IBM DB2 Information Integrator

データ・ソース構成ガイド: 付録 BioRS ラッパーおよびライフ・サイエンス ユーザー定義関数

バージョン 8

IBM DB2 Information Integrator

データ・ソース構成ガイド: 付録 BioRS ラッパーおよびライフ・サイエンス ユーザー定義関数

バージョン 8

本書および本書で紹介する製品をご使用になる前に、 83 ページの『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： DB2 Information Integrator
Addendum to the Data Source Configuration Guide:
BioRS Wrapper and Life Sciences User-Defined Functions
Version 8

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2003.7

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

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

目次

第 1 章 BioRS データ・ソースへのアクセスの構成 1
BioRS とは何か 1
フェデレーテッド (連合)・システムへの BioRS の追加 3
BioRS ラッパーのカスタム関数の登録. . . . 3
BioRS ラッパーの登録 4
BioRS ラッパーの DB2_DJ_COMM プロファイル変数の設定 5
BioRS データ・ソースのサーバーの登録 . . . 6
BioRS データ・ソースのユーザー・マッピングの登録 6
BioRS データ・ソースのニックネームの登録 . 8
CREATE NICKNAME ステートメント - BioRS ラッパーの例 9
BioRS 列カーディナリティー統計. 11
BioRS ラッパー・パフォーマンスの最適化方法 13
カスタム関数と BioRS 照会. 14
BioRS ラッパーの等価結合述部 17
BioRS ラッパー - 照会例 19
BioRS 統計情報. 25
BioRS データ・バンク・カーディナリティー統計の確認 26
BioRS ニックネーム・カーディナリティー統計の更新 26
BioRS _ID_ 列カーディナリティーの更新 . . 27
BioRS AllText エレメント 28
ニックネーム変更時の注意 - BioRS ラッパー 29
カスタム関数表 - BioRS ラッパー 29
BioRS ラッパーのメッセージ 30
CREATE NICKNAME ステートメント構文 - BioRS ラッパー. 35
CREATE SERVER ステートメント・オプション - BioRS ラッパー 36
CREATE USER MAPPING ステートメント・オプション - BioRS ラッパー 38

第 2 章 ライフ・サイエンス・ユーザー定義関数 39
ライフ・サイエンス・ユーザー定義関数の概要 39
機能カテゴリーごとのライフ・サイエンス・ユーザー定義関数 39
ライフ・サイエンス・ユーザー定義関数の登録 40
ライフ・サイエンス・ユーザー定義関数の除去 41
逆変換ユーザー定義関数 42
LSPep2AmbNuc ユーザー定義関数. . . . 42
LSPep2AmbNuc ユーザー定義関数の例 . . 44
LSPep2AmbNuc ユーザー定義関数のエラー・メッセージ 45
LSPep2ProbNuc ユーザー定義関数. . . . 46
LSPep2ProbNuc ユーザー定義関数の例 . . 47
LSPep2ProbNuc ユーザー定義関数のエラー・メッセージ 48
定義行構文解析のユーザー定義関数 49
LSDeflineParse ユーザー定義関数 49
LSDeflineParse ユーザー定義関数の例 . . 52
一般化されたパターン・マッチングのユーザー定義関数 56
LSPatternMatch ユーザー定義関数 56
LSPatternMatch ユーザー定義関数の例 . . 56
LSPrositePattern ユーザー定義関数. . . . 58
LSPrositePattern ユーザー定義関数の例 . . 59
正規表現のサポート 60
GeneWise ユーザー定義関数. 60
GeneWise へのリンク 60
LSGeneWise ユーザー定義関数. 61
LSGeneWise ユーザー定義関数の例 . . . 62
Motifs ユーザー定義関数. 63
LSBarCode ユーザー定義関数 63
LSBarCode ユーザー定義関数の例. . . . 64
LSMultiMatch ユーザー定義関数 65
LSMultiMatch ユーザー定義関数の例 . . . 66
LSMultiMatch3 ユーザー定義関数 67
LSMultiMatch3 ユーザー定義関数の例 . . 67
反転ユーザー定義関数. 69
LSRevComp ユーザー定義関数 69
LSRevComp ユーザー定義関数の例 . . . 70

LSRevNuc ユーザー定義関数 71
LSRevNuc ユーザー定義関数の例 71
LSRevPep ユーザー定義関数 72
LSRevPep ユーザー定義関数の例 73
変換. 74
LSNuc2Pep ユーザー定義関数 74
LSNuc2Pep ユーザー定義関数の例. . . . 74
LSTransAllFrames ユーザー定義関数 . . . 76
LSTransAllFrames ユーザー定義関数 . . . 76
コドン度数表の形式 78
コドン度数表の例 78
変換表の形式. 79
変換表の例 80

アクセシビリティー 81
キーボードによる入力およびナビゲーション 81
アクセスしやすい表示. 81
フォントの設定 81
色覚への非依存 81
代替アラート・キュー. 82
支援テクノロジーとの互換性 82
入手可能な資料 82

特記事項 83
商標. 85

索引 87

IBM と連絡を取る. 89
製品情報 89

第 1 章 BioRS データ・ソースへのアクセスの構成

この章では、BioRS の概要、および BioRS データ・ソースをご使用のフェデレーテッド (連合)・システムに加える方法について解説し、BioRS ラッパーに関連するエラー・メッセージのリストを示します。

BioRS とは何か

BioRS は、ドイツ Biomax 社が開発した照会および検索システムです。BioRS を使用すると、フラット・ファイルやリレーショナル・データベースを含む、複数のデータ・ソースから情報を検索できます。通常 SwissProt や GenBank のような公用データは、フラット・ファイルとして BioRS システムにダウンロードします。BioRS は公用データ・ソースと専用データ・ソース (組織独自で保守する専用データベースなど) を共通環境に統合できます。

BioRS システムに統合されたデータ・ソースは、データ・バンク と呼ばれます。各データ・バンクのエントリーに含まれるエレメントは、まとめてスキーマ と呼ばれます。BioRS システムで索引付けされたデータ・バンクのエレメントのみを、BioRS 照会で使用できます。データ・バンクのエントリー同士を関連付け、BioRS システムでデータ・バンクを結合することも可能です。

BioRS データ・バンクには、親子関係を指定できます (ネスト可能)。親子関係では、子データ・バンクは PARENT と呼ばれる Reference データ・タイプ・エレメントを持ちます。PARENT エレメントは親データ・バンクの _ID_ エレメントを参照します。ネスト化されたデータ・バンクとネスト化されていないデータ・バンクが収容するデータの違いは、この定義済み PARENT エレメントのみです。

BioRS は BioRS データ・バンクのデータを照会できる Web ベースのインターフェースを提供します。BioRS ラッパーは、BioRS の Web ベースのインターフェースと同じアプリケーション・プログラミング・インターフェース (API) を使用して情報を照会します。

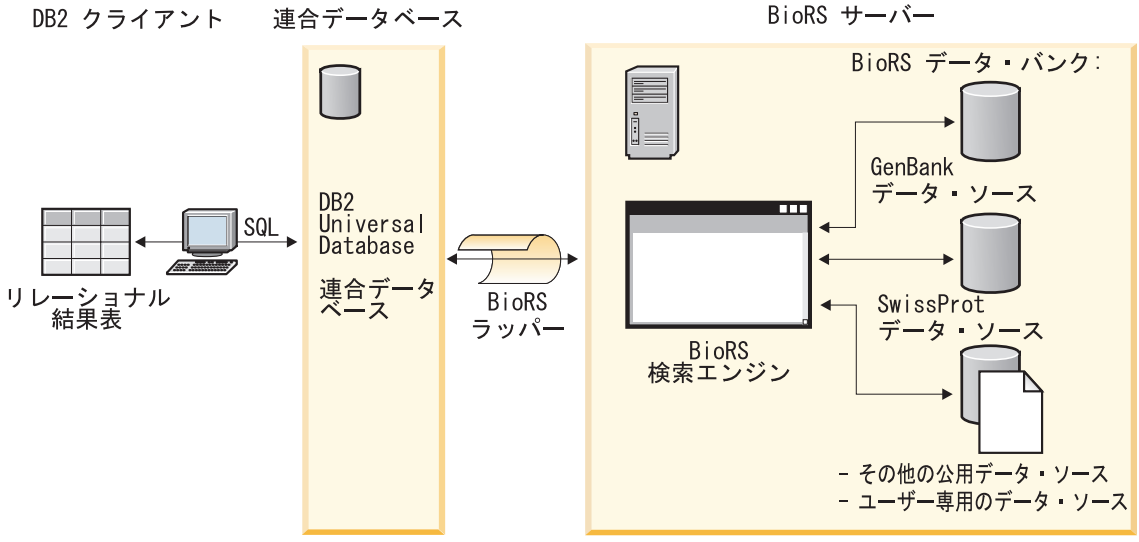


図 1. BioRS ラッパーが機能する構造

ユーザーまたはアプリケーションは、クライアントから SQL ステートメントを使用した照会をサブミットします。照会は BioRS ラッパーがインストールされたフェデレーテッド (連合)・システムに送信されます。照会の構成により、DB2® Universal Database と BioRS サーバーの両方が照会の処理に使用される場合があります。BioRS サーバーは、フェデレーテッド (連合)・システムとは異なるコンピューター上にある場合もあります。ただし、各照会ごとに、フェデレーテッド (連合)・システムから BioRS サーバーに認証情報を提供する必要があります。認証情報は、ユーザー ID とパスワードの組み合わせか、非認証標識 (通常はゲスト・アカウント) のいずれかです。

BioRS ラッパーは BioRS バージョン 5.0.14 と連動します。

BioRS 製品の詳細については、Biomax の Web サイト http://www.biomax.com を参照してください。

関連タスク:

- 3 ページの『フェデレーテッド (連合)・システムへの BioRS の追加』

関連情報:

- 19 ページの『BioRS ラッパー - 照会例』

フェデレーテッド (連合)・システムへの BioRS の追加

カスタム関数と BioRS ラッパーを登録すると、フェデレーテッド (連合)・サーバーで BioRS データ・ソースを使用できます。さらに、対応する BioRS サーバー、ユーザー・マッピング、およびニックネームを登録して、フェデレーテッド (連合)・サーバーで BioRS データを検索および処理できるようにします。

SQL ステートメントは、DB2 コントロール・センターまたは DB2 コマンド行プロセッサーから発行できます。BioRS をフェデレーテッド (連合)・システムに追加すると、BioRS データ・ソースに対して照会を実行できます。

手順:

BioRS データ・ソースをフェデレーテッド (連合)・サーバーに追加するには、以下のようにします。

1. CREATE FUNCTION ステートメントを使用してカスタム関数を登録します。
2. CREATE WRAPPER ステートメントを使用して BioRS ラッパーを登録します。
3. オプション: DB2_DJ_COMM 環境変数を設定して照会のパフォーマンスを向上させます。
4. CREATE SERVER ステートメントを使用して BioRS サーバーを登録します。
5. オプション: CREATE USER MAPPING ステートメントを使用して許可ユーザーを登録します。
6. CREATE NICKNAME ステートメントを使用してニックネームを登録します。
7. オプション: BioRS 列に対するカーディナリティー統計を更新します。

関連タスク:

- 3 ページの『BioRS ラッパーのカスタム関数の登録』
- 4 ページの『BioRS ラッパーの登録』
- 5 ページの『BioRS ラッパーの DB2_DJ_COMM プロファイル変数の設定』
- 6 ページの『BioRS データ・ソースのサーバーの登録』
- 6 ページの『BioRS データ・ソースのユーザー・マッピングの登録』
- 8 ページの『BioRS データ・ソースのニックネームの登録』
- 11 ページの『BioRS 列カーディナリティー統計』

BioRS ラッパーのカスタム関数の登録

BioRS ラッパーのカスタム関数の登録は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。カスタム関数の登録後、ラッパーを登録する必要があります。

カスタム関数の登録にサンプル・ファイル create_function_mappings.ddl を使用できます。サンプル・ファイルは、sqllib/samples/lifesci/biors ディレクトリーにあります。create_function_mappings.ddl ファイルは、各カスタム関数の定義を含みます。この DDL ファイルを実行して、BioRS ラッパーがインストールされている各 DB2 データベースにカスタム関数を登録できます。

前提条件:

- BioRS ラッパーのカスタム関数はすべて、BioRS というスキーマ名を使用して登録する必要があります。
- BioRS ラッパーがインストールされている各 DB2 データベースに対して、各カスタム関数を 1 回ずつ登録する必要があります。

手順:

カスタム関数を登録するには、キーワード AS TEMPLATE を使用して CREATE FUNCTION ステートメントを発行してください。

各関数の完全修飾名は、BioRS.<function-name> です。

以下の例では、CONTAINS 関数の 1 つのバージョンを登録します。

```
CREATE FUNCTION biors.contains (varchar(), varchar())
RETURNS INTEGER AS TEMPLATE;
```

この一連の作業における次のタスクは、適切な BioRS ラッパーの登録です。

関連タスク:

- 4 ページの『BioRS ラッパーの登録』

関連情報:

- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (ソースまたはテンプレート) ステートメント』
- 14 ページの『カスタム関数と BioRS 照会』
- 19 ページの『BioRS ラッパー - 照会例』
- 29 ページの『カスタム関数表 - BioRS ラッパー』

BioRS ラッパーの登録

BioRS ラッパーの登録は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。データ・ソースにアクセスするには、ラッパーを登録する必要があります。ラッパーは、フェデレーテッド (連合)・サーバーがデータ・ソースとの通信やデータ・ソースからのデータ検索に使用するメカニズムです。ラッパーは、ライブラリー・ファイルとしてシステムにインストールされています。 5 ページの表 1 に、デフォルトの BioRS ライブラリー・ファイルと各ファイルをサポートするオペレーティング・システムをリストします。

表 1. BioRS ライブラリー・ファイルと対応オペレーティング・システム

BioRS ライブラリー・ファイル	オペレーティング・システム
libdb2lsbiors.a	IBM AIX バージョン 4.3.3 以上
db2lsbiors.dll	Microsoft Windows NT バージョン 4 Microsoft Windows 2000 Microsoft Windows XP

手順:

BioRS ラッパーを登録するには、CREATE WRAPPER ステートメントを発行します。

例えば、AIX 上でデフォルト・ライブラリー・ファイル `libdb2lsbiors.a` から `wrap_biors` という BioRS ラッパーを登録するには、次のステートメントを発行します。

```
CREATE WRAPPER wrap_biors LIBRARY 'libdb2lsbiors.a';
```

関連タスク:

- 「*DB2 Information Integrator インストール・ガイド*」の『非リレーショナル・ラッパーおよびライフ・サイエンス・ユーザー定義関数ライブラリーの検査』
- 5 ページの『BioRS ラッパーの DB2_DJ_COMM プロファイル変数の設定』

関連情報:

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

BioRS ラッパーの DB2_DJ_COMM プロファイル変数の設定

BioRS ラッパーの DB2_DJ_COMM DB2 プロファイル変数の設定は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。オプションで、DB2_DJ_COMM DB2 プロファイル変数を設定し、BioRS データ・ソースにアクセスする際のパフォーマンスを向上できます。この変数は、初期設定時にフェデレーテッド (連合)・サーバーがラッパーをロードするかどうかを指定します。

データベースの始動時にフェデレーテッド (連合)・サーバーがラッパー・ライブラリーをロードすると、プロセッサーの使用量が増加します。過度の使用を避けるには、アクセスする必要のあるライブラリーのみを指定してください。

手順:

DB2_DJ_COMM DB2 プロファイル変数を設定するには、関連する CREATE WRAPPER ステートメントに指定したラッパーに対応するラッパー・ライブラリーを使用して **db2set** コマンドを発行します。

例えば、以下のようになります。

```
db2set DB2_DJ_COMM='libdb2lsbiors.a'
```

等号 (=) の両側にスペースを入れないようにしてください。

この一連の作業における次のタスクは、BioRS サーバーの登録です。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『環境変数およびプロファイル・レジストリー』

関連タスク:

- 6 ページの『BioRS データ・ソースのサーバーの登録』

関連情報:

- 「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』

BioRS データ・ソースのサーバーの登録

BioRS データ・ソースのサーバーの登録は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。ラッパーの登録後、対応するサーバーを登録する必要があります。

手順:

BioRS サーバーをフェデレーテッド (連合)・システムに登録するには、CREATE SERVER ステートメントを発行します。

例えば、以下のようになります。

```
CREATE SERVER brs_server WRAPPER wrap_biors OPTIONS(NODE 'biors_server2.com');
```

関連タスク:

- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 36 ページの『CREATE SERVER ステートメント・オプション - BioRS ラッパー』

BioRS データ・ソースのユーザー・マッピングの登録

ユーザー・マッピングの登録は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。アカウントへのアクセス方式または BioRS システムで使用している方式により、ユーザー・マッピングを作成する必要がない場合があります。

- BioRS サーバーですべてのユーザー・アカウントにゲスト・アクセスを設定している場合、DB2 Information Integrator にユーザー・マッピングを作成する必要はありません。
- BioRS サーバーでユーザー・アカウントの認証に ID とパスワードを使用している場合、BioRS ラッパーを使用するアカウントのユーザー・マッピングをフェデレーテッド (連合)・データベースに作成する必要があります。
- BioRS サーバーでゲストおよび認証ユーザー・アカウントの両方を使用している場合、BioRS ラッパーを使用するアカウントに対して、認証ユーザー・アカウントのユーザー・マッピングをフェデレーテッド (連合)・データベースに作成する必要があります。

ユーザー・マッピングは、BioRS ラッパーを使用して BioRS データ・ソースを照会するユーザーまたはアプリケーションのアクセスを認証します。ユーザーまたはアプリケーションが SQL 照会を登録済み BioRS ニックネームにサブミットし、そのユーザーまたはアプリケーションに対するユーザー・マッピングが未定義の場合、BioRS ラッパーはデフォルトのユーザー ID とパスワードを使用して、リモート BioRS サーバーからデータを取り出そうとします。照会対象のデータ・バンクで認証が必要な場合、エラー・メッセージが戻される可能性があります。

正しいユーザー ID とパスワードを確実に BioRS サーバーに渡すには、BioRS データ・ソースの検索を許可されたユーザーのユーザー・マッピングをフェデレーテッド (連合)・システムに作成します。ユーザー・マッピングを作成する際、パスワードは暗号化された形式でフェデレーテッド (連合)・データベースのシステム・カタログ表に格納されます。

手順:

BioRS ユーザー・マッピングを登録するには、CREATE USER MAPPING ステートメントを使用します。

例えば、以下に示す CREATE USER MAPPING ステートメントは、ユーザー Charlie を Biors_Server1 サーバーのユーザー Charlene にマップします。

```
CREATE USER MAPPING FOR Charlie SERVER Biors_Server1
OPTIONS(REMOTE_AUTHID 'Charlene', REMOTE_PASSWORD 'Charlene_pw');
```

独自のユーザー・マッピングを定義することもできます。以下の例の USER は、USER というユーザー名ではなく、現行のユーザーを示すキーワードです。

```
CREATE USER MAPPING FOR USER SERVER Biors_Server1
OPTIONS(REMOTE_AUTHID 'Yudong', REMOTE_PASSWORD 'Yudong_pw')
```

この一連の作業における次のタスクは、BioRS ラッパーのニックネームの登録です。

関連タスク:

- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 「*SQL リファレンス 第 2 巻*」の『CREATE USER MAPPING ステートメント』
- 38 ページの『CREATE USER MAPPING ステートメント・オプション - BioRS ラッパー』

BioRS データ・ソースのニックネームの登録

BioRS データ・ソースのニックネームの登録は、BioRS をフェデレーテッド (連合)・システムに追加する作業の一部です。サーバーの登録後、アクセスする各データ・ソースに対応するニックネームを登録する必要があります。ニックネームは、照会で BioRS データ・ソースを参照する場合に使用します。

重要: BioRS システムに統合されたデータ・ソースは、BioRS 内でデータ・バンク として参照されます。BioRS データ・バンクはフェデレーテッド (連合)・システム上のニックネームと同じです。

前提条件:

- BioRS データ・バンク名が DB2 フェデレーテッド (連合) 構文の規格に適合しない場合は、ニックネームの登録に REMOTE_OBJECT ニックネーム・オプションを使用する必要があります。
- BioRS エレメント名が DB2 フェデレーテッド (連合) 構文の規格に適合しない場合は、ニックネームの登録に ELEMENT_NAME 列オプションを使用する必要があります。

制約事項:

BioRS AllText エレメントをニックネームの 1 列目に使用しないでください。その他の列 (2 列目や 3 列目) では使用できます。

手順:

BioRS ニックネームを登録するには、CREATE NICKNAME ステートメントを使用します。

フェデレーテッド (連合)・ニックネームは BioRS データ・バンクと直接的に同等と見なされます。フェデレーテッド (連合)・ニックネームを作成する場合、ニックネーム列のリストを定義してください。指定したニックネーム列は、特定の BioRS データ・バンク形式のエレメントに対応する必要があります。BioRS では、エレメントに対して、使用可能な 5 つのデータ・タイプ (Text、Number、Date、Author、および Reference) を定義しています。これらのデータ・タイプは CLOB、CHAR、または VARCHAR フェデレーテッド (連合)・システム・データ・タイプにのみマップできます。

BioRS データ・バンクのニックネームを登録する最も簡単な方法は、BioRS データ・バンクと同じ名前のニックネームを付けることです。例えば、以下のようになります。

```
CREATE NICKNAME SwissProt
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR(128),
 ENTRYDATE VARCHAR (64))
FOR SERVER brs_server;
```

基本 BioRS データ・バンクである SwissProt がニックネームの名前です。

この単純な CREATE NICKNAME 構文を使用した場合、1 つの DB2 スキーマにつき 1 つのニックネーム・ファミリーのみを使用できます。REMOTE_OBJECT を指定すると、他の名前を使用できます。このニックネーム・オプションはニックネームに関連する BioRS オブジェクト・タイプの名前を指定します。ニックネームに対応するスキーマと BioRS データ・バンクは REMOTE_OBJECT オプションに指定された名前で決まります。REMOTE_OBJECT オプションはニックネーム間の関係も指定します。

以下の例は、前の例と同じニックネーム特性の設定を使用していますが、ニックネームを変え、REMOTE_OBJECT オプションを使用してニックネームを定義する BioRS データ・バンクを指定しています。

```
CREATE NICKNAME NewSP
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR(128),
 ENTRYDATE VARCHAR (64))
FOR SERVER brs_server
 OPTIONS (REMOTE_OBJECT 'SwissProt');
```

基本 BioRS データ・バンクは SwissProt、ニックネーム名は NewSP です。

関連概念:

- 25 ページの『BioRS 統計情報』

関連タスク:

- 26 ページの『BioRS ニックネーム・カーディナリティー統計の更新』

関連情報:

- 28 ページの『BioRS AllText エレメント』
- 9 ページの『CREATE NICKNAME ステートメント - BioRS ラッパーの例』
- 35 ページの『CREATE NICKNAME ステートメント構文 - BioRS ラッパー』
- 29 ページの『ニックネーム変更時の注意 - BioRS ラッパー』

CREATE NICKNAME ステートメント - BioRS ラッパーの例

このトピックでは、CREATE NICKNAME ステートメントを使用して BioRS ラッパーのニックネームを登録する方法を例示します。

例 1

以下の例は、DB2 Information Integrator 構文の規格に適合しないリモート BioRS データ・バンクのニックネームを作成する方法を示します。

```
CREATE NICKNAME SwissFT
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR (128),
 ENTRYDATE VARCHAR (64),
 FtLength VARCHAR (16),
 FOR SERVER biors1
 OPTIONS (REMOTE_OBJECT 'SwissProt.Features');
```

ニックネームの名前は SwissFT です。表の列は ID、ALLTEXT、ENTRYDATE、および FtLength です。ELEMENT_NAME 列オプションは ID 列に対して指定します。BioRS エレメントが、列名に対する有効な DB2 フェデレーテッド (連合) 構文の規格に適合しない場合、ELEMENT_NAME オプションを指定する必要があります。この例では、BioRS エレメント _ID_ は DB2 フェデレーテッド (連合) 構文に準拠しますが、_ID_ は DB2 Information Integrator ユーザーが混乱する可能性のある名前です。ID は簡潔でわかりやすい名前です。通常、以下のような場合に ELEMENT_NAME オプションを使用します。

- BioRS エレメント名が有効な DB2 フェデレーテッド (連合) 構文の規格に適合しない場合
- BioRS エレメント名の大文字小文字の区別が、設定した DB2 フェデレーテッド (連合)・システムの規格に適合しない場合
- BioRS エレメント名が DB2 Information Integrator ユーザーにとって不明白な場合

さらに、REMOTE_OBJECT オプションは、ニックネームが同じ BioRS データ・バンクの名前の指定にも利用します。BioRS データ・バンク名が、有効な DB2 フェデレーテッド (連合) 構文の規格に適合しない場合、REMOTE_OBJECT オプションを指定する必要があります。この例では、データ・バンク名「SwissProt.Features」は有効な DB2 フェデレーテッド (連合) 構文の規格に合いません。通常、以下のような場合に REMOTE_OBJECT オプションを使用します。

- BioRS データ・バンク名の大文字小文字の区別が、設定した DB2 フェデレーテッド (連合)・システムの規格に適合しない場合
- BioRS データ・バンク名が有効な DB2 フェデレーテッド (連合) 構文の規格に適合しない場合
- BioRS データ・バンク名が DB2 Information Integrator ユーザーにとって不明白な場合

例 2

以下の例は、他の BioRS データ・バンクにリンクする BioRS データ・バンクを使用する表のニックネームを作成する方法を示します。

```
CREATE NICKNAME SwissFT2
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR (1200),
 FtKey VARCHAR (32),
 FtLength VARCHAR (64),
   FtDescription VARCHAR (128),
   Parent VARCHAR (32) OPTIONS (REFERENCED_OBJECT 'SwissProt'))
 FOR SERVER biors1
 OPTIONS (REMOTE_OBJECT 'SwissProt.Features');
```

ニックネームの名前は SwissFT2 です。表列は ID、ALLTEXT、FtKey、FtLength、FtDescription、および Parent です。ELEMENT_NAME 列オプションは ID 列に対して指定します。REMOTE_OBJECT オプションは、ニックネームが対応する BioRS データ・バンクの名前の指定にも利用します。

さらに、Parent 列は REFERENCED_OBJECT オプションを使用します。このオプションは BioRS Reference データ・タイプ・エレメントに対応する列に指定する必要があります。REFERENCED_OBJECT オプションは、列が参照する BioRS データ・バンクの名前を指定します。この場合、Parent エレメントは BioRS SwissProt データ・バンクを参照します。

関連タスク:

- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 35 ページの『CREATE NICKNAME ステートメント構文 - BioRS ラッパー』
- 29 ページの『ニックネーム変更時の注意 - BioRS ラッパー』

BioRS 列カーディナリティー統計

フェデレーテッド (連合)・システムの BioRS 列カーディナリティー統計を更新するには、SYSSTAT.COLUMNS カタログ・ビューを変更する必要があります。

BioRS 列の正しいカーディナリティー統計を維持することで、オプティマイザーと BioRS ラッパーが照会処理中に最適なデータ・アクセス・プランを選択できます。

BioRS をフェデレーテッド (連合)・システムに追加する作業の一部として BioRS 列カーディナリティーをオプションで更新できます。BioRS データ・ソースへの照会のパフォーマンスを向上させたい場合にも BioRS 列カーディナリティーを更新します。

制約事項:

この手順は、BioRS _ID_ エレメントに対応する列のカーディナリティー統計の更新には使用しないでください。BioRS _ID_ エレメントに対応する列のカーディナリティー統計を更新する場合は、別の手順で行う必要があります。

手順:

BioRS 列カーディナリティー統計を更新するには、以下の構文を使用して UPDATE ステートメントを発行してください。

```
UPDATE sysstat.columns SET colcard=(SELECT COUNT(DISTINCT <column-name>)
                                    FROM <nickname-schema>.<nickname-name>)
WHERE
  tabschema=<nickname-schema>
  AND tabname=<nickname-name>
  AND colname=<column-name>
```

- *<column-name>* は、更新対象のカーディナリティー統計を持つ列の名前です。
- *<nickname-schema>* は、指定列を使用するニックネームに関連するスキーマの名前です。
- *<nickname-name>* は、指定列を使用するニックネームの名前です。

ニックネームに指定されたデータ・バンクのエントリーをすべて検索するため、照会の実行には数分かかる場合があります。

列に複数の値を含められる場合 (SwissProt データベース形式の PublicationYear エレメントなど)、SQL 参照を使用するには計算が複雑になり過ぎます。このような列には、カーディナリティー値を手動で計算後、SYSSTAT.COLUMNS カタログ・ビューを更新する必要があります。カーディナリティー値を計算するには、列の別個の値を、行ごとの値の平均値で割ります。計算したカーディナリティー値は、表のカーディナリティーより小さくなります。

例:

3 つの行にニックネームがあるとします。それぞれの行に対する PublicationYear 列の値は、以下のとおりです。

- 1997 1992 1985
- 1997 1992 1982
- 1992 1991 1990 1976 1974 1971

9 つの異なる値があり、行内の値の平均数は 4 です。PublicationYear 列のカーディナリティーは 9/4 または 3 (2.25 は次に大きい整数に切り上げ) です。カーディナリティーの計算後、SYSSTAT.COLUMNS カタログ・ビューを次の UPDATE ステートメントを使用して更新します。

```
UPDATE sysstat.columns SET colcard=3
WHERE
  tabschema=<nickname-schema>
  AND tabname=<nickname-name>
  AND colname=<column-name>
```

- *3* は列のカーディナリティー値です。

- <*nickname-schema*> は、指定列を使用する基本ニックネームに関連するスキーマの名前です。
- <*nickname-name*> は、指定列を使用する基本ニックネームの名前です。
- <*column-name*> は、更新対象のカーディナリティー統計を持つ列の名前です。

関連概念:

- 25 ページの『BioRS 統計情報』

関連タスク:

- 26 ページの『BioRS ニックネーム・カーディナリティー統計の更新』
- 27 ページの『BioRS _ID_ 列カーディナリティーの更新』

BioRS ラッパー・パフォーマンスの最適化方法

このトピックでは、BioRS ラッパーを使用した場合の照会のパフォーマンスを最適化する方法を示します。

検索エンジン間の転送データ量を最小化する。

フェデレーテッド (連合) 環境は 2 つの照会エンジンを使用します。BioRS ラッパーの場合、照会エンジンは DB2® Universal Database と BioRS です。DB2 エンジンはニックネーム列に指定された述部 (=、BETWEEN、LIKE、<> などの関係演算子) を処理します。BioRS エンジンは、BioRS ラッパーの 4 つのカスタム関数を使用して指定された述部を処理します。

2 つの検索エンジン間で転送するデータ量を最小化するには、可能な場合は常にデータ処理を BioRS システムにプッシュダウンするよう照会を構成します。

照会で結合操作を行う必要がある場合、可能な場合は BioRS データ・バンクの既存の親子関係を活用して等価結合操作を行います。等価結合操作は BioRS で処理され、ここでも DB2 と BioRS 照会エンジン間のデータ転送量を最小化できます。

重要: DB2 Information Integrator から BioRS への照会を中断しないでください (コマンド行プロセッサーで **Ctrl-D** または **Ctrl-Z** を使用する、あるいはアプリケーション・プログラムを停止する、など)。照会を中断すると、BioRS サーバーで実行中の処理がデッド状態になります。「デッド」プロセスは、BioRS と DB2 Information Integrator 両方のシステム・パフォーマンスを急速に低下させます。また、「デッド」プロセスの数が増えると、DB2 Information Integrator 照会処理中に予期しないエラーが発生する恐れがあります。例えば、行があると予想される場合に、有効な照会が 0 行を戻す、などです。極端な状態では、BioRS、DB2 Information Integrator のいずれかまたは両方が停止または異常終了する恐れがあります。

フェデレーテッド (連合) 環境で BioRS 統計情報を保守する。

フェデレーテッド (連合)・システムでは、フェデレーテッド (連合)・データベ

ースは、ニックネームが付けられたオブジェクトのカタログ統計に基づいて、照会処理を最適化します。BioRS データ・ソースに関する現行統計の保守は、BioRS ラッパーのパフォーマンス最適化に不可欠です。ニックネームが定義されたリモート・オブジェクトの統計データまたは構造の特性を変更した場合、フェデレーテッド (連合)・システムで対応するニックネーム列カーディナリティー統計を更新する必要があります。

BioRS ラッパーのパフォーマンスを最適化するには、これらの更新を DB2 Information Integrator で定期的に行う必要があります。

関連概念:

- 「*連合システム・ガイド*」の『照会処理のチューニング』
- 17 ページの『BioRS ラッパーの等価結合述部』
- 25 ページの『BioRS 統計情報』

関連情報:

- 14 ページの『カスタム関数と BioRS 照会』
- 19 ページの『BioRS ラッパー - 照会例』

カスタム関数と BioRS 照会

フェデレーテッド (連合) 環境は 2 つの照会エンジンを使用します。BioRS ラッパーでの照会エンジンは DB2 Universal Database と BioRS です。以下に挙げる 4 つの BioRS カスタム関数を使用して、述部を BioRS エンジンにプッシュ・ダウンすることを指定できます。

- BIORS.CONTAINS
- BIORS.CONTAINS_LE
- BIORS.CONTAINS_GE
- BIORS.SEARCH_TERM

4 つのカスタム関数は BioRS スキーマに登録されます。BioRS スキーマを使用して関数を参照してください。

カスタム関数 BIORS.CONTAINS、BIORS.CONTAINS_LE、および BIORS.CONTAINS_GES には検索語列引き数と照会テキスト引き数が必要です。以下に、BIORS.CONTAINS ステートメントの例を示します。

```
BIORS.CONTAINS (<search term column>,<query term>)
```

検索語列引き数値は、索引付けされた BioRS 列を参照します。索引付けされていない列を使用すると、エラー・メッセージ SQL30090N (「操作がアプリケーション実行環境で無効です。」) が表示されます。

照会語引き数の値は、リテラル、ホスト変数、または列参照のいずれかになります。演算またはストリング連結は使用できません。また、使用する検索語列が NULL 値を許可するよう定義済みでも、照会語引き数値は NULL を取れません。

照会語引き数では、大文字小文字は区別されません。

照会語引き数で有効なデータ・タイプと形式は、使用される検索語の BioRS データ・タイプによって決まります。BioRS では、使用可能な 5 つのデータ・タイプ (Text、Author、Date、Number、および Reference) を定義しています。BioRS データ・タイプおよびデータ・タイプごとの有効な関数照会語を、表 2 にリストします。

表 2. BioRS データ・タイプと有効なカスタム関数照会語

検索語列のデータ・タイプ	有効な照会語	形式
Text	VARCHAR() または CHAR()	BioRS テキスト用語 (ワイルドカードを含む)。
Author	VARCHAR() または CHAR()	″<last>, <init>″ 書式の BioRS 作成者参照。 ″<last>″ は作成者の姓です。″<init>″ は作成者のイニシャルですが、ピリオドはつけません。コンマとイニシャルの間には空白文字を挿入できます。 また、コンマやイニシャルを含めずに <last> だけを指定することもできます。
Date	VARCHAR()、 CHAR()、 DATE、または TIMESTAMP	文字ストリングでは、DB2 形式の日付、yyyy/mm/dd。
Number	VARCHAR() または CHAR()、 INTEGER、SMALLINT、 BIGINT REAL、DOUBLE、 DECIMAL	DB2 形式の数字。
Reference	VARCHAR() または CHAR()	BioRS テキスト用語。

BioRS データ・タイプ検索語列と照会語引き数などのその他の組み合わせを使用すると、エラー・メッセージ SQL30090N (「操作がアプリケーション実行環境で無効です。」) が表示されます。表 2 に示した組み合わせでのみ使用できます。

Text、Author、および Reference データ・タイプ検索語列の照会語引き数は、BioRS 照会言語パターンと一致する必要があります。BioRS では、英数字とワイルドカードで照会語引き数を作成できます。BIORS.CONTAINS は、疑問符 (**?**) とアスタリスク (*****) の 2 つのワイルドカードをサポートします。

? ワイルドカードは英数字 1 文字に相当します。例えば、述部 `BioRS.CONTAINS (description, 'bacteri?')=1` は用語 bacteria (バクテリア) と一致しますが、bacterial (バクテリアの) とは一致しません。

***** ワイルドカードは、ゼロまたは複数の文字に相当します。例えば、述部 `BioRS.CONTAINS (description, 'bacteri*')=1` は用語 bacteri (バクテリ)、bacteria (バクテリア)、および bacterial (バクテリアの) と一致します。

BioRS 照会言語パターンの詳細については、ご使用の BioRS の資料を参照してください。

BIORS.CONTAINS 関数はすべての BioRS 列型に指定できます。

BIORS.CONTAINS_GE と BIORS.CONTAINS_LE カスタム関数は基本 BioRS データ・タイプが Number もしくは Date である列にのみ指定できます。BIORS.CONTAINS_GE 関数は、照会語引き数が表す値より大きいか等しい値が入っている列の行を選択します。BIORS.CONTAINS_LE 関数は、照会語引き数が表す値より小さいか等しい値が入っている列の行を選択します。

BIORS.CONTAINS、BIORS.CONTAINS_GE、および BIORS.CONTAINS_LE 関数は整数の結果を戻します。3 つの CONTAINS 関数のいずれかを述部で使用する場合、戻り値は **=** または **<>** 演算子を使用して値 1 と比較されます。例えば、以下のようになります。

```
SELECT * FROM s.MySP WHERE BIORS.CONTAINS (s.AllText, 'muscus') = 1;
```

表現形式 `NOT (BioRS.Contains (col,value) = 1)` は、`BioRS.CONTAINS (col,value) <> 1` と同じです。

BIORS.SEARCH_TERM 関数を実行すると、別の方法では実行が難しい照会を実行できます。BioRS 形式を使用した検索語の指定にこの関数を使用できます。BIORS.SEARCH_TERM 関数には 2 つの引き数が必要です。最初の引き数は、用語が適用されるニックネームの _ID_ 列への参照です。2 番目の引き数は、データ・バンク名を持たない用語を含む文字ストリングです。

以下の例では、SeqLength エレメントが 100 以上の値を含む MyEMBL データ・バンクのエントリーに対するすべての列を選択しています。

```
SELECT * FROM MyEMBL s WHERE
 BIORS.SEARCH_TERM (s.ID, '[SeqLength GREATER number:100;]') = 1;
```

以下の例では、Swiss ニックネームから MolWeight エレメントの値が 100368 以上の MolWeight 列を選択しています。

```
SELECT s.molweight FROM Swiss s WHERE
   BIORS.SEARCH_TERM (s.ID, '[MolWeight GREATER number:100368;]') = 1;
```

BIORS.SEARCH_TERM 関数を指定した場合、その他のカスタム関数は照会に使用できません。ただし、BIORS.CONTAINS、BIORS.CONTAINS_GE、および BIORS.CONTAINS_LE 関数は、任意の組み合わせで同一の照会に使用できます。

関連概念:

- 「*連合システム・ガイド*」の『プッシュダウン分析』
- 13 ページの『BioRS ラッパー・パフォーマンスの最適化方法』
- 17 ページの『BioRS ラッパーの等価結合述部』

関連タスク:

- 3 ページの『BioRS ラッパーのカスタム関数の登録』

関連情報:

- 19 ページの『BioRS ラッパー - 照会例』
- 29 ページの『カスタム関数表 - BioRS ラッパー』

BioRS ラッパーの等価結合述部

4 つの BioRS カスタム関数を使用して BioRS エンジンに対する述部を指定できます。ただし、例外が 1 つあります。それは、照会中に等価結合操作を行う場合です。*結合操作*では、一致する列の値を基に複数の表からデータを取り出します。*等価結合* は、式 = 式形式の結合条件をとる結合操作です。BioRS 照会では、等価結合条件は片方のデータ・バンクの _ID_ エレメントともう一方のデータ・バンクの Reference タイプ・エレメントを含む必要があります。

例:

ここでは、ニックネーム定義のサンプルとサンプルのニックネームを使用した等価結合照会を例示します。

2 つの BioRS データ・バンク、SwissProt と SwissProt.features を照会すると仮定します。SwissProt.features データ・バンクは SwissProt データ・バンクの子で、Parent と呼ばれるエレメントを含みます。Parent エレメントは SwissProt の _ID_ エレメントが示すエントリーへの参照を含みます。2 つのデータ・バンクに対して 2 つのニックネーム定義を登録します。

ニックネーム定義 1 は以下のとおりです。

```
CREATE NICKNAME tc600sprot (
   ID                VARCHAR (32) OPTIONS (ELEMENT_NAME '_ID_'),
   AllText           VARCHAR (128),
   EntryDate         VARCHAR (128),
   Update            VARCHAR (128),
   Description       VARCHAR (1200),
   Crossreference    VARCHAR (32),
   Authors           VARCHAR (256),
```

```
   Journal           VARCHAR (256),
   JournalIssue      VARCHAR (64) OPTIONS (IS_INDEXED 'N'),
   PublicationYear   VARCHAR (1024),
   Gene              VARCHAR (20) OPTIONS (IS_INDEXED 'Y'),
   Remarks           VARCHAR (1200),
   RemarkType        CHAR    (20),
   CatalyticActivity VARCHAR (20),
   CoFactor          VARCHAR (64),
   Disease           VARCHAR (128),
   Function          VARCHAR (128),
   Pathway           VARCHAR (128),
   Similarity        VARCHAR (128),
   Complex           VARCHAR (64),
   FtKey             VARCHAR (32),
   FtDescription     VARCHAR (128),
   FtLength          VARCHAR (256),
   MolWeight         VARCHAR (64),
   ProteinLen        VARCHAR (32) OPTIONS (ELEMENT_NAME 'Protein_length'),
   Sequence          CLOB,
   AccNumber         VARCHAR (32),
   Taxonomy          VARCHAR (128),
   Organelle         VARCHAR (128),
   Organism          VARCHAR (128),
   Keywords          VARCHAR (1200),
   Localization      VARCHAR (128),
   FtKey_count       VARCHAR (32)) FOR SERVER biors_server_600
      OPTIONS (REMOTE_OBJECT 'SwissProt');
```

ニックネーム定義 2 は以下のとおりです。

```
CREATE NICKNAME tc600feat (
  ID     VARCHAR (32) OPTIONS (ELEMENT_NAME '_ID_'),
  AllText VARCHAR (1200),
  FtKey   VARCHAR (32),
  FtLength VARCHAR (64),
  FtDescription VARCHAR (128),
  Parent VARCHAR (32) OPTIONS (REFERENCED_OBJECT 'SwissProt'))
    FOR SERVER biors_server_600 OPTIONS (REMOTE_OBJECT 'SwissProt.features');
```

以下の照会は、等価結合にて両方のニックネームを参照します。

```
SELECT s.ID, f.ID, f.FtKey FROM tc600sprot s, tc600feat f
   WHERE BioRS.CONTAINS (s.AllText, 'anopheles') = 1
    AND BioRS.CONTAINS (s.PublicationYear, 1997) = 1
 AND BioRS.CONTAINS (f.FtKey, 'signal') = 1
 AND f.Parent = s.ID;
```

前述した照会では、2 つの述部が tc600sprot ニックネーム (SwissProt データ・バンク) に適用されます。2 つの述部は用語 anopheles を含み、発行年が 1997 年である行をフィルターに掛けます。1 つの述部が tc600feat ニックネーム (SwissProt.features データ・バンク) に適用され、用語 signal を含む FtKey エレメントのある行をフィルターに掛けます。2 つのニックネームは条件 f.Parent = s.ID. を使用して結合されます。

最終結果セットには、これらの基準に適合し、フィーチャー・エントリーが SwissProt データ・バンク内の一致するエントリーを参照する行のみが含まれます。

関連概念:

- 13 ページの『BioRS ラッパー・パフォーマンスの最適化方法』

関連情報:

- 14 ページの『カスタム関数と BioRS 照会』
- 19 ページの『BioRS ラッパー - 照会例』

BioRS ラッパー - 照会例

このトピックでは、ニックネーム swiss と swissft を使用した照会のサンプルをいくつか紹介します。

ニックネーム swiss は、次の CREATE NICKNAME ステートメントを使用して登録されています。

```
CREATE NICKNAME swiss
  (
  ID                 CHAR (30) OPTIONS (ELEMENT_NAME '_ID_'),
  EntryDate          VARCHAR (15),
  Update             CLOB (15),
  Description        CLOB (15),
  Crossreference     CLOB (15),
  Authors            CLOB (15),
  Journal            VARCHAR (15),
  JournalIssue       VARCHAR (15),
  PublicationYear    CLOB (15),
  PublicationTitle   CLOB (15),
  Gene               CLOB (15),
  Remarks            CLOB (15),
  RemarkType         VARCHAR (15),
  CatalyticActivity  VARCHAR (15),
  CoFactor           VARCHAR (15),
  Disease            VARCHAR (15),
  Function           CLOB (15),
  Pathway            VARCHAR (15),
  Similarity         CLOB (15),
  Complex            VARCHAR (15),
  FtKey              VARCHAR (15),
  FtDescription      CLOB (15),
  FtLength           VARCHAR (15),
  MolWeight          CHAR (15),
  Protein_Length     VARCHAR (15),
  Sequence           CLOB (15),
  AccNumber          VARCHAR (15),
  Taxonomy           CLOB (15),
  Organelle          VARCHAR (15),
  Organism           VARCHAR (15),
  Keywords           VARCHAR (15),
```

```
  Localization       VARCHAR (15),
  FtKey_count        VARCHAR (15),
  AllText            CLOB (15)
  )
    FOR SERVER biors_server
      OPTIONS (REMOTE_OBJECT 'swissprot');
```

ニックネーム swissft は、次の CREATE NICKNAME ステートメントを使用して登録されています。

```
CREATE NICKNAME swissft
  (
  ID                VARCHAR (30) OPTIONS (ELEMENT_NAME '_ID_'),
  FtKey             VARCHAR (15),
  FtLength          VARCHAR (15),
  FtDescription     VARCHAR (15),
  Parent            VARCHAR (30) OPTIONS (REFERENCED_OBJECT 'swissprot'),
  AllText           CLOB (15)
  )
    FOR SERVER biors_server
      OPTIONS (REMOTE_OBJECT 'swissprot.features');
```

表 3 の照会と結果は、照会を構造化し、フェデレーテッド (連合)・システムと BioRS サーバー間でワークロードを最適化する方法を示します。

表 3. 異なる照会で同じ結果が得られる例

照会	結果
select s.id from Swiss s where biors.CONTAINS(s.id, '100K_RAT') = 1 fetch first 3 rows only	ID --------------- 100K_RAT 1 record(s) selected.
select s.id from Swiss s where s.id LIKE '%100K_RAT%' fetch first 3 rows only	ID --------------- 100K_RAT 1 record(s) selected.

表 3 のいずれの照会も同じ結果となります。ただし、1 番目の照会は、2 番目の照会と比べかなり短い時間で実行されます。最初の照会は入力述部の指定に BIORS.CONTAINS 関数を使用しています。その結果、BioRS は swissprot データ・バンクのデータを検索し、選択したデータを DB2 Information Integrator に渡します。2 番目の照会では、LIKE 入力述部が Swiss ニックネームに直接指定されています。そのため、BioRS は swissprot データ・バンク全体を DB2 Information Integrator に転送します。データ・バンク・コンテンツの転送後、DB2 Information Integrator がデータを選択します。

表 4 に、BIORS.CONTAINS 関数にワイルドカード文字を使用した照会と結果の例を示します。異なるワイルドカード文字を使用していますが、表 4 の照会結果はすべて同じになります。

表 4. BIORS.CONTAINS 関数にワイルドカードを使用した照会の例

照会	結果
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, 'MEDLINE') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 record(s) selected.
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, '?ED?IN?') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 record(s) selected.
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, '*D*N*') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 record(s) selected

表 5 に、BIORS.CONTAINS 関数を使用して BioRS Author データ・タイプ・エレメントの情報にアクセスする方法を示します。

表 5 のすべての照会構文はほとんど同じです。照会語の先頭のイニシャルの有無、および、ファーストネームと最後のイニシャルの間のスペースの数のみが異なります。

表 5. BioRS Author データ・タイプ列にアクセスする照会の例

照会	結果
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller') = 1 fetch first 3 rows only	AUTHORS --------------- Mueller D. Rehb Mayer K.F.X. Sc Zemmour J. Litt 3 record(s) selected.

表 5. BioRS Author データ・タイプ列にアクセスする照会の例 (続き)

照会	結果
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller,D') = 1 fetch first 3 rows only	AUTHORS ---------- 0 record(s) selected.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller ,D') = 1 fetch first 3 rows only	AUTHORS ---------- 0 record(s) selected.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller, D') = 1 fetch first 3 rows only	AUTHORS ---------- Mueller D. Rehb Zou P.J. Borovo Davies J.D. Mue 3 record(s) selected.

表 6 の照会と結果は、BIORS.CONTAINS 関数を使用して BioRS Date タイプ・エレメントの情報にアクセスする方法を示します。

BioRS Date タイプのフィールドがデータのシーケンスを含む場合、表 6 の 2 例目に見られるように、結果に余分な情報が含まれます。BioRS Numeric データ・タイプ・エレメント (Date と Number) には、複数の値を含めることができます。そのため、BioRS Date または Number エレメントに対して実行した照会結果も複数の値を含みます。値は常にスペースで区切ります。

表 6. BioRS Date データ・タイプ列にアクセスする照会の例

照会	結果
select e.entrydate from embl e where biors.CONTAINS(e.entrydate, date('11/01/1997')) = 1 fetch first 3 rows only	ENTRYDATE ---------- 01-NOV-1997 01-NOV-1997 01-NOV-1997 3 record(s) selected.
select g.update from gen g where biors.CONTAINS(g.update, date('11/01/1997')) = 1 fetch first 3 rows only	UPDATE ---------- 01-NOV-1997 11- 01-NOV-1997 12- 01-NOV-1997 06- 3 record(s) selected.

表 7 の照会と結果は、BIORS.CONTAINS_LE および BIORS.CONTAINS_GE 関数の使用方法を示します。

表 7. BIORS.CONTAINS_LE および BIORS.CONTAINS_GE 関数を使用した照会の例

照会	結果
select s.molweight from Swiss s where biors.CONTAINS_LE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT --------------- 100368 10576 8523 3 record(s) selected.
select s.molweight from Swiss s where biors.CONTAINS_GE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT --------------- 100368 103625 132801 3 record(s) selected.
select s.journalissue from Swiss s where biors.CONTAINS_GE(s.journalissue, 172) = 1 fetch first 3 rows only	JOURNALISSUE --------------- 172 21 242 196 3 record(s) selected.

表 8 の照会と結果は、BioRS 形式を使用した検索語の指定に BIORS.SEARCH_TERM を使用する方法を示します。

表 8. BIORS.SEARCH_TERM 関数を使用した照会の例

照会	結果
select s.publicationyear from Swiss s where biors.SEARCH_TERM (s.id, '[PublicationYear EQ number:1997;]')=1 fetch first 10 rows only	PUBLICATIONYEAR --------------- 1997 1997 2000 1988 1991 1997 1994 1997 1997 1998 1994 1995 1997 1997 1999 1997 1994 1994 1995 1993 1992 1997 10 record(s) selected.

表 8. BIORS.SEARCH_TERM 関数を使用した照会の例 (続き)

照会	結果
select s.molweight from Swiss s where biors.SEARCH_TERM (s.id, '[MolWeight EQ number:100368;]') = 1 fetch first 10 rows only	MOLWEIGHT --------------- 100368 100368 2 record(s) selected.
select s.molweight from Swiss s where biors.SEARCH_TERM (s.id, '[MolWeight GREATER number:100368;]') = 1 fetch first 10 rows only	MOLWEIGHT --------------- 100368 103625 132801 194328 130277 287022 289130 135502 112715 112599 10 record(s) selected.

以下の照会は、関係述部を使用して、親子関係のある 2 つのデータ・バンク間で等価結合を構成する方法を示します。

```
select s.id, f.id, f.parent from Swiss s, Swissft f
where (f.parent = s.id) fetch first 10 rows only
```

照会結果は以下のようになります。

```
ID                   ID                  PARENT
-------------------- ------------------- ------------------------------
100K_RAT             100K_RAT.1          swissprot:100K_RAT
100K_RAT             100K_RAT.2          swissprot:100K_RAT
100K_RAT             100K_RAT.3          swissprot:100K_RAT
100K_RAT             100K_RAT.4          swissprot:100K_RAT
100K_RAT             100K_RAT.5          swissprot:100K_RAT
100K_RAT             100K_RAT.6          swissprot:100K_RAT
100K_RAT             100K_RAT.7          swissprot:100K_RAT
100K_RAT             100K_RAT.8          swissprot:100K_RAT
100K_RAT             100K_RAT.9          swissprot:100K_RAT
104K_THEPA           104K_THEPA.1        swissprot:104K_THEPA

  10 record(s) selected.
```

上記の照会結果では、100K_RAT レコードが 9 つの子レコードの親になります (100K_RAT.1 から 100K_RAT.9)。

関連概念:

- 13 ページの『BioRS ラッパー・パフォーマンスの最適化方法』
- 17 ページの『BioRS ラッパーの等価結合述部』

関連情報:

- 14 ページの『カスタム関数と BioRS 照会』
- 9 ページの『CREATE NICKNAME ステートメント - BioRS ラッパーの例』
- 35 ページの『CREATE NICKNAME ステートメント構文 - BioRS ラッパー』

BioRS 統計情報

フェデレーテッド (連合)・システムでは、フェデレーテッド (連合)・データベースは、ニックネームが付けられたオブジェクトのカタログ統計に基づいて、照会処理を最適化します。これらの統計は、CREATE NICKNAME ステートメントを使用したニックネーム作成時に BioRS データ・ソースから抽出されます。フェデレーテッド (連合)・データベースはデータ・ソース側にオブジェクトが存在するかチェックし、次に既存のデータ・ソース統計データを収集します。情報はデータ・ソース・カタログから読み取られ、フェデレーテッド (連合)・サーバー上の DB2® フェデレーテッド (連合)・データベース・システム・カタログに置かれます。

BioRS データ・ソースでは、主要な統計情報は以下のものを含みます。

- ニックネームのカーディナリティー。BioRS データ・ソースでは、ニックネーム・カーディナリティーは対応する BioRS データ・バンクのエントリー数と同じです。
- BioRS _ID_ エレメントに対応する列のカーディナリティー。この列のカーディナリティーは、列を参照するニックネームのカーディナリティーと一致する必要があります。
- BioRS ラッパーが使用する必要があるすべての列のカーディナリティー。

BioRS データ・ソースに関する現行統計を保守し、BioRS ラッパーのパフォーマンスを最適化してください。ニックネームが定義されたリモート・オブジェクトの統計データまたは構造の特性を変更した場合、フェデレーテッド (連合)・システムで対応するカーディナリティー統計を更新する必要があります。カーディナリティー統計は SYSSTAT.TABLES カタログ・ビューおよび SYSSTAT.COLUMNS カタログ・ビューに保管されます。

フェデレーテッド (連合)・システムの BioRS カーディナリティー統計を保守するには、以下のタスクを行ってください。

1. 必要に応じて、必要なニックネームのカーディナリティー統計を確認する。
2. 必要な 1 つまたは複数のカタログ・ビューの適切なカーディナリティー統計を更新する。

関連概念:

- 「*連合システム・ガイド*」の『照会処理のチューニング』

関連タスク:

- 26 ページの『BioRS データ・バンク・カーディナリティー統計の確認』
- 26 ページの『BioRS ニックネーム・カーディナリティー統計の更新』
- 11 ページの『BioRS 列カーディナリティー統計』
- 27 ページの『BioRS _ID_ 列カーディナリティーの更新』

BioRS データ・バンク・カーディナリティー統計の確認

ニックネーム統計を更新、または、BioRS _ID_ エレメントに対応する列のカーディナリティーを更新する前に BioRS データ・バンク・カーディナリティー統計を確定する必要があります。

手順:

BioRS 内の特定のデータ・バンクのカーディナリティー統計を確認するには、BioRS ユーティリティー・プログラム admin_find または www_find.cgi を使用します。-c (カーディナリティー) オプションを指定してください。これら 2 つの BioRS ユーティリティー・プログラムの詳細については、ご使用の BioRS の資料を参照してください。

関連概念:

- 25 ページの『BioRS 統計情報』

関連タスク:

- 26 ページの『BioRS ニックネーム・カーディナリティー統計の更新』
- 11 ページの『BioRS 列カーディナリティー統計』
- 27 ページの『BioRS _ID_ 列カーディナリティーの更新』

BioRS ニックネーム・カーディナリティー統計の更新

ニックネームを作成する BioRS データ・バンクのコンテンツを大幅に変更する場合、BioRS ニックネーム・カーディナリティー統計を更新する必要があります。ニックネームの正しいカーディナリティー統計を維持することで、オプティマイザーと BioRS ラッパーが最適なデータ・アクセス・プランを選択できます。

BioRS ニックネーム・カーディナリティー統計を更新するには、SYSSTAT.TABLES カタログ・ビューを正しいカーディナリティー番号で変更してください。

前提条件:

更新する統計のニックネームに対応する BioRS データ・バンクのカーディナリティー番号を判別してください。

手順:

以下の構文を使用して UPDATE ステートメントを発行してください。

```
UPDATE sysstat.tables SET card=<cardinality>
WHERE tabschema=<nickname-schema>
AND tabname=<nickname-name>
```

- <*cardinality*> は、更新対象の統計を持つニックネームに対応する BioRS データ・バンク・カーディナリティー番号です。
- <*nickname-schema*> は、更新対象の統計を持つニックネームに関連するスキーマの名前です。
- <*nickname-name*> は、更新対象の統計を持つニックネームの名前です。

関連概念:

- 25 ページの『BioRS 統計情報』

関連タスク:

- 26 ページの『BioRS データ・バンク・カーディナリティー統計の確認』
- 11 ページの『BioRS 列カーディナリティー統計』
- 27 ページの『BioRS _ID_ 列カーディナリティーの更新』

BioRS _ID_ 列カーディナリティーの更新

BioRS _ID_ エレメントにマップする列の正しいカーディナリティー統計を維持することで、オプティマイザーと BioRS ラッパーが最適なデータ・アクセス・プランを選択できます。

BioRS _ID_ エレメントにマップする列のカーディナリティー番号を更新するには、SYSSTAT.COLUMNS カタログ・ビューを変更する必要があります。

前提条件:

列を参照するニックネームに対応する BioRS データ・バンクのカーディナリティー番号を判別してください。BioRS _ID_ エレメントにマップする列のカーディナリティー番号は、列を参照するニックネームのカーディナリティーと一致する必要があります。

手順:

BioRS _ID_ 列カーディナリティー統計を更新するには、以下の構文を使用して UPDATE ステートメントを発行してください。

```
UPDATE sysstat.columns SET colcard=<<cardinality>)
WHERE
     tabschema=<nickname-schema>
  AND tabname=<nickname-name>
  AND colname IN (SELECT colname FROM syscat.coloptions
                  WHERE
```

```
tabschema=<nickname-name>
AND tabname=<nickname-name>
AND option='ELEMENT_NAME';
AND setting='_ID_')
```

- <*cardinality*> は、列のニックネームに対応する BioRS データ・バンク・カーディナリティー番号です。
- <*nickname-schema*> は、列のニックネームに関連するスキーマの名前です。
- <*nickname-name*> は、列を使用するニックネームの名前です。

関連概念:

- 25 ページの『BioRS 統計情報』

関連タスク:

- 26 ページの『BioRS データ・バンク・カーディナリティー統計の確認』
- 26 ページの『BioRS ニックネーム・カーディナリティー統計の更新』
- 11 ページの『BioRS 列カーディナリティー統計』

BioRS AllText エレメント

BioRS システムのデータ・バンクにはそれぞれ、AllText と呼ばれるエレメントがあります。BioRS は、この索引付きエレメントをすべてのデータ・バンクに自動作成します。

AllText エレメントにより、特定の索引付けされたエレメントのみでなく、エントリーのテキストすべてを検索できるようになります。例えば、用語 muscus を検索すると、単語 muscus が表題、要約、説明、または有機体に使用されているエントリーを戻せます。

AllText エレメントを DB2 Information Integrator 照会で使用するには、AllText エレメントをニックネーム列にマップする必要があります。AllText エレメントをニックネーム列に適切にマップすると、そのニックネーム列を CONTAINS カスタム関数呼び出しで使用できます。

照会の SELECT リストが AllText エレメントをマップする列を参照している場合、常に NULL 値が戻ります。

関連タスク:

- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 19 ページの『BioRS ラッパー - 照会例』

ニックネーム変更時の注意 - BioRS ラッパー

事前に登録した BioRS ニックネームを ALTER NICKNAME ステートメントを使用して変更できます。ALTER NICKNAME ステートメントで、以下の事項が可能です。

- 列名の変更
- 列のデータ・タイプの変更
- 列のオプションの追加、変更、または削除

制約事項:

ニックネームが参照または使用する BioRS データ・バンク名は変更できません。ニックネームが使用する BioRS データ・バンクの名前を変更した場合、ニックネーム全体を除去し、再作成する必要があります。

REMOTE_OBJECT オプションを指定した場合、オプション値を削除または変更できません。

列のデータ・タイプを変更した場合、新しいデータ・タイプは、対応する BioRS エレメントのデータ・タイプと互換性を持つ必要があります。

ELEMENT_NAME オプションを使用して列のエレメント名を変更した場合、新規名の正確性はチェックされません。照会内で列が参照される場合、不正確なオプションがエラーを引き起こす可能性があります。

IS_INDEXED 列オプションを変更した場合、変更は BioRS サーバーで検査されません。照会内で列が参照される場合、不正確なオプションがエラーを引き起こす可能性があります。

関連情報:

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

カスタム関数表 - BioRS ラッパー

30 ページの表 9 に、4 つの BioRS カスタム関数それぞれの登録方法の例を示します。

カスタム関数を登録する際に参考となるサンプル・ファイル create_function_mappings.ddl が sqllib/samples/lifesci/biors ディレクトリーにあります。create_function_mappings.ddl ファイルは、各カスタム関数の定義を含みます。この DDL ファイルを実行すると、BioRS ラッパーをインストールした DB2 データベースごとにカスタム関数を登録できます。

表 9. BioRS ラッパーのカスタム関数

関数名	説明
CONTAINS (col VARCHAR(), term VARCHAR()), CONTAINS (col VARCHAR(), term CHAR()) CONTAINS (col VARCHAR(), term DATE) CONTAINS (col VARCHAR(), term TIMESTAMP)	与えられた式を使用して索引付きの列を検索します。 **col** 索引付き列 **term** 検索語
CONTAINS_LE (col VARCHAR(), term VARCHAR()), CONTAINS_LE (col VARCHAR(), term SMALLINT) CONTAINS_LE (col VARCHAR(), term BIGINT) CONTAINS_LE (col VARCHAR(), term DECIMAL) CONTAINS_LE (col VARCHAR(), term DOUBLE) CONTAINS_LE (col VARCHAR(), term REAL)	与えられた式を使用して索引付きの列を検索します。 **col** 索引付き列 **term** 検索語
CONTAINS_GE (col CHAR(), term CHAR()) CONTAINS_GE (col CHAR(), term DATE) CONTAINS_GE (col CHAR(), term TIMESTAMP) CONTAINS_GE (col CHAR(), term INTEGER) CONTAINS_GE (col CHAR(), term SMALLINT) CONTAINS_GE (col CLOB(), term DATE)	与えられた式を使用して索引付きの列を検索します。 **col** 索引付き列 **term** 検索語
SEARCH_TERM (col VARCHAR(), term VARCHAR()) SEARCH_TERM (col VARCHAR(), term CHAR()) SEARCH_TERM (col CHAR(), term VARCHAR()) SEARCH_TERM (col CHAR(), term CHAR())	BioRS 検索語を BioRS 検索エンジンに渡します。 **col** 索引付き列 **term** 検索語

関連タスク:

• 3 ページの『BioRS ラッパーのカスタム関数の登録』

BioRS ラッパーのメッセージ

このトピックでは、BioRS ラッパーの使用時に表示される可能性のあるメッセージについて説明します。メッセージの詳細については、「*DB2 メッセージ・リファレンス*」を参照してください。

表 10. BioRS のラッパーが発行するメッセージ

エラー・コード	メッセージ	説明
SQL0604N	列の長さ、精度、または位取り属性、特殊タイプ、構造化タイプ、構造化タイプの属性、関数、あるいはタイプ・マッピング <data-item> が無効です。	ニックネーム列のデータ・タイプが、基本データ・バンク・エレメントの BioRS タイプと互換性がありません。CREATE NICKNAME ステートメント内の列のデータ・タイプをチェックしてください。
SQL0901N	重大ではないシステム・エラーにより、SQL ステートメントが失敗しました。後続の SQL ステートメントは処理できます。 (理由 ″Error creating wrapper object″)	新しいラッパー・オブジェクトの作成時にエラーが発生しました。IBM ソフトウェア・サポート担当者に連絡してください。
SQL0901N	重大ではないシステム・エラーにより、SQL ステートメントが失敗しました。後続の SQL ステートメントは処理できます。 (理由 ″BioRS <trace-point>/<code>″)	これは内部エラーです。IBM ソフトウェア・サポート担当者に連絡してください。
SQL0901N	重大ではないシステム・エラーにより、SQL ステートメントが失敗しました。後続の SQL ステートメントは処理できます。 (理由 ″Memory allocation failed: <trace-point>″)	メモリー割り振りの際に、エラーが発生しました。フェデレーテッド (連合)・サーバーのホストに十分なメモリーがあるか確認し、照会を再度サブミットしてください。問題が解決しない場合は、IBM ソフトウェア・サポートに連絡してください。
SQL0901N	重大ではないシステム・エラーにより、SQL ステートメントが失敗しました。後続の SQL ステートメントは処理できます。 (理由 ″sqlno_crule_save_plans[100]:rc(-214272209) Empty plan list″)	オプティマイザー・プログラムと BioRS ラッパーが照会実行プランで一致しません。照会を単純化し、再実行してください。
SQL401N	演算 ″=″ のオペランドのデータ・タイプが一致していません。	照会が無効です。カスタム関数述部の右側の式は整数値でなければなりません。

表 10. BioRS のラッパーが発行するメッセージ (続き)

エラー・コード	メッセージ	説明
SQL1822N	予期しないエラー・コード "" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Databank not found" です。	CREATE NICKNAME ステートメントが参照する BioRS データ・バンクが BioRS サーバーで見つかりません。CREATE NICKNAME ステートメントをチェックし、参照先のデータ・バンク名が正しいか確認してください。
SQL1822N	予期しないエラー・コード "" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Connection timed out" です。	BioRS サーバーが TIMEOUT オプションが指定した時間内に通信要求に応答できませんでした。
SQL1822N	予期しないエラー・コード "<trace_point>" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Error reading from server" です。	BioRS サーバーからのデータ読み取り中に通信エラーが発生しました。エラーの詳細については、<trace_point> エラー・コードの値を確認してください。
SQL1822N	予期しないエラー・コード "<trace_point>" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Host not found" です。	HOST サーバー・オプションで指定した BioRS サーバー・ホストが見つかりません。CREATE SERVER ステートメントをチェックし、HOST サーバー・オプションの値が正しいか確認してください。
SQL1822N	予期しないエラー・コード "<trace_point>" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Unable to connect to server" です。	ラッパーは HOST サーバー・オプションが示すサーバーに接続できませんでした。エラーの詳細については、<trace_point> エラー・コードの値を確認してください。
SQL1822N	予期しないエラー・コード "<trace_point>" をデータ・ソース "BioRS wrapper" から受け取りました。関連したテキストとトークンは "Unable to create TCPIP socket" です。	ラッパーは TCPIP ソケットを作成できませんでした。エラー・コードの詳細については、<trace_point> エラー・コードの値を確認してください。

表 10. BioRS のラッパーが発行するメッセージ (続き)

エラー・コード	メッセージ	説明
SQL1822N	予期しないエラー・コード ″<trace_point>″ をデータ・ソース ″BioRS wrapper″ から受け取りました。関連したテキストとトークンは ″Error sending to server″ です。	ラッパーは要求を BioRS サーバーに送信できませんでした。エラーの詳細については、<trace_point> エラー・コードの値を確認してください。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Cannot change case-sensitivty of server″。	SQL ステートメントで CASE_SENSITIVE サーバー・オプションの値は変更できません。このオプションの値を変更するには、サーバーを除去する必要があります。その後、CREATE SERVER ステートメントを使用してサーバーを再作成し、CASE_SENSITIVE オプションに正しい値を指定する必要があります。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Multiple joins between two nicknames″。	2 つのニックネーム間では 1 つの結合述部のみ可能なため、照会が無効です。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Right side of function predicate must be constant″。	照会が無効です。カスタム関数述部の右側の式は定数でなければなりません。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Arg 1 of custom function not a column″。	照会が無効です。カスタム関数の最初の引き数は BioRS ニックネームの列を参照する必要があります。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Arg 1 of CONTAINS function not indexed″。	照会が無効です。BIORS.CONTAINS、BIORS.CONTAINS_LE、または BIORS.CONTAINS_GE 関数の最初の引き数が参照する列は索引付き列である必要があります。

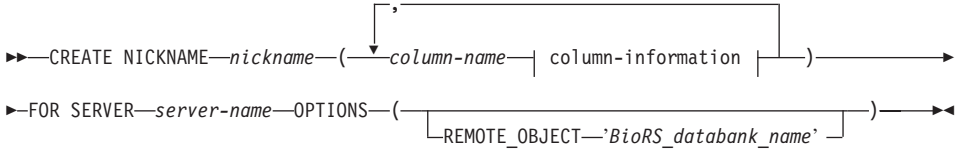
表 10. BioRS のラッパーが発行するメッセージ (続き)

エラー・コード	メッセージ	説明
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Bad type for arg1 of <function-name> function″。	照会が無効です。BIORS.CONTAINS、BIORS.CONTAINS_LE、または BIORS.CONTAINS_GE 関数の最初の引き数が参照する列が、不正なデータ・タイプです。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Arg 1 of SEARCH_TERM not _ID_ column″。	照会が無効です。SEARCH_TERM 関数の最初の引き数が参照する列が BioRS _ID_ をマップしません。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Bind parameter cannot be NULL″。	BIORS.CONTAINS 関数の 2 番目の引き数が参照する列またはホスト変数が NULL です。BioRS ラッパーは NULL 値を処理できません。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Cannot convert value to BioRS literal″。	値はリテラル、列、またはホスト変数でラッパーにサブミットされましたが、有効な BioRS リテラルに変換できませんでした。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Cannot change server version″。	ALTER SERVER ステートメントでサーバー・バージョンは変更できません。サーバー・バージョンを変更するには、サーバーを除去する必要があります。その後、CREATE SERVER ステートメントを使用してサーバーを現行バージョンに再作成してください。
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Bad type for arg2 of <function-name> function″。	照会が無効です。BIORS.CONTAINS、BIORS.CONTAINS_LE、または BIORS.CONTAINS_GE 関数の 2 番目の引き数が参照する列が、不正なデータ・タイプです。

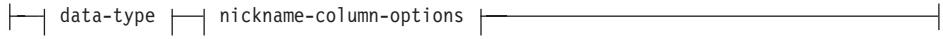
表 10. BioRS のラッパーが発行するメッセージ (続き)

エラー・コード	メッセージ	説明
SQL30090N	操作がアプリケーション実行環境で無効です。理由コード = ″Nickname has no columns″。	CREATE NICKNAME ステートメントに列宣言が指定されていません。ニックネームの作成時には、列宣言が必要です。

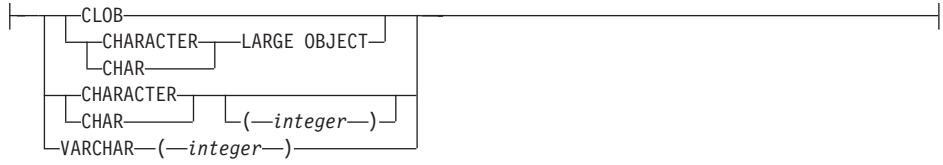
CREATE NICKNAME ステートメント構文 - BioRS ラッパー



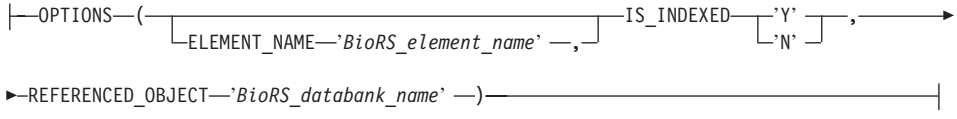
column-information:



data-type:



nickname-column-options:



ニックネーム列オプション

ニックネーム列オプションの値は、単一引用符で囲む必要があります。

ELEMENT_NAME

BioRS エレメント名を指定します。名前の大文字小文字の区別は、BioRS サーバーでの区別および CASE_SENSITIVE サーバー・オプションの値に依存します。BioRS エレメント名と列名が異なる場合のみ、エレメント名を指定する必要があります。

IS_INDEXED

対応する列が索引付けされているか (述部で列が参照可能か) を示します。有効値は 'Y' と 'N' です。値 'Y' は、BioRS サーバーにより対応するエレメントが索引付けされた列にのみ指定できます。

ニックネームの作成時に、このオプションは値 'Y' により BioRS が索引付けしたエレメントに対応する列に追加されます。

REFERENCED_OBJECT

このオプションは BioRS データ・タイプが Reference である列にのみ有効です。現行列が参照する BioRS データ・バンク名を指定します。名前の大文字小文字の区別は、BioRS サーバーでの区別および CASE_SENSITIVE サーバー・オプションの値に依存します。

ニックネーム・オプション

ニックネーム・オプション値は、単一引用符で囲む必要があります。

REMOTE_OBJECT

ニックネームに関連する BioRS データ・バンク名を指定します。ニックネームのスキーマと BioRS データ・バンクはこの名前で決まります。ニックネーム間の関係も指定します。名前の大文字小文字の区別は、BioRS サーバーでの区別および CASE_SENSITIVE サーバー・オプションの値に依存します。

重要: ALTER NICKNAME ステートメントではこの名前を変更または削除できません。このオプションが使用する BioRS データ・バンクの名前を変更した場合、ニックネーム全体を削除し、再作成する必要があります。

関連タスク:

- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 「*SQL リファレンス 第 2 巻*」の『CREATE NICKNAME ステートメント』
- 9 ページの『CREATE NICKNAME ステートメント - BioRS ラッパーの例』

CREATE SERVER ステートメント・オプション - BioRS ラッパー

BioRS の CREATE SERVER ステートメントのオプションは以下のとおりです。

TYPE サーバー・タイプを指定します。デフォルト値は `BioRS` です。BioRS ラッパーがサポートする値はデフォルト値のみです。このオプションを指定する必要はありません。

VERSION

サーバー・バージョンを指定します。デフォルト値は 1.0 です。BioRS ラッパーがサポートする値はデフォルト値のみです。このオプションを指定する必要はありません。

NODE BioRS 照会ツールが使用可能なシステムのホスト名を指定します。デフォルト値は *localhost* です。このオプションは指定する必要があります。

PORT BioRS サーバーへの接続に使用するポート番号を指定します。デフォルト値は 5014 です。このオプションは指定する必要があります。

TIMEOUT

BioRS ラッパーの BioRS サーバーからの応答待機時間を分単位で指定します。デフォルト値は 10 です。このオプションは指定する必要があります。

CASE_SENSITIVE

BioRS サーバーが、名前の大文字小文字を区別するかどうかを指定します。有効値は 'Y' または 'N' です。デフォルト値は 'Y' です。

BioRS 製品では、BioRS サーバー・マシンに格納されているデータの大文字小文字の区別を構成パラメーターで制御します。CASE_SENSITIVE オプションは、DB2 Information Integrator における BioRS システム構成パラメーターに相当します。BioRS システムおよび DB2 Information Integrator で、BioRS サーバーの大文字小文字の区別の構成設定を同期化する必要があります。BioRS および DB2 Information Integrator 間で大文字小文字の区別の構成設定を同じにしないと、DB2 Information Integrator を介して BioRS データへアクセスする際にエラーが発生します。

重要: DB2 Information Integrator に新規の BioRS サーバーを作成した後で、CASE_SENSITIVE オプションを変更または削除することはできません。CASE_SENSITIVE オプションを変更する必要がある場合は、サーバー全体を除去し、再作成する必要があります。BioRS サーバーを除去した場合、対応する BioRS ニックネームもすべて再作成する必要があります。DB2 Information Integrator は、除去されたサーバーに対応するすべてのニックネームを自動的に除去します。

関連タスク:

- 6 ページの『BioRS データ・ソースのサーバーの登録』
- 8 ページの『BioRS データ・ソースのニックネームの登録』

関連情報:

- 「*SQL リファレンス 第 2 巻*」の『CREATE SERVER ステートメント』
- 35 ページの『CREATE NICKNAME ステートメント構文 - BioRS ラッパー』

CREATE USER MAPPING ステートメント・オプション - BioRS ラッパー

GUEST

BioRS サーバーの BioRS ゲスト認証メカニズムを使用して運用するかどうかを指定します。有効値は 'Y' または 'N' です。デフォルト値は 'Y' です。

このオプションを 'Y' に設定すると、対象となる DB2 Information Integrator ユーザーはゲストの認証を使用して BioRS サーバーにアクセスします。

このオプションを 'N' に設定すると、対象となる DB2 Information Integrator ユーザーは BioRS サーバーへのアクセスに BioRS 許可 IDとパスワードが必要になります。

ユーザー・マッピングが作成されない場合、またはオプションを指定せずにユーザー・マッピングが作成された場合、DB2 Information Integrator ユーザーはゲスト認証で BioRS サーバーにアクセスします。

REMOTE_AUTHID

DB2 ユーザーの BioRS データ・ソースへのアクセスを可能にするユーザー ID を指定します。このリモート ID は、BioRS アプリケーションで受け入れ可能な形式でなければなりません。このオプションは、GUEST オプションを 'N' に設定した場合に設定する必要があります。

REMOTE_PASSWORD

このリモート ID のパスワードを指定します。このオプションは、GUEST オプションを 'N' に設定した場合に設定する必要があります。

例:

以下に示す CREATE USER MAPPING ステートメントは、ユーザー Charlie を Biors_Server1 サーバーのユーザー Charlene にマップします。

```
CREATE USER MAPPING FOR Charlie SERVER Biors_Server1
OPTIONS(GUEST 'N' REMOTE_AUTHID 'Charlene', REMOTE_PASSWORD 'Charlene_pw');
```

関連タスク:

- 6 ページの『BioRS データ・ソースのユーザー・マッピングの登録』

関連情報:

- 「*SQL リファレンス 第 2 巻*」の『CREATE USER MAPPING ステートメント』

第 2 章 ライフ・サイエンス・ユーザー定義関数

この章では、ライフ・サイエンス・ユーザー定義関数の概要、それらのフェデレーテッド (連合)・システムへの追加方法、そして照会における使用方法について説明します。

ライフ・サイエンス・ユーザー定義関数の概要

ライフ・サイエンス・ユーザー定義関数は、一般にデータ分析に使用されるアルゴリズムを研究者に提供します。これらの関数は、AIX® および Linux プラットフォームで使用可能な GeneWise 関数を除き、Windows® NT、AIX、および Linux 32 ビット・プラットフォームで使用可能です。

ライフ・サイエンス・ユーザー定義関数は、標準の 1 文字コード、および IUPAC-IUB あいまいコードを用いてアミノ酸およびヌクレオチドを示します。

ライフ・サイエンス・ユーザー定義関数を使用するには、それらを登録する必要があります。登録後は、必要に応じて削除可能です。

関連タスク:

- 40 ページの『ライフ・サイエンス・ユーザー定義関数の登録』
- 41 ページの『ライフ・サイエンス・ユーザー定義関数の除去』

関連情報:

- 39 ページの『機能カテゴリーごとのライフ・サイエンス・ユーザー定義関数』

機能カテゴリーごとのライフ・サイエンス・ユーザー定義関数

表 11 に、ライフ・サイエンス・ユーザー定義関数を機能カテゴリーごとにリストし、各カテゴリーの要旨を示します。

表 11. ライフ・サイエンス・ユーザー定義関数

機能カテゴリー	ユーザー定義関数	説明
逆変換	LSPep2AmbNuc、 LSPep2ProbNuc	アミノ・シーケンスをヌクレオチド・シーケンスに変換します。

表 11. ライフ・サイエンス・ユーザー定義関数 (続き)

機能カテゴリー	ユーザー定義関数	説明
定義行構文解析	LSDeflineParse	BLAST ラッパーにより戻される定義行エレメント、または FASTA 形式のデータ・ファイルにある定義行エレメントの構文を解析します。
一般化されたパターン・マッチング	LSPatternMatch、LSPrositePattern	ヌクレオチド・シーケンスまたはペプチド・シーケンスなどの所定ストリングの特定箇所を識別します。
GeneWise	LSGeneWise	プロテイン・シーケンスをゲノム・シーケンスに合わせます。
Motifs	LSMultiMatch、LSMultiMatch3、LSBarCode	ヌクレオチド・シーケンスまたはアミノ酸シーケンスのパターンの一致を図ります。
反転	LSRevNuc、LSRevPep、LSRevComp	ヌクレオチド・シーケンスまたはアミノ酸シーケンスを反転します。
変換	LSNuc2Pep、LSTransAllFrames	ヌクレオチド・シーケンスをペプチド・シーケンスに変換します。

関連概念:

- 39 ページの『ライフ・サイエンス・ユーザー定義関数の概要』

関連タスク:

- 40 ページの『ライフ・サイエンス・ユーザー定義関数の登録』
- 41 ページの『ライフ・サイエンス・ユーザー定義関数の除去』

ライフ・サイエンス・ユーザー定義関数の登録

ライフ・サイエンス・ユーザー定義関数を使用するには、それらを登録する必要があります。

手順:

ライフ・サイエンス・ユーザー定義関数を登録するには、`sqllib/bin` ディレクトリーにて Windows NT および AIX で使用可能な enable_LSFunctions コマンドを使用します。

enable_LSFunctions コマンドの構文は以下のとおりです。

```
enable_LSFunctions -n dbName -u userID -p password [-force]
```

dbName
関数を登録するデータベース名。

userID データベースに対して有効なユーザー ID。

password
ユーザー ID の有効なパスワード。

force 関数の除去と再登録に使用するフラグ。このフラグは、誤って破損または除去した関数の再登録に使用します。

コマンドは、スキーマ名 DB2LS を使用して関数を作成します。

enable_LSFunctions コマンドの出力例は以下のとおりです。

```
C:> enable_LSFunctions -n test -u db2admin -p db2admin

(0) Life Sciences Functions were found
    -- Create Life Sciences Functions ...
    Create Life Sciences Functions Successfully.

*** Please allow a few seconds to clean up the system ......
```

関数が登録されており、強制フラグを使用した場合、enable_LSFunctions コマンドの出力は以下のようになります。

```
C:> enable_LSFunctions -n test -u db2admin -p db2admin -force

(21) Life Sciences Functions were found

Life Sciences functions already exist ...
Reinstall Life Sciences functions ...
    -- Drop Life Sciences Functions ...
    Drop Life Sciences Functions Successfully.
    -- Create Life Sciences Functions ...
    Create Life Sciences Functions Successfully.


*** Please allow a few seconds to clean up the system ......
```

ライフ・サイエンス・ユーザー定義関数の除去

システム上のライフ・サイエンス・ユーザー定義関数が不要な場合は、除去できます。

手順:

ライフ・サイエンス・ユーザー定義関数を除去するには、sqllib/bin ディレクトリーにて Windows NT および AIX で使用可能な disable_LSFunctions コマンドを使用します。

disable_LSFunctions コマンドの構文は以下のとおりです。

```
disable_LSFunctions -n dbName -u userID -p password
```

dbName
関数を除去するデータベース名。

userID データベースに対して有効なユーザー ID。

password
ユーザー ID の有効なパスワード。

disable_LSFunctions コマンドの出力例は以下のとおりです。

```
C:>disable_LSFunctions -n test -u db2admin -p db2admin
a

(21) Life Sciences Functions were found
    -- Drop Life Sciences Functions ...
    Drop Life Sciences Functions Successfully.


*** Please allow a few seconds to clean up the system ......
```

逆変換ユーザー定義関数

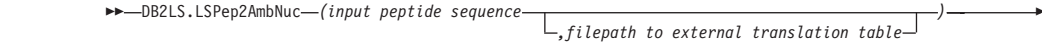
逆変換ユーザー定義関数を使用して、ペプチド・シーケンスをヌクレオチド・シーケンスに変換できます。逆変換では変換の逆の処理を行います。

アミノ酸からヌクレオチド・トリプレット・コドンへのマッピングは一対多数であるため、逆変換では 2 つの結果が発生します。

あいまい変換
単純なテキスト変換と検索。あいまいな変換を実行するには、LSPep2AmbNuc ユーザー定義関数を使用します。

最近似変換
コドン度数表から追加情報が必要です。最近似変換を実行するには、LSPep2ProbNuc ユーザー定義関数を使用します。

LSPep2AmbNuc ユーザー定義関数



input peptide sequence
ペプチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング

表記は、データ・タイプ VARCHAR であり、実際の長さが 10890 バイト以下である必要があります。入力データには、標準アミノ酸記号およびあいまいコードを使用します。

filepath to external translation table
カスタマイズした変換表を使用する場合は、その表へのファイル・パス情報を指定してください。パスのストリング値は、255 文字以下にしてください。

スキーマ名は DB2LS です。

LSPep2AmbNuc 関数を使用し、変換表に基づいてペプチド・シーケンスからあいまいなヌクレオチド・シーケンスを作成します。

関数の結果は、データ・タイプ VARCHAR の文字ストリングで、実際の長さは 32672 バイト以下になります。この結果は、内蔵または指定した変換表に基づくあいまいなヌクレオチド・シーケンスを示します。

変換表を指定しない場合、関数はデフォルトで表 12 を使用します。

表 12. デフォルトの変換表

アミノ酸記号	省略形	コドン
A	Ala	GCX
B	Asx	RAY
C	Cys	TGY
D	Asp	GAY
E	Glu	GAR
F	Phe	TTY
G	Gly	GGX
H	His	CAY
I	Ile	ATH
K	Lys	AAR
L	Leu	YTX
M	Met	ATG
N	Asn	AAY
P	Pro	CCX
Q	Gln	CAR
R	Arg	MGX
S	Ser	WSX
T	Thr	ACX

表 12. デフォルトの変換表 (続き)

アミノ酸記号	省略形	コドン
V	Val	GTX
W	Trp	TGG
X	Xxx	XXX
Y	Tyr	TAY
Z	Glx	SAR
*	End	TRR

関連情報:

- 45 ページの『LSPep2AmbNuc ユーザー定義関数のエラー・メッセージ』
- 46 ページの『LSPep2ProbNuc ユーザー定義関数』
- 44 ページの『LSPep2AmbNuc ユーザー定義関数の例』

LSPep2AmbNuc ユーザー定義関数の例

この関数は、values ステートメントで呼び出せます。入力は、以下の例のようにペプチド・シーケンス 1 つです。

```
values db2ls.LSPep2AmbNuc('HR');
```

上記の例では、あいまい変換および内蔵の変換表を使用してペプチドをヌクレオチドに変換します。この結果、標準アミノ酸記号から次のようなヌクレオチド・シーケンスが作成されます。

```
CAYMGX
```

以下の例では、あいまい変換と内蔵表を使用してペプチドをヌクレオチドに変換します。

```
values db2ls.LSPep2AmbNuc('SRGFGFITYSHSSMIDEAQKSRPHKIDGRVVEPKRA');
```

この values ステートメントの結果が、次のヌクレオチド・シーケンスです。(シーケンスは、このページに収まるように改行されています。)

```
WSXMGXGGXTTYGGXTTYATHACXTAYWSXCAYWSXWSXATGATHGAYGARGCXCARA
ARWSXMGXCCXCAYAARATHGAYGGXMGXGTXGTXGARCCXAARMGXGCX
```

次の例では、表またはニックネームから抽出された値のセットに適用する関数を示します。

```
SELECT DB2LS.LsPep2AmbNuc(peptide_seq) FROM table protein_table;
```

表 protein_table の peptide_seq 列のデータは、以下のようになります。

表 13. peptide_seq 列のデータ

peptide_seq
GIKEDTEEHHLRDYFE
QKYHTVNGHNCEVRKA
.....

select ステートメントの結果は次のとおりです。

```
GGXATHAARGARGAYACXGARGARCAYCAYYTXMGXGAYTAYTTYGAR
CARAARTAYCAYACXGTXAAYGGXCAYAAYTGYGARGTXMGXAARGCX
...
```

以下の例では、あいまい変換およびユーザー定義表を使用してペプチドをヌクレオチドに変換します。通常、変換表間の差は、あまりありません。固有記号が、1 つか 2 つ存在することがあります。これは、より多くのコドンを持つ種や、より少ないコドンを持つ種が存在するためです。例えば、ショウジョウバエはコドン AGG を持ちません。

```
values db2ls.LSPep2AmbNuc('RGNMGGGNYGNQNGGGNWNNG',
                           '\data\transl_table_06.txt')
```

ショウジョウバエ用の変換表を入力した場合、values ステートメントの結果は次のようになります。

```
MGRGGXAAYATGGGXGGXGGXAAYTAYGGXAAYTARAAYGGXGGXGGXAAYTGGAAYAAYGGX
```

関連情報:

- 42 ページの『LSPep2AmbNuc ユーザー定義関数』
- 74 ページの『LSNuc2Pep ユーザー定義関数の例』

LSPep2AmbNuc ユーザー定義関数のエラー・メッセージ

表 14. LSPep2AmbNuc ユーザー定義関数で発行されるメッセージ

エラー・コード	メッセージ	説明
SQL0443N	ルーチン ″DB2LS.LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUC″) が、診断テキスト ″Sequence not valid″ とともにエラー SQLSTATE を返しました。SQLSTATE=38608	指定したシーケンスが無効です。
SQL0443N	ルーチン ″DB2LS.LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUCUT″) が、診断テキスト ″No translation found″ とともにエラー SQLSTATE を返しました。SQLSTATE=38610	変換表ファイルが空です。

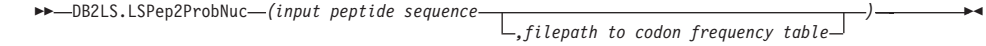
表 14. LSPep2AmbNuc ユーザー定義関数で発行されるメッセージ (続き)

エラー・コード	メッセージ	説明
SQL0443N	ルーチン ″LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUCUT″) が、診断テキスト ″Can open the translation table file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38612	指定された変換表ファイルが存在しません。
SQL0443N	ルーチン ″DB2LS.LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUCUT″) が、診断テキスト ″Line too long reading from file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38614	ファイルに規定よりも長い行が含まれています。
SQL0443N	ルーチン ″DB2LS.LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUCUT″) が、診断テキスト ″Invalid data file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38615	ファイルの形式が無効です。
SQL0443N	ルーチン ″LSPEP2AMBNUC″ (特定名 ″LSPEP2AMBNUCUT″) が、診断テキスト ″Can't construct the translation table″ とともにエラー SQLSTATE を返しました。SQLSTATE=38611	ファイルに無効な記号が検出されました。

関連情報:

- 42 ページの『LSPep2AmbNuc ユーザー定義関数』

LSPep2ProbNuc ユーザー定義関数



input peptide sequence
ペプチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 10890 バイト以下である必要があります。入力データには、標準アミノ酸記号を使用します。

filepath to codon frequency table
コドン度数表です。度数表へのファイル・パス情報を指定してください。パスのストリング値は、255 文字以下にしてください。

スキーマ名は DB2LS です。

LSPep2ProbNuc 関数を使用して、2 番目の引き数で指定されたコドン度数表に基づき、ペプチド・シーケンスから最近似のヌクレオチド・シーケンスを生成します。

関数の結果は、データ・タイプ VARCHAR の文字ストリングで、実際の長さは 32672 バイト以下になります。また、コドン度数表を用いた最近似ヌクレオチド・シーケンスを表します。

関連情報:

- 42 ページの『LSPep2AmbNuc ユーザー定義関数』
- 48 ページの『LSPep2ProbNuc ユーザー定義関数のエラー・メッセージ』
- 47 ページの『LSPep2ProbNuc ユーザー定義関数の例』

LSPep2ProbNuc ユーザー定義関数の例

yeast_high.cod 度数表に定義された最近似変換を使用して、ペプチド・シーケンスをヌクレオチド・シーケンスに変換する方法は以下のとおりです。

```
values db2ls.LSPep2ProbNuc('RDNNDDDN', '\data\yeast_high.cod')
```

この values ステートメントの結果は以下のとおりです。

```
AGAGACAATAACGACGATGATAAC
```

同じステートメントの 2 回目の実行では以下のストリングが形成されます。

AGA**GAT**AATAACGACGAT**GAC**AAC

同じステートメントの 3 回目の実行では以下のランダム値のストリングが形成されます。

AGAGAT**AAC**AACGAC**GACGATAAT**

太字で強調表示されたコドンは、現在と以前に行われた変換の相違を示します。

単一の values ステートメントの結果は、LSPep2ProbNuc 関数が以前の統計に基づき有力な記号を選択することを表します。この点は、変換可能なものがより多く存在する場合にあいまい記号を使用する LSPep2AmbNuc 関数と異なります。

LSPep2ProbNuc 関数は、各記号の最近似変換を選出し、すべての記号を事前選出したセットからランダム変換を使用して置き換えます。度数表に次のようなデータがあると想定します。

表 15. 度数表データのサンプル

アミノ酸	コドン	度数
Ala	GCG	0.17
Ala	GCA	0.13

表 15. 度数表データのサンプル (続き)

アミノ酸	コドン	度数
Ala	GCT	0.17
Ala	GCC	0.53

ペプチド・シーケンスが、4 つの「A」記号 (Ala) を含むと想定します。この関数は、A を GCC に 2 回、そして GCG と GCT にそれぞれ 1 回ずつ変換します。ただし、変換の順序はランダムになります。照会では、最初の A をセット {GCC、GCC、GCG、GCT} からの各変換に変換します。結果出力の DNA シーケンスには、常に 2 つの GCC、1 つの GCG、そして 1 つの GCT が表れます。同一シーケンス上で関数を複数回実行すると、値が置き換えられた DNA シーケンスが戻されることがあります。

関連情報:

- 46 ページの『LSPep2ProbNuc ユーザー定義関数』
- 48 ページの『LSPep2ProbNuc ユーザー定義関数のエラー・メッセージ』
- 44 ページの『LSPep2AmbNuc ユーザー定義関数の例』

LSPep2ProbNuc ユーザー定義関数のエラー・メッセージ

表 16. LSPep2ProbNuc ユーザー定義関数で発行されるメッセージ

エラー・コード	メッセージ	説明
SQL0443N	ルーチン ″DB2LS.LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″Sequence not valid″ とともにエラー SQLSTATE を返しました。SQLSTATE=38608	入力されたシーケンスが無効です。
SQL0443N	ルーチン ″DB2LS.LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″No translation found″ とともにエラー SQLSTATE を返しました。SQLSTATE=38610	コドン度数表ファイルが空です。
SQL0443N	ルーチン ″LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″Can’t open the translation table file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38612	ファイルが存在しません。

表 16. LSPep2ProbNuc ユーザー定義関数で発行されるメッセージ (続き)

エラー・コード	メッセージ	説明
SQL0443N	ルーチン ″DB2LS.LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″Line too long reading from file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38614	ファイルに規定より長い行が含まれます。
SQL0443N	ルーチン ″DB2LS.LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″Invalid data file″ とともにエラー SQLSTATE を返しました。SQLSTATE=38615	ファイルの形式が無効です。
SQL0443N	ルーチン ″LSPEP2PROBNUC″ (特定名 ″LSPEP2PROBNUC″) が、診断テキスト ″Can't construct the translation table″ とともにエラー SQLSTATE を返しました。SQLSTATE=38611	ファイルに無効な記号が含まれます。

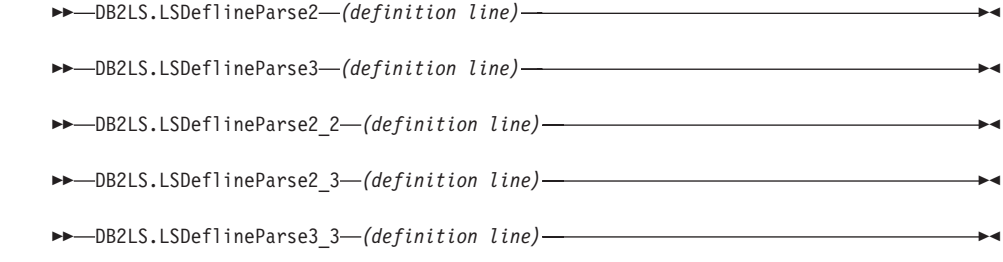
関連情報:

- 46 ページの『LSPep2ProbNuc ユーザー定義関数』
- 47 ページの『LSPep2ProbNuc ユーザー定義関数の例』

定義行構文解析のユーザー定義関数

定義行構文解析のユーザー定義関数は、定義行エレメントを解析します。例えば、定義行から解析したシーケンス ID の他のデータ・ソースとの結合を可能にし、あるいは `'species = "human"'` のような定義行部分の述部を評価します。defline 関数は、最も一般的な定義行の形式に対応します。例えば、BLAST ラッパーが戻す定義行エレメント、または FASTA 形式のデータ・ファイルに示される定義行エレメントです。

LSDeflineParse ユーザー定義関数



definition line

FASTA 形式で定義行を示す有効なストリングです。ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 1024 バイト以下である必要があります。

スキーマ名は DB2LS です。

各 LSDeflineParse 関数は、NCBI 標準 FASTA シーケンス ID (NSID) およびその記述のフィールドを解析し、表の列へ出力します。複合定義である定義行は、各行に単一のコンポーネント定義を含む複数行に出力されます。

LSDeflineParse2 は 2 つのフィールドから成る NSID を持つ定義行を解析します。関数の結果は、4 列の表になります。

表 17. LSDeflineParse2 ユーザー定義関数の結果表の列の説明

列名	説明
ROWID	関数が戻す行数 (整数)。
TAG	NSID タグを表す 3 文字以下の VARCHAR。
IDENTIFIER	NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
DESCRIPTION	1019 文字以下の VARCHAR。

3 つのフィールドから成る NSID を持つ定義行を解析します。関数の結果は、5 列の表になります。

表 18. LSDeflineParse3 ユーザー定義関数の結果表の列の説明

列名	説明
ROWID	関数が戻す行数 (整数)。
TAG	NSID タグを表す 3 文字以下の VARCHAR。
ACCESSION	NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
LOCUS	NSID の 3 番目の ID フィールドを表す 20 文字以下の VARCHAR。
DESCRIPTION	1017 文字以下の VARCHAR。

LSDeflineParse2_2 は、2 つのフィールドを持つ NSID が連結されたペアから成る複合 ID の定義行を解析します。関数の結果は、6 列の表になります。

表 19. LSDeflineParse2_2 ユーザー定義関数の結果表の列の説明

列名	説明
ROWID	関数が戻す行数 (整数)。
TAG1	1 番目の ID の NSID タグを表す 3 文字以下の VARCHAR。
IDENTIFIER1	1 番目の NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
TAG2	1 番目の ID の NSID タグを表す 3 文字以下の VARCHAR。
IDENTIFIER2	2 番目の NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
説明	1015 文字以下の VARCHAR。

LSDeflineParse2_3 は、2 つのフィールドから成る NSID が 3 つのフィールドから成る NSID と連結された複合 ID の定義行を解析します。入力定義行での連結順 (2 つのフィールドから成る NSID と 3 つの フィールドから成る NSID の順) は問題でありません。関数の結果は、7 列の表になります。

表 20. LSDeflineParse2_3 ユーザー定義関数の結果表の列の説明

列名	説明
ROWID	関数が戻す行数 (整数)。
TAG1	2 つのフィールド ID の NSID タグを表す 3 文字以下の VARCHAR。
IDENTIFIER	2 つの フィールドから成る NSID の 2 番目の ID を表す 20 文字以下の VARCHAR。
TAG2	3 つのフィールドから成る ID の NSID タグを表す 3 文字以下の VARCHAR。
ACCESSION	3 つのフィールドから成る NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
LOCUS	3 つのフィールドの NSID の 3 番目の ID フィールドを表す 20 文字以下の VARCHAR。
DESCRIPTION	1013 文字以下の VARCHAR。

LSDeflineParse3_3 は、3 つのフィールドを持つ NSID ペアから成る複合 ID の定義行を解析します。関数の結果は、8 列の表になります。

表 21. LSDeflineParse3_3 ユーザー定義関数の結果表の列の説明

列名	説明
ROWID	関数が戻す行数 (整数)。
TAG1	1 番目の ID の NSID タグを表す 3 文字以下の VARCHAR。
ACCESSION1	1 番目の NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
LOCUS1	1 番目の NSID の 3 番目の ID フィールドを表す 20 文字以下の VARCHAR。
TAG2	1 番目の ID の NSID タグを表す 3 文字以下の VARCHAR。
ACCESSION2	2 番目の NSID の 2 番目の ID フィールドを表す 20 文字以下の VARCHAR。
LOCUS2	2 番目の NSID の 3 番目の ID フィールドを表す 20 文字以下の VARCHAR。
説明	1014 文字以下の VARCHAR。

関連情報:

- 52 ページの『LSDeflineParse ユーザー定義関数の例』

LSDeflineParse ユーザー定義関数の例

このトピックでは、LSDeflineParse ユーザー定義関数が定義行を結果表へ解析する方法を、7 つの例を使用して解説します。

以下の照会および結果表の例では、LSDeflineParse2 ユーザー定義関数が 2 つのフィールドを持つ NSID の定義行を解析する方法を示します。

```
select *
from table(DB2LS.LSDeflineParse2(
           '>gi|12346 hypothetical protein 185 -wheat chloroplast')) as t
```

結果表は次のデータを含みます。

表 22. LSDeflineParse2 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG	gi
IDENTIFIER	12346
DESCRIPTION	hypothetical protein 185 - wheat chloroplast

以下の照会および結果表の例では、LSDeflineParse2 ユーザー定義関数が 3 つのフィールドを持つ NSID を含む定義行を解析する方法を示します。

```
select *
from table(DB2LS.LSDeflineParse3('
          >gb|U37104|APU37104 Aethia pusilla cytochrome b gene')) as t
```

結果表は次のデータを含みます。

表 23. LSDeflineParse3 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG	gb
ACCESSION	U37104
LOCUS	APU37104
DESCRIPTION	Aethia pusilla cytochrome b gene

以下の照会および結果表の例では、LSDeflineParse2 ユーザー定義関数が 2 つのフィールドを持つ NSID の複合 ID を含む定義行を解析する方法を示します。

```
select *
from table(DB2LS.LSDeflineParse2_2(
          '>gb|U37104|gim|73401A Aethia pusilla cytochrome b gene')) as t
```

結果表は次のデータを含みます。

表 24. LSDeflineParse2 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG1	gb
IDENTIFIER1	U37104
TAG2	gim
IDENTIFIER2	73401A
DESCRIPTION	Aethia pusilla cytochrome b gene

以下の照会例では、3 つのフィールドを持つ NISD と 2 つのフィールドを持つ NISD の連結からなる複合 ID を持つ定義行を含みます。LSDeflineParse2_3 関数が定義行を解析する方法を次に示します。

```
select *
from table(DB2LS.LSDeflineParse2_3('
          >gi|12346|gp|CAA44030.1|CHTAHSRA_4
          hypothetical protein 185 - wheat chloroplast')) as t
```

結果表は次のデータを含みます。

表 25. LSDeflineParse2 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG1	gi
IDENTIFIER	12346
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 - wheat chloroplast

以下の照会例では、2 つのフィールドから成る NISD が 3 つのフィールドから成る NISD に連結した複合 ID の定義行を含みます。LSDeflineParse2_3 関数が定義行を解析する方法を次に示します。

```
select *
from table(DB2LS.LSDeflineParse2_3('
           >gp|CAA44030.1|CHTAHSRA_4|gi|12346
           hypothetical protein 185 - wheat chloroplast')) as t
```

結果表は次のデータを含みます。

表 26. LSDeflineParse2 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG1	gi
IDENTIFIER	12346
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 - wheat chloroplast

以下の照会および結果表の例では、LSDeflineParse3_3 ユーザー定義関数が 3 つのフィールドを持つ NSID のペアから成る複合 ID を持つ定義行を解析する方法を示します。

```
select * from table(DB2LS.LSDeflineParse3_3('
                   >dbj|AAD55586.1|AF055084_1|gp|CAA44030.1|CHTAHSRA_4
                   hypothetical protein 185 - wheat chloroplast')) as t
```

結果表は次のデータを含みます。

表 27. LSDeflineParse3 ユーザー定義関数の結果データ

列名	データ
ROWID	1
TAG1	dbj
ACCESSION1	AAD55586.1
LOCUS1	AF055084_1
TAG2	gp
ACCESSION2	CAA44030.1
LOCUS2	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 - wheat chloroplast

複合定義行の解析には、任意の定義行ユーザー定義関数を使用できます。以下の照会例は、コントロール A で分離された複数定義を持つ複合定義行を含みます。このタイプの定義行は、NCBI の非冗長プロテイン・データベース nr で検出できます。LSDeflineParse2_3 関数が定義行を解析する方法を次に示します。

```
select *
from table(DB2LS.LSDeflineParse2_3('
           >gi|12346|gp|CAA44030.1|CHTAHSRA_4
           hypothetical protein 185 - wheat chloroplast
     ^Agp|CAA44030.1|CHTAHSRA_4|gi|12346
           hypothetical protein 185 - wheat chloroplast')) as t
```

結果表は次のデータを含みます。

表 28. LSDeflineParse2 ユーザー定義関数の結果データ

列名	データ	データ
ROWID	1	2
TAG1	gi	gi
IDENTIFIER	12346	12346
TAG2	gp	gp
ACCESSION	CAA44030.1	CAA44030.1
LOCUS	CHTAHSRA_4	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 - wheat chloroplast	hypothetical protein 185 - wheat chloroplast

関連情報:

- 49 ページの『LSDeflineParse ユーザー定義関数』

一般化されたパターン・マッチングのユーザー定義関数

一般化されたパターン・マッチングのユーザー定義関数は、ヌクレオチド・シーケンスまたはペプチド・シーケンスなどの、所定ストリングの特定箇所を識別します。

LSPatternMatch ユーザー定義関数

```
►►—DB2LS.LSPatternMatch—(input character sequence, pattern)—————————►◄
```

input character sequence
文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

pattern 有効な任意の Perl 正規表現で指定されたパターンです。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

LSPatternMatch ユーザー定義関数を使用して、指定パターンの入力されたヌクレオチド・シーケンスまたはペプチド・シーケンスを検索できます。

関数は、シーケンス内で最初に一致するパターンの位置を表す整数を結果として戻します。一致するものがない場合、この関数はゼロを戻します。

ROSITE 構文で記述されたパターンがある場合は、それらを LSPrositePatternax ユーザー定義関数で Perl 構文に変換できます。変換した構文は、LSPatternMatch ユーザー定義関数で使用できます。

関連情報:

- 56 ページの『LSPatternMatch ユーザー定義関数の例』
- 58 ページの『LSPrositePattern ユーザー定義関数』

LSPatternMatch ユーザー定義関数の例

次の例では、「coward」、「cowage」、「cowboy」、または「cowl」と一致するストリングの開始位置を検索します。

```
values DB2LS.LSPatternMatch('joe the cowboy is next', 'cow(ard|age|boy|l)')
```

関数は文字を検索します。この例では、値 9 を戻します。ストリング「cowboy」は、先頭位置を 1 とすると、9 の位置から開始します。

次の例では、「not」および「non」と一致するストリングの開始位置を検索します。

```
values DB2LS.LSPatternMatch('match not and non but
                             no match for no or none', 'no[tn] ')
```

関数は文字を検索します。この例では、値 7 を戻します。ストリング「not」は、先頭位置を 1 とすると、7 の位置から開始します。

LSPatternMatch は、select ステートメントで SQL LIKE ステートメントより影響力のある PERL 構文を使用して、結果をフィルターにかけるのに役立ちます。次の例では、Blast 出力上の LSPatternMatch を使用して、特定のパターンと一致する遺伝子をフィルターにかけます。

```
SELECT BlastOutput.*
 FROM BlastOutput
 WHERE db2ls.LSPatternMatch(HSP_H_Seq, 'F[GSTV]PRL') > 0;
```

PROSITE 構文により精通している場合は、上記の照会で LSPrositePattern も使用できます。照会を以下のように変更します。

```
SELECT BlastOutput.*
  FROM BlastOutput
  WHERE db2ls.LSPatternMatch(HSP_H_Seq,
          db2ls.LSPrositePattern('F-[GSTV]-P-R-L.') ) > 0;
```

パターン・マッチング関数は、ヌクレオチド・シーケンスまたはペプチド・シーケンスと同様に、他のテキスト・タイプの検索にも役立ちます。パフォーマンスに配慮する場合は、SQL LIKE ステートメントを使用してください。

次の例では、アライメントのサブジェクトまたはターゲット行にあるプロテイン motif に基づいて、BLAST hsp アライメントをフィルターにかける照会を示します。この例は、「Zhang,Z., Schaffer,A.A., Miller,W., Madden,T.L., Lipman,D.J., Koonin,E.V. and Altschul,S.F. (1998) Protein sequence similarity searches using patterns as seeds. *Nucl. Acids Res.*, **26**, 3896-3990.」から採用しました。

以下の照会は、P-loop ATPase ドメイン [GA]xxxxGK[ST] を含むサブジェクト・シーケンスであるアライメントのみを戻します。照会は、シノラブディス・エレガンス の細胞死を調整する CED4 を、NCBI の非冗長プロテイン・シーケンス・データベースに対する照会シーケンスに使用します。データベースは、GenBank エントリー X69016 の CDS 機能の変換から、Blast 照会を検索します。

```
SELECT HSP_Q_Seq, HSP_Midline, HSP_H_Seq
FROM BlastP b, GBseq gs, gbfeat gf, gbqual gq
WHERE gs.PRIMARYACCESSION = 'X69016' and
                  gs.sequencekey = gf.sequencekey and
                  gf.featurejoinkey = gq.featurejoinkey and
                  gf.FeatureKey = 'CDS' and
                  gq.QualifierName = 'translation' and
                  gq.QualifierValue = b.BlastSeq and
                  db2ls.LSPatternMatch(HSP_H_Seq,
                    db2ls.LSPrositePattern('[GA]-x(4)-G-K-[ST].') ) > 0;
```

正規の照会シーケンスに該当し、推定上の単一ヌクレオチド多形態 (SNP) を含むゲノム・シーケンスから HSP を検索するには、以下の照会例を使用できます。これは、

「Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky, and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.」から採用しています。

照会は、Blast hsp の中央線上のパターン・マッチングを使用して、≥20 完全一致、単一の不一致、≥20 完全一致が連続するパターンを検索します。つまり、20 の「|」文字、シングル・スペース、そして 20 の「|」文字がアライメントの中央線にあります。

またこの例では、ヌクレオチド・シーケンスまたはペプチド・シーケンスではないストリングに対する LSPatternMatch ユーザー定義関数の使用法を示します。

```
SELECT   HSP_Info, HSP_Midline, HSP_H_Seq
FROM BlastOutput
WHERE db2ls.LSPatternMatch(HSP_Midline, '\|{20} \|{20}') > 0;
```

以前の照会を以下のように再書き込みできます。

```
SELECT HSP_Info, HSP_Midline, HSP_H_Seq, func.Position, func.Match
FROM BlastOutput,
     table( select * as c from table(
            LSMultiMatch(HSP_Midline, '\|{20} \|{20}') )
     as f) as func
```

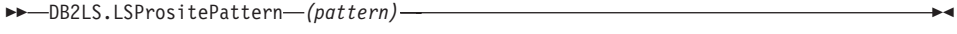
2 番目の照会は、一致するストリングおよびそのシーケンスでの位置の他、一致する Blast 行を戻します。

BlastOutput は BlastN ニックネームに関するビューです。

関連情報:

- 59 ページの『LSPrositePattern ユーザー定義関数の例』
- 56 ページの『LSPatternMatch ユーザー定義関数』
- 58 ページの『LSPrositePattern ユーザー定義関数』

LSPrositePattern ユーザー定義関数



pattern Prosite 構文で指定されたパターン・マッチング構文です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

LSPrositePattern ユーザー定義関数を使用して、PROSITE 構文から PERL 構文に変換できます。変換された構文は、LSPatternMatch、LSMultiMatch、および LSMultiMatch3 のユーザー定義関数で使用できます。

関数の結果は、Perl 構文中で正規表現を表す文字ストリングになります。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

関連情報:

- 59 ページの『LSPrositePattern ユーザー定義関数の例』
- 56 ページの『LSPatternMatch ユーザー定義関数』

LSPrositePattern ユーザー定義関数の例

次の例では、PROSITE 構文から PERL 構文へパターンを変換します。

```
values db2ls.LSPrositePattern('[AC]-x-V-x(4)-{ED}.');
```

この関数は、以下の例のように PROSITE 構文の入力パターンを Perl 構文で相当するパターンに変換します。

```
[AC].V.{4}[^ED]
```

次の例では、PROSITE の別の構文パターンを PERL 構文に変換します。

```
values db2ls.LSPrositePattern('<A-x-[ST](2)-x(0,1)-V.');
```

この関数は、入力パターンに基づいて PROSITE 構文からストリングを変換し、以下を戻します。

```
\AA.[ST]{2}.{0,1}V
```

次の例では、ID 番号 PS01205 を所有する PROSITE データベース・エントリーに対応するパターンを、パターン・マッチング関数で入力として用いられる PERL パターンに変換します。

```
values db2ls.LSPrositePattern('R-P-L-[IV]-x-[NS]-F-G-S-[CA]-T-C-P-x-F.')
```

この照会の結果は次のとおりです。

```
RPL[IV].[NS]FGS[CA]TCP.F
```

次の例では、この関数を照会で使用する方法を示します。照会は、指定された PROSITE パターンと一致するシーケンスのみを出力します。

```
SELECT H_Accession, HSP_Info, HSP_H_Seq
FROM BlastOutput
WHERE db2ls.LSPatternMatch( HSP_H_Seq,
  db2ls.LSPrositePattern('R-P-L-[IV]-x-[NS]-F-G-S-[CA]-T-C-P-x-F.') ) > 0;
```

次の例では、ID PS00261 を持つ PROSITE エントリーに対応するパターンを変換します。

```
values db2ls.LSPrositePattern('C-[STAGM]-G-[HFYL]-C-x-[ST].')
```

この照会の結果は次のとおりです。

```
C[STAGM]G[HFYL]C.[ST]
```

関連情報:

- 56 ページの『LSPatternMatch ユーザー定義関数の例』
- 58 ページの『LSPrositePattern ユーザー定義関数』

正規表現のサポート

正規表現は、オープン・ソース・ソフトウェアである PCRE のライブラリー・パッケージによりサポートされます。これは Philip Hazel 氏によって記述され、著作権はイギリスのケンブリッジ大学に属します。

ソースは ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/ で参照できます。

GeneWise ユーザー定義関数

GeneWise ユーザー定義関数は、プロテイン・シーケンスをゲノム・シーケンスにより調整します。

GeneWise は一般的に使用されているコンポーネントで、プロテイン・シーケンスをゲノム DNA シーケンスにより調整します。ただし、イントロンおよびフレームシフト・エラーが発生する可能性があります。

GeneWise へのリンク

このトピックでは、GeneWise ライブラリーへのリンクの手順を解説します。

手順:

1. Wise2 パッケージのバージョン 2.1.20c を http://www.ebi.ac.uk/Wise2 からダウンロードします。
2. 必要に応じて、アーカイブをフォルダーに拡張します。
3. pthread サポートでパッケージをコンパイルします。このステップに関する詳細は、Wise2 の資料を参照してください。
4. **make api** のルート・ディレクトリーでこれを実行します。
5. WISE2_HOME 環境変数を、Wise2 パッケージのルート・ディレクトリーを提示するように設定します。
6. sqllib/cfg/db2dj.ini ファイルの WISECONFIGDIR 変数を、wisecfg サブディレクトリーを示すように設定します。

   例えば、Wise2 パッケージを /usr/wise2.1.20c/ にインストールする場合、WISECONFIGDIR=/usr/wise2.1.20c/wisecfg/ を db2dj.ini ファイルに追加します。

7. **djxlinkLSGeneWise** を sqllib/bin ディレクトリーから実行し、その出力を検査します。
8. sqllib/lib ディレクトリーの djxlinkLSGeneWise.out を検査します。
9. エラーが報告されない場合は、このライブラリーの作成が正常に行われたことを示します。

関連情報:

- 61 ページの『LSGeneWise ユーザー定義関数』

LSGeneWise ユーザー定義関数

```
►►—DB2LS.LSGeneWise—(protein sequence, DNA_sequence)—————————————►◄
```

protein sequence
ペプチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

DNA_sequence
ヌクレオチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

表 29 では、LSGeneWise 関数が戻す 1 行の表を説明します。

表 29. LSGeneWise 関数による出力表の列名、タイプ、および説明

列名	タイプ	説明
PROTEIN_OFFSET	INTEGER	アライメントが検索された入力プロテイン・シーケンス内の開始オフセットです。
DNA_OFFSET	INTEGER	アライメントが検索された入力 DNA シーケンス内の開始オフセットです。
PROTEIN	VARCHAR(32672)	位置合わせされたシーケンスを表す入力シーケンスのフラグメントです。
SIMILARITY	VARCHAR(32672)	プロテイン・シーケンスと DNA シーケンス間の一致を示します。完全一致は、それに対応する記号文字で記されます。スコアが正で完全一致でないものは「+」で示され、不一致はスペースで示されます。
TRANSLATED_DNA	VARCHAR(32672)	DNA シーケンスが変換されたものです。シーケンスには、ダッシュおよび削除やイントロンのような特殊記号が含まれる可能性があります。

表 29. *LSGeneWise* 関数による出力表の列名、タイプ、および説明 *(続き)*

列名	タイプ	説明
DNA	VARCHAR(32672)	フレーム・シフトおよびイントロンなどの特殊マーカーを持つ DNA シーケンスです。

GeneWise プログラムの出力と LSGeneWise UDF の出力の一致点は以下のとおりです。

- GeneWise プログラムで出力されるプロテイン・オフセットおよび DAN オフセットは、PROTEIN_OFFSET および DNA_OFFSET 列と一致します。
- GeneWise で先頭行に出力されるプロテイン・シーケンスは、PROTEIN 列と一致します。
- GeneWise で出力される 2 行目は、SIMILARITY 列と一致します。
- GeneWise で出力される 3 行目は、TRANSLATED_DNA 列と一致します。
- GeneWise で出力される 4 行目、5 行目、および 6 行目は、縦方向に読み取ることで DNA 列に結合されます。

LSGeneWise ユーザー定義関数を使用して、イントロンおよびフレームシフト・エラーを考慮しながら、プロテイン・シーケンスをゲノム・シーケンスで調整してください。

LSGeneWise ユーザー定義関数の出力に関する詳細は、http://www.ebi.ac.uk/Wise2 を参照してください。

関連タスク:

- 60 ページの『GeneWise へのリンク』

関連情報:

- 62 ページの『LSGeneWise ユーザー定義関数の例』

LSGeneWise ユーザー定義関数の例

次の例は、LSGeneWise ユーザー定義関数を使用した照会およびその結果を示します。

```
select protein_offset, dna_offset, protein, similarity, translated_dna, dna
from table( db2ls.LSGeneWise( '
             VEPKRAVPRQDIDSPNAGATVKKLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKETGK
             KRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVDVKKALPKQNDQQGGGGGRGGPGGRAGGNR
             GNMGGGNYGNQNGGGNWNNGGNNWGNNR',
            'CACTTAACTGTGAAAGATATTTGTTGGTGGCATTAAAGAAGACACTGAAGAACATCACCTAAG
             AGATTATTTTGAACAGTATGGAAAAATTGAAGTGATTGAAATCATGACTGACCGAGGCAGTGG
             CAAGAAAAGGGGCTTTGCCTTRGTAACCTTTGACGACCATGACTCCGTGGATAAGATTGTCAT
             TCAGAAATACCATACTGTGAATGGCCACAACTGTGAAGTTAGAAAAGCCCTGTCAAAGCAAGA
             GATGGCTAGTGCTTCATCCAGCCAAAGAGGTCGAAGTGGTTCTGGAAACTTTGGTGGTGGTCG
             TGGAGGTGGTTTCGGTGGGAATGACAACTTCGGTCGTGGAGGAAACTTCAGTGGTCGTGGTYG
             CTTTGGTGGCAGCCGTGGTGGTGGTGGATATGGTGGC' ) ) as f;
```

表 30. 結果表

列	データ
PROTEIN_OFFSET	23
DNA_OFFSET	14
PROTEIN	KLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKET GKKRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVD VKKALPKQNDQQGGGGGRGGPGGRAGGNRGNMGG GNYGNQNGGGNWNNGGN
SIMILARITY	K+FVG +K+D +E +RDYF+ +G I I I+ D+ +GKKRGFA+V FDD+D VDK+V+QK H +NG +V+KAL KQ RG G GN+GGG G G N+ GGN
TRANSLATED_DNA	KIFVGGIKEDTEEHHLRDYFEQYGKIEVIEIMTDRGSGK KRGFAxVTFDDHDSVDKIVIQKYHTVNGHNCEVRKAL SKQEMASASSSQRGRSGS------GNFGGGRGGGFGGNDNFGRGGN
DNA	aagatatttgttggtggcattaaagaagacactgaagaacatcacctaagagat...

関連タスク:

- 60 ページの『GeneWise へのリンク』

関連情報:

- 61 ページの『LSGeneWise ユーザー定義関数』

Motifs ユーザー定義関数

Motif ユーザー定義関数は、ヌクレオチド・シーケンスまたはアミノ・シーケンスのパターンを一致させます。

LSBarCode ユーザー定義関数

```
►►—DB2LS.LSBarCode—(input string sequence)—————————————————►◄
```

input string sequence
2 つのシーケンス・フラグメント間の HSP の配置を示す有効な文字ストリングです。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

LSBarCode ユーザー定義関数を使用してシーケンスを入力として使用し、垂直バー記号 (|) でスペースや正符号 (+) を除くすべての文字を置き換えることで、別のシーケンスを生成します。

関数の結果は、バーコード・シーケンスを表す変数文字シーケンスになります。

関連情報:

- 64 ページの『LSBarCode ユーザー定義関数の例』
- 65 ページの『LSMultiMatch ユーザー定義関数』
- 67 ページの『LSMultiMatch3 ユーザー定義関数』

LSBarCode ユーザー定義関数の例

この例では、ストリング・シーケンスからバーコードを作成します。

```
values db2ls.LSBarCode(
   'MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP ')
```

この values ステートメントの結果は以下のとおりです。

```
||| +|++|  ||  ++ +||||||+|| ||| +|+ + +|   |+||||+||||  ||
```

次の例で、この関数のより実際的な使用方法を解説します。BLAST 検索を実行する研究者が、完全一致の中から 25% 以下のプロリンを含む HSP アライメントのみを戻すと想定します。この例では、関数を使用して BLAST が戻す完全一致の中からプロリン (記号 P) のパーセンテージを計算します。この例では、LSMultiMatch3 ユーザー定義関数も呼び出します。照会は、match 関数を使用して完全一致を検出します。これは、Blast が常にアライメントにシーケンス・バー (「|」) を戻すとは限らないため、LSBarCode 関数とともに使用されます。次のようになります。

```
Query:       MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alignment:   MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Target:      MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

出力を、確実にバーの正しいシーケンスで位置合わせするには、LSBarCode 関数を使用します。この関数は、垂直バーでスペースおよび正符号を除くすべての文字を置換します。

```
SELECT BlastOutput.* , float( p )/ float( m ) AS percent_prolines
  FROM
  BlastOutput b,
  table(SELECT COUNT(*) AS p FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, 'P',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, 'P')
          ) AS f
        ) AS y,
  table(SELECT COUNT(*) AS m FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, '.',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
```

```
            b.HSP_H_Seq, '.')
          ) AS f
        ) AS z
WHERE float(p) / float(m) < 0.25;
```

この照会では、BlastOutput は Blast ニックネームに関するビューです。また、この照会は、LSMultiMatch3 関数を使用し、アライメントの完全一致を戻します。最初の使用で「P」記号の完全一致を戻し、2 回目の使用ですべての完全一致を戻します。結果表の行を表 31 に示します。

表 31. 結果行の例

HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_PROLINES
NIWDFMQGN...	NIWDFMQGN...	一致 = 80/80 (100%)、肯定 = 80/80 (100%)、ギャップ = 0/80 (0%)	+2.50000000000000E-002

上記の照会は、「Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.」から採用しました。

関連情報:

- 67 ページの『LSMultiMatch3 ユーザー定義関数の例』
- 63 ページの『LSBarCode ユーザー定義関数』

LSMultiMatch ユーザー定義関数

input nucleotide or peptide sequence
ヌクレオチド・シーケンスまたはペプチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

pattern Perl 言語で示されるパターン・マッチング構文です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

LSMultiMatch ユーザー定義関数を使用して、入力シーケンスでオーバーラップしない各一致に対し、表を戻すことができます。各表は、開始位置とマッチング・シーケンス・フラグメントから構成されます。

関数の結果は、2 列の表になります。最初の列は、シーケンス内パターンの一致の開始位置を表す整数です。2 列目は、マッチング・シーケンス・フラグメントです。

関連情報:

- 66 ページの『LSMultiMatch ユーザー定義関数の例』
- 63 ページの『LSBarCode ユーザー定義関数』
- 67 ページの『LSMultiMatch3 ユーザー定義関数』

LSMultiMatch ユーザー定義関数の例

次の例では、入力データから取得される重複しないすべての一致に対する位置およびマッチング・フラグメントを検索します。

```
SELECT position, match FROM table
  (LSMultiMatch('match not and non but no match for no or none',
      'no[tn] ')) as f
```

この照会は、一致の結果を表す select ステートメントに基づく表を戻します。

表 32. 複数行を戻す LSMultiMatch の結果

POSITION	MATCH
7	not
15	non

LSMultiMatch は、すべての一致に対して、その位置および一致するストリングを戻します。次の例では、特定の motif を含むシーケンス・エントリーを、Entrez Nucleotide から検索します。照会は、シーケンス ID および一致したシーケンスを出力します。開始および終了のサブパターン「.{0,9}」は、シーケンスの前と後で最大 9 文字と一致する必要があります。照会はこれらの文字も出力します。

```
select SequenceKey, Position, Match from GBSeq,
  table(db2ls.LSMultiMatch(Sequence, '.{0,9}(ATG|CGC)ACGGGC.{0,9}') )
  as fmatch
  WHERE entrez.contains(KeywordList,
        'Na/K/2Cl cotransporter AND nkcc1 gene') = 1;
```

この照会の結果は次のとおりです。

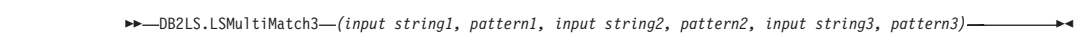
表 33. Entrez データの検索

SEQUENCEKEY	POSITION	MATCH
N02B59AE0.04DD4E84	1	TGCTTGGTGATGACGGGCTACCCCAAC
N02B59AE0.04DD4E84	91	GGCCATGTTCGCACGGGCTCCAGAAGG
N02B59AE0.04DC5EF4	1	TGCTTGGTGATGACGGGCTACCCCAAC
N02B59AE0.04DC5EF4	91	GGCCATGTTCGCACGGGCTCCAGAAGG

関連情報:

- 65 ページの『LSMultiMatch ユーザー定義関数』
- 63 ページの『LSBarCode ユーザー定義関数』
- 67 ページの『LSMultiMatch3 ユーザー定義関数』

LSMultiMatch3 ユーザー定義関数



input strings
有効な文字ストリングによる、ヌクレオチド・シーケンスまたはペプチド・シーケンス、または Blast 配置からの HSP_Midline ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

pattern Perl 言語で示されるパターン・マッチング構文です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

LSMultiMatch3 ユーザー定義関数を使用して 3 つのパターンおよび 3 つのストリングを入力し、3 つすべてのストリングが該当するそれぞれのパターンと一致する位置を戻します。このユーザー定義関数を使用してアライメント上に対するパターン・マッチングを実行できます。

関数の結果は、4 列の表になります。最初の列は、すべてのシーケンスにおけるパターンの一致の開始位置を表す整数です。関数は、先頭位置ですべてのストリングを一緒に固定します。2、3、および 4 列目は、マッチング・シーケンス・フラグメントです。

関連情報:

- 67 ページの『LSMultiMatch3 ユーザー定義関数の例』
- 65 ページの『LSMultiMatch ユーザー定義関数』
- 63 ページの『LSBarCode ユーザー定義関数』

LSMultiMatch3 ユーザー定義関数の例

以下の例では、この関数を使用し、Blast により戻された完全一致における特定のアミノ酸記号のパーセンテージを計算します。この例では、LSBarCode ユーザー定義関数も呼び出します。Blast は常に配置のバーのシーケンス (「|」) を戻すとは限らないため、必要になります。次のようになります。

```
Query:       MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alignment:   MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Target:      MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

出力を、確実に正しいバーのシーケンスで位置合わせするには、LSBarCode 関数を使用してそのシーケンスを変換します。この関数は、垂直バーでスペースおよび「+」を除くすべての文字を置き換えます。

```
SELECT BlastOutput.* , float( p )/ float( m ) AS percent_prolines
  FROM
  BlastOutput b,
  table(SELECT COUNT(*) AS p FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, 'P',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, 'P')
          ) AS f
        ) AS y,
  table(SELECT COUNT(*) AS m FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, '.',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, '.')
          ) AS f
        ) AS z
WHERE float(p) / float(m) < 0.25;
```

この照会では、BlastOutput は Blast 選択に関するビューです。また、この照会は、LSMultiMatch3 関数を使用し、アライメントの完全一致を戻します。最初の使用で「P」記号の完全一致を戻し、2 回目の使用ですべての完全一致を戻します。結果表の行を表 34 に示します。

表 34. 結果行の例

HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_PROLINES
NIWDFMQG...	NIWDFMQG...	一致 = 80/80 (100%)、肯定 = 80/80 (100%)、ギャップ = 0/80 (0%)	+2.50000000000000E-002

上記の照会は、「Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.」から採用しました。

以下の例では、3 つの異なるストリング・フラグメントで 3 つの異なるパターンを検索します。

```
SELECT position, match_1, match_2, match_3
  FROM table(db2ls.LSMultiMatch3('zaza', 'a', 'abab',
             'b', 'bcbc', 'c')) as f
```

この例では、以下の表に示すように、すべての一致に対する位置および一致するストリングを戻します。

表 35. 3 つの入力データを使用した複数一致の結果

POSITION	MATCH_1	MATCH_2	MATCH_3
2	a	b	c
4	a	b	c

次の例では、3 つの異なるストリング・フラグメント中で 3 つの異なるパターンを検索します。

```
SELECT position, match_1, match_2, match_3
  FROM table
  (LSMultiMatch3('cbccbbcccbbbccccbbbbccccc','c{1,3}b{1,3}c{1,3}',
   'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz',
   '.','01234567890123456789012345678901234567890123456789','\d')) as f
```

結果は以下のとおりです。

表 36. 3 つの入力データを使用した複数一致の結果

POSITION	MATCH_1	MATCH_2	MATCH_3
1	cbcc	a	0
7	cccbbbccc	g	6

関連情報:

- 64 ページの『LSBarCode ユーザー定義関数の例』
- 63 ページの『LSBarCode ユーザー定義関数』
- 67 ページの『LSMultiMatch3 ユーザー定義関数』

反転ユーザー定義関数

反転ユーザー定義関数が、ヌクレオチド・シーケンスまたはアミノ・シーケンスを反転させます。

LSRevComp ユーザー定義関数

input nucleotide sequence
ヌクレオチド・シーケンスを示す有効な文字ストリング表記です。シーケンスは、IUPAC あいまいコードを含むことができます。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

スキーマ名は DB2LS です。

関数の結果は、実際の長さが 32672 バイト以下であるデータ・タイプ VARCHAR の文字ストリングであり、ヌクレオチド・シーケンスの逆補数を示します。

関連情報:

- 70 ページの『LSRevComp ユーザー定義関数の例』
- 71 ページの『LSRevNuc ユーザー定義関数』
- 72 ページの『LSRevPep ユーザー定義関数』

LSRevComp ユーザー定義関数の例

LSRevComp 関数は、ヌクレオチド・シーケンスを受け入れる任意の内蔵関数を使用するように SQL ステートメント中で使用できます。例えば、以下のようになります。

```
SELECT DB2LS.LSRevComp(:NucSeq) FROM SYSDUMMY1;
```

この例では、この関数を使用して、ホスト変数からの入力シーケンスの逆補数を戻します。

無効ストリングおよび無効データ・タイプを使用した場合、以下のようなエラー・メッセージが示されます。

```
SQL0443N ルーチン "DB2LS.LSREVCOMP" (特定名 "LSREVCOMP") が、
診断テキスト "Sequence not valid" とともにエラー SQLSTATE を
返しました。SQLSTATE=38608
```

入力されたアルファベットが不適切である場合は、例外が発生します。

LSRevComp ユーザー定義関数は照会で以下のように使用されます。

```
SELECT HSP_H_Seq, db2ls.LSRevComp(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastN
WHERE BlastSeq='ccgctagtattggtcaatcttttgatatccaccgaa'
```

照会の結果は以下のとおりです。

```
HSP_H_SEQ                      REV_HSP_H_SEQ
------------------------------ ----------------------------

AGTATTGGTCAATCTTTTGAT          ATCAAAAGATTGACCAATACT

TGGTCAATCTTTTGATA              TATCAAAAGATTGACCA
```

```
TTGGCCAATCTTTTGATATCC           GGATATCAAAAGATTGGCCAA

TCAATCTTTTGATATCC              GGATATCAAAAGATTGA

GGATATCAAAAGATTGA              TCAATCTTTTGATATCC


  5 record(s) selected.
```

反転関数をその他のライフ・サイエンス・ユーザー定義関数と併用し、次の例のようにヌクレオチド・シーケンスの逆補数を変換できます。

```
values db2ls.LSNuc2Pep(
       db2ls.LSRevComp('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT'))
```

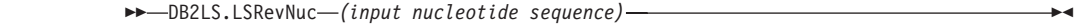
この照会は以下を戻します。

```
TSAT*EIR*GRQ*EK
```

関連情報:

- 69 ページの『LSRevComp ユーザー定義関数』

LSRevNuc ユーザー定義関数



input nucleotide sequence

ヌクレオチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。ヌクレオチド・シーケンスは、DNA アルファベットすべて、またはその一部である必要があります。

スキーマ名は DB2LS です。

関数の結果は、実際の長さが 32672 バイト以下であるデータ・タイプ VARCHAR の文字ストリングであり、ヌクレオチド・シーケンスの逆順を示します。

関連情報:

- 71 ページの『LSRevNuc ユーザー定義関数の例』
- 69 ページの『LSRevComp ユーザー定義関数』
- 72 ページの『LSRevPep ユーザー定義関数』

LSRevNuc ユーザー定義関数の例

LSRevNuc 関数は、ヌクレオチド・シーケンスを受け入れる任意の内蔵関数を使用するように、SQL ステートメント中で使用できます。例えば、以下のようになります。

```
SELECT DB2LS.LSRevNuc(:NucSeq) FROM SYSDUMMY1;
```

この例では、この関数を使用して、ホスト変数からの入力データを反転します。

無効ストリングおよび無効データ・タイプを使用した場合、以下のようなエラー・メッセージが示されます。

```
SQL0443N ルーチン "DB2LS.LSREVNUC" (特定名 "LSREVNUC") が、
診断テキスト "Sequence not valid" とともにエラー SQLSTATE を
返しました。SQLSTATE=38608
```

照会での LSRevNuc ユーザー定義関数の使用例は以下のとおりです。

```
SELECT HSP_H_Seq, db2ls.LSRevNuc(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastN
WHERE BlastSeq='gtaatacgtaggggctagcgcgggcaaactgaagataaagc'
```

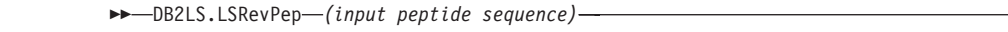
照会が戻す変換されたヌクレオチド・シーケンスは以下のとおりです。

```
HSP_H_SEQ                                          REV_HSP_H_SEQ

-------------------------------------------------- --------------------------
CGCGGGCAAACTGAAGATAAAGC                            CGAAATAGAAGTCAAACGGGCGC

GCGCTAGCCCCCTACGTATTAC                             CATTATGCATCCCCCGATCGCG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG


  5 record(s) selected.
```

関連情報:

- 71 ページの『LSRevNuc ユーザー定義関数』

LSRevPep ユーザー定義関数



input peptide sequence

ペプチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。入力シーケンスは、プロテイン・アルファベットの一部である必要があります。

スキーマ名は DB2LS です。

関数の結果は、実際の長さが 32672 バイト以下であるデータ・タイプ VARCHAR の文字ストリングであり、ペプチド・シーケンスの逆順を示します。

関連情報:

- 73 ページの『LSRevPep ユーザー定義関数の例』
- 69 ページの『LSRevComp ユーザー定義関数』
- 71 ページの『LSRevNuc ユーザー定義関数』

LSRevPep ユーザー定義関数の例

LSRevPep 関数は、ペプチド・シーケンスを受け入れる任意の内蔵関数を使用するように、SQL ステートメント中で使用できます。例えば、以下のようになります。

```
SELECT DB2LS.LSRevPep(:NucSeq) FROM SYSDUMMY1;
```

この例では、この関数を使用して、ホスト変数からの入力データを反転します。

無効ストリングおよび無効データ・タイプを使用した場合、以下のようなエラー・メッセージが示されます。

```
SQL0443N ルーチン "DB2LS.LSREVPEP" (特定名 "LSREVPEP") が、
診断テキスト "Sequence not valid" とともにエラー SQLSTATE を
返しました。SQLSTATE=38608
```

照会での LSRevPep ユーザー定義関数の使用例は以下のとおりです。

```
SELECT  HSP_H_Seq, db2ls.LSRevPep(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastP
WHERE BlastSeq='MLCEIECRALSTAHTRLIHDFEPRDALTYLEGKNIFTEDH'
```

照会が戻す変換されたペプチド・シーケンスは以下のとおりです。

```
HSP_H_SEQ                          REV_HSP_H_SEQ
---------------------------------- ----------------------------------

MLCEIECRALSTAHTRLIHDFEPRDALTYL...  HDETFINKGELYTLADRPEFDHILRTHATS...

RVVSTEHTRLVTDAYPEFSISFTATKN        NKTATFSISFEPYADTVLRTHETSVVR

STAHIRVLRDMVPGDEITCFYGSEFF         FFESGYFCTIEDGPVMDRLVRIHATS

AHTRRCPDHEPRGVITYL                 LYTIVGRPEHDPCRRTHA


  4 record(s) selected.
```

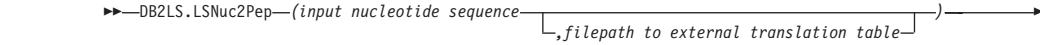
関連情報:

- 72 ページの『LSRevPep ユーザー定義関数』

変換

変換ユーザー定義関数は、ヌクレオチド・シーケンスをペプチド・シーケンスへと変換します。

LSNuc2Pep ユーザー定義関数



input nucleotide sequence
ヌクレオチド・シーケンスを示す有効な文字ストリング表記です。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

filepath to external translation table
カスタマイズした変換表を使用する場合は、その表へのファイル・パス情報を指定してください。パスのストリング値は、255 文字以下にしてください。

スキーマ名は DB2LS です。

関数の結果は、10890 バイト以下であるデータ・タイプ VARCHAR の文字ストリングであり、ペプチド・シーケンスを示します。

入力データは、IUB 文字セットを使用するヌクレオチド・シーケンスです。この関数では、ヌクレオチド・シーケンスの先頭文字から最初のコドンが開始されると想定します。最初のコドンが先頭文字から開始されない場合は、入力シーケンスに SUBSTR 関数を使用してください。

関数の結果は、標準アミノ酸記号を用いたペプチド・シーケンスになります。

機能:

- 入力シーケンスのスペースを削除します。
- 読み取りフレーム外の無関係なヌクレオチドを無視します。
- NULL ヌクレオチド・シーケンスを入力した場合、NULL 出力を戻します。

関連情報:

- 74 ページの『LSNuc2Pep ユーザー定義関数の例』
- 76 ページの『LSTransAllFrames ユーザー定義関数』

LSNuc2Pep ユーザー定義関数の例

ヌクレオチド・シーケンス・データをペプチド・シーケンスに変換するとします。この例では、ヌクレオチド・シーケンスの先頭文字から最初のコドンが開始すると想定します。

この関数は、values ステートメントで呼び出せます。入力は、以下の例のようにヌクレオチド・シーケンス 1 つです。

```
values db2ls.LSNuc2Pep('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT')
```

この結果は、標準アミノ酸記号を用いたペプチド・シーケンスになります。

```
FFLLSSSSYFLCC*C
```

+2 読み取りフレームで変換する場合は、以下を使用してください。

```
values LSNuc2Pep(SUBSTR('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',2))
```

ステートメント中の整数は、コドン検索の開始位置を示します。

この関数を照会の述部として使用する場合は以下のようになります。

```
SELECT  *
  FROM proteindata
  WHERE peptideseq=DB2LS.LSNuc2Pep('TTTTTCTTATTGTCTTCCTCATCG
                                   TATTTCTTATGTTGCTGATGT');
```

照会の結果を表 37 に示します。

表 37. LSNuc2Pep 関数を述部として使用する場合の結果

ID	PROTEINNAME	PEPTIDESEQ
1	proteinA	FSYCLPHRISYVAD

外部変換表を使用してヌクレオチド・シーケンスをペプチド・シーケンスに変換する場合は以下のようになります。最初のパラメーターはヌクレオチド・シーケンス、2 番目のパラメーターは外部変換表へのパスになります。

```
values db2ls.LSNuc2Pep('TTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',
                        'C:\translation.txt')
```

このように特定の変換表を使用した場合の結果は、以下のようなストリングになります。

```
FSYCLPHRISYVAD
```

次の例では 2 つのユーザー定義関数を組み合わせ、関数の別の使用方法を示します。

```
values DB2LS.LSNuc2Pep(DB2LS.LSRevCompNuc('TTT..')
```

上記の例は、以下の照会と同じ結果を戻します。

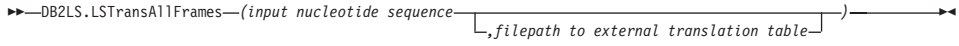
```
select * from table (DB2LS.LSTransAllFrames ('TTT..')) as t
where t.readframe = -1
```

関連情報:

- 71 ページの『LSRevNuc ユーザー定義関数の例』

- 76 ページの『LSTransAllFrames ユーザー定義関数』
- 74 ページの『LSNuc2Pep ユーザー定義関数』

LSTransAllFrames ユーザー定義関数



input nucleotide sequence
ヌクレオチド・シーケンスを示す有効な文字ストリング表記です。入力シーケンスは、IUPAC あいまいコードを含むことができます。文字ストリング表記は、データ・タイプ VARCHAR であり、実際の長さが 32672 バイト以下である必要があります。

filepath to external translation table
カスタマイズした変換表を使用する場合は、その表へのファイル・パス情報を指定してください。パスのストリング値は、255 文字以下にしてください。

スキーマ名は DB2LS です。

LSTransAllFrames ユーザー定義関数を使用して、入力したヌクレオチド・シーケンスから 1 つのペプチド・シーケンス・セットを形成します。これらのペプチド・シーケンスは、入力された 6 つのフレームごとのヌクレオチド・シーケンスの可能な変換形態を示します。この関数は、入力にエラーが含まれる場合や、読み取りフレームが不明な場合に役立ちます。

関数の結果は、2 列の表になります。1 列目は READFRAME という名前で、変換時に使用されるフレームを表します。この列には、変換の開始位置を示す整数値が入ります。負の整数値は、より線の反対側の変換を示します。2 列目は PEPTIDE と呼ばれ、データ・タイプ VARCHAR の文字ストリングが入ります。これは 10890 バイト以下であり、ペプチド・シーケンスを示します。

機能:

- 入力シーケンスのスペースを削除します。
- 読み取りフレーム外の無関係なヌクレオチドを無視します。
- NULL ヌクレオチド・シーケンスを入力した場合、NULL 出力を戻します。

関連情報:

- 76 ページの『LSTransAllFrames ユーザー定義関数』
- 74 ページの『LSNuc2Pep ユーザー定義関数』

LSTransAllFrames ユーザー定義関数

内蔵された変換表を使用して 6 つの読み取りフレームすべてのヌクレオチド・シーケンスを変換するとします。この方法を例示します。

```
SELECT * FROM table(DB2LS.LSTransAllFrames('TTTTTCTTATTGTCTTCCTCATCG
                                           TATTTCTTATGTTGCTGATGT')) as t;
```

この照会は、以下の例のようにペプチドを表に戻します。

表 38. ヌクレオチド・シーケンスの変換の結果

READFRAME	PEPTIDE
1	FFLLSSSSYFLCC*C
2	FSYCLPHRISYVAD
3	FLIVFLIVFLMLLM
-1	TSAT*EIR*GRQ*EK
-2	HQQHKKYDEEDNKK
-3	ISNIRNTMRKTIRK

次の例では、カスタマイズされた変換表を使用して 6 つの読み取りフレームすべてのヌクレオチド・シーケンスを変換します。

```
SELECT * FROM table
  (DB2LS.LSTransAllFrames
   ('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',
    'C:\msvs6\MyProjects\alin_udf\test\files\translation.txt')) as t;
```

関数に指定した入力シーケンスと変換表が同じであるため、結果表は前の例と同じになります。

次の例では 2 つのユーザー定義関数を組み合わせ、関数の別の使用方法を示します。

```
values DB2LS.LSNuc2Pep(DB2LS.LSRevCompNuc('TTT..')
```

上記の例は、以下の照会と同じ結果を戻します。

```
select * from table (DB2LS.LSTransAllFrames ('TTT..')) as t
where t.readframe = -1
```

次の例では、LSTransAllFrames 関数の出力から、特定の読み取りフレームを選択します。

```
SELECT * FROM
  TABLE(db2ls.LSTransAllFrames('TTTTTCTTATTGTCTTCCTCATCG
                               TATTTCTTATGTTGCTGATGT')) AS t
  WHERE t.readframe=-2
```

この照会の結果は次のとおりです。

表 39. Readframe 関数の使用法

READFRAME	PEPTIDE
-2	HQQHKKYDEEDNKK

関連情報:

- 74 ページの『LSNuc2Pep ユーザー定義関数の例』
- 71 ページの『LSRevNuc ユーザー定義関数の例』
- 76 ページの『LSTransAllFrames ユーザー定義関数』

コドン度数表の形式

コドン度数表は、アミノ酸が特定のコドンに逆変換される頻度を示します。
LSPep2ProbNuc ユーザー定義関数はコドン度数表を使用して、入力されたペプチド・シーケンスからヌクレオチド・シーケンスを決定します。

以下に、コドン度数表ファイルの形式について示します。

- 2 つ並んだピリオドは、表の先頭を示します。ピリオドの前のテキストはすべてコメントです。コメントがない場合でも、2 つのピリオドが必要です。
- 表には次の列が含まれます。
  1. Am-Acid: アミノ酸記号を示す 3 文字のコード。
  2. Codon: アミノ酸記号を示すコドン。
  3. Number: 表のコンパイルの対象となった遺伝子中のコドンの発生回数。
  4. x/1000: 遺伝子の変換数 1000 に対するアミノ酸とコドンのペアの予想発生回数。
  5. Fraction: 同義コドン・ファミリーで該当コドンが発生する割合。

この製品のコドン度数表のサンプルは、サブディレクトリー sqllib/samples/lifesci/ls_udfs にあります。

関連情報:

- 46 ページの『LSPep2ProbNuc ユーザー定義関数』
- 78 ページの『コドン度数表の例』

コドン度数表の例

79 ページの図 2 に、コドン度数表のサンプル形式を示します。

```
Am-Acid Codon    Number       x/1000   Fraction   ..

Gly     GGG      198.00       18.34        0.23
Gly     GGA       71.00        6.58        0.08
Gly     GGT       66.00        6.11        0.08
Gly     GGC      527.00       48.81        0.61

Glu     GAG      534.00       49.46        0.88
Glu     GAA       71.00        6.58        0.12
Asp     GAT       31.00        2.87        0.06
Asp     GAC      481.00       44.55        0.94

Val     GTG      396.00       36.68        0.47
Val     GTA       22.00        2.04        0.03
Val     GTT       44.00        4.08        0.05
Val     GTC      384.00       35.57        0.45

Ala     GCG      446.00       41.31        0.39
Ala     GCA       71.00        6.58        0.06
Ala     GCT      116.00       10.74        0.10
Ala     GCC      503.00       46.59        0.44
... (truncated)
```

図 2. コドン度数表の例

関連情報:

- 46 ページの『LSPep2ProbNuc ユーザー定義関数』
- 78 ページの『コドン度数表の形式』

変換表の形式

このトピックでは、LSPep2AmbNuc、LSTransAllFrames、および LSNuc2Pep ライフ・サイエンス・ユーザー定義関数が使用する変換表の形式について説明します。

以下に、コドン度数表ファイルの形式について示します。

- 2 つ並んだピリオドは、表の先頭を示します。ピリオドの前のテキストはすべてコメントです。
- 表の各行には、1 文字のアミノ酸記号、3 文字のアミノ酸名、確定コドン、感嘆符、および、あいまいなコドンが示されます。行中の各文字は空白で区切られます。
- 各コドンおよびアミノ酸記号は、1 ファイルに 1 回のみ表記可能です。
- 終止コドンは、記号「*」に変換されます。
- 小文字で表記されるコドンは、開始コドンを示します。
- その他のコドンは大文字で表記されます。
- 変換する対応アミノ酸記号がないコドンは、「X」に変換されます。

この製品の変換表のサンプルは、サブディレクトリー sqllib/samples/lifesci/ls_udfs にあります。

変換表の例

図 3 に、変換表の例の形式を示します。

標準変換表

```
Symbol   3-letter    Codons                    ! IUPAC     ..

 A        Ala         GCT GCC GCA GCG           ! GCX
 B        Asx                                   ! RAY
 C        Cys         TGT TGC                   ! TGY
 D        Asp         GAT GAC                   ! GAY
 E        Glu         GAA GAG                   ! GAR
 F        Phe         TTT TTC                   ! TTY
 G        Gly         GGT GGC GGA GGG           ! GGX
 H        His         CAT CAC                   ! CAY
 I        Ile         ATT ATC ATA               ! ATH
 K        Lys         AAA AAG                   ! AAR
 L        Leu         TTG TTA CTT CTC CTA CTG   ! TTR CTX YTR     ; YTX
 M        Met         atg                       ! ATG
 N        Asn         AAT AAC                   ! AAY
 P        Pro         CCT CCC CCA CCG           ! CCX
 Q        Gln         CAA CAG                   ! CAR
 R        Arg         CGT CGC CGA CGG AGA AGG   ! CGX AGR MGR     ; MGX
 S        Ser         TCT TCC TCA TCG AGT AGC   ! TCX AGY         ; WSX
 T        Thr         ACT ACC ACA ACG           ! ACX
 V        Val         GTT GTC GTA GTG           ! GTX
 W        Trp         TGG                       ! TGG
 X        Xxx                                   ! XXX
 Y        Tyr         TAT TAC                   ! TAY
 Z        Glx                                   ! SAR
 *        End         TAA TAG TGA               ! TAR TRA         ; TRR
```

図 3. 変換表の例

アクセシビリティー

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) ユーザーは、アクセシビリティー機能を使用することによって、ソフトウェア製品を十分活用できます。DB2 Information Integrator バージョン 8 に備わっている主なアクセシビリティー機能は以下のとおりです。

- すべての機能をマウスの代わりにキーボードを使用して操作できます。
- フォントのサイズと色をカスタマイズできます。
- アラートを表示にするか音声にするかを指定できます。
- DB2 は、Java™ Accessibility API を使用するアクセシビリティー・アプリケーションをサポートします。
- DB2 の資料は、アクセスしやすい形式で提供されています。

キーボードによる入力およびナビゲーション

キーボードだけを使用して、コントロール・センター、データウェアハウス・センター、レプリケーション・センターなどの、DB2 データベース・ツールを操作できます。マウスの代わりに複数のキーまたはキーの組み合わせを使用してほとんどの操作を実行できます。

UNIX ベースのシステムでは、キーボード・フォーカスの置かれている位置が強調表示されます。この強調表示によって、アクティブなウィンドウ領域が示されます。そのウィンドウ領域が、ユーザーのキー・ストロークの対象となります。

アクセスしやすい表示

DB2 データベース・ツールには、視力の弱いユーザーのためにユーザー・インターフェースを拡張し、アクセシビリティー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ー拡張機能には、フォント・プロパティー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

DB2 データベース・ツールでは、「ツール設定」ノートブックを使用して、メニューおよび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。

色覚への非依存

この製品の機能を使用するために、ユーザーは色を識別する必要はありません。

代替アラート・キュー

「ツール設定」ノートブックを使用して、アラートを音声にするか、表示にするかを指定できます。

支援テクノロジーとの互換性

DB2 Information Integrator インターフェースは、身体に障害を持つ人々によって使用されているスクリーン・リーダーおよび他の支援テクノロジーに採用されている Java Accessibility API をサポートしています。

入手可能な資料

DB2 ファミリー製品の資料は HTML 形式で入手可能です。資料は、ご使用のブラウザーに設定されている表示設定に従って表示することができます。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。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、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で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

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 Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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すが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

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にお願いします。

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、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. _*西暦年*_. All rights reserved.

商標

以下は、IBM Corporation の商標です。

IBM
AIX
DB2
Domino
Informix
Lotus
Lotus Notes
QuickPlace
WebSphere

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

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。

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[カ行]

カスタム関数
  BioRS 3, 14, 29
コドン度数表 78

[サ行]

サンプル
  照会
    BioRS 19
正規表現のサポート 60

[ハ行]

変換表 79, 80

[ヤ行]

ユーザー定義関数 (UDF)
  ライフ・サイエンス 39

[ラ行]

ライフ・サイエンス・ユーザー定義関数
  除去 41
  登録 40
  リスト 39

B

BioRS
  カスタム関数
    使用 14
BioRS *(続き)*
  カスタム関数 *(続き)*
    登録 3
  照会例 19
  統計情報、維持 25
  ニックネーム、変更 29
  フェデレーテッド (連合)・システムへの追加
    カスタム関数の登録 29
    ニックネーム例 9
    CREATE NICKNAME ステートメント 35
    CREATE SERVER ステートメント 36
    CREATE USER MAPPING ステートメント 6, 38

C

CREATE NICKNAME ステートメント
  BioRS 9, 35
CREATE SERVER ステートメント
  BioRS 36
CREATE USER MAPPING ステートメント
  BioRS 6, 38

G

GeneWise 60, 61, 63

L

LSBarCode ユーザー定義関数 63, 64
LSDeflineParse ユーザー定義関数 49, 55
LSGeneWise ユーザー定義関数 61, 63
LSMultiMatch ユーザー定義関数 65, 66
LSMultiMatch3 ユーザー定義関数 67
LSNuc2Pep ユーザー定義関数 74
LSPatternMatch ユーザー定義関数 56
LSPep2AmbNuc ユーザー定義関数 42, 44, 45
LSPep2ProbNuc ユーザー定義関数 46, 47, 48
LSPrositePattern ユーザー定義関数 58, 59
LSRevComp ユーザー定義関数 69, 70
LSRevNuc ユーザー定義関数 71
LSRevPep ユーザー定義関数 72, 73, 76

U

UDF (ユーザー定義関数)
  ライフ・サイエンス 39

IBM と連絡を取る

お住まいの国または地域の IBM 事業所を探すには、 www.ibm.com/planetwide の IBM Directory of Worldwide Contacts をお調べください。

製品情報

DB2 Information Integrator に関する情報は、電話または Web で入手可能です。

Web をご利用の場合は、www.ibm.com/software/data/integration にアクセスしてください。このサイトには、技術ライブラリー、資料の注文方法、クライアント・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。

お住まいの国または地域の IBM 事業所を探すには、 www.ibm.com/planetwide の IBM Directory of Worldwide Contacts をお調べください。

Printed in Japan

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