IBM DB2 Information Integrator

数据源配置指南附录：BioRS 包装器和生命科学用户定义的函数

版本 8

IBM DB2 Information Integrator

数据源配置指南附录：BioRS 包装器和生命科学用户定义的函数

版本 8

在使用本资料及其支持的产品之前，请阅读第 79 页的『声明』中的一般信息。

本文档包含 IBM 的专利信息。它在许可证协议下提供，并受版权法保护。本出版物包含的信息不包括任何产品保证，且本手册提供的任何声明不应作如此解释。

可以在线方式或通过您当地的 IBM 代表订购 IBM 出版物。

- 要以在线方式订购出版物，可访问“IBM 出版物中心”（IBM Publications Center），网址为 www.ibm.com/shop/publications/order
- 要查找您当地的 IBM 代表，可访问“IBM 全球联系人目录”（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

目录

第 1 章 配置对 BioRS 数据源的存取

本章说明 BioRS 是什么、如何将 BioRS 数据源添加至联合系统并列示与 BioRS 包装器相关联的错误消息。

什么是 BioRS?

BioRS 是由 Biomax Informatics 开发的查询和检索系统。可以使用 BioRS 来从包括平面文件和关系数据库在内的多个数据源检索信息。您通常将公用数据（如 SwissProt 和 GenBank）作为平面文件下载到 BioRS 系统。BioRS 可以将公用数据源和私有数据源（例如，由您所在的组织维护的私有数据库）集成到公共环境中。

数据源在集成到 BioRS 系统中之后，它被称为*数据银行*。包含在每个数据银行条目中的元素统称*模式*。只有在 BioRS 系统中建立了索引的数据银行元素才可以在 BioRS 查询中使用。可以在数据银行中的条目之间建立关系，以便可以在 BioRS 系统中将数据银行链接到一起。

BioRS 数据银行可以具有父子关系（数据银行可以嵌套）。在这样的关系中，子数据银行包含名为 PARENT 的 “引用” 数据类型元素。PARENT 元素引用父数据银行的 _ID_ 元素。除了存在这个预定义的 PARENT 元素之外，嵌套的数据银行与不嵌套的数据银行包含相同的数据。

BioRS 提供了基于 Web 的界面，此界面使用户能够对 BioRS 数据银行中的数据运行查询。BioRS 包装器与 BioRS 的基于 Web 的界面使用相同的应用程序编程接口（API）来运行查询。

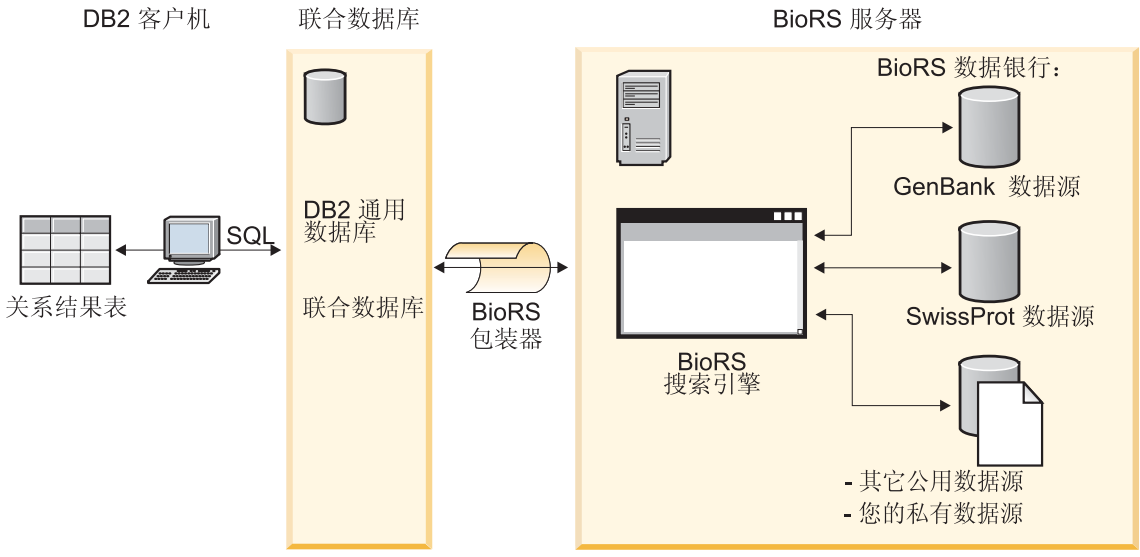


图 1. BioRS 包装器的工作方式

用户或应用程序使用 SQL 语句来从客户机提交查询。然后，将查询发送到安装了 BioRS 包装器的联合系统。根据查询的构造方式的不同，DB2® 通用数据库和 BioRS 服务器都可用来处理查询。BioRS 服务器与联合系统可以位于不同的计算机上。对于每个查询，联合系统都必须向 BioRS 服务器提供认证信息。此信息可以是用户标识和密码信息，也可以是未经认证的指示（通常是来宾帐户）。

BioRS 包装器与 BioRS V5.0.14 配合工作。

有关 BioRS 产品的详细信息，请参阅 Biomax Web 站点，网址为: http://www.biomax.com。

相关任务:

- 第 2 页的『将 BioRS 添加至联合系统』

相关参考:

- 第 17 页的『BioRS 包装器 - 示例查询』

将 BioRS 添加至联合系统

可以通过注册定制函数和 BioRS 包装器来将 BioRS 数据源与联合服务器配合使用。接着，注册相应的 BioRS 服务器、用户映射和昵称以使联合服务器能够检索和处理 BioRS 数据。

可以从 “DB2 控制中心” 或从 DB2 命令行处理器运行 SQL 语句。将 BioRS 添加至联合系统之后，就可以对 BioRS 数据源运行查询了。

过程:

要将 BioRS 数据源添加至联合服务器:

1. 通过使用 CREATE FUNCTION 语句注册定制函数。
2. 通过使用 CREATE WRAPPER 语句注册 BioRS 包装器。
3. 可选: 设置 DB2_DJ_COMM 环境变量以改进查询性能。
4. 通过使用 CREATE SERVER 语句注册 BioRS 服务器。
5. 可选: 通过使用 CREATE USER MAPPING 语句注册经授权的用户。
6. 通过使用 CREATE NICKNAME 语句注册昵称。
7. 可选: 更新 BioRS 列的基数统计信息。

相关任务:

- 第 3 页的『为 BioRS 包装器注册定制函数』
- 第 4 页的『注册 BioRS 包装器』
- 第 5 页的『为 BioRS 包装器设置 DB2_DJ_COMM 概要文件变量』
- 第 6 页的『为 BioRS 数据源注册服务器』
- 第 6 页的『为 BioRS 数据源注册用户映射』
- 第 7 页的『为 BioRS 数据源注册昵称』
- 第 10 页的『更新 BioRS 列基数统计信息』

为 BioRS 包装器注册定制函数

为 BioRS 包装器注册定制函数是将 BioRS 添加至联合系统这一大型任务的一部分。注册定制函数之后，必须注册包装器。

可以使用样本文件 create_function_mappings.ddl 来注册定制函数。此文件位于 sqllib/samples/lifesci/biors 目录中。create_function_mappings.ddl 文件包含每个定制函数的定义。可以运行此 DDL 文件来为每个安装了 BioRS 包装器的 DB2 数据库注册定制函数。

先决条件:

- 必须使用模式名 BioRS 来注册 BioRS 包装器的所有定制函数。
- 必须为每个安装了 BioRS 包装器的 DB2 数据库将每个定制函数注册一次。

过程:

要注册定制函数，请发出带有 AS TEMPLATE 关键字的 CREATE FUNCTION 语句。

每个函数的全限定名是 BioRS.<函数名>。

以下示例将注册一个版本的 CONTAINS 函数:

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

在这一系列任务中的下一个任务是注册适当的 BioRS 包装器。

相关任务:

- 第 4 页的『注册 BioRS 包装器』

相关参考:

- 『CREATE FUNCTION (Sourced or Template) statement』（在 *SQL Reference, Volume 2* 中）
- 第 13 页的『定制函数和 BioRS 查询』
- 第 17 页的『BioRS 包装器 – 示例查询』
- 第 28 页的『定制函数表 – BioRS 包装器』

注册 BioRS 包装器

注册 BioRS 包装器是将 BioRS 添加至联合系统这一大型任务的一部分。必须注册包装器才能存取数据源。包装器是联合服务器用来与数据源通信以及从数据源中检索数据的一种机制。包装器是作为库文件安装在系统上的。表 1 列示了缺省的 BioRS 库文件以及每个文件支持的操作系统。

表 1. BioRS 库文件和受支持的操作系统

BioRS 库文件	操作系统
libdb2lsbiors.a	IBM AIX V4.3.3 或更高版本
db2lsbiors.dll	Microsoft Windows NT V4 Microsoft Windows 2000 Microsoft Windows XP

过程:

要注册 BioRS 包装器，请发出 CREATE WRAPPER 语句。

例如，在 AIX 上，要根据缺省库文件 `libdb2lsbiors.a` 来注册名为 `wrap_biors` 的 BioRS 包装器，请发出以下语句:

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

相关任务:

- 『检查非关系包装器和生命科学用户定义的函数库』（在《*DB2 Information Integrator 安装指南*》中）
- 第 5 页的『为 BioRS 包装器设置 DB2_DJ_COMM 概要文件变量』

相关参考:

- 『CREATE WRAPPER statement』（在 *SQL Reference, Volume 2* 中）

为 BioRS 包装器设置 DB2_DJ_COMM 概要文件变量

为 BioRS 包装器设置 DB2_DJ_COMM DB2 概要文件变量是将 BioRS 添加至联合系统这一大型任务的一部分。为了在存取 BioRS 数据源时提高性能，可以选择设置 DB2_DJ_COMM DB2 概要文件变量。此变量指定联合服务器在进行初始化后是否装入包装器。

在数据库启动期间，当联合服务器装入包装器库时，处理器使用率将提高。为了避免过度使用，应当只指定要存取的库。

过程:

要设置 DB2_DJ_COMM DB2 概要文件变量，对与您在相关联的 CREATE WRAPPER 语句中指定的包装器相应的包装器库发出 **db2set** 命令。

例如:

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

确保等号（=）两边都没有空格。

在这一系列任务中的下一个任务是为 BioRS 注册服务器。

相关概念:

- 『环境变量和概要文件注册表』（在《*管理指南: 实现*》中）

相关任务:

- 第 6 页的『为 BioRS 数据源注册服务器』

相关参考:

- 『db2set - DB2 Profile Registry Command』（在 *Command Reference* 中）

为 BioRS 数据源注册服务器

为 BioRS 数据源注册服务器是将 BioRS 添加至联合系统这一大型任务的一部分。在注册包装器之后，必须注册相应的服务器。

过程:

要向联合系统注册 BioRS 服务器，请发出 CREATE SERVER 语句。

例如:

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

相关任务:

- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 第 34 页的『CREATE SERVER 语句选项 – BioRS 包装器』

为 BioRS 数据源注册用户映射

注册用户映射是将 BioRS 添加至联合系统这一大型任务的一部分。根据帐户访问方法或 BioRS 系统中使用的方法的不同，可能不需要创建用户映射。

- 如果 BioRS 服务器是针对所有用户帐户的来宾访问配置的，则不需要在 DB2 Information Integrator 中创建用户映射。
- 如果 BioRS 服务器配置为使用标识和密码来认证用户帐户，则必须在联合数据库中为必须使用 BioRS 包装器的帐户创建用户映射。
- 如果 BioRS 服务器配置为既使用来宾也使用经过认证的用户帐户，则必须在联合数据库中为必须使用 BioRS 包装器的帐户创建经过认证的用户帐户的用户映射。

用户映射可以用来认证使用 BioRS 包装器查询 BioRS 数据源的用户或应用程序的存取权。如果用户或应用程序对已注册的 BioRS 昵称提交 SQL 查询，但是没有为该用户或应用程序定义任何用户映射，则 BioRS 包装器使用缺省用户标识和密码来尝试从远程 BioRS 服务器检索数据。如果正在查询的数据银行要求认证，则可能返回错误消息。

为了确保将正确的用户标识和密码传递给 BioRS 服务器，请在联合数据库中为经授权可搜索 BioRS 数据源的用户创建用户映射。创建用户映射时，密码以加密格式存储在联合数据库系统目录表中。

过程:

要注册 BioRS 用户映射，使用 CREATE USER MAPPING 语句。

例如，以下 CREATE USER MAPPING 语句将用户 Charlie 映射至 Biors_Server1 服务器上的用户 Charlene。

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

还可以定义您自己的用户映射。在以下示例中，USER 是标识当前用户的关键字，而不是 USER 的用户名。

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

在这一系列任务中的下一个任务是为 BioRS 包装器注册昵称。

相关任务:

- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 『CREATE USER MAPPING statement』（在 *SQL Reference, Volume 2* 中）
- 第 35 页的『CREATE USER MAPPING 语句选项 – BioRS 包装器』

为 BioRS 数据源注册昵称

为 BioRS 数据源注册昵称是将 BioRS 添加至联合系统这一大型任务的一部分。在注册服务器之后，必须为每一个要存取的 BioRS 数据源注册昵称。当您在查询中引用 BioRS 数据源时就要使用昵称。

要点: 在将数据源集成到 BioRS 系统中之后，它在 BioRS 中就被称为*数据银行*。BioRS 数据银行等同于联合系统中的昵称。

先决条件:

- 如果 BioRS 数据银行名不符合 DB2 联合语法，则注册昵称时就必须使用 REMOTE_OBJECT 昵称选项。
- 如果 BioRS 元素名称不符合 DB2 联合语法，则注册昵称时就必须使用 ELEMENT_NAME 列选项。

限制:

不要使用 BioRS AllText 元素作为昵称的第一列。可以在任何其它列位置（例如，作为第二列或作为第三列）中使用 BioRS AllText 元素。

过程:

要注册 BioRS 昵称，使用 CREATE NICKNAME 语句。

联合昵称直接等同于 BioRS 数据银行。创建联合昵称时，您定义昵称列列表。指定的昵称列必须与特定 BioRS 数据银行格式的元素相应。BioRS 为元素定义 5 种可能的数据类型：文本、数字、日期、作者和引用。这些数据类型只能映射到 CLOB、CHAR 或 VARCHAR 联合系统数据类型。

为 BioRS 数据银行注册昵称的最简单的方法是对昵称指定与 BioRS 数据银行相同的名称。例如:

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

下层 BioRS 数据银行 SwissProt 是昵称的名称。

使用这个简单的 CREATE NICKNAME 语法将限制您对每个 DB2 模式只能使用一个系列的昵称。可以通过指定 REMOTE_OBJECT 选项来使用其它名称。此昵称选项指定要与昵称相关联的 BioRS 对象类型的名称。在 REMOTE_OBJECT 选项中指定的名称确定了模式以及昵称的 BioRS 数据银行。REMOTE_OBJECT 选项还指定昵称与其它昵称的关系。

以下示例显示与上一个示例相同的一组昵称特征，但是更改了昵称名称，并使用 REMOTE_OBJECT 选项来指定正在为其定义昵称的 BioRS 数据银行:

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

下层 BioRS 数据银行是 SwissProt，昵称的名称是 NewSP。

相关概念:

- 第 24 页的『BioRS 统计信息』

相关任务:

- 第 25 页的『更新 BioRS 昵称基数统计信息』

相关参考:

- 第 27 页的『BioRS AllText 元素』
- 第 9 页的『CREATE NICKNAME 语句 – BioRS 包装器的示例』
- 第 32 页的『CREATE NICKNAME 语句语法 – BioRS 包装器』
- 第 27 页的『与改变昵称相关的注意事项 – BioRS 包装器』

CREATE NICKNAME 语句 – BioRS 包装器的示例

本主题提供了示例来说明如何使用 CREATE NICKNAME 语句来为 BioRS 包装器注册昵称。

示例 1:

以下示例说明如何为不符合 DB2 Information Integrator 语法的远程 BioRS 数据银行创建昵称:

```
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');
```

此昵称的名称是 SwissFT。表列是 ID、ALLTEXT、ENTRYDATE 和 FtLength。对 ID 列指定了 ELEMENT_NAME 列选项。当 BioRS 元素的名称不符合列名的有效 DB2 联合语法时，必须指定 ELEMENT_NAME 选项。在此示例中，BioRS 元素 _ID_ 符合 DB2 联合语法，但 _ID_ 对于 DB2 Information Integrator 用户而言是有可能会令人混淆的名称。名称 ID 简单易懂。通常，在下列情况下使用 ELEMENT_NAME 选项:

- 当 BioRS 元素名称不符合有效的 DB2 联合语法时
- 当 BioRS 元素名称的区分大小写性不符合所建立的 DB2 联合系统标准时
- 当 BioRS 元素名称对 DB2 Information Integrator 用户可能不明显时

另外，REMOTE_OBJECT 选项用于指定与昵称等同的 BioRS 数据银行名。当 BioRS 数据银行的名称不符合有效的 DB2 联合语法时，必须指定 REMOTE_OBJECT 选项。在此示例中，数据银行名 “SwissProt.Features” 不符合有效的 DB2 联合语法。通常，在下列情况下使用 REMOTE_OBJECT 选项:

- 当 BioRS 数据银行名的区分大小写性不符合所建立的 DB2 联合系统标准时
- 当 BioRS 数据银行名不符合有效的 DB2 联合语法时
- 当 BioRS 数据银行名对 DB2 Information Integrator 用户可能不明显时

示例 2:

以下示例说明如何为使用链接到另一个 BioRS 数据银行的 BioRS 数据银行的表创建昵称:

```
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');
```

此昵称的名称是 SwissFT2。表列是 ID、ALLTEXT、FtKey、FtLength、FtDescription 和 Parent。对 ID 列指定了 ELEMENT_NAME 列选项。REMOTE_OBJECT 选项用于指定与昵称相应的 BioRS 数据银行名。

另外，Parent 列使用 REFERENCED_OBJECT 选项。必须对与 BioRS 引用数据类型元素相应的列指定此选项。REFERENCED_OBJECT 选项指定列所引用的 BioRS 数据银行的名称。在此情况下，Parent 元素引用 BioRS SwissProt 数据银行。

相关任务:

- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 第 32 页的『CREATE NICKNAME 语句语法 - BioRS 包装器』
- 第 27 页的『与改变昵称相关的注意事项 - BioRS 包装器』

更新 BioRS 列基数统计信息

要更新联合系统中的 BioRS 列基数统计信息，必须修改 SYSSTAT.COLUMNS 目录视图。

通过维护 BioRS 列的正确基数统计信息，优化器和 BioRS 包装器在处理查询期间就能够选择性能最佳的数据存取方案。

您可以可选地更新 BioRS 列基数统计信息来作为将 BioRS 添加至联合系统这一大型任务的一部分。当您想要改进 BioRS 数据源的查询性能时，也可以更新 BioRS 列基数统计信息。

限制:

不要使用此过程来更新与 BioRS _ID_ 元素相应的列的基数统计信息。必须使用另一个过程来更新与 BioRS _ID_ 元素相应的列的基数统计信息。

过程:

要更新 BioRS 列基数统计信息，请使用以下语法来发出 UPDATE 语句:

```
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*> 是要更新其基数统计信息的列的名称。
- <*nickname-schema*> 是与使用了指定的列的昵称相关联的模式的名称。
- <*nickname-name*> 是使用了指定的列的昵称的名称。

因为必须检索昵称中指定的数据银行的所有条目，所以此查询可能需要运行几分钟。

如果列可以包含多个值（例如，具有 SwissProt 数据库格式的 PublicationYear 元素），则计算就会变得太复杂而导致无法使用 SQL 查询。对于这样的列，必须手工计算基数值，然后更新 SYSSTAT.COLUMNS 目录视图。要计算基数值，请将列中的相异值的数目除以每行的值的平均数目。计算得出的基数值不能大于表的基数。

示例:

假定某个昵称带有三行。这三行的 PublicationYear 列的值为:

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

共有 9 个相异值，各行中的值的平均数目是 4。这个 PublicationYear 列的基数为 9/4 或 3（2.25 舍入到下一个最大整数）。在计算了基数之后，就可以使用以下 UPDATE 语句来更新 SYSSTAT.COLUMNS 目录视图:

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

- *3* 是列基数值。
- <*nickname-schema*> 是与使用了指定的列的下层昵称相关联的模式的名称。
- <*nickname-name*> 是使用了指定的列的下层昵称的名称。
- <*column-name*> 是要更新其基数统计信息的列的名称。

相关概念:

- 第 24 页的『BioRS 统计信息』

相关任务:

- 第 25 页的『更新 BioRS 昵称基数统计信息』
- 第 26 页的『更新 BioRS _ID_ 列基数』

有关优化 BioRS 包装器性能的准则

本主题提供有关使用 BioRS 包装器时如何优化查询性能的准则。

最大程度地减少在搜索引擎之间传输的数据量。

联合环境使用两个查询引擎。对于 BioRS 包装器，这些查询引擎是 DB2® 通用数据库和 BioRS。DB2 引擎处理对昵称列指定的谓词（关系运算符，如 =、BETWEEN、LIKE 和 <>）。BioRS 引擎处理使用 BioRS 包装器的 4 个定制函数指定的谓词。

要最大程度地减少在两个搜索引擎之间传输的数据量，请将查询构建成尽可能将数据处理下推到 BioRS 系统。

如果需要在查询中执行连接操作，则请利用 BioRS 数据银行中已经存在的任何父子关系并尽可能地执行等值连接操作。等值连接操作是在 BioRS 中处理的，这还最大程度地减少了在 DB2 和 BioRS 查询引擎之间传输的数据量。

注意: 不要中断对 BioRS 进行的 DB2 Information Integrator 查询（例如，在命令行处理器中使用 **Ctrl-D** 或 **Ctrl-Z**，或停止应用程序）。将查询中断会导致“死亡”进程继续在 BioRS 服务器上运行。这些“死亡”进程将迅速地降低 BioRS 和 DB2 Information Integrator 系统性能。如果有足够的此类“死亡”进程在运行的话，在 DB2 Information Integrator 查询处理期间就会发生意外的错误。例如，当预期会返回行的时候，有效的查询可能返回 0 行。在极端的情况下，BioRS、DB2 Information Integrator 或两个产品都会停止或异常结束。

在联合环境中维护 BioRS 统计信息。

在联合系统中，联合数据库依靠带有昵称的对象的目录统计信息来优化查询处理。维护关于 BioRS 数据源的当前统计信息对于优化 BioRS 包装器性能而言是必需的。如果定义昵称所基于的远程对象的统计数据或结构特征已更改，则必须在联合系统中更新相应的昵称列基数统计信息。

要优化 BioRS 包装器性能，请在 DB2 Information Integrator 中定期执行这些更新。

相关概念:

- 『调整查询处理』（在*《联合系统指南》*中）
- 第 16 页的『BioRS 包装器的等值连接谓词』

- 第 24 页的『BioRS 统计信息』

相关参考:

- 第 13 页的『定制函数和 BioRS 查询』
- 第 17 页的『BioRS 包装器 - 示例查询』

定制函数和 BioRS 查询

联合环境使用两个查询引擎。对于 BioRS 包装器，这些查询引擎是 DB2 通用数据库和 BioRS。可以通过使用 4 个 BioRS 定制函数来指定将谓词下推至 BioRS 引擎，这些定制函数是:

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

这四个定制函数是在 BioRS 模式中注册的。必须使用 BioRS 模式来引用这些函数。

定制函数 BIORS.CONTAINS、BIORS.CONTAINS_LE 和 BIORS.CONTAINS_GE 需要搜索项列自变量和查询文本自变量。以下示例显示了 BIORS.CONTAINS 语句:

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

搜索项列自变量的值必须引用已建立索引的 BioRS 列。使用未建立索引的列将产生错误消息 SQL30090N（“操作对应用程序执行环境无效”）。

查询项自变量的值只能是文字、主变量或列引用。不能使用算术或字符串并置。并且，即使将所使用的搜索项列定义为允许空值，查询项自变量的值也不能为 NULL。

查询项自变量的大小写无关紧要。

查询项自变量的有效数据类型和格式取决于所使用的搜索项列的 BioRS 数据类型。BioRS 定义了 5 种可能的数据类型: 文本、作者、日期、数字和引用。第 14 页的表 2 列示了 BioRS 数据类型和每种数据类型的有效函数查询项。

表 2. BioRS 数据类型和有效的定制函数查询项

搜索项列的数据类型	有效的查询项	格式
文本	VARCHAR() 或 CHAR()	BioRS 文本项，包括通配符。
作者	VARCHAR() 或 CHAR()	具有格式“<last>, <init>”的 BioRS 作者引用。“<last>”是作者的姓。“<init>”是作者姓名的首字母（不带句点）。可以接受逗号与姓名首字母之间的空格。 另外，也可以单独指定 <last> 而不带逗号或姓名首字母。
日期	VARCHAR()、CHAR()、DATE 或 TIMESTAMP	如果是字符串，则具有 DB2 格式日期 yyyy/mm/dd。
数字	VARCHAR() 或 CHAR()、INTEGER、SMALLINT、BIGINT REAL、DOUBLE 和 DECIMAL	DB2 格式数字。
引用	VARCHAR() 或 CHAR()	BioRS 文本项。

BioRS 数据类型搜索项列与查询项自变量的所有其它组合都将产生错误消息 SQL30090N（“操作对应用程序执行环境无效”）。只可以使用表 2 中显示的组合。

文本、作者和引用数据类型搜索项列的查询项自变量必须与 BioRS 查询语言模式相匹配。在 BioRS 中，查询项自变量可以包含字母数字字符串和通配符。BIORS.CONTAINS 函数支持两个通配符：**?**（问号）和 *****（星号）。

? 通配符与单个字符相匹配。例如，谓词 `BioRS.CONTAINS (description, 'bacteri?')=1` 与项 bacteria 相匹配，但不与项 bacterial 相匹配。

***** 通配符与零个或更多个字符相匹配。例如，谓词 `BioRS.CONTAINS (description, 'bacteri*')=1` 与项 bacteri、bacteria 和 bacterial 相匹配。

有关 BioRS 查询语言模式的详细信息，请参阅 BioRS 文档。

可以对所有 BioRS 列类型指定 BIORS.CONTAINS 函数。

只能对其下层 BioRS 数据类型是数字或日期的列指定 BIORS.CONTAINS_GE 和 BIORS.CONTAINS_LE 定制函数。BIORS.CONTAINS_GE 函数选择符合以下条件的行：列包含的值大于或等于由查询项自变量表示的值。BIORS.CONTAINS_LE 函数选择符合以下条件的行：列包含的值小于或等于由查询项自变量表示的值。

BIORS.CONTAINS、BIORS.CONTAINS_GE 和 BIORS.CONTAINS_LE 函数返回整数结果。当在谓词中使用这三个 CONTAINS 函数的任何之一时，必须使用 **=** 或 **<>** 运算符来将返回值与值 1 作比较。例如:

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

具有 `NOT (BioRS.Contains (col,value) = 1)` 格式的表达式等同于 `BioRS.CONTAINS (col,value) <> 1`。

可以通过发出 BIORS.SEARCH_TERM 函数来运行采用别的方法不可能进行的查询。可以使用此函数来指定使用 BioRS 格式的搜索项。BIORS.SEARCH_TERM 函数需要两个自变量。第一个自变量是对项将要应用于的昵称的 _ID_ 列的引用。第二个自变量是包含不带数据银行名的项的字符串。

以下示例从 MyEMBL 数据银行中选择 SeqLength 元素包含大于或等于 100 的值的条目的所有列。

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

以下示例从 Swiss 昵称中选择 MolWeight 元素值大于或等于 100368 的 MolWeight 列。

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

如果指定 BIORS.SEARCH_TERM 函数，就不能在查询中使用任何其它定制函数。但是，可以在同一个查询中使用 BIORS.CONTAINS、BIORS.CONTAINS_GE 和 BIORS.CONTAINS_LE 函数的任意组合。

相关概念:

- 『下推分析』（在*《联合系统指南》*中）
- 第 12 页的『有关优化 BioRS 包装器性能的准则』
- 第 16 页的『BioRS 包装器的等值连接谓词』

相关任务:

- 第 3 页的『为 BioRS 包装器注册定制函数』

相关参考:

- 第 17 页的『BioRS 包装器 - 示例查询』
- 第 28 页的『定制函数表 - BioRS 包装器』

BioRS 包装器的等值连接谓词

可以使用 4 个 BioRS 定制函数来指定 BioRS 引擎的谓词，但有一个例外情况。在查询期间执行等值连接操作就是这种例外情况。*连接*操作涉及根据匹配列值来从两个或更多个表检索数据。*等值连接*是连接条件具有“表达式 = 表达式”格式的连接操作。对于 BioRS 查询，等值连接项必须包含一个数据银行的 _ID_ 元素和另一个数据银行的“引用”类型元素。

示例:

此示例显示样本昵称定义以及使用样本昵称的等值连接查询。

假定要查询两个 BioRS 数据银行 SwissProt 和 SwissProt.features。SwissProt.features 数据银行是 SwissProt 数据银行的子数据银行，它包含名为 Parent 的元素。Parent 元素包含对 SwissProt 的 _ID_ 元素标识的条目的引用。您为两个数据银行注册两个昵称定义。

昵称定义 1:

```
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');
```

昵称定义 2:

```
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');
```

以下查询在等值连接中引用了这两个昵称:

```
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;
```

在上一个查询中，将两个谓词应用于 tc600sprot 昵称（SwissProt 数据银行）。这两个谓词对包含 anopheles 项并且发布年份为 1997 的行进行过滤。一个谓词应用于 tc600feat 昵称（SwissProt.features 数据银行），它对 FtKey 元素包含 signal 项的那些行进行过滤。这两个昵称是使用 f.Parent = s.ID 项进行连接的。

最终结果集只包含满足这些条件并且 features 条目引用 SwissProt 数据银行中的匹配条目的行。

相关概念:

- 第 12 页的『有关优化 BioRS 包装器性能的准则』

相关参考:

- 第 13 页的『定制函数和 BioRS 查询』
- 第 17 页的『BioRS 包装器 - 示例查询』

BioRS 包装器 - 示例查询

本主题提供了数个使用昵称 swiss 和 swissft 的样本查询。

昵称 swiss 是使用以下 CREATE NICKNAME 语句注册的:

```
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');
```

昵称 swissft 是使用以下 CREATE NICKNAME 语句注册的:

```
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');
```

第 19 页的表 3 中的查询和结果举例说明了可以如何构建查询以优化联合系统与 BioRS 服务器之间的工作负载。

表 3. 生成完全相同的结果的不同查询的样本

查询	结果
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 中的两个查询产生相同的结果。但是，第一个查询的运行速度比第二个查询快得多。第一个查询使用 BIORS.CONTAINS 函数来指定输入谓词。因此，BioRS 选择 swissprot 数据银行中的数据并接着将所选数据传递到 DB2 Information Integrator。在第二个查询中，直接对 Swiss 昵称指定输入谓词 LIKE。因此，BioRS 将整个 swissprot 数据银行传输到 DB2 Information Integrator。在传输数据银行内容之后，DB2 Information Integrator 接着选择数据。

表 4 中的查询和结果显示了在 BIORS.CONTAINS 函数中使用通配符。尽管使用了不同的通配符，但是表 4 中的所有查询结果都是完全相同的。

表 4. 在 BIORS.CONTAINS 函数中使用通配符的样本查询

查询	结果
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.

表 4. 在 BIORS.CONTAINS 函数中使用通配符的样本查询 (续)

查询	结果
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 中的查询和结果显示了可以如何使用 BIORS.CONTAINS 函数来存取 BioRS 作者数据类型元素中的信息。

表 5 中的所有查询的语法几乎完全相同。唯一的区别是查询项中存在或不存在名字首字母以及名字与姓氏首字母之间的空格数目。

表 5. 存取 BioRS 作者数据类型列的样本查询

查询	结果
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.
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.

第 21 页的表 6 中的查询和结果举例说明可以如何使用 BIORS.CONTAINS 函数来存取 BioRS 日期类型元素中的信息。

当 BioRS 日期类型字段包含日期序列时，结果可以包含额外的信息，如表 6 的第二个示例所示。BioRS 数字数据类型元素（日期和数字）可以包含多个值。因此，对 BioRS 日期或数字元素运行的查询的结果也可以包含多个值。如果包含多个值，则始终用空格将值分隔开。

表 6. 存取 BioRS 日期数据类型列的样本查询

查询	结果
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 中的查询和结果显示了可以如何使用 BIORS.CONTAINS_LE 和 BIORS.CONTAINS_GE 函数。

表 7. 使用 BIORS.CONTAINS_LE 和 BIORS.CONTAINS_GE 函数的样本查询

查询	结果
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.

表 7. 使用 BIORS.CONTAINS_LE 和 BIORS.CONTAINS_GE 函数的样本查询 (续)

查询	结果
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 中的查询和结果显示了可以如何使用 BIORS.SEARCH_TERM 函数来指定使用 BioRS 格式的搜索项。

表 8. 使用 BIORS.SEARCH_TERM 函数的样本查询

查询	结果
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.
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.

表 8. 使用 BIORS.SEARCH_TERM 函数的样本查询 (续)

查询	结果
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.

以下查询显示如何使用关系谓词来构成具有父子关系的两个数据银行之间的等值连接:

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

查询结果如下:

```
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.
```

在前面的查询结果中，100K_RAT 记录是 9 个子记录（100K_RAT.1 到 100K_RAT.9）的父记录。

相关概念:

- 第 12 页的『有关优化 BioRS 包装器性能的准则』
- 第 16 页的『BioRS 包装器的等值连接谓词』

相关参考:

- 第 13 页的『定制函数和 BioRS 查询』

- 第 9 页的『CREATE NICKNAME 语句 – BioRS 包装器的示例』
- 第 32 页的『CREATE NICKNAME 语句语法 – BioRS 包装器』

BioRS 统计信息

在联合系统中，联合数据库依靠带有昵称的对象的目录统计信息来优化查询处理。这些统计信息是使用 CREATE NICKNAME 语句创建昵称时从 BioRS 数据源中检索到的。联合数据库验证数据源中的对象是否存在，然后尝试收集现有数据源的统计数据。信息是从数据源目录中读取的，并且放入联合服务器上的 DB2® 联合数据库系统目录。

对于 BioRS 数据源，关键的统计信息包括:

- 昵称的基数。对于 BioRS 数据源，昵称基数等同于相应的 BioRS 数据银行中的条目数目。
- 与 BioRS _ID_ 元素相应的列的基数。此列的基数必须与引用列的昵称的基数相匹配。
- BioRS 包装器可能需要使用的所有列的基数。

必须维护关于 BioRS 数据源的当前统计信息以优化 BioRS 包装器的性能。如果定义昵称所基于的远程对象的统计数据或结构特征发生更改，则必须在联合系统中更新相应的基数统计信息。基数统计信息存储在 SYSSTAT.TABLES 目录视图和 SYSSTAT.COLUMNS 目录视图中。

执行下列任务来在联合系统中维护 BioRS 基数统计信息:

1. 确定必需的昵称的基数统计信息（如果有必要的话）。
2. 在必需的目录视图中适当地更新基数统计信息。

相关概念:

- 『调整查询处理』（在*《联合系统指南》*中）

相关任务:

- 第 25 页的『确定 BioRS 数据银行基数统计信息』
- 第 25 页的『更新 BioRS 昵称基数统计信息』
- 第 10 页的『更新 BioRS 列基数统计信息』
- 第 26 页的『更新 BioRS _ID_ 列基数』

确定 BioRS 数据银行基数统计信息

在可以更新昵称统计信息或更新与 BioRS _ID_ 元素相应的列的基数之前，必须确定 BioRS 数据银行基数统计信息。

过程:

要确定 BioRS 中的特定数据银行的基数统计信息，请使用 BioRS 实用程序 admin_find 或 www_find.cgi。指定 -c（基数）选项。有关这两个 BioRS 实用程序的更多信息，请参阅 BioRS 文件。

相关概念:

- 第 24 页的『BioRS 统计信息』

相关任务:

- 第 25 页的『更新 BioRS 昵称基数统计信息』
- 第 10 页的『更新 BioRS 列基数统计信息』
- 第 26 页的『更新 BioRS _ID_ 列基数』

更新 BioRS 昵称基数统计信息

当要为其创建昵称的 BioRS 数据银行的内容发生显著更改时，必须更新 BioRS 昵称基数统计信息。通过维护昵称的正确基数统计信息，优化器和 BioRS 包装器就能够选择性能最佳的数据存取方案。

要更新 BioRS 昵称基数统计信息，您用正确的基数数目修改 SYSSTAT.TABLES 目录视图。

先决条件:

必须确定与要更新其统计信息的昵称相应的 BioRS 数据银行的基数数目。

过程:

使用以下语法来发出 UPDATE 语句:

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

- <*cardinality*> 是与要更新其统计信息的昵称相应的 BioRS 数据银行的基数数目。
- <*nickname-schema*> 是与要更新其统计信息的昵称相关联的模式的名称。
- <*nickname-name*> 是要更新其统计信息的昵称的名称。

相关概念:

- 第 24 页的『BioRS 统计信息』

相关任务:

- 第 25 页的『确定 BioRS 数据银行基数统计信息』
- 第 10 页的『更新 BioRS 列基数统计信息』
- 第 26 页的『更新 BioRS _ID_ 列基数』

更新 BioRS _ID_ 列基数

通过维护映射至 BioRS _ID_ 元素的列的正确基数统计信息，优化器和 BioRS 包装器就能够选择性能最佳的数据存取方案。

要更新映射到 BioRS _ID_ 元素的列的基数数目，必须修改 SYSSTAT.COLUMNS 目录视图。

先决条件:

必须确定与引用了列的昵称相应的 BioRS 数据银行的基数数目。映射到 BioRS _ID_ 元素的列的基数数目必须与引用列的昵称的基数相匹配。

过程:

要更新 BioRS _ID_ 列基数统计信息，请使用以下语法来发出 UPDATE 语句:

```
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*> 是与列的昵称相应的 BioRS 数据银行的基数数目。
- <*nickname-schema*> 是与列的昵称相关联的模式的名称。
- <*nickname-name*> 是使用了列的昵称的名称。

相关概念:

- 第 24 页的『BioRS 统计信息』

相关任务:

- 第 25 页的『确定 BioRS 数据银行基数统计信息』
- 第 25 页的『更新 BioRS 昵称基数统计信息』

- 第 10 页的『更新 BioRS 列基数统计信息』

BioRS AllText 元素

BioRS 系统中的每个数据银行都包含一个称为 AllText 的元素。BioRS 自动为所有数据银行创建这个已建立索引的元素。

AllText 元素使您能够对条目中的所有文本执行搜索，而不仅仅是对特定的已建立索引的元素进行搜索。例如，对项 `muscus` 执行的搜索可以返回标题、摘要、描述或有机体中出现了 `muscus` 一词的条目。

要在 DB2 Information Integrator 查询中使用 AllText 元素，必须将 AllText 元素映射到昵称列。在 AllText 元素正确地映射到昵称列之后，就可以在 CONTAINS 定制函数调用中使用该昵称列。

如果在查询的 SELECT 列表中引用了映射 AllText 元素的列，则始终返回 NULL 值。

相关任务:

- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 第 17 页的『BioRS 包装器 - 示例查询』

与改变昵称相关的注意事项 - BioRS 包装器

可以使用 ALTER NICKNAME 语句来修改先前注册的 BioRS 昵称。借助 ALTER NICKNAME 语句，您可以:

- 更改列的名称
- 更改列的数据类型
- 添加、更改或删除列的选项

限制:

不能更改由昵称引用或在昵称中使用的 BioRS 数据银行的名称。如果昵称中使用的 BioRS 数据银行的名称发生更改，则必须删除并接着重新创建整个昵称。

如果指定了 REMOTE_OBJECT 选项，则不能更改或删除选项值。

如果更改列的数据类型，则新数据类型必须与相应的 BioRS 元素的数据类型兼容。

如果使用 ELEMENT_NAME 选项来更改列的元素名称，则不会对新名称进行检查以确保它正确。当查询引用了该列时，不正确的选项可能会导致错误。

如果更改 IS_INDEXED 列选项，则 BioRS 服务器不会对更改进行验证。当查询引用了该列时，不正确的选项可能会导致错误。

相关参考:

- 『ALTER NICKNAME statement』（在 *SQL Reference, Volume 2* 中）

定制函数表 – BioRS 包装器

表 9 提供了有关可以如何注册 4 个 BioRS 定制函数的示例。

为了帮助您注册定制函数，在 sqllib/samples/lifesci/biors 目录中提供了样本文件 create_function_mappings.ddl。create_function_mappings.ddl 文件包含每个定制函数的定义。可以运行此 DDL 文件来为每个安装了 BioRS 包装器的 DB2 数据库注册定制函数。

表 9. BioRS 包装器的定制函数

函数名	描述
CONTAINS (col VARCHAR(), term VARCHAR()), CONTAINS (col VARCHAR(), term CHAR()) CONTAINS (col VARCHAR(), term DATE) CONTAINS (col VARCHAR(), term TIMESTAMP)	使用给定的表达式来搜索已建立索引的列。 **col** 已建立索引的列。 **term** 搜索项。
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)	使用给定的表达式来搜索已建立索引的列。 **col** 已建立索引的列。 **term** 搜索项。
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)	使用给定的表达式来搜索已建立索引的列。 **col** 已建立索引的列。 **term** 搜索项。

表 9. BioRS 包装器的定制函数 (续)

函数名	描述
SEARCH_TERM (col VARCHAR(), term VARCHAR()) SEARCH_TERM (col VARCHAR(), term CHAR()) SEARCH_TERM (col CHAR(), term VARCHAR()) SEARCH_TERM (col CHAR(), term CHAR())	将 BioRS 搜索项传递至 BioRS 搜索引擎。 **col** 已建立索引的列。 **term** 搜索项。

相关任务:

- 第 3 页的『为 BioRS 包装器注册定制函数』

BioRS 包装器的消息

本主题说明在使用 BioRS 的包装器时可能会接收到的消息。有关这些消息的更多信息，请参阅《*DB2 消息参考*》。

表 10. 由 BioRS 的包装器发出的消息

错误代码	消息	说明
SQL0604N	列的长度、精度或小数位属性、单值类型、结构化类型、结构化类型的属性、函数或类型映射 <data-item> 无效。	昵称列的数据类型与下层数据银行元素的 BioRS 类型不兼容。检查 CREATE NICKNAME 语句中该列的数据类型。
SQL0901N	SQL 语句因不太严重的系统错误而失败。可以处理后续 SQL 语句。（原因: “创建包装器对象时出错。”）	创建新的包装器对象时出错。请与 IBM 软件支持机构联系。
SQL0901N	SQL 语句因不太严重的系统错误而失败。可以处理后续 SQL 语句。（原因: “BioRS <trace-point>/<code>。”）	这是一个内部错误。请与 IBM 软件支持机构联系。
SQL0901N	SQL 语句因不太严重的系统错误而失败。可以处理后续 SQL 语句。（原因: “内存分配失败: <trace-point>。”）	分配内存时出错。确保联合服务器主机有足够的内存可用并再次提交查询。如果问题仍存在，请与 IBM 软件支持机构联系。
SQL0901N	SQL 语句因不太严重的系统错误而失败。可以处理后续 SQL 语句。（原因: “sqlno_crule_save_plans[100]:rc(-214272209) 方案列表是空的。”）	优化器程序和 BioRS 包装器未能在运行查询的方案上达成一致。简化查询并再次运行它。

表 10. 由 BioRS 的包装器发出的消息 (续)

错误代码	消息	说明
SQL401N	运算 “=” 的操作数的数据类型不兼容。	由于定制函数谓词右边的表达式必须是整数值，所以该查询无效。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “”。相关联的文本和标记是 “Databank not found”。	在 BioRS 服务器上找不到 CREATE NICKNAME 语句中引用的 BioRS 数据银行。检查 CREATE NICKNAME 语句并确保所引用的数据银行的名称是正确的。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “”。相关联的文本和标记是 “Connection timed out”。	BioRS 服务器未能在 TIMEOUT 选项指定的时间段内响应通信请求。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “<trace_point>”。相关联的文本和标记是 “Error reading from server”。	从 BioRS 服务器读取数据时发生通信错误。<trace_point> 错误代码的值可能提供了有关该错误的更多信息。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “<trace_point>”。相关联的文本和标记是 “Host not found”。	找不到在 HOST 服务器选项中标识的 BioRS 服务器主机。检查 CREATE SERVER 语句并确保 HOST 服务器选项值是正确的。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “<trace_point>”。相关联的文本和标记是 “Unable to connect to server”。	包装器无法连接至由 HOST 服务器选项标识的服务器。<trace_point> 错误代码的值可能提供了有关该错误的更多信息。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “<trace_point>”。相关联的文本和标记是 “Unable to create TCPIP socket”。	包装器未能创建 TCPIP 套接字。<trace_point> 错误代码的值可能提供了有关该错误代码的更多信息。
SQL1822N	从数据源 “BioRS 包装器” 接收到意外的错误代码 “<trace_point>”。相关联的文本和标记是 “Error sending to server”。	包装器未能将请求发送至 BioRS 服务器。<trace_point> 错误代码的值可能提供了有关该错误的更多信息。

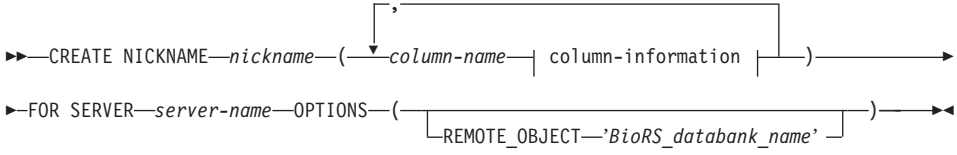
表 10. 由 BioRS 的包装器发出的消息 (续)

错误代码	消息	说明
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Cannot change case-sensitivty of server”。	不能使用 SQL 语句来更改 CASE_SENSITIVE 服务器选项的值。要更改此选项的值，必须删除服务器。然后，必须使用 CREATE SERVER 语句再次创建服务器并对 CASE_SENSITIVE 选项指定正确的值。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Multiple joins between two nicknames”。	由于在任何两个昵称之间只允许一个连接谓词，所以该查询无效。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Right side of function predicate must be constant”。	由于定制函数谓词右边的表达式必须是常量，所以该查询无效。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Arg 1 of custom function not a column”。	由于定制函数的第一个自变量必须引用 BioRS 昵称的列，所以该查询无效。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Arg 1 of CONTAINS function not indexed”。	该查询无效。BIORS.CONTAINS、BIORS.CONTAINS_LE 或 BIORS.CONTAINS_GE 函数的第一个自变量中引用的列必须是已建立索引的列。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Bad type for arg1 of <function-name> function”。	该查询无效。BIORS.CONTAINS、BIORS.CONTAINS_LE 或 BIORS.CONTAINS_GE 函数的第一个自变量中引用的列不具有正确的数据类型。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Arg 1 of SEARCH_TERM not _ID_ column”。	该查询无效。SEARCH_TERM 函数的第一个自变量中引用的列没有映射 BioRS _ID_ 元素。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Bind parameter cannot be NULL”。	在 BIORS.CONTAINS 函数的第二个自变量中引用的列或主变量值是 NULL。BioRS 包装器无法处理空值。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Cannot convert value to BioRS literal”。	已在文字、列或主变量中将值提交给包装器，未能将该值转换为有效的 BioRS 文字。

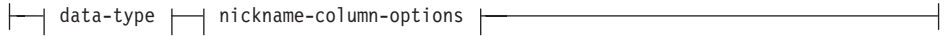
表 10. 由 BioRS 的包装器发出的消息 (续)

错误代码	消息	说明
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Cannot change server version”。	不能使用 ALTER SERVER 语句来更改服务器版本。要更改服务器版本，必须删除服务器。然后，必须使用 CREATE SERVER 语句再次创建具有正确版本的服务器。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Bad type for arg2 of <function-name> function”。	该查询无效。BIORS.CONTAINS、BIORS.CONTAINS_LE 或 BIORS.CONTAINS_GE 函数的第二个自变量中引用的列不具有正确的数据类型。
SQL30090N	操作对应用程序执行环境无效。原因代码 = “Nickname has no columns”。	没有在 CREATE NICKNAME 语句中指定列声明。需要列声明才能创建昵称。

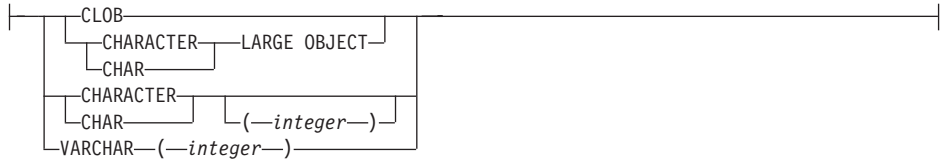
CREATE NICKNAME 语句语法 – BioRS 包装器



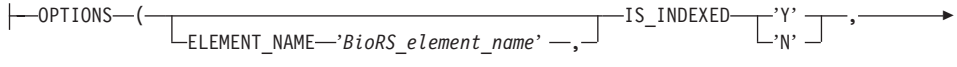
column-information:

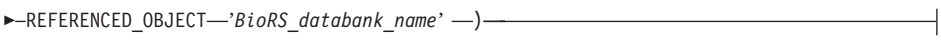


data-type:



nickname-column-options:





昵称列选项

昵称列选项值必须用单引号括起来。

ELEMENT_NAME

指定 BioRS 元素名称。此名称的区分大小写性取决于 BioRS 服务器的区分大小写性以及 CASE_SENSITIVE 服务器选项的值。仅当 BioRS 元素名称与列名不相同时才需要指定 BioRS 元素名称。

IS_INDEXED

指示是否对相应的列建立索引（是否可以在谓词中引用该列）。有效值为 'Y' 和 'N'。只能对这样的列指定 'Y' 值: BioRS 服务器为该列的相应元素建立了索引。

在创建昵称时，将把带有 'Y' 值的此选项自动添加至任何与 BioRS 的已建立索引的元素相应的列。

REFERENCED_OBJECT

此选项仅对其 BioRS 数据类型为 “引用” 的列有效。此选项指定当前列引用的 BioRS 数据银行的名称。此名称的区分大小写性取决于 BioRS 服务器的区分大小写性以及 CASE_SENSITIVE 服务器选项的值。

昵称选项

昵称选项值必须用单引号括起来。

REMOTE_OBJECT

指定与昵称相关联的 BioRS 数据银行的名称。此名称确定了模式以及昵称的 BioRS 数据银行。此名称还指定昵称与其它昵称的关系。此名称的区分大小写性取决于 BioRS 服务器的区分大小写性以及 CASE_SENSITIVE 服务器选项的值。

要点: 不能用 ALTER NICKNAME 语句更改或删除此名称。如果此选项中使用的 BioRS 数据银行的名称发生更改，则必须删除并接着重新创建整个昵称。

相关任务:

- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 『CREATE NICKNAME statement』（在 *SQL Reference, Volume 2* 中）
- 第 9 页的『CREATE NICKNAME 语句 – BioRS 包装器的示例』

CREATE SERVER 语句选项 – BioRS 包装器

BioRS 的 CREATE SERVER 语句的选项为:

TYPE 指定服务器类型。缺省值为 BioRS。缺省值是唯一受 BioRS 包装器支持的值。不需要指定此选项。

VERSION
指定服务器版本。缺省值为 1.0。缺省值是唯一受 BioRS 包装器支持的值。不需要指定此选项。

NODE 指定可以使用 BioRS 查询工具的系统的主机名。缺省值为 *localhost*。必须指定此选项。

PORT 指定要用来连接至 BioRS 服务器的端口号。缺省值为 5014。必须指定此选项。

TIMEOUT
指定 BioRS 包装器应该等待来自 BioRS 服务器的响应时间（以分钟计）。缺省值为 10。必须指定此选项。

CASE_SENSITIVE
指定 BioRS 服务器是否以区分大小写的方式处理名称。有效值为 'Y' 或 'N'。缺省值为 'Y'。

在 BioRS 产品中，一个配置参数控制着存储在 BioRS 服务器上的数据的区分大小写性。DB2 Information Integrator 的 CASE_SENSITIVE 选项与 BioRS 系统配置参数相应。必须在 BioRS 系统中以及在 DB2 Information Integrator 中将 BioRS 服务器区分大小写性配置设置同步。如果不在 BioRS 与 DB2 Information Integrator 之间保持区分大小写性配置设置同步，则当您尝试通过 DB2 Information Integrator 存取 BioRS 数据时将会发生错误。

要点: 在 DB2 Information Integrator 中创建新的 BioRS 服务器之后，就不能更改或删除 CASE_SENSITIVE 选项了。如果需要更改 CASE_SENSITIVE 选项，则必须删除并接着重新创建整个服务器。如果删除 BioRS 服务器，则还必须重新创建所有相应 BioRS 昵称。DB2 Information Integrator 自动删除所有与已删除的服务器相应的昵称。

相关任务:

- 第 6 页的『为 BioRS 数据源注册服务器』
- 第 7 页的『为 BioRS 数据源注册昵称』

相关参考:

- 『CREATE SERVER statement』（在 *SQL Reference, Volume 2* 中）
- 第 32 页的『CREATE NICKNAME 语句语法 – BioRS 包装器』

CREATE USER MAPPING 语句选项 – BioRS 包装器

GUEST

指定是否将在 BioRS 服务器上的 BioRS 来宾认证机制下执行操作。有效值为 'Y' 或 'N'。缺省值为 'Y'。

如果将此选项设置为 'Y'，则对此 DB2 Information Integrator 用户使用来宾认证来访问 BioRS 服务器。

如果将此选项设置为 'N'，则必须提供 BioRS 授权标识和密码才允许此 DB2 Information Integrator 用户访问 BioRS 服务器。

如果没有创建用户映射或者创建用户映射时没有指定选项，则对 DB2 Information Integrator 用户使用来宾认证来访问 BioRS 服务器。

REMOTE_AUTHID

指定允许此 DB2 用户存取 BioRS 数据源的用户标识。此远程标识必须具有 BioRS 应用程序期望的格式。如果将 GUEST 选项设置为 'N'，则此选项是必需的。

REMOTE_PASSWORD

指定此远程标识的密码。如果将 GUEST 选项设置为 'N'，则此选项是必需的。

示例:

以下 CREATE USER MAPPING 语句将用户 Charlie 映射至 Biors_Server1 服务器上的用户 Charlene。

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

相关任务:

- 第 6 页的『为 BioRS 数据源注册用户映射』

相关参考:

- 『CREATE USER MAPPING statement』（在 *SQL Reference, Volume 2* 中）

第 2 章 生命科学用户定义的函数

本章说明什么是生命科学用户定义的函数、如何将它们添加至联合系统以及如何在查询中使用它们。

生命科学用户定义的函数 – 概述

生命科学用户定义的函数为研究员提供他们通常用来分析数据的算法。这些函数可以在 Windows® NT、AIX 和 Linux 32 位平台上使用，但是 GeneWise 函数只可以在 AIX® 和 Linux 平台上使用。

生命科学用户定义的函数使用标准的单字母代码和 IUPAC-IUB 多义代码来表示氨基酸和核苷酸。

在可以使用生命科学用户定义的函数之前，必须注册它们。在注册它们之后，可以根据需要除去它们。

相关任务:

- 第 38 页的『注册生命科学用户定义的函数』
- 第 39 页的『除去生命科学用户定义的函数』

相关参考:

- 第 37 页的『按函数类别排列的生命科学用户定义的函数』

按函数类别排列的生命科学用户定义的函数

表 11 列示了按函数类别排列的生命科学用户定义的函数。它还提供了每个类别的简要描述。

表 11. 生命科学用户定义的函数

函数类别	用户定义的函数	描述
向后转换	LSPep2AmbNuc 和 LSPep2ProbNuc	将氨基酸序列转换为核苷酸序列。
定义行语法分析	LSDeflineParse	对定义行的元素（如 BLAST 包装器返回的元素或存在于 FASTA 格式数据文件中的元素）进行语法分析。

表 11. 生命科学用户定义的函数 (续)

函数类别	用户定义的函数	描述
一般化模式匹配	LSPatternMatch 和 LSPrositePattern	标识给定字符串中令人感兴趣的区域，如核苷酸或缩氨酸序列。
GeneWise	LSGeneWise	将蛋白质序列与基因的序列比对。
图谱	LSMultiMatch、LSMultiMatch3 和 LSBarCode	匹配核苷酸或氨基酸序列中的模式。
逆向	LSRevNuc、LSRevPep 和 LSRevComp	将核苷酸或氨基酸序列掉转。
转换	LSNuc2Pep 和 LSTransAllFrames	将核苷酸序列转换为缩氨酸序列。

相关概念:

- 第 37 页的『生命科学用户定义的函数 - 概述』

相关任务:

- 第 38 页的『注册生命科学用户定义的函数』
- 第 39 页的『除去生命科学用户定义的函数』

注册生命科学用户定义的函数

在可以使用生命科学用户定义的函数之前，必须注册它们。

过程:

要注册生命科学用户定义的函数，请使用 Windows NT 和 AIX 上的 `sqllib/bin` 目录中的 enable_LSFunctions 命令。

enable_LSFunctions 命令的语法为:

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

dbName
要对其注册函数的数据库的名称。

userID
数据库的有效用户标识。

password
用户标识的有效密码。

force 可以用来除去函数并重新注册它们的标志。当函数毁坏或意外地被删除时，可以使用此标志来重新注册它们。

此命令创建具有模式名 DB2LS 的函数。

以下示例显示 enable_LSFunctions 命令的样本输出:

```
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 ......
```

以下示例显示当使用 force 标志并且函数已注册时 enable_LSFunctions 命令的输出:

```
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 ......
```

除去生命科学用户定义的函数

如果不再需要系统上的生命科学用户定义的函数，则可以将它们除去。

过程:

要除去生命科学用户定义的函数，请使用 Windows NT 和 AIX 上的 sqllib/bin 目录中的 disable_LSFunctions 命令。

disable_LSFunctions 命令的语法为:

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

dbName
要从中除去函数的数据库的名称。

userID
数据库的有效用户标识。

password

用户标识的有效密码。

以下示例显示 disable_LSFunctions 命令的输出:

```
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 ......
```

向后转换用户定义的函数

使用向后转换用户定义的函数来将缩氨酸序列转换为核苷酸序列。向后转换就是将转换反转。

因为从氨基酸到核苷酸三基码的映射是一对多，所以向后转换将生成两个结果:

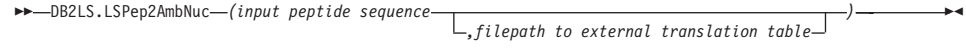
最不确定

单一文本转换和查找。使用 LSPep2AmbNuc 用户定义的函数来进行最不确定转换。

最有可能

需要来自基码频率表的附加信息。使用 LSPep2ProbNuc 用户定义的函数来进行最有可能转换。

LSPep2AmbNuc 用户定义的函数



input peptide sequence

描述缩氨酸序列的有效字符串表示法。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 10890 字节。输入数据使用标准的氨基酸符号和多义代码。

filepath to external translation table

如果使用定制的转换表，则包括文件路径信息以查找转换表。此路径的字符串值一定不能超过 255 个字符。

模式名是 DB2LS。

使用 LSPep2AmbNuc 函数来根据转换表从缩氨酸序列生成最不确定的核苷酸序列。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 32672 字节的字符串。此结果表示根据转换表（内置转换表或您指定的转换表）生成的最不确定的核苷酸序列。

如果不指定转换表，则此函数在缺省情况下使用表 12。

表 12. 缺省转换表

氨基酸符号	缩写	基码
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
V	Val	GTX
W	Trp	TGG
X	Xxx	XXX
Y	Tyr	TAY
Z	Glx	SAR
*	End	TRR

相关参考:

- 第 43 页的『LSPep2AmbNuc 用户定义的函数 – 错误消息』
- 第 44 页的『LSPep2ProbNuc 用户定义的函数』

• 第 42 页的『LSPep2AmbNuc 用户定义的函数 – 示例』

LSPep2AmbNuc 用户定义的函数 – 示例

可以用 values 语句调用此函数。唯一的输入是缩氨酸序列，如以下示例所示:

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

以上示例使用不确定转换和内置转换表来将缩氨酸转换为核苷酸。以上语句的结果是从标准氨基酸符号创建的核苷酸序列:

```
CAYMGX
```

以下示例使用不确定转换和内置表来将缩氨酸转换为核苷酸:

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

这个 values 语句的结果是以下核苷酸序列。(为了能够印刷在页面上，已对此序列作了折行处理。)

```
WSXMGXGGXTTYGGXTTYATHACXTAYWSXCAYWSXWSXATGATHGAYGARGCXCARA
ARWSXMGXCCXCAYAARATHGAYGGXMGXGTXGTXGARCCXAARMGXGCX
```

下一个示例显示将此函数应用于一组从表或昵称中抽取的值:

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

protein_table 表的 peptide_seq 列中的数据看起来是这样的:

表 13. peptide_seq 列中的数据

peptide_seq
GIKEDTEEHHLRDYFE
QKYHTVNGHNCEVRKA
.....

该 Select 语句的结果是:

```
GGXATHAARGARGAYACXGARGARCAYCAYYTXMGXGAYTAYTTYGAR
CARAARTAYCAYACXGTXAAYGGXCAYAAYTGYGARGTXMGXAARGCX
...
```

以下示例使用不确定转换和用户定义的表来将缩氨酸转换为核苷酸。通常，转换表之间的差别很小。可能只有一两个符号是独特的。这可能是由于一些物种具有较多的基码或一些物种具有较少的基码而导致的。例如，基码 AGG 在果蝇中是不存在的。

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

假设输入转换表是用于果蝇的，则 values 语句的结果如以下示例所示:

```
MGRGGXAAYATGGGXGGXGGXAAYTAYGGXAAYTARAAYGGXGGXGGXAAYTGGAAYAAYGGX
```

相关参考:

- 第 40 页的『LSPep2AmbNuc 用户定义的函数』
- 第 71 页的『LSNuc2Pep 用户定义的函数 - 示例』

LSPep2AmbNuc 用户定义的函数 - 错误消息

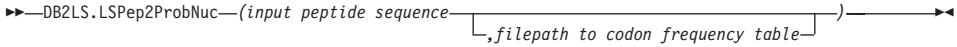
表 14. 由 LSPep2AmbNuc 用户定义的函数发出的消息

错误代码	消息	说明
SQL0443N	例程“DB2LS.LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUC”）已经返回带有诊断文本“序列无效”的错误 SQLSTATE。SQLSTATE=38608	所给序列无效。
SQL0443N	例程“DB2LS.LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUCUT”）已经返回带有诊断文本“找不到转换”的错误 SQLSTATE。SQLSTATE=38610	转换表文件是空的。
SQL0443N	例程“LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUCUT”）已经返回带有诊断文本“无法打开转换表文件”的错误 SQLSTATE。SQLSTATE=38612	指定的转换表文件不存在。
SQL0443N	例程“DB2LS.LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUCUT”）已经返回带有诊断文本“从文件读取的行太长”的错误 SQLSTATE。SQLSTATE=38614	文件包含超过所允许的长度的行。
SQL0443N	例程“DB2LS.LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUCUT”）已经返回带有诊断文本“数据文件无效”的错误 SQLSTATE。SQLSTATE=38615	文件格式无效。
SQL0443N	例程“LSPEP2AMBNUC”（特定名称“LSPEP2AMBNUCUT”）已经返回带有诊断文本“无法构造转换表”的错误 SQLSTATE。SQLSTATE=38611	在文件中找到无效的符号。

相关参考:

- 第 40 页的『LSPep2AmbNuc 用户定义的函数』

LSPep2ProbNuc 用户定义的函数



input peptide sequence

描述缩氨酸序列的有效字符串表示法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 10890 字节。输入数据使用标准的氨基酸符号。

filepath to codon frequency table

这是基码频率表。包含用于查找频率表的文件路径信息。此路径的字符串值一定不能超过 255 个字符。

模式名是 DB2LS。

使用 LSPep2ProbNuc 函数来根据第二个自变量中指定的基码频率表从缩氨酸序列生成最有可能的核苷酸序列。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 32672 字节的字符串，它表示最有可能的使用基码频率表的核苷酸序列。

相关参考:

- 第 40 页的『LSPep2AmbNuc 用户定义的函数』
- 第 45 页的『LSPep2ProbNuc 用户定义的函数 - 错误消息』
- 第 44 页的『LSPep2ProbNuc 用户定义的函数 - 示例』

LSPep2ProbNuc 用户定义的函数 - 示例

以下示例显示可以如何使用 yeast_high.cod 频率表中定义的最有可能的转换来将缩氨酸序列转换为核苷酸序列。

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

以上 values 语句的结果为:

```
AGAGACAATAACGACGATGATAAC
```

第二次执行同一个语句将生成以下字符串:

```
AGA GAT AATAACGACGAT GAC AAC
```

第三次执行同一个语句将生成以下具有随机值的字符串:

```
AGAGAT AAC AACGAC GACGATAAT
```

粗体的基码突出显示当前转换和上一转换的区别。

来自单个 values 语句的结果显示函数 LSPep2ProbNuc 根据先前统计信息选择了其中一个可能的符号。这与 LSPep2AmbNuc 函数不同，后者使用存在更多可能转换的不确定符号。

函数 LSPep2ProbNuc 为每个符号挑选最有可能的转换，然后用先前挑选的集合中的随机转换来替换每个符号。假定频率表包含以下数据:

表 15. 样本频率表数据

氨基酸	基码	频率
Ala	GCG	0.17
Ala	GCA	0.13
Ala	GCT	0.17
Ala	GCC	0.53

假定缩氨酸序列包含 4 个“A”符号（Ala）。函数将 A 转换为 GCC 两次；转换为 GCG 一次，转换为 GCT 一次。但是，函数生成转换的顺序是随机的。查询可以将第一个 A 转换为 {GCC, GCC, GCG, GCT} 集合中的每个转换。结果始终是这样的: 在输出 DNA 序列中，GCC 出现两次，GCG 出现一次，GCT 出现一次。对同一个序列多次执行此函数可能会返回交换了值的 DNA 序列。

相关参考:

- 第 44 页的『LSPep2ProbNuc 用户定义的函数』
- 第 45 页的『LSPep2ProbNuc 用户定义的函数 - 错误消息』
- 第 42 页的『LSPep2AmbNuc 用户定义的函数 - 示例』

LSPep2ProbNuc 用户定义的函数 - 错误消息

表 16. 由 LSPep2ProbNuc 用户定义的函数发出的消息

错误代码	消息	说明
SQL0443N	例程“DB2LS.LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“序列无效”的错误 SQLSTATE。SQLSTATE=38608	输入序列无效。
SQL0443N	例程“DB2LS.LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“找不到转换”的错误 SQLSTATE。SQLSTATE=38610	基码频率表文件是空的。

表 16. 由 LSPep2ProbNuc 用户定义的函数发出的消息 (续)

错误代码	消息	说明
SQL0443N	例程“LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“无法打开转换表文件”的错误 SQLSTATE。SQLSTATE=38612	该文件不存在。
SQL0443N	例程“DB2LS.LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“从文件读取的行太长”的错误 SQLSTATE。SQLSTATE=38614	文件包含的行超出所允许的长度。
SQL0443N	例程“DB2LS.LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“数据文件无效”的错误 SQLSTATE。SQLSTATE=38615	文件格式无效。
SQL0443N	例程“LSPEP2PROBNUC”（特定名称“LSPEP2PROBNUC”）已经返回带有诊断文本“无法构造转换表”的错误 SQLSTATE。SQLSTATE=38611	文件包含无效的符号。

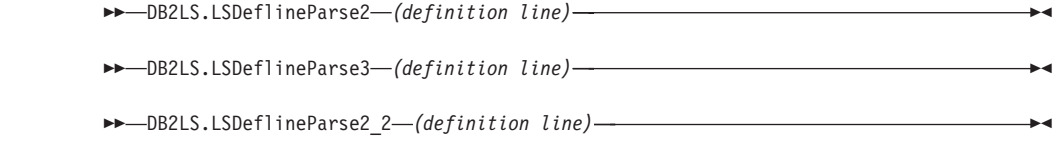
相关参考:

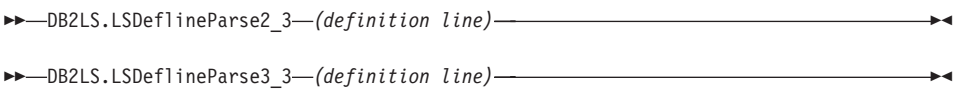
- 第 44 页的『LSPep2ProbNuc 用户定义的函数』
- 第 44 页的『LSPep2ProbNuc 用户定义的函数 - 示例』

定义行语法分析用户定义的函数

定义行语法分析用户定义的函数对定义行的元素进行语法分析，例如，以便能够与从定义行分析出来的序列标识中的其它数据源连接，或者对定义行各部分的谓词（如 `'species = "human"'`）求值。定义行函数涵盖了大多数常见的定义行格式。示例包括 BLAST 包装器返回的定义行元素或存在于 FASTA 格式数据文件中的定义行元素。

LSDeflineParse 用户定义的函数





definition line

具有 FASTA 格式的定义行的有效字符串表示法。此字符串必须具有 VARCHAR 数据类型，并且实际长度不大于 1024 字节。

模式名是 DB2LS。

每个 LSDeflineParse 函数都对 NCBI 标准 FASTA 序列标识（NSID）的字段进行语法分析并将描述存放到表列中。作为复合定义的定义行输出到多个行中，每一行包含一个组件定义。

LSDeflineParse2 对带有双字段 NSID 的定义行进行语法分析。此函数的结果是包含 4 个列的表:

表 17. LSDeflineParse2 用户定义的函数结果表的列描述

列名	描述
ROWID	一个整数，它对函数返回的行进行编号。
TAG	最多包含 3 个字符的 VARCHAR，它表示 NSID 标记。
IDENTIFIER	最多包含 20 个字符的 VARCHAR，它表示 NSID 中的第二个标识字段。
DESCRIPTION	最多包含 1019 个字符的 VARCHAR。

LSDeflineParse3 对带有三字段 NSID 的定义行进行语法分析。此函数的结果是包含 5 个列的表:

表 18. LSDeflineParse3 用户定义的函数结果表的列描述

列名	描述
ROWID	一个整数，它对函数返回的行进行编号。
TAG	最多包含 3 个字符的 VARCHAR，它表示 NSID 标记。
ACCESSION	最多包含 20 个字符的 VARCHAR，它表示 NSID 中的第二个标识字段。
LOCUS	最多包含 20 个字符的 VARCHAR，它表示 NSID 中的第三个标识字段。
DESCRIPTION	最多包含 1017 个字符的 VARCHAR。

LSDeflineParse2_2 对带有由一对并置的双字段 NSID 组成的混合标识的定义行进行语法分析。此函数的结果是包含 6 个列的表:

表 19. LSDeflineParse2_2 用户定义的函数结果表的列描述

列名	描述
ROWID	一个整数，它对函数返回的行进行编号。
TAG1	最多包含 3 个字符的 VARCHAR，它表示第一个标识的 NSID 标记。
IDENTIFIER1	最多包含 20 个字符的 VARCHAR，它表示第一个 NSID 的第二个标识字段。
TAG2	最多包含 3 个字符的 VARCHAR，它表示第一个标识的 NSID 标记。
IDENTIFIER2	最多包含 20 个字符的 VARCHAR，它表示第二个 NSID 的第二个标识字段。
DESCRIPTION	最多包含 1015 个字符的 VARCHAR。

LSDeflineParse2_3 对带有由双字段 NSID 与三字段 NSID 的并置组成的复合标识的定义行进行语法分析。输入定义行中的并置次序（无论是双字段 NSID 位于三字段 NSID 之前还是相反）并不重要。此函数的结果是包含 7 个列的表:

表 20. LSDeflineParse2_3 用户定义的函数结果表的列描述

列名	描述
ROWID	一个整数，它对函数返回的行进行编号。
TAG1	最多包含 3 个字符的 VARCHAR，它表示双字段标识的 NSID 标记。
IDENTIFIER	最多包含 20 个字符的 VARCHAR，它表示双字段 NSID 的第二个标识字段。
TAG2	最多包含 3 个字符的 VARCHAR，它表示三字段标识的 NSID 标记。
ACCESSION	最多包含 20 个字符的 VARCHAR，它表示三字段 NSID 的第二个标识字段。
LOCUS	最多包含 20 个字符的 VARCHAR，它表示三字段 NSID 的第三个标识字段。
DESCRIPTION	最多包含 1013 个字符的 VARCHAR。

LSDeflineParse3_3 对带有由一对三字段 NSID 组成的复合标识的定义行进行语法分析。此函数的结果是包含 8 个列的表:

表 21. LSDeflineParse3_3 用户定义的函数结果表的列描述

列名	描述
ROWID	一个整数，它对函数返回的行进行编号。
TAG1	最多包含 3 个字符的 VARCHAR，它表示第一个标识的 NSID 标记。
ACCESSION1	最多包含 20 个字符的 VARCHAR，它表示第一个 NSID 的第二个标识字段。
LOCUS1	最多包含 20 个字符的 VARCHAR，它表示第一个 NSID 的第三个标识字段。
TAG2	最多包含 3 个字符的 VARCHAR，它表示第一个标识的 NSID 标记。
ACCESSION2	最多包含 20 个字符的 VARCHAR，它表示第二个 NSID 的第二个标识字段。
LOCUS2	最多包含 20 个字符的 VARCHAR，它表示第二个 NSID 的第三个标识字段。
DESCRIPTION	最多包含 1014 个字符的 VARCHAR。

相关参考:

- 第 49 页的『LSDeflineParse 用户定义的函数 – 示例』

LSDeflineParse 用户定义的函数 – 示例

本主题包含 7 个示例，这些示例显示 LSDeflineParse 用户定义的函数如何对定义行进行语法分析以生成结果表。

以下示例查询和结果表显示 LSDeflineParse2 用户定义的函数如何对包含双字段 NSID 的定义行进行语法分析:

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

结果表包含下列数据:

表 22. LSDeflineParse2 用户定义的函数结果数据

列名	数据
ROWID	1
TAG	gi
IDENTIFIER	12346
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

以下示例查询和结果表显示 LSDeflineParse3 用户定义的函数如何对包含三字段 NSID 的定义行进行语法分析:

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

结果表包含下列数据:

表 23. LSDeflineParse3 用户定义的函数结果数据

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

以下示例查询和结果表显示 LSDeflineParse2_2 用户定义的函数如何对包含由一对双字段 NSID 组成的复合标识的定义行进行语法分析:

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

结果表包含下列数据:

表 24. LSDeflineParse2_2 用户定义的函数结果数据

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

以下示例查询包含带有复合标识的定义行，该复合标识由双字段 NSID 和三字段 NSID 的并置组成。此示例显示 LSDeflineParse2_3 函数如何对该定义行进行语法分析。

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

结果表包含下列数据:

表 25. LSDeflineParse2_3 用户定义的函数结果数据

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

以下示例查询包含带有复合标识的定义行，该复合标识由三字段 NSID 和双字段 NSID 的并置组成。此示例显示 LSDeflineParse2_3 函数如何对该定义行进行语法分析。

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

结果表包含下列数据:

表 26. LSDeflineParse2_3 用户定义的函数结果数据

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

以下示例查询和结果表显示 LSDeflineParse3_3 用户定义的函数如何对包含带有一对三字段 NSID 的复合标识的定义行进行语法分析:

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

结果表包含下列数据:

表 27. LSDeflineParse3_3 用户定义的函数结果数据

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

可以使用任何定义行用户定义的函数来对复合定义行进行语法分析。以下示例查询包含带有由 Control-A 字符分隔的多个定义的复合定义行。可以在 NCBI 的非冗余蛋白质数据库 nr 中找到此类型的定义行。此示例显示 LSDeflineParse2_3 函数如何对该定义行进行语法分析。

```
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
```

结果表包含下列数据:

表 28. LSDeflineParse2_3 用户定义的函数结果数据

列名	数据	数据
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

相关参考:

- 第 46 页的『LSDeflineParse 用户定义的函数』

一般化模式匹配用户定义的函数

一般化模式匹配用户定义的函数标识给定字符串中的令人感兴趣的区域，如核苷酸或缩氨酸序列。

LSPatternMatch 用户定义的函数

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

input character sequence

此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

pattern

在任何有效的 Perl 正则表达式中指定的模式。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

可以使用 LSPatternMatch 用户定义的函数来搜索指定的模式的输入核苷酸或缩氨酸序列。

此函数的结果是一个整数，它表示模式的第一个匹配在序列中的位置。如果没有匹配，则此函数返回零值。

如果模式是使用 PROSITE 语法书写的，则可以通过 LSPrositePattern 用户定义的函数将它们转换为 Perl 语法。然后，可以将经过转换的语法与 LSPatternMatch 用户定义的函数配合使用。

相关参考:

- 第 53 页的『LSPatternMatch 用户定义的函数 – 示例』
- 第 55 页的『LSPrositePattern 用户定义的函数』

LSPatternMatch 用户定义的函数 – 示例

在以下示例中，查找与 “coward”、“cowage”、“cowboy” 或 “cowl” 相匹配的字符串的开始位置。

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

此函数按字符进行搜索，在此示例中，将返回值 9。字符串 “cowboy” 从位置 9 开始（假定第一个位置是 1）。

在下一个示例中，查找与 “not” 或 “non” 相匹配的字符串的开始位置:

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

此函数按字符进行搜索，在此示例中，将返回值 7。字符串 “not” 从位置 7 开始（假定第一个位置是 1）。

LSPatternMatch 在 Select 语句中非常有用，它可以用来通过使用 PERL 语法来对结果进行过滤，与 SQL LIKE 语句相比，PERL 语法是功能更为强大的语法。在以下示例中，对 Blast 输出使用 LSPatternMatch 来过滤与特定模式相匹配的基因:

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

如果您更熟悉 PROSITE 语法，则可以将 LSPrositePattern 函数与上述查询配合使用。将查询更改为:

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

模式匹配函数对于搜索其它类型的文本以及核苷酸或缩氨酸序列而言非常有用。当性能是关心的主要问题时，请考虑使用 SQL LIKE 语句。

以下示例显示了一个查询，该查询根据在比对的主题或目标行中找到的蛋白质图谱来过滤 BLAST hsp 比对。此示例摘自 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）。

以下查询只返回主题序列包含 P 循环 ATPase 域 [GA]xxxxGK[ST] 的比对。此查询对 NCBI 的非冗余蛋白质序列数据库使用 CED4（即细胞死亡 *Caenorhabditis elegans* 调节器）作为查询序列。数据库通过转换 GenBank 条目 X69016 的 CDS 特征来检索 Blast 查询序列。

```
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;
```

可以使用下一个示例查询来在基因的序列中查找包含与规范查询序列相关的公认单一核苷酸多态性（SNP）的 HSP。此查询摘自 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。

此查询使用对 Blast HSP 中间行的模式匹配来查找以下模式: 大于或等于 20 个完全匹配，后跟单一不匹配，后跟大于或等于 20 个完全匹配。即，比对的中间行包含 20 个 “|” 字符，接着是一个空格，接着又是 20 个 “|” 字符。

此示例还显示了对不是核苷酸或缩氨酸序列的字符串使用 LSPatternMatch 用户定义的函数。

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

可以将上一个查询重新编写为:

```
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
```

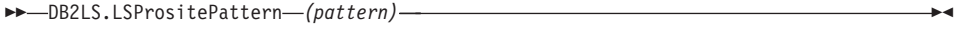
第二个查询将返回包含匹配以及匹配的字符串及其在序列中的位置的 Blast 行。

BlastOutput 是 BlastN 昵称的视图。

相关参考:

- 第 56 页的『LSPrositePattern 用户定义的函数 - 示例』
- 第 53 页的『LSPatternMatch 用户定义的函数』
- 第 55 页的『LSPrositePattern 用户定义的函数』

LSPrositePattern 用户定义的函数



pattern

Prosite 语法指定的模式匹配语法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

使用 LSPrositePattern 用户定义的函数来从 PROSITE 语法转换为 PERL 语法。然后，可以将经过转换的语法与 LSPatternMatch、LSMultiMatch 和 LSMultiMatch3 用户定义的函数配合使用。

此函数的结果是一个字符串，它表示具有 Perl 语法的正则表达式。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

相关参考:

- 第 56 页的『LSPrositePattern 用户定义的函数 - 示例』
- 第 53 页的『LSPatternMatch 用户定义的函数』

LSPrositePattern 用户定义的函数 - 示例

在以下示例中，将模式从 PROSITE 语法转换为 PERL 语法。

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

此函数将具有 PROSITE 语法的输入模式转换为具有 Perl 语法的等价模式，如以下示例所示:

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

下一个示例将另一个语法模式从 PROSITE 转换为 PERL 语法:

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

此函数根据输入模式从 PROSITE 语法转换字符串并返回以下结果:

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

下一个示例将与标识号为 PS01205 的 PROSITE 数据库条目相应的模式转换为作为模式匹配函数的输入使用的 PERL 模式。

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

此查询的结果是:

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

下一个示例显示可以如何在查询中使用此函数。此查询只打印出与指定的 PROSITE 模式相匹配的序列。

```
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;
```

下一个示例转换与标识为 PS00261 的 PROSITE 条目相应的模式:

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

此查询的结果是:

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

相关参考:

- 第 53 页的『LSPatternMatch 用户定义的函数 – 示例』
- 第 55 页的『LSPrositePattern 用户定义的函数』

正则表达式支持

正则表达式支持是由 PCRE 库软件包提供的，这是由 Philip Hazel 编写的开放式源代码软件，版权归英国剑桥大学所有。

可以在 ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/ 找到源代码。

GeneWise 用户定义的函数

GeneWise 用户定义的函数将蛋白质序列与基因的序列比对。

GeneWise 是常常用来将蛋白质序列与基因的 DNA 序列比对的组件，从而允许基因内区和移码错误。

链接到 GeneWise

本主题描述链接到 GeneWise 库的过程。

过程:

1. 从 http://www.ebi.ac.uk/Wise2 下载 Wise2 软件包 V2.1.20c。
2. 将归档展开到您选择的文件夹中。
3. 借助 pthread 支持编译软件包。有关此步骤的更多信息，请参阅 Wise2 文档。
4. 在 **make api** 的根目录中运行它。
5. 将 WISE2_HOME 环境变量设置为指向 Wise2 软件包根目录。
6. 在 sqllib/cfg/db2dj.ini 文件中设置 WISECONFIGDIR 变量以指向 wisecfg 子目录。

   例如，如果 Wise2 软件包安装在 /usr/wise2.1.20c/ 中，则将 WISECONFIGDIR=/usr/wise2.1.20c/wisecfg/ 添加到 db2dj.ini 文件中。
7. 从 sqllib/bin 目录中运行 **djxlinkLSGeneWise** 并检查其输出。
8. 检查 sqllib/lib 目录中的 djxlinkLSGeneWise.out。
9. 如果没有报告错误的话，则表示库已构建成功。

相关参考:

- 第 58 页的『LSGeneWise 用户定义的函数』

LSGeneWise 用户定义的函数

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

protein sequence

描述缩氨酸序列的有效字符串表示法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

DNA_sequence

描述核苷酸序列的有效字符串表示法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

表 29 显示了 LSGeneWise 函数返回的一行输出表。

表 29. LSGeneWise 函数返回的输出表的列名、类型和描述

列名	类型	描述
PROTEIN_OFFSET	INTEGER	表示在输入蛋白质序列中找到比对的位置的开始偏移。
DNA_OFFSET	INTEGER	表示在输入 DNA 序列中找到比对的位置的开始偏移。
PROTEIN	VARCHAR(32672)	输入序列中的片段，它表示比对的序列。
SIMILARITY	VARCHAR(32672)	显示蛋白质与 DNA 序列之间的匹配。完全匹配由相应的符号字母作标记。不完全但具有肯定评分的匹配由“+”号指示，不匹配由空格指示。
TRANSLATED_DNA	VARCHAR(32672)	经过转换的 DNA 序列。此序列可能包含短横线和特殊符号，如删除和基因内区。
DNA	VARCHAR(32672)	带有特殊标记（如移码和基因内区）的 DNA 序列。

GeneWise 程序的输出与 LSGeneWise UDF 的输出之间的相应关系如下:

- GeneWise 程序打印的蛋白质和 DNA 偏移与 PROTEIN_OFFSET 和 DNA_OFFSET 列相匹配。
- GeneWise 在第一行上打印的蛋白质序列与 PROTEIN 列相匹配。
- 相似行（即 GeneWise 输出的第二行）与 SIMILARITY 列相匹配。
- GeneWise 输出的第三行与 TRANSLATED_DNA 列相匹配。
- 通过垂直地阅读 GeneWise 输出的第四、第五和第六行，这些行组合到 DNA 列中。

使用 LSGeneWise 用户定义的函数来将蛋白质序列与基因的 DNA 序列比对，从而允许基因内区和移码错误。

有关 LSGeneWise 用户定义的函数输出的更多信息，请参阅 http://www.ebi.ac.uk/Wise2。

相关任务:

- 第 57 页的『链接到 GeneWise』

相关参考:

- 第 59 页的『LSGeneWise 用户定义的函数 - 示例』

LSGeneWise 用户定义的函数 - 示例

以下示例显示了使用 LSGeneWise 用户定义的函数的查询以及结果数据。

```
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. 结果表

列	数据
PROTEIN_OFFSET	23
DNA_OFFSET	14
PROTEIN	KLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKETGKKRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVDVKKALPKQNDQQGGGGGRGGPGGRAGGNRGNMGGGNYGNQNGGGNWNNGGN
SIMILARITY	K+FVG +K+D +E +RDYF+ +G I I I+ D+ +GKKRGFA+VFDD+D VDK+V+QK H +NG +V+KAL KQ RG G GN+GGG G G N+ GGN
TRANSLATED_DNA	KIFVGGIKEDTEEHHLRDYFEQYGKIEVIEIMTDRGSGKKRGFAxVTFDDHDSVDKIVIQKYHTVNGHNCEVRKALSKQEMASASSSQRGRSGS------GNFGGGRGGGFGGNDNFGRGGN
DNA	aagatatttgttggtggcattaaagaagacactgaagaacatcacctaagagat...

相关任务:

- 第 57 页的『链接到 GeneWise』

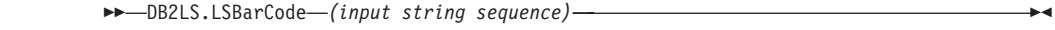
相关参考:

- 第 58 页的『LSGeneWise 用户定义的函数』

图谱用户定义的函数

图谱用户定义的函数匹配核苷酸或氨基酸序列中的模式。

LSBarCode 用户定义的函数



input string sequence

表示两个序列段之间的 HSP 比对的有效字符串。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

使用 LSBarCode 用户定义的函数来将某个序列用作输入并通过将空格和加号之外的每个字符替换为垂直竖线（|）生成另一个序列。

此函数的结果是表示条形码序列的变量字符序列。

相关参考:

- 第 60 页的『LSBarCode 用户定义的函数 - 示例』
- 第 62 页的『LSMultiMatch 用户定义的函数』
- 第 63 页的『LSMultiMatch3 用户定义的函数』

LSBarCode 用户定义的函数 - 示例

此示例从字符串序列创建条形码:

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

以上 values 语句的结果是:

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

下一个示例显示此函数的更实际的用法。假定运行 BLAST 搜索的研究员只希望返回某些 HSP 比对: 这些 HSP 比对所包含的脯氨酸少于它们的完全匹配的 25%。

此示例使用此函数来计算脯氨酸（符号 'P'）在 BLAST 返回的比对所包含的完全匹配中所占的百分比。注意，此示例还调用了 LSMultiMatch3 用户定义的函数。此查询使用匹配函数来查找完全匹配。由于 Blast 并不总是在比对中返回竖线（ "|" ）序列，所以在此查询中将此函数与 LSBarCode 函数配合使用。以下示例显示了这一点:

```
查询:         MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
比对:         MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
目标:         MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

为了确保输出与正确的竖线序列比对，请使用 LSBarCode 函数。该函数将使用竖线来替换除空格和加号以外的所有字符。

```
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;
```

在此查询中，BlastOutput 实际上是 Blast 昵称的视图。此查询使用 LSMultiMatch3 函数来返回比对的完全匹配。第一处使用返回符号 "P" 的完全匹配，第二处使用返回所有完全匹配。表 31 显示了结果表中的行。

表 31. 样本结果行

HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_PROLINES
NIWDFMQGN...	NIWDFMQGN...	Identities = 80/80 (100%), Positives = 80/80 (100%), Gaps = 0/80 (0%)	+2.50000000000000E-002

上一个查询摘自 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。

相关参考:

- 第 64 页的『LSMultiMatch3 用户定义的函数 - 示例』
- 第 60 页的『LSBarCode 用户定义的函数』

LSMultiMatch 用户定义的函数

```
►►—DB2LS.LSMultiMatch—(input nucleotide or peptide sequence, pattern)—————►◄
```

input nucleotide or peptide sequence

描述核苷酸或缩氨酸序列的有效字符串表示法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

pattern

Perl 语言指定的模式匹配语法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

使用 LSMultiMatch 用户定义的函数来为输入序列中的每个不重叠的匹配返回一个表。每个表都由开始位置和匹配序列片段组成。

此函数的结果是带有 2 个列的表。第一个列是一个整数，它表示模式的匹配在序列中的开始位置。第二个列是匹配序列片段。

相关参考:

- 第 62 页的『LSMultiMatch 用户定义的函数 - 示例』
- 第 60 页的『LSBarCode 用户定义的函数』
- 第 63 页的『LSMultiMatch3 用户定义的函数』

LSMultiMatch 用户定义的函数 - 示例

此示例查找所有从输入获取的非重叠匹配的位置和匹配片段。

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

此查询返回基于这个 Select 语句的表，该表显示了匹配结果:

表 32. 返回多行的 LSMultiMatch 的结果

POSITION	MATCH
7	not
15	non

LSMultiMatch 返回所有匹配的位置和匹配字符串。以下示例在 Entrez 核苷酸中搜索包含特定图谱的序列条目。此查询打印序列标识和匹配的序列。位于开头和结尾处的子模式 ".{0,9}" 必须与序列之前和之后的最多 9 个字符相匹配。此查询还打印这些字符。

```
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;
```

此查询的结果如下:

表 33. 搜索 Entrez 数据

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

相关参考:

- 第 62 页的『LSMultiMatch 用户定义的函数』
- 第 60 页的『LSBarCode 用户定义的函数』
- 第 63 页的『LSMultiMatch3 用户定义的函数』

LSMultiMatch3 用户定义的函数

input strings

描述核苷酸或缩氨酸序列的有效字符串表示法，或者是来自 Blast 比对的 HSP_Midline 字符串。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

pattern

Perl 语言指定的模式匹配语法。此字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

使用 LSMultiMatch3 用户定义的函数来输入 3 个模式和 3 个字符串并返回全部 3 个字符串与它们各自的模式相匹配的任何位置。可以使用这个用户定义的函数来对比对执行模式匹配。

此函数的结果是带有 4 个列的表。第一个列是一个整数，它表示模式的匹配在所有序列中的开始位置。此函数将所有字符串一起固定在第一个位置处。第二列、第三列和第四列是匹配序列片段。

相关参考:

- 第 64 页的『LSMultiMatch3 用户定义的函数 - 示例』
- 第 62 页的『LSMultiMatch 用户定义的函数』
- 第 60 页的『LSBarCode 用户定义的函数』

LSMultiMatch3 用户定义的函数 - 示例

以下示例使用此函数来计算特定氨基酸符号在 Blast 返回的完全匹配中所占的百分比。注意，此示例还调用了 LSBarCode 用户定义的函数。查询需要这样做的原因是 Blast 并不总是在比对中返回竖线（“|”）序列。以下示例说明了这一点:

```
查询:         MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
比对:         MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
目标:         MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

为了确保输出与正确的竖线序列比对，请使用 LSBarCode 函数来转换序列。此函数使用竖线来替换所有的非空格和非“+”字符。

```
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;
```

在此查询中，BlastOutput 是 Blast Select 的视图。此查询使用 LSMultiMatch3 函数来返回比对的完全匹配。第一处使用返回符号“P”的完全匹配，第二处使用返回所有完全匹配。第 65 页的表 34 显示了结果表中的行。

表 34. 样本结果行

HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_PROLINES
NIWDFMQG...	NIWDFMQG...	Identities = 80/80 (100%), Positives = 80/80 (100%), Gaps = 0/80 (0%)	+2.50000000000000E-002

上一个查询摘自 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。

以下示例在三个独立的字符串片段中查找三个独立的模式:

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

它返回所有匹配的位置和匹配字符串，如下表所示:

表 35. 使用三个输入的多重匹配的结果

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

下一个示例在三个独立的字符串片段中查找三个独立的模式:

```
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
```

下表显示了结果:

表 36. 使用三个输入的多重匹配的结果

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

相关参考:

- 第 60 页的『LSBarCode 用户定义的函数 - 示例』
- 第 60 页的『LSBarCode 用户定义的函数』
- 第 63 页的『LSMultiMatch3 用户定义的函数』

逆向用户定义的函数

逆向用户定义的函数将核苷酸或氨基酸序列反转。

LSRevComp 用户定义的函数

```
►►—DB2LS.LSRevComp—(input nucleotide sequence)—————————————————►◄
```

input nucleotide sequence

描述核苷酸序列的有效字符串表示法。此序列可以包含 IUPAC 多义代码。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

模式名是 DB2LS。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 32672 字节的字符串，它表示核苷酸序列的逆向补充。

相关参考:

- 第 66 页的『LSRevComp 用户定义的函数 - 示例』
- 第 67 页的『LSRevNuc 用户定义的函数』
- 第 68 页的『LSRevPep 用户定义的函数』

LSRevComp 用户定义的函数 - 示例

每当要使用任何接受核苷酸序列的内置函数时，都可以在 SQL 语句中使用 LSRevComp 函数。例如:

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

此示例使用该函数来返回来自主变量的输入序列的逆向补充。

如果使用无效的字符串或无效的数据类型，则会收到以下错误消息:

```
SQL0443N 例程“DB2LS.LSREVCOMP”（特定名称“LSREVCOMP”）已经返回带有诊断文本
“序列无效”的错误 SQLSTATE。SQLSTATE=38608
```

如果输入字母不正确，则将发生异常。

以下示例显示 LSRevComp 用户定义的函数是如何在查询中工作的:

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

此查询的结果显示如下:

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

AGTATTGGTCAATCTTTTGAT          ATCAAAAGATTGACCAATACT

TGGTCAATCTTTTGATA              TATCAAAAGATTGACCA

TTGGCCAATCTTTTGATATCC          GGATATCAAAAGATTGGCCAA

TCAATCTTTTGATATCC              GGATATCAAAAGATTGA

GGATATCAAAAGATTGA              TCAATCTTTTGATATCC


  5 record(s) selected.
```

可以将这个逆向函数与其它生命科学用户定义的函数一起使用以转换核苷酸序列的逆向补充，如以下示例所示:

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

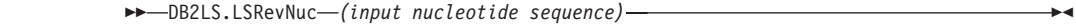
此查询返回以下结果:

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

相关参考:

- 第 66 页的『LSRevComp 用户定义的函数』

LSRevNuc 用户定义的函数



input nucleotide sequence

描述核苷酸序列的有效字符串表示法。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。核苷酸序列必须是 DNA 字母表的部分或全部。

模式名是 DB2LS。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 32672 字节的字符串，它表示核苷酸序列的逆序。

相关参考:

- 第 68 页的『LSRevNuc 用户定义的函数 – 示例』
- 第 66 页的『LSRevComp 用户定义的函数』
- 第 68 页的『LSRevPep 用户定义的函数』

LSRevNuc 用户定义的函数 – 示例

每当要使用任何接受核苷酸序列的内置函数时，都可以在 SQL 语句中使用 LSRevNuc 函数。例如:

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

此示例使用该函数来颠倒来自主变量的输入数据。

如果使用无效的字符串或无效的数据类型，则会收到以下错误消息:

```
SQL0443N 例程“DB2LS.LSREVNUC”（特定名称“LSREVNUC”）已经返回带有诊断文本
“序列无效”的错误 SQLSTATE。SQLSTATE=38608
```

以下示例显示 LSRevNuc 用户定义的函数在查询中的使用。

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

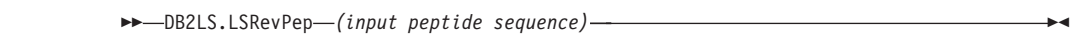
以下结果表显示了该查询返回的逆向核苷酸序列:

```
HSP_H_SEQ                                          REV_HSP_H_SEQ
-------------------------------------------------- --------------------------
CGCGGGCAAACTGAAGATAAAGC                            CGAAATAGAAGTCAAACGGGCGC

GCGCTAGCCCCCTACGTATTAC                             CATTATGCATCCCCCGATCGCG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG


  5 record(s) selected.
```

相关参考:

- 第 67 页的『LSRevNuc 用户定义的函数』

LSRevPep 用户定义的函数



input peptide sequence

描述缩氨酸序列的有效字符串表示法。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。输入序列必须是蛋白质字母表的一部分。

模式名是 DB2LS。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 32672 字节的字符串，它表示缩氨酸序列的逆序。

相关参考:

- 第 69 页的『LSRevPep 用户定义的函数 - 示例』
- 第 66 页的『LSRevComp 用户定义的函数』
- 第 67 页的『LSRevNuc 用户定义的函数』

LSRevPep 用户定义的函数 - 示例

每当要使用任何接受缩氨酸序列的内置函数时，都可以在 SQL 语句中使用 LSRevPep 函数。例如:

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

此示例使用该函数来颠倒来自主变量的输入数据。

如果使用无效的字符串或无效的数据类型，则会收到以下错误消息:

```
SQL0443N 例程“DB2LS.LSREVPEP”（特定名称“LSREVPEP”）已经返回
带有诊断文本“序列无效”的错误 SQLSTATE。SQLSTATE=38608
```

以下示例显示如何在查询中使用 LSRevPep 用户定义的函数:

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

下表显示查询返回的逆向缩氨酸序列。

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

MLCEIECRALSTAHTRLIHDFEPRDALTYL...  HDETFINKGELYTLADRPEFDHILRTHATS...

RVVSTEHTRLVTDAYPEFSISFTATKN        NKTATFSISFEPYADTVLRTHETSVVR

STAHIRVLRDMVPGDEITCFYGSEFF         FFESGYFCTIEDGPVMDRLVRIHATS
```

```
AHTRRCPDHEPRGVITYL                LYTIVGRPEHDPCRRTHA

  4 record(s) selected.
```

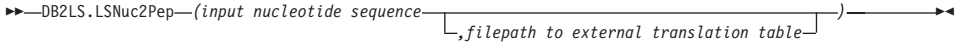
相关参考:

- 第 68 页的『LSRevPep 用户定义的函数』

转换

转换用户定义的函数将核苷酸序列转换为缩氨酸序列。

LSNuc2Pep 用户定义的函数



input nucleotide sequence

描述核苷酸序列的有效字符串表示法。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

filepath to external translation table

如果使用定制的转换表，则包括文件路径信息以查找转换表。此路径的字符串值一定不能超过 255 个字符。

模式名是 DB2LS。

此函数的结果是具有 VARCHAR 数据类型并且实际长度不大于 10890 字节的字符串，它表示缩氨酸序列。

输入是使用 IUB 字符集的核苷酸序列。此函数假定第一个基码从核苷酸序列的第一个字符处开始。如果第一个基码不是从核苷酸序列的第一个字符开始的，则对输入序列使用 SUBSTR 函数。

此函数的结果是使用标准氨基酸符号的缩氨酸序列。

此函数:

- 删除输入序列中的空格。
- 忽略读码框架之外的外来核苷酸。
- 如果输入空的核苷酸序列，则返回空输出。

相关参考:

- 第 71 页的『LSNuc2Pep 用户定义的函数 - 示例』
- 第 72 页的『LSTransAllFrames 用户定义的函数』

LSNuc2Pep 用户定义的函数 – 示例

假定要将核苷酸序列数据转换为缩氨酸序列。此示例假定第一个基码以核苷酸序列的首字符开始。

可以使用 values 语句来调用此函数。唯一的输入是核苷酸序列，如以下示例所示:

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

以上语句的结果是使用标准氨基酸符号的缩氨酸序列:

```
FFLLSSSSYFLCC*C
```

如果要使用 +2 读码框架来进行转换，则使用以下示例:

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

语句中的整数指示开始搜索基码的位置。

这里是在查询中使用此函数作为谓词的一个示例。

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

表 37 显示了结果。

表 37. 使用 LSNuc2Pep 函数作为谓词的结果

ID	PROTEINNAME	PEPTIDESEQ
1	proteinA	FSYCLPHRISYVAD

以下示例使用外部转换表来将核苷酸序列转换为缩氨酸序列。第一个参数是核苷酸序列，第二个参数是外部转换表的路径。

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

以上使用这个特定转换表的语句的结果是以下字符串:

```
FSYCLPHRISYVAD
```

以下示例将两个用户定义的函数组合起来使用以演示这些函数的其它用途:

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

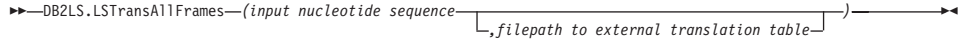
注意，上一个示例返回与以下查询相同的结果:

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

相关参考:

- 第 68 页的『LSRevNuc 用户定义的函数 - 示例』
- 第 73 页的『LSTransAllFrames 用户定义的函数 - 示例』
- 第 70 页的『LSNuc2Pep 用户定义的函数』

LSTransAllFrames 用户定义的函数



input nucleotide sequence

描述核苷酸序列的有效字符串表示法。输入序列可以包含 IUPAC 多义代码。字符串表示法必须具有 VARCHAR 数据类型，并且实际长度不大于 32672 字节。

filepath to external translation table

如果使用定制的转换表，则包括文件路径信息以查找转换表。此路径的字符串值一定不能超过 255 个字符。

模式名是 DB2LS。

使用 LSTransAllFrames 用户定义的函数来从给定的核苷酸序列生成一组缩氨酸序列。这些缩氨酸序列表示输入核苷酸序列的可能转换（6 个框架为一组）。当输入包含错误或未知读码框架时，此函数非常有用。

此函数的结果是带有 2 个列的表。第一列的标号是 READFRAME，它表示用于转换的框架。此列包含表示转换的起始位置的整数值。负整数指示反向的转换。第二列（称为 PEPTIDE）是具有 VARCHAR 数据类型并且实际长度不大于 10890 字节的字符串，它表示缩氨酸序列。

此函数:

- 删除输入序列中的空格。
- 忽略读码框架之外的外来核苷酸。
- 如果输入空的核苷酸序列，则返回空输出。

相关参考:

- 第 73 页的『LSTransAllFrames 用户定义的函数 - 示例』
- 第 70 页的『LSNuc2Pep 用户定义的函数』

LSTransAllFrames 用户定义的函数 – 示例

假定您想要使用内置转换表来转换核苷酸序列，并且全都以 6 个读码框架为一组。以下示例显示了如何做到这一点:

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

此查询在一个表中返回缩氨酸，如以下示例所示:

表 38. 转换核苷酸序列的结果

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

下一个示例使用定制的转换表来转换核苷酸序列，并且全都以 6 个读码框架为一组。

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

因为输入序列是相同的，并且转换表与构建到函数中的转换表相同，所以结果表与上一个示例的结果表相同。

以下示例将两个用户定义的函数组合起来使用以演示这些函数的其它用途:

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

注意，上一个示例返回与以下查询相同的结果:

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

以下示例从 LSTransAllFrames 函数生成的输出中选择特定的读码框架。

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

此查询的结果为:

表 39. Readframe 函数的使用

READFRAME	PEPTIDE
–2	HQQHKKYDEEDNKK

相关参考:

- 第 71 页的『LSNuc2Pep 用户定义的函数 – 示例』
- 第 68 页的『LSRevNuc 用户定义的函数 – 示例』
- 第 72 页的『LSTransAllFrames 用户定义的函数』

基码频率表格式

基码频率表显示氨基酸向后转换为特定基码时采用的频率。LSPep2ProbNuc 用户定义的函数使用基码频率表来根据给定的缩氨酸序列确定核苷酸序列。

以下列表描述了基码频率表文件的格式:

- 两个相邻的句点标记了表的开头。之前的任何文本都是注释。两个相邻的句点是必需的，即使它们前面没有注释亦如此。
- 此表包含下列各列:
  1. Am-Acid: 氨基酸符号的三字母代码。
  2. Codon: 该氨基酸符号的基码。
  3. Number: 该基码在编译生成该表的基因中的出现次数。
  4. x/1000: 氨基酸的预期出现次数，基因中的每 1000 个转换的基码对。
  5. Fraction: 基码在其同义基码系列中的出现的分数。

产品在 sqllib/samples/lifesci/ls_udfs 子目录中提供了样本基码频率表。

相关参考:

- 第 44 页的『LSPep2ProbNuc 用户定义的函数』
- 第 74 页的『基码频率表 – 示例』

基码频率表 – 示例

第 75 页的图 2 显示了样本基码频率表的格式。

```
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
...（截断）
```

图 2. 样本基码频率表

相关参考:

- 第 44 页的『LSPep2ProbNuc 用户定义的函数』
- 第 74 页的『基码频率表格式』

转换表格式

本主题描述 LSPep2AmbNuc、LSTransAllFrames 和 LSNuc2Pep 生命科学用户定义的函数使用的转换表的格式。

以下列表描述了基码频率表文件的格式:

- 两个相邻的句点标记表的开头。在此之前的任何文本都是注释。
- 表的每一行都由单字母氨基酸符号、三字母氨基酸名称、确定的基码、感叹号和不确定的基码组成。用空格将行中的每个字分隔开。
- 每个基码和氨基酸符号在文件中都只能出现一次。
- 停止基码转换为符号“*”。
- 由小写字母组成的基码是起始基码。
- 所有其它基码都是大写的。
- 将无法转换为相应氨基酸符号的基码转换为符号“X”。

产品在 sqllib/samples/lifesci/ls_udfs 子目录中提供了样本转换表。

转换表 – 示例

图 3 显示了样本转换表的格式。

```
                Standard Translation Table

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
```

图 3. 样本转换表

易使用性

身体有某些缺陷（如活动不方便或视力不太好）的用户可通过易使用性功能来成功使用软件产品。以下是 DB2 Information Integrator V8 中主要的易使用性功能:

- 通过键盘即可对所有功能进行操作，而不必使用鼠标。
- 您可以定制字体的大小和颜色。
- 您可以接收可视或声音警告提示。
- DB2 支持使用 Java™ Accessibility API 的易使用性应用程序。
- B2 文档以易使用的格式提供。

键盘输入和导航

只使用键盘就可对 DB2 数据库工具（如控制中心、数据仓库中心和复制中心）进行操作。使用键或组合键就可执行大多数操作，而不必使用鼠标。

在基于 UNIX 的系统中，键盘焦点的位置是突出显示的。此突出显示指示窗口的哪个区域处于活动状态且击键对何处生效。

易使用的界面显示

DB2 数据库工具中的功能增强了用户界面，使视力不太好的用户更易使用。这些易使用性方面的增强包括了对可定制字体属性的支持。

字体设置

对于 DB2 数据库工具，可以使用“工具设置”笔记本来选择菜单和窗口中文本的颜色、大小和字体。

与颜色的不相关性

不需要分辨颜色就可以使用此产品中的任何功能。

备用的警告提示

可使用“工具设置”笔记本来指定是否想要通过声音提示或可视提示接收警告。

与辅助技术的兼容性

DB2 Information Integrator 图形界面支持 Java Accessibility API，它支持使用屏幕阅读器以及其它由有某些缺陷的用户使用的辅助性技术。

可访问文档

DB2 产品系列的文档提供了 HTML 格式的版本。您可以根据浏览器中设置的显示首选项来查看文档。可使用屏幕阅读器和其它辅助性技术。

声明

本信息是为在美国提供的产品和服务编写的。IBM 可能在其它国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: 国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是默示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证。因此，本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息，而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

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

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可证协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其它操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其它可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其它关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本资料中可能包含用于日常业务运作的数据和报表的示例。为了尽可能完整地说明问题，这些示例可能包含个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有雷同，纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、营销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年*）。此部分代码是根据 IBM 公司样本程序衍生出来的。© Copyright IBM Corp. （*输入年份*）。All rights reserved.

商标

下列各项是国际商业机器公司在美国和／或其它国家或地区的商标:

IBM
AIX
DB2
Domino
Informix
Lotus
Lotus Notes
QuickPlace
WebSphere

下列各项是其它公司的商标或注册商标:

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其它国家或地区的商标。

UNIX 是 The Open Group 在美国和其它国家或地区的注册商标。

Java 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc. 在美国和／或其它国家或地区的商标或注册商标。

其它公司、产品或服务名称可能是其它公司的商标或服务标记。

索引

与 IBM 联系

要在美国或加拿大联系 IBM，请致电以下号码之一:

- 客户服务中心: 1-800-IBM-SERV (1-800-426-7378)
- DB2 市场营销和销售中心: 1-800-IBM-4YOU (1-800-426-4968)

要了解所提供的服务项目，请致电以下号码之一:

- 在美国: 1-888-426-4343
- 在加拿大: 1-800-465-9600

要查找您所在国家或地区的 IBM 分部，可查看 “IBM 全球联系人目录”（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

产品信息

还可通过电话或 Web 获取关于 DB2 Information Integrator 的信息。

您如果住在美国，请致电以下号码之一:

- 订购产品或获取一般信息，请致电: 1-800-IBM-CALL (1-800-426-2255)
- 订购出版物，请致电: 1-800-879-2755

在 Web 上，访问 www.ibm.com/software/data/integration。此站点包含有关技术库、订购书籍、客户机下载、新闻组、修订包、新闻和 Web 资源链接的最新信息。

要查找您所在国家或地区的 IBM 分部，可查看 “IBM 全球联系人目录”（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

关于文档的意见

您的反馈将有助于 IBM 提供质量信息。请发送您对该书或其它 DB2 Information Integrator 文档的任何意见。可使用以下任一方法提供意见:

- 使用在线读者意见表发送意见，网址为 www.ibm.com/software/data/rcf。
- 将您的意见以电子邮件方式发送至 comments@us.ibm.com。确定包括产品名称、产品版本号以及书籍的名称和部件号（如果可应用的话）。如果您的意见针对特定文本，请包括该文本的位置（例如，标题、表号或页码）。

中国印刷