DBMaster

DBMaster 入門編



CASEMaker Inc./Corporate Headquarters

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

Contact Information:

CASEMaker US Division E-mail : <u>info@casemaker.com</u> Europe Division E-mail : <u>casemaker.europe@casemaker.com</u> Asia Division E-mail : <u>casemaker.asia@casemaker.com</u>(Taiwan) E-mail : <u>info@casemaker.co.jp</u>(Japan)

www.casemaker.com/support

©Copyright 1995-2015 by Syscom Computer Engineering Co. Document No. 645049-236296/DBM54J-M09302015-TUTO 発行日:2015-09-30

ALL RIGHTS RESERVED.

本書の一部または全部を無断で、再出版、情報検索システムへ保存、その他の形式へ転作することは禁止 されています。

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	こめに	1-1
	1.1	その他のマニュアル	
	1.2	字体の規則	1-3
2	RD	BMSの基礎	2-1
	2.1	構文ダイアグラム	
	2.2	RDBMS の機能	
	2.3	データ・モデル	
	2.4	データの独立性 物理的データ独立性 論理的データ独立性	2-4 2-4 2-5
	2.5	高水準言語サポート	
	2.6	トランザクション管理 トランザクションとは 同時実行制御 ロックの概念	2-6
	2.7	整合性制御	

	2.8	アクセス制御	2-10
		ユーザー権限	2-11
		トランザクション権限	2-11
	2.9	RDBMS リカバリ	2-11
		システム障害	2-12
		メディア障害	2-12
3	RD	BMSアーキテクチャ	3-1
	3.1	RDBMS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	3-3
		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	RDBMS 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
		RDBMSエンジン	3-9
4	デー	-タベース	
_	4_1	★ 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、	4.2
	4.0		4 Q
	4.2	amconig.mッアイル	4- 2
		IF灰	4-3 4 4
		フォーマット	
		マネー、フィー	4-5
		キーワード	4-5
		コメント	
	4.3	dmSQL	4. 6

	作業スペース	
4.4	J ツール	
	JConfiguration Tool	
	JServer Manager	
	JDBA Tool	4-10
4.5	データベースの作成	4-10
	練習データベース	4-10
	接続ハンドル	
	初期設定ユーザー	4-12
4.6	データベース・モード	
	シングルユーザー・モード	
	マルチユーザー・モード	
	クライアント/サーバー・モード	
表		5-1
5.1	表領域	5-1
	標準表領域	
	自動拡張表領域	
	システム表領域	
	初期設定ユーザー表領域	
	一時表領域	
5.2	データ型	5-3
	BIGINT	
	BIGSERIAL (開始番号)	
	BINARY(サイズ)	
	CHAR (サイズ)	
	DATE	
	DECIMAL (NUMERIC)	
	DOUBLE	
	FILE	
	FLOAΓ	

5

	INTEGER	
	JSONCOLS	5-10
	LONG BINARY(BLOB)	5-15
	LONG VARCHAR(CLOB)	5-15
	NCHAR(サイズ)	5-16
	NVARCHAR(サイズ)	5-17
	OID	5-18
	REAL	5-18
	SERIAL (開始番号)	5-19
	SMALLINT	
	TIME	5-20
	TIMESTAMP	5-21
	VARCHAR (サイズ)	5-22
	メディア・タイプ	5-22
5.3	表の作成	5-23
	カラムの初期設定値	
	ロック・モード	
	フィルファクタ	
	非キャッシュ	
	一時表	
*	L	
テー	-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	6-7
	 ホスト変数を使った更新	6-7
	OIDを使った更新	6-8
6.3	結果セット	6-9

6

		表の選択	6-9
		カラムの選択	6-12
		行の選択	6-12
	6.4	演算子の種類	6-13
		比較演算子	6-13
		論理演算子	6-15
		算術演算子	6-16
	6.5	削除	6-16
		標準SQLを使った削除	6-17
		ホスト変数を使った削除	6-17
		OIDを使った削除	6-18
7	デー	-タベース・オブジェクト	7-1
	7.1	ビュー	
		ビューの作成	7-2
		ビューの削除	
	7.2	シノニム	
		シノニムの作成	
		シノニムの削除	
	7.3	索리	
		索引の作成	
		索引の削除	
•	_		
8		- サーと権限	8-1
	8.1	セキュリティ管理	8-1
	8.2	権限レベル	
		CONNECT	
		RESOURCE	
		313DDA	

	SYSADM	8-3
8.3	新規ユーザー	8-4
	ユーザー・アクセス	8-4
	複数のユーザー	8-5
8.4	権限レベルを上げる	8-6
	複数のユーザー	8-6
8.5	権限レベルを下げる	8-7
8.6	ユーザーを削除する	8-8
8.7	パスワード	8-8
8.8	グループを管理する	8-9
0.0	作成	
	メンバーの追加	
	メンバーの削除	8-11
	グループの削除	8-11
	入れ子グループ	
8.9	表レベル権限	8-12
	SELECT(選択)	
	INSERT (挿入)	
	DELETE(削除)	8-13
	UPDATE(更新)	8-13
	INDEX(索引)	
	ALTER (修正)	
	REFERENCE(参照)	8-14
8.10	GRANT権限	8-14
	GRANT表権限	
	GRANTカラム権限	8-16
8.11	REVOKE権限	8-17
	REVOKE表権限	8-17
	REVOKEカラム権限	8-19

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
	メディア障害
9.2	リカバリの方法
	ジャーナル・ファイル
	チェックポイント・イベント
	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
	差分バックアップ9-6
	増分バックアップ9-7
	オフライン・バックアップ9-7
	オンライン・バックアップ9-8
	現在までのオンライン増分バックアップ
	バックアップの組み合わせ9-9
9.4	バックアップ・モード 9-10
	NONBACKUPモード9-10
	BACKUP-DATAモード9-10
	BACKUP-DATA-AND-BLOBモード9-11
	表領域BLOBバックアップ・モード9-11
	ファイルオブジェクトのバックアップ・モード9-12
	ストアドブロシーシャのバックアップ・モード9-14
	バックアッフファイルの上縮
	ハックアップ・モードの設定
9.5	オフライン完全バックアップ
	dmSQLを使ったオフフイン完全バックアップ9-21
	JServer Managerを使ったオフフイン完全バックアップ …9-21
9.6	バックアップ・サーバー 9-22

Q 7

••••		
	バックアップ履歴ファイルの割り当て	
	バックアップ履歴ファイルを理解する	
	バックアップ履歴ファイルの使用	
	ファイルオブジェクトのバックアップ履歴フ	アイル9-64
9.8	レプリケーションデータベースにバックアッ	プする .9-64
9.9	リストアの選択肢	9-66
	リストア方法の判断	
	リストアの準備	
	リストアの実行	

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 っ柔軟なSQLデータベース管理システム(RDBMS)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。DBMasterはJavaツールrのインターフェースとCOBOL言語に対して のDCIインターフェースもサポートします。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DBMasterの外部ファイルを編集 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、1.1章「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」 をご覧下さい。
- 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 ご覧下さい。
- 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」をご覧下さい。
- 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」をご覧下さい。
- 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、JDBCプログラムについての詳細は、「ODBCプログラマー参 照編」と「JDBC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」をご 覧下さい。

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	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
→ 運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

RDBMSの基礎

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 RDBMSの機能

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。
- アクセス制御一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。RDBMS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

RDBMS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。

RDBMS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。この物理構造の変更によって、アプリケーション・ プログラムの実行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、RDBMS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 透明させるデータ・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、RDBMS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、 RDBMS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、RDBMS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、RDBMSは初めての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。RDBMS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RDBMS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。

RDBMS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RDBMS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。RDBMS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。トランザクションを実行する場合に、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。RDBMS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、RDBMS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、RDBMS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、RDBMS 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RDBMS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時間ではなく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 RDBMS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RDBMSのためには、不正ユーザーのアクセス を防ぎ、正規ユーザーにのみ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 ョン権限の二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RDBMSのみ知っていて、RDBMS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 商品の価格を確認す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RDBMS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 RDBMS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、RDBMS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、RDBMS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。RDBMS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RDBMS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、RDBMSはトラ ンザクション・ジャーナルを使って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つの有効なデータベースバックアップを作成します。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 RDBMSアーキテクチャ

RDBMSのアーキテクチャには2つの側面があります。論理RDBMSアーキ テクチャと物理RDBMS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、 RDBMS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 RDBMS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、RDBMS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RDBMS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RDBMS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、RDBMS 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確認することになります。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確認する必要があります。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 RDBMS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、 RDBMS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RDBMSの他の機能も担ってい ます。

フロントエンドは、RDBMS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、RDBMS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RDBMS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 RDBMS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 RDBMSへのメインのインターフェースです。RDBMS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、RDBMS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、RDBMS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、RDBMS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、結果を直ちに確認す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。DBMaster のコマンドラインインタフェースはdmSQL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、RDBMS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。DBMasterは三つの重要な GUIツールをサポートします: JConfigurationツール、JServer Managerと JDBA ツール。

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RDBMS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RDBMS特有のものなので、あるRDBMSのAPIを使って書かれている プログラムは、他のRDBMS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RDBMSであれば、同様にサポートしている他のRDBMS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RDBMS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。

RDBMSエンジン

RDBMSエンジンは、RDBMS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。

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。
- DBMasterのユーティリティを使用してデータベースを管理する方法
- 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。データベース を名づけると、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、長さは1~128文字からの名前を慎 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			
l_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DBMaster5.4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.** カレントディレクトリ
- 2. DBMASTERの環境変数で指定したディレクトリ
- **3.** DBMasterのインストールディレクトリ: -dbmakster\Version

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。

Э 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	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 ちに結果セットを確認することができます。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:
 - 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。例えば、5.4。
 - 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

DBMaster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へ 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 5.4>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 5.4>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確認するこ 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 5.4> 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:
 - 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確認する場合は、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確認する:

- 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。
- 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:

- 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:

- 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

マルチューザー・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以上の権限を 持つユーザー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. [ユーザー名] の欄で、SYSADMを入力します。
 - 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;
 - 7. Enterを押して下さい。

- **8.** 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。
 dmSOL> DISCONNECT;
- 9. Enterを押して下さい。

UNIX

- **○** UNIXでdmServerを起動する:
 - 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。 例えば、5.4。
 - 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。
 - 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
```

- 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

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 を提供します。

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、dmconfg.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。

一時表領域

ー時表領域(TMPTABLESPACE)は外部の一時表(ETT)をストアするため 使用されます。一時表領域は自動拡張表領域です。これは二種類ファイル があります:データファイルとBLOBファイルです。データファイルのロ ジック名はDB_TMPDBで、物理名はDB_TMPDIR/DBNAME.TDBです; BLOBファイルのロジック名はDB_TMPBBで、物理名は DB_TMPDIR/DBNAME.TBBです。

ユーザーは "create temporary table" または "select into" 句を呼び出す場合、 ETTs を生成して"TMPTABLESPACE "まで保存されます。ユーザーは "TMPTABLESPACE" (システムはディフォルトにETTをTMPTABLESPACE に保存する) に一時表を作成できますが、全部の標準表ではありませ ん。ユーザーは "ALTER TABLESPACE TMPTABLESPACE SET AUTOEXTEND OFF/ON;" と"ALTER DATAFILE *DB_TMPDB/DB_TMPBB* ADD *n* PAGES;"をしますが、TMPTABLESPACEにファイルを追加できなく て、TMPTABLESPACEからファイルを削除もできません。データベースが 作成する場合、TMPTABLESPACEが作成されます。データベースが起動す ると、TMPTABLESPACEは初期値を再設置します。

- TMPTABLESPACEだけに一時表を作成することができます。
- TMPTABLESPACE.に標準表を作成することができません。
- TMPTABLESPACEにファイルを追加できなくて、TMPTABLESPACEからファイルを削除もできません。
- TMPTABLESPACEを削除することができません.。

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、23種のデータ型をサポートします。

BIGINT

BIGINTは、19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。 BIGINTデータ型は8バイトのストレージを使用し、最大値は+9 223 372 036 854 775 807、最小値は-9 223 372 036 854 775 808です。

許容範囲外の値をBIGINT、INTEGER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。

37654

Э 例2

857823

BIGSERIAL (開始番号)

BIGSERIAL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増分します。

BIGSERIALカラムを定義するときに、開始番号のオプションパラメタを指 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BIGSERIALデータ型のカ ラムをもつことができます。

BIGSERIAL番号は整数値であり、BIGSERIALデータ型はBIGSERIAL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INTEGERと 同じ値の範囲をもち、最大値9,223,372,036,854,775,806と最小値-9,223,372,036,854,775,808をもちます。

BIGSERIALカラムに次のシリアル番号を挿入するには、新しい行を挿入するときにBIGSERIALカラムをNULLにします。表の内部シリアル番号が新レコードのBIGSERIALカラムに挿入され、自動的に1増分されます。

Э 例1

NULLの代わりに整数値を与えたBIGSERIAL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られた整数値が挿入されます。内部 シリアル番号は増分されません。与えられ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られた値にリセットさ れます。

Э 例1

10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007

Э 例2

10000, 10001, 5000, 10002, 10003, 11000, 11001, 11002

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、CHARデータ型と同様、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進数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。

Э 例1

'AaBbCcDdEe'x

Э 例2

'41614262436344644565'x

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。

Э 例1

'This is a CHAR string.'

Э 例2

'This is another CHAR string.'

DATE

DATEデータは二種類のタイプがあります:DATEリテラルとDATE定数。 DATEリテラルは現在の日付と表示されて、DATE定数は時間のポイントセットと表示されます。DATE データは(年、月、日)をもつ固定長のデータ型です。DATE型は4バイトのストレージを使用します。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 ることができない場合は、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。

Э 例1a

'0001/01/01'

Э 例1b

'0001/01/01'd

Э 例1c

DATE '0001/01/01'

● 例2a '1999/12/31'

● 例2b '1999/12/31'd

⊃ 例2c

DATE '1999/12/31'

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。

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。

of bytes =
$$\frac{p+1}{2} + 2$$

= $\frac{6+1}{2} + 2$
= 5.5

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1

3452.8373645

Э 例2

736.383732652

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。

Э 例1

2.89837457884451E285

Э 例2

-1.93873634847372E-174

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。 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。

マルチレコードは同じファイルにリンクする場合、DBMasterは余分なファ イルを生成できなく、全部のレコードは同じファイルオブジェクトに共有 されてデスクスペースを節約します。でも、ユーザーに対して、各のレコ ードは専用なファイルオブジェクトと関連します。共有ファイルを更新す ると、DBMasterは新しいファイルを生成します、同時にほかのレコードの 共有ファイルはどんな変更が発生しなくて別のユーザーは元のファイルと 関連できます。そうすると、一つのレコードはファイルに変更することは ほかのレコードがファイルを共有することが影響がありません。

FLOAT

FLOAT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。FLOATは8バイトのストレージ を使用し、有効値の範囲は1.0E308~-1.0E308です。キーワード DB_FLTDBを通じてディフォルトFLOAT型をREAL或いはDOUBLEのデ ータ型を設置することができます。 入力できる最小値は1.0E-308 and -1.0E-308です。

Э 例1

2.89837457884451E285

Э 例2

-1.93873634847372E-174

INTEGER

INTEGERは、10桁の精度とスケール0桁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。INTEGERデータ型はINTを略記で きます。

Э 例1

393848

Э 例2

-298376

JSONCOLS

JSONCOLSタイプはダイナミックカラムのカラムセットです。DBMasterは ダイナミックカラムをサポートしています。JSON文字列から派生された キーで、そして表はその一つのカラムをJSONCOLSカラムとして宣言する 場合にのみ使用されます。ダイナミックカラムに関する詳細につきまして は、データベース管理者参照編のダイナミックカラムの使用というセクシ ョンをご参照ください。表にあるダイナミックカラムはLONG VARBINARYに由来したJSONCOLSタイプとして格納されます。

Э 例1

JSONCOLSタイプを含む表を作成します。

dmSQL> CREATE TABLE student (name CHAR(30), info JSONCOLS);

或いは、

dmSQL> CREATE TABLE student (name CHAR(30)); dmSQL> ALTER TABLE student ADD COLUMN info JSONCOLS;

JSONCOLSタイプの名称を使用して、表studentにデータを挿入します。

```
dmSQL> INSERT INTO student(name,info) VALUES
('jessia','{"desk_id":3,"birthday":"1986-09-19","score":90}');
1 rows inserted
dmSQL> INSERT INTO student(name,info) VALUES ('pine','{"desk_id":4,"birthday":"1987-
03-03","score":95}');
```

1 rows inserted

"SELECT *"を使用して表studentをクエリします。

dmSQL>	SET blobw	idth 80;			
dmSQL>	SELECT * 1	FROM studer	it;		
	NAME			INFO	
====== jessia pine		== ===================================	:90,"birthday :95,"birthday	7":"1986-09-19' 7":"1987-03-03'	","desk_id":3} ',"desk_id":4}
2 rows	selected				

JSONCOLSタイプの名称を使用して、表studentをクエリします。

dmSQL>	SELECT 1	name,	info	FROM	student;					
	NAME					INFO				
jessia			{"scoi	ce":90),"birthda	ay":"19	986-09-1	9","des	k_id":3}	
pine		ł	("scoi	e":95	,"birthda	ay":"19	987-03-0	3","des	k_id":4}	

2 rows selectedJSONCOLSタイプの名称を使用して、表studentのデータを更新します。

dmSQL> UPDATE student SET info = '{"desk_id":7, "birthday":"1986-09-19","score":88}'
WHERE name='jessia';

1 rows updated

birthdayというダイナミックカラムのデータタイプをDATEに修正します。

dmSQL> ALTER TABLE student ADD DYNAMIC COLUMN birthday DATE; dmSQL> SELECT info FROM student;

INFO

```
{"score":88,"birthday":"1986-09-19","desk_id":7}
{"score":95,"birthday":"1987-03-03","desk_id":4}
{"BIRTHDAY":477244800000,"DESK ID":"8","SCORE":"92"}3 rows selected
```

student という JSONCOLSカラムでテキストインデックスを作成します。 dmSQL> CREATE TEXT INDEX idx_stu ON student (INFO);

student という JSONCOLSカラムでビューを作成します。 dmSQL> CREATE VIEW view1 AS SELECT info FROM student; dmSQL> SELECT * FROM view1;

INFO

```
{"score":88,"birthday":"1986-09-19","desk_id":7}
{"score":95,"birthday":"1987-03-03","desk_id":4}
{"BIRTHDAY":477244800000,"DESK_ID":"8","SCORE":"92"}
3 rows selected
```

Э 例2

下記の操作は表studentに基づきもので、詳しい情報については、例1をご 参照ください。

```
ダイナミックカラムの名称を使用して、表studentにデータを挿入します。
/* 暗黙的なデータ転換はデフォルトとしてOFFになっています */
dmSQL> INSERT INTO student (name, score) VALUES (?,?);
dmSQL/Val> 'demi','85'; /* これはokです */
1 rows inserted
dmSQL/Val> 'finly',82; /* INTタイプはCHARタイプに転換できません */
ERROR (9629): 値リスト・シンタックス・エラー
dmSQL/Val> END;
dmSQL> SET itemd ON;
dmSQL> INSERT INTO student (name, score) VALUES (?,?);
dmSQL/Val> 'finly',82; /* 暗黙的なデータ転換機能を使用します */
```

```
1 rows inserted
dmSQL/Val> END;
dmSQL> SET itcmd OFF;
dmSQL> INSERT INTO student(name,desk_id,birthday,score) VALUES('linda','1','1982-01-
01','91');
1 rows inserted
dmSQL> INSERT INTO student(name,desk_id,birthday,score) VALUES('glow','2','1984-03-
25','93');
1 rows inserted
dmSQL> INSERT INTO student (name,desk_id,birthday,score) VALUES('kitty','abc','1980-
02-27','97');
1 rows inserted
```

"SELECT *"を使用して表studentをクエリします。

dmSQL>	SELECT	*	FROM	student;
--------	--------	---	------	----------

NAME	INFO
jessia	{"score":88,"birthday":"1986-09-19","desk_id":7}
pine	{"score":95,"birthday":"1987-03-03","desk_id":4}
mike	{"BIRTHDAY":477244800000,"DESK_ID":"8","SCORE":"92"}
demi	{"SCORE":"85"}
finly	{"SCORE":"82"}
linda	{"BIRTHDAY":378662400000,"DESK_ID":"1","SCORE":"91"}
glow	{"BIRTHDAY":448992000000,"DESK_ID":"2","SCORE":"93"}
kitty	{"BIRTHDAY":320428800000,"DESK_ID":"abc","SCORE":"97"]
8 rows selected	1

ダイナミックカラムの名称を使用して、表studentをクエリします。

dmSQL> SELECT	name, desk_id,	birthday, sco	re FROM student;
NAME	DESK_ID	BIRTHDAY	SCORE
jessia	7	19*	88
pine	4	19*	95
mike	8	19*	92
demi	NULL	NU*	85
finly	NULL	NU*	82
linda	1	19*	91
glow	2	19*	93
kitty	abc	19*	97
8 rows selecte	ed		

ダイナミックカラムの名称を使用して、表**student**のデータを更新または削除します。

```
dmSQL> UPDATE student SET score='88' WHERE name='linda';
1 rows updated
dmSQL> DELETE FROM student WHERE desk_id='2';
1 rows deleted
```

当該表にダイナミックカラムの説明を追加します。 dmSQL> ALTER TABLE student ADD DYNAMIC COLUMN desk_id INT; dmSQL> ALTER TABLE student ADD DYNAMIC COLUMN score DOUBLE;

データを表studentに挿入します。

dmSQL> INSERT INTO student(name, desk id, age, score) VALUES('jane','12','1982-05-07',96);

ERROR (6150): [DBMaster] INSTER/UPDATE値のタイプとカラムのデータ型が矛盾しているか、比較オペランドの値と記述したカラムのデータ型が矛盾しています

dmSQL> INSERT INTO student(name, desk id, age, score) VALUES('jim',8,'1984-09-26',98);

1 rows inserted

dmSQL> SELECT name, desk id, birthday, score FROM student;

NAME	DESK_ID		BIRTHDAY	SCORE		
		==		=		
jessia		7	1986-09-19	8.80000000000000e+001		
pine		4	1987-03-03	9.500000000000000e+001		
mike		8	1985-02-15	9.20000000000000e+001		
demi	NULL		NULL	8.50000000000000e+001		
finly	NULL		NULL	8.20000000000000e+001		
linda		1	1982-01-01	8.80000000000000e+001		
kitty	NULL		1980-02-27	9.700000000000000e+001		
jim		8	NULL	9.800000000000000e+001		
8 rows selected						

scoreというダイナミックカラムのデータタイプを修正します。 dmSQL> ALTER TABLE student MODIFY DYNAMIC COLUMN score TYPE TO INT;

desk_idというダイナミックカラムでインデックスを作成します。 dmSQL> CREATE INDEX idx1 ON student(desk id);

birthdayというダイナミックカラムの記述情報を削除します。 dmSOL> ALTER TABLE student DROP DYNAMIC COLUMN birthday;

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。BLOBカラムの最大長は8Tを超えません。BINARYデータ 型と比べて、異なったがあります。BLOB型は入力データだけがデータ ベースに格納されて、余分なスペースが0で満たされません。

CHAR型と同じようにLONG VARBINARY型のデータを入力すると、デー タの前後に引用符(')で括る必要があります。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。

Э 例1

'AaBbCcDdEe'

Э 例2

'41614262436344644565'x

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8Tを超えません。

入力したCHARデータと同じように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。カラムのサイズを計算するときは、この点を考慮に入れ ます。

Э 例1

'This is a LONG VARCHAR string.'

Э 例2

'This is another LONG VARCHAR string.'

NCHAR(サイズ)

NCHARデータ・タイプはどんなUnicode文字も含むことができる固定長デ ータ型です。各Unicode文字は、UTF16リトル・エンディアン(LE)符号づけ の中で記憶装置を2バイト占めます。(サイズ)パラメーターは、カラムの2 バイトの文字の数を決定します。NCHARカラムを作る場合(サイズ)パラメ ーターを入力しなければならず、範囲は最小1から最大1996です。

NCHARデータのカラム長さはサイズより短くと、データがスペース文字 で埋められるでしょう。NCHARデータを入力する場合、シングルクォー ト(')でUnicode文字を囲んで、「N」を前に付けてください。

Э 例1

下記は、Unicodeデータ・エントリーのシンタックスを実証します: N'Unicode Data'

NCHARデータが16進法のフォーマットに入力される場合は、引用符('')で 16進法のストリングを囲んで、「u」文字を追加してください。

Э 例2

下記は、3文字の16進法Unicodeデータ・エントリーのシンタックを実証します:

'610a620b63f1'u

文字ストリングがUnicodeカラムへ入力された時「N」が前に付けられない 場合は、自動的にローカル・コードからUnicodeに変換されるでしょう。 Unicode文字が規則的なCHARタイプ・カラムに入力される場合、Unicode 文字はdmconfig.iniパラメーター**Db_LCode**によって定義されたローカル・ コードに変換されるでしょう。ローカル・コードに定義されない文字は□ によって表わされるでしょう。

NCHARデータ型の同意語はNATIONAL CHAR(サイズ)およびNATIONAL CHARACTER(サイズ)です。
NVARCHAR(サイズ)

NVARCHARデータ・タイプはどんなUnicode文字も含むことができる可変 長さのデータ・タイプです。各Unicode文字は、UTF16リトル・エンディ アン(LE)符号づけの中で記憶装置を2バイト占めます。(サイズ)パラメータ ーは、カラムの2バイトの文字の数を決定します。NVARCHARカラムを作 る場合(サイズ)パラメーターを入力しなければならず、範囲は最小1から最 大1996までです。

NVARCHARデータが、カラム長さより短いカラムに入力される場合、デ ータがスペースで当てがわれません。NVARCHARデータを入力する場 合、シングルクォート('')でUnicode文字を囲んで、「N」を備えた引用の 前に付けてください。

Э 例1

下記は、Unicodeデータ・エントリーのシンタックスを実証します: N'Unicode Data'

NVARCHARデータが16進法のフォーマットに入力される場合は、引用で 16進法のストリングを囲んで、「u」文字を追加してください。

Э 例2

下記は、3文字の16進法Unicodeデータ・エントリーのシンタックスを実証 します:

'610a620b63f1'u

文字ストリングがUnicodeカラムへ入力された時「N」が前に付けられない 場合は、自動的にローカル・コードからUnicodeに変換されるでしょう。 Unicode文字が規則的なVARCHARタイプ・カラムに入力される場合、 Unicode文字はdmconfig.iniパラメーター**Db_LCode**によって定義されたロー カル・コードに変換されるでしょう。ローカル・コードに定義されない文 字は□によって表わされるでしょう。

NVARCHARデータ・タイプの同意語はNATIONAL CHAR VARYING(サイズ)、NCHAR VARYING(サイズ)、NATIONAL VARCHAR(サイズ)および NATIONAL CHARACTER VARYING(サイズ)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。レ コードを挿入する場合、DBMaster は自動的に各のレコードにOIDを生成さ れ、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。

生成する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カラムは表に隠しカラムです。SELECT * FROM CUSTOMERSのような問合せでは検索できません。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/更新し、SQL言語では、 このような使用はあまり行いません。OIDは、通常に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。

REAL

REAL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。REALは4バイトのストレージを使 用し、有効な入力値の範囲は3.402823466E38から-3.402823466E38までで す。最小のエラー範囲は1.175494351E-38から-1.175494351E-38までです。 DOUBLEのようなデータ型から最大有効値を超える値を含む移動は失敗と なり、DBMasterは変換エラーを返します。

Э 例1

3.583837E34

Э 例2

-1.873653E-21

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。

Э 例1

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

Э 例2

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

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。

Э 例1

4769

Э 例2

8376

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。

Э 例1a

'22:04:05'

'22:04:05't

Э 例1c

TIME '22:04:05'

Э 例2a

'10:04:05.666 PM'

Э 例2b

10:04:05 PM't

Э 例2c

TIME 10:04:05 PM'

TIMESTAMP

TIMESTAMPは二つの種類があります:TIMESTAMP関数とTIMESTAMP カラム。TIMESTAMP関数は現在の日付と時刻を表示され、時間によっ て変わります;TIMESTAMPカラムは固定時刻を表示します。

この二種類TIMESTAMP型は固定長のデータ型です。日付と時刻を保存して11バイトのストレージを使用し、精度17とスケール10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 使用して日付と時刻を表示し、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。

Э 例1a

'1997/01/01 10:02:03.444 PM'

Э 例1b

'1997/01/01 22:02:03'ts

Э 例1c

TIMESTAMP '1997/01/01 10:02:03'

Э 例2a

'01.01.1997 22:02:03'

'01.01.1997 22:02:03'ts

Э 例2c

TIMESTAMP '01.01.1997 22:02:03'

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。

Э 例1

'This is a VARCHAR string.'

Э 例2

'This is another VARCHAR string.'

メディア・タイプ

Microsoft[™] Word[™]ドキュメント用全文検索のようなメディア・プロセス機 能の補完するために、ラージオブジェクト・カラムをメディア・タイプと して指定することができます。次のメディア・タイプが利用可能で す:MsWordType、HtmlType、XmlType、MsPPTType, MsExcelType, PDFType、MsWordFileType、HtmlFileType、XmlFileType、MsPPTFileType, MsExcelFileType, PDFFileType。

メディア・タイプは既存のデータ・タイプのドメインとして扱われま す。;MsWordType、MsPPTType, MsExcelType, PDFType, HtmlType と XmlTypeはLONG VARBINARYに相当します。また、MsWordFileType、 HtmlFileType、MsPPTFileType, MsExcelFileType and PDFFileTypeおよび XmlFileType、MsPPTFileType、MsExcelFileType及びPDFFileTypeはFILE タイプ・カラムに相当します。カラムのデータ・タイプを別のものに変更 するALTER TABLEを使用する場合に重要となります。各メディア型の特性 はものとのデータ型の特性と類似しています。 以下のはXMLTYPEの特性です:

- XML完全フォーマットのチェック:挿入/更新した xml内容は完全フォ ーマットになければなりません
- XML 有効性: xmltype カラムを作成する場合、任意に有効な udf を指 定すると、DBMasterはxml 内容を有効になさせます
- XMLデータは本来のフォーマットとして保存されます
- XPath で問い合わせを捜索: xpathを任意に指定してextract関数を使用 してXMLデータのノードをクエリ/見つけます
- Xpathから指定したXML内容を更新します
- XPath extractで索引を作成:頻繁にクエリするxpath表現式に索引を作 成することを通じてクエリのスピードを上げます。
- xmltypeカラム或いはほかのデータタイプをxmltypeに変更することが 許可しません
- ⊃ 例

dmSQL> CREATE TABLE minutes (id INT, meeting_date DATE, doc MSWORDFILETYPE); dmSQL> INSERT INTO minutes VALUES (1, 3/3/2003, 'c:\meeting\20030303.doc');

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、 2000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データ型のような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。

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。

Э 例2

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,

Туре3	INT,	Type4	INT,
Rating	CHAR(10),	Length	SMALLINT,
Color	CHAR (3) ,	BW	CHAR(3),
Таре	CHAR(3),	Disc	CHAR(3),
Quantity	SMALLINT,	Supplier	INT,
Data	DATE,	Price	FLOAT,
Rate	SMALLINT)		
dmSQL> CREATE TABLE Receipts (Numb	er SERIAL,	Customer	INT,
Employ	ee INT)		
dmSQL> CREATE TABLE LineItems (Rec	eipt INT N	OT NULL, I	LineItem INT NOT NULL,
M	ovie INT,	Ç	Quantity SMALLINT,
Am	ount SMALL	INT)	

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



<table_column_definition>構文



< as_select_statement > 構文

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、NULL又は組み込み関数です。組み込み関 数の詳しい情報については、「SQL文と関数参照編」をご参照ください。

DBMasterはキーワードUSER、SYSTEMとON UPDATEで挿入及び更新操作 をしている際のディフォルトカラムの属性を設定することをサポートしま す。キーワードUSER/SYSTEMが選択可能です。これらのキーワードは、 ユーザーがINSERT/UPDATEステートメントを使用してデフォルト値を持 つカラムの値を変更できるかどうかを指定します。キーワードUSERがデ フォルトとして使用されます。キーワードUSERはユーザーがその値を変 更できるのを指定し、キーワードSYSTEMはユーザーがその値を変更でき ないのを指定します。キーワードON UPDATEは選択可能です。このキー ワードは、その他のカラムの値が変更される場合に、デフォルト値を持つ カラムの値が自動的に変更できるのを指定します。この三つのキーワード は主に表定義に用いられ、表定義に関する詳しい情報については、「SQL 文と関数参照編」のCREATE TABLE、ALTER TABLE ADD COLUMN及び ALTER TABLE MODIFY COLUMNとのセクションをご参照ください。

INSERT/UPDATE文を使用してカラムの値を指定する場合、データベース の表をロードしている間にSYSTEM DEFAULT属性を持つカラムの値が上 書きされるかどうかを指定することができます。このオプションをONに 設定する場合は、その値はデフォルト値に更新され、このオプションを OFFに設定する場合は、元の値がユーザーによって指定された値に更新さ れます。このオプションのデフォルト設定はOFFです。

Э 例1

表tb_staffのカラムnationの初期値を定数'R.O.C.'に指定し、カラムjoinDate の初期値を組み込み関数curdate()に指定する: dmSQL> CREATE TABLE tb staff (nation CHAR(20) DEFAULT 'R.O.C',

ID INTEGER, name CHAR(30), joinDate DATE DEFAULT CURDATE(), height FLOAT, degree VARCHAR(200), picture LONG VARCHAR) IN **ts reg**;

Э 例2a

dmSQL> CREATE TABLE computer(id INT, buy_time TIMESTAMP DEFAULT '2012-03-04 12:12:12', price int); //buy_timeの属性はUSERです。 dmSQL> INSERT INTO computer VALUES(1, '2012-10-10 10:10:20', 3400); //buy_timeの値は ユーザーが指定する'2012-10-10 10:10:20'に変更されます。 1 rows inserted

dmSQL> INSERT INTO computer VALUES(2, '2012-10-11 10:10:20', 5400);					
1 rows insert	ed				
dmSQL> SELECT	* FROM computer;				
ID	BUY_TIME	PRICE			
=		= =======			
1 2	012-10-10 10:10:20	3400			
2 2	012-10-11 10:10:20	5400			
2 rows select	ed				
dmSQL> UPDATE	computer SET price=3200	WHERE id=1; //buy_time	eの値が更新されていません。		
1 rows update	d				
dmSQL> SELECI	* FROM computer;				
ID	BUY_TIME	PRICE			
=					
1 2	012-10-10 10:10:20	3200			
2 2	012-10-11 10:10:20	5400			
2 rows select	ed				

● 例2b

dmSQL> A	ALTER TABLE computer MODIFY (buy	/ time TO buy time TIMESTAMP DEFAULT '2012-03	-
04 12:12	2:12' ON UPDATE); // buy timeの履		
dmSQL> (UPDATE computer SET price=3000 W	WHERE id=1; // buy timeの値は'2012-03-04	
12:12:12	2'に変更されます。		
1 rows u	updated		
dmSQL> S	SELECT * FROM computer;		
ID	BUY TIME	PRICE	
	1 2012-03-04 12:12:12	3000	
	2 2012-10-11 10:10:20	5400	
2 rows s	selected		
dmSQL> (UPDATE computer SET price=3000,	buy time='2012-10-10' WHERE id=1;// buy time	
の値はユー	ーザーが指定する'2012-10-10'に変更さ		
1 rows ι	updated		
dmSQL> S	SELECT * FROM computer;		
ID	BUY TIME	PRICE	
	_ 		
	1 2012-10-10 00:00:00	3000	
	2 2012-10-11 10:10:20	5400	
2 2007 20	aplastad		

2 rows selected

Э 例2c

```
dmSQL> ALTER TABLE computer MODIFY (buy time TO buy time TIMESTAMP SYSTEM DEFAULT
'2012-03-04 12:12:12'); // buy timeの属性はSYSTEM
dmSQL> INSERT INTO computer VALUES(3, '2012-11-10 10:10:20', 4700); // buy timeの値は
ユーザーが指定する'2012-11-10 10:10:20'に変更されません。
1 rows inserted
dmSQL> INSERT INTO computer VALUES(4, '2012-12-11 10:10:20', 2800);// buy timeの値は
ユーザーが指定する'2012-12-10 10:10:20'に変更されません。
1 rows inserted
dmSQL> SELECT * FROM computer;
                          PRICE
  ID BUY TIME
 ____
       1 2012-10-10 00:00:00
                                   3000
       2 2012-10-11 10:10:20
                                   5400
       3 2012-03-04 12:12:12
                                   4700
       4 2012-03-04 12:12:12 2800
4 rows selected
dmSQL> UPDATE computer SET price=4500 WHERE id=3; // buy timeの値が更新されていません。
1 rows updated
dmSQL> SELECT * FROM computer;
                           PRICE
  ID BUY_TIME
  _____
                           ____
       1 2012-10-10 00:00:00
                                   3000
       2 2012-10-11 10:10:20
                                   5400
       3 2012-03-04 12:12:12
                                   4500
       4 2012-03-04 12:12:12 2800
4 rows selected
```


3 2012-03-04 12:12:12	4000	
4 2012-03-04 12:12:12	2800	
4 rows selected		

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。

Э 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。

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。

Э 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

ー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。

Э 例

test E	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。

Э 例1

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。

Э 例2

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。

Э 例3

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

OR REPLACE: OR REPLACEオプションは選択可能です。このオプション はDBMasterが2つの行(新しい行と古い行)が一部のカラムに同じ値を持 っている場合に、新しい行で古い行を置き換えるかどうかを指定します。 すなわち、ユーザーがテーブルに挿入する行がすでにテーブル(主キーま たは一意のインデックスにより判断する)に存在する場合は、古い行がテ ーブルから削除されてから、新しい行がテーブルに挿入され、そうでない 場合は、新しい行がテーブルに直接挿入されます。

INSERTコマンドの詳細については、「*SQL文と関数参照編」のSQL文*をご 参照ください。

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>プロンプトのValue状態になります。

Э 例1

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>

Э 例2

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');
```

```
INSERTとホスト変数を使う:
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/Val> end;
```

Э 例4

```
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 & INTEGER:

123, -252783

FLOATとDOUBLE:

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

CHAR & 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。

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

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。

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。

Э 例1

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

ホスト変数でBLOBデータを挿入する: dmSQL> INSERT INTO blob table VALUES (,'5f5f5f5f5f5f'x);

Э 例3

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。

Э 例4

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はUPDATE文の構文を表示します。



標準SQLを使った更新

Э 例

標準SQLを使って、管理者ファイルを更新する: cmSQL> update Employee set Manager = 10000 where LastName = 'Carter';

1 row updated

Employeeは表の名前です。この文は管理者の名前を**Greg Carter**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>プロンプトの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。

Э 例1

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>

Э 例2

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、相応的な行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。

Э 例

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。

Э 例1

表の全情報を選択する: dmSQL> select * from

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**表から全従業員のデータを選択する場合、以下のコマンドを使用 します。

Э 例2

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 ムのデータを快速に確認することができます。

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。

Э 例3

LINEWIDTH 機能をOFFにし、カラムの全データを確認する: dmSQL> SET LINEWIDTH off; 全カラムのデータ全てを確認する場合は、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。

Э 例4

全カラムのデータ全てを確認する:

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

戻りデータの例:

SELECT * from Employee;						
Number FirstName	e LastName	e Manage	er Phone	BirthDate	HireDate	
10000 Gabriel	Davis 1	.0000	736-8376	1957-01-21 1	995-04-24	
10001 Greg	Carter	10000	NULL	NULL	NULL	
10002 Dean	Cougar	NULL	NULL	1970-12-08	NULL	
10003 Benson	Armstron	ıg 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	975-02-09	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。

Э 例5

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、エラーメッセージが戻されます。

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。

Э 例1

カラムを指定する:

dmSQL> SELECT <column name>, <column name>, ... FROM

コマンドは、カラム名のリストで指定したカラムのデータを選択します。

Э 例2

Employee表のFirstName、LastName、Phoneカラムを確認する: 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	= ====================================	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。 WHERE句で定義する条件式に合致しているデータは結果セットに含みます。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。

Э 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。

演算子	解説
=	数式の <i>等しい</i> 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 <i>等しくない</i> 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。
IS	特別値のテストを表します

比較演算子は以下のとおりです。

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。

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* H	HireDate
10006	======================================	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、レコードの唯 一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2

LastNameがA、B、Cで始まる全従業員を回収する: cmSQL> SELECT * FROM Employee WHERE LastName BETWEEN 'Aa' and 'Cz';

戻りデータの例:

Number	FirstName	LastName	Manager	Phone	BirthDa* 1	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'をクエリ条件として使用した場合、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。

Э 例3

LastNameが'C'で始まる従業員の名前を取得する: chmSQL> 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。

以下の例は1995年にカナダで製作された映画の名前を回収することを評します:

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た後結果を戻ります。

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。UPDATE文の構文は下記のようになります。



図 6-4: UPDATE文の構文

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。

Э 例1

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。

Э 例2

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状態を終了し文の更新を完了します。

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>

Э 例2

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

OIDs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 OIDsを使用することはありません。

Э 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 _______ 010000002000800 1 rows selected

dmSQL> DELETE FROM Employee WHERE OID='010000002000800'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、ユーザー定 義ビューとデータベース・オブジェクト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。 ビューがクエリの表からから派生しますので、ビューはデータだけ問い合わせます。ユーザーはビューに更新、挿入、削除することができません。

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

Э 例1

Employee表の3つのカラムにのみ閲覧を制限する:

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のみ確認することができます。

注 ビューを定義する問合せに、UNION演算子を含むことができません。

CREATE OR REP LACE VIEW 文を使用します。例えば、ビューvi_staffはも う存在すると、ほかのユーザーはtb_staffから二つのカラムだけ見えます が、今は権限を変更しなくてもユーザーに三つのカラムを見せるため、以 下のようにSQLコマンドCREATE OR REP LACE VIEWを使用してこのビュ ーを再び作成します。それから、ユーザーはビューvi_staffを通じて表 tb_staff から三つのカラム(名、IDと年齢)を確認することができます。

以下の例はビュー**vi_staff**を再作成する方法を表示します。ユーザーは表 の3つのカラムがビューできます: **name、ID**と**age**。 dmSQL> CREATE OR REPLACE VIEW vi_staff (empName, empId, empAge) AS SELECT name, ID, age FROM tb staff;

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削除され、元の表には影響 しません。



図 6-4: DROP VIEW文の構文

IF EXISTS:ビューが存在しない場合エラーを起こさないことを確保します。

DROP VIEWの詳しい情報については、「*SQL文と関数参照編」のSQL文*を ご参照ください。

Э 例1

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。

Э 例2

DROP VIEW IF EXISTSコマンドを使用してビューvi_staffを削除する: dmSQL> DROP VIEW IF EXISTS vi_staff;

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。シノニムは所有者の名前が持っていないので、、全 データベース内でシノニムの名前は一意でなければなりません。シノニム はdmSQL或いはJDBAツールで作成/削除できます。

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 OR REPLACE SYNONYM文の構文

Э 例1

CREATE SYNONYMコマンドを使用します: dmSQL> CREATE SYNONYM staff FOR SYSADM.tb staff;

表*tb_staff*の所有者は*SYSADM*であることを仮定します。このコマンドは表 SYSADM.tb_staffに別名staffを作成します。全部のデータベースユーザはシ ノニムstaffを通じて直接に表SYSADM.tb_staffを参考できます。

CREATE OR REPLACEコマンドを使用してシノニムstaffを削除します: dmSQL> CREATE OR REPLACE SYNONYM staff FOR SYSADM.tb staff;

表SYSADM.tb_staffの別名*staff*はもう存在したことを仮定します。削除しな くてもこれを再作成できます。

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削除されます。

Э 例1

DROP SYNONYMコマンドを使用してシノニム**Employee**を削除する: dmSQL> DROP SYNONYM Employee;

Э 例2

DROP SYNONYM IF EXISTSコマンドを使用してシノニム**Employee**を削除 する:

dmSQL> DROP SYNONYM IF EXISTS 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。

一つの索引は1-32カラムから構成されています、表にすべてのカラムは索引を作成することができます。

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:

Э 例

	name	age
ŀ		
ċ	Jeff	49
F	Kevin	40
ċ	Jerry	38
F	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

表に索引を作成するため索引名と索引にするカラムを指定する必要があり 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。

DBMasterに、ユーザーは表をほかの表領域に移動することができます。同時に、索引とこの表は同じ表領域に存在すると、索引も移動できます。異なる表領域にある場合、索引がほかの表領域に移動できません。だから、ユーザーはほかの表領域に索引を再作成することができます。



図 7-3: CREATE INDEX文の構文

Э 例1

表EmployeeのカラムNumber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。

表**Employee**のカラム**Number**に一意索引idx1を作成する: dmSQL> CREATE UNIQUE INDEX idx1 ON Employee (Number);

Э 例3

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

ユーザーはJDBAツール或いはdmSQLのDROP INDEX文を使って索引を 削除します。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。

Э 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、データをランダムに変更したりする こともできません。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。

ー意のユーザー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)、SYSDBA及びSYSADM(システム管理者)等の5種類の権限 レベルがあります。CONNECT、RESOURCE、DBAまたはSYSDBA権限を 持つユーザーは複数存在するかもしれませんが、SYSADM権限を持つユー ザーは一つのみです。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

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。表の所有者或いはDBA以上の 権を持つ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。表権限が与えられた後、 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

Resource権限は、データベースに表を作成したり、以前に自分で作成した 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、所有表の権限の 付与と取り消しもできますが、表の所有者またはDBA権限を持つユーザー 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

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。

SYSDBA

SYSDBA権限を持つユーザーは、SYSADMまたはSYSDBA以外の権限を持つ ユーザーにCONNECT、RESOURCE及びDBA権限を与えるまたは取り消す ことができ、これらのユーザーのパスワードを変更することもでき、低権 限ユーザーにACL(アクセスコントロールリスト)を設定することもでき ます。また、SYSDBA権限を持つユーザーはDBA権限を持つユーザーのあ らゆる権限を有しますが、そして、SYSADMのみ他ユーザーにSYSDBA権 限を与えるまたは取り消すことができます。SYSDBA権限を持つユーザー に対して、SYSADMはそのSYSDBA権限を取り消す場合には、当該ユーザ ーはDBA権限を持っていますが、DBA権限を取り消す場合には、当該ユー ザーはSYSDBA権限及びDBA権限を持っていません。

SYSADM

各データベースにSYSADM権限ユーザーは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及びSYSDBA権限を持つユーザ ーのみ、他のユーザーの権限レベルの付与/取り消し/変更することがで き、ユーザーのパスワードも変更できます。

8.3 新規ユーザー

SYSADMまたはSYSDBA権限を持つユーザーのみデータベースに新規ユー ザーを追加し、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:
 - 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>;
 - 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 "";
 - 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128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か引数**DB_IDCap**によって決 定します。**DB_IDCap=0**になると、全部の識別子(表名、ユーザー名、パ スワード)も大文字と小文字を識別します。つまり、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。**DB_IDCap**のディフォ ルト値は1です、この引数の値を指定しないと、全部の識別子は大文字と 小文字を識別しません。

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。

ユーザーが持つ権限をより高いレベル(RESOURCE、DBA、SYSDBA及び SYSADM)にアップする前に、まずは当該ユーザーに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。

複数のユーザー

同時に複数のユーザーにアカウントを作成します。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、シング ルユーザーを追加する構文と同じです。

コ 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:

- 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>;

- 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;
- 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。

Э 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

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、シングル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。

二人の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。

Э 例1

ユーザーのDBA権限を取り消す: dmSOL> REVOKE DBA FROM Judy;

Э 例2

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権限が連続に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。

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。このコ マンドはユーザー権限を下げることと同じです。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るユーザーリストから削除さ れます。

Э 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

アカウントを作成する場合、SYSADMは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

©Copyright 1995-2015 Syscom Computer Engineering Co.

```
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;
```

Э 例2

```
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。

Э 例3

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。

Э 例4

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; dmSOL> 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。

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。

CREATE GROUP — group_name —

図 8-4: CREATE GROUP文の構文

グループ名は文字英数字を含みます、グループ名の長さに制限はありませんが、最初の8文字のみ使用されます。例えば、グループ

CompanyEmployeeと**CompanyExecutives**は同じで**CompanyE**になります。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。

グループ名は大文字と小文字が区別します。だから、companyExecutivesと CompanyExecutivesは二つの異なるグループを代表します。このような状 況をさけるために、グループ名を8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。

Э 例

マーケティング部の全社員のグループを作成する: dmSOL> 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

Э 例1

グループ**SalesRep**に、ユーザー**Judy**を追加する: dmSQL> ADD Judy TO GROUP SalesRep;

Э 例2

グループSalesRepに、ユーザーJudy、Jeff、Trentを追加する: cmSQL> ADD Judy, Jeff, Trent TO GROUP SalesRep;

メンバーの削除

REMOVE GROUPコマンドを使用して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

Э 例1

グループ**SalesRep**から、ユーザー**Judy**を削除する: cmSQL> REMOVE Judy FROM GROUP SalesRep;

Э 例2

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。 DROP GROUP ____ group_name _____•

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。

Э 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:

- 以下のコマンドを入力します: dmSQL> CREATE GROUP NYSales;
- SalesRepグループにそれを追加します。 dmSQL> ADD NYSales TO GROUP SalesRep;
- 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以上の権限を持つユーザーは使用できる表権限が 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。表の所有者またはDBA以上の権限を持つユーザーは他のユーザーにSELECT権限を与えることができます。

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 の所有者またはDBA以上の権限を持つユーザーは他のユーザーにINSERT 権限を与えることができます。

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 有者またはDBA以上の権限を持つユーザーは他のユーザーにDELETE権限 を与えることができます。

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 者またはDBA以上の権限を持つユーザーは他のユーザーにUPDATE権限を 与えることができます。

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。表の所有者またはDBA以上の権限を持つユーザーは他のユーザーにINDEX権限を与えることができます。

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。表の所有者またはDBA以上の権限を持つユーザーは他のユーザーにALTER権限を与えることができます。

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。表の所有者またはDBA以上の権限を持つユーザーは他のユー ザーにREFERENCE権限を与えることができます。

8.10 GRANT権限

表の所有者或いはDBA以上の権限を持つユーザーは、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。

GRANT表権限

Э 例1

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。

Э 例2

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、キーワードALL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。

Э 例3

JudyにSalesRep表への全ての表権限を与える: 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4

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。

Э 例5

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。

GRANTカラム権限

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。

Э 例1

JudyにSalesRep表のNameカラムへのINSERT権限を与える: cmSQL> GRANT INSERT (Name) ON SalesRep TO Judy;

Э 例2

JudyにSalesRep表の複数カラムへのINSERT権限を与える: cmSQL> 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。

Э 例3

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 REVOKE権限

表の所有者或いはDBA以上の権限を持つユーザーは、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。

REVOKE表権限

Э 例1

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。

Э 例2

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、キーワードALL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。

Э 例3

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。

Э 例4

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。

Э 例5

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与えるまたは取り消すことはできません。

REVOKEカラム権限

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。

Э 例1

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;

Э 例2

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。

Э 例3

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。



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(DBMS)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。RDBMS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、 RDBMSにはデータを回復する方法がいくつかあります。これは、RDBMS 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まれています。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。

本章はデータベースの障害型の可能性及びデータの損失を避けるステップ を説明します。上記の章にdmSQLコマンドラインツールを使用する方法を 例をあげて説明しました、本章はJserver Managerツールを使用してデ ータベースのバックアップとバックアップをする方法について例を挙 げして紹介します。

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。RDBMS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3種類のバックアップは、完全バックアップ、差分 バックアップと増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。DBMasterは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サポートします。

完全バックアップ

完全バックアップはある時点のデータベース全体の複製を作ります、全デ ータと全ジャーナル・ファイルのコピーをとります。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最新の完全バックアップを基礎として、つまり差分の ベースです。そのため、差分バックアップを実行する前、差分ベースが存 在しなければなりません。

差分バックアップは差分ベースを変更されたデータのみ含みます。普通 に、差分ベースは連続な差分バックアップに使用されます。リストアタイ ムで完全バックアップと差分バックアップはこの完全バックアップを基づ いて完全データベースを生成します。

DBMaster差分バックアップはデータファイル(全部のDBファイルとBBフ ァイル)だけをコピーしますが、ジャーナルではありません。ジャーナル ファイルは頻繁に変更されるからです。それで、差分バックアップを実行 する場合、有効なジャーナルブロックだけはコピーされます。

増分バックアップ

増分バックアップは、前回のバックアップ以降に変更したジャーナル・フ アイルのコピーのみ作成します。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。増分バックア 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、完全バックアップ或いは差分バックアッ 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、増分バックアップを実行することができます。

バックアップモードが開いていつので、増分バックアップは全部のトラン ザクションをレコードするジャーナルファイルから構成します。データベ ースが普通モード(DB_SMODE = 1)で実行する場合、完全バックアップ或 いは差分バックアップを終了した前増分バックアップを実行できません; レプリケーションモード(DB_SMODE = 4)で、完全バックアップ或いは差 分バックアップが必要なく増分バックアップが実行できます。

増分バックアップは、ジャーナル・ファイルのみ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。増分バックアップと完全バックアップ或いは差分バックアッ プを併用すると、前回の完全バックアップ、差分バックアップから増分バ 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。オフラインの間は増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、RDBMS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、差分と増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RDBMS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。

現在までのオンライン増分バックアップ

DBMasterには、現在までのオンライン増分バックアップというバックアップもあります。

オンライン増分バックアップと、複数のジャーナルファイルを用いる現在 までのオンライン増分バックアップの違いは、些細ではありますが重要で す。オンライン増分バックアップでは、DBMasterは最後のバックアップ以 降に使用された全ジャーナルファイルをバックアップしますが、使用中の ジャーナルファイルはバックアップしません。現在までのオンライン増分 バックアップでは、DBMasterは使用中のジャーナルファイルも含めたすべ てのジャーナルファイルをバックアップします。これは、オンライン増分 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フルジャ
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増分バックアップは、使用中のジャーナルフ 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、オンライン増分バ ックアップと現在までのオンライン増分バックアップは、全く同じです。 この場合、唯一のジャーナルファイルが使用中のジャーナルファイルで す。このジャーナルファイルが、いずれの増分バックアップの際でもバッ クアップされます。

バックアップの組み合わせ

DBMasterには、完全バックアップ、差分と増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、差分 と増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/増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、オンライン増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増分バッ 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、オンライン増分 バックアップと現在までのオンライン増分バックアップの大きい違いがあ りませんが、注意な所があります。オンライン増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 のオンライン増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、オンライン増分バックア 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増分バックアップは、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、オンライン増分 バックアップと現在までのオンライン増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。いずれの増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、オンライン増分バックアップを実行するかどう かや、増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、オンライン増分バックアップは 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 増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、オンライン増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。

データの損失が許されない場合に上記の方法は有効です。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のバックアップ・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。

Э 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。

ストアドプロシージャのバックアップ・モード

ユーザーはデータベースでのBLOBデータ及びファイルオブジェクトを定 期的にバックアップする以外に、ストアドプロシージャのバックアップも 選択できます。ストアドプロシージャのバックアップはバックアップデー モンによって自動的に完全バックアップを行う際にのみ実行されます。そ れを実行するには、ユーザーはまずバックアップサーバーを開いて、バッ クアップのスケジュール及びディレクトリを設定する必要があります。完 全バックアップに関する詳細については、「データベース管理者参照編」 のセクション15.6 バックアップサーバをご参照ください。

dmconfig.iniのキーワード**DB_BKSPM**の設定によって、ストアドプロシー ジャのバックアップ・モードを指定することができます。

- **DB_BKSPM**=0:ストアドプロシージャをバックアップしません
- DB_BKSPM=1:あらゆるストアドプロシージャをバックアップする

Э 例

関連するキーワードを含むdmconfig.iniファイルからの抜粋を以下のよう に示します。

[MyDB]		
DB_BkSvr = 1	; >	バックアップサーバを起動させます
DB_FBKTm = 14/05/01 00:00:00	; 20	014年の5月1日の午前0時から始まります
DB_FBKTV = 1-00:00:00	; 間	調ね毎日です
DB_BkDir = /home/dbmaster/backup	;	バックアップディレクトリ
DB BKSPM = 1	; æ	ららゆるストアドプロシージャをバックアップします

ストアドプロシージャのバックアップファイルはデフォルトとして、キー ワードDB_BkDirにより指定されたプロシージャを格納するディレクトリ の下のサブディレクトリにあります。ユーザーはdmconfig.iniファイルにて キーワードDB_BkOdrを設定する場合、完全バックアップを行う過程で、 前回のストアドプロシージャのバックアップシーケンスがSPというサブデ ィレクトリ(DB_BkOdrにより指定された古いバックアップディレクトリ の下にある)に移動され、その後、デフォルトのディレクトリからこれら のバックアップシーケンスを削除します。

ストアドプロシージャのバックアップ処理においては、バックアップサー バーはまずは*dmSpBk.his*という名前のファイルを生成し、それから、スト アドプロシージャのファイルをコピーし、その同時に、ファイル拡張子.s0 と.b0を使用するファイル及びバックアップしたストアドプロシージャの ロードに用いられるファイルが作成されます。ESQLストアドプロシージャ を及びSQLストアドプロシージャについては、バックアップする必要があ るファイルはソースファイル及びオブジェクトファイルで、JAVAストア ドプロシージャについては、バックアップする必要があるファイルはオブ ジェクトファイルのみです。ストアドプロシージャのリビルドが便利なよ うに、ESQLストアドプロシージャのソースファイルはspnameowner.ecのフ オーマットでリネームされ、SQLストアドプロシージャのソースファイル はspnameowner.spのフォーマットでリネームされます。例えば、ストアド プロシージャ名はy1で、そしてSYSADMに属する場合、コピーしたソース ファイル名はy1SYSADM.spにリネームされます。

ファイル**dmSpBk.his**はバックアップ情報の記録に用いられます。ストアド プロシージャのバックアップリストファイルの詳細については、「データ ベー*ス管理者参照編」*のセクション15.7のバックアップの履歴ファイルを ご参照ください。

データベース管理者は、オブジェクトファイルのバックアップ時に完全バ ックアップのために多くの時間を必要とすることを考えなければなりませ ん。完全な完全バックアップは下記の五つのことを含めます: (1) DB BkOdrが設定される場合、以前の完全バックアップをコピーする、

(2) あらゆるデータベースファイルのコピー、(3) あらゆるジャーナル ファイルのコピー、(4) DB_BkFoMが設定される場合、あらゆるオブジ ェクトファイルをコピーする、(5) DB_BKSPMが設定される場合、あら ゆるストアドプロシージャをコピーする。また、バックアップの失敗を防 ぐために、ファイルを完全バックアップをする時、DB_BkDirにより指定 されたバックアップディレクトリのディスクスペースが十分であることを 確保しなければなりません。

バックアップファイルの圧縮

データベースファイルは非常に大きくなることがあるので、バックアップ ファイルを保存するには大量の空きスペースが必要です。DBMasterは現在 バックアップファイルの圧縮をサポートしています。この機能を使用する と、dmconfig.iniにキーワードDB_BKZIPを設定することで、またはデー タベースが実行している中にシステムプロシージャSetSystemOptionを使用 してBKZIPも変更できます。

- **DB_BKZIP**=1:バックアップファイルを圧縮します。
- DB_BKZIP=0:バックアップファイルは圧縮されません。(デフォルト)

圧縮形式はGZIPなので、GZIP互換ツールを使えば、圧縮されたファイル の読み取りができます。

注 バックアップファイルを有効にするようにDB_BKZIPを設定しても、FO ファイルは圧縮されません。

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による と、dmSQLコマンドライン・ユーティリティやJServer Manager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 完全バックアップ或いは差分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 ックアップ或いは差分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JServer Manager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。JServer Managerを使ったバックアップ・モードの 変更方法については「*IServer Manager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。

Э 例1

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;

Э 例2

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。

しかし、実行中にユーザーはNONBACKUPをBACKUP-DATA-AND-BLOB モードに、または直接にBACKUP-DATA-AND-BLOBをBACKUP-DATAに変 更できません。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:
 - 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。
 - 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
 - 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。

- 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。

JServer Managerを使ってバックアップモードを設定する

データベースがオフラインの場合、JServer Managerグラフィックユーティ リティを使用してバックアップ・モードを変更することができます。 JServer Managerは、dmconfig.iniファイルのDB_BModeキーワードの値を自 動的に変更します。新しいバックアップモードは、次回にデータベースを 起動するときに使用されます。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、DB_BModeキーワードの値は有効になりませ ん。データベースのバックアップモードはNONBACKUPからBACKUP-DATAまたはBACKUP-DATA-AND-BLOBに、BACKUP-DATAからBACKUP-DATAまたはBACKUP-DATA-AND-BLOBに、BACKUP-DATAからBACKUP-DATAまたはBACKUP-DATA-AND-BLOBに、CO前にオフライン完全バック アップをしなければなりません。JServer Managerでバックアップモードを 設定する方法については、「JServer Managerユーザーガイド」を参照して 下さい。

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、ユーザーはDBA以上の権限を持つ必要です。

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:
 - 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。
 - 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。
 - 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。
 - 【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以上の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。

- [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。
- 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方法を提供していま す、データベースをバックアップする知識を了解する必要があります。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、増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アンラインアックアップサーバーだけ 実行できることを注意してください。データベースを起動した後、バック アップサーバーは起動できるからです。

JServer Managerツールの「バックアップサーバーでバックアップする」プロセスは実行されている場合に、データベース管理人もバックアップをすることができます。

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、増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、増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、ジャーナルファイルは頻繁に変更される からです。それで、バックアップ・サーバーはデータ(すべてのDBファ イルと.BBファイル)と有効なデータブロックだけコピーします。読み込 み専用表領域にデータファイルは差分バックアップ以外です。

差分バックアップは差分ベースを変更されたデータのみ含みます。普通 に、差分ベースは連続な差分バックアップに使用されます。リストアタイ ムで完全バックアップと差分バックアップはこの完全バックアップを基づ いて完全データベースを生成します。

増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、時間間隔と完全バ ックアップ以後の差分バックアップの最大数、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。

dmSQLは実行する中にSetSystemOptionストアドプロシージャを使用してバ ックアップを設定することができます。つまり、データベースが実行して いる中にSetSystemOptionストアドプロシージャを使用してBKSVR、 BKDIR、BKITV、DBKTV、BKTIM、BKFUL、BKFOM、BKZIP、 BKCMP、BKRTS、BKCHK、FBKTM、FBKTV、DBKMX、BKODR、 BKFRM を変更されることができます。

バックアップ・サーバーを起動する

バックアップサーバーはデッモンです。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。バックアップサーバーのライフサイクルはデータベースサ ーバと同じです。初期値は、バックアップ・サーバーを起動しません。バ ックアップサーバーはマルチユーザーモードだけで起動できます。

バックアップサーバーは二種類状態があります:インアクティブとアクテ ィブです。ユーザーはDB_BkSvrを通じてバックアップサーバーの状態を 制御することができます。DB_BkSvrの値が0の場合、バックアップサーバ ーはインアクティブにします、またどんなバックアップ要求を実行しませ ん。DB_BkSvrキーワードの値が1の場合、バックアップサーバーはアクテ ィブにします、また多様のバックアップ要求を実行してユーザーは任意の バックアップをすることができます。

バックアップサーバーをアクティブにするため、三つの方法があります: dmconfig.iniファイルでDB_BkSvrキーワードの値を1に設定すること、デ ータベースを起動した後 call setsystemoption('bksvr','1')でBkSvrを1に変更す ること、データベースが運行している場合Jserver Managerのランタイム設定 でバックアップ設定を変更することです。

バックアップサーバーでバックアップをする前に、ユーザーはバックアッ プを実行する方法を指定するため幾つのパラメータを設定する必要があり ます。例えば、バックアップディレクトリ、コンパクトバックアップモー ドなどです。

パラメータを設定する方法は以下の通りです:

- ユーザーはデータベースを起動する前にdmconfig.iniで関連キーワード を設定することができます。次回に、ユーザーはそのデータベースを 起動してから、バックアップサーバーはこれらのキーワードを使用し て関連のパラメータを初期化にします。
- データベースが起動された場合、ユーザーはパラメータの値を変更す るためにJserver Managerのランタイム設定を使用することができます。 その上に、ユーザーは*call SetSystemOption('option_name','value')*を使用 してパラメータを設定することができます。個別のパラメータは構文 の設定によって設定されることができます。例えば、バックアップオ フの設定、データのバックアップオンの設定、およびblobのバックア ップオンの設定。

バックアップサーバーはアクティブにされて、dmconfig.iniに適当のバッ クアップパラメータを設定する場合、システムストアドプロシージャ SetSystemOptionをコールして、バックアップを始めます。このストアドプ ロシージャはどんなクライアント或いはユーザーアプリケーションでも使 用できます。

Э 例

オンラインで完全、増分、差分バックアップをする構文: dmSQL> Call SetSystemOption('STARTBACKUP','1'); //完全バックアップをします dmSQL> Call SetSystemOption('STARTBACKUP','2'); //増分バックアップをします dmSQL> Call SetSystemOption('STARTBACKUP','3'); //差分バックアップをします

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、dmconfig.iniファイルのキーワードDB_BkSvr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:
 - データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
 - 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。
 - バックアップモードはBACKUP-DATAまたはBACKUP-DATA-AND-BLOBになることを確保します。DB_Bmodeを1に設定してBACKUP-DATAモードになり、2の場合BACKUP-DATA-AND-BLOBになりま す。
 - 4. 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。
 - 5. データベースを再起動します。

dmSQLを使ってバックアップ・サーバーを起動する

データベースがオンラインする場合、ユーザーはdmSQLコマンドラインに 以下のコマンドを使って動的なバックアップサーバーがアクティブさせる ことができます。

dmSQL> Call SetSystemOption('BKSVR','1');

ユーザーは*Call SetSytemOption('BKSVR', '1')*を使用してBkSvrを変更することができます。同時にdmconfig.iniファイルにDB_BkSvrの値を変更するため、ユーザーは*call setsystemoptionW('option', 'value')*を使用できます。

バックアップサーバーはアクティブにされる場合、dmconfig.iniファイル に適当的なパラメーターを設定されると、システムストアドプロシージャ SetSystemOptionをコールして、バックアップを実行させます。ストアドプ ロシージャはどんなクライアントとユーザーアアプリケーションでも使用 できます。

```
dmSQL> Call SetSystemOption('STARTBACKUP','1'); //完全バックアップをします
dmSQL> Call SetSystemOption('STARTBACKUP','2'); //増分バックアップをします
dmSQL> Call SetSystemOption('STARTBACKUP','3'); //差分バックアップをします
```

増分バックアップの時間間隔を変更する構文は: dmSQL> Call SetSystemOption('bkitv', 'Interval')

JServer Managerを使ってバックアップ・サーバーを起動 する

データベースがオンラインするかどうかとかかわらず、ユーザーはJServer Managerグラフィックユーティリティを使用してバックアップ・サーバー のランタイム設定をすることができます。JServer Managerはdmconfig.iniフ ァイルのキーワードDB_BkSvrの値を自動的に変更します。データベース がオフラインする場合、次回データベースを起動する際にバックアップサ ーバーも起動します。オンライン状態でJServer Managerでバックアップ・ サーバーを起動する方法については、「JServer Managerユーザーガイド」 をご参照下さい。

⇒ 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:

- メイン画面或いは [データベース] メニューから [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。
- [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. バックアップモードを選択します。
- もしデータだけをバックアップすると [データのみバックアップする] を選択します。
- データとBLOBファイルをバックアップすると [データとBLOBをバ ックアップする] を選択します。
- 9. [保存] ボタンをクリックします。
- **10. [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

以下のは差分バックアップのファイル名形式です:

DTimeStamp_DataFileName.dif(2) — 差分バックアップ識別、これは必要な オプションです。

TimeStamp — Timestampは1970(00:00:00 GMT)1月の秒数です。

DataFileName — データファイルが属するデータベース名。

.dif — 全ての差分バックアップファイル・オブジェクトには、拡張子 .dif が付きます。差分バックアップソースファイルが完全バックアップに存在 しないと、ファイルの拡張子は.dif2.でなければなりません。

2009/12/01 14:11にはじめの差分バックアップを実行した後、また差分バッ クアップファイル名を生成します。ファイル名は D1259647860_DBNAME.BB.dif, D1259647860_DBNAME.DB.dif, D1259647860_DBNAME.SBB.dif とD1259647860_DBNAME.SDB.difです。ジ

ャーナルファイル名はD1259647860_DBNAME.JNLです。

増分バックアップのファイル名の形式を設定する

バックアップのファイル名形式は<I><TimeStamp><_><DB_BKFRM>,で、例
 えば、I1234567890_%2F%4N%4B.JNL。ファイル名の長さは256文字を超
 えません。タイムスタンプはシステムの10桁の有効数値で、
 <DB BKFRM>はテキスト定数とフォーマットシーケンスを含みます、

(例、エスケープシーケンス)、これは特別な文字ストリングを表します。

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。 %[x]F—完全バックアップID、変数は以下の四つのフォーマットがありま す:

1:完全バックアップのフォーマット:YYYYMMDD、例えば: 20010917

2:完全バックアップのフォーマット:MMDD、例えば:0917

3:完全バックアップのフォーマット:MMDDhhmm、例えば:09171305

4:完全バックアップのフォーマット:DDhhmmss,、例えば:17130558

%[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、単一の数値または有 効なフォーマット文字が続きます。こ%記号の後にそれ以外の文字がある 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 ーメッセージを返します。

長さn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 定する整数です。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[n]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、長さ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、F、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、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_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。
 - 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256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。

dmSQLを使ってバックアップファイル名のフォーマッ トを設定する

データベースが実行している中にdmSQL SetSystemOptionストアドプロシ ージャを使用してバックアップファイル名のフォーマットが変更できま す。普通の構文は:

dmSQL> Call SetSystemOption('bkfrm', 'name');

Э 例

バックアップファイル名のフォーマットはI1234567890_%2F%4N%4B.JNL に変更すると、dmSQLコマンドプロンプトに以下のラインを入力してくだ さい。

dmSQL> Call SetSystemOption('bkfrm', 'I1234567890 %2F%4N%4B.JNL');

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:
 - 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。DBMasterはシングルバックアップファイルのパ スと複数のバックアップファイルのパスをサポートします。バックアップ サーバーは自動にBkDirを作成します。しかし、バックアップ・ディレク トリを、データベースとは一つ以上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、256字以内でなければなりません。

ただし、DBMasterデータベースはレプリケーションモード(マスター或い はスレーブ)に実行する場合、シングルBkDirディレクトリだけ使用でき ます。BKDIRをマルチパスを設定する場合、最初のは使用されてパスサイ ズが無視になります。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。データベースはオンラインするかオフラインするかによってdmSQL コマンドツールで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。

- データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
- 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。
- 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プロシージャを使用すると、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、256文字以下です。

Э 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、ランタイム設定にバックアップ・ディレクトリを直 ちに変更することもできますが、データベースは対話型のバックアップを する場合、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。
 - [ランタイムの設定]から[バックアップ]ウィンドウが表示され ます。
 - **3.** [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。
 - 4. [ログイン] ダイアログが表示します。
 - 5. [OK] をクリックして、ログインするデータベースは [ランタイム の設定] ダイアログの [データベース名] にあります。
 - 6. 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。
 - **7.** 更新した変更を既存だけに適用する場合、 [dmconfig.iniに保存] の 欄をチェックしません。
 - 8. [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。
 - **9.** [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、一つのディレクトリまたは一グループのディレクト リ(32以下)です。これは最後のバックアップシーケンスの前のバックア ップシーケンスを保存するため使用されます。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・サーバーは以前のバックアッ プ・ファイルを捨てます。

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。

dmSQLを使って古いディレクトリを設定する

データベースが実行している中にdmSQL SetSystemOptionストアドプロシ ージャを使用して古いディレクトリを設定することができます。当該コマ ンドの一般的な構文は下のようになります: dmSQL> Call SetSystemOption('bkodr', 'path')

*Path*は新規な古いバックアップディレクトリの完全パスです。*Path*にスト リングの長さは256文字を超えることが許可しません。

Э 例

古いディレクトリのパスを*E:/storage/database/backup/WebDB*に変更すると、 dmSQLコマンドプロンプトに以下のラインを入力してください。 dmSQL> Call SetSystemOption('bkodr', 'E:/storage/database/backup/WebDB');

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、次回の増分バックアップまでの時間を表しています。

完全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、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_DBKTV**で指定しま す。始めのバックアップは**DB_FBKTM + DB_DBKTV**で終了されます。キ ーワード**DB_DBKTV**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。

完全バックアップ以後の差分バックアップの最大数はキーワード DB_DBKMXで指定します。差分バックアップの数は完全バックアップ以 後でDB_DBKMXを超える場合、バックアップサーバーは古い差分バック アップを削除します。

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を使った差分バックアップスケジュールを設定:

- データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
- 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_FBKTMの値を指定します。
- ndays-HH:MM:SS形式を使って、時間間隔のキーワードDB_DBKTV 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

バックアップサーバーを起動するためSetSystemOptionプロシージャが使用 できます。普通の構文は:

dmSQL> CALL SETSYSTEMOPTION('BKSVR','1');

バックアップサーバーを起動した場合、システムストアドプロシージャ SetSystemOptionをコールして、バックアップサーバーに差分バックアップ を実行させます。 dmSQL> Call SetSystemOption('STARTBACKUP','3')

差分バックアップの時間間隔を変更する構文は: dmSQL> Call SetSystemOption('dbktv', '*Interval*')

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グラフィカルユーティ リティを使用して差分バックアップのスケジュールを変更することができ ます。JServer Managerは、自動的にdmconfig.iniファイルのキーワード DB_FBKTMとDB_DBKTV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・サーバーはこれらの設定を新しい差分バックアッ プ・のスケジュールとして使用します。データベースがオンラインの場 合、次回にデータベースが再起動するまで、JServer Managerは直ちにバッ クアップのスケジュールを変更するまたは必要に応じて当該変更を遅らせ ることができます。JServer Managerで差分バックアップの時間スケジュー ルを設定する方法については、「JServer Managerユーザーガイド」をご参 照下さい。

増分バックアップ設定

増分バックアップ・スケジュールは、バックアップ・サーバーがオンライ ン増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 バーが最初の増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、次回の増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、増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。

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。

DBMasterには、増分バックアップ・スケジュールを設定する方法がいくつ かあります。データベースはオンラインするかオフラインするかによって 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を使った増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、増分バックアップ・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:
 - 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。
 - 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。
- 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を使った増分バックアップ設定の変更

SetSystemOptionプロシージャを使用すると、データベースの起動中に増分 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。増分バッ 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')

増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、増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、増分バックアップが 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。

バックアップサーバーを起動した場合、システムストアドプロシージャ SetSystemOptionをコールして、バックアップサーバーに増分バックアップ を実行させます。

dmSQL> Call SetSystemOption('STARTBACKUP', '2')

Э 例

dmSQLコマンド・プロンプトに次のように入力し、増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を使った増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グラフィックユーティ リティを使用して増分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ファイルのキーワード DB_BkTimとDB_BkItvの値を変更します。次回にデータベースが起動する 時、バックアップ・サーバーはこれらの設定を新しい増分バックアップの スケジュールとして使用します。データベースがオンラインの場合、次回 にデータベースが再起動するまで、JServer Managerは直ちにバックアップ のスケジュールを変更するまたは必要に応じて当該変更を遅らせることが できます。JServer Managerで増分バックアップを設定する方法について は、「JServer Managerユーザーガイド」をご参照下さい。

- 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. [増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. [増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。
- 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.** [**ランタイムの設定**] ウィンドウが表示されます。
 - **3.** [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。
 - 4. [ログイン] ウィンドウが表示されます。
 - 5. [OK] をクリックして、ログインするデータベースは [ランタイム の設定] ダイアログの [データベース名] にあります。

- 6. 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。
- **7.** 更新した変更を既存だけに適用する場合、 [dmconfig.iniに保存] の 欄をチェックしません。
- **8.** 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。
- 9. 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。
- **10.** [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増分バックアップ 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90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。 データベースがオンラインするかオフラインするかによって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。

- - データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
 - 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プロシージャを使用すると、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グラフィックユーティリティ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、次回にデータベースが起動 する場合、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場合、次回にデータベースが再起動するまで、 JServer Managerは直ちにジャーナルトリガーの値を変更するまたは必要に 応じて当該変更を遅らせることができます。JServer Managerでジャーナ ル・トリガー値を設定する方法については、「JServer Managerユーザーガ イド」をご参照下さい。

- ⇒ 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. ジャーナル・ファイルが特定の割合に達した時に、増分バックアップ 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増分バックアップを実行させる場合、「増分バックアップの起動イベント」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増分バックア ップを実行させる場合、[%一杯です]の欄に50~100の範囲内 の値を入力します。
 - **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。
- 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. [ランタイムの設定] ウィンドウが表示されます。
 - 3. [データベース名] で、設定するデータベース名を選びます。

- 4. [ログイン] ウィンドウが表示されます。
- [OK] をクリックして、ログインするデータベースは [ランタイム の設定] ダイアログの [データベース名] にあります。
- 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。
- **7.** 更新した変更を既存だけに適用する場合、 [dmconfig.iniに保存] の 欄をチェックしません。
- ジャーナル・ファイルが特定の割合に達した時に、自動的に増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増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増分バックア 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。
- **9.** [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 イン増分、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。データベースはオンラインするかオフラインするかにようて 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:

- データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
- 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。
- データベースを再起動します、新しいコンパクト・バックアップ・モードを有効にします。

dmSQLを使ってコンパクト・バックアップモードの設 定

データベースが実行している中にdmSQL SetSystemOptionストアドプロシ ージャを使用してコンパクト・バックアップモードを変更することができ ます。成功的なバックアップはジャーナル・ファイル内にすべてのジャー ナル・ブロックがある必要としません。キーワード DB_BKCMP を1に設定 すると、バックアップサーバーはジャーナルブロックのみバックアップし ます、構文は以下のようになります:

dmSQL> Call SetSystemOption('bkcmp', '1');

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場合、JServer Managerグラフィカルユーティ リティ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、次回にデータベースが 起動する場合、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場合、次回にデータベースが再起 動するまで、JServer Managerは直ちにコンパクトバックアップモードを変 更するまたは必要に応じて当該変更を遅らせることができます。JServer Managerでコンパクトバックアップモードを設定する方法については、 *「JServer Managerユーザーガイド」を*ご参照下さい。

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。
- [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。
- 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.** [ランタイムの設定] ウィンドウが表示されます。
 - **3.** [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。
 - 4. [ログイン] ウィンドウが表示されます。
 - [OK] をクリックして、ログインするデータベースは [ランタイム の設定] ダイアログの [データベース名] にあります。
 - 6. 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。
 - **7.** 更新した変更を既存だけに適用する場合、 [dmconfig.iniに保存] の 欄をチェックしません。
 - 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。
 - **9.** [OK] をクリックします。

完全バックアップのスケジュール

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\或いは増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。

最後、キーワード DB_BKCHK は完全バックアップと差分バックアップす る前にデータベースをチェックするかを指定します。キーワード DB_BKRTS は完全バックアップを実行する場合にバックアップサーバー が読み取り表領域ファイルを含むかどうか指定します。この二つの機能を 使用すると、ユーザーはdmconfig.iniにキーワードDB_BKCHK と DB_BKRTS を設定できます、またはデータベースが実行している中にシ ステムプロシージャSetSystemOption を使用してDB_BKCHK と DB_BKRTS を変更します。

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ファイルを使って完全バックアップ・スケジュールを設定
 する:
 - データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
 - 完全バックアップ・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。
 - 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。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。
 - データベースを再起動して、新しい完全バックアップ・スケジュール を有効にします。

dmSQLを使って完全バックアップ・スケジュールを設 定する

データベースが実行している場合に完全バックアップの起動時間と時間間 隔を変更するためプロシージャSetSystemOptionが使用されます。完全バッ クアップの起動時間を変更する構文は下のようになります: dmSQL> Call SetSystemOption('fbktm', 'StartTime');

完全バックアップの時間間隔を変更する構文は下のようになります: dmSQL> Call SetSystemOption('fbktv', '*Interval*');

Start Time は初めての完全バックアップの起動時間です。フォーマットは YY:MM:DD HH:MM:SSです。 *Interval* は完全バックアップの発生間隔で す、フォーマットはD-HH:MM:SSになります。

バックアップサーバーは起動した後、システムストアドプロシージャ SetSystemOptionをコールして完全バックアップが始めます。 dmSQL> Call SetSystemOption('STARTBACKUP','1');

Э 例

完全バックアップの時間間隔は一時間を設定すると、dmSQLコマンドプロ ンプトに以下のラインを入力してください:

dmSQL>Call SetSystemOption('fbktv', '0-1:00:00');

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

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。

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、これはデータベースにいくつのファイルオ ブジェクトがあるか依存します。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, fo000000000.bak, "/DBMaster/mydb/fo/ZZ000001.bmp"
u, fo000000001.bak, "/home2/data/image.jpg"
....
s, fo0000002345.bak, "/DEMaster/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。

Э 例1

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プロシージャを使用すると、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。

Э 例2

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

ファイルオブジェクトのバックアップモードの設定は、完全バックアップ をしている間にファイルオブジェクトのコピー方法に影響を与えます。フ ァイルオブジェクトをバックアップしないモードを選択する場合には、フ ァイルオブジェクトがコピーできません。ファイルオブジェクトのみバッ クアップするモードを選択する場合には、ファイルオブジェクトが自動完 全バックアップのプロセスでコピーされます。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するモードを選択する場合には、システム ファイルオブジェクト及びユーザーファイルオブジェクトが両方とも自動 完全バックアップのプロセスでコピーされます。

- 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:
 - メイン画面の [データベースの起動] の [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。
 - **2.** [データベース起動の高度な設定] ウィンドウの [バックアップ] タ ブをクリックします。
 - [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。
 - 4. 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) [完全バックアップの開始日時]の欄に、日付と時間を入力して下さい。表示して下さい。
 - c) [完全バックアップの間隔]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。
 - 5. 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、**[FO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FO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FOとユーザーFOをバックアップ]を選択します。
- 6. [保存] ボタンをクリックします。
- [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、[スタート] ボタンをクリックしてデータベースを起動し ます。

ストアドプロシージャのバックアップモード

ストアドプロシージャのバックアップモードは、データベース管理者がバ ックアップサーバが完全バックアップ時にストアドプロシージャをバック アップするかどうかを決定することができます。

ストアドプロシージャのバックアップモードの設定方法は幾つかありま す。

データベースがスタートする場合、キーワード**DB_BKSPM**によってその 設定方法を決めることができますが、dmSQLまたはJServer Managerツール で実行している間に、それが変更されることがあります。

バックアップサーバに通して、前回のバックアップを**DB_BkOdr**により指 定された古いバックアップディレクトリに移動することができます。

ストアドプロシージャのバックアップを開始すると、データベースは完全 バックアップを完了するために多くの時間を必要とするようになります。 この時間はデータベース内のファイルオブジェクトの数によって決めま す。

完全な完全バックアップは下記の五つのことを含めます:(1) DB BkOdrが設定される場合、以前の完全バックアップをコピーする、

(2) あらゆるデータベースファイルのコピー、(3) あらゆるジャーナル ファイルのコピー、(4) **DB_BkFoM**が設定される場合、あらゆるオブジ ェクトファイルをコピーする、(5) **DB_BKSPM**が設定される場合、あら ゆるストアドプロシージャをコピーする。また、バックアップの失敗を防 ぐために、ファイルを完全バックアップをする時、DB_BkDirにより指定 されたバックアップディレクトリのディスクスペースが十分であることを 確保しなければなりません。

ストアドプロシージャが完全バックアップを行う際にバックアップディレ クトリにて作成されたサブディレクトリSPにコピーされます。その場合、 ファイルオブジェクトがリネームされます。コピーされたストアドプロシ ージャのバックアップ情報はファイル**dmSpBk.his**に記録されます。

バックアップストアドプロシージャの情報ファイル

バックアップ情報は、データベース名、バックアップ時間及びバックアッ プされた行の情報記録用のバックアップリストで構成します。バックアッ プ情報ファイルdmSpBk.hisはDB_BkDirによって指定されたディレクトリ の下にあるサブディレクトリSPにて作成されます。このファイルはコピー されたストアドプロシージャのバックアップ情報を記録する純粋なASCII テキストファイルで、そのフォーマットは下記のようになります。

Database Name: MYDB Begin Backup SP time: 2014/06/20 09:13:06 [Backup List] SQLSP, SYSADM, SP1, SP1SYSADM.luo, SP1SYSADM.sp ESQLSP, SYSADM, A4, A4SYSADM.dll, A4SYSADM.ec

JAR, "", "", employee.jar, ""

「バックアップリスト」前の内容はただユーザー参照用の説明です。「バ ックアップリスト」後の各ラインはそれぞれ一つのレコードを表します。 当該レコードにストアドプロシージャのタイプ、ストアドプロシージャの 所有者、ストアドプロシージャ名、ストアドプロシージャのオブジェクト ファイル名及びそのソースファイル名を含めます。このバックアップ情報 ファイルはストアドプロシージャの回復のために必要であります。

dmconfig.iniファイルを使ってストアドプロシージャのバ ックアップモードを設定する

ファイルのキーワード**DB_BKSPM**は、ストアドプロシージャのバックア ップモードを決定します。

- DB_BKSPM=0:ストアドプロシージャをバックアップしません
- DB_BKSPM=1:あらゆるストアドプロシージャをバックアップします

Э 例

指定されたストアドプロシージャのバックアップパラメータは dmconfig.iniファイルのエントリは以下のようになります。

[MyDB]	
DB_BkSvr = 1	; バックアップサーバを起動します
DB_FBKTm = 14/05/01 00:00:00	; 2014年5月1日の午前零時に開始
DB_FBKTV = 1-00:00:00	;時間間隔は毎日一回です
DB_BkDir = /home/dbmaster/backup	; バックアップディレクトリ
DB BKSPM = 1	; あらゆるストアドプロシージャをバックアップします

上記の例では、**DB_BKSPM**の値は1であるため、バックアップサーバがあ らゆるストアドプロシージャを**DB_BkDir**によって指定されたディレクト リの下にあるサブディレクトリ**SP**にバックアップします。サブディレクト リ**SP**が存在しない場合は、バックアップサーバーはそれを作成します。

dmSQLを使ってストアドプロシージャのバックアップ モードを設定する

データベースが動作している間に、プロシージャSetSystemOptionはストア ドプロシージャのバックアップモードの変更に用いられます。ファイルオ ブジェクトのバックアップモードを変更する一般的な構文は以下のように なります。

dmSQL> Call SetSystemOption('BKSPM', 'n');

上記の例では、ユーザーはnに0または1を割り当てることができます。0を 割り当てる場合は、バックアップサーバはストアドプロシージャをバック アップしていませんが、1を割り当てる場合は、バックアップサーバは完 全バックアップを行っている間にあらゆるストアドプロシージャをバック アップしています。

Э 例

バックアップサーバを設定することによって、あらゆるストアドプロシー ジャで完全バックアップをします。dmSQLコマンドプロンプトに下記のコ マンドを入力します。

dmSQL> Call SetSystemOption('BKSPM', '1');

JServer Managerツールを使ってストアドプロシージャの バックアップモードを設定する

データベースがオンラインするかどうかとかかわらず、ユーザーはJServer Managerグラフィックユーティリティを使用してバックアップストアドプ ロシージャを設定することができます。JServer Managerは**dmconfig.ini**ファ イルのキーワード**DB_BKSPM**の値を自動的に変更します。データベース がオフラインする場合、次回データベースを起動する際にこの新しい設定 が有効になります。データベース起動時にまたはJServer Managerのランタ イム設定でストアドプロシージャのバックアップモードを設定する方法に ついては、「*IServer Managerユーザーガイド」*をご参照下さい。

バックアップ・サーバーをインアクティブにする

データベースは起動した後、DBMasterは自動にバックアップ・サーバーを 起動します。バックアップサーバーの初期値はインアクティブにします。 ユーザーはDB_BkSvrキーワードの値を設定してバックアップサーバーの状 態を変更します。DB_BkSvrキーワードの値を0にしてインアクティブにし ます;1にしてアクティブにします。バックアップサーバーをインアクテ ィブにするため、DB_BkSvrキーワードの値を0にして、或いは*call* setsystemoption('bksvr','0')を通じてDB_BkSvrキーワードの値を変更します。

dmconfig.iniを使ってバックアップ・サーバーをインアク ティブにする

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。

- Immediate and a set of the se
 - データベース・サーバーで、ASCIIテキスト・エディタを使って dmconfig.iniファイルを開きます。
 - 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。

dmSQLを使ってバックアップ・サーバーをインアクテ ィブにする

データベースが動作している間に、プロシージャSetSystemOptionはバック アップサーバのステータスの変更に用いられます。バックアップサーバの ステータスを変更する一般的な構文は以下のようになります。 dmSQL> Call SetSystemOption('bksvr', 'n');

nは0または1に設定可能です。0に設定する場合は、バックアップサーバ をアクティブにしないを表し、1に設定する場合は、バックアップサーバ をあくてぃぶにするのを表します。

Э 例

バックアップサーバをインアクティブにするには、dmSQLコマンドプロン プトに下記のコマンドを入力します。

dmSQL> Call SetSystemOption('bksvr', '0');

JServer Managerを使ってバックアップ・サーバーをイン アクティブにする

データベースがオフラインの場合、JServer Managerグラフィカルユーティ リティ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、次回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場合、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。JServer Managerでバックアップ・サーバーのインア クティブにする方法については、「JServer Managerユーザーガイド」をご 参照下さい。

- オフライン時にJServer Managerでバックアップ・サーバーをインアクテ ィブにする:
 - **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

バックアップ・サーバーの自動バックアップ機能を使用してバックアップ ジャーナルファイル、バックアップ時間とバックアップディレクトリなど の情報を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

バックアップ履歴ファイルはテキストファイルで、dmconfig.iniファイルの キーワード**DB_BKDIR**の第一ディレクトリに置いています。バックアップ 履歴ファイルはオンラインバックアップディレクトリに作成されて *dmBackup.his*という名前で記録されます。データベースのリストアの際に 自動的に使用されます。但し、オフラインバックアップは*offBackup.his*と いう名前で記録されます。

バックアップ履歴ファイルを理解する

バックアップ履歴ファイルには、ID番号のほか、ファイル名、バックアッ プした日付、時刻に分けられた全ての情報があります。DBMasterでは、バ ックアップ履歴ファイルを使って、バックアップの順序をたどり、各段階 で完全、差分バックアップと増分バックアップの整合性を取ります。

⊃ 例

<backup id>: file name -> archive file name, time, event

上記のコマンドはイベントの原因でfile_nameというファイルを archive_file_nameというアーカイブファイルにコピーされたことを意味 します。<イベント>はバックアップの理由を示し、JOURNAL-FULL、 TIME-OUT、USER、ON-LINE-FULL-BACKUP-BEGIN、ON-LINE-FULL-BACKUP、ON-LINE-FULL-BACKUP-END、のいずれかです。JOURNAL-FULLは、ジャーナルが一杯になったので増分バックアップが行われたこ とを示します。TIME-OUTは、スケジュールに定めた予定時刻がきたので 差分或いは増分バックアップが実行されたことを示します。USERは、ユ ーザーによる手動増分バックアップを表します。ON-LINE-FULL-BACKUPxxxxは、完全バックアップを意味します。

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。これを避けるため、 ユーザーはバックアップの時間間隔を短くにする必要があります。

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。

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, fo000000000.bak, "/DEMaster/mydb/fo/ZZ000001.bmp" u, fo000000001.bak, "/nome2/data/image.jpg" s, fo0000002345.bak, "/DEMaster/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

レプリケーションデータベースにバック アップする

スレーブデータベース以外の普通データベースとマスターでは、ユーザー は完全バックアップ、差分バックアップ、および増分バックアップをする ことができます。方法は以前と同じです。しかし、JServerManagerは、マス ターデータベースでインタラクティブに増分バックアップをすることがで きません。さらに、マスターデータベースでインタラクティブに完全バックアップをする場合、増分バックアップを消去することができません。

マスターデータベースでは、潜在的に、完全バックアップより先に多くの 増分バックアップファイルがまだレプリケーションためにバックアップシ ーケンスに残されていることを注意してください。同時に、完全バックア ップのために、レプリケーションサーバーは増分バックアップファイルを 消去しない可能性があります。それで、長い期間に次の完全バックアップ が実行されていない場合、大量のファイルがDB_BKDIRに存在していま す。

ー言で、レプリケーションサーバーはバックアップサーバーとよく協力す る必要があって、彼らはお互いに妨げることができません。一方では、バ ックアップはレプリケーションを壊すことができません。言い換えるな ら、完全、差分バックアップをする中或いは終了した場合、レプリケーシ ョンサーバーはスレーブサイトにすべてのトランザクションを複写するこ とができます。つまり、レプリケーションはバックアップシーケンスを壊 すことができません。

バックアップシーケンスを使用してマスターデータベースがリストアできます。ただし、リストアした後、レプリケーションデータベースは終了します。ユーザーはレプリケーションデータベースを続くと、すべてのスレ ーブデータベースが新しいマスタデータベースを代わらなければなりません。つまり、ユーザーはマスタデータベースファイルをコピーして全部のスレーブファイルを代わります。

レプリケーションデータベースにいくつの制限があります:

- マスタデータベースが起動する場合、dmconfig.iniファイルのBMODE とBKSVRは起動状態を設定します。
- マスターとスレーブデータベースのランタイムにBMODE、BKSVR、 BKDIRを変更することはできません。例えば、 setsystemoption('bkdir','new-bkdir')をコールしてエラーを返します。
- マスターとスレーブデータベースサイトに、BKDIRキーワードがシン グルパスになるべきです。ユーザーはマルチパスを設定すると、第一 番のパスを使用されて、パスサイズが無視になります。

- マスターデータベースにJServerManagerで増分バックアップをすること ができません。
- スレーブデータベースに、完全、差分と増分バックアップをすること ができません。

9.9 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れ以降の差分と増分バックアップが何処にあ るかを見つけます。例えば、毎月30日に完全バックアップを取り、毎月15 日に差分バックアップを取り、10日毎に増分バックアップを取るとしま す。5月25日にシステムが損傷したならば、4月30日の完全バックアップ と、5月15日の差分バックアップと、5月20日の増分バックアップと、5月 25日の損傷ジャーナルファイルが必要になります。これらのファイルを見 つけると、DBMasterは、5月25日の障害の前の状態にデータベースをリス トアすることができます。フルバックアップファイルと連なった差分ファ イルと増分ファイルで構成された正確なバックアップ順はリストアの必要 不可欠です。オンラインバックアップ順はdmbackup.hisというファイル名 のバックアップ履歴に記録され、オフラインバックアップ順は offbackup.hisというファイル名のバックアップ履歴に記録され、このバッ クアップ履歴ファイルは非常に重要です。DBMasterは、データベースをリ ストアするときに、その情報を読み込みます。

リストアの実行

リストアプロセス時、DBMasterは以下の動作を行います:

- データファイル、BLOBファイル、ジャーナルファイルを含む完全バックアップファイルをdmconfig.iniファイルのDB_DBDIRで指定されたディレクトリにコピー。このオペレーションは元のファイルを上書きします。そのためリストア実行前に元のデータベースファイルを別箇所へのコピーにて退避しておくことが望ましいです。少なくともジャーナルファイルが保存されていることを確認し、リストアに失敗した場合でもデータベースを現状に近い状態まで戻す可能性を残せます。
- 差分と/或いは増分バックアップファイルをデータベースに適用

リストアツール使用時の指定可能設定。

- dmconfig.iniのデータベースセクションをリストアするかの指定。リストアする場合、リストアされるdmconfig.iniのフルパスを指定。
- ユーザーはバックアップ順で普通の位置と違った位置にデータベース をリストアしたい場合、データファイル"path"、例えばDB_DBDIR, DB_DBFIL, DB_USRDBなどに、或いはデータファイルとblobファイル のパスに再びキーワードを設定するべきです。もしバックアップ順は 他の場所または新しいコンピュータに移動されると、データベースを リストアする場合、ユーザーは以下の状況をお考えください:

- a) 構成セクションは新規なデータベースのdmcnfig.iniファイルにあ ると、データベース名はバックアップデータベース名と一致にし なければなりません。
- b) 新規なキーワード (BKDIR或いは他のデータファイルとblobファ イルのキーワード)を設置します。
- c) 一つ以上のjnlファイルがあると、ユーザーはDB_JNFILの値をセ ットして、jnlファイルのナンバーはバックアップデータベースの jnlファイルと同じでなければなりません。
- d) バックアップファイルは複数のフォルダーにあると、ユーザーは キーワードDB_BKDIR(全てのバックアップファイルのフォルダ ーを含み、dmbackup.hisファイルは始めのBKDIRだとのことを確 保する)を設置しなければなりません。
- 注 ユーザーはバックアップ順があるフォルダーから既存のdmconfig.ini ファイルをコピーすることができます。関連キーワードを変更する ことで、新規のdmconfig.iniファイルも設置できます。
- バックアップ履歴ファイルdmBackup.his或いは offBackup.hisのオンラインまたはオフラインのフルパス。dmBackup.his 或いはoffBackup.hisがデフォルトのディレクトリに置かれていない場合に指定。
- リストアタイム(RTime) RTimeはデータベースがリストアされる時間を 表示し、バックアップ順が適切であるか判断し、差分と増分バックア ップファイルをデータベースに適用します。リストアツールまたはシ ステムdmconfig.iniのキーワードDB_RTimeにて設定するかリストアさ れるバックアップdmconfig.iniに指定します。RTimeが定義されないと デフォルトの値として現在の時刻が使用されます。

DBMasterはリストアに対してJServer Manager Toolとコマンドラインツール 上でのRolloverによる2種類の方法を用意しています。

JServer Managerの使用については「*JServer Managerユーザガイド」*をご覧く ださい。ロールオーバーコマンドラインツールに関しては次のセクション *Rolloverによるデータベースリストア*をご覧ください。

Rolloverによるデータベースリストア

Rolloverコマンドラインツールをデータベースのリストアに使用できま す。方式はJServer Managerによるデータベースのリストアのものと同じで す。

Rollover使用方法:

rollover database_name [-i inifile] [-r rtime] [-h hisfile] [-m foMapfile] [-f FOtype] [-t SpResTyp]

六つの鉤括弧のオプション引数があります:

-i dmconfig.iniのフルパスを指定。dmconfig.iniのリストアを指定した場合、rolloverはdmconfig.iniの指定に対応したシステムdmconfig.iniのデータベースセクションを置き換えます。そうでなければdmconfig.iniはリストアされません。

-r データベースがリストアされる時間を示します。-rオプションはRTime を設定する一つ目の方法で、キーワードDB_RTimeをシステムdmconfig.in またはリストアデータベースを設定するバックアップdmconfig.iniに定義 する方法があります。-rオプションとDB_RTimeのどちらも定義されてい ない場合は現在の時刻がRTIMEとして使用されます。

-h dmBackup.his或いはoffBackup.hisの完全パスを指定する。初期値は "DB_BkDir/dmBackup.his"又は"DB_BkDir/offBackup.his"です。

-m dmFoMap.hisの完全パスを指定する。初期値は "DB_BkDir\FO\dmFoMap.his"です。

-f FOファイルのユーザ/システムでどちらをリストアするかを指定しま す。このオプションには4つの設定値があります。0はFOファイルをリス トアしない;1はシステムFOをリストア;2はユーザFOを、3は初期値で全て のFOをリストアします。

 -t ストアドプロシージャをリストアするかどうかを指定します。このオ プションには二つの値があります。0はリストアする必要があるストアド プロシージャがないことを指しますが、1はあらゆるストアドプロシージ ャをリストアすることを指します。ディフォルト値は1です。

