

IBM® DB2 Universal Database™



データ移動ユーティリティー ガイドおよびリファレンス

バージョン 8.2

IBM® DB2 Universal Database™



データ移動ユーティリティー ガイドおよびリファレンス

バージョン 8.2

ご注意！

本書および本書で紹介する製品をご使用になる前に、『特記事項』に記載されている情報をお読みください。

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

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

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

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

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典： SC09-4830-01
IBM® DB2 Universal Database™
Data Movement Utilities Guide and Reference
Version 8.2

発 行： 日本アイ・ピー・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 2004.8

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

© Copyright International Business Machines Corporation 1999, 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

本書について	vii
本書の対象読者	vii
本書の編成	vii

第 1 章 エクスポート

エクスポートの概要	1
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
ID 列を使用したエクスポート	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	6
EXPORT	9
db2Export - エクスポート	14
エクスポート用のファイル・タイプ修飾子	22
エクスポート・セッション - CLP の例	27

第 2 章 インポート

インポートの概要	29
インポートの使用に必要な特権、権限、および許可	31
インポートの使用	31
クライアント/サーバー環境でのインポートの使用	33
バッファ挿入を介したインポートの使用	33
インポートでの ID 列の使用	34
インポートでの生成列の使用	36
インポートを使用した、エクスポートされる表の再作成	37
ラージ・オブジェクト (LOB) のインポート	38
ユーザー定義特殊タイプ (UDT) のインポート	39
インポート時の表のロック	40
IMPORT	41
db2Import - インポート	55
インポート用のファイル・タイプ修飾子	68
文字セットと NLS についての考慮事項	78
インポート・セッション - CLP の例	79

第 3 章 ロード

ロードの概要	84
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	88
バージョン 8 で導入された以前のロード動作の変更	89
並列処理とロード	91
ロードの使用に必要な特権、権限、および許可	92
ロードの使用	93
読み取りアクセス・ロード操作	95
索引の作成	97
ロードでの ID 列の使用	99
ロードでの生成列の使用	101
保全性違反のチェック	103
従属即時マテリアライズ照会表のリフレッシュ	106

従属即時ステージング表の伝搬	107
マルチディメンション・クラスタリングの考慮事項	108
中断したロード操作の再開	110
読み取りアクセス許可ロード操作の再始動または終了	110
ロード・コピー・ロケーション・ファイルを使ったデータのリカバリ	111
LOAD	113
LOAD QUERY	137
db2Load - ロード	139
db2LoadQuery - ロードの照会	164
ロード用のファイル・タイプ修飾子	168
ロード例外表	182
ロード・ダンプ・ファイル	182
ロード一時ファイル	183
ロード・ユーティリティのログ・レコード	184
表ロック、表状態、および表スペース状態	184
文字セットと各国語のサポート	187
ロード操作後のペンディング状態	188
ロードのパフォーマンスの最適化	189
ロード - CLP の例	195

第 4 章 パーティション・データベース環境でのデータのロード

パーティション・データベース・ロードの概説	201
パーティション・データベース環境でのロードの使用	203
ロード照会コマンドを使用したパーティション・データベース・ロードのモニター	209
パーティション・データベース環境でのロード操作の再始動または終了	211
パーティション・データベース・ロード構成オプション	213
パーティション・データベース・ロード・セッションの例	219
移行およびバックレベル互換性	222
パーティション・データベース環境でのデータのロード - ヒント	224

第 5 章 DB2 Data Links Manager のデータの移動

エクスポートを使用した DB2 Data Links Manager データの移動 - 概念	227
エクスポートを使用した DB2 Data Links Manager データの移動	230
インポートを使用した DB2 Data Links Manager データの移動	231
ロードを使用した DB2 Data Links Manager データの移動	232

第 6 章 システム間のデータの移動 . . . 235

プラットフォーム間のデータの移動 - ファイル・フォーマットの考慮事項	235
PC/IXF ファイル・フォーマット	235
区切り付き ASCII (DEL) ファイル・フォーマット	236
WSF ファイル・フォーマット	236
DB2 Connect によるデータの移動	237
db2move - データベース移動ツール	239
db2relocatedb - データベースの再配置	245
データ移動での区切り文字の制限	249
型付き表間のデータ移動	250
型付き表間のデータ移動 - 詳細	252
走査順序	252
データ移動中の選択	253
型付き表間のデータ移動の例	253
レプリケーションを使ったデータの移動	255
IBM レプリケーション・ツールのコンポーネント	257
IBM レプリケーション・ツールのコンポーネント	257
データウェアハウス・センターによるデータの移動	257
カーソル・ファイル・タイプを使用したデータの移動	259

付録 A. 構文図の読み方 261

付録 B. インポート・ユーティリティとロード・ユーティリティの相違点 . . 265

付録 C. エクスポート/インポート/ロード・セッション - API サンプル・プログラム 267

付録 D. ファイル・フォーマット . . . 277

エクスポート/インポート/ロード・ユーティリティのファイル・フォーマット	277
区切り付き ASCII (DEL) ファイル・フォーマット	278
例およびデータ・タイプの説明	280
DEL ファイル例	280
DEL のデータ・タイプの説明	281
区切りなし ASCII (ASC) ファイル・フォーマット	283
例およびデータ・タイプの説明	284
ASC ファイル例	284
ASC のデータ・タイプの説明	285
PC バージョンの IXF ファイル・フォーマット	287
PC バージョンの IXF ファイル・フォーマット - 詳細	289
PC/IXF レコード・タイプ	289
PC/IXF データ・タイプ	307
PC/IXF のデータ・タイプの説明	313
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	318
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	320
FORCEIN オプション	323

PC/IXF およびバージョン 0 の System/370 IXF の相違	330
ワークシート・ファイル・フォーマット (WSF)	331

付録 E. エクスポート/インポート/ロード・ユーティリティの Unicode の考慮事項 333

コード・ページ 1394、1392、および 5488 の制約事項	334
非互換性	335

付録 F. エクスポート、インポート、およびロード・ユーティリティによって使用されるバインド・ファイル 337

付録 G. 警告、エラー、および完了メッセージ 339

付録 H. DB2 Universal Database 技術情報 341

DB2 資料とヘルプ	341
DB2 資料の更新	341
DB2 PDF 資料および印刷された資料	342
DB2 の基本情報	342
管理情報	343
アプリケーション開発情報	343
ビジネス・インテリジェンス情報	344
DB2 Connect 情報	345
入門情報	345
チュートリアル情報	346
オプション・コンポーネント情報	346
リリース・ノート	347
PDF ファイルからの DB2 資料の印刷方法	348
DB2 の印刷資料の注文方法	349
DB2 ツールからコンテキスト・ヘルプを呼び出す	349
コマンド行プロセッサからメッセージ・ヘルプを呼び出す	350
コマンド行プロセッサからコマンド・ヘルプを呼び出す	351
コマンド行プロセッサから SQL 状態ヘルプを呼び出す	352
DB2 インフォメーション・センターの呼び出し	352
コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール	353
DB2 トラブルシューティング情報	354
アクセス支援	355
キーボードによる入力およびナビゲーション	356
アクセスしやすい表示	356
支援テクノロジーとの互換性	356
アクセスしやすい資料	357
ドット 10 進シンタックス・ダイアグラム	357
DB2 チュートリアル	359
DB2 インフォメーション・センター	360

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)	DB2 Universal Database 製品の共通基準認証
362	370
DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)	付録 I. 特記事項 371
364	商標 373
DB2 インフォメーション・センターのインストール・シナリオ	索引 375
367	IBM と連絡をとる. 381
DB2 インフォメーション・センターにおける特定の言語でのトピックの表示	製品情報 381
370	

本書について

本書では、以下に示す IBM DB2 Universal Database (UDB) データ移動ユーティリティに関する情報と、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・ユーティリティは、表またはビュー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 データベース同士の間で、データを移動します。さらに、DB2 Connect 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、ロードを使用した方がインポート・ユーティリティ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・ユーティリティでは、エクスポート・ユーティリティ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。
- ロード・ユーティリティがパーティション・データベース環境で使用される場合には、大量のデータをパーティション化し、異なるデータベース・パーティションにロードすることができます。
- DataPropagator (DPROP) は DB2 Universal Database の 1 つのコンポーネントであり、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 データベース同士の間で、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 Universal Database、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。

本書の編成

以下のトピックを取り上げています。

第 1 章

DB2 エクスポート・ユーティリティについて説明します。このユーティリティは DB2 表からファイルヘデータを移動します。

第 2 章

DB2 インポート・ユーティリティについて説明します。このユーティリティは、ファイルから DB2 表またはビューヘデータを移動します。

第 3 章

DB2 ロード・ユーティリティについて説明します。このユーティリティは大量のデータを DB2 表へ移動します。

第 4 章

パーティション・データベース環境でのデータのロードについて説明します。

第 5 章

DB2 のエクスポート、インポート、およびロード・ユーティリティを使用することにより、DB2 Data Links Manager のデータを移動する方法について説明します。

第 6 章

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

構文図で使用されている表記規則について説明します。

付録 B

DB2 ロード・ユーティリティとインポート・ユーティリティの主要な相違点を要約しています。

付録 C

データをファイルへエクスポートする方法、データを表へインポートする方法、データを表にロードする方法、およびロード操作の状況をチェックする方法を示した API サンプル・プログラムを記載しています。

付録 D

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でサポートされる外部ファイル・フォーマットについて説明します。

付録 E

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を使用する場合の Unicode の考慮事項について解説します。

付録 F

バインド・ファイルとそのデフォルト分離レベルをリストし、さらにどのユーティリティがどんな目的で使用するかについて説明します。

付録 G

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・ユーティリティについて説明します。このユーティリティは、DB2 データベースからデータベースの外部に保管される 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・ユーティリティまたは DB2 ロード・ユーティリティを使用します。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。

以下のトピックを取り上げています。

- 『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『ID 列を使用したエクスポート』
- 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』
- 6 ページの『データの並列エクスポート』
- 9 ページの『EXPORT』
- 14 ページの『db2Export - エクスポート』
- 27 ページの『エクスポート・セッション - CLP の例』

DB2 Data Links Manager のデータのエクスポートについては、230 ページの『エクスポートを使用した DB2 Data Links Manager データの移動』を参照してください。型付き表からのデータのエクスポートについては、250 ページの『型付き表間のデータ移動』を参照してください。DRDA サーバー・データベースから DB2 Connect 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、237 ページの『DB2 Connect 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・フォーマットのいずれか）にデータをエクスポートします。その後、オペレーティング・システム・ファイルを使用して、表データを DB2® UDB for iSeries™ などの異なるサーバーに移動することができます。

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 DEL フォーマット・ファイル。フォーマットは IXF、WSF、または DEL です。
- 型付き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 フォーマットを使用する場合は、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りますことに注意してください。型付き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。

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、既存の表またはビューの中での列名が、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。
- メッセージ・ファイル名。データのエクスポート、インポート、ロード、バインド、リストアなどの 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。

複数データベース・パーティション環境でエクスポート・ユーティリティを使用する場合は、**db2batch** を使用することによって、各データベース・パーティションごとに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
```

関連概念:

- 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』
- 4 ページの『ID 列を使用したエクスポート』
- 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』
- 6 ページの『データの並列エクスポート』
- 250 ページの『型付き表間のデータ移動』
- 「管理ガイド: パフォーマンス」の『db2batch テストの例』

関連タスク:

- 3 ページの『エクスポートの使用』

関連資料:

- 14 ページの『db2Export - エクスポート』
- 「コマンド・リファレンス」の『db2batch - ベンチマーク・ツール・コマンド』
- 27 ページの『エクスポート・セッション - CLP の例』
- 277 ページの『エクスポート/インポート/ロード・ユーティリティのファイル・フォーマット』
- 9 ページの『EXPORT』

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・マネージャーの保守およびユーティリティ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。

関連資料:

- 14 ページの『db2Export - エクスポート』
- 9 ページの『EXPORT』

エクスポートの使用

前提条件:

エクスポート・ユーティリティを起動するには、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。

制約事項:

エクスポート・ユーティリティには、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では、構造タイプ列を持つ表はサポートされません。

手順:

エクスポート・ユーティリティは、コマンド行プロセッサ (CLP)、コントロール・センターの「エクスポート (Export)」ノートブック、またはアプリケーション・プログラミング・インターフェース (API)、**db2Export** から起動できます。

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
```

「エクスポート (Export)」ノートブックをオープンするには、以下のようになります。

1. コントロール・センターから、「表 (Tables)」フォルダーまたは「ビュー (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、既存の表またはビュー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ビュー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詳細は、オンライン・ヘルプ機能によって提供されます。

関連資料:

- 14 ページの『db2Export - エクスポート』

関連サンプル:

- 『tbmove.out -- HOW TO MOVE TABLE DATA (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.out -- HOW TO MOVE TABLE DATA (C++)』
- 『tbmove.sqC -- How to move table data (C++)』

ID 列を使用したエクスポート

エクスポート・ユーティリティーを使用して、ID 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、ID 列プロパティを 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 オプションを使用して、ID 列プロパティを示した表を再作成することができます。タイプ GENERATED ALWAYS の ID 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、すべての行がリジェクトされます (SQL3550W)。

関連概念:

- 「管理ガイド: プランニング」の『ID 列』

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・フォーマット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。

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・ユーティリティを複数回起動することができます。

DEL および **ASC** ファイル・フォーマットには、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・フォーマット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、ロードまたはインポート・ユーティリティを使用して、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) を使用すると、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。

関連概念:

- 37 ページの『インポートを使用した、エクスポートされる表の再作成』

関連資料:

- 「コマンド・リファレンス」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』

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。

注: IXF ファイル・フォーマットには、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。

LOB ロケーション指定子 (LLS) は、LOB 情報のエクスポートの際に単一ファイルに複数の LOB を格納するために使用されます。lobsinfile 修飾子を使用してデータをエクスポートする際に、エクスポート・ユーティリティは LOB ファイル全体を選択し、これを、LOB ファイルの 1 つに配置します。LOB ファイルごとに複数の LOB を、そして LOB パスごとに複数の LOB ファイルを入れることができます。データ・ファイルには、LLS レコードが入っています。

LLS は、ファイル内で LOB データが見つかる位置を示す文字列です。LLS のフォーマットは、filename.ext.nnn.mmm/ です。ここで、filename.ext は LOB の入ったファイルの名前、nnn はファイル内の LOB のオフセット (バイト単位)、mmm は LOB の長さ (バイト単位) です。たとえば、LLS が db2exp.001.123.456/ である場合、これは、LOB がファイル db2exp.001 にあり、ファイル内の 123 バイトのオフセットで始まり、長さが 456 バイトであることを示します。LLS で示されたサイズが 0 の場合、LOB の長さは 0 であると見なされます。長さが -1 の場合には、LOB は NULL と見なされ、オフセットおよびファイル名は無視されます。

関連資料:

- 14 ページの『db2Export - エクスポート』
- 9 ページの『EXPORT』
- 「SQL リファレンス 第 1 巻」の『ラージ・オブジェクト (LOB)』

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 (表のパーティションごとに 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** コマンドを使用します。このユーティリティーは、パーティション・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に照会を実行する
- それぞれのパーティション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パーティションごとに作成する

注: ターゲット・ファイル・パスを複数のデータベース・パーティションで共有する場合、そのパスを共有するすべてのパーティションからの出力は 1 つのファイルに収められます。

各パーティションで照会が並列実行されて、そのパーティションのデータが検索されます。-p s オプションを指定した **db2batch** コマンドを実行すると、オリジナルの選択照会は並列実行されます。-p t オプションまたは -p d オプションを指定して **db2batch** コマンドを実行すると、指定の照会を使ってステージング表にエクスポート・データがロードされ、さらに、各パーティションごとにステージング表に対して **SELECT *** 照会が並列実行されてそのデータがエクスポートされます。特定のパーティション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、そのパーティション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、表の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。元の照会内の最初の列名を使ってこのパラメーターが設定されます。

db2batch コマンドによって SQL 照会が実行され、その出力がターゲット・ファイルに送られるということを理解することが重要です。このコマンドは、エクスポート・ユーティリティーを使用しません。エクスポート・ユーティリティー・オプションは、**db2batch** コマンドを使った並列エクスポートでは用いられません。**db2batch** コマンドを使って LOB 列をエクスポートすることはできません。

コマンド・オプションの総合的な説明を見るには、-h オプションを指定した **db2batch** コマンドをコマンド・ウィンドウから実行します。

db2batch コマンドは、並列 SQL 照会を実行して、指定されたファイルに出力を送ります。このコマンドは、エクスポート・ユーティリティではなく、選択ステートメントを実行することに注意してください。データ長に関係なく、この方法を使って LOB 列をエクスポートすることはできません。

スタッフ表の内容をエクスポートするには、次のようにします。

```
db2batch -p s -d sample -f staff.batch -r /home/userid/staff.asc -q on
```

この例の詳細は次のとおりです。

- 照会は、1 つの表に対して並列で実行されます (-p s オプション)。
- サンプル・データベースへの接続が確立されます (-d sample オプション)。
- 制御ファイル `staff.batch` には、セミコロンで終了する SQL 選択ステートメント `select * from staff;` が入ります。
- 各データベース・パーティションのアクセス可能な `/home/userid/staff.asc` ファイルに出力が保管されます。また、デフォルトの出力フォーマットは定位置 ASCII です (db2batch は、エクスポート・ユーティリティを使用しないことに注意してください)。
- 照会の出力のみがこのファイルに送られます (-q オプション)。

区切り文字で区切られている ASCII ファイルにエクスポートするには、以下のようになります。

```
db2batch -p s -d sample -f emp_resume.batch -r /home/userid/emp_resume.del,  
/home/mmilek/userid/emp_resume.out -q del
```

この例の詳細は次のとおりです。

- `emp_resume` 表から LOB 以外の列だけが選択されます (制御ファイル `emp_resume.batch` には、`select empno,resume_format from emp_resume;` と示されます)。
- `emp_resume.del` ファイルには、区切り文字で区切られている ASCII フォーマットの照会出力が入ります (-q del オプション)。コンマがデフォルトの列区切り文字であり、`|` がデフォルトの `char` 区切り文字になります。
- `emp_resume.out` ファイルには、照会の統計が入ります。

照会する表や、その表が置かれているパーティション・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。

- パーティション表からの並列エクスポートか、または連結された複数の表に対する結合または副照会からの並列エクスポート (**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見なすことができます。

- それぞれの表が、1 つのパーティションで定義されている 1 つのデータベース・パーティション・グループ内にある。
- それぞれの表が同じデータベース内にあって、同一数および同一タイプの列を持つパーティション・キーを持っている。パーティション・キーの対応列は、パーティションに対応し、表は、パーティション・キー全体またはパーティション・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 機能を使用して、各パーティションで照会を実行して、そのパーティションのエクスポート・データ・ファイルを生成することができます。(表が 1 つのパーティション内にだけ存在する場合、データは 1 つのパーティション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、ステージング表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表を作成します。)

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表を使用します。このステージング表は、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。ステージング表の作成の完了後、エクスポート・ユーティリティーは、ステージング表に対して次のコマンドを実行して、各パーティションでエクスポート・データ・ファイルを生成します。

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

ステージング表を使用して、1 つのパーティション表を並列にエクスポートすることもできます。たいいていの場合、単一パーティションから複数パーティションのステージング表にデータを転送してから、すべてのパーティションでステージング表を並列にエクスポートする方が、1 つのパーティション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、各パーティションで並列に照会を実行して、それぞれのパーティション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表がロードされます。また SELECT * 照会は、各パーティションのステージング表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。特定のパーティション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、そのパーティション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、表の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、パーティション・キーを組み込む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** 文節があれば各パーティション上のファイル

はソート順になります。1つのソート・ファイルを得たい場合は、各パーティション上のソート・ファイルを組み合わせて1つのソート・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つのソート・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ソートされません。

関連概念:

- 1 ページの『エクスポートの概要』

関連資料:

- 「コマンド・リファレンス」の『db2batch - ベンチマーク・ツール・コマンド』

EXPORT

データベースから、いくつかある外部ファイル・フォーマットのどれかにデータをエクスポートします。ユーザーは、SQL `SELECT` ステートメントを提供するか、型付き表の階層情報を提供して、エクスポートするデータを指定します。

権限:

以下のどれかが必要です。

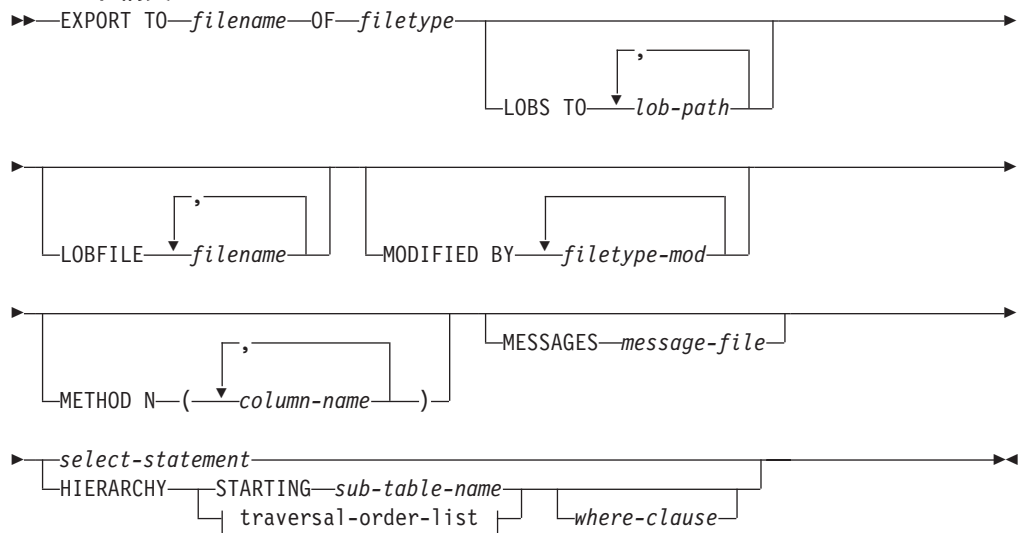
- `sysadm`
- `dbadm`

または、関係する各表またはビューに対する `CONTROL` または `SELECT` 特権

必要な接続:

データベース。暗黙接続が可能な場合には、デフォルト・データベースへの接続が確立されます。

コマンド構文:



traversal-order-list:

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。すべての副表は、PRE-ORDER 方式でリストされてい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、または WSF ファイルの OUTER 順序、または 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* から始まる副階層をエクスポートします。

LOBFILE filename

LOB ファイルに 1 つ以上の基本ファイル名を指定します。最初の名前の名前スペースがいっぱいになると、2 番目の名前が使用され、以下 3 番目、4 番目と続きます。

エクスポート操作中に LOB ファイルを作成するときに、まずこのリストから現行パス (*lob-path* で指定されたパス) に現行の基本名を追加し、それに 3 桁のシーケンス番号を追加したファイル名が構成されます。たとえば、現行 LOB パスがディレクトリー `/u/foo/lob/path/` で、現行 LOB ファイル名が `bar` の場合、LOB ファイルは、`/u/foo/lob/path/bar.001`、`/u/foo/lob/path/bar.002` (以下 003、004 と続く) などのように作成されます。

LOBS TO lob-path

LOB ファイルが保管される、ディレクトリーへの 1 つ以上のパスを指定します。LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が組み込まれます。

MESSAGES message-file

エクスポート操作中に生じ得る警告およびエラー・メッセージの宛先を指定します。宛先ファイルがすでに存在している場合、エクスポート・ユーティリティは情報を追加します。 *message-file* を省略すると、メッセージは標準出力に書き込まれます。

METHOD N column-name

出力ファイルで使用される 1 つ以上の列名を指定します。このパラメーターが指定されない場合、表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すが、階層データをエクスポートするときは無効です。

MODIFIED BY filetype-mod

ファイル・タイプ修飾子オプションを指定します。『エクスポート用のファイル・タイプ修飾子』を参照してください。

OF filetype

次のような出力ファイルのデータ・フォーマットを指定します。

- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャーで使⽤します。
- WSF (ワークシート・フォーマット)。以下のプログラムで使⽤します。
 - Lotus 1-2-3
 - Lotus Symphony

注: BIGINT または DECIMAL データをエクスポートする場合、タイプ DOUBLE の範囲内の値のみが正確にエクスポートされます。この範囲内にない値もエクスポートされますが、オペレーティング・システムによっては、これらの値のインポートまたはエクスポートの結果、データに間違いが生じる場合があります。

- SELECT ステートメントで列が指定してある場合を除き、ほとんどの表属性である IXF (統合交換フォーマット、PC バージョン) と、既存の索引が IXF ファイルに保管されます。このフォーマットを使用すると、表は再作成されます。一方、他のファイル・フォーマットを使用する場合、データをそこにインポートするには表が存在していなければなりません。

select-statement

エクスポートされるデータを戻す SELECT ステートメントを指定します。SELECT ステートメントによってエラーが発生する場合、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 である場合、エクスポート操作は続行します。これ以外のエラー・コードの場合、操作は停止します。

TO filename

データのエクスポート先のファイルの名前を指定します。このファイルへの完全パスが指定されていない場合、エクスポート・ユーティリティーは現在のディレクトリーおよびデフォルトのドライブ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ファイルの内容を上書きします。情報の追加は行いません。

例:

次に示すのは、SAMPLE データベースにある STAFF 表から、ファイル myfile.ixf に情報をエクスポートする方法の一例です。これは、IXF フォーマットで出力されます。コマンドを発⾏する前に、SAMPLE データベースと接続していなければなりません。データベース接続が DB2 Connect を介して確⽴されていない場合、索引定義 (もしあれば) は出力ファイルに保管されます。

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

次に示すのは、SAMPLE データベースの STAFF 表から部門 20 の従業員に関する情報を、エクスポートする方法の一例です。これは IXF フォーマットで出力され、awards.ixf ファイルに入ります。コマンドを発⾏する前に、まず SAMPLE データベースと接続しなければなりません。また、表の中の実際の列名は、'department' ではなく 'dept' であることにも注意してください。

```
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。

EXPORT

```
| 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
```

使用上の注意:

エクスポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。

SELECT ステートメントでは表の別名を使用できます。

メッセージ・ファイルに置かれるメッセージには、メッセージ検索サービスから戻される情報が入っています。各メッセージは改行してから始まります。

DEL フォーマット・ファイルへエクスポートするために 254 よりも長い文字データの列が選択されると、エクスポート・ユーティリティーは警告メッセージを生成します。

PC/IXF インポートはデータベース間でデータを移動する場合に使用します。行区切り文字の入った文字データを区切り文字付き ASCII (DEL) ファイルにエクスポートし、テキスト転送プログラムにより処理を行うと、行区切り文字の入ったフィールドは長さが伸縮します。

ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。

DB2 Connect は、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーから表をエクスポートするために使用できます。PC/IXF エクスポートだけがサポートされています。

エクスポート・ユーティリティーは、AIX システムから呼び出される場合、複数部分からなる PC/IXF ファイルを作成しません。

エクスポート・ユーティリティーは、提供される SELECT ステートメントが、SELECT * FROM tablename という形式である場合、IXF ファイルの表の NOT NULL WITH DEFAULT 属性を保管します。

型付き表をエクスポートする場合、副選択ステートメントは、ターゲット表名と WHERE 文節を指定することによってのみ表現することができます。階層をエクスポートするとき、全選択と選択ステートメント は指定できません。

IXF 以外のファイル・フォーマットの場合は、走査順序リストを指定するようお勧めします。このリストは、階層を走査する方法やエクスポートする副表を DB2 に指示します。このリストを指定しない場合、階層内のすべての表がエクスポートされ、デフォルトの順序は OUTER 順序になります。または、OUTER 関数によって指定される順序である、デフォルトの順序を使用することができます。

注: インポート操作の間も同じ走査順序を使用します。ロード・ユーティリティーは、階層や副階層のロードをサポートしていません。

DB2 Data Links Manager の考慮事項:

エクスポート時に、整合のとれた表のコピーと、 DATALINK 列によって参照される対応するファイルが確実にコピーされるようにするには、以下のようにします。

1. QUIESCE TABLESPACES FOR TABLE tablename SHARE コマンドを発行する。

これで、EXPORT の実行時に更新トランザクションが進行しなくなります。

2. EXPORT コマンドを発行します。

3. 各データ・リンク・サーバーで **dlfm_export** ユーティリティーを実行します。
dlfm_export ユーティリティーへの入力制御ファイル名です。これは、エクスポート・ユーティリティーによって生成されます。これにより、制御ファイル内にリストされるファイルの tar (または同等の) アーカイブが作成されます。

4. QUIESCE TABLESPACES FOR TABLE tablename RESET コマンドを発行する。

これにより、表は更新に使用できるようになります。

EXPORT は、SQL アプリケーションとして実行されます。SELECT ステートメントを満たす行と列がデータベースから抽出されます。DATALINK 列の場合、SELECT ステートメントはスカラー関数を指定できません。

EXPORT が正常に実行されると、以下のファイルが生成されます。

- EXPORT コマンドで指定したエクスポート・データ・ファイル。このファイルの DATALINK 列値は、インポートおよびロード・ユーティリティーによって使用されるフォーマットと同じです。DATALINK 列の値が SQL NULL 値である場合、他のデータ・タイプと同じ処理が行われます。
- 各データ・リンク・サーバー用に生成される制御ファイル *server_name*。
Windows オペレーティング・システムでは、単一制御ファイル、ctrlfile.lst がすべてのデータ・リンク・サーバーによって使用されます。これらの制御ファイルは、ディレクトリー <data-file path>%dlfm %YYYYMMDD%HHMMSS に入れます (Windows NT オペレーティング・システムの場合、ctrlfile.lst はディレクトリー <data-file path>%dlfm%YYYYMMDD%HHMMSS に入れます)。YYYYMMDD は日付 (年月日)を、HHMMSS は時刻 (時、分、秒)を表します。

ファイルをデータ・リンク・サーバーからエクスポートするため、**dlfm_export** ユーティリティーが提供されています。このユーティリティーが生成するアーカイブ・ファイルを使用して、ターゲットのデータ・リンク・サーバーにファイルをリストアすることができます。

関連概念:

- 1 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』

関連タスク:

- 3 ページの『エクスポートの使用』

関連資料:

- 14 ページの『db2Export - エクスポート』
- 27 ページの『エクスポート・セッション - CLP の例』
- 22 ページの『エクスポート用のファイル・タイプ修飾子』
- 249 ページの『データ移動での区切り文字の制限』

db2Export - エクスポート

データベースから、いくつかある外部ファイル・フォーマットのいずれかにデータをエクスポートします。ユーザーは、SQL SELECT ステートメントによって、または型付き表の階層情報によってエクスポートするデータを指定します。

許可:

以下のいずれかです。

- *sysadm*
- *dbadm*

または、関係するそれぞれの表またはビューに対する CONTROL または SELECT 特権

必要な接続:

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2Export */
/* ... */

SQL_API_RC SQL_API_FN
db2Export (
    db2UInt32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);
```

```

typedef SQL_STRUCTURE db2ExportStruct
{
    char                                *piDataFileName;
    struct sqlu_media_list              *piLobPathList;
    struct sqlu_media_list              *piLobFileList;
    struct sqldcol                      *piDataDescriptor;
    struct sqllob                       *piActionString;
    char                                *piFileType;
    struct sqlchar                      *piFileTypeMod;
    char                                *piMsgFileName;
    db2int16                            iCallerAction;
    struct db2ExportOut                 *poExportInfoOut;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportOut
{
    db2UInt64                           oRowsExported;
} db2ExportOut;
/* ... */

```

汎用 API 構文:

```

/* File: db2ApiDf.h */
/* API: db2gExport */
/* ... */

```

```

SQL_API_RC SQL_API_FN
db2gExport (
    db2UInt32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);

```

```

typedef SQL_STRUCTURE db2gExportStruct
{
    char                                *piDataFileName;
    struct sqlu_media_list              *piLobPathList;
    struct sqlu_media_list              *piLobFileList;
    struct sqldcol                      *piDataDescriptor;
    struct sqllob                       *piActionString;
    char                                *piFileType;
    struct sqlchar                      *piFileTypeMod;
    char                                *piMsgFileName;
    db2int16                            iCallerAction;
    struct db2ExportOut                 *poExportInfoOut;
    db2UInt16                           iDataFileNameLen;
    db2UInt16                           iFileTypeLen;
    db2UInt16                           iMsgFileNameLen;
} db2gExportStruct;
/* ... */

```

API パラメーター:

versionNumber

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

pParmStruct

入力。 *db2ExportStruct* 構造を指すポインター。

pSqlca

出力。 *sqlca* 構造へのポインター。

iDataFileNameLen

入力。データ・ファイル名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

iFileTypeLen

入力。ファイル・タイプの長さを示す 2 バイトの符号なし整数 (バイト単位) です。

iMsgFileNameLen

入力。メッセージ・ファイル名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

piDataFileName

入力。データがエクスポートされるパスおよび外部ファイル名の入ったストリングを指定します。

piLobPathList

入力。 *media_type* `SQLU_LOCAL_MEDIA` を使用する *sqlu_media_list*、および LOB ファイルが保管されるクライアント上のパスをリストする *sqlu_media_entry* 構造。

このリスト内の最初のパス上でファイル・スペースが使い尽くされると、API は 2 番目のパスを使用し、以下同様に続きます。

piLobFileList

入力。 *media_type* `SQLU_CLIENT_LOCATION` を使用する *sqlu_media_list*、およびベース・ファイル名を組み込んだ *sqlu_location_entry* 構造です。

このリスト内の最初の名前を使用している名前スペースが使い尽くされると、API は 2 番目の名前を使用し、以下同様に続きます。

エクスポート操作中に LOB ファイルが作成されるときには、現行パス (*pLobFilePath* から) にこのリストからの現行ベース名を追加し、その後に 3 桁のシーケンス番号を追加した形のファイル名が構成されます。たとえば、現行の LOB パスが `/u/foo/lob/path` ディレクトリーで、現行の LOB ファイル名が `bar` の場合、作成される LOB ファイルの名前は、`/u/foo/lob/path/bar.001`、`/u/foo/lob/path/bar.002` (以下同様) となります。

piDataDescriptor

入力。出力ファイルの列名を指定する *sqldcol* 構造を指すポインター。
dcolmeth フィールドの値によって、このパラメーターに提供される残りの情報をエクスポート・ユーティリティーがどのように解釈するかが判別されます。このパラメーターに有効な値 (*sqlutil* で定義) 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。表の既存の列の名前が、出力ファイルで使用されます。この場合、列数および列指定配列は、どちらも無視されます。列名は、*pActionString* で指定された `SELECT` ステートメントの出力から派生します。

piActionString

入力。有効な動的 SQL SELECT ステートメントを備えた *sqllob* 構造を指すポインター。この構造には、4 バイトの長さフィールドと、SELECT ステートメントを構成する文字が順に組み込まれます。SELECT ステートメントは、データベースからデータを取り出し、外部ファイルに書き込むことを指定します。

外部ファイルの列 (*piDataDescriptor* からの) と、SELECT ステートメントからのデータベース列とは、それぞれのリストまたは構造における位置に従って対応付けられます。データベースから選択されたデータの最初の列は、外部ファイルの最初の列に置かれ、その列名は外部列配列の最初のエレメントから取られます。

piFileType

入力。外部ファイル内のデータのフォーマットを示すストリングを指定します。サポートされている外部ファイルのフォーマット (*sqlutil* で定義) は、以下のとおりです。

SQL_DEL

区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、IBM パーソナル・デシジョン・シリーズ・プログラム、およびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための形式です。

SQL_WSF

ワークシート・フォーマット。Lotus Symphony および Lotus 1-2-3 プログラムとの交換のためのフォーマットです。

SQL_IXF

IXF (統合交換フォーマット、PC バージョン)。表からデータをエクスポートする場合の推奨方式です。このファイル・フォーマットにエクスポートされたデータは、後で同じ表または別のデータベース・マネージャー表にインポートまたはロードできます。

piFileTypeMod

入力。2 バイトの長さフィールドと、1 つまたは複数の処理オプションを指定する文字の配列の入った *sqldcol* 構造を指すポインターです。このポインターが NULL であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。『エクスポート用のファイル・タイプ修飾子』を参照してください。

piMsgFileName

入力。このユーティリティーが戻すエラー、警告、および情報メッセージの宛先の入ったストリングを指定します。オペレーティング・システム・ファイルまたは標準装置のパスおよび名前を指定できます。ファイルがすでに存在する場合は上書きされます。存在していない場合は、新たに作成されます。

iCallerAction

入力。呼び出し側が要求するアクションを示します。有効な値 (sqlutil で定義) は、以下のとおりです。

SQLU_INITIAL

最初の呼び出し。この値は、API への最初の呼び出しの際には必ず使用してください。

最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたエクスポート操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定しなければなりません。

SQLU_CONTINUE

処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (たとえば、テープの終わり条件への応答) を要求した後で、API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションの完了後、ユーティリティーが最初の要求の処理を続行するように指定するものです。

SQLU_TERMINATE

処理の終了。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (たとえば、テープの終わり条件への応答) を要求した後で、API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが実行されなかった場合、ユーティリティーが最初の要求の処理を中断するように指定するものです。

poExportInfoOut

db2ExportOut 構造を指すポインター。

oRowsExported

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。

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 ステートメントの入った REXX ホスト変数。このステートメントにより、データベースから取り出すデータが指定されます。

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・ファイルのデータのフォーマット。サポートされているファイル・フォーマットは、以下のとおりです。

DEL 区切り文字付き ASCII

WSF ワークシート・フォーマット

IXF 統合交換フォーマットの PC バージョン

filetmod

追加の処理オプションの入ったホスト変数。

dcoldata

エクスポート・ファイルで使用する列名の入ったコンパウンド REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 列数 (残りの変数内のエレメントの数)

XXX.1 最初の列名。

XXX.2 2 番目の列名。

XXX.3 以降、3 番目、4 番目 ... と続きます。

このパラメーターが NULL の場合、または *dcoldata* に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からの列名を使用します。

msgfile

エラーおよび警告メッセージが送られるファイル、パス、または装置の名前。

number

エクスポートされた行の数が入れられるホスト変数。

使用上の注意:

エクスポート操作を開始する前に、すべての表操作を完了し、すべてのロックを解除するようにしてください。このことは、**WITH HOLD** でオープンしているカーソルをすべてクローズした後に **COMMIT** を発行するか、または **ROLLBACK** を発行することにより、行うことができます。

SELECT ステートメントでは表の別名を使用できます。

メッセージ・ファイルに置かれたメッセージには、メッセージ検索サービスから戻される情報が入っています。各メッセージは新しい行から始まります。

DEL フォーマット・ファイルへエクスポートするために 254 を超える長さのストリングを選択すると常に、エクスポート・ユーティリティーによって警告メッセージが出されます。

外部列名配列 *piDataDescriptor* の列数 (*dcolnum*) が **SELECT** ステートメントによって生成される列数と同じでない場合には、警告メッセージが出されます。この場合、外部ファイルに書き込まれる列数はそれらのうち小さい方の数になります。出力ファイルを生成するために、余分のデータベース列または外部列名が使用されることはありません。

db2Export - エクスポート

db2uexpm.bnd モジュールまたは配布された他の .bnd ファイルを手動でバインドする場合には、バインド・プログラムの **フォーマット・オプション** を使用しないでください。

PC/IXF インポートは、データベース間でデータを移動する場合に使用します。行区切り文字の入った文字データが区切り付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字の入ったフィールドは長さが変わることがあります。

DB2 Connect を使用して、DB2 for z/OS および OS/390、DB2 for VM and VSE、および DB2 for iSeries などの DRDA サーバーからデータをエクスポートすることができます。PC/IXF エクスポートだけがサポートされています。

エクスポート・ユーティリティーは、AIX システムから呼び出されたときには複数部から成る PC/IXF ファイルを作成しません。

表の索引定義が PC/IXF ファイルに組み込まれるのは、単一のデータベース表の内容が、`SELECT * FROM tablename` で始まる *pActionString* を指定して PC/IXF ファイルにエクスポートされ、*piDataDescriptor* パラメーターにデフォルト名が指定されているときです。ビューの索引は保管されません。*pActionString* の `SELECT` 文節に結合が入っている場合も同様です。*pActionString* の `WHERE` 文節、`GROUP BY` 文節、または `HAVING` 文節は索引の保管を妨げません。どの場合も、型付き表からのエクスポート時に、階層全体をエクスポートする必要があります。

提供された `SELECT` ステートメントが `SELECT * FROM tablename` の形式である場合には、エクスポート・ユーティリティーにより表の `NOT NULL WITH DEFAULT` 属性が IXF ファイルに保管されます。

型付き表をエクスポートする場合、副選択ステートメントを表すことができるのは、ターゲット表の名前と `WHERE` 文節だけです。階層をエクスポートする場合、全選択と *select-statement* は指定できません。

IXF 以外のファイル・フォーマットの場合は、階層の走査の方法とエクスポートする副表とが DB2 に知らされるよう、走査順序リストを指定することをお勧めします。このリストが指定されていないと、階層のすべての表がエクスポートされ、`OUTER` 順序がデフォルトの順序になります。`OUTER` 関数によって指定されるデフォルトの順序を使うこともできます。

注: インポート操作時には、同じ走査順序を使用してください。ロード・ユーティリティーでは、階層または副階層のロードはサポートされていません。

DB2 Data Links Manager の考慮事項:

エクスポート時に、整合のとれた表のコピーと、`DATALINK` 列によって参照される対応するファイルが確実にコピーされるようにするには、以下のようにします。

1. `QUIESCE TABLESPACES FOR TABLE tablename SHARE` コマンドを発行する。

これにより、`EXPORT` の実行時に更新トランザクションが進行しなくなります。

2. `EXPORT` コマンドを発行する。

3. それぞれのデータ・リンク・サーバーで、**dlfm_export** ユーティリティを実行する。**dlfm_export** ユーティリティへの入力制御ファイル名です。これは、エクスポート・ユーティリティによって生成されます。このようにすると、制御ファイル内でリストされるファイルの tar (または同等の) アーカイブが生成されます。**dlfm_export** は、アーカイブされるファイルの ACL 情報をキャプチャーしません。
4. QUIESCE TABLESPACES FOR TABLE tablename RESET コマンドを発行する。

この操作により表が更新可能になります。

EXPORT は、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 列の場合、SELECT ステートメントではスカラー関数を指定しないようにしてください。

EXPORT が正常に実行されると、以下のファイルが生成されます。

- EXPORT コマンドで指定されたエクスポート・データ・ファイル。このファイルの DATALINK 列の値は、IMPORT および LOAD ユーティリティが使用するのと同じ DEL フォーマット・ファイルです。DATALINK 列の値が SQL NULL 値の場合、処理は他のデータ型の場合と同様になります。
- 制御ファイル *server_name* は、各データ・リンク・サーバーに対して生成されます。Windows NT オペレーティング・システムでは、単一の制御ファイル *ctrlfile.lst* がすべてのデータ・リンク・サーバーによって使用されます。これらの制御ファイルは、<data-file path>/dlfm/YYYYMMDD/HHMMSS ディレクトリーに置かれます (Windows NT オペレーティング・システムでは、*ctrlfile.lst* は <data-file path>¥dlfm¥YYYYMMDD ¥HHMMSS ディレクトリーに置かれます)。YYYYMMDD は日付 (年月日) を表し、HHMMSS は時刻 (時分秒) を表します。

データ・リンク・サーバーからファイルをエクスポートするため、**dlfm_export** ユーティリティが提供されています。このユーティリティが生成するアーカイブ・ファイルを使用して、ターゲットのデータ・リンク・サーバーにファイルをリストアすることができます。

関連概念:

- 227 ページの『エクスポートを使用した DB2 Data Links Manager データの移動 - 概念』

関連資料:

- 「管理 API リファレンス」の『SQLCA』
- 「管理 API リファレンス」の『SQLCHAR』
- 「管理 API リファレンス」の『SQLDCOL』
- 「管理 API リファレンス」の『SQLU-MEDIA-LIST』
- 22 ページの『エクスポート用のファイル・タイプ修飾子』
- 249 ページの『データ移動での区切り文字の制限』

関連サンプル:

- 『exp samp.sqb -- Export and import tables with table data to a DRDA database (IBM COBOL)』
- 『imp exp.sqb -- Export and import tables with table data (IBM COBOL)』
- 『tload.sqb -- How to export and load table data (IBM COBOL)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.sqC -- How to move table data (C++)』

エクスポート用のファイル・タイプ修飾子

表 1. エクスポートで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット

修飾子	説明
lobsinfile	<p><i>lob-path</i> には、LOB データの入ったファイルへのパスを指定します。</p> <p>各パスには、データ・ファイル内で LOB ロケーション指定子 (LLS) によって示される 1 つ以上の LOB の入った、少なくとも 1 つのファイルが組み込まれます。LLS は、LOB ファイル・パスに保管されるファイル内の LOB のロケーションのストリング表現です。LLS の形式は、<i>filename.ext.nnn.mmm/</i> です。<i>filename.ext</i> は LOB を収めたファイルの名前、<i>nnn</i> はファイル内の LOB のオフセット (バイト単位)、<i>mmm</i> は LOB の長さ (バイト単位) を表します。たとえば、ストリング <i>db2exp.001.123.456/</i> がデータ・ファイルに保存される場合、LOB はファイル <i>db2exp.001</i> のオフセット 123 に位置し、456 バイト長です。</p> <p>EXPORT の使用時に "lobsinfile" 修飾子を指定した場合、LOB データは LOBS TO 文節に指定されたロケーションに置かれます。指定しない場合、LOB データは現行作業ディレクトリーに送られます。LOBS TO 文節は、LOB ファイルが保管されるディレクトリーに、1 つまたは複数のパスを指定します。LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が入ります。</p> <p>NULL LOB を指定するには、サイズに -1 と入力します。サイズを 0 と指定すると、長さが 0 の LOB として扱われます。長さが -1 の NULL LOB の場合、オフセットとファイル名は無視されます。たとえば、NULL LOB の LLS は <i>db2exp.001.7.-1/</i> です。</p>

表 2. エクスポートで有効なファイル・タイプ修飾子: DEL (区切り文字で区切られている ASCII) ファイル・フォーマット

修飾子	説明
chardelx	<p><i>x</i> 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。指定した文字は、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。² 文字ストリング区切り文字として明示的に二重引用符を指定したい場合、次のように指定します。</p> <p style="text-align: center;">modified by charde1""</p> <p>単一引用符 (') も、以下のように文字ストリングの区切り文字として指定できます。</p> <p style="text-align: center;">modified by charde1''</p>

表 2. エクスポートで有効なファイル・タイプ修飾子: *DEL* (区切り文字で区切られている *ASCII*) ファイル・フォーマット (続き)

修飾子	説明
codepage= <i>x</i>	<p><i>x</i> は <i>ASCII</i> 文字ストリングです。この値は、出力データ・セット内のデータのコード・ページと解釈されます。エクスポート操作時に、文字データをアプリケーションのコード・ページからこのコード・ページに変換します。</p> <p>DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。</p> <p>注: codepage 修飾子を lobsinfile 修飾子と一緒に使うことはできません。</p>
coldel <i>x</i>	<p><i>x</i> は単一文字カラム区切り文字です。デフォルト値はコンマ (,) です。指定した文字は、列の終わりを表すために、コンマの代わりに使用されます。 ²</p> <p>以下の例では、coldel; が指定されており、エクスポート・ユーティリティーは検出するすべてのセミコロン (;) を列の区切り文字として解釈します。</p> <pre>db2 "export to temp of del modified by coldel; select * from staff where dept = 20"</pre>
datesiso	日付形式。すべての日付データ値を ISO フォーマット ("YYYY-MM-DD ") でエクスポートします。 ³
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) ではなくブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進数の前に正符号 (+) が付けられます。
decpt <i>x</i>	<i>x</i> は、小数点文字としてピリオドの代わりに使用される単一の置換文字です。デフォルト値はピリオド (.) です。指定した文字は、小数点文字としてピリオドの代わりに使用されます。 ²
dlldel <i>x</i>	<p><i>x</i> 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。指定した文字は、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 値には 2 つ以上の副値を指定できるので、この区切り文字が必要です。 ²</p> <p>注: 行、列、または文字ストリング区切り文字と同じ文字を <i>x</i> に指定することはできません。</p>
nochardel	<p>列データは区切り文字で囲まれません。データを DB2 を使用してインポートまたはロードするつもりの場合は、このオプションを指定しないでください。これは、区切り文字を持たないペンダー・データ・ファイルをサポートするために用意されています。不適切に使用すると、データが損失または破壊される場合があります。</p> <p>このオプションを chardel<i>x</i> または nodoubledel と一緒に指定することはできません。これらは相互に排他的なオプションです。</p>
nodoubledel	二重になっている区切り文字 ² の認識を抑止します。

db2Export - エクスポート

表 2. エクスポートで有効なファイル・タイプ修飾子: *DEL* (区切り文字で区切られている *ASCII*) ファイル・フォーマット (続き)

[illegible]

表 2. エクスポートで有効なファイル・タイプ修飾子: DEL (区切り文字で区切られている ASCII) ファイル・フォーマット (続き)

修飾子	説明
timestampformat="x"	<p>x はソース・ファイルのタイム・スタンプのフォーマットです。 ⁴ 有効なタイム・スタンプ・エレメントは以下のとおりです。</p> <p>YYYY - 年 (0000 から 9999 の範囲の 4 桁の数字)</p> <p>M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数)</p> <p>MM - 月 (01 から 12 の 2 桁の数。 M と MMM とは相互に排他的)</p> <p>MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的)</p> <p>D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数)</p> <p>DD - 日 (1 から 31 の 2 桁の数字。 D とは相互に排他的)</p> <p>DDD - 元日から数えた日数 (001 から 366 の 3 桁の数字。 他の日または月のエレメントとは相互に排他的)</p> <p>H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数。)</p> <p>HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の範囲の 2 桁の数。 H と相互に排他的)</p> <p>M - 分 (0 から 59 の 1 桁または 2 桁の数字。)</p> <p>MM - 分 (0 から 59 の範囲の 2 桁の数字。 M (分) とは相互に排他的)</p> <p>S - 秒 (0 から 59 の 1 桁または 2 桁の数字。)</p> <p>SS - 秒 (0 から 59 の範囲の 2 桁の数字。 S と相互に排他的)</p> <p>SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の 5 桁の数字。 他の時刻エレメントとは相互に排他的)</p> <p>UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数字。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数字。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>TT - 午前/午後の指定子 (AM または PM)</p> <p>タイム・スタンプ・フォーマットの例を以下に示します。</p> <p>"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>MMM エレメントは、以下の値を生成します。「Jan」、「Feb」、「Mar」、「Apr」、「May」、「Jun」、「Jul」、「Aug」、「Sep」、「Oct」、「Nov」、および「Dec」。「Jan」は 1 月と等しく、「Dec」は 12 月と等しいです。</p> <p>以下の例は、「schedule」という表から、ユーザー定義のタイム・スタンプ・フォーマットを示すデータをエクスポートする方法を示しています。</p> <pre>db2 export to delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" select * from schedule</pre>

表 3. エクスポートで有効なファイル・タイプ修飾子: WSF ファイル・フォーマット

修飾子	説明
1	Lotus 1-2-3 リリース 1、または Lotus 1-2-3 リリース 1a との互換がある WSF ファイルを作成します。 ⁵ この値がデフォルトです。
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。 ⁵
3	Lotus 1-2-3 バージョン 2、または Lotus Symphony リリース 1.1 との互換がある WSF ファイルを作成します。 ⁵
4	DBCS 文字から成る WSF ファイルを作成します。

注:

- サポートされていないファイル・タイプを **MODIFIED BY** オプションで使用しようとしても、エクスポート・ユーティリティーは警告を出しません。サポートされていないファイル・タイプを使おうとすると、エクスポート操作は失敗し、エラー・コードが戻されます。
- データ移動のための区切り文字の制約事項 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。
- 通常、エクスポート・ユーティリティーでの記述フォーマットは次のとおりです。

- 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.aaaaaa* の形式

エクスポート操作のために **SELECT** ステートメントで指定される日時列に組み込まれたデータも、これらの形式になります。

- タイム・スタンプ・フォーマットの場合、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、区別があいまいにならないように注意する必要があります。月のフィールドは、他の日付フィールドと隣接していなければなりません。分フィールドは、他の時刻フィールドに隣接していなければなりません。以下に、いくつかのあいまいなタイム・スタンプ・フォーマットを示します。

"M" (月または分のどちらにもとれる)
 "M:M" (月と分の区別がつかない)
 "M:YYYY:M" (両方とも月と解釈される)
 "S:M:YYYY" (時刻値と日付値の両方に隣接している)

あいまいな場合、ユーティリティーはエラー・メッセージを報告し、操作は失敗します。

以下に示すのは、明確なタイム・スタンプ・フォーマットです。

"M:YYYY" (M (月))
 "S:M" (M (分))
 "M:YYYY:S:M" (M (月)...M (分))
 "M:H:YYYY:M:D" (M (分)...M (月))

- filetype-mod* パラメーター・ストリングの中で、Lotus 1-2-3 の場合は **L**、Symphony の場合は **S** を指定すれば、これらのファイルを特定の製品に送ることができます。指定できるのは、1 つの値または製品指定子だけです。

関連資料:

- 14 ページの『db2Export - エクスポート』
- 9 ページの『EXPORT』
- 249 ページの『データ移動での区切り文字の制限』

エクスポート・セッション - 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 フォーマット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 Connect を介していない場合、索引定義が存在するならば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 フォーマット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
```

関連資料:

- 9 ページの『EXPORT』

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ビューにデータを書き込みます。ターゲットの表またはビュー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。

以下のトピックを取り上げています。

- 『インポートの概要』
- 31 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』
- 31 ページの『インポートの使用』
- 33 ページの『クライアント/サーバー環境でのインポートの使用』
- 33 ページの『バッファー挿入を介したインポートの使用』
- 34 ページの『インポートでの ID 列の使用』
- 36 ページの『インポートでの生成列の使用』
- 37 ページの『インポートを使用した、エクスポートされる表の再作成』
- 38 ページの『ラージ・オブジェクト (LOB) のインポート』
- 39 ページの『ユーザー定義特殊タイプ (UDT) のインポート』
- 40 ページの『インポート時の表のロックング』
- 41 ページの『IMPORT』
- 55 ページの『db2Import - インポート』
- 78 ページの『文字セットと NLS についての考慮事項』
- 79 ページの『インポート・セッション - CLP の例』

DB2 Data Links Manager のデータのインポートについては、231 ページの『インポートを使用した DB2 Data Links Manager データの移動』を参照してください。型付き表からのデータのインポートについては、250 ページの『型付き表間のデータ移動』を参照してください。DB2 Connect 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、237 ページの『DB2 Connect 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ビュー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ビュー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・ビューの名前または別名。

- 入力ファイル内のデータのフォーマット。フォーマットは IXF、WSF、DEL、または ASC です。
- 入力データを表またはビューに挿入するのか、それとも表またはビュー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) **squimpr** 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。
- 型付き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 **ORDER BY** を指定する必要があるかもしれません。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。

型付き表を処理する場合、副表リストを用意する必要もあります。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。

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、または相対列ロケーション。
- 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ビュー内の列の名前。
- メッセージ・ファイル名。データのエクスポート、インポート、ロード、バインド、リストアなどの **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。

注: 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。

関連概念:

- 250 ページの『型付き表間のデータ移動』

関連資料:

- 55 ページの『db2Import - インポート』
- 79 ページの『インポート・セッション - CLP の例』

- 277 ページの『エクスポート/インポート/ロード・ユーティリティのファイル・フォーマット』
- 41 ページの『IMPORT』

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・マネージャーの保守およびユーティリティ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。既存の表またはビュー内のデータを置換するには、その表またはビューに対する SYSADM 権限、DBADM 権限、または CONTROL 特権か、あるいはその表またはビューに対する INSERT、SELECT、UPDATE、および DELETE 特権が必要です。既存の表またはビューにデータを追加するには、その表またはビューに対する SELECT 特権と INSERT 特権が必要です。

関連資料:

- 55 ページの『db2Import - インポート』
- 41 ページの『IMPORT』

インポートの使用

前提条件:

インポート・ユーティリティを起動するには、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い必要があります。このユーティリティは 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。

制約事項:

インポート・ユーティリティには、以下の制約事項が適用されます。

- このユーティリティでは、ニックネーム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主キーを備えた親表である場合、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マテリアライズ照会表の基本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、サマリー表、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・ユーティリティを使ってビュー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、主キー定義は保存されます。)
- インポート・ユーティリティは独自の 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は最初の 30KB と最後の 30KB を保持します。

手順:

インポート・ユーティリティは、コマンド行プロセッサ (CLP)、コントロール・センターの「インポート (Import)」ノートブック、またはアプリケーション・プログラミング・インターフェース (API)、**sqluimpr** から起動できます。

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
```

「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詳細は、オンライン・ヘルプ機能によって提供されます。

関連資料:

- 55 ページの『db2Import - インポート』

関連サンプル:

- 『tbmove.out -- HOW TO MOVE TABLE DATA (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.out -- HOW TO MOVE TABLE DATA (C++)』
- 『tbmove.sqC -- How to move table data (C++)』

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用さ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。
- 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場合、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。
- それらのファイル用の十分なディスク・スペースがある。
- 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。

関連概念:

- 29 ページの『インポートの概要』

バッファー挿入を介したインポートの使用

パーティション・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。

バッファー挿入を使用すると、インポートではデフォルトの WARNINGCOUNT 値は 1 に設定されます。そのため、行が 1 つでもリジェクトされるとユーティリティーは失敗します。レコードがリジェクトされた場合、ユーティリティーは現在のトランザクションをロールバックします。コミット済みのレコード数を調べれば、

どのレコードが正常にデータベースに挿入されたかを判別することができます。コミット済みのレコード数がゼロ以外になりうるのは、COMMITCOUNT オプションを指定していた場合のみです。

別の WARNINGCOUNT 値をインポート・コマンドに明示的に指定していた場合にいずれかの行がリジェクトされると、ユーティリティーからの行サマリー出力は誤っていることがあります。その原因は、バッファ挿入で使用される非同期エラー・レポートと、行グループの挿入時にエラーが検出されたことに起因したそのグループのすべての行のバックアウトが組み合わさったことにあります。どの入力レコードがリジェクトされたかに関するユーティリティーからのレポートは信頼できないので、どのレコードがコミット済みで、どのレコードをデータベースに再挿入する必要があるかを判別するのは困難になります。

バッファ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 オプションを使用して、データベースに対してインポート・パッケージ db2uimpb.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 db2uimpb.bnd insert buf
```

バッファ挿入は、INSERT_UPDATE パラメーターを指定したインポート操作と連携して使用することはできません。この制約事項の実施のために、新しいバインド・ファイル (db2uimpb2.bnd) が導入されています。この新しいファイルは、INSERT BUF オプションにバインドしてはなりません。もしバインドすると、INSERT_UPDATE パラメーターを指定したインポート操作が失敗する原因になります。ただし INSERT、REPLACE、または REPLACE_CREATE パラメーターを指定したインポート操作は、この新しいファイルのバインドによって影響を受けません。

関連概念:

- 29 ページの『インポートの概要』

インポートでの ID 列の使用

インポート・ユーティリティーを使用して、ID 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。ID 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。

- ID 列が GENERATED ALWAYS の場合、入力ファイル内の対応する行の中に ID 列用の値がないか、または NULL 値が明示的に指定されているときに、表の行用の ID 値が生成されます。ID 列に非 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。
- ID 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用します。データが欠落しているかまたは明示的に NULL 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 ID 値に関して、ID 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、ID 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。

ID 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 ID 列の値が入っていない (NULL すらない) 場合に、ID 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たします。

```
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、ID 列を持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` コマンドを使用して、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・ユーティリティは、入力データ・ファイルに ID 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 ID 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 ID 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` 修飾子を使用すると、構文が単純化されます。

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

ID 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` オプションを使用して、ID 列プロパティを備えた表を再作成することができます。タイプ `GENERATED ALWAYS` の ID 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、すべての行がリジェクトされます (SQL3550W)。

関連概念:

- 「管理ガイド: プランニング」の『ID 列』

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用して、(ID 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 NULL 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 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。
- NULL 可能列でない生成列用にサーバーが NULL 値を生成すると、このフィールドが属するデータ行はリジェクトされます (SQL0407N)。これが起きるのは、たとえば、NULL 可能列でない生成列が 2 つの表の列の合計として定義されていて、それらの表の列に入力ファイル内で NULL 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 (NULL 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て。

```
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、生成列を持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` コマンドを使用して、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るとしても、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 `NULL` 値 10、11、12 (`g1` の場合)、および 15、16、17 (`g2` の場合) により、行はリジェクトされます (`SQL3550W`)。リジェクト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` 修飾子を使用すると、構文が単純化されます。

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

関連概念:

- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『生成列』

インポートを使用した、エクスポートされる表の再作成

インポート・ユーティリティーを使用すると、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。

- 主キーの名前および定義
- ユニーク制約の名前と定義。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。
 - `ID` プロパティ
 - 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

- ID オプション
- NULL 可能列または NULL 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該当しません。
- 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ソートされるか
 - 索引がユニーク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
 - 索引で逆スキャンが可能かどうか
 - *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、マテリアライズ照会表、ビュー、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
 - マテリアライズ照会表定義 (該当する場合)
 - マテリアライズ照会表オプション (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)
 - チェック制約 (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)

関連概念:

- 4 ページの『エクスポートされた表の再作成』

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でも、または別のファイルからのものでもかまいません。データが別のファイルからのものである場合には、 **LOBSINFILE** ファイル・タイプ修飾子が指定されていなければなりません。

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 修飾子と、LOBS FROM 文節を使用します。

LOB ロケーション指定子 (LLS) を使用すると、LOB 情報のインポート、エクスポート、およびロードの実行時に、単一ファイルに複数の LOB を格納することができます。

LLS は、ファイル内で LOB データが見つかる位置を示すストリングです。LLS のフォーマットは、filename.ext.nnn.mmm/ です。ここで、filename.ext は LOB の入ったファイルの名前、nnn はファイル内の LOB のオフセット (バイト単位)、mmm は LOB の長さ (バイト単位) です。たとえば、LLS が db2exp.001.123.456/ である場合、これは、LOB がファイル db2exp.001 にあり、ファイル内の 123 バイトのオフセットで始まり、長さが 256 バイトであることを示します。LLS で示されたサイズが 0 の場合、LOB の長さは 0 であると見なされます。長さが -1 の場合には、LOB は NULL と見なされ、オフセットおよびファイル名は無視されます。

modified by lobsinfile オプションを指定してデータのインポートまたはロードを行う際には、対応するそれぞれの LOB 列ごとに LLS があるものと見なされます。LLS 以外のものが LOB 列にある場合には、データベースはこれを LOB ファイルと見なし、ファイル全体を LOB としてロードします。

関連資料:

- 318 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』
- 320 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データ・タイプ固有の規則』
- 41 ページの『IMPORT』
- 「SQL リファレンス 第 1 巻」の『ラージ・オブジェクト (LOB)』

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。

関連概念:

- ・「アプリケーション開発ガイド サーバー・アプリケーションのプログラミング」の『ユーザー定義特殊タイプ』

インポート時の表のロックング

インポート・ユーティリティでは、2 つの表ロックング・モードがサポートされます。オフライン・モード (ALLOW NO ACCESS) では、複数の同時アプリケーションから表データにアクセスできないようになります。これがデフォルト・モードです。オンライン・モード (ALLOW WRITE ACCESS) では、複数の同時アプリケーションからインポート・ターゲット表へ読み取りおよび書き込みアクセスすることができます。

デフォルトでは、インポート・ユーティリティは、分離レベル RS (読み取り固定) でデータベースにバインドされます。

オンライン・インポート (ALLOW WRITE ACCESS):

インポート・ユーティリティは、ターゲット表に非専用 (IX) ロックをかけます。表に対してこのロックをかけることには 2 つの意義があります。

- ・ 非互換の表ロックをかけている他のアプリケーションがあると、そのようなアプリケーションがすべて変更内容をコミットまたはロールバックするまでインポート・ユーティリティはデータの挿入を開始しません。
- ・ インポートが実行されているかぎり、非互換の表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、そのインポートが現在のトランザクションをコミットするかまたはロールバックするまで待機します。インポートでの表ロックは、次のトランザクションに持ち越されることはないことに注意してください。というわけで、オンライン・インポートでは、各コミットの完了ごとに表ロックの要求と、おそらくは待機を行う必要があります。
- ・ 非互換の行ロックをかけている他のアプリケーションがある場合、そのようなアプリケーションがすべて変更内容をコミットまたはロールバックするまでインポート・ユーティリティはデータの挿入を停止します。
- ・ インポートが実行されているかぎり、非互換の行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が現在のトランザクションをコミットするかまたはロールバックするまで待機します。

オンライン・インポートは、オンライン・プロパティを維持しながら、デッドロックの可能性を減らすために、現在のトランザクションを定期的にコミットして行ロックをすべて解放してから、排他的 (X) 表ロックへエスカレートします。したがって、オンライン・インポート中は、commitcount オプションを使わなくてもコミットが実行されることがあります。コミットの頻度は明示的に指定できますが、AUTOMATIC コミット・モードを使用してもかまいません。ゼロの commitcount 値を明示的に指定した場合、コミットは実行されません。競合する行ロックをかけている同時アプリケーションが表ロックへのエスカレートを試みると、デッドロックが発生することに注意してください。

「ALLOW WRITE ACCESS」を指定した場合、インポートはオンライン・モードで実行されます。オンライン・モードには、以下のものとの互換性はありません。

- ・ REPLACE、CREATE、および REPLACE_CREATE インポート・モード

- バッファ挿入
- ターゲット・ビューへのインポート
- 階層表へのインポート
- 表ロック・サイズを使ったターゲット表へのインポート

オフライン・インポート (ALLOW NO ACCESS):

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エスカレート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留していると、そのロックが排他ロックにエスカレート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。これがデフォルトのインポート動作です。

表に対するロックを保留することには 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留していると、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。「ALLOW WRITE ACCESS」を指定しなかった場合、インポートはオフライン・モードで実行されます。

関連概念:

- 184 ページの『表ロック、表状態、および表スペース状態』

IMPORT

外部ファイルのデータを、サポートされているファイル・フォーマットで表、階層、またはビューに挿入します。LOAD はより高速な代替方法です。しかしロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。

権限:

- INSERT オプションを使用して IMPORT コマンドを実行する場合、以下のどれかが必要です。
 - *sysadm*
 - *dbadm*
 - 関係する表またはビューのそれぞれに対する CONTROL 特権
 - 関係するそれぞれの表またはビューに対する INSERT および SELECT 特権
- INSERT_UPDATE オプションを使用して、既存の表に IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 表またはビューに対する CONTROL 特権
 - 関係するそれぞれの表またはビューに対する INSERT、SELECT、UPDATE および DELETE 特権

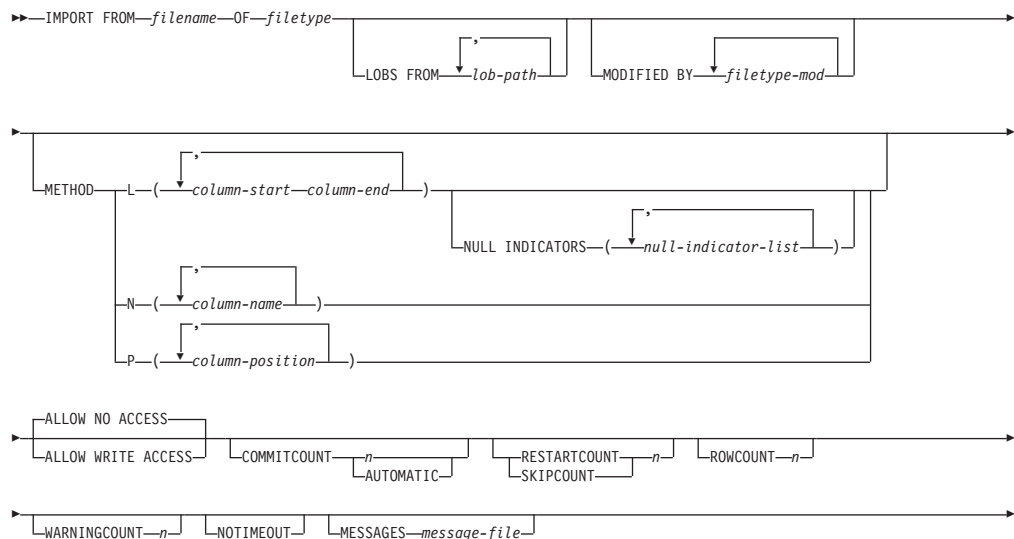
IMPORT

- REPLACE または REPLACE_CREATE オプションを使用して、既存の表に IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 表またはビューに対する CONTROL 特権
 - 表またはビューに対する INSERT、SELECT、および DELETE 特権
- CREATE または REPLACE_CREATE オプションを使用して新規の表に IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
 - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
 - スキーマに対する CREATIN 特権 (表のスキーマ名が既存のスキーマを指す場合)
- REPLACE オプションを使用して既存の階層に IMPORT するには、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 階層内のすべての副表に対する CONTROL 特権

必要な接続:

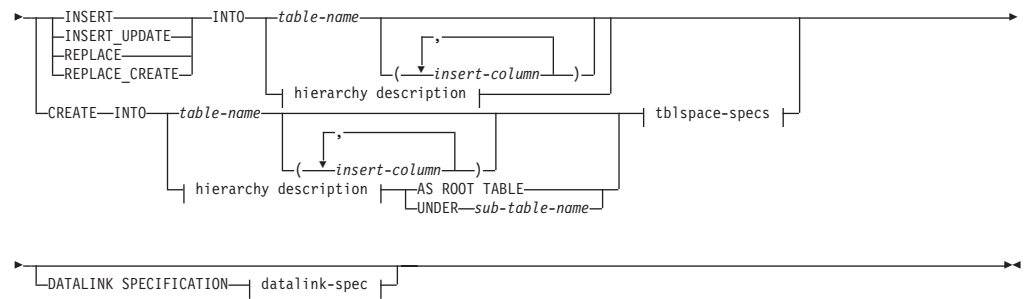
データベース。暗黙接続が可能な場合には、デフォルト・データベースへの接続が確立されます。

コマンド構文:

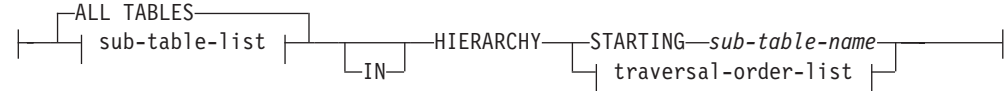


4

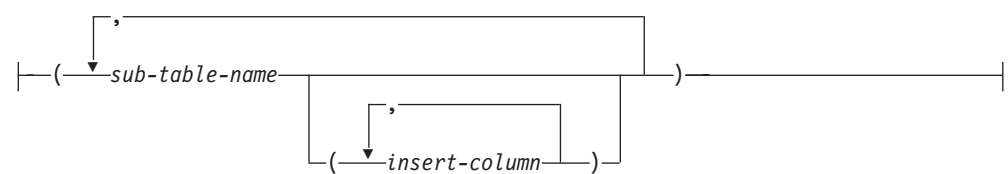
4



hierarchy description:



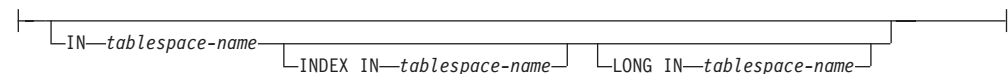
sub-table-list:



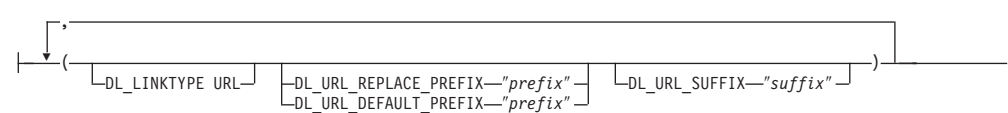
traversal-order-list:



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、走査順序で指定されるすべての表をインポートすることがデフォルトです。

ALLOW NO ACCESS

オフライン・モードでインポートを実行します。行の挿入の前には常に、ターゲット表に排他 (X) ロックがかけられます。これで、同時アプリケーションは表データにアクセスできなくなります。これがデフォルトのインポート動作です。

ALLOW WRITE ACCESS

オンライン・モードでインポートを実行します。最初の行の挿入時には、ターゲット表に意図的な排他 (IX) ロックがかけられます。これで、表データ

IMPORT

への同時の読み取りおよび書き出しアクセスが可能になります。オンライン・モードには、REPLACE、CREATE、または REPLACE_CREATE インポート・オプションとの互換性はありません。オンライン・モードとバッファ挿入との連携はサポートされません。インポート操作によって挿入後のデータが定期的にコミットされるので、表ロックへのロック・エスカレーションが妨げられて、アクティブなログ・スペースが使い果たされることはなくなります。このようなコミットは、COMMITCOUNT オプションを使わなくても実行されます。各コミットごとに、インポートでは IX 表ロックが外されるので、コミットの完了後に再びロックの設定が試みられます。

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*/AUTOMATIC

n 個のレコードがインポートされるたびに COMMIT を実行します。数 *n* を指定すると、インポートでは *n* 個のレコードのインポートの後にそのつど COMMIT が実行されます。コンパウンド挿入を使用した場合、ユーザー指定のコミット頻度 *n* は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。AUTOMATIC を指定すると、コミットの必要時機はインポート操作で内部的に判別されます。次の 2 つのうちのいずれかの理由で、このユーティリティーはコミットを行います。

- アクティブ・ログ・スペースが枯渇しないようにするため。
- ロックが行レベルから表レベルにエスカレーションしないようにするため。

ALLOW WRITE ACCESS オプションを指定した場合に COMMITCOUNT オプションを指定しないと、インポート・ユーティリティーは、COMMITCOUNT AUTOMATIC が指定されたものとしてコミットを実行します。

CREATE

データベースのコード・ページで表の定義と行の内容を作成します。DB2 の表、副表、または階層からエクスポートされたデータの場合、索引も作成されます。このオプションが階層に対するものである場合に、DB2 からデータがエクスポートされると、タイプ階層も作成されます。このオプションは、IXF ファイルの場合にのみ使用することができます。

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、ページ・サイズで計算した長さが 254 より少ない LONGVAR フィールドを収めている場合、CREATE は行が長過ぎるために失敗します。制約事項のリストの詳細は、『インポートを使用した、エクスポートされる表の再作成』を参照してください。この場合、その表は手動で作成します。そして、IMPORT に INSERT を指定して呼び出すか、または LOAD コマンドを使用してください。

DATALINK SPECIFICATION

各 DATALINK 列ごとに、それぞれ 1 つの列指定を括弧で囲んで指定できます。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。接頭部指定は、DL_URL_REPLACE_PREFIX または DL_URL_DEFAULT_PREFIX のどちらかになります。

DATALINK 列指定は、表で定義されている DATALINK 列の数と同じ数だけ指定できます。指定の順序は、挿入列 リストの中での DATALINK 列の順序、または表定義内での順序 (挿入列 リストが指定されていない場合) に従います。

DL_LINKTYPE

指定した場合は、列定義の LINKTYPE に一致していなければなりません。そうすることによって、列定義に LINKTYPE URL が指定されている場合に DL_LINKTYPE URL が受け入れ可能になります。

DL_URL_DEFAULT_PREFIX "prefix"

これを指定すると、同じ列内のすべての DATALINK 値のデフォルト接頭部になります。ここでいう接頭部とは、URL 指定の「スキーム・ホスト・ポート」部分のことです。

接頭部の例

```
"http://server"
"file://server"
"file:"
"http://server:80"
```

列のデータの中に接頭部がない場合、DL_URL_DEFAULT_PREFIX でデフォルトの接頭部が指定されている場合、列の値の接頭部としてそのデフォルト接頭部が付けられます (NULL でない場合)。

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

- 列入力値 "/x/y/z" は "http://toronto/x/y/z" として保管されます。
- 列入力値 "http://coyote/a/b/c" は "http://coyote/a/b/c" として保管されます。
- 列入力値 NULL は NULL として保管されます。

DL_URL_REPLACE_PREFIX "prefix"

この文節は、それ以前にエクスポート・ユーティリティによって生成されたデータをロードまたはインポートする際に、ユーザーが、データに入っているホスト名を別のホスト名に一括置換したい場合に便利です。指定する場合には、それがすべての 非 NULL 列値の接頭部になります。列値にすでに接頭部がある場合、それは置き換えられます。列値に接頭部がない場合、DL_URL_REPLACE_PREFIX で指定される接頭部がその列値の接頭部になります。

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

- 列入力値 "/x/y/z" は "http://toronto/x/y/z" として保管されます。
- 列入力値 "http://coyote/a/b/c" は "http://toronto/a/b/c" として保管されます。"coyote" は "toronto" に置き換えられます。
- 列入力値 NULL は NULL として保管されます。

DL_URL_SUFFIX "suffix"

これを指定すると、それはその列のすべての非 NULL 列値に付加されます。これは実際には、DATALINK 値の URL 部分の「パス」コンポーネントに付加されます。

FROM filename

インポートするデータの入ったファイルを指定します。パスを省略すると、現行の作業ディレクトリーが使用されます。

HIERARCHY

インポートする階層データを指定します。

IN tablespace-name

表を作成する表スペースを指定します。表スペースは存在している必要があり、REGULAR 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、すべての表パーツはこの表スペースに保管されます。この文節を指定しない場合、表は許可 ID によって作成された表スペース中に作成されます。何も検出されない場合、その表はデフォルト表スペースの USERSPACE1 に入れます。USERSPACE1 がドロップされていた場合、表作成は失敗します。

INDEX IN tablespace-name

表の索引の作成先の表スペースを指定します。このオプションは、IN 文節で指定した PRIMARY 表スペースが DMS 表スペースである場合のみ使用できます。指定した表スペースは存在している必要があり、かつ REGULAR または LARGE DMS 表スペースでなければなりません。

注: どの表スペースに索引を配置するかは、表を作成するときのみ指定できます。

insert-column

データの挿入先となる表またはビュー内の列名を指定します。

INSERT

既存の表データを変更することなく、インポートしたデータを表に追加します。

INSERT_UPDATE

インポートしたデータ行をターゲット表に追加するか、または主キーが一致するものがあれば、ターゲット表の既存行を更新します。

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、宣言一時表、またはサマリー表は指定できません。

下位のサーバーの場合を除き、INSERT、INSERT_UPDATE、および REPLACE オプションには、完全修飾または非修飾の表名を使用しなければならないようなときでも、別名を指定することができます。修飾子付き表名は、*schema.tablename* の形式です。*schema* には、表作成時のユーザー名が入ります。

LOBS FROM lob-path

LOB ファイルを保管するパスを指定します。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、LOB 列にロードされる列内に保管されます。lobsinfile 修飾子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。

LONG IN tablespace-name

ロング列の値 (LONG VARCHAR、LONG VARGRAPHIC、LOB データ・タイプ、またはソース・タイプとしてこれらが指定されている特殊タイプ)

を保管する表スペースを指定します。このオプションは、IN 文節で指定した PRIMARY 表スペースが DMS 表スペースである場合のみ使用できます。指定した表スペースは存在している必要があります、LARGE 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生じ得る警告およびエラー・メッセージの宛先を指定します。宛先ファイルがすでに存在している場合、インポート・ユーティリティーは情報を追加します。このファイルへの完全パスが指定されていない場合、このユーティリティーは現行のディレクトリーおよびデフォルトのドライブを宛先として使用します。message-file を省略すると、メッセージは標準出力に書き込まれます。

METHOD

L データのインポートを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。

注: このメソッドは、ASC ファイルの場合にのみ使用することができ、そのファイル・タイプに対してのみ有効なオプションです。

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 ファイルの場合にのみ使用することができます。

P インポートする入力データ・フィールドのフィールド番号を指定します。

注: この方式は、IXF または DEL ファイルの場合にのみ使用でき、DEL ファイル・タイプに対してのみ有効なオプションです。

MODIFIED BY filetype-mod

ファイル・タイプ修飾子オプションを指定します。『インポート用のファイル・タイプ修飾子』を参照してください。

NOTIMEOUT

インポート・ユーティリティーは、ロック待ちの間にタイムアウトしないことを指定します。このオプションの方が、locktimeout データベース構成パラメーターより優先されます。他のアプリケーションは影響を受けません。

NULL INDICATORS null-indicator-list

このオプションは、METHOD L パラメーターを指定した場合にのみ使用できます。つまり、入力ファイルが ASC ファイルの場合です。NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。NULL 標識リストには、METHOD L パラメーターで定義された各データ・フィールドに対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

4
4
4
4

NULL 標識列中の Y の値は、その列データが NULL であることを指定します。 NULL 標識列に Y 以外 の文字を指定した場合は、列データが NULL ではなく、METHOD L オプションで指定された列データがインポートされることを指定することになります。

nullindchar 修飾子を指定した MODIFIED BY オプションを使用すれば、NULL 標識文字を変更することができます。

OF filetype

入力ファイル内のデータのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)
- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャーで使います。
- WSF (ワークシート・フォーマット)。以下のプログラムで使います。
 - Lotus 1-2-3
 - Lotus Symphony
- IXF (統合交換フォーマット、PC バージョン)。同一のあるいは別の DB2 表からエクスポートされたことを意味します。 IXF ファイルには、表定義および既存の索引定義も入ります。ただし、SELECT ステートメントに列が指定されている場合は除きます。

REPLACE

データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除してから、インポートしたデータを挿入します。表定義および索引定義は変更されません。表がない場合は、このオプションを使用できません。 DATALINK 列を備えた表では無効です。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

REPLACE_CREATE

表がすでにある場合には、データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除し、表定義や索引定義は変えることなく、インポートしたデータを挿入します。

3 表がまだない場合には、データベースのコード・ページで、表と索引の定義
3 と行の内容を作成します。制約事項のリストの詳細は、『インポートを使用
3 した、エクスポートされる表の再作成』を参照してください。

このオプションは、IXF ファイルの場合にのみ使用することができます。 DATALINK 列を備えた表では無効です。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

RESTARTCOUNT *n*

n + 1 の位置のレコードからインポート操作を開始することを指定します。最初の *n* レコードはスキップされます。このオプションは機能的には SKIPCOUNT と同等です。 RESTARTCOUNT と SKIPCOUNT は相互に排他的です。

ROWCOUNT *n*

インポート (挿入または更新) するファイル内の物理レコードの数 *n* を指定します。ユーザーは、SKIPCOUNT または RESTARTCOUNT オプションで

指示されたレコードから始めて、ファイルの n 行だけをインポートすることができます。SKIPCOUNT または RESTARTCOUNT オプションの指定がないと、最初の n 行がインポートされます。SKIPCOUNT m または RESTARTCOUNT m を指定すると、行 $m+1$ から $m+n$ がインポートされます。コンパウンド挿入を使用した場合、ユーザー指定の rowcount n は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。

SKIPCOUNT n

$n + 1$ の位置のレコードからインポート操作を開始するよう指定します。最初の n 個のレコードはスキップされます。このオプションは機能的には RESTARTCOUNT と同等です。SKIPCOUNT と RESTARTCOUNT は相互に排他的です。

STARTING sub-table-name

階層専用キーワード。sub-table-name から始まるデフォルト順を要求します。PC/IXF ファイルの場合、デフォルト順は入力ファイルに保管されている順です。PC/IXF ファイル・フォーマットの場合、デフォルト順は有効な唯一の順序です。

sub-table-list

型付き表で INSERT または INSERT_UPDATE オプションを指定した場合、データのインポート先副表を指定するために副表名のリストが使われます。

traversal-order-list

型付き表で INSERT、INSERT_UPDATE、または REPLACE オプションを指定した場合、インポートする階層内の副表の走査順序を指定するために副表名のリストを使います。

UNDER sub-table-name

1 つ以上の副表を作成する場合に親表を指定します。

WARNINGCOUNT n

n 個の警告後に、インポート操作を停止します。このパラメーターは、警告は出ないはずであるけれども、正しいファイルと表が使用されているかどうかを検査したい場合に設定してください。インポート・ファイルまたはターゲット表を誤って指定した場合、インポート・ユーティリティーは、インポートを試みた行ごとに警告を生成して、それがインポートの失敗の原因になります。 n をゼロにした場合、またはこのオプションを指定しない場合、発行された警告の回数に関係なくインポート操作は続行します。

例:

例 1

次に示すのは myfile.ixf から STAFF 表に情報をインポートする方法の例です。

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff
```

SQL3150N PC/IXF ファイルの H レコードには、製品 "DB2 01.00"、日付 "19970220"、および時刻 "140848" が入っています。

SQL3153N PC/IXF ファイルの T レコードは、名前 "myfile"、修飾子 " "、およびソース " " を持っています。

SQL3109N ユーティリティーが、ファイル "myfile" からデータのロードを

IMPORT

開始しています。

SQL3110N ユーティリティが処理を完了しました。"58" 行が、入力ファイルから読み取られました。

SQL3221W ...COMMIT WORK が開始されました。入力レコード数 = "58"。

SQL3222W ...すべてのデータベース変更のコミットが成功しました。

SQL3149N "58" 行が、入力ファイルから処理されました。"58" 行が表に挿入され、成功しました。"0" 行がリジェクトされました。

例 2

下記に示す例は、DEL 形式のデータをもった入力ファイル delfile1 から、表 MOVIE TABLE をインポートする方法を示す例です。

```
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. この表には下記の 4 つの列が示されます。

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

2. 入力ファイルの中の DATALINK データのサブフィールド区切り文字は、縦線 (|) 文字です。
3. url_making_of の列値に接頭部文字シーケンスが組み込まれていない場合、"http://narang" が使用されます。
4. url_movie の非 NULL 列値には、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 の非 NULL 列値のパスには、".mpeg" が付加されます。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となります。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となります。

例 3 (ID 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 TABLE1 と同じですが、C2 が GENERATED ALWAYS ID 列である点が異なります。

DATAFILE1 のデータ・レコード (DEL フォーマット):


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

DATAFILE2 のデータ・レコード (DEL フォーマット):

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

以下のコマンドは、DATAFILE1 で行 1 および 2 への ID 値が入力されていないので、それらの行のための ID 値を生成します。ただし、行 3 および 4 は、それぞれユーザー提供の ID 値 100 と 101 が割り当てられます。

```
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対する ID 値を生成するには、以下のコマンドのいずれかを発行します。

```
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 にインポートして各行に対する ID 値を生成するには、以下のコマンドのいずれかを発行します。

```
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
```

DATAFILE1 を TABLE2 に、ID 関連のファイル・タイプ修飾子を使用せずにインポートした場合、行 1 と 2 は挿入されますが、行 3 と 4 はリジェクトされます。その理由は、それらが固有の非 NULL を提供し、ID 列が GENERATED ALWAYS であるからです。

使用上の注意:

インポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。

インポート・ユーティリティーは、SQL INSERT ステートメントを使ってターゲット表に行を追加します。ユーティリティーは入力ファイル中のデータの行ごとに 1 つの INSERT ステートメントを発行します。INSERT ステートメントが失敗すると、以下のどちらかのアクションが起きます。

- 後続の INSERT ステートメントを正常に実行できると思われる場合、メッセージ・ファイルに警告メッセージが書き込まれ、処理は続行します。
- 後続の INSERT ステートメントが失敗しそうで、データベースの損傷の可能性がある場合、メッセージ・ファイルにエラー・メッセージが書き込まれ、処理は停止します。

ユーティリティーは、REPLACE または REPLACE_CREATE 操作中に、古い行が削除された後、自動 COMMIT を実行します。したがって、表オブジェクトが切り捨

てられた後、システムに障害が起こったり、アプリケーションがデータベース・マネージャーに割り込んだりすると、元のデータがすべて失われてしまいます。これらのオプションを使用する前に、古いデータがもう必要ないことを必ず確認してください。

ログが CREATE、REPLACE、または REPLACE_CREATE 操作中にいっぱいになった場合、ユーティリティーは挿入されたレコード上で自動 COMMIT を実行します。自動 COMMIT の後に、システムに障害が起こるか、またはアプリケーションがデータベース・マネージャーに割り込むと、部分的にデータの挿入された表はデータベース内に残ります。REPLACE または REPLACE_CREATE オプションを使用してインポート操作全体をやり直すか、または正常にインポートされる行の数に設定した RESTARTCOUNT パラメーターを指定して INSERT を使用してください。

デフォルトでは、自動 COMMIT は INSERT または INSERT_UPDATE オプションには実行されません。しかし、COMMITCOUNT パラメーターがゼロでない場合は実行されます。自動の COMMIT が実行されない場合にログが満杯になると、ROLLBACK が実行されます。

以下のいずれかの条件が真であると、オフライン・インポートでは自動の COMMIT は実行されません。

- ターゲットは表ではなくビューである。
- コンパウンド挿入を使用している。
- バッファー挿入を使用している。

デフォルトでは、オンライン・インポートは自動 COMMIT を実行して、アクティブ・ログ・スペースとロック・リストを両方とも解放します。自動 COMMIT が実行されないのは、ゼロの COMMITCOUNT 値を指定した場合のみです。

インポート・ユーティリティーが COMMIT を実行するときにはいつでも、2 つのメッセージがメッセージ・ファイルに書き込まれます。1 つはコミットされるレコードの数を示し、もう 1 つは正常に終了した COMMIT の後に書き込まれます。失敗の後にインポート操作を再始動する場合、最後に正常に終了した COMMIT から決定されたとおり、スキップするレコードの数を指定してください。

インポート・ユーティリティーは、小さい非互換性問題 (たとえば、文字データは埋め込みまたは切り捨てを使ってインポートでき、数値データは異なる数値データ型を使ってインポートできる) は受け入れますが、大きな非互換性に関する問題は受け入れません。

それ自体以外への依存があるオブジェクト表や、基本表に何らかの依存 (それ自体も含めて) があるオブジェクト・ビューを、REPLACE または REPLACE_CREATE することはできません。このような表またはビューを置換するには、次のようにします。

1. 表が親であるすべての外部キーをドロップします。
2. インポート・ユーティリティーを実行します。
3. 表を変更して外部キーを再作成します。

外部キーの再作成中にエラーが発生する場合、参照保全を保守するためにデータを変更してください。

参照制約および外部キー定義は、PC/IXF ファイルから表を作成する場合は保存されません。(主キー定義は、データが前に SELECT * を使ってエクスポートされた場合、保存されます。)

リモート・データベースへのインポートには、入力データ・ファイルのコピー、出力メッセージ・ファイル、およびデータベースがサイズが大きくなる可能性に備えて、十分なディスク・スペースをサーバー上に確保する必要があります。

インポート操作がリモート・データベースに対して実行され、出力メッセージ・ファイルが非常に長い (60KB より長い) 場合、クライアント上でユーザーに戻されるメッセージ・ファイルがインポート操作中に欠落することがあります。メッセージ情報の最初の 30KB と最後の 30KB は、常に保存されます。

PC/IXF ファイルのリモート・データベースへのインポートは、PC/IXF ファイルがディスクにあるときよりも、ハード・ディスクにあるときの方がより速く行うことができます。

ASC、DEL、または WSF のファイル形式をインポートするためには、それ以前にデータベース表または階層がすでに存在していなければなりません。ただし、表がまだ存在していない場合でも、IMPORT CREATE または IMPORT REPLACE_CREATE を使えば、PC/IXF ファイルからデータをインポートする際に表が作成されます。型付き表の場合、IMPORT CREATE によってタイプ階層と表階層も作成されます。

データ (階層データを含む) を別のデータベースに移動するには、PC/IXF インポートを使う必要があります。行区切り文字の入った文字データを区切り文字付き ASCII (DEL) ファイルにエクスポートし、テキスト転送プログラムにより処理を行うと、行区切り文字の入ったフィールドは長さが伸縮します。ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。

ここでは、ASC ファイルおよび DEL ファイル内のデータは、インポートを実行するクライアント・アプリケーションのコード・ページのデータであると想定します。別々のコード・ページにデータをインポートする場合、PC/IXF ファイル (異なるコード・ページへのインポートが考慮されたファイル) の使用をお勧めします。PC/IXF ファイルとインポート・ユーティリティーが同一のコード・ページの場合、処理は通常のアプリケーションと同じようになります。両者が異なるコード・ページであっても、FORCEIN オプションが指定されている場合には、インポート・ユーティリティーにおいて、PC/IXF ファイル内のデータが、インポートを実行するアプリケーションと同一のコード・ページであると見なされます。この処理は、それら 2 つのコード・ページ用の変換テーブルが存在する場合であっても行われます。それぞれのコード・ページが異なっており、FORCEIN オプションが指定されておらず、変換テーブルが存在する場合、PC/IXF ファイルのすべてのデータは、そのファイルのコード・ページからアプリケーションのコード・ページに変換されます。それぞれのコード・ページが異なっており、FORCEIN オプションが指定されておら

ず、変換テーブルが存在しない場合、インポート操作は失敗します。これが該当するのは、AIX オペレーティング・システムの DB2 UDB クライアント上の PC/IXF ファイルの場合だけです。

8 KB ページ上の表オブジェクトの量が 1012 列の制限に近い場合、PC/IXF データ・ファイルをインポートすると、SQL ステートメントの最大サイズを超過するため、DB2 はエラーを戻します。このようになるのは、CHAR、VARCHAR、または CLOB 型の列の場合だけです。DEL または ASC ファイルのインポートでは、この制限は当てはまりません。PC/IXF ファイルを使って新しい表を作成している場合、別の方法として、**db2look** を使って表を作成した DDL ステートメントをダンプしてから、そのステートメント CLP から発行する、という方法があります。

DB2 Connect を使用すると、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーにデータをインポートできます。PC/IXF インポート (INSERT オプション) だけがサポートされています。RESTARTCOUNT パラメーターもサポートされていますが、COMMITCOUNT パラメーターはサポートされていません。

型付き表に対して CREATE オプションを使うと、PC/IXF ファイルの中で定義されているすべての副表が作成されます。副表定義は変更されません。型付き表に対して CREATE 以外のオプションを使うと、走査順序リストによって、走査順序を指定できます。その場合、走査順序リストはエクスポート操作で使用されたものと一致していなければなりません。PC/IXF ファイル・フォーマットの場合は、ターゲット副表の名前を指定して、ファイルに格納されている走査順序を使用するだけです。

インポート・ユーティリティーを使って、前に PC/IXF ファイルにエクスポートされた表をリカバリーできます。表は、エクスポート時の状態に戻ります。

データは、システム表、宣言一時表、またはサマリー表にはインポートできません。

インポート・ユーティリティーを介してビューを作成することはできません。

Windows オペレーティング・システムの場合は、以下のとおりです。

- 論理分割 PC/IXF ファイルのインポートはサポートされていません。
- 正しくない DEL フォーマット・ファイルの PC/IXF または WSF ファイルのインポートはサポートされていません。

DB2 Data Links Manager の考慮事項:

DB2 インポート・ユーティリティーを実行する場合、その前に下記のことを実行しておいてください。

1. 参照されるファイルを、適切なデータ・リンク・サーバーにコピーする。
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. 必要な接頭部名を DB2 Data Links Manager に登録する。さらに、必要な場合データベースを登録するなど、その他の管理作業も必要になることがあります。

3. 必要な場合、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においても同じ場合、そのデータ・リンク・サーバー名は更新する必要はありません。)
4. DB2 Data Links Manager 構成ファイルのターゲット構成で、データ・リンク・サーバーを定義する。

インポート・ユーティリティーがターゲット・データベースに対して実行される場合、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。

挿入操作の間、DATALINK 列の処理では、ターゲット・データベースでの列指定に従って、該当するデータ・リンク・サーバー中のファイルがリンクされます。

関連概念:

- 29 ページの『インポートの概要』
- 31 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』

関連タスク:

- 31 ページの『インポートの使用』

関連資料:

- 55 ページの『db2Import - インポート』
- 「コマンド・リファレンス」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』
- 79 ページの『インポート・セッション - CLP の例』
- 113 ページの『LOAD』
- 68 ページの『インポート用のファイル・タイプ修飾子』
- 249 ページの『データ移動での区切り文字の制限』

db2Import - インポート

サポートされているファイル・フォーマットを用いて、外部ファイルから表、階層、またはビューにデータを挿入します。これに代わる Load の方がより高速ですが、ロード・ユーティリティーは、階層レベルでのデータのロードをサポートしていません。

許可:

- INSERT オプションを使用して IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 関係するそれぞれの表またはビューに対する CONTROL 特権
 - 関係するそれぞれの表またはビューに対する INSERT および SELECT 特権
- INSERT_UPDATE オプションを使用して、既存の表に IMPORT する場合、以下のいずれかが必要です。

- *sysadm*
- *dbadm*
- 表またはビューに対する CONTROL 特権
- 関係するそれぞれの表またはビューに対する INSERT、SELECT、UPDATE および DELETE 特権
- REPLACE または REPLACE_CREATE オプションを使用して、既存の表に IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 表またはビューに対する CONTROL 特権
 - 表またはビューに対する INSERT、SELECT、および DELETE 特権
- CREATE または REPLACE_CREATE オプションを使用して新規の表に IMPORT する場合、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
 - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
 - スキーマに対する CREATIN 特権 (表のスキーマ名が既存のスキーマを指す場合)
- CREATE または REPLACE_CREATE オプションを使って、存在しない表または階層に IMPORT するには、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - データベースに対する CREATETAB 権限と、次のいずれか
 - データベースに対する IMPLICIT_SCHEMA 権限 (表のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (表のスキーマが存在する場合)
 - 階層全体に対して REPLACE_CREATE オプションが使用されている場合は、階層内のすべての副表に対する CONTROL 特権
- REPLACE オプションを使用して既存の階層に IMPORT するには、以下のいずれかが必要です。
 - *sysadm*
 - *dbadm*
 - 階層内のすべての副表に対する CONTROL 特権

必要な接続:

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```

4  /* db2Import - API */
4  SQL_API_RC SQL_API_FN
4  db2Import (
4      db2UInt32 versionNumber,
4      void * pParmStruct,
4      struct sqlca * pSqlca);
4
4  /* db2Import parameter structure */
4  typedef SQL_STRUCTURE db2ImportStruct
4  {
4      char                                *piDataFileName;
4      struct sqlu_media_list              *piLobPathList;
4      struct sqldcol                      *piDataDescriptor;
4      struct sqlchar                      *piActionString;
4      char                                *piFileType;
4      struct sqlchar                      *piFileTypeMod;
4      char                                *piMsgFileName;
4      db2int16                            iCallerAction;
4      struct db2ImportIn                  *piImportInfoIn;
4      struct db2ImportOut                 *poImportInfoOut;
4      db2int32                            *piNullIndicators;
4  } db2ImportStruct;
4
4  /* Import input structure */
4  typedef SQL_STRUCTURE db2ImportIn
4  {
4      db2UInt64                            iRowcount;
4      db2UInt64                            iRestartcount;
4      db2UInt64                            iSkipcount;
4      db2int32                             *piCommitcount;
4      db2UInt32                             iWarningcount;
4      db2UInt16                             iNoTimeout;
4      db2UInt16                             iAccessLevel;
4  } db2ImportIn;
4
4  /* Import output structure */
4  typedef SQL_STRUCTURE db2ImportOut
4  {
4      db2UInt64                            oRowsRead;
4      db2UInt64                            oRowsSkipped;
4      db2UInt64                            oRowsInserted;
4      db2UInt64                            oRowsUpdated;
4      db2UInt64                            oRowsRejected;
4      db2UInt64                            oRowsCommitted;
4  } db2ImportOut;

```

汎用 API 構文:

```

4  /* db2gImport - Generic API */
4  SQL_API_RC SQL_API_FN
4  db2gImport (
4      db2UInt32 versionNumber,
4      void * pParmStruct,
4      struct sqlca * pSqlca);
4
4  /* db2gImport parameter structure */
4  typedef SQL_STRUCTURE db2gImportStruct
4  {
4      char                                *piDataFileName;
4      struct sqlu_media_list              *piLobPathList;
4      struct sqldcol                      *piDataDescriptor;
4      struct sqlchar                      *piActionString;
4      char                                *piFileType;
4      struct sqlchar                      *piFileTypeMod;
4      char                                *piMsgFileName;

```

db2Import - インポート

```
4          db2int16          iCallerAction;
4          struct db2gImportIn *piImportInfoIn;
4          struct db2gImportOut *poImportInfoOut;
4          db2int32          *piNullIndicators;
4          db2UInt16          iDataFileNameLen;
4          db2UInt16          iFileTypeLen;
4          db2UInt16          iMsgFileNameLen;
4      } db2gImportStruct;
4
4      /* Generic Import input structure */
4      typedef SQL_STRUCTURE db2gImportIn
4      {
4          db2UInt64          iRowcount;
4          db2UInt64          iRestartcount;
4          db2UInt64          iSkipcount;
4          db2int32          *piCommitcount;
4          db2UInt32          iWarningcount;
4          db2UInt16          iNoTimeout;
4          db2UInt16          iAccessLevel;
4      } db2gImportIn;
4
4      /* Generic Import output structure */
4      typedef SQL_STRUCTURE db2gImportOut
4      {
4          db2UInt64          oRowsRead;
4          db2UInt64          oRowsSkipped;
4          db2UInt64          oRowsInserted;
4          db2UInt64          oRowsUpdated;
4          db2UInt64          oRowsRejected;
4          db2UInt64          oRowsCommitted;
4      } db2gImportOut;
```

API パラメーター:

versionNumber

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

pParmStruct

入出力。 *db2ImportStruct* 構造を指すポインター。

pSqlca

出力。 *sqlca* 構造へのポインター。

piDataFileName

入力。データがインポートされるパスおよび外部入力ファイル名の入ったストリングを指定します。

piLobPathList

入力。 *media_type* *SQLU_LOCAL_MEDIA* を使用する *sqlu_media_list*、および LOB ファイルがあるクライアント上のパスをリストする *sqlu_media_entry* 構造を示します。

piDataDescriptor

入力。外部ファイルからインポートするよう選択された列に関する情報の入った *sqldcol* 構造を指すポインターです。 *dcolmeth* フィールドの値によって、このパラメーターに提供される残りの情報をインポート・ユーティリティーがどのように解釈するかが判別されます。このパラメーターの有効値は以下のとおりです。

SQL_METH_N

名前。外部入力ファイルの列は、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無効であるロケーションの対を指定したインポート呼び出しは、データベース・マネージャによってリジェクトされます。

- 開始または終了ロケーションが有効な範囲 (1 から符号付き 2 バイト整数の最大値まで) に入っていない場合。
- 終了ロケーションが開始ロケーションよりも小さい場合。
- ロケーションの対により定義された入力列幅に、ターゲット列のタイプおよび長さとの互換性がない場合。

開始ロケーションと終了ロケーションの対がゼロに等しい場合は、NULL 可能列が NULL で埋め込まれることを示します。

SQL_METH_D

デフォルト。 *piDataDescriptor* が NULL の場合、または SQL_METH_D に設定されている場合、外部入力ファイルの列のデフォルト選択が実行されます。この場合、列数および列指定配列は、どちらも無視されます。DEL、IXF、または WSF ファイルの場合、外部入力ファイルにある最初の *n* 個の列のデータは、そのままの順序で取り出されます。 *n* は、データがインポートされるデータベース列の数です。

piActionString

入力。 2 バイトの長さフィールドと、データをインポートする列を識別する文字の配列の入った *sqlchar* 構造を指すポインターです。

文字配列の形式は、以下のようになります。

```
{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]}
[DATA LINK SPECIFICATION datalink-spec]
```

INSERT

既存の表データを変更することなく、インポートされたデータを表に追加します。

INSERT_UPDATE

主キー値が表にない場合はインポートした行を追加し、主キー値がある場合はそれらの行を更新に使用します。このオプションは、ターゲット表に主キーがあり、指定された (または暗黙指定された) インポートされるターゲット列のリストに、主キーのすべての列が組み込まれているときのみ有効です。このオプションは、ビューには適用されません。

REPLACE

表オブジェクトを切り捨てることによって表から既存データをすべて削除し、インポートされたデータを挿入します。表定義および索引定義は変更されません。(indexixf が *FileTypeMod* に入っていて、*FileType* が *SQL_IXF* である場合、索引は削除および置換されます。) 表がまだ定義されていない場合には、エラーが戻されます。

重要: 既存のデータを削除した後にエラーが発生した場合、そのデータは失われてしまいます。

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 ファイルにある情報を使用して表定義と列の内容が作成されます。データベース・マネージャーによりファイルが事前にエクスポートされている場合、索引も作成されます。指定された表がすでに存在している場合、エラーが戻されます。このオプションは、PC/IXF ファイル・フォーマットにのみ有効です。

REPLACE_CREATE

指定された表が定義されている場合に、PC/IXF ファイルにある PC/IXF 行情報を使用して表の内容が置き換えられます。表がまだ定義されていない場合は、指定された PC/IXF ファイルにある情報を使用して表定義と行の内容が作成されます。DB2 により PC/IXF ファイルが事前にエクスポートされている場合は、索引も作成されます。このオプションは、PC/IXF ファイル・フォーマットにのみ有効です。

重要: 既存のデータを削除した後にエラーが発生した場合、そのデータは失われてしまいます。

tname データが挿入される表、型付き表、ビュー、またはオブジェクト・ビューの名前。下位のサーバーの場合を除き、REPLACE、INSERT_UPDATE、または INSERT には、修飾または非修飾の名前を使用しなければならないようなときでも、別名を指定することができます。ビューの場合、読み取り専用ビューにすることはできません。

tcolumn-list

データが挿入される先の表またはビュー内にある列名のリスト。列名は、コンマで区切らなければなりません。列名が指定されない場合、CREATE TABLE または ALTER TABLE ステートメントで定義された列名が使用されます。型付き表に指定されている列のリストがない場合、それぞれの副表のすべての列にデータが挿入されます。

sub-table-name

CREATE オプションで 1 つまたは複数の副表を作成する際に、親表を指定します。

ALL TABLES

階層専用の暗黙キーワード。階層をインポートする際、デフォルトでは走査順序リストで指定されているすべての表がインポートされます。

HIERARCHY

階層データをインポートするよう指定します。

STARTING

階層専用のキーワード。指定された副表の名前から開始して、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。新しい階層、副階層、または副表を、指定された副表の下に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。新しい階層、副階層、または副表を、独立型の階層として作成するよう指定します。

DATALINK SPECIFICATION *datalink-spec*

DB2 Data Links Manager 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 コマンドと同じ構文を使って指定できます。

tname および *tcolumn-list* は、SQL INSERT ステートメントの *tablename* および *colname* リストに対応し、同一の制限の下にあります。

tcolumn-list 内の列と、外部列 (指定または暗黙指定された) とは、リストまたは構造における位置に従って対応付けられます (*sqldcol* 構造で指定された最初の列からのデータは、*tcolumn-list* の最初のエレメントに対応する表またはビュー・フィールドに挿入されます)。

異なる数の列が指定された場合、実際に処理される列の数は 2 つのうちの小さい方です。このことにより、エラーが発生する (一部の NULL 不可の表フィールドに入れるべき値がないため) か、または情報メッセージが表示される (一部の外部ファイル列が無視されるため) 可能性があります。

piFileType

入力。外部ファイル内のデータのフォーマットを示すストリングを指定します。サポートされている外部ファイルのフォーマットは、以下のとおりです。

SQL_ASC

区切りなし ASCII。

SQL_DEL

区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、IBM パーソナル・デシジョン・シリーズ・プログラム、およびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための形式です。

SQL_IXF

IXF (統合交換フォーマットの PC バージョン)。表からデータをエ

クスポートする場合の推奨方式で、同じ表または別のデータベース・マネージャー表にそれを再インポートすることが可能です。

SQL_WSF

ワークシート・フォーマット。 Lotus Symphony および 1-2-3 プログラムとの交換のためのフォーマットです。

piFileTypeMod

入力。 2 バイトの長さフィールドと、 1 つまたは複数の処理オプションを指定する文字の配列の入った構造を指すポインターです。このポインターが NULL であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。『インポート用のファイル・タイプ修飾子』を参照してください。

piMsgFileName

入力。このユーティリティーが戻すエラー、警告、および情報メッセージの宛先の入ったストリングを指定します。オペレーティング・システム・ファイルまたは標準装置のパスおよび名前を指定できます。ファイルがすでに存在する場合は、そのファイルが付加されます。存在していない場合は、新たに作成されます。

iCallerAction

入力。呼び出し側が要求するアクションを示します。有効な値は以下のとおりです。

SQLU_INITIAL

最初の呼び出し。この値は、API への最初の呼び出しの際には必ず使用してください。

最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたインポート操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定しなければなりません。

SQLU_CONTINUE

処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力（たとえば、テープの終わり条件への応答）を要求した後で、API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが完了したら、ユーティリティーが最初の要求の処理を続行するよう指定するものです。

SQLU_TERMINATE

処理の終了。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力（たとえば、テープの終わり条件への応答）を要求した後で、API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが実行されなかった場合、ユーティリティーが最初の要求の処理を中断するよう指定するものです。

piImportInfoIn

入力。 *db2ImportIn* 構造を指すポインター。

piImportInfoOut

出力。 *db2ImportOut* 構造を指すポインター。

piNullIndicators

入力。 ASC ファイルの場合にのみ使用します。列データが NULL 可能であるかどうかを示す整数の配列です。この配列内のエレメント数は、入力ファイル内の列数と一致していなければなりません。この配列のエレメントとデータ・ファイルからインポートされる列との間には、1 対 1 の順序付けられた対応関係があります。このため、エレメントの数は、

piDataDescriptor 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。配列の各エレメントには、NULL 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数値、または表の列が NULL 可能ではないことを示すゼロが入っています。エレメントがゼロでない場合は、データ・ファイル内の識別された列に、Y と N のどちらかが入っているはずです。Y は表の列のデータが NULL であることを示し、N は表の列のデータが NULL ではないことを示します。

iRowcount

入力。ロードされる物理レコードの数。これを使用すると、ファイル内の最初の *iRowcount* 個の行だけをロードすることができます。 *iRowcount* がゼロの場合、インポートではファイルのすべての行の処理が試みられます。

iSkipcount

入力。レコードを挿入または更新する前にスキップするレコードの数。機能的には *iRestartcount* と同等。

piCommitcount

入力。データベースにコミットする前にインポートするレコードの数。コミットは、 *piCommitcount* 個のレコードがインポートされるたびに実行されます。 NULL 値はデフォルトのコミット・カウント値を指定しますが、それは、オフライン・インポートではゼロ、オンライン・インポートでは AUTOMATIC です。 Commitcount AUTOMATIC を指定するには、値 DB2IMPORT_COMMIT_AUTO に入れて引き渡します。

iWarningcount

入力。 *iWarningcount* 個の警告後に、インポート操作を停止します。このパラメーターは、警告は出ないはずであるけれども、正しいファイルと表が使用されているかどうかを検査したい場合に設定してください。インポート・ファイルまたはターゲット表を誤って指定した場合、インポート・ユーティリティーは、インポートを試みた行ごとに警告を生成して、それがインポートの失敗の原因になります。 *iWarningcount* を 0 にした場合、またはこのオプションを指定しない場合、発行された警告の回数に関係なくインポート操作は続行します。

iNoTimeout

入力。インポート・ユーティリティーは、ロック待ちの間にタイムアウトしないことを指定します。このオプションの方が、 *locktimeout* データベース構成パラメーターより優先されます。他のアプリケーションは影響を受けません。有効な値は以下のとおりです。

DB2IMPORT_LOCKTIMEOUT

locktimeout 構成パラメーターの値が優先されることを示します。

DB2IMPORT_NO_LOCKTIMEOUT

タイムアウトはないことを示します。

iAccessLevel

入力。アクセス・レベルを指定します。有効な値は以下のとおりです。

SQLU_ALLOW_NO_ACCESS

インポート・ユーティリティーが表を排他ロックするように指定します。

SQLU_ALLOW_WRITE_ACCESS

インポートの進行中も表内のデータに読み取りおよび書き出しのために引き続きアクセスできることを指定します。

oRowsRead

出力。インポート中にファイルから読み取られたレコードの数。

oRowsSkipped

出力。挿入または更新を開始する前にスキップしたレコードの数。

oRowsInserted

出力。ターゲット表に挿入された行の数。

oRowsUpdated

出力。インポートされたレコード (主キーの値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、ターゲット表内の行数。

oRowsRejected

出力。インポートできなかったレコードの数。

oRowsCommitted

出力。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です。

使用上の注意:

インポート操作を開始する前に、すべての表操作が完了し、すべてのロックが解除されていることを確認してください。それには、**WITH HOLD** でオープンしたカーソルをすべてクローズした後で **COMMIT** を発行するか、または **ROLLBACK** を発行します。

インポート・ユーティリティーは、**SQL INSERT** ステートメントを使用してターゲット表に行を追加します。このユーティリティーは、入力ファイル中の各行のデータにつき 1 つずつ **INSERT** ステートメントを発行します。**INSERT** ステートメントが失敗した場合、以下の 2 通りの結果のいずれかになります。

- 後続の **INSERT** ステートメントが成功すると予測される場合には、警告メッセージがメッセージ・ファイルに書き込まれ、処理が継続されます。
- 後続の **INSERT** ステートメントが失敗すると予測され、データベースが損傷する可能性がある場合には、エラー・メッセージが書き込まれ、処理が停止されます。

このユーティリティーは、REPLACE または REPLACE_CREATE 操作時に以前の行が削除された後、自動 COMMIT を実行します。したがって、表オブジェクトが切り捨てられた後、システムに障害が起こったり、アプリケーションがデータベース・マネージャーに割り込んだりすると、元のデータがすべて失われてしまいます。これらのオプションを使用する前に、古いデータがもう必要ないことを必ず確認してください。

CREATE、REPLACE、または REPLACE_CREATE 操作時にログが満杯になると、このユーティリティーは挿入されたレコードに対して自動 COMMIT を実行します。自動 COMMIT の後に、システムに障害が起こるか、またはアプリケーションがデータベース・マネージャーに割り込むと、部分的にデータの挿入された表はデータベース内に残ります。REPLACE または REPLACE_CREATE オプションを使用してインポート操作全体を再実行するか、*iRestartcount* パラメーターを正常にインポートされた行の数に設定して INSERT を使用してください。

デフォルトでは、INSERT または INSERT_UPDATE オプションでの自動 COMMIT は実行されません。ただし、**piCommitcount* パラメーターがゼロでない場合は実行されます。ログが満杯になると、ROLLBACK が実行されます。

インポート・ユーティリティーが COMMIT を実行するときにはいつでも、2 つのメッセージがメッセージ・ファイルに書き込まれます。1 つはコミットされるレコードの数を示し、もう 1 つは正常に終了した COMMIT の後に書き込まれます。失敗の後にインポート操作を再始動する場合、最後に正常に終了した COMMIT から決定されたとおり、スキップするレコードの数を指定してください。

インポート・ユーティリティーは、小さい非互換性問題（たとえば、文字データは埋め込みまたは切り捨てを使ってインポートでき、数値データは異なる数値データ型を使ってインポートできる）は受け入れますが、大きな非互換性に関する問題は受け入れません。

オブジェクト表にその表自身以外との依存関係がある場合のオブジェクト表に対して、またはその基本表にその表自身も含めて依存関係がある場合のオブジェクト・ビューに対して、REPLACE または REPLACE_CREATE を実行することはできません。そのような表またはビューを置換するには、以下のようにしてください。

1. その表が親となっているすべての外部キーをドロップします。
2. インポート・ユーティリティーを実行します。
3. 表を変更して、外部キーを再作成します。

外部キーの作成中にエラーが発生した場合には、データを修正して参照保全を保持してください。

PC/IXF ファイルから表を作成するときは、参照制約と外部キー定義は保存されません。（以前に SELECT * を使用してエクスポートされたデータの場合、主キー定義は保持されます。）

リモート・データベースへのインポートでは、入力データ・ファイルのコピー用、出力メッセージ・ファイル用、およびデータベースのサイズ拡大に備えた十分なディスク・スペースがサーバーで必要になります。

リモート・データベースに対してインポート操作を実行した場合に、出力メッセージ・ファイルが非常に長い (60KB より長い) と、クライアントのユーザーに戻されるメッセージ・ファイル中のメッセージがインポート操作の途中で失われることがあります。メッセージ情報の最初の 30 KB と最後の 30 KB は、常に保存されます。

PC/IXF ファイルのリモート・データベースへのインポートは、PC/IXF ファイルがディスクットにあるときよりも、ハード・ディスクにあるときの方がより速く行うことができます。 *piDataDescriptor* でデフォルト以外の値を使用したり、 *piActionString* で明示的な表の列のリストを指定したりすると、リモート・データベースへのインポート速度は遅くなります。

ASC、DEL、または WSF のファイル形式をインポートするためには、それ以前にデータベース表または階層がすでに存在していなければなりません。ただし、表がまだ存在していない場合でも、IMPORT CREATE または IMPORT REPLACE_CREATE を使えば、PC/IXF ファイルからデータをインポートする際に表が作成されます。型付き表の場合、IMPORT CREATE はタイプ階層と表階層も作成することができます。

データ (階層データを含む) を別のデータベースに移動するには、PC/IXF インポートを使う必要があります。行区切り文字の入った文字データが区切り付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字の入ったフィールドは長さが変わることがあります。

ASC および DEL ファイルのデータは、インポートを実行するクライアント・アプリケーションのコード・ページであると見なされます。さまざまなコード・ページのデータをインポートする場合は、さまざまなコード・ページの使用を許容する PC/IXF ファイルをお勧めします。PC/IXF ファイルとインポート・ユーティリティーが同じコード・ページである場合は、通常のアプリケーションの場合のように処理が行われます。それぞれのコード・ページが異なっており、FORCEIN オプションが指定されている場合、インポート・ユーティリティーは、PC/IXF ファイルのデータのコード・ページと、インポートを実行中のアプリケーションのコード・ページが同じであると見なします。この処理は、それら 2 つのコード・ページ用の変換テーブルが存在する場合であっても行われます。それぞれのコード・ページが異なっており、FORCEIN オプションが指定されておらず、変換テーブルが存在する場合、PC/IXF ファイルのすべてのデータは、そのファイルのコード・ページからアプリケーションのコード・ページに変換されます。それぞれのコード・ページが異なっており、FORCEIN オプションが指定されておらず、変換テーブルが存在しない場合、インポート操作は失敗します。これが該当するのは、DB2 for AIX クライアント上の PC/IXF ファイルの場合だけです。

8KB ページ上の表オブジェクトの量が 1012 列の制限に近い場合に、PC/IXF データ・ファイルをインポートすると、SQL ステートメントの最大サイズを超過するため、DB2 はエラーを戻します。この状態が発生する可能性があるのは、列が CHAR、VARCHAR、または CLOB タイプの場合だけです。DEL または ASC ファイルのインポートには、この制限事項は適用されません。

DB2 Connect は、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーにデータをインポートするのに使用できます。サポ

ートされているのは、PC/IXF インポート (INSERT オプション) だけです。
restartcnt パラメーターもサポートされていますが、commitcnt パラメーターはサポ
ートされていません。

型付き表で CREATE オプションを使用するときは、PC/IXF ファイルで定義され
ているすべての副表を作成してください。副表の定義は変更できません。型付き表
で CREATE 以外のオプションを使用するときは、走査順序リストによって走査順
序を指定できます。このため、走査順序リストはエクスポート操作時に使用したも
のト一致する必要があります。PC/IXF ファイル・フォーマットの場合は、ターゲ
ット副表の名前を指定して、ファイルに格納されている走査順序を使用するだけ
です。

インポート・ユーティリティーを使って、前に PC/IXF ファイルにエクスポートさ
れた表をリカバリーできます。表は、エクスポート時の状態に戻ります。

データは、システム表、宣言一時表、またはサマリー表にはインポートできま
せん。

インポート・ユーティリティーを使用して、ビューを作成することはできません。

Windows NT オペレーティング・システムの場合は、以下のとおりです。

- ・ 論理分割された PC/IXF ファイルのインポートはサポートされていません。
- ・ 不正な DEL フォーマット・ファイルの PC/IXF または WSF ファイルのインポ
ートは、サポートされていません。

DB2 Data Links Manager についての考慮事項

DB2 インポート・ユーティリティーを実行する場合、その前に下記のことを実行し
ておいてください。

1. 参照されているファイルを適切なデータ・リンク・サーバーにコピーする。
dlfm_import ユーティリティーを使用すれば、**dlfm_export** ユーティリティー
で生成されたアーカイブからファイルを抽出することができます。
2. 必要な接頭名を DB2 Data Links Manager に登録する。必要な場合は、データベ
ースの登録など他の管理タスクを行うこともできます。
3. 必要な場合は、(DATALINK 列の) URL のデータ・リンク・サーバー情報を、
SQL 表のエクスポート済みデータから更新する。(元の構成のデータ・リンク・
サーバーのターゲット位置が同じである場合、データ・リンク・サーバー名を更
新する必要はありません。)
4. データ・リンク・サーバーを、DB2 Data Links Manager 構成ファイルのターゲ
ット構成で定義する。

インポート・ユーティリティーをターゲット・データベースに対して実行する場
合は、DATALINK 列データが参照するファイルは、適切なデータ・リンク・サー
バー上でリンクします。

挿入操作の間、DATALINK 列の処理では、ターゲット・データベースでの列指定に
従って、該当するデータ・リンク・サーバー中のファイルがリンクされます。

関連資料:

- 「管理 API リファレンス」の『SQLCA』
- 「管理 API リファレンス」の『SQLDCOL』
- 「管理 API リファレンス」の『SQLU-MEDIA-LIST』
- 68 ページの『インポート用のファイル・タイプ修飾子』
- 249 ページの『データ移動での区切り文字の制限』

関連サンプル:

- 『dtformat.sqc -- Load and import data format extensions (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『exp samp.sqb -- Export and import tables with table data to a DRDA database (IBM COBOL)』
- 『impexp.sqb -- Export and import tables with table data (IBM COBOL)』
- 『tbmove.sqC -- How to move table data (C++)』

インポート用のファイル・タイプ修飾子

表 4. インポートで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット

修飾子	説明
compound= <i>x</i>	<p><i>x</i> は 1 から 100 の数字です。非アトミック・コンパウンド SQL を使用してデータを挿入します。毎回 <i>x</i> 個のステートメントが試行されます。</p> <p>この修飾子が指定され、トランザクション・ログに十分な大きさがいない場合、インポート操作は失敗します。トランザクション・ログは、COMMITCOUNT によって指定された行数か、または COMMITCOUNT が指定されていない場合はデータ・ファイルの行数を入れる十分な大きさが必要です。したがって、トランザクション・ログのオーバーフローを避けるために、COMMITCOUNT オプションを指定することをお勧めします。</p> <p>この修飾子は、INSERT_UPDATE モード、階層表、および修飾子 usedefaults、identitymissing、identityignore、generatedmissing、generatedignore とは互換性がありません。</p>
generatedignore	<p>この修飾子は、インポート・ユーティリティに、すべての生成列のデータはデータ・ファイルに存在するが、それらを見捨てるべきことを知らせます。この結果として、生成列のすべての値は、このユーティリティによって生成されます。この修飾子は、generatedmissing 修飾子と共に使用することはできません。</p>
generatedmissing	<p>この修飾子が指定されている場合、ユーティリティは、生成列のデータが入力データ・ファイルに入っていない (NULL も入っていない) ものを見なし、行ごとに値を生成します。この修飾子は、generatedignore 修飾子と共に使用することはできません。</p>
identityignore	<p>この修飾子は、インポート・ユーティリティに、ID 列のデータはデータ・ファイルに存在するが、それらを見捨てるべきことを知らせます。この結果、すべての識別値はユーティリティによって生成されます。この動作は、GENERATED ALWAYS および GENERATED BY DEFAULT ID 列のどちらの場合も同じです。つまり、GENERATED ALWAYS 列の場合には、リジェクト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。</p>

表 4. インポートで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット (続き)

修飾子	説明
identitymissing	この修飾子を指定すると、ユーティリティーは、ID 列のデータが入力データ・ファイルに入っていない (NULL も入っていない) ものを見なし、行ごとに値を生成します。この動作は、GENERATED ALWAYS および GENERATED BY DEFAULT ID 列のどちらの場合も同じです。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	<p><i>lob-path</i> には、LOB データの入ったファイルへのパスを指定します。</p> <p>各パスには、データ・ファイル内で LOB ロケーション指定子 (LLS) によって示される 1 つ以上の LOB の入った、少なくとも 1 つのファイルが組み込まれます。LLS は、LOB ファイル・パスに保管されるファイル内の LOB のロケーションのストリング表現です。LLS のフォーマットは、<i>filename.ext.nnn.mmm/</i> です。<i>filename.ext</i> は LOB の入ったファイルの名前、<i>nnn</i> はファイル内の LOB のオフセット (バイト単位)、<i>mmm</i> は LOB の長さ (バイト単位) を表します。たとえば、ストリング <i>db2exp.001.123.456/</i> がデータ・ファイルに保存される場合、LOB はファイル <i>db2exp.001</i> のオフセット 123 に位置し、456 バイト長です。</p> <p>LOBS FROM 文節は、“lobsinfile” 修飾子が使用されているときの、LOB ファイルの場所を指定します。LOBS FROM 文節には、lobsinfile 修飾子のコンテキスト以外の意味はありません。LOBS FROM 文節は、データのインポート中に、IMPORT ユーティリティーに LOB ファイルを検索するためのパスのリストを送ります。</p> <p>NULL LOB を指定するには、サイズに -1 と入力します。サイズを 0 と指定すると、長さが 0 の LOB として扱われます。長さが -1 の NULL LOB の場合、オフセットとファイル名は無視されます。たとえば、NULL LOB の LLS は <i>db2exp.001.7.-1/</i> です。</p>
no_type_id	単一の副表にインポートするときのみ有効です。これを使う場合として典型的な例は、正規の表からデータをエクスポートした後、この修飾子を使ってインポート操作を呼び出してそのデータを単一の副表に変換する場合です。
nodefaults	<p>ターゲット表の列に対応するソース列が明示的に指定されていない場合、その表列が NULL 不可能なら、デフォルトはロードされません。このオプションを指定せず、あるターゲット表列のためのソース列が明示的に指定されていない場合、以下のいずれかになります。</p> <ul style="list-style-type: none"> • 列にデフォルトを指定できる場合、そのデフォルトがロードされます。 • 列が NULL 可能で、デフォルトがその列に指定できない場合、NULL がロードされます。 • 列が NULL 不可能で、デフォルトがその列に指定できない場合、エラーが戻され、ユーティリティーは処理を停止します。
norowwarnings	リジェクトされた行についての警告をすべて抑制します。

4

db2Import - インポート

表 4. インポートで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット (続き)

修飾子	説明
usedefaults	<p>ターゲット表の列のソース列が指定されているが、1 つまたは複数の行インスタンスのデータが入っていない場合は、デフォルト値がロードされます。欠落データの例は、以下のとおりです。</p> <ul style="list-style-type: none"> • DEL ファイルの場合、列に ".,," が指定された場合 • ASC ファイルの場合、列の NULL 標識が yes に設定された場合 • DEL/ASC/WSF ファイルの場合、列が不足している行、または元の指定では十分な長さのない行。 <p>このオプションが指定されていない場合、行インスタンスのソース列にデータがないと、以下のいずれかの処理が行われます。</p> <ul style="list-style-type: none"> • 列が NULL 可能な場合、NULL がロードされます。 • 列が NULL 不可能の場合、ユーティリティーはその行をリジェクトします。

表 5. インポートで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL)

修飾子	説明
codepage=x	<p>x は ASCII 文字ストリングです。この値は、出力データ・セット内のデータのコード・ページと解釈されます。インポート操作中に、文字データをアプリケーション・コード・ページからこのコード・ページに変換します。</p> <p>以下の規則が適用されます。</p> <ul style="list-style-type: none"> • 純 DBCS (GRAPHIC)、混合 DBCS、および EUC では、区切り文字は x00 から x3F の範囲に制限されます。 • nullindchar には、標準の ASCII セットに組み込む (コード・ポイント x20 から x7F の範囲の) 記号を指定する必要があります。これは、ASCII 記号およびコード・ポイントを示します。 <p>注:</p> <ol style="list-style-type: none"> 1. codepage 修飾子を lobsinfile 修飾子と一緒に使うことはできません。 2. コード・ページをアプリケーション・コード・ページからデータベース・コード・ページに変換している途中でデータ拡張が起こると、データが切り捨てられてデータ損失が起こる場合があります。
dateformat="x"	<p>x はソース・ファイルの日付のフォーマットです。² 有効な日付エレメントは以下のとおりです。</p> <p>YYYY - 年 (0000 から 9999 の範囲の 4 桁の数字) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。 M と相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の 2 桁の数字。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の 3 桁の数字。 他の日または月エレメントとは相互に排他的)</p> <p>デフォルト値の 1 が、指定されない各エレメントに割り当てられます。日付形式の例を以下に示します。</p> <p>"D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"</p>

表 5. インポートで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
implieddecimal	暗黙指定されている小数点の位置が列定義によって決定され、値の終わりにあるとは見なされなくなります。たとえば、値 12345 は、12345.00 ではなく、123.45 として DECIMAL(8,2) 列にロードされます。
noeofchar	オプションのファイル終了文字 x'1A' は、ファイルの終わりとして認識されません。通常の文字の場合のように処理は継続されます。
timeformat="x"	<p>x はソース・ファイル内の時刻のフォーマットです。² 有効な時刻エレメントは以下のとおりです。</p> <ul style="list-style-type: none"> H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M と相互排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) <p>デフォルト値の 0 が、指定されない各エレメントに割り当てられます。時刻フォーマットの例を以下に示します。</p> <p>"HH:MM:SS" "HH.MM TT" "SSSSS"</p>

表 5. インポートで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
timestampformat="x"	<p>x はソース・ファイル内のタイム・スタンプのフォーマットです。 ² 有効なタイム・スタンプ・エレメントは以下のとおりです。</p> <p>YYYY - 年 (0000 から 9999 の範囲の 4 桁の数字)</p> <p>M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数)</p> <p>MM - 月 (01 から 12 の 2 桁の数。M および MMM とは相互に排他的)</p> <p>MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的)</p> <p>D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数)</p> <p>DD - 日 (1 から 31 の 2 桁の数字。D とは相互に排他的)</p> <p>DDD - 元日から数えた日数 (001 から 366 の 3 桁の数字。 他の日または月のエレメントとは相互に排他的)</p> <p>H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数。)</p> <p>HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の範囲の 2 桁の数字。H と相互に排他的)</p> <p>M - 分 (0 から 59 の 1 桁または 2 桁の数字。)</p> <p>MM - 分 (0 から 59 の範囲の 2 桁の数。M (分) とは相互に排他的)</p> <p>S - 秒 (0 から 59 の 1 桁または 2 桁の数字。)</p> <p>SS - 秒 (0 から 59 の範囲の 2 桁の数。S と相互に排他的)</p> <p>SSSSS - 夜中の 12 時から数えた秒数。(00000 から 86399 の 5 桁の数。 他の時刻エレメントとは相互に排他的)</p> <p>UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数字。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数字。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的)</p> <p>TT - 午前/午後の指定子 (AM または PM)</p> <p>YYYY、M、MM、D、DD、または DDD エレメントが指定されていない場合、デフォルト値として 1 が割り当てられます。値が指定されていない MMM エレメントには、デフォルト値の「Jan」が割り当てられます。他のエレメントが指定されていない場合には、デフォルト値として 0 が割り当てられます。タイム・スタンプ・フォーマットの例を以下に示します。</p> <p>"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>MMM エレメントの有効な値は、「jan」、「feb」、「mar」、「apr」、「may」、「jun」、「jul」、「aug」、「sep」、「oct」、「nov」、および「dec」です。これらの値は大文字小文字の区別をします。</p> <p>次の例では、ユーザー定義の日時フォーマットを指示するデータを、schedule という表にインポートする方法を示します。</p> <pre>db2 import from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre>

表 5. インポートで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
usegraphiccodepage	<p>usegraphiccodepage が指定された場合、GRAPHIC または 2 バイト文字ラージ・オブジェクト (DBCLOB) データ・フィールドにインポートされるデータは、GRAPHIC コード・ページであると見なされます。データの残りは、文字コード・ページであると見なされます。GRAPHIC コード・ページは、文字コード・ページと関連付けられます。IMPORT は、codepage 修飾子 (指定されている場合)、または codepage 修飾子が指定されていない場合はアプリケーションのコード・ページを介して、文字コード・ページを決定します。</p> <p>この修飾子は、リカバリーされている表中に GRAPHIC データがある場合にのみ、表リカバリーのドロップによって生成された区切りデータ・ファイルとともに使用される必要があります。</p> <p>制約事項</p> <p>usegraphiccodepage 修飾子は、EXPORT ユーティリティーで作成された DEL または ASC ファイルで指定することはできません。usegraphiccodepage 修飾子はまた、ファイル内の 2 バイト文字ラージ・オブジェクト (DBCLOB) には無視されます。</p>

表 6. インポートで有効なファイル・タイプ修飾子: ASC (区切り文字で区切られていない ASCII) ファイル・フォーマット

修飾子	説明
nochecklengths	nochecklengths が指定されていると、ソース・データの列定義がターゲット表の列のサイズを超えるものであっても、各行のインポートが試行されます。このような行が正常にインポートされるのは、コード・ページ変換でソース・データが縮小する場合です。たとえば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小すれば、必要スペースは半分になります。このオプションが特に役立つのは、列の定義は不一致であるがソース・データが常に適合することが分かっている場合です。
nullindchar=x	<p>x は単一文字です。NULL 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。³</p> <p>文字が 1 つの英字である場合を除いて、この修飾子は、EBCDIC データ・ファイルで大文字小文字を区別します。たとえば、NULL 標識文字が文字 N と指定されている場合、n も NULL 標識と認識されます。</p>
reclen=x	x は、最大値 32 767 の整数です。各行ごとに x 個の文字が読み取られ、行の終わりを示すのに改行文字は使用されません。

db2Import - インポート

表 6. インポートで有効なファイル・タイプ修飾子: *ASC* (区切り文字で区切られていない *ASCII*) ファイル・フォーマット (続き)

修飾子	説明
striptblanks	<p>データを可変長フィールドにロードする際に、後書きブランク・スペースを切り捨てます。このオプションを指定しない場合、ブランク・スペースはそのまま保持されます。</p> <p>次の例の場合、インポート・ユーティリティーは、 <code>striptblanks</code> によって後書きブランク・スペースを切り捨てます。</p> <pre>db2 import from myfile.asc of asc modified by striptblanks method 1 (1 10, 12 15) messages msgs.txt insert into staff</pre> <p>このオプションは、 <code>striptnulls</code> と一緒に指定することはできません。これらは相互に排他的なオプションです。</p> <p>注: このオプションは以前の <code>t</code> オプションを置き換えるもので、後方互換性のためにのみサポートされています。</p>
striptnulls	<p>可変長フィールドにデータをインポートする場合に、後書き <code>NULL (0x00 文字)</code> 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、<code>NULL</code> はそのまま保持されます。</p> <p>このオプションは、 <code>striptblanks</code> と一緒に指定することはできません。これらは相互に排他的なオプションです。</p> <p>注: このオプションは、廃止された <code>padwithzero</code> オプション (後方互換性のためだけにサポートされる) に代わるものです。</p>

表 7. インポートで有効なファイル・タイプ修飾子: *DEL* (区切り文字で区切られている *ASCII*) ファイル・フォーマット

修飾子	説明
chardelx	<p><code>x</code> 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。指定した文字は、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。³⁴ 文字ストリング区切り文字として明示的に二重引用符を指定したい場合、次のように指定します。</p> <pre>modified by chardel""</pre> <p>単一引用符 (') も、文字ストリングの区切り文字として指定できます。以下の例では、<code>chardel''</code> が指定されており、インポート・ユーティリティーは検出するすべての単一引用符 (') を文字ストリングの区切り文字として解釈します。</p> <pre>db2 "import from myfile.del of del modified by chardel'' method p (1, 4) insert into staff (id, years)"</pre>
coldelx	<p><code>x</code> 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。指定した文字は、列の終わりを表すために、コンマの代わりに使用されます。³⁴</p> <p>以下の例では、<code>coldel;</code> が指定されており、インポート・ユーティリティーは検出するすべてのセミコロン (;) を列の区切り文字として解釈します。</p> <pre>db2 import from myfile.del of del modified by coldel; messages msgs.txt insert into staff</pre>
datesiso	日付形式。すべての日付データ値を <code>ISO</code> フォーマットでインポートします。

表 7. インポートで有効なファイル・タイプ修飾子: DEL (区切り文字で区切られている ASCII) ファイル・フォーマット (続き)

修飾子	説明
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) ではなくブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進数の前に正符号 (+) が付けられます。
decptx	<p>x は、小数点文字としてピリオドの代わりに使用される単一の置換文字です。デフォルト値はピリオド (.) です。指定した文字は、小数点文字としてピリオドの代わりに使用されます。³⁴</p> <p>以下の例では、decpt; が指定されており、インポート・ユーティリティーは検出するすべてのセミコロン (;) を小数点として解釈します。</p> <pre>db2 "import from myfile.del of del modified by char del' decpt; messages msgs.txt insert into staff"</pre>
delprioritychar	<p>区切り文字の現在のデフォルト優先順位は、(1) レコード区切り文字、(2) 区切り文字、(3) 列区切り文字です。この修飾子を使用すると、区切り文字の優先順位が (1) 区切り文字、(2) レコード区切り文字、(3) 列区切り文字に戻り、以前の優先順位に依存している既存のアプリケーションが保護されます。構文は以下のとおりです。</p> <pre>db2 import ... modified by delprioritychar ...</pre> <p>たとえば、以下のような DEL データ・ファイルがあるとします。</p> <pre>"Smith, Joshua",4000,34.98<row delimiter> "Vincent,<row delimiter>, is a manager", 4005,44.37<row delimiter></pre> <p>delprioritychar 修飾子が指定されている場合、このデータ・ファイルには 2 行しかありません。2 番目の <row delimiter> は 2 番目の行の最初のデータ列の一部と解釈されますが、1 番目と 3 番目の <row delimiter> は実レコードの区切り文字と解釈されます。この修飾子が指定されていない場合、このデータ・ファイルでは 3 行になり、各行は <row delimiter> によって区切られます。</p>
dlldelx	<p>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。指定した文字はセミコロンの代わりに、DATALINK 値のフィールド間区切り文字として使用されます。DATALINK 値には 2 つ以上の副値を指定できるので、この区切り文字が必要です。³⁴</p> <p>注: x は、行、列、または文字ストリングの区切り文字とは異なる文字にしてください。</p>
keepblanks	タイプが 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。このオプションを指定しないと、区切り文字で囲まれていないすべての前後のブランクは除去され、表のすべてのブランク・フィールドに NULL が挿入されます。

db2lmpport - インポート

表 7. インポートで有効なファイル・タイプ修飾子: *DEL* (区切り文字で区切られている *ASCII*) ファイル・フォーマット (続き)

修飾子	説明
nochardel	インポート・ユーティリティーは、列区切り文字の間にあるすべてのバイトを列のデータの一部であると見なします。区切り文字は、列データの一部として構文解析されます。データが <i>DB2</i> を使用してエクスポートされている場合は、このオプションを指定しないでください (エクスポート時に <i>nochardel</i> が指定されない限り)。これは、区切り文字を持たないペンダー・データ・ファイルをサポートするために用意されています。不適切に使用すると、データが損失または破壊される場合があります。
	このオプションを <i>chardelx</i> 、 <i>delprioritychar</i> または <i>nodoubledel</i> と一緒に指定することはできません。これらは相互に排他的なオプションです。
nodoubledel	二重になっている区切り文字の認識を抑止します。

表 8. インポートで有効なファイル・タイプ修飾子: *IXF* ファイル・フォーマット

修飾子	説明
forcein	コード・ページが不一致でもデータを受け入れ、コード・ページ間の変換を抑止するようにユーティリティーに指示します。 固定長ターゲット・フィールドは、データが入るだけの十分な大きさがあるかどうかチェックされます。 <i>nochecklengths</i> が指定されていると、チェックは実行されず、各行のインポートが試行されます。
indexixf	既存の表に現在定義されている索引をすべてドロップし、 <i>PC/IXF</i> ファイルの索引定義に基づいて新しい索引を作成するようにユーティリティーに指示します。このオプションを使用できるのは、表の内容を置換する場合だけです。ビューでは使用できません。また、 <i>insert-column</i> が指定されている場合にも使用できません。
indexschema= <i>schema</i>	指定した <i>schema</i> を、索引作成時の索引名として使用します。 <i>schema</i> を指定しなかった場合 (しかしキーワード <i>indexschema</i> は指定した 場合) には、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、 <i>IXF</i> ファイルのスキーマが使用されます。
nochecklengths	<i>nochecklengths</i> が指定されていると、ソース・データの列定義がターゲット表の列のサイズを超えるものであっても、各行のインポートが試行されます。このような行が正常にインポートされるのは、コード・ページ変換でソース・データが縮小する場合です。たとえば、ソースにある 4 バイトの <i>EUC</i> データがターゲットで 2 バイトの <i>DBCS</i> データに縮小すれば、必要スペースは半分にになります。このオプションが特に役立つのは、列の定義は不一致であるがソース・データが常に適合することが分かっている場合です。

注:

- サポートされていないファイル・タイプを *MODIFIED BY* オプションで使用しようとしても、インポート・ユーティリティーは警告を出しません。この場合、インポート操作が失敗し、エラー・コードが戻されます。
- 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* から *z*、*A* から *Z*、および *0* から *9* を使用することはできません。フィールド区切り文字は、区切り文字、または *DEL* ファイル・フォーマットのフィールド区切り文字と同じであってはなりません。エレメント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。あい

まいさが生じうるのは、項目の長さが一定でない D、H、M、または S などのエレメント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・フォーマットの場合、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、区別があいまいにならないように注意する必要があります。月のフィールドは、他の日付フィールドと隣接していなければなりません。分フィールドは、他の時刻フィールドに隣接していなければなりません。以下に、いくつかのあいまいなタイム・スタンプ・フォーマットを示します。

```
"M" (月または分のどちらにもとれる)
"M:M" (月と分の区別がつかない)
"M:YYYY:M" (両方とも月と解釈される)
"S:M:YYYY" (時刻値と日付値の両方に隣接している)
```

あいまいな場合、ユーティリティーはエラー・メッセージを報告し、操作は失敗します。

以下に、明確なタイム・スタンプ・フォーマットを示します。

```
"M:YYYY" (月)
"S:M" (分)
"M:YYYY:S:M" (月....分)
"M:H:YYYY:M:D" (分....月)
```

二重引用符や円記号などの文字の前には、エスケープ文字 (たとえば、¥) を付けなければなりません。

- この文字は、ソース・データのコード・ページで指定してください。

文字コード・ポイント (文字記号ではない) は、xJJ または 0xJJ という構文で指定することができます (JJ はコード・ポイントの 16 進表記)。たとえば、列区切りとして # 文字を指定するには、以下のいずれかを使用します。

```
... modified by coldel# ...
... modified by coldel0x23 ...
... modified by coldelX23 ...
```

- データ移動のための区切り文字の制約事項 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。

表 9. *codepage* および *usegraphiccodepage* 使用時の *IMPORT* 動作

codepage=N	usegraphiccodepage	IMPORT 動作
なし	なし	ファイル内のすべてのデータは、アプリケーション・コード・ページであると見なされます。
あり	なし	<p>ファイル内のすべてのデータは、コード・ページ N であると見なされます。</p> <p>警告: N が単一バイト・コード・ページの場合、GRAPHIC データをデータベースにインポートすると、壊れます。</p>

表 9. `codepage` および `usegraphiccodepage` 使用時の `IMPORT` 動作 (続き)

<code>codepage=N</code>	<code>usegraphiccodepage</code>	<code>IMPORT</code> 動作
なし	あり	<p>ファイル内の文字データは、アプリケーション・コード・ページであると見なされます。 <code>GRAPHIC</code> データは、アプリケーション <code>GRAPHIC</code> データのコード・ページであると見なされます。</p> <p>アプリケーション・コード・ページが単一バイトの場合は、すべてのデータはアプリケーション・コード・ページであると見なされます。</p> <p>警告: アプリケーション・コード・ページが単一バイトの場合、 <code>GRAPHIC</code> データは、データベースにたとえ <code>GRAPHIC</code> 列が収められていても、データベースにインポートされると壊れます。</p>
あり	あり	<p>文字データは、コード・ページ <code>N</code> であると見なされます。 <code>GRAPHIC</code> データは、<code>N</code> の <code>GRAPHIC</code> コード・ページであると見なされます。</p> <p><code>N</code> が単一バイトまたは 2 バイト・コード・ページの場合は、すべてのデータは、コード・ページ <code>N</code> であると見なされます。</p> <p>警告: <code>N</code> が単一バイト・コード・ページの場合、 <code>GRAPHIC</code> データをデータベースにインポートすると、壊れます。</p>

関連資料:

- 55 ページの『db2Import - インポート』
- 41 ページの『IMPORT』
- 249 ページの『データ移動での区切り文字の制限』

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エンコード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 `NULL` 可能であれば `NULL` がその列に挿入されます。そうでない場合、要求はリジェクトされます。 `nochecklengths` 修飾子を指定した場合は、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事が明らかになった場合、その行はリジェクトされます。そうでなければ、データはインポートされます。

関連概念:

- 187 ページの『文字セットと各国語のサポート』

関連資料:

インポート・セッション - 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 フォーマットのデータが入っている入力ファイル delfile1 から、表 MOVIE TABLE 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 dl|del|
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 (LINKTYPE URL を使用)
url_movie	DATALINK (LINKTYPE URL を使用)

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 の列値に接頭部文字シーケンスが組み込まれていない場合、"http://narang" が使用されます。
4. url_movie の NULL 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 NULL 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 (ID 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 ID 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 フォーマット):

```
"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 フォーマット):

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

以下のコマンドでは、行 1 および 2 の ID 値が生成されます。これは、DATAFILE1 内にこれらの行の ID 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 ID 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、すべての行の ID 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、それぞれの行ごとに ID 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 はリジェクトされます。これは、行 3 と 4 では独自に非 NULL 値が提供されており、ID 列が GENERATED ALWAYS であるためです。

例 4 (NULL 標識を使用したインポート)

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 つのエレメントがあります。

- ELE1、位置 01 から 20
- ELE2、位置 21 から 22
- ELE5、位置 23 から 23
- ELE3、位置 24 から 27
- ELE4、位置 28 から 31
- ELE6、位置 32 から 32
- ELE6、位置 33 から 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
```

以下のコマンドは、ASCFILE1 から TABLE1 にレコードをインポートします。

```
db2 import from ascfile1 of asc
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. 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。
2. 位置 23 および 32 は、TABLE1 の COL2 と COL3 に NULL をロードするかどうかを行ごとに示すために使用されます。特定のレコードのうち列の NULL 標識位置が Y なら、その列は NULL 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用されます。この例の場合、行 1 ではどちらの列も NULL ではなく、行 2 では COL2 が NULL、行 3 では COL3 が NULL です。
3. 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 NULL 可能でないことを示しています。
4. 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。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『ニックネームとデータ・ソース・オブジェクト』

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、装置、またはカーソルから 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。

以下のトピックを取り上げています。

- 84 ページの『ロードの概要』
- 91 ページの『並列処理とロード』
- 92 ページの『ロードの使用に必要な特権、権限、および許可』
- 93 ページの『ロードの使用』
- 95 ページの『読み取りアクセス・ロード操作』
- 97 ページの『索引の作成』
- 99 ページの『ロードでの ID 列の使用』
- 101 ページの『ロードでの生成列の使用』
- 103 ページの『保全性違反のチェック』
- 106 ページの『従属即時マテリアライズ照会表のリフレッシュ』
- 107 ページの『従属即時ステージング表の伝搬』
- 108 ページの『マルチディメンション・クラスタリングの考慮事項』
- 110 ページの『中断したロード操作の再開』
- 111 ページの『ロード・コピー・ロケーション・ファイルを使ったデータのリカバリー』
- 113 ページの『LOAD』
- 137 ページの『LOAD QUERY』
- 139 ページの『db2Load - ロード』
- 164 ページの『db2LoadQuery - ロードの照会』
- 182 ページの『ロード例外表』
- 182 ページの『ロード・ダンプ・ファイル』
- 183 ページの『ロード一時ファイル』
- 184 ページの『ロード・ユーティリティーのログ・レコード』
- 184 ページの『表ロック、表状態、および表スペース状態』
- 187 ページの『文字セットと各国語のサポート』
- 188 ページの『ロード操作後のペンディング状態』
- 189 ページの『ロードのパフォーマンスの最適化』
- 195 ページの『ロード - CLP の例』

DB2 Data Links Managerのデータのロードについては、231 ページの『インポートを使用した DB2 Data Links Manager データの移動』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は、ラージ・オブジェクト (LOB) やユーザー定義タイプ (UDT) などのほとんどのデータ・タイプを処理できます。インポート・ユーティリティは SQL INSERT を実行するのに対し、ロード・ユーティリティはフォーマット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・ユーティリティよりも処理が高速です。ロード・ユーティリティはトリガーを起動したり、参照制約や表制約のチェックを実行したりしません (索引の一意性の妥当性チェックを除く)。

ロードのプロセスは、以下に示す 4 つの明確なフェーズ (段階) で構成されています (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。保管点または整合点は、LOAD コマンドの SAVECOUNT パラメーターによって指定されるインターバルで確立されます。保管点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、NULL 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整合点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場合、失敗したロード操作がロールバックされます。



図 1. ロード・プロセスの 4 つのフェーズ。ロード、構築、削除、および索引コピー: ロード操作の実行中は、ターゲット表はロード進行中状態にあります。表に制約がある場合には、この表もチェック・ペンディング状態になります。ALLOW READ ACCESS オプションが指定された場合には、この表も読み取りアクセス専用状態になります。

- 構築。このフェーズ中に索引が生成されます。

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生成されます。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。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場合、ロード操作が適切な点から自動的に再開されます。

- 削除。このフェーズ中に、ユニーク・キー違反や DATALINK 違反の発生した行が表から除かれます。ユニーク・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。また、リジェクトされた行に関するメッセージはメッセージ・フ

ファイルに書き込まれます。ロード・プロセスの完了後、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場合、ロード操作が適切な点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。

- 索引コピー。ここでは、SYSTEM TEMPORARY 表スペースから元の表スペースに索引データがコピーされます。これは、READ ACCESS オプションを指定したロード操作時に、索引作成に SYSTEM TEMPORARY 表スペースが指定されていた場合にのみ発生します。

注:

| ロード・ユーティリティーを呼び出した後で LIST UTILITIES コマンドを使って、ロード操作の進行状況をモニターすることができます。詳しくは、『LIST UTILITIES コマンド』を参照してください。

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。
- 入力ソースのフォーマット。このフォーマットは、DEL、ASC、PC/IXF、または CURSOR です。
- 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) **db2Load** 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、または相対列ロケーション。
- ユーティリティーが整合点を確立する頻度。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、最後の整合点から開始されます。
- データの挿入先となる表の列の名前。
- ロード操作の進行中に、表の中に事前に存在するデータを照会できるかどうか。

注: これは、READ ACCESS オプションを使用することにより実行できますが、ロード・ユーティリティーが REPLACE モードで呼び出される場合にはサポートされません。

- ロード操作で、他のユーティリティまたはアプリケーションが表の使用を終了するのを待機するようにするか、あるいは続行する前にその他のアプリケーションを強制的にオフにするか。
- 索引を作成する代替 SYSTEM TEMPORARY 表スペース。

注: これは、全索引の再構築で READ ACCESS オプションが指定された場合にのみサポートされます。

- LOB が格納されている入力ファイルのパスと名前。lobsinfile 修飾子を使用すると、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・ユーティリティに指示することができます。
- メッセージ・ファイル名。データのエクスポート、インポート、ロード、バインド、リストアなどの 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. メッセージ・ファイルの内容を表示できるのは、操作が終了したあとでのみです。
 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。
- 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・ユーティリティに指示するために使用します。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、使用可能なフリー・スペースの大きさをユーティリティが変更するかどうか。フリー・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。フリー・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場合があります。
 - 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 ユーティリティを呼び出します。ユニーク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た場合、それらのレコードの削除に合わせて統計情報が更新されることはありません。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 ユーティリティを呼び出すようにしてください。

- ロード操作中に統計を収集するかどうか。統計は、表に定義されているプロファイルに従って収集されます。ロード・コマンドの実行の前に、RUNSTATS コマンドを使ってこのプロファイルをあらかじめ作成しておく必要があります。プロフ

|
|

ファイルを作成していない場合に、プロファイルに従って統計を収集するようロード操作に指示すると、エラー・メッセージが戻されます。

- 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・リカバリーが可能になります。このオプションは、データベースの順方向ログ・リカバリー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・リカバリー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・ペンディング状態のままになります。

データベースを完全にリカバリーできるようにするには、ログを記録する必要があります。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ギング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。障害発生後にデータベースをリカバリー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り。

すでにデータが入っている表をロードしており、データベースがリカバリー可能でない場合は、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、エラーがあってもリカバリーできるようにしておいてください。

リカバリー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リカバリー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で、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 オプションを使用すると、あるロード・トランザクションをリカバリー不能としてマークし、それ以降にロールフォワード操作を実行してもリカバリー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、このような表はドロップ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・ペンディング状態になりません。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。

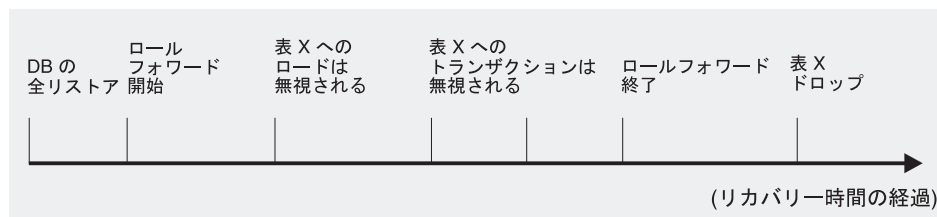


図 2. ロールフォワード操作時のリカバリー不能処理

- すべての索引修正をログ記録するかどうか。データベース構成パラメーター `logindexbuild` を設定し、しかも `COPY YES` リカバリー可能性オプションと `INCREMENTAL` 索引付けオプションを指定してロード操作を起動した場合、そのロードでは索引の修正がすべてログ記録されます。これらのオプションを使用する利点は、このロードのログ・レコードを先へ順にたどっていけば、索引のリカバリーも行われるという点にあります (これに対して、通常は、ロードで `REBUILD` 索引付けモードを使わないかぎり、索引はリカバリーされません)。
- 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ロケーションを指定する必要があります。

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 およびバージョン 7 で導入された構文の変更とロード動作の変更のサマリーを示します。

- 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整合点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・エンジンで使用される新規のソート・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。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、`TEMPORARY` 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、`TEMPORARY` 表スペースに関連したバッファ・プールにスピルされます。`TEMPORARY` 表スペースに関係付けるバッファ・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並列処理を実現するには、`TEMPORARY` 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を指定することをお勧めします。また、`TEMPORARY` 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をお勧めします。これによって、ディスク・リソースを使用していないときにそのリソースを保留することなく大量のデータの収容時にこのスペースを拡大させることができます。

- 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、ユーティリティはこれまでどおり REMOTE FILE パラメータを受け入れます)。これは、このパラメータ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 パラメータは、ファイルではなくディレクトリを参照します。
- 現在、ロード・ユーティリティは、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)、および AUTOSELECT (自動選択) モード (ランタイム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 オプションを使用して、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、表スペースをリストア・ペンディング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、表データはリストアされない ことに注意してください。

バージョン 7 のロード・ユーティリティは、リモート接続されたクライアントに置かれており、完全修飾ファイルまたは名前付きパイプ内にあるデータをロードすることができます。(lobsinfile ファイル・タイプ修飾子が指定された場合には、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。)

バージョン 8 で導入された以前のロード動作の変更

以下に、バージョン 8 で導入された構文の変更とロード動作の変更のサマリーを示します。

- バージョン 8 より前では、ロード時に、ロードされる表に属するオブジェクトの入った表スペースへの排他アクセスが必要でした。バージョン 8 では、ロードは表レベルで操作し、表スペースへの排他アクセスは必要なくなりました。ロードは、実行中のロード操作に関連する表オブジェクトにのみロックを置きます。同じ表スペースにある他の表オブジェクトへの並行アクセスが許可されています。

注: バージョン 8 より前は、リカバリー可能データベースに COPY NO オプションが指定された場合には、表スペースは、ロード操作がコミットされた後にのみバックアップ・ペンディング状態になっていました。バージョン 8 では、ロード操作が開始した時点で表スペースはバックアップ・ペンディング状態になり、ロード操作が失敗してロールバックされた後もこの状態を保ちます。以前のリリースと同様、COPY NO オプションが指定され、ロード操作が正常に実行されると、ロールフォワード・ユーティリティは、ロールフォワード操作時に従属表スペースをリストア・ペンディング状態にします。

- また、ユーザーが、ロード前に表に存在していたデータへの読み取りアクセス権を持つように指定することもできます。つまり、ロード操作の完了後は、表に制約があり、保全性チェックが完了していない場合には、新規データを表示することはできないという意味です。また、READ ACCESS および INDEXING MODE REBUILD オプションを指定することにより、ロード操作時に、索引が別個の表

スペースに再構築されるように指定することもできます。索引は、ロード操作の他のフェーズの後に実行される索引コピー・フェーズで、元の表スペースにコピーされます。

- **LOAD QUERY** コマンドの機能が拡張され、進行中のロード操作で以前に組み込まれた状況情報に加えて、データのロード先のターゲットの表の状態を戻すようになりました。ロード操作が表で進行中かどうかに関係なく、**LOAD QUERY** コマンドを使用して、表の状態を照会することもできます。
- **DMS** 表スペースでのエクステント割り振りがログに記録されるようになりました。**LOAD** コマンドは、**DMS** 表スペースに割り振る各エクステントごとに 2 つずつログ・レコードを書き込むようになります。また、**READ ACCESS** および **INDEXING MODE INCREMENTAL** オプションが指定される際には、データが増分的に索引に挿入される間に、ログ・レコードの一部が書き込まれます。
- ロード操作の前に、従属表スペースが静止されることはなくなりました。**COPY NO** オプションが指定されている場合には、新規の表スペース状態ロード進行中が使用されます。ロード進行中の表スペース状態では、ロード操作時に従属表のバックアップを行えません。すべてのロード操作がロード進行中の表状態を使用するという点で、ロード進行中の表スペース状態はロード進行中の表状態とは異なりますが、**COPY NO** オプションが指定されているロード操作でも、ロード進行中の表スペース状態を使用します。
- **ALLOW READ ACCESS** および **INDEXING MODE REBUILD** オプションを使用してロード操作を実行する際には、元の索引に加えて、索引の新しいコピーが作成されます。これは、索引表スペースのスペース所要量を 2 倍にする必要がありますことを意味します。これを避けるためには、**USE TABLESPACE** オプションを使用して、新規索引のストレージに **TEMPORARY** 表スペースを指定することができます。**TEMPORARY** 表スペースで新規索引が作成された後は、新規索引がターゲット表スペースにコピーされる前に、ターゲット表がオフラインになります。
- **LOAD** コマンドから表スペースを静止するための呼び出しは除去されました。ロード操作前に排他モードで表スペースを静止する場合には、静止される排他状態から表スペースを明示的に除去しなければなりません。以前のリリースでは、以下のコマンドの発行後、**LOAD** は、静止された表スペースをリセットし、これらが他のアプリケーションにアクセスできるようにしていました。

```
quiesce tablespaces for table t1 exclusive
load from data.del of del insert into t1
```

バージョン 8 では、以下のコマンドを発行して、静止された排他状態から表スペースを除去する必要があります。

```
quiesce tablespaces for table t1 reset
```

- **LOCK WITH FORCE** オプションが **LOAD** コマンドに追加されています。これにより、他のアプリケーションが表に対して持っているロックを強制的に解放させ、ロード操作が進行し、必要なロックを獲得できるようにすることができます。
- ロード・ユーティリティーには、新規の **CURSOR** ファイル・タイプを使用して、**SQL** ステートメントからロードする機能が追加されました。
- 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、以下の条件でサポートされるようになりました。

- クライアントが接続しているデータベースがパーティション・データベース環境である。
- クライアントが接続しているデータベースが、すでにカタログされているデータベースに対してカタログされている。
- マルチディメンション・クラスタリング (MDC) 表へのデータのロードがサポートされています。
- バージョン 8 より前では、ターゲット表中に生成列がある場合には、このターゲット表は、ロード操作の後もチェック・ペンディング状態のままでした。バージョン 8 では、ロード・ユーティリティにより列値が生成され、ロード操作の後に SET INTEGRITY ステートメントを発行する必要はなくなりました。
- パーティション・データベース環境に表をロードできるようになりました。これを実行するために、オートローダー・ユーティリティ (**db2atld**) を使用する必要はなくなりました。ロード API (**db2Load**) も拡張されて、パーティション・データベース・ロード・オプションをサポートするようになりました。

関連概念:

- 「データ・リカバリーと高可用性 ガイドおよびリファレンス」の『ロールフォワード・リカバリー』

関連資料:

- 「コマンド・リファレンス」の『RUNSTATS コマンド』
- 137 ページの『LOAD QUERY』
- 113 ページの『LOAD』
- 139 ページの『db2Load - ロード』
- 「コマンド・リファレンス」の『LIST UTILITIES コマンド』

並列処理とロード

ロード・ユーティリティは、複数のプロセッサや複数の記憶装置が使用されているハードウェア構成 (対称マルチプロセッサ (SMP) 環境など) を利用します。ロード・ユーティリティ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用する方法であり、ロード操作中に入出力の並列処理が可能になります (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の使用が関係しており、パーティション内の並列処理が可能になります (92 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。

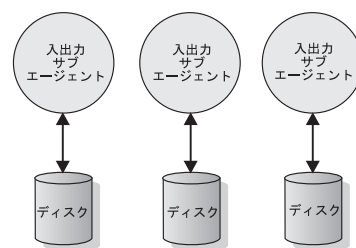


図 3. データ・ロード時に入出力の並列処理を利用する

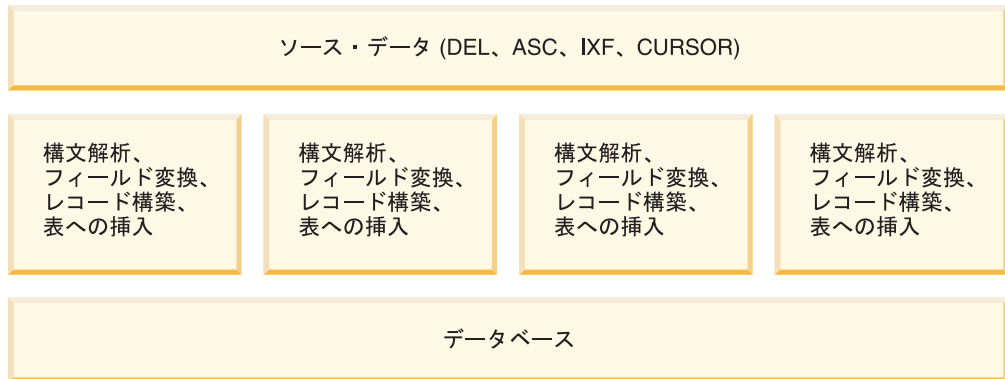


図4. データ・ロード時のパーティション内の並列処理の利用

関連概念:

- 189 ページの『ロードのパフォーマンスの最適化』

ロードの使用に必要な特権、権限、および許可

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。

Windows[®] NT、Windows 2000 および Windows .NET オペレーティング・システムで、DB2 が Windows サービスとして実行されている場合、ネットワーク・ドライブに常駐するファイルからデータをロードするには、これらのファイルに対する読み取りアクセスを持つユーザー・アカウントの下で実行するように DB2 サービスを構成する必要があります。

関連資料:

- 113 ページの『LOAD』

ロードの使用

前提条件:

ロード・ユーティリティを起動するには、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 (マルチディメンション・クラスター (MDC) 表を使用する場合を除く)、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。

クラスタリングする必要がある場合は、ロードする前にクラスタリング索引に従ってデータをソートしておいてください。MDC 表にデータをロードする際には、ロード操作前のソートは必要ありません。データは MDC 表定義に従ってクラスタ化されます。

制約事項:

ロード・ユーティリティには、以下の制約事項が適用されます。

- ニックネームへのデータのロードはサポートされていません。
- 型付き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。
- バックアップ・ペンディング状態にある表スペースで表の作成やドロップを実行しようすると失敗します。
- DB2 Connect、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で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・ユーティリティによって適用されません。

手順:

ロード・ユーティリティは、コマンド行プロセッサ (CLP)、コントロール・センターの「ロード (Load)」ノートブック、またはアプリケーション・プログラミング・インターフェース (API)、**db2Load** から起動できます。

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
```


この例の詳細は次のとおりです。

- 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。
- 4,000 ページ分のバッファ・スペースがロード操作中に使用されます。

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
```

この例の詳細は次のとおりです。

- 表データは置換されます。
- `TEMPFILES PATH` パラメーターを使用して、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。

「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詳細は、オンライン・ヘルプ機能によって提供されます。

ロード・ユーティリティーを呼び出した後で `LIST UTILITIES` コマンドを使って、ロード操作の進行状況をモニターすることができます。INSERT モード、REPLACE モード、または RESTART モードで実行するロード操作の場合、進行状況の詳細モニタリングのサポートを利用することができます。ユーティリティーが現在いるロードの段階に関する詳細情報を表示するには、`SHOW DETAILS` オプションを指定して `LIST UTILITIES` コマンドを出します。これに対して、TERMINATE モードで実行するロード操作では、詳細情報は得られません。LIST UTILITIES コマンドは、ロード終了ユーティリティーが現在稼働中であることを示すだけです。詳しくは、『LIST UTILITIES コマンド』を参照してください。

ロードでは、UNIQUE 制約以外のどんな制約も維持されません。つまり、表は、ロードの開始時点にチェック・ペンディング状態に置かれます。ロード操作が完了したら、`SET INTEGRITY` コマンドを使って、表のチェック・ペンディング状態を終了しなければなりません。

関連資料:

- 「コマンド・リファレンス」の『LIST UTILITIES コマンド』

- 「データ・リカバリーと高可用性 ガイドおよびリファレンス」の『Tivoli Storage Manager』
- 113 ページの『LOAD』
- 139 ページの『db2Load - ロード』

関連サンプル:

- 『tbmove.out -- HOW TO MOVE TABLE DATA (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.out -- HOW TO MOVE TABLE DATA (C++)』
- 『tbmove.sqC -- How to move table data (C++)』

読み取りアクセス・ロード操作

ロード・ユーティリティは、他のアプリケーションがロード中の表に対して行うアクセスの量を制御する 2 つのオプションを提供します。 **ALLOW NO ACCESS** オプションは表を排他的にロックし、表のロード時には、表データへのアクセスを許可しません。これがデフォルトの動作です。 **ALLOW READ ACCESS** オプションは、他のアプリケーションによる表へのすべての書き込みアクセスを行えないようにしますが、事前ロードされたデータへの読み取りアクセスは許可します。この項では、**ALLOW READ ACCESS** オプションを扱います。

ロード操作を開始する前に存在する表データおよび索引データは、ロード操作の進行中に照会で表示できます。次のような例を考察してみます。

1. 1 つの整数列のある表を作成します。

```
create table ED (ed int)
```

2. 3 つの行をロードします。

```
load from File1 of del insert into ED
...
Number of rows read           = 3
Number of rows skipped        = 0
Number of rows loaded         = 3
Number of rows rejected       = 0
Number of rows deleted        = 0
Number of rows committed     = 3
```

3. 表を照会します。

```
select * from ED

ED
-----
      1
      2
      3
```

3 行が選択されます。

4. **ALLOW READ ACCESS** オプションを指定してロード操作を実行し、さらに 2 つの行のデータをロードします。

```
load from File2 of del insert into ED allow read access
```

5. 同時に、他の接続ではロード操作の進行中に表を照会します。

```
select * from ED

ED
```



```

-----
1
2
3

```

3 行が選択されます。

6. ロード操作が終了するのを待ってから、表を照会します。

```
select * from ED
```

```

ED
-----
1
2
3
4
5

```

5 行が選択されます。

ALLOW READ ACCESS オプションは、ロード操作が進行中の場合やロード操作が失敗した後も、ユーザーがいつでも表データにアクセスできるようにするため、大量のデータをロードする際に大変便利です。 **ALLOW READ ACCESS** モードのロード操作の動作は、アプリケーションの分離レベルに依存しません。つまり、読取装置がどの分離レベルであっても、事前に存在するデータを常に読み取ることができますが、ロード操作が終了するまでは新しくロードされたデータを読み取ることはできません。

ロード操作の最後を除いては、その操作中ずっと読み取りアクセスが提供されます。データがコミットされる前に、ロード・ユーティリティは表に対する排他ロック (Z-lock) を獲得します。ロード・ユーティリティは、表に対してロックを持っているすべてのアプリケーションがこれらを解放するまで待機します。これは、データがコミットされる前の遅延の原因となることがあります。 **LOCK WITH FORCE** オプションを使用すると、競合するアプリケーションを強制的にオフにし、待機せずにロード操作を継続できるようにすることができます。

通常、**ALLOW READ ACCESS** モードのロード操作は、短期間の排他ロックを獲得します。しかし、**USE <tablespaceName>** オプションが指定されている場合には、索引コピー・フェーズの期間ずっと排他ロックが続きます。

注:

1. ロード操作が打ち切られると、このロード操作の発行時に指定されたものと同じアクセス・レベルで継続します。したがって、**ALLOW NO ACCESS** モードのロード操作が打ち切られると、ロードが終了するか、またはロード再始動が発行されるまで、表データはアクセス不可になります。 **ALLOW READ ACCESS** モードのロード操作が打ち切られても、事前ロードされた表データはこれまでどおり読み取りアクセスが可能です。
2. 打ち切られたロード操作に **ALLOW READ ACCESS** オプションが指定されていた場合には、ロード再始動操作またはロード終了操作にもこれを指定することができます。しかし、打ち切られたロード操作に **ALLOW NO ACCESS** オプションが指定されていた場合には、ロード再始動操作またはロード終了操作に **ALLOW READ ACCESS** オプションを指定することはできません。

以下の場合には、**ALLOW READ ACCESS** オプションはサポートされません。

- REPLACE オプションが指定されている場合。ロード置換操作では、新規データをロードする前に既存の表データを切り捨てるため、ロード操作の完了後まで照会できる既存のデータはありません。
- 索引には無効のマークが付き、再作成されるのを待機します。一部のロールフォワード・シナリオで、または **db2dart** コマンドの使用時に、索引に無効のマークが付く可能性があります。
- INDEXING MODE DEFERRED オプションが指定されている場合。このモードでは、索引に再作成が必要というマークが付けられます。
- ALLOW NO ACCESS ロード操作が再始動処理中または終了処理中の場合。これが完全にオンラインになるまで、表に対して ALLOW READ ACCESS モードのロード操作を実行することはできません。
- ロード操作は、チェック・ペンディング状態であり、読み取りアクセス状態ではない表で実行されます。これは、制約付きの表に対する複数のロード操作でも同じです。SET INTEGRITY ステートメントが発行されるまで、表がオンラインになることはありません。

一般に、表データがオフラインになっている場合には、この表がオンラインになるまでロード操作時に読み取りアクセスを使用することはできません。

関連概念:

- 103 ページの『保全性違反のチェック』
- 184 ページの『表ロック、表状態、および表スペース状態』
- 97 ページの『索引の作成』

索引の作成

索引は、ロード操作の構築フェーズで作成されます。LOAD コマンドで指定できる索引モードは 4 つあります。

1. REBUILD。すべての索引を再作成します。
2. INCREMENTAL。索引を新しいデータで拡張します。
3. AUTOSELECT。ロード・ユーティリティが REBUILD モードか INCREMENTAL モードかを自動的に決定します。これがデフォルトです。

注: REBUILD モードと INCREMENTAL モードの動作はかなり異なるため、索引モードを明示的に選択するように決定することもできます。

4. 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、最初にアクセスされるときに強制的に再作成されることがあります。このオプションは索引を保守することはなく、索引スキャナーには有効な索引が必要であるため、このオプションと ALLOW READ ACCESS オプションとの間に互換性はありません。

ALLOW READ ACCESS オプションを指定するロード操作では、選択した索引モードによっては、スペース使用量およびロギングに特に配慮する必要があります。ALLOW READ ACCESS オプションが指定されると、ロード・ユーティリティは、索引が再作成中であっても引き続きそれらを照会に使用できるようにします。

ALLOW READ ACCESS モードのロード操作で INDEXING MODE INCREMENTAL オプションを指定する際には、ロード・ユーティリティは、索引ツリーの整合性を保護するログ・レコードを作成します。書き込まれるログ・レコードの数は、挿入されるキーの数の一部であり、同様の SQL 挿入操作で必要とされる数よりずっと少ない数です。INDEXING MODE INCREMENTAL オプションを指定した ALLOW NO ACCESS モードのロード操作は、通常のスペース割り振りログのほかには、小さなログ・レコードしか作成しません。

ALLOW READ ACCESS モードのロード操作で INDEXING MODE REBUILD オプションを指定すると、元の索引と同じ表スペースか、または SYSTEM TEMPORARY 表スペースのいずれかで、新しい索引がシャドー として作成されます。元の索引は変更されずにロード操作で使うことができ、表は排他ロックされたままでロード操作の終わりに新規索引に置き換えられるだけです。ロード操作に失敗してトランザクションがロールバックされる場合でも、元の索引は変更されません。

元の表スペースと同じ表スペースでの新規索引の作成

デフォルトでは、シャドー 索引は、元の索引と同じ表スペースに作成されます。元の索引と新規索引の両方が同時に保守されるため、同時に両方の索引を保留できる十分な表スペースがなければなりません。ロード操作が打ち切られると、新規索引の作成に使用される余分のスペースが解放されます。ロード操作がコミットされると、元の索引に使用されるスペースが解放され、新規索引が現行の索引になります。元の索引と同じ表スペースに新規索引が作成されると、元の索引の置換がほとんど同時に行われます。

DMS 表スペースで索引が作成される場合、新規のシャドー 索引はユーザーには見えません。SMS 表スペースで索引が作成される場合、ユーザーは .IN1 接尾部および .INX 接尾部のある表スペース・ディレクトリで索引ファイルを見ることができます。これらの接尾部は、どれが元の索引で、どれがシャドー 索引であるかを示しません。

SYSTEM TEMPORARY 表スペースでの新規索引の作成

元の表スペースでスペースが不足しないようにするために、新規索引を SYSTEM TEMPORARY 表スペースに作成することができます。USE <tablespaceName> オプションを使用すると、INDEXING MODE REBUILD および ALLOW READ ACCESS オプションを使用する際に、SYSTEM TEMPORARY 表スペースで索引を再作成できます。システム TEMPORARY 表は SMS 表スペースまたは DMS 表スペースのどちらでもかまいませんが、SYSTEM TEMPORARY 表スペースのページ・サイズは、元の索引表スペースのページ・サイズに一致しなければなりません。

ロード操作が ALLOW READ ACCESS モードでない場合、または索引モードに互換性がない場合には、USE <tablespaceName> オプションは無視されます。USE <tablespaceName> オプションは、INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT オプションでのみサポートされます。INDEXING MODE AUTOSELECT オプションが指定されており、ロード・ユーティリティが索引の増分保守を選択する場合には、USE <tablespaceName> オプションは無視されます。

ロード再始動操作では、元のロード操作で代替表スペースを使用しなかった場合でも、代替表スペースを使用して索引を作成できます。元のロード操作が `ALLOW READ ACCESS` モードで発行されなかった場合には、`ALLOW READ ACCESS` モードでロード再始動操作を発行することはできません。ロード終了操作では索引を再作成しないため、`USE <tablespaceName>` オプションは無視されます。

ロード操作の構築フェーズでは、`SYSTEM TEMPORARY` 表スペースに索引が作成されます。その後、索引コピー・フェーズで、`SYSTEM TEMPORARY` 表スペースから元の索引表スペースに索引がコピーされます。元の索引表スペースに新規索引用の十分なスペースがあることを確認するには、構築フェーズで元の表スペースにスペースを割り振らなければなりません。したがって、ロード操作で索引スペースが不足しそうな場合には、構築フェーズでこれを実行します。この場合、元の索引が消失することはありません。

索引コピー・フェーズは、構築および削除フェーズの後に実行されます。索引コピー・フェーズが始まる前に、表が排他的にロックされます。つまり、索引コピー・フェーズ全体に渡って、読み取りアクセスは使用不可になります。索引コピー・フェーズは物理コピーであるため、表はかなりの期間使用できなくなります。

注: `SYSTEM TEMPORARY` 表スペースまたは索引表スペースのどちらかが `DMS` 表スペースである場合、`SYSTEM TEMPORARY` 表スペースの読み取りにより、`SYSTEM TEMPORARY` 表スペースでのランダム入出力が発生し、そのために遅延が生じる可能性があります。索引表スペースへの書き込みはこれまでどおり最適化され、`DISK_PARALLELISM` 値が使用されます。

関連概念:

- 84 ページの『ロードの概要』
- 95 ページの『読み取りアクセス・ロード操作』

ロードでの ID 列の使用

ロード・ユーティリティを使用して、`ID` 列の入った表にデータをロードすることができます。`ID` 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は次のような規則に従って動作します。

- `ID` 列が `GENERATED ALWAYS` の場合、入力ファイル内の対応する行の中に `ID` 列用の値がないか、または `NULL` 値が明示的に指定されているときに、表の行用の `ID` 値が生成されます。`ID` 列に非 `NULL` 値を指定すると、その行はリジェクトされます (SQL3550W)。
- `ID` 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・ユーティリティはその値を使用します。データが欠落しているかまたは明示的に `NULL` であれば、値が生成されます。

ロード・ユーティリティは、ユーザー提供の `ID` 値に関して、`ID` 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。

パーティション・データベースでは、表の `ID` 列がパーティション・キー内にある場合、またはパーティション・キーの一部を成す生成列内で `ID` 列が参照されてい

て、しかも `identityoverride` 修飾子の指定がない場合には、パーティションのすべてのロードが「ロード」段階から再開しないと、ロードの `RESTART` 操作は許可されません。このようなロードが許可されないのは、`ID` 列に対する依存性が原因で、ロードの再開時の行のハッシュが最初のロードでのハッシュと異なることがあるためです。その場合、通常はロードの `TERMINATE` オプションを使って、ロードを終了する必要があります。

`ID` 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 `ID` 列の値が入っていない (`NULL` ずらない) 場合に、`ID` 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、`ID` 列を持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` コマンドを使用して、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 `ID` 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 `ID` 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 `ID` 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` 修飾子を使用すると、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 ID** 列をもつ表にユーザー指定値をロードするために使用します。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、または **ROLLFORWARD DATABASE** コマンドで **DROPPED TABLE RECOVERY** オプションを使用してリカバリー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、**ID** 列でデータ (または **NULL** データ) の入っていない行はリジェクトされます (**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。

関連概念:

- 「管理ガイド: プランニング」の『ID 列』

ロードでの生成列の使用

ロード・ユーティリティを使用して、(**ID** 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。列値はこのユーティリティによって生成されます。

注: バージョン 7 以前のクライアントとバージョン 8 以降のサーバーとの間でロード操作を開始する場合、ロード・ユーティリティは、生成列がある表をチェック・ペンディング状態にします。

バージョン 7 以前のクライアントが生成列のある表にデータをロードするのに使用されたために、表がチェック・ペンディング状態になった場合には、以下のステートメントで、表のチェック・ペンディング状態を解除し、値の生成を強制します。

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

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・ユーティリティは次のような規則に従って動作します。

- データ・ファイルの対応する行に列の値が欠落している場合や **NULL** 値が提供されている場合には、生成列に値が作成されます。生成列に非 **NULL** 値を指定すると、その行はリジェクトされます (**SQL3550W**)。
- **NULL** 可能列でない生成列用に **NULL** 値が作成されると、データの行全体がリジェクトされます (**SQL0407N**)。これが起きるのは、たとえば、**NULL** 可能列でない生成列が 2 つの表の列の合計として定義されていて、それらの表の列にデータ・ファイルに **NULL** 値が組み込まれた場合です。

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 3 つのファイル・タイプ修飾子がロード・ユーティリティ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 (**NULL** 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たします。

```
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、生成列を持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 コマンドを使用して、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るとしても、そのデータを無視して、生成される各列に計算された値をロードします。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 NULL 値 10、11、12 (g1 の場合)、および 15、16、17 (g2 の場合) により、行はリジェクトされます (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 修飾子を使用すると、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ために使用します。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、または ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションを使用してリカバリー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、NULL 不能の生成列でデータ (または NULL データ) の入っていない行はリジェクトされます (SQL3116W)。

この修飾子が使用されるとき、表はロード操作の後にチェック・ペンディング状態におかれます。ユーザーが提供する値を検証せずに、表をチェック・ペンディング状態から解放するには、以下のコマンドを発行します。

```
SET INTEGRITY FOR table-name GENERATED COLUMN IMMEDIATE  
UNCHECKED
```

表をチェック・ペンディング状態から解放し、ユーザーが提供する値の検査を強制するには、以下のコマンドを発行します。

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED.
```

ロード・ユーティリティは、パーティション・キー内に生成列が存在する表のロードをサポートします。こうした生成列の場合、ロードされる各行のデータの最初の 32KB 以内に従属列のデータがなければなりません。

たとえば、次のような SQL ステートメントによって作成された表があるとして。

```
create table table1 (c1 int, c2 int, g1 int generated always as (c1 + c2))  
partitioning key (g1)
```

この表にデータを正常にロードするには、列 c1 および c2 のすべてのデータは、ロードされる各行の最初の 32KB 以内になければなりません。この制約事項を満たさない行はリジェクトされます。

注: 例値の生成がロードでサポートされないケースが 1 つあります。それは、生成された列式のうちの 1 つに、FENCED であるユーザー定義関数が入っている場合です。そのような表へのロードを試みると、ユーティリティは失敗します。ただし、ロードの generatedoverride ファイル・タイプ修飾子を使えば、その種の生成列に自分独自の値を指定することができます。

関連概念:

- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『生成列』

保安全性違反のチェック

以下のいずれかの状態が存在する場合は、ロード操作の後に、その表が READ または NO ACCESS モードでチェック・ペンディング状態になっていることがあります。

- 表に表チェック制約または参照保全制約が定義されている場合。
- この表に、データ・リンク列が定義されている場合。
- この表に生成列があり、バージョン 7 以前のクライアントを使用してロード操作が開始された場合。
- 表に下層即時マテリアライズ照会表またはこれを参照する下層即時ステージング表がある場合。
- 表がステージング表またはマテリアライズ照会表の場合。

ロードした表のチェック・ペンディング状態は、その表に対応する SYSCAT.TABLES 項目の STATUS フラグに示されます。ロードした表を完全に使用可能にするには、

STATUS の値として N を、また ACCESS MODE の値として F を指定します。これは表が完全にアクセス可能であり、通常状態であることを示します。

ロードされる表に下層表がある場合には、CHECK PENDING CASCADE パラメーターを指定して、ロードされる表のチェック・ペンディング状態が即時に下層表にカスケードされるようにするかどうかを指示できます。

ロードされる表に、下層外部キー表、従属マテリアライズ照会表、および従属ステージング表と共に制約があるときに、すべての表がロード操作前に通常状態である場合には、指定されるロード・パラメーターに基づいて、以下のような結果になります。

INSERT、ALLOW READ ACCESS、および CHECK PENDING CASCADE IMMEDIATE

ロードされる表、その従属マテリアライズ照会表、および従属ステージング表は、読み取りアクセスを持つチェック・ペンディング状態になります。

INSERT、ALLOW READ ACCESS、および CHECK PENDING CASCADE DEFERRED

ロードされる表だけが、読み取りアクセスを持つチェック・ペンディング状態になります。下層外部キー表、下層マテリアライズ照会表、および下層ステージング表は元の状態のままになります。

INSERT、ALLOW NO ACCESS、および CHECK PENDING CASCADE IMMEDIATE

ロードされる表、その従属マテリアライズ照会表、および従属ステージング表は、アクセスを持たないチェック・ペンディング状態になります。

INSERT または REPLACE、ALLOW NO ACCESS、および CHECK PENDING CASCADE DEFERRED

ロードされる表だけが、アクセスを持たないチェック・ペンディング状態になります。下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表は元の状態のままになります。

REPLACE、ALLOW NO ACCESS、および CHECK PENDING CASCADE IMMEDIATE

表およびそのすべての下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表は、アクセスを持たないチェック・ペンディング状態になります。

注: ロード置換操作で ALLOW READ ACCESS オプションを指定すると、エラーが発生します。

チェック・ペンディング状態を除去するには、SET INTEGRITY ステートメントを使用します。SET INTEGRITY ステートメントは表をチェックして制約違反がないかどうかを調べ、その表のチェック・ペンディング状態を終了します。すべてのロード操作が INSERT モードで実行される場合、SET INTEGRITY ステートメントを使用して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チェックして、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チェックするのは TABLE1 のうち追加部分だけです。追加部分だけをチェックして制約違反がないかどうかを調べることにより、表全体をチェックするよりも時間が短くて済みます。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。

CHECK PENDING CASCADE DEFERRED オプションを指定して表がロードされるときに、SET INTEGRITY ステートメントを使用して保全性違反をチェックする場合には、下層表はアクセスを持たないチェック・ペンディング状態になります。表をこの状態から解除する場合には、明示的要求を発行する必要があります。

従属マテリアライズ照会表または従属ステージング表を持つ表が INSERT オプションを使用してロードされるときに、SET INTEGRITY ステートメントを使用して保全性違反をチェックする場合には、表はチェック・ペンディング状態ではなくなり、NO DATA MOVEMENT モードに入れられます。これは、従属マテリアライズ照会表の後続の増分リフレッシュ、および従属ステージング表の増分伝搬を容易にするために実行されます。NO DATA MOVEMENT では、表内の行の移動の原因となるような操作は許可されません。

NO DATA MOVEMENT モードは、SET INTEGRITY ステートメントを発行する際に FULL ACCESS オプションを指定することによりオーバーライドできます。表は完全にアクセス可能になりますが、従属マテリアライズ照会表の完全再計算が後続の REFRESH TABLE ステートメントで実行され、従属ステージング表は不完全な状態になります。

ロード操作で ALLOW READ ACCESS オプションが指定されている場合には、SET INTEGRITY ステートメントを使用して制約違反がチェックされるまで、表は読み取りアクセス状態のままになります。ロード操作がコミットされたら、アプリケーションは表で、ロード操作前に存在したデータを照会できますが、SET INTEGRITY ステートメントが発行されるまでは、新しくロードされたデータを表示することはできません。

制約違反をチェックする前に、いくつかのロード操作を表で実行できます。ALLOW READ ACCESS モードですべてのロード操作が完了した場合には、最初のロード操作の前に表に存在していたデータだけが照会に使用できます。

表が 1 つでも複数でも、このステートメントを 1 回呼び出すだけでチェックできます。従属表を独自にチェックする場合、その親表がチェック・ペンディング状態になってはなりません。そうしないと、親表と従属表の両方を同時にチェックしなければならなくなります。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 1 回の SET INTEGRITY ステートメント呼び出しに組み込む必要があります。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チェックするのがよいかもしれません。ただしこれが可能なのは、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、ほとんどの場合 DB2® は増分的な処理を選択するため、このオプションは不要です。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。

- 表がチェック・ペンディング状態の間に、この表に新しく制約が追加された場合。
- 表に対する最後の保全性チェックの後で、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チェックされた場合。
- 表が、移行前のチェック・ペンディング状態にあった場合。移行後に最初に表が保全性チェックされるときは、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でロールフォワードされており、表とその親が異なる表スペースに常駐する場合。

表で SYSCAT.TABLES カタログの CONST_CHECKED 列に 1 つまたは複数の W 値があるときに、SET INTEGRITY ステートメントに NOT INCREMENTAL オプションが指定されていない場合、表は増分に処理され、SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

制約違反のある行に関する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。

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、表がチェック・ペンディング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。これについては考慮が必要です。1 つの選択肢は、INSERT トリガーをいったんドロップ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。

関連概念:

- 188 ページの『ロード操作後のペンディング状態』
- 95 ページの『読み取りアクセス・ロード操作』

関連資料:

- 「SQL リファレンス 第 2 巻」の『SET INTEGRITY ステートメント』

従属即時マテリアライズ照会表のリフレッシュ

即時リフレッシュ・マテリアライズ照会表の基本表が INSERT オプションを使用してロードされる場合には、REFRESH IMMEDIATE で定義される従属マテリアライズ照会表で SET INTEGRITY ステートメントを実行すると、マテリアライズ照会表で増分リフレッシュが実行されます。増分リフレッシュ時には、基本表にある追加行に対応する行が更新され、マテリアライズ照会表に挿入されます。基本表が大きく、追加データが少ない場合には、増分リフレッシュの速度はそれだけ速くなります。増分リフレッシュが許可されず、完全リフレッシュ (つまり、マテリアライズ照会表定義照会の再計算) が使用される場合もあります。

INCREMENTAL オプションを指定したにもかかわらず、マテリアライズ照会表の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。

- マテリアライズ照会表の基本表でロード置換操作が実行されるか、または基本表に対する最後の保全性チェックの後で NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化された場合。
- マテリアライズ照会表がロードされている場合 (REPLACE または INSERT モードのいずれかで)。
- 保全性チェック時に FULL ACCESS オプションを使用することによってマテリアライズ照会表がリフレッシュされる前に、基本表がチェック・ペンディング状態でなくなった場合。
- マテリアライズ照会表の基本表で、保全性のチェックが非増分的に実行された場合。
- マテリアライズ照会表が、移行前のチェック・ペンディング状態にあった場合。
- マテリアライズ照会表またはその基本表が入った表スペースがある時点でロールフォワードされており、マテリアライズ照会表とその基本表が異なる表スペースに常駐する場合。

マテリアライズ照会表で、SYSCAT.TABLES カタログの CONST_CHECKED 列に 1 つまたは複数の W 値がある場合で、SET INTEGRITY ステートメントに NOT INCREMENTAL オプションが指定されていない場合、表は増分にリフレッシュされ、SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

以下の例は、マテリアライズ照会表 AST1 の基本表 UT1 へのロード挿入操作を示します。UT1 でデータ保全性がチェックされ、データ移動なしモードになります。AST1 の増分リフレッシュが完了すると、UT1 は完全アクセス状態に戻ります。このシナリオでは、UT1 の保全性チェックと AST1 のリフレッシュの両方が増分的に処理されます。

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;  
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;  
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;  
REFRESH TABLE AST1;
```

関連概念:

- 103 ページの『保全性違反のチェック』

従属即時ステージング表の伝搬

ロードされる表が即時伝搬属性を持つステージング表の基本表であり、ロード操作が挿入モードで実行される場合には、従属即時ステージング表への後続の伝搬は増分になります。

増分伝搬時には、基本表にある追加行に対応する行がステージング表に追加されます。基本表が大きく、追加データが少ない場合には、増分伝搬の速度はそれだけ速くなります。また、ステージング表を使用して、その従属据え置きマテリアライズ照会表をリフレッシュすると、パフォーマンスが改善されます。増分伝搬が許可さ

れず、ステージング表に不完全というマークが付けられる場合もあります。つまり、CONST_CHECKED 列のステージング・バイトの値は F になります。この状態では、ステージング表を使用して、従属据え置きマテリアライズ照会表をリフレッシュすることはできず、マテリアライズ照会表の保守プロセスで完全リフレッシュが必要になります。

表が不完全状態で、INCREMENTAL オプションが指定されているにもかかわらず、表の増分伝搬が実行できない場合、エラーが戻されます。以下のいずれかが発生すると、システムは即時データ伝搬をオフにし、表状態を不完全に設定します。

- ステージング表の基本表でロード置換操作が実行されるか、または基本表に対する最後の保全性チェックの後で NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化された場合。
- ステージング表の従属マテリアライズ照会表、またはステージング表が REPLACE または INSERT モードでロードされた場合。
- 保全性チェック時に FULL ACCESS オプションを使用することによってステージング表が伝搬される前に、基本表がチェック・ペンディング状態でなくなった場合。
- ステージング表の基本表で、保全性のチェックが非増分的に実行された場合。
- ステージング表またはその基本表が入った表スペースがある時点でロールフォワードされており、ステージング表とその基本表が異なる表スペースに常駐する場合。

ステージング表で、SYSCAT.TABLES カタログの CONST_CHECKED 列に W 値があり、NOT INCREMENTAL オプションが指定されていない場合、ステージング表への増分伝搬が実行され、SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

以下の例は、ステージング表 G1 の基本表 UT1 およびその従属据え置きマテリアライズ照会表 AST1 へのロード挿入操作を示します。このシナリオでは、UT1 の保全性チェックと AST1 のリフレッシュの両方が増分的に処理されます。

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;  
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;  
SET INTEGRITY FOR UT1,G1 IMMEDIATE CHECKED;
```

```
REFRESH TABLE AST1 INCREMENTAL;
```

関連概念:

- 103 ページの『保全性違反のチェック』

マルチディメンション・クラスタリングの考慮事項

以下の制約事項はマルチディメンション・クラスタリング (MDC) 表に適用されます。

- LOAD コマンドの SAVECOUNT オプションはサポートされていません。
- これらの表は独自のフリー・スペースを管理するため、TOTALFREESPACE ファイル・タイプ修飾子はサポートされていません。

- MDC 表には ANYORDER 修飾子が必要です。ANYORDER 修飾子を使わないで MDC 表へのロードを実行すると、この表はユーティリティによって明示的に使用可能になります。

MDC 付きの LOAD コマンドを使用する場合には、ユニーク制約の違反は以下のよう
に処理されます。

- ロード操作の前に表にユニーク・キーが組み込まれ、重複レコードが表にロード
される場合には、元のレコードは残り、新規レコードが削除フェーズで削除され
ます。
- ロード操作の前に表にユニーク・キーが組み込まれず、ユニーク・キーと重複レ
コードの両方が表にロードされる場合には、ユニーク・キーのあるレコードが 1
つだけロードされ、その他のレコードは削除フェーズで除去されます。

注: どのレコードがロードされて、どのレコードが削除されるかを判別するた
めの明示的な技法はありません。

パフォーマンスの考慮

MDC 表のロード時のロード・ユーティリティのパフォーマンスを改善するには、
UTIL_HEAP_SZ データベース構成パラメーター値を大きくしなければなりません。
ユーティリティで利用できるメモリーを増やすと、mdc-load アルゴリズムのパフ
ォーマンスが大きく向上します。こうすると、ロード・フェーズで実行されるデー
タのクラスタリング時に、ディスクの入出力を減らすことができます。LOAD コマ
ンドの DATA BUFFER オプションが指定される際には、この値も大きくしなけれ
ばなりません。LOAD コマンドを使用して複数の MDC 表を同時にロードする場
合には、それに応じて、UTIL_HEAP_SZ 構成パラメーターの値を大きくしなければ
なりません。

すべての MDC 表にはブロック索引があるため、MDC ロード操作には常に構築フ
ェーズがあります。

ロード・フェーズでは、ブロック・マップの保守のために余分のログギングが実行さ
れます。割り振られるエクステントごとに、おおよそ 2 つの余分のログ・レコード
があります。パフォーマンスを良くするためには、このことを考慮に入れた値に
LOGBUFSZ データベース構成パラメーターを設定する必要があります。

MDC 表にデータをロードするために、索引付きのシステム一時表が使用されます。
表のサイズはロードされる個々のセルの数に比例します。表にあるそれぞれの行の
サイズは MDC ディメンション・キーのサイズに比例します。ロード操作時にこの
表の操作によるディスク入出力を最小限に抑えるには、TEMPORARY 表スペース
のバッファ・プールの大きさが十分であることを確認してください。

関連概念:

- 189 ページの『ロードのパフォーマンスの最適化』
- 「管理ガイド: プランニング」の『多次元クラスター化索引』

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・ユーティリティを開始できない場合、ロード・ユーティリティは表を通常の状態にしたまま終了します。

データのロード中に障害が発生した場合は、最後の整合点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が見つけれられるようにしてください。マルチディメンション・クラスタリング (MDC) 表では SAVECOUNT パラメーターがサポートされていないため、ロードの再始動は、ロード、構築、または削除フェーズの開始時にのみ実行されます。

注: ALLOW READ ACCESS オプションを指定したロード操作は、ALLOW READ ACCESS オプションまたは ALLOW NO ACCESS オプションのいずれかを使用して再始動できます。逆に、ALLOW NO ACCESS オプションを指定したロード操作は、ALLOW READ ACCESS オプションを使用して再始動することはできません。

読み取りアクセス許可ロード操作の再始動または終了

ALLOW READ ACCESS オプションを指定したロード操作が打ち切られた場合にも、ALLOW READ ACCESS オプションを使用してその操作を再始動または終了することができます。これにより、終了または再始動操作の進行中に、他のアプリケーションは表データを照会することができます。ALLOW READ ACCESS モードのロード操作と同様に、表はデータがコミットされるまで排他的にロックされます。

索引オブジェクトが使用できない場合や、無効のマークが付いている場合には、ALLOW READ ACCESS モードでロード再始動または終了操作を実行することは許可されません。

索引コピー・フェーズで元のロード操作が打ち切られた場合、索引が壊れている可能性があるために、ALLOW READ ACCESS モードの再始動操作は許可されません。

ロード・フェーズで ALLOW READ ACCESS モードのロード操作が打ち切られた場合、ロード・フェーズで再始動します。ロード・フェーズ以外のフェーズで打ち切られた場合には、構築フェーズで再始動します。元のロード操作が ALLOW NO ACCESS モードだった場合、元のロード操作がそのフェーズに到達したときに索引が有効であれば、削除フェーズで再始動操作が行われる可能性があります。索引に無効のマークが付いている場合には、ロード・ユーティリティは構築フェーズからロード操作を再始動します。

注: INDEXING MODE INCREMENTAL オプションが指定されている場合でも、すべてのロード再始動操作は REBUILD 索引モードを選択します。

LOAD TERMINATE コマンドを発行すると、通常、打ち切られたロード操作が最小限の遅延でロールバックされます。ただし、ALLOW READ ACCESS および

INDEXING MODE INCREMENTAL が指定されているロード操作で LOAD TERMINATE コマンドを発行すると、ロード・ユーティリティーが索引をスキャンし、矛盾があればそれを修正する際に、遅延が生じる可能性があります。この遅延の長さは、索引のサイズによって異なり、ロード終了操作に ALLOW READ ACCESS オプションが指定されているかどうかにかかわらず発生します。構築フェーズの前に元のロード操作が失敗した場合には、この遅延は発生しません。

注: 索引での矛盾を修正することから発生する遅延は、索引に無効というマークを付けて、これらを再構築することによって発生する遅延よりもはるかに小さくなります。

ロード再始動操作は、ロード再始動不能表状態の表では実行できません。ロールフォワード操作時には、この表はロード再始動不能表状態になる可能性があります。これはロード操作の終了前のある時点でロールフォワードを実行する場合、あるいは打ち切られたロード操作を介してロールフォワードを実行するものの、ロード終了操作またはロード再始動操作の終わりにロールフォワードしない場合に発生します。

関連概念:

- 184 ページの『表ロック、表状態、および表スペース状態』
- 211 ページの『パーティション・データベース環境でのロード操作の再始動または終了』

関連資料:

- 213 ページの『パーティション・データベース・ロード構成オプション』

ロード・コピー・ロケーション・ファイルを使ったデータのリカバリー

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・リカバリー中にロード・コピーの位置情報として使用されます。その中には次の情報があります。

- 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・コピーのロケーション
- 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用されます。

ファイル内の情報は、ロールフォワード・リカバリーの実行前に上書きされることがあります。

注:

1. パーティション・データベース環境では、**db2set** コマンドを使用して、すべてのデータベース・パーティション・サーバーに DB2LOADREC レジストリー変数を設定する必要があります。

2. パーティション・データベース環境では、それぞれのデータベース・パーティション・サーバーにロード・コピー・ファイルが存在しなければならず、ファイル名 (パスも含む) は同じでなければなりません。
3. DB2LOADREC レジストリー変数によって識別されるファイルの項目が有効ではない場合、無効な項目を置き換える情報を提供するために古いロード・コピー・ロケーション・ファイルが使用されます。

ロケーション・ファイルには、以下の情報が提供されます。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用されます。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。たとえば、

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 TSM 0 for other vendors
LOCATIONnumber	3	* Number of locations
ENTRY	/u/toronto/dbt.payroll.employees.001	
ENT	/u/toronto/dbt.payroll.employees.002	
ENT	/dev/rmt0	
TIM	19950725192054	
SCH	PAYROLL	
TAB	DEPT	
DAT	DBT	
DB2®	toronto	
BUF	NULL	
SES	NULL	
TYP	A	
TIM	19940325192054	
SCH	PAYROLL	
TAB	DEPT	
DAT	DBT	
DB2	toronto	
BUF	NULL	
SES	NULL	
TYP	0	
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 NULL にすることができます。SES が NULL の場合には、*numloadrecses* 構成パラメーターによって指定される値が使用されます。BUF が NULL の場合には、デフォルト値は SES+2 です。
4. ロケーション・ファイルにある項目が 1 つでも無効な場合には、以前のロード・コピー・ロケーション・ファイルの値が使用されます。
5. メディアの種類には、ローカル装置 (テープ、ディスク、またはディスクセットの場合は L)、TSM (A)、あるいは他のベンダー (0) のいずれかを指定できます。

種類が 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 0 の場合は、共有ライブラリー名を指定する必要があります。

6. SHRLib パラメーターは、ロード・コピー・データを格納する機能を持つライブラリーを指します。
7. COPY NO または NONRECOVERABLE オプションを指定してロード操作を呼び出し、かつ操作の完了後にデータベースまたは関連する表スペースのバックアップ・コピーを取らない場合、ロード操作の後の時点でこのデータベースまたは表スペースをリストアすることはできません。つまり、ロールフォワード・リカバリー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リストアできるのは、ロード操作より前の時点の状態だけです。

特定のロード・コピーを使用する場合には、データベースのリカバリー履歴ファイルを使用して、特定のロード操作のタイム・スタンプを判別できます。パーティション・データベース環境では、リカバリー履歴ファイルは各データベース・パーティションにローカルに存在します。

関連資料:

- 「データ・リカバリーと高可用性 ガイドおよびリファレンス」の『Tivoli Storage Manager』

LOAD

データを DB2 表にロードします。サーバー上に存在するデータは、ファイル、テープ、または名前付きパイプの形式にすることができます。リモートに接続されたクライアント上に存在するデータは、完全修飾ファイルまたは Named Pipe の形式にすることができます。ユーザー定義カーソルからデータをロードすることも可能です。

制約事項:

- 4 ロード・ユーティリティでは、階層レベルのデータのロードはサポートされてい
- 4 ません。ロード・ユーティリティには、範囲クラスター表との互換性はありません。
- 4

有効範囲:

このコマンドは、一度の要求で複数のデータベース・パーティションに対して発行できます。

許可:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*
- データベースに対するロード権限と以下のもの

LOAD

- ロード・ユーティリティーが INSERT モード、 TERMINATE モード (それまでのロード挿入操作を終了する)、または RESTART モード (以前のロード挿入操作を再開する) で呼び出された場合には、その表に対する INSERT 特権。
- ロード・ユーティリティーが REPLACE モード、 TERMINATE モード (それまでのロード置換操作を終了する)、または RESTART モード (以前のロード置換操作を再開する) で呼び出された場合には、その表に対する INSERT および DELETE 特権。
- 例外表がロード操作の一部として使用される場合、その例外表に対する INSERT 特権。

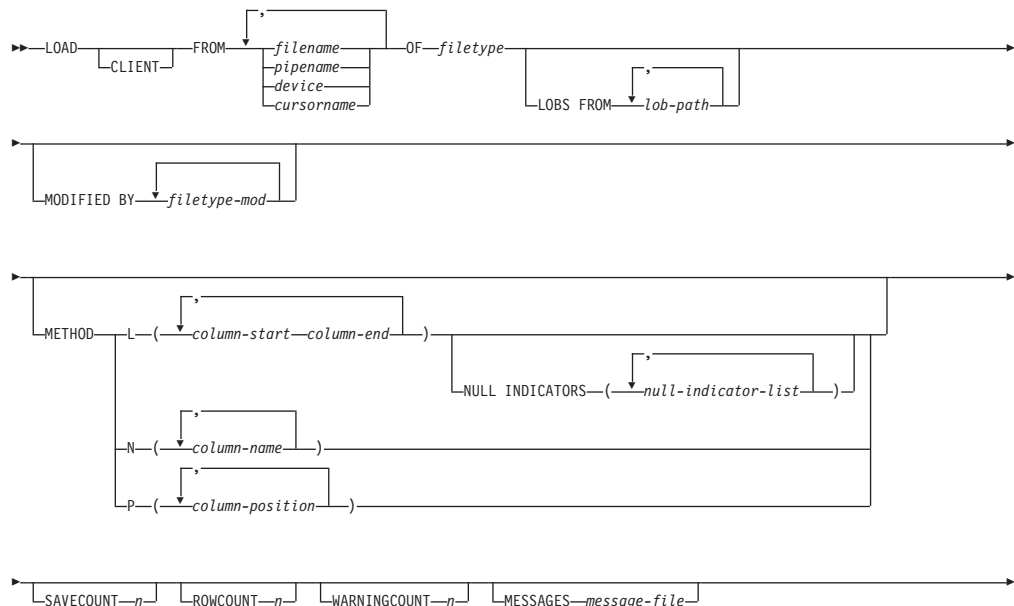
すべてのロード・プロセス (および一般にすべての DB2 サーバー・プロセス) はインスタンス所有者によって所有されており、それらのプロセスすべてにおいて、必要なファイルにアクセスするためにそのインスタンス所有者の ID を使用するため、インスタンス所有者には入力データ・ファイルに対する読み取りアクセス権が必要です。このコマンドをだれが呼び出すかには関係なく、それらの入力データ・ファイルをインスタンス所有者から読むことができればなりません。

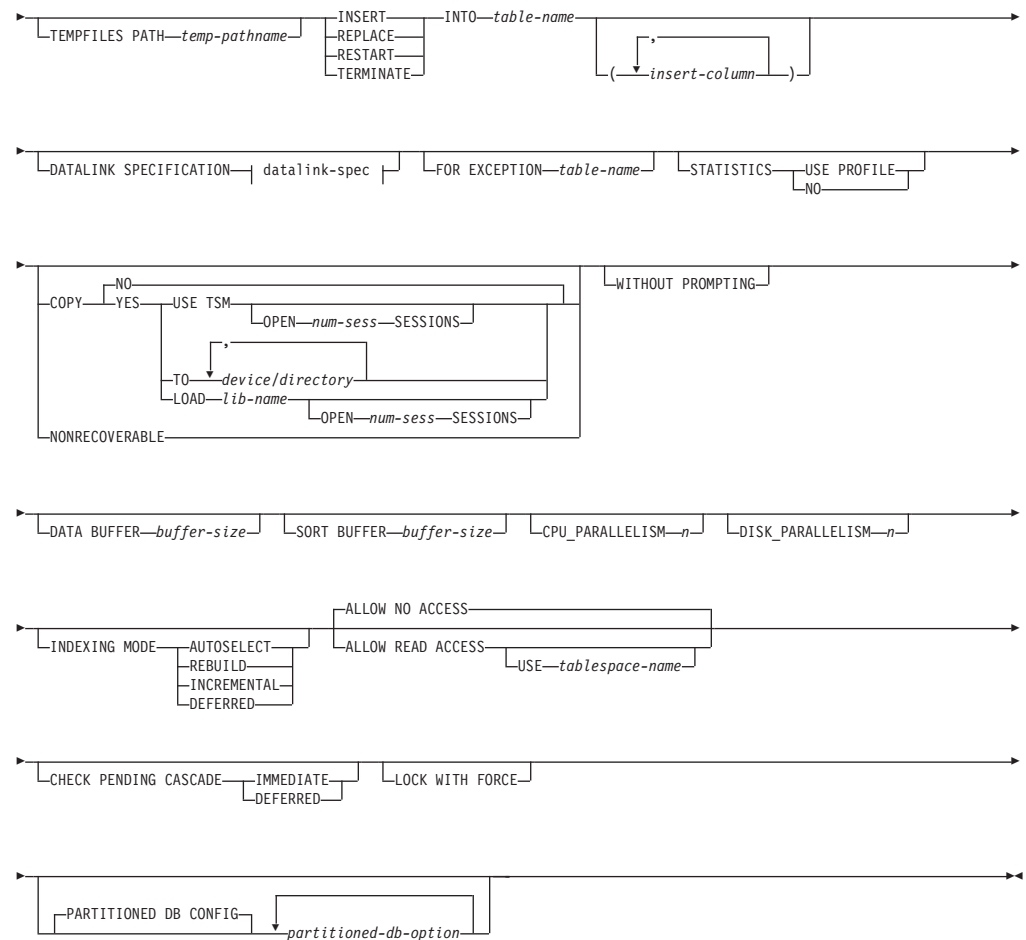
必要な接続:

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

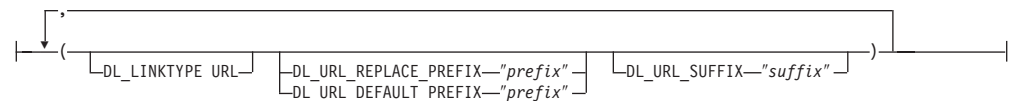
インスタンス。明示的なアタッチは必要ありません。データベースへの接続が確立している場合には、ローカル・インスタンスへの暗黙的なアタッチが試行されます。

コマンド構文:





datalink-spec:



コマンド・パラメーター:

ALLOW NO ACCESS

ロードを使用すると、ロード中に、排他的アクセスのターゲット表がロックされます。ロード中、表の状態は **LOAD IN PROGRESS** に設定されます。**ALLOW NO ACCESS** はデフォルトの動作です。これは、**LOAD REPLACE** で唯一有効なオプションです。

表に制約があると、表の状態は、**LOAD IN PROGRESS** の他に、**CHECK PENDING** に設定されます。表の **CHECK PENDING** を解除するには、**SET INTEGRITY** ステートメントを使用する必要があります。

ALLOW READ ACCESS

ロードを使用すると、ターゲット表は共用モードでロックされます。表の状態は、**LOAD IN PROGRESS** および **READ ACCESS** の両方に設定されます。表のロード中、データの非デルタ部分にアクセスすることができます。つまり、表を読み取る側はロードの開始前に存在していたデータにはアクセスができ、ロード中のデータはロードが完了するまで利用できない、という

ことです。ALLOW READ ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できますが、ALLOW NO ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できません。また、ターゲット表上の索引が要再作成のマークが付けられると、このオプションは無効になります。

表に制約があると、表の状態は、LOAD IN PROGRESS、READ ACCESS の他に、CHECK PENDING に設定されます。ロードの終了時に、表の状態 LOAD IN PROGRESS は解除されますが、CHECK PENDING と READ ACCESS はそのまま残ります。表の CHECK PENDING を解除するには、SET INTEGRITY ステートメントを使用する必要があります。表が CHECK PENDING および READ ACCESS の状態にある間、データの非デルタ部分には引き続き読み取りアクセスできますが、データの新しい (デルタ) 部分には、SET INTEGRITY ステートメントが完了するまでアクセス不能のままになります。ユーザーは、SET INTEGRITY ステートメントを発行しないで、同じ表上で複数のロードを実行できます。ただし、元の (チェック済み) データは、SET INTEGRITY ステートメントが発行されるまで可視のままです。

ALLOW READ ACCESS は、以下の修飾子もサポートします。

USE tablespace-name

索引が再作成される場合、表スペース *tablespace-name* に索引のシャドー・コピーが作成され、INDEX COPY PHASE のロード終了時に、元の表スペース上にコピーされます。このオプションと一緒に使用できるのは、システム TEMPORARY 表スペースだけです。これを指定しないと、索引オブジェクトと同じ表スペースにシャドー索引が作成されます。索引オブジェクトと同じ表スペースにシャドー・コピーが作成される場合、シャドー索引オブジェクトは瞬間的に古い索引オブジェクトの上にコピーされます。シャドー・コピーが索引オブジェクトと別の表スペースにある場合、物理コピーが実行されます。これにはかなりの入出力および時間を要します。コピーは、表がオフラインの間、INDEX COPY PHASE のロード終了時に行われます。

このオプションを指定しない場合は、シャドー索引は元の索引と同じ表スペースに作成されます。デフォルトでは、元の索引とシャドー索引の両方が同時に同じ表スペースに常駐するため、1 つの表スペース内に両方の索引を保留するためのスペースが不足する場合があります。このオプションを使用すれば、索引用の十分な表スペースを確保できます。

ユーザーが INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT を指定しない場合、このオプションは無視されます。このオプションは INDEXING MODE AUTOSELECT が選択され、ロードが索引を徐々に更新することを選択した場合にも無視されます。

CHECK PENDING CASCADE

LOAD によって表がチェック・ペンディング状態になる場合、CHECK PENDING CASCADE オプションを使用することによってユーザーはロード

される表を即時にすべての下層（下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表を含む）にカスケードするかどうか指定することができます。

IMMEDIATE

外部キー制約のチェック・ペンディング状態（読み取りまたは非アクセス・モード）が即時にすべての下層外部キー表に拡張されることを示します。表に下層即時マテリアライズ照会表または下層即時ステージング表がある場合、チェック・ペンディング状態は即時にマテリアライズ照会表およびステージング表に拡張されます。

LOAD INSERT 操作の場合、IMMEDIATE オプションが指定されている場合でも、チェック・ペンディング状態は下層外部キー表に拡張されないことに注意してください。

後で (SET INTEGRITY ステートメントの IMMEDIATE CHECKED オプションを使用して) ロードされる表の制約違反をチェックする際、チェック・ペンディング読み取り状態だった下層外部キー表は、チェック・ペンディング非アクセス状態になります。

DEFERRED

ロードされる表だけがチェック・ペンディング状態（読み取りまたは非アクセス・モード）になることを示します。下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表は、未変更のままになります。

下層外部キー表は、(SET INTEGRITY コマンドの IMMEDIATE CHECKED オプションを使用して) その親表の制約違反がチェックされるとき、後で暗黙的にチェック・ペンディング非アクセス状態になる場合があります。下層即時マテリアライズ照会表および下層即時ステージング表は、その基礎表のいずれかの保全性違反がチェックされる際、暗黙的にチェック・ペンディング非アクセス状態になります。従属表がチェック・ペンディング状態になったことを示す警告 (SQLSTATE 01586) が出されます。この従属表がいつチェック・ペンディング状態になるかについては、「SQL リファレンス」にある SET INTEGRITY ステートメントの「注」の項を参照してください。

CHECK PENDING CASCADE オプションが指定されない場合、次のようになります。

- ロードされる表だけが、チェック・ペンディング状態になります。下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表は、未変更のままになり、後にロードされた表の制約違反がチェックされる際に、暗黙的にチェック・ペンディング状態になる場合があります。

LOAD によってターゲット表がチェック・ペンディング状態にならない場合、CHECK PENDING CASCADE オプションは無視されます。

CLIENT

ロードするデータが、リモートに接続するクライアント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ない場合、このオプションは無視されます。ファイル・タイプが CURSOR の場合、このオプションはサポートされていません。

注:

1. `dumpfile` および `lobsinfile` 修飾子は、`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。
2. コード・ページ変換は、リモートの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コード・ページとは異なる場合、`codepage` 修飾子を使用してデータのコード・ページを指定する必要があります。

以下の例では、リモートに接続されたクライアント上に存在するデータ・ファイル (`/u/user/data.del`) は、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

順方向リカバリーが使用可能 (つまり、`logretain` または `userexit` がオン) になっていれば、表が存在している表スペースをバックアップ・ペンディング状態にするよう指定します。COPY NO オプションを使用する場合も、表スペース状態は `LOAD IN PROGRESS` になります。これは、一時的な状態であり、ロードが完了するか打ち切られると解除されます。表スペースのバックアップまたはデータベースの完全バックアップを実行しない限り、表スペースのどの表のデータも更新または削除できません。ただし、SELECT ステートメントを使用すれば、どの表のデータにもアクセス可能です。

リカバリー可能データベースでの COPY NO を指定した LOAD は、表スペースをバックアップ・ペンディング状態のままにします。たとえば、COPY NO を指定した LOAD および INDEXING MODE DEFERRED を実行すると、索引はリフレッシュが必要な状態になります。表での照会には、索引スキャンが必要なものがあり、索引がリフレッシュされるまで、成功しません。バックアップ・ペンディング状態にある表スペース内に常駐する場合、索引はリフレッシュできません。この場合、表へのアクセスは、バックアップが行われるまで許可されません。

注: 索引リフレッシュは、索引が照会によってアクセスされたときに、自動的に行われます。

COPY YES

ロードするデータのコピーを保存することを指定します。順方向リカバリーが使用禁止 (つまり `logretain` と `userexit` が両方ともオフ) であれば、このオプションは無効です。このオプションは `DATALINK` 列をもつ表ではサポートされません。

USE TSM

Tivoli Storage Manager (TSM) を使ってコピーを保管することを指定します。

OPEN num-sess SESSIONS

TSM またはベンダー製品とともに使用する入出力セッションの数です。デフォルト値は 1 です。

TO device/directory

コピー・イメージを作成する先の装置またはディレクトリーを指定します。

LOAD lib-name

使用するバックアップおよびリストア I/O 関数の入った共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。絶対パスで指定することができます。絶対パスを指定しない場合、デフォルトでユーザー出口プログラムの存在するパスになります。

CPU_PARALLELISM n

表オブジェクトの作成時に、レコードの解析、変換、およびフォーマット設定のためにロード・ユーティリティーが spawn 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、パーティション内並列処理を活用するために設計されています。これは、事前にソートされたデータをロードする際に役立ちます (ソース・データのレコード順序が保持されるため)。このパラメーターの値が 0 の場合や、このパラメーターを指定しなかった場合、ロード・ユーティリティーは、ランタイムにインテリジェントなデフォルト (通常は使用可能な CPU の数に基づく) を使用します。

注:

1. LOB または LONG VARCHAR フィールドのどちらかの入った表でこのパラメーターを使用する場合、システムの CPU の数またはユーザーが指定した値には関係なく、値は 1 になります。
2. SAVECOUNT パラメーターに指定する値が小さいと、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、ローダーがさらに多くの入出力操作を実行することになります。CPU_PARALLELISM が 1 より大きい場合、フラッシュ操作は非同期になり、ローダーは CPU を活用できません。CPU_PARALLELISM が 1 に設定されている場合、ローダーは整合点において IO を待ちます。CPU_PARALLELISM を 2 に設定し、SAVECOUNT を 10 000 に設定したロード操作は、CPU が 1 つしかなくても、同じ操作で CPU_PARALLELISM を 1 に設定した場合より速く完了します。

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・スペースとして使用する 4KB ページ数を設定します (並列処理の度合いには依存しません)。指定する値がアルゴリズム上の最小値より小さい場合、最小限必要なリソースが使用され、警告は戻されません。

このメモリーは、ユーティリティー・ヒープから直接に割り当てられ、そのサイズは *util_heap_sz* データベース構成パラメーターで修正可能です。

値を指定しないと、ランタイムにユーティリティーによって適切なデフォルトが計算されます。デフォルトは、表の特性だけでなく、ローダーのインスタンス生成時にユーティリティー・ヒープ中で使用可能なフリー・スペースの割合に基づいています。

DATALINK SPECIFICATION

各 DATALINK 列ごとに、それぞれ 1 つの列指定を括弧で囲んで指定できます。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。接頭部指定は、DL_URL_REPLACE_PREFIX または DL_URL_DEFAULT_PREFIX のどちらかになります。

DATALINK 列指定の数は、表で定義されている DATALINK 列の数と同じだけ指定できます。指定の順序は、挿入列 リストの中での DATALINK 列の順序、または表定義内での順序 (挿入列 リストが指定されていない場合) に従います。

DISK_PARALLELISM *n*

表スペース・コンテナにデータを書き込むためにロード・ユーティリティーが生成するプロセスまたはスレッドの数を指定します。値を指定しない場合、ユーティリティーは表スペース・コンテナの数と表の特性に基づいて、適切なデフォルトを選択します。

DL_LINKTYPE

指定した場合は、列定義の LINKTYPE に一致していなければなりません。そうすることによって、列定義に LINKTYPE URL が指定されている場合に DL_LINKTYPE URL が受け入れ可能になります。

DL_URL_DEFAULT_PREFIX "*prefix*"

これを指定すると、同じ列内のすべての DATALINK 値のデフォルト接頭部になります。ここでいう接頭部とは、URL 指定の「スキーム・ホスト・ポート」部分のことです。

接頭部の例

```
"http://server"
"file://server"
"file:"
"http://server:80"
```

列データの中に接頭部がない場合、DL_URL_DEFAULT_PREFIX でデフォルトの接頭部が指定されている場合、列の値の接頭部としてそのデフォルト接頭部が付けられます (NULL でない場合)。

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

- 列入力値 "/x/y/z" は "http://toronto/x/y/z" として保管されます。
- 列入力値 "http://coyote/a/b/c" は "http://coyote/a/b/c" として保管されます。
- 列入力値 NULL は NULL として保管されます。

DL_URL_REPLACE_PREFIX "*prefix*"

この文節は、それ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をロードまたはインポートする際に、ユーザーがデータに入っているホスト名を別のホスト名に一括置換したい場合に便利です。指定する場合には、それがすべての 非 NULL 列値の接頭部になります。列値にすでに接頭部がある場合、それは置き換えられます。列値に接頭部がない場合、DL_URL_REPLACE_PREFIX で指定される接頭部がその列値の接頭部になります。

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

- 列入力値 "/x/y/z" は "http://toronto/x/y/z" として保管されます。
- 列入力値 "http://coyote/a/b/c" は "http://toronto/a/b/c" として保管されます。"coyote" は "toronto" に置き換えられます。

- 列入力値 NULL は NULL として保管されます。

DL_URL_SUFFIX "suffix"

これを指定すると、それはその列のすべての非 NULL 列値に付加されます。これは実際には、DATALINK 値のデータ・ロケーション部分の「パス」コンポーネントに付加されます。

FOR EXCEPTION table-name

エラーが発生した行のコピー先となる例外表を指定します。ユニーク索引または主キー索引に違反した行がすべてコピーされます。DATALINK 例外も例外表にキャプチャーされます。非修飾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。パーティション・データベース環境では、ロードする表を定義されたパーティションの例外表を定義する必要があります。一方ダンプ・ファイルには、無効であるか構文エラーであるためにロードできない行が入ります。

FROM filename/pipename/device/cursorname

ロードするデータを備えた SQL ステートメントを参照するファイル、パイプ、装置、またはカーソルを指定します。入力ソースがファイル、パイプ、または装置の場合、CLIENT オプションが指定されていなければ、データベースが存在するデータベース・パーティション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は、別のテープを使用するようユーザーに対してプロンプトが出ます。有効な応答オプションは、次のとおりです。

- c** 続行。警告メッセージを生成した装置を使用し続けます (たとえば、新しいテープがマウントされた場合)。
- d** 装置の終了。警告メッセージを生成した装置の使用を停止します (たとえば、それ以上テープがない場合)。
- t** 終了。すべての装置を終了します。

注:

1. 可能なかぎり完全修飾ファイル名を使用してください。リモート・サーバーの場合は、常に完全修飾ファイル名を使用する必要があります。呼び出し側と同じデータベース・パーティションにデータベースが存在する場合には、相対パスを使用することもできます。
2. ファイルが物理的には分割されているが論理的には 1 つのファイルである場合には、複数の IXF ファイルからのデータのロードがサポートされています。ファイルが論理的にも物理的にも分割されている場合は、サポートされていません。(複数の物理ファイルがすべて一度の EXPORT コマンドの呼び出しで作成された場合、それらは論理的には 1 つであると見なされます。)
3. クライアント・マシン上に存在するデータをロードする場合、そのデータは、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。

INDEXING MODE

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

AUTOSELECT

REBUILD モードと INCREMENTAL モードのどちらにするかを、ロード・ユーティリティーが自動的に決定します。

REBUILD

すべての索引が再作成されます。古い表データの索引キー部分も、追加される新しい表データの索引キー部分もすべてソートできるようにするため、ロード・ユーティリティーには十分なリソースが必要となります。

INCREMENTAL

索引に新しいデータが取り込まれて拡張します。このアプローチでは、索引のフリー・スペースが消費されます。このアプローチでは、新たに挿入されるレコードの索引キーを追加するためのソート・スペースがあれば十分です。この方式がサポートされるのは、索引オブジェクトが有効で、かつロード操作の開始時にアクセス可能な場合だけです (たとえば、DEFERRED モードが指定されたロード操作の直後では、この方式は無効です)。このモードを指定したものの、索引の状態などの理由でサポートされない場合は、警告が戻され、REBUILD モードでロード操作が続行されます。同様に、ロード作成フェーズでロード再開操作を開始した場合も、INCREMENTAL モードはサポートされません。

以下の条件がすべて真の場合、増分索引の作成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 (*logretain* または *userexit* が使用可能である)。
- 表が DMS 表スペース内に存在している。
- 索引オブジェクトの存在している表スペースが、ロードしようとしている表に属する他の表オブジェクトによって共有されている。

この制限をう回するため、索引は別々の表スペースに置くようお勧めします。

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の作成を試みません。最新表示が必要であることを示すマークが索引に付けられます。ロード操作とは関係のないこのような索引に最初にアクセスするときは、再作成が強制的に実行されたり、データベースの再始動時に索引が再作成されたりする場合があります。このアプローチでは、最も大きい索引のキー部分をすべて処理できるだけのソート・スペースが必要です。索引を作成するためにその後かかる合計時間は、REBUILD モードの場合よりも長くなります。したがって、この索引作成据え置きモードで複数のロード操作を実行する場合、最初の非ロード・アクセス時に索引を再作成できるようにしておくよりも、順序列内の最後のロード操作で索引の再作成を実行できるようにした方が (パフォーマンスの観点から) 賢明であるといえます。

据え置き索引作成がサポートされるのは、非ユニークな索引がある表だけです。そのため、ロード・フェーズで挿入される複写キーがロード操作後は永続的ではなくなります。

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

INSERT

ロード・ユーティリティを実行できる 4 つのモードのうちの 1 つ。既存の表データを変更することなく、ロードされたデータを表に追加します。

insert-column

データの挿入先となる表の列を指定します。

ロード・ユーティリティは、1 つ以上のスペースを使った名前の列を解析できません。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)
```

は、Int 4 列があるためエラーになります。これは、次のようにして二重引用符で列名を囲むことによって解決できます。

```
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。この表として、システム表または宣言一時表は指定できません。別名、完全修飾、または非修飾の表名を指定できます。修飾子付き表名は、*schema.tablename* の形式です。非修飾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 値が収められているデータ・ファイルへのパス。パスの最後は斜線 (/) でなければなりません。CLIENT オプションを指定した場合、パスは完全修飾し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、LOB 列にロードされる列内に保管されます。*filetype-mod* スtring内に 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。

ファイル・タイプが CURSOR の場合、このオプションはサポートされていません。

LOCK WITH FORCE

ユーティリティはロード・プロセス中に、表ロックなどの様々なロックを獲得します。ロックを獲得する際、このオプションを使用すると、ロードは待機することなく、またタイムアウトになることなく、ターゲット表に競合するロックを持つ他のアプリケーションを強制的にオフにします。システム・カタログ表に対する競合するロックを保留するアプリケーションは、ロード・ユーティリティによって強制的にオフにされることはありません。強制されたアプリケーションは、ロールバックし、ロード・ユーティリティが必要とするロックをリリースします。その後、ロード・ユーティリティ

LOAD

- 2 ーを続行できます。このオプションは、FORCE APPLICATIONS コマンド
2 と同じ権限 (SYSADM または SYSCTRL) を必要とします。
- 2 ALLOW NO ACCESS は、ロード操作の開始時に競合するロックを持つア
2 プリケーションを強制する場合があります。ロードの開始時に、ユーティリ
2 ティーは、表の照会または変更を試みているアプリケーションを強制する場
2 合があります。
- 2 ALLOW READ ACCESS は、ロード操作の開始時および終了時に競合する
2 ロックを持つアプリケーションを強制する場合があります。ロードの開始時
2 に、ロード・ユーティリティーは、表の変更を試みているアプリケーション
2 を強制する場合があります。ロードの終了時に、ロード・ユーティリティー
2 は、表の照会または変更を試みているアプリケーションを強制する場があ
2 ります。

MESSAGES message-file

ロード操作中に生じ得る警告およびエラー・メッセージの宛先を指定します。メッセージ・ファイルを指定しなかった場合、メッセージは標準出力に書き込まれます。このファイルへの完全パスが指定されていない場合、ロード・ユーティリティーは現行のディレクトリーおよびデフォルトのドライブを宛先として使用します。すでに存在するファイル名を指定すると、ロード時に情報が追加されます。

通常、メッセージ・ファイルには、ロード操作の終了時にメッセージが入れますが、それ自体は操作の進行状況のモニターには適していません。

METHOD

- L** データのロードを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。

注: このメソッドは、ASC ファイルの場合にのみ使用することができます、そのファイル・タイプに対してのみ有効なメソッドです。

- N** ロードするデータ・ファイルの中の列の名前を指定します。それらの列名の太文字小文字は、システム・カタログ中の対応する名前の太文字小文字と一致している必要があります。NULL 可能ではない各表の列には、METHOD N リスト内に対応する項目が必要です。たとえば、データ・フィールドが F1、F2、F3、F4、F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、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、ファイル・タイプ IXF または CURSOR の場合にのみ使用することができます。

- P** ロードする入力データ・フィールドのフィールド番号 (1 から始まる) を指定します。NULL 可能ではない各表の列には、METHOD P リスト内に対応する項目が必要です。たとえば、データ・フィールドが F1、F2、F3、F4、F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、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、ファイル・タイプ IXF、DEL、または CURSOR の場合にのみ使用でき、DEL ファイル・タイプに対してのみ有効な方式です。

MODIFIED BY filetype-mod

ファイル・タイプ修飾子オプションを指定します。『LOAD のファイル・タイプ修飾子』を参照してください。

NONRECOVERABLE

ロード・トランザクションがリカバリー不能としてマークされており、それ以降のロールフォワード・アクションによってそれをリカバリーさせることは不可能であることを指定します。ロールフォワード・ユーティリティは、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に "invalid" (無効) としてマークします。さらに、ユーティリティは、その表に対する後続のすべてのトランザクションを無視します。ロールフォワード操作が完了すると、そのような表は、ドロップするか、またはリカバリー不能なロード操作完了後のコミット・ポイントの後に取られたバックアップ (全バックアップまたは表スペースのバックアップ) からのみ、リストアすることができます。

このオプションを使用すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロード操作中にロードされたデータのコピーが作成される必要もなくなります。

FILE LINK CONTROL が指定された DATALINK 列が表に存在している場合、またはそのような列を表に追加しようとしている場合には、このオプションを使用しないでください。

NULL INDICATORS null-indicator-list

このオプションは、METHOD L パラメーターを指定した場合だけ使用できます (つまり、入力ファイルが ASC ファイルの場合)。NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。NULL 標識リストには、METHOD L パラメーターで定義された各データ・フィールドに対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

NULL 標識列中の Y の値は、その列データが NULL であることを指定します。NULL 標識列に Y 以外の文字を指定した場合は、列データが NULL ではなく、METHOD L オプションで指定された列データがロードされることを指定することになります。

NULL 標識文字は MODIFIED BY オプションを使用して変更できます。

OF filetype

データのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)
- DEL (区切り付き ASCII フォーマット)
- IXF (統合交換フォーマット、PC バージョン)。同一のあるいは別の DB2 表からエクスポートされたことを意味します

LOAD

- CURSOR (SELECT または VALUES ステートメントに対して宣言されたカーソル)。

PARTITIONED DB CONFIG

パーティション表へのロードの実行を可能にします。PARTITIONED DB CONFIG パラメーターを使用すると、パーティション・データベース固有の構成オプションを指定することができます。partitioned-db-option の値は以下のいずれかになります。

```
HOSTNAME x
FILE_TRANSFER_CMD x
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

これらのオプションの詳細は、『パーティション・データベース・ロード構成オプション』に説明されています。

REPLACE

ロード・ユーティリティを実行できる 4 つのモードのうちの 1 つ。表の既存データをすべて削除し、ロードされたデータを挿入します。表定義および索引定義は変更されません。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

このオプションは DATALINK 列をもつ表ではサポートされません。

RESTART

ロード・ユーティリティを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を再開します。ロード操作は、ロード、作成、または削除フェーズの最後の整合点から自動的に続行されます。

RESTARTCOUNT

予約済み。

ROWCOUNT n

ロードするファイル内の物理レコードの数 *n* を指定します。ユーザーはファイル内の最初の *n* 個の行だけをロードできます。

SAVECOUNT n

ロード・ユーティリティが *n* 行ごとに整合点を取ることを指定します。この値はページ・カウントに変換され、エクステント・サイズのインターバルに切り上げられます。メッセージは整合点において発行されるので、LOAD QUERY を使用してロード操作をモニターする場合には、このオプ

ションを選択する必要があります。 n の値が十分な大きさにない場合、各整合点で実行される活動の同期化によってパフォーマンスに影響してしまいます。

デフォルトはゼロですが、それは、必要がなければ整合点は確立されないことを意味します。

ファイル・タイプが `CURSOR` の場合、このオプションはサポートされていません。

SORT BUFFER buffer-size

このオプションは、ロード操作時に `SORTHEAP` データベース構成パラメーターをオーバーライドする値を指定します。これは、索引とともに表をロードする場合、また `INDEXING MODE` パラメーターが `DEFERRED` として指定されていない場合にのみ関係があります。指定された値は `SORTHEAP` の値を超えることはありません。このパラメーターは、`SORTHEAP` の値を変更せずに多くの索引を持つ表をロードする際に使用されるソート・メモリーのスロットルで役に立ちます。これは、一般的な照会処理にも影響を与えます。

STATISTICS USE PROFILE

この表で定義されているプロファイルに従ってロード中に統計を収集するようにロード操作に指示します。そのプロファイルは、ロードの実行前に作成されていなければなりません。なおそのプロファイルは、`RUNSTATS` コマンドで作成します。プロファイルが存在しない場合に、プロファイルに従って統計を収集するようにロード操作に指示すると、警告メッセージが戻されて統計は収集されません。

STATISTICS NO

統計データを収集せず、したがってカタログ内の統計データも変更しないことを指定します。これがデフォルトです。

TEMPFILES PATH temp-pathname

ロード操作時に一時ファイルを作成する場合に使用するパスの名前を指定します。これはサーバー・データベース・パーティションに従って完全に修飾し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使用します。場合によっては、このスペースが相当必要になります。以下に示すのは、すべての一時ファイルにどの程度のファイル・システム・スペースを割り振るべきかの見積もりです。

- `DATALINK` 値のある重複行またはリジェクト行ごとに 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・データまたは `LOB` が入っている場合は、15KB のオーバーヘッド。 `INSERT` オプションを指定した場合で、表の中に多量の長フィールドまたは `LOB` データがすでにある場合には、この数値はこれよりもかなり大きくなる場合があります。

TERMINATE

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を終了し、ロード操作が開始された時点まで操作をロールバックします。途中で整合点があっても通過します。その操作に関係する表スペースの状態は通常に戻され、すべての表オブジェクトの整

合性が保たれます (索引オブジェクトが無効とマークされる場合がありますが、そのような場合には、次のアクセス時に索引の再作成が自動的に行われます)。終了するロード操作がロード REPLACE の場合、その表はロード TERMINATE 操作完了後に空の表まで切り捨てられます。終了するロード操作がロード INSERT の場合、その表はロード TERMINATE 操作完了後も元のレコードをすべて保持します。

ロード終了オプションでは、表スペースのバックアップ・ペンディング状態は解除されません。

注: このオプションは DATALINK 列をもつ表ではサポートされません。

USING directory

予約済み。

WARNINGCOUNT *n*

n 個の警告後に、ロード操作を停止します。このパラメーターは、警告は出ないはずであるけれども、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。ロード・ファイルまたはターゲット表を誤って指定した場合、ロード・ユーティリティーは、ロードを試みた行ごとに警告を生成して、それがロードの失敗の原因になります。*n* がゼロの場合、またはこのオプションが指定されていない場合、何度警告が出されてもロード操作は続行します。警告のしきい値に達したためにロード操作が停止された場合でも、あらためて RESTART モードでロード操作を開始できます。ロード操作は最後の整合点から自動的に続行されます。または、入力ファイルの先頭から REPLACE モードであらためてロード操作を開始できます。

WITHOUT PROMPTING

データ・ファイルのリストにロードするすべてのファイルを含め、しかもリストに入っている装置またはディレクトリーがロード操作全体で十分であるということを指定します。続きの入力ファイルが見つからなかったり、ロード操作が終了する前にコピー先がいっぱいになるとロード操作は失敗し、表はロード・ペンディング状態のままになります。

このオプションを指定しない場合に、テープ装置がコピー・イメージ用のテープの終わりに達した場合、またはリスト中の最後の項目がテープ装置であった場合は、ユーザーに対してその装置に新しいテープを装着するよう求めるプロンプトが出されます。

例:

例 1

TABLE1 に以下の 5 つの列があるとしします。

- 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 に以下の 6 つのエレメントがあるとしします。

- ELE1、位置 01 から 20
- ELE2、位置 21 から 22
- ELE5、位置 23 から 23
- ELE3、位置 24 から 27
- ELE4、位置 28 から 31
- ELE6、位置 32 から 32
- ELE6、位置 33 から 40

データ・レコードは以下のとおりです。

```
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
```

以下のコマンドは、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 パラメーターで striptblanks を指定すると、VARCHAR 列の中のブランクが切り捨てられるようになります (たとえば、行 1、2、および 3 の長さがそれぞれ 11、17、および 19 バイトである COL1)。
2. MODIFIED BY パラメーターで reclen=40 を指定すると、各入力レコードの最後が改行文字でなく、各レコードが 40 バイト長であることを指定することになります。最後の 8 バイトは、表のロードには使用されません。
3. COL4 は入力ファイルにはないので、そのデフォルト (NOT NULL WITH DEFAULT と定義されている) を使用して TABLE1 に挿入されます。
4. 位置 23 と 32 は、特定の行で TABLE1 の COL2 と COL3 が NULL としてロードされるかどうかを指示するために使用されます。ある特定のレコードの、その列の NULL 標識位置が Y である場合、その列は NULL になります。N なら、入力レコード中のその列のデータ位置のデータ値 (L(.....)) で定義される) は、その行の列データのソースとして使用されます。この例では、行 1 のどの列も NULL ではなく、行 2 の COL2 は NULL であり、行 3 の COL3 は NULL です。
5. この例では、COL1 と COL5 の NULL INDICATORS は 0 (ゼロ) として指定されますが、それはそのデータを NULL 不可能である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 に次の 3 つの列があるとします。

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

LOAD

ASCFILE1 には次の 3 つのエレメントがあるとします。

- ELE1、位置 01 から 04
- ELE2、位置 06 から 13
- ELE3、位置 15 から 22

次に示すファイルは、 /u/user1 または /u/user1/bin のどちらかにあります。

- ASCFILE2 - LOB データを持つ
- ASCFILE3 - LOB データを持つ
- ASCFILE4 - LOB データを持つ
- ASCFILE5 - LOB データを持つ
- ASCFILE6 - LOB データを持つ
- ASCFILE7 - LOB データを持つ

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
```

以下のコマンドは、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 パラメーターの中で lobsinfile を指定すると、ファイルからすべての LOB データをロードすることをローダーに対して指定することになります。
2. MODIFIED BY パラメーターで reclen=22 を指定すると、各入力レコードの最後が改行文字でなく、各レコードが 22 バイト長であることを指定することになります。
3. LOB データは、ASCFILE2 から ASCFILE7 までの 6 つのファイルに入っています。各ファイルには、特定の行の LOB 列をロードするのに使用されるデータが入れています。LOB と他のデータのリレーションシップは、ASCFILE1 に指定します。このファイルの最初のレコードは、REC1 を行 1 の COL1 にするようローダーに指示します。行 1 の LOB1 をロードするには ASCFILE2 の内容が使われ、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、行 2 の LOB1 および LOB2 をロードするには ASCFILE4 と ASCFILE5 が使われ、行 3 の LOB をロードするには ASCFILE6 と ASCFILE7 が使われます。
4. これらのファイルがローダーで必要になった場合には、名前の指定された LOB ファイルを探索するのに使われる 2 つのパスが、LOBS FROM パラメーターに入っています。
5. lobsinfile 修飾子を指定しないで ASCFILE1 (区切りなしの ASCII ファイル) から直接 LOB をロードする場合は、以下の規則を守ってください。
 - LOB を含めたレコードの全長は 32KB 以下でなければなりません。

- 入力レコード内の LOB フィールドは固定長でなければならず、必要な場合 LOB データにブランクを埋め込まなければなりません。
- LOB をデータベースに挿入する際に、LOB の埋め込みに使われる後続ブランクを除去できるよう、`striptblanks` 修飾子を指定する必要があります。

例 3 (ダンプ・ファイルの使用)

表 FRIENDS は、次のように定義されています。

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

以下のデータ・レコードをこの表にロードしようとする、

```
23, 24, bobby
, 45, john
4,, mary
```

最初の INT が NULL で、列定義に NOT NULL が指定されているため、第 2 行はリジェクトされます。DEL 形式と互換でない開始文字の入った列は、エラーを生成し、レコードはリジェクトされます。そのようなレコードは、ダンプ・ファイルに書き込むことができます。

区切り文字の外側にある列の DEL データは無視されますが、警告が生成されます。例:

```
22,34,"bob"
24,55,"sam" sdf
```

ユーティリティーは、表の第 3 列に "sam" をロードし、警告の中で文字 "sdf" にフラグが付けられます。このレコードはリジェクトされません。別の例を考えましょう。

```
22 3, 34,"bob"
```

ユーティリティーは 22,34,"bob" をロードし、列 1 の中で 22 より後のデータは無視されたという警告を生成します。このレコードはリジェクトされません。

例 4 (DATALINK データのロード)

下記のコマンドは、DEL 形式のデータをもった入力ファイル `delfile1` から、表 MOVIE TABLE 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. この表には下記の 4 つの列が入っています。

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

2. 入力ファイルの中の DATALINK データのサブフィールド区切り文字は、縦線 (|) 文字です。

3. url_making_of の列値に接頭部文字シーケンスが組み込まれていない場合、
"http://narang" が使用されます。
4. url_movie の非 NULL 列値には、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 の非 NULL 列値のパスには、".mpeg" が付加されます。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. 表のロード中にユニーク索引または 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 に入れます。

例 5 (ID 列がある表へのロード)

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 TABLE1 と同じですが、C2 が GENERATED ALWAYS ID 列である
点が異なります。

DATAFILE1 のデータ・レコード (DEL フォーマット):

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

DATAFILE2 のデータ・レコード (DEL フォーマット):

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

注:

1. 以下のコマンドは、DATAFILE1 で行 1 および 2 への ID 値の提供がないの
で、それらの行のための ID 値を生成します。ただし、行 3 および 4 は、それ
ぞれユーザー提供の ID 値 100 と 101 が割り当てられます。

```
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対する ID 値を生成するに
は、以下のコマンドのいずれかを発行します。

```
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 にロードして各行に対する ID 値を生成するには、以
下のコマンドのいずれかを発行します。

```
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. ID 値 100 と 101 を行 3 および 4 に割り当てるために DATAFILE1 を TABLE2 にロードするには、以下のコマンドを発行します。

```
db2 load from datafile1.del of del modified by identityoverride
replace into table2
```

この場合、ユーティリティには、ユーザー提供の値を優先して、システム生成の ID 値に上書きするように指示しているため、行 1 および 2 はリジェクトされます。ユーザー提供の値が存在しない場合でも、ID 列が暗黙的に非 NULL であるため、この行はリジェクトする必要があります。

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 はリジェクトされます。これは、行 3 と 4 では独自に非 NULL 値が提供されており、ID 列が GENERATED ALWAYS であるためです。

例 6 (CURSOR ファイル・タイプを使用したロード)

表 ABC.TABLE1 には次の 3 つの列があります。

```
ONE INT
TWO CHAR(10)
THREE DATE
```

表 ABC.TABLE2 には次の 3 つの列があります。

```
ONE VARCHAR
TWO INT
THREE DATE
```

以下のコマンドを実行すると、すべてのデータが ABC.TABLE1 から ABC.TABLE2 にロードされます。

```
db2 declare mycurs cursor for select two,one,three from abc.table1
db2 load from mycurs of cursor insert into abc.table2
```

使用上の注意:

データは、入力ファイル内に並んでいる順序でロードされます。特定の順序にしたい場合には、ロードが試行される前にデータをソートしてください。

ロード・ユーティリティは、既存の定義に基づいて索引を作成します。ユニーク・キーの重複を処理するには、例外表が使用されます。ユーティリティは、参照保全を強制したり、制約検査を実行したり、ロードする表に従属するサマリー表を更新したりすることはありません。参照制約またはチェック制約を組み込まれた表は、チェック・ペンディング状態になります。REFRESH IMMEDIATE として定義されているサマリー表、およびロードする表に依存するサマリー表もまた、チェック・ペンディング状態になります。表のチェック・ペンディング状態を解除するには、SET INTEGRITY ステートメントを発行してください。ロード操作は、複製されたサマリー表に対しては実行できません。

クラスタリング索引が表に存在する場合、ロード前にクラスタリング索引でデータをソートしてください。ただし、データはマルチディメンション・クラスタリング (MDC) 表にロードする前にソートする必要はありません。

DB2 Data Links Manager の考慮事項:

各 DATALINK 列ごとに、括弧内にそれぞれ 1 つの列を指定できます。それぞれの列の指定は、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。接頭部情報は、DL_URL_REPLACE_PREFIX、または DL_URL_DEFAULT_PREFIX の指定のいずれかになります。

DATALINK 列指定の数は、表で定義されている DATALINK の数と同じだけ定義できます。指定の順序は、挿入列リストの中での DATALINK 列の順序 (挿入列リストが INSERT INTO (insert-column, ...) で指定されている場合) か、または表定義内での順序 (insert-column が指定されていない場合) に従います。

たとえば、表に列 C1、C2、C3、C4、および C5 があり、そのうち C2 と C5 だけが DATALINK 型であり、挿入列 (insert-column) リストが (C1、C5、C3、C2) である場合、2 つの DATALINK 列を指定する必要があります。最初の列の指定は C5 用であり、2 番目の列の指定は C2 用です。挿入列リストを指定しない場合、最初の列の指定は C2 用になり、2 番目の列の指定は C5 用になります。

複数の DATALINK 列があり、一部の列では特別な指定が必要ではない場合、列の指定には、指定の順序をはっきりと示すために、少なくとも括弧を使用する必要があります。どの列にも指定が行われない場合は、空の括弧のリスト全体を除くことができます。したがって、デフォルトで十分な場合には、DATALINK を指定する必要はありません。

FILE LINK CONTROL で定義されている DATALINK 列を備えた表にデータをロードする場合は、ロード・ユーティリティを呼び出す前に、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 として定義されている場合、これらのステップは必要ありません。)

1. DATALINK 列の値によって参照される データ・リンク・サーバーに、DB2 Data Links Manager がインストールされていることを確認する。
2. データベースが DB2 Data Links Manager 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すべてのファイルを、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 Data Links Manager に接頭部名を定義します。
5. (ロードする) DATALINK データによって参照されるデータ・リンク・サーバーを、DB2 Data Links Manager 構成ファイルに登録します。

ロード・ユーティリティの実行中に、DB2 とデータ・リンク・サーバー間の接続が失敗し、ロード操作も失敗してしまう場合があります。その場合には、以下のようになしてください。

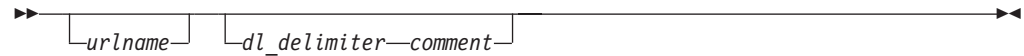
1. データ・リンク・サーバーおよび DB2 Data Links Manager を開始します。
2. ロード再開操作を起動します。

ロード操作中に失敗したリンクはデータ保全性違反と見なされ、ユニーク索引違反と同じような方法で処理されます。そのため、1 つ以上の DATALINK 列の入った表をロードする場合の特別な例外が定義されています。

入力ファイルでの DATALINK 情報の表現

LINKTYPE (現在のところ URL のみサポート) は、DATALINK 情報の一部として指定されていません。LINKTYPE は、LOAD または IMPORT コマンドで指定されますが、PC/IXF タイプの入力ファイルの場合は、適切な列記述子レコードの中で指定されます。

URL LINKTYPE の DATALINK 情報の構文は、以下のとおりです。



urlname と *comment* はいずれもオプションです。どちらも省略した場合、NULL 値が代入されます。

urlname

この URL 名は有効な URL 構文に適合していなければなりません。

注:

1. 現在のところ、「http」、「file」、および「unc」がスキーマ名として許可されています。
2. URL 名の接頭部 (スキーマ、ホスト、およびポート) はオプションです。接頭部がない場合は、ロード・ユーティリティまたはインポート・ユーティリティの DL_URL_DEFAULT_PREFIX または DL_URL_REPLACE_PREFIX 指定の接頭部が使用されます。そのどちらも指定されていない場合、デフォルトの接頭部として "file://localhost" が使われます。したがって、ローカル・ファイルの場合、LOAD または IMPORT コマンド内で DATALINK 列を指定せずに、絶対パス名で指定したファイル名を URL 名として入力することができます。
3. 接頭部が URL 名に付けられている場合も、ロードまたはインポート操作時には、DL_URL_REPLACE_PREFIX で指定した異なる接頭部名によってオーバーライドされます。
4. (DL_URL_SUFFIX を指定した場合、それを追加した後の) 「path」は、リモート・サーバーにあるリモート・ファイルの絶対パス名です。相対パス名は使用できません。HTTP サーバーのデフォルトのパス接頭部は使用されません。

dl_delimiter

区切り付き ASCII (DEL) ファイル形式の場合、*dl_del* 修飾子で指定した文字、あるいは LOAD または IMPORT コマンドのデフォルトの文字。区切りなし ASCII (ASC) ファイル形式の場合、これを文字順序 ¥; (円記号とそれに続くセミコロン) に対応させる必要があります。空白文字 (ブランクやタブなど) を、このパラメーターに指定した値の前後に置くことができます。

comment

DATALINK 値のコメント部分。区切り付き ASCII (DEL) ファイル形式で指定する場合、*comment* テキストは、文字ストリング区切り文字で囲む必要があります。文字ストリング区切り文字は、デフォルトでは二重引用符 (") です。この文字ストリング区切り文字は、LOAD または IMPORT コマンドで MODIFIED BY *filetype-mod* を指定することによりオーバーライドできます。

LOAD

コメントを指定しない場合、このコメントはデフォルトでは長さがゼロのストリングになります。

次に示すのは、区切り付き ASCII (DEL) ファイル形式での DATALINK データの例です。

- `http://www.almaden.ibm.com:80/mrep/intro.mpeg; "Intro Movie"`

これは、以下のような部分から構成されています。

- スキーム = `http`
- サーバー = `www.almaden.ibm.com`
- パス = `/mrep/intro.mpeg`
- 注釈 = `"Intro Movie"`

- `file://narang/u/narang; "InderPal's Home Page"`

これは、次のような部分から構成されています。

- スキーム = `file`
- サーバー = `narang`
- パス = `/u/narang`
- 注釈 = `"InderPal's Home Page"`

次に示すのは、区切りなし ASCII (ASC) ファイル形式での DATALINK データの例です。

- `http://www.almaden.ibm.com:80/mrep/intro.mpeg¥;Intro Movie`

これは、以下のような部分から構成されています。

- スキーム = `http`
- サーバー = `www.almaden.ibm.com`
- パス = `/mrep/intro.mpeg`
- 注釈 = `"Intro Movie"`

- `file://narang/u/narang¥; InderPal's Home Page`

これは、次のような部分から構成されています。

- スキーム = `file`
- サーバー = `narang`
- パス = `/u/narang`
- 注釈 = `"InderPal's Home Page"`

以下に、DATALINK データの例を示します。列のロードまたはインポート指定が `DL_URL_REPLACE_PREFIX` ("`http://qso`") であるとしています。

- `http://www.almaden.ibm.com/mrep/intro.mpeg`

これは、以下のような部分から構成されています。

- スキーマ = `http`
- サーバー = `qso`
- パス = `/mrep/intro.mpeg`

- 注釈 = NULL スtring
- /u/me/myfile.ps

これは、次のような部分から構成されています。

- スキーマ = http
- サーバー = qso
- パス = /u/me/myfile.ps
- 注釈 = NULL スtring

関連概念:

- 84 ページの『ロードの概要』
- 92 ページの『ロードの使用に必要な特権、権限、および許可』

関連タスク:

- 93 ページの『ロードの使用』

関連資料:

- 「コマンド・リファレンス」の『QUIESCE TABLESPACES FOR TABLE コマンド』
- 「コマンド・リファレンス」の『db2atld - オートローダー・コマンド』
- 195 ページの『ロード - CLP の例』
- 213 ページの『パーティション・データベース・ロード構成オプション』
- 139 ページの『db2Load - ロード』
- 168 ページの『ロード用のファイル・タイプ修飾子』

LOAD QUERY

処理中にロード操作の状況を調べ、表の状態を戻します。ロードが行われていない場合は、表の状態だけが戻されます。このコマンドを正常に呼び出すためには、同じデータベースへの接続と、別の 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。

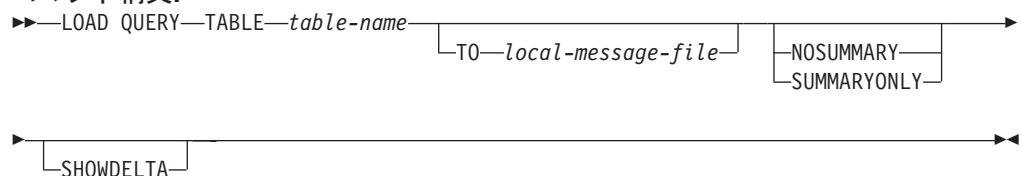
権限:

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:

LOAD QUERY

NOSUMMARY

ロード・サマリー情報 (読み取られた行、スキップされた行、ロードされた行、リジェクトされた行、削除された行、コミットされた行、警告の数) のレポートを生成しないよう指定します。

SHOWDELTA

新しい情報 (最後の LOAD QUERY コマンド以後に発生したロード・イベントに関する) のレポートだけを生成するよう指定します。

SUMMARYONLY

ロード・サマリー情報のレポートだけを生成するよう指定します。

TABLE table-name

データが現在ロード中の表の名前を指定します。非修飾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

ロード操作中に生じ得る警告およびエラー・メッセージの宛先を指定します。このファイルは、LOAD コマンド用に指定された *message-file* であってはなりません。ファイルがすでに存在する場合、ロード・ユーティリティが生成するメッセージはすべてそのファイルに追加されます。

例:

大量のデータを STAFF 表にロードしている場合、ロード操作の状況をチェックすることが必要になるかもしれません。ユーザーは次のように指定することができます。

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

出力ファイル /u/mydir/staff.tempmsg は、次のようになります。

```
SQL3501W The table space(s) in which the table resides will not be placed in
backup pending state since forward recovery is disabled for the database.
```

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

```
SQL3500W The utility is beginning the "LOAD" phase at time "03-21-2002
11:31:16.597045".
```

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

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3519W Begin Load Consistency Point. Input record count = "104416".
```

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3519W Begin Load Consistency Point. Input record count = "205757".
```

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3519W Begin Load Consistency Point. Input record count = "307098".
```

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3519W Begin Load Consistency Point. Input record count = "408439".
```

```
SQL3520W Load Consistency Point was successful.
```

SQL3532I The Load utility is currently in the "LOAD" phase.

```
Number of rows read      = 453376
Number of rows skipped   = 0
Number of rows loaded    = 453376
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 408439
Number of warnings       = 0
```

Tablestate:
Load in Progress

使用上の注意:

ロード・ユーティリティーは、ロックに加えて、表状態を使用して、表へのアクセスを制御します。表状態の判別には、LOAD QUERY コマンドを使用できます。LOAD QUERY で説明される表状態については、『表ロック、表状態、および表スペース状態』に説明されています。

ロード操作の進行状況は、LIST UTILITIES コマンドを使ってモニターすることもできます。

関連概念:

- 84 ページの『ロードの概要』
- 184 ページの『表ロック、表状態、および表スペース状態』

関連資料:

- 113 ページの『LOAD』
- 「コマンド・リファレンス」の『LIST UTILITIES コマンド』

db2Load - ロード

データを DB2 表にロードします。サーバー上にあるデータは、ファイル、カーソル、テープ、または名前付きパイプの形式とすることができます。リモートで接続しているクライアント上にあるデータは、完全修飾ファイル、カーソル、または名前付きパイプの形式とすることができます。ロード・ユーティリティーは、階層レベルでのデータのロードをサポートしていません。

許可:

以下のいずれかです。

- *sysadm*
- *dbadm*
- データベースのロード権限と以下のもの
 - 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)。

注: 一般的に、すべてのロード処理、およびすべての DB2 サーバー処理は、インスタンス所有者に所有されています。これらのすべての処理では、インスタンス所有者の ID を使用して、必要なファイルにアクセスします。そのため、インスタンス所有者は、誰がコマンドを呼び出すかに関係なく、入力ファイルへの読み取りアクセスを持っている必要があります。

必要な接続:

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

インスタンス。明示的なアタッチは必要ありません。データベースへの接続が確立されている場合には、ローカル・インスタンスへの暗黙的な接続が試みられます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: Load */
/* ... */
SQL_API_RC SQL_API_FN
db2Load (
    db2UInt32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
    struct sqlu_media_list *piSourceList;
    struct sqlu_media_list *piLobPathList;
    struct sql_dcol *piDataDescriptor;
    struct sqlchar *piActionString;
    char *piFileType;
    struct sqlchar *piFileTypeMod;
    char *piLocalMsgFileName;
    char *piTempFilesPath;
    struct sqlu_media_list *piVendorSortWorkPaths;
    struct sqlu_media_list *piCopyTargetList;
    db2int32 *piNullIndicators;
    struct db2LoadIn *piLoadInfoIn;
    struct db2LoadOut *poLoadInfoOut;
    struct db2PartLoadIn *piPartLoadInfoIn;
    struct db2PartLoadOut *poPartLoadInfoOut;
    db2int16 iCallerAction;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadIn
{
    db2UInt64                                iRowcount;
    db2UInt64                                iRestartcount;
    char                                       *piUseTablespace;
```

```

        db2UInt32          iSavecount;
        db2UInt32          iDataBufferSize;
        db2UInt32          iSortBufferSize;
        db2UInt32          iWarningcount;
        db2UInt16          iHoldQuiesce;
        db2UInt16          iCpuParallelism;
        db2UInt16          iDiskParallelism;
        db2UInt16          iNonrecoverable;
        db2UInt16          iIndexingMode;
        db2UInt16          iAccessLevel;
        db2UInt16          iLockWithForce;
        db2UInt16          iCheckPending;
        char               iRestartphase;
        char               iStatsOpt;
    } db2LoadIn;

typedef SQL_STRUCTURE db2LoadOut
{
    db2UInt64          oRowsRead;
    db2UInt64          oRowsSkipped;
    db2UInt64          oRowsLoaded;
    db2UInt64          oRowsRejected;
    db2UInt64          oRowsDeleted;
    db2UInt64          oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2PartLoadIn
{
    char               *piHostname;
    char               *piFileTransferCmd;
    char               *piPartFileLocation;
    struct db2LoadNodeList *piOutputNodes;
    struct db2LoadNodeList *piPartitioningNodes;
    db2UInt16          *piMode;
    db2UInt16          *piMaxNumPartAgents;
    db2UInt16          *piIsolatePartErrs;
    db2UInt16          *piStatusInterval;
    struct db2LoadPortRange *piPortRange;
    db2UInt16          *piCheckTruncation;
    char               *piMapFileInput;
    char               *piMapFileOutput;
    db2UInt16          *piTrace;
    db2UInt16          *piNewline;
    char               *piDistfile;
    db2UInt16          *piOmitHeader;
    SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
    SQL_PDB_NODE_TYPE *piNodeList;
    db2UInt16          iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
    db2UInt16          iPortMin;
    db2UInt16          iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
    db2UInt64          oRowsRdPartAgents;
    db2UInt64          oRowsRejPartAgents;
    db2UInt64          oRowsPartitioned;
    struct db2LoadAgentInfo *poAgentInfoList;
    db2UInt32          iMaxAgentInfoEntries;

```

```

        db2UInt32                                oNumAgentInfoEntries;
    } db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
    db2int32                                oSqlcode;
    db2UInt32                                oTableState;
    SQL_PDB_NODE_TYPE                        oNodeNum;
    db2UInt16                                oAgentType;
} db2LoadAgentInfo;
/* ... */

```

汎用 API 構文:

```

/* File: db2ApiDf.h */
/* API: Load */
/* ... */
SQL_API_RC SQL_API_FN
db2gLoad (
    db2UInt32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
{
    struct sqlu_media_list *piSourceList;
    struct sqlu_media_list *piLobPathList;
    struct sqldcol *piDataDescriptor;
    struct sqlchar *piActionString;
    char *piFileType;
    struct sqlchar *piFileTypeMod;
    char *piLocalMsgFileName;
    char *piTempFilesPath;
    struct sqlu_media_list *piVendorSortWorkPaths;
    struct sqlu_media_list *piCopyTargetList;
    db2int32 *piNullIndicators;
    struct db2gLoadIn *piLoadInfoIn;
    struct db2LoadOut *poLoadInfoOut;
    struct db2gPartLoadIn *piPartLoadInfoIn;
    struct db2PartLoadOut *poPartLoadInfoOut;
    db2int16 iCallerAction;
    db2UInt16 iFileTypeLen;
    db2UInt16 iLocalMsgFileLen;
    db2UInt16 iTempFilesPathLen;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
    db2UInt64                                iRowcount;
    db2UInt64                                iRestartcount;
    char *piUseTablesapace;
    db2UInt32                                iSavecount;
    db2UInt32                                iDataBufferSize;
    db2UInt32                                iSortBufferSize;
    db2UInt32                                iWarningcount;
    db2UInt16                                iHoldQuiesce;
    db2UInt16                                iCpuParallelism;
    db2UInt16                                iDiskParallelism;
    db2UInt16                                iNonrecoverable;
    db2UInt16                                iIndexingMode;
    db2UInt16                                iAccessLevel;
    db2UInt16                                iLockWithForce;
    db2UInt16                                iCheckPending;
    char *iRestartphase;
    char *iStatsOpt;
    db2UInt16                                iUseTablesapaceLen;
} db2gLoadIn;

```

```

typedef SQL_STRUCTURE db2LoadOut
{
    db2UInt64                oRowsRead;
    db2UInt64                oRowsSkipped;
    db2UInt64                oRowsLoaded;
    db2UInt64                oRowsRejected;
    db2UInt64                oRowsDeleted;
    db2UInt64                oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2gPartLoadIn
{
    char                    *piHostname;
    char                    *piFileTransferCmd;
    char                    *piPartFileLocation;
    struct db2LoadNodeList *piOutputNodes;
    struct db2LoadNodeList *piPartitioningNodes;
    db2UInt16               *piMode;
    db2UInt16               *piMaxNumPartAgents;
    db2UInt16               *piIsolatePartErrs;
    db2UInt16               *piStatusInterval;
    struct db2LoadPortRange *piPortRange;
    db2UInt16               *piCheckTruncation;
    char                    *piMapFileInput;
    char                    *piMapFileOutput;
    db2UInt16               *piTrace;
    db2UInt16               *piNewline;
    char                    *piDistfile;
    db2UInt16               *piOmitHeader;
    SQL_PDB_NODE_TYPE       *piRunStatDBPartNum;
    db2UInt16               iHostnameLen;
    db2UInt16               iFileTransferLen;
    db2UInt16               iPartFileLocLen;
    db2UInt16               iMapFileInputLen;
    db2UInt16               iMapFileOutputLen;
    db2UInt16               iDistfileLen;
} db2gPartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
    SQL_PDB_NODE_TYPE       *piNodeList;
    db2UInt16               iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
    db2UInt16               iPortMin;
    db2UInt16               iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
    db2UInt64                oRowsRdPartAgents;
    db2UInt64                oRowsRejPartAgents;
    db2UInt64                oRowsPartitioned;
    struct db2LoadAgentInfo *poAgentInfoList;
    db2UInt32                iMaxAgentInfoEntries;
    db2UInt32                oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
    db2int32                 oSqlcode;
    db2UInt32               oTableState;
}

```

```
SQL_PDB_NODE_TYPE
db2UInt16
} db2LoadAgentInfo;
/* ... */

oNodeNum;
oAgentType;
```

API パラメーター:

versionNumber

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

pParmStruct

入力。 *db2LoadStruct* 構造を指すポインター。

pSqlca

出力。 *sqlca* 構造へのポインター。

piSourceList

入力。ソース・ファイル、装置、ベンダー、パイプ、または SQL ステートメントを提供するのに使用される、 *sqlu_media_list* 構造を指すポインター。

この構造に提供される情報は、 *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) は、以下のとおりです。

SQLU_SQL_STMT

media_type フィールドがこの値に設定されている場合、呼び出し側は、ターゲット・フィールドの *pStatement* フィールドで SQL 照会を提供します。 *pStatement* フィールドは、 *sqlu_statement_entry* のタイプです。セッション・フィールドは値を 1 に設定していなければなりません。これは、ロード・ユーティリティーはロードごとに 1 つの SQL 照会だけを受け取るからです。

SQLU_SERVER_LOCATION

media_type フィールドがこの値に設定されている場合、呼び出し側から *sqlu_location_entry* 構造によって情報が提供されます。 *sessions* フィールドは、提供される *sqlu_location_entry* 構造の数を示します。これは、ファイル、装置、および Named PIPE に使用されます。

SQLU_CLIENT_LOCATION

media_type フィールドがこの値に設定されている場合、呼び出し側から *sqlu_location_entry* 構造によって情報が提供されます。 *sessions* フィールドは、提供される *sqlu_location_entry* 構造の数を示します。これは、完全修飾ファイル、および Named PIPE に使用されます。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を注意してください。

SQLU_TSM_MEDIA

media_type フィールドがこの値に設定されている場合、 *sqlu_vendor* 構造が使用されます。 *filename* には、ロードされるデータにユニークな ID が入ります。 *sessions* の値がいくつであっても、 *sqlu_vendor* 項目の数は 1 つだけにする必要があります。 *sessions* フィールドは、開始される TSM セッションの数を示します。ロー

ド・ユーティリティは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの *sqlu_vendor* 項目にあるものと同じです。

SQLU_OTHER_MEDIA

media_type フィールドがこの値に設定されている場合、*sqlu_vendor* 構造が使用されます。*shr_lib* には共有ライブラリー名、*filename* にはロードされるデータにユニークな ID が入ります。*sessions* の値がいくつであっても、*sqlu_vendor* 項目の数は 1 つだけにする必要があります。*sessions* フィールドは、開始されるその他のベンダー・セッションの数を示します。ロード・ユーティリティは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの *sqlu_vendor* 項目にあるものと同じです。

piLobPathList

入力。*sqlu_media_list* 構造を指すポインター。ファイル・タイプが IXF、ASC、および DEL の場合は、ロードされる個々の LOB ファイルのロケーションを識別する、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 ファイルで検索され、提供されたパスに追加されます。

この構造に提供される情報は、*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) は、以下のとおりです。

SQLU_LOCAL_MEDIA

この値に設定されている場合、呼び出し側から *sqlu_media_entry* 構造によって情報が提供されます。*sessions* フィールドは、提供される *sqlu_media_entry* 構造の数を示します。

SQLU_TSM_MEDIA

この値に設定されている場合、*sqlu_vendor* 構造が使用されます。*filename* には、ロードされるデータにユニークな ID が入ります。*sessions* の値がいくつであっても、*sqlu_vendor* 項目の数は 1 つだけにする必要があります。*sessions* フィールドは、開始される TSM セッションの数を示します。ロード・ユーティリティは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの *sqlu_vendor* 項目にあるものと同じです。

SQLU_OTHER_MEDIA

この値に設定されている場合、*sqlu_vendor* 構造が使用されます。*shr_lib* には共有ライブラリー名、*filename* にはロードされるデータにユニークな ID が入ります。*sessions* の値がいくつであっても、*sqlu_vendor* 項目の数は 1 つだけにする必要があります。*sessions* フィールドは、開始されるその他のベンダー・セッションの数を示します。ロード・ユーティリティは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの *sqlu_vendor* 項目にあるものと同じです。

piDataDescriptor

入力。外部ファイルからロードするよう選択された列に関する情報を備えた *sqldcol* 構造を指すポインター。

pFileType パラメーターが *SQL_ASC* に設定されている場合、この構造の *dcolmeth* フィールドは、*SQL_METH_L* または *SQL_METH_D* に設定し、ファイル名は、開始と終了の対および *NULL* 標識の位置を示す *POSITIONSFILE* *pFileTypeMod* 修飾子とともに指定する必要があります。ユーザーは、ロードされる列ごとに開始ロケーションと終了ロケーションを指定します。

ファイル・タイプ *SQL_DEL* の場合、*dcolmeth* は *SQL_METH_P* または *SQL_METH_D* のどちらかにすることができます。 *SQL_METH_P* の場合、ソース列の位置を提供する必要があります。 *SQL_METH_D* の場合は、ファイル内の最初の列が表の最初の列にロードされ、以下同様に続きます。

ファイル・タイプが *SQL_IXF* の場合、*dcolmeth* は *SQL_METH_P*、*SQL_METH_D*、または *SQL_METH_N* のいずれかにすることができます。この場合は、*SQL_METH_N* が *sqldcol* 構造でファイル列名が提供されるべきであることを示す点を除き、*DEL* ファイルに関する規則が適用されます。

piActionString

入力。 *sqlchar* 構造を指すポインターと、それに続いて表に影響するアクションを指定する文字の配列。

文字配列の形式は、以下のようになります。

```
"INSERT|REPLACE|RESTART|TERMINATE
INTO tname [(column_list)]
[ATALINK SPECIFICATION datalink-spec]
[FOR EXCEPTION e_tname]"
```

INSERT

既存の表データを変更することなく、ロードされたデータを表に追加します。

REPLACE

表から既存データをすべて削除し、ロードされたデータを挿入します。表定義および索引定義は変更されません。

RESTART

以前に割り込みを受けたロード操作を再開します。ロード操作は、ロード、作成、または削除フェーズの最後の整合点から自動的に続行されます。

TERMINATE

以前に割り込みを受けたロード操作を終了し、ロード操作が開始された時点まで操作をロールバックします。途中に整合点があっても通過します。その操作に関係する表スペースの状態は通常に戻され、すべての表オブジェクトの整合性が保たれます (索引オブジェクトが無効とマークされる場合がありますが、そのような場合には、次のアクセス時に索引の再作成が自動的に行われます)。表の存在する表スペースがロード・ペンディング状態でなければ、このオプションは表スペースの状態に影響しません。

ロード終了オプションでは、表スペースのバックアップ・ペンディング状態は解除されません。

tname 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、完全修飾、または非修飾の表

名を指定できます。修飾された表名は、*schema.tablename* の形式になります。非修飾の表名を指定すると、その表は **CURRENT SCHEMA** で修飾されます。

(column_list)

データの挿入先の表の列名のリスト。列名は、コンマで区切らなければなりません。名前にスペースまたは小文字が使われている場合には、それを引用符で囲まなければなりません。

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** コマンドと同じ構文を使って指定できます。

FOR EXCEPTION *e_tbname*

エラーが発生した行のコピー先となる例外表を指定します。ユニーク索引または主キー索引に違反した行がすべてコピーされます。**DATALINK** 例外も例外表にキャプチャーされます。

piFileType

入力。入力データ・ソースのフォーマットを示すストリング。サポートされている外部のフォーマット (*sqlutil* で定義) は、以下のとおりです。

SQL_ASC

区切り文字なし ASCII。

SQL_DEL

区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、IBM Personal Decision Series プログラム、およびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための **DEL** フォーマット・ファイルです。

SQL_IXF

IXF (統合交換フォーマットの PC バージョン)。表からデータをエクスポートする場合の推奨方式で、同じ表または別のデータベース・マネージャー表にそれをロードすることが可能です。

SQL_CURSOR

SQL 照会。 *piSourceList* パラメーターによって渡された *sqlu_media_list* 構造のタイプは **SQLU_SQL_STMT** で、実際の SQL 照会を参照し、それに対して宣言されているカーソルは参照しません。

piFileTypeMod

入力。 *sqlchar* 構造を指すポインターと、それに続いて 1 つ以上の処理オプションを指定する文字の配列。このポインターが **NULL** であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。『**LOAD** のファイル・タイプ修飾子』を参照してください。

piLocalMsgFileName

入力。出力メッセージの書き込み先となるローカル・ファイルの名前を示したストリング。

piTempFilePath

入力。一時ファイル用のサーバー上で使用されるパス名を示したストリング。一時ファイルは、メッセージや整合点を格納したり、フェーズ情報を削除したりするために作成されます。

piVendorSortWorkPaths

入力。ベンダー・ソート作業ディレクトリーを指定する *sqlu_media_list* 構造を指すポインター。

piCopyTargetList

入力。 *sqlu_media_list* 構造を指すポインター。これは、(コピー・イメージを作成する予定の場合) コピー・イメージの書き込み先となるターゲット・パス、装置、または共用ライブラリーのリストを提供するときに使用します。

この構造に入力する値は、*media_type* フィールドの値によって異なります。このフィールドに有効な値 (*sqlutil* で定義) 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予定の場合、*media_type* をこの値に設定し、ターゲットに関する情報を *sqlu_media_entry* 構造に提供してください。 *sessions* フィールドは、提供される *sqlu_media_entry* 構造の数を示します。

SQLU_TSM_MEDIA

コピーを TSM に書き込む予定の場合、この値を使用してください。それ以外の情報は特に必要ありません。

SQLU_OTHER_MEDIA

ベンダー製品を使用する予定の場合、この値を使用し、 *sqlu_vendor* 構造を介して追加の情報を提供してください。この構造の *shr_lib* フィールドをベンダー製品の共用ライブラリー名に設定してください。 *sessions* の値に関係なく、1 つの *sqlu_vendor* 項目だけを提供してください。 *sessions* フィールドは、提供される *sqlu_media_entry* 構造の数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの *sqlu_vendor* 項目で提供されているものと同じです。

piNullIndicators

入力。ASC ファイルの場合にのみ使用します。列データが NULL 可能であるかどうかを示す整数の配列です。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の間には、1 対 1 の順序付けられた対応関係があります。要するに、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、NULL 標識フィールドとして使用される、データ・ファイル内のロケーションを識別する数値、または表の列が NULL 可能ではないことを示すゼロが組み込まれます。要素がゼロでない場合には、データ・ファイル内の識別されたロケーションに Y または N が入っていなければなりません。

ん。 Y は表の列のデータが NULL であることを示し、 N は表の列のデータが NULL ではないことを示します。

piLoadInfoln

入力。 *db2LoadIn* 構造を指すポインター。

poLoadInfoOut

入力。 *db2LoadOut* 構造を指すポインター。

piPartLoadInfoln

入力。 *db2PartLoadIn* 構造を指すポインター。

poPartLoadInfoOut

出力。 *db2PartLoadOut* 構造を指すポインター。

iCallerAction

入力。呼び出し側が要求するアクションを示します。有効な値 (sqlutil で定義) は、以下のとおりです。

SQLU_INITIAL

最初の呼び出し。この値 (または SQLU_NOINTERRUPT) は、API への最初の呼び出しの際には必ず使用してください。

SQLU_NOINTERRUPT

最初の呼び出し。処理を中断しません。この値 (または SQLU_INITIAL) は、API への最初の呼び出しの際には必ず使用してください。

最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたロード操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定しなければなりません。

SQLU_CONTINUE

処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティがユーザー入力 (たとえば、テープの終わり条件への応答) を要求した後で、API への後続呼び出しを出す場合だけです。この値は、ユーティリティが要求したユーザー・アクションが完了したら、ユーティリティが最初の要求の処理を続行するよう指定するものです。

SQLU_TERMINATE

処理の終了。ロード中の表スペースを LOAD_PENDING 状態にしたまま、ロード・ユーティリティを早期に終了させます。このオプションは、これ以上データの処理が行われない場合に指定します。

SQLU_ABORT

処理の終了。ロード中の表スペースを LOAD_PENDING 状態にしたまま、ロード・ユーティリティを早期に終了させます。このオプションは、これ以上データの処理が行われない場合に指定します。

SQLU_RESTART

処理の再開。

SQLU_DEVICE_TERMINATE

単一の装置の終了。このオプションは、ユーティリティーが装置からの読み取りを停止しても、データの処理をさらに続ける場合に指定します。

iFileTypeLen

入力。 *iFileType* の長さ (バイト単位) を指定します。

iLocalMsgFileLen

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

iTempFilesPathLen

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

iRowcount

入力。ロードされる物理レコードの数。これを使用すると、ファイル内の最初の *rowcnt* 個の行だけをロードすることができます。

iRestartcount

入力。将来の使用のために予約されています。

piUseTablespace

入力。索引が再作成されている場合、索引のシャドー・コピーが表スペース *iUseTablespaceName* 内に作成され、ロード終了時に元の表スペースにコピーされます。システム TEMPORARY 表スペースのみ、このオプションを使用できます。指定されない場合、シャドー索引が、索引オブジェクトと同じ表スペース内に作成されます。

シャドー・コピーが索引オブジェクトと同じ表スペース内に作成される場合、古い索引オブジェクトを介したシャドー索引オブジェクトのコピーは瞬時に終了します。シャドー・コピーが索引オブジェクトとは異なる表スペースにある場合、物理コピーが実行されます。これにはかなりの入出力および時間を要します。コピーは、表がロード終了時にオフラインであるときに行われます。

iAccessLevel が SQLU_ALLOW_NO_ACCESS である場合、このフィールドは無視されます。

ユーザーが INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT を指定しない場合、このオプションは無視されます。このオプションは INDEXING MODE AUTOSELECT が選択され、ロードが索引を徐々に更新することを選択した場合にも無視されます。

iSavecount

整合点を確立する前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・サイズのインターバルに切り上げられます。それぞれの整合点でメッセージが発行されるため、*db2LoadQuery* - 照会のロード を用いてロード操作をモニターする場合には、このオプションを選択する必要があります。 *savecnt* の値を大きく設定しておかないと、それぞれの整合点で実行される活動の同期がとられるときにパフォーマンスに影響がおよびます。

デフォルト値は 0 であり、これは、必要のない限り整合点が確立されないことを意味します。

iDataBufferSize

ユーティリティ内でデータ転送用のバッファ・スペースとして使用される 4KB ページの数 (並列処理の度合いとは無関係)。指定された値がアルゴリズムの最小値よりも小さい場合には、必要最低限のページが使用され、警告は戻されません。

このメモリーは、ユーティリティ・ヒープから直接に割り振られます (ユーティリティ・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を用いて修正できます)。

値を指定しないと、ランタイムにユーティリティによって適切なデフォルトが計算されます。デフォルトは、表の特性だけでなく、ローダーのインスタンス生成時にユーティリティ・ヒープ中で使用可能なフリー・スペースの割合に基づいています。

iSortBufferSize

入力。このオプションは、ロード操作時に `SORTHEAP` データベース構成パラメーターをオーバーライドする値を指定します。これは表を索引とともにロードする場合、および `iIndexingMode` パラメーターが `SQLU_INX_DEFERRED` として指定されない場合にのみ関係があります。指定される値は、`SORTHEAP` の値を超えることはできません。このパラメーターは、一般的な照会処理にも影響を与える `SORTHEAP` の値を変更せずに、`LOAD` によって使用されるソート・メモリーをスロットルするために役立ちます。

iWarningcount

入力。 `warningcnt` 個の警告後に、ロード操作を停止します。このパラメーターは、警告は出ないはずであるけれども、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。ロード・ファイルまたはターゲット表を誤って指定した場合、ロード・ユーティリティは、ロードを試みた行ごとに警告を生成して、それがロードの失敗の原因になります。 `warningcnt` が 0 であるか、またはこのオプションを指定していない場合には、ロード操作は、発行された警告の数に関係なく続行されます。

警告のしきい値を超過したためにロード操作が停止された場合には、`RESTART` モードでもう一度ロード操作を開始することができます。ロード操作は、最後の整合点から自動的に続行します。または、入力ファイルの先頭から `REPLACE` モードであらためてロード操作を開始できます。

iHoldQuiesce

入力。ユーティリティによって、ロード後に表を排他静止状態のままにする場合は `TRUE`、それ以外の場合は `FALSE` に値が設定されるフラグ。

iCpuParallelism

入力。ユーティリティが表オブジェクトの作成時にレコードを解析、変換、およびフォーマット設定するために作成するプロセスつまりスレッドの数。このパラメーターは、パーティション内並列処理を活用するために設計されています。これは、事前にソートされたデータをロードする際に役立ちます (ソース・データのレコード順序が保持されるため)。このパラメーターの値がゼロである場合には、ロード・ユーティリティはランタイムに適切なデフォルト値を使用します。注: このパラメーターが `LOB` または `LONG`

VARCHAR フィールドを備えた表で使用されると、システム CPU の数やユーザーによって指定された値に関係なく、値は 1 になります。

iDiskParallelism

入力。ユーティリティがデータを表スペース・コンテナに書き込むために作成するプロセスつまりスレッドの数。値を指定しないと、ユーティリティは表スペース・コンテナの数と表の特性に基づいて適切なデフォルトを選択します。

iNonrecoverable

入力。ロード・トランザクションがリカバリー不能としてマークされ、後続のロールフォワード・アクションによってリカバリーできない場合には、`SQLU_NON_RECOVERABLE_LOAD` に設定します。ロールフォワード・ユーティリティは、このトランザクションをスキップし、データがロードされようとしていた表を「無効」としてマークします。さらに、ユーティリティは、その表に対する後続のすべてのトランザクションを無視します。ロールフォワードが完了したら、そのような表はドロップするしかありません。このオプションを使用すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロード操作中にロードされたデータのコピーが作成される必要もなくなります。ロード・トランザクションがリカバリー可能としてマークされる場合には、`SQLU_RECOVERABLE_LOAD` に設定します。

iIndexingMode

入力。索引付けモードを指定します。有効な値 (`sqlutil` で定義) は、以下のとおりです。

SQLU_INX_AUTOSELECT

LOAD は REBUILD と INCREMENTAL 索引モードの間で選択します。

SQLU_INX_REBUILD

表索引を再作成します。

SQLU_INX_INCREMENTAL

既存の索引を拡張します。

SQLU_INX_DEFERRED

表索引を更新しません。

iAccessLevel

入力。アクセス・レベルを指定します。有効な値は以下のとおりです。

SQLU_ALLOW_NO_ACCESS

ロードが表を排他ロックするように指定します。

SQLU_ALLOW_READ_ACCESS

表の元データ (非差分部分) が、ロードが進行中の間、リーダーに対して可視のままであるように指定します。このオプションは、ロードの付加 (たとえば、ロードの挿入など) に対してのみ有効です。ロード置換に対しては無視されます。

iLockWithForce

入力。ブール・フラグ。TRUE に設定された場合、ロードは必要に応じて他のアプリケーションに対し、必ず即時に表ロックを得るように強制しま

す。このオプションは、FORCE APPLICATIONS コマンド (SYSADM または SYSCTRL) と同じ権限を必要とします。

SQLU_ALLOW_NO_ACCESS ロードは、ロード操作の開始時に、アプリケーションの競合を強制終了させることができます。ロードの開始時に、ユーティリティは、表の照会または変更のいずれかを試みるアプリケーションを強制する場合があります。

SQLU_ALLOW_READ_ACCESS ロードは、ロード操作の開始時または終了時に、アプリケーションの競合を強制終了させることができます。ロードの開始時に、ロード・ユーティリティは、表の変更を試みるアプリケーションを強制する場合があります。ロードの終了時に、ロード・ユーティリティは、表の照会または変更のいずれかを試みるアプリケーションを強制する場合があります。

iCheckPending

入力。表をチェック・ペンディング状態にするように指定します。
SQLU_CHECK_PENDING_CASCADE_IMMEDIATE が指定されている場合、チェック・ペンディング状態は即時にすべての従属表および下層表にカスケードされます。SQLU_CHECK_PENDING_CASCADE_DEFERRED が指定されている場合、チェック・ペンディング状態の従属表へのカスケードは、ターゲット表の保全性違反がチェックされるまで据え置かれます。このオプションが指定されていない場合、SQLU_CHECK_PENDING_CASCADE_DEFERRED がデフォルトとなります。

iRestartphase

入力。予約済み。有効な値は、シングルスペース文字 ' ' です。

iStatsOpt

入力。収集する統計の細分性。有効な値は以下のとおりです。

SQLU_STATS_NONE

統計は収集されません。

SQLU_STATS_USE_PROFILE

現在の表で定義されているプロファイルに基づいて統計が収集されます。そのプロファイルは、RUNSTATS コマンドを使って作成されていなければなりません。現在の表のプロファイルが存在しない場合、警告が戻されて、統計は収集されません。

iUseTablespaceLen

入力。piUseTablespace の長さ (バイト単位)。

oRowsRead

出力。ロード操作中に読み取られたレコードの数。

oRowsSkipped

出力。ロード操作が開始される前にスキップされたレコードの数。

oRowsLoaded

出力。ターゲット表にロードされた行の数。

oRowsRejected

出力。ロードできなかったレコードの数。

oRowsDeleted

出力。削除された重複行の数。

oRowsCommitted

出力。処理されたレコードの合計数。正常にロードされ、データベースにコミットされたレコードの数と、スキップまたはリジェクトされたレコードの数の合計。

piHostname

入力。 *iFileTransferCmd* パラメーターのホスト名。 NULL の場合、ホスト名のデフォルトは「nohost」です。

piFileTransferCmd

入力。ファイル転送コマンドのパラメーター。必要ない場合、NULL に設定しなければなりません。このパラメーターの詳細については、「Data Movement Guide」を参照してください。

piPartFileLocation

入力。 PARTITION_ONLY、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、このパラメーターは、パーティション・ファイルのロケーションを指定するために使用できます。このロケーションは、*piOutputNodes* オプションで指定された各パーティションに存在している必要があります。

SQL_CURSOR ファイル・タイプの場合、このパラメーターは NULL にすることはできません。ロケーションはパスを参照しませんが、完全修飾されたファイル名を参照します。これは、PARTITION_ONLY モードの場合は、各出力パーティションで作成されたパーティション・ファイルの完全修飾された基本ファイル名、または LOAD_ONLY モードの場合は、各パーティションから読み取られるファイルのロケーションです。PARTITION_ONLY モードでターゲット表に LOB 列が存在する場合、指定された基本名のファイルが複数作成されることがあります。SQL_CURSOR 以外のファイル・タイプでは、このパラメーターの値が NULL の場合、デフォルトで現行ディレクトリーになります。

piOutputNodes

入力。ロード出力パーティションのリスト。 NULL は、ターゲット表が定義されたすべてのノードを示します。

piPartitioningNodes

入力。パーティション・ノードのリスト。 NULL はデフォルトを示します。デフォルトを決定する方法については、「データ移動ユーティリティー・ガイドおよびリファレンス」の『Load コマンド』を参照してください。

piMode

入力。パーティション・データベースのロード・モードを指定します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2LOAD_PARTITION_AND_LOAD

データは (多くの場合は並列で) パーティション化され、それぞれ対応するデータベース・パーティションに同時にロードされます。

DB2LOAD_PARTITION_ONLY

データは (多くの場合は並列で) パーティション化され、それぞれのロード・パーティションの指定したファイルに出力が書き込まれます。SQL_CURSOR 以外のファイル・タイプに関して、各パーティション上の出力ファイルの名前の形式は filename.xxx となり、ここで、filename は、*piSourceList* で指定された最初の入力ファイルの名前で、xxx はパーティションの番号です。SQL_CURSOR ファイル・タイプの場合、各パーティション上の出力ファイルの名前は、*piPartFileLocation* パラメーターによって判別されます。各パーティション上のパーティション・ファイルの位置の指定方法については、*piPartFileLocation* パラメーターを参照してください。

注: このモードは CLI LOAD には使用できません。

DB2LOAD_LOAD_ONLY

データはすでにパーティション化されているものとします。この場合はパーティション・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・パーティションに同時にロードされます。

SQL_CURSOR 以外のファイル・タイプの場合、各パーティションの入力ファイル名のフォーマットは filename.xxx となり、ここで、filename は *piSourceList* で指定された最初のファイルの名前で、xxx は 3 桁のパーティション番号です。SQL_CURSOR ファイル・タイプの場合、各パーティション上の入力ファイルの名前は *piPartFileLocation* パラメーターによって判別されます。各パーティション上のパーティション・ファイルの位置の指定方法については、*piPartFileLocation* パラメーターを参照してください。

注: このモードは、リモート・クライアント上にあるデータ・ファイルのロード時に使用したり、または CLI LOAD には使用できません。

DB2LOAD_LOAD_ONLY_VERIFY_PART

データはすでにパーティション化されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。パーティション化プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時に、各行が正しいパーティション上にあるかがチェックされます。パーティション違反のある行は、ダンプ・ファイル修飾子が指定されている場合、ダンプ・ファイルに置かれます。指定されていない場合、その行は廃棄されます。特定のロード・パーティションにパーティション違反がある場合、そのパーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。各パーティションの入力ファイル名のフォーマットは filename.xxx となり、ここで、filename は *piSourceList* で指定された最初のファイルの名前で、xxx は 3 桁のパーティション番号です。

注: このモードは、リモート・クライアント上にあるデータ・ファイルのロード時に使用したり、または CLI LOAD には使用できません。

DB2LOAD_ANALYZE

すべてのデータベース・パーティション間で均等に分散される最適なパーティション化マップが生成されます。

piMaxNumPartAgents

入力。パーティション・エージェントの最大数。 NULL 値はデフォルトを示します。デフォルトは 25 です。

pilsolatePartErrs

入力。ロード操作が、個々のパーティションで発生するエラーに対応する方法を示します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2LOAD_SETUP_ERRS_ONLY

このモードでは、セットアップ時にパーティションで生じるエラー (たとえば、パーティションへのアクセスに関する問題や、パーティションの表スペースまたは表へのアクセスに関する問題) によって、失敗したパーティションではロード操作が停止してしまいますが、残りのパーティションでは操作が続行されます。データのロード中にパーティションで生じるエラーによって、全操作が失敗し、各パーティションの最後の整合点にロールバックされます。

DB2LOAD_LOAD_ERRS_ONLY

このモードでは、セットアップ時にパーティションで生じるエラーによって、ロード操作全体が失敗します。データのロード中にエラーが生じた場合、エラーのあるパーティションは最後の整合点にロールバックされます。ロード操作は、失敗が生じるまで、またはすべてのデータがロードされるまで、残りのパーティションで続行します。すべてのデータがロードされたパーティションでは、ロード操作後は、データは可視ではありません。他のパーティションで生じたエラーのため、トランザクションは打ち切られます。すべてのパーティション上のデータは、ロードの再開操作が実行されるまで、不可視のままです。これにより、新たにロードされたデータはパーティション上で可視になります。パーティションでは、ロード操作が完了し、エラーが発生したパーティションでのロード操作が再開されます。

注: *iAccessLevel* が `SQLU_ALLOW_READ_ACCESS` に設定されている場合や、コピー・ターゲットが指定されている場合、このモードは使用できません。

DB2LOAD_SETUP_AND_LOAD_ERRS

このモードでは、セットアップまたはデータのロード時に生じるパーティション・レベルのエラーによって、影響を受けたパーティション上でのみ、処理が停止します。

DB2LOAD_LOAD_ERRS_ONLY モードと同様、データ・ロード中にパーティション・エラーが生じた場合、すべてのパーティション上のデータは、ロードの再開操作が実行されるまで不可視のままです。

注: *iAccessLevel* が `SQLU_ALLOW_READ_ACCESS` に設定されている場合や、コピー・ターゲットが指定されている場合、このモードは使用できません。

DB2LOAD_NO_ISOLATION

ロード操作時にエラーが生じると、トランザクションは打ち切られます。

パラメーターが NULL の場合、*iAccessLevel* が SQLU_ALLOW_READ_ACCESS に設定されない限り、またはコピー・ターゲットが指定されない限り、デフォルトは DB2LOAD_LOAD_ERRS_ONLY になります。設定または指定されている場合、デフォルトは DB2LOAD_NO_ISOLATION です。

piStatusInterval

入力。進行メッセージを生成する前に、ロードするデータの MB 数を指定します。有効な値は、1 から 4000 の範囲の整数です。NULL が指定される場合、デフォルト値の 100 が使用されます。

piPortRange

入力。内部通信用の TCP ポート範囲。NULL の場合、使用されるポート範囲は 6000 から 6063 です。

piCheckTruncation

入力。ロードで入出力時にレコードの切り捨てをチェックします。有効な値は TRUE および FALSE です。NULL の場合、デフォルトは FALSE です。

piMapFileInput

入力。パーティション化マップの入力ファイル名。モードが ANALYZE ではない場合、このパラメーターは NULL に設定する必要があります。モードが ANALYZE の場合、このパラメーターは指定する必要があります。

piMapFileOutput

入力。パーティション化マップの出力ファイル名。piMapFileInput に対する規則は、ここでも同じく適用されます。

piTrace

入力。すべてのデータ変換プロセスのダンプ、およびハッシュ値の出力を検討する必要がある場合、トレースするレコードの数を指定します。NULL の場合、レコード数のデフォルトは 0 です。

piNewline

入力。RECLen ファイル・タイプ修飾子も指定されている場合、ロードで ASC データ・レコードの終端で改行文字をチェックするように強制します。指定可能な値は TRUE および FALSE です。NULL の場合、値のデフォルトは FALSE です。

piDistfile

入力。パーティション分散ファイル名。NULL が指定された場合、値のデフォルトは "DISTFILE" です。

piOmitHeader

入力。DB2LOAD_PARTITION_ONLY モードを使用する場合に、パーティション化マップのヘッダーをパーティション・ファイルに組み込まないことを示します。指定可能な値は TRUE および FALSE です。NULL の場合、デフォルトは FALSE です。

piRunStatDBPartNum

統計を収集するデータベース・パーティションを指定します。デフォルト値は、出力パーティション・リスト内の最初のデータベース・パーティションです。

iHostnameLen

入力。 *piHostname* の長さ (バイト単位)。

iFileTransferLen

入力。 *piFileTransferCmd* の長さ (バイト単位)。

iPartFileLocLen

入力。 *piPartFileLocation* の長さ (バイト単位)。

iMapFileInputLen

入力。 *piMapFileInput* の長さ (バイト単位)。

iMapFileOutputLen

入力。 *piMapFileOutput* の長さ (バイト単位)。

iDistfileLen

入力。 *piDistfile* の長さ (バイト単位)。

piNodeList

入力。ノード番号の配列。

iNumNodes

入力。 *piNodeList* 配列内のノードの数。 0 がデフォルトで、これはターゲット表が定義されているすべてのノードです。

iPortMin

入力。小さいポート番号。

iPortMax

入力。大きいポート番号。

oRowsRdPartAgents

出力。すべてのパーティション・エージェントによって読み取られる行の総数。

oRowsRejPartAgents

出力。すべてのパーティション・エージェントによってリジェクトされる行の総数。

oRowsPartitioned

出力。すべてのパーティション・エージェントによってパーティション分割される行の総数。

poAgentInfoList

出力。パーティション・データベースへのロード操作時には、ロード・エージェント、パーティション・エージェント、事前パーティション・エージェント、ファイル転送コマンド・エージェント、およびファイルへのロード・エージェントといった、ロード処理項目が関係してくる場合があります (これらは「Data Movement Guide」で説明されています)。 *poAgentInfoList* 出力パラメーターの目的は、呼び出し側に、ロード操作に関係した各ロード・エージェントに関する情報を戻すことです。リスト内の各項目には、以下の情報が示されます。

- oAgentType。項目が記述するロード・エージェントの種類を示すタグ。
- oNodeNum。エージェントが実行されたパーティションの数。
- oSqlcode。エージェントの処理の結果の最終 sqlcode。
- oTableState。エージェントが実行されるパーティション上の表の最終状況 (ロード・エージェントに関係するもののみ)。

API を呼び出す前に、このリストにメモリーを割り振るのは、API の呼び出し側の責任です。呼び出し側は、*iMaxAgentInfoEntries* パラメーターにメモリーを割り振った項目の数も示す必要があります。呼び出し側が *poAgentInfoList* を NULL に設定する場合、または *iMaxAgentInfoEntries* を 0 に設定する場合、ロード・エージェントに関する情報は戻されません。

iMaxAgentInfoEntries

入力。 *poAgentInfoList* 用にユーザーが割り振ったエージェント情報の項目の最大数。一般に、このパラメーターは、ロード操作に関係したパーティション数の 3 倍の数に設定すれば十分です。

oNumAgentInfoEntries

出力。ロード操作によって生成されたエージェント情報の項目の実際の数。*iMaxAgentInfoEntries* が *oNumAgentInfoEntries* の値以上である場合に限り、この項目数は *poAgentInfoList* パラメーターでユーザーに戻されます。*iMaxAgentInfoEntries* が *oNumAgentInfoEntries* より小さい場合、*poAgentInfoList* に戻される項目数は *iMaxAgentInfoEntries* と等しくなります。

oSqlcode

出力。エージェントの処理の結果の最終 sqlcode。

oTableState

出力。この出力パラメーターの目的は、ロード操作後に、表のいかなる状態も報告しないことです。その目的は、ロード処理中に表に何が起きたかについての一般情報を呼び出し側に提供するために、発生し得る表の状況の、小さなサブセットだけを報告することです。この値は、ロード・エージェントにのみ関係があります。以下の値を指定することができます。

DB2LOADQUERY_NORMAL

ロードがパーティションで正常に完了し、表が LOAD IN PROGRESS (または LOAD PENDING) 状態ではなくなったことを示します。この場合、制約事項をさらに処理する必要があるために、表が引き続きチェック・ペンディング状態であることがありますが、これは正常な状態なので報告はされません。

DB2LOADQUERY_UNCHANGED

エラーが原因でロード・ジョブが処理を打ち切ったが、db2Load を呼び出す前の状態がどのようなものであっても、パーティション上の表の状態はまだ変更されていないことを示します。ロードの再始動、またはそのようなパーティション上での操作の終了を実行する必要はありません。

DB2LOADQUERY_LOADPENDING

処理中にロード・ジョブが打ち切られたが、パーティション上の表

は LOAD PENDING 状態のままであることを示します。これは、パーティションでのロード・ジョブを、終了または再始動する必要があることを意味しています。

oNodeNum

出力。エージェントが実行されたパーティションの数。

oAgentType

出力。エージェント・タイプ。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2LOAD_LOAD_AGENT

DB2LOAD_PARTITIONING_AGENT

DB2LOAD_PRE_PARTITIONING_AGENT

DB2LOAD_FILE_TRANSFER_AGENT

DB2LOAD_LOAD_TO_FILE_AGENT

使用上の注意:

データは、入力ファイル内に並んでいる順序でロードされます。特定の順序にしたい場合には、ロードが試行される前にデータをソートしてください。

ロード・ユーティリティーは、既存の定義に基づいて索引を作成します。ユニーク・キーの重複を処理するのに、例外表が使用されます。ユーティリティーは、参照保全を強制したり、制約検査を実行したり、ロードする表に従属するサマリー表を更新したりすることはありません。参照制約またはチェック制約を組み込まれた表は、チェック・ペンディング状態になります。REFRESH IMMEDIATE として定義されているサマリー表、およびロードする表に依存するサマリー表は、チェック・ペンディング状態になります。表のチェック・ペンディング状態を解除するには、SET INTEGRITY ステートメントを発行してください。ロード操作は、複製されたサマリー表では実行できません。

クラスタリング索引の場合、ロードする前に、データをクラスタリング索引でソートする必要があります。マルチディメンション・クラスターされた (MDC) 表にロードする場合は、データをソートする必要はありません。

DB2 Data Links Managerについての考慮事項

各 DATALINK 列ごとに、括弧内にそれぞれ 1 つの列を指定できます。それぞれの列の指定は、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 情報は、DL_URL_REPLACE_PREFIX、または DL_URL_DEFAULT_PREFIX の指定のいずれかになります。

DATALINK 列指定の数は、表で定義されている DATALINK 列の数と同じだけ指定できます。指定の順序は、挿入列リストの中での DATALINK 列の順序 (挿入列リストが INSERT INTO (insert-column, ...) で指定されている場合) か、または表定義内での順序 (insert-column が指定されていない場合) に従います。

たとえば、表に列 C1、C2、C3、C4、および C5 があり、そのうち C2 と C5 だけが DATALINK 型であり、挿入列 (insert-column) リストが (C1、C5、C3、C2) である場合、2 つの DATALINK 列を指定する必要があります。最初の列の指定は C5

用であり、2 番目の列の指定は C2 用です。挿入列リストを指定しない場合、最初の列の指定は C2 用になり、2 番目の列の指定は C5 用になります。

複数の DATALINK 列があり、一部の列では特別な指定が必要ではない場合、列の指定には、指定の順序をはっきりと示すために、少なくとも括弧を使用する必要があります。どの列にも指定が行われない場合は、空の括弧のリスト全体を除くことができます。したがって、デフォルトで十分な場合には、DATALINK を指定する必要はありません。

FILE LINK CONTROL で定義されている DATALINK 列を備えた表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呼び出す前に、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 として定義されている場合、これらのステップは必要ありません。)

1. DATALINK 列の値によって参照されるデータ・リンク・サーバーに、DB2 Data Links Manager がインストールされていることを確認する。
2. データベースが DB2 Data Links Manager に登録されていることを確認する。
3. DATALINK 値として挿入されるすべてのファイルを、適切なデータ・リンク・サーバーにコピーする。
4. データ・リンク・サーバー上の DB2 Data Links Manager に接頭部名を定義する。
5. (ロードする) DATALINK データによって参照されるデータ・リンク・サーバーを、DB2 Data Links Manager 構成ファイルに登録します。

ロード・ユーティリティーの実行中に、DB2 とデータ・リンク・サーバー間の接続が失敗し、ロード操作も失敗してしまう場合があります。その場合には、以下のようしてください。

1. データ・リンク・サーバーおよび DB2 Data Links Manager を開始する。
2. ロード再開操作を起動する。

ロード操作中に失敗したリンクはデータ保全性違反と見なされ、ユニーク索引違反と同じ方法で処理されます。そのため、1 つ以上の DATALINK 列の入った表をロードする場合の特別な例外が定義されています。

入力ファイル内での DATALINK 情報の表示

LINKTYPE (現在のところ URL のみサポート) は、DATALINK 情報の一部として指定されていません。LINKTYPE は、LOAD または IMPORT コマンドで指定され、PC/IXF タイプの入力ファイルの場合は、適切な列記述子レコードで指定されます。

URL LINKTYPE の DATALINK 情報の構文は、以下のとおりです。

```

┌──────────┴──────────┐
┌──urlname──┐┌──dl_delimiter──comment──┐

```

urlname と *comment* はいずれもオプションです。どちらも省略した場合、NULL 値が代入されます。

urlname

この URL 名は有効な URL 構文に適合していなければなりません。

注:

1. 現在のところ、「http」、「file」、および「unc」がスキーマ名として許可されています。
2. URL 名の接頭部 (スキーマ、ホスト、およびポート) はオプションです。接頭部がない場合は、ロード・ユーティリティまたはインポート・ユーティリティの `DL_URL_DEFAULT_PREFIX` または `DL_URL_REPLACE_PREFIX` 指定の接頭部が使われます。そのどちらも指定されていない場合、デフォルトの接頭部として "file://localhost" が使われます。したがって、ローカル・ファイルの場合、LOAD または IMPORT コマンド内で DATALINK 列を指定せずに、絶対パス名で指定したファイル名を URL 名として入力することができます。
3. 接頭部が URL 名に付けられている場合も、ロードまたはインポート操作時には、`DL_URL_REPLACE_PREFIX` で指定した異なる接頭部名によってオーバーライドされます。
4. (`DL_URL_SUFFIX` を指定した場合、それを追加した後の) 「path」は、リモート・サーバーにあるリモート・ファイルの絶対パス名です。相対パス名は使用できません。HTTP サーバーのデフォルトのパス接頭部は使用されません。

dl_delimiter

区切り付き ASCII (DEL) ファイル形式の場合、`dl_del` 修飾子で指定した文字、あるいは LOAD または IMPORT コマンドのデフォルトの文字。区切りなし ASCII (ASC) ファイル形式の場合、これを文字順序 ¥; (円記号とそれに続くセミコロン) に対応させる必要があります。空白文字 (ブランクやタブなど) は、このパラメーターに指定した値の前後に置くことができます。

comment

DATALINK 値のコメント部分。区切り付き ASCII (DEL) ファイル形式で指定する場合、*comment* テキストは、文字ストリング区切り文字で囲む必要があります。文字ストリング区切り文字は、デフォルトでは二重引用符 (") です。この文字ストリング区切り文字は、LOAD または IMPORT コマンドで MODIFIED BY *filetype-mod* を指定することによりオーバーライドできます。

コメントを指定しない場合、このコメントはデフォルトでは長さがゼロのストリングになります。

次に示すのは、区切り付き ASCII (DEL) ファイル形式での DATALINK データの例です。

- `http://www.almaden.ibm.com:80/mrep/intro.mpeg; "Intro Movie"`

これは、以下の部分とともに格納されます。

- スキーマ = http
- サーバー = www.almaden.ibm.com
- パス = /mrep/intro.mpeg
- 注釈 = "Intro Movie"

- `file://narang/u/narang; "InderPal's Home Page"`

これは、以下の部分とともに格納されます。

- スキーマ = file
- サーバー = narang
- パス = /u/narang
- 注釈 = "InderPal's Home Page"

次に示すのは、区切りなし ASCII (ASC) ファイル形式での DATALINK データの例です。

- http://www.almaden.ibm.com:80/mrep/intro.mpeg%;Intro Movie

これは、以下の部分とともに格納されます。

- スキーマ = http
- サーバー = www.almaden.ibm.com
- パス = /mrep/intro.mpeg
- 注釈 = "Intro Movie"

- file://narang/u/narang%; InderPal's Home Page

これは、以下の部分とともに格納されます。

- スキーマ = file
- サーバー = narang
- パス = /u/narang
- 注釈 = "InderPal's Home Page"

以下に、DATALINK データの例を示します。列のロードまたはインポート指定が DL_URL_REPLACE_PREFIX ("http://qso") であるとしています。

- http://www.almaden.ibm.com/mrep/intro.mpeg

これは、以下の部分とともに格納されます。

- スキーマ = http
- サーバー = qso
- パス = /mrep/intro.mpeg
- 注釈 = NULL スtring

- /u/me/myfile.ps

これは、以下の部分とともに格納されます。

- スキーマ = http
- サーバー = qso
- パス = /u/me/myfile.ps
- 注釈 = NULL スtring

関連資料:

- 「管理 API リファレンス」の『sqluvqdp - 表の表スペースの静止』
- 164 ページの『db2LoadQuery - ロードの照会』
- 「管理 API リファレンス」の『SQLDCOL』

db2Load - ロード

- 「管理 API リファレンス」の『SQLU-MEDIA-LIST』
- 14 ページの『db2Export - エクスポート』
- 55 ページの『db2Import - インポート』
- 「管理 API リファレンス」の『db2DatabaseQuiesce - データベースの静止』
- 「管理 API リファレンス」の『db2InstanceQuiesce - インスタンスの静止』
- 168 ページの『ロード用のファイル・タイプ修飾子』
- 249 ページの『データ移動での区切り文字の制限』

関連サンプル:

- 『dtformat.sqc -- Load and import data format extensions (C)』
- 『tbload.sqc -- How to load into a partitioned database (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.sqC -- How to move table data (C++)』

db2LoadQuery - ロードの照会

処理中のロード操作の状況をチェックします。

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: db2LoadQuery */
/* ... */
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;
    db2UInt32 oTableState;
} db2LoadQueryOutputStruct;
/* ... */

```

汎用 API 構文:

```

/* File: db2ApiDf.h */
/* API: db2gLoadQuery */
/* ... */
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;
    db2UInt32 oTableState;
} db2LoadQueryOutputStruct;
/* ... */

```

API パラメーター:

versionNumber

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

pParmStruct

入力。 *db2LoadQueryStruct* 構造を指すポインター。

pSqlca

出力。 *sqlca* 構造へのポインター。

iStringType

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

DB2LOADQUERY_TABLENAME

db2LoadQuery API が使用する表の名前を指定します。

iStringLen

入力。 *piString* の長さ (バイト単位) を指定します。

piString

入力。 *iStringType* 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) は、以下のとおりです。

DB2LOADQUERY_SHOW_ALL_MSGS

すべてのロード・メッセージを戻す。

DB2LOADQUERY_SHOW_NO_MSGS

ロード・メッセージを戻さない。

DB2LOADQUERY_SHOW_NEW_MSGS

この API を最後に呼び出した後で生成されたメッセージだけを戻す。

poOutputStruct

出力。ロード・サマリー情報の入った、*db2LoadQueryOutputStruct* 構造を指すポインター。サマリーが必要でない場合は、NULL に設定してください。

iLocalMessageFileLen

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

piLocalMessageFile

入力。出力メッセージ用に使用されるローカル・ファイル名を指定します。

oRowsRead

出力。ロード・ユーティリティーがこれまでに読み取ったレコードの数を示します。

oRowsSkipped

出力。ロード操作が開始される前にスキップされたレコードの数を示します。

oRowsCommitted

出力。これまでにターゲット表にコミットされた行数を示します。

oRowsLoaded

出力。これまでにターゲット表にロードされた行の数を示します。

oRowsRejected

出力。これまでにターゲット表からリジェクトされた行数を示します。

oRowsDeleted

出力。これまでにターゲット表から (削除フェーズで) 削除された行数を示します。

oCurrentIndex

出力。現在 (作成フェーズ時に) 作成中の索引を示します。

oCurrentMPPNode

出力。照会されるデータベース・パーティション・サーバーを示します (パーティション・データベース環境モードのみ)。

oLoadRestarted

出力。照会中のロード操作がロード再始動操作である場合に値が TRUE になるフラグを示します。

oWhichPhase

出力。照会中のロード操作の現在のフェーズを示します。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2LOADQUERY_LOAD_PHASE

ロード・フェーズ。

DB2LOADQUERY_BUILD_PHASE

作成フェーズ。

DB2LOADQUERY_DELETE_PHASE

削除フェーズ。

oNumTotalIndexes

出力。(作成フェーズで) 作成する索引の合計数を示します。

oWarningCount

出力。これまでに戻された警告の合計数を示します。

oTableState

出力。表の状態。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2LOADQUERY_NORMAL

表の状態は表には影響を及ぼしません。

DB2LOADQUERY_CHECK_PENDING

表には制約事項があり、その制約事項を検証する必要があります。
SET INTEGRITY コマンドを使用して、表を
DB2LOADQUERY_CHECK_PENDING 状態から解放してください。
制約事項のある表でロードが開始されると、ロード・ユーティリティーは表を DB2LOADQUERY_CHECK_PENDING 状態にします。

DB2LOADQUERY_LOAD_IN_PROGRESS

この表は現在ロードが進行中です。

DB2LOADQUERY_LOAD_PENDING

この表でロードがアクティブでしたが、ロードがコミットする前に打ち切られました。表を DB2LOADQUERY_LOAD_PENDING 状態から解放するために、ロードの終了、ロードの再開、またはロードの置換を発行してください。

DB2LOADQUERY_READ_ACCESS

表のデータは読み取りアクセス照会に使用可能です。
DB2LOADQUERY_READ_ACCESS オプションを使用してロードし、表を読み取り専用状態にします。

DB2LOADQUERY_NOTAVAILABLE

表は使用不可です。表は単にドロップされたか、またはバックアップからリストアされた可能性があります。リカバリー不能なロードによるロールフォワードによって、表は利用不能状態になります。

DB2LOADQUERY_NO_LOAD_RESTART

表は部分的にロードされた状態で、ロードの再開は許可されません。さらにこの表はロード・ペンディング状態です。ロードの終了またはロードの置換を発行し、表をロード再開不能状態から解放します。ロールフォワード操作中、表は DB2LOADQUERY_NO_LOAD_RESTART 状態にすることができます。ロード操作の終了前の時点までロールフォワードした場合、または打ち切られたロード操作を介してロールフォワードするが、ロード終了操作またはロード再開操作の終了時点までロールフォワードしない場合、この状態が発生します。

DB2LOADQUERY_TYPE1_INDEXES

現在、表はタイプ 1 索引を使用しています。この索引に対して REORG ユーティリティーを使用する場合、この索引は、 CONVERT オプションを使用してタイプ 2 に変換できます。

使用上の注意:

この API は、 *piString* で指定された表に対するロード操作の状況を読み取り、 *pLocalMsgFileName* で指定されたファイルに状況を書き込みます。

関連概念:

- 209 ページの『ロード照会コマンドを使用したパーティション・データベース・ロードのモニター』

関連資料:

- 「管理 API リファレンス」の『SQLCA』

関連サンプル:

- 『loadqry.sqb -- Query the current status of a load (MF COBOL)』
- 『tbload.sqc -- How to load into a partitioned database (C)』
- 『tbmove.sqc -- How to move table data (C)』
- 『tbmove.sqC -- How to move table data (C++)』

ロード用のファイル・タイプ修飾子

表 10. ロードで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット

修飾子	説明
anyorder	この修飾子は、 <i>cpu_parallelism</i> パラメーターと共に使用されます。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。そのため、SMP システムでは、パフォーマンスがかなり向上します。 <i>cpu_parallelism</i> の値が 1 である場合、このオプションは無視されます。 SAVECOUNT > 0 の場合、整合点後のクラッシュ・リカバリーでは、データを順番にロードする必要があるため、このオプションはサポートされません。

表 10. ロードで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット (続き)

修飾子	説明
generatedignore	この修飾子は、ロード・ユーティリティに、すべての生成列のデータはデータ・ファイルに存在するが、それらを見捨てるべきことを知らせます。この結果、すべての生成列の値はユーティリティによって生成されます。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 (NULL も入っていない) ものを見なします。この結果、すべての生成列の値はユーティリティによって生成されます。この修飾子は、generatedignore または generatedoverride 修飾子とともに使用することはできません。
generatedoverride	<p>この修飾子は、(こうした列のタイプの通常の規則に反して) 表内のすべての生成列で、ユーザーのデータを受け入れるようにロード・ユーティリティに指示します。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、または ROLLFORWARD DATABASE コマンドで RECOVER DROPPED TABLE オプションを使用してリカバリー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、NULL 不可の生成列でデータまたは NULL データの入っていない行はリジェクトされます (SQL3116W)。</p> <p>注: この修飾子が使用される場合、表はチェック・ペンディング状態になります。ユーザー提供の値をチェックせずに表をチェック・ペンディング状態から解放するには、ロード操作後に以下のコマンドを発行します。</p> <pre>SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED</pre> <p>表のチェック・ペンディング状態を解除し、ユーザー定義の値のチェックを強制するには、ロード操作の後以下のコマンドを発行してください。</p> <pre>SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.</pre> <p>この修飾子は、generatedmissing または generatedignore 修飾子と共に使用することはできません。</p>
identityignore	この修飾子はロード・ユーティリティに対して、ID 列のデータがデータ・ファイル内に存在するが、それらのデータは無視すべきものであることを通知します。この結果として、すべて ID 値はこのユーティリティによって生成されます。この動作は、GENERATED ALWAYS および GENERATED BY DEFAULT ID 列のどちらの場合も同じです。つまり、GENERATED ALWAYS 列の場合には、リジェクト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	この修飾子を指定すると、ユーティリティは、ID 列のデータが入力データ・ファイルに入っていない (NULL も入っていない) ものを見なし、行ごとに値を生成します。この動作は、GENERATED ALWAYS および GENERATED BY DEFAULT ID 列のどちらの場合も同じです。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。

表 10. ロードで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット (続き)

修飾子	説明
identityoverride	<p>この修飾子は、GENERATED ALWAYS として定義した ID 列が、ロードする表に存在している場合にのみ使用するべきです。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 NULL データを受け入れる (これらのタイプの ID 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、または ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合です。この修飾子を使用すると、データが入っていない行や ID 列に対する NULL データはすべてリジェクトされます (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。</p> <p>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 ID 列内の値の固有性の保守または検証を行いません。</p>
indexfreespace= <i>x</i>	<p><i>x</i> は 0 から 99 の整数です。その値は、各索引ページの中で索引再作成ロード時のフリー・スペースとして残しておく部分の割合を示すパーセントとして解釈されます。ロードに INDEXING MODE INCREMENTAL を指定すると、このオプションは無視されます。ページの最初の項目は、制限なしで追加されます。それより後の項目は、フリー・スペースのパーセントしきい値内である場合に追加されます。デフォルト値は、CREATE INDEX の実行時に使用した値です。</p> <p>この値は、CREATE INDEX ステートメントに指定された PCTFREE 値よりも優先して使用され、レジストリー変数 DB2 INDEX FREE は indexfreespace よりも優先して使用されます。 indexfreespace オプションは、索引のリーフ・ページにのみ影響を与えます。</p>
lobsinfile	<p><i>lob-path</i> には、LOB データの入った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 列に LOB データが入っているファイルの名前が入っています。</p> <p>ファイル・タイプが CURSOR の場合、このオプションはサポートされていません。</p> <p>LOBS FROM 文節は、“lobsinfile” 修飾子が使用されているときの、LOB ファイルの場所を指定します。LOBS FROM 文節には、“lobsinfile” 修飾子のコンテキスト以外の意味はありません。LOBS FROM 文節は、データのロード中に、LOAD ユーティリティーに LOB ファイルを検索するためのパスのリストを送ります。</p> <p>各パスには、データ・ファイル内で LOB ロケーション指定子 (LLS) によって示される 1 つ以上の LOB の入った、少なくとも 1 つのファイルが組み込まれます。LLS は、LOB ファイル・パスに保管されるファイル内の LOB のロケーションのストリング表現です。LLS のフォーマットは、<i>filename.ext.nnn.mmm/</i> です。<i>filename.ext</i> は LOB の入ったファイルの名前、<i>nnn</i> はファイル内の LOB のオフセット (バイト単位)、<i>mmm</i> は LOB の長さ (バイト単位) を表します。たとえば、ストリング db2exp.001.123.456/ がデータ・ファイルに保存される場合、LOB はファイル db2exp.001 のオフセット 123 に位置し、456 バイト長です。</p> <p>NULL LOB を指定するには、サイズに -1 と入力します。サイズを 0 と指定すると、長さが 0 の LOB として扱われます。長さが -1 の NULL LOB の場合、オフセットとファイル名は無視されます。たとえば、NULL LOB の LLS は db2exp.001.7.-1/ です。</p>

表 10. ロードで有効なファイル・タイプ修飾子: すべてのファイル・フォーマット (続き)

修飾子	説明
noheader	<p>ヘッダー検査コードをスキップします (単一パーティション・データベースのパーティション・グループに存在する表へのロード操作にのみ適用します)。</p> <p>オートローダー・ユーティリティーは、複数パーティションのデータベース・パーティション・グループの表にデータを提供している各ファイルに、ヘッダーを書き込みます。単一パーティションのデータベース・パーティション・グループに存在する表に対してデフォルトの MPP ロード (モード PARTITION_AND_LOAD) が使用される場合、ファイルにはヘッダーが組み込まれないと想定されます。したがって、noheader 修飾子は必要ありません。 LOAD_ONLY モードが使用される場合、ファイルにはヘッダーが付いていると想定されます。ヘッダーが組み込まれないファイルを使って操作を実行したい場合にのみ、 noheader 修飾子を使用する必要があります。</p>
norowwarnings	リジェクトされた行についての警告をすべて抑制します。
pagefreespace=x	<p>x は 0 から 100 の整数です。この値は、各データ・ページ内でフリー・スペースとして残される部分のパーセンテージとして解釈されます。最小行サイズのため、指定した値が無効である場合 (たとえば、最も小さい行の大きさが 3 000 バイトで、x の値が 50 である場合)、その行は新しいページに置かれます。値 100 が指定された場合には、各行がそれぞれ新しいページに置かれます。</p> <p>注: 表の PCTFREE 値は、ページごとに指定されたフリー・スペースの量を決定します。ロード操作の pagefreespace 値、または表の PCTFREE 値が設定されていない場合、ユーティリティーは、それぞれのページにできるだけ大きなスペースを割り当てます。 pagefreespace によって設定された値は、表に対して指定されている PCTFREE 値をオーバーライドします。</p>
subtableconvert	単一副表へのロードの場合のみ有効。これを使う場合として典型的な例は、正規の表からデータをエクスポートした後、この修飾子を使ってロード操作を呼び出してそのデータを単一の副表に変換する場合です。
totalfreespace=x	<p>x は 0 以上の整数です。この値は表内の合計ページのうち、表の終わりにフリー・スペースとして追加される部分のパーセンテージと解釈されます。たとえば、x が 20 で、データのロード後に、表に 100 のデータ・ページがある場合、20 の追加の空ページが付加されます。その表のデータ・ページの合計数は 120 になります。データ・ページの総数は、表の索引ページの数には影響を与えません。このオプションは、索引オブジェクトには影響を与えません。</p> <p>注: このオプションを指定して 2 つのロードが行われる場合、 2 番目のロードは、最初のロードによって最後に付加された余分のスペースを再利用しません。</p>
usedefaults	<p>ターゲット表の列のソース列が指定されているが、 1 つまたは複数の行インスタンスのデータが入っていない場合は、デフォルト値がロードされます。欠落データの例は次のとおりです。</p> <ul style="list-style-type: none"> • DEL ファイルの場合、列に ".,," が指定された場合 • DEL/ASC/WSF ファイルの場合、列の数が不足している行、または元の指定に対して十分な長さではない行。 <p>このオプションが指定されていない場合、行インスタンスのソース列にデータがないと、以下のいずれかの処理が行われます。</p> <ul style="list-style-type: none"> • その列が NULL 可能な場合は、NULL がロードされます。 • 列が NULL 可能でない場合、ユーティリティーはその行をリジェクトします。

表 11. ロードで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL)

修飾子	説明
codepage= <i>x</i>	<p><i>x</i> 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、文字データ (および文字内で指定された数値データ) は、このコード・ページからデータベースのコード・ページへ変換されます。</p> <p>以下の規則が適用されます。</p> <ul style="list-style-type: none"> • 純粋な DBCS (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字は x00 から x3F の範囲に制限されます。 • 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一致しない場合があります。 • nullindchar には、標準の ASCII セットに組み込む (コード・ポイント x20 から x7F の範囲の) 記号を指定する必要があります。これは、ASCII 記号およびコード・ポイントを示します。EBCDIC データでは、コード・ポイントが異なるとしても、対応する記号を使用できます。 <p>ファイル・タイプが CURSOR の場合、このオプションはサポートされていません。</p>
dateformat=" <i>x</i> "	<p><i>x</i> は、ソース・ファイルの日付のフォーマットです。¹ 有効な日付エレメントは次のとおりです。</p> <p>YYYY - 年 (0000 から 9999 の範囲の 4 桁の数字) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。M と相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の 2 桁の数字。D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の 3 桁の数字。 他の日または月エレメントとは相互に排他的)</p> <p>デフォルト値の 1 が、指定されない各エレメントに割り当てられます。日付形式の例を以下に示します。</p> <p>"D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"</p>

表 11. ロードで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
dumpfile = <i>x</i>	<p><i>x</i> は、リジェクトされた行を書き込む例外ファイルの (サーバー・データベース・パーティションによる) 完全修飾名です。1 レコードにつき、最大で 32 KB のデータが書き込まれます。以下に、ダンプ・ファイルの指定方法の例を示します。</p> <pre>db2 load from data of del modified by dumpfile = /u/user/filename insert into table_name</pre> <p>ファイルは、インスタンスの所有者によって作成されて所有されます。デフォルトのファイル許可をオーバーライドするには、dumpfileaccessall ファイル・タイプ修飾子を使用します。</p> <p>注:</p> <ol style="list-style-type: none"> パーティション・データベース環境の場合、パスはロードを実行するデータベース・パーティションにローカルなものでなければなりません。それによって、並行して実行される複数のロード操作が同じファイルに書き込むことを防ぐことができます。 ファイルの内容は、非同期バッファ・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り割り込まれたりした場合、ディスクにコミットされたレコード数を正確に判別できないので、LOAD RESTART 後も一貫性が保たれるとは限りません。ファイルが完全であるとされるのは、1 回のパスの中で開始して完了するロード操作の場合だけです。 この修飾子では、ファイル拡張子が複数あるファイル名はサポートされません。たとえば、以下のように指定します。 <pre>dumpfile = /home/svtdbm6/DUMP.FILE</pre> <p>ロード・ユーティリティーでは、上のファイル名は受け入れられます。</p> <pre>dumpfile = /home/svtdbm6/DUMP.LOAD.FILE</pre> <p>しかし、このファイル名は受け入れられません。</p>
dumpfileaccessall = <i>x</i>	<p>ダンプ・ファイルの作成時に「OTHERS」への読み取りアクセスを認可します。</p> <p>このファイル・タイプ修飾子が有効なのは、以下の場合のみです。</p> <ol style="list-style-type: none"> dumpfile ファイル・タイプ修飾子と一緒に使用された場合。 ロード・ターゲット表に対してユーザーが SELECT 特権をもっている場合。 UNIX ベースのオペレーティング・システムに置かれている DB2 サーバー・データベース・パーティション上で発行された場合。

表 11. ロードで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
fastparse	<p>ユーザー提供の列値に対して簡略化された構文チェックが実行され、パフォーマンスが向上します。このオプションの下でロードした表は、体系的な正確性が確保されて、セグメント化違反またはトラップを防ぐための十分なデータ・チェックが必ずユーティリティで実行されます。正しい形式のデータが正しくロードされます。</p> <p>たとえば、ASC ファイル内の整数列のフィールド項目として 123qwr4 の値が検出された場合、この値は有効な数値を表すものではないので、ロード・ユーティリティは通常、構文エラーのフラグを付けます。fastparse を指定した場合、構文エラーは検出されず、整数フィールドに任意の数値がロードされます。この修飾子を使用する場合は、正しい形式のデータだけを使用するように注意してください。ASCII データでこのオプションを使用した場合のパフォーマンスの向上は絶大です。</p> <p>ファイル・タイプが CURSOR または IXF の場合、このオプションはサポートされていません。</p>
implieddecimal	<p>暗黙指定されている小数点の位置が列定義によって決定され、値の終わりにあるとは見なされなくなります。たとえば、値 12345 は、12345.00 ではなく、123.45 として DECIMAL(8,2) 列にロードされます。</p> <p>この修飾子は、packeddecimal 修飾子と共に使用することはできません。</p>
timeformat="x"	<p>x はソース・ファイル内の時刻のフォーマットです。¹ 有効な時刻エレメントは以下のとおりです。</p> <ul style="list-style-type: none"> H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M と相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) <p>デフォルト値の 0 が、指定されない各エレメントに割り当てられます。時刻フォーマットの例を以下に示します。</p> <p>"HH:MM:SS" "HH.MM TT" "SSSSS"</p>

表 11. ロードで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
timestampformat="x"	x はソース・ファイル内のタイム・スタンプのフォーマットです。 ¹ 有効なタイム・スタンプ・エレメントは以下のとおりです。
	YYYY - 年 (0000 から 9999 の範囲の 4 桁の数字) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (01 から 12 の 2 桁の数。M と MMM とは相互に排他的) MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の 2 桁の数字。D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の 3 桁の数字。 他の日または月のエレメントとは相互に排他的) H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 1 桁または 2 桁の数。) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数字。H と相互に排他的) M - 分 (0 から 59 の 1 桁または 2 桁の数字。) MM - 分 (0 から 59 の範囲の 2 桁の数。M (分) とは相互に排他的) S - 秒 (0 から 59 の 1 桁または 2 桁の数字。) SS - 秒 (0 から 59 の範囲の 2 桁の数。S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数。(00000 から 86399 の 5 桁の数。 他の時刻エレメントとは相互に排他的) UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的) UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999999 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数字。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数字。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM)
	YYYY、M、MM、D、DD、または DDD エレメントが指定されていない場合、デフォルト値として 1 が割り当てられます。値が指定されていない MMM エレメントには、デフォルト値の「Jan」が割り当てられます。他のエレメントが指定されていない場合には、デフォルト値として 0 が割り当てられます。タイム・スタンプ・フォーマットの例を以下に示します。 "YYYY/MM/DD HH:MM:SS.UUUUUU"
	MMM エレメントの有効な値は、「jan」、「feb」、「mar」、「apr」、「may」、「jun」、「jul」、「aug」、「sep」、「oct」、「nov」、および「dec」です。これらの値は大文字小文字の区別をします。 次の例では、ユーザー定義の日時形式を指示するデータを、schedule という表にインポートする方法を示します。 <pre> db2 import from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule </pre>

表 11. ロードで有効なファイル・タイプ修飾子: ASCII ファイル・フォーマット (ASC/DEL) (続き)

修飾子	説明
noeofchar	オプションのファイル終了文字 'x'1A' は、ファイルの終わりとして認識されません。それが普通の文字であるものとして処理は継続されます。
usegraphiccodepage	<p>usegraphiccodepage が指定された場合、GRAPHIC または 2 バイト文字ラージ・オブジェクト (DBCLOB) データ・フィールドにロードされるデータは、GRAPHIC コード・ページであると見なされます。データの残りは、文字コード・ページであると見なされます。GRAPHIC コード・ページは、文字コード・ページと関連付けられます。LOAD は、指定されている場合の codepage 修飾子、または codepage 修飾子が指定されていない場合は、データベースのコード・ページを通じて、文字コード・ページを決定します。</p> <p>この修飾子は、リカバリーされている表中に GRAPHIC データがある場合にのみ、表リカバリーのドロップによって生成された区切りデータ・ファイルとともに使用される必要があります。</p> <p>制約事項</p> <p>usegraphiccodepage 修飾子は、EXPORT ユーティリティーで作成された DEL または ASC ファイルで指定することはできません。usegraphiccodepage 修飾子はまた、ファイル内の 2 バイト文字ラージ・オブジェクト (DBCLOB) には無視されます。</p>

表 12. ロードで有効なファイル・タイプ修飾子: ASC ファイル・フォーマット (区切り文字で区切られていない ASCII)

修飾子	説明
binarynumerics	<p>数値データ (DECIMAL 以外) は、文字表記ではなく、バイナリー形式でなければなりません。これによって、コストの大きい変換操作を避けることができます。</p> <p>このオプションがサポートされるのは、定位置 ASC において、reclen オプションによって固定長レコードが指定されている場合だけです。noeofchar オプションが前提となります。</p> <p>以下の規則が適用されます。</p> <ul style="list-style-type: none"> • BIGINT、INTEGER、および SMALLINT を除き、データ・タイプ間の変換は実行されません。 • データ長は、それぞれのターゲット列定義と一致している必要があります。 • FLOAT は、IEEE 浮動小数点フォーマットでなければなりません。 • ロード・ソース・ファイル中のバイナリー・データは、ロード操作を実行するプラットフォームに関係なく、ビッグ・エンディアンであると見なされます。 <p>注: この修飾子の影響を受ける列のデータに NULL があってはなりません。この修飾子を使用すると、ブランク (通常は NULL と解釈される) は、バイナリー値であると解釈されます。</p>
nochecklengths	nochecklengths を指定した場合は、ソース・データの中にターゲット表の列のサイズを超える列定義がある場合であっても、各行のロードが試みられます。コード・ページ変換によってソース・データが縮小されれば、そのような行であったとしても正常にロードすることができます。たとえば、ソースに 4 バイトの EUC データがある場合、それがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースは半分で済みます。列定義が一致しなくてもソース・データがきちんと収まるのが明らかな場合に、このオプションは特に便利です。

表 12. ロードで有効なファイル・タイプ修飾子: ASC ファイル・フォーマット (区切り文字で区切られていない ASCII) (続き)

修飾子	説明
nullindchar= <i>x</i>	<p><i>x</i> は単一文字です。 NULL 値を示す文字を <i>x</i> に変更します。 <i>x</i> のデフォルト値は Y です。²</p> <p>文字が 1 つの英字である場合を除いて、この修飾子は EBCDIC データ・ファイルで大文字小文字を区別します。たとえば、NULL 標識文字を文字 N に指定した場合、n も NULL 標識と認識されます。</p>
packeddecimal	<p>binarynumerics 修飾子は DECIMAL フィールド・タイプで構成されないため、パック 10 進数データを直接ロードします。</p> <p>このオプションがサポートされるのは、定位置 ASC において、reclen オプションによって固定長レコードが指定されている場合だけです。noeofchar オプションが前提となります。</p> <p>符号ニブル用にサポートされる値は以下のとおりです。</p> <pre> + = 0xC 0xA 0xE 0xF - = 0xD 0xB </pre> <p>この修飾子の影響を受ける列のデータに NULL があってはなりません。この修飾子を使用すると、ブランク (通常は NULL と解釈される) は、バイナリー値であると解釈されます。</p> <p>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・ファイルに入っているバイナリー・データのバイト順はビッグ・エンディアンであることが前提となっています。つまり、この修飾子を Windows オペレーティング・システムで使用する場合も、バイト順を逆にしてはなりません。</p> <p>この修飾子は、implieddecimal 修飾子と共に使用することはできません。</p>
reclen= <i>x</i>	<p><i>x</i> は、32,767 以下の整数です。各行ごとに <i>x</i> 個の文字が読み取られ、行の終わりを示すのに改行文字は使用されません。</p>
striptblanks	<p>データを可変長フィールドにロードする際に、後書きブランク・スペースを切り捨てます。このオプションを指定しないと、ブランク・スペースは保持されます。</p> <p>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。これらは相互に排他的なオプションです。</p> <p>注: このオプションは、廃止された t オプション (後方互換性のためだけにサポートされる) に代わるものです。</p>
striptnulls	<p>データを可変長フィールドにロードする際に、後書き NULL (0x00 文字) を切り捨てます。このオプションを指定しないと、NULL は保持されます。</p> <p>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。これらは相互に排他的なオプションです。</p> <p>注: このオプションは、廃止された padwithzero オプション (後方互換性のためだけにサポートされる) に代わるものです。</p>

db2LoadQuery - ロードの照会

表 12. ロードで有効なファイル・タイプ修飾子: *ASC* ファイル・フォーマット (区切り文字で区切られていない *ASCII*) (続き)

修飾子	説明
zoneddecimal	<p>BINARYNUMERICS 修飾子は DECIMAL フィールド・タイプでは構成されないため、ゾーン 10 進数データをロードします。このオプションがサポートされるのは、定位置 ASC において、RECLEN オプションによって固定長レコードが指定されている場合だけです。NOEOFCHAR オプションが前提となります。</p> <p>ハーフバイト符号値は、以下のいずれかになります。</p> <p>+ = 0xC 0xA 0xE 0xF - = 0xD 0xB</p> <p>サポートされている数値は、0x0 から 0x9 です。</p> <p>サポートされているゾーン値は、0x3 および 0xF です。</p>

表 13. ロードで有効なファイル・タイプ修飾子: *DEL* ファイル・フォーマット (区切り文字で区切られている *ASCII*)

修飾子	説明
chardelx	<p><i>x</i> は単一文字のストリング区切り文字です。デフォルトは二重引用符 (") です。指定した文字は、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。 ²³ 文字ストリング区切り文字として明示的に二重引用符 (") を指定したい場合、以下のように指定します。</p> <p>modified by charde1""</p> <p>単一引用符 (') も、以下のように文字ストリングの区切り文字として指定できます。</p> <p>modified by charde1''</p>
coldelx	<p><i>x</i> 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。指定した文字は、列の終わりを表すために、コンマの代わりに使用されます。 ²³</p>
datesiso	<p>日付形式。すべての日付データ値を ISO フォーマットでロードします。</p>
decplusblank	<p>正符号文字。正の 10 進値の接頭部として、正符号 (+) ではなくブランク・スペースを使用します。デフォルトのアクションでは、10 進値に正符号の接頭部が付けられます。</p>
decptx	<p><i>x</i> は、小数点文字としてピリオドの代わりに使用される単一の置換文字です。デフォルト値はピリオド (.) です。指定した文字は、小数点文字としてピリオドの代わりに使用されます。 ²³</p>

表 13. ロードで有効なファイル・タイプ修飾子: DEL ファイル・フォーマット (区切り文字で区切られている ASCII) (続き)

修飾子	説明
delprioritychar	<p>区切り文字の現在のデフォルト優先順位は、(1) レコード区切り文字、(2) 区切り文字、(3) 列区切り文字です。この修飾子は、区切り文字の優先順位を区切り文字、レコード区切り文字、列区切り文字と逆順にすることにより、以前の優先順位に依存する既存のアプリケーションを保護します。構文は以下のとおりです。</p> <pre>db2 load ... modified by delprioritychar ...</pre> <p>たとえば、以下のような DEL データ・ファイルがあるとします。</p> <pre>"Smith, Joshua",4000,34.98<row delimiter> "Vincent,<row delimiter>, is a manager", 4005,44.37<row delimiter></pre> <p>delprioritychar 修飾子が指定されている場合、このデータ・ファイルには 2 行しかありません。2 番目の <row delimiter> は 2 番目の行の最初のデータ列の一部と解釈されますが、最初と 3 番目の <row delimiter> は実レコードの区切り文字と解釈されます。この修飾子を指定しなかった場合は、このデータ・ファイルは 3 行のままで、各行は <row delimiter> によって区切られます。</p>
dldelx	<p>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。指定した文字は、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 値の中には副値が複数個ある場合があるため、これが必要になります。²³⁴</p> <p>注: 行、列、または文字ストリング区切り文字と同じ文字を x に指定することはできません。</p>
keepblanks	<p>タイプが 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。このオプションを指定しないと、区切り文字で囲まれていないすべての前後のブランクは除去され、表のすべてのブランク・フィールドに NULL が挿入されます。</p> <p>以下の例では、データ・ファイルにある前後のブランクを保存しながら、TABLE1 という表にデータをロードする方法を示します。</p> <pre>db2 load from delfile3 of del modified by keepblanks insert into table1</pre>
nochardel	<p>ロード・ユーティリティは、列区切り文字と列区切り文字の間にあるすべてのバイトが列データの一部であると見なします。区切り文字は、列データの一部として構文解析されます。データが DB2 を使用してエクスポートされている場合は、このオプションを指定しないでください (エクスポート時に nochardel が指定されない限り)。これは、区切り文字を持たないペンダー・データ・ファイルをサポートするために用意されています。不適切に使用すると、データが損失または破壊される場合があります。</p> <p>このオプションを chardelx、delprioritychar または nodoubledel と一緒に指定することはできません。これらのオプションは、相互に排他的です。</p>
nodoubledel	<p>二重区切り文字の認識を抑止します。</p>

表 14. ロードで有効なファイル・タイプ修飾子: IXF ファイル・フォーマット

修飾子	説明
forcein	コード・ページが不一致でもデータを受け入れ、コード・ページ間の変換を抑止するようにユーティリティに指示します。 固定長ターゲット・フィールドに、そのデータが入るだけの十分な大きさがあるかどうかチェックされます。 <code>nochecklengths</code> を指定した場合、そのような検査は実行されず、各行のロードが試みられます。
nochecklengths	<code>nochecklengths</code> を指定した場合は、ソース・データの中にターゲット表の列のサイズを超える列定義がある場合であっても、各行のロードが試みられます。コード・ページ変換によってソース・データが縮小されれば、そのような行であったとしても正常にロードすることができます。たとえば、ソースに 4 バイトの EUC データがある場合、それがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースは半分で済みます。列定義が一致しなくてもソース・データがきちんと収まることが明らかな場合に、このオプションは特に便利です。

注:

1. 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` から `z`、`A` から `Z`、および `0` から `9` を使用することはできません。フィールド区切り文字は、区切り文字、または `DEL` ファイル・フォーマットのフィールド区切り文字と同じであってはなりません。エレメント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。あいまいさが生じうるのは、項目の長さが一定でない `D`、`H`、`M`、または `S` などのエレメント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・フォーマットの場合、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、区別があいまいにならないように注意する必要があります。月のフィールドは、他の日付フィールドと隣接していなければなりません。分フィールドは、他の時刻フィールドに隣接していなければなりません。以下に、いくつかのあいまいなタイム・スタンプ・フォーマットを示します。

"M" (月または分のどちらにもとれる)
 "M:M" (月と分の区別がつかない)
 "M:YYYY:M" (両方とも月と解釈される)
 "S:M:YYYY" (時刻値と日付値の両方に隣接している)

あいまいな場合、ユーティリティはエラー・メッセージを報告し、操作は失敗します。

以下に、明確なタイム・スタンプ・フォーマットを示します。

"M:YYYY" (月)
 "S:M" (分)
 "M:YYYY:S:M" (月....分)
 "M:H:YYYY:M:D" (分....月)

二重引用符や円記号などの文字の前には、エスケープ文字 (たとえば、`¥`) を付けなければなりません。

2. この文字は、ソース・データのコード・ページで指定してください。

文字コード・ポイント (文字記号ではない) は、`xJJ` または `0xJJ` という構文で指定することができます (`JJ` はコード・ポイントの 16 進表記)。たとえば、列区切りとして `#` 文字を指定するには、以下のいずれかを使用します。

```
... modified by coldel# ...
... modified by coldel0x23 ...
... modified by coldelX23 ...
```

- データ移動のための区切り文字の制約事項に、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。
- DATALINK 区切り文字が URL 構文内で有効な文字である場合でも、ロード操作の有効範囲内では、その特別な意味はなくなります。
- サポートされていないファイル・タイプを MODIFIED BY オプションで使用しても、ロード・ユーティリティーは警告を出しません。この場合、ロード操作が失敗し、エラー・コードが戻されます。

表 15. codepage および usegraphiccodepage 使用時の LOAD 動作

codepage=N	usegraphiccodepage	LOAD 動作
なし	なし	ファイル内のすべてのデータは、アプリケーション・コード・ページであると見なされます。
あり	なし	<p>ファイル内のすべてのデータは、コード・ページ N であると見なされます。</p> <p>警告: N が単一バイト・コード・ページの場合、GRAPHIC データをデータベースにロードすると壊れます。</p>
なし	あり	<p>CLIENT オプションの指定があっても、ファイル内の文字データは、データベース・コード・ページであると見なされます。 CLIENT オプションの指定があっても、GRAPHIC データは、データベース GRAPHIC データのコード・ページであると見なされます。</p> <p>データベース・コード・ページが単一バイトの場合は、すべてのデータはデータベース・コード・ページであると見なされます。</p> <p>警告: 単一バイト・データベースに GRAPHIC データをロードすると、壊れます。</p>
あり	あり	<p>文字データは、コード・ページ N であると見なされます。 GRAPHIC データは、N の GRAPHIC コード・ページであると見なされます。</p> <p>N が単一バイトまたは 2 バイト・コード・ページの場合は、すべてのデータは、コード・ページ N であると見なされます。</p> <p>警告: N が単一バイト・コード・ページの場合、GRAPHIC データをデータベースにロードすると壊れます。</p>

関連資料:

- 113 ページの『LOAD』
- 139 ページの『db2Load - ロード』
- 249 ページの『データ移動での区切り文字の制限』

ロード例外表

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入っています。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、`ID` 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 `ID` 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 `NULL` 可能性の属性しか入れることはできません。例外表はユニーク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用されます。ユーティリティーは独特の違反を除いて制約または外部キーの違反のチェックをしません。さらに、`DATALINK` 例外も例外表にキャプチャーされます。

ロード例外表は、ロードされている表の常駐する表スペース、またはその他の表スペースに割り当てることができます。どちらの場合でも、ロード例外表は、ロードされる表と同じデータベース・パーティション・グループに割り当てなければならず、同じパーティション・キーがなければなりません。

ユニーク・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用される仕組みは、ユニーク索引と呼ばれます。主キーはユニーク・キーの特殊な例です。同一の表に 2 つ以上の主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リジェクトされた行は、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が必要な場合には、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する例外表は、`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ユニーク索引があり、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。

関連資料:

- 「*SQL* リファレンス 第 1 巻」の『例外表』

ロード・ダンプ・ファイル

| *dumpfile* 修飾子を指定すると、リジェクトされた行を書き込む例外表の名前とロケ
| ションをロード・ユーティリティーに対して指示できます。パーティション・デ
| ータベース環境での実行時には、パーティション化サブエージェントまたはロー
| ド・サブエージェントによって行がリジェクトされることがあります。そのため、

ダンプ・ファイル名には、サブエージェントのタイプを特定する拡張子と、例外が生成されたパーティション番号が付けられます。たとえば、次のようなダンプ・ファイル値を指定したとします。

```
dumpfile = "/u/username/dumpit"
```

この場合、パーティション 5 でロード・サブエージェントによってリジェクトされた行は、`/u/username/dumpit.load.005` という名前のファイルに保管され、パーティション 2 でロード・サブエージェントによってリジェクトされた行は、`/u/username/dumpit.load.002` という名前のファイルに保管され、パーティション 2 でパーティション化サブエージェントによってリジェクトされた行は、`/u/username/dumpit.part.002` という名前のファイルに保管される、などとなります。

ロード・サブエージェントによって行がリジェクトされた場合に、その行が 32,768 バイト未満の長さであると、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、行の断片 (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。

パーティション化サブエージェントによって行がリジェクトされた場合、レコード・サイズに関係なく、その行全体がダンプ・ファイルにコピーされます。

関連資料:

- 113 ページの『LOAD』

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・クラッシュ・リカバリー、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用されます。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 `db2Load` API の `piTempFilePath` 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。

重要: 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。

関連資料:

- 113 ページの『LOAD』
- 139 ページの『db2Load - ロード』

ロード・ユーティリティのログ・レコード

ユーティリティ管理機能は、ログ・ユーティリティなどのいくつかの DB2® ユーティリティに関連するログ・レコードを生成します。以下のログ・レコードには、ロード操作時の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。
- 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、主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。
- ロード削除終了。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。この削除フェーズは、正常なロード操作のロールフォワード・リカバリー中に繰り返されます。
- ロード・ペンディング・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、通常のトランザクション・コミット・ログ・レコードの代わりに使用されます。

以下のリストは、ロード・ユーティリティが、入力データのサイズに従って作成するログ・レコードを説明します。

- ユーティリティによって、割り振られたり削除されたりするすべての表スペース・エクステントにつき、2 つのログ・レコードが DMS 表スペースに作成されます。
- 消費される ID 値の塊ごとにログ・レコードが作成されます。
- ログ・レコードは、ロード操作の削除フェーズで削除されるデータ行または索引キーごとに作成されます。
- ALLOW READ ACCESS および INDEXING MODE INCREMENTAL オプションを指定してロード操作を実行する際に、索引ツリーの保全性を保守するためのログ・レコードが作成されます。ログ記録されるレコードの数は、完全にログ記録される索引への挿入よりずっと少なくなります。

関連資料:

- 113 ページの『LOAD』
- 139 ページの『db2Load - ロード』

表ロック、表状態、および表スペース状態

大抵の場合、ロード・ユーティリティは、表レベル・ロックを使用して、表へのアクセスを制限します。ロード・ユーティリティは、ロード操作に関係する表スペースを静止することではなく、COPY NO オプションが指定されているロード操作の場合にのみ表スペース状態を使用します。ロックのレベルは、ロード操作が読み取りアクセスを許可するかどうかにより異なります。ALLOW NO ACCESS モードのロード操作は、ロード中に表に対して排他ロック (Z-lock) を使用します。

ALLOW READ ACCESS モードのロード操作は、ロード操作の継続中、共用ロック

(S-lock) を取得して保守します。そして、データがコミットされると、排他ロック (Z-lock) へロックをアップグレードします。

ALLOW READ ACCESS モードのロード操作を開始する前に、ロード・ユーティリティーは、ロード操作前に開始したすべてのアプリケーションが、ターゲット表に対するロックを解放するのを待機します。ロックは持続しないため、ロード操作が打ち切られても存続する表状態により補足されます。それらの状態は LOAD QUERY コマンドを使って調べることができます。LOCK WITH FORCE オプションを使用することにより、ロード・ユーティリティーは、ロードしようとしている表から、競合するロックを保留しているアプリケーションを強制的に除きます。

ALLOW READ ACCESS モードでのロード操作のロックの動作

ロード操作の始めに、ロード・ユーティリティーは表に対する共用ロック (S-lock) を獲得します。これは、データがコミットされるまで、このロックを保留します。共用ロックを使用すると、ロード操作時に、互換性のあるロックを持つアプリケーションが表にアクセスすることができます。たとえば、読み取り専用照会を使用するアプリケーションは表にアクセスできる一方で、表にデータを挿入しようとするアプリケーションは拒否されます。ロード・ユーティリティーが表に対する共用ロックを獲得すると、ロード操作の開始前に表に対するロックを保留するすべてのアプリケーションは、互換性のあるロックを持っている場合にも、それらのロックを解放するのを待機します。ロード・ユーティリティーは、データのコミット時に共用ロックを排他 (Z-lock) にアップグレードするため、コミット時には、競合するロックを持つアプリケーションが終了するまでロード・ユーティリティーが待機することで、いくらかの遅延が発生する場合があります。

注: アプリケーションが表に対するロックを解放するのを待機する間に、ロード操作がタイムアウトになることはありません。

LOCK WITH FORCE オプション

2 LOCK WITH FORCE オプションを使用すると、ターゲット表に対する競合するロ
2 ックを保留するアプリケーションを強制的にオフにし、ロード操作を継続でき
2 るようにすることができます。システム・カタログ表に対する競合するロックを保留す
2 るアプリケーションは、ロードによって強制的にオフにされることはありません。
2 ロード・ユーティリティーによってアプリケーションが強制的にシステムからオフ
2 にされると、データベース接続が消失し、エラーが戻されます (SQL1224N)。

ALLOW NO ACCESS モードのロード操作では、ロード操作の開始時に存在する表
ロックを保留するアプリケーションが強制的にオフされます。

ALLOW READ ACCESS モードのロード操作では、以下のロックを保留するアプ
リケーションが強制的にオフされます。

- 表共用ロックと競合する表ロック (例: インポートまたは挿入)。
- ロード操作のコミット・フェーズで存在するすべての表ロック。

リカバリー可能データベースのロード操作に COPY NO オプションが指定される場
合、表スペースがバックアップ・ペンディング状態になるまで、ターゲット表ス
ペースのすべてのオブジェクトが共用モードでロックされます。これは、アクセス・
モードにかかわらず発生します。LOCK WITH FORCE オプションが指定されて

いる場合には、共用ロックと競合する表スペースにあるオブジェクトに対してロックを保留するすべてのアプリケーションが強制的にオフになります。

表状態

ロード・ユーティリティーは、ロックに加えて、表状態を使用して、表へのアクセスを制御します。表状態は `LOAD QUERY` コマンドを使って調べることができます。 `LOAD QUERY` コマンドにより戻される表状態は以下のとおりです。

正常 どの表状態も表に影響を与えていません。

チェック・ペンディング

表には、未確認の制約があります。チェック・ペンディング状態を解除するには、`SET INTEGRITY` ステートメントを使用してください。ロード・ユーティリティーは、制約のある表でロード操作を開始する際に、表をチェック・ペンディング状態にします。

ロード進行中

この表には、進行中のロード操作があります。

ロード・ペンディング

この表でのロード操作はアクティブでしたが、データがコミットできるようになる前に打ち切られました。表がこの状態から解除するには、`LOAD TERMINATE`、`LOAD RESTART`、または `LOAD REPLACE` コマンドを発行してください。

読み取りアクセス専用

表データを読み取りアクセス照会に使用することができます。 `ALLOW READ` オプションを使用するロード操作では、表は読み取りアクセス専用状態になります。

使用不可

表は使用できません。表のドロップまたはバックアップからのリストアのどちらかしが行えません。リカバリー不能のロード操作からロールフォワードを実行すると、表は使用不可状態になります。

ロード再始動不可

表は、ロード再始動操作を実行できない部分ロード状態になっています。表はロード・ペンディング状態にもなります。 `LOAD TERMINATE` または `LOAD REPLACE` コマンドを使用すると、表はロード再始動不可状態から解除されます。正常に再始動または終了しないで失敗したロード操作の後でロールフォワード操作が実行される場合、または表がロード進行中またはロード・ペンディング状態で実行されたオンライン・バックアップからリストア操作が実行される場合には、この表はロード再始動不可状態になります。いずれの場合でも、ロード再始動操作で必要な情報は信頼できず、ロード再始動不可状態では、ロードの再始動操作はできません。

タイプ 1 索引 (Type-1 Indexes)

現在、表はタイプ 1 索引を使用しています。この索引に対して `REORG` ユーティリティーを使用する場合、この索引は、`CONVERT` オプションを使用してタイプ 2 に変換できます。

不明 `LOAD QUERY` コマンドで、表状態を判別できません。

1 つの表が、同時に複数の状態になっている可能性があります。たとえば、データが制約付きの表にロードされる場合で ALLOW READ ACCESS オプションが指定されている場合には、表状態は以下のようになります。

```
Tablestate:  
  Check Pending  
  Load in Progress  
  Read Access Only
```

ロード操作の後、および SET INTEGRITY ステートメントの前には、表状態は以下のようになります。

```
Tablestate:  
  Check Pending  
  Read Access Only
```

SET INTEGRITY ステートメントの発行後は、表状態は以下のようになります。

```
Tablestate:  
  Normal
```

COPY NO が指定されている場合の表スペース状態

リカバリー可能データベースで COPY NO オプションを指定したロード操作が実行される際は、ロード操作に関連する表スペースは、バックアップ表スペース状態およびロード進行中の表状態になります。これは、ロード操作の始めに起こります。表スペース内で表に対するロックの獲得に時間がかかるので、ロード操作が遅れる可能性があります。

表スペースがバックアップ・ペンディング状態にある場合には、これまでどおり読み取りアクセスが可能です。表スペースのバックアップ・ペンディング状態を解除するには、この表スペースのバックアップを取るしか方法はありません。ロード操作が打ち切られても、表スペース状態はロード操作の始めに変更されているため、表スペースは引き続きバックアップ・ペンディング状態になっており、失敗してもロールバックできません。ロード進行中の表スペース状態では、データのロード中に COPY NO オプションが指定されたロード操作のオンライン・バックアップを行えません。ロード操作が完了するか、または打ち切られた場合には、ロード進行中状態ではなくなります。

COPY NO オプションが指定された LOAD コマンドでのロールフォワード操作では、関連する表スペースは、リストア・ペンディング状態になります。リストア・ペンディング状態から表スペースを除去するには、リストア操作を実行しなければなりません。ロード操作が正常に完了すると、ロールフォワード操作では、表スペースをリストア・ペンディング状態にすることだけを行います。

関連概念:

- 188 ページの『ロード操作後のペンディング状態』

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・ユーティリティでは、 `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エンコード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 NULL 可能であれば NULL がその列に挿入されます。そうでない場合、要求はリジェクトされます。 `nochecklengths` 修飾子を指定した場合は、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ということが明らかになった場合、その行はリジェクトされます。それ以外の場合、データはロードされます。

関連資料:

- 113 ページの『LOAD』

ロード操作後のペンディング状態

ロード・ユーティリティは、表の状態を使用して、ロード操作時のデータベースの整合性を保持します。それらの状態は `LOAD QUERY` コマンドを使って調べることができます。

ロード・プロセスのロード・フェーズと構築フェーズでは、ロードのターゲット表をロード進行中の表状態にします。また、ロード・ユーティリティは、リカバリ可能データベースに `COPY NO` オプションが指定されている場合に、表スペースをロード進行中状態にします。表スペースはロード操作の間はこの状態のままであり、トランザクションがコミットされるかロールバックされると通常の状態に戻ります。

`NO ACCESS` オプションが指定されている場合には、ロードの進行中に表にアクセスすることはできません。 `ALLOW READ ACCESS` オプションが指定される場合には、ロード・コマンドを呼び出す前に存在した表のデータが、ロード操作時に読み取り専用モードで使用可能になります。 `ALLOW READ ACCESS` オプションが指定されているときに、ロード操作が失敗した場合には、ロード操作の前に表に存在していたデータが、失敗後も読み取り専用モードで引き続き使用可能になります。

ロード操作が失敗したか中断された場合にロード進行中の表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 コマンドを使ってリカバリーした後、それ以降のリカバリー処理を実行します。

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・ペンディング状態になり、さらに、

- 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 (チェック・ペンディング状態) は、参照制約とチェック制約、DATALINKS 制約、生成行制約、マテリアライズ照会計算、またはステージング表伝搬に関係しています。たとえば、既存の表が親表であり、そこに組み込まれた主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両方の表 (表スペースではなく) がチェック・ペンディング状態になります。表がチェック・ペンディング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チェック制約についての妥当性チェックを発行するには、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。

関連概念:

- 103 ページの『保全性違反のチェック』
- 184 ページの『表ロック、表状態、および表スペース状態』

関連資料:

- 「コマンド・リファレンス」の『LIST TABLESPACES コマンド』

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。

ユニーク索引がある場合、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方が効率はよくなります (190 ページの図 5 を参照)。

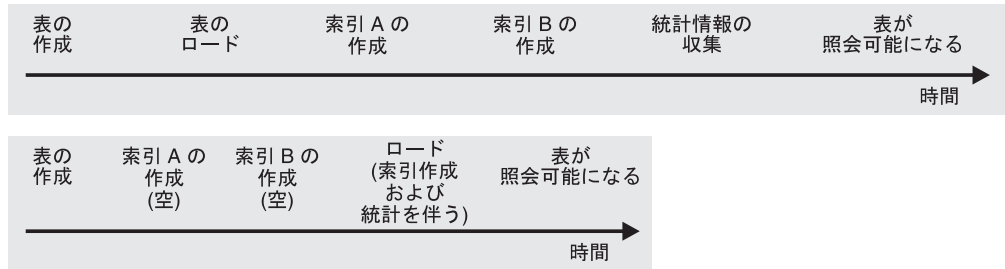


図5. 索引作成と統計集合の並行処理によるロードパフォーマンスの向上：多くの場合、表はデータのロード、索引の構築、統計の集合という3つのステップによって構築されます。これが原因となって、ロード操作中、索引の作成中（表ごとに複数の索引が可能）、および統計の集合中（表データと索引すべてに対する入出力が発生）に、複数のデータ入出力が発生することになります。別の方法として、ロード・ユーティリティがデータを一括してこれらの作業を実行するようにすれば、さらに時間が短くなります。

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 ページの主メモリを使用するようロード・ユーティリティに指示するには、*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 TEMPORARY 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、TEMPORARY 表スペースのバッファ・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並列処理を実現するため、TEMPORARY 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。ランタイムにロードするソート・ライブラリーのロケーションは、**DB2SORT** 環境変数（レジストリー値）を使って指定します。

SET INTEGRITY ステートメントを使用すると、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チェックして制約違反がないかどうかを（表の追加部分だけをチェックすることにより）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チェックできない場合は表全体のチェックが実行されますが、これは表が再び使用可能になる前に実行されます。

同様に、マテリアライズ照会表の基礎表でロード操作が実行される際は、REFRESH TABLE ステートメントを使用すると、基礎表とマテリアライズ照会表の両方を再び完全に使用可能にするのに必要な時間が長くなる可能性があります。REFRESH IMMEDIATE マテリアライズ照会表の基本表に対して INSERT モードでいくつかの順次ロード操作が実行される際は、ほとんどの場合、SET INTEGRITY ステートメントはマテリアライズ照会表を増分でリフレッシュします。全体のリフ

レッシュが必要であるとシステムが判断した場合には、マテリアライズ照会表が再計算されますが、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・ユーティリティのパフォーマンスは、INSERT モードと REPLACE モードとは同じです。

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・ヒープにおける使用可能なサイズとフリー・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 (順方向リカバリー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場合があります。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、順方向リカバリーが有効になっている場合には表スペースがバックアップ・ペンディング状態になって、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、パーティション内並列処理 (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・ユーティリティがデータ・レコードの構文解析、変換、およびフォーマット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 (図 6 を参照)。

表に LOB または LONG VARCHAR のいずれかのデータが収められていると、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 (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、明確なパフォーマンスの向上は期待できません。



図 6. ロード操作中にパーティション内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として割り当てるメモリの合計を指定します。このバッファは、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ストレージ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。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。

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 のいずれかが大きい方です。デフォルトでは、DISK_PARALLELISM はロードする表のオブジェクトを備えたすべての表スペース上にある表スペース・コンテナ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、ユーザー提供の列の値に対して実行するデータ・チェック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 から 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リカバリー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・ペンディング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: これ以後のリストアやロールフォワード・リカバリー操作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が完了した後は、表をドロップするか、または LOAD TERMINATE コマンドを発行してこれをオンラインに戻すことができます。

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想される場合に、リジェクト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。

ALLOW READ ACCESS

このオプションを使用すると、ユーザーはロード操作の進行中に表を参照することができます。ユーザーは、ロード操作前に表に存在していたデータのみ表示できます。INDEXING MODE INCREMENTAL オプションも指定されているときに、ロード操作が失敗した場合には、後続のロード終了操作で、索引での矛盾を訂正する必要があります。これには、かなりの量の入出力を伴う索引スキャンが必要です。ロード終了操作に ALLOW READ ACCESS オプションも指定されている場合には、入出力にバッファ・プールが使用されます。

SAVECOUNT

このパラメーターは、ロード操作中に整合点を確立するインターバルを設定するのに使用します。整合点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 億にもおよぶロード操作の場合は値 10,000,000 など)。

LOAD RESTART 操作は、最後の整合点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` ユーティリティを呼び出す場合よりも効率よくデータ分散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のデータ分散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。

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。

USE <tablespaceName>

このパラメーターを使用すると、SYSTEM TEMPORARY 表スペースで索引を再構築し、ロード操作の索引コピー・フェーズで索引表スペースにコピーし直すことができます。ALLOW READ ACCESS モードのロード操作が完全に索引を再構築すると、新規の索引はシャドー として作成されます。ロード操作の終わりに、元の索引は新規索引に置き換えられます。

デフォルトでは、シャドー 索引は、元の索引と同じ表スペースに作成されます。この場合、元の索引とシャドー 索引の両方が同じ表スペースに同時に常駐しているため、リソース問題の原因となる可能性があります。シャドー 索引が元の索引と同じ表スペースに作成される場合には、元の索引は即座にシャドー に置き換えられます。ただし、SYSTEM TEMPORARY 表スペースにシャドー 索引が作成される場合には、ロード操作で索引コピー・フェーズが必要になります。このフェーズでは、SYSTEM TEMPORARY 表スペースから索引表スペースに索引をコピーします。このコピーに関連した入出力はかなり大きくなります。表スペースのいずれかが DMS 表スペースである場合、SYSTEM TEMPORARY 表スペースの入出力の順序が変わる可能性があります。DISK_PARALLELISM オプションにより指定される値は、索引コピー・フェーズで優先されます。

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。警告がわずかかまたはまったくないと予想される場合には、WARNINGCOUNT パラメーターをその推定警告数に近い数に設定し、予想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ドロップ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、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。

関連概念:

- 108 ページの『マルチディメンション・クラスタリングの考慮事項』
- 「管理ガイド: パフォーマンス」の『DB2 レジストリー変数と環境変数』

関連資料:

- 「管理ガイド: パフォーマンス」の『util_heap_sz - 「ユーティリティ・ヒープ・サイズ」構成パラメーター』
- 「管理ガイド: パフォーマンス」の『stat_heap_sz - 「統計ヒープ・サイズ」構成パラメーター』

- 「SQL リファレンス 第 2 巻」の『SET INTEGRITY ステートメント』
- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

ロード - 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 つのエレメントがあります。

- ELE1、位置 01 から 20
- ELE2、位置 21 から 22
- ELE3、位置 23 から 23
- ELE4、位置 24 から 27
- ELE5、位置 28 から 31
- ELE6、位置 32 から 32
- ELE7、位置 33 から 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   XXN
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 NULL 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 NULL 標識位置が Y なら、その列は NULL 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用されます。この例の場合、行 1 ではどちらの列も NULL ではなく、行 2 では COL2 が NULL、行 3 では COL3 が NULL 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 NULL 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 (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 NULL であり、列定義では NOT NULL が指定されているためにリジェクトされます。DEL 形式と互換でない開始文字の入った列は、エラーを生成し、レコードはリジェクトされます。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。

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。このレコードはリジェクトされません。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。このレコードはリジェクトされません。

例 3 (DATALINK データのロード)

以下に示すコマンドは、DEL フォーマットのデータの入った入力ファイル delfile1 から表 MOVIE TABLE 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 (LINKTYPE URL を使用)
url_movie	DATALINK (LINKTYPE URL を使用)

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縦線
(|) 文字が使われています。
3. url_making_of の列値が接頭部文字シーケンスに組み込まれていない場合、
"http://narang" が使用されます。
4. url_movie の NULL 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 NULL 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. 表のロード中にユニーク索引または 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 に入れます。

例 4 (ID 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 ID 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 フォーマット):

```
"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 フォーマット):

```
"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 ID 値が生成されます。これは、
DATAFILE1 内にこれらの行の ID 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 ID 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、すべての行の ID 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、それぞれの行ごとに ID 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)
db2load from datafile2.del of del modified by identitymissing
  replace into table1
```

4. DATAFILE1 を TABLE2 にロードして、ID 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 はリジェクトされます。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 ID 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、ID 列は暗黙的に NULL でないことになるので、行はリジェクトされます。

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 はリジェクトされます。これは、行 3 と 4 では独自に非 NULL 値が提供されており、ID 列が GENERATED ALWAYS であるためです。

例 5 (CURSOR からのロード)

MY.TABLE1 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

MY.TABLE2 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

カーソル MYCURSOR は以下のように定義されます。

```
declare mycursor cursor for select * from my.table1
```

次のコマンドは、すべてのデータを MY.TABLE1 から MY.TABLE2 にロードします。

```
load from mycursor of cursor method P(1,2,3) insert into
  my.table2(one,two,three)
```

注:

1. 単一の LOAD コマンドには、カーソル名を 1 つだけ指定できます。つまり、load from mycurs1, mycurs2 of cursor... は許可されません。

2. カーソルからのロードに有効な METHOD 値は、P および N だけです。
3. この例では、METHOD P および挿入列リスト (one,two,three) はデフォルト値を示すため、これらを省略してもかまいません。
4. MY.TABLE1 は、表、ビュー、別名、またはニックネームが使用できます。

関連概念:

- 84 ページの『ロードの概要』

関連資料:

- 137 ページの『LOAD QUERY』
- 219 ページの『パーティション・データベース・ロード・セッションの例』
- 113 ページの『LOAD』

第 4 章 パーティション・データベース環境でのデータのロード

この章では、パーティション・データベース環境でのデータのロードについて説明します。

以下のトピックを取り上げています。

- 『パーティション・データベース・ロードの概説』
- 203 ページの『パーティション・データベース環境でのロードの使用』
- 209 ページの『ロード照会コマンドを使用したパーティション・データベース・ロードのモニター』
- 211 ページの『パーティション・データベース環境でのロード操作の再始動または終了』
- 213 ページの『パーティション・データベース・ロード構成オプション』
- 219 ページの『パーティション・データベース・ロード・セッションの例』
- 222 ページの『移行およびバックレベル互換性』
- 224 ページの『パーティション・データベース環境でのデータのロード - ヒント』

パーティション・データベース・ロードの概説

パーティション・データベースでは、非常に大きなデータがたくさんのパーティション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・パーティションに入るかは、パーティション・キーによって決定されます。また、適切なデータベース・パーティション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。パーティション・データベース環境に表をロードする際に、ロード・ユーティリティは以下を実行できます。

- 入力データを並列でパーティションする。
- データをそれぞれ対応するデータベース・パーティションに同時にロードする。
- あるシステムから別のシステムにデータを転送する。

パーティション・データベースのロード操作は 2 つのフェーズで実行されます。セットアップ・フェーズでは表ロックなどのパーティション・リソースが獲得され、ロード・フェーズではデータがパーティションにロードされます。LOAD コマンドの ISOLATE_PART_ERRS オプションを使用すると、これらのフェーズのいずれかで発生するエラーを処理する方法、および 1 つまたは複数のパーティションでのエラーが、エラーのないパーティションでのロード操作にどのように影響を与えるかを選択できます。

パーティション・データベースにデータをロードする際には、以下のいずれかのモードを使用できます。

- PARTITION_AND_LOAD。データは (多くの場合は並列で) パーティションされ、それぞれ対応するデータベース・パーティションに同時にロードされます。

- **PARTITION_ONLY**。データは (多くの場合は並列で) パーティションされ、それぞれのロード・パーティションの指定したファイルに出力が書き込まれます。それぞれのファイルにはパーティション・ヘッダーがあり、データがパーティションされた方法、および **LOAD_ONLY** モードを使用してデータベースにファイルをロードできることを示します。
- **LOAD_ONLY**。データはすでにパーティション化されているものとします。この場合はパーティション化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・パーティションに同時にロードされます。
- **LOAD_ONLY_VERIFY_PART**。データはすでにパーティション化されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。パーティション化プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいパーティションにあることがチェックされます。 *dumpfile* ファイル・タイプ修飾子が指定されている場合には、パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。ロードしている特定のパーティションにパーティション違反がある場合、そのパーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。
- **ANALYZE**。すべてのデータベース・パーティションに均一に分散する最適なパーティション化マップが生成されます。

概念および用語

パーティション・データベース環境でのロード・ユーティリティの動作および操作について説明する際には、以下の用語が使用されます。

- **コーディネーター・パーティション** は、ロード操作を実行するためにユーザーが接続するデータベース・パーティションです。
PARTITION_AND_LOAD、**PARTITION_ONLY**、および **ANALYZE** モードでは、ロード・コマンドの **CLIENT** オプションが指定されていない限り、データ・ファイルはこのパーティションに常駐するものとされます。ロード・コマンドに **CLIENT** オプションを指定すると、ロードされるデータが、リモートで接続されるクライアントに常駐するように指示します。
- **PARTITION_AND_LOAD**、**PARTITION_ONLY**、および **ANALYZE** モードでは、事前パーティション化エージェント がユーザー・データを読み取り、データをパーティションするパーティション化エージェント にラウンドロビン方式でこれを分散します。この処理は、コーディネーター・パーティションで常に行われます。どのロード操作の場合でも、1 つのパーティションにつき最大 1 つのパーティション化エージェントが許可されます。
- **PARTITION_AND_LOAD**、**LOAD_ONLY**、および **LOAD_ONLY_VERIFY_PART** モードでは、それぞれの出力パーティションでロード・エージェント が実行し、そのパーティションへのデータのロードを調整します。
- **ファイル・エージェントへのロード** は、**PARTITION_ONLY** ロード操作時にそれぞれの出力パーティションで実行します。これらはパーティション化エージェントからデータを受信し、これをパーティションにあるファイルに書き込みます。
- **ファイル転送コマンド・エージェント** はコーディネーター・パーティションで実行し、ファイル転送コマンドの実行を担当します。

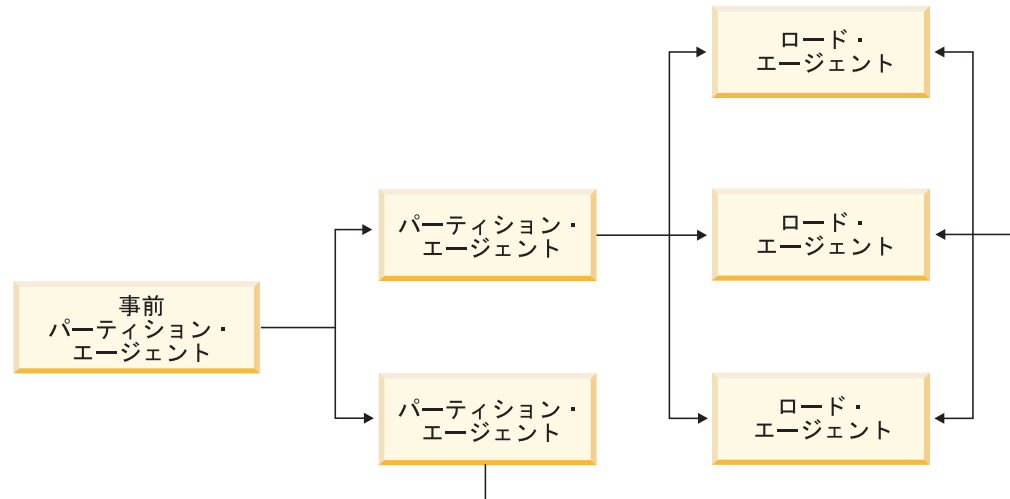


図7. パーティション・データベース・ロードの概説： 事前パーティション化エージェントによりソース・データを読み、2つのパーティション化エージェントそれぞれに約半分のデータが送られます。パーティション化エージェントはデータをパーティション化し、各パーティションを3つのデータベース・パーティションのいずれかに送ります。各パーティションのロード・エージェントがデータをロードします。

関連概念:

- 224 ページの『パーティション・データベース環境でのデータのロード - ヒント』
- 209 ページの『ロード照会コマンドを使用したパーティション・データベース・ロードのモニター』
- 211 ページの『パーティション・データベース環境でのロード操作の再始動または終了』

関連資料:

- 213 ページの『パーティション・データベース・ロード構成オプション』

パーティション・データベース環境でのロードの使用

前提条件:

パーティション・データベース環境に表をロードする前に、以下を実行してください。

1. データベース・マネージャ構成パラメーター *svcname*、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。ロード・ユーティリティーは TCP/IP を使用して事前パーティション化エージェントからパーティション化エージェントにデータを転送し、さらにパーティション化エージェントからロード・エージェントにデータを転送するため、このことは重要です。
2. 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。ロード・ユーティリティーは COMMIT ステートメントを発行するため、ロード操作の開始前に COMMIT または ROLLBACK ステートメントを発行することにより、すべてのトランザクションを完了し、すべてのロックを

解除しておかなければなりません。 PARTITION_AND_LOAD、PARTITION_ONLY、または ANALYZE モードが使用されている場合には、ロードされるデータ・ファイルは、以下の状態になっていない限り、このパーティションになければなりません。

- a. CLIENT オプションが指定されている場合。この場合には、データがクライアント・マシンになければなりません。
- b. 入力ソース・データが CURSOR である場合。この場合には、入力ファイルはありません。

3. 設計アドバイザーを実行して、各表ごとに最適なパーティションを決定することができます。詳しくは、『設計アドバイザー』を参照してください。

制約事項:

ロード・ユーティリティを使用してパーティション・データベース環境にデータをロードする際には、以下の制約が適用されます。

- ロード操作の入力ファイルのロケーションとして磁気テープ装置を指定することはできません。
- ANALYZE モードが使用されていない限り、ROWCOUNT オプションはサポートされません。
- ターゲット表にパーティション化に必要な ID 列 があり、*identityoverride* 修飾子が指定されていない場合、または複数のデータベース・パーティションを使ってデータをパーティション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。
- ID 列がパーティション・キーの一部を構成している場合は、PARTITION_AND_LOAD モードだけがサポートされます。
- LOAD コマンドの CLIENT オプションを指定する際には、LOAD_ONLY および LOAD_ONLY_VERIFY_PART モードは使用できません。
- CURSOR 入力ソース・タイプを指定する際には、LOAD_ONLY_VERIFY_PART モードは使用できません。
- LOAD コマンドの ALLOW READ ACCESS および COPY YES オプションを指定する際には、パーティション・エラー分離モード LOAD_ERRS_ONLY および SETUP_AND_LOAD_ERRS は使用できません。
- OUTPUT_DBPARTNUMS および PARTITIONING_DBPARTNUMS オプションによって指定されるパーティションがオーバーラップしない場合には、複数のロード操作が同じ表に同時にデータをロードすることができます。たとえば、表がパーティション 0 から 3 に定義されている場合、1 つのロード操作がパーティション 0 および 1 にデータをロードする一方で、2 番目のロード操作でパーティション 2 および 3 にデータをロードすることができます。
- 区切りなし ASCII (ASC) および区切り付き ASCII (DEL) ファイルのみパーティション化できます。PC/IXF ファイルはパーティション化できません。PC/IXF ファイルを複数パーティションの表にロードするには、まず単一パーティションの表にこれをロードしてから、CURSOR ファイル・タイプを使用してロード操作を実行し、複数パーティション表にデータを移動します。また、LOAD_ONLY_VERIFY_PART モードでロードを行って、複数パーティション表に PC/IXF ファイルをロードすることもできます。

手順:

以下の例では、LOAD コマンドを使用して各種のロード操作を開始する方法を説明します。以下の例で使用されるデータベースには、5 つのパーティション、0、1、2、3、および 4 があります。各パーティションにはローカル・ディレクトリー /udb/data/ があります。パーティション 0、1、3、および 4 では、2 つの表、TABLE1 および TABLE2 が定義されます。クライアントからロードする際には、ユーザーにはデータベース・パーティションのいずれかではないリモート・クライアントへのアクセスがあります。

サーバー・パーティションからのロード

パーティションおよびロードの例

このシナリオでは、TABLE1 が定義されているか、または定義されていないパーティションに接続しているものとします。データ・ファイル load.del は、このパーティションの現行作業ディレクトリーにあります。load.del から、TABLE1 が定義されているすべてのパーティションにデータをロードするには、次のコマンドを発行します。LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1

注: この例では、すべてのパーティション・データベース構成パラメーターでデフォルト値を使用します。MODE パラメーターのデフォルトは PARTITION_AND_LOAD で、OUTPUT_DBPARTNUMS オプションのデフォルトは、TABLE1 が定義されているすべてのノードです。また、PARTITIONING_DBPARTNUMS のデフォルトは、LOAD コマンド規則に従って選択したノードのセットです。この規則は、パーティション化ノードが指定されていない場合にそれを選択するためのものです。

パーティション化ノードとしてノード 3 および 4 を使用してロード操作を実行するには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

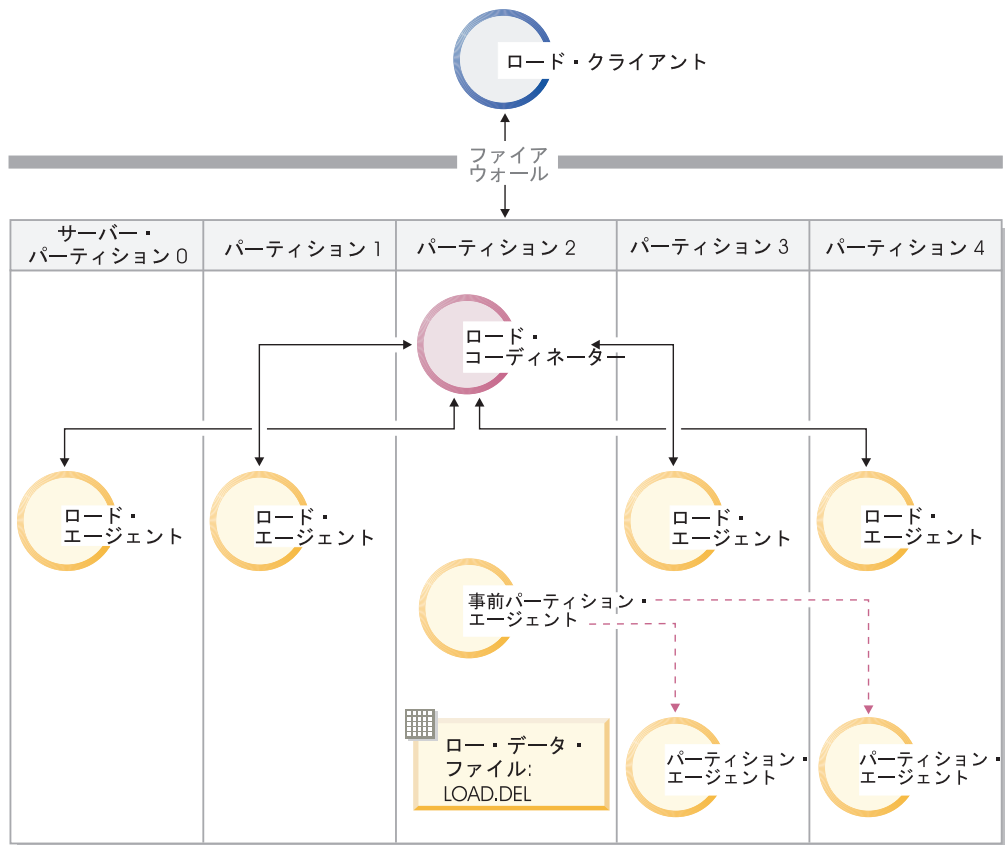



図 8. : この図では、上記のコマンドが発行された結果の動作を説明します。データは、パーティション 3 および 4 にロードされます。

パーティションのみの例

このシナリオでは、TABLE1 が定義されているか、または定義されていないパーティションに接続しているものとします。データ・ファイル load.del は、このパーティションの現行作業ディレクトリーにあります。TABLE1 が定義されているすべてのデータベース・パーティションに load.del をパーティション化する (ロードはしない) には、パーティション 3 および 4 をパーティション化ノードとして使用し、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
PART_FILE_LOCATION /udb/data
PARTITIONING_DBPARTNUMS (3,4)
```

これにより、ファイル load.del.xxx は、それぞれのパーティションにある /udb/data ディレクトリーに保管されます。ここで、xxx は、3 桁表記のパーティション番号です。

パーティション 0 (PARTITIONING_DBPARTNUMS のデフォルト) で実行しているパーティション化エージェントを 1 つだけ使用して、load.del ファイルをパーティション 1 および 3 にパーティションするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
PART_FILE_LOCATION /udb/data
OUTPUT_DBPARTNUMS (1,3)
```

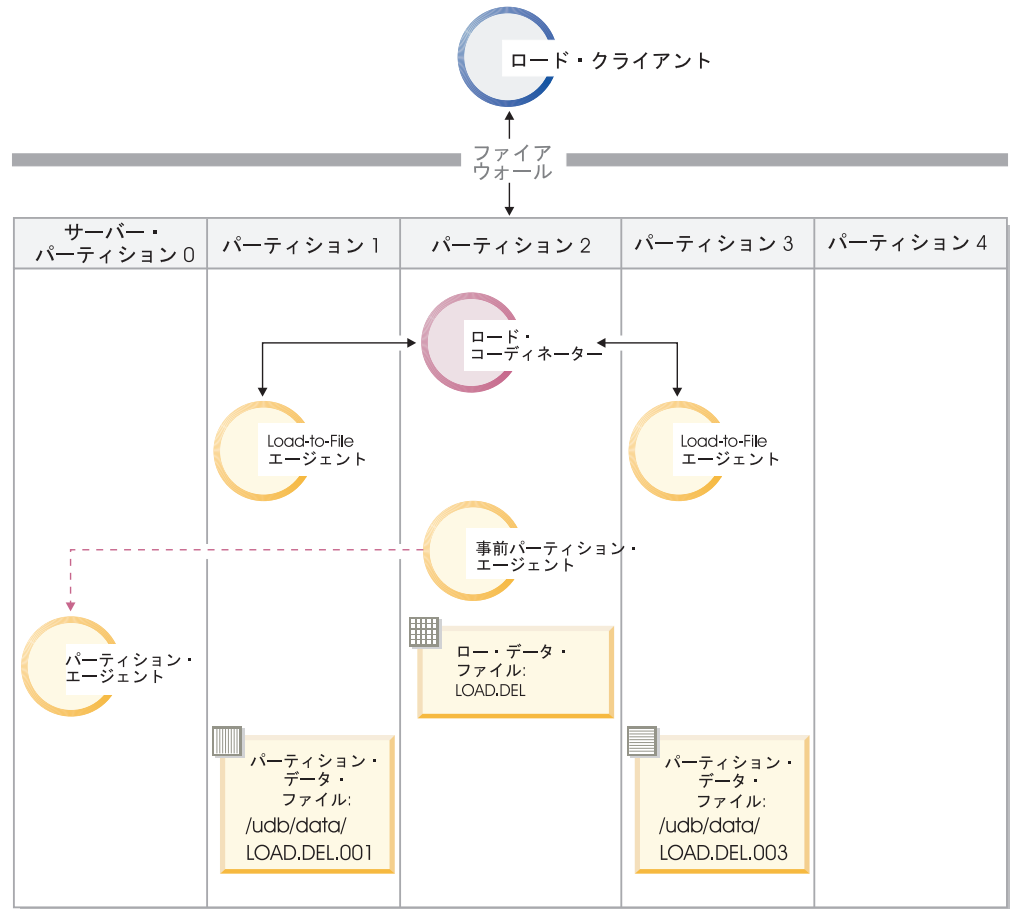


図 9. : この図では、上記のコマンドが発行された結果の動作を説明します。データは、パーティション 0 で実行する 1 パーティション化エージェントを使用して、パーティション 1 および 3 にロードされます。

ロードのみの例

すでに PARTITION_ONLY モードでロード操作を実行しており、TABLE1 が定義されているすべてのパーティションにそれぞれのロード・パーティションの /udb/data ディレクトリーにあるパーティション・ファイルをロードする場合には、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /udb/data
```

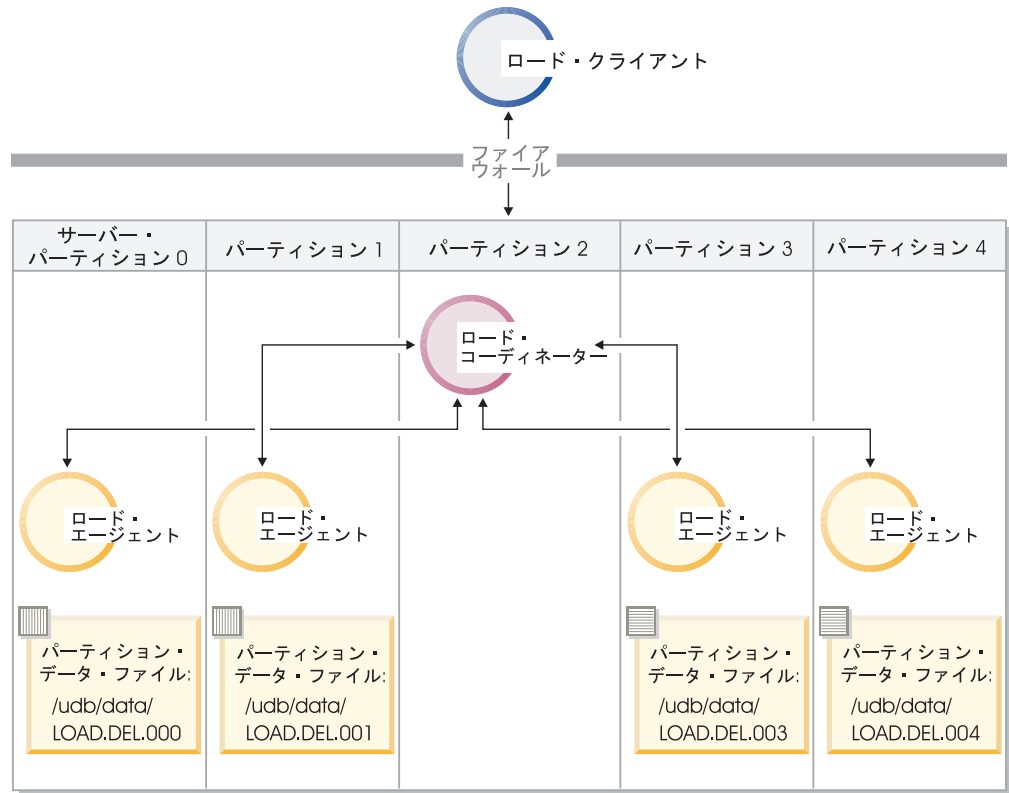


図 10. : この図では、上記のコマンドが発行された結果の動作を説明します。パーティション・データは、TABLE1 が定義されているすべてのパーティションにロードされます。

パーティション 4 にだけロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /udb/data
OUTPUT_DBPARTNUMS (4)
```

パーティション化マップ・ヘッダーのない事前パーティション・ファイルのロード

LOAD コマンドを使用して、パーティション・ヘッダーのないデータ・ファイルを、いくつかのデータベース・パーティションに直接ロードすることができます。TABLE1 が定義されているそれぞれのパーティションの /udb/data ディレクトリーにデータ・ファイルが存在しており、この名前が load.del.xxx である場合 (xxx はパーティション番号)、以下のコマンドを発行することによってファイルをロードできます。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
PART_FILE_LOCATION /udb/data
```

パーティション 1 にだけデータをロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
PART_FILE_LOCATION /udb/data
OUTPUT_DBPARTNUMS (1)
```

注: ロード元のパーティションにない行が指定された場合には、これはリジェクトされ、ダンプ・ファイルに入れます。

リモート・クライアントからパーティション・データベースへのロード

リモート・クライアントにあるファイルからパーティション・データベースにデータをロードするには、LOAD コマンドの CLIENT オプションを指定して、サーバー・パーティションにデータ・ファイルが存在しないことを示す必要があります。たとえば、次のようにします。

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

注: LOAD_ONLY または LOAD_ONLY_VERIFY_PART モードを CLIENT オプションと共に使用することはできません。

カーソルからのロード

単一パーティション・データベースの場合と同様に、カーソルから複数パーティション・データベースにロードすることができます。この例では、PARTITION_ONLY および LOAD_ONLY モードの場合、PART_FILE_LOCATION オプションは完全修飾ファイル名を指定しなければなりません。この名前は、それぞれの出力パーティションで作成またはロードされるパーティション・ファイルの完全修飾ベース・ファイル名になります。ターゲット表に LOB 列がある場合には、指定されたベース名で複数のファイルを作成できます。

将来 TABLE2 にロードする目的で、/udb/data/select.out.xxx (ここで、xxx はノード番号) という名前のそれぞれのパーティションにあるファイルに、ステートメント SELECT * FROM TABLE1 の応答セット内のすべての行をパーティションするには、DB2 コマンド行プロセッサから以下のコマンドを発行します。

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED DB CONFIG MODE PARTITION_ONLY
PART_FILE_LOCATION /udb/data/select.out
```

上記の操作によって生成されるデータ・ファイルは、以下の LOAD コマンドを発行することによってロードできます。

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED CB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /udb/data/select.out
```

関連概念:

- ・ 「管理ガイド: パフォーマンス」の『設計アドバイザー』
- ・ 259 ページの『カーソル・ファイル・タイプを使用したデータの移動』

関連資料:

- ・ 139 ページの『db2Load - ロード』

ロード照会コマンドを使用したパーティション・データベース・ロードのモニター

パーティション・データベースのロード時に生成されるメッセージ・ファイル

ロード操作時には、いくつかのロード・プロセスによって、それらが実行されるパーティションにメッセージ・ファイルが作成されます。これらのファイルには、すべての情報、プロセスの実行時に生成される警告およびエラー・メッセージが保管されます。ユーザーが表示できるメッセージ・ファイルを生成するロード・プロセスは、ロード・エージェント、事前パーティション化エージェント、およびパーティション化エージェントです。

ユーザーは、ロード操作時に個々のパーティションに接続し、ターゲット表に対して `LOAD QUERY` コマンドを発行できます。このコマンドが `CLP` から発行されると、`LOAD QUERY` コマンドで指定された表の、現在このパーティションにあるすべてのメッセージ・ファイルの内容が表示されます。

たとえば、データベース `WSDB` のパーティション 0 から 3 に表 `TABLE1` が定義されているとします。ユーザーはパーティション 0 に接続され、以下の `LOAD` コマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
partitioning_dbpartnums (1)
```

このコマンドは、パーティション 0、1、2、および 3 で実行するロード・エージェント、パーティション 1 で実行するパーティション化エージェント、パーティション 0 で実行する事前パーティション化エージェントを組み込むロード操作を開始します。

パーティション 0 には、事前パーティション化エージェントのメッセージ・ファイルが 1 つ、およびそのパーティションでのロード・エージェントのファイルが 1 つ配備されます。これらのファイルの内容を同時に表示するには、新しいセッションを開始し、以下のコマンドを `CLP` から発行します。

```
set client connect_node 0
connect to wsdb
load query table table1
```

パーティション 1 には、ロード・エージェントのファイルが 1 つ、およびパーティション化エージェントのファイルが 1 つ配備されます。これらのファイルの内容を表示するには、新しいセッションを開始し、以下のコマンドを `CLP` から発行します。

```
set client connect_node 1
connect to wsdb
load query table table1
```

注: `STATUS_INTERVAL` ロード構成オプションによって生成されるメッセージは、事前パーティション化エージェント・メッセージ・ファイルに表示されます。ロード操作時にこれらのメッセージを表示するには、コーディネーター・パーティションに接続してから、`LOAD QUERY` コマンドを発行しなければなりません。

メッセージ・ファイルの内容の保管

ロード API (`db2Load`) を介してロード操作が開始される場合には、メッセージ・オプション (`piLocalMsgFileName`) を指定しなければならず、メッセージ・ファイルはサーバーからクライアントに移動して、ユーザーが表示できるように保管されます。

CLP から開始されるパーティション・データベースのロード操作では、メッセージ・ファイルがコンソールに表示されたり、保存されたりすることはありません。パーティション・データベースのロードの完了後にこれらのファイルの内容を保管または表示するには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。このオプションが使用される場合、ロード操作が完了すると、それぞれのパーティションのメッセージ・ファイルはクライアント・マシンに転送され、MESSAGES オプションによって示される基本名を使用してファイルに保管されます。パーティション・データベースのロード操作の場合、生成されたロード・プロセスに対応するファイルの名前を以下にリストします。

プロセス・タイプ	ファイル名
ロード・エージェント	<message-file-name>.LOAD.<partition-number>
パーティション化エージェント	<message-file-name>.PART.<partition-number>
事前パーティション化エージェント	<message-file-name>.PREP.<partition-number>

たとえば、MESSAGES オプションが /wsdb/messages/load を指定すると、パーティション 2 のロード・エージェント・メッセージ・ファイルは /wsdb/messages/load.LOAD.002 になります。

注: CLP から開始したパーティション・データベースのロード操作には、MESSAGES オプションを使用することを強くお勧めします。

関連資料:

- 164 ページの『db2LoadQuery - ロードの照会』

パーティション・データベース環境でのロード操作の再始動または終了

パーティション・データベース環境でのロード・プロセスは 2 つのステージで構成されています。1 つは、出力パーティションに対する表ロックなどのパーティション・レベルのリソースが獲得されるセットアップ・ステージ、もう 1 つは、データがフォーマットされ、パーティションの表にロードされるロード・ステージです。これらのステージのいずれかまたは両方でエラーが発生した場合には、4 つのパーティション・エラー分離モード (LOAD_ERRS_ONLY、SETUP_ERRS_ONLY、SETUP_AND_LOAD_ERRS、および NO_ISOLATION) が作動して、ロード再始動操作および終了操作が実行されます。通常、セットアップ・ステージで障害が発生した場合には、操作の再始動および終了は必要ではありません。しかし、ロード・ステージで障害が発生した場合には、このロード操作に関係するすべてのパーティションで LOAD RESTART または LOAD TERMINATE を実行する必要があります。

セットアップ・ステージ中の障害

セットアップ・ステージで少なくとも 1 つのパーティションでロード操作が失敗し、このセットアップ・ステージ・エラーが分離されていない場合 (つまり、エラー分離モードが LOAD_ERRS_ONLY または NO_ISOLATION のいずれかである場合)、ロード操作全体が打ち切れ、それぞれのパーティションの表の状態は、ロード操作以前の状態にロールバックされます。この場合、LOAD RESTART または LOAD TERMINATE コマンドを発行する必要はありません。

初期セットアップ・ステージで少なくとも 1 つのパーティションでロード操作が失敗し、セットアップ・ステージ・エラーが分離されている場合 (つまり、エラー分離モードが `SETUP_ERRS_ONLY` または `SETUP_AND_LOAD_ERRS` のいずれかである場合)、ロード操作は、セットアップ・ステージが正常に実行されたパーティションで継続しますが、失敗したそれぞれのパーティションにある表は、ロード操作以前の状態にロールバックされます。この場合、ロード・ステージにも障害がある場合を除き、ロードの再始動操作または終了操作を実行する必要はありません。

セットアップ・ステージ中にロード操作が失敗したパーティションでロード・プロセスを完了するためには、`LOAD REPLACE` または `LOAD INSERT` コマンドを発行してから、`OUTPUT_DBPARTNUMS` オプションを使用して、元のロード操作時に失敗したパーティションのパーティション番号のみ指定します。

たとえば、データベース `WSDB` のパーティション 0 から 3 に表 `TABLE1` が定義されているとします。以下のコマンドが発行されます。

```
load from load.del of del replace into table1 partitioned db config
  isolate_part_errs setup_and_load_errs
```

ロード操作のセットアップ・ステージでは、パーティション 1 および 3 に障害がありました。セットアップ・ステージ・エラーは分離されているため、パーティション 0 および 2 ではロード操作が完了し、データがロードされます。パーティション 1 および 3 にデータをロードしてロード操作を完了するには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
  output_dbpartnums (1, 3)
```

ロード・ステージ中の障害

パーティション・データベースのロード操作のロード・ステージで少なくとも 1 つのパーティションにおいてロード操作が失敗した場合には、データのロード中にエラーが発生したかどうかにかかわらず、このロード操作に関係するすべてのパーティションで `LOAD RESTART` または `LOAD TERMINATE` コマンドを発行しなければなりません。これが必要になるのは、パーティション・データベース環境でのデータのロードが単一のトランザクションで実行されるためです。ロードの再始動操作が開始されると、ロードが止まった時点から、すべてのパーティションでロードが継続されます。

たとえば、データベース `WSDB` のパーティション 0 から 3 に表 `TABLE1` が定義されているとします。以下のコマンドが発行されます。

```
load from load1.del of del replace into table1 partitioned db config
  isolate_part_errs no_isolation
```

ロード操作のロード・ステージでは、パーティション 1 および 3 に障害がありました。ロード操作を再開するには、すべてのパーティションでロード操作が再始動されなければならないため、`LOAD RESTART` コマンドで、元のコマンドと同じ出力パーティションのセットを指定しなければなりません。

```
load from load.del of del restart into table1 partitioned db config
  isolate_part_errs no_isolation
```

注: ロード再始動操作では、LOAD RESTART コマンドで指定されるオプションが使用されるため、元の LOAD コマンドで指定されるものと同一であることが重要です。

ロード・ステージ中にパーティション・データベース・ロード操作が失敗したときに LOAD TERMINATE コマンドが使用される場合は、以前のロード操作で行われた作業はすべて消失し、それぞれのパーティションの表は初期ロード操作の前の状態に戻されます。

たとえば、データベース WSDB のパーティション 0 から 3 に表 TABLE1 が定義されているとします。以下のコマンドが発行されます。

```
load from load.del of del replace into table1 partitioned db config
isolate_part_errs no_isolation
```

ロード・ステージで障害が発生する場合には、元のコマンドと同じ出力パラメータを指定する LOAD TERMINATE コマンドを発行することにより、ロード操作を終了できます。

```
load from load.del of del terminate into table1 partitioned db config
isolate_part_errs no_isolation
```

関連概念:

- 110 ページの『中断したロード操作の再開』
- 201 ページの『パーティション・データベース・ロードの概説』

パーティション・データベース・ロード構成オプション

HOSTNAME X

このオプションは、FILE_TRANSFER_CMD オプションと共に使用される場合にのみ関係します。X は、データ・ファイルが常駐するリモート・マシンの名前です。これは、z/OS ホストまたは別のワークステーションのいずれかにすることができます。このオプションが指定されていない場合で FILE_TRANSFER_CMD オプションが指定されている場合には、ホスト名 *nohost* が使用されます。

FILE_TRANSFER_CMD X

データが任意のパーティションにロードされる前に呼び出される、ファイル実行可能プログラム、バッチ・ファイル、またはスクリプトを指定します。指定される値は、完全修飾パスでなければなりません。絶対パスは、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。コマンドは、以下の構文を使用して呼び出されます。

```
<COMMAND> <logpath> <hostname> <basepipename> <nummedia>
<source media list>
```

ここで、

<COMMAND>

FILE:TRANSFER:CMD 修飾子によって指定されるコマンドです。

<logpath>

データのロード元のファイルのログ・パスです。診断または一時データがこのパスに書き込まれます。

<hostname>

HOSTNAME オプションの値です。

<basepipename>

これは、ロード操作が作成し、データの受信元とする名前付きパイプのベース名です。LOAD コマンドのソース・ファイルごとに 1 つのパイプが作成されます。これらのファイルの末尾は .xxx になります。xxx は、LOAD コマンドのソース・ファイルの索引です。たとえば、LOAD コマンドに 2 つのソース・ファイルがあり、<basepipename> が pipe123 である場合、pipe123.000 および pipe123.001 という、2 つの名前付きパイプが作成されます。<COMMAND> ファイルは、これらの名前付きパイプにユーザー・データを入れます。

<nummedia>

後に続くメディア引き数の数を指定します。

<source media lst>

LOAD コマンドで指定されるソース・ファイルのリストです。各ソース・ファイルは二重引用符で囲む必要があります。

PART_FILE_LOCATION X

PARTITION_ONLY、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、このパラメーターは、パーティション・ファイルのロケーションを指定するために使用できます。このロケーションは、OUTPUT_DBPARTNUMS オプションによって指定される各パーティションに存在しなければなりません。指定されたロケーションが相対パス名の場合には、そのパスが現行ディレクトリーに追加されて、パーティション・ファイルのロケーションが作成されます。

CURSOR ファイル・タイプの場合、このオプションを指定しなければならず、ロケーションは完全修飾されたファイル名を参照していなければなりません。この名前は、PARTITION_ONLY モードの場合は、各出力パーティションで作成されたパーティション・ファイルの完全修飾された基本ファイル名、または LOAD_ONLY モードの場合は、各パーティションから読み取られるファイルのロケーションです。PARTITION_ONLY モードの使用時に、ターゲット表中に LOB 列があると、指定した基本名のファイルが複数作成されることがあります。

CURSOR 以外のファイル・タイプの場合、このオプションが指定されないと、現行ディレクトリーがパーティション・ファイルに使用されます。

OUTPUT_DBPARTNUMS X

X は、パーティション番号のリストを示します。パーティション番号は、ロード操作が実行されるデータベース・パーティションを示します。パーティション番号は、表が定義されているデータベース・パーティションのサブセットでなければなりません。デフォルトでは、すべてのデータベース・パーティションが選択されます。リストは括弧で囲まなければならない、リスト内のアイテムはコンマで区切らなければなりません。範囲を指定できます (たとえば、(0, 2 から 10, 15))。

PARTITIONING_DBPARTNUMS X

X は、パーティション・プロセスで使用するパーティション番号のリスト

を示します。リストは括弧で囲まなければならない、リスト内のアイテムはコンマで区切らなければならない。範囲を指定できます (たとえば、(0, 2 から 10, 15))。指定されるパーティション化ノードは、ロードされるデータベース・パーティションと異なってもかまいません。指定されない場合には、最適なパフォーマンスを実現するために、LOAD コマンドにより必要なパーティションの数と使用するパーティションが判別されます。

LOAD コマンドで ANYORDER 修飾子が指定されていない場合、ロード・セッションではパーティション・エージェントが 1 つだけ使用されます。さらに、OUTPUT_DBPARTNUMS オプションにパーティションが 1 つだけ指定されている場合、またはロード操作のコーディネーター・ノードが OUTPUT:DBPARTNUMS のエレメントではない場合、パーティション化パーティションとしてロード操作のコーディネーター・ノードが使用されます。その他の場合には、OUTPUT_DBPARTNUMS で最初のパーティション (コーディネーター・ノードではない) がパーティション化パーティションとして使用されます。

ANYORDER 修飾子が指定されている場合には、パーティション化ノードの数は、(OUTPUT_DBPARTNUMS のパーティションの数)/4 + 1 で決定されます。その後、データをロードするために使用されているパーティションを除いて、OUTPUT_DBPARTNUMS からパーティション化ノードの数が選択されます。

MODE X

パーティション・データベースをロードする際に実行するロード操作のモードを指定します。PARTITION_AND_LOAD がデフォルトです。有効な値は以下のとおりです。

- PARTITION_AND_LOAD。データは (多くの場合は並列で) パーティションされ、それぞれ対応するデータベース・パーティションに同時にロードされます。
- PARTITION_ONLY。データは (多くの場合は並列で) パーティションされ、それぞれのロード・パーティションの指定したファイルに出力が書き込まれます。CURSOR 以外のファイル・タイプの場合、各パーティションの出力ファイル名のフォーマットは filename.xxx となります。ここで filename は、LOAD コマンドで指定された入力ファイル名で、xxx は 3 桁のパーティション番号です。CURSOR ファイル・タイプの場合、各パーティションの出力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各パーティションのパーティション・ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

注:

1. このモードは CLI ロード操作には使用できません。
 2. パーティション化に必要な ID 列が表に収められている場合は、*identityoverride* 修飾子が指定されない限り、このモードはサポートされません。
- LOAD_ONLY。データはすでにパーティション化されているものとします。この場合はパーティション・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・パーティションに同時にロードされます。CURSOR 以外のファイル・タイプの場合、各パーティションの入力ファ

イル名のフォーマットは filename.xxx となります。ここで filename は、LOAD コマンドで指定されたファイルの名前で、xxx は 3 桁のパーティション番号です。CURSOR ファイル・タイプの場合、各パーティションの入力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各パーティションのパーティション・ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

注:

1. このモードは CLI ロード操作には使用できず、LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。
 2. パーティション化に必要な ID 列が表に収められている場合は、identityoverride 修飾子が指定されない限り、このモードはサポートされません。
- **LOAD_ONLY_VERIFY_PART**。データはすでにパーティション化されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。パーティション・プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいパーティションにあることがチェックされます。dumpfile ファイル・タイプ修飾子が指定されている場合には、パーティション違反のある行がダンプ・ファイルに入れます。そうでなければ、行は廃棄されます。特定のロード・パーティションにパーティション違反がある場合、そのパーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。各パーティションの入力ファイル名のフォーマットは filename.xxx となり、ここで filename は LOAD コマンドで指定されたファイルの名前で、xxx は 3 桁のパーティション番号です。各パーティションのパーティション・ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

注:

1. このモードは CLI ロード操作には使用できず、LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。
 2. パーティション化に必要な ID 列が表に収められている場合は、identityoverride 修飾子が指定されない限り、このモードはサポートされません。
- **ANALYZE**。すべてのデータベース・パーティションに均一に分散する最適なパーティション化マップが生成されます。

MAX_NUM_PART_AGENTS X

ロード・セッションで使用されるパーティション化エージェントの最大数を指定します。デフォルトは 25 です。

ISOLATE_PART_ERRS X

個々のパーティションで発生するエラーにロード操作がどのように対応するかを指示します。LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合 (この場合のデフォルトは NO_ISOLATION) を除き、デフォルトは LOAD_ERRS_ONLY です。有効な値は以下のとおりです。

- **SETUP_ERRS_ONLY**。セットアップ時にパーティションにエラーが発生すると (パーティションへのアクセス時の障害、またはパーティション上の表スペースまたは表へのアクセス時の障害など)、ロード操作は障害のあるパーティションでは停止しますが、残りのパーティションでは実行を継続します。データのロード中にパーティションでエラーが発生すると、全体の操作が失敗し、それぞれのパーティションで整合性のある最終地点までロールバックします。
- **LOAD_ERRS_ONLY**。セットアップ時にパーティションにエラーが発生すると、ロード操作全体が失敗します。データのロード中にエラーが発生すると、エラーのあるパーティションは、整合性のある最終地点までロールバックされます。ロード操作は、障害が発生するか、すべてのデータがロードされるまで残りのパーティションで継続されます。すべてのデータがロードされたパーティションでは、ロード操作の後にデータが表示されることはありません。他のパーティションで発生するエラーが原因で、このトランザクションは打ち切られます。すべてのパーティションのデータは、ロード再始動操作が実行されるまで、表示されることはありません。これにより、新しくロードされるデータは、ロード操作が完了したパーティションで表示できるようになり、エラーが発生したパーティションでのロード操作が再開します。

注: **LOAD** コマンドで **ALLOW READ ACCESS** および **COPY YES** オプションの両方が指定される場合には、このモードは使用できません。

- **SETUP_AND_LOAD_ERRS**。このモードでは、セットアップまたはデータのロード時に生じるパーティション・レベルのエラーによって、影響を受けたパーティション上でのみ、処理が停止します。 **LOAD_ERRS_ONLY** モードの場合と同様、データのロード中にエラーが発生した場合、すべてのパーティションのデータは、ロード再始動操作が実行されるまで、表示されることはありません。

注: **LOAD** コマンドで **ALLOW READ ACCESS** および **COPY YES** オプションの両方が指定される場合には、このモードは使用できません。

- **NO_ISOLATION**。ロード操作時にエラーがあれば、トランザクションは失敗します。

STATUS_INTERVAL X

X は、読み取られたデータのボリュームを通知する頻度を示します。メジャー単位はメガバイト (MB) です。デフォルトは 100 MB です。有効な値は 1 から 4000 の整数です。

PORT_RANGE X

X は、内部通信のためのソケットの作成に使う TCP ポートの範囲を表します。デフォルトの範囲は 6000 から 6063 です。 **DB2ATLD_PORTS** DB2 レジストリー変数の値が起動時に定義される場合には、その値は **PORT_RANGE** ロード構成オプションの値で置き換えられます。 **DB2ATLD_PORTS** レジストリー変数の場合、範囲は以下のフォーマットで提供されます。

<lower-port-number>:<higher-port-number>

CLP からの場合は、以下のフォーマットです。

(lower-port-number, higher-port-number)

CHECK_TRUNCATION

プログラムが入出力時にデータ・レコードの切り捨てをチェックするように指定します。デフォルトの動作では、入出力時にはデータの切り捨てをチェックしません。

MAP_FILE_INPUT X

X は、パーティション化マップの入力ファイル名を指定します。このパラメーターはカスタマイズされたパーティション化マップの入ったファイルを示すため、パーティション化マップがカスタマイズされている場合にはこのパラメーターを指定しなければなりません。カスタマイズされたパーティション化マップを作成するには、**db2gpmmap** プログラムを使用してデータベース・システム・カタログ表からマップを抽出するか、または、LOAD コマンドの ANALYZE モードを使用して最適なマップを生成します。ロード操作を継続するには、その前に ANALYZE モードを使用して生成されるマップをデータベース内のそれぞれのデータベース・パーティションに移動する必要があります。

MAP_FILE_OUTPUT X

X は、パーティション化マップの出力ファイル名を示します。このパラメーターは、ANALYZE モードが指定される際に使用しなければなりません。すべてのデータベース・パーティションに均一に分散する最適なパーティション化マップが生成されます。この修飾子が指定されておらず ANALYZE モードが指定されている場合には、プログラムは終了してエラーを戻します。

TRACE X

データ変換プロセスとハッシュ値の出力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。デフォルトは 0 です。

NEWLINE

入力データ・ファイルが、それぞれのレコードが改行文字によって区切られた ASC ファイルであり、LOAD コマンドの RecLen パラメーターが指定されている場合に使用されます。このオプションが指定されると、それぞれのレコードの改行文字がチェックされます。また、RecLen パラメーターで指定されたレコード長もチェックされます。

DISTFILE X

このオプションを指定すると、LOAD ユーティリティーは、指定された名前のパーティション分散ファイルを生成します。パーティション分散ファイルには 4096 個の整数が入っており、それぞれはターゲット表のパーティション化マップの各項目に対応しています。ファイル内の各整数は、ロードされる入力ファイルの中で、対応するパーティション化マップ項目にハッシュされる行数を表します。この情報はデータの中のスキューを識別するのに役立ちます。さらに、ユーティリティーの ANALYZE モードを使って表の新しいパーティション化マップを生成すべきかどうか判断するのに役立ちます。このオプションを指定しない場合、ロード・ユーティリティーのデフォルト動作として、配布ファイルを生成しません。

注: このオプションを指定すると、最大で 1 つのパーティション化エージェントがロード操作のために使用されます。ユーザーによって複数のパーティション化エージェントが明示的に要求される場合、1 つだけが使用されます。

OMIT HEADER

パーティション・ファイルにパーティション化マップ・ヘッダーを組み込まないように指定します。指定されていなければ、ヘッダーが生成されます。

RUN STAT DBPARTNUM X

LOAD コマンドに STATISTICS YES パラメーターが指定された場合には、1 つのデータベース・パーティションでのみ統計が収集されます。このパラメーターは、統計を収集するデータベース・パーティションを指定します。値が -1 か、またはまったく指定されない場合には、出力パーティション・リストの最初のデータベース・パーティションで統計が収集されます。

関連タスク:

- 203 ページの『パーティション・データベース環境でのロードの使用』

関連資料:

- 「コマンド・リファレンス」の『REDISTRIBUTE DATABASE PARTITION GROUP コマンド』
- 113 ページの『LOAD』

パーティション・データベース・ロード・セッションの例

以下の例では、データベースに 0 から 4 の番号の付いた 4 つのパーティションがあります。データベース **WSDB** がすべてのパーティションで定義されており、表 **TABLE1** が、すべてのパーティションでも定義されているデフォルト・ノード・グループにあります。

例 1

パーティション 0 にあるユーザー・データ・ファイル load.del から TABLE1 にデータをロードするには、パーティション 0 に接続してから、以下のコマンドを発行します。

```
load from load.del of del replace into table1
```

ロード操作が正常に実行された場合、出力は以下のようになります。

Agent Type	Node	SQL Code	Result
LOAD	000	+00000000	Success.
LOAD	001	+00000000	Success.
LOAD	002	+00000000	Success.
LOAD	003	+00000000	Success.
PARTITION	001	+00000000	Success.
PRE_PARTITION	000	+00000000	Success.

RESULTS: 4 of 4 LOADs completed successfully.

Summary of Partitioning Agents:
Rows Read = 100000
Rows Rejected = 0
Rows Partitioned = 100000

Summary of LOAD Agents:
Number of rows read = 100000
Number of rows skipped = 0
Number of rows loaded = 100000
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows committed = 100000

出力では、それぞれのパーティションごとに 1 つのロード・エージェントがあり、それぞれが正常に実行されたことを示しています。また、コーディネーター・パーティションで実行する事前パーティション化エージェントが 1 つ、およびパーティション 1 で実行するパーティション化エージェントが 1 つあったことも示しています。これらのプロセスは、通常の SQL 戻りコード 0 を戻して正常に完了しました。統計のサマリーでは、事前パーティション化エージェントは 100,000 行を読み取り、パーティション化エージェントは 100,000 行をパーティション化し、ロード・エージェントによってロードされるすべての行の合計は、100,000 行であることを示します。

例 2

以下の例では、データは `PARTITION_ONLY` モードで `TABLE1` にロードされます。パーティション出力ファイルは、ディレクトリー `/udb/data` の出力パーティションのそれぞれに保管されます。

```
load from load.del of del replace into table1 partitioned db config mode
partition_only part_file_location /udb/data
```

ロード・コマンドからの出力は、以下のようになります。

Agent Type	Node	SQL Code	Result
LOAD_TO_FILE	000	+00000000	Success.
LOAD_TO_FILE	001	+00000000	Success.
LOAD_TO_FILE	002	+00000000	Success.
LOAD_TO_FILE	003	+00000000	Success.
PARTITION	001	+00000000	Success.
PRE_PARTITION	000	+00000000	Success.

Summary of Partitioning Agents:
Rows Read = 100000
Rows Rejected = 0
Rows Partitioned = 100000

出力では、それぞれの出力ノードで実行する `load-to-file` エージェントがあり、これらのエージェントが正常に実行されたことを示しています。コーディネーター・パーティションには事前パーティション化エージェントがあり、ノード 1 で実行する

パーティション化エージェントがあります。統計のサマリーでは、事前パーティション化エージェントによって 100,000 行が正常に読み取られ、パーティション化エージェントによって 100,000 行が正常にパーティションされたことを示しています。表には行がロードされなかったため、ロードされた行の数のサマリーは表示されません。

例 3

上記の PARTITION_ONLY ロード操作時に生成されたファイルをロードするには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config mode
load_only part_file_location /udb/data
```

ロード・コマンドからの出力は、以下のようになります。

Agent Type	Node	SQL Code	Result
LOAD	000	+00000000	Success.
LOAD	001	+00000000	Success.
LOAD	002	+00000000	Success.
LOAD	003	+00000000	Success.
RESULTS:	4 of 4 LOADs completed successfully.		

```
Summary of LOAD Agents:
Number of rows read      = 100000
Number of rows skipped   = 0
Number of rows loaded    = 100000
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 100000
```

出力では、それぞれの出力パーティションのロード・エージェントが正常に実行し、すべてのロード・エージェントによってロードされる行の数の合計が 100,000 であることを示しています。パーティション化は実行されなかったため、パーティションされる行のサマリーはありません。

例 4 - ロード操作の失敗

以下のロード・コマンドを発行したとします。

```
load from load.del of del replace into table1
```

ロード操作中に、ロード・パーティションの 1 つが表スペースでスペース不足になっているため、以下の出力が戻されます。

```
SQL0289N Unable to allocate new pages in table space "DMS4KT".
SQLSTATE=57011
```

Agent Type	Node	SQL Code	Result
LOAD	000	+00000000	Success.
LOAD	001	-00000289	Error. May require RESTART.
LOAD	002	+00000000	Success.

LOAD	003	+00000000	Success.
PARTITION	001	+00000000	Success.
PRE_PARTITION	000	+00000000	Success.
RESULTS:	3 of 4 LOADs completed successfully.		

Summary of Partitioning Agents:
 Rows Read = 0
 Rows Rejected = 0
 Rows Partitioned = 0

Summary of LOAD Agents:
 Number of rows read = 0
 Number of rows skipped = 0
 Number of rows loaded = 0
 Number of rows rejected = 0
 Number of rows deleted = 0
 Number of rows committed = 0

出力では、ロード操作でエラー SQL0289 が戻されたことを示します。このパーティションのサマリーでは、パーティション 1 でスペースが足りないことを示しています。デフォルト・エラー分離モードは NO_ISOLATION であるため、ロード操作はすべてのパーティションで打ち切られ、ロードの再始動操作またはロード終了操作が呼び出されなければなりません。パーティション 1 の表スペースのコンテナにスペースが追加された場合、以下のようにしてロード操作を再始動できます。

```
load from load.del of del restart into table1
```

関連タスク:

- 203 ページの『パーティション・データベース環境でのロードの使用』

関連資料:

- 113 ページの『LOAD』

移行およびバックレベル互換性

パーティション・データベース環境でのロード・コマンドの使用

バージョン 8 では、LOAD コマンドに変更が加えられました。これは **db2atld** ユーティリティの機能と置き換わるものです。その結果、使用される用語にもいくつかの変更があります。

db2atld ユーティリティでは、スプリッター および分割ファイル (スプリッターの出力) を使用していました。LOAD コマンドを使用する際には、同等のものとして、パーティション化エージェント およびパーティション・ファイル を使用します。また、LOAD コマンドを使用する際には、データ・ファイルは、分割 ではなく、パーティション化 されていると表現します。ロード・モードおよび構成オプションは、この変更を反映して名前が付けられています。たとえば、オートローダーに SPLIT_ONLY モードおよび SPLIT_NODES オプションが指定されている場合、LOAD コマンドには PARTITION_ONLY モードおよび PARTITIONING_DBPARTNUMS オプションが指定されるようになりました。

オートローダーの **FORCE** オプションと同等のものは、**ISOLATE_PART_ERRS** ロード構成オプションです。このオプションを使用すると、ユーザーは 4 つのタイプのパーティション・エラー分離を指定できます。これら (**SETUP_AND_LOAD_ERRS**) の 1 つは、**FORCE YES** オプションによって提供されるエラー分離に類似しています。

db2atld の使用

db2atld ユーティリティーは今までどおり使用可能であり、バージョン 8 より前で受け入れていたものと同じ構文の構成ファイルを受け入れます。 **db2atld** を使用する際には、以下を考慮してください。

1. **db2atld** は、現在、複数パーティション・データベースにロードする際に接続を 1 つだけ確立します。バージョン 8 以前は、 **db2atld** は、出力パーティションごとに 1 つの接続を確立していました。
2. **db2atld** は、パーティション・データベース環境で使用されるロード・ユーティリティーと同じ、単一トランザクション・モデルを使用します。
3. **db2atld** 実行プログラムを介してパーティション・データベースのロードを開始すると、 **LOAD_ONLY_VERIFY_PART** モードおよび拡張ロード・ユーティリティーで提供されるその他の新機能は使用できません。
4. 以前にサポートされているオートローダー構成パラメーターはすべて認識されません。
5. **db2atld** によりコンソールに表示される情報は多少異なります。たとえば、 **SPLIT_AND_LOAD** モードでは、パーティション化統計およびロード統計は常に表示されるようになりました。以前は、統計に矛盾がある場合にのみ表示されていました。さらに、ロード操作時に状況インターバル情報はコンソールにダンプされず、その代わりに事前パーティション化エージェント・メッセージ・ファイル (<messagefile>.PREP.<nodenum>) にダンプされます。スプリッター・プロセスおよびロード・プロセスの作成に関する状況情報は、使用できなくなりました。
6. クロス・マウントされるパスから **db2atld** 実行プログラムを立ち上げる必要はなくなりました。
7. パーティション・データベース環境にデータをロードする際、入力ソースの最大数は 999 です。
8. バージョン 8 より前では、 **db2atld** ユーティリティーの **LOAD_ONLY** モードが使用される場合に、各パーティションの分割ファイルの名前は **.00xxx** または **.xxx** という拡張子が可能でした。ここで **xxx** はパーティション番号です。バージョン 8 以降では、 **.00xxx** 拡張子はもはやサポートされていません。3 桁の拡張子 (**.xxx**) だけを指定できます。

DB2®_PARTITIONEDLOAD_DEFAULT 登録変数を使用したバージョン 8 以前のロード動作への復帰

バージョン 8 より前では、ロードされるパーティションに有効なパーティション・ヘッダー情報が入力データ・ファイルに組み込まれていれば、ロード・ユーティリティーを使用して複数パーティション・データベースの単一パーティションをロードすることができました。このヘッダーは通常、オートローダーの **SPLIT_ONLY** モード、または **db2split** ユーティリティーを介して作成されていました。パーティション・ヘッダーは、残りのデータがロードされる前にロード・ユーティリティー

ーによって検査されました。EEE 環境で単一パーティション・データベース・パーティション・グループに常駐する表では、 `noheader` ファイル・タイプ修飾子が指定されていれば、パーティション・ヘッダー情報のないデータ・ファイルをロードすることができました。

LOAD コマンドの動作は、その後変更されました。パーティション・データベース環境では、パーティション・データベース・ロード構成オプションが指定されない場合には、表が定義されるすべてのパーティションでロード操作が実行されるものと見なされます。入力ファイルにはパーティション・ヘッダーは必要ではなく、MODE オプションはデフォルトの `PARTITION_AND_LOAD` になります。単一パーティションをロードするには、`OUTPUT_DBPARTNUMS` オプションを指定しなければなりません。

LOAD コマンドのバージョン 8 より前の動作を、パーティション・データベース環境で保守することが可能です。これにより、パーティション・データベース構成オプションを追加で指定しなくても、有効なパーティション・ヘッダーのあるファイルを単一データベース・パーティションにロードすることができます。これは、`DB2_PARTITIONEDLOAD_DEFAULT` レジストリー変数を `NO` に設定することにより発行できます。単一データベース・パーティションに LOAD コマンドを発行する既存のスクリプトを変更したくない場合、このオプションを使用することをお勧めします。たとえば、4 パーティション・ノード・グループに常駐する表のパーティション 3 にパーティション・ファイルをロードするには、以下のコマンドを発行します。

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

それから、DB2 コマンド行プロセッサから以下のコマンドを発行します。

```
CONNECT RESET
```

```
SET CLIENT CONNECT_NODE 3
```

```
CONNECT TO DB MYDB
```

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

関連資料:

- 113 ページの『LOAD』

パーティション・データベース環境でのデータのロード - ヒント

パーティション・データベース環境に表をロードする前に、以下のことを考慮してください。

- パーティション・ロード構成オプションに慣れるために、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とき、パーティション化に使用するデータベース・パーティションは 1 つだけにしてください。並列パーティション化では、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、ロード・ユーティリティーはデフォルトで単一パーティション化エージェントを選択します。

- 複数の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・パーティション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。
- `ISOLATE_PART_ERRS` オプションを `SETUP_ERRS_ONLY` または `SETUP_AND_LOAD_ERRS` に設定することにより、(始動時に) ロード操作でロード・パーティションや関連する表スペースまたは表がオフラインになっていることを検出した場合でも、パーティション・データベース・ジョブを続行させることができます。
- `STATUS_INTERVAL` パーティション・ロード構成オプションを使用して、パーティション・データベース・ロード・ジョブの進行をモニターします。ロード操作は、指定された時間間隔でメッセージを生成し、事前パーティション化エージェントによって読み取られたデータの量をメガバイト単位で表示します。これらのメッセージは、事前パーティション化エージェント・メッセージ・ファイルにダンプされます。ロード操作中にこのファイルの内容を表示するには、コーディネーター・パーティションに接続してから、ターゲット表に対して `LOAD QUERY` コマンドを発行します。
- (`PARTITIONING_DBPARTNUMS` オプションで定義される) パーティション化ロードが、(`OUTPUT_DBPARTNUM` オプションで定義される) ロード・パーティションと異なっている場合、CPU サイクルに対する競合が少なくなるため、パフォーマンスの改善が望めます。パーティション・データベースにデータをロードする際には、ロード・ユーティリティ自体をパーティション化操作にもロード操作にも関係しないデータベース・パーティションで起動してください。
- `LOAD` コマンドに `MESSAGES` パラメーターを指定すると、事前パーティション化、パーティション化、およびロード・エージェントからのメッセージを保管して、ロード操作の終了時に参照できるようにします。ロード操作中にこれらのファイルの内容を表示するには、希望するパーティションに接続してから、ターゲット表に対して `LOAD QUERY` コマンドを発行します。
- ロード・ユーティリティは、統計情報を収集する出力データベース・パーティションを 1 つだけ選択します。そのパーティションを指定するには、`RUN_STAT_DBPARTNUM` パーティション・ロード・データベース構成オプションを使用できます。
- パーティション・データベース環境でデータをロードする場合、事前に設計アドバイザーを実行して、各表ごとに最適なパーティションを判別することができます。詳しくは、『設計アドバイザー』を参照してください。

トラブルシューティング

ロード・ユーティリティが停止した場合には、以下を実行できます。

- `STATUS_INTERVAL` パラメーターを使用して、パーティション・データベース・ロード操作の進行をモニターする。状況インターバル情報は、コーディネーター・パーティションの事前パーティション化エージェント・メッセージ・ファイルにダンプされます。
- パーティション化エージェント・メッセージをチェックして、それぞれのパーティション・データベース・パーティションでのパーティション化エージェント・

プロセスの状況を調べる。エラーなしでロードが進行しており、TRACE オプションが設定された場合には、これらのメッセージ・ファイル内に多くのレコードに関するトレース・メッセージがあるはずです。

- ロード・メッセージ・ファイルをチェックして、ロード・エラー・メッセージがあるかどうかを調べる。

注: これらのファイルが存在するためには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。

- ロード・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場合、現在のロード操作を中断する。

関連資料:

- 137 ページの『LOAD QUERY』
- 113 ページの『LOAD』

第 5 章 DB2 Data Links Manager 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 Data Links Manager 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利用できるファイル・フォーマットについては、277 ページの『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・フォーマット』を参照してください。

以下のトピックを取り上げています。

- 『エクスポートを使用した DB2 Data Links Manager データの移動 - 概念』
- 230 ページの『エクスポートを使用した DB2 Data Links Manager データの移動』
- 231 ページの『インポートを使用した DB2 Data Links Manager データの移動』
- 232 ページの『ロードを使用した DB2 Data Links Manager データの移動』。

エクスポートを使用した DB2 Data Links Manager データの移動 - 概念

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。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 `d1fm/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 Data Links Manager のファイル名のリストが入れられます。

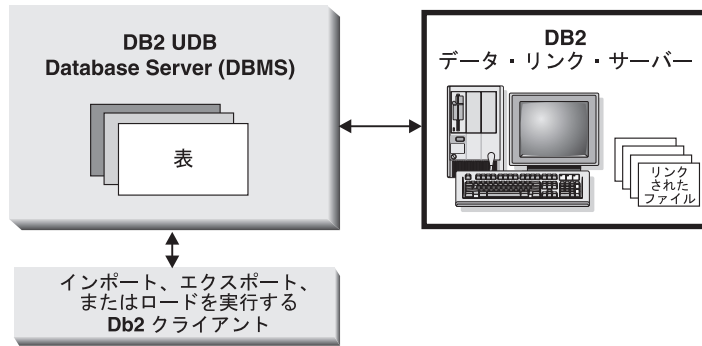


図 11. DB2 Data Links Manager のデータの移動： 表データがデータベース内にあり、一方 DATALINK 列の参照するファイルがデータ・リンク・サーバー上にあるため、エクスポート、インポート、およびロードの各ユーティリティーは、データベース上のデータを移動することに加えて、対応するデータ・リンク・サーバー上のデータ・ファイルも移動する必要があります。

Windows® オペレーティング・システムでは、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、`d1fm¥YYYYMMDD¥HHMMSS` という名前の新しいディレクトリーに作成されます。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 Data Links Manager 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。Windows オペレーティング・システムでは、ただ 1 つのコントロール・ファイルを、参照されているすべてのデータ・リンク・サーバーに移送する必要があります。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・サーバーのコントロール・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。

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 列の値のフォーマットは、インポートおよびロード・ユーティリティーで使用されるフォーマットと同じです。DATALINK 列の値が NULL の場合は、他の NULL 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。 Windows 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 つのコントロール・ファイルが作成されます。

以下のようにして、**dlfm_export** ユーティリティを使って 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リストアするには、**dlfm_import** という補足ユーティリティを使用します。アーカイブ・ファイルと同じデータ・リンク・サーバーにリストアするか別のサーバーにリストアするかに関係なく、このユーティリティを実行する必要があります。

以下のようにして、**dlfm_import** ユーティリティ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 はファイルのリストアに使うアーカイブ・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・サーバーにリストアする場合、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** ユーティリティを実行したサーバーとは異なっている場合がありますからです。 **DLFM_FS_ENVIRONMENT** レジストリー変数を適切に設定してから、 **dlfm_import** ユーティリティを実行してください。

注: **dlfm_export** ユーティリティを実行したデータ・リンク・サーバーとは別のサーバーで **dlfm_import** ユーティリティを実行した場合、ファイルは正しいパスにリストアされます。インポートしているマシンにユーザー ID のうち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。

以下の表は、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リストア	DLFMY 上でアーカイブからリストア	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 パラメーターを使用して、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。			

関連タスク:

- 3 ページの『エクスポートの使用』
- 230 ページの『エクスポートを使用した DB2 Data Links Manager データの移動』

関連資料:

- 14 ページの『db2Export - エクスポート』

エクスポートを使用した DB2 Data Links Manager データの移動

手順:

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** ユーティリティを実行します。こうすると、データ・リンク・ファイル・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。このユーティ

リティアーは、アーカイブされているファイルの ACL 情報はキャプチャーしません。 **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
```

関連概念:

- 1 ページの『エクスポートの概要』
- 227 ページの『エクスポートを使用した DB2 Data Links Manager データの移動 - 概念』

関連タスク:

- 231 ページの『インポートを使用した DB2 Data Links Manager データの移動』
- 232 ページの『ロードを使用した DB2 Data Links Manager データの移動』

インポートを使用した DB2 Data Links Manager 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、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。

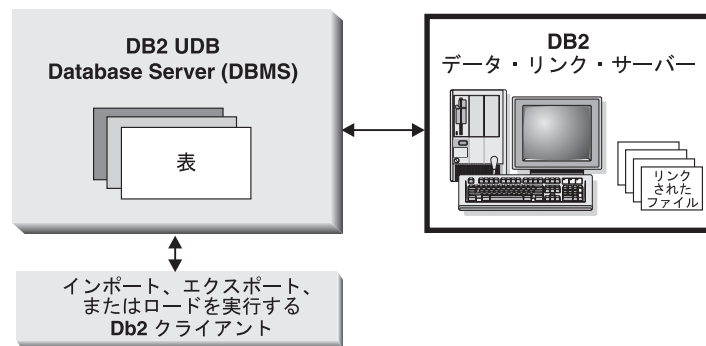


図 12. DB2 Data Links Manager のデータの移動：表データがデータベース内にあり、一方 DATALINK 列の参照するファイルがデータ・リンク・サーバー上にあるため、エクスポート、インポート、およびロードの各ユーティリティーは、データベース上のデータを移動することに加えて、対応するデータ・リンク・サーバー上のデータ・ファイルも移動する必要があります。

手順:

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・ファイル・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. 必要な場合、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場合、データ・リンク・サーバー名を更新する必要はありません。)
4. DB2 Data Links Manager 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。

インポート・ユーティリティーがターゲット・データベースに対して実行される場合、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。

関連概念:

- 29 ページの『インポートの概要』

関連タスク:

- 230 ページの『エクスポートを使用した DB2 Data Links Manager データの移動』
- 232 ページの『ロードを使用した DB2 Data Links Manager データの移動』

ロードを使用した DB2 Data Links Manager データの移動

手順:

FILE LINK CONTROL で定義された DATALINK 列を備えた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 Data Links Manager がインストールされていることを確認します。
2. データベースが DB2 Data Links Manager 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 Data Links Manager に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 Data Links Manager 構成ファイルの中で登録します。

ロード操作中に失敗したリンクはデータ保全性違反と見なされ、ユニーク索引違反と同じ方法で処理されます。そのため、1 つ以上の DATALINK 列の入った表をロードする場合の特別な例外が定義されています。

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

関連資料:

- 113 ページの『LOAD』

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 iSeries 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。

以下のトピックを取り上げています。

- ・ 『プラットフォーム間のデータの移動 - ファイル・フォーマットの考慮事項』
- ・ 237 ページの『DB2 Connect によるデータの移動』
- ・ 239 ページの『db2move - データベース移動ツール』
- ・ 245 ページの『db2relocatedb - データベースの再配置』
- ・ 250 ページの『型付き表間のデータ移動』
- ・ 255 ページの『レプリケーションを使ったデータの移動』
- ・ 257 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』。
- ・ 259 ページの『カーソル・ファイル・タイプを使用したデータの移動』。

プラットフォーム間のデータの移動 - ファイル・フォーマットの考慮事項

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・フォーマットの考慮事項について説明します。

PC/IXF ファイル・フォーマット

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・フォーマットです。PC/IXF を使用すると、ロード・ユーティリティやインポート・ユーティリティは、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・ユーティリティは Intel フォーマットの数値データによるファイルを作成し、インポート・ユーティリティはこのフォーマット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 製品はバイト反転を使用して数値を Intel フォーマットから非 Intel フォーマットに変換します。

DB2® の UNIX ベースのインプリメンテーションは、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。そうしない場合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 for Windows® によってインポートできます。

区切り付き ASCII (DEL) ファイル・フォーマット

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ります。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用すると、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入っていれば、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想定外の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・フォーマットを使用してください。

WSF ファイル・フォーマット

WSF フォーマット・ファイルの数値データは、Intel マシン形式を使って保管されます。このフォーマットでは、異なる Lotus® オペレーティング環境 (たとえば Intel ベースと UNIX ベースのシステム) で Lotus WSF ファイルを転送および使用できます。

このように内部フォーマット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 Lotus 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・フォーマット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・フォーマットを使用してください。

関連資料:

- 277 ページの『エクスポート/インポート/ロード・ユーティリティのファイル・フォーマット』

DB2 Connect 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 Connect (ホストとワークステーションの間のデータ転送のゲートウェイ) を使用できます (図 13 を参照)。

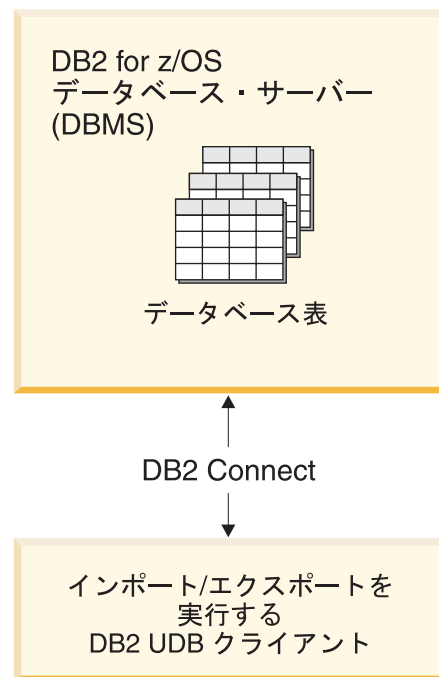


図 13. DB2 Connect によるインポート/エクスポート

DB2 のエクスポートおよびインポート・ユーティリティを使用すると、ホストまたは iSeries サーバー・データベースから DB2 Connect 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・フォーマット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、ホストまたは iSeries サーバー・データベースから PC/IXF ファイルにデータをエクスポートして、さらにそれを DB2 for Windows 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 Connect 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、コンパウンド 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。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。

制約事項:

DB2 Connect 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ターゲット表が作成されていなければなりません。ソース表の属性を取得するには、**db2look** ユーティリティーを使用できます。DB2 Connect 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。

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。

混合データ (1 バイト・データと 2 バイト・データの両方の入った列) 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はない限り、文字タイプの可変長列を使用することをお勧めします。

ワークステーションからホスト・サーバーへのデータの移動:

データをホストまたは AS/400 および iSeries サーバー・データベースに移動するには、以下を実行してください。

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 ファイルをホスト・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。

ホスト・サーバー・データベースからワークステーションにデータを移動するには、次のようにします。

1. ホスト・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。

例

以下の例では、ワークステーションからホストまたは AS/400 および iSeries サーバー・データベースにデータを移動する方法を示します。

1. 次のコマンドを発行して、外部 IXF フォーマットにデータをエクスポートします。

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

2. 次のコマンドを発行して、ターゲット DB2 UDB サーバーに DRDA 接続を確立します。

```
db2 connect to cbc664 user admin using xxx
```

3. これがまだ存在していない場合には、ターゲット DB2 UDB サーバーにターゲット表を作成します。

```
CREATE TABLE mydb.staff (ID SMALLINT NOT NULL, NAME VARCHAR(9),  
DEPT SMALLINT, JOB CHAR(5), YEARS SMALLINT, SALARY DECIMAL(7,2),  
COMM DECIMAL(7,2))
```

4. データをインポートするには、以下のコマンドを発行します。

```
db2 import from staff.ixf of ixf insert into mydb.staff
```

IXF フォーマットのファイルからデータの各行が読み取られ、表 mydb.staff に行を挿入するために、SQL INSERT ステートメントが発行されます。すべてのデータがターゲット表に挿入されるまで、単一行が引き続き挿入されます。

詳細については、「Moving Data Across the DB2 Family」という IBM レッドブックを参照してください。このレッドブックは、
<http://www.redbooks.ibm.com/redbooks/SG246905.html> という URL にあります。

関連概念:

- 235 ページの『プラットフォーム間のデータの移動 - ファイル・フォーマットの考慮事項』

関連資料:

- 9 ページの『EXPORT』
- 41 ページの『IMPORT』

db2move - データベース移動ツール

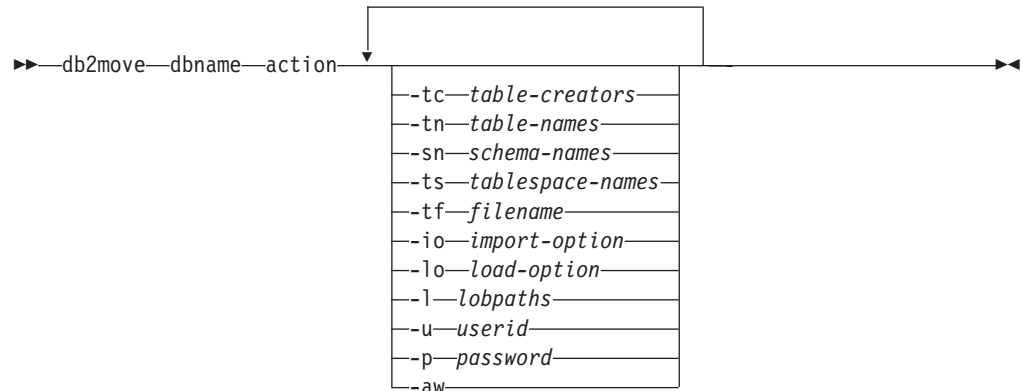
このツールは、ワークステーション上にある DB2 データベース間で、大量の表の移動を容易にします。また、特定のデータベースのシステム・カタログ表を照会し、すべてのユーザー表のリストをコンパイルします。そして、これらの表を PC/IXF DEL フォーマット・ファイルでエクスポートします。PC/IXF ファイルは、同じシステム上の別のローカル DB2 データベースにインポートまたはロードするか、または別のワークステーション・プラットフォームに転送し、そのプラットフォームで DB2 データベースにインポートまたはロードすることができます。

注: 構造型列がある表は、このツールを使用しても移動しません。

権限:

このツールは、ユーザーから要求されるアクションにしたがって、DB2 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、要求元ユーザー ID には、これらの API に求められる正しい権限がなければなりません。この権限がないと、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 のいずれかです。

-tc 表の作成者。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。指定されると、このオプションでリストされる作成者が作成する表のみがエクスポートされます。指定されない場合、デフォルトでは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、それぞれをコンマで区切る必要があります。作成者 ID 間にブランクを入れることはできません。指定できる作成者の最大数は 10 です。このオプションを "-tn" 表名オプションとともに使用すると、エクスポートする表を選択できます。

アスタリスク (*) は、ストリング中のどこにでも入れられるワイルドカード文字として使用できます。

-tn 表名。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。指定されると、指定されたストリング中の名前と完全に一致する名前を持つ表のみがエクスポートされます。指定されない場合、デフォルトでは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、それぞれをコンマで区切る必要があります。表名間にブランクを入れることはできません。指定できる表名の最大数は 10 です。このオプションを "-tc" 表作成者オプションとともに使用すると、エクスポートする表を選択できます。 **db2move** は、名前が指定された表名と一致し、かつ作成者が指定された表作成者と一致する表のみをエクスポートします。

アスタリスク (*) は、ストリング中のどこにでも入れられるワイルドカード文字として使用できます。

-sn スキーマ名。デフォルトはすべてのスキーマです。

これが指定されると、完全に一致するスキーマ名の表だけがエクスポートされます。スキーマ名の部分にアスタリスク・ワイルド・カード文字 (*) が使用された場合は、それがパーセント記号 (%) に変更され、WHERE 文節の LIKE 述部にパーセント付きの表名が使用されます。指定されない場合、デフォルトではすべてのスキーマを使用します。複数のスキーマ名を指定する場合は、それぞれの名前をコンマで区切る必要があります。複数のスキーマ名の間にブランクを使用することはできません。指定できるスキーマ名の最大数は 10 です。-tn または -tc オプションと合わせて使用する場合、**db2move** は、スキーマが指定されたスキーマ名と一致し、作成者が指定された作成者と一致する表だけをエクスポートします。

注: 8 文字より短いスキーマ名は、8 文字の長さになるまで埋め込まれます。たとえば、'fred' のようなスキーマ名の場合、アスタリスクを使用するときは、"-sn fr*d" ではなく "-sn fr*d*" のような指定が必要になります。

2 **-ts** 表スペース名。デフォルトはすべての表スペースです。
2
2 これは EXPORT アクションのみです。このオプションが指定されると、指
2 定した表スペースにある表だけがエクスポートされます。表スペース名の部
2 分にアスタリスク・ワイルド・カード文字 (*) が使用された場合は、それが
2 パーセント記号 (%) に変更され、WHERE 文節の LIKE 述部にパーセント
2 付きの表名が使用されます。-ts オプションが指定されない場合、デフォ
2 ルトではすべての表スペースを使用します。複数の表スペース名を指定する場
2 合は、それぞれの名前をコンマで区切る必要があります。複数の表スペース
2 名の間にブランクを使用することはできません。指定できる表スペース名の
2 最大数は 10 です。

2 **注:** 8 文字より短い表スペース名は、8 文字の長さになるまで埋め込まれま
2 す。たとえば、'mytb' のような表スペース名の場合、アスタリスクを使
2 用するときは、"-sn my*b" ではなく "-ts my*b*" のような指定が必要
2 になります。

2 **-tf** filename
2
2 これは EXPORT アクションのみです。指定されると、特定のファイルにリ
2 ストされている表だけがエクスポートされます。表は 1 行に 1 つずつリス
2 トする必要があり、各表は完全に修飾する必要があります。以下は、ファイ
2 ルの内容の例です。
2 "SCHEMA1"."TABLE NAME1"
2 "SCHEMA NAME77"."TABLE155"

-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 および REPLACE です。

-l LOB パス。デフォルトは現行ディレクトリです。
このオプションは、LOB ファイルが (EXPORT の一部として) 作成される
か、または (IMPORT または LOAD の一部として) 検索される絶対パス名

を指定します。複数の LOB パスを指定する場合、それぞれをコンマで区切る必要があります。LOB パス間にブランクを入れることはできません。最初のパスでスペースを使い尽くした場合 (EXPORT 中)、またはパスでファイルが見つからない場合 (IMPORT または LOAD 中)、2 番目のパスが使用される、という方法でパスが使用されます。

アクションが EXPORT で LOB パスが指定される場合、LOB パス・ディレクトリーのすべてのファイルが削除され、ディレクトリーは除去され、新しいディレクトリーが作成されます。指定されない場合、現行ディレクトリーが LOB パスに使用されます。

- u** ユーザー ID。デフォルトはログオン・ユーザー ID です。
ユーザー ID とパスワードはどちらも任意指定です。しかし、一方を指定した場合、他方も必ず指定する必要があります。コマンドがリモート・サーバーに接続するクライアント上で実行される場合、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・パスワードです。
ユーザー ID とパスワードはどちらも任意指定です。しかし、一方を指定した場合、他方も必ず指定する必要があります。コマンドがリモート・サーバーに接続するクライアント上で実行される場合、ユーザー ID とパスワードを指定する必要があります。
- aw** 警告を許します。'-aw' が指定されていない場合、エクスポート中に警告があった表は db2move.lst ファイルに組み込まれません (表の .ixf ファイルと .msg ファイルが生成されていても)。しかし、あるシナリオ (データ切り捨てなど) では、そのように警告があった表でも db2move.lst ファイルに組み込んでしまいたい場合があります。そのようなとき、このオプションを指定すると、エクスポート中に警告を受け取った表を .lst ファイルに組み込むことができます。

例:

- db2move sample export

この例では、SAMPLE データベースのすべての表をエクスポートします。すべてのオプションにデフォルトが使用されます。

- db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2

この例では、"userid1" または "us%rid2" のようなユーザー ID で作成され、"tbname1" という名前、または "%tbname2" のような表名を持つすべての表がエクスポートされます。

- db2move sample import -l D:¥LOBPATH1,C:¥LOBPATH2

この例は、Windows オペレーティング・システムのみに適用されます。コマンドは、SAMPLE データベース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" で、LOB ファイルが検索されます。

- db2move sample load -l /home/userid/lobpath,/tmp

この例は、UNIX ベースのシステムのみに適用されます。コマンドは、SAMPLE データベースのすべての表をロードします。 /home/userid/lobpath サブディレクトリーと tmp サブディレクトリーの両方で、LOB ファイルが検索されます。

- db2move sample import -io replace -u userid -p password

この例では、SAMPLE データベースのすべての表が REPLACE モードでインポートされます。指定されたユーザー ID およびパスワードが使用されます。

使用上の注意:

このツールはユーザーが作成した表をエクスポート、インポート、またはロードします。データベースが、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・システムに複製される場合、**db2move** によって表の移動が容易になります。表と関連する他のすべてのオブジェクト、たとえば別名、ビュー、トリガーなども移動する必要があります。REPLACE_CREATE オプションを指定したインポート・ユーティリティーを使って、ターゲット・データベース上で表を作成する場合、『インポートを使用した、エクスポートされる表の再作成』に概略されている制限を受けます。REPLACE_CREATE オプションの使用時の **db2move** インポート・フェーズ中に想定外のエラーが生じた場合、該当する tabnnn.msg メッセージ・ファイルを調べて、表の作成に対する制限事項が原因でエラーが起きたかどうかを確かめてください。

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 パラメーターが lobsinfile に設定されます。つまり、LOB データが PC/IXF ファイルとは別に保持されます。LOB ファイルでは、26,000 のファイル名を使用することができます。

LOAD アクションは、データベースおよびデータ・ファイルが常駐するマシンでローカルに実行する必要があります。ロード API が **db2move** によって呼び出されると、CopyTargetList パラメーターが NULL に設定されます。つまり、コピーは行われません。logretain がオンである場合、ロード操作を後でロールフォワードすることはできません。ロードされた表が常駐する表スペースはバックアップ・ペンディング状態にされ、アクセスできません。表スペースをバックアップ・ペンディング状態から解除するには、全データベースのバックアップまたは表スペースのバックアップが必要です。

注: 'db2move import' のパフォーマンスは、デフォルトのバッファ・プール IBMDEFAULTBP を調整し、構成パラメーター *sortheap*、*util_heap_sz*、*logfilsize*、および *logprimary* を更新することによって改善される場合があります。

EXPORT 使用時に必要とされるファイル/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 オリジナル表名のリスト、その対応する PC/IXF ファイル名 (tabnnn.ixf)、およびメッセージ・ファイル名 (tabnnn.msg)。このリスト、エクスポートされた PC/IXF ファイル、および LOB フ

ファイル (tabnnnc.yyy) は、**db2move** IMPORT または LOAD アクションへの入力として使用されます。

tabnnn.ixf 特定の表の、エクスポートされる 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 特定の表の、エクスポートされる LOB ファイル。

"nnn" は表番号です。"c" はアルファベットの文字です。"yyy" は 001 から 999 の範囲内の数値です。

これらのファイルは、エクスポートされている表に LOB データが入っている場合のみ作成されます。作成されると、これらの LOB ファイルは "lobpath" ディレクトリーに入れられます。

LOB ファイルには、合計 26,000 の使用可能な名前があります。

system.msg ファイルまたはディレクトリー・コマンドを作成または削除するための、システム・メッセージの入ったメッセージ・ファイル。これは、アクションが EXPORT で、LOB パスが指定される場合のみ使用されます。

IMPORT 使用時に必要とされるファイル/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 使用時に必要とされるファイル/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 LOAD メッセージ・ファイル。

関連資料:

- 「コマンド・リファレンス」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』

db2relocatedb - データベースの再配置

ユーザー提供の構成ファイルに指定されたように、データベースを名前変更、またはデータベース、またはデータベースの一部（たとえばコンテナおよびログ・ディレクトリ）を再配置します。このツールは、DB2 インスタンスおよびデータベース・サポート・ファイルに、必要な変更を行います。

許可:

なし

コマンド構文:

►►db2relocatedb—f—configFilename—►►

コマンド・パラメーター:

-f configFilename

データベースを再配置するために必要な構成情報の入ったファイルの名前を指定します。これは、相対ファイル名でも絶対ファイル名でもかまいません。構成ファイルのフォーマットは以下のとおりです。

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
```

ここで、

DB_NAME

再配置されるデータベースの名前を指定します。データベース名が変更する場合は、古い名前と新規の名前の両方を指定する必要があります。このフィールドは必須です。

DB_PATH

再配置されるデータベースのパスを指定します。このパスは、データベースが初めに作成された場所です。データベース・パスが変更される場合、古いパスと新規のパスの両方を指定する必要があります。このフィールドは必須です。

INSTANCE

データベースが存在する場所のインスタンスを指定します。データベースが新規のインスタンスに移動される場合、古いインスタンスと新規のインスタンスの両方を指定する必要があります。このフィールドは必須です。

NODENUM

変更されるデータベース・ノードのノード番号を指定します。デフォルトは 0 です。

LOG_DIR

ログ・パスのロケーション内の変更を指定します。ログ・パスが変更されている場合、古いパスと新規のパスの両方を指定する必要が

あります。ログ・パスがデータベース・パスの下にある場合、パスは自動的に更新されるので、この指定はオプションです。

CONT_PATH

表スペース・コンテナのロケーション内の変更を指定します。古いコンテナと新規のコンテナの両方を指定する必要があります。複数コンテナ・パスが変更される場合、複数 CONT_PATH 行が提供されます。コンテナ・パスがデータベース・パスの下にある場合、パスは自動的に更新されるので、この指定はオプションです。同じ古いパスが共通の新規パスで置換される場所で、複数のコンテナに変更を行う場合、単一の CONT_PATH 項目が使用されます。このような場合、古いパス、新規パスの両方にアスタリスク (*) をワイルドカードとして使用できます。

注: ブランク行またはコメント文字 (#) で開始される行は無視されます。

使用上の注意:

データベースが属するインスタンスが変更されている場合、インスタンスおよびデータベース・サポート・ファイルへの変更が行われるのを確認するために、このコマンドの実行前に以下を行う必要があります。

- データベースが他のインスタンスに移動されている場合は、新規のインスタンスを作成します。
- 新規インスタンスが常駐するシステムにコピーされているデータベースに属するファイル/デバイスをコピーします。パス名を変更する必要があります。ただし、データベース・ファイルの移動先のディレクトリー内にデータベースがすでにある場合、不用意に既存の sqlbdir ファイルを上書きしてしまって、既存のデータベースへの参照を除去する可能性があります。そのような事態になった場合、**db2relocatedb** ユーティリティーを使用することはできません。その場合、**db2relocatedb** の代わりにリダイレクト・リストアを使用します。
- インスタンス所有者に所有されるように、コピーされたファイル/デバイスのアクセス権を変更します。

インスタンスが変更されている場合、ツールは新規のインスタンス所有者によって実行される必要があります。

パーティション・データベース環境では、このツールは変更が必要なすべてのパーティションに対して実行される必要があります。それぞれのパーティションには、変更されているパーティションの NODENUM 値の入った、別々の構成ファイルが提供されなければなりません。たとえば、データベースの名前が変更されている場合、すべてのパーティションは影響を受け、それぞれのパーティションの、別々の構成ファイルで **db2relocatedb** コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナが移動されている場合、**db2relocatedb** コマンドは、そのパーティションで一度だけ実行される必要があります。

例:

例 1

データベース TESTDB の名前を PRODDB に、パス /home/db2inst1 にあるインスタンス db2inst1 で変更するには、以下の構成ファイルを作成します。

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

構成ファイルを relocate.cfg として保管し、以下のコマンドを使用して、データベース・ファイルへの変更を行います。

```
db2relocatedb -f relocate.cfg
```

例 2

データベース DATAB1 をパス /dbpath のインスタンス jsmith からインスタンス prodinst に移動するには、以下のようになります。

1. ディレクトリー /dbpath/jsmith 内のファイルを /dbpath/prodinst に移動します。
2. 以下の構成ファイルと **db2relocatedb** コマンドを使用して、データベース・ファイルに変更を行います。

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

例 3

パス /databases/PRODDB のインスタンス inst1 内に存在するデータベース PRODDB です。2 つの表スペース・コンテナのロケーションを、以下のように変更する必要があります。

- SMS コンテナ /data/SMS1 を /DATA/NewSMS1 に移動する必要があります。
- DMS コンテナ /data/DMS1 を /DATA/DMS1 に移動する必要があります。

物理ディレクトリーおよびファイルが、新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと **db2relocatedb** コマンドを使用して、データベース・ファイルに変更を行います。

```
DB_NAME=PRODDB
DB_PATH=/databases/PRODDB
INSTANCE=inst1
NODENUM=0
CONT_PATH=/data/SMS1,/DATA/NewSMS1
CONT_PATH=/data/DMS1,/DATA/DMS1
```

例 4

インスタンス db2inst1 に存在するデータベース TESTDB は、パス /databases/TESTDB に作成されました。表スペースは、以下のコンテナと共に作成されました。

```
TS1
TS2_Cont0
TS2_Cont1
```


db2relocatedb - データベースの再配置

```
/databases/TESTDB/TS3_Cont0
/databases/TESTDB/TS4/Cont0
/Data/TS5_Cont0
/dev/rTS5_Cont1
```

TESTDB は新規システムに移動されます。新規システムのインスタンスは **newinst** になり、データベースのロケーションは **/DB2** になります。

データベースを移動する場合、**/databases/TESTDB/db2inst1** ディレクトリーに存在するすべてのファイルは、**/DB2/newinst** ディレクトリーに移動する必要があります。これは、最初の 5 つのコンテナが、この移動の一部として再配置されることを意味します。(最初の 3 つはデータベース・ディレクトリーに相対で、次の 2 つはデータベース・パスに相対です。) これらのコンテナがデータベース・ディレクトリーまたはデータベース・パス内にあるため、構成ファイルにリストする必要はありません。2 つの残りのコンテナが新規システムで異なるロケーションに移動された場合は、構成ファイルにリストする必要があります。

物理ディレクトリーおよびファイルが新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと **db2relocatedb** を使用して、データベース・ファイルに変更を行います。

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

例 5

データベース TESTDB には、データベース・パーティション・サーバー 10 および 20 に 2 つのパーティションがあります。このインスタンスは **servinst** で、データベース・パスは両方のデータベース・パーティション・サーバーで **/home/servinst** です。データベースの名前は **SERVDB** に変更され、データベース・パスは両方のデータベース・パーティション・サーバーで **/databases** に変更されます。さらに、ログ・ディレクトリーはデータベース・パーティション・サーバー 20 で、**/testdb_logdir** から **/servdb_logdir** に変更されます。

両方のデータベース・パーティションに変更が行われているため、構成ファイルは各データベース・パーティションに作成され、**db2relocatedb** は対応する構成ファイルを使用する各データベース・パーティション・サーバーで実行される必要があります。

データベース・パーティション・サーバー 10 では、以下の構成ファイルが使用されます。

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

データベース・パーティション・サーバー 20 では、以下の構成ファイルが使用されます。

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

例 6

パス /home/maininst のインスタンス maininst 内に存在するデータベース MAINDB です。4 つの表スペース・コンテナのロケーションを、以下のように変更する必要があります。

```
/maininst_files/allconts/C0 needs to be moved to /MAINDB/C0
/maininst_files/allconts/C1 needs to be moved to /MAINDB/C1
/maininst_files/allconts/C2 needs to be moved to /MAINDB/C2
/maininst_files/allconts/C3 needs to be moved to /MAINDB/C3
```

物理ディレクトリーおよびファイルが、新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと **db2relocatedb** コマンドを使用して、データベース・ファイルに変更を行います。

注: 同様の変更が、すべてのコンテナに対して行われました。すなわち、
/maininst_files/allconts/ が /MAINDB/ で置換され、ワイルドカード文字のある単一項目が使用できるようになります。

```
DB_NAME=MAINDB
DB_PATH=/home/maininst
INSTANCE=maininst
NODE_NUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

関連資料:

- 「コマンド・リファレンス」の『db2inidb - ミラーリングされたデータベースの初期化コマンド』

データ移動での区切り文字の制限

区切り文字についての制約事項:

選択した区切り文字が、移動するデータの一部を成していないことを確かめるのは、ユーザーの責任です。区切り文字がデータの一部になっている場合、想定外のエラーが発生する場合があります。データを移動する際は、以下の制限が列、ストリング、DATALINK、および小数点区切り文字に適用されます。

- 区切り文字は相互に排他的である。
- 区切り文字として バイナリー・ゼロ、改行文字、ブランク・スペースを使用することはできない。
- デフォルトの小数点 (.) をストリング区切り文字として使用することはできない。
- 以下の文字は、ASCII ファミリー・コード・ページと EBCDIC ファミリー・コード・ページでは仕様が異なります。
 - シフトイン (0x0F) とシフトアウト (0x0E) 文字を、EBCDIC MBCS データ・ファイルの区切り文字として使用することはできない。

- MBCS、EUC、または DBCS コード・ページの区切り文字は、0x40 より大きくなければならない (EBCDIC MBCS データのデフォルト小数点 0x4b は例外)。
- 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、二重引用符。ストリング区切り文字)
 - , (0x2c、コンマ; 列区切り文字)
- EBCDIC SBCS コード・ページにおけるデータ・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、二重引用符。ストリング区切り文字)
 - , (0x6B、コンマ。列区切り文字)
- ASCII データ・ファイルのデフォルトの小数点は 0x2e (ピリオド) です。
- EBCDIC データ・ファイルのデフォルトの小数点は 0x4B (ピリオド) です。
- サーバーのコード・ページがクライアントのコード・ページと異なっている場合は、非デフォルトの区切り文字を 16 進表示で指定するようお勧めします。たとえば、以下のように指定します。

```
db2 load from ... modified by charde10x0C colde1X1e ...
```

DEL ファイルでの二重区切り文字の認識サポートに関する以下の情報は、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適用されます。

- 区切り文字を、DEL ファイルの文字ベースのフィールド内で使用することができます。これは、タイプが CHAR、VARCHAR、LONG VARCHAR、または CLOB (lobsinfile が指定されている場合を除く) のフィールドに適用されます。区切り文字で囲まれている区切り文字の対は、データベースにインポートまたはロードされます。たとえば、以下のように指定します。

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

これは、以下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、逆の規則が適用されます。たとえば、以下のように指定します。

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

これは、以下のように DEL ファイルにエクスポートされます。

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

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

型付き表間のデータ移動

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、型付き表からデータを移動したり、型付き表にデータを移動したりできます。型付き表は、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、次のようなものがあります。

- 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、表階層と型階層を作成することができます。

階層内の型の識別方法はデータベースによって違います。つまり、同じ型でもデータベースが異なると ID 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型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それぞれの型の行が書き出される前に、ID が索引値に変換されます。この索引値は、1 から、階層内の関連する型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、それぞれの型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」副表まで達した場合、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。

図 9 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。

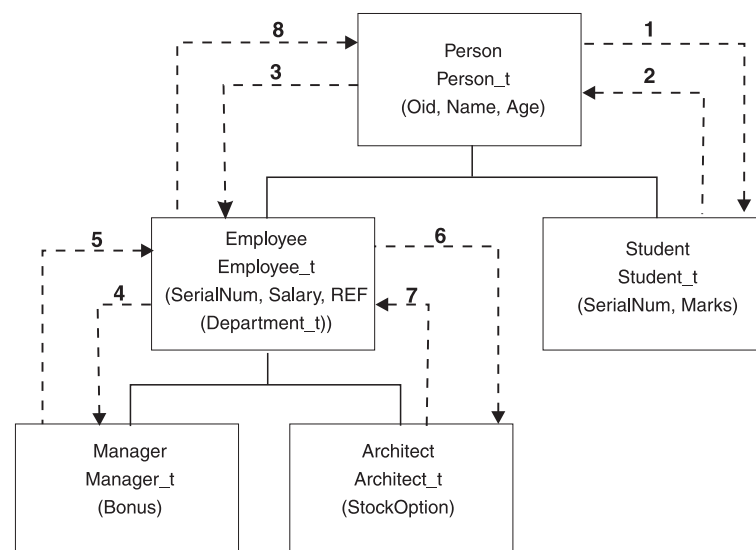


図 14.

関連概念:

- 1 ページの『エクスポートの概要』
- 29 ページの『インポートの概要』

型付き表間のデータ移動 - 詳細

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、関連するすべての型が、階層内の特定の開始点から到達可能な階層内の型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組み込まれており、それぞれの表の順序は OUTER 順序述部で使用さ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、関連のある型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・ユーティリティを起動する場合とインポート・ユーティリティ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 方式で走査する必要があります。つまり、階層内のそれぞれの分岐の最下位にまで達しないと、新しい分岐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・フォーマットが違っていると、デフォルト走査順序の動作が違います。これ以降では、表階層と型の関係が同一であるとします。

PC/IXF ファイル・フォーマットにデータをエクスポートすると、関連のあるすべての型、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・フォーマット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・フォーマットでは、ソースとターゲットの階層の構造が同じであっても、型行および型表の作成順が異なる可能性があります。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、それぞれの型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、それぞれの型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・ユーティリティで必ず同じ走査順が使用されるようにしてください。以下の場合、インポート・ユーティリティは宛先データベースにデータを確実に移動することができます。

- 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。

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、それぞれの分岐の最後まで走査が終わってからでないと、階層内の次の分岐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。

関連資料:

- 278 ページの『区切り付き ASCII (DEL) ファイル・フォーマット』
- 283 ページの『区切りなし ASCII (ASC) ファイル・フォーマット』
- 287 ページの『PC バージョンの IXF ファイル・フォーマット』
- 331 ページの『ワークシート・ファイル・フォーマット (WSF)』

データ移動中の選択

ある階層構造の型付き表から、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** 表名、**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。

関連資料:

- 41 ページの『IMPORT』

型付き表間のデータ移動の例

この項の例は、以下のような階層構造に基づいています。

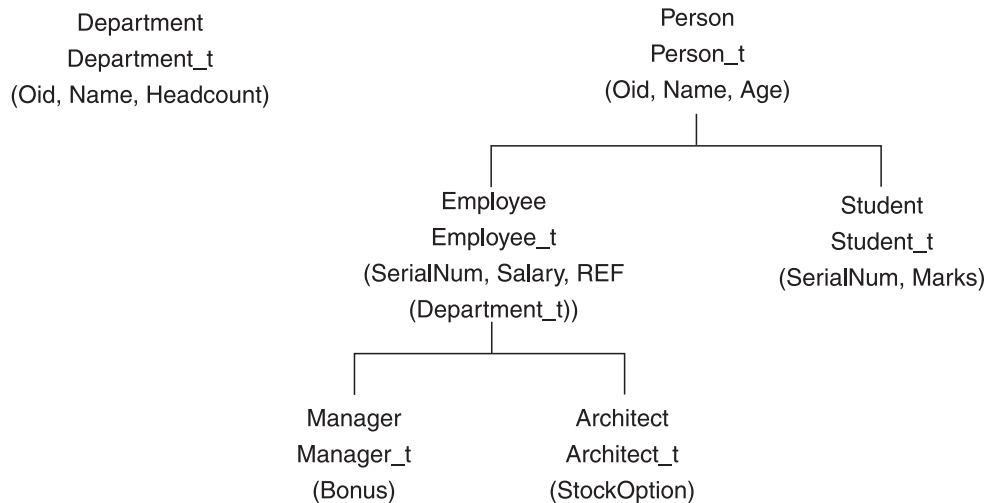


図 15.

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、次に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
  
```

階層内のそれぞれの型が存在しない場合は、それが作成されます。型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がその中に存在してはなりません。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

もっと複雑な例では、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、以下のように、ターゲット・データベースにインポートされ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。つまり、次のようにインポートされます。

- 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

次の例では、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 (型表でない表) 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どの行でも、その最初の列は Object-id であると見なされます。METHOD 文節によって、最後の 2 つの属性の順序が逆転します。

関連概念:

- 250 ページの『型付き表間のデータ移動』

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場合、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™ と DB2 Universal Database™ のツールとし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 Universal Database プラットフォーム間
- DB2 Universal Database 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 (Distributed Relational Database Architecture™) (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® を使用すると、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複製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 つのコントロール・ポイントから企業全体のデータの複製操作を定義、同期化、自動化、および管理できます。DB2 Universal Database 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) と一緒に使用すると、IMS および VSAM データを複製でき、Lotus® NotesPump と一緒に使用すると、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・キャプチャー、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製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、Windows®, OS/390®, SCO UnixWare、Solaris™ Operating Environment、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 製品と一緒に使用すると、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。さらに、Windows CE、または Palm OS 装置上の DB2 Everywhere にデータをレプリケーションすることもできます。

関連概念:

IBM レプリケーション・ツール

IBM レプリケーション・ツールのコンポーネント

IBM® レプリケーション・ツール・ソリューションのコンポーネントには、キャプチャー・プログラムとアプライ・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コンポーネントをセットアップします。これらのコンポーネント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 キャプチャー・プログラムは、ソース表に加えられた変更内容をキャプチャーします。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
- ・ 以前にアプライ・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。アプライ・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用します。

- ・ 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。

関連タスク:

- ・ 「IBM DB2 Information Integrator SQL レプリケーション・ガイドおよびリファレンス」の『SQL レプリケーションの計画』

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 を使用すると、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 を使用すると、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・フォーマット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・ユーティリティとエクスポート・ユーティリティ

これらの DB2 ユーティリティを使用すると、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 ESE 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 Universal Database™ Enterprise Server Edition 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・パーティション化マップが獲得され、各ファイルがデータベース・パーティションにロードできるように入力ファイルがパーティションされ、すべてのパーティションでリモート・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用して、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キャプチャすることもできます。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、しかも指定インターバル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。

ポイント・イン・タイム指定

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。

ステージング表

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 Universal Database ワークステーションのオペレーティング環境、DB2 UDB for OS/390®、DB2 for AS/400®, 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

データウェアハウス・センターを使用すると、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ディメンション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複製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は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。

関連概念:

- 「データウェアハウス・センター 管理ガイド」の『データウェアハウジングが提供するソリューション』

カーソル・ファイル・タイプを使用したデータの移動

LOAD コマンドを使用する際に CURSOR ファイルを指定することにより、中間エクスポート・ファイルを作成しなくても、SQL 照会の結果を直接ターゲット表にロードすることができます。SQL 照会内でニックネームを参照すると、ロード・ユーティリティは、単一のステップで別のデータベースからデータをロードすることもできます。

CLP からカーソル操作からのロードを実行するには、まず SQL 照会に対してカーソルを宣言しなければなりません。これが発行できたら、宣言されるカーソルの名前を *cursorname* に、また CURSOR をファイル・タイプにして、LOAD コマンドを発行できます。

たとえば、

表 ABC.TABLE1 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

表 ABC.TABLE2 には次の 3 つの列があります。

- ONE VARCHAR
- TWO INT
- THREE DATE

以下の CLP コマンドを実行すると、すべてのデータが ABC.TABLE1 から ABC.TABLE2 にロードされます。


```
DECLARE mycurs CURSOR FOR SELECT TWO,ONE,THREE FROM abc.table1  
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

注:

1. 上記の例では、CLP を介して SQL 照会からロードする方法を示します。ただし、*db2LoadStruct* 構造の *piSourceList* および *piFileType* 値を適切に定義することにより、**db2Load** API を介して SQL 照会からロードすることもできます。
2. すでに示したとおり、SQL 照会のソース列タイプはターゲット列タイプと互換性がなければなりません、同一である必要はありません。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『ニックネームとデータ・ソース・オブジェクト』

関連資料:

- 113 ページの『LOAD』
- 「SQL リファレンス 第 1 巻」の『割り当てと比較』

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使用するようになしてください。

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。

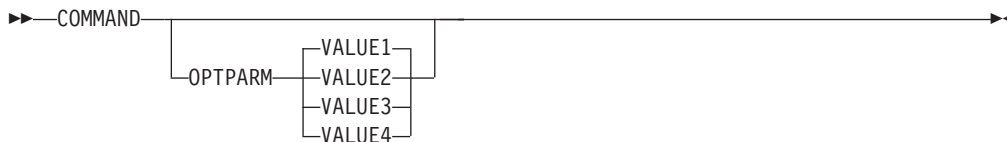
必須パラメーターは、主線と同じ高さに示されます。

▶▶—COMMAND—required parameter—▶▶

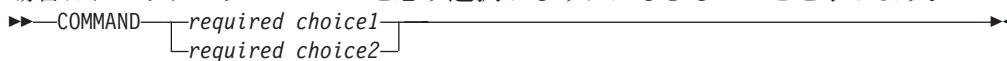
オプション・パラメーターは、主線の下側に示されます。

▶▶—COMMAND—
└optional parameter┘—▶▶

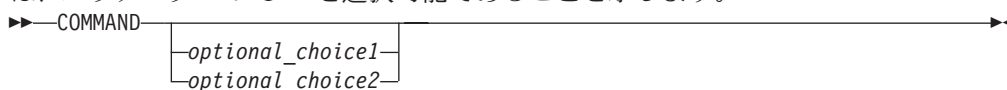
パラメーターのデフォルト値は、線の上側に示されます。



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。

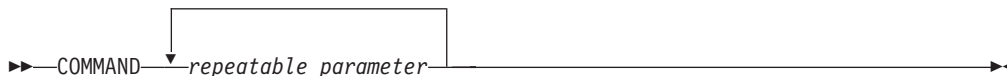


パラメーターのスタックで、最初のパラメーターが主線の下側に示されている場合は、パラメーターの 1 つを選択可能であることを示します。



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。

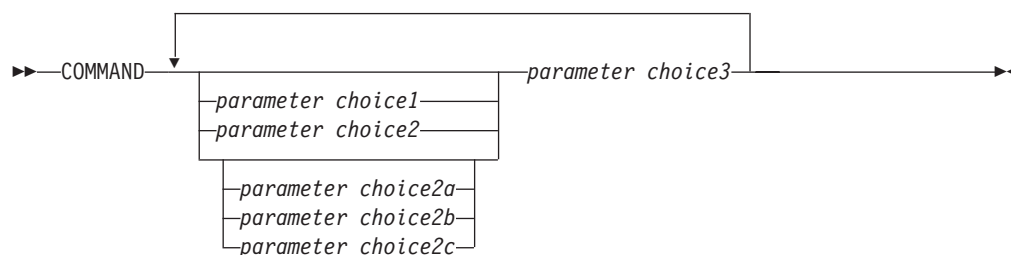


- 矢印にコンマが使用されている場合、項目はコンマによって区切られたリストの形式で繰り返すことができます。

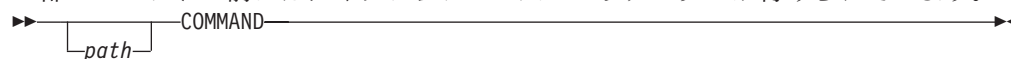


パラメーター・スタックの項目は、前述の必須およびオプション・パラメーターのスタック規則に従って繰り返すことができます。

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・スタックの中に収めら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、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、その組み合わせ (*choice2* と *choice2a*) を繰り返すことができます。

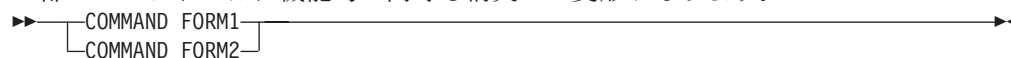


一部のコマンドの前には、オプションのパス・パラメーターが付けられています。



このパラメーターを指定しない場合、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。コマンドが見つからない場合、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、フォーマット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。
パーティション内並列処理の利用が制限される。	パーティション内並列処理を利用する。一般にこれには対称マルチプロセッサ (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。
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 フォーマットでの表、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。
マテリアライズ照会表へのインポートはサポートされない。	マテリアライズ照会表へのロードがサポートされる。
WSF フォーマットがサポートされる。	WSF フォーマット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。	GENERATEDOVERRIDE および IDENTITYOVERRIDE ファイル・タイプ修飾子を使用して、GENERATED ALWAYS 列をオーバーライドすることができる。
表およびビュー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。
すべての行がログに記録される。	最小限のロギングが実行される。
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、 <i>commitcount</i> が指定されていた場合、表は使用可能であり、最後の COMMIT までにロードされた行が収められている。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、 <i>savecount</i> が指定されていると、表はロード・ベンディング状態のままになる。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リストアされるまで、使用できません。

インポート・ユーティリティー	ロード・ユーティリティー
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 TEMPORARY 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・ユーティリティーは、固有性をチェックし、生成される列値を計算するが、他のすべての制約 SET INTEGRITY を使用してチェック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。	パーティション・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。パーティション・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。
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リカバリー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。

関連概念:

- 29 ページの『インポートの概要』
- 84 ページの『ロードの概要』

関連資料:

- 41 ページの『IMPORT』
- 113 ページの『LOAD』

付録 C. エクスポート/インポート/ロード・セッション - API サンプル・プログラム

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。

- データのファイルへのエクスポート
- データの表へのインポート
- データの表へのロード
- ロード操作の状況のチェック

このサンプル・プログラムのソース・ファイル (tbmove.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 bldapp.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入っています。

サンプル・プログラム (実行可能ファイル) を実行するには、tbmove 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。

```

/*****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2002
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/
**
** SOURCE FILE NAME: tbmove.sqc
**
** SAMPLE: How to move table data
**
** DB2 APIs USED:
**      db2Export -- Export
**      db2Import -- Import
**      sqluvqdp -- Quiesce Table Spaces for Table
**      db2Load -- Load
**      db2LoadQuery -- Load Query
**
** SQL STATEMENTS USED:
**      PREPARE
**      DECLARE CURSOR
**      OPEN
**      FETCH
**      CLOSE
**      CREATE TABLE
**      DROP
**
** OUTPUT FILE: tbmove.out (available in the online documentation)
*****/
**
```

```

** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**      http://www.software.ibm.com/data/db2/udb/ad
**/
*****/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"

int DataExport(char *);
int TbImport(char *);
int TbLoad(char *);
int TbLoadQuery(void);

/* support function */
int ExportedDataDisplay(char *);
int NewTableDisplay(void);

EXEC SQL BEGIN DECLARE SECTION;
    char strStmt[256];
    short deptnumb;
    char deptname[15];
EXEC SQL END DECLARE SECTION;

int main(int argc, char *argv[])
{
    int rc = 0;
    char dbAlias[SQL_ALIAS_SZ + 1];
    char user[USERID_SZ + 1];
    char pswd[PSWD_SZ + 1];
    char dataFileName[256];

    /* check the command line arguments */
    rc = CmdLineArgsCheck1(argc, argv, dbAlias, user, pswd);
    if (rc != 0)
    {
        return rc;
    }

    printf("%nTHIS SAMPLE SHOWS HOW TO MOVE TABLE DATA.%n");

    /* connect to database */
    rc = DbConn(dbAlias, user, pswd);
    if (rc != 0)
    {
        return rc;
    }

    #if defined(DB2NT))
        sprintf(dataFileName, "%s%stbmove.DEL", getenv("DB2PATH"), PATH_SEP);
    #else /* UNIX */
        sprintf(dataFileName, "%s%stbmove.DEL", getenv("HOME"), PATH_SEP);
    #endif
}

```

```

rc = DataExport(dataFileName);
rc = TbImport(dataFileName);
rc = TbLoad(dataFileName);
rc = TbLoadQuery();

/* disconnect from the database */
rc = DbDisconn(dbAlias);
if (rc != 0)
{
    return rc;
}

return 0;
} /* main */

int ExportedDataDisplay(char *dataFileName)
{
    struct sqlca sqlca;
    FILE *fp;
    char buffer[100];
    int maxChars = 100;
    int numChars;
    int charNb;

    fp = fopen(dataFileName, "r");
    if (fp == NULL)
    {
        return 1;
    }

    printf("%n The content of the file '%s' is:%n", dataFileName);
    printf(" ");
    numChars = fread(buffer, 1, maxChars, fp);
    while (numChars > 0)
    {
        for (charNb = 0; charNb < numChars; charNb++)
        {
            if (buffer[charNb] == '\n')
            {
                printf("%n");
                if (charNb < numChars - 1)
                {
                    printf(" ");
                }
            }
            else
            {
                printf("%c", buffer[charNb]);
            }
        }
        numChars = fread(buffer, 1, maxChars, fp);
    }

    if (ferror(fp))
    {
        fclose(fp);
        return 1;
    }
    else
    {
        fclose(fp);
    }

    return 0;
} /* ExportedDataDisplay */

int NewTableDisplay(void)

```

```

{
    struct sqlca sqlca;

    printf("%n SELECT * FROM newtable%n");
    printf("    DEPTNUMB DEPTNAME      %n");
    printf("    ----- %n");

    strcpy(strStmt, "SELECT * FROM newtable");

    EXEC SQL PREPARE stmt FROM :strStmt;
    EMB_SQL_CHECK("statement -- prepare");

    EXEC SQL DECLARE c0 CURSOR FOR stmt;

    EXEC SQL OPEN c0;
    EMB_SQL_CHECK("cursor -- open");

    EXEC SQL FETCH c0 INTO :deptnumb, :deptname;
    EMB_SQL_CHECK("cursor -- fetch");

    while (sqlca.sqlcode != 100)
    {
        printf("    %8d %-s%n", deptnumb, deptname);

        EXEC SQL FETCH c0 INTO :deptnumb, :deptname;
        EMB_SQL_CHECK("cursor -- fetch");
    }

    EXEC SQL CLOSE c0;

    return 0;
} /* NewTableDisplay */

int DataExport(char *dataFileName)
{
    int rc = 0;
    struct sqlca sqlca;
    struct sqldcol dataDescriptor;
    char actionString[256];
    struct sqllob *pAction;
    char msgFileName[128];
    struct db2ExportOut outputInfo;
    struct db2ExportStruct exportParmStruct;

    printf("%n-----");
    printf("%nUSE THE DB2 API:%n");
    printf("  db2Export -- Export%n");
    printf("TO EXPORT DATA TO A FILE.%n");

    printf("%n Be sure to complete all table operations and release%n");
    printf("  all locks before starting an export operation. This%n");
    printf("  can be done by issuing a COMMIT after closing all%n");
    printf("  cursors opened WITH HOLD, or by issuing a ROLLBACK.%n");
    printf("  Please refer to the 'Administrative API Reference'%n");
    printf("  for the details.%n");

    /* export data */
    dataDescriptor.dcolmeth = SQL_METH_D;
    strcpy(actionString, "SELECT deptnumb, deptname FROM org");
    pAction = (struct sqllob *)malloc(sizeof(sqluint32) +
                                     sizeof(actionString) + 1);
    pAction->length = strlen(actionString);
    strcpy(pAction->data, actionString);
    strcpy(msgFileName, "tbexport.MSG");

    exportParmStruct.piDataFileName = dataFileName;
    exportParmStruct.piLobPathList = NULL;

```

```

exportParmStruct.piLobFileList      = NULL;
exportParmStruct.piDataDescriptor  = &dataDescriptor;
exportParmStruct.piActionString     = pAction;
exportParmStruct.piFileType        = SQL_DEL;
exportParmStruct.piFileTypeMod     = NULL;
exportParmStruct.piMsgFileName     = msgFileName;
exportParmStruct.iCallerAction     = SQLU_INITIAL;
exportParmStruct.poExportInfoOut   = &outputInfo;

printf("\n  Export data.\n");
printf("    client destination file name: %s\n", dataFileName);
printf("    action                      : %s\n", actionString);
printf("    client message file name    : %s\n", msgFileName);

/* export data */
db2Export(db2Version820,
          &exportParmStruct,
          &sqlca);

DB2_API_CHECK("data -- export");

/* free memory allocated */
free(pAction);

/* display exported data */
rc = ExportedDataDisplay(dataFileName);

return 0;
} /* DataExport */

int TbImport(char *dataFileName)
{
    int rc = 0;
    struct sqlca sqlca;
    struct sqlldcol dataDescriptor;
    char actionString[256];
    struct sqlchar *pAction;
    char msgFileName[128];
    struct db2ImportIn inputInfo;
    struct db2ImportOut outputInfo;
    struct db2ImportStruct importParmStruct;
    long commitcount = 10;

    printf("\n-----");
    printf("\nUSE THE DB2 API:\n");
    printf("  db2Import -- Import\n");
    printf("TO IMPORT DATA TO A TABLE.\n");

    /* create new table */
    printf("\n  CREATE TABLE newtable(deptnumb SMALLINT NOT NULL,");
    printf("\n                                deptname VARCHAR(14))\n");

    EXEC SQL CREATE TABLE newtable(deptnumb SMALLINT NOT NULL,
                                    deptname VARCHAR(14));
    EMB_SQL_CHECK("new table -- create");

    /* import table */
    dataDescriptor.dcolmeth = SQL_METH_D;
    strcpy(actionString, "INSERT INTO newtable");
    pAction = (struct sqlchar *)malloc(sizeof(short) +
                                      sizeof(actionString) + 1);
    pAction->length = strlen(actionString);
    strcpy(pAction->data, actionString);
    strcpy(msgFileName, "tbimport.MSG");

    /* Setup db2ImportIn structure */
    inputInfo.iRowcount = inputInfo.iRestartcount = 0;

```



```

inputInfo.iSkipcount = inputInfo.iWarningcount = 0;
inputInfo.iNoTimeout = 0;
inputInfo.iAccessLevel = SQLU_ALLOW_NO_ACCESS;
inputInfo.piCommitcount = &commitcount;

printf("\n Import table.\n");
printf(" client source file name : %s\n", dataFileName);
printf(" action : %s\n", actionString);
printf(" client message file name: %s\n", msgFileName);

ImportparmStruct.piDataFileName = dataFileName;
importParmStruct.piLobPathList = NULL;
importParmStruct.piDataDescriptor = &dataDescriptor;
importParmStruct.piActionString = pAction;
importParmStruct.piFileType = SQL_DEL;
importParmStruct.piFileTypeMod = NULL;
importParmStruct.piMsgFileName = msgFileName;
importParmStruct.piImportInfoIn = &inputInfo;
importParmStruct.piImportInfoOut = &outputInfo;
importParmStruct.piNullIndicators = NULL;
importParmStruct.iCallerAction = SQLU_INITIAL;

/* import table */
db2Import(db2Version820,
          &importParmStruct,
          &sqlca);

DB2_API_CHECK("table -- import");

/* free memory allocated */
free(pAction);

/* display import info */
printf("\n Import info.\n");
printf(" rows read : %ld\n", (int)outputInfo.oRowsRead);
printf(" rows skipped : %ld\n", (int)outputInfo.oRowsSkipped);
printf(" rows inserted : %ld\n", (int)outputInfo.oRowsInserted);
printf(" rows updated : %ld\n", (int)outputInfo.oRowsUpdated);
printf(" rows rejected : %ld\n", (int)outputInfo.oRowsRejected);
printf(" rows committed: %ld\n", (int)outputInfo.oRowsCommitted);

/* display content of the new table */
rc = NewTableDisplay();

/* drop new table */
printf("\n DROP TABLE newtable\n");

EXEC SQL DROP TABLE newtable;
EMB_SQL_CHECK("new table -- drop");

return 0;
} /* TbImport */

int TbLoad(char *dataFileName)
{
    int rc = 0;
    struct sqlca sqlca;

    struct db2LoadStruct paramStruct;
    struct db2LoadIn inputInfoStruct;
    struct db2LoadOut outputInfoStruct;

    struct sqlu_media_list mediaList;
    struct sqlldcol dataDescriptor;
    char actionString[256];
    struct sqlchar *pAction;
    char localMsgFileName[128];

```

```

printf("%n-----");
printf("%nUSE THE DB2 API:%n");
printf("  sqluvqdp -- Quiesce Table Spaces for Table%n");
printf("  db2Load -- Load%n");
printf("TO LOAD DATA INTO A TABLE.%n");

/* create new table */
printf("%n CREATE TABLE newtable(deptnumb SMALLINT NOT NULL,");
printf("%n                                deptname VARCHAR(14))%n");

EXEC SQL CREATE TABLE newtable(deptnumb SMALLINT NOT NULL,
                                deptname VARCHAR(14));
EMB_SQL_CHECK("new table -- create");

/* quiesce table spaces for table */
printf("%n Quiesce the table spaces for 'newtable'.%n");

EXEC SQL COMMIT;
EMB_SQL_CHECK("transaction -- commit");

/* quiesce table spaces for table */
sqluvqdp("newtable", SQLU_QUIESCEMODE_RESET_OWNED, NULL, &sqlca);
DB2_API_CHECK("tablespaces for table -- quiesce");

/* load table */
mediaList.media_type = SQLU_CLIENT_LOCATION;
mediaList.sessions = 1;
mediaList.target.location =
    (struct sqlu_location_entry *)malloc(sizeof(struct sqlu_location_entry) *
                                         mediaList.sessions);
strcpy(mediaList.target.location->location_entry, dataFileName);

dataDescriptor.dcolmeth = SQL_METH_D;

strcpy(actionString, "INSERT INTO newtable");
pAction = (struct sqlchar *)malloc(sizeof(short) +
                                   sizeof(actionString) + 1);
pAction->length = strlen(actionString);
strcpy(pAction->data, actionString);

strcpy(localMsgFileName, "tblload.MSG");

/* Setup the input information structure */
inputInfoStruct.piUseTablespace = NULL;
inputInfoStruct.iSavecount = 0; /* consistency points */
/* as infrequently as possible */
/* start at row 1 */
inputInfoStruct.iRestartcount = 0; /* load all rows */
inputInfoStruct.iRowcount = 0;
inputInfoStruct.iWarningcount = 0; /* don't stop for warnings */
inputInfoStruct.iDataBufferSize = 0; /* default data buffer size */
inputInfoStruct.iSortBufferSize = 0; /* def. warning buffer size */
inputInfoStruct.iHoldQuiesce = 0; /* don't hold the quiesce */
inputInfoStruct.iRestartphase = ' '; /* ignored anyway */
inputInfoStruct.iStatsOpt = SQLU_STATS_NONE; /* don't bother with them */
inputInfoStruct.iIndexingMode = SQLU_INX_AUTOSELECT; /* let load choose */
/* indexing mode */

inputInfoStruct.iCpuParallelism = 0;
inputInfoStruct.iNonrecoverable = SQLU_NON_RECOVERABLE_LOAD;
inputInfoStruct.iAccessLevel = SQLU_ALLOW_NO_ACCESS;
inputInfoStruct.iLockWithForce = SQLU_NO_FORCE;
inputInfoStruct.iCheckPending = SQLU_CHECK_PENDING_CASCADE_DEFERRED;

/* Setup the parameter structure */
paramStruct.piSourceList = &mediaList;
paramStruct.piLobPathList = NULL;
paramStruct.piDataDescriptor = &dataDescriptor;

```

```

paramStruct.piActionString = pAction;
paramStruct.piFileType = SQL_DEL;
paramStruct.piFileTypeMod = NULL;
paramStruct.piLocalMsgFileName = localMsgFileName;
paramStruct.piTempFilesPath = NULL;
paramStruct.piVendorSortWorkPaths = NULL;
paramStruct.piCopyTargetList = NULL;
paramStruct.piNullIndicators = NULL;
paramStruct.piLoadInfoIn = &inputInfoStruct;
paramStruct.poLoadInfoOut = &outputInfoStruct;
paramStruct.piPartLoadInfoIn = NULL;
paramStruct.poPartLoadInfoOut = NULL;
paramStruct.iCallerAction = SQLU_INITIAL;

printf("\n Load table.\n");
printf(" client source file name : %s\n", dataFileName);
printf(" action : %s\n", actionString);
printf(" client message file name: %s\n", localMsgFileName);

/* load table */
db2Load (db2Version810, /* Database version number */
        &paramStruct, /* In/out parameters */
        &sqlca); /* SQLCA */

DB2_API_CHECK("table -- load");

/* free memory allocated */
free(pAction);

/* display load info */
printf("\n Load info.\n");
printf(" rows read : %d\n", (int)outputInfoStruct.oRowsRead);
printf(" rows skipped : %d\n", (int)outputInfoStruct.oRowsSkipped);
printf(" rows loaded : %d\n", (int)outputInfoStruct.oRowsLoaded);
printf(" rows deleted : %d\n", (int)outputInfoStruct.oRowsDeleted);
printf(" rows rejected : %d\n", (int)outputInfoStruct.oRowsRejected);
printf(" rows committed: %d\n", (int)outputInfoStruct.oRowsCommitted);

/* display content of the new table */
rc = NewTableDisplay();

/* drop new table */
printf("\n DROP TABLE newtable\n");

EXEC SQL DROP TABLE newtable;
EMB_SQL_CHECK("new table -- drop");

return 0;
} /* TbLoad */

int TbLoadQuery(void)
{
    int rc = 0;
    struct sqlca sqlca;
    char tableName[128];
    char loadMsgFileName[128];
    db2LoadQueryStruct loadQueryParameters;
    db2LoadQueryOutputStruct loadQueryOutputStructure;

    printf("\n-----");
    printf("\nUSE THE DB2 API:\n");
    printf(" db2LoadQuery -- Load Query\n");
    printf("TO CHECK THE STATUS OF A LOAD OPERATION.\n");

    /* Initialize structures */
    memset(&loadQueryParameters, 0, sizeof(db2LoadQueryStruct));
    memset(&loadQueryOutputStructure, 0, sizeof(db2LoadQueryOutputStruct));

```

```

/* Set up the tablename to query. */
loadQueryParameters.iStringType = DB2LOADQUERY_TABLENAME;
loadQueryParameters.piString = tableName;

/* Specify that we want all LOAD messages to be reported. */
loadQueryParameters.iShowLoadMessages = DB2LOADQUERY_SHOW_ALL_MSGS;

/* LOAD summary information goes here. */
loadQueryParameters.poOutputStruct = &loadQueryOutputStructure;

/* Set up the local message file. */
loadQueryParameters.piLocalMessageFile = loadMsgFileName;

/* call the DB2 API */
strcpy(tableName, "ORG");
strcpy(loadMsgFileName, "tblqry.MSG");

/* load query */
db2LoadQuery(db2Version810, &loadQueryParameters, &sqlca);
printf("\n Note: the table load for '%s' is NOT in progress.\n", tableName);
printf(" So an empty message file '%s' will be created,\n", loadMsgFileName);
printf(" and the following values will be zero.\n");
    DB2_API_CHECK("status of load operation -- check");

printf("\n Load status has been written to local file %s.\n",
        loadMsgFileName);

printf("    Number of rows read          = %d\n",
        loadQueryOutputStructure.oRowsRead);

printf("    Number of rows skipped        = %d\n",
        loadQueryOutputStructure.oRowsSkipped);

printf("    Number of rows loaded          = %d\n",
        loadQueryOutputStructure.oRowsLoaded);

printf("    Number of rows rejected        = %d\n",
        loadQueryOutputStructure.oRowsRejected);

printf("    Number of rows deleted          = %d\n",
        loadQueryOutputStructure.oRowsDeleted);

printf("    Number of rows committed        = %d\n",
        loadQueryOutputStructure.oRowsCommitted);

printf("    Number of warnings              = %d\n",
        loadQueryOutputStructure.oWarningCount);

return 0;
} /* TbLoadQuery */

```

付録 D. ファイル・フォーマット

エクスポート/インポート/ロード・ユーティリティのファイル・フォーマット

ここでは、DB2® のエクスポート、インポート、およびロード・ユーティリティによってサポートされている 5 種類のオペレーティング・システム・ファイル・フォーマット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もつフラット・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、内部表の外部表記の入った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・フォーマット。Lotus 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・ユーティリティでは、このファイル・フォーマットはサポートされません。

CURSOR

SQL 照会に対して宣言されるカーソル。このファイル・タイプは、ロード・ユーティリティによってのみサポートされます。

DEL、WSF、または ASC データ・ファイル・フォーマットを使用する場合は、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、および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などがあります。

PC/IXF データ・ファイル・フォーマットを使用する場合、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) は新しい表の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・フォーマット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。

CURSOR ファイル・タイプを使用する際には、ロード操作を開始する前に、列名およびデータ・タイプなどで表を定義する必要があります。SQL 照会の列タイプは、ターゲット表の対応する列タイプと互換性がなければなりません。ロード操作

を開始する前に、指定されたカーソルをオープンする必要はありません。ロード・ユーティリティーは、カーソルが行の取り出しに使用されていたかどうかに関係なく、指定されたカーソルに関連する照会の結果全体を処理します。

関連概念:

- 「SQL リファレンス 第 1 巻」の『照会と表式』

関連資料:

- 278 ページの『区切り付き ASCII (DEL) ファイル・フォーマット』
- 283 ページの『区切りなし ASCII (ASC) ファイル・フォーマット』
- 287 ページの『PC バージョンの IXF ファイル・フォーマット』
- 「SQL リファレンス 第 1 巻」の『データ・タイプ間のキャスト』
- 「SQL リファレンス 第 1 巻」の『割り当てと比較』

関連サンプル:

- 『dtformat.out -- HOW TO LOAD AND IMPORT DATA FORMAT EXTENSIONS (C)』
- 『dtformat.sqc -- Load and import data format extensions (C)』

区切り付き ASCII (DEL) ファイル・フォーマット

区切り付き ASCII (DEL) ファイルは、行および列の区切り文字を使った順次 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 ファイルのフォーマットを示します。

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はない場合、
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

End-of-file character ::= Hex '1A' (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 から 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 から 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) であると見なされます。
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. 区切り文字
3. 列区切り文字

- ^d 数値の ASCII 表記に指数が使用されている場合、それは FLOAT 定数です。小
数点を使っているが指数は使っていない場合、それは DECIMAL 定数です。小数
点も指数も使用されていない場合、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。

関連資料:

- 281 ページの『DEL のデータ・タイプの説明』

例およびデータ・タイプの説明

DEL ファイル例

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 (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を示しています。文字データ中にコンマが使われているため、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競合するため、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 (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から 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はありません。
6. Windows オペレーティング・システムの場合、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. NULL 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 16. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット

データ・タイプ	エクスポート・ユーティリティによって作成されるファイルの形式	インポート・ユーティリティにとって受け入れ可能な形式
BIGINT	-9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲の INTEGER 定数。	-9,223,372,036,854,775,808 から 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。
BLOB、CLOB	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	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、必要な場合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	区切り文字なしの <i>yyyymmdd</i> (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 フォーマット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・データベースの地域別コードに一致する ISO フォーマットの日付値の入った区切り付きまたは区切りなし文字ストリング、あるいは <i>yyyymmdd</i> の形式の区切りなし文字ストリング。
DBCLOB (DBCS のみ)	GRAPHIC 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 16. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	エクスポート・ユーティリティ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。入力値の小数部の列数が、データベース列で受け入れ可能なものより多い場合、余分な列は切り捨てられます。
FLOAT(long)	-10E307 から 10E307 の範囲の FLOAT 定数。	-10E307 から 10E307 の範囲の数値の ASCII 表記。
GRAPHIC (DBCS のみ)	GRAPHIC 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、必要な場合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 から 2,147,483,647 の範囲の INTEGER 定数。	-2,147,483,648 から 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	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)	GRAPHIC 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。
SMALLINT	-32,768 から 32,767 の範囲の INTEGER 定数。	-32,768 から 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)。区切り文字によって囲まれた ISO フォーマットの時刻値。たとえば “09.39.43”	ターゲット・データベースの地域別コードに合致するフォーマットの時刻値を使った区切り付きまたは区切りなし文字ストリング。

表 16. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	エクスポート・ユーティリティによって作成されるファイルの形式	インポート・ユーティリティにとって受け入れ可能な形式
TIMESTAMP	yyyy-mm-dd-hh.mm.ss.nnnnnn (年、月、日、時、分、秒、マイクロ秒)。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・スタンプ値を使った区切り付きまたは区切りなし文字ストリング。
VARCHAR	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、必要な場合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)	GRAPHIC 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、必要な場合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・フォーマット

区切りなし ASCII (ASC) ファイルは、行区切り文字を使った順次 ASCII ファイルです。このファイルは、ワード・プロセッサを含め、縦欄フォーマット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/終了ロケーションの対 (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・ロケーションとして指定された行内のロケーションを表します。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エレメントは列の開始バイト、2 番目のエレメント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) であると見なされます。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。

関連資料:

- 285 ページの『ASC のデータ・タイプの説明』

例およびデータ・タイプの説明

ASC ファイル例

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 (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、列名が入っていないものと見なされます。
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、NULL 可能データベース列に NULL 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/終了ロケーション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/終了ロケーション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入っていない場合
 - NULL INDICATORS ロード・オプションが使用されていて、NULL 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. NULL 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、その行はリジェクトされます。
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 NULL 可能でない場合は、エラーが検出された場所に応じて NULL がインポートされるか、または行がリジェクトされます。列が NULL 可能でない場合は、行がリジェクトされます。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 17. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット

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	<p>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 から 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リジェクトされます。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見なされます。 3 桁ごとの区切り文字は使用できません。</p> <p>開始ロケーションと終了ロケーションは、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。</p>
BLOB/CLOB	<p>文字のストリング。文字ストリングは、必要な場合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。</p>
BLOB_FILE、 CLOB_FILE、 DBCLOB_FILE (DBCS のみ)	<p>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。</p>
CHAR	<p>文字のストリング。文字ストリングは、必要な場合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。</p>
DATE	<p>ターゲット・データベースの地域別コードに合致するフォーマットの日付値を表す文字ストリング。</p> <p>開始ロケーションと終了ロケーションは、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。</p>
DBCLOB (DBCS のみ)	<p>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、必要な場合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。</p>
DECIMAL	<p>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リジェクトされます。入力値の小数部の列数が、データベース列の位取りより多い場合、余分な列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見なされます。 3 桁ごとの区切り文字は使用できません。</p> <p>開始ロケーションと終了ロケーションは、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。</p>

表 17. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (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
FLOAT(long)	<p>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見なされます。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。</p> <p>開始ロケーションと終了ロケーションは、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。</p>
GRAPHIC (DBCS のみ)	<p>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、必要な場合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。</p>
INTEGER	<p>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 から 2,147,483,647 の範囲外の整数があれば、それに関してはリジェクトされます。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見なされます。3 桁ごとの区切り文字は使用できません。</p> <p>開始ロケーションと終了ロケーションは、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。</p>
LONG VARCHAR	<p>文字のストリング。文字ストリングは、必要な場合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。</p>
LONG VARGRAPHIC (DBCS のみ)	<p>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、必要な場合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。</p>
SMALLINT	<p>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 から 32,767 の範囲外の値があれば、その特定の値に関してはリジェクトされます。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見なされます。3 桁ごとの区切り文字は使用できません。</p> <p>開始ロケーションと終了ロケーションは、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。</p>
TIME	<p>ターゲット・データベースの地域別コードに合致するフォーマットの時間値を表す文字ストリング。</p> <p>開始ロケーションと終了ロケーションは、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。</p>

表 17. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (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
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。 開始ロケーションと終了ロケーションは、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。
VARCHAR	文字のストリング。文字ストリングは、必要な場合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、必要な場合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・フォーマット

PC バージョンの IXF (PC/IXF) ファイル・フォーマットは、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合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に必要なことも、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・データベース交換フォーマットです。PC/IXF ファイル・フォーマット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。すべてのインプリメンテーションですべての 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、制限付きのインプリメンテーションでも、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 フォーマット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 ファイルに組み込む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 ID によって暗黙指定されるデータ・フォーマット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ない、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も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には、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 ファイルに LOB ロケーション指定子 (LLS) 列が入っている場合には、それぞれの LLS 列ごとに D レコードがなければなりません。D レコードはエクスポート・ユーティリティによって自動的に作成されますが、PC/IXF ファイルを生成するためにサード・パーティーのツールを使用している場合には、これらを手動で作成する必要があります。さらに、NULL 値のある列を含め、表内の各 LOB 列ごとに LLS が必要です。LOB 列が NULL である場合には、NULL LOB を表す LLS を作成する必要があります。

PC/IXF ファイルのレコードは、文字データの入った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 フィールド内に入れら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 フィールド内に入れら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 レコードに入ってい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 PC/IXF ファイル・フォーマットは、System/370 と同じではありません。

関連資料:

- 289 ページの『PC/IXF レコード・タイプ』
- 307 ページの『PC/IXF データ・タイプ』
- 318 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』
- 320 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データ・タイプ固有の規則』
- 323 ページの『FORCEIN オプション』
- 330 ページの『PC/IXF およびバージョン 0 の System/370 IXF の相違』
- 313 ページの『PC/IXF のデータ・タイプの説明』

PC バージョンの IXF ファイル・フォーマット - 詳細

PC/IXF 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
- ID

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 ID
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
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
IXHFHIL1	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入っています。

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・フォーマット ID (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 フォーマット・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、このカウントに入りません。

IXFHSBCP

SBCS CPGID または '00000' の 1 バイト文字表記の入った 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 バイト文字表記の入った 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・フォーマット = 'M'
IXFTMFRM	005-BYTE	CHARACTER	マシン形式 = '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	主キー名
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入っています。

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 つの表だけがある場合、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) フォーマット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・データのフォーマット。データベース・マネージャーは、このフィールドが PCbbb 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・フォーマットになっていることを指定します。

IXFTLOC

データのロケーション。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 NULL 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主キーの名前 (ある場合)。この名前は NULL 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 NULL 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	主キーでの位置
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 ID
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	ディメンション数
IXFCDSIZ	varying	CHARACTER	各ディメンションのサイズ

列記述子レコードには、以下のフィールドが入っています。

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 NULL 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

主キーの一部としての列の位置。有効値は 01 から 16 ですが、主キーの一部でない列の場合は N です。

IXFCCLAS

IXFCTYPE フィールドで使用するデータ・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。

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。

このフィールドのセマンティクスは、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ゼロ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。この値がゼロの場合、列はビット・ストリング・データが入ると解釈されます。
- 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・ユーティリティでは 0 にセットされ、インポート・ユーティリティでは無視されます。
- 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・ユーティリティでは無視されます。
- GRAPHIC 列の場合、このフィールドは 0 でなければなりません。

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。

このフィールドのセマンティクスは、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・ユーティリティでは 0 にセットされ、インポート・ユーティリティでは無視されます。
- 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・ユーティリティでは 0 にセットされ、インポート・ユーティリティでは無視されます。
- GRAPHIC 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。

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 ID。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 NULL 可能なら IXFCPOSN は NULL 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ある場合、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はありません。

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・タイプとして使用さ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 番目の文字はリカバリー・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ポイント・イン・タイム指定リカバリーをサポートする) 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 (リストア) 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

列のディメンションの数。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

各ディメンションのサイズまたは範囲。このフィールドの長さは、1 ディメンション当たり 5 バイトです。配列はサポートされない (ディメンションの数は 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 ID
IXDFIL1	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入っています。

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入っていることを示します。

IXFDRID

レコード ID。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 ID が実際に存在していなければなりません。

IXDFIL1

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 ID 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。

列項目データのフォーマットは、列が NULL 可能であるかどうかによって異なります。

- 列が NULL 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 NULL 標識が組み込まれます。列が NULL 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。 NULL 標識は、NULL でない場合は x'0000' にセットされ、NULL の場合は x'FFFF' にセットされる 2 バイトの値です。

- 列が NULL 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、現行長標識が組み込まれます。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 バイトを超えてはなりません。

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 ID
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入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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 文字を、データベース・マネージャーを識別する定数に設定して、最後の 6 文字を、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する定数に設定することができます。

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 ID = '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	ユニークな規則
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入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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 (主キー)、U (ユニーク索引)、および D (非ユニーク索引) 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 から 99 です。-1 またはゼロの値を指定すると、システム・デフォルト値が使用されます。

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 から 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 ID = '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 つ使用されます。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ない、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入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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の副表レコードの予想数を指定します。

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 ID = '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	現行の表の開始列索引
IXFAENDX	005-BYTE	CHARACTER	現行の表の終了列索引

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 つ使用されます。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入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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 ID = '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	最後のディスク・ボリューム番号
IXFATHIS	002-BYTE	SHORT INT	このディスク・ボリューム番号
IXFANEXT	002-BYTE	SHORT INT	次のディスク・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、マルチボリュームの 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、マルチボリュームの 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入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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・フォーマットの 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・フォーマットの 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・フォーマットの 2 進数フィールドです。レコードがファイルの先頭でない場合、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 ID = '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スです。DB2 終了レコードには、以下のフィールドが入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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 ID 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 ID
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	ID 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	ID 列の START AT 値
IXFAINCR	33-BYTE	CHARACTER	ID 列の INCREMENT BY 値
IXFACACH	10-BYTE	CHARACTER	ID 列の CACHE 値
IXFAMINV	33-BYTE	CHARACTER	identity MINVALUE
IXFAMAXV	33-BYTE	CHARACTER	identity MAXVALUE
IXFACYCL	01-BYTE	CHARACTER	identity CYCLE ('Y' or 'N')
IXFAORDR	01-BYTE	CHARACTER	identity ORDER ('Y' or 'N')
IXFARMRL	03-BYTE	CHARACTER	identity Remark length
IXFARMRK	254-BYTE	CHARACTER	identity Remark value

DB2 識別レコードには、以下のフィールドが入っています。

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 ID によって暗黙指定されるデータの内容とフォーマット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 ID。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 ID 列の列番号。

IXFAITYP

ID 列のタイプ。"Y" の値は、ID 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 ID 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 ID 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 ID 列の CACHE 値。"1" の値は、NO CACHE オプションに対応します。

IXFAMINV

表の作成時に CREATE TABLE ステートメントで指定された ID 列の MINVALUE。

IXFAMAXV

表の作成時に CREATE TABLE ステートメントで指定された ID 列の MAXVALUE。

IXFACYCL

表の作成時に CREATE TABLE ステートメントで指定された ID 列の CYCLE 値。値 "Y" は CYCLE オプションに対応し、それ以外の値は NO CYCLE に対応します。

IXFAORDR

表の作成時に CREATE TABLE ステートメントで指定された ID 列の ORDER 値。値 "Y" は ORDER オプションに対応し、それ以外の値は NO ORDER に対応します。

IXFARMRL

IXFARMRK フィールド内の注釈の長さ (バイト単位)。

IXFARMRK

これは ID 列に関連付けられたユーザー入力の注釈です。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。

関連資料:

- 307 ページの『PC/IXF データ・タイプ』
- 313 ページの『PC/IXF のデータ・タイプの説明』

PC/IXF データ・タイプ

表 18. 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 から 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	<p>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。</p> <p>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。</p> <p>次の記述は CLOB にのみ適用されます。IXFCDBCP がゼロ以外の場合、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 バイト文字も組み込むことができます。</p>
BLOB_LOCATION_SPECIFIER、 BLOB_LOCATION_SPECIFIER、 および DBCLOB_LOCATION_SPECIFIER	960、964、968	<p>固定長フィールド。255 バイトを超えてはなりません。LOB ロケーション指定子 (LLS) は、IXFCSBCP によって指定されるコード・ページにあります。IXFCSBCP が 0 の場合、LLS 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。IXFCDBCP がゼロ以外の場合、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 バイト文字も組み込むことができます。</p> <p>LLS 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 LLS の長さで作成されるため、このタイプの列の入った PC/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。</p>

表 18. 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	<p><i>name_length</i> および <i>name</i> フィールドにデータが入れられた SQLFILE 構造体を収める固定長フィールド。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 がゼロ以外の場合、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 バイト文字も組み込むことができます。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。</p> <p>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 <i>sql_lobfile_len</i> の長さで作成されるため、タイプ BLOB_FILE、CLOB_FILE、または DBCLOB_FILE の列をもつ 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。</p>
CHAR	452	<p>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 がゼロ以外の場合、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 バイト文字も組み込むことができます。IXFCSBCP がゼロの場合、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。</p>
DATE	384	<p>グレゴリオ暦に準拠したポイント・イン・タイム。それぞれの日付は、国際標準化機構規格 (ISO) フォーマット (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 から 9999 です。月の部分の範囲は 01 から 12 です。日の部分の範囲は 01 から <i>n</i> です。ここで、<i>n</i> は月によって異なり、月の日数と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。</p> <p>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。</p>

表 18. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
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。
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 から 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フォーマットになっています。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・フォーマットです。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・マネージャではサポートされません。

表 18. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
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。
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 から +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 がゼロ以外の場合、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 バイト文字も組み込むことができます。IXFCSBCP がゼロの場合、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 18. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
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。
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 から +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 時間制によるポイント・イン・タイム。それぞれの時刻は、ISO フォーマット (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 から 24 で、その他の部分の範囲は 00 から 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 18. 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 がゼロ以外の場合、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 バイト文字も組み込むことができます。IXFCSBCP がゼロの場合、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 GRAPHIC 列では、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 GRAPHIC 列は、無効なデータ・タイプです。

表 19. 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。		

関連資料:

- 289 ページの『PC/IXF レコード・タイプ』
- 323 ページの『FORCEIN オプション』
- 313 ページの『PC/IXF のデータ・タイプの説明』

PC/IXF のデータ・タイプの説明

表 20. PC/IXF ファイル・フォーマットで受け入れられるデータ・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 から 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リジェクトされます。
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、BLOB_FILE、または BLOB_LOCATION_SPECIFIER 列が受け入れ可能です。 <ul style="list-style-type: none"> • 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。

表 20. PC/IXF ファイル・フォーマットで受け入れられるデータ・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。	<p>次の場合、PC/IXF CHAR、VARCHAR、または LONG VARCHAR 列が受け入れ可能です。</p> <ul style="list-style-type: none"> 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。 <p>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 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、必要な場合右側に 1 バイト・スペース (x'20') が埋められます。</p>
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 列の単一バイト・コード・ページの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・ページの値がゼロか、データベース列の DBCS CPGID と等しい場合には、PC/IXF CHAR、VARCHAR、LONG VARCHAR、CLOB、CLOB_FILE、または CLOB_LOCATION_SPECIFIER 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。
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・データベースの地域コードと互換性のあるフォーマットの日付でなければなりません。

表 20. PC/IXF ファイル・フォーマットで受け入れられるデータ・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・ユーティリティによって受け入れ可能な形式
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等しい場合は、PC/IXF GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、DBCLOB、DBCLOB_FILE、または DBCLOB_LOCATION_SPECIFIER 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。
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リジェクトされます。
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。
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は互換性がなければなりません。データは、必要な場合右側に 2 バイト・スペース (x'8140') が埋められます。
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 から 2,147,483,647 の範囲外の整数があれば、それに関してはリジェクトされます。

表 20. PC/IXF ファイル・フォーマットで受け入れられるデータ・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・ユーティリティにとって受け入れ可能な形式
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。	<p>次の場合、PC/IXF CHAR、VARCHAR、または LONG VARCHAR 列が受け入れ可能です。</p> <ul style="list-style-type: none"> 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。 <p>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 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。</p>
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。
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 から 32,767 の範囲外の値があれば、その特定の値に関してはリジェクトされます。
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・データベースの地域コードと互換性のあるフォーマットの時刻データでなければなりません。

表 20. PC/IXF ファイル・フォーマットで受け入れられるデータ・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・ユーティリティにとって受け入れ可能な形式
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ています。PC/IXF ファイルの文字タイプの列に入っているデータは、タイム・スタンプの入力フォーマットになっていなければなりません。
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。 <ul style="list-style-type: none"> 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場合、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。
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。

関連資料:

- 289 ページの『PC/IXF レコード・タイプ』
- 307 ページの『PC/IXF データ・タイプ』

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・ユーティリティは、PC/IXF フォーマットのファイル (IXFHID = 'IXF') のみ受け入れます。その他のフォーマットの IXF ファイルはインポートできません。
- インポート・ユーティリティは、1024 個より多くの列が入っている PC/IXF ファイルをリジェクトします。
- 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 との間の遷移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・ユーティリティは、FORCEIN オプションが指定されない限り、PC/IXF ファイルをリジェクトします。
- 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・ユーティリティは終了します。PC/IXF ファイル全体がリジェクトされ、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。

- ターゲット表の列が NULL 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、表の列の値は NULL にセットされます。
- ターゲット表の列が NULL 可能でない場合、インポート・ユーティリティは終了します。PC/IXF ファイル全体がリジェクトされ、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、NULL 可能な PC/IXF 列は NULL 可能なデータベース列を生成し、NULL 可能でない PC/IXF 列は NULL 可能でないデータベース列を生成します。
- NULL 可能でない PC/IXF 列を、NULL 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。
- NULL 可能 PC/IXF 列を、NULL 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 NULL 値が検出されると、インポート・ユーティリティは NULL 値を備えた PC/IXF 行のすべての列の値をリジェクトし (行全体がリジェクトされ)、次の PC/IXF 行から処理が継続されます。つまり、NULL のターゲット表の列が NULL 可能でない場合、その NULL 値の入った 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 ファイル全体がリジェクトされ、表は作成されず、データはインポートされません。

注: IMPORT 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。

- ターゲット表の列が NULL 可能である場合は、PC/IXF 列のすべての値が無視され、表の列の値は NULL にセットされます。
- ターゲット表の列が NULL 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 ファイル全体がリジェクトされ、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 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、無効な値の入った PC/IXF 行のすべての列の値をリジェクトし (行全体がリジェクトされ)、次の PC/IXF 行から処理が継続されます。

- DBCS データの入った 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・ページ番号が組み込まれ、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 に変換するための遷移表が入っています。

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。

関連資料:

- 307 ページの『PC/IXF データ・タイプ』
- 323 ページの『FORCEIN オプション』

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 のための遷移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はありません。

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。あるいは遷移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はありません。

FORCEIN オプションを使用すると、コード・ページの同等性チェック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。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 GRAPHIC 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 GRAPHIC 列をデータベースの GRAPHIC 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、必要な場合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 GRAPHIC タイプの列 (つまり GRAPHIC 列) にインポートされる PC/IXF 値は、必要な場合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 21 にまとめます。

表 21. 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サマリー

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			グラフ	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							

表 21. 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サマリー (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			グラフ	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0) ^d	(SBCS, DBCS) ^b	^b	DATE	TIME	TIME STAMP
-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
-(SBCS, DBCS)								N		E ^c	E ^c	E ^c
						E		E				
GRAPHIC												
									N			
						E			E			
日時												
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E
注:												
<p>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)。</p> <p>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、</p> <ul style="list-style-type: none">• 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。 <p>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。</p>												
^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リジェクトされます。												
^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リジェクトされます。												
^d データ・タイプは DBCS 環境では使用不可です。												

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・タイプまたは値が収めら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内の 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 GRAPHIC (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の列を生成します。
 - 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の列を生成します。
 - PC/IXF GRAPHIC 列は、SBCS CPGID が「未定義」で、DBCS CPGID がデータベースと等しい表 GRAPHIC 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 列を考察してみます。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 を指定しない場合、ユーティリティは終了し、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 NULL が入れられます (データベース列が NULL 可能である場合)。FORCEIN を指定した場合、ユーティリティは、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・ユーティリティによってリジェクトされること。

表 22. 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サマリー (新しい表)： この表では、a と x の間の遷移表がないことを想定しています。もしそれが ある場合、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	データベース表の列のコード・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)	リジェクト	(a,0)
(x,y)	リジェクト	(a,0)
(a,y)	リジェクト	(a,0)
(0,y)	リジェクト	(0,0)
DBCS		
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)
(x,0)	リジェクト	(a,b)
(a,b)	(a,b)	(a,b)
(x,y)	リジェクト	(a,b)
(a,y)	リジェクト	(a,b)
(x,b)	リジェクト	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	リジェクト	(-,b)

表 22. 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サマリー (新しい表) (続き): この表では、a と x の間の遷移表がないことを想定しています。もしそれがあある場合、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	データベース表の列のコード・ページ属性	
	FORCEIN を使用しない場合	FORCEIN を使用する場合
<p>注:</p> <ol style="list-style-type: none"> 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 はゼロ以外の 1 バイト・コード・ページ値、y はゼロ以外の 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 GRAPHIC データ・タイプだけです。 4. 新しい表の入った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。 <p>DBCS 環境において、新しい表を収めた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 GRAPHIC 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、GRAPHIC データ・タイプの場合は未定義であるため、'.' と示されています。</p> <ol style="list-style-type: none"> 5. 結果のデータ・タイプは、328 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. リジェクトの結果は、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。 		

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の列 (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の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、それらをリジェクトします。

表 23. 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サマリー (既存の表): この表では、a と x の間の遷移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

表 23. 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サマリー
(既存の表) (続き): この表では、a と x の間の遷移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
(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)	NULL または リジェクト	受け入れ
(a,0)	(a,0)	受け入れ	受け入れ
(x,0)	(a,0)	NULL または リジェクト	受け入れ
(x,y)	(a,0)	NULL または リジェクト	受け入れ
(a,y)	(a,0)	NULL または リジェクト	受け入れ
(0,y)	(a,0)	NULL または リジェクト	NULL または リジェクト
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)	NULL または リジェクト	受け入れ
(a,0)	(a,b)	受け入れ	受け入れ
(x,0)	(a,b)	NULL または リジェクト	受け入れ
(a,b)	(a,b)	受け入れ	受け入れ
(x,y)	(a,b)	NULL または リジェクト	受け入れ
(a,y)	(a,b)	NULL または リジェクト	受け入れ

表 23. 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サマリー
(既存の表) (続き): この表では、a と x の間の遷移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,b)	(a,b)	NULL または リジェクト	受け入れ
(0,b)	(a,b)	NULL または リジェクト	NULL または リジェクト
(0,y)	(a,b)	NULL または リジェクト	NULL または リジェクト
(0,0)	(-,b)	NULL または リジェクト	受け入れ
(a,0)	(-,b)	NULL または リジェクト	NULL または リジェクト
(x,0)	(-,b)	NULL または リジェクト	NULL または リジェクト
(a,b)	(-,b)	NULL または リジェクト	NULL または リジェクト
(x,y)	(-,b)	NULL または リジェクト	NULL または リジェクト
(a,y)	(-,b)	NULL または リジェクト	NULL または リジェクト
(x,b)	(-,b)	NULL または リジェクト	NULL または リジェクト
(0,b)	(-,b)	受け入れ	受け入れ
(0,y)	(-,b)	NULL または リジェクト	受け入れ
注:			
1. 325 ページの表 22 の注を参照してください。			
2. NULL またはリジェクトの結果は、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。			

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 IXFCBCP = 0 = IXFCDBCP) を、文字タイプのデータベース列 (SBCS CPGID がゼロ以外、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 IXFCDBC がゼロ以外) 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 (既存の表では常に可能)。
- 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。

有効な PC/IXF データ・タイプとして定義されていないデータ・タイプの PC/IXF 列は、FORCEIN オプションが指定されているかどうかにかかわらず、インポートでは無効です。

- 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 GRAPHIC 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 GRAPHIC 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 GRAPHIC 列へのインポートでは無効です。
 - 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 24 にまとめます。

表 24. 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サマリー

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			グラフ	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											
	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						

表 24. 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サマリー (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			グラフ	日時		
	SMALL INT	INT	BIGINT	DEC	FLT	(0,0)	(SBCS, 0) ^e	(SBCS, DBCS) ^b	^b	DATE	TIME	TIME STAMP
						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				
GRAPHIC												
						N w/F ^d			N			
						E			E			
日時												
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E

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 GRAPHIC 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・タイプの範囲内にはない値は、その特定の値に関してリジェクトされます。

^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リジェクトされます。

^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 環境では使用不可です。

関連資料:

- 307 ページの『PC/IXF データ・タイプ』
- 318 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』

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 ID 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・フォーマットと、その他の IXF またはデータベース・ファイル・フォーマットとの間の変換では、FOR BIT DATA 列内の値のコード・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・フォーマットのみ可能です。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ストレージの節約を活用するため、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 ID はサポートされます。可変長フィールドまたは NULL 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・フォーマット (WSF)

Lotus 1-2-3 と Lotus Symphony の製品では、同じ基本フォーマットが使用されており、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 Lotus 製品で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 Lotus 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。

- NULL 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、または公式レコードがない場合)。

注: NULL 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 フォーマットに準拠したファイルを作成すると、いくつかのデータが失われることがあります。

WSF ファイルでは、Lotus 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、Lotus 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、Lotus 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 E. エクスポート/インポート/ロード・ユーティリティの Unicode の考慮事項

エクスポート、インポート、およびロード・ユーティリティは、非 Unicode データベースに接続されている Unicode クライアントで使用される場合にはサポートされません。Unicode クライアント・ファイルは、Unicode クライアントが Unicode データベースに接続される場合にのみサポートされます。

この項で説明されているとおり、UCS-2 データベースには DEL、ASC、および PC/IXF ファイル・フォーマットがサポートされます。WSF フォーマットはサポートされません。

UCS-2 データベースから ASCII 区切り文字付き (DEL) ファイルにエクスポートする際に、すべての文字データはアプリケーション・コード・ページに変換されます。文字ストリングと GRAPHIC ストリング・データの両方が、クライアントの同じ SBCS または MBCS コード・ページに変換されます。区切り文字付き ASCII ファイル全体でコード・ページは 1 つしかないため、これがどのデータベースのエクスポートの場合でも所定の動作となり、変更できません。したがって、区切り文字付き ASCII ファイルにエクスポートする場合には、ご使用のアプリケーション・コード・ページに存在する UCS-2 文字だけが保管されます。その他の文字は、アプリケーション・コード・ページのデフォルト置換文字に置き換えられます。UTF-8 クライアント (コード・ページ 1208) の場合、UTF-8 クライアントによりすべての UCS-2 文字がサポートされているため、データの脱落はありません。

ASCII ファイル (DEL または ASC) から UCS-2 データベースにインポートする場合、文字ストリング・データはアプリケーション・コード・ページから UTF-8 に変換され、GRAPHIC ストリング・データはアプリケーション・コード・ページから UCS-2 に変換されます。データの脱落はありません。異なるコード・ページに保管された ASCII データをインポートする場合には、IMPORT コマンドを発行する前にデータ・ファイル・コード・ページを変更しなければなりません。これを実行する方法の 1 つとして、DB2CODEPAGE を ASCII データ・ファイルのコード・ページに設定します。

SBCS および MBCS クライアントで有効な ASCII 区切り文字の範囲は、これらのクライアント用に DB2[®] UDB が現在サポートしているものと同じです。UTF-8 クライアントの有効な区切り文字の範囲は X'01' から X'7F' で、通常の制約事項が適用されます。

UCS-2 データベースから PC/IXF ファイルにエクスポートする際には、文字ストリング・データはクライアントの SBCS/MBCS コード・ページに変換されます。GRAPHIC ストリング・データは変換されず、UCS-2 (コード・ページ 1200) に保管されます。データの脱落はありません。

PC/IXF ファイルから UCS-2 データベースにインポートする際には、文字ストリング・データは PC/IXF ヘッダーに格納される SBCS/MBCS コード・ページに、GRAPHIC ストリング・データは PC/IXF ヘッダーに格納される DBCS コード・ページにあるものと見なされます。文字ストリング・データは、インポート・ユーテ

イリティーにより、PC/IXF ヘッダーで指定されるコード・ページからクライアントのコード・ページに変換され、その後 (INSERT ステートメントによって) クライアント・コード・ページから UTF-8 に変換されます。GRAPHIC スtring・データは、インポート・ユーティリティーによって、PC/IXF ヘッダーで指定される DBCS コード・ページから UCS-2 (コード・ページ 1200) に直接変換されます。

ロード・ユーティリティーは、データをデータベースに直接入れ、デフォルトでは、ASC または DEL ファイル内のデータは、データベースのコード・ページにあるものと見なします。したがって、デフォルトでは、ASCII ファイルのコード・ページ変換は実行されません。データ・ファイルのコード・ページが (codepage 修飾子を使用して) 明示的に指定された場合、ロード・ユーティリティーは、データをロードする前にこの情報を使用して、指定されるコード・ページからデータベース・コード・ページに変換します。PC/IXF ファイルでは、ロード・ユーティリティーは、常に IXF ヘッダーに指定されるコード・ページからデータベース・コード・ページ (CHAR の場合は 1208、GRAPHIC の場合は 1200) に変換します。

DBCLOB ファイルのコード・ページは常に UCS-2 では 1200 です。CLOB ファイルのコード・ページは、インポート、ロード、あるいはエクスポートされるデータ・ファイルのコード・ページと同じです。たとえば、PC/IXF フォーマットを使用してデータをロードまたはインポートする際には、CLOB ファイルは、PC/IXF ヘッダーによって指定されるコード・ページにあるものと見なされます。DBCLOB ファイルが ASC または DEL フォーマットの場合には、ロード・ユーティリティーは、(codepage 修飾子を使って明示的に指定されないかぎり) CLOB データがデータベースのコード・ページにあるものと見なし、インポート・ユーティリティーは、これがクライアント・アプリケーションのコード・ページにあるものと見なします。

以下の理由で、UCS-2 データベースには nochecklengths 修飾子が常に指定されます。

- DBCS コード・ページのないデータベースには任意の SBCS を接続できる。
- UTF-8 フォーマットの文字ストリングは通常、クライアント・コード・ページにあるものとは異なる長さになる。

コード・ページ 1394、1392、および 5488 の制約事項

インポート、エクスポート、およびロード・ユーティリティーを使用して、新しい中国語コード・ページ GB 18030 (コード・ページ ID 1392 および 5488) および新しい日本語コード・ページ ShiftJISX 0213 (コード・ページ ID 1394) から DB2 UDB Unicode データベースにデータを転送できるようになりました。加えて、エクスポート・ユーティリティーを使用して、DB2 UDB Unicode データベースから GB 18030 または ShiftJIS X0213 コード・ページ・データにデータを転送することもできます。

たとえば、以下のコマンドは、リモートで接続されるクライアントに常駐する Shift_JISX0213 データ・ファイル u/jp/user/x0213/data.del を MYTABLE にロードします。

```
db2 load client from /u/jp/user/x0213/data.del  
of del modified by codepage=1394 insert into mytable
```


ここで、MYTABLE は DB2 UDB Unicode データベースにあります。

Unicode クライアントと Unicode サーバーとの間の接続しかサポートされないため、Unicode クライアントを使用するか、あるいはロード、インポート、またはエクスポート・ユーティリティを使用する前に、DB2 レジストリー変数 DB2CODEPAGE を 1208 に設定する必要があります。

コード・ページ 1394、1392、または 5488 から Unicode に変換すると、拡張が行われます。たとえば、2 バイト文字は、GRAPHIC 列で 2 つの 16 ビット Unicode 文字として格納されます。Unicode データベースのターゲット列は、拡張された Unicode バイトを入れるのに十分な幅であることを確認する必要があります。

非互換性

UCS-2 データベースに接続されているアプリケーションの場合、GRAPHIC ストリング・データは常に UCS-2 (コード・ページ 1200) にあります。非 UCS-2 データベースに接続されるアプリケーションの場合、GRAPHIC ストリング・データは、アプリケーションの DBCS コード・ページにあるか、またはアプリケーション・コード・ページが SBCS の場合には許可されません。たとえば、日本語の非 UCS 2 データベースに 932 クライアントが接続される場合、GRAPHIC ストリング・データはコード・ページ 301 に入れられます。UCS-2 データベースに接続されている 932 クライアント・アプリケーションの場合、GRAPHIC ストリング・データは UCS-2 にあります。

関連資料:

- 281 ページの『DEL のデータ・タイプの説明』
- 283 ページの『区切りなし ASCII (ASC) ファイル・フォーマット』
- 287 ページの『PC バージョンの IXF ファイル・フォーマット』

付録 F. 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使用されるバインド・ファイル

以下の表は、バインド・ファイルとそのデフォルト分離レベルをリストし、さらにどのユーティリティーがどんな目的で使用するかを説明しています。

バインド・ファイル (デフォルト分離レベル)	ユーティリティー/目的
db2ueiwi.bnd (CS)	インポート/エクスポート。表の列および索引に関する情報を照会するために使用されます。
db2uexpm.bnd (CS)	エクスポート。エクスポート操作に指定される SQL 照会を取り出すのに使用されます。
db2uimpb.bnd (RS)	インポート。ソース・データ・ファイルからターゲット表にデータを挿入するために使用されます。
db2uipkg.bnd (CS)	インポート。 BIND オプションをチェックするために使用されます。
db2uiici.bnd (RR)	インポート。 IXF CREATE オプションが指定されたときに索引を作成するために使用されます。
db2ucktb.bnd (CS)	ロード。ロード操作で一般初期化プロセスを実行するために使用されます。
db2ulxld.bnd (CS)	ロード。カーソル操作によってロード時に提供される SQL 照会を処理するために使用されます。
db2uigsi.bnd (UNIX ベースのシステムでは RS、その他のプラットフォームでは RR)	インポート/エクスポート。索引をドロップし、インポート置換操作で参照制約をチェックするために使用されます。また、 IXF ファイルをエクスポートするための ID 列情報を検索するのに使用されます。
db2uiict.bnd (RR)	インポート。 IXF CREATE オプションが指定される際に表を作成するために使用されます。
db2uqtpd.bnd (RR)	インポート/エクスポート。階層表の処理を実行するために使用されます。
db2uqtnm.bnd (RR)	インポート。 IXF CREATE オプションが指定されたときに階層表の処理を実行するために使用されます。
db2uimtb.bnd (RS)	インポート。インポート操作で一般初期化プロセスを実行するために使用されます。

付録 G. 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、さまざまなユーティリティによって生成されるものが組み込まれます。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 ID 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 ID 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 ID 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。

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、適切なリカバリー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。UNIX ベースのシステムでは、2 重引用符区切り文字の使用をお勧めします。これにより、ディレクトリーに単一文字ファイル名がある場合の問題を回避できます。

```
db2 "? SQLnnnnn "
```

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 ID 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、次のコマンドを実行することによって検索できます。

```
db2 ? nnnnn  
or  
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 H. DB2 Universal Database 技術情報

DB2 資料とヘルプ

DB2® 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
 - トピック
 - DB2 ツールのヘルプ
 - サンプル・プログラム
 - チュートリアル
- ダウンロード可能な PDF ファイル、CD 上の PDF ファイル、および印刷された資料
 - ガイド
 - リファレンス・マニュアル
- コマンド行ヘルプ
 - コマンド・ヘルプ
 - メッセージ・ヘルプ
 - SQL 状態ヘルプ
- インストール済みソース・コード
 - サンプル・プログラム

ibm.com® にある技術資料、白書、Redbooks™ その他の DB2 Universal Database™ 技術情報にオンラインでアクセスできます。DB2 Information Management ソフトウェア・ライブラリー・サイト (www.ibm.com/software/data/pubs/) にアクセスしてください。

DB2 資料の更新

IBM® は、DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。DB2 インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2help/>) にアクセスすれば、常に最新の情報が掲載されます。DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、DB2 インフォメーション・センター CD からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

関連タスク:

- 349 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

関連資料:

- 342 ページの『DB2 PDF 資料および印刷された資料』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役立つ内容です。

表 25. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database コマンド・リファレンス」	SC88-9140	db2n0j81
「IBM DB2 Universal Database 用語集」	資料番号なし	db2t0j81
「IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻」	GC88-9152 (ハードコピーな し)	db2m1j81
「IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻」	GC88-9153 (ハードコピーな し)	db2m2j81

表 25. DB2 の基本情報 (続き)

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database 新機能」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 26. 管理情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database 管理ガイド: プランニング」	SC88-9135	db2d1j81
「IBM DB2 Universal Database 管理ガイド: インプリメンテー ション」	SC88-9133	db2d2j81
「IBM DB2 Universal Database 管理ガイド: パフォーマンス」	SC88-9134	db2d3j81
「IBM DB2 Universal Database 管理 API リファレンス」	SC88-9136	db2b0j81
「IBM DB2 Universal Database データ移動ユーティリティー ガイドおよびリファレンス」	SC88-9142	db2dmj81
「IBM DB2 Universal Database データ・リカバリーと高可用性 ガイドおよびリファレンス」	SC88-9143	db2haj81
「IBM DB2 Universal Database データウェアハウス・センター 管理ガイド」	SC88-9165	db2ddj81
「IBM DB2 Universal Database SQL リファレンス 第 1 巻」	SC88-9155	db2s1j81
「IBM DB2 Universal Database SQL リファレンス 第 2 巻」	SC88-9156	db2s2j81
「IBM DB2 Universal Database システム・モニター ガイドお よびリファレンス」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 27. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および 実行」	SC88-9137	db2axj81
「IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーション のプログラミング」	SC88-9138	db2a1j81
「IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションの プログラミング」	SC88-9139	db2a2j81
「IBM DB2 Universal Database コール・レベル・インターフェ ース ガイドおよびリファレン ス 第 1 巻」	SC88-9159	db211j81
「IBM DB2 Universal Database コール・レベル・インターフェ ース ガイドおよびリファレン ス 第 2 巻」	SC88-9160	db212j81
「IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド」	SC88-9166	db2adj81
「IBM DB2 Universal Database XML Extender 管理およびプロ グラミングのガイド」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 28. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Warehouse Manager Standard Edition イン フォメーション・カタログ・セ ンター 管理ガイド」	SC88-9167	db2dij81
「IBM DB2 Warehouse Manager Standard Edition イン ストール・ガイド」	GC88-9164	db2idj81

表 28. ビジネス・インテリジェンス情報 (続き)

資料名	資料番号	PDF ファイル名
「IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 29. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「IBM コネクティビティ 補足」	資料番号なし	db2h1j81
「IBM DB2 Connect Enterprise Edition 概説およびインストール」	GC88-9145	db2c6j81
「IBM DB2 Connect Personal Edition 概説およびインストール」	GC88-9146	db2c1j81
「IBM DB2 Connect ユーザーズ・ガイド」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 30. 入門情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール」	GC88-9144 (ハードコピーなし)	db2itj81
「IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール」	GC88-9148	db2isj81
「IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール」	GC88-9150	db2i1j81
「IBM DB2 Universal Database インストールおよび構成 補足」	GC88-9149 (ハードコピーなし)	db2iyj81

表 30. 入門情報 (続き)

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database DB2 Data Links Manager 概説 およびインストール」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 31. チュートリアル情報

資料名	資料番号	PDF ファイル名
「ビジネス・インテリジェン ス・チュートリアル: データ ウェアハウス・センターの紹 介」	資料番号なし	db2tuj81
「ビジネス・インテリジェン ス・チュートリアル: データ ウェアハウジングの上級者向 けガイド」	資料番号なし	db2taj81
「インフォメーション・カタ ログ・センター チュートリア ル」	資料番号なし	db2aij81
「Video Central for e-business チュートリアル」	資料番号なし	db2twj81
「Visual Explain チュートリア ル」	資料番号なし	db2tvj81

オプション・コンポーネント情報

このカテゴリーの情報は、DB2 のオプション・コンポーネントを使用する方法について説明しています。

表 32. オプション・コンポーネント情報

資料名	資料番号	PDF ファイル名
「IBM DB2 Cube Views Guide and Reference」	SC18-7298	db2aax81
「IBM DB2 Query Patroller イ ンストール、管理、使用法の ガイド」	GC88-9154	db2dwj81
「IBM DB2 Spatial Extender and Geodetic Extender ユーザ ーズ・ガイドおよびリファレ ンス」	SC88-9171	db2sbj81

表 32. オptional・コンポーネント情報 (続き)

資料名	資料番号	PDF ファイル名
「IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス」	SC88-9169	db2z0x82
「DB2 Net Search Extender 管理およびユーザズ・ガイド」 注: この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 33. リリース・ノート

資料名	資料番号	PDF ファイル名
「DB2 リリース・ノート」	「注」を参照。	「注」を参照。
「DB2 インストール情報」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『既知の問題と予備手段』および『リリース間の非互換性』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、Release.Notes ファイルを参照してください。このファイルは、DB2DIR/Readme/%L ディレクトリーに収録されています。 %L はロケール名を表しています。 DB2DIR は以下になります。

- AIX オペレーティング・システムの場合: /usr/opt/db2_08_01
- その他のすべての UNIX ベースのオペレーティング・システムの場合: /opt/IBM/db2/V8.1

関連概念:

- 341 ページの『DB2 資料とヘルプ』

関連タスク:

- 348 ページの『PDF ファイルからの DB2 資料の印刷方法』

- 349 ページの『DB2 の印刷資料の注文方法』
- 349 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料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できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。Adobe Acrobat Reader をインストールする必要がある場合、Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. DB2 PDF ドキュメンテーション CD をドライブに挿入します。UNIX オペレーティング・システムの場合、DB2 PDF ドキュメンテーション CD をマウントします。UNIX オペレーティング・システムで CD をマウントする方法については、「概説およびインストール」を参照してください。
2. index.htm を開きます。ブラウザ・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。Acrobat Reader で PDF が開きます。
4. 「ファイル」→「印刷」を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 360 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROM のマウント (AIX)』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『HP-UX 上での CD-ROM のマウント』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROM のマウント (Linux)』
- 349 ページの『DB2 の印刷資料の注文方法』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROM のマウント (Solaris)』

関連資料:

- 342 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようになすことができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (<http://www.ibm.com/shop/publications/order>) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は DB2 PDF ドキュメンテーション CD にある PDF 形式の資料と同じものです。さらに、DB2 インフォメーション・センター CD に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、PDF 資料にならない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、PDF またはハードコピーの資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、DB2 インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2help/>) を参照して最新の情報を入手してください。

関連タスク:

- 348 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 342 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある「ヘルプ」ボタンからアクセス可能なヘルプ

- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある「ヘルプ」ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

また、それぞれの DB2 ツール・センターの上部にある「ヘルプ」メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。

- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの「**文書 (Documentation)**」ページの「**infopops の自動表示**」チェック・ボックスを選択します。

infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。

- 必須フィールド。
- 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 350 ページの『コマンド行プロセッサからメッセージ・ヘルプを呼び出す』
- 351 ページの『コマンド行プロセッサからコマンド・ヘルプを呼び出す』
- 352 ページの『コマンド行プロセッサから SQL 状態ヘルプを呼び出す』

コマンド行プロセッサからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

メッセージ・ヘルプを呼び出すには、コマンド行プロセッサを開いて以下のように入力します。

`? XXXnnnnn`

ここで、`XXXnnnnn` は有効なメッセージ ID 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。

関連タスク:

- 349 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 351 ページの『コマンド行プロセッサからコマンド・ヘルプを呼び出す』
- 352 ページの『コマンド行プロセッサから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサの呼び出しコマンド』

コマンド行プロセッサからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサでのコマンドの構文を説明します。

手順:

コマンド・ヘルプを呼び出すには、コマンド行プロセッサを開いて以下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` コマンドのヘルプだけが表示されます。

関連タスク:

- 349 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 350 ページの『コマンド行プロセッサからメッセージ・ヘルプを呼び出す』
- 352 ページの『コマンド行プロセッサから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサの呼び出しコマンド』

コマンド行プロセッサから SQL 状態ヘルプを呼び出す

DB2 Universal Database は、SQL ステートメントの結果の原因となったと考えられる条件の SQLSTATE 値を戻します。SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順:

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサを開いて以下のように入力します。

`? sqlstate` または `? 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` を指定すると SQL 状態 08003 のヘルプが表示され、`? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

関連タスク:

- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 350 ページの『コマンド行プロセッサからメッセージ・ヘルプを呼び出す』
- 351 ページの『コマンド行プロセッサからコマンド・ヘルプを呼び出す』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、DB2 Connect、DB2 Information Integrator、DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- DB2 インフォメーション・センターがインストールされているイントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- オプション: 希望する言語でトピックを表示するようブラウザーを構成する
- オプション: コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようになります。

- (Windows オペレーティング・システムの)「スタート」メニューから:「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」をクリックします。
- コマンド行プロンプトから:
 - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。
 - Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピュータにインストール済みの DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ <http://<host-name>:<port-number>/> を開きます (<host-name> はホスト名、<port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 360 ページの『DB2 インフォメーション・センター』

関連タスク:

- 370 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 349 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 353 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 350 ページの『コマンド行プロセッサからメッセージ・ヘルプを呼び出す』
- 351 ページの『コマンド行プロセッサからコマンド・ヘルプを呼び出す』
- 352 ページの『コマンド行プロセッサから SQL 状態ヘルプを呼び出す』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

<http://publib.boulder.ibm.com/infocenter/db2help/> から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

DB2 インフォメーション・センターの更新をコンピューターまたはイントラネット・サーバーにインストールするには、以下のようにします。

1. IBM の Web サイト (<http://publib.boulder.ibm.com/infocenter/db2help/>) にある DB2 インフォメーション・センターを開きます。
2. 「DB2 インフォメーション・センターによるこそ」 ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、「**DB2 資料**」リンクをクリックします。
3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターによるこそ」 ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の DB2 インフォメーション・センター・イメージをダウンロードします。
5. 最新の DB2 インフォメーション・センター・イメージをインストールするには、Web ページの指示に従ってください。

関連概念:

- 367 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 362 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 364 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 トラブルシューティング情報

DB2® 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション

トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中にご利用いただけます。DB2 インフォメーション・センターで、(ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト

現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。

Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最

新リスト、その他のリソースが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト

(<http://www.ibm.com/software/data/db2/udb/winos2unix/support>) にアクセスしてください。

DB2 Problem Determination Tutorial Series

DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support

(<http://www.ibm.com/software/data/support/pdm/db2tutorials.html>) には、DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 360 ページの『DB2 インフォメーション・センター』
- 『問題判別の手引き』の中の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。DB2[®] バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、356 ページの『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、356 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java[™] Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、356 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、357 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティを向上させる機能が備わっています。これらのアクセシビリティ拡張機能には、フォント・プロパティのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。XHTML により、ご使用のブラウザに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 357 ページの『ドット 10 進シンタックス・ダイアグラム』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。* シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のあ

る別の行に表示できます。またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。% シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数ある場合、? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?, 5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。? シンボルは、線路型ダイアグラムのバイパス線に相当します。
- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。* シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

注:

1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
 2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
 3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+, 2 HOST、および 2 STATE」という行を聞き取る場合には、HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、+ シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連資料:

- 「SQL リファレンス 第 2 巻」の『構文図の見方』

DB2 チュートリアル

DB2® チュートリアルは、DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2help/>) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介 データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド

データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル

インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

Visual Explain チュートリアル

Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 インフォメーション・センター

DB2[®] インフォメーション・センターを使用すると、DB2 Universal Database[™]、DB2 Connect[™]、DB2 Information Integrator および DB2 Query Patroller[™]などのDB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft[®] Internet Explorer 5.5 以上で表示する場合、DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、JavaScript[™] のサポートを使用可能にする必要があります:

柔軟なインストール・オプション

以下の中から、ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、IBM[®] の Web サイト (<http://publib.boulder.ibm.com/infocenter/db2help/>) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピュータに DB2 資料をインストールします。

検索 「検索」テキスト・フィールドに検索語を入力することにより、DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次

単一の目次の中から、DB2 資料のトピックを見付けることができます。目次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報（ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど）が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」ボタンをクリックするか、コンテンツ・フレーム内の「目次に表示 (Show in Table of Contents)」ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 367 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 353 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 370 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 352 ページの『DB2 インフォメーション・センターの呼び出し』
- 362 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 364 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスする方法として、IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、DB2 インフォメーション・センター CD から資料をインストールする必要があります。DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

• ハードウェア要件

以下のいずれかのプロセッサが必要です。

- PowerPC (AIX)
- HP 9000 (HP-UX)
- Intel 32 ビット (Linux)
- Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

• オペレーティング・システム要件

以下のいずれかのオペレーティング・システムが必要です。

- IBM AIX 5.1 (PowerPC 上)
- HP-UX 11i (HP 9000 上)
- Red Hat Linux 8.0 (Intel 32 ビット上)
- SuSE Linux 8.1 (Intel 32 ビット上)
- Sun Solaris バージョン 8 (Solaris オペレーティング環境の UltraSPARC コンピューター上)

注: DB2 インフォメーション・センターは、DB2 クライアントをサポートする UNIX オペレーティング・システム上で稼動します。このため、IBM Web サイトから DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

• ソフトウェア要件

- 以下のブラウザがサポートされています。
 - Mozilla バージョン 1.0 以上

- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードのグラフィカル・ユーザー・インターフェースを表示可能にする X Window システム・ソフトウェアをインプリメン

トする必要があります。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、コマンド・プロンプトで

```
export DISPLAY=9.26.163.144:0.
```

というコマンドを入力します。

- 通信要件

- TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. システムにログオンします。
2. DB2 インフォメーション・センター製品 CD を挿入してシステムにマウントします。
3. 次のコマンドを入力して、CD がマウントされているディレクトリに移動します。

```
cd /cd
```

`/cd` は、CD のマウント・ポイントを表します。

4. **`/db2setup`** コマンドを入力して、DB2 セットアップ・ウィザードを開始します。
5. IBM DB2 セットアップ・ランチパッドが開きます。DB2 インフォメーション・センターのインストールに直接進むには、「製品のインストール」をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、「ヘルプ」をクリックします。「キャンセル」をクリックすれば、いつでもインストールを終了できます。
6. 「インストールしたい製品を選択します」ページでは、「次へ」をクリックします。
7. 「DB2 セットアップ・ウィザードによるこそ (Welcome to the DB2 Setup wizard)」ページで、「次へ」をクリックします。DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
8. インストールを続行するには、使用許諾条件に同意する必要があります。「ご使用条件」ページで、「ご使用条件に同意します (I accept the terms in the license agreement)」をクリックして、「次へ」をクリックします。
9. 「インストール・アクションの選択」で、「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、「設定を応答ファイルに保管する」を選択します。「次へ」をクリックします。
10. 「インストールする言語の選択」ページでは、DB2 インフォメーション・センターをインストールする言語を選択します。「次へ」をクリックします。

11. 「DB2 インフォメーション・センター・ポートの指定」ページでは、DB2 インフォメーション・センターへの着信通信を構成します。「次へ」をクリックしてインストールを続けます。
12. 「ファイルのコピーの開始」ページでは、インストールの選択項目を確認します。設定を変更するには、「戻る」をクリックします。「インストール」をクリックすると、DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

このほか、応答ファイルを使って DB2 インフォメーション・センターをインストールすることもできます。

インストール・ログ db2setup.his、db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに置かれます。

db2setup.log ファイルは、エラーも含めた DB2 製品のインストール情報をすべてキャプチャーします。db2setup.his ファイルは、コンピューター上の DB2 製品インストール内容をすべて記録します。DB2 は、db2setup.log ファイルを db2setup.his に付加します。db2setup.err ファイルは、Java から戻されるすべてのエラー出力 (例外やトラップの情報など) をキャプチャーします。

インストールが完了したら、ご使用の UNIX オペレーティング・システムに応じて、DB2 は以下のいずれかのディレクトリーにインストールされます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境: /opt/IBM/db2/V8.1

関連概念:

- 367 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 370 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 364 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)

DB2 製品資料にアクセスする方法として、IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、DB2 インフォメーション・センター CD から DB2 資料をインストールする必要があります。DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

• ハードウェア要件

以下のいずれかのプロセッサが必要です。

- 32 ビット・コンピュータ: Pentium または Pentium 互換の CPU

• オペレーティング・システム要件

以下のいずれかのオペレーティング・システムが必要です。

- Windows 2000
- Windows XP

注: DB2 インフォメーション・センターは、DB2 クライアントをサポートする Windows オペレーティング・システム上で稼動します。このため、IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

• ソフトウェア要件

- 以下のブラウザがサポートされています。

- Mozilla 1.0 以上
- Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)

• 通信要件

- TCP/IP

制約事項:

- DB2 インフォメーション・センターをインストールするには、管理権限をもつアカウントが必要です。

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようになります。

1. DB2 インフォメーション・センターのインストールで定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能が使用可能になっていれば、IBM DB2 セットアップ・ランチパッドが起動します。
3. DB2 セットアップ・ウィザードは、システム言語を判別して、その言語用のセットアップ・プログラムを立ち上げます。英語以外の言語でセットアップ・プログラムを実行したい場合、またはセットアップ・プログラムの自動始動が失敗した場合には、DB2 セットアップ・ウィザードを手動で開始できます。

次のようにして、DB2 セットアップ・ウィザードを手動で開始します。

- a. 「スタート」をクリックし、「ファイル名を指定して実行」を選択します。
- b. 「開く」フィールドで、以下のコマンドを入力します。

x:¥setup.exe /i 2-letter language identifier

ここで、x: は CD ドライブ、2-letter language identifier (2 文字の言語識別子) はセットアップ・プログラムを実行する言語を表します。

c. 「OK」をクリックします。

4. IBM DB2 セットアップ・ランチパッドが開きます。DB2 インフォメーション・センターのインストールに直接進むには、「製品のインストール」をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、「ヘルプ」をクリックします。「キャンセル」をクリックすれば、いつでもインストールを終了できます。
5. 「インストールしたい製品を選択します」ページでは、「次へ」をクリックします。
6. 「DB2 セットアップ・ウィザードによるこそ (Welcome to the DB2 Setup wizard)」ページで、「次へ」をクリックします。DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
7. インストールを続行するには、使用許諾条件に同意する必要があります。「ご使用条件」ページで、「ご使用条件に同意します (I accept the terms in the license agreement)」をクリックして、「次へ」をクリックします。
8. 「インストール・アクションの選択」で、「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、「設定を応答ファイルに保管する」を選択します。「次へ」をクリックします。
9. 「インストールする言語の選択」ページでは、DB2 インフォメーション・センターをインストールする言語を選択します。「次へ」をクリックします。
10. 「DB2 インフォメーション・センター・ポートの指定」ページでは、DB2 インフォメーション・センターへの着信通信を構成します。「次へ」をクリックしてインストールを続けます。
11. 「ファイルのコピーの開始」ページでは、インストールの選択項目を確認します。設定を変更するには、「戻る」をクリックします。「インストール」をクリックすると、DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

応答ファイルを使って DB2 インフォメーション・センターをインストールすることができます。また、db2rspgn コマンドを使って、既存のインストール内容に基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、「マイ ドキュメント」¥DB2LOG¥ ディレクトリー内の db2.log ファイルと db2wi.log ファイルを参照してください。「マイ ドキュメント」ディレクトリーの場所は、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルは、DB2 の最新のインストール情報をキャプチャーします。db2.log は、DB2 製品のインストールの履歴をキャプチャーします。

関連概念:

- 367 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 370 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 362 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまに異なる業務環境のもとでは、DB2® 情報にどのようにアクセスするかの要件もそれぞれ異なります。DB2 インフォメーション・センターにアクセスするには、IBM® の Web サイト、サーバーまたは組織のネットワーク、あるいはコンピューターへのインストールという 3 つの方法が可能です。この 3 つのケースのいずれも、資料は DB2 インフォメーション・センター内に置かれます。インフォメーション・センターは、ブラウザを使って表示できるように設計されたトピック・ベースの情報の Web サイトです。デフォルトでは、DB2 製品から、IBM Web サイト上の DB2 インフォメーション・センターにアクセスします。これに対して、イントラネット・サーバーまたはご自分のコンピューターから DB2 インフォメーション・センターにアクセスしたい場合、製品メディア・パック内にある DB2 インフォメーション・センター CD から DB2 インフォメーション・センターをインストールする必要があります。以下では、DB2 資料へのアクセス・オプションの要約、および 3 つのインストール・シナリオを示します。これを参考にして、お客様の業務環境で DB2 インフォメーション・センターにアクセスするにはどの方法が最適か、どのようなインストール上の問題に配慮する必要があるかを判断してください。

DB2 資料にアクセスするオプションの要約:

以下の表は、お客様の実際の業務環境で、DB2 インフォメーション・センターの DB2 製品情報にアクセスする方法としてどんなオプションが推奨されるかを示します。

インターネット・アクセス	イントラネット・アクセス	推奨されるアクション
はい	はい	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス、またはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
はい	いいえ	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス
いいえ	はい	イントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
いいえ	いいえ	ローカル・コンピューター上の DB2 インフォメーション・センターへのアクセス

シナリオ: コンピューター上の DB2 インフォメーション・センターへのアクセス:

Tsu-Chen 氏は小さな町で工場を経営していますが、その町には、インターネット・アクセスを提供する地元のインターネット・サービス・プロバイダーがありません。彼は、在庫、製品オーダー、銀行口座情報、および営業経費を管理するために DB2 Universal Database™ を購入しました。Tsu-Chen 氏は以前に DB2 製品を利用したことがないので、DB2 の使用方法を習得するために、DB2 製品資料を参照する必要があります。

Tsu-Chen 氏は 標準インストール・オプションを使って DB2 Universal Database を自分のコンピューターにインストールした後、DB2 資料にアクセスしようとしします。しかし、開こうとしているページが見つからないというエラー・メッセージがブラウザから通知されました。Tsu-Chen 氏は DB2 製品のインストール・マニュアルを調べた結果、DB2 資料を自分のコンピューター上で利用するには、DB2 インフォメーション・センターをインストールしなければならないことに気がきます。そしてメディア・バックの中にあった DB2 インフォメーション・センター CD を見つけ出して、インストールしました。

これで、Tsu-Chen 氏はオペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターにアクセスできるようになり、より良い業務成果をあげるために DB2 製品を利用する方法を習得できます。

シナリオ: IBM Web サイト上の DB2 インフォメーション・センターへのアクセス:

Colin は、あるセミナー企業に所属する情報技術コンサルタントです。彼の専門はデータベース・テクノロジーおよび SQL で、DB2 Universal Database を使って北米一帯の企業を対象にこれらの科目のセミナーを開催しています。Colin のセミナーでは、教材として DB2 資料も使用されます。たとえば、SQL の講習コースでは、データベース照会の基本構文と拡張構文を教えるために SQL に関する DB2 資料が使用されます。

Colin が教えている企業の大半はインターネット・アクセスを配備しています。このような状況から判断して、Colin は、最新バージョンの DB2 Universal Database を自分のモバイル・コンピューターにインストールしたとき、IBM Web サイト上の DB2 インフォメーション・センターにアクセスするよう構成しました。この構成によって、Colin はセミナーで教えるときに最新の DB2 資料にオンライン・アクセスすることができます。

しかし、時折、Colin は移動中にインターネット・アクセスを利用できないことがあります。これは問題となります。担任するセミナーの準備のために DB2 資料にアクセスする必要のある場合には、とくにそうです。このような事態が起きないようにするために、Colin は自分のモバイル・コンピューターに DB2 インフォメーション・センターのコピーをインストールしました。

こうして、Colin は常に DB2 資料のコピーを自在に活用できるようになりました。**db2set** コマンドを使って自分のモバイル・コンピューターのレジストリー変数を簡単に構成し、どこにいるかに応じて、IBM Web サイトまたは自分のモバイル・コンピューターから DB2 インフォメーション・センターにアクセスできます。

シナリオ: イン트라ネット・サーバー上の DB2 インフォメーション・センターへのアクセス:

Eva は、生命保険会社のデータベース上級管理者です。彼女は管理業務の一環として、会社の UNIX® データベース・サーバーに最新バージョンの DB2 Universal Database をインストールおよび構成します。彼女の会社は最近、セキュリティ上の理由から、インターネット・アクセスをほぼ業務で利用できないようにすると社員に通知しました。同社はネットワーク環境を装備しているため、Eva は DB2 インフォメーション・センターのコピーをイントラネット・サーバー上にインストールして、社内のデータウェアハウスを定期的に利用するすべての社員（営業担当者、営業部長、および業務分析担当者）から DB2 資料へのアクセスを可能にすることにしました。

Eva は、応答ファイルを使って全社員のコンピューター上に最新バージョンの DB2 Universal Database をインストールするようデータベース・チームに指示します。その際、イントラネット・サーバーのホスト名とポート番号を使って DB2 インフォメーション・センターにアクセスできるよう、確実に各コンピューターを構成します。

しかし、Eva のチームの下級データベース管理者である Migual の誤解によって、数人の社員のコンピューター上で、イントラネット・サーバーの DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成する代わりに、DB2 インフォメーション・センターのコピーをそれらのコンピューターにインストールしてしまいました。これを訂正するために、Eva は、**db2set** コマンドを使ってこれらのコンピューター上の DB2 インフォメーション・センターのレジストリー変数（ホスト名は DB2_DOCHOST、ポート番号は DB2_DOCPORT）を変更するよう Migual に指示しました。これで、ネットワーク上の適切なすべてのコンピューターが DB2 インフォメーション・センターにアクセスできるようになり、社員は DB2 に関する質問の答えを DB2 資料から見つけることができます。

関連概念:

- 360 ページの『DB2 インフォメーション・センター』

関連タスク:

- 353 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 362 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 364 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 インフォメーション・センターにおける特定の言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、DB2 インフォメーション・センターでは英語でトピックが表示されます。

手順:

Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の「ツール」→「インターネット オプション」→「言語...」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
 - ・ リストに新しい言語を追加するには、「追加...」ボタンをクリックします。

注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。

- ・ リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上へ」ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

Mozilla Web ブラウザーの場合に、使いたい言語でトピックを表示するには、以下のようにします。

1. Mozilla の「編集」→「設定」→「言語」ボタンをクリックします。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
 - ・ リストに新しい言語を追加するには、「追加...」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
 - ・ リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上に移動」ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 <http://niap.nist.gov/cc-scheme/>

付録 I. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービス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、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービス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 は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
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を提供し、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。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

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、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

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
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができませんが、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。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者をお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標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。より具体性を与えるために、それらの例には、個人、企業、ブランド、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。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、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手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

商標

以下は、IBM Corporation の商標です。

ACF/VTAM	iSeries
AISPO	LAN Distance
AIX	MVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	NetView
BookManager	OS/390
C Set++	OS/400
C/370	PowerPC
CICS	pSeries
Database 2	QBIC
DataHub	QMF
DataJoiner	RACF
DataPropagator	RISC System/6000
DataRefresher	RS/6000
DB2	S/370
DB2 Connect	SP
DB2 Extenders	SQL/400
DB2 OLAP Server	SQL/DS
DB2 Information Integrator	System/370
DB2 Query Patroller	System/390
DB2 Universal Database	SystemView
Distributed Relational Database Architecture	Tivoli
DRDA	VisualAge
eServer	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
IBM	WebSphere
IMS	WIN-OS/2
IMS/ESA	z/OS
	zSeries

以下は、それぞれ各社の商標または登録商標です。

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。

Pentium は、Intel 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 の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]

アクセス可能性
 小数点付き 10 進数構文図 357
 フィーチャー 355
アプリケーション・レコード、
 PC/IXF 289
一時ファイル
 ロード・コマンド 113
 ロード・ユーティリティ 183
移動、データの
 データベース間の 41, 55
印刷
 PDF ファイル 348
印刷資料、注文 349
インストール
 インフォメーション・センター 362, 364, 367
インフォメーション・センター
 インストール 362, 364, 367
インポート
 型付き表へ 55
 コード・ページについての考慮事項 55
 制約事項 55
 存在しない表または階層へ 55
 データ 41
 ファイルからデータベース表へ 55
 ファイル・タイプ修飾子 55
 リモート・データベースへ 55
 DB2 Connect によるデータベース・アクセス 55
 DB2 Data Links Manager の考慮事項 55
 PC/IXF ファイル、一般規則 318
 PC/IXF ファイル、データ・タイプ固有の規則 320
 PC/IXF ファイルの、forcein の 323
 PC/IXF、複数パーツ・ファイル 55
インポート API 55
インポート、メッセージ・ファイル 1, 29, 84
エクスポート
 データベース表ファイル 9, 14
 ファイル・タイプ修飾子 9, 14

エクスポート (続き)
 列名の指定 14
 DB2 Data Links Manager の考慮事項 9
エクスポート API 14
エクスポート、メッセージ・ファイル 1, 29, 84
エクスポートされた表
 インポート・ユーティリティを使用
 した再作成 37
 表属性が IXF ファイルに格納されな
 い場合の再作成 37
 表属性が IXF ファイルに格納され
 る場合の再作成 37
 EXPORT ユーティリティを使用した
 再作成 4
エクスポート操作、データウェアハウス・
 センター 257
エラー・メッセージ
 概要 339
オプション
 forcein 323
オンライン
 ヘルプ、アクセス 349

[カ行]

階層レコード、PC/IXF 289
型付き表
 インポート 250
 エクスポート 250
 走査順序 29, 252
 データ移動中の選択 253
 データ移動の例 253
 データの移動 250
完了メッセージ 339
キーボード・ショートカット
 サポート 355
キーワード
 構文 261
区切り付き ASCII (DEL) ファイル・フォ
 ーマット 278
 プラットフォーム間のデータの移動
 235
区切りなし ASCII (ASC) ファイル・フォ
 ーマット 283
区切り文字ストリング 280
警告メッセージ
 概要 339
継続レコード・タイプ、PC/IXF 289

コード・ページ
 インポート API 55
 インポート・ユーティリティについ
 ての考慮事項 78
エクスポート API 14
変換
 ファイル 318
 PC/IXF データのインポートまたは
 ロード時 318
ロード・ユーティリティについての
 考慮事項 187
EXPORT コマンド 9
IMPORT コマンド 41
更新
 HTML 文書 353
構造
 区切り付き ASCII (DEL) ファイル
 278
 区切りなし ASCII (ASC) ファイル
 283
構文
 変更、ロード・ユーティリティ 84
構文図
 読み方 261
互換性のない列 318
コマンド
 db2move 239
 db2relocatedb 245
 EXPORT 9
 IMPORT 41
 LOAD 113
 LOAD QUERY 137
コマンド構文
 解釈 261
コマンド・ヘルプ
 呼び出し 351

[サ行]

再始動、ロード操作の
 パーティション・データベースのロー
 ド操作 211
 読み取りアクセス・モードの許可 110
索引
 作成 97
索引の作成 97
索引レコード、PC/IXF 289
サマリー表
 インポートの制約事項 31
識別レコード、PC/IXF 289

修飾子
 ファイル・タイプ
 ロード・コマンド 113
 EXPORT コマンド 9
 IMPORT コマンド 41
修飾子ファイル・タイプ
 インポート・ユーティリティ 55
 エクスポート・ユーティリティ 14
 ロード API 139
終了
 ロード操作
 パーティション・データベースでの 211
 読み取りアクセス・モードの許可 110
終了レコード、PC/IXF 289
小数点付き 10 進数構文図 357
状態
 削除ペンディング 188
 チェック・ペンディング 188
 バックアップ・ペンディング中 188
 ロード・ペンディング 188
身体障害 355
ステージング表
 従属即値 107
 伝搬 107
ストアード・プロシージャ
 トランスフォーマー 257
生成列
 インポート・ユーティリティの使用 36
 ロード・ユーティリティの使用 101
制約
 検査
 ロード操作の後の 103
セマンティクス
 FORCEIN、一般 323
 FORCEIN、コード・ページ 323
 FORCEIN、データ・タイプ 323
ゾーン 10 値ファイル・タイプ修飾子 113, 139
走査順序
 型付き表 29, 252
 デフォルト 252
 ユーザー指定 252

[タ行]

ダンプ・ファイル
 ロード・ユーティリティ 182
チュートリアル 359
 トラブルシューティングと問題判別 354
注文、DB2 ブックの 349
データ
 プラットフォーム間の移動 235

データウェアハウス・センター
 移動、データの 257
 概要 257
データ転送
 プラットフォーム間 235
 ホストおよびワークステーション間の 237
データベース
 ウェアハウス 257
 表からファイルへのエクスポート 9, 14
 ファイルから表へのインポート 41, 55
 ファイルから表へのロード 113
 リカバリー可能ロード・オプション 84
 リカバリー不能ロード・オプション 84
データベース移動ツール・コマンド 239
データベースの再配置コマンド 245
データ・タイプ
 PC/IXF 307
データ・タイプの説明
 ASC 285
 DEL ファイル・フォーマット 281
 PC/IXF 313
データ・レコード、PC/IXF 289
統合交換フォーマット (IXF) 287
特権
 インポート 31
 エクスポート 3
 LOAD 92
トラブルシューティング
 オンライン情報 354
 チュートリアル 354
トランスフォーマー
 ストアード・プロシージャ 257

[ハ行]

パーティション化データ
 ロード、データの 201
パーティション・キー
 ロード、データの 201
パーティション・データベース
 ロード制約事項 203
パーティション・データベース環境
 ロード、データの 222, 224
 ロード操作のモニター 209
バインド・ファイル
 エクスポート、インポート、およびロードによって使用 337
バッファ挿入
 インポート・ユーティリティ 33
パフォーマンス
 インポート 29

パフォーマンス (続き)
 ロード・ユーティリティ 189
パラメーター
 構文 261
表
 エクスポートされた、再作成 37
 状態 184
 ファイルからのロード 113
 ファイルのインポート 41, 55
 ファイルへのエクスポート 9, 14
 ロッキング 184
標識、レコード長 287
表スペース
 状態 184
表レコード、PC/IXF 289
表ロード削除開始ログ・レコード 184
ファイル・タイプ修飾子
 インポート API 55
 エクスポート API 14
 ロード API 139
 ロード・コマンド 113
 EXPORT ユーティリティ 9
 IMPORT コマンド 41
ファイル・フォーマット
 区切り付き ASCII (DEL) 278
 区切りなし ASCII (ASC) 283
 表からファイルへのエクスポート 9
 ファイルから表へのインポート 41
 ワークシート (WSF) 331
 PC バージョンの IXF (PC/IXF) 287
複製
 データウェアハウス・センター
 サポートされるタイプ 257
副表レコード、PC/IXF
 概要 289
文書
 表示 352
並列処理
 ロード・ユーティリティ 91
ヘッダー・レコード、PC/IXF 289
ヘルプ
 コマンド用
 呼び出し 351
 表示 352, 370
 メッセージ用
 呼び出し 350
 SQL ステートメント用
 呼び出し 352
変数
 構文 261
ペンディング 188
保全性検査 103

【マ行】

マテリアライズ照会表定義 (MQT)
 従属即値 106
 チェック・ペンディング状態 106
 リフレッシュ 106
マルチディメンション・クラスタリング (MDC)
 考慮事項 108
メッセージ
 概要 339
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 1, 29, 84
メッセージ・ヘルプ
 呼び出し 350
文字ストリング
 区切り文字 280
問題判別
 オンライン情報 354
 チュートリアル 354

【ヤ行】

ユーザー定義タイプ (UDT)
 特殊タイプ
 インポート 39
ユーティリティ
 ファイル・フォーマット 277
有効な PC/IXF データ・タイプ 307
呼び出し
 コマンド・ヘルプ 351
 メッセージ・ヘルプ 350
 SQL ステートメント・ヘルプ 352

【ラ行】

ラージ・オブジェクト (LOB) データ・タイプ
 インポート 38
 エクスポート 5
リカバリー可能データベース
 ロード・オプション 84
リカバリー不能データベース
 ロード・オプション 84
例
 ファイル
 ASC 284
 DEL 280
例外表
 ロード・ユーティリティ 182
レコード長標識 287
レコード・タイプ、PC/IXF
 アプリケーション 289
 階層 289
 継続 289
レコード・タイプ、PC/IXF (続き)
 索引 289
 識別 289
 終了 289
 データ 289
 表 289
 副表 289
 ヘッダー 289
 リスト 287
 列記述子 289
レジストリー変数
 DB2LOADREC 111
列
 値、無効な 318
 インポートの指定 55
 互換性のない 318
列記述子レコード、PC/IXF 289
ロード
 データ
 パーティション 201
 ファイルからデータベース表へ 113
 ファイル・タイプ修飾子 113
ロード API 139
ロード、データの
 パーティション 224
ロード、メッセージ・ファイル 1, 29, 84
ロード開始ログ・レコード 184
ロード削除開始補正ログ・レコード 184
ロード操作、データウェアハウス・センター 257
ロード・コピー・ロケーション・ファイル、ロールフォワードの使用 111
ロード・コマンド 113
 パーティション・データベース環境での 203, 222
ロード・ペンディング・リスト・ログ・レコード 184
ロード・ユーティリティ
 一時ファイル 113, 183
 インポート・ユーティリティとの比較 265
 概要 84
 コード・ページについての考慮事項 187
 構築フェーズ 84
 索引コピー・フェーズ 84
 削除フェーズ 84
 障害からのリカバリー 110
 使用に必要な権限と特権 92
 制限 93
 生成列 101
 制約事項 93
 ダンプ・ファイル 182
 データベースのリカバリー 84
 パフォーマンスの最適化 189
 表の状態 184

ロード・ユーティリティ (続き)
 表のロックング 184
 ファイル・タイプ修飾子 139
 ファイル・フォーマット 277
 プロセスの概要 84
 並列処理 91
 変更された構文と動作 84
 例外表 182
 ロード・フェーズ 84
 ログ・レコード 184
 DB2 Data Links Manager 232
 ID 列 99
ロールフォワード・ユーティリティ
 ロード・コピー・ロケーション・ファイル、使用 111
ログ・レコード
 ロード・ユーティリティ 184
ロックング
 インポート・ユーティリティ 40
 表のレベル 184

【ワ行】

ワークシート
 ファイル・フォーマット (WSF) 331

A

anyorder ファイル・タイプ修飾子 113, 139
API
 db2Load 139
 db2LoadQuery 164
 sqluxpr 14
 sqluimpr 55
ASC インポート・ファイル・タイプ 41
ASC のデータ・タイプの説明 285
ASC ファイル
 フォーマット 283
 例 284

B

binarynumerics ファイル・タイプ修飾子 113, 139

C

chardel ファイル・タイプ修飾子
 インポート 41, 55
 エクスポート 9, 14
 ロード 113, 139
code page ファイル・タイプ修飾子 113, 139

codel ファイル・タイプ修飾子
 インポート 41, 55
 エクスポート 9, 14
 ロード 113, 139
compound ファイル・タイプ修飾子 41, 55
CURSOR ファイル・タイプ
 データ移動 259

D

Data Links Manager
 移動、データの 230
dateformat ファイル・タイプ修飾子 41, 55, 113, 139
datesiso ファイル・タイプ修飾子 9, 14, 41, 55, 113, 139
DB2 Data Links Manager
 インスタンス間でのエクスポート 227
 インポート・ユーティリティ 231
 エクスポート・ユーティリティ 227
 ロード・ユーティリティ 232
DB2 インフォメーション・センター 360
 呼び出し 352
DB2 チュートリアル 359
DB2 ブック
 PDF ファイルの印刷 348
db2batch を使った並列エクスポート 6
db2Load API 139
db2LoadQuery API 164
DB2LOADREC レジストリー変数 111
db2move コマンド 239
db2relocatedb コマンド 245
decplusblank ファイル・タイプ修飾子 9, 14, 41, 55, 113, 139
decpt ファイル・タイプ修飾子 9, 14, 41, 55, 113, 139
DEL データ・タイプの説明 281
DEL ファイル
 フォーマット 278
 例 280
delprioritychar ファイル・タイプ修飾子 41, 55, 113, 139
ldel ファイル・タイプ修飾子 9, 14, 41, 55, 113, 139
dumpfile ファイル・タイプ修飾子 113, 139

E

EXPORT コマンド 9
EXPORT ユーティリティ
 エクスポートされた表の再作成 4
 概要 1
 使用に必要な権限と特権 3

EXPORT ユーティリティ (続き)
 制約事項 3
 ファイル・フォーマット 277
 ホストおよびワークステーション間で
 データを転送 237
 ラージ・オブジェクト (LOB) 5
 DB2 Data Links Manager 227
 db2batch を使った並列エクスポート
 6
 ID 列 4

F

fastparse ファイル・タイプ修飾子 113, 139
forcein ファイル・タイプ修飾子 41, 55, 113, 139, 323

G

generatedignore ファイル・タイプ修飾子 41, 55, 113, 139
generatedmissing ファイル・タイプ修飾子 41, 55, 113, 139
generatedoverride ファイル・タイプ修飾子 113, 139

H

HTML 文書
 更新 353

I

IBM リレーショナル・データ複製ツール
 概要 255
 コンポーネント 257
ID 列 4
 インポート・ユーティリティの使用 34
 ロード・ユーティリティの使用 99
ID 列でない生成列 36, 101
identityignore 41
identityignore ファイル・タイプ修飾子 55, 113, 139
identitymissing ファイル・タイプ修飾子 41, 55, 113, 139
identityoverride ファイル・タイプ修飾子 113, 139
implieddecimal ファイル・タイプ修飾子 41, 55, 113, 139
IMPORT コマンド 41
IMPORT ユーティリティ
 エクスポートされた表の再作成 37
 概要 29

IMPORT ユーティリティ (続き)
 クライアント/サーバー 33
 権限 31
 コード・ページについての考慮事項 78
 制限 31
 生成列 36
 制約事項 31
 特権 31
 バッファ挿入 33
 パフォーマンス 29
 パフォーマンスの最適化 29
 表のロック 40
 ファイル・フォーマット 277
 ホストおよびワークステーション間で
 データを転送 237
 ユーザー定義特殊タイプ (UDT) 39
 ラージ・オブジェクト (LOB) 38
 リモート・データベース 33
 ロード・ユーティリティとの比較 265
 DB2 Data Links Manager 231
 ID 列 34
indexfreespace ファイル・タイプ修飾子 113, 139
indexxf ファイル・タイプ修飾子 41, 55
indexschema ファイル・タイプ修飾子 41, 55

K

keepblanks ファイル・タイプ修飾子 41, 55, 113, 139

L

Load Query API 164
LOAD QUERY コマンド 137
 パーティション・データベース環境での 209
LOB (ラージ・オブジェクト) データ・タイプ
 インポート 38
 エクスポート 5
LOB ロケーション指定子 (LLS) 287
lobsinfile
 エクスポート API 14
lobsinfile ファイル・タイプ修飾子 9, 41, 55, 113, 139

N

nochecklengths ファイル・タイプ修飾子 41, 55, 113, 139

nodefaults ファイル・タイプ修飾子 41, 55
nodoublede1 ファイル・タイプ修飾子 9, 14, 41, 55, 113, 139
noeofchar ファイル・タイプ修飾子 41, 55, 113, 139
noheader ファイル・タイプ修飾子 113, 139
norowwarnings ファイル・タイプ修飾子 113, 139
notypeid ファイル・タイプ修飾子 41, 55
nullindchar ファイル・タイプ修飾子 41, 55, 113, 139

P

packeddecimal ファイル・タイプ修飾子 113, 139
pagefreespace ファイル・タイプ修飾子 113, 139
PC バージョンの IXF (PC/IXF) ファイル・フォーマット 287
PC/IXF
コード・ページ変換ファイル 318
データ・タイプ 313
有効 307
比較、System370 IXF との 330
無効
データ・タイプ 307, 318
列値 318
レコード・タイプ 289
列値、無効な 318
PC/IXF ファイル・インポート
規則 318, 320
データ・タイプ固有の規則 320
FORCEIN を使用する場合 323
PC/IXF ファイル・フォーマット
記述 287
プラットフォーム間のデータの移動 235

R

reclen ファイル・タイプ修飾子 41
インポート 55
ロード 113
ロード API 139

S

SELECT ステートメント
EXPORT コマンド 9
SQL ステートメント・ヘルプ
呼び出し 352
SQL メッセージ 339

SQLCODE
概要 339
SQLSTATE
概要 339
sqluexpr API 14
sqluimpr API 55
striptblanks ファイル・タイプ修飾子 41, 55, 113, 139
striptnulls ファイル・タイプ修飾子 41, 55, 113, 139
subtableconvert ファイル・タイプ修飾子 113
System370 IXF
比較、PC/IXF との 330
比較、System370 との 330

T

timeformat ファイル・タイプ修飾子 41, 55, 113, 139
timestampformat ファイル・タイプ修飾子 41, 55, 113, 139
totalfreespace ファイル・タイプ修飾子 113, 139

U

Unicode (UCS-2)
データ移動の考慮事項 333
usedefaults ファイル・タイプ修飾子 41, 55, 113, 139

W

WSF (ワークシート) ファイル・フォーマット
記述 331
プラットフォーム間のデータの移動 235

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
<http://www.ibm.com/software/data/db2/udb> から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。



Printed in Japan

SC88-9142-01



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