DB2® ユニバーサル・データベース

# イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミングの手引き

*バージョン 7*

SC88-8609-00
(英文原典：SC26-9929-00)

DB2® ユニバーサル・データベース

# イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミングの手引き

*バージョン 7*

SC88-8609-00
(英文原典：SC26-9929-00)

**ご注意!**

本書、および本書がサポートする製品をご使用になる前に、 633ページの『付録C. 特記事項』にある一般的な情報を必ずお読みください。

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミング、またはサービスを、日本で発表する意図があることを必ずしも示すものではありません。

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

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

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

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

(URL は、変更になる場合があります)


| | |
|---|---|
| **原 典:** | SC26-9929-00<br>IBM® DB2® Universal Database<br>Image, Audio, and Video Extenders<br>Administration and Programming<br>Version 7 |
| **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 2000.6


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

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

# 目次

# 図

# 表

# 本書について

本書では、DB2 エクステンダーを使って画像や、音声、ビデオのデータを保管する DB2® データベースをどのように準備し、保守するかについて説明します。本書ではさらに、DB2 エクステンダーのユーザー定義関数 (UDF) とアプリケーション・プログラミング・インターフェース (API) を使って、これらのタイプのデータにどのようにアクセスし、操作するかについても説明します。これらの API をユーザー・プログラムの SQL ステートメント内の UDF に組み込んだり、API を組み込んだりすることによって、従来の数値データや文字データの他に、画像やビデオ・クリップなど従来扱えなかったデータにアクセスすることができます。

本書において ″DB2″ とは、DB2 UDB を指しています。

## 本書の対象読者

本書は、DB2 管理の概念、ツール、および手法の知識がある DB2 データベース管理者を対象としています。

本書はさらに、SQL の知識と、DB2 アプリケーション・プログラムに使用するプログラミング言語の 1 つについて知識がある DB2 アプリケーション・プログラマーをも対象としています。

本書は、DB2 イメージ、オーディオ、およびビデオ・エクステンダーを使用して作業する方々を対象としています。テキスト・エクステンダーを使用して作業される方は、次をご覧ください。 *DB2 テキスト・エクステンダー 管理およびプログラミング*

## 本書の使い方

本書の構成は次のとおりです。

『第 1 部 入門』

第 1 部は DB2 エクステンダーの概説です。 DB2 エクステンダーの管理やプログラミングが初めての読者は、ここをお読みください。

『第 2 部 画像、音声、ビデオのデータ管理』

第 2 部は、DB2 データベースの画像、音声、およびビデオ・データの準備と保守についても説明します。画像、音声、またはビデオ・データを含む DB2 データベースを管理する 必要がある読者は、ここをお読みください。

『第 3 部 画像、音声、ビデオのデータのためのプログラミング』

第 3 部は、DB2 エクステンダーの UDF と API を使用して画像、音声、ビデオのデータに対する操作を要求する方法についての説明です。

『第 4 部 参照情報』

第 4 部は、DB2 エクステンダーの UDF、API、管理コマンド、およびメッセージやコードなどの診断情報についての説明です。 DB2 エクステンダーの概念とタスクについての知識がある読者が、DB2 エクステンダーの特定の UDF、API、コマンド、メッセージ、コードについて情報を必要とするときに参照してください。

『付録』

付録は、以下について説明します。

- DB2 エクステンダーによって使用される環境変数をどのように設定すれば、画像、音声、ビデオのオブジェクトのファイルを検出し、それらの表示プログラムまたは再生プログラムを指定することができるか。
- エクステンダーで提供されるサンプル・プログラムとメディア・ファイルのインストールと使用法。

## プラットフォーム固有の情報

DB2 エクステンダーは、DB2 ユニバーサル・データベースの単一区分データベース環境、または DB2 ユニバーサル・データベース エンタープライズ拡張エディションの複数区画環境で使用できます。

本書には、上記の環境で DB2 エクステンダーを使用するにあたっての情報が収められています。 DB2 ユニバーサル・データベース エンタープライズ拡張エディションの複数区画環境でのエクステンダーの使用にのみ該当する情報には、″**EEE のみ**″と記載されます。一方、DB2 ユニバーサル・データベースの単一区画環境でのエクステンダーの使用にのみ該当する情報には、″**EEE 以外のみ**″と記載されます。特定の環境に該当する記載がない情報は、両方の環境に適用されます。

## 強調表示の規則

本書では、次の規則を使用します。

**太字**　太字テキストは、新出用語の定義を示すときに使用します。

*イタリック*
: イタリック体は、値で置き換える変数パラメーターを示したり、テキストで使用された語を強調するときに使用します。

**大文字** 英大文字は、次のものを表すときに使用します。

- データ・タイプ
- ディレクトリー名
- フィールド名
- API 呼び出し
- コマンド
- キーワード
- 変数名

**例** 例を示すテキストは、システム・メッセージや、ユーザーが入力する値を示すときに使用します。例のテキストは、コーディング例でも使用します。

## 構文図の読み方

本書では、コマンドや SQL の構文を構文図を使って説明します。構文図は、次のように読みます。

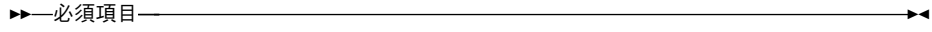
- 構文図は、左から右、上から下に、線に沿って読みます。

  ▶▶── 記号は、ステートメントの始まりを示します。

  ──▶ 記号は、ステートメント構文が次の行に続くことを示します。

  ▶── 記号は、そのステートメントが前の行からの続きであることを示します。

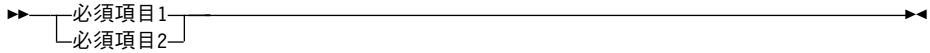
  ──▶◀ 記号は、ステートメントの終わりを示します。
- 必須項目は、水平線 (メインパス) 上に表示します。




- オプション項目は、メインパスの下に表示します。


- 2 つ以上の項目から選択可能な場合は、それらを上下に重ねてスタック表示します。

  それらの項目から 1 つを選択しなければならない 場合は、スタック項目の 1 つをメインパスに表示します。




それらの項目を全く選択しないことが可能な場合は、スタック全体をメインパスの下に表示します。


スタックの上の繰り返し矢印は、スタック項目から複数の項目を選択できることを示します。


- キーワードは英大文字で表します (たとえば、`/DB2IMAGE:`)。キーワードを指定する場合には、そのとおりに入力しなければなりません。変数は小文字で表します (たとえば、`srcpath`)。変数は、構文内のユーザー指定の名前や値を表します。
- 句点記号、括弧、算術演算子などの記号が表示されている場合には、それらをその構文の一部として指定する必要があります。

## 関連情報

### DB2 ユニバーサル・データベース

*概説およびインストール*、GC88-8534 (OS/2®)、GC88-8537 (Windows®)、GC88-8536 (UNIX)。これらの資料は、該当するプラットフォームで DB2 を計画し、インストールし、構成し、移行する方法を説明しています。

*DB2 ユニバーサル・データベース エンタープライズ拡張エディション 概説およびインストール*, GC88-8530 (AIX®)、GC88-8529 (Windows)。これらの資料は、該当するプラットフォームで DB2 ユニバーサル・データベース・エンタープライズ拡張エディションを計画し、インストールし、構成する方法を説明しています。

*管理の手引き: 計画* (第 1 巻) (SC88-8513)。本書は、データベース概念について概説し、設計 (たとえば、論理および物理データベース設計) に関する情報を提供し、高い可用性について解説しています。

*管理の手引き: インプリメンテーション* (第 2 巻) (SC88-8511)。本書は、設計、データベースへのアクセス、監査、バックアップ、および回復などのインプリメンテーションについて説明しています。

*管理の手引き: パフォーマンス* (第 3 巻) (SC88-8512)。本書は、データベース環境について解説し、さらにアプリケーションのパフォーマンスの評価と調整の方法について説明しています。

*アプリケーション開発の手引き* (SC88-8516)。この資料は、組み込み SQL または JDBC を使用して DB2 データベースにアクセスするアプリケーションを開発する方法について説明しています。また、ストアード・プロシージャーおよびユーザー定義関数を作成する方法、ユーザー定義タイプを定義する方法、およびトリガーを使用する方法についても説明しています。

*コール・レベル・インターフェースの手引きおよび解説書* (SC88-8517)。この資料は、Microsoft ODBC 仕様と互換性のある呼び出し可能 SQL インターフェースである DB2 コール・レベル・インターフェースを使用して DB2 データベースにアクセスするアプリケーションを開発する方法について説明しています。

*コマンド解説書* (SC88-8518)。この資料は、DB2 コマンド行プロセッサーの使用法を説明し、DB2 コマンドに関する参照情報を提供します。

*メッセージ解説書* (GC88-8543) および (GC88-8544)。この資料は、DB2 の使用するメッセージおよびコードをリストし、指定されたエラーまたは問題から回復するためにユーザーが行える処置を説明します。

**DB2 ユニバーサル・データベース テキスト・エクステンダー**

*DB2 ユニバーサル・データベース テキスト・エクステンダー 管理およびプログラミング バージョン 7* (SC88-8610)。テキスト・データの DB2 データベースを管理する方法について説明しています。さらに、DB2 テキスト・エクステンダーのアプリケーション・プログラミング・インターフェースを使ってテキスト・データのアクセスと操作を行う方法についても説明しています。

**DB2 ユニバーサル・データベース XML エクステンダー**

*DB2 ユニバーサル・データベース XML エクステンダー 管理およびプログラミングの手引き (SB88-8505)*。 XML 文書用の DB2 データベースを管理する方法について説明しています。さらに、DB2 XML エクステンダーのアプリケーション・プログラミング・インターフェースを使って XML データのアクセスと操作を行う方法についても説明しています。

**DB2 ユニバーサル・データベース 地理情報エクステンダー**

*地理情報エクステンダー 使用者の手引きおよび解説書* (SC88-8624)。本書は、地理情報エクステンダーのインストール、構成、管理、プログラミング、およびトラブルシューティングに関する情報を提供します。また、地理情報データの概念についての重要事項を示し、地理情報エクステンダー固有の参照情報 (メッセージおよび SQL) を提供します。

**DB2 ユニバーサル・データベース (OS/390 版) イメージ、オーディオ、およびビデオ・エクステンダー**

*DB2 Universal Database for OS/390 Version 6 Image, Audio, and Video Extenders Administration and Programming* (SC26-9650)。この資料は、画像データ、音声データ、およびビデオ・データ用の DB2 (OS/390 版) データベース・サーバーの管理方法を説明しています。また、ユーザー定義関数、および DB2 (OS/390 版) イメージ、オーディオ、およびビデオ・エクステンダーの提供するアプリケーション・プログラミング・インターフェースを使用して、画像、音声、およびビデオ・データにアクセスして操作する方法も説明しています。

**DB2 ユニバーサル・データベース (OS/390 版) テキスト・エクステンダー**

*DB2 Universal Database for OS/390 Version 6 Text Extender Administration and Programming* (SC26-9651)。 DB2 (OS/390 版) データベース・サーバーをテキスト・データ用に管理する方法について説明しています。さらに、DB2 (OS/390 版) テキスト・エクステンダーのユーザー定義関数とアプリケーション・プログラミング・インターフェースを使って、テキスト・データのアクセスと操作を行う方法についても説明しています。

**WWW (ワールド・ワイド・ウェブ)**

DB2 エクステンダー Web サイト。この Web サイトでは、DB2 エクステンダーの情報やエクステンダーで使用する技術の情報を提供しています。 DB2 エクステンダーのホーム・ページの URL は、次のとおりです。

http://www.ibm.com/software/data/db2/extenders

# 第1部 入門

# 第1章 概説

DB2 (DB2) ユニバーサル・データベース (UDB) は、強力なオブジェクト・リレーショナル・データベース・マネージャーです。これは、従来の数値や文字のデータだけでなく、複雑なオブジェクト (LOB) を保管し、保護します。

DB2 エクステンダーは、DB2 のオブジェクト関連機能を活用するのに役立ちます。エクステンダーでは、画像、音声、ビデオ、テキストのオブジェクトに対して個別のデータ・タイプと特別な関数が定義されます。エクステンダーを使用すれば、アプリケーションでこれらのデータ・タイプと関数を定義する時間と労力が節約できます。これらのデータ・タイプと関数は SQL により使用します。したがって、エクステンダーは、ユーザーのアプリケーションが単一のアクセス・ポイントから、従来の数値や文字のデータを含むあらゆる種類のこれらのデータにアクセスできるようにします。さらに、エクステンダーは、アプリケーションが新規の方法で情報探索できるようにします。たとえば、視覚的な特性により、つまり、色やテクスチャーの眼に見える例を使うことによって、画像を探索することができます。

## DB2 の利用

DB2 エクステンダーは、DB2 のオブジェクト指向機能を利用します。特に、DB2 を使って、次の事柄を行えます。

- 最高 2 ギガバイトの LOB を DB2 データベースに保管する。
- これらの大型複合オブジェクトに対し個別のデータ・タイプを定義する。これらのユーザー定義タイプ (UDT) は、オブジェクト (たとえば、画像または音声によって表されたデータのタイプ) を識別するために使用されます。
- ユーザー定義のデータ・タイプに対して使用できる関数を定義する。たとえば、画像の中に色がいくつあるかを数える関数や、音声のサンプリング・レートを知る関数を定義することができます。これらのユーザー定義関数 (UDF) は、他の SQL 関数と同じ方法で SQL ステートメントに指定することができます。

DB2 エクステンダーは、画像、音声、ビデオ、テキストのオブジェクトに対し UDT と UDF を作成します。これらの UDT と UDF は、ユーザーが次のことを行う上で重要な役割を果たします。

- アプリケーションの開発。データ・タイプと関数はエクステンダーが定義するので、それらをアプリケーションで定義する必要はありません。

- 一貫性の確保。同じ組み合わせのエクステンダーの UDT と UDF が、すべてのアプリケーションに使用できます。ラージ・オブジェクトを扱う複数のアプリケーションでは実現できないようなある一定レベルの一貫性が確保できます。
- 強力な照会の作成。 UDF は、他の SQL 関数と同じ方法で要求されますので、アプリケーションから複数データ・タイプの照会を行うことができます。また、1 つの SQL ステートメントから、従来の数値や文字のデータだけでなく、画像、音声、ビデオ、テキストのオブジェクトを同時にアクセスすることができます。 UDF と UDT は、組み込み SQL ステートメントに指定することもできますし、DB2 コール・レベル・インターフェース (DB2 CLI) 呼び出しに指定することもできます。

エクステンダーが処理するオブジェクトは DB2 データベースに保管できますので、これらのオブジェクトの場合、データベースに保管された従来のデータ・タイプと同じ機密保護、保全性、回復が保証されます。

また、DB2 エクステンダーは DB2 ユニバーサル・データベース・エンタープライズ拡張エディションの区分データベース環境を活用します。区画化により、単一のコンピューターには大き過ぎて納まりきらないデータベースでもアプリケーションが使用できるようになります。また、区画化により SQL 操作を並列に実行することもでき、そのようにして SQL の照会やユーティリティーの処理速度を向上させることができます。

## 新しい強力な方法による情報の探索

DB2 エクステンダーでは、アプリケーションからいろいろな方法で情報を探索することができます。たとえば、アプリケーションから、データベースに保管された従来のデータのタイプに対応するオブジェクトを探索することができます。具体的には、音声クリップをその名称や録音された日付で探索できます。あるいは、アプリケーションから、ビデオ・クリップの再生時間など、それ固有の特性によってオブジェクトを探索することができます。エクステンダーは、これらの特性を自動的に判断し、保管することによって、探索で使用します。

アプリケーションから画像を内容によって探索することさえできます。たとえば、アプリケーションから眼に見える例を使って画像を探索するとします。そのようなアプリケーションでは、ユーザーが例としての画像を選択すれば、その例に似た色やテクスチャーのイメージが探索されます。 DB2 エクステンダーのイメージ内容照会 (QBIC) 機能を使用すれば、このような方法を使って画像を探索するアプリケーションを作成することができます。

## DB2 エクステンダー

DB2 エクステンダー・パッケージは、イメージ・エクステンダー、オーディオ・エクステンダー、ビデオ・エクステンダー、およびテキスト・エクステンダーから構成されます。

本書では、イメージ、オーディオ、およびビデオ・エクステンダーについて網羅しています。これ以後、本書で 『エクステンダー』 または 『DB2 エクステンダー』 という場合には、特に断らない限りイメージ、オーディオ、およびビデオ・エクステンダーを意味します。テキスト・エクステンダーについては、テキスト・エクステンダー *管理およびプログラミング* を参照してください。 XML エクステンダーについては、 *XML エクステンダー管理およびプログラミングの手引き* を参照してください。

## SDK および実行時環境

DB2 エクステンダーのインストール・パッケージでは、ソフトウェア開発者キット (SDK) およびクライアントとサーバーの実行時環境が提供されています。 DB2 エクステンダー・アプリケーションは、 DB2 エクステンダー SDK をインストールしたクライアントまたはサーバー・マシン上で開発できます。

DB2 エクステンダー・アプリケーションは、DB2 エクステンダーのクライアント実行時コードおよびサーバー実行時コードが組み込まれたサーバー・マシン上で実行できます。 (サーバー実行時コードをインストールすると、クライアント実行時コードは自動的にインストールされます。) また、DB2 エクステンダー・アプリケーションを、DB2 エクステンダーのクライアント実行時コードがインストールされているクライアント・マシン上で実行することもできます。クライアント・マシンからエクステンダー・アプリケーションを実行する場合には、サーバーへの接続が可能であることを確認する必要があります。

## エクステンダーの用法

エクステンダー UDF は、DB2 アプリケーション・プログラムから要求することもできますし、DB2 コマンド行プロセッサーを使用して対話的に呼び出すこともできます。

エクステンダーには、以下のアプリケーション・プログラミング・インターフェース (API) が備わっています。

- 画像データ、音声データおよびビデオ・データのデータベースを準備および維持する管理 API。

- 画像を表示したり、ビデオ・クリップや音声クリップを再生したりする表示および再生 API。
- QBIC API。内容による画像の準備および画像の探索を行うためのものです。(内容による探索は、UDF を使って行うこともできます。)
- ビデオ・ショット検出 API。ビデオのシーンの切り替えにもとづいてフレームの順序を識別するためのものです。

さらに、DB2 エクステンダーでは、管理コマンドを出すコマンド行プロセッサーを提供しています。エクステンダーのコマンド行プロセッサーと DB2 のコマンド行プロセッサーを区別するために、前者を 『db2ext コマンド行プロセッサー』、後者を『DB2 コマンド行プロセッサー』と呼びます。

## 例

ある広告代理店が広告の DB2 データベースを保持しているとします。これまで、この代理店は、広告キャンペーンを行うたびに、クライアントの名前や広告が終了した日付など、数値や文字のデータを保管していました。 DB2 UDB と DB2 エクステンダーのインストールにより、この代理店は広告の内容もデータベースに保管することにしました。これには、印刷広告の画像や、テレビ広告のビデオ、ラジオ広告の録音があります。 7ページの図1 に示すように、関連するすべての広告情報は、ADS という 1 つのデータベース表に保管されます。


*図1. マルチメディア・データベース表.* この表には、従来のデータ・タイプの他に、画像、音声、ビデオのデータが入ります。ビデオ、音声、画像が示されます。

## 例 1: ビデオをその特性によって検索する

広告代理店のある顧客マネージャーは、1997 年に IBM のために作成したビデオ広告のうち 30 秒以内のものを探しています。

8ページの図2 は、これらのビデオにアクセスするための照会です。この照会には、`Filename` と `Duration` という名前のビデオ・エクステンダー UDF が指定されていることに注意してください。

```
SELECT FILENAME(ADS_VIDEO)
FROM ADS
WHERE CLIENT='IBM' AND
SHIP_DATE>='01/01/1997' AND
DURATION(ADS_VIDEO) <=30
```

*図 2. ビデオをアクセスする照会*

この照会によって、該当するビデオのファイル名が戻されます。そうすると、顧客マネージャーは好きなビデオ・プレーヤーを開始して、それぞれのビデオ・ファイルの内容を再生することができます。

図2 は、顧客マネージャーが対話的に発行できる照会の例です。通常、顧客マネージャーは、アプリケーション・プログラムを使ってビデオを検出し、再生します。たとえば、図3 は、C でコーディングされたそのようなアプリケーションの主要要素を示しています。このアプリケーションでは、hvVid_fname という名前の DB2 ホスト変数にビデオ・ファイルの名前が取り出されます。さらに、このアプリケーションでは、DBvPlay という名前の再生 API を使ってビデオが再生されます。

```
#include <dmbvideo.h>

int count = 0;

EXEC SQL BEGIN DECLARE SECTION;
char hvClient[30];                      /*client name*/
char hvCampaign[30];                    /*campaign name*/
char hvSdate[8];                        /*ship date*/
char hvVid_fname [251]                  /*video file name*/


EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE c1 CURSOR FOR
     SELECT CLIENT, CAMPAIGN, SHIP_DATE, FILENAME(ADS_VIDEO)
       FROM ADS
       WHERE CLIENT='IBM' AND
         SHIP_DATE≥'01/01/1997' AND
         DURATION(ADS_VIDEO)≤30
FOR FETCH ONLY;
```

*図 3. ビデオのアクセスと再生を行うアプリケーション (1/2)*

```
EXEC SQL OPEN c1;
for (;;){
        EXEC SQL FETCH c1 INTO :hvClient, :hvCampaign,
                          :hvSdate, :hvVid_fname;
    if (SQLCODE != 0)
              break;
    printf("¥nRecord %d:¥n", ++count);
    printf("Client = '%s'¥n", hvClient);
    printf("Campaign = '%s'¥n", hvCampaign);
    printf("Sdate = '%s'¥n", hvSdate);
    rc=DBvPlay(NULL,MMDB_PLAY_FILE,hvVid_fname,MMDB_PLAY_WAIT);
}
EXEC SQL CLOSE c1;
```

*図3. ビデオのアクセスと再生を行うアプリケーション (2/2)*

## 例 2: 画像をその内容によって探索する

広告代理店のあるグラフィック・イラストレーターが、クライアントの新しい印刷広告を製作しています。このイラストレーターは、広告の背景に特定の青で薄い影を入れたいと思い、この代理店で以前に製作した印刷広告でその色が使われたことがあるかどうかを調べようとしています。そのために、イラストレーターは、内容によって画像を探索するアプリケーションを実行します。画像は、データベース表に保管されています (7ページの図1 を参照)。このアプリケーションでは、眼に見える例 (つまり、求める色を示す画像) をユーザーが指定する必要があります。これを指定すると、アプリケーションによってその色が分析され、その例に最も似ている色の画像が検索されます。

10ページの図4 は、眼に見える例と、その色に最も似ているものとして検索された画像を示しています。

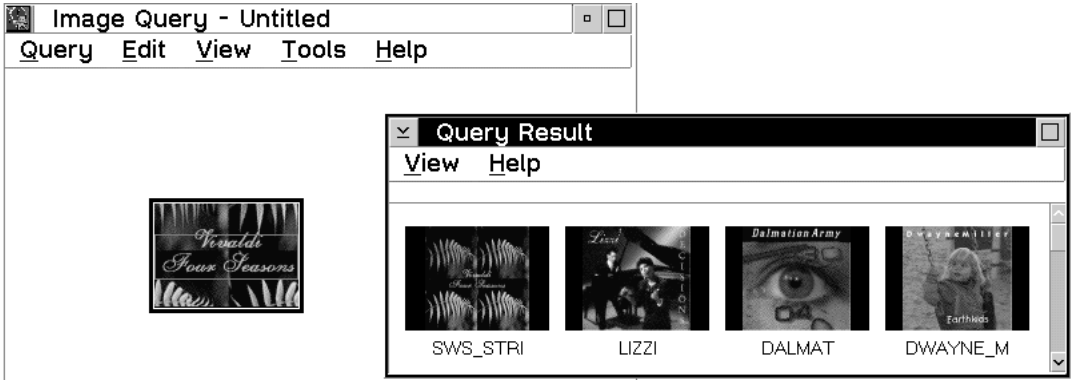



*図 4. 内容による画像の探索.* 眼に見える例によって、平均的な色による画像の探索が行われます。

11ページの図5 は、このアプリケーションの主要要素を示したものです。このアプリケーションは、`QbQueryCreate` という QBIC API を使って QBIC 照会を作成し、`QbQueryAddFeature` と `QbQuerySetFeatureData` を使って色の選択をその照会に追加し、`QbQuerySearch` を使ってその照会を実行し、`QbQueryDelete` を使ってその照会を削除します。さらに、このアプリケーションは、`DBiBrowse` というグラフィカル API を使って、検索された画像を表示します。

```
#include <dmbqbqpi.h>

#define MaxQueryReturns  10

static SQLHENV henv;
static SQLHDBC hdbc;
static SQLHSTMT hstmt;
static SQLRETURN rc;

void main(int argc, char* argv[])
{
 char           line[4000];
 char*          handles[MaxQueryReturns];
 QbQueryHandle  qHandle=0;
 QbResult       results[MaxQueryReturns];
 SQLINTEGER     count;
 SQLINTEGER     resultType=qbiArray;

 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc, (SQLCHAR*)"qtest", SQL_NTS,
              (SQLCHAR*)"", SQL_NTS, (SQLCHAR*)"", SQL_NTS);

 if (argc !=2) {
 printf("usage: query colorname¥n");
          exit(1);
 }

 QbImageSource is;
 is.type = qbiSource_AverageColor;

 /* run the get color subroutine */
 getColor(argv[1], is.average.Color);

 QbQueryCreate(&qhandle);
 QbQueryAddFeature(qhandle, "QbColorFeatureClass");
 QbQuerySetFeatureData(qhandle, "QbColorFeatureClass",&is);
 QbQuerySearch(qhandle, "ADS", "ADS_IMAGE", 10, 0, resultType
                              &count, results);
 for (int j = 0; j <count; j++) {
 printf(j,":¥n");


DBiBrowse("usr/local/bin/xv %s", MMDB_PLAY_HANDLE, handles[j],
                MMDB_PLAY_WAIT);
 }
```

*図 5. 内容によって画像を探索するアプリケーション (1/2)*

## 例

```
QbQueryDelete(qhandle);

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
```

*図 5. 内容によって画像を探索するアプリケーション (2/2)*

## 操作環境

DB2 エクステンダー バージョン 7 は、クライアント / サーバー環境において、DB2 ユニバーサル・データベース　バージョン 7.1 (またはそれ以降) で動作します。サポートされるプラットフォームで必要なバージョンとリリースのレベルは、DB2 ユニバーサル・データベース バージョン 7.1 の場合と同じです。

サポートされるクライアント・プラットフォームは次のとおりです。 OS/2、AIX、Windows NT® およびそれ以降、Windows 95、Windows 98、Solaris 実行環境、および HP-UX。

サポートされるサーバーは次のとおりです。 OS/2、AIX、Windows NT およびそれ以降、Solaris 実行環境、および HP-UX。

13ページの図6 は、サポートされるプラットフォームを示しています。

もう 1 つの DB2 エクステンダー製品である DB2 ユニバーサル・データベース (OS/390 版) エクステンダーでは、OS/390 クライアントおよびサーバーをサポートしています。 DB2 ユニバーサル・データベース (OS/390 版) エクステンダーの詳細については、 *DB2 Universal Database for OS/390 Image, Audio, and Video Extenders Administration and Programming* または *DB2 Universal Database for OS/390 Text Extender Administration and Programming* を参照してください。

DB2 エクステンダーは単一区分データベース環境で動作することができます。

**EEE のみ:** エクステンダーは、以下のプラットフォームの複数区分データベース環境で動作することもできます。AIX、Solaris 実行環境、および Windows NT およびそれ以降。

DB2 エクステンダーを複数データベース環境で操作するには、DB2 ユニバーサル・データベース エンタープライズ拡張エディションをともに使用することが必要です。

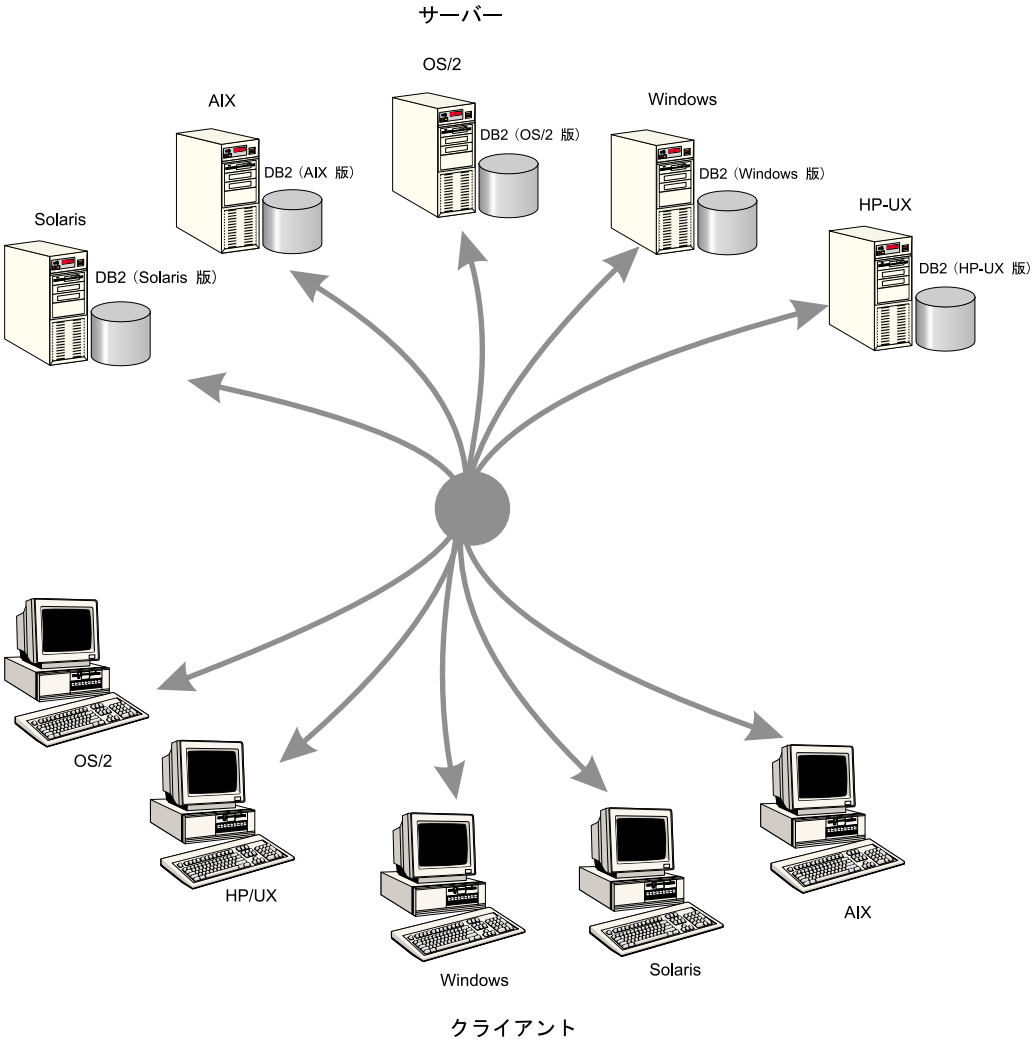



*図 6. DB2 エクステンダー・プラットフォーム*

## 例

# 第2章 DB2 エクステンダーの概念

この章では、DB2 エクステンダーを使用する前に知っておくべき概念について説明します。



オブジェクト指向の概念については、 *DB2 アプリケーション開発の手引き* を参照してください。

## オブジェクト指向の概念

DB2 は、**オブジェクト指向**をサポートしています。この概念は、具象または抽象のいずれであれ、すべてのものがアプリケーションの中でオブジェクト (操作とデータ値の集合で構成される) として表現できるというものです。たとえば、文書は、文書データと、その文書に対して行う操作 (ファイリング、送信、印刷など) からなる文書オブジェクトによって表すことができます。ビデオ・クリップは、ビデオ・データと、ビデオ・クリップの再生や特定のビデオ・フレームの検索などの操作からなるビデオ・オブジェクトとして表すことができます。実世界の対象物の場合と同じように、何かを表すオブジェクトには属性があります。たとえば、ビデオ・オブジェクトには、圧縮タイプやサンプリング率などの属性を与えることができます。

オブジェクトは、タイプごとにグループ化することができます。同じタイプのオブジェクトは、同じ属性を持ち、同じように動作します。つまり、同じ操作に対応しています。たとえば、あるビデオ・タイプに、ある圧縮タイプ属性を定義すると、そのビデオ・タイプのすべてのオブジェクトがその属性をもちます。そのビデオ・タイプのあるオブジェクトが再生可能であれば、そのビデオ・タイプのすべてのオブジェクトが再生可能です。

DB2 のオブジェクト指向のサポートでは、オブジェクト・タイプのインスタンスを表の列に保管し、それらを SQL ステートメントの関数によって操作することが可能です。たとえば、ビデオ・オブジェクトを表の列に保管し、それら

を SQL 関数を使って操作することができます。また、保管したオブジェクトの属性と動作は、アプリケーションの間で共用することができます。すべてのアプリケーションでは、同じオブジェクト・タイプは、同じ属性と動作の組み合わせとして「認識」されます。

ビデオ・オブジェクトは大きくて複雑であるのが普通です。画像オブジェクトや音声オブジェクトも同様です。 DB2 では、オブジェクト指向サポートの一環として、ラージ・オブジェクト (LOB) をデータベースに保管することが可能です。さらに、これらの LOB を、ユーザー定義タイプ (UDT)、ユーザー定義関数 (UDF)、およびトリガーを使って定義および操作することができます。

## ラージ・オブジェクト

DB2 では、**ラージ・オブジェクト** (LOB) を次のオブジェクトとしてデータベースに保管することができます。

- 2 進ラージ・オブジェクト (BLOB)
- 文字ラージ・オブジェクト (CLOB)
- 2 バイト文字ラージ・オブジェクト (DBCLOB)

BLOB は 2 進ストリングです。画像、音声、ビデオの各オブジェクトは、BLOB として DB2 データベースに保管されます。 CLOB は、1 バイト文字からなる文字ストリングで、対応するコード・ページがあります。このデータ・タイプは、1 バイト文字を含むテキスト・オブジェクトに対して使用されます。 DBCLOB は、2 バイト文字からなる文字ストリングで、対応するコード・ページがあります。このデータ・タイプは、2 バイト文字を含むテキスト・オブジェクトに対して使用されます。

各 LOB の長さは 2 ギガバイトまで可能ですが、DB2 では 1 つの表で多くの LOB 列が使用できます。 1 つの行では最高 24 ギガバイトの LOB スペースが、1 つの表では最高 4 テラバイトの LOB スペースが使用できます。

LOB の内容は、そのサイズが大きいため、ユーザー表に直接は保管されません。表の中では、各 LOB はラージ・オブジェクト記述子によって識別されます。ディスクの他の場所に保管されているラージ・オブジェクトをアクセスするときには、その記述子が使用されます。

さらに、DB2 エクステンダーでは、LOB の内容をファイルに保管し、データベースからそれを指し示すことができます。この方法を使用するには、DB2 エクステンダーでオブジェクトを保管するときにそのように指定します。

## ユーザー定義タイプ

画像、ビデオ、音声の各オブジェクトは、データベースに BLOB として表されます。**ユーザー定義タイプ** (UDT、**特殊タイプ**ともいいます) を使用すれば、BLOB を区別することができます。たとえば、ある UDT を画像オブジェクトに対して作成し、別のものを音声オブジェクトに対して作成することができます。画像および音声オブジェクトは、BLOB として保管されていても、BLOB や互い同士とは区別される異なるタイプとして処理されます。

UDT は、SQL CREATE DISTINCT TYPE ステートメントによって作成します。たとえば、マップの地理特性を処理するアプリケーションを開発するとします。この場合、マップ・オブジェクトに対して map という別個のタイプを作成することができます。

```
CREATE DISTINCT TYPE map AS BLOB (1M)
```

マップ・タイプのオブジェクトは、内部的には長さが 1 メガバイトの BLOB として表されますが、別個のタイプをもつオブジェクトとして扱われます。

表の列に保管されたデータを記述する場合、UDT を SQL の組み込みタイプと同じように使用することができます。次の例では、表の作成で、列がマップ・データのタイプをもつように指定されます。

```
CREATE TABLE places
  (locid     INTEGER NOT NULL,
   location  CHAR (50),
   grid      map)
```

それぞれの DB2 エクステンダーでは、そのタイプ (つまり、画像、音声、ビデオ) に対して UDT が作成されます。

## ユーザー定義関数

**ユーザー定義関数** (UDF) の機能を使用すれば、SQL 関数を作成して、DB2 で提供される組み込み関数群に追加することができます。具体的には、画像や、音声、ビデオのオブジェクトに特有な操作を行う UDF を作成することができます。たとえば、UDF を作成して、ビデオの圧縮形式を得たり、音声のサンプリング率を返したりすることが可能です。この方法を使えば、特定タイプのオブジェクトに対しその動作を定義することができます。たとえば、ビデオ・オブジェクトはビデオ・タイプ用に作成された関数に従って動作します。また、イメージ・オブジェクトはイメージ・タイプ用に作成された関数に従って動作します。

UDF は、SQL CREATE FUNCTION ステートメントを使って作成します。このステートメントでは、特にその UDF を適用するデータ・タイプを指定しま

す。たとえば、次のステートメントを指定すれば、マップのスケールを計算する map_scale という UDF が作成されます。この UDF では、それが適用されるデータ・タイプとして map が指定されています。この関数は C 言語で書かれ、その名前が EXTERNAL NAME 文節で指定されています。

```
CREATE FUNCTION map_scale (map)
  RETURNS SMALLINT
  EXTERNAL NAME 'scale!map'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
```

UDF は、組み込み関数と同じようにして SQL ステートメントで使用することができます。次の例では、SQL SELECT ステートメントに map_scale UDF を指定して、grid という表の列に保管されているマップのスケールを戻します。

```
SELECT map_scale (grid)
  FROM places
  WHERE location='SAN JOSE, CALIFORNIA'
```

それぞれの DB2 エクステンダーでは、そのタイプに対して UDF 群 (つまり、画像特有の UDF、音声特有の UDF、ビデオ特有の UDF) が作成されます。これらの UDF を SQL ステートメントで使用すれば、画像を表に保管したり、ビデオのフレーム率を入手したり、音声の注釈を追加したり、といったエクステンダー機能を要求することができます。

## UDF 名と UDT 名

DB2 関数の正式名は、*スキーマ名.関数名* です。ここで、*スキーマ名* は SQL オブジェクトの論理的なグループを表す識別子です。 DB2 エクステンダー UDF のスキーマ名は MMDBSYS です。 MMDBSYS というスキーマ名は、DB2 エクステンダー UDT の修飾子でもあります。

UDF や UDT を参照する場合には、どこでも正式な名前が使用できます。たとえば、MMDBSYS.CONTENT は、スキーマ名が MMDBSYS で、関数名が CONTENT の UDF を表します。 MMDBSYS.DB2IMAGE は、スキーマ名が MMDBSYS で、特殊タイプ名が DB2IMAGE の UDT を表します。スキーマ名は、UDF や UDT を参照するとき省略することができます。この場合、DB2 は関数パスを使って、必要な関数と特殊データ・タイプを判別します。

### 関数パス

**関数パス**とは、スキーマ名が順に並んだリストです。 DB2 は、リスト内のスキーマ名の順序を使って、関数や特殊データ・タイプへの参照を解決します。関数パスの指定には、SQL ステートメント SET CURRENT FUNCTION PATH

を使用します。これを使用すると、関数パスが CURRENT FUNCTION PATH という特殊レジスターに設定されます。

DB2 エクステンダーでは、mmdbsys スキーマを関数パスに追加するのが適切です。これにより、DB2 エクステンダー UDF および UDT を、名前の前に mmdbsys を付けなくても入力することができます。以下の例は、mmdbsys スキーマを関数パスに追加する方法を示しています。

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

mmdbsys としてログオンする場合、**mmdbsys を関数パスの最初のスキーマとして追加しないでください。** mmdbsys ユーザー ID でログオンする場合、関数パスにおける最初のスキーマは mmdbsys に設定されます。次に SET CURRENT FUNCTION PATH ステートメントで関数パスの最初のスキーマを mmdbsys に設定すると、その関数パスの始めの 2 つが mmdbsys スキーマとなり、エラー状態になります。

### 多義化された関数名

関数名を**多義化**することができます。つまり、複数の UDF が、同じスキーマにおいてさえ同じ名前をもつことがあります。しかし、2 つの関数が同じ**シグニチャー**をもつことはできません。シグニチャーとは、修飾された関数名と、すべての関数パラメーターの定義済みデータ・タイプとを連結したものです。

## トリガー

**トリガー**は、表の変更によって活動化されるアクション群を定義するためのものです。トリガーは、入力データの妥当性を検査したり、新たに挿入された行の値を自動的に生成したり、相互参照のために他の表を読み込んだり、監査の目的で他の表に書き込んだりするために使用することができます。トリガーは、保全性を検査したり、業務上の規則を適用するためにしばしば使用されます。

トリガーを作成するには、SQL CREATE TRIGGER ステートメントを使用します。次のステートメントでは、部品目録に関する業務処理を行うためのトリガーが作成されます。このトリガーは、在庫数が在庫可能な最大数の 10 パーセントより少なくなると、部品を再注文します。

```
CREATE TRIGGER reorder
       AFTER UPDATE OF on_hand, max_stocked ON parts
       REFERENCING NEW AS n_row
       FOR EACH ROW MODE DB2SQL

       WHEN (n_row.on_hand < 0.10 * n_row.max_stocked)
       BEGIN ATOMIC
```

```
                VALUES(issue_ship_request(n_row.max_stocked -
                                          n_row.on_hand,
                                          n_row.partno));
        END
```

DB2 エクステンダーは管理サポート表を作成し、保守して、データベース内に保管されている画像、音声、ビデオのデータに関する情報を記録します。 (これらの表の詳細については、『管理サポート表』を参照してください。) 画像、音声、ビデオのデータがデータベースに挿入されたり、更新されたり、削除されたりすると、エクステンダーはトリガーを使ってこれらの表を更新します。

## エクステンダーのデータ構造

イメージ、オーディオ、およびビデオ・エクステンダーは、管理サポート表とハンドルを作成および使用して、画像、音声、ビデオのデータを保管し、アクセスします。さらに、イメージ・エクステンダーは、QBIC カタログを作成および使用して、内容に応じて画像にアクセスします。ビデオ・エクステンダーは、さらに索引ファイルとショット・カタログを使って、ビデオのシーンの変化に関する情報にアクセスします。

### 管理サポート表

**管理サポート表** (メタデータ表とも呼ぶ) には、画像、音声、ビデオのオブジェクトに対するユーザーの要求をエクステンダーが処理するために必要な情報が入ります。管理サポート表の情報は、しばしば「メタデータ」とも呼ばれます。

21ページの図7 が示すように、一部の管理サポート表は、エクステンダーで使用されるユーザー表と列を識別します。これらの表は、使用可能になった列にあるオブジェクトの属性情報を入れるために作成された他の管理サポート表を参照します。これらの表では、特定のエクステンダー定義のデータ・タイプに固有の属性だけでなく、エクステンダー・データ・タイプに共通する属性がエクステンダーによって維持されます。たとえば、イメージ・エクステンダーは、画像の幅や、高さ、色数だけでなく、画像、音声、ビデオのオブジェクトに共通する属性についてもその情報を維持します。たとえば、そのオブジェクトをデータベースにインポートした人や、そのオブジェクトを最後に更新した人の名前などです。

管理サポート表には、保管オブジェクトの内容を BLOB 形式で入れることもできます。あるいは、オブジェクトをファイルに保管しておき、管理サポート表によってそのオブジェクトを参照することもできます。たとえば、ビデオ・

クリップは、BLOB として管理サポート表に保管することもできますし、ファイルに保管して表から参照することもできます。

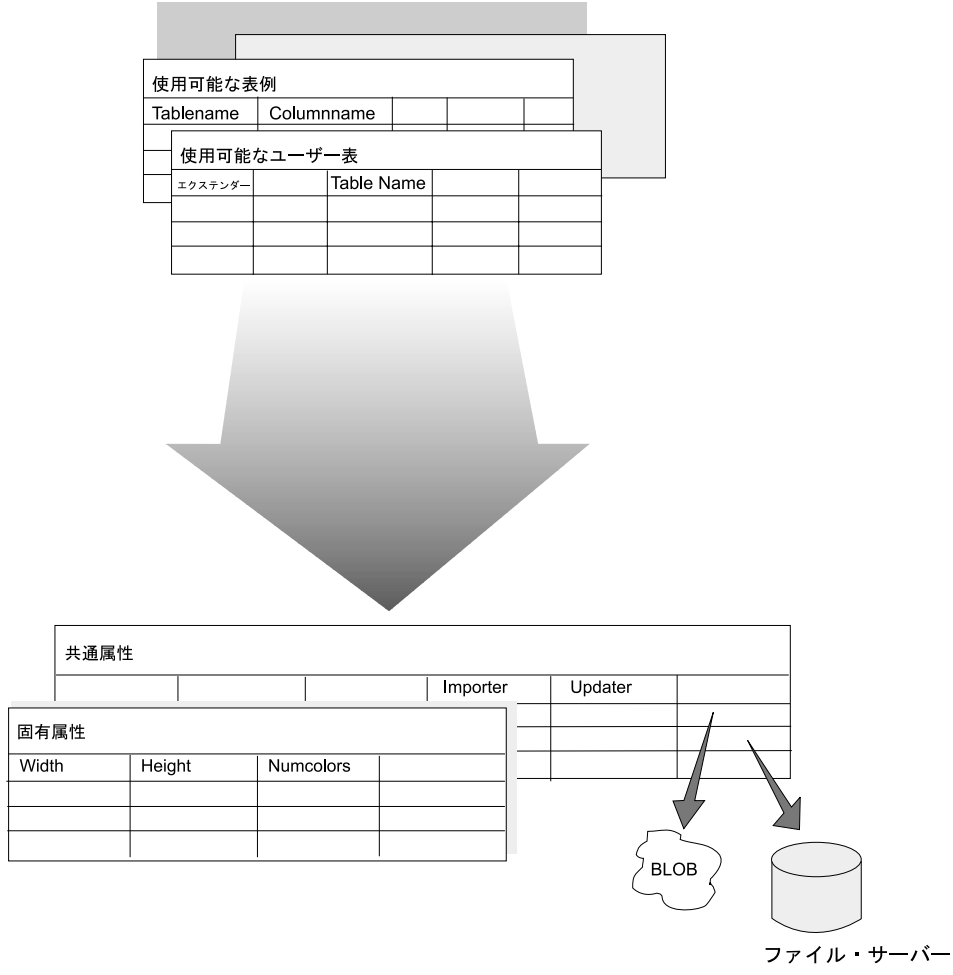



*図 7. 管理サポート表*

## ハンドル

画像や、音声、ビデオのオブジェクトをユーザー表に保管する場合、そのオブジェクトは、実際にはその表に保管されません。その代わりにエクステンダーは、そのオブジェクトを表す**ハンドル**と呼ぶ文字ストリングを作成し、そのハンドルを表に保管します。エクステンダーは、そのオブジェクトを管理サポート表に保管するか、そのオブジェクトの内容がファイルにあれば、ファイル識

別子を管理サポート表に保管します。さらに、そのオブジェクトの属性とハンドルを管理サポート表に保管します。このようにエクステンダーは、ユーザー表に保管されたハンドルと、管理サポート表に保管されたオブジェクト情報をリンクすることができます。 図8 は、ユーザー表の 2 つの画像に対して保管される情報を示しています。

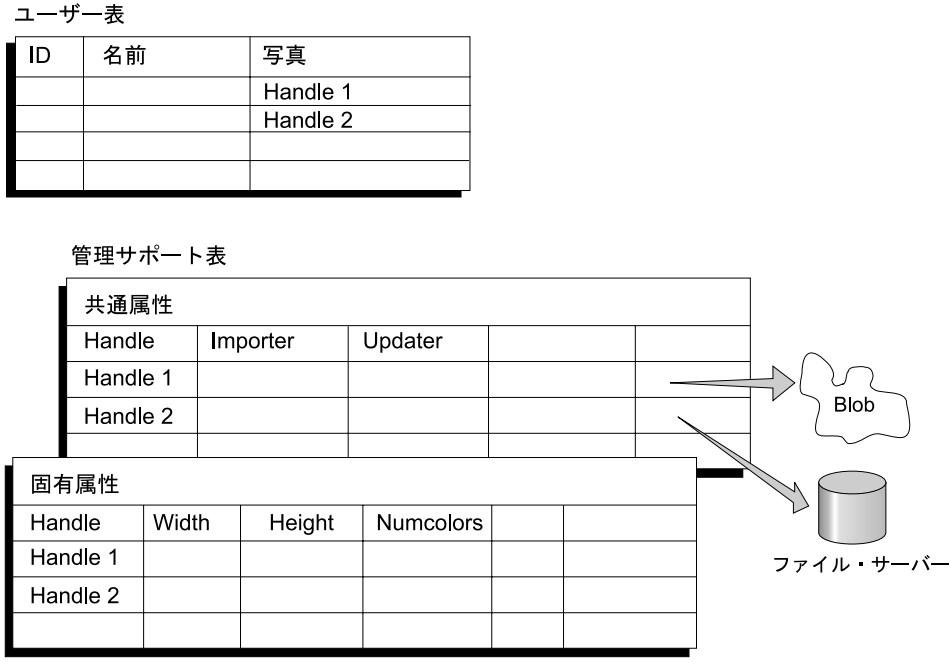



*図 8. ハンドル*

## QBIC カタログ

**QBIC カタログ**は、画像の視覚的な特徴を示すデータが入っているファイルの集合です。イメージ・エクステンダーは、このデータを使って画像を内容に応じて照会します。

ユーザー表内の画像の列 (欄) ごとに QBIC カタログを作成して、それを内容による探索で使用できるようにします。 QBIC カタログを作成する際には、イメージ・エクステンダーによってデータを分析、保管、照会する対象の特徴を指定します。これらの特徴は、カタログを作成した後でも、QBIC カタログへ追加したり、そこから削除したりすることができます。

QBIC カタログには、画像の特徴を表す次のデータを入れることができます。

**平均色** 画像の全ピクセルの色値の合計を画像のピクセル数で除算したもの。 (ピクセルとは、色や輝度を割り当てることのできる画像の最小要素です。) たとえば、画像の 50% が青いピクセルで、残りの 50% が赤いピクセルであれば、画像の平均の色値は紫です。平均色は、優勢な色をもつ画像を探索するときに使用します。画像が優勢な色を持っていれば、その平均色はその優勢な色と類似したものになります。

**ヒストグラム色** 画像の色の分散を 64 色のスペクトルに対して測ります。ヒストグラム色では、64 色のそれぞれについて、その色をもつピクセルの割合を識別します。たとえば、画像のヒストグラム色が 40% の白と 50% の青と 10% の赤であれば、ヒストグラム・スペクトルの他の色をもつピクセルはありません。ヒストグラム色は、さまざまな色をもつ画像を探索するときに使用します。

**定位置色** 画像の特定のエリアにあるピクセルの平均の色値。たとえば、画像の右上に明るい黄色の太陽があれば、このエリアの定位置色は明るい黄色です。定位置色は、特定のエリアに優勢な色がある画像を探索するときに使用します。

**テクスチャー** 画像のきめの粗さ、コントラスト、方向性を示します。きめの粗さは、画像に繰り返し現れる対象の大きさを示します (たとえば、小石か大石か)。コントラストは、画像における輝度の変化を示します (淡い色か、濃い色か)。方向性は、画像の中で方向性が重要であるか (棒塀の場合、縦方向)、ないか (砂の画像など) を示します。テクスチャーは、特定のパターンをもつ画像を探索するときに使用します。

画像を内容で探索できるようにするには、その画像をカタログする必要があります。画像をカタログすると、イメージ・エクステンダーが、その画像の特徴を表す値 (フィーチャー値) を計算することによってその画像を分析し、その値を QBIC カタログに保管します。

画像を内容で探索する際には、1 つまたは複数のフィーチャー (平均色など)、それぞれのフィーチャーを示すソース (例示する画像など)、ターゲットとしてのカタログ・イメージ群をその照会に指定します。イメージ・エクステンダーは、ソースのフィーチャー値を計算し、それをターゲット・イメージのカタログされたフィーチャー値と比較します。次にイメージ・エクステンダーは、ターゲット・イメージのフィーチャー値がソースのそれにいかに似ているかを示す点数を計算します。

これによって、特徴がソースに最も類似した画像をイメージ・エクステンダーから受け取ることができます。イメージ・エクステンダーからは、各画像のハンドルと画像の点数が戻されます。イメージ・エクステンダーから、単一画像の点数だけを受け取ることもできます。

## ビデオ索引

**ビデオ索引**とは、ビデオ・クリップの特定のショットやフレームを検索するときにビデオ・エクステンダーが使用するファイルです。

ビデオ・エクステンダーは、ビデオのシーンの変化を検知することができます。**シーンの変化**とは、ビデオ・クリップの連続する 2 つのシーンの間に顕著な違いがあるところをいいます。たとえば、このような状態は、ビデオの収録中にカメラが視点を変えたときに起こります。シーンが変化してから次に変化するまでの間のフレーム (複数) が**ショット**を構成します。

ビデオ・エクステンダーのシーン検知機能を使えば、ビデオ・クリップのあるショットだけでなく、個々のフレームを見つけることもできます。このためには、エクステンダーは、そのショットまたはフレームの索引情報を必要とします。この索引情報は、**索引ファイル**に保管されます。

## ショット・カタログ

**ショット・カタログ**は、ビデオ・クリップにあるショットに関するデータを保管するために使用します。ショット・カタログはデータベースまたはファイルに保管することができます。

ファイルに保管されるショット・カタログには、ショット関連の次のデータが含まれます。

- ショット・カタログのファイル名
- ビデオ・エクステンダーがショットを検出する方法を制御する値 (たとえば、ショット内のフレームの最小番号)
- ショットの代表フレームとして、どのようなフレームをいくつ保管するかを制御する値
- ショットの番号
- 開始フレームの番号
- 終了フレームの番号
- 代表フレームの番号
- 代表フレームの内容が入っているファイルの名前

ショット・カタログ・ファイル内のデータにアクセスするか、データベース内に保管されているショット・カタログの視点にアクセスすることができます。視点には、ショット関連の次のデータが入った列が含まれます。

- ショット・ハンドル
- ビデオ表の名前
- ビデオ列の名前
- ビデオ・ハンドル
- ビデオ・ファイルの名前
- 開始フレームの番号
- 終了フレームの番号
- 代表フレームの番号
- 代表フレームのデータ
- 注釈

## 区分データベースの概念 (EEE のみ)

DB2 エクステンダーは、DB2 エンタープライズ拡張エディションとともに動作することができ、そのようにして DB2 エンタープライズ拡張エディションの区分データベース・サポートを活用できます。

**区分データベース**とは、複数の独立したマシンに分散しているデータベースのことです。エンド・ユーザーやアプリケーション開発者からは、このデータベースは単一のマシン上にある単一のデータベースのように感じられます。区画化することによって、1 台のマシンでは大き過ぎて処理しきれないデータベースを、アプリケーションが効率的に使えるようになります。

区分データベースは複数の区画で構成されています。各区画は、それ自身の**データベース区画サーバー**によって管理されています。データベース区画サーバーには、データベース・マネージャーとそれによって管理されるデータやシステム資源の集合が組み込まれています。通常は、1 台のマシンに 1 つのデータベース区画サーバーが割り当てられます。とはいえ、1 台のマシンに複数のデータベース区画サーバーを割り当てることも可能です。各データベース区画サーバーはデータベース全体の一部分を保持します。データベース区画サーバーは、**ノード**と呼ばれることがよくあります。

26ページの図9 に示されているように、データベース区画を論理的にグループにまとめ、それに名前を割り当てることができます。そして、まとめられたデータベース区画の各グループを**ノード・グループ**と呼びます。たとえば、ノー

ド・グループを定義することによって、選択したデータベース区画へのアプリケーション照会を制限し、それによってトランザクション時間の速度を向上させることができます。ノード・グループには、1 つのデータベース区画しか入れることができないものと、複数のデータベース区画を入れることができるものとがあります。複数のデータベース区画を入れることができるものを、**複数区画ノード・グループ**と呼びます。複数区画ノード・グループに指定されたすべてのデータベース区画は、みな同じデータベースの中になければなりません。

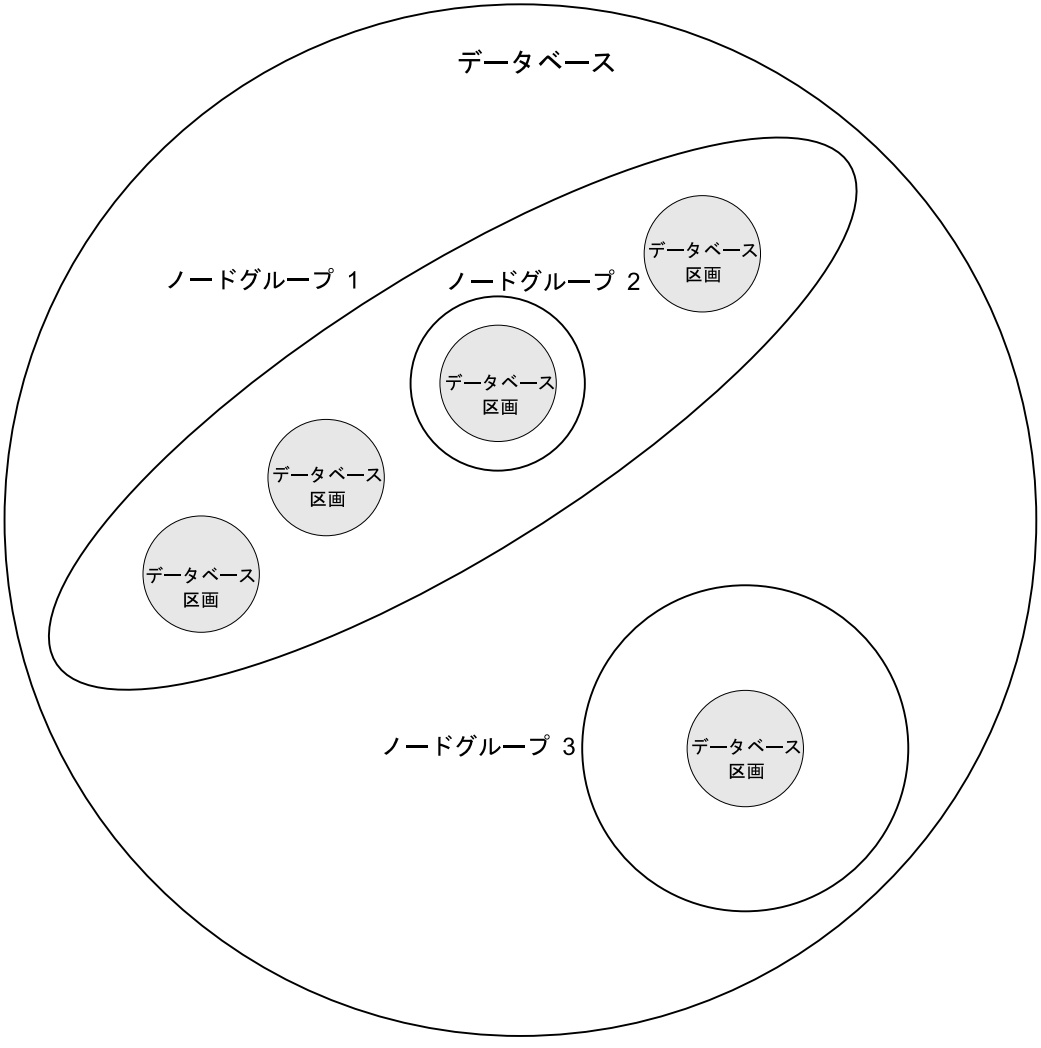



*図 9. データベース内のノード・グループ*

区分データベース・システム内でエクステンダーを使用することには、以下の利点があります。

- 複数の区画にデータを分散させることにより、入出力や処理上のボトルネックを削減できる。
- 複数のマシンを追加し、それらの間でデータを再分散することにより、データベースのサイズを増やすことができる。

### 並列処理

区分データベースでは複数の CPU を使用することができるので、情報の要求に十分対応できます。検索や更新といった要求は自動的に副要求に割り振られ、各マシンのデータベース区画サーバー上で並列に処理されます。

区分データベース・システムの処理能力を示す一例として、単一区分データベース内で 100,000,000 レコードを走査することを考えてみます。この走査を 1 つのデータベース・マネージャーが行うとすれば、単独で 100,000,000 レコードを検索することになります。では、これらのレコードが 20 以上のデータベース区画サーバーに均一に分散されているとしましょう。そうすると、各データベース・マネージャーが走査するレコードの数は 5,000,000 レコードだけになります。これらのデータベース・マネージャーが同じ時間に同じ速度で走査するとすれば、この走査を完了するまでに要する時間は、同じタスクを 1 つの区画システムで処理するのに要する時間の約 5 % です。

### スケーラビリティー

データベースのサイズが増加するにつれて、データベース区画サーバーをデータベース・システムに追加することによりパフォーマンスを改善することができますが、このプロセスのことをデータベース・システムの**スケーリング**といいます。

データベースをスケーリングすると、データベース区画サーバーが追加され、それにともなってデータベース・システム内の既存の各データベースにデータベース区画が 1 つ追加されます。その後、新しいデータベース区画を、データベースの既存のノード・グループに割り当てることができます。最後に、このノード・グループ内でデータを分散し直して、新しいデータベース区画を使用します。

### 区分データベース環境での DB2 エクステンダーの使用

区分データベース環境で DB2 エクステンダーを使用することによって、LOB の操作をサポートする面で際立っている諸機能を活用できます。 1 つのデータ

ベースを多数のマシンに分散できるので、LOB のリポジトリー (その 1 つの長さが最大 2 GB にまでなることがある) は、たとえそれが大きいとしても 1 つのデータベースに保管できます。

さらに、DB2 エクステンダーは、DB2 エンタープライズ拡張エディションに管理されながら、SQL 操作の並列処理に参加します。 DB2 エンタープライズ拡張エディションが照会を並列処理で実行すると、その照会の DB2 エクステンダー UDF も各データベース区画で並列に実行されます。

## 機密保護および回復

画像、音声、およびビデオ・オブジェクト (DB2 データベース内で BLOB として保管されているもの) には、従来の数値データおよび文字データの場合と同じ機密保護および回復保護機能が与えられています。メタデータ表にあるオブジェクト用に保管された情報も同様です。ユーザーには、オブジェクトの選択、挿入、または更新を行うための特権が必要です。

ユーザーは UDF を発行して、ユーザー表からオブジェクトを選択、挿入、更新、または削除します。要求された操作を行うには、UDF はオブジェクトの属性情報を保持する管理サポート表にアクセス (必要であれば更新) できなければなりません。ユーザー表に対する適切な特権をユーザーが持っているならば、エクステンダーは UDF による管理サポート表に対する上記の操作を許可します。

エクステンダー関連の管理操作によっては、DBADM 権限が必要になることがあります。 DB2 エクステンダーで必要となる権限については、285ページの『第16章 アプリケーション・プログラミング・インターフェース』を参照してください。 DB2 エクステンダー管理コマンドで必要となる権限については、509ページの『第17章 クライアント側の管理コマンド』を参照してください。

画像や、音声、ビデオの内容が、そのデータベースから参照されるファイルに保管される場合には、そのオブジェクトのメタデータも DB2 によって保護されます。そのファイルは、PUBLIC つまりすべてのユーザーによって読み取り可能なディレクトリーになければなりません。

BLOB とメタデータは、DB2 の他のデータと同様に、バックアップや回復を行うことができます。ファイルに保管されたオブジェクトの内容は、非 DB2 ツールを使ってバックアップや回復を行うことができます。さらに、QBIC カタログとビデオ索引も、非 DB ツールを使ってバックアップや回復を行うことが

できます。 QBIC カタログのバックアップについては、147 ページを参照してください。ビデオ索引のバックアップについては、197 ページを参照してください。

# 第3章 エクステンダーの機能

DB2 エクステンダーは、画像、音声、およびビデオのデータの処理要求に幅広く応じます。エクステンダーがどのように動くのかを理解するには、ご自分で試しに使ってみるのが最善です。この章では、イメージ・エクステンダーとオーディオ・エクステンダーを扱った 1 つのシナリオについて説明します。このシナリオでは、ユーザーが行う操作と、それに対するエクステンダーの応答について説明します。

## エクステンダーのシナリオ

ある企業の人事部門が、従業員の写真を含む人事データベースを (DB2 (AIX 版) で) 作成しようとしています。

**写真入りのデータベース:** 図10 が示すように、データベースの従業員表には、各従業員の番号と名前が写真とともに保管されています。

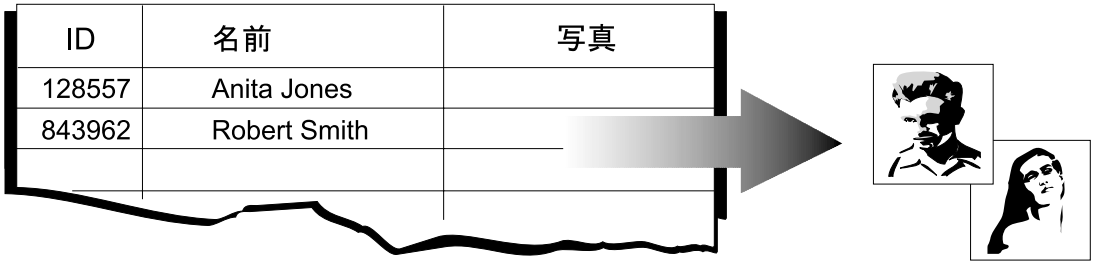



*図 10. 従業員表*

人事データベースの画像処理ができるように、システム管理者 (つまり、SYSADM 権限をもつ人) が、まずエクステンダー・サービスを開始します。システム管理者は、次にデータベースを作成し、それをイメージ・エクステンダーから使用できるようにします。

データベース管理者 (DBA) または同等な権限をもつ人が、従業員表を作成し、その表と従業員写真列をイメージ・エクステンダーから使用できるようにします。

**音声入りのデータベース:** 人事データベースと従業員表を画像処理用に準備した後、人事部門は各従業員の音声記録を表示に追加することにします。これを図11 に示します。

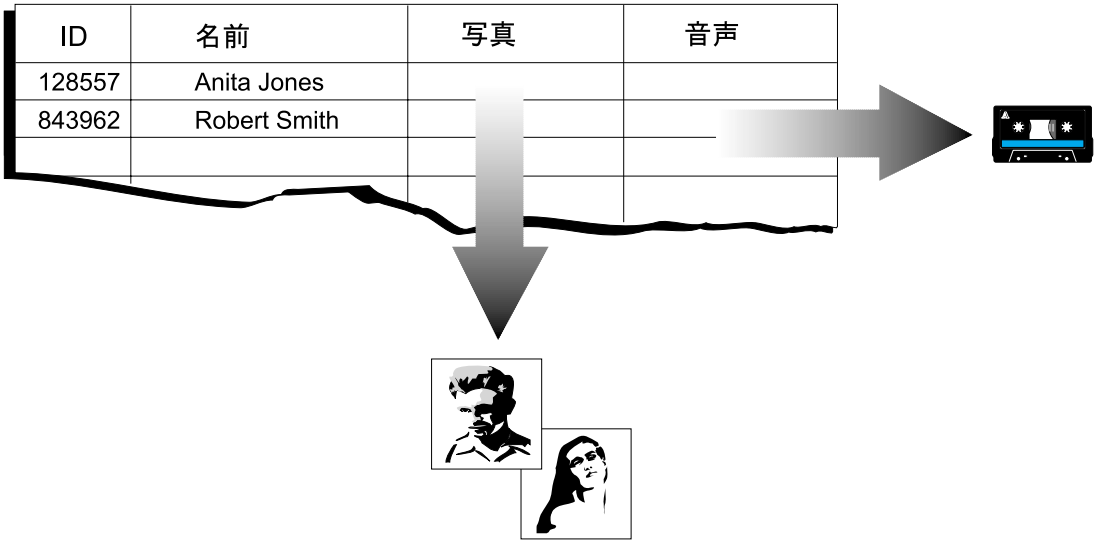



*図 11. 音声の列が追加された従業員表*

システム管理者は新しい列を追加して表を変更し、そのデータベースと、表、列をオーディオ・エクステンダーで使用できるようにします。

その後で、人事部門のユーザーは、その表にデータを追加し、そこからデータを選択および表示し、更新し、削除します。

## エクステンダー・サービスの開始

エクステンダーは、その操作の一部としてサーバーのサービスを使用します。これらのサービスが、サーバーの通常の「開始」操作の機能として使用できるようになっていない場合は、システム管理者がそれらを開始します。

**システム管理者が行うこと:** システム管理者はエクステンダー・インスタンス所有者として AIX サーバーにログオンします。次にシステム管理者は、サーバーで次のコマンドを実行します。

```
DMBSTART
```

**その結果:** サーバー上のエクステンダー・インスタンスに対してエクステンダー・サービスが開始されます。 DMBSTART コマンドは、DB2 インスタンスが開始されていなければ、それも開始します。

## データベースの準備

システム管理者は、人事データベースを作成し、イメージ・エクステンダーによって使用できるようにします。

**システム管理者が行うこと:** システム管理者は、人事データベースを、DB2 (AIX 版) で、次の SQL ステートメントを使って作成します。

```
CREATE DATABASE personnl           /*name of the database*/
       ON /persdb                  /*name of the database directory*/
       WITH "Personnel database"   /*comment*/
```

システム管理者は、そのデータベースに接続し、そのデータベースをイメージ・エクステンダーから使用できるようにします。システム管理者は、db2ext コマンド行プロセッサーから次のコマンドを実行します。

```
CONNECT TO personnl
ENABLE DATABASE FOR DB2IMAGE
```

**その結果:** ENABLE DATABASE コマンドへの応答として、イメージ・エクステンダーは次のことを行います。

- 画像オブジェクトに対して DB2IMAGE というユーザー定義タイプを作成します。
- 画像オブジェクトに対して管理サポート表を作成します。
- 画像オブジェクトに対してユーザー定義関数を作成します。 UDF のリストが 表1にあります。

*表 1. イメージ・エクステンダーによって作成されるユーザー定義関数*

| UDF 名 | 説明 |
|---|---|
| Comment | ユーザーの注釈を入手または更新する。 |
| Content | 画像の内容を入手または更新する。 |
| DB2Image | 画像の内容を保管する。 |
| Filename | 画像が入っているファイルの名前を入手する。 |
| Format | イメージ形式 (たとえば、GIF) を入手する。 |
| Height | 画像の高さをピクセル単位で入手する。 |
| Importer | 画像をインポートした人のユーザー ID を入手する。 |
| ImportTime | 画像がインポートされたときのタイム・スタンプを入手する。 |

*表 1. イメージ・エクステンダーによって作成されるユーザー定義関数 (続き)*

| UDF 名 | 説明 |
|---|---|
| NumColors | 画像で使用されている色の数を入手する。 |
| QbScoreFromName | 画像の類似性得点を入手する (名前付き照会を使用する)。 |
| QbScoreFromStr | 画像の類似性得点を入手する (照会ストリングを使用する)。 |
| QbScoreTBFromName | 画像の列に関する類似性得点の表を入手する (名前付き照会を使用する)。 |
| QbScoreTBFromStr | 画像の列に関する類似性得点の表を入手する (照会ストリングを使用する)。 |
| Replace | 画像の内容とユーザー注釈を更新する。 |
| Size | 画像のサイズをバイト単位で入手する。 |
| Thumbnail | 縮小 (サムネール) 画像を入手する。 |
| Updater | 画像を更新した人のユーザー ID を入手する。 |
| UpdateTime | 画像が更新されたときのタイム・スタンプを入手する。 |
| Width | 画像の幅をピクセル単位で入手する。 |

## 表の準備

DBA は従業員表を作成し、その表と写真の列をイメージ・エクステンダーから使用できるようにします。

**DBA の行うこと:** 便宜上、DBA は mmdbsys スキーマを次の SQL ステートメントを使って、現行関数パスに追加します。

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

こうすることで、UDT や UDF の名前を指定するとき、それらの接頭部としてスキーマ名 (mmdbsys) を付ける必要がありません。 (mmdbsys スキーマを関数パスの最初のスキーマにする必要はありません。) UDT と UDF の名前については、18ページの『UDF 名と UDT 名』を参照してください。

DBA は従業員表を作成します。 DBA は DB2 コマンド行プロセッサーを使用して、次の SQL ステートメントを発行します。

```
CREATE TABLE   employee            /*name of the table*/
       (id      CHAR(6)            /*employee identification*/
       name    VARCHAR(40)         /*employee name*/
       picture DB2IMAGE)           /*employee picture*/
```

次に DBA は、db2ext コマンド行プロセッサーから次のコマンドを実行します。

```
ENABLE TABLE employee FOR DB2IMAGE
ENABLE COLUMN employee picture FOR DB2IMAGE
```

**その結果:** ENABLE TABLE コマンドへの応答として、イメージ・エクステンダーは次のことを行います。

- 使用する従業員表を識別します。
- 使用可能な列内の画像オブジェクトの属性情報を保持する管理サポート表を作成します。

ENABLE COLUMN コマンドへの応答として、イメージ・エクステンダーは次のことを行います。

- 使用するピクチャー列を識別します。
- トリガーを作成します。従業員表に対して挿入や、更新、削除の操作が行われると、これらのトリガーによって、さまざまな管理サポート表が更新されます。

## 表の変更

DBA は音声の列を従業員表に追加し、それをオーディオ・エクステンダーから使用できるようにします。

**DBA が行うこと:** DBA は db2ext コマンド行プロセッサーを使って、人事データベースがオーディオ・エクステンダーから使用できるようにします。

```
ENABLE DATABASE FOR DB2AUDIO
```

それから、DBA は次の SQL ステートメントを発行して、従業員表を変更します。 DBA は、DB2 コマンド行プロセッサーを使ってこの SQL ステートメントを発行します。

```
ALTER TABLE employee              /*name of the table*/
      ADD voice DB2AUDIO          /*employee audio recording*/
```

DBA は、db2ext コマンド行プロセッサーを使って、従業員表と音声列がオーディオ・エクステンダーから使用できるようにします。

```
ENABLE TABLE employee FOR DB2AUDIO
ENABLE COLUMN employee voice FOR DB2AUDIO
```

**その結果:** ENABLE DATABASE コマンドへの応答として、オーディオ・エクステンダーは次のことを行います。

- 音声オブジェクト用の DB2AUDIO というユーザー定義タイプを作成します。
- 音声オブジェクト用の管理サポート表を作成します。
- 音声オブジェクト用のユーザー定義関数を作成します。 UDF のリストは、表2 にあります。

*表 2. オーディオ・エクステンダーによって作成されるユーザー定義関数*

| UDF 名 | 説明 |
|---|---|
| AlignValue | 音声のサンプルごとのバイト数の値を入手する。 |
| BitsPerSample | 音声を表すのに使用されるビット数を入手する。 |
| BytesPerSec | 1 秒あたりの音声に対する平均バイト数を入手する。 |
| Comment | ユーザーの注釈を入手または更新する。 |
| Content | 音声の内容を入手または更新する。 |
| DB2Audio | 音声の内容を保管する。 |
| Duration | 音声の再生時間を入手する。 |
| Filename | 音声が入っているファイルの名前を入手する。 |
| FindInstrument | 特定の楽器が録音されている音声トラックの番号を入手する。 |
| FindTrackName | 音声録音における指定のトラックのトラック番号を入手する。 |
| Format | 音声形式を入手する。 |
| GetInstruments | 音声に録音されている楽器の名前を入手する。 |
| GetTrackNames | 音声のトラック名を入手する。 |
| Importer | 音声をインポートした人のユーザー ID を入手する。 |
| ImportTime | 音声がインポートされたときのタイム・スタンプを入手する。 |
| NumAudioTracks | 音声の録音されているトラックの数を入手する。 |
| NumChannels | 音声チャネルの数を入手する。 |
| Replace | 音声録音の内容とユーザー注釈を更新する。 |
| SamplingRate | 音声のサンプリング率を入手する。 |
| Size | 音声のサイズをバイト単位で入手する。 |
| TicksPerQNote | 音声の録音で使用された、四分音符あたりのクロック・ティックの数を入手する。 |
| TicksPerSec | 音声の録音で使用された、秒あたりのクロック・ティックの数を入手する。 |
| Updater | 音声を更新した人のユーザー ID を入手する。 |

*表 2. オーディオ・エクステンダーによって作成されるユーザー定義関数 (続き)*

| UDF 名 | 説明 |
|---|---|
| UpdateTime | 音声が更新されたときのタイム・スタンプを入手する。 |

ENABLE TABLE コマンドへの応答として、オーディオ・エクステンダーは次のことを行います。

- 使用する従業員表を識別します。
- 使用可能な列内の音声オブジェクトの属性情報を保持する管理サポート表を作成します。

ENABLE COLUMN コマンドへの応答として、オーディオ・エクステンダーは次のことを行います。

- 使用する音声列を識別します。
- トリガーを作成します。従業員表に対して挿入や、更新、削除の操作が行われると、これらのトリガーによって、さまざまな管理サポート表が更新されます。

## 表へのデータの挿入

ユーザーが Anita Jones のレコードを従業員表に挿入します。レコードには、Anita の識別番号 (128557) と、名前、写真、音声録音が入っています。ソース・イメージと音声自体は、サーバーのファイルにあります。画像は BLOB として表に保管されており、音声の内容はサーバーのファイルに残ります (表の項目はサーバー・ファイルを参照します)。

**ユーザーが行うこと:** ユーザーは、38ページの図12 に示すステートメントを組み込んだアプリケーション・プログラムを使用してレコードを従業員表に挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvInt_Stor;
long hvExt_Stor;
EXEC SQL END DECLARE SECTION;

hvInt_Stor = MMDB_STORAGE_TYPE_INTERNAL;
hvExt_Stor = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
           CURRENT SERVER,                 /*database server name in*/
                                           /CURRENT SERVER register*/
           '/employee/images/ajones.bmp'   /*image source file*/
           'ASIS',                         /*keep the image format*/
           :hvInt_Stor,                    /*store image in DB as BLOB*/
           'Anita''s picture'),            /*comment*/
        DB2AUDIO(                          /*Audio Extender UDF*/
           CURRENT SERVER,                 /*database server name in*/
                                           /*CURRENT SERVER register*/
           '/employee/sounds/ajones.wav',  /*audio source file*/
           'WAVE',                         /* audio format */
           :hvExt_Stor,                    /*retain content in server file*/
           'Anita''s voice')               /*comment*/
      );
```

*図 12. 表へのデータの挿入*

**その結果:** INSERT ステートメントの DB2Image UDF に応答して、イメージ・エクステンダーは以下のことを行います。

- ソース・イメージ・ファイルのヘッダーから画像の属性を読み込みます。たとえば、その画像の高さや、幅、色数などです。
- その画像固有のハンドルを作成し、次のものを管理サポート表に記録します。
  - 画像のハンドル
  - タイム・スタンプ
  - イメージ・サイズ (バイト単位で)
  - 『Anita's picture』 という注釈
  - 画像の内容

  イメージ・ソースは、ajones.bmp という名前のサーバー・ファイルにあります。ファイルの内容は、管理サポート表のレコードに BLOB として挿入されます。保管されるイメージの形式はソース・イメージと同じです (つまり、形式の変換は行われません)。

- レコードを管理サポート表に保管します。レコードには、画像の色数などの画像固有の属性だけでなく、サムネール・サイズの画像も入ります。

INSERT ステートメントの DB2Audio UDF への応答として、オーディオ・エクステンダーは以下のことを行います。

- オーディオ・ファイルのヘッダーから音声の属性 (たとえば、音声のトラックやチャネルの数などを) 読み込みます。
- その音声固有のハンドルを作成します。
- レコードを管理サポート表に保管します。このレコードには、次のものが含まれています。
  - その音声のハンドル
  - タイム・スタンプ
  - 音声のサイズ (バイト単位で)
  - 『Anita's voice』 という注釈

  音声の内容は ajones.wav というサーバー・ファイルにありますが、管理サポート表のレコードによりそのファイルを参照されます。
- レコードを別の管理サポート表に保管します。レコードには、音声のサンプリング率などの音声特有の属性が入ります。

トリガーは、画像や音声の属性データをいろいろな管理サポート表に挿入します。

## 表からのデータの選択

Robert Smith の画像と音声録音が従業員表にいつ保管されたのかについて、ユーザーが情報を検索します。

**ユーザーが行うこと:** ユーザーは、40ページの図13 に示す SQL ステートメントを組み込んだアプリケーション・プログラムを使用して、情報を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
char[255]  hvImg_Time;
char[255]  hvAud_Time;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT IMPORTTIME(PICTURE),              /*when image was stored*/
       IMPORTTIME(VOICE)                          /*when audio was stored*/
       INTO :hvImg_Time, :hvAud_Time
       FROM EMPLOYEE
       WHERE NAME='Robert Smith';
```

*図 13. 表からデータの選択*

**その結果:** ピクチャー列に対する ImportTime UDF への応答として、イメージ・エクステンダーは、画像が保管された日付と時刻を示すタイム・スタンプを戻します。音声列に対する ImportTime UDF への応答として、オーディオ・エクステンダーは、その音声録音が保管された日付と時刻を示すタイム・スタンプを戻します。

## オブジェクトの表示と再生

ユーザーは、Robert Smith の画像を表示し、Robert Smith の音声録音を再生します。画像は従業員表に BLOB として保管されており、音声録音の内容はサーバー・ファイルにあります。

**ユーザーが行うこと:** ユーザーは、41ページの図14 に示す SQL ステートメントを組み込んだアプリケーション・プログラムを使用して、画像を表示し、声の録音を再生します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl [251];
char hvAud_hdl [251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT PICTURE,                /*Get image handle*/
                VOICE                   /*Get audio handle*/
       INTO :hvImg_hdl, :hvAud_hdl
       FROM EMPLOYEE
       WHERE NAME='Robert Smith';

rc=DBiBrowse(
        NULL,                           /*Use default image browser*/
        MMDB_PLAY_HANDLE,               /*Use handle*/
        hvImg_hdl,                      /*Image handle*/
        MMDB_PLAY_NO_WAIT);             /*Run browser independently*/

rc=DBaPlay(
        NULL,                           /*Use default audio player*/
        MMDB_PLAY_HANDLE,               /*Use handle*/
        hvAud_hdl,                      /*Audio handle*/
        MMDB_PLAY_WAIT);                /*Wait for player to end*/
                                        /*before continuing*/
```

*図 14. オブジェクトの表示と再生*

**その結果:** DB2 は Robert Smith の画像と録音された声のハンドルを検索します。次に DBiBrowse API への応答として、イメージ・エクステンダーは、検索した画像ハンドルに関連するイメージ内容を入手します。イメージ・エクステンダーは、データベースからイメージ内容を検索し、画像ブラウザーが表示できるように、それを一時クライアント・ファイルに入れます。 NULL パラメーターを指定すると、ユーザーのシステムの省略時画像ブラウザーが使用されます。ブラウザーは呼び出し側のプログラムとは独立して実行されます。つまり、呼び出し側のプログラムは、画像ブラウザーが終了するのを待たずに次の処理へ進みます。

DBaPlay API への応答として、オーディオ・エクステンダーは、検索した音声ハンドルに対応する音声のファイル名を入手し、それをオーディオ・プレーヤーに渡します。 NULL パラメーターを指定すると、ユーザーのシステムの省略時のオーディオ・プレーヤーが使用されます。呼び出し側のプログラムは、ユーザーがオーディオ・プレーヤーを終了してから次へ進みます。

## 表データの更新

Anita Jones は、従業員表の自分の写真を新しいものと取り替えます。新しい写真の内容はサーバー・ファイルにあります。

**ユーザーが行うこと:** ユーザーは、図15 に示す SQL ステートメントを組み込んだアプリケーション・プログラムを使用して、従業員表の写真を取り替えます。 SQL ステートメントを組み込んだアプリケーション・プログラムを使用して、従業員表の写真を取り替えます。

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvComment [16385];
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

strcpy(hvComment, "Picture taken at Anita's promotion");
hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
       SET PICTURE=REPLACE(
              PICTURE,                      /*image handle*/
              '/myimages/newone.bmp',       /*source image content*/
              'BMP',                        /*source format*/
              :hvStorageType,               /*store image in table as BLOB*/
              :hvComment)                   /*replace comment*/
       WHERE NAME='Anita Jones';
```

*図 15. 表データの更新*

**その結果:** UPDATE ステートメントの置換 UDF への応答として、イメージ・エクステンダーは新しい画像の属性を読み取ります。イメージ・エクステンダーは新しい画像の属性を使用して、古い画像用に管理サポート表に保管されている属性を更新します。イメージ・ソースは、newone.bmp という名前のサーバー・ファイルにあります。そのファイルの内容が管理サポート表のレコードに BLOB として挿入され、古い画像の BLOB 内容が置き換えられます。

トリガーは、いろいろな管理サポート表の画像属性データを置き換えます。

## 表からのデータの削除

ユーザーは、従業員表から Anita Jones のレコードを削除します。

**ユーザーが行うこと:** ユーザーは次のような SQL ステートメントを組み込んだアプリケーション・プログラムを使用して、従業員表からレコードを削除します。

```
DELETE FROM EMPLOYEE
       WHERE NAME='Anita Jones';
```

**その結果:** トリガーによって、各種の管理サポート表にある Anita Jones の項目が削除されます。

# 第2部 画像、音声、ビデオのデータ管理

# 第4章 管理の概要

この章では、DB2 エクステンダーを使ってアプリケーションを作成するときに必要な管理タスクについて概要を説明します。

DB2 エクステンダーのほとんどの管理タスクは、次の 2 つの方法で行うことができます。

- 管理アプリケーション・プログラミング・インターフェース (API)。この場合には、DB2 エクステンダー API を C 言語プログラムに組み込みます。これらの API の参照情報については、285ページの『第16章 アプリケーション・プログラミング・インターフェース』を参照してください。
- 管理コマンド。管理コマンドは、db2ext コマンド行プロセッサーに実行依頼することができます。これらのコマンドを DB2 コマンド行から実行することはできません。管理コマンドの入力方法および追加の参照情報については、509ページの『第17章 クライアント側の管理コマンド』を参照してください。

## DB2 エクステンダーで行う管理タスク

管理タスクには次の 5 つの区分があります。

- エクステンダー・サービスの管理。 DB2 エクステンダーは、DB2 の上位にあるそれぞれ独自のサーバーで稼働します。アプリケーションからエクステンダーのデータを使用するためには、システム管理者がエクステンダー・サービスをすでに開始しており、ユーザーがそのエクステンダー・データが入っているデータベースに接続されていなければなりません。
- エクステンダー・データのためのデータ・オブジェクトの準備。データベース、表、および列を作成して、エクステンダー・データを入れるためには、それらを使用可能にする必要があります。データ・オブジェクトを使用可能にすると、エクステンダーは管理サポート表 (メタデータ表とも呼ばれる) の作成と保守を行うことにより、エクステンダー・データを管理します。
- **EEE のみ。** 区画環境でのエクステンダー・データの再分散。区分データベースで区画を追加または削除した場合、データを再分散して新しいノード構成を利用できます。
- データ・オブジェクトとメディア・ファイルの追跡。 DB2 エクステンダーを使用するアプリケーションをデバッグする際には、エクステンダー・デー

タに対して、どのデータ・オブジェクトが使用可能になっているかを知っていると便利です。また、ユーザー表と外部メディア・ファイルの関連を知っていると役に立ちます。

- 管理サポート表のクリーンアップ。 DB2 エクステンダーを使用していると、古い項目が次第に管理サポート表にたまります。古いメタデータを削除すれば、パフォーマンスを改善し、記憶スペースを再利用することができます。

表3 は、エクステンダー・データの管理で必要なすべてのタスクをリストしたものです。表には、各タスクを実行するにはどのツールを使用するか、また、詳しい情報はどこを参照すればよいかが示されています。

**エクステンダー API** の列で、各 API ステートメントの 3 つ目の文字が `x` で表示されています。この文字は、使用するエクステンダーによって次のように変わります。

| 文字 | エクステンダー |
|---|---|
| a | オーディオ |
| i | イメージ |
| v | ビデオ |

たとえば、画像データの表を使用可能にする API は DB**i**EnableTable、音声の表を使用可能にする API は DB**a**EnableTable、ビデオの表を使用可能にする API は DB**v**EnableTable です。エクステンダー API 列内の `No` という値は、そのタスクに対するエクステンダー API が存在しないことを示します。エクステンダー・コマンド列内の `No` という値は、そのタスクに対するエクステンダー・コマンドが存在しないことを示します。

**QBIC では追加の管理が必要です:** イメージ・エクステンダーのイメージ内容照会 (QBIC) 機能を使用する場合には、QBIC カタログの作成などの管理タスクがさらに必要です。これらのタスクについては、143ページの『第13章 イメージの内容による照会』を参照してください。

*表 3. DB2 エクステンダーの管理タスクと管理機能*

| タスク | エクステンダー API | エクステンダー・コマンド | 参照 |
|---|---|---|---|
| **エクステンダー・サービスの管理** | | | |
| エクステンダー・サービスを開始する | No | DMBSTART | 51 ページ |

*表 3. DB2 エクステンダーの管理タスクと管理機能 (続き)*

| タスク | エクステンダー API | エクステンダー・コマンド | 参照 |
|---|---|---|---|
| エクステンダー・サービスの状況を入手する | No | DMBSTAT | 54 ページ |
| エクステンダー・サービスを停止する | No | DMBSTOP | 53 ページ |
| データベースに接続する | No | CONNECT | 51 ページ |
| 使用するデータベースのエクステンダー・サービスを開始する | No | START SERVER | 53 ページ |
| 使用するデータベースのエクステンダー・サービスの状況を入手する | No | GET SERVER STATUS | 54 ページ |
| 使用するデータベースのエクステンダー・サービスを停止する | No | STOP SERVER | 53 ページ |
| エクステンダー・インスタンスを作成し管理する | No | DMBICRT、DMBILIST、DMBIDROP、DMBIMIGR | 55 ページ |
| **マルチメディア・データのためのデータ・オブジェクトの準備** | | | |
| データベースを使用可能にする | DBxEnableDatabase | ENABLE DATABASE | 59 ページ |
| データベースを使用不能にする | DBxDisableDatabase | DISABLE DATABASE | 67 ページ |
| 表を使用可能にする | DBxEnableTable | ENABLE TABLE | 63 ページ |
| 表を使用不能にする | DBxDisableTable | DISABLE TABLE | 67 ページ |
| 列を使用可能にする | DBxEnableColumn | ENABLE COLUMN | 66 ページ |
| 列を使用不能にする | DBxDisableColumn | DISABLE COLUMN | 67 ページ |
| **区画環境でのエクステンダー・データの再分散 (EEE のみ)** | | | |
| 新しいノード・グループ構成に基づいたエクステンダー・データの再分散 | DMBRedistribute | REDISTRIBUTE NODEGROUP | 69 ページ |
| **データ・オブジェクトとメディア・ファイルの追跡** | | | |
| データベースが使用可能になっているかどうかを調べる | DBxIsDatabaseEnabled | GET EXTENDER STATUS | 71 ページ |
| 表が使用可能になっているかどうかを調べる | DBxIsTableEnabled | GET EXTENDER STATUS | 71 ページ |

*表 3. DB2 エクステンダーの管理タスクと管理機能 (続き)*

| タスク | エクステンダー API | エクステンダー・コマンド | 参照 |
|---|---|---|---|
| 列が使用可能になっているかどうかを調べる | DBxIsColumnEnabled | GET EXTENDER STATUS | 71 ページ |
| 修飾子が現行ユーザー ID である表の中のファイルを参照する表項目を見つける | DBxIsFileReferenced | No | 72 ページ |
| 特定の修飾子をもつすべての表、またはデータベースのすべての表にあるファイルを参照する表項目を見つける | DBxAdminIsFileReferenced | No | 72 ページ |
| 修飾子が現行ユーザー ID である表の中の表項目によって参照されるファイルを見つける | DBxGetReferencedFiles | GET REFERENCED FILES | 73 ページ |
| 特定の修飾子をもつすべての表、またはデータベースのすべての表にある表項目によって参照されるファイルを見つける | DBxAdminGetReferencedFiles | GET REFERENCED FILES | 73 ページ |
| 修飾子が現行ユーザー ID であるすべての表の中の表項目によって参照されるファイルのうち、アクセス不能なものを見つける | DBxGetInaccessibleFiles | GET INACCESSIBLE FILES | 74 ページ |
| 特定の修飾子をもつすべての表、またはデータベースのすべての表にある表項目によって参照されるファイルのうち、アクセス不能なものを見つける | DBxAdminGetInaccessibleFiles | GET INACCESSIBLE FILES | 74 ページ |
| **管理サポート (メタデータ) 表のクリーンアップ** | | | |
| 特定のユーザー表のメタデータ表、または修飾子が現行ユーザー ID であるすべてのユーザー表のメタデータ表を、すべてクリーンアップする | DBxReorgMetadata | REORG | 77 ページ |

*表 3. DB2 エクステンダーの管理タスクと管理機能 (続き)*

| タスク | エクステンダー API | エクステンダー・コマンド | 参照 |
|---|---|---|---|
| 特定の修飾子をもつすべてのユーザー表のメタデータ表、またはデータベースにあるすべてのユーザー表のメタデータ表を、すべてクリーンアップする | DBxAdminReorgMetadata | REORG | 77 ページ |

**管理タスクの順序:** 次のリストは、エクステンダーを初めて使用するときに行う管理タスクの要約を順番に並べたものです。タスクによっては DB2 コマンドまたはステートメントを使用して実行します。その他のタスクは DB2 エクステンダーを使用して実行します。この順序ではご使用の DB2 システムがすでに稼働中であると想定しています。

必須タスク

1. エクステンダー・サービスを開始する。
2. データベースを作成する (DB2 を使って)。
3. データベース・サーバーに接続する。
4. データベースを使用可能にする。
5. 表と列を作成する (DB2 を使って)。
6. データベースの表を使用可能にする。
7. 表の列を使用可能にする。

任意選択タスク

1. データ・オブジェクトとメディア・ファイルを追跡する。
2. 関数パスを設定する (DB2 を使って)。
3. 管理サポート表をクリーンアップする。

**例:** 次の 5 つの章のほとんどの例では、システム管理者 (SYSADM) またはデータベース管理者 (DBA) がこれらのタスクを行うものとします。いくつかのタスクには、DBA 権限も SYSADM 権限も必要ありません。

これらの例では、DBA が現行関数パスに MMDBSYS スキーマを追加したものとします。これによって、DBA は、UDT の名前を指定するとき、接頭部としてスキーマ名 MMDBSYS を指定する必要はありません。 UDT 名については、さらに 18ページの『UDF 名と UDT 名』を参照してください。

次の API 例の多くでは、エクステンダーとともに提供されるサンプルのアプリケーション・コードを使用します。サンプル・コードは、クライアントの SAMPLES サブディレクトリーにあります。

# 第5章 エクステンダー・サーバーの管理

DB2 エクステンダーは DB2 のクライアント / サーバー環境で稼働します。この環境は、データベース・サーバーと 1 つ以上のリモート・データベース・クライアントで構成されます。 DB2 エクステンダー・サービスはサーバーで実行されます。このサービスにアクセスする場合には、まずそのサービスを開始しておく必要があります。

環境が設定されていれば、クライアントからエクステンダー・サービスを停止したり、再始動したりすることができます。エクステンダーの状況は、クライアントからでもサーバーからでも入手できます。

**EEE のみ:** 複数区画環境では、データベース区画を追加または削除することもできます。

## エクステンダー環境の設定

エクステンダー・サービスを開始するには、サーバーのオペレーティング・システムのコマンド行から DMBSTART コマンドを入力します。

```
dmbstart
```

DMBSTART コマンドは、エクステンダー・データを収容できるようになっているすべてのデータベースで、エクステンダー・サービスを開始します。このコマンドは、DB2 が稼働していなければ、それも開始します。このコマンドを実行するには、SYSADM、 SYSCTRL、SYSMAINT の権限のどれかが必要です。 AIX の場合は、エクステンダー・インスタンスの所有者としてログオンされていなければなりません。

この時点でご使用の C 言語のアプリケーションは、データベースとの接続を確立していれば、API を通してエクステンダー・サービスにアクセスすることができます。同様に、db2ext コマンド行を使用する場合にも、使用するデータベースに接続しなければなりません。 db2ext コマンド行を使用する場合には、DB2 コマンド行によって使用するものとは別の接続が必要です。

クライアントで db2ext コマンド行プロセッサーをオープンし、DB2 エクステンダーの CONNECT コマンドを実行してください。次の例では、このコマンドによって、PERSONNL データベースに接続します。さらに、パスワード ANPASS を使って、修飾子が ANITAS の表をアクセスします。

```
connect to personnl user anitas using anpass
```

**EEE のみ:** 区分データベース環境で DB2 エクステンダーを使用している場合、DMBSTART コマンドはインスタンスに定義されているすべてのデータベース区画サーバーでエクステンダーを開始します。エクステンダー・サービスを 1 つのデータベース区画サーバーだけで開始したい場合は、開始したいノードをコマンドに指定します。下の例は、ノード番号 2 でエクステンダー・サービスを開始する場合を示しています。

```
dmbstart nodenum 2
```

**EEE のみ:** データベース区画サーバーを 1 つだけ開始する場合は、その前にそのノードで DB2 を開始しなければなりません。

これで、509ページの『第17章 クライアント側の管理コマンド』にあるその他の DB2 エクステンダー・コマンドを実行することができます。

## データベース区画の追加と除去 (EEE のみ)

区分データベース環境でエクステンダーを使用するには、エクステンダー用に定義された区画が DB2 用に定義されたものと一致していなければなりません。 DMBSTART コマンドは、現行インスタンス用に定義された各ノード上で、エクステンダー・サーバーを開始します。サーバーは、そのサーバー自体が稼働しているノードが最近作成されたものかどうかを自動的に検出し、必要な初期化を実行します。ノードを DB2 から除去した場合は、そのノードに関連付けられたエクステンダー・ファイルを手動で削除しなければなりません。

区画を追加および除去するための DB2 コマンドの詳細については、 *IBM DB2 ユニバーサル・データベース エンタープライズ拡張エディション 概説およびインストール* を参照してください。

データベース区画を追加するには、以下のステップが必要です。

1. DB2NCRT コマンドまたは DB2START ADDNODE コマンドを使用して、DB2 用の区画を作成します。
2. エクステンダーの DMBSTART NODENUM コマンドを使用して、エクステンダー用の区画を作成します。
3. 新しいノード構成を利用するために、DB2 の REDISTRIBUTE NODEGROUP コマンドを使用して、DB2 データを再分散します。
4. 新しいノード構成を利用するために、エクステンダーの REDISTRIBUTE NODEGROUP コマンドを使用して、エクステンダー・データを再分散します。

データベース区画を除去するには、以下のステップが必要です。

1. 除去したい区画から DB2 データを除去するために、DB2 の REDISTRIBUTE NODEGROUP コマンドを使用して、DB2 データを再分散します。
2. 除去したい区画からエクステンダー・データを除去するために、エクステンダーの REDISTRIBUTE NODEGROUP コマンドを使用して、DB2 データを再分散します。
3. DB2 の DB2NDROP コマンドまたは DB2STOP DROP コマンドを使用して、DB2 用の区画を除去します。
4. エクステンダーの DMBSTART NODENUM コマンドを使用して、エクステンダー用の区画を除去します。
5. 除去した区画に関連付けられているエクステンダー・ファイルを手動で除去します。

データベース区画用のエクステンダー・データは node*num* というサブディレクトリーにあります。ここで、*num* はデータベース区画に対応するノード番号です。このサブディレクトリーは、DB2MMDATAPATH 環境変数の値として指定されているディレクトリーの中にあります。除去したデータベース区画のエクステンダー・データを除去するには、該当する node*num* サブディレクトリーおよびその下のすべてのサブディレクトリーを削除してください。
(DB2MMDATAPATH の詳細については、615ページの『DB2MMDATAPATH 環境変数の使用方法 (EEE のみ)』を参照してください。)

## エクステンダー・サーバーの停止と開始

エクステンダー・サービスを使用するアプリケーションを停止しても、サーバーは、明示的に停止されるか、サーバー・マシンがリサイクルされるまで、活動状態のままです。すべてのエクステンダー・サービスを停止するには、サーバー・マシンからオペレーティング・システムのコマンド行に DMBSTOP コマンドを入力します。

クライアントからエクステンダー・サービスの停止と再始動を行うには、それぞれ、db2ext コマンド行から STOP SERVER と START SERVER コマンドを実行します。これらのコマンドによって、現行データベースに対するエクステンダー・サービスの停止と開始が行われます。

**EEE のみ:** 区分データベース環境では、インスタンス用に定義されているすべてのデータベース区画サーバー、または単一のデータベース区画サーバーだけを開始するために、DMBSTART を使用することができます。パラメーターを指定せずに DMBSTART を実行すると、すべてのデータベース区画サーバーが

開始します。 1 つのデータベース区画サーバーだけを開始したい場合は、開始したいノードを次のようにコマンドに指定します。

```
dmbstart nodenum 2
```

すでに特定のノードでサーバーを開始している場合には、そのサーバーをデータベースに再接続しなければなりません。次のようにエクステンダーの RECONNECT SERVER コマンドを使用します。

```
reconnect server at nodenum 2
```

**EEE のみ:** 区分データベース環境で DB2 エクステンダーを使用している場合、パラメーターを指定せずに DMBSTOP コマンドを実行すると、インスタンスに定義されているすべてのデータベース区画サーバーが停止します。 1 つのデータベース区画サーバーだけを停止したい場合は、まずデータベースからそのサーバーを切断します。次のようにエクステンダーの DISCONNECT SERVER コマンドを使用します。

```
disconnect server at nodenum 2
```

次いで、停止したいノードを指定して DMBSTOP コマンドを実行することができます。下の例は、ノード番号 2 でエクステンダー・サービスを停止する場合に、サーバーのコマンド行に入力するコマンドを示しています。

```
dmbstop nodenum 2
```

**EEE のみ:** データベースを保守モードで実行していない場合には、特定のノードで DMBSTOP を実行しないでください。また、ノードをオフにしている間に、そのノードでエクステンダー活動が起動されることのないようにしてください。さもないと、予期しない動作が引き起こされる結果になりかねません。

## サーバー状況の表示

サーバーから DMBSTAT コマンドを出せば、エクステンダー・サーバーの状況を表示することができます。たとえば、次のコマンドによって、使用可能なデータベースと、エクステンダーが開始され、稼働しているかどうかが表示されます。このコマンドを出す場合には、サーバーに接続されていなければなりません。

```
dmbstat
```

クライアントから GET SERVER STATUS コマンドを実行すれば、データベースに対するエクステンダー・サーバーの状況を入手することができます。たとえば、次のコマンドによって、人事データベースの状況が表示されます。

```
get server status personnl
```

# 複数のサーバー・インスタンスの作成と管理

DB2 エクステンダー・サーバーの複数のインスタンスを作成し、使用することができます。 DB2 サーバーの複数のインスタンスを作成した場合には、複数のインスタンスを作成する必要があります。 DB2 エクステンダー・サーバーのインスタンスはそれぞれ、DB2 サーバーのインスタンスと関連付けられており、同じ名前を持っています。また、システム上で使用可能な DB2 エクステンダー・サーバーのインスタンスをリストし、複数のインスタンスを同時に実行して、削除することもできます。

## 複数の DB2 エクステンダー・サーバー・インスタンスの作成

DB2 エクステンダーをインストールして、それにデフォルトの DB2 インスタンスと同じ名前を付けると、DB2 エクステンダーの初期またはデフォルトのインスタンスが作成されます。 Windows および OS/2 では、デフォルトの DB2 エクステンダー・インスタンスには、DB2 という名前が付けられます。 UNIX では、デフォルトの DB2 エクステンダー・インスタンスは、初期のデフォルトの DB2 インスタンスに付けられた名前と同じになります。 DB2 エクステンダー・サーバーのインスタンスをさらに作成するには SYSADMIN 権限が必要です。また、UNIX では root 権限が必要です。

DMBICRT コマンドを使用して、DB2 イメージ、オーディオ、ビデオ・エクステンダーのサーバーのインスタンスをさらに作成します。 DB2 インスタンス DEVINST の DB2 エクステンダー・サーバー・インスタンスを作成したい場合には、オペレーティング・システムのコマンド行で、次のように入力します。

```
dmbicrt  devinst
```

DMBICRT コマンドを実行すると、インスタンスのサブディレクトリーが作成され、DB2 エクステンダーが保守するインスタンスのリストにそのインスタンスが追加されます。

**EEE のみ:**

- デフォルトの DB2 エクステンダー・サーバー・インスタンスは、Windows では DB2MPP という名前になります。
- DMBICRT を使用して DB2 イメージ、オーディオ、ビデオ・エクステンダー・サーバーのインスタンスをさらに作成する際には、区分データベース環境におけるさまざまな操作のために DB2 エクステンダーが使用するディレクトリーを指定する必要があります。これは、DB2MMDATAPATH 環境変数 (UNIX の場合) またはレジストリー項目 (Windows の場合) で指定され

たディレクトリーです。このディレクトリーは共用ディレクトリーでなければならず、そのインスタンスにかかわるすべてのノード上に存在していなければなりません。

- Windows では、使用する TCP/IP ポートの範囲を指定する必要もあります。UNIX では、そのポート範囲を /etc/services ファイルに追加する必要があります (554ページの『DMBICRT』を参照)。

## インスタンスのリスト表示

DMBILIST コマンドを使用して、システム上で使用可能な DB2 エクステンダー・サーバーのすべてのインスタンスをリストします。どのインスタンスがアクティブであるかを見つけるには、次のコマンドを入力します。

```
echo  %DB2INSTANCE%    (Windows または OS/2)

echo $DB2INSTANCE      (UNIX)
```

## 複数インスタンスの同時実行

DB2 エクステンダー・サーバーの複数インスタンスを同時に実行するには、次のステップを実行します。

### Windows または OS/2 の場合

コマンド行で、次のようにします。

1. 次のように入力して、DB2INSTANCE 変数を、開始したいインスタンスの名前に設定します。

   ```
   set db2instance=instanceName
   ```

2. エクステンダー・サービスを開始します。

### UNIX の場合

1. インスタンス所有者またはそのインスタンスのシステム管理権限を持つユーザーとしてログインします。
2. 環境を設定します。
3. データベース・マネージャーを開始します。

## 現行インスタンスの設定

インスタンスのサービスを開始または停止するコマンドを実行すると、そのコマンドは現行のインスタンスに適用されます。そのため、DB2INSTANCE 変数をインスタンス名に設定することによって、使用する DB2 エクステンダー・サーバーのインスタンスを指定します。

## インスタンスの削除

DB2 エクステンダーのインスタンスを削除するには、次のステップを実行します。

1. 現在インスタンスを使用しているすべてのアプリケーションを停止します。
2. DMBSTOP および db2ext TERMINATE コマンドを使用して、エクステンダー・サービスとすべての db2ext コマンド行プロセッサー・セッションを停止します。
3. DB2 エクステンダー・インスタンス・ディレクトリーにある、保管したいファイルをバックアップします (QBIC カタログ・ファイルなど)。このディレクトリーにあるファイルは、インスタンスが除去されると、同時に削除されます。
4. 除去するインスタンスに対して、DMBIDROP コマンドを入力します。たとえば、DEVINST インスタンスを除去するには、次のように入力します。

```
dmbidrop devinst
```

DMBIDROP コマンドを使用して DB2 エクステンダーのインスタンスを削除しても、関連付けられた DB2 インスタンスは削除されません。それらのインスタンスは別個に削除する必要があります。 DB2 エクステンダーのインスタンスに関連付けられた DB2 インスタンスを除去しても、DB2 エクステンダー・インスタンスは削除されません。しかし、そのインスタンスを使用することはできません。

## インスタンスの移行

UNIX システムでは、DB2 UDB および DB2 エクステンダーの新規バージョンをインストールした後、使用している DB2 エクステンダーのインスタンスを移行する必要があります。

以前のバージョンで作成された、既存の DB2 エクステンダー・インスタンスを移行するには、次のようにします。

1. DB2 エクステンダーのインスタンスに関連付けられた DB2 UDB インスタンスを移行します。
2. DMBIMIGR コマンドを入力して、インスタンスを移行します。たとえば、OLDINST インスタンスを移行するには、次のように入力します。

```
dmbimigr  oldinst
```

# 第6章 エクステンダー・データのためのデータ・オブジェクトの準備

データベース、表、および列を作成して、エクステンダー・データを入れるためには、それらを使用可能にする必要があります。最初にデータベースを使用可能にします。次にデータベースの表を使用可能にします。最後に表の列を使用可能にします。

データ・オブジェクトのエクステンダー・データが必要なくなったら、それらのオブジェクトを使用不能にすることができます。

オブジェクトを使用可能にしたり、使用不能にしたりするには、C 言語プログラムから API を使用するか、db2ext コマンド行を使用します。この章の例では、それぞれの方法を示します。

## データベースの使用可能化

DBxEnableDatabase API を使用してください (ここで、 x は音声の場合は a、画像の場合は i、ビデオの場合は v になります)。または、 ENABLE DATABASE コマンドで DB2 エクステンダー用にデータベースを使用可能にします。

データベースを使用可能にすると、エクステンダーは次のことを行います。

- ご使用のデータ・オブジェクトに対して DB2xxxxx というユーザー定義タイプ (UDT) を作成します。ここで、xxxxx は、Image、Audio、Video のどれかです。 UDT は、そのタイプのオブジェクトのハンドルを保持するユーザー表の列を定義するために使用します。
- データベースのための管理サポート表 (メタデータ表と呼ばれる) を作成します。これらの表はユーザー表 (ユーザーが業務データを保管する表) ではありません。エクステンダーは、エクステンダー・データを管理するためにそれらを使用します。これらの表は手動で編集しないでください。
- そのエクステンダーに対応するユーザー定義関数 (UDF) を作成します。UDF のリストは、216ページの『ユーザー定義関数』を参照してください。

データベースを使用可能にする場合、データベースの管理サポート表 (およびその索引) を保持するための表スペースも指定しなければなりません。指定し

た表スペースの 1 つまたはそれ以上がヌル値であることがありますが、その場合には省略時表スペースが使用されます。

データベースを使用可能にするには、DBA 権限が必要です。

**EEE のみ:** 区画環境でエクステンダー用にデータベースを使用可能にする場合、指定する表スペースを、区分データベース・システム内のすべてのノードを含むノード・グループ内に定義しなければなりません。さらに、指定した表スペースは、ユーザー表と同じノード・グループ内に配置することが必要です。

## 例

次の例では、データベースが使用可能にされ、省略時の表スペースに画像データが入れられます。

**API の使用:** 61ページの図16 のコードは、 使用可能にする前の既存のデータベースに接続します。この例は、DB2 コール・レベル・インターフェースを使って作成されています。この例には、セットアップとエラー検査のコードが組み込まれています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの ENABLE.C ファイルにあります。

```
/*---- Set-up -----------------------------------------------*/
#include <stdio.h>
#include <stlib.h>
#include <string.h>
#include "dmbimage.h" /* image extender function prototypes (DBi) */
#include "utility.h"  /* utility functions */

#define MMDB_ERROR_MSG_TEXT_LEN         1000
#define SERVER_IS_DB2390 (strcmp(dbms,"DB2")==0 || strcmp(dmbs,"DSN06010")==0)

int
main(int argc, char *argv[])
{
        SQLHENV henv = SQL_NULL_HENV;
        SQLHDBC hdbc = SQL_NULL_HDBC;
        SQLHSTMT hstmt = SQL_NULL_HSTMT;
        SQLCHAR uid[18+1];
        SQLCHAR pwd[30+1];
        SQLCHAR dbname[SQL_MAX_DSN_LENGTH+1];
        SQLCHAR buffer[500];
        SQL SMALLINT dbms_sz = 0;
        char    dbms[20];

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char *program = "enable;
        char *step;
```

*図 16. データベースを使用可能にするサンプル・コード (1/3)*

```
/*---- Prompt for database name, userid, and password ----*/
       if (argc > 5) ||  (argc >=2 && strcmp(argv[1],"?")== 0))
          {
            printf("Syntax for enable - enabling a DB2 UDB database: ¥n"
                   "    enable database_name userid password¥n");
            exit(0);
          }

        if (argc == 4) {
          strcpy((char *)dbname, argv[1]);
          strcpy((char *)uid   , argv[2]);
          strcpy((char *)pwd   , argv[3]);
          }
        else {
          printf("Enter database name:¥n");
          gets((char *) dbName);
          printf("Enter userid:¥n");
          gets((char *) uid);
          printf("Enter password:¥n");
          gets((char *) pwd);
          }
/*----- connect to the database ------------------------------------------------*/
         rc = cliInitialize(&henv, &hdbc, dbname, uid, pwd);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;

/*----- find out if application is connected to DB2/UDB or DB2/390?------------*/
         rc = SQLGetInfo(hdbc, SQL_DBMS_NAME, (SQLPOINTER) &dbms,
                         sizeof(dbms), &dbms_sz);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;
```

*図 16. データベースを使用可能にするサンプル・コード (2/3)*

```
/****** enable server for image extender ************************************/
         if (!SERVER_IS_DB2390)
           {
            printf("%s: Enabling database......¥n", program);
           }
         printf("%s: This may take a few minutes, please wait......¥n", program);

         if (!SERVER_IS_DB2390)
           {
            step="DBiEnableDatabase with NULL tablespace"
            rc=DBiEnableDatabase(NULL);
           }
         if (rc < 0) {
            printf ("%s: %s failed!¥n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            if (sqlcode)
              printf("sqlcode=%i, ",sqlcode);
         }else if (rc > 0) {
            printf("%s: %s, warning detected.¥n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            printf("warning MsgText=%s¥n", errorMsgText);
         } else
            printf("%s: %s OK¥n", program, step);
/****** end of enable server ************************************************/
```

*図 16. データベースを使用可能にするサンプル・コード (3/3)*

**db2ext コマンド行の使用:** この例では、データベースはすでに接続されています。

```
enable database for db2image
```

## 表を使用可能にする

DBxEnableTable API (ここで、x は音声の場合は a、画像の場合は i、ビデオの場合は v) または、ENABLE TABLE コマンドで DB2 エクステンダー用の表を使用可能にします。

ユーザー表を使用可能にする場合、ともに使用する管理サポート表 (およびその索引) を保持するための表スペースも指定しなければなりません。指定した表スペースの 1 つまたはそれ以上がヌル値であることがありますが、その場合には省略時表スペースが使用されます。

**EEE のみ:** 区画環境でエクステンダー用に表を使用可能にする場合、指定する表スペースを、区分データベース・システム内のすべてのノードを含むノード・グループ内に定義しなければなりません。さらに、指定した表スペースは、ユーザー表と同じノード・グループ内に配置しなければなりません。

**EEE のみ:** DB2 エクステンダー列を、区分データベース環境の区分化キーとして使用することはできません。

ユーザー表に対する Control か Alter 権限が必要です。表を使用可能にするには、その表が含まれているデータベースが使用可能になっていなければなりません。

次の例では、表が使用可能にされ、省略時の表スペースに画像データが入れられます。このデータベースはすでに使用可能になっています。

**API の使用:** 65ページの図17 で、表が使用可能にされる前に、コードで表が作成され、変更がコミットされます。この例には、エラー検査コードが組み込まれています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの ENABLE.C ファイルにあります。

```
        SQLCHAR szCreate_DB2UDB[]="CREATE TABLE %s(%s mmdbsys.DB2Image,
        %s mmdbsys.DB2Video, %s mmdbsys.DB2Audio, artist varchar(25), title varchar(25)
        stock_no char(11), tw char(10), price char(10))";

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char tableName[8+18+1] = "sobay_catalog";
        char audioColumn[18+1] = "music";
        char imageColumn[18+1] = "covers";
        char videoColumn[18+1] = "video";
        char *program = "enable";
        char *step;

/*-----create table ----------------------------------------------------*/
       printf("%s: Creating table ......¥n", program);
       if (!SERVER_IS_DB2390)
         sprintf((char*) buffer, (char*) szCreate_DB2UDB,
                   tableName, imageColumn, videoColumn, audioColumn):

       rc = SQLAllocStmt(hdbc, &hstmt);
       cliCheckError(SQL_NULL_HENV, hdbc, SQL_NULL_HSTMT, rc);
       rc = SQLExecDirect(hstmt, buffer, SQL_NTS);
       cliCheckError(SQL_NULL_HENV, SQL_NULL_HDBC, hstmt, rc);

/*---- enable table for image extender ---------------------------------*/
     printf("%s: Enabling table......¥n", program);
     step="DBiEnableTable";
     if (!SERVER_IS_DB2390)
         rc = DBiEnableTable(NULL,tableName);
           }
         if (rc < 0) {
                printf("%s: %s failed!¥n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                if (sqlcode)
                  printf("sqlcode=%i, "sqlcode");
                printf("errorMsgText=%s¥n", errorMsgText);
         } else if (rc > 0) {
                printf("%s: %s, warning detected.¥n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                printf("warningMsgText=%s¥n", errorMsgText);
         } else
                printf("%s: %s OK¥n", program, step)
/*---- end of enable table ---------------------------------------------*/
```

*図 17. 表を使用可能にするサンプル・コード*

**db2ext コマンド行の使用:** この例では、表がすでに存在しており、データベースが使用可能になっています。

```
enable table employee for db2image
```

## 列を使用可能にする

DBxEnableColumn API (ここで、x は音声の場合は a、画像の場合は i、ビデオの場合は v) または、ENABLE COLUMN コマンドで DB2 エクステンダー用の列を使用可能にします。 API またはコマンドを発行するときは、適切な表および列を指定します。

列を使用可能にすると、エクステンダーは、そのユーザー表に属する管理サポート表に情報を追加します。ユーザーは、その列が含まれているユーザー表に対し Control か Alter 権限が必要です。列を使用可能にする場合には、データベースと表が使用可能になっていなければなりません。

次の例では、EMPLOYEE 表の PICTURE 列が使用可能にされ、画像データが入れられます。データベースと表は両方とも使用可能になっています。

**API の使用:** この例には、エラー検査コードが組み込まれています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの ENABLE.C ファイルにあります。

```
char imageColumn[18+1] = "covers";

/*---- enable column for image extender ----*/
     printf("%s: Enabling columns......¥n", program);
     step="DBiEnableColumn";
     rc = DBiEnableColumn(tableName, imageColumn);
     if (rc < 0) {
            printf("%s: %s failed!¥n", program, step);
            printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           if (sqlcode)
              printf("sqlcode=%i, ", sqlcode);
           printf("errorMsgText=%s¥n", errorMsgText)

    } else if (rc > 0) {
           printf("%s: %s, warning detected.¥n", program, step);
           printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           printf("warningMsgText=%s¥n", errorMsgText);
    } else
           printf("%s: %s OK¥n", program, step);
/*---- enable column for image extender ----*/
```

*図 18. 列を使用可能にするサンプル・コード*

**db2ext コマンド行の使用:** この例では、その列がすでに存在し、データベースおよび表が使用可能になっています。

```
enable column employee picture for db2image
```

## データ・オブジェクトを使用不能にする

データベースや表、列からエクステンダー・データを取り除いたら、それ以後、それを使用可能にする必要はありません。データ・オブジェクトを使用不能にするには、DISABLE コマンドを使用するか、API を使用する方法があります。エクステンダー・コマンドについては、509ページの『第17章 クライアント側の管理コマンド』を参照してください。エクステンダー API については、285ページの『第16章 アプリケーション・プログラミング・インターフェース』を参照してください。

エクステンダー・データが入っている表やデータベースを消去する場合には、それを使用不能にしてから、そのデータベースのサーバーを停止する必要があります。

## 使用不能にする

# 第7章 区分データベース・システムでのエクステンダー・データの再分散 (EEE のみ)

DB2 エンタープライズ拡張エディションを使用すると、区分データベース環境でデータベース区画サーバー (ノードともいう) を追加および削除することができます。ノードを追加した後に (またはノードを削除する前に)、新しい構成を利用するために既存のデータを再分散できます。

エクステンダー・データを再分散するには、2 つのステップを実行する必要があります。最初に、DB2 データを再分散しなければなりません。次に、DB2 エクステンダー・データを再分散できます。

## DB2 データの再分散

データを再分散する前に、DB2 の REDISTRIBUTE NODEGROUP コマンドを使用して DB2 データを再分散しなければなりません。

DB2 データの再分散の詳細については、*DB2 管理の手引き* を参照してください。

## エクステンダー・データの再分散

DB2 データを再分散したならば、次にエクステンダー・データを再分散することができます。エクステンダーの REDISTRIBUTE NODEGROUP コマンドを入力して、エクステンダー・データの再分散を開始します。

```
redistribute nodegroup
```

REDISTRIBUTE NODEGROUP コマンドは、音声、画像、およびビデオ形式のエクステンダー・データおよび QBIC 機能データを、対応するユーザー・データと同じノードに配置することによって、再分散します。

再分散プロセスがエラーを戻した場合は、もう一度コマンドを実行することができます。コマンドの応答にある指示に従い CONTINUE パラメーターをコマンドに指定して (指定しないこともある)、もう一度コマンドを実行することができます。このオプションは、プロセスを最初からやり直すのではなく、プロセスが停止した場所から続行するよう、システムに指示します。 DB2 の

REDISTRIBUTE NODEGROUP コマンドを実行した後、初めてエクステンダーの REDISTRIBUTE NODEGROUP を実行するときには、CONTINUE パラメーターを使用することはできません。

データ保全性を維持するには、一度に 1 つのノード・グループを再分散します。そのノード・グループの再分散が終わるまで待ってから、次のノード・グループの再分散を開始します。

このコマンドを使用する前に、必ずデータベースに接続してください。

このコマンドを実行するには、SYSADM または DBADM 権限が必要です。

# 第8章 データ・オブジェクトとメディア・ファイルの追跡

DB2 エクステンダーを使用するアプリケーションを作成したり、デバッグしたりする場合、エクステンダー・データに対してどのオブジェクトが使用可能になっているかが分かると便利です。たとえば、イメージ・データの場合にある表を使用できることが分かれば、アプリケーションでイメージ・ファイルをその表に保管することができます。

また、ユーザー表と外部メディア・ファイルの関連を知っていると役に立ちます (たとえば、どの表が特定のファイルを参照しているか、またはどのファイルが特定の表によって参照されているかなど)。さらに、使用する表が、システムにもはや存在しないファイルを参照していないかどうかを知ることもできます。

**適切な特権が必要:** 表のデータを追跡するには、表へのアクセスが必要です。広範囲の追跡操作を行いたい場合 (データベース内のユーザー表全体の中のどの項目がファイルを参照しているかを見つける場合など)、検索されるユーザー表および関連する管理サポート表の使用可能の列に対する SYSADM 権限、DBADM 権限、または SELECT 特権が必要です。すべての表に対するアクセス権限がない場合、アクセス可能な表についてのみエクステンダーは追跡情報を戻します。また、必要ないくつかの表に対するアクセス権限がないことを示すコードも戻します。

## データ・オブジェクトの状況の検査

データベース、表、列が使用可能であり、エクステンダー・データを入れることができるかどうかを検査できます。次の例は、現行のデータベースがイメージ・エクステンダー用に使用可能になっているかどうかを判断します。このデータベースはすでに使用可能になっています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの API.C ファイルにあります。

**API の使用:** 72ページの図19　のサンプル・コードには、エラー検査コードが組み込まれています。

```
/*---- Query the database using DBiIsDatabaseEnabled API. ----------*/
step="DBiIsDatabaseEnabled API";
rc = DBiIsDatabaseEnabled(&status);
if (rc < 0) {
     printf("%s: %s FAILED!¥n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
     fail = TRUE;
} else if (rc > 0) {
     printf("%s: %s, warning detected.¥n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
} else {
     if (status == 1) {
         printf("%s: ¥"%s¥" database is enabled for Image Extender¥n",
                argv[0], dbName);
         printf("%s: %s PASSED¥n¥n", argv[0], step);
     } else if (status == 0) {
         printf("%s: ¥"%s¥" database is not enabled for Image Extender¥n",
                argv[0], dbName);
         printf("%s: %s PASSED¥n¥n", argv[0], step);
     } else
         printf("%s: %s FAILED, invalid status!¥n", argv[0], step);
}
```

*図 19. データベースが使用可能になっているかどうかを調べるサンプル・コード*

### db2ext コマンド行の使用:

```
get extender status
```

ユーザー表や列の状況の検査は、データベースの状況の検査と同じようにして行います。 DBxIsTableEnabled API と DBxIsColumnEnabled API を使用するか、GET EXTENDER STATUS コマンドを使用します。

## ファイルを参照する表項目の検索

ユーザー表のどの項目が外部メディア・ファイルを参照しているのかを調べることができます。 DBxAdminIsFileReferenced API を使用して、現行のデータベース内にあるユーザー表全体またはサブセットのどの項目が外部メディア・ファイルを参照しているかを調べることができます。 DBxIsFileReferenced API を使用して、特定のユーザー表のどの項目が外部メディア・ファイルを参照しているかを調べることができます。

**API の使用:** 73ページの図20 のサンプル・コードは、そのファイルが参照されている回数、どこで参照されているかを戻します。この例には、エラー検査コ

ードが組み込まれています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの API.C ファイルにあります。

```
/*---- Query the database using DBiAdminIsFileReferenced API. ------*/
step="DBiAdminIsFileReferenced API";
rc = DBiAdminIsFileReferenced((char*) uid, filename, &count, &filelist);
if (rc < 0) {
     printf("%s: %s FAILED!¥n", program, step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
} else if (rc > 0) {
     printf("%s: %s, warning detected.¥n", program, step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsg Text);
} else {
     if (count == 0)
          printf("%s: ¥"%s¥" file is not referenced¥n",
                 program, filename);
     else {
          printf("%s: ¥"%s¥" file is referenced %d times¥n",
                 program, filename);
          for (i=0; i < count; i++)
             {
                 /* filename is NULL for any IsFileReferenced APIs */

                  printf ("filename = %s¥n", filelist[i].filename);
                  printf ("¥tqualifier = %s¥n", filelist[i].tqualifier);
                  printf ("¥ttable = %s¥n", filelist[i].tname);
                  printf ("¥thandle = %s¥n", filelist[i].handle);
                  printf ("¥tcolumn = %s¥n", filelist[i].column);
                  if (filelist[i].filename)
                          free (filelist[i].filename);
             }
          }
          if (filelist)
                free (filelist);
          printf("%s: %s PASSED¥n¥n", argv[0], step);
}
```

*図 20. ファイルがユーザー表によって参照されているかどうかを調べるサンプル・コード*

## 表項目によって参照されているファイルの検索

DBxAdminGetReferencedFiles API または GET REFERENCED FILES コマンドを使用して、現行のデータベース内のユーザー表全体またはサブセットによって参照されている外部メディア・ファイルをリストします。

DBxGetReferencedFiles API または GET REFERENCED FILES コマンドを使用して、特定の表で参照されている外部メディア・ファイルをリストします。

**API の使用:** 図21 のサンプル・コードは、見つかったファイルの数と、それらのファイルのリストを戻します。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの API.C ファイルにあります。

```
/*---- Query the database using DBiAdminGetReferencedFiles API. ----------*/
step="DBiAdminGetReferencedFiles API"
rc = DBiAdminGetReferencedFiles((char*) uid, &count, &filelist);
if (rc < 0) {
   printf("%s: %s FAILED!¥n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf{"sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
} else if (rc > 0) {
   printf("%s: %s, warning detected.¥n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
} else {
   if (count == 0)
     printf("%s: no referenced files¥n", program);
  else {
     printf("%s: %d referenced files¥n", program, count);
     for (i=0; i < count; i++)
       {
   printf ("filename = %s¥n", filelist[i].filename);
   printf ("¥tqualifier = %s¥n", filelist[i].tqualifier);
   printf ("¥ttable = %s¥n", filelist[i].tname);
   printf ("¥thandle = %s¥n", filelist[i].handle);
   printf ("¥tcolumn = %s¥n", filelist[i].column);
       if (filelist[i].filename)
               free (filelist[i].filename);
     }
   }
   if (filelist)
       free (filelist);
   printf("%s: %s PASSED¥n¥n", argv[0], step);
}
```

*図 21. 参照されているファイルのリストを入手するサンプル・コード*

**db2ext コマンド行の使用:**

```
get referenced files user anitas for db2image
```

## メディア・ファイルが存在するかどうかの検査

メディア・ファイルがシステムから削除されたが、それを参照するユーザー表が更新されていないとします。その場合、そのユーザー表によって参照されるメディア・ファイルのうち、アクセス不能なファイルをリストしてください。

DBxAdminGetInaccessibleFiles API または GET INACCESSIBLE FILES コマンドを使用して、現行のデータベース内のユーザー表全体またはサブセットによって参照されている外部メディア・ファイルをリストします。
DBxGetInaccessibleFiles API または GET INACCESSIBLE FILES コマンドを使用して、特定の表で参照されている外部メディア・ファイルをリストします。

# 第9章 管理サポート表のクリーンアップ

DB2 エクステンダーを使用していると、古い項目が次第に管理サポート表にたまります。また、メディア・ファイルが削除されたのに、その参照がデータベースから削除されない場合もあります。古いメタデータを削除すれば、パフォーマンスを改善し、記憶スペースを再利用することができます。

**API の使用:** 図22 のサンプル・コードは、ANITAS によって所有されている全ユーザー表の画像メタデータをクリーンアップします。この例には、エラー検査コードが組み込まれています。このサンプル・プログラム全体は、SAMPLES サブディレクトリーの API.C ファイルにあります。

```
/*---- query database using DBiAdminReorgMetadata API ----*/
step="DBiAdminReorgMetadata API";
rc = DBiAdminReorgMetadata("anitas");
if (rc < 0) {
        printf("%s: %s FAILED!¥n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
        fail = TRUE;
} else if (rc > 0) {
        printf("%s: %s, warning detected.¥n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s¥n", sqlcode, errorMsgText);
} else
        printf("%s: %s PASSED¥n¥n", argv[0], step);

/*---- end of query using DBiAdminReorgMetadata API ----*/
```

*図 22. 管理サポート表のクリーンアップを行うサンプル・コード*

### db2ext コマンド行の使用:

```
reorg database user anitas for db2image
```

DBA でなくても、CONTROL 権限があれば、DBxReorgMetadata API か REORG コマンドを使って、所有する表に対しメタデータのクリーンアップを行うことができます。

# 第3部 画像、音声、ビデオのデータのためのプログラミング

# 第10章 プログラミングの概要

この章では、DB2 エクステンダーのプログラミングの概要を説明します。つまり、エクステンダーのプログラミングを始める前に知っておかなければならない情報や、エクステンダーのコーディングのしかたを示すサンプル・アプリケーションを提供します。

## エクステンダーの UDF と API の使用

DB2 エクステンダーはデータベース内の画像、音声、およびビデオ・データを保管、アクセス、および操作するユーザー定義関数を提供します。アプリケーション・プログラムでこれらの UDF を要求するコーディングをするには、SQL 組み込み関数を要求するのと同じように、SQL ステートメントを使います。組み込み関数と同様に、UDF はデータベース・サーバーで実行されます。

C のアプリケーション・プログラムから次の SQL ステートメントを実行すると、DB2Image という名前のイメージ・エクステンダー UDF が画像をデータベース表に保管します。ソース・イメージの内容はサーバー・ファイルにあります。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
           CURRENT SERVER,                 /*database */
           '/employee/images/ajones.bmp', /*image content*/
           'ASIS',                         /*keep the image format*/
           :hvStorageType,                 /*store image in DB as BLOB*
           'Anita''s picture')             /*comment*/
       );
```

エクステンダーのアプリケーション・プログラミング・インターフェース (API) は、画像を表示し、音声やビデオのオブジェクトを再生するためにも使用することができます。これらの API をコーディングするには、C のクライアント関数呼び出しを使います。これらの関数は、データベース・クライアント・ワークステーションで実行されます。

次の C ステートメントには、DBiBrowse という名前の API が組み込まれています。 API は画像ハンドルのデータを検索してから、次のようにブラウザーを開始して画像を表示します。

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvImg_hdl[251];
EXEC SQL END DECLARE SECTION

EXEC SQL SELECT PICTURE INTO :hvImg_hdl
   WHERE NAME= 'Robert Smith';

rc=DBiBrowse(
             "ib %s",                 /*image browser*/
             MMDB_PLAY_HANDLE,        /*use image handle*/
             hvImg_hdl,               /*image handle*/
             MMDB_PLAY_NO_WAIT);      /*run browser independently*/
```

**UDF はインスタンスのユーザー ID で実行しなければなりません:** DB2 エクステンダー UDF は、DB2 エクステンダー・インスタンスと同じユーザー ID で実行しなければなりません。さらに、DB2 エクステンダー・インスタンスを作成したり、既存の DB2 エクステンダー・インスタンスを使用する場合には、UDF を DB2 インスタンスと同じユーザー ID で実行しなければなりません。

**DB2 の構成が適正でなければなりません:** DB2 エクステンダー、特に DB2 エクステンダー UDF が適正に操作されるようにするには、DB2 の構成が適正でなければなりません。とりわけ、APP_CTL_HEAP_SZ データベース構成パラメーターの設定が適正でなければなりません。

## エクステンダーの UDF と API で行えるタスク

表4 は、エクステンダーの UDF と API で行えるタスクと、各タスクが説明されている場所を示したものです。

*表 4. DB2 エクステンダー API で行えるタスク*

| タスク | 参照 |
|---|---|
| 画像、音声、ビデオのオブジェクトを保管する | 98 ページ |
| 画像、音声、ビデオのオブジェクトを検索する | 112 ページ |
| 画像、音声、ビデオの属性を検索して使用する | 117 ページ |
| 画像、音声、ビデオのオブジェクトに対応する注釈を検索する | 120 ページ |
| 画像、音声、ビデオのオブジェクトを更新する | 121 ページ |
| 画像オブジェクトを表示する | 137 ページ |
| サムネール・サイズの画像またはビデオ・フレームを表示する | 140 ページ |

*表 4. DB2 エクステンダー API で行えるタスク (続き)*

| タスク | 参照 |
|---|---|
| 音声やビデオのオブジェクトを再生する | 142 ページ |
| 画像を内容で照会する | 143 ページ |
| ビデオ・シーンの変化を検知する | 185 ページ |

## エクステンダー例のためのサンプル表

この章では、DB2 エクステンダーを使用するプログラミング例を随所に示します。これらの例では、人事情報を含む EMPLOYEE という名前のデータベース表がすでに作成されているものとします。この表には、従業員の識別番号と名前が入った列があります。エクステンダーの種類によっては、この表に、さらに従業員の写真、音声によるあいさつ、およびビデオ・クリップのための列があります。

84ページの図23 は、従業員表の構造と、その表を作成するために使用する SQL ステートメントを示したものです。

```
CREATE TABLE  employee(
    id          CHAR(6),
    name        VARCHAR(40),
    picture     DB2Image,

    sound       DB2Audio,

    video       DB2Video
    );
```

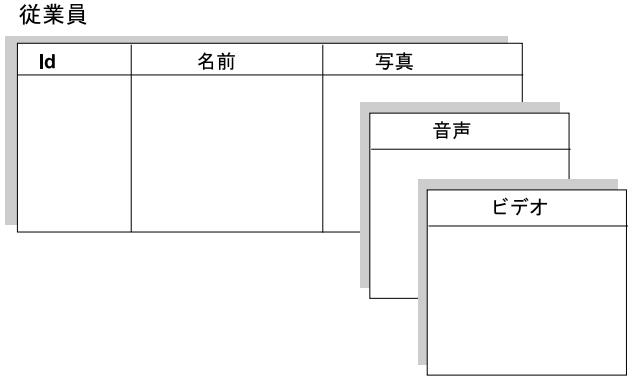



*図 23. DB2 エクステンダーのプログラミング例で使用する表*

## DB2 エクステンダーのプログラミングを始める前に

DB2 エクステンダーを使用するプログラムを開発する前に、DB2 アプリケーションの開発プロセスとプログラミング技法について知っておく必要があります。 *DB2 アプリケーション開発の手引き* を参照してください。 DB2 エクステンダーを使ったプログラムを開発するプロセスは、本質的には従来の DB2 アプリケーションのそれと同じです。

エクステンダーによって新しいデータ・タイプと関数が定義されるので、このアプリケーション・プログラムのコードは従来の DB2 アプリケーションと異なります。たとえば、86ページの図24 に示す C でコーディングされたアプリケーションは、イメージ・エクステンダーを使用して、データベース表に保管されている GIF 形式の画像を識別します。画像を識別すると、プログラムは、画像ブラウザーを使ってそれらを表示します。

この例が示すように、DB2 エクステンダーを使用するアプリケーションでは、次の機能を行う必要があります。

**1** エクステンダー定義を組み込む。この例では、dmbimage.h ファイルが、イメージ・エクステンダーのためのインクルード (ヘッダー) ファイルです。このインクルード・ファイルによって、エクステンダー用の定数や、変数、関数プロトタイプが定義されます。

**2** 必要に応じて、UDF への入力またはそこからの出力を入れるホスト変数、または API 呼び出しへの入力を入れるホスト変数を定義する。この例では、hvFormat、hvSize、hvWidth、hvHeight、hvComment がホスト変数で、イメージ・エクステンダーの UDF によって検索されるデータを入れるために使用されます。ホスト変数 hvImg_hdl には、イメージ・エクステンダー API 呼び出しの入力として指定する画像ハンドルが入ります。

**3** 必要に応じて UDF 要求を指定する。この例では、SIZE、WIDTH、HEIGHT、COMMENT、FORMAT がイメージ・エクステンダー UDF です。

**4** 必要に応じて API 呼び出しを指定する。この例では、DBiBrowse がローカル C 関数に対する API 呼び出しであり、表からハンドルが取り出された画像を表示します。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlcodes.h>
#include <dmbimage.h>  1

int count=0;

long
main(int argc,char *argv[])
{
EXEC SQL BEGIN DECLARE SECTION;  2
         char hvImg_hdl[251];                  /* image handle */
         char hvDBName[19];                    /* database name */
         char hvName[40];                      /* employee name */
         char hvFormat[9];                     /* image format */
         long hvSize;                          /* image size */
         long hvWidth;                         /* image width */
         long hvHeight;                        /* image height */
         struct  {
                  short len;
                  char data[32700]
         } hvComment;                          /* comment about the image */
 EXEC SQL END DECLARE SECTION;

/*  Connect to database  */

strcpy(hvDBName, argv[1]);                     /* copy the database name */

EXEC SQL CONNECT TO :hvDBName IN SHARE MODE;
/*
 * Set current function path
*/
EXEC SQL SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH;
```

*図 24. DB2 エクステンダーを使用するアプリケーション (1/2)*

```
/*
 * Select (query) using Image Extender UDF
 *
 * The SQL statement below finds all images in GIF format.
 */
EXEC SQL DECLARE c1 CURSOR FOR
         SELECT PICTURE, NAME,                  3
                SIZE(PICTURE), WIDTH(PICTURE),
                HEIGHT(PICTURE), COMMENT(PICTURE)
         FROM EMPLOYEE
         WHERE PICTURE IS NOT NULL AND
               FORMAT(PICTURE) LIKE 'GIF%'
FOR FETCH ONLY;

EXEC SQL OPEN c1;
for (;;) {
         EXEC SQL FETCH c1 INTO :hvImg_hdl, :hvName, :hvSize,
                                :hvWidth, :hvHeight, :hvComment;
         if (SQLCODE != 0)
                   break;

         printf("¥nRecord %d:¥n", ++count);
         printf("employee name = '%s'¥n", hvName);
         printf("image size = %d bytes, width=%d, height=%d¥n",
                                              hvSize, hvWidth, hvHeight);
         hvComment.data[Comment.len]='¥0';
         printf("comment len = %d¥n", hvComment.len);
 printf("comment = %s¥n", hvComment.data);
/*
* The API call below displays the images
*/
 4  rc=DBiBrowse ("ib %s",MMDB_PLAY_HANDLE,hvImg_hdl,
                         MMDB_PLAY_WAIT);
}

EXEC SQL CLOSE c1;

/* end of program */
```

*図 24. DB2 エクステンダーを使用するアプリケーション (2/2)*

## エクステンダー定義の組み込み

アプリケーション・プログラムには、使用するエクステンダーごとにインクルード (ヘッダー) ファイルが必要です。このインクルード・ファイルによって、エクステンダー用の定数や、変数、関数プロトタイプが定義されます。インクルード・ファイルの名前は次のようになります。

| インクルード・ファイル | エクステンダー |
|---|---|
| **dmbimage.h** | イメージ |

| | |
|---|---|
| **dmbqbapi.h** | イメージ (イメージ内容照会) |
| **dmbaudio.h** | オーディオ |
| **dmbvideo.h** | ビデオ |
| **dmbshot.h** | ビデオ (シーン変化の検知) |

インクルード・ファイルを C プログラムに指定するには、#include 命令文を使用します。たとえば、次の命令文を指定すれば、イメージ・エクステンダーのインクルード・ファイルが組み込まれます。

```
#include <dmbimage.h>
```

## UDF 名と UDT 名の指定

DB2 エクステンダー UDF の正式名は mmdbsys.function-name です。 DB2 エクステンダー UDT の正式名は mmdbsys.type-name です。ここで mmdbsys は、その関数または特殊タイプのスキーマ名です。たとえば、Content UDF の正式名は mmdbsys.Content、イメージ・エクステンダーによって作成される DB2Image データ・タイプの正式名は mmdbsys.DB2Image です。次のようにして、現行関数パスをすでに mmdbsys に設定していれば、mmdbsys というスキーマ名は省略することができます。

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
SET CURRENT PATH = mmdbsys, CURRENT PATH
```

## ラージ・オブジェクトの伝送

画像、音声クリップ、ビデオ・クリップなどのラージ・オブジェクトをアプリケーションと DB2 データベースの間で伝送する場合、いくつかの方法があります。どの方法を使うかは、オブジェクトをファイルとの間で送受信するか、メモリー・バッファーとの間で送受信するかによって異なります。さらにどの方法を使うかは、ファイルがクライアント・マシンにあるか、データベース・サーバー・マシンにあるかによっても異なります。

### オブジェクトが表とサーバー・ファイルの間で伝送される場合

オブジェクトをデータベース表とサーバー・ファイルとの間で伝送する場合には、該当するエクステンダー UDF 要求にファイル・パスを指定します。エクステンダー UDF とファイルは両方ともサーバーにあるので、エクステンダーはそのファイルを見つけることができます。たとえば、次の SQL ステートメントでは、内容がサーバー・ファイルにある画像がデータベース表に保管されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;
```

```
hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Image(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'ASIS',
          :hvStorageType,
          'Anita''s picture')
       );
```

### オブジェクトがクライアント・バッファーとの間で送受信される場合

エクステンダーがメモリー・バッファーに直接アクセスすることはできません。クライアント・マシンのバッファーとの間でオブジェクトを送受信する場合には、バッファーの位置を指定する以外の方法でそれを行う必要があります。バッファーとの間でオブジェクトを送受信する方法の 1 つは、ホスト変数を使う方法です。アプリケーションと DB2 アプリケーションとの間でオブジェクトを伝送するには、通常この方法を使います。

ラージ・オブジェクトに対するホスト変数の定義や使用は、従来の文字や数値のオブジェクトの場合と同じようにして行います。つまり、ホスト変数を DECLARE セクションに宣言し、伝送する値をそれに割り当てるか、そこに伝送された値にアクセスします。

画像、音声、ビデオのデータに対しホスト変数を宣言する場合には、データ・タイプとして BLOB を指定する必要があります。 UDF を使ってオブジェクトの保管や、取り出し、更新を行う場合には、適切なホスト変数を UDF 要求の引き数として指定します。その形式は、SQL ステートメントに指定する他のホスト変数の場合と同じです。

たとえば、次の SQL ステートメントでは、hvaudio という LOB ロケーターを宣言して使用し、ビデオ・クリップをデータベース表から取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) hvaudio;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Audio(
          CURRENT SERVER,
          :hvaudio,
```

```
        'WAVE',
        CAST(NULL as LONG VARCHAR),
        'Anita''s voice')
    );
```

### LOB ロケーターの使用

音声クリップやビデオ・クリップなどのラージ・オブジェクトは非常に大きい場合があるので、ホスト変数を使うのがそれらを操作する最も効率的な方法とは限りません。アプリケーションで LOB を操作するときには、**LOB ロケーター**を使う方がよい場合があります。

LOB ロケーターはホスト変数に保管される小さな (4 バイトの) 値です。プログラムではこれを使って、DB2 データベースのより大きな LOB を参照することができます。 LOB ロケーターを使えば、プログラムでは、LOB をそれが通常のホスト変数に保管されているかのように操作することができます。違いは、LOB をクライアント・マシンのアプリケーションとデータベース・サーバーとの間で転送する必要がないということです。たとえば、データベース表の LOB を選択すると、その LOB はサーバーに残り、LOB ロケーターがクライアントに移ります。

LOB ロケーターは、DECLARE セクションに宣言し、ホスト変数と同じようにして使用します。 LOB ロケーターを画像や、音声、ビデオのデータに対して宣言する場合には、データ・タイプとして BLOB_LOCATOR を指定する必要があります。たとえば、次の SQL ステートメントでは、video_loc という LOB ロケーターを宣言して使用し、ビデオ・クリップをデータベース表から取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR video_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(VIDEO)
     INTO :video_loc
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

**UDF は LOB ロケーターを使用します:** 画像、音声、ビデオのオブジェクトを保管、取り出し、および更新する DB2 エクステンダーの UDF は LOB ロケーターを使用します。 DB2 エクステンダー V1 の UDF は LOB ロケーターを使用しなかったので、2 MB より大きいオブジェクトを処理できませんでした。この制限のため、ユーザーは 2 MB より大きなオブジェクトを複数のセグメントに分けて伝送しなければなりませんでした。これらの UDF は現在、LOB ロケーターを使用するので、2 MB の制限は除去されました。

### オブジェクトがクライアント・ファイルとの間で送受信される場合

クライアント・ファイル参照変数を使用すれば、アプリケーション・プログラムでラージ・オブジェクトのためのバッファー・スペースを割り振らずに済みます。 UDF でファイル参照変数を使用すると、DB2 は、BLOB 内容をファイルと UDF の間で直接渡します。

ファイル参照変数は、DECLARE セクションで宣言し、ホスト変数と同じようにして使用します。画像、音声、ビデオのデータに対してファイル参照変数を宣言する場合には、データ・タイプとして BLOB_FILE を指定する必要があります。しかし、オブジェクトの内容が入るホスト変数とは異なり、ファイル参照変数にはファイルの名前が入ります。ファイルのサイズは、その UDF に定義された BLOB のサイズ以下でなければなりません。

入力や出力におけるファイル参照変数の使用方法については、各種のオプションがあります。オプションを選択するには、プログラムのファイル参照変数の構造体において、FILE_OPTIONS フィールドの値を設定します。その場合、次のオプションから選択できます。

#### 入力オプション

SQL_FILE_READ。このファイルは、オープン、読み取り、クローズが可能です。ファイルのデータの長さ (バイト単位) は、ファイルのオープン時に判別されます。ファイルの長さ (バイト単位) が、ファイル参照変数の構造体の data_length フィールドに入ります。

#### 出力オプション

SQL_FILE_CREATE。このオプションを指定すると、ファイルがなければ、新規ファイルが作成されます。ファイルがすでにある場合は、エラー・メッセージが戻されます。ファイルの長さ (バイト単位) が、ファイル参照変数の構造体の data_length フィールドに入ります。

SQL_FILE_OVERWRITE。このオプションを指定すると、ファイルがなければ、新規ファイルが作成されます。ファイルがすでにある場合は、そのファイルのデータが新しいデータによって上書きされます。ファイルの長さ (バイト単位) が、ファイル参照変数の構造体の data_length フィールドに入ります。

SQL_FILE_APPEND。このオプションを指定すると、ファイルがすでにあれば、そのファイルに出力が追加されます。ファイルがなければ、新規ファイルが作成されます。ファイルに追加されたデータの長さ (ファイル全体の長さではない) が、ファイル参照変数の構造体の data_length フィールドに入ります (バイト単位)。

たとえば、次のステートメントでは、Img_file というファイル参照変数を宣言し、それを使って、画像 (その内容はクライアント・ファイルにある) をデータベース表に保管します。 FILE_OPTIONS フィールドに、SQL_FILE_READ が割り当てられていることに注目してください。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
  EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name,"/employee/images/ajones.bmp");
Img_file.name_length=strlen(Img_file.name);
Img_file.file_options=SQL_FILE_READ;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Image(
          CURRENT SERVER,
          :Img_file,
          'ASIS',
          CAST(NULL as LONG VARCHAR),
          'Anita''s picture')
       );
```

### オブジェクトの伝送時にファイル名を指定する場合

DB2 エクステンダーでは、オブジェクトを保管、検索、または更新する際、ファイル名を指定する方法に柔軟性を持たせることができます。

保管、取り出し、および更新操作に対して、完全に修飾されたファイル名 (つまり、ファイル名の前に完全なパス) を指定できますが、相対ファイル名を指定した方が良いでしょう。 AIX、HP-UX、および Solaris では、相対ファイル名は、スラッシュで始まらない任意のファイル名です。 OS/2 および Windows では、相対ファイル名は、ドライブ文字とコロンと ¥ が先頭に付かない任意のファイル名です。

相対ファイル名を指定する場合、エクステンダーはさまざまなクライアントおよびサーバーの環境変数のディレクトリー指定を使用して、ファイル名を解決します。全パス名は、先行部分 (通常はマウント・ポイントと関連付けられている) と、必要なファイルを一意的に識別する後続パス名から構成されます。後続パス名は UDF で指定されます。環境変数は、相対ファイル名を解決する際に検索する主要なパス名のリストを提供します。ファイル名を解決するために DB2 エクステンダーが使用する環境変数については、613ページの『付録A. DB2 エクステンダー用の環境変数の設定』を参照してください。

エクステンダーは、さらに、ファイル名の形式を適宜変更します。ファイル名は、サーバーに渡されると、そのオペレーティング・システムにとって適切な

形式に変換されます。たとえば、c:¥dir1¥abc.bmp という OS/2 のファイル名は、AIX サーバーに渡されると /dir1/abc.bmp に変換されます。

## 戻りコードの処理

DB2 エクステンダー UDF を要求する場合を含め、プログラムのすべての組み込み SQL ステートメントまたは DB2 CLI 呼び出しは、その組み込み SQL ステートメントまたは DB2 CLI 呼び出しが正常に行われたかどうかを示すコードを生成します。また、管理 API など、他の DB2 エクステンダー API も、処理が正常に行われたかどうかを示すコードを戻します。プログラムでは、組み込み SQL ステートメント、CLI 呼び出し、および API から戻されたコードを検査し、応答する必要があります。

これらの戻りコードの処理方法については、567ページの『第19章 診断情報』を参照してください。

エクステンダー API が正常に作業単位を完了できない場合、ロールバック操作が実行されます。 API はエラー・コードも戻します。ロールバック操作は、データベースが直前の整合性ポイントに戻ることができるように実行されます。詳細については、285ページの『第16章 アプリケーション・プログラミング・インターフェース』を参照してください。

## ユニコード・サポート

イメージ、オーディオ、およびビデオ・エクステンダーのユニコード・サポートについては、次の点を守ってください。

- ユニコード・ストリングを指定できるパラメーターは、次の UDF の注釈欄だけです。
  - mmdbsys.db2image( ) イメージのインポート
  - mmdbsys.db2audio( ) オーディオのインポート
  - mmdbsys.db2video( ) ビデオのインポート
  - mmdbsys.replace( ) イメージ、オーディオ、またはビデオの置換
  - mmdbsys.comment( ) 注釈の更新
- ユニコード・データベースにアクセスする予定であれば、ユニコードをサポートするように DB2 エクステンダー・インスタンスをセットアップしなければなりません。ユニコード・インスタンスはユニコード・データベースしか処理しません。

## 始める前に

エクステンダー・インスタンスがユニコードをサポートするためには、DMBSTART を呼び出す前に、環境変数 DB2CODEPAGE を 1208 に設定します。

# 第11章 オブジェクトの保管、取り出し、および更新

この章では、DB2 エクステンダーのユーザー定義関数を使って、画像、音声、ビデオの保管、取り出し、更新をどのように行うかについて説明します。

## 画像、音声、ビデオの形式

表5 は、画像、音声、ビデオのオブジェクトを保管、取り出し、または更新する際の形式を示したものです。画像オブジェクトの場合に限り、画像を保管、取り出し、または更新する際に、イメージ・エクステンダーでその形式を変換することができます。 (音声とビデオのオブジェクト形式は、保管、取り出し、更新の際に変換することはできません。)

表の読み取りの列と書き込みの列は、どの形式が読み取り可能で、どの形式が書き込み時に変換可能かを示します。読み取り列に o がある場合には、その対応するオブジェクト形式は、保管、取り出し、更新で使用できます。書き込みの列に o がある場合には、オブジェクト (画像のみ) は、保管、取り出し、更新で、対応する形式に変換が可能です。たとえば、BMP 形式の画像は、保管、取り出し、更新の際に GIF 形式に変換が可能です。 JPG 形式の画像は TIF 形式に変換できます。しかし、TIF 形式の画像は JPG 形式に変換できません。

形式指定が表では大文字になっていますが、保管、取り出し、更新の要求では大文字小文字の区別はありません。たとえば、GIF、gif、Gif と指定しても同じことです。

*表 5. DB2 エクステンダーで処理可能な形式*

| 形式 | 説明 | 読み取り | 書き込み |
|---|---|---|---|
| | **画像形式** | | |
| _IM | PS/55 音声ビデオ接続 (AVC) | o | |
| BMP | OS/2 - Microsoft Windows ビットマップ[1] | o | o |
| EPS | カプセル化された PostScript | | o |
| EP2 | カプセル化されたレベル 2 PostScript | | o |
| GIF | Compuserve GIF89a (アニメーション化された GIF を含む[2]) および 87 | o | o |
| IMG | IOCA イメージ | o | o |

## 形式

*表 5. DB2 エクステンダーで処理可能な形式 (続き)*

| 形式 | 説明 | 読み取り | 書き込み |
|---|---|---|---|
| IPS | Brooktrout FAX カード・ファイル | o | o |
| JPG | JPEG[3] (JFIF 形式) | o | |
| PCX | PC ペイント・ファイル (グレースケールのみ) | o | o |
| PGM | ポータブル・グレー・マップ (PBMPLUS からの) | o | o |
| PS | PostScript | | o |
| PSC | 圧縮された PostScript イメージ | | o |
| PS2 | PostScript レベル 2 (カラー) | | o |
| TIF | すべての TIFF 5.0 形式 | o | o |
| YUV | YUV 用ディジタル・ビデオ | o | o |
| | **音声形式** | | |
| AIF または AIFF | 音声交換ファイル形式 | o | |
| AIFFC | 圧縮された音声交換ファイル形式 | o | |
| AU | Sun 音声ファイル形式 | o | |
| MIDI | 楽器ディジタル・インターフェース | o | |
| MPG1 または MPEG1 | 動画エキスパート・グループ 1 | o | |
| WAV または WAVE | ウェーブ | o | |
| | **ビデオ形式** | | |
| AVI | 音声 / ビデオ・インターリーブ | o | |
| MPG1 または MPEG1 | 動画コーディング・エキスパート・グループ 1 | o | |
| MPG2 または MPEG2 | 動画コーディング・エキスパート・グループ 2 | o | |
| QT | Quicktime (AVI) | o | |

## 画像変換オプション

表6 は、画像の保管、取り出し、または更新時に指定できる変換オプション (形式変換に加えて) をリストしています。イメージ・エクステンダーは指定をターゲット・イメージに適用します。ソース・イメージは変更されません。

各変換オプションはパラメーター / 値のペアとして指定します。各パラメーターに指定できる値を表にリストしました。

*表 6. 画像変換オプション*

| パラメーター | 説明 | 値 |
|---|---|---|
| -b | 各画像サンプルを表すのに使用されるビット数 | 1 または 8 ビット |
| -s[4] | 倍率 | ゼロより大きい任意の 10 進数値。倍率は、元の画像に対する変換後の画像のサイズの比率を指定します。たとえば、倍率 0.5 は元のサイズの半分に画像を変換します。倍率 2.0 は元のサイズの 2 倍に画像を変換します。 |
| -p | 光度 (画像反転)。指定された値に基づいて、このオプションは画像の解釈を変更します。画像自体を変更させることはできません。このオプションは、白黒画像またはグレースケール画像のみに適用され、 GIF 形式の画像には適用されません。 | 0 = 黒<br>1 = 白 |
| -n | 光度 (画像反転)。このオプションは、黒から白、また白から黒に画像を反転させます。このオプションは、白黒またはグレースケール画像にのみ適用されます。 | なし |
| -r[4] | 回転 | 0 = 0 度 (回転なし)<br>1 = 90 度 (左回り)<br>2 = 90 度 (右回り)<br>3 = 180 度 |

1. 読み取りは、OS/2 バージョン 1、OS/2 バージョン 2、Windows バージョン 2、Windows バージョン 3、および Windows NT BMP 形式でサポートされます。書き込みは、OS/2 バージョン 1 BMP 形式でサポートされます。

2. DB2 イメージ・エクステンダーは、最初の画像のみの属性情報をアニメーション化された GIF ファイルに保管します。

3. このサポートには、独立 JPEG グループの成果に一部依存するソフトウェアが使用されます。

## 形式

*表 6. 画像変換オプション (続き)*

| パラメーター | 説明 | 値 |
|---|---|---|
| -x[4] | ピクセルの幅 | ピクセルの数 |
| -y[4] | ピクセルの高さ | ピクセルの数 |
| -c | 圧縮タイプ | 0 = IBM MMR<br>1 = CCITT グループ 3 1-D<br>2 = CCITT グループ 3 2-D (k=2)<br>3 = CCITT グループ 3 2-D (k=4)<br>4 = CCITT グループ 4<br>6 = TIFF タイプ 2<br>10 = 圧縮なし<br>14 = LZW<br>15 = TIFF Packbits<br>25 = JBIG |

## 画像、音声、ビデオのオブジェクトの保管

SQL INSERT ステートメントで DB2Image、DB2Audio、または DB2Video UDF を使用して、画像、音声、またはビデオ・オブジェクトをデータベースに保管します。

ソースがバッファーまたはファイルにあるオブジェクトは、クライアント・マシンまたはサーバー・ファイルに保管できます。このようなソースの場合、そのオブジェクトをデータベース表に BLOB として保管するか、データベース・サーバーのファイルに保管することができます。

UDF の要求では、次のものを指定しなければなりません。

- 現在接続されているデータベース・サーバーの名前。これは、特殊レジスター CURRENT SERVER に入っています。
- オブジェクト内容のソース。これは、クライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルのいずれかにあります。
- 内容をデータベース表に BLOB として保管するか、ファイル・サーバーに保管するか。
- ソースの形式。
- オブジェクトとともに保管する注釈 (注釈を保管しないなら、ヌル値または空ストリング)。

4. インターレース GIF 画像にこのオプションを指定する場合、圧縮タイプ LZW も指定する必要があります。

イメージ、オーディオ、およびビデオ・エクステンダーを使用すれば、それらによってオブジェクトの形式が認識されなくても、オブジェクトを保管することができます。形式が認識されない場合には、そのオブジェクトの属性を指定する必要があります。画像やビデオを保管するときにユーザー提供の属性を指定して、サムネールを保管することもできます。サムネールとは、画像またはビデオを表すミニ・サイズの画像です。

画像の場合に限り、保管するときに、その形式を変換することができます。形式変換を要求する場合には、画像のソースとターゲットの形式を指定する必要があります。形式変換要求では、画像の切り取りや回転などの変更を追加指定することもできます。変換オプションを指定することによって、これらの変更を指示します。

**保管操作のコミット:** データベースに画像、音声、ビデオ・オブジェクトを保管したら、その作業単位をコミットしてください。これによってエクステンダーが保持するロックが解放されるので、保管されているそのオブジェクトに対して更新操作を行うことができます。

## DB2Image、DB2Audio、および DB2Video UDF の形式

DB2Image、DB2Audio、DB2Video UDF は、多重定義されます。つまり、UDF の使用方法に応じて異なる形式を持ちます。各 UDF には、以下に示す形式があります (形式内の xxxxx は Image、Audio または Video です)。

形式 1: クライアント・バッファーまたはクライアント・ファイルからオブジェクトを保管する。

```
DB2xxxxx(
     CURRENT SERVER,       /* database name name in CURRENT SERVER REGISTER */
     content,              /* object content */
     format,               /* source format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

形式 2: サーバー・ファイルからオブジェクトを保管する。

```
DB2xxxxx(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* source file name */
     format,               /* source format */
     stortype,             /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                           /* in file server*/
                           /* MMDB_STORAGE_TYPE_INTERNAL=store */
                           /* as a BLOB*/
     comment               /* user comment */
  );
```

形式 3: ユーザー提供の属性をもつオブジェクトをクライアント・バッファーまたはクライアント・ファイルから保管する。

```
DB2xxxxx(
     CURRENT SERVER,        /* database name in CURRENT SERVER REGISTER */
     content,               /* object content */
     target_file,           /* target file name for storage in file server */
                            /* or NULL for storage in table as BLOB */
     comment,               /* user comment */
     attrs,                 /* user-supplied attributes */
     thumbnail              /* thumbnail (image and video only) */
  );
```

形式 4: ユーザー提供の属性をもつオブジェクトをサーバー・ファイルから保管する。

```
DB2xxxxx(
     CURRENT SERVER,           /* database name in CURRENT SERVER REGISTER */
     source_file,              /* source file name */
     stortype,                 /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                               /* in file server*/
                               /* MMDB_STORAGE_TYPE_INTERNAL=store */
                               /* as a BLOB*/
     comment,                  /* user comment */
     attrs,                    /* user-supplied attributes */
     thumbnail                 /* thumbnail (image and video only) */
  );
```

DB2Image UDF には、さらに以下の形式があります。

形式 5: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換して保管します。

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     content,              /* object content */
     source_format,        /* source format */
     target_format,        /* target format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

形式 6: サーバー・ファイルからの画像は、以下のように形式変換して保管します。

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* server file name */
     source_format,        /* source format */
     target_format,        /* target format */
```

```
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

形式 7: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換および追加の変更をして保管します。

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    content,              /* object content */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options,   /* Conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

形式 8: サーバー・ファイルからの画像は、以下のように形式変換および追加の変更をして保管します。

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    source_file,          /* server file name */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options    /* conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

たとえば、C アプリケーションの次のステートメントでは、画像を含む行を従業員表に挿入します。ソース画像は、ajones.bmp という名前のサーバー・ファイルにあります。画像は従業員表に BLOB として保管されます。 (これは上の形式 2 に対応します。)

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
```

```
          'ASIS',                          /*keep the image format*/
          :hvStorageType                   /*store image in DB as BLOB*/
          'Anita''s picture')              /*comment */
       );
```

C アプリケーションの次のステートメントでは、前の例と同じ行を従業員表に挿入します。しかしこの場合には、画像を保管する際に、形式を BMP から GIF へ変換します。 (これは上の形式 6 に対応します。)

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
          'ASIS',                          /*source image format*/
          'GIF',                           /*target image format*/
          'Anita''s picture')              /*comment*/
       );
```

画像、音声、ビデオのオブジェクトを保管すると、その画像で使用されている色数、音声の再生時間、ビデオの圧縮形式などの属性をエクステンダーが計算します。形式が認識できないオブジェクトを保管する場合には、UDF への入力としてこれらの属性を指定する必要があります。エクステンダーは、それらの属性を他の属性 (オブジェクトの注釈やオブジェクトを保管したユーザーの識別子など) とともにデータベースに保管します。これらの属性は照会で使用することができます。

## クライアントにあるオブジェクトの保管

画像、音声、ビデオのオブジェクトの内容をクライアント・バッファーまたはクライアント・ファイルからサーバーに送信するには、ホスト変数またはファイル参照変数を使用します。

オブジェクトがクライアント・ファイルにある場合は、ファイル参照変数を使ってその内容を伝送し、サーバーに保管します。たとえば、C アプリケーション・プログラムの次のステートメントでは、Audio_file というファイル参照変数を宣言し、それを使って音声クリップ (その内容はクライアント・ファイルにある) を伝送します。オーディオ・クリップは、サーバー上のデータベース表に保管されます。ファイル参照変数の file_option フィールドは、入力用の SQL_FILE_READ に設定されています。また、ファイル参照変数が DB2Audio UDF への内容引き数として使用されています。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/sounds/ajones.wav");
```

```
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          :Audio_file,          /* file reference variable */
          'WAVE',
          CAST(NULL as LONG VARCHAR),

          'Anita''s voice')
      );
```

オブジェクトがクライアント・バッファーにある場合は、BLOB または BLOB_LOCATOR のいずれかとして定義されているホスト変数を使ってその内容を伝送し、サーバーに保管します。次の C アプリケーション・プログラム・ステートメントでは、Video_loc というホスト変数を使ってビデオ・クリップの内容を伝送し、サーバーに保管します。ビデオ・クリップは、データベース表に BLOB として保管されます。ホスト変数が DB2Video UDF へ内容の引き数として使用されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Video_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          :Video_loc,         /* host variable */
          'MPEG1',
          '',
          'Anita''s video')
       );
```

**UDF メモリーは十分とってください:** 保管するオブジェクトの内容がクライアント・バッファーにある場合は、データベース・マネージャー構成の UDF_MEM_SZ パラメーターを 4 MB 以上に設定することが必要です。UDF_MEM_SZ パラメーターは、DB2 コマンドの UPDATE DATABASE MANAGER CONFIGURATION を使用すれば更新できます。 UPDATE DATABASE MANAGER コマンドの詳細については、 *DB2 コマンド解説書* を参照してください。

### サーバーにあるオブジェクトの保管

保管する画像、音声、ビデオがサーバー・ファイルにある場合は、そのパスを UDF への内容引き数として指定する必要があります。たとえば、C アプリケーション・プログラムの次のステートメントでは、画像を含む行をデータベースに挿入します。画像の内容はサーバーのファイルにあります。保管された画像は、そのままサーバー・ファイルに残り、データベースからポイントされます。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp', /*source in server file */
          'BMP',
          :hvStorageType,
          'Anita''s picture')
       );
```

**正しいパスを指定してください:** ソースがサーバー・ファイルにあるオブジェクトを保管する場合は、ファイルの完全修飾名か、相対名を指定することができます。相対ファイル名を指定する場合には、そのファイルの正しいパスが DB2 サーバーの適切な環境変数で指定されていなければなりません。これらの環境変数の設定については、613ページの『付録A. DB2 エクステンダー用の環境変数の設定』を参照してください。

### データベースまたはファイルの記憶域の指定

画像、音声、ビデオのオブジェクトは、BLOB としてデータベース表に保管することも、サーバー・ファイルに保管することもできます。オブジェクトをサーバー・ファイルに保管すると、そのファイルがデータベースによってポイントされます。

オブジェクトをクライアント・バッファーまたはクライアント・ファイルから保管する場合には、target_file パラメーターへの指定内容によって、BLOB かサーバー・ファイル記憶域のどちらかを表します。ファイル名を指定すると、オブジェクトをサーバー・ファイルに保管することを表します。ファイルにヌル値または空ストリングを指定すると、オブジェクトを BLOB としてデータベース表に保管することを表します。 target_file パラメーターのデータ・タイ

プは LONG VARCHAR です。ヌル値を指定する場合には、それを必ず LONG VARCHAR データ・タイプにキャストしてください。

たとえば、C アプリケーション・プログラムの次のステートメントでは、画像を含む行をデータベース表に保管します。イメージ・ソースは、クライアント・バッファーにあります。画像はサーバー・ファイルに保管されます。データベース表はサーバー・ファイルを指しています。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          :Img_buf,
          'ASIS',
          '/employee/images/ajones.bmp',    /* store image in server file */
          'Anita''s picture')
       );
```

オブジェクトをサーバー・ファイルから保管する場合には、定数 MMDB_STORAGE_TYPE_INTERNAL を指定して、オブジェクトを BLOB としてデータベース表に保管します。オブジェクトは保管するが、その内容をサーバー・ファイルに残しておく場合には、定数 MMDB_STORAGE_TYPE_EXTERNAL を指定します。
MMDB_STORAGE_TYPE_INTERNAL は整数値 1 を、
MMDB_STORAGE_TYPE_EXTERNAL は整数値 0 をそれぞれもちます。

たとえば、次の C アプリケーション・プログラムでは、音声クリップがサーバー・ファイルに保管されます。ソースの音声内容はすでにサーバー・ファイルにあります。保管操作はファイル名をデータベースに置くので、SQL ステートメントによってそのファイルにアクセスできるようになります。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          '/employee/sounds/ajones.wav',
```

```
        'WAVE',
        :hvStorageType,            /* store audio in server file */
        'Anita''s voice')
    );
```

## 記憶域の形式の識別

オブジェクトを保管するときには、その形式を識別する必要があります。指定できる形式は、95ページの表5 のとおりです。エクステンダーは、画像、音声、ビデオのオブジェクトをソースと同じ形式で保管します。画像オブジェクトの場合は、保管する画像の形式をイメージ・エクステンダーで変換することができます。画像形式を変換する場合には、ソース・イメージの形式とターゲット・イメージの形式を指定する必要があります。ターゲット・イメージとは、保管された画像です。

### 変換せずに保管する場合の形式の識別

変換せずにオブジェクトを保管する場合、ソースの画像、音声、ビデオのオブジェクトの形式を指定します。たとえば、C アプリケーション・プログラムの次のステートメントでは、ビットマップ・イメージ (BMP) をデータベース表に保管します。ソースの内容はサーバー・ファイルにあります。ターゲット・イメージの形式はソースと同じです。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /*image in BMP format */
          '',
          'Anita''s picture')
       );
```

形式には、ヌル値か空ストリング、さらにイメージ・エクステンダーの場合には、文字ストリング ASIS が指定できます。その場合は、エクステンダーがソースを調べてその形式を判別します。

**認識可能な形式に対する NULL または ASIS の使用:** ヌル値、空ストリング、または ASIS を指定するのは、その形式がエクステンダーにとって認識可能な場合、つまり 95ページの表5 のリストにある形式の場合だけにしてください。そうでないと、エクステンダーがそのオブジェクトを保管することはできません。

### 形式変換を行って保管するための形式と変換オプションの指定

形式変換を行って画像を保管する場合には、ソースとターゲットの画像形式を両方とも指定します。どの変換が可能かについては、95ページの表5 を参照してください。

さらに、追加の変更 (保管する画像に適用する必要がある回転や圧縮など) を識別する変換オプションを指定することができます。各変換オプションはパラメーターおよび関連する値によって指定します。パラメーターと指定できる値は 97ページの表6 に示してあります。複数のパラメーター / 値のペアを指定することによって、保管する画像に対する複数の変更を要求することができます。

次の例では、ビットマップ (BMP) イメージ (その内容はサーバー・ファイルにある) が、データベース表に保管されるときに GIF 形式に変換されます。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '',
          'Anita''s picture')
       );
```

次の例では、上記の例からの画像が、データベース表に保管されるときに GIF 形式に変換されます。さらに、画像は保管されるときに 110 ピクセルの幅と 150 ピクセルの高さに切り取られ、LZW 圧縮を使用して圧縮されます。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '-x 110 -y 150 -c 14',          /* conversion options */
          '/employee/images/ajones.gif',
          'Anita''s picture')
       );
```

## ユーザー指定の属性をもつオブジェクトの保管

画像、音声、ビデオのオブジェクトを保管する場合、形式は、エクステンダーが解釈できるものだけには限定されません。つまり、独自の形式を指定することができます。その場合、エクステンダーはその形式を認識できませんので、

ソース・オブジェクトの属性をユーザーが指定する必要があります。属性値は、属性構造体に割り当ててください。属性構造体は、UDF の LONG VARCHAR FOR BIT DATA 変数のデータ・フィールドに保管しなければなりません。

サーバー上の UDF コードは常に、『ビッグ・エンディアン形式』 のデータを期待します。ビッグ・エンディアン形式とは、ほとんどの UNIX プラットフォームで使用される形式です。オブジェクトを 『リトル・エンディアン形式』 で保管する場合、サーバー上の UDF コードが正しく処理できるように、ユーザー指定の属性データを作成する必要があります。リトル・エンディアン形式は、Intel® および他のマイクロプロセッサー・プラットフォームで一般に使用されている形式です。 (オブジェクトをリトル・エンディアン形式で保管していないとしても、ユーザー指定の属性を作成するのは良いことです。) 画像オブジェクトの属性を作成するには DBiPrepareAttrs API を使用します。音声オブジェクトの属性を作成するには DBaPrepareAttrs API を使用します。ビデオ・オブジェクトの属性を作成するには DBvPrepareAttrs API を使用してください。

たとえば、C アプリケーション・プログラムの次のステートメントでは、画像を含む行をデータベース表に保管します。サーバー・ファイルにあるソース画像は、高さが 640 ピクセル、幅が 480 ピクセルのユーザー定義形式です。属性は画像の保管前に作成されることに注意してください。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr→format,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

DBEXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
```

```
          '/employee/images/ajones.bmp',
          :hvStorageType,
          'Anita''s picture',
          :hvImgattrs,                    /* user-specified attributes */
          CAST(NULL as LONG VARCHAR)
       );
```

C アプリケーション・プログラムの次のステートメントでは、音声クリップを含む行をデータベース表に保管します。サーバー・ファイルにある音声クリップは、サンプリング率が 44.1 kHz で、2 つのチャネルに録音されたユーザー定義形式です。オーディオ・クリップは MIDI ではないので、トラック名および楽器に対して空ストリングが指定されます。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct (
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr→cFormat,"FormatA");
paudioAttr→ulSamplingRate=44100;
paudioAttr→usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

DBaPrepareAttrs(paudioAttr);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          '/employee/sounds/ajones.aud',
          :hvStorageType,
          'Anita''s voice',
          :hvAudattr)                  /* user-specified attributes */
       );
```

## サムネール・サイズの保管 (画像とビデオのみ)

独自形式の画像を保管する場合には、**サムネール・サイズ** (画像のミニチュア版) を同時に保管することができます。サムネール・サイズの大きさと形式はユーザーが制御します。イメージ・エクステンダーが認識できる形式で画像を保管する場合には、そのオブジェクトのサムネール・サイズが自動的に生成さ

れ、保管されます。イメージ・エクステンダーは、サイズが 112 × 84 ピクセルの GIF 形式のサムネール・サイズを作成します。

独自形式のビデオ・オブジェクトを保管する場合には、ビデオ・オブジェクトを象徴するサムネール・サイズを同時に保管することができます。ビデオ・エクステンダーが認識できる形式でビデオ・オブジェクトを保管する場合には、そのオブジェクトのサムネール・サイズが自動的に生成され、保管されます。ビデオ・エクステンダーは、サイズが 108 × 78 ピクセルの GIF 形式のサムネール・サイズを作成します。

ユーザー指定属性の画像またはビデオ・オブジェクトを保管するときにサムネール・サイズを保管したくなければ、サムネール・サイズに代わってヌル値か空ストリングを指定します。

サムネール・サイズは、プログラムの中で生成してください。エクステンダーには、サムネール・サイズを生成する API はありません。サムネール・サイズの構造体をプログラムの中に作成し、その構造体を UDF に指定してください。

C アプリケーション・プログラムの次のステートメントでは、ビデオ・クリップを含む行をデータベース表に保管します。ソース・ビデオ・クリップ (その内容はサーバー・ファイルにある) の形式は、ユーザー定義形式です。ビデオの内容はサーバーに残り、表からポイントされます。代表的なビデオ・フレームのサムネール・サイズが同時に保管されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[4000];
       }hvVidattrs;
struct {
       short len;
       char data[10000];
       }hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *) hvVidattrs.data;
strcpy(pvideoAttr→cFormat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);

/* Generate thumbnail and assign data in video structure */

EXEC SQL INSERT INTO EMPLOYEE VALUES(
```

```
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.vid',
          :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
          :hvThumbnail)                    /* Thumbnail*/
       );
```

## 注釈の保管

画像、音声、ビデオのオブジェクトと一緒に注釈を保管する場合には、その注釈を UDF 要求に指定します。注釈は、データ・タイプが LONG VARCHAR の自由形式のテキストで、長さは最高 32,700 バイトです。オブジェクトの保管時に注釈を保管しないのであれば、注釈の代わりにヌル値または空ストリングを指定します。ヌル値を指定する場合には、それを必ず LONG VARCHAR データ・タイプにキャストします。

たとえば、C アプリケーション・プログラムの次のステートメントでは、ビデオ・クリップと一緒に注釈を保管します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.mpg',
          'MPEG1',
          :hvStorageType,
          'Anita''s video')                /* comment */
       );
```

C アプリケーション・プログラムの次のステートメントでは、画像を注釈なしで保管します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
```

```
    DB2IMAGE(
      CURRENT SERVER,
      '/employee/images/ajones.bmp',
      'GIF',
      :hvStorageType,
      Cast(NULL as LONG VARCHAR)            /* no comment */
    );
```

## 画像、音声、ビデオのオブジェクトの取り出し

SQL SELECT で Content UDF を使用すると、画像、音声、ビデオのオブジェクトをデータベース表から取り出すことができます。オブジェクトは、クライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルに取り出すことができます。

### 取り出しのための Content UDF 形式

Content UDF は多重定義されます。つまり、UDF の使用方法に応じて異なる形式を持ちます。形式は次のとおりです。

形式 1: クライアント・バッファーまたはクライアント・ファイルにオブジェクトを取り出す。

```
Content(
     handle,                      /* object handle */
  );
```

形式 2: クライアント・バッファーまたはクライアント・ファイルにオブジェクトのセグメントを取り出す。

```
Content(
     handle,                      /* object handle */
     offset,                      /* offset where retrieval begins */
     size                         /* number of bytes to retrieve */
  );
```

形式 3: サーバー・ファイルにオブジェクトを取り出す。

```
Content(
     handle,                    /* object handle */
     target_file,               /* server file name */
     overwrite                  /* 0=Do not overwrite target file if it exists */
                                /* 1=Overwrite target file */

  );
```

さらに、画像オブジェクトの場合には、Content UDF は次の形式をとります。

形式 4: 形式変換をしてクライアント・バッファーまたはファイルに画像を取り出す。

```
Content(
     handle,                      /* object handle */
     target format                /* target format */
       );
```

形式 5: 形式変換をしてオブジェクトをサーバー・ファイルに取り出す。

```
Content(
     handle,                 /* object handle */
     target_file,            /* server file name */
     overwrite,              /* 0=Do not overwrite target file if it exists */
                             /* 1=Overwrite target file */
     target format           /* target format */

  );
```

形式 6: 形式変換および追加の変更をして、オブジェクトをクライアント・バッファーまたはファイルに取り出す。

```
Content(
     handle,                 /* object handle */
     target format,          /* target format */
     conversion_options      /* conversion options */
  );
```

形式 7: 形式変換および追加の変更をして、オブジェクトをサーバー・ファイルに取り出す。

```
Content(
     handle,                 /* object handle */
     target_file,            /* server file name */
     overwrite,              /* 0=Do not overwrite target file if it exists */
                             /* 1=Overwrite target file */
     target format,          /* target format */
     conversion_options      /* conversion options */
  );
```

たとえば、次のステートメントでは、画像を従業員表からサーバーのファイルへ取り出します。 (これは上の形式 3 に対応します。)

```
EXEC SQL SELECT CONTENT(                /* retrieval UDF */
       PICTURE,                         /* image handle */
       '/employee/images/ajones.bmp',   /* target file */
       1)                               /* overwrite target file */
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

C アプリケーション・プログラムの次のステートメントでは、画像を従業員表からサーバーのファイルへ取り出します。取り出しの際、画像の形式が変換されます。 (これは上の形式 5 に対応します。)

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvImg_fname[255];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(                    /* retrieval UDF */
       PICTURE,                             /* image handle */
       '/employee/images/ajones.bmp',       /* target file */
       1,                                   /* overwrite target file */
       'GIF')                               /* target format */
INTO :hvImg_fname
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

## オブジェクトをクライアントに取り出す場合

Content UDF を使えば、形式変換をせずに画像、音声、ビデオのオブジェクトをクライアント・バッファーかクライアント・ファイルに取り出すことができます。さらに取り出しの際に、その画像の形式をイメージ・エクステンダーによって変換することもできます。

### 形式変換せずにオブジェクトをクライアントに取り出す場合

LOB ロケーターを使用して、画像、音声、ビデオのオブジェクトをクライアント・バッファーに取り出すか、または LOB を取り出します。画像、音声、ビデオのオブジェクトをクライアント・ファイルに取り出すには、ファイル参照変数を使用します。

オブジェクトの内容が BLOB としてデータベース表に保管されている場合、画像、音声、ビデオのオブジェクトをクライアント・バッファーまたはクライアント・ファイルへ取り出すには、ファイル参照変数が適しています。内容がサーバー・ファイルにある場合は、その内容をサーバー・ファイルからクライアント・ファイルへコピーする方が効率的かもしれません。

オブジェクトのハンドルを指定します。さらに、取り出しを開始するオフセット (バイト 1 から始まる) と、取り出すバイト数を指定することもできます。

C アプリケーション・プログラムの次のステートメントでは、audio_loc という LOB ロケーターを使って、音声クリップをクライアント・バッファーに取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        SOUND)                                  /* audio handle */
     INTO :audio_loc
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

**UDF メモリーは十分とってください:** 取り出すオブジェクトの内容がクライアント・バッファーにある場合は、データベース・マネージャー構成の UDF_MEM_SZ パラメーターを 4 MB 以上に設定することが必要です。 UDF_MEM_SZ パラメーターは、DB2 コマンドの UPDATE DATABASE MANAGER CONFIGURATION を使用すれば更新できます。 UPDATE DATABASE MANAGER コマンドの詳細については、*DB2 コマンド解説書* を参照してください。

### 変換して画像をクライアントに取り出す場合

LOB ロケーターを使用して、保管イメージを形式変換してクライアント・ファイルに取り出すか、または LOB を取り出します。保管イメージを形式変換してクライアント・ファイルに取り出すには、ファイル参照変数を使用します。

画像の内容が BLOB としてデータベース表に保管されている場合に、画像をホスト変数を使用してクライアント・バッファーに取り出したり、ファイル参照変数を使用してクライアント・ファイルに取り出すのが適しています。内容がサーバー・ファイルにある場合は、その内容をサーバー・ファイルからクライアント・ファイルへコピーする方が効率的かもしれません。

画像を形式変換して取り出す場合には、そのターゲット形式 (つまり、変換後の形式) を指定する必要があります。 95ページの表5 に可能形式変換が示されています。さらに、追加の変更 (取り出した画像に適用する回転や拡大縮小など) を識別する変換オプションを指定することができます。 97ページの表6 に指定できる変換オプションが示されています。

たとえば、C アプリケーション・プログラムの次のステートメントでは、画像をクライアント・ファイルに取り出します。ソース・イメージは、データベース表に BLOB として保管されます。取り出された画像は GIF 形式に変換され、元のサイズの 3 倍に拡大されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name, "/employee/images/ajones.gif");
Img_file.name_length= strlen(Img_file.name);
Img_file.file_options= SQL_FILE_CREATE;

EXEC SQL SELECT CONTENT(
        PICTURE,                          /* image handle */
        'GIF',                            /* target format */
        '-s 3.0')                         /* conversion options */
     INTO :Img_file,
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

### オブジェクトをサーバー・ファイルに取り出す場合

Content UDF を使えば、形式変換をせずに、画像、音声、ビデオのオブジェクトをサーバー・ファイルに取り出すことができます。さらに、Content UDF によって、画像を形式変換してサーバー・ファイルに取り出すこともできます。

画像、音声、ビデオのオブジェクトを変換せずにサーバーのファイルへ取り出す場合には、オブジェクトのハンドル、ターゲット・ファイルの名前、重ね書き標識を指定します。重ね書き標識は、ターゲット・ファイルがサーバーにすでにある場合、取り出したデータでそのファイルを重ね書きするかどうかをエクステンダーに知らせます。ターゲット・ファイルが存在しなければ、エクステンダーは、ターゲット・ファイルをサーバーに作成します。

重ね書き標識の値として 1 を指定すると、エクステンダーは、取り出したデータでターゲット・ファイルを重ね書きします。重ね書き標識の値として 0 を指定すると、エクステンダーは、ターゲット・ファイルを重ね書きしないので、データは取り出されません。

取り出すオブジェクトが BLOB としてデータベース表に保管されている場合には、重ね書き標識は無視されます。ターゲット・ファイルは、重ね書き標識の指定が何であれ、作成または重ね書きされます。

オブジェクトをサーバー・ファイルに取り出すと、そのサーバー・ファイルの名前が戻されます。たとえば、C アプリケーション・プログラムの次のステートメントでは、ビデオがサーバーのファイルに取り出されます。サーバーのファイル名は、ホスト変数 hvVid_fname に保管されます。

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
       short len;
       char data[250];
      }hvVid_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        VIDEO,                                  /* video handle */
        '/employee/videos/ajones.mpg',          /* server file */
        1)                                      /* overwrite target file */
     INTO :hvVid_fname;
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

オブジェクトが BLOB としてデータベース表に保管されている場合、そのオブジェクトを変換せずにサーバー・ファイルに取り出すには、Content UDF を

使用するのが適切です。オブジェクトがサーバー・ファイルに保管されている場合には、ソース・ファイルの内容をターゲット・ファイルにコピーする方が効率的かもしれません。

画像を形式変換してサーバーのファイルに取り出す場合には、画像ハンドル、ターゲット・ファイルの名前、ターゲットの重ね書き標識、ターゲット形式を指定します。どの変換が可能かについては、95ページの表5 を参照してください。ターゲットの形式には、ヌル値か空ストリング、またはストリング ASIS を指定することができます。この場合には、取り出された画像の形式はソースと同じになります。

たとえば、C アプリケーション・プログラムの次のステートメントでは、画像がサーバーのファイルに取り出されます。ソース・イメージはビットマップ形式で、データベース表に BLOB として保管されています。取り出された画像は GIF 形式に変換されます。サーバー・ファイルのファイル名は、ホスト変数 hvImg_fname に保管されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  struct{
        short len;
        char [400];
        }hvImg_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        PICTURE,                                /* image handle */
        '/employee/images/ajones.gif',          /* target file */
        1,                                      /* overwrite target file */
        'GIF')                                  /* target format */
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

**サーバー・ファイルがアクセス可能でなければなりません:** オブジェクトをサーバー・ファイルに取り出す場合には、そのターゲット・ファイルの完全修飾名を指定する必要があります。または、DB2IMAGEEXPORT、DB2AUDIOEXPORT、および DB2VIDEOEXPORT 環境変数を適切に設定することによって、不完全なファイル名の指定を解決できなければなりません。

## 属性の取り出し方と使用法

画像や、音声、ビデオのオブジェクトをデータベースに保管すると、エクステンダーによって、そのオブジェクトの属性も同時にデータベースに保管されます。オブジェクトを更新すると、エクステンダーはそのデータベースに保管されているオブジェクトの属性も更新します。これらの属性は、照会で使用することができます。

## 属性の使用法

エクステンダーは、ユーザーが管理する属性ごとに UDF を作成します。したがって、SQL ステートメントに UDF を指定することによって、オブジェクト属性をアクセスし使用することができます。 表7 に、エクステンダーが使用する属性とその UDF を示します。さらに、各属性に対するオブジェクト・タイプも示します。オブジェクトの形式やファイル名など、一部の属性は、すべてのオブジェクト・タイプに共通です。これらの属性は、画像、音声、およびビデオのオブジェクトに対応しています。サンプリング率や圧縮タイプなどの属性は、音声やビデオなどの特定のオブジェクト・タイプに固有のものです。

*表 7. DB2 エクステンダーによって管理される属性.* これらの属性にアクセスするには、それぞれの UDF を使用します。

| 属性 | UDF | イメージ | オーディオ | ビデオ |
|---|---|---|---|---|
| オブジェクトが保管されているサーバー・ファイルの名前 | Filename | o | o | o |
| オブジェクトを保管した人のユーザー ID | Importer | o | o | o |
| オブジェクトが保管された日付と時刻 | ImportTime | o | o | o |
| オブジェクトの大きさ (バイト単位で) | Size | o | o | o |
| オブジェクトを最後に更新した人のユーザー ID | Updater | o | o | o |
| オブジェクトが最後に更新された日付と時刻 | UpdateTime | o | o | o |
| オブジェクトの形式 (たとえば、GIF や MPEG1) | Format | o | o | o |
| オブジェクトの注釈 | Comment | o | o | o |
| オブジェクトの高さ (ピクセル単位で) | Height | o | | o |
| オブジェクトの幅 (ピクセル単位で) | Width | o | | o |
| オブジェクトの色数 | NumColors | o | | |
| オブジェクトのサムネール・サイズの画像 | Thumbnail | o | | o |
| 音声のサンプルごと、またはビデオの音声トラックのサンプルごとの、戻されるバイト数 | AlignValue | | o | o |

*表 7. DB2 エクステンダーによって管理される属性 (続き).* これらの属性にアクセスするには、それぞれの UDF を使用します。

| 属性 | UDF | イメージ | オーディオ | ビデオ |
|---|---|---|---|---|
| 各サンプルを表すのに必要なビット数 | BitsPerSample | | o | o |
| 記録されているチャネルの数 | NumChannels | | o | o |
| 所要時間 (秒単位で) | Duration | | o | o |
| サンプリング率 (秒あたりのサンプル数で) | SamplingRate | | o | o |
| 秒あたりの平均転送バイト数 | BytesPerSec | | o | |
| 楽器の音声トラック番号 | FindInstrument | | o | |
| 指定されたトラックのトラック番号 | FindTrackName | | o | |
| 録音されている楽器の名前 | GetInstruments | | o | |
| 録音されている楽器のトラック番号と名前 | GetTrackNames | | o | |
| 音声の秒あたりのクロック・ティック | TicksPerSec | | o | |
| 音声の四分音符あたりのクロック・ティック | TicksPerQNote | | o | |
| 縦横比 | AspectRatio | | | o |
| ビデオ圧縮形式 (MPEG1 など) | CompressType | | | o |
| 秒あたりのスループット・フレーム数 | FrameRate | | | o |
| 最大スループット (秒あたりのバイト数で) | MaxBytesPerSec | | | o |
| 音声トラックの数 | NumAudioTracks | | o | o |
| フレームの数 | NumFrames | | | o |
| ビデオ・トラックの数 | NumVideoTracks | | | o |

属性 UDF は、SQL ステートメントの SELECT 文節の式か、WHERE 文節の探索条件で使用することができます。 UDF を指定する場合には、そのオブジェクトのハンドルが入っている列 (データベース表の) の名前を指定する必要があります。

たとえば、次のステートメントでは、Updater UDF を SQL SELECT ステートメントの SELECT 文節に指定することによって、従業員表にある画像を最後に更新した人のユーザー ID を取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdatr[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(PICTURE)
     INTO :hvUpdatr
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

次のステートメントでは、Filename UDF を SELECT ステートメントの SELECT 文節に、NumAudioTracks UDF を WHERE 文節に指定することによって、従業員表に保管されているビデオから、音声トラックがあるものを取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO)>0;
```

### 注釈の取り出し

Comment UDF を使用して、画像、音声、ビデオのオブジェクトと一緒に保管されている注釈を取り出します。オブジェクトの注釈を取り出すときには、そのオブジェクトのハンドルが入っている列 (データベース表の) を指定します。たとえば、次のステートメントでは、音声クリップとともに従業員表に保管されている注釈を取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[32700];
       }hvComment
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT COMMENT(SOUND)
     INTO :hvComment
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

また、Comment UDF は、SQL 照会の WHERE 文節に述部として使用することもできます。たとえば、次のステートメントでは、『touched up』 という注記のあるすべての画像のファイル名を従業員表から取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[250];
       }hvImg_fname
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
           LIKE '%touch%up';
```

## 画像、音声、ビデオのオブジェクトの更新

SQL UPDATE ステートメントで Content UDF を使用すると、データベース表の画像、音声、ビデオのオブジェクトを更新することができます。データベース表の画像、音声、ビデオのオブジェクトを更新し、そのオブジェクトに対応する注釈を更新するには、SQL UPDATE ステートメントに Replace UDF を指定します。どちらの場合にも、そのオブジェクトに対応する属性が更新されます。

データベース表に BLOB として保管されているオブジェクトや、サーバー・ファイルに保管されている (そしてデータベースからポイント指定されている) オブジェクトが更新可能です。更新のソースは、バッファー、クライアント・ファイル、またはサーバー・ファイルのいずれかにすることができます。

95ページの表5 は、画像、音声、ビデオのオブジェクトを更新できる形式を示しています。しかし、エクステンダーによって形式を認識できないオブジェクトを更新することもできます。この場合には、オブジェクトの属性は、そのオブジェクトが保管されたときにユーザーが指定しました。ユーザー指定の属性をもつオブジェクトを更新する場合には、そのオブジェクトの更新属性を指定する必要があります。 SQL UPDATE 内の ContentA UDF を使用して、データベース表の画像、音声、またはビデオをユーザー指定の属性で更新することができます。 SQL UPDATE ステートメントの ReplaceA UDF を使用して、データベース表の画像、音声、またはビデオをユーザー指定の属性で更新したり、そのオブジェクトに関連する注釈を更新することもできます。ユーザー指定の属性でオブジェクトを更新する際には、オブジェクトの属性、形式、圧縮形式 (ビデオ・オブジェクトの場合のみ) を指定する必要があります。

さらに、保管されている画像やビデオのサムネール・サイズを更新することもできます。

**保管操作のコミット:** データベースの画像、音声、またはビデオのオブジェクトを更新したら、その作業単位をコミットしてください。これによってエクステンダーが保持するロックが解放されるので、保管されたそのオブジェクトに対してさらに更新操作を行うことができます。

## 更新のための Content UDF 形式

Content UDF は多重定義されます。つまり、UDF の使用方法に応じて異なる形式を持ちます。形式は次のとおりです。

形式 1: クライアント・バッファーまたはクライアント・ファイルからオブジェクトを更新する。

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_file                /* target file name for storage in file */
                                /* server or NULL for storage in table as BLOB */
  );
```

形式 2: サーバー・ファイルからオブジェクトを更新する。

```
Content(
     handle,                      /* object handle */
     source_file,                 /* server file name */
     source_format,               /* source format */
     stortype                     /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                  /* in file server */
                                  /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
  );
```

形式 3: ユーザー提供の属性をもつオブジェクトをクライアント・バッファーまたはクライアント・ファイルから更新する。

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     target_file,               /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail (image and video only) */
  );
```

形式 4: ユーザー提供の属性をもつオブジェクトをサーバー・ファイルから更新する。

```
Content(
     handle,                    /* object handle */
     source_file,               /* source file name */
     stortype,                  /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store */
```

```
                                /* as a BLOB*/
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail (image and video only) */
  );
```

画像オブジェクトでは、Content UDF はさらに 2 つの形式をとります。

形式 5: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換して更新する。

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     source format,             /* source format */
     target format,             /* target format */
     target_file                /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
  );
```

形式 6: サーバー・ファイルからのオブジェクトを形式変換して更新する。

```
Content(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source format,             /* source format */
     target format,             /* target format */
     target_file                /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
  );
```

形式 7: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換および追加の変更をして更新する。

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     source format,             /* source format */
     target format,             /* target format */
     conversion_options,        /* conversion options */
     target_file                /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
  );
```

形式 8: サーバー・ファイルからのオブジェクトは、以下のように形式変換および追加の変更をして更新する。

```
Content(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source format,             /* source format */
     target format,             /* target format */
     conversion_options,        /* conversion options */
     target_file                /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
  );
```

たとえば、C アプリケーション・プログラムの次のステートメントでは、従業員表の画像が更新されます。この更新のソースの内容は ajones.bmp という名前

のサーバー・ファイルにあります。更新された画像は従業員表に BLOB として保管されます。 (これは上の形式 2 に対応します。)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,                          /*image handle*/
                '/employee/newimg/ajones.bmp',    /*source file */
                'ASIS',                           /*keep the image format*/
                '');                              /*store image in DB as BLOB*/
   WHERE NAME='Anita Jones';
```

C アプリケーションの次のステートメントでは、前の例と同じ画像が更新されます。しかしこの場合には、更新の際に、形式が BMP から GIF へ変換されます。 (これは上の形式 6 に対応します。)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,                          /*image handle*/
                '/employee/newimg/ajones.bmp',    /*source file */
                'BMP',                            /*source format*/
                'GIF',                            /*target format*/
                '');                              /*store image in DB as BLOB*/
   WHERE NAME='Anita Jones';
```

### 更新のための Replace UDF 形式

Replace UDF は多重定義されます。つまり、UDF の使用方法に応じて異なる形式を持ちます。形式は次のとおりです。

形式 1: クライアント・バッファーまたはクライアント・ファイルからのオブジェクトを更新し、その注釈を更新する。

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_file,               /* target file name for storage in file */
     comment                    /* user comment */
  );
```

形式 2: サーバー・ファイルからのオブジェクトを更新し、その注釈を更新する。

```
Replace(
     handle,                  /* object handle */
     source_file,             /* server file name */
     source_format,           /* source format */
     stortype,                /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                              /* in file server*/
                              /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                  /* user comment */
  );
```

形式 3: ユーザー提供の属性をもつオブジェクトをクライアント・バッファーまたはクライアント・ファイルから置換する。

```
Replace(
     handle,                      /* object handle */
     content,                     /* object content */
     target_file,                 /* target file name for storage in file */
                                  /* or NULL for storage in table as BLOB */
     comment,                     /* user comment */
     attrs,                       /* user-supplied attributes */
     thumbnail                    /* thumbnail */
  );
```

形式 4: ユーザー提供の属性をもつオブジェクトをサーバー・ファイルから保管する。

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     stortype,                  /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment,                   /* user comment */
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail */
  );
```

画像オブジェクトでは、Replace UDF はさらに 2 つの形式をとります。

形式 5: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換して更新し、注釈を更新する。

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_format,             /* target format */
     target_file,               /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
     comment                    /* user comment */
  );
```

形式 6: サーバー・ファイルからのオブジェクトは、以下のように形式変換して更新し、注釈を更新する。

```
Replace(
     handle,                     /* object handle */
     source_file,                /* server file name */
     source_format,              /* source format */
     target_format,              /* target format */
     target_file,                /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                 /* in file server */
                                 /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                     /* user comment */
  );
```

形式 7: クライアント・バッファーまたはクライアント・ファイルからの画像は、以下のように形式変換および追加の変更をして更新し、注釈を更新する。

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_format,             /* target format */
     conversion_options,        /* conversion options */
     target_file,               /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
     comment                    /* user comment */
  );
```

形式 8: サーバー・ファイルからのオブジェクトを形式変換して更新し、追加の変更を行い、注釈を更新する。

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     target_format,             /* target format */
     conversion_options,        /* conversion options */
     target_file,               /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                    /* user comment */
  );
```

たとえば、C アプリケーションの次のステートメントでは、従業員表の音声クリップを更新し、それに対応する注釈を更新します。この更新のソースの内容は ajones.wav という名前のサーバー・ファイルにあります。更新された音声クリップは、形式変換されずに従業員表に BLOB として保管されます (オーディオ・エクステンダーは形式変換をサポートしません)。これは上の形式 2 に対応します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=REPLACE(
                SOUND,                          /*audio handle*/
                '/employee/newaud/ajones.wav', /*source file */
                'WAV',                          /*keep the audio format*/
                :hvStorageType,                 /*store audio in DB as BLOB*/
                'Anita''s new greeting')        /*user comment*/
   WHERE NAME= 'Anita Jones';
```

次の例では、画像とそれに対応する注釈が更新されます。この更新のソース内容はサーバー・ファイルにあります。更新された画像は従業員表に BLOB として保管され、更新の際、その形式が BMP から GIF に変換されます。 (これは上の形式 6 に対応します。)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                PICTURE,                          /*image handle*/
                '/employee/newimg/ajones.bmp',    /*source file */
                'BMP',                            /*source format*/
                'GIF',                            /*target format*/
                ''                                /*store image in DB as BLOB*/
                'Anita''s new picture')
   WHERE NAME='Anita Jones';                      /* user comment */
```

## オブジェクトをクライアントから更新する

画像、音声、ビデオのオブジェクトをクライアント・バッファーまたはクライアント・ファイルから更新するには、ホスト変数かファイル参照変数を使用します。

更新のソースがクライアント・ファイルにある場合は、ファイル参照変数を使って内容を伝送します。たとえば、C アプリケーション・プログラムの次のステートメントでは、Audio_file というファイル参照変数を定義し、それを使ってデータベース表に BLOB として保管されている音声クリップを更新します。この更新のソースはクライアント・ファイルにあります。ファイル参照変数の file_options フィールドは、SQL_FILE_READ、つまり入力用に設定されています。また、ファイル参照変数が Content UDF への内容引き数として使用されています。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/newsound/ajones.wav");
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                SOUND,
                :Audio_file                 /*file reference variable*/
                'WAVE',                     /*keep the image format*/
                CAST(NULL as LONG VARCHAR))

   WHERE NAME='Anita Jones';
```

オブジェクトがクライアント・バッファーにある場合は、ホスト変数を使ってその内容を伝送し、更新します。次の C アプリケーション・プログラムの例

では、Video_seg というホスト変数を使ってビデオ・クリップの内容を伝送し、更新します。そのビデオ・クリップに対応する注釈も更新されます。ビデオ・クリップは、データベース表に BLOB として保管されます。ホスト変数が Replace UDF へ内容の引き数として使用されています。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Video_seg
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                :Video_seg                  /*host variable*/
                'MPEG1',
                CAST(NULL as LONG VARCHAR),

                'Anita''s new video')
   WHERE NAME='Anita Jones';
```

**UDF メモリーは十分とってください:** 更新するオブジェクトの内容がクライアント・バッファーにある場合は、データベース・マネージャー構成の UDF_MEM_SZ パラメーターを 4 MB 以上に設定することが必要です。UDF_MEM_SZ パラメーターは、DB2 コマンドの UPDATE DATABASE MANAGER CONFIGURATION を使用すれば更新できます。

### オブジェクトをサーバーから更新する

画像、音声、ビデオのオブジェクトの更新用ソース内容がサーバー・ファイルにある場合には、そのファイル・パスを UDF への内容引き数として指定します。たとえば、C アプリケーション・プログラムの次のステートメントでは、データベースの画像が更新されます。画像の内容は、サーバー・ファイルにあります。データベースはサーバー・ファイルを指しています。この更新のソースもサーバー・ファイルにあります。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,                              /* image handle */
                '/employee/newimg/ajones.bmp',        /* source file */
                'ASIS',
                :hvStorageType)
   WHERE NAME='Anita Jones';
```

**正しいパスを指定してください:** 更新するオブジェクトのソースがサーバー・ファイルにある場合は、ファイルの完全修飾名か、相対名を指定することができます。相対ファイル名を指定する場合には、そのファイルの正しいパスが DB2 サーバーの適切な環境変数で指定されていなければなりません。これらの環境変数の設定については、613ページの『付録A. DB2 エクステンダー用の環境変数の設定』を参照してください。

## データベースまたはファイル記憶域を指定した更新

データベース表に BLOB として保管されている画像、音声、またはビデオのオブジェクトや、サーバー・ファイルにある (そしてデータベースからポイント指定されている) オブジェクトが更新可能です。

オブジェクトをクライアント・バッファーまたはクライアント・ファイルから保管する場合には、filename パラメーターでの指定内容によって、BLOB かサーバー・ファイル記憶域のどちらかを表します。ファイル名を指定すると、更新されるオブジェクトの内容はサーバー・ファイルにあるものとみなされます。ファイル名にヌル値を指定すると、更新されるオブジェクトは、データベース表に BLOB として保管されているものとみなされます。

たとえば、C アプリケーション・プログラムの次のステートメントでは、更新される画像の内容はサーバー・ファイルにあります。更新ソースはクライアント・バッファーにあります。画像の注釈も同時に更新されます。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                  PICTURE,
                  :Img_buf,
                  'ASIS',
                  '/employee/newimg/ajones.bmp',      /*update image in*/
                                                      /*server file*/
                  'Anita''s new picture')
   WHERE NAME='Anita Jones';
```

サーバー・ファイルにあるオブジェクトを更新する場合には、MMDB_STORAGE_TYPE_INTERNAL を指定して、データベース表に BLOB として保管されているオブジェクトを更新します。内容がサーバー・ファイルにあるオブジェクトを更新するには、MMDB_STORAGE_TYPE_EXTERNAL と指定します。

たとえば、次の C アプリケーション・プログラムでは、音声クリップが更新されます。音声クリップの内容はサーバー・ファイルにあります。この更新のソースもサーバー・ファイルにあります。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                SOUND,
                '/employee/newimg/ajones.wav',
                'WAVE',
                :hvStorageType)        /*update audio in server file*/
   WHERE NAME='Anita Jones';
```

## 更新のための形式の識別

オブジェクトを更新する場合には、その形式を識別する必要があります。エクステンダーは、更新された画像、音声、ビデオのオブジェクトをソースと同じ形式で保管します。画像オブジェクトの場合は、更新された画像の形式をイメージ・エクステンダーで変換することができます。画像の形式を変換する場合には、更新ソースの形式とターゲット・イメージの形式を指定する必要があります。ターゲット・イメージとは、保管された更新画像です。

### 変換せずに更新する場合の形式の識別

変換せずにオブジェクトを更新する場合、ソースの画像、音声、ビデオのオブジェクトの形式を指定します。たとえば、C アプリケーション・プログラムの次のステートメントでは、更新されるビットマップ (BMP) イメージの内容はサーバー・ファイルにあります。更新された画像の形式は変換されません。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                'BMP',                          /*image format*/
                '')
   WHERE NAME='Anita Jones';
```

形式には、ヌル値か空ストリング、さらにイメージ・エクステンダーの場合には、文字ストリング ASIS が指定できます。その場合は、エクステンダーがソースを調べてその形式を判別します。

**認識可能な形式に対する NULL または ASIS の使用:** ヌル値、空ストリング、または ASIS を指定するのは、その形式がエクステンダーにとって認識可

能な場合、つまり 95ページの表5 のリストにある形式の場合だけにしてください。さもないと、エクステンダーはそのオブジェクトを更新できません。

### 形式変換を行って更新するための形式と変換オプションの指定

形式変換を行って画像を更新する場合には、ソースとターゲットのイメージ形式を両方とも指定します。どの変換が可能かについては、95ページの表5 を参照してください。

さらに、追加の変更 (更新する画像に適用する必要がある回転や圧縮など) を識別する変換オプションを指定することができます。各変換オプションはパラメーターおよび関連する値によって指定します。パラメーターと指定できる値は 97ページの表6 に示してあります。複数のパラメーター / 値のペアを指定することによって、更新する画像に対する複数の変更を要求することができます。

次の例では、内容がサーバー・ファイルにある画像が更新されます。この更新のソースはビットマップ (BMP) 形式です。その形式は、更新の際 BMP から GIF に変換されます。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                'BMP',                                  /*source format*/
                'GIF',                                  /*target format*/
                '')
   WHERE NAME='Anita Jones';
```

次の例では、更新時に同じ画像を GIF 形式に変換します。さらに、画像を更新時に 90 度右回りに回転させます。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                'BMP',                                  /*source format*/
                'GIF',                                  /*target format*/
                '-r 1',                                 /* conversion options */
                '')
   WHERE NAME='Anita Jones';
```

## ユーザー指定の属性をもつオブジェクトの更新

ユーザー指定の属性で保管された画像、音声、ビデオのオブジェクトを更新する場合には、更新内容の属性を指定する必要があります。属性値は、属性構造体に割り当ててください。属性構造体は、UDF の LONG VARCHAR FOR BIT DATA 変数のデータ・フィールドに保管しなければなりません。

サーバー上の UDF コードは常に、『ビッグ・エンディアン形式』 のデータを期待します。ビッグ・エンディアン形式とは、ほとんどの UNIX プラットフォームで使用される形式です。オブジェクトを 『リトル・エンディアン形式』 で保管する場合、サーバー上の UDF コードが正しく処理できるように、ユーザー指定の属性データを作成する必要があります。リトル・エンディアン形式は、Intel および他のマイクロプロセッサー・プラットフォームで一般に使用されている形式です。 (オブジェクトをリトル・エンディアン形式で保管していないとしても、ユーザー指定の属性を作成するのは良いことです。) 画像オブジェクトの属性を作成するには DBiPrepareAttrs API を使用します。音声オブジェクトの属性を作成するには DBaPrepareAttrs API を使用します。ビデオ・オブジェクトの属性を作成するには DBvPrepareAttrs API を使用してください。

たとえば、C アプリケーション・プログラムの次のステートメントでは、更新される画像の内容はサーバー・ファイルにあります。その画像は、高さが 640 ピクセル、幅が 480 ピクセルのユーザー定義形式です。属性は画像の更新前に作成されることに注意してください。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr→Format,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                :hvStorageType,
                'Anita''s new picture',
                :ImgAttrs,                      /*user-supplied attributes*/
                CAST(NULL as LONG VARCHAR))
   WHERE NAME='Anita Jones';
```

## サムネール・サイズの更新 (画像とビデオのみ)

画像やビデオのオブジェクト用に保管されたサムネール・サイズを更新するには (または、保管された画像やビデオに対応するサムネール・サイズがない場合、それを追加するには)、Thumbnail UDF を使用します。 Thumbnail UDF を使用する場合には、更新されるサムネール・サイズのオブジェクトのハンドルと、更新される (または新規の) サムネール・サイズの内容を指定します。

サムネール・サイズは、プログラムの中で生成してください。エクステンダーには、サムネール・サイズを生成する API はありません。更新するサムネール・サイズの大きさと形式はユーザーが制御します。サムネール・サイズの構造体をプログラムの中に作成し、その構造体を UDF に指定してください。

たとえば、C アプリケーション・プログラムの次のステートメントでは、保管されたビデオ・クリップに対応するサムネール・サイズを更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;

/*Create thumbnail and store in hvThumbnail*/

EXEC SQL UPDATE employee
   SET picture=Thumbnail(
                 picture,
                 :hvThumbnail)
   WHERE name='Anita Jones';
```

ユーザー指定属性で画像またはビデオ・オブジェクトを更新する際に、サムネールを更新することもできます。ユーザー指定の属性をもつ画像やビデオを更新する場合には、入力としてサムネール・サイズを指定しなければなりません。それらのオブジェクトを更新する際にサムネール・サイズを更新したくなければ、サムネール・サイズの指定の代わりに、ヌル値か空ストリングを指定してください。

C アプリケーション・プログラムの次のステートメントでは、ユーザー指定の属性をもつビデオ・クリップと、そのビデオに対応するサムネール・サイズを更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
   struct {
       short len;
       char data[400];
   }hvVidattrs;
   struct {
```

```
        short len;
        char data[10000];
    }hvThumbnail;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

MMDBVideoAttrs         *pvideoAttr;
pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr→cformat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);

/* Update video content and thumbnail */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/employee/newvid/ajones.mpg',
                :hvStorageType,
                'Anita''s new video',
                :VidAttrs,
                :hvThumbnail)              /*thumbnail*/
   WHERE NAME='Anita Jones';
```

## 注釈の更新

注釈は、それ自体で更新することもできますし、それに対応するオブジェクトを更新するときに更新することもできます。

注釈をそれ自体で更新するときには、Comment UDF を使用します。この場合、更新される注釈の内容と、そのオブジェクトのハンドルをもつ表の列を指定します。その内容をサーバーに伝送するには、ホスト変数を使用します。たとえば、次のステートメントでは、hvRemarks というホスト変数を宣言し、それを使用して、保管されたビデオ・クリップの既存の注釈を更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
  struct {
         short len;
         char data [40];
  }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';

/* Append to old comment */

hvRemarks.data[Remarks.len]='¥0';
hvRemarks.len=strlen(hvRemarks.data);
```

```
strcat (hvRemarks.data, "Updated video");
EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME = 'Anita Jones';
```

注釈の更新を、それに対応するオブジェクトを更新するときに行うには、Replace UDF を使用します。たとえば、次のステートメントでは、サーバー・ファイルに保管されたビデオ・クリップを更新するとともに、それに対応する注釈も更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/employee/newvid/ajones.mpg',
                'MPEG1',
                :hvStorageType,
                'Anita''s new video')        /*updated comment*/
   WHERE NAME='Anita Jones';
```

## 更新

# 第12章 画像、音声、ビデオのオブジェクトを表示または再生

この章では、DB2 エクステンダーのアプリケーション・プログラミング・インターフェースを使って、データベースに保管された画像、音声、ビデオのオブジェクトの表示や再生をどのように行うかについて説明します。

## 表示 / 再生 API の使用

エクステンダー API を使用すれば、データベースに保管された音声やビデオ・フレームを表示することができます。画像やビデオ・フレームは、サムネールで表示することもできますし、元のサイズで表示することもできます。さらに、エクステンダー API を使用すれば、データベースに保管された音声やビデオのオブジェクトを再生することもできます。

オブジェクトの表示や再生には、次の API を使用します。

| API | 機能 |
|---|---|
| DBiBrowse | 画像またはビデオ・フレームを表示する |
| DBaPlay | 音声クリップを再生する |
| DBvPlay | ビデオ・クリップを再生するか、ビデオ・フレームを表示する |

これらの API のどれかを使用する場合には、次の項目を指定する必要があります。

- 表示または再生プログラムの名前
- 表示または再生するオブジェクトがデータベース表に BLOB として保管されているのか、その表からポイント指定されるファイルにあるのか
- ソース・ファイルの名前、またはデータベース表に保管されているハンドル
- ユーザーが表示や再生のプログラムをクローズするまでアプリケーションの処理を待機させるかどうか

### 表示または再生プログラムの識別

使用したい画像ブラウザーや、オーディオ・プレーヤー、ビデオ・プレーヤーのプログラム名を指定します。名前の後ろには %s を付けてください。エクステンダーがオブジェクト内容をもつファイルで %s を置き換えます。たとえ

ば、C アプリケーション・プログラムの次のステートメントでは、OS/2 イメージ・ブラウザー (ib) を開始し、画像を表示します。

```
rc = DBiBrowse(
     "ib %s",                    /* image display program */
     MMDB_PLAY_FILE,
     "/employee/images/ajones.bmp",
     MMDB_PLAY_NO_WAIT
     );
```

特定の表示や再生のプログラムを指定する代わりに、ヌル値を指定することもできます。この場合、エクステンダーは DB2IMAGEBROWSER、DB2AUDIOPLAYER、または DB2VIDEOPLAYER 環境変数に指定されている省略時の画像ブラウザー、オーディオ・プレーヤー、ビデオ・プレーヤーを開始します。 DB2 エクステンダーが環境変数をどのように使用するかについては、613ページの『付録A. DB2 エクステンダー用の環境変数の設定』を参照してください。

たとえば、C アプリケーション・プログラムの次のステートメントでは、DB2AUDIOPLAYER 環境変数に指定された省略時のオーディオ・プレーヤーを開始します。

```
rc = DBaPlay(
     NULL,                                /* use default audio player */
     MMDB_PLAY_FILE,
     "/employee/sounds/ajones.wav",
     MMDB_PLAY_NO_WAIT
     );
```

**環境変数でプログラムを指定する必要があります:** (ヌル値を指定することによって) 省略時の表示 / 再生プログラムを要求する場合には、適切な環境変数によって、表示または再生プログラムを指定しておかなければなりません。プログラムを指定していないと、その API でエラー・コードが戻されます。

## BLOB またはファイル内容の指定

表示や再生が可能なオブジェクトは、データベース表に BLOB として保管されているオブジェクト、またはその内容がファイルに保管され、データベース表からポイント指定されているオブジェクトです。オブジェクトが BLOB として保管されている場合には、MMDB_PLAY_HANDLE を、オブジェクトの内容がファイルに保管されている場合には、MMDB_PLAY_FILE をそれぞれ指定します。 MMDB_PLAY_HANDLE と MMDB_PLAY_FILE は、エクステンダーによって定義された定数です。

たとえば、C アプリケーション・プログラムの次のステートメントでは、内容がファイルにあるビデオを再生します。

```
rc = DBvPlay(
     "explore %s",
     MMDB_PLAY_FILE,                    /* content in file */
     "/employee/videos/ajones.mpg",
     MMDB_PLAY_NO_WAIT
     );
```

一般に、表示や再生のプログラムに対する入力は、ファイルから行われます。MMDB_PLAY_FILE を指定すると、エクステンダーは環境変数の値を使用して、ファイルの相対ファイル名およびパスを解決します。それから、エクステンダーはブラウズ・プログラムを開始して、それにファイル名を渡します。MMDB_PLAY_HANDLE を指定すると、エクステンダーはハンドルからファイル名を抽出します (ファイル名がヌル値でない場合)。ハンドルのファイル名が空である場合、オブジェクトは BLOB として保管されます。エクステンダーは、一時ファイルをクライアントに作成し、オブジェクトの内容をデータベース表からそのクライアント・ファイルへコピーします。次に、エクステンダーはプログラムを開始し、その内容をもつファイル (または一時ファイル) の名前をそれに渡します。

たとえば、C アプリケーション・プログラムの次のステートメントでは、BLOB として保管されている画像のハンドルを入手し、そのハンドルを使って画像を表示します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl[251];
EXEC SQL END DECLARE SECTION;

rc = DBiBrowse(
     "ib %s",
     MMDB_PLAY_HANDLE,                  /* content is BLOB */
     hvImg_hdl,
     MMDB_PLAY_NO_WAIT
     );
```

**内容はアクセス可能でなければなりません:** オブジェクトの内容は、表示や再生のプログラムからアクセス可能でなければなりません。内容がサーバー・ファイルにあるが、そのプログラムではそれがクライアントになければならない場合には、そのファイルをクライアント・ファイルへコピーするか、Content UDF を使用します。その内容が BLOB として保管されている場合は、エクステンダーがそれをクライアントへ自動的に取り出します。

## 待ち標識の指定

アプリケーションが処理を続ける前にユーザーが表示や再生のプログラムを終わるまで (つまり、DBiBrowse、DBaPlay、または DBvPlay API がコードを戻すまで)、アプリケーション・プログラムを待機させるかどうかを指定すること

ができます。アプリケーション・プログラムの処理を待機させたい場合は MMDB_PLAY_WAIT を、待機させたくない場合は MMDB_PLAY_NO_WAIT をそれぞれ指定します。 MMDB_PLAY_WAIT と MMDB_PLAY_NO_WAIT は、エクステンダーによって定義された定数です。

MMDB_PLAY_WAIT を指定すると、表示や再生のプログラムは、アプリケーション・プログラムと同じスレッドまたはプロセスで実行されます。MMDB_PLAY_NO_WAIT を指定すると、表示や再生のプログラムは、アプリケーション・プログラムから独立した独自のスレッドまたはプロセスで実行されます。

たとえば、次のステートメントを実行すると、アプリケーション・プログラムは、ユーザーが画像ブラウザーをクローズするのを待ってからアプリケーションの処理を続けます。

```
rc = DBiBrowse(
     "explore %s",
     MMDB_PLAY_FILE,
     "/employee/images/ajones.bmp",
     MMDB_PLAY_WAIT                    /* wait for browser to close */
     );
```

**DBxPlay と MMDB_PLAY_NO_WAIT を指定する場合には注意が必要です:** DBaPlay または DBvPlay を実行すると、次のいずれかの条件が当てはまる場合、エクステンダーは一時ファイルを作成します。

- オブジェクトが BLOB として保管されている。
- 相対ファイル名が環境変数の値を使用して解決できない。
- クライアント・マシンでファイルにアクセスできない。

一時ファイルは、TMP 環境変数によって指定されるディレクトリーに作成されます。 MMDB_PLAY_WAIT を指定すると、エクステンダーは、オブジェクトの再生後、その一時ファイルを削除します。しかし、MMDB_PLAY_NO_WAIT を指定すると、一時ファイルは削除されません。したがって、一時ファイルを自分で削除する必要があります。

## サムネール・サイズの画像またはビデオ・フレームの表示

サムネールは、ミニチュア版の保管画像や保管ビデオ・フレームです。画像をデータベースに保管すると、イメージ・エクステンダーがその画像のサムネールを作成し、それを属性表に保管します。ビデオをデータベースに保管すると、ビデオ・エクステンダーは属性表に、そのビデオ・オブジェクトを表す汎用サムネールを保管します。

省略時では、イメージ・エクステンダーによって自動的に作成される画像のサムネールのサイズは、約 112 × 84 ピクセルです。ビデオ・エクステンダーが挿入する汎用ビデオ・サムネールは、108 × 78 ピクセルです。画像サムネールおよび汎用ビデオ・サムネールは、どちらも GIF 形式で保管されます。これらは、画像やビデオ・フレームのデータ密度にもよりますが、およそ 4.5 KB から 5 KB のデータに相当します。ユーザー指定の属性をもつ画像やビデオを保管または更新する場合には、サムネールの大きさと形式を指定することができます。

サムネールをデータベースから取り出すには、SQL SELECT ステートメントで Thumbnail UDF を使用します。そして、そのサムネールをファイルに伝送するには、ファイル参照変数を使用します。 UDF を指定するときには、画像かビデオのハンドルが入っている列 (データベース表の) の名前を指定する必要があります。次に DBiBrowse API を使用すれば、画像やビデオ・フレームのサムネールを表示することができます。

たとえば、次のステートメントでは、サムネールの画像を取り出して、それを表示します。

```
long rc, outCount;
char Thumbnail_filename[254];
FILE *file_handle;

EXEC SQL BEGIN DECLARE SECTION;
  struct {
         short len
         char data[10000];
         }Thumbnail_buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :Thumbnail_buffer
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';

strcpy (Thumbnail_filename,"/tmp/ajones.tmb");
file_handle=fopen(Thumbnail_filename,"wb+");
outCount=fwrite(Thumbnail_buffer.data, 1, Thumbnail_buffer.len, file_handle);
fclose(file_handle);
rc = DBiBrowse (
               NULL,                          /* use the default display program */
               MMDB_PLAY_FILE,                /* thumbnail image in file */
               Thumbnail_filename,            /* thumbnail image content */
               MMDB_PLAY_WAIT);               /* wait for user to finish */
```

## フルサイズの画像やビデオ・フレームの表示

DBiBrowse API を使用して、データベース表に保管されている画像を表示します。この API の使用法の詳細については、137ページの『表示 / 再生 API の使用』を参照してください。

## 画像の表示

フルサイズのビデオ・フレームを入手するには、DBvGetNextFrame API か DBvSeekFrame API を使用します。フレームは YUV 形式でバッファーに保管されるので、DBvFrameDatato24BitRGB API を使って RGB 形式に変換することができます。次に、変換後のフレームにヘッダーを追加し (たとえば、BMP ファイル・タイプのヘッダー)、そのヘッダーとフレーム・データをファイルに書き込みます。 DBiBrowse API を使用すれば、そのファイルの内容を表示することができます。 DBvGetNextFrame、DBvSeekNextFrame、DBvFrameDatato24BitRGB API の使用方法の詳細、およびビデオ・フレームの表示方法については、185ページの『第14章 ビデオ・シーンの変化の検出』を参照してください。

## 音声やビデオの再生

データベース表に保管されている音声を再生するには、DBaPlay API を使用します。データベース表に保管されているビデオを再生するには、DBvPlay API を使用します。これらの API の使用法の詳細については、137ページの『表示 / 再生 API の使用』を参照してください。

# 第13章 イメージの内容による照会

図25 は、眼に見える例を探索基準として使ってデータベースの画像 (つまり、支配的な色やテクスチャー・パターンをもつ画像) を探索するアプリケーション・プログラムを示したものです。このようなアプリケーションでは、探索の入力として画像を指定することができます。ソース・イメージの色やテクスチャーが保管されている画像のそれと比較され、色やテクスチャーが入力と最もよく一致する画像が戻されます。

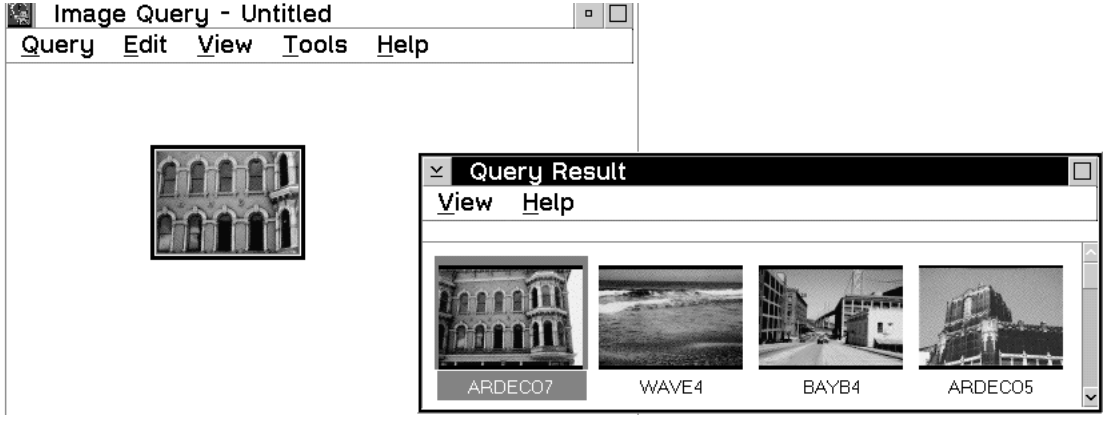



*図 25. イメージの内容による照会.* データベース表に保管されている画像の探索には、色やテクスチャーの眼に見える例が使用される。

画像を眼に見えるフィーチャーで照会する機能を**イメージ内容による照会 (QBIC)** と呼びます[5]。この章では、イメージ・エクステンダーの API と UDF を使って上で述べたようなアプリケーションを作成する方法について説明します。さらに、イメージ・エクステンダーのコマンドと API を使って QBIC 管理タスクを行う方法についても説明します。

## イメージ内容による照会方法

イメージ内容によって照会するには、次のようにします。

1. 画像の QBIC カタログを作成する。

5. イメージ・エクステンダーには、カリフォルニア大学バークレー校およびその提供者によって開発されたソフトウェアが含まれています。

2. 画像をカタログする。つまり、画像に対する項目をカタログに追加し、画像のフィーチャーに対する値を保管します。

   QBIC カタログと画像のフィーチャーに関する説明については、22ページの『QBIC カタログ』を参照してください。
3. 照会を作成する。照会では、探索基準として使用するフィーチャー、その値および重み付け (すなわち、各フィーチャーに置く強調) を指定します。このような照会属性は照会ストリングという文字ストリングに指定することができます。別の方法として、照会オブジェクトを作成し、このような属性を照会オブジェクトに関連付けることもできます。そうすれば、照会ストリングを保管して再利用することができます。
4. 照会を実行する。照会を実行する場合、入力として照会ストリングを指定するか、または照会に関する照会オブジェクトを指定します。いずれの場合も、探索する画像も指定します。いずれの場合も、DB2 コマンド行から、またはプログラム内から照会を実行依頼することができます。

応答として、イメージ・エクステンダーは、その照会のフィーチャー値を戻します。次にその値を、ターゲット・イメージの QBIC カタログに保管されているフィーチャー値と比較します。次にイメージ・エクステンダーは、各ターゲット・イメージのフィーチャー値がソースにどの程度似ているかを示す得点を計算します。

イメージ・エクステンダーからは、フィーチャー値がソースに最も似ている画像を得ることができます。さらに、1 つまたは複数の画像の得点を戻すようにイメージ・エクステンダーに指示することもできます。

## QBIC カタログの管理

イメージを内容によって照会するには、画像が QBIC カタログにカタログされていなければなりません。 QBIC カタログには、画像の視覚的なフィーチャーを示すデータが入っています。

QBIC カタログは、内容による探索で使用する必要のある、ユーザー表内の画像の列ごとに作成する必要があります。ユーザー表の各画像列に対し複数の QBIC カタログをもったり、同じ QBIC カタログを複数の列で共用したりすることはできません。

QBIC カタログを作成する際には、イメージ・エクステンダーによってデータを保管する対象のフィーチャーを指定します。さらに、イメージ・エクステンダーによって画像を自動的にカタログするかどうかも指定します。自動カタログを指定すると、イメージ・エクステンダーは、画像をユーザー表に保管する

際、自動的に画像の項目をカタログに作成します。画像を自動的にカタログしない場合は、手動でカタログする必要があります。つまり、画像の項目をカタログに作成するようにイメージ・エクステンダーに明示的に指定する必要があります。

QBIC カタログを作成したら、次のことができます。

- カタログに対してアクションを行えるようにカタログをオープンする。
- 設定を自動カタログから手動カタログに、または手動から自動に変更する。
- カタログにフィーチャーを追加する。これにより、イメージ・エクステンダーによってデータを保管する対象のフィーチャーを指定します。
- カタログからフィーチャーを除去する。
- カタログに関する情報を取り出す。そのカタログに対応するユーザー表と列の名前や、カタログに保管されているデータに対応するフィーチャーなど。
- 画像を手動でカタログに登録する。
- 画像をアンカタログする (つまり、その画像の項目をカタログから除去する)。
- 画像を再カタログする。
- カタログを再分散する (つまり、区分データベース・システム内でノードを追加または除去したとき)。
- カタログをクローズする。
- カタログを削除する。

QBIC カタログの作成を含め、イメージ・エクステンダーが提供する API を使ってこれらのタスクを行うことができます。また、これらのタスクの多くは db2ext コマンド行プロセッサーを使用して実行することもできます。

## QBIC カタログの作成

QBIC カタログを作成するには、QbCreateCatalog API または CREATE QBIC CATALOG コマンドを使用します。カタログを作成するには、カタログに登録する画像が入っているユーザー表の所有者でなければなりません。さらに、そのカタログが入るデータベースに対し CREATE TABLE 権限が必要です。画像の QBIC カタログを作成する場合には、そのユーザー表と画像列がイメージ・エクステンダーに対して使用可能になっていなければなりません。

QBIC カタログを作成する際には、次の指定を行います。

- カタログする画像が入っているユーザー表と列を指定する。

- 画像を自動的にカタログするかどうかを指定します。自動カタログを指定すると、イメージ・エクステンダーは、画像をユーザー表に保管する後、自動的に画像をカタログします。画像がカタログされるために待機しているかどうかを、エクステンダーは周期的に検査します。この周期は、DB2CATALOGDELAY 環境変数の値に秒単位で設定することができます。この値は 1 秒からかなり大きい値までを範囲として設定することができます。省略時値は 60 です。

  手動カタログを指定すると、画像のカタログをイメージ・エクステンダーに明示的に要求する必要があります。 (画像を手動でカタログする方法については、153ページの『画像を手動でカタログする場合』を参照してください。)

**ユーザー表と列が使用可能になっていなければなりません:** ユーザー表の列内の画像の QBIC カタログを作成する場合には、その表と列がイメージ・エクステンダーに対して使用可能になっていなければなりません。 (イメージ・エクステンダーに対してユーザー表と列を使用可能にする方法については、59ページの『第6章 エクステンダー・データのためのデータ・オブジェクトの準備』を参照してください。)

**API の使用:** QbCreateCatalog API を使用する際には、自動カタログか手動カタログかを自動カタログ値によって指定します。値 1 は自動カタログを、値 0 は手動カタログをそれぞれ示します。

たとえば、次のステートメントでは、従業員表の写真列の画像に対し QBIC カタログを作成します。これらの画像は、従業員表に保管される際に自動的にカタログされます。

```
SQLINTEGER autoCatalog=1;                          /* automatic cataloging */

rc=QbCreateCatalog(
                  "employee",                      /* user table */
                  "picture",                       /* image column */
                  autoCatalog);                    /* auto catalog setting */
```

**コマンド行の使用:** CREATE QBIC CATALOG コマンドを出すと、ON を指定すれば自動カタログが指定されます。 OFF を指定すれば、手動カタログが指定されます。省略時値は OFF です。

たとえば、次のステートメントでは、上の API の例と同じ QBIC カタログが作成されます。

```
CREATE QBIC CATALOG employee picture on
```

**QBIC カタログのバックアップ:** イメージ・エクステンダーは QBIC カタログをファイルに保管します。カタログの回復に備えて、これらのファイルのバックアップを定期的にとる必要があります。 AIX、HP-UX、または Sun Solaris サーバーでは、これらのファイルは /home/*instance_owner*/dmb/qbic ディレクトリーにあります。ここで、*instance_owner* はインスタンス所有者のユーザー ID です。 OS/2 や Windows のサーバーでは、これらのファイルは、¥*destination*¥instance¥*instance_name*¥qbic ディレクトリーにあります。*destination* はイメージ・エクステンダーがインストールされているディレクトリー、*instance_name* はそのエクステンダー・インスタンスの名前です。

## QBIC カタログのオープン

カタログを変更したりするアクションを行うには、まず QBIC カタログをオープンする必要があります。たとえば、フィーチャーをカタログに追加する前には、まず QBIC カタログをオープンしなければなりません。

QBIC カタログをオープンするには、QbOpenCatalog API 呼び出しか OPEN QBIC CATALOG コマンドを使用します。 QBIC カタログをオープンする際には、次の指定が必要です。

- そのカタログに対するユーザー表と画像列を指定する。
- カタログをオープンするモードを指定する (OPEN QBIC CATALOG コマンドを使用すると、これは暗黙に指定されます)。カタログは、内容による画像の探索などの読み取り操作用にオープンすることができます。あるいは、フィーチャーの追加などの更新用にオープンすることができます。読み取り操作用にカタログをオープンするには、そのユーザー表に対する SELECT 権限が必要です。更新操作用にカタログをオープンするには、そのユーザー表に対する UPDATE 権限が必要です。

**カタログがすでにオープンされている場合:** 別のセッションでカタログがすでに更新用にオープンされている場合には、そのカタログを更新操作用にオープンすることはできません。 QBIC カタログをオープンすると、イメージ・エクステンダーが現行セッションでオープンされている QBIC カタログをすべてクローズします。

**API の使用:** QbOpenCatalog API を使用する際には、カタログをオープンするモードを明示的に指定します。その場合、次のように指定します。

- カタログを読み取り操作用にオープンするには、API パラメーター qbiRead を指定します。
- カタログを更新用にオープンするには、API パラメーター qbiUpdate を指定します。

QbiRead と QbiUpdate は、QBIC ファイルの組み込み (ヘッダー) ファイルである dmbqbapi.h に定義されている定数です。

さらに、カタログ・ハンドルをポイント指定する必要があります。カタログ・ハンドルは、QBIC 固有のデータ・タイプである QbCatalogHandle をもちます。このデータ・タイプも dmbqbapi.h で定義します。カタログ・ハンドルの値は、API からの出力としてイメージ・エクステンダーによって戻されます。

たとえば、次の API 呼び出しでは、QBIC カタログを読み取り操作用にオープンします。

```
SQLINTEGER mode;
QbCatalogHandle  *CatHdl;

mode=qbiRead;                                   /* open catalog for */
                                                /* read operations */
rc=QbOpenCatalog(
                 "employee",                    /* user table */
                 "picture",                     /* image column */
                 mode,                          /* open catalog mode */
                 &CatHdl);                      /* catalog handle */
```

**コマンド行の使用:** OPEN QBIC CATALOG コマンドを出すと、イメージ・エクステンダーは、カタログを更新操作用にオープンしようとします。カタログがすでに別のセッションで更新用にオープンされていると、イメージ・エクステンダーはカタログを読み取り操作用にオープンします。

たとえば、次のコマンドを出せば、QBIC カタログがオープンされます。イメージ・エクステンダーは、カタログを更新操作用にオープンしようとします。

```
OPEN QBIC CATALOG employee picture
```

**QBIC 関連の作業が終わったらカタログをクローズします:** QBIC カタログをオープンすると、イメージ・エクステンダーは、メモリーなどの資源をそのカタログに割り当てます。 QBIC 関連の作業が終わったらカタログをクローズしてください。それによって、割り当てられた資源が解放されます。

## 自動カタログ設定の変更

設定を自動カタログから手動カタログに、または手動から自動に変更するには、QbSetAutoCatalog API または SET QBIC AUTOCATALOG コマンドを使用します。カタログ設定を変更するためには、QBIC カタログが更新用にオープンされていなければなりません。

**変更はさかのぼっては行われません:** 自動カタログ設定を変更しても、その変更は、その変更後にユーザー表の列に追加された画像にしか適用されません。

ユーザー表の列にすでに格納されている画像は影響を受けません。たとえば、設定を手動カタログから自動カタログへ変更した場合、その変更後にユーザー表の列に追加された画像だけが自動的にカタログされます。表の列にすでに格納されている画像をカタログするには、それらを手動でカタログする必要があります。 (画像を手動でカタログする方法については、153ページの『画像を手動でカタログする場合』を参照してください。)

**API の使用:** QbSetAutoCatalog API を使用する場合には、QBIC カタログのハンドル (QbOpenCatalog API を使用してカタログをオープンすると戻されます) を指定します。さらに、自動カタログの場合は、自動カタログの値として 1 を、手動カタログの場合は値 0 をそれぞれ指定します。

次の例では、従業員表の写真列の画像に対応する QBIC カタログに対し、手動カタログが指定されます。 QBIC カタログはまず更新操作用にオープンされます。

```
SQLINTEGER mode;
SQLINTEGER autoCatalog=0;                          /* manual cataloging */

QbCatalogHandle  *CatHdl;

mode=qbiUpdate;                                    /* open catalog for */
                                                   /* update */
/* Open a QBIC catalog */
rc=QbOpenCatalog(
                 "employee",                       /* user table */
                 "picture",                        /* image column */
                 mode,                             /* open catalog mode */
                 &CatHdl);                         /* catalog handle */

/* Change the auto catalog setting */
rc=QbSetAutoCatalog(
                  CatHdl,                          /* catalog handle */
                  autoCatalog);                    /* auto catalog flag */
```

**コマンド行の使用:** SET QBIC AUTOCATALOG コマンドを出すと、ON を指定すれば自動カタログが指定されます。 OFF を指定すれば、手動カタログが指定されます。このコマンドは、現在オープンされているカタログに対して作用します。

たとえば、次のコマンドでは、現在オープンされている QBIC カタログの自動カタログをオフに設定します。

```
SET QBIC AUTOCATALOG off
```

## フィーチャーを QBIC カタログに追加する場合

フィーチャーを QBIC カタログに追加するには、QbAddFeature API または ADD QBIC FEATURE コマンドを使用します。少なくとも 1 つのフィーチャーを QBIC カタログに追加しないと、画像をそこにカタログすることはできません。また QBIC カタログが更新用にオープンされていないと、フィーチャーを追加することはできません。

フィーチャーをカタログに追加する際には、追加するフィーチャーの名前を指定します (フィーチャー名のリストを表8 に示します)。

*表 8. QBIC のフィーチャー名*

| フィーチャー名 | 説明 |
|---|---|
| QbColorFeatureClass | 平均色 |
| QbColorHistogramFeatureClass | ヒストグラム色 |
| QbDrawFeatureClass | 位置色 |
| QbTextureFeatureClass | テクスチャー |

**画像を再カタログする必要があるかもしれません:** ユーザーがフィーチャーを QBIC カタログに追加する際、イメージ・エクステンダーは、自動カタログがオンにセットされていても、すでにカタログされている画像に対して新しいフィーチャーに関するデータを自動的には保管しません。

**API の使用:** QbAddFeature API を使用する場合には、フィーチャー名の他に、QBIC カタログのハンドルを指定する必要があります。フィーチャー名の長さに対し、定数 qbiMaxFeatureName が使用されていることに注意してください。この定数は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に値 50 で定義されています。

次の例では、QbAddFeature API を使って、ヒストグラム色というフィーチャーを QBIC カタログに追加します。

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbAddFeature(
               CatHdl,                          /* catalog handle */
               featureName);                    /* feature name */
```

**コマンド行の使用:** ADD QBIC FEATURE コマンドは、現在オープンされているカタログに対して作用します。次の例では、このコマンドを使って、位置色というフィーチャーを現在オープンされているカタログに追加します。

```
ADD QBIC FEATURE QbDrawFeatureClass
```

## フィーチャーを QBIC カタログから除去する場合

フィーチャーを QBIC カタログから除去するには、QbRemoveFeature API または REMOVE QBIC FEATURE コマンドを使用します。イメージ・エクステンダーはそのフィーチャーに関してカタログ表を削除します。その結果、画像をカタログしても、そのフィーチャーのデータは保管されません。フィーチャーを除去する場合には、その QBIC カタログが更新用にオープンされていなければなりません。

フィーチャーをカタログから除去する場合には、除去するフィーチャーの名前を指定します。

**API の使用:** QbRemoveFeature API を使用する場合には、フィーチャー名の他に、QBIC カタログのハンドルを指定する必要があります。

次の例では、QbRemoveFeature API を使って、ヒストグラム色というフィーチャーを QBIC カタログから除去します。

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbRemoveFeature(
                CatHdl,                          /* catalog handle */
                featureName);                    /* feature name */
```

**コマンド行の使用:** REMOVE QBIC FEATURE コマンドは、現在オープンされているカタログを処理します。次の例では、このコマンドを使って、位置色というフィーチャーを現在オープンされている QBIC カタログから除去します。

```
REMOVE QBIC FEATURE QbDrawFeatureClass
```

## QBIC カタログに関する情報を取り出す場合

QBIC カタログに関し、次の情報を取り出すことができます。

- そのカタログに対応するユーザー表と画像列の名前
- カタログにデータが保管されているフィーチャーの数と、それらのフィーチャーの名前
- 画像をユーザー表に保管するときに、イメージ・エクステンダーによって画像を自動的にカタログするかどうか

ユーザー表と列名、フィーチャーの数、および自動カタログ設定を取り出すには、QbGetCatalogInfo API を使用します。フィーチャー名を取り出すには、QbListFeatures API を使用します。また、すべての情報を取り出すには、GET QBIC CATALOG INFO コマンドを使用します。

情報を取り出すには、あらかじめ、その QBIC カタログが更新用にオープンされていなければなりません。

**API の使用:** QbGetCatalogInfo API を使用するときは、QBIC カタログのハンドルを指定する必要があります。さらに、イメージ・エクステンダーがカタログ情報を戻す構造体をポイントする必要があります。カタログ情報の構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqapi.h に次のように定義されています。

```
typedef struct{
          char       tableName[qbiMaxTableName+1]    /* user table */
          char       columnName[qbiMaxColumnName+1]  /* image column */
          SQLINTEGER featureCount;                   /* number of features */
          SQLINTEGER autoCatalog;                    /* auto catalog flag */
} QbCatalogInfo;
```

QbListFeatures API 呼び出しを使用する場合には、戻されるフィーチャー名を入れるバッファーを割り振る必要があります。バッファーに保管されるフィーチャー名は、ブランク文字で区切ります。さらに、カタログ・ハンドルと、戻されるフィーチャー名のバッファー・サイズを指定する必要があります。必要なバッファー・サイズを見積もるには、QbGetCatalogInfo API で戻されるフィーチャー・カウントを使用して、そのカウントに、最長のフィーチャー名の長さを掛けます。最も長いフィーチャー名の大きさとして、定数 qbiMaxFeatureName が使用できます。

次の例の API 呼び出しによって、QBIC カタログに関する情報が取り出されます。 QbListFeatures API のバッファー・サイズを計算するのに、QbGetCatalogInfo API によって戻されるフィーチャー数と qbiMaxFeatureName 定数がどのように使用されているかに注目してください。

```
long  bufSize;
long  count;
char  *featureNames;

QbCatalogHandle  CatHdl;
QbCatalogInfo    catInfo;

/* Get user table name, image column name, feature count, */
/* and auto catalog setting */
rc=QbGetCatalogInfo(
                CatHdl,                          /* catalog handle */
                &catInfo);                       /* catalog info. structure */

/* List feature names */
```

```
bufSize=catInfo.featureCount*qbiMaxFeatureName;
featureNames=malloc(bufSize);

rc=QbListFeatures(
               CatHdl,                    /* catalog handle */
               bufSize                    /* size of buffer */
               count,                     /* feature count */
               featureNames);             /* buffer for feature names */
```

**コマンド行の使用:** GET QBIC CATALOG INFO コマンドは、現在オープンされているカタログに対して作用します。次の例では、このコマンドを使って、現在オープンされている QBIC カタログの情報を取り出します。

```
GET QBIC CATALOG INFO
```

## 画像を手動でカタログする場合

画像をユーザー表に保管する際にイメージ・エクステンダーに画像を自動的にカタログさせるかどうかをカタログの作成時に指定します。画像を自動的にカタログしない場合は、画像をユーザー表に保管した後で、それを手動でカタログしなければなりません。手動でカタログする場合、単一の画像をカタログすることも、表全体の画像をカタログすることもできます。

### 手動で単一画像をカタログする場合

手動で 1 つの画像をカタログするには、QbCatalogImage API を使用します。コマンド行からは個別の画像を指定することができないので、コマンドで画像をカタログすることはできません。 API を使用する際には、カタログ・ハンドルと画像ハンドルを指定します (画像ハンドルはユーザー表から取り出せます)。画像を手動でカタログする場合には、QBIC カタログがオープンされていなければなりません。

たとえば、次のステートメントでは、画像ハンドルをユーザー表から取り出し、続いてその画像をカタログします。

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;

char Img_hdl[251];

EXEC SQL END DECLARE SECTION;
QbCatalogHandle   CatHdl;
EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';

/* Catalog the image*/

rc=QbCatalogImage(
               CatHdl,                              /* catalog handle */
               Img_hdl);                            /* image handle */
```

### 手動で 1 列の画像をカタログする場合

1 つの列全体の画像を手動でカタログするには、QbCatalogColumn API か CATALOG QBIC COLUMN コマンドを使用します。イメージ・エクステンダーは、列が最後にカタログされて以降に新しく挿入、更新、削除された列の画像のみをカタログします。イメージ・エクステンダーは、カタログのすべてのフィーチャーについてそれらの画像をカタログします。列全体の画像を手動でカタログする場合には、QBIC カタログが更新用にオープンされていなければなりません。

**API の使用:** QbCatalogColumn API を使用する場合には、カタログ・ハンドルを指定します。イメージ・エクステンダーは、指定されたカタログに対応するユーザー表の列の画像を使用します。

イメージ・エクステンダーは、指定されたカタログに対応するユーザー表の列の画像を使用します。それらの画像は、カタログにあるすべてのフィーチャーに対してカタログされます。

```
QbCatalogHandle   CatHdl;

rc=QbCatalogColumn(
                CatHdl);                              /* catalog handle */
```

**コマンド行の使用:** 画像に関する 1 つの列を手動でカタログするには、CATALOG QBIC COLUMN コマンドを使用します。このコマンドは画像の再カタログにも使用します (155ページの『画像を再カタログする場合』を参照)。パラメーター FOR と NEW を使用してください。 (FOR と NEW は省略時パラメーターです。)

次の例では、このコマンドを使って、現在オープンされているカタログに対応する列の画像のうち、アンカタログのものをカタログします。それらの画像は、カタログにあるすべてのフィーチャーに対してカタログされます。

```
CATALOG QBIC COLUMN FOR NEW
```

## 画像をアンカタログする場合

画像のアンカタログとは、画像に対する項目を QBIC カタログから除去することです。画像をアンカタログするには、QbUncatalogImage API を使用します。コマンド行からは個別の画像を指定することができないので、コマンドで画像をアンカタログすることはできません。 API を使用する際には、カタログ・ハンドルと画像ハンドルを指定します (画像ハンドルはユーザー表から取り出せます)。画像をアンカタログする場合には、その QBIC カタログが更新用にオープンされていなければなりません。

たとえば、次のステートメントでは、画像ハンドルをユーザー表から取り出し、続いてその画像をアンカタログします。

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;
char Img_hdl[251];
EXEC SQL END DECLARE SECTION;

QbCatalogHandle   CatHdl;

EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';

/* Uncatalog the image */

rc=QbUncatalogImage(
                CatHdl,                          /* catalog handle */
                Img_hdl);                        /* image handle */
```

## 画像を再カタログする場合

画像をカタログすると、イメージ・エクステンダーは、その QBIC カタログに指定されているフィーチャーを分析し、それらのフィーチャーに対する値をカタログに保管します。ユーザーがフィーチャーを QBIC カタログに追加しても、イメージ・エクステンダーは、すでにカタログされている画像に対して新しいフィーチャーを自動的には分析しません。新しいフィーチャーの値をカタログに追加するには、すべての画像を再カタログする必要があります。

画像を QBIC カタログに再カタログするには、QbReCatalogColumn API か CATALOG QBIC COLUMN コマンドを使用します。イメージ・エクステンダーは、カタログにあるすべてのフィーチャー・データを除去します。それから、画像のすべてのフィーチャーについて (新しいフィーチャーも含め) 分析し、画像をカタログします。画像を再カタログする場合には、その QBIC カタログがオープンされていなければなりません。

**API の使用:** QbReCatalogColumn API を使用する場合には、カタログ・ハンドルを指定します。

次の例では、QBIC カタログの画像が再度分析されます。

```
QbCatalogHandle   CatHdl;

rc=QbReCatalogColumn(
                CatHdl);                         /* catalog handle */
```

**コマンド行の使用:** 画像を再カタログするには、CATALOG QBIC COLUMN コマンドを使用します。このコマンドは、現在オープンされているカタログに対して作用します。さらに、コマンドを使用して画像を手動でカタログします (153ページの『画像を手動でカタログする場合』を参照してください)。

このコマンドを出す場合には、パラメーター FOR と ALL を指定してください。これによって、イメージ・エクステンダーは、すべての画像を再カタログします。

次の例では、現在オープンされている QBIC カタログにカタログされている画像が再カタログされます。

```
CATALOG QBIC COLUMN FOR ALL
```

## QBIC カタログの再分散 (EEE のみ)

ノードをノード・グループに追加したり、ノード・グループから除去するときに、QBIC フィーチャー・データを再分散するには、DMBRedistribute API または REDISTRIBUTE NODEGROUP コマンドを使用します。このコマンドは、QBIC フィーチャー・データを対応するユーザー・データと同じノードに配置します。

再分散プロセスがエラーを戻した場合は、コマンドの応答にある指示に従って CONTINUE パラメーターをコマンドに指定するかどうかを決めて、もう一度コマンドを実行することができます。このオプションは、プロセスを最初からやり直すのではなく、プロセスが停止した場所から続行するよう、システムに指示します。 DB2 の REDISTRIBUTE コマンドを実行した後、初めてエクステンダーの REDISTRIBUTE NODEGROUP を実行するときには、CONTINUE パラメーターを使用することはできません。

データ保全性を維持するには、一度に 1 つのノード・グループを再分散します。そのノード・グループの再分散が終わるまで待ってから、次のノード・グループの再分散を開始します。

**API の使用:** 次の例は、groupone という名前のノード・グループ内で QBIC フィーチャー・データを再分散する方法を示しています。

```
#include <dmbrdst.h>

rc = DMBRedistribute(groupone,"continue");
```

**コマンド行の使用:** 次の例は、REDISTRIBUTE NODEGROUP コマンドとともに CONTINUE パラメーターを使用して、my_nodegroup という名前のノードのデータを再分散する方法を示したものです。

```
redistribute nodegroup my_nodegroup continue
```

## QBIC カタログをクローズする場合

QBIC カタログをクローズするには、QbCloseCatalog API または CLOSE QBIC CATALOG コマンドを使用します。カタログをクローズするためには、そのカタログがオープンされていなければなりません。

**API の使用:** QbCloseCatalog API 呼び出しを使用する場合には、カタログ・ハンドルを指定します。次はその例です。

```
QbCatalogHandle   CatHdl;

rc=QbCloseCatalog(
                CatHdl);                              /* catalog handle */
```

**コマンド行の使用:** CLOSE QBIC CATALOG コマンドは、現在オープンされているカタログに対して作用します。次の例では、このコマンドを使って、現在オープンされている QBIC カタログをクローズします。

```
CLOSE QBIC CATALOG
```

## QBIC カタログを削除する場合

QBIC カタログを削除すると、そのカタログ表のすべてのフィーチャー・データが削除されます。その結果、それに対応する画像を内容で照会することはできなくなります。 QBIC カタログを削除するには、そのカタログに対応する表に対し、ALTER か CONTROL 権限が必要です。カタログを削除するためには、その QBIC カタログがオープンされていなければなりません。

QBIC カタログを削除するには、QbDeleteCatalog API か DELETE QBIC CATALOG コマンドを使用します。 QBIC カタログを削除する場合には、そのカタログに対応するユーザー表と列を指定します。

**API の使用:** 次の例では、QbDeleteCatalog API を使用して、QBIC カタログを削除します。

```
rc=QbDeleteCatalog(
                 "employee",                      /* user table */
                 "picture");                      /* image column */
```

**コマンド行の使用:** DELETE QBIC CATALOG コマンドは、現在オープンされているカタログを処理します。次の例では、このコマンドを使って、現在オープンされている QBIC カタログを削除します。

```
DELETE QBIC CATALOG employee picture
```

## QBIC カタログのサンプル・プログラム

159ページの図26 は、QBIC カタログを作成する C プログラムの一部を示したものです。このプログラムは、列全体の画像を QBIC カタログにカタログする機能もあります。このプログラムはその全体が SAMPLES サブディレクトリーの QBCATDMO.C ファイルにあります。このプログラムを実行する場合には、ENABLE と POPULATE のサンプル・プログラム (これらのプログラムも SAMPLES サブディレクトリーにあります) をまず実行する必要があります。サンプル・プログラムの詳細については、623ページの『付録B. サンプル・プログラムとメディア・ファイル』を参照してください。

プログラムの次の点に注目してください。

1. dmbqbapi ヘッダー・ファイルをインクルードする。
2. データベースに接続する。
3. カタログを作成する。カタログは、自動カタログがオフで作成されます。
4. カタログを更新用にオープンする。
5. 平均色というフィーチャーをカタログに追加する。
6. 1 つの列全体の画像をカタログする。
7. カタログをクローズする。

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>

/*****************************************************************************/
/*  Define the function prototypes                                           */
/*****************************************************************************/

void printError(SQLHSTMT hstmt);
void createCatalog();
void openCatalog();
void closeCatalog();
void addFeature();
void catalogImageColumn();

QbCatalogHandle cHdl = 0;

static SQLHENV henv;
static SQLHDBC hdbc;
static SQLHSTMT hstmt;
static SQLRETURN rc;
char tableName[] = "sobay_catalog";
char columnName[] = "covers";

SQLCHAR uid[18+1];
SQLCHAR pwd[30+1];
SQLCHAR dbnName[SQL_MAX_DSN_LENGTH+1];

void main ()
{
/*---- prompt for database name, userid, and password ----*/
  printf("Enter database name:¥n");
  gets((char *) dbName);
  printf("Enter userid:¥n");
  gets((char *) pwd);
/* set up the SQL CLI environment */
 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc,dbname,SQL_NTS,uid,SQL_NTS,pwd,SQL_NTS); 2
 if (rc != SQL_SUCCESS)
 {
  printError(SQL_NULL_HSTMT);
  exit(1);
 }
```

*図 26. QBIC カタログのサンプル・プログラム (1/4)*

```
 createCatalog();
   openCatalog();
   addFeature();
 getCatalogInfo();
 listFeatures();
 catalogImageColumn();
 closeCatalog();

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
/*********************************************************************/
void createCatalog()
{
 SQLINTEGER autoCatalog = 0;
 SQLINTEGER retLen;
   SQLINTEGER errCode = 0;
   char errMsg[500];

   QbCreateCatalog( 3
         (char *) tableName,
         (char *) columnName,
         autoCatalog,
         0
        );

   DBiGetError(&errCode, errMsg);
   if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
/*********************************************************************/
void openCatalog()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];
 SQLINTEGER mode = qbiUpdate;

   QbOpenCatalog( 4
         (char *) tableName,
         (char *) columnName,
         mode,
         &cHdl
        );

   DBiGetError(&errCode, errMsg);
   if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
```

*図 26. QBIC カタログのサンプル・プログラム (2/4)*

```
/*********************************************************************/
void addFeature()
{
 SQLINTEGER errCode=0;
 char errMsg[5
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
   char featureName*lbrk.] = "QbColorFeatureClass"; 5
     QbaddFeature(
         cHdl,
         featureName
        );

     DBiGetError(&errCode, errMsg);
     if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
 }
 else
 {
  exit(1);
 }
}
/*********************************************************************/
void catalogImageColumn()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];

 if(cHdl) /* if we have an open catalog, else do nothing */
 {
  SQLRETURN rc;
    QbCatalogColumn(  6
         cHdl,
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
 else
 {

  exit(1);
 }
}
```

*図 26. QBIC カタログのサンプル・プログラム (3/4)*

```
/*********************************************************************/
void closeCatalog()
{
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
    QbCloseCatalog( 7
         cHdl,
        );
 }
}
/*********************************************************************/
```

*図 26. QBIC カタログのサンプル・プログラム (4/4)*

## 照会の作成

内容によって画像を照会するときは、その照会の入力と、ターゲットのカタログ・イメージを指定します。照会の入力では、照会で使用するフィーチャー名、フィーチャーの値および重み (すなわち、各フィーチャーに置く強調) を指定します。

この入力を提供する方法は 2 つあります。

- 照会で照会ストリングを指定します。照会ストリングは、照会に関するフィーチャー、フィーチャーの値および重みを指定する文字ストリングです。
- 照会オブジェクトを作成し、照会で参照します。照会オブジェクトは、フィーチャーおよびフィーチャーの重みを指定します。照会オブジェクトは各フィーチャーのデータ・ソースも指定します。各データ・ソースは各フィーチャーの値を提供します。

### 照会ストリングを指定する場合

照会を行うためにフィーチャー、フィーチャーの値および重みを指定するためには照会ストリングを使用することができます。**照会ストリング**は、*feature_name value* の形式を持つ文字ストリングです。ここで、*feature_name* (フィーチャー名) は QBIC フィーチャー名であり、*value* (値) はフィーチャーに関連する値です。

1 つの照会に複数のフィーチャーを指定することができます。その後、フィーチャーごとにフィーチャー名と値のペアを、163ページの『フィーチャーの値』の説明に従って指定します。各ペアは、AND 文節で区切ります。 1 つの照会に複数のフィーチャーを指定する場合、165ページの『フィーチャーの重み』で説明するように、1 つまたは複数のフィーチャーに重みを割り当てることができます。その後、照会ストリングは *feature_name value weight* の形式を持ちます。 *weight* (重み) はフィーチャーに割り当てる重みです。

イメージ・エクステンダーには、照会ストリングを使用する 1 つの API (QbQueryStringSearch) と 2 つの UDF (QbScoreFromStr と QbScoreTBFromStr) が用意されています。照会を出す場合、適切な API または UDF を使用し、入力パラメーターとして照会ストリングを指定してください。 (詳細については、174ページの『イメージ内容による照会の実行』を参照してください。)

### フィーチャーの値

照会内で各フィーチャーに関する照会ストリングにフィーチャー値 (feature value) を指定します。

DB2 コマンド内部で照会を渡す際、照会を適正に機能させるためには、特定のファイル命名規則に従わなければなりません。スペースまたは右不等号括弧 (>) を含むファイル名は、二重引用符で囲む必要があります。その他のファイル名は、二重引用符で囲んでも囲まなくてもかまいません。ファイル名の前後に引用符を使う場合には、それぞれの引用符の前にエスケープ文字 (¥) を付けなければなりません。照会が DB2 コマンド内で渡されない場合は、引用符にエスケープ文字を付ける必要はありません。

次の例では、照会ストリングは DB2 コマンドで渡されます。

```
db2 "select image_id from table
(mmdbsys.QbScoreTBFromStr
('texture file=<server,patterns/ptrn07.gif>',
'fabric',
'swatch_img',
10))
as T1"
```

164ページの表9 は、各フィーチャーに指定できる値をリストしています。各フィーチャー名のすぐ下にあるのは、代わりに使用できる短縮名です。

*表 9. 照会ストリングに指定できるフィーチャー値*

| フィーチャー名 | 値 |
|---|---|
| averageColor、average、または QbColorFeatureClass | color=<*Rvalue, Gvalue, Bvalue*><br><br>各色の値は、0 から 255 の整数で、画像の赤の値 (*Rvalue*)、緑の値 (*Gvalue*)、および青の値 (*Bvalue*) を指定します。<br><br>file=<*file_location, filename*><br><br>*file_location* は、サーバー・ファイル用のサーバーです。 *filename* は、ファイルが常駐しているシステムに適した形式で指定した完全なファイル・パス、または相対ファイル名です。 DB2 エクステンダーは環境変数を使用して相対ファイル名を解析します (613ページの『環境変数の使用によるファイル名の解決方法』を参照)。 |
| histogram、histogramcolor、または QbColorHistogramFeatureClass | histogram=<(*hist_value, Rvalue, Gvalue, Bvalue*>), ...<br><br>各ヒストグラム色の値は、ヒストグラム (*hist_value*) 内のその色のパーセント (1 から 100)、およびその色の赤の値 (*Rvalue*)、緑の値 (*Gvalue*)、および青の値 (*Bvalue*) を示す文節に指定します。<br><br>file=<*file_location, filename*><br><br>*file_location* は、サーバー・ファイル用のサーバーです。 *filename* は、ファイルが常駐しているシステムに適した形式で指定した完全なファイル・パス、または相対ファイル名です。 DB2 エクステンダーは環境変数を使用して相対ファイル名を解析します。 |
| draw、positional、または QbDrawFeatureClass | file=<*file_location, filename*><br><br>handle=<*image_handle*><br><br>*file_location* は、サーバー・ファイル用のサーバーです。 *filename* は、ファイルが常駐しているシステムに適した形式で指定した完全なファイル・パス、または相対ファイル名です。 DB2 エクステンダーは環境変数を使用して相対ファイル名を解析します。 |

*表 9. 照会ストリングに指定できるフィーチャー値 (続き)*

| フィーチャー名 | 値 |
|---|---|
| texture または QbTextureFeatureClass | file=<*file_location, filename*><br><br>handle=<*image_handle*><br><br>*file_location* は、サーバー・ファイル用のサーバーです。 *filename* は、ファイルが常駐しているシステムに適した形式で指定した完全なファイル・パス、または相対ファイル名です。 DB2 エクステンダーは環境変数を使用して相対ファイル名を解析します。 |

### フィーチャーの重み

1 つの照会ストリングに複数のフィーチャーを指定する場合、1 つまたは複数のフィーチャーに重み (feature weight) を割り当てることもできます。フィーチャーの重みは、イメージ・エクステンダーが類似性得点を計算し、イメージ内容による照会の結果を戻すときに、フィーチャーに置く強調を示します。フィーチャーに指定する重みが大きければ大きいほど、照会に組み込まれたそのフィーチャーに対する強調が大きくなります。重みは、0.0 より大きい実数 (たとえば 2.5 または 10.0) です。照会ストリングで重みを割り当てない場合、イメージ・エクステンダーはそのフィーチャーの省略時の重みを使用します。フィーチャーが 1 つの照会ストリングに指定された唯一のフィーチャーである場合、重みの割り当ては無意味です。 (そのフィーチャーは照会内で常にすべての重みを持ちます。)

フィーチャーの重みは、照会内に指定した他のフィーチャーに相対的です。たとえば、照会ストリングに平均色とテクスチャーのフィーチャーを指定し、平均色に 2.0 の重み値を指定するとします。これにより、イメージ・エクステンダーは、平均色の値にテクスチャー値の 2 倍の強調を与えることになります。

### 例

次の照会ストリングは、平均的な赤色を指定します。

```
averageColor color=<255, 0, 0>
```

次の照会ストリングは、10% の赤、50% の緑、および 40% の青からなるヒストグラムを指定します。

```
histogram histogram=<(10, 255, 0, 0), (50, 0, 255, 0),
                                      (40, 0, 0, 255)>
```

次の照会ストリングは、平均色値とテクスチャー値を指定します。テクスチャー値はサーバー・ファイル内の画像によって提供されます。テクスチャーの重みは平均色の 2 倍です。

```
averageColor color=<30, 200, 25> and
  texture file=<server, "¥patterns¥pattern7.gif"> weight=2.0
```

## 照会オブジェクトの使い方

照会のためにフィーチャー、フィーチャーの値および重みを指定するために照会オブジェクトを使用することができます。照会オブジェクトを作成し、それにフィーチャーを追加します。次に、各フィーチャーごとにデータ・ソースを指定します。データ・ソースで、各フィーチャーの値を提供します。たとえば、データ・ソースはファイル内の画像である場合があります。平均色が該当するフィーチャーである場合、画像の平均色が照会オブジェクトに関連付けられます。 1 つの照会オブジェクトに複数のフィーチャーを追加する場合、1 つまたは複数のフィーチャーに重みを割り当てることができます。

イメージ・エクステンダーでは、照会オブジェクトを使用する 3 つの API (QbQuerySearch、QbQueryStringSearch、および QbQueryNameSearch) および 2 つの UDF (QbScoreFromName および QbScoreTBFromName) が用意されています。照会を出す場合、適切な API または UDF を使用し、入力パラメーターとして照会オブジェクトを指定してください。 (詳細については、174ページの『イメージ内容による照会の実行』を参照してください。)

### 照会オブジェクトの作成

照会オブジェクトを作成するには、QbQueryCreate API を使用します。 応答として、イメージ・エクステンダーは、その照会オブジェクトのハンドルを戻します。このハンドルは、QBIC 用のインクルード (ヘッダー) ファイル dmbqbapi.h に定義されている QBIC 固有のデータ・タイプ QbQueryHandle を持ちます。

この API を使用するときは、照会オブジェクト・ハンドルをポイント指定する必要があります。このハンドルは、照会オブジェクトに関して行う他の操作の API (フィーチャーの追加など) にも指定する必要があります。

たとえば、次の API 呼び出しは、照会オブジェクトを作成します。

```
QbQueryHandle  qHandle;

rc=QbQueryCreate(
              &qHandle);                          /* query object handle */
```

### フィーチャーを照会オブジェクトに追加する

画像のフィーチャーをイメージ・エクステンダーに照会させる場合は、それらのフィーチャーを照会オブジェクトに追加しておきます。

フィーチャーを照会オブジェクトに追加するには、QbQueryAddFeature API を使用します。 この API を使用するときは、照会オブジェクト・ハンドルを指定します。さらに、フィーチャー名を指定する必要があります。この API にはフィーチャーを 1 つしか指定できません。照会オブジェクトに追加するフィーチャーごとに別個の API 呼び出しを出す必要があります。

次の例では、QbQueryAddFeature API を使って、平均色というフィーチャーを照会オブジェクトに追加します。

```
char  featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;

rc=QbQueryAddFeature(
                qHandle,                          /* query object handle */
                "QbColorFeatureClass");           /* feature name */
```

### フィーチャーのデータ・ソースを照会オブジェクトに指定する

フィーチャーのデータ・ソースを照会オブジェクトに指定するには、QbQuerySetFeatureData API を使用します。次のものがデータ・ソースになります。

- ユーザー表の列にあるカタログまたはアンカタログされた画像
- クライアント・ワークステーションにあるイメージ・ファイル
- クライアント・ワークステーションのバッファーにある画像

さらに、平均色やヒストグラムのフィーチャーに対してはデータを明示的に指定することができます。たとえば、平均的な赤、緑、青の値を指定することができます。

この API を使用するときは、次のようにします。

- 照会オブジェクト・ハンドルを指定する。
- フィーチャーを指定する。
- QbImageSource 構造をポイント指定する (詳細は、168ページを参照)。

***データ・ソース構造体の使用:*** 照会オブジェクトのデータ・ソース情報には、いくつかの構造体が使用されます。これらの構造体には、次のものがあります。

- QbImageSource
- QbColor

- QbHistogramColor

**QbImageSource:** QbImageSource 構造は、フィーチャーのソースのタイプを照会オブジェクトに指定します。この構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に次のように定義されています。

```
typedef struct{
          SQLINTEGER    type;
          union {
                char              imageHandle[MMDB_BASE_HANDLE_LEN+1];
                QbImageFile       clientFile;
                QbImageBuffer     buffer;
                QbSampleSource    reserved;
                QbColor           averageColor;
                QbHistogramColor  histogramColor[qbiHistogramCount];
          };
} QbImageSource;
```

QbImageSource 構造体のタイプ・フィールドがソースのタイプです。このフィールドの値には、次のものが指定できます。

| 値 | 意味 |
|---|---|
| qbiSource_ImageHandle | ソースはユーザー表の列にあります。 |
| qbiSource_ClientFile | ソースはクライアント・ワークステーション・ファイルにあります。 |
| qbiSource_Buffer | ソースはクライアント・ワークステーション・バッファーにあります。 |
| qbiSource_ServerFile | ソースはサーバー・ファイルにあります。 |
| qbiSource_AverageColor | ソースは平均色指定です。 |
| qbiSource_HistogramColor | ソースはヒストグラム指定です。 |

これらの設定は、該当するフィーチャーに対してのみ有効です。たとえば、qbiSource_AverageColor は平均色のフィーチャーに対してのみ有効です。

タイプ・フィールドを qbiSource_ServerFile に設定する場合には、サーバー上のファイルの名前とタイプに clientFile を使用してください。

ソースのタイプによっては、イメージ・エクステンダーは指定された他の情報も検査します。これを表10 に示します。

*表 10. QbImageSource でイメージ・エクステンダーが調べる項目*

| ソース | イメージ・エクステンダーが調べる項目 | 指定する場所 |
|---|---|---|
| ユーザー表 | 画像ハンドル | QbImageSource の画像ハンドル・フィールド |

*表 10. QbImageSource でイメージ・エクステンダーが調べる項目 (続き)*

| ソース | イメージ・エクステンダーが調べる項目 | 指定する場所 |
|---|---|---|
| ファイル | ファイルの名前<br>ファイルの形式 | QbImageSource の clientFile フィールド |
| バッファー | ファイルの名前 | QbImageBuffer (この構造体の使用についての詳細は、169ページを参照) |
| 平均色の指定 | 赤、緑、青の値 | QbColor (この構造体の使用についての詳細は、169ページを参照) |
| ヒストグラム色の指定 | 色の値とパーセント | QbHistogramColor (この構造体の使用についての詳細は、169ページを参照) |

**QbImageBuffer:** データ・ソースがバッファーにある場合、画像の形式、長さ、内容を指定するには、QbImageBuffer 構造体を使用します。この構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に次のように定義されています。

```
typedef struct{
          char          format[qbiImageFormatLength+1];
          SQLINTEGER    length;
          char*         image;
} QbImageBuffer;
```

**QbColor:** データ・ソースが平均色指定の場合、平均的な赤、緑、青の値を指定するには、QbColor 構造体を使用します。この構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に次のように定義されています。

```
typedef struct{
          SQLUSMALLINT    red;          /*0 off - 65535 (fully on) */
          SQLUSMALLINT    green;        /*0 off - 65535 (fully on) */
          SQLUSMALLINT    blue;         /*0 off - 65535 (fully on) */
} QbColor;
```

平均値の計算で係数として使用する赤、緑、青のピクセル量を指定するには、QbColor に値を指定します。この値の範囲は 0 から 65535 です。値に 0 を指定すると、この項目は無視されます。

**QbHistogramColor:** ヒストグラム色指定の各色構成要素を指定するには、QbHistogramColor 構造体を使用します。 1 つのヒストグラム色に対する指定は、QbHistogramColor 構造体の配列で行います。各構造体には、1 つの色とパーセントを指定します。色値は、赤、緑、青のピクセル値からなります。パーセントでは、ターゲット・イメージにおけるその色の割合を指定します。

この構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に次のように定義されています。

```
typedef struct{
          QbColor         color;
          SQLUSMALLINT    percentage;   /*0 - 100 */
} QbHistogramColor;
```

その色の赤、緑、青のピクセル量を指定するには、QbColor に値を指定します。この値の範囲は 0 から 65535 です。パーセントでは、ターゲット・イメージにおけるその色の割合を指定します。この値の範囲は 1 から 100 です。それぞれの色構成要素のパーセントの合計は 100 以下でなければなりません。

***例:*** 次の例の API では、ヒストグラム色に対するデータ・ソースを照会オブジェクトに指定します。データ・ソースはクライアント・ワークステーションのファイルにあります。

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;
QbImageSource  imgSource;
imgSource.type=qbiSource_ClientFile;
strcpy(imgSource.clientFile.fileName,"/tmp/image.gif");
strcpy(imgSource.clientFile.format,"GIF");

rc=QbQuerySetFeatureData(
                qHandle,                              /* query object handle */
                "QbColorHistogramFeatureClass",       /* feature name */
                &imgSource);                          /* feature data source */
```

次の例のデータ・ソースには、平均色の指定として赤を指定します。

```
char                 featureName[qbiMaxFeatureName];
QbColor              avgColor;
QbImageSource        imgSource;
imgSource.type=qbSource_AverageColor;
avgColor.red=255;
avgColor.green=0;
avgColor.blue=0;
strcpy(featureName,"QbColorFeatureClass");
rc=QbQuerySetFeatureData(
                qHandle,                             /* query object handle */
                featureName,                         /* feature name */
                &imgSource);                         /* feature data source */
```

### 照会オブジェクトにフィーチャーの重みを設定する

照会オブジェクトに複数のフィーチャーを追加した場合、照会で 1 つまたは複数のフィーチャーに与える重みを指定することができます。フィーチャーの重みを指定するには、QbQuerySetFeatureWeight API を使用します。フィーチャーの重みは、イメージ・エクステンダーが類似性得点を計算し、イメージ内容による照会の結果を戻すときに、フィーチャーに置く強調を示します。フィーチ

ャーに指定する重みが大きければ大きいほど、照会オブジェクトに組み込まれたそのフィーチャーに対する強調が大きくなります。

照会オブジェクトに組み込まれた 1 つまたは複数のフィーチャーの重みを指定できますが、QbQuerySetFeatureWeight API を出すたびに、1 つのフィーチャーの重みしか指定できません。照会オブジェクトでフィーチャーに重みを割り当てない場合、イメージ・エクステンダーはそのフィーチャーの省略時の重みを使用します。フィーチャーが 1 つの照会ストリングに指定された唯一のフィーチャーである場合、重みの割り当ては無意味です。フィーチャーが 1 つの照会オブジェクトの指定された唯一のフィーチャーである場合、重みの割り当ては無意味です。 (そのフィーチャーは照会オブジェクト内で常にすべての重みを持ちます。)

この API を使用するときは、次のようにします。

- 照会オブジェクト・ハンドルを指定する。
- フィーチャー名を指定する。
- フィーチャーの重みをポイント指定する。重みは、0 より大きい実数 (たとえば、2.5 または 10.0) にセットすることができます。指定する値が高ければ高いほど、そのフィーチャーに対する強調が大きくなります。この設定は、照会オブジェクト内のフィーチャーに以前セットされた重みを変更します。

次の例では、照会オブジェクトに平均色フィーチャーと少なくとも 1 つの他のフィーチャーが含まれています。 QbQuerySetFeatureWeight API を使って、照会オブジェクトに組み込まれた平均色フィーチャーの重みを指定します。

```
char                 featureName[qbiMaxFeatureName];
double               weight;
QbQueryObjectHandle  qoHandle;
strcpy(featureName,"QbColorFeatureClass");
weight=5.00;
rc=QbQuerySetFeatureWeight(
                qoHandle,                          /* query object handle */
                featureName,                       /* feature name */
                &weight);                          /* feature weight */
```

### 照会ストリングの保管と再利用

照会オブジェクトは、保管しない限り一時的なものです。照会オブジェクトは、1 度のデータベース接続の間だけ存在します。照会によって取得した照会ストリングを保管して、プログラム内で再び使用することができます。また、現行のデータベース接続を除去した後や、別のプログラムを呼び出している間にも使用することもできます。

イメージ・エクステンダーには、照会オブジェクトから照会ストリングを戻す QbQueryGetString API があります。それで、その照会ストリングを QbQueryStringSearch API への入力として、またはイメージ内容による他の照会で QbScoreFromStr および QbScoreTBFromStr UDF への入力として使用することができます (174ページの『イメージ内容による照会の実行』 を参照)。

照会ストリングは、以下の API を使用して照会を作成するときに作成されます。

- QbQueryCreate
- QbQueryAddFeature
- QbQuerySetFeatureData
- QbQuerySetFeatureWeight
- QbQueryRemoveFeature

照会を作成した後、QbQueryGetString を呼び出して、そのストリングを入手することができます。この照会ストリングを、そのプログラム中の呼び出しで使用したり、ファイルに保管しておいて、アプリケーションのその後の呼び出しや他のデータベース接続で使用したりすることができます。 QbQueryGetString によって戻された照会ストリングの使用を終えたならば、スペースを明示的に解放する必要があります。

次の例では、QbQueryGetString を使って、照会オブジェクトから照会ストリングを取り出します。

```
SQLRETURN rc;
char* qryString;
QbQueryHandle  qHandle;
.....           /* Here you create and use the query */
rc = QbQueryGetString(qHandle, &qryString);
if ( rc == 0) {
   ...         /* Use the query string as input here  */
   free((void *)qryString);
   qryString=(char *)0;
}
```

**制約事項:** クライアント・ファイルを使用してフィーチャーのデータ・ソースを指定する場合、照会ストリングはそのフィーチャー・データを反映しません。

### 照会オブジェクトに関する情報を取り出す

どのようなフィーチャー (もしあれば) が照会オブジェクトに追加されているかを知ることができます。さらに、フィーチャーの現在の重みも知ることができます。

| API | 取り出す情報 |
|---|---|
| QbQueryGetFeatureCount | 照会オブジェクトに組み込まれているフィーチャーの数 |
| QbQueryListFeatures | 照会オブジェクトに組み込まれているフィーチャーの名前 |

QbQueryGetFeatureCount API を出す場合には、照会オブジェクト・ハンドルを指定します。さらに、カウンターをポイント指定する必要があります。イメージ・エクステンダーはフィーチャー数をこのカウンターに戻します。

次の例では、QbQueryGetFeatureCount API を使って、照会オブジェクトに組み込まれているフィーチャー数を判別します。

```
SQLINTEGER     count;
QbQueryHandle  qHandle;
rc=QbQueryGetFeatureCount(
                qHandle,                              /* query object handle */
                &count);                              /* feature count */
```

QbQueryListFeatures API 呼び出しを出す場合には、戻されるフィーチャー名が入るバッファーを割り振る必要があります。さらに、カタログ・ハンドルと、戻されるフィーチャー名のバッファー・サイズを指定する必要があります。

次の例では、QbQueryListFeatures API を使って、照会オブジェクトに組み込まれている各フィーチャーの名前を取り出します。

```
SQLINTEGER     retCount,bufSize;
char*          featureName;
QbQueryHandle  qHandle;
bufSize=qbiMaxFeatureName;
featureName=(char*)malloc(bufSize);
rc=QbQueryListFeatures(
                qHandle,                              /* query object handle */
                bufSize                               /* size of buffer */
                &retCount,                            /* feature count */
                featureName);                         /* buffer for feature names */
```

### フィーチャーを照会オブジェクトから除去する

フィーチャーを照会オブジェクトから除去するには、QbQueryRemoveFeature API を使用します。 この API を使用する場合には、照会オブジェクト・ハンドルとフィーチャーの名前を指定します。

次の例では、QbQueryRemoveFeature API を使って、ヒストグラム色というフィーチャーを照会オブジェクトから除去します。

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;
strcpy(featureName,"QbColorHistogramFeatureClass");
```

```
rc=QbQueryRemoveFeature(
                qHandle,                              /* query object handle */
                featureName);                         /* feature name */
```

### 照会オブジェクトを削除する

名前の付いていない照会オブジェクトを削除するには、QbQueryDelete API を使用します。 イメージ・エクステンダーは、現在接続されているデータベースからその照会を削除します。

QbQueryDelete API を使用する場合には、照会オブジェクト・ハンドルを指定します。

次の例では、QbQueryDelete API を使って照会オブジェクトを削除します。

```
QbQueryHandle  qHandle;
rc=QbQueryDelete(
            qHandle);                                 /* query object handle */
```

名前付きの照会を使用していた場合には、QbQueryNameDelete API を使って照会オブジェクトを削除してください。

## イメージ内容による照会の実行

画像をカタログした後で、1 つまたは複数の画像を内容によって照会することができます。内容によって画像を照会するときは、その照会の入力と、カタログ・イメージのターゲット・セットを指定します。入力は、照会ストリング (162ページの『照会ストリングを指定する場合』を参照) または照会オブジェクト (166ページの『照会オブジェクトの使い方』を参照) に指定することができます。

照会ストリングを使用する場合、DB2 コマンド行から、またはプログラム内から照会を実行依頼することができます。照会オブジェクトを使用する場合、そのハンドルを参照することによって、プログラム内から照会を実行依頼することができます。

イメージ・エクステンダーは、その照会に指定されたフィーチャー値をターゲット・イメージのそれと比較し、各画像の得点を計算します。この得点は、ターゲット・イメージのフィーチャー値が、照会に指定されたフィーチャー値にどの程度似ているかを表します。

フィーチャー値が照会に最も似ている画像を取り出すことができます。あるいは、単一のカタログ・イメージを照会してその得点だけを入手したり、表列内のすべてのカタログ・イメージの得点を入手することができます。

## 画像の照会

イメージ・エクステンダーには、表列にあるカタログ・イメージを照会するための API が 3 つ用意されています。これらの API は、入力として照会ストリングまたは照会オブジェクトを必要としているかどうかだけが異なります。

| API | 入力 |
|---|---|
| QbQueryStringSearch | 照会ストリング |
| QbQuerySearch | 照会オブジェクト・ハンドル |
| QbQueryNameSearch | 照会オブジェクト名 |

3 つのすべての API において、次のことも行う必要があります。

- 探索する画像が入っている表と列を指定する。これらの画像は QBIC カタログにカタログされていなければなりません。
- 戻される結果の最大数を指定する。
- 照会の範囲を指定する構造体をポイント指定する。このポインターを 0、NULL 値、または空ストリングに指定すると、その表の列にあるすべてのカタログ・イメージが探索されます。
- 結果を配列に保管するには、定数 qbiArray を指定する。定数 qbiArray は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に定義されています。

さらに、この探索の結果を入れる出力構造体配列をポイント指定する必要があります。この結果、イメージ・エクステンダーは、フィーチャー値が照会のフィーチャー値に最も似ているターゲット・イメージのハンドルをこれらの構造体に入れて戻します。さらに、イメージ・エクステンダーは、その画像のフィーチャー値が照会にどのくらい似ているかを示す得点を画像ごとに戻します。この構造体は、QBIC のインクルード (ヘッダー) ファイル dmbqbapi.h に次のように定義されています。

```
typedef struct{
          char       imageHandle[MMDB_BASE_HANDLE_LEN+1];
          SQLDOUBLE  SCORE
} QbResult;
```

指定する結果の最大数が入るだけの配列を割り振り、その配列を API で指定する必要があります。さらに、カウンターをポイントしなければなりません。イメージ・エクステンダーは、戻す結果の数をこのカウンターに設定します。

次の例では、QbQueryStringSearch API を使用して、表列内のカタログ・イメージを内容によって照会します。照会範囲へのポインターが値 0 に設定されていることに注意してください。

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];
rc=QbQueryStringSearch(
          "QbColorFeatureClass color=<255, 0, 0>" /*query string */
          "employee",                          /* user table */
          "picture",                           /* image column */
          maxResults,                          /* maximum number of results */
          0,                                   /* query scope pointer * /
          qbiArray,                            /* store results in an array */
          &count,                              /* count of returned images */
          results);                            /* array of returned results */
```

QbQuerySearch API を使用する要求を以下に示します。入力として照会オブジェクト・ハンドルが指定されていることに注意してください。

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];
rc=QbQuerySearch(
                 qHandle,                      / query object handle */
                "employee",                    /* user table */
                "picture",                     /* image column */
                maxResults,                    /* maximum number of results */
                0,                             /* query scope pointer * /
                qbiArray,                      /* store results in an array */
                &count,                        /* count of returned images */
                results);                      /* array of returned results */
```

## 画像の得点の取り出し

イメージ・エクステンダーには、表列内のカタログ・イメージの得点を取り出すために、SQL ステートメントで使用できる 4 つの UDF があります。得点は倍精度の浮動小数点数で、値は 0.0 から無限大までです。得点が低ければ低いほど、その画像のフィーチャー値は、照会に指定したフィーチャー値に類似しています。値 0.0 は、その画像が厳密に一致していることを意味します。

以下の UDF があります。

- QbScorefromStr
- QbScoreTBfromStr
- QbScoreFromName
- QbScoreTBFromName

**推奨事項**: 単一のカタログ・イメージの得点を入手するには、QbScoreFromStr UDF を使用します。表列内の複数のカタログ・イメージの得点を入手するには、QbScoreTBFromStr UDF を使用します。

### 単一画像の得点の取り出し

表列内の単一のカタログ・イメージの得点を入手するには、QbScoreFromStr UDF を使用します。 QbScoreFromStr UDF への入力としては照会ストリングを指定します。 QbScoreFromName UDF を使用する場合には、QbScoreFromName UDF への入力として照会オブジェクトの名前を指定します。いずれの UDF を使用する場合でも、ターゲット・イメージを含んでいる表列の名前を指定します。

次の照会では、QbScoreFromStr UDF を使用して、表列のカタログ・イメージで、平均色が赤に最も近いものを見つけます。

```
SELECT name, description
decimal (QbScoreFromStr(swatch_img,
            'QbColorFeatureClass color=<255, 0, 0>'),   /* query string *
                  10, 5) AS score
FROM fabric                                             /* table column */
ORDER BY score
```

### 複数の画像の得点の取り出し

表列内の複数のカタログ・イメージの得点を入手するには、QbScoreTBFromStr UDF を使用します。名前付きの照会の場合には、QbScoreTBFromName UDF を使用することができます。 2 つの UDF は、画像ハンドルと得点からなる 2 列の表を戻します。表内の行は得点の昇順になります。結果の表のハンドル列の名前は IMAGE_ID であり、得点列の名前は SCORE です。

QBScoreTBFromStr UDF に対する入力として、照会ストリングを指定します。QbScoreTBFromName UDF への入力としては照会オブジェクトの名前を指定します。いずれの UDF を使用する場合でも、ターゲット・イメージを含んでいる表と列の名前を指定します。結果の表に戻す行の最大数も指定することができます。結果の最大数を指定しない場合、UDF はターゲット表列内のカタログ・イメージごとに 1 行を戻します。

次の照会では、QbScoreTBFromStr UDF を使用して、表列のカタログ・イメージで、テクスチャーがサーバー・ファイルの画像のテクスチャーに最も近いものを 10 個見つけます。

```
SELECT name, description
FROM fabric
WHERE CAST (swatch_img as varchar(250))IN
 SELECT CAST (image_id as varchar(25)) FROM TABLE
```

```
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'   /*query string */
      'fabric',                                               /* table */
      'swatch_img',                                           /* table column */
      10))                                          /* maximum number of results */
AS T1));
```

## QBIC 照会のサンプル・プログラム

179ページの図27 は、QBIC 照会を作成して実行する C プログラムの一部を示したものです。この図のコードでは、画像を平均色によって照会します。ユーザーは、色またはイメージ・ファイルの名前の入力を求められます。ユーザーは、照会によって戻される画像をサンプル・イメージとして使って、それ以後の照会を行うこともできます。プログラムは、その指定された色か、その画像の色を平均色として使って、列の画像を照会します。

このプログラムの全体は、SAMPLES サブディレクトリーの QBICDEMO.C ファイルにあります。このプログラムでは、ヒストグラム色か位置色で画像を照会することもできますし、平均色で照会することもできます。プログラム全体を実行するには、ENABLE、POPULATE、QBCATDMO の各サンプル・プログラム (SAMPLES サブディレクトリーにあります) を実行する必要があります。サンプル・プログラムの詳細については、623ページの『付録B. サンプル・プログラムとメディア・ファイル』を参照してください。

179ページの図27 の次の点に注目してください。

**1** dmbqbapi ヘッダー・ファイルをインクルードする。

**2** データベース情報をユーザーに要求する。

**3** データベースに接続する。

**4** 照会オブジェクトを作成する。

**5** 照会オブジェクトにフィーチャーを追加する。

**6** 入力のタイプをユーザーに要求する (色名、イメージ・ファイル、または前に取り出した画像)。

**7** フィーチャーのデータ・ソースを指定する。データ・ソースは、平均色の明示的な指定です。

**8** 照会を行う。イメージ・エクステンダーは、その列全体の画像を探索します。さらに、戻す画像の最大数として 10 を指定しています。

**9** 戻される画像群の中の次の画像を表示する。画像の表示については、さらに 141ページの『フルサイズの画像やビデオ・フレームの表示』を参照してください。

**10** 照会オブジェクトを削除する。

SAMPLES サブディレクトリーには、QBIC 照会を作成し使用する方法を示す別のプログラムが含まれています。プログラム QbicQry.java は、QBIC 照会の探索基準を図によって指定する方法を示しています。たとえば、このプログラムは平均色を選択するためのカラー・セレクターを示しています。プログラムはこの選択を照会ストリングに変換します。

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <color.h>
#include <ctype.h>
#define   MaxQueryReturns        10
#define   MaxDatabaseNameLength  SQL_SH_IDENT
#define   MaxUserIdLength        SQL_SH_IDENT
#define   MaxPasswordLength      SQL_SH_IDENT
#define   MaxTableNameLength     SQL_LG_IDENT
#define   MaxColumnNameLength    SQL_LG_IDENT
static  char       databaseName[MaxDatabaseNameLength+1];
static  char       userid[MaxUserIdLength+1];
static  char       password[MaxPasswordLength+1];
static  char       tableName[MaxTableNameLength+1];
static  char       columnName[MaxColumnNameLength+1];
static  char       line[4000];
static  QbResult   results[MaxQueryReturns];
static  long       currentImage   = -1;
static  long       imageCount  = 0;
static  char*      tableAndColumn;
static  QbQueryHandle averageHandle = 0;
static  QbQueryHandle histogramHandle = 0;
static  QbQueryHandle drawHandle = 0;
static  QbQueryHandle lastHandle = 0;
static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;
static  char*    listQueries =
"SELECT NAME,DESCRIPTION FROM MMDBSYS.QBICQUERIES ORDER BY NAME";
static  char*    menu[] = {
```

*図 27. QBIC 照会のサンプル・プログラム (1/6)*

```
/*
 1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890 */
"",
"+-------------------------------------------------------------------------------+",
"| AVERAGE COLOR colorname                                                       |",
"| AVERAGE FILE filename format                                                  |",
"| AVERAGE LAST                                                                  |",
"| Press Enter to display the next image in the series                           |",
"+-------------------------------------------------------------------------------+",
"",
0
};
static  char*       help[] = {
"",
"AVERAGE     Execute an average color query",
"  COLOR       Specifies the color to query for",
"  FILE        Specifies the file to compute the average color from",
"  LAST        Specifies the last displayed image be used to compute the color",
"  NEXT        Displays the next image from the current query or nothing if",
"                all of the image have been displayed."
"",
">>pause<<",
0
};
/*****************************************************************************/
/* doNext()                                                                  */
/*****************************************************************************/
static void doNext(void)
{
int ret;
   if (currentImage < imageCount)
      currentImage++;
   if (currentImage < imageCount)
      ret = DBiBrowse("/usr/local/bin/xv %s", MMDB_PLAY_HANDLE,
            results[currentImage].imageHandle, MMDB_PLAY_NO_WAIT); 9
}
```

*図 27. QBIC 照会のサンプル・プログラム (2/6)*

```
/*****************************************************************************/
/* doAverage()                                                               */
/*****************************************************************************/
static void doAverage(void)
{
   QbQueryHandle qohandle = 0; QbImageSource is; char* type;
   char* arg1; char* arg2;
   type = nextWord(0);
   if (abbrev(type, "color", 1)) {
      is.type = qbiSource_AverageColor;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE COLOR command requires a colorname argument.¥n");
         return;
      }
      if (getColor(arg1, &is.averageColor) == 0) {
         printf("The colorname entered was not recognized.¥n");
         return;
      }
   }
   else if (abbrev(type, "file", 1)) {
      is.type = qbiSource_ClientFile;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE FILE command requires a filename argument.¥n");
         return;
      }
      arg2 = nextWord(0);
      if (arg2 == 0) {
         printf("AVERAGE FILE command requires a file format argument.¥n");
         return;
      }
      strcpy(is.clientFile.fileName, arg1);
      strcpy(is.clientFile.format, arg2);
   }
   else if (abbrev(type, "last", 1)) {
      is.type = qbiSource_ImageHandle;
      if (0 <= currentImage &&; currentImage < imageCount)
         strcpy(is.imageHandle, results[currentImage]imageHandle);
      else {
         printf("No last image for AVERAGE LAST command¥n");
         return;
      }
   }
```

*図 27. QBIC 照会のサンプル・プログラム (3/6)*

```
   else {
      printf("AVERAGE command only supports COLOR, FILE, and LAST types.¥n");
      return;
   }

   _QbQuerySetFeatureData(averageHandle, "QbColorFeatureClass", &is); 7
   _QbQuerySearch(averageHandle, tableAndColumn, "IMAGE",
         MaxQueryReturns, 0, 0, &imageCount, results); 8
   lastHandle = averageHandle;

   currentImage = -1;
}
/*****************************************************************************/
/* commandLoop()                                                             */
/*****************************************************************************/
void commandLoop(void)
{
   int done = 0;
   while (!done) { 6
      displayText(menu);
      printf("%d", currentImage + 1);
      if (0 <= currentImage &&; currentImage < imageCount)
            printf("  %8.6f", results[currentImage].score);
      printf("> ");
      gets(line);
      done = processCommand(line);
   }
}
```

*図 27. QBIC 照会のサンプル・プログラム (4/6)*

```
/*****************************************************************************/
/* main()                                                                    */
/*****************************************************************************/
void main(void)
{
   char* inst;
   int i;
   printf("¥n¥n");
   printf("Please enter: database_name [user_id] [password] "¥n"); 2
   gets(line);
   if (copyWord(line, databaseName, sizeof(databaseName)) == 0)
      exit(0);
   copyWord(0, userid, sizeof(userid));
   copyWord(0, password, sizeof(password));
   printf("¥n");

   if (SQLAllocEnv(&henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLAllocConnect(henv, &hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLConnect(hdbc, 3
                  (SQLCHAR*)databaseName,
                  SQL_NTS,
                  (SQLCHAR*)userid,
                  SQL_NTS,
                  (SQLCHAR*)password,
                  SQL_NTS) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   printf("Initializing . . .¥n");
```

*図 27. QBIC 照会のサンプル・プログラム (5/6)*

```
   inst = getenv("DB2INSTANCE");
   if (inst != 0 &&; strcmp(inst, "keeseyt") == 0)
      tableAndColumn = "KEESEY.TEST";
   else
      tableAndColumn = "QBICDEMO.TEST";
   _QbQueryCreate(&averageHandle); 4
   _QbQueryAddFeature(averageHandle, "QbColorFeatureClass");
   _QbQueryCreate(&histogramHandle);
   _QbQueryAddFeature(histogramHandle, "QbColorHistogramFeatureClass");
   _QbQueryCreate(&drawHandle);
   _QbQueryAddFeature(drawHandle, "QbDrawFeatureClass"); 5
   commandLoop();

   _QbQueryDelete(drawHandle);
   _QbQueryDelete(histogramHandle); 10
   _QbQueryDelete(averageHandle);
   if (SQLDisconnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeConnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeEnv(henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
}
```

*図 27. QBIC 照会のサンプル・プログラム (6/6)*

# 第14章 ビデオ・シーンの変化の検出

この章では、DB2 ビデオ・エクステンダー API を使ってビデオ・クリップのシーン (場面) の変化を検出する方法について説明します。これらの API は、Windows 3.1 を除くすべての DB2 ビデオ・エクステンダーのプラットフォームで使用できます。ビデオ・シーンの変化の検出は、MPEG-1 形式のビデオ・クリップについてのみサポートされます。

## ビデオ・シーン (場面) の変化とは

後で放送するために番組をビデオ・テープに録画するテレビ・スタジオを想像してください。最近、このスタジオでは、ビデオ・テープのクリップをビデオ・エクステンダーを使って DB2 データベースに格納するようになりました。これによって、スタジオのスタッフは、番組について、従来のタイプの情報を照会できるだけでなく、そのクリップを見ることができるようになりました。

このスタジオでは、ビデオ・クリップをプリビューする機能を必要としています。ストーリーボードと呼ばれる眼に見える要約を表示する機能を必要としています。ストーリーボードの例を 186ページの図28 に示してあります。スタジオのスタッフは、ストーリーボードを見れば、ビデオ全体を見ないで、ビデオの要旨を入手することができます。さらに、そのビデオが必要なビデオであるかどうか (たとえば、ダウンロードして見る価値があるかどうか) を判断することもできます。このような要件は、スタジオにとって非常に重要です。ビデオ全体ではなくストーリーボードを見ることによって、ダウンロードして見る時間を大幅に削減することができます。ビデオ・シーンの変化の検出機能をこのように使用する方法については、204ページの『ビデオ内のすべてのショットに関する情報の保管』を参照してください。

## シーンの変化

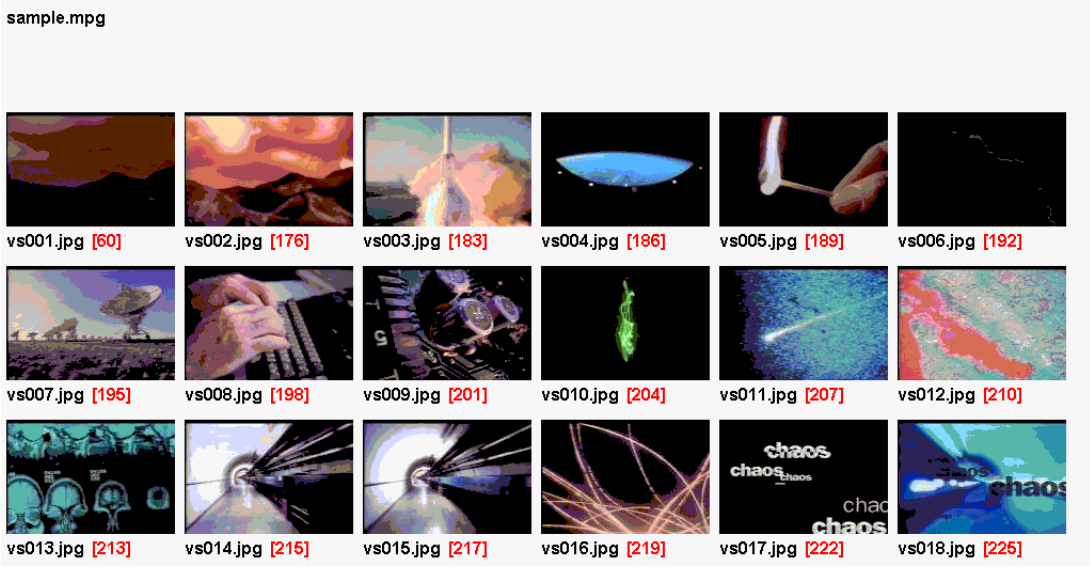



*図 28. ビデオのストーリーボード.* 代表フレームはビデオの内容と流れを要約します。

このスタジオでは、ビデオ・エクステンダーのビデオ・シーンの変化を検出する機能を使って、これらのストーリーボードの代表的なフレームを取り込むことを計画しています。

**ビデオ・シーンの変化**とは、ビデオ・クリップにおいて連続する 2 つのシーンの間に顕著な違いがあるポイントをいいます。たとえば、このような状態は、ビデオの収録中にカメラが視点を変えたときに起こります。シーンが変化してから次に変化するまでの間のフレーム (複数) が**ショット**を構成します。

ビデオ・エクステンダーは、シーンの変化[6]を検出すると、それに対応するショットのデータを記録します。このデータには、そのショットが始まるフレームの番号、そのショットが終わるフレームの番号、そのショットの代表的なフレームの番号などがあります。さらに、このショット・データには、代表的フレームのピクセル内容が含まれます。

6. ビデオ・シーンの変化検出コードには、カリフォルニア大学バークレー校の MPEG デコーダーと、ボストン大学 Multimedia Communication Laboratory による修正が含まれます。

## シーンの変化の検出と使用

一連のビデオ・エクステンダーのアプリケーション・プログラミング・インターフェースを使えば、ビデオ・クリップのショットやフレームを検出することができます。ショットやフレームを検出したら、そのデータにアクセスすることができます。たとえば、そのショットの始めと終わりのフレーム番号や、フレームのピクセル内容です。次にこの情報をプログラムに渡して、さらに処理を行うことができます。たとえば、フレームの内容をプログラムに渡すことによって、それを表示することができます。

さらに、ビデオ・エクステンダーには、**ショット・カタログ**にショット・データを保管するための API があります。ショット・カタログはデータベース内にあっても、ファイルにあってもかまいません。ファイル内のショット・カタログにアクセスすることも、データベース内のショット・カタログの読み取り専用視点にアクセスすることもできます。

ショット・カタログ・ファイルには以下のデータに関するフィールドが含まれます。

- ショット・カタログ名
- ビデオ・エクステンダーがショットを検出する方法を制御する値 (たとえば、ショット内のフレームの最小番号)
- ショットの代表フレームとして、どのようなフレームをいくつ保管するかを制御する値
- ショットの番号
- 開始フレームの番号
- 終了フレームの番号
- 代表フレームの番号
- 代表フレームの内容が入っているファイルの名前

データベース内のショット・カタログの視点には、以下のデータの列が含まれています。

- ショット・ハンドル
- ビデオ表の名前
- ビデオ列の名前
- ビデオ・ハンドル
- ビデオ・ファイルの名前
- 開始フレームの番号
- 終了フレームの番号

- 代表フレームの番号
- 代表フレームのデータ
- 注釈

ショット・カタログがデータベースにある場合、ショット・カタログ・ファイルのデータにアクセスしたり、データを照会したりすることができます。代表フレーム情報は、ストーリーボードを表示する場合に特に便利です。さらに、ショット・カタログがデータベースにある場合には、ショット・データと他の表の関連データを結合することができます。たとえば、ビデオ・スタジオのスタッフは、ショット・カタログをデータベース内に作成することができます。そうすると、そのカタログ・データと、ビデオ・クリップとクリップに関する情報をもつ表とを結合します。このようにすれば、1 つの照会によって、そのクリップに関するクリップ情報と業務情報を取り出せるだけでなく、そのクリップ内のショットを識別することもできます。

## ショット検出のデータ構造

ショット検出に関連するデータは、構造体 (これはショット検出ヘッダー・ファイル dmbshot.h にあります) に保管されます。多くのショット検出 API では、これらの構造体の 1 つまたはそれ以上をポイントする必要があります。これらの構造体のいくつかには、ビデオ・エクステンダーが入力として使用するデータを指定します。たとえば、ショット制御構造体には、ショット検出を制御する情報を指定します。ほとんどの構造体は、ビデオ・エクステンダーがビデオから取り出したデータを保管するために使用されます。たとえば、ビデオ・フレーム・データ構造には、フレームのピクセル内容が入ります。

ショットの検出に使用する構造体には、DBvIOType、DBvShotControl、DBvShotType、DBvFrameData、および DBvStoryboardCtrl があります。

### DBvIOType

DBvIOType データ構造には、ビデオに関する情報 (その形式、寸法、フレーム数など) が入ります。このデータ構造は次のように定義されています。

```
typedef struct {

   FILE *hFile;                    /* file handle for the video */
   char vhandle[255];              /* video handle (if from database)
   char vtable[255];               /* video table name (if from database) */
   char vcolumn[255];              /* video column name (if from database) */
   char vFile[255];                /* name of video file */
   char idxFile[255];              /* name of index file */
   char isIdx;                     /* 1 if the index exists, 0 otherwise */
   char isInDb;                    /* 1 if from DB, 0 if from file */
   int format;                     /* Format of the video */
   unsigned long dx, dy;           /* Dimensions of the video */
   unsigned long totalFrames;      /* TotalFrames in the video */
   unsigned long markFrame;        /* used by shot detection */
```

```
   unsigned long currentFrame;     /* The current video frame */
   DBvFrameData fd;                /* Frame data for current frame */
   DBvDCFrameData fdDc;            /* Frame data for DC images */
   unsigned char BGRValid;         /* reserved */
   unsigned short usDeviceID;      /* reserved */
   unsigned long hwnd;             /* reserved */
   int videoReset;                 /* Flag if video is opened or seeked */
   int firstshot;                  /* Used internally to indicate the first call */
   void *reserved                  /* reserved */

} DBvIOType;
```

### DBvShotControl

DBvShotControl データ構造には、ショット検出を制御するための情報 (検出方式など) が入ります。このデータ構造は次のように定義されています。

```
typedef struct {

  unsigned long reserved;
  unsigned long method;           /* detection method */

    #define DETECT_CORRELATION  0x00000001
    #define DETECT_HISTOGRAM    0x00000002
    #define DETECT_CORRHIST     0x00000003
    #define DETECT_CORRHISTDISS 0x00000004


  int normalCorrValue;            /* Correlation threshold */
  int sceneCutSkipXY;             /* reserved */
  int CorrHistThresh;             /* Histogram threshold */
  int DissThresh;                 /* Dissolve threshold */
  int DissCacheSize;              /* Dissolve cache size */
  int DissNumCaches;              /* Dissolve cache number */
  int minShotSize;                /* Minimum frames in a shot */

} DBvShotControl;
```

表11 は、DBvShotControl の各フィールドとその指定できる設定値と省略時の設定値を示しています。これらのフィールドを省略時値に初期設定するには、193ページの『ショット検出データ構造内の値の初期設定』に説明してある DBvInitShotControl API を使用してください。

**DBvShotControl の設定値はビデオのタイプによって異なります:** ディジタル化ビデオにおけるシーン変化は、ビデオの内容と形式に大きく依存します。さらに、シーン変化アルゴリズムの正確性もビデオによって異なります。シーン変化が明確に定義され、全体的なフレーム表示において明らかな違いがあれば、それより変化が微妙なものや、全体的な色内容が変わらないものよりも正確に検出されます。省略時の DBvShotControl フィールドの設定値はほとんどのアプリケーションで有効ですが、誤った検出をしたり、検出をしないことが少なくなるように、これらの設定値を調整する必要がある場合があります。

*表 11. DBvShotControl フィールド*

| フィールド | 意味 |
|---|---|
| method | シーン変化を検出するためにビデオ・エクステンダーが使用する方式を指定します。次の方式から 1 つを選択することができます。<br><br>DETECT_CORRELATION。 連続する 2 つのフレームのピクセルを比較します。この相違が相関しきい値を超えると、シーン変化が検出されます。<br><br>DETECT_HISTOGRAM。 連続する 2 つのフレームのそれぞれのヒストグラム値を比較します。このヒストグラム値は、そのフレームにおける色の分散を示します。この相違がヒストグラムしきい値を超えると、シーン変化が検出されます。<br><br>DETECT_CORRHIST。まず、相関方式を使ってシーン変化の可能性があるフレームを識別し、次に、ヒストグラム相関を使ってそれらのフレームを検査します。その値がヒストグラムしきい値よりも大きいと、シーン変化が検出されます。<br><br>DETECT_CORRHISTDISS。 DETECT_CORRHIST と同じですが、さらにディゾルブ (ある場面から次の場面へダブりながら映像が転換すること) を探すために、他のフレームを調べます。<br><br>省略時のメソッドは DETECT_CORRHIST です。 |
| normalCorrValue | 相関しきい値を指定する 0 から 100 の整数値です。これは、2 つのフレームのピクセル間の相関係数の最小値を与えます。値に 0 を指定すると、次のフレームに対し常にシーン変化が検出されます。値に 100 を指定すると、あるフレームと次のフレームですべてのピクセルが変化した場合にのみ、シーン変化が検出されます。省略時値は 60 です。 |
| sceneCutSkipXY | 予備。 |
| CorrHistThresh | ヒストグラムしきい値を指定する 0 から 100 の整数値です。この値は、連続するフレーム間のヒストグラム値の相違を示します。 値に 0 を指定すると、あるフレームと次のフレームでヒストグラム値が全く異なるときだけ、シーン変化が検出されます。値に 100 を指定すると、次のフレームに対し常にシーン変化が検出されます。省略時値は 10 です。 |
| DissThresh | テストしきい値を指定する 0 から 100 の整数値です。あるフレームでディゾルブ・テストに合格するピクセルの割合がこの値以上だと、ディゾルブが検出されます。値に 0 を指定すると、そのフレームに対しディゾルブが常に検出されます。値に 100 を指定すると、そのフレームのすべてのピクセルがディゾルブ・テストに合格する場合だけ、ディゾルブが検出されます。省略時値は 15 です。 |
| DissCacheSize | ディゾルブ・テストのスロープ部分で使用されるフレーム数を指定する整数値です。省略時値は 4 です。 |

*表 11. DBvShotControl フィールド (続き)*

| フィールド | 意味 |
|---|---|
| DissNumCaches | ディゾルブ・テストの一定部分で使用されるフレーム数を指定する整数値です。省略時値は 7 です。 |
| minShotSize | ショットの最少フレーム数を指定する整数値です。ショットが検出されるためには、そのショットのフレーム数が少なくとも最少数と同じでなければなりません。省略時値は 5 です。 |

### DBvShotType

DBvShotType データ構造には、ショットに関する情報が入ります。たとえば、その開始フレーム番号、終了フレーム番号、代表的なフレームの番号、それに、その代表的なフレームのピクセル内容を指すポインターなどです。このデータ構造は次のように定義されています。

```
typedef struct {

  unsigned long startFrame;       /* starting frame number */
  unsigned long endFrame;         /* ending frame number */
  unsigned long repFrame;         /* representative frame number */
  DBvFrameData fd;                /* data for representative shot */
  unsigned long dx;               /* frame data width in pixels */
  unsigned long dy;               /* frame data height in pixels */
  char *comment;                  /* shot remark */

} DBvShotType;
```

### DBvFrameData

DBvFrameData データ構造には、フレームのピクセル内容が入ります。このデータ構造は次のように定義されています。

```
typedef struct                     /* video frame data */
{
  /* MPEG 1 pixels */
  unsigned char *luminance;        /* Luminance pixel plane (black and white) */
  unsigned char *Cr;               /* Cr pixel plane */
  unsigned char *Cb;               /* Cb pixel plane */
  unsigned char *reserved;

  } DBvFrameData;
```

### DBvStoryboardCtrl

DBvStoryboardCtrl データ構造には、ショットのどの代表フレームをいくつビデオ・カタログに保管するかを制御する値が入ります。これらの値を使用する方法については、206ページの『ストーリーボードの作成』を参照してください。このデータ構造は次のように定義されています。

```
typedef struct {

  int thresh1;                    /* threshold for small to medium scenes */
  int thresh2;                    /* threshold for medium to large scenes */
  int delta;                      /* offset used for representative frames */

} DBvStoryboardCtrl;
```

表12 は、DBvStoryboardCtrl の各フィールドとその省略時の設定値を示しています。これらのフィールドを省略時値に初期設定するには、193ページの『ショット検出データ構造内の値の初期設定』に説明してある DBvInitStoryboardCtrl API を使用してください。

**DBvStoryboardCtrl の設定値は、ビデオのタイプによって異なります:** ストーリーボードに最適な代表フレームとその数は、ビデオのタイプによって異なります。省略時値の DBvStoryboardCtrl フィールドの設定値は多くのタイプのビデオで有効ですが、これらの設定値をテスト・サブセットのビデオに使用した方がよいでしょう。そうすると、必要に応じて設定値を調整してから、より広い範囲の一連のビデオに関するストーリーボードを作成することができます。

*表 12. DBvStoryboardCtrl フィールド*

| フィールド | 意味 |
|---|---|
| thresh1 | ショート・ショットのしきい値を指定します。 thresh1 の値より少ないフレームを持つショットは、ショート・ショットです。ショート・ショットに関する情報をカタログする場合には、1 つの代表フレーム (中間フレーム) をその情報に含めます。<br><br>省略時値は 90 です。 thresh1 の値を -1 に設定すると、そのショットはショート・ショットとみなされます (実際の長さに関係なく)。 |

*表 12. DBvStoryboardCtrl フィールド (続き)*

| フィールド | 意味 |
|---|---|
| thresh2 | ミディアムからロング・ショットのしきい値を指定します。 thresh2 の値以下で、thresh1 の値以上の値を持つショットは、ミディアム・ショットとみなされます。ミディアム・ショットの情報をカタログする場合には、その情報には 2 つの代表フレームが含まれます。代表フレームの位置は、delta フィールドの値によって制御されます。 thresh2 の値より多いフレームを持つショットは、ロング・ショットです。ロング・ショットの情報をカタログする場合には、その情報には 3 つの代表フレームが含まれます。最初と最後の代表フレームの位置は、delta フィールドの値によって制御されます。 2 番目のフレームは中間フレームです。<br><br>省略時値は 150 です。 thresh2 の値が -1 にセットされている場合、そのショットはショート・ショットとみなされます (実際の長さに関係なく)。 |
| delta | 代表フレームに使用するオフセットを指定します。ミディアムおよびロング・ショットの場合、最初の代表フレームは、ショットの先頭から delta に指定したフレーム数だけオフセットされます。最後の代表フレームは、ショットの最後から delta に指定したフレーム数だけオフセットされます。<br><br>省略時値は 5 です。 |

### ショット検出データ構造内の値の初期設定

DBvShotControl データ構造内の値はショット検出を制御します。 DBvStoryboardCtrl データ構造内の値は、ストーリーボードの作成を制御します。これらのデータ構造内のフィールドの値を明示的に指定することができます。さらに、これらの構造内の値を省略時値に初期設定することができます。 DBvShotControl データ構造内の省略時値については、190ページの表11 を参照してください。 DBvStoryboardCtrl データ構造内の省略時値については、192ページの表12 を参照してください。

DBvInitShotControl API を使用すれば、DBvShotControl データ構造内の値を初期設定することができます。この API を使用するときは、ショット制御構造を指定する必要があります。たとえば、次のステートメントでは、 DBvShotControl 構造内の各フィールドを省略時値に初期設定します。

```
DBvShotControl    shotCtrl;

rc=DBvInitShotControl(
              shotCtrl);          /* pointer to shot control structure */
```

DBvInitStoryboardCtrl API を使用すれば、DBvStoryboardCtrl データ構造内の値を初期設定することができます。この API を使用するときは、ストーリーボード制御構造を指定する必要があります。たとえば、次のステートメントでは、DBvStoryboardCtrl 構造の各フィールドを省略時値に初期設定します。

```
DBvStoryboardCtrl    sbCtrl;

rc=DBvInitStoryboardCtrl(
              sbCtrl);          /* pointer to storyboard control structure */
```

## ショットまたはフレームの入手

ビデオ・エクステンダーを使用して、ビデオのショットやフレームを得ることができます。ショットやフレームを入手する前に、ビデオをオープンして、ショットを検出できるようにする必要があります。ビデオ・エクステンダーは、索引を使ってフレームやショットにアクセスします。ショットやフレームを入手するためには、そのビデオの索引を作成する必要があります。

ビデオをオープンし、索引を作成したら、そのビデオ内の次のショットやフレームを入手したり、フレーム番号別に特定のフレームを入手したりできます。ビデオ・エクステンダーは MPEG-1 型式のビデオ・クリップを処理することができます。 RGB 型式を必要とするプログラムで検索したフレームを使用する計画がある場合は、ビデオ・エクステンダー API を使用してフレームをその型式に変換することができます。

### ショット検出のためにビデオをオープンする

データベース表に保管されているビデオをオープンするには、DBvOpenFile API を使用します。そのファイルは、クライアントからアクセス可能でなければなりません。データベース表に保管されているビデオをオープンするには、DBvOpenHandle API を使用します。アプリケーションは、まずそのデータベースに接続する必要があります。ビデオがデータベース表に保管されている場合には、ビデオ・エクステンダーはそのビデオを一時ファイルにコピーします。この一時ファイルは、クライアント環境変数 DB2VIDEOTEMP によって指定されるディレクトリーにあります。ビデオをオープンすると、ショット検出のためにビデオが初期設定されます。ビデオ・エクステンダーは、ポインターをビデオの始め (つまり、フレーム 0) にセットします。

これらの API のどちらかを使用する際には、ビデオ・データ構造 (DBvIOType) へのポインターが入っている区域をポイントしなければなりません。 API 呼び出しの応答として、ビデオ・エクステンダーは、この構造体を割り当て、ここにそのビデオに関する情報を保管します。さらに、この構造体は、現行フレームのピクセル内容をもつフレーム・データ構造 (DBvFrameData) をポイントします。これらの構造体の説明については、188ページの『ショット検出のデータ構造』を参照してください。 DBvOpenFile API の場合は、ビデオ・ファイルの名前も指定する必要があります。 DBvOpenHandle API の場合は、ビデオ・ハンドルをさらに指定する必要があります。

たとえば、次のステートメントでは、ファイルに保管されているビデオをショット検出のためにオープンします。

```
DBvIOType    *videoptr;

rc=DBvOpenFile (
       &videoptr,                        /* pointer to video structure pointer */
       "/employee/video/rsmith.mpg");    /* video file */
```

次のステートメントでは、データベース表に保管されているビデオをショット検出のためにオープンします。

```
EXEC SQL BEGIN DECLARE SECTION;
char Vid_hdl[251];
EXEC SQL END DECLARE SECTION;

DBvIOType    *videoptr;

EXEC SQL SELECT VIDEO INTO :Vid_hdl
FROM EMPLOYEE
WHERE NAME="Anita Jones";

rc=DBvOpenHandle(
       &videoptr,                         /* pointer to video structure pointer */
       vid_hdl);                          /* video handle*/
```

### ビデオの索引付け

ビデオ・エクステンダーは、索引を使ってビデオのフレームやショットにアクセスします。ビデオからショットやフレームを入手する前に、そのビデオの索引を作成する必要があります (MPEG 形式はフレームとショットの索引を備えていません)。索引は、フレーム番号を、MPEG-1 ビデオを表すビット・ストリームにマップします。

DBvCreateIndexFromVideo API または DBvCreateIndex API を使用することにより、ビデオの索引を作成することができます。しかし、DBvOpenFile API または DBvOpenHandle API を使用してショット検査のためにビデオをオープンした場合は、索引を明示的に作成する必要はありません。ビデオ・エクステン

ダーが自動的に索引を作成してくれるからです。 (ビデオをオープンする方法については、194ページの『ショット検出のためにビデオをオープンする』を参照してください。)

索引が (明示的または自動的に) 作成されると、DB2 ビデオ・エクステンダーはその索引をビデオ・ファイルと同じパスに保管しようとします。まず最初に、索引ファイルを fname.ext.idx (fname はビデオ・ファイルの名前で、ext はビデオ・ファイルの拡張子) という名前で保管しようとします。これに失敗すると、ビデオ・エクステンダーは索引ファイルを fname.idx という名前で、ビデオ・ファイルと同じディレクトリーに保管しようとします。これにも失敗すると、索引ファイルをローカル・ディレクトリーに、最初は fname.ext.idx という名前で、次には fname.idx という名前で保管しようとします。

ファイルがオープンしているときには、ビデオ・エクステンダーは次の順序で索引ファイルを探します。

1. 書き込み可能バージョンの索引ファイル。
2. ビデオ・ファイルと同じパスにある索引ファイル、続いて現行ディレクトリー内の索引ファイル。
3. fname.ext.idx という名前の索引、続いて fname.idx という名前をもつもの (fname はビデオ・ファイルの名前で、ext はビデオ・ファイルの拡張子)。

たとえば、myvideo.mpg という名前のビデオ・ファイルの索引を作成した場合、まずビデオ・エクステンダーはこのビデオ・ファイルと同じパスで myvideo.mpg.idx という名前の書き込み可能な索引を探します。

DBvCreateIndexFromVideo API を使用するときは、DBvIOType データ構造を指定します。ビデオ・エクステンダーは、索引ファイルの名前を構造に保管します。この構造については、188ページの『ショット検出のデータ構造』を参照してください。たとえば、次のステートメントでは、ショット検出のためにすでにオープンしたビデオのための索引を作成します。

```
DBvIOType    *video;

rc=DBvCreateIndexFromVideo(
       video);                  /* pointer to video structure */
```

DBvCreateIndex API を使用するときは、ビデオ・ファイルの名前を指定してください。ビデオ・エクステンダーは、その索引を (そのビデオがあるのと同じディレクトリーの) ファイルに保管します。たとえば、次のステートメントでは、ビデオ・ファイルのための索引を作成します (ファイルはショット検出のためにあらかじめオープンされていません)。

```
rc=DBvCreateIndex(
       "/employee/video/rsmith.mpg");    /* video file */
```

さらに、ビデオの索引が存在するかどうかを判断することもできます。索引があるかどうかを検査するには、DBvIsIndex API を使用します。この API によって、ビデオのための索引がなければ 0 が、索引があれば 1 が状況変数にセットされます。たとえば、次のステートメントでは、ビデオ・ファイルの索引があるかどうかを検査します。

```
short  *status

rc=DBvIsIndex(
       "/employee/video/rsmith.mpg",     /* video file */
       &status);                         /* status indicator */
```

**ビデオ索引のバックアップ:** 回復しなければならないときに備えて、ビデオ索引ファイルのバックアップを取ります。このファイルは、ビデオ・エクステンダーがインストールされているのと同じディレクトリーにあります。

### フレームの入手

ビデオ内の現行フレームを入手することができます。さらに、現行フレームを特定のフレーム番号にセットすることもできます。ビデオ内の現行フレームを入手するには DBvGetFrame API を使用します。特定のフレーム番号に現行フレームをセットするには DBvSetFrameNumber API を使用します。

DBvGetFrame API を使用するときは、ビデオ構造を指定します。たとえば、次のステートメントでは、ビデオ内の現行フレームを入手します。

```
DBvIOType    *video;

rc=DBvGetFrame(
       video);                  /* pointer to video structure */
```

DBvSetFrameNumber API を使用する場合、ビデオ構造および現行フレームとしてセットしたいフレームの番号を指定します。たとえば、次のステートメントは、現行フレームをフレーム番号 85 にセットしてから、そのフレームを入手します。

```
DBvIOType    *video;

rc=DBvSetFrameNumber(
       video,                   /* pointer to video structure */
       85);                     /* frame number */

rc=DBvGetFrame(
       video);                  /* pointer to video structure */
```

出力のとき、DBvSetFrameNumber API は DBvIOType 構造内の currentFrame フィールドをリセットします。 DBvGetFrame API は、フレームのピクセル内容を DBvFrameData 構造に入れます。これらの構造の説明については、188ページの『ショット検出のデータ構造』を参照してください。

### ショットの入手

ビデオの次のショットを入手するには、DBvDetectShot API を使用します。 DBvDetectShot API を使用する場合には、次のデータ構造をポイントする必要があります。

- ビデオ (DBvIOType)
- ショット制御 (DBvShotControl)
- ショット・タイプ (DBvShotType)

さらに、探索を開始するフレームをポイントする必要があります。ビデオ・エクステンダーは、ビデオ内のその地点から次のショットの探索を開始します。

API からの結果として、ビデオ・エクステンダーは、shotDetected フラグをセットし、次のショットに関する開始フレームとそのフレーム・データをポイントします。 shotDetected フラグが 1 にセットされている場合は、ショットが検出されたことを示します。この場合、ビデオ・エクステンダーは次のようにします。

- DBvIOType の currentFrame フィールドに次のショットの開始フレームをセットする。
- DBvIOType の fd フィールドに次のショットの開始フレームに関するデータをセットする。
- DBvShotType に、次のショットの開始フレーム番号、終了フレーム番号、代表的なフレームの番号、代表的なフレームのデータ、注釈をセットする。

shotDetected フラグが 0 にセットされている場合は、ショットが検出されなかったことを示します。この場合、ビデオ・エクステンダーは、ビデオの最後に到達したことを示すコードを戻します。

これらの構造の説明については、188ページの『ショット検出のデータ構造』を参照してください。

たとえば、次のステートメントでは、ビデオの次のショットを要求します。

```
DBvIOType    *video;
long start_frame = 1;
char shotDetected = 0;
DBvShotControl    shotCtrl;
DBvShotType    shot;
```

```
shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

rc=DBvDetectShot(
       video,              /* pointer to video structure */
       start_frame,        /* starting frame for search */
       &shotDetected,      /* shot detected flag */
                           /* 1=detected, 0=not detected */
       shotCtrl,           /* pointer to shot control structure */
       &shot);             /* pointer to shot type structure */
```

### 取り出されたフレーム形式の変換

MPEG-1 フレームの内容は YUV 形式です。これは、フレームの明度ピクセル・プレーン、Cr ピクセル・プレーン、Cb ピクセル・プレーンの情報を含む形式です。 このビデオ・フレームを編集する場合には、フレームの形式を YUV から RGB へ変換した方が便利な場合があります。ビデオ・エクステンダーには、取り出された MPEG-1 フレームを YUV 形式から 24 ビットの RGB 形式に変換する DBvFrameDatato24BitRGB API が用意されています。この API を使用する場合には、まずターゲット・バッファーを割り当ててください。

この API を出すときは、ターゲット・バッファーと、変換したいフレーム・データをポイントする必要があります。さらに、フレームの高さと幅を指定します。 (フレームのデータ、高さ、幅は、そのフレームの DBvIOType 構造体から入手することができます。) たとえば、次のステートメントでは、MPEG-1 フレームを 24 ビットの RGB 形式に変換します。

```
char RGB[18000];
DBvIOType      *video;
DBvFrameData   fd;

rc=DBvGetNextFrame(
       video);                  /* pointer to video structure */

fd=video.fd
dx=video.dx
dy=video.dy

rc=DBvFrameDataTo24BitRGB (
       RGB,                      /* pointer to target buffer */
       &fd,                      /* pointer to frame data */
       dx,                       /* frame width */
       dy);                      /* frame height */
```

### ビデオ・ファイルのクローズ

ショット検出のためにオープンしたビデオ・ファイルをクローズするには、DBvClose API を使用します。 API を使用するときは、ファイルのビデオ構造を指すポインターを指定する必要があります。

たとえば、次のステートメントでは、ショット検出のためにオープンされていたビデオ・ファイルをクローズします。

```
DBvIOType    *video;

rc=DBvClose (video);
```

### 取り出したフレームの表示

取り出される MPEG-1 フレームの内容は YUV 形式です。これは、ほとんどの画像表示プログラムが表示できる形式ではありません。取り出されるビデオ・フレームを表示するには、画像表示プログラムが認識できる形式 (たとえば、BMP 形式) に変換する必要があります。たとえば、MPEG-1 フレームを表示するには、次のようにします。

1. DBvFrameDatato24BitRGB API を使って、取り出された MPEG-1 フレームを YUV 形式から 24 ビットの RGB 形式に変換する。
   DBvFrameDatato24BitRGB API の使い方については、199ページの『取り出されたフレーム形式の変換』を参照してください。
2. 変換したフレームに適切なヘッダーを付加する。たとえば、BMP 形式では、画像の高さや幅などの情報をもつヘッダーが必要です。
3. フレーム内容 (とそのヘッダー) をファイルへコピーする。
4. DBiBrowse API を使って、そのファイルを表示する。 DBiBrowse API の使い方については、137ページの『表示 / 再生 API の使用』を参照してください。

## ショットのカタログ登録

ショットに関する情報をショット・カタログに保管することができます。ビデオ・エクステンダーには、次の処理を行う API が用意されています。

- ショット・カタログをデータベース内に作成して管理します。 API を使用して次の処理を行うことができます。
  - ショット・カタログをデータベース内に作成する
  - 単一のショットに関するショット・カタログ情報を保管する
  - すべてのショットに関するショット・カタログ情報をビデオに保管する
  - ショット・カタログに保管されているショット情報を変更する
  - ショット・カタログ内のショット情報をマージする

  - ショット情報をショット・カタログから削除する
  - ショット・カタログをデータベースから削除する
- ショット・カタログ・ファイルを作成し、その中にビデオ内のすべてのショットに関する情報を保管します。カタログ・ファイルを作成して、そこにショット・データを保管する API が用意されています。ショット・カタログ内のデータをアクセスして操作できますが、そのための API は用意されていません。

**カタログ登録されるショットは、ストーリーボードへの入力を提供します:** ショット情報をショット・カタログに保管した後は (カタログがデータベースにあるか、ファイルにあるかに関係なく)、ショット関連のアプリケーションで使用することができます。たとえば、ビデオ内のすべてのショットに関する代表フレームを入手して、ストーリーボードに表示するアプリケーションを作成することができます。

**ユーザーはデータベースにショット・カタログを作成するだけでよいです:** カタログをデータベースに常駐させたい場合にのみ、ショット・カタログを作成する必要があります。ビデオ内のショットに関するデータを保管し、ファイルへの出力が必要であることを示すと、ビデオ・エクステンダーは自動的にショット・カタログ・ファイルを作成します。

**カタログの作成前 (データベースのみ)**

データベースにカタログを作成して使用する前に、次のことを行う必要があります。

- SQLConnect 呼び出しを行います。 DB2 データベース内のショット・カタログは、表の集合で構成されます。データベースにショット・カタログを作成したり、それに対して操作を行ったりする前に、SQLConnect 呼び出しを使って、そのデータベースに接続しなければなりません。 (SQLConnect は DB2 コール・レベル・インターフェースの呼び出しです。) この呼び出しは、ショット・カタログを管理する API に指定する必要がある接続ハンドルを戻します。
- データベースをイメージ・データで使用できるようにします。ショット・カタログをデータベースに作成する前に、そのデータベースが DB2Image データ・タイプで使用できるようになっていなければなりません。ビデオ・エクステンダーは、他の情報とともに、カタログされた各ショットの代表的なフレーム・データをショット・カタログに保管します。代表的なフレーム・データのデータ・タイプは DB2Image です。

### ショット・カタログの作成 (データベースのみ)

ショット・カタログをデータベース内に作成するには、DBvCreateShotCatalog API を使用します。 (ショットに関するデータを保管し、ファイルに出力が必要であることを示すと、ビデオ・エクステンダーは自動的にショット・カタログ・ファイルを作成します。) このカタログは、ショットに関連する情報が入った表から成り立っています。 SQL を使用すれば、表の視点を照会することができます。表13 は、視点内の列を示しています。

*表 13. ショット・カタログの視点の列*

| 列名 | データ・タイプ | 説明 |
|---|---|---|
| SHOTHANDLE | CHAR(36) | ショット・ハンドル |
| VIDEOHANDLE | VARCHAR(254) | ビデオ・ハンドル。この列には、ビデオが DBvOpenHandle API でオープンされている場合にだけ値が入ります。 |
| VIDEOTABLE | VARCHAR(254) | ビデオが入る表。この列には、ビデオが DBvOpenHandle API でオープンされている場合にだけ値が入ります。 |
| VIDEOCOLUMN | VARCHAR(254) | ビデオが入る表の列。この列には、ビデオが DBvOpenHandle API でオープンされている場合にだけ値が入ります。 |
| VIDEOFILE | VARCHAR(254) | ビデオ・ファイルの名前。この列には、ビデオが DBvOpenFile API でオープンされている場合にだけ値が入ります。 |
| STARTFRAME | INTEGER | 開始フレームの番号。 |
| ENDFRAME | INTEGER | 終了フレームの番号。 |
| REPFRAME | INTEGER | 代表フレームの番号。 |
| REPFRAMEDATA | DB2IMAGE | 代表フレームのデータ。 |
| COMMENTS | LONG VARCHAR | 注釈。 |

データベースにショット・カタログをいくつ作成するか、また、各ショット・カタログにどのショットの情報を保管するかについては、いくつかの選択肢が

あります。カタログを 1 つ作って、そこに多くのビデオのショット情報を保管することもできますし、別々のカタログに各ビデオのショット情報を保管することもできます。あるいは、同じビデオの複数のショットの情報を複数のカタログに保管することもできます。

この API を使用するときは、このカタログの名前を指定する必要があります。名前が 16 文字より長いと、超える分は切り捨てられます。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、ホット・ショットという名前のショット・カタログを作成します。

```
SQLHDBC  hdbc;

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvCreateShotCatalog(
                "hotshots",           /* shot catalog name */
                hdbc);                /* database connection handle */
```

ショット・カタログの視点の名前は、MMDBSYS.SVcatname となります。catname はそのショット・カタログの名前です。たとえば、カタログ名が hotshots の場合、その視点の名前は MMDBSYS.SVHOTSHOTS です。

**単一のショットに関する情報の保管 (データベースのみ)**

単一のショットに関する情報をショット・カタログに保管するには、DBvInsertShot API を使用します。ショット・カタログがデータベースにある場合にのみ、ビデオ内の単一ショットに関する情報を保管することができます。このカタログに保管される情報には、次のものがあります。

- ショット・ハンドル
- ビデオ表の名前 (表に保管されているビデオ・クリップの)
- ビデオ列の名前 (表に保管されているビデオ・クリップの)
- ビデオ・ハンドル (表に保管されているビデオ・クリップの)
- ビデオ・ファイルの名前 (ファイルに保管されているビデオ・クリップの)
- 開始フレームの番号
- 終了フレームの番号
- 代表フレームの番号
- 代表フレームのデータ

ただし、ショットの注釈は保管されません。保管されているショット情報に注釈を追加する方法については、208ページの『ショットの注釈の指定 (データベースのみ)』を参照してください。

DBvInsertShot API を使用するときは、ショット・カタログの名前と、ショットを指すポインターを指定する必要があります。ショットを指すポインターをセットする唯一の方法は、198ページの『ショットの入手』で説明したように次のショットを入手します。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、フレーム 1 の後の次のショットを入手してから、そのショットに関する情報を hotshots というショット・カタログに保管します。

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType    *video;
long start_frame = 1;
char shotDetected = 0;
DBvShotControl    shotCtrl;
DBvShotType    shot;

shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvDetectShot(
       video,
       start_frame,
       &shotDetected,
       &shotCtrl,
       &shot)

rc=DBvInsertShot (
       "hotshots",               /*shot catalog name*/
       shot,                     /*pointer to shot*/
       hdbc);                    /*database connection handle*/
```

### ビデオ内のすべてのショットに関する情報の保管

ビデオのすべてのショットに関する情報をショット・カタログに保管するには、DBvBuildStoryboardTable API または DBvBuildStoryboardFile API を使用します。 DBvBuildStoryboardTable API はデータベース内に常駐するショット・カタログに情報を保管します。 DBvBuildStoryboardFile API はショット・カタログ・ファイルを作成して、ショット情報をファイルに保管します。

いずれの API でも、ソース・ビデオはデータベース表またはファイルに置くことができます。

いずれかの API を使用するときは、このカタログの名前を指定する必要があります。

- ショット・カタログ名を指定します。
- ビデオ構造をポイントします。
- DBvShotControl データ構造をポイントします。
- DBvStoryboardCtrl データ構造をポイントします。このデータ構造の値は、代表フレームとしてどのビデオ・フレームをいくつショット・カタログに保管するかを制御します。これらの値のセットについては、206ページの『ストーリーボードの作成』を参照してください。

DBvBuildStoryboardTable API の場合にのみ、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルも指定する必要があります。

たとえば、次のステートメントでは、ビデオのすべてのショットに関する情報をショット・カタログに保管します。ショット・カタログはデータベースにあります。

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType    *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvBuildStoryboardTable (
       "hotshots",               /*shot catalog name*/
       video,                    /*pointer to video structure*/
       shotCtrl,                 /*pointer to shot control structure*/
       sbctrl,                   /*pointer to storyboard control structure*/
       hdbc);                    /*database connection handle*/
```

次のステートメントでは、ショット・カタログ・ファイルを作成して、ビデオのすべてのショットに関する情報をそのファイルに保管します。

```
DBvIOType    *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

rc=DBvBuildStoryboardFile (
       "hotshots",               /*shot catalog file name*/
       video,                    /*pointer to video structure*/
       shotCtrl,                 /*pointer to shot control structure*/
       sbctrl);                  /*pointer to storyboard control structure*/
```

### ストーリーボードの作成

DBvBuildStoryboardTable API と DBvBuildStoryboardFile API は、その名前が示すように、ストーリーボードで使用される情報を保管する場合に特に便利です。**ストーリーボード**は、ビデオを視覚的に要約したものです。ストーリーボードは、ショット・カタログ内にビデオのために保管された代表フレームを表示することによって、作成することができます。

DBvBuildStoryboardTable API および DBvBuildStoryboardFile API は 1 つのショットに関して 1 つまたはそれ以上の代表フレームを保管します。DBvStoryboardCtrl 構造に指定した値によって、1 つのショットに関していくつの代表フレームを保管し、どのフレームを使用するかを制御することができます。 DBvStoryboardCtrl 構造の定義については、188ページの『ショット検出のデータ構造』を参照してください。 図29 は、DBvStoryboardCtrl フィールドの値がどのように使用されるかを示しています。

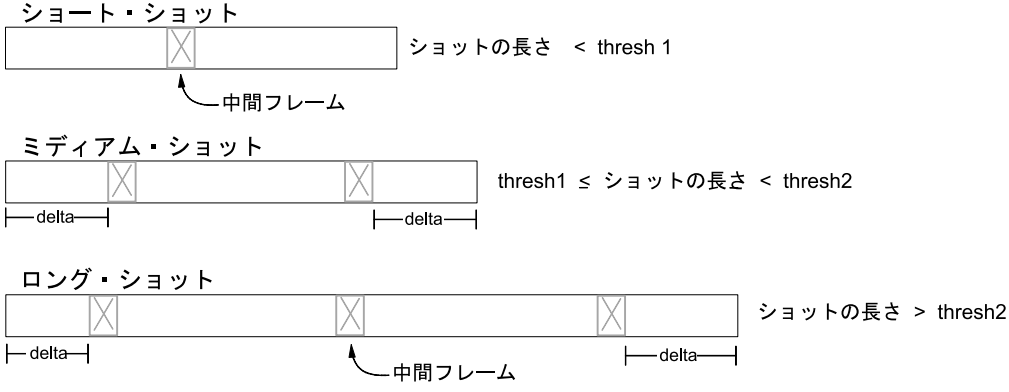



*図 29. DBvStoryboardCtrl 構造内の値が使用される方法*

図29 は次のことを示しています。

- ショート・ショットの場合、代表フレームは 1 つだけ保管されます。ショート・ショット内のフレーム数は、DBvStoryboardCtrl データ構造内の thresh1 値より小です。代表フレームはショットの中間フレームです。
- ミディアム・ショットの場合、代表フレームは 2 つ保管されます。ミディアム・ショット内のフレーム数は、DBvStoryboardCtrl データ構造内の thresh1 値より大で、thresh2 値より小です。 DBvStoryboardCtrl データ構造内の delta 値が、ビデオの先頭から最初の代表フレームまでのフレーム数を指定します。 delta 値は、2 番目の代表フレームからビデオの最後までのフレーム数も指定します。
- ロング・ショットの場合、代表フレームは 3 つ保管されます。ロング・ショット内のフレーム数は、DBvStoryboardCtrl データ構造内の thresh2 値より大です。 DBvStoryBoardCtrl データ構造内の delta 値が、ビデオの先頭から最初の代表フレームまでのフレーム数を指定します。 2 番目の代表フレームはビデオの中間フレームです。ビデオの最後から 3 番目の代表フレームまでの距離は delta 値によって指定します。

thresh1 または thresh2 の値を -1 にセットすれば、任意のショットをショート・ショットとして処理することができます。この場合、唯一の代表フレームである中間フレームがショット・カタログのショットとして保管されます。

DBvStoryboardCtrl データ構造の値のほかに、DBvShotControl データ構造内には、後でストーリーボードに表示するためにどの代表フレームを保管するかに影響を与えるフィールドがいくつかあります。たとえば、DBvShotControl データ構造内の CorrHistThresh、normalcorrValue、および minShotSize フィールドは、ショット検出のしきい値を指定するので、ビデオのストーリーボードにどのフレームが表示されるかに影響を与えます。 DBvBuildStoryBoardTable API と DBvBuildStoryBoardFile API を使用してストーリーボードで使用するショット情報を保管する場合、DBvStoryBoardCtrl と DBvShotControl データ構造のための初期設定値を使用して、まず試験的な実行を試みることができます。そうしてから、これらのデータ構造の各フィールドの値を変更することによって、結果を調整することができます。

### ストーリーボードの表示

ストーリーボードを表示するプログラムを作成することができます。そのためには、ビデオに関するショット・カタログに保管されている代表フレームにアクセスします。ビデオに関するショットを保管するために DBvBuildStoryboardFile API を使用した場合、ショット・カタログ・ファイルは代表フレームに関する GIF ファイルをポイントします。ブラウザーを使用してこれらの GIF ファイルを表示するか、または必要に応じてプログラムを表示します。

ビデオに関するショットを保管するために DBvBuildStorybBoardTable API を使用した場合、 (データベースに保管されている) ショット・カタログには代表フレームのデータが入っています。ショット・カタログ視点内の代表フレーム・データにアクセスすることができます (視点については、202ページの表13を参照してください)。代表フレーム・データは YUV 形式です。これは、ほとんどの画像表示プログラムでは表示できない形式です。代表フレームを表示するためには、200ページの『取り出したフレームの表示』で説明した DBvFrameDatato24BitRGB API を使用してフレーム・データを変換します。そうしておくと、ブラウザーを使用してこれらの代表フレームを表示したり、または必要に応じてプログラムを表示することができます。

### ストーリーボードのサンプル・プログラム

SAMPLES サブディレクトリーには、ビデオに関するストーリーボードの作成と表示を示す 2 つのサンプル・プログラムが入っています。一方のサンプル・プログラムはファイル `makesf.exe` に入っており、DBvBuildStoryBoardFile API を使用してショット・カタログ・ファイルを作成し、ショット・データをファイルに保管します。他方のサンプル・プログラム `makehtml.exe` は、ショット・カタログ・ファイルにアクセスし、Web ブラウザーで表示するために HTML ページを作成します。

### ショットの注釈の指定 (データベースのみ)

ショット・カタログ内のショットに他の情報とともに保管する注釈を指定することができます。注釈を指定するには、DBvSetShotComment API を使用します。

この API を使用するときは、その注釈を保管するショット・カタログの名前、その注釈を追加する対象のショットのハンドル、そしてその注釈を指定する必要があります。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、ショット (フレーム番号 85 から始まる) の注釈を hotshots というショット・カタログに追加します。

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

rc=DBvSetShotComment (
```

```
        "hotshots",                     /*shot catalog name*/
        shothandle,                     /*shot handle*/
        "shot of beach at sunset",      /*comment*/
        hdbc);                          /*database connection handle*/
```

### ショット・カタログに保管されているショット情報を変更する

ショット・カタログに保管されているショットの情報を変更することができます。その場合、DBvUpdateShot API を使用します。置き換える情報は DBvShotType 構造体に入れてください。さらに、変更しないフィールドがある場合は、それらも指定する必要があります。 DBvUpdateShot API を使用する場合には、カタログの名前と、DBvShotType 構造体へのポインターを指定します。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。

ショットの情報を変更する場合には、その情報とともに保管されている注釈 (ある場合) も変更することができます。注釈を変更する場合には、それを DBvShotType 構造体に指定します。古い注釈を保持する場合には、DBvShotType 構造体にヌル値を指定してください。

たとえば、次のステートメントでは、hotshots というカタログに保管されているショット情報を変更します。このショットはフレーム番号 85 から始まります。

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];
DBvShotType    shot;
DBvFrameData  fd110;

/* get shot handle */

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

/* change shot attribute */

shot.startFrame=110;
shot.endFrame=200;
shot.repframe=110;
shot.fd=fd110;
shot.comment=NULL;

/* update shot information */

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvUpdateShot (
```

```
        "hotshots",                     /*shot catalog name*/
        shot,                           /*shot information*/
        hdbc);                          /*database connection handle*/
```

### ショット・カタログにおけるショット情報のマージ (データベースのみ)

ショット・カタログに保管されている 2 つのショットの情報を組み合わせることができます。ショット情報をマージする場合には、最初のショットと 2 つ目のショットを指定することによって、マージの順序を指定する必要があります。最初のショットの開始フレーム番号が、マージしたショットの開始フレーム番号として保管されます。最初のショットと 2 つ目のショットの間の最大フレームの番号が、マージしたショットの終了フレーム番号として保管されます。このマージによって、最初のショットに対して保管されている情報が、マージされたショットに関する情報で置き換わります。 2 つ目のショットに対して保管されている情報は、ショット・カタログから削除されます。

ショット・カタログにおいて 2 つのショットに関する情報をマージするには、DBvMergeShots API を使用します。この API を使用する場合には、ショット・カタログの名前と、それに続けて、マージする最初のショットと 2 つ目のショットのハンドルを指定します。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、hotshots というカタログに保管されている 2 つのショットのショット情報をマージします。最初のショットはフレーム番号 85 から始まり、2 つ目のショットはフレーム番号 210 から始まります。

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle1[37];
char shothandle2[37];

EXEC SQL SELECT SHOTHANDLE INTO :shothandle1
  FROM MMDBSYS.SVHOTSHOTS1
  WHERE STARTFRAME=85;

EXEC SQL SELECT SHOTHANDLE INTO :shothandle2
  FROM MMDBSYS.SVHOTSHOTS2
  WHERE STARTFRAME=210;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvMergeShots (
        "hotshots",                     /*shot catalog name*/
        shothandle1,                    /*shot handle for first shot*/
        shothandle2,                    /*shot handle for second shot*/
        hdbc);                          /*database connection handle*/
```

### ショット・カタログからショット情報の削除 (データベースのみ)

ショットに関する情報をショット・カタログから削除するには、DBvDeleteShot API を使用します。この API を使用するときは、ショット・カタログの名前と、それに続けてショット・ハンドルを指定します。さらに、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、hotshots というショット・カタログにあるショット (フレーム番号 85 から始まる) の情報を削除します。

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

EXEC SQL SELECT shothandle INTO :shothandle
  FROM mmdbsys.svhotshots
  WHERE startframe=85;

rc=DBvDeleteShot (
        "hotshots",                    /*shot catalog name*/
        shothandle,                    /*shot handle*/
        hdbc);                         /*database connection handle*/
```

### ショット・カタログの削除 (データベースのみ)

ショット・カタログを削除するには、DBvDeleteShotCatalog API を使用します。この API を使用する場合には、削除するショット・カタログの名前と、データベースに対する SQLConnect 呼び出しによって戻されるデータベース接続ハンドルを指定する必要があります。たとえば、次のステートメントでは、ショット検出のためにオープンされていたビデオ・ファイルをクローズします。

```
SQLHDBC  hdbc;
SQLHENV  henv;

rc=DBvDeleteShotCatalog (
        "hotshots",             /*shot catalog name*/
        hdbc);                  /*database connection handle*/
```

# 第4部 参照情報

# 第15章 ユーザー定義タイプとユーザー定義関数

この章では、DB2 エクステンダーによって作成される UDT と UDF の参照情報を示します。

## スキーマ

エクステンダーは、UDT および UDF を含むすべてのオブジェクト関連オブジェクトに対して MMDBSYS スキーマを使用します。

## ユーザー定義タイプ

表14 は、DB2 エクステンダーによって作成されるユーザー定義タイプをリストし、説明しています。さらに、UDT の DB2 ソース・データ・タイプも示します。

*表 14. DB2 エクステンダーによって作成されるユーザー定義タイプ*

| UDT | ソース・データ・タイプ | 説明 |
|---|---|---|
| DB2IMAGE | VARCHAR(250) | 画像ハンドル。画像オブジェクトにアクセスするために必要となる情報が入る可変長ストリング。画像ハンドルは、イメージ・エクステンダーが使用できるようにされたユーザー表の列に保管されます。 |
| DB2AUDIO | VARCHAR(250) | 音声ハンドル。音声オブジェクトにアクセスするために必要な情報が入る可変長ストリング。音声ハンドルは、オーディオ・エクステンダーが使用できるようにされたユーザー表の列に保管されます。 |
| DB2VIDEO | VARCHAR(250) | ビデオ・ハンドル。ビデオ・オブジェクトをアクセスするために必要な情報が入る可変長ストリング。ビデオ・ハンドルは、ビデオ・エクステンダーが使用できるようにされたユーザー表の列に保管されます。 |

## ユーザー定義関数

この節では、DB2 エクステンダーの参照情報を示します。ユーザー定義関数 (UDF) は、アルファベット順にリストされています。

UDF ごとに、次の情報を示します。

- UDF を提供するエクステンダー
- 要旨
- その UDF 用のインクルード (ヘッダー) ファイル
- その UDF の SQL 構文
- その UDF パラメーターの説明 (そのデータ・タイプを含む)
- その UDF によって戻される値 (そのデータ・タイプを含む)
- 使用例

表15 は、UDF をリストし、各 UDF を使用するエクステンダーを示しています。この表は、各 UDF の情報があるページも示しています。この表にある UDF は、組み込み SQL ステートメントまたは DB2 CLI 呼び出しにコーディングすることができます。

*表 15. DB2 エクステンダー UDF*

| UDF | 説明 | イメージ | オーディオ | ビデオ | 参照ページ |
|---|---|---|---|---|---|
| AlignValue | WAVE オーディオ、またはビデオの音声トラックにおけるサンプルごとのバイト数を戻します。 | | o | o | 220 |
| AspectRatio | MPEG1 および MPEG2 ビデオの最初のトラックの縦横比を戻します。 | | | o | 222 |
| BitsPerSample | 音声や、ビデオの音声トラックにおいて、 WAVE オーディオや AIFF オーディオの各サンプルを表すのに使用されるデータのビット数を戻します。 | | o | o | 223 |
| BytesPerSec | WAVE オーディオのデータ転送速度を秒当たりの平均バイト数で戻します。 | | o | | 224 |
| Comment | 画像や、音声、ビデオとともに保管されている注釈を戻したり、更新したりします。 | o | o | o | 225 |

*表 15. DB2 エクステンダー UDF (続き)*

| UDF | 説明 | イメージ | オーディオ | ビデオ | 参照ページ |
|---|---|---|---|---|---|
| CompressType | ビデオの圧縮形式 (MPEG-1 など) を戻します。 | | | o | 227 |
| Content | 画像や、音声、ビデオの内容をデータベースから取り出したり、更新したりします。 | o | o | o | 228 |
| DB2Audio | 音声の内容をデータベース表に保管します。 | | o | | 234 |
| DB2Image | 画像の内容をデータベース表に保管します。 | o | | | 238 |
| DB2Video | ビデオの内容をデータベース表に保管します。 | | | o | 243 |
| Duration | WAVE オーディオや AIFF オーディオ、またはビデオの所要時間 (つまり、秒単位による再生時間) を戻します。 | | o | o | 247 |
| Filename | 画像や、音声、ビデオの内容が入っているサーバー・ファイルの名前を戻します。 | o | o | o | 248 |
| FindInstrument | MIDI オーディオにおいて、指定する楽器が最初に現れるトラック番号を戻します。 | | o | | 249 |
| FindTrackName | MIDI オーディオにおける指定の名前のトラックの番号を戻します。 | | o | | 250 |
| Format | 画像や、音声、ビデオの形式を戻します。 | o | o | o | 251 |
| FrameRate | ビデオのスループットを秒当たりのフレーム数で戻します。 | | | o | 252 |
| GetInstruments | MIDI オーディオにおけるすべての楽器の楽器名を戻します。 | | o | | 253 |
| GetTrackNames | MIDI オーディオにおけるすべてのトラックの名前を戻します。 | | o | | 254 |
| Height | 画像やビデオ・フレームの高さをピクセル数で戻します。 | o | | o | 255 |

*表 15. DB2 エクステンダー UDF (続き)*

| UDF | 説明 | イメージ | オーディオ | ビデオ | 参照ページ |
|---|---|---|---|---|---|
| Importer | 画像や、音声、ビデオをデータベース表に保管した人のユーザー ID を戻します。 | o | o | o | 256 |
| ImportTime | 画像や、音声、ビデオがいつデータベース表に保管されたのかを示す時刻スタンプを戻します。 | o | o | o | 257 |
| MaxBytesPerSec | ビデオの最大スループットを秒当たりのバイト数で戻します。 | | | o | 258 |
| NumAudioTracks | ビデオや MIDI オーディオにおける音声トラックの数を戻します。 | | o | o | 259 |
| NumChannels | WAVE オーディオや AIFF オーディオ、またはビデオに録音されたオーディオ・チャネルの数を戻します。 | | o | o | 260 |
| NumColors | 画像における色の数を戻します。 | o | | | 261 |
| NumFrames | 画像におけるフレームの数を戻します。 | | | o | 262 |
| NumVideoTracks | ビデオでのビデオ・トラックの数を戻します。 | | | o | 263 |
| QbScoreFromName | 画像の得点を戻します (名前付き照会オブジェクトを使用します)。(QbScore の代わりです。) | o | | | 264 |
| QbScoreFromStr | 画像の得点を戻します (照会ストリングを使用します)。 | o | | | 266 |
| QbScoreTBFromName | 画像列からの得点の表を戻します (名前付き照会オブジェクトを使用します)。 | o | | | 268 |
| QbScoreTBFromStr | 画像列からの得点の表を戻します (照会ストリングを使用します)。 | o | | | 270 |
| Replace | データベースに保管されている画像や、音声、ビデオの内容を更新します。また、その注釈を更新します。 | o | o | o | 272 |

*表 15. DB2 エクステンダー UDF (続き)*

| UDF | 説明 | イメージ | オーディオ | ビデオ | 参照ページ |
|---|---|---|---|---|---|
| SamplingRate | WAVE オーディオや AIFF オーディオ、またはビデオの音声トラックのサンプリング率を秒当たりのサンプル数で戻します。 | | o | o | 276 |
| Size | 画像や、音声、ビデオのサイズをバイト数で戻します。 | o | o | o | 277 |
| Thumbnail | データベースに保管されているサムネールの画像やビデオ・フレームを戻したり、更新したりします。 | o | | o | 278 |
| TicksPerQNote | 録音されている MIDI オーディオのクロック速度を四分音符当たりのティック数で戻します。 | | o | | 280 |
| TicksPerSec | 録音されている MIDI オーディオのクロック速度を秒当たりのティック数で戻します。 | | o | | 281 |
| Updater | データベース表の画像や、音声、ビデオを最後に更新した人のユーザー ID を戻します。 | o | o | o | 282 |
| UpdateTime | データベース表の画像や、音声、ビデオが最後に更新されたのがいつかを示す時刻スタンプを戻します。 | o | o | o | 283 |
| Width | 画像やビデオ・フレームの幅をピクセル数で戻します。 | o | | o | 284 |

## AlignValue

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

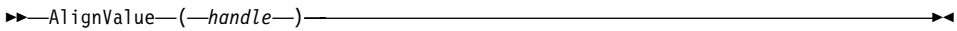
WAVE オーディオ、またはビデオの音声トラックにおけるサンプルごとのバイト数を戻します。 WAVE オーディオはそのデータを保管するのに、サンプルごとに 1 バイト (8 ビット・モノ、『バイト境界合わせ』 と呼ばれる)、2 バイト (8 ビット・ステレオまたは 16 ビット・モノ、『ワード境界合わせ』 と呼ばれる)、または 4 バイト (16 ビット・ステレオ、『ダブルワード境界合わせ』 と呼ばれる) を使用することができます。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

WAVE オーディオ、またはビデオにおける音声トラックの、サンプルごとのバイト数の値 (SMALLINT)。値は次のとおりです。

**1** バイト境界合わせ

**2** ワード境界合わせ

**4** ダブルワード境界合わせ

**ヌル値** 他の形式の音声

### 例

ワード境界合わせがされている従業員表の、音声列に保管されているすべての音声のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE ALIGNVALUE(SOUND) = 2;
```

ビデオ内の音声トラックのサンプル値ごとのバイト数を調べます。このビデオは Anita Jones の従業員表のビデオ列に保管されています。

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvAlign_val;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ALIGNVALUE(VIDEO)
      INTO :hvAlign_val
      FROM EMPLOYEE
      WHERE NAME='Anita Jones';
```

## AspectRatio

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

MPEG ビデオの最初のトラックの縦横比を戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
MPEG ビデオの最初のトラックの縦横比、または他の形式 (SMALLINT) のビデオの場合はヌル値。

### 例
従業員表のビデオ列に保管されている Robert Smith の縦横比を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvAsp_ratio;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ASPECTRATIO(VIDEO)
      INTO :hvAsp_ratio
      FROM EMPLOYEE
      WHERE NAME='Robert Smith';
```

## BitsPerSample

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

音声や、ビデオの音声トラックにおいて、WAVE オーディオや AIFF オーディオの各サンプルを表すのに使用されるデータのビット数を戻します。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ビデオや、WAVE オーディオまたは AIFF オーディオ (SMALLINT) の各サンプルを表すために必要なデータのビット数。他の形式の音声の場合はヌル値を戻します。

### 例

従業員表のサウンド列に保管されているすべての WAVE オーディオのうちサンプルのビット数が 8 に等しいもののファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='WAVE'
      AND BITSPERSAMPLE(SOUND) = 8;
```

## BytesPerSec

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

WAVE オーディオのデータ転送速度を秒当たりの平均バイト数で戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
データ転送速度 (INTEGER)。他の形式の音声の場合はヌル値を戻します。

### 例
従業員表のサウンド列に保管されているすべての音声のうち転送速度 (平均バイト / 秒) が 44100 より小さいもののファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE BYTESPERSEC(SOUND) < 44100;
```

## Comment

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオとともに保管されている注釈を戻したり、更新したりします。

### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

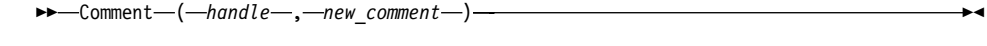
### 構文

**注釈の検索**




### 構文

**注釈の更新**




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、 DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

**new_comment (LONG VARCHAR)**
更新用の新しい注釈。ヌル値または空ストリングを指定すると、既存の注釈は削除されます。

### 戻り値 (データ・タイプ)

更新の場合、画像、音声、またはビデオのハンドル (DB2IMAGE、DB2AUDIO、または DB2VIDEO)。取り出しの場合、注釈 (LONG VARCHAR)。

### 例

従業員表のピクチャー列に保管されているすべての画像のうち、対応する注釈に 『confidential』 という語のあるすべての画像のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[255;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
        LIKE '%confidential%';
```

従業員のビデオ列の Anita Jones のビデオ・クリップに関連している注釈を更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
      short len;
      char data[4000];
      }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME =  'Anita Jones';

/* Update the comment */

hvRemarks.data[hvRemarks.len]='¥;0';
strcat (hvRemarks.data, "Updated video");
hvRemarks.len=strlen(hvRemarks.data);

EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME =  'Anita Jones';
```

## CompressType

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオの圧縮形式 (MPEG-1 など) を戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
ビデオの圧縮形式 (VARCHAR(8))。

### 例
従業員表のビデオ列に保管されている、その圧縮形式が MPEG-1 であるビデオの名前をすべて入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE COMPRESSTYPE(VIDEO) = 'MPEG1';
```

## Content

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオの内容をデータベースから取り出したり、更新したりします。その内容は、クライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルへ取り出せます。

### インクルード・ファイル

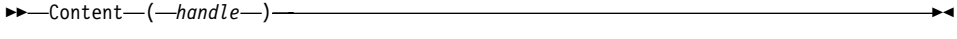
**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

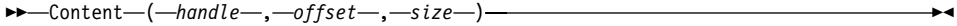
### 構文

内容をバッファーまたはクライアント・ファイルへ取り出す




### 構文

内容のセグメントをバッファーまたはクライアント・ファイルへ取り出す




### 構文

内容をサーバー・ファイルへ取り出す



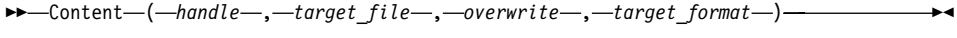

### 構文

内容をバッファーまたはクライアント・ファイルへ取り出し、形式変換する − 画像のみ



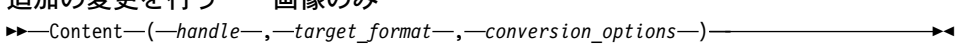

### 構文

内容をサーバー・ファイルへ取り出し、形式変換する − 画像のみ



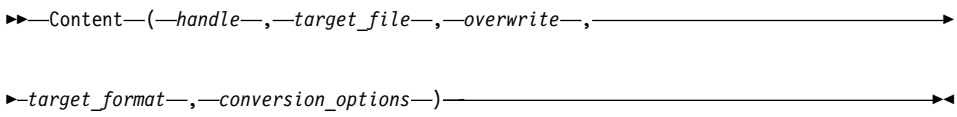
**構文**

内容をバッファーまたはクライアント・ファイルへ取り出し、形式変換および追加の変更を行う － 画像のみ




**構文**

内容をサーバー・ファイルへ取り出し、形式変換および追加の変更を行う -- 画像のみ



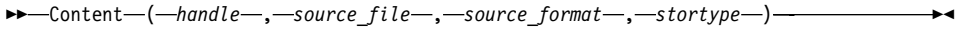

**構文**

内容をバッファーまたはクライアント・ファイルから更新する



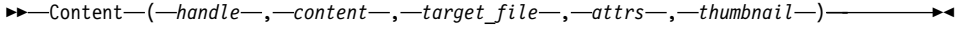

**構文**

内容をサーバー・ファイルから更新する



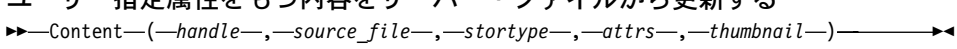

**構文**

ユーザー指定属性をもつ内容をバッファーまたはクライアント・ファイルから更新する



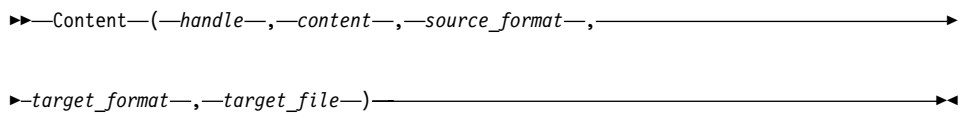

**構文**

ユーザー指定属性をもつ内容をサーバー・ファイルから更新する



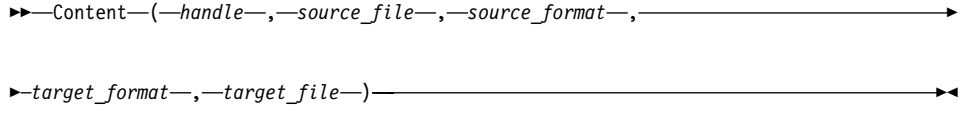

**構文**

内容をバッファーまたはクライアント・ファイルから更新し、形式変換する － 画像のみ



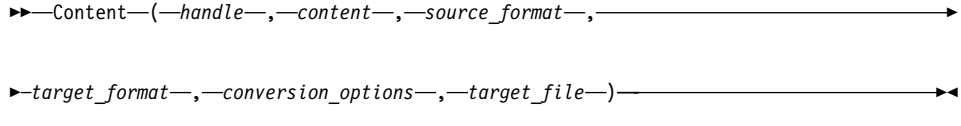

**構文**

**内容をサーバー・ファイルから更新し、形式変換する － 画像のみ**



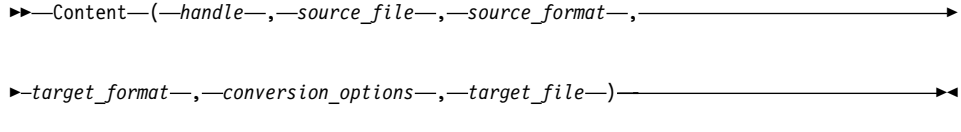

**構文**

**内容をバッファーまたはクライアント・ファイルから更新し、形式変換および追加の変更を行う － 画像のみ**




**構文**

**内容をサーバー・ファイルから更新し、形式変換および変更の追加を行う － 画像のみ**




**パラメーター (データ・タイプ)**

**handle (DB2IMAGE, DB2AUDIO, or DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

**offset (INTEGER)**
取り出す画像や、音声、ビデオの開始オフセット (起点は 1)。

**size (INTEGER)**
取り出す画像や、音声、ビデオのバイト数。

**source_file (LONG VARCHAR)**
画像や、音声、ビデオの更新内容をもつファイルの名前。

**target_file (LONG VARCHAR)**

取り出しの場合、取り出した画像や、音声、ビデオを入れるファイルの名前。更新の場合、更新する画像や、音声、ビデオが入ったファイルの名前。

**stortype (INTEGER)**

更新した画像や、音声、ビデオをどこに保管するかを示す値。定数 MMDB_STORAGE_TYPE_INTERNAL (値=1) の場合、更新したオブジェクトは BLOB としてデータベースに保管されます。定数 MMDB_STORAGE_TYPE_EXTERNAL (値=0) の場合、更新したオブジェクトはサーバー・ファイルに保管されます。

**overwrite (INTEGER)**

ターゲット・ファイルがすでにある場合、それに重ね書きするかどうかを示す値。値は 0 か 1 です。値が 0 の場合、ターゲット・ファイルは重ね書きされません (実際には、取り出しが行われません)。値が 1 の場合、ターゲット・ファイルが存在すると、それが重ね書きされます。

**target_format (VARCHAR(8))**

取り出し後または更新後の画像の形式。ソース・イメージの形式が、適宜、変換されます。画像をサーバー・ファイルに取り出す場合、target_file が source_file と同じなら、ターゲットの形式はソースの形式と同じでなければなりません。 MPG1 形式の場合、MPG1、mpg1、MPEG1、または mpeg1 を指定できます。 MPG2 形式の場合、MPG2、mpg2、MPEG2、または mpeg2 を指定できます。

**conversion_options (VARCHAR(100))**

画像の取り出し時または更新時に適用する回転や圧縮などの変更を指定します。サポートされる変換オプションについては、97ページの表6 を参照してください。

**content (BLOB(2G) AS LOCATOR)**

画像や、音声、ビデオの更新内容をもつホスト変数。ホスト変数のタイプは BLOB、BLOB_FILE、または BLOB_LOCATOR です。 DB2 は、内容のデータ・タイプを BLOB_LOCATOR にプロモートし、LOB ロケーターを Content UDF に渡します。

**source_format (VARCHAR(8))**

画像や、音声、ビデオの更新ソースの形式。ヌル値または空ストリングを指定できます。あるいは、画像の場合、文字ストリング ASIS を指定できます。これら 3 つの場合、その形式はエクステンダーが自動的

に決定しようとします。 MPG1 形式の場合、 MPG1、mpg1、MPEG1、または mpeg1 を指定できます。 MPG2 形式の場合、MPG2、mpg2、MPEG2、または mpeg2 を指定できます。

**attrs (LONG VARCHAR FOR BIT)**
画像や、音声、ビデオの属性。

**thumbnail (LONG VARCHAR FOR BIT DATA)**
その画像やビデオ・フレームのサムネール (画像とビデオのみ)。

**戻り値 (データ・タイプ)**
バッファーに取り出す場合、取り出される画像や、音声、ビデオの内容 (BLOB(2G) AS LOCATOR)。ファイルに取り出す場合、VARCHAR(254)。

更新の場合、更新される画像や、音声、ビデオのハンドル (DB2IMAGE、DB2AUDIO、または DB2VIDEO)。

**例**
従業員表のピクチャー列に保管されている Anita Jones の画像をサーバー・ファイルに取り出します。

```
struct{
      short len;
      char data[250];
      }hvImg_fname;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (PICTURE,
       '/employee/images/ajones.bmp',1)
      INTO :hvImg_fname
      FROM EMPLOYEE
      WHERE NAME='Anita Jones';
```

従業員表のサウンド列に保管されている Robert Smith の 1 MB の音声クリップをクライアント・バッファーに取り出します。

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (SOUND, 1, 1000000)
      INTO :audio_loc
      FROM EMPLOYEE
      WHERE NAME='Robert Smith';
```

従業員表のピクチャー列に保管されている Anita Jones の画像を更新します。その際、その画像の形式を BMP から GIF へ変換し、画像のサイズを元の 50% に縮小します。

```
EXEC SQL UPDATE EMPLOYEE
  SET picture = CONTENT(PICTURE,
       '/employee/newimg/ajones.bmp',
       'BMP',
       'GIF',
        '-s 0.5',
       '');
  WHERE NAME='Anita Jones';
```

## DB2Audio

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

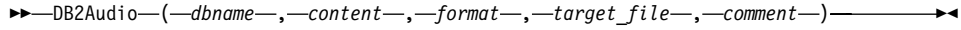
音声の内容をデータベース表に保管します。オーディオ・ソースはクライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルのどれにあってもかまいません。音声は、データベース表に BLOB として保管することも、サーバー・ファイル (データベースによって参照される) に保管することもできます。音声ソースは、サポートされる形式であっても、サポートされない形式であってもかまいません。サポートされる形式の場合には、DB2 オーディオ・エクステンダーがその属性を識別して保管します。サポートされない形式の場合には、その属性を UDF に指定しなければなりません。

### インクルード・ファイル

dmbaudio.h

### 構文

内容をバッファーまたはクライアント・ファイルから保管する



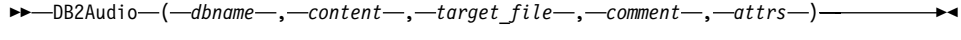

### 構文

内容をサーバー・ファイルから保管する




### 構文

ユーザー指定属性をもつ内容をバッファーまたはクライアント・ファイルから保管する




### 構文

ユーザー指定属性をもつ内容をサーバー・ファイルから保管する

### パラメーター (データ・タイプ)

**dbname (VARCHAR(18))**
現在接続されているデータベース・サーバーの名前。これは、特殊レジスター CURRENT SERVER で示されます。

**content (BLOB(2G) AS LOCATOR)**
音声の内容をもつホスト変数。ホスト変数のタイプは BLOB、BLOB_FILE、または BLOB_LOCATOR です。 DB2 は、内容のデータ・タイプを BLOB－LOCATOR にプロモートし、LOB ロケーターを DB2Audio UDF に渡します。

**format (VARCHAR(8))**
ソース音声の形式。ヌル値や空ストリングも指定できます。その場合には、オーディオ・エクステンダーがそのソース形式を自動的に判定します。音声は、そのソースと同じ形式で保管されます。サポートされる音声形式については、95ページの表5 を参照してください。

**target_file (LONG VARCHAR)**
ターゲットのサーバー・ファイルの名前 (サーバー・ファイルへ保管する場合)、またはヌル値か空ストリング (データベースへ BLOB として保管する場合)。ターゲット・ファイルには完全修飾名を指定できます。名前が修飾されていない場合は、そのファイルを見つけるのにサーバーの DB2AUDIOSTORE と DB2MMSTORE 環境変数が使用されます。

**source_file (LONG VARCHAR)**
ソースのサーバー・ファイルの名前。ソース・ファイルの名前は、完全修飾されていても、修飾されていなくてもかまいませんが、ヌル値や空ストリングであってはなりません。名前が修飾されていない場合は、そのファイルを見つけるのにサーバーの DB2AUDIOPATH と DB2MMPATH 環境変数が使用されます。

**stortype (INTEGER)**
その音声をどこに保管するかを示す値。定数 MMDB_STORAGE_TYPE_INTERNAL (値=1) の場合、音声はデータベースに BLOB として保管されます。定数 MMDB_STORAGE_TYPE_EXTERNAL (値=0) の場合、音声内容はサーバー・ファイル (データベースからポイント指定される) に保管されます。

**comment (LONG VARCHAR)**

音声とともに保管する注釈。

**attrs (LONG VARCHAR FOR BIT DATA)**
音声の属性。

**戻り値 (データ・タイプ)**
音声のハンドル (DB2AUDIO)

**例**
Anita Jones の音声クリップが入ったレコードを従業員表に挿入します。音声ソースはクライアント・バッファーにあります。音声クリップは表に BLOB として保管します。

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB (5M) aud_seg;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2AUDIO(
         CURRENT SERVER,
         :aud_seg,
         'WAVE',
         CAST(NULL as LONG VARCHAR),

         'Anita''s voice'));
```

Robert Smith の音声クリップが入ったレコードを従業員表に挿入します。音声ソースはサーバー・ファイルにあります。そのファイルは、従業員表レコードからポイントされます。

```
EXEC SQL BEGIN DECLARE SECTION;
     long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2AUDIO(
         CURRENT SERVER,
         '/employee/sounds/rsmith.wav',
         'WAV',
         :hvStorageType,
         'Robert''s voice'));
```

Anita Jones の音声クリップが入ったレコードを従業員表に挿入します。音声クリップは BLOB として保管します。サーバー・ファイルにあるソース音声クリップは、サンプリング率が 44.1 KHz で、2 つのチャネルに録音されたユーザー定義形式のものです。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr→cFormat,"cFormatA");
paudioAttr→ulSamplingRate=44100;
paudioAttr→usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
               DB2AUDIO(
               CURRENT SERVER,
               '/employee/sounds/ajones.aud',
               :hvStorageType,
               'Anita"s voice',
               :hvAudattr)
       );
```

## DB2Image

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

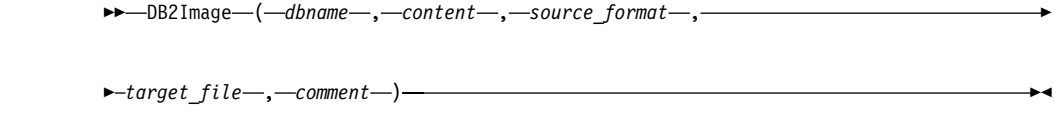
画像の内容をデータベース表に保管します。イメージ・ソースは、クライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルのどれにあってもかまいません。画像は、データベース表に BLOB として保管することも、サーバー・ファイル (データベースによって参照される) に保管することもできます。イメージ・ソースは、サポートされる形式であっても、サポートされない形式であってもかまいません。サポートされる形式の場合には、DB2 イメージ・エクステンダーがその属性を識別して保管します。サポートされない形式の場合には、その属性を UDF に指定しなければなりません。

### インクルード・ファイル

dmbimage.h

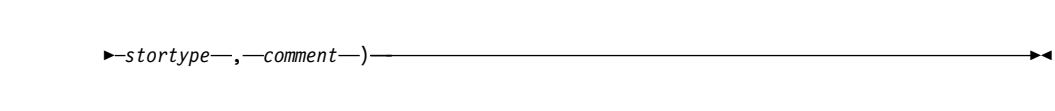
### 構文

内容をバッファーまたはクライアント・ファイルから保管する



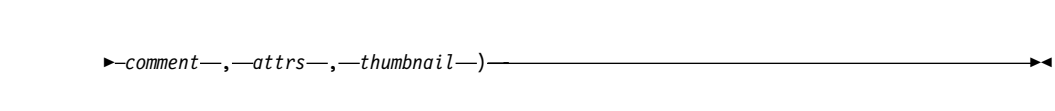

### 構文

内容をサーバー・ファイルから保管する



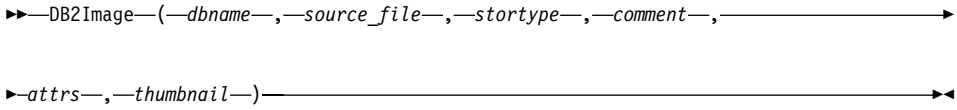

### 構文

ユーザー指定属性をもつ内容をバッファーまたはクライアント・ファイルから保管する



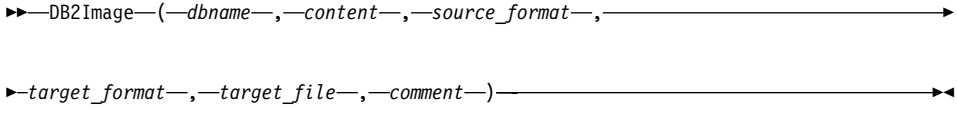
**構文**

ユーザー指定属性をもつ内容をサーバー・ファイルから保管する



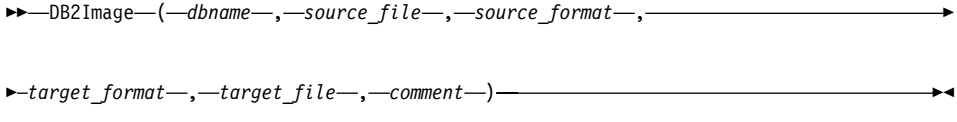

**構文**

内容をバッファーまたはクライアント・ファイルから保管し、形式変換する



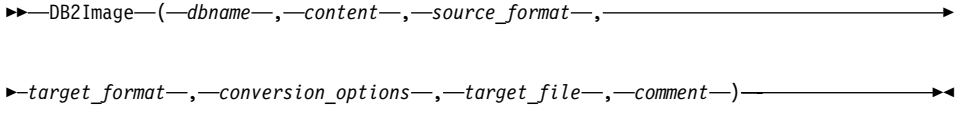

**構文**

内容をサーバー・ファイルから保管し、形式変換する



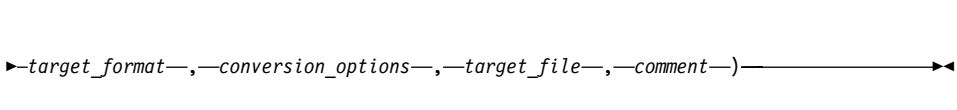

**構文**

内容をバッファーまたはクライアント・ファイルから保管し、形式変換する




**構文**

内容をサーバー・ファイルから保管し、形式変換および変更の追加を行う




**パラメーター (データ・タイプ)**

**dbname (VARCHAR(18))**

現在接続されているデータベース・サーバーの名前。これは、特殊レジスター CURRENT SERVER で示されます。

**content (BLOB(2G) AS LOCATOR)**
画像の内容をもつホスト変数。ホスト変数のタイプは BLOB、BLOB_FILE、または BLOB_LOCATOR です。 DB2 は、内容のデータ・タイプを BLOB_LOCATOR にプロモートし、LOB ロケーターを DB2Image UDF に渡します。

**source_format (VARCHAR(8))**
ソース・イメージの形式。ヌル値や、空ストリング、文字ストリング ASIS が指定できます。これら 3 つの場合、その形式はイメージ・エクステンダーが自動的に決定します。画像は、そのソースと同じ形式で保管されます。サポートされる画像形式については、95ページの表5 を参照してください。

**target_format (VARCHAR(8))**
保管後の画像の形式。ソース・イメージの形式が、適宜、変換されます。

**target_file (LONG VARCHAR)**
ターゲットのサーバー・ファイルの名前 (サーバー・ファイルへ保管する場合)、またはヌル値か空ストリング (データベースへ BLOB として保管する場合)。ターゲットのファイル名には完全修飾名を指定できます。名前が修飾されていない場合は、そのファイルを見つけるのにサーバーの DB2IMAGESTORE と DB2MMSTORE 環境変数が使用されます。画像を形式変換して保管する場合には、ターゲット・ファイルへのパスを DB2IMAGEPATH と DB2MMPATH 環境変数に指定しなければなりません。

**source_file (LONG VARCHAR)**
ソースのサーバー・ファイルの名前。ソース・ファイルの名前は、完全修飾されていても、修飾されていなくてもかまいませんが、ヌル値や空ストリングであってはなりません。名前が修飾されていないと、そのファイルを見つけるのにサーバーの DB2IMAGEPATH と DB2MMPATH 環境変数が使用されます。

**stortype (INTEGER)**
その画像をどこに保管するかを示す値。定数 MMDB_STORAGE_TYPE_INTERNAL (値=1) の場合、画像は、データベースに BLOB として保管されます。定数 MMDB_STORAGE_TYPE_EXTERNAL (値=0) の場合、イメージ内容は、サーバー・ファイル (データベースからポイント指定される) に保管されます。

**comment (LONG VARCHAR)**
画像とともに保管する注釈。

**attrs (LONG VARCHAR FOR BIT DATA)**
画像の属性。

**thumbnail (LONG VARCHAR FOR BIT DATA)**
その画像のサムネール。

**conversion_options (VARCHAR(100))**
画像の保管時に適用する回転や圧縮などの変更を指定します。サポートされる変換オプションについては、97ページの表6 を参照してください。

### 戻り値 (データ・タイプ)
画像のハンドル (DB2IMAGE)

### 例
Anita Jones の画像が入ったレコードを従業員表に挿入します。イメージ・ソースは、クライアント・バッファーにあります。画像を表に BLOB として保管します。

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (2M) hvImg
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
         :hvImg,
         'ASIS',
         CAST(NULL as LONG VARCHAR),

         'Anita''s picture'));
```

Robert Smith の画像が入ったレコードを従業員表に挿入します。イメージ・ソースは、サーバー・ファイルにあります。そのファイルは、従業員表レコードからポイントされます。画像の形式を保管時に BMP から GIF に変換します。さらに、画像を 110 ピクセルの幅と 150 ピクセルの高さに切り取り、LZW タイプの圧縮を使用して圧縮します。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2IMAGE(
         CURRENT SERVER,
         '/employee/pictures/rsmith.bmp',
```

```
        'BMP',
        'GIF',
        '-x 110 -y 150 -c 14',
        '',
        'Robert"s picture'));
```

Robert Smith の画像が入ったレコードを従業員表に挿入します。サーバー・ファイルにあるソース画像は、高さが 640 ピクセル、幅が 480 ピクセルのユーザー定義形式です。画像は BLOB として保管します。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattrt→cFormat,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          :hvStorageType,
          'Anita''s picture',
          :hvImgattrs,
          CAST(NULL as LONG VARCHAR))
       );
```

## DB2Video

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

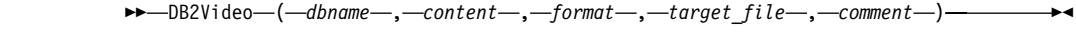
ビデオの内容をデータベース表に保管します。ビデオ・ソースは、クライアント・バッファー、クライアント・ファイル、またはサーバー・ファイルのどれにあってもかまいません。ビデオは、データベース表に BLOB として保管することも、サーバー・ファイル (データベースによって参照される) に保管することもできます。ビデオ・ソースは、サポートされる形式であっても、サポートされない形式であってもかまいません。サポートされる形式の場合には、DB2 ビデオ・エクステンダーがその属性を識別して保管します。サポートされない形式の場合には、その属性を UDF に指定しなければなりません。

### インクルード・ファイル

dmbvideo.h

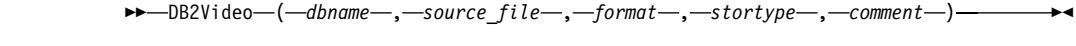
### 構文

内容をバッファーまたはクライアント・ファイルから保管する



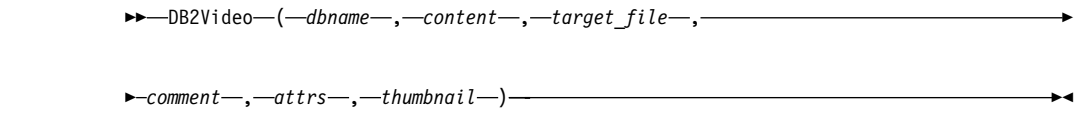

### 構文

内容をサーバー・ファイルから保管する



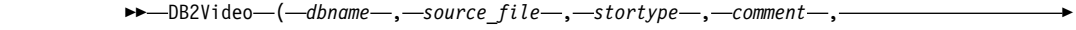

### 構文

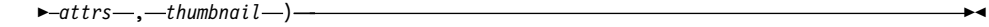
ユーザー指定属性をもつ内容をバッファーまたはクライアント・ファイルから保管する




### 構文

ユーザー指定属性をもつ内容をサーバー・ファイルから保管する






### パラメーター (データ・タイプ)

**dbname (VARCHAR(18))**
現在接続されているデータベース・サーバーの名前。これは、特殊レジスター CURRENT SERVER で示されます。

**content (BLOB(2G) AS LOCATOR)**
ビデオの内容をもつホスト変数。ホスト変数のデータ・タイプは BLOB、BLOB_FILE、または BLOB_LOCATOR です。 DB2 は、内容を BLOB_LOCATOR にプロモートし、LOB ロケーターを DB2Video UDF に渡します。内容がクライアント・バッファーにある場合は、ビデオ・ヘッダーが完全に読み取れるように、そのバッファーには、その内容の少なくとも最初の 640KB が入っていなければなりません。

**format (VARCHAR(8))**
ソース・ビデオの形式。ヌル値や空ストリングも指定できます。ビデオ・エクステンダーがそのソース形式を自動的に判定します。ビデオは、そのソースと同じ形式で保管されます。サポートされるビデオ形式については、95ページの表5 を参照してください。 MPG1 形式の場合、 MPG1、mpg1、MPEG1、または mpeg1 を指定できます。 MPG2 形式の場合、MPG2、mpg2、MPEG2、または mpeg2 を指定できます。

**target_file (LONG VARCHAR)**
ターゲットのサーバー・ファイルの名前 (サーバー・ファイルへ保管する場合)、またはヌル値か空ストリング (データベースへ BLOB として保管する場合)。サーバー・ファイルは、完全修飾名にすることができます。ファイル名が修飾されていない場合は、サーバーの DB2VIDEOSTORE と DB2MMSTORE 環境変数を使ってファイルが見つけられます。

**source_file (LONG VARCHAR)**
ソースのサーバー・ファイルの名前。この名前は、完全修飾されていても、修飾されていなくてもかまいませんが、ヌル値や空ストリングであってはなりません。名前が修飾されていないと、サーバーの DB2VIDEOPATH と DB2MMPATH 環境変数を使って、そのファイルが見つけられます。

**stortype (INTEGER)**
そのビデオをどこに保管するかを示す値。定数 MMDB_STORAGE_TYPE_INTERNAL (値=1) の場合、ビデオはデータ

ベースに BLOB として保管されます。定数 MMDB_STORAGE_TYPE_EXTERNAL (値=0) の場合、ビデオ内容はサーバー・ファイル (データベースからポイントされる) に保管されます。

**comment (LONG VARCHAR)**
ビデオとともに保管する注釈。

**attrs (LONG VARCHAR FOR BIT DATA)**
ビデオの属性。

**thumbnail (LONG VARCHAR FOR BIT DATA)**
そのビデオを代表するサムネールの画像。

### 戻り値 (データ・タイプ)

ビデオのハンドル (DB2VIDEO)

### 例

Anita Jones のビデオ・クリップが入ったレコードを従業員表に挿入します。ビデオ・ソースは、クライアント・バッファーにあります。ビデオ・クリップは表に BLOB として保管します。

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (8M) vid;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2VIDEO(
         CURRENT SERVER,
         :vid,
         'MPEG1',
         CAST(NULL as LONG VARCHAR),
         'Anita''s video'));
```

Robert Smith のビデオ・クリップが入ったレコードを従業員表に挿入します。ビデオ・ソースはサーバー・ファイルにあります。そのファイルは、従業員表のレコードからポイントされます。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2VIDEO(
```

```
        CURRENT SERVER,
        '/employee/videos/rsmith.mpg',
        'MPEG1',
        :hvStorageType,
        'Robert''s video'));
```

ビデオ・クリップが入ったレコードをデータベース表に挿入します。ソース・ビデオ・クリップ (その内容はサーバー・ファイルにある) の形式はユーザー定義形式です。ビデオ内容は、サーバー・ファイルに保持します (データベース表のレコードからそのファイルをポイントする)。さらに、そのビデオを代表するサムネールを保管します。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvVidattrs;
struct {
       short len;
       char data[10000];
}hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr→cFormat,"Formatv");
pvideoAttr.len=sizeof(MMDBVideoAttrs);
⋮

/* Generate thumbnail and assign data   */
/* in video structure                   */
⋮

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.vid',
          :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
          :hvThumbnail)
       );
```

## Duration

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

WAVE オーディオや AIFF オーディオ、またはビデオの所要時間 (つまり、秒単位による再生時間) を戻します。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ビデオ、または WAVE、AIFF、あるいはユーザー定義形式のオーディオの、秒単位の所要時間 (INTEGER)。他の形式の音声の場合はヌル値を戻します。

### 例

従業員表のビデオ列に保管されているすべてのビデオの所要時間を表示します。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvDur_vid;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT DURATION(VIDEO)
     INTO :hvDur_vid
     FROM EMPLOYEE;
```

## Filename

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

オブジェクト内容がファイル (データベース表からポイントされる) にある場合、画像、音声、またはビデオの内容が入っているサーバー・ファイルの名前を戻します。画像、音声、またはビデオが BLOB としてデータベース表に保管されている場合には、ヌル値が戻されます。

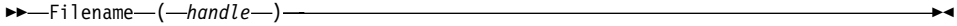
### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

オブジェクト内容がサーバー・ファイルにある場合、(VARCHAR(250))、オブジェクトが BLOB として保管されている場合には、ヌル値。

### 例

従業員表の Robert Smith という項目に対応するビデオのファイル名を表示します。

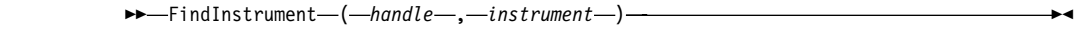
```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

## FindInstrument

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

MIDI オーディオにおいて、指定する楽器が最初に現れるトラック番号を戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

**instrument (VARCHAR(255))**
探索する楽器の名前。オーディオ・エクステンダーは、指定する名前と完全に一致する名前の楽器を探します。

### 戻り値 (データ・タイプ)
指定する楽器名が最初に現れるトラック番号 (SMALLINT)。指定する名前の楽器が見つからないと、値 -1 が戻されます。その他の形式の音声の場合は、NULL が戻されます。

### 例
従業員表のサウンド列に保管されている Robert Smith の MIDI オーディオで PIANO が最初に現れる場所を見つけます。

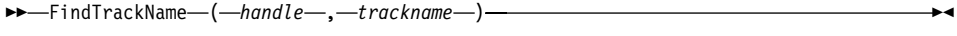
```
EXEC SQL BEGIN DECLARE SECTION;
      short hvInstr;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDINSTRUMENT(SOUND, 'PIANO')
      INTO :hvInstr
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

## FindTrackName

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

MIDI オーディオにおける指定の名前のトラックの番号を戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

**trackname (VARCHAR(255))**
探索するトラックの名前。オーディオ・エクステンダーは、指定する名前と完全に一致する名前のトラックを探します。

### 戻り値 (データ・タイプ)
指定の楽器名の、指定されたトラックの番号 (SMALLINT)。指定の名前が見つからないと、値 -1 が戻されます。他の形式の音声の場合は、ヌル値が戻されます。

### 例
Robert Smith の MIDI オーディオ録音に WELCOME という名前のトラックがあるかどうかを判別します。オーディオ録音は、従業員表のサウンド列に保管されています。

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvTrack;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDTRACKNAME(SOUND,
           'WELCOME')
      INTO :hvTrack
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

## Format

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオの形式を戻します。

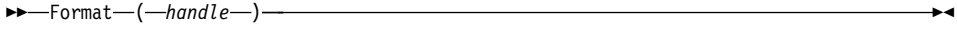
### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

画像や、音声、ビデオの形式 (VARCHAR(8))。サポートされる画像、音声、およびビデオの形式については、95ページの表5 を参照してください。

### 例

従業員表のピクチャー列に保管されているすべての従業員のうち、その画像の形式が GIF 形式である従業員の名前をすべて入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvName[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
     INTO :hvName
     FROM EMPLOYEE
     WHERE FORMAT(PICTURE)='GIF';
```

## FrameRate

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオのスループットを秒当たりのフレーム数で戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
ビデオのフレーム率 (SMALLINT)。スループット率が変動する場合は、ヌル値を戻します。

### 例
Anita Jones の従業員表のビデオ列に保管されているビデオの枠率を入手しています。

```
EXEC SQL BEGIN DECLARE SECTION;
short hvFm_rate;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FRAMERATE (VIDEO)
     FROM EMPLOYEE
      INTO :hvFm_rate
     WHERE NAME='Anita Jones';
```

## GetInstruments

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

MIDI オーディオにおけるすべての楽器の楽器名を戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
MIDI オーディオにおけるすべての楽器の楽器名 (VARCHAR(1536))。値は、トラック番号順に戻されます (たとえば、PIANO; TRUMPET; BASS など)。結果は *n* 個のフィールドに分けられます。ここで *n* は MIDI オーディオ内のトラック数です。トラックに関連する楽器がない場合、そのフィールドはブランクになります。 MIDI 以外のオーディオ形式の場合は、ヌル値が戻されます。

### 例
Robert Smith の MIDI オーディオ録音にあるすべての楽器を見つけます (つまり、トラック番号と楽器名)。オーディオ録音は、従業員表のサウンド列に保管されています。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_Instr[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETINSTRUMENTS(SOUND)
      INTO :hvAud_Instr
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

## GetTrackNames

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

MIDI オーディオにおけるすべてのトラックの名前を戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
MIDI オーディオにおけるすべてのトラックの名前 (VARCHAR(1536))。値は、トラック番号順に戻されます (たとえば、PIANO TUNE; TRUMPET FANFARE など)。結果は *n* 個のフィールドに分けられます。ここで *n* は MIDI オーディオ内のトラック数です。トラックに名前がない場合、そのフィールドはブランクになります。 MIDI 以外のオーディオ形式の場合は、ヌル値が戻されます。

### 例
従業員表のサウンド列に保管されている Robert Smith の MIDI オーディオ録音にあるすべてのトラック番号とトラック名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvTracks[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETTRACKNAMES(SOUND)
      INTO :hvTracks
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

## Height

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | O |

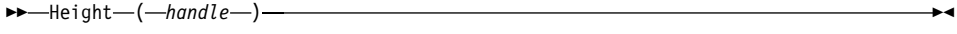
画像やビデオ・フレームの高さをピクセル数で戻します。

### インクルード・ファイル

**イメージ** dmbimage.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2IMAGE または DB2VIDEO)**
画像またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ピクセル数での高さ (INTEGER)

### 例

従業員表のピクチャー列に保管されているすべての画像のうち、高さが 500 ピクセル未満のすべての画像のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE HEIGHT(PICTURE)<500;
```

## Importer

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオをデータベース表に保管した人のユーザー ID を戻します。

### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

インポーターのユーザー ID (CHAR(8))

### 例

ユーザー ID が rsmith で従業員表のサウンド列に保管されているすべての音声のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE IMPORTER(SOUND)='rsmith';
```

## ImportTime

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオがいつデータベース表に保管されたのかを示す時刻スタンプを戻します。

### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

画像や、音声、ビデオが保管された時刻スタンプ (TIMESTAMP)。

### 例

従業員表のピクチャー列に 1 年以上前に保管されたすべての画像のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
        IMPORTTIME(PICTURE))>365;
```

## MaxBytesPerSec

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオの最大スループットを秒当たりのバイト数で戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
ビデオのスループット (INTEGER)。スループット率が変動する場合は、ヌル値を戻します。

### 例
Anita Jones の従業員表のビデオ列に保管されているビデオの最大スループットを入手しています。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvMax_BytesPS;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT MAXBYTESPERSEC(VIDEO)
     INTO :hvMax_BytesPS
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

## NumAudioTracks

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

ビデオや MIDI オーディオにおける音声トラックの数を戻します。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ビデオや MIDI オーディオの音声トラックの数 (SMALLINT)。他の形式の音声の場合はヌル値を戻します。

### 例

音声トラックがないビデオ・ファイルの名前を従業員表のビデオ列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO) = 0;
```

## NumChannels

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

WAVE オーディオや AIFF オーディオ、またはビデオに録音されたオーディオ・チャネルの数を戻します。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ビデオ、または WAVE オーディオや AIFF オーディオに録音されているオーディオ・チャネルの数 (SMALLINT)。他の形式の音声の場合はヌル値を戻します。

### 例

ステレオで (つまり、2 つのチャネルで) 録音されているすべての音声ファイルの名前を従業員表のサウンド列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE NUMCHANNELS(SOUND) = 2;
```

## NumColors

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像における色の数を戻します。

### インクルード・ファイル
dmbimage.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2IMAGE)**
画像のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
画像の色数 (INTEGER)

### 例
16 色より少ない画像のイメージ・ファイルの名前を従業員表のピクチャー列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NUMCOLORS(PICTURE) < 16;
```

## NumFrames

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

画像におけるフレームの数を戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
ビデオのフレーム数 (INTEGER)。スループット率が変動する場合は、ヌル値を戻します。

### 例
Robert Smith の従業員表のビデオ列に保管されているビデオの枠数を入手しています。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvNum_Frames;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NUMFRAMES (VIDEO)
     INTO :hvNum_Frames
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

## NumVideoTracks

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオでのビデオ・トラックの数を戻します。

### インクルード・ファイル
dmbvideo.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2VIDEO)**
ビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
ビデオ・トラックの数 (SMALLINT)

### 例
複数のビデオ・トラックをもつすべてのビデオのファイル名を従業員表のビデオ列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMVIDEOTRACKS(VIDEO) > 1;
```

## QbScoreFromName

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像の得点を戻します。これは、画像のフィーチャーが照会オブジェクトのフィーチャーにどの程度一致するかを表す数値です。その画像ハンドルが属している列の QBIC カタログが、その画像の得点を計算するときに使用されます。得点が低ければ低いほど、画像のフィーチャーは指定された照会オブジェクトのフィーチャーに類似しています。 (QbScoreFromName は QbScore の代わりに使用するようになりましたが、QbScore もまだ使用することができます。)

**注:**

1. **EEE のみ:** QbScoreFromName は区分データベース環境ではサポートされません。その代わりに、QbQueryGetString API を使用して照会ストリングを入手してから、QbScoreFromStr UDF を使用してください。
2. 非区分データベース環境の場合、今後のリリースでは QbScoreFromName を使用できません。照会を再使用するには、QbQueryGetString API を使用して照会ストリングを入手し、それを保管して、今後のアプリケーションでの使用に備えてください。

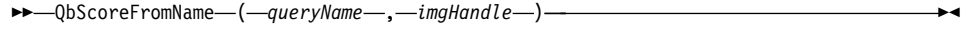
### インクルード・ファイル

なし

### 構文


### 構文

**使用できないバージョン**




### パラメーター (データ・タイプ)

**imgHandle (DB2Image)**
画像のハンドル。

**queryName (varchar(18))**
照会オブジェクトの名前。

### 戻り値 (データ・タイプ)

画像の得点 (DOUBLE)。得点は 0.0 から無限大までの範囲です。得点が低ければ低いほど、ターゲットの画像のフィーチャー値は、照会に指定したフィーチャー値に類似しています。得点が 0.0 であれば、正確に一致しています。得点がヌル値であれば、画像がカタログされていない、という意味です。使用できないバージョンの UDF では、画像がカタログされていなかった場合に得点 -1 を戻します。

### 例

表列にカタログされている画像で、平均色が赤に最も近いものを見つけます。

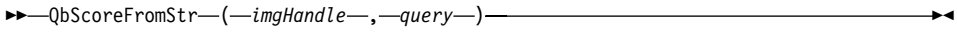
```
EXEC SQL BEGIN DECLARE SECTION;
char Img_fnd[100];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
  INTO :Img_fnd
  FROM FABRIC
  WHERE (QBSCOREFROMNAME(SWATCH_IMG,
         'fshavgcol'))<0.1;
```

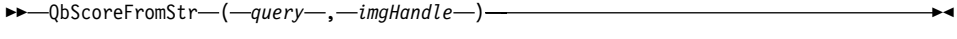
## QbScoreFromStr

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像の得点を戻します。これは、画像のフィーチャーが照会ストリングのフィーチャーにどの程度一致するかを表す数値です。その画像ハンドルが属している列に関連する QBIC カタログが、その画像の得点を計算するときに使用されます。得点が低ければ低いほど、画像のフィーチャーは照会ストリングのフィーチャーに類似しています。

### インクルード・ファイル
なし

### 構文



### 構文
**使用できないバージョン**



### パラメーター (データ・タイプ)

**imgHandle (DB2Image)**
画像のハンドル。

**query (VARCHAR(1024))**
照会ストリング。

### 戻り値 (データ・タイプ)
画像の得点 (DOUBLE)。得点は 0.0 から無限大までの範囲です。得点が低ければ低いほど、ターゲットの画像のフィーチャー値は、照会に指定したフィーチャー値に類似しています。得点が 0.0 であれば、正確に一致しています。得点がヌル値であれば、画像がカタログされていない、という意味です。使用できないバージョンの UDF では、画像がカタログされていなかった場合に得点 -1 を戻します。

### 例
表列にカタログされている画像で、平均色が赤に最も近いものを見つけます。

```
SELECT name
FROM fabric
WHERE (QbScoreFromStr(Swatch_Img,
       'QbColorFeatureClass color=<255, 0, 0>'))<0.1
```

## QbScoreTBFromName

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像列の得点表を戻します。各得点は、画像のフィーチャーが照会オブジェクトのフィーチャーにどの程度一致するかを表す数値です。指定の表およびその画像ハンドルが属している列に関連する QBIC カタログが、各画像の得点を計算するときに使用されます。画像の得点が低ければ低いほど、その画像のフィーチャーは照会オブジェクトのフィーチャーに類似しています。

**注:**

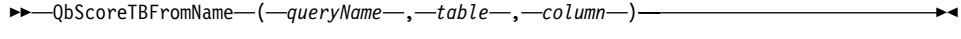
1. **EEE のみ:** QbScoreTBFromName は区分データベース環境ではサポートされません。その代わりに、QbQueryGetString API を使用して照会ストリングを入手してから、QbScoreFromStr UDF を使用してください。
2. 非区分データベース環境の場合、今後のリリースでは QbScoreTBFromName を使用できません。照会を再使用するには、QbQueryGetString API を使用して照会ストリングを入手し、それを保管して、今後のアプリケーションでの使用に備えてください。

### インクルード・ファイル

なし

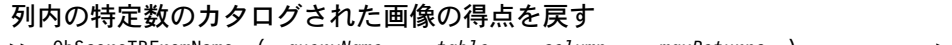
### 構文

**列内のすべてのカタログされた画像の得点を戻す**




### 構文

**列内の特定数のカタログされた画像の得点を戻す**




### パラメーター (データ・タイプ)

**queryName (VARCHAR(18))**
照会オブジェクトの名前。

**table (CHAR(18))**
この画像列が入っている表の修飾名。表スキーマが、DB2 エクステンダー・サービスを開始するために使用するユーザー ID と同じであれば、非修飾表名を使用できます。

**column (CHAR(18))**
画像列の名前。

**maxReturns (INTEGER)**
結果の表が戻すハンドルの最大数。この値を指定しない場合、戻されるハンドルの最大数は 100 になります。

### 戻り値 (データ・タイプ)

画像ハンドルと列内の画像の得点の表。結果表には 2 つの列があります。画像ハンドルが入っている IMAGE_ID (DB2Image) と、得点が入っている SCORE (DOUBLE) です。結果表は得点の昇順に並べられます。得点は 0.0 から無限大までの範囲です。得点が低ければ低いほど、ターゲットの画像のフィーチャー値は、照会に指定したフィーチャー値に類似しています。得点が 0.0 であれば、正確に一致しています。得点が -1 であれば、画像がカタログされていない、という意味です。

### 例

表列内の画像のテクスチャーを、照会オブジェクトに指定したテクスチャーと比較し、画像ハンドルとその得点を戻します。

```
SELECT name, description
INTO :hvName, :hvDesc
FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 (SELECT CAST (image_id as varchar(250)) FROM TABLE
 (QbScoreTBFromName
     'fstxtr',
     'clothes.fabric',
     'swatch_img'))
 AS T1));
```

## QbScoreTBFromStr

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像列からの得点の表を戻します。各得点は、画像のフィーチャーが照会ストリングに指定されたフィーチャーにどの程度一致するかを表す数値です。表およびその画像ハンドルが属している列に関連する QBIC カタログが、各画像の得点を計算するときに使用されます。画像の得点が低ければ低いほど、その画像のフィーチャーは照会ストリングのフィーチャーに類似しています。

### インクルード・ファイル
なし

### 構文
**列内のすべてのカタログされた画像の得点を戻す**




### 構文
**列内の特定数のカタログされた画像の得点を戻す**




### パラメーター (データ・タイプ)

**query (VARCHAR(1024))**
照会ストリング。

**table (CHAR(18))**
この画像列が入っている表の修飾名。表スキーマが、DB2 エクステンダー・サービスを開始するために使用するユーザー ID と同じであれば、非修飾表名を使用できます。

**column (CHAR(18))**
照会する画像列。

**maxReturns (INTEGER)**
結果の表が戻すハンドルの最大数。この値を指定しない場合、戻される画像ハンドルの最大数は 100 になります。

### 戻り値 (データ・タイプ)

画像ハンドルと列内の画像の得点の表。結果表には 2 つの列があります。画像ハンドルが入っている IMAGE_ID (DB2Image) と、得点が入っている SCORE (DOUBLE) です。結果表は得点の昇順に並べられます。得点は 0.0 から無限大までの範囲です。得点が低ければ低いほど、ターゲットの画像のフィーチャー値は、照会に指定したフィーチャー値に類似しています。得点が 0.0 であれば、正確に一致しています。得点が -1 であれば、画像がカタログされていない、という意味です。

### 例

表列内にカタログされている画像で、サーバー・ファイル内の画像のテクスチャーに最も近いテクスチャーをもつ画像を 10 個見つけます。

```
SELECT name, description
FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 (SELECT CAST (image_id as varchar(250)) FROM TABLE
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'
      'clothes.fabric',
      'swatch_img',
      10))
  AS T1));
```

## Replace

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

データベースに保管されている画像や、音声、ビデオの内容を更新します。また、その注釈を更新します。

### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

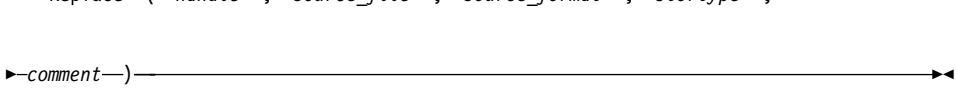
### 構文

内容をバッファーまたはクライアント・ファイルから更新し、注釈を更新する




### 構文

内容をサーバー・ファイルから更新し、注釈を更新する

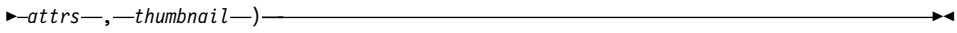



### インクルード・ファイル

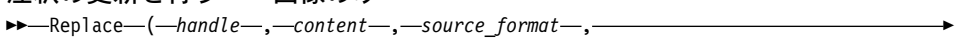
ユーザー指定属性をもつ内容をバッファーかクライアント・ファイルから更新し、注釈を更新する

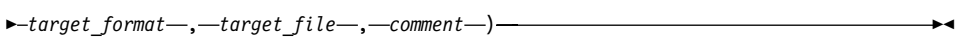

### インクルード・ファイル

ユーザー指定属性をもつ内容をサーバー・ファイルから更新し、注釈を更新する

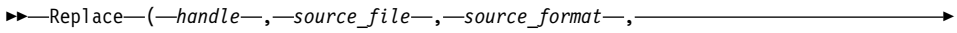


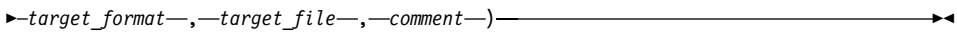

**構文**
内容をバッファーまたはクライアント・ファイルから更新し、形式変換および注釈の更新を行う － 画像のみ



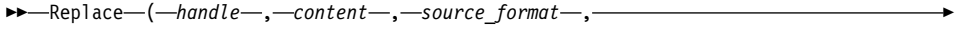



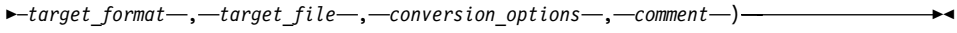

**構文**
内容をサーバー・ファイルから更新し、形式変換を行い、注釈を更新する － 画像のみ



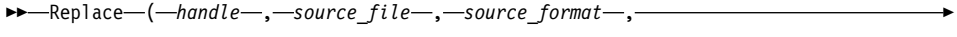



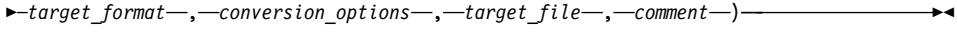

**構文**
内容をバッファーまたはクライアント・ファイルから更新し、形式変換および追加の変更を行い、注釈を更新する － 画像のみ




**構文**
内容をサーバー・ファイルから更新し、形式変換および追加の変更を行い、注釈を更新する － 画像のみ




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

**source_file (LONG VARCHAR)**
画像や、音声、ビデオの更新内容をもつファイルの名前。

**target_file (LONG VARCHAR)**
更新前の画像や、音声、ビデオの内容をもつファイルの名前。

**create_target (INTEGER)**
ソース内容がサーバー・ファイルにある場合にターゲット・ファイルを作成するかどうかを指定する値。値は 0 か 1 です。値が 0 の場合、ターゲット・ファイルは作成されません (実際には、取り出しが行われません)。値が 1 の場合、ファイルが作成されます (ターゲット・ファイルがすでにある場合には、この値によって、実際にはそのファイルが重ね書きされます)。ソース内容が BLOB の場合には、ターゲット・ファイルが常に作成されます (ファイルがすでにある場合には、それが重ね書きされます)。

**target_format (VARCHAR(8))**
取り出し後の画像の形式。ソース・イメージの形式が、適宜、変換されます。その内容を形式変換して更新する場合には、そのターゲット・ファイルへのパスを DB2IMAGEPATH と DB2MMPATH 環境変数に指定する必要があります。 MPG1 形式の場合、 MPG1、mpg1、MPEG1、または mpeg1 を指定できます。 MPG2 形式の場合、MPG2、mpg2、MPEG2、または mpeg2 を指定できます。

**content (BLOB(2G) AS LOCATOR)**
画像や、音声、ビデオの更新内容をもつホスト変数。ホスト変数のタイプは BLOB、BLOB_FILE、または BLOB_LOCATOR です。 DB2 は、データ・タイプを BLOB_LOCATOR にプロモートし、LOB ロケーターを Replace UDF に渡します。

**source_format (VARCHAR(8))**
画像や、音声、ビデオの更新ソースの形式。ヌル値や空ストリングを指定できます。あるいは、画像の場合、文字ストリング ASIS を指定できます。これら 3 つの場合、その形式はエクステンダーが自動的に判定します。 MPG1 形式の場合、 MPG1、mpg1、MPEG1、または mpeg1 を指定できます。 MPG2 形式の場合、MPG2、mpg2、MPEG2、または mpeg2 を指定できます。

**comment (LONG VARCHAR)**
注釈。

**attrs (LONG VARCHAR FOR BIT DATA)**
画像や、音声、ビデオの属性。

**thumbnail (LONG VARCHAR FOR BIT DATA)**
その画像やビデオ・フレームのサムネール (画像とビデオのみ)。

**conversion_options (VARCHAR(100))**
画像の更新時に適用する回転や圧縮などの変更を指定します。サポートされる変換オプションについては、97ページの表6 を参照してください。

### 戻り値 (データ・タイプ)
更新される画像や、音声、ビデオのハンドル (DB2IMAGE、DB2AUDIO、または DB2VIDEO)。

### 例
従業員表のピクチャー列に保管されている Anita Jones の画像を更新します。その際、その画像の形式を BMP から GIF へ変換し、注釈を更新します。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
  SET PICTURE = REPLACE(PICTURE,
        '/employee/newimg/ajones.bmp',
        'BMP',
        'GIF',
        :hvStorageType,
        'Anita''s new picture')
  WHERE NAME='Anita Jones';
```

## SamplingRate

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | O |

WAVE オーディオや AIFF オーディオ、またはビデオの音声トラックのサンプリング率を秒当たりのサンプル数で戻します。

### インクルード・ファイル

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2AUDIO または DB2VIDEO)**
音声またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

ビデオ、または WAVE オーディオや AIFF オーディオのサンプリング率 (INTEGER)。他の形式の音声の場合はヌル値を戻します。

### 例

サンプリング率が 44.1 KHz のすべての音声について、それらのファイル名を従業員表のサウンド列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE SAMPLINGRATE(SOUND) = 44100;
```

## Size

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

画像や、音声、ビデオのサイズをバイト数で戻します。

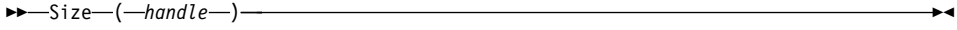
### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

画像や、音声、ビデオのサイズのバイト数 (INTEGER)。

### 例

従業員表のピクチャー列に保管されているすべての画像のうち、サイズが 310 KB より大きいすべての画像のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE SIZE(PICTURE) > 310000;
```

## Thumbnail

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | O |

データベースに保管されているサムネールの画像やビデオ・フレームを戻したり、更新したりします。

### インクルード・ファイル

**イメージ** dmbimage.h

**ビデオ** dmbvideo.h

### 構文

**サムネールの検索**




### 構文

**サムネールの更新**




### パラメーター (データ・タイプ)

**handle (DB2IMAGE または DB2VIDEO)**
画像またはビデオのハンドルをもつ列名またはホスト変数。

**new_thumbnail (LONG VARCHAR FOR BIT DATA)**
サムネールの更新ソースの内容。

### 戻り値 (データ・タイプ)

取り出しの場合、取り出したサムネールの内容 (LONG VARCHAR FOR BIT DATA)。更新の場合、画像かビデオのハンドル (DB2IMAGE または DB2VIDEO)。

### 例

従業員表に保管されている Anita Jones のサムネールの画像を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 struct{
       short len;
       char data [32000];
         }hvThumbnail;
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :hvThumbnail
     FROM EMPLOYEE
     WHERE NAME =  'Anita Jones';
```

従業員表に保管されている Anita Jones のビデオに関連したサムネールを更新します。

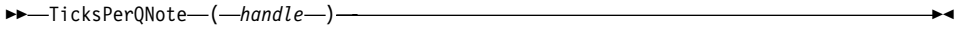
```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;

/*  Create thumbnail and    */
/*  store in hvThumbnail    */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=THUMBNAIL(
                VIDEO,
                :hvThumbnail)
   WHERE NAME='Anita Jones';
```

## TicksPerQNote

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

録音されている MIDI オーディオのクロック速度を四分音符当たりのティック数で戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
MIDI オーディオの四分音符当たりのクロック・ティック数 (SMALLINT)。他の形式の音声の場合はヌル値を戻します。

### 例
四分音符当たりのクロック・ティック数が 200 より大きいすべての MIDI オーディオのファイル名を従業員表のサウンド列から入手します。

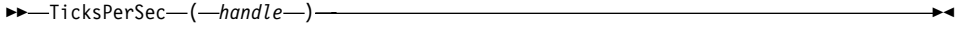
```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='MIDI'
      AND TICKSPERQNOTE(SOUND)>200;
```

## TicksPerSec

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

録音されている MIDI オーディオのクロック速度を秒当たりのティック数で戻します。

### インクルード・ファイル
dmbaudio.h

### 構文



### パラメーター (データ・タイプ)

**handle (DB2AUDIO)**
音声のハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)
MIDI オーディオの秒当たりのクロック・ティック数 (SMALLINT)。他の形式の音声の場合はヌル値を戻します。

### 例
秒当たりのクロック・ティック数が 50 未満のすべての MIDI オーディオのファイル名を従業員表のサウンド列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='MIDI'
      AND TICKSPERSEC(SOUND)<50;
```

## Updater

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

データベース表の画像や、音声、ビデオを最後に更新した人のユーザー ID を戻します。

### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

画像や、音声、ビデオを最後に更新した人のユーザー ID (CHAR(8))。

### 例

従業員表のビデオ列に保管されている Robert Smith のビデオを最後に更新した人のユーザー ID を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdater[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(VIDEO)
     INTO :hvUpdater
     FROM EMPLOYEE
     WHERE NAME='rsmith';
```

## UpdateTime

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

データベース表の画像や、音声、ビデオが最後に更新されたのがいつかを示す時刻スタンプを戻します。

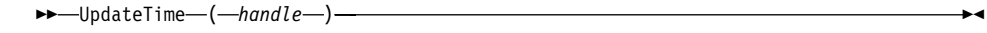
### インクルード・ファイル

**イメージ** dmbimage.h

**オーディオ** dmbaudio.h

**ビデオ** dmbvideo.h

### 構文




### パラメーター (データ・タイプ)

**handle (DB2IMAGE、DB2AUDIO、または DB2VIDEO)**
画像、音声、またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

画像や、音声、ビデオが最後に更新された時刻スタンプ (TIMESTAMP)。

### 例

過去 2 日の間に更新された画像のファイル名を従業員表のピクチャー列から入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
       UPDATETIME(PICTURE))< 2;
```

## Width

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | O |

画像やビデオ・フレームの幅をピクセル数で戻します。

### インクルード・ファイル

イメージ　dmbimage.h

ビデオ　dmbvideo.h

### 構文


### パラメーター (データ・タイプ)

**handle (DB2IMAGE または DB2VIDEO)**
画像またはビデオのハンドルをもつ列名またはホスト変数。

### 戻り値 (データ・タイプ)

幅のピクセル数 (INTEGER)

### 例

従業員表のピクチャー列に保管されているすべての画像のうち、幅が 300 ピクセル未満のすべての画像のファイル名を入手します。

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE WIDTH(PICTURE)<300;
```

# 第16章 アプリケーション・プログラミング・インターフェース

この章では、DB2 エクステンダーの管理 API の参照情報を提供します。 API は、アルファベット順に列挙します。

それぞれの API に関して、次の情報を提供します。

- その API を提供するエクステンダー
- 要旨
- その API を使用するために必要な許可
- その API のライブラリー・ファイル
- その API のインクルード (ヘッダー) ファイル
- その API 呼び出しの C 構文
- その API パラメーターの説明
- その API の戻り値
- 使用例

## DBaAdminGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

ユーザー表の音声列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**SQL_ERROR または他の SQL 戻りコード**
DB2 から戻されたエラー。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ユーザー ID rsmith が所有する表の音声列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
#include <dmbaudio.h>
long idx;
rc = DBaAdminGetInaccessibleFiles("rsmith",
       &count, &filelist);
```

## DBaAdminGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

ユーザー表の音声列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この

修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ajones が所有する表の音声列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbaudio.h>
long idx;
rc = DBaAdminGetReferencedFiles("ajones",
       &count, &fileList);
```

## DBaAdminIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

指定されているファイルを参照する、ユーザー表の音声列項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**fileName (in)**
参照されるファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ファイル /audios/asmith.wav を参照する、現行データベースのすべての表の音声列の項目のリストを表示するには、次のようにします。

```
#include <dmbaudio.h>
long idx;
rc = DBaAdminIsFileReferenced(NULL,
       "/audios/asmith.wav",
       &count, &tableList);
```

## DBaAdminReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

音声関連のメタデータ表を 『クリーンアップ』 します。以下に例を示します。

- 音声メタデータ表で使用されなくなったスペースを再利用します。
- 音声メタデータ表にある、存在しなくなったオーディオ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaAdminReorgMetadata(
      char *qualifier
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表がクリーンアップされます。ヌル値を指定すると、現行データベースのすべての表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ユーザー ID rsmith が所有する表の音声列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaAdminReorgMetadata("rsmith");
```

## DBaDisableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

列を音声 (DB2Audio データ) の場合に使用不可にして、オーディオ・データを保持できないようにします。列項目の内容は NULL に設定され、この列に関連するメタデータは消去されます。この列の、オーディオ・エクステンダーで定義したすべてのトリガーも消去されます。使用不可にした列が入っている表に新しい行が挿入され、それらの新しい行に DB2Audio タイプを使用して定義したデータが入っていることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaDisableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
この音声列が入っている表の名前。

**colName (in)**
音声列の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のサウンド列を音声 (DB2Audio データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaDisableColumn("employee", "sound");
```

## DBaDisableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

データベースを音声 (DB2Audio データ) の場合に使用不可にして、音声データを保持できないようにします。また、DB2Audio に定義されているデータベースのすべての表も使用不可にします。このデータベースの、オーディオ・エクステンダーで定義したメタデータおよび UDF は消去されます。 DB2Audio タイプを使用して定義されている、データベースに入っている表に新しい行が挿入されることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

DBADM、SYSADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaDisableDatabase(
     );
```

### パラメーター

DBaDisableDatabase にはパラメーターがありません。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

現行データベースを音声 (DB2Audio データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaDisableDatabase();
```

## DBaDisableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

表を音声 (DB2Audio データ) の場合に使用不可にして、音声データを保持できないようにします。また、DB2Audio に定義されている表のすべての列も使用不可にします。この表の、オーディオ・エクステンダーで定義した一部のメタデータは消去されます。 DB2Audio タイプを使用して定義されている表に新しい行が挿入されることがありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaDisableTable(
      char *tableName
      );
```

### パラメーター

**tableName (in)**
音声列が入っている表の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表を音声 (DB2Audio データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaDisableTable("employee");
```

## DBaEnableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

列を音声 (DB2Audio データ) 用に使用可能にします。この API は、この列とメタデータ表との間の関係を定義して管理します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

Control、Alter、SYSADM、DBADM

API パラメーターで指定された表スペースおよびバッファー・プールに対する使用特権も必要です。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaEnableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
この音声列が入っている表の名前。

**colName (in)**
音声列の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
列はすでに使用可能になっています。

**MMDB_RC_WRONG_SIGNATURE**
指定された列のデータ・タイプが正しくありません。ユーザー定義のデータ・タイプ MMDBSYS.DB2AUDIO が期待されています。

**MMDB_RC_COLUMN_DOESNOT_EXIST**
指定された表に列が定義されていません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NOT_ENABLED**
データベースまたは表が使用可能になっていません。

## 例

従業員表のサウンド列を音声 (DB2Audio データ) 用に使用可能にするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaEnableColumn("employee", "sound");
```

## DBaEnableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

データベースを音声 (DB2Audio データ) 用に使用可能にします。この API は、データベースごとに 1 回呼び出します。 DB2 ユーザー定義タイプ DB2Audio をデータベース・マネージャーに定義します。また、DB2Audio データを操作するすべての UDF も作成します。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

DBADM、SYSADM、SYSCTRL

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaEnableDatabase(
      char *tableSpace
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに *longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** エクステンダー用のデータベースを使用可能にしたときに指定した表スペースは、区分データベース・システムのすべてのノードを含むノードグループで定義する必要があります。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
このデータベースはすでに使用可能になっています。

**MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER**
接続先のサーバーがこのコマンドをサポートしていません。

**MMDB_WARN_NOT_ALL_NODES**
指定された表スペースには、エクステンダー用のすべてのノードが含まれていません。 **(EEE のみ)**

**MMDB_RC_NOT_SAME_NODEGROUP**
指定された表スペースが同じノード・グループにありません。 **(EEE のみ)**

## 例

表スペース MYTS で、現行データベースを音声 (DB2Audio データ) 用に使用可能にします。索引表スペースおよび長形式表スペースに省略時値を使用します。

```
#include <dmbaudio.h>
rc = DBaEnableDatabase("myts,,");
```

現行データベースを音声 (DB2Audio データ) 用に使用可能にします。省略時表スペースを使用します。

```
#include <dmbaudio.h>
rc = DBaEnableDatabase(NULL);
```

## DBaEnableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

表を音声 (DB2Audio データ) 用に使用可能にします。この API は、表ごとに 1 回呼び出します。表の音声列属性を保管し、管理するためのメタデータ表を作成します。ロッキングが発生する可能性を回避するため、アプリケーションでトランザクションをコミットしてから、この API を呼び出してください。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。この API を呼び出した後に SQL COMMIT ステートメントを発行することを推奨します。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaEnableTable(
      char *tableSpace,
      char *tableName
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに *longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・

タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** 指定された表スペースは、ユーザー表と同じノード・グループになければなりません。

**tableName (in)**
この音声列が入れられる表の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
表はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_TABLE_DOESNOT_EXIST**
表がありません。

**MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP**
指定された表スペースが、ユーザー表と同じノード・グループにありません。 **(EEE のみ)**

## 例

表スペース MYTS で、従業員表を音声 (DB2Audio データ) 用に使用可能にするには、次のようにします。索引表スペースおよび長形式表スペースには省略時値を使用します。

```
#include <dmbaudio.h>
rc = DBaEnableTable("myts,,",
       "employee");
```

音声 (DB2Audio データ) の従業員表を使用可能にします。省略時表スペースを使用します。

```
#include <dmbaudio.h>
rc = DBaEnableTable(NULL,
       "employee");
```

## DBaGetError

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

最後のエラーの説明を戻します。他の API が何らかのエラー・コードを戻してから、この API を呼び出してください。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

### パラメーター

**sqlcode (out)**
総称 SQL エラー・コード。

**errorMsgText (out)**
SQL エラー・メッセージ・テキスト。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

### 例

最後のエラーを獲得し、SQL エラー・コードを errCode、メッセージ・テキストを errMsg に保管するには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaGetError(&errCode, &errMsg);
```

## DBaGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

ユーザー表の音声列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な音声列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、アクセス不能ファイルへの参照を探します。ヌル値を指定すると、この修飾子が指定されているすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表の音声列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
long idx;
#include <dmbaudio.h>
rc = DBaGetInaccessibleFiles("employee",
       &count, &filelist);
```

## DBaGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

ユーザー表の音声列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な音声列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名が指定されている場

合、その表が探索され、ファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID データベースが所有するすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表の音声列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbaudio.h>
long idx;
rc = DBaGetReferencedFiles("employee",
       &count, &filelist);
```

## DBaIsColumnEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

列が音声 (DB2Audio データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

ユーザー表に対する SYSADM、DBADM、表所有者、または SELECT 特権

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

### パラメーター

**tableName (in)**
修飾子付きまたは修飾子なしの表名。

**colName (in)**
列の名前。

**status (out)**
列が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

**-1** MMDB_INVALID_DATATYPE

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のサウンド列が音声用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaIsColumnEnabled("employee",
       "sound", &status);
```

## DBaIsDatabaseEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

データベースが音声 (DB2Audio データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaIsDatabaseEnabled(
      short *status
      );
```

### パラメーター

**status (out)**

データベースが使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

### エラー・コード

**MMDB_SUCCESS**

正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

personnl データベースが音声用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaIsDatabaseEnabled(&status);
```

## DBaIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

指定されているファイルを参照する表項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な音声列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、指定したファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID が所有するすべての表が探索されます。

**fileName (in)**
参照されるファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ファイル /audios/ajones.wav を参照する従業員表の音声列の項目のリストを表示するには、次のようにします。

```
#include <dmbaudio.h>
long idx;
rc = DBaIsFileReferenced(NULL,
       "/audios/ajones.wav",
       &count, &tableList);
```

## DBaIsTableEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

表が音声 (DB2Audio データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaIsTableEnabled(
      char *tableName,
      short *status
      );
```

### パラメーター

**tableName (in)**
表の名前。

**status (out)**
表が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

**-1** MMDB_INVALID_DATATYPE

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表が音声 (DB2Audio データ) 用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaIsTableEnabled("employee", &status);
```

## DBaPlay

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

クライアントでオーディオ・プレーヤーをオープンし、音声クリップを再生します。このクリップは、音声列や外部ファイルに保管できます。

- 音声クリップを外部ファイルに保管する場合は、ファイル名か音声ハンドルのどちらかをこの API に渡すことができます。この API は、クライアント環境変数 DB2AUDIOPATH を使用してファイル位置を解決します。そのファイルは、クライアント・ワークステーションからアクセス可能でなければなりません。
- 音声クリップを列に保管する場合は、音声ハンドルをこの API に渡す必要があります。アプリケーションは、データベースに接続されている必要があり、オーディオ・クリップが保管されている表への読み取りアクセスを持っている必要があります。

音声が列に保管されている場合、エクステンダーは一時ファイルを作成して、列からそのファイルにオブジェクトの内容を複写します。音声が外部ファイルに保管されており、その相対ファイル名が環境変数内の値を使用して解決できない場合、またはそのファイルにクライアント・マシン上でアクセスできない場合にも、エクステンダーは一時ファイルを作成します。この一時ファイルは、DB2AUDIOTEMP 環境変数で指定されたディレクトリーに作成されます。エクステンダーは、後にこの一時ファイルから音声を再生します。

### 許可

音声クリップを列から再生する場合は、ユーザー表に対する Select 権限。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

## 構文

列に保管されている音声を再生する

```
long DBaPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Audio *audioHandle,
      waitFlag
      );
```

## 構文

ファイルとして保管されている音声を再生する

```
long DBaPlay(
      char *playerName,
      MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

## パラメーター

**playerName (in)**
オーディオ・プレーヤーの名前。 NULL に設定すると、DB2AUDIOPLAYER 環境変数で指定されている省略時のオーディオ・プレーヤーが使用されます。

**MMDB_PLAY_HANDLE (in)**
音声が BLOB として保管されていることを示す定数。

**MMDB_PLAY_FILE (in)**
音声がクライアントからアクセスできるファイルとして保管されていることを示す定数。

**audioHandle (in)**
音声のハンドル。列にある音声クリップを再生する場合は、このパラメーターを渡す必要があります。音声ハンドルが外部ファイルを表す場合は、クライアント環境変数 DB2VIDEOPATH を使用してファイル位置を解決します。

**fileName (in)**
この音声が入っているファイルの名前。

**waitFlag (in)**
続行する前に、ユーザーがプレーヤーをクローズするまでプログラムに待機させるかどうかを示す定数。 MMDB_PLAY_WAIT は、アプリケ

ーションと同じスレッドでプレーヤーを実行します。
MMDB_PLAY_NO_WAIT は別々のスレッドでプレーヤーを実行します。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

audioHandle が識別する音声を再生します。アプリケーションと同じスレッドで省略時プレーヤーを実行するには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaPlay(NULL, MMDB_PLAY_HANDLE,
       audioHandle, MMDB_PLAY_WAIT);
```

## DBaPrepareAttrs

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

ユーザー提供の音声属性を作成します。この API は、ユーザー提供の属性を持つ音声オブジェクトの保管時または更新時に使用します。サーバーで実行する UDF コードは常に、たいていの UNIX プラットフォームで使用される『ビッグ・エンディアン』 形式のデータを期待します。音声オブジェクトを『リトル・エンディアン』 形式で (つまり非 UNIX クライアントから) 保管または更新する場合、保管要求または更新要求の前に DBaPrepare API を使用する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudio.lib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
void DBaPrepareAttrs(
      MMDBAudioAttrs *audAttr
      );
```

### パラメーター

**audAttr (in)**
音声のユーザー提供の属性。

### 例

ユーザー提供の音声属性を作成するには、次のようにします。

```
#include <dmbaudio.h>
DBaPrepareAttrs(&imgattr);
```

## DBaReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | O | |

音声関連のメタデータ表を 『クリーンアップ』 します。以下に例を示します。

- 音声メタデータ表で使用されなくなったスペースを再利用します。
- 音声メタデータ表にある、存在しなくなったオーディオ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Alter、Control、SYSADM、SYSCTRL、SYSMAINT、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbaudiolib | libdmbaudio.a (AIX)<br>libdmbaudio.sl (HP-UX)<br>libdmbaudio.so (Solaris) |

### インクルード・ファイル

dmbaudio.h

### 構文

```
long DBaReorgMetadata(
      char *tableName
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、指定されたユーザー表と関連する音声メタデータ表のクリーンアップが実行されます。ヌル値を指定すると、現行ユーザー ID が所有するすべての表の中の音声列のメタデータ表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表の音声列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbaudio.h>
rc = DBaReorgMetadata("employee");
```

## DBiAdminGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ユーザー表の画像列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ユーザー ID rjones が所有する表の画像列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiAdminGetInaccessibleFiles
       ("rjones", &count, &filelist);
```

## DBiAdminGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ユーザー表の画像列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この

修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ajones が所有する表の画像列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiAdminGetReferencedFiles("ajones",
       &count, &filelist);
```

## DBiAdminIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

指定されているファイルを参照する、ユーザー表の画像列項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**fileName (in)**
参照されるファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ファイル /images/asmith.bmp を参照する、現行データベースのすべての表の画像列の項目のリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiAdminIsFileReferenced(NULL,
       "/images/asmith.bmp",
       &count, &tableList);
```

## DBiAdminReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像関連のメタデータ表を 『クリーンアップ』 します。

- 画像メタデータ表で使用されなくなったスペースを再利用します。
- 画像メタデータ表にある、存在しなくなったイメージ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiAdminReorgMetadata(
      char *qualifier
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表がクリーンアップされます。ヌル値を指定すると、現行データベースのすべての表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ユーザー ID rsmith が所有する表の画像列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiAdminReorgMetadata("rsmith");
```

## DBiBrowse

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

クライアントで画像ブラウザーをオープンし、画像を表示します。この画像は、画像列や外部ファイルに保管できます。

- 画像を外部ファイルに保管する場合は、ファイル名か画像ハンドルのどちらかをこの API に渡すことができます。 API は、クライアント環境変数 DB2IMAGEPATH を使用してファイル位置を解決します。ファイルは、クライアント・ワークステーションからアクセス可能です。
- 画像を列に保管する場合は、画像ハンドルをこの API に渡す必要があります。アプリケーションは、データベースに接続されている必要があり、画像が保管されている表への読み取りアクセスを持っている必要があります。

ブラウザーから直接画像にアクセスできない場合、エクステンダーは DB2IMAGETEMP 環境変数に指定されているディレクトリーに一時ファイルを作成します。次いでエクステンダーは、一時ファイルから画像を表示します。

### 許可

画像を列からブラウズする場合は、ユーザー表に対する Select 権限。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

#### 列に保管されている画像をブラウズする

```
long DBiBrowse(
      char *browserName,
      MMDB_PLAY_HANDLE,
      DB2Image *imageHandle,
      waitFlag
      );
```

## 構文

ファイルとして保管されている画像をブラウズする

```
long DBiBrowse(
      char *browserName,
     MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

## パラメーター

**browserName (in)**
画像ブラウザーの名前。 NULL に設定すると、DB2IMAGEBROWSER 環境変数で指定されている省略時の画像ブラウザーが使用されます。

**MMDB_PLAY_HANDLE (in)**
画像が BLOB として保管されていることを示す定数。

**MMDB_PLAY_FILE (in)**
画像がクライアントからアクセスできるファイルとして保管されていることを示す定数。

**imageHandle (in)**
画像のハンドル。列にある画像をブラウズする場合は、このパラメーターを渡す必要があります。画像ハンドルが外部ファイルを表す場合は、クライアント環境変数 DB2IMAGEPATH を使用してファイル位置を解決します。

**fileName (in)**
この画像が入っているファイルの名前。

**waitFlag (in)**
続行する前に、ユーザーがブラウザーをクローズするまでプログラムに待機させるかどうかを示す定数。 MMDB_PLAY_WAIT は、アプリケーションと同じスレッドでブラウザーを実行します。
MMDB_PLAY_NO_WAIT は別々のスレッドでブラウザーを実行します。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

imageHandle で識別される画像を表示します。アプリケーションと同じスレッドで省略時ブラウザーを実行するには、次のようにします。

```
#include <dmbimage.h>
rc = DBiBrowse(NULL, MMDB_PLAY_HANDLE,
       imageHandle, MMDB_PLAY_WAIT);
```

## DBiDisableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

列を画像 (DB2Image データ) の場合に使用不可にして、イメージ・データを保持できないようにします。列項目の内容は NULL に設定され、この列に関連するメタデータは消去されます。この列に関連する QBIC カタログも削除されます。この列の、イメージ・エクステンダーで定義したすべてのトリガーも消去されます。使用不可にした列が入っている表に新しい行が挿入され、それらの新しい行に DB2Image タイプを使用して定義したデータが入っていることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiDisableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
この画像列が入っている表の名前。

**colName (in)**
画像列の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のピクチャー列を画像 (DB2Image データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiDisableColumn("employee",
       "picture");
```

## DBiDisableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

データベースを画像 (DB2Image データ) の場合に使用不可にして、イメージ・データを保持できないようにします。また、DB2Image に定義されているデータベースのすべての表も使用不可にします。このデータベースの、イメージ・エクステンダーで定義したメタデータおよび UDF は消去されます。
DB2Image タイプを使用して定義されている、データベースに入っている表に新しい行が挿入されることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。

### 許可

DBADM、SYSADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiDisableDatabase(
      );
```

### パラメーター

DBiDisableDatabase にはパラメーターがありません。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

現行データベースを画像 (DB2Image データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiDisableDatabase();
```

## DBiDisableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

表を画像 (DB2Image データ) の場合に使用不可にして、イメージ・データを保持できないようにします。また、DB2Image に定義されている表のすべての列も使用不可にします。この表の、イメージ・エクステンダーで定義した一部のメタデータは消去されます。この表の画像列に関連するすべての QBIC カタログも削除されます。 DB2Image タイプを使用して定義されている表に新しい行が挿入されることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiDisableTable(
      char *tableName
      );
```

### パラメーター

**tableName (in)**
画像列が入っている表の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表を画像 (DB2Image データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiDisableTable("employee");
```

## DBiEnableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

列を画像 (DB2Image データ) 用に使用可能にします。この API は、この列とメタデータ表との間の関係を定義して管理します。この API を呼び出す前に、アプリケーションをデータベースに接続してユーザー表をコミットする必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiEnableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
この画像列が入っている表の名前。

**colName (in)**
画像列の名前。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
列はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_WRONG_SIGNATURE**
指定された列のデータ・タイプが正しくありません。ユーザー定義のタイプ MMDBSYS.DB2IMAGE が期待されます。

**MMDB_RC_COLUMN_DOESNOT_EXIST**
指定された表に列が定義されていません。

**MMDB_RC_NOT_ENABLED**
データベースまたは表が使用可能になっていません。

## 例

従業員表のピクチャー列を画像用に使用可能にするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiEnableColumn("employee",
       "picture");
```

## DBiEnableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

データベースを画像 (DB2Image データ) 用に使用可能にします。この API は、データベースごとに 1 回呼び出します。 DB2 ユーザー定義タイプ DB2Image をデータベース・マネージャーに定義します。また、DB2Image データを処理するすべての UDF も作成します。

### 許可

DBADM、SYSADM、SYSCTRL

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiEnableDatabase(
      char *tableSpace
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに、*longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** エクステンダー用のデータベースを使用可能にしたときに指定した表スペースは、区分データベース・システムのすべてのノードを含むノードグループで定義する必要があります。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
このデータベースはすでに使用可能になっています。

**MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER**
接続先のサーバーがこのコマンドをサポートしていません。

**MMDB_WARN_NOT_ALL_NODES**
指定された表スペースには、エクステンダー用のすべてのノードが含まれていません。 **(EEE のみ)**

**MMDB_RC_NOT_SAME_NODEGROUP**
指定された表スペースが同じノード・グループにありません。 **(EEE のみ)**

## 例

MYTS という名前の表スペースで、現行データベースを画像 (DB2Image データ) 用に使用可能にします。索引表スペースおよび長形式表スペースに省略時値を使用します。

```
#include <dmbimage.h>
rc = DBiEnableDatabase("myts,,");
```

データベースを画像 (DB2Image データ) 用に使用可能にします。省略時表スペースを使用します。

```
#include <dmbimage.h>
rc = DBiEnableDatabase(NULL);
```

## DBiEnableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

表を画像 (DB2Image データ) 用に使用可能にします。この API は、表ごとに 1 回呼び出します。表の画像列属性を保管し、管理するためのメタデータ表を作成します。ロッキングが発生する可能性を回避するため、アプリケーションでトランザクションをコミットしてから、この API を呼び出してください。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiEnableTable(
      char *tableSpace,
      char *tableName
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに *longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・

タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** 指定された表スペースは、ユーザー表と同じノード・グループになければなりません。

**tableName (in)**
画像列が入っている表の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
表はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_TABLE_DOESNOT_EXIST**
表がありません。

**MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP**
指定された表スペースが、ユーザー表と同じノード・グループにありません。 **(EEE のみ)**

## 例

表スペース MYTS で、従業員表を画像 (DB2Image データ) 用に使用可能にするには、次のようにします。索引表スペースおよび長形式表スペースには省略時値を使用します。

```
#include <dmbimage.h>
rc = DBiEnableTable("myts,,",
       "employee");
```

従業員表を画像 (DB2Image データ) の場合に使用可能にするには、次のようにします。省略時表スペースを使用します。

## DBiEnableTable

```
#include <dmbimage.h>
rc = DBiEnableTable(NULL,
       "employee");
```

## DBiGetError

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

最後のエラーの説明を戻します。他の API が何らかのエラー・コードを戻してから、この API を呼び出してください。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

### パラメーター

**sqlcode (out)**
総称 SQL エラー・コード。

**errorMsgText (out)**
SQL エラー・メッセージ・テキスト。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

### 例

最後のエラーを獲得し、SQL エラー・コードを errCode、メッセージ・テキストを errMsg に保管するには、次のようにします。

```
#include <dmbimage.h>
rc = DBiGetError(&errCode, &errMsg);
```

## DBiGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ユーザー表の画像列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な画像列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、アクセス不能ファイルへの参照を探します。ヌル値を指定すると、この修飾子が指定されているすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表の画像列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiGetInaccessibleFiles("employee",
       &count, &filelist);
```

## DBiGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ユーザー表の画像列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な画像列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名が指定されている場

合、その表が探索され、ファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID が所有するすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表の画像列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiGetReferencedFiles("employee",
       &count, &filelist);
```

## DBiIsColumnEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

列が画像 (DB2Image データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

ユーザー表に対する SYSADM、DBADM、表所有者、または SELECT 特権

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

### パラメーター

**tableName (in)**
修飾子付きまたは修飾子なしの表名。

**colName (in)**
列の名前。

**status (out)**
列が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

**-1** MMDB_INVALID_DATATYPE

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
列はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のピクチャー列が画像用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbimage.h>
rc = DBiIsColumnEnabled("employee",
       "picture", &status);
```

## DBiIsDatabaseEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

データベースが画像 (DB2Image データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiIsDatabaseEnabled(
      short *status
      );
```

### パラメーター

**status (out)**

データベースが使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

### エラー・コード

**MMDB_SUCCESS**

正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

personnl データベースが画像用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbimage.h>
rc = DBiIsDatabaseEnabled(&status);
```

## DBiIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

指定されているファイルを参照する、画像列の中の表項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能な画像列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、指定したファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID が所有するすべての表が探索されます。

**fileName (in)**
参照されるファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ファイル /images/ajones.bmp を参照する従業員表の画像列の項目のリストを表示するには、次のようにします。

```
#include <dmbimage.h>
long idx;
rc = DBiIsFileReferenced(NULL,
       "/images/ajones.bmp",
       &count, &tableList);
```

## DBiIsTableEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

表が画像 (DB2Image データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiIsTableEnabled(
      char *tableName,
      short *status
      );
```

### パラメーター

**tableName (in)**
表の名前。

**status (out)**
表が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表が画像用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbimage.h>
rc = DBiIsTableEnabled("employee",
       &status);
```

## DBiPrepareAttrs

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ユーザー提供の画像属性を作成します。この API は、ユーザー提供の属性を持つ画像オブジェクトの保管時または更新時に使用します。サーバーで実行する UDF コードは常に、たいていの UNIX プラットフォームで使用される『ビッグ・エンディアン』 形式のデータを期待します。画像オブジェクトを『リトル・エンディアン』 形式で (つまり非 UNIX クライアントから) 保管または更新する場合、保管要求または更新要求の前に DBiPrepare API を使用する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
void DBiPrepareAttrs(
      MMDBImageAttrs *imgAttr
      );
```

### パラメーター

**imgAttr (in)**
画像のユーザー提供の属性。

### 例

ユーザー提供の画像の属性を作成するには、次のようにします。

```
#include <dmbimage.h>
DBiPrepareAttrs(&imgattr);
```

## DBiReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

たとえば、次の方法で画像関連のメタデータ表を 『クリーンアップ』 します。

- 画像メタデータ表で使用されなくなったスペースを再利用します。
- 画像メタデータ表にある、存在しなくなったイメージ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Alter、Control、SYSADM、SYSCTRL、SYSMAINT、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbimage.lib | libdmbimage.a (AIX)<br>libdmbimage.sl (HP-UX)<br>libdmbimage.so (Solaris) |

### インクルード・ファイル

dmbimage.h

### 構文

```
long DBiReorgMetadata(
      char *tableName
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、指定されたユーザー表と関連する画像メタデータ表のクリーンアップが実行されます。ヌル値を指定すると、現行ユーザー ID が所有するすべての表の中の画像列のメタデータ表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表の画像列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbimage.h>
rc = DBiReorgMetadata("employee");
```

## DBvAdminGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ユーザー表のビデオ列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。 (in) ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ユーザー ID rsmith が所有する表のビデオ列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
#include <dmbvideo.h>
long idx;
rc = DBvAdminGetInaccessibleFiles
       ("rsmith", &count,
       &filelist);
```

## DBvAdminGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ユーザー表のビデオ列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この

修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ajones が所有する表のビデオ列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbvideo.h>
long idx;
rc = DBvAdminGetReferencedFiles
       ("ajones", &count,
       &filelist);
```

## DBvAdminIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

指定されているファイルを参照する、ユーザー表のビデオ列項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表が探索されます。ヌル値を指定すると、現行データベースのすべての表が探索されます。

**fileName (in)**
参照されるファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ファイル /videos/asmith.mpg を参照する、現行データベースのすべての表のビデオ列の項目のリストを表示するには、次のようにします。

```
#include <dmbvideo.h>
long idx;
rc = DBvAdminIsFileReferenced(NULL,
       "/videos/asmith.mpg",
       &count, &tableList);
```

## DBvAdminReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

たとえば次の方法で、ビデオ関連のメタデータ表を 『クリーンアップ』 します。

- ビデオ・メタデータ表で使用されなくなったスペースを再利用します。
- ビデオ・メタデータ表にある、存在しなくなったビデオ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

SYSADM、SYSCTRL、SYSMAINT

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvAdminReorgMetadata(
      char *qualifier
      );
```

### パラメーター

**qualifier (in)**

有効なユーザー ID またはヌル値。ユーザー ID を指定すると、この修飾子が指定されているすべての表がクリーンアップされます。ヌル値を指定すると、現行データベースのすべての表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_NO_AUTH**
この API を呼び出すためのユーザー権限が正しくありません。

## 例

ユーザー ID rsmith が所有する表のビデオ列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvAdminReorgMetadata("rsmith");
```

## DBvBuildStoryboardFile

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショット・カタログ・ファイルを作成して、ビデオ内のすべてのショットに関する項目をこのファイル内に作成します。ソース・ビデオはデータベースまたはファイルに置くことができます。 API はショットごとに、ショット番号、開始フレーム番号、終了フレーム番号、および少なくとも 1 つの代表フレームに関する情報を保管します。 DBvStoryboardCtrl データ構造内の値は、 1 つのショットにいくつの代表フレーム数が指定されているかを示します。長さが DBvStoryboardCtrl のしきい値より短いショットの場合、この API は 1 つの代表フレームを示します。長さが DBvStoryboardCtrl の下位しきい値と上位しきい値の間であるショットの場合、この API は 2 つの代表フレームを示します。長さが DBvStoryboardCtrl の上位しきい値より長いショットの場合、この API は 3 つの代表フレームを示します。代表フレーム情報には、フレーム番号、およびフレーム内容が含まれているファイルの名前が含まれています。この情報はストーリーボード、すなわち、ビデオの眼に見える要約を表示するために使用することができます。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvBuildStoryboardFile(
      char *fileName,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl
            );
```

## パラメーター

**catalogName (in)**
ショット・カタログ・ファイルの名前を指すポインター。

**video (in)**
ビデオ構造を指すポインター。

**shotCtrl (in)**
ショット制御構造を指すポインター。

**sbCtrl (in)**
ストーリーボード制御構造を指すポインター。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

hotshots という名前のカタログ・ファイルを作成して、ビデオ内のすべてのショットに関するデータをこのファイルに入れます。

```
#include <dmbshot.h>
rc = DBvBuildStoryboardFile("hotshots",
       video, &shotCtrl, &sbCtrl);
```

## DBvBuildStoryboardTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオ内のすべてのショットに関する項目をショット・カタログに作成します。ソース・ビデオはデータベースまたはファイルに置くことができます。ショット・カタログはデータベースにあります。ショットごとに、API はソース・ビデオに関するハンドル情報またはファイル情報を保管します。さらに、ショット番号、開始フレーム番号、終了フレーム番号、および少なくとも 1 つの代表フレームに関する情報も保管します。 DBvStoryboardCtrl データ構造内の値は、1 つのショットにいくつの代表フレーム数が指定されているかを示します。長さが DBvStoryboardCtrl のしきい値より短いショットの場合、この API は 1 つの代表フレームを示します。長さが DBvStoryboardCtrl の下位しきい値と上位しきい値の間であるショットの場合、この API は 2 つの代表フレームを示します。長さが DBvStoryboardCtrl の上位しきい値より長いショットの場合、この API は 3 つの代表フレームを示します。代表フレーム情報には、フレーム番号およびフレーム・データが含まれます。ショット・カタログに保管される代表フレーム情報は、ストーリーボード、すなわち、ビデオの眼に見える要約を表示するために使用することができます。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

## 構文

```
long DBvBuildStoryboardTable(
      char *catalogName ,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl,
      SQLHDBC hdbc
      );
```

## パラメーター

**catalogName (in)**
ショット・カタログの名前を指すポインター。

**video (in)**
ビデオ構造を指すポインター。

**shotCtrl (in)**
ショット制御構造を指すポインター。

**sbCtrl (in)**
ストーリーボード制御構造を指すポインター。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

ビデオに hotshots という名前を付けたショット・カタログの項目を作成します。

```
#include <dmbshot.h>
rc = DBvBuildStoryboardTable("hotshots",
       video, &shotCtrl, &sbCtrl, hdbc);
```

## DBvClose

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

シーン変更を検出するためにオープンしたビデオ・ファイルをクローズします。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvClose(
      DB2vIOType *video
      );
```

### パラメーター

**video (in)**
ビデオ構造を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_CANNOT_CLOSE**
ビデオ・ファイルをクローズできませんでした。

## 例

ビデオ・シーンの変更を検出するために直前にオープンしたビデオ・ファイルをクローズするには、次のようにします。

```
#include <dmbshot.h>
rc = DBvClose(video);
```

## DBvCreateIndex

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ファイルに保管されているビデオに関する索引を作成します。索引はビデオ・エクステンダーがビデオ内のショットとフレームにアクセスするときに使用されます。この索引は、ソース・ビデオ・ファイルと同じディレクトリーのフラット・ファイルに保管されます。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvCreateIndex(
      char *fileName
      );
```

### パラメーター

**fileName (in)**
ビデオ・ファイル名を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_OPEN_VIDEO**
ビデオ・ファイルをオープンして処理できませんでした。

**MMDB_RC_INDEX_FAIL**
索引を作成できませんでした。

## 例

ファイル ¥videos¥ajones.mpg 内のビデオの索引を作成します。

```
#include <dmbshot.h>
rc = DBvCreateIndex("¥videos¥ajones.mpg");
```

## DBvCreateIndexFromVideo

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオの索引を作成します。ビデオはショット検出のために最初にオープンされます。索引はビデオ・エクステンダーがビデオ内のショットとフレームにアクセスするときに使用されます。索引はフラット・ファイルに保管されます。ファイル名は DBvIOType データ構造に保管されます。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvCreateIndexFromVideo(
      DBvIOType *video
      );
```

### パラメーター

**video (update)**
ビデオ構造を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_OPEN_VIDEO**
ビデオ・ファイルをオープンして処理できませんでした。

**MMDB_RC_INDEX_FAIL**
索引を作成できませんでした。

## 例

ビデオの索引を作成します。

```
#include <dmbshot.h>
rc = DBvCreateIndexFromVideo(video);
```

## DBvCreateShotCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショット・カタログ (フレーム番号などのショットに関する情報を入れる表のセット) を作成します。

このアプリケーションは、db2video と db2image の両方で使用できるようになっているデータベースに接続する必要があります。

### 許可

Create、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvCreateShotCatalog(
      char *catalogName ,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
作成するショット・カタログの名前。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

hotshots という名前のショット・カタログを作成するには、次のようにします。

```
#include <dmbshot.h>
rc = DBvCreateShotCatalog("hotshots", hdbc);
```

## DBvDeleteShot

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

カタログからショットを削除します。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvDeleteShot(
      char *catalogName ,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
カタログの名前。

**shotHandle (in)**
ショット・ハンドル。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_ACCESS**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

あるショットのハンドルを使用して、hotshots カタログからこのショットを削除するには、次のようにします。

```
#include <dmbshot.h>
rc = DBvDeleteShot("hotshots", shot,
       hdbc);
```

## DBvDeleteShotCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショット・カタログを削除します。

### 許可

Control、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvDeleteShotCatalog(
      char *catalogName ,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
削除するショット・カタログの名前。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_ACCESS**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

hotshots ショット・カタログを削除するには、次のようにします。

## 例

```
#include <dmbshot.h>
rc = DBvDeleteShotCatalog("hotshots",
       hdbc);
```

## DBvDetectShot

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオ・ファイル内の次のショットを探索します。ショットを検出したら、検出したショットの最初のフレームのフレーム番号とフレーム・データを記録します。ショットが検出されたかどうか判別するには、shotDetected フラグを検査する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvDetectShot(
      DBvIOType *video,
      unsigned long *start_frame,
      char *shotDetected,
      DBvShotControl *shotCtrl,
      DBvShotType *shot,
            );
```

### パラメーター

**video (update)**
ビデオ構造を指すポインター。

**start_frame (in/out)**
探索の開始点として使用されるフレーム番号。戻り時にパラメーターが更新されて、次のショットの探索の開始点を示します。

**shotDetected (out)**
ショット検出フラグ: 1= フレームが検出された、0= フレームは検出されなかった。

**shotCtrl (in)**
ショット制御データを指すポインター。

**shot (out)**
検出されたショットとショット・データを指すポインター。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_EOF**
ファイルの終わり。

**MMDB_NO_INDEX**
ビデオ索引がありません。

## 例

フレーム 1 から開始する、ビデオ・ファイル内の次のショットを探索します。

```
#include <dmbshot.h>
long start_frame=1;
rc = DBvDetectShot(video, start_frame&Detected,
       &shotCtrl, &shot);
```

## DBvDisableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

列をビデオ (DB2Video データ) の場合に使用不可にして、ビデオ・データを保持できないようにします。列項目の内容は NULL に設定され、この列に関連するメタデータは消去されます。この列の、ビデオ・エクステンダーで定義したすべてのトリガーも消去されます。使用不可にした列が入っている表に新しい行が挿入され、それらの新しい行に DB2Video タイプを使用して定義したデータが入っていることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvDisableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
このビデオ列が入っている表の名前。

**colName (in)**
ビデオ列の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表の tv_ads 列をビデオ (DB2Video データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvDisableColumn("employee",
       "tv_ads");
```

## DBvDisableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

データベースをビデオ (DB2Video データ) の場合に使用不可にして、ビデオ・データを保持できないようにします。また、DB2Video に定義されているデータベースのすべての表も使用不可にします。このデータベースの、ビデオ・エクステンダーで定義したメタデータおよび UDF は消去されます。 DB2Video タイプを使用して定義されている、データベースに入っている表に新しい行が挿入されることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。

### 許可

DBADM、SYSADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvDisableDatabase(
      );
```

### パラメーター

DBvDisableDatabase にはパラメーターがありません。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

現行データベースをビデオ (DB2Video データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvDisableDatabase();
```

## DBvDisableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

表をビデオ (DB2Video データ) の場合に使用不可にして、ビデオ・データを保持できないようにします。また、DB2Video に定義されている表のすべての列も使用不可にします。この表の、ビデオ・エクステンダーで定義した一部のメタデータは消去されます。 DB2Video タイプを使用して定義されている表に新しい行が挿入されることもありますが、これらの新しい行に関連するメタデータは管理サポート表にありません。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvDisableTable(
      char *tableName
      );
```

### パラメーター

**tableName (in)**
ビデオ列が入っている表の名前。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表をビデオ (DB2Video データ) の場合に使用不可にするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvDisableTable("employee");
```

## DBvEnableColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

列をビデオ (DB2Video データ) 用に使用可能にします。この API は、この列とメタデータ表との間の関係を定義して管理します。この API を呼び出す前に、アプリケーションをデータベースに接続してユーザー表をコミットする必要があります。

### 許可

Control、Alter、SYSADM、DBADM

API パラメーターで指定された表スペースおよびバッファー・プールに対する使用特権も必要です。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvEnableColumn(
      char *tableName,
      char *colName,
      );
```

### パラメーター

**tableName (in)**
このビデオ列が入っている表の名前。

**colName (in)**
ビデオ列の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
列はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_WRONG_SIGNATURE**
指定された列のデータ・タイプが正しくありません。ユーザー定義のデータ・タイプ MMDBSYS.DB2VIDEO が期待されます。

**MMDB_RC_COLUMN_DOESNOT_EXIST**
指定された表に列が定義されていません。

**MMDB_RC_NOT_ENABLED**
データベースまたは表が使用可能になっていません。

## 例

従業員表のビデオ列をビデオ用に使用可能にするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvEnableColumn("employee",
       "video");
```

## DBvEnableDatabase

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

データベースをビデオ (DB2Video データ) 用に使用可能にします。この API は、データベースごとに 1 回呼び出します。 DB2 ユーザー定義タイプ DB2Video をデータベース・マネージャーに定義します。また、DB2Video データを処理するすべての UDF も作成します。

### 許可

DBADM、SYSADM、SYSCTRL

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvEnableDatabase(
      char *tableSpace
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに、*longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** エクステンダー用のデータベースを使用可能にしたときに指定した表スペースは、区分データベース・システムのすべてのノードを含むノードグループで定義する必要があります。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
このデータベースはすでに使用可能になっています。

**MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER**
接続先のサーバーがこのコマンドをサポートしていません。

**MMDB_WARN_NOT_ALL_NODES**
指定された表スペースには、エクステンダー用のすべてのノードが含まれていません。 **(EEE のみ)**

**MMDB_RC_NOT_SAME_NODEGROUP**
指定された表スペースが同じノード・グループにありません。 **(EEE のみ)**

## 例

MYTS という名前の表スペース内で、現行データベースをビデオ (DB2Video データ) 用に使用可能にします。索引表スペースおよび長形式表スペースに省略時値を使用します。

```
#include <dmbvideo.h>
rc = DBvEnableDatabase("myts,,");
```

現行データベースをビデオ (DB2Video データ) 用に使用可能にします。省略時表スペースを使用します。

```
#include <dmbvideo.h>
rc = DBvEnableDatabase(NULL);
```

## DBvEnableTable

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

表をビデオ (DB2Video データ) 用に使用可能にします。この API は、表ごとに 1 回呼び出します。表のビデオ列属性を保管し、管理するためのメタデータ表を作成します。ロッキングが発生する可能性を回避するため、アプリケーションでトランザクションをコミットしてから、この API を呼び出してください。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Control、Alter、SYSADM、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvEnableTable(
      char *tableSpace,
      char *tableName
      );
```

### パラメーター

**tableSpace (in)**

管理表を保管する先のコンテナーの集まりである表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、さらに *longts* はメタデータ表の長い列 (たとえば、LONG VARCHAR および LOB データ・

タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** 指定された表スペースは、ユーザー表と同じノード・グループになければなりません。

**tableName (in)**
ビデオ列が入っている表の名前。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_WARN_ALREADY_ENABLED**
表はすでに使用可能になっています。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_TABLE_DOESNOT_EXIST**
表がありません。

**MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP**
指定された表スペースが、ユーザー表と同じノード・グループにありません。 **(EEE のみ)**

## 例

従業員表をビデオ (DB2Video データ) の場合に使用可能にするには、次のようにします。索引表スペースおよび長形式表スペースには省略時値を使用します。

```
#include <dmbvideo.h>
rc = DBvEnableTable("myts,,",
       "employee");
```

従業員表をビデオ (DB2Video データ) の場合に使用可能にするには、次のようにします。省略時表スペースを使用します。

## DBvEnableTable

```
#include <dmbvideo.h>
rc = DBvEnableTable(NULL,
       "employee");
```

## DBvFrameDataTo24BitRGB

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオ・フレームを、MPEG などの YUV カラー値形式から、24 ビット RGB 形式に変換します。 API 呼び出しを出す前に、ターゲット・バッファーを割り振る必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvFrameDataTo24BitRGB(
      unsigned char *RGB,
      DBvFrameData *fd,
      unsigned long dx,
      unsigned long dy
      );
```

### パラメーター

**RGB (out)**
RGB バッファーを指すポインター。

**fd (in)** 変換するフレーム・データを指すポインター。

**dx (in)**
フレーム幅。

**dy (in)**
フレームの高さ。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

## 例

ビデオ・フレームを MPEG から 24 ビット RGB に変換するには、次のようにします。

```
#include <dmbshot.h>
rc = DBvFrameDataTo24BitRGB(RGB, &video->fd,
       video->dx, video->dy);
```

## DBvGetError

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

最後のエラーの説明を戻します。他の API が何らかのエラー・コードを戻してから、この API を呼び出してください。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

### パラメーター

**sqlcode (out)**
総称 SQL エラー・コード。

**errorMsgText (out)**
SQL エラー・メッセージ・テキスト。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

### 例

最後のエラーを獲得し、SQL エラー・コードを errCode、メッセージ・テキストを errMsg に保管するには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvGetError(&errCode, &errMsg);
```

## DBvGetFrame

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオ・ファイル内の現行フレームを入手します。フレーム・データは、DBvFrameData ビデオ構造で戻されます。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvGetFrame(
      DBvIOType *video
      );
```

### パラメーター

**video (update)**
ビデオ構造を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_EOF**
ファイルの終わり。

### 例

ビデオ・ファイル内の現行フレームを入手します。

## DBvGetFrame

```
#include <dmbshot.h>
rc = DBvGetFrame(video);
```

## DBvGetInaccessibleFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ユーザー表のビデオ列で参照されているアクセス不能なファイルの名前を戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能なビデオ列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、アクセス不能ファイルへの参照を探します。ヌル値を指定すると、この修飾子が指定されているすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているアクセス不能なファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表のビデオ列で参照されているすべてのアクセス不能ファイルのリストを表示するには、次のようにします。

```
#include <dmbvideo.h>
long idx;
rc = DBvGetInaccessibleFiles("employee",
       &count, &filelist);
```

## DBvGetReferencedFiles

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ユーザー表のビデオ列で参照されているファイルの名前を戻します。ファイルがアクセス不能の場合 (たとえば、環境変数の指定を使ってそのファイル名を解決できない場合)、そのファイル名の先頭にはアスタリスクが付きます。この API では FILEREF データ構造の FILENAME フィールドは使用しないので、このフィールドは NULL に設定されます。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造を解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能なビデオ列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名が指定されている場

合、その表が探索され、ファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID が所有するすべての表が探索されます。

**count (out)**
出力リストの項目数。

**fileList (out)**
表で参照されているファイルのリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

従業員表のビデオ列で参照されているすべてのファイルのリストを表示するには、次のようにします。

```
#include <dmbvideo.h>
long idx;
rc = DBvGetReferencedFiles("employee",
       &count, &filelist);
```

## DBvInitShotControl

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショット制御データ構造内の値を初期設定します。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvInitShotControl(
      DBvShotControl *shotCtrl,
      );
```

### パラメーター

**shotCtrl (in)**
ショット制御データ構造を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_ACCESS**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

### 例

ショット制御データ構造内の値を初期設定します。

```
#include <dmbshot.h>
rc = DBvInitShotControl(shotCtrl);
```

## DBvInitStoryboardCtrl

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ストーリーボード制御データ構造内の値を初期設定します。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvInitStoryboardCtrl(
      DBvStoryboardCtrl *sbCtrl,
      );
```

### パラメーター

**shotCtrl (in)**
ショット制御データ構造を指すポインター。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_ACCESS**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

### 例

ストーリーボード制御データ構造内の値を初期設定します。

```
#include <dmbshot.h>
rc = DBvInitStoryboardCtrl(shotCtrl);
```

## DBvInsertShot

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショットをショット・カタログに挿入します。

### 許可

Insert、Control

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvInsertShot(
      char *catalogName ,
      DBvShotType *shot,
      DBvIOType *video,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
カタログの名前。

**shot (in)**
カタログに挿入する拡張ショットを指すポインター。

**shotHandle (in)**
ショット・ハンドル。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_ACCESS**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

ショットをホット・ショットと呼ばれるショット・カタログに挿入します。

```
rc = DBvInsertShot(
"hotshots",                    /* shot catalog name */
shot,                          /* pointer to shot structure */
video,                         /* pointer to video structure */
shotHandle,                    /* pointer to shot handle */
hdbc);                         /* database connection handle */
```

## DBvIsColumnEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

列がビデオ (DB2Video データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

ユーザー表に対する SYSADM、DBADM、表所有者、または SELECT 特権

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

### パラメーター

**tableName (in)**
修飾子付きまたは修飾子なしの表名。

**colName (in)**
列の名前。

**status (out)**
列が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

**-1** MMDB_INVALID_DATATYPE

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のビデオ列がビデオ用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvIsColumnEnabled("employee",
       "video", &status);
```

## DBvIsDatabaseEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

データベースがビデオ (DB2Video データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvIsDatabaseEnabled(
      short *status
      );
```

### パラメーター

**status (out)**

データベースが使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

### エラー・コード

**MMDB_SUCCESS**

正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

personnl データベースがビデオ用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvIsDatabaseEnabled(&status);
```

## DBvIsFileReferenced

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

指定されているファイルを参照する、ビデオ列の中の表項目のリストを戻します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

呼び出し後、この API が割り振った資源を解放するのは重要なことです。具体的には、filelist データ構造と、ファイル・リストのそれぞれの項目のファイル名フィールドを解放する必要があります。

### 許可

SELECT 特権。探索されたすべてのユーザー表および関連する管理サポート表の使用可能なビデオ列に対するもの。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、その表が探索され、指定したファイルへの参照を探します。ヌル値を指定すると、現行ユーザー ID が所有するすべての表が探索されます。

**fileName (in)**
参照ファイルの名前。

**count (out)**
出力リストの項目数。

**tableList (out)**
指定されているファイルを参照する表項目のリスト。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_MALLOC**
システムは、結果を戻すメモリーを割り振ることができません。

## 例

ファイル /videos/ajones.mpg を参照する従業員表のビデオ列の項目のリストを表示するには、次のようにします

```
#include <dmbvideo.h>
long idx;
rc = DBvIsFileReferenced(NULL,
       "/videos/ajones.mpg",
       &count, &tableList);
```

## DBvIsIndex

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ビデオ索引の有無を検査します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvIsIndex(
      char *fileName,
      short *status
      );
```

### パラメーター

**fileName (in)**
参照ファイルの名前。

**status (out)**
索引があるかどうかを示します。値 1 は、索引があることを意味します。値 0 は、索引がないことを意味します。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_ERROR**
状況が有効ではありません。

## 例

ビデオ・ファイル ¥videos¥ajones.mpg の索引の存在をチェックします。

```
#include <dmbshot.h>
rc = DBvIsIndex("¥videos¥ajones.mpg", &status);
```

## DBvIsTableEnabled

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

表がビデオ (DB2Video データ) 用に使用可能になっているかどうかを判別します。この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvIsTableEnabled(
      char *tableName,
      short *status
      );
```

### パラメーター

**tableName (in)**
表の名前。

**status (out)**
表が使用可能になっているかどうかを示します。このパラメーターは、数値を戻します。また、エクステンダーは状況を示す定数も戻します。これらの値と定数を次に示します。

**1** MMDB_IS_ENABLED

**0** MMDB_IS_NOT_ENABLED

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表がビデオ用に使用可能になっているかどうかを判別するには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvIsTableEnabled("employee",
       &status);
```

## DBvMergeShots

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

2 つのショットを 1 つに組み合わせます。この結果作成されるショットは、先頭のショットのショット・ハンドルと開始フレームを使用します。結果のショットの終了フレームには、2 つのショットのうち大きい方のフレームが使用されます。 2 番目のショット・ハンドルが指している行は、削除されます。

### 許可

Control、Select、Delete、Update

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvMergeShots(
      char *catalogName ,
      char *shotHandle1,
      char *shotHandle2,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
ショット・カタログの名前。

**shotHandle1 (in)**
先頭のショットのハンドル。

**shotHandle2 (in)**
2 番目のショットのハンドル。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_CANNOT_MERGE**
ショットを組み合わせることができません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

ハンドルがそれぞれ shotHandle1 と shotHandle2 になっているショットを hotshots カタログで組み合わせるには、次のようにします。

```
#include <dmbshot.h>
rc = DBvMergeShots("hotshots", shotHandle1,
       shotHandle2, hdbc);
```

## DBvOpenFile

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

DBvIOType 構造のスペースを割り振り、ビデオ・ファイルをオープンしてピクセルをアクセスします。ビデオが正常にオープンした場合は、先頭のフレーム番号 (フレーム 0) を指した状態になっています。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvOpenFile(
      DBvIOType **video,
      char *fileName,
      );
```

### パラメーター

**video (out)**
ビデオ構造ポインターを指すポインター。

**fileName (in)**
オープンするビデオ・ファイルの名前。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_CANNOT_OPEN**
ビデオ・ファイルをオープンできませんでした。

**MMDB_RC_NO_MEMORY**
メモリーが不足しています。

**MMDB_RC_NO_INDEX**
ビデオのランダム・アクセス索引がありません。

## 例

ビデオ・ファイル ¥videos¥ajones.mpg をオープンするには、次のようにします。

```
#include <dmbshot.h>
rc = DBvOpenFile(&videoa,
       "¥videos¥ajones.mpg");
```

## DBvOpenHandle

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

DBvIOType 構造のスペースを割り振り、ビデオ・ハンドルをオープンしてピクセルをアクセスします。この構造は、先頭のフレーム番号 (フレーム 0) を指します。このビデオは、BLOB にすることができます。ビデオは、DB2VIDEOTEMP 環境変数で指定されているディレクトリーの一時ファイルに複写されます。 isIdx フラグは、ランダム・アクセス索引があるかどうかに基づいて設定されます。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvOpenHandle(
      DBvIOType **video,
      DB2Video *videoHandle
      SQLHDBC hdbc
      );
```

### パラメーター

**video (out)**
ビデオ構造を指すポインター。

**videoHandle (in)**
ビデオ・ハンドル。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_CANNOT_OPEN**
ビデオ・ファイルをオープンできませんでした。

**MMDB_RC_NO_MEMORY**
メモリーが不足しています。

**MMDB_RC_NO_INDEX**
ビデオのランダム・アクセス索引がありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_INVALID_HANDLE**
ビデオ・ハンドルが有効ではありません。

## 例

videoa ポインターを使用して videoHandle をオープンするには、次のようにします。

```
#include <dmbshot.h>
rc = DBvOpenHandle(&oa, videoHandle, hdbc);
```

## DBvPlay

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

クライアントでビデオ・プレーヤーをオープンし、ビデオを再生します。このビデオは、ビデオ列または外部ファイルに保管できます。

- ビデオを外部ファイルに保管すると、ファイル名かビデオ・ハンドルのどちらかをこの API に渡すことができます。 API は、クライアント環境変数 DB2VIDEOPATH を使用してファイル位置を解決します。ファイルは、クライアント・ワークステーションからアクセス可能です。
- ビデオを列に保管する場合は、ビデオ・ハンドルをこの API に渡す必要があります。アプリケーションは、データベースに接続されている必要があり、ビデオが保管されている表への読み取りアクセスを持っている必要があります。

ビデオが列に保管されている場合、エクステンダーは一時ファイルを作成して、列からそのファイルにオブジェクトの内容を複写します。ビデオが外部ファイルに保管されおり、その相対ファイル名が環境変数内の値を使用して解決できない場合、またはそのファイルにクライアント・マシン上でアクセスできない場合にも、エクステンダーは一時ファイルを作成します。この一時ファイルは、DB2VIDEOTEMP 環境変数で指定されたディレクトリーに作成されます。エクステンダーは、後にこの一時ファイルからビデオを再生します。

### 許可

ビデオを列から再生する場合は、ユーザー表に対する Select 権限。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

## 構文

列に保管されているビデオを再生する

```
long DBvPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Video *videoHandle,
      waitFlag
      );
```

## 構文

ファイルとして保管されているビデオを再生する

```
long DBvPlay(
      char *playerName,
      MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

## パラメーター

**playerName (in)**
ビデオ・プレーヤーの名前。 NULL に設定すると、DB2VIDEOPLAYER 環境変数で指定されている省略時のビデオ・プレーヤーが使用されます。

**MMDB_PLAY_HANDLE (in)**
ビデオが列に保管されていることを示す定数。

**MMDB_PLAY_FILE (in)**
ビデオがクライアントからアクセスできるファイルとして保管されていることを示す定数。

**videoHandle (in)**
ビデオのハンドル。列でビデオを再生する場合は、このパラメーターを渡す必要があります。ビデオ・ハンドルが外部ファイルを表す場合は、クライアント環境変数 DB2VIDEOPATH を使用してファイル位置を解決します。

**fileName (in)**
このビデオが入っているファイルの名前。 API は、クライアント環境変数 DB2VIDEOPATH を使用してファイル位置を解決します。ファイルは、クライアント・ワークステーションからアクセス可能です。

**waitFlag (in)**
続行する前に、ユーザーがプレーヤーをクローズするまでプログラムに待機させるかどうかを示す定数。 MMDB_PLAY_WAIT は、アプリケ

ーションと同じスレッドでプレーヤーを実行します。
MMDB_PLAY_NO_WAIT は別々のスレッドでプレーヤーを実行します。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

videoHandle で識別されるビデオを再生します。アプリケーションと同じスレッドで省略時プレーヤーを実行するには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvPlay(NULL, MMDB_PLAY_HANDLE, videoHandle,
       MMDB_PLAY_WAIT);
```

## DBvPrepareAttrs

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ユーザー提供のビデオ属性を作成します。この API は、ユーザー提供の属性を持つビデオ・オブジェクトの保管時または更新時に使用します。サーバーで実行する UDF コードは常に、たいていの UNIX プラットフォームで使用される 『ビッグ・エンディアン』 形式のデータを期待します。ビデオ・オブジェクトを 『リトル・エンディアン』 形式で (つまり非 UNIX クライアントから) 保管または更新する場合、保管要求または更新要求の前に DBvPrepare API を使用する必要があります。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
void DBvPrepareAttrs(
      MMDBVideoAttrs *vidAttr
      );
```

### パラメーター

**vidAttr (in)**
ビデオのユーザー提供の属性。

### 例

ユーザー提供のビデオ属性を作成するには、次のようにします。

```
#include <dmbvideo.h>
DBvPrepareAttrs(&vidattr);
```

## DBvReorgMetadata

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

たとえば次の方法で、ビデオ関連のメタデータ表を 『クリーンアップ』 します。

- ビデオ・メタデータ表で使用されなくなったスペースを再利用します。
- ビデオ・メタデータ表にある、存在しなくなったビデオ・ファイルへの参照を削除します。

この API を呼び出す前に、アプリケーションをデータベースに接続する必要があります。

### 許可

Alter、Control、SYSADM、SYSCTRL、SYSMAINT、DBADM

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbvideo.lib | libdmbvideo.a (AIX)<br>libdmbvideo.sl (HP-UX)<br>libdmbvideo.so (Solaris) |

### インクルード・ファイル

dmbvideo.h

### 構文

```
long DBvReorgMetadata(
      char *tableName,
      );
```

### パラメーター

**tableName (in)**

修飾子付き、修飾子なし、またはヌルの表名。表名を指定すると、指定されたユーザー表と関連するビデオ・メタデータ表のクリーンアップが実行されます。ヌル値を指定すると、現行ユーザー ID が所有するすべての表の中のビデオ列のメタデータ表がクリーンアップされます。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NO_AUTH**
呼び出し側のアクセス権限が正しくありません。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

## 例

従業員表のビデオ列のメタデータ表をクリーンアップするには、次のようにします。

```
#include <dmbvideo.h>
rc = DBvReorgMetadata("employee");
```

## DBvSetFrameNumber

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

現行フレームを指定のフレーム番号にセットします。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbmpeg.lib | libdmbmpeg.a (AIX)<br>libdmbmpeg.sl (HP-UX)<br>libdmbmpeg.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvSetFrameNumber(
      DBvIOType *video
      unsigned long frameNumber
      );
```

### パラメーター

**video (in)**
ビデオ構造を指すポインター。

**frameNumber (in)**
要求されたフレームの番号。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_FRAME_NOT_FOUND**
要求されたフレームは見つかりませんでした。

**MMDB_NO_INDEX**
ビデオ索引がありません。

## 例

ビデオ・ファイル内の現行フレームをフレーム番号 85 にセットするには、次のようにします。

```
#include <dmbshot.h>
rc = DBvSetFrameNumber(video, 85);
```

## DBvSetShotComment

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

ショットの中の読み取り専用注釈を更新します。

### 許可

Control、Update

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvSetShotComment(
      char *catalogName ,
      char *shotHandle,
      char *comment,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
カタログの名前。

**shotHandle (in)**
更新するショットのハンドル。

**comment (in)**
ショットの新しい注釈。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_CANNOT_UPDATE**
この API ではショットを更新できません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

hotshots カタログにある、shotHandle が指定されているショットについて記述した注釈を変更するには、次のようにします。

```
#include <dmbshot.h>
rc = DBvSetShotComment("hotshot", shotHandle,
       "This is a hot shot.", hdbc);
```

## DBvUpdateShot

| イメージ | オーディオ | ビデオ |
|---|---|---|
| | | O |

カタログに入っているビデオ・ショットの属性を置き換えます。注釈の属性を除くすべての属性は、DBvShotType 構造の属性で置き換えられます。注釈ポインターが NULL の場合、既存の注釈は未変更のままになります。

### 許可

Control、Update

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbshot.lib | libdmbshot.a (AIX)<br>libdmbshot.sl (HP-UX)<br>libdmbshot.so (Solaris) |

### インクルード・ファイル

dmbshot.h

### 構文

```
long DBvUpdateShot(
      char *catalogName ,
      DBvShotType *shot,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

### パラメーター

**catalogName (in)**
カタログの名前。

**shot (in)**
ショットの属性が入っているショット情報構造を指すポインター。

**shotHandle (in)**
ショット・ハンドル。

**hdbc (in)**
SQLConnect からのデータベース・ハンドル。

## エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RC_NOT_CONNECTED**
アプリケーションのデータベース接続が有効ではありません。

**MMDB_RC_CANNOT_UPDATE**
この API ではショットを更新できません。

**MMDB_RC_NO_SHOT**
ショットは存在しません。

**MMDB_RC_INVALID_CATALOG**
カタログは有効でないか、存在しません。

## 例

hotshots カタログに入っているショットの属性を更新するには、次のようにします。

```
#include <dmbshot.h>
rc = DBvUpdateShot("hotshots", shot,
       shothandle, hdbc);
```

## DMBRedistribute (EEE のみ)

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

QBIC フィーチャー・データを再配布するのは、ノードをノード・グループに追加したり、そこから削除したとき、またはノード・グループの新しい区画化マップを設定したときです。

### 許可

この API はインスタンスの所有する ID から実行しなければなりません。

### ライブラリー・ファイル

| Windows | AIX および Solaris |
|---|---|
| dmbrd.lib | libdmbrd.a (AIX) |
| | libdmbrd.so (Solaris) |

### インクルード・ファイル

dmbrdst.h

### 構文

```
long DMBRedistribute (
       char  *pNodeGroupName,
       char  DataRedistOption  /* ""continue"" use CONTINUE parameter */
       );
                               /* blank:start redistribution */
```

### パラメーター

**pNodeGroupName (in)**
再配布するノード・グループの名前。

### エラー・コード

**MMDB_SUCCESS**
正常に処理された API 呼び出し。

**MMDB_RD_NO_CONTINUE**
CONTINUE パラメーターを指定しないで再実行依頼します。

**MMDB_RD_CONTINUE**
CONTINUE パラメーターを指定して再実行依頼します。

## 例

groupone というノード・グループに QBIC エクステンダー・データを再配布するには、次のようにします。

```
#include <dmbrdst.h>
rc = DMBRedistribute(groupone,"continue");
```

## QbAddFeature

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

現在オープンされているカタログにフィーチャーを追加します。データベースにおいて、QbAddFeature は指定されたフィーチャーのフィーチャー表を作成します。ユーザー表の画像列に画像を追加してから、QbReCatalogColumn API を使用して、フィーチャー表にそれぞれの画像の項目を追加して画像を分析します。

### 許可

Alter

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbAddFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**featureName (in)**
フィーチャーの名前。イメージ・エクステンダーには、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass

- QbDrawFeatureClass
- QbTextureFeatureClass

## エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECCatalogReadOnly**
このカタログは、読み取り専用でオープンされています。

**qbicECDupFeature**
このフィーチャーは、すでにカタログに組み込まれています。

**qbiECinvalidFeatureClass**
指定されたフィーチャーの名前の形式が有効ではありません。

## 例

CatHdl ハンドルで識別されるカタログに QbColorFeatureClass を追加するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbAddFeature(CatHdl,
       QbColorFeatureClass);
```

## QbCatalogColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

カタログ登録されていない画像を、ユーザー表の画像列にカタログします。この API は、それぞれの画像の項目をフィーチャー表に追加した後、画像を分析します。画像を分析する際、この API はイメージ・データを作成し、その画像の項目をフィーチャー表に保管します。フィーチャーの省略時パラメーターが使用されます。カタログは、オープンされている必要があります。

### 許可

Insert

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbCatalogColumn(
      QbCatalogHandle cHdl
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECInvalidCatalog**
指定されたハンドルまたは表列はカタログに対して有効ではありません。

**qbicECCatalog Errors**
個々の画像のカタログ作成時にエラーが発生し、そのエラーのログが記録されました。ロールバックはありません。

**qbicECImageNotFound**
画像が見つからないか、アクセスできません。

**qbicECCatalogRO**
カタログは読み取り専用になっています。

**qbicECSQLError**
SQL エラーが発生しました。

## 例

```
#include <dmbqbapi.h>
rc = QbCatalogColumn(CatHdl);
```

## QbCatalogImage

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像全体をカタログ登録します。この API は、画像の項目をフィーチャー表に追加した後、画像を分析します。画像を分析する際、この API はイメージ・データを作成し、その画像の項目をフィーチャー表に保管します。画像ハンドルは、現行 QBIC カタログに関連する画像列のハンドルを使用する必要があります。この画像は、現在定義されているフィーチャー・クラスに従ってカタログされます。カタログに登録されているフィーチャーの省略時パラメーターが使用されます。

### 許可

Insert

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbCatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**imgHandle (in)**
画像のハンドル。

## エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECImageNotFound**
画像が見つからないか、アクセスできません。

**qbicECCatalogRO**
カタログは読み取り専用になっています。

## 例

Img_hdl ハンドルで識別される画像をカタログ登録するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbCatalogColumn(CatHdl, Img_hdl);
```

## QbCloseCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

カタログをクローズします。この API は、オープンされているカタログ・ハンドルと割り振られている資源を解放します。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbCloseCatalog(
      QbCatalogHandle cHdl
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

### 例

CatHdl ハンドルで識別されるカタログをクローズするには、次のようにします。

## QbCloseCatalog

```
#include <dmbqbapi.h>
rc = QbCloseCatalog(CatHdl);
```

## QbCreateCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

現在接続されているデータベースに、指定されている画像列のカタログを作成します。 この列は、画像データ用に使用可能になっている必要があります。この API は、修飾子として使用されるカタログの名前を作成します。

### 許可

Alter

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbCreateCatalog(
     char *tableName,
     char *columnName,
     SQLINTEGER autoCatalog,
     char *reserved
     );
```

### パラメーター

**tableName (in)**
画像列が入っている表の名前。

**columnName (in)**
カタログを作成する画像列の名前。

**autoCatalog (in)**
画像列に追加された画像を自動的にカタログ登録するか、つまり、フィーチャー表に追加して分析するかどうかを指定します。自動カタログ化機能をオンに設定する場合は 1、オフに設定する場合は 0 を指定しま

す。自動カタログ化機能をオンに設定しない場合は、QbCatalogColumn API か QbCatalogImage API を使用して、画像列に追加した画像をカタログ化してください。

**reserved (in)**
現在使用されていません。

## エラー・コード

**qbicECSqlError**
SQL エラーが発生しました。

**qbicECNotEnabled**
データベース、表、または列が、DB2Image データ・タイプ用に使用可能になっていません。

**qbicECDupCatalog**
カタログはすでに存在しています。

**qbicECunsupportedOption**
サポートされないオプションが指定されました。

**qbicECerrorParameterTooLong**
パラメーターが長すぎて処理できませんでした。

**qbicECqerr**
QBIC エラーが発生し、メッセージが生成されました。

**qbicECqerrUnknown**
内部 QBIC エラーが発生し、一般エラー・メッセージが生成されました。

## 例

従業員表のピクチャー列に画像のカタログを作成するには、次のようにします。自動カタログ化機能をオンに設定するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbCreateCatalog("employee",
        "picture", 1);
```

## QbDeleteCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

指定されたカタログを現行データベースから削除します。

### 許可

Alter

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbDeleteCatalog(
      char *tableName,
      char *columnName
      );
```

### パラメーター

**tableName (in)**
この画像列が入っている表の名前。

**columnName (in)**
カタログと関連する画像列の名前。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECCatalogInUse**
だれか他の人がこのカタログを使用していました。

**qbicECCatalogRO**
カタログは読み取り専用になっています。

**qbicECSystem**
システム・エラーが発生しました。

**qbicECSqlError**
SQL エラーが発生しました。

## 例

従業員表内のピクチャー列と関連する QBIC カタログを削除するには、次のようにします。

```
#include <dmbqbapi.h>
rc=QbDeleteCatalog("employee", "picture");
```

## QbGetCatalogInfo

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

次の情報が組み込まれている QbCatalogInfo 構造を戻します。

- ユーザー表の名前とカタログが属す画像列。
- カタログに組み込まれているフィーチャーの数。
- 自動カタログ機能がオンに設定されているかどうか。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbGetCatalogInfo(
      QbCatalogHandle cHdl,
      QbCatalogInfo *catInfo
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**catInfo (out)**
カタログ情報構造。

### エラー・コード

**qbicECInvalidHandle**
カタログ・ハンドルが有効ではありません。

### 例

CatHdl ハンドルで識別されるカタログに関する情報を入手し、catInfo と呼ばれる構造で戻します。

```
#include <dmbqbapi.h>
rc = QbGetCatalogInfo(CatHdl, &catInfo);
```

## QbListFeatures

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

現在カタログに組み込まれている活動フィーチャーのリストを戻します。このリストは、割り振られているバッファーに戻されます。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbListFeatures(
      QbCatalogHandle cHdl,
      SQLINTEGER bufSize,
      SQLINTEGER *count,
      char *featureNames
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**bufSize (in)**
バッファーのサイズ。必要なバッファー・サイズを見積もるには、QbGetCatalogInfo API で戻されるフィーチャー・カウントを使用して、そのカウントに、最長のフィーチャー名の長さを掛けます。バッファーに保管されるフィーチャー名は、ブランク文字で区切ります。

**count (out)**
戻されるフィーチャー名の数。

**featureNames (out)**
バッファーに入っているフィーチャー名の配列。

## エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECTruncateData**
戻りバッファーが小さすぎたため、戻されたデータは切り捨てられました。

## 例

CatHdl ハンドルで識別されるカタログにある活動フィーチャーのリストを入手するには、次のようにします。この情報は、featureNames 配列に保管します。

まず、リストに必要なバッファー・サイズである bufSize を計算します。QbGetCatalogInfo API を使用して、catInfo 構造でフィーチャー数を戻します。そしてこの数値に、最長フィーチャー名のサイズである qbiMaxFeatureName 定数を掛けます。

```
#include <dmbqbapi.h>
rc = QbGetCatalogInfo(CatHdl, &catInfo);
bufSize =
   catInfo.featureCount*qbiMaxFeatureName;
rc = QbListFeatures(CatHdl, bufSize,
       count, featureNames);
```

## QbOpenCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

特定の画像列の QBIC カタログをオープンします。カタログは、読み取りモードまたは更新モードでオープンできます。この API は、オープンしたカタログのハンドルを戻します。そうすると、他の API のハンドルを使用してカタログを管理し、カタログにデータを入れることができます。

カタログでの作業が終わったら、そのカタログは必ずクローズするようにしてください。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbOpenCatalog(
      char *tableName,
      char *columnName,
      SQLINTEGER mode,
      QbCatalogHandle *cHdl
      );
```

### パラメーター

**tableName (in)**
その画像列が組み込まれている表の名前。

**columnName (in)**
画像列の名前。

**mode (in)**
カタログをオープンするときのモード。有効値は、qbiRead と qbiUpdate です。

**cHdl (out)**
カタログのハンドルを指すポインター。

## エラー・コード

**qbicECCatalogNotFound**
カタログが見つかりません。

**qbicECCatalogInUse**
だれか他の人がこのカタログを使用していました。

**qbicECOpenFailed**
カタログをオープンできません。

**qbicECNotEnabled**
カタログが使用可能になっていません。

**qbicECNoCatalogFound**
カタログがありません。

**qbicECSqlError**
SQL エラーが発生しました。

**qbicECSystem**
システム・エラーが発生しました。

## 例

従業員表のピクチャー列のカタログを読み取りモードでオープンするには、次のようにします。

```
#include <dmbqbapi.h>
rc=QbOpenCatalog("employee", "picture",
     qbiread, &CatHdl);
```

## QbQueryAddFeature

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

指定のフィーチャーを QBIC カタログに追加します。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryAddFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

### パラメーター

**qObj (in)**
照会オブジェクトのハンドル。

**featureName (in)**
追加される照会フィーチャーの名前。イメージ・エクステンダーには、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

**qbiECunknownFeatureClass**
指定されたフィーチャーのクラス名は、認識されていません。

**qbiECinvalidFeatureClass**
指定されたフィーチャーの名前の形式が有効ではありません。

**qbiECfeaturePresent**
指定されたフィーチャーはすでに照会オブジェクトのメンバーになっています。

**qbiECallocation**
システムが十分なメモリーを割り振ることができません。

## 例

qoHandle ハンドルによって識別される照会オブジェクトに QbColorFeatureClass フィーチャーを追加するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbQueryAddFeature(qoHandle,
        "QbColorFeatureClass");
```

## QbQueryCreate

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトを作成し、ハンドルを戻します。このハンドルを他の API で使用して、照会オブジェクトを操作することができます。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryCreate(
      QbQueryHandle *qObj
      );
```

### パラメーター

**qObj (out)**
照会ハンドルを指すポインター。正常完了の場合、このハンドルは 0 に設定されます。

### エラー・コード

**qbiECallocation**
システムが十分なメモリーを割り振ることができません。

### 例

照会オブジェクトを作成し、ハンドルを qoHandle に戻します。

## QbQueryCreate

```
#include <dmbqbapi.h>
rc = QbQueryCreate(&qoHandle);
```

## QbQueryDelete

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

名前が付けられていない照会オブジェクトを削除します。この API は、照会オブジェクトが使用しているすべてのメモリーと、追加されているフィーチャーすべてを解放します。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryDelete(
      QbQueryHandle qObj
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

### エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会を参照していません。

### 例

ハンドル qoHandle によって識別される照会オブジェクトを削除するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbQueryDelete(qoHandle);
```

## QbQueryGetFeatureCount

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトに追加されたフィーチャー数を戻します。イメージ・エクステンダーと共に、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryGetFeatureCount(
      QbQueryHandle qObj,
      SQLINTEGER* count
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**count (out)**
存在するフィーチャーの数を設定する変数を指すポインター。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

## 例

ハンドル qoHandle によって識別される照会オブジェクトのフィーチャーの数を戻すには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbQueryGetFeatureCount(qoHandle,
       &count);
```

## QbQueryGetString

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会から照会ストリングを戻します。ご使用のアプリケーションで照会ストリングを UDF への入力として (たとえば、UDF QbScoreFromStr または API QbQueryStringSearch で) 使用することができます。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryGetString(
      QbQueryHandle qObj,
      (char*)*  queryString
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**queryString (out)**
照会オブジェクトの照会ストリングを指すポインター。

### エラー・コード

**qbiECinvalidQueryHandle**
指定された照会ハンドルが有効な照会を参照していません。

### 例

ハンドル qrHandle によって識別される照会オブジェクトの照会ストリングの数を戻すには、次のようにします。

```
#include <dmbqbapi.h>
SQLRETURN  rc;
char *queryString;
QbQueryHandle qrHandle
rc = QbQueryGetString(qrHandle, &queryString);
if (rc == 0)  {
   ...                  /* use the returned queryString for input to UDFs */
   free((void*)queryString);  /* you must free queryString  */
   queryString=(char*)0;
}
```

## QbQueryListFeatures

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクト内のフィーチャーの現行リストを戻します。この API は、割り当てられているバッファーにこのリストを戻します。イメージ・エクステンダーと共に、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryListFeatures(
      QbQueryHandle qObj,
      SQLINTEGER bufSize,
      SQLINTEGER* count,
      char *featureNames
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**bufSize (in)**
featureNames バッファーのサイズ。 qbiMaxFeatureName 定数をバッファー・サイズとして使用します。照会オブジェクトのフィーチャーは文字ストリング名によって識別されます。

**count (out)**
戻されるフィーチャー名の数。

**featureNames (out)**
照会オブジェクトのフィーチャー名を指すポインター。この配列は、割り振られているバッファーに保管されます。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会ハンドルが有効な照会を参照していません。

## 例

ハンドル qoHandle によって識別される照会オブジェクトのフィーチャーの数を戻すには、次のようにします。必要なバッファーのサイズを判別するには、qbiMaxFeatureName 定数を使用します。フィーチャー名をフィーチャーのバッファーに戻し、フィーチャーの数を retCount 変数に戻します。

```
#include <dmbqbapi.h>
bufSize = qbiMaxFeatureName;
rc = QbQueryListFeatures(qoHandle, bufSize,
        &retCount, feats);
```

## QbQueryNameCreate

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトに名前を付けて保管し、UDF で使用できるようにします。照会オブジェクトの名前と説明は、ユーザーが提供します。

**注:**

1. **EEE のみ:** QbQueryNameCreate は区分データベース環境ではサポートされません。
2. 非区分データベース環境の場合、今後のリリースでは QbQueryNameCreate を使用できません。照会を保管するには、QbQueryGetString を使用して照会ストリングを入手し、それを保管して、今後のアプリケーションでの使用に備えてください。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryNameCreate(
      QbQueryHandle qObj,
      char *name,
      char *description
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**name (in)**
照会オブジェクトの名前。名前は 18 文字までです。

**description (in)**
照会オブジェクトの要旨。最高 250 文字までです。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会を参照していません。

## 例

QbQueryCreate API で作成した照会オブジェクトに名前と説明を与えるには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbQueryNameCreate(qHandle,
       "fshavgcol",
       "average color query, 10/15/96");
```

## QbQueryNameDelete

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトを削除します。この照会オブジェクトは、QbQueryNameCreate API を使用して名前を指定し、保管したものでなければなりません。

**注:**

1. **EEE のみ:** QbQueryNameDelete は区分データベース環境ではサポートされません。
2. 非区分データベース環境の場合、今後のリリースでは QbQueryNameDelete を使用できません。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryNameDelete(
      char *name
      );
```

### パラメーター

**name (in)**
削除する照会オブジェクトの名前。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

## 例

fshavgcol という名前の照会オブジェクトを削除するには、次のようにします。

```
#include <dmbqbapi.h>
rc = QbQueryNameDelete("fshavgcol",);
```

## QbQueryNameSearch

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトに組み込まれている探索基準と一致する画像に関する QBIC カタログを探索します。この照会オブジェクトは、名前で識別されます。結果値には画像ハンドルと QBIC 探索得点が含まれており、これらの値はクライアント・メモリーの結果配列に保管されます。これらの結果値は、得点に応じてソートされます。

**注:**

1. **EEE のみ:** QbQueryNameSearch は区分データベース環境ではサポートされません。
2. 非区分データベース環境の場合、今後のリリースでは QbQueryNameSearch を使用できません。照会を保管するには、QbQueryGetString を使用して照会ストリングを入手し、それを保管して、今後のアプリケーションでの使用に備えてください。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryNameSearch(
      char *qName,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
```

```
SQLINTEGER resultType,
SQLINTEGER* count,
QbResult* returns
);
```

## パラメーター

**qName (in)**
照会オブジェクトの名前。

**tableName (in)**
探索する画像列が入っている表の名前。

**columnName (in)**
画像列の名前。この列は、画像データ用に使用可能になっている必要があります。

**maxReturns (in)**
戻されるようにする画像の最大数。

**scope (in) (予約済み)**
0 (NULL) に設定する必要があります。

**resultType (in) (予約済み)**
qbiArray に設定する必要があります。

**count (out)**
戻される画像の数を指すポインター。ゼロが戻された場合、照会オブジェクト内のすべてのフィーチャーに関して画像列がカタログ化されていることを確認してください。

**returns (out)**
戻り結果値が保持されている QbResult 構造の配列を指すポインター。期待されるすべての結果値を保持できるよう、必ず十分な大きさのバッファーを割り振るようにしてください。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

## 例

従業員表のピクチャー列内のカタログ・イメージに対して照会 FSHAVGCOL を実行するには、次のようにします。画像が 7 個以上戻されることがないようにしてください。

```
#include <dmbqbapi.h>
rc = QbQueryNameSearch("fshavgcol",
       "employee", "picture",
       6, 0, qbiArray, &count, &returns);
```

## QbQueryRemoveFeature

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトから照会フィーチャーを除去して、関連するメモリーの割り振りを解除します。イメージ・エクステンダーと共に、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryRemoveFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**featureName (in)**
除去するフィーチャーの名前。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

**qbiECinvalidFeatureClass**
指定されたフィーチャーの名前の形式が有効ではありません。

**qbiECfeatureNotPresent**
指定されたフィーチャーは照会オブジェクトのメンバーではありません。

## 例

ハンドル qoHandle によって識別される照会オブジェクトから QbColorFeatureClass フィーチャーを除去します。

```
#include <dmbqbapi.h>
rc = QbQueryRemoveFeature(qoHandle,
        "QbColorFeatureClass");
```

## QbQuerySearch

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクトに組み込まれている探索基準と一致する画像に関する QBIC カタログを探索します。この照会オブジェクトは、照会オブジェクト・ハンドルによって識別されます。結果値には画像ハンドルと QBIC 探索得点が含まれており、これらの値はクライアント・メモリーの結果配列に保管されます。これらの結果値は、得点に応じてソートされます。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQuerySearch(
      QbQueryHandle qObj,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**tableName (in)**
探索する画像列が入っている表の名前。

**columnName (in)**
画像列の名前。 この列は、画像データ用に使用可能になっている必要があります。

**maxReturns (in)**
戻されるようにする画像の最大数。

**scope (in) (予約済み)**
0 (NULL) に設定する必要があります。

**resultType (in) (予約済み)**
qbiArray に設定する必要があります。

**count (out)**
戻される画像の数を指すポインター。ゼロが戻された場合、照会オブジェクト内のすべてのフィーチャーに関して画像列がカタログ化されていることを確認してください。

**returns (out)**
戻り結果値が保持されている QbResult 構造の配列を指すポインター。期待されるすべての結果値を保持できるよう、必ず十分な大きさのバッファーを割り振るようにしてください。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

## 例

従業員表のピクチャー列内のカタログ・イメージについて照会するには、次のようにします。画像が 7 個以上戻されることがないようにしてください。

```
#include <dmbqbapi.h>
rc = QbQuerySearch(qHandle, "employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

## QbQuerySetFeatureData

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクト内のフィーチャーに関して、画像データのソースをセットします。データ・ソースは、照会オブジェクトにフィーチャーを追加した後でなければセットできません。このデータ・ソースは、ユーザー表、ファイル、またはワークステーション・バッファー内の画像にすることができます。クライアント・ファイルまたはワークステーション・バッファーをデータ・ソースとして使用できるのは、非区分データベース環境である場合に限ります。さらに、平均色やヒストグラムのフィーチャーに対してはデータを明示的に指定することができます。

QbQuerySetFeatureData を使ってサーバー・ファイル内に画像データのソースをセットした後で、QbQueryStringSearch を使用してください。 QbQuerySearch では、QbQuerySetFeatureData を使ってサーバー・ファイル内にセットした画像データのソースを使用することはありません。

イメージ・エクステンダーには、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

## 構文

```
SQLRETURN QbQuerySetFeatureData(
      QbQueryHandle qObj,
      char *featureName,
      QbImageSource* imgSource
      );
```

## パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**featureName (in)**
設定するフィーチャーの名前。

**imgSource (in)**
イメージ・ソース構造を指すポインター。 imgSource に 0 (NULL) を指定すると、フィーチャー情報の変更を禁止することになります。詳細については、167ページの『データ・ソース構造体の使用』を参照してください。

## エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

**qbiECunknownFeatureClass**
指定されたフィーチャーのクラス名は、認識されていません。

**qbiECinvalidFeatureClass**
指定されたフィーチャーの名前の形式が有効ではありません。

**qbiECfeatureNotPresent**
指定されたフィーチャーは照会オブジェクトのメンバーではありません。

**qbiECfileUnreadable**
イメージ・ソース・ファイルが見つからないか、読み取れません。

## 例

ヒストグラム色に対するデータ・ソースを照会オブジェクトに指定します。このフィーチャーのデータ・ソースは、クライアント・ワークステーション上のファイルです。

```
#include <dmbqbapi.h>
QbQueryHandle qoHandle;
QbImageSource        imgSource;
imgSource.sourceType = qbiSource_ClientFile;
strcpy(featureName,"QbColorHistogramFeatureClass");
strcpy(imgSource.clientFile, "/tmp/image.gif");
rc = QbQuerySetFeatureData(qoHandle, featureName, &imgSource);
```

## QbQuerySetFeatureWeight

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会オブジェクト内の指定されたフィーチャーの重みをセットします。

### 許可

なし。

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQuerySetFeatureWeight(
      QbQueryHandle qObj,
      sqldouble* weight
      );
```

### パラメーター

**qObj (in)**
照会オブジェクト・ハンドル。

**weight (out)**
フィーチャーの重みにセットする変数を指すポインター。

### エラー・コード

**qbiECinvalidQueryHandle**
指定された照会オブジェクト・ハンドルが有効な照会オブジェクトを参照していません。

### 例

ハンドル qoHandle によって識別される照会オブジェクト内の平均カラー・フィーチャーの重みをセットするには、次のようにします。

```
#include <dmbqbapi.h>
weight=2.0
rc = QbQuerySetFeatureWeight(qoHandle, "QbColorFeatureClass", &weight);
```

## QbQueryStringSearch

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

照会ストリングに組み込まれている探索基準と一致する画像に関する QBIC カタログを探索します。結果値には画像ハンドルと QBIC 探索得点が含まれており、これらの値はクライアント・メモリーの結果配列に保管されます。これらの結果値は、得点に応じてソートされます。

### 許可

Select

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqqry.lib | libdmbqqry.a (AIX)<br>libdmbqqry.sl (HP-UX)<br>libdmbqqry.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbQueryStringSearch(
      char *queryString,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

### パラメーター

**queryString (in)**
照会ストリング。

**tableName (in)**
探索する画像列が入っている表の名前。

**columnName (in)**
画像列の名前。この列は、画像データ用に使用可能になっている必要があります。

**maxReturns (in)**
戻されるようにする画像の最大数。

**scope (in) (予約済み)**
0 (NULL) に設定する必要があります。

**resultType (in) (予約済み)**
qbiArray に設定する必要があります。

**count (out)**
戻される画像の数を指すポインター。ゼロが戻された場合、照会ストリング内のすべてのフィーチャーに関して画像列がカタログ化されていることを確認してください。

**returns (out)**
戻り結果値が保持されている QbResult 構造の配列を指すポインター。期待されるすべての結果値を保持できるよう、必ず十分な大きさのバッファーを割り振るようにしてください。

## エラー・コード

**qbiECinvalidQueryString**
指定された照会ストリングが無効です。

## 例

従業員表のピクチャー列内のカタログ・イメージについて照会するには、次のようにします。画像が 7 個以上戻されることがないようにしてください。

```
#include <dmbqbapi.h>
rc = QbQueryStringSearch("QbColorFeatureClass color=<255, 0, 0>"
       "employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

## QbReCatalogColumn

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

オープンした QBIC カタログに入っているすべての既存の画像を、新しいフィーチャー用に再分析します。フィーチャーの省略時パラメーターが使用されます。この API は、すでに画像が入っているカタログに新しいフィーチャーを追加したときに使用してください。

### 許可

Update、Insert

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbReCatalogColumn (
      QbCatalogHandle cHdl
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECInvalidCatalog**
指定されたハンドルまたは表列はカタログに対して有効ではありません。

**qbicECCatalog Errors**
個々の画像のカタログ作成時にエラーが発生し、そのエラーのログが記録されました。ロールバックはありません。

**qbicECImageNotFound**
画像が見つからないか、アクセスできません。

**qbicECCatalogRO**
カタログは読み取り専用になっています。

**qbicECSQLError**
SQL エラーが発生しました。

## 例

オープンした QBIC カタログに入っているすべての既存の画像を、新しいフィーチャー用に再分析します。

```
#include <dmbqbapi.h>
rc = QbReCatalogColumn(CatHdl);
```

## QbRemoveFeature

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

オープンされているカタログから、指定されたフィーチャーを削除します。

### 許可

Alter

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbRemoveFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**featureName (in)**
フィーチャーの名前。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECCatalogReadOnly**
このカタログは、読み取り専用でオープンされています。

**qbicECFeatureNotFound**
このフィーチャーは、カタログにありません。

**qbiECinvalidFeatureClass**
指定されたフィーチャーの名前の形式が有効ではありません。

## 例

CatHdl ハンドルで識別されるカタログから QbColorHistogramFeatureClass フィーチャーを除去するには、次のようにします。

```
#include <dmbqbapi.h>
rc=QbRemoveFeature(CatHdl,
     "QbColorHistogramFeatureClass");
```

## QbSetAutoCatalog

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像列にインポートされている画像を自動的にカタログします。この API は、それぞれの画像の項目をフィーチャー表に追加した後、画像を分析します。画像を分析する際、この API はイメージ・データを作成し、その画像の項目をフィーチャー表に保管します。

自動カタログ化機能をオンに設定しない場合は、QbCatalogColumn API か QbCatalogImage API を使用して、画像列に追加した画像をカタログ登録してください。

### 許可

Alter

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbSetAutoCatalog(
      QbCatalogHandle cHdl
      SQLINTEGER autoCatalog
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**autoCatalog (in)**
画像列に追加された画像をフィーチャー表に自動的に追加して分析する

かどうかを指定します。自動カタログ化機能をオンに設定する場合は 1、オフに設定する場合は 0 を指定します。

## エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

## 例

CatHdl ハンドルで識別されるカタログの自動カタログ化機能をオンに設定するには、次のようにします。

```
#include <dmbqbapi.h>
rc=QbSetAutoCatalog(CatHdl, 1);
```

## QbUncatalogImage

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

カタログから画像を除去します。画像ハンドルは、オープンされている QBIC カタログに関連する画像列のハンドルを使用する必要があります。画像は、オープンされているカタログから除去されます。画像属性表に対応する行がある場合は、画像がカタログされていないことを示します。

### 許可

Delete

### ライブラリー・ファイル

| OS/2 および Windows | AIX、HP-UX、および Solaris |
|---|---|
| dmbqbapi.lib | libdmbqbapi.a (AIX)<br>libdmbqbapi.sl (HP-UX)<br>libdmbqbapi.so (Solaris) |

### インクルード・ファイル

dmbqbapi.h

### 構文

```
SQLRETURN QbUncatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

### パラメーター

**cHdl (in)**
カタログのハンドルを指すポインター。

**imgHandle (in)**
画像のハンドル。このハンドルは、ユーザー表から検索できます。

### エラー・コード

**qbicECIvalidHandle**
カタログ・ハンドルが有効ではありません。

**qbicECImageNotFound**
画像が見つからないか、アクセスできません。

**qbicECCatalogRO**
カタログは読み取り専用になっています。

## 例

CatHdl ハンドルで識別されるカタログから、Img_hdl ハンドルで識別される画像を除去するには、次のようにします。

```
#include <dmbqbapi.h>
rc=QbUncatalogImage(CatHdl, Img_hdl);
```

# 第17章 クライアント側の管理コマンド

この章ではクライアント側で DB2 エクステンダー管理コマンドを入力する方法について説明します。さらに、クライアント側の各 DB2 エクステンダー管理コマンドに関する参照情報も提供します。

## DB2 エクステンダー管理コマンドの入力

DB2 エクステンダー管理コマンドは、対話モード、またはコマンド・モードの db2ext コマンド行プロセッサーに実行依頼することができます。対話モードの特徴は db2ext プロンプトです。このモードでは、DB2 エクステンダー管理コマンドしか入力することができません。コマンド・モードでは、オペレーティング・システムのコマンド・プロンプトからコマンドを入力します。 DB2 エクステンダー・コマンドだけでなく、DB2 コマンドやオペレーティング・システムのコマンドも入力することができます。

DB2 エクステンダー・コマンドを DB2 コマンド・プロンプトから入力しないでください。

対話モードで db2ext コマンド行プロセッサーを開始するには、次のようにします。

| クライアント | 処置 |
|---|---|
| OS/2 | DB2 エクステンダー・フォルダー内の DB2EXT コマンド行プロセッサー・アイコンをダブルクリックするか、OS/2 コマンド・プロンプトから DB2EXT コマンドを入力します。 |
| AIX、HP-UX、Solaris | オペレーティング・システムのコマンド・プロンプトから DB2EXT コマンドを入力します。 |
| Windows | DB2 エクステンダー・フォルダー内の DB2EXT コマンド行プロセッサー・アイコンをダブルクリックするか、DB2 コマンド・ウィンドウから DB2EXT コマンドを入力します。 |

対話モードを終了するには、quit または terminate コマンドを入力します。 quit コマンドは対話モードを終了しますが、DB2 との現在の接続は保持します。 terminate コマンドは対話モードを終了し、DB2 との現行接続を解除します。

DB2 エクステンダー・コマンドをコマンド・モードで実行依頼するには、オペレーティング・システムのコマンド行から入力します。各 DB2 エクステンダー・コマンドの前に db2ext を入力する必要があります。以下に例を示します。

```
db2ext enable database for db2image using mydataspace, myindxspace, mylongspace
```

## DB2 エクステンダー・コマンドに関するオンライン・ヘルプの入手

すべての DB2 エクステンダー・コマンドに関するオンライン・ヘルプを入手するには、次のように入力します。

```
db2ext ?
```

## ADD QBIC FEATURE

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

現行カタログで指定されているフィーチャーのフィーチャー表を作成します。このカタログにある既存の画像は、イメージ・エクステンダーにより自動的に再分析されます。

### 許可

Alter、Control、SYSADM、DBADM

### コマンド構文


### コマンド・パラメーター

**feature_name**

QBIC カタログに追加するフィーチャーの名前。イメージ・エクステンダーには、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 例

現在オープンされているカタログに QbColorFeatureClass フィーチャーを追加するには、次のようにします。

```
add qbic feature qbcolorfeatureclass
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

カタログは、オープンされている必要があります。

## CATALOG QBIC COLUMN

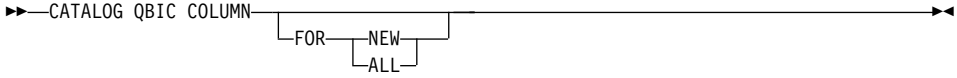
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像を画像列にカタログし、現在オープンされている QBIC カタログをフィーチャー・データで更新します。このカタログは、画像列にあるすべての画像について更新できます。または、最後にカタログを分析した後で画像列に追加された新しい画像だけについて更新することもできます。

### 許可

Insert、Control、SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

なし。

### 例

新しい画像、つまり、カタログ登録されていない画像を現行カタログにカタログ登録するには、次のようにします。

```
catalog qbic column for new
```

### 使用上の注意

NEW を指定すると、イメージ・エクステンダーはカタログされていない画像だけでカタログを更新します。 ALL を指定すると、イメージ・エクステンダーは現行カタログの画像列にあるすべての画像を分析します。 NEW が省略時値です。

このコマンドを使用する前に、必ずデータベースに接続してください。

カタログは、オープンされている必要があります。

## CLOSE QBIC CATALOG

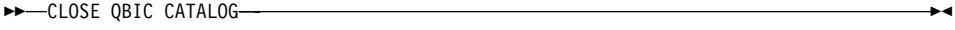
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

QBIC カタログをクローズします。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

なし。

### 例

現行カタログをクローズするには、次のようにします。

```
close qbic catalog
```

### 使用上の注意

QBIC カタログは、オープンされている必要があります。

## CONNECT

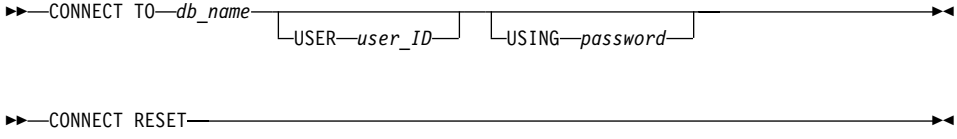
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

データベースに接続します。エクステンダーでは、DB2 接続とは別にデータベースに接続する必要があります。

### 許可

Connect

### コマンド構文




### コマンド・パラメーター

**db_name**
データベースの名前。

**user_ID**
データベースへの接続を許可されているユーザー ID。

**password**
そのユーザー ID のパスワード。

**RESET**
保留中の変更をすべてコミットしてからデータベースを切断します。

### 例

PERSONNL データベースに接続するには、次のようにします。ユーザー ID は anita、パスワードは anitapas です。

```
connect to personnl user anita
   using anitapas
```

### 使用上の注意

データベースは共用モードで接続されます。

このコマンドは、他のエクステンダー・コマンドより先に実行してください。

## CREATE QBIC CATALOG

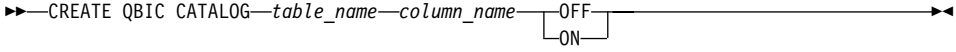
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

現行データベースに、指定されている DB2IMAGE 列の QBIC カタログを作成します。エクステンダーは、カタログ名を自動生成します。

### 許可

Alter、Control、SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

**table_name**
DB2IMAGE が使用可能になっている表の名前。

**column_name**
DB2IMAGE が使用可能になっている列の名前。

**OFF** 画像は手動でカタログ化されます。

**ON** 画像は自動でカタログ化されます。

**tablespace_name**
QBIC カタログの表スペース指定および索引オプション。この指定は、以下の 4 つの部分からなります。

- フィーチャー・データが入るカタログ表の表スペースの名前。この表スペースの指定は必須です。この表スペースは、セグメント化表スペースにする必要があります。
- カタログ表に作成する索引の場合、使用ブロック、空きブロック、gbpcache ブロック、またはタイプ 2 非区画化索引の索引オプションを任意に組み合わせます。この指定は任意です。この部分を指定しないと省略時値になります。
- カタログ・ログ表用の表スペースの名前。この表スペースは、単純表スペースまたはセグメント化表スペースのどちらでも構いません。この指定は任意です。ログ表に表スペースを指定しない場合、フィーチャー・データ表に指定された表スペースが使用されます。

- ログ・データ表に作成する索引の場合、使用ブロック、空きブロック、gbpcache ブロック、またはタイプ 2 非区画化索引の索引オプションを任意に組み合わせます。この指定は任意です。この部分を指定しないと省略時値になります。

## 例

自動カタログ化機能を ON に設定して、従業員表のピクチャー列の QBIC カタログを作成するには、次のようにします。

```
create qbic catalog employee picture on
```

## 使用上の注意

ON を指定すると、列にインポートされる画像は関連する QBIC カタログに自動的にカタログされます。省略時値は OFF です。

このコマンドを使用する前に、必ずデータベースに接続してください。

## DELETE QBIC CATALOG

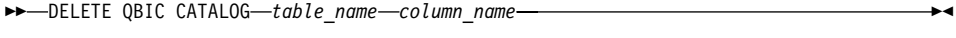
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

すべての QBIC 探索サポート・データを含めて、QBIC カタログを削除します。

### 許可

Alter、Control、SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

**table_name**
DB2IMAGE が使用可能になっている表の名前。

**column_name**
DB2IMAGE が使用可能になっている列の名前。

### 例

従業員表のピクチャー列と関連するカタログを削除するには、次のようにします。

```
delete qbic catalog employee picture
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## DISABLE COLUMN

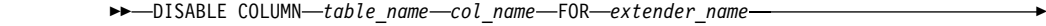
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定した列を使用不可にして、指定したメディア・データを保管できないようにします。

### 許可

SYSADM、DBADM、Control、Alter

### コマンド構文




### コマンド・パラメーター

**table_name**
現行データベースに入っている表の名前。

**col_name**
使用不可にする列の名前。

**extender_name**
列を使用不可にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

従業員表にある写真列を使用不可にして、画像データを保持できないようにするには、次のようにします。

```
disable column employee photo for db2image
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

列を使用不可にすると、次のようになります。

- この列は、指定したエクステンダーのデータを保管できなくなります。これは、同じ表に入っている他の列でマルチメディア・データ・タイプを使用できるかどうかには影響を与えません。
- 列項目の内容は NULL に設定され、管理表の対応する行は削除されます。

- この列に関連するトリガーは、消去されます。

## DISABLE DATABASE

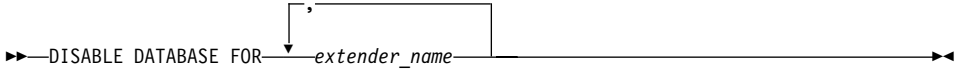
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

データベースを使用不可にして、メディア・データを保管できないようにします。

### 許可

SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

**extender_name**
現行データベースを使用不可にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

現行データベースを使用不可にして、イメージ・データを保持できないようにするには、次のようにします。

```
disable database for db2image
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

データベースを使用不可にすると、システムは次のことを行います。

- 指定したエクステンダーについてのみ、使用可能になっているすべての表を使用不可にします。
- 指定したエクステンダーの UDF 管理サポート表を消去します。

## DISABLE TABLE

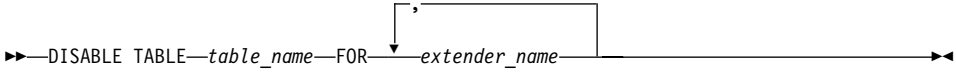
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定した表を使用不可にして、メディア・データを保管できないようにします。

### 許可

SYSADM、DBADM、Control、Alter

### コマンド構文




### コマンド・パラメーター

**table_name**
現行データベースで使用不可にしたい表の名前。

**extender_name**
表を使用不可にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

従業員表を使用不可にして、イメージ・データを保持できないようにするには、次のようにします。

```
disable table employee for db2image
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

表を使用不可にすると、システムは次のことを行います。

- この表の中で、指定したエクステンダーについて使用可能になっているすべての列を使用不可にします。
- この表に関連した管理サポート表を除去します。

## DISCONNECT SERVER AT NODENUM (EEE のみ)

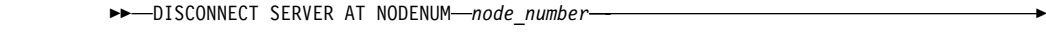
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

すべてのデータベース上で、指定したノードからサーバーを切断します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**node_number**
サーバーから切断したいノード。

### 例

ノード番号 2 ですべてのデータベースからサーバーを切断するには、次のようにします。

```
disconnect server at nodenum 2
```

### 使用上の注意

すべてのノード上ですべてのデータベースからサーバーを切断するには、DMBSTOP コマンドを使用します。

## DISCONNECT SERVER FOR DATABASE (EEE のみ)

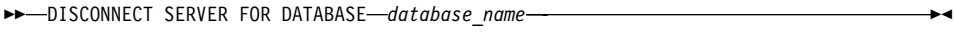
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定したデータベースのすべてのノードからサーバーを切断します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**database_name**
サーバーから切断したいデータベース。

### 例

MY_DATABASE というデータベースからサーバーを切断します。

```
disconnect server for database my_database
```

### 使用上の注意

すべてのノード上ですべてのデータベースからサーバーを切断するには、DMBSTOP コマンドを使用します。

## DISCONNECT SERVER FOR DATABASE AT NODENUM (EEE のみ)

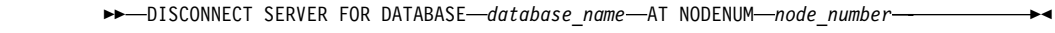
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定したノード上で、指定したデータベースからサーバーを切断します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**database_name**
サーバーから切断したいデータベース。

**node_number**
サーバーから切断したいノード。

### 例

ノード番号 2 で MY_DATABASE というデータベースからサーバーを切断するには、次のようにします。

```
disconnect server for database my_database at nodenum 2
```

### 使用上の注意

すべてのノード上ですべてのデータベースからサーバーを切断するには、DMBSTOP コマンドを使用します。

## ENABLE COLUMN

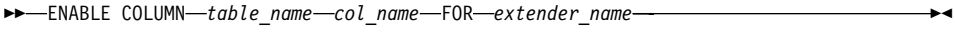
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定した列を使用可能にして、メディア・データを保管できるようにします。

### 許可

SYSADM、DBADM、Control、Alter

### コマンド構文




### コマンド・パラメーター

**table_name**
現行データベースに入っている表の名前。

**col_name**
使用可能にする列の名前。

**extender_name**
表を使用可能にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

従業員表の写真列を使用可能にして、イメージ・データを保持できるようにするには、次のようにします。

```
enable column employee photo for db2image
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## ENABLE DATABASE

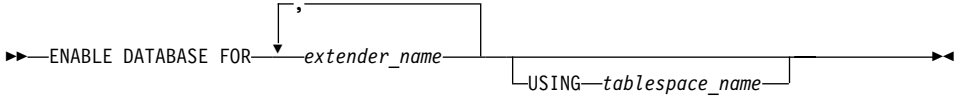
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

現行データベースを使用可能にし、指定した表スペースを使用してメディア・データを保管します。

### 許可

SYSADM、SYSCTRL、DBADM

### コマンド構文




### コマンド・パラメーター

**extender_name**
現行データベースを使用可能にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

**tablespace_name**
管理表の保管先のコンテナーの集まりである、表スペースの名前。表スペース名は、*datats*、*indexts*、*longts* という 3 つの部分からなります。*datats* はメタデータ表が作成される表スペースの名前、*indexts* はメタデータ表の索引が作成される表スペースの名前、*longts* はメタデータ表内の長い列 (たとえば、LONG VARCHAR や LOB データ・タイプが入っている列) の値が保管される表スペースの名前です。表スペース名のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースの名前が使用されます。指定する表スペースは、区分データベース・システムのすべてのノードを含むノードグループで定義する必要があります。

### 例

現行データベースを使用可能にして、イメージ・データを保持できるようにするには、次のようにします。

```
enable database for db2image using mydataspace, myindxspace, mylongspace
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

表スペースを指定しない場合、システムは管理表の USERSPACE1 表スペースを使用します。

## ENABLE TABLE

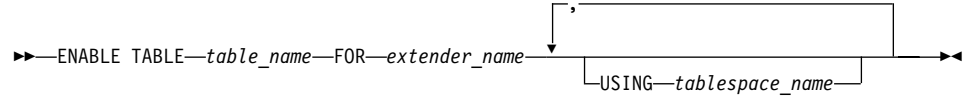
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定した表を使用可能にし、指定した表スペースを使用してメディア・データを保管します。

### 許可

SYSADM、DBADM、Control、Alter

### コマンド構文




### コマンド・パラメーター

**table_name**
現行データベースの中の、使用可能にする表の名前。

**extender_name**
表を使用可能にする対象のエクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

**tablespace_name**
管理表の保管先のコンテナーの集まりである、表スペースの名前。表スペースの指定は、*datats*、*indexts*、*longts* という 3 つの部分からなります。ここで、*datats* はメタデータ表を作成する表スペース、*indexts* はメタデータ表の索引を作成する表スペース、*longts* はメタデータ表内の長い列 (たとえば、LONG VARCHAR および LOB データ・タイプが入っている列) の値が保管される表スペースです。表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

表スペース指定のいずれかの部分にヌル値を指定すると、その部分の省略時の表スペースが使用されます。

**EEE のみ:** 指定された表スペースは、ユーザー表と同じノード・グループになければなりません。

## 例

従業員表を使用可能にして、イメージ・データを保持できるようにするには、次のようにします。

```
enable table employee for db2image
   using mydataspace, myindxspace, mylongspace
```

従業員表を使用可能にして、イメージ・データを保持できるようにするには、次のようにします。省略時表スペースを使用します。

```
enable table employee for db2image
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

表スペースを指定しない場合、システムは現行データベースを使用可能にしたときに定義された表スペースを使用します。

## GET EXTENDER STATUS

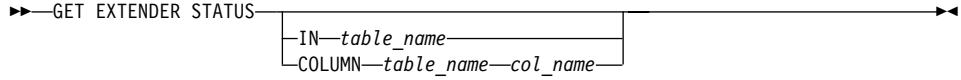
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

列、表、または現行データベースで使用可能になっているエクステンダーが存在する場合、その名前を表示します。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

**table_name**
現行データベースに入っている表の名前。

**col_name**
列の名前。

### 例

データベースで使用可能になっているエクステンダーの名前を表示するには、次のようにします。

```
get extender status
```

従業員表の状況を表示します。

```
get extender status in employee
```

従業員表の ADDRESS 列の状況を表示するには、次のようにします。

```
get extender status column employee address
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## GET INACCESSIBLE FILES

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

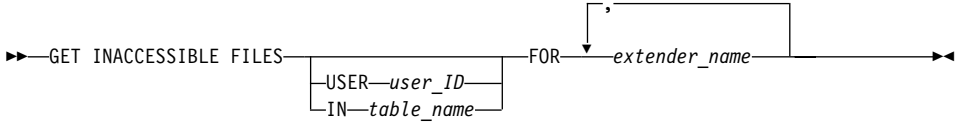
特定の表、特定の修飾子が指定されているすべての表、または現行データベース内のすべての表によって参照されているメディア・ファイルの中で、アクセス不能なものをすべてリストします。

### 許可

現行データベースのすべての表の場合 (つまり、USER または IN を指定しない場合) SYSADM、SYSCTRL、SYSMAINT、DBADM

特定の表の場合 (IN を指定)、または特定の修飾子が指定されている表 (USER を指定) の場合、Select

### コマンド構文




### コマンド・パラメーター

**user_ID**
アクセス不能ファイルのリストを表示したい、現行データベースの表の修飾子。

**table_name**
アクセス不能ファイルのリストを表示したい、現行データベースの表の名前。

**extender_name**
エクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

データベース内の表によって参照されている、アクセス不能なすべての画像ファイルのリストを表示するには、次のようにします。

```
get inaccessible files
   for db2image
```

修飾子 anita を含む表の中で参照されている、アクセス不能なすべての画像ファイルのリストを表示するには、次のようにします。

```
get inaccessible files
   user anita for db2image
```

従業員表の中で項目で参照されている、アクセス不能なすべての画像ファイルのリストを表示するには、次のようにします。

```
get inaccessible files
   in employee FOR db2image
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

表を指定してこのコマンドを実行すると、その表のアクセス不能ファイルのリストが表示されます。修飾子を指定してこのコマンドを実行すると、その修飾子が指定されている表だけのアクセス不能ファイルのリストが表示されます。どちらも指定しないでこのコマンドを実行すると、現行データベースのすべての表のアクセス不能ファイルのリストが表示されます。

## GET QBIC CATALOG INFO

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

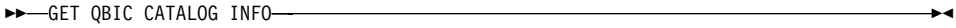
現在オープンされているカタログに関して次の情報を戻します。

- ユーザー表の名前とカタログに関連する画像列。
- カタログに組み込まれているフィーチャーの名前。
- カタログに組み込まれているフィーチャーの数。
- 自動分析機能がオンになっているかどうか。

### 許可

Select、Control、SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

なし。

### 例

現在オープンされているカタログに関する情報を入手するには、次のようにします。

```
get qbic catalog info
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

カタログは、オープンされている必要があります。

## GET REFERENCED FILES

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

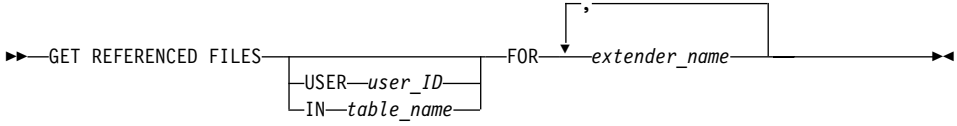
すべてのメディア・ファイルのリストと、特定の表、特定の修飾子が指定されているすべての表、または現行データベース内のすべての表の中でこれらのファイルを参照している列の名前のリストを表示します。

### 許可

現行データベースにあるすべての表の場合 (つまり、USER または IN を指定しない場合)、SYSADM、SYSCTRL、SYSMAINT、DBADM

特定の表の場合 (IN を指定)、または特定の修飾子が指定されている表 (USER を指定) の場合、Select

### コマンド構文




### コマンド・パラメーター

**user_ID**
参照するファイルのリストを表示したい、データベースの表の修飾子。コマンドを実行すると、この修飾子が指定されている表だけが探索されます。

**table_name**
参照ファイルのリストを表示する、現行データベースの表の名前。コマンドを実行すると、ここで指定する表だけが探索されます。

**extender_name**
エクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

データベースのすべての表の中で表項目によって参照されているすべての画像ファイルのリストを表示するには、次のようにします。

```
get referenced files
   for db2image
```

修飾子 anita を含む表の中の項目によって参照されているすべての画像ファイルのリストを表示するには、次のようにします。

```
get referenced files
   user anita for db2image
```

従業員表内で項目によって参照されているすべての画像ファイルのリストを表示するには、次のようにします。

```
get referenced files
   in employee for db2image
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

パラメーターを指定しないでこのコマンドを実行すると、データベース内のすべての表が探索されます。

## GET SERVER STATUS

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

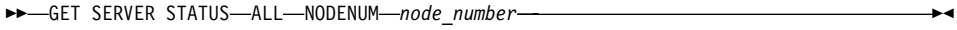
現行データベースまたはすべてのデータベースの、エクステンダー・サーバーの状況を表示します。

**EEE のみ:** ノードが指定されている場合は、このコマンドは、そのノードだけのエクステンダー・サーバーの状況 (現行データベースまたはすべてのデータベースの) を表示します。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

**ALL** すべてのデータベースの状況を表示します。

**node_number**
ノードの番号。このコマンドが表示するのは、このノードの状況です。 **(EEE のみ)**

### 例

現行データベースのエクステンダー・サーバーの状況を表示するには、次のようにします。

```
get server status
```

すべてのデータベースについてエクステンダー・サーバーの状況を表示するには、次のようにします。

```
get server status all
```

すべてのデータベースについてノード番号 2 のエクステンダー・サーバーの状況を表示するには、次のようにします。

```
get server status all nodenum 2
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

パラメーターを指定しないでこのコマンドを実行すると、db2nodes.cfg ファイルにリストされている現行データベースのすべてのノードの状況が表示されます。

## OPEN QBIC CATALOG

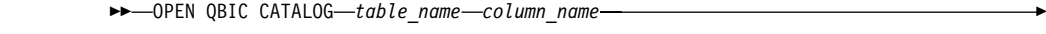
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

指定した DB2IMAGE 列のカタログをオープンします。データベースは、常に更新モードでカタログのオープンを試行します。カタログがすでに更新モードになっている場合、このカタログは読み取りモードでオープンされます。

### 許可

Connect

### コマンド構文




### コマンド・パラメーター

**table_name**
DB2IMAGE が使用可能になっている表の名前。

**column_name**
DB2IMAGE が使用可能になっている列の名前。

### 例

従業員表のピクチャー列の QBIC カタログをオープンするには、次のようにします。

```
open qbic catalog employee picture
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

このコマンドを実行すると、すべてのオープンされているカタログがクローズされます。

## QUIT

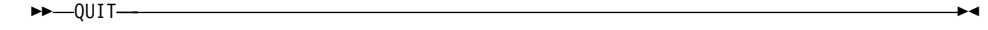
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

対話モードでコマンド入力するための db2ext コマンド行プロセッサーをシャットダウンします。 DB2 への接続は保持されるので、コマンド・モードの db2ext コマンド行プロセッサーには引き続きコマンドの実行依頼を要求することができます。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

なし。

### 例

対話モードのコマンド行インターフェースを遮断します。

```
quit
```

### 使用上の注意

QUIT はデータベースへの接続を保持します。

## RECONNECT SERVER AT NODENUM (EEE のみ)

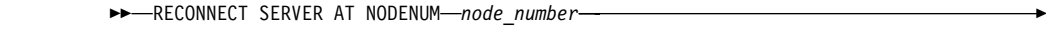
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

すべてのデータベース上で、指定したノードにサーバーを再接続します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**node_number**
サーバーに再接続したいノード。

### 例

ノード番号 2 ですべてのデータベースにサーバーを再接続するには、次のようにします。

```
reconnect server at nodenum 2
```

### 使用上の注意

すべてのノードですべてのデータベースからサーバーを再接続するには、DMBSTART コマンドを使用します。

## RECONNECT SERVER FOR DATABASE (EEE のみ)

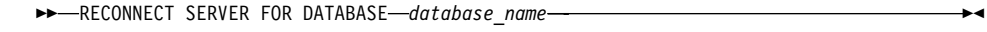
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定したデータベースのすべてのノードにサーバーを再接続します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**database_name**
サーバーに再接続したいデータベース。

### 例

MY_DATABASE というデータベースにサーバーを再接続します。

```
disconnect server for database my_database
```

### 使用上の注意

すべてのノード上ですべてのデータベースにサーバーを再接続するには、DMBSTART コマンドを使用します。

## RECONNECT SERVER FOR DATABASE AT NODENUM (EEE のみ)

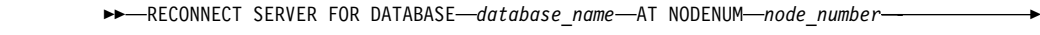
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

指定したノード上で、指定したデータベースにサーバーを再接続します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**database_name**
サーバーに再接続したいデータベース。

**node_number**
サーバーに再接続したいノード。

### 例

ノード番号 2 で MY_DATABASE というデータベースにサーバーを再接続するには、次のようにします。

```
reconnect server for database my_database at nodenum 2
```

### 使用上の注意

すべてのノード上ですべてのデータベースにサーバーを再接続するには、DMBSTART コマンドを使用します。

## REDISTRIBUTE NODEGROUP (EEE のみ)

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

ノードをノード・グループに追加したり、そこから削除したとき、またはノード・グループの新しい区画化マップを設定したときに、エクステンダー・データを再配布します。

### 許可

SYSADM、DBADM

### コマンド構文


### コマンド・パラメーター

**nodegroup**
再配布するノード・グループの名前。

**CONTINUE**
再分散プロセスがエラーを戻した場合は、コマンドの応答にある指示に従って CONTINUE パラメーターをコマンドに指定するかどうかを決めて、もう一度コマンドを実行することができます。このオプションは、プロセスを最初からやり直すのではなく、プロセスが停止した場所から続行するよう、システムに指示します。

### 例

my_nodegroup というノード・グループを再配布するには、次のようにします。

```
redistribute nodegroup my_nodegroup
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

CONTINUE パラメーターは、DB2 の REDISTRIBUTE コマンドの後で REDISTRIBUTE NODEGROUP コマンドを初めて実行するときには、使用しないでください。初めて実行するときに使用すると、エラーが記録され、再配布を最初からやり直します。

データ保全性を維持するには、ノード・グループを一度に 1 つずつ再配布する必要があります。 1 つのノード・グループの再配布が完了するまで待ってから、次のノード・グループの再配布を開始してください。

REDISTRIBUTE NODEGROUP が失敗した場合には、次のディレクトリーのうちいずれかにある ″redist.log″ ファイルで詳細を参照することができます。

- **Unix:** /<home-instance>/dmb/redist
- **Windows:**
  \\<instance_owning_machine>¥DB2<instance_name>¥<instance_name>¥dmb¥redist

## REMOVE QBIC FEATURE

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

オープンされているカタログから、指定されたフィーチャーのフィーチャー表を削除します。

### 許可

Alter、Control、SYSADM、DBADM

### コマンド構文




### コマンド・パラメーター

**feature_name**

QBIC カタログから除去するフィーチャーの名前。イメージ・エクステンダーには、次のフィーチャーが提供されています。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

### 例

現在オープンされているカタログから QbColorFeatureClass フィーチャーを除去するには、次のようにします。

```
remove qbic feature qbcolorfeatureclass
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

カタログは、オープンされている必要があります。

## REORG

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

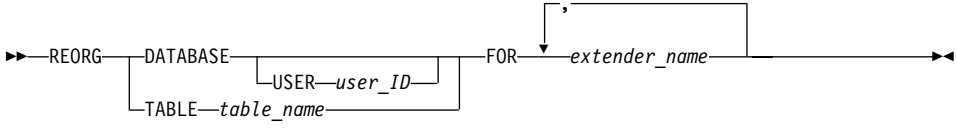
特定の表、特定の修飾子が指定されているすべての表、または現行データベース内のすべての表と関連する管理表 (管理表と属性表) をクリーンアップします。

### 許可

特定の表 (REORG TABLE)、または特定の修飾子が指定されている表 (REORG DATABASE) の場合、SYSADM、SYSCTRL、SYSMAINT、DBADM、Control

データベースにあるすべての表 (REORG DATABASE) の場合、SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

**user_ID**
表の修飾子。

**table_name**
管理表をクリーンアップする、現行データベースにある表の名前。

**extender_name**
エクステンダーの名前。有効なエクステンダー名は、db2image、db2audio、および db2video です。

### 例

現行データベースの画像管理表を再編成し、クリーンアップするには、次のようにします。

```
reorg database for db2image
```

修飾子 anita が指定されているすべての表に入っている画像管理表を再編成し、クリーンアップするには、次のようにします。

```
reorg database user anita for db2image
```

従業員表の画像管理表を再編成し、クリーンアップするには、次のようにします。

```
reorg table employee for db2image
```

## 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## SET QBIC AUTOCATALOG

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | | |

画像が列にインポートされるときに、画像を自動カタログします。画像は、列と関連する QBIC カタログに追加されます。

### 許可

Alter、Control、SYSADM、DBADM

### コマンド構文


### コマンド・パラメーター

なし。

### 例

自動カタログ化機能をオンに設定するには、次のようにします。

```
set qbic autocatalog on
```

### 使用上の注意

QBIC カタログは、オープンされている必要があります。

## START SERVER (EEE 以外の場合のみ)

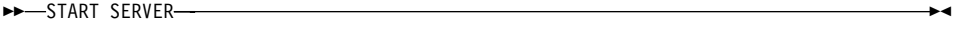
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

現行データベースについてエクステンダー・サーバーを開始します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

なし。

### 例

現行データベースについてエクステンダー・サーバーを開始するには、次のようにします。

```
start server
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## STOP SERVER (EEE 以外の場合のみ)

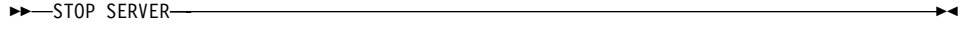
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

現行データベースについてエクステンダー・サーバーを停止します。

### 許可

SYSADM、SYSCTRL、SYSMAINT、DBADM

### コマンド構文




### コマンド・パラメーター

なし。

### 例

現行データベースについてエクステンダー・サーバーを停止するには、次のようにします。

```
stop server
```

### 使用上の注意

このコマンドを使用する前に、必ずデータベースに接続してください。

## TERMINATE

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

db2ext コマンド行プロセッサーをシャットダウンし、DB2 への接続を解除します。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

なし。

### 例

db2ext コマンド行プロセッサーをシャットダウンします。

```
quit
```

### 使用上の注意

TERMINATE はデータベースへの接続を解除します。

# 第18章 サーバー側の管理コマンド

この章に記載するコマンドは、サーバーのオペレーティング・システムのコマンド行で実行されます。 DB2 コマンド行や db2ext コマンド行からは実行しないでください。サーバー・システムをシャットダウンして再始動した場合は、常に DMBSTART コマンドを実行してください。

**EEE のみ:** 複数区分データベース環境では、DMBSTART および DMBSTOP サーバー・コマンドを発行することもできます。複数区分データベース環境でサーバー・コマンドを実行する際に、ノード番号を含めない場合はそのコマンドはすべてのノードに適用されます。ノード番号を指定すると、コマンドは指定されたノードにのみ適用されます。

**EEE のみ:** 複数区画環境では、DMBSTAT コマンドは実行できません。複数区画環境でサーバー状況を調べるには、クライアント・コマンド GET SERVER STATUS ALL を実行します。

## DMBICRT

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

DB2 エクステンダー・インスタンスを作成します。 DB2 のインスタンスが複数存在する場合には、 DB2 エクステンダー・サーバーの複数のインスタンスを作成する必要があります。 UNIX では、DB2 エクステンダー・クライアントをインストールする時にクライアント・インスタンスを作成します。クライアント・インスタンスを作成することによって、DB2 エクステンダーを使用するための環境が設定されます。

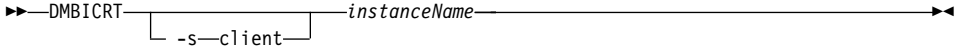
### 許可

SYSADM

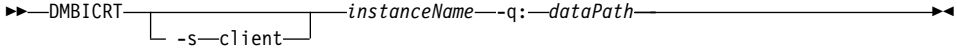
UNIX では root 権限が必要です。

### コマンド構文

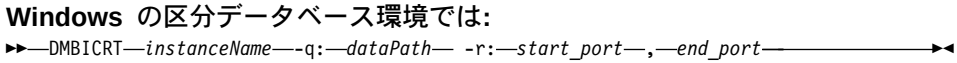
**非区分データベース環境では:**




**UNIX の区分データベース環境では:**




**Windows の区分データベース環境では:**




### コマンド・パラメーター

**instanceName**
既存の DB2 インスタンスの名前。この名前の DB2 インスタンスが存在しない場合には、それを作成するかどうか尋ねられます。

**-s client**
クライアントだけのインスタンスを作成することを指定します。このパラメーターを使用する場合には、*instanceName* がク

ライアントのユーザー ID となります。クライアント・インスタンスを作成することによって、クライアント用の環境が設定されます。 **(UNIX の場合のみ)**

**dataPath** 共用ディレクトリーまたは共用ファイル・システムの名前。ディレクトリーは、すべてのノードに存在していなければなりません。これは、UNIX の場合は DB2MMDATAPATH 環境変数で、Windows の場合はレジストリーで、それぞれ設定されます。 **(EEE のみ)**

**start_port, end_port**
使用する TCP/IP ポートの範囲。ポートの範囲は、使用するノードの数に等しいか、それより大きくなければなりません。ポート番号は Windows のレジストリーに書き込まれます。 **(Windows EEE のみ)**

## 例

非区分データベース環境では、DB2 インスタンス DEVINST 用の DB2 エクステンダー・サーバーのインスタンスを以下のように作成します。

```
dmbicrt  devinst
```

## 使用上の注意

DMBICRT コマンドは、インスタンスが使用するファイル用の DB2 エクステンダー・インスタンス・ディレクトリーを作成します。このディレクトリーには以下のような名前が付けられます。

- *install_directory*¥INSTANCE¥*instance_name*。ここで *install_directory* は DB2 エクステンダーをインストールしたディレクトリー **(Windows、OS/2 の場合)**。
- *INSTHOME*/dmb。ここで *INSTHOME* はインスタンス所有者のホーム・ディレクトリー **(UNIX の場合)**。

DMBICRT コマンドを使用する際に、指定された名前の DB2 インスタンスが存在しない場合には、それを作成するかどうか尋ねられます。

**EEE のみ:**

DMBICRT は参与している任意のノードの root ユーザー ID から実行することはできますが、1 つの同じノードを使用して、すべての DB2 エクステンダー・サーバー・インスタンスを作成することをお勧めします。また、そのノードは DB2 インスタンスの作成に使用し、DB2 インスタンス・ディレクトリーが存在しているノードと同じものにしてください。異なるノード

を使用して DB2 エクステンダー・サーバー・インスタンスを作成した場合、いずれかのノードに保管されているインスタンスのリストが完全ではない可能性があります。

*dataPath* として指定した共用ディレクトリーまたは共用ファイル・システムは、UNIX では *$INSTHOME*/dmb/dmbprofile 内の DB2MMDATAPATH 環境変数の値として保管され、Windows では以下のレジストリー・キーに保管されます。

```
¥¥HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM\DB2 Extenders\PROFILES
     ¥instance_name¥DB2MMDATAPATH
```

UNIX の場合、インスタンスを作成する前に、/etc/services ファイルにポート範囲を追加しなければなりません。次の構文を使用して、このファイルに 2 つの項目を追加します。

– DMB_*instance_name start_port*

– DMB_*instance_name*_END *end_port*

設定する範囲は、区分データベース環境のすべてのノードが使用できる大きさにする必要があります。

DB2 エクステンダー・サーバーのインスタンスを作成する前に、DB2 インスタンスを作成しなければなりません。

区分データベース環境でテキスト・エクステンダー・インスタンスを作成するには、 *DB2 ユニバーサル・データベース テキスト・エクステンダー 管理およびプログラミング* に説明されている TXTICRT コマンドを使用します。

## DMBIDROP

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

DB2 エクステンダー・インスタンスを除去します。

### 許可

SYSADM

UNIX では root 権限が必要です。

### コマンド構文




### コマンド・パラメーター

**instanceName**
除去したい DB2 エクステンダー・インスタンスの名前。

### 例

DEVINST という名前の DB2 エクステンダー・サーバー・インスタンスを削除するには、次のようにします。

```
dmbidrop  devinst
```

### 使用上の注意

このコマンドを実行する前に、以下の操作が必要です。

- このインスタンスを使用するすべてのアプリケーション、およびすべての db2ext コマンド行プロセッサーを停止する。
- エクステンダー・サービスを停止する。

DMBIDROP コマンドは DB2 エクステンダー・インスタンス・ディレクトリーを削除し、インスタンスのリストからこのインスタンス項目を削除し、さらにインスタンスに関するその他の情報も削除します。

DMBIDROP コマンドは DB2 エクステンダー・インスタンスだけを削除します。それに関連した DB2 インスタンスは削除しません。 DB2 インスタンスは明示的に除去する必要があります。

DB2 エクステンダー・サーバー・インスタンスに関連付けられた DB2 インスタンスを除去しても、DB2 エクステンダー・サーバー・インスタンスは除去されません。しかし、そのインスタンスを使用することはできません。

**(EEE のみ)** DB2 エクステンダー・インスタンスを削除する場合には、インスタンスの開始ポートおよび終了ポートの項目を、/etc/services ファイル (UNIX の場合) から、または ¥WINNT¥system32¥drivers¥etc¥Services ファイル (Windows の場合) からそれぞれ削除しなければなりません。削除すべき項目は、DMB_*instanceName*mpp、および DMB_*instanceName*mpp_END です。

## DMBILIST

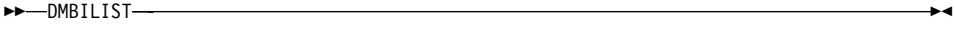
| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

DB2 エクステンダーのすべてのインスタンスをリストします。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

なし。

### 例

DB2 エクステンダー・インスタンスをリストするには、次のようにします。

```
dmbilist
```

## DMBIMIGR

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

**(UNIX のみ)** DB2 エクステンダー・インスタンスを、以前のリリースから現行リリースへ移行します。

### 許可

root 権限が必要です。

### コマンド構文




### コマンド・パラメーター

**instanceName**
移行したい DB2 エクステンダー・インスタンスの名前。

### 例

OLDINST という名前の DB2 エクステンダー・インスタンスを移行するには、次のようにします。

```
dmbimigr  oldinst
```

### 使用上の注意

このコマンドを実行する前に、以下の操作が必要です。

- DB2 エクステンダーの現行リリースをインストールしてある。
- 関連した DB2 インスタンスを移行する。

それぞれの DB2 エクステンダー・インスタンスごとに DMBIMIGR を 1 回実行します。各インスタンスをリストするには、DMBILIST を使用します。

## DMBSTART

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

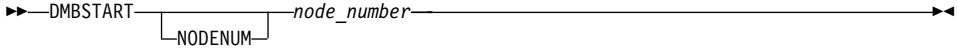
エクステンダー・インスタンスのすべてのエクステンダー・サービスを開始します。

**EEE のみ:** ノードが指定されている場合は、このコマンドを実行すると、そのノードだけでエクステンダー・サービスが開始されます。 DMBSTART は各ノードにおいて、「ノードの作成 / 削除 (Node Create/Drop)」機能も開始します。「ノードの作成 / 削除 (Node Create/Drop)」機能は、DB2 に定義されたノードが、エクステンダーに定義されたノードと一致するかどうかを確認します。一致しない場合、この機能は必要に応じてノードを追加したり削除したりします。

### 許可

SYSADM

### コマンド構文




### コマンド・パラメーター

**node_number**
エクステンダー・サービスを開始するノード。 **(EEE のみ)**

### 例

エクステンダー・サービスを開始するには、次のようにします。

```
dmbstart
```

ノード番号 2 でエクステンダー・サービスを開始するには、次のようにします。

```
dmbstart nodenum 2
```

### 使用上の注意

このコマンドは、次の要領で実行してください。

- (AIX、HP-UX、または Solaris では) インスタンス所有者としてログオンしているときに実行する。
- (OS/2 または Windows では) DB2INSTANCE 環境変数が、開始するインスタンスと同じになっているウィンドウから実行する。
- サーバー・システムをシャットダウンして再始動したときには必ず実行する。

単一区画環境では、DB2 インスタンスが実行されていなければ、DMBSTART はそのインスタンスも開始します。

**EEE のみ:**

複数区画環境では、DMBSTART は DB2 インスタンスを開始しません。区画環境で DMBSTART を実行する前に、DB2 を開始しておく必要があります。

DMBSTART が失敗した場合には、次の検査を行ってください。

- DBD2MMDATAPATH 変数の値が正しいことを確認する。
- 変数内の共用ディレクトリーまたは共用ファイル・システムが存在し、すべてのノードからアクセス可能であることを確認する。

## DMBSTAT

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

使用可能になっているデータベースを表示し、それらのデータベースのエクステンダー・サービスが起動され実行されているかどうかを示します。

### 許可

なし。

### コマンド構文




### コマンド・パラメーター

なし。

### 例

エクステンダー・サービスの状況を表示するには、次のようにします。

```
dmbstat
```

## DMBSTOP

| イメージ | オーディオ | ビデオ |
|---|---|---|
| O | O | O |

エクステンダー・インスタンスのためのエクステンダー・サービスを停止します。

**EEE のみ:** ノードを指定した場合、DMBSTOP はそのノードでのみエクステンダー・サービスを停止します。

### 許可

SYSADM

### コマンド構文


### コマンド・パラメーター

**node_number**
エクステンダー・サービスを停止するノード。 **(EEE のみ)**

### 例

エクステンダー・サービスを停止するには、次のようにします。

```
dmbstop
```

ノード番号 2 でエクステンダー・サービスを停止するには、次のようにします。

```
dmbstop nodenum 2
```

### 使用上の注意

このコマンドは、次の要領で実行してください。

- (AIX、HP-UX、または Solaris では) インスタンス所有者としてログオンしているときに実行する。
- (OS/2 または Windows では) DB2INSTANCE 環境変数が、停止するインスタンスと同じになっているウィンドウから実行する。

DMBSTOP を実行しても、DB2 インスタンスは停止しません。

**EEE のみ:** 保守モードで作動しているとき以外には、特定のノードで DMBSTOP を実行しないでください。さらに、ノードがオフになっているときには、そのノードでエクステンダー活動が起動されないようにする必要があります。こうしないと、予期しない動作が発生する可能性があります。

# 第19章 診断情報

DB2 エクステンダー UDF を呼び出すステートメントを含め、プログラムに組み込まれたすべての SQL ステートメントおよび DB2 CLI 呼び出しは、その組み込み SQL ステートメントまたは DB2 CLI 呼び出しが正常に行われたかどうかを示すコードを生成します。また、管理 API など、他の DB2 エクステンダー API も、処理が正常に行われたかどうかを示すコードを戻します。プログラムでは、これらの戻りコードを検査し、応答する必要があります。

また、プログラムでこれらのコードの補足情報を取り出すこともできます。この情報には、SQLSTATE 情報とエラー・メッセージがあります。この診断情報を使って、プログラム内で発生した問題を検出し修正することができます。

問題の発生源を診断することが容易でない場合もあります。このような場合、サービス技術員に情報を提供して、問題を検出し修正してもらわなければならないこともあります。 DB2 エクステンダーには、エクステンダーの活動を記録するトレース機能が付属しています。サービス技術員にとって、トレース情報は貴重な情報源です。トレース機能を使用するのは、IBM サービス技術員の指示があった場合だけにしてください。

この章では、このような診断情報にアクセスする方法について記載します。次の点について説明します。

- DB2 エクステンダー UDF 戻りコードと API 戻りコードを処理する方法。
- トレース機能を制御する方法。

また、エクステンダーが戻す可能性のある SQLSTATE とエラー・メッセージのリストを記載し、説明します。

## UDF 戻りコードの処理

組み込み SQL ステートメントは、SQLCA 構造の SQLCODE、SQLWARN、および SQLSTATE フィールドにコードを戻します。この構造は、SQLCA インクルード・ファイルに定義されています。 (SQLCA 構造体および SQLCA INCLUDE ファイルの詳細は、 *DB2 アプリケーション開発の手引き* を参照してください。)

DB2 CLI 呼び出しは SQLCODE 値と SQLSTATE 値を戻します。これらの値は、SQLError 関数を使用して取り出すことができます。 (SQLError 関数を使

ってエラー情報を取り出す方法の詳細は、 *コール・レベル・インターフェースの手引きおよび解説書* を参照してください。)

SQLCODE 値が 0 であれば、ステートメントが正常に実行されたことを意味します (ただし、警告条件が検出されている可能性もあります)。 SQLCODE 値が正であれば、ステートメントが正常に実行されたものの、警告が発生したことを意味します。 (組み込み SQL ステートメントは、0 または正の SQLCODE 値に関連した警告を SQLWARN フィールドに戻します。) SQLCODE 値が負であれば、エラー状態が発生したことを意味します。

DB2 は、メッセージをそれぞれの SQLCODE 値に関連付けます。 DB2 エクステンダー UDF が警告またはエラー状態を検出した場合は、DB2 に関連情報を渡して、この情報が SQLCODE メッセージに含まれるようにします。

SQLSTATE 値には、SQLCODE メッセージを補足するコードが含まれます。 DB2 エクステンダーが戻すそれぞれの SQLSTATE コードについては、569ページの『SQLSTATE コード』を参照してください。

DB2 エクステンダー UDF を呼び出す組み込み SQL ステートメントおよび DB2 CLI 呼び出しは、それらの UDF に固有の SQLCODE メッセージと SQLSTATE 値を戻す場合がありますが、DB2 は他の組み込み SQL ステートメントまたは他の DB2 CLI 呼び出しの場合と同じ方法でこれらの値を戻します。したがって、これらの値にアクセスする方法は、DB2 エクステンダー UDF を開始しない組み込み SQL ステートメントまたは DB2 CLI 呼び出しの場合と同じです。

エクステンダーが戻す可能性のある関連メッセージの SQLSTATE 値とメッセージ番号については、569ページの『SQLSTATE コード』を参照してください。それぞれのメッセージの詳細については、574ページの『メッセージ』を参照してください。

## API 戻りコードの処理

それぞれの DB2 エクステンダー API 呼び出しは、コードを戻します。戻りコード 0 は、API 呼び出しが正常に処理されたことを示します。 0 以外の戻りコードは、API 呼び出しが正常に処理されたものの警告条件が検出されたか、またはエラー状態が原因で正常に処理できなかったことを示します。

285ページの『第16章 アプリケーション・プログラミング・インターフェース』に、DB2 エクステンダー API の記号値のリストと、DB2 エクステンダー API が戻す可能性のあるそれぞれのコードの説明を記載します。

API が検出するエラーについては、追加情報を取り出すことができます。この追加情報を取り出すには、DB*x*GetError API を使用します。ここで、*x* はオーディオ・エクステンダーでは a、イメージ・エクステンダーでは i、そしてビデオ・エクステンダーでは v です。 DB*x*GetError API は、エラーを検出した最後の DB2 エクステンダー API の SQL エラー・コードと関連メッセージを戻します。 SQL エラー・コードの詳細については、 *DB2 メッセージ解説書* を参照してください。 DB*x*GetError API が戻す可能性のあるそれぞれのメッセージの詳細については、574ページの『メッセージ』 を参照してください。

たとえば、C アプリケーション・プログラムの次のステートメントによって、ある表がオーディオ・エクステンダーについて使用可能になり、続いてエラー検査が行われます。

```
rc=DBaEnableTable((char *)NULL, "employee");
rc=DBaGetError(&errCode, &errMsg);
```

## SQLSTATE コード

表16 に、DB2 エクステンダーが戻す可能性のある SQLSTATE 値のリストと説明を記載します。それぞれの SQLSTATE 値の説明には、記号の説明も含まれます。また、この表には、それぞれの SQLSTATE 値に関連するメッセージ番号のリストも示します。それぞれのメッセージの詳細については、574ページの『メッセージ』 を参照してください。

*表 16. SQLSTATE コードと関連するメッセージ番号*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 00000 | | MMDB_SQLSTATE_OK<br>正常完了。 |
| 01H01 | DMB0211W | MMDB_SQLSTATE_WARN_NO_OVERWRITE<br>ファイルは上書きされません。 |
| 38A00 | DMB0101E | MMDB_SQLSTATE_AUDIO_NULL_PARM<br>UDF への入力パラメーターとしてヌルは許可されていません。 |
| 38A02 | DMB0209E | MMDB_SQLSTATE_AUDIO_OPEN_HDR_ERROR<br>オーディオ・ファイル・ヘッダーのオープン時にエラーが発生しました。 |
| 38A03 | DMB0209E | MMDB_SQLSTATE_AUDIO_BAD_WAVE_HDR<br>無効な wave ファイルが提供されました。 |
| 38V00 | DMB0101E | MMDB_SQLSTATE_VIDEO_NULL_PARM<br>UDF への入力パラメーターとしてヌルは許可されていません。 |
| 38V02 | DMB0051E | MMDB_SQLSTATE_VIDEO_OPEN_HDR_ERROR<br>ビデオ・ファイル・ヘッダーのオープン時にエラーが発生しました。 |

*表 16. SQLSTATE コードと関連するメッセージ番号 (続き)*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 38V03 | DMB0105E | MMDB_SQLSTATE_VIDEO_BAD_MPEG1_HDR<br>無効な mpeg1 ファイルが提供されました。 |
| 38V04 | DMB0104E | MMDB_SQLSTATE_VIDEO_BLOB_TOO_SHORT<br>提供されたビデオ・バッファーが小さすぎます。 |
| 38V05 | DMB0106E | MMDB_SQLSTATE_VIDEO_BAD_AVI_HDR<br>無効な AVI ファイルが提供されました。 |
| 38V07 | DMB0106E | MMDB_SQLSTATE_VIDEO_BAD_QT_HDR<br>無効な Quicktime ファイルが提供されました。 |
| 38600 | DMB0075E<br>DMB0101E<br>DMB0102E<br>DMB0103E<br>DMB0210E | MMDB_SQLSTATE_INVALID_INPUT<br>UDF への入力パラメーターが無効です。 |
| 38601 | DMB0009E | MMDB_SQLSTATE_MALLOC_FAIL<br>メモリー割り振りが失敗しました。 |
| 38602 | DMB0386E | MMDB_SQLSTATE_CANNOT_COLLOCATE<br>ユーザーのデータを連結できません。 |
| 38603 | DMB0077E | MMDB_SQLSTATE_READ_FILE_FAIL<br>ファイルから読み取れません。 |
| 38604 | DMB0080E | MMDB_SQLSTATE_WRITE_FILE_FAIL<br>ファイルに書き込めません。 |
| 38610 | DMB0070E | MMDB_SQLSTATE_INVALID_HANDLE<br>メディア列に無効なデータが入っています。 |
| 38611 | DMB0073E | MMDB_SQLSTATE_INVALID_SESSION_HANDLE<br>UDF セッション・ハンドルが無効です。 |
| 38612 | DMB0074E | MMDB_SQLSTATE_INVALID_STATEMENT_HANDLE<br>UDF ステートメント・ハンドルが無効です。 |
| 38613 | DMB0083E | MMDB_SQLSTATE_INVALID_IMPORT_REQUEST<br>インポート要求が無効です。 |
| 38615 | DMB0071E | MMDB_SQLSTATE_CONNECT_FAIL<br>データベースへの接続時にエラーが発生しました。 |
| 38617 | DMB0071E | MMDB_SQLSTATE_ALLOC_STMT_FAIL<br>新しいステートメント・ハンドルの割り振りの際にエラーが発生しました。 |
| 38618 | DMB0208E<br>DMB0138E | MMDB_SQLSTATE_FREE_STMT_FAIL<br>ステートメントの解放時にエラーが発生しました。 |

*表 16. SQLSTATE コードと関連するメッセージ番号 (続き)*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 38619 | DMB0208E<br>DMB0132E | MMDB_SQLSTATE_BIND_FAIL<br>バインド時にエラーが発生しました。 |
| 38620 | DMB0208E | MMDB_SQLSTATE_BIND_COLUMN_FAIL<br>列のバインド時にエラーが発生しました。 |
| 38621 | DMB0208E | MMDB_SQLSTATE_BIND_FILE_FAIL<br>ファイルのバインド時にエラーが発生しました。 |
| 38622 | DMB0208E<br>DMB0132E | MMDB_SQLSTATE_SET_PARAM_FAIL<br>パラメーターの設定時にエラーが発生しました。 |
| 38623 | DMB0208E<br>DMB0131E | MMDB_SQLSTATE_PREPARE_FAIL<br>SQL ステートメントの作成時にエラーが発生しました。 |
| 38624 | DMB0208E<br>DMB0133E<br>DMB0172E | MMDB_SQLSTATE_EXECUTE_FAIL<br>ステートメントの実行時にエラーが発生しました。 |
| 38625 | DMB0208E<br>DMB0133E | MMDB_SQLSTATE_EXEC_DIRECT_FAIL<br>UDF で SQL ステートメントを直接実行しているときにエラーが発生しました。 |
| 38626 | DMB0208E<br>DMB0133E | MMDB_SQLSTATE_FETCH_FAIL<br>データの次の行の検索時にエラーが発生しました。 |
| 38627 | DMB0208E | MMDB_SQLSTATE_COMMIT_FAIL<br>トランザクションのコミット時にエラーが発生しました。 |
| 38628 | DMB0208E | MMDB_SQLSTATE_GET_LENGTH_FAIL<br>ストリング値の長さの検索時にエラーが発生しました。 |
| 38629 | DMB0208E | MMDB_SQLSTATE_GET_SUBSTRING_FAIL<br>ストリング値の一部分の検索時にエラーが発生しました。 |
| 38650 | DMB0077E<br>DMB0079E | MMDB_SQLSTATE_COPY_BLOB_2_FILE_FAIL<br>BLOB をファイルに複写している時にエラーが発生しました。 |
| 38651 | DMB0086E | MMDB_SQLSTATE_BLOB_BUFFER_TOO_SMALL<br>提供されたバッファーが小さすぎます。 |
| 38652 | DMB0082E | MMDB_SQLSTATE_BUILD_HANDLE<br>メディア列データの構成時にエラーが発生しました。 |
| 38653 | DMB0083E | MMDB_SQLSTATE_INVALID_INSERT_VIA_SELECT<br>選択を介した挿入要求が無効です。 |
| 38654 | DMB0081E | MMDB_SQLSTATE_INVALID_OFFSET_SIZE<br>オフセット・サイズが無効です。 |
| 38655 | DMB0068E | MMDB_SQLSTATE_METATABLE_DOESNOT_EXIST<br>要求されたメタデータ表は存在していません。 |

*表 16. SQLSTATE コードと関連するメッセージ番号 (続き)*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 38670 | DMB0134E<br>DMB0103E | MMDB_SQLSTATE_UNKNOWN_FORMAT<br>保管されているメディアの形式が不明です。 |
| 38671 | DMB0135E | MMDB_SQLSTATE_CREATE_THUMBNAIL_FAIL<br>サムネールの作成時にエラーが発生しました。 |
| 38672 | DMB0114E | MMDB_SQLSTATE_FORMAT_CONVERSION_FAIL<br>ファイル形式の変換時にエラーが発生しました。 |
| 38673 | DMB0363E | MMDB_SQLSTATE_INVALID_UPDATE<br>表を参照しないで UDF の更新が呼び出された時にエラーが発生しました。 |
| 38674 | DMB0361E | MMDB_SQLSTATE_NOT_ENABLED<br>エクステンダーについて使用可能になっていない列に UDF のインポートが適用された時に、エラーが発生しました。 |
| 38675 | DMB0129E | MMDB_SQLSTATE_VIDEO_INTERNAL<br>ビデオ・エクステンダー UDF で内部エラーが発生しました。 |
| 38676 | DMB0129E | MMDB_SQLSTATE_AUDIO_INTERNAL<br>オーディオ・エクステンダー UDF で内部エラーが発生しました。 |
| 38677 | DMB0129E | MMDB_SQLSTATE_IMAGE_INTERNAL イメージ・エクステンダー UDF で内部エラーが発生しました。 |
| 38678 | DMB0089E<br>DMB0208E | MMDB_SQLSTATE_BASE_INTERNAL_ERROR<br>共通層で内部エラーが発生しました。 |
| 38681 | DMB0108E | MMDB_SQLSTATE_IMPORT_ENV_NOT_SETUP<br>インポートの環境変数の設定が無効です。 |
| 38682 | DMB0111E | MMDB_SQLSTATE_STORE_ENV_NOT_SETUP<br>保管操作の環境変数の設定が無効です。 |
| 38683 | DMB0107E | MMDB_SQLSTATE_EXPORT_ENV_NOT_SETUP<br>エクスポート操作の環境変数の設定が無効です。 |
| 38684 | DMB0117E | MMDB_SQLSTATE_TEMP_ENV_NOT_SETUP<br>一時ファイル作成の環境変数の設定が無効です。 |
| 38686 | DMB0109E | MMDB_SQLSTATE_CANT_RESOLVE_IMPORT_FILE<br>インポート・ファイル名の解決時にエラーが発生しました。 |
| 38687 | DMB0112E | MMDB_SQLSTATE_CANT_RESOLVE_STORE_FILE<br>保管ファイル名の解決時にエラーが発生しました。 |
| 38688 | DMB0110E | MMDB_SQLSTATE_CANT_RESOLVE_EXPORT_FILE<br>エクスポート・ファイル名の解決時にエラーが発生しました。 |
| 38689 | DMB0116E | MMDB_SQLSTATE_CANT_CREATE_TMP_FILE<br>一時ファイルの作成時にエラーが発生しました。 |

*表 16. SQLSTATE コードと関連するメッセージ番号 (続き)*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 38690 | DMB0076E | MMDB_SQLSTATE_OPEN_IMPORT_FILE_FAIL<br>インポート・ファイルをオープンできません。 |
| 38691 | DMB0115E | MMDB_SQLSTATE_OPEN_STORE_FILE_FAIL<br>保管ファイルをオープンできません。 |
| 38692 | DMB0114E | MMDB_SQLSTATE_OPEN_EXPORT_FILE_FAIL<br>エクスポート・ファイルをオープンできません。 |
| 38693 | DMB0118E | MMDB_SQLSTATE_OPEN_TEMP_FILE_FAIL<br>一時ファイルをオープンできません。 |
| 38694 | DMB0117E | MMDB_SQLSTATE_OPEN_CONTENT_FILE_FAIL<br>内容ファイルをオープンできません。 |
| 38695 | DMB0135E | MMDB_SQLSTATE_OPEN_THUMBNAIL_FILE_FAIL<br>サムネール・ファイルをオープンできません。 |
| 38696 | DMB0135E | MMDB_SQLSTATE_READ_THUMBNAIL_FILE_FAIL<br>サムネール・ファイルを読み取れません。 |
| 38697 | DMB0207E | MMDB_SQLSTATE_OVERWRITE_NOT_ALLOWED<br>上書き操作を実行できませんでした。 |
| 38699 | DMB0171E | MMDB_SQLSTATE_QUERY_NAME_NOT_FOUND<br>その名前の照会が見つかりません。 |
| 38700 | | MMDB_SQLSTATE_NO_MANAGEBLOB |
| 38701 | | MMDB_SQLSTATE_UDFLOCATOR_FAIL |
| 38702 | | MMDB_SQLSTATE_SQL_FAIL |
| 38703 | | MMDB_SQLSTATE_INVALID_UPDATE |
| 38704 | | MMDB_SQLSTATE_NOT_ENABLED |
| 38705 | DMB0366E<br>DMB0382E | MMDB_SQLSTATE_QBIC_QUERY_FAIL_TO_BUILD<br>照会の作成中に障害が発生しました。 |
| 38706 | DMB0205E | MMDB_SQLSTATE_QBIC_TABLE_COLUMN_PAIR_NOT_VALID<br>QBIC カタログにアクセスしようとした時に障害が発生しました。<br>イメージ・ハンドルがカタログに見つからないか、または表名と列名の組み合わせがカタログに登録されていません。 |
| 38707 | DMB0383E | MMDB_SQLSTATE_QBIC_QUERY_EXECUTE_FAILED<br>照会の実行中に障害が発生しました。 |
| 38708 | | MMDB_SQLSTATE_QBIC_UKNOWN_ERROR<br>QBIC に不明な障害が発生しました。 |
| 38709 | DMB0208E | MMDB_COPY_FILE_TO_LOCATOR_FAILURE<br>ファイルを LOB ロケーターに複写する時に障害が発生しました。 |

## SQLSTATE

*表 16. SQLSTATE コードと関連するメッセージ番号 (続き)*

| SQLSTATE | メッセージ番号 | 説明 |
|---|---|---|
| 38710 | DMB0534E | MMDB_SQLSTATE_QBIC_UNSUPPORTED_UDF<br>この UDF はサポートされていません。 |

## メッセージ

**DMB0001E DB2 エクステンダー・サーバーが接続されませんでした。理由: ″<コード>″。**

**原因:** 試行された操作を実行するには、DB2 エクステンダー・サービスが実行中でなければなりません。

**処置:** サーバーのオペレーティング・システムのコマンド行で DMBSTART を実行してください。

**DMB0003W DB2 エクステンダー・トレース機能がこのセッションで実行中です。**

**原因:** トレース機能がシステム資源を使い尽くしています。

**処置:** システムのパフォーマンスに影響する場合は、トレース機能をオフにすることができます。

**DMB0004I このプログラムを実行できるのは、インスタンス所有者 ″<名前>″ だけです。**

**原因:** DB2 エクステンダー・サーバーは、インスタンスを作成したユーザー ID から起動する必要があります。

**処置:** インスタンスを作成したユーザー ID で DMBSTART コマンドを実行してください。

**DMB0005E 現行データベースは、″<エクステンダー名>″ エクステンダー用に使用可能になっていません。**

**原因:** データベースを特定の DB2 エクステンダーについて使用可能にしなければならない操作が試行されました。たとえば、ある表で DB2IMAGE データを使用できるようにするには、まずその表が保管されているデータベースで DB2IMAGE データを使用可能にする必要があります。

**処置:** 必要なエクステンダー・データ・タイプをデータベースで使用できるようにして、再試行してください。

**DMB0006E ユーザー ″<名前>″ はこの API を呼び出す権限を持っていません。**

**原因:** アプリケーション・プログラミング・インターフェースで求められているレベルの権限を持たないユーザー ID から、この API への呼び出しが試行されました。

**処置:** 別のユーザー ID からこのアプリケーションを実行するか、最初に試行したときのユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

**DMB0007E ユーザー表 ″<表名>″ は、エクステンダー ″<エクステンダー名>″ 用に使用できるようになっていません。**

**原因:** この操作を試みた表は、その DB2 エクステンダーについて使用可能になっていません。たとえば、表内の列をオーディオについて使用可能

にする前に、表そのものがオーディオ・データを保持できるようになっていなければなりません。

**処置:** まず、表をそのエクステンダーについて使用可能にしてください。その後、列を使用可能にします。

---

**DMB0008E ストアード・プロシージャー ″<名前>″ の実行時にエラーが発生しました。**

**原因:** メッセージで識別されたストアード・プロシージャーでエラーが発生したか、環境に問題があります。

**処置:** アプリケーションを検査してから再試行してください。

---

**DMB0009E メモリー割り振りエラーです。**

**原因:** システムは、試行された操作をサポートするために必要なメモリーを割り振ることができませんでした。

**処置:** システムにその操作を完了するだけの十分なメモリーがあることを確かめてください。

---

**DMB0010E ″<エクステンダー名>″ エクステンダーは、UDT ″<名前>″ に事前定義されています。**

**原因:** ユーザー定義タイプ (UDT) の名前は、別の DB2 エクステンダーに定義されている UDT ですでに使用されています。

**処置:** 別の UDT 名を選択してください。

---

**DMB0011E ユーザー列 ″<列名>″ を ″<エクステンダー名>″ エクステンダー用に使用可能にすることができません。このユーザー列の定義には、このエクステンダーに関連する特殊タイプ ″MMDBSYS.<名前>″ との互換性がありません。**

**原因:** 示されている列は、メッセージに表示されているデータ・タイプに定義されていないため、そのエクステンダーについて使用可能にできません。

**処置:** 使用可能にする列が、エクステンダーと同じデータ・タイプを使って定義されていることを確かめてください。

---

**DMB0012E 指定されたユーザー表 ″<表名>″ は存在しません。**

**原因:** 指定された名前の表は存在しません。

**処置:** 表名と、その表が存在するかどうかを検査してください。

---

**DMB0013E 列 ″<列名>″ は表 ″<表名>″ に定義されていません。**

**原因:** 試行された操作は、識別されている表に存在しない列名を参照しました。

**処置:** 表と列の名前を検査してください。

---

**DMB0014W ユーザー表 ″<表名>″ の列 ″<列名>″ は、″<エクステンダー名>″ エクステンダー用にすでに使用可能になっています。**

**原因:** その列がすでに使用可能になっているエクステンダーについて、列を使用可能にするための操作が試行されました。

**処置:** 処置は必要ありません。

**DMB0015W このデータベースはエクステンダー ″<エクステンダー名>″ 用にすでに使用可能になっています。**

**原因:** そのデータベースがすでに使用可能になっているエクステンダーについて、データベースを使用可能にするための操作が試行されました。

**処置:** 処置は必要ありません。

**DMB0016W ユーザー表 ″<表名>″ は、エクステンダー ″<エクステンダー名>″ 用にすでに使用可能になっています。**

**原因:** その表がすでに使用可能になっているエクステンダーについて、表を使用可能にするための操作が試行されました。

**処置:** 処置は必要ありません。

**DMB0017E ユーザー表 ″<表名>″ は、エクステンダー ″<エクステンダー名>″ 用にすでに使用可能になっています。しかし、少なくとも 1 つの関連したメタデータ表 ″<表名>″ あるいは ″<表名>″ が存在しません。**

**原因:** この表と関連する 1 つまたは複数の管理サポート (メタデータ) 表が破損または破棄されました。これらのメタデータ表がないと、ユーザー表でそのエクステンダーのタイプのデータを使用することはできません。

**処置:** ユーザー表をいったん使用不可にし、再度そのエクステンダーについて使用可能になるように再設定してください。

**DMB0018E システムは、表 ″<表名>″ にある列 ″<列名>″ のために固有のトリガー名を作成できません。**

**原因:** システムがユーザー表内の列を使用可能にしようとしたときに、DB2 エクステンダーが使用するトリガーの作成中にエラーが発生しました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、まずデータベース管理者、次いで IBM サービスに連絡してください。

**DMB0019I エクステンダー ″<エクステンダー名>″ の表 ″<表名>″ では、″<カウント>″ 個のファイルが参照されています。**

**原因:** このメッセージには、特定のエクステンダーのためのユーザー表が参照している外部メディア・ファイルの数が表示されます。

**処置:** 処置は必要ありません。

**DMB0020I エクステンダー ″<エクステンダー名>″ の表スキーマ ″<名前>″ が指定されている表では、″<カウント>″ 個のファイルが参照されています。**

**原因:** このメッセージには、特定のスキーマ名が指定されているユーザー表が参照している外部メディア・ファイルの数が表示されます。

**処置:** 処置は必要ありません。

**DMB0021I エクステンダー ″<エクステンダー名>″ の表 ″<表名>″ では、″<カウント>″ 個のアクセス不能ファイルが参照されています。**

**原因:** このメッセージには、特定のエクステンダーのユーザー表が参照している、アクセス不能な外部メディア・ファイルの数が表示されます。これらのファイルは、消去された可能性があります。

**処置:** 処置は必要ありません。

**DMB0022I エクステンダー ″<エクステンダー名>″ では、″<カウント>″ 個のアクセス不能ファイルが参照されています。**

**原因:** このメッセージには、次のような外部メディア・ファイルの数が表示されます。

- 現行データベース内のユーザー表が参照するファイル。
- 特定のエクステンダー・メディア・タイプのファイル (ビデオなど)。
- アクセス不能ファイル。たとえば、消去されたと思われるファイル。

**処置:** 処置は必要ありません。

---

**DMB0023I エクステンダー ″<エクステンダー名>″ の表スキーマ ″<名前>″ が指定されている表では、″<カウント>″ 個のアクセス不能ファイルが参照されています。**

**原因:** このメッセージには、特定のスキーマ名が指定されているユーザー表が参照している、アクセス不能な外部メディア・ファイルの数が表示されます。これらのファイルは、消去された可能性があります。このメッセージには、これらの表を使用できるエクステンダーの数も示されます。

**処置:** 処置は必要ありません。

---

**DMB0024I 現行データベースは、″<カウント>″ 個のエクステンダー用に使用可能になっています。**

**原因:** このメッセージには、現行データベースが使用可能になっている DB2 エクステンダーの数がリストで表示されます。

**処置:** 処置は必要ありません。

---

**DMB0025I 表 ″<表名>″ は、″<カウント>″ 個のエクステンダー用に使用可能になっています。**

**原因:** このメッセージには、示されている表が使用可能になっている DB2 エクステンダーの数がリストされます。

**処置:** 処置は必要ありません。

---

**DMB0026I 表 ″<表名>″ の列 ″<列名>″ は、″<カウント>″ 個のエクステンダー用に使用可能になっています。**

**原因:** このメッセージには、示されている列が使用可能になっている DB2 エクステンダーの数がリストされます。

**処置:** 処置は必要ありません。

---

**DMB0027I 現行データベースは ″<エクステンダー名>″ エクステンダー用に使用可能になっています。**

**原因:** このメッセージには、現行データベースが使用可能になっている DB2 エクステンダーが示されます。

**処置:** 処置は必要ありません。

---

**DMB0028I 表 ″<表名>″ は、″<エクステンダー名>″ エクステンダー用に使用可能になっています。**

**原因:** このメッセージには、ユーザー表で保持できるようになっているメディア・データ・タイプが示されます。

**処置:** 処置は必要ありません。

---

**DMB0029I 表 ″<表名>″ の列 ″<列名>″ は、″<エクステンダー名>″ エクステンダー用に使用可能になっています。**

**原因:** このメッセージには、ユーザー列で保持できるようになっているメディア・データ・タイプが示されます。

**処置:** 処置は必要ありません。

**DMB0030E　現行データベースを、″<エクステンダー名>″ エクステンダー用に使用可能にすることができません。 RC = ″<コード>″。**

**原因:** データベースが存在しないか、このデータベースを使用可能にする権限がありません。

**処置:** データベースが存在し、このデータベースを使用可能にする権限があることを確認してください。

**DMB0031E　表を、″<エクステンダー名>″ エクステンダー用に使用可能にすることができません。 RC = ″<コード>″。**

**原因:** データベースが存在しないか、表が使用可能になっていないか、またはこの表を使用可能にする権限がありません。

**処置:** データベースが存在し、データベースと表の両方がエクステンダーについて使用可能になっていることを確認してください。表を使用可能にする権限があることを確かめてください。

**DMB0032E　列を、″<エクステンダー名>″ エクステンダー用に使用可能にすることができません。 RC = ″<コード>″。**

**原因:** 列がこのエクステンダー用のデータ・タイプを使って定義されていないか、列が存在しないか、表が使用可能になっていないか、または列を使用可能にする権限がありません。

**処置:** 列が正しいデータ・タイプを使用して定義されていることを確認してください。表が使用可能になっており、その列を使用可能にする権限があることを確かめてください。

**DMB0033E　このコマンドを実行する権限がありません。**

**原因:** ユーザー ID の権限レベルは、このコマンドを実行するのに必要なレベルに達していません。

**処置:** 別のユーザー ID からこのコマンドを実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

**DMB0034I　″<データベース名>″ データベースに対して DB2 エクステンダー・サーバーが正常に起動されました。**

**原因:** 現行データベースに対してサーバーが正常に起動されました。

**処置:** 処置は必要ありません。

**DMB0035I　″<データベース名>″ データベースに対して DB2 エクステンダー・サーバーが停止しました。**

**原因:** 現行データベースに対してサーバーが正常に停止しました。

**処置:** 処置は必要ありません。

**DMB0036E　DB2 エクステンダー・サーバーを起動 / 停止できません。 DB2 エクステンダー・サーバーのデーモンが実行されていない可能性があります。データベース管理者に連絡してください。**

**原因:** DB2 エクステンダー・サーバーの起動 / 停止時にエラーが発生しました。 DB2 エクステンダー・サーバーのデーモンが実行されていない可能性があります。

**処置:** データベース管理者に連絡してください。

**DMB0037E　USE セッション・ハンドルが無効です。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0038E　USE ステートメント・ハンドルが無効です。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0039E　USE エラー: ″<エラー>″。**

**原因:** 内部エラーが起きました。

**処置:** 関連するエラー・メッセージに示されている指示に従い、操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0040E　SQL エラー: ″<エラー>″。**

**原因:** 内部エラーが起きました。

**処置:** 関連するエラー・メッセージに示されている指示に従い、操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0041W　現行データベースは、新しく指定された表スペースを使用して ″<エクステンダー名>″ エクステンダーで再度使用可能にされました。**

**原因:** 現行データベースが以前に使用可能にされたときに、異なる表スペースを使用していました。今回は、管理サポート表の新しい表スペースを使用してこのデータベースを使用可能にしました。

**処置:** 処置は必要ありません。

**DMB0042E　表 ″<表名>″ の列 ″<列名>″ は、″<エクステンダー名>″ エクステンダー用に使用可能になっていません。**

**原因:** 示されている列は、操作が試行されたエクステンダーについて使用可能になっていません。たとえば、示されているエクステンダーが現在使用可能になっていない列を使用不可にしようとした可能性があります。

**処置:** このメッセージに示されているエクステンダーについてその列が使用可能になっていることを確かめてください。

**DMB0043I　現行データベースは、″<エクステンダー名>″ エクステンダーの場合に使用不可になっています。**

**原因:** 使用不可操作が正常完了しました。

**処置:** 処置は必要ありません。

**DMB0044I　表 ″<表名>″ は、エクステンダー ″<エクステンダー名>″ の場合に使用不可になっています。**

**原因:** 使用不可操作が正常完了しました。

**処置:** 処置は必要ありません。

**DMB0045I　表 ″<表名>″ の列 ″<列名>″ は、″<エクステンダー名>″ エクステンダーの場合に使用不可になっています。**

**原因:** 使用不可操作が正常完了しました。

**処置:** 処置は必要ありません。

**DMB0046E　現行データベースを、″<エクステンダー名>″ エクステンダーの場合に使用不可にすることができません。 RC = ″<コード>″。**

**原因:** データベースが存在しないか、このエクステンダーについて使用可能になっていないか、またはこのデータベースを使用不可にする権限がありません。

**処置:** データベースが存在し、エクステンダーについて使用可能になっていることを確認してください。データベースを使用不可にする権限があることを確かめてください。

**DMB0047E　表を、″<エクステンダー名>″ エクステンダーの場合に使用不可にすることができません。 RC = ″<コード>″。**

**原因:** 表が存在しないか、このエクステンダーについて使用可能になっていないか、またはこの表を使用不可にする権限がありません。

**処置:** 表が存在し、エクステンダーについて使用可能になっていることを確認してください。表を使用不可にする権限があることを確かめてください。

**DMB0048E　列を、″<エクステンダー名>″ エクステンダーの場合に使用不可にすることができません。 RC = ″<コード>″。**

**原因:** 列がメッセージに示されたエクステンダーについて使用可能になっていないため、そのエクステンダー用に使用不可にすることはできません。

**処置:** エクステンダーの名前を確認し、ユーザー列が使用不可操作の対象になっているものかどうか確かめてください。

**DMB0049E　このコマンドを実行する権限がありません。**

**原因:** ユーザー ID の権限レベルは、このコマンドを実行するのに必要なレベルに達していません。

**処置:** 別のユーザー ID からこのアプリケーションを実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

**DMB0050E　表 ″<表名>″ に対して ″<権限レベル>″ 権限がありません。**

**原因:** この操作には、試行したユーザー ID の権限レベルよりも高い権限レベルが必要です。

**処置:** 正しい権限を持つユーザー ID から操作を実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

**DMB0051E　メディア・ファイル・ヘッダーが不良です。**

**原因:** システムは、このメディア・ファイルのヘッダーを読み取り / オープンできません。ファイルが損傷しているか、メディア・ファイルではありません。

**処置:** ファイルがメディア・ファイルであり、損傷していないことを確認してください。

**DMB0052I　″<データベース名>″ データベースに対して DB2 エクステンダー・サーバーが正常に起動されました。**

**原因:** サーバーが正常に起動されました。

**処置:** 処置は必要ありません。

**DMB0053I** ″<データベース名>″ データベース用の DB2 エクステンダー・サーバーが正常に停止しました。

**原因:** サーバーが正常に停止しました。

**処置:** 処置は必要ありません。

**DMB0054E** DB2 エクステンダー・サーバーがデータベースに接続できません。または DB2 ステートメント・ハンドルを割り振ることができません。″<データベース名>″ データベースに対して DB2 エクステンダー・サーバーが実行されていない可能性があります。

**原因:** DB2 エクステンダー・サーバーがデータベースに接続できません。または DB2 ステートメント・ハンドルを割り振ることができません。このデータベース用の DB2 エクステンダー・サーバーが実行されていない可能性があります。

**処置:** このデータベース用の DB2 エクステンダー・サーバーが実行されていることを確認してください。実行されていない場合は、このデータベースの特定のエクステンダー・サーバーを起動するか、システム管理者に依頼してエクステンダー・サービスを再起動してもらってください。

**DMB0055I** ″コマンド名″ コマンドは正常完了しました。

**原因:** コマンドは正常に終了しました。

**処置:** 処置は必要ありません。

**DMB0056E** ″<キーワード>″ の後に期待されていないトークン ″<トークン>″ が検出されました。期待されているトークンは、<エクステンダー名> です。

**原因:** このコマンドは、メッセージに示されたトークンではなく DB2 エクステンダーの名前を期待していました。

**処置:** コマンドの構文に従い、再試行してください。

**DMB0057E** 表スペース ″<表スペース名>″ が無効です。

**原因:** メッセージに示された表スペースは、存在しません。

**処置:** 表スペースの名前と、この表スペースが存在するかどうかを検査してください。

**DMB0058I** エクステンダー ″<エクステンダー名>″ は、″<カウント>″ 個のファイルを参照しています。

**原因:** このメッセージには、特定のエクステンダーが参照している外部メディア・ファイルの数が表示されます。

**処置:** 処置は必要ありません。

**DMB0059E** ″<名前>″ は DB2 エクステンダーの有効な名前ではありません。有効なエクステンダー名は ″<エクステンダー名>″ DB2VIDEO、DB2AUDIO、および DB2IMAGE です。

**原因:** エクステンダー名のスペルが間違っています。

**処置:** エクステンダー名を検査してください。

**DMB0060E　″<関数>″ の正しい構文は ″<構文>″ です。**

**原因:** 入力したコマンドの構文は間違っています。

**処置:** メッセージに説明されている構文に従ってください。

**DMB0061E　″<キーワード>″ の後の表名 ″<名前>″ は無効です。**

**原因:** このコマンドは表の名前を期待していました。

**処置:** コマンドの構文に従い、再試行してください。

**DMB0062E　″<キーワード>″ の後の列名 ″<名前>″ は無効です。**

**原因:** このコマンドは列の名前を期待していました。

**処置:** コマンドの構文に従い、再試行してください。

**DMB0064E　システムは、″<キーワード>″ の後のトークン ″<トークン>″ を認識しません。**

**原因:** このコマンドは、メッセージに示されているトークン以外のものを期待していました。

**処置:** コマンドの構文に従い、再試行してください。

**DMB0065E　″<キーワード>″ の後のユーザー ID ″<識別子>″ は無効です。**

**原因:** このコマンドは、有効なユーザー ID を期待していました。

**処置:** 必要なユーザー ID を検査して、再試行してください。

**DMB0066E　″<キーワード>″ の後のパスワード ″<パスワード>″ は無効です。**

**原因:** このコマンドは、メッセージに示されたトークンではなく有効なパスワードを期待していました。

**処置:** パスワードを検査して、再試行してください。

**DMB0067E　入力したコマンドは間違っています。**

**原因:** コマンド名のスペルまたは構文が間違っています。

**処置:** コマンドの構文に従い、再試行してください。

**DMB0068E　メタデータ表は存在していません。**

**原因:** この関数は、データ・オブジェクトに存在する必要のある管理サポート (メタデータ) 表を使用しようとしました。メタデータ表が損傷したか消去された可能性があります。

**処置:** 名前を検査し、メタデータ表の有無を確かめてください。メタデータ表が誤って消去 / 損傷した場合は、データ・オブジェクトをいったん使用不可にしてからもう一度使用可能にしてください。

**DMB0069E　DBname ″<名前>″ が無効です。**

**原因:** この名前のデータベースは存在しません。

**処置:** 名前を検査し、データベースの有無を確かめてください。

**DMB0070E　ハンドルが無効です。**

**原因:** アプリケーションに渡されたハンドル値は損傷している可能性があります。

**処置:** アプリケーションを検査して、エクステンダー・ハンドル値が変更されていないことを確認してください。

**DMB0071E　″<データベース名>″ に接続できません。**

**原因:** データベースの DB2 エクステンダー・サーバーが起動されていない可能性があります。

**処置:** サーバーの状況をチェックしてください。サーバーが実行中でない場合は、DMB コマンド行で START SERVER コマンドを使用して起動してください。

**DMB0072E　UDF SQL サーバーを DB から切断できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0073E　USE セッション・ハンドルが無効です。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0074E　USE ステートメント・ハンドルが無効です。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0075E　ファイル名を指定してください。**

**原因:** この操作には、メディア・ファイル名が必要です。

**処置:** メディア・ファイルの名前を入力してください。

**DMB0076E　インポート・ファイルをオープンできません。**

**原因:** インポート・ファイルは欠落しているか、損傷しています。

**処置:** インポート・ファイルの名前、およびそのファイルの有無を検査してください。

**DMB0077E　内容ファイルをオープン / 読み取りできません。**

**原因:** エクステンダー・ハンドルは、存在しないか破壊されているファイルを指しています。エクステンダーからこのファイルにアクセスすることはできなくなりました。

**処置:** FILENAME UDF を使用してファイルの名前を探すか、内容ファイルの有無を検査してください。

**DMB0078E　エクスポート・ファイルを作成できません。**

**原因:** エクスポート・ファイルは欠落しているか、破壊されています。

**処置:** エクスポート・ファイルの名前、およびそのファイルの有無を確かめてください。

**DMB0079E　ファイルに BLOB を複写できません。**

**原因:** このファイルは、BLOB を受け入れられません。 BLOB を記憶するための十分な記憶スペースがない可能性があります。

**処置:** BLOB のサイズと使用可能な記憶域を比較し、必要に応じて記憶域を増やしてください。

**DMB0080E　ファイルに書き込めません。**

**原因:** このファイルは損傷しているか、存在しないか、名前のスペルが間違っています。

**処置:** ファイルの名前、およびそのファイルの有無を確かめてください。

**DMB0081E　オフセットまたはサイズが無効です。**

**原因:** この操作では、データ構造の中に期待されているデータが見つかりませんでした。フィールドのサイズまたはオフセットが正しくありません。

**処置:** オフセットとフィールドのサイズを検査してください。

**DMB0082E　ハンドルを作成できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0083E　″<エクステンダー名>″ と ″<エクステンダー名>″ は非互換です。**

**原因:** メッセージに指定されている 2 つのエクステンダーは、この使用法では互換性がありません。全選択でも副選択でもこの挿入操作は無効です。

**処置:** ソース・オブジェクトで使用できるエクステンダーと同じものが、ターゲット・オブジェクトでも使用可能になっていることを確認してください。

**DMB0084E　インポート要求のファイル名、内容、記憶タイプが無効です。**

**原因:** インポート操作は失敗しました。ファイル名、内容、または記憶タイプが無効でした。

**処置:** データを検査して、再試行してください。

**DMB0085E　更新要求のファイル名、内容、記憶タイプが無効でした。**

**原因:** 更新操作は失敗しました。ファイル名、内容、または記憶タイプが無効でした。

**処置:** データを検査して、再試行してください。

**DMB0086E　要求されたサイズは大きすぎます。**

**原因:** 要求されたサイズは、UDF の最大 blob サイズより大きいサイズです。

**処置:** サイズを小さくして要求してください。

**DMB0087E　ファイル名が無効です。**

**原因:** この名前のファイルはありません。

**処置:** ファイルの名前、およびそのファイルの有無を確かめてください。

**DMB0088E　ハンドル値が NULL になっています。**

**原因:** UDF は非空文字のハンドルを期待していました。

**処置:** アプリケーションのハンドルが有効で、そのハンドルが UDF に渡されていることを確かめてください。

**DMB0089E　ハンドル値が存在しません。**

**原因:** UDF に渡されたハンドルが無効です。

**処置:** アプリケーションが有効なハンドルを渡していることを確認してください。

**DMB0090E　データが切り捨てられました。**

**原因:** ファイルまたはバッファーが小さすぎて、データを受け入れられませんでした。

**処置:** ファイルまたはバッファーのサイズを大きくしてください。

**DMB0091W　ファイルにすでに内容が含まれています。**

**原因:** ファイルにはすでに内容が含まれています。この内容は上書きされます。

**処置:** 処置は必要ありません。

**DMB0092E　列 ″<列名>″ で試行された挿入操作が無効です。この列は、″<エクステンダー名>″ エクステンダー用に使用可能になっています。**

**原因:** 挿入するデータのデータ・タイプは、列が使用可能になっているエクステンダーと異なっています。

**処置:** ソース・オブジェクトを使用できるエクステンダーと同じものが、ターゲット・オブジェクトでも使用可能になっていることを確認してください。

**DMB0093E　列 ″<列名>″ で試行された更新操作が無効です。この列は、″<エクステンダー名>″ エクステンダー用に使用可能になっています。**

**原因:** 更新中のデータのデータ・タイプは、列が使用可能になっているエクステンダーと異なっています。

**処置:** ソース・オブジェクトで使用できるエクステンダーと同じものが、ターゲット・オブジェクトでも使用可能になっていることを確認してください。

**DMB0094I　表 ″<表名>″ が存在しません。**

**原因:** システムは、その名前の表を検出できません。別のデータベースに存在する可能性があります。

**処置:** 処置は必要ありません。

**DMB0095W　表 ″<表名>″ は、エクステンダー ″<エクステンダー名>″ の場合に使用可能になっていません。**

**原因:** この表は、このエクステンダーについて使用可能になっていません。

**処置:** 処置は必要ありません。

**DMB0096W　表 ″<表名>″ の列 ″<列名>″ は、″<エクステンダー名>″ エクステンダーの場合に使用可能になっていませんでした。**

**原因:** システムは、列が使用可能になっていると期待していました。

**処置:** 処置は必要ありません。

**DMB0097W　現行データベースは、″<エクステンダー名>″ エクステンダー用に使用可能になっていません。**

**原因:** システムは、データベースが使用可能になっていると期待していました。

**処置:** データベースをこのメッセージに示されているエクステンダーについて使用可能にしてください。

**DMB0098E　ユーザーは、表 ″<表名>″ に対する ″<権限レベル>″ 権限を持っていません。**

**原因:** この操作には、試行したユーザー ID の権限レベルよりも高い権限レベルが必要です。

**処置:** 表を所有するユーザー ID から操作を実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

**DMB0099E　トランザクションをコミットできません。**

**原因:** 現行データベースのエクステンダー・サーバーが停止した可能性があります。

**処置:** サーバーの状況をチェックしてください。サーバーが実行中でない場合は、db2ext コマンド行で START SERVER コマンドを使用して起動してください。

**DMB0100E　″<名前>″ は有効な表名ではありません。**

**原因:** その名前の表は存在しません。

**処置:** 表の名前と有無を確かめて、再試行してください。

**DMB0101E　NULL パラメーターは無効です。**

**原因:** コマンドは非空文字のパラメーターを期待していました。

**処置:** 構文を検査して、再試行してください。

**DMB0102E　記憶タイプが無効です。**

**原因:** DB2 エクステンダーでは、記憶タイプはメディア・データが記憶される場所を指定します。

**処置:** 外部 (ファイルに) を指示する場合は 0、外部 (データベースに) を指示する場合は 1 を指定してください。

**DMB0103E　この形式はサポートされていません。**

**原因:** DB2 エクステンダーはこのオブジェクトの形式をサポートしていません。

**処置:** オブジェクトをサポートされている形式に変換してください。

**DMB0104E　ビデオ内容バッファーが小さすぎます。**

**原因:** 割り振られたバッファーに対してビデオ・クリップが大きすぎます。

**処置:** もっと大きなバッファーを割り振ってください。

**DMB0105E　MPEG1 ヘッダーが無効です。**

**原因:** MPEG1 ファイルのヘッダーは欠落しているか破壊されています。

**処置:** ファイルが MPEG1 ファイルであることを確認してください。

**DMB0106E　AVI ヘッダーが無効です。**

**原因:** AVI ファイルのヘッダーは欠落しているか破壊されています。

**処置:** ファイルが AVI ファイルであることを確認してください。

**DMB0107E　エクスポート環境が設定されていません。**

**原因:** DB2 エクステンダーに設定されているエクスポート環境の環境変数が正しくありません。

**処置:** 環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの設定になっていることを確認してください。

**DMB0108E　インポート環境が設定されていません。**

**原因:** DB2 エクステンダーに設定されているインポート環境の環境変数が正しくありません。

**処置:** 環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの設定になっていることを確認してください。

**DMB0109E　インポート・ファイルを解決できません。**

**原因:** この名前のインポート・ファイルはありません。

**処置:** ファイルの名前と有無、および環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの正しい設定になっていることを確認してください。

**DMB0110E　エクスポート・ファイルを解決できません。**

**原因:**　この名前のエクスポート・ファイルはありません。

**処置:**　ファイルの名前と有無、および環境変数が613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの正しい設定になっていることを確認してください。

**DMB0111E　保管環境が設定されていません。**

**原因:**　保管環境の環境変数が正しく設定されていません。

**処置:**　環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの設定になっていることを確認してください。

**DMB0112E　記憶ファイルを解決できません。**

**原因:**　この名前の記憶ファイルはありません。

**処置:**　ファイルの名前と有無、および環境変数が613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの正しい設定になっていることを確認してください。

**DMB0113E　インポート・ファイルをオープンできません。**

**原因:**　このファイルはだれか他の人によってロックされているか、欠落しているか、または破壊されている可能性があります。

**処置:**　ファイルの名前、有無、状況、および権限レベルを検査してください。

**DMB0114E　エクスポート・ファイルをオープンできません。**

**原因:**　このファイルはだれか他の人によってロックされているか、欠落しているか、または破壊されている可能性があります。

**処置:**　ファイルの名前、有無、状況、および権限レベルを検査してください。

**DMB0115E　記憶ファイルをオープンできません。**

**原因:**　システムはファイル書き込みを試行していますが、このファイルはすでに存在しています。サーバーにはファイルを上書きする権限がありません。

**処置:**　ファイルの名前、有無、状況、および権限レベルを検査してください。

**DMB0116E　一時ファイルを作成できません。**

**原因:**　一時ファイルを作成するだけの十分な記憶スペースがない可能性があります。

**処置:**　エクステンダーの一時環境変数が正しく設定されていることを確認してください。必要に応じて記憶域を増やしてください。

**DMB0117E　一時環境が設定されていません。**

**原因:**　一時環境の環境変数が正しく設定されていません。

**処置:**　環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの設定になっていることを確認してください。

**DMB0118E　一時ファイルをオープンできません。**

**原因:**　一時ファイルは上書きされたか損傷している可能性があります。

**処置:**　環境変数が 613ページの『付録A. DB2 エクステンダー用の環境変数の設定』に説明されているとおりの設定になっていることを確認してください。

**DMB0119I　dmbsrv サーバーが ″<カウント>″ 個の接続を持つ ″<名前>″ に対して起動中です。**

**原因:** このメッセージには、サーバー起動時の接続数が示されます。

**処置:** 処置は必要ありません。

**DMB0120E　dmbsrv サーバーは ″<カウント>″ 個の接続を持つ ″<名前>″ に対して起動しようとして失敗しました。**

**原因:** DB2 がまだ起動していないか、データベースが存在していないか、またはシステムがライセンスを取得している接続を使い尽くした可能性があります。

**処置:** DB2 が起動され、データベースが存在することを確認してください。問題が解決しない場合は、IBM に連絡してさらに多くのライセンスを取得してください。

**DMB0121I　dmbsrv サーバーが ″<カウント>″ 個の接続を持つ ″<名前>″ に対して起動されました。**

**原因:** このメッセージには、サーバー起動時の接続数が示されます。

**処置:** 処置は必要ありません。

**DMB0122I　dmbssd サーバーは作動可能です。**

**原因:** サーバーは、アプリケーションを実行する準備が完了しています。

**処置:** 処置は必要ありません。

**DMB0129E　″<操作名>″ 操作が無効です。**

**原因:** この名前のコマンドまたは API はありません。

**処置:** コマンドまたは API を検査して、再試行してください。

**DMB0130E　列 ″<列名>″ を SQL ステートメントにバインドできませんでした。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0131E　SQL 作成ステートメントが失敗しました。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0132E　SQL 設定パラメーターが失敗しました。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0133E　SQL 実行ステートメントが失敗しました。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0134E　ファイル形式の変換が失敗しました。**

**原因:** 保管されているマルチメディア・データの形式は形式変換でサポートされていません。

**処置:** このファイルの形式は変換できません。

**DMB0135E　サムネールをオープン / 読み取りできません。**

**原因:** サムネール・ファイルは欠落しているか損傷しています。

**処置:** サムネール・ファイルの名前、有無、および整合性を検査してください。

**DMB0136E　バインド・ファイルを検出できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0137E　DB ″<データベース名>″ に接続できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0138E　SQL ステートメントを解放できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0139E　″<キーワード>″ の後のフィーチャー名 ″<名前>″ が無効です。**

**原因:** イメージ・エクステンダーは、このコマンドの有効なフィーチャー名を期待していました。

**処置:** 有効なフィーチャー名を指定してコマンドを再試行してください。有効なフィーチャー名は、次のとおりです。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

**DMB0141E　″<キーワード>″ の後の修飾子 ″<修飾子>″ が無効です。**

**原因:** システムはコマンドの修飾子を識別できません。

**処置:** 修飾子を検査して、再試行してください。

**DMB0142E　オープンされたカタログはありませんでした。**

**原因:** DB2 エクステンダーでは、現行コマンドで QBIC カタログがオープンされている必要があります。

**処置:** OPEN QBIC CATALOG コマンドで QBIC カタログをオープンしてください。

**DMB0143I　表 ″<表名>″ の列 ″<列名>″ の QBIC カタログで、自動カタログ機能の設定が ″<状況>″ になっています。 ″<カウント>″ 個のフィーチャーがあります。**

**原因:** このメッセージには、特定の画像列の QBIC カタログに定義されているフィーチャーの数と、自動カタログ機能がオンになっているかどうかが示されます。

**処置:** 処置は必要ありません。

**DMB0145E　照会ハンドルが無効です。**

**原因:** API 呼び出しで使用されている照会ハンドルが無効です。

**処置:** アプリケーションを検査して、取得している照会ハンドルが正しいかどうか確認してください。

**DMB0146E　フィーチャー・クラス名 ″<フィーチャー・クラス>″ が無効です。**

**原因:** この名前のフィーチャー・クラスはありません。有効なフィーチャー名は、次のとおりです。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

**処置:** フィーチャーの名前を訂正して、再試行してください。

**DMB0147E　フィーチャー・クラス名 ″<フィーチャー・クラス>″ が欠落しているか、無効です。**

**原因:** 有効なフィーチャー名は、次のとおりです。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

**処置:** フィーチャーの名前を訂正して、再試行してください。

**DMB0148E　フィーチャー ″<フィーチャー名>″ は、すでに照会のメンバーになっています。**

**原因:** この照会は、メッセージに示されているフィーチャーをすでにサポートしています。

**処置:** 処置は必要ありません。

**DMB0149E　フィーチャー ″<フィーチャー名>″ は、照会のメンバーではありません。**

**原因:** この照会には、指定されたフィーチャー名は含まれていません。

**処置:** このフィーチャーにアクセスする他の API が呼び出される前に、このフィーチャーを照会に追加するには、QbQueryAddFeature API を使用してください。

**DMB0150E　システムはメモリーを割り振ることができません。**

**原因:** システムは、試行された操作をサポートするために必要なメモリーを割り振ることができませんでした。

**処置:** システムにその操作を完了するだけの十分なメモリーがあることを確かめてください。

**DMB0151E　戻り値を指すポインターが NULL になっています。**

**原因:** 戻り値を指すポインターとして NULL は無効なため、API 呼び出しは失敗しました。

**処置:** API 呼び出しに有効なパラメーターが提供されており、構文が正しいことを確認してください。

**DMB0152E　リスト戻り値を指すポインターが NULL になっています。**

**原因:** 戻り値を指すポインターとして NULL は無効なため、API 呼び出しは失敗しました。

**処置:** API 呼び出しに有効なパラメーターが提供されており、構文が正しいことを確認してください。

**DMB0153E　範囲パラメーターは予約されており、0 になっている必要があります。**

**原因:** このパラメーターは、将来の使用のため予約されています。

**処置:** 範囲を 0 に設定してください。

**DMB0154E　フィーチャー・クラス名を指すポインターが無効です。**

**原因:** この API 呼び出しは、入力フィーチャー・クラス名を指す有効なポインターを期待していました。

**処置:** API 呼び出しに有効なパラメーターが提供されており、構文が正しいことを確認してください。

**DMB0155I　″<関数名>″ 関数に、バッファー・サイズ 0 が渡されました。**

**原因:** API 呼び出しには、情報を戻すためのバッファーが必要です。

**処置:** 処置は必要ありません。

**DMB0156E　QbImageSource ポインターが NULL になっています。**

**原因:** NULL 値は、構造が変更禁止であることを示しています。

**処置:** 処置は必要ありません。

**DMB0157E　QbImageSource タイプ ″<タイプ>″ が無効です。**

**原因:** この DB2 エクステンダー API が参照するデータ構造のデータ・タイプは間違っています。

**処置:** 構造のデータ・タイプは QbImageSource でなければなりません。

**DMB0159E　QbImageSource 画像バッファーを指すポインターが NULL です。**

**原因:** この API 呼び出しは、ポインターが戻されることを期待していました。

**処置:** アプリケーションを検査して、API 呼び出しとバッファーが正しく指定されているかどうか確かめてください。

**DMB0160I　画像バッファーまたはファイルの長さがゼロになっています。**

**原因:** 長さがゼロです。

**処置:** 処置は必要ありません。

**DMB0161E　表または列あるいはその両方の名前を指すポインターが NULL になっています。**

**原因:** この API 呼び出しは、ポインターが提供されることを期待していました。

**処置:** アプリケーションを検査して、API 呼び出しへの入力が正しく指定されているかどうか確かめてください。

**DMB0162I　requestedHits をゼロに設定しました。**

**原因:** requestedHits をゼロに設定すると、何も戻されなくなります。

**処置:** 処置は必要ありません。

**DMB0163I　この関数はまだサポートされていません。**

**原因:** この関数はまだサポートされていません。

**処置:** 処置は必要ありません。

**DMB0164E　システムは照会 (<照会名>) を処理できません。**

**原因:** 照会が作成されたときにエラーが発生しました。

**処置:** コマンドまたは API への入力を検査して、再試行してください。

**DMB0165E　システムは照会 (<照会名>) を実行できません。**

**原因:** 照会が作成されたときにエラーが発生しました。

**処置:** コマンドまたは API への入力を検査して、再試行してください。

---

**DMB0166E ″<名前>″ を実行中に ″<名前>″ でステートメント・エラーが検出されました : ″<エラー>″。**

**原因:** 内部 IBM エラーが発生しました。

**処置:** データベース管理者に連絡してください。

---

**DMB0167E QbGenericImageDataClass の読み取り中にエラーが発生しました (<エラー>)。**

**原因:** 内部 IBM エラーが発生しました。

**処置:** データベース管理者に連絡してください。

---

**DMB0168E 照会のフィーチャー ″<フィーチャー名>″ が探索前に設定されていません。**

**原因:** 照会にフィーチャーが割り当てられていないので、照会が実行されません。

**処置:** QbAddFeature API か ADD QBIC FEATURE コマンドのどちらかを使用して照会にフィーチャーを追加してください。

---

**DMB0169E コール・レベル・インターフェースで次のエラーが発生しました : ″<エラー>″。**

**原因:** CLI エラーです。

**処置:** メッセージ・テキストの指示に従ってください。

---

**DMB0170E 照会名 ″<照会名>″ はすでに使用中です。**

**原因:** この名前の別の照会が存在します。

**処置:** 別の名前を選択してください。

---

**DMB0171E 照会名 ″<照会名>″ は保管されていませんでした。**

**原因:** システムは、照会を作成した後、保管できませんでした。

**処置:** 書き込み権限、および照会を保管する十分な記憶域があるかどうか確かめてください。

---

**DMB0172E SQL エラーです : ″<エラー>″。**

**原因:** 内部エラーが起きました。

**処置:** 関連するエラー・メッセージに示されている指示に従い、操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

---

**DMB0173E カタログはオープンされていますが、読み取り専用になっています : ″<カタログ名>″。**

**原因:** だれか他の人が書き込みモードですでにこのカタログをオープンしているか、書き込み権限がないため、このカタログを更新できません。

**処置:** 他のユーザーの作業が終わるまで待つか、別のユーザー ID からこのアプリケーションを実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

---

**DMB0174E システム・エラーが発生しました : ″<エラー>″。**

**原因:** 内部 IBM エラーが発生しました。

**処置:** 関連するエラー・メッセージに示されている指示に従い、操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

**DMB0175I　画像が検出されませんでした : ″<情報>″。**

**原因:** 照会と一致する画像が検出されません。データベースに何も入っていない可能性があります。

**処置:** 処置は必要ありません。

**DMB0176I　列にはすでに QBIC カタログがあります : ″<表名 列名>″。**

**原因:** この名前の別のカタログが存在します。

**処置:** 処置は必要ありません。

**DMB0177E　システムはカタログをオープンできません。エラー・メッセージは次のとおりです : ″<エラー>″。**

**原因:** このカタログは損傷しています。

**処置:** メッセージ・テキストの指示に従ってください。

**DMB0178E　システムはこのカタログを削除できません。エラー・メッセージは次のとおりです : ″<エラー>″。**

**原因:** カタログが存在しないか、損傷しています。

**処置:** カタログの名前と有無を確かめて、再試行してください。

**DMB0179E　カタログ・ハンドルが無効です : ″<エラー>″。**

**原因:** API 呼び出しで使用されているカタログ・ハンドルが無効です。

**処置:** アプリケーションを検査して、取得しようとしているカタログ・ハンドルが正しいかどうか確かめてください。

**DMB0180I　カタログへのアクセスが拒否されました : ″<エラー>″。**

**原因:** アクセスが拒否されました。

**処置:** 処置は必要ありません。

**DMB0181I　カタログは使用中です: ″<エラー>″**

**原因:** 別の操作でこのカタログを使用中です。

**処置:** 処置は必要ありません。

**DMB0184I　トレース機能がすでに開始されています。**

**原因:** トレース機能がすでに開始されています。

**処置:** 処置は必要ありません。

**DMB0185I　トレース機能はまだ開始されていません。**

**原因:** トレース機能はまだ開始されていません。

**処置:** 処置は必要ありません。

**DMB0186I　トレース機能は ″<ディレクトリー名>″ ディレクトリーで ″<時刻>″ にオンに設定されました。このトレース・ファイルは ″<ファイル名>″ です。 ″<バイト数>″ バイトのトレース・データが書き込まれました。**

**原因:** トレース機能はオンになっています。

**処置:** 処置は必要ありません。

**DMB0187E　システムがファイル ″<ファイル名>″ を書き込み用にオープンできないため、通信を確立できません。**

**原因:** 環境変数 DB2INSTANCE で記述されている現行インスタンスの所有者でないか、または DB2MMTOP などの環境変数が正しく設定されていません。

**処置:** インスタンスを所有するユーザー ID を使ってログに記録してください。環境変数が正しく設定されているかどうか確認してください。

---

**DMB0188I トレース・デーモンの作成時にエラーが発生しました : ″<エラー>″。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

---

**DMB0189I トレース機能がすでに正常に開始されています。**

**原因:** トレース機能がすでに開始されています。

**処置:** 処置は必要ありません。

---

**DMB0190E トレース機能を開始できません。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

---

**DMB0191E 環境変数 ″<名前>″ を設定する必要があります。**

**原因:** システム構成が正しくありません。

**処置:** 変数を設定して、再試行してください。

---

**DMB0192I トレース機能が正常にオフにされました。**

**原因:** トレース機能はオフになっています。

**処置:** 処置は必要ありません。

---

**DMB0193E システムはファイル ″<ファイル名>″ に書き込めません。**

**原因:** 指定されたファイルのディレクトリーへの書き込み権限を持っていません。

**処置:** データベース管理者に連絡して、権限を取得してください。

---

**DMB0194E システムはファイル ″<ファイル名>″ から読み取れません。**

**原因:** ファイルが存在していないか、ファイルの読み取り権限がありません。

**処置:** ファイルが存在し、このファイルの読み取り権限があることを確認してください。

---

**DMB0198E 入力ファイルのトレース・コード ″<コード>″ は不明です。入力ファイルは損傷している可能性があります。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発する場合は、IBM サービス員に連絡してください。

---

**DMB0199E 参照されているどの表に対しても ″<権限レベル>″ 権限がありません。**

**原因:** ユーザー ID の権限レベルは、この操作に必要とされているレベルに達していません。

**処置:** 別のユーザー ID からこの操作を実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

---

**DMB0200W 参照されている表のうち少なくとも 1 つに対して ″<権限レベル>″ 権限がありません。**

**原因:** ユーザー ID の権限レベルは、いくつかの表で必要とされているレベルに達していません。

参照先ファイルのリストを表示している場合、リストに示されているファイルは、SELECT 権限を持っている表に参照されています。システムの表の中で SELECT 権限を持っていない表がある場

合、それらの表が参照するファイルはリストに表示されません。

メタデータを再編成中の場合、システムが再編成するのは制御権限を持っている表のメタデータだけです。

**処置:** すべてのファイルを表示するには、別のユーザー ID からこの操作を実行するか、現行ユーザー ID の権限レベルをデータベース管理者に変更してもらってください。

---

**DMB0201I この名前のフィーチャーはすでに存在します : ″<フィーチャー名>″。**

**原因:** この名前のフィーチャーは、QBIC カタログにすでに含まれています。

**処置:** 処置は必要ありません。

---

**DMB0202E フィーチャー名が無効です : ″<フィーチャー名>″。**

**原因:** この名前のフィーチャー・クラスはありません。有効なフィーチャー名は、次のとおりです。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

**処置:** フィーチャーの名前を訂正して、再試行してください。

---

**DMB0203E フィーチャーが検出されませんでした : ″<フィーチャー名>″。**

**原因:** この名前のフィーチャー・クラスは存在しないか、QBIC カタログに含まれていません。有効なフィーチャー名は、次のとおりです。

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

**処置:** フィーチャーの名前を訂正して、再試行してください。

---

**DMB0204E 列 ″<列名>″ は DB2IMAGE で使用可能になっていません。**

**原因:** この列はイメージ・エクステンダーについて使用可能になっていません。

**処置:** この列が DB2 イメージ・エクステンダーについて使用可能になっていることを確かめてください。

---

**DMB0205E ″<表名 列名>″ にカタログは検出されませんでした。**

**原因:** 指定されている列と関連する QBIC カタログがありません。

**処置:** 他の QBIC 操作を実行する前に、この列の QBIC カタログを作成してください。

---

**DMB0206W 指定された列は、エクステンダーで使用可能になっていません。**

**原因:** この列は存在していないか、データ・タイプにエクステンダーとの互換がありません。

**処置:** 列が正しいデータ・タイプで定義されていることを確認してください。

---

**DMB0207E ファイルを重ね書きできません。**

**原因:** ファイルはすでに存在していますが、EXPORT UDF で上書きできません。

**処置:** ファイルを異なるファイル名にエクスポートするか、EXPORT UDF でファイルを上書きできるようにしてください。

---

**DMB0208E sqlcode=<コード> clistate=<コード>。**

**原因:** 内部エラーが起きました。

**処置:** 操作を繰り返してください。エラーが再発

する場合は、IBM サービス員に連絡してください。

---

**DMB0209E　音声ヘッダーが無効です。**

**原因:** オーディオ・ファイルのヘッダーが欠落しているか破壊されています。

**処置:** このオーディオ・ファイルの形式が DB2 エクステンダーでサポートされていることを確認してください。

---

**DMB0211W　ファイルが存在しており、重ね書きはされません。**

**原因:** 指定されているターゲット・ファイルはすでに存在しており、上書きされません。

**処置:** 処置は必要ありません。

---

**DMB0212E　resultType パラメーターは予約されており、0 になっている必要があります。**

**原因:** このパラメーターは、将来の使用のため予約されています。

**処置:** resultType を 0 に設定してください。

---

**DMB0214E　照会名を指すポインターが無効です。**

**原因:** この API 呼び出しは、入力照会名を指す有効なポインターを期待していました。

**処置:** API 呼び出しに有効なパラメーターが提供されており、構文が正しいことを確認してください。

---

**DMB0352E　コマンド行環境が初期設定されていません。**

**原因:** コマンド行環境が、 db2ext コマンド行プロセッサーを実行できるように初期設定されていません。 (このメッセージは Windows NT および Windows 95 環境にのみ適用されます。)

**処置:** DB2CLP ウィンドウをオープンする db2ext コマンドを出してから、そのウィンドウ内で db2ext コマンド行プロセッサーを実行する db2ext コマンドを出してください。

---

**DMB0353E　db2ext コマンド行プロセッサーのバックグラウンド・プロセスと通信できません。**

**原因:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスが実行中ですが、db2ext コマンド行プロセッサーがそのプロセスと通信できません。

**処置:** その db2ext コマンドを別のウィンドウで試みてください。

---

**DMB0354E　db2ext コマンド行プロセッサーのバックグラウンド・プロセスを起動できません。**

**原因:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスが実行中ですが、db2ext コマンド行プロセッサーがそのプロセスと通信できません。

**処置:** バックグラウンド・プロセスの実行可能モジュール (db2extb または db2extb.exe) が存在し、そのディレクトリーが PATH 環境変数にあることをチェックしてください。

---

**DMB0355E　db2ext コマンド行プロセッサーのバックグラウンド・プロセスがタイムアウトになりました。**

**原因:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスが正常に起動しましたが、db2ext コマンド行プロセッサーが許された時間制限内にそのプロセスと通信できませんでした。

**処置:** その db2ext コマンドを別のウィンドウで試みてください。

**DMB0356E　db2ext コマンド行プロセッサーのバックグラウンド・プロセスと通信できません。**

**原因:** db2ext コマンド行プロセッサーがそのバックグラウンド・プロセスに要求を送信しましたが、要求が受信されませんでした。

**処置:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスがまだ実行中であることを確認してください。

**DMB0357E　db2ext コマンド行プロセッサーのバックグラウンド・プロセスが応答しません。**

**原因:** db2ext コマンド行プロセッサーがそのバックグラウンド・プロセスに要求を送信しましたが、バックグラウンド・プロセスが許された時間制限内に応答しませんでした。

**処置:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスがまだ実行中であることを確認してください。

**DMB0359E　db2ext コマンド行プロセッサーのバックエンド・プロセスの要求待ち行列または入力待ち行列が、タイムアウト時間内に作成されませんでした。**

**原因:** db2ext コマンド行プロセッサーのバックグラウンド・プロセスが、許された時間制限内にメッセージ待ち行列を作成できませんでした。(このメッセージは UNIX 環境にのみ適用されます。)

**処置:** DB2 インスタンスのホーム・ディレクトリーが常駐するディスクがいっぱいでないことを確認してください (バックグラウンド・プロセスがメッセージ待ち行列用のファイルを作成するには、このホーム・ディレクトリーが必要です)。このディスクがいっぱいでない場合は、起動した db2extb プロセスが多すぎないかどうかをチェックしてください。これは、多くのウィンドウで db2ext コマンド行プロセッサーを実行中である場合に、発生する可能性があります。バックグラウンド・プロセスは、db2ext コマンド行プロセッサー要求をコマンド・モードで最初に出したときにウィンドウで起動されます。 db2ext コマンド行プロセッサーが不要になった場合には、それを終了するためのコマンド `db2ext terminate` を必ず出してください。 `terminate` コマンドを出した場合のみ、バックエンド・プロセスに関するメッセージ待ち行列が削除されます。

**DMB0361E　列または表が使用可能になっていません。**

**原因:** インポート UDF が指定されましたが、指定された表列がエクステンダーについて使用可能になっていません。

**処置:** 表列を使用可能にしてから、再試行してください。

**DMB0363E　表名と列名が欠落しています。**

**原因:** 更新 UDF が呼び出されましたが、表が指定されていません。

**処置:** 表を指定してから、再試行してください。

**DMB0364E　"<エクステンダー名>" エクステンダーは、"<表スペース名>" 表スペースに事前定義されています。**

**原因:** 指定されたデータベース、表、または列が、指定された表スペースと異なる表スペースを使用して、そのエクステンダーについてすでに使用可能になっています。

**処置:** 表スペースの指定が正しいことをチェックしてください。

**DMB0365E　"<スキーマ名>"."<表名>" のメタデータ表である "<メタデータ表名>" と "<メタデータ表名>" に対する CONTROL 特権がありません。**

**原因:** 指定されたユーザー表のメタデータ表に対する CONTROL 特権がないので、要求が拒否されました。

**処置:** メタデータ表に対する CONTROL 特権を授与するように、データベース管理者に依頼してください。

**DMB0366E　フィーチャー名が欠落しています。**

**原因:** 照会ストリングにフィーチャー名が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0367E　カラー|カラー・ヒストグラム|ファイルが欠落しています。**

**原因:** 照会ストリングに『カラー』、『ヒストグラム』、または『ファイル』のいずれかが必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0368E　',' が欠落しています。**

**原因:** 照会ストリングに ',' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0369E　ファイルが無効です。**

**原因:** 照会ストリングに指定されたファイルが無効です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0370E　ファイル名が欠落しています。**

**原因:** 照会ストリングにファイル名が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0371E　サーバー|クライアントが欠落しています。**

**原因:** 照会ストリングに 『サーバー』 または 『クライアント』 が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0372E　'(' が欠落しています。**

**原因:** 照会ストリングに '(' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0373E　')' が欠落しています。**

**原因:** 照会ストリングに ')' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0374E　パーセンテージが欠落しています。**

**原因:** 照会ストリングにパーセント値が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0375E　カラーが欠落しています。**

**原因:** 照会ストリングに赤、緑、青の値が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0376E '=' が欠落しています。**

**原因:** 照会ストリングに '=' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0377E '<' が欠落しています。**

**原因:** 照会ストリングに '<' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0378E '>' が欠落しています。**

**原因:** 照会ストリングに '>' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0379E ' と ' が欠落しています。**

**原因:** 照会ストリングに ' と ' が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0380E 「重み」が欠落しています。**

**原因:** 照会ストリングに「重み」が必要です。

**処置:** 照会ストリングを訂正して、再試行してください。

**DMB0381E フィーチャーが設定されていません。**

**原因:** フィーチャーが QBIC カタログに追加されていません。

**処置:** フィーチャーを QBIC カタログに追加し、画像を再カタログしてください。

**DMB0382E 照会を作成できませんでした。**

**原因:** 現行データベースのエクステンダー・サーバーが停止した可能性があります。

**処置:** サーバーの状況をチェックしてください。サーバーが実行中でない場合は、db2ext コマンド行で START SERVER コマンドを使用して起動してください。

**DMB0383E 照会を実行できませんでした。**

**原因:** 現行データベースのエクステンダー・サーバーが停止した可能性があります。

**処置:** サーバーの状況をチェックしてください。サーバーが実行中でない場合は、db2ext コマンド行で START SERVER コマンドを使用して起動してください。

**DMB0384E 次の項目を入手できませんでした。**

**原因:** リストの最後に到達しています。

**処置:** アプリケーションがリストの最後を超えて項目の取り出しを試みていないかどうかをチェックしてください。

**DMB0386E ユーザーのデータを連結できません。**

**原因:** SQL API sqluihsh() が非ゼロの戻りコードを戻しました。

**処置:** 再試行してください。問題が解決しない場合は、IBM サポートに連絡してください。

**DMB0387E 指定されたテーブル・スペースに対するノードグループが、ユーザー・テーブルのものと異なります。**

**原因:** 表を使用可能にするための入力データとして渡された、1 つまたは複数の表スペース (メタデータ表、索引、または BLOB 用) が、ユーザー表が定義されたものとは異なるノード・グループに対して定義されています。

**処置:** ユーザー表が使用可能にされているのと同じノード・グループで定義された表スペースを使用してください。

**DMB0388E 正規、ロング、または索引のテーブル・スペースが、同一のノードグループで定義されていません。**

**原因:** データベースを使用可能化するための入力データとして渡された、1 つまたは複数の表スペース (メタデータ表、索引、または BLOB 用) が、他の表スペースと同じノード・グループで定義されていません。

**処置:** 同じノード・グループで定義された表スペースを使用してください。

**DMB0389W 指定されたテーブル・スペースのノードグループは、すべての区画サーバーをカバーしているわけではありません。**

**原因:** 入力データとして渡された表スペースが、一部の区画サーバーを含んでいないノード・グループで定義されています。

**処置:** 処置は必要ありません。ただし、すべての区画サーバーを包含するノード・グループで表スペースが定義されていれば、インポート UDF や更新 UDF がより効率的に実行されます。これは特に、エクステンダー・アプリケーションがメディア内容を BLOB 形式で保管する場合に当てはまります。

**DMB0391I このコマンドは、DB2 UDB サーバーにアクセスしている DB2 UDB クライアントに対してのみ適用できます。**

**原因:** db2ext コマンド行プロセッサーが DB2 UDB サーバーに接続されていないか、または db2ext コマンド行プロセッサーが DB2 UDB クライアントによって開始されていないかのどちらかです。たとえば、コマンド START SERVER が有効なのは、db2ext コマンド行プロセッサーが、DB2 エンタープライズ拡張エディション以外のサーバーに接続されている場合だけです。

**処置:** このコマンドは、現行のクライアント / サーバー構成では発行しないでください。

**DMB0392I このコマンドは、DB2 UDB エンタープライズ拡張エディション・サーバーにアクセスしている DB2 UDB クライアントに対してのみ適用できます。たとえば、DISCONNECT SERVER コマンドが有効なのは、db2ext コマンド行プロセッサーが DB2 エンタープライズ拡張エディション・サーバーに接続されている時だけです。**

**原因:** db2ext コマンド行プロセッサーが DB2 UDB エンタープライズ拡張エディションのサーバーに接続されていないか、または db2ext コマンド行プロセッサーが DB2 UDB クライアントから開始されていないかのどちらかです。

**処置:** このコマンドは、現行のクライアント / サーバー構成では発行しないでください。

**DMB0402E コマンド ″<コマンド名>″ のオプション ″<オプション名>″ は、アプリケーションが DB2 ″<サーバー・タイプ>″ サーバーに接続されている場合のみ有効です。**

**原因:** db2ext コマンド行プロセッサーがそのオプションをサポートするタイプのサーバーに接続されていないため、指定されたパラメーターが無効です。たとえば、コマンド GET SERVER STATUS をパラメーター NODENUM <nodenum> とともに指定できるのは、db2ext コマンド行プロセッサーが、DB2 エンタープライズ拡張エディションのサーバーに接続されている場合だけです。

**処置:** このコマンドとパラメーターの組み合わせは、現行のクライアント / サーバー構成では発行しないでください。

**DMB0411E 無効な基本ポート**

**原因:** インスタンス作成中に、無効な TCP/IP ポート番号が基本ポートとして入力されました。

**処置:** 正しい構文は次のとおりです。
dmbicrt-r:base_port,end_port -t:base_port,end_port
パラメーターを訂正して、コマンドを再試行してください。

---

**DMB0412E 無効な終了ポート**

**原因:** インスタンス作成中に、誤った TCP/IP ポート番号が終了ポートとして入力されました。

**処置:** 正しい構文は次のとおりです。
dmbicrt-r:base_port,end_port -t:base_port,end_port
パラメーターを訂正して、コマンドを再試行してください。

---

**DMB0413E DB2 エクステンダーのインストール・パスを分析できません。**

**原因:** インスタンス作成プログラムが環境変数 ″DMBPATH″ の値を検出できませんでした。

**処置:** 変数 ″DMBPATH″ を設定して、アプリケーションを再試行してください。

---

**DMB0414E コンピューター・ホスト名を分析できません。**

**原因:** コンピューター名を解決しようとしていたときに、内部エラーが検出されました。

**処置:** IBM サポートに連絡してください。

---

**DMB0415E このマシンのノード数が分析できません。**

**原因:** インスタンス作成を実行しているマシンが、ファイル ″db2nodes.cfg″ にリストされていません。

**処置:** そのマシンを ″db2nodes.cfg″ に追加して、アプリケーションを再試行してください。

---

**DMB0416E このプログラムは root で開始されなければなりません。続行できません。**

**原因:** プログラムが実行されているユーザー ID が、root 権限を持っていません。

**処置:** root としてログオンして、アプリケーションを再試行してください。

---

**DMB0417E このプログラムは管理者権限のあるユーザーによって実行されなければなりません。続行できません。**

**原因:** プログラムを実行しているユーザー ID が、管理権限を持っていません。

**処置:** 管理権限を持っているユーザー ID でログオンして、アプリケーションを再試行してください。

---

**DMB0418E ユーザーに関する情報を入手できません: ″<ユーザー ID>″。**

**原因:** 作成中のインスタンスに関連したユーザー情報を取得しようとしているときに、内部エラーが発生しました。

**処置:** 作成中のインスタンスと同じ名前を持つ有効なユーザー ID があることを確認して、アプリケーションを再試行してください。

---

**DMB0419E AIV エクステンダー・ディレクトリー ″<ディレクトリー名>″ を作成できません。 リターン・コード = <コード>。**

**原因:** 指定されたディレクトリーを作成しようとしているときに、エラーが発生しました。戻りコードは、オペレーティング・システムから戻されたエラーを表しています。

**処置:** ディレクトリー名で指定されたファイル・システム / ドライブが存在し、ディレクトリーを作成する許可が与えられていることを確認してください。

**DMB0420E AIV エクステンダー・ディレクトリー ″<ディレクトリー名>″ へのリンクを作成できません。リターン・コード = <コード>。**

**原因:** 指定された記号リンクを作成しようとしているときに、エラーが発生しました。戻りコードは、オペレーティング・システムから戻されたエラーを表しています。

**処置:** ディレクトリー名で指定されたファイル・システム / ドライブが存在し、リンクを作成する許可が与えられていることを確認してください。

**DMB0421E ファイル ″<ファイル名>″ をオープンできません。リターン・コード = <コード>。**

**原因:** 指定されたファイルを開こうとしているときに、エラーが発生しました。戻りコードは、オペレーティング・システムから戻されたエラーを表しています。

**処置:** ファイルが存在し、そのファイルを開く許可を与えられていることを確認してください。

**DMB0422E ファイル ″<ファイル名>″ へ書き込めません。リターン・コード = <コード>。**

**原因:** 指定されたファイルに書き込もうとしているときに、エラーが発生しました。戻りコードは、オペレーティング・システムから戻されたエラーを表しています。

**処置:** ファイルが存在し、そのファイルに書き込む許可を与えられていることを確認してください。

**DMB0424E db2nodes.cfg を検出できません。**

**原因:** DB2 ファイル ″db2nodes.cfg″ が見つかりませんでした。

**処置:** 正しいバージョンの DB2 UDB エンタープライズ拡張エディションがインストールされていることを確かめて、アプリケーションを再試行してください。

**DMB0426E エラー: ″<エラー・コード>″ がキー ″<レジストリー・キー>″ をオープンしています。**

**原因:** 指定されたレジストリー・キーを開こうとしているときに、エラーが発生しました。

**処置:** 戻りコードを記録して、IBM サポートに連絡してください。

**DMB0427E 変数 ″<変数>″ は、プロファイル・レジストリーに設定されていません。**

**原因:** 指定された値が Windows NT レジストリーにありませんでした。

**処置:** 有効な DB2 エクステンダー変数名を指定しているかを確認してください。

**DMB0430E DB2 レジストリー値を見つけられません。**

**原因:** DB2 の使用する登録値が見つかりませんでした。

**処置:** 正しいバージョンの DB2 UDB エンタープライズ拡張エディションがインストールされていることを確かめて、アプリケーションを再試行してください。

**DMB0431E エクステンダー・レジストリー・キー ″<レジストリー・キー>″ を作成できません。**

**原因:** エクステンダー・レジストリー・キーを作成しようとしているときに、内部エラーが発生しました。

**処置:** IBM サポートに連絡してください。

**DMB0432E　エクステンダー・レジストリー・キー ″<レジストリー・キー>″ に対する値を設定できません。**

**原因:** エクステンダー・レジストリー・キー値を設定しようとしているときに、内部エラーが発生しました。

**処置:** IBM サポートに連絡してください。

**DMB0435E　制御ファイル ″<制御ファイル>″ にアクセス不可能です。**

**原因:** 指定された制御ファイルは見つかりませんでした。

**処置:** IBM サポートに連絡してください。

**DMB0443E　ディレクトリー ″<ディレクトリー名>″ をオープンできません。リターン・コード = <コード>。**

**原因:** 指定されたディレクトリーを開こうとしているときに、エラーが発生しました。戻りコードは、オペレーティング・システムから戻されたエラーを表しています。

**処置:** ディレクトリー名で指定されたファイル・システム / ドライブが存在し、ディレクトリーを開く許可が与えられていることを確認してください。

**DMB0449W　-q:datapath は、AIV エクステンダー・インスタンス作成に必要です。**

**原因:** DB2 エクステンダー・インスタンスを作成しているとき、-q パラメーターが指定されませんでした。

**処置:** '-q' パラメーターを指定して、アプリケーションを再試行してください。

**DMB0450W　1 つまたはそれ以上の指定された ″<ポート>″ ポートは、すでに使用中です。**

**原因:** サービス・ファイル内にすでに使用中とリストされているポートが、DB2 エクステンダーで使用するために指定されました。

**処置:** 使用中でないポートを指定して、アプリケーションを再試行してください。

**DMB0452E　ノード数 ″<ノード数>″ は、″db2nodes.cfg″ で検出できません。**

**原因:** このマシンのノード番号が db2nodes.cfg ファイルにありませんでした。

**処置:** そのノード番号を db2nodes.cfg ファイルに追加して、アプリケーションを再試行してください。

**DMB0460W　TCP/IP ポートが使用可能かどうか判別できません。**

**原因:** 指定された TCP/IP ポートがすでに使用中かどうかを確認しようとしているときに、エラーが発生しました。

**処置:** 指定されたポートが、別のアプリケーションによって使用中であるとサービス・ファイルにリストされていないかを確認してください。

**DMB0462E　このノードを初期化できません。リターン・コード = <コード>。**

**原因:** 現行ノードを初期化しようとしているときに、エクステンダー始動プログラムがエラーを検出しました。

**処置:** IBM サポートに連絡してください。

**DMB0495E このバージョンの AIV エクステンダーは、ロング・ネームをサポートしません。**

**原因:** エクステンダー管理 API の呼び出し中、または db2ext コマンド行コマンドの発行時に、長い識別子が指定されました。このバージョンの AIV エクステンダーでサポートされている識別子の最大長は、次のとおりです。

- ローカル許可 ID (AUTHID) − 8 文字
- 表スキーマ (TABSCHEMA) − 8 文字
- 表名 (TABNAME) − 18 文字
- 列名 − 18 文字

API 呼び出しまたはコマンドを調べて、短い識別子を使用してください。

**DMB0496E 無効な表名または列名が指定されました。**

**原因:** エクステンダーの管理 API の呼び出し中、または db2ext コマンド行コマンドの発行時に、無効な識別子が指定されました。その原因として、識別子が長過ぎた可能性が考えられます。UDB Db2 で有効な名前の長さについて、詳しくは*概説およびインストール* を参照してください。

API 呼び出しまたはコマンドを調べて、短い識別子を使用してください。

**DMB497E DB2MMDATAPATH でアクセスが拒否されました。**

**原因: (EEE のみ)** 一部のノードからアクセスできないディレクトリー名または共用名が指定されました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB498E DB2MMDATAPATH パスのうち、少なくとも 1 部分がディレクトリーではありません。**

**原因: (EEE のみ)** ノード上のディレクトリーではないディレクトリー名または共用名が指定されました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB499E DB2MMDATAPATH パス・ストリングが長過ぎます。**

**原因: (EEE のみ)** 指定したディレクトリー名または共用名では、変数 DB2MMDATAPATH が長くなってしまいます。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名が正しいかどうか、およびそれがすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB500E DB2MMDATAPATH ディレクトリーが存在しません。**

**原因: (EEE のみ)** ノード上に存在しないディレクトリー名または共用名が指定されました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB501E DB2MMDATAPATH で不明な stat() エラーです。**

**原因: (EEE のみ)** この環境変数の中のディレクトリー名または共用名にアクセスしようとしたとき、問題が発生しました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB502E DB2MMDATAPATH は存在しますが、ディレクトリーではありません。**

**原因: (EEE のみ)** 指定されたディレクトリー名または共用名は、すべてのノード上のディレクトリー名または共用名というわけではありません。DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB503E DB2MMDATAPATH は存在しますが、読み取れません。**

**原因: (EEE のみ)** 一部のノードから読み取れないディレクトリー名または共用名が指定されました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能なものでなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB504E DB2MMDATAPATH 存在しますが、書き込めません。**

**原因: (EEE のみ)** 一部のノードに書き込めないディレクトリー名または共用名が指定されました。 DB2 エクステンダーのインスタンスの作成時に指定されるディレクトリー名または共用名はすべてのノード上に存在し、アクセス可能かつ読み取り / 書き込み可能でなければなりません。インスタンスの作成時に指定したディレクトリー名または共用名がすべてのノード上に存在し、アクセス可能であることを確認してください。

**DMB504E DB2MMDATAPATH 存在しますが、書き込めません。**

**原因: (EEE のみ)** DB2 エクステンダー・インスタンスの作成時に、環境変数 DB2MMDATAPATH が設定されませんでした。これが新規の DB2 エクステンダー・インスタンスであれば、DMBIDROP を使ってインスタンスを除去し、その後 `-q` オプションを正しく指定して再作成してください。

これが新規の DB2 エクステンダー・インスタンスでない場合は、次のようにします。

- UNIX 環境では:
  1. ディレクトリー名が正しく、すべてのノードに存在し、アクセス可能であることを確認します。
  2. *$INSTHOME*/dmb/dmbprofile を変更して、DB2MMDATAPATH をディレクトリーとしてエクスポートします。
- Windows 環境では:
  1. そのディレクトリーの共用名が正しいかどうか、およびディレクトリーがすべてのノード上に存在し、アクセス可能であることを確認します。
  2. IAV エクステンダー・インスタンスのレジストリーに、その共用名を持つレジストリー項目 DB2MMDATAPATH を値として追加します。キーは

¥¥HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥DB2 Extenders¥PROFILE¥*instance_name* ¥DB2MMDATAPATH です。

**DMB506E　インスタンス名がセットされていません。**

**原因:** DMBSTART の実行時に、DB2INSTANCE 環境変数が設定されていませんでした。
DB2START を使って DB2 エクステンダー・サービスを開始する前に、DMBSTART が正しく機能することを確認してください。

**DMB507E　dmbssd *name* ノード *arguments***

**原因:** 内部エラー。 IBM 担当員に連絡してください。

**DMB508E　ノード番号は、0 以上でなければなりません。**

**原因:** 内部エラー。 IBM 担当員に連絡してください。

**DMB509E　このプログラムは、手動で開始してはいけません。**

**原因:** 内部エラー。 IBM 担当員に連絡してください。

**DMB512E　使用法： *arguments* dmbInstName。**

**原因:** 内部エラー。 IBM 担当員に連絡してください。

**DMB513E　*Name* は有効なインスタンスではありません。**

**原因:** DB2 エクステンダー・インスタンスを除去しようとした時に指定した名前は、インスタンスの名前として認識されませんでした。指定したインスタンス名が正しいかどうか、およびその名前を含むディレクトリー *$INSTHOME*/dmb が存在することを確かめてください。

**DMB514I　このインスタンスには、サーバーもクライアントもインストールされていません。**

**原因:** DB2 エクステンダー・インスタンスを除去しようとしましたが、エクステンダーはインストールされていませんでした。正しくインストールされているかどうか、およびインストール先ディレクトリーの名前が変更されていないかどうかを確認してください。

**DMB515I　このインスタンスには、サーバーもクライアントもインストールされていません。**

**原因:** DB2 エクステンダー・インスタンスを除去する時に、関連する DB2 のインスタンスが除去されません。 DB2 インスタンスを除去するには、DB2IDROP を使用してください。

**DMB518E　予期しないエラーです。関数 = *function name*、リターン・コード = *return_code*。**

**原因:** DB2 エクステンダー・インスタンスを作成または除去しようとした時に、予期しないエラーが発生しました。インストールや設定が正しいことを確認してください。

**DMB520E　このプログラムは root では実行できません。**

**原因:** この処置を実行するための正しい権限を持っていることを確認してください。

**DMB521E　*name* の許可の変更に失敗しました。**

**原因:** 許可を変更するための正しい権限を持っていることを確認してください。

**DMB522E** *name* **の所有権の変更に失敗しました。**

**原因:** 所有権を変更するための正しい権限を持っていることを確認してください。

**DMB523E** *name* **のグループ所有権の変更に失敗しました。**

**原因:** グループ所有権を変更するための正しい権限を持っていることを確認してください。

**DMB524E** **ファイルまたはディレクトリー** *name* **は既に存在します。**

**原因:** 指定した名前のファイルまたはディレクトリーがすでに存在します。別の名前を選択して、コマンドを再実行してください。

**DMB525E** *name* **の作成に失敗しました。**

**原因:** この処置を実行するための正しい権限を持っていることを確認してください。

**DMB526E** **ファイルまたはディレクトリー** *name* **がありません。**

**原因:** 指示されたファイルまたはディレクトリーがありません。ファイルまたはディレクトリーの名前を正しく指定したかどうか確認してください。

**DMB527E** **ファイルまたはディレクトリー** *name* **の** *name* **へのコピーに失敗しました。**

**原因:** ファイルまたはディレクトリーの複写に必要な権限を持っていることを確認してください。また、複写のための十分なスペースがあることを確認してください。

**DMB528E** **ユーザー ID** *user_ID* **は無効です。**

**原因:** 無効なユーザー ID が指定されました。ユーザー ID を確認して、コマンドを再実行してください。

**DMB529E** **ユーザー ID** *user_ID* **の 1 次グループ** *group* **は無効です。**

**原因:** ユーザー ID の 1 次グループの指定が無効です。正しい 1 次グループを確認して、コマンドを再実行してください。

**DMB530E** **インスタンス名** *name* **は無効です。**

**原因:** インスタンスを作成または使用しようとした時に、無効な名前を指定しました。有効なインスタンス名を確認して、コマンドを再実行してください。

**DMB531E** **オペレーティング・システム** *name*、**バージョン** *version_number* **はサポートされていません。**

**原因:** サポートされていないバージョンのオペレーティング・システム上でこのコマンドを実行しようとしました。この操作の実行に必要な要件を確認してください。

**DMB535E** **指定したファイルにアクセスできません。**

**原因:** このコマンドを実行する前に、ファイルへのアクセスが可能かどうか確かめてください。

**DMB0533E** **API <***API 名***> 区分データベース・サーバー環境ではサポートされていません。**

**原因:** 指定された API は、区分データベース環境では使用できません。

**処置:** ご使用のアプリケーションでこの機能を扱

う方法について、この API に関するセクションを参照してください。

**DMB0534E UDF はサポートされていません。**

**原因:** 指定されたユーザー定義関数は、区分データベース環境では使用できません。

**処置:** メッセージ SQL0443N を調べて、どの UDF が問題を起こしたかを判別してください。ご使用のアプリケーションでこの機能を扱う方法について、その UDF に関するセクションを参照してください。

## 診断トレース

DB2 エクステンダーには、エクステンダー・サーバー活動を記録するトレース機能が付属しています。トレース機能を使用するのは、IBM サービス技術員の指示があった場合だけにしてください。

トレース機能は、DB2 エクステンダー構成要素への入り口または出口、DB2 エクステンダー構成要素が出すエラー・コードの戻りなど、さまざまな事象に関する情報をサーバー・ファイルに記録します。トレース機能は、多くの事象に関する情報を記録するので、エラー条件を調査しているときなど、必要な場合だけ使用してください。さらに、トレース機能の使用中は活動アプリケーションの数も限定する必要があります。活動アプリケーションの数を限定すると、問題の原因を突きとめるのが容易になります。

トレースを制御するには、DMBTRC コマンドを使用します。このコマンドは、OS/2 サーバー、AIX サーバー、または Windows NT 以降のサーバーのコマンド行から発行できます。このコマンドを出すには、SYSADM、SYSCTRL、または SYSMINT 権限を持っている必要があります。

DMBTRC コマンドを使用して、次のことを実行します。

- トレース機能の開始
- トレース機能の停止
- トレース情報を再フォーマットして、もっと読みやすくする
- トレース状況の表示

### トレース機能の開始

次のコマンドを入力すれば、トレース機能を開始できます。

dmbtrc on *path*

*path* はトレース情報を保管するサーバー・ファイルのパスです。

たとえば、次のコマンドでトレース機能が開始します。

```
dmbtrc on /tmp/trace.txt
```

## トレース機能の停止

次のコマンドを入力すれば、トレース機能を停止できます。

```
dmbtrc off
```

## トレース情報の再フォーマット

トレース情報は 2 進形式で記録されます。次のコマンドを入力してこの情報を再フォーマットすれば、情報がもっと読みやすくなります。

```
dmbtrc format input_file output_file
```

*input_file* は 2 進形式でトレース情報を保管するファイル、*output_file* は再フォーマットした情報を保管するファイルです。 *output_file* パラメーターは任意指定です。このパラメーターを指定しない場合、再フォーマット後の情報が画面に表示されます。

たとえば、次のコマンドで、トレース情報が再フォーマットされます。

```
dmbtrc format /tmp/trace.txt /tmp/fmttrace.txt
```

## トレース状況の表示

次のコマンドを使用します。

```
dmbtrc info
```

このコマンドは、次のトレース状況情報を表示する場合に使用します。

- トレース機能のオン / オフ
- トレース情報が入っているファイルのパス

# 第5部 付録および後付け

# 付録A. DB2 エクステンダー用の環境変数の設定

DB2 エクステンダーを使用すると、画像、音声、ビデオのオブジェクトを保管、検索、または更新する際に、柔軟にファイル名を指定することができます。また、データベース表から検索された画像、音声、またはビデオのオブジェクトを表示 / 再生する際にプログラムを指定する方法も柔軟になります。

## 環境変数の使用によるファイル名の解決方法

保管、取り出し、および更新操作に対して、完全に修飾されたファイル名 (つまり、ファイル名の前に完全なパス) を指定できますが、相対ファイル名を指定した方がよいでしょう。 AIX、HP-UX、または Solaris では、相対ファイル名は、スラッシュが先頭に付かない任意のファイル名です。 OS/2 および Windows では、相対ファイル名は、ドライブ文字の後にコロンと ¥ を続けた形で始まらない任意のファイル名です。

相対ファイル名を指定すると、エクステンダーはさまざまなクライアントおよびサーバーの環境変数のディレクトリー指定を使用してファイル名を解決します。こうすることによって、ファイル名を変更せずにファイルをクライアント / サーバー環境で移動できます。完全に修飾されたファイル名は、ファイルを移動するたびに変更する必要があります。

表17 に、イメージ、オーディオ、およびビデオ・エクステンダーでファイル名の解決に使用する環境変数のリストと説明を記載します。

*表 17. DB2 エクステンダーの環境変数*

| イメージ・エクステンダー | オーディオ・エクステンダー | ビデオ・エクステンダー | 説明 |
|---|---|---|---|
| | | **サーバー環境変数** | |
| DB2IMAGEPATH | DB2AUDIOPATH | DB2VIDEOPATH | サーバー・ファイルからの保管、取り出し、および更新操作に使うソース・ファイル名の解決に使用します。 |
| DB2IMAGESTORE | DB2AUDIOSTORE | DB2VIDEOSTORE | サーバー・ファイルへの保管と更新の操作に使うターゲット・ファイル名の解決に使用します。 |

*表 17. DB2 エクステンダーの環境変数 (続き)*

| イメージ・エクステンダー | オーディオ・エクステンダー | ビデオ・エクステンダー | 説明 |
|---|---|---|---|
| DB2IMAGEEXPORT | DB2AUDIOEXPORT | DB2VIDEOEXPORT | サーバー・ファイルへの取り出し操作に使うターゲット・ファイル名の解決に使用します。 |
| DB2IMAGETEMP | | | 一時サーバー・ファイルを作成する操作に使うターゲット・ファイル名の解決に使用します。ただし、TMP 環境変数が指定されていれば、ディレクトリー TMP がファイル名の解決に使用されます。 |
| | | **クライアント環境変数** | |
| DB2IMAGEPATH | DB2AUDIOPATH | DB2VIDEOPATH | クライアント・ファイルでの表示 / 再生操作に使うソース・ファイル名の解決に使用します。 |
| DB2IMAGETEMP | DB2AUDIOTEMP | DB2VIDEOTEMP | 一時クライアント・ファイルを作成する操作の場合にターゲット・ファイル名の解決に使用します。ただし、TMP 環境変数が指定されていれば、ディレクトリー TMP がファイル名の解決に使用されます。 |

特定のエクステンダーの適切な環境変数を設定していなければ、エクステンダーは次の環境変数を使用してファイル名を解決します。

| 環境変数 | 説明 |
|---|---|
| **DB2MMPATH** | 保管、取り出し、および更新操作の場合にソース・ファイル名の解決に使用します。 |
| **DB2MMSTORE** | 保管と更新の操作の場合に使うターゲット・ファイル名の解決に使用します。 |
| **DB2MMEXPORT** | 取り出し操作の場合に使うターゲット・ファイル名の解決に使用します。 |
| **DB2MMTEMP** | 一時ファイルを作成する操作の場合に使うファイル名の解決に使用します。 |

## 環境変数を使って表示 / 再生プログラムを識別する方法

環境変数は、ファイル名の解決だけでなく、イメージ・エクステンダーが検索する画像オブジェクトを表示し、オーディオ・エクステンダーやビデオ・エクステンダーが取り出す音声またはビデオのオブジェクトを再生するプログラムを識別するためにも使用されます。これらのオブジェクトを表示 / 再生するには、DBiBrowse、DBaPlay、および DBvPlay API を使用します。それぞれの API を使用するときに、特定の表示 / 再生プログラムを指定するか、または省略時プログラムを使ってオブジェクトを表示 / 再生するよう指示できます。

DB2 エクステンダーは、次のクライアント環境変数を使用して、省略時の表示 / 再生プログラムを識別します。

| 環境変数 | 説明 |
|---|---|
| **DB2IMAGEBROWSER** | 省略時の画像表示プログラムの識別に使用します。 |
| **DB2AUDIOPLAYER** | 省略時のオーディオ・プレーヤー・プログラムの識別に使用します。 |
| **DB2VIDEOPLAYER** | 省略時のビデオ・プレーヤー・プログラムの識別に使用します。 |

## DB2MMDATAPATH 環境変数の使用方法 (EEE のみ)

DB2 エクステンダーは DB2MMDATAPATH 環境変数を使用して、区分データベース環境における各種操作のためのロケーションを解決します。たとえば、DB2 イメージ・エクステンダーは DB2MMDATAPATH の値を使用して、区分データベース環境で QBIC データを保管します。

DB2MMDATAPATH を設定するのは、DB2 エクステンダー・インスタンスの作成時です。これについて、 554ページの『DMBICRT』および次のインストール 『readme』 ファイルに説明があります。

- aixeee ディレクトリーの install.txt (DB2 エンタープライズ拡張エディション (AIX 版) で使用するために DB2 エクステンダーをインストールする場合)
- soleee ディレクトリーの install.txt (DB2 エンタープライズ拡張エディション (Solaris 実行環境版) で使用するために DB2 エクステンダーをインストールする場合)

DB2MMDATAPATH を使用する例として、QBIC フィーチャーおよび索引データを保管する場合があります。 UNIX では、DB2 イメージ・エクステンダーはこの QBIC データを次のディレクトリーに保管します。

*db2mmdatapath* /NODE*node_num*/QBIC/*database_name*

*db2mmdatapath* は DB2MMDATAPATH 環境変数の値で、*node_num* はノード番号、*database_name* はデータベース名です。

次の AIX の例を考慮してください。 DB2MMDATAPATH が /localfs/dmbdata に設定されているとします。また、sample というデータベースが、ノード 0、2、および 5 に区画化されているとします。 QBIC データは、次のディレクトリーにあるサンプル (sample) データベース用に保管されます。

ノード 0: /localfs/dmbdata/NODE0000/QBIC/sample

ノード 2: /localfs/dmbdata/NODE0002/QBIC/sample

ノード 5: /localfs/dmbdata/NODE0005/QBIC/sample

## 環境変数の設定

環境変数は、AIX、HP-UX、Solaris、OS/2、および Windows で設定できます。

### AIX、HP-UX、Solaris サーバーおよびクライアントでの環境変数の設定

AIX、HP-UX、および Solaris の場合、環境変数は C シェル、Korn シェル、および Bourne シェル・スクリプトで指定します。サーバー用の環境変数は、DB2 エクステンダーのインストール時に次のように設定されます。

#### C シェル

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/tmp
setenv DB2MMTEMP /tmp
setenv DB2MMSTORE /tmp
setenv DB2MMEXPORT /tmp
```

#### Korn および Bourne シェル

```
DB2MMPATH=/usr/lpp/db2ext/samples:/tmp
export DB2MMPATH

DB2MMSTORE=/tmp
export DB2MMSTORE

DB2MMEXPORT=/tmp
export DB2MMEXPORT

DB2MMTEMP=/tmp
export DB2MMTEMP
```

サーバーの環境変数は、DB2 エクステンダー付属のサンプル・プログラムで使用するメディア・ファイルへのアクセスを可能にする値に初期設定されます。(サンプル・プログラムとメディア・ファイルの詳細については、623ページの『付録B. サンプル・プログラムとメディア・ファイル』を参照してください。)

クライアント環境変数は、AIX、HP-UX、または Solaris クライアントへのインストール時に、次のように設定されます。

### C シェル

```
setenv DB2MMPATH /tmp
setenv DB2MMTEMP /tmp
```

### Korn および Bourne シェル

```
DB2MMPATH=/tmp
export DB2MMPATH

DB2MMTEMP=/tmp
export DB2MMTEMP
```

ファイル名の解決に使用されるサーバーおよびクライアント環境変数を設定します。ご使用の環境に合った値を指定してください。 PATH で終わる環境変数には、複数のディレクトリーを区切り文字で区切って指定することができます。 STORE、EXPORT、および TEMP で終わる環境変数に設定できるディレクトリーは、1 つだけです。

適切な画像表示、音声再生、およびビデオ再生プログラムの名前をそれぞれ DB2IMAGEBROWSER、DB2AUDIOPLAYER、および DB2VIDEOPLAYER に指定してください。

環境変数の初期設定値は、次のようにして変更することができます。

### C シェル

SETENV コマンドを使用して、環境変数を設定します。

```
setenv env-var directory
```

次はその例です。

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/media
setenv DB2IMAGEPATH /employee/pictures:/images
setenv DB2AUDIOSTORE /employee/sounds
setenv DB2IMAGEBROWSER 'xv %s'
```

### Bourne シェル

EXPORT コマンドを使用して、環境変数を設定します。

```
env-var=directory
export env-var
```

次はその例です。

```
DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2MMPATH

DB2IMAGEPATH=/employee/pictures:/images
export DB2IMAGEPATH

DB2AUDIOSTORE=/employee/sounds
export DB2AUDIOSTORE
```

#### Korn シェル

EXPORT コマンドを使用して、環境変数を設定します。

```
export env-var=directory
```

次はその例です。

```
export DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2IMAGEPATH=/employee/pictures:/images
export DB2AUDIOSTORE=/employee/sounds
```

### OS/2 サーバーおよびクライアントでの環境変数の設定

OS/2 の場合、環境変数はインストール時に CONFIG.SYS ファイルに追加され、自動設定されます。

OS/2 サーバーに DB2 エクステンダーをインストールする場合、サーバー環境変数は次のように設定されます。

```
SET DB2MMPATH=install-dir¥SAMPLES;temp-file-dir
SET DB2MMSTORE=temp-file-dir
SET DB2MMEXPORT=temp-file-dir
SET DB2MMTEMP=temp-file-dir
```

*install-dir* はインストール・ディレクトリー、*temp-file-dir* は一時ファイル・ディレクトリーです。省略時のインストール・ディレクトリーは C:¥DMB、省略時の一時ファイル・ディレクトリーは C:¥DMB¥TMP です。どちらのディレクトリーもインストール時に位置を変更できます。一時ファイル・ディレクトリーの位置を正しく指定することが重要です。

サーバーの環境変数は、DB2 エクステンダー付属のサンプル・プログラムで使用するメディア・ファイルへのアクセスを可能にする値に初期設定されます。

(サンプル・プログラムとメディア・ファイルの詳細については、623ページの『付録B. サンプル・プログラムとメディア・ファイル』 を参照してください。)

OS/2 クライアントに DB2 エクステンダーをインストールする場合、クライアント環境変数は次のように設定されます。

```
SET DB2MMPATH=temp-file-dir
SET DB2MMTEMP=temp-file-dir
```

環境変数を再設定するには、SET コマンドを使用します。 PATH で終わる環境変数には、複数のディレクトリーを区切り文字で区切って指定することができます。 STORE、EXPORT、および TEMP で終わる環境変数に設定できるディレクトリーは、1 つだけです。

SET コマンドを使用して、適切な画像表示、オーディオ・プレーヤー、およびビデオ・プレーヤー・プログラムを、それぞれ DB2IMAGEBROWSER、DB2AUDIOPLAYER、および DB2VIDEOPLAYER に指定してください。

SET コマンドは次のように指定します。

```
SET env-var=directory
```

次はその例です。

```
SET DB2MMPATH=C:¥DMB¥SAMPLES;¥D:¥MEDIA
SET DB2IMAGEPATH=C:¥EMPLOYEE¥PICTURES;D:¥IMAGES
SET DB2AUDIOSTORE=C:¥EMPLOYEE¥SOUNDS
SET DB2IMAGEBROWSER=ib.exe %s
```

## Windows サーバーおよびクライアントでの環境変数の設定

Windows では、環境変数の設定方法は、DB2 エクステンダーを使用するのが区分データベース環境 (つまり、Windows 用の DB2 エンタープライズ拡張エディション) であるか、非区分データベース環境であるかによって異なります。

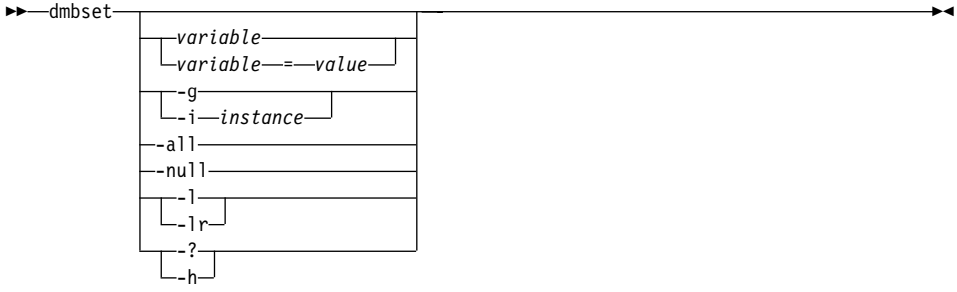
### Windows 非区分データベース環境での環境変数の設定 (EEE 以外のみ)

Windows では、環境変数がシステム・レジストリーに保管されます。変数を設定するには、Windows の「コントロール パネル」を開き、「システム」アイコンを選択します。「システムのプロパティ」ダイアログから、「環境」タブを選択します。環境変数とその値が入っている 2 つのウィンドウがあります。上のウィンドウは、すべてのユーザーに有効な変数を表示しています。下のウィンドウは、現行ユーザーだけに有効な変数を表示しています。

### Windows 区分データベース環境での環境変数の設定 (EEE のみ)

Windows 区分環境では、DB2 エクステンダーの使用するすべての変数が、システム・レジストリーの専用記憶域に保管されます。エクステンダー変数を検査して変更するために、DMBSET というプログラムが提供されています。

このプログラムの構文は次のとおりです。




変数の値を照会するには、dmbset *variable_name* と入力します。次はその例です。

```
dmbset DB2MMPATH
```

変数の値を設定するには、dmbset *variable_name=value* と入力します。次はその例です。

```
dmbset DB2MMPATH=C:¥DMB¥SAMPLES
```

定義されたインスタンスのすべての変数の値を表示するには、dmbset -i *instance_name* と入力します。次はその例です。

```
dmbset -i dmbinst1
```

値を空白値に設定するには、dmbset *variable_name -null* と入力します。次はその例です。

```
dmbset DB2MMPATH -null
```

すべてのインスタンスで使用される変数の値を表示するには、`dmbset -g` と入力します。

DB2 エクステンダーが使用するすべての変数名をリストするには、`dmbset -lr` と入力します。

レジストリーに定義されたすべてのインスタンス・プロファイルの名前をリストするには、`dmbset -l` と入力します。

区分データベース環境での DB2 エクステンダー用の環境変数の設定は、かなり柔軟に行えます。たとえば、DB2MMDATAPATH 以外の環境変数の値は、次のいずれの形式でも指定することができます。

- 汎用命名規則名: ¥¥*machine_name*¥*share_name*。次はその例です。

  ```
  ¥¥harmony¥JimsShr
  ```

- ドライブ:パス。次はその例です。

  ```
  f:¥media
  ```

- その他: *share_name*¥*directory_name*。次はその例です。

  ```
  JimsShr¥images
  ```

# 付録B. サンプル・プログラムとメディア・ファイル

DB2 エクステンダーには、さまざまなサンプル・プログラムが付属しています。これらのサンプル・プログラムは、エクステンダー付属のイメージ、オーディオ、およびビデオ・ファイルを使用します。ほとんどのサンプル・プログラムは、C で作成されています。すべての C のサンプル・プログラムは、コール・レベル・インターフェース (CLI) 形式です。また、いくつかの Java サンプル・プログラムと 1 つの Net.Data サンプル・マクロも提供されています。

これらのサンプル・プログラムは、DB2 エクステンダーのインストール時に、ターゲット・ディレクトリーの SAMPLES サブディレクトリーにインストールされます。イメージ、オーディオ、およびビデオ・ファイルも、DB2 エクステンダーのインストール時に、ターゲット・ディレクトリーの SAMPLES サブディレクトリーにインストールされます。エクステンダーの環境変数は、インストール時にターゲット・ディレクトリーの samples サブディレクトリーを指すように設定されます。

## サンプル・プログラム

DB2 エクステンダーのサンプル・プログラムは、多くのファイルによって構成されています。次のようなファイルです。

| ファイル | 説明 |
|---|---|
| **enable.c** | データベースをオーディオ、イメージ、およびビデオ・エクステンダー用に使用可能にして、表を作成し、表とその列を使用可能にします。 |
| **populate.c** | データを表にインポートします (プログラムは C 形式)。 |
| **Populate.java** | データを表にインポートします (プログラムは Java 形式)。 |
| **query.c** | 表のデータを照会します (プログラムは C 形式)。 |
| **Query.java** | 表のデータを照会します (プログラムは Java 形式)。 |
| **api.c** | エクステンダー API を使用してデータベースを照会します。 |
| **handle.c** | UDF でハンドルを使用する方法と、SELECT ステートメントで where 文節を比較する方法を例示します。 |

**qbcatdmo.c** QBIC カタログを作成して、画像の列をカタログに登録します。

**qbicdemo.c** QBIC カタログを照会します。

**color.c** qbicdemo.c のカラー・テーブルを宣言します。

**QbicQry.java** QBIC 照会の平均色セレクターおよびヒストグラム色セレクターを示します。

**makesf.c** makehtml.exe で使用するショット・カタログ・ファイルを作成します。

**makehtml.c** ショット・カタログにアクセスし、Web ブラウザーによる表示のための HTML ページを作成します。

**storybrd.java** ショットを表示するアプレット (makehtml.c によって生成される HTML ページで呼び出される)。

**utility.c** ユーティリティー・ルーチン。

**utility.h** ユーティリティー・ルーチンのヘッダー・ファイル。

**makefile.aix** AIX でプログラムを作成する Makefile。

**makefile.os2** OS/2 でプログラムを作成する Makefile。

**makefile.iva** IBM VisualAge C++ を使用して Windows NT (またはそれ以降) でプログラムを作成する Makefile。

**makefile.mvc** Microsoft Visual C++ を使用して Windows でプログラムを作成する Makefile。

**makefile.sun** Solaris でプログラムを作成する Makefile。

**makefile.hp** HP-UX でプログラムを作成する Makefile。

次のサンプル・プログラムには、実行可能ファイルが提供されています。これらのサンプル・プログラムは、次の順序で実行するよう意図されています。

1. Enable
2. Populate
3. Query
4. API
5. Handle
6. Qbcatdmo
7. Qbicdemo
8. QbicQry

9. Makesf
10. Makehtml

実行可能クラス・ファイル (Populate.class、Query.class、QbicQry.class、および storybrd.class) には、サンプルの Java プログラムが付属しています。

サンプル・プログラムを実行する前に、サーバー上にデータベースを作成する必要があります。エクステンダー・サービスをサーバー上で開始する必要もあります。サンプル・プログラムを実行するには、プログラム名を入力します (これによってプログラムの実行可能ファイルが開始されます)。データベース名、ユーザー ID、およびパスワードを入力するよう求めるプロンプトが表示されます。データベースを作成したユーザーのユーザー ID とパスワードを使用してください。

また、サンプル・プログラムの実行可能ファイルも独自に作成することができます。そうするには、次の操作が必要です。

1. サンプル・プログラム・ファイルを書き込み可能ディレクトリーに複写します。
2. makefile を編集して、DB2、エクステンダー、およびコンパイラーがインストールされているシステムの位置を指定します。
3. make または nmake を使用して、これらのファイルを実行可能プログラムにコンパイルします。

サンプル・プログラムのインストールおよび使用方法の詳細については、サンプル・プログラム・ディレクトリーにある README.CNT ファイルを参照してください。

## 画像、音声、およびビデオ・ファイルのサンプル

DB2 エクステンダーが提供するサンプルの画像ファイル、音声ファイル、およびビデオ・ファイルは、次のとおりです。

- 画像ファイル
  - lizzi.bmp
  - sws_stri.bmp
  - nitecry.bmp
  - ranger_r.bmp
  - fuzzblue.bmp
- 音声ファイル

  - lizzi.wav
  - sws_stri.wav
  - nitecry.wav
  - ranger_r.wav
  - fuzzblue.wav
- ビデオ・ファイル
  - nitecry.avi
  - sample.mpg

## サンプル Net.Data マクロ・ファイル

DB2 エクステンダーには、extender.d2w という Net.Data マクロ・ファイルが付属しています。 Web サーバー上でこのマクロ・ファイルを実行すると、DB2 エクステンダー UDF を呼び出す SQL ステートメントを実行します。このマクロ・ファイルの戻す結果は Web ブラウザーによって表示されます。また、627ページの図30 が示すように、結果を生成した SQL がそれぞれの結果ページに表示されます。 628ページの図31 では、サンプル Net.Data マクロ・ファイルの内容を示しています。

サンプル Net.Data マクロ・ファイルを実行するには、Web ブラウザーで以下の URL を入力してください。 http://*your server*/cgi-bin/db2www/extender.d2w/startHere

ここで *your server* は、ご使用の Web サーバーの名前です。

select cast(mmdbsys.thumbnail(covers) as blob(10000), cast(mmdbsys.thumbnail(video) blob(3000)), mmdbsys.comment(music), artist, title, price, stock_no from sobay_catalog

| Cover | Video | Audio | Artist | Title | Price |
|---|---|---|---|---|---|
|  |  | [Listen] | Lizzi | Decisions | 25.00 |
|  |  | [Listen] | SWS Strings | Vivaldi: Four Seasons | 25.50 |
|  |  | [Listen] | Nitecry | Run for Cover | 15.00 |
|  |  | [Listen] | Ranger Rick | Handy Sue | 12.25 |
|  |  | [Listen] | Fuzzy Blues | Aurora | 22.00 |

*図 30. サンプル Net.Data マクロ・ファイルを実行する Web アプリケーション.* 各結果ページは、結果を生成した SQL ステートメントを表示します。

```
%{ ------------------------------------------------------------------------ %}
%{ Copyright International Business Machines Corporation, 1998.            %}
%{ All rights reserved.                                                    %}
%{                                                                         %}
%{ Sample Net.Data macro which shows how to call image, audio, and video   %}
%{ extender UDFs.                                                          %}
%{                                                                         %}
%{ To run, put this macro in your MACRO_PATH root, make sure the tmplobs   %}
%{ directory exists under your web server's document root, and create      %}
%{ the database to be used when running the extender sample programs       %}
%{ 'enable' and 'populate'.  Run 'enable' and 'populate'.  If you name your %}
%{ database something other than 'testdb2', you'll need to change the     %}
%{ definition of DATABASE below.  The extender environment variable        %}
%{ DB2MMEXPORT needs to be set for the instance used by Net.Data to point  %}
%{ to the webserver's <document root>/tmplobs directory.  Then restart DB2 %}
%{ and the extenders to have the variable take effect.                     %}
%{ If you are not running Net.Data's Connection Manager, you'll need to    %}
%{ provide the LOGIN and PASSWORD to the database.  If these instructions  %}
%{ seem unfamiliar to you, you should read the Net.Data documentation at   %}
%{ http://www.software.ibm.com/data/netdata/docs (or the extender documen- %}
%{ tation on the extender sample programs).                                %}
%{                                                                         %}
%{ To disable the showing of SQL statements, change the value of SHOWSQL   %}
%{ below to "no".                                                          %}
%{ ------------------------------------------------------------------------ %}
%{ ------------------------------------------------------------------------ %}
%{ Definitions section                                                     %}
%{ ------------------------------------------------------------------------ %}
%define{
  DATABASE="testdb2"
  SHOWSQL="yes"
%}
```

*図 31. Net.Data サンプル・マクロ・ファイル (1/5)*

```
%{ ------------------------------------------------------------------------ %}
0%{ SQL functions                                                            %}
%{ ------------------------------------------------------------------------ %}
%function (DTW_SQL) startHereSQL(){
  select artist, title, stock_no, price from sobay_catalog
  %REPORT{
    <table border="2" bgcolor="#b1b1b1">
           <tr><th>Artist <th> Title <th> Stock<th> Number <th> Price </tr>
    %ROW{ <tr><td> $(V_artist) <td> $(V_title) <td>  $(V_stock_no) <td> $(V_price) <tr>
    %}
    </table>
   %}
%}
%function (DTW_SQL) addThumbsSQL(){
  select cast(mmdbsys.thumbnail(covers) as blob(10000)),
         cast(mmdbsys.thumbnail(video) as blob(3000)),
         mmdbsys.comment(music), artist, title, price, stock_no
  from sobay_catalog
  %REPORT{
   <table border="2" bgcolor="#b1b1b1">
                <tr><th>Cover <th>Video <th>Audio <th>Artist <th>Title <th>Price </tr>
    %ROW{  <tr><td>< a href="showCover?stock_no=$(V_stock_no)"><img src="$V1"></a>
               <td><  a href="getVideo?stock_no=$(V_stock_no)"><img src="$V2"></a>
               <td><  a href="getAudio?stock_no=$(V_stock_no)&filename=$V3")>[Listen]</a>
               <td> $(V_artist) <td> $(V_title) <td> $(V_price) </tr>
    %}
    </table>
  %}
%}
%function (DTW_SQL) showCoverSQL(){
  select cast(mmdbsys.content(covers, 'GIF') as blob(150000)), mmdbsys.format(covers)
    from sobay_catalog
   where stock_no = '$(stock_no)'
  %REPORT{
    %ROW{  <img src="$(V1) "> <br><br><b>Original image format: $(V2)</b>%}
  %}
%}
```

*図 31. Net.Data サンプル・マクロ・ファイル (2/5)*

```
%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}
%function (DTW_SQL) showVideoSQL(){
  select mmdbsys.comment(video), mmdbsys.content(video, mmdbsys.comment(video), 1),
         mmdbsys.format(video)
    from sobay_catalog
   where stock_no = '$(stock_no)'
  %REPORT{
    %ROW{ <a href="/tmplobs/$(V1)"><i><b> Play Video Clip></b></i></a>
       <br><br><b>Format: $(V3) <br>(Note: NT/Win95 may not come with
          a decompressor<br>for this video format.  OS/2 Warp does.)</br>
    %}
  %}
%}
%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}
%function (DTW_SQL) showAudioSQL(){
  select mmdbsys.comment(music), mmdbsys.content(music, mmdbsys.comment(music), 1),
         mmdbsys.format(music)
    from sobay_catalog
   where stock_no = '$(stock_no)'
  %REPORT{
      %ROW{ < a href="/tmplobs/$(V1) "<i><b>Play Audio Clip</b></i></a>
         <br><br><b>Format: $(V3)</b>
    %}
  %}
%}
```

*図 31. Net.Data サンプル・マクロ・ファイル (3/5)*

```
%{ ------------------------------------------------------------------------- %}
%{ HTML sections                                                              %}
%{   E.g., http://<your server>/cgi-bin/db2www/extender.d2w/startHere       %}
%{ ------------------------------------------------------------------------%}
%{   E.g., http://
%{   E.g., http://
%HTML(startHere){
<html>
   <head><title>UDB Extenders Macro Sample: Simple Row Listing</title></head>
   <body bgcolor="#ffffff">
   <font color="#3300ff" size="3"><b>If no data appears below, you might need
    to run the UDB Extender sample programs <i>enable</i> and <i>populate</i>.
    This first HTML section of the extender.d2w macro simply retrieves all the
    traditional data for all the rows in the UDB Extenders' sample database.
    %if ( "$(SHOWSQL)" == "yes" || "$(SHOWSQL)" == "YES" )
    <br><br> By default, every page generated by this macro shows the SQL used
      to generate that page.  Here is the SQL statement for this page:
    %else
      <br>
    %endif
    </b></font>
     <br>@startHereSQL()
    <br><b>Click < a href="addThumbs"><i>here</i></a> to display thumbnails
    and links to image/audio/video data.</b>
   </body>
</html>
%}
%HTML(addThumbs){
<html>
   <head><title>UDB Extenders Macro Sample: Add Thumbnails</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>This page adds album cover thumbnails
    and links to display the multimedia content of the database.  To access
    the multimedia content:
    <ul>
      <li> Click on a thumbnail of a CD cover to view a full-size image
      <li> Click on a "video thumbnail" to view a video
      <li> Click on a "[Listen]" link to listen to an audio clip
    </ul>
    </b></font> @addThumbsSQL()
    <br><b>Click < a href="startHere"><i>here</i></a> to go back to the first page.</b>
  </body>
</html>
%}
```

*図 31. Net.Data サンプル・マクロ・ファイル (4/5)*

```
%HTML(showCover){
<html>
   <head><title>UDB Extenders Macro Sample: Cover for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>For this page, the macro gets a full-size cover
image, converting the image format to GIF so that a browser can show it:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showCoverSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}
%HTML(getVideo){
<html>
   <head><title>UDB Extenders Macro Sample: Video clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can view a video clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showVideoSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}
%HTML(getAudio){
<html>
   <head><title>UDB Extenders Macro Sample: Audio clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can listen to an audio clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showAudioSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 <body>
</html>
%}
```

*図 31. Net.Data サンプル・マクロ・ファイル (5/5)*

# 付録C. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミングまたはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、 IBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品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。これらのプログラムまたは製品に代えて、 IBM の知的所有権を侵害することのない機能的に同等な他社のプログラム、製品またはサービスを使用することができます。ただし、IBM によって明示的に指定されたものを除き、これらのプログラムまたは製品に関連する稼働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 および他社は、本書で説明する主題に関する特許権 (特許出願を含む)、商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & 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。

本書に含まれる情報には、技術的に不正確なもの、または誤植が含まれる場合があります。これらに対する変更は、定期的に行われます。これらの変更は、資料の改訂版に含まれます。 IBM は、本書で説明している製品、プログラムに対して、予告なく改良、変更を加える場合があります。

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、お客様になんら義務も負わせない適切な方法で、使用もしくは配布することがあります。

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 Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

本プログラムに関する上記の情報は、適切な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。

本書に含まれるパフォーマンス・データは、制御された環境下で決定されています。したがいまして、その他の稼働環境で得られる結果とは、かなり異なる可能性もあります。一部の測定値は、開発中のシステムを使用している場合があり、これらの測定値が一般的に提供可能なシステムで同様の数値になることを保証するものではありません。さらに、一部の測定値が推定されたものもあります。実測値と異なる場合があります。本書のユーザーは、使用される特定の環境での該当データを確認してください。

IBM 以外の製品については、当該製品の提供者から直接、出版されている資料または一般公開されている情報から入手しました。 IBM は、これらの製品についてはテストを行っておらず、これらの IBM 以外の製品に関する性能、互換性またはその他の主張について確認することはできません。 IBM 以外の製品の機能に対する質問は、それぞれの製品提供者にお問い合わせください。

IBM の将来の方向性または意図については、予告なしに変更または中止する場合があり、 IBM の目的および目標のみ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が、これは説明に具体性を与えるために記載されたものであり、それらの例には、個人、企業、ブランドの、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。それらの名前はすべて架空のものであり、また名称や住所が類似する企業が実在しても、それは偶然に過ぎません。

著作権：

本書に含まれる情報には、サンプル・アプリケーション・プログラムがソース言語の形式で含まれており、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技法を示しています。お客様は、これらのサンプル・プログラムが書かれているオペレーティング・プラットフォームでアプリケーション・プログラミング・インターフェースが実行可能となるためのアプリケーション・プログラムを開発、使用、販売または配布もしくは転送する目的のためにのみ、サンプル・プログラムを、IBM に対する別途料金を支払うことなく、複製、変更、配布または転送することができます。これらのサンプルは、すべての条件下で十分にテストを行っていません。したがいまして、IBM は、これらのプログラムの信頼性、実用性または機能について、いかなる保証も負いません。

サンプル・プログラムまたはその改変版の複製物には、全部複製か部分複製かを問わず、次の著作権表示を必ず行うものとします。

## 商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

次のものは、他社の商標または登録商標です。

Tivoli およびその NetView は、米国ならびに他の国における Tivoli Systems Inc. の商標です。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

# 用語集

**管理サポート表 (administrative support table).** イメージ、オーディオ、およびビデオ・オブジェクトに対するユーザー要求を処理するために DB2 エクステンダーが使用する 1 つの表。管理サポート表には、エクステンダーで使用可能になっているユーザー表と列を識別するものがある。他の管理サポート表には、使用可能な列内のオブジェクトに関する属性情報が含まれている。*メタデータ表 (metadata table)* とも呼ばれる。

**分析 (analyze).** あるイメージ (画像) のフィーチャー (特徴) を表す数値を計算して、その値を QBIC カタログに追加すること。

**API.** *アプリケーション・プログラミング・インターフェース (application programming interface)* を参照。

**アプリケーション・プログラミング・インターフェース (Application programming interface、API).**

(1) オペレーティング・システムに、または個別に発注可能なライセンス・プログラムに用意されている機能インターフェース。 API によって、高水準言語で書かれたアプリケーション・プログラムは、オペレーティング・システムやライセンス・プログラムの特定のデータまたは機能を使うことができる。

(2) DB2 では、インターフェース内の機能 (たとえば、エラー・メッセージ取得 API)。

(3) DB2 エクステンダーは、ユーザー定義関数、管理操作、表示操作、およびビデオ・シーンの変化の検出を要求するための API を提供している。

**音声 (audio).** 再生して聞くことのできる、録音された情報。

**平均色 (average color).** イメージのピクセル (画素) に含まれるそれぞれの色の平均値として計算される色の尺度。

**オーディオ・クリップ (audio clip).** 録音された音声データの一節。

**2 進ラージ・オブジェクト (binary large object、BLOB).** 最大 2 GB の長さまで有効な 2 進ストリング。イメージ、オーディオ、およびビデオ・オブジェクトは、DB2 データベースでは BLOB として保管される。

**文字ラージ・オブジェクト (character large object、CLOB).** 最大 2 GB まで有効な 1 バイト文字の文字ストリング。 CLOB には、関連するコード・ページがある。 1 バイト文字を含むテキスト・オブジェクトは、DB2 データベースでは CLOB として保管される。

**きめの粗さ (coarseness).** きめの細かさ (小石と大石など) を示すテクスチャー *(texture)* の属性。

**コントラスト (contrast).** テクスチャー *(texture)* の属性の 1 つで、模様の鮮明度を示す。グレー・レベル・ヒストグラムの変動を表す関数でもある。

**データベース区画 (database partition).** 固有のユーザー・データ、索引、構成ファイル、およびトランザクション・ログからなるデータベースの部分。ノードまたはデータベース・ノードということもある。

**データベース区画サーバー (database partition server).** *データベース区画 (database partition)* を管理するサーバー。データベース区画サーバーは、データベース・マネージャーと、データベース・マネージャーの管理するデータやシステム資

源から構成される。通常は、1 台のマシンに 1 つのデータベース区画サーバーが割り当てられる。

**DB2 エクステンダー (DB2 extender).** 従来の数値データや文字データ以外のデータ・タイプ (たとえばイメージ、オーディオ、ビデオ・データ) および複雑な構造の文書の保管や検索を可能にする、複数のプログラムからなるグループ。

**方向性 (directionality).** テクスチャー *(texture)* の属性の 1 つで、(たとえば草のように) イメージに特定の望ましい向きがあるか、それとも (ガラスのように) 平たんなオブジェクトであるかを示す。

**ディゾルブ (dissolve).** 次のビデオ・フレームのシグナルの強度が増すにつれて、ビデオ・フレームのシグナルの強度を弱めること。

**特殊タイプ (distinct type).** *ユーザー定義タイプ (user-defined type)* を参照。

**2 バイト文字ラージ・オブジェクト (double-byte character large object、DBCLOB).** 2 バイト文字からなる文字ストリング、または単一バイト文字と 2 バイト文字の組み合わせによる文字ストリングで、ストリングの長さの上限は 2 GB。各 DBCLOB には、関連した 1 つのコード・ページがある。 2 バイト文字を含むテキスト・オブジェクトは、DB2 データベースでは DBCLOB として保管される。

**環境変数 (environment variable).** DB2 エクステンダーの稼働環境を表し、環境値のデフォルトを提供する変数。

**エクステンダー (extender).** *DB2 エクステンダー (DB2 extender)* を参照。

**フィーチャー (feature).** イメージの目に見える属性、たとえば*平均色 (average color)*。

**ファイル参照変数 (file reference variable).** プログラミング変数の 1 つで、クライアント・ワークステーション上のファイルに LOB を移動したり戻したりするのに使用できる。

**ギガバイト (gigabyte、GB).** 10 億 ($10^9$) バイト。メモリー容量の場合は、1 073 741 824 バイト。

**ハンドル (handle).** 表内のイメージ、オーディオ、またはビデオ・オブジェクトを表すために使う、エクステンダーによって作成される文字ストリング。各オブジェクトのハンドルが、ユーザー表および*管理サポート表 (administrative support table)* に保管される。この方法でエクステンダーは、ユーザー表に保管されているハンドルと、管理サポート表に保管されているオブジェクトに関する情報とをリンクする。

**ヒストグラム色 (histogram color).** あるイメージ内のそれぞれの色ごとの尺度。それぞれの色についてのデータは、*QBIC カタログ (QBIC catalog)* に別々に保管される。

**ホスト変数 (host variable).** 組み込み SQL ステートメントで参照可能なアプリケーション・プログラムの変数。ホスト変数は、データベースとアプリケーション・プログラムの作業域との間でデータをやりとりする際の基本的なメカニズムである。

**画像 (image).** ピクチャーを電子的に表現したもの。

**索引ファイル (index file).** ビデオ・エクステンダーがショット *(shot)* やビデオ・クリップの個々のフレームを検出するときに使用する索引情報が入っているファイル。

**インスタンス (instance).** 論理的な DB2 エクステンダーのサーバー環境。同一のワークステーション上に DB2 エクステンダー・サーバーのインスタンスを複数持つことができるが、個々の DB2

| インスタンスには 1 つのインスタンスしか持てない。これらのインスタンスを使用すると、次のことが可能である。

- 開発環境と実稼働環境を分ける
- 機密情報へのアクセスを特定の人々のみに制限する

**キロバイト (kilobyte、KB).** 千 ($10^3$) バイト。メモリー容量の場合は 1024 バイト。

**ラージ・オブジェクト (large object、LOB).** 長さの上限が 2 GB のバイト順序列。 LOB には、*2 進ラージ・オブジェクト (BLOB)*、*文字ラージ・オブジェクト (CLOB)*、および *2 バイト文字ラージ・オブジェクト (DBCLOB)* の 3 つのタイプがある。

**LOB ロケーター (LOB locator).** ホスト変数の中に保管される短精度 (4 バイトの) 値で、プログラムはこれを使用して DB2 データベース内のより大きな LOB を参照する。 LOB ロケーターを使用すると、ユーザーは LOB が通常のホスト変数の中に保管されているかのように操作でき、クライアント・マシン上のアプリケーションとデータベース・サーバーとの間で LOB を移送する必要がない。

**メガバイト (megabyte、MB).** 百万 ($10^6$) バイト。メモリー容量の場合は 1 048 576 バイト。

**メタデータ表 (metadata table).** *管理サポート表 (administrative support table)* を参照。

**複数区画ノード・グループ (multipartition nodegroup).** 複数の*データベース区画サーバー (database partition server)* を含む*ノード・グループ (nodegroup)*。

**ノード (node).** データベース区画化では、データベース区画 (database partition) と同義。

**ノードグループ (nodegroup).** 1 つまたは複数のデータベース区画をまとめて名前を付けたグループ。

**オブジェクト (object).** オブジェクト指向プログラミングにおいて、あるデータとそれに関連付けられた操作からなる抽象的な実体。

**オブジェクト指向 (object orientation).** 実際のものか抽象的な実体かを問わず、アプリケーション内のすべてのものを、(一連の操作およびデータ値からなる) オブジェクトとして表すプログラミング上のアプローチ。たとえば、1 つの文書は、文書データとその文書に対して実行できる操作 (ファイリング、送信、印刷など) からなる 1 つの文書オブジェクトとして表すことができる。 1 つのビデオ・クリップは、ビデオ・データと (ビデオ・クリップの再生や特定のビデオ・フレームの検索などの) 一連の操作からなる 1 つのビデオ・オブジェクトとして表すことができる。

**区分データベース (partitioned database).** 複数のデータベース区画をもつデータベース。ユーザー表のデータは、1 つまたは複数のデータベース区画に置くことができる。 1 つの表が複数の区画に分散している場合、一部の行がある区画に保管され、それ以外の行が他の区画に保管される。

**ピクセル (pixel).** イメージにおいて、画面で表示できる最も小さい要素。

**定位置色 (positional color).** イメージ内の特定領域におけるピクセルの*平均色 (average color)* の値。

**イメージ内容による照会 (Query by Image Content、QBIC).** イメージ・エクステンダーが提供する機能。この機能を使用すると、ユーザーは*平均色*、テクスチャー などの可視特性によって画像を探索できるようになる。

**QBIC カタログ (QBIC catalog).** イメージの視覚的な特徴についてのデータを保持しているリポジトリー。

**照会オブジェクト (query object).** QBIC 照会に関するフィーチャー、フィーチャーの値、およびフィーチャーの重みを指定するオブジェクト。

オブジェクトに名前を付けて保管し、後で QBIC 照会で使用することができる。照会ストリング (query string) と対比。

**照会ストリング (query string).** QBIC 照会に関するフィーチャー、フィーチャーの値、およびフィーチャーの重みを指定する文字ストリング。照会ストリングは DB2 コマンド行からの照会で入力することができる。照会オブジェクト (query object) と対比。

**スケーリング (scaling).** 記憶スペースを増やしたりパフォーマンスを改善するために、データベースに*ノード (node)* を追加すること。

**シーンの変化 (scene change).** 連続する 2 つのフレームの間に大きな違いが見られる、*ビデオ・クリップ (video clip)* 内の 1 点。たとえば、ビデオの撮影中にカメラが視点を変えると、これが発生する。

**スコア (得点)(score).** フィーチャー値が、イメージ内容による照会で指定された値とどの程度類似しているかを示す計算値。数値が大きいほど、マッチの度合いが高い。スコアは、イメージ内容による照会の結果をソートするのに使われる。

**ショット (shot).** 2 つのシーン変化間の一連のフレーム。

**ショット・カタログ (shot catalog).** ビデオ・クリップ内のショットに関するデータ (たとえば、ショットの開始フレーム番号と終了フレーム番号) の保管に使われるデータベース表またはファイル。ユーザーは SQL 照会を介して表の視点にアクセスしたり、またはファイル内のデータにアクセスすることができる。

**ストーリーボード (storyboard).** ビデオの可視的な要約。ビデオ・エクステンダーには、ビデオ内の代表ショットであるビデオ・フレームを識別して保管する機能がある。これらの代表フレームを使用して、ストーリーボードを作成することができる。

**テラバイト (terabyte).** 1,000,000,000,000 ($10^{12}$) バイト。 10 の 12 乗バイト。メモリー容量の場合は、1 099 511 627 776 バイト。

**テクスチャー (texture).** イメージ内容による照会に使用できるフィーチャーの 1 つ。イメージのきめの粗さ、コントラスト、または方向性を表す。

**サムネール (thumbnail).** イメージの縮小版。

**トリガー (trigger).** 表が変更される時に実行される一連のアクションの定義。トリガーを使って実行できるアクションには、入力データの妥当性検査、新たに挿入された行の値の自動生成、相互参照のための他の表の読み取り、または監査のための他の表への書き込みがある。トリガーは、保全性を検査したり、業務上の規則を適用するためにしばしば使用される。

**UDF.** *ユーザー定義関数 (user-defined function)* を参照。

**UDT.** *ユーザー定義タイプ (user-defined type)* を参照。

**ユーザー定義関数 (user-defined function、UDF).** ユーザーが DB2 に定義する関数。関数が定義されると、SQL 照会やビデオ・オブジェクトで使用できる。たとえば、ビデオの圧縮形式を得る UDF や、オーディオのサンプリング・レートを戻す UDF を作成できる。この方法を使えば、特定タイプのオブジェクトに対しその動作を定義することができる。

**ユーザー定義タイプ (user-defined type、UDT).** ユーザーが DB2 に定義するデータ・タイプ。 UDT は 1 つの LOB を別の LOB と区別するために使用される。たとえば、イメージ・オブジェクト用とオーディオ・オブジェクト用とに、別の UDT を作成することができる。こうすることによって、イメージ・オブジェクトとオーディオ・オブジェクトは BLOB として保管されるものの、BLOB とは違うタイプとして、また互いに違うタイプとして扱われる。

**ビデオ (video).** 再生して見ることのできる、録画された情報を指す。

**ビデオ・クリップ (video clip).** フィルム撮影またはビデオ録画されたデータの 1 つのセクション。

**ビデオ索引 (video index).** ビデオ・エクステンダーが、ビデオ・クリップの特定のショット *(shot)* やフレームを検索するときに使用するファイル。

# 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］



## ［数字］

## A



## B



## C



## D

## E



## F



## G



## H



## I



## L



## M

## N



## O



## Q



## R



## S



## T



## U

## W

# IBM と連絡をとる

技術上の問題がある場合は、時間をとって「問題判別の手引き」に定義されている処置を検討し、それらの提案を実行した後で、DB2 顧客サービスに連絡をとってください。この資料には、DB2 顧客サービスがお客さまを支援するために必要とする情報が説明されています。

## 製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

**http://www.ibm.com/software/data/**

DB2 World Wide Web ページには、ニュース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供されています。ただし、提供されている情報は英語です。

**http://www.ibm.com/software/data/db2/library/**

「DB2 Product and Service Technical Library」では、よくされる質問 (FAQ)、修正内容、資料、および最新の DB2 技術情報などの情報へのアクセスが提供されています。

**注:** この情報のご提供は英語のみとなりますのでご注意ください。

**http://www.elink.ibmlink.ibm.com/pbl/pbl/**

「International Publications」注文用 Web サイトでは、マニュアルの注文方法についての情報を提供しています。ただし、提供されている情報は英語です。

**http://www.ibm.com/education/certify/**

IBM の「Professional Certification Program」Web サイトでは、 DB2 を含むさまざまな IBM 製品の認証テストの情報を提供しています。ただし、提供されている情報は英語です。

**ftp.software.ibm.com**

匿名でログオンしてください。ディレクトリー `/ps/products/db2` には、 DB2 および多数の他製品に関連したデモ、修正プログラム、情報、およびツールがあります。ただし、提供されている情報は英語です。

**comp.databases.ibm-db2, bit.listserv.db2-l**

これらのインターネット・ニュースグループは、ユーザーが DB2 製品に関する自分の経験について話し合うために利用できます。ただし、提供されている情報は英語です。

**Compuserve: GO IBMDB2**

このコマンドを入力すると、IBM DB2 Family forum にアクセスできます。すべての DB2 製品が、このフォーラムでサポートされています。ただし、提供されている情報は英語です。

米国以外の国で IBM に連絡する方法については、 IBM Software Support Handbook の Appendix A を参照してください。この資料にアクセスするには、 Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

**注:** 国によっては、 IBM が承認している販売業者が、 IBM サポート・センターの代わりにそれら販売業者のサポート・センターに連絡する場合があります。

Printed in Japan

SC88-8609-00

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