DB2® ユニバーサル・データベース

データ移動ユーティリティー 手引きおよび解説書

バージョン 7

SC88-8522-00
(英文原典：SC09-2955-00)

DB2® ユニバーサル・データベース

データ移動ユーティリティー 手引きおよび解説書

バージョン 7

SC88-8522-00
(英文原典：SC09-2955-00)

ご注意!

本書、および本書がサポートする製品をご使用になる前に、 315ページの『付録F. 特記事項』にある一般的な情報を必ずお読みください。

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

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

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

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

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

(URL は、変更になる場合があります)

原 典:	SC09-2955-00 IBM® DB2® Universal Database Data Movement Utilities Guide and Reference Version 7
発 行:	日本アイ・ビー・エム株式会社
担 当:	ナショナル・ランゲージ・サポート

第1刷 2000.6

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

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

目次

本書について vii
本書の対象読者 vii
本書の編成 viii

第1章 エクスポート 1
エクスポートの概要 2
エクスポートの使用に必要な特権、権限、および許可 3
エクスポートの使用 3
エクスポートの使用の前に 3
エクスポートの起動 3
エクスポートでの識別列の使用 4
エクスポートされた表の再作成 4
ラージ・オブジェクト (LOB) のエクスポート 5
データの並列エクスポート 5
EXPORT コマンド 8
エクスポート API 11
SQLUEXPT-OUT データ構造体 17
ファイル・タイプ修飾子 (エクスポート) . . 18
区切り文字に関する制限 19
エクスポート・セッションの例 21
CLP の例 21
API の例 22
制約事項 25
トラブルシューティング 26

第2章 インポート 27
インポートの概要 28
インポートの使用に必要な特権、権限、および許可 29
インポートの使用 29
インポートを使用する前に 29
インポートの起動 29
クライアント / サーバー環境でのインポートの使用 30
バッファー挿入によるインポートの使用 . . . 31
インポートでの識別列の使用 31
インポートでの生成列の使用 33
エクスポートされた表の再作成 35
ラージ・オブジェクト (LOB) のインポート 36
ユーザー定義特殊タイプ (UDT) のインポート 37
インポート時の表のロッキング 37
IMPORT コマンド 38
インポート API 46
SQLUIMPT-IN データ構造体 57
SQLUIMPT-OUT データ構造体 58
ファイル・タイプ修飾子 (インポート) . . . 60
文字セットと NLS についての考慮事項 . . . 70
インポート・セッションの例 70
CLP の例 70
API の例 72
インポートのパフォーマンスの最適化 . . . 72
制約事項と制限 72
トラブルシューティング 73

第3章 ロード 75
ロードの概要 76
バージョン 6 とバージョン 7 で導入された以前のロード動作の変更 81
並列性とロード 82
ロードの使用に必要な特権、権限、および許可 83
ロードの使用 84
ロードを使用する前に 84
ロードの起動 85
ロードでの識別列の使用 86
ロードでの生成列の使用 88
制約違反の検査 90
中断したロード操作の再開 91
ロード・コピー・ロケーション・ファイルの使用 92
LOAD コマンド 94
LOAD QUERY コマンド 109
ロード API 111
データ構造体: SQLULOAD-IN 121
データ構造体: SQLULOAD-OUT 126
db2LoadQuery - ロード照会 API 128
ファイル・タイプ修飾子 (ロード) 133
例外表 147
ダンプ・ファイル 148
ロード一時ファイル 148
ロード・ユーティリティーのログ・レコード 149

文字セットと各国語のサポート 149
ロード・セッションの例 150
CLP の例 150
API の例. 156
ロード操作後の保留状態 161
ロードのパフォーマンスの最適化 162
制約事項と制限 167
トラブルシューティング 168

第4章 オートローダー 169
オートローダーの概要 169
オートローダーの使用に必要な特権、権限、および許可 170
オートローダーの使用 171
オートローダーを使用する前に 171
オートローダーの起動 171
複数のデータベース区画へのロード. . . . 172
オートローダーのオプション 173
オートローダー・セッションの例 181
移行およびバック・レベル互換性 183
オートローダーに関するヒント 184
制約事項と制限 186
オートローダーのトラブルシューティング 186

第5章 DB2 データ・リンク・マネージャーのデータの移動 189
DB2 データ・リンク・マネージャーのデータの移動 - エクスポートの使用. 190
DB2 データ・リンク・マネージャーのデータの移動 - インポートの使用 194
DB2 データ・リンク・マネージャーのデータの移動 - ロードの使用 194

第6章 システム間のデータの移動 197
プラットフォーム間のデータの移動. . . . 197
PC/IXF ファイル形式 197
区切り付き ASCII (DEL) ファイル形式 198
WSF ファイル形式 199
db2move ツールを使用したデータの移動 . . 200
DB2 コネクトによるデータの移動 205
エクスポート・ユーティリティーおよびインポート・ユーティリティーの使用. . . 205
タイプ表間のデータ移動 207
走査順序. 208
データ移動中の選択 209
タイプ表間のデータ移動の例 210
レプリケーションを使ったデータの移動 . . 212
IBM レプリケーション・ツールの構成要素 214
データウェアハウスセンターによるデータの移動 214

付録A. 構文図の読み方. 217

付録B. インポート・ユーティリティーとロード・ユーティリティーの相違点 221

付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式 . . . 225
区切り付き ASCII (DEL) ファイル形式 . . 226
DEL ファイルの例 227
DEL のデータ・タイプの説明. 228
区切りなし ASCII (ASC) ファイル形式 . . 231
ASC ファイルの例 232
ASC のデータ・タイプの説明. 233
PC バージョンの IXF ファイル形式 . . . 236
PC/IXF のレコード・タイプ 238
PC/IXF データ・タイプ. 259
PC/IXF データ・タイプの説明 266
PC/IXF ファイルのデータベースへのインポートを制御する一般規則. 273
PC/IXF ファイルのデータベースへのインポートを制御するデータ・タイプ固有の規則 275
FORCEIN オプション 279
PC/IXF およびバージョン 0 の System/370 IXF の相違 288
ワークシート・ファイル形式 (WSF) . . . 289

付録D. 警告、エラー、および完了メッセージ. 291

付録E. DB2 ライブラリーの使用法 . . . 293
DB2 PDF ファイルおよびハードコピー版資料 293
DB2 情報 293
PDF 資料の印刷 305
印刷資料の注文方法 305
DB2 オンライン文書. 305
オンライン・ヘルプへのアクセス . . . 305
オンライン情報の表示 308
DB2 ウィザードの使用 310
文書サーバーのセットアップ 312
オンライン情報の検索 313

付録F. 特記事項 315
商標 318

索引 321

IBM と連絡をとる 327
製品情報. 327

本書について

本書では、以下に示す IBM DB2 ユニバーサル・データベース (UDB) データ移動ユーティリティーに関する情報と、それぞれの使用方法について説明します。

- インポートおよびエクスポート・ユーティリティーは、表または視点と別のデータベースまたはスプレッドシート・プログラムとの間で、また DB2 データベースどうしの間で、データを移動します。さらに、DB2 コネクトを使用することにより、DB2 データベースとホスト・データベースとの間でデータを移動します。エクスポート・ユーティリティーは、データベースからオペレーティング・システム・ファイルへデータを移動します。その後、それらのファイルを使用して、別のデータベースへそのデータをインポートあるいはロードすることができます。
- ロード・ユーティリティーはデータを表に移動したり、既存の索引を拡張したり、統計情報を生成したりします。データが大量にある場合、ロードを使用したほうがインポート・ユーティリティーを使用する場合よりはるかに速くデータを移動できます。ロード・ユーティリティーでは、エクスポート・ユーティリティーを使用してアンロードされたデータをロードすることができます。
- オートローダー・ユーティリティーは大量のデータを分割し、区分データベースの別々の区画に分割データをロードします。
- DataPropagator (DPROP) は DB2 ユニバーサル・データベースの 1 つの構成要素であり、表の更新情報を他の DB2 リレーショナル・データベース内の他の表に自動コピーするためのものです。
- データウェアハウスセンター (DWC) は、操作可能データベースからウェアハウス・データベースにデータを移動するためのものです。

データベースにデータを出し入れするために他社の製品も利用できますが、本書では扱いません。

本書の対象読者

このマニュアルは、データベース管理者、アプリケーション・プログラマー、その他以下の作業を実行する DB2 UDB ユーザーを対象としています。

- オペレーティング・システム・ファイルから DB2 表へデータをロードする。

- DB2 データベースどうしの間で、また DB2 と他のアプリケーション (たとえばスプレッドシート) との間でデータを移動する。
- データをアーカイブする。

読者は DB2 ユニバーサル・データベース、構造化照会言語 (SQL)、および DB2 UDB が稼働するオペレーティング・システム環境に精通しているものと想定されています。 DB2 UDB についての一般情報については、*管理の手引き* を参照してください。 SQL については、*SQL 解説書* を参照してください。 DB2 UDB コマンド行プロセッサーの構成、起動、および使用については、 *コマンド解説書* を参照してください。 DB2 UDB アプリケーション・プログラミング・インターフェース (API) については、 *管理 API 解説書* を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、*アプリケーション構築の手引き* を参照してください。本書では、DB2 のインストール方法については説明しません。それはご使用のオペレーティング・システムによって異なります。インストール情報は、 *概説およびインストール* マニュアル (各オペレーティング・システムに対応するもの) に記載されています。

本書の編成

以下のトピックが含まれています。

『第1章 エクスポート』
DB2 エクスポート・ユーティリティーについて説明します。このユーティリティーは DB2 表からファイルへデータを移動します。

『第2章 インポート』
DB2 インポート・ユーティリティーについて説明します。このユーティリティーは、ファイルから DB2 表または視点へデータを移動します。

『第3章 ロード』
DB2 ロード・ユーティリティーについて説明します。このユーティリティーは大量のデータをファイルから DB2 表へ移動します。

『第4章 オートローダー』
オートローダー・ユーティリティーについて説明します。このユーティリティーは大量のデータを分割し、分割データを区分データベースの別々の区画にロードします。

『第5章 DB2 データ・リンク・マネージャーのデータの移動』
DB2 のエクスポート、インポート、およびロード・ユーティリティーを使用することにより、 DB2 ファイル・マネージャーのデータを移動する方法について説明します。

『第6章 システム間のデータの移動』
DB2 のエクスポート、インポート、およびロード・ユーティリティーを使用することにより、異なるプラットフォーム間で、また DRDA ホスト・データベースとの間でデータをやりとりする方法について説明します。社内の複数のデータベース間でのデータの移動のもう 1 つの方法である DataPropagator (DPROP) についても説明されています。操作可能データベースからウェアハウス・データベースにデータを移動するために使用できるデータウェアハウスセンター (DWC) についても説明されています。

『付録A. 構文図の読み方』
構文図で使用されている表記規則について説明します。

『付録B. インポート・ユーティリティーとロード・ユーティリティーの相違点』
DB2 ロード・ユーティリティーとインポート・ユーティリティーの主要な相違点を要約しています。

『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』
データベース・マネージャーのエクスポート、インポート、およびロードの各ユーティリティーでサポートされる外部ファイル形式について説明します。

『付録D. 警告、エラー、および完了メッセージ』
警告またはエラー状態が検出された場合にデータベース・マネージャーが生成するメッセージの解釈に関する情報を提供します。

第1章 エクスポート

この章では、DB2 UDB エクスポート・ユーティリティーについて説明します。このユーティリティーは、DB2 データベースからデータベースの外部に保管されている 1 つまたは複数のファイルへデータを書き出します。エクスポートされたデータは、別の DB2 データベースにインポートまたはロードすることができます。それにはそれぞれ、DB2 インポート・ユーティリティーまたは DB2 ロード・ユーティリティーを使用します。あるいは、エクスポートされたデータを別のアプリケーション (たとえばスプレッドシート) にインポートすることもできます。

以下のトピックが含まれています。

- 2ページの『エクスポートの概要』
- 3ページの『エクスポートの使用に必要な特権、権限、および許可』
- 3ページの『エクスポートの使用』
- 4ページの『エクスポートでの識別列の使用』
- 4ページの『エクスポートされた表の再作成』
- 5ページの『ラージ・オブジェクト (LOB) のエクスポート』
- 5ページの『データの並列エクスポート』
- 8ページの『EXPORT コマンド』
- 11ページの『エクスポート API』
- 17ページの『SQLUEXPT-OUT データ構造体』
- 18ページの『ファイル・タイプ修飾子 (エクスポート)』
- 21ページの『エクスポート・セッションの例』
- 25ページの『制約事項』
- 26ページの『トラブルシューティング』

DB2 データ・リンク・マネージャーのデータのエクスポートについては、190ページの『DB2 データ・リンク・マネージャーのデータの移動 - エクスポートの使用』を参照してください。タイプ表からのデータのエクスポートについては、207ページの『タイプ表間のデータ移動』を参照してください。 DRDA サーバー・データベースから DB2 コネクト・ワークステーション上のファイルへのデータのエクスポート (およびその逆) については、205ページの『DB2 コネクトによるデータの移動』を参照してください。

エクスポートの概要

エクスポート・ユーティリティーは、データベースからオペレーティング・システム・ファイル (いくつかの外部ファイル形式のいずれか) にデータをエクスポートします。

データをエクスポートするには、以下の情報が必要になります。

- エクスポートするデータを指定する SQL SELECT ステートメント。
- エクスポートするデータを入れるオペレーティング・システム・ファイルのパスおよび名前。
- 入力ファイル内のデータの形式。形式は IXF、WSF、または DEL です。225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。
- メッセージ・ファイル名。
- タイプ表をエクスポートする場合は、階層内の副表の走査順序を指定する必要があります。 IXF 形式を使用する場合は、可能な限りデフォルトの順序にしてください。順序を指定する場合は、副表を PRE-ORDER 方式で走査する必要があることに注意してください。タイプ表をエクスポートする場合、SELECT ステートメントを直接指定することはできません。その場合にはターゲットとなる副表名を必ず指定し、オプションとして WHERE 文節を指定します。エクスポート・ユーティリティーはこの情報と操作順序とを使用して、必要な SELECT ステートメントを生成および実行します。詳細は、207ページの『タイプ表間のデータ移動』を参照してください。

さらに、以下の情報を指定することもできます。

- IXF または WSF ファイルへエクスポートする場合は、新しい列名。新しい列名を指定しない場合は、既存の表または視点の中での列名が、エクスポート後のファイルの中でも使用されます。
- エクスポート操作をカスタマイズするための追加オプション (18ページの『ファイル・タイプ修飾子 (エクスポート)』を参照)。

複数データベース区画環境でエクスポート・ユーティリティーを使用する場合は、 **db2batch** を使用することによって、各データベース区画ごとにすべての作業を完了できます。 SELECT ステートメントは、ローカルに検出されるデータだけを戻すようになっている必要があります。選択条件は次のように記述します。

```
SELECT * FROM tablename WHERE NODENUMBER(column-name) = CURRENT NODE
```

db2batch については、*コマンド解説書* または*管理の手引き* を参照してください。

エクスポートの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

SYSADM または DBADM 権限か、またはエクスポート操作に関係する各表に対する CONTROL または SELECT 特権が必要です。

エクスポートの使用

エクスポートの使用の前に

エクスポート・ユーティリティーを起動するには、その前にデータのエクスポート元となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。このユーティリティーは COMMIT ステートメントを発行するため、エクスポートの起動前に COMMIT または ROLLBACK を実行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておいてください。別個の接続を使って表にアクセスしているその他のユーザー・アプリケーションを切断する必要はありません。

エクスポートの起動

エクスポート・ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  CLP によって発行する EXPORT コマンドの例を以下に示します。

  ```
  db2 export to staff.ixf of ixf select * from userid.staff
  ```

- コントロール・センターの「エクスポート (Export)」ノートブック。「エクスポート (Export)」ノートブックをオープンするには、以下のようにします。

  1. コントロール・センターから、「表 (Tables)」フォルダーまたは「視点 (Views)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。

2. 対象となるフォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存の表または視点がすべて表示されます。
3. 目次ペイン内で対象となる表または視点をマウスの右ボタンでクリックし、ポップアップ・メニューから「エクスポート (Export)」を選択します。「エクスポート (Export)」ノートブックがオープンします。

コントロール・センターについての一般情報については、*管理の手引き* を参照してください。詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

- アプリケーション・プログラミング・インターフェース (API) としての **sqluexpr**。この API については、11ページの『エクスポート API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、*アプリケーション構築の手引き* を参照してください。

エクスポートでの識別列の使用

エクスポート・ユーティリティーを使って、識別列の入った表からデータをエクスポートすることができます。エクスポート操作で指定した SELECT ステートメントが「select * from tablename」の形式であって、 METHOD オプションを使用しない場合に、識別列プロパティーを IXF ファイルにエクスポートすることができます。次に、IMPORT コマンドの REPLACE_CREATE および CREATE オプションを使って、識別列プロパティーを含む表を再作成することができます。タイプ GENERATED ALWAYS の識別列の入った表からこのような IXF ファイルが作成されている場合、 `identityignore` 修飾子を指定するのが、データ・ファイルのインポートを正常に完了する唯一の方法です。このようにしないと、すべての行が拒否されます (SQL3550W)。

エクスポートされた表の再作成

エクスポート・ユーティリティーを使用し、IXF ファイル形式を指定することによって、表を保管することができます。保管された表 (その索引を含む) は、後でインポート・ユーティリティーを使って再作成することができます (詳細は、 35ページの『エクスポートされた表の再作成』を参照してください。表に関して IXF ファイル形式が保有する属性のリストが記載されています)。

エクスポートするデータが、エクスポート・ファイルを作成する基盤となるファイル・システムの使用可能なスペースを超えると、エクスポート操作は失敗します。その場合は、WHERE 文節で条件を指定することにより、選択されるデータの量を制限して、エクスポート・ファイルがターゲット・ファイル・シ

ステムにうまく収まるようにしてください。すべてのデータをエクスポートするには、エクスポート・ユーティリティーを複数回起動することができます。

DEL および ASC ファイル形式には、ターゲット表の記述は入っていませんが、レコード・データは入っています。このようなファイル形式のデータを持つ表を再作成するには、ターゲット表を作成してから、ロード、オートローダー、またはインポート・ユーティリティーを使って、これらのファイルから表にデータを入れます。 **db2look** (DB2 統計抽出ツール、 *コマンド解説書* を参照) を使えば、元の表定義を取り込んで、対応するデータ定義言語 (DDL) を生成することができます。

ラージ・オブジェクト (LOB) のエクスポート

ラージ・オブジェクト (LOB) の列からデータをエクスポートする場合のデフォルト・アクションは、データのうち最初の 32KB を選択し、列データの残りの部分と同じファイル内にそのデータを入れるというものです。 `lobsinfile` 修飾子 (18ページの『ファイル・タイプ修飾子 (エクスポート)』を参照) を指定した場合、エクスポート・ユーティリティーは LOB 全体 (最大 2GB) を選択してそれを別個のファイルに入れます。

注: IXF ファイル形式には、LOB 列がログ記録されるかどうかなどの、列の LOB オプションが保管されません。つまり、インポート・ユーティリティーでは、 1GB 以上の大きさに定義された LOB 列の入っている表を再作成できません。

データの並列エクスポート

データを並列にエクスポートすると、データ転送量が減り、結果セットの書き込みおよび定様式出力の生成が、他の方法の場合より効果的に複数のノードに分散されます。データを並列にエクスポートする (表の区分ごとに 1 つずつ、複数のエクスポート操作を呼び出す) と、そのデータは抽出され、ローカル・ノードで変換されてから、ローカル・ファイル・システムに書き込まれます。これに対して、データを逐次的にエクスポートする (単一エクスポート操作でエクスポートする) と、データは並列抽出されてからクライアントに送られ、1 つのプロセスによって、変換が行われて結果セットがローカル・ファイル・システムに書き込まれます。

SQL ステートメントのパフォーマンス特性と実行所要時間をモニターするには、 **db2batch** コマンドを使います。このユーティリティーは、区分データベース環境内で以下を行うための並列エクスポート機能も備えています。

- エクスポートするデータを定義するための照会を実行する

- それぞれの区画に置かれているエクスポート・データの入ったファイルを各区画ごとに作成する

照会する表や、その表が置かれている区分データベース環境内の位置に応じて、並列エクスポートは次のように 2 種類あります。

- 1 つの区分表からの並列エクスポート。または、連結された複数の表に対する結合または副照会 (**db2batch** コマンドで `-p s` を指定します)。

  次の 2 通りの場合に、表は連結されているとみなすことができます。

  – それぞれの表が、1 つの区画で定義されている 1 つの区画ノードグループ内にある。

  – それぞれの表が同じノードグループ内にあって、同一数および同一タイプの列を持つ区分化キーを持っている。区分化キーの対応列は、区画に対応し、表は、区分化キー全体または区分化キーのスーパーセットで等価結合されます。

  いずれの場合も、この後で説明するとおりに NODENUMBER 機能を使って、各区画で照会を実行して、その区画のエクスポート・データ・ファイルを生成することができます。 (表が 1 つの区画内にだけ存在する場合、データは 1 つの区画でしか検索されないので、並列エクスポートは無効になることに注意してください。このような場合に並列エクスポートを有効化する方法の詳細は、次の中黒の項を参照してください。)

- 複数の非連結表からの選択 (**db2batch** コマンドで `-p t` *tablename* または `-p d` を指定します。前者では、ステージング・テーブルとして使う既存の表を指定でき、後者では、エクスポート・ユーティリティーがステージング・テーブルを作成します。)

  エクスポート・ユーティリティーは、エクスポート照会によってデータを入れられるステージング・テーブルを使います。このステージング・テーブルは、全選択照会を挿入することによって「エクスポート」結果セットの行を見つけ出すために使用します。ステージング・テーブルの作成の完了後、エクスポート・ユーティリティーは、ステージング・テーブルに対して次のコマンドを実行して、各区画でエクスポート・データ・ファイルを生成します。

  ```
  "select * WHERE NODENUMBER(colname) = CURRENT NODE"
  ```

  ステージング・テーブルを使って、 1 つの区分表を並列にエクスポートすることもできます。たいていの場合、単一区画から複数区画のステージング・テーブルにデータを転送してから、すべての区画でステージング・テーブルを並列にエクスポートするほうが、 1 つの区分表を逐次的にエクスポートするよりも高速です。

エクスポート・ユーティリティーは、各区画で並列に照会を実行して、それぞれの区画でデータを検索します。 db2batch -p s の場合、元の選択照会は並列に実行されます。 db2batch -p t と db2batch -p d の場合、指定された照会を使ってエクスポート・データと一緒にステージング・テーブルがロードされます。また SELECT * 照会は、各区画のステージング・テーブルに対して並列に実行されて、データをエクスポートします。特定の区画に置かれているデータだけをエクスポートするには、 **db2batch** で、その区画で実行される照会の WHERE 文節に対して、述部 NODENUMBER(*colname*) = CURRENT NODE を追加します。 *colname* パラメーターは、表列の修飾名または非修飾名に設定しなければなりません。エクスポート・ユーティリティーは、元の照会内の最初の列名を使ってこのパラメーターを設定します。

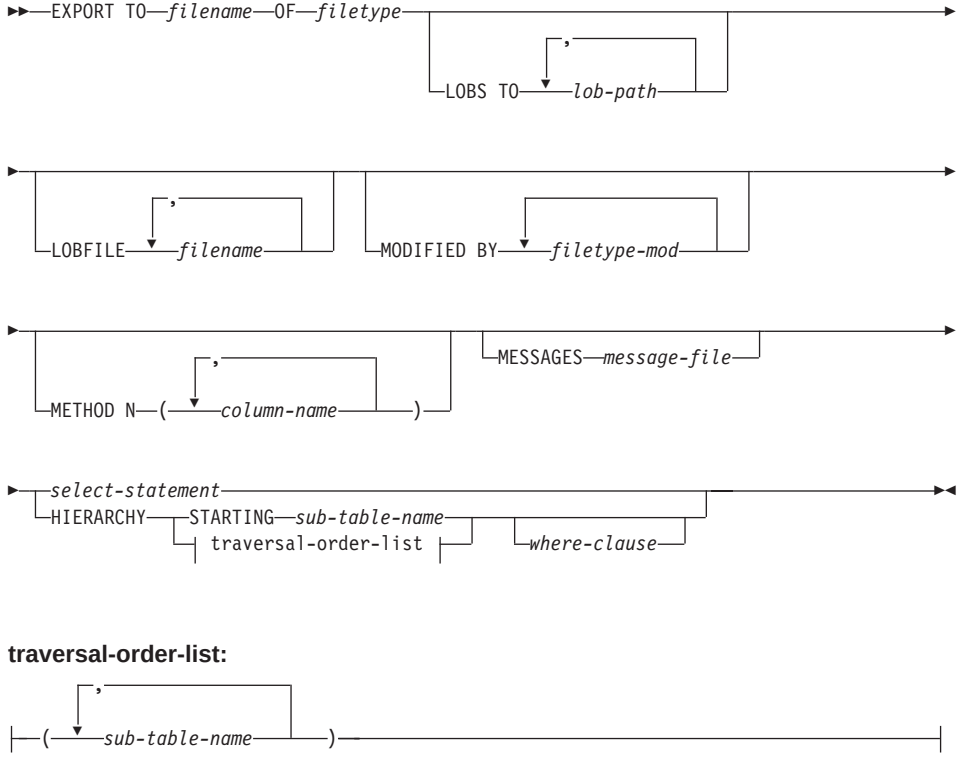
以下に示すのは、エクスポート・ユーティリティーが使用する照会に対する制限事項です。

- db2batch -p s の指定時には、照会に列関数を指定するだけでは不十分です。これは、NODENUMBER *colname* 述部には列名が必要であるためです。
- db2batch -p s を指定する場合、集合体 (min、max、avg など) は、区分化キーを含めたグループ化に基づいていなければなりません。
- db2batch -p t または db2batch -p d を指定する場合、照会に ORDER BY を入れることはできません。これは、INSERT ステートメント内の全選択での ORDER BY が DB2 UDB でサポートされていないためです。

db2batch コマンドに -p s を指定したときに、 -r オプションを使って結果出力ファイルを作成すると、 ORDER BY 文節があれば各区画上のファイルは分類順になります。 1 つの分類ファイルを得たい場合は、各区画上の分類ファイルを組み合わせて 1 つの分類ファイルにします。たとえば、UNIX ベースのシステムでは、コマンド sort -m を使ってファイルを組み合わせて 1 つの分類ファイルにします。 NFS マウント・ファイル・システムに出力を送信する場合、 ORDER BY 文節を指定しても出力ファイルは分類されません。

EXPORT コマンド

コマンド構文



コマンド・パラメーター

HIERARCHY traversal-order-list
指定した走査順序を使用して、副階層をエクスポートします。副表はすべて PRE-ORDER 方式でリストしなければなりません。最初の副表名が、SELECT ステートメントのターゲット表名として使用されます。

HIERARCHY STARTING sub-table-name
デフォルトの走査順序 (ASC、DEL、WSF ファイルの場合は OUTER、それ以外は PC/IXF データ・ファイルに保管されている順序) を使用して、 *sub-table-name* から始めて副階層をエクスポートします。

LOBFILE filename
LOB ファイルの基本ファイル名を 1 つまたは複数個指定します。最初

の名前のためのネーム・スペースが使用済みの場合は 2 番目の名前が使用される、というようになります。

エクスポート操作中に LOB ファイルを作成する場合は、このリストによる現在の基本名を現行パス (*lob-path* による) に付加し、さらに 3 桁の順序番号を追加することによってファイル名が構成されます。たとえば、現行の LOB パスがディレクトリー `/u/foo/lob/path` で、現行の LOB ファイル名が `bar` の場合に作成される LOB ファイル名は、`/u/foo/lob/path/bar.001`、`/u/foo/lob/path/bar.002`、などになります。

LOBS TO lob-path
LOB ファイルを保管するディレクトリーのパス名を 1 つまたは複数個指定します。最初のパスでファイル・スペースが使用済みの場合は 2 番目のパスが使用される、というようになります。

MESSAGES message-file
エクスポート操作中に発生する警告およびエラー・メッセージの出力先を指定します。ファイルがすでに存在する場合、エクスポート・ユーティリティーは情報をそこに追加します。 *message-file* を省略した場合、メッセージは標準出力に出力されます。

METHOD N column-name
出力ファイル内で使用する 1 つ以上の列名を指定します。このパラメーターを指定しない場合は、表の中の列名が使用されます。このパラメーターは WSF および IXF ファイルでのみ有効であり、階層データのエクスポート時には無効です。

MODIFIED BY filetype-mod
追加オプションを指定します (18ページの表2 を参照)。

OF filetype
出力ファイル内のデータの形式を指定します。

- DEL (区切り付き ASCII 形式)。これはさまざまなデータベース・マネージャーおよびファイル・マネージャー・プログラムで使用されます。
- WSF (ワークシート形式)。以下のようなプログラムで使用されます。
  - ロータス 1-2-3
  - Lotus Symphony

  注: BIGINT または DECIMAL のデータをエクスポートする場合に正確にエクスポートできるのは、DOUBLE 型の範囲内に収まる

値だけです。この範囲内に収まらない値もエクスポートできますが、オペレーティング・システムによっては、そのような値を再びインポートまたはロードすると不正確なデータになる場合があります。

- IXF (統合交換フォーマット、PC バージョン)。IXF ファイルには、表属性のほとんどと既存の索引が保管されます。ただし、SELECT ステートメントで列を指定した場合は除きます。他のファイル形式ではデータを表にインポートする前にあらかじめ表が存在している必要がありますが、この形式の場合は表を再作成できます。表に関して IXF ファイル形式が保有する属性のリストは、 35ページの『エクスポートされた表の再作成』を参照してください。

ファイル形式については、 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

select-statement

エクスポートするデータを戻す SELECT ステートメントを指定します。 SELECT ステートメントでエラーが発生した場合は、メッセージ・ファイル (または標準出力) にメッセージが書き込まれます。エラー・コードが SQL0012W、SQL0347W、SQL0360W、SQL0437W、または SQL1824W のいずれかであればエクスポート操作は続行され、それ以外の場合は停止します。

TO filename

データのエクスポート先となるファイルの名前を指定します。ファイルの完全パスを指定しなければ、エクスポート・ユーティリティーはデフォルト・ドライブと現行ディレクトリーを宛先として使用します。

すでに存在するファイルの名前を指定した場合、エクスポート・ユーティリティーはそのファイルの内容を上書きします。情報を追加することはありません。

エクスポート API

C API 構文

```
/* File: sqlutil.h */
/* API: Export */
/* ... */
SQL_API_RC SQL_API_FN
  sqluexpr (
    char * pDataFileName,
    sqlu_media_list * pLobPathList,
    sqlu_media_list * pLobFileList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pMsgFileName,
    short CallerAction,
    struct sqluexpt_out*  pOutputInfo,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

汎用 API 構文

```
/* File: sqlutil.h */
/* API: Export */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgexpr (
    unsigned short DataFileNameLen,
    unsigned short FileTypeLen,
    unsigned short MsgFileNameLen,
    char * pDataFileName,
    sqlu_media_list * pLobPathList,
    sqlu_media_list * pLobFileList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pMsgFileName,
    short CallerAction,
    struct sqluexpt_out*  pOutputInfo,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

API パラメーター

DataFileNameLen
入力。データ・ファイル名の長さをバイト数で表す 2 バイトの符号なし整数。

FileTypeLen
入力。ファイル・タイプの長さをバイト数で表す 2 バイトの符号なし整数。

MsgFileNameLen
入力。メッセージ・ファイル名の長さをバイト数で表す 2 バイトの符号なし整数。

pDataFileName
入力。データのエクスポート先となる外部ファイルのパスおよび名前を入れるストリング。

pLobPathList
入力。 *media_type* SQLU_LOCAL_MEDIA、および *sqlu_media_entry* 構造体を使用する *sqlu_media_list* で、クライアント上で LOB ファイルを格納する場所を示すパスのリストを入れます。

このリスト内の最初のパスでファイル・スペースが使用済みの場合、API は 2 番目のパスを使用する、という具合になります。

詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

pLobFileList
入力。 *media_type* SQLU_CLIENT_LOCATION、および *sqlu_location_entry* 構造体を使用する *sqlu_media_list* で、基本ファイル名を入れます。

このリスト内の最初の名前でファイル・スペースが使用済みの場合、API は 2 番目の名前を使用する、という具合になります。

詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

エクスポート操作中に LOB ファイルを作成する場合は、このリストによる現在の基本名を現行パス (*pLobFilePath* による) に付加し、さらに 3 桁の順序番号を追加することによってファイル名が構成されます。たとえば、現行の LOB パスがディレクトリー /u/foo/lob/path で、現行の LOB ファイル名が bar の場合に作成される LOB ファイル名は、 /u/foo/lob/path/bar.001、/u/foo/lob/path/bar.002、などになります。

pDataDescriptor

入力。出力ファイルの列名を指定する *sqldcol* 構造体へのポインター。このパラメーターで与えられる情報の残りの部分をエクスポート・ユーティリティーがどう解釈するかは、 *dcolmeth* フィールドの値によって決まります。このパラメーターの有効な値 (`sqlutil` で定義される) は、以下のとおりです。

SQL_METH_N
名前。出力ファイルで使用する列名を指定します。

SQL_METH_D
デフォルト。表に既存の列名を出力ファイルで使用します。この場合、列の数と列指定配列はどちらも無視されます。列名は、*pActionString* で指定される SELECT ステートメントの出力から導出されます。

詳しくは、 *管理 API 解説書* の『SQLDCOL』を参照してください。

pActionString

入力。有効な動的 SQL SELECT ステートメントを入れる *sqlchar* 構造体へのポインター。その構造体には、2 バイト長フィールドに続いて、SELECT ステートメントを構成する文字列が含まれます。SELECT ステートメントは、データベースから抽出されて外部ファイルに書き込まれるデータを指定します。

外部ファイルの列 (*pDataDescriptor* より) と、SELECT ステートメントのデータベース列とは、それぞれのリスト / 構造体の中での位置に従って突き合わせられます。データベースから選択されるデータの最初の列は外部ファイルの最初の列になり、その列名は外部列配列の最初の要素から取られます。

詳しくは、*管理 API 解説書* の『SQLCHAR』を参照してください。

注: タイプ表で使用される構文については、 8ページの『EXPORT コマンド』を参照してください。

pFileType

入力。外部ファイル内のデータの形式を示すストリング。サポートされる外部ファイル形式 (`sqlutil` で定義される) は、以下のとおりです。

SQL_DEL
区切り付き ASCII。dBase、BASIC、IBM パーソナル・デシジョン・シリーズのプログラム、その他多くのデータベース・マネージャーおよびファイル・マネージャーとのデータ交換に使用できます。

SQL_WSF
ワークシート形式。Lotus Symphony およびロータス 1-2-3 のプログラムとのデータ交換に使用できます。

SQL_IXF
PC バージョンの IXF (統合交換フォーマット)。表からのデータ・エクスポートとして望ましい方式です。このファイル形式にエクスポートされたデータは、後で同じ表にインポートまたはロードしたり、別のデータベース・マネージャー表にインポートまたはロードしたりすることができます。

pFileTypeMod
入力。 *sqldcol* 構造体へのポインター。この構造体には、2 バイト長フィールドに続いて、 1 つ以上の処理オプションを指定する文字の配列が含まれます。このポインターがヌルか、またはポイント先の構造体の文字数が 0 の場合、このアクションはデフォルト指定を選択したものと解釈されます。

サポートされているすべてのファイル・タイプですべてのオプションが使えるわけではありません。

詳しくは、*管理 API 解説書* の『SQLCHAR』、および 18ページの『ファイル・タイプ修飾子 (エクスポート)』を参照してください。

pMsgFileName
入力。ユーティリティーが戻すエラー、警告、通知メッセージの宛先を入れるストリング。オペレーティング・システム・ファイルのパスと名前、または標準装置を指定できます。ファイルがすでに存在する場合は上書きされます。ファイルが存在しなければ作成されます。

CallerAction
入力。呼び出し元が要求するアクション。有効な値 (`sqlutil` で定義される) は、以下のとおりです。

SQLU_INITIAL
初期呼び出し。 API の最初の呼び出しでは、この値を使用しなければなりません。

初期呼び出しまたはそれ以降の呼び出しから戻る際、要求されたエクスポート操作を完了する前に呼び出し元アプリケーションでなんらかのアクションを実行することが必要な場合は、この呼び出し元のアクションを以下のいずれかに設定する必要があります。

SQLU_CONTINUE
処理を続行します。この値は API の後続呼び出しでのみ使用

できます。つまり、初期呼び出しから戻った時点でユーティリティーがユーザー入力 (「テープの終わり」状態への応答など) を要求した場合に、その後で使用します。これは、ユーティリティーの要求したユーザー処置が完了し、ユーティリティーが初期要求の処理を続行できるということを指定します。

SQLU_TERMINATE
処理を終了します。この値は API の後続呼び出しでのみ使用できます。つまり、初期呼び出しから戻った時点でユーティリティーがユーザー入力 (「テープの終わり」状態への応答など) を要求した場合に、その後で使用します。これは、ユーティリティーの要求したユーザー処置が実行されず、ユーティリティーが初期要求の処理を終了することを指定します。

pOutputInfo
出力。ターゲット・ファイルにエクスポートされたレコードの数を戻します。この構造体については、17ページの『SQLUEXPT-OUT データ構造体』を参照してください。

pReserved
将来の利用のために予約済み。

pSqlca
出力。 *sqlca* 構造体へのポインター。この構造体については、 *管理 API 解説書* の『SQLCA』を参照してください。

REXX API 構文

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]
CONTINUE EXPORT
STOP EXPORT
```

REXX API パラメーター

stmt 有効な動的 SQL SELECT ステートメントを入れる REXX ホスト変数。このステートメントは、データベースから抽出されるデータを指定します。

datafile
データのエクスポート先となるファイルの名前。

filetype

エクスポート・ファイル内のデータの形式。サポートされるファイル形式は以下のとおりです。

DEL 区切り付き ASCII

WSF ワークシート形式

IXF PC バージョンの IXF (統合交換フォーマット)

filetmod

追加の処理オプションを入れるホスト変数 (18ページの『ファイル・タイプ修飾子 (エクスポート)』を参照)。

dcoldata

エクスポート・ファイルで使用する列名を入れる REXX 複合ホスト変数。以下の記述の中の XXX は、ホスト変数の名前を表します。

XXX.0 列の数 (変数の残りの部分の要素数)。

XXX.1 最初の列名。

XXX.2 第 2 の列名。

XXX.3 以下同様。

このパラメーターがヌルであるか、または *dcoldata* の値が指定されていない場合、ユーティリティーはデータベース表の列名を使用します。

msgfile

エラーおよび警告メッセージの送信先のファイル、パス、または装置名。

number

エクスポートされた行の数が入れられるホスト変数。

SQLUEXPT-OUT データ構造体

この構造体は、 11ページの『エクスポート API』からの情報を渡すのに使用されます。

表 1. SQLUEXPT-OUT 構造体のフィールド

フィールド名	データ・タイプ	説明
SIZEOFSTRUCT	INTEGER	構造体のサイズ。
ROWSEXPORTED	INTEGER	データベースからターゲット・ファイルへエクスポートされたレコードの数。

言語構文

C 構造体

```
/* File: sqlutil.h */
/* Structure: SQL-UEXPT-OUT */
/* ... */
SQL_STRUCTURE sqluexpt_out
{
  sqluint32       sizeOfStruct;
  sqluint32       rowsExported;
};
/* ... */
```

COBOL 構造体

```
* File: sqlutil.cbl
01 SQL-UEXPT-OUT.
    05 SQL-SIZE-OF-UEXPT-OUT  PIC 9(9) COMP-5 VALUE 8.
    05 SQL-ROWSEXPORTED       PIC 9(9) COMP-5 VALUE 0.
*
```

ファイル・タイプ修飾子 (エクスポート)

表 2. 有効なファイル・タイプ修飾子 (エクスポート)

修飾子	説明
すべてのファイル形式	
lobsinfile	*lob-path* で、LOB 値を含むファイルへのパスを指定します。
DEL (区切り付き ASCII) ファイル形式	
chardel*x*	*x* は単一文字のストリング区切り文字です。デフォルト値は二重引用符 (") です。ここに指定する文字は、文字ストリングを囲むために二重引用符の代わりに使用されます。[a] 文字ストリング区切り文字として、以下のように単一引用符 (') を指定することもできます。 `modified by chardel''`
coldel*x*	*x* は単一文字の列区切り文字です。デフォルト値はコンマ (,) です。ここに指定する文字は、列の終わりを示す記号としてコンマの代わりに使用されます。[a] 次の例の場合、`coldel;` の指定によりエクスポート・ユーティリティーは、検出するセミコロン (;) すべてを列の区切り文字として解釈します。 `db2 "export to temp of del modified by coldel; select * from staff where dept = 20"`
datesiso	日付形式。すべての日付データ値を ISO 形式 (*"YYYY-MM-DD"*) でエクスポートします。[b]
decplusblank	正符号文字。正の 10 進数値の前に、正符号 (+) の代わりにブランク・スペースを付けます。デフォルト・アクションでは、正の 10 進数値の前に正符号が付けられます。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使う単一文字です。デフォルト値はピリオド (.) です。ここに指定する文字は、ピリオドの代わりに小数点文字として使われます。[a]
dldel*x*	*x* は単一文字の DATALINK 区切り文字です。デフォルト値はセミコロン (;) です。ここに指定する文字は、DATALINK 値のフィールド間区切り文字としてセミコロンの代わりに使用されます。 DATALINK 値の副値が複数になることがあるので、これが必要になります。[a] **注:** *x* を、行、列、または文字ストリングの区切り文字として指定された文字と同じにすることはできません。
nodoubledel	二重文字区切り文字の認識を抑止します。詳しくは、19ページの『区切り文字に関する制限』を参照してください。

表 2. 有効なファイル・タイプ修飾子 (エクスポート) (続き)

修飾子	説明
WSF ファイル形式	
1	ロータス 1-2-3 リリース 1、またはロータス 1-2-3 リリース 1a と互換性のある WSF ファイルを作成します。[c] これはデフォルトです。
2	Lotus Symphony リリース 1.0 と互換性のある WSF ファイルを作成します。[c]
3	ロータス 1-2-3 バージョン 2、または Lotus Symphony リリース 1.1 と互換性のある WSF ファイルを作成します。[c]
4	DBCS 文字を含む WSF ファイルを作成します。

注:

1. サポートされないファイル・タイプを MODIFIED BY オプションで使用しようとした場合、エクスポート・ユーティリティーは警告を出しません。それを試みた場合、エクスポート操作は失敗してエラー・コードが戻されます。
2. [a] 『区切り文字に関する制限』に、区切り文字の指定変更として使用できる文字に適用される制限のリストが示されています。
3. [b] 通常、エクスポート・ユーティリティーでの記述形式は次のとおりです。
   - 日付データ: *YYYYMMDD* の形式
   - 文字 (日付) データ: *YYYY-MM-DD* の形式
   - 時刻データ: *HH.MM.SS* の形式
   - タイム・スタンプ・データ: *YYYY-MM-DD-HH.MM.SS.uuuuuu* の形式

   エクスポート操作用の SELECT ステートメントで指定されたすべての日時列に入れられるデータも、上記の形式になります。
4. [c] これらのファイルは特定の製品に対して出力することもできます。そうするには、*filetype-mod* パラメーター・ストリングに L (ロータス 1-2-3 の場合) または S (Lotus Symphony の場合) を指定します。 1 つの値または製品指定文字だけを指定できます。

区切り文字に関する制限

選択する区切り文字が、移動するデータの一部として含まれていないことを確認するのはユーザーの責任です。含まれている場合、予期しないエラーが発生することがあります。以下の制限は、データの移動における列、ストリング、DATALINK、および小数点の各区切り文字に適用されます。

- 区切り文字はそれぞれ異なっていなければなりません。

- 区切り文字を、2 進数 0、改行 (LF)、復帰 (CR)、ブランク・スペースにすることはできません。
- デフォルトの小数点 (.) をストリング区切り文字に使うことはできません。
- 以下の文字の指定は、ASCII 系のコード・ページと EBCDIC 系のコード・ページでは異なっています。
  - EBCDIC MBCS データ・ファイルにおいて、シフトイン (0x0F) およびシフトアウト (0x0E) 文字を区切り文字として使うことはできません。
  - MBCS、EUC、または DBCS コード・ページ用の区切り文字は 0x40 以下でなければなりません。ただし、EBCDIC MBCS データ用のデフォルト小数点は例外であり、それは 0x4b です。
  - ASCII コード・ページまたは EBCDIC MBCS コード・ページでのデータ・ファイルのデフォルト区切り文字は以下のとおりです。

    ```
    " (0x22, double quotation mark; string delimiter)
    , (0x2c, comma; column delimiter)
    ```

  - EBCDIC SBCS コード・ページでのデータ・ファイルのデフォルト区切り文字は以下のとおりです。

    ```
    " (0x7F, double quotation mark; string delimiter)
    , (0x6B, comma; column delimiter)
    ```

  - ASCII データ・ファイルのデフォルト小数点は 0x2e (ピリオド) です。
  - EBCDIC データ・ファイルのデフォルト小数点は 0x4B (ピリオド) です。
  - サーバーとクライアントとでコード・ページが異なっている場合、可能な限りデフォルト以外の区切り文字を 16 進表示で指定するようにしてください。たとえば、

    ```
    db2 load from ... modified by chardel0x0C coldelX1e ...
    ```

エクスポート、インポート、およびロード・ユーティリティーには、DEL ファイル内の二重文字区切り文字のサポートに関する以下の情報が適用されます。

- DEL ファイルの文字ベースのフィールド内では、文字区切り文字を使用できます。これは、タイプが CHAR、VARCHAR、LONG VARCHAR、または CLOB のフィールドに適用されます (ただし `lobsinfile` を指定した場合を除く)。文字区切り文字で囲まれている範囲内に文字区切り文字の対が検出された場合、それはデータベース内にインポートまたはロードされます。たとえば、

  ```
  "What a ""nice"" day!"
  ```

  は、次のようにインポートされます。

  ```
  What a "nice" day!
  ```

エクスポートの場合は、この規則が逆に適用されます。たとえば、

```
I am 6" tall.
```

を DEL ファイルにエクスポートすると、次のようになります。

```
"I am 6"" tall."
```

- DBCS 環境では、パイプ (|) 文字区切り文字はサポートされません。

エクスポート・セッションの例

CLP の例

次の例は、SAMPLE データベースの中の STAFF 表から myfile.ixf へ、IXF 形式の出力で情報をエクスポートする方法を示しています。ユーザーはすでにデータベースに接続していることが必要です。データベース接続が DB2 コネクトを介していない場合、索引定義が存在するならそれは出力ファイルに格納されます。そうでなければ、データだけが格納されます。

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

次の例は、SAMPLE データベースの中の STAFF 表から awards.ixf へ、部署 (dept) 20 の従業員に関する情報を IXF 形式の出力でエクスポートする方法を示しています。ユーザーはすでにデータベースに接続していることが必要です。

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
   where dept = 20
```

次の例は LOB を DEL ファイルにエクスポートする方法を示しています。

```
db2 export to myfile.del of del lobs to mylobs
   lobfile lobs1, lobs2 modified by lobsinfile
   select * from emp_photo
```

次の例は LOB を DEL ファイルにエクスポートする方法を示しています。ここでは、最初のディレクトリーにファイルを入れることができない場合のために 2 番目のディレクトリーを指定しています。

```
db2 export to myfile.del of del
   lobs to /db2exp1, /db2exp2 modified by lobsinfile
   select * from emp_photo
```

次の例はデータを DEL ファイルにエクスポートする方法を示しています。ここでは、単一引用符をストリング区切り文字として使用し、セミコロンを列の区切り文字として使用し、コンマを小数点として使用します。データを再びデータベースにインポートする場合、これと同じ規則を使用する必要があります。

```
db2 export to myfile.del of del
   modified by chardel'' coldel; decpt,
   select * from staff
```

API の例

下記のサンプル・プログラムは、次のことを実行する方法を示しています。

- SAMPLE データベースの中の STAFF 表からファイル EXPTABLE.DEL へ情報をエクスポートします。
- その情報を、区切り付きテキスト・ファイルから新しい表 IMPTABLE へインポートします。

SAMPLE データベースについては、*管理の手引き* を参照してください。

このサンプル・プログラムのソース・ファイル (impexp.sqc) は、¥sqllib¥samples¥c ディレクトリーにあります。そこには、DB2 API と組み込み SQL 呼び出しの両方が入っています。同じディレクトリーにあるスクリプト・ファイル bldvaemb.cmd には、このサンプル・プログラムや他のサンプル・プログラムを構築するためのコマンドが含まれています。 DB2 管理用 API を含むアプリケーションの作成についての一般情報、およびコンパイルとリンクのオプションについては、*アプリケーション構築の手引き* を参照してください。 OS/2 で、ソース・ファイル impexp.sqc からサンプル・プログラム impexp を構築するには、以下のようにします。

1. ファイル impexp.sqc、bldvaemb.cmd、util.c、および util.h を作業ディレクトリーにコピーします。
2. データベース・マネージャーが実行されていないなら、コマンド db2start を発行します。
3. bldvaemb impexp sample と入力します。以下のファイルが生成されます。

   ```
   impexp.bnd
   impexp.c
   util.obj
   impexp.obj
   impexp.exe
   ```

サンプル・プログラム (実行可能ファイル) を実行するには、impexp と入力します。どんなファイルが生成されるかを調べてみてください。たとえば、メッセージ・ファイルや区切り付き ASCII データ・ファイルが生成されます。

```
/******************************************************************************
**
** Source File Name = impexp.sqc  1.4
**
**    PURPOSE :
```

```
**       This program is an example of how APIs are implemented in order to
**       export and import tables and table data.  The order of the program
**       is as follows:
**          - export a table to a comma-delimited text file
**          - import the comma-delimited text file to a DB2 table
**       This program needs the embedded SQL calls in order to connect to
**       an existing database, then to create a temporary table to work with.
**
**    STRUCTURES USED :
**       sqldcol
**       sqlchar
**       sqluexpt_out
**       sqluimp_in
**       sqluimp_out
**       sqlca
**
**    APIs USED :
**             IMPORT TO             sqluimpt_api
**             EXPORT                sqlgexpt
**
**    FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**          string.h -  fgets, strncpy
**
**       DBMS LIBRARY :
**          sqlenv.h -  see "APIs USED" above
**
**       OTHER :
**          internal :
**             list_dcs :         Displays a directory of databases
**
**          external :
**             check_error :     Checks for SQLCODE error, and prints out any
**             [in UTIL.C]       related information available.
**
**    EXTERNAL DEPENDANCIES :
**       - Ensure existence of database (SAMPLE) for precompile purposes.
**       - Precompile with the SQL precompiler (PREP in DB2)
**       - Bind to a database (BIND in DB2)
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#ifndef DB2MAC
#include <malloc.h>
#endif
#include "util.h"
#ifdef DB268K
```

エクスポート・セッションの例

```
/* Need to include ASLM for 68K applications */
#include <LibraryManager.h>
#endif
#define  CHECKERR(CE_STR)   if (check_error (CE_STR, &sqlca) != 0) return 1;
EXEC SQL INCLUDE SQLCA;
int main (int argc, char *argv[]) {
   short int            callerAction = 0;
   struct sqldcol       columnData;
   struct sqlchar       *columnStringPointer;
   struct sqluexpt_out  outputInfo;
   struct sqluimpt_in   impInput;
   struct sqluimpt_out  impOutput;
   char        datafile[] = "EXPTABLE.DEL";
   char        statement[] = "select name, id from staff";
   char        impStatement[] = "insert into imptable (name, id)";
   char        msgfile_x[] = "EXPMSG.TXT";
   char        msgfile_m[] = "IMPMSG.TXT";
   char        fileFormat[] = "DEL";
   EXEC SQL BEGIN DECLARE SECTION;
      char userid[9];
      char passwd[19];
   EXEC SQL END DECLARE SECTION;
#ifdef DB268K
   /* Before making any API calls for 68K environment,
      need to initial the Library Manager */
InitLibraryManager(0,kCurrentZone,kNormalMemory);
atexit(CleanupLibraryManager);
#endif

   /* need to preset the size of structure field and counts */
   outputInfo.sizeOfStruct = SQLUEXPT_OUT_SIZE;
   impInput.sizeOfStruct = SQLUIMPT_IN_SIZE;
   impOutput.sizeOfStruct = SQLUIMPT_OUT_SIZE;
   impInput.restartcnt = impInput.commitcnt = 0;
   /*****************************************************************¥
   * need to allocate the proper amount of space for the SQL statement *
   ¥*****************************************************************/
   columnStringPointer = (struct sqlchar *)malloc(strlen(statement)
      + sizeof (struct sqlchar));
   columnStringPointer->length = strlen(statement);
   strncpy (columnStringPointer->data, statement, strlen(statement));
   /* DELimited format can not have specified names, therefore the
      column method is 'D'efault */
   columnData.dcolmeth = 'D';
   if (argc == 1) {
      EXEC SQL CONNECT TO sample;
  CHECKERR ("CONNECT TO SAMPLE");
   }
   else if (argc == 3) {
      strcpy (userid, argv[1]);
      strcpy (passwd, argv[2]);
      EXEC SQL CONNECT TO sample USER :userid USING :passwd;
      CHECKERR ("CONNECT TO SAMPLE");
   }
   else {
```

```
      printf ("¥nUSAGE: impexp [userid passwd]¥n¥n");
      return 1;
   } /* endif */

   printf ("exporting NAME and ID from STAFF table into file '%s'¥n", datafile);
   /*******************¥
   * EXPORT API called *
   ¥*******************/
   sqluexpr (datafile, NULL, NULL, &columnData, columnStringPointer,
      fileFormat, NULL, msgfile_x, 0, &outputInfo, NULL, &sqlca);
   CHECKERR ("exporting table");
   printf ("rows exported %d¥n", outputInfo.rowsExported);
   free (columnStringPointer);
   /*********************************************************************¥
   * need to allocate the proper amount of space for the SQL statement *
   ¥*********************************************************************/
   columnStringPointer = (struct sqlchar *)malloc(strlen(impStatement)
      + sizeof (struct sqlchar));
   columnStringPointer->length = strlen(impStatement);
   strncpy (columnStringPointer->data, impStatement, strlen(impStatement));

   printf ("creating a temporary table 'imptable' to import into¥n");
   /* create a temporary table to import into */
   EXEC SQL CREATE TABLE imptable (name VARCHAR(15), id INT);
   CHECKERR ("CREATE TABLE");
   printf ("importing the file '%s' into the 'imptable'¥n", datafile);
   /*******************¥
   * IMPORT API called *
   ¥*******************/
   sqluimpr (datafile, NULL, &columnData, columnStringPointer, fileFormat,
      NULL, msgfile_m, 0, &impInput, &impOutput, NULL, NULL, &sqlca);
   CHECKERR ("importing table");
   printf ("rows imported %d¥nnumber of rows committed %d¥n",
      impOutput.rowsInserted, impOutput.rowsCommitted);
   free (columnStringPointer);
   /* drop the table  */
   EXEC SQL DROP TABLE imptable;
   CHECKERR ("DROP TABLE");

   EXEC SQL CONNECT RESET;
   CHECKERR ("CONNECT RESET");
}
/* end of program : impexp.sqc */
```

制約事項

エクスポート・ユーティリティーには、以下の制約事項が適用されます。

- このユーティリティーでは、通称の使用はサポートされません。
- このユーティリティーでは、構造タイプ列を持つ表はサポートされません。

トラブルシューティング

データのエクスポート、インポート、ロード、バインド、復元などの DB2 操作においては、メッセージ・ファイルを作成するよう指定できます。メッセージ・ファイルには、それらの操作に関連したエラー、警告、および通知の各メッセージが入れられます。 MESSAGES パラメーターでそのファイルの名前を指定します。

それらのメッセージ・ファイルは標準 ASCII テキスト・ファイルです。メッセージ・ファイル内では、各メッセージはそれぞれ新たな行で始まり、 DB2 メッセージ検索機能により提供される情報がそこに入れられます。これを印刷するには、ご使用のオペレーティング・システムの印刷手順を使用します。表示するには、任意の ASCII エディターを使用してください。

第2章 インポート

この章では、DB2 UDB インポート・ユーティリティーについて説明します。このユーティリティーは SQL INSERT ステートメントを使って入力ファイルから表または視点にデータを書き込みます。ターゲットの表または視点にすでにデータが入っている場合は、既存のデータを置き換えたりそこに追加したりすることができます。

以下のトピックが含まれています。

- 28ページの『インポートの概要』
- 29ページの『インポートの使用に必要な特権、権限、および許可』
- 29ページの『インポートの使用』
- 31ページの『バッファー挿入によるインポートの使用』
- 31ページの『インポートでの識別列の使用』
- 33ページの『インポートでの生成列の使用』
- 35ページの『エクスポートされた表の再作成』
- 36ページの『ラージ・オブジェクト (LOB) のインポート』
- 37ページの『ユーザー定義特殊タイプ (UDT) のインポート』
- 37ページの『インポート時の表のロッキング』
- 38ページの『IMPORT コマンド』
- 46ページの『インポート API』
- 60ページの『ファイル・タイプ修飾子 (インポート)』
- 70ページの『文字セットと NLS についての考慮事項』
- 70ページの『インポート・セッションの例』
- 72ページの『インポートのパフォーマンスの最適化』
- 72ページの『制約事項と制限』
- 73ページの『トラブルシューティング』

DB2 データ・リンク・マネージャーのデータのインポートについては、194ページの『DB2 データ・リンク・マネージャーのデータの移動 - インポートの使用』を参照してください。タイプ表からのデータのインポートについては、207ページの『タイプ表間のデータ移動』を参照してください。 DB2 コネクト・ワークステーション上のファイルから DRDA サーバー・データベースへ

のデータのエクスポート (およびその逆) については、205ページの『DB2 コネクトによるデータの移動』を参照してください。

インポートの概要

インポート・ユーティリティーは、入力ファイルから表または更新可能な視点へデータを挿入します。インポート・データを受け取る表または視点にすでにデータが入っている場合は、既存のデータを置き換えたりそこに追加したりすることができます。

データをインポートするには、以下の情報が必要になります。

- 入力ファイルのパスと名前。
- ターゲット表またはターゲット視点の名前または別名。
- 入力ファイル内のデータの形式。形式は IXF、WSF、DEL、または ASC です。 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。
- 入力データを表または視点に挿入するのか、それとも表または視点の既存のデータを入力データによって更新あるいは置換するのか。
- アプリケーション・プログラミング・インターフェース (API) **sqluimpr** によってユーティリティーが起動される場合、メッセージ・ファイル名。
- タイプ表を処理する場合は、構造化されたタイプ全体の進め方についての方式あるいは順序を指定する必要があります。階層内の上位表と副表の全体を上から下へ、あるいは左から右へ進む順序を*走査* 順序といいます。階層間でデータを移動する場合、この順序は他のデータとの相対関係でデータがどこに移動するかを決めるものとなるため、これは重要です。

  タイプ表を処理する場合、副表リストを用意することも必要です。このリストは、どの副表および属性にデータをインポートするかを示します。

  詳しくは、207ページの『タイプ表間のデータ移動』を参照してください。

さらに、以下の情報を指定することもできます。

- データのインポートに使用する方式: 列のロケーション、列名、または相対列位置。
- 表に対する変更をコミットする INSERT 行数。定期的に COMMIT を要求することによって、インポート操作中に障害や ROLLBACK が発生した場合に失われる行の数を少なくすることができます。また、大きな入力ファイルを処理する場合に DB2 ログがいっぱいになってしまうのを避けることもできます。

- インポート操作の開始前にスキップするファイル・レコードの数。エラーが発生した場合、正常にインポートされコミットされた最後の行の直後からインポート操作を再開することができます。
- データの挿入先となる表または視点内の列の名前。

インポートの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

インポート・ユーティリティーを使って新しい表を作成するには、SYSADM 権限、DBADM 権限、またはそのデータベースに対する CREATETAB 特権が必要です。既存の表または視点内のデータを置換するには、SYSADM 権限、DBADM 権限、あるいはその表または視点に対する CONTROL 特権が必要です。既存の表または視点にデータを追加するには、その表または視点に対する SELECT 特権と INSERT 特権が必要です。

インポートの使用

インポートを使用する前に

インポート・ユーティリティーを起動するには、その前にデータのインポート先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。このユーティリティーは COMMIT または ROLLBACK ステートメントを発行するため、インポートの起動前に COMMIT または ROLLBACK を実行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておいてください。

インポートの起動

インポート・ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  CLP によって発行する IMPORT コマンドの例を以下に示します。

```
db2 import from stafftab.ixf of ixf insert into userid.staff
```

- コントロール・センターの「インポート (Import)」ノートブック。「インポート (Import)」ノートブックをオープンするには、以下のようにします。
  1. コントロール・センターから、「表 (Tables)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
  2. 「表 (Tables)」フォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存の表がすべて表示されます。
  3. 目次ペイン内で対象となる表をマウスの右ボタンでクリックし、ポップアップ・メニューから「インポート (Import)」を選択します。「インポート (Import)」ノートブックがオープンします。

  コントロール・センターについての一般情報については、*管理の手引き* を参照してください。詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。
- アプリケーション・プログラミング・インターフェース (API) としての **sqluimpr**。この API については、46ページの『インポート API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、*アプリケーション構築の手引き* を参照してください。

クライアント / サーバー環境でのインポートの使用

リモート・データベースにファイルをインポートする場合は、ストアード・プロシージャーを呼び出してサーバー上でインポートを実行することができます。ストアード・プロシージャーは、以下の場合には呼び出されません。

- アプリケーションとデータベースとでコード・ページが違っている場合。
- インポートされるファイルが複数パーツの PC/IXF ファイルである場合。
- データのインポートに使われる方式が列名か相対列位置のいずれかである場合。
- 指定されたターゲット列リストが 4KB を超えている場合。
- OS/2 または DOS クライアントがディスケットからファイルをインポートしている場合。
- LOBS FROM 文節または `lobsinfile` 修飾子が指定されている場合。
- ASC ファイルに NULL INDICATORS 文節が指定されている場合。

インポートでストアード・プロシージャーを使用する場合は、サーバーにインストールされているデフォルト言語を使ってメッセージ・ファイル内にメッセージが作成されます。クライアントとサーバーとで言語が同じなら、メッセージはアプリケーションの言語になります。

インポート・ユーティリティーは、 sqllib ディレクトリー (または **DB2INSTPROF** レジストリー変数が指定されている場合はそれによって示されるディレクトリー) の tmp サブディレクトリーに、 2 つの一時ファイルを作成します。 1 つのファイルはデータ用、もう 1 つのファイルはインポート・ユーティリティーが生成するメッセージ用です。

サーバー上でのデータの書き込みまたはオープンに関してエラーが出された場合は、以下の点を確認してください。

- ディレクトリーが存在すること。
- それらのファイル用の十分のディスク・スペースがあること。
- インスタンス所有者にそのディレクトリーでの書き込み許可があること。

バッファー挿入によるインポートの使用

区分データベース環境では、インポート・ユーティリティーでバッファー挿入を使用可能にすることができます。これによって、データのインポート時に発生するメッセージ交換が少なくなるため、パフォーマンスが向上します。ただし、バッファー挿入の失敗に関する詳細は戻されないため、このオプションを有効にするのはエラーが報告される心配がない場合だけにしてください。

バッファー挿入を要求するには、DB2 バインド・ユーティリティーを使用します。 INSERT BUF オプションを使って、データベースに対してインポート・パッケージ db2uimpm.bnd を再バインドする必要があります。たとえば、

```
db2 connect to your_database
db2 bind db2uimpm.bnd insert buf
```

注: INSERT_UPDATE パラメーターが指定されたインポート操作の場合、バッファー挿入機能は使用できません。

インポートでの識別列の使用

インポート・ユーティリティーを使って、識別列の入った表にデータをインポートすることができます。識別関連のファイル・タイプ修飾子が使用されない場合、このユーティリティーは次のような規則に従って動作します。

- 識別列が GENERATED ALWAYS の場合、入力ファイル内の対応する行の中に識別列用の値がないか、またはヌル値が明示的に指定されているときに、表行用の識別値が生成されます。識別列に非ヌル値を指定すると、その行は拒否されます (SQL3550W)。

- 識別列が GENERATED BY DEFAULT の場合、ユーザー提供値が指定されていれば、インポート・ユーティリティーはその値を使います。データが欠落しているかまたは明示的にヌルであれば、値が生成されます。

インポート・ユーティリティーは、ユーザー提供の識別値に関して、識別列のデータ・タイプ (SMALLINT、INT、BIGINT、または DECIMAL) の値に対して通常行う以外の余分な妥当性検査を行いません。値が重複していても報告されません。しかも、識別列を持つ表へのデータのインポート時には compound=*x* 修飾子を使用できません。

識別列の入った表の使用を単純化するために、次のような 2 つのファイル・タイプ修飾子がインポート・ユーティリティーでサポートされています。

- identitymissing 修飾子は、入力データ・ファイルに識別列の値が入っていない (ヌルすらない) 場合に、識別列を持つ表のインポートを容易にします。たとえば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 char(30),
                     c2 int generated by default as identity,
                     c3 real,
                     c4 char(1))
```

ユーザーが、データをファイル (import.del) から TABLE1 にインポートするとします。このとき、このデータは、識別列をもたない表からエクスポートされたものであると想定します。以下に、このようなファイルの例を示します。

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

このファイルをインポートする 1 つの方法は、次のように IMPORT コマンドを使って、インポートする列を明示的にリストすることです。

```
db2 import from import.del of del replace into table1 (c1, c3, c4)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをインポートする別の方法は、次のように identitymissing ファイル・タイプ修飾子を使うことです。

```
db2 import from import.del of del modified by identitymissing replace into table1
```

- identityignore 修飾子は、ある意味では identitymissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、インポート・ユーティリティーは、入力データ・ファイルに識別列用の値が入っていても、そのデータを無

視して、各行ごとに識別値を生成します。たとえば、上記で定義したように、ユーザーが次のようなデータをファイル (import.del) から TABLE1 にインポートするとします。

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

ユーザー指定値の 1、2、および 3 が識別列に使われない場合、ユーザーは次のような IMPORT コマンドを使うことができます。

```
db2 import from import.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように identityignore 修飾子を使うと、構文が単純化されます。

```
db2 import from import.del of del modified by identityignore replace into table1
```

識別列を持つ表を IXF ファイルにエクスポートするときには、 IMPORT コマンドの REPLACE_CREATE および CREATE オプションを使って、識別列プロパティーを含む表を再作成することができます。タイプ GENERATED ALWAYS の識別列の入った表からこのような IXF ファイルが作成されている場合、 identityignore 修飾子を指定するのが、データ・ファイルのインポートを正常に完了する唯一の方法です。このようにしなければ、すべての行が拒否されます (SQL3550W)。

インポートでの生成列の使用

インポート・ユーティリティーを使って、 (識別列でない) 生成列の入った表にデータをインポートすることができます。

生成列関連のファイル・タイプ修飾子が使用されない場合、インポート・ユーティリティーは次のような規則に従って動作します。

- 入力ファイル内の対応する行の中に列用の値がないか、またはヌル値が明示的に指定されているときに、生成列用の値が生成されます。生成列に非ヌル値を指定すると、その行は拒否されます (SQL3550W)。
- ヌル可能列でない生成列用にサーバーがヌル値を生成すると、このフィールドが属するデータ行は拒否されます (SQL0407N)。これが起きるのは、たとえば、ヌル可能列でない生成列が 2 つの表列の合計として定義されていて、それらの表列に入力ファイル内でヌル値が指定された場合です。

生成列の入った表の使用を単純化するために、次のような 2 つのファイル・タイプ修飾子がインポート・ユーティリティーでサポートされています。

- generatedmissing 修飾子は、表内に存在する生成列の値が入力データ・ファイル内に入っていない (ヌルすらない) 場合に、生成列を持つ表へのデータ・インポートを容易にします。たとえば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 int,
                     c2 int,
                     g1 int generated always as (c1 + c2),
                     g2 int generated always as (2 * c1),
                     c3 char(1))
```

  ユーザーが、データをファイル (load.del) から TABLE1 にインポートするとします。このとき、このデータは、生成列をもたない表からエクスポートされたものであると想定します。以下に、このようなファイルの例を示します。

```
1, 5, J
2, 6, K
3, 7, I
```

  このファイルをインポートする 1 つの方法は、次のように IMPORT コマンドを使って、インポートする列を明示的にリストすることです。

```
db2 import from import.del of del replace into table1 (c1, c2, c3)
```

  ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをインポートする別の方法は、次のように generatedmissing ファイル・タイプ修飾子を使うことです。

```
db2 import from import.del of del modified by generatedmissing replace into table1
```

- generatedignore 修飾子は、ある意味では generatedmissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、インポート・ユーティリティーは、すべての生成列用のデータが入力データ・ファイルに入っていても、そのデータを無視して、各行ごとに値を生成します。たとえば、上記で定義したように、ユーザーが次のようなデータをファイル (import.del) から TABLE1 にインポートするとします。

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

  ユーザー提供の非ヌル値 10、11、12 (g1 の場合)、および 15、16、17 (g2 の場合) により、行は拒否されます (SQL3550W)。拒否されないようにするには、次のような IMPORT コマンドを発行します。

```
db2 import from import.del of del method P(1, 2, 5) replace into table1 (c1, c2, c3)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように generatedignore 修飾子を使うと、構文が単純化されます。

```
db2 import from import.del of del modified by generatedignore replace into table1
```

エクスポートされた表の再作成

インポート・ユーティリティーを使うと、エクスポート・ユーティリティーによって保管した表を再作成することができます。表は IXF ファイルにエクスポートされていて、エクスポート操作の際に使用した SELECT ステートメントが一定の条件にかなっていることが必要です。 (たとえば SELECT 文節では列名を使用できません。select * だけが可能です。) IXF ファイルから表を作成する場合、元の表のすべての属性が保存されるわけではありません。たとえば、参照制約、外部キー定義、およびユーザー定義のデータ・タイプは保存されません。元の表の属性で保存されるのは以下のとおりです。

- 基本キーの名前および定義
- 固有制約の名前と定義。ただし他のタイプの制約やトリガーは格納されません。
- 列情報
  - 列名
  - 列データ・タイプ。ユーザー定義の特殊タイプはその基本タイプとして保存されます。
  - 識別プロパティー
  - 長さ (lob_file タイプの場合を除く)
  - コード・ページ (該当する場合)
  - DATALINK オプション
  - 識別オプション
  - ヌル可能列またはヌル不可列のどちらとして列が定義されているか
  - 定数のデフォルト値 (ある場合)。ただし他のタイプのデフォルト値は含まれません。
- 索引情報
  - 索引名
  - 索引の作成者名
  - 列名と、昇順または降順のどちらで各列が分類されるか
  - 索引が固有索引として定義されているかどうか
  - 索引がクラスター化されているかどうか

  - 索引で逆走査が可能かどうか
  - *pctfree* 値
  - *minpctused* 値

元の表の属性のうち、以下のものは保存されません。

- ソースが、通常の表、要約表、視点、またはこれらのソースの全部または一部から取られた列集合のうちのどれであったか
- 表情報
  - 要約表定義 (該当する場合)
  - 要約表オプション (該当する場合)
  - 表スペース・オプション。ただしこの情報は、 IMPORT コマンドを使って指定することができます。
- 列情報
  - 定数値以外の任意のデフォルト値
  - LOB オプション (ある場合)
  - create table ステートメントの references 文節 (ある場合)
  - 参照制約 (ある場合)
  - 検査制約 (ある場合)
  - 生成列オプション (ある場合)
- 索引情報
  - 組み込み列 (ある場合)

ラージ・オブジェクト (LOB) のインポート

ラージ・オブジェクト (LOB) 列にインポートする場合、データは残りの列データと同じファイルからのものか、または別のファイルからのものです。後者の場合、それぞれの LOB インスタンスごとに 1 つのファイルがあります。

メイン入力データ・ファイル内の列には、インポート・データが入っている (デフォルト) か、またはインポート・データが格納されているファイルの名前が入っています。

注:

1. LOB データをメイン入力データ・ファイルに格納する場合、32KB を超えるデータは許されません。切り捨ての警告は無視されます。
2. LOB データすべてをメイン・ファイルに格納するか、それぞれの LOB を別ファイルに格納するかのどちらかでなければなりません。メイン・ファイ

ルに LOB データとファイル名を混在させることはできません。 LOB 値を別ファイルからインポートするには、lobsinfile 修飾子 (60ページの『ファイル・タイプ修飾子 (インポート)』を参照) と、LOBS FROM 文節 (38ページの『IMPORT コマンド』を参照) を使用します。

ユーザー定義特殊タイプ (UDT) のインポート

インポート・ユーティリティーは、ユーザー定義特殊タイプ (UDT) をそれと類似の基本データ・タイプに自動的にキャストします。それにより、UDT を基本データ・タイプに明示的にキャストする手間が省けます。キャストによって、SQL で UDT と基本データ・タイプとの間の比較が可能になります。

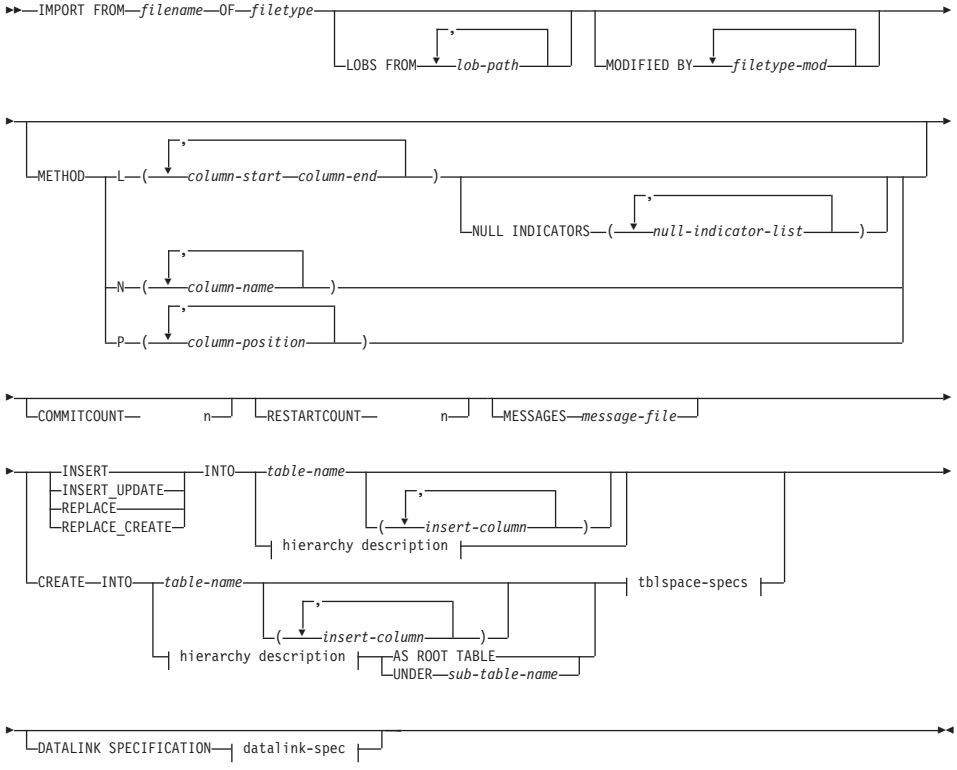
インポート時の表のロッキング

デフォルトでは、インポート・ユーティリティーは、分離レベル RR (反復可能読み取り) でデータベースにバインドされます。多数の行を表にインポートする場合、既存のロックが排他ロックに格上げされることがあります。同じ表を処理している別のアプリケーションが何らかの行ロックを保持していると、そのロックが排他ロックに格上げされたときにデッドロックが起きます。これが起きないようにするため、インポート・ユーティリティーは、操作の開始時点で表の排他ロックを要求します。

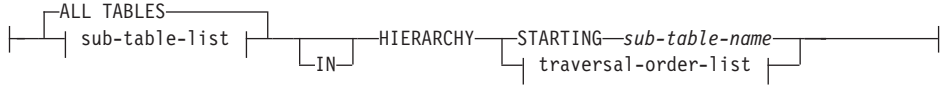
表に対するロックを保持することには 2 つの意義があります。第 1 に、他のアプリケーションがインポート・ターゲット表に対する表ロックまたは行ロックを保持していると、インポート・ユーティリティーは、そのようなアプリケーションすべてが変更内容をコミットまたはロールバックするまで待機します。第 2 に、インポートが実行されている間、ロックを要求している他のすべてのアプリケーションは、そのインポート操作の完了を待機します。

IMPORT コマンド

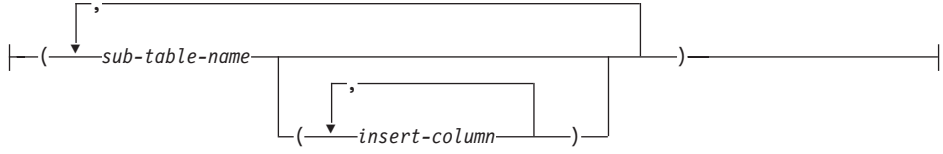
コマンド構文



hierarchy description:

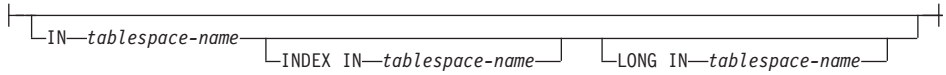


sub-table-list:

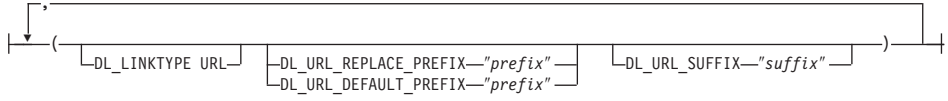


traversal-order-list:

tblspace-specs:



datalink-spec:



コマンド・パラメーター

ALL TABLES
暗黙のキーワード (階層のみ)。階層をインポートする場合、デフォルトでは走査順序で指定されたすべての表をインポートします。

AS ROOT TABLE
1 つ以上の副表を独立した表階層として作成します。

COMMITCOUNT n
n 個のレコードをインポートするたびに COMMIT を実行します。

CREATE
表定義と行の内容を作成します。データが DB2 表、副表、または階層からエクスポートされたものである場合は、索引が作成されます。このオプションを階層に対して使用する場合、データが DB2 からエクスポートされたものであるなら、タイプ階層も作成されます。このオプションは、IXF ファイルでのみ使用できます。

注: データが MVS ホスト・データベースからエクスポートされたものである場合、そのデータに長さが 254 未満 (ページ・サイズで計算) の LONGVAR フィールドが含まれているなら、行が長すぎるために CREATE が失敗することがあります。その場合は、表を手動で作成してから、INSERT を指定して IMPORT を起動するか、あるいは LOAD コマンドを使用してください。

DATALINK SPECIFICATION
各 DATALINK 列ごとに、列指定を括弧で囲んで 1 つだけ指定できま

す。各列指定は、1 つ以上の DL_LINKTYPE、接頭部、および DL_URL_SUFFIX 指定で構成されます。接頭部の指定は DL_URL_REPLACE_PREFIX か DL_URL_DEFAULT_PREFIX のどちらかです。

表で定義されている DATALINK 列の数と同数の DATALINK 列指定を指定できます。列指定の順序は、*insert-column* リスト内での DATALINK 列の順序に従います。 *insert-column* リストが指定されていない場合は、表定義の中での順序に従います。

DL_LINKTYPE

これを指定する場合、列定義の LINKTYPE と一致している必要があります。それで、列定義で LINKTYPE URL を指定しているなら、DL_LINKTYPE URL が受け入れ可能になります。

DL_URL_DEFAULT_PREFIX ″**prefix**″

これを指定した場合、これは同じ列内のすべての DATALINK 値のデフォルト接頭部となります。このコンテキストで接頭部とは、URL 指定のうち ″スキーム・ホスト・ポート″ 部分のことです。 (分散ファイル・システム (DFS) の場合、接頭部とは、 URL 指定のうち ″スキーム・セル名ファイル・スペース接合″ 部分のことです。)

接頭部の例を以下に示します。

```
"http://server"
"file://server"
"file:"
"http://server:80"
"dfs://.../cellname/fs"
```

DL_URL_DEFAULT_PREFIX でデフォルト接頭部が指定されているなら、列のデータに接頭部がない場合、ヌルでない列値にはデフォルト接頭部が付けられます。

たとえば、DL_URL_DEFAULT_PREFIX でデフォルト接頭部 "http://toronto" が指定されている場合、

- 列の入力値 ″/x/y/z″ は ″http://toronto/x/y/z″ として格納されます。
- 列の入力値 ″http://coyote/a/b/c″ は ″http://coyote/a/b/c″ として格納されます。
- 列の入力値がヌルなら、ヌルとして格納されます。

DL_URL_REPLACE_PREFIX ″**prefix**″

この文節は、以前にエクスポート・ユーティリティーによって生成されたデータのロードまたはインポートにおいて、データ内のホスト名を別のホスト名に一括して置換したい場合に役立ちます。これを指定した場

合、これはヌルでないすべての 列値の接頭部になります。列値に接頭部がある場合、それは置換されます。列値に接頭部がない場合は、列値の接頭部として DL_URL_REPLACE_PREFIX で指定された接頭部が付けられます。分散ファイル・システム (DFS) の場合、接頭部とは、URL 指定のうち ″スキーム・セル名ファイル・スペース接合″ 部分のことです。

たとえば、DL_URL_REPLACE_PREFIX で接頭部 "http://toronto" が指定されている場合、

- 列の入力値 ″/x/y/z″ は ″http://toronto/x/y/z″ として格納されます。
- 列の入力値 ″http://coyote/a/b/c″ は ″http://toronto/a/b/c″ として格納されます。 ″coyote″ から ″toronto″ に置換されたことに注意してください。
- 列の入力値がヌルなら、ヌルとして格納されます。

DL_URL_SUFFIX ″suffix″
これを指定した場合、列のうちヌルでないすべての列値にこれが付加されます。実際には、DATALINK 値の URL 部分の ″パス″ 構成要素に付加されます。

FROM filename
インポートするデータが入っているファイルを指定します。パスを省略した場合、現行作業ディレクトリーが使用されます。

HIERARCHY
階層データをインポートするように指定します。

IN tablespace-name
どの表スペースの中に表を作成するかを指定します。表スペースは、すでに存在している REGULAR (正規) 表スペースでなければなりません。他の表スペースを指定しない場合、表のすべての部分がこの表スペースに格納されます。この文節を指定しない場合は、許可 ID によって作成される表スペースに表が作成されます。それがみつからなければ、デフォルトの表スペース USERSPACE1 に表が入れられます。USERSPACE1 が除去されていた場合、作成は失敗します。

INDEX IN tablespace-name
どの表スペースの中に表の索引を作成するかを指定します。このオプションを指定できるのは、IN 文節で指定されている 1 次表スペースが DMS 表スペースである場合だけです。指定する表スペースは、すでに存在している REGULAR (正規) DMS 表スペースでなければなりません。

注: 索引が入れられる表スペースを指定できるのは、表を作成する場合だけです。

insert-column
データの挿入先となる表または視点の列の名前を指定します。

INSERT
既存の表データに変更を加えることなく、インポート・データを表に追加します。

INSERT_UPDATE
インポート・データの行をターゲット表に追加するか、またはターゲット表のうち基本キーが一致する既存の行があればそれを更新します。

INTO table-name
データのインポート先となるデータベース表を指定します。この表として、システム表、宣言された一時表、または要約表を指定することはできません。

INSERT、INSERT_UPDATE、または REPLACE には別名を使用できます。ただし、下位レベルのサーバーの場合は、完全修飾または非修飾の表名を指定する必要があります。修飾された表名は、*schema.tablename* という形式です。 *schema* は、表を作成したユーザー名です。

LOBS FROM lob-path
LOB ファイルを格納する 1 つ以上のパスを指定します。 LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) のうち LOB 列にロードされる列に格納されます。
`lobsinfile` 修飾子が指定されていなければ、このオプションは無視されます。

LONG IN tablespace-name
長い列の値を格納する表スペースを指定します。長い列とは、LONG VARCHAR、LONG VARGRAPHIC、LOB データ・タイプ、またはこれらのいずれかをソース・タイプとする特殊タイプの列のことです。このオプションを指定できるのは、IN 文節で指定されている 1 次表スペースが DMS 表スペースである場合だけです。表スペースは、すでに存在している LONG DMS 表スペースでなければなりません。

MESSAGES message-file
インポート操作中に発生する警告およびエラー・メッセージの出力先を指定します。ファイルがすでに存在する場合、インポート・ユーティリティーは情報をそこに追加します。ファイルの完全パスを指定しなけれ

ば、ユーティリティーはデフォルト・ドライブと現行ディレクトリーを宛先として使用します。 *message-file* を省略した場合、メッセージは標準出力に出力されます。

METHOD

L データのインポート元となる開始列番号と終了列番号を指定します。列番号は、データ行の先頭からのバイト・オフセットです。番号は 1 から付けられます。

注: この方式は ASC ファイル専用であり、そのファイル・タイプで有効な唯一のオプションです。

N インポートする列の名前を指定します。

注: この方式は、IXF ファイルでのみ使用できます。

P インポートする入力データ・フィールドの索引 (1 から始まる番号付き) を指定します。

注: この方式は IXF または DEL ファイル専用であり、DEL ファイル・タイプではこれが唯一の有効なオプションです。

MODIFIED BY filetype-mod

追加オプションを指定します (60ページの表5 を参照)。

NULL INDICATORS null-indicator-list

このオプションを使用できるのは、 METHOD L パラメーターを指定したときだけです。つまり、入力ファイルが ASC ファイルであるときです。ヌル標識リストは、各ヌル標識フィールドの列番号を指定する正整数をコンマで区切ったリストです。列番号は、データ行の先頭からのヌル標識フィールドのバイト・オフセットです。ヌル標識リスト内には、 METHOD L パラメーターで定義されているデータ・フィールドごとに 1 つずつ項目がなければなりません。列番号ゼロは、対応するデータ・フィールドに常にデータが入っていることを示します。

ヌル標識列の値が `Y` なら、それはその列データがヌルになることを指定します。ヌル標識列の値が `Y` *以外の* 文字なら、それはその列データがヌルではなく、METHOD L オプションで指定される列データがインポートされることを指定します。

ヌル標識の文字は MODIFIED BY オプションで変更できます (60ページの表5 の中の `nullindchar` 修飾子についての説明を参照)。

OF filetype

入力ファイル内のデータの形式を指定します。

- ASC (区切りなし ASCII 形式)
- DEL (区切り付き ASCII 形式)。これはさまざまなデータベース・マネージャーおよびファイル・マネージャー・プログラムで使用されます。
- WSF (ワークシート形式)。以下のようなプログラムで使用されます。
  - ロータス 1-2-3
  - Lotus Symphony
- IXF (統合交換フォーマット、PC バージョン)。同じ DB2 表または別の DB2 表からエクスポートされたものであることを意味します。IXF ファイルには、表定義や既存の索引の定義も入っています。ただし、SELECT ステートメントで列が指定されている場合は例外です。

ファイル形式については、 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

REPLACE

データ・オブジェクトを切り捨てることにより表から既存のデータをすべて削除してから、インポート・データを挿入します。表定義と索引定義は変更されません。このオプションを使用できるのは、表が存在する場合だけです。 DATALINK 列のある表に対しては無効です。階層間でのデータ移動にこのオプションを使用した場合、個々の副表ではなく階層全体についてのデータだけが置換できます。

REPLACE_CREATE

表が存在する場合、データ・オブジェクトを切り捨てることにより表から既存のデータをすべて削除してから、インポート・データを挿入します。表定義や索引定義は変更しません。

表が存在しない場合は、表定義と索引定義、および行の内容を作成します。

このオプションは、IXF ファイルでのみ使用できます。 DATALINK 列のある表に対しては無効です。階層間でのデータ移動にこのオプションを使用した場合、個々の副表ではなく階層全体についてのデータだけが置換できます。

RESTARTCOUNT n

インポート操作をレコード *n* + 1 から始めることを指定します。最初の *n* 個のレコードはスキップされます。

STARTING sub-table-name

sub-table-name から始まるデフォルト順序を要求する階層専用のキーワード。 PC/IXF ファイルの場合のデフォルト順序は、入力ファイルに保管されている順序です。 PC/IXF ファイル形式の場合、デフォルト順序が唯一の有効な順序です。

sub-table-list

INSERT または INSERT_UPDATE オプションを指定したタイプ表においては、データのインポート先の副表を示すために副表名のリストが使用されます。

traversal-order-list

INSERT、INSERT_UPDATE、または REPLACE オプションを指定したタイプ表では、階層内の副表のインポートの走査順序を示すために副表名のリストが使用されます。

UNDER sub-table-name

1 つ以上の副表を作成するための親表を指定します。

インポート API

C API 構文

```
/* File: sqlutil.h */
/* API: Import */
/* ... */
SQL_API_RC SQL_API_FN
  sqluimpr (
    char * pDataFileName,
    sqlu_media_list * pLobPathList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pMsgFileName,
    short CallerAction,
    struct sqluimpt_in*  pImportInfoIn,
    struct sqluimpt_out*  pImportInfoOut,
    sqlint32 * pNullIndicators,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

汎用 API 構文

```
/* File: sqlutil.h */
/* API: Import */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgimpr (
    unsigned short DataFileNameLen,
    unsigned short FileTypeLen,
    unsigned short MsgFileNameLen,
    char * pDataFileName,
    sqlu_media_list * pLobPathList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pMsgFileName,
    short CallerAction,
    struct sqluimpt_in*  pImportInfoIn,
    struct sqluimpt_out*  pImportInfoOut,
    sqlint32 * NullIndicators,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

API パラメーター

DataFileNameLen
入力。入力ファイル名の長さをバイト数で表す 2 バイトの符号なし整数。

FileTypeLen
入力。入力ファイルのタイプの長さをバイト数で表す 2 バイトの符号なし整数。

MsgFileNameLen
入力。メッセージ・ファイル名の長さをバイト数で表す 2 バイトの符号なし整数。

pDataFileName
入力。データのインポート元となる外部入力ファイルのパスおよび名前を入れるストリング。

pLobPathList
入力。 *media_type* SQLU_LOCAL_MEDIA、および *sqlu_media_entry* 構造体を使用する *sqlu_media_list* で、クライアント上で LOB ファイルのある場所を示すパスのリストを入れます。

詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

pDataDescriptor

入力。外部ファイルからのインポートで選択する列に関する情報が入っている *sqldcol* 構造体へのポインター。このパラメーターで与えられる情報の残りの部分をインポート・ユーティリティーがどう解釈するかは、*dcolmeth* フィールドの値によって決まります。このパラメーターの有効な値 (`sqlutil` で定義される) は、以下のとおりです。

SQL_METH_N

名前。外部入力ファイルからの列は、列名によって選択されます。

SQL_METH_P

位置。外部入力ファイルからの列は、列の位置によって選択されます。

SQL_METH_L

ロケーション。外部入力ファイルからの列は、列のロケーションによって選択されます。以下のいずれかの条件のためにロケーション対が無効である場合、データベース・マネージャーはそのインポート呼び出しを拒否します。

- 開始または終了ロケーションが、1 から符号付き 2 バイト整数の最大値までの範囲に収まっていない。
- 終了ロケーションが開始ロケーションよりも小さい。
- ロケーション対によって定義される入力列の幅が、ターゲット列のタイプおよび長さと互換性がない。

ロケーション対の 2 つのロケーションが 0 に等しい場合、それはヌル可能な列にヌルを埋めることを示しています。

SQL_METH_D

デフォルト。 *pDataDescriptor* がヌルであるか `SQL_METH_D` に設定されている場合、外部入力ファイルからの列のデフォルト選択が実行されます。この場合、列の数と列指定配列はどちらも無視されます。外部入力ファイルのデータの最初の *n* 列がその自然の順序に従って取られます (*n* はデータのインポート先のデータベースの列数)。

詳しくは、 *管理 API 解説書* の『SQLDCOL』を参照してください。

pActionString

入力。 *sqlchar* 構造体へのポインター。この構造体には、2 バイト長フィールドに続いて、データのインポート先となる列を指定する文字配列が含まれています。

文字配列の形式は以下のとおりです。

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
[DATALINK SPECIFICATION datalink-spec]
```

INSERT

既存の表データに変更を加えることなく、インポート・データを表に追加します。

INSERT_UPDATE

基本キー値が表にない場合はインポート行を追加し、基本キー値が表にある場合はインポート行によって更新します。このオプションが有効なのは、ターゲット表に基本キーがあり、かつインポートのターゲット列として指定された (または暗黙の) リストに基本キーのすべての列が含まれている場合だけです。このオプションは視点には適用できません。

REPLACE

表オブジェクトを切り捨てることにより表から既存のデータをすべて削除してから、インポート・データを挿入します。表定義と索引定義は変更されません。 (`indexixf` が *FileTypeMod* に含まれており、*FileType* が `SQL_IXF` の場合、索引は削除されて置換されます。) 表がまだ定義されていない場合は、エラーが戻されます。

考慮事項: 既存のデータが削除された後にエラーが発生した場合、そのデータは失われます。

CREATE

指定された表が定義されていない場合に、指定された PC/IXF ファイル内の情報を使って表定義と行の内容を作成します。以前に DB2 でエクスポートされたファイルの場合は、索引も作成されます。表がすでに定義されている場合は、エラーが戻されます。このオプションは PC/IXF ファイル形式に対してのみ有効です。

REPLACE_CREATE
指定された表が定義されている場合、PC/IXF ファイル内の PC/IXF 行情報を使って表の内容を置き換えます。表がまだ定義されていない場合は、指定された PC/IXF ファイル内の情報を使って表定義と行の内容が作成されます。以前に DB2 でエクスポートされた PC/IXF ファイルの場合は、索引も作成されます。このオプションは PC/IXF ファイル形式に対してのみ有効です。

考慮事項: 既存のデータが削除された後にエラーが発生した場合、そのデータは失われます。

tname データの挿入先となる表、タイプ表、視点、またはオブジェクト視点のいずれかの名前。 REPLACE、INSERT_UPDATE、または INSERT には別名を使用できます。ただし、下位レベルのサーバーの場合は、完全修飾または非修飾の名前を指定する必要があります。視点の場合、読み取り専用視点であってはなりません。

tcolumn-list
データの挿入先となる表または視点の列名のリスト。列名と列名の間はコンマで区切ってください。列名を指定しない場合、CREATE TABLE または ALTER TABLE ステートメントで定義されている列名が使用されます。タイプ表に列リストが指定されていない場合、各副表の中のすべての列にデータが挿入されます。

sub-table-name
CREATE オプションを指定して 1 つ以上の副表を作成する場合の親表を指定します。

ALL TABLES
暗黙のキーワード (階層のみ)。階層をインポートする場合、デフォルトでは *traversal-order-list* で指定されたすべての表をインポートします。

HIERARCHY
階層データをインポートするように指定します。

STARTING
階層専用のキーワード。指定された副表名から始めるというデフォルト順序を使用するよう指定します。

UNDER
階層および CREATE 専用のキーワード。指定された副表の下に新しい階層、副階層、または副表を作成するよう指定します。

AS ROOT TABLE
階層および CREATE 専用のキーワード。独立型の階層として新しい階層、副階層、または副表を作成するよう指定します。

DATALINK SPECIFICATION *datalink-spec*
DB2 データ・リンクに関連するパラメーターを指定します。これらのパラメーターは IMPORT コマンドの場合と同じ構文を使って指定できます (38ページの『IMPORT コマンド』を参照)。

tname および *tcolumn-list* パラメーターは SQL INSERT ステートメントの *tablename* および *colname* リストに対応するものであり、それらと同じ制限が当てはまります。

tcolumn-list 内の列と外部列 (指定された場合または暗黙指定された場合) は、リストまたは構造体の中の位置に従って突き合わされます (*sqldcol* 構造体の中で指定された最初の列のデータは、表または視点のフィールドのうち *tcolumn-list* の最初の要素に対応するものに挿入されます。)

指定された列の数が等しくない場合、実際に処理される列の数は 2 つの数のうちの小さい方になります。その結果として、表フィールドのうちヌル可能でない位置に値がないためにエラーになったり、いくつかの外部ファイル列が無視されるために通知メッセージが出されたりすることがあります。

詳しくは、*管理 API 解説書* の『SQLCHAR』を参照してください。

pFileType
入力。外部ファイル内のデータの形式を示すストリング。サポートされる外部ファイル形式 (`sqlutil` で定義される) は、以下のとおりです。

SQL_ASC
区切りなし ASCII。

SQL_DEL
区切り付き ASCII。dBase、BASIC、IBM パーソナル・デシジ

ョン・シリーズのプログラム、その他多くのデータベース・マネージャーおよびファイル・マネージャーとのデータ交換に使用できます。

SQL_IXF
PC バージョンの IXF (統合交換フォーマット)。表からのデータ・エクスポートとして望ましい方式であり、後でそのデータを同じ表または別のデータベース・マネージャー表へインポートできます。

SQL_WSF
ワークシート形式。Lotus Symphony およびロータス 1-2-3 のプログラムとのデータ交換に使用できます。

ファイル形式については、 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

pFileTypeMod
入力。構造体へのポインター。この構造体には、2 バイト長フィールドに続いて、1 つ以上の処理オプションを指定する文字の配列が含まれます。このポインターがヌルか、またはポイント先の構造体の文字数が 0 の場合、このアクションはデフォルト指定を選択したものと解釈されます。

サポートされているすべてのファイル・タイプですべてのオプションが使えるわけではありません。

詳しくは、*管理 API 解説書* の『SQLCHAR』、および 60ページの『ファイル・タイプ修飾子 (インポート)』を参照してください。

pMsgFileName
入力。ユーティリティーが戻すエラー、警告、通知メッセージの宛先を入れるストリング。オペレーティング・システム・ファイルのパスと名前、または標準装置を指定できます。ファイルがすでに存在する場合は、そこに追加されます。ファイルが存在しなければ作成されます。

CallerAction
入力。呼び出し元が要求するアクション。有効な値 (`sqlutil` で定義される) は、以下のとおりです。

SQLU_INITIAL
初期呼び出し。 API の最初の呼び出しでは、この値を使用しなければなりません。

初期呼び出しまたはそれ以降の呼び出しから戻る際、要求されたインポート操作を完了する前に呼び出し元アプリケーションでなんらかのアクションを実行することが必要な場合は、この呼び出し元のアクションを以下のいずれかに設定する必要があります。

SQLU_CONTINUE
処理を続行します。この値は API の後続呼び出しでのみ使用できます。つまり、初期呼び出しから戻った時点でユーティリティーがユーザー入力 (「テープの終わり」状態への応答など) を要求した場合に、その後で使用します。これは、ユーティリティーの要求したユーザー処置が完了し、ユーティリティーが初期要求の処理を続行できるということを指定します。

SQLU_TERMINATE
処理を終了します。この値は API の後続呼び出しでのみ使用できます。つまり、初期呼び出しから戻った時点でユーティリティーがユーザー入力 (「テープの終わり」状態への応答など) を要求した場合に、その後で使用します。これは、ユーティリティーの要求したユーザー処置が実行されず、ユーティリティーが初期要求の処理を終了することを指定します。

pImportInfoIn
入力。追加の入力パラメーターが入っている *sqluimpt_in* 構造体へのポインター (オプション)。この構造体については、
57ページの『SQLUIMPT-IN データ構造体』を参照してください。

pImportInfoOut
出力。追加の出力パラメーターが入っている *sqluimpt_out* 構造体へのポインター (オプション)。この構造体については、
58ページの『SQLUIMPT-OUT データ構造体』を参照してください。

NullIndicators
入力。 ASC ファイル専用。列データがヌル可能かどうかを示す整数の配列。この配列内の要素数は、入力ファイル内の列数と同じでなければなりません。この配列の要素とデータ・ファイルからインポートされる列とは、順番に 1 対 1 で対応します。したがって、要素の数は *pDataDescriptor* パラメーターの *dcolnum* フィールドと同じでなければなりません。配列の各要素には、ヌル標識フィールドとして使用されるデータ・ファイル内の列を識別する番号が入れられます。それが 0 の場合は、その表列がヌル可能でないことを示しています。その要素が 0 でない場合、データ・ファイル内の指定された列の内容は Y または N でなければなりません。 Y はその表列のデータがヌルであり、N はその表列のデータがヌルでないことを示します。

pReserved
将来の利用のために予約済み。

pSqlca
出力。 *sqlca* 構造体へのポインター。この構造体については、 *管理 API 解説書* の『SQLCA』を参照してください。

REXX API 構文

```
IMPORT FROM datafile OF filetype
[MODIFIED BY :filetmod]
[METHOD {L|N|P} USING :dcoldata]
[COMMITCOUNT :commitcnt] [RESTARTCOUNT :restartcnt]
MESSAGES msgfile
{INSERT|REPLACE|CREATE|INSERT_UPDATE|REPLACE_CREATE}
INTO tname [(:columns)]
[OUTPUT INTO :output]

CONTINUE IMPORT

STOP IMPORT
```

REXX API パラメーター

datafile
データのインポート元となるファイルの名前。

filetype
インポート・ファイル内のデータの形式。サポートされるファイル形式は以下のとおりです。

DEL 区切り付き ASCII

ASC 区切りなし ASCII

WSF ワークシート形式

IXF PC バージョンの IXF (統合交換フォーマット)

filetmod
追加の処理オプションを入れるホスト変数 (60ページの『ファイル・タイプ修飾子 (インポート)』を参照)。

L|N|P 外部入力ファイル内で列の選択に使用する方式を指定する文字。有効な値は以下のとおりです。

L ロケーション

N 名前

P 位置

dcoldata

外部ファイルからのインポートで選択する列に関する情報が入っている REXX 複合ホスト変数。この構造体の内容は、*method* の指定によって異なります。以下の記述の中の XXX は、ホスト変数の名前を表します。

- ロケーション方式

  XXX.0 変数の残りの部分の要素の数。

  XXX.1 入力ファイル内でのこの列の開始ロケーションを示す数値。この列はデータベース表で最初の列になります。

  XXX.2 列の終了ロケーションを示す数。

  XXX.3 入力ファイル内でのこの列の開始ロケーションを示す数値。この列はデータベース表で第 2 の列になります。

  XXX.4 列の終了ロケーションを示す数。

  XXX.5 以下同様。

- 名前方式

  XXX.0 ホスト変数に含まれる列名の数。

  XXX.1 最初の名前。

  XXX.2 第 2 の名前。

  XXX.3 以下同様。

- 位置方式

  XXX.0 ホスト変数に含まれる列位置の数。

  XXX.1 外部入力ファイル内の列位置。

  XXX.2 外部入力ファイル内の列位置。

  XXX.3 以下同様。

tname ターゲット表またはターゲット視点の名前。読み取り専用の視点にデータをインポートすることはできません。

columns

データの挿入先となる表または視点の中の列の名前を内容とする REXX ホスト変数。以下の記述の中の XXX は、ホスト変数の名前を表します。

XXX.0 列の数。

XXX.1 最初の列名。

XXX.2 第 2 の列名。

XXX.3 以下同様。

msgfile

エラーおよび警告メッセージの送信先のファイル、パス、または装置名。

commitcnt

commitcnt 個のレコードをインポートするたびに COMMIT を実行します。

restartcnt

インポート操作をレコード *restartcnt* + 1 から始めることを指定します。最初の *restartcnt* 個のレコードはスキップされます。

output

インポート操作から渡される情報を入れる REXX 複合ホスト変数。以下の記述の中の XXX は、ホスト変数の名前を表します。

XXX.1 インポート操作中に外部入力ファイルから読み取られたレコード数。

XXX.2 挿入または更新が始まる前にスキップされたレコード数。

XXX.3 ターゲット表に挿入された行数。

XXX.4 ターゲット表のうち、インポートされたレコードからの情報によって更新された行数。

XXX.5 インポートできなかったレコードの数。

XXX.6 正常にインポートされ、データベースにコミットされたレコード数。これには挿入、更新、スキップ、およびリジェクトされた行が含まれます。

SQLUIMPT-IN データ構造体

この構造体は、 46ページの『インポート API』に情報を渡すのに使用します。

表 3. SQLUIMPT-IN 構造体のフィールド

フィールド名	データ・タイプ	説明
SIZEOFSTRUCT	INTEGER	構造体のサイズ (バイト数)。
COMMITCNT	INTEGER	データベースにコミットすることになるインポート・レコード数。 *commitcnt* 個のレコードがインポートされるたびに COMMIT が実行されます。
RESTARTCNT	INTEGER	レコードの挿入または更新を始める前にスキップするレコード数。いくつかのレコードがデータベースにコミットされた後にレコードをインポートしようとして失敗した場合は、このパラメーターを使用してください。指定する値は、次回のインポート操作の開始点を表します。

言語構文

C 構造体

```
/* File: sqlutil.h */
/* Structure: SQLUIMPT-IN */
/* ... */
SQL_STRUCTURE sqluimpt_in
{
  sqluint32       sizeOfStruct;
  sqluint32       commitcnt;
  sqluint32       restartcnt;
};
/* ... */
```

COBOL 構造体

```
* File: sqlutil.cbl
01 SQL-UIMPT-IN.
    05 SQL-SIZE-OF-UIMPT-IN   PIC 9(9) COMP-5 VALUE 12.
    05 SQL-COMMITCNT          PIC 9(9) COMP-5 VALUE 0.
    05 SQL-RESTARTCNT         PIC 9(9) COMP-5 VALUE 0.
*
```

SQLUIMPT-OUT データ構造体

この構造体は、 46ページの『インポート API』からの情報を渡すのに使用されます。

表 4. SQLUIMPT-OUT 構造体のフィールド

フィールド名	データ・タイプ	説明
SIZEOFSTRUCT	INTEGER	構造体のサイズ (バイト数)。
ROWSREAD	INTEGER	インポート中にファイルから読み取られたレコード数。
ROWSSKIPPED	INTEGER	挿入または更新が始まる前にスキップされたレコード数。
ROWSINSERTED	INTEGER	ターゲット表に挿入された行数。
ROWSUPDATED	INTEGER	ターゲット表のうち、インポートされたレコード (基本キーがすでに表内に存在するレコード) からの情報によって更新された行の数。
ROWSREJECTED	INTEGER	インポートできなかったレコードの数。
ROWSCOMMITTED	INTEGER	正常にインポートされ、データベースにコミットされたレコード数。

言語構文

C 構造体

```
/* File: sqlutil.h */
/* Structure: SQLUIMPT-OUT */
/* ... */
SQL_STRUCTURE sqluimpt_out
{
  sqluint32       sizeOfStruct;
  sqluint32       rowsRead;
  sqluint32       rowsSkipped;
  sqluint32       rowsInserted;
  sqluint32       rowsUpdated;
  sqluint32       rowsRejected;
  sqluint32       rowsCommitted;
};
/* ... */
```

COBOL 構造体

```
* File: sqlutil.cbl
01 SQL-UIMPT-OUT.
    05 SQL-SIZE-OF-UIMPT-OUT  PIC 9(9) COMP-5 VALUE 28.
    05 SQL-ROWSREAD           PIC 9(9) COMP-5 VALUE 0.
    05 SQL-ROWSSKIPPED        PIC 9(9) COMP-5 VALUE 0.
    05 SQL-ROWSINSERTED       PIC 9(9) COMP-5 VALUE 0.
    05 SQL-ROWSUPDATED        PIC 9(9) COMP-5 VALUE 0.
    05 SQL-ROWSREJECTED       PIC 9(9) COMP-5 VALUE 0.
    05 SQL-ROWSCOMMITTED      PIC 9(9) COMP-5 VALUE 0.
*
```

ファイル・タイプ修飾子 (インポート)

表 5. 有効なファイル・タイプ修飾子 (インポート)

修飾子	説明
すべてのファイル形式	
compound=*x*	*x* は、1 ～ 100 の数値です。非アトミック複合 SQL を使用してデータを挿入します。毎回 *x* 個のステートメントが試行されます。 この修飾子を指定した場合、トランザクション・ログに十分な大きさがないと、インポート操作は失敗します。トランザクション・ログには、 COMMITCOUNT で指定された行数、またはデータ・ファイル内の行数 (COMMITCOUNT が指定されていない場合) が入るだけの大きさが必要です。それで、トランザクション・ログがあふれるのを避けるため、COMMITCOUNT を指定するようお勧めします。 この修飾子には、INSERT_UPDATE モード、階層形式の表、および usedefaults、identitymissing、 identityignore、generatedmissing、 generatedignore の各修飾子との互換性はありません。
generatedignore	この修飾子はインポート・ユーティリティーに対して、すべての生成列のデータがデータ・ファイル内にあるものの、それらのデータは無視するべきものであることを通知します。その結果、生成列のすべての値はユーティリティーが生成します。この修飾子は、generatedmissing 修飾子とともに使用することはできません。
generatedmissing	この修飾子が指定されている場合、ユーティリティーは、生成列のデータが入力データ・ファイルに入っていない (ヌルも入っていない) ものと見なし、行ごとに値を生成します。この修飾子は、generatedignore 修飾子とともに使用することはできません。
identityignore	この修飾子はインポート・ユーティリティーに対して、識別列のデータがデータ・ファイル内にあるものの、それらのデータは無視するべきものであることを通知します。その結果、すべての識別値はユーティリティーが生成します。GENERATED ALWAYS 識別列と GENERATED BY DEFAULT 識別列のどちらの場合も動作は同じになります。つまり、GENERATED ALWAYS 列の場合には、拒否される行はありません。この修飾子は、identitymissing 修飾子とともに使用することはできません。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
identitymissing	この修飾子が指定されている場合、ユーティリティーは、識別列のデータが入力データ・ファイルに入っていない (ヌルも入っていない) ものと見なし、行ごとに値を生成します。GENERATED ALWAYS 識別列と GENERATED BY DEFAULT 識別列のどちらの場合も動作は同じになります。この修飾子は、`identityignore` 修飾子とともに使用することはできません。
lobsinfile	*lob-path* で、LOB 値を含むファイルへのパスを指定します。
no_type_id	単一の副表にインポートする場合のみ有効です。多くの場合、正規の表からデータをエクスポートした後、この修飾子を使用してインポート操作を起動してデータを単一の副表に変換する場合に使用します。
nodefaults	ターゲット表の列に対するソース列を明示的に指定せず、表の列がヌル可能でない場合、デフォルト値はロードされません。このオプションを指定しない場合、ターゲット表のいずれかの列に対するソース列を明示的に指定しない場合に、以下のいずれかのようになります。 • 列のデフォルト値を指定できる場合は、デフォルト値がロードされます。 • 列がヌル可能で、その列にデフォルト値が指定できない場合は、ヌルがロードされます。 • 列がヌル可能ではなく、その列にデフォルト値も指定できない場合は、エラーが戻され、ユーティリティーは処理を停止します。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
usedefaults	ターゲット表の列に対するソース列を指定したが、1 つ以上の行インスタンスにデータが入っていない場合に、デフォルト値がロードされます。欠落データの例を次に示します。 • DEL ファイルの場合: ",," が列に指定されている。 • ASC ファイルの場合: 列のヌル標識が yes に設定されている。 • DEL/ASC/WSF ファイルの場合: 列の数が足りない行がある。または行が元の指定の長さに達していない。 このオプションが指定されていない場合、ソース列に行インスタンスのデータがないと、以下のいずれかの処理が行われます。 • 列がヌル可能なら、ヌルがロードされます。 • 列がヌル可能でないなら、ユーティリティーはその行を拒否します。
ASCII ファイル形式 (ASC/DEL)	

dateformat="*x*"	*x* はソース・ファイル内の日付の形式です。 [a] 有効な日付要素は以下のとおりです。
	`YYYY - 年 (0000 ～ 9999 の 4 桁)`
	`M - 月 (1 ～ 12 の 1 桁または 2 桁)`
	`MM - 月 (1 ～ 12 の 2 桁; M と相互に排他)`
	`D - 日 (1 ～ 31 の 1 桁または 2 桁)`
	`DD - 日 (1 ～ 31 の 2 桁; D と相互に排他)`
	`DDD - 元日から数えた日数 (001 ～ 366 の 3 桁; 他の日または月要素と相互に排他)`
	指定されていない要素には、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。
	`"D-M-YYYY"`
	`"MM.DD.YYYY"`
	`"YYYYDDD"`

implieddecimal	暗黙の小数点の位置は列定義によって決められます。つまり、小数点が値の最後にあるとは想定しません。たとえば、12345 という値が DECIMAL(8,2) 列にロードされると、その値は 123.45 になります。12345.00 ではありません。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
noeofchar	オプションのファイル終わり (EOF) 文字 x'1A' があっても、ファイルの終わりとは認識しません。通常の文字の場合と同じように処理が続行されます。
timeformat="*x*"	*x* はソース・ファイル内の時刻の形式です。 [a] 有効な時刻要素は以下のとおりです。 H - 時 (12 時間系では 0 ～ 12 の 1 桁または 2 桁 24 時間系では 0 ～ 24 の 1 桁または 2 桁) HH - 時 (12 時間系では 0 ～ 12 の 2 桁。 24 時間系では 0 ～ 24 の 2 桁; H と相互に排他) M - 分 (0 ～ 59 の 1 桁または 2 桁) MM - 分 (0 ～ 59 の 2 桁; M と相互に排他) S - 秒 (0 ～ 59 の 1 桁または 2 桁) SS - 秒 (0 ～ 59 の 2 桁; S と相互に排他) SSSSS - 夜の 12 時から数えた秒数 (00000 ～ 86399 の 5 桁; 他の時刻要素と相互に排他) TT - 正午の標識 (AM または PM) 指定されていない要素には、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明

timestampformat="*x*"	*x* はソース・ファイル内のタイム・スタンプの形式です。 [a] 有効なタイム・スタンプ要素は以下のとおりです。
	YYYY - 年 (0000 ～ 9999 の 4 桁)
	M - 月 (1 ～ 12 の 1 桁または 2 桁)
	MM - 月 (1 ～ 12 の 2 桁; M と相互に排他)
	D - 日 (1 ～ 31 の 1 桁または 2 桁)
	DD - 日 (1 ～ 31 の 2 桁; D と相互に排他)
	DDD - 元日から数えた日数 (001 ～ 366 の 3 桁; 他の月日要素と相互に排他)
	H - 時 (12 時間系では 0 ～ 12 の 1 桁または 2 桁 24 時間系では 0 ～ 24 の 1 桁または 2 桁)
	HH - 時 (12 時間系では 0 ～ 12 の 2 桁。24 時間系では 0 ～ 24 の 2 桁; H と相互に排他)
	M - 分 (0 ～ 59 の 1 桁または 2 桁)
	MM - 分 (0 ～ 59 の 2 桁; M と相互に排他)
	S - 秒 (0 ～ 59 の 1 桁または 2 桁)
	SS - 秒 (0 ～ 59 の 2 桁; S と相互に排他)
	SSSSS - 夜の 12 時から数えた秒数 (00000 ～ 86399 の 5 桁; 他の時刻要素と相互に排他)
	UUUUUU - マイクロ秒 (000000 ～ 999999 の 6 桁)
	TT - 正午の標識 (AM または PM)
	
	YYYY、M、MM、D、DD、または DDD 要素に値が指定されていない場合、デフォルト値の 1 が割り当てられます。それ以外の要素に値が指定されていない場合、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	以下の例では、ユーザー定義の日付および時刻の形式を含んでいるデータを、 schedule という表にインポートする方法を示しています。
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

ASC (区切りなし ASCII) ファイル形式	

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
nochecklengths	nochecklengths を指定した場合、ソース・データにターゲット表の列のサイズを上回る列定義があっても、行ごとにインポートが試行されます。そのような行も、コード・ページ変換によってソース・データが短縮されるなら正常にインポートできます。たとえば、ソースでは 4 バイトの EUC データがターゲットでは 2 バイトの DBCS データに短縮されるなら、必要なスペースは半分になります。このオプションが特に有用なのは、列定義の不一致にもかかわらず、ソース・データがすべての場合に適合することがわかっている場合です。
nullindchar=*x*	*x* は単一の文字です。ヌル値を示す文字を *x* に変更します。*x* のデフォルト値は Y です。[b] この修飾子は EBCDIC データ・ファイルでは大文字小文字が区別されません。たとえば、ヌル標識文字を文字 N に指定した場合、 n もヌル標識として認識されます。
reclen=*x*	*x* は 32 767 以下の整数です。各行ごとに *x* 文字が読み取られ、行の終わりを示すのに改行文字は使用されません。

striptblanks	可変長フィールドにデータをロードする場合に、後書きブランク・スペースをすべて切り捨てます。このオプションを指定しない場合、ブランク・スペースは保持されます。
	
	次の例では、striptblanks によってインポート・ユーティリティーは後書きブランク・スペースを切り捨てます。
	
	`db2 import from myfile.asc of asc`
	`  modified by striptblanks`
	`  method l (1 10, 12 15) messages msgs.txt`
	`  insert into staff`
	
	このオプションは striptnulls と一緒に指定することはできません。これらは互いに排他的なオプションです。
	注: このオプションは以前の t オプションにとって代わるものです。t オプションは後方互換性だけのためにサポートされています。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
striptnulls	可変長フィールドにデータをロードする場合に、後書きヌル (文字 0x00) をすべて切り捨てます。このオプションを指定しない場合、ヌルは保持されます。 このオプションは striptblanks と一緒に指定することはできません。これらは互いに排他的なオプションです。 **注:** このオプションは以前の padwithzero オプションにとって代わるものです。padwithzero オプションは後方互換性だけのためにサポートされています。
DEL (区切り付き ASCII) ファイル形式	

chardel*x*	*x* は単一文字のストリング区切り文字です。デフォルト値は二重引用符 (") です。ここに指定する文字は、文字ストリングを囲むために二重引用符の代わりに使用されます。[bc]
	
	文字ストリング区切り文字として、単一引用符 (') を指定することもできます。次の例では、chardel'' を指定することにより、インポート・ユーティリティーは検出する単一引用符 (') すべてを文字ストリング区切り文字として解釈します。
	
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x* は単一文字の列区切り文字です。デフォルト値はコンマ (,) です。ここに指定する文字は、列の終わりを示すためにコンマの代わりに使用されます。[bc]
	
	次の例の場合、coldel; の指定によりインポート・ユーティリティーは、検出するセミコロン (;) すべてを列の区切り文字として解釈します。
	
	`db2 import from myfile.del of del`
	`modified by coldel;`
	`messages msgs.txt insert into staff`

datesiso	日付形式。すべての日付データ値を ISO 形式でインポートします。
decplusblank	正符号文字。正の 10 進数値の前に、正符号 (+) の代わりにブランク・スペースを付けます。デフォルト・アクションでは、正の 10 進数値の前に正符号が付けられます。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明

decpt*x*	*x* は、小数点文字としてピリオドの代わりに使う単一文字です。デフォルト値はピリオド (.) です。
	ここに指定する文字は、小数点文字号としてピリオドの代わりに使用されます。[bc]
	
	次の例の場合、`decpt;` の指定によりインポート・ユーティリティーは、検出するセミコロン (;) すべてを小数点として解釈します。
	
	`db2 "import from myfile.del of del`
	`modified by chardel'`
	`decpt; messages msgs.txt insert into staff"`

delpriority char	区切り文字の現在のデフォルト優先順位は、レコード区切り文字、文字区切り文字、列の区切り文字、という順序です。この修飾子は、古い優先順位に依存する既存のアプリケーションを、区切り文字の優先順位を逆転させることによって保護します。つまり、文字区切り文字、レコード区切り文字、列の区切り文字の順とします。
	構文:
	
	`db2 import ... modified by delprioritychar ...`
	
	たとえば、次のような DEL データ・ファイルがあるとします。
	
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	
	`delprioritychar` 修飾子を指定すると、このデータ・ファイルの行は 2 つだけになります。第 2 の <row delimiter> は第 2 の行の最初のデータ列の一部として解釈されますが、第 1 と第 3 の <row delimiter> は実際のレコード区切り文字として解釈されます。この修飾子を指定しない 場合、このデータ・ファイルは、それぞれが <row delimiter> で区切られる行が 3 行で構成されることになります。

dldel*x*	*x* は単一文字の DATALINK 区切り文字です。デフォルト値はセミコロン (;) です。ここに指定する文字は、DATALINK 値のフィールド間区切り文字としてセミコロンの代わりに使用されます。 DATALINK 値の副値が複数になることがあるので、これが必要になります。[bc] **注:** *x* を、行、列、または文字ストリングの区切り文字として指定された文字と同じにすることはできません。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明
keepblanks	タイプ CHAR、VARCHAR、LONG VARCHAR、または CLOB の各フィールドの前後のブランクを保持します。このオプションが指定されていない場合、文字区切り文字の外側にある前後のブランクはすべて除去され、ブランクになっているすべてのフィールドについては表にヌルが挿入されます。
nodoubledel	二重文字区切り文字の認識を抑止します。詳しくは、19ページの『区切り文字に関する制限』を参照してください。
IXF ファイル形式	
forcein	コード・ページの不一致があってもデータを受け入れて、コード・ページ間の変換を抑止するよう、ユーティリティーに指示します。 固定長のターゲット・フィールドの長さが、データを入れるのに十分かどうか検査されます。 `nochecklengths` を指定しない場合、検査なしで各行のインポートが試みられます。
indexixf	既存の表に現在定義されているすべての索引を除去し、PC/IXF ファイルの索引定義から新たに作成するよう、ユーティリティーに指示します。このオプションは、表の内容を置換する場合だけに使用できます。これは視点では使用できず、*insert-column* が指定されている場合は使用できません。
indexschema=*schema*	索引作成時に、索引名に指定された *schema* を使用します。*schema* を指定しない場合 (しかしキーワード `indexschema` は指定した場合)、接続ユーザー ID が使用されます。このキーワードを指定しない場合、IXF ファイル内のスキーマが使用されます。
nochecklengths	`nochecklengths` を指定した場合、ソース・データにターゲット表の列のサイズを上回る列定義があっても、行ごとにインポートが試行されます。そのような行も、コード・ページ変換によってソース・データが短縮されるなら正常にインポートできます。たとえば、ソースでは 4 バイトの EUC データがターゲットでは 2 バイトの DBCS データに短縮されるなら、必要なスペースは半分になります。このオプションが特に有用なのは、列定義の不一致にもかかわらず、ソース・データがすべての場合に適合することがわかっている場合です。

表 5. 有効なファイル・タイプ修飾子 (インポート) (続き)

修飾子	説明

注:
1. サポートされないファイル・タイプを MODIFIED BY オプションで使用しようとした場合、インポート・ユーティリティーは警告を出しません。それを試みた場合、インポート操作は失敗してエラー・コードが戻されます。
2. [a] 日付形式ストリングは必ず二重引用符で囲まなければなりません。フィールド区切り文字には、 a ～ z、A ～ Z、および 0 ～ 9 を含めることはできません。フィールド区切り文字は、 DEL ファイル形式の文字区切り文字またはフィールド区切り文字と同じにすることはできません。要素の開始および終了位置が明らかな場合、フィールド区切り文字は任意指定です。開始および終了位置が明らかでないのは、項目が長さが一定でない D、H、M、または S などの要素が使用されている場合です (修飾の仕方によって異なります)。
タイム・スタンプ形式の場合、月の記述子と分の記述子のどちらも文字 M を使用するため、区別が明白であるように注意しなければなりません。月のフィールドは他の日付フィールドに隣接していなければなりません。分のフィールドは他の時刻フィールドに隣接していなければなりません。以下のタイム・スタンプ形式は、あいまいな形式の例です。
`"M" (月と分のどちらかがはっきりしない)`
`"M:M" (どちらが何を表しているのか分からない)`
`"M:YYYY:M" (どちらも月として解釈される)`
`"S:M:YYYY" (時刻値と日付値の両方に隣接している)`
あいまいな形式になっている場合、ユーティリティーをエラー・メッセージを報告し、操作は失敗します。
以下のタイム・スタンプ形式では、何を指しているのかがはっきりしています。
`"M:YYYY" (月)`
`"S:M" (分)`
`"M:YYYY:S:M" (月....分)`
`"M:H:YYYY:M:D" (分....月)`
注: 二重引用符や円記号などの文字の前には、拡張文字 (たとえば、¥) を付けなければなりません。
3. [b] 文字はソース・データのコード・ページで指定する必要があります。
文字シンボルの代わりに文字コード・ポイントを指定できます。その場合、構文として xJJ または 0xJJ を使用します (JJ はコード・ポイントの 16 進表記)。たとえば、# 文字を列の区切り文字として指定するには、以下のいずれかを使用します。
`... modified by coldel# ...`
`... modified by coldel0x23 ...`
`... modified by coldelX23 ...`
4. [c] 19ページの『区切り文字に関する制限』に、区切り文字の指定変更として使用できる文字に適用される制限のリストが示されています。	

文字セットと NLS についての考慮事項

場合によって、コード・ページが異なるために文字データの長さに変化が生じることがあります。たとえば、日本語または中国語 (繁体字) の拡張 UNIX コード (EUC) と 2 バイト文字セット (DBCS) では、同じ文字が別々の長さで符号化されていることがあります。普通、入力データの長さとターゲット列の長さの比較は、まだどのデータも読み取らないうちに実行されます。入力の長さがターゲットの長さを超えている場合、列がヌル可能であればヌルがその列に挿入されます。そうでない場合、要求は拒否されます。 nochecklengths 修飾子 (60ページの『ファイル・タイプ修飾子 (インポート)』を参照) を指定した場合は、初期の比較をしないでデータをインポートしようとします。変換完了後にデータが長すぎることが明らかになったなら、その行は拒否されます。そうでなければ、データはインポートされます。

インポート・セッションの例

CLP の例

例 1

次の例は、myfile.ixf から STAFF 表へ情報をインポートする方法を示しています。

```
   db2 import from myfile.ixf of ixf messages msg.txt insert into staff
SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19970220", and time "140848".
SQL3153N  The T record in the PC/IXF file has name "myfile", qualifier "        ",
and source "            ".
SQL3109N  The utility is beginning to load data from file "myfile".
SQL3110N  The utility has completed processing.  "58" rows were read from the
input file.
SQL3221W  ...Begin COMMIT WORK. Input Record Count = "58".
SQL3222W  ...COMMIT of any database changes was successful.
SQL3149N  "58" rows were processed from the input file.  "58" rows were
successfully inserted into the table.  "0" rows were rejected.
```

例 2

次の例は、DEL 形式のデータが入っている入力ファイル delfile1 から、表 MOVIETABLE をインポートする方法を示しています。

```
db2 import from delfile1 of del
    modified by dldel|
    insert into movietable (actorname, description, url_making_of, url_movie)
    datalink specification (dl_url_default_prefix "http://narang"),
    (dl_url_replace_prefix "http://bomdel" dl_url_suffix ".mpeg")
```

注:

1. この表には 4 つの列があります。

```
actorname               VARCHAR(n)
description             VARCHAR(m)
url_making_of           DATALINK (with LINKTYPE URL)
url_movie               DATALINK (with LINKTYPE URL)
```

2. 入力ファイル内の DATALINK データでは、サブフィールド区切り文字として垂直バー (|) 文字が使われています。
3. url_making_of の列値のいずれかに接頭部文字列がない場合は ″http://narang″ が使用されます。
4. url_movie のヌルでない列値には、それぞれ接頭部として ″http://bomdel″ が付けられます。既存の値は置き換えられます。
5. url_movie のヌルでない列値では、それぞれ ″.mpeg″ がパスに付加されます。たとえば、url_movie の列値が ″http://server1/x/y/z″ であれば ″http://bomdel/x/y/z.mpeg″ として格納されることになります。また、値が ″/x/y/z″ であれば ″http://bomdel/x/y/z.mpeg″ として格納されることになります。

例 3 (識別列がある表へのインポート)

TABLE1 には、以下の 4 つの列があります。

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 は、C2 が GENERATED ALWAYS 識別列であることを除き、TABLE1 と同じです。

DATAFILE1 内のデータ・レコード (DEL 形式):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 内のデータ・レコード (DEL 形式):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

インポート・セッションの例

以下のコマンドでは、行 1 および 2 の識別値が生成されます。これは、DATAFILE1 内にこれらの行の識別値が存在しないためです。ただし、行 3 および 4 には、ユーザー提供の識別値である 100 および 101 がそれぞれ割り当てられます。

```
db2 import from datafile1.del of del replace into table1
```

DATAFILE1 を TABLE1 にインポートして、すべての行について識別値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

```
db2 import from datafile1.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore replace into table1
```

DATAFILE2 を TABLE1 にインポートして、それぞれの行について識別値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing replace into table1
```

識別に関係するファイル・タイプ修飾子を使用せずに DATAFILE1 を TABLE2 にインポートすると、行 1 と 2 は挿入されますが、行 3 と 4 は拒否されます。これは、行 3 と 4 では独自に非ヌル値が提供されており、識別列が GENERATED ALWAYS であるためです。

API の例

22ページの『API の例』を参照してください。

インポートのパフォーマンスの最適化

ターゲット表の列名を指定したり特定のインポート方式を使用したりすると、リモート・データベースへのインポートが遅くなることがあります。

制約事項と制限

インポート・ユーティリティーには、以下の制約事項が適用されます。

- このユーティリティーでは、通称の使用はサポートされません。
- 既存の表が、従属表の外部キーから参照される基本キーを含む親表である場合、そのデータは置換できず、追加だけが可能です。
- 最新表示即時モードで定義された要約表の基礎表へのインポート置換操作は実行できません。
- システム表、要約表、または構造タイプ列を持つ表にデータをインポートすることはできません。
- 宣言された一時表にデータをインポートすることはできません。

- インポート・ユーティリティーを使って視点を作成することはできません。
- PC/IXF ファイルから表を作成する場合、参照制約と外部キー定義は保存されません。 (以前に SELECT * を使ってエクスポートされたデータの場合、基本キー定義は保存されます。)
- インポート・ユーティリティーは独自の SQL ステートメントを生成するので、場合によっては最大ステートメント・サイズの 64KB を超えることがあります。

インポート・ユーティリティーには、以下の制限が適用されます。

リモート・データベースに対するインポート操作で生成された出力メッセージの量が 60KB を超える場合、このユーティリティーは最初の 30KB と最後の 30KB を保持します。

トラブルシューティング

データのエクスポート、インポート、ロード、バインド、復元などの DB2 操作においては、メッセージ・ファイルを作成するよう指定できます。メッセージ・ファイルには、それらの操作に関連したエラー、警告、および通知の各メッセージが入れられます。 MESSAGES パラメーターでそのファイルの名前を指定します。

それらのメッセージ・ファイルは標準 ASCII テキスト・ファイルです。メッセージ・ファイル内では、各メッセージはそれぞれ新たな行で始まり、 DB2 メッセージ検索機能により提供される情報がそこに入れられます。これを印刷するには、ご使用のオペレーティング・システムの印刷手順を使用します。表示するには、任意の ASCII エディターを使用してください。

第3章 ロード

この章では、DB2 UDB ロード・ユーティリティーについて説明します。このユーティリティーは、データをファイル、名前付きパイプ、または装置から DB2 表に移動するためのものです。これらのデータ・ソースは、データベースが存在するノード上か、リモート接続されたクライアント上に置くことができます。ロードする表は、存在していなければなりません。新しいデータを受け取る表にすでにデータが入っている場合は、既存のデータを置き換えたりそこに追加したりすることができます。

以下のトピックが含まれています。

- 76ページの『ロードの概要』
- 82ページの『並列性とロード』
- 83ページの『ロードの使用に必要な特権、権限、および許可』
- 84ページの『ロードの使用』
- 86ページの『ロードでの識別列の使用』
- 88ページの『ロードでの生成列の使用』
- 90ページの『制約違反の検査』
- 91ページの『中断したロード操作の再開』
- 92ページの『ロード・コピー・ロケーション・ファイルの使用』
- 94ページの『LOAD コマンド』
- 109ページの『LOAD QUERY コマンド』
- 111ページの『ロード API』
- 121ページの『データ構造体: SQLULOAD-IN』
- 126ページの『データ構造体: SQLULOAD-OUT』
- 128ページの『db2LoadQuery - ロード照会 API』
- 133ページの『ファイル・タイプ修飾子 (ロード)』
- 147ページの『例外表』
- 148ページの『ダンプ・ファイル』
- 148ページの『ロード一時ファイル』
- 149ページの『ロード・ユーティリティーのログ・レコード』
- 149ページの『文字セットと各国語のサポート』

- 150ページの『ロード・セッションの例』
- 161ページの『ロード操作後の保留状態』
- 162ページの『ロードのパフォーマンスの最適化』
- 167ページの『制約事項と制限』
- 168ページの『トラブルシューティング』

DB2 データ・リンク・マネージャーのデータのロードについては、194ページの『DB2 データ・リンク・マネージャーのデータの移動 - ロードの使用』を参照してください。

ロードの概要

ロード・ユーティリティーを使用すれば、新しく作成した表やすでにデータが入っている表に、大量のデータを効率よく移動することができます。このユーティリティーは、ラージ・オブジェクト (LOB) やユーザー定義タイプ (UDT) を含むすべてのデータ・タイプを処理できます。インポート・ユーティリティーは SQL INSERT を実行するのに対し、ロード・ユーティリティーは形式設定したページをデータベースに直接書き込むため、インポート・ユーティリティーよりも処理が高速です。ロード・ユーティリティーはトリガーを起動したり、参照制約や表制約の検査を実行したりしません (索引の一意性の妥当性検査を除く)。ロード・ユーティリティーとインポート・ユーティリティーの違いの詳細については、221ページの『付録B. インポート・ユーティリティーとロード・ユーティリティーの相違点』を参照してください。

ロードのプロセスは、以下に示す 3 つの明確なフェーズ (段階) で構成されています (77ページの図1 を参照)。

- ロード。このフェーズ中にデータが表に書き込まれます。

  ロード・フェーズ中に、データは表にロードされ、必要に応じて索引キーと表統計が集められます。保管ポイントまたは一貫性ポイントは、LOAD コマンドの SAVECOUNT パラメーターによって指定される間隔で確立されます。保管ポイントの時点で正常にロードされた入力行の数を示すメッセージが生成されます。 FILE LINK CONTROL を指定して定義された DATALINK 列の場合、ヌルでない列値に対してリンク操作が実行されます。障害が発生した場合はロード操作をもう一度開始できます。 RESTART オプションを指定した場合、ロード操作の再開位置は最後に成功した一貫性ポイントに自動的に設定されます。 TERMINATE オプションが指定されているなら、失敗したロード操作がロールバックされます。

ロード操作の 3 つのフェーズ

図 1. ロード・プロセスの 3 つのフェーズ: ロード、構築、削除。. 関連する表スペースは、ロード・フェーズの始点から構築フェーズの終点まではロード保留状態、また構築フェーズの終点から削除フェーズの終点までは削除保留状態になります。

- 構築。このフェーズ中に索引が作成されます。

  構築フェーズ中には、ロード・フェーズ中に収集した索引キーに基づいて索引が作成されます。索引キーはロード・フェーズ中にソートされ、INDEXES オプションに STATISTICS YES を指定した場合は索引統計データが集められます。この統計データは、RUNSTATS コマンドによって収集される統計とよく似たものです (*コマンド解説書* を参照)。構築フェーズ中に障害が発生した場合、RESTART オプションが指定されているなら、ロード操作が適切なポイントから自動的に再開されます。
- 削除。このフェーズ中に、固有キー違反や DATALINK 違反の発生した行が表から除かれます。固有キー違反は、例外表が指定されていればそこに入れられます (147ページの『例外表』を参照)。また、拒否された行に関するメッセージはメッセージ・ファイルに書き込まれます。ロード・プロセスが完了したなら、これらのメッセージを見て、何か問題があればそれを解決し、訂正済みの行を該当する表に挿入してください。

  ロード・ユーティリティーが作成した一時ファイルは、決して削除したり変更したりしないでください。一時ファイルの中には、削除フェーズにおいて非常に重要なものがあります。削除フェーズ中に障害が発生した場合、RESTART オプションが指定されているなら、ロード操作が適切なポイントから自動的に再開されます。

  注: 削除イベントは、イベントごとにログに記録されます。固有条件に違反しているレコードがたくさんある場合は、削除フェーズ中にログがいっぱいになってしまう可能性があります。

データをロードするには、以下の情報が必要になります。

- 入力となるファイル、名前付きパイプ、または装置のパスと名前。
- ターゲット表の名前または別名。
- 入力ファイル内のデータの形式。形式は DEL、ASC、または PC/IXF です。225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

- 入力データを表に追加するか、それとも表内の既存データを置換するか。
- アプリケーション・プログラミング・インターフェース (API) **sqluload** によってユーティリティーが起動される場合、メッセージ・ファイル名。

さらに、以下の情報を指定することもできます。

- ロードするデータがクライアントに置かれていること (リモート接続されたクライアントからロード・ユーティリティーを起動する場合)。
- データのロードに使用する方式: 列のロケーション、列名、または相対列位置。
- ユーティリティーが一貫性ポイントを確立する頻度。その値は SAVECOUNT パラメーターを使って指定します。このパラメーターを指定すると、ロードの再開操作は最初からではなく、最後の一貫性ポイントから開始されます。
- データの挿入先となる表列の名前。
- LOB が格納されている入力ファイルのパスと名前。 `lobsinfile` 修飾子を使うと、すべての LOB データをファイルからロードするようロード・ユーティリティーに指示することができます (133ページの『ファイル・タイプ修飾子 (ロード)』を参照)。
- ロードしている列値に暗黙の小数点があるかどうか。 `implieddecimal` 修飾子は、データを表に入力する時点で小数点を適用するようロード・ユーティリティーに指示するのに使います (133ページの『ファイル・タイプ修飾子 (ロード)』を参照)。たとえば、値 12345 は、DECIMAL(8,2) 列に 12345.00 ではなく 123.45 としてロードされます。
- 表をロードした後、使用可能な空きスペースの大きさをユーティリティーが変更するかどうか。空きスペースが大きくなると、ロード操作の完了後に INSERT および UPDATE が使える領域もそれだけ大きくなります。空きスペースが小さくなると、関連している行がさらに互いに近接することになるため、表のパフォーマンスが上がることがあります。
  - `totalfreespace` 修飾子を使うと、ロードした表の末尾に空のデータ・ページが追加されます。このパラメーターで指定する数は、表の末尾に空きスペースとして追加されるページ数の、表の合計ページ数に対する割合 (パーセント) です。たとえば、このパラメーターに 20 を指定した場合、表にデータ・ページが 100 ページあるとすると、空のページが 20 ページ分追加されます。それで、表のデータ・ページの合計数は 120 になります。
  - `pagefreespace` 修飾子は、ロードする各データ・ページに含めることのできる空きスペースの大きさを制御するのに使います。このパラメーターに指定する数は、各データ・ページごとに空きスペースとして残しておく分

のパーセントです。ページの最初の行は制限なしで追加されます。そのため、非常に大きな行があり、このパラメーターに大きな数を指定した場合、各ページに残される空きスペースはこのパラメーターに指定した値より小さくなる場合があります。

– `indexfreespace` 修飾子は、ロードする各索引ページに含めることのできる空きスペースの大きさを制御するのに使います。このパラメーターに指定する数は、各索引ページごとに空きスペースとして残しておく分のパーセントです。ページの最初の索引は制限なしで追加されます。追加の索引項目は索引ページに入れられるため、空きスペースの限界値として指定したパーセントはそのまま保たれます。 CREATE INDEX の実行時に使われる値がデフォルト値になります。 `indexfreespace` 値は、CREATE INDEX ステートメントに指定された PCTFREE 値より優先されます。

`pagefreespace` 修飾子を指定する場合、表に索引があるなら `indexfreespace` を指定することを検討してください。それぞれに残す空きスペースの大きさを決める際には、表に挿入される各行のサイズが索引に挿入される関連キーのサイズより大きくなることが多いという点に注意してください。さらに、表スペースのページ・サイズが表と索引とで違っている可能性があります。

- ロード・プロセス中に統計情報を収集するかどうか。このオプションは、ロード操作を REPLACE モードで実行する場合にのみサポートされます。

  データを表に追加する場合、統計情報は収集されません。追加がなされた表に関する現行の統計情報を収集するには、ロード・プロセスの完了後に runstats ユーティリティーを呼び出します。固有索引のある表に関する統計情報を収集していて、削除フェーズ中に重複キーが削除される場合、それらの削除されたレコードについては統計情報の更新がなされません。重複レコードの有効な数を入手したい場合は、ロード操作中には統計情報を収集しないでください。その代わりに、ロード・プロセスの完了後に runstats ユーティリティーを呼び出すようにしてください。

- 変更部分のコピーを保持するかどうか。これにより、データベースのロールフォワード回復が可能になります。このオプションは、データベースの順方向ログ回復が無効になっている場合、つまりデータベース構成パラメーター *logretain* および *userexit* が無効になっている場合にはサポートされません。コピーが作成されていないのに順方向ログ回復が有効になっていると、ロード操作の完了時に表スペースがバックアップ保留状態のままになります (161 ページの『ロード操作後の保留状態』を参照)。

  データベースを完全に回復できるようにするには、ログを記録することが必要です。ロード・ユーティリティーを利用した場合、データのロードに関連したログ記録はほとんど不要です。ログを記録する代わりに、表のうちのロードした部分のコピーを作成することもできます。 DB2 がロードのコピー

を追跡する方法については、92ページの『ロード・コピー・ロケーション・ファイルの使用』を参照してください。障害発生後にデータベースを回復できるようになっているデータベース環境では、以下のいずれかを実行できます。

– 表のうちのロードした部分のコピーを作成することを明示的に要求する。
– 表の属する表スペースのバックアップを、ロード操作の完了直後に取る。

すでにデータが入っている表をロードしており、データベースが回復可能でない場合は、ロード・ユーティリティーを呼び出す前にそのデータベースあるいはロードする表の表スペースのバックアップ・コピーがあることを確認し、エラーがあっても回復できるようにしておいてください。

回復可能なデータベースで複数のロード操作を連続して実行したい場合は、それぞれのロード操作を回復不能に指定しておき、一連のロードの最後にバックアップを取ることにより、各ロード操作を COPY YES オプション付きで呼び出す場合よりも短い時間で一連の操作を実行できます。NONRECOVERABLE オプションを使えば、あるロード・トランザクションを回復不能としてマークし、それ以降にロールフォワードを実行しても回復できなくするよう指定できます。ロールフォワード・ユーティリティーはそのトランザクションをスキップし、データのロード先の表を "invalid" (無効) としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワードの完了後、このような表は除去することしかできません (図2 を参照)。このオプションを指定すると、表スペースはロード操作後にバックアップ保留状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。

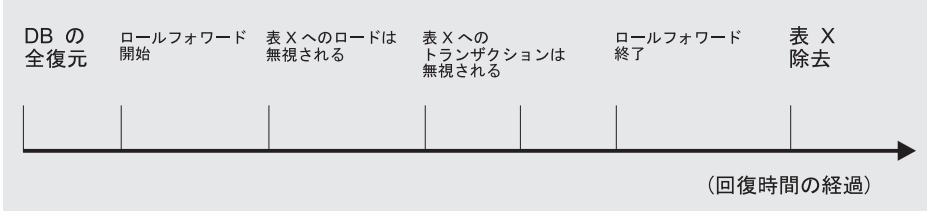


図 2. ロールフォワード・アクション中の回復不能処理

詳しくは、*管理の手引き* にあるデータベースの回復に関する章を参照してください。

- ロード操作中の一時ファイル作成時に使用する完全修飾パス。名前は、LOAD コマンドの TEMPFILES PATH パラメーターで指定します。デフォルト値はデータベースのパスです。このパスはサーバー・マシン上にあり、

DB2 のインスタンスが排他的にアクセスします。そのため、このパラメーターに指定するパス名の修飾は、クライアントではなくサーバーのディレクトリー構造を反映していなければならず、 DB2 インスタンス所有者にはそのパスに対して読み取りと書き込みの両方の許可が必要です。これはインスタンス所有者であっても適用されます。インスタンス所有者でない場合は、インスタンス所有者にとって書き込み可能な場所を指定する必要があります。一時ファイルについては、148ページの『ロード一時ファイル』を参照してください。

バージョン 6 とバージョン 7 で導入された以前のロード動作の変更

バージョン 6 とバージョン 7 のロード・ユーティリティーには、旧リリースとの完全なバック・レベル互換性があります。つまり、旧リリースの構文を受け入れ、正常に稼働します。以下に、バージョン6 で導入された構文の変更とロード動作の変更の要約を示します。

- ロードの再始動時に RESTARTCOUNT 値は使われなくなりました。現在、このパラメーターは予約されています。以前に中断したロード操作を再開すると、ロード操作はロード、構築、または削除フェーズのうちの最後の一貫性ポイントから自動的に続行します。
- 索引作成時の索引キーのソートでは、 DB2 UDB バージョン 6 データベース・エンジンで使われる新規のソート・アルゴリズムが活用されます。ソート専用のメモリー容量は、ソート・ヒープ (*sortheap*) データベース構成パラメーターの値と、ソート・ヒープしきい値 (*sheapthres*) データベース・マネージャー構成パラメーターの値によって制御されます。つまり、以前の LOAD オプションである SORT BUFFER と、 USING 文節で指定されていた一時ソート・ディレクトリーは、現在は使われていません。バージョン 6 でこれらのオプションが指定されていると、通知用の警告メッセージが戻されますが、ロード操作は正常に進行します。

  ロード索引の作成時に発生するソート・スピルは、一時表スペース内で実行されるようになりました。ソート操作は、ディスクに直接スピルされるのではなく、一時表スペースに関連したバッファー・プールにスピルされます。一時表スペースに関係付けるバッファー・プールを大きくすれば、索引作成の時間を改善することができます。バージョン 6 より前のロード・ソート操作で (複数の一時ソート・ディレクトリーを指定して) 実現できたものと同じタイプの入出力の並列性を実現するには、一時表スペースの宣言時に、それぞれが異なるディスク装置に存在する複数のコンテナーを指定することをお勧めします。また、一時表スペースを SMS (システム管理スペース) として宣言することもお勧めします。これによって、大量のデータの収容時にこのスペースが拡大し、ディスク・リソースを使っていないときにそのリソースを保持することがなくなります。

- REMOTE FILE オプションの名前が変更されました (ただし、一時ファイルのパスの指定時には、ユーティリティーは依然として REMOTE FILE パラメーターを受け入れます)。これは、このパラメーターの意味と目的をより良く反映するための構文上の変更にすぎません。 TEMPFILES PATH パラメーターは、ファイルではなくディレクトリーを参照します (148ページの『ロード一時ファイル』を参照してください)。
- 現在、ロード・ユーティリティーは、複数の索引作成モードをサポートします。それらは、完全な REBUILD (再構築)、INCREMENTAL (増分) 拡張、ロード操作が完了するまでの索引保守の DEFERRED (据え置き)、AUTOSELECT (自動選択) モード (実行時に全再構築と増分保守のどちらかを選択する) です。全再構築モードでは、バージョン 6 より前のリリースでの動作が反映されます。バージョン 6 でのデフォルト動作は AUTOSELECT モードです。
- バージョン 6 では、TERMINATE オプションを使って、ロード操作をロールバックすることができます。これまでこのオプションは、表スペースを復元保留状態にしていました。ただし、LOAD REPLACE 操作が失敗した後で TERMINATE 要求を出しても、表データは復元されない ことに注意してください。

バージョン 7 のロード・ユーティリティーは、リモート接続されたクライアントに置かれているデータを、完全修飾ファイルまたは名前付きパイプ内でロードすることができます。 (`lobsinfile` ファイル・タイプ修飾子の指定時には、LOB 値の入った別のファイルがサーバーになければなりません。)

並列性とロード

ロード・ユーティリティーは、複数のプロセッサーや複数の記憶装置が使われているハードウェア構成 (対称マルチプロセッサー (SMP) 環境など) を利用します。ロード・ユーティリティーを使って大容量データの並列処理を実行する方法はいくつかあります。その 1 つは複数の記憶装置を使う方法であり、ロード操作中に入出力の並列処理が可能になります (83ページの図3 を参照)。別の方法は SMP 環境における複数のプロセッサーの使用が関係しており、区画内の並列処理が可能になります (83ページの図4 を参照)。これらの両方の方法を併用すれば、データのロード時間をさらに短くすることができます。詳しくは、162ページの『ロードのパフォーマンスの最適化』を参照してください。

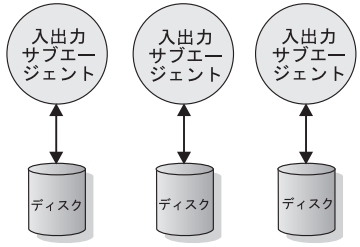


図 3. データ・ロード時に入出力の並列処理を利用する

図 4. データ・ロード時に区画内の並列処理を利用する

ロードの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

データを表にロードするには、以下のいずれかが必要です。

- SYSADM 権限
- DBADM 権限
- データベースに対する LOAD 権限、および以下に示す特権
  - 表の INSERT 特権 (ロード・ユーティリティーが INSERT モード、TERMINATE モード、または RESTART モードで呼び出される場合)。TERMINATE モードは直前のロード挿入操作を終了するためのもので、RESTART モードは直前のロード挿入操作を再開するためのものです。

- 表の INSERT および DELETE 特権 (ロード・ユーティリティーが REPLACE モード、 TERMINATE モード、または RESTART モードで呼び出される場合)。 TERMINATE モードは直前のロード置換操作を終了するためのもので、 RESTART モードは直前のロード置換操作を再開するためのものです。
- 例外表の INSERT 特権 (例外表をロード操作の一部として使用する場合)。

すべてのロード・プロセス (および一般にすべての DB2 サーバー・プロセス) はインスタンス所有者が所有しており、これらのプロセスはすべてインスタンスの ID を使って必要なファイルにアクセスするため、インスタンス所有者にはデータ・ファイルを入力するために読み取りアクセスが必要です。だれがコマンドを呼び出すかに関係なく、これらの入力データ・ファイルはインスタンス所有者から読み取り可能になっていなければなりません。

DB2 (Windows NT 版) が Windows NT オペレーティング・システムに対するサービスとして定義されている場合、そのサービスのユーザー・アカウントには、 LAN 資源 (ドライブ、ディレクトリー、およびファイル) を使用するのに必要な読み取り / 書き込みファイル許可がなければなりません。

ロードの使用

ロードを使用する前に

ロード・ユーティリティーを起動するには、その前にデータのロード先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。このユーティリティーは COMMIT ステートメントを発行するため、ロードの起動前に COMMIT または ROLLBACK を実行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておいてください。

データは入力ファイルに示されている順序でロードされるため、特定の順序でロードしたい場合には、ロード操作の開始前にデータをソートしておいてください。

クラスター化する必要がある場合は、ロードする前にクラスター化索引に従ってデータをソートしておいてください。

ロードの起動

ロード・ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  CLP によって発行する LOAD コマンドの例を以下に示します。

  ```
  db2 load from stafftab.ixf of ixf messages staff.msgs
     insert into userid.staff copy yes use tsm data buffer 4000
  ```

  この例で、

  – 警告メッセージやエラー・メッセージはすべて staff.msgs ファイルに入れられます。

  – 変更された部分のコピーは、 Tivoli Storage Manager (TSM、旧称 ADSM) に保管されます。 TSM については、 *管理の手引き* の中のデータベース回復に関する章の『Tivoli Storage Manager』の項を参照してください。

  – 4,000 ページ分のバッファー・スペースがロード操作中に使われます。

  CLP によって発行する LOAD コマンドの別の例を以下に示します。

  ```
  db2 load from stafftab.ixf of ixf messages staff.msgs
     tempfiles path /u/myuser replace into staff
  ```

  この例で、

  – 表データは置換されます。

  – TEMPFILES PATH パラメーターを使って、一時ファイルを書き込むサーバー・パスとして /u/myuser を指定しています。

  注: これらの例では、ロード用入力ファイルに相対パス名を使っています。相対パス名を使用できるのは、データベースと同じノード上にあるクライアントから呼び出す場合だけです。できれば完全修飾パス名を使用してください。

- コントロール・センターの「ロード (Load)」ノートブック。「ロード (Load)」ノートブックをオープンするには、次のようにします。

  1. コントロール・センターから、「表 (Tables)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
  2. 「表 (Tables)」フォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存の表がすべて表示されます。
  3. 目次ペイン内で対象となる表をマウスの右ボタンでクリックし、ポップアップ・メニューから「ロード (Load)」を選択します。「ロード (Load)」ノートブックがオープンします。

コントロール・センターについての一般情報については、*管理の手引き* を参照してください。詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

- アプリケーション・プログラミング・インターフェース (API) としての **sqluload**。この API については、111ページの『ロード API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、*アプリケーション構築の手引き* を参照してください。

ロードでの識別列の使用

ロード・ユーティリティーを使って、識別列の入った表にデータをロードすることができます。識別関連のファイル・タイプ修飾子が使用されない場合、このユーティリティーは次のような規則に従って動作します。

- 識別列が GENERATED ALWAYS の場合、入力ファイル内の対応する行の中に識別列用の値がないか、またはヌル値が明示的に指定されているときに、表行用の識別値が生成されます。識別列に非ヌル値を指定すると、その行は拒否されます (SQL3550W)。
- 識別列が GENERATED BY DEFAULT の場合、ユーザー提供値が指定されていれば、ロード・ユーティリティーはその値を使います。データが欠落しているかまたは明示的にヌルであれば、値が生成されます。

ロード・ユーティリティーは、ユーザー提供の識別値に関して、識別列のデータ・タイプ (SMALLINT、INT、BIGINT、または DECIMAL) の値に対して通常行う以外の余分な妥当性検査を行いません。値が重複していても報告されません。

識別列の入った表の使用を単純化するために、次のような 3 つの (相互に排他的な) ファイル・タイプ修飾子がロード・ユーティリティーでサポートされています。

- `identitymissing` 修飾子は、入力データ・ファイルに識別列の値が入っていない (ヌルすらない) 場合に、識別列を持つ表のロードを容易にします。たとえば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 varchar(30),
                     c2 int generated by default as identity,
                     c3 decimal(7,2),
                     c4 char(1))
```

ユーザーが、識別列をもたない表からエクスポートされたファイル (`load.del`) からデータを TABLE1 にロードするとします。以下に、このようなファイルの例を示します。

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

このファイルをロードする 1 つの方法は、次のように LOAD コマンドを使って、ロードする列を明示的にリストすることです。

```
db2 load from load.del of del replace into table1 (c1, c3, c4)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをロードする別の方法は、次のように identitymissing ファイル・タイプ修飾子を使うことです。

```
db2 load from load.del of del modified by identitymissing replace into table1
```

- identityignore 修飾子は、ある意味では identitymissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、ロード・ユーティリティーは、入力データ・ファイルに識別列用の値が入っていても、そのデータを無視して、各行ごとに識別値を生成します。たとえば、上記で定義したように、ユーザーが次のようなデータの入ったデータ・ファイル (load.del) から TABLE1 をロードするとします。

  ```
  Robert, 1, 45.2, J
  Mike, 2, 76.9, K
  Leo, 3, 23.4, I
  ```

  ユーザー指定値の 1、2、および 3 が識別列に使われない場合、ユーザーは次のような LOAD コマンドを使うことができます。

  ```
  db2 load from load.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
  ```

  ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように identityignore 修飾子を使うと、構文が単純化されます。

  ```
  db2 load from load.del of del modified by identityignore replace into table1
  ```

- identityoverride 修飾子は、 GENERATED ALWAYS 識別列をもつ表にユーザー指定値をロードするのに使います。これが非常に役立つのは、別のデータベース・システムからデータを移行するときに GENERATED ALWAYS として表を定義しなければならない場合や、 ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用して回復したデータから表をロードする場合です。この修飾子を使用した場合、識別列でデータ (またはヌル・データ) の入っていない行は拒否されます (SQL3116W)。

  注: この修飾子を使用すると、 GENERATED ALWAYS 列の固有性特性に違反する可能性があります。

ロードでの生成列の使用

ロード・ユーティリティーを使って、 (識別列でない) 生成列の入った表にデータをロードすることができます。ただし現在、ユーティリティー内での非識別列値の生成はサポートされていないので、そのような列をもつ表にロードすると、常に表は検査保留状態に置かれたままになります。表を検査保留状態から解放して強制的に値を生成するには、以下のコマンドを発行します。

```
SET INTEGRITY FOR tablename IMMEDIATE CHECKED FORCE GENERATED;
```

ロードされた値を受け入れるには、以下のコマンドを発行します。

```
SET INTEGRITY FOR tablename GENERATED COLUMN IMMEDIATE UNCHECKED;
```

生成列関連のファイル・タイプ修飾子が使用されない場合、ロード・ユーティリティーは次のような規則に従って動作します。

- ヌル可能な生成列の場合、データ・ファイル内の列用にヌル値を提供すると、その値は受け入れられて表にロードされます。ヌル可能でない生成列の場合、データ・ファイル内の列用にヌル値を提供すると、生成列のデータ・タイプのデフォルト値がロードされます。ヌル可能およびヌル不能のどちらの生成列でも、データ・ファイル内に非ヌル値を入れると、その行は拒否されます (SQL3550W)。
- ロード操作の完了後、 FORCE GENERATED オプションを指定した SET INTEGRITY ステートメントを使って表の検査保留状態を解除すると、ロードした値は生成値に置き換えられます。 FORCE GENERATED オプションを使用しないと、次の場合にロード値はそのままになります。
  - 生成値の制約事項のいずれにも違反していない
  - GENERATED COLUMN IMMEDIATE UNCHECKED オプションを使用している

  GENERATED COLUMN IMMEDIATE UNCHECKED オプションを使用していない 場合、列の制約事項に違反している行は、例外表に入れられます (例外表がある場合)。

生成列の入った表の使用を単純化するために、次のような 3 つのファイル・タイプ修飾子がロード・ユーティリティーでサポートされています。

- generatedmissing 修飾子は、表内に存在する生成列の値が入力データ・ファイル内に入っていない (ヌルすらない) 場合に、生成列を持つ表のロードを容易にします。たとえば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 int,
                     c2 int,
                     g1 int generated always as (c1 + c2),
                     g2 int generated always as (2 * c1),
                     c3 char(1))
```

ユーザーが、生成列をもたない表からエクスポートされたファイル (load.del) からデータを TABLE1 にロードするとします。以下に、このようなファイルの例を示します。

```
1, 5, J
2, 6, K
3, 7, I
```

このファイルをロードする 1 つの方法は、次のように LOAD コマンドを使って、ロードする列を明示的にリストすることです。

```
db2 load from load.del of del replace into table1 (c1, c2, c3)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをロードする別の方法は、次のように generatedmissing ファイル・タイプ修飾子を使うことです。

```
db2 load from load.del of del modified by generatedmissing replace into table1
```

- generatedignore 修飾子は、ある意味では generatedmissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、ロード・ユーティリティーは、すべての生成列用のデータが入力データ・ファイルに入っていても、そのデータを無視して、各列にヌルをロードします。たとえば、上記で定義したように、ユーザーが次のようなデータの入ったデータ・ファイル (load.del) から TABLE1 をロードするとします。

  ```
  1, 5, 10, 15, J
  2, 6, 11, 16, K
  3, 7, 12, 17, I
  ```

  ユーザー提供の非ヌル値 10、11、12 (g1 の場合)、および 15、16、17 (g2 の場合) により、行は拒否されます (SQL3550W)。拒否されないようにするには、次のような LOAD コマンドを発行します。

  ```
  db2 load from load.del of del method P(1, 2, 5) replace into table1 (c1, c2, c3)
  ```

  ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように generatedignore 修飾子を使うと、構文が単純化されます。

  ```
  db2 load from load.del of del modified by generatedignore replace into table1
  ```

- generatedoverride 修飾子は、生成列をもつ表にユーザー指定値をロードするのに使います。これが非常に役立つのは、別のデータベース・システムか

らデータを移行する場合や、 ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用して回復したデータから表をロードする場合です。この修飾子を使用した場合、ヌル不能の生成列でデータ (またはヌル・データ) の入っていない行は拒否されます (SQL3116W)。

制約違反の検査

ロードした表に表検査制約または参照保全が定義されている場合は、ロード操作の後にその表が検査保留状態になっていることがあります。ロードした表の検査保留状態は、その表に対応する SYSCAT.TABLES 項目の STATUS フラグに示されます。ロードした表を使用可能にするには、STATUS に値として N (通常の状態を示す) を指定しなければなりません。

検査保留状態を除去するには、SET INTEGRITY ステートメントを使用します (*SQL 解説書* を参照)。 SET INTEGRITY ステートメントは表を検査して制約違反がないかどうかを調べ、その表の検査保留状態を終了します。すべてのロード操作が INSERT モードで実行される場合、SET INTEGRITY ステートメントはデフォルトでは制約を増分的に処理します (つまり表のうち追加された部分だけを検査して、制約違反がないかどうかを調べます)。たとえば、

```
db2 load from infile1.ixf of ixf insert into table1
db2 set integrity for table1 immediate checked
```

制約違反がないかどうかを検査するのは TABLE1 のうち追加部分だけです。追加部分だけを検査して制約違反がないかどうかを調べることにより、表全体を検査するよりも時間が短くてすみます。これは、大きな表にデータを少しだけ追加した場合に特に有効です。

表が 1 つでも複数でも、このステートメントを 1 回呼び出すだけで検査できます。従属表を検査する場合、その親表が検査保留状態になっていてはなりません。参照保全が循環している場合、その循環に関係しているすべての表を SET INTEGRITY ステートメントのその 1 回の呼び出しに含める必要があります。従属表をロードしている間に、親表に制約違反がないかどうかを検査するのがよいかもしれません。これが当てはまるのは、2 つの表が同じ表スペースにない場合だけです。

SET INTEGRITY ステートメントの発行時に INCREMENTAL オプションを指定すると、増分的な処理を明示的に要求することができます。 しかし、増分的な処理はデフォルトの動作であるため、このオプションはほとんどの場合不要です。 増分的な処理を実行できない場合には、自動的に全処理が実行されま

す。 INCREMENTAL オプションを指定したにもかかわらず増分的な処理を実行できない場合、以下に示す条件が成立するならエラーが戻されます。

- 新しい制約を表またはその親表に追加したが、どちらの表も検査保留状態である場合。
- 表に対する最後の保全性検査の後で、ロード置換操作が行われたか、または NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化された場合。
- 親表が、ロード置換されたか、または増分的ではない方法で保全性検査された場合。
- 表が、移行前の検査保留状態にあった場合。移行後に最初に表が保全性検査されるときは、完全処理が必要です。
- 表またはその親表の入った表スペースが、ある時点にロールフォワードされた場合。

T 表において SYSCAT.TABLES カタログの CONST_CHECKED 列に 1 つまたは複数の W 値がある場合 (W の状態については、 *SQL 解説書* の SET INTEGRITY ステートメントの項目を参照)、INCREMENTAL オプションが指定されていなければシステムはその表全体について制約違反がないかどうかを調べます。このオプションが*指定*されている 場合にもこの処理は可能ですが、SYSTABLES の CONST_CHECKED 列には、すべてのデータをシステムが検査したわけではないことを示す U のマークが付けられます。

制約違反のある行についての情報を取得するには、ロード例外表オプションを使用します (147ページの『例外表』を参照)。

SET INTEGRITY ステートメントは、制約違反のある行を削除した結果として DELETE トリガーを起動することはありませんが、表が検査保留状態でなくなるとトリガーが起動されます。そのため、例外表のデータを収集して、ロードした表に例外表の行を挿入すると、その表に定義されている INSERT トリガーが起動されます。これが何を意味するかを考えてください。 1 つの選択肢は、INSERT トリガーをいったん除去してから例外表から行を挿入し、その後で INSERT トリガーを再作成することです。

中断したロード操作の再開

実在しないデータ・ファイルや無効な列名などのユーザー・エラーが原因でロード・ユーティリティーを開始できない場合、ロード・ユーティリティーは表スペースを通常の状態にしたまま終了してしまいます。

中断したロード操作の再開

データのロード中に障害が発生した場合は、最後の一貫性ポイントからロード操作を再開 (RESTART オプションを使用) するか、表全体を再ロード (REPLACE オプションを使用) することができます。前の呼び出しと同じパラメーターを指定することにより、必要な一時ファイルをユーティリティーが見つけられるようにしてください。

ロード・コピー・ロケーション・ファイルの使用

DB2LOADREC レジストリー変数は、ロード・コピーのロケーション情報の入っているファイルを示すのに使用します。このファイルは、ロールフォワード回復中にロード・コピーの位置情報として使われます。その中には次の情報が入れられます。

- メディアの種類
- 使用するメディア装置の数
- 表のロード操作中に生成されるロード・コピーの位置
- ロード・コピーのファイル名 (もしあれば)

ロケーション・ファイルが存在しない場合、あるいはファイル内に一致する項目がない場合は、ログ・レコードからの情報が使われます。

ファイル内の情報は、ロールフォワード回復の実行前に上書きされることがあります。

区分データベース環境では、ロード・コピー位置ファイルがそれぞれのデータベース区画サーバーごとに存在していなければならず、ファイル名 (パスを含む) は同一でなければなりません。

ロケーション・ファイルの例を下記に示します。最初の 5 つのパラメーターには有効な値を指定する必要があり、これらのパラメーターはロード・コピーを識別するために使われます。全体の構造は記録されるロード・コピーごとに繰り返されます。

```
TIMestamp      19950725182542         * Time stamp generated at load time
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
DB2instance    TORONTO                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for recovery
SESsionnumber  NULL                   * Number of sessions to be used for recovery
TYPeofmedia    L                      * Type of media - L for local device
                                                       A for Tivoli Storage Manager (TSM, formerly ADSM)
                                                       O for other vendors
LOCationnumber 3                      * Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
```

```
DB2             TORONTO
SES             NULL
BUF             NULL
TYP             A
TIM             19940325192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             TORONTO
SES             NULL
BUF             NULL
TYP             O
SHRlib          /@sys/lib/backup_vendor.a
```

注:

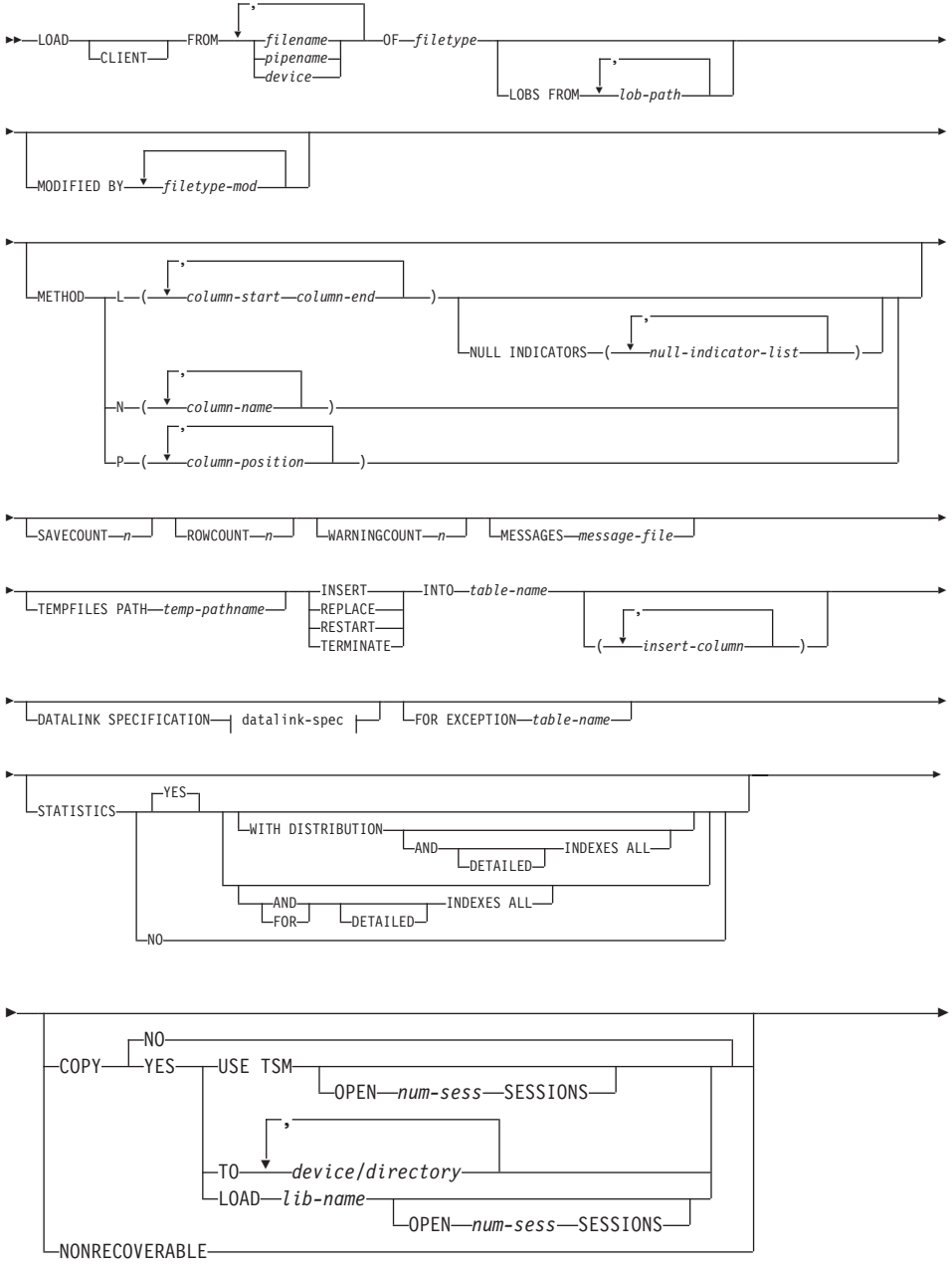
1. 重要なのは各キーワードの最初の 3 文字です。すべてのキーワードは、指定された順序になっている必要があります。ブランク行は使用できません。
2. タイム・スタンプの形式は *yyyymmddhhmmss* です。
3. フィールドはすべて必須ですが、BUF と SES だけはヌルにすることができます。
4. メディアの種類は、ローカル装置 (L - テープ、ディスクまたはディスケットの場合)、 Tivoli Storage Manager (A - 旧称 ADSM)、あるいは他のベンダー (O) のいずれかを指定できます。これが L の場合、ロケーション項目に続けてロケーションの数を指定する必要があります。種類が A の場合、それ以上入力する必要はありません。種類が O の場合は、共用ライブラリー名を指定する必要があります。 Tivoli Storage Manager (TSM) や他のベンダー製品をバックアップ・メディアとして使用する方法については、 *管理の手引き* の中のデータベース回復に関する章の『Tivoli Storage Manager』の項を参照してください。
5. SHRlib パラメーターは、LOAD COPY データを格納する機能を持つライブラリーを指します。

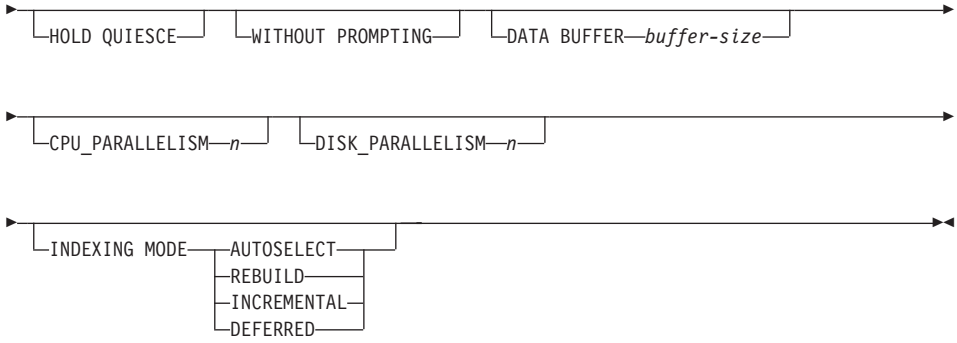
LOAD COPY NO を実行した場合、ロード操作実行後にデータベースや影響を受ける表スペースのバックアップ・コピーを取っていないなら、ロード操作より後の時点の状態にデータベースや表スペースを復元することはできません。つまり、ロールフォワード回復を使ってもデータベースや表スペースをロード操作後の状態に再構築することはできません。データベースや表スペースを復元できるのは、ロード操作より*前* の時点の状態だけです。

特定のロード・コピーを使用したい場合には、データベースの回復履歴ファイルにロードのタイム・スタンプが記録されています。区分データベース環境の場合、回復履歴ファイルは各データベース区画にローカルに存在します。

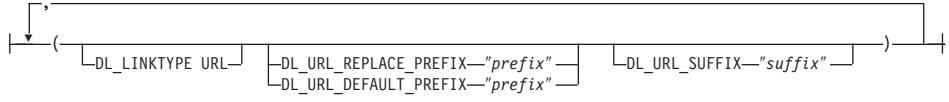
LOAD コマンド

コマンド構文





datalink-spec:



コマンド・パラメーター

CLIENT

ロードするデータが、リモートで接続されているクライアント上にあることを指定します。ロード操作がリモート・クライアントから呼び出されていない場合、このオプションは無視されます。

注:

1. リモートで接続しているクライアント上にあるデータのロードは、以下の条件ではサポートされていません。
   - クライアントが接続しているデータベースが DB2 エンタープライズ拡張エディション環境である。
   - クライアントが接続しているデータベースが、すでにカタログ化されているデータベースに対してカタログ化されている。
2. DUMPFILE および LOBSINFILE 修飾子 (133ページの表8 を参照) は、 CLIENT キーワードが指定されている場合でも、サーバー上のファイルを参照します。
3. コード・ページの変換は、リモート・ロード操作時には実行されません。データのコード・ページがサーバーのものと異なる場合は、CODEPAGE 修飾子 (133ページの表8 を参照) を使用して、そのデータ・コード・ページを指定しなければなりません。

4. ファイル (名前付きパイプではない) からリモート・クライアント・データをロードする場合、上限は 1 ファイルにつき 2GB です。

以下の例では、リモートで接続されているクライアントにあるデータ・ファイル (/u/user/data.del) が、サーバー・データベース上の MYTABLE にロードされます。

```
db2 load client from /u/user/data.del of del
   modified by codepage=850 insert into mytable
```

COPY NO

順方向回復が有効になっている場合 (つまり *logretain* または *userexit* がオンの場合) に、表の属する表スペースがバックアップ保留状態になるよう指定します。表スペース・バックアップまたは全データベース・バックアップが作成されるまで、表スペース内の表にあるデータを更新したり削除したりすることはできません。ただし、SELECT ステートメントを使用して、任意の表にあるデータにアクセスすることは可能です。

COPY YES

ロードするデータのコピーが保管されるように指定します。順方向回復が無効になっている場合 (*logretain* と *userexit* の両方がオフの場合)、このオプションは無効です。このオプションは、DATALINK 列のある表に対してはサポートされていません。

USE TSM

コピーは、Tivoli Storage Manager (TSM) を使用して格納されます。

OPEN num-sess SESSIONS

TSM またはベンダー製品で使われる入出力セッションの数。デフォルト値は 1 です。

TO device/directory

コピーのイメージを作成する装置またはディレクトリーを指定します。 OS/2 の場合、テープはサポートされていません。また、SCO UnixWare 7 上で動作する DB2 サーバーの場合、テープへのコピーはサポートされていません。

LOAD lib-name

使用するベンダー提供のバックアップおよび復元の入出力機能が含まれている共用ライブラリー (OS/2 や Windows オペレーティング・システムの DLL) の名前。名前にはフルパスを含め

ることができます。フルパスを指定しない場合、ユーザー出口プログラムが置かれているパスがデフォルトとして使われます。

CPU_PARALLELISM n

レコードの構文解析、変換、および形式設定を実行するためにロード・ユーティリティーが表オブジェクトの構築時に作成するプロセスまたはスレッドの数を指定します。このパラメーターは区画内の並列処理を利用するように設計されています。ソース・データ内のレコードの順序は保持されるため、これはあらかじめソートされたデータをロードする場合に特に役立ちます。このパラメーターの値が 0 または指定されていない場合、実行時にロード・ユーティリティーは、インテリジェント・デフォルト値 (通常は使用可能な CPU 数に基づく) を使用します。

注:

1. LOB または LONG VARCHAR フィールドのいずれかが含まれる表でこのパラメーターを使用した場合は、システムの CPU 数やユーザーが指定した値には関係なく、その値は 1 になります。
2. SAVECOUNT パラメーターに小さな値を指定すると、データと表のメタデータの両方をフラッシュするためにローダーがより多くの入出力操作を実行するようになります。 CPU_PARALLELISM が 1 より大きい場合にはフラッシュの操作が非同期に実行されるため、ローダーは CPU を活用することができます。
   CPU_PARALLELISM が 0 に設定されていると、ローダーは一貫性ポイントの間、入出力を待機します。 CPU_PARALLELISM を 2 に設定し、SAVECOUNT を 10 000 に設定してロード操作を実行すると、CPU が 1 個しかなくても、同じ操作で CPU_PARALLELISM を 1 に設定した場合より短い時間で完了します。

DATA BUFFER buffer-size

ユーティリティー内でデータを転送するためのバッファー用スペースとして使用する 4KB 単位のページ数を、並列処理の程度に関係なく指定します。指定した値が計算上の最小値よりも小さい場合には最小限必要な資源が使われ、警告は戻されません。

このメモリーはユーティリティー・ヒープから直接割り当てられます。ユーティリティー・ヒープのサイズは *util_heap_sz* データベース構成パラメーターで変更できます。

この値を指定しないなら、インテリジェント・デフォルトが実行時にユーティリティーによって算出されます。このデフォルトは、ローダーの

インスタンス生成時において使用可能なユーティリティー・ヒープ内の空きスペースのパーセント値と、表の一部の特性に基づいて決められます。

DATALINK SPECIFICATION

各 DATALINK 列ごとに、列指定を括弧で囲んで 1 つだけ指定できます。各列指定は、1 つ以上の DL_LINKTYPE、接頭部、および DL_URL_SUFFIX 指定で構成されます。接頭部の指定は DL_URL_REPLACE_PREFIX か DL_URL_DEFAULT_PREFIX のどちらかです。

表で定義されている DATALINK 列の数と同数の DATALINK 列指定を指定できます。列指定の順序は、*insert-column* リスト内での DATALINK 列の順序に従います。 *insert-column* リストが指定されていない場合は、表定義の中での順序に従います。

DISK_PARALLELISM n

データを表スペース・コンテナーに書き込むためにロード・ユーティリティーが作成するプロセスまたはスレッドの数を指定します。値を指定しないなら、ユーティリティーは表スペース・コンテナーの数と表の特性に基づいてインテリジェント・デフォルトを選択します。

DL_LINKTYPE

これを指定する場合、列定義の LINKTYPE と一致している必要があります。それで、列定義で LINKTYPE URL を指定しているなら、DL_LINKTYPE URL が受け入れ可能になります。

DL_URL_DEFAULT_PREFIX ″prefix″

これを指定した場合、これは同じ列内のすべての DATALINK 値のデフォルト接頭部となります。このコンテキストで接頭部とは、URL 指定のうち ″スキーム・ホスト・ポート″ 部分のことです。 (分散ファイル・システム (DFS) の場合、接頭部とは、 URL 指定のうち ″スキーム・セル名ファイル・スペース接合″ 部分のことです。)

接頭部の例を以下に示します。

```
"http://server"
"file://server"
"file:"
"http://server:80"
"dfs://.../cellname/fs"
```

DL_URL_DEFAULT_PREFIX でデフォルト接頭部が指定されているなら、列データに接頭部がない場合、ヌルでない列値にはデフォルト接頭部が付けられます。

たとえば、DL_URL_DEFAULT_PREFIX でデフォルト接頭部 "http://toronto" が指定されている場合、

- 列の入力値 ″/x/y/z″ は ″http://toronto/x/y/z″ として格納されます。
- 列の入力値 ″http://coyote/a/b/c″ は ″http://coyote/a/b/c″ として格納されます。
- 列の入力値がヌルなら、ヌルとして格納されます。

DL_URL_REPLACE_PREFIX ″prefix″

この文節は、以前にエクスポート・ユーティリティーによって生成されたデータのロードまたはインポートにおいて、データ内のホスト名を別のホスト名に一括して置換したい場合に役立ちます。これを指定した場合、これはヌルでないすべての 列値の接頭部になります。列値に接頭部がある場合、それは置換されます。列値に接頭部がない場合は、列値の接頭部として DL_URL_REPLACE_PREFIX で指定された接頭部が付けられます。分散ファイル・システム (DFS) の場合、接頭部とは、URL 指定のうち ″スキーム・セル名ファイル・スペース接合″ 部分のことです。

たとえば、DL_URL_REPLACE_PREFIX で接頭部 "http://toronto" が指定されている場合、

- 列の入力値 ″/x/y/z″ は ″http://toronto/x/y/z″ として格納されます。
- 列の入力値 ″http://coyote/a/b/c″ は ″http://toronto/a/b/c″ として格納されます。 ″coyote″ から ″toronto″ に置換されたことに注意してください。
- 列の入力値がヌルなら、ヌルとして格納されます。

DL_URL_SUFFIX ″suffix″

これを指定した場合、列のうちヌルでないすべての列値にこれが付加されます。実際には、DATALINK 値のデータ・ロケーション部分の ″パス″ 構成要素に付加されます。

FOR EXCEPTION table-name

エラーになった行のコピー先の例外表を指定します。固有索引や基本キー索引に違反している行はすべてコピーされます。さらに、DATALINK 例外も例外表に取り込まれます。修飾なしの表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

例外表に書き込まれる情報は、ダンプ・ファイルには書き込まれません (dumpfile 修飾子については、133ページの表8 を参照)。区分データベース環境の場合、例外表は、ロードする表が定義されているノードに対して定義する必要があります。一方、ダンプ・ファイルには、無効だっ

たため、または構文エラーのためにロードできない行が入れられます。詳しくは、147ページの『例外表』を参照してください。

FROM filename/pipename/device

ロードするデータが入っているファイル、パイプ、または装置を指定します。このファイル、パイプ、または装置は、 CLIENT オプションが指定されている場合を除いて、データベースの存在するノード上になければなりません。複数の名前を指定すると、それらは順番に処理されます。最後に指定した項目がテープ装置の場合、ユーザーは別のテープを入れるよう要求されます。有効な応答オプションは以下のとおりです。

c 続行。警告メッセージを生成した装置をそのまま使用します (たとえば新しいテープを装着する場合など)。

d 装置を終了。警告メッセージを生成した装置の使用を中止します (たとえばもうテープがない場合など)。

t 終了。すべての装置を終了します。

注:

1. テープは OS/2 ではサポートされていません。
2. できる限り完全修飾したファイル名を使ってください。サーバーがリモートの場合、完全修飾したファイル名を使用する必要があります。データベースが呼び出し元と同じノードにある場合は、相対パスが使われることがあります。
3. 複数の IXF ファイルからのデータのロードは、ファイルが物理的に分割されていても論理的に 1 つのファイルであればサポートされます。ファイルが論理的にも物理的にも分割されている場合にはサポートされません。
4. OS/2 で *pipename* を指定するときに、予期していたより少ない量のデータをロードする場合は、システム資源をクリーンアップして (IPL することをお勧めします)、 LOAD コマンドを再発行してください。
5. クライアント・マシンにあるデータをロードする場合、データは完全修飾ファイルまたは名前付きパイプのいずれかの形式でなければなりません。

HOLD QUIESCE

ロード操作後にユーティリティーが表を静止排他状態のままにしておくよう指定します。表スペースの静止状態を解除するには、次のコマンドを発行します。

```
db2 quiesce tablespaces for table <tablename> reset
```

注: *ファントム静止* が作成されないことを確認してください (*コマンド解説書* を参照)。

INDEXING MODE

ロード・ユーティリティーが索引を再作成するか、それとも索引を増分的に拡張するかを指定します。有効な値は以下のとおりです。

AUTOSELECT

ロード・ユーティリティーが REBUILD モードか INCREMENTAL モードかを自動的に決定します。

REBUILD

すべての索引を再作成します。ユーティリティーには、元の表データと追加表データの両方におけるすべての索引キー部分をソートするのに十分な資源が必要です。

INCREMENTAL

索引を新しいデータで拡張します。この方法では、索引の空きスペースを消費します。必要となるのは、挿入したレコードの索引キーを追加するのに十分なソートのスペースだけです。この方式は、ロード操作の開始時に索引オブジェクトが有効かつアクセス可能である場合にのみサポートされます (たとえば DEFERRED モードを指定したロード操作の直後には有効ではありません)。このモードを指定したにもかかわらず索引の状態が原因でサポートされない場合は警告が戻され、ロード操作は REBUILD モードで続行します。同じように、ロード構築フェーズでロード再開操作が始まった場合、INCREMENTAL モードはサポートされません。

増分索引作成は、以下のすべての条件を満たしている場合にはサポートされません。

- LOAD COPY オプションが指定されている場合 (*logretain* または *userexit* が有効になっている場合)。
- 表が DMS 表スペースにある。
- 索引オブジェクトが置かれている表スペースが、ロードする表に属する他の表オブジェクトと共有している。

この制約事項を回避するため、索引は別々の表スペースに配置するようにしてください。

DEFERRED

このモードが指定されている場合、ロード・ユーティリティーは索引を作成しようとしません。索引には、最新表示が必要で

あるというマークが付けられます。ロード操作とは関連のないそのような索引に最初にアクセスすると強制的に再作成される場合 (詳しくは*管理の手引き* を参照) と、データベースの再始動時に索引が再作成される場合があります。この方法では、最も大きい索引のすべてのキー部分をソートするために十分なスペースが必要になります。その後で索引の構築にかかる合計時間は、REBUILD モードで必要とされる時間よりも長くなります。そのため、パフォーマンスの観点から、複数のロード操作を据え置き索引作成により実行する場合は、ロード以外の最初のアクセス時に索引を再作成するのではなく、一連のロード操作の最後の操作で索引の再作成を実行するようにしてください。

据え置き索引作成がサポートされるのは非固有索引を持つ表の場合だけなので、ロード・フェーズ中に挿入された重複キーはロード操作後には持続しません。

注: DATALINK 列がある表では、据え置き索引作成はサポートされません。

INSERT

ロード・ユーティリティーを実行できる 4 つのモードの中の 1 つ。既存の表データに変更を加えることなく、ロード・データを表に追加します。

insert-column

データの挿入先となる表列を指定します。

ロード・ユーティリティーは、名前に 1 つまたは複数のスペースが含まれている列を解析することができません。たとえば、

```
db2 load from delfile1 of del modified by noeofchar noheader
   method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
   insert into table1 (BLOB1, S2, I3, Int 4, I5, I6, DT7, I8, TM9)
```

この例では Int 4 という列があるため失敗します。これを解決する方法は、そのような列の名前を二重引用符で囲むことです。

```
db2 load from delfile1 of del modified by noeofchar noheader
   method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
   insert into table1 (BLOB1, S2, I3, "Int 4", I5, I6, DT7, I8, TM9)
```

INTO table-name

データのロード先となるデータベース表を指定します。システム表または宣言された一時表を指定することはできません。別名または完全修飾の表名や修飾なしの表名は指定できます。修飾された表名は、

schema.tablename という形式です。修飾なしの表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

LOBS FROM lob-path

ロードする LOB 値が入っているデータ・ファイルへのパス。パスの末尾には必ずスラッシュ (/) を付けてください。 CLIENT オプションが指定されている場合、このパスは完全修飾パスでなければなりません。LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) のうち LOB 列にロードされる列に格納されます。 *filetype-mod* ストリング中に `lobsinfile` が指定されていない場合、このオプションは無視されます (133ページの表8 を参照)。

MESSAGES message-file

ロード操作中に発生する警告およびエラー・メッセージの出力先を指定します。メッセージ・ファイルを指定しない場合、メッセージは標準出力に書き込まれます。ファイルの完全パスを指定しなければ、ロード・ユーティリティーはデフォルト・ドライブと現行ディレクトリーを宛先として使用します。既存ファイルの名前を指定すると、ユーティリティーは情報をそこに追加します。

メッセージ・ファイルには、ロード操作の最後でメッセージが書き込まれるので、操作の進行をモニターするのには適していません。ロード操作のリアルタイム・モニターについては、 109ページの『LOAD QUERY コマンド』を参照してください。

METHOD

L データのロード元となる開始列番号と終了列番号を指定します。列番号は、データ行の始まりからのバイト・オフセットです。この番号は 1 から始まります。

注: この方式は ASC ファイル専用であり、そのファイル・タイプで有効な唯一のオプションです。

N データ・ファイルのうちロードする列の名前を指定します。指定する列の名前の大文字小文字の別は、システム・カタログ内の対応する名前の大文字小文字の別と一致していなければなりません。ヌル可能でない表列のそれぞれには、対応する項目が METHOD N リストになければなりません。たとえば、データ・フィールドが F1、F2、F3、F4、F5、F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、C4 INT であるとすると、 `method N (F2, F1, F4, F3)` は有効な要求ですが、 `method N (F2, F1)` は無効です。

注: この方式は、IXF ファイルでのみ使用できます。

P ロードする入力データ・フィールドの索引 (1 から始まる番号) を指定します。ヌル可能でない表列のそれぞれには、対応する項目が METHOD P リストになければなりません。たとえば、データ・フィールドが F1、F2、F3、F4、F5、F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、C4 INT であるとすると、 `method P (2, 1, 4, 3)` は有効な要求ですが、 `method P (2, 1)` は無効です。

注: この方式は IXF または DEL ファイル専用であり、DEL ファイル・タイプではこれが唯一の有効なオプションです。

MODIFIED BY filetype-mod
追加オプションを指定します (133ページの表8 を参照)。

NONRECOVERABLE
ロード・トランザクションを回復不能としてマークし、それ以降にロールフォワードを実行しても回復できなくするように指定します。ロールフォワード・ユーティリティーはそのトランザクションをスキップし、データのロード先の表を "invalid" (無効) とマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。 ロールフォワード操作の完了後、その表は、回復不能なロード操作の完了に続くコミット・ポイント後に作成されたバックアップ (完全または表スペース) からのみ除去または復元できます。

このオプションを指定すると、表スペースはロード操作後にバックアップ保留状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。

このオプションは、FILE LINK CONTROL 属性の指定された DATALINK 列が表に存在する場合、またはそのような列を表に追加している場合には使わないでください。

NULL INDICATORS null-indicator-list
このオプションを使用できるのは、 METHOD L パラメーターを指定したときだけです。つまり、入力ファイルが ASC ファイルであるときです。ヌル標識リストは、各ヌル標識フィールドの列番号を指定する正整数をコンマで区切ったリストです。列番号は、データ行の先頭からのヌル標識フィールドのバイト・オフセットです。ヌル標識リスト内には、 METHOD L パラメーターで定義されているデータ・フィールドごとに 1 つずつ項目がなければなりません。列番号ゼロは、対応するデータ・フィールドに常にデータが入っていることを示します。

ヌル標識列の値が Y なら、それはその列データがヌルになることを指定します。ヌル標識列が Y *以外* の文字である場合、その列データはヌルではなく、 METHOD L オプションによって指定された列データがロードされます。

ヌル標識の文字は MODIFIED BY オプションで変更できます (133ページの表8 の中の nullindchar 修飾子についての説明を参照)。

OF filetype

入力ファイル内のデータの形式を指定します。

- ASC (区切りなし ASCII 形式)
- DEL (区切り付き ASCII 形式)
- IXF (統合交換フォーマット、PC バージョン)。同じかまたは別の DB2 表からエクスポートされたもの。

ファイル形式については、 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

REPLACE

ロード・ユーティリティーを実行できる 4 つのモードの中の 1 つ。表から既存のデータをすべて削除してから、ロード・データを挿入します。表定義と索引定義は変更されません。階層間でのデータ移動にこのオプションを使用した場合、個々の副表ではなく階層全体についてのデータだけが置換できます。

このオプションは、DATALINK 列のある表に対してはサポートされていません。

RESTART

ロード・ユーティリティーを実行できる 4 つのモードの中の 1 つ。以前に中断されたロード操作を再開します。ロード操作は、ロード、構築、または削除フェーズのうち最後の一貫性ポイントから自動的に続行します。

RESTARTCOUNT

予約済み。

ROWCOUNT n

ファイルのうちロードする物理レコードの数を *n* に指定します。これを指定すると、ファイル内の最初の *n* 行だけがロードされます。

SAVECOUNT n

ロード・ユーティリティーが *n* 行ごとに一貫性ポイントを確立するように指定します。この値はページ・カウントに変換され、エクステン

ト・サイズの間隔に切り上げられます。各一貫性ポイントごとにメッセージが出されるため、109ページの『LOAD QUERY コマンド』を使ってロード操作をモニターする場合はこのオプションを選択してください。 *n* の値が十分に大きくないなら、各一貫性ポイントで実行される活動の同期化がパフォーマンスに影響します。

デフォルト値は 0 であり、それは必要がない限り一貫性ポイントが確立されないことを意味します。

SORT BUFFER buffer-size
予約済み。

STATISTICS NO
統計情報を収集せず、カタログ内の統計情報を変更しないことを指定します。これはデフォルトです。

STATISTICS YES
表と既存の索引に対して統計情報を収集することを指定します。このオプションは、ロード操作が REPLACE モードになっている場合にのみサポートされます。

WITH DISTRIBUTION
分布統計情報を収集することを指定します。

AND INDEXES ALL
表統計情報と索引統計情報の両方を収集することを指定します。

FOR INDEXES ALL
索引統計情報だけを収集することを指定します。

DETAILED
拡張索引統計情報を収集することを指定します。

TEMPFILES PATH temp-pathname
ロード操作中の一時ファイル作成時に使用するパスの名前を指定します。これはサーバー・ノードに従って完全に修飾されていなければなりません。

一時ファイルはファイル・システムのスペースを占有します。必要となるスペースがかなりの大きさになる場合もあります。すべての一時ファイルに割り当てられるファイル・システムのスペースがどれだけの大きさになるかの概算を以下に示します。

- DATALINK 値が入っている重複行または拒否された行ごとに 4 バイト

- ロード・ユーティリティーが生成するメッセージごとに 136 バイト
- データ・ファイルに長フィールド・データや LOB が含まれている場合は 15KB のオーバーヘッド。 INSERT オプションが指定されており、大量の長フィールド・データや LOB データがすでに表に存在しているなら、この量はかなり大きくなることがあります。

一時ファイルについては、148ページの『ロード一時ファイル』を参照してください。

TERMINATE

ロード・ユーティリティーを実行できる 4 つのモードの中の 1 つ。以前に中断されたロード操作を終了し、一貫性ポイントを過ぎていてもその操作を開始時点までロールバックします。この操作に関係している表スペースの状態はすべて通常に戻り、すべての表オブジェクトが整合性のある状態にされます (索引オブジェクトが無効としてマークされることがありますが、そのような場合は次回のアクセス時に索引の再作成が自動的に実行されます)。終了するロード操作が load REPLACE の場合は、load TERMINATE 操作の後にその表の内容が切り捨てられ、空の表になります。終了するロード操作が load INSERT の場合は、load TERMINATE 操作の後にその元のレコードはすべて保持されます。

ロード終了オプションを指定しても、表スペースのバックアップ保留状態は解除されません。

注: このオプションは、DATALINK 列のある表に対してはサポートされていません。

USING directory

予約済み。

WARNINGCOUNT n

警告が *n* 回出された後、ロード操作を停止します。このパラメーターは、警告を予期してはいないものの、使われているファイルや表が正しいかどうかを検査したい場合に設定します。 *n* が 0 になっている場合、またはこのオプションを指定していない場合、出される警告の数に関係なくロード操作は続行します。警告の限界値に達したためにロード操作が停止した場合には、別のロード操作を RESTART モードで開始することができます。ロード操作は、最後の一貫性ポイントから自動的に続行します。別の方法として、別のロード操作を REPLACE モードで、入力ファイルの先頭から開始することもできます。

WITHOUT PROMPTING

ロードするすべてのファイルがデータ・ファイルのリストに入ってお

り、リストに示されている装置やディレクトリーがロード操作全体で十分であることを指定します。連結入力ファイルが見つからない場合、あるいはロード操作が終了する前にコピーの宛先がいっぱいになった場合、ロード操作は失敗し、表はロード保留状態のままになります。

このオプションが指定されておらず、テープ装置がコピー・イメージのテープの終わりを検出した場合、あるいはリストの最後に示されている項目がテープ装置である場合、ユーザーは新しいテープを装置に入れるよう要求されます。テープは OS/2 ではサポートされていません。

LOAD QUERY コマンド

ロード操作の処理中の状態を検査します。このコマンドを正常に呼び出すには、同じデータベースへの接続だけでなく別個の CLP セッションも必要になります。このコマンドは、ローカル・ユーザーでもリモート・ユーザーでも使用できます。

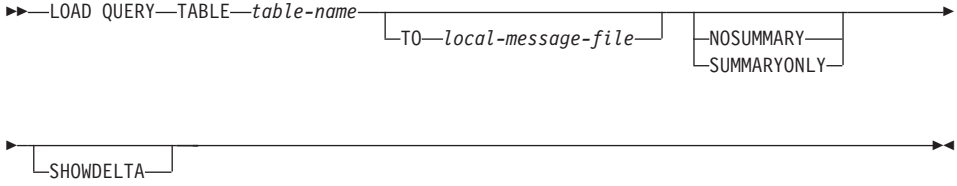
許可

なし

必要な接続

データベース

コマンド構文



コマンド・パラメーター

NOSUMMARY
ロード要約情報 (読み取った行数、スキップした行数、ロードした行数、拒否された行数、削除した行数、コミットした行数、および警告の数) を何も報告しないよう指定します。

SHOWDELTA
新しい情報 (最後の LOAD QUERY コマンドの呼び出し以後に発生したロード・イベントに関係した情報) だけを報告するよう指定します。

SUMMARYONLY
ロード要約情報だけを報告するよう指定します。

TABLE table-name
現在データをロードしているロード先の表の名前を指定します。修飾なしの表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

TO local-message-file
ロード操作中に発生する警告およびエラー・メッセージの出力先を指定

します。 LOAD コマンドに指定した *message-file* をこのファイルに指定することはできません。ファイルがすでに存在しているなら、ロード・ユーティリティーの生成したすべてのメッセージはそれに追加されます。

例

STAFF 表に大量のデータをロードしているユーザーが、そのロード操作の状態を調べるとします。その場合、このユーザーは次のように指定できます。

```
db2 connect to <database>
db2 load query table staff to /u/mydir/staff.tempmsg
```

出力ファイル /u/mydir/staff.tempmsg の内容は次のようになります。

```
SQL3500W The utility is beginning the "LOAD" phase at time
"02-13-1997 19:40:29.645353".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3109N The utility is beginning to load data from file
"/u/mydir/data/staffbig.ixf".

SQL3150N The H record in the PC/IXF file has product "DB2 01.00",
date "19970111", and time "194554".

SQL3153N The T record in the PC/IXF file has name
"data/staffbig.ixf", qualifier " ", and source " ".

SQL3519W  Begin Load Consistency Point. Input record count =
"111152".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count =
"222304".

SQL3520W  Load Consistency Point was successful.
```

参照

94ページの『LOAD コマンド』

ロード API

C API 構文

```
/* File: sqlutil.h */
/* API: Load */
/* ... */
SQL_API_RC SQL_API_FN
  sqluload (
    sqlu_media_list * pDataFileList,
    sqlu_media_list * pLobPathList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pLocalMsgFileName,
    char * pRemoteMsgFileName,
    short CallerAction,
    struct sqluload_in * pLoadInfoIn,
    struct sqluload_out * pLoadInfoOut,
    sqlu_media_list * pWorkDirectoryList,
    sqlu_media_list * pCopyTargetList,
    sqlint32 * pNullIndicators,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

汎用 API 構文

```
/* File: sqlutil.h */
/* API: Load */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgload (
    unsigned short FileTypeLen,
    unsigned short LocalMsgFileNameLen,
    unsigned short RemoteMsgFileNameLen,
    sqlu_media_list * pDataFileList,
    sqlu_media_list * pLobPathList,
    struct sqldcol * pDataDescriptor,
    struct sqlchar * pActionString,
    char * pFileType,
    struct sqlchar * pFileTypeMod,
    char * pLocalMsgFileName,
    char * pRemoteMsgFileName,
    short CallerAction,
    struct sqluload_in * pLoadInfoIn,
    struct sqluload_out * pLoadInfoOut,
    sqlu_media_list * pWorkDirectoryList,
    sqlu_media_list * pCopyTargetList,
    sqlint32 * pNullIndicators,
    void * pReserved,
    struct sqlca * pSqlca);
/* ... */
```

API パラメーター

FileTypeLen
入力。ファイル・タイプの長さをバイト数で表す 2 バイトの符号なし整数。

LocalMsgFileNameLen
入力。ローカル・メッセージ・ファイル名の長さをバイト数で表す 2 バイトの符号なし整数。

RemoteMsgFileNameLen
入力。一時ファイルのパス名の長さをバイト数で表す 2 バイトの符号なし整数。

pDataFileList
入力。ソース・ファイル、装置、ベンダー、またはパイプのリストを提供するのに使用する *sqlu_media_list* 構造体へのポインター。テープは OS/2 ではサポートされていません。

この構造体の情報は、*media_type* フィールドの値によって異なります。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLU_SERVER_LOCATION
media_type フィールドがこの値に設定された場合、呼び出し元は *sqlu_location_entry* 構造体によって情報を提供します。提供される *sqlu_location_entry* 構造体の数は、*sessions* フィールドに示されます。これはファイル、装置、名前付きパイプの場合に使用します。

SQLU_CLIENT_LOCATION
media_type フィールドがこの値に設定された場合、呼び出し元は *sqlu_location_entry* 構造体によって情報を提供します。提供される *sqlu_location_entry* 構造体の数は、sessions フィールドに示されます。これは完全修飾ファイルおよび名前付きパイプの場合に使用します。この *media_type* が有効なのは、リモートで接続されているクライアントを使用して API を呼び出している場合だけであることに注意してください。

SQLU_TSM_MEDIA
media_type フィールドがこの値に設定された場合、*sqlu_vendor* 構造体が使われます。 *filename* には、ロードするデータの固有識別子を指定します。 *sessions* の値には関係なく、*sqlu_vendor* 項目は 1 つだけです。 *sessions* フィールドは、開始する TSM セッションの数を示します。ロード・ユーティリティーは別個の順序番号を使ってセッションを開始しますが、1 つの *sqlu_vendor* 項目の中ではデータは同じです。

SQLU_OTHER_MEDIA
media_type フィールドがこの値に設定された場合、*sqlu_vendor* 構造体が使われます。 *shr_lib* には共用ライブラリーの名前、また *filename* にはロードするデータの固有識別子を指定します。 *sessions* の値には関係なく、*sqlu_vendor* 項目は 1 つだけです。 *sessions* フィールドは、開始するその他のベンダー・セッションの数を示します。ロード・ユーティリティーは別個の順序番号を使ってセッションを開始しますが、1 つの *sqlu_vendor* 項目の中ではデータは同じです。

ファイル名を指定する場合は、必ず完全に修飾してください。詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

pLobPathList
入力。 *sqlu_media_list* 構造体へのポインター。ファイル・タイプが

IXF、ASC、DEL の場合は、ロードする個々の LOB ファイルの位置を示す完全修飾パスまたは装置のリスト。ファイル名は IXF、ASC、または DEL のいずれかのファイルの中に入っており、指定されたパスに付加されます。テープは OS/2 ではサポートされていません。

この構造体の情報は、*media_type* フィールドの値によって異なります。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLU_LOCAL_MEDIA
この値に設定された場合、呼び出し元は *sqlu_media_entry* 構造体によって情報を提供します。提供される *sqlu_media_entry* 構造体の数は、*sessions* フィールドに示されます。

SQLU_TSM_MEDIA
この値に設定された場合、*sqlu_vendor* 構造体が使われます。*filename* には、ロードするデータの固有識別子を指定します。*sessions* の値には関係なく、*sqlu_vendor* 項目は 1 つだけです。 *sessions* フィールドは、開始する TSM セッションの数を示します。ロード・ユーティリティーは別個の順序番号を使ってセッションを開始しますが、1 つの *sqlu_vendor* 項目の中ではデータは同じです。

SQLU_OTHER_MEDIA
この値に設定された場合、*sqlu_vendor* 構造体が使われます。*shr_lib* には共用ライブラリーの名前、また *filename* にはロードするデータの固有識別子を指定します。 *sessions* の値には関係なく、*sqlu_vendor* 項目は 1 つだけです。 *sessions* フィールドは、開始するその他のベンダー・セッションの数を示します。ロード・ユーティリティーは別個の順序番号を使ってセッションを開始しますが、1 つの *sqlu_vendor* 項目の中ではデータは同じです。

詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

pDataDescriptor
入力。外部ファイルからのロードで選択する列に関する情報が入っている *sqldcol* 構造体へのポインター。

pFileType パラメーターを SQL_ASC に設定した場合、この構造体の *dcolmeth* フィールドは SQL_METH_L に設定しなければなりません。ユーザーはロードする列ごとに開始位置と終了位置を指定します。

ファイル・タイプが SQL_DEL の場合、*dcolmeth* には SQL_METH_P または SQL_METH_D のいずれかを指定できます。 SQL_METH_P を指定した場

合には、ソース列の位置をユーザーが指定する必要があります。SQL_METH_D を指定した場合、ファイルの最初の列が表の最初の列にロードされる、という具合になります。

ファイル・タイプが SQL_IXF の場合、*dcolmeth* には SQL_METH_P、SQL_METH_D、または SQL_METH_N のいずれかを指定できます。SQL_METH_N はファイルの列名が *sqldcol* 構造体で提供されることを示しており、それ以外は DEL ファイルの規則が適用されます。

詳しくは、 *管理 API 解説書* の『SQLDCOL』を参照してください。

pActionString

入力。 *sqlchar* 構造体へのポインター。この構造体には、2 バイト長フィールドに続いて、表に影響を与えるアクションを指定する文字配列が含まれます。

文字配列の形式は以下のとおりです。

```
"INSERT|REPLACE|RESTART|TERMINATE
INTO tbname [(column_list)]
[DATALINK SPECIFICATION datalink-spec]
[FOR EXCEPTION e_tbname]"
```

INSERT

既存の表データに変更を加えることなく、ロード・データを表に追加します。

REPLACE

表から既存のデータをすべて削除してから、ロード・データを挿入します。表定義と索引定義は変更されません。

RESTART

以前に中断されたロード操作を再開します。ロード操作は、ロード、構築、または削除フェーズのうち最後の一貫性ポイントから自動的に続行します。

TERMINATE

以前に中断されたロード操作を終了し、一貫性ポイントを過ぎていてもその操作を開始時点までロールバックします。この操作に関係している表スペースの状態はすべて通常に戻り、すべての表オブジェクトが整合性のある状態にされます (索引オブジェクトが無効としてマークされることがありますが、そのような場合は次回のアクセス時に索引の再作成が自動的に実行されます)。表の属する表スペースがロード保留状態でない場合、このオプションは表スペースの状態には影響しません。

ロード終了オプションを指定しても、表スペースのバックアップ保留状態は解除されません。

tbname データのロード先の表の名前。システム表または宣言された一時表を指定することはできません。別名または完全修飾の表名や修飾なしの表名は指定できます。修飾された表名は、*schema.tablename* という形式です。修飾なしの表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

(*column_list***)**
データの挿入先となる表列の名前のリスト。列名と列名の間はコンマで区切ってください。また、名前にスペースや英小文字が含まれている場合は、二重引用符で囲む必要があります。

DATALINK SPECIFICATION *datalink-spec*
DB2 データ・リンクに関連するパラメーターを指定します。これらのパラメーターは LOAD コマンドの場合と同じ構文を使って指定できます (94ページの『LOAD コマンド』を参照)。

FOR EXCEPTION *e_tbname*
エラーになった行のコピー先の例外表を指定します。固有索引や基本キー索引に違反している行はすべてコピーされます。さらに、DATALINK 例外も例外表に取り込まれます。

pFileType
入力。外部ファイル内のデータの形式を示すストリング。サポートされる外部ファイル形式 (`sqlutil` で定義される) は、以下のとおりです。

SQL_ASC
区切りなし ASCII。

SQL_DEL
区切り付き ASCII。dBase、BASIC、IBM パーソナル・デシジョン・シリーズのプログラム、その他多くのデータベース・マネージャーおよびファイル・マネージャーとのデータ交換に使用できます。

SQL_IXF
PC バージョンの IXF (統合交換フォーマット)。表からのデータ・エクスポートとして望ましい方式であり、後でそのデータを同じ表または別のデータベース・マネージャー表へロードできます。

ファイル形式については、 225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

pFileTypeMod

入力。構造体へのポインター。この構造体には、2 バイト長フィールドに続いて、1 つ以上の処理オプションを指定する文字の配列が含まれます。このポインターがヌルか、またはポイント先の構造体の文字数が 0 の場合、このアクションはデフォルト指定を選択したものと解釈されます。

サポートされているすべてのファイル・タイプですべてのオプションが使えるわけではありません。

詳しくは、*管理 API 解説書* の『SQLCHAR』、および 133ページの『ファイル・タイプ修飾子 (ロード)』を参照してください。

pLocalMsgFileName

入力。出力メッセージの書き込み先ローカル・ファイルの名前を内容とするストリング。

pRemoteMsgFileName

入力。一時ファイルのためにサーバーで使われるパス名を内容とするストリング。一時ファイルはメッセージ、一貫性ポイント、削除フェーズ情報などを格納するために作成されます。 一時ファイルについては、148ページの『ロード一時ファイル』を参照してください。

CallerAction

入力。呼び出し元が要求するアクション。有効な値 (`sqlutil` で定義される) は、以下のとおりです。

SQLU_INITIAL

初期呼び出し。 API の最初の呼び出しでは、この値 (または SQLU_NOINTERRUPT) を使用しなければなりません。

SQLU_NOINTERRUPT

初期呼び出し。処理を中断しません。 API の最初の呼び出しでは、この値 (または SQLU_INITIAL) を使用しなければなりません。

初期呼び出しまたはそれ以降の呼び出しから戻る際、要求されたロード操作を完了する前に呼び出し元アプリケーションでなんらかのアクションを実行することが必要な場合は、この呼び出し元のアクションを以下のいずれかに設定する必要があります。

SQLU_CONTINUE
処理を続行します。この値は API の後続呼び出しでのみ使用できます。つまり、初期呼び出しから戻った時点でユーティリティーがユーザー入力 (「テープの終わり」状態への応答など) を要求した場合に、その後で使用します。これは、ユーティリティーの要求したユーザー処置が完了し、ユーティリティーが初期要求の処理を続行できるということを指定します。

SQLU_TERMINATE
処理を終了します。ロード・ユーティリティーを途中で終了し、ロードしている表スペースを LOAD_PENDING 状態のままにします。データをこれ以上処理する予定がない場合は、このオプションを指定してください。

SQLU_ABORT
処理を終了します。ロード・ユーティリティーを途中で終了し、ロードしている表スペースを LOAD_PENDING 状態のままにします。データをこれ以上処理する予定がない場合は、このオプションを指定してください。

SQLU_RESTART
処理を再開します。

SQLU_DEVICE_TERMINATE
装置を 1 つだけ終了します。ユーティリティーが該当する装置からのデータの読み取りを停止した場合に、それ以後もデータを処理したい場合は、このオプションを指定してください。

pLoadInfoIn
入力。追加の入力パラメーターが入っている *sqluload_in* 構造体へのポインター (オプション)。この構造体については、121ページの『データ構造体: SQLULOAD-IN』を参照してください。

pLoadInfoOut
出力。追加の出力パラメーターが入っている *sqluload_out* 構造体へのポインター (オプション)。この構造体については、126ページの『データ構造体: SQLULOAD-OUT』を参照してください。

pWorkDirectoryList
予約済み。

pCopyTargetList
入力。コピー・イメージが作成される場合に、コピー・イメージの書き込み先となるパス、装置、または共用ライブラリーのリストを提供するのに使われる *sqlu_media_list* 構造体へのポインター。

この構造体に含まれる値は、*media_type* フィールドの値によって異なります。このフィールドの有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLU_LOCAL_MEDIA
コピーをローカル・メディアに書き込む場合は、*media_type* をこの値に設定し、出力先に関する情報を *sqlu_media_entry* 構造体に指定します。提供される *sqlu_media_entry* 構造体の数は、*sessions* フィールドで指定します。

SQLU_TSM_MEDIA
コピーが TSM に書き込まれる場合はこの値を使用します。それ以上の情報は不要です。

SQLU_OTHER_MEDIA
ベンダー製品を使用する場合にはこの値を使用し、*sqlu_vendor* 構造体によってさらに情報を指定します。この構造体の *shr_lib* フィールドには、該当するベンダー製品の共用ライブラリー名を設定します。 *sessions* の値には関係なく、*sqlu_vendor* 項目は 1 つだけです。提供される *sqlu_media_entry* 構造体の数は、*sessions* フィールドで指定します。ロード・ユーティリティーは別個の順序番号を使ってセッションを開始しますが、1 つの *sqlu_vendor* 項目の中で提供されるデータは同じです。

詳しくは、*管理 API 解説書* の『SQLU-MEDIA-LIST』を参照してください。

pNullIndicators
入力。 ASC ファイル専用。列データがヌル可能かどうかを示す整数の配列。この配列の要素とデータ・ファイルからロードされる列とは、順番に 1 対 1 で対応します。つまり、要素の数は *pDataDescriptor* パラメーターの *dcolnum* フィールドと同じでなければなりません。配列の各要素には、ヌル標識フィールドとして使用されるデータ・ファイル内の位置を識別する番号が入れられます。それが 0 の場合は、その表列がヌル可能でないことを示しています。その要素が 0 でない場合、データ・ファイル内の指定された位置は Y または N でなければなりません。 Y はその表列のデータがヌルであり、N はその表列のデータがヌルでないことを示します。

pReserved
将来の利用のために予約済み。

pSqlca

出力。 *sqlca* 構造体へのポインター。この構造体については、 *管理 API 解説書* および *SQL 解説書* を参照してください。

REXX API 構文

この API は REXX から SQLDB2 インターフェースによって呼び出すことができます。 *アプリケーション開発の手引き* を参照してください。構文については、94ページの『LOAD コマンド』を参照してください。

データ構造体: SQLULOAD-IN

この構造体は、 111ページの『ロード API』の呼び出しにおいて情報を入力するのに使用します。

表 6. SQLULOAD-IN 構造体の中のフィールド

フィールド名	データ・タイプ	説明
SIZEOFSTRUCT	sqluint32	構造体のサイズ (バイト数)。
SAVECNT	sqluint32	一貫性ポイントの確立までにロードするレコードの数。この値はページ・カウントに変換され、エクステント・サイズの間隔に切り上げられます。各一貫性ポイントごとにメッセージが出されるため、128ページの『db2LoadQuery - ロード照会 API』を使ってロード操作をモニターする場合はこのオプションを選択してください。 *savecnt* の値が十分に大きくないなら、各一貫性ポイントで実行される活動の同期化がパフォーマンスに影響します。 デフォルト値は `0` であり、それは必要がない限り一貫性ポイントが確立されないことを意味します。
RESTARTCNT	sqluint32G	予約済み。
ROWCNT	sqluint32	ロードする物理レコードの数。これを指定すると、ファイル内の最初の *rowcnt* 行だけがロードされます。
WARNINGCNT	sqluint32	警告が *warningcnt* 回出された後、ロード操作を停止します。このパラメーターは、警告を予期してはいないものの、使われているファイルや表が正しいかどうかを検査したい場合に設定します。 *warningcnt* が `0` の場合、またはこのオプションを指定していない場合、出される警告の数に関係なくロード操作は続行します。 警告の限界値を超えたためにロード操作が停止した場合には、別のロード操作を RESTART モードで開始することができます。ロード操作は、最後の一貫性ポイントから自動的に続行します。別の方法として、別のロード操作を REPLACE モードで、入力ファイルの先頭から開始することもできます。

表 6. SQLULOAD-IN 構造体の中のフィールド (続き)

フィールド名	データ・タイプ	説明
DATA_BUFFER_SIZE	sqluint32	ユーティリティー内でデータを転送するためのバッファー用スペースとして使用する 4KB 単位のページ数を、並列処理の程度に関係なく指定します。指定した値が計算上の最小値よりも小さい場合には最小限必要のものが使われ、警告は戻されません。 このメモリーはユーティリティー・ヒープから直接割り当てられます。ユーティリティー・ヒープのサイズは *util_heap_sz* データベース構成パラメーターで変更できます。 この値を指定しないなら、インテリジェント・デフォルトが実行時にユーティリティーによって算出されます。このデフォルトは、ローダーのインスタンス生成時において使用可能なユーティリティー・ヒープ内の空きスペースのパーセント値と、表の一部の特性に基づいて決められます。
SORT_BUFFER_SIZE	sqluint32	予約済み。
HOLD_QUIESCE	UNSIGNED SHORT	ユーティリティーがロード後に表を静止排他状態にしておく場合は TRUE、それ以外の場合は FALSE に値を設定するフラグ。
RESTARTPHASE	CHAR(1)	予約済み。
STATSOPT	CHAR(1)	収集する統計情報の詳細度。値については下記参照。
CPU_PARALLELISM	UNSIGNED SHORT	レコードの構文解析、変換、および形式設定を実行するためにロード・ユーティリティーが表オブジェクトの構築時に作成するプロセスまたはスレッドの数。このパラメーターは区画内の並列処理を利用するように設計されています。ソース・データ内のレコードの順序は保持されるため、これはあらかじめソートされたデータをロードする場合に特に役立ちます。このパラメーターの値が 0 の場合、ロード・ユーティリティーは実行時にインテリジェント・デフォルト値を使用します。 **注:** LOB または LONG VARCHAR フィールドのいずれかが含まれる表でこのパラメーターを使用した場合は、システムの CPU 数やユーザーが指定した値には関係なく、その値は 1 になります。
DISK_PARALLELISM	UNSIGNED SHORT	データを表スペース・コンテナーに書き込むためにロード・ユーティリティーが作成するプロセスまたはスレッドの数。値を指定しないなら、ユーティリティーは表スペース・コンテナーの数と表の特性に基づいてインテリジェント・デフォルトを選択します。

表 6. SQLULOAD-IN 構造体の中のフィールド (続き)

フィールド名	データ・タイプ	説明
NON_RECOVERABLE	UNSIGNED SHORT	ロード・トランザクションを回復不能としてマークし、それ以降にロールフォワードを実行しても回復できなくする場合は、SQLU_NON_RECOVERABLE_LOAD に設定します。ロールフォワード・ユーティリティーはそのトランザクションをスキップし、データのロード先の表を ″invalid″ (無効) としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワードの完了後、このような表は除去することしかできません。 このオプションを指定すると、表スペースはロード操作後にバックアップ保留状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。 このロード・トランザクションを回復可能としてマークする場合は SQLU_RECOVERABLE_LOAD に設定します。
INDEXING_MODE	UNSIGNED SHORT	ロード・ユーティリティーが索引を再作成するか、それとも索引を増分的に拡張するかを指定します。値については下記参照。

STATSOPT の有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLU_STATS_NONE

SQL_STATS_EXTTABLE_ONLY

SQL_STATS_EXTTABLE_INDEX

SQL_STATS_INDEX

SQL_STATS_TABLE

SQL_STATS_EXTINDEX_ONLY

SQL_STATS_EXTINDEX_TABLE

SQL_STATS_ALL

SQL_STATS_BOTH

INDEXING_MODE の有効な値 (sqlutil で定義) は、以下のとおりです。

SQLU_INX_AUTOSELECT

SQLU_INX_REBUILD

SQLU_INX_INCREMENTAL

SQLU_INX_DEFERRED

これらの索引作成モードについては、 94ページの『LOAD コマンド』を参照してください。

言語構文

C 構造体

```
/* File: sqlutil.h */
/* Structure: SQLULOAD-IN */
/* ... */
SQL_STRUCTURE sqluload_in
{
  sqluint32       sizeOfStruct;
  sqluint32       savecnt;
  sqluint32       restartcnt;
  sqluint32       rowcnt;
  sqluint32       warningcnt;
  sqluint32       data_buffer_size;
  sqluint32       sort_buffer_size;  /* No longer used. */
  unsigned short  hold_quiesce;
  char            restartphase;
  char            statsopt;
  unsigned short  cpu_parallelism;
  unsigned short  disk_parallelism;
  unsigned short  non_recoverable;
  unsigned short  indexing_mode;
};
/* ... */
```

COBOL 構造体

```
* File: sqlutil.cbl
01 SQLULOAD-IN.
    05 SQL-SIZE-OF-STRUCT     PIC 9(9) COMP-5 VALUE 40.
    05 SQL-SAVECNT            PIC 9(9) COMP-5.
    05 SQL-RESTARTCOUNT       PIC 9(9) COMP-5.
    05 SQL-ROWCNT             PIC 9(9) COMP-5.
    05 SQL-WARNINGCNT         PIC 9(9) COMP-5.
    05 SQL-DATA-BUFFER-SIZE   PIC 9(9) COMP-5.
    05 SQL-SORT-BUFFER-SIZE   PIC 9(9) COMP-5.  * No longer used.
    05 SQL-HOLD-QUIESCE       PIC 9(4) COMP-5.
    05 SQL-RESTARTPHASE       PIC X.
    05 SQL-STATSOPT           PIC X.
    05 SQL-CPU-PARALLELISM    PIC 9(4) COMP-5.
    05 SQL-DISK-PARALLELISM   PIC 9(4) COMP-5.
    05 SQL-NON-RECOVERABLE    PIC 9(4) COMP-5.
    05 SQL-INDEXING-MODE      PIC 9(4) COMP-5.
*
```

データ構造体: SQLULOAD-OUT

この構造体は、 111ページの『ロード API』の呼び出し後に情報を出力するのに使用されます。

表 7. SQLULOAD-OUT 構造体の中のフィールド

フィールド名	データ・タイプ	説明
SIZEOFSTRUCT	sqluint32	構造体のサイズ (バイト数)。
ROWSREAD	sqluint32	ロード操作中に読み取ったレコードの数。
ROWSSKIPPED	sqluint32	ロード操作が始まる前にスキップされたレコード数。
ROWSLOADED	sqluint32	ターゲット表にロードされた行数。
ROWSREJECTED	sqluint32	ロードできなかったレコードの数。
ROWSDELETED	sqluint32	削除された重複行の数。
ROWSCOMMITTED	sqluint32	処理されたレコードの合計数。これは、データベースに正常にロードされたりコミットされたりしたレコードの数に、スキップされたり拒否されたりしたレコードの数を加算したものです。

言語構文

C 構造体

```
/* File: sqlutil.h */
/* Structure: SQLULOAD-OUT */
/* ... */
SQL_STRUCTURE sqluload_out
{
  sqluint32       sizeOfStruct;
  sqluint32       rowsRead;
  sqluint32       rowsSkipped;
  sqluint32       rowsLoaded;
  sqluint32       rowsRejected;
  sqluint32       rowsDeleted;
  sqluint32       rowsCommitted;
};
/* ... */
```

COBOL 構造体

```
* File: sqlutil.cbl
01 SQLULOAD-OUT.
    05 SQL-SIZE-OF-STRUCT     PIC 9(9) COMP-5 VALUE 28.
    05 SQL-ROWS-READ          PIC 9(9) COMP-5.
    05 SQL-ROWS-SKIPPED       PIC 9(9) COMP-5.
    05 SQL-ROWS-LOADED        PIC 9(9) COMP-5.
    05 SQL-ROWS-REJECTED      PIC 9(9) COMP-5.
    05 SQL-ROWS-DELETED       PIC 9(9) COMP-5.
    05 SQL-ROWS-COMMITTED     PIC 9(9) COMP-5.
*
```

db2LoadQuery - ロード照会 API

ロード操作の処理中の状態を検査します。

許可

なし

必要な接続

データベース

API 組み込みファイル

db2ApiDf.h

C API 構文

```
/* File: db2ApiDf.h */
/* API: Load Query */
/* ... */
SQL_API_RC SQL_API_FN
db2LoadQuery (
  db2Uint32 versionNumber,
  void * pParmStruct,
  struct sqlca *pSqlca);
typedef struct
{
  db2Uint32 iStringType;
  char * piString;
  db2Uint32 iShowLoadMessages;
  db2LoadQueryOutputStruct * poOutputStruct;
  char * piLocalMessageFile;
} db2LoadQueryStruct;
typedef struct
{
  db2Uint32 oRowsRead;
  db2Uint32 oRowsSkipped;
  db2Uint32 oRowsCommitted;
  db2Uint32 oRowsLoaded;
  db2Uint32 oRowsRejected;
  db2Uint32 oRowsDeleted;
  db2Uint32 oCurrentIndex;
  db2Uint32 oNumTotalIndexes;
  db2Uint32 oCurrentMPPNode;
  db2Uint32 oLoadRestarted;
  db2Uint32 oWhichPhase;
  db2Uint32 oWarningCount;
} db2LoadQueryOutputStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Load Query */
/* ... */
SQL_API_RC SQL_API_FN
db2gLoadQuery (
  db2Uint32 versionNumber,
  void * pParmStruct,
  struct sqlca *pSqlca);
typedef struct
{
  db2Uint32 iStringType;
  db2Uint32 iStringLen;
  char * piString;
  db2Uint32 iShowLoadMessages;
  db2LoadQueryOutputStruct * poOutputStruct;
  db2Uint32 iLocalMessageFileLen;
  char * piLocalMessageFile
} db2gLoadQueryStruct;
typedef struct
{
  db2Uint32 oRowsRead;
  db2Uint32 oRowsSkipped;
  db2Uint32 oRowsCommitted;
  db2Uint32 oRowsLoaded;
  db2Uint32 oRowsRejected;
  db2Uint32 oRowsDeleted;
  db2Uint32 oCurrentIndex;
  db2Uint32 oNumTotalIndexes;
  db2Uint32 oCurrentMPPNode;
  db2Uint32 oLoadRestarted;
  db2Uint32 oWhichPhase;
  db2Uint32 oWarningCount;
} db2LoadQueryOutputStruct;
/* ... */
```

API パラメーター

versionNumber

入力。第 2 のパラメーター *pParmStruct* として渡される構造体のバージョンとリリース・レベルを指定します。

pParmStruct

入力。 *db2LoadQueryStruct* 構造体へのポインター。

pSqlca

出力。 *sqlca* 構造体へのポインター。この構造体については、 *管理 API 解説書* および *SQL 解説書* を参照してください。

iStringType
入力。 *piString* のタイプを指定します。有効な値 (db2ApiDf.h で定義される) は、以下のとおりです。

DB2LOADQUERY_TABLENAME
db2LoadQuery API で使用する表名を指定することを示します。

iStringLen
入力。 *piString* の長さを指定します (バイト単位)。

piString
入力。 *iStringType* の値に応じて、一時ファイルのパス名または表名を指定します。

iShowLoadMessages
入力。ロード・ユーティリティーが戻すメッセージのレベルを指定します。有効な値 (db2ApiDf.h で定義される) は、以下のとおりです。

DB2LOADQUERY_SHOW_ALL_MSGS
すべてのロード・メッセージを戻します。

DB2LOADQUERY_SHOW_NO_MSGS
ロード・メッセージを何も戻しません。

DB2LOADQUERY_SHOW_NEW_MSGS
この API への最後の呼び出し以降に生成されたメッセージだけを戻します。

poOutputStruct
出力。ロード要約情報が入っている *db2LoadQueryOutputStruct* 構造体へのポインター。要約が不要の場合はヌルに設定してください。

iLocalMessageFileLen
入力。 *piLocalMessageFile* の長さを指定します (バイト単位)。

piLocalMessageFile
入力。出力メッセージ用に使用するローカル・ファイルの名前を指定します。

oRowsRead
出力。それまでにロード・ユーティリティーが読み取ったレコードの数。

oRowsSkipped
出力。ロード操作が始まる前にスキップされたレコード数。

oRowsCommitted
出力。それまでにターゲット表にコミットされた行数。

oRowsLoaded
出力。それまでにターゲット表にロードされた行数。

oRowsRejected
出力。それまでにターゲット表で拒否された行数。

oRowsDeleted
出力。それまでにターゲット表から削除された行数 (削除フェーズ中)。

oCurrentIndex
出力。現在構築中の索引 (構築フェーズ中)。

oCurrentMPPNode
出力。照会中のノードを示します (MPP モードの場合のみ)。

oLoadRestarted
出力。照会しているロード操作がロード再開操作である場合に、値が TRUE であるフラグ。

oWhichPhase
出力。照会しているロード操作の現時点でのフェーズを示します。有効な値 (db2ApiDf.h で定義される) は、以下のとおりです。

DB2LOADQUERY_LOAD_PHASE
ロード・フェーズ。

DB2LOADQUERY_BUILD_PHASE
構築フェーズ。

DB2LOADQUERY_DELETE_PHASE
削除フェーズ。

oNumTotalIndexes
出力。構築する索引の数 (構築フェーズ中)。

oWarningCount
出力。それまでに戻された警告の合計数。

REXX API 構文

この API は REXX から SQLDB2 インターフェースによって呼び出すことができます。 *アプリケーション開発の手引き* を参照してください。構文については、109ページの『LOAD QUERY コマンド』を参照してください。

サンプル・プログラム

C	¥sqllib¥samples¥c¥loadqry.sqc
COBOL	¥sqllib¥samples¥cobol¥loadqry.sqb

使用上の注意

この API は *piString* に指定された表に対するロード操作の状態を読み、*pLocalMsgFileName* で指定されるファイルにその状態を書き込みます。

ファイル・タイプ修飾子 (ロード)

表 8. 有効なファイル・タイプ修飾子 (ロード)

修飾子	説明
すべてのファイル形式	
anyorder	この修飾子は *cpu_parallelism* パラメーターに関連して使われます。ソース・データの順序を保持する必要がないことを指定します。これを指定すると、SMP システムではパフォーマンスがさらに大幅に向上します。 *cpu_parallelism* の値が 1 の場合、このオプションは無視されます。また、一貫性ポイントの後の破損回復にはデータを順序に従ってロードしなければならないため、SAVECOUNT > 0 の場合、このオプションはサポートされません。
fastparse	ユーザー提供の列値に対して簡略化した構文検査を実行し、パフォーマンスを向上させます。このオプションでロードした表は構造上正しいことが保証されており、セグメント化の違反やトラップを防ぐためにユーティリティーが十分なデータ検査を実行することも保証されています。データの形式が正しければ、正しくロードされます。 たとえば、ASC ファイル内のある整数列で `123qwr4` という値がフィールド項目として検出されると、ロード・ユーティリティーはその値が有効な番号を表すものでないため、通常は構文エラーのフラグを付けます。 `fastparse` を指定すると構文エラーは検出されないため、任意の番号が整数フィールドにロードされます。この修飾子を使用する場合は、クリーンなデータだけを指定するよう注意が必要です。このオプションを ASCII データに対して使うとパフォーマンスが大幅に向上することがありますが、`fastparse` を PC/IXF データに対して使ってもそれほどパフォーマンスは向上しません。 IXF はバイナリー形式であり、`fastparse` が影響を与えるのは ASCII から内部形式への解析および変換処理だからです。
generatedignore	この修飾子はロード・ユーティリティーに対して、すべての生成列のデータがデータ・ファイル内にあるものの、それらのデータは無視するべきものであることを通知します。ヌル可能な生成列の場合、列にはヌルがロードされます。ヌル不可の生成列の場合、生成列のデータ・タイプのデフォルト値がロードされます。ロード操作の最後に、 SET INTEGRITY ステートメントを呼び出して、ロードされた値を、生成列の定義に従って計算された値に強制置換することができます。この修飾子は、 `generatedmissing` または `generatedoverride` 修飾子とともに使用することはできません。
generatedmissing	この修飾子が指定されている場合、ユーティリティーは、生成列のデータが入力データ・ファイルに入っていない (ヌルも入っていない) ものと見なし、ヌルを列にロードします。ロード操作の最後に、 SET INTEGRITY ステートメントを使用して、ヌルを、生成列の定義に従って計算された値に置換することができます。この修飾子は、`generatedoverride` または `generatedoverride` 修飾子とともに使用することはできません。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
generatedoverride	この修飾子はロード・ユーティリティーに対し、表内のすべての生成列に関して、明示的な非ヌル・データを受け入れる (これらのタイプの列に関する通常の規則に反する) ように指示します。これが役立つのは、別のデータベース・システムからデータを移行する場合や、 ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用して回復したデータから表をロードする場合です。この修飾子を使用した場合、ヌル不可の生成列でデータまたはヌル・データの入っていない行は拒否されます (SQL3116W)。 **注:** このオプションが使用されていると、ロード・ユーティリティーは生成列の値の妥当性検査を行いません。 この修飾子は、 `generatedmissing` または `generatedignore` 修飾子とともに使用することはできません。
identityignore	この修飾子はロード・ユーティリティーに対して、識別列のデータがデータ・ファイル内にあるものの、それらのデータは無視するべきものであることを通知します。その結果、すべての識別値はユーティリティーが生成します。 GENERATED ALWAYS 識別列と GENERATED BY DEFAULT 識別列のどちらの場合も動作は同じになります。つまり、GENERATED ALWAYS 列の場合には、拒否される行はありません。この修飾子は、 `identitymissing` または `identityoverride` 修飾子とともに使用することはできません。
identitymissing	この修飾子が指定されている場合、ユーティリティーは、識別列のデータが入力データ・ファイルに入っていない (ヌルも入っていない) ものと見なし、行ごとに値を生成します。 GENERATED ALWAYS 識別列と GENERATED BY DEFAULT 識別列のどちらの場合も動作は同じになります。この修飾子は、 `identityignore` または `identityoverride` 修飾子とともに使用することはできません。
identityoverride	この修飾子を使用するのは、 GENERATED ALWAYS として定義されている識別列が、ロードされる表にある場合だけです。この修飾子はユーティリティーに対し、そのような列に関して、明示的な非ヌル・データを受け入れる (これらのタイプの識別列に関する通常の規則に反する) ように指示します。これが役立つのは、別のデータベース・システムからデータを移行するときに GENERATED ALWAYS として表を定義しなければならない場合や、 ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用して回復したデータから表をロードする場合です。この修飾子を使用した場合、識別列でデータまたはヌル・データの入っていない行は拒否されます (SQL3116W)。この修飾子は、 `identitymissing` または `identityignore` 修飾子とともに使用することはできません。 **注:** このオプションが使用されていると、ロード・ユーティリティーは、表の識別列内の値の固有性の保守または検査を行いません。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
indexfreespace=*x*	*x* は 0〜99 の整数です。この値は、索引のロード時に各索引ページごとに空きスペースとして残す部分のパーセント値として解釈されます。ページの最初の項目を追加する場合に制限はありません。それ以後の項目は、空きスペースに関するパーセントの限界値を超えない場合に追加されます。 CREATE INDEX の実行時に使われる値がデフォルト値になります。 この値は、CREATE INDEX ステートメントに指定された PCTFREE 値より優先され、索引の葉ページだけに影響します。
lobsinfile	*lob-path* で、LOB 値を含むファイルへのパスを指定します。ASC、DEL、または IXF ロード入力ファイルには、LOB 列内の LOB データが含まれているファイルの名前が入っています。
noheader	ヘッダー検査コードをスキップします (単一ノードのノードグループ内にある表へのロード操作にのみ適用されます)。 オートローダー・ユーティリティー (169ページの『第4章 オートローダー』を参照) は、複数のノードからなるノードグループ内の表へデータを提供する各ファイルにヘッダーを書き込みます。ヘッダーにはノード番号、区分化マップ、区分化キーの指定などが含まれています。ロード・ユーティリティーは、データが正しいノードでロードされていることを検査するためにこの情報を必要とします。ノードが 1 つしかないノードグループ上の表へファイルをロードする場合、ヘッダーは存在せず、このオプションが指定されているならロード・ユーティリティーはヘッダー検査コードをスキップします。
norowwarnings	拒否された行に関するすべての警告を抑止します。
pagefreespace=*x*	*x* は 0〜100 の整数です。この値は、各データ・ページごとに空きスペースとして残しておく分のパーセントとして解釈されます。 指定した値が行の最小サイズのために無効になる場合 (たとえば行長が 3 000 バイト以上の場合に *x* 値が 50 になっている場合など)、その行は新しいページに入れられます。値として 100 を指定すると、各行がそれぞれ新しいページに入れられます。 **注:** 1 ページあたりの空きスペースの大きさは表の PCTFREE 値によって決まります。ロード操作の `pagefreespace` 値、あるいは表の PCTFREE 値が設定されていない場合、ユーティリティーは各ページのスペースを限界まで使用します。 `pagefreespace` に設定する値は、表に指定されている PCTFREE 値に優先します。
totalfreespace=*x*	*x* は 0〜100 の整数です。この値は、末尾に空きスペースとして追加するページ数の、表の合計ページ数に対する割合 (パーセント) として解釈されます。たとえば *x* が 20 の場合、表のデータ・ページが 100 だとすると、空のページが 20 ページ分追加されます。表のデータ・ページの合計数は 120 になります。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
usedefaults	ターゲット表の列に対するソース列を指定したが、1 つ以上の行インスタンスにデータが入っていない場合に、デフォルト値がロードされます。欠落データの例を次に示します。 • DEL ファイルの場合: ",," が列に指定されている。 • DEL/ASC/WSF ファイルの場合: 列の数が足りない行がある。または行が元の指定の長さに達していない。 このオプションを指定しない場合に、ある行インスタンスのデータがソース列に入っていないと、以下のいずれかになります。 • 列がヌル可能なら、ヌルがロードされます。 • 列がヌル可能でないなら、ユーティリティーはその行を拒否します。
ASCII ファイル形式 (ASC/DEL)	
codepage=*x*	*x* は ASCII 文字ストリングです。この値は、入力データ・セット内のデータのコード・ページとして解釈されます。ロード操作時には、文字データ (および文字で指定された数値データ) を、このコード・ページからデータベースのコード・ページに変換します。 次の規則が適用されます。 • DBCS のみ (グラフィック)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00〜x3F に制限されます。 • EBCDIC コード・ページで指定された DEL データの場合、区切り文字は DBCS のシフトイン文字およびシフトアウト文字と同じであってはなりません。 • `nullindchar` には、標準 ASCII セットのうちコード・ポイント x20〜x7F の範囲内の記号を指定しなければなりません。これは ASCII 記号とコード・ポイントを参照します。 EBCDIC データでは、コード・ポイントが異なる場合でもそれに対応する記号を使うことができます。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
dateformat="*x*"	*x* はソース・ファイル内の日付の形式です。 [a] 有効な日付要素は以下のとおりです。 YYYY - 年 (0000 ～ 9999 の 4 桁) M - 月 (1 ～ 12 の 1 桁または 2 桁) MM - 月 (1 ～ 12 の 2 桁; M と相互に排他) D - 日 (1 ～ 31 の 1 桁または 2 桁) DD - 日 (1 ～ 31 の 2 桁; D と相互に排他) DDD - 元日から数えた日数 (001 ～ 366 の 3 桁; 他の日または月要素と相互に排他) 指定されていない要素には、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。 "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

dumpfile = *x*	*x* は、拒否された行が書き込まれる例外ファイルの (サーバー・ノードでの) 完全修飾名です。 1 レコードにつき最大 32KB のデータが書き込まれます。ダンプ・ファイルの指定方法の例を以下に示します。
	`db2 load from data of del`
	`modified by dumpfile = /u/user/filename`
	`insert into table_name`
	注:
	1. 区分データベース環境の場合は、並列実行している複数のロード操作が同一のファイルに書き込むことがないようにするため、ロードしているノード側でローカルなパスでなければなりません。
	2. ファイルの内容は、非同期バッファー・モードでディスクに書き込まれます。ロード操作が失敗した場合または中断された場合は、ディスクにコミットされたレコードの数を正確に把握することはできず、LOAD RESTART の後は一貫性が保証されません。このファイルが完了しているとみなすことができるのは、単一のパスで開始して完了するロード操作の場合だけです。
	3. この修飾子では、複数のファイル拡張子が付いているファイル名はサポートされません。たとえば、
	`dumpfile = /home/svtdbm6/DUMP.FILE`
	しかし、次のファイル名は使えません。
	`dumpfile = /home/svtdbm6/DUMP.LOAD.FILE`

implieddecimal	暗黙の小数点の位置は列定義によって決められます。つまり、小数点が値の最後にあるとは想定しません。たとえば、`12345` という値が DECIMAL(8,2) 列にロードされると、その値は `123.45` になります。`12345.00` ではありません。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
timeformat="*x*"	*x* はソース・ファイル内の時刻の形式です。 [a] 有効な時刻要素は以下のとおりです。 H - 時 (12 時間系では 0 ～ 12 の 1 桁または 2 桁 24 時間系では 0 ～ 24 の 1 桁または 2 桁) HH - 時 (12 時間系では 0 ～ 12 の 2 桁。 24 時間系では 0 ～ 24 の 2 桁; H と相互に排他) M - 分 (0 ～ 59 の 1 桁または 2 桁) MM - 分 (0 ～ 59 の 2 桁; M と相互に排他) S - 秒 (0 ～ 59 の 1 桁または 2 桁) SS - 秒 (0 ～ 59 の 2 桁; S と相互に排他) SSSSS - 夜の 12 時から数えた秒数 (00000 ～ 86399 の 5 桁; 他の時刻要素と相互に排他) TT - 正午の標識 (AM または PM) 指定されていない要素には、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

timestampformat="*x*"	*x* はソース・ファイル内のタイム・スタンプの形式です。 [a] 有効なタイム・スタンプ要素は以下のとおりです。
	
	YYYY - 年 (0000 ～ 9999 の 4 桁)
	M - 月 (1 ～ 12 の 1 桁または 2 桁)
	MM - 月 (1 ～ 12 の 2 桁; M と相互に排他)
	D - 日 (1 ～ 31 の 1 桁または 2 桁)
	DD - 日 (1 ～ 31 の 2 桁; D と相互に排他)
	DDD - 元日から数えた日数 (001 ～ 366 の 3 桁; 他の月日要素と相互に排他)
	H - 時 (12 時間系では 0 ～ 12 の 1 桁または 2 桁 24 時間系では 0 ～ 24 の 1 桁または 2 桁)
	HH - 時 (12 時間系では 0 ～ 12 の 2 桁。24 時間系では 0 ～ 24 の 2 桁; H と相互に排他)
	M - 分 (0 ～ 59 の 1 桁または 2 桁)
	MM - 分 (0 ～ 59 の 2 桁; M と相互に排他)
	S - 秒 (0 ～ 59 の 1 桁または 2 桁)
	SS - 秒 (0 ～ 59 の 2 桁; S と相互に排他)
	SSSSS - 夜の 12 時から数えた秒数 (00000 ～ 86399 の 5 桁; 他の時刻要素と相互に排他)
	UUUUUU - マイクロ秒 (000000 ～ 999999 の 6 桁)
	TT - 正午の標識 (AM または PM)
	
	YYYY、M、MM、D、DD、または DDD 要素に値が指定されていない場合、デフォルト値の 1 が割り当てられます。それ以外の要素に値が指定されていない場合、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	以下の例では、ユーザー定義の日付および時刻の形式を含んでいるデータを、 schedule という表にインポートする方法を示しています。
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

noeofchar	オプションのファイル終わり (EOF) 文字 x'1A' があっても、ファイルの終わりとは認識しません。通常の文字の場合と同じように処理が続行されます。
	ASC (区切りなし ASCII) ファイル形式

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
binarynumerics	数値データ (DECIMAL 以外) は文字として表示せず、常にバイナリー形式でなければなりません。これにより、コストのかかる変換をしなくてすみます。 このオプションは、reclen オプションで指定した固定長レコードを使用する定位置 ASC の場合にのみサポートされています。また、noeofchar オプションが前提です。 次の規則が適用されます。 • BIGINT、INTEGER、および SMALLINT を除いて、データ・タイプ間の変換は実行されません。 • データ長はそのターゲットとなる列定義と一致していなければなりません。 • FLOAT の形式は IEEE 浮動小数点形式でなければなりません。 • ロードのソース・ファイル内のバイナリー・データは、ロード操作を実行しているプラットフォームには関係なくビッグ・エンディアンとみなされます。 **注:** この修飾子の影響を受ける列のデータの中にヌルがあってはなりません。この修飾子が使われている場合、ブランク (通常はヌルとして解釈される) はバイナリー値として解釈されます。
nochecklengths	nochecklengths を指定した場合、ソース・データにターゲット表の列のサイズを上回る列定義があっても、行ごとにロードが試行されます。そのような行も、コード・ページ変換によってソース・データが短縮されるなら正常にロードできます。たとえば、ソースでは 4 バイトの EUC データがターゲットでは 2 バイトの DBCS データに短縮されるなら、必要なスペースは半分になります。このオプションが特に有用なのは、列定義の不一致にもかかわらず、ソース・データがすべての場合に適合することがわかっている場合です。
nullindchar=*x*	*x* は単一の文字です。ヌル値を示す文字を *x* に変更します。 *x* のデフォルト値は Y です。[b] この修飾子は EBCDIC データ・ファイルでは大文字小文字が区別されません。たとえば、ヌル標識文字を文字 N に指定した場合、n もヌル標識として認識されます。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

packeddecimal	binarynumerics 修飾子には DECIMAL フィールド・タイプが含まれていないため、これによってパック 10 進データが直接ロードされます。
	
	このオプションは、reclen オプションで指定した固定長レコードを使用する定位置 ASC の場合にのみサポートされています。また、noeofchar オプションが前提です。
	
	サポートされる符号ニブルの値は以下のとおりです。
	
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	注: この修飾子の影響を受ける列のデータの中にヌルがあってはなりません。この修飾子が使われている場合、ブランク (通常はヌルとして解釈される) はバイナリー値として解釈されます。
	
	サーバーのプラットフォームには関係なく、ロードのソース・ファイル中のバイナリー・データのバイト順序はビッグ・エンディアンとみなされます。つまり、この修飾子を OS/2 や Windows などのオペレーティング・システム上で使用する場合に、バイト順序が逆であってはなりません。

reclen=*x*	*x* は 32 767 以下の整数です。各行ごとに *x* 文字が読み取られ、行の終わりを示すのに改行文字は使用されません。
striptblanks	可変長フィールドにデータをロードする場合に、後書きブランク・スペースをすべて切り捨てます。このオプションを指定しない場合、ブランク・スペースは保持されます。 このオプションは striptnulls と一緒に指定することはできません。これらは互いに排他的なオプションです。 **注:** このオプションは以前の t オプションにとって代わるものです。t オプションは後方互換性だけのためにサポートされています。
striptnulls	可変長フィールドにデータをロードする場合に、後書きヌル (文字 0x00) をすべて切り捨てます。このオプションを指定しない場合、ヌルは保持されます。 このオプションは striptblanks と一緒に指定することはできません。これらは互いに排他的なオプションです。 **注:** このオプションは以前の padwithzero オプションにとって代わるものです。padwithzero オプションは後方互換性だけのためにサポートされています。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

zoneddecimal	BINARYNUMERICS 修飾子には DECIMAL フィールド・タイプが含まれていないため、これによってパック 10 進データがロードされます。このオプションは、RECLEN オプションで指定した固定長レコードを使用する定位置 ASC の場合にのみサポートされています。また、NOEOFCHAR オプションが前提です。
	
	ハーフバイト符号値は次のいずれかです。
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	10 進数としてサポートされている値は `0x0` ～ `0x9` です。
	
	ゾーンとしてサポートされている値は `0x3` および `0xF` です。

	DEL (区切り付き ASCII) ファイル形式
chardel*x*	*x* は単一文字のストリング区切り文字です。デフォルト値は二重引用符 (") です。ここに指定する文字は、文字ストリングを囲むために二重引用符の代わりに使用されます。[bc] 文字ストリング区切り文字として、以下のように単一引用符 (') を指定することもできます。 `modified by chardel''`
coldel*x*	*x* は単一文字の列区切り文字です。デフォルト値はコンマ (,) です。ここに指定する文字は、列の終わりを示すためにコンマの代わりに使用されます。[bc]
datesiso	日付形式。すべての日付データ値を ISO 形式でロードします。
decplusblank	正符号文字。正の 10 進数値の前に、正符号 (+) の代わりにブランク・スペースを付けます。デフォルト・アクションでは、正の 10 進数値の前に正符号が付けられます。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使う単一文字です。デフォルト値はピリオド (.) です。 ここに指定する文字は、小数点文字号としてピリオドの代わりに使用されます。[bc]

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

delprioritychar	区切り文字の現在のデフォルト優先順位は、レコード区切り文字、文字区切り文字、列の区切り文字、という順序です。この修飾子は、古い優先順位に依存する既存のアプリケーションを、区切り文字の優先順位を逆転させることによって保護します。つまり、文字区切り文字、レコード区切り文字、列の区切り文字の順とします。
	構文:
	`db2 load ... modified by delprioritychar ...`
	たとえば、次のような DEL データ・ファイルがあるとします。
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	`delprioritychar` 修飾子を指定すると、このデータ・ファイルの行は 2 つだけになります。第 2 の <row delimiter> は第 2 の行の最初のデータ列の一部として解釈されますが、第 1 と第 3 の <row delimiter> は実際のレコード区切り文字として解釈されます。この修飾子を指定*しない* 場合、このデータ・ファイルは、それぞれが <row delimiter> で区切られる行が 3 行で構成されることになります。

dldel*x*	*x* は単一文字の DATALINK 区切り文字です。デフォルト値はセミコロン (;) です。ここに指定する文字は、DATALINK 値のフィールド間区切り文字としてセミコロンの代わりに使用されます。 DATALINK 値の副値が複数になることがあるので、これが必要になります。[bcd] **注:** *x* を、行、列、または文字ストリングの区切り文字として指定された文字と同じにすることはできません。

keepblanks	タイプが CHAR、VARCHAR、LONG VARCHAR、または CLOB のフィールド内の前後のブランクを保存します。このオプションが指定されていない場合、文字区切り文字の外側にある前後のブランクはすべて除去され、ブランクになっているすべてのフィールドについては表にヌルが挿入されます。
	以下の例では、データ・ファイル内の前後のスペースをすべて保持しながら、 TABLE1 という表にデータをロードする方法を示しています。
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nodoubledel	二重文字区切り文字の認識を抑止します。詳しくは、19ページの『区切り文字に関する制限』を参照してください。
IXF ファイル形式	

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明
forcein	コード・ページの不一致があってもデータを受け入れて、コード・ページ間の変換を抑止するよう、ユーティリティーに指示します。 固定長のターゲット・フィールドの長さが、データを入れるのに十分かどうか検査されます。 `nochecklengths` を指定しない場合、検査なしで各行のロードが試みられます。
nochecklengths	`nochecklengths` を指定した場合、ソース・データにターゲット表の列のサイズを上回る列定義があっても、行ごとにロードが試行されます。そのような行も、コード・ページ変換によってソース・データが短縮されるなら正常にロードできます。たとえば、ソースでは 4 バイトの EUC データがターゲットでは 2 バイトの DBCS データに短縮されるなら、必要なスペースは半分になります。このオプションが特に有用なのは、列定義の不一致にもかかわらず、ソース・データがすべての場合に適合することがわかっている場合です。

表 8. 有効なファイル・タイプ修飾子 (ロード) (続き)

修飾子	説明

注:
1. サポートされないファイル・タイプを MODIFIED BY オプションで使用しようとした場合、ロード・ユーティリティーは警告を出しません。それを試みた場合、ロード操作は失敗してエラー・コードが戻されます。
2. [a] 日付形式ストリングは必ず二重引用符で囲まなければなりません。フィールド区切り文字には、 a 〜 z、A 〜 Z、および 0 〜 9 を含めることはできません。フィールド区切り文字は、 DEL ファイル形式の文字区切り文字またはフィールド区切り文字と同じにすることはできません。要素の開始および終了位置が明らかな場合、フィールド区切り文字は任意指定です。開始および終了位置が明らかでないのは、項目が長さが一定でない D、H、M、または S などの要素が使用されている場合です (修飾の仕方によって異なります)。
タイム・スタンプ形式の場合、月の記述子と分の記述子のどちらも文字 M を使用するため、区別が明白であるように注意しなければなりません。月のフィールドは他の日付フィールドに隣接していなければなりません。分のフィールドは他の時刻フィールドに隣接していなければなりません。以下のタイム・スタンプ形式は、あいまいな形式の例です。
`"M" (月と分のどちらかがはっきりしない)`
`"M:M" (どちらが何を表しているのか分からない)`
`"M:YYYY:M" (どちらも月として解釈される)`
`"S:M:YYYY" (時刻値と日付値の両方に隣接している)`
あいまいな形式になっている場合、ユーティリティーをエラー・メッセージを報告し、操作は失敗します。
以下のタイム・スタンプ形式では、何を指しているのかがはっきりしています。
`"M:YYYY" (月)`
`"S:M" (分)`
`"M:YYYY:S:M" (月....分)`
`"M:H:YYYY:M:D" (分....月)`
注: 二重引用符や円記号などの文字の前には、拡張文字 (たとえば、¥) を付けなければなりません。
3. [b] 文字はソース・データのコード・ページで指定する必要があります。
文字シンボルの代わりに文字コード・ポイントを指定できます。その場合、構文として xJJ または 0xJJ を使用します (JJ はコード・ポイントの 16 進表記)。たとえば、# 文字を列の区切り文字として指定するには、以下のいずれかを使用します。
`... modified by coldel# ...`
`... modified by coldel0x23 ...`
`... modified by coldelX23 ...`
4. [c] 19ページの『区切り文字に関する制限』に、区切り文字の指定変更として使用できる文字に適用される制限のリストが示されています。
5. [d] DATALINK 区切り文字が URL 構文では有効な文字であっても、ロード操作の範囲ではその特別な意味を失います。	

例外表

例外表はロード中の表の定義を反映するユーザー作成の表であり、追加の列がいくつか含まれています。この表は LOAD コマンドの FOR EXCEPTION 文節で指定します。例外表には、識別列も、他のどのタイプの生成列も入れることはできません。 1 次表内に識別列があると、例外表内のそれに対応する列には、その列のタイプ、長さ、およびヌル可能性の属性しか入れることはできません。例外表は固有索引の規則に違反している行のコピーを格納するのに使われます。ユーティリティーは固有性の違反を除いて制約または外部キーの違反の検査をしません。さらに、DATALINK 例外も例外表に取り込まれます。

固有キーとは、等しい 2 つの値が存在しないキーのことです。この制約を実現するのに使われる仕組みは、固有索引と呼ばれます。基本キーは固有キーの特殊な例です。同一の表に 2 つ以上の基本キーを設定することはできません。

注: 無効なデータが原因で索引の構築前に拒否された行は、例外表に挿入されません。

行は例外表の中の既存の情報に追加されます。それには以前のロード操作で無効だった行も含まれることがあります。現在のロード操作で無効な行だけを含めたい場合には、ユーティリティーを呼び出す前に既存の行を削除しておく必要があります。

ロード・ユーティリティーで使用される例外表は、SET INTEGRITY ステートメントが使用する例外表と同一のものです。

ロードするデータに固有索引があり、重複レコードが存在する可能性がある場合は、例外表を使用してください。例外表を指定していない場合に重複レコードが検出されると、ロード操作はそのまま続行してしまい、削除された重複レコードに関する警告メッセージだけが出されます。この場合、レコード自体はログに記録されません。

ロード操作の完了後、エラーになったデータを例外表の中にある情報を使って訂正することができます。その後、訂正したデータを表に挿入することができます。

例外表については、*SQL 解説書* を参照してください。

ダンプ・ファイル

dumpfile 修飾子を指定すると、拒否された行を書き込む例外表の名前と位置をロード・ユーティリティーに対して指示できます。区分データベース環境で実行している場合、例外が生成された区分番号を示す拡張子が名前に付けられます。たとえば、

```
dumpfile = "/u/usrname/dumpit"
```

この例では、区画 0 に /u/usrname/dumpit.000 という名前のファイルが生成されます。区画 5 には /u/usrname/dumpit.005 という名前のファイルが生成され、以下同様になります。

レコードが 32 768 バイト未満の長さの場合、そのレコード全体がダンプ・ファイルにコピーされますが、それ以上の長さの場合は、レコード断片 (レコードの最終バイトを含む) がファイルに書き込まれます。

ロード・ファイル・タイプ修飾子については、133ページの『ファイル・タイプ修飾子 (ロード)』を参照してください。

ロード一時ファイル

DB2 は、ロード処理中にバイナリーの一時ファイルを作成します。このファイルは、ロード破損回復、ロード終了操作、警告およびエラー・メッセージ、および実行時制御データに使われます。これらの一時ファイルは、ロード操作がエラーなしで完了した時点で削除されます。

一時ファイルが書き込まれるパスは、 LOAD コマンドの *temp-pathname* パラメーターまたは **sqluload** API の *pRemoteMsgFileName* パラメーターで指定できます。デフォルトのパスは、データベース・ディレクトリーのサブディレクトリーです。

一時ファイルのパスはサーバー・マシン上にあり、DB2 のインスタンスが排他的にアクセスします。そのため、*temp-pathname* パラメーターに指定するパス名の修飾はクライアントではなくサーバーのディレクトリー構造を反映したものでなければならず、 DB2 インスタンス所有者にはそのパスに対して読み取りと書き込みの両方の許可が必要です。

注: MPP システムにおいては、一時ファイルのパスは NFS マウント上ではなく、ローカル・ディスク上のパスにしてください。パスが NFS マウント上にあると、ロード操作中のパフォーマンスがかなり低下します。

考慮事項: このパスに書き込まれる一時ファイルは、どんな状況にあっても決して手を加えないでください。一時ファイルに手を加えるとロード操作における誤動作の原因となり、データベースが危険な状態になります。

ロード・ユーティリティーのログ・レコード

ユーティリティー管理機能は、ログ・ユーティリティーを含むいくつかの DB2 ユーティリティーに関連するログ・レコードを生成します。ログ・レコードには、特定の活動の開始点または終了点が記録されます。ロード操作には、以下に示すログ・レコードが関連しています。

- ロード開始。このログ・レコードはロード操作の開始に関連したものです。
- 表ロード削除開始。このログ・レコードはロード操作の削除フェーズの開始に関連したものです。削除フェーズは、基本キー値が重複している場合にのみ開始されます。削除フェーズでは、表レコードに対する各削除操作または索引キーが記録されます。
- ロード削除開始補正。このログ・レコードはロード操作の削除フェーズの終了に関連したものです。
- ロード保留リスト。このログ・レコードは、ロード・トランザクションがコミットした時点で書き込まれます。保留リストは、トランザクションがコミットするまで延期される回復不能操作のリンク・リストです。このトランザクションの後にはログ・レコードのコミットは実行されません。
- (回復不能ロード操作中に) 表スペースのエクステントが割り振りまたは削除されるたびに、ログ・レコードが書き込まれます。

これらのログ・レコードの構造については、*管理 API 解説書* を参照してください。

文字セットと各国語のサポート

DB2 UDB データ移動ユーティリティーには、次のような各国語サポート (NLS) が備わっています。

- インポートおよびエクスポート・ユーティリティーには、クライアント・コード・ページからサーバー・コード・ページへの自動コード・ページ変換が備わっています。
- ロードおよびオートローダー・ユーティリティーでは、 `codepage` 修飾子とともに DEL および ASC ファイルを指定して、任意のコード・ページからサーバー・コード・ページにデータを変換することができます。

- どのユーティリティーでも、IXF データは、元のコード・ページ (IXF ファイルに保管されているもの) からサーバーのコード・ページに自動的に変換されます。

場合によって、コード・ページが異なるために文字データの長さに変化が生じることがあります。たとえば、日本語または中国語 (繁体字) の拡張 UNIX コード (EUC) と 2 バイト文字セット (DBCS) では、同じ文字が別々の長さで符号化されていることがあります。普通、入力データの長さとターゲット列の長さの比較は、まだどのデータも読み取らないうちに実行されます。入力の長さがターゲットの長さを超えている場合、列がヌル可能であればヌルがその列に挿入されます。そうでない場合、要求は拒否されます。 `nochecklengths` 修飾子 (133ページの『ファイル・タイプ修飾子 (ロード)』を参照) を指定した場合は、初期の比較をしないでデータをロードしようとします。変換完了後にデータが長すぎることが明らかになったなら、その行は拒否されます。そうでなければ、データはロードされます。

ロード・セッションの例

CLP の例

例 1

TABLE1 には以下に示す 5 つの列があります。

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 には以下に示す 6 つの要素があります。

- ELE1、位置 01～20
- ELE2、位置 21～22
- ELE5、位置 23～23
- ELE3、位置 24～27
- ELE4、位置 28～31
- ELE6、位置 32～32
- ELE6、位置 33～40

データ・レコード:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

以下に示すコマンドで、ファイルから表をロードします。

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
   method L (1 20, 21 22, 24 27, 28 31)
   null indicators (0,0,23,32)
   insert into table1 (col1, col5, col2, col3)
```

注:

1. MODIFIED BY パラメーターに striptblanks を指定することにより、VARCHAR 列でブランクの切り捨てを強制的に実行します (たとえば行 1、2、3 の COL1 はそれぞれ長さ 11、17、19 バイトになります)。
2. MODIFIED BY パラメーターに reclen=40 を指定することにより、各入力レコードの末尾には改行文字がなく、各レコードの長さが 40 バイトであることを示しています。最後の 8 バイトは表のロードに使われません。
3. COL4 は入力ファイルにないため、TABLE1 にはそのデフォルト値 (NOT NULL WITH DEFAULT で定義) で挿入されます。
4. 位置 23 および 32 は、TABLE1 の COL2 と COL3 にヌルをロードするかどうかを行ごとに示すのに使用されます。特定のレコードのうち列のヌル標識位置が Y なら、その列はヌルになります。それが N の場合は、入力レコードのその列のデータ位置 (L(.....) で定義) にあるデータ値が、その行の列データのソースとして使われます。この例の場合、行 1 ではどちらの列もヌルではなく、行 2 では COL2 がヌル、行 3 では COL3 がヌルです。
5. この例では COL1 と COL5 の NULL INDICATORS が 0 (ゼロ) として指定されており、そのデータがヌル可能でないことを示しています。
6. 特定の列に対する NULL INDICATOR は入力レコードのどの位置でも可能ですが、その位置は必ず指定しなければならず、Y または N のいずれかの値が提供される必要があります。

例 2 (ファイルから LOB をロードする)

TABLE1 には以下に示す 3 つの列があります。

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

ASCFILE1 には以下に示す 3 つの要素があります。

- ELE1、位置 01～04
- ELE2、位置 06～13
- ELE3、位置 15～22

以下に示すファイルが /u/user1 または /u/user1/bin のいずれかにあります。

- ASCFILE2、LOB データを含む
- ASCFILE3、LOB データを含む
- ASCFILE4、LOB データを含む
- ASCFILE5、LOB データを含む
- ASCFILE6、LOB データを含む
- ASCFILE7、LOB データを含む

ASCFILE1 内のデータ・レコード:

```
1...5....10...15...20...25...30.
REC1 ASCFILE2 ASCFILE3
REC2 ASCFILE4 ASCFILE5
REC3 ASCFILE6 ASCFILE7
```

以下に示すコマンドで、ファイルから表をロードします。

```
db2 load from ascfile1 of asc
   lobs from /u/user1, /u/user1/bin
   modified by lobsinfile reclen=22
   method L (1 4, 6 13, 15 22)
   insert into table1
```

注:

1. MODIFIED BY パラメーターに lobsinfile を指定することによって、すべての LOB データをファイルからロードするようローダーに指示しています。

2. MODIFIED BY パラメーターに reclen=22 を指定することにより、各入力レコードの末尾には改行文字がなく、各レコードの長さが 22 バイトであることを示しています。
3. LOB データは 6 つのファイル (ASCFILE2～ASCFILE7) の中に入っています。それぞれのファイルの中には、特定の行に LOB 列をロードするのに使われるデータが入っています。 LOB と他のデータとの間の関係は ASCFILE1 の中で指定されています。このファイルの最初のレコードは、REC1 を行 1 の COL1 に配置するようローダーに指示しています。ASCFILE2 の内容は行 1 の LOB1 をロードするのに使われ、ASCFILE3 の内容は行 1 の LOB2 をロードするのに使われます。同じように、ASCFILE4 と ASCFILE5 は行 2 の LOB1 と LOB2 をロードするのに使われ、 ASCFILE6 と ASCFILE7 は行 3 の LOB をロードするのに使われます。
4. LOBS FROM パラメーターに含まれている 2 つのパスは、名前の指定された LOB ファイルがローダーで必要になった場合にそれらのファイルを検索するパスです。
5. LOB を ASCFILE1 (区切りなし ASCII ファイル) から直接に、lobsinfile 修飾子を指定せずにロードするには、以下に示す規則に従う必要があります。
   - LOB も含め、どのレコードも合計の長さが 32KB 以下でなければなりません。
   - 入力レコードの中の LOB フィールドは固定長フィールドでなければならず、その LOB データには必要に応じてブランクを埋め込まなくてはなりません。
   - striptblanks 修飾子を指定する必要があります。それによって、LOB をデータベースに挿入する時点で、LOB の埋め込みに使った後書きブランクが除去されるようになります。

例 3 (ダンプ・ファイルの使用)

表 FRIENDS は以下のように定義されています。

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

この表に対して、以下に示すデータ・レコードのロードを試みたとします。

```
23, 24, bobby
, 45, john
4,, mary
```

第 2 行は最初の INT がヌルであり、列定義では NOT NULL が指定されているために拒否されます。 DEL 形式に合致していない初期文字を含んでいる列

ではエラーが生成され、そのレコードは拒否されます。そのようなレコードはダンプ・ファイルに書き込まれます (133ページの表8 を参照)。

文字区切り文字の外側の列の中にある DEL データは無視されますが、警告が生成されます。たとえば、

```
22,34,"bob"
24,55,"sam" sdf
```

ユーティリティーはこの表の第 3 列にある ″sam″ をロードし、文字 ″sdf″ には警告のフラグが付けられます。このレコードは拒否されません。別の例として、

```
22 3, 34,"bob"
```

ユーティリティーは 22,34,"bob" をロードし、列 1 の 22 の後にある一部のデータが無視されたという警告を生成します。このレコードは拒否されません。

例 4 (DATALINK データのロード)

以下に示すコマンドは、DEL 形式のデータを含む入力ファイル delfile1 から表 MOVIETABLE をロードします。

```
db2 load from delfile1 of del
    modified by dldel|
    insert into movietable (actorname, description, url_making_of, url_movie)
    datalink specification (dl_url_default_prefix "http://narang"),
    (dl_url_replace_prefix "http://bomdel" dl_url_suffix ".mpeg")
    for exception excptab
```

注:

1. この表には 4 つの列があります。

   ```
   actorname              VARCHAR(n)
   description            VARCHAR(m)
   url_making_of          DATALINK (with LINKTYPE URL)
   url_movie              DATALINK (with LINKTYPE URL)
   ```

2. 入力ファイル内の DATALINK データでは、サブフィールド区切り文字として垂直バー (|) 文字が使われています。
3. url_making_of の列値のいずれかに接頭部文字列がない場合は ″http://narang″ が使用されます。
4. url_movie のヌルでない列値には、それぞれ接頭部として ″http://bomdel″ が付けられます。既存の値は置き換えられます。
5. url_movie のヌルでない列値では、それぞれ ″.mpeg″ がパスに付加されます。たとえば、url_movie の列値が ″http://server1/x/y/z″ であれば

″http://bomdel/x/y/z.mpeg″ として格納されることになります。また、値が ″/x/y/z″ であれば ″http://bomdel/x/y/z.mpeg″ として格納されることになります。

6. 表のロード中に固有索引または DATALINK の例外が発生すると、影響を受けるレコードは表から削除されて例外表 excptab に入れられます。

例 5 (識別列がある表のロード)

TABLE1 には、以下の 4 つの列があります。

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 は、C2 が GENERATED ALWAYS 識別列であることを除き、TABLE1 と同じです。

DATAFILE1 内のデータ・レコード (DEL 形式):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 内のデータ・レコード (DEL 形式):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

注:

1. 以下のコマンドでは、行 1 および 2 の識別値が生成されます。これは、DATAFILE1 内にこれらの行の識別値が存在しないためです。ただし、行 3 および 4 には、ユーザー提供の識別値である 100 および 101 がそれぞれ割り当てられます。

   ```
   db2 load from datafile1.del of del replace into table1
   ```

2. DATAFILE1 を TABLE1 にロードして、すべての行について識別値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

   ```
   db2 load from datafile1.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
   db2load from datafile1.del of del modified by identityignore replace into table1
   ```

3. DATAFILE2 を TABLE1 にロードして、それぞれの行について識別値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
db2 load from datafile2.del of del modified by identitymissing replace into table1
```

4. DATAFILE1 を TABLE2 にロードして、識別値である 100 および 101 が行 3 および 4 にそれぞれ割り当てられるようにするには、以下のコマンドを発行してください。

```
db2 load from datafile1.del of del modified by identityoverride replace into table2
```

この場合、行 1 および 2 は拒否されます。これは、ユーティリティーへの指示により、ユーザー提供の値を優先し、システムが生成した識別値をオーバーライドするようになっているためです。ただし、ユーザー提供の値が指定されていない場合、識別列は暗黙的にヌルでないことになるので、行は拒否されます。

5. 識別に関係するファイル・タイプ修飾子を使用せずに DATAFILE1 を TABLE2 にロードすると、行 1 と 2 はロードされますが、行 3 と 4 は拒否されます。これは、行 3 と 4 では独自に非ヌル値が提供されており、識別列が GENERATED ALWAYS であるためです。

API の例

下記のサンプル・プログラムは、次のことを実行する方法を示しています。

- SAMPLE データベースの中の EMP_RESUME 表からファイル EXPTABLE.DEL へ情報をエクスポートします。
- その情報を、区切り付きテキスト・ファイルから新しい表 LOADTABLE へロードします。

SAMPLE データベースについては、*管理の手引き* を参照してください。

このサンプル・プログラムのソース・ファイル (tload.sqc) は、 Windows オペレーティング・システムと OS/2 では ¥sqllib¥samples¥c ディレクトリーにあり、 UNIX ベースのシステムでは sqllib/samples/c にあります。そこには、DB2 API と組み込み SQL 呼び出しの両方が入っています。同じディレクトリーにあるスクリプト・ファイル bldapp には、このサンプル・プログラムや他のサンプル・プログラムを構築するためのコマンドが含まれています。DB2 管理用 API を含むアプリケーションの作成についての一般情報、およびコンパイルとリンクのオプションについては、*アプリケーション構築の手引き* を参照してください。ソース・ファイル tload.sqc からサンプル・プログラム tload を構築するには、以下のようにします。

1. ファイル tload.sqc、bldapp*、utilemb.c、および utilemb.h を作業ディレクトリーにコピーします。

2. データベース・マネージャーが実行されていないなら、コマンド db2start を発行します。
3. bldapp tload sample と入力します。以下のファイルが生成されます。

```
tload.bnd
tload.c
util.obj/util.o
tload.obj/tload.o
tload.exe/tload
```

サンプル・プログラム (実行可能ファイル) を実行するには、tload と入力します。どんなファイルが生成されるかを調べてみてください。たとえば、メッセージ・ファイルや区切り付き ASCII データ・ファイルが生成されます。

```
/******************************************************************************
**
** Source File Name = tload.sqc  1.4
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 1997
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    PURPOSE :
**      To show the use of the QUIESCE TABLESPACE and the LOAD APIs.
**        - EXPORT the EMP_RESUME table into a comma delimited file.
**        - create a temporary table ('loadtable').
**        - QUIESCE the TABLESPACES, preparing the temporary table to be
**          LOADable.
**        - LOAD the comma delimited file into a temporary table ('loadtable').
**
**    STRUCTURES USED :
**       sqldcol
**       sqlchar
**       sqluexpt_out
**       sqlca
**
**    APIs USED :
**       EXPORT                            sqluexpr
**       QUIESCE TABLESPACE FOR TABLES     sqluvqdp
**       LOAD                              sqluload
**
**    FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**          string.h -  fgets, strncpy
**
**       DBMS LIBRARY :
**          sqlenv.h -  see "APIs USED" above
**
**       OTHER :
**          external :
**             check_error :     Checks for SQLCODE error, and prints out any
**             [in UTIL.C]       related information available.
**
**    EXTERNAL DEPENDANCIES :
**       - Ensure existence of database (SAMPLE) for precompile purposes.
**       - Precompile with the SQL precompiler (PREP in DB2)
**       - Bind to a database (BIND in DB2)
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
```

```
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <malloc.h>
#include "utilemb.h"
#ifdef DB2AIX
#define  WORKDIR  "/u/workdir"
#else
#define WORKDIR "."
#endif
EXEC SQL INCLUDE SQLCA;
int main (int argc, char *argv[]) {
   short int            callerAction = 0;
   struct sqldcol       DataDescriptor;
   struct sqlchar       *ActionString;
   struct sqlchar       *FileTypeMod;
   struct sqluexpt_out  outputInfo;
   char        datafile[] = "EXPTABLE.DEL";
   char        statement[] = "SELECT empno, photo_format, picture FROM emp_photo";
   char        impStatement[] = "INSERT INTO loadtable (num, format, photo)";
/*
   char        statement[] = "SELECT empno, photo_format FROM emp_photo";
   char        impStatement[] = "INSERT INTO loadtable (num, format)";
*/
   char        msgfile_x[] = "EXPMSG.TXT";
   char        FileType[] = SQL_DEL;
   char        table_name[18];
   /* Variables for the LOAD API */
   struct sqlu_media_list       DataFileList;
   struct sqlu_media_list       *pLobPathList;
   struct sqluload_in           InputInfo;
   struct sqluload_out          OutputInfo;
   struct sqlu_media_list       *pWorkDirectoryList = NULL;
   struct sqlu_media_list       *pCopyTargetList;
   char                         LocalMsgFileName[] = "LOADMSG";
   char                         RemoteMsgFileName[] = "RLOADMSG";
   short                        CallerAction;
   sqlint32                     *pNullIndicators;
   void                         *pReserved;
   EXEC SQL BEGIN DECLARE SECTION;
      char userid[9];
      char passwd[19];
   EXEC SQL END DECLARE SECTION;
   printf ("This is sample program 'tload.sqc'¥n");
   /* Initialize structures. */
   memset(&DataFileList, 0, sizeof(struct sqlu_media_list));
   memset(&InputInfo, 0, sizeof(struct sqluload_in));
   memset(&OutputInfo, 0, sizeof(struct sqluload_out));
   /* need to preset the size of structure field and counts */
   outputInfo.sizeOfStruct = SQLUEXPT_OUT_SIZE;
   /*****************************************************************¥
   * need to allocate the proper amount of space for the SQL statement *
   ¥*****************************************************************/
   ActionString = (struct sqlchar *)malloc(strlen(statement)
      + sizeof (struct sqlchar));
   ActionString->length = strlen(statement);
   strncpy (ActionString->data, statement, strlen(statement));
   FileTypeMod = (struct sqlchar *)malloc(strlen("lobsinfile")
      + sizeof (struct sqlchar));
   FileTypeMod->length = strlen("lobsinfile");
   strncpy (FileTypeMod->data, "lobsinfile", FileTypeMod->length);
   /* DELimited format can not have specified names, therefore the
      column method is 'D'efault */
   DataDescriptor.dcolmeth = SQL_METH_D;
```

```
if (argc == 1) {
   EXEC SQL CONNECT TO sample;
EMB_SQL_CHECK("CONNECT TO SAMPLE");
}
else if (argc == 3) {
   strcpy (userid, argv[1]);
   strcpy (passwd, argv[2]);
   EXEC SQL CONNECT TO sample USER :userid USING :passwd;
   EMB_SQL_CHECK("CONNECT TO SAMPLE");
}
else {
   printf ("¥nUSAGE: tload [userid passwd]¥n¥n");
   return 1;
} /* endif */

printf ("Exporting EMP_RESUME table into file '%s'¥n", datafile);
/*******************¥
* EXPORT API called *
¥*******************/
sqluexpr (datafile, NULL, NULL, &DataDescriptor, ActionString,
   FileType, FileTypeMod, msgfile_x, 0, &outputInfo, NULL, &sqlca);
EMB_SQL_CHECK("exporting table");
printf ("Rows exported %d¥n", outputInfo.rowsExported);
free (ActionString);
/* need to allocate the proper amount of space for the SQL statement */
ActionString = (struct sqlchar *)malloc(strlen(impStatement)
   + sizeof (struct sqlchar));
ActionString->length = strlen(impStatement);
strncpy (ActionString->data, impStatement, strlen(impStatement));
printf ("Creating a temporary table 'loadtable' to load into¥n");
/* create a temporary table to import into */
EXEC SQL CREATE TABLE loadtable (num CHARACTER(6), format VARCHAR(10),
   photo BLOB(100K));
EMB_SQL_CHECK("CREATE TABLE");

/* end the transaction so the program can quiesce the tablespace */
EXEC SQL COMMIT;
printf ("Quiescing tablespaces for table 'loadtable'¥n");
/*******************************¥
* QUIESCE TABLESPACE FOR TABLE *
¥*******************************/
sqluvqdp ("loadtable", SQLU_QUIESCEMODE_EXCLUSIVE, NULL, &sqlca);
EMB_SQL_CHECK("QUIESCE TABLESPACES FOR TABLE");
printf ("Loading the file '%s' into 'loadtable'¥n", datafile);
/* initializing the variables for the LOAD API */
/* the DataFileList structure */
DataFileList.media_type = SQLU_SERVER_LOCATION;
DataFileList.sessions = 1;
DataFileList.target.location = (sqlu_location_entry *) malloc
   (sizeof(sqlu_location_entry) * DataFileList.sessions);
strcpy (DataFileList.target.location->location_entry, datafile);
pLobPathList = NULL;
CallerAction = SQLU_INITIAL;
/* the sqluload input structure */
InputInfo.sizeOfStruct = SQLULOAD_IN_SIZE;      /* this should never change */
InputInfo.savecnt = 0;                          /* consistency points as */
                                                /*   infrequently as possible */
InputInfo.restartcnt = 0;                       /* start at row 1 */
InputInfo.rowcnt = 0;                           /* load all rows */
InputInfo.warningcnt = 0;                       /* don't stop for warnings */
InputInfo.data_buffer_size = 0;                 /* default data buffer size */
InputInfo.sort_buffer_size = 0;                 /* default warning buffer */
                                                /*   size */
InputInfo.hold_quiesce = 0;                     /* don't hold the quiesce */
InputInfo.restartphase = ' ';                   /* ignored anyway, but */
                                                /*   must be ' ',L,B,D */
InputInfo.statsopt = SQLU_STATS_NONE;           /* don't bother collecting */
InputInfo.indexing_mode = SQLU_INX_AUTOSELECT;  /* let load choose indexing */
                                                /*   mode
/* the sqluload output structure */
```

```
   OutputInfo.sizeOfStruct = SQLULOAD_OUT_SIZE;
   /* the CopyTargetList structure */
      pCopyTargetList = NULL;
   OutputInfo.sizeOfStruct = SQLULOAD_OUT_SIZE;
   /******¥
   * LOAD *
   ¥******/
   sqluload (&DataFileList,
             pLobPathList,
             &DataDescriptor,
             ActionString,
             FileType,
             FileTypeMod,
             LocalMsgFileName,
             RemoteMsgFileName,
             CallerAction,
             &InputInfo,
             &OutputInfo,
             pWorkDirectoryList,
             pCopyTargetList,
             pNullIndicators,
             pReserved,
             &sqlca);
   EMB_SQL_CHECK("LOADing table");
   printf ("Rows loaded %d¥nrows committed %d¥n", OutputInfo.rowsLoaded,
      OutputInfo.rowsCommitted);
   free (ActionString);
   /* drop the table  */
   EXEC SQL DROP TABLE loadtable;

   EXEC SQL CONNECT RESET;
   EMB_SQL_CHECK("CONNECT RESET");
}
/* end of program : tload.sqc */
```

サンプル・プログラムのソース・ファイル (loadqry.sqc) は、¥sqllib¥samples¥c ディレクトリーにあります。このサンプル・プログラムは、プログラムの接続先データベースに対するロード操作の現在の状態を、API を使って照会する方法を示すものです。そこには、DB2 API と組み込み SQL 呼び出しの両方が入っています。同じディレクトリーにあるスクリプト・ファイル bldvaemb.cmd には、このサンプル・プログラムや他のサンプル・プログラムを構築するためのコマンドが含まれています。 DB2 管理用 API を含むアプリケーションの作成についての一般情報、およびコンパイルとリンクのオプションについては、*アプリケーション構築の手引き* を参照してください。OS/2 で、ソース・ファイル loadqry.sqc からサンプル・プログラム loadqry を構築するには、以下のようにします。

1. ファイル loadqry.sqc、bldvaemb.cmd、util.c、および util.h を作業ディレクトリーにコピーします。
2. データベース・マネージャーが実行されていないなら、コマンド db2start を発行します。
3. bldvaemb loadqry sample と入力します。以下のファイルが生成されます。

```
loadqry.bnd
loadqry.c
util.obj
loadqry.obj
loadqry.exe
```

サンプル・プログラム (実行可能ファイル) を実行するには、loadqry と入力します。メッセージ・ファイルを調べると役立つかもしれません。このファイルに情報が入っているのは、ロード操作の進行中にこのプログラムを実行した場合だけです。

ロード操作後の保留状態

定期ログ記録は実行されないため、ロード・ユーティリティーはデータベースの整合性を保つためにいくつかの*保留* 状態を利用します。それらの状態は LIST TABLESPACES コマンドを使って調べることができます (*コマンド解説書* を参照)。

ロード・プロセスのロード・フェーズと構築フェーズでは、関連しているすべての表スペースがロード保留状態になります。ロード操作が失敗したか中断された場合にロード保留状態を除去するには、以下のいずれかを実行します。

- ロード操作を再開します。まず障害の原因に対処します。たとえば、ロード・ユーティリティーがディスク・スペースを使い果たした場合、表スペースにコンテナーを追加してから、ロード再開操作を試みます。
- ロード操作を終了します。
- ロード操作が失敗したその同じ表に対し、LOAD REPLACE 操作を呼び出します。
- ロードする表の表スペースを、最新の表スペースまたはデータベース・バックアップを指定した RESTORE DATABASE コマンドを使って回復した後、それ以降の回復処理を実行します。
- ロードする表の表スペースを除去してから再作成します。

削除フェーズでは、関連するすべての表スペースが削除保留状態にされます。これが起きる可能性があるのは、ロード操作が失敗した場合か、または (固有キー違反を除去する) 削除フェーズで中断された場合です。削除はログに記録されるので、ログ・スペースがなくなると障害が起きることがあります。ロード保留状態を除去するのと同じ処置により、削除保留状態を除去することができます。

ロード・プロセスが完了すると表スペースはバックアップ保留状態になり、さらに、

ロード操作後の保留状態

- データベース構成パラメーター *logretain* が recovery に設定されるか、または *userexit* が有効になります。
- ロード・オプション COPY YES は指定されません。
- ロード・オプション NONRECOVERABLE は指定されません。

ロード・プロセスに関連した第 4 の状態 (検査保留状態) は、参照制約と検査制約、DATALINKS 制約、AST 制約、または生成列制約に関係しています。たとえば、既存の表が親表であり、そこに含まれる基本キーが従属表内の外部キーによって参照されている場合、その親表の中のデータを置き換えるとその従属表 (表スペースではない) は検査保留状態になります。表が検査保留状態のままになっている場合、その表に対して参照保全制約や検査制約についての妥当性検査を実行するには、ロード・プロセスの完了後に SET INTEGRITY ステートメントを発行します。検査保留状態については、90ページの『制約違反の検査』を参照してください。

ロードのパフォーマンスの最適化

ロード・ユーティリティーのパフォーマンスは、データの性質や量、索引の数、そして指定したロード・オプションによって異なります。

固有索引がある場合、重複データが検出されるとロードのパフォーマンスは低下します。それでもほとんどの場合、ロード操作の完了後に索引ごとに CREATE INDEX ステートメントを呼び出すよりは、ロード操作中に索引を作成するほうが効率はよくなります (図5 を参照)。

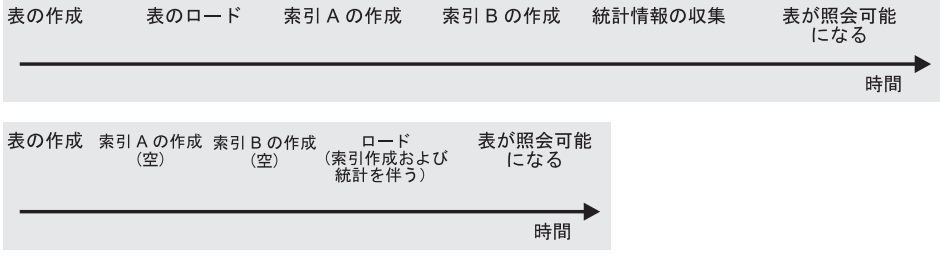


図 5. 索引作成と統計収集を並行して実行することによりロードのパフォーマンスを向上させる. 多くの場合、表はデータのロード、索引の構築、統計の収集という 3 つのステップによって構築されます。それで、ロード操作中、索引の作成中 (表ごとに複数の索引が可能)、および統計の収集中 (表データと索引すべてに対する入出力が発生) に、複数のデータ入出力が発生することになります。別の方法として、ロード・ユーティリティーがデータを一括してこれらの作業を実行するようにすれば、さらに時間が短くなります。

索引作成のパフォーマンスを調整する場合、ロード操作中に索引キーのソート処理に使用されるメモリーの大きさを *sortheap* データベース構成パラメーター

で制御することができます。たとえば、キーのソート処理で索引ごとに 4000 ページの主メモリーを使用するようロード・ユーティリティーに指示するには、 *sortheap* データベース構成パラメーターを 4000 ページに設定し、データベースからすべてのアプリケーションを切り離した後、LOAD コマンドを発行します。索引が大きすぎてメモリー内でソートできないと、ソート・スピルが起こります。つまり、データは、複数の「ソート実行」で分割され、後で組み合わせられる一時表スペースに保管されます。 *sortheap* パラメーターのサイズを大きくしてもソート・スピルを避けられない場合、一時表スペースのバッファー・プールを十分大きくして、スピルが原因で生じるディスク入出力量を最小化することが大切です。さらに、複数のソート実行の組み合わせにおいて入出力の並列性を実現するため、一時表スペースの宣言時に、それぞれが異なるディスク装置に存在する複数のコンテナーを指定することをお勧めします。

ロードのパフォーマンスは、サード・パーティー・ベンダーが提供する高性能なソート・ライブラリーをインストールして、ロード操作中に索引を作成することによって向上させることができます。サード・パーティーによるソート製品の一例として SyncSort があります。実行時にロードするソート・ライブラリーの場所は、**DB2SORT** 環境変数 (レジストリー値) を使って指定します。環境変数については、*管理の手引き* を参照してください。

SET INTEGRITY ステートメントを使うと、表をロードしてからそれをもう一度使用できるようにするのに必要な合計時間が長くなることがあります。すべてのロード操作が INSERT モードで実行される場合、SET INTEGRITY ステートメントは表を検査して制約違反がないかどうかを (表の追加部分だけを検査することにより) 増分的に調べます。表に制約違反がないかどうかを増分的に検査できない場合は表全体の検査が実行されますが、これは表が再び使用可能になる前に実行されます。

ロード・ユーティリティーのパフォーマンスは、INSERT モードと REPLACE モードとでは同じです。

DISK_PARALLELISM、CPU_PARALLELISM、DATA_BUFFER の各パラメーターがユーザーによって指定されていない場合、ロード・ユーティリティーはこれらのパラメーターの最適な値を決定することにより、パフォーマンスを最大にするよう試みます。最適化は、ユーティリティー・ヒープにおける使用可能なサイズと空きスペースに基づいてなされます。これらのパラメーターの値は、自分で特定の必要に合わせて調整する前に、ロード・ユーティリティーに選ばせてみることをお勧めします。

以下に、ロード・ユーティリティーで利用可能な各種オプションとパフォーマンスとの関係を説明します。

ANYORDER

このファイル・タイプ修飾子は、ロードするデータの順序を保存するのを延期してパフォーマンスを上げたい場合に指定します。ロードするデータがあらかじめソートされている場合、anyorder を指定するとその順序が崩れてしまい、あらかじめソートしておくことによるメリットがそれ以降の照会で失われてしまいます。

BINARY NUMERICS および PACKED DECIMAL

これらのファイル・タイプ修飾子は、定位置数値 ASC データを固定長レコードにロードする場合のパフォーマンスを上げたい場合に使用します。

COPY YES または NO

このパラメーターは、ロード操作中に入力データのコピーを作成するかどうかを指定するのに使用します。ロードするデータはすべてロード操作中にコピーされるため (順方向回復が有効になっている必要がある)、COPY YES を指定するとロードのパフォーマンスは低下します。入出力活動が増加すると、入出力制約のシステムにおけるロード時間が大きくなることがあります。複数の装置やディレクトリー (異なるディスク上にある) を指定することにより、この操作によって生じるパフォーマンス上の不利をいくらか相殺できます。 COPY NO を指定すると、順方向回復が有効になっている場合に表がバックアップ保留状態になり、データベースや選択した表スペースをバックアップしてからでないと表にアクセスできなくなるため、全体のパフォーマンスが低下することがあります。

CPU_PARALLELISM

このパラメーターは、区画内の並列処理 (マシンが対応している場合) を利用してロードのパフォーマンスを大幅に向上させたい場合に使用します。このパラメーターには、ロード・ユーティリティーがデータ・レコードの構文解析、変換、および形式設定に使用するプロセス数またはスレッド数を指定します。指定可能な最大数は 30 です。指定された値をサポートするためのメモリーが十分でない場合、ユーティリティーはこの値を調整します。このパラメーターを指定しない場合、ロード・ユーティリティーはシステムの CPU 数に基づくデフォルト値を選択します。

ソース・データ内のレコードの順序は保持されます (165ページの図6を参照)。

表に LOB または LONG VARCHAR のいずれかのデータが含まれていると、CPU_PARALLELISM は 1 に設定されます。この場合、並列処理はサポートされません。

このパラメーターの使用は対称マルチプロセッサー (SMP) ハードウェアに限定されていませんが、非 SMP 環境でこのパラメーターを使っても、判然としたパフォーマンスの向上は期待できません。

図 6. ロード操作中に区画内の並列処理を利用した場合にソース・データのレコード順序は保持される

DATA BUFFER

DATA BUFFER パラメーターは、ロード・ユーティリティーにバッファーとして割り当てるメモリーの合計を指定します。このバッファーは、サイズに応じていくつかのエクステント にしておくことをお勧めします。エクステントは DB2 内でのデータの移動単位であり、エクステントのサイズは 4KB ページ単位で 1 ページ分または複数ページ分にすることができます。 DATA BUFFER パラメーターを使用すると入出力の待機時間が短くなるため、ラージ・オブジェクト (LOB) を扱う場合に有効です。データ・バッファーはユーティリティー・ヒープから割り当てられます。 DB2 ユーティリティーが使用するメモリーは、システム上の利用可能な記憶域の大きさに応じてなるべく大きくするようにしてください。それに応じて、データベース構成パラメーター *util_heap_sz* を変更できます。 UPDATE DATABASE CONFIGURATION コマンドについては、*コマンド解説書* を参照してください。ユーティリティー・ヒープのサイズ (Utility Heap Size) 構成パラメーターのデフォルト値は 4KB ページを単位として 5 000 ページです。ロードはユーティリティー・ヒープからのメモリーを利用するいくつかのユーティリティーの 1 つにすぎないため、ロード・ユーティリティーから利用可能なページ数としてこのパラメーターで定義する数はなるべく 50% を超えないようにし、ユーティリティー・ヒープにも十分な大きさを定義するようにしてください。 *util_heap_sz* については、*管理の手引き* を参照してください。

DISK_PARALLELISM

DISK_PARALLELISM パラメーターは、ロード・ユーティリティーがデータ・レコードをディスクに書き込むのに使用するプロセス数またはスレッド数を指定します。このパラメーターは、データのロード時に使用可能なコンテナーを利用してロードのパフォーマンスを大幅に向上させたい場合に使用します。指定可能な最大数は、CPU_PARALLELISM 値 (ロード・ユーティリティーが実際に使用している値) の 4 倍か 50

のいずれか大きいほうです。デフォルトでは、 DISK_PARALLELISM はロードする表のオブジェクトを含んでいるすべての表スペース上にある表スペース・コンテナーの合計数と等しい値です (この値が指定可能な最大数を超えていない場合)。

FASTPARSE

fastparse ファイル・タイプ修飾子は、ユーザー提供の列値に対して実行するデータ検査を簡略化することでパフォーマンスを上げる場合に使用します。このオプションは、ロードするデータが有効であることがわかっている場合にのみ使用してください。これによってパフォーマンスは 10～20% 程度向上します。

NONRECOVERABLE

このパラメーターは、表に対するロード・トランザクションを回復する必要がない場合に使用します。表にデータを移動させること以外の活動が不要になり、ロード操作はその表スペースをバックアップ保留状態にすることなく完了するため、ロードのパフォーマンスが向上します。

注: これ以後の復元やロールフォワード回復の間にそれらのロード・トランザクションが検出された場合、表は更新されずに「無効 (invalid)」としてマークされます。それ以降、この表に対する処理は無視されます。ロールフォワード操作の完了後、この表は除去することしかできません。

NOROWWARNINGS

norowwarnings ファイル・タイプ修飾子は、警告がたくさん出されることが予期される場合に、拒否された行に関する警告の記録処理を抑止してパフォーマンスを向上させます。

SAVECOUNT

このパラメーターは、ロード操作中に一貫性ポイントを確立する間隔を設定するのに使用します。一貫性ポイントを確立するために実行される活動を同期化するには時間がかかります。これをあまりに頻繁に実行すると、ロードのパフォーマンスがかなり低下します。大量の行をロードすることになっている場合は、SAVECOUNT 値を大きく指定するようにしてください (たとえばレコード数が 1 億に上るロード操作の場合は値 10,000,000 など)。

LOAD RESTART 操作は、最後の一貫性ポイントから自動的に続行します。

STATISTICS YES

このパラメーターを使用すると、ロード操作の完了後に runstats ユーティリティーを呼び出す場合よりも効率よくデータ分布と索引統計情報

を収集することができます。ただし、ロード操作そのもののパフォーマンスは低下します (特に DETAILED INDEXES ALL を指定した場合)。

最適なパフォーマンスを得るために、アプリケーションには入手可能な最大限のデータ分布と索引統計情報が必要です。統計情報が更新されると、アプリケーションではその最新の統計情報に基づいて表データへの新しいアクセス・パスを使用できます。表への新しいアクセス・パスは、DB2 BIND コマンドを使ってアプリケーション・パッケージを再バインドすることにより作成できます (*コマンド解説書* を参照)。

データを大規模な表にロードする場合は、*stat_heap_sz* (統計ヒープのサイズ) データベース構成パラメーターに指定する値をさらに大きくすることをお勧めします。 UPDATE DATABASE CONFIGURATION コマンドについては、*コマンド解説書* を参照してください。 *stat_heap_sz* については、*管理の手引き* を参照してください。

WARNINGCOUNT

このパラメーターには、ロード操作を強制終了するまでにユーティリティーが戻すことのできる警告の数の限界値を指定します。警告がわずかであること、またはまったくないことが予想される場合には、WARNINGCOUNT パラメーターをその予期される警告数に近い数に設定するか、予期される警告がない場合には 20 に設定してください。WARNINGCOUNT の数に達すると、ロード操作は停止します。これを指定することにより、ロード操作を完了する前にデータを訂正する (またはロードする表を除去してから再作成する) ことが可能になります。これはロード操作のパフォーマンスには直接影響しませんが、WARNINGCOUNT の限界値を指定しておくと、ロード操作全体が完了するのを待った後に初めて問題があることが明らかになるという事態を避けることができます。

制約事項と制限

ロード・ユーティリティーには、以下の制約事項が適用されます。

- このユーティリティーでは、通称の使用はサポートされません。
- タイプ表または構造タイプ列をもつ表へのデータのロードはサポートされていません。
- 宣言された一時表へのデータのロードはサポートされていません。
- ロード保留状態にある表スペースで表の作成や除去を実行しようとすると失敗します。

- DB2 コネクト、あるいは DB2 バージョン 2 より前の下位レベル・サーバーを経由してアクセスされるデータベースにデータをロードすることはできません。このリリースの DB2 でのみ利用可能なオプションは、旧リリースのサーバーでは使用できません。
- LOAD REPLACE 操作中にエラーが発生すると、表のオリジナル・データは失われます。ロード操作を再開できるようにするには、入力データのコピーを作成しておいてください。
- 新しくロードした行についてはトリガーが起動されません。トリガーに関連付けられたビジネス・ルールは、ロード・ユーティリティーによって適用されません。

トラブルシューティング

データのエクスポート、インポート、ロード、バインド、復元などの DB2 操作においては、メッセージ・ファイルを作成するよう指定できます。メッセージ・ファイルには、それらの操作に関連したエラー、警告、および通知の各メッセージが入れられます。 MESSAGES パラメーターでそのファイルの名前を指定します。

それらのメッセージ・ファイルは標準 ASCII テキスト・ファイルです。これを印刷するには、ご使用のオペレーティング・システムの印刷手順を使用します。表示するには、任意の ASCII エディターを使用してください。

注:

1. 操作終了後は、メッセージ・ファイルの内容を表示することしかできません。
2. メッセージ・ファイル内では、各メッセージはそれぞれ新たな行で始まり、DB2 メッセージ検索機能により提供される情報がそこに入れられます。

第4章 オートローダー

この章では DB2 UDB のオートローダー・ユーティリティーについて説明します。区分データベース環境でこのユーティリティーを利用すると、すべての区画またはいくつかの区画にわたって同時にデータをロードできます。

以下のトピックが含まれています。

- 『オートローダーの概要』
- 170ページの『オートローダーの使用に必要な特権、権限、および許可』
- 171ページの『オートローダーの使用』
- 172ページの『複数のデータベース区画へのロード』
- 173ページの『オートローダーのオプション』
- 181ページの『オートローダー・セッションの例』
- 183ページの『移行およびバック・レベル互換性』
- 184ページの『オートローダーに関するヒント』
- 186ページの『制約事項と制限』
- 186ページの『オートローダーのトラブルシューティング』

オートローダーの概要

オートローダーは、次のことを実行するためのユーティリティーです。

- あるシステム (たとえば MVS) から別のシステム (たとえば UNIX) にデータを転送する。
- そのデータを並列に区分化する。
- データをそれぞれ対応するデータベース区画に同時にロードする。

オートローダーは、次の 4 つのモードのいずれか 1 つで実行できます。

- SPLIT_AND_LOAD。データは (多くの場合は並列で) 区分化され、それぞれ対応するデータベース区画に同時にロードされます。
- SPLIT_ONLY。データは (多くの場合は並列で) 区分化され、指定した場所またはオートローダーの現行作業ディレクトリーに出力ファイルが書き込まれます。

- LOAD_ONLY。データはすでに区分化されているとします。この場合は分割プロセスが省略され、データはそれぞれ対応するデータベース区画に同時にロードされます。
- ANALYZE。すべてのデータベース区画に均一に分散する最適な区分化マップが生成されます。

区分データベースでは、非常に大きなデータがたくさんの区画にわたって収められます。データの各部分がどのデータベース区画に入るかは、区分化キーによって決定されます。また、ふさわしいデータベース区画にデータをロードする前に、データを*分割* しておく必要があります。オートローダー・ユーティリティーは、その両方の操作を実行します (図7 を参照)。

オートローダー・ユーティリティーは、ハッシュ・アルゴリズムを使うことによって、表が定義されたノードグループ内にあるデータベース区画と同じ数の出力ソケットにデータを区分化します。その後、ノードグループ内のデータベース区画にわたって、それらの出力ソケットから同時にロードします。このユーティリティーの最大の特徴は、分割プロセスとロード・プロセスの両方で必要なすべてのデータ転送のために、ソケットを使った直接 TCP/IP 通信を利用することです。また、分割フェーズで複数のデータベース区画を使用できるため、パフォーマンスが大幅に向上します。

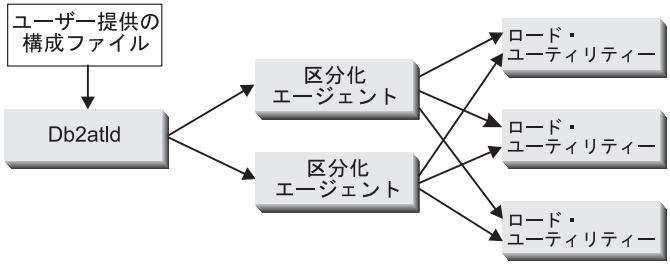


図 7. オートローダーの概要. この例では、オートローダーがソース・データを読み、2 つの区分化エージェントに半分ずつ送られます。区分化エージェントはデータを区分化して、各区分を 3 つのデータベース区画のいずれかに送ります。各区画のロード・ユーティリティーがデータをロードします。

オートローダーの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御され

ます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

オートローダー・ユーティリティーの使用に必要な権限は、ロード・ユーティリティーに必要な権限と同じです (83ページの『ロードの使用に必要な特権、権限、および許可』を参照)。さらに、ロード・ダンプ・ファイルとロード一時ファイルに、インスタンス所有者が書き込みアクセスできなければなりません。

オートローダーの使用

オートローダーを使用する前に

オートローダーを起動する前に、

1. 一時作業ディレクトリーを作成してください。このディレクトリーは、関連のあるすべてのデータベース区画からアクセス可能でなければなりません。このディレクトリーから、オートローダー・ユーティリティーを起動します。
2. オートローダー構成ファイル (181ページの『オートローダー・セッションの例』を参照) に変更を加え、それを作業ディレクトリーにコピーしてください。
3. データベース・マネージャー構成パラメーター *svcename*、およびプロファイル・レジストリー変数 **DB2COMM** が正しく設定されていることを確認してください。この確認は重要です。これは、オートローダー・ユーティリティーが、このユーティリティーを起動する作業ディレクトリーから、表を定義するデータベース区画とのリモート・データベース接続を確立するからです。

オートローダーの起動

オートローダー・ユーティリティーは、次の **db2atld** コマンドによって起動します。

```
db2atld [-config config_file] [-restart] [-terminate]

   where "-config config_file" specifies an AutoLoader configuration file
   (the default is "autoloader.cfg"); "-restart" requests restart
   of an interrupted AutoLoader operation (the configuration file
   does not need to be modified to restart); and "-terminate" requests
   termination of an interrupted AutoLoader operation.
```

構成ファイルのサンプル autoloader.cfg が、sqllib/samples/autoloader ディレクトリーにあります。このサンプル構成ファイルをコピーし、名前を変更した上で、このユーティリティーを使って実行したい操作の内容に応じてカスタマイズしてください。

複数のデータベース区画へのロード

複数のデータベース区画からなるノードグループの中にある表にデータをロードする場合、ロード・ユーティリティーを使用するには、ロードするファイルが分割されていて、適切なヘッダー情報がなければなりません。ロード・ユーティリティーは、オートローダーの分割操作において各データ・ファイルに書き込まれるヘッダー情報を調べて、データが正しい場所に送られることを確認します。

単一のデータベース区画からなるノードグループの中の表にデータをロードする場合は、その表に区分化キーが定義されていても、ファイルを分割する必要はありません。その場合、ロード操作に noheader 修飾子を指定することになります。

ロード・ユーティリティーは、オートローダーの分割操作に使用される区分化マップが、表のロードにおいて指定されるものと同じかどうかを検査します。もし同じでなければ、エラーが戻されます。また、ファイル区画が適切なデータベース区画にロードされていること、および分割時に指定された区分化キー列のデータ・タイプがカタログ内の現在の定義と一致していることを確認します。データ・ファイルが区分化された時点から、対応するデータベース表に各部分がロードされる時点までの間は、表のロード先のノードグループを再分配することはできません。もし再分配すると、ユーティリティーは区分化されたデータをロードできません。

ロード・ユーティリティーは、以下のフラット・ファイル形式をサポートしています。

- 区切りなし ASCII (ASC)
- 区切り付き ASCII (DEL)
- PC/IXF

オートローダーで区分化できるのは ASC ファイルと DEL ファイルだけです。 PC/IXF は分割できませんが、ロード操作に noheader 修飾子を使うことによって、単一のデータベース区画からなるノードグループにロードできます。 IXF ファイルを複数区画の表にロードしたい場合、まず単一区画の表に

そのファイルをロードしてから、その単一区画表から SELECT を使って複数区画表にデータを挿入する必要があります。

LOAD ROWCOUNT 文節は、オートローダー操作ではサポートされません。このパラメーターは、非区分データベース環境でのみ有効です。 LOAD SAVECOUNT 文節は、オートローダー操作で複数のスプリッターを使う場合にはサポートされません。

区分化キーの中の列が無効または拒否された場合、その行に関連するデータはどれもロードされません。*ダンプ・ファイル* が指定されていても、そこに行は書き込まれません。その代わりに、レコードが拒否されたことを示すメッセージがスプリッター・ログ・ファイルに書き込まれます。オートローダー・プロセスの完了後に、必ずスプリッター・ログ・ファイルを調べるようにしてください。

オートローダーのオプション

オートローダー構成ファイルには、たくさんのオプションを指定できます。

RELEASE レベル

この構成ファイルのリリース・レベル。構成ファイル内のこの行を削除したり変更したりしないでください。

LOAD コマンド

LOAD コマンドは、構成ファイルの中で最も重要な部分です。選択した操作モードではロードの必要がない場合でも、オートローダーがデータ処理を指示するために LOAD が必要になります。たとえば、SPLIT_ONLY モードの操作を実行する場合でも、オートローダーは LOAD コマンドから有用な情報を取り出します。 LOAD コマンドの指定によって、データの源、データ・タイプ (たとえば区切り付き ASCII)、データのロード方法、およびターゲット表の名前を示します。

スキーマ名、ファイル名、ファイル・タイプ、および表名を含む完全な LOAD コマンドを指定してください。オートローダー・ユーティリティーを使用するには、LOAD コマンドが 『db2 -f』 ファイルの形式に従っている必要があります (先行 『db2』 キーワードを除く)。この点について、またコマンド行プロセッサー (CLP) のその他のオプションについては、*コマンド解説書* を参照してください。 LOAD コマンドで特別なエスケープ・シェル文字を使う必要はありません。また、ある行の最後の文字が円記号 (¥) であれば、その次の行は現在行の続きになります。その場合、その円記号と行末文字は無視されます。

LOAD コマンドで CLIENT キーワードを指定すると、オートローダーからエラーが戻されます。

LOAD コマンドで使用できるすべてのパラメーターについては、94ページの『LOAD コマンド』を参照してください。

DATABASE パラメーター

このパラメーターは、データのロード先のデータベースを指定します。名前を指定しない場合は、デフォルト値として SAMPLE が使われます。

HOSTNAME パラメーター

このパラメーターは、データ・ファイルが属するリモート・マシンの名前を指定します。そのマシンとしては、MVS ホストまたは別のワークステーションを指定できます。名前を指定しない場合、FILE_TRANSFER_CMD パラメーターが設定されているなら、FILE_TRANSFER_CMD パラメーターの <hostname> 引き数にホスト名として nohost が渡されます。このパラメーターにデフォルト値はありません。

FILE_TRANSFER_CMD パラメーター

旧バージョンのオートローダーでは、ホスト・ファイル転送の概念 (リモート・ホストからデータを転送するようオートローダー・ユーティリティーを構成する機能) がサポートされていました。そのオプションは、FILE_TRANSFER_CMD オプションに置き換えられました。このパラメーターは、リモート・ホストからのデータ転送に使用する実行可能ファイル、バッチ・ファイル、またはスクリプトの完全修飾名を指定します。指定するパスは、オートローダーからアクセス可能でなければなりません。全パスは、実行ファイル名も含めて 254 文字 (バイト) 以下でなければなりません。

指定されたファイルを起動する前に、オートローダーはホストからデータが送られるのを予想して、名前付きパイプを確立します。作成される名前付きパイプの数は、LOAD コマンドの FROM 文節内に指定されているファイルまたは装置の数と同じです。また、LOAD コマンドのこの情報は、実行可能ファイル、バッチ・ファイル、またはスクリプトに渡されるパラメーターを指定するためにも使われます。

この情報に基づいて、オートローダーは次のようなコマンドを作成します。

```
<COMMAND> <logpath> <hostname> <basepipename>
<nummedia> <source media list>
```

ここで、

- <COMMAND> は、ホストからのデータ移動に使われる実行可能ファイル、バッチ・ファイル、またはスクリプトへの完全修飾パスです。

残りの項目はこのコマンドで使用できるパラメーターです。

- <logpath> はオートローダー・ログのパスです。 COMMAND プログラムはこのパスを使って診断データや一時データを書き込みます。
- <hostname> は HOSTNAME パラメーターで指定されるホスト名です。
- <basepipename> はオートローダーが作成する名前付きパイプのベース名です。このベース名はオートローダー・ユーティリティーによって生成され、それがシステム内で固有の名前であることが保証されます。ユーティリティーはこのベース名に追加して、必要な名前付きパイプを作成します。
- <nummedia> は、データを提供するファイルまたは装置 (LOAD コマンドの FROM 文節内に指定されているもの) の数です。
- <source media list> は、データを提供するファイルまたは装置 (LOAD コマンドの FROM 文節に指定されているもの) のそれぞれの名前です。名前に含まれる特定の文字によって問題が発生するのを未然に防ぐため、名前は二重引用符で区切られます。

AIX のサンプル・ファイル (`atldftp.drv`) が、`sqllib/samples/autoloader` ディレクトリーにあります。このサンプルは、リモート・ホストからのデータ移動に FTP を使用する方法を示しています。

SPLIT_FILE_LOCATION パラメーター

このパラメーターは、以下の 2 つの方法で使用します。

- ユーティリティーが LOAD_ONLY モードの場合、分割されたファイルのある場所のパス名を提供します。
- ユーティリティーが SPLIT_ONLY モードの場合、すでに区分化されているファイルを収める場所のパス名を提供します。

このパラメーターに値を指定しない場合、ユーティリティーが SPLIT_ONLY モードで実行されているなら、分割されたファイルは現行作業ディレクトリーに入れられます。ユーティリティーが LOAD_ONLY モードで実行されているなら、ユーティリティーは分割されたファイルを現行作業ディレクトリー内で探します。

SPLIT_FILE_LOCATION が指しているのが、 SPLIT_ONLY 操作の後ですべての区画にまたがってマウントされた NFS ディレクトリーである場合、 LOAD_ONLY 操作では、分割データ・ファイルにユーザー介入なしでアクセスすることができます。ただし SPLIT_FILE_LOCATION が、 SPLIT_ONLY 操作の後で NFS ディレクトリーを指していない場合、 LOAD_ONLY 操作が行われるディレクトリーに、各区画から分割データ・ファイルをユーザーが手動で移動しなければなりません。

OUTPUT_NODES パラメーター

このパラメーターは、ロード操作の実行されるデータベース区画を指定します。指定する区分番号は、表が定義されているデータベース区画のサブセットでなければなりません。デフォルト値はすべてです。これは、表が定義されているデータベース区画のすべてにデータがロードされることを意味します。

SPLIT_NODES パラメーター

このパラメーターでは、分割プロセスに関係するデータベース区画を指定します。これらのデータベース区画としては、ロードするデータベース区画と同じものを指定したり別のものを指定したりできます。このパラメーターに値を指定しない場合、オートローダーは分割に必要な区画の数を調べ、さらに最高のパフォーマンスを達成するにはどの区画を使うべきかを調べます。分割に必要な区画の数を決定する際の規則は以下のとおりです。

- LOAD コマンド内で ANYORDER 修飾子が指定されていない場合、オートローダー・セッションではスプリッターが 1 つだけ使われます。さらに、
  - OUTPUT_NODES パラメーターで区画が 1 つしか指定されていない場合、またはオートローダーの作業区画が OUTPUT_NODES パラメーターに指定された値の要素ではない場合は、分割区画としてオートローダーの作業区画が使用されます。
  - それ以外の場合、OUTPUT_NODES にある区画の中でオートローダー作業区画ではない最初の区画を、分割区画として使います。
- LOAD コマンド内で anyorder 修飾子を指定している 場合は、
  1. 分割区画の数を決定する計算式は次のとおりです。

     (OUTPUT_NODES 内の区画の数)/4 + 1
  2. この数の区画が、OUTPUT_NODES パラメーターに指定された区画の中から選ばれます (オートローダー作業区画を除く)。

RUN_STAT_NODE パラメーター

LOAD コマンドの STATISTICS YES 指定と組み合わせることによって、統計情報収集の対象となるデータベース区画を指定することができます。ブランクのままにした場合、または -1 の値を指定した場合、デフォルト値として出力区画リストの中の最初のデータベース区画が指定されます。

MODE パラメーター

このパラメーターは、オートローダー・ユーティリティーを実行するモードを指定します。有効な値は、SPLIT_AND_LOAD (デフォルト)、SPLIT_ONLY、LOAD_ONLY、または ANALYZE です。

SPLIT_AND_LOAD

このモードでは、データが区分化されて適切なデータベース区画にロードされます。データは、ソケットを使った直接 TCP/IP 通信で転送されます。複数の入力ファイルを使用できます。

SPLIT_ONLY

このモードでは、データの分割のみが実行されます。指定されたデータベース区画に収めるため、分割されたデータ・ファイルのセットが生成されます。各分割データ・ファイルごとに、十分な大きさの記憶域が必要です。分割機能によって、*SPLIT_FILE_LOCATION* パラメーターで指定された場所、または現行作業ディレクトリーにファイルが書き込まれます。このディレクトリー場所は書き込み可能でなければなりません。データは別個のファイルに区分化され、ファイルには *filename.xxx* (*xxx* はその分割されたファイルの区分番号) という規則に従った名前が付けられます。 LOAD コマンドで複数の入力データが指定されていれば、それらはすべて分割されます。しかし、各データベース区画ごとに、分割ファイルは 1 つずつしか生成されません。分割ファイルの名前は、最初の入力データ・ファイルの名前と同じです。

LOAD_ONLY

このモードでは、以前に分割されたデータがロードされます。データは、*filename.xxx* または *filename*.00*xxx* という規則に従った名前の別個のファイルに入れられます (*xxx* は分割ファイルの区分番号)。オートローダーは、これらのファイルを *SPLIT_FILE_LOCATION* または現行作業ディレクトリーから探します。このディレクトリー場所は読み取り可能でなければなりません。分割ファイルは、対応する区画に同時にロードされ

ます。 LOAD コマンドで infile1、infile2 のように複数の入力データが指定されている場合、オートローダーは infile1.xxx が存在すればそれをロードします。それがない場合、infile1.00xxx が存在すればそれをロードします。どちらもない場合、オートローダーはエラーを戻します。両方が存在する場合、オートローダーは infile1.xxx をロードします。*xxx* または 00*xxx* のどちらかのファイル・タイプの最初の infile1 がロードされたら、infile2 の検査が始まります。すべての入力ファイルがロードされるまで、このプロセスが繰り返されます。

ANALYZE

このモードでは、あるノードグループに対するカスタマイズされた最適区分化マップが生成されます。たくさんのレコードが含まれている 1 つのデータ・ファイルを入力ファイルとして指定することをお勧めします (複数の入力ファイルも使用できます)。そうすれば、ノードグループ内の各データベース区画にわたって、より均一なデータ分配がマップにより作成されます。出力は、*MAP_FILE_OUTPUT* パラメーターで指定されるファイルに書き込まれます。新しい区分化マップを有効にするには、その前に REDISTRIBUTE NODEGROUP コマンド (*コマンド解説書* を参照) を起動する必要があります。それ以降にオートローダーを SPLIT_AND_LOAD モードで起動すると、自動的に新しい区分化マップが使われます。ノードグループのデフォルト区分化マップを変更することなく新しい区分化マップに従ってデータを区分化する場合は、*MAP_FILE_INPUT* パラメーターを使用できます。

LOGFILE パラメーター

このパラメーターは、オートローダー・ユーティリティーの使用する一時ファイルと永続ファイルのベース名を指定するのに使用されます。

```
<logfile>.split.cfg ...
   Configuration file for all splitters.
<logfile>.split.<3-digit-node-number>.log ...
   Log file for each splitter.
<logfile>.pmap.<pid> ...
   Internal temporary file, where <pid> is the process ID
      of this AutoLoader job.
<logfile>.load.<3-digit-node-number> ...
   Message file for each loading process if there is no
      message file specified in the LOAD command.
```

LOGFILE パラメーターにパスを指定することはできますが、そのパスが存在して、アクセス可能であることを確認する必要があります。デフォルト値は ./autoloader.log です。

注: 複数のオートローダー・セッションを同時に実行する場合は、指定したベース名またはパス名が固有のものであることを確認してください。

AUTHENTICATION パラメーターと PASSWORD パラメーター

これらのパラメーターは、ロード時に、スプリッター・プログラムのリモート呼び出しまたはクライアント / サーバー・データベース接続でパラメーターが必要な場合に指定する必要があります。AUTHENTICATION のデフォルト値は NO (パスワード検査をしない) であり、PASSWORD パラメーターに指定した値は無視されます。

MPP 環境のローカル・データベース接続の概念が拡張され、任意の MPP インスタンスのどのノードからの接続も含まれるようになりました。つまり、AUTHENTICATION サーバーを使ってインスタンスが構成されていても、 db2nodes.cfg ファイルで定義されているノードのいずれかから接続を試みているなら、パスワードは必要ありません。オートローダー構成ファイル内に AUTHENTICATION フラグが設定されていない場合、または NO に設定されている場合、PASSWORD パラメーターの値が指定されていないなら、オートローダーはこの新しい接続方法を利用します。オートローダーのパスワード指定が必要になるのは、システム上のプログラムのリモート実行にパスワードが必要な場合だけです。たとえば、UNIX システムにおいて **rsh** の実行のために .rhosts ファイルが正しく設定されていない場合、パスワードが必要になります。

別の方法として、パスワードが必要な場合に、ユーザー作成のパスワード・ファイルへの完全修飾パスを定義する DB2 レジストリー変数 **DB2ATLD_PWFILE** を設定することもできます。パスワード・ファイルと完全修飾パスは、どちらもユーティリティーからアクセス可能でなければなりません。この変数が定義されているなら、その値が指すファイルの中の最初の語がパスワードになります。

MAX_NUM_SPLITTERS パラメーター

このパラメーターは、1 つのオートローダー・ジョブ内で使用できるスプリッター・プロセスの最大数を指定します。デフォルト値は 25 です。

FORCE パラメーター

このパラメーターは、起動時に宛先の区画または表スペースのいくつか

がオフラインになっていることをオートローダー・ユーティリティーが検出した場合でも、オートローダー・ジョブが続行されるようにするために使います。この値に NO を指定した場合、いくつかの区画が使用できないなら、データ処理は実行されません。値に YES を指定した場合、使用可能なデータベース区画にデータがロードされ、それ以外はすべて無視されます。このパラメーターのデフォルト値は NO です。

STATUS_INTERVAL パラメーター

このパラメーターは、進行中メッセージを生成する前にロードするデータのメガバイト数 (MB) を指定します。有効な値の範囲は 1〜4000 の整数です。デフォルト値は 100 です。

PORTS パラメーター

このパラメーターは、内部オートローダー通信のためのソケットの作成に使う TCP ポートの範囲を指定します。デフォルトの範囲は 6000〜6063 です。オートローダー起動時に DB2 レジストリー変数 **DB2ATLD_PORTS** が定義されているなら、このパラメーターに指定した値はその値で置き換えられます。

CHECK_LEVEL パラメーター

このパラメーターは、入出力時にレコード切り捨てのための検査を実行するかどうかを指定します。有効な値は CHECK と NOCHECK です。デフォルト値は NOCHECK です。

MAP_FILE_INPUT パラメーター

このパラメーターは、カスタマイズされた区分化マップが入っているファイルを指す入力ファイルの名前を指定します。区分化マップが (デフォルトではない) カスタマイズされたマップの場合は、このパラメーターを指定する必要があります。カスタマイズされた区分化マップを作成するには、オートローダーを ANALYZE モードで起動して、最適化マップを生成します。実際にロードを実行するには、その前にデータベースの各データベース区画にこのマップを移動しておく必要があります。

MAP_FILE_OUTPUT パラメーター

このパラメーターは、オートローダーを ANALYZE モードで起動する場合の区分化マップの名前を指定します。最適区分化マップは、すべてのデータベース区画にわたってデータを均一に分配します。このパラメーターに値が指定されていない場合にユーティリティーを ANALYZE モードで実行すると、エラーが戻されます。

TRACE パラメーター

このパラメーターは、すべてのデータ変換プロセスと出力ハッシュ値の

ダンプを調べることが必要になった場合に、トレースするレコードの数を指定します。デフォルト値は 0 (トレースしない) です。

NEWLINE パラメーター

このパラメーターは、データ・ファイル内の各レコードを区切るのに使われる文字を指定します。このパラメーターが有効なのは、入力データ・ファイルが固定長 ASC ファイルであり、その各レコードが改行文字で区切られていて、しかも LOAD コマンドの reclen 修飾子が指定されている場合だけです。値として YES を指定すると、オートローダーはレコードが改行文字で終わっているかどうかを常に検査します。また、レコード長が reclen 修飾子で指定された長さと一致するかどうかも検査されます。デフォルト値は NO です。

オートローダー・セッションの例

次の例は、AIX でのオートローダー構成ファイルのサンプルです。

```
###############
# release level
###############
RELEASE=V7.00
#################
# CLP load command
#################
db2 load from /home/user/atld_work/test.dat of del replace into user.test
###############
# database name
###############
database=wsdb
################
# split partition list
################
SPLIT_NODES=(0,2)
##############
# running mode
##############
mode=split_and_load
###############
# log file token
###############
logfile=mylog
####################################
# frequency of progressive information
#
# print out progressive info every 10
# mega-bytes of data
####################################
STATUS_INTERVAL=10
```

オートローダー・セッションの例

以下のコマンドは、この構成ファイルに対応して出されたものであり、関連する各データベース区画からアクセス可能なパスおよび一時作業ディレクトリーを含んでいます。また、構成ファイルの名前 (sample.atld.cfg) も指定されています。

```
/home/user/atld_work/ $ db2atld -config sample.atld.cfg
```

このコマンドを起動すると、出力は次のようになります。

```
/home/user/atld_work/ $ db2atld -config sample.atld.cfg
Utility program: "db2atld". Version: "07010".
Start reading autoloader configuration file: sample.atld.cfg.
Finish reading autoloader configuration file: sample.atld.cfg.
Start initializing autoloader process.
Finish initializing autoloader process.
The AutoLoader is now issuing all LOAD requests.
The LOAD operation has begun on partition "2".
The LOAD operation has begun on partition "0".
The LOAD operation has begun on partition "1".
The AutoLoader is now issuing all split requests.
Start db2split on node "0" in background.
Start db2split on node "2" in background.
The AutoLoader is waiting for all splitters to complete.
The utility has read "10" megabytes from the source data.
The utility has read "20" megabytes from the source data.
The utility has read "30" megabytes from the source data.
The utility has read "40" megabytes from the source data.
The utility has read "50" megabytes from the source data.
The utility has read "60" megabytes from the source data.
The utility has completed reading "62" megabytes from the user data.
The remote execution of the splitter utility on partition "0"
finished with remote execution code "0".
The AutoLoader is waiting for all LOAD operations to complete.
The remote execution of the splitter utility on partition "2"
finished with remote execution code "0".

  Operation      Node     SQL Code     Result
_____________________________________________________________________
  LOAD           002      +00000000    Success.
_____________________________________________________________________
  LOAD           000      +00000000    Success.
_____________________________________________________________________
  LOAD           001      +00000000    Success.
_____________________________________________________________________
  SPLIT          000      +00000000    Success.
_____________________________________________________________________
  SPLIT          002      +00000000    Success.
_____________________________________________________________________
  PRE-SPLIT      000      +00000000    Success.
_____________________________________________________________________
  RESULTS:       3 of 3 LOADs completed successfully.
_____________________________________________________________________
Summary of Splitters:
Rows Read         500003
```

```
Rows Rejected     125214
Rows Partitioned  374789
Summary of LOADs:
Rows Read      374789
Rows Skipped   0
Rows Loaded    374789
Rows Rejected  0
Rows Deleted   0
Rows Committed 374789
```

オートローダーの生成するメッセージの本体は、関連するデータベース区画の初期化に関するものです。分割プロセスとロード・プロセスの進行状況は、別個のテキスト・ファイルに記録されます。また、オートローダー・プロセスの終了も記録されます。

さらに、実行された操作、使用された区画、戻された SQL コード、および得られた結果について要約した一覧表も生成されます。 0 以外の SQL コードが戻された場合は、記録された特定の警告またはエラーがメッセージ・ファイルに示されます。

出力の最後は、オートローダー・ジョブのレコード要約です。

移行およびバック・レベル互換性

オートローダー・ユーティリティーに関連して、移行およびバック・レベル互換性の問題があります。

- 旧バージョンのオートローダー・ユーティリティーは、**db2autold** コマンドによって起動されました。現行バージョンは、**db2atld** コマンドによって起動します。
- **db2atld** は名前付きパイプではなくソケットを内部通信チャネルとして使い、デフォルト範囲 (6063〜6000) の中から TCP ポート番号を選択します。しかし、システムがこの範囲を他のアプリケーションのために使う必要がある場合は、移行時に次の 2 つのいずれかを選択できます。
  - オートローダー構成ファイルの PORTS パラメーターを使うことによって、デフォルト以外のポート範囲を指定する。
  - 以下のように範囲を指定することによって、DB2 レジストリー変数 **DB2ATLD_PORTS** を定義する。

    ```
    <lower-port-number>:<higher-port-number>
    ```

  TCP ポート範囲決定の優先順位は、DB2 レジストリー変数 **DB2ATLD_PORTS**、オートローダー構成パラメーター PORTS、デフォルトの順です。

- クライアント / サーバー・データベース接続にパスワードが必要な場合は、移行時に次の 2 つのいずれかを選択できます。
  - オートローダー構成パラメーター AUTHENTICATION および PASSWORD を使用する。 AUTHENTICATION を YES に設定した場合、PASSWORD が定義されているなら、認証の際にパスワードが使われます。 AUTHENTICATION を YES に設定していて PASSWORD が定義されていないなら、パスワードを定義するよう促されます。
  - パスワードが入っているファイルを指すように DB2 レジストリー値 **DB2ATLD_PWFILE** を設定する。ファイルを指定すると、ファイルの内容が評価されて、ブランクで区切られた最初の文字列がパスワードになります。レジストリー値を定義した場合、レジストリー値は最後に評価されるため、その他のパスワードはそれによってオーバーライドされます。

オートローダーに関するヒント

オートローダー・ユーティリティーのご利用の前に、以下のことを考慮してください。

- オートローダー・ユーティリティーに慣れるために、まず小さなデータを使ってこのユーティリティーを操作してください。
- 入力データがあらかじめソートされている場合、または特定の順序になっている場合に、ロード・プロセス中にその順序を維持したいなら、分割に使うデータベース区画は 1 つだけにしてください。並列分割では、必ずしもデータを受け取ったのと同じ順序でロードするとは限りません。 LOAD コマンドで anyorder 修飾子を指定しないと、オートローダーはデフォルトで単一スプリッターを選択します。
- 複数に分割されたファイルからラージ・オブジェクト (LOB) をロードする場合 (つまりロード・ユーティリティーの lobsinfile 修飾子を使っている場合)、 LOB ファイルの入っているすべてのディレクトリーは、ロード先のすべてのデータベース区画から読み取り可能でなければなりません。 LOB を処理する場合、LOAD パラメーター *lob-path* は完全修飾パスでなければなりません。
- すべてのオートローダー一時ファイルは、オートローダー構成パラメーター LOGFILE で指定されるディレクトリーに入れられます。このディレクトリーは、分割が実行されるすべての区画からネットワーク・アクセス可能 (読み取りと書き込みの両方が可能) でなければなりません。一時ファイル用に複数の異なるディレクトリーを指定することによって、それぞれ異なる表スペースに含まれる別個の表にデータをロードする複数のオートローダー・ジョブを同時に実行できます。

- 1 つのオートローダー・ジョブあたりのアクティブなデータベース接続の最大数は、オートローダー構成パラメーター OUTPUT_NODES で定義されるロード区画の数です。データベース構成パラメーター *maxxappls* (アクティブ・アプリケーションの最大数) が十分に大きな値に設定されていることを確認してください。
- 起動時にロード区画や関連する表スペースのいくつかがオフラインになっていることをオートローダーが検出した場合でも、オートローダー・ジョブを続行させることができます。そうするには、オートローダー構成ファイルで FORCE=YES と指定してください。
- オートローダー・ジョブの進行状況をモニターするには、オートローダー構成パラメーター STATUS_INTERVAL を使います。オートローダーは、指定された時間間隔でメッセージを戻して、処理の終わったデータの量 (MB) を表示します。
- SPLIT_NODES パラメーターで定義される分割区画が、OUTPUT_NODES パラメーターで定義されるロード区画と異なっている場合、CPU サイクルを巡る競合が少なくなるため、パフォーマンスの改善が期待されます。オートローダー・ユーティリティー自体は、分割操作にもロード操作にも関係していないデータベース区画上から起動してください。 SMP システムでは、使用可能な CPU ごとにスプリッター・タスクを少なくとも 1 つずつ割り当てることによって、パフォーマンスを改善できます。
- オートローダーは、LOAD コマンド内の MESSAGES パラメーターを無視し、LOAD コマンドからのすべてのメッセージを load_log.*XXX* というファイルに出力します。このファイルには、データベース区画 *XXX* におけるロード・プロセスから受け取ったメッセージが入れられます。また、オートローダーは splt_log.*XXX* というファイルも作成します。このファイルには、データベース区画 *XXX* 上の分割プロセスから受け取ったメッセージが入れられます。さらに、このユーティリティーは、オートローダーのメイン・スクリプトから受け取ったメッセージを入れるファイル autoload.log を作成します。パイプと一時ディレクトリーがすべて正しく設定されていることを確認するには、そのファイルを調べてください。
- オートローダーは、統計情報を収集する出力データベース区画を 1 つだけ選択します。その区画を指定するには、オートローダー構成パラメーター RUN_STAT_NODE を使います。
- オートローダーを複数起動することによって、別々の表に同時にデータをロードできます。次のことを確認してください。
  - すべての表が別々の表スペース上にある。
  - すべてのオートローダー操作を別々のディレクトリーから起動する。

– 一時ファイルの作成に使うデータ・ファイル名が、それぞれのオートローダー操作ごとに固有である。

制約事項と制限

オートローダー・ユーティリティーには、以下の制約事項が適用されます。

- ロード操作の入力ファイルの位置として磁気テープ装置を指定することはできません。
- オートローダーは、LOAD コマンドの CLIENT オプションをサポートしません。
- オートローダーは、LOAD コマンドの ROWCOUNT オプションをサポートしません。
- 複数のデータベース区画を使ってデータを区分化してからデータをロードする場合、LOAD コマンドにおいて 0 より大きい SAVECOUNT 値はサポートされていません。
- オートローダーによる、構造タイプ列をもつ表へのデータのロードはサポートされていません。

ロードに関する制約事項 (オートローダーにも適用される) については、 167 ページの『制約事項と制限』を参照してください。

オートローダーのトラブルシューティング

オートローダー・ユーティリティーが停止したと思われる場合は、以下のことをすることができます。

- オートローダー構成ファイルのパラメーター STATUS_INTERVAL を使って、オートローダー・ジョブの進行状況をモニターする。
- `<logfile>.split.<3-digit-node-number>.log` ファイルを調べて、各分割データベース区画のスプリッター・プロセスの状況を確認する。そこに異常がなく、オートローダー構成ファイルに TRACE パラメーターが設定されていれば、それらのログ・ファイル内に特定の数のレコードについてのトレース・メッセージがあるはずです。
- LOAD メッセージ・ファイルまたは `<logfile>.load.<3-digit-node-number>` ファイルを調べて、ロード・エラー・メッセージがあるかどうかを確認する。
- オートローダー・プロセスのいずれかに問題が発生したことを暗示するエラーを発見したなら、現在のオートローダー・ジョブを中断する。

それでもオートローダー・ユーティリティーの障害が継続するようであれば、次のことをすることができます。

1. オートローダー構成ファイルの MODE パラメーターを SPLIT_ONLY に設定して、もう一度ユーティリティーを起動する。
2. 分割データ・ファイルを調べて、それら自体に異常があるかどうかを確認する。分割ファイルが正常なら、それらのファイルの 1 つを適切なデータベース区画に手動でロードする。
3. データが正常にロードされるなら、それ以外にオートローダーの問題またはデータベース・システムの問題が存在する可能性があります。 IBM サービス技術員に連絡してください。

AIX バージョン 4.2 では、次に示すライブラリー (示してあるレベル以上のもの) がない場合、 **db2atld** がハングすることがあります。

```
bos.rte.libc            4.2.1.13 (PTF U458582)
bos.rte.libpthreads     4.2.1.5  (PTF U458538)
```

以下のエラー・シナリオは、DB2 エンタープライズ拡張エディション (Windows NT 版) 上で実行しているオートローダー・ユーティリティーに当てはまります。

マルチ・ホーム・マシン (複数のネットワーク・カードがインストールされているマシン) 上で **db2atld** を実行している場合は、そのマシンが正しく構成されているかどうかを確認するために、オートローダーを実行しているマシン上から **hostname** コマンドを入力した後、同じマシンからこのホスト名に対して PING を実行します。そこで戻される IP アドレスは、DB2 MPP ノード・リスト内の別のマシンからこのホスト名を PING した場合に戻される IP アドレスと同じであるはずです。マシンが正しく構成されていない場合は、ユーティリティーが SQL6555N エラーを戻します。さらに、オートローダー構成ファイルの OUTPUTNODES パラメーターで定義されているいくつかのロード・ノード上にある db2diag.log ファイルに、エラー・メッセージ errno = 10061 (接続が拒否された (connection refused)) が出力されます。

Windows NT マシンにおいて、ローカル・ホスト名に対して戻される IP アドレスは、DNS や hosts ファイルから取り出したものではなく、「コントロール パネル」ネットワーク・アイコンでローカルに構成された情報から取り出したものです。 Windows NT バージョン 4.0 の障害として、マルチ・ホーム・マシン上で戻される IP アドレスの順序では、「コントロール パネル」ネットワーク・アイコンで構成されているバインド順序が無視されます。この問題を解決するための情報は、Microsoft サポートのオンライン文書 Q171320 にあります。

第5章 DB2 データ・リンク・マネージャーのデータの移動

この章では、DB2 のエクスポート、インポート、およびロード・ユーティリティーを使用することにより、DB2 データ・リンク・マネージャーのデータを移動する方法について説明します。

これらのユーティリティーで使用できるファイル形式については、225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

DB2 データ・リンク・マネージャーについては、*DB2 データ・リンク・マネージャー (Windows 版) 概説およびインストール* を参照してください。

以下のトピックが含まれています。

- 190ページの『DB2 データ・リンク・マネージャーのデータの移動 - エクスポートの使用』
- 194ページの『DB2 データ・リンク・マネージャーのデータの移動 - インポートの使用』
- 194ページの『DB2 データ・リンク・マネージャーのデータの移動 - ロードの使用』

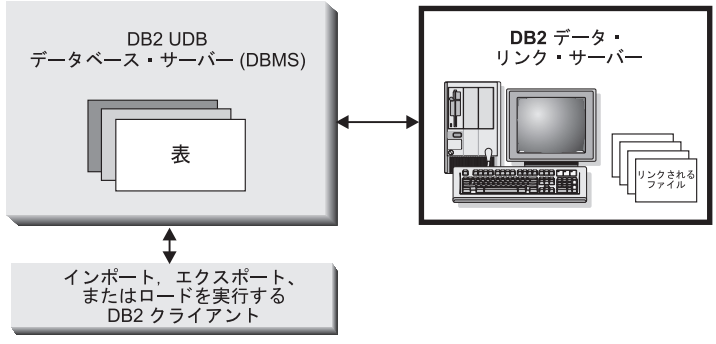


図 8. DB2 データ・リンク・マネージャーのデータの移動. 表データがデータベース内にあり、一方 DATALINK 列の参照するファイルがデータ・リンク・サーバー上にあるため、エクスポート、インポート、およびロードの各ユーティリティーは、データベース上のデータを移動することに加えて、対応するデータ・リンク・サーバー上のデータ・ファイルも移動する必要があります。

DB2 データ・リンク・マネージャーのデータの移動 - エクスポートの使用

表データがデータベース内にあり、一方 DATALINK 列の参照するファイルがデータ・リンク・サーバー上にあるため、エクスポート・ユーティリティーは、データベース上のデータを移動することに加えて、対応するデータ・リンク・サーバー上のデータ・ファイルも移動する必要があります (189ページの図 8 を参照)。そのためにエクスポート・ユーティリティーは、それぞれのデータ・リンク・サーバーごとに 1 つの制御ファイルを作成します。制御ファイルの名前は、データ・リンク・サーバーの名前と同じです。制御ファイルは、すべて `dlfm/`*YYYYMMDD/HHMMSS* という名前の新しいディレクトリーの中に作成されます (*YYYYMMDD* は*年月日*、また *HHMMSS* は*時分秒* を表す)。このディレクトリーは、エクスポートされたデータ・ファイルが入る場所と同じディレクトリーの下に作成されます。制御ファイルには、エクスポートされる行の DATALINK 列によって参照される対応する DB2 データ・リンク・マネージャーのファイル名のリストが入れられます。

AIX 上の分散ファイル・システム (DFS) の場合、エクスポート・ユーティリティーは、セル内のすべてのデータ・リンク・サーバーに対して制御ファイルを 1 つ作成します。制御ファイルはセルごとに 1 つずつ存在します。制御ファイルの名前は、セルの名前と同じです。この制御ファイルには、セル内にあるすべての DB2 データ・リンク・マネージャー・ファイルの URL (エクスポートされる行の DATALINK 列によって参照されます) のリストが入れられます。

WINDOWS NT オペレーティング・システムにおいてエクスポート・ユーティリティーは、すべてのデータ・リンク・サーバーに対して制御ファイルを 1 つだけ作成します。この制御ファイルの名前は `ctrlfile.lst` です。これは、`dlfm¥`*YYYYMMDD¥HHMMSS* という名前の新しいディレクトリーの中に作成されます。このディレクトリーは、エクスポートされたデータ・ファイルが入る場所と同じディレクトリーの下に作成されます。この制御ファイルには、エクスポートされる行の DATALINK 列によって参照されるすべての DB2 データ・リンク・マネージャーの URL のリストが入れられます。

NO LINK CONTROL プロパティーの DATALINK 値は、制御ファイルには入れられません。

制御ファイルは、それぞれ対応するデータ・リンク・サーバーに移送する必要があります。 DFS の場合、各セルの制御ファイルは、該当セル内のデータ・リンク・サーバーのいずれかに移送されなければなりません。 Windows NT オペレーティング・システムでは、ただ 1 つの制御ファイルを、参照されているすべてのデータ・リンク・サーバーに移送しなければなりません。各デー

タ・リンク・サーバーごとに、制御ファイル名を指定して **dlfm_export** ユーティリティーを実行する必要があります。このユーティリティーは、そのデータ・リンク・サーバーの制御ファイルに示されているファイルのアーカイブを作成します。 DFS の場合は、制御ファイルが移送された先のデータ・リンク・サーバーで、 **dlfm_export** ユーティリティーを実行しなければなりません。このユーティリティーは、セル内すべてのデータ・リンク・サーバーの制御ファイルに示されているファイルのアーカイブを作成します。
DLFM_FS_ENVIRONMENT レジストリー変数を適切に設定してから、 **dlfm_export** ユーティリティーを実行してください。

DATALINK 列が参照する表および対応するファイルの一貫性のあるコピーを作成するため、次のことを実行してください。

1. 以下のコマンドを出して、エクスポート操作中に更新トランザクションが実行されないようにします。

   ```
   db2 quiesce tablespaces for table tablename share
   ```

2. エクスポート・ユーティリティーを起動します。
3. 各データ・リンク・サーバーで、root 権限で **dlfm_export** ユーティリティーを実行します。こうすると、データ・リンク・マネージャーの管理者からアクセスできないアーカイブ・ファイルが正常に作成されます。 DFS の場合、**dlfm_export** ユーティリティーは、制御ファイルに示されているファイルをアーカイブする前に、 DCE ネットワークのルート認証を取得します。このユーティリティーは、アーカイブされているファイルの ACL 情報は取り込みません。 **dlfm_export** への入力として、エクスポート・ユーティリティーが生成した制御ファイルの名前を指定します。
4. 以下のコマンドを出して、表の更新を有効にします。

   ```
   db2 quiesce tablespaces for table tablename reset
   ```

エクスポート・ユーティリティーが SQL アプリケーションとして実行されます。 SELECT ステートメントの条件を満たす行と列がデータベースから抽出されます。 DATALINK 列に対しては、SELECT ステートメントでスカラー関数を指定しないでください。

エクスポート・ユーティリティーは、以下のようなファイルを生成します。

- エクスポート・データ・ファイル。このファイルの DATALINK 列の値の形式は、インポートおよびロード・ユーティリティーで使われる形式と同じです。 DATALINK 列の値がヌルの場合は、他のヌル列と同じように扱われます。
- それぞれのデータ・リンク・サーバーの制御ファイル。制御ファイルには、そのデータ・リンク・サーバーからエクスポートされるすべてのファイルの

完全なパスと名前のリストが入れられます。 DFS の場合、制御ファイルはセルごとに 1 つずつ存在します。 Windows NT オペレーティング・システムの場合、DATALINK 列の値が参照しているすべてのデータ・リンク・サーバーに対してただ 1 つの制御ファイルが作成されます。

次のようにして、 **dlfm_export** ユーティリティーを使って 1 つ以上のデータ・リンク・サーバーからファイルをエクスポートします。

```
dlfm_export control-file-name archive-file-name
```

control-file-name は DB2 クライアント上でエクスポート・ユーティリティーを実行して生成される制御ファイルの名前、 *archive-file-name* は生成されるアーカイブ・ファイルの名前です。 *archive-file-name* のデフォルトは `export.tar` で、現行作業ディレクトリーに入れられます。

dlfm_export が生成したアーカイブからファイルを取り出して復元するには、**dlfm_import** という補足ユーティリティーを使用します。アーカイブ・ファイルを同じデータ・リンク・サーバーに復元するか別のサーバーに復元するかに関係なく、このユーティリティーを実行しなければなりません。

次のようにして、**dlfm_import** ユーティリティーを使ってアーカイブからファイルを取り出します。

```
dlfm_import archive-file-name [LISTFILES]
```

archive-file-name はファイルの復元に使うアーカイブ・ファイルの名前です。 *archive-file-name* のデフォルトは `export.tar` です。 LISTFILES はオプションのキーワードであり、これを指定すると、アーカイブに入っているファイルのリストが戻されます。各データ・リンク・サーバーごとに **dlfm_import** ユーティリティーを root 権限で実行します。これは、アーカイブ・ファイルを別のデータ・リンク・サーバーに復元する場合、そのサーバーのディレクトリー構造やユーザー ID が、**dlfm_export** を実行したサーバーとは異なっている場合があるからです。 DFS の場合、ターゲット・セル内のデータ・リンク・サーバーのいずれかで **dlfm_import** ユーティリティーを実行します。このユーティリティーは、アーカイブからファイルを抽出する前に、 DCE ネットワークのルート認証を取得します。 **DLFM_FS_ENVIRONMENT** レジストリー変数を適切に設定してから、 **dlfm_import** ユーティリティーを実行してください。

注: **dlfm_export** ユーティリティーを実行したデータ・リンク・サーバーとは別のサーバーで **dlfm_import** ユーティリティーを実行した場合、ファイルは正しいパスに復元されます。インポートしているマシンにユーザー ID のいくつかが存在しない場合は、root がファイルを所有します。これらの

ファイルをデータベースに挿入する前に、すべてのファイルに適切な許可があること、また適切なユーザー ID に属していることを確認してください。 DFS の場合、必要なファイル・セットを作成し、必要な ACL をセットアップしてから、 **dlfm_import** ユーティリティーを実行してください。このユーティリティーは、ファイルの抽出に必要なディレクトリーを作成します (ない場合)。

次の表は、データベース SystemA が参照している DB2 データとファイルを、データベース SystemB へエクスポートする方法 (DFS 以外の場合) を示しています。 SystemA は、データ・リンク・サーバー DLFM1 および DLFM2 を使用しています。 SystemB は、データ・リンク・サーバー DLFMX および DLFMY を使用しています。 DLFM1 上のファイルを DLFMX にエクスポートし、DLFM2 上のファイルを DLFMY にエクスポートします。

データベース SystemA (データ・リンク・サーバー DLFM1 および DLFM2 を使用)			**ステップ**
ファイル上の DB2 データ	File1 (DLFM1 でのファイル名)	File2 (DLFM2 でのファイル名)	1) 2 つのデータ・リンク・サーバーで dlfm_export コマンドを root として実行します。これによって、2 つのデータ・リンク・サーバーにアーカイブが作成されます。
データベース SystemB (データ・リンク・サーバー DLFMX および DLFMY を使用)			
	DLFMX 上でアーカイブから復元	DLFMY 上でアーカイブから復元	2) 2 つのデータ・リンク・サーバーで dlfm_import コマンドを root として実行します。
			3) SystemB 上で IMPORT コマンドを実行します (DL_URL_REPLACE_PREFIX パラメーターを使って、各エクスポート・ファイルに対する適切なデータ・リンク・サーバーを指定)。
SystemB 上で IMPORT コマンドを実行すると、DATALINK 列が参照している SystemA のデータとすべてのファイルがインポートされます。			

DB2 データ・リンク・マネージャーのデータの移動 - インポートの使用

表データがデータベース内にあり、一方 DATALINK 列の参照するファイルがデータ・リンク・サーバー上にあるため、インポート・ユーティリティーは、データベース上のデータを移動することに加えて、対応するデータ・リンク・サーバー上のデータ・ファイルも移動する必要があります (189ページの図8 を参照)。

ターゲット・データベースに対してインポート・ユーティリティーを実行する前に、次のことを実行してください。

1. 参照されることになるファイルを、適切なデータ・リンク・サーバーにコピーします。 **dlfm_import** ユーティリティーを使用することによって、**dlfm_export** ユーティリティーの生成したアーカイブからファイルを取り出すことができます。
2. データ・リンク・サーバー上のデータ・リンク・マネージャーに対して、1 つまたは複数の接頭部名を定義します。 (データベース登録などのその他の管理作業も実行できます。)
3. 必要なら、DATALINK 列の URL のデータ・リンク・サーバー情報を、SQL 表のエクスポート・データから更新します。 (元の構成のデータ・リンク・サーバーがターゲット・ロケーションのサーバーと同じなら、データ・リンク・サーバー名を更新する必要はありません。) DFS の場合、必要に応じて、URL のセル名情報 (DATALINK 列のもの) を、 SQL 表のエクスポート・データから更新します。
4. DB2 データ・リンク・マネージャー構成ファイル内で、ターゲット構成にデータ・リンク・サーバーを定義します。 DFS の場合、DB2 データ・リンク・マネージャー構成ファイル内で、ターゲット構成にデータ・リンク・サーバーを定義します。 **dlfm server_conf** コマンドと **dlfm client_conf** コマンドを使って、ターゲット・セル内の DB2 データ・リンク・マネージャーを登録します。

ターゲット・データベース・ファイルに対してインポート・ユーティリティーを実行すると、DATALINK 列データの参照するファイルが、該当するデータ・リンク・サーバー上でリンクされます。

DB2 データ・リンク・マネージャーのデータの移動 - ロードの使用

FILE LINK CONTROL で定義された DATALINK 列を含む表の中にデータをロードする場合は、ロード・ユーティリティーを起動する前に以下のステップを実行してください。 (すべての DATALINK 列が NO LINK CONTROL で定義されている場合、これらのステップは不要です。)

1. DATALINK 列の値が参照する予定のデータ・リンク・サーバー上に DB2 データ・リンク・マネージャーがインストールされていることを確認します。 DFS の場合、ターゲット・セル内の DB2 データ・リンク・マネージャーが登録されていることを確認します。
2. データベースが DB2 データ・リンク・マネージャーに登録されていることを確認します。
3. DATALINK 値として挿入されることになるファイルを、適切なデータ・リンク・サーバーにコピーします。
4. データ・リンク・サーバー上の DB2 データ・リンク・マネージャーに対して、1 つまたは複数の接頭部名を定義します。
5. ロードする DATALINK データが参照するデータ・リンク・サーバーを、DB2 データ・リンク・マネージャー構成ファイルの中で登録します。 DFS の場合、ロードする DATALINK データが参照するターゲット構成のセルを、 DB2 データ・リンク・マネージャー構成ファイルの中で登録します。

ロード・ユーティリティー実行中に DB2 とデータ・リンク・サーバーとの接続に失敗して、ロード操作が失敗する可能性があります。その場合、以下のようにします。

1. データ・リンク・サーバーと DB2 データ・リンク・マネージャーを開始する。
2. LOAD RESTART コマンドを出す (94ページの『LOAD コマンド』を参照)。

ロード操作中に失敗したリンクはデータ保全性違反とみなされ、固有索引違反と同じように扱われます。その結果、1 つまたは複数の DATALINK 列を含む表をロードするために、特別な例外が定義されています。追加情報については、*SQL 解説書* の中の例外に関する説明を参照してください。

ロード・ユーティリティーの以下の機能は、 DATALINK 列をもつ表のロード時にはサポートされません。

- CPU_PARALLELISM (値は強制的に 1 になります)
- LOAD REPLACE
- LOAD TERMINATE
- LOAD COPY

第6章 システム間のデータの移動

この章では、DB2 のエクスポート、インポート、およびロード・ユーティリティーを使用することにより、異なるプラットフォーム間で、また DRDA ホスト・データベースとの間でデータをやりとりする方法について説明します。社内の複数のデータベース間でのデータの移動のもう 1 つの方法である DB2 DataPropagator についても説明されています。操作可能データベースからウェアハウス・データベースにデータを移動するために使用できるデータウェアハウスセンター (DWC) についても説明されています。

以下のトピックが含まれています。

- 『プラットフォーム間のデータの移動』
- 200ページの『db2move ツールを使用したデータの移動』
- 205ページの『DB2 コネクトによるデータの移動』
- 207ページの『タイプ表間のデータ移動』
- 212ページの『レプリケーションを使ったデータの移動』
- 214ページの『データウェアハウスセンターによるデータの移動』

プラットフォーム間のデータの移動

プラットフォームを超えてデータをエクスポート、インポート、またはロードする場合、互換性は重要です。以下の部分では、異なるオペレーティング・システム間でデータを移動する際の PC/IXF、区切り付き ASCII (DEL)、および WSF ファイル形式の考慮事項について説明します。 DB2 データ移動ユーティリティーで使用できるファイル形式については、225ページの『付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

PC/IXF ファイル形式

PC/IXF は、プラットフォーム間でデータを転送する際に望ましいファイル形式です。 PC/IXF を使うと、ロード・ユーティリティーやインポート・ユーティリティーは、普通はマシンによって異なる数値データをマシンから独立した形で処理できます。たとえば、Intel とその他のハードウェア体系とでは、数値データの保管および処理の方法が異なります。

DB2 ファミリー全製品で PC/IXF ファイルの互換性を保つために、エクスポート・ユーティリティーは Intel 形式の数値データによるファイルを作成し、インポート・ユーティリティーはこの形式でデータを受け入れることを想定します。

ハードウェア・プラットフォームによっては、エクスポートおよびインポート操作中に、DB2 製品はバイト逆転を使用して数値を Intel 形式から非 Intel 形式に変換します。

DB2 の UNIX ベースの実装は、エクスポート時に複数パーツの PC/IXF ファイルを作成しません。しかし、DB2 の作成した複数パーツの PC/IXF ファイルをインポートすることはできます。このタイプのファイルをインポートする場合は、すべての部分を同じディレクトリーに入れる必要があります。そうしないならエラーが戻されます。

DB2 エクスポート・ユーティリティーの UNIX ベースの実装によって作成された単一部分の PC/IXF ファイルは、DB2 (OS/2 版) および DB2 (Windows NT 版) にインポートできます。

区切り付き ASCII (DEL) ファイル形式

DEL ファイルは、それらが作成されたオペレーティング・システムによって異なっています。相違点は次のとおりです。

- 行区切り文字
  - UNIX ベースのテキスト・ファイルでは、改行 (LF) 文字を使用します。
  - 非 UNIX ベースのテキスト・ファイルでは、復帰 / 改行 (CRLF) シーケンスを使用します。
- ファイル終わり文字
  - UNIX ベースのテキスト・ファイルでは、ファイル終わり文字を使用しません。
  - 非 UNIX ベースのテキスト・ファイルでは、ファイル終わり文字 (X'1A') を使用します。

DEL エクスポート・ファイルはテキスト・ファイルなので、あるオペレーティング・システムから別のオペレーティング・システムに転送できます。テキスト・モードでファイルを転送する場合、ファイル転送プログラムを使うと、オペレーティング・システムによる違いを処理できます。バイナリー・モードの場合、行区切り文字やファイル終わり文字の変換は実行されません。

注: 文字データ・フィールドに行区切り文字が含まれていれば、それらもまたファイル転送中に変換されます。そのような変換によって予期しないデータの変更が発生するため、プラットフォームを超えてデータを移動する場合は DEL エクスポート・ファイルを使わないことをお勧めします。その代わりに、PC/IXF ファイル形式を使用してください。

WSF ファイル形式

WSF 形式ファイルの数値データは、Intel 機械形式を使って保管されます。この形式では、異なるロータス操作環境 (たとえば Intel ベースと UNIX ベースのシステム) でロータスの WSF ファイルを転送および使用できます。

このように内部形式に整合性があるため、DB2 製品からエクスポートした WSF ファイルを、別のプラットフォームで実行しているロータス 1-2-3 または Lotus Symphony で使用することができます。また、DB2 製品は別のプラットフォームで作成された WSF をインポートできます。

オペレーティング・システム間の WSF ファイルの転送は、テキスト・モードではなくバイナリー・モードで実行してください。

注: 異なるプラットフォームの DB2 データベース間のデータ転送には WSF ファイル形式を使わないでください。そのようにすると、データが失われる可能性があります。その代わりに、PC/IXF ファイル形式を使用してください。

db2move ツールを使用したデータの移動

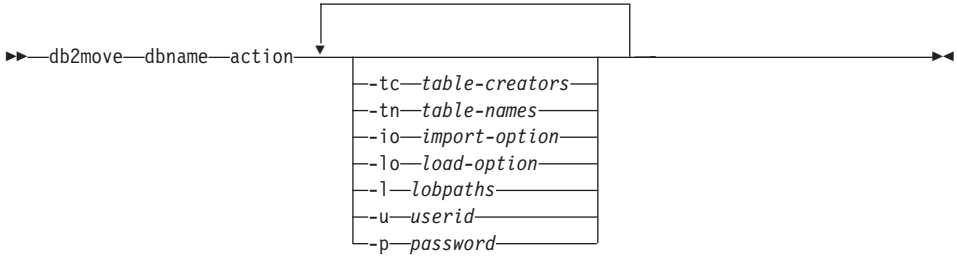
このツールを使用すると、たくさんの表を複数のワークステーション上の DB2 データベース間で容易に移動できます。 このツールは、システム・カタログ表から特定のデータベースを問い合わせたり、すべてのユーザー表からなるリストをコンパイルしたりします。 その後、これらの表を PC/IXF 形式でエクスポートします。 PC/IXF ファイルは、同じシステム上の別のローカル DB2 データベースにインポートまたはロードできます。または、別のワークステーション・プラットフォームに転送してから、そのプラットフォームの DB2 データベースにインポートまたはロードできます。

注: このツールを使用しても、構造タイプ列をもつ表は移動されません。

許可

ユーザーの要求するアクションに応じて、このツールは DB2 のエクスポート、インポート、およびロード API を呼び出します。したがって、要求を出しているユーザー ID には、それら API に必要な適切な許可が必要です。許可がない場合は、要求は失敗します。

コマンド構文



コマンド・パラメーター

dbname
データベースの名前。

action EXPORT、IMPORT、または LOAD のうちいずれか 1 つ。

-tc 表作成者。デフォルトはすべての作成者です。

これは EXPORT アクションのみです。このオプションを指定すると、ここに示す作成者の作成した表だけがエクスポートされます。指定しない場合は、デフォルトとしてすべての作成者を使用します。複数の作成者を指定する場合は、それぞれの作成者をコンマで区切る必要がありま

す。作成者 ID と作成者 ID の間にブランクを入れないようにしてください。指定できる作成者の最大数は 10 です。このオプションを表名オプション 『-tn』 と併用することにより、エクスポートする表を選択できます。

文字ストリング内の任意の場所で、ワイルドカード文字としてアスタリスク (*) を使用できます。

-tn 表の名前。デフォルトはすべてのユーザー表です。

これは EXPORT アクションのみです。このオプションを指定すると、指定されたストリングの中の名前と正確に一致する表だけがエクスポートされます。指定しない場合は、デフォルトとしてすべてのユーザー表を使用します。複数の表名を指定する場合は、それぞれの表名をコンマで区切る必要があります。表名と表名の間にブランクを入れないようにしてください。指定できる表名の最大数は 10 です。このオプションを表作成者オプション 『-tc』 と併用することにより、エクスポートする表を選択できます。 **db2move** は、指定された表名と名前が一致し、かつ指定された表作成者と作成者が同じ表だけをエクスポートします。

文字ストリング内の任意の場所で、ワイルドカード文字としてアスタリスク (*) を使用できます。

-io インポート・オプション。デフォルトは REPLACE_CREATE です。

有効なオプションは、 INSERT、 INSERT_UPDATE、 REPLACE、 CREATE、および REPLACE_CREATE です。

-lo ロード・オプション。デフォルトは INSERT です。

有効なオプションは INSERT と REPLACE です。

-l lobpath。デフォルトは現行ディレクトリーです。

このオプションは、EXPORT の一部として LOB ファイルを作成したり、IMPORT や LOAD の一部として LOB ファイルを検索したりする場所の絶対パス名を指定します。複数の LOB パスを指定する場合は、それぞれの LOB パスをコンマで区切る必要があります。LOB パスと LOB パスの間にブランクを入れてはなりません。 EXPORT 中に最初のパスのスペースが不足した場合、または IMPORT や LOAD 中にファイルがみつからない場合は、2 番目以降のパスを順次使用します。

EXPORT アクションの場合、LOB パスが指定されているなら、LOB パス・ディレクトリー内のファイルはすべて削除され、ディレクトリー

は除去されて、新しいディレクトリーが作成されます。これを指定しない場合、現行ディレクトリーが LOB パスとして使用されます。

-u userid (ユーザー ID)。デフォルトはログオンしているユーザー ID です。

ユーザー ID とパスワードはどちらもオプションです。ただし、どちらか一方を指定すると、もう一方も指定しなければなりません。リモート・サーバーに接続しているクライアント上でコマンドを実行している場合は、ユーザー ID とパスワードを指定する必要があります。

-p パスワード。デフォルトはログオンしているパスワードです。

ユーザー ID とパスワードはどちらもオプションです。ただし、どちらか一方を指定すると、もう一方も指定しなければなりません。リモート・サーバーに接続しているクライアント上でコマンドを実行している場合は、ユーザー ID とパスワードを指定する必要があります。

例

- `db2move sample export`

  これは、すべての表を SAMPLE データベースにエクスポートします。すべてのオプションにデフォルト値が使われます。

- `db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2`

  これは、ユーザー ID 『userid1』 またはたとえば 『us%rid2』 によって作成された表で、表名が 『tbname1』 またはたとえば 『%tbname2』 である表を、すべてエクスポートします。

- `db2move sample import -l D:¥LOBPATH1,C:¥LOBPATH2`

  この例は、OS/2 または Windows オペレーティング・システムのみに当てはまります。このコマンドは、SAMPLE データベース内のすべての表をインポートします。 LOB パス 『D:¥LOBPATH1』 および 『C:¥LOBPATH2』 から LOB ファイルを検索します。

- `db2move sample load -l /home/userid/lobpath,/tmp`

  この例は UNIX ベースのシステムのみに当てはまります。このコマンドによって、SAMPLE データベース内のすべての表をロードします。サブディレクトリー `/home/userid/lobpath` および `tmp` の中から LOB ファイルを検索します。

- `db2move sample import -io replace -u userid -p password`

  これは、SAMPLE データベース内のすべての表を REPLACE モードでインポートします。指定されたユーザー ID とパスワードが使われます。

使用上の注意

このツールは、ユーザー作成の表をエクスポート、インポート、またはロードします。あるデータベースを、1 つのオペレーティング・システムから別のオペレーティング・システムへ複製する際には、 **db2move** を使用すると表の移動が容易になります。また、表に関連付けられた他のオブジェクト (別名、ビュー、トリガー、ユーザー定義関数など) をすべて移動する必要があります。 **db2look** (DB2 統計および DDL 抽出ツール、 *コマンド解説書* を参照) を利用すると、データ定義言語 (DDL) ステートメントをデータベースから取り出すことによって、いくつかのオブジェクトの移動が容易になります。

エクスポート、インポート、またはロード API が **db2move** によって呼び出されると、`FileTypeMod` パラメーターは `lobsinfile` に設定されます。つまり、LOB データは PC/IXF ファイルとは別個のファイルの中に入れられます。 LOB ファイルの名前に使用できるファイル名は 26 000 個あります。

LOAD アクションは、データベースとデータ・ファイルがあるマシン上でローカルに実行しなければなりません。ロード API が **db2move** によって呼び出されると、`CopyTargetList` パラメーターはヌルに設定され、コピーは実行されません。 *logretain* がオンに設定されていると、後でロード操作をロールフォワードすることはできません。ロードされる表が入れられる表スペースはバックアップ保留状態になり、アクセスできなくなります。データベース全体のバックアップ、または個々の表スペースのバックアップを取るには、表スペースをバックアップ保留状態から解除する必要があります。

バージョン 5.2 のクライアントからバージョン 6 のデータベースに対して発行される場合、このツールは、長さが 18 文字より長い表名または列名をサポートしません。

EXPORT 使用時に必要なファイルと生成されるファイル

- 入力: なし。
- 出力:

EXPORT.out EXPORT アクションの結果の要約。

db2move.lst 元の表名、および対応する PC/IXF ファイル名 (tabnnn.ixf) とメッセージ・ファイル名 (tabnnn.msg) のリスト。このリストに加えて、エクスポートされた PC/IXF ファイル、および LOB ファイル (tabnnnc.yyy) が、**db2move** IMPORT または LOAD アクションの入力として使われます。

tabnnn.ixf 特定の表からエクスポートした PC/IXF ファイル。

tabnnn.msg 対応する表のエクスポート・メッセージ・ファイル。

tabnnnc.yyy 特定の表からエクスポートした LOB ファイル。

『nnn』 は表番号。 『c』 は英字。 『yyy』 は 001～999 の数

これらのファイルが作成されるのは、エクスポートする表に LOB データが含まれている場合だけです。それらの LOB ファイルが作成される場合は、『lobpath』 ディレクトリーに入れられます。 LOB ファイルに使用できる名前は全部で 26 000 個あります。

system.msg ファイルやディレクトリーの作成または削除コマンドに対するシステム・メッセージを入れるメッセージ・ファイル。これが使われるのは、アクションが EXPORT で、LOB パスが指定されている場合だけです。

IMPORT 使用時に必要なファイルと生成されるファイル

- 入力:

  db2move.lst EXPORT アクションからの出力ファイル。

  tabnnn.ixf EXPORT アクションからの出力ファイル。

  tabnnnc.yyy EXPORT アクションからの出力ファイル。

- 出力:

  IMPORT.out IMPORT アクションの結果の要約。

  tabnnn.msg 対応する表のインポート・メッセージ・ファイル。

LOAD 使用時に必要なファイルと生成されるファイル

- 入力:

  db2move.lst EXPORT アクションからの出力ファイル。

  tabnnn.ixf EXPORT アクションからの出力ファイル。

  tabnnnc.yyy EXPORT アクションからの出力ファイル。

- 出力:

  LOAD.out LOAD アクションの結果の要約。

  tabnnn.msg 対応する表のロード・メッセージ・ファイル。

DB2 コネクトによるデータの移動

ホスト・データベース・システムとワークステーションの間でデータを移動する必要のある複合環境では、 DB2 コネクト (ホストからワークステーションへ、またその逆のデータ転送のゲートウェイ) を使用できます (図9 を参照)。

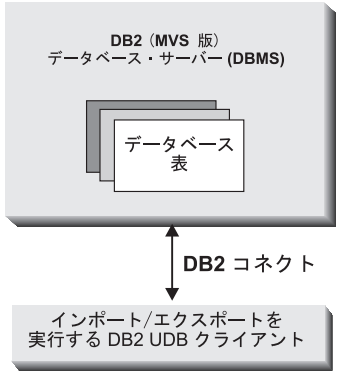


図 9. DB2 コネクトによるインポート / エクスポート

以下の部分では、DB2 コネクトを使ったデータのエクスポートおよびインポートについて説明します。

エクスポート・ユーティリティーおよびインポート・ユーティリティーの使用

DB2 のエクスポートおよびインポート・ユーティリティーを使用すると、DRDA サーバー・データベースから DB2 コネクト ワークステーション上のファイルに、またはその逆にデータを移動できます。その後、このエクスポートおよびインポート形式をサポートしている他のすべてのアプリケーションやリレーショナル・データベース管理システムで、データを使用できます。たとえば、DB2 (MVS/ESA 版) から区切り付き ASCII ファイルにデータをエクスポートして、さらにそれを DB2 (OS/2 版) データベースにインポートすることができます。

エクスポートおよびインポート操作は、データベース・クライアントから、または DB2 コネクト ワークステーションから実行できます。

注:

1. エクスポートまたはインポートされるデータは、両方のデータベースに適用されるサイズとデータ・タイプの制約事項に従っていなければなりません。
2. インポートのパフォーマンスを改善するため、複合 SQL を使用することができます。インポート・ユーティリティーで `compound` ファイル・タイプ修飾子を指定することにより、指定した数の SQL ステートメントをブロック

にまとめてください (60ページの『ファイル・タイプ修飾子 (インポート)』を参照)。このようにすればネットワーク・オーバーヘッドが少なくなり、応答時間が改善されます。

3. エクスポートおよびインポート・ユーティリティーの構文については、8ページの『EXPORT コマンド』、および 38ページの『IMPORT コマンド』を参照してください。

ワークステーションから DRDA サーバーへのデータの移動

DRDA サーバー・データベースにデータを移動するには、次のようにします。

1. DB2 表から PC/IXF ファイルにデータをエクスポートします。
2. INSERT オプションを使って、PC/IXF ファイルを DRDA サーバー・データベース内の互換性のある表にインポートします。

DRDA サーバーからワークステーションへのデータの移動

DRDA サーバー・データベースからデータを移動するには、次のようにします。

1. DRDA サーバー・データベースの表から PC/IXF ファイルにデータをエクスポートします。
2. PC/IXF ファイルを DB2 表にインポートします。

制約事項

DB2 コネクトを使用する場合、エクスポートおよびインポートの操作は次の条件を満たしている必要があります。

- ファイル・タイプは PC/IXF でなければなりません。
- インポート開始前に、データと互換性のある属性の表がすでに存在していなければなりません。 DB2 コネクトによるインポートでは、サポートされているオプションは INSERT だけなので、表は作成できません。
- インポート操作にコミット・カウント・インターバルを指定することはできません。

これらの条件のいずれかが満たされていない場合、操作は失敗し、エラー・メッセージが戻されます。

注: 索引定義はエクスポートにおいて保管されず、インポートにおいて使用されません。

1 バイト・データと 2 バイト・データの混合

混合データ (1 バイト・データと 2 バイト・データの両方を含む列) をエクスポートまたはインポートする場合は、以下のことを考慮してください。

- EBCDIC でデータを保管するシステム (MVS、OS/390、OS/400、VM、および VSE) では、シフトアウトおよびシフトイン文字が 2 バイト・データのそれぞれ開始と終了を表します。データベース表の列の長さを定義する場合は、これらの文字のための十分な余地を見込んでください。
- 列データのパターンが一貫しているのでない限り、文字タイプの可変長列を使用することをお勧めします。

タイプ表間のデータ移動

DB2 エクスポートおよびインポート・ユーティリティーを使用すると、タイプ表からデータを移動したりタイプ表にデータを移動したりできます。タイプ表は、階層になっている場合も可能です。複数の階層にわたるデータ移動には、次のものが含まれることがあります。

- 1 つの階層からそれと同一の階層への移動。
- 1 つの階層から、より大きな階層のサブセクションへの移動。
- 大きな階層のサブセクションから別の階層への移動。

IMPORT CREATE オプションを使用すると、表階層とタイプ階層を作成することができます。

階層内のタイプの識別方法はデータベースによって違います。つまり、同じタイプでもデータベースが異なると識別子が違います。そのため、それらのデータベース間でデータを移動する場合、データを正しく移動するために同じタイプのマッピングを実行する必要があります。

エクスポート操作中に各タイプ行が書き出される前に、識別子が索引値に変換されます。この索引値は、1 から、階層内の関連するタイプの数までの範囲のいずれかの数になります。階層内を特定の順序で移動する場合、それぞれのタイプに番号を付けることによって索引値が生成されます。この順序を*走査順序*といいます。これは、階層内の上位表と副表の全体を上から下へ、あるいは左から右へ進む順序です。階層間でデータを移動する場合、走査順序は他のデータとの相対関係でデータがどこに移動するかを決めるものとなるため、これは重要です。

1 つの方法は、階層の最上部 (ルート表) から下の階層 (副表) に進んで最も低い副表に達します。それからその上位の表に戻って、次の 『右端の』 副表まで達したなら、またその上位表のさらに上に戻って、その副表を下っていき、このような過程を続けます。

次の図は、以下のような 4 つの有効な走査順序のある階層を示しています。

- Person、Employee、Manager、Architect、Student。
- Person、Student、Employee、Manager、Architect (この走査順序は破線で示されています)。
- Person、Employee、Architect、Manager、Student。
- Person、Student、Employee、Architect、Manager。

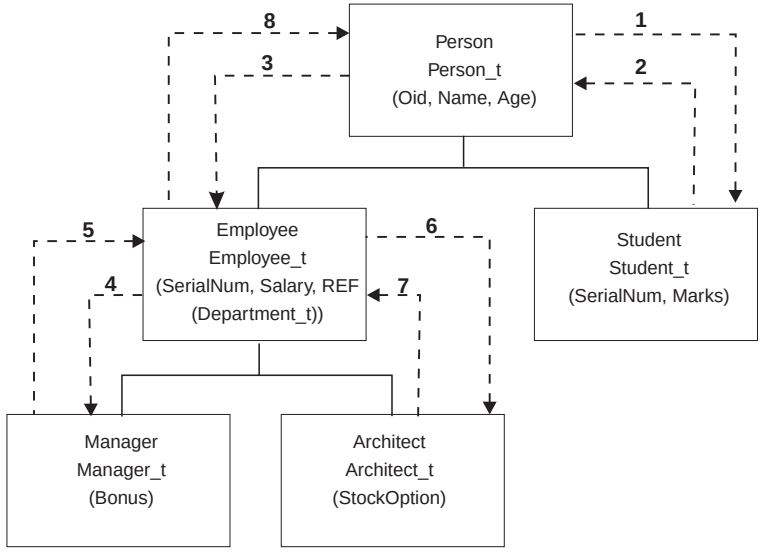


図 10.

走査順序

デフォルトの走査順序があります。デフォルトでは、関連するすべてのタイプが、階層内の特定の開始点から到達可能な階層内のタイプをすべて参照します。デフォルト順序には階層内のすべての表が含まれており、それぞれの表の順序は OUTER 順序述部で使われている方式に従ったものです。また、ユーザー指定の走査順序もあります。これは、関連のあるタイプをユーザーが走査順序リストで定義します。エクスポート・ユーティリティーを起動する場合とインポート・ユーティリティーを起動する場合では、同じ走査順序を使用しなければなりません。

走査順を指定する場合は、副表を PRE-ORDER 方式で走査することが必要です。つまり、階層内のそれぞれのブランチの最下位にまで達しないと、新しいブランチの走査を開始できません。

デフォルトの走査順序

ファイル形式が違うと、デフォルト走査順序の動作が違います。これ以降では、表階層とタイプの関係が同一であるとします。

PC/IXF ファイル形式にデータをエクスポートすると、関連のあるすべてのタイプ、それらの定義、および関連している表に関するレコードが作成されます。また、エクスポートによって索引値からそれぞれの表へのマッピングが完了します。インポート中には、このマッピングを使うことによって、宛先データベースへのデータ移動が正確に実行されます。 PC/IXF ファイル形式で作業をする場合は、デフォルトの走査順序を使用してください。

ASC、DEL、または WSF ファイル形式では、ソースとターゲットの階層の構造が同じであっても、タイプ行およびタイプ表の作成順が違う可能性があります。その場合、デフォルト走査順序で階層を進むうちに時間の差が発生することになります。デフォルト走査順を使用する場合、ソースとターゲットのどちらにおいても、それぞれのタイプの作成日時が階層内の走査の順序を決定します。ソース階層およびターゲット階層の両方で、それぞれのタイプの作成順序が同じであること、およびソースとターゲットの構造が同じであることを確かめてください。この条件が満たされない場合は、ユーザー指定の走査順序を選択してください。

ユーザー指定の走査順序

階層の走査順序を制御したい場合は、エクスポートおよびインポート・ユーティリティーで必ず同じ走査順が使われるようにしてください。以下の場合、

- ソースとターゲットの両方のデータベースで、副表の定義が同じである。
- ソースとターゲットの両方のデータベースで、副表間の階層関係が同じである。
- 走査順序が同じである。

インポート・ユーティリティーは宛先データベースにデータを正確に移動することが保証されます。

走査順を定義する場合、開始点と階層を下るパスを決定できますが、それぞれのブランチの最後まで走査が終わってからでないと、階層内の次のブランチの走査を開始できません。エクスポートおよびインポート・ユーティリティーは、指定された走査順序がこの条件に違反していないかどうかを検査します。

データ移動中の選択

ある階層構造のタイプ表から、別の階層構造の表へのデータ移動は、特定の走査順序により、また中間フラット・ファイルを作成することにより実行されま

す。エクスポート・ユーティリティーは、走査順とともに機能することによってそのファイルの内容を制御します。指定する必要があるのは、ターゲット表の名前と WHERE 文節だけです。エクスポート・ユーティリティーは、これらの選択基準を使用して適切な中間ファイルを作成します。

インポート・ユーティリティーは、ターゲット・データベースに入れられる内容を制御します。それぞれの副表名の最後に属性リストを指定することによって、ターゲット・データベースに移される属性を制限できます。属性リストを使用しない場合は、それぞれの副表のすべての列が移動されます。

インポート・ユーティリティーは、CREATE、INTO table-name、UNDER、および AS ROOT TABLE パラメーターによって、移動する階層のサイズと配置を制御します。 IMPORT コマンド・パラメーターについては、38ページの『IMPORT コマンド』を参照してください。

タイプ表間のデータ移動の例

このセクションの例は、以下のような階層構造に基づいています。

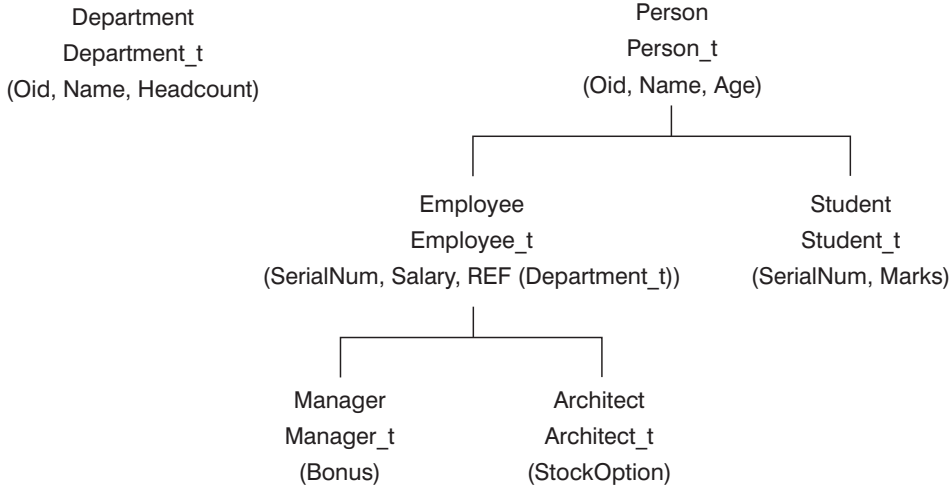


図 11.

例 1

以下のようにして、ある階層の全体をエクスポートして、インポート操作によりその階層を再作成します。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.ixf OF IXF HIERARCHY STARTING Person
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.ixf OF IXF CREATE INTO
   HIERARCHY STARTING Person AS ROOT TABLE
```

階層内の各タイプが存在しない場合は、それが作成されます。タイプがすでに存在している場合は、ターゲット・データベースとソース・データベースとで同じ定義でなければなりません。もし同じでなければ、SQL エラー (SQL20013N) が戻されます。ここでは新たに階層を作成しているので、ターゲット・データベース (Target_db) に移動されるデータ・ファイルで定義されている副表は存在しません。ソース・データベース階層のすべての表が新たに作成されます。ソース・データベースから移されるデータは、ターゲット・データベース内の適切な副表にインポートされます。

例 2

もっと複雑な例として、ソース・データベースの階層の全体をエクスポートして、それをターゲット・データベースにインポートすることにします。年齢が 20 歳以上の人の全データをエクスポートしますが、ターゲット・データベースにインポートするのは選択したデータだけです。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.del OF DEL HIERARCHY (Person,
   Employee, Manager, Architect, Student) WHERE Age>=20
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.del OF DEL INSERT INTO (Person,
   Employee(Salary), Architect) IN HIERARCHY (Person, Employee,
   Manager, Architect, Student)
```

ターゲット表 Person、Employee、および Architect はすでに存在していなければなりません。データは副表 Person、Employee、および Architect の中にインポートされます。インポートされるデータは次のとおりです。

- Person 内のすべての列を Person へ
- Person のすべての列、および Employee 内の Salary を Employee へ
- Person のすべての列、Employee 内の Salary、および Architect のすべての列を Architect へ

SerialNum 列および REF(Employee_t) 列は、Employee とその副表 (データがインポートされる唯一の副表である Architect) にインポートされません。

注: Architect は Employee の副表であり、Employee に指定されている唯一のインポート列が Salary であるため、 Architect にインポートされる

Employee 固有の列は Salary だけになります。つまり、SerialNum 列と REF(Employee_t) 列はいずれも、Employee 行や Architect 行にインポートされません。

表 Manager と表 Student のデータはインポートされません。

例 3

この例では、正規の表からエクスポートした後、ある階層内の 1 つの副表としてインポートします。 EXPORT コマンドが正規の表 (タイプ表でない表) で実行されるので、データ・ファイルの中に Type_id 列はありません。修飾子 no_type_id がこのことを示すのに使用され、それによってインポート・ユーティリティーは最初の列が Type_id 列であることを想定しません。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO Student_sub_table.del OF DEL SELECT * FROM
   Regular_Student
DB2 CONNECT TO Target_db
DB2 IMPORT FROM Student_sub_table.del OF DEL METHOD P(1,2,3,5,4)
   MODIFIED BY NO_TYPE_ID INSERT INTO HIERARCHY (Student)
```

この例では、ターゲット表 Student がすでに存在していなければなりません。Student は副表であるため、最初の列に Type_id が存在しないことを示すために修飾子 no_type_id が使用されています。しかし、Object_id 列、および Student 表内の他のすべての属性が存在することを確認する必要があります。Student 表にインポートされるすべての行について、その最初の列が Object-id であることが想定されています。 METHOD 文節によって、最後の 2 つの属性の順序が逆転します。

レプリケーションを使ったデータの移動

レプリケーションによって、複数のリモート・データベースに定期的にデータをコピーすることができます。マスター・データベースに対する更新を自動的に他のデータベースにコピーすることが必要なら、DB2 のレプリケーション機能を利用することによって、コピーするデータの選択、コピー先データベース表の選択、および更新内容をコピーする頻度を指定できます。 DB2 のレプリケーション機能は、小規模から大規模までの企業のデータ・レプリケーション用に IBM が提供する広範囲にわたるソリューションの一部を成します。

IBM レプリケーション・ツールは、DB2 DataPropagator (DPROP) と DB2 ユニバーサル・データベース・ツールとて構成されるセットです。これは、次の分散リレーショナル・データベース管理システム間でデータをコピーします。

- DB2 ユニバーサル・データベース・プラットフォーム間

- DB2 ユニバーサル・データベース・プラットフォームと、分散リレーショナル・データベース体系 (DRDA) 接続性をサポートするホスト・データベースとの間
- DRDA 接続性をサポートするホスト・データベース間

DB2 DataJoiner を使えば、 IBM 以外のリレーショナル・データベース管理システムにデータをレプリケートすることもできます。

IBM レプリケーション・ツールを利用すると、1 つの制御点から企業全体のデータのコピー操作を定義、同期化、自動化、および管理できます。 DB2 ユニバーサル・データベースのレプリケーション・ツールでは、リレーショナル・データベース間でレプリケーションを行うことができます。またこのツールを IMS DataPropagator (旧称 DPropNR) と一緒に使うと、 IMS および VSAM データをレプリケートでき、Lotus NotesPump と一緒に使うと、 Lotus Notes データベースとの間でレプリケーションを行うことができます。

レプリケーションによってエンド・ユーザーやアプリケーションは、実動データベースに余分の負荷をかけずに実動データを利用できます。ユーザーやアプリケーションのローカル・データベースにデータをコピーすることにより、リモートからデータにアクセスする必要はありません。レプリケーションの典型的な例では、ソース表のコピーが 1 つまたは複数のリモート・データベースに存在します (たとえば銀行の本店と地方の各支店)。事前に決定されたタイミングで DB2 データベースの自動更新が開始し、ソース・データベースの変更内容がすべてターゲット・データベース表にコピーされます。

レプリケーション・ツールでは、コピー表の構造をカスタマイズできます。ターゲット・データベースへのコピー中に、SQL を使用してデータを拡張することができます。また、ソース表の複製となる読み取り専用コピーの作成、指定した日時におけるデータ取り込み、変更履歴の提供、および追加のターゲット表にコピーするデータのステージングが可能です。さらに、エンド・ユーザーやアプリケーションが更新できる読み取り / 書き込みコピーを作成して、変更内容をマスター表に複写することもできます。ソース表のビューやコピーのビューのレプリケーションも可能です。また、イベント・ドリブン・レプリケーションも使用できます。

DB2 データベースの間で相互レプリケーションが可能なプラットフォームは、AIX、AS/400、HP-UX、Linux、Microsoft Windows (95、98、2000、および NT)、OS/2、OS/390、SCO UnixWare、Sun Solaris、Sequent、VM、および VSE です。また、DB2 DataJoiner を使い、DB2 と、 Informix、Microsoft Jet、Microsoft SQL Server、Oracle、Sybase、および Sybase SQLAnywhere のそれぞれの非 DB2 データベースとの間でレプリケーションを行うことができます。

他の IBM 製品と一緒に使えば、 IMS、VSAM、または Lotus Notes との間で DB2 データのレプリケーションを行うことができます。最後に、Windows CE、または Palm OS 装置上の DB2 Everywhere にデータをレプリケートすることもできます。

IBM レプリケーション・ツールの構成要素

IBM レプリケーション・ツール・ソリューションの構成要素には、収集プログラムと変更適用プログラムの 2 つがあります。 DB2 コントロール・センターを利用してこれらの構成要素をセットアップします。これらの構成要素による操作とモニターは、コントロール・センターでは制御されません。

IBM 収集プログラムは、ソース表に加えられた変更内容を取り込みます。使用できるソース表は次のとおりです。

- DB2 DataPropagator の外部にロードされたファイル・システムまたは非リレーショナル・データベース・マネージャーからの SQL データの入った外部表
- データベース内の既存の表
- 以前に変更適用プログラムで更新した表。この場合、変更内容をソースにコピーしたり、他のターゲット表にコピーしたりすることができます。

変更内容は変更データ表にコピーされ、ターゲット・システムでコピーの用意ができるまでそこに保管されます。変更適用プログラムは変更データ表から変更内容を取り出して、ターゲット表にコピーします。

コントロール・センターは、次のことのために使います。

- レプリケーション環境の設定。
- ソース表およびターゲット表の定義。
- 自動コピーの時間指定。
- データの SQL 拡張の指定。
- ソース表とターゲット表の関係の定義。

詳しくは、*レプリケーションの手引きおよび解説書* を参照してください。

データウェアハウスセンターによるデータの移動

データウェアハウスセンター (DWC) を使用すれば、操作可能データベースからウェアハウス・データベースにデータを移動し、意思決定時に照会することができます。また、DWC を使うと、ソース と呼ばれる操作可能データベースの構造を定義することもできます。このとき、操作可能データをウェアハウス

用に移動および変換する方法を指定することができます。ターゲット と呼ばれるウェアハウス・データベース内の表の構造をモデル化するか、データ移動操作の定義プロセスの一環として自動的に表を作成することができます。

データウェアハウスセンターは、次のような DB2 機能を使ってデータの移動と変換を行います。

- SQL

  SQL を使うと、ソースからデータを選択し、そのデータをターゲットに挿入することができます。また、データをウェアハウス形式に変換することもできます。データウェアハウスセンターでは、SQL を生成したり、独自の SQL を作成したりすることができます。

- ロード・ユーティリティーとエクスポート・ユーティリティー

  これらの DB2 ユーティリティーを使うと、データをソースからエクスポートしてから、そのデータをターゲットにロードすることができます。これらのユーティリティーが役立つのは、大量のデータを移動する必要がある場合です。データウェアハウスセンターは、次のようなタイプのロードおよびエクスポート操作をサポートします。

  DB2 データのエクスポート
  ローカル DB2 データベースから、区切り付きファイルへデータをエクスポートします。

  ODBC データのエクスポート
  ODBC に登録されているデータベース内の表からデータを選択してから、区切り付きファイルにそのデータを書き込みます。

  DB2 ロード
  区切り付きファイルのデータを DB2 表にロードします。

  DB2 UDB EEE データベースへの DB2 のロード (AIX のみ)
  区切り付きファイルのデータを、 DB2 ユニバーサル・データベース エンタープライズ拡張エディションのデータベースにロードし、表内の既存データを新規のデータに置き換えます。この操作では、データベース用のターゲット区分化マップが獲得され、各ファイルがデータベース区画にロードできるように入力ファイルが分割され、すべての区画でリモート・ロード操作が実行されます。

- レプリケーション

  また、レプリケーションを使って、大量のデータをウェアハウス・ソースからウェアハウス・ターゲットにコピーしてから、ソース・データのその後のすべての変更を取り込むこともできます。データウェアハウスセンターは、次のようなタイプのレプリケーションをサポートします。

基本集約
ユーザー表用の集約データが入っていて、しかも指定間隔で追加が行われるターゲット表を作成します。

変更集約
集約データが入っていて、しかもソース表について記録された変更に基づいたターゲット表を作成します。

時刻指定
ソース表に一致するターゲット表を作成し、タイム・スタンプ列をターゲット表に追加します。

ステージング・テーブル
複数のターゲット表に対する更新データのソースとして使える「一貫した変更データ」表を作成します。

ユーザー・コピー
コピーの作成時のソース表に一致するターゲット表を作成します。

上記の操作は、すべての DB2 ユニバーサル・データベース・ワークステーションの操作環境、 DB2 ユニバーサル・データベース (OS/390 版)、DB2 (AS/400 版)、および DataJoiner でサポートされています。

- 変換機能ストアード・プロシージャー

  データウェアハウスセンターを使えば、データを OLAP (オンライン分析処理) データベースに移動することができます。データをウェアハウスに入れた後、変換機能ストアード・プロシージャーを使ってデータをクリーンアップしてから、それを集約してファクト表および次元表にすることができます。また、変換機能を使って統計データを生成することもできます。データのクリーンアップと変換が完了したら、それを OLAP キューブにロードするか、部門別に分けられたサーバーにレプリケートすることができます。このようなサーバーは、データマート と呼ばれることがあります。変換機能は、DB2 製品の特定のものにしか組み込まれていません。詳しくは、IBM 担当員にお問い合わせください。

データウェアハウスセンターの詳細は、データウェアハウスセンター *管理の手引き* を参照してください。

付録A. 構文図の読み方

構文図は、入力の内容がオペレーティング・システムによって正しく解釈されるようにコマンドを指定する方法を示すものです。

構文図は、水平線 (主線) に沿って左から右、上から下へ読みます。線が矢印で終わっている場合、コマンド構文は継続しており、次の線は矢印で始まります。垂直線は、コマンド構文の終わりを示します。

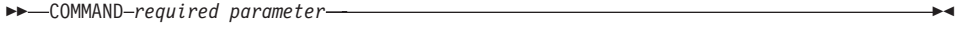
構文図から情報を入力する際には、引用符や等号などの句読点を必ず含めるようにしてください。

パラメーターは、キーワードと変数に分類されます。

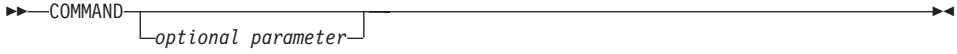
- キーワードは定数を表し、大文字で示されます。実際のコマンド・プロンプトでは、キーワードを大文字、小文字、または大文字と小文字の混合で入力することができます。キーワードの例としてはコマンド名があります。
- 変数は、ユーザーによって提供される名前または値を表し、小文字で示されます。実際のコマンド・プロンプトでは、変数を大文字、小文字、または大文字と小文字の混合で入力することができます (大文字小文字の制限が明示されているのでない限り)。変数の例としては、ファイル名があります。

1 つのパラメーターがキーワードと変数の組み合わせである場合があります。

必須パラメーターは、主線と同じ高さに示されます。

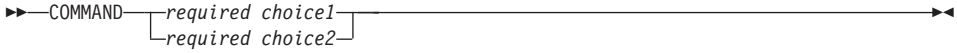


オプション・パラメーターは、主線の下側に示されます。



パラメーターのデフォルト値は、線の上側に示されます。

パラメーターのスタックで、最初のパラメーターが主線と同じ高さに示されている場合は、パラメーターの 1 つを必ず選択しなければならないことを示します。



パラメーターのスタックで、最初のパラメーターが主線の下側に示されている場合は、パラメーターの 1 つを選択可能であることを示します。

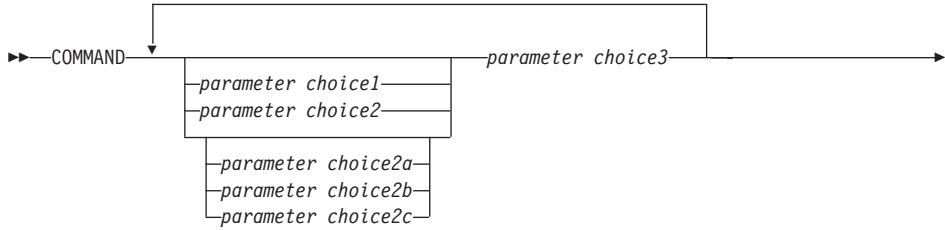
線の上側で左に戻る矢印がある場合は、項目が以下の規則に従って繰り返し可能であることを示します。

- 矢印が中断されていない場合、項目はブランク・スペースによって区切られたリストの形式で繰り返すことができます。

- 矢印にコンマが含まれている場合、項目はコンマによって区切られたリストの形式で繰り返すことができます。

パラメーター・スタックの項目は、前述の必須およびオプション・パラメーターのスタック規則に従って繰り返すことができます。

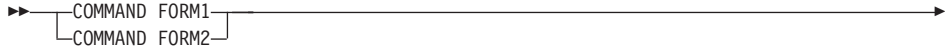
一部の構文図では、パラメーター・スタックが他のパラメーター・スタックの中に含まれていることがあります。スタックの項目を繰り返すには、前述の規則に従わなければなりません。つまり、内側のスタックの上に繰り返し矢印がなく、外側のスタックの上にはある場合には、内側のスタックから 1 つのパラメーターのみを選択し、外側のスタックからの任意のパラメーターと組み合わせて、その組み合わせを繰り返すことができます。たとえば、次の図では、パラメーター *choice2a* をパラメーター *choice2* と組み合わせて、その組み合わせ (*choice2* と *choice2a*) を繰り返すことができます。



一部のコマンドの前には、オプションのパス・パラメーターが付けられています。

このパラメーターを指定しないなら、システムはコマンドを現行ディレクトリーから検索します。コマンドがみつからない場合、システムは、.profile にリストされているすべてのディレクトリーからコマンドの検索を続行します。

一部のコマンドには、機能的に同等な構文上の変形があります。



付録B. インポート・ユーティリティーとロード・ユーティリティーの相違点

下記の表に、DB2 ロード・ユーティリティーとインポート・ユーティリティーの主要な相違点を要約します。

インポート・ユーティリティー	ロード・ユーティリティー
大量のデータを移動する場合、処理速度が遅くなる。	大量のデータを移動する場合、インポート・ユーティリティーより処理が速い。ロード・ユーティリティーは、形式設定されたページをデータベースに直接書き込むためです。
区画内並列性の利用が制限される。	区画内並列性を利用する。一般にこれには対称マルチプロセッサー (SMP) マシンが必要です。
FASTPARSE サポートなし。	FASTPARSE サポートによって、ユーザー提供データのデータ・チェックが簡略化される。
CODEPAGE サポートなし。	CODEPAGE サポートによって、ロード操作時に文字データ (および文字で指定された数値データ) が、このコード・ページからデータベースのコード・ページに変換される。
階層データがサポートされる。	階層データはサポートされない。
PC/IXF 形式での表、階層、および索引の作成がサポートされる。	表および索引は存在していなければならない。
要約表へのインポートはサポートされない。	合計表へのインポートがサポートされる。
WSF 形式がサポートされる。	WSF 形式はサポートされない。
BINARYNUMERICS サポートなし。	BINARYNUMERICS サポートがある。
PACKEDDECIMAL サポートなし。	PACKEDDECIMAL サポートがある。
ZONEDDECIMAL サポートなし。	ZONEDDECIMAL サポートがある。
GENERATED ALWAYS として定義された列をオーバーライドできない。	GENERATEDIGNORE および IDENTITYIGNORE ファイル・タイプ修飾子を使って、 GENERATED ALWAYS 列をオーバーライドすることができる。

インポート・ユーティリティーとロード・ユーティリティーの相違点

インポート・ユーティリティー	ロード・ユーティリティー
表および視点へのインポートがサポートされる。	表へのロードのみサポートされる。
表とその索引が属する表スペースは、インポート操作中にオンラインになる。	表とその索引が属する表スペースは、ロード操作中にオフラインになる。
すべての行がログに記録される。	最小限のログ記録が実行される。
トリガー・サポートがある。	トリガー・サポートなし。
インポート操作が中断された場合に、*commitcount* が指定されていたなら、表は使用可能であり、最後の COMMIT までにロードされた行が含まれている。ユーザーはインポート操作を再開するか、表を現状のまま受け入れることができます。	ロード操作が中断された場合に、*savecount* が指定されていたなら、表はロード保留状態のままになる。その表は、ロード操作が再開されるか、ロード終了操作が呼び出されるか、またはロード操作の試行前に作成されたバックアップ・イメージから表スペースが復元されるまで、使用できません。
必要なスペースは、最大の索引のサイズ + 10% とほぼ同じである。このスペースは、データベース内の一時表スペースから取得されます。	必要なスペースは、表に関して定義されているすべての索引のサイズの合計とほぼ同じであり、そのサイズの 2 倍まで大きくなる可能性がある。このスペースは、データベース内の一時スペースから取得されます。
インポート操作中にすべての制約が妥当性検査される。	ロード操作中に固有性は検査されるが、その他のすべての制約は SET INTEGRITY ステートメントを使用して検査しなければならない。
キー値は、インポート操作中に一度に 1 つずつ索引に挿入される。	データがロードされた後で、キー値がソートされ、索引が作成される。
統計情報を更新する必要がある場合は、インポート操作の後で runstats ユーティリティーを実行しなければならない。	表内のすべてのデータを置換する場合は、ロード操作中に統計情報を収集できる。
DB2 Connect によってホスト・データベースへのインポートが可能。	ホスト・データベースへのロードはできない。
インポート・ファイルは、インポート・ユーティリティーが呼び出されるノードになければならない。	区分データベース環境では、ロード・ファイルまたはパイプは、データベースを備えたノードになければならない。区分データベース以外の環境では、ロード・ファイルまたはパイプは、データベースを備えたノード上か、またはロード・ユーティリティーを呼び出すリモート接続クライアント上に置くことができます。

インポート・ユーティリティー	ロード・ユーティリティー
バックアップ・イメージは不要。インポート・ユーティリティーでは SQL の挿入が使用されるため、DB2 によって活動がログに記録され、障害時にそれらの操作を回復するのにバックアップは不要です。	ロード操作中にバックアップ・イメージを作成することができる。

付録C. エクスポート / インポート / ロード・ユーティリティーのファイル形式

ここでは、DB2 のエクスポート、インポート、およびロード・ユーティリティーによってサポートされている 4 種類のオペレーティング・システム・ファイル形式について説明します。

DEL 区切り付き ASCII。さまざまなデータベース・マネージャーおよびファイル・マネージャーの間でのデータ交換に使用されます。このデータ保管の一般的なアプローチでは、列値を分離するために特殊な文字区切り文字が使用されます。

ASC 区切りなし ASCII。位置合わせされた列データを含むフラット・テキスト・ファイルを作成する他のアプリケーションからのデータのインポートまたはロードに使用されます。

PC/IXF

PC バージョンの IXF (統合交換フォーマット)。データベース・マネージャー内でのデータ交換のために望ましい方式です。 PC/IXF は、内部表の外部表記を含むデータベース表の構造化された記述です。

WSF ワークシート形式。ロータス 1-2-3 や Lotus Symphony などの製品とのデータ交換に使用されます。ロード・ユーティリティーでは、このファイル形式はサポートされません。

DEL、WSF、または ASC データ・ファイル形式を使用する場合は、ファイルをインポートする前に、表 (その列名とデータ・タイプを含む) を定義してください。オペレーティング・システム・ファイルのフィールドのデータ・タイプは、データベース表内の対応するデータ・タイプに変換されます。インポート・ユーティリティーは、多少の非互換性問題があるデータを受け入れます。これには、埋め込みまたは切り捨ての可能性を伴って文字データをインポートする場合や、数値データをそれとは異なるタイプの数値フィールドにインポートする場合が含まれます。

PC/IXF データ・ファイル形式を使用する場合、インポート操作の前に表が存在している必要はありません。ユーザー定義の特殊タイプ (UDT) は新しい表列タイプの一部とはならず、基本タイプが使用されます。同じように、PC/IXF データ・ファイル形式にエクスポートする場合、UDT は PC/IXF ファイル内で基本データ・タイプとして保管されます。

区切り付き ASCII (DEL) ファイル形式

区切り付き ASCII (DEL) ファイルは、行および列の区切り文字を含む順次 ASCII ファイルです。各 DEL ファイルは、行と列によって配列されたセル値から構成される ASCII 文字のストリームです。データ・ストリーム内の行は行区切り文字によって区切られ、それぞれの行の中の個々のセル値は列区切り文字によって区切られます。

以下の表は、インポートしたり、またはエクスポート・アクションの結果として生成したりできる DEL ファイルの形式を示します。

```
DEL ファイル ::= 行 1 のデータ || 行区切り文字 ||
                 行 2 のデータ || 行区切り文字 ||
             .
             .
             .
                 行 n のデータ || オプションの行区切り文字

行 i のデータ ::= セル値(i,1) || 列区切り文字 ||
                  セル値(i,2) || 列区切り文字 ||
               .
               .
               .
                  セル値(i,m)

行区切り文字 ::= ASCII 改行シーケンス[a]

列区切り文字 ::= デフォルト値 ASCII コンマ (,)[b]
セル値(i,j) ::= 先行スペース
             || 数値の ASCII 表記
                (整数、10 進数、または浮動小数点数)
             || 区切り文字ストリング
             || 非区切り文字ストリング
             || 後続スペース

非区切り文字ストリング ::= 行区切り文字または列区切り文字以外の
                            文字の集合
区切り文字ストリング ::= 文字ストリング区切り文字 ||
                                外字ストリング ||
                          文字ストリング区切り文字 ||
                                後続ガーベッジ
後続ガーベッジ ::= 行区切り文字または列区切り文字以外の
                   文字の集合
文字ストリング区切り文字 ::= デフォルト値 ASCII 二重引用符記号
                              (")[c]
外字ストリング ::= || NODOUBLEDEL 修飾子が指定されている場合、
                      行区切り文字または文字ストリング区切り文字以外の
                      文字の集合
                   || 文字ストリングが 2 つの連続した
                      文字ストリング区切り文字の一部でない場合、
                      行区切り文字または文字ストリング区切り文字
                      以外の文字の集合
                   || 文字ストリング区切り文字が 2 つの連続した
                      文字ストリング区切り文字の一部ではなく、
                      かつ DELPRIORITYCHAR 修飾子が指定されて
                      いる場合、文字ストリング区切り文字以外の
                      文字の集合
ファイル終わり文字 ::= 16 進 '1A' (OS/2 または Windows オペレーティング・システムのみ)

数値の ASCII 表記[d] ::= オプションの符号 '+' または '-'
    || 1〜31 桁の 10 進数字に、オプションとして小数点をその前、その後、
       または数字と数字の間に入れたもの
    || オプションの指数
```

```
指数 ::= 文字 'E' または 'e'
    || オプションの符号 '+' または '-'
    || 1〜3 桁の数字 (小数点なし)

10 進数字 ::= 文字 '0'、'1'、... '9' のいずれか

小数点 ::= デフォルト値 ASCII ピリオド (.)[e]
```

- [a] レコード区切り文字は、改行文字 (ASCII x0A) であるとみなされます。OS/2 または Windows オペレーティング・システムで生成されるデータでは、復帰 / 改行の 2 バイト標準 (0x0D0A) が使用されることがあります。EBCDIC コード・ページのデータでは、レコード区切り文字として EBCDIC LF 文字 (0x25) を使用します (EBCDIC データは、LOAD コマンドの CODEPAGE オプションを使用してロードできます)。
- [b] 列区切り文字は、COLDEL オプションで指定できます。
- [c] 文字ストリング区切り文字は、CHARDEL オプションで指定できます。

  注: 区切り文字のデフォルトの優先順位は、次のとおりです。

  1. レコード区切り文字
  2. 文字区切り文字
  3. 列区切り文字

  133ページの表8 にある delprioritychar 修飾子の説明も参照してください。
- [d] 数値の ASCII 表記に指数が含まれる場合、それは FLOAT 定数です。小数点が含まれるが指数は含まれない場合、それは DECIMAL 定数です。小数点も指数も含まれない場合、それは INTEGER 定数です。
- [e] 小数点文字は、DECPT オプションで指定できます。

DEL ファイルの例

DEL ファイルの例を下記に示します。各行は改行文字列で終わります (OS/2 または Windows オペレーティング・システムでは、各行は復帰 / 改行文字シーケンスで終わります)。

```
"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78
```

次の例は、区切り文字なし文字ストリングの使用方法を示しています。文字データにコンマが含まれているため、列区切り文字はセミコロンに変更されています。

```
Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78
```

注:

1. スペース (X'20') は有効な区切り文字ではありません。
2. セル値の最初の文字の前または最後の文字の後にあるスペースは、インポート時に破棄されます。セル値の途中にあるスペースは破棄されません。
3. ピリオド (.) は、タイム・スタンプ値のピリオドと対立するため、有効な文字ストリング区切り文字ではありません。
4. DBCS のみ (グラフィック)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00〜x3F に制限されます。
5. EBCDIC コード・ページで指定された DEL データの場合、区切り文字は DBCS のシフトイン文字およびシフトアウト文字と同じであってはなりません。
6. OS/2 または Windows オペレーティング・システムの場合、文字区切り文字の中にない最初のファイル終わり文字 (X'1A') がファイル終わりを示します。それ以降のデータはインポートされません。
7. ヌル値は、通常はセル値があるはずの場所にセル値がないことによって、あるいはスペースのストリングによって示されます。
8. 一部の製品では文字フィールドが 254 または 255 バイトに制限されるため、エクスポート・ユーティリティーは、最大長が 254 バイトより長い文字タイプの列がエクスポート用に選択されるたびに警告メッセージを生成します。インポート・ユーティリティーは、最も長い LONG VARCHAR および LONG VARGRAPHIC 列と同じ長さのフィールドを受け入れます。

DEL のデータ・タイプの説明

表 9. DEL ファイル形式についての受け入れ可能なデータ・タイプの形式

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	-9 223 372 036 854 775 808〜9 223 372 036 854 775 807 の範囲の INTEGER 定数。	-9 223 372 036 854 775 808〜9 223 372 036 854 775 807 の範囲の数値の ASCII 表記。10 進数および浮動小数点数は整数値に切り捨てられます。

表 9. DEL ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BLOB、CLOB	文字区切り文字 (たとえば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
BLOB_FILE、CLOB_FILE	各 BLOB/CLOB 列ごとに文字データが個別のファイルに保管され、そのファイル名は文字区切り文字によって囲まれます。	データが入っているファイルの区切り付き名前または区切りなし名前。
CHAR	文字区切り文字 (たとえば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、必要ならデータベース列の幅に合わせて切り捨てられるか、またはスペース (X'20') が埋められます。
DATE	文字区切り文字なしの *yyyymmdd* (年、月、日)。たとえば、19931029。 あるいは、DATESISO オプションを使用して、すべての日付値が ISO 形式でエクスポートされるように指定することもできます。	ターゲット・データベースの国別コードに一致する ISO 形式の日付値を含む区切り付きまたは区切りなし文字ストリング、あるいは *yyyymmdd* の形式の区切りなし文字ストリング。
DBCLOB (DBCS のみ)	グラフィック・データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
DBCLOB_FILE (DBCS のみ)	各 DBCLOB 列ごとに文字データが個別のファイルに保管され、そのファイル名は文字区切り文字によって囲まれます。	データが入っているファイルの区切り付き名前または区切りなし名前。

表 9. DEL ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
DECIMAL	エクスポートされるフィールドの精度および位取りによる DECIMAL 定数。DECPLUSBLANK オプションを使用すると、正の 10 進値の前に正符号 (+) ではなく、ブランク・スペースが付けられるように指定できます。	フィールドのインポート先のデータベース列の範囲からあふれない数値の ASCII 表記。入力値の小数部の桁数が、データベース列で受け入れ可能なものより多い場合、余分な桁は切り捨てられます。
FLOAT(long)	-10E307～10E307 の範囲の FLOAT 定数。	-10E307～10E307 の範囲の数値の ASCII 表記。
GRAPHIC (DBCS のみ)	グラフィック・データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、必要ならデータベース列の幅に合わせて切り捨てられるか、または 2 バイト・スペース (たとえば X'8140') が埋められます。
INTEGER	-2 147 483 648～2 147 483 647 の範囲の INTEGER 定数。	-2 147 483 648～2 147 483 647 の範囲の数値の ASCII 表記。 10 進数および浮動小数点数は整数値に切り捨てられます。
LONG VARCHAR	文字区切り文字 (たとえば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
LONG VARGRAPHIC (DBCS のみ)	グラフィック・データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。

表 9. DEL ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
SMALLINT	-32 768～32 767 の範囲の INTEGER 定数。	-32 768～32 767 の範囲の数値の ASCII 表記。 10 進数および浮動小数点数は整数値に切り捨てられます。
TIME	*hh.mm.ss* (時、分、秒)。文字区切り文字によって囲まれた ISO 形式の時刻値。たとえば “09.39.43”	ターゲット・データベースの国別コードに合致する形式の時刻値を含む区切り付きまたは区切りなし文字ストリング。
TIMESTAMP	*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒)。文字区切り文字によって囲まれた日時を表す文字ストリング。	データベースでの保管用に受け入れ可能なタイム・スタンプ値を含む区切り付きまたは区切りなし文字ストリング。
VARCHAR	文字区切り文字 (たとえば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、必要ならデータベース列の最大幅に合わせて切り捨てられます。
VARGRAPHIC (DBCS のみ)	グラフィック・データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、必要ならデータベース列の最大幅に合わせて切り捨てられます。

区切りなし ASCII (ASC) ファイル形式

区切りなし ASCII (ASC) ファイルは、行区切り文字を含む順次 ASCII ファイルです。このファイルは、ワード・プロセッサーを含め、縦欄形式のデータを使用する ASCII 製品とのデータ交換に使用することができます。各 ASC ファイルは、行と列によって配列されたセル値から構成される ASCII 文字のストリームです。データ・ストリーム内の行は行区切り文字によって区切られます。行内の各列は、開始 / 終了位置の対 (IMPORT パラメーターで指定される) によって定義されます。それぞれの対は、バイト位置として指定された行

内の位置を表します。行内の最初の位置はバイト位置 1 です。それぞれの対の最初の要素は列の開始バイト、2 番目の要素は列の終了バイトです。列が互いに重なり合うことも可能です。 1 つの ASC ファイル内の各行の列定義はすべて同じです。

ASC ファイルの定義は、次のとおりです。

```
ASC ファイル ::= 行 1 のデータ || 行区切り文字 ||
                 行 2 のデータ || 行区切り文字 ||
             .
             .
             .
                 行 n のデータ

行 i のデータ ::= ASCII 文字 || 行区切り文字

行区切り文字 ::= ASCII 改行シーケンス[a]
```

- [a] レコード区切り文字は、改行文字 (ASCII x0A) であるとみなされます。OS/2 または Windows オペレーティング・システムで生成されるデータでは、復帰 / 改行の 2 バイト標準 (0x0D0A) が使用されることがあります。EBCDIC コード・ページのデータでは、レコード区切り文字として EBCDIC LF 文字 (0x25) を使用します (EBCDIC データは、LOAD コマンドの CODEPAGE オプションを使用してロードできます)。レコード区切り文字がデータのフィールドの一部として解釈されることはありません。

ASC ファイルの例

ASC ファイルの例を下記に示します。各行は改行文字列で終わります (OS/2 または Windows オペレーティング・システムでは、各行は復帰 / 改行文字シーケンスで終わります)。

```
Smith, Bob       4973     15.46
Jones, Suzanne   12345    16.34
Williams, Sam    452123  193.78
```

注:

1. ASC ファイルには、列名が含まれないものとみなされます。
2. 文字ストリングは、区切り文字によって囲まれません。 ASC ファイルの列のデータ・タイプは、データベース表のターゲット列のデータ・タイプによって決まります。
3. 次の場合、ヌル可能データベース列にヌルがインポートされます。
   - ブランクのフィールドのターゲットが数値、DATE、TIME、または TIMESTAMP タイプのデータベース列である場合
   - 開始 / 終了位置の対のないフィールドが指定されている場合
   - 0 に等しい開始 / 終了位置の対が指定されている場合

- ある行のデータが短すぎて、ターゲット列にとって有効な値が含まれていない場合
- NULL INDICATORS ロード・オプションが使用されていて、ヌル標識列に N (またはユーザーによって指定されたその他の値) が検出された場合

4. ヌル可能でないターゲット列に数値、DATE、TIME、または TIMESTAMP 列にブランクのフィールドをインポートしようとすると、その行は拒否されます。
5. 入力データにターゲット列との互換性がなく、その列がヌル可能でない場合は、エラーが検出された場所に応じてヌルがインポートされるか、または行が拒否されます。列がヌル可能でない場合は、行が拒否されます。互換性がないことを示すメッセージがメッセージ・ファイルに書き込まれます。

ASC のデータ・タイプの説明

表 10. ASC ファイル形式についての受け入れ可能なデータ・タイプの形式

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -9 223 372 036 854 775 808～ 9 223 372 036 854 775 807 の範囲外の値があれば、その特定の値に関しては拒否されます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であるとみなされます。 3 桁ごとの区切り文字は使用できません。 開始位置と終了位置は、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
BLOB/CLOB	文字のストリング。文字ストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク切り捨て オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
BLOB_FILE、 CLOB_FILE、 DBCLOB_FILE (DBCS のみ)	データが入っているファイルの区切り付き名前または区切りなし名前。
CHAR	文字のストリング。文字ストリングは、必要ならターゲット列の幅に合わせて切り捨てられるか、またはスペースが埋められます。

表 10. ASC ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
DATE	ターゲット・データベースの国別コードに合致する形式の日付値を含む文字ストリング。 開始位置と終了位置は、日付の外部表記の範囲内のフィールド幅になるように指定してください。
DBCLOB (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。
DECIMAL	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。インポート先のデータベース列の範囲外の値があれば、その特定の値に関しては拒否されます。入力値の小数部の桁数が、データベース列の位取りより多い場合、余分な桁は切り捨てられます。コンマ、ピリオド、またはコロンは小数点であるとみなされます。 3 桁ごとの区切り文字は使用できません。 開始位置と終了位置は、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
FLOAT(long)	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。すべての値が有効です。コンマ、ピリオド、またはコロンは小数点であるとみなされます。大文字または小文字の E は、FLOAT 定数の指数の始まりとして受け入れられます。 開始位置と終了位置は、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
GRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられるか、または 2 バイト・スペース (0x8140) が埋められます。

表 10. ASC ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
INTEGER	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -2 147 483 648～2 147 483 647 の範囲外の整数があれば、それに関しては拒否されます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であるとみなされます。 3 桁ごとの区切り文字は使用できません。 開始位置と終了位置は、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
LONG VARCHAR	文字のストリング。文字ストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク切り捨て オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
LONG VARGRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。
SMALLINT	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -32 768～32 767 の範囲外の値があれば、その特定の値に関しては拒否されます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であるとみなされます。 3 桁ごとの区切り文字は使用できません。 開始位置と終了位置は、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
TIME	ターゲット・データベースの国別コードに合致する形式の時間値を含む文字ストリング。 開始位置と終了位置は、時間の外部表記の範囲内のフィールド幅になるように指定してください。
TIMESTAMP	データベースでの保管用に受け入れ可能なタイム・スタンプ値を表す文字ストリング。 開始位置と終了位置は、タイム・スタンプの外部表記の範囲内のフィールド幅になるように指定してください。

表 10. ASC ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
VARCHAR	文字のストリング。文字ストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク切り捨て オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
VARGRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要ならターゲット列の最大長に合わせて右側が切り捨てられます。

PC バージョンの IXF ファイル形式

PC バージョンの IXF (PC/IXF) ファイル形式は、データベース・マネージャーにおいて統合交換フォーマット (IXF) データ交換アーキテクチャーに適応するためのものです。 IXF アーキテクチャーは、リレーショナル・データベースの構造とデータの交換を可能にするために特に設計されたものです。
PC/IXF アーキテクチャーによりデータベース・マネージャーは、データベースのエクスポート時に受け取り側の製品の要件や特性を予測する必要がなくなります。同じように、PC/IXF ファイルをインポートする側の製品で必要なことも、PC/IXF アーキテクチャーを理解するだけになります。ファイルをエクスポートした製品の特性は影響しなくなります。 PC/IXF ファイル・アーキテクチャーは、エクスポート側とインポート側の両方のデータベース・システムからの独立性を保ちます。

IXF アーキテクチャーは、特定のリレーショナル・データベース製品によってサポートされていない一部のタイプを含め、リレーショナル・データ・タイプの豊富なセットをサポートする汎用リレーショナル・データベース交換形式です。 PC/IXF ファイル形式ではこの柔軟性が保たれます。たとえば、PC/IXF アーキテクチャーでは、1 バイト文字セット (SBCS) と 2 バイト文字セット (DBCS) の両方のデータ・タイプがサポートされます。すべての実装ですべての PC/IXF データ・タイプがサポートされるわけではありませんが、制限付きの実装でも、インポート操作において、サポートされないデータ・タイプの検出と後処理が実行されます。

一般に PC/IXF ファイルは、中断なしの一連の可変長レコードから構成されます。ファイルには、次のレコードが入れられます。順序はここに示す順序です。

- レコード・タイプ H の 1 つのヘッダー・レコード

- レコード・タイプ T の 1 つの表レコード
- レコード・タイプ C の複数の列記述子レコード (表の列ごとに 1 つのレコード)
- レコード・タイプ D の複数のデータ・レコード (表の各行が 1 つまたは複数の D レコードによって表現されます)。

PC/IXF ファイルにおいて、H レコードの後であればどこにでもアプリケーション (A) レコードを入れることができます。 PC/IXF ファイルにおいてそれらのレコードは、PC/IXF 形式で定義されていない追加のデータをアプリケーションが PC/IXF ファイルに含めることができるようになっています。 PC/IXF ファイルを読み取るプログラムに、A レコード内のアプリケーション識別子によって暗黙指定されるデータ形式と内容に関する特別の知識がない場合、そのレコードは無視されます。

PC/IXF ファイル内のレコードは、いずれもレコード長標識で始まります。これは、PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の、6 バイトの右寄せ文字表記です (合計レコード・サイズ - 6 バイト)。 PC/IXF ファイルを読むプログラムは、これらのレコード長を使用することにより、現行レコードの終わりと次のレコードの始まりを検出します。 H、T、および C レコードは、それらに定義されているすべてのフィールドを入れるのに十分な大きさでなければならず、当然のこととしてそれらのレコード長フィールドは、それらの実際の長さと一致していなければなりません。しかし、これらのいずれかのレコードの終わりに余分なデータ (たとえば*新しい* フィールド) が追加された場合、 PC/IXF ファイルを読む従来のプログラムは余分のデータを無視し、警告メッセージしか生成しません。ただし、PC/IXF ファイルに書き込むプログラムの場合は、すべての定義済みフィールドを入れるのにちょうど必要な長さの H、T、および C レコードを書き込まなければなりません。

PC/IXF ファイルのレコードは、文字データを含むフィールドで構成されます。インポートおよびエクスポート・ユーティリティーは、ターゲット・データベースの CPGID を使用してこの文字データを解釈します。ただし、2 つの例外があります。

- A レコードの IXFADATA フィールド。

  IXFADATA フィールドに含まれる文字データのコード・ページ環境は、特定の A レコードを作成および処理するアプリケーションによって設定されます。つまり、環境は実装ごとに違います。
- D レコードの IXFDCOLS フィールド。

IXFDCOLS フィールドに含まれる文字データのコード・ページ環境は、特定の列とそのデータを定義する C レコードに含まれる情報によって決まります。

H、T、および C レコード内の数値フィールド、そして D および A レコードの接頭部内の数値フィールドは、整数値の 1 バイト文字表記を右寄せして先行 0 またはブランクを埋め込んだものでなければなりません。値 0 は、ブランクではなく、少なくとも 1 つの (右寄せされた) 0 によって示されなければなりません。長さがデータ・タイプによって暗黙指定される数値フィールド (たとえば IXFCLENG) が使用されない場合、それにはブランクを埋め込む必要があります。そのような数値フィールドは、次のとおりです。

```
IXFHRECL, IXFTRECL, IXFCRECL, IXFDRECL, IXFARECL,
IXFHHCNT, IXFHSBCP, IXFHDBCP, IXFTCCNT, IXFTNAML,
IXFCLENG, IXFCDRID, IXFCPOSN, IXFCNAML, IXFCTYPE,
IXFCSBCP, IXFCDBCP, IXFCNDIM, IXFCDSIZ, IXFDRID
```

注: データベース・マネージャーの PC/IXF ファイル形式は、System/370 の IXF 形式と同じではありません (288ページの『PC/IXF およびバージョン 0 の System/370 IXF の相違』を参照)。

PC/IXF のレコード・タイプ

PC/IXF の基本レコード・タイプには、次の 5 種類があります。

- ヘッダー
- 表
- 列記述子
- データ
- アプリケーション

これに、DB2 UDB が使用する、以下の 6 つのアプリケーション・サブタイプが加わります。

- 索引
- 階層
- 副表
- 継続
- 終了
- 識別

PC/IXF の各レコード・タイプは一連のフィールドとして定義されます。それらのフィールドは必須であり、示されている順序になっていなければなりません。

ヘッダー・レコード

フィールド名	長さ	タイプ	備考
IXFHRECL	06-BYTE	CHARACTER	レコード長
IXFHRECT	01-BYTE	CHARACTER	レコード・タイプ = 'H'
IXFHID	03-BYTE	CHARACTER	IXF 識別子
IXFHVERS	04-BYTE	CHARACTER	IXF のバージョン
IXFHPROD	12-BYTE	CHARACTER	製品
IXFHDATE	08-BYTE	CHARACTER	作成日付
IXFHTIME	06-BYTE	CHARACTER	作成時刻
IXFHHCNT	05-BYTE	CHARACTER	ヘッダー・レコードのカウント
IXFHSBCP	05-BYTE	CHARACTER	単一バイト・コード・ページ
IXFHDBCP	05-BYTE	CHARACTER	2 バイト・コード・ページ
IXFHFIL1	02-BYTE	CHARACTER	予約済み

ヘッダー・レコードには、以下のフィールドが含まれています。

IXFHRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 H レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFHRECT
IXF レコード・タイプ (このレコードの場合、H にセットされる)。

IXFHID
ファイル形式識別子 (このファイルの場合、IXF にセットされる)。

IXFHVERS
ファイルの作成時に使用された PC/IXF 形式レベル (0002 にセットされる)。

IXFHPROD
ファイルを作成しているプログラムが、それ自体を識別するために使用できるフィールド。このフィールドにデータが入っている場合、その最初の 6 バイトはファイルを作成した製品を識別するために使用され、最後の 6 バイトはその製品のバージョンまたはリリースを示すために使用されます。データベース・マネージャーは、データベース・マネージャー固有データの存在を通知するためにこのフィールドを使用します。

IXFHDATE
ファイルの作成日付 (*yyyymmdd* の形式)。

IXFHTIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドはオプションであり、ブランクのままにしておくことができます。

IXFHHCNT
このファイルのうち最初のデータ・レコードより前にある H、T、および C レコードの数。 A レコードは、このカウントに含まれません。

IXFHSBCP
SBCS CPGID または 00000 の 1 バイト文字表記を含む 1 バイト・コード・ページ・フィールド。

エクスポート・ユーティリティーは、このフィールドを、エクスポートされるデータベース表の SBCS CPGID と等しい値にセットします。たとえば、表の SBCS CPGID が 850 の場合、このフィールドの内容は 00850 です。

IXFHDBCP
DBCS CPGID または 00000 の 1 バイト文字表記を含む 2 バイト・コード・ページ・フィールド。

エクスポート・ユーティリティーは、このフィールドを、エクスポートされるデータベース表の DBCS CPGID と等しい値にセットします。たとえば、表の DBCS CPGID が 301 の場合、このフィールドの内容は 00301 です。

IXFHFIL1
ホスト IXF ファイルの予約フィールドに一致させるために、2 つのブランクにセットされるスペア・フィールド。

表レコード

フィールド名	長さ	タイプ	備考
IXFTRECL	006-BYTE	CHARACTER	レコード長
IXFTRECT	001-BYTE	CHARACTER	レコード・タイプ = 'T'
IXFTNAML	003-BYTE	CHARACTER	名前の長さ
IXFTNAME	256-BYTE	CHARACTER	データの名前
IXFTQULL	003-BYTE	CHARACTER	修飾子の長さ
IXFTQUAL	256-BYTE	CHARACTER	修飾子
IXFTSRC	012-BYTE	CHARACTER	データ・ソース
IXFTDATA	001-BYTE	CHARACTER	データ規則 = 'C'
IXFTFORM	001-BYTE	CHARACTER	データ形式 = 'M'
IXFTMFRM	005-BYTE	CHARACTER	機械形式 = 'PC'
IXFTLOC	001-BYTE	CHARACTER	データ・ロケーション = 'I'
IXFTCCNT	005-BYTE	CHARACTER	'C' レコード・カウント

```
IXFTFIL1        002-BYTE   CHARACTER   予約済み
IXFTDESC        030-BYTE   CHARACTER   データの記述
IXFTPKNM        257-BYTE   CHARACTER   基本キー名
IXFTDSPC        257-BYTE   CHARACTER   予約済み
IXFTISPC        257-BYTE   CHARACTER   予約済み
IXFTLSPC        257-BYTE   CHARACTER   予約済み
```

表レコードには、以下のフィールドが含まれています。

IXFTRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 T レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFTRECT
IXF レコード・タイプ (このレコードの場合、T にセットされる)。

IXFTNAML
IXFTNAME フィールド内の表名の長さ (バイト単位)。

IXFTNAME
表の名前。各ファイルごとに 1 つの表だけが含まれる場合、これは単なる情報フィールドです。データベース・マネージャーは、データのインポート時にこのフィールドを使用しません。 PC/IXF ファイルへの書き込み時にデータベース・マネージャーは、DOS ファイル名 (場合によってはパス情報) をこのフィールドに書き込みます。

IXFTQULL
IXFTQUAL フィールド内の表名修飾子の長さ (バイト単位)。

IXFTQUAL
リレーショナル・システム内で表の作成者を識別する表名修飾子。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFTSRC
データのオリジナル・ソースを指示するために使用されます。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または

表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFTDATA
データの記述に使用される規則。インポートまたはエクスポートの場合、このフィールドは C にセットしなければなりません。これは、個々の列属性が次の列記述子 (C) レコードで記述されており、データが PC/IXF 規則に従っていることを示します。

IXFTFORM
数値データの保管に使用される規則。このフィールドは、M にセットしなければなりません。これは、データ (D) レコード内の数値データが IXFTMFRM フィールドによって指定されるマシン (内部) 形式で保管されていることを示します。

IXFTMFRM
PC/IXF ファイル内のマシン・データの形式。データベース・マネージャーは、このフィールドが PC*bbb* にセットされている場合にのみ、ファイルの読み取りまたは書き込みを実行します。 *b* はブランクを表し、PC は、PC/IXF ファイルのデータが IBM PC マシン形式になっていることを指定します。

IXFTLOC
データの位置。データベース・マネージャーは、値として I のみサポートします。これは、データがこのファイルの内部にあることを意味します。

IXFTCCNT
この表内の C レコードの数。これは、整数値の右寄せ文字表記です。

IXFTFIL1
ホスト IXF ファイルの予約フィールドに一致させるために、2 つのブランクにセットされるスペア・フィールド。

IXFTDESC
表に関する記述データ。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。列がデフォルトによってヌルになっておらず、表名がワークステーションのデータベースからのものである場合、このフィールドには NOT NULL WITH DEFAULT が入ります。

IXFTPKNM
表で定義されている基本キーの名前 (ある場合)。この名前はヌル文字で終了するストリングとして格納されます。

IXFTDSPC
このフィールドは将来の利用のために予約済み。

IXFTISPC
このフィールドは将来の利用のために予約済み。

IXFTLSPC
このフィールドは将来の利用のために予約済み。

列記述子レコード

フィールド名	長さ	タイプ	備考
IXFCRECL	006-BYTE	CHARACTER	レコード長
IXFCRECT	001-BYTE	CHARACTER	レコード・タイプ = 'C'
IXFCNAML	003-BYTE	CHARACTER	列名長
IXFCNAME	256-BYTE	CHARACTER	列名
IXFCNULL	001-BYTE	CHARACTER	列がヌル可能
IXFCDEF	001-BYTE	CHARACTER	列にデフォルトがある
IXFCSLCT	001-BYTE	CHARACTER	列選択フラグ
IXFCKPOS	002-BYTE	CHARACTER	基本キーでの位置
IXFCCLAS	001-BYTE	CHARACTER	データ・クラス
IXFCTYPE	003-BYTE	CHARACTER	データ・タイプ
IXFCSBCP	005-BYTE	CHARACTER	単一バイト・コード・ページ
IXFCDBCP	005-BYTE	CHARACTER	2 バイト・コード・ページ
IXFCLENG	005-BYTE	CHARACTER	列データ長
IXFCDRID	003-BYTE	CHARACTER	'D' レコード識別子
IXFCPOSN	006-BYTE	CHARACTER	列位置
IXFCDESC	030-BYTE	CHARACTER	列記述
IXFCLOBL	020-BYTE	CHARACTER	LOB 列の長さ
IXFCUDTL	003-BYTE	CHARACTER	UDT 名の長さ
IXFCUDTN	256-BYTE	CHARACTER	UDT 名
IXFCDEFL	003-BYTE	CHARACTER	デフォルト値の長さ
IXFCDEFV	254-BYTE	CHARACTER	デフォルト値
IXFCDLPR	010-BYTE	CHARACTER	データ・リンク特性
IXFCREF	001-BYTE	CHARACTER	参照タイプ
IXFCNDIM	002-BYTE	CHARACTER	次元数
IXFCDSIZ	varying	CHARACTER	各次元のサイズ

列記述子レコードには、以下のフィールドが含まれています。

IXFCRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 C レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFCRECT
IXF レコード・タイプ (このレコードの場合、C にセットされる)。

IXFCNAML
IXFCNAME フィールド内の列名の長さ (バイト単位)。

IXFCNAME
列の名前。

IXFCNULL
この列でヌルが可能かどうかを指定します。有効な設定は、Y または N です。

IXFCDEF
このフィールドのデフォルト値が定義されているかどうかを指定します。有効な設定は、Y または N です。

IXFCSLCT
データ中の列のサブセットの選択を可能にすることを目的としたフィールドで、現在は廃止されています。 PC/IXF ファイルへの書き込みを実行するプログラムでは、このフィールドに常に Y を保管しなければなりません。 PC/IXF ファイルを読むプログラムでは、このフィールドを無視しなければなりません。

IXFCKPOS
基本キーの一部としての列の位置。有効値は 01〜16 ですが、基本キーの一部でない列の場合は N です。

IXFCCLAS
IXFCTYPE フィールドで使用されるデータ・タイプのクラス。データベース・マネージャーは、リレーショナル・タイプ (R) のみサポートします。

IXFCTYPE
列のデータ・タイプ。データ・タイプについては、259ページの『PC/IXF データ・タイプ』を参照してください。

IXFCSBCP
SBCS CPGID の 1 バイト文字表記。このフィールドは、この列の D レコードの IXFDCOLS フィールドに入れられる 1 バイト文字データの CPGID を指定します。

このフィールドの意味は、列のデータ・タイプ (IXFCTYPE フィールドで指定される) によって違います。

- 文字ストリング列の場合、このフィールドには、通常、H レコードの IXFHSBCP フィールドの値と等しい 0 以外の値が入っていなけ

ればなりません。ただし、その他の値も許可されます。この値が 0 の場合、列はビット・ストリング・データを含むものとして解釈されます。

- 数値列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- 日付列または時刻列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは IXFHSBCP フィールドの値にセットされ、インポート・ユーティリティーでは無視されます。
- グラフィック列の場合、このフィールドは 0 でなければなりません。

266ページの表12 を参照してください。

IXFCDBCP

DBCS CPGID の 1 バイト文字表記。このフィールドは、この列の D レコードの IXFDCOLS フィールドに入れられる 2 バイト文字データの CPGID を指定します。

このフィールドの意味は、列のデータ・タイプ (IXFCTYPE フィールドで指定される) によって違います。

- 文字ストリング列の場合、このフィールドには、0 か、または H レコードの IXFHDBCP フィールドの値と等しい値が入っていなければなりません。ただし、その他の値も許可されます。 IXFCSBCP フィールドの値が 0 の場合、このフィールドの値は 0 でなければなりません。
- 数値列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- 日付列または時刻列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- グラフィック列の場合、このフィールドの値は IXFHDBCP フィールドの値と同じでなければなりません。

266ページの表12 を参照してください。

IXFCLENG

記述されている列のサイズに関する情報を提供します。データ・タイプによっては、このフィールドは使用されず、ブランクを入れる必要があ

ります。他のいくつかのデータ・タイプの場合、このフィールドには、列の長さを指定する整数の右寄せ文字表記が入れられます。また、さらに別のいくつかのデータ・タイプの場合、このフィールドは 2 つのサブフィールド (精度を表す 3 バイトと位取りを表す 2 バイト) に分割されます。これらのサブフィールドはいずれも整数の右寄せ文字表記です。

IXFCDRID

D レコード識別子。このフィールドの内容は、整数値の右寄せ文字表記です。 PC/IXF ファイルでは、各行のデータを入れるために複数の D レコードが使用されることがあります。このフィールドは、あるデータ行を表現する D レコードのうち、どの D レコードに列のデータが入っているかを指定します。値 1 (たとえば 001) は、列のデータがデータ行の最初の D レコードに入っていることを示します。最初の C レコードの IXFCDRID 値は 1 でなければなりません。それ以降のすべての C レコードの IXFCDRID 値は、その直前の C レコードと等しい値か、または 1 大きい値でなければなりません。

IXFCPOSN

このフィールドの値は、表のあるデータ行を表現する D レコードの 1 つの中で、列のデータを見つけるのに使用されます。これは、D レコードの IXFDCOLS フィールド内でのこの列のデータの開始位置です。列がヌル可能なら IXFCPOSN はヌル標識を指し、それ以外の場合にはデータ自体を指します。列に可変長データが含まれる場合、データ自体が現行長標識で始まります。 D レコードの IXFDCOLS フィールド内の最初のバイトに対応する IXFCPOSN 値は 1 です (0 ではありません)。列が新しい D レコードに入っている場合、IXFCPOSN 値は 1 になります。それ以外の場合、IXFCPOSN 値はデータ値が重なり合わない範囲で列ごとに増加します。

IXFCDESC

列に関する記述情報。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFCLOBL

この列内で定義される long または LOB の長さ (バイト単位)。この列が long または LOB でない場合、このフィールドの値は 000 になります。

IXFCUDTL
IXFCUDTN フィールド内のユーザー定義タイプ (UDT) 名の長さ (バイト単位)。この列のタイプが UDT でない場合、このフィールドの値は 000 になります。

IXFCUDTN
この列のデータ・タイプとして使われるユーザー定義タイプの名前。

IXFCDEFL
IXFCDEFV フィールド内のデフォルト値の長さ (バイト単位)。この列にデフォルト値がない場合、このフィールドの値は 000 になります。

IXFCDEFV
この列にデフォルト値が定義されている場合に、それを指定します。

IXFCDLPR
列が DATALINK 列である場合、このフィールドは次のような特性を記述します。

- 先頭文字は、リンク・タイプを表し、値 U を持ちます。
- 2 番目の文字は、リンク制御タイプを表します。有効値は、 N (制御なし) および F (ファイル制御) です。
- 3 番目の文字は保全性のレベルを表し、値 A を持ちます (データベース・マネージャーがすべての DATALINK 値を制御する場合)。
- 4 番目の文字は、読み取り許可を表します。有効値は、 D (データベース決定の許可) および F (ファイル・システム決定の許可) です。
- 5 番目の文字は、書き込み許可を表します。有効値は、 B (ブロック・アクセス) および F (ファイル・システム決定の許可) です。
- 6 番目の文字は回復オプションを表します。有効値は Y (DB2 がこの列で参照されているファイルの時刻指定回復をサポートする) と N (サポートなし) です。
- 7 番目の文字は、データ・ファイルのリンク解除時に実行される処置を表します。有効値は、R (復元) と D (ファイルの削除) です。

IXFCREF
列が階層の一部を成している場合、このフィールドは、その列がデータ列 (D) または参照列 (R) のどちらであるかを指定します。

IXFCNDIM
列の次元の数。配列は、このバージョンの PC/IXF ではサポートされていません。したがって、このフィールドの内容は整数値 0 の文字表記でなければなりません。

IXFCDSIZ
各次元のサイズまたは範囲。このフィールドの長さは、1 次元あたり 5 バイトです。配列はサポートされない (次元の数は 0 でなければならない) ため、このフィールドは長さ 0 であり、実際には存在しません。

データ・レコード

フィールド名	長さ	タイプ	備考
IXFDRECL	06-BYTE	CHARACTER	レコード長
IXFDRECT	01-BYTE	CHARACTER	レコード・タイプ = 'D'
IXFDRID	03-BYTE	CHARACTER	'D' レコード識別子
IXFDFIL1	04-BYTE	CHARACTER	予約済み
IXFDCOLS	varying	variable	縦欄データ

データ・レコードには、以下のフィールドが含まれています。

IXFDRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 D レコードは、レコードに保管される最後のデータ列の現行オカレンスのすべての有効なデータを入れるのに十分な長さでなければなりません。

IXFDRECT
IXF レコード・タイプ (このレコードの場合、D にセットされる)。これは、このレコードに表のデータ値が含まれることを示します。

IXFDRID
レコード識別子。これは、あるデータ行を表現する一連の D レコードの中で特定の D レコードを識別します。あるデータ行の最初の D レコードの場合、このフィールドの値は 1 になります。第 2 の D レコードの場合、このフィールドの値は 2 になり、以下同様です。各データ行の中には、C レコードの中で指定されているすべての D レコード識別子が実際に存在していなければなりません。

IXFDFIL1
ホスト IXF ファイルの中で、予約フィールドに対応するブランク 4 個に設定されるフィールドで、使用される可能性があるシフトアウト文字のプレースホルダーとなるもの。

IXFDCOLS
縦欄データ用の領域。データ・レコード (D レコード) のデータ域は、1 つまたは複数の列項目で構成されます。その D レコードと同じ D

レコード識別子の列記述子レコードごとに、1 つの列項目があります。D レコードにおける列項目の開始位置は、C レコードの IXFCPOSN 値によって指示されます。

列項目データの形式は、列がヌル可能であるかどうかによって異なります。

- 列がヌル可能である場合 (IXFCNULL フィールドが Y にセットされている場合)、列項目データにはヌル標識が含まれます。列がヌルでない場合は、標識の後にデータ・タイプ固有の情報 (実際のデータベース値を含む) が続きます。ヌル標識は、ヌルでない場合は x'0000' にセットされ、ヌルの場合は x'FFFF' にセットされる 2 バイトの値です。
- 列がヌル可能でない場合、列項目データの内容はデータ・タイプ固有の情報 (実際のデータベース値を含む) だけです。

可変長データ・タイプの場合のデータ・タイプ固有の情報には、現行長標識が含まれます。現行長標識は、IXFTMFRM フィールドによって指定される形式の 2 バイトの整数です。

D レコードのデータ域の長さは、32 771 バイト以下です。

アプリケーション・レコード

フィールド名	長さ	タイプ	備考
IXFARECL	06-BYTE	CHARACTER	レコード長
IXFARECT	01-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	12-BYTE	CHARACTER	アプリケーション識別子
IXFADATA	varying	variable	アプリケーション固有のデータ

アプリケーション・レコードには、以下のフィールドが含まれています。

IXFARECL

レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT

IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID

アプリケーション識別子。これは、A レコードを作成したアプリケーションを識別します。データベース・マネージャーによって作成された PC/IXF ファイルの A レコードの場合、このフィールドの最初の 6 文字はデータベース・マネージャーを識別する定数にセットされ、最後の 6 文字は、データベース・マネージャーまたは A レコードを作成している別のアプリケーションのリリースまたはバージョンを識別します。

IXFADATA

このフィールドには、アプリケーションに依存する補足データが入れられます。その形式と内容は、A レコードを作成するプログラムと、A レコードを処理する他のアプリケーションによってのみ認識されます。

DB2 索引レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション識別子 = 'DB2 02.00'
IXFAITYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'I'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFANDXL	002-BYTE	SHORT INT	索引名の長さ
IXFANDXN	256-BYTE	CHARACTER	索引名
IXFANCL	002-BYTE	SHORT INT	索引作成者名の長さ
IXFANCN	256-BYTE	CHARACTER	索引作成者名
IXFATABL	002-BYTE	SHORT INT	表名の長さ
IXFATABN	256-BYTE	CHARACTER	表名
IXFATCL	002-BYTE	SHORT INT	表作成者名の長さ
IXFATCN	256-BYTE	CHARACTER	表作成者名
IXFAUNIQ	001-BYTE	CHARACTER	固有の規則
IXFACCNT	002-BYTE	CHARACTER	列カウント
IXFAREVS	001-BYTE	CHARACTER	逆スキャン・フラグの許可
IXFAPCTF	002-BYTE	CHARACTER	空き PCT の量
IXFAPCTU	002-BYTE	CHARACTER	最小使用 PCT の量
IXFAEXTI	001-BYTE	CHARACTER	予約済み
IXFACNML	002-BYTE	SHORT INT	列名の長さ
IXFACOLN	varying	CHARACTER	索引内の列の名前

ユーザー定義索引ごとに、このタイプのレコードが 1 つずつ指定されます。このレコードは、表のすべての C レコードの後に置かれます。 DB2 索引レコードには、以下のフィールドが含まれています。

IXFARECL

レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT

IXF レコード・タイプ (このレコードの場合、A にセットされる)。こ

れがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAITYP
これがサブタイプ "I" の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFANDXL
IXFANDXN フィールド内の索引名の長さ (バイト単位)。

IXFANDXN
索引の名前。

IXFANCL
IXFANCN フィールド内の索引作成者名の長さ (バイト単位)。

IXFANCN
索引作成者の名前。

IXFATABL
IXFATABN フィールド内の表名の長さ (バイト単位)。

IXFATABN
表の名前。

IXFATCL
IXFATCN フィールド内の表作成者名の長さ (バイト単位)。

IXFATCN
表作成者の名前。

IXFAUNIQ
索引のタイプを指定します。有効値は、`P` (基本キー)、`U` (固有索引)、および `D` (非固有索引) です。

IXFACCNT
索引定義内の列の数を指定します。

IXFAREVS
この索引で逆スキャンを行えるかどうかを指定します。有効値は、 Y (逆スキャン可) と N (逆スキャン不可) です。

IXFAPCTF
空き状態にしておく索引ページのパーセントを指定します。有効値は -1〜99 です。 -1 またはゼロの値を指定すると、システム・デフォルト値が使われます。

IXFAPCTU
2 つの索引ページを組み合わせる場合に、あらかじめ空きになっている必要のある索引ページの最小パーセントを指定します。有効値は 00〜99 の範囲です。

IXFAEXTI
将来の利用のために予約済み。

IXFACNML
IXFACOLN フィールド内の列名の長さ (バイト単位)。

IXFACOLN
この索引の一部を成す列の名前。有効値は *+name-name...* の形式です。 + は列の昇順ソートを指定し、 - は列の降順ソートを指定します。

DB2 階層レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション識別子 = 'DB2 02.00'
IXFAXTYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'X'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFAYCNT	010-BYTE	CHARACTER	この階層の 'Y' レコード・カウント
IXFAYSTR	010-BYTE	CHARACTER	この階層の開始列

階層を記述する際、このタイプのレコードが 1 つ使われます。すべての副表レコード (以下を参照) は階層レコードの直後に置かれなければならず、階層レコードは表のすべての C レコードの後に置かれます。 DB2 階層レコードには、以下のフィールドが含まれています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字

表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAXTYP
これがサブタイプ ″X″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFAYCNT
この階層レコードの後に予期される副表レコードの数を指定します。

IXFAYSTR
エクスポート・データの先頭における副表レコードの索引を指定します。階層のエクスポートがルート以外の副表から開始された場合、その副表のすべての親表がエクスポートされます。このフィールドには、IXF ファイル内のこの副表の位置も格納されます。最初の X レコードは、ゼロの索引をもつ列を表します。

DB2 副表レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション識別子 = 'DB2 02.00'
IXFAYTYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'Y'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFASCHL	003-BYTE	CHARACTER	タイプ・スキーマ名の長さ
IXFASCHN	256-BYTE	CHARACTER	タイプ・スキーマ名
IXFATYPL	003-BYTE	CHARACTER	タイプ名の長さ
IXFATYPN	256-BYTE	CHARACTER	タイプ名
IXFATABL	003-BYTE	CHARACTER	表名の長さ
IXFATABN	256-BYTE	CHARACTER	表名

```
IXFAPNDX       010-BYTE   CHARACTER   親表の副表索引
IXFASNDX       005-BYTE   CHARACTER   現在の表の開始列索引
IXFAENDX       005-BYTE   CHARACTER   現在の表の終了列索引
```

階層の一部として副表を記述する際、このタイプのレコードが 1 つ使われます。階層に属するすべての副表レコードは、対応する階層レコードの直後に、一緒に格納されている必要があります。副表は 1 つ以上の列で構成され、おのおのの列は列レコード内で記述されます。副表内の各列は、連続した C レコード・セット内で記述しなければなりません。 DB2 副表レコードには、以下のフィールドが含まれています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAYTYP
これがサブタイプ ″Y″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFASCHL
IXFASCHN フィールド内の副表スキーマ名の長さ (バイト単位)。

IXFASCHN
副表スキーマの名前。

IXFATYPL
IXFATYPN フィールド内の副表名の長さ (バイト単位)。

IXFATYPN
副表の名前。

IXFATABL
IXFATABN フィールド内の表名の長さ (バイト単位)。

IXFATABN
表の名前。

IXFAPNDX
親副表の副表レコード索引。この副表が階層のルートである場合、このフィールドには値 -1 が入ります。

IXFASNDX
この副表を構成する列レコードの開始索引。

IXFAENDX
この副表を構成する列レコードの終了索引。

DB2 継続レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション識別子 = 'DB2 02.00'
IXFACTYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'C'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFALAST	002-BYTE	SHORT INT	最後のディスケット・ボリューム番号
IXFATHIS	002-BYTE	SHORT INT	このディスケット・ボリューム番号
IXFANEXT	002-BYTE	SHORT INT	次のディスケット・ボリューム番号

ファイルが最終ボリュームでない限り、このレコードは、複数ボリュームの IXF ファイルの一部を成す各ファイルの末尾にあります。また、ファイルが先頭ボリュームでない限り、このレコードは、複数ボリュームの IXF ファイルの一部を成す各ファイルの先頭にもあります。このレコードの目的は、ファイル順序を記録しておくことです。 DB2 継続レコードには、以下のフィールドが含まれています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFACTYP
これがサブタイプ ″C″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFALAST
このフィールドは、リトル・エンディアン形式の 2 進数フィールドです。値は、IXFATHIS の値より 1 小さくなければなりません。

IXFATHIS
このフィールドは、リトル・エンディアン形式の 2 進数フィールドです。連続ボリューム上では、このフィールドの値も連続している必要があります。最初のボリュームでは 1 の値を持ちます。

IXFANEXT
このフィールドは、リトル・エンディアン形式の 2 進数フィールドです。レコードがファイルの先頭にない場合、値は、IXFATHIS 内の値よりも 1 大きくなければなりません。先頭にある場合、値はゼロでなければなりません。

DB2 終了レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション識別子 = 'DB2 02.00'
IXFAETYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'E'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻

このレコードは、IXF ファイルの末尾にあるファイル終わりマーカーです。
DB2 終了レコードには、以下のフィールドが含まれています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAETYP
これがサブタイプ ″E″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

DB2 識別レコード

フィールド名	長さ	タイプ	備考
IXFARECL	06-BYTE	CHARACTER	レコード長
IXFARECT	01-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	12-BYTE	CHARACTER	アプリケーション識別子
IXFATYPE	01-BYTE	CHARACTER	アプリケーション固有のレコード・タイプ = 'S'
IXFADATE	08-BYTE	CHARACTER	アプリケーション・レコードの作成日
IXFATIME	06-BYTE	CHARACTER	アプリケーション・レコードの作成時刻
IXFACOLN	06-BYTE	CHARACTER	識別列の列番号
IXFAITYP	01-BYTE	CHARACTER	常に生成される ('Y' または 'N')
IXFASTRT	33-BYTE	CHARACTER	識別列の START AT 値
IXFAINCR	33-BYTE	CHARACTER	識別列の INCREMENT BY 値
IXFACACH	10-BYTE	CHARACTER	識別列の CACHE 値

DB2 識別レコードには、以下のフィールドが含まれています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション識別子によって暗黙指定されるデータの内容と形式に関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション識別子。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFATYPE
アプリケーション固有のレコード・タイプ。このフィールドの値は常に ″S″ でなければなりません。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFACOLN
表内の識別列の列番号。

IXFAITYP
識別列のタイプ。 ″Y″ の値は、識別列が常に GENERATED であることを示します。他のすべての値は、列が GENERATED BY DEFAULT タイプであることを意味すると解釈されます。

IXFASTRT
表の作成時に CREATE TABLE ステートメントで指定された識別列の START AT 値。

IXFAINCR
表の作成時に CREATE TABLE ステートメントで指定された識別列の INCREMENT BY 値。

IXFACACH

表の作成時に CREATE TABLE ステートメントで指定された識別列の CACHE 値。 ″1″ の値は、NO CACHE オプションに対応します。

PC/IXF データ・タイプ

表 11. PC/IXF データ・タイプ

名前	IXFCTYPE の値	説明
BIGINT	492	IXFTMFRM によって指定される形式の 8 バイトの整数。 -9 223 372 036 854 775 808～9 223 372 036 854 775 807 の範囲の整数を表します。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。IXFCLENG は使用されず、ブランクを入れる必要があります。
BLOB、CLOB	404、408	可変長文字ストリング。ストリングの最大長は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 32 767 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 4 バイトの整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 次の記述は BLOB にのみ適用されます。IXFCSBCP が 0 の場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。 次の記述は CLOB にのみ適用されます。IXFCDBCP が 0 以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も含めることができます。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
BLOB_FILE、 CLOB_FILE、 DBCLOB_FILE	916、920、924	*name_length* および *name* フィールドにデータが入れられた SQLFILE 構造体を含む固定長フィールド。ストリングの長さは、列記述子レコードの IXFCLENG フィールドに入れられ、それは 255 バイト以下です。ファイル名のコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP が 0 以外の場合、ファイル名には IXFCDBCP によって指定されるコード・ページの 2 バイト文字も含めることができます。 IXFCSBCP が 0 の場合、ファイル名はビット・データであり、変換プログラムによって変換しないようにしてください。 構造体の長さが IXFCLENG に保管されるため、元の LOB の実際の長さは失われます。 LOB は *sql_lobfile_len* の長さで作成されるため、タイプ BLOB_FILE、CLOB_FILE、または DBCLOB_FILE の列を含む IXF ファイルを使用して LOB フィールドを再作成しないようにしてください。
CHAR	452	固定長文字ストリング。ストリングの長さは、列記述子レコードの IXFCLENG フィールドに入れられ、それは 254 バイト以下です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP が 0 以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も含めることができます。 IXFCSBCP が 0 の場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
DATE	384	グレゴリオ暦に準拠した時刻。それぞれの日付は、国際標準化機構規格 (ISO) 形式 (*yyyy-mm-dd*) の 10 バイトの文字ストリングです。年の部分の範囲は 0001～9999 です。月の部分の範囲は 01～12 です。日の部分の範囲は 01～*n* です。ここで、*n* は月によって異なり、月の日数とうるう年に関する一般的な規則に従います。どの部分においても、先行 0 は省略できません。IXFCLENG は使用されず、ブランクを入れる必要があります。 DATE 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。
DBCLOB	412	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の最大文字数を指定するもので、16 383 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 4 バイトの整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
DECIMAL	484	精度が P (列記述子レコードの IXFCLENG の最初の 3 バイトによって指定される) で、位取りが S (IXFCLENG の最後の 2 バイトによって指定される) であるパック 10 進数。パック 10 進数の長さは (P+2)/2 です (バイト単位)。精度は 1〜31 の範囲の奇数でなければなりません。パック 10 進数は、IXFTMFRM によって指定される内部形式になっています。 IXFTMFRM では、PC のパック 10 進数が System/370 のパック 10 進数と同じになるように定義されます。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。
FLOATING POINT	480	長精度 (8 バイト) または短精度 (4 バイト) 浮動小数点数。これは、IXFCLENG が 8 または 4 のどちらにセットされているかによって決まります。データは、IXFTMFRM によって指定される内部マシン形式です。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 4 バイトの浮動小数点数は、データベース・マネージャーではサポートされません。
GRAPHIC	468	2 バイト文字の固定長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の文字数を指定するもので、127 以下です。ストリングの実際の長さ (バイト単位) は、IXFCLENG フィールドの値の 2 倍です。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
INTEGER	496	IXFTMFRM によって指定される形式の 4 バイトの整数。これは、-2 147 483 648～+2 147 483 647 の範囲の整数を表します。IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 IXFCLENG は使用されず、ブランクを入れる必要があります。
LONGVARCHAR	456	可変長文字ストリング。ストリングの最大長は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 32 767 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 2 バイトの整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP が 0 以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も含めることができます。 IXFCSBCP が 0 の場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。
LONG VARGRAPHIC	472	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の最大文字数を指定するもので、16 383 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 2 バイトの整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
SMALLINT	500	IXFTMFRM によって指定される形式の 2 バイトの整数。これは -32 768〜+32 767 の範囲の整数を表します。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 IXFCLENG は使用されず、ブランクを入れる必要があります。
TIME	388	24 時間制による時刻。それぞれの時刻は、ISO 形式 (*hh.mm.ss*) の 8 バイトのストリングです。時の部分の範囲は 00〜24 で、その他の部分の範囲は 00〜59 です。時が 24 の場合、その他の部分は 00 です。最小の時刻は 00.00.00、最大の時刻は 24.00.00 です。どの部分においても、先行 0 は省略できません。 IXFCLENG は使用されず、ブランクを入れる必要があります。 TIME 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。
TIMESTAMP	392	マイクロ秒の精度の日時。各タイム・スタンプは、*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) の形式の文字ストリングです。 IXFCLENG は使用されず、ブランクを入れる必要があります。 TIMESTAMP 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。

表 11. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
VARCHAR	448	可変長文字ストリング。ストリングの最大長 (バイト単位) は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 254 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 2 バイトの整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP が 0 以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も含めることができます。 IXFCSBCP が 0 の場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。
VARGRAPHIC	464	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の最大文字数を指定するもので、127 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 2 バイトの整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。

PC/IXF 文字またはグラフィック列について、IXFCSBCP 値と IXFCDBCP 値の一部の組み合わせは無効です。 IXFCSBCP と IXFCDBCP の組み合わせが無効である PC/IXF 文字またはグラフィック列は、無効なデータ・タイプです。

表 12. 有効な PC/IXF データ・タイプ

PC/IXF データ・タイプ	有効な (IXFCSBCP,IXFCDBCP) 対	無効な (IXFCSBCP,IXFCDBCP) 対
CHAR、VARCHAR、または LONG VARCHAR	(0,0)、(x,0)、または (x,y)	(0,y)
BLOB	(0,0)	(x,0)、(0,y)、または (x,y)
CLOB	(x,0)、(x,y)	(0,0)、(0,y)
GRAPHIC、VARGRAPHIC、 LONG VARGRAPHIC、または DBCLOB	(0,y)	(0,0)、(x,0)、または (x,y)
注: x および y は 0 ではありません。		

PC/IXF データ・タイプの説明

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	データベース列と同じ BIGINT 列が作成されます。	すべての数値タイプ (SMALLINT、INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。 -9 223 372 036 854 775 808〜 9 223 372 036 854 775 807 の範囲外の値があれば、その特定の値に関しては拒否されます。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BLOB	PC/IXF BLOB 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、 VARCHAR、LONG VARCHAR、 BLOB、または BLOB_FILE 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC の BLOB 列も受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
CHAR	PC/IXF CHAR 列が作成されます。データベース列の長さ、SBCS CPGID 値、および DBCS CPGID 値が PC/IXF 列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされているなら、PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の長さと互換性がなければなりません。データは、必要なら右側に 1 バイト・スペース (x'20') が埋められます。 279ページの『FORCEIN オプション』も参照してください。
CLOB	PC/IXF CLOB 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベースの列の DBCS CPGID と等しい場合、 PC/IXF CHAR、 VARCHAR、 LONG VARCHAR、 CLOB、または CLOB_FILE 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
DATE	データベース列と同じ DATE 列が作成されます。	タイプ DATE の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。 PC/IXF ファイル内の文字タイプの列の内容は、ターゲット・データベースの国別コードと互換性のある形式の日付でなければなりません。
DBCLOB	PC/IXF DBCLOB 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、PC/IXF GRAPHIC、 VARGRAPHIC、 LONG VARGRAPHIC、 DBCLOB、または DBCLOB_FILE 列が受け入れ可能です。PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。
DECIMAL	データベース列と同じ DECIMAL 列が作成されます。列の精度と位取りが列記述子レコードに保管されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。インポート先の DECIMAL 列の範囲外の値があれば、その特定の値に関しては拒否されます。
FLOAT	データベース列と同じ FLOAT 列が作成されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。すべての値は範囲内です。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
GRAPHIC (DBCS のみ)	PC/IXF GRAPHIC 列が作成されます。データベース列の長さ、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の長さと互換性がなければなりません。データは、必要なら右側に 2 バイト・スペース (x'8140') が埋められます。 279ページの『FORCEIN オプション』も参照してください。
INTEGER	データベース列と同じ INTEGER 列が作成されます。	すべての数値タイプ (SMALLINT、INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。 -2 147 483 648～2 147 483 647 の範囲外の整数があれば、それに関しては拒否されます。
LONG VARCHAR	PC/IXF LONG VARCHAR 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされているなら、PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
LONG VARGRAPHIC (DBCS のみ)	PC/IXF LONG VARGRAPHIC 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。
SMALLINT	データベース列と同じ SMALLINT 列が作成されます。	すべての数値タイプ (SMALLINT、INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。-32 768～32 767 の範囲外の値があれば、その特定の値に関しては拒否されます。
TIME	データベース列と同じ TIME 列が作成されます。	タイプ TIME の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。 PC/IXF ファイルの文字タイプの列に含まれる時刻データは、ターゲット・データベースの国別コードに合致する形式になっていなければなりません。
TIMESTAMP	データベース列と同じ TIMESTAMP 列が作成されます。	タイプ TIMESTAMP の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。 PC/IXF ファイルの文字タイプの列に含まれるデータは、タイム・スタンプの入力形式になっていなければなりません。

表 13. PC/IXF ファイル形式についての受け入れ可能なデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
VARCHAR	データベース列の最大長が <= 254 の場合は、 PC/IXF VARCHAR 列が作成されます。データベース列の最大長が > 254 の場合は、 PC/IXF LONG VARCHAR 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされているなら、PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の最大長と互換性がなければなりません。279ページの『FORCEIN オプション』も参照してください。
VARGRAPHIC (DBCS のみ)	データベース列の最大長が <= 127 の場合は、 PC/IXF VARGRAPHIC 列が作成されます。データベース列の最大長が > 127 の場合は、 PC/IXF LONG VARGRAPHIC 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。 279ページの『FORCEIN オプション』も参照してください。

PC/IXF ファイルのデータベースへのインポートを制御する一般規則

データベース・マネージャーのインポート・ユーティリティーでは、SBCS または DBCS 環境での PC/IXF ファイルのインポート時に、以下の規則が適用されます。

- インポート・ユーティリティーは、PC/IXF 形式のファイル (IXFHID = 'IXF') のみ受け入れます。その他の形式の IXF ファイルはインポートできません。
- インポート・ユーティリティーは、1024 個より多くの列が入っている PC/IXF ファイルを拒否します。
- PC/IXF H レコードの IXFHSBCP の値は、SBCS CPGID と等しくなければなりません。あるいは、IXFHSBCP/IXFHDBCP と、ターゲット・データベースの SBCS/DBCS CPGID との間の変換表がなければなりません。IXFHDBCP の値は '00000' か、またはターゲット・データベースの DBCS CPGID と等しくなければなりません。これらの条件のいずれも満たされない場合、インポート・ユーティリティーは、279ページの『FORCEIN オプション』が指定されていないなら、PC/IXF ファイルを拒否します。
- 無効なデータ・タイプ - 新しい表

  PC/IXF ファイルの*新しい* 表へのインポートは、IMPORT コマンドの CREATE または REPLACE_CREATE キーワードによって指定されます。新しい表へのインポートにおいて無効なデータ・タイプの PC/IXF 列が選択されると、インポート・ユーティリティーは終了します (有効なデータ・タイプについては、 259ページの『PC/IXF データ・タイプ』で定義されています)。 PC/IXF ファイル全体が拒否され、表は作成されず、データはインポートされません。
- 無効なデータ・タイプ - 既存の表

  PC/IXF ファイルの*既存の* 表へのインポートは、IMPORT コマンドの INSERT、INSERT_UPDATE、または REPLACE_CREATE キーワードによって指定されます。既存の表へのインポートにおいて無効なデータ・タイプの PC/IXF 列が選択されると、次のいずれかの処理が実行されます。
  - ターゲット表列がヌル可能である場合は、無効な PC/IXF 列のすべての値が無視され、表列の値はヌルにセットされます。
  - ターゲット表列がヌル可能でない場合、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体が拒否され、データはインポートされません。既存の表は未変更のままです。
- 新しい表へのインポートにおいて、ヌル可能な PC/IXF 列はヌル可能なデータベース列を生成し、ヌル可能でない PC/IXF 列はヌル可能でないデータベース列を生成します。

- ヌル可能でない PC/IXF 列を、ヌル可能なデータベース列にインポートすることができます。
- ヌル可能 PC/IXF 列を、ヌル可能でないデータベース列にインポートすることができます。 PC/IXF 列内でヌル値が検出されると、インポート・ユーティリティーはヌル値を含む PC/IXF 行のすべての列の値を拒否し (行全体が拒否され)、次の PC/IXF 行から処理が継続されます。つまり、ヌルのターゲット表列がヌル可能でない場合、そのヌル値を含む PC/IXF 行からはデータがインポートされません。
- 互換性のない列 - 新しい表

  新しい データベース表へのインポート中に、ターゲット・データベース列と互換性がない PC/IXF 列が選択されると、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体が拒否され、表は作成されず、データはインポートされません。

  注: IMPORT で 279ページの『FORCEIN オプション』を使用すると、互換性のある列の範囲が広がります。
- 互換性のない列 - 既存の表

  既存の データベース表へのインポート中に、ターゲット・データベース列と互換性がない PC/IXF 列が選択されると、次の 2 つの処理のうちいずれかが実行されます。

  – ターゲット表列がヌル可能である場合は、PC/IXF 列のすべての値が無視され、表列の値はヌルにセットされます。
  – ターゲット表列がヌル可能でない場合、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体が拒否され、データはインポートされません。既存の表は未変更のままです。

  注: IMPORT で 279ページの『FORCEIN オプション』を使用すると、互換性のある列の範囲が広がります。
- 無効な値

  インポート中に、ターゲット・データベース列にとって無効な PC/IXF 列値が検出されると、インポート・ユーティリティーは、無効な値を含む PC/IXF 行のすべての列の値を拒否し (行全体が拒否され)、次の PC/IXF 行から処理が継続されます。
- DBCS データを含む PC/IXF ファイルをインポートまたはロードするには、クライアント・マシンで、対応する変換ファイルが (`sqllib¥conv` に) インストールされていることが必要です。これらの変換ファイルの名前には、ソースとターゲットの両方のコード・ページ番号が含まれ、拡張子は常に `.cnv`

です。たとえば、ファイル 09320943.cnv には、コード・ページ 932 を 943 に変換するための変換表が含まれています。

クライアント・マシンに適切な変換ファイルがない場合は、それらをサーバー・マシンからクライアント・マシンの sqllib¥conv ディレクトリーにコピーすることができます。ファイルは、互換性のあるプラットフォームからコピーするようにしてください。たとえば、クライアントが UNIX ベースのオペレーティング・システムで実行されている場合には、UNIX ベースのオペレーティング・システムで実行されているサーバーからファイルをコピーします。

PC/IXF ファイルのデータベースへのインポートを制御するデータ・タイプ固有の規則

- 有効な PC/IXF 数値列は、互換性のある任意のデータベース数値列にインポートできます。 4 バイトの浮動小数点データが入っている PC/IXF 列は無効なデータ・タイプであるため、インポートできません。
- データベースの日付 / 時刻列は、対応する PC/IXF 日付 / 時刻列 (DATE、TIME、および TIMESTAMP) からの値、および列の長さおよび値の互換性制限に従っている PC/IXF 文字タイプ列 (CHAR、VARCHAR、および LONG VARCHAR) からの値を受け入れることができます。
- 有効な PC/IXF 文字タイプ列 (CHAR、VARCHAR、または LONG VARCHAR) は、FOR BIT DATA としてマークされている*既存の* データベース文字タイプ列に常にインポートできます。それ以外の場合、
  - IXFCSBCP と SBCS CPGID は一致していなければなりません。
  - IXFCSBCP/IXFCDBCP と SBCS/DBCS のための変換表がなければなりません。
  - 1 つのセットがすべて 0 (FOR BIT DATA) でなければなりません。

  IXFCSBCP が 0 でない場合、IXFCDBCP の値は 0、またはターゲット・データベース列の DBCS CPGID の値と等しくなければなりません。

  これらの条件のいずれかが満たされていない場合、PC/IXF とデータベース列には互換性がありません。

  有効な PC/IXF 文字タイプ列を*新しい* データベース表にインポートする場合、IXFCSBCP の値は 0 またはデータベースの SBCS CPGID と等しくなければなりません。あるいは変換表がなければなりません。 IXFCSBCP が 0 の場合は、IXFCDBCP も 0 でなければなりません (そうでなければ、PC/IXF 列は無効なデータ・タイプです)。この場合、IMPORT は、新しい表の中に FOR BIT DATA としてマークされた文字タイプの列を作成します。IXFCSBCP が 0 でなくデータベースの SBCS CPGID と等しい場合、

IXFCDBCP の値は 0 またはデータベースの DBCS CPGID でなければなりません。この場合、ユーティリティーは、新しい表の中に SBCS および DBCS CPGID 値がデータベースと等しい文字タイプの列を作成します。これらの条件が満たされていない場合、PC/IXF とデータベース列には互換性がありません。

279ページの『FORCEIN オプション』を使用すると、コード・ページの同等性検査をオーバーライドすることができます。しかし、IXFCSBCP が 0 で IXFCDBCP が 0 でない PC/IXF 文字タイプ列は無効なデータ・タイプであり、FORCEIN が指定されていてもインポートできません。

- 有効な PC/IXF 文字タイプ列 (GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC) は、 FOR BIT DATA としてマークされている*既存の* データベース文字タイプ列に常にインポートできます。 279ページの『FORCEIN オプション』を使用すると、この制限を緩和することができます。しかし、IXFCSBCP が 0 でないか、または IXFCDBCP が 0 の PC/IXF グラフィック列は無効なデータ・タイプであり、FORCEIN が指定されていてもインポートできません。

  有効な PC/IXF グラフィック列をデータベースのグラフィック列にインポートする場合、 IXFCDBCP の値はターゲット・データベース列の DBCS CPGID と等しくなければなりません (つまり 2 つの列の 2 バイト・コード・ページが一致していなければなりません)。
- PC/IXF ファイルの既存のデータベース表へのインポート中に固定長ストリング列 (CHAR または GRAPHIC) が選択され、その長さがターゲット列の最大長より長い場合、それらの列には互換性がありません。
- PC/IXF ファイルの既存のデータベース表へのインポート中に、可変長ストリング列 (VARCHAR、LONG VARCHAR、VARGRAPHIC、または LONG VARGRAPHIC) が選択され、その長さがターゲット列の最大長より長い場合、それらの列には互換性があります。個々の値は、データベース・マネージャー INSERT ステートメントを制御する規則に従って処理され、ターゲット・データベース列にとって長すぎる PC/IXF 値は無効です。
- 固定長のデータベースの文字 タイプの列 (つまり CHAR 列) にインポートされる PC/IXF 値は、必要なら値の長さがデータベース列と等しくなるよう右側に 1 バイト・スペース (0x20) が埋められます。固定長のデータベースのグラフィック ・タイプの列 (つまり GRAPHIC 列) にインポートされる PC/IXF 値は、必要なら値の長さがデータベース列と等しくなるよう右側に 2 バイト・スペース (0x8140) が埋められます。
- PC/IXF VARCHAR 列の最大長は 254 バイトであるため、最大長が *n* (254 < *n* < 4001) の PC/IXF VARCHAR 列は、最大長が *n* のデータベース LONG VARCHAR 列にエクスポートしなければなりません。

- PC/IXF LONG VARCHAR 列の最大長は 32 767 バイトで、データベース LONG VARCHAR 列には 32 700 バイトの最大長制限がありますが、長さが 32 700 を超える (ただし 32 768 バイトよりも小さい) PC/IXF LONG VARCHAR 列は有効であり、データベース LONG VARCHAR 列にインポートできます。ただしデータが失われる可能性があります。
- PC/IXF VARGRAPHIC 列の最大長は 127 バイトであるため、最大長が *n* (127 < *n* < 2001) の PC/IXF VARGRAPHIC 列は、最大長が *n* のデータベース LONG VARGRAPHIC 列にエクスポートしなければなりません。
- PC/IXF LONG VARGRAPHIC 列の最大長は 16 383 バイトで、データベース LONG VARGRAPHIC 列には 16 350 バイトの最大長制限がありますが、長さが 16 350 を超える (ただし 16 384 バイトよりも小さい) PC/IXF LONG VARGRAPHIC 列は有効であり、データベース LONG VARGRAPHIC 列にインポートできます。ただしデータが失われる可能性があります。

FORCEIN オプションを使用せずに PC/IXF ファイルを新しいデータベース表または既存のデータベース表にインポートする場合について、表14 にまとめます。

表 14. FORCEIN オプションを使用しない場合の PC/IXF ファイルのインポートの要約

PC/IXF の列データ・タイプ	データベースの列データ・タイプ											
	数値					文字			GRAPH	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0)[d]	(SBCS, DBCS)[b]	[b]	DATE	TIME	TIME STAMP
数値												
-SMALLINT	N											
	E	E	E	E[a]	E							
-INTEGER		N										
	E[a]	E	E	E[a]	E							
-BIGINT			N									
	E[a]	E[a]	E	E[a]	E							
-DECIMAL				N								
	E[a]	E[a]	E[a]	E[a]	E							
-FLOAT					N							
	E[a]	E[a]	E[a]	E[a]	E							
文字												
-(0,0)						N						
						E				E[c]	E[c]	E[c]
-(SBCS,0)							N	N				
						E	E	E		E[c]	E[c]	E[c]

表 14. FORCEIN オプションを使用しない場合の PC/IXF ファイルのインポートの要約 (続き)

PC/IXF の列データ・タイプ	データベースの列データ・タイプ											
	数値					文字			GRAPH	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0)[d]	(SBCS, DBCS)[b]	[b]	DATE	TIME	TIME STAMP
-(SBCS, DBCS)								N		E[c]	E[c]	E[c]
						E		E				
グラフィック												
									N			
						E			E			
日時												
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E

注:

1. この表は、すべての有効な PC/IXF およびデータベース・マネージャー・データ・タイプの一覧表です。 PC/IXF 列をデータベース列にインポートできる場合は、 PC/IXF データ・タイプの行とデータベース・マネージャー・データ・タイプの列が交差する位置のセルに文字が示されています。 'N' は、ユーティリティーが新しいデータベース表を作成することを示します (示されているデータ・タイプのデータベース列が作成されます)。 'E' は、ユーティリティーが既存のデータベース表にデータをインポートすることを示します (示されているデータ・タイプのデータベース列は有効なターゲットです)。
2. 文字ストリング・データ・タイプは、コード・ページ属性によって区別されます。これらの属性は、(SBCS,DBCS) の順序対として示されています。ここで、
   - SBCS は 0、または文字データ・タイプの 1 バイト・コード・ページ属性の非 0 値を示します。
   - DBCS は 0、または文字データ・タイプの 2 バイト・コード・ページ属性の非 0 値を示します。
3. この表で文字タイプの PC/IXF 列が文字タイプのデータベース列にインポートできることが示されている場合、それらのコード・ページ属性の対の値はコード・ページの同等性を制御する規則を満たします。

[a] ターゲットの数値データ・タイプの範囲内にない値は、その特定の値に関して拒否されます。

[b] データ・タイプは DBCS 環境でのみ使用可能です。

[c] 有効な日付または時刻の値でない値は、その特定の値に関して拒否されます。

[d] データ・タイプは DBCS 環境では使用不能です。

FORCEIN オプション

FORCEIN オプションを使用すると、PC/IXF ファイルとターゲット・データベースとでデータのコード・ページが違っていても、PC/IXF ファイルをインポートすることができます。このオプションにより、互換性のある列を定義する上でさらに柔軟性が高くなります。

FORCEIN の一般的なセマンティクス

SBCS または DBCS 環境で FORCEIN オプションを使用する場合、下記の一般的なセマンティクスが適用されます。

- FORCEIN オプションを使用する際には、注意が必要です。普通は、このオプションを使用可能にせずにインポートを試みるようにしてください。しかし、PC/IXF データ交換アーキテクチャーの一般的な性質のために、一部の PC/IXF ファイルには、介入なしではインポートできないデータ・タイプまたは値が含まれている可能性があります。
- FORCEIN を使用して*新しい* 表へのインポートを実行する場合、既存の表へのインポートとは異なる結果が生じる可能性があります。既存の表には、PC/IXF の各データ・タイプごとに事前定義のターゲット・データ・タイプが対応しています。
- LOBSINFILE オプションを使用して LOB データがエクスポートされている場合に、ファイルがコード・ページの異なる別のクライアントに移動されると、それぞれ別個のファイルに含まれている CLOB および DBCLOB は、その他のデータの場合とは異なり、データベースへのインポートまたはロード時にクライアントのコード・ページに変換されません。

FORCEIN のコード・ページのセマンティクス

SBCS または DBCS 環境で FORCEIN オプションを使用する場合、下記のコード・ページのセマンティクスが適用されます。

- FORCEIN オプションを指定すると、インポート・ユーティリティーのすべてのコード・ページ比較が使用できなくなります。

  この規則は、新しいデータベース・ファイルまたは既存のデータベース・ファイルへのインポート時に、列レベルとファイル・レベルのコード・ページ比較に適用されます。列 (たとえばデータ・タイプ) レベルでは、この規則は次のデータベース・マネージャーおよび PC/IXF データ・タイプにのみ適用されます。つまり、文字 (CHAR、VARCHAR、および LONG VARCHAR) とグラフィック (GRAPHIC、VARGRAPHIC、および LONG VARGRAPHIC)。この制限は、その他のデータ・タイプのコード・ページ属性がデータ・タイプ値の解釈に関係しないという事実に基づいています。
- FORCEIN オプションを指定しても、データ・タイプを判別するためのコード・ページ属性の検査は使用不可にされません。

たとえば、データベース・マネージャーでは、FOR BIT DATA 属性を使用して CHAR 列を宣言することができます。このような宣言により、その列の SBCS CPGID と DBCS CPGID の両方が 0 に設定されます。それらの CPGID の値が 0 の場合、列値は文字ストリングではなくビット・ストリングとして識別されます。

- FORCEIN オプションは、コード・ページ変換を暗黙指定しません。

  FORCEIN オプションの影響を受けるデータ・タイプの値は、"元のまま" コピーされます。コード・ページ環境の変更に対応するためのコード・ページ・マッピングは使用されません。ターゲット列が固定長の場合には、インポートされた値にスペースを埋めることが必要になることがあります。
- FORCEIN オプションを使用して*既存の* 表にデータをインポートする場合には、
  - ターゲット・データベース表および列のコード・ページ値が常に優先されます。
  - PC/IXF ファイルおよび列のコード・ページ値は無視されます。

  この規則は、FORCEIN オプションが使用されるかどうかに関係なく適用されます。データベース・マネージャーは、データベースの作成後にデータベースまたは列のコード・ページ値の変更を許可しません。
- FORCEIN オプションを使用して*新しい* 表へのインポートを実行する場合には、
  - ターゲット・データベースのコード・ページ値が優先されます。
  - IXFCSBCP = IXFCDBCP = 0 の文字タイプの PC/IXF 列は、FOR BIT DATA としてマークされた表列を生成します。
  - その他のすべての PC/IXF 文字タイプ列は、 SBCS および DBCS CPGID 値がデータベースと等しい文字タイプの表列を生成します。
  - PC/IXF グラフィック列は、SBCS CPGID が "未定義" で、DBCS CPGID がデータベースと等しい表グラフィック列を生成します (DBCS データベースのみ)。

FORCEIN の例

IXFCSBCP = ’00897’ および IXFCDBCP = ’00301’ である PC/IXF CHAR 列について考えてみましょう。この列を、SBCS CPGID = ’00850’ および DBCS CPGID = ’00000’ のデータベース CHAR 列にインポートするとします。
FORCEIN を指定しないなら、ユーティリティーは終了し、データがインポートされないか、あるいは PC/IXF 列値が無視され、データベース列にヌルが入れられます (データベース列がヌル可能である場合)。 FORCEIN を指定した場合、ユーティリティーは、コード・ページの非互換性を無視して処理を継続し

ます。データ・タイプにその他 (長さなど) の非互換性がなければ、PC/IXF 列の値は "元のまま" インポートされ、データベース列のコード・ページ環境で解釈できるようになります。

下記の表には、次のことが示されています。

- PC/IXF ファイルのうち指定されたコード・ページ属性を持つデータ・タイプがインポートされる場合に、*新しい* データベース表で作成される列のコード・ページ。
- PC/IXF データ・タイプが無効であるか、または互換性がない場合に、それらがインポート・ユーティリティーによって拒否されること。

表 15. インポート・ユーティリティーのコード・ページに関するセマンティクスの要約 (新しい表). この表では、a と x の間の変換表がないことを想定しています。もしそれがあるなら、項目 3 および 4 は、FORCEIN オプションが使用されなくても正常に機能します。

PC/IXF データ・タイプのコード・ページ属性	データベース表列のコード・ページ属性	
	FORCEIN を使用しない場合	FORCEIN を使用する場合
SBCS		
(0,0)	(0,0)	(0,0)
(a,0)	(a,0)	(a,0)
(x,0)	拒否	(a,0)
(x,y)	拒否	(a,0)
(a,y)	拒否	(a,0)
(0,y)	拒否	(0,0)
DBCS		
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)
(x,0)	拒否	(a,b)
(a,b)	(a,b)	(a,b)
(x,y)	拒否	(a,b)
(a,y)	拒否	(a,b)
(x,b)	拒否	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	拒否	(-,b)

表 15. インポート・ユーティリティーのコード・ページに関するセマンティクスの要約 (新しい表) (続き). この表では、a と x の間の変換表がないことを想定しています。もしそれがあるなら、項目 3 および 4 は、FORCEIN オプションが使用されなくても正常に機能します。

PC/IXF データ・タイプのコード・ページ属性	データベース表列のコード・ページ属性	
	FORCEIN を使用しない場合	FORCEIN を使用する場合

注:

1. PC/IXF データ・タイプのコード・ページ属性は順序対として示されています。 x は 0 以外の 1 バイト・コード・ページ値、y は 0 以外の 2 バイト・コード・ページ値を表します。 '-' は未定義のコード・ページ値を表します。
2. 各種のコード・ページ属性で、あえて異なる文字を使用しています。異なる文字は値が異なることを暗示しています。たとえば、PC/IXF データ・タイプが (x,y) として示されており、データベース列が (a,y) として示されている場合、x は a と等しくありませんが、 PC/IXF ファイルとデータベースの 2 バイト・コード・ページ値は同じ y です。
3. FORCEIN のコード・ページのセマンティクスによって影響を受けるのは、文字およびグラフィック・データ・タイプだけです。
4. 新しい表を含むデータベースのコード・ページ属性は (a,0) であることが想定されています。このため、新しい表のうち文字タイプのすべての列のコード・ページ属性は (0,0) または (a,0) でなければなりません。

   DBCS 環境において、新しい表を含むデータベースのコード・ページ属性は (a,b) であることが想定されています。そのため、新しい表のうちすべてのグラフィック列のコード・ページ属性は (-,b) でなければならず、文字タイプのすべての列のコード・ページは (a,b) でなければなりません。 SBCS CPGID は、グラフィック・データ・タイプについては未定義であるため、'-', と示されています。
5. 結果のデータ・タイプは、285ページの『FORCEIN のデータ・タイプのセマンティクス』で説明されている規則によって決まります。
6. 拒否の結果は、無効なまたは互換性のないデータ・タイプに関する規則を反映したものです (273ページの『PC/IXF ファイルのデータベースへのインポートを制御する一般規則』を参照)。

下記の表には、次のことが示されています。

- インポート・ユーティリティーが、さまざまなコード・ページ属性を持つ PC/IXF データ・タイプを、指定されたコード・ページ属性を持つ*既存の* 表列 (ターゲット 列) に受け入れること。
- インポート・ユーティリティーが、特定のコード・ページ属性を持つ PC/IXF データ・タイプを、指定されたコード・ページ属性を持つ*既存の* 表

列にインポートするのを許可しないこと。ユーティリティーは、PC/IXF データ・タイプが無効であるか、または互換性がない場合、それらを拒否します。

表 16. インポート・ユーティリティーのコード・ページに関するセマンティクスの要約 (既存の表). この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	ターゲット・データベース列のコード・ページ属性	インポートの結果	
		FORCEIN を使用しない場合	FORCEIN を使用する場合
SBCS			
(0,0)	(0,0)	受け入れ	受け入れ
(a,0)	(0,0)	受け入れ	受け入れ
(x,0)	(0,0)	受け入れ	受け入れ
(x,y)	(0,0)	受け入れ	受け入れ
(a,y)	(0,0)	受け入れ	受け入れ
(0,y)	(0,0)	受け入れ	受け入れ
(0,0)	(a,0)	ヌルまたは拒否	受け入れ
(a,0)	(a,0)	受け入れ	受け入れ
(x,0)	(a,0)	ヌルまたは拒否	受け入れ
(x,y)	(a,0)	ヌルまたは拒否	受け入れ
(a,y)	(a,0)	ヌルまたは拒否	受け入れ
(0,y)	(a,0)	ヌルまたは拒否	ヌルまたは拒否
DBCS			
(0,0)	(0,0)	受け入れ	受け入れ
(a,0)	(0,0)	受け入れ	受け入れ
(x,0)	(0,0)	受け入れ	受け入れ
(a,b)	(0,0)	受け入れ	受け入れ
(x,y)	(0,0)	受け入れ	受け入れ
(a,y)	(0,0)	受け入れ	受け入れ
(x,b)	(0,0)	受け入れ	受け入れ
(0,b)	(0,0)	受け入れ	受け入れ
(0,y)	(0,0)	受け入れ	受け入れ
(0,0)	(a,b)	ヌルまたは拒否	受け入れ
(a,0)	(a,b)	受け入れ	受け入れ

表 16. インポート・ユーティリティーのコード・ページに関するセマンティクスの要約 (既存の表) (続き). この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	ターゲット・データベース列のコード・ページ属性	インポートの結果	
		FORCEIN を使用しない場合	FORCEIN を使用する場合
(x,0)	(a,b)	ヌルまたは拒否	受け入れ
(a,b)	(a,b)	受け入れ	受け入れ
(x,y)	(a,b)	ヌルまたは拒否	受け入れ
(a,y)	(a,b)	ヌルまたは拒否	受け入れ
(x,b)	(a,b)	ヌルまたは拒否	受け入れ
(0,b)	(a,b)	ヌルまたは拒否	ヌルまたは拒否
(0,y)	(a,b)	ヌルまたは拒否	ヌルまたは拒否
(0,0)	(-,b)	ヌルまたは拒否	受け入れ
(a,0)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(x,0)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(a,b)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(x,y)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(a,y)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(x,b)	(-,b)	ヌルまたは拒否	ヌルまたは拒否
(0,b)	(-,b)	受け入れ	受け入れ
(0,y)	(-,b)	ヌルまたは拒否	受け入れ

注:

1. 282ページの表15 の注を参照してください。
2. ヌルまたは拒否の結果は、無効なまたは互換性のないデータ・タイプに関する規則を反映したものです (273ページの『PC/IXF ファイルのデータベースへのインポートを制御する一般規則』を参照)。

FORCEIN のデータ・タイプのセマンティクス

FORCEIN オプションを使用すると、特定の PC/IXF 列を、データ・タイプが違う、またはその他の点で互換性がないデータ・タイプのターゲット・データベース列にインポートすることができます。 SBCS または DBCS 環境で FORCEIN オプションを使用する場合、下記のデータ・タイプのセマンティクスが適用されます (一部の例外を除く)。

- SBCS 環境では、FORCEIN オプションにより、次のインポートが可能になります。

  - PC/IXF BIT データ・タイプ (PC/IXF の文字タイプの列で IXFCSBCP = 0 = IXFCDBCP) を、文字タイプのデータベース列 (SBCS CPGID が 0 以外、DBCS CPGID = 0) にインポートすること。既存の表のみ。
  - PC/IXF MIXED データ・タイプ (IXFCSBCP および IXFCDBCP が 0 以外) を文字タイプのデータベース列にインポートすること。新しい表と既存の表の両方。
  - PC/IXF GRAPHIC データ・タイプを、データベース FOR BIT DATA 列 (SBCS CPGID = 0 = DBCS CPGID) にインポートすること。新しい表のみ (既存の表については常に可能)。
- FORCEIN オプションは、有効な PC/IXF データ・タイプの範囲を拡張しません。

  259ページの『PC/IXF データ・タイプ』で有効として定義されていないデータ・タイプの PC/IXF 列は、FORCEIN オプションが使用されるかどうかに関係なく、インポートについては無効です。
- DBCS 環境では、FORCEIN オプションにより、次のインポートが可能になります。
  - PC/IXF BIT データ・タイプを文字タイプのデータベース列にインポートすること。
  - PC/IXF BIT データ・タイプをデータベース・グラフィック列にインポートすること。ただし、PC/IXF BIT 列が固定長の場合、長さは偶数でなければなりません。長さが奇数である固定長の PC/IXF BIT 列は、データベース・グラフィック列と互換性がありません。可変長の PC/IXF BIT 列は、長さが奇数であっても偶数であっても、互換性があります。ただし、可変長列の奇数の長さの値は、データベース・グラフィック列にインポートについて無効です。
  - PC/IXF MIXED データ・タイプを文字タイプのデータベース列にインポートすること。

FORCEIN オプションを使用して PC/IXF ファイルを新しい表または既存の表にインポートする場合について、表17 にまとめます。

表 17. FORCEIN オプションを使用する場合の PC/IXF ファイルのインポートの要約

PC/IXF の列データ・タイプ	データベースの列データ・タイプ											
	数値					文字			GRAPH	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0)[e]	(SBCS, DBCS)[b]	[b]	DATE	TIME	TIME STAMP
数値												
-SMALLINT	N											

表 17. FORCEIN オプションを使用する場合の PC/IXF ファイルのインポートの要約 (続き)

PC/IXF の列データ・タイプ	データベースの列データ・タイプ											
	数値					文字			GRAPH	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0)[e]	(SBCS, DBCS)[b]	[b]	DATE	TIME	TIME STAMP
	E	E	E	E[a]	E							
-INTEGER		N										
	E[a]	E	E	E[a]	E							
-BIGINT			N									
	E[a]	E[a]	E	E[a]	E							
-DECIMAL				N								
	E[a]	E[a]	E[a]	E[a]	E							
-FLOAT					N							
	E[a]	E[a]	E[a]	E[a]	E							
文字												
-(0,0)						N						
						E	E w/F	E w/F	E w/F	E[c]	E[c]	E[c]
-(SBCS,0)							N	N				
						E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)							N w/F[d]	N		E[c]	E[c]	E[c]
						E	E w/F	E				
グラフィック												
						N w/F[d]			N			
						E			E			
日時												
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E

表 17. FORCEIN オプションを使用する場合の PC/IXF ファイルのインポートの要約 (続き)

PC/IXF の列データ・タイプ	データベースの列データ・タイプ											
	数値					文字			GRAPH	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0)[e]	(SBCS, DBCS)[b]	[b]	DATE	TIME	TIME STAMP

注: FORCEIN オプションを使用しなければ PC/IXF 列をデータベース列にインポートできない場合は、'N' または 'E' に 'w/F' を付けて示しています。 'N' は、ユーティリティーが新しいデータベース表を作成することを示します。'E' は、ユーティリティーが既存のデータベース表にデータをインポートすることを示します。 FORCEIN オプションは、文字およびグラフィック・データ・タイプの互換性にのみ影響を与えます。

[a] ターゲットの数値データ・タイプの範囲内にない値は、その特定の値に関して拒否されます。

[b] データ・タイプは DBCS 環境でのみ使用可能です。

[c] 有効な日付または時刻の値でない値は、その特定の値に関して拒否されます。

[d] ソース PC/IXF データ・タイプがターゲット・データベースによってサポートされない場合にのみ適用されます。

[e] データ・タイプは DBCS 環境では使用不能です。

PC/IXF およびバージョン 0 の System/370 IXF の相違

以下に、PC/IXF (データベース・マネージャーによって使用される) と、バージョン 0 System/370 の IXF (いくつかのホスト・データベース製品によって使用される) の間の相違点について説明します。

- PC/IXF ファイルは、EBCDIC 指向ではなく ASCII です。 PC/IXF ファイルでは、H レコードの新しいコード・ページ識別子や列記述子レコードでの実際のコード・ページ値の使用を含め、コード・ページ識別機能が大幅に拡張されています。さらに、文字データの列を FOR BIT DATA としてマークするためのメカニズムもあります。 PC/IXF ファイル形式と、その他の IXF またはデータベース・ファイル形式との間の変換では、FOR BIT DATA 列に含まれる値のコード・ページ変換ができないため、 FOR BIT DATA 列には特別な意義があります。
- マシン・データ形式のみ可能です。つまり、IXFTFORM フィールドの内容は常に `M` でなければなりません。さらに、マシン・データは PC 形式でなければなりません。つまり、IXFTMFRM フィールドの値は `PC` でなければなりません。したがって、PC/IXF データ・レコードのデータ部分の整数、浮動小数点数、および 10 進数は PC 形式でなければなりません。
- PC/IXF ファイルにおいて、H レコードの後であればどこにでもアプリケーション (A) レコードを入れることができます。それらは、IXFHHCNT フィールドの値の計算ではカウントされません。

- すべての PC/IXF レコードはレコード長標識で始まります。これは、PC/IXF レコードのうちレコード長標識自体を除く長さの整数値を 6 バイトの文字で表記したものです (つまり合計レコード長 - 6 バイト)。レコード長フィールドの目的は、PC プログラムがレコード境界を識別できるようにすることです。
- 可変長データによる記憶域の節約を活用にするため、またフィールドが複数のレコードに分割されている場合の複雑な処理を回避するため、 PC/IXF では、バージョン 0 の IXF X レコードはサポートされませんが、D レコード識別子はサポートされます。可変長フィールドまたはヌル可能フィールドがデータの D レコードの最後のフィールドである場合には、そのフィールドの最大長の全体を PC/IXF ファイルに書き込む必要はありません。

ワークシート・ファイル形式 (WSF)

ロータス 1-2-3 と Lotus Symphony の製品では、同じ基本形式が使用されており、それぞれの新しいリリースでの追加の機能があります。データベース・マネージャーでは、すべてのロータス製品について同じであるワークシート・レコードのサブセットがサポートされます。つまり、データベース・マネージャーによってサポートされるロータス 1-2-3 および Lotus Symphony 製品のリリースでは、3 文字の拡張子 (たとえば WKS、WK1、WRK、WR1、WJ2) の付いたすべてのファイル名が受け入れられます。

それぞれの WSF ファイルは 1 つのワークシートを表現します。データベース・マネージャーは、次の規則を使用してワークシートを解釈し、そのエクスポート操作によって生成されるワークシートに整合性を提供します。

- 最初の行 (ROW 値 0) のセルは、ワークシート全体に関する記述情報のために予約されています。この行のすべてのデータはオプションです。それらは、インポート時には無視されます。
- 2 番目の行 (ROW 値 1) のセルは、列のラベルとして使用されます。
- 残りの行は、データ行 (レコードまたは表のデータ行) です。
- ある列見出しの下にあるセル値は、その特定の列またはフィールドの値です。
- ヌル値は、セル内容レコードのうち特定の行の中で、特定の列に対応する実際のセル内容レコードがないことによって示されます (たとえば整数、数値、ラベル、または式レコードがない場合)。

  注: ヌルで構成される行は、インポートもエクスポートもされません。

エクスポート操作中に、WSF 形式に準拠したファイルを作成すると、いくらかのデータが失われることがあります。

WSF ファイルでは、ロータスのコード・ポイント・マッピングが使用されます。それは、DB2 によってサポートされる既存のコード・ページと必ずしも同じではありません。その結果、WSF ファイルのインポートまたはエクスポート時に、ロータスのコード・ポイントと、アプリケーションのコード・ページによって使用されるコード・ポイントの間でデータ変換が実行されます。 DB2 では、ロータスのコード・ポイントと、コード・ページ 437、819、850、860、863、および 865 によって定義されるコード・ポイントとの間の変換がサポートされています。

注: マルチバイト文字セット・ユーザーの場合、変換は実行されません。

付録D. 警告、エラー、および完了メッセージ

SQL メッセージの中には、さまざまなデータ移動ユーティリティーによって生成されるものが含まれます。それらのメッセージは、警告またはエラー条件が検出された場合にデータベース・マネージャーによって生成されます。各メッセージには、接頭部 (SQL) と 4 桁または 5 桁のメッセージ番号から構成されるメッセージ識別子があります。メッセージ・タイプには、通知、警告、および重大の 3 種類があります。 N で終わるメッセージ識別子は、エラー・メッセージです。 W で終わるメッセージ識別子は、警告または通知メッセージです。 C で終わるメッセージ識別子は、重大なシステム・エラーです。

メッセージ番号は *SQLCODE* とも呼ばれます。 SQLCODE は、そのメッセージ・タイプ (N、W、または C) に従って、正数または負数としてアプリケーションに渡されます。 N および C の場合は負の値、W の場合は正の値になります。 DB2 は SQLCODE をアプリケーションに戻し、アプリケーションでは SQLCODE に関連するメッセージを入手することができます。さらに DB2 は、SQL ステートメントの結果として発生することのある条件を表す *SQLSTATE* 値を戻します。一部の SQLCODE 値には、それに関連する SQLSTATE 値があります。

すべての DB2 メッセージについては、*メッセージ解説書* を参照してください。この資料に記載されている情報を使用すると、エラーまたは問題を識別し、適切な回復処置を使用することによって問題を解決することができます。また、この情報はメッセージが生成および記録された場所を理解するためにも使用できます。

SQL メッセージ、および SQLSTATE 値に関連するメッセージ・テキストについては、オペレーティング・システムのコマンド行で調べることもできます。これらのエラー・メッセージのヘルプを表示するには、オペレーティング・システムのコマンド・プロンプトで次のように入力します。

```
db2 ? SQLnnnnn
```

nnnnn はメッセージ番号です。

db2 コマンドのパラメーターとして受け入れられるメッセージ識別子には大文字小文字の区別がなく、最後の文字は省略可能です。したがって、以下のコマンドはどれも同じ結果になります。

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

メッセージ・テキストが長すぎて画面に入らない場合は、次のコマンドを使用してください (UNIX ベースのオペレーティング・システム、および ″more″ パイプをサポートするその他のオペレーティング・システムの場合)。

```
db2 ? SQLnnnnn | more
```

出力をファイルにリダイレクトし、後でそれを見ることもできます。

ヘルプは、対話式入力モードから呼び出すこともできます。このモードにアクセスするには、オペレーティング・システムのコマンド・プロンプトで次のように入力します。

```
db2
```

このモードで DB2 メッセージ・ヘルプを表示するには、コマンド・プロンプト (db2 =>) で次のように入力します。

```
? SQLnnnnn
```

SQLSTATE に関連するメッセージ・テキストは、次のコマンドを発行することによって検索できます。

```
db2 ? nnnnn
 または
db2 ? nn
```

nnnnn は 5 文字の SQLSTATE 値 (英数字)、*nn* は 2 桁の SQLSTATE クラス・コード (SQLSTATE 値の最初の 2 桁) です。

付録E. DB2 ライブラリーの使用法

DB2 ユニバーサル・データベース ライブラリーは、オンライン・ヘルプ、ブック (PDF および HTML)、および HTML 形式のサンプル・プログラムから成っています。このセクションでは、ユーザーに提供される情報について紹介し、その入手方法を示します。

オンライン製品情報をご利用になるには、インフォメーション・センターを使用することができます。詳細については、309ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。ここではタスク情報、DB2 ブック、トラブルシューティング情報、サンプル・プログラム、および Web の DB2 情報を見ることができます。

DB2 PDF ファイルおよびハードコピー版資料

DB2 情報

以下に示す表では、DB2 ブックを 4 つのカテゴリーに分類しています。

DB2 の手引きおよび解説書
これらの資料は、すべてのプラットフォームに共通の DB2 情報を含んでいます。

DB2 のインストールおよび構成の情報
これらの資料は、特定のプラットフォーム上の DB2 ごとに用意されています。たとえば、 OS/2、Windows、および UNIX ベースのプラットフォームで稼働するそれぞれの DB2 用に、別個の*概説およびインストール* 資料が用意されています。

プラットフォーム共通のサンプル・プログラム (HTML 形式)
これらのサンプルは、アプリケーション開発クライアントとともにインストールされるサンプル・プログラムの HTML 版です。これらのサンプルは参考用であり、実際のプログラムに代わるものではありません。

リリース情報
これらのファイルには、DB2 ブックには含められなかった最新の情報が記載されています。

インストール情報、リリース情報、およびチュートリアルは、製品 CD-ROM から HTML 形式で参照することができます。ほとんどの資料は、製品

CD-ROM から HTML 形式で表示できますし、 DB2 の資料 CD-ROM から Adobe Acrobat (PDF) 形式で表示し印刷することができます。 IBM にハードコピー版の資料を注文したい場合は、 305ページの『印刷資料の注文方法』を参照してください。注文可能な資料については、以下の表をご覧ください。

OS/2 および Windows プラットフォームの場合、 HTML ファイルは `sqllib¥doc¥html` ディレクトリーにインストールできます。 DB2 情報はいくつかの言語で提供されています。しかし、すべての言語に翻訳されているわけではありません。ある言語で情報が提供されていない場合は、英語版の情報が提供されます。

UNIX プラットフォームの場合、言語ごとに異なる複数の HTML ファイルを `doc/%L/html` ディレクトリーにインストールできます。ここで、*%L* は地域を表しています。詳細については、適切な「概説およびインストールの手引き」を参照してください。

DB2 ブックを入手して情報を利用するには、次のようなさまざまな方法があります。

- 308ページの『オンライン情報の表示』
- 313ページの『オンライン情報の検索』
- 305ページの『印刷資料の注文方法』
- 305ページの『PDF 資料の印刷』

表 18. DB2 情報

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
	DB2 の手引きおよび解説書情報		
管理の手引き	*管理の手引き: 計画* は、データベース概念について概説し、設計 (たとえば、論理および物理データベース設計) に関する情報を提供し、高い可用性について解説しています。 *管理の手引き: インプリメンテーション* は、設計、データベースへのアクセス、監査、バックアップ、および回復などのインプリメンテーションについて説明しています。 *管理の手引き: パフォーマンス* は、データベース環境について解説し、さらにアプリケーションのパフォーマンスの評価と調整の方法について説明しています。	「第 1 巻」 SC88-8513 db2d1x70 「第 2 巻」 SC88-8511 db2d2x70 「第 3 巻」 SC88-8512 db2d3x70	db2d0
管理 API 解説書	データベースの管理に使用できる DB2 アプリケーション・プログラミング・インターフェース (API) およびデータ構造について説明します。また、この資料は、アプリケーションから API を呼び出す方法も示します。	SC88-8514 db2b0x70	db2b0
アプリケーション構築の手引き	環境設定に関する情報を提供し、Windows、OS/2、および UNIX ベースのプラットフォームでの DB2 アプリケーションのコンパイル、リンク、実行の各ステップについて説明します。	SC88-8515 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 ユニバーサル・データベース製品をご使用中に発生する可能性のあるセンス・コード APPC、CPI-C、および SNA についての一般情報を提供します。 HTML 形式でのみご利用いただけます。	資料番号なし db2apx70	db2ap

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
アプリケーション開発の手引き	DB2 データベースにアクセスするアプリケーションを、組み込み SQL または Java (JDBC および SQLJ) を使用して開発する方法について説明します。さらに、ストアード・プロシージャーの作成方法、ユーザー定義関数の作成方法、ユーザー定義タイプの作成方法、トリガーの使用方法、区画化されている環境または統合されているシステムでのアプリケーションの開発方法などについて解説されています。	SC88-8516 db2a0x70	db2a0
コール・レベル・インターフェースの手引きおよび解説書	DB2 データベースにアクセスするアプリケーションを、 DB2 コール・レベル・インターフェース (Microsoft ODBC 仕様互換の呼び出し可能 SQL) を使用して開発する方法について説明します。	SC88-8517 db2l0x70	db2l0
コマンド解説書	コマンド行プロセッサーの使用法について説明し、データベースの管理に使用できる DB2 コマンドについて解説しています。	SC88-8518 db2n0x70	db2n0
コネクティビティー 補足	DB2 (AS/400 版)、DB2 (OS/390 版)、DB2 (MVS 版)、または DB2 (VM 版) を DRDA アプリケーション・リクエスターとして DB2 ユニバーサル・データベースとともに使用するためのセットアップ情報および参照情報を提供します。また、この資料は DRDA アプリケーション・サーバーを DB2 コネクト アプリケーション・リクエスターとともに使用する方法の詳細を示します。 HTML と PDF でのみ利用可能	資料番号なし db2h1x70	db2h1
データ移動ユーティリティー 手引きおよび解説書	データの移動を行う DB2 ユーティリティー (インポート、エクスポート、ロード、AutoLoader、および DPROP など) の使用法について説明しています。	SC88-8522 db2dmx70	db2dm

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
データウェアハウスセンター 管理の手引き	データウェアハウスセンターを使用してデータウェアハウスを構築および保守する方法を説明します。	SC88-8545 db2ddx70	db2dd
データウェアハウスセンター アプリケーション統合の手引き	プログラマーがアプリケーションをデータウェアハウスセンターおよび情報カタログ・マネージャーと統合するのに役立つ情報を提供します。	SC88-8546 db2adx70	db2ad
DB2 コネクト 使用者の手引き	DB2 コネクト製品の概念、プログラミング、および一般的な使用方法に関する情報を提供します。	SC88-8521 db2c0x70	db2c0
DB2 クエリー・パトローラー 管理の手引き	DB2 クエリー・パトローラー・システムの運用の概説を行い、運用および管理に関する詳細情報、および管理用グラフィカル・ユーザー・インターフェース・ユーティリティーについてのタスク情報を提供します。	SC88-8525 db2dwx70	db2dw
DB2 クエリー・パトローラー 使用者の手引き	DB2 クエリー・パトローラーのツールや関数の使用方法を説明します。	SC88-8527 db2wwx70	db2ww
用語集	DB2 およびその構成要素で使用される用語の定義を示します。 HTML 形式と SQL 解説書 で利用可能	資料番号なし db2t0x70	db2t0
イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミングの手引き	DB2 エクステンダーの一般情報について提供し、画像、音声、およびビデオ (IAV) エクステンダーの管理と構成について、および IAV エクステンダーを使用したプログラミングについて説明しています。さらに、参照情報、診断情報 (メッセージ解説)、およびサンプルも収録されています。	SC88-8609 dmbu7x70	dmbu7
情報カタログ・マネージャー 管理の手引き	情報カタログを管理するためのガイドです。	SC88-8547 db2dix70	db2di
情報カタログ・マネージャー プログラミングの手引きおよび解説書	情報カタログ・マネージャー用の体系化されたインターフェースの定義を示します。	SC88-8549 db2bix70	db2bi

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
情報カタログ・マネージャー 使用者の手引き	情報カタログ・マネージャー・ユーザー・インターフェースの使用に関する情報を提供します。	SC88-8548 db2aix70	db2ai
インストールおよび構成補足	プラットフォーム固有の DB2 クライアントの計画、インストール、およびセットアップのガイドです。この補足資料には、バインド、クライアント / サーバー通信の設定、 DB2 GUI ツール、DRDA AS、分散インストール、分散要求の構成、および異種データ・ソースへのアクセスについても説明されています。	GC88-8524 db2iyx70	db2iy
メッセージ解説書	DB2、情報カタログ・マネージャー、およびデータウェアハウスセンターから出されるメッセージとコードをリストし、取るべき処置を解説しています。	第 1 巻 GC88-8543 db2m1x70 第 2 巻 GC88-8544 db2m2x70	db2m0
OLAP Integration Server Administration Guide	OLAP Integration Server の Administration Manager 構成要素の使用方法を説明します。	SC27-0782 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	標準の OLAP Metaoutline インターフェースを使用して (Metaoutline Assistant を使用するのではなく) OLAP metaoutline を作成しデータを取り込む方法を説明しています。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	(Model Assistant ではなく) 標準的な OLAP Model Interface を使用して OLAP モデルを作成する方法を説明します。	SC27-0783 db2lpx70	n/a
OLAP Setup and User's Guide	OLAP Starter Kit の構成およびセットアップに関する情報を提供します。	SC27-0702 db2ipx70	db2ip

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
OLAP Spreadsheet Add-in User's Guide for Excel	Excel 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC27-0786 db2epx70	db2ep
OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3	ロータス 1-2-3 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC27-0785 db2tpx70	db2tp
レプリケーションの手引きおよび解説書	DB2 に付属の IBM レプリケーション・ツールの計画、構成、管理、および使用方法に関する情報を提供します。	SC88-8550 db2e0x70	db2e0
地理情報エクステンダー使用者の手引きおよび解説書	地理情報エクステンダーのインストール、構成、管理、プログラミング、およびトラブルシューティングに関する情報を提供します。また、地理情報データの概念についての重要事項を示し、地理情報エクステンダー固有の参照情報 (メッセージおよび SQL) を提供します。	SC88-8624 db2sbx70	db2sb
SQL 概説	SQL の概念を紹介し、構造体とタスクの例を多数提供しています。	SC88-8539 db2y0x70	db2y0
SQL 解説書	SQL の構文、セマンティクス、および言語規則について説明します。また、この資料には、各リリース間の互換性、製品の制限事項、およびカタログ・ビューも含まれます。	第 1 巻 SC88-8540 db2s1x70 第 2 巻 SC88-8657 db2s2x70	db2s0
システム・モニター 手引きおよび解説書	データベースおよびデータベース・マネージャーに関連したさまざまな情報を収集する方法を示します。この資料は、この情報を利用して、データベース活動の把握、パフォーマンス向上、および問題原因の判別を行う方法を説明しています。	SC88-8523 db2f0x70	db2f0

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
テキスト・エクステンダー *管理およびプログラミング*	DB2 エクステンダーの一般情報、テキスト・エクステンダーの管理および構成情報、およびテキスト・エクステンダーを使用したプログラミングの方法について解説します。この資料には、参照情報、診断情報 (メッセージ解説)、およびサンプルが含まれています。	SC88-8610 desu9x70	desu9
問題判別の手引き	エラーの原因の判別、問題からの回復、および DB2 カスタマー・サービスの支援の下での診断ツールの使用法を記載しています。	GD88-7271 db2p0x70	db2p0
新機能	DB2 ユニバーサル・データベース バージョン 7 の新しい機能および拡張機能について説明します。	SC88-8541 db2q0x70	db2q0
DB2 のインストールおよび構成の情報			
DB2 コネクト エンタープライズ・エディション (OS/2 および Windows 版) 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム版の DB2 コネクト エンタープライズ・エディションで、計画、移行、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8520 db2c6x70	db2c6
DB2 コネクト エンタープライズ・エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 コネクト エンタープライズ・エディションの計画、移行、インストール、構成、およびタスクに関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8519 db2cyx70	db2cy

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 コネクト パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システムの DB2 コネクト パーソナル・エディションで、計画、移行、インストール、および構成を行う場合のタスク情報を提供します。また、この資料はサポートされているすべてのクライアントのインストールおよびセットアップについても説明します。	GC88-8533 db2c1x70	db2c1
DB2 コネクト パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 コネクト パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8528 db2c4x70	db2c4
DB2 データ・リンク・マネージャー (Windows 版) 概説およびインストール	AIX および Windows 32 ビット・オペレーティング・システムの DB2 データ・リンク・マネージャーで、計画、インストール、構成を行う場合の情報を提供します。	GC88-8532 db2z6x70	db2z6
DB2 エンタープライズ拡張エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 エンタープライズ拡張エディションの計画、インストール、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8530 db2v3x70	db2v3
DB2 エンタープライズ拡張エディション (Windows 版) 概説およびインストール	Windows 32 ビット・オペレーティング・システムの DB2 エンタープライズ拡張エディションで、計画、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8529 db2v6x70	db2v6

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 ユニバーサル・データベース (OS/2 版) 概説およびインストール	OS/2 オペレーティング・システムでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8534 db2i2x70	db2i2
DB2 ユニバーサル・データベース (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。 また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8536 db2ixx70	db2ix
DB2 ユニバーサル・データベース (Windows 版) 概説およびインストール	Windows 32 ビット オペレーティング・システムの DB2 ユニバーサル・データベースで、計画、インストール、移行、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8537 db2i6x70	db2i6
DB2 パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム 版の DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8535 db2i1x70	db2i1
DB2 パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 ユニバーサル・データベース・パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8538 db2i4x70	db2i4
DB2 クエリー・パトローラー インストールの手引き	DB2 クエリー・パトローラーのインストール情報を提供します。	GC88-8526 db2iwx70	db2iw

表 18. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
ウェアハウス・マネージャー *インストールの手引き*	ウェアハウス・エージェント、ウェアハウス・トランスフォーマー、および情報カタログ・マネージャーのインストール情報を提供します。	GC88-8572 db2idx70	db2id
プラットフォーム共通のサンプル・プログラム (HTML 形式)			
サンプル・プログラム (HTML)	DB2 のサポートするすべてのプラットフォームでのプログラム言語用に、サンプル・プログラム (HTML 形式) を提供します。これらのサンプル・プログラムは、参照用としてのみ提供されています。サンプルは、すべてのプログラミング言語で利用できるわけではありません。HTML サンプルが利用できるのは、 DB2 アプリケーション開発クライアントがインストールされている場合だけです。 プログラムの詳細については、*アプリケーション構築の手引き* を参照してください。	資料番号なし	db2hs
リリース情報			
DB2 コネクト リリース情報	DB2 コネクトの資料には含められなかった最新の情報が収録されています。	注 #2 を参照してください。	db2cr
DB2 インストール情報	DB2 ブックには含められなかったインストールに関する最新の情報が収録されています。	製品 CD-ROM からのみ利用できます。	
DB2 リリース情報	DB2 ブックには含められなかった DB2 製品とその機能に関する最新の情報が収録されています。	注 #2 を参照してください。	db2ir

注:

1. ファイル名の 6 桁目の文字 *x* は、その資料の言語を表します。たとえば、ファイル名 `db2d0e70` は、*管理の手引き* の英語版であることを示し、ファイル名 `db2d0f70` は同じ資料のフランス語版を示します。資料の言語を表すためにファイル名の 6 桁目で使用されている文字は以下のとおりです。

言語	識別子
ブラジル・ポルトガル語	b
ブルガリア語	u
チェコ語	x
デンマーク語	d
オランダ語	q
英語	e
フィンランド語	y
フランス語	f
ドイツ語	g
ギリシャ語	a
ハンガリー語	h
イタリア語	i
日本語	j
韓国語	k
ノルウェー語	n
ポーランド語	p
ポルトガル語	v
ロシア語	r
簡体字中国語	c
スロベニア語	l
スペイン語	z
スウェーデン語	s
繁体字中国語	t
トルコ語	m

2. DB2 ブックには含められなかった最新の情報が、「リリース情報」で HTML 形式および ASCII ファイルとして利用できます。 HTML 版は、インフォメーション・センターおよび製品 CD-ROM からご利用になれます。 ASCII ファイルの参照方法:
   - UNIX ベースのプラットフォームでは、ファイル `Release.Notes` を参照してください。このファイルは `DB2DIR/Readme/%L` ディレクトリーにあります。ここで `%L` は地域名を、 `DB2DIR` は以下のものを表します。
     - `/usr/lpp/db2_07_01` (AIX の場合)
     - `/opt/IBMdb2/V7.1` (HP-UX、DYNIX/ptx、Solaris、および Silicon Graphics IRIX の場合)
     - `/usr/IBMdb2/V7.1` (Linux の場合)
   - これ以外のプラットフォームでは、ファイル `RELEASE.TXT` を参照してください。このファイルは、製品がインストールされているディレクトリーにあります。 OS/2 プラットフォームでは、 **IBM DB2** フォルダーをダブルクリックし、 **Release Notes** アイコンをダブルクリックすることもできます。

PDF 資料の印刷

資料のハードコピー版が必要な場合、 DB2 の資料 CD-ROM にある PDF ファイルを印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷することができます。ライブラリー内の各資料のファイルについては、295ページの表18 を参照してください。

Adobe Acrobat Reader の最新版は、Adobe の Web サイト http://www.adobe.com から入手できます。

PDF ファイルは、DB2 の資料 CD-ROM に収録されており、ファイル拡張子 PDF が付いています。 PDF ファイルにアクセスするには以下のようにします。

1. DB2 の資料 CD-ROM を挿入します。 UNIX ベースのプラットフォームの場合は、 DB2 資料 CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
2. Acrobat Reader を起動します。
3. 以下に示すいずれかの位置から必要な PDF ファイルを開きます。
   - OS/2 および Windows プラットフォームでは:

     x:¥doc¥*language* ディレクトリー。ここで、*x* は CD-ROM ドライブを、*language* は 2 桁の言語を表す国コード (たとえば、EN は英語) を示します。
   - UNIX ベースのプラットフォームでは:

     CD-ROM の */cdrom*/doc/*%L* ディレクトリー。ここで、*/cdrom* は CD-ROM のマウント・ポイントを、 *%L* は地域名を表します。

さらに、PDF ファイルを CD-ROM からローカル・ドライブまたはネットワーク・ドライブにコピーし、そこから参照することもできます。

印刷資料の注文方法

ハードコピー版の DB2 ブックは、個別に注文することができます。資料を注文するには、IBM 承認の販売業者または営業担当員に連絡してください。

DB2 オンライン文書

オンライン・ヘルプへのアクセス

すべての DB2 構成要素で、オンライン・ヘルプを利用できます。以下の表に、さまざまな種類のヘルプを示します。

ヘルプの種類	内容	利用方法
コマンド・ヘルプ	コマンド行プロセッサーのコマンド構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? command` ここで *command* はキーワードまたはコマンド全体を表します。 たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、`? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプが表示されます。
クライアント構成アシスタントのヘルプ コマンド・センターのヘルプ コントロール・センターのヘルプ データウェアハウスセンターのヘルプ イベント・アナライザーのヘルプ 情報カタログ・マネージャーのヘルプ サテライト管理センターのヘルプ スクリプト・センターのヘルプ	そのウィンドウまたはノートブックで実行できるタスクについて説明します。このヘルプは、知っておく必要のある概説および前提条件に関する情報を含みます。また、ウィンドウやノートブックの制御の使用方法を示します。	ウィンドウまたはノートブックから、**「ヘルプ (Help)」**押しボタンをクリックするか、または **F1** キーを押します。

ヘルプの種類	内容	利用方法

メッセージ・ヘルプ	メッセージの原因、および取るべき処置を説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。
		`? XXXnnnnn`
		ここで、*XXXnnnnn* は有効なメッセージ識別子を表します。
		たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。
		一度に 1 画面分のメッセージ・ヘルプを表示させるには、次のように入力します。
		`? XXXnnnnn | more`
		メッセージ・ヘルプをファイルに保管するには、次のように入力します。
		`? XXXnnnnn > filename.ext`
		ここで、*filename.ext* はメッセージ・ヘルプを保管するファイルを表します。

SQL ヘルプ	SQL ステートメントの構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `help statement` ここで、*statement* は SQL ステートメントを表します。 たとえば、`help SELECT` と入力すると、 SELECT ステートメントのヘルプが表示されます。 **注:** UNIX ベースのプラットフォームでは、SQL ヘルプを利用できません。
SQLSTATE ヘルプ	SQL 状態およびクラス・コードについて説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? sqlstate or ? class code` ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。 たとえば、`? 08003` によって SQL 状態 08003 のヘルプが表示され、 `? 08` によってクラス・コード 08 のヘルプが表示されます。

オンライン情報の表示

この製品に付属のブックは、ハイパーテキスト・マークアップ言語 (HTML) ソフトコピー形式です。ソフトコピー形式では情報を検索または表示したり、ハイパーテキスト・リンクを利用して関連情報に移動したりすることができます。また、1 つの端末を超えてライブラリーを容易に共用することができます。

オンライン・ブックやサンプル・プログラムは、 HTML バージョン 3.2 仕様に準拠するすべてのブラウザーを使って表示できます。

オンライン・ブックまたはサンプル・プログラムは、次のようにして表示します。

- DB2 管理ツールを実行している場合、インフォメーション・センターを使用します。
- ブラウザーで、**ファイル (File) —>ページを開く (Open Page)** をクリックします。次のようなページを開いて、DB2 情報に関する説明とリンクを表示してください。
  - UNIX ベースのプラットフォームでは、以下のページを開きます。

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    ここで *%L* はロケール名です。
  - その他のプラットフォームでは、以下のページを開きます。

    sqllib¥doc¥html¥index.htm

    パスは DB2 がインストールされているドライブです。

    インフォメーション・センターをインストールしていない場合、 **DB2 Information** アイコンをダブルクリックしてページを開くことができます。このアイコンは、ご使用のシステムに応じて、製品のメイン・フォルダー内または Windows 「スタート」メニューにあります。

Netscape ブラウザーのインストール

システムに Web ブラウザーがインストールされていない場合、製品の箱の中にある Netscape CD-ROM から Netscape をインストールすることができます。インストールに関する詳細な説明については、以下を参照してください。

1. Netscape CD-ROM を挿入します。
2. UNIX ベースのプラットフォームでは、CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。

3. インストールの手順については、 CDNAV*nn*.txt ファイルを参照します。ここで、*nn* は 2 桁の言語識別子を表します。ファイルは CD-ROM のルート・ディレクトリーにあります。

インフォメーション・センターを使用した情報へのアクセス

インフォメーション・センターを使用すると、DB2 製品情報にすばやくアクセスすることができます。インフォメーション・センターは、DB2 管理ツールを使用できるすべてのプラットフォームで利用できます。

インフォメーション・センターは「インフォメーション・センター (Information Center)」アイコンをダブルクリックすることによってオープンできます。このアイコンのある場所はシステムによって異なります。メイン・プロダクト・フォルダーか Windows の**「スタート」**メニューのどちらかです。

Windows プラットフォームの DB2 では、ツールバーおよび**ヘルプ**・メニューを使用して、インフォメーション・センターにアクセスすることもできます。

インフォメーション・センターは 6 種類の情報を提供します。適切なタブをクリックすると、種類ごとに提供されているトピックが表示されます。

タスク (Tasks)
DB2 を使用して実行できる主要なタスク。

参照 (Reference)
DB2 参照情報 (キーワード、コマンド、API など)。

ブック (Books)
DB2 ブック。

トラブルシューティング (Troubleshooting)
エラー・メッセージのカテゴリーと、メッセージに対する回復処置。

サンプル・プログラム (Sample Programs)
DB2 アプリケーション開発クライアントに付属のサンプル・プログラム。 DB2 アプリケーション開発クライアントをインストールしていない場合、このタブは表示されません。

Web
WWW 上にある DB2 情報。この情報にアクセスするには、ご使用のシステムから Web への接続が必要です。

リストから項目を 1 つ選択すると、インフォメーション・センターはビューアーを立ち上げて情報を表示します。選択した情報の種類に応じて、ビューアーはシステム・ヘルプ・ビューアー、エディター、または Web ブラウザーです。

インフォメーション・センターには検索機能が備わっており、リストを参照せずに特定のトピックを探すことができます。

テキストの全検索を行うには、インフォメーション・センター内のハイパーテキスト・リンク「**DB2 オンライン情報の検索 (Search DB2 Online Information)**」検索フォームに従います。

通常、HTML 検索サーバーは自動的に始動します。 HTML 情報の検索がうまくいかない場合は、以下の方法の 1 つを使用して、検索サーバーを始動しなければならない場合もあります。

Windows では

「スタート」をクリックし、**「プログラム」—>「IBM DB2」—>「Information」—>「Start HTML Search Server」**を選択します。

OS/2 では

「DB2 (OS/2 版)」フォルダーをダブルクリックして、**「Start HTML Search Server」**アイコンをダブルクリックします。

HTML 情報の検索でこの他の問題が発生した場合は、リリース情報を参照してください。

注: 検索機能は、Linux、DYNIX/ptx、および Silicon Graphics IRIX 環境では利用できません。

DB2 ウィザードの使用

ウィザードを使用すると、各タスクをステップごとに進めることによって、さまざまな管理タスクを遂行することができます。ウィザードは、コントロール・センターおよびクライアント構成アシスタントを通して使用できます。以下の表では、ウィザードとその目的をリストしています。

注: データベース作成、索引作成、複数サイト更新の構成、およびパフォーマンス構成ウィザードは、区分データベース環境で使用できます。

ウィザード	内容	利用方法
データベース追加 (Add Database)	クライアント・ワークステーション上にデータベースのカタログを作成します。	クライアント構成アシスタントから、「**追加 (Add)**」をクリックします。

ウィザード	内容	利用方法
データベース・バックアップ *(Back up Database)*	バックアップ計画を決定、作成、およびスケジュールします。	「コントロール・センター (Control Center)」からバックアップするデータベースを右クリックし、**「バックアップ (Backup)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
複数サイト更新の構成 (Configure Multisite Update)	複数サイト更新、分散トランザクション、または 2 フェーズ・コミットを構成します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「複数サイト更新 (Multisite Update)」**を選択します。
データベース作成 (Create Database)	データベースを作成し、いくつかの基本的な構成タスクを実行します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「作成 (Create)」—> 「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
表作成 (Create Table)	基本的なデータ・タイプを選択して、表の基本キーを作成します。	「コントロール・センター (Control Center)」から、**「表 (Tables)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する表 (Table Using Wizard)」**を選択します。
表スペース作成 (Create Table Space)	新しい表スペースを作成します。	「コントロール・センター (Control Center)」から、**「表スペース (Table Spaces)」**アイコンを右クリックして、**「作成 (Create)」—> 「ウィザードを使用する表スペース (Table Space Using Wizard)」**を選択します。
索引作成 (Create Index)	すべての照会について、作成すべき索引および除去すべき索引を提案します。	「コントロール・センター (Control Center)」から、**「索引 (Index)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する索引 (Index Using Wizard)」**を選択します。

ウィザード	内容	利用方法
パフォーマンス構成 *(Performance Configuration)*	ビジネス要件に適合するように構成パラメーターを更新して、データベースのパフォーマンスを調整します。	「コントロール・センター (Control Center)」から、調整したいデータベースを右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。 区分データベース環境では、「Database Partitions」視点から、調整したい最初のデータベース区画を右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。
データベース復元 *(Restore Database)*	障害の後、データベースを回復します。どのバックアップを使用し、どのログを再生するかの判別を支援します。	「コントロール・センター (Control Center)」から復元するデータベースを右クリックし、**「復元 (Restore)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。

文書サーバーのセットアップ

デフォルトでは、DB2 情報はローカル・システムにインストールされます。つまり、DB2 情報にアクセスする必要のある各担当者が同じファイルをインストールする必要があります。 DB2 情報を 1 か所に格納するには、次のようにします。

1. `¥sqllib¥doc¥html` のすべてのファイルとサブディレクトリーを、ローカル・システムから Web サーバーにコピーします。各ブックには独自のサブディレクトリーがあり、そのブックを構成する必要な HTML および GIF ファイルが入っています。ディレクトリー構造は常に同じ状態に保つ必要があります。
2. Web サーバーを構成して、ファイルを新しい場所で検索するようにします。さらに詳しい情報については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。
3. インフォメーション・センターの Java バージョンをご使用の場合は、すべての HTML ファイルのベース URL を指定できます。この URL はブックのリストに使用してください。

4. 資料ファイルが表示されるようになったなら、よく使うトピックにはブックマークを付けておいてください。ブックマークを付けるページは、たとえば以下のものがあります。
   - ブックのリスト
   - 頻繁に使用されるブックの目次
   - 頻繁に参照する情報 (たとえば、ALTER TABLE トピックなど)
   - 検索フォーム

中央のマシンから DB2 ユニバーサル・データベース オンライン文書ファイルを提供する方法については、 *インストールおよび構成 補足* の NetQuestion Appendix を参照してください。

オンライン情報の検索

HTML ファイルの情報を検索するには、以下の方法のどれか 1 つを使用してください。

- 最上部にある**「検索 (Search)」**をクリックします。検索フォームを使用して特定のトピックを見つけます。この機能は、 Linux、DYNIX/ptx、または Silicon Graphics IRIX 環境ではご利用になれません。
- 最上部にある**「索引 (Index)」**をクリックします。索引を使用して、ブック内の特定のトピックを見つけます。
- HTML 資料またはヘルプの目次あるいは索引を表示してから、 Web ブラウザーの検索機能を利用して資料内の特定のトピックを見つけます。
- Web ブラウザーのブックマーク機能を使用して、特定のトピックにすばやく戻ります。
- インフォメーション・センターの検索機能を使用して、特定のトピックを検索します。詳しくは、309ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。

付録F. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミングまたはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、 IBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品のみが使用可能であることを意味するものではありません。これらのプログラムまたは製品に代えて、 IBM の知的所有権を侵害することのない機能的に同等な他社のプログラム、製品またはサービスを使用することができます。ただし、IBM によって明示的に指定されたものを除き、これらのプログラムまたは製品に関連する稼働の評価および検証はお客様の責任で行っていただきます。

IBM および他社は、本書で説明する主題に関する特許権 (特許出願を含む)、商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

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

本書に含まれる情報には、技術的に不正確なもの、または誤植が含まれる場合があります。これらに対する変更は、定期的に行われます。これらの変更は、資料の改訂版に含まれます。 IBM は、本書で説明している製品、プログラムに対して、予告なく改良、変更を加える場合があります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するもので

はありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様になんら義務も負わせない適切な方法で、使用もしくは配布することがあります。

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

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

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

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

本書に含まれるパフォーマンス・データは、制御された環境下で決定されています。したがいまして、その他の稼働環境で得られる結果とは、かなり異なる可能性もあります。一部の測定値は、開発中のシステムを使用している場合があり、これらの測定値が一般的に提供可能なシステムで同様の数値になることを保証するものではありません。さらに、一部の測定値が推定されたものもあります。実測値と異なる場合があります。本書のユーザーは、使用される特定の環境での該当データを確認してください。

IBM 以外の製品については、当該製品の提供者から直接、出版されている資料または一般公開されている情報から入手しました。 IBM は、これらの製品についてはテストを行っておらず、これらの IBM 以外の製品に関する性能、互換性またはその他の主張について確認することはできません。 IBM 以外の製品の機能に対する質問は、それぞれの製品提供者にお問い合わせください。

IBM の将来の方向性または意図については、予告なしに変更または中止する場合があり、 IBM の目的および目標のみを示しているものです。

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

著作権：

本書に含まれる情報には、サンプル・アプリケーション・プログラムがソース言語の形式で含まれており、様々な、オペレーティング・プラットフォームでのプログラミング技法を示しています。お客様は、これらのサンプル・プログラムが書かれているオペレーティング・プラットフォームでアプリケーション・プログラミング・インターフェースが実行可能となるためのアプリケーション・プログラムを開発、使用、販売または配布もしくは転送する目的のためにのみ、サンプル・プログラムを、IBM に対する別途料金を支払うことなく、複製、変更、配布または転送することができます。これらのサンプルは、すべての条件下で十分にテストを行っていません。したがいまして、IBM は、これらのプログラムの信頼性、実用性または機能について、いかなる保証も負いません。

サンプル・プログラムまたはその改変版の複製物には、全部複製か部分複製かを問わず、次の著作権表示を必ず行うものとします。

商標

次のものは、 IBM Corporation の米国およびその他の国における商標です。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

次のものは、他社の商標または登録商標です。

Tivoli および NetView は、米国およびその他の国における Tivoli Systems Inc. の商標です。

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

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

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

索引

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

[ア行]

アプリケーション・レコード、PC/IXF 249
一時ファイル
  ロード・ユーティリティー 106, 148
移動、データの
  プラットフォーム間 197
インストール
  Netscape ブラウザー 308
インフォメーション・センター 309
インポート
  ファイル・タイプ修飾子 60
  PC/IXF ファイルから表へ 39
  PC/IXF ファイルの、FORCEIN を使用する場合 286
インポート、PC/IXF ファイルの
  一般規則 273
  データ・タイプ固有の規則 275
インポートのメッセージ・ファイル 26, 73, 168
インポート・ユーティリティー
  エクスポートされた表の再作成 35
  概要 28
  クライアント / サーバー 30
  コード・ページについての考慮事項 70
  識別列 31
  使用に必要な権限と特権 29
  制限 72
  生成列 33
  バッファー挿入 31
  パフォーマンス 72
インポート・ユーティリティー *(続き)*
  パフォーマンスの最適化 72
  表のロッキング 37
  ホストおよびワークステーション間でデータを転送 205
  ユーザー定義特殊タイプ (UDT) 37
  ラージ・オブジェクト (LOB) 36
  リモート・データベース 30
  ロード・ユーティリティーとの比較 221
  DB2 データ・リンク・マネージャー 194
インポート・ユーティリティーのファイル形式 225
ウィザード
  索引 311
  タスクを遂行する 310
  データベース作成 311
  データベース追加 310, 311, 312
  データベース復元 312
  データベース・バックアップ 310
  パフォーマンス構成 311
  表作成 311
  表スペース作成 311
  複数サイト更新の構成 311
エクスポート
  ファイル・タイプ修飾子 18
  列名の指定 12
エクスポート操作
  データウェアハウスセンターでサポートされる 215
エクスポートのメッセージ・ファイル 26, 73, 168
エクスポート・ユーティリティー
  エクスポートされた表の再作成 4
  概要 2
  識別列 4
エクスポート・ユーティリティー *(続き)*
  使用に必要な権限と特権 3
  制約事項 25
  ホストおよびワークステーション間でデータを転送 205
  ラージ・オブジェクト (LOB) 5
  DB2 データ・リンク・マネージャー 190
  db2batch を使った並列エクスポート 5
エクスポート・ユーティリティーのファイル形式 225
エラー・メッセージ 291
オートローダー・ユーティリティー
  概要 169
  使用に必要な権限と特権 171
  制限 186
  制約事項 186
  トラブルシューティング 186
オプション
  forcein 279
オンライン情報
  検索 313
  表示 308
オンライン・ヘルプ 305

[カ行]

階層レコード、PC/IXF 252
回復可能データベース
  使用するロード・オプション 79
回復不能データベース
  使用するロード・オプション 79
環境変数
  DB2LOADREC 92
完了メッセージ 291
キーワード
  構文 217
規則、PC/IXF ファイルのインポートを制御する 273, 275

区切り付き ASCII (DEL) ファイル形式 226
プラットフォーム間のデータの移動 198
区切りなし ASCII (ASC) ファイル形式 231
区切り文字
文字ストリング 228
区分化、データの
オートローダー・ユーティリティー 169
区分化キー 169
警告メッセージ 291
継続レコード、PC/IXF 255
権限
インポート・ユーティリティーに必要な 29
エクスポート・ユーティリティーに必要な 3
オートローダー・ユーティリティーに必要な 171
必要、ロード・ユーティリティーに 83
言語識別子
ブック 303
検索
オンライン情報 310, 313
コード・ページについての考慮事項
インポート・ユーティリティー 70
ロード・ユーティリティー 149
コード・ページ変換
ファイル 274
PC/IXF データのインポートまたはロード時 274
構造
区切り付き ASCII (DEL) 226
区切りなし ASCII (ASC) ファイル 232
構文図 217
互換性のない列 274
コマンド構文
解釈 217

[サ行]

再作成、エクスポートされた表
インポート・ユーティリティー 35
エクスポート・ユーティリティー 4
IXF ファイルに格納されない表属性 36
IXF ファイルに格納される表属性 35
最新情報 304
索引ウィザード 311
索引レコード、PC/IXF 250
サンプル・プログラム
プラットフォーム共通の 303
HTML 303
識別列 4, 31, 86
識別列でない生成列 33, 88
識別レコード、PC/IXF 257
修飾子、ファイル・タイプ
インポート・ユーティリティー 60
エクスポート・ユーティリティー 18
ロード・ユーティリティー 133
終了レコード、PC/IXF 256
状態
検査保留 161
削除保留 161
バックアップ保留 161
ロード保留 161
ストアード・プロシージャー
変換機能 216
生成列 33, 88
制約検査 90
セットアップ、文書サーバーの 312
セマンティクス
FORCEIN、一般 279
FORCEIN、コード・ページ 279
FORCEIN、データ・タイプ 285
相違点、PC/IXF とSystem/370 IXF の間の 288
走査順序 28
タイプ表 208
デフォルト 209
走査順序 28 *(続き)*
ユーザー指定 209

[タ行]

タイプ表
インポート・ユーティリティー 207
エクスポート・ユーティリティー 207
走査順序 28, 208
データ移動中の選択 209
データ移動の例 210
データの移動 207
ダンプ・ファイル
ロード・ユーティリティー 148
データウェアハウスセンター (DWC)
概要 214
データ転送
オートローダー・ユーティリティー 169
プラットフォーム間 197
ホストおよびワークステーション間の 205
データベース
ウェアハウス 214
回復可能および回復不能、使用するロード・オプション 79
データベース移動ツール 200
データベース作成ウィザード 311
データベース追加ウィザード 310, 311, 312
データベース・バックアップ・ウィザード 310
データ・タイプ
PC/IXF 259
データ・タイプの説明
ASC 233
DEL 228
PC/IXF 266
データ・レコード、PC/IXF 248
統合交換フォーマット (IXF) 236
特権
インポート・ユーティリティーに必要な 29

特権 *(続き)*
エクスポート・ユーティリティーに必要な 3
オートローダー・ユーティリティーに必要な 171
必要、ロード・ユーティリティーに 83

［ハ行］

ハッシュ・アルゴリズム 170
バッファー挿入
インポート・ユーティリティー 31
パフォーマンス
インポート・ユーティリティー 72
ロード・ユーティリティー 162
パフォーマンス構成ウィザード 311
パラメーター
構文 217
表
エクスポートされた、再作成 35
表作成ウィザード 311
表示
オンライン情報 308
標識
レコード長 237
表スペース作成ウィザード 311
表レコード、PC/IXF 240
表ロード削除開始ログ・レコード 149
ファイル形式
区切り付き ASCII (DEL) 226
区切りなし ASCII (ASC) 231
表からファイルへのエクスポート 9
表へのファイルのインポート 44
ワークシート (WSF) 289
PC バージョンの IXF (PC/IXF) 236
ファイル・タイプ修飾子
インポート・ユーティリティー 60
エクスポート・ユーティリティー 18
ファイル・タイプ修飾子 *(続き)*
ロード・ユーティリティー 133
復元ウィザード 312
複数サイト更新の構成ウィザード 311
副表レコード、PC/IXF 253
ブック 293, 305
分割、データの 169
並列エクスポート
db2batch の使用 5
並列性
ロード・ユーティリティー 82
ヘッダー・レコード、PC/IXF 239
変換機能ストアード・プロシージャー 216
変更された構文と動作
ロード・ユーティリティー 81
変数
構文 217
保留状態 161

［マ行］

無効な PC/IXF データ・タイプ 265
無効な PC/IXF 列値 274
メッセージ 291
メッセージ・ファイル: エクスポート、インポート、およびロード 26, 73, 168
文字ストリング区切り文字 228

［ヤ行］

ユーザー定義特殊タイプ (UDT)
インポート・ユーティリティー 37
ユーティリティーのファイル形式 225
有効な PC/IXF データ・タイプ 265
要約表
インポートの制約事項 72

［ラ行］

ラージ・オブジェクト (LOB)
インポート・ユーティリティー 36
ラージ・オブジェクト (LOB) *(続き)*
エクスポート・ユーティリティー 5
リリース情報 304
列
インポートの指定 51
列、互換性のない 274
列値、無効な 274
列記述子レコード、PC/IXF 243
例
forcein 280
例、DEL ファイル 227
例外表
ロード・ユーティリティー 147
レコード長標識 237
レコード・タイプ
PC/IXF 236, 238
レコード・タイプ、PC/IXF
アプリケーション 249
階層 252
継続 255
索引 250
識別 257
終了 256
データ 248
表 240
副表 253
ヘッダー 239
列記述子 243
レプリケーション
データウェアハウス style=″xpp:(width=85)″センターでサポートされるタイプ 215
ロード
ファイル・タイプ修飾子 133
ロード、データの
データベース区画にデータをロードするためのオートローダー・ユーティリティー 169
ロード開始ログ・レコード 149
ロード削除開始補正ログ・レコード 149
ロード操作
データウェアハウスセンターでサポートされる 215

ロードのメッセージ・ファイル 26, 73, 168
ロード保留リスト・ログ・レコード 149
ロード・ユーティリティー
一時ファイル 148
インポート・ユーティリティーとの比較 221
概要 76
権限と特権、使用に必要な 83
コード・ページについての考慮事項 149
構築フェーズ 77
削除フェーズ 77
識別列 86
障害からの回復 91
制限 167
生成列 88
制約事項 167
ダンプ・ファイル 148
データベースの回復 79
パフォーマンスの最適化 162
プロセスの概要 76
並列性 82
変更された構文と動作 81
例外表 147
ロード・フェーズ 76
ログ・レコード 149
DB2 データ・リンク・マネージャー 194
ロード・ユーティリティーのファイル形式 225
ログ・レコード
ロード・ユーティリティー 149
ロッキング
インポート・ユーティリティー 37

[ワ行]

ワークシート・ファイル形式 (WSF) 289

A

anyorder 133
ASC のデータ・タイプの説明 233
ASC ファイル
形式 231
例 232
ASC ファイルの例 232
ASC、インポート・ファイル・タイプとしての 41

B

binarynumerics 141

C

chardel 18, 66, 143
codepage 136
coldel 18, 66, 143
compound 60

D

dateformat 62, 137
datesiso 18, 66, 143
DB2 データ・リンク・マネージャー
インスタンス間でのエクスポート 193
インポート・ユーティリティー 194
エクスポート・ユーティリティー 190
ロード・ユーティリティー 194
ロード・ユーティリティーのトラブルシューティング 195
DB2 ライブラリー
印刷版のブックの注文 305
インフォメーション・センター 309
ウィザード 310
オンライン情報の検索 313
オンライン情報の表示 308
オンライン・ヘルプ 305
構成内容 293
最新情報 304
セットアップ、文書サーバーの 312
ブック 293
ブックの言語識別子 303
PDF 資料の印刷 305
db2LoadQuery - ロード照会 128
DB2LOADREC 92
db2move 200
decplusblank 18, 66, 143
decpt 18, 67, 143
DEL データ・タイプの説明 228
DEL ファイル
形式 226
例 227
delprioritychar 67, 144
dldel 18, 67, 144
dumpfile 138
DWC (データウェアハウスセンター) 214

F

fastparse 133
forcein 68, 145
一般的なセマンティクス 279
オプション 279
コード・ページのセマンティクス 279
データ・タイプのセマンティクス 285
要約、PC/IXF ファイルのインポート 286
例 280

G

generatedignore 60, 133
generatedmissing 60, 133
generatedoverride 134

H

HTML
サンプル・プログラム 303

I

IBM リレーショナル・データ・レプリケーション・ツール
概要 212
構成要素 214

identityignore 60, 134
identitymissing 61, 134
identityoverride 134
implieddecimal 62, 138
indexfreespace 135
indexixf 68
indexschema 68

K

keepblanks 68, 144

L

LOAD
  一時ファイル 106
LOAD QUERY 109
LOAD QUERY (db2LoadQuery) 128
lobsinfile 18, 61, 135

N

Netscape ブラウザー
  インストール 308
nochecklengths 65, 68, 141, 145
nodefaults 61
nodoubledel 18, 68, 144
noeofchar 63, 140
noheader 135
norowwarnings 135
no_type_id 61
nullindchar 65, 141

P

packeddecimal 142
pagefreespace 135
PC バージョンの IXF (PC/IXF) ファイル形式 236
PC/IXF
  コード・ページ変換ファイル 274
  データ・タイプ 259
  データ・タイプの説明 266
  比較、System/370 IXF との 288
  無効なデータ・タイプ 265, 273
PC/IXF *(続き)*
  無効な列値 274
  有効なデータ・タイプ 265
  レコード・タイプ 236, 238
PC/IXF ファイル
  形式 236
PC/IXF ファイル形式
  移動、プラットフォーム間のデータの 197
PC/IXF ファイルのインポート
  一般規則 273
  データ・タイプ固有の規則 275
  FORCEIN を使用する場合 286
PC/IXF レコード・タイプ
  アプリケーション 249
  階層 252
  継続 255
  索引 250
  識別 257
  終了 256
  データ 248
  表 240
  副表 253
  ヘッダー 239
  列記述子 243
PDF 305
PDF 資料の印刷 305

R

reclen 65, 142

S

SELECT ステートメント
  EXPORT コマンド 10
SmartGuides
  ウィザード 310
SQL メッセージ 291
SQLCODE 291
SQLSTATE 291
SQLUIMPT-IN 構造体 57
SQLUIMPT-OUT 構造体 58
SQLULOAD-IN 構造体 121
SQLULOAD-OUT 構造体 126
SQL-UEXPT-OUT 構造体 17
striptblanks 65, 142
striptnulls 66, 142
System/370 IXF 288
  比較、PC/IXF との 288

T

timeformat 63, 139
timestampformat 64, 140
totalfreespace 135

U

usedefaults 62, 136

W

WSF ファイル
  形式 289
WSF ファイル形式
  プラットフォーム間のデータの移動 199

Z

zoneddecimal 143

IBM と連絡をとる

技術上の問題がある場合は、時間をとって「問題判別の手引き」に定義されている処置を検討し、それらの提案を実行した後で、DB2 顧客サービスに連絡をとってください。この資料には、DB2 顧客サービスがお客さまを支援するために必要とする情報が説明されています。

製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

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

DB2 World Wide Web ページには、ニュース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供されています。ただし、提供されている情報は英語です。

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

「DB2 Product and Service Technical Library」では、よくされる質問 (FAQ)、修正内容、資料、および最新の DB2 技術情報などの情報へのアクセスが提供されています。

注: この情報のご提供は英語のみとなりますのでご注意ください。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

「International Publications」注文用 Web サイトでは、マニュアルの注文方法についての情報を提供しています。ただし、提供されている情報は英語です。

http://www.ibm.com/education/certify/

IBM の「Professional Certification Program」Web サイトでは、 DB2 を含むさまざまな IBM 製品の認証テストの情報を提供しています。ただし、提供されている情報は英語です。

ftp.software.ibm.com

匿名でログオンしてください。ディレクトリー `/ps/products/db2` には、 DB2 および多数の他製品に関連したデモ、修正プログラム、情報、およびツールがあります。ただし、提供されている情報は英語です。

comp.databases.ibm-db2, bit.listserv.db2-l

これらのインターネット・ニュースグループは、ユーザーが DB2 製品に関する自分の経験について話し合うために利用できます。ただし、提供されている情報は英語です。

Compuserve: GO IBMDB2

このコマンドを入力すると、IBM DB2 Family forum にアクセスできます。すべての DB2 製品が、このフォーラムでサポートされています。ただし、提供されている情報は英語です。

米国以外の国で IBM に連絡する方法については、 *IBM Software Support Handbook* の Appendix A を参照してください。この資料にアクセスするには、 Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

注: 国によっては、 IBM が承認している販売業者が、 IBM サポート・センターの代わりにそれら販売業者のサポート・センターに連絡する場合があります。

Printed in Japan

SC88-8522-00

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