.COSTTOBUY * ICOLADM.FACTTABLE.QUANTITYSOLD)
FROM ICOLADM.FACTTABLE, ICOLADM.CUSTOMER, ICOLADM.TIME AS MYTIME,
ICOLADM.PRODUCT, ICOLADM.MARKET
WHERE ICOLADM.FACTTABLE.BRANCHID = ICOLADM.CUSTOMER.BRANCHID AND
ICOLADM.FACTTABLE.TIMEID = MYTIME.TIMEID AND ICOLADM.FACTTABLE.UPC =
ICOLADM.PRODUCT.UPC AND ICOLADM.FACTTABLE.MARKETID = ICOLADM.MARKET.MARKETID
AND MYTIME.YEAR = 2003
GROUP BY CONCAT(RTRIM(ICOLADM.CUSTOMER.STORENAME), CONCAT(' ',
RTRIM(ICOLADM.CUSTOMER.BRANCHNAME))), MYTIME.MONTH,
CONCAT(RTRIM(ICOLADM.PRODUCT.NAME), CONCAT(' ', CONCAT(CAST(ICOLADM.PRODUCT.SIZE
AS CHAR(2)), CONCAT(' OZ ', RTRIM(ICOLADM.PRODUCT.PACKAGETYPE)))),
CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE))
```

販売データをロードする QUERY : SAL-2003

```
SELECT CONCAT(RTRIM(ICOLADM.CUSTOMER.STORENAME), CONCAT(' ',
RTRIM(ICOLADM.CUSTOMER.BRANCHNAME))), MYTIME.MONTH,
CONCAT(RTRIM(ICOLADM.PRODUCT.NAME), CONCAT(' ', CONCAT(CAST(ICOLADM.PRODUCT.SIZE
AS CHAR(2)), CONCAT(' OZ ', RTRIM(ICOLADM.PRODUCT.PACKAGETYPE))))),
CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE)), 'SALES',
SUM(ICOLADM.FACTTABLE.TOTALPRICE)
FROM ICOLADM.FACTTABLE, ICOLADM.CUSTOMER, ICOLADM.TIME AS MYTIME,
ICOLADM.PRODUCT, ICOLADM.MARKET
WHERE ICOLADM.FACTTABLE.BRANCHID = ICOLADM.CUSTOMER.BRANCHID AND
ICOLADM.FACTTABLE.TIMEID = MYTIME.TIMEID AND ICOLADM.FACTTABLE.UPC =
ICOLADM.PRODUCT.UPC AND ICOLADM.FACTTABLE.MARKETID = ICOLADM.MARKET.MARKETID
AND MYTIME.YEAR = 2003
GROUP BY CONCAT(RTRIM(ICOLADM.CUSTOMER.STORENAME), CONCAT(' ',
RTRIM(ICOLADM.CUSTOMER.BRANCHNAME))), MYTIME.MONTH,
CONCAT(RTRIM(ICOLADM.PRODUCT.NAME), CONCAT(' ', CONCAT(CAST(ICOLADM.PRODUCT.SIZE
AS CHAR(2)), CONCAT(' OZ ', RTRIM(ICOLADM.PRODUCT.PACKAGETYPE))))),
CONCAT(RTRIM(ICOLADM.MARKET.CITY), CONCAT(' ', ICOLADM.MARKET.STATE))
```

EXP-2003 と EXP-2004 の QUERY の唯一の相違点は、SAL-2003 と SAL-2004 の QUERY 同様、最後の where 文節で指定する年度です。

- AND MYTIME.YEAR = 2004

第 10 章 特記事項

本書の情報は特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。他社製品への言及および参照は、単に情報提供目的で記載されたものであり、IBM がそれらの製品を推奨するものではありません。本書に含まれるパフォーマンス・データは、管理環境下で標準の IBM ベンチマークを使用し得られた測定結果と予測に基づくものです。ユーザーが実際に得られるスループットまたはパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、I/O 構成、記憶域構成、および処理されるワークロードなどの考慮事項によって異なります。したがって、個々のユーザーがここで述べる比率と同等のスループットまたはパフォーマンスの向上を得られるという保証はありません。

第 11 章 参照

- スター・スキーマの設計 (Designing the Star Schema)

(http://www.ciobriefings.com/whitepapers/starschema.asp)

- IBM(R) DB2 OLAP Server(TM) 8.1 Database Administrators Guide
- IBM DB2 Universal Database ビジネス・インテリジェンス・チュートリアル (IBM DB2 Universal Database Business Intelligence Tutorial)

(http://www-306.ibm.com/software/data/db2/db2olap/docs/V71docs/db2tu/frame3.htm#db2tussw)

- Hyperion(R) Analyzer Release 6.1 Installation Guide

Printed in Japan

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