# DBMaster

## DBMaster 入門編

SYSCOM

CASEMaker Inc./Corporate Headquarters

1680 Civic Center Drive
Santa Clara, CA 95050, U.S.A.

**Contact Information:**
CASEMaker US Division
E-mail : info@casemaker.com
Europe Division
E-mail : casemaker.europe@casemaker.com
Asia Division
E-mail : casemaker.asia@casemaker.com(Taiwan)
E-mail : info@casemaker.co.jp(Japan)

www.casemaker.com
www.casemaker.com/support


Document No. 645049-231802/DBM50J-M01312008-TUTO
発行日:2008-01-31



本文には記されていない新しい機能についての説明は、CASEMakerのDBMasterをインストールしてからREADME.TXTを読んでください。

**登録商標**
CASEMaker、CASEMakerのロゴは、CASEMaker社の商標または登録商標です。
DBMasterは、Syscom Computer Engineering社の商標または登録商標です。
Microsoft、MS-DOS、Windows、Windows NTは、Microsoft社の商標または登録商標です。
UNIXは、The Open Groupの商標または登録商標です。
ANSIは、American National Standards Institute, Incの商標または登録商標です。

ここで使用されているその他の製品名は、その所有者の商標または登録商標で、情報として記述しているだけです。SQLは、工業用語であって、いかなる企業、企業集団、組織、組織集団の所有物でもありません。

**注意事項**
本書で記述されるソフトウェアは、ソフトウェアと共に提供される使用許諾書に基づきます。

保証については、ご利用の販売店にお問い合わせ下さい。販売店は、特定用途への本コンピュータ製品の商品性や適合性について、代表または保証しません。販売店は、突然の衝撃、過度の熱、冷気、湿度等の外的な要因による本コンピュータ製品へ生じたいかなる損害に対しても責任を負いません。不正な電圧や不適合なハードウェアやソフトウェアによってもたらされた損失や損害も同様です。

本書の記載情報は、その内容について十分精査していますが、その誤りについて責任を負うものではありません。本書は、事前の通知無く変更することがあります。

# 目次

1 はじめに .....1-1
1.1 その他のマニュアル.....1-3
1.2 字体の規則 .....1-4
2 DBMSの長所 .....2-1
2.1 構文ダイアグラム .....2-2
2.2 データベース管理システム.....2-3
2.3 データ・モデル.....2-4
2.4 データの独立性.....2-4
物理的データ独立性.....2-4
論理的データ独立性.....2-5
2.5 高水準言語サポート.....2-6
2.6 トランザクション管理.....2-6
トランザクションとは.....2-6
同時実行制御.....2-7
ロックの概念.....2-8
2.7 整合性制御 .....2-9

2.8 アクセス制御 ..... 2-10
ユーザー権限 ..... 2-10
トランザクション権限 ..... 2-11
2.9 DBMSリカバリ ..... 2-11
システム障害 ..... 2-11
メディア障害 ..... 2-12
3 DBMSアーキテクチャ ..... 3-1
3.1 DBMSの論理アーキテクチャ ..... 3-1
内部/物理的レベル ..... 3-2
概念レベル ..... 3-2
外部レベル/ビューレベル ..... 3-3
レベル間のマップ ..... 3-3
3.2 DBMSの物理アーキテクチャ ..... 3-4
アプリケーションとユーティリティ ..... 3-5
アプリケーション・プログラム・インターフェース(API)3-8
問合せ言語プロセッサ ..... 3-8
DBMSエンジン ..... 3-9
4 データベース ..... 4-1
4.1 ネーミングの規則 ..... 4-2
4.2 dmconfig.iniファイル ..... 4-2
作成 ..... 4-3
ディレクトリ ..... 4-4
フォーマット ..... 4-4
セクション名 ..... 4-5
キーワード ..... 4-5
コメント ..... 4-5
4.3 dmSQL ..... 4-6
起動 ..... 4-6

作業スペース ..... 4-7
**4.4 Jツール ..... 4-8**
JConfiguration Tool ..... 4-8
JServer Manager ..... 4-9
JDBA Tool ..... 4-9
**4.5 データベースの作成 ..... 4-9**
練習データベース ..... 4-10
接続ハンドル ..... 4-10
初期設定ユーザー ..... 4-11
**4.6 データベース・モード ..... 4-12**
シングルユーザー・モード ..... 4-13
マルチユーザー・モード ..... 4-13
クライアント/サーバー・モード ..... 4-13
**5 表 ..... 5-1**
**5.1 表領域 ..... 5-1**
標準表領域 ..... 5-1
自動拡張表領域 ..... 5-2
システム表領域 ..... 5-2
初期設定ユーザー表領域 ..... 5-2
**5.2 データ型 ..... 5-2**
CHAR (サイズ) ..... 5-3
VARCHAR (サイズ) ..... 5-3
BINARY (サイズ) ..... 5-3
SMALLINT ..... 5-4
INTEGER ..... 5-4
FLOAT ..... 5-5
DOUBLE ..... 5-5
DECIMAL (NUMERIC) ..... 5-5
DATE ..... 5-6
TIME ..... 5-6

TIMESTAMP ..... 5-7
LONG VARCHAR(CLOB) ..... 5-7
LONG BINARY(BLOB) ..... 5-8
SERIAL (開始番号) ..... 5-8
FILE ..... 5-9
OID ..... 5-10
**5.3 表の作成 ..... 5-10**
カラムの初期設定値 ..... 5-13
ロック・モード ..... 5-13
フィルファクタ ..... 5-14
非キャッシュ ..... 5-14
一時表 ..... 5-15
**6 データ ..... 6-1**
**6.1 挿入 ..... 6-1**
ホスト変数を使った挿入 ..... 6-3
様々なデータ型 ..... 6-4
BLOBデータの挿入 ..... 6-5
**6.2 更新 ..... 6-6**
標準SQLを使った更新 ..... 6-6
ホスト変数を使った更新 ..... 6-7
OIDを使った更新 ..... 6-7
**6.3 結果セット ..... 6-8**
表の選択 ..... 6-8
カラムの選択 ..... 6-11
行の選択 ..... 6-11
**6.4 演算子の種類 ..... 6-12**
比較演算子 ..... 6-12
論理演算子 ..... 6-14
算術演算子 ..... 6-15

6.5 削除 ..... 6-15
標準SQLを使った削除 ..... 6-15
ホスト変数を使った削除 ..... 6-16
OIDを使った削除 ..... 6-16

7 データベース･オブジェクト ..... 7-1
7.1 ビュー ..... 7-1
ビューの作成 ..... 7-2
ビューの削除 ..... 7-2
7.2 シノニム ..... 7-3
シノニムの作成 ..... 7-3
シノニムの削除 ..... 7-4
7.3 索引 ..... 7-4
索引の作成 ..... 7-5
索引の削除 ..... 7-6

8 ユーザーと権限 ..... 8-1
8.1 セキュリティ管理 ..... 8-1
8.2 権限レベル ..... 8-2
CONNECT権限 ..... 8-2
RESOURCE権限 ..... 8-2
DBA権限 ..... 8-3
SYSADM権限 ..... 8-3
8.3 新規ユーザー ..... 8-3
ユーザー・アクセス ..... 8-4
複数のユーザーの作成 ..... 8-5
8.4 権限レベルを上げる ..... 8-5
複数のユーザーの権限を上げる ..... 8-6
8.5 権限レベルを下げる ..... 8-6

8.6 ユーザーを削除する ..... 8-7
8.7 パスワード ..... 8-7
8.8 グループを管理する ..... 8-9
グループの作成 ..... 8-9
メンバーの追加 ..... 8-10
メンバーの削除 ..... 8-10
グループの削除 ..... 8-11
入れ子グループ ..... 8-11
8.9 表レベル権限 ..... 8-11
SELECT（選択） ..... 8-12
INSERT（挿入） ..... 8-12
DELETE（削除） ..... 8-12
UPDATE（更新） ..... 8-12
INDEX（索引） ..... 8-12
ALTER（修正） ..... 8-12
REFERENCE（参照） ..... 8-13
8.10 表権限を与える ..... 8-13
表全体への表権限を与える ..... 8-14
特定カラムへの表権限を与える ..... 8-15
8.11 表権限を取り消す ..... 8-16
表全体への権限を取り消す ..... 8-16
特定カラムへの表権限を取り消す ..... 8-17
9 データベース・リカバリ ..... 9-1
9.1 障害の種類 ..... 9-1
システム障害 ..... 9-1
メディア障害 ..... 9-2
9.2 リカバリの方法 ..... 9-2
ジャーナル・ファイル ..... 9-2

チェックポイント・イベント ..... 9-3
リカバリ処理 ..... 9-4
**9.3 バックアップの種類 ..... 9-5**
完全バックアップ ..... 9-5
差分バックアップ ..... 9-6
オフライン・バックアップ ..... 9-6
オンライン・バックアップ ..... 9-7
バックアップの組み合わせ ..... 9-7
**9.4 バックアップ・モード ..... 9-8**
NONBACKUPモード ..... 9-9
BACKUP-DATAモード ..... 9-9
BACKUP-DATA-AND-BLOBモード ..... 9-9
表領域BLOBバックアップ・モード ..... 9-10
ファイルオブジェクト・バックアップ・モード ..... 9-11
バックアップ・モードの設定 ..... 9-13
**9.5 オフライン完全バックアップ ..... 9-16**
dmSQLを使ったオフライン完全バックアップ ..... 9-16
JServer Managerを使ったオフライン完全バックアップ 9-17
**9.6 バックアップ・サーバー ..... 9-18**
バックアップ・サーバーを起動する ..... 9-19
差分バックアップのファイル名形式を設定する ..... 9-21
バックアップ・ディレクトリを設定する ..... 9-24
古いディレクトリを設定する ..... 9-27
差分バックアップ設定 ..... 9-29
ジャーナル・トリガー値を設定する ..... 9-32
コンパクト・バックアップ・モードを設定する ..... 9-36
完全バックアップ・スケジュール ..... 9-39
ファイルオブジェクトのバックアップ・モード ..... 9-41
バックアップ・サーバーを停止する ..... 9-46

**9.7 バックアップ履歴ファイル.....9-47**
バックアップ履歴ファイルの割り当て.....9-47
バックアップ履歴ファイルの要素.....9-48
バックアップ履歴ファイルの使用.....9-48
ファイルオブジェクトのバックアップ履歴ファイル..9-49
**9.8 リストアの選択肢.....9-49**
リストア方法の判断.....9-50
リストアの準備.....9-50
リストアの実行.....9-50

# 1 はじめに

**CASEMaker**製品をご利用頂きありがとうございます。DBMasterは、強力かつ柔軟なSQLデータベース管理システム(DBMS)です。会話型構造の問い合わせ言語(SQL)、Microsoftのオープンデータベース結合(ODBC)互換インターフェース、およびC言語対応埋め込みSQL(ESQL/C)をサポートします。唯一の公開アーキテクチャでネイティブなODBCインターフェースは、多種多様なプログラミングツールを使用して顧客アプリケーションを構築し、既存のODBC-適合アプリケーションを用いたデータベースへの問い合わせを可能にします。

DBMasterは、シングルユーザーの個人データベースから、企業全体に分散するデータベースまでに容易にスケール化することができます。どのようなデータベース構成を選択しても、重要データの安全性は、DBMasterのセキュリティ、整合性、信頼性の先進的機能によって確実に保証されます。広範なクロス-プラットフォームのサポートは、現在あるハードウェアの性能を高め、需要の変化に応じてより強力なハードウェアに拡大し、グレードアップすることを可能にします。

DBMasterは、優れたマルチメディア処理機能を提供し、あらゆるタイプのマルチメディアデータを保存、検索、回収、操作を可能にします。バイナリ・ラージオブジェクト(BLOB)は、DBMasterの先進的セキュリティと損傷リカバリ機能を全面的に利用して、マルチメディアデータの整合性を確実にします。ファイル・オブジェクト(FO)は、マルチメディアデータを管理する一方で、既存のアプリケーションで各ファイルを編集できる機能を保持します。

本マニュアルは、リレーショナル・データベースやSQL言語にあまり詳しくないエンドユーザー向けに書かれたものです。但し、コンピューターの一般的動作に関する知識と、DBMasterを運用するオペレーション・システムを難なく使いこなせる知識を必要とします。オペレーティング・システムに関する情報は、本マニュアルの対象外ですので、お使いのオペレーティング・システムのマニュアルを参照して下さい。

本マニュアルでは、DBMasterを使って作成/維持されるデータベースの組織や構造といったことを理解するための概念や原理についての一般的な事項を説明します。本書では、例や図を用いて見やすく、又一つのトピックスごとにわかりやすく説明します。

本書は、DBMasterで用いるSQL言語について説明します。そのSQL文が最初に登場する時、その構文ダイアグラムについて説明します。構文ダイアグラムは、各SQL文に使用できるオプションや構文のバリエーションを一目でわかるように解説します。SQL文の説明には、多くの例や注意すべき重要なポイントも合わせて掲載しています。

本書で紹介するSQL文と例の多くに、DBMasterのdmSQLコマンドライン・ツールを使っています。これ以外のDBMasterのアプリケーション・ツールやユーティリティを使って操作することもあります。DBMasterのアプリケーション・ツールとユーティリティについての詳細は、「その他のマニュアル」をご参照ください。

## 1.1 その他のマニュアル

DBMasterには、本マニュアル以外にも多くのユーザーガイドや参照編があります。特定のテーマについての詳細は、以下の書籍を参照して下さい。

- DBMasterの設計、管理、保守についての詳細は、*「データベース管理者参照編」*をご覧下さい。
- DBMasterの管理についての詳細は、*「JServer Managerユーザーガイド」*を参照して下さい。
- DBMasterの環境設定についての詳細は、*「JConfiguration Tool参照編」*をご覧下さい。
- DBMasterの機能についての詳細は、*「JDBA Toolユーザーガイド」*を参照して下さい。
- DBMasterで使用しているdmSQLのインターフェースについての詳細は、*「dmSQLユーザーガイド」*を参照して下さい。
- DBMasterで採用しているSQL言語についての詳細は、*「SQL文と関数参照編」*を参照して下さい。
- ESQLプログラムについての詳細は、*「ESQL/Cプログラマー参照編」*をご覧下さい。
- ODBCプログラムについての詳細は、*「ODBCプログラマー参照編」*をご覧下さい。
- エラーと警告メッセージについての詳細は、*「エラー・メッセージ参照編」*をご覧下さい。
- ネイティブDCI APIについての詳細は、*「DCI ユーザーガイド」*を参照して下さい。

## 1.2 字体の規則

本書は、標準の字体規則を使用しているので、簡単かつ明確に読むことができます。

| | |
|---|---|
| *斜体* | 斜体は、ユーザー名や表名のような特定の情報を表します。斜体の文字そのものを入力せず、実際に使用する名前をそこに置き換えてください。斜体は、新しく登場した用語や文字を強調する場合にも使用します。 |
| **太字** | 太字は、ファイル名、データベース名、表名、カラム名、関数名やその他同様なケースに使用します。操作の手順においてメニューのコマンドを強調する場合にも、使用します。 |
| キーワード | 文中で使用するSQL言語のキーワードは、すべて英大文字で表現します。 |
| 小さい英大文字 | 小さい英大文字は、キーボードのキーを示します。２つのキー間のプラス記号 (+) は、最初のキーを押したまま次のキーを押すことを示します。キーの間のコンマ(,)は、最初のキーを放してから次のキーを押すことを示します。 |
| 注 | 重要な情報を意味します。 |
| **➲ プロシージャ** | 一連の手順や連続的な事項を表します。ほとんどの作業は、この書式で解説されます。ユーザーが行う論理的な処理の順序です。 |
| **➲ 例** | 解説をよりわかりやすくするために与えられる例です。一般的に画面に表示されるテキストと共に表示されます。 |
| コマンドライン | 画面に表示されるテキストを意味します。この書式は、一般的にdmSQLコマンドやdmconfig.iniファイルの内容の入/出力を表示します。 |

# 2 DBMSの長所

リレーショナル・データベースの概念や原理、又はSQL言語に不慣れの方は、他のDBMasterマニュアルの前に本書をお読みください。

本書をくまなく読んで、各章にある例で練習データベースを完成して下さい。各例は、そのプロシージャに従って実行して下さい。例を省略すると次のステップで予想外の結果やエラーが起こる可能性があります。

既にデータベースを熟知していて、ある章の特定の項目についてのみ参照する場合は、まずその章で利用するために練習データベースを作成して下さい。それから本書にあるスクリプト・ファイルの一つを実行します。

今日のコンピューター・システムの最も重要な役割の一つは、データの保管と管理です。データには、事象や統計、写真やマルチメディアといったものがあります。一般的に、データベースはある特定のテーマについての収集した情報です。

コンピューターの利用が広がる以前、情報はファイル・フォルダとファイル・キャビネットに保管していました。情報を集めるためにファイル・キャビネットに行き、ファイルを取って、フォルダの中にある情報を見つけました。情報量が大きくなるにつれて、データを適時に集めるのは大変な仕事になってきました。どこに情報があるのかといったことを覚えることすら難しくなったのです。

コンピューターを使った情報管理が始まった頃、ファイルは特定のフォーマットで保管されていました。ファイルがどこにあるかを記憶し、そのファイルからどうやってデータを取り出すかについての知識が必要でした。

ファイルから情報を取り出すためには、データを回収するための特別なプログラムが必要でした。一旦このプログラムを実行すると、すばやくデー

タを回収できますが、データの保管方法を変更した場合や、別のデータを探したいという時、新しいプログラムが必要になりました。

企業の情報システム部門のプログラマーが、通常このようなプログラムを作成していました。コンピューターで扱える情報量が増大するにつれて、データを閲覧するための新しく多種多様な方法への欲求も高まりました。プログラムのリクエストが処理しきれない状態が一般化し、新しいプログラムの遅れは数週間、時には数ヶ月にもなりました。これが、独立したデータ保管システムと新しいデータ回収方法のニーズへとつながりました。

## 2.1 構文ダイアグラム

構文ダイアグラムは、SQL文の構文を表します。構文ダイアグラムは、SQL文の作成時にその構文オプションを調べる場合に役立ちます。構文ダイアグラムの例を以下に図示します。

構文ダイアグラムは、始点から終点までの線をたどって使用します。進路上にあるSQL文の要素は必須のものです。進路から分岐している要素は、オプションやフレキシブルな構文を示します。


*図 2-1 サンプル構文ダイアグラム*

*斜体*の文字の部分には、データベースで使用している実際の名前を指定します。上のダイアグラムでは、<*表名*>をデータベースにある表の名前に置き換えます。例えば**tutorial**データベースの場合、<*表名*>を**Customers**に置き換えると、**Customers**表にこのSQL文を実行します。

矢印の向きにも注意する必要があります。構文ダイアグラムがループになっている場所では、SQL文の中に複数のアイテムを指定することができます。上記の<*カラム名*>には、カンマで区切った複数のカラム名を含むことができます。

## 2.2 データベース管理システム

データベース管理システムは、データベースに保存されたデータを管理します。さらにデータを維持し、需要に応じてデータを利用できるようにする記録管理システムです。DBMasterはその傑出した機能性で、強力で柔軟性のあるシステムを実現し、他の情報システムの基礎を成します。

標準的なデータベース管理システムには、多くの特殊な機能があります。

- **データ・モデル**—ユーザーが理解しやすい様に、データを置き換えた概念です。データ・モデルは、実際は数学的な抽象概念です。これにより、データベース内で表される全データは、ユーザーが利用でき、目に見える形になります。
- **データの独立性**—データベース内でデータを物理的な格納構造の変更から分離することです。データベースの物理的な格納構造が変更した場合でも、データへの具体的なリクエストは、正しく返されます。
- **高水準言語サポート**—データベースの情報には、高水準言語でアクセスします。この言語を使えば、データベースの物理的な格納構造を知らなくても、ユーザーがデータを定義し、アクセスし、操作することが可能です。
- **トランザクション管理**—同一データへの複数のトランザクションが相互に干渉しないようにする手法です。これにより、複数のユーザーが同時にデータベースを利用することができます。
- **整合性制御**—データベースのデータの無効値や不整合を排除するようにします。これにより、ユーザーが不用意に無効なデータを入力したり、データの独立性を脅かす操作を実行したりすることを防ぎます。
- **アクセス制御**—不正ユーザーからデータのセキュリティとプライバシーを守る機能です。これにより、不正ユーザーが貴重なデータにアクセスしたり、閲覧したりすることを防ぎます。
- **リカバリ手法**—システム障害が起こった際に、データのバックアップやリストアをする方法です。

## 2.3 データ・モデル

データがどのようにコンピューターに物理的に保管されているか、おそらくユーザーにとってはほとんど、或いは全く重要ではないでしょう。どのデータも、複数のファイルやディスクにまたがった単純なバイナリ数として保管されているはずです。DBMSは*データ・モデル*を利用して、保管データをユーザーに理解しやすく意味のある形式に置き換えます。

データ・モデルは、データに構造的にアクセスする手段を提供する、データの数学的な抽象概念です。これによって、ユーザーによるすばやい操作やデータ回収と、データの場所や保管方法を記憶するといったことに悩まされないアプリケーション・プログラムを実現します。

数多くのデータ・モデルがありますが、現在もっとも広く使われているのはリレーショナル・データベース・モデルです。DBMasterでも、利用しているデータ・モデルです。リレーショナル・データベース・モデルは、行とカラム(列)で構成される表で、ユーザーにわかりやすい形で情報を提供します。各行には1つのテーマやアイテムのデータが、各カラムにはそのテーマやアイテムの属性（名前、サイズ等）があります。

## 2.4 データの独立性

DBMS最大の長所の1つに、*データの独立性*があります。データの独立性とは、データベースの構造を変更する際に、アプリケーション・プログラムやデータへのアクセス方法の変更が必要ないということです。データの独立性には、*物理的データ独立性*と*論理的データ独立性*の2種類あります。

### 物理的データ独立性

ファイルをベースとした初期のシステムでは、特定のフォーマットでファイルに情報を保管していました。ファイルからデータを取り出すためには、そのファイルのフォーマット知識のあるプログラマーがプログラムを作成しなければなりませんでした。データ構造に変更があった場合は、新しい構造からデータを読めるようにするために、そのプログラムも適切なものに変更する必要がありました。ユーザーが新しい方法でデータを閲覧する必要がある場合には、新しいプログラムが必要でした。データ組織や

データ回収のためのアクセス技術がアプリケーションのロジックとコードの中に組み込まれている、このようなシステムはデータが独立していません。

DBMSを使えば、アプリケーション・プログラムに影響を与えたり、データの閲覧方法を変更したりすることなく、データベースの物理構造を変えることができます。この変更によって、アプリケーション・プログラムの速度や能率に影響を及ぼすかもしれませんが、ユーザー・プログラムは変更する必要がありません。これは、DBMSがデータベースの物理構造を、ユーザーとアプリケーション・プログラム双方向に透過させるデータ・モデルの抽象概念を利用しているからです。データは、物理的なディスクへの保存/アクセス形態から、外部世界で使用する表現やアクセス技術や論理ビューに変換されます。

物理構造を変更しても、DBMSは同じ論理ビューを使い続けます。なぜなら、外部世界に代表される論理ビューは普遍性を維持し、データの論理ビューに基づくアプリケーション・プログラムとユーザーの相互作用は、物理構造の修正によって変更する必要がないからです。それゆえ、DBMSには*物理的にデータの独立性*があります。

## 論理的データ独立性

データの物理構造を変更しなければならない時がありますが、既存データの論理ビューが同じである限り、アプリケーション・プログラムやユーザーの相互作用へ影響を与えません。データ・モデルは、ファイルをベースとしたシステムで使用する物理的な特質の替わりに、データにアクセスするために例えば名前のような抽象的な特質を使えるようにします。データの追加によってデータ・アイテムのこれら抽象的な特性が変わることはないので、アクセス方法や技術も変更する必要がありません。既存のプログラムやユーザーの問合せは影響なく実行できます。新しいデータを使用する場合のみ、修正の必要があります。それゆえ、DBMSは*論理的にデータの独立性*があります。

## 2.5 高水準言語サポート

通常ほとんどのデータベースに、数種類の高水準問合せ言語を扱う機能が備わっています。この高水準言語によって、ユーザーはデータベースの物理的な格納構造に言及することなく、データのアクセス定義とデータ操作を行うことができます。

高水準問合せ言語サポートは、アプリケーション・プログラム・インターフェース（API）を使ったプログラムを書くことで、データベースのデータにアクセスする要求を省略することができます。低水準アクセス方法は、日常の繰り返し業務を自動化するユーザー・アプリケーションを開発する場合には非常に役立ちます。しかし、1度きりでその場限りの問合せは容易ではありません。その場限りの問合せをしようとする度に、プログラムを用意しなければなりません。これには、多大な労力とユーザーの訓練が必要です。又このアプリケーション・プログラム用の業務は、飛躍的に増えます。

高水準言語を採用すると、その場限りの問合せの実行が比較的簡単な業務になります。データベースで使用できる高水準言語の多くは、覚えやすいように英語的な構文を使っています。高水準問合せ言語は、非常に強力でデータベースにおいて必要なあらゆる関数も実行することができます。DBMasterは、今日業界での事実上の標準である問合せ言語である*SQL (Structured Query Language)*を採用しています。

## 2.6 トランザクション管理

データベース管理システムは、大量の情報を保存し、ユーザーの同時アクセスが可能なように設計されています。また、同時にデータ操作が実行されているかもしれません。トランザクション管理の種類によっては、データを正しい順序でデータベースに書き込む必要があります。

### トランザクションとは

トランザクションとは本来、*作業の論理単位*と定義付けられているものです。データベースを整合性のある状態に維持するためには、データベース上での複数の操作は、同時に完了する必要があります。データベース上で

の1操作は、成功した場合はデータを変更する、或いは失敗した場合はデータを変更せずに残すというように、それ自体完結したトランザクションです。

複数の操作で、1つのトランザクションを形成します。データベースに2種類の情報、例えば顧客に発送した船荷の記録と現在の在庫商品の記録が保存されているとします。商品が顧客に発送された時、それは船荷リストに追加されます。これは、データベースの1操作です。同時に、発送された商品の量も現在の在庫商品から差し引く必要があります。

もし、これら両方が同時に完了しなければ、データベースは矛盾した状態になります。商品が発送されたにも関わらず、在庫商品から差し引かれない場合は、在庫商品の量が多すぎます。または在庫商品から差し引いたのにも関わらず、商品が発送されなかった場合は、在庫商品が少なすぎるということになります。これらの両方の操作が1つのトランザクションを形成し、同時に完了しなければなりません。それ以外の場合では2操作とも失敗になります。

トランザクションを完了してデータを変更した場合、そのトランザクションは、*コミットされた*と言います。トランザクションが失敗してデータが変更されなかった場合を、*ロールバック*と呼びます。

## 同時実行制御

同時にデータにアクセスしようユーザーが数人存在すると、生産性の低下につながることがあります。そのために、多くのデータベースで*同時実行アクセス*の機能をサポートしています。これは、複数のユーザーが同時にデータベースにアクセスできる機能です。

各ユーザーが別々のデータにアクセスする場合は問題ありませんが、ユーザーが同一データを操作しようとする場合は問題が起こります。2ユーザーのトランザクションで、何の取り決めも無く同じデータを扱った場合、どういう結果になるかは予想できません。トランザクションによっては、古いデータを読み込むかもしれませんし、外見上は完了したはずの修正が取り消されてしまうかもしれません。

このような問題が起こらないために、トランザクションにはシリアル番号がふられています。同時に並行して実行された2つのトランザクション

は、実際には1つを実行してからもう一方を実行した場合と同じ結果になります。トランザクションによっては、他のトランザクションがデータ・アイテムを使い終わるのを待機します。このような場合にその2番目のトランザクションが調整無く処理された場合、その結果はやはり予想不可能です。

あるトランザクションでデータを修正した後、さらに別の作業を実行し続けたとします。別の作業を実行している間に、2つ目のトランザクションがその同じデータを修正し、続行します。両方のトランザクションがコミットされる前に、1番目のトランザクションがエラーになってロールバックすると、DBMSはトランザクションを実行する前の状態にデータベースを戻し、データをもとの値にします。2番目のトランザクションもコミットされません。データに与えた値はキャンセルされます。

トランザクションにシリアル番号を付けて、データベースへの任意のアクセスを防ぐために、同時実行制御が必要になります。DBMSで頻繁に利用される同時実行制御は、*ロック*です。

## ロックの概念

データにロックをかけると、トランザクションが排他的なアクセスであることが保証されます。データがロックされている間は、他のトランザクションは操作を実行できません。標準的なマルチユーザー型のDBMSにおいて、この方法はあまり現実的ではありません。

替わりに、以下のようなより複雑なモデルが使用されています。

- 共有と排他の2種類のロック。
- 行ロック、ページ・ロック、表ロックの異なるロック・レベル。

### ロックの種類

共有ロックは、複数のトランザクションが同時に1つのデータへアクセスすることができます。但し、そのトランザクションはデータを修正できないという制限が1つあります。これによって、複数のトランザクションが1つのデータの値を読み込んでも、その値を変更しないということになります。これは複数のアクセスが互いに干渉しないために、実現しています。

あるトランザクションがデータを修正する時に、他のトランザクションにもそのデータを参照させ修正させると、データベース内の矛盾をもたらします。あるトランザクションがデータを修正する時に、そのデータへの他のトランザクションのアクセスを拒否させる場合、*排他ロック*を使います。これにより、そのトランザクションはそのサイクルの期間中、データを一定の状態に保ったまま、他の操作を継続し続けることが可能です。

DBMSには、3種類のロック・レベルも備わっています。もっとも、この本来の意義は、同時実行制御以上にパフォーマンスのためにあります。リレーショナルDBMSでは、ロックを設定することができる最小の単位は、行ロックです。これらの行が集まってページを形成し、さらに集まって表を形成します。DBMSでは、ページや表を1つのアイテムとしてロックすることが可能で、同時にその中にあるデータもロックされます。

### ロック拡張

トランザクションが1ページのうちの多くの行にアクセスしなければならない場合、個々にロックするために必要な時間と、データの軌跡を保つために使用されるリソースは、かなりの量になります。ページ・ロックを使えば、回数と使用されるリソースが減りますが、他のトランザクションとの同時実行性は低下します。2番目のトランザクションが同じページにある行の1つをロックしようとしても、現状では不可能です。但し、これは通常そのパフォーマンス効果で相殺されると考えることができます。

トランザクションが1つの表にある多くのページにアクセスする際にも、同じような状況がおこります。ページ・ロックの代わりに表ロックを使うと、同時実行性を犠牲にする替わりに、回数と使用されるリソースが減少します。トランザクションの中で、特定のデータに使用するロック・レベルを手動で設定することができます。DBMSでは*自動ロック拡張*を使って、パフォーマンスが改善されると判断した時、許容可能な同時実行性のレベルを維持し続けながら、1つ上のレベルへロックを自動的に移行します。

## 2.7 整合性制御

データベースの中のデータは正確であると考えられています。これは、正確なデータをタイムリーに回収する能力を備えるべく、データベース・シ

ステムが発達した初期の理由です。この目的のために、DBMSには必ず何らかの整合性制御があります。整合性制御は、データの整合性と有効性を確かにします。

同じデータがデータベース内の2箇所に存在し、DBMSがその重複に気づかない場合、データの矛盾につながります。2つの内の1データのみが更新されるといったトランザクションが起こりえます。結果として、DBMSは間違った情報や矛盾した情報をユーザーに与え、あきらかに一貫性の無い状態になります。適切に設計されたデータベースでこのようなことが起こった場合、整合性制御を備えたDBMSはエラーを返します。

また、データベース内の重複がコントロールされている限り、それを維持することも可能です。この場合、一方を変更すると、もう一方も同様に更新されます。これは、一般的に*カスケード更新*と呼ばれています。更新作業の間、データベースを一時的に矛盾した状態にし、更新が終了するまでユーザーにデータを使用させないようにすることができます。

データが有効な値に保つことも、データベースの重要な役割です。1週間**40**時間の替わりに**400**時間働いたことを示す従業員の給料支払簿は、明らかに無効なデータを含んでいると言えます。この値が無効であるかどうかDBMSが判断することはできませんが、データベース管理者に*整合性の制約*を定義、実行させます。この整合性の制約は、トランザクションでデータの修正を試みるたびに、データが有効であるかをチェックします。

## 2.8 アクセス制御

集中したマルチユーザーのDBMSのためには、不正ユーザーのアクセスを防ぎ、正規ユーザーのみにアクセスを制限するためのセキュリティ制御が必要です。セキュリティ制御は、一般的にユーザー権限とトランザクション権限の２つのエリアに分けられます。

### ユーザー権限

ユーザー権限は、通常ユーザーがユーザー名とパスワードを入力してシステムへ進入することによって、不正使用からデータベースを保護します。通常パスワードはユーザーとDBMSのみが知っていて、DBMSによって保護されています。ユーザー名とパスワードのスキーマは、データベースの

セキュリティを保障するものではありません。パスワードには、見破られにくく、配偶者やペットの名前といった個人情報ではないものを選ぶことが大切です。パスワードは絶対にコンピューターの表面のような見つけやすい場所に残さないで下さい。

### トランザクション権限

一般的に、全ユーザーにデータベースに対して同レベルのアクセス権を与えるということはしません。従業員の給料といったデリケートなデータには、それが必要なユーザーのみアクセスさせます。また別の例では、あるユーザーはデータを参照した後、更に更新する必要があるかもしれませんが、他のユーザーは同じデータをただ参照したいだけかもしれません。

POS（販売時点情報管理）システムが良い例です。店で働いている店員は商品の価格を見るためにアクセスすることができますが、価格を変更することはできません。本社の従業員は、商品の新しい価格を入力するために、そのデータを閲覧して変更する必要があるかもしれません。

トランザクション権限は、故意または不意にデータに進入しようとするアクセス権のない正規ユーザーからデータベースを保護します。ユーザーが各データに対して有する権限の記録はDBMSに保管され、トランザクションでデータベースにアクセスしようとするたびに、その権限がチェックされます。ユーザーがデータへの適切な権限を持っていない場合、トランザクションは認められません。データベース管理者は、各ユーザーに対し明確に権限を与えなければなりません。

## 2.9 DBMSリカバリ

ソフトウェア或いはハードウェアの障害によって、DBMSが被害を受けることがあります。障害は*システム障害*と*メディア障害*の2種類に分類されます。障害が発生した後のリカバリ手法は、DBMSがファイル・ベースのシステムに勝る主な長所の一つです。

### システム障害

コンピューター・システムで*揮発性メモリ*がエラーになった時に、システム障害が起こります。揮発性メモリはコンピューター・システムにおいて

主要なメモリとして使用されています。システム障害は、パワー障害やプログラム/オペレーション・システムの障害、或いはその他の理由で起こります。システム障害を防ぐ一般的な方法は、トランザクション・ジャーナルやトランザクション・ログの利用です。

トランザクション・ジャーナルとは、データベースに加えられた変更の全ての履歴です。システム障害の際に進行中のトランザクションの正確な状態は、信頼性を持って判断することはできず、システムを再起動する時に完了することができません。DBMSは、トランザクション・ジャーナルを使って、ディスクに記録された不正常に終了したトランザクションの全変更を、元に戻します。

ディスクに全変更を記録せずに、システム障害の前にトランザクションを完了することが可能です。データは、エラーの際にDBMSシステム・バッファーに保存されている可能性があります。この場合、DBMSはトランザクション・ジャーナルを使って、完了してディスクに記録されていない全トランザクションをやり直すかロールオーバーします。

## メディア障害

メディア障害は、ディスク・ストレージ・システムで発生します。通常メディア障害は、ヘッドクラッシュ、火災、地震、振動、物理的な操作限界を超えた重力のようなディスク自身への物理的なトラウマによって引き起こされます。影響されるディスクのデータの損失を防ぐ手段はありません。但し、データベースに*アーカイブ*や*データ・ミラー*を備えている場合は、データベースをリストアすることができます。

アーカイブは、一定の期間毎、例えば毎晩行われるデータベースのバックアップです。これにより、前回のバックアップ以降に発生したファイル・トランザクション毎のバックアップ・コピーを保存します。メディア障害が発生した時、前回のバックアップ時にまでデータベースを復元するためにバックアップ・コピーを使います。前回のバックアップ以降の変更は全て消失します。このようなアーカイブは、データベース・システムによっては適合しますが、電子バンキングや航空券の予約システムのような重要なアプリケーションに対応するほど強固ではありません。

データ・ミラーは、データベース全体のアーカイブ・コピーを継続的に作成する働きをします。ある時点でのデータベース全体のコピーと複製のト

ランザクション・ジャーナルを作成します。データベースへの変更は、両方のログに同時に書き込まれ、事実上2つのデータベースを作成します。メディア障害がログの時間に発生した場合、2つ目のログに書き込まれなかった部分のみ、復活することができません。1つ目のログにまだレコードがあり、リカバリの際にエラー・メッセージでその損失について知らせるエラー・メッセージが戻ります。

両方の手法を使う場合、リストア時に確実にデータベースを復旧させるために、オリジナルのデータベースとは別の場所にバックアップ・コピーを保存して下さい。

# 3 DBMSアーキテクチャ

DBMSのアーキテクチャには2つの側面があります。論理DBMSアーキテクチャと物理DBMSアーキテクチャです。論理アーキテクチャは、データ保管と提供方法について取り扱い、一方物理アーキテクチャは、DBMSを形成するソフトウェアのコンポーネントについて触れます。

## 3.1 DBMSの論理アーキテクチャ

論理アーキテクチャは、ユーザーがデータベースのデータを認識する方法の概念です。これは、DBMSでデータがどのように扱われ、処理されるかということではなく、それがどのように見えるかということです。ファイル・システム上にデータを保管する方法は、ユーザーには透過的です。ユーザーは、データがどこにあるのか、又は実際にはどこに保管されているのかといったことを心配する事無く、そのデータを操作することができます。それゆえ、データベースに様々なレベルの抽象概念が存在することになります。

今日使用されている業務用データベース管理システムの多くは、*ANSI/SPARC*研究グループによって提案されたデータベース管理システム、ANSI/SPARCの一般化されたDBMSアーキテクチャに基づいています。ANSI/SPARCアーキテクチャは、そのシステムを3つのレベルの抽象概念に分けることができます。内部/物理的レベル、概念レベル、外部/ビューレベルです。

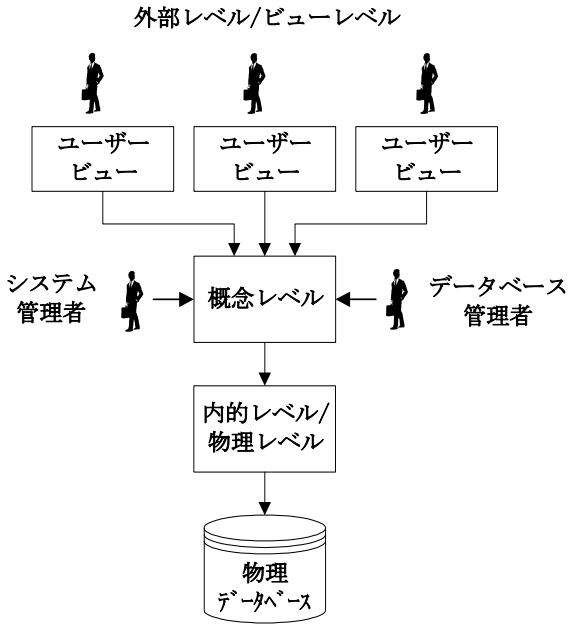



*図 3-1: 典型的なDBMSの論理アーキテクチャ*

## 内部/物理的レベル

2つ目のストレージ端末に永久に保管されるファイルの集まりが、物理レベルです。物理レベルまたは内部レベルは、物理ストレージに最も近くに存在します。これは、物理データベースの低レベルの概念を表し、オペレーティング・システムのファイル・システムと抽象概念の高位で使われる記録構造の間のインターフェースになります。このレベルでレコードの種類とストレージ方法が定義されます。同時に保管されたフィールドがどのように表されているか、保管されたレコードの物理シーケンスが何か、他の物理構造が存在するかも定義します。

## 概念レベル

概念レベルは、データベース全体の論理ビューを表します。データベースの全データは整合性のあるビューに統合されています。データベース設計

の最初のステップは、概念ビューを定義することです。つまり、DBMSのいデータ定義言語を利用します。

概念レベルは、データの独立性を実現します。概念レベルを作成するために使用するデータ定義言語で、物理レベルで扱われるいかなる物理ストレージ事項を定義してはなりません。ストレージやアクセス情報も定義せず、データの内容のみを定義します。

## 外部レベル/ビューレベル

外部レベル/ビューレベルは、概念ビューのウィンドウです。ユーザーは自身に関係のあるデータのみを見ることになります。この場合のユーザーは、アプリケーション・プログラム、或いはエンドユーザーです。複数の外部スキーマを定義し、相互にオーバーラップすることもできます。

システム/データベース管理者は、特殊なケースです。データベースの設計と保守を担っているために、データベース全体を見る必要があります。これら2種類のユーザーにとって、外部ビューと概念ビューは機能的に同等のものです。

## レベル間のマップ

データベースにある3つのレベルの抽象対象は、お互いに独立して存在していません。レベル間で対応しているか、マップする必要する必要があります。実際には、概念と内部マッピングと外部と概念マッピングの2種類のマッピングがあります。

概念と内部マッピングは、概念レベルと内部レベル間で存在し、概念ビューのレコードとフィールドの間と、内部ビューのファイル構造とデータ構造の間の関係を定義します。保存したデータベース構造が変わった場合、更に概念/内部マッピングも修正しなければなりません。このマッピングは、データベースにおける物理的なデータの独立性を実現します。

外部と概念マッピングは、外部レベルと概念レベル間に存在し、とりわけ外部ビューと概念ビューの間の関係を定義します。これら2種類のレベルは似通っていますが、とりわけ外部ビューに見られるいくつかの要素は、概念ビューと異なります。例えば、いくつかのフィールドを元のフィールドと異なる名前を持つ1つの(仮想)フィールドに統合することができます。

概念レベルのデータベース構造が変更した場合は、外部レベルからのビューが整合性を保つように、外部/概念マッピングも変更する必要があります。データベースの論理データを独立させるのがこのマッピングです。

その他のマッピングも可能です。ある外部ビューを別の形態の外部ビューで表現するような場合、これを外部と外部マッピングと呼ぶこともできます。複数の外部ビューがもう1つの別の外部ビューに密接に関係している場合にこれは役に立ちます。これにより、ユーザーは似通った外部ビューをそれぞれ直接概念レベルにマッピングする必要がなくなります。

## 3.2 DBMSの物理アーキテクチャ

物理的なアーキテクチャは、データ入力と処理に使用するソフトウェア・コンポーネントと、これらのソフトウェア・コンポーネントがどのように相互接続しているのかといった概念です。最も基礎のレベルでは、DBMSの物理アーキテクチャは、バックエンドとフロントエンドの2つの部分に分けることができます。

バックエンドは、物理的なデータベースを管理して、内部レベル、概念レベル、そして外部レベルで必要なサポートとマッピングを行います。セキュリティ、整合性、アクセス制御のようなDBMSの他の機能も担っています。

フロントエンドは、DBMSで作動しているアプリケーションから成ります。それらのアプリケーションは、DBMSベンダー、ユーザー、又はサード・パーティによって提供されます。ユーザーは、フロントエンドとのみ接し、バックエンドの存在すら気が付かないかもしれません。

バックエンドとフロントエンドは、ほとんどのDBMSで共通のソフトウェア・コンポーネントにさらに分けることができます。

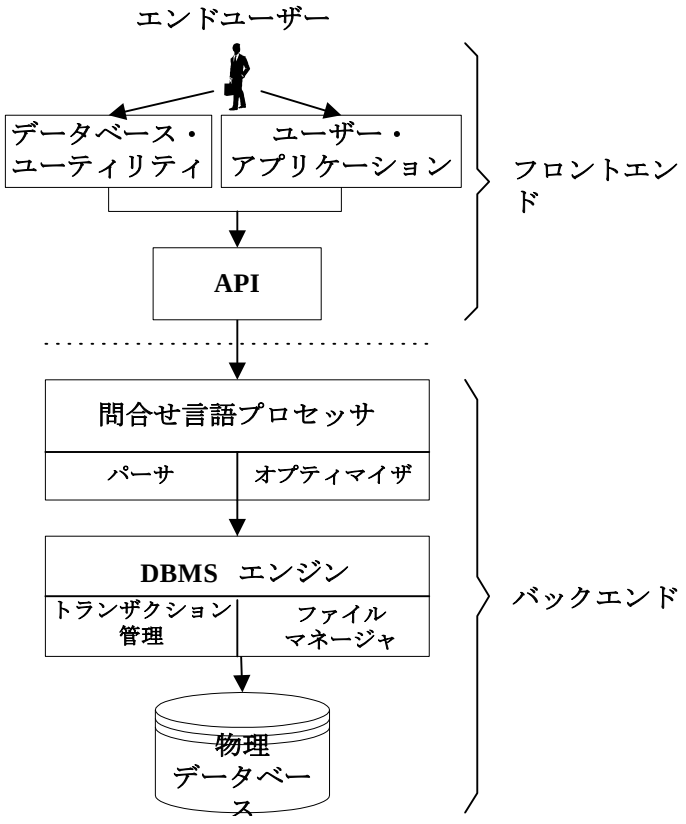



*図3-2 DBMSの一般機能とコンポーネント*

## アプリケーションとユーティリティ

アプリケーションとユーティリティは、ほとんどのユーザーにとってDBMSへのメインのインターフェースです。DBMS用のアプリケーションとユーティリティの主な3種類のソースは、ベンダー、ユーザー、サード・パーティです。

## ベンダー・アプリケーション

ベンダー・アプリケーションとユーティリティは、データベースを使用したり保守したりするためのものです。ユーザーがカスタム・アプリケーションを書かなくても、データベースの作成と操作が可能になります。通常これらは一般使用目的のアプリケーションで、特定の業務をこなすために最適なツールとはいえません。

## ユーザー・アプリケーション

ユーザー・アプリケーションは、従来のプログラミングム言語を使って書かれたカスタム仕様のアプリケーション・プログラムです。このプログラミング言語は、API(アプリケーション・プログラム・インターフェース)を経由して、DBMSの問合せ言語に連結されます。これにより、カスタム・アプリケーションを柔軟に利用して、DBMSの問合せ言語の能力を最大限に活用することができます。

## サード・パーティのアプリケーション

サード・パーティのアプリケーションは、強化されたベンダー・アプリケーションのようなものです。或いは、ベンダー・アプリケーションに含まれない、ユーザーのニーズをも満たしているかもしれません。また、大多数のユーザーが必要と考える特定用途のための書かれたユーザー・アプリケーションとも考えることができます。

## アプリケーションとユーティリティのリスト

データベースで一般的に使用されるアプリケーションとユーティリティのほとんどは、明確に定義されたカテゴリに分類することができます。

### コマンドライン・インターフェース

文字ベースで、DBMSの問合せ言語の能力と機能性を直接使う会話型インターフェースです。その場限りの問合せを実行するといったデータベース操作や、結果を直ちに見ることいったことが可能です。これは従来のプログラミング言語でプログラムを記述せずに、データベースの能力を最大限活用することができる唯一の手段であることが多いです。

## グラフィカル・ユーザー・インターフェース(GUI)ツール

これは、DBMSと問合せ言語の複雑さを、直感的で理解しやすく便利なインターフェースに隠した、グラフィカルで会話型のインターフェースです。一般ユーザーでも問合せ言語を学習する事無く、データベースにアクセスできるようになります。上級ユーザーにはとっては、形式的な命令文を入力する煩わしさが無く、データベースの管理/操作が手早く行えるようになります。グラフィカル・インターフェースは、コマンドやオプションの全てを実行することができないので、通常コマンドライン・インターフェースほどの機能性は持ち合わせていません。

## バックアップ/リストア・ユーティリティ

これは、データベースの障害による影響を最小限にし、障害が発生した時点の整合した状態にデータベースをリストアするために考案されました。自動ユーティリティがユーザーの介入無しに定期的にバックアップを行うのに対し、手動のバックアップ/リストア・ユーティリティは、ユーザーにバックアップの開始を促します。バックアップ/リストア・ユーティリティを正しく使うと、システム障害から適切に正確にDBMSをリカバリします。

## ロード/アンロード・ユーティリティ

これは、ユーザーがデータベース全体やその一部をアンロードして、同じコンピューターや遠隔地にある別のコンピューターにデータを再ロードします。これは、様々な状況で役に立ちます。例えば、ある時点のデータベースのバックアップ・コピーを作成したり、データを新バージョンのデータベースや全く異なるデータベースにロードしたりすることができます。これらのロード/アンロード・ユーティリティは、パフォーマンスを向上させるためのデータの再編集にも使用することができます。例えば、特殊な方法でデータのクラスタや、古くなったデータで占有されたスペースの再利用です。

### レポート/分析ユーティリティ

これは、データベースに含まれるデータを分析、報告するために使用します。データ傾向の分析、データ値の解析、或いは特別な条件に当てはまるデータの表示、この情報を含むレポートの表示と出力を行います。

## アプリケーション・プログラム・インターフェース(API)

アプリケーション・プログラム・インターフェース(API)は、データベース・エンジンで直接操作する低レベル・ルーチンのライブラリです。APIは、通常ソフトウェア・アプリケーションをC++やVisual Basicのような一般目的のプログラミング言語を記述する時に使用されます。これを使用することにより、ユーザーはストレージ・アーキテクチャを開発せずに、ビジネス需要に見合うカスタム・アプリケーションを書くことができます。データベース・エンジンは、データのストレージを扱います。入力や特殊な分析/レポート機能は、カスタム・アプリケーションによって操作されます。

APIはDBMS特有のものなので、あるDBMSのAPIを使って書かれているプログラムは、他のDBMSでは使用できません。どのAPIにも通常、データベース操作と密接に関係する独自の関数呼び出しがあります。2つのデータベースが同じ機能を持っている場合でも、データベースの設計いかんでは、異なるパラメータと関数を使用しているかもしれません。唯一の例外は、Microsoft社のODBC (Open Database Connectivity)のAPIです。これをサポートするDBMSであれば、同様にサポートしている他のDBMSでも使用することができます。

## 問合せ言語プロセッサ

問合せ言語プロセッサは、問合せ言語文の受け取りと、問合せ言語の英語的な構文をDBMSが許容しうる形式に変換する役目を果たしています。通常、問合せ言語プロセッサは、パーサと問合せオプティマイザの2つの部分で構成されています。

### パーサ

パーサは、アプリケーション・プログラムやコマンドライン・ユーティリティから問合せ文を受け取り、その構文を調べて正しいかどうかを確認します。パーサは、問合せ文を構文の基本単位に分解し、それらが適切なコンポーネントで構成されているかどうかを調べます。その文が構文ルールに従っている場合は、そのトークンは問合せオプティマイザに渡されます。

### 問合せオプティマイザ

問合せオプティマイザは、問合せ文を調べて、最も効率的な実行方法を見つけようとします。オプティマイザは、操作を実行するための問合せ計画を様々な順序でいくつか生成し、最も効率よく実行できる計画を調べます。その際、問合せオプティマイザは、CPU時間、ディスク時間、ネットワーク時間、ソート方法、スキャン方法等を評価します。

## DBMSエンジン

DBMSエンジンは、DBMSの心臓部でデータ管理の全てを担っています。通常トランザクション管理とファイル管理の2つの部分に分けられます。

### トランザクション管理

トランザクション管理は、権限表と同時実行制御の表を保守します。トランザクション管理の際に、ユーザーがデータベースで問合せ言語文を実行する権限を有するかどうかを確認するために権限表が使用されます。権限表でチェックされた認証を与えられたユーザーのみ、権限表を修正することができます。同時実行制御表は、同時にコンフリクトする命令文が実行された際に、コンフリクトを防ぎます。問合せ言語文を実行する前に、他の文でその対象がロックされていないかどうかを確認するために、同時実行制御表がチェックされます。

### ファイル管理

ファイル管理は、データベース上の全ての物理的な入力/出力操作を受け持つコンポーネントです。ディスクにあるデータの物理アドレスに関与し、

ホスト・オペレーティング・システムとの相互作用、読み込み、書き込みを担っています。

# 4　データベース

DBMasterを使えば、簡単にデータベースを作成、管理することができます。拡張性のあるクロスプラットフォーム・サポートとユニークなオープン・アーキテクチャで、複数のプラットフォームに及ぶデータベース・アプリケーションを展開させ、システムの成長にともない更に大規模なシステムへ移植することができます。ノートブックPC使用の小規模なシングルユーザーのデータベースから、世界中に分散配置された大規模なマルチユーザー・データベースまで完全にかつ容易に拡大することができます。

本書は、データベース管理の関数を実行するコマンドライン・ユーティリティのdmSQLを使って、わかりやすく解説することを目的としています。DBMasterには、データベース管理を簡略化したグラフィカル・ユーティリティの数種類のJ Toolもあります。

本章では、以下の項目について解説します。

- 有効なデータベース名の選択方法。
- データの分割方法。
- dmSQLコマンドライン・ツールの起動方法。
- データベースの作成方法。
- データベースの環境設定の方法。
- データベースの起動と終了の方法。
- データベースとの接続と切断の方法。

## 4.1 ネーミングの規則

DBMasterでは、一台のコンピューターで同時に複数のデータベースを作動させることができます。接続するデータベースを見極めるために、そのデータベースを他のデータベースと識別することが必要です。DBMasterでは、このためにネーミング・スキーマを用います。

DBMasterは、dmconfig.iniファイルにローカルとリモート双方のデータベースの全環境設定情報を保管します。２つの異なるデータベースに同じ名前を使用すると、データベースはコンフリクトするかもしれません。どの環境設定のセクションがどのデータベースのものかわからないために、間違った場所に環境設定の情報を書き込まれるかもしれません。dmconfig.iniファイルのセクション見出しで、データベース名が既に使われているかどうかを確認し、他には無いデータベース名を付けます。

CREATE DBコマンドを実行する前に、32文字以下の名前を慎重に選んで下さい。データベース名には、文字、数字、アンダーバーを使うこともできます。

➲ **例:**

```
Tutorial
Parts_db
Region_1
1_Region
```

データベース名は、大文字と小文字を識別しません。つまり、データベース名を**Tutorial**として作成しても、ログイン時には**tutorial**や**TUTORIAL**と入力することができることを意味します。**Tutorial**は、本書の中で使用するデータベース名です。

## 4.2 dmconfig.iniファイル

dmconfig.iniと呼ばれるファイルには、データベース名を含むデータベースの環境設定の全情報が保管されています。このファイルには、各データベースの環境設定セクションがあります。dmconfig.iniファイルは標準ASCIIテキスト・ファイルですので、どのようなテキスト・エディタでも編集することができます。

DBMasterには、dmconfig.iniファイルの保守を簡単に行うことができるGUIツールのJConfiguration Toolもあります。このインターフェースを使えば、DBMasterの環境設定のパラメータに慣れるまでに要する時間を短縮し、パラメータを明確に定義されたカテゴリに配置することができます。

ほとんどの場合、データベース起動時に環境設定情報が参照されます。データベース起動後に情報が変更された場合、その変更された情報は次回のデータベース起動時まで適用されません。但し、環境設定によっては、データベースに接続する際にのみ使用されますので、データベースに接続する前であれば、随時この情報を変更することができます。新しい値は、次に接続する際に適用されます。

環境設定のパラメータは、DBMasterのパフォーマンスの重要な役割を果たします。各環境設定の役割を知り、DBMasterをスムーズに運用できる最適値を見つける必要があります。環境設定のパラメータとそれらを制御するためのキーワードについての完全な解説は、*「データベース管理者参照編」*をご覧下さい。

## 作成

dmconfig.iniファイルは、インストール・プログラムで自動的に作成されるので、通常ユーザーが新たにそのファイルを作る必要はありません。データベースを作成する際に、dmconfig.iniファイルに新規データベース名と同じ名前の環境設定のセクション名が存在する場合、作成時オプション（データベース作成時に使用される環境設定オプション）が新規データベースに適用されます。

データベースを作成する前に、テキスト・エディタでデータベースの環境設定セクションを作る必要があります。作成時オプションに初期設定値以外の値を利用している場合は、データベース作成時にそのパラメータが適用されます。データベース作成の際にdmconfig.iniに環境設定セクションが無い場合、最初に見つけたdmconfig.iniファイルに自動的にセクションを作成します。dmconfig.iniファイルが見つからない場合は、新しいdmconfig.iniファイルにそれを作成します。新しい環境設定セクションには、全作成時オプションの初期設定値が採用されます。一般的に、作成時オプションの初期設定値は、ほとんどのデータベースで問題無いはずです。

## ディレクトリ

Windowsシステムの場合、dmconfig.iniファイルはWINDOWSディレクトリにあります。Windows 98、又はWindows NT 4.0を使用している場合、ユーザーはスタート・ボタンからdmconfig.iniファイルを開くこともできます。スタート・ボタンをクリックして、プログラムのDBMasterを選択し、DBMaster環境設定ファイル(dmconfig.ini)を開いて下さい。

Unixシステムの場合、dmconfig.iniファイルの場所は3個所考えられます。データベースを作成する時、DBMasterはdmconfig.iniファイルにデータベースに対応するセクション名を配置するために、以下の順にこれらの3つの場所をスキャンします。

1. 現在のディレクトリ
2. DBMASTERの環境変数で指定したディレクトリ
3. ~DBMaster/データ・ディレクトリ

dmconfig.iniファイルとセクション名が見つかった場合、そのセクションのキーワードが使われます。ファイルにセクション名が無かった場合、次のディレクトリでdmconfig.iniファイルをチェックします。

## フォーマット

dmconfig.iniファイルは、データベース環境設定セクションと呼ばれるセクションに分かれています。各データベースにデータベース環境設定セクションがあり、そのセクションの値がデータベースの環境設定の操作を制御します。

各データベースの環境設定セクションは、セクション見出しとそれに続くキーワードで構成されています。セクション見出しは、[]かっこで括られたデータベース名です。キーワード行は、キーワードと対応する値から成ります。

➲ **例:**

```
[セクション見出し1]
キーワード1 = 値1                ;セミコロンの後ろの文字はコメントです
キーワード2 = 値2
          .
```

```
          .
[セクション見出し2]
キーワード3 = 値3 値4              ;値の間にデリミタとして、
キーワード4 = 値5                  ;スペースやカンマを使うことができます
          .
          .
```

dmconfig.iniファイルのキーワードは、大文字と小文字を識別しません。値が大文字と小文字を識別するかどうかは、キーワード、又はデータベースが作動しているオペレーティング・システムによって決定します。

## セクション名

各セクション名はデータベース名に対応し、データベース起動時にそのセクションの環境設定オプションが参照されます。セクション名は、データベース名をかっこ([ ])で括ったものです。左側の括弧([]は、必ず行の1文字目に配置します。

## キーワード

各セクション名に続いて、キーワードと値の一覧があります。データベースの起動時に、これらの値がそのセクション見出しに対応するデータベースによって使用されます。その記述「キーワード=値」は、キーワードに値を割り当てます。キーワードに複数の値を与える場合、スペースかカンマで各値を区切ります。この値は、整数か文字列のいずれかです。

dmconfig.iniにキーワードが無い場合は、初期設定値が使用されます。キーワードは、その用途によってはデータベース起動時や接続時に使用されることがあります。キーワードとその値の全リストは、*「データベース管理者参照編」*の付録Bをご覧下さい。

## コメント

セミコロン(;)の後に書かれている文字列や記号は、コメントとみなされ無視されます。キーワードの目的や、そのキーワードにその値を指定した理由や、値を一時的に変更した場合の元の値など、ユーザーが思い出すためにコメントを利用することができます。

# 4.3 dmSQL

dmSQLは、文字ベースの会話型のユーザー・インターフェースです。DBMasterのSQL問合せ言語の能力と機能性を最大限に活用します。dmSQLを使ってデータベースを操作し、その場限りのSQL問合せを実行して、直ちに結果セットを見ることができます。dmSQLは、形式的なプログラミング言語を使ったプログラムを作成する事無く、データベースの能力をフルに活用する唯一の方法であることが多いです。

## 起動

この入門書で紹介するほとんどの例でdmSQLを使用しますので、このアプリケーションの起動方法を知り、それに慣れる必要があります。クライアント／サーバー操作の新しいデータベースの環境設定の方法については、本章の後の部分で説明します。

### Windows

Windowsのプラットフォームでは、dmsqlsとdmsqlcの2つの機能性がdmsql32.exeプログラムに統合されました。

➲ **Windows98かWinNTでdmSQLコマンドライン・ユーティリティを起動する:**

1. ［**スタート**］ボタンをクリックします。
2. ［**プログラム**］をクリックしてから、［**DBMaster**］を選択し、［**dmSQL**］をクリックして下さい。
3. **dmSQL**アプリケーションが起動します。

### Unix

DBMasterのUNIXバージョンには、dmSQLアプリケーションのシングルユーザー・バージョン(dmsqls)とクライアント/サーバー・バージョン(dmsqlc)があります。UNIXでシングルユーザー、又はクライアント/サーバー・データベースを作成する場合は、dmsqlsを使用して下さい。

➲ **UNIXでdmSQLコマンドライン・ユーティリティを起動する:**

1. コマンドラインに、次のように入力して下さい。
```
cd ~DBMaster/current version/bin
```
*注: <current version>をDBMasterの現バージョンに置き換えて下さい。例、4.0。*

2. ENTERを押して下さい。

3. コマンドラインに、次のコマンドを入力して下さい。
```
dmsqls
```

4. ENTERを押して下さい。

5. **dmSQL**アプリケーションが起動します。

## 作業スペース

dmSQLを起動すると、WindowsシステムにはdmSQLの作業スペースが、UNIXシステムにはdmSQL>コマンドライン・プロンプトが表示されます。

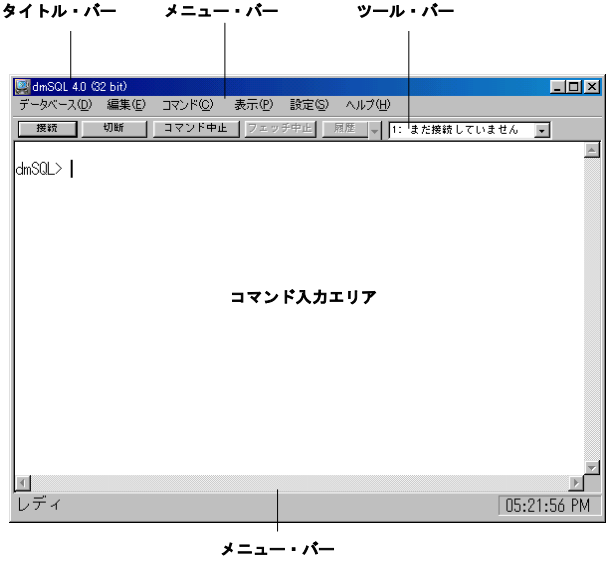



*図4-1: dmSQLのWindowsバージョン*

dmSQLウィンドウは、以下の要素で構成されています。

- **タイトル・バー**—タイトル・バーには、プログラム名(“dmSQL”)と最小化、最大化、閉じるの各ボタンがあります。
- **メニュー・バー**—メニュー・バーには、dmSQLプルダウン・メニューの項目があります。各メニューには、関連するコマンド・リストがあります。
- **ツール・バー**—ツール・バーには、よく使用する関数のコマンド・ボタンのパレットとドロップダウン・リストボックスがあります。
- **コマンド入力エリア**—コマンド入力域は、dmSQL作業スペースのメイン・ウィンドウです。コマンドを入力し、dmSQLがスクリプトを実行し、テキストを表示します。
- **ステータス・バー**—ステータス・バーは、作業スペースの現在の状態と現在の時刻を表示します。

## 4.4 Jツール

DBMakeには、データベースを管理するための、3つのJavaベースのクロスプラットフォーム・ユーティリティがあります。これらのツールは、使いやすく、直感的なインターフェースでデータベースに直ぐに慣れることができるよう設計されています。各Jツールに、マニュアルとオンラインヘルプが用意されています。ODBCをサポートしているオペレーティング・システムであれば、Jツールを使用することができます。以下の節では、これらのツールとその機能について説明します。但し、本書の対象外の内容については、各ツールのマニュアルを参照して下さい。

### JConfiguration Tool

4.2節の「dmconfig.iniファイル」で説明したように、JConfiguration toolは、データベースの環境設定パラメータを管理するために使用します。
JConfiguration toolは、dmconfig.iniファイルのキーワードの意味やその有効値を覚える事なく、データベースの環境設定をすることができる記述型インターフェースです。Windowsのオペレーティング・システムでは、スタート>プログラム>DBMaster 4.0>JConfiguration Toolの順にクリックする

と、JConfiguration Toolを起動することができます。JConfiguration Toolの使い方の詳細については、*「JConfiguration Tool ユーザーガイド」*、又はツールのオンラインヘルプを参照して下さい。

### JServer Manager

JServer Managerは、データベースの作成、起動、終了、削除、バックアップ、リストアといった、最も一般的なデータベース管理ルーチンを実行するための簡単なグラフィカル・インターフェースです。JServer Managerの利用は本書の範囲外ですが、JServer Managerを使ったデータベースのバックアップ方法とリストア方法の例は、9章の「データベース・リカバリ」で説明します。これらのルーチンの単純化に役立てるためにJServer Managerの利用を強くお勧めします。Windowsのオペレーティング・システムでは、スタート>プログラム>DBMaster 4.0>JServer Managerの順にクリックすると、JServer Managerを起動することができます。JServer Managerの使い方の詳細については、*「JServer Manager ユーザーガイド」*、又はツールのオンラインヘルプを参照して下さい。

### JDBA Tool

JDBA Toolは、各データベースの論理組織を明快かつ機能的に見ることができるようにするツールです。データベース管理者は、スキーマ・オブジェクトを作成、削除、修正するためにそれを使うことができます。Windows オペレーティング・システムでは、スタート>プログラム >DBMaster 4.0> JDBA Toolの順にクリックすると、JDBA Toolを起動することができます。JDBA Toolの使い方の詳細については、*「JDBA Tool ユーザーガイド」*、又はツールのオンラインヘルプを参照して下さい。

## 4.5 データベースの作成

データベースの作成は、DBMasterを使ったデータベースの設計と実行の中でおそらく最も簡単な作業です。練習データベースを作成する場合は、データベース名となる**“Tutorial”**を付けて、**CREATE DATABASE**文を入力するだけです。

## 練習データベース

➲ **dmSQLを使って、練習データベースを作成する:**

1. dmSQLコマンド・プロンプトに次の文を入力します。
```
dmSQL> CREATE DATABASE Tutorial;
```
2. ENTERを押します。
3. 以下のラインが画面に表示されます。
```
USE db #1 connected to db:<Tutorial> by user:<SYSADM>
```

この文では、**Tutorial**という名前の空のデータベースを作成します。データベース作成前に、そのセクション名が既に存在するかどうかを確認するために、dmconfig.iniファイルが参照されます。既に存在する場合、DBMasterはそのセクションの作成時オプションのキーワード値を使って、新規データベースを作成します。

**Tutorial**の例では、データベース作成前にデータベースの環境設定セクションを作成しませんでしたので、DBMasterが環境設定セクションを1つ作成し、作成時オプションには全てその初期設定値を使います。

## 接続ハンドル

dmSQLを使って、1つのデータベースで同時に8接続まで行うことができます。DBMasterは、どのデータベース接続が現在アクティブであるかを表示するためにterm USEを使います。

このUSEは、接続ハンドルとしても知られています。USE#1からUSE#8まで8つの接続ハンドルがあります。最初に設けるデータベース接続は、USE#1になり、2番目にUSE#2、そしてUSE#8まで同様です。dmSQLで現在接続しているデータベースを見る場合は、dmSQLコマンドラインにUSEコマンドを入力して下さい。

### USEコマンド

このコマンドを実行すると、dmSQLは現在接続している全データベースの一覧を表示します。データベースが1つだけの場合、接続ハンドル(USE#1)は1つだけです。これは、**Tutorial**という名前で作成したデータベースが1

つのハンドル**USE#1**を持ち、ユーザーが**SYSADM**というユーザー名で接続したこと意味します。

➲ **dmSQLを使って、現在接続している全データベースを見る:**

1. dmSQLコマンド・プロンプトに**USE**と入力します。
```
dmSQL> USE;
```
2. ENTERを押します。
3. 以下のように表示されます。
```
dmSQL> USE;
USE db #1 connected to db:<TUTORIAL> by user:<SYSADM>(CURRENT)
```
4. 以下は、複数のデータベース接続を表しています。
```
dmSQL> use;
USE db #1 connected to db:<TUTORIAL> by user:<SYSADM>(CURRENT)
USE db #2 connected to db:<DBSAMPLE> by user:<SYSADM>
USE db #3 connected to db:<EXDM35> by user:<SYSADM>
```

この例では、接続情報の後ろに(CURRENT)という文字が表示されていることで、現在接続しているデータベースが**Tutorial**であることがわかります。

USE#1以外の接続ハンドルでデータベースに接続する場合、接続する前に接続するUSE#に変更して下さい。dmSQLで他の接続ハンドルに変更する場合は、USEコマンドの後ろに接続ハンドル番号を付けて、dmSQLコマンド・プロンプトに入力して下さい。

➲ **dmSQLを使って接続ハンドル番号2に変更する:**

1. dmSQLコマンド・プロンプトに**USE 2**と入力します。
```
dmSQL> USE 2;
```
2. ENTERを押します。

## 初期設定ユーザー

データベース作成時に、ユーザーは自動的にSYSADMというユーザー名で接続します。SYSADMは、データベース内で最大の権限を持つユーザーです。新規ユーザーのアカウントを作成することができ、データベースの全オブジェクトへのあらゆる権利と権限を持っています。但し、データベー

スを作成したばかりの時は、データベースにはデータベース・オブジェクトは全くありません。

新しいデータベースを作成すると、自動的にシングルユーザー・モードで起動します。シングルユーザー・モードは、一度に1人のユーザーしか接続することができません。この場合、SYSADMのパスワードを初期設定値(パスワード無し)から変更することができます。SYSADMのパスワードを変更しないと、だれでもSYSADMのアカウントを使ってデータベースに接続し、それを完全に制御することができます。SYSADMパスワードの変更についての詳細は、後ろの節で説明します。

他のユーザーにもデータベースに接続できるようにする場合、マルチユーザー・モードのいずれかでデータベースを実行します。これは、UNIXのシングルユーザー・モードであり、Windowsのマルチユーザー・モードであり、WindowsとUNIXのクライアント/サーバー・モードのことです。マルチユーザー・モードでデータベースを実行する場合は、データベースを一旦終了してから再起動します。クライアント/サーバー・モードで、実行する場合は、データベースを終了し、dmconfig.iniファイルにいくつかのキーワードを追加します。

➲ **dmSQLを使って、Tutorialデータベースを終了する:**

1. dmSQLコマンド・プロンプトに次のコマンドを入力します。
```
dmSQL> TERMINATE DATABASE;
```
2. ENTERを押します。

## 4.6 データベース・モード

DBMasterでは、データベースを様々なモードで起動することができます。各モードは、データベースへの接続やアクセス用にいくつかのオプションを提供し、1台のコンピューターを使った単純なシングルユーザー・システムから数台のコンピューターに分散された大規模マルチユーザー・システムへとデータベースをスケールアップする機能を備えています。

利用できるデータベース・モードは、データベース・サーバーが作動しているプラットフォームと接続する方法に関係します。DBMasterには、シン

グルユーザー、マルチユーザー、クライアント/サーバーの3つのデータベース・モードがあります。

## シングルユーザー・モード

シングルユーザー・モードは、UNIX/Linuxプラットフォームでのみ使用することができます。これは、DBMasterの非共有データベース用バージョンを単純化したものです。このモードの主な利点は、アプリケーションのサイズが小さくなり、データベース操作のほとんどの実行速度が速くなることです。つまり、ロックやセキュリティ、ネットワーク・サポートが、必要ありません。このモードの制限は、データベースに一度に1つの接続しかできないことです。バックアップ・サーバー、レプリケーション・サーバー、グローバル・トランザクション・サーバーといった他のサーバーやデーモンを起動することができません。このデータベースは、ネットワーク上で利用することができないので、ユーザーはホスト機からデータベースにアクセスする必要があります。UNIXでシングルユーザー・モードを使う際には、セキュリティが無いことに留意する以外に特別な環境設定は必要ありません。

## マルチユーザー・モード

マルチユーザー・モードは、Windowsプラットフォームでのみ使用することができます。このモードの1つの利点は、DBMasterのセキュリティと信頼性の機能を完全に利用しつつ、データベースに複数接続が可能なことです。シングルユーザー・モード同様、ネットワーク・サポートが無いので、データベースへの全接続は必ずホスト機からのアクセスになります。このモードの制限は、バックアップ・サーバー、レプリケーション・サーバー、グローバル・トランザクション・サーバーといった他のサーバーやデーモンを起動することができないことです。Windowsでマルチユーザー・モードを使う際には、ネットワーク・サポートが無いことに留意する以外に特別な環境設定は必要ありません。

## クライアント/サーバー・モード

クライアント/サーバー・モードは、全てのプラットフォームで使用することができます。このモードは、接続しているホスト・コンピューターか

ら、TCP/IPネットワークを経由して、データベースへと複数の接続を設けることが可能です。また、DBMasterのセキュリティと信頼性の全ての機能を利用することができます。更に、セキュリティを強化するために、ネットワーク上で送信されるデータを暗号化することも可能です。このモードは、バックアップ・サーバー、レプリケーション・サーバー、グローバル・トランザクション・サーバーといった他のサーバーやデーモンを使用することができます。クライアント/サーバー・モードを適切に起動させるためには、いくつかの環境設定が必要です。データベースを実行するためには、TCP/IPネットワークに接続し、データベースに接続するために使用する全コンピューターにTCP/IPネットワーク・プロトコルがインストールされていなければなりません。

## dmconfig.iniを変更する

dmconfig.iniファイルを変更後、DBMaster Serverを使ってクライアント/サーバー・モードのデータベースを再起動し、dmSQLのようなクライアント・アプリケーションで接続します。

➲ **dmconfig.iniの[Tutorial]セクションに、クライアント/サーバー・モード用に以下の行を追加する:**

```
[TUTORIAL]
DB_DBDIR=C:\DBMASTER\TUTORIAL\DATABASE
DB_USRID=SYSADM
DB_SVADR=127.0.0.1
DB_PTNUM=54321
```

### DB_SvAdr

**DB_SvAdr**キーワードは、クライアント/サーバー・モードでデータベースを起動する場合、クライアントとサーバー双方で必要です。このキーワードは、データベースのサーバーとなるコンピューターのIPアドレスを指定します。上記の番号をご使用のコンピューターのIPアドレスに置き換える必要があります。注、オペレーティング・システムがDNS(Domain Name System)を使用している場合、IPアドレスの替わりにDBMasterをインストールしたサーバーのDNS名を使うことができます。

## DB_PtNum

**DB_PtNum**キーワードは、クライアント/サーバー・モードでデータベースを起動する場合、クライアントとサーバー双方で必要です。このキーワードは、サーバーが接続要求を受け付けるポート番号です。

### クライアント/サーバーの起動

DBMaster Serverを使って、クライアント/サーバー・データベースを起動します。DBMaster Serverはデータベースを起動させると、dmSQLのようなデータベース・クライアントが接続するまで待機します。クライアントが接続すると、コマンドを受け付け、その結果を戻します。DBAかSYSADMのみデータベースを起動することができます。DBMaster Serverを立ち上げる際には、クライアント/サーバーのデータベース名を与えます。

データベースへの接続には、CONNECTコマンドを使います。このコマンドは、シングルユーザー、マルチユーザー、クライアント/サーバー・データベースのいずれにも適用されますが、クライアント/サーバーを使用する際には先にデータベースを起動することを忘れないで下さい。CONNECTコマンドには、データベース名、ユーザー名、パスワードの3つのパラメータがあります。とりあえず、パスワードの無いSYSADMのユーザー名を使います。

Windowsでデータベースから切断する場合は、DISCONNECTコマンドを使います。UNIXの場合はTERMINATE DBコマンドを使います。このコマンドは、シングルユーザー、マルチユーザー、クライアント/サーバー・データベースに適用されます。このコマンドにはパラメータはありません。アクティブ接続ハンドルでデータベースを切断するだけです。

### Windows

➲ **Windowsでクライアント/サーバー・データベースを起動する:**

1. ［スタート］ボタンをクリックします。
2. ［プログラム］をクリック、［**DBMaster**］を選択し、［**DBMaster Server**］をクリックします。

3. **DBMaster Server**アプリケーションが起動し、［**データベースの起動**］ダイアログボックスが表示されます。
4. ［**データベース名**］の欄で、**TUTORIAL**を選択します。
5. dmSQLコマンド・プロンプトに以下のコマンドを入力します。
```
dmSQL> CONNECT TO TUTORIAL SYSADM;
```
6. ENTERを押して下さい。
7. データベースを終了するときは、dmSQLコマンド・プロンプトに以下のように入力して下さい。
```
dmSQL> DISCONNECT;
```
8. ENTERを押して下さい。

## UNIX

➲ **UNIXでdmServerを起動する:**

1. コマンドラインに、次のように入力します。
```
$ cd ~DBMaster/<current version>/bin
```
*注: <current version> をDBMasterのバージョンに置き換えて下さい。例、4.0。*
2. ENTERを押して下さい。
3. カレント・ディレクトリが~DBMaster/version number/binに変わります。
4. コマンドラインに、次のように入力して下さい。
```
$ dmserver -u SYSADM TUTORIAL
```
5. ENTERを押して下さい。
6. DBMaster Serverが起動し、**Tutorial**データベースが立ち上がります。
7. 以下のメッセージが表示されます。:
```
DBMaster (current version number)
Copyright 1995-1999 CASEMaker Inc. All rights reserved.
SQL Server bound to port 54321
The database has started successfully.
Database Server is running in the background mode.
Process ID = 28030
```

8. コマンドラインに、次のように入力して下さい。
```
$ dmsqls
```
9. ENTERを押して下さい。
10. **dmSQL**アプリケーションが立ち上がります。
11. dmSQLコマンド・プロンプトに以下のコマンドを入力して下さい。
```
dmSQL> CONNECT TO TUTORIAL SYSADM;
```
12. ENTERを押して下さい。
13. データベースを終了する時は、以下のように入力して下さい。
```
dmSQL> TERMINATE DB;
```
14. ENTERを押して下さい。

# 5 表

データベースを作成しましたが、まだデータを保管するためのスペースがありません。つまり、空のファイリング・キャビネットのようなものです。ファイル・フォルダが無ければ、情報を保管することができません。データベースに情報を保管する前に、表を作成する必要があります。

表を作成する前に、それらをどこに作るか、又その中にどのような種類のデータを入れるのかを検討します。

## 5.1 表領域

DBMasterのデータベースは、*表領域*と呼ばれるいくつかの論理的なストレージ領域に仕切られています。これは、関連するデータが集まった表といった論理的な理由や、データを別々のディスクに配置するといった物理的な理由で、データベースを管理可能な領域に分けることです。つまり、データをまとめたり、別々の物理ディスクにデータを分散させたりすることで、アクセス時間を高速化します。

表領域には、サイズが固定しているものと、そのサイズが自動的に拡張するものがあります。サイズが固定している表領域は、*標準表領域*と呼ばれています。サイズが自動的に拡張する表領域は、*自動拡張表領域*と呼ばれています。またDBMasterには、*システム表領域*と*初期設定ユーザー表領域*と呼ばれる特殊な表領域があります。

### 標準表領域

標準表領域は固定サイズの表領域で、最低1つのデータファイルがあります。ファイルのサイズが、保存しようとするデータを全て入れるには十分

な大きさでない場合、ファイルを手動で拡大、或いは別のファイルを追加することできます。1つの標準表領域に、**32767**データファイルまで入れることができます。全ファイルのデータページの許容総数は**2**GBかそれ以下です。標準表領域は自動拡張表領域に変更することができます。

### 自動拡張表領域

自動拡張表領域は、データをファイルに格納するのに伴って拡大します。表領域を拡張させない時、又はサイズ許容値の2GBに近づいた時、自動拡張表領域を標準表領域に変更することができます。初期設定の表領域は、自動拡張表領域です。自動拡張表領域にあるデータファイルの初期サイズは、dmconfig.iniで指定したページ数です。

### システム表領域

DBMasterの全データベースに自動拡張のシステム表領域があります。データベースの作成時に、*システム・カタログ表*を記録するためのシステム表領域が必ず生成されます。システム・カタログ表はDBMasterによって管理され、データベースに保存されているあらゆる事についての詳細情報と統計があります。システム表領域に他の表を保存することはできません。

### 初期設定ユーザー表領域

DBMasterの全データベースには、自動拡張の初期設定表領域もあります。データベースの作成時に、ユーザー表を保管するための空の表領域が必ず生成されます。作成した表は全て、初期設定でここに保存されます。他の表領域に表を保管する場合は、表領域のディレクトリを指定します。

## 5.2 データ型

表のカラムを定義するときは、カラムのデータ型を指定します。不適切なデータ型を選ぶと、データベースのスペースを浪費し、アプリケーション・プログラムで使用可能な形式にデータを変換する余分なステップが必要になります。DBMasterは、16種のデータ型をサポートします。

## CHAR (サイズ)

CHARは、キーボード上の任意の文字をもつことができる固定長のデータ型です。CHARカラムの最小サイズは1字、最大サイズは**3992**字です。CHARカラムの作成時には、サイズ・パラメータの値を指定します。

CHARカラムにカラムサイズよりも短いデータを入力すると、スペースが補充されます。CHARデータを入力するときは、文字データを引用符 (‘ ’) で括ります。全角文字は2バイトを使用します。全角文字を入力するカラムのサイズを指定するときは、この点を考慮に入れます。

例　‘This is a CHAR string.’

‘This is another CHAR string.’

## VARCHAR (サイズ)

VARCHARは、任意の文字をもつことができる可変長のデータ型です。VARCHARカラムの最小サイズは1字、最大サイズは3992字です。VARCHARカラムの作成時には、サイズ・パラメータの値を指定します。

VARCHARカラムには、入力された文字だけが格納されます。データを入力するには、文字データを引用符(‘ ’)で括ります。全角文字は2バイトを使用します。全角文字を入力するカラムのサイズを指定するときは、この点を考慮に入れます。

例　‘This is a VARCHAR string.’

‘This is another VARCHAR string.’

## BINARY (サイズ)

BINARYは、固定長の任意のバイナリの値をもつことができるデータ型です。BINARYカラムの最小サイズは1バイト、最大サイズは**3992**バイトです。BINARYカラムを作成するときは、サイズ・パラメータの値を指定します。BINARYカラムにカラムサイズよりも短いデータを入力すると、0値のバイトが補充されます。

文字データは、文字データを引用符 (‘ ’)で括って入力します。BINARYカラムの文字データは、入力された文字ではなく、文字のASCIIコードを表す16進の数値として格納されます。

直接16進数を入力するには、16進文字列を引用符で括り後ろに16進の値であることを示す‘x’を付けます(‘ ’x)。各バイトの値を16進数で表すには2桁が必要です。偶数桁の16進数字を入力します。

例　‘AaBbCcDdEe’

‘41614262436344644565’x

## SMALLINT

SMALLINTは、精度**5**とスケール**0**をもつ符号付き真数のデータ型です。SMALLINTは2バイトのストレージを使用し、最大値**32,767**と最小値**-32,768**をもちます。

許容範囲以外の値をINTEGERやDOUBLEなどのデータ型から移そうとすると、変換エラーが表示されデータは移されません。

例　**4769**

**8376**

## INTEGER

INTEGERは、**10**桁の精度とスケール**0**をもつ符号付き真数のデータ型です。INTEGERデータ型は4バイトのストレージを使用し、最大値は**2,147,483,647**、最小値は**-2,147,483,648**です。INTEGERデータ型はINTと略記することができます。

許容範囲外の値を、DOUBLEなどのデータ型から移そうとすると、変換エラーが表示されデータは移せません。

例　**393848**

**-298376**

## FLOAT

FLOATは、7桁の精度をもつ符号付き概数のデータ型です。精度は小数点以上と以下の両方の合計桁数を表します。FLOATは4バイトのストレージを使用し、有効値の範囲は**3.402823466E38**～**–3.402823466E38**です。最小有効入力値は**1.175494351E-38**と**–1.175494351E-38**です。許容範囲外の値をDOUBLEなどのデータ型から移そうとすると、変換エラーが表示されデータは移せません。

例　**3.583837E34**

**-1.827362E-27**

## DOUBLE

DOUBLEは、15桁の精度をもつ符号付き概数のデータ型です。精度は小数点以上と以下の両方の合計桁数を表します。DOUBLEは8バイトのストレージを使用し、有効な入力値の範囲は**1.0E308**～**–1.0E308**です。最小有効入力値は、**–1.0E-308**と**1.0E-308**です。

例　2.89837457884451E285

-1.93873634847372E-174

## DECIMAL (NUMERIC)

DECIMALは、可変長の精度とスケールをもつ符号付き真数のデータ型です。精度は小数点以上と以下の両方の合計桁数を表し、スケールは小数点以下の桁数を表します。精度の初期設定値は**17**、最高精度は38、スケールの初期設定値は**6**です。

DECIMALカラムのストレージ・サイズは、実際に入力した数値で決められます。カラム定義で指定した精度とスケール、あるいは初期設定の精度とスケールではありません。DECIMALデータ型は、DECと略記することができます。

ストレージ・サイズは次の式で計算します。

$$\#\,\text{of bytes} = \frac{p+2}{2} + 2$$

例えば、**9283.83**は**6**バイトに格納されます。

このバイト数の計算を次ページに示します。

$$\#\text{ of bytes} = \frac{p+2}{2} + 2$$
$$= \frac{6+2}{2} + 2$$
$$= 6$$

許容範囲外の値をFLOATやDOUBLEなどのデータ型から移そうとすると、変換エラーが表示されデータは移せません。

例　　3452.8373645

　　736.383732652

## DATE

DATEは、日付(年、月、日)をもつ固定長のデータ型です。DATEデータ型は**6**バイトのストレージを使用し、**11**桁の精度をもちます。年の有効値は**0001**～**9999**です。

DATEデータ型には、多くの入力と出力のフォーマットがあります。データベース上の日付が正しく表示されなかったり、正しい日付なのに入力することができなかったりする場合は、日付の入力フォーマットと出力フォーマットが正しいかを確かめてください。

例　　‘0001/01/01’

　　‘1999/12/31’

## TIME

TIMEは、時刻をもつ固定長のデータ型です。6バイトのストレージを使用し、精度15とスケール3のデータ型です。オプションの‘AM’あるいは‘PM’を指定しないと、時刻は初期設定の24時間形式で挿入されます。

TIMEデータ型には、多くの入力フォーマットと出力フォーマットがあります。データベース上の時刻が正しく表示されなかったり、正しい時刻なのに入力することができなかったりする場合は、時刻の入力フォーマットと出力フォーマットが正しいかチェックして確かめてください。

例　‘22:04:05.666’

‘10:04:05.666 PM’

## TIMESTAMP

TIMESTAMPは、日付と時刻をもつ固定長のデータ型です。12バイトのストレージを使用し、精度27とスケール3をもちます。年の有効値は0001～9999です。オプションの‘AM’あるいは ‘PM’を指定しないと、時刻は初期設定の24時間形式で挿入されます。

TIMESTAMPは、TIMEとDATEの入力フォーマットと出力フォーマットを使用して値を表示し、入力値の正しさを検査します。データベース上の値が正しく表示されなかったり、正しい値なのに入力することができなかったりする場合は、入力フォーマットと出力フォーマットが正しいか確かめてください。

例　**‘1997/01/01 10:02:03.444 PM’**

**’01.01.1997 22:02:03.444’**

## LONG VARCHAR(CLOB)

LONG VARCHARは、可変長の任意の文字をもつことができるデータ型です。LONG VARCHARカラムの最大長は、約2,000,000,000字(2GB)です。

入力したデータだけがデータベースに格納されます。LONG VARCHARカラムに文字データを入力するときは、文字データを引用符(‘ ’)で括ります。全角文字は2バイトを使用します。全角文字を入力するカラムのサイズを指定するときは、この点を考慮に入れます。

例　‘This is a VARCHAR string.’

‘This is another VARCHAR string.’

## LONG BINARY(BLOB)

LONG VARBINARYは、可変長の任意のバイナリの値をもつことができるデータ型です。LONG VARBINARYカラムの最大長は、約2,000,000,000バイト(2GB)です。入力データだけがデータベースに格納されます。

文字データは、引用符(‘ ’)で括って入力します。LONG VARBINARYカラムは、実際に入力された文字ではなく、文字のASCIIコードを表す16進数としてデータを格納します。

直接16進数を入力するには、16進文字列を引用符で括り後ろに16進の値であることを示す‘x’を付けます(‘ ’x)。各バイトの値は2桁の16進数で表します。偶数桁の16進数字の値を入力します。

例　　‘AaBbCcDdEe’

　　‘41614262436344644565’x

## SERIAL (開始番号)

SERIALは、連続するシリアル番号を生成する特殊なデータ型です。データベースの各表には整数番号が一つ割り当てられており、この整数番号を使用して表に一意なシリアル番号が生成されます。DBMasterは各表の整数番号を内部的に管理・維持し、使用される度に自動的に1つ増分します。SERIALカラムを定義するときに、開始番号のオプション・パラメータを指定してシリアル番号の初期値を設定することができます。開始番号を指定しないと初期設定値の1になります。表は1個だけSERIALデータ型のカラムをもつことができます。

SERIALデータ型はINTEGERデータ型と同じ属性をもちます。SERIALは、4バイトのストレージを占め、精度10とスケール0をもつ符号付き真数のデータ型です。SERIALは、最大値2,147,483,647と最小値–2,147,483,648をもちます。

SERIALカラムに次のシリアル番号を挿入するには、新しいレコードを挿入するときにSERIALカラムにNULLを入力、又は空白にします。表の内部シリアル番号が新レコードのSERIALカラムに挿入され、自動的に1増分されます。

NULLの代わりに整数値を与えたSERIALカラムをもつ行を挿入すると、次のシリアル番号の代わりに与えた整数値が挿入されます。与えた値が内部シリアル番号より小さい場合は、内部シリアル番号は増分されません。与えた値が最後に生成したシリアル番号よりも大きいときは、内部シリアル番号が与えた値にリセットされます。

例　100, 101, 102, 103, 104, 105, 106, 107

100, 101, 50, 102, 103, 110, 111, 112

## FILE

FILEは、任意の既存ファイルを外部ファイルとして格納する特殊なデータ型です。他のデータと同様にDBMasterで参照することができます。データは、内部オブジェクトとしてではなく、外部ファイルとして格納されます。ファイルを使用する第三者ツールは、元のフォーマットでデータにアクセスし変更することができるだけでなく、データベースに変更を登録するためにデータを再インポートする必要も無くなります。

FILEカラムには、システム・カタログ表のレコードへの参照が格納されます。システム・カタログは、データベース上のファイル・オブジェクトを見つけるための情報をもっています。FILEカラムを表示しても、カラムに格納されている情報は表示されません。DBMasterは、ファイル名、ファイルサイズ、ファイル内容の3種類のビューのどれかで、システム・カタログやファイルに格納されている情報を表示します。

FILEデータ型は、システム・ファイル・オブジェクトまたはユーザー・ファイル・オブジェクトのいずれかでデータを格納することができます。

システム・ファイル・オブジェクトは、データベースのファイル・オブジェクト・ディレクトリに元のファイルをコピーして固有の名前を付けます。コピーされたファイルはデータベースによって管理され、FILEカラムからの参照が無くなると削除されます。

ユーザー・ファイル・オブジェクトは、元のファイルの場所と名前をそのままにし、ファイルへのリンクを作成します。ファイルはユーザーが作成したものであり、データベースからの参照が無くなっても削除されません。ユーザー・ファイル・オブジェクトとしてファイルを挿入するときは、ファイルの読み込み権限を*DBMaster*に与えておきます。

FILEは、24バイトのストレージを占める構造的なデータ型です。ファイル・オブジェクトのパス名の最大長は255字です。

### OID

OID(object identifier)は、データベースに格納される各オブジェクト、レコード、BLOBに割り当てられる一意IDのデータ型です。OIDは8バイトのストレージを使用し、精度10とスケール0をもつ構造的なデータ型です。OIDは自動的に生成され、各レコードに挿入されます。OIDは*DBMaster*によって内部的に管理・維持され、ユーザーが直接使用することはありません。

OIDの値は、オブジェクトのストレージ場所に関係します。連続して生成された二つのOIDが順序通り並んでいる保証はありません。OIDは表の隠しカラムです。“SELECT * FROM CUSTOMERS”のような問合せでは表示されません。OIDカラムを検索するには、問合せの中でカラム名として‘OID’を明示的に指定します。

問合せの中でOIDを使用して表からデータを検索し、そのOIDを使用して表のデータを更新することは可能ですが、*SQL*言語では、このような使用はあまり行いません。OIDは、通常、プログラミングの内部インターフェースに使用します。インタラクティブなdm*SQL*環境で直接使用することはありません。

## 5.3 表の作成

表は、その名称といくつかのカラムによって定義されます。各表とも、252カラムまで格納することができます。カラムは、カラム名とデータ型から成ります。INTEGERデータ型ではカラムのサイズを、DECIMALデータ型では精度とスケールを、SERIALデータ型のカラムには開始番号を、予め決定します。

表を作成する際には、表名、カラム定義、関連付ける表領域名を与えます。表領域に関連付けない場合、表は初期設定ユーザー表領域に作成されます。

➲ **例:**

Employeeという名前の表を作成する。:

```
dmSQL> CREATE TABLE Employee (Number SERIAL,   FirstName CHAR(15),
                              LastName CHAR(20),  Manager INT,
                                 Phone CHAR(15), HireDate DATE,
                             BirthDate DATE);
```

コマンド文字列は、次の7つのカラムを持つ表**Employee**を作成します。**Number**、**FirstName**、**LastName**、**Manager**、**Phone**、**BirthDate**、**HireDate**。**Number**カラムには、SERIAL番号を使用し、employeeを追加するたびに自動的に増加するemployeeナンバリング・スキーマを与えます。**FirstName**、**LastName**、**Phone**には、CHARデータ型を使います。電話番号には括弧やスラッシュ、ピリオドが含まれている可能性があるので、**Phone**カラムには、名前同様CHARデータ型を使用します。最後の2つのカラム**BirthDate**と**HireDate**は、DATEデータ型を使用します。

➲ **例:**

Tutorialデータベースに残りの表を作成する:

```
dmSQL> CREATE TABLE Regions (Number INT NOT NULL, Name CHAR(40))

dmSQL> CREATE TABLE IDTypes (Number INT NOT NULL,  Type CHAR(50),
                        Description CHAR(255))

dmSQL> CREATE TABLE Customers (Number SERIAL, FirstName CHAR(15),
                              LastName CHAR(20),   Phone CHAR(15),
                                IDType INT,     IDRegion INT,
                              IDNumber CHAR(20),  Credit SMALLINT)

dmSQL> CREATE TABLE Suppliers (Number SERIAL,      Name CHAR (50),
                                 Phone CHAR(15), Contact CHAR(35))

dmSQL> CREATE TABLE MovieTypes (Number INT NOT NULL, Type CHAR(30),
                            Description CHAR(255))

dmSQL> CREATE TABLE Movies (Number SERIAL,       Name CHAR(75),
                               Year CHAR(4),   Country CHAR(30),
                              Type1 INT,         Type2 INT,
                              Type3 INT,         Type4 INT,
                             Rating CHAR(10),   Length SMALLINT,
                              Color CHAR(3),        BW CHAR(3),
                               Tape CHAR(3),      Disc CHAR(3),
                           Quantity SMALLINT, Supplier INT,
                               Data DATE,        Price FLOAT,
                               Rate SMALLINT)

dmSQL> CREATE TABLE Receipts (Number SERIAL, Customer INT,
                             Employee INT)

dmSQL> CREATE TABLE LineItems (Receipt INT NOT NULL, LineItem INT NOT NULL,
                                 Movie INT,          Quantity SMALLINT,
```

```
Amount SMALLINT)
```

上記で新たに作成されたいくつかの表には、キーワード**NOT NULL**が使用されています。これは、新規レコードを挿入する際に、そのカラムに必ずデータを入れなければならないことを意味します。その他にも多くのキーワードを、表の作成時に使用します。

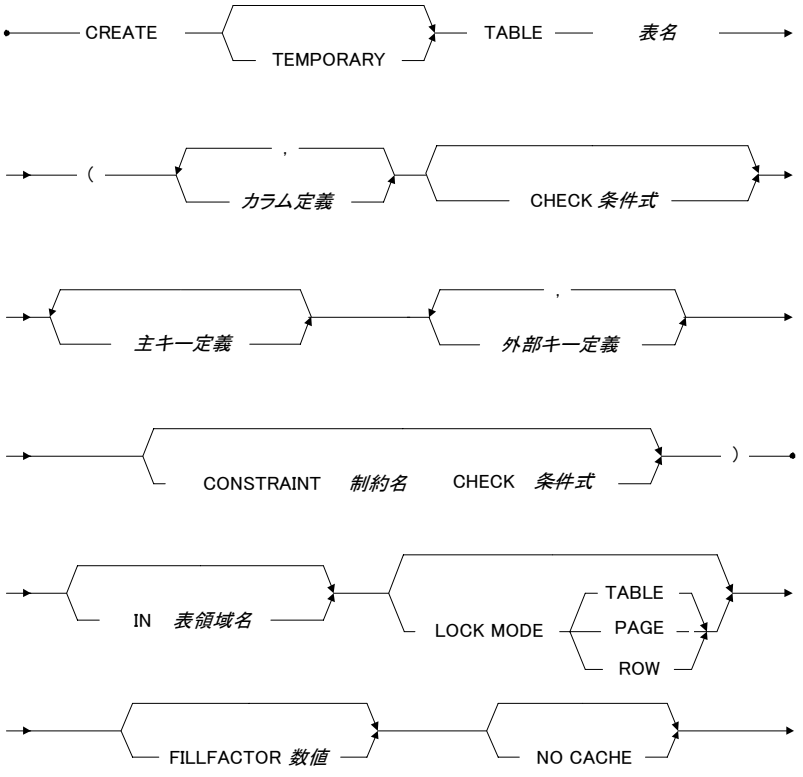



*図 5-1: CREATE TABLE 文の構文ダイアグラム*

## カラムの初期設定値

カラムには初期設定値が割り当てられています。新規行の挿入時にカラムの値が無い場合、或いはカラムが省略された場合、初期設定値が自動的に入力されます。カラム毎に別々の初期設定値を指定することができます。初期設定値を指定しない場合は、**NULL**にセットされます。指定することができる初期設定値は、定数又は組み込み関数です。

➲ **例:**

Suppliers表のContactカラムに初期設定値を設定する:

```
dmSQL> CREATE TABLE Suppliers (Number SERIAL,
                                 Name CHAR (50),
                               Phone CHAR(15),
                             Contact CHAR(35) DEFAULT 'Unknown')
```

## ロック・モード

LOCK MODEオプションは、データベースにアクセスする時に自動的にオブジェクトに設けられるロックの種類を表します。DBMasterでは、表／ページ／行の3レベルのロックを使用することができます。表の作成時にロック・モードを指定しない場合、初期設定のページ・ロックが使用されます。

ロック・モードが、表ロックのような高いレベルに設定されている場合、データベース・アクセスにおける同時実行レベルは低下しますが、必要なロック・リソースと共有メモリは少なくなります。ロック・モードが、行ロックのような低いレベルに設定されている場合、データベース・アクセスの同時実行レベルは向上しますが、必要なロック・リソースと共有メモリは多くなります。言い換えると、表ロックが設定されている表に行を挿入、又は修正すると、他の人はその表にアクセスすることができません。

➲ **例:**

表にロック・モードを設定する:

```
dmSQL> CREATE TABLE Suppliers (Number SERIAL,
                                 Name CHAR (50),
                               Phone CHAR(15),
                             Contact CHAR(35) default 'Unknown')
                                LOCK MODE ROW;
```

## フィルファクタ

FILLFACTORオプションは、各レコードが将来拡張できるようにデータページに一定のスペースを確保しておくことで、スペース利用を最適化します。つまり、ページの内容量が一定の割合に達した時点で、新規レコードを挿入できなくするように、その割合を指定します。この方法により、1レコードの情報を複数のページから回収する必要がなくなり、より効率的にレコードにアクセスすることができます。

➲ **例:**

Suppliers表のFILLFACTORを80%に指定する:

```
dmSQL> CREATE TABLE Suppliers (Number SERIAL,
                                 Name CHAR (50),
                                Phone CHAR(15),
                              Contact CHAR(35) default 'Unknown')
                                 LOCK MODE ROW FILLFACTOR 80;
```

この場合、ページ領域のデータが80%に達すると、このデータページにそれ以上新規行を挿入することはできません。フィルファクタの有効値は、50%～100%の間です。

## 非キャッシュ

NOCACHE機能は、表スキャンで大きな表にアクセスする際に役に立ちます。回収したデータをキャッシュし、頻繁なディスクI/Oを避けるために、共有メモリでページ・バッファが使用されます。ページ・バッファ数よりも大きいデータページでの表スキャンは、ページ・バッファの全てを費やし、頻繁なディスクI/Oアクティビティを引き起こします。表の作成時にNOCACHEオプションを指定すると、1ページのバッファを使用して、表スキャンの際に表から回収したデータをキャッシュします。1回の大きな表スキャンで、ページ・バッファが使い尽くされることはありません。

➲ **例:**

NOCACHEオプションを設定する:

```
dmSQL> CREATE TABLE Suppliers (Number SERIAL,
                                 Name CHAR (50),
                                Phone CHAR(15),
                              Contact CHAR(35) default 'Unknown')
                                 LOCK MODE ROW FILLFACTOR 80 NOCACHE;
```

## 一時表

一時的なデータ保存のために、一時表を作成することができます。一時表は、1セッションの間のみ存在し、データベースから切断した時に自動的に表から削除されます。一時表は高速データ操作をサポートし、作成者のみ使用することができます。

➲ **例:**

testという名前の一時表を作成する:

```
dmSQL> CREATE TEMPORARY TABLE test (Number SERIAL,
                                    Name CHAR(50),
                                   Phone DATE)
```

# 6 データ

表を作成し、データベースのスキーマがセットされました。データベースでデータを受け付ける準備ができました。ファイリング・キャビネットにファイルとファイル・フォルダがありますが、その中にまだ情報はありません。本章では以下について解説します。

- レコードや行を追加して、データベースにデータを挿入する方法
- 既存レコードを修正、又は更新する方法
- 表の中のデータを問い合せる方法
- 表から不要なデータを削除する方法

## 6.1 挿入

SQLを使って表にデータを挿入するには2つの方法があります。1つ目の方法は標準SQL構文を使い、もう一方の方法はホスト変数を使います。コマンド実行時に挿入されるデータが不明の場合に、挿入文にホスト変数は使用し、複数のレコード値が入力可能な*Value状態*にします。

➲ **例:**

SQLのINSERT文を使う:

```
dmSQL> INSERT INTO Employee VALUES (10000, 'Gabriel', 'Davis', 10000, '228-6932',
'1/21/57', '4/24/95');
1 row inserted
```

**Employee**は表の名前です。この文は、カラム**Number**、**FirstName**、**LastName**、**Manager**、**Phone**、**BirthDate**、**HireDate**に、値**10000**、**Gabriel**、**Davis**、**10000**、**228-6932**、**1/21/57**、**4/24/95**を挿入します。

➲ **例:**

指定したカラムにSQLのINSERT文を使う:

```
dmSQL> INSERT INTO Employee (FirstName, LastName, Manager) values ('Greg',
'Carter', 10002);
1 row inserted
```

この文は、**FirstName**、**LastName**、**Manager**カラムに、値**Greg**、**Carter**、**10002**を挿入します。指定しないカラムの値は、**NULL**になります。

➲ **例:**

NULLを付けて、SQLのINSERT文を使う:

```
dmSQL> INSERT INTO Employee VALUES (NULL, 'Dean', 'Cougar');
1 row inserted
```

この文は、通し番号順に次のSERIAL値を挿入し、**FirstName**と**LastName**カラムには値**Dean**と**Cougar**を挿入します。カラムが指定されていないので、自動的に最初の3つのカラムに値が入力されます。その他いくつかのキーワードがINSERT文には使用されます。

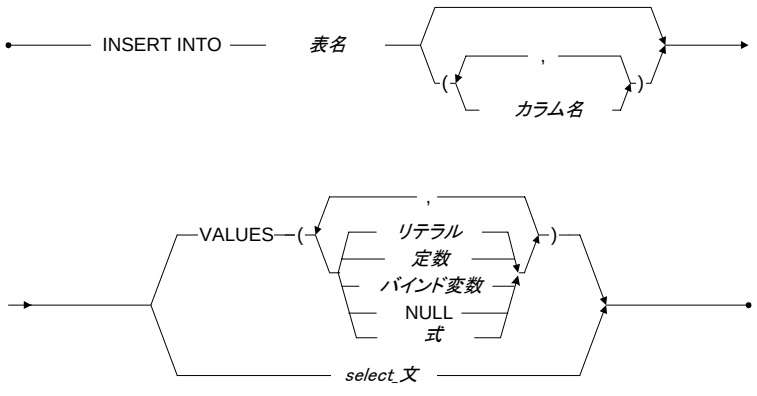



*図 6-1: INSERT 文の構文*

## ホスト変数を使った挿入

ホスト変数を付けたINSERT文の構文は、SQLの標準と同じです。ホスト変数をINSERT文に使うと、dmSQLのスクリーン・プロンプトが dmSQL/Val>のValue状態になります。

➲ **例:**

ホスト変数を使う:

```
dmSQL> INSERT INTO Employee VALUES (?,?,?,?,?,?,?);

dmSQL/Val> NULL, 'Benson', 'Armstrong', 10002, '918-3517', '12/9/70', '3/2/93';
1 row inserted

dmSQL/Val> NULL, 'Lyn', 'Belger', 10000, '363-4511', '5/9/59', '12/6/91';
1 row inserted

dmSQL/Val> end;

dmSQL>
```

➲ **例:**

同等のSQLのINSERT文を使う:

```
dmSQL> insert into Employee values (NULL, 'Benson', 'Armstrong', 10002, '918-
3517', '12/9/70', '3/2/93');

dmSQL> insert into Employee values (NULL, 'Lyn', 'Belger', 10000, '363-4511',
'5/9/59', '12/6/91');
```

➲ **例:**

INSERT文の一部にホスト変数を使う:

```
dmSQL> insert into Employee values (NULL, ?, ?, 10002, ?, ?, ?);

dmSQL/Val> 'Murphy', 'Flaherty', '575-8846', '10/17/77', '11/17/90';
1 row inserted

dmSQL/Val> 'Taylor', 'Galbreath', '648-6633', '2/9/75', '10/22/94';
1 row inserted

dmSQL/Val> end;

dmSQL>
```

➲ **例:**

同等のSQLのINSERT文を使う:

```
dmSQL> insert into Employee values (NULL, 'Murphy', ‘Flaherty’, 10002, ‘575-8846’,
‘10/17/77’, ‘11/17/90’);

dmSQL> insert into Employee values (NULL, 'Taylor', ‘Galbreath’, 10002, ‘648-
6633’, ‘2/9/75’, ‘10/22/94’);
```

## 様々なデータ型

DBMasterでは、以下の入力データ型を使用することができます。

SMALLINTとINTEGER:

```
123, -252783
```

FLOATとDOUBLE:

```
float:   30000.05, -234.56
double:  234.56e-257, 6.04E+23
```

CHARとVARCHAR:

```
'こんにちは', 'DBMasterは強力なデータベースです!'
```

BINARY:

dmSQLは、バイナリ型を認識する2つのフォーマットがあります。1つは16進数フォーマットで、もう1つはCHARデータ型と同じです。

CHARフォーマットでは、どの文字も1バイトを意味します。但し、データベースに保管された値は、文字のASCII値です。

CHARフォーマット:

```
'abcedf', '!@#$%'
```

16進数フォーマットでは、どの２つの16進数コードも1バイトを意味します。バイナリ・データを表すために、16進数コードの**0-9**とa-f (又はA–F)を使って下さい。16進数フォーマットのバイナリ・データは、「'」で括り、後ろにx又はXを付けます。

例、バイナリ・データ'**61**'x は、16進数フォーマットでは、'a'のASCII値です。:

```
'0001020304'x, '3f2eff5c'x
```

DATEとTIME:

```
'1994-12-20'd, '14:10:20't
```

DECIMAL:
```
 12.34, –0.123
```

## BLOBデータの挿入

DBMasterでは、BLOBデータを使うことができます。このデータ型は、LONG VARCHARとLONG VARBINARYです。LONG VARCHARとLONG VARBINARYの違いは、CHARとBINARYの違いと同じです。dmSQLでは、BLOBデータを挿入する方法がいくつかあります。

➲ **例:**

SQLコマンドでBLOBデータを挿入する:
```
dmSQL> CREATE TABLE blob_table VALUES (lcharcol long varchar,
    2> lbincol long varbinary);

dmSQL> INSERT INTO blob_table ('this is blob data', '2d2d2d2d'x);

1 row inserted
```

➲ **例:**

ホスト変数でBLOBデータを挿入する:
```
dmSQL> INSERT INTO blob_table VALUES (?,'5f5f5f5f5f'x);
```

➲ **例:**

ホスト変数にBLOBデータをバインドする:
```
dmSQL/Val> 'blob using host variable';

1 row inserted
```

替わりに、外部ファイルからBLOBデータを挿入することができます。

➲ **例:**

INSERTモードでファイル名を入力して、外部ファイルからBLOBデータを挿入する:
```
dmSQL/Val> &comment.txt;

1 row inserted
```

*注:　comment.txtは、現在のディレクトリにあるファイルです。*

# 6.2 更新

データベースにデータを挿入した後で、その値を変更する必要があるかもしれません。このような場合、SQLのUPDATE文を使います。DBMasterには、標準SQL、ホスト変数、OIDを使った3つのカラム・データの更新方法があります。

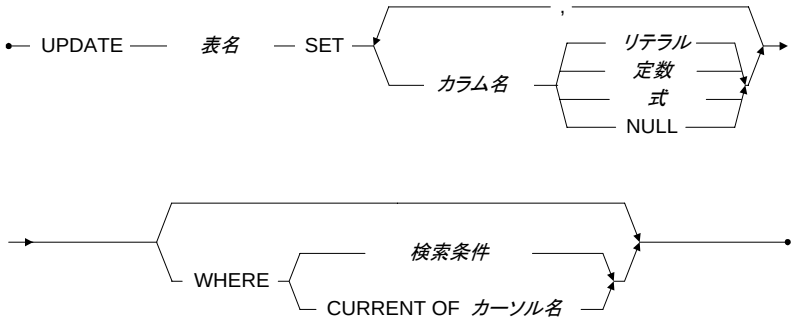



*図 6-2: UPDATE 文の構文*

## 標準SQLを使った更新

➲ **例:**

標準SQLを使って、Employee表を更新する:

```
dmSQL> update Employee set Manager = 10000 where LastName = 'Carter';

1 row updated
```

**Employee**は表の名前です。この文は**Greg Carter**のManagerカラムを10000に変更します。

## ホスト変数を使った更新

ホスト変数をUPDATE文に使うと、dmSQLのスクリーン・プロンプトが`dmSQL/Val>`のValue状態になり、対応するホスト変数のデータを入力することができます。END文でValue状態を終了し、更新文を完了することができます。

➲ **例:**

ホスト変数を使ってEmployee表を更新する:

```
dmSQL> update Employee set Phone = ? where FirstName = ? and LastName = ?;

dmSQL/Val> '736-8376', 'Gabriel', 'Davis';

1 row updated

dmSQL/Val> '837-7847', 'Lyn', 'Belger';

1 row updated

dmSQL/Val> end;

dmSQL>
```

➲ **例:**

標準SQLを使ってEmployee表を更新する:

```
dmSQL> update Employee set Phone = '736-8376' where FirstName = 'Gabriel' and
LastName = 'Davis';

dmSQL> update Employee set Phone = '837-7847' where FirstName = 'Lyn' and LastName
= 'Belger';
```

**Employee**表の従業員Gabriel DavisとLyn Belgerの**Phone**カラムが更新されます。

## OIDを使った更新

OID(オブジェクトid)は、オブジェクトのレコードIDを持つ特殊なバイナリ型のデータです。表にある各行は、一意のOIDです。表からOIDを選択し、そのデータを更新することができます。OIDは、内部プログラミング・インターフェースで使用し、インタラクティブdmSQL環境で直接使用することはありません。

➲ **例:**

OIDを使用してEmployee表を更新する:

```
dmSQL> select oid from Employee where FirstName = 'Dean' and LastName = 'Cougar';

      oid
===============
0200000002000200

1 rows selected

dmSQL> update Employee set BirthDate = '12/8/70' where oid='0200000002000200'x;
```

この入力は、Employee表のDean CougarのBirthdateを更新します。

# 6.3 結果セット

SELECT文の出力は、*結果セット*と呼ばれています。結果セットには、SELECT文で指定した条件に合致する全データがあります。

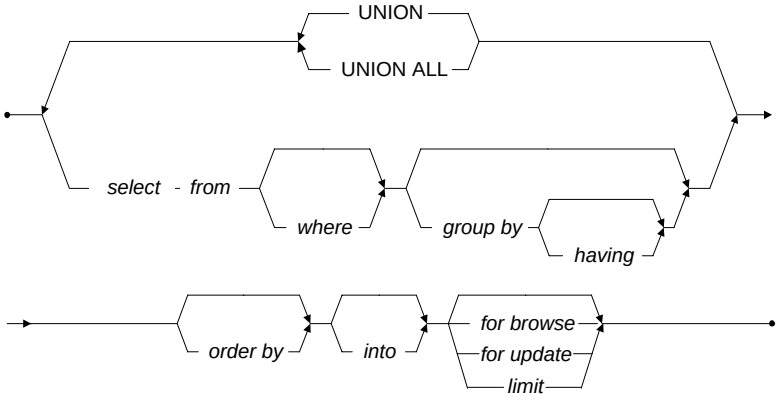



*図 6-3: SELECT文の構文*

## 表の選択

最も簡単なSELECT文は、表の全情報を選択します。

➲ **例:**

表の全情報を選択する:

```
dmSQL> select * from <table_name>
```

このコマンドでは、指定した表の全カラムからデータを選択します。アスタリスク(‘*’)は、表の全カラムを意味します。

**Employee**表から全データを選択する場合、以下のコマンドを使用します。

➲ **例:**

**Employee**表の全従業員のデータを選択する:

```
dmSQL> SELECT * from Employee;
```

戻りデータの例:

```
dmSQL> SELECT * from Employee;

 Number   FirstName     LastName     Manager     Phone    BirthDa* HireDate
======== =========== =============== ======== =========== ======== ========
   10000 Gabriel     Davis              10000 736-8376    1957-01* 1995-04*
   10001 Greg        Carter             10000    NULL       NULL     NULL
   10002 Dean        Cougar             NULL      NULL     1970-12*   NULL
   10003 Benson      Armstrong          10002 918-3517    1970-12* 1993-03*
   10004 Lyn         Belger             10000 837-7847    2059-05* 1991-12*
   10005 Murphy      Flaherty           10002 575-8846    1977-10* 1990-11*
   10006 Taylor      Galbreath          10002 648-6633    1975-02* 1994-10*

7 rows selected
```

dmSQLには、カラム名、そのカラムの各データ、表示された行数が表示されます。

これは、表のカラム名を覚えていない時に、システム表をチェックせずに、それらの名称を見つけることができる便利な方法です。また、全カラムのデータを見る場合、比較的速い方法です。

サンプル出力の全カラムが完全に表示されないかもしれません。dmSQLの表示ラインの幅は、初期設定で**80**にセットされています。フォーマット上の理由により、本書ではこの初期値を採用しています。

➲ **例:**

LINEWIDTH 機能をOFFにし、カラムの全データを見る:

```
dmSQL> SET LINEWIDTH off;
```

全カラムのデータ全てを見る場合は、SELECT文に全カラム名を使用します。**Employee**表にあるカラムの名前は、**Number**、**FirstName**、**LastName**、**Manager**、**Phone**、**BirthDate**、**HireDate**です。

➲ **例:**

全カラムのデータ全てを見る:

```
dmSQL> SELECT Number, FirstName, LastName, Manager, Phone, BirthDate, HireDate
from Employee;
```

戻りデータの例:

```
SELECT * from Employee;
  Number       FirstName          LastName         Manager        Phone
========== ============== ================== ========== ==============
 BirthDate   HireDate
========== ==========
      10000 Gabriel        Davis                   10000 736-8376
1957-01-21  1995-04-24
      10001 Greg           Carter                  10000      NULL
NULL        NULL
      10002 Dean           Cougar                 NULL          NULL
1970-12-08     NULL
      10003 Benson         Armstrong               10002 918-3517
1970-12-09  1993-03-02
      10004 Lyn            Belger                  10000 837-7847
2059-05-09  1991-12-06
      10005 Murphy         Flaherty                10002 575-8846
1977-10-17  1990-11-17
      10006 Taylor         Galbreath               10002 648-6633
1975-02-09  1994-10-22

7 rows selected
```

*注:　データベースにある名前は、大文字・小文字を識別します。*

カラム名入力の時に、スペルに間違いがある場合、或いは大文字と小文字が異なる場合、エラーが返ります。

➲ **例:**

```
dmSQL> select numbe, FirstName, LastName, Manager, Phone, HireDate, BirthDate from
Employee;
ERROR (6523): [DBMaster]無効なカラム名です
```

最初に起きたエラーのみ戻されます。間違いを正した後に更に間違いがある場合、全ての間違いが修正されるまで、別のエラーが返されます。

表名が間違っている場合、エラーが戻されます。

➲ **例:**

```
dmSQL> SELECT Number, FirstName, LastName, Manager, Phone, HireDate, BirthDate
FROM employee;
ERROR (6521): [DBMaster]表又はビューが存在しません
```

## カラムの選択

カラムを指定して、表から特定のカラムを選択することもできます。

➲ **例:**

カラムを指定する:

```
dmSQL> SELECT <column_name>, <column_name>, ... FROM <table_name>
```

コマンドは、カラム名のリストで指定したカラムを選択します。

➲ **例:**

Employee表のFirstName、LastName、Phoneカラムを見る:

```
dmSQL> SELECT FirstName, LastName, Phone FROM Employee;
```

戻りデータの例:

```
dmSQL>  SELECT FirstName, LastName, Phone FROM Employee;

   FirstName          LastName            Phone
=============== ==================== ==============
Gabriel         Davis                736-8376
Greg            Carter                    NULL
Dean            Cougar                    NULL
Benson          Armstrong            918-3517
Lyn             Belger               837-7847
Murphy          Flaherty             575-8846
Taylor          Galbreath            648-6633

7 rows selected
```

## 行の選択

特定カラムの選択同様に、データベースの特定のレコードを選択することができます。この場合、WHERE句が伴います。

結果セットに含むデータは、**WHERE**句で定義する条件式に合致している必要があります。条件に合致しないデータは、含まれません。WHERE句の使い方についての詳細は、6.4節の“演算子の種類”を参照して下さい。

➲ 例:

```
dmSQL> SELECT * FROM Employee where <expression>;
```

# 6.4 演算子の種類

WHERE句の式には、3種類の演算子を使用します。算術演算子、比較演算子、論理演算子です。これらの各演算子は、それぞれ別々の目的に使用します。最も頻繁に使用するのは比較演算子です。

## 比較演算子

比較演算子は、2つの演算子の値を比較するために使用します。一般的には、行を結果セットに含むかどうかを判断するために使用します。

比較演算子は以下のとおりです。

| 演算子 | 解説 |
|---|---|
| = | 数式の*等しい*を表します。 |
| > | 数式の*大なり*を表します。 |
| < | 数式の*小なり*を表します。 |
| >= | 数式の*大なりイコール*を表します。 |
| <= | 数式の*小なりイコール*を表します。 |
| <> | 数式の*等しくない*と同等の意味を表します。 |
| BETWEEN | 値の範囲を表します。 |
| LIKE | パターンマッチングを表します。 |
| IN | データベースのレコードを表します。 |

まず、比較演算子を使った極めて単純な条件句を試して、比較演算子がどのような役割を果たすか表示させます。

➲ **例:**

比較演算子を使う:

```
dmSQL> SELECT * FROM Employee WHERE Number = 10006;
```

戻りデータの例:

```
Number   FirstName     LastName     Manager     Phone    BirthDa* HireDate
======== =========== =============== ======== ========== ======== ========
   10006 Taylor      Galbreath          10002 648-6633    1975-02* 1994-10*

1 rows selected
```

この問合せでは、カラム一覧にアスタリスク('*')を使って、**Employee**表から全てのカラムを選択します。WHERE条件句は、numbers = 10006の従業員のレコードのみを結果セットに返すことを指定しています。この場合、従業員番号10006はTaylor Galbreathです。このような問合せは、レコードの一部分のデータだけがわかっていて、残るデータを表示させたい場合に有益です。

➲ **例:**

LastNameがA、B、Cで始まる全従業員を回収する:

```
dmSQL> SELECT * FROM Employee WHERE LastName BETWEEN 'Aa' and 'Cz';
```

戻りデータの例:

```
Number   FirstName     LastName     Manager     Phone    BirthDa* HireDate
======== =========== =============== ======== ========== ======== ========
   10001 Greg        Carter             10000    NULL       NULL     NULL
   10002 Dean        Cougar            NULL      NULL     1970-12*   NULL
   10003 Benson      Armstrong          10002 918-3517    1970-12* 1993-03*
   10004 Lyn         Belger             10000 837-7847    2059-05* 1991-12*

4 rows selected
```

この問合せでは、カラム一覧にアスタリスク('*')を使って、**Employee**表から全てのカラムを選択します。WHERE条件句で使用されるBETWEENキーワードは、AかBかCで始まるLastNameを持つ従業員のレコードが結果セットに戻されることを意味します。ANDで仕切られた2つの引数を取るBETWEEN比較演算子を使ってこれを実行します。このANDは、下記で説明する論理演算子ANDと同じように使用しますが、論理演算子ではなくBETWEENキーワードの一部であることに注意して下さい。A、B、Cで始まる全LastNameを取得するために、問合せには'Aa'と'Cz'の値を使用します。'A'と'C'の値のみ使用した場合、AとBで始まる名前のみ取得すること

になり、Cで始まる名前は取得しません。‘Cz’を使用することで、AaとCz間の全ての名前を取得することができます。AやCzarのような名前の人は、範囲外のために含まれません。

➲ **例:**

LastNameが‘C’で始まる従業員を取得する:

```
dmSQL> SELECT * FROM Employee WHERE LastName LIKE ‘C%’;
```

戻りデータの例:

```
 Number   FirstName     LastName     Manager     Phone    BirthDa* HireDate
======== =========== ============== ======== =========== ======== ========
   10001 Greg        Carter            10000    NULL       NULL     NULL
   10002 Dean        Cougar            NULL     NULL     1970-12*   NULL

2 rows selected
```

この問合せでは、カラムの一覧にアスタリスク(‘*’)を使うことで、**Employee**表から全てのカラムを選択します。WHERE条件句で使用されるLIKEキーワードは、名前が“C”で始まる従業員のレコードのみ返すよう指定します。ワイルドカード文字‘%’は、文字Cの後ろにあらゆる文字が付く可能性があることを指定するために使用します。このワイルドカードを省略すると、このSELECT文ではLastNameが‘C’の従業員のみ返します。

## 論理演算子

論理演算子は、WHERE条件句で2つの式の関係を表し、繋ぐために使用します。

論理演算子は、以下のとおりです。

| 演算子 | 解説 |
|---|---|
| AND | 2つの式が真であることを意味します。 |
| OR | いずれかが真であることを意味します。 |
| NOT | 等式に含まれない式を意味します。 |

➲ **例:**

1995年にカナダで製作された映画の名前を回収する:

```
dmSQL> SELECT LastName FROM Employee WHERE HireDate > 01/01/1995 AND Manager =
10000;
```

戻りデータの例:

```
      LastName
====================
Davis
Belger

2 rows selected
```

### 算術演算子

算術演算子は、数学的な計算を実行するために使用します。これは比較演算子の一部で、通常計算が実行されるまで結果がわかりません。

| 演算子 | 解説 |
|---|---|
| + | 数学的な足し算を意味します。 |
| - | 数学的な引き算/unary negationを意味します。 |
| * | 数学的な掛け算を意味します。 |
| / | 数学的な割り算を意味します。 |

## 6.5 削除

dmSQLを使った表から行を削除する方法は、標準SQL、ホスト変数、OIDの3つの方法があります。

### 標準SQLを使った削除

次のコマンドは、**Employee**表の全ての行を削除します。

➲ **例:**

Employee 表から全行を削除する:

```
dmSQL> DELETE FROM Employee;
```

次のコマンドは、**Employee**表から条件**Number > 10030**に合う全ての行を削除します。

➲ **例:**

Employee 表から行を削除する:

```
dmSQL> DELETE FROM Employee where Number > 10030;
```

## ホスト変数を使った削除

ホスト変数を使ったDELETE文は、dmSQLをdmSQL/Val>プロンプトのValue状態にし、対応するホスト変数のデータを入力できるようにします。ENDと入力すると、Value状態を終了しDELETE文を完了します。

➲ **例:**

ホスト変数を使ってEmployee表から従業員を削除する:

```
dmSQL> DELETE FROM Employee WHERE FirstName = ?;

dmSQL/Val> 'Benson';

dmQL/Val> 'Murphy';

dmSQL/Val> END;

dmSQL>
```

➲ **例:**

条件付きの標準SQLを使って前述の文を実行する:

```
dmSQL> DELETE FROM Employee WHERE FirstName = 'Benson';

dmSQL> DELETE FROM Employee WHERE FirstName = 'Murphy';
```

## OIDを使った削除

OIDを使ってデータを操作することは可能ですが、OIDはデータベース内部で使用される特殊なバイナリ型のデータです。一般的にはdmSQLで直接OIDを使用することはありません。

➲ **例:**

OIDを使ってEmployee表から従業員Taylorを削除する:

```
dmSQL> SELECT OID FROM Employee WHERE FirstName = 'Taylor';
```

```
      OID
================
0100000002000800

1 rows selected

dmSQL> DELETE FROM Employee WHERE OID='0100000002000800'x;
```

# 7 データベース･オブジェクト

データベースはオブジェクトと呼ばれる論理的な構造体に分かれています。表、表領域、ビュー、シノニム、索引などが、データベース・オブジェクトの例です。ビュー、シノニム、索引は、データアクセスをカスタマイズし、高速化させる便利な手段です。ビューとシノニムは、データベース・オブジェクト用のユーザー定義ビューや名前をサポートするために使用します。索引は、カラムに索引を付けることで、表からデータを回収する問合せをより速くします。

## 7.1 ビュー

DBMasterには、*ビュー*と呼ばれる*仮想表*を定義する機能があります。既存の表を元に、定義とユーザー定義のビュー名をデータベースに保存します。ビュー定義はデータベースに保存されますが、ユーザーが目にする実際のデータは、物理的にはどこにも保存されません。データはビューが参照する行がある元の表に保存されています。ビューは、最低1つ表や他のビューを参照する問合せによって定義されます。

ビューは、データベースを利用する上で大変役に立つメカニズムです。例えば、複雑な問合せを一度定義すると、それらを再度作成しなくても繰り返し使うことができます。加えて、ビューは予め定めた行やカラムにのみにアクセスを制限することで、データベースのセキュリティを強化することができます。

ビューが1つの表からの問合せで作成されている場合、そのビューを使用して、元の表の更新、挿入、削除することができます。但し、問合せに

DISTINCT/集計/結合/副問合せ/リモートを使用している場合は、不可能です。

## ビューの作成

各ビューは、表や他のビューを参照する問合せと名前によって定義されます。元の表にあるカラムを別々のビューにそれぞれ指定することができます。新しいカラム名を指定しない場合、ビューは元の表のカラム名をそのまま使用します。

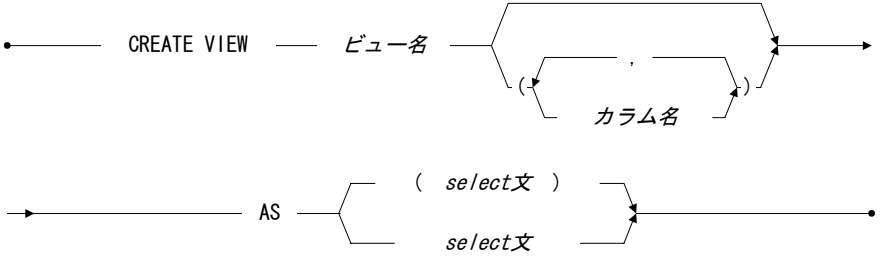



*図 7-1 CREATE VIEW文の構文*

➲ **例:**

Employee表の3つのカラムのみに閲覧を制限する:

```
dmSQL> CREATE VIEW empView (FirstName, LastName, Telephone) AS
              SELECT FirstName, LastName, Phone FROM Employee;
```

ユーザーは、**empView**で**Employee**表のカラムのうち**FirstName**、**LastName**、**Telephone**のみ見ることができます。

*注:　ビューを定義する問合せに、ORDER BY句やUNION演算子を含むことができません。*

## ビューの削除

必要の無くなったビューは削除することができます。ビューを削除すると、システム・カタログに保存された定義のみが削除され、元の表には影響しません。

➲ **例:**

ビューを削除する:
```
dmSQL> DROP VIEW empView;
```

**Employee**表から**empView**が削除されました。

# 7.2 シノニム

シノニムは、表やビューの別名、または替わりの名前です。シノニムは、単なる別名なので、システム・カタログにその定義を保存する以外にストレージを必要としません。

シノニムは、完全に認証された表やビューを単純化するのに役立ちます。DBMasterでは通常、表やビューを所有者とオブジェクト名からなる完全に認証された名前で識別します。シノニムを使うと、完全に認証された名前を使うことなくだれでも対応するシノニムを通じて表やビューにアクセスすることができます。DBMasterがそれらを識別するために、全シノニムはデータベース内で一意でなければなりません。

## シノニムの作成

表**Employee**の所有者が**SYSADM**の場合、このコマンドは、表**SYSADM.Employee**の別名**Employee**を作成します。データベースの全ユーザーは、シノニム**Employee**を通じて直接、表**SYSADM.Employee**を参照することができます。

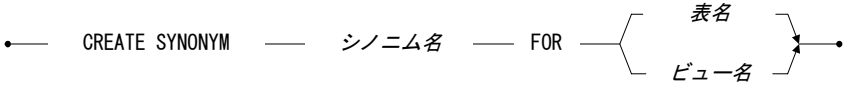



*図 7-2 CREATE SYNONYM文の構文*

➲ **例:**

シノニムEmployeeを作成する:
```
dmSQL> CREATE SYNONYM Employee FOR Employee;
```

### シノニムの削除

必要の無くなったシノニムは削除することができます。シノニムを削除すると、システム・カタログからその定義のみが削除されます。

➲ **例:**

シノニムEmployeeを削除する:

```
dmSQL> DROP SYNONYM Employee;
```

## 7.3 索引

索引は、行への高速ランダム・アクセスを可能にします。検索をスピードアップするために表に索引を設けます。例えば、“SELECT Name FROM Employee WHERE Number = 10005”のような問合せで、**NUMBER**カラムに索引が設定されている場合、より速くデータを回収することができます。

索引は、最高16までの複数のカラムで構成されています。1つの表には、252カラムまで入れることができますが、索引を作成することができるのは最初の**127**カラムだけです。

索引は、*一意*の場合と、*非一意*の場合があります。一意の索引では、NULL値を持つ行を除いて、複数の行が同じキー値を持つことができません。空でない表に一意の索引を作成する場合、DBMasterは既存の全キーが異なっているかどうかをチェックします。重複するキーがある場合、エラー・メッセージが返ります。表に一意の索引を作成した後に、表に行を挿入すると、同じキーを持つ行がないことを意味します。

索引を作成する時、各索引カラムのソート順を、昇順か降順に定義することができます。例えば、値**1**、**3**、**9**、**2**、**6**がある表に5つのキーがあるとします。昇順では、索引のキーの順序は、**1**、**2**、**3**、**6**、**9**になり、降順では、**9**、**6**、**3**、**2**、**1**になります。

問合せを実行した際に、索引の順序がデータ出力の並びに影響することがあります。

➲ **例、**

**NAME**カラムと**AGE**カラムがあるfriendsという名前の表があるとします。**AGE**カラムに降順索引を作成して、以下の問合せを実行します。

```
dmSQL> SELECT name, age FROM friends WHERE AGE > 20
```

その出力は以下のようになります:

➲ **例:**

```
      name                    age
----------------        ----------------
Jeff                              49
Kevin                             40
Jerry                             38
Hughes                            30
Cathy                             22
Cathy                             22
```

索引を作成する時にその*フィルファクタ*を指定します。フィルファクタは、そのキーが索引のページでどれぐらいの密度になるかを表します。正当なフィルファクタ値の範囲は**1%～100%**で、初期設定値は**100%**です。索引を作成した後に頻繁にデータを更新する場合、索引のフィルファクタを例えば**60%**のように、低くセットします。この表でデータを全く更新しない場合は、フィルファクタを初期設定のままにします。

膨大な量のデータをロードし終わる前に索引を作成するより、そのデータをロードした後に索引を作成するほうがより効率的です。表に索引を作成する場合、とりわけその表に大量のデータがある場合は、まず全データをロードすることをお勧めします。表にデータをロードする前に索引を作成する場合、新しい行をロードする度に索引を更新します。

## 索引の作成

索引名と索引にするカラムを指定して、表に索引を作成します。各カラムのソート順を、昇順(ASC)、降順(DESC)のいずれかに指定します。初期設定ソート順は、昇順です。

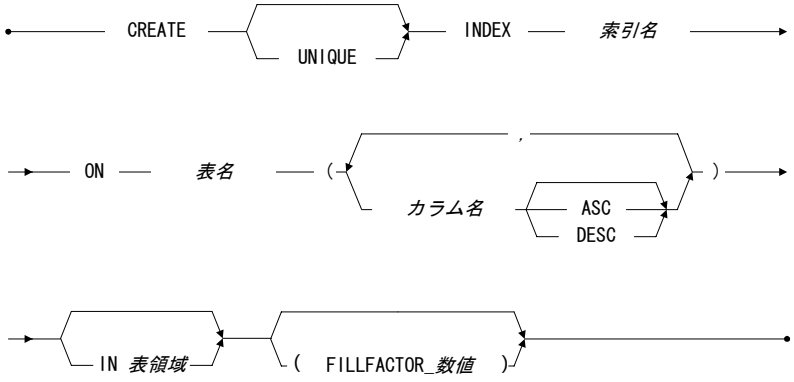



*図 7-3: CREATE INDEX 文の構文*

➲ **例:**

表Employeeのカラムに降順の索引idx1を作成する:

```
dmSQL> CREATE INDEX idx1 ON Employee (Number desc);
```

加えて、一意の索引を作成する際には、一意であることを指定します。さもないと、非一意索引が作成されます。

➲ **例:**

表Employeeのカラムに一意索引idx1を作成する:

```
dmSQL> CREATE UNIQUE INDEX idx1 ON Employee (Number);
```

➲ **例:**

フィルファクタを定義して、索引を作成する:

```
dmSQL> CREATE INDEX idx2 ON Employee(Number, LastName DESC) FILLFACTOR 60;
```

## 索引の削除

必要の無くなった索引は削除することができます。一般的に、索引が断片化されその効果が低減した時に、削除する必要があります。索引を再編成すると、より密度の濃い断片化されていない索引を作成します。索引が主キーで他の表に参照されている場合、その索引を削除することはできません。

➲ **例:**

表Employeeから索引idx1を削除する:

```
dmSQL> DROP INDEX idx1 FROM Employee;
```

# 8 ユーザーと権限

情報の保護は、データベースの重要な問題です。ユーザーがデータベースの全エリアに自由にアクセスしたり、データを意のままに変更したりすることは望ましことではありません。DBMasterには、データを保護しアクセスを制御する方法がいくつかあります。

## 8.1 セキュリティ管理

セキュリティ管理は、データが必要なユーザーにのみデータへのアクセスを制限する大切な要素です。

DBMasterのセキュリティ管理は以下のとおりです。

- **ユーザー・アカウント** — ユーザーIDとパスワードを使用して、データベースへのアクセスを制御します。
- **権限レベル** — ユーザーによって実行できるアクションを制御します。
- **表権限** — データを複数ユーザーで共有できるようにしたり、特定のユーザー専用にしたりします。
- **入れ子グループ** —同種のユーザーをグループにすることで権限の付与を単純化します。

一意のユーザーIDとオプションのパスワードで、データベースにアクセスする各ユーザーを認証します。各ユーザーには、それぞれに権限レベルが与えられます。これらユーザー権限レベルは、許可されたアクセスの種類を意味します。DBMasterには、CONNECT(接続)、RESOURCE、DBA(データベース管理者)、SYSADM(システム管理者)の4種類の権限レベルがあり

ます。CONNECT、RESOURCE、DBAのユーザーは複数存在するかもしれませんが、SYSADMのユーザーは一人だけです。SYSADMユーザーの役割は、データベースとユーザー・アカウントをセットアップし、保守することです。

表権限は、データへのアクセスを制御します。表権限を使うことで、ユーザーは表に作成されたデータにアクセスすることができます。表の許可は、ユーザーに表での操作を許可するために使用します。許可の種類は、データの閲覧、挿入、削除、更新、索引の作成、表の参照、新規カラムを追加した表の修正です。PUBLICキーワードを使えば、表権限を全ユーザーに与えることができます。

# 8.2 権限レベル

## CONNECT権限

データベースに接続するために、ユーザーには権限が必要です。接続権限のあるユーザーは、極めて制限されたデータベースへのアクセスを有することになります。これらのユーザーは、表やビューのような新規オブジェクトを作成することはできません。権限がPUBLICに与えられた表のデータのみ、閲覧し、修正することができます。表の所有者かDBA権のユーザーから表権限を与えられない限り、他のユーザーが作成した表を見たり修正したりすることはできません。表権限が与えられるか、表をPUBLICにすると、そのユーザーは表権限を使うことができます。

## RESOURCE権限

リソース権限は、データベースに表を作成したり、以前に作成した表を削除したりすることができる権限です。表の所有者も、ビューの作成や削除、作成した表に索引を設けることができます。又、所有する表の権限の付与と取り消しもできます。表の所有者かDBA権のユーザーから表権限を与えられない限り、他のユーザーが作成した表を見たり修正したりすることはできません。

### DBA権限

DBA権限は、データベース上の制御においてより大きな権限があります。このユーザーは、データベースの全ての表とビューの全権限を有します。DBAは、新規表やビューの作成と修正、全表の表権限を与えることができます。DBAは、表領域やデータファイルのような新しいデータベース構造も作成することができます。一度にアクセスするユーザー数を制御するために、DBAはグループを作成したり削除したり、グループにユーザーを加えたり削除したりすることもできます。但し、DBAユーザーは、現在のユーザー権限レベルの変更、データベース接続許可の付与、ユーザーのパスワードの変更を行うことはできません。

### SYSADM権限

各データベースにSYSADMは1人だけです。SYSADM IDは、データベースを作成したユーザーに初期設定で割り当てられます。SYSADMは、データベース上のあらゆる権限をもち、表やビューの作成、他ユーザーの表の削除、表への権限の付与、表領域やデータファイルの作成といったことが全てできます。SYSADMのみが、ユーザー権限レベルの付与/取り消し/変更、又ユーザー・パスワードの変更を行うことができます。

## 8.3 新規ユーザー

SYSADMのみがデータベースに新規ユーザーを追加する、つまりCONNECT権限を与えることができます。

➲ **データベースに新規ユーザーを追加する:**

1. データベースを起動し、**SYSADM**としてログインします。
```
dmSQL> START DB tutorial SYSADM <password>;
```

2. **SYSADM**パスワードで**Tutorial**に接続します。
```
dmSQL> CONNECT TO tutorial SYSADM <password>;
```

3. パスワードの無い新規ユーザーを作成する場合は、以下のいずれかを選択して下さい。
```
dmSQL> GRANT CONNECT TO Judy;
dmSQL> GRANT CONNECT TO Judy NULL;
dmSQL> GRANT CONNECT TO Judy “”;
```

4. パスワードを付ける場合は、以下のように入力します。

```
dmSQL> GRANT CONNECT TO Judy secret;
```

## ユーザー・アクセス

SYSADMとしてデータベースに接続すると、新規ユーザーを追加することができます。GRANTキーワードを使って、新規ユーザーを追加します。GRANTコマンドは、ユーザー名、パスワード、権限レベルを定義します。

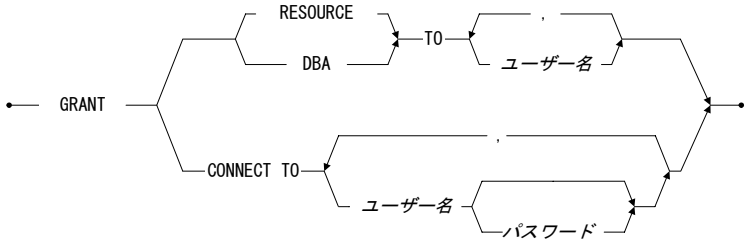



*図 8-1: GRANT 文の構文*

ユーザーIDは、最大32文字の英数字から成ります。ユーザーIDの最初の文字が数字の場合は、ダブル・クオート(“”)でユーザーIDを括って下さい。ユーザーIDは大文字と小文字を識別するので、名前が**judy**と**Judy**という別々のユーザーが存在する可能性があります。

ユーザーID同様、パスワードは最大16文字の英数字から成ります。最初の文字が数字の場合は、ダブル・クオート(“”)でパスワードを括って下さい。パスワードも、大文字と小文字を識別します。定義したものと同じように正確に入力して下さい。

ユーザーをRESOURCEやDBAのような高位の権限レベルに上げる前に、新規ユーザーにCONNECT権限を与えて下さい。新規ユーザーを作成する際に、パスワードを付けるか、パスワードの無いユーザーを作成して後でユーザーに作成させるかを選択します。セキュリティを考える場合は、ユーザーの作成時に一時的なパスワードを作成し、ユーザーが1回目にログオンする際にそのパスワードを変更するかどうかを確認するようにします。

データベースにログオンする際には、ユーザーIDを正確に入力する必要があります。統一性を持たせるために、新規ユーザーの作成時には一定の形式を使用することが理想的です。例えば、常に最初の文字を大文字にする、又は全文字を小文字とする、或いは全て大文字にするということです。

### 複数のユーザーの作成

同時に複数のユーザーを作成します。この構文は、コマンドラインに複数のユーザー名とパスワードを指定することを除いて、1ユーザーを追加する場合と同じです。

➲ **パスワードを設定しないで、2人の新規ユーザーを追加する:**

1. データベースを起動し、**SYSADM**としてログインします。
```
dmSQL> START DB tutorial SYSADM <password>;
```
2. **SYSADM**パスワードで**Tutorial**に接続します。
```
dmSQL> CONNECT TO connect to tutorial SYSADM <password>;
```
3. パスワードの無い新規ユーザーを作成する場合は、以下のいずれかを選択して下さい。
```
dmSQL> GRANT CONNECT TO Tom, Judy;
dmSQL> GRANT CONNECT TO Tom “”, Judy “”;
dmSQL> GRANT CONNECT TO Tom NULL, Judy NULL;
```
4. パスワードを付ける場合は、以下のように入力します。
```
dmSQL> GRANT CONNECT TO Tom secret1, Judy secret2;
```

## 8.4 権限レベルを上げる

GRANTキーワードは、既存ユーザーの権限レベルを上げるために使用します。これは、新規ユーザーにCONNECT権限を与える手順と同じです。ユーザーは既にパスワードを持っていますので、別の人間がそのユーザーIDを使うことにならない限り、新しいパスワードを指定しないで下さい。

➲ **例:**

CONNECT権限のユーザーをRESOURCE或いはDBA権限に上げる:
```
dmSQL> GRANT RESOURCE TO Judy;
dmSQL> GRANT DBA TO Judy;
```

ユーザーにDBA権限を与えることは、自動的にRESOURCE権限を与えることにはなりません。これは、ユーザー権限レベルを下げる時に重要になります。ユーザーにRESOURCEとDBA権限の両方を与えようとする時、GRANT文を2度使う必要があります。RESOURCE権限レベルを与え、更にDBA権限レベルを与えます。

### 複数のユーザーの権限を上げる

一度に複数のユーザーの権限を上げることができます。この構文は、コマンドラインに複数のユーザー名を指定する以外は、1ユーザーの権限を上げる場合と同じです。これらのユーザーは、同じ権限レベルに上がります。1つのコマンドで、複数のユーザーを異なる権限レベルに上げることはできません。

➲ **例:**

2ユーザーの権限を上げる:

```
dmSQL> GRANT RESOURCE TO Tom, Judy;
dmSQL> GRANT DBA TO Tom, Judy;
```

## 8.5 権限レベルを下げる

ユーザーの権限レベルを下げる方法は、ユーザーの権限レベルを上げる手順に似ています。GRANTの替わりにREVOKEキーワードを使います。

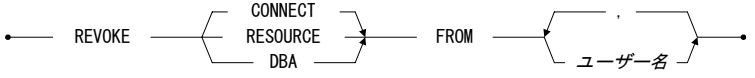



*図 8-2: REVOKE 文の構文*

ユーザーの権限レベルを取り消すと、そのユーザーの権限は現在与えられている権限レベルのうちで1つ低いレベルに落ちます。例えば、CONNECT権限とDBA権限がありRESOURCE権限が無いユーザーは、DBA権限を取り消された場合、CONNECT権限しかありません。

➲ **例:**

ユーザーのDBA権限を取り消す:

```
dmSQL> REVOKE DBA FROM Judy;
```

➲ **例:**

ユーザーをRESOURCE権限からCONNECT権限に下げる:

```
dmSQL> REVOKE RESOURCE FROM Judy;
```

*注: ユーザーに*RESOURCEとDBA*権限が与えられている場合は*、DBA*と*RESOURCE*権限の両方を取り消し*、CONNECT*権限に下げます。*

➲ **例:**

DBA権限のユーザーをCONNECT権限に下げる:

```
dmSQL> REVOKE DBA FROM Judy;
dmSQL> REVOKE RESOURCE FROM Judy;
```

## 8.6 ユーザーを削除する

既存のユーザーを削除するには、REVOKEキーワードを使います。既存のユーザーを削除することは、ユーザー権限を下げることです。CONNECT権限を取り消されたユーザーは、データベースに接続できなくなります。

➲ **例:**

データベースからユーザーを削除する:

```
dmSQL> REVOKE CONNECT FROM Judy;
```

## 8.7 パスワード

SYSADMは、ユーザーの作成時に一時的なパスワードを割り当てます。SYSADMが一時パスワードを割り当てなかった場合、或いはこのパスワードを変更しようとする場合、ユーザーはALTER PASSWORDコマンドで新しいパスワードを設定することができます。SYSADMが、新しいパスワードを再度設定することも可能です。ユーザーがパスワードを忘れた場合にこの手段を利用します。

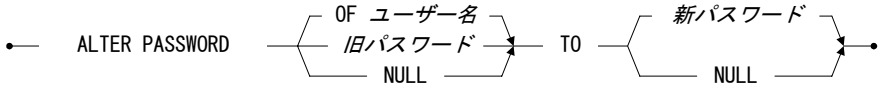



*図8-3: ALTER PASSWORD文の構文*

➲ **例:**

ユーザーのパスワードを作成する:

```
dmSQL> ALTER PASSWORD NULL TO newpass;
dmSQL> ALTER PASSWORD “” TO newpass;
```

➲ **例:**

ユーザーの一時パスワードを変更する:

```
dmSQL> ALTER PASSWORD X9elx4 TO newpass;
```

現在のパスワードをNULLに変更するとパスワードを削除できます。

➲ **例:**

パスワードをNULLに変更する:

```
dmSQL> ALTER PASSWORD oldpass TO NULL;
dmSQL> ALTER PASSWORD oldpass TO “”;
```

SYSADMは、ユーザーのパスワードを変更/削除することができます。それ以外のユーザーは、他のユーザーのパスワードを変更することはできません。SYSADMは、現在のパスワードがわからなくても、パスワードを変更することができます。

➲ **例:**

SYSADMは以下のいずれかを入力して、ユーザーのパスワードを変更する:

```
dmSQL> ALTER PASSWORD OF Judy TO newpass;
dmSQL> ALTER PASSWORD OF Judy TO NULL;
dmSQL> ALTER PASSWORD OF Judy TO “”;
```

# 8.8 グループを管理する

データベースが大きくなりユーザー数が増えると、個々にユーザー権限を与えるのが大変になります。この問題を解決するために、DBMasterでは、複数ユーザーをグループにまとめます。グループを使えば、同じデータベース権限を必要とするユーザーを集め、権限の管理を単純化することができます。グループ内の全てのユーザーに一斉に表権限を付与/取り消すといったことが可能です。

## グループの作成

新規グループを作成するには、CREATE GROUPコマンドを使用します。


*図 8-4: CREATE GROUP 文の構文*

グループ名は、英数字、文字、アンダースコア、記号$と#で構成されています。グループ名の長さに制限はありませんが、最初の32文字のみが使用されます。グループに名前を付けるときには注意して下さい。例えば、グループ**TaipeiDepartmentMarketingCompanyEmployee**と**TaipeiDepartmentMarketingCompanyExecutives**には、ともに同じ文字**TaipeiDepartmentMarketingCompany**が含まれています。同じ文字を含むグループを作成しようとすると、エラーになります。このような状況をさけるために、グループ名を32文字以下に制限し、長い説明的な名前を使用しないようにします。

➲ **例:**

マーケティング部の全社員のグループを作成する:

```
dmSQL> CREATE GROUP marketing;
```

marketingという名前の新規グループに、マーケティング部の社員全員を追加します。それ以降にこのグループに権限を与えると、メンバー全員が同じオブジェクトへのアクセス権をもつことになります。

## メンバーの追加

ユーザーをグループに追加します。

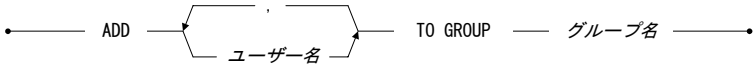



*図 8-5: ADD TO GROUP 文の構文*

➲ **例:**

グループSalesRepに、ユーザーJudyを追加する:

```
dmSQL> ADD Judy TO GROUP SalesRep;
```

➲ **例:**

グループSalesRepに、ユーザーJudy、Jeff、Trentを追加する:

```
dmSQL> ADD Judy, Jeff, Trent TO GROUP SalesRep;
```

## メンバーの削除

グループからユーザーを削除します。

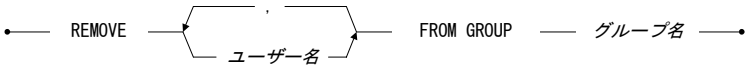



*図 8-6: REMOVE FROM GROUP 文の構文*

➲ **例:**

グループSalesRepから、ユーザーJudyを削除する:

```
dmSQL> REMOVE Judy FROM GROUP SalesRep;
```

➲ **例:**

グループSalesRepから、ユーザーJudy、Jeff、Trentを削除する:

```
dmSQL> REMOVE Judy, Jeff, Trent FROM GROUP SalesRep;
```

## グループの削除

グループにユーザーがいなくなった時、或いはグループが必要なくなった時、DROP GROUP文でグループを削除することができます。

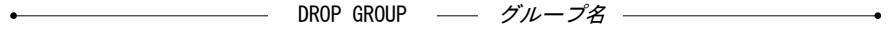



*図8-7: DROP GROUP文の構文*

グループにユーザーが残っている場合、上記REMOVE FROM GROUP文でで全てのユーザーを削除して下さい。

➲ **例:**

グループが空であることを確認して、グループSalesRepを削除します:

```
dmSQL> DROP GROUP SalesRep;
```

## 入れ子グループ

入れ子グループは、グループの機能性を更に強化します。入れ子グループを作成するために、ADD TO GROUPE文を使い、ユーザー名の部分にグループ名を指定します。入れ子グループを削除する場合は、REMOVE FROM GROUP文を使います。

➲ **NYSalesという名前のグループを作成する**:

1. グループNYSalesを作成します。
   ```
   dmSQL> CREATE GROUP NYSales;
   ```
2. **SalesRep**グループにそれを追加します。
   ```
   dmSQL> ADD NYSales TO GROUP SalesRep;
   ```
3. さらに、それを削除します。
   ```
   dmSQL> REMOVE NYSales FROM GROUP SalesRep;
   ```

# 8.9 表レベル権限

表の所有者、或いはDBA権限のユーザーが使用できる表権限は7つです。SELECT、INSERT、DELETE、UPDATEの4つの権限は、ユーザーが表のデータを閲覧したり、修正したりすることができる権限です。残る3つの権

限の1つINDEXは、索引を作成することができる権限です。ALTERは、表の定義を変更することができる権限です。REFERENCEは、表に参照整合性を設定することができる権限です。表の権限は、特定のカラムへの変更をセットするためにも使用します。

## SELECT（選択）

SELECT権限は、表やビューのあらゆるデータを参照することができる権限です。表の所有者かDBA権限のユーザーによって与えられます。

## INSERT（挿入）

INSERT権限は、表に新規データを挿入することができる権限です。表全体または特定のカラムにINSERT権限を与えます。特定のカラムにINSERT権限を与える場合、INSERTキーワードの後にカラム名を追加します。表の所有者かDBA権限のユーザーによって与えられます。

## DELETE（削除）

DELETE権限は、表からデータを削除することができる権限です。表の所有者かDBA権限のユーザーによって与えられます。

## UPDATE（更新）

UPDATE権限は、表の値を更新することができる権限です。表全体または特定のカラムにUPDATE権限を与えます。特定のカラムにUPDATE権限を与える場合、UPDATEキーワードの後にカラム名を追加します。表の所有者かDBA権限のユーザーによって与えられます。

## INDEX（索引）

INDEX権限は、表に索引を作成することができる権限です。表の所有者かDBA権限のユーザーによって与えられます。

## ALTER（修正）

ALTER権限は、表の定義を修正することができる権限です。表の所有者かDBA権限のユーザーによって与えられます。

### REFERENCE（参照）

REFERENCE権限は、表に外部キーを作成することができる権限です。表全体または特定のカラムにREFERENCE権限を与えます。特定のカラムにREFERENCE権限を与える場合、REFERENCEキーワードの後にカラム名を追加します。表の所有者かDBA権限のユーザーによって与えられます。

## 8.10 表権限を与える

表の所有者或いはDBA権限のユーザーは、ユーザーへ表全体または特定のカラムへの表権限を与えることができます。

表に権限を割り当てるために、GRANTコマンドを使います。

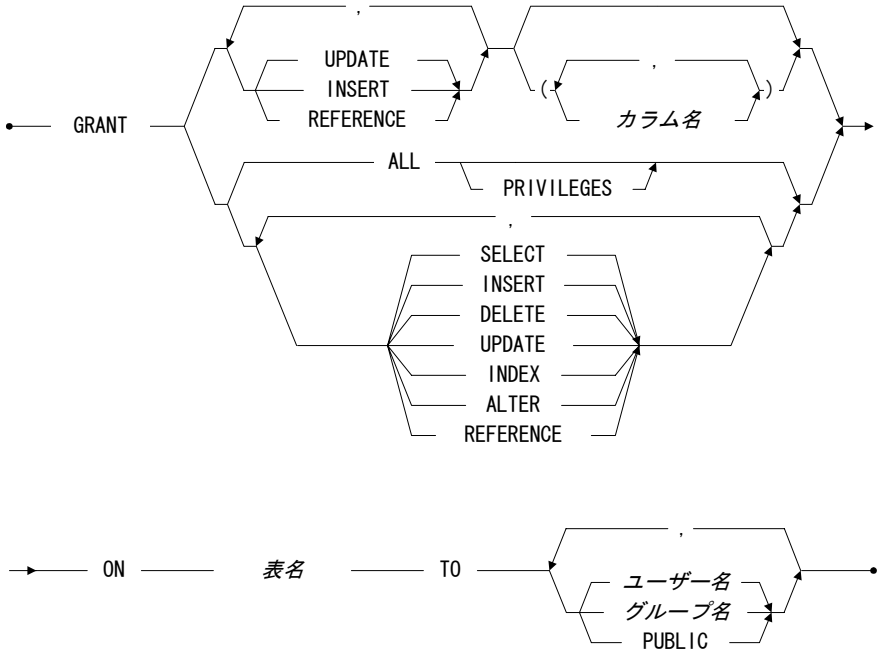



*図8-8: GRANT文の構文*

表全体と特定カラムに同時に権限を与えることはできません。表全体に権限を与えるコマンドと、特定カラムに権限を与えるコマンドを、2回実行します。シングルユーザーやグループ、或いはPUBLICキーワードを使って全員に権限を与えることができます。

## 表全体への表権限を与える

➲ **例:**

JudyにSalesRep表へのSELECT権限を与える:

```
dmSQL> GRANT SELECT ON SalesRep TO Judy;
```

一度に複数の権限を与えることもできます。コマンドラインに権限を列記し、カンマで区切ります。

➲ **例:**

JudyにSalesRep表へのSELECTとUPDATE権限を与える:

```
dmSQL> GRANT SELECT, UPDATE ON SalesRep TO Judy;
```

コマンドラインに全てのキーワードを列記するか、ALLキーワードを使うと、ユーザーに表の全権限を与えることができます。

➲ **例:**

JudyにSalesRep表への全ての表権限、SELECT、INSERT、UPDATE、DELETE、ALTER、INDEX、REFERENCEを与える:

```
dmSQL> GRANT ALL ON SalesRep TO Judy;
```

複数のユーザー名を指定すると、複数のユーザーに権限を与えることができます。

➲ **例:**

JudyとJeffにSalesRep表へのSELECTとUPDATE権限を与える:

```
dmSQL> GRANT SELECT, UPDATE ON SalesRep TO Judy, Jeff;
```

複数のグループ名を指定すると、複数のグループに権限を与えることができます。

➲ **例:**

グループPersonnelとSalesMgrに表SalesRepへのSELECTとUPDATE 権限を与える:

```
dmSQL> GRANT SELECT, UPDATE ON SalesRep TO Personnel, SalesMgr;
```

*注:　一度に複数の表の権限を与えることはできません。*

## 特定カラムへの表権限を与える

特定カラムにのみINSERT、UPDATE、REFERENCE権限を与えることもできます。カラムに権限を与える際に、同時に表全体に他の権限を与えることはできません。

➲ **例:**

JudyにSalesRep表のNameカラムへのINSERT権限を与える:

```
dmSQL> GRANT INSERT (Name) ON SalesRep TO Judy;
```

➲ **例:**

JudyにSalesRep表の複数カラムへのINSERT権限を与える:

```
dmSQL> GRANT INSERT (Name, Age, RepOffice, Title) ON SalesRep TO Judy;
```

一度に複数の権限を与える時、その権限の全ては同じカラムへ適用されます。1回のコマンドで異なるカラムへの別々の権限の与えることはできません。

➲ **例:**

JudyにSalesRep表のNameとAgeカラムへのUPDATE、INSERT、REFERENCE権限を与える:

```
dmSQL> GRANT INSERT, UPDATE, REFERENCE (Name, Age) ON SalesRep TO Judy;
```

表に権限を与える場合と同様、複数のユーザーやグループにカラムへの表権限を与えることができます。ユーザーやグループ名を列記し、カンマで区切ります。

# 8.11 表権限を取り消す

表の所有者或いはDBA権限のユーザーは、ユーザーへ表全体または特定のカラムからユーザーの権限を取り消すことができます。REVOKEコマンドで、権限を取り消します。

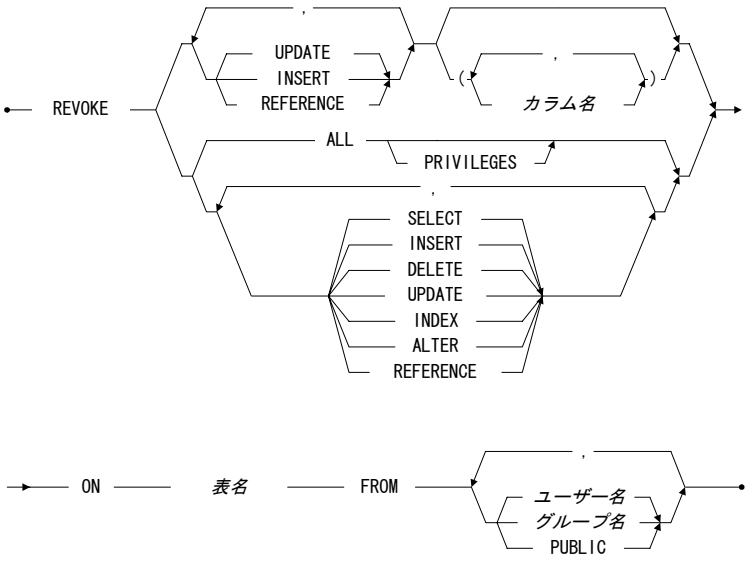



*図 8-9: REVOKE 文の構文*

表全体への権限と特定カラムへの権限を、同時に取り消すことはできません。表全体への権限を取り消すコマンドと、特定カラムへの権限を取り消すコマンドを、別々に実行します。シングルユーザーやグループ、或いはPUBLICキーワードで全ユーザーへの権限を取り消します。

## 表全体への権限を取り消す

➲ **例:**

JudyからSalesRep表へのSELECT権限を取り消す:

```
dmSQL> REVOKE SELECT ON SalesRep FROM Judy;
```

コマンドラインに取り消す権限を列記し、カンマで区切ると、複数の権限を取り消すことができます。

➲ **例:**

JudyからSalesRep表へのSELECTとUPDATE権限を取り消す:

```
dmSQL> REVOKE SELECT, UPDATE ON SalesRep FROM Judy;
```

全てのキーワードを列記するか、ALLキーワードを使うと、ユーザーの全ての表権限(SELECT、INSERT、UPDATE、DELETE、ALTER、INDEX、REFERENCE)を取り消すことができます。

➲ **例:**

JudyからSalesRep表への全ての表権限を取り消す:

```
dmSQL> REVOKE ALL ON SalesRep FROM Judy;
```

複数のユーザー名をカンマで区切って指定すると、複数のユーザーの権限を取り消すことができます。

➲ **例:**

JudyとJeffからSalesRep表へのSELECTとUPDATE権限を取り消す:

```
dmSQL> REVOKE SELECT, UPDATE ON SalesRep FROM Judy, Jeff;
```

ユーザー名をグループ名に置き換えると、グループの権限を取り消すことができます。

➲ **例:**

PersonnelとSalesMgrグループから、SalesRep表へのSELECTとUPDATE権限を取り消す:

```
dmSQL> REVOKE SELECT, UPDATE ON SalesRep FROM Personnel, SalesMgr;
```

*注:　一度に複数の表の表権限を取り消すことはできません。*

## 特定カラムへの表権限を取り消す

特定カラムへのINSERT、UPDATE、REFERENCE権限を取り消すことができます。カラムへの権限を取り消す際に、同時に表全体の他の権限を取り消すことはできません。

➲ **例:**

JudyからSalesRep表のNameカラムへのINSERT権限を取り消す:

```
dmSQL> REVOKE INSERT (Name) ON SalesRep FROM Judy;
```

➲ **例:**

JudyからSalesRep表の複数カラムへのINSERT権限を取り消す:

```
dmSQL> REVOKE INSERT (Name, Age, RepOffice, Title) ON SalesRep FROM Judy;
```

1つのコマンドで複数の権限を取り消す場合、その権限全てが指定するカラム全部に適用されている必要があります。

➲ **例:**

JudyからSalesRep表のNameとAgeカラムへのUPDATE、INSERT、REFERENCE権限を取り消す:

```
dmSQL> REVOKE INSERT, UPDATE, REFERENCE (Name, Age) ON SalesRep FROM Judy;
```

表の権限を取り消す場合と同様、複数のユーザーやグループの特定カラムへの表権限を取り消すことができます。ユーザー名やグループ名をカンマで区切って列記します。

# 9 データベース・リカバリ

いかなるデータベース管理システムにも、ハードウェアやソフトウェアの障害の可能性は常に存在します。DBMSは、警告も無くこのような障害によって被害を被ることがあります。障害が起こった場合、DBMSにはデータを回復する方法がいくつかあります。これは、DBMSがファイル・ベースのシステムにまさる大きな長所の1つです。

DBMasterには、障害によるデータの損失や故障時間を防ぐための高度なデータ保護機能が組み入れています。そのリカバリ、バックアップ、リストアの各機能で、DBMasterのデータベースの信頼性とデータの整合性を強固なものとします。

## 9.1 障害の種類

データベース障害の最も一般的な２種類は、システム障害とメディア障害です。障害が発生した時、データ不整合やデータベースからデータを損失する可能性があります。DBMSは、障害から復活し、損傷したデータベースをバックアップしておいたコピーに置き換える機能を備えていなければなりません。

### システム障害

インスタンス障害として知られるシステム障害は、メイン・メモリの障害です。システム障害は、パワー障害やアプリケーションやオペレーション・システムのクラッシュ、或いはそれ以外の理由によって起こります。これにより、データベース管理システムは予想できない結果になります。

システム障害が起こったとき、アプリケーションとアクティブ・トランザクションは、不正常に終了するかもしれません。実行中のトランザクションの正確な状態や、完全にディスクに書き込まれなかったトランザクションを、システム障害の後に正確に判断することはできません。このようなトランザクションは、リカバリが必要です。システム障害からデータを保護する最も一般的な方法は、トランザクション・ログとジャーナル・ファイルの利用です。

### メディア障害

ディスク障害として知られるメディア障害は、ディスク・ストレージ・システムの障害です。メディア・エラーは、ヘッドクラッシュ、火災、振動、又は物理的な操作限界を超えた重力等の、ディスクへの物理的なトラウマによって引き起こされます。

一般的に、メディア障害発生の際に影響を受けたディスクのデータ損失を防ぐ手段はありませんが、バックアップとリストア機能を利用するば、データベースをリストアすることができます。

## 9.2 リカバリの方法

データベース障害後のリカバリの最終目標は、コミットされたトランザクションをデータベースに確実に反映させること、或いはコミットされていないトランザクションをデータベースに反映させないこと、障害による問題からユーザーを隔離していている間に、できる限り迅速に通常の操作に戻すことです。

これらの目標を達成するために、ジャーナル・ファイルとチェックポイントを使います。ジャーナル・ファイルとチェックポイントは、合わせて利用し、可能な限り1回の短い時間で全トランザクションのリカバリを行います。

### ジャーナル・ファイル

ジャーナル・ファイルは、データベースへの全変更の現時点と過去の記録と各変更の状態です。システム障害の際には、ジャーナル・ファイルに残

された変更の履歴を使って、実行された後にディスクに書き込まれていないトランザクションによる変更を復活・再試行、或いは不正常に終了したトランザクションによる変更を元に戻します。

データベースがバックアップ・モードで運用されている場合、ジャーナル・ファイルは、DBMasterで使用される追加データも保存します。このデータは、バックアップされるまでジャーナル・ファイルに残ります。ジャーナル・ファイルをバックアップした後、新しいトランザクション用にスペースが解放されます。リストア処理の間、バックアップ・ジャーナル・ファイルからの情報が、データベースのバックアップ・コピーに追加されます。これは、完全バックアップの間にデータベースに加えた変更を含むジャーナル・ファイルのみをバックアップします。

## チェックポイント・イベント

チェックポイントは、データベースをきれいな状態にするシステムのイベントです。DBMasterは、内部メモリ・バッファの全ジャーナル・レコードと汚れたデータページを全てディクスに書き込み、バックアップやリカバリ用に必要の無くなったジャーナル・ブロックを再使用します。最も古いアクティブ・トランザクションの開始前に完了した非アクティブ・トランザクションを含むジャーナル・ブロックを再使用することができます。

システム障害の後にチェックポイントを取ると、起動時間が削減します。DBMasterは、前回のチェックポイントの日時とチェックポイント時の全アクティブ・トランザクションの一覧を、ジャーナル・ファイルのヘッダーに書き込みます。データベース・リカバリの際、どのトランザクションを元に戻し、再試行、無視するべきかを決定するために、この情報が使用されます。

いくつかのジャーナル・ブロックを再利用しようとした時に、ジャーナル・ファイルが一杯である場合、自動的にチェックポイントが取られます。チェックポイントが現在のトランザクションを完了するために充分なスペースを捻出できない場合、トランザクションは中止されます。データベースの起動と終了やオンライン・バックアップ時にも、自動的にチェックポイントが取られます。

データベース管理者がCHECKPOINT文を実行して、チェックポイントを手動で初期化することができます。2つのチェックポイント間の最適間隔は、データベースでの操作間隔、トランザクションの平均サイズ、ジャーナル・ファイルのサイズと数によります。これらの要素は、データベースごとで劇的に異なるので、経験を通じてのみ決定することができます。手動チェックポイントは、データベースの起動と終了、或いはデータベースや完全ジャーナルのバックアップに要する時間を削除します。

前チェックポイント以降のトランザクションのサイズと数によっては、チェックポイントを完了するまで非常に多くの時間を要します。チェックポイントが発生した際にアクティブのトランザクションは、どのジャーナル・レコードを再利用するかを計算している間、全て中断されます。ジャーナル・レコードと汚れたデータページが、ディスクに書き込み始められる際、トランザクションは処理されます。

## リカバリ処理

DBMasterでは、システム障害後にデータベースを起動しようとする場合、或いは起動時にエラーが発生した場合、自動リカバリを利用することができます。リカバリ処理では、常に再試行と元に戻すの2つの異なるステップが実行されます。

リカバリ処理の最初のステップは、ジャーナルに記録されたデータベースへの全変更を再試行することです。データベースに書き込まれたトランザクションによる変更を採用せずに、システム障害前に完了したトランザクションのみ採用することが可能です。但し、これらの変更はジャーナルに保管され、このステップの際にデータベースに書き込まれる可能性があります。このステップが終わると、データベースにはコミットされた全トランザクションによる変更と、コミットされていない全トランザクションによる変更があります。

リカバリ処理の2つめのステップは、システム障害が発生する前に完了しなかったトランザクションによる変更を元に戻すことです。システム障害が起こった場合、進行中のトランザクションの正確な状態は、信頼性があるとはみなされないので、そのまま続行することはできません。トランザクションは自己完結のものであるので、トランザクションを順調に実行し

データを変更するか、又は失敗してデータを変更せずおくかのいずれかになります。そのため、これらの不完全なトランザクションは、削除しなければなりません。このステップが終わると、データベースにはコミットされた全トランザクションによる変更がありますが、コミットされていないトランザクションによる変更はありません。

DBMasterでは、自動リカバリ処理では修復することができないような、データベースの不整合をもたらすメディア障害やシステム障害後に、データベースを起動することも可能です。このような場合、通常データベースは起動できず、ユーザーはバックアップ・コピーからデータベースをリストアします。但し、データベースを一度もバックアップしたことが無い場合、dmconfig.iniファイルのDB_FORCSキーワードを強制起動モードに設定して、データベースを強制的に起動させることができます。これにより、データベースを起動し、影響を受けていないデータをアンロードすることができます。強制起動モードについての詳細は、*「データベース管理者参照編」*をご覧下さい。

## 9.3 バックアップの種類

バックアップは、メディア障害などの問題からデータベースを保護するために使用します。メディア障害の後、いくつかのデータベース・ファイルは、物理的に損傷して使用できなくなっているかもしれません。このような場合、最新のバックアップを使って、損傷を受けたファイルを取り替え、データベースを再構築します。

データベースの主要な2種類のバックアップは、*完全バックアップ*と*差分バックアップ*です。また*オンライン*と*オフライン*の2つの状態で、実行されます。バックアップの種類とデータベース状態を組み合わせて、様々なバックアップを行うことができます。

### 完全バックアップ

完全バックアップは、全データと全ジャーナル・ファイルのコピーをとり、ある時点のデータベース全体の複製を作ります。同様に、データベースが使用しているユーザー仕様の環境設定を保存するために、dmconfig.ini

ファイルのバックアップをとっておくこともできます。DBMasterでは、オンライン時でもオフライン時でも、完全バックアップを実行することができます。

完全バックアップは、データベース全体をアーカイブするので、大量のストレージ領域を必要とします。但し、損傷を受けた元のデータベースにバックアップ・ファイルをコピーするだけなので、比較的すばやくデータベースを回復することができます。完全バックアップを使用すると、前回の完全バックアップ時、つまりデータベース・ファイルをコピーした時点までデータベースをリストアすることができます。

## 差分バックアップ

差分バックアップは、前回のバックアップ以降に変更したジャーナル・ファイルのコピーのみを作成します。これらのファイルには、前回のバックアップ以降にデータベースに加えた変更のコピーがあります。差分バックアップは、データベースへの変更しかバックアップしないので、データベースのリストアを行う時のために、完全バックアップを実行する必要があります。DBMasterでは、データベースがオンライン時にのみ、差分バックアップを実行することができます。

差分バックアップは、ジャーナル・ファイルのみをアーカイブするので、僅かのストレージ領域しか必要ありません。但し、バックアップしたジャーナル・ファイルを使って全トランザクションをロールオーバーするために、完全バックアップでデータベースをリストアするよりも多くの時間がかかります。差分バックアップと完全バックアップを併用すると、前回の完全バックアップから差分バックアップの完了までのいかなる時点へもデータベースをリストアすることができます。

## オフライン・バックアップ

オフライン・バックアップは、データベースを終了してから実行します。データベース管理者は、データベースを終了するスケジュールを決め、全ユーザーにデータベースから切断することを伝えなければなりません。データベース終了前に、全てのアクティブ・トランザクションを完了させ、データベースから切断しなければならないので、ユーザーにとっては不便

であるかもしれません。オフラインの間は差分バックアップを実行することができません。

データベースを終了した後では、オペレーティング・システムを使ってデータベースをバックアップすることができます。そのため、DBMSにオフライン・バックアップのための機能は必要がありませんが、DBMasterには、オペレーティング・システムの使用に頼らないで、オフライン・バックアップを実行することができる、使いやすいグラフィカルなツールのJServer Managerがあります。

## オンライン・バックアップ

オンライン・バックアップ は、データベースの起動中に実行することができます。データベース管理者がデータベースを終了したり、ユーザーがデータベースから切断したりする必要がありません。ユーザーにとっては、特別なアクションが必要ないのでより便利です。DBMasterは、オンラインで完全と差分バックアップを実行することができます。

データベースを起動し続け、ユーザーを接続させ続けるためには、オンライン時にデータベースのバックアップを可能にする能力がDBMSに要求されます。DBMasterのdmSQLやオペレーティング・システムのコマンドを使った手動のオンライン・バックアップも可能ですが、使いやすいグラフィカルなツールのJServer Managerを利用することもできます。JServer Managerは、オペレーティング・システムの使用に頼らないで、オンライン・バックアップを実行することができます。

## バックアップの組み合わせ

DBMasterには、完全バックアップと差分バックアップがありますが、それらはお互いに独立しています。つまり、完全バックアップと差分バックアップを一緒に実行することはできません。同様に、DBMasterではデータベースがオフライン時、又はオンライン時にバックアップを実行することができます。オンライン時/オフライン時と完全/差分バックアップを組み合わせる場合、特定の組み合わせは実行できません。

DBMasterでは、オフライン完全バックアップ、オンライン完全バックアップ、オンライン差分バックアップの組み合わせを使用することができます。その他に現在までのオンライン差分バックアップも実行することができます。

複数のジャーナル・ファイルのあるデータベースの場合、オンライン差分バックアップと現在までのオンライン差分バックアップの違いは、些細ではありますがとても重要なことです。オンライン差分バックアップでは、アクティブ・ジャーナル・ファイルを除く、前回のバックアップ時以降に使用された全てのジャーナル・ファイルがバックアップされます。現在までのオンライン差分バックアップでは、それに加えアクティブ・ジャーナル・ファイルもバックアップされます。これは、オンライン差分バックアップが、最後にコミットされたトランザクションが最後の完全ジャーナル・ファイルに書き込まれた時点まで、データベースをリストアするのに対し、現在までのオンライン差分バックアップは、アクティブ・ジャーナル・ファイルがバックアップされた時点まで、データベースをリストアすることを意味します。

ジャーナル・ファイルが1つしかないデータベースでは、オンライン差分バックアップと現在までのオンライン差分バックアップは同じです。この場合、その唯一のジャーナル・ファイルがアクティブ・ジャーナル・ファイルになります。いずれの差分バックアップでも、この唯一のジャーナル・ファイルがバックアップされます。

## 9.4 バックアップ・モード

バックアップ・モードは、オンライン差分バックアップを実行するかどうかや、差分バックアップ時にバックアップするデータの種類を指定します。また、他のトランザクションで使用するために、非アクティブのトランザクションに属するジャーナル・ブロックをいつ解放するかどうかを決定します。DBMasterには、NONBACKUP、BACKUP-DATA、BACKUP-DATA-AND-BLOBの3つのバックアップ・モードがあります。

## NONBACKUPモード

NONBACKUPモードは、前回の完全バックアップ以降に挿入、更新されたデータを保護しません。このモードでは、オンライン差分バックアップは使用できません。システム障害の場合、完全にリカバリするためにジャーナルを使うことができますが、メディア障害の場合はデータの損失をもたらすかもしれません。アクティブ・トランザクションによって使用されるジャーナル・ブロックは、チェックポイント後にすぐ再使用することができますが、一旦上書きされると前回の完全バックアップの時点までしかデータベースをリストアすることができません。

## BACKUP-DATAモード

BACKUP-DATAモードは、前回の完全バックアップ以降に挿入、更新されたBLOBデータ以外のデータを保護します。このモードでは、オンライン差分バックアップを実行することができますが、非BLOBデータのみバックアップ・ファイルに保管することができます。システム障害の場合は、ジャーナルを使って完全にリカバリすることができ、メディア障害の場合は、部分的にリカバリすることができます。前回のバックアップを使って、メディア障害の時点までデータベースをリストアすることができますが、BLOBデータへの変更は全て消失します。アクティブ・トランザクションが使用していないジャーナル・ブロックは、チェックポイントを取ってジャーナル・ファイルがバックアップされた後にのみ、再使用することができます。

## BACKUP-DATA-AND-BLOBモード

BACKUP-DATA-AND-BLOBモードは、前回の完全バックアップ以降に挿入、更新されたBLOBデータを含む全データを保護します。このモードでは、オンライン差分バックアップを実行し、全データをバックアップ・ファイルに保存することができます。システム障害の場合、ジャーナルを使って完全にリカバリすることができ、メディア障害からも完全にリカバリすることができます。前回のバックアップを使って、メディア障害が発生した時点まで、BLOBデータを含めデータベースを完全にリストアするこ

とができます。アクティブ・トランザクションが使用していないジャーナル・ブロックは、チェックポイントを取ってジャーナル・ファイルがバックアップされた後にのみ、再使用することができます。

## 表領域BLOBバックアップ・モード

DBMasterは通常、バックアップ・モードの設定をデータベース全体に適用し、同様にデータベースの全表領域にも適用します。データベースがBACKUP-DATA-AND-BLOBモードの場合、DBMasterはデータへの全変更をジャーナルに記録します(BLOBデータを含む)。ジャーナルにBLOBデータを記録すると、ジャーナル・スペースを速く消費し、頻繁なバックアップと大きなバックアップ・ファイルを生み出す結果となります。

これは、BLOBデータの損失が許されない場合に必要なことですが、多くの場合、同時にさほど重要でないBLOBデータもバックアップしています。このようなケースでは、どのバックアップ・モードを選択するかは難しい問題です。こういう状況を解決するために、DBMasterでは表領域の作成時に個々の表領域のバックアップ・モードを修正することが可能です。

特定の表領域でBLOBデータをバックアップしようと考える場合は、CREATE TABLESPACE文を実行する時に、BACKUP BLOB ONオプションを指定します。特定の表領域でBLOBデータをバックアップしないようにする場合、CREATE TABLESPACE文を実行する時に、BACKUP BLOB OFFオプションを指定します。

各表領域のバックアップ・モードは、以下のデータベースのバックアップ・モードと表領域のバックアップ・モードの組み合わせによります。

- データベースがBACKUP-DATA-AND-BLOBモードで、表領域がBACKUP BLOB ONオプションで作成された場合、その表領域のBLOBデータはバックアップされます。
- データベースがBACKUP-DATA-AND-BLOBモードで、表領域がBACKUP BLOB OFFオプションで作成された場合、その表領域のBLOBデータはバックアップされません。

- データベースがBACKUP-DATA モードの場合、表領域がBACKUP BLOB ONかBACKUP BLOB OFFオプションで作成されたに関わらず、BLOBデータはバックアップされません。

初期設定は、BACKUP BLOB ONです。この場合、データベースがBACKUP-DATA-AND-BLOBモードの時、表領域のBLOBデータへの変更は、全てジャーナル・ファイルに記録されます。

## ファイルオブジェクト・バックアップ・モード

データベースの一般データとBLOBデータのバックアップに加え、ファイルオブジェクトのバックアップを選択することができます。ファイルオブジェクトは、バックアップデーモンによって自動完全バックアップの際にのみバックアップされます。まずバックアップ・サーバーを起動し、完全バックアップ・スケジュールをセットし、バックアップ・ディレクトリをセットする必要があります。

ファイルオブジェクトには、ユーザー・ファイルオブジェクトとシステム・ファイルオブジェクトの2種類があります。システム・ファイルオブジェクトのみバックアップすることもできますし、両方ともバックアップする、或いはしないことも可能です。**dmconfig.ini**のキーワードDB_BkFoMは、ファイルオブジェクトのバックアップ・モードを指定します。

- **DB_BkFoM** = 0: ファイルオブジェクトをバックアップしない。
- **DB_BkFoM** = 1: システム・ファイルオブジェクトのみバックアップする。
- **DB_BkFoM** = 2: システム・ファイルオブジェクトとユーザー・ファイルオブジェクト双方ともバックアップする。

ファイルオブジェクトをバックアップする時(**DB_BkFoM** = 1、2)、バックアップ・サーバーは、ファイルオブジェクトの全外部ファイルを**DB_BkDir**キーワードで指定したディレクトリのサブディレクトリの“*fo*”にコピーします。スケジュールは、**DB_FBkTm**と**DB_FBkTv**で指定した完全バックアップのスケジュールに基づきます。

➲ 例

関連キーワードを含む**dmconfig.ini**ファイルからの引用:

```
[MyDB]
DB_BkSvr = 1                          ; バックアップ・サーバーを起動させる
DB_FBKTm = 01/05/01 00:00:00          ; 2001年5月1日の深夜に開始
DB_FBKTV = 1-00:00:00                 ; 1日間隔
DB_BkDir = /home/DBMaster/backup      ; バックアップ・ディレクトリ
DB_BkFoM = 2                          ; システムFOとユーザーFO両方をバックアップする
```

ファイル・オブジェクトのバックアップ・モードが2なので、バックアップ・サーバーはデータベースの全外部ファイルオブジェクトを、"**/home/DBMaster/backup/FO**"ディレクトリにコピーします。**FO**サブディレクトリが存在しない場合、バックアップ・サーバーはそれを生成します。

FOサブディレクトリにあるファイルは、連続する番号に名前を付け替えられます。例えば、元の外部ファイルの名前が"**/DBMaster/mydb/fo/ZZ000123.bmp**"の場合、バックアップ・サーバーはそれを**FO**サブディレクトリにコピーし、344番目のファイルオブジェクトを意味する、'**fo0000000344.bak**'という名前に付け替えます。ソースの完全なファイル名とその新しい名前の間のマッピングは、ファイルオブジェクトのマッピング一覧ファイル**dmFoMap.his**に保存されます。

バックアップ・サーバーはまた、古いバージョンのファイルオブジェクトを**DB_BkOdr**で指定した古いバックアップ・ディレクトリにある**FO**サブディレクトリに移動します。

データベース管理者は、ファイルオブジェクトのバックアップを使用可能にすると、完全バックアップにより時間がかかる点を考慮する必要があります。完全バックアップ全体のコストには、(1)**DB_BkOdr**にセットしている場合、以前の完全バックアップのコピー; (2)全データベース・ファイルのコピー; (3)全ジャーナル・ファイルのコピー;(4)**DB_BkFoM**にセットしている場合、全ファイルオブジェクトのコピーが含まれます。また、バックアップ・エラーを防ぐために、**DB_BkDir**で指定したバックアップ・ディレクトリに全バックアップ・ファイルのために十分なスペースがあることを確認して下さい。

## バックアップ・モードの設定

DBMasterには、バックアップ・モードをセットする方法がいくつかあります。採用する方法は、データベースがオンラインかオフラインか、或いは直接環境設定を編集することに慣れているかどうかや、dmSQLコマンドライン・ユーティリティやJConfiguration Toolのグラフィカル・ユーティリティを使うかどうかによります。

バックアップ保護のレベルを上げるために、データベースのバックアップ・モードを変更すると、ジャーナル使用に影響します。ジャーナルは、バックアップ・モードを変更する前に、今まで保存されていないデータ変更を記録し始めます。結果として、バックアップ・モードを変更する時に完全バックアップを実行することになります。これは、リストア処理時に、バックアップ・ジャーナル・ファイルを更新するための開始ポイントを設けることです。

バックアップ保護のレベルを下げるために、データベースのバックアップ・モードを変更する場合、必要な操作はありません。単にジャーナルへのデータ変更の保存が中止されるだけです。リストア処理時に更新するバックアップ・ジャーナル・ファイルの開始ポイントとして、以前の完全バックアップが使用されます。但し、バックアップ保護を低いレベルへの変更した後にデータベースをリストアする場合、データ変更は消失しているかもしれません。

dmconfig.iniファイルやJConfiguration Toolを使うと、データベースのバックアップ・モードをオフラインで変更することができます。バックアップ・モードはジャーナル使用に影響するので、新しいバックアップ・モード設定でデータベースを起動する前に、オフライン完全バックアップを実行する必要があります。バックアップ・モードをオフラインで変更する場合、いずれのバックアップ・モードからでも自由に別のバックアップ・モードに変更することができますが、バックアップ保護を低レベルから高レベルに移行する際には、完全バックアップを行います。オフライン完全バックアップについての詳細は、この章の9.5節の「オフライン完全バックアップ」を参照して下さい。JConfiguration Toolを使ったバックアップ・モード

の変更方法については*「JConfiguration Toolユーザーガイド」*をご覧下さい。

dmSQLを使うと、オンラインでデータベースのバックアップ・モードを変更することができます。バックアップ・モードはジャーナル使用に影響するので、完全バックアップの開始から終了までの間に、低レベルのバックアップ保護から高レベル（例えば、NONBACKUPからBACKUP-DATAやBACKUP-DATA-AND-BLOB、或いはBACKUP-DATAからBACKUP-DATA-AND-BLOB）に変更しなければなりません。

➲ **例:**

次のコマンドを入力する:

```
dmSQL> BEGIN BACKUP;
dmSQL> SET DATA BACKUP ON;
dmSQL> END BACKUP DATAFILE;
dmSQL> END BACKUP JOURNAL;
```

➲ **例:**

次のコマンドを入力する:

```
dmSQL> BEGIN BACKUP;
dmSQL> END BACKUP DATAFILE;
dmSQL> SET DATA BACKUP ON;
dmSQL> END BACKUP JOURNAL;
```

まず先にNONBACKUPモードに変更しない限り、高レベルのバックアップ保護から低レベルへ移行することは認められていません。BACKUP-DATA-AND-BLOBモードからBACKUP-DATAモードに変更したい場合、まず先にNONBACKUPモードに変更してから、上記の低レベルのバックアップ保護から高レベルへ変更する手法を用います。BACKUP-DATA-AND-BLOBやBACKUP-DATAからは、いつでもNONBACKUPに変更することができます。つまり、完全バックアップの開始から終了までの間にする必要はありません。オンライン完全バックアップの実行についての詳細は、本章の「オンライン完全バックアップ」の節を参照して下さい。

## dmconfig.iniを使ったバックアップ・モードの設定

データベースがオフラインの場合は、dmconfig.iniファイルのキーワードDB_BModeを使って、直接バックアップ・モードを変更することができま

す。次回のデータベース起動時に、新しいバックアップ・モードが使用されます。データベースがオンラインの場合、DB_BModeキーワードの値を変更しても、データベースを終了し再起動するまでその変更は適用されません。NONBACKUPからBACKUP-DATAやBACKUP-DATA-AND-BLOBに変更、或いはBACKUP-DATAからBACKUP-DATA-AND-BLOBに変更する場合、必ずオフライン完全バックアップを実行して下さい。

➲ **dmconfig.iniファイルを使って、バックアップ・モードをセットする:**

1. **ASCII**テキスト・エディタを使って、データベース・サーバーでdmconfig.iniファイルを開きます。
2. **バックアップ・モード**を変更するデータベースの**環境設定セクション**を見つけます。
3. **DB_BMode**キーワードの値を次のいずれかに変更します。

```
0 - NONBACKUP mode
1 - BACKUP-DATA mode
2 - BACKUP-DATA-AND-BLOB mode
```

4. 新しいバックアップ・モードを有効にするために、データベースを再起動します。

バックアップ・モードを変更しようとしているデータベースの環境設定セクションに、DB_BModeキーワードが無い場合、新たにそれを追加します。データベースの環境設定セクションの1行目から、次の環境設定セクションまでのどこかに独立したキーワードの1行を追加します。キーワードの並びは関係ありません。DB_BModeに値を指定しない場合、初期設定値の0(NONBACKUPモード)が採用されます。

## dmSQLを使ったバックアップ・モードの設定

データベースがオンラインで、ユーザーがdmSQLを使い慣れている場合、SQLのSET文を使ってバックアップ・モードを変更することができます。このコマンドは、オンライン完全バックアップの間に実行します。新しいバックアップ・モードは、コマンドが実行されると有効になります。

➲ **dmSQLを使って、バックアップ・モードをセットする:**

1. dmSQLを使って、データベースに接続します。

2. **BEGIN BACKUP**文を使って、**オンライン完全バックアップ**を開始します。

3. 次のいずれかの**SET**コマンドを与えることによって、完全バックアップの途中にバックアップ・モードを変更します。

```
dmSQL> SET BACKUP OFF;
dmSQL> SET DATA BACKUP ON;
dmSQL> SET BLOB BACKUP ON;
```

4. **オンライン完全バックアップ**を終了します。

*注: SET BACKUP OFF文はNONBACKUPモード、SET DATA BACKUP ON文はBACKUP-DATAモード、SET BLOB BACKUP ON文は、BACKUP-DATA-AND-BLOBモードにそれぞれ対応します。*

# 9.5 オフライン完全バックアップ

オフライン完全バックアップを実行するには、オペレーティング・システムからデータベースのファイルを読み込む権限と、オペレーティング・システムからバックアップ・ディレクトリに書き込む権限が必要です。まずデータベースを終了しなければならない場合、ユーザーにはDBAかSYSADM権限が必要です。

バックアップ・モードに関わらず、オフライン完全バックアップを実行することができます。つまり、データベースはNON-BACKUP、BACKUP-DATA、BACKUP-DATA-AND-BLOBのいずれかのモードです。オフライン完全バックアップを使うと、データベースを終了した時点までデータベースをリストアすることができます。

## dmSQLを使ったオフライン完全バックアップ

➲ **dmSQLを使って、オフライン完全バックアップを実行する:**

1. 指定した時間にデータベースを終了する旨を全ユーザーに通知し、その前にデータベースから切断するように伝えます。

2. データベースが起動中の場合、**TERMINATE DB**文でデータベースを終了します。データベース終了の際にエラーが発生した場合は、データベースを再起動し、問題を解決してから再度終了します。
3. **dmconfig.ini**ファイルを調べ、ファイル・オブジェクト・ディレクトリを含め、バックアップするファイルとディレクトリを決定します。
4. オペレーティング・システムのコマンドやユーティリティを使って、データベース・ファイル(データファイル、ジャーナル・ファイル、ファイル・オブジェクト、**dmconfig.ini**ファイルを含む)をバックアップ・ディレクトリ又はバックアップ端末にコピーします。

## JServer Managerを使ったオフライン完全バックアップ

➲ **JServer Managerを使って、オフライン完全バックアップを実行する:**

1. データベース・サーバーで、**JServer Manager**アプリケーションを起動します。
2. メインコンソールの［**データベースのバックアップ**］をクリックします。バックアップの一覧が表示されます。
3. ［**オフライン完全バックアップ**］をクリックします。［**オフライン完全バックアップ**］のウィンドウが表示されます。
4. ［**データベース名**］からデータベースを選択します。ログイン・ウィンドウが表示されます。
5. ［**ユーザー**ID］の欄に、ユーザーIDを入力して下さい。

   *注: DBA権限を持つユーザーが、データベースのバックアップをすることができます。*
6. ［**パスワード**］の欄に、パスワードを入力して下さい。
7. ［OK］をクリックして下さい。データベースへのシングルユーザー接続が創設されます。バックアップするオペレーティング・システムのファイルの一覧を表示した［**オフライン完全バックアップ**］のウィンドウが現れます。

8. ブラウズ・ボタン(…)をクリックしてから、バックアップ・ディレクトリへの新規パスを選択します。
9. ［**OK**］をクリックして、バックアップ・ディレクトリに全ファイルを保存します。

   *注: そのファイルが既にバックアップ・ディレクトリに存在する場合、それらはデータベース管理者によって上書きされるかもしれません。*

10. **dmconfig.ini**ファイルで、ファイル・オブジェクトのディレクトリを調べます。
11. オペレーティング・システムのコマンドやユーティリティを使って、ファイル・オブジェクトをバックアップ・ディレクトリ又はバックアップ端末にコピーします。

## 9.6 バックアップ・サーバー

DBMasterは手動でデータベースのバックアップを取る種々のツールや方法を提供していますが、定期的にバックアップを取る方法も知っておく必要があります。如何に信頼できる人でも、ときには忘れることがあります。データベースをバックアップする知識によっては、データの安全が損なわれることがあります。これを解決するために、DBMasterは、バックアップ・サーバーを使用して完全に自動化したオンライン差分バックアップを取る便利な方法を提供します。

バックアップ・サーバーは、バックグラウンドで走行し、定期的なスケジュールで、またはジャーナルファイルがフルになったときに、あるいはこの両方でオンライン差分バックアップを取ります。バックアップ・サーバーは、データベース・サーバーと相互に通信して、いつバックアップを取るか、差分バックアップのタイプは何か、どのバックアップ・オプションを変更するかを柔軟に判断します。バックアップ・サーバーは、データベース・サーバーと同時に起動し、バックアップ・サーバーを停止させるか、データベースを終了するまで走行し続けます。

完全バックアップを実行する時、バックアップ・サーバーは、前回の完全バックアップをバックアップ・ディレクトリから古いディレクトリにコピーします。そして、ジャーナルファイルと**dmconfig.ini**を含むデータベースの全ファイルをバックアップ・ディレクトリにコピーし、以前の完全バックアップを上書きします。

差分バックアップを実行する時、バックアップ・サーバーは、必要なジャーナルファイルをバックアップ・ディレクトリにコピーします。ユーザーは、バックアップ・ファイル名のフォーマットを指定してファイル名をセットします。

バックアップ・サーバーには、種々の構成オプションがあります。構成オプションは、バックアップ・ファイル名フォーマット、バックアップ・ディレクトリの場所、バックアップを取るスケジュール、バックアップを取るときのジャーナルファイルの充満度、バックアップ・ファイルの保存方法等を指定します。

## バックアップ・サーバーを起動する

DBMasterは、dmconfig.iniファイルのDB_BkSvrキーワードの値が1の場合、バックアップ・サーバーを起動させます。DB_BkSvrの値が0の場合は、バックアップ・サーバーは起動しません。

DB_BkSvrキーワードを設定した後に、バックアップ・サーバーを明示的に起動させる必要はありません。DBMasterは、データベースを起動させると同時にバックアップ・サーバーも起動します。初期値は、バックアップ・サーバーを起動しません。

### dmconfig.iniを使ってバックアップ・サーバーを起動する

データベースがオフライン時に、dmconfig.iniファイルのDB_BkSvrキーワードで直接バックアップ・サーバーを起動させることができます。バックアップ・サーバーは、次にデータベースが起動する際に同時に起動します。データベースがオンラインのときは、DB_BkSvrキーワードの値を変更してもデータベースを再起動するまで有効にはなりません。

➲ **dmconfig.iniファイルでバックアップ・サーバーを起動させる:**

1. データベース・サーバーで、テキスト・エディタを使ってdmconfig.iniファイルを開きます。
2. バックアップ・サーバーを起動させるデータベース・セクションを見つけます。
3. DB_BkSvrキーワードを1にし、バックアップ・サーバーを起動可能にします。
4. データベースを再起動します。

## JServer Managerを使ってバックアップ・サーバーを起動する

データベースがオフラインのときは、JServer Managerツールを使用してバックアップ・サーバーを起動させることができます。JServer Managerは、dmconfig.iniファイルのDB_BkSvrキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースが起動した際に同時に起動します。データベースがオンラインのときにバックアップ・サーバーを動作可能にしても、データベースを再起動するまでは有効になりません。

➲ **オフライン時にJServer Managerでバックアップ・サーバーを起動させる:**

1. メイン画面の[データベースの起動]をクリックします。
2. **［データベース名］**から、修正するデータベース名を選びます。
3. **［設定］**ボタンをクリックします。**［データベース起動の高度な設定］**ウィンドウが表示されます。
4. **［バックアップ］**タブをクリックします。
5. **［バックアップ・サーバーを起動する］**チェックボックスをクリックします。
6. **［保存］**ボタンをクリックします。
7. **［取消］**ボタンをクリックして、**［データベースの起動］**ウィンドウに戻ります。

*注：　dmconfig.iniファイルのデータベース・セクションにDB_BkSvrキーワードが無いときは、JServer Managerが自動的に追加します。*

## 差分バックアップのファイル名形式を設定する

バックアップのファイル名形式は、バックアップ・ジャーナルファイルに付ける名前のフォーマットを指定します。バックアップ・ファイルに意味のある名前を付けることによって、データベースをリストアするときに、バックアップファイルを簡単に識別できるようになります。バックアップのファイル名形式は、テキスト文字とフォーマットシーケンス（エスケープシーケンス）の両方を含みます。

フォーマットシーケンスは、バックアップを取る年・月・日、データベース名、バックアップ識別番号を表すのに使用します。テキスト文字とフォーマットシーケンスは、オペレーティング・システムで使用できるファイル名である限り、どのように組み合わせてもかまいません。フォーマットシーケンスは、3つの部分：エスケープ文字、長さ、フォーマット文字から成ります。次のフォーマットシーケンスがあります。

**%[*n*]Y**—ジャーナルファイルがバックアップされた年

**%[*n*]M**—ジャーナルファイルがバックアップされた月

**%[*n*]D**—ジャーナルファイルがバックアップされた日

**%[*n*]B**—バックアップ識別番号

**%[*n*]N**—ジャーナルファイルが属するデータベース名

エスケープ文字は%記号で表し、フォーマットシーケンスの始めを示します。バックアップファイル名のテキスト文字に%記号を含めるときは、２つの%記号（即ち%%）にします。%記号の後には、数字またはY、M、D、B、Nのフォーマット文字が続きます。これ以外の文字が%記号の後にあると、DBMasterは不正なバックアップファイル名フォーマットとみなしエラーを返します。

長さは、フォーマットシーケンスが生成する文字列の長さを１～９で指定します。フォーマットシーケンスが指定した長さより短い文字列を生成するときは、０を埋めて指定した長さに合せます。データベース名は名前の

右側に 0 を埋め、その他は左側に 0 を埋めます。フォーマットシーケンスが指定した長さより長い文字列を生成するときは、切り捨てます。データベース名は右側を切り捨て、その他は左側を切り捨てます。長さを囲う[ ]は、長さの指定がオプションであることを示します。実際にフォーマットシーケンスを与えるときに、[ と ] を入力してはいけません。長さを指定しないときは、フォーマットシーケンスが生成する文字列全体を使用します。

フォーマット文字は、フォーマットシーケンスが生成する文字列のタイプを指定します。フォーマット文字は、Y、M、D、B、Nのいずれかでなければなりません；他の文字を使用すると、DBMasterは不正なバックアップファイル名フォーマットとみなしエラーを返します。正しいフォーマット文字であっても、エスケープ文字の直後あるいはエスケープ文字と数字の直後になければ、テキスト文字とみなされます。

年・月・日はシステムの日付から取られるので、システムの日付が合っていれば、年・月・日も正しくなります。バックアップ識別番号は、バックアップを取る一連のジャーナルファイルの順序番号です。DBMasterは、バックアップ・サーバーが各ジャーナルファイルをバックアップするごとに、自動的に識別番号を付けます。

バックアップファイル名フォーマットは、dmconfig.iniファイルのDB_BkFrmキーワードで指定します。バックアップファイル名フォーマットを指定しないときは、初期設定のフォーマット%2F%4N%4B.JNLを使用します。バックアップファイル名フォーマットが生成するファイル名の長さは、256文字以下でなければなりません。

バックアップファイル名フォーマットは、いろいろな方法で設定することができます。環境設定テキスト・ファイルを直接編集するか、或いはJServer Managerツールを使用するか、いずれかの使いやすい方法で行います。

## dmconfig.iniを使ったバックアップのファイル形式の設定

データベースがオフラインのときは、dmconfig.iniファイルのDB_BkFrmキーワードに直接バックアップのファイル名形式を設定することができます。バックアップ・サーバーは、次にデータベースが起動するときに、設

定したバックアップのファイル名形式を全てのバックアップ・ジャーナルファイルに適用します。データベースがオンラインのときにDB_BkFrmキーワードの値を変更しても、データベースを再起動しなければ有効にはなりません。

➲ **dmconfig.iniでバックアップのファイル名形式を設定する:**

1. データベース・サーバーのコンピュータ上でASCIIテキストエディタを使用してdmconfig.iniファイルを開きます。
2. バックアップのファイル名形式を設定するデータベースのセクションを見つけます。
3. DB_BkFrmキーワードにバックアップのファイル名形式の文字列を設定します。

*注：　文字列には任意のフォーマットシーケンスとテキスト文字を含めることができますが、結果として得られるファイル名は79字以下にしなければなりません。*

4. データベースを再起動すると、設定したバックアップのファイル名形式を使用開始します。

## JServer Managerを使ったバックアップのファイル名形式の設定

データベースがオンラインであるかオフラインであるかに関わらず、JServer Managerを使用してバックアップのファイル名形式を設定することができます。JServer Managerは、dmconfig.iniのDB_BkFrmキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースを起動したときに、このバックアップのファイル名形式を全てのバックアップ・ジャーナルファイルに適用します。

➲ **JServer Managerを使用してバックアップのファイル名形式を設定する:**

1. メイン画面、又は［**データベース**］メニューの［**データベースの起動**］をクリックします。
2. ［**データベース名**］で、修正するデータベース名を選びます。

3. ［設定］ボタンをクリックします。［データベース起動の高度な設定］ウィンドウが表示されます。
4. ［バックアップ］タブをクリックします。
5. ［バックアップ・サーバーを起動する］チェックボックスをクリックします。
6. ［バックアップ・ファイル・フォーマット］の欄に、バックアップ・ジャーナル・ファイルのフォーマットを入力します。
7. ［保存］ボタンをクリックします。
8. ［取消］ボタンをクリックして、［データベースの起動］ウィンドウに戻ります。

*注:* *データベース構成セクションにDB_BkFrmキーワードが無いときは、JServer Managerが自動的に追加します。*

## バックアップ・ディレクトリを設定する

バックアップ・ディレクトリは、バックアップ・ジャーナルファイルを何処に置くかを指定します。バックアップ・ディレクトリは、既に存在するディレクトリでなければなりません。バックアップ・ディレクトリが無いときは、事前にオペレーティング・システムを使用して作成しておきます。バックアップ・ディレクトリを、データベースとは別のディスクに作成すると、メディア障害が発生した際に、データベース・ファイルとバックアップファイルの両方が失われてしまうのを防ぐことができます。

バックアップ・ディレクトリは、dmconfig.iniファイルのDB_BkDirキーワードで定義します。DB_BkDirキーワードには、バックアップ・ディレクトリの絶対パスまたは相対パスを指定します。バックアップ・ディレクトリを指定しない場合は、初期設定の「backup」ディレクトリがデータベース・ディレクトリの下に作成されます。（データベース・ディレクトリは、dmconfig.iniファイルのDB_DbDirキーワードで指定します。）バックアップ・ディレクトリのパス名は、255字以内でなければなりません。

複数のデータベースが同じディレクトリにある場合は、初期設定のバックアップ・ディレクトリを利用しないようにします。この場合、データベースのバックアップ履歴情報が別のデータベースによって上書きされて、一方または双方のバックアップが使用不能の状態になります。このような問題をさけるためには、データベースを別々のデータベース・ディレクトリに入れるという方法と、各データベースのバックアップ・ディレクトリを別々に指定するという方法がありますが、前者の方法がより望ましいです。こうすることで、各ファイルがどのデータベースに属するかを正確に知ることができるようになります。

バックアップ・ディレクトリは、いろいろな方法で設定することができます。dmconfig.ini環境設定ファイルを編集する方法と、JServer Managerグラフィカルツールを使用する方法のいずれかの使い易い方法を採用して下さい。

## dmconfig.iniを使ったバックアップ・ディレクトリの設定

データベースのオフライン時に、dmconfig.iniファイルのDB_BkDirキーワードに直接バックアップ・ディレクトリを指定することができます。次にデータベースが起動するときに、バックアップ・サーバーが設定したディレクトリをバックアップ・ディレクトリとして使用します。データベースのオンライン時に、DB_BkDirキーワードの値を変更しても、データベースを再起動するまで、その変更は有効にはなりません。

➲ **dmconfig.iniファイルでバックアップ・ディレクトリを設定する:**

1. データベース・サーバーで、テキスト・エディタを使ってdmconfig.iniファイルを開きます。
2. バックアップ・ディレクトリを設定するデータベースの構成セクションを見つけます。
3. DB_BkDirキーワードに既存のバックアップ・ディレクトリ名の文字列を指定します。
4. データベースを再起動すると、設定したバックアップ・ディレクトリが有効になります。

## dmSQLを使ったバックアップ・ディレクトリの設定

SetSystemOption文を使用すると、データベースの起動中のみバックアップ・ディレクトリを変更することができます。典型的な構文は、次のとおりです。

```
CALL SetSystemOption('bkdir', 'path')
```

*path* には、新しいバックアップ・ディレクトリを指定します。指定する文字列のサイズは、255文字以下です。

➲ **例**

dmSQLコマンド・プロンプトに次のように入力し、ディレクトリのパスをE:/storage/database/backup/WebDBに変更します。

```
dmSQL> CALL SetSystemOption('bkdir', 'E:/storage/database/backup/WebDB');
```

## JServer Managerを使ったバックアップ・ディレクトリの設定

データベースがオフラインのときは、JServer Managerツールを使用してバックアップ・ディレクトリを設定することができます。JServer Managerは、dmconfig.iniのDB_BkDirキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースが起動したときに、このディレクトリをバックアップ・ディレクトリとして使用します。データベースがオンラインのときは、バックアップ・ディレクトリを直ちに変更することもできますが、次にデータベースを再起動するときまで遅延することもできます。いずれの場合も、JServer Managerは、新しいバックアップ・ディレクトリにも、バックアップ履歴ファイルのコピーを作成します。

➲ **オフライン時にJServer Managerでバックアップ・ディレクトリを設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。

4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。
6. ［**バックアップ・ファイルのディレクトリ**］の欄にパスを入力、或いはブラウザ・ボタン  でパスを選択します。
7. ［**保存**］ボタンをクリックします。
8. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

➲ **オンライン時にJServer Managerでバックアップ・ディレクトリを設定する:**

1. ［**データベース**］メニューの［**ランタイムの設定**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**ユーザー名**］と［**パスワード**］欄に入力し、［**OK**］をクリックします。
4. バックアップ設定の［**ディレクトリ**］の欄にパスを入力、或いはブラウザ・ボタン  でパスを選択します。
5. ［**OK**］をクリックします。

*注:* *dmconfig.iniファイルにデータベース・セクションにDB_BkDirキーワードが無いときは、JServer Managerが自動的に追加します。*

## 古いディレクトリを設定する

古いディレクトリは、バックアップ・サーバーが以前の完全バックアップ・ファイルを配置する場所を指定します。メディア障害の際にデータベースとバックアップ・ファイルの両方を消失することが無いように、データベースとは別のディスクを選択する必要があります。

dmconfig.iniファイルのキーワードDB_BkOdrで、古いディレクトリを設定します。指定しない場合、バックアップ・サーバーは以前の完全バックアップ・ファイルをバックアップしません。

## dmconfig.iniを使って古いディレクトリをセットする

dmconfig.iniファイルのキーワードDB_BkOdrに、バックアップ・サーバーが使用する古いディレクトリを直接セットすることができます。次回データベースを起動する際、バックアップ・サーバーは古いディレクトリとしてこのディレクトリを使用します。データベースがオンラインであれば、データベースを再起動するまで、キーワードDB_BkOdrの値の変更は無効です。

## JServer Managerを使った古いディレクトリの設定

データベースがオフラインの場合、JServer Managerのグラフィカル・ユーティリティを使って、以前のバックアップのためのディレクトリをセットすることができます。JServer Managerは、dmconfig.iniファイルの**DB_BkOdr**キーワードの値を自動的に変更します。次にデータベースを起動する際、バックアップ・サーバーはバックアップ・ディレクトリ同様、このディレクトリを使用します。データベースがオンラインの場合、JServer Managerは直ちに古いバックアップ・ディレクトリを変更するか、或いはデータベースを再起動する時までに変更を遅らせることができます。

➲ **JServer Managerを使って、オフライン時に古いバックアップ・ディレクトリをセットする:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。

5. ［以前の完全バックアップのディレクトリ］の欄にパスを入力、或いはブラウザ・ボタン  でパスを選択します。
6. ［保存］ボタンをクリックします。
7. ［取消］ボタンをクリックして、［データベースの起動］ウィンドウに戻ります。

*注:* *dmconfig.iniファイルにデータベース・セクションにDB_BkOdrキーワードが無いときは、JServer Managerが自動的に追加します。*

## 差分バックアップ設定

差分バックアップ・スケジュールは、バックアップ・サーバーがオンライン差分バックアップを実行するタイミングを指定します。スケジュールは、バックアップ開始日時と、それ以降にバックアップが実行される時間間隔で構成されています。バックアップ開始日時は、バックアップ・サーバーが最初の差分バックアップを実行する日付と時刻を表しています。時間間隔は、次回の差分バックアップまでの時間を表しています。

定期スケジュールに加え、後述するジャーナル・トリガー値も合わせて使用することが可能です。ジャーナル・トリガー値は、ジャーナルファイルが指定した割合に達した時に、データベースのバックアップを行います。これら2種類のバックアップを組み合わせることができます。定期スケジュールを定義しない場合、バックアップ・サーバーは、特定のタイミングでデータベースをバックアップすることはありません。但し、ジャーナル・ファイルが一杯になった場合には、差分バックアップを実行します。

バックアップ開始日時は、dmconfig.iniファイルのキーワードDB_BkTimで指定します。キーワードDB_BkTimに、yy/mm/dd hh:mm:ss形式で日付と時刻を入力して下さい。初期設定値はありません。但し、バックアップ・サーバーを使用するためにJServer Managerを使用する場合、JServer Managerは初期設定値を設け、この値をdmconfig.iniファイルに書き込みます。

時間間隔は、dmconfig.iniファイルのキーワードDB_BkItvで指定します。キーワードDB_BkItvに、d-hh:mm:ss形式で時間間隔を入力して下さい。初期設定値はありません。但し、バックアップ・サーバーを使用するために

JServer Managerを使用する場合、JServer Managerは初期設定値1-00:00:00を設定し、この値をファイルに書き込みます。

DBMasterには、差分バックアップ・スケジュールを設定する方法がいくつかあります。dmconfig.ini環境設定ファイルを編集する方法と、JServer Managerグラフィカルツールを使用する方法のいずれかの使い易い方法を採用して下さい。

## dmconfig.iniを使った差分バックアップの設定

データベースがオフラインの場合、dmconfig.iniファイルのキーワードDB_BkTimとDB_BkItvを使って直接バックアップ・サーバーが利用するスケジュールを設定することができます。次回データベースを起動する時、バックアップ・サーバーは、差分バックアップ・スケジュールにこの設定を使います。データベースがオンラインの場合、データベースを再起動するまで、キーワードDB_BkTimとDB_BkItvの値の変更は無効です。

➲ **dmconfig.iniファイルを使ってバックアップ・スケジュールを設定する:**

1. データベース・サーバーで、テキスト・エディタを使ってdmconfig.iniファイルを開きます。
2. バックアップ・スケジュールを設定するデータベースのセクションを見つけます。
3. **yy/mm/dd hh:mm:ss**形式を使って、日付と時刻のキーワード**DB_BkTim**の値を指定します。
4. **d-hh:mm:ss**形式を使って、時間間隔のキーワード**DB_BkItv**の値を指定します。
5. データベースを再起動します。

## dmSQLを使った差分バックアップ設定の変更

SetSystemOption文を使用すると、データベースの起動中に差分バックアップの起動時刻と起動間隔を変更することができます。差分バックアップの起動時刻を変更する典型的な構文は、次のとおりです。

```
CALL SetSystemOption('bktim', 'StartTime')
```

差分バックアップの起動間隔を変更する典型的な構文は、次のとおりです。

```
CALL SetSystemOption('bkitv', 'Interval')
```

*StartTime* は、差分バックアップが最初に起動する時刻を意味し、そのフォーマットはYY:MM:DD HH:MM:SSです。*Interval* は、差分バックアップが作動する時間間隔を意味し、そのフォーマットはD-HH:MM:SSです。

➲ **例**

dmSQLコマンド・プロンプトに次のように入力し、差分バックアップの間隔を1時間に設定します。

```
dmSQL> CALL SetSystemOption('bkitv', '0-1:00:00');
```

## JServer Managerを使った差分バックアップの設定

データベースがオフラインの場合、JServer Managerのグラフィカル・ユーティリティを使ってバックアップが使用する差分バックアップ・スケジュールを設定することができます。JServer Managerは、自動的にdmconfig.iniファイルのキーワードDB_BkTimとDB_BkItvの値を変更します。次回データベースを起動する時、バックアップ・サーバーは、差分バックアップ・スケジュールにこの設定を使います。データベースがオンラインの場合、JServer Managerは直ちにバックアップ・スケジュールを変更、若しくは次回データベースを再起動する際に変更します。

➲ **オフライン時に、JServer Managerでバックアップ・スケジュールを設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。

6. ［**差分バックアップの開始日時**］の欄に、日付と時刻を入力します。
7. ［**差分バックアップ実行時間の間隔**］の欄に、日数と時間を入力します。
8. ［**保存**］ボタンをクリックします。
9. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

*注:* *dmconfig.iniファイルのデータベース・セクションにキーワードDB_BkTimとDB_BkItvが存在しない場合、JServer Managerが自動的にそれらを追加します。*

➲ **オンライン時に、JServer Managerでバックアップ・スケジュールを設定する:**

1. ［**データベース**］メニューの［**ランタイムの設定**］をクリックします。
2. ［**データベース名**］から、修正するデータベース名を選びます。
3. ［**ユーザー名**］と［**パスワード**］欄に入力し、［**OK**］をクリックします。
4. 更新した変更を次回以降にも適用する場合は、バックアップ設定の［**dmconfig.iniに保存**］の欄をチェックします。
5. バックアップ設定の［**開始日時**］の欄に、日付と時刻を入力します。
6. バックアップ設定の［**間隔**］の欄に、日数と時間を入力します。
7. ［**OK**］をクリックします。

## ジャーナル・トリガー値を設定する

ジャーナル・トリガー値は、ジャーナルファイルが指定した割合まで書き込まれたときに、バックアップ・サーバーがオンライン差分バックアップを取るジャーナルファイルの割合(%)の値です。ジャーナル・トリガー値

とバックアップ・スケジュールを組み合わせて、ジャーナルファイルが指定パーセントに達した時点に加え、定期的にデータベースをバックアップすることができます。

ジャーナル・トリガー値は、dmconfig.iniファイルのDB_BkFulキーワードで指定します。DB_BkFulキーワードの値は、50～100の範囲内の整数値または0です。50～100は、バックアップの実行を引き起すジャーナルファイルに書き込まれた割合(%)を表します。0はジャーナルファイルが完全に一杯になったときにバックアップを取ります。0と100は実質的に同じです。どちらもジャーナルファイルが完全に一杯（100%）になったときにバックアップを取ります。ジャーナル・トリガー値を指定しない場合、バックアップ・サーバーは初期値0を採用します。

ジャーナル・トリガー値は、いろいろな方法で設定することができます。dmconfig.ini環境設定ファイルを編集する方法と、JServer Managerグラフィカルツールを使用する方法のいずれかの使い易い方法を採用して下さい。

## dmconfig.iniを使ったジャーナル・トリガー値の設定

データベースがオフラインのときは、**dmconfig.ini**ファイルのDB_BkFulキーワードに直接ジャーナル・トリガー値を設定することができます。バックアップ・サーバーは、次にデータベースが起動するときに、設定したジャーナル・トリガー値を使用します。データベースがオンラインのときにDB_BkFulキーワードの値を変更しても、データベースを再起動するまで有効にはなりません。

➲ **dmconfig.iniファイルでジャーナルトリガー値を設定する:**

1. データベース・サーバーで、テキスト・エディタを使って**dmconfig.ini**ファイルを開きます。
2. ジャーナル・トリガー値を設定するデータベースのセクションを見つけます。
3. DB_BkFulキーワードの値を、50～100の範囲内の整数値、又は0にします。
4. データベースを再起動します。

## DMSQLを使ったジャーナル・トリガー値の変更

SetSystemOption文を使用すると、データベースの起動中のみジャーナル・トリガー値を変更することができます。典型的な構文は、次のとおりです。

```
CALL SetSystemOption('bkful', 'n')
```

*n* には、0或いは50～100の間の数値を指定します。*n* を0に設定すると、ジャーナル・ファイルが一杯になった時にバックアップ・サーバーが作動します。*n* を50～100の間の数値に指定すると、ジャーナル・ファイルの割合が指定した値に達した時に、バックアップ・サーバーが作動します。

➲ **例**

dmSQLコマンド・プロンプトに次のように入力し、ジャーナル・トリガー値を75%に設定します。

```
dmSQL> SetSystemOption('bkful', '75');
```

## JServer Managerを使ったジャーナル・トリガー値の設定

データベースがオフラインのときは、JServer Managerの画面ツールを使用してジャーナルトリガー値を設定することができます。JServer Managerは、dmconfig.iniファイルのDB_BkFulキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースが起動したときに、この設定を新しいジャーナルトリガー値として使用します。データベースがオンラインのときは、直ちにジャーナルトリガー値を変更することもできますが、データベースを再起動するときまで延すこともできます。

➲ **オフライン時にJServer Managerでジャーナル・トリガー値を設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。

5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。
6. ジャーナル・ファイルが特定の割合に達した時に、差分バックアップを自動的に実行させるために、次のいずれかを選択します。
   - ジャーナル・ファイルが一杯になった時に差分バックアップを実行させる場合、［**差分バックアップの起動タイミング**］の［**ジャーナルファイルが100%に達した時**］をクリックします。
   - ジャーナル・ファイルが指定した割合に達した時に差分バックアップを実行させる場合、［**%に達した時**］の欄に50～100の範囲内の値を入力します。
7. ［**保存**］ボタンをクリックします。
8. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

*注：　dmconfig.iniファイルのデータベース・セクションにDB_BkFulキーワードが無いときは、JServer Managerが自動的に追加します。*

➲ **オンライン中にJServer Managerでジャーナルトリガー値を設定する:**

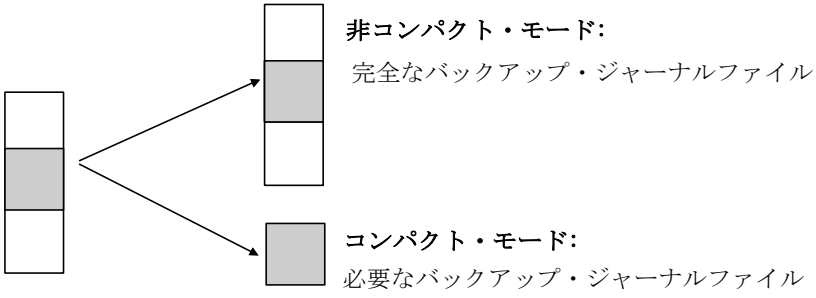
1. ［**データベース**］メニューの［**ランタイムの設定**］をクリックします。
2. ［**データベース名**］で、設定するデータベース名を選びます。
3. ［**ユーザー名**］と［**パスワード**］欄に入力し、［**OK**］をクリックします。
4. 更新した変更を次回以降にも適用する場合は、バックアップ設定の［**dmconfig.iniに保存**］の欄をチェックします。
5. ジャーナル・ファイルが特定の割合に達した時に、自動的に差分バックアップを実行させるために、次のいずれかを選択して設定します。

- ジャーナル・ファイルが一杯になった時に差分バックアップを実行させる場合、［**ジャーナルパーセンテージ**］の［**初期設定を使う**］を選択します。
- ジャーナル・ファイルが指定した割合に達した時に差分バックアップを実行させる場合、［**ジャーナルパーセンテージ**］の［**50-100**］を選択し、範囲内の値を入力します。

6. ［**OK**］をクリックします。

## コンパクト・バックアップ・モードを設定する

コンパクト・バックアップ・モードは、バックアップ・サーバーがオンライン差分バックアップを取るときに、ジャーナルファイル全体をバックアップせずに、フルジャーナルブロックだけをバックアップします。コンパクト・バックアップは、データベースのリストアに必要の無いジャーナルブロックは無視し、必要なジャーナルブロックだけをバックアップします。コンパクト・バックアップ・モードを使用すると、バックアップ領域を節約しますが、データベースのリストアにはより多くの時間がかかります。




コンパクト・バックアップ・モードは、dmconfig.iniファイルのDB_BkCmpキーワードで指定します。DB_BkCmpキーワードの値は0或いは1です。1はコンパクト・バックアップ・モードを有効にし、0は無効にします。このキーワードの値を指定しない場合は、初期値1（有効）を使用します。

コンパクト・バックアップ・モードは、いろいろな方法で設定することができます。dmconfig.ini環境設定ファイルを編集する方法と、JServer

Managerグラフィカルツールを使用する方法のいずれかの使い易い方法を採用して下さい。

## dmconfig.iniを使ったコンパクト・バックアップモードの設定

データベースがオフラインのときは、dmconfig.iniファイルのDB_BkCmpキーワードで直接コンパクト・バックアップ・モードを設定することができます。バックアップ・サーバーは、次にデータベースが起動するときに、設定したコンパクト・バックアップ・モードを使用します。データベースがオンラインのときにDB_BkCmpキーワードの値を変更しても、データベースを再起動するまで有効にはなりません。

➲ **dmconfig.iniファイルでコンパクト・バックアップ・モードを設定する:**

1. データベース・サーバーで、テキスト・エディタを使ってdmconfig.iniファイルを開きます。
2. コンパクト・バックアップ・モードを設定するデータベース・セクションを見つけます。
3. コンパクト・バックアップ・モードを有効にする場合はDB_BkCmpキーワードの値を1に、無効にする場合は0にします。
4. データベースを再起動します。

## JServer Managerを使ったコンパクト・バックアップ・モードの設定

データベースがオフラインのときは、JServer Manager画面ツールを使用してコンパクト・バックアップ・モードを設定することができます。JServer Managerは、**dmconfig.ini**ファイルのDB_BkCmpキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースが起動したときに、この設定を新しいコンパクト・バックアップ・モードとして使用します。データベースがオンラインのときは、直ちにコンパクトバックアップモードを変更することもできますが、データベースを再起動するときまで延すこともできます。

➲ **オフライン時にJServer Managerでコンパクト・バックアップ・モードを設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。
6. ［**コンパクト・バックアップを有効にする**］チェックボックスをクリックします。
7. ［**保存**］ボタンをクリックします。
8. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

*注:* *dmconfig.iniファイルのデータベース・セクションにDB_BkCmpキーワードが無いときは、JServer Managerが自動的に追加します。*

➲ **オンライン時にJServer Managerでコンパクト・バックアップ・モードを設定する:**

1. ［**データベース**］メニューの［**ランタイムの設定**］をクリックします。
2. ［**データベース名**］から、修正するデータベース名を選びます。
3. ［**ユーザー名**］と［**パスワード**］欄に入力し、［**OK**］をクリックします。
4. 更新した変更を次回以降にも適用する場合は、バックアップ設定の［**dmconfig.iniに保存**］の欄をチェックします。
5. バックアップ設定の［**コンパクト・バックアップ・モード**］のチェックボックスをクリックします。

6. ［**OK**］をクリックします。

## 完全バックアップ・スケジュール

完全バックアップ・スケジュールは、バックアップ・サーバーがオンライン完全バックアップを実行する日時を指定します。そのスケジュールは、開始日時と時間間隔の2つの部分で構成されています。バックアップ開始日時は、バックアップ・サーバーが最初の完全バックアップを実行する日付と時刻を決定します。時間間隔は、それ以降の完全バックアップの間隔を決定します。

データベースのバックアップに、差分と完全バックアップ・スケジュールを組み合わせることができます。完全バックアップ・スケジュールを指定しない場合、バックアップ・サーバーは定期的に完全バックアップを行いません。

バックアップ開始日時は、**dmconfig.ini**ファイルのキーワードDB_FBkTmで指定します。キーワードDB_FBkTmに日付と時刻をyy/mm/dd hh:mm:ss形式で入力して下さい。初期設定値はありません。

時間間隔は、**dmconfig.ini**ファイルのキーワードDB_FBkTvで指定します。キーワードDB_FBKTVに時間間隔をd-hh:mm:ssフォーマットで入力して下さい。初期設定値はありません。

### dmconfig.iniを使った完全バックアップ・スケジュールの設定

データベースがオフラインの場合、**dmconfig.ini**ファイルのキーワードDB_FBkTmとDB_FBkTvに直接、完全バックアップ・スケジュールをセットすることができます。次回データベースを起動する際、バックアップ・サーバーは完全バックアップ・スケジュールにこれらの設定を使用します。データベースがオンラインの場合、キーワードDB_FBkTmとDB_FBkTvの変更はデータベースを再起動するまで反映されません。

➲ **dmconfig.iniファイルを使ってバックアップ・スケジュールを設定する:**

1. データベース・サーバーで、テキスト・エディタを使って**dmconfig.ini**ファイルを開きます。
2. バックアップ・スケジュールを設定するデータベース・セクションを見つけます。
3. **yy/mm/dd hh:mm:ss**形式を使って、日付と時刻のキーワード**DB_FBkTm**の値を指定します。
4. **d-hh:mm:ss**形式を使って、時間間隔のキーワード**DB_FBkTv**の値を指定します。
5. データベースを再起動します。

## JServer Managerを使った完全バックアップ・スケジュールの設定

データベースがオフライン時に、JServer Managerを使って完全バックアップのスケジュールを設定することができます。JServer Managerは自動的に**dmconfig.ini**ファイルの**DB_FBkTm**と**DB_FBkTv**キーワードの値を修正します。次にデータベースを起動する時、新しい完全バックアップ・スケジュールが適用されます。

➲ **オフライン時にJServer Managerで完全バックアップ・スケジュールを設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。

6. ［**完全バックアップの開始日時**］の欄に、日付と時刻を入力します。
7. ［**完全バックアップ・デーモンの間隔**］の欄に、日数と時間を入力します。
8. ［**保存**］ボタンをクリックします。
9. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

*注：　dmconfig.iniファイルのデータベース・セクションにDB_FBkTmとDB_FBkTvキーワードが無いときは、JServer Managerが自動的に追加します。*

## ファイルオブジェクトのバックアップ・モード

ファイルオブジェクト・バックアップ・モードは、完全バックアップの際にファイルオブジェクトをバックアップさせるかどうかを決定します。システム・ファイルオブジェクトのみをバックアップ、又はシステム・ファイルオブジェクトとユーザー・ファイルオブジェクト両方をバックアップのいずれかを選択することもできます。

ファイルオブジェクト・バックアップ・モードを設定する方法はいくつかあります。データベースの起動時に環境設定ファイルのキーワード**DB_BkFoM**が参照されますが、ランタイム時にdmSQLやJServer Managerを使ってその設定を変更することもできます。

バックアップ・サーバーは、以前のバックアップを**DB_BkOdr**で指定した古いディレクトリに移動します。

ファイルオブジェクトのバックアップを使用可能にすると、完全バックアップにより時間がかかります。完全バックアップ全体のコストには、(1)**DB_BkOdr**にセットしている場合、以前の完全バックアップのコピー; (2)全データベース・ファイルのコピー; (3)全ジャーナル・ファイルのコピー;(4)**DB_BkFoM**にセットしている場合、全ファイルオブジェクトのコピーが含まれます。また、バックアップ・エラーを防ぐために、**DB_BkDir**(必要な場合は**DB_BkOdr**も)で指定したバックアップ・ディレク

トリに全バックアップ・ファイルのために十分なスペースがあることを確認して下さい。

ファイルオブジェクトは、完全バックアップが実行された時にバックアップ・ディレクトリに生成されたFOディレクトリにコピーされます。ファイルオブジェクトは、バックアップ・ファイルオブジェクト・ディレクトリにコピーされる際に、順番に名前が付けられます。 The files in *fo* サブディレクトリにあるファイルの名前は、FOで始まり10桁の通し番号が付きます。バックアップしたファイル・オブジェクトには、全て拡張子.BAKが付きます。元のファイル名とパスとバックアップしたファイル名の間のマッピングは、ファイルオブジェクトのマッピング・ファイル***dmFoMap.his***に記録されます。

## バックアップされたファイルオブジェクトのマッピング・ファイル

ファイルオブジェクトのマッピングファイル*dmFoMap.his* は、 *「DB_BkDir/FO」* ディレクトリに生成されます。このファイルは、単なるテキストファイルです。外部ファイル名とバックアップされたファイル名を記録されています。その形式は以下のとおりです。

```
Database Name: MYDB
Begin Backup FO Time: 2001.5.13 2:33
FO Backup Directory: /DBMaster/mydb/backup/FO (i.e. DB_BkDir/FO)
[Mapping List]
s, fo00000000000.bak, "/DBMaster/mydb/fo/ZZ000001.bmp"
u, fo00000000001.bak, "/home2/data/image.jpg"
....
s, fo00000002345.bak, "/DBMaster/mydb/fo/ZZ00AB32.txt"
```

[Mapping List]の前の内容は、ユーザーの参照のための説明です。[Mapping List]の後ろの各行は、ファイルオブジェクトの種類(s = システム・ファイルオブジェクト、u = ユーザー・ファイルオブジェクト)、*FO*サブディレクトリにある新しいファイル名、その元のファイル名とパスを表します。このマッピングファイルは、ファイルオブジェクトのリストア時に必要になります。

## dmconfig.iniを使ったファイルオブジェクト・バックアップ・モードの設定

dmconfig.iniのキーワードDB_BkFoMは、ファイルオブジェクトのバックアップ・モードを指定します。

- **DB_BkFoM** = 0: ファイルオブジェクトをバックアップしない。
- **DB_BkFoM** = 1: システム・ファイルオブジェクトのみバックアップする。
- **DB_BkFoM** = 2: システム・ファイルオブジェクトとユーザー・ファイルオブジェクト双方ともバックアップする。

ファイルオブジェクトをバックアップする時(**DB_BkFoM** = 1、2)、バックアップ・サーバーは、ファイルオブジェクトの全外部ファイルを**DB_BkDir**キーワードで指定したディレクトリのサブディレクトリの“*fo*”にコピーします。スケジュールは、**DB_FBkTm**と**DB_FBkTv**で指定した完全バックアップのスケジュールに基づきます。

➲ **例:**

関連キーワードを含むdmconfig.iniファイルからの引用:

```
[MyDB]
DB_BkSvr = 1                          ; バックアップ・サーバーを起動させる
DB_FBKTm = 01/05/01 00:00:00          ; 2001年5月1日の深夜に開始
DB_FBKTV = 1-00:00:00                 ; 1日間隔
DB_BkDir = /home/DBMaster/backup      ; バックアップ・ディレクトリ
DB_BkFoM = 2                          ; システムFOとユーザーFO両方をバックアップする
```

ファイル・オブジェクトのバックアップ・モードが2なので、バックアップ・サーバーはデータベースの全外部ファイルオブジェクトを、“**/home/DBMaster/backup/FO**”ディレクトリにコピーします。**FO**サブディレクトリが存在しない場合、バックアップ・サーバーはそれを生成します。

## dmSQLを使ったファイルオブジェクト・バックアップ・モードの設定

SetSystemOption文を使用すると、データベースの起動中のみファイルオブジェクト・バックアップ・モードを変更することができます。ファイルオブジェクト・バックアップ・モードを変更する典型的な構文は、次のとおりです。

```
CALL SetSystemOption('bkfom', 'n')
```

*n* には、0又は1、或いは2を指定します。*n* を0に設定すると、ファイルオブジェクト・バックアップ・モードはOFFになります。*n* を1に設定すると、完全バックアップの際にシステム・ファイルオブジェクトを全てバックアップします。*n* を2に設定すると、完全バックアップの際にシステム・ファイルオブジェクトとユーザー・ファイルオブジェクトを全てバックアップします。

➲ **例:**

dmSQLコマンド・プロンプトに次のように入力し、システム・ファイルオブジェクトとユーザー・ファイルオブジェクトを全てバックアップするように、バックアップ・サーバーを設定します。

```
dmSQL> CALL SetSystemOption('bkfom', '2');
```

## JServer Managerを使ったファイルオブジェクト・バックアップ・モードの設定

データベースがオフライン時に、或いはランタイム時にJServer Managerを使ってファイルオブジェクト・バックアップ・モードを設定することができます。次にデータベースを起動する際に、新しい完全バックアップ・スケジュールが適用されます。

➲ **データベースの起動時にファイルオブジェクトのバックアップ・モードを設定する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。

3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスをクリックします。
6. バックアップ・サーバーで完全バックアップを実行させます。
   a) バックアップ・ディレクトリの位置を表示するために、パスを入力、又は［**バックアップ・ファイルのディレクトリ**］わきのブラウズ・ボタン  で選択して下さい。
   b) ［**完全バックアップ開始日時**］の欄に、日付と時間を入力して下さい。表示して下さい。
   c) ［**完全バックアップ・デーモン**］間隔の欄に、実行したい完全バックアップ間隔の日数、時間、分、秒を入力して下さい。
7. バックアップ処理の際にどの種類のファイルオブジェクトをバックアップさせるかを選択します。
   a) ファイルオブジェクトをバックアップさせない場合は、［**ファイルオブジェクトをバックアップしない**］を選択します。
   b) システム・ファイルオブジェクトのみをバックアップさせる場合は、［**システム・ファイルオブジェクトのみバックアップ**］を選択します。
   c) 全てのファイルオブジェクトをバックアップさせる場合は、［**システムとユーザーファイルオブジェクトをバックアップ**］を選択します。
8. ［**保存**］ボタンをクリックします。
9. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

## バックアップ・サーバーを停止する

バックアップ・サーバーを起動させる必要が無いときは、DB_BkSvrキーワードの値を0にして明示的に停止する必要があります。バックアップ・サーバーは、データベースを再起動するまでは走行し続けます。

### dmconfig.iniを使ってバックアップ・サーバーを停止する

データベースがオフラインのときは、dmconfig.iniファイルのDB_BkSvrキーワードを使用して直接バックアップ・サーバーを停止することができます。バックアップ・サーバーは、次にデータベースが起動するときには起動しません。データベースがオンラインのときは、DB_BkSvrキーワードの値を変更してもデータベースを再起動するまで有効にはなりません。

➲ **dmconfig.iniファイルでバックアップ・サーバーを停止する:**

1. データベース・サーバーで、テキスト・エディタを使ってdmconfig.iniファイルを開きます。
2. バックアップ・サーバーを停止するデータベース・セクションを見つけます。
3. DB_BkSvrキーワードの値を0にして、バックアップ・サーバーを停止させます。
4. データベースを再起動します。

### JServer Managerを使ってバックアップ・サーバーを停止する

データベースがオフラインのときは、JServer Managerツールを使用してバックアップ・サーバーを停止することができます。JServer Managerは、dmconfig.iniのDB_BkSvrキーワードの値を自動的に変更します。バックアップ・サーバーは、次にデータベースが起動したときには起動しません。データベースがオンラインのときは、バックアップ・サーバーをOFFにしても、データベースを再起動するまでその変更は有効になりません。

➲ **オフライン時にJServer Managerでバックアップ・サーバーを停止する:**

1. メイン画面の［**データベースの起動**］をクリックします。
2. ［**データベース名**］から、設定するデータベース名を選びます。
3. ［**設定**］ボタンをクリックします。［**データベース起動の高度な設定**］ウィンドウが表示されます。
4. ［**バックアップ**］タブをクリックします。
5. ［**バックアップ・サーバーを起動する**］チェックボックスを空白にします。
6. ［**保存**］ボタンをクリックします。
7. ［**取消**］ボタンをクリックして、［**データベースの起動**］ウィンドウに戻ります。

## 9.7 バックアップ履歴ファイル

手動の差分バックアップの場合、どのジャーナル・ファイルが、いつバックアップされ、どこにそのバックアップ・ファイルが配置されたのかを記録しておく必要があります。不注意でこの情報を忘れたり、紛失したりすることがあります。バックアップ・サーバーを使った自動バックアップは、この情報をバックアップ履歴ファイルに自動的に保存することで、このようなミスを防ぎます。

差分バックアップを手動で実行する場合、どのジャーナル・ファイルが、いつバックアップされ、どこにバックアップ・ファイルが配置されたのかを記録しておく必要があります。一方、バックアップ・サーバーを起動する場合、この情報は全てバックアップ履歴ファイルに保管されます。

### バックアップ履歴ファイルの割り当て

バックアップ・パスに生成されるdmBackup.hisファイルが、データベースのリストアの際に自動的に使用されます。

## バックアップ履歴ファイルの要素

バックアップ履歴ファイルは、複数の行で構成されています。

➲ **例:**

```
<backup_id>: journal_file_name -> archive_file_name: time: event
```

これは、*journal_file_name*という名前のジャーナル・ファイルが、*event*のイベントのために、*time*の日時に、*archive_file_name*という名前のアーカイブ・ファイルにコピーされたことを示します。*event*は、バックアップの原因を意味し、テキスト文字列で表現されます。この文字列は、JOURNAL-FULL又はTIME-OUTのいずれかです。JOURNAL-FULLは、ジャーナル・ファイルが一杯になったために差分バックアップが実行されたことを意味します。TIME-OUTは、バックアップのスケジュールの時間になったので、バックアップが実行されたことを意味します。

## バックアップ履歴ファイルの使用

ジャーナルが一杯になるような状況が頻繁に発生する場合、バックアップのジャーナルの充填度を下げるか、時間間隔を短くする必要があります。バックアップ履歴ファイルをチェックして、バックアップの時間間隔が短すぎないかを確認することもできます。同じジャーナル・ファイルが連続してバックアップされている場合、時間間隔が短すぎる可能性があります。このような場合、各ファイルにはわずかの変更ブロックしかないので、ディスク領域を浪費します。これを避けるために、コンパクト・バックアップ・モードをONにするか、バックアップの時間間隔を長くします。

毎回多くのジャーナル・ファイルがバックアップされる場合、時間間隔が長すぎると考えることができます。このような場合、ディスク障害の際に多くのデータを紛失する可能性があるので危険です。一度の差分バックアップで、1つのジャーナル・ファイルがバックアップされるの理想的です。これはストレージ領域を節約し、ジャーナル・データを紛失するリスクを低減します。

メディア障害から回復する時間を短縮するためには、バックアップ・サーバーを使っている場合でも、完全バックアップを定期的に実行します。加えて、これはバックアップ・ストレージ領域の節約にも貢献します。

バックアップ・サーバーが起動していても、手動の差分バックアップを行うことができます。その場合、前述のようにバックアップID、日時、バックアップ・ファイルの場所を書き留めておきます。

### ファイルオブジェクトのバックアップ履歴ファイル

ファイルオブジェクトのバックアップ履歴ファイル、**dmFoMap.his**には、環境設定パラメータに基づいてバックアップされた全ファイルオブジェクトの記録が保管されます。**dmFoMap.his**は、「**<DB_BkDir>/FO**」ディレクトリにあり、元の外部ファイル名とバックアップファイル名を記録した純粋なASCIIテキストファイルです。

ファイル・フォーマットは、以下のとおりです。:

```
Database Name: MYDB
Begin Backup FO Time: 2001.5.13 2:33
FO Backup Directory: /DBMaster/mydb/backup/FO (i.e. DB_BkDir/FO)
[Mapping List]
s, fo0000000000.bak, "/DBMaster/mydb/fo/ZZ000001.bmp"
u, fo0000000001.bak, "/home2/data/image.jpg"
....
s, fo0000002345.bak, "/DBMaster/mydb/fo/ZZ00AB32.txt"
```

最初のカラムの**s**や**u**は、それぞれシステム・ファイルオブジェクトとユーザー・ファイルオブジェクトを表します。2番目のカラムは、バックアップ名を与え、3番目のカラムは元のファイルオブジェクトの完全名と絶対パスを与えます。

## 9.8 リストアの選択肢

データベースのリストアは、データベースを最新の完全バックアップ時点の状態にして、バックアップしたジャーナル・ファイルに残された変更箇所を加えてデータベースを再構築することです。

## リストア方法の判断

データベースがNONBACKUPモードの場合、ディスク障害後のリストアのための唯一の方法は、最新の完全バックアップをリストアし、データベースを再起動することです。最新の完全バックアップ以降に実行した作業は全て失い、再度入力しなければなりません。

データベースがBACKUP-DATA、BACKUP-DATA-AND-BLOBモードの場合は、データベースを構築するいくつかのリカバリ方法があります。

## リストアの準備

ディスク障害後にデータベースをリストアする場合には、以下の点を考慮します。

- データベースをリストアする時点

ディスク障害発生時点にリストアするためには、損傷したデータベースの全ジャーナル・ファイルをバックアップします。DBMasterではこれらのファイルを使って、最も新しい時点にデータベースをリストアします。

- 以前バックアップしたファイル

最新の完全バックアップとその後に続く差分バックアップがどこにあるかを調べます。例えば、毎月30日に完全バックアップを、10日毎に差分バックアップを実行したと想定します。システム障害が5月の25日に起こった場合、4月30日の完全バックアップと、5月10日と5月20日の差分バックアップと、5月25日の損傷を受けたジャーナル・ファイルが必要になります。これらのファイルの位置を確認すると、5月25日の障害発生前の状態までデータベースをリストアすることができます。オンライン・バックアップの全情報はバックアップ履歴ファイルに保管されているので、DBMasterはデータベースをリストアする際に、この情報を読み込んで取得します。

## リストアの実行

DBMasterのJServer Managerを使ってリストアを実行することができます。

➲ **データベースをリストアする:**

1. JServer Managerは、最新の完全バックアップ(BLOBファイル、データファイル、ジャーナル・ファイル、dmconfig.ini)の全ファイルを、dmconfig.iniファイルのDB_DbDirキーワードで指定したディレクトリにコピーします。
2. データベースを特定の時点の状態に再構築する場合、データベースのリストアの日時をセットします。最も新しい日時にリストアする場合は、このステップを省略して下さい。
3. 差分バックアップのジャーナル・ファイルの位置を指定し、バックアップIDの順にファイルをリストします。
4. ディスク障害発生後に作成したバックアップしたジャーナル・ファイルか、それ以外のバックアップされた全ジャーナル・ファイルが、リストア処理に使われます。
5. ファイルのリストアを完了し、データベースを安定した状態に回復すると、ユーザーはデータベースを使い始めることができます。