IBM DB2 Information Integrator

Konfiguration von Datenquellen - Addendum: BioRS-Wrapper und benutzerdefinierte Funktionen von Life Sciences

Version 8

IBM DB2 Information Integrator

Konfiguration von Datenquellen - Addendum: BioRS-Wrapper und benutzerdefinierte Funktionen von Life Sciences

Version 8

Hinweis:

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Bemerkungen" auf Seite 95 gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business-Symbol ist eine Marke der International Business Machines Corporation.
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Information Integrator Addendum to the Data Source Configuration Guide: BioRS Wrapper and Life Sciences User-Defined Functions, Version 8,

herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
Juli 2003

Inhaltsverzeichnis

Kapitel 1. Konfigurieren des Zugriffs auf BioRS-Datenquellen

In diesem Kapitel wird erläutert, was BioRS ist und wie BioRS-Datenquellen dem System zusammengeschlossener Datenbanken hinzugefügt werden. Außerdem werden die Fehlernachrichten aufgeführt, die bei der Arbeit mit dem BioRS-Wrapper ausgegeben werden können.

Was ist BioRS?

BioRS ist ein von Biomax Informatics entwickeltes System zum Abfragen und Abrufen von Daten. Mit Hilfe von BioRS können Sie Informationen aus mehreren Datenquellen (einschließlich Flachdateien und relationalen Datenbanken) abrufen. Allgemein zugängliche Daten wie beispielsweise aus SwissProt und GenBank werden normalerweise als Flachdateien in das BioRS-System heruntergeladen. BioRS kann öffentlich zugängliche Datenquellen und proprietäre Datenquellen (beispielsweise nicht öffentliche Datenbanken, die von Ihrem Unternehmen verwaltet werden) in eine einheitliche Umgebung integrieren.

Nach der Integration einer Datenquelle in das BioRS-System wird die betreffende Datenquelle als *Datenbank* bezeichnet. Die unter einem Datenbankeintrag enthaltenen Elemente werden zusammengefasst als *Schema* bezeichnet. In einer BioRS-Abfrage können lediglich diejenigen Elemente einer Datenbank verwendet werden, die im BioRS-System indexiert sind. Zwischen Einträgen in Datenbanken können Abhängigkeiten erstellt werden, sodass sich Datenbanken im BioRS-System miteinander verbinden lassen.

BioRS-Datenbanken können über Eltern-Kind-Abhängigkeiten verfügen, das heißt, sie können verschachtelt sein. In einer solchen Abhängigkeit enthält die untergeordnete Datenbank (Kind) ein Element namens PARENT vom Datentyp 'Reference' (Verweis). Das Element PARENT verweist auf das Element _ID_ der übergeordneten Datenbank (Elter). Abgesehen davon, dass dieses vordefinierte Element PARENT vorhanden ist, enthalten verschachtelte Datenbanken die gleichen Daten wie nicht verschachtelte Datenbanken.

BioRS stellt eine webbasierte Schnittstelle zur Verfügung, mit deren Hilfe Benutzer Abfragen nach Daten in BioRS-Datenbanken ausführen können. Der BioRS-Wrapper verwendet zum Ausführen von Abfragen die gleichen Anwendungsprogrammierschnittstellen (Application Programming Interfaces, APIs) wie die webbasierte BioRS-Schnittstelle.

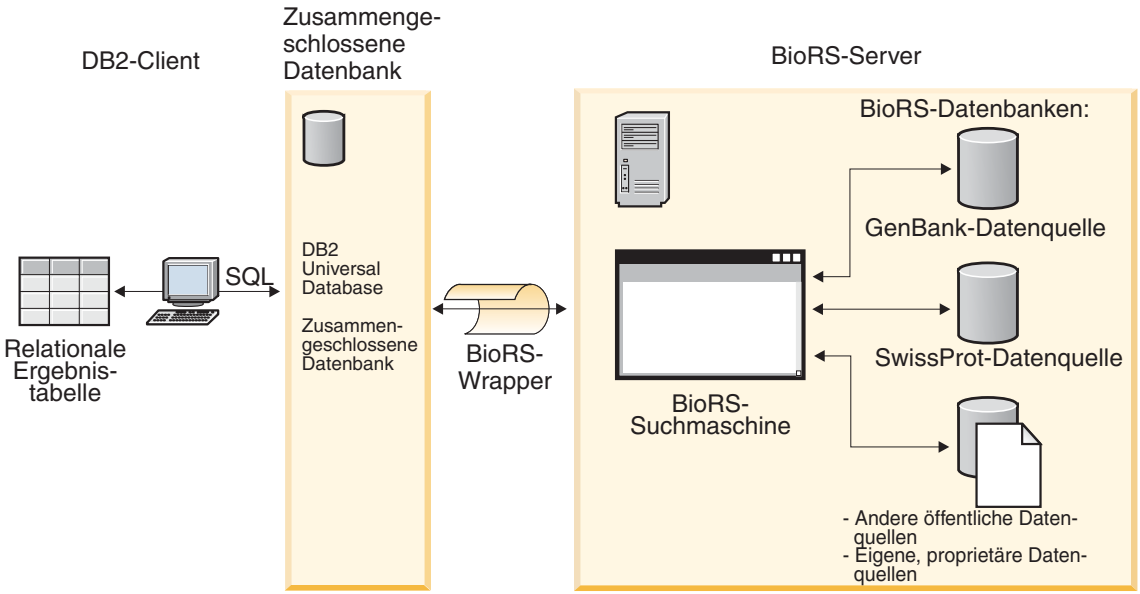


Abbildung 1. Funktionsweise des BioRS-Wrappers

Vom Client aus können Benutzer oder Anwendungen Abfragen mit Hilfe von SQL-Anweisungen übergeben. Anschließend werden die Abfragen an das System zusammengeschlossener Datenbanken gesendet, in dem der BioRS-Wrapper installiert ist. Je nach Aufbau der Abfragen können diese entweder mit DB2® Universal Database oder dem BioRS-Server verarbeitet werden. Der BioRS-Server muss sich nicht auf demselben Computer befinden wie das System zusammengeschlossener Datenbanken. Für jede Abfrage muss das System zusammengeschlossener Datenbanken dem BioRS-Server entsprechende Authentifizierungsinformationen zur Verfügung stellen. Bei diesen Informationen kann es sich entweder um eine Kombination aus Benutzer-ID und Kennwort handeln oder um eine nicht authentifizierte Angabe (normalerweise ein Gastaccount).

Der BioRS-Wrapper funktioniert mit BioRS Version 5.0.14.

Ausführliche Informationen zu BioRS finden Sie auf der Website von Biomax unter: http://www.biomax.com

Zugehörige Tasks:

- „Hinzufügen von BioRS zu einem System zusammengeschlossener Datenbanken" auf Seite 3

Zugehörige Referenzen:

- „BioRS-Wrapper - Beispielabfragen" auf Seite 22

Hinzufügen von BioRS zu einem System zusammengeschlossener Datenbanken

Sie können auf dem Server mit zusammengeschlossenen Datenbanken mit einer BioRS-Datenquelle arbeiten, indem Sie angepasste Funktionen und einen BioRS-Wrapper registrieren. Anschließend müssen ein zugehöriger BioRS-Server, Benutzerzuordnungen und Kurznamen registriert werden, um dem Server mit zusammengeschlossenen Datenbanken das Abrufen und Verarbeiten von BioRS-Daten zu ermöglichen.

Sie können die SQL-Anweisungen von der DB2-Steuerzentrale oder vom DB2-Befehlszeilenprozessor aus ausführen. Nachdem Sie BioRS dem System zusammengeschlossener Datenbanken hinzugefügt haben, können Sie Abfragen für BioRS-Datenquellen ausführen.

Vorgehensweise:

Um eine BioRS-Datenquelle einem Server mit zusammengeschlossenen Datenbanken hinzuzufügen, gehen Sie wie folgt vor:

1. Registrieren Sie angepasste Funktionen mit Hilfe der Anweisung CREATE FUNCTION.
2. Registrieren Sie den BioRS-Wrapper mit Hilfe der Anweisung CREATE WRAPPER.
3. Optional: Definieren Sie die Umgebungsvariable DB2_DJ_COMM zur Verbesserung der Abfrageleistung.
4. Registrieren Sie den BioRS-Server mit Hilfe der Anweisung CREATE SERVER.
5. Optional: Registrieren Sie berechtigte Benutzer mit Hilfe der Anweisung CREATE USER MAPPING.
6. Registrieren Sie Kurznamen mit Hilfe der Anweisung CREATE NICKNAME.
7. Optional: Aktualisieren Sie die Kardinalitätsstatistiken für BioRS-Spalten.

Zugehörige Tasks:

- „Registrieren von angepassten Funktionen für den BioRS-Wrapper" auf Seite 4
- „Registrieren des BioRS-Wrappers" auf Seite 5
- „Festlegen der Profilvariablen DB2_DJ_COMM für den BioRS-Wrapper" auf Seite 6
- „Registrieren des Servers für BioRS-Datenquellen" auf Seite 7
- „Registrieren von Benutzerzuordnungen für BioRS-Datenquellen" auf Seite 7
- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten" auf Seite 13

Registrieren von angepassten Funktionen für den BioRS-Wrapper

Die Registrierung von angepassten Funktionen für den BioRS-Wrapper ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Nachdem Sie die angepassten Funktionen registriert haben, muss der Wrapper registriert werden.

Angepasste Funktionen können mit Hilfe der Beispieldatei 'create_function_mappings.ddl' registriert werden. Diese Datei befindet sich im Verzeichnis 'sqllib/samples/lifesci/biors'. Die Datei 'create_function_mappings.ddl' enthält Definitionen für die einzelnen angepassten Funktionen. Sie können diese DDL-Datei ausführen, um die angepassten Funktionen für jede DB2-Datenbank zu registrieren, in der der BioRS-Wrapper installiert ist.

Voraussetzungen:

- Sämtliche angepassten Funktionen für den BioRS-Wrapper müssen mit dem Schemanamen BioRS registriert werden.
- Jede angepasste Funktion muss einmal für jede DB2-Datenbank registriert werden, in der der BioRS-Wrapper installiert ist.

Vorgehensweise:

Um angepasste Funktionen zu registrieren, setzen Sie die Anweisung CREATE FUNCTION mit dem Schlüsselwort AS TEMPLATE ab.

Der vollständig qualifizierte Name jeder Funktion lautet BioRS.<funktionsname>.

Im folgenden Beispiel wird eine Version der Funktion CONTAINS registriert:

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

Die nächste Task in dieser Tasksequenz ist das Registrieren des entsprechenden BioRS-Wrappers.

Zugehörige Tasks:

- „Registrieren des BioRS-Wrappers" auf Seite 5

Zugehörige Referenzen:

- „CREATE FUNCTION (Sourced or Template) statement" in *SQL Reference, Volume 2*
- „Angepasste Funktionen und BioRS-Abfragen" auf Seite 16
- „BioRS-Wrapper - Beispielabfragen" auf Seite 22
- „Tabelle der angepassten Funktionen - BioRS-Wrapper" auf Seite 33

Registrieren des BioRS-Wrappers

Die Registrierung des BioRS-Wrappers ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Für den Zugriff auf eine Datenquelle muss der Wrapper registriert werden. Wrapper sind Mechanismen, mit deren Hilfe Server mit zusammengeschlossenen Datenbanken mit Datenquellen kommunizieren und Daten aus den Datenquellen abrufen. Wrapper werden auf dem System als Bibliotheksdateien installiert. Tabelle 1 enthält eine Liste der standardmäßigen BioRS-Bibliotheksdateien und der für die einzelnen Dateien jeweils unterstützten Betriebssysteme.

Tabelle 1. BioRS-Bibliotheksdateien und unterstützte Betriebssysteme

BioRS-Bibliotheksdatei	Betriebssystem
libdb2lsbiors.a	IBM AIX Version 4.3.3 oder höher
db2lsbiors.dll	Microsoft Windows NT Version 4 Microsoft Windows 2000 Microsoft Windows XP

Vorgehensweise:

Um den BioRS-Wrapper zu registrieren, setzen Sie die Anweisung CREATE WRAPPER ab.

Um beispielsweise einen BioRS-Wrapper mit dem Namen `wrap_biors` aus der Standardbibliotheksdatei `libdb2lsbiors.a` unter AIX zu registrieren, setzen Sie die folgende Anweisung ab:

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

Zugehörige Tasks:

- „Überprüfen der Bibliotheken der nicht relationalen Wrapper und benutzerdefinierten Life Sciences-Funktionen" in *DB2 Information Integrator Installation*
- „Festlegen der Profilvariablen DB2_DJ_COMM für den BioRS-Wrapper" auf Seite 6

Zugehörige Referenzen:

- „CREATE WRAPPER statement" in *SQL Reference, Volume 2*

Festlegen der Profilvariablen DB2_DJ_COMM für den BioRS-Wrapper

Das Festlegen der DB2-Profilvariablen DB2_DJ_COMM für den BioRS-Wrapper ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Zur Verbesserung der Leistung beim Zugriff auf BioRS-Datenquellen haben Sie die Möglichkeit, die DB2-Profilvariable DB2_DJ_COMM zu definieren. Diese Variable gibt an, ob der Server mit zusammengeschlossenen Datenbanken den Wrapper bei der Initialisierung lädt.

Die Prozessorbelegung ist größer, wenn der Server mit zusammengeschlossenen Datenbanken die Wrapperbibliotheken während des Datenbankstarts lädt. Um eine übermäßige Belegung zu vermeiden, geben Sie nur die Bibliotheken an, auf die Sie zugreifen möchten.

Vorgehensweise:

Um die DB2-Profilvariable DB2_DJ_COMM zu definieren, setzen Sie den Befehl **db2set** mit der Wrapperbibliothek ab, die dem Wrapper entspricht, der in der entsprechenden Anweisung CREATE WRAPPER angegeben wurde.

Beispiel:

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

Stellen Sie sicher, dass auf keiner Seite des Gleichheitszeichens (=) Leerzeichen angegeben sind.

Die nächste Task in dieser Tasksequenz ist das Registrieren des Servers für BioRS.

Zugehörige Konzepte:

- „Umgebungsvariablen und die Profilregistrierdatenbank" in *Systemverwaltung: Implementierung*

Zugehörige Tasks:

- „Registrieren des Servers für BioRS-Datenquellen" auf Seite 7

Zugehörige Referenzen:

- „db2set - DB2 Profile Registry Command" in *Command Reference*

Registrieren des Servers für BioRS-Datenquellen

Die Registrierung des Servers für eine BioRS-Datenquelle ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Nach der Registrierung des Wrappers muss ein entsprechender Server registriert werden.

Vorgehensweise:

Um den BioRS-Server für das System zusammengeschlossener Datenbanken zu registrieren, setzen Sie die Anweisung CREATE SERVER ab.

Beispiel:

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

Zugehörige Tasks:

- „Registrieren von Kurznamen für BioRS-Datenquellen” auf Seite 9

Zugehörige Referenzen:

- „Anweisungsoptionen für CREATE SERVER - BioRS-Wrapper” auf Seite 41

Registrieren von Benutzerzuordnungen für BioRS-Datenquellen

Die Registrierung von Benutzerzuordnungen ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Je nachdem, welche Zugriffsmethode für Accounts im BioRS-System verwendet wird, ist es möglicherweise nicht erforderlich, Benutzerzuordnungen zu erstellen.

- Ist der BioRS-Server für Gastzugriff für alle Benutzeraccounts konfiguriert, müssen in DB2 Information Integrator keine Benutzerzuordnungen erstellt werden.
- Ist der BioRS-Server für die Authentifizierung von Benutzeraccounts durch IDs und Kennwörter konfiguriert, müssen Sie in Ihrer zusammengeschlossenen Datenbank Benutzerzuordnungen für die Accounts erstellen, die mit dem BioRS-Wrapper arbeiten müssen.
- Ist der BioRS-Server für die Verwendung sowohl von Gast- als auch von authentifizierten Benutzeraccounts konfiguriert, müssen Sie in Ihrer zusammengeschlossenen Datenbank Benutzerzuordnungen für diejenigen authentifizierten Benutzeraccounts erstellen, die mit dem BioRS-Wrapper arbeiten müssen.

Benutzerzuordnungen bieten eine Möglichkeit für die Authentifizierung des Zugriffs von Benutzern oder Anwendungen, die eine BioRS-Datenquelle mit dem BioRS-Wrapper abfragen.

Übergibt ein Benutzer oder eine Anwendung eine SQL-Abfrage an einen registrierten BioRS-Kurznamen, ohne dass für den betreffenden Benutzer oder die betreffende Anwendung Benutzerzuordnungen definiert sind, so verwendet der BioRS-Wrapper eine standardmäßige Benutzer-ID mit Kennwort und versucht, Daten vom fernen BioRS-Server abzurufen. Ist für eine abgefragte Datenbank eine Authentifizierung erforderlich, wird unter Umständen eine Fehlernachricht zurückgegeben.

Um sicherzustellen, dass die korrekte Benutzer-ID mit Kennwort an den BioRS-Server übergeben wird, erstellen Sie in Ihrem System zusammengeschlossener Datenbanken Benutzerzuordnungen für Benutzer, die für die Suche in BioRS-Datenquellen berechtigt sind. Wenn Sie eine Benutzerzuordnung erstellen, wird das Kennwort in verschlüsseltem Format in einer Katalogtabelle des Systems zusammengeschlossener Datenbanken gespeichert.

Vorgehensweise:

Um BioRS-Benutzerzuordnungen zu registrieren, verwenden Sie die Anweisung CREATE USER MAPPING.

Mit der folgenden Anweisung CREATE USER MAPPING beispielsweise wird der Benutzer 'Charlie' dem Benutzer 'Charlene' auf dem Server 'Biors_Server1' zugeordnet.

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

Sie können auch eine eigene Benutzerzuordnung definieren. In folgendem Beispiel ist USER ein Schlüsselwort, das den aktuellen Benutzer identifiziert, und kein Benutzername von USER.

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

Die nächste Task in dieser Tasksequenz ist das Registrieren von Kurznamen für den BioRS-Wrapper.

Zugehörige Tasks:

- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9

Zugehörige Referenzen:

- „CREATE USER MAPPING statement" in *SQL Reference, Volume 2*
- „Anweisungsoptionen für CREATE USER MAPPING - BioRS-Wrapper" auf Seite 42

Registrieren von Kurznamen für BioRS-Datenquellen

Das Registrieren von Kurznamen für BioRS-Datenquellen ist Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken. Nach der Registrierung eines Servers muss ein Kurzname für jede BioRS-Datenquelle registriert werden, auf die zugegriffen werden soll. Kurznamen werden in Abfragen zum Verweisen auf BioRS-Datenquellen verwendet.

Wichtig: Nach der Integration einer Datenquelle in das BioRS-System wird die betreffende Datenquelle als *Datenbank* bezeichnet. BioRS-Datenbanken entsprechen Kurznamen in einem System zusammengeschlossener Datenbanken.

Voraussetzungen:

- Falls der Name einer BioRS-Datenbank nicht der DB2-Syntax zusammengeschlossener Systeme entspricht, müssen Sie beim Registrieren des Kurznamens die Kurznamenoption REMOTE_OBJECT verwenden.
- Falls der Name eines BioRS-Elements nicht der DB2-Syntax zusammengeschlossener Systeme entspricht, müssen Sie beim Registrieren des Kurznamens die Spaltenoption ELEMENT_NAME verwenden.

Einschränkungen:

Sie dürfen das BioRS-Element 'AllText' nicht als erste Spalte eines Kurznamens verwenden. In allen anderen Spaltenpositionen (beispielsweise als zweite oder dritte Spalte) kann das BioRS-Element 'AllText' verwendet werden.

Vorgehensweise:

Um einen BioRS-Kurznamen zu registrieren, verwenden Sie die Anweisung CREATE NICKNAME.

Ein Kurzname für zusammengeschlossene Systeme entspricht direkt einer BioRS-Datenbank. Wenn Sie einen Kurznamen für zusammengeschlossene Systeme erstellen, definieren Sie eine Liste von Kurznamenspalten. Die angegebenen Kurznamenspalten müssen den Elementen eines bestimmten BioRS-Datenbankformats entsprechen. BioRS definiert fünf mögliche Datentypen für Elemente: Text, Number (Nummer), Date (Datum), Author (Autor) und Reference (Verweis). Diese Datentypen können lediglich den Datentypen CLOB, CHAR und VARCHAR des Systems zusammengeschlossener Datenbanken zugeordnet werden.

Am einfachsten lässt sich ein Kurzname für eine BioRS-Datenbank registrieren, wenn der Kurzname denselben Namen erhält wie die BioRS-Datenbank. Beispiel:

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

Die zugrunde liegende BioRS-Datenbank 'SwissProt' ist der Name des Kurznamens.

Durch Verwendung dieser einfachen Syntax für CREATE NICKNAME werden die Kurznamen auf eine einzige Kurznamenfamilie pro DB2-Schema begrenzt. Sie können jedoch auch andere Namen verwenden, indem Sie die Option REMOTE_OBJECT angeben. Diese Kurznamenoption gibt den Namen des BioRS-Objekttyps an, der dem Kurznamen zugeordnet werden soll. Der in der Option REMOTE_OBJECT angegebene Name legt das Schema und die BioRS-Datenbank für den Kurznamen fest. Außerdem gibt die Option REMOTE_OBJECT die Abhängigkeit zwischen dem Kurznamen und anderen Kurznamen an.

Das folgende Beispiel enthält die gleiche Gruppe von Kurznamenmerkmalen wie das vorherige Beispiel. In diesem Beispiel wird der Name des Kurznamens jedoch geändert, und es wird die Option REMOTE_OBJECT verwendet, um die BioRS-Datenbank anzugeben, für die der Kurzname definiert wird:

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

Die zugrunde liegende BioRS-Datenbank ist 'SwissProt', und der Name des Kurznamens lautet 'NewSP'.

Zugehörige Konzepte:

- „Statistische BioRS-Daten" auf Seite 28

Zugehörige Tasks:

- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen" auf Seite 30

Zugehörige Referenzen:

- „Das BioRS-Element 'AllText'" auf Seite 32
- „Anweisung CREATE NICKNAME - Beispiele für den BioRS-Wrapper" auf Seite 11

- „Anweisungssyntax für CREATE NICKNAME - BioRS-Wrapper" auf Seite 39
- „Überlegungen zum Ändern von Kurznamen - BioRS-Wrapper" auf Seite 32

Anweisung CREATE NICKNAME - Beispiele für den BioRS-Wrapper

Dieser Abschnitt enthält Beispiele, die zeigen, wie die Anweisung CREATE NICKNAME verwendet wird, um Kurznamen für den BioRS-Wrapper zu registrieren.

Beispiel 1:

Das folgende Beispiel zeigt, wie ein Kurzname für eine ferne BioRS-Datenbank erstellt wird, die nicht der Syntax von DB2 Information Integrator entspricht:

```
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');
```

Der Name dieses Kurznamens lautet 'SwissFT'. Die Tabellenspalten sind ID, ALLTEXT, ENTRYDATE und FtLength. Die Spaltenoption ELEMENT_NAME ist für die ID-Spalte angegeben. Die Option ELEMENT_NAME muss angegeben werden, wenn der Name eines BioRS-Elements nicht der gültigen DB2-Syntax zusammengeschlossener Systeme für Spaltennamen entspricht. In diesem Beispiel entspricht das BioRS-Element _ID_ zwar der DB2-Syntax zusammengeschlossener Systeme, doch ist _ID_ ein möglicherweise verwirrender Name für Benutzer von DB2 Information Integrator. Der Name ID ist einfach und leicht verständlich. Im Allgemeinen wird die Option ELEMENT-_NAME unter folgenden Umständen verwendet:

- Wenn der Name eines BioRS-Elements nicht der gültigen DB2-Syntax zusammengeschlossener Systeme entspricht;
- Wenn die Groß-/Kleinschreibung des Namens eines BioRS-Elements nicht den erstellten DB2-Standards des Systems zusammengeschlossener Datenbanken entspricht;
- Wenn der Name eines BioRS-Elements für Benutzer von DB2 Information Integrator möglicherweise nicht eindeutig ist.

Außerdem wird mit der Option REMOTE_OBJECT der Name der BioRS-Datenbank angegeben, dem der Kurzname entspricht. Die Option REMOTE_OBJECT muss angegeben werden, wenn der Name einer BioRS-Datenbank nicht der gültigen DB2-Syntax zusammengeschlossener Systeme entspricht. In diesem Beispiel entspricht der Datenbankname "SwissProt.Features" nicht der

gültigen DB2-Syntax zusammengeschlossener Systeme. Im Allgemeinen wird die Option REMOTE_OBJECT unter folgenden Umständen verwendet:

- Wenn die Groß-/Kleinschreibung des Namens einer BioRS-Datenbank nicht den erstellten DB2-Standards des Systems zusammengeschlossener Datenbanken entspricht;
- Wenn der Name einer BioRS-Datenbank nicht der gültigen DB2-Syntax zusammengeschlossener Systeme entspricht;
- Wenn der Name einer BioRS-Datenbank für Benutzer von DB2 Information Integrator möglicherweise nicht eindeutig ist.

Beispiel 2:

Das folgende Beispiel zeigt, wie ein Kurzname für eine Tabelle erstellt wird, die eine BioRS-Datenbank verwendet, die mit einer anderen BioRS-Datenbank verbunden ist:

```
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');
```

Der Name dieses Kurznamens lautet 'SwissFT2'. Die Tabellenspalten sind ID, ALLTEXT, FtKey, FtLength, FtDescription und Parent. Die Spaltenoption ELEMENT_NAME ist für die ID-Spalte angegeben. Mit der Option REMOTE_OBJECT wird der Name der BioRS-Datenbank angegeben, dem der Kurzname entspricht.

Die Spalte 'Parent' verwendet außerdem die Option REFERENCED_OBJECT. Sie müssen diese Option für Spalten angeben, die BioRS-Elementen vom Datentyp 'Reference' (Verweis) entsprechen. Die Option REFERENCED_OBJECT gibt den Namen der BioRS-Datenbank an, auf die die Spalte verweist. In diesem Fall verweist das Element 'Parent' auf die BioRS-Datenbank 'SwissProt'.

Zugehörige Tasks:

- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9

Zugehörige Referenzen:

- „Anweisungssyntax für CREATE NICKNAME - BioRS-Wrapper" auf Seite 39
- „Überlegungen zum Ändern von Kurznamen - BioRS-Wrapper" auf Seite 32

Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten

Um die Kardinalitätsstatistiken von BioRS-Spalten im System zusammengeschlossener Datenbanken zu aktualisieren, müssen Sie die Katalogsicht SYSSTAT.COLUMNS modifizieren.

Durch ordnungsgemäße Verwaltung der Kardinalitätsstatistiken von BioRS-Spalten können das Optimierungsprogramm und der BioRS-Wrapper während der Abfrageverarbeitung denjenigen Datenzugriffsplan auswählen, der die beste Leistung erbringt.

Sie haben die Möglichkeit, die Kardinalitätsstatistiken von BioRS-Spalten als Teil der umfangreicheren Task des Hinzufügens von BioRS zu einem System zusammengeschlossener Datenbanken zu aktualisieren. Außerdem können Sie die Kardinalitätsstatistiken von BioRS-Spalten auch dann aktualisieren, wenn Sie die Abfrageleistung für BioRS-Datenquellen verbessern wollen.

Einschränkungen:

Sie dürfen diese Vorgehensweise nicht verwenden, um die Kardinalitätsstatistiken für Spalten zu ändern, die dem BioRS-Element _ID_ entsprechen. Um die Kardinalitätsstatistiken von Spalten zu ändern, die dem BioRS-Element _ID_ entsprechen, ist eine andere Vorgehensweise erforderlich.

Vorgehensweise:

Um die Kardinalitätsstatistiken von BioRS-Spalten zu aktualisieren, setzen Sie die Anweisung UPDATE mit folgender Syntax ab:

```
UPDATE sysstat.columns SET colcard=(SELECT COUNT(DISTINCT <spaltenname>)
                                    FROM <kurnamenschema>.<name_des_kurznamens>)
 WHERE
  tabschema=<kurznamenschema>
  AND tabname=<name_des_kurznamens>
  AND colname=<spaltenname>
```

- *<spaltenname>* ist der Name der Spalte, deren Kardinalitätsstatistiken Sie aktualisieren wollen.
- *<kurznamenschema>* ist der Name des Schemas, das dem Kurznamen zugeordnet ist, in dem die angegebene Spalte verwendet wird.
- *<name_des_kurznamens>* ist der Name des Kurznamens, in dem die angegebene Spalte verwendet wird.

Die Ausführung der Abfrage kann einige Minuten in Anspruch nehmen, da alle Einträge für die im Kurznamen angegebene Datenbank abgerufen werden müssen.

Wenn eine Spalte mehrere Werte enthalten kann (beispielsweise beim Format des Elements 'PublicationYear' der Datenbank 'SwissProt'), wird die Berechnung für die Verwendung von SQL-Abfragen zu komplex. Für solche Spalten müssen Sie den Kardinalitätswert manuell berechnen und die Katalogsicht SYSSTAT.COLUMNS anschließend entsprechende aktualisieren. Um den Kardinalitätswert zu berechnen, dividieren Sie die Anzahl der unterschiedlichen Werte in der Spalte durch die durchschnittliche Anzahl an Werten pro Zeile. Der berechnete Kardinalitätswert kann nicht größer sein als die Kardinalität der Tabelle.

Beispiel:

Angenommen, Sie haben einen Kurznamen mit drei Zeilen. Die Werte der Spalte 'PublicationYear' für diese drei Zeilen lauten wie folgt:
- 1997 1992 1985
- 1997 1992 1982
- 1992 1991 1990 1976 1974 1971

Es gibt neun unterschiedliche Werte, und die durchschnittliche Anzahl an Werten in einer Zeile beträgt vier. Die Kardinalität dieser Spalte 'PublicationYear' ist demnach 9:4 bzw. 3 (2,25 aufgerundet auf die nächsthöhere ganze Zahl). Nachdem Sie nun die Kardinalität berechnet haben, können Sie die Katalogsicht SYSSTAT.COLUMNS mit Hilfe der folgenden Anweisung UPDATE entsprechend aktualisieren:

```
UPDATE sysstat.columns SET colcard=3
 WHERE
  tabschema=<kurznamenschema>
  AND tabname=<name_des_kurznamens>
  AND colname=<spaltenname>
```

- *3* ist der Kardinalitätswert der Spalte.
- <*kurznamenschema*> ist der Name des Schemas, das dem zugrunde liegenden Kurznamen zugeordnet ist, in dem die angegebene Spalte verwendet wird.
- <*name_des_kurznamens*> ist der Name des zugrunde liegenden Kurznamens, in dem die angegebene Spalte verwendet wird.
- <*spaltenname*> ist der Name der Spalte, deren Kardinalitätsstatistiken Sie aktualisieren wollen.

Zugehörige Konzepte:
- „Statistische BioRS-Daten" auf Seite 28

Zugehörige Tasks:
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen" auf Seite 30
- „Aktualisieren der Kardinalität der BioRS-Spalte _ID_" auf Seite 31

Richtlinien zur Leistungsoptimierung des BioRS-Wrappers

Dieser Abschnitt enthält Richtlinien dazu, wie die Abfrageleistung bei Verwendung des BioRS-Wrappers optimiert werden kann.

Minimierung der zwischen Suchmaschinen übertragenen Datenmenge.
Die Umgebung zusammengeschlossener Datenbanken verwendet zwei Steuerkomponenten für die Abfrage. Für den BioRS-Wrapper sind diese Steuerkomponenten 'DB2® Universal Database' und 'BioRS'. Die DB2-Steuerkomponente verarbeitetet Vergleichselemente (Vergleichsoperatoren wie beispielsweise =, BETWEEN, LIKE und <>), die in Kurznamenspalten angegeben sind. Die BioRS-Steuerkomponente verarbeitet Vergleichselemente, die mit Hilfe der vier angepassten Funktionen für den BioRS-Wrapper angegeben werden.

Um die Datenmenge, die zwischen den beiden Suchmaschinen übertragen wird, zu minimieren, strukturieren Sie Ihre Abfragen so, dass die Datenverarbeitung so oft wie möglich im Pushdown-Modus an das BioRS-System gesendet wird.

Wenn Sie in einer Abfrage Verknüpfungsoperationen durchführen müssen, nutzen Sie die in den BioRS-Datenbanken bereits vorhandenen Elter-Kind-Abhängigkeiten und führen Sie so oft wie möglich Verknüpfungsoperationen mit Gleichheitsattributen durch. Verknüpfungsoperationen mit Gleichheitsattributen werden in BioRS verarbeitet, was ebenfalls die Datenmenge minimiert, die zwischen den DB2- und BioRS-Steuerkomponenten für die Abfrage übertragen wird.

Achtung: Sie dürfen DB2 Information Integrator-Abfragen an BioRS nicht unterbrechen, indem Sie beispielsweise **Strg-D** oder **Strg-Z** im Befehlszeilenprozessor eingeben bzw. ein Anwendungsprogramm stoppen. Abgebrochene Abfragen hinterlassen sogenannte ″tote″ Prozesse, die auf dem BioRS-Server ausgeführt werden. Diese ″toten″ Prozesse führen rasch zu einer Beeinträchtigung der Systemleistung sowohl von BioRS als auch von DB2 Information Integrator. Sind viele dieser ″toten″ Prozesse aktiv, können bei der Verarbeitung von Abfragen mit DB2 Information Integrator unerwartete Fehler auftreten. Eine gültige Abfrage beispielsweise könnte 0 Zeilen zurückgeben, obwohl Zeilen erwartet werden. In extremen Situationen kann es passieren, dass BioRS, DB2 Information Integrator oder beide Produkte gestoppt oder abnormal beendet werden.

Verwaltung von statistischen BioRS-Daten in der Umgebung zusammengeschlossener Datenbanken.

In einem System zusammengeschlossener Datenbanken verwendet die zusammengeschlossene Datenbank Katalogstatistiken für Objekte mit Kurznamen, um die Abfrageverarbeitung zu optimieren. Das Verwalten von aktuellen Statistiken über die BioRS-Datenquellen ist von wesentlicher Bedeutung, um die Leistung des BioRS-Wrappers zu optimieren. Wenn sich die statistischen Daten oder Strukturmerkmale eines fernen Objekts, für das ein Kurzname definiert wurde, geändert haben, müssen Sie die Kardinalitätsstatistiken der entsprechenden Kurznamenspalte in Ihrem System zusammengeschlossener Datenbanken aktualisieren. Um die Leistung des BioRS-Wrappers zu optimieren, sollten Sie diese Aktualisierungen in regelmäßigen Abständen in DB2 Information Integrator durchführen.

Zugehörige Konzepte:

- „Optimieren der Abfrageverarbeitung" in *Systeme zusammengeschlossener Datenbanken*
- „Vergleichselemente für Verknüpfungen mit Gleichheitsattributen für den BioRS-Wrapper" auf Seite 20
- „Statistische BioRS-Daten" auf Seite 28

Zugehörige Referenzen:

- „Angepasste Funktionen und BioRS-Abfragen" auf Seite 16
- „BioRS-Wrapper - Beispielabfragen" auf Seite 22

Angepasste Funktionen und BioRS-Abfragen

Die Umgebung zusammengeschlossener Datenbanken verwendet zwei Steuerkomponenten für die Abfrage. Für den BioRS-Wrapper sind diese Steuerkomponenten 'DB2 Universal Database' und 'BioRS'. Mit Hilfe der folgenden vier angepassten BioRS-Funktionen können Sie angeben, dass Vergleichselemente im Pushdown-Modus an die BioRS-Steuerkomponente gesendet werden:

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

Diese vier angepassten Funktionen werden im BioRS-Schema registriert. Sie müssen das BioRS-Schema verwenden, um auf die Funktionen zu verweisen.

Die angepassten Funktionen BIORS.CONTAINS, BIORS.CONTAINS_LE und BIORS.CONTAINS_GE erfordern ein Argument für die Suchbegriffsspalte und ein Argument für den Abfragetext. Das folgende Beispiel zeigt eine Anweisung BIORS.CONTAINS:

```
BIORS.CONTAINS (<suchbegriffsspalte>,<abfragebegriff>)
```

Der Wert des Arguments für die Suchbegriffsspalte muss auf eine indexierte BioRS-Spalte verweisen. Bei Verwendung einer nicht indexierten Spalte wird Fehlernachricht SQL30090N zurückgegeben (″Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig″).

Der Wert des Arguments für den Abfragebegriff kann nur ein Literal, eine Hostvariable oder ein Spaltenbezug sein. Arithmetische Verkettungen und Verkettungen von Zeichenfolgen können nicht verwendet werden. Außerdem darf der Wert des Arguments für den Abfragebegriff nicht NULL sein, und zwar auch dann nicht, wenn die verwendete Suchbegriffsspalte so definiert ist, dass Nullwerte zulässig sind.

Die Groß-/Kleinschreibung ist beim Argument für den Abfragebegriff unerheblich.

Welche Datentypen und Formate des Arguments für den Abfragebegriff gültig sind, hängt vom BioRS-Datentyp der verwendeten Suchbegriffsspalte ab. BioRS definiert fünf mögliche Datentypen: Text, Author (Autor), Date (Datum), Number (Zahl) und Reference (Verweis). Tabelle 2 enthält eine Liste der BioRS-Datentypen und der gültigen Abfragebegriffe für die einzelnen Datentypen.

Tabelle 2. BioRS-Datentypen und gültige Abfragebegriffe für angepasste Funktionen

Datentyp der Suchbegriffsspalte	**Gültiger Abfragebegriff**	**Format**
Text	VARCHAR() oder CHAR()	BioRS-Textbegriff, einschließlich Platzhalterzeichen
Author	VARCHAR() oder CHAR()	BioRS-Autorenverweis im Format ″<nachname>, <initialen>″. ″<nachname>″ ist der Nachname des Autors. ″<initialen>″ sind die Initialen des Autors ohne Punkte. Leerzeichen zwischen dem Komma und den Initialen sind akzeptabel. Alternativ kann <nachname> auch allein angegeben werden, ohne das Komma oder die Initialen.
Date	VARCHAR(), CHAR(), DATE oder TIMESTAMP	Bei Zeichenfolge: Datum im DB2-Format, jjjj/mm/tt.

Tabelle 2. BioRS-Datentypen und gültige Abfragebegriffe für angepasste Funktionen (Forts.)

Datentyp der Suchbegriffsspalte	Gültiger Abfragebegriff	Format
Number	VARCHAR() oder CHAR(), INTEGER, SMALLINT, BIGINT REAL, DOUBLE, DECIMAL	Zahlen im DB2-Format.
Reference	VARCHAR() oder CHAR()	BioRS-Textbegriff.

Alle anderen Kombinationen aus Suchbegriffsspalten mit BioRS-Datentypen und Argumenten für Abfragebegriffe resultieren in Fehlernachricht SQL30090N ("Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig"). Sie können lediglich die in Tabelle 2 auf Seite 17 aufgeführten Kombinationen verwenden.

Das Abfragebegriffsargument für Suchbegriffsspalten mit dem Datentyp 'Text', 'Author' und 'Reference' muss dem Muster der BioRS-Abfragesprache entsprechen. In BioRS können Argumente für Abfragebegriffe aus alphanumerischen Zeichenfolgen und Platzhalterzeichen bestehen. Die Funktion BIORS.CONTAINS unterstützt zwei Platzhalterzeichen: **?** (Fragezeichen) und * (Stern).

Das Platzhalterzeichen **?** entspricht einem einzelnen Zeichen. Beispiel: Das Vergleichselement `BioRS.CONTAINS (description, 'bacteri?')=1` entspricht dem Begriff 'bacteria' (Bakterien), aber nicht dem Begriff 'bacterial' (bakteriell).

Das Platzhalterzeichen * entspricht keinem oder mehreren Zeichen. Beispiel: Das Vergleichselement `BioRS.CONTAINS (description, 'bacteri*')=1` entspricht dem Begriff 'bacteri', 'bacteria' und 'bacterial'.

Die BioRS-Dokumentation enthält ausführliche Informationen zu den Mustern der BioRS-Abfragesprache.

Die Funktion BIORS.CONTAINS kann für alle BioRS-Spaltentypen angegeben werden.

Die angepassten Funktionen BIORS.CONTAINS_GE und BIORS.CONTAINS_LE können lediglich für Spalten angegeben werden, deren zugrunde liegender BioRS-Datentyp 'Number' oder 'Date' ist. Die Funktion BIORS.CONTAINS_GE wählt Zeilen aus, in denen die Spalte einen Wert enthält, der größer-gleich dem Wert ist, der vom Argument für den Abfragebegriff dargestellt wird. Die Funktion BIORS.CONTAINS_LE wählt Zeilen aus,

in denen die Spalte einen Wert enthält, der kleiner-gleich dem Wert ist, der vom Argument für den Abfragebegriff dargestellt wird.

Die Funktionen BIORS.CONTAINS, BIORS.CONTAINS_GE und BIORS.CONTAINS_LE geben ein Ergebnis aus ganzen Zahlen zurück. Wird eine der drei CONTAINS-Funktionen in einem Vergleichselement verwendet, muss der Rückgabewert mit dem Wert 1 unter Verwendung des Operators = oder <> verglichen werden. Beispiel:

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

Ein Ausdruck im Format `NOT (BioRS.Contains (col,value) = 1)` entspricht `BioRS.CONTAINS (col,value) <> 1`.

Durch Absetzen der Funktion BIORS.SEARCH_TERM können Sie Abfragen ausführen, die ansonsten unter Umständen nicht möglich wären. Mit Hilfe dieser Funktion können Sie einen Suchbegriff im BioRS-Format angeben. Die Funktion BIORS.SEARCH_TERM erfordert zwei Argumente. Das erste Argument ist ein Verweis auf die Spalte _ID_ des Kurznamens, auf den der Begriff angewendet werden soll. Das zweite Argument ist eine Zeichenfolge, die den Begriff ohne einen Datenbanknamen enthält.

Im folgenden Beispiel werden alle Spalten für Einträge in der Datenbank MyEMBL ausgewählt, in denen das Element 'SeqLength' einen Wert größer-gleich 100 enthält:

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

In folgendem Beispiel wird die Spalte 'MolWeight' aus dem Kurznamen 'Swiss' ausgewählt, in der der Wert des Elements 'MolWeight' größer-gleich 100368 ist:

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

Wenn Sie die Funktion BIORS.SEARCH_TERM angeben, können Sie in einer Abfrage keine anderen angepassten Funktionen verwenden. Sie können in derselben Abfrage jedoch eine beliebige Kombination aus den Funktionen BIORS.CONTAINS, BIORS.CONTAINS_GE und BIORS.CONTAINS_LE verwenden.

Zugehörige Konzepte:

- „Pushdown-Analyse" in *Systeme zusammengeschlossener Datenbanken*
- „Richtlinien zur Leistungsoptimierung des BioRS-Wrappers" auf Seite 15
- „Vergleichselemente für Verknüpfungen mit Gleichheitsattributen für den BioRS-Wrapper" auf Seite 20

Zugehörige Tasks:

- „Registrieren von angepassten Funktionen für den BioRS-Wrapper" auf Seite 4

Zugehörige Referenzen:

- „BioRS-Wrapper - Beispielabfragen" auf Seite 22
- „Tabelle der angepassten Funktionen - BioRS-Wrapper" auf Seite 33

Vergleichselemente für Verknüpfungen mit Gleichheitsattributen für den BioRS-Wrapper

Mit Hilfe der vier angepassten BioRS-Funktionen können Sie Vergleichselemente für die BioRS-Steuerkomponente angeben. Hierbei gibt es jedoch eine Ausnahme. Diese Ausnahme besteht bei der Ausführung von Verknüpfungsoperationen mit Gleichheitsattributen während einer Abfrage. Eine *Verknüpfungsoperation* (JOIN) umfasst das Abrufen von Daten aus zwei oder mehreren Tabellen auf der Grundlage übereinstimmender Spaltenwerte. Eine *Verknüpfung mit Gleichheitsattributen* ist eine Verknüpfungsoperation, bei der die Verknüpfungsbedingung im Format 'ausdruck = ausdruck' vorliegt. Bei BioRS-Abfragen müssen Bedingungen für Verknüpfungen mit Gleichheitsattributen das Element _ID_ einer Datenbank und ein Element vom Typ 'Reference' (Verweis) einer anderen Datenbank enthalten.

Beispiel:

Dieses Beispiel zeigt Definitionen von Beispielkurznamen sowie eine Abfrage mit einer Verknüpfung mit Gleichheitsattribut, in der die Beispielkurznamen verwendet werden.

Angenommen, Sie wollen zwei BioRS-Datenbanken abfragen: 'SwissProt' und 'SwissProt.features'. Die Datenbank 'SwissProt.features' ist ein untergeordnetes Element (Kind) der Datenbank 'SwissProt' und enthält ein Element namens PARENT. Das Element PARENT enthält Verweise auf Einträge, die durch das SwissProt-Element _ID_ identifiziert werden. Für die beiden Datenbanken werden zwei Kurznamendefinitionen registriert.

Kurznamendefinition 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');
```

Kurznamendefinition 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');
```

Die folgende Abfrage verweist in einer Verknüpfung mit Gleichheitsattribut auf beide Kurznamen:

```
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;
```

In der vorstehenden Abfrage werden zwei Vergleichselemente auf den Kurznamen 'tc600sprot' (Datenbank 'SwissProt') angewendet. Diese beiden Vergleichselemente filtern die Zeilen heraus, die den Begriff 'anopheles' enthalten und deren Publikationsjahr (PublicationYear) 1997 ist. Ein Vergleichselement wird auf den Kurznamen 'tc600feat' (Datenbank 'SwissProt.features') angewendet und filtert die Zeilen heraus, deren Element 'FtKey' den Begriff 'signal' enthält. Beide Kurznamen werden mit Hilfe der Bedingung 'term f.Parent = s.ID' miteinander verknüpft.

Die endgültige Ergebnismenge enthält nur diejenigen Zeilen, die diese Kriterien erfüllen und in denen die Merkmalseinträge auf einen übereinstimmenden Eintrag in der Datenbank 'SwissProt' verweisen.

Zugehörige Konzepte:

- „Richtlinien zur Leistungsoptimierung des BioRS-Wrappers" auf Seite 15

Zugehörige Referenzen:

- „Angepasste Funktionen und BioRS-Abfragen" auf Seite 16
- „BioRS-Wrapper - Beispielabfragen" auf Seite 22

BioRS-Wrapper - Beispielabfragen

Dieser Abschnitt enthält mehrere Beispielabfragen, in denen die Kurznamen 'swiss' und 'swissft' verwendet werden.

Der Kurzname 'swiss' wurde mit Hilfe der folgenden Anweisung CREATE NICKNAME registriert:

```
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');
```

Der Kurzname 'swissft' wurde mit Hilfe der folgenden Anweisung CREATE NICKNAME registriert:

```
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');
```

Die Abfragen und Ergebnisse in Tabelle 3 zeigen, wie Sie Ihre Abfragen strukturieren können, um die Verteilung der Verarbeitungslast zwischen dem System zusammengeschlossener Datenbanken und dem BioRS-Server zu optimieren.

Tabelle 3. Beispiele verschiedener Abfragen mit identischen Ergebnissen

Abfrage	Ergebnis
select s.id from Swiss s where biors.CONTAINS(s.id, '100K_RAT') = 1 fetch first 3 rows only	ID---------- 100K_RAT 1 Satz/Sätze ausgewählt.
select s.id from Swiss s where s.id LIKE '%100K_RAT%' fetch first 3 rows only	ID---------- 100K_RAT 1 Satz/Sätze ausgewählt.

Beide Abfragen in Tabelle 3 liefern die gleichen Ergebnisse. Die erste Abfrage wird jedoch viel schneller verarbeitet als die zweite. Die erste Abfrage verwendet die Funktion BIORS.CONTAINS zur Angabe des Eingabevergleichselements. Demzufolge wählt BioRS die Daten in der Datenbank 'swissprot' aus und übergibt die ausgewählten Daten an DB2 Information Integrator. In der zweiten Abfrage wird das Eingabevergleichselement LIKE direkt im Kurznamen 'Swiss' angegeben. Demzufolge leitet BioRS die gesamte Datenbank 'swissprot' an DB2 Information Integrator weiter. Nachdem der Inhalt der Datenbank weitergeleitet wurde, wählt DB2 Information Integrator die Daten aus.

Die Abfragen und Ergebnisse in Tabelle 4 zeigen die Verwendung von Platzhalterzeichen in der Funktion BIORS.CONTAINS. Alle Abfrageergebnisse in Tabelle 4 sind identisch, obwohl unterschiedliche Platzhalterzeichen verwendet werden.

Tabelle 4. Beispielabfragen mit Platzhalterzeichen in der Funktion BIORS.CONTAINS

Abfrage	Ergebnis
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 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.

Die Abfragen und Ergebnisse in Tabelle 5 zeigen, wie mit Hilfe der Funktion BIORS.CONTAINS auf Daten in BioRS-Elementen vom Datentyp 'Author' zugegriffen werden kann.

Die Syntax aller Abfragen in Tabelle 5 ist fast identisch. Der einzige Unterschied besteht darin, dass die Vornamensinitiale im Abfragebegriff entweder vorhanden ist oder nicht, und dass die Anzahl der Leerzeichen zwischen dem Nachnamen und der Vornamensinitiale variiert.

Tabelle 5. Beispielabfragen zum Zugriff auf BioRS-Spalten vom Datentyp 'Author'

Abfrage	Ergebnis
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 Satz/Sätze ausgewählt.

Tabelle 5. Beispielabfragen zum Zugriff auf BioRS-Spalten vom Datentyp 'Author' (Forts.)

Abfrage	Ergebnis
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller,D') = 1 fetch first 3 rows only	AUTHORS ---------- 0 Satz/Sätze ausgewählt.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller ,D') = 1 fetch first 3 rows only	AUTHORS ---------- 0 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.

Die Abfragen und Ergebnisse in Tabelle 6 zeigen, wie mit Hilfe der Funktion BIORS.CONTAINS auf Daten in BioRS-Elementen vom Datentyp 'Date' zugegriffen werden kann.

Wenn ein BioRS-Feld vom Typ 'Date' eine Folge von Datumsangaben enthält, können die Ergebnisse zusätzliche Informationen enthalten, wie aus dem zweiten Beispiel in Tabelle 6 ersichtlich ist. BioRS-Elemente vom Datentyp 'Numeric' ('Date' und 'Number') können mehrere Werte enthalten. Daher können die Ergebnisse von Abfragen, die für BioRS-Elemente vom Typ 'Date' oder 'Number' ausgeführt werden, ebenfalls mehrere Werte enthalten. Mehrere Werte werden stets durch Leerzeichen getrennt.

Tabelle 6. Beispielabfragen zum Zugriff auf BioRS-Spalten vom Datentyp 'Date'

Abfrage	Ergebnis
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 Satz/Sätze ausgewählt.

Tabelle 6. Beispielabfragen zum Zugriff auf BioRS-Spalten vom Datentyp 'Date' (Forts.)

Abfrage	Ergebnis
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 Satz/Sätze ausgewählt.

Die Abfragen und Ergebnisse in Tabelle 7 zeigen die Verwendung der Funktionen BIORS.CONTAINS_LE und BIORS.CONTAINS_GE.

Tabelle 7. Beispielabfragen mit den Funktionen BIORS.CONTAINS_LE und BIORS.CONTAINS_GE

Abfrage	Ergebnis
select s.molweight from Swiss s where biors.CONTAINS_LE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT ---------- 100368 10576 8523 3 Satz/Sätze ausgewählt.
select s.molweight from Swiss s where biors.CONTAINS_GE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT ---------- 100368 103625 132801 3 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.

Die Abfragen und Ergebnisse in Tabelle 8 auf Seite 27 zeigen, wie mit Hilfe der Funktion BIORS.SEARCH_TERM ein Suchbegriff im BioRS-Format eingegeben werden kann.

Tabelle 8. Beispielabfragen mit der Funktion BIORS.SEARCH_TERM

Abfrage	Ergebnis
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 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.
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 Satz/Sätze ausgewählt.

Die folgende Abfrage zeigt, wie relationale Vergleichselemente verwendet werden, um eine Verknüpfung mit Gleichheitsattribut zwischen zwei Datenbanken zu erstellen, die in einer Elter-Kind-Abhängigkeit zueinander stehen:

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

Die Abfrageergebnisse sehen wie folgt aus:

```
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 Satz/Sätze ausgewählt.
```

In den vorstehenden Abfrageergebnissen ist der Eintrag 100K_RAT das übergeordnete Element (Elter) für neun untergeordnete Elemente, den Kindern (100K_RAT.1 bis 100K_RAT.9).

Zugehörige Konzepte:

- „Richtlinien zur Leistungsoptimierung des BioRS-Wrappers" auf Seite 15
- „Vergleichselemente für Verknüpfungen mit Gleichheitsattributen für den BioRS-Wrapper" auf Seite 20

Zugehörige Referenzen:

- „Angepasste Funktionen und BioRS-Abfragen" auf Seite 16
- „Anweisung CREATE NICKNAME - Beispiele für den BioRS-Wrapper" auf Seite 11
- „Anweisungssyntax für CREATE NICKNAME - BioRS-Wrapper" auf Seite 39

Statistische BioRS-Daten

In einem System zusammengeschlossener Datenbanken verwendet die zusammengeschlossene Datenbank Katalogstatistiken für Objekte mit Kurznamen, um die Abfrageverarbeitung zu optimieren. Diese Statistiken werden aus BioRS-Datenquellen abgerufen, wenn Sie unter Verwendung der Anweisung CREATE NICKNAME einen Kurznamen erstellen. Die zusammengeschlossene Datenbank stellt das Vorhandensein des Objekts in der Datenquelle sicher und versucht dann, statistische Daten zu bestehenden Datenquellen zu sammeln. Informationen werden aus den Datenquellenkatalogen gelesen und in den DB2®-Katalog des Systems zusammengeschlossener Datenbanken auf dem Server mit zusammengeschlossenen Datenbanken aufgenommen.

Für BioRS-Datenquellen sind folgende statistische Daten von wesentlicher Bedeutung:

- Die Kardinalität eines Kurznamens. Bei BioRS-Datenquellen entspricht die Kardinalität eines Kurznamens der Anzahl von Einträgen in der zugehörigen BioRS-Datenbank.
- Die Kardinalität der Spalte, die dem Element BioRS _ID_ entspricht. Die Kardinalität dieser Spalte muss der Kardinalität des Kurznamens entsprechen, in dem auf die Spalte verwiesen wird.

- Die Kardinalität aller Spalten, die der BioRS-Wrapper möglicherweise verwenden muss.

Um die Leistung des BioRS-Wrappers zu optimieren, müssen aktuelle Statistiken über die BioRS-Datenquellen verwaltet werden. Wenn sich die statistischen Daten oder Strukturmerkmale eines fernen Objekts, für das ein Kurzname definiert wurde, ändern, müssen Sie die entsprechenden Kardinalitätsstatistiken in Ihrem System zusammengeschlossener Datenbanken aktualisieren. Die Kardinalitätsstatistiken werden in der Katalogsicht SYSSTAT.TABLES und in der Katalogsicht SYSSTAT.COLUMNS gespeichert.

Um die BioRS-Kardinalitätsstatistiken in Ihrem System zusammengeschlossener Datenbanken zu verwalten, führen Sie die folgenden Tasks durch:

1. Ermitteln Sie ggf. die Kardinalitätsstatistiken des gewünschten Kurznamens.
2. Aktualisieren Sie die entsprechenden Kardinalitätsstatistiken in den erforderlichen Katalogsichten.

Zugehörige Konzepte:

- „Optimieren der Abfrageverarbeitung" in *Systeme zusammengeschlossener Datenbanken*

Zugehörige Tasks:

- „Ermitteln der Kardinalitätsstatistiken von BioRS-Datenbanken" auf Seite 29
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen" auf Seite 30
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten" auf Seite 13
- „Aktualisieren der Kardinalität der BioRS-Spalte _ID_" auf Seite 31

Ermitteln der Kardinalitätsstatistiken von BioRS-Datenbanken

Bevor Sie Kurznamenstatistiken oder die Kardinalität der dem BioRS-Element _ID_ entsprechenden Spalte aktualisieren können, müssen Sie zunächst die Kardinalitätsstatistiken der entsprechenden BioRS-Datenbanken ermitteln.

Vorgehensweise:

Um die Kardinalität einer bestimmten Datenbank in BioRS zu ermitteln, verwenden Sie das BioRS-Dienstprogramm 'admin_find' oder 'www_find.cgi'. Geben Sie die Option `-c` (Kardinalität) an. Die BioRS-Dokumentation enthält weitere Informationen zu diesen beiden BioRS-Dienstprogrammen.

Zugehörige Konzepte:

- „Statistische BioRS-Daten" auf Seite 28

Zugehörige Tasks:

- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen" auf Seite 30
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten" auf Seite 13
- „Aktualisieren der Kardinalität der BioRS-Spalte _ID_" auf Seite 31

Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen

Sie müssen die Kardinalitätsstatistiken von BioRS-Kurznamen aktualisieren, wenn sich der Inhalt einer BioRS-Datenbank, für die Sie einen Kurznamen erstellet haben, maßgeblich ändert. Durch ordnungsgemäße Verwaltung der Kardinalitätsstatistiken von Kurznamen können das Optimierungsprogramm und der BioRS-Wrapper den Datenzugriffsplan auswählen, der die beste Leistung erbringt.

Um die Kardinalitätsstatistiken von BioRS-Kurznamen zu aktualisieren, modifizieren Sie die Katalogsicht SYSSTAT.TABLES anhand der korrekten Kardinalitätszahl.

Voraussetzungen:

Sie müssen die Kardinalitätszahl der BioRS-Datenbank ermitteln, die dem Kurznamen entspricht, dessen Statistiken Sie aktualisieren wollen.

Vorgehensweise:

Setzen Sie die Anweisung UPDATE mit folgender Syntax ab:

```
UPDATE sysstat.tables SET card=<kardinalität>
WHERE tabschema=<kurznamenschema>
AND tabname=<name_des_kurznamens>
```

- Hierbei ist <*kardinalität*> die Kardinalitätszahl der BioRS-Datenbank, die dem Kurznamen entspricht, dessen Statistiken Sie aktualisieren wollen.
- <*kurznamenschema*> ist der Name des Schemas, das dem Kurznamen zugeordnet ist, dessen Statistiken Sie aktualisieren wollen.
- <*name_des_kurznamens*> ist der Name des Kurznamens, dessen Statistiken Sie aktualisieren wollen.

Zugehörige Konzepte:

- „Statistische BioRS-Daten" auf Seite 28

Zugehörige Tasks:

- „Ermitteln der Kardinalitätsstatistiken von BioRS-Datenbanken" auf Seite 29
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten" auf Seite 13
- „Aktualisieren der Kardinalität der BioRS-Spalte _ID_" auf Seite 31

Aktualisieren der Kardinalität der BioRS-Spalte _ID_

Durch ordnungsgemäße Verwaltung der Kardinalitätsstatistiken der Spalte, die dem BioRS-Element _ID_ zugeordnet ist, können das Optimierungsprogramm und der BioRS-Wrapper den Datenzugriffsplan auswählen, der die beste Leistung erbringt.

Um die Kardinalitätszahl der Spalte zu aktualisieren, die dem BioRS-Element _ID_ zugeordnet ist, müssen Sie die Katalogsicht SYSSTAT.COLUMNS modifizieren.

Voraussetzungen:

Sie müssen die Kardinalitätszahl der BioRS-Datenbank ermitteln, die dem Kurznamen entspricht, in dem auf die Spalte verwiesen wird. Die Kardinalitätszahl der Spalte, die dem BioRS-Element _ID_ zugeordnet ist, muss der Kardinalität des Kurznamens entsprechen, in dem auf die Spalte verwiesen wird.

Vorgehensweise:

Um die Kardinalitätsstatistiken der BioRS-Spalte _ID_ zu aktualisieren, setzen Sie die Anweisung UPDATE mit folgender Syntax ab:

```
UPDATE sysstat.columns SET colcard=<<kardinalitä>t)
 WHERE
  tabschema=<kurznamenschema>
  AND tabname=<name_des_kurznamens>
  AND colname IN (SELECT colname FROM syscat.coloptions
 WHERE
                   tabschema=<name_des_kurznamens>
                   AND tabname=<name_des_kurznamens>
                   AND option='ELEMENT_NAME';
                   AND setting='_ID_')
```

- Hierbei ist <*kardinalität*> die Kardinalitätszahl der BioRS-Datenbank, die dem Kurznamen der Spalte entspricht.
- <*kurznamenschema*> ist der Name des Schemas, das dem Kurznamen der Spalte zugeordnet ist.
- <*name_des_kurznamens*> ist der Name des Kurznamens, in dem die Spalte verwendet wird.

Zugehörige Konzepte:

- „Statistische BioRS-Daten” auf Seite 28

Zugehörige Tasks:

- „Ermitteln der Kardinalitätsstatistiken von BioRS-Datenbanken” auf Seite 29

- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Kurznamen" auf Seite 30
- „Aktualisieren der Kardinalitätsstatistiken von BioRS-Spalten" auf Seite 13

Das BioRS-Element 'AllText'

Jede Datenbank im BioRS-System enthält ein Element namens 'AllText'. Dieses indexierte Element wird von BioRS automatisch für alle Datenbanken erstellt.

Mit dem Element 'AllText' können Sie den gesamten Text eines Eintrags durchsuchen, nicht nur bestimmte indexierte Elemente. Die Suche nach dem Begriff `muscus` beispielsweise kann Einträge zurückgeben, in denen das Wort `muscus` im Titel ('title'), in der Kurzdarstellung ('abstract', in der Beschreibung ('description') oder im Organismus ('organism') vorkommt.

Um das Element 'AllText' in einer DB2 Information Integrator-Abfrage verwenden zu können, müssen Sie das Element 'AllText' einer Kurznamenspalte zuordnen. Nachdem dieses Element einer Kurznamenspalte ordnungsgemäß zugeordnet wurde, können Sie die betreffende Spalte beim Aufruf einer angepassten CONTAINS-Funktion verwenden.

Wird auf eine Spalte, die dem Element 'AllText' zugeordnet ist, in der Liste SELECT einer Abfrage verwiesen, wird stets der Wert NULL zurückgegeben.

Zugehörige Tasks:

- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9

Zugehörige Referenzen:

- „BioRS-Wrapper - Beispielabfragen" auf Seite 22

Überlegungen zum Ändern von Kurznamen - BioRS-Wrapper

Sie haben die Möglichkeit, zuvor registrierte BioRS-Kurznamen mit Hilfe der Anweisung ALTER NICKNAME zu ändern. Mit der Anweisung ALTER NICKNAME können Sie folgende Aktionen ausführen:

- Ändern des Namens einer Spalte
- Ändern des Datentyps einer Spalte
- Hinzufügen, Ändern oder Löschen von Optionen für eine Spalte

Einschränkungen:

Der Name der Datenbank, auf die ein Kurzname verweist oder die in einem Kurznamen verwendet wird, kann nicht geändert werden. Ändert sich der

Name einer BioRS-Datenbank, die in einem Kurznamen verwendet wird, müssen Sie den gesamten Kurznamen zunächst löschen und dann erneut erstellen.

Wird die Option REMOTE_OBJECT angegeben, kann der Optionswert nicht geändert oder gelöscht werden.

Wird der Datentyp einer Spalte geändert, muss der neue Datentyp mit dem Datentyp des entsprechenden BioRS-Elements kompatibel sein.

Wird der Elementname einer Spalte mit Hilfe der Option ELEMENT_NAME geändert, wird der neue Name nicht auf Richtigkeit überprüft. Eine inkorrekte Option kann zu Fehlern führen, wenn in einer Abfrage auf die Spalte verwiesen wird.

Wenn an der Spaltenoption IS_INDEXED Änderungen vorgenommen werden, werden diese Änderungen nicht mit dem BioRS-Server überprüft. Eine inkorrekte Option kann zu Fehlern führen, wenn in einer Abfrage auf die Spalte verwiesen wird.

Zugehörige Referenzen:

- „ALTER NICKNAME statement" in *SQL Reference, Volume 2*

Tabelle der angepassten Funktionen - BioRS-Wrapper

Tabelle 9 enthält Beispiele dazu, wie Sie die vier angepassten BioRS-Funktionen jeweils registrieren können.

Die Beispieldatei 'create_function_mappings.ddl' im Verzeichnis 'sqllib/samples/lifesci/biors' soll Sie beim Registrieren der angepassten Funktionen unterstützen. Die Datei 'create_function_mappings.ddl' enthält Definitionen für die einzelnen angepassten Funktionen. Sie können diese DDL-Datei ausführen, um die angepassten Funktionen für jede DB2-Datenbank zu registrieren, in der der BioRS-Wrapper installiert ist.

Tabelle 9. Angepasste Funktionen für den BioRS-Wrapper

Funktionsname	Beschreibung
CONTAINS (col VARCHAR(), term VARCHAR()), CONTAINS (col VARCHAR(), term CHAR()) CONTAINS (col VARCHAR(), term DATE) CONTAINS (col VARCHAR(), term TIMESTAMP)	Durchsucht eine indexierte Spalte unter Verwendung des angegebenen Ausdrucks. **col** Indexierte Spalte. **term** Suchbegriff.

Tabelle 9. Angepasste Funktionen für den BioRS-Wrapper (Forts.)

Funktionsname	Beschreibung
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)	Durchsucht eine indexierte Spalte unter Verwendung des angegebenen Ausdrucks. **col** Indexierte Spalte. **term** Suchbegriff.
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)	Durchsucht eine indexierte Spalte unter Verwendung des angegebenen Ausdrucks. **col** Indexierte Spalte. **term** Suchbegriff.
SEARCH_TERM (col VARCHAR(), term VARCHAR()) SEARCH_TERM (col VARCHAR(), term CHAR()) SEARCH_TERM (col CHAR(), term VARCHAR()) SEARCH_TERM (col CHAR(), term CHAR())	Übergibt einen BioRS-Suchbegriff an die BioRS-Suchmaschine. **col** Indexierte Spalte. **term** Suchbegriff.

Zugehörige Tasks:

- „Registrieren von angepassten Funktionen für den BioRS-Wrapper" auf Seite 4

Nachrichten für den BioRS-Wrapper

In diesem Abschnitt werden die Nachrichten erläutert, die bei der Verwendung des BioRS-Wrappers angezeigt werden können. Weitere Informationen zu Nachrichten finden Sie im Handbuch zu *DB2 Fehlernachrichten*.

Tabelle 10. Vom BioRS-Wrapper ausgegebene Nachrichten

Fehlercode	Nachricht	Erläuterung
SQL0604N	Das Attribut 'Length', 'Precision' oder 'Scale' für die Spalte, den einzigartigen Datentyp, den strukturierten Typ, das Attribut eines strukturierten Typs, die Funktion oder Typenzuordnung <datenelement> ist ungültig.	Der Datentyp für eine Kurznamenspalte ist mit dem BioRS-Typ des zugrunde liegenden Datenbankelements nicht kompatibel. Überprüfen Sie den Datentyp der Spalte in der Anweisung CREATE NICKNAME.

Tabelle 10. Vom BioRS-Wrapper ausgegebene Nachrichten (Forts.)

Fehlercode	Nachricht	Erläuterung
SQL0901N	Die SQL-Anweisung schlug auf Grund eines nicht schwer wiegenden (nicht kritischen) Systemfehlers fehl. Nachfolgende SQL-Anweisungen können verarbeitet werden. (Ursache: ″Fehler beim Erstellen eines Wrapperobjekts.″)	Beim Erstellen eines neuen Wrapperobjekts ist ein Fehler aufgetreten. Wenden Sie sich an die IBM Unterstützungsfunktion.
SQL0901N	Die SQL-Anweisung schlug auf Grund eines nicht schwer wiegenden (nicht kritischen) Systemfehlers fehl. Nachfolgende SQL-Anweisungen können verarbeitet werden. (Ursache: ″BioRS <tracepunkt><code>.″)	Dies ist ein interner Fehler. Wenden Sie sich an die IBM Unterstützungsfunktion.
SQL0901N	Die SQL-Anweisung schlug auf Grund eines nicht schwer wiegenden (nicht kritischen) Systemfehlers fehl. Nachfolgende SQL-Anweisungen können verarbeitet werden. (Urache: ″Zuordnung von Speicherkapazität ist fehlgeschlagen: <tracepunkt>.″)	Bei der Speicherzuordnung trat ein Fehler auf. Stellen Sie sicher, dass für den Host des Servers mit zusammengeschlossenen Datenbanken ausreichend Speicherkapazität zur Verfügung steht und übergeben Sie die Abfrage erneut. Falls der Fehler weiterhin auftritt, wenden Sie sich an die IBM Unterstützungsfunktion.
SQL0901N	Die SQL-Anweisung schlug auf Grund eines nicht schwer wiegenden (nicht kritischen) Systemfehlers fehl. Nachfolgende SQL-Anweisungen können verarbeitet werden. (Ursache: ″sqlno_crule_save_plans[100]:rc(-214272209) Leere Planliste.″)	Das Optimierungsprogramm und der BioRS-Wrapper konnten keinen gemeinsamen Plan für die Ausführung der Abfrage finden. Vereinfachen Sie die Abfrage und führen Sie sie erneut aus.
SQL401N	Die Datentypen der Operanden für die Operation ″=″ sind nicht kompatibel.	Die Abfrage ist ungültig, da der Ausdruck auf der rechten Seite im Vergleichselement einer angepassten Funktion eine ganze Zahl sein muss.
SQL1822N	Unerwarteter Fehlercode ″″ von Datenquelle ″BioRS-Wrapper″ empfangen. Zugeordneter Text und Token sind ″Die Datenbank wurde nicht gefunden″.	Die BioRS-Datenbank, auf die in der Anweisung CREATE NICKNAME verwiesen wurde, konnte auf dem BioRS-Server nicht gefunden werden. Überprüfen Sie die Anweisung CREATE NICKNAME und stellen Sie sicher, dass der Name der Datenbank korrekt ist.

Tabelle 10. Vom BioRS-Wrapper ausgegebene Nachrichten (Forts.)

Fehlercode	Nachricht	Erläuterung
SQL1822N	Unerwarteter Fehlercode "" von Datenquelle "BioRS-Wrapper" empfangen. Zugeordneter Text und Token sind "Die Zeitsperre für die Verbindung wurde erreicht".	Der BioRS-Server hat auf eine Kommunikationsanforderung nicht innerhalb dem von der Option TIMEOUT angegebenen Zeitraum geantwortet.
SQL1822N	Unerwarteter Fehlercode "<tracepunkt>" von der Datenquelle "BioRS Wrapper" empfangen. Zugeordneter Text und Token sind "Fehler beim Lesen vom Server."	Während des Lesens von Daten vom BioRS-Server trat ein Kommunikationsfehler auf. Der Wert des Fehlercodes <tracepunkt> enthält möglicherweise weitere Informationen zu diesem Fehler.
SQL1822N	Unerwarteter Fehlercode "<tracepunkt>" von der Datenquelle "BioRS Wrapper" empfangen. Zugeordneter Text und Token sind "Der Host wurde nicht gefunden".	Der in der Serveroption HOST angegebene Host des BioRS-Servers wurde nicht gefunden. Überprüfen Sie die Anweisung CREATE SERVER und stellen Sie sicher, dass der Wert der Serveroption HOST korrekt ist.
SQL1822N	Unerwarteter Fehlercode "<tracepunkt>" von der Datenquelle "BioRS Wrapper" empfangen. Zugeordneter Text und Token sind "Verbindung zum Server kann nicht hergestellt werden".	Der Wrapper konnte keine Verbindung zu dem Server herstellen, der von der Serveroption HOST angegeben wurde. Der Wert des Fehlercodes <tracepunkt> enthält möglicherweise weitere Informationen zu diesem Fehler.
SQL1822N	Unerwarteter Fehlercode "<tracepunkt>" von der Datenquelle "BioRS Wrapper" empfangen. Zugeordneter Text und Token sind "TCPIP-Socket konnte nicht erstellt werden".	Der Wrapper konnte keinen TCPIP-Socket erstellen. Der Wert des Fehlercodes <tracepunkt> enthält möglicherweise weitere Informationen zu diesem Fehler.
SQL1822N	Unerwarteter Fehlercode "<tracepunkt>" von der Datenquelle "BioRS Wrapper" empfangen. Zugeordneter Text und Token sind "Fehler beim Senden zum Server."	Der Wrapper konnte eine Anforderung nicht an den BioRS-Server senden. Der Wert des Fehlercodes <tracepunkt> enthält möglicherweise weitere Informationen zu diesem Fehler.

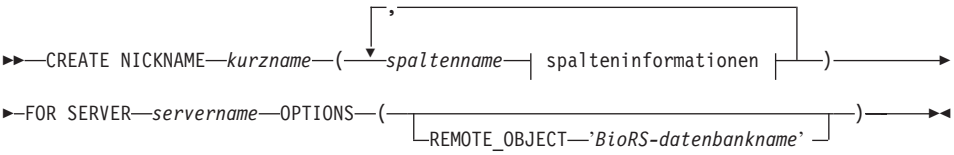
Tabelle 10. Vom BioRS-Wrapper ausgegebene Nachrichten (Forts.)

Fehlercode	Nachricht	Erläuterung
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Groß-/Kleinschreibung des Servers kann nicht geändert werden."	Der Wert der Serveroption CASE_SENSITIVE kann nicht mit Hilfe von SQL-Anweisungen geändert werden. Um den Wert dieser Option zu ändern, müssen Sie den Server zunächst löschen. Anschließend müssen Sie den Server mit Hilfe der Anweisung CREATE SERVER erneut erstellen und dann den korrekten Wert für die Option CASE_SENSITIVE angeben.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Mehrere Verknüpfungen zwischen zwei Kurznamen."	Die Abfrage ist ungültig, da zwischen zwei Kurznamen jeweils nur ein Verknüpfungsprädikat zulässig ist.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Die rechte Seite eines Funktionsprädikats muss eine Konstante sein."	Die Abfrage ist ungültig, da der Ausdruck auf der rechten Seite des Prädikats einer angepassten Funktion eine Konstante sein muss.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Argument 1 der angepassten Funktion ist keine Spalte."	Die Abfrage ist ungültig, da das erste Argument einer angepassten Funktion auf eine Spalte eines BioRS-Kurznamens verweisen muss.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Argument 1 der Funktion CONTAINS ist nicht indexiert."	Die Abfrage ist ungültig. Die Spalte, auf die im ersten Argument der Funktion BIORS.CONTAINS, BIORS.CONTAINS_LE oder BIORS.CONTAINS_GE verwiesen wird, muss eine indexierte Spalte sein.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = "Ungültiger Typ für Argument 1 der Funktion <funktionsname>."	Die Abfrage ist ungültig. Der Datentyp der Spalte, auf die im ersten Argument der Funktion BIORS.CONTAINS, BIORS.CONTAINS_LE oder BIORS.CONTAINS_GE verwiesen wird, ist nicht korrekt.

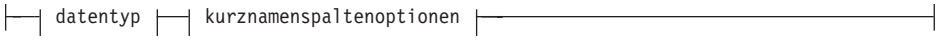
Tabelle 10. Vom BioRS-Wrapper ausgegebene Nachrichten (Forts.)

Fehlercode	Nachricht	Erläuterung
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Argument 1 der Funktion SEARCH_TERM ist nicht die Spalte _ID_″.	Die Abfrage ist ungültig. Der Spalte, auf die im ersten Argument der Funktion SEARCH_TERM verwiesen wird, ist nicht dem BioRS-Element _ID_ zugeordnet.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Der Bindeparameter kann nicht NULL sein.″	Der Wert einer Spalte oder Hostvariablen, auf den im zweiten Argument der Funktion BIORS.CONTAINS verwiesen wurde, war NULL. Der BioRS-Wrapper kann keine Nullwerte verarbeiten.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Wert kann nicht in BioRS-Literal umgewandelt werden.″	In einem Literal, einer Spalte oder einer Hostvariable wurde ein Wert an den Wrapper übergeben, der nicht in ein gültiges BioRS-Literal umgewandelt werden konnte.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Die Serverversion kann nicht geändert werden.″	Die Serverversion kann nicht mit Hilfe der Anweisung ALTER SERVER geändert werden. Um die Serverversion zu ändern, müssen Sie den Server zunächst löschen. Anschließend müssen Sie den Server mit Hilfe der Anweisung CREATE SERVER erneut erstellen und dabei die korrekte Version angeben.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Ungültiger Typ für Argument 2 der Funktion <funktionsname>.″	Die Abfrage ist ungültig. Der Datentyp der Spalte, auf die im zweiten Argument der Funktion BIORS.CONTAINS, BIORS.CONTAINS_LE oder BIORS.CONTAINS_GE verwiesen wird, ist nicht korrekt.
SQL30090N	Die Operation ist für die Umgebung der Anwendungsausführung nicht gültig. Ursachencode = ″Der Kurzname hat keine Spalten.″	Für die Anweisung CREATE NICKNAME wurden keine Spaltendeklarationen angegeben. Für das Erstellen von Kurznamen sind Spaltendeklarationen jedoch erforderlich.

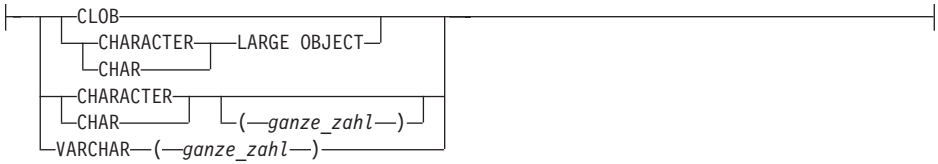
Anweisungssyntax für CREATE NICKNAME - BioRS-Wrapper



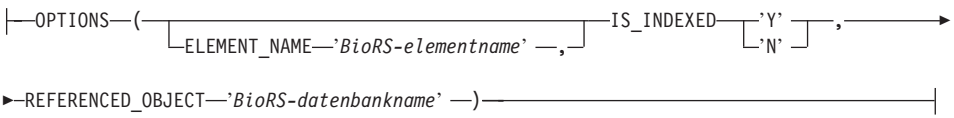
spalteninformationen:



datentyp:



kurznamenspaltenoptionen:



Kurznamenspaltenoptionen

Werte für Kurznamenspaltenoptionen müssen in einfache Anführungszeichen eingeschlossen werden.

ELEMENT_NAME
Gibt den Namen des BioRS-Elements an. Die Groß-/Kleinschreibung dieses Namens hängt von der Groß-/Kleinschreibung des BioRS-Servers und des Wertes der Serveroption CASE_SENSITIVE ab. Der Name des BioRS-Elements braucht nur dann angegeben zu werden, wenn er sich vom Namen der Spalte unterscheidet.

IS_INDEXED

Gibt an, ob die entsprechende Spalte indexiert ist, d. h. ob auf die Spalte in einem Vergleichselement verwiesen werden kann. Gültige Werte sind 'Y' (ja) und 'N' (nein). Der Wert 'Y' kann nur für Spalten angegeben werden, deren entsprechendes Element vom BioRS-Server indexiert ist.

Bei der Erstellung eines Kurznamens wird diese Option automatisch mit dem Wert 'Y' allen Spalten hinzugefügt, die einem indexierten BioRS-Element entsprechen.

REFERENCED_OBJECT

Diese Option ist nur für Spalten gültig, deren BioRS-Datentyp 'Reference' (Verweis) lautet. Diese Option gibt den Namen der BioRS-Datenbank an, auf die die aktuelle Spalte verweist. Die Groß-/Kleinschreibung dieses Namens hängt von der Groß-/Klein- schreibung des BioRS-Servers und des Wertes der Serveroption CASE_SENSITIVE ab.

Kurznamenoptionen

Optionswerte für Kurznamen müssen in einfache Anführungszeichen eingeschlossen werden.

REMOTE_OBJECT

Gibt den Namen der BioRS-Datenbank an, die dem Kurznamen zugeordnet ist. Dieser Name legt das Schema und die BioRS-Datenbank für den Kurznamen fest. Außerdem gibt dieser Name die Abhängigkeit zwischen dem Kurznamen und anderen Kurznamen an. Die Groß-/Kleinschreibung dieses Namens hängt von der Groß-/Kleinschreibung des BioRS-Servers und des Wertes der Serveroption CASE_SENSITIVE ab.

Wichtig: Dieser Name kann nicht mit der Anweisung ALTER NICKNAME geändert oder gelöscht werden. Ändert sich der Name der BioRS-Datenbank, der in dieser Option verwendet wird, müssen Sie den gesamten Kurznamen zunächst löschen und dann erneut erstellen.

Zugehörige Tasks:

- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9

Zugehörige Referenzen:

- „CREATE NICKNAME statement" in *SQL Reference, Volume 2*
- „Anweisung CREATE NICKNAME - Beispiele für den BioRS-Wrapper" auf Seite 11

Anweisungsoptionen für CREATE SERVER - BioRS-Wrapper

Folgende Optionen sind der Anweisung CREATE SERVER für BioRS zugeordnet:

TYPE Gibt den Servertyp an. Der Standardwert ist `BioRS`. Für den BioRS-Wrapper wird ausschließlich dieser Standardwert unterstützt. Diese Option braucht nicht angegeben zu werden.

VERSION
Gibt die Serverversion an. Der Standardwert ist `1.0`. Für den BioRS-Wrapper wird ausschließlich dieser Standardwert unterstützt. Diese Option braucht nicht angegeben zu werden.

NODE
Gibt den Hostnamen des Systems an, auf dem das BioRS-Abfragetool zur Verfügung steht. Der Standardwert ist *localhost* (lokaler Host). Diese Option muss angegeben werden.

PORT Gibt die Nummer des Ports an, der verwendet werden soll, um eine Verbindung zum BioRS-Server herzustellen. Der Standardwert ist 5014. Diese Option muss angegeben werden.

TIMEOUT
Gibt die Zeit (in Minuten) an, die der BioRS-Wrapper auf Antworten vom BioRS-Server warten soll. Der Standardwert ist 10. Diese Option muss angegeben werden.

CASE_SENSITIVE
Gibt an, ob der BioRS-Server bei Namen die Groß-/Kleinschreibung berücksichtigen soll. Gültige Werte sind 'Y' (ja) und 'N' (nein). Der Standardwert ist 'Y'.

Bei BioRS wird die Groß-/Kleinschreibung der auf der BioRS-Servermaschine gespeicherten Daten durch einen Konfigurationsparameter gesteuert. Die Option CASE_SENSITIVE ist das von DB2 Information Integrator verwendete Gegenstück zum BioRS-Systemkonfigurationsparameter. Die Konfigurationseinstellungen für die Groß-/Kleinschreibung auf dem BioRS-Server müssen im BioRS-System und in DB2 Information Integrator synchronisiert werden. Werden die Konfigurationseinstellungen für die Groß-/Kleinschreibung zwischen BioRS und DB2 Information Integrator nicht synchronisiert, führt dies zu Fehlern, wenn versucht wird, über DB2 Information Integrator auf BioRS-Daten zuzugreifen.

Wichtig: Nach Erstellung eines neuen BioRS-Servers unter DB2 Information Integrator kann die Option CASE_SENSITIVE nicht mehr geändert oder gelöscht werden. Ist eine Änderung der Option CASE_SENSITIVE erforderlich, müssen Sie den gesamten Server zunächst löschen und anschließend erneut erstellen. Wenn Sie den BioRS-Server löschen, müssen Sie auch alle zugehörigen BioRS-Kurznamen erneut erstellen. DB2 Information Integrator löscht automatisch alle Kurznamen, die einem gelöschten Server zugeordnet sind.

Zugehörige Tasks:

- „Registrieren des Servers für BioRS-Datenquellen" auf Seite 7
- „Registrieren von Kurznamen für BioRS-Datenquellen" auf Seite 9

Zugehörige Referenzen:

- „CREATE SERVER statement" in *SQL Reference, Volume 2*
- „Anweisungssyntax für CREATE NICKNAME - BioRS-Wrapper" auf Seite 39

Anweisungsoptionen für CREATE USER MAPPING - BioRS-Wrapper

GUEST

Gibt an, ob Operationen unter dem BioRS-Authentifizierungsmechanismus für Gäste (GUEST) auf dem BioRS-Server ausgeführt werden sollen. Gültige Werte sind 'Y' (ja) und 'N' (nein). Der Standardwert ist 'Y'.

Wird diese Option auf 'Y' gesetzt, wird für den Zugriff auf den BioRS-Server die Gastauthentifizierung für den betreffenden Benutzer von DB2 Information Integrator verwendet.

Wird diese Option auf 'N' gesetzt, muss der betreffende Benutzer von DB2 Information Integrator eine BioRS-Berechtigungs-ID und ein BioRS-Kennwort angeben, um auf den BioRS-Server zugreifen zu können.

Wird keine Benutzerzuordnung erstellt oder wird eine Benutzerzuordnung ohne Angabe von Optionen erstellt, wird für den Zugriff auf den BioRS-Server die Gastauthentifizierung für den betreffenden Benutzer von DB2 Information Integrator verwendet.

REMOTE_AUTHID

Gibt eine Benutzer-ID an, die den betreffenden DB2-Benutzer für den Zugriff auf BioRS-Datenquellen berechtigt. Diese ferne ID muss über das Format verfügen, das von der BioRS-Anwendung erwartet wird. Diese Option ist erforderlich, wenn die Option GUEST auf 'N' gesetzt ist.

REMOTE_PASSWORD
Gibt das Kennwort für diese ferne ID an. Diese Option ist erforderlich, wenn die Option GUEST auf 'N' gesetzt ist.

Beispiel:

Mit der folgenden Anweisung CREATE USER MAPPING wird der Benutzer 'Charlie' dem Benutzer 'Charlene' auf dem Server 'Biors_Server1' zugeordnet.

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

Zugehörige Tasks:

- „Registrieren von Benutzerzuordnungen für BioRS-Datenquellen" auf Seite 7

Zugehörige Referenzen:

- „CREATE USER MAPPING statement" in *SQL Reference, Volume 2*

Kapitel 2. Benutzerdefinierte Life Sciences-Funktionen

In diesem Kapitel wird erläutert, was die benutzerdefinierten Life Sciences-Funktionen sind, wie sie einem System zusammengeschlossener Datenbanken hinzugefügt und in Abfragen verwendet werden.

Benutzerdefinierte Life Sciences-Funktionen - Übersicht

Die benutzerdefinierten Life Sciences-Funktionen stellen Forschern häufig verwendete Algorithmen zur Datenanalyse zur Verfügung. Die Funktionen sind auf Windows® NT-, AIX- und Linux-32-Bit-Plattformen verfügbar, mit Ausnahme der Funktion 'GeneWise', die auf AIX®- und Linux-Plattformen zur Verfügung steht.

Die benutzerdefinierten Life Sciences-Funktionen verwenden zur Darstellung von Aminosäuren und Nukleotiden die standardmäßigen Einzelbuchstabencodes sowie die IUPAC-IUB-Ambiguitätscodes.

Vor Verwendung der benutzerdefinierten Life Sciences-Funktionen müssen diese registriert werden. Nach der Registrierung können die Funktionen bei Bedarf wieder entfernt werden.

Zugehörige Tasks:

- „Registrieren von benutzerdefinierten Life Sciences-Funktionen" auf Seite 46
- „Entfernen von benutzerdefinierten Life Sciences-Funktionen" auf Seite 48

Zugehörige Referenzen:

- „Benutzerdefinierte Life Sciences-Funktionen nach Funktionskategorie" auf Seite 45

Benutzerdefinierte Life Sciences-Funktionen nach Funktionskategorie

Tabelle 11 enthält eine Liste der benutzerdefinierten Life Sciences-Funktionen nach ihrer jeweiligen Funktionskategorie mit kurzer Beschreibung.

Tabelle 11. Benutzerdefinierte Life Sciences-Funktionen

Funktionskategorie	Benutzerdefinierte Funktionen	Beschreibung
Rückübersetzung	LSPep2AmbNuc, LSPep2ProbNuc	Wandelt eine Aminosäuresequenz in eine Nukleotidsequenz um.

Tabelle 11. Benutzerdefinierte Life Sciences-Funktionen (Forts.)

Funktionskategorie	Benutzerdefinierte Funktionen	Beschreibung
Syntaktische Analyse von Definitionszeilen	LSDeflineParse	Führt eine syntaktische Analyse von Elementen einer Definitionszeile durch, wie sie beispielsweise vom BLAST-Wrapper zurückgegeben wird oder in Datendateien im FASTA-Format vorliegt.
Allgemeine Mustererkennung	LSPatternMatch, LSPrositePattern	Ermittelt relevante Bereiche in einer angegebenen Zeichenfolge, wie beispielsweise Nukleotid- oder Peptidsequenzen.
GeneWise	LSGeneWise	Führt ein Alignment einer Proteinsequenz mit einer Genomsequenz durch.
Motive	LSMultiMatch, LSMultiMatch3, LSBarCode	Sucht übereinstimmende Muster in Nukleotid- oder Aminosäuresequenzen.
Umkehrung	LSRevNuc, LSRevPep, LSRevComp	Kehrt Nukleotid- oder Aminosäuresequenzen um.
Übersetzung	LSNuc2Pep, LSTransAllFrames	Wandelt eine Nukleotidsequenz in eine Peptidsequenz um.

Zugehörige Konzepte:

- „Benutzerdefinierte Life Sciences-Funktionen - Übersicht" auf Seite 45

Zugehörige Tasks:

- „Registrieren von benutzerdefinierten Life Sciences-Funktionen" auf Seite 46
- „Entfernen von benutzerdefinierten Life Sciences-Funktionen" auf Seite 48

Registrieren von benutzerdefinierten Life Sciences-Funktionen

Vor Verwendung der benutzerdefinierten Life Sciences-Funktionen müssen diese registriert werden.

Vorgehensweise:

Sie können die benutzerdefinierten Life Sciences-Funktionen mit Hilfe des Befehls 'enable_LSFunctions' registrieren. Dieser Befehl steht unter Windows NT und AIX im Verzeichnis `'sqllib/bin'` zur Verfügung.

Die Syntax für den Befehl 'enable_LSFunctions' lautet wie folgt:

```
enable_LSFunctions -n dbname -u benutzer-id -p kennwort [-force]
```

dbname
Der Name der Datenbank, für die die Funktionen registriert werden sollen.

benutzer-id
Eine gültige Benutzer-ID für die Datenbank.

kennwort
Ein gültiges Kennwort für die Benutzer-ID.

force Eine Markierung, mit der die Funktionen entfernt und erneut registriert werden können. Mit Hilfe dieser Markierung können Sie die Funktionen erneut registrieren, falls sie beschädigt oder versehentlich gelöscht werden.

Der Befehl erstellt die Funktionen mit dem Schemanamen DB2LS.

Das folgende Beispiel zeigt eine Beispielausgabe für den Befehl 'enable_LS-Functions':

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

(0) Life Sciences-Funktionen wurden gefunden
    -- Erstellen von Life Sciences-Funktionen ...
    Life Sciences-Funktionen erfolgreich erstellt.

*** Das Bereinigen des Systems kann einige Sekunden dauern. Bitte warten Sie ...
```

Das folgende Beispiel zeigt die Ausgabe für den Befehl 'enable_LSFunctions', wenn Sie die Markierung 'force' verwenden und die Funktionen bereits registriert sind:

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

(21) Life Sciences-Funktionen wurden gefunden

Life Sciences-Funktionen sind bereits vorhanden ...
Erneute Installation von Life Sciences-Funktionen ...
    -- Löschen von Life Sciences-Funktionen ...
    Life Sciences-Funktionen erfolgreich gelöscht.
    -- Erstellen von Life Sciences-Funktionen ...
    Life Sciences-Funktionen erfolgreich erstellt.


*** Das Bereinigen des Systems kann einige Sekunden dauern. Bitte warten Sie ...
```

Entfernen von benutzerdefinierten Life Sciences-Funktionen

Sie können die benutzerdefinierten Life Sciences-Funktionen entfernen, wenn Sie die Funktionen in Ihrem System nicht mehr benötigen.

Vorgehensweise:

Um die benutzerdefinierten Life Sciences-Funktionen zu entfernen, verwenden Sie den Befehl 'disable_LSFunctions'. Dieser Befehl steht unter Windows NT und AIX im Verzeichnis `'sqllib/bin'` zur Verfügung.

Die Syntax für den Befehl 'disable_LSFunctions' lautet wie folgt:

```
disable_LSFunctions -n dbname -u benutzer-id -p kennwort
```

dbname
Der Name der Datenbank, aus der die Funktionen entfernt werden sollen.

benutzer-id
Eine gültige Benutzer-ID für die Datenbank.

kennwort
Ein gültiges Kennwort für die Benutzer-ID.

Das folgende Beispiel zeigt die Ausgabe für den Befehl 'disable_LSFunctions':

```
C:>disable_LSFunctions -n test -u db2admin -p db2admin
a
(21) Life Sciences-Funktionen wurden gefunden
    -- Löschen von Life Sciences-Funktionen ...
    Life Sciences-Funktionen erfolgreich gelöscht.


*** Das Bereinigen des Systems kann einige Sekunden dauern. Bitte warten Sie ...
```

Benutzerdefinierte Rückübersetzungsfunktionen

Mit den benutzerdefinierten Rückübersetzungsfunktionen können Sie eine Peptidsequenz in eine Nukleotidsequenz umwandeln. Eine Rückübersetzung ist die Umkehrung einer Übersetzung.

Da die Zuordnung von Aminosäuren zu Nukleotid-Triplet-Codons eine Zuordnung "eins zu vielen" ist, führt eine Rückübersetzung zu zwei Ergebnissen:

Größte Mehrdeutigkeit
Einfache Textumwandlung und Suchfunktion. Verwenden Sie die benutzerdefinierte Funktion 'LSPep2AmbNuc', um eine Übersetzung mit der größten Mehrdeutigkeit durchzuführen.

Größte Wahrscheinlichkeit
Erfordert zusätzliche Informationen aus einer Codon-Frequenztabelle. Verwenden Sie die benutzerdefinierte Funktion 'LSPep2ProbNuc', um eine Übersetzung mit der größten Wahrscheinlichkeit durchzuführen.

Benutzerdefinierte Funktion 'LSPep2AmbNuc'

eingabe-peptidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Peptidsequenz beschreibt. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 10890 Byte. Die Eingabedaten verwenden die standardmäßigen Aminosäuresymbole und Mehrdeutigkeitscodes (Ambiguitätscodes).

dateipfad zur externen Übersetzungstabelle
Wenn Sie eine angepasste Übersetzungstabelle verwenden, müssen Sie den entsprechenden Dateipfad angeben, damit die Tabelle gefunden wird. Der Zeichenfolgewert des Pfads darf 255 Zeichen nicht überschreiten.

Der Schemaname lautet DB2LS.

Mit Hilfe der Funktion 'LSPep2AmbNuc' erstellen Sie gemäß einer Übersetzungstabelle aus einer Peptidsequenz die Nukleotidsequenz mit der größten Mehrdeutigkeit (Ambiguität).

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 32672 Byte. Das Ergebnis stellt gemäß einer integrierten oder von Ihnen angegebenen Übersetzungstabelle die Nukleotidsequenz mit der größten Mehrdeutigkeit dar. Wird keine Übersetzungstabelle angegeben, verwendet die Funktion standardmäßig Tabelle 12.

Tabelle 12. Standardübersetzungstabelle

Aminosäuresymbol	Abkürzung	Codon
A	Ala	GCX
B	Asx	RAY
C	Cys	TGY
D	Asp	GAY
V	Glu	GAR
F	Phe	TTY
G	Gly	GGX
H	His	CAY

Tabelle 12. Standardübersetzungstabelle (Forts.)

Aminosäuresymbol	Abkürzung	Codon
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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2AmbNuc' - Fehlernachrichten" auf Seite 52
- „Benutzerdefinierte Funktion 'LSPep2ProbNuc'" auf Seite 53
- „Benutzerdefinierte Funktion 'LSPep2AmbNuc' - Beispiel" auf Seite 50

Benutzerdefinierte Funktion 'LSPep2AmbNuc' - Beispiel

Sie können die Funktion mit einer values-Anweisung aufrufen. Die einzige Eingabe ist eine Peptidsequenz, wie in folgendem Beispiel gezeigt:

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

Das vorstehende Beispiel setzt eine Peptid in eine Nukleotid um, wobei die mehrdeutigen Übersetzungen und die integrierte Übersetzungstabelle verwendet werden. Das Ergebnis der vorstehenden Anweisung ist eine Nukleotidsequenz, die aus den standardmäßigen Aminosäuresymbolen erstellt wird:

```
CAYMGX
```

Das folgende Beispiel setzt ein Peptid in eine Nukleotid um, wobei die mehrdeutigen Übersetzungen und die integrierte Tabelle verwendet werden:

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

Das Ergebnis dieser values-Anweisung ist die folgende Nukleotidsequenz (die Sequenz wurde durch Teilung an das Seitenformat angepasst):

```
WSXMGXGGXTTYGGXTTYATHACXTAYWSXCAYWSXWSXATGATHGAYGARGCXCARA
ARWSXMGXCCXCAYAARATHGAYGGXMGXGTXGTXGARCCXAARMGXGCX
```

Das nächste Beispiel zeigt die Anwendung der Funktion auf eine Gruppe von Werten, die aus einer Tabelle oder einem Kurznamen extrahiert wurden:

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

Die Daten in der Spalte 'peptide_seq' der Tabelle 'protein_table' sehen folgendermaßen aus:

Tabelle 13. Daten in der Spalte 'peptide_seq'

peptide_seq
GIKEDTEEHHLRDYFE
QKYHTVNGHNCEVRKA
.....

Das Ergebnis der SELECT-Anweisung sieht wie folgt aus:

```
GGXATHAARGARGAYACXGARGARCAYCAYYTXMGXGAYTAYTTYGAR
CARAARTAYCAYACXGTXAAYGGXCAYAAYTGYGARGTXMGXAARGCX
...
```

Das folgende Beispiel setzt ein Peptid in eine Nukleotid um, wobei die mehrdeutigen Übersetzungen und eine benutzerdefinierte Tabelle verwendet werden. Normalerweise sind die Unterschiede zwischen den Übersetzungstabellen nur geringfügig und machen meist nur ein bis zwei eigene Symbole aus. Diese Unterschiede können vorkommen, da einige Spezies über mehr oder weniger Codons verfügen. Bei der Spezies Drosophila beispielsweise fehlt das Codon AGG.

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

Unter der Annahme, dass die Eingabe-Übersetzungstabelle für Drosophila gilt, sieht das Ergebnis der values-Anweisung wie folgt aus:

```
MGRGGXAAYATGGGXGGXGGXAAYTAYGGXAAYTARAAYGGXGGXGGXAAYTGGAAYAAYGGX
```

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2AmbNuc'" auf Seite 49
- „Benutzerdefinierte Funktion 'LSNuc2Pep' - Beispiel" auf Seite 85

Benutzerdefinierte Funktion 'LSPep2AmbNuc' - Fehlernachrichten

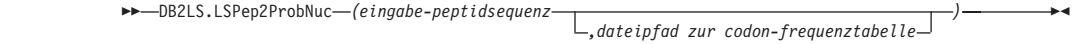
Tabelle 14. Von der benutzerdefinierten Funktion 'LSPep2AmbNuc' ausgegebene Nachrichten

Fehlercode	Nachricht	Erläuterung
SQL0443N	Die Routine "DB2LS.LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Sequenz ist ungültig". SQLSTATE=38608	Die angegebene Sequenz ist ungültig.
SQL0443N	Die Routine "DB2LS.LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUCUT") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Es wurde keine Übersetzung gefunden". SQLSTATE=38610	Die Datei mit der Übersetzungstabelle ist leer.
SQL0443N	Die Routine "LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUCUT") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Datei mit der Übersetzungstabelle kann nicht geöffnet werden". SQLSTATE=38612	Die angegebene Datei mit der Übersetzungstabelle existiert nicht.
SQL0443N	Die Routine "DB2LS.LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUCUT") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Eine Zeile war beim Lesen aus der Datei zu lang". SQLSTATE=38614	Die Datei enthielt eine Zeile, die länger war als zulässig.
SQL0443N	Die Routine "DB2LS.LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUCUT") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Ungültige Datendatei". SQLSTATE=38615	Das Dateiformat ist ungültig.
SQL0443N	Die Routine "LSPEP2AMBNUC" (spezifischer Name "LSPEP2AMBNUCUT") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Übersetzungstabelle kann nicht erstellt werden". SQLSTATE=38611	Die Datei enthielt ungültige Symbole.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2AmbNuc'" auf Seite 49

Benutzerdefinierte Funktion 'LSPep2ProbNuc'



eingabe-peptidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Peptidsequenz beschreibt. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 10890 Byte. Die Eingabedaten verwenden die standardmäßigen Aminosäuresymbole.

dateipfad zur codon-frequenztabelle
Hierbei handelt es sich um die Codon-Frequenztabelle. Geben Sie den Dateipfad an, damit die Frequenztabelle gefunden wird. Der Zeichenfolgewert des Pfads darf 255 Zeichen nicht überschreiten.

Der Schemaname lautet DB2LS.

Mit der Funktion 'LSPep2ProbNuc' generieren Sie gemäß der im zweiten Argument angegebenen Codon-Frequenztabelle aus einer Peptidsequenz die Nukleotidsequenz mit der größten Wahrscheinlichkeit.

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 32672 Byte. Die Zeichenfolge ist eine Darstellung der wahrscheinlichsten Nukleotidsequenz auf der Grundlage der verwendeten Codon-Frequenztabelle.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2AmbNuc'" auf Seite 49
- „Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Fehlernachrichten" auf Seite 55
- „Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Beispiel" auf Seite 53

Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Beispiel

Das folgende Beispiel zeigt, wie Sie eine Peptidsequenz in eine Nukleotidsequenz umsetzen können, indem Sie die in der Frequenztabelle 'yeast_high.cod' definierten Übersetzungen mit der größten Wahrscheinlichkeit verwenden.

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

Das Ergebnis aus den vorstehenden Werten sieht wie folgt aus:

```
AGAGACAATAACGACGATGATAAC
```

Eine zweite Ausführung der gleichen Anweisung gibt die nachstehende Zeichenfolge zurück:

```
AGAGATAATAACGACGATGACAAC
```

Eine dritte Ausführung der gleichen Anweisung gibt die nachstehende Zeichenfolge mit Zufallswerten zurück:

```
AGAGATAACAACGACGACGATAAT
```

Die fett gedruckten Codons heben jeweils die Unterschiede zwischen den aktuellen und vorherigen Umsetzungen hervor.

Die Ergebnisse aus der einzelnen values-Anweisungen zeigen, dass die Funktion 'LSPep2ProbNuc' auf Grundlage zugänglicher Statistiken eines der möglichen Symbole auswählt. Diese Funktion unterscheidet sich von Funktion 'LSPep2AmbNuc', die mehrdeutige Symbole verwendet, wenn mehrere Übersetzungen möglich sind.

Die Funktion 'LSPep2ProbNuc' berücksichtigt für jedes Symbol die Übersetzungen mit der größten Wahrscheinlichkeit, und ersetzt anschließend jedes Symbol durch eine Zufallsübersetzung aus der zuvor gewählten Gruppe. Angenommen, eine Frequenztabelle enthält die folgenden Daten:

Tabelle 15. Beispieldaten einer Frequenztabelle

Aminosäure	Codon	Frequenz
Ala	GCG	0.17
Ala	GCA	0.13
Ala	GCT	0.17
Ala	GCC	0.53

Angenommen, die Peptidsequenz enthält vier „A"-Symbole (Ala). Die Funktion übersetzt A zweimal in GCC, einmal in GCG und einmal in GCT. Die Reihenfolge, in der die Funktion die Übersetzungen generiert, ist jedoch willkürlich. Die Abfrage könnte das erste A in jede der Übersetzungen aus der Gruppe (GCC, GCC, GCG, GCT} umsetzen. Das Ergebnis enthält stets zwei Elemente 'GCC', ein Element 'GCG' und ein Element 'GCT' in der Ausgabe-DNA-Sequenz. Mehrere Ausführungen der Funktion für die gleiche Sequenz können unter Umständen DNA-Sequenzen zurückgeben, in denen die Werte ausgetauscht sind.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2ProbNuc'" auf Seite 53
- „Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Fehlernachrichten" auf Seite 55
- „Benutzerdefinierte Funktion 'LSPep2AmbNuc' - Beispiel" auf Seite 50

Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Fehlernachrichten

Tabelle 16. Von der benutzerdefinierten Funktion 'LSPep2ProbNuc' ausgegebene Nachrichten

Fehlercode	Nachricht	Erläuterung
SQL0443N	Die Routine "DB2LS.LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Sequenz ist ungültig". SQLSTATE=38608	Die Eingabesequenz ist ungültig.
SQL0443N	Die Routine "DB2LS.LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Es wurde keine Übersetzung gefunden". SQLSTATE=38610	Die Datei mit der Codon-Frequenztabelle ist leer.
SQL0443N	Die Routine "LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Datei mit der Übersetzungstabelle kann nicht geöffnet werden". SQLSTATE=38612	Die Datei ist nicht vorhanden.
SQL0443N	Die Routine "DB2LS.LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Eine Zeile war beim Lesen aus der Datei zu lang". SQLSTATE=38614	Die Datei enthält Zeilen, die länger sind als zulässig.
SQL0443N	Die Routine "DB2LS.LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Ungültige Datendatei". SQLSTATE=38615	Das Dateiformat ist ungültig.
SQL0443N	Die Routine "LSPEP2PROBNUC" (spezifischer Name "LSPEP2PROBNUC") gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Übersetzungstabelle kann nicht erstellt werden". SQLSTATE=38611	Die Datei enthält ungültige Symbole.

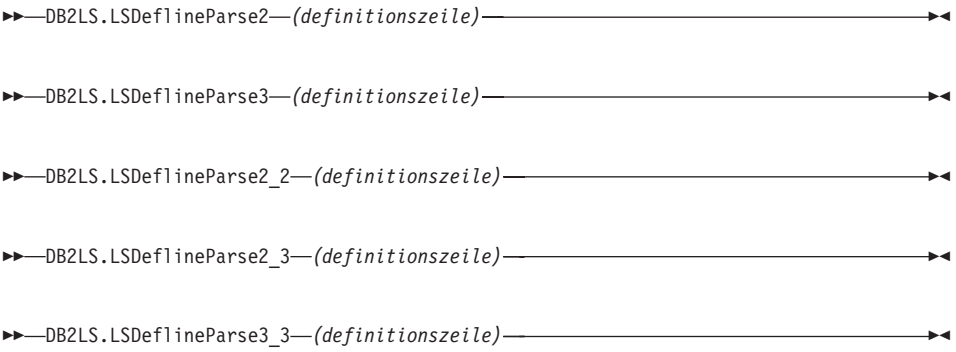
Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2ProbNuc'" auf Seite 53
- „Benutzerdefinierte Funktion 'LSPep2ProbNuc' - Beispiel" auf Seite 53

Benutzerdefinierte Funktionen zur syntaktischen Analyse von Definitionszeilen

Benutzerdefinierte Funktionen zur syntaktischen Analyse von Definitionszeilen führen eine syntaktische Analyse von Elementen einer Definitionszeile durch, um beispielsweise Verknüpfungen mit anderen Datenquellen gemäß Sequenzkennungen zu ermöglichen, deren syntaktische Analyse außerhalb der Definitionszeile erfolgte, oder um Vergleichselemente für Abschnitte der Definitionszeile zu bewerten (beispielsweise `'species = "human"'`). Die Definitionszeilenfunktionen umfassen die häufigsten Definitionszeilenformate. Zu den Beispielen gehören die Definitionszeilenelemente, die vom BLAST-Wrapper zurückgegeben werden oder die in einer Datendatei im FASTA-Format vorliegen.

Benutzerdefinierte LSDeflineParse-Funktionen



definitionszeile
Eine gültige Zeichenfolgedarstellung einer Definitionszeile im FASTA-Format. Die Zeichenfolge muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 1024 Byte.

Der Schemaname lautet DB2LS.

Jede LSDeflineParse-Funktion führt eine syntaktische Analyse der Felder der NSID (NCBI Standard FASTA Sequence Identifier) und der Beschreibung durch und stellt die Ergebnisse in Spalten in einer Tabelle dar. Definitionszeilen mit zusammengesetzten Definitionen werden in mehreren Zeilen ausgegeben, wobei jede Zeile die Definition einer einzelnen Komponente enthält.

LSDeflineParse2 führt eine syntaktische Analyse einer Definitionszeile durch, die eine aus zwei Feldern bestehende NSID enthält. Das Ergebnis der Funktion ist eine Tabelle mit vier Spalten:

Tabelle 17. Benutzerdefinierte Funktion 'LSDeflineParse2' - Spaltenbeschreibung für die Ergebnistabelle

Spaltenname	Beschreibung
ROWID	Eine ganze Zahl zur Nummerierung der von der Funktion zurückgegebenen Zeilen.
TAG	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung.
IDENTIFIER	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds in der NSID.
DESCRIPTION	Ein VARCHAR-Wert mit bis zu 1019 Zeichen.

LSDeflineParse3 führt eine syntaktische Analyse einer Definitionszeile durch, die eine aus drei Feldern bestehende NSID enthält. Das Ergebnis der Funktion ist eine Tabelle mit fünf Spalten:

Tabelle 18. Benutzerdefinierte Funktion 'LSDeflineParse3' - Spaltenbeschreibung für die Ergebnistabelle

Spaltenname	Beschreibung
ROWID	Eine ganze Zahl zur Nummerierung der von der Funktion zurückgegebenen Zeilen.
TAG	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung.
ACCESSION	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds in der NSID.
LOCUS	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des dritten Kennungsfelds in der NSID.
DESCRIPTION	Ein VARCHAR-Wert mit bis zu 1017 Zeichen.

LSDeflineParse2_2 führt eine syntaktische Analyse einer Definitionszeile durch, die über eine zusammengesetzte Kennung verfügt, die aus einem Paar von zwei verknüpften NSIDs mit jeweils zwei Feldern besteht. Das Ergebnis der Funktion ist eine Tabelle mit sechs Spalten:

Tabelle 19. Benutzerdefinierte Funktion 'LSDeflineParse2_2' - Spaltenbeschreibung für die Ergebnistabelle

Spaltenname	Beschreibung
ROWID	Eine ganze Zahl zur Nummerierung der von der Funktion zurückgegebenen Zeilen.
TAG1	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der ersten Kennung.
IDENTIFIER1	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der ersten NSID.
TAG2	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der zweiten Kennung.
IDENTIFIER2	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der zweiten NSID.
DESCRIPTION	Ein VARCHAR-Wert mit bis zu 1015 Zeichen.

LSDeflineParse2_3 führt eine syntaktische Analyse einer Definitionszeile durch, die über eine zusammengesetzte Kennung verfügt, die aus einer NSID mit zwei Feldern besteht, die mit einer aus drei Feldern bestehenden NSID verknüpft ist. Die Reihenfolge der Verknüpfung in der Eingabedefinitionszeile ist unerheblich, das heißt, es spielt keine Rolle, ob die NSID mit zwei Feldern vor der NSID mit drei Feldern steht oder umgekehrt. Das Ergebnis der Funktion ist eine Tabelle mit sieben Spalten:

Tabelle 20. Benutzerdefinierte Funktion 'LSDeflineParse2_3' - Spaltenbeschreibung für die Ergebnistabelle

Spaltenname	Beschreibung
ROWID	Eine ganze Zahl zur Nummerierung der von der Funktion zurückgegebenen Zeilen.
TAG1	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der aus zwei Feldern bestehenden Kennung.
IDENTIFIER	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der aus zwei Feldern bestehenden NSID.
TAG2	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der aus drei Feldern bestehenden Kennung.

Tabelle 20. Benutzerdefinierte Funktion 'LSDeflineParse2_3' - Spaltenbeschreibung für die Ergebnistabelle (Forts.)

Spaltenname	Beschreibung
ACCESSION	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der aus drei Feldern bestehenden NSID.
LOCUS	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des dritten Kennungsfelds der aus drei Feldern bestehenden NSID.
DESCRIPTION	Ein VARCHAR-Wert mit bis zu 1013 Zeichen.

LSDeflineParse3_3 führt eine syntaktische Analyse einer Definitionszeile durch, die über eine zusammengesetzte Kennung verfügt, die aus einem Paar von NSIDs mit jeweils drei Feldern besteht. Das Ergebnis der Funktion ist eine Tabelle mit acht Spalten:

Tabelle 21. Benutzerdefinierte Funktion 'LSDeflineParse3_3' - Spaltenbeschreibung für die Ergebnistabelle

Spaltenname	Beschreibung
ROWID	Eine ganze Zahl zur Nummerierung der von der Funktion zurückgegebenen Zeilen.
TAG1	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der ersten Kennung.
ACCESSION1	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der ersten NSID.
LOCUS1	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des dritten Kennungsfelds der ersten NSID.
TAG2	Ein VARCHAR-Wert mit bis zu drei Zeichen zur Darstellung der NSID-Markierung der zweiten Kennung.
ACCESSION2	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des zweiten Kennungsfelds der zweiten NSID.
LOCUS2	Ein VARCHAR-Wert mit bis zu 20 Zeichen zur Darstellung des dritten Kennungsfelds der zweiten NSID.
DESCRIPTION	Ein VARCHAR-Wert mit bis zu 1014 Zeichen.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSDeflineParse' - Beispiele" auf Seite 60

Benutzerdefinierte Funktion 'LSDeflineParse' - Beispiele

Dieser Abschnitt enthält sieben Beispiele, die zeigen, wie die benutzerdefinierten LSDeflineParse-Funktionen aus Definitionszeilen mittels syntaktischer Analyse Ergebnistabellen erstellen.

Die folgende Beispielabfrage und Ergebnistabelle zeigen, wie die benutzerdefinierte Funktion 'LSDeflineParse2' eine Definitionszeile, die eine aus zwei Feldern bestehende NSID enthält, syntaktisch analysiert:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 22. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse2'

Spaltenname	Daten
ROWID	1
TAG	gi
IDENTIFIER	12346
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Die folgende Beispielabfrage und Ergebnistabelle zeigen, wie die benutzerdefinierte Funktion 'LSDeflineParse3' eine Definitionszeile, die eine aus drei Feldern bestehende NSID enthält, syntaktisch analysiert:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 23. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse3'

Spaltenname	Daten
ROWID	1
TAG	gb
ACCESSION	U37104
LOCUS	APU37104
DESCRIPTION	Aethia pusilla cytochrome b gene

Die folgende Beispielabfrage und Ergebnistabelle zeigen, wie die benutzerdefinierte Funktion 'LSDeflineParse2_2' eine Definitionszeile syntaktisch analysiert, die eine zusammengesetzte Kennung enthält, die aus einem Paar aus NSIDs mit jeweils zwei Feldern besteht:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 24. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse2_2'

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

Die folgende Beispielabfrage enthält eine Definitionszeile mit einer zusammengesetzten Kennung, die sich aus einer NSID mit 2 Feldern zusammensetzt, die mit einer aus drei Feldern bestehenden NSID verknüpft ist. Das Beispiel zeigt, wie die Funktion 'LSDeflineParse2_3' diese Definitionszeile syntaktisch analysiert:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 25. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse2_3'

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

Die folgende Beispielabfrage enthält eine Definitionszeile mit einer zusammengesetzten Kennung, die sich aus einer NSID mit 3 Feldern zusammen-

setzt, die mit einer aus zwei Feldern bestehenden NSID verknüpft ist. Das Beispiel zeigt, wie die Funktion 'LSDeflineParse2_3' diese Definitionszeile syntaktisch analysiert:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 26. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse2_3'

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

Die folgende Beispielabfrage und Ergebnistabelle zeigen, wie die benutzerdefinierte Funktion 'LSDeflineParse3_3' eine Definitionszeile syntaktisch analysiert, die eine zusammengesetzte Kennung enthält, die aus einem Paar aus NSIDs mit jeweils drei Feldern besteht:

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

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 27. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse3_3'

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

Zur syntaktischen Analyse einer zusammengesetzten Definitionszeile kann eine beliebige der benutzerdefinierten defline-Funktionen verwendet werden. Die folgende Beispielabfrage enthält eine zusammengesetzte Definitionszeile mit mehreren Definitionen, die durch ein Steuerung-A-Zeichen voneinander getrennt werden. Dieser Definitionszeilentyp kommt in der NCBI-Datenbank 'nr' der nicht-redundanten Proteine vor. Das Beispiel zeigt, wie die Funktion 'LSDeflineParse2_3' diese Definitionszeile syntaktisch analysiert:

```
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
```

Die Ergebnistabelle enthält die folgenden Daten:

Tabelle 28. Ergebnisdaten der benutzerdefinierten Funktion 'LSDeflineParse2_3'

Spaltenname	Daten	Daten
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

Zugehörige Referenzen:

- „Benutzerdefinierte LSDeflineParse-Funktionen" auf Seite 56

Benutzerdefinierte Funktionen für allgemeine Mustererkennung

Die benutzerdefinierten Funktionen für allgemeine Mustererkennung identifizieren relevante Bereiche in einer angegebenen Zeichenfolge, wie beispielsweise Nukleotid- oder Peptidsequenzen.

Benutzerdefinierte Funktion 'LSPatternMatch'

```
►►—DB2LS.LSPatternMatch—(eingabe-zeichenfolge, muster)———————————————►◄
```

eingabe-zeichenfolge
Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

muster

Das Muster gemäß Angabe in einem gültigen regulären PERL-Ausdruck. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Mit Hilfe der benutzerdefinierten Funktion 'LSPatternMatch' können Sie die Eingabe-Nukleotidsequenz oder Eingabe-Peptidsequenz nach einem angegebenen Muster durchsuchen.

Das Ergebnis der Funktion ist eine ganze Zahl zur Darstellung der Position der ersten Übereinstimmung des Musters in der Sequenz. Ist keine Übereinstimmung vorhanden, gibt die Funktion den Wert Null zurück.

Sind Muster vorhanden, die in der PROSITE-Syntax geschrieben wurden, können Sie diese mit Hilfe der benutzerdefinierten Funktion 'LSPrositePattern' in die PERL-Syntax umwandeln. Anschließend können Sie die umgewandelte Syntax mit der benutzerdefinierten Funktion 'LSPatternMatch' verwenden.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPatternMatch' - Beispiel" auf Seite 64
- „Benutzerdefinierte Funktion 'LSPrositePattern'" auf Seite 66

Benutzerdefinierte Funktion 'LSPatternMatch' - Beispiel

In folgendem Beispiel wird nach der Anfangsposition der Zeichenfolge gesucht, die „coward", „cowage", „cowboy" oder „cowl" entspricht.

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

Die Funktion sucht nach Zeichen und gibt in diesem Beispiel den Wert neun zurück. Die Zeichenfolge „cowboy" beginnt an Position neun, wobei davon ausgegangen wird, dass die erste Position eins ist.

Im nächsten Beispiel wird nach der Anfangsposition der Zeichenfolge gesucht, die „not " oder „non" entspricht:

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

Die Funktion sucht nach Zeichen und gibt in diesem Beispiel den Wert sieben zurück. Die Zeichenfolge „not" beginnt an Position sieben, wobei davon ausgegangen wird, dass die erste Position eins ist.

'LSPatternMatch' ist nützlich in SELECT-Anweisungen zum Filtern der Ergebnisse anhand der PERL-Syntax. Diese Syntax ist leistungsfähiger als die SQL-

Anweisung LIKE. In folgendem Beispiel wird die Funktion 'LSPatternMatch' für eine BLAST-Ausgabe verwendet, um die Gene zu filtern, die einem bestimmten Muster entsprechen:

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

Wenn Sie mit der PROSITE-Syntax besser vertraut sind, können Sie mit der vorstehenden Abfrage die Funktion 'LSPrositePattern' verwenden. Ändern Sie in diesem Fall die Abfrage wie folgt:

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

Die Mustererkennungsfunktionen sind nützlich für die Suche in anderen Texttypen sowie in Nukleotid- und Peptidsequenzen. Falls die Leistungsfähigkeit ein Problem darstellen könnte, sollte die Verwendung der SQL-Anweisung LIKE in Betracht gezogen werden.

Das folgende Beispiel zeigt eine Abfrage, die BLAST-HSP-Alignments auf der Grundlage von Proteinmotiven filtert, die in der Betreff- oder Zielzeile des Alignments gefunden wurden. Grundlage des Beispiels ist folgende Publikation: 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.

Die folgende Abfrage gibt nur diejenigen Alignments zurück, in denen die betreffende Sequenz die P-Schleife-ATPase-Domäne [GA]xxxxGK[ST] enthält. Die Abfrage verwendet das Protein CED4 (den Apoptoseregulator des Nematoden *Caenorhabditis elegans*) als Abfragesequenz für die NCBI-Datenbank der nicht-redundanten Proteinsequenzen. Die Datenbank ruft die BLAST-Abfragesequenz aus der Übersetzung des CDS-Merkmals des GenBAnk-Eintrags X69016 ab.

```
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;
```

Mit Hilfe der nächsten Beispielabfrage können Sie in einer Genomsequenz nach HSPs suchen, die mutmaßliche Einzel-Nukleotid-Polymorphismen (Single Nucleotide Polymorphisms, SNPs) im Hinblick auf eine kanonische Abfragesequenz enthalten. Grundlage dieser Abfrage ist folgende Publikation:

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.

Die Abfrage verwendet die Mustererkennung für die BLAST-HSP-Mittellinie (Midline), um ein Muster von ≥20 perfekten Übereinstimmungen zu finden, gefolgt von einer einzelnen Diskrepanz, gefolgt von ≥20 perfekten Übereinstimmungen, das heißt, zunächst 20 "|"-Zeichen, dann ein einzelnes Leerzeichen und anschließend 20 "|"-Zeichen in der Mittellinie des Alignments.

Außerdem zeigt dieses Beispiel die Verwendung der benutzerdefinierten Funktion 'LSPatternMatch' für Zeichenfolgen, bei denen es sich nicht um Nukleotid- oder Peptidsequenzen handelt.

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

Sie können die vorstehende Abfrage wie folgt umschreiben:

```
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
```

Diese zweite Abfrage gibt diejenigen BLAST-Zeilen zurück, die eine Übereinstimmung enthalten, zusammen mit der passenden Zeichenfolge und den entsprechenden Positionen in der Sequenz.

Die BLAST-Ausgabe (BlastOutput) ist eine Sicht eines BlastN-Kurznamens.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPrositePattern' - Beispiel" auf Seite 67
- „Benutzerdefinierte Funktion 'LSPatternMatch'" auf Seite 63
- „Benutzerdefinierte Funktion 'LSPrositePattern'" auf Seite 66

Benutzerdefinierte Funktion 'LSPrositePattern'

muster

Das Muster, das der von der PROSITE-Syntax angegebenen Syntax entspricht. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Mit Hilfe der benutzerdefinierten Funktion 'LSPrositePattern' können Sie eine Umwandlung von der PROSITE-Syntax in die PERL-Syntax vornehmen. Anschließend können Sie die umgewandelte Syntax mit den benutzerdefinierten Funktionen 'LSPatternMatch', 'LSMultiMatch' und 'LSMultiMatch3' verwenden.

Das Ergebnis der Funktion ist eine Zeichenfolge zur Darstellung eines regulären Ausdrucks in der PERL-Syntax. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPrositePattern' - Beispiel" auf Seite 67
- „Benutzerdefinierte Funktion 'LSPatternMatch'" auf Seite 63

Benutzerdefinierte Funktion 'LSPrositePattern' - Beispiel

In folgendem Beispiel wird ein Muster aus der PROSITE-Syntax in die PERL-Syntax umgewandelt.

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

Die Funktion wandelt das Eingabemuster in PROSITE-Syntax in ein äquivalentes Muster in PERL-Syntax um, wie aus folgendem Beispiel ersichtlich wird:

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

Im nächsten Beispiel wird ein anderes Muster aus der PROSITE-Syntax in die PERL-Syntax umgewandelt:

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

Die Funktion übersetzt die Zeichenfolge aus der PROSITE-Syntax auf Grundlage des Eingabemusters und gibt Folgendes zurück:

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

Im nächsten Beispiel wird das Muster, das dem PROSITE-Datenbankeintrag mit der ID-Nummer PS01205 entspricht, in ein PERL-Muster umgewandelt, das von den Mustererkennungsfunktionen als Eingabe verwendet wird.

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

Die Abfrage gibt folgendes Ergebnis zurück:

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

Das nächste Beispiel zeigt, wie Sie die Funktion in einer Abfrage verwenden können. Die Abfrage druckt nur diejenigen Sequenzen aus, die dem angegebenen PROSITE-Muster entsprechen.

```
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;
```

Im nächsten Beispiel wird das Muster umgewandelt, das dem PROSITE-Eintrag mit der ID PS00261 entspricht:

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

Die Abfrage gibt folgendes Ergebnis zurück:

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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPatternMatch' - Beispiel" auf Seite 64
- „Benutzerdefinierte Funktion 'LSPrositePattern'" auf Seite 66

Unterstützung für reguläre Ausdrücke

Die Unterstützung für reguläre Ausdrücke wird vom PCRE-Bibliothekspaket zur Verfügung gestellt, einer von Philip Hazel geschriebenen Software mit offener Quelle, deren Urheberrecht bei der University of Cambridge, England, liegt.

Die Quelle befindet sich unter ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Benutzerdefinierte Funktion 'GeneWise'

Die benutzerdefinierte Funktion 'GeneWise' führt ein Alignment einer Proteinsequenz mit einer Genomsequenz durch.

'GeneWise' ist eine häufig verwendete Komponente, die ein Alignment einer Proteinsequenz mit einer genomischen DNA-Sequenz durchführt, unter Berücksichtigung von Introns- und Frameshifting-Fehlern.

Verbindung herstellen mit 'GeneWise'

Dieser Abschnitt beschreibt, wie eine Verbindung zur Bibliothek 'GeneWise' hergestellt wird.

Vorgehensweise:

1. Laden Sie das Wise2-Paket Version 2.1.20c von 'http://www.ebi.ac.uk/Wise2' herunter.
2. Erweitern Sie das Archiv im gewünschten Ordner.
3. Kompilieren Sie das Paket mit pthread-Unterstützung. Die Dokumentation von Wise2 enthält weitere Informationen zu diesem Schritt.
4. Führen Sie den Befehl **make api** im entsprechenden Stammverzeichnis aus.

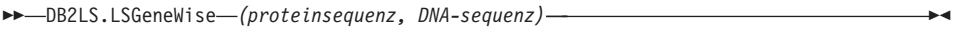
5. Setzen Sie die Umgebungsvariable WISE2_HOME so, dass sie auf das Stammverzeichnis des Wise2-Pakets verweist.
6. Setzen Sie die Variable WISECONFIGDIR in der Datei 'sqllib/cfg/db2dj.ini' so, dass sie auf das Unterverzeichnis 'wisecfg' verweist.

   Beispiel: Ist das Wise2-Paket im Verzeichnis '/usr/wise2.1.20c/' installiert, fügen Sie der Datei 'db2dj.ini' Folgendes hinzu: WISECONFIGDIR=/usr/wise2.1.20c/wisecfg/.
7. Führen Sie über das Verzeichnis 'sqllib/bin' den Befehl **djxlinkLSGeneWise** aus und überprüfen Sie die Ausgabe.
8. Überprüfen Sie 'djxlinkLSGeneWise.out' im Verzeichnis 'sqllib/lib'.
9. Wurden keine Fehler dokumentiert, wurde die Bibliothek erfolgreich erstellt.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSGeneWise'" auf Seite 69

Benutzerdefinierte Funktion 'LSGeneWise'



proteinsequenz
Eine gültige Zeichenfolgedarstellung, die eine Peptidsequenz beschreibt. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

DNA-sequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotidsequenz beschreibt. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Tabelle 29 zeigt die von der Funktion 'LSGeneWise' zurückgegebene einzeilige Ausgabetabelle

Tabelle 29. Spaltennamen, Typen und Beschreibungen für die von der Funktion 'LSGeneWise' zurückgegebene Ausgabetabelle

Spaltenname	Typ	Beschreibung
PROTEIN_OFFSET	INTEGER	Zeigt das Start-Offset in der Eingabe-Proteinsequenz, an dem ein Alignment gefunden wurde.
DNA_OFFSET	INTEGER	Zeigt das Start-Offset in der Eingabe-DNA-Sequenz, an dem ein Alignment gefunden wurde.

Tabelle 29. Spaltennamen, Typen und Beschreibungen für die von der Funktion 'LSGeneWise' zurückgegebene Ausgabetabelle (Forts.)

Spaltenname	Typ	Beschreibung
PROTEIN	VARCHAR(32672)	Ein Fragment aus der Eingabesequenz, das die alignierte Sequenz darstellt.
SIMILARITY	VARCHAR(32672)	Zeigt die Übereinstimmung zwischen dem Protein und der DNA-Sequenz. Perfekte Übereinstimmungen werden mit dem entsprechenden Symbolbuchstaben markiert. Nicht perfekte Übereinstimmungen mit einem positiven Score werden mit einem Pluszeichen (″+″) und Diskrepanzen mit einem Leerzeichen markiert.
TRANSLATED_DNA	VARCHAR(32672)	Die übersetzte DNA-Sequenz. Unter Umständen enthält die Sequenz Gedankenstriche und Sondersymbole wie beispielsweise Deletionen und Introns.
DNA	VARCHAR(32672)	Die DNA-Sequenz mit Sondermarkierungen wie beispielsweise Frameshifting und Introns.

Die Übereinstimmung zwischen der Ausgabe des Programms 'GeneWise' und der Ausgabe der benutzerdefinierten Funktion 'LSGeneWise' sieht wie folgt aus:

- Die vom Programm 'GeneWise' gedruckten Protein- und DNA-Offsets entsprechen den Spalten PROTEIN_OFFSET und DNA_OFFSET.
- Die von 'GeneWise' in der ersten Zeile gedruckte Proteinsequenz entspricht der Spalte PROTEIN.
- Die Zeile 'similarity', die zweite Zeile in der Ausgabe von 'GeneWise', entspricht der Spalte SIMILARITY.
- Die dritte Zeile in der Ausgabe von 'GeneWise' entspricht der Spalte TRANSLATED_DNA.
- Die vierte, fünfte und sechste Zeile der Ausgabe von 'GeneWise' sind (vertikal gelesen) zur Spalte DNA zusammengefasst.

Mit der benutzerdefinierten Funktion 'LSGeneWise' können Sie ein Alignment einer Proteinsequenz mit einer genomischen DNA-Sequenz durchführen, unter Berücksichtigung von Introns- und Frameshifting-Fehlern.

Weitere Informationen zur Ausgabe der benutzerdefinierten Funktion 'LSGeneWise' finden Sie unter 'http://www.ebi.ac.uk/Wise2'.

Zugehörige Tasks:

- „Verbindung herstellen mit 'GeneWise'" auf Seite 68

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSGeneWise' - Beispiel" auf Seite 71

Benutzerdefinierte Funktion 'LSGeneWise' - Beispiel

Das folgende Beispiel zeigt eine Abfrage mit der benutzerdefinierten Funktion 'LSGeneWise' und die entsprechenden Ergebnisdaten.

```
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;
```

Tabelle 30. Ergebnistabelle

Spalte	Daten
PROTEIN_OFFSET	23
DNA_OFFSET	14
PROTEIN	KLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKET GKKRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVD VKKALPKQNDQQGGGGRGGPGGRAGGNRGNMGG 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...

Zugehörige Tasks:

- „Verbindung herstellen mit 'GeneWise'" auf Seite 68

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSGeneWise'" auf Seite 69

Benutzerdefinierte Motivfunktionen

Benutzerdefinierte Motivfunktionen suchen übereinstimmende Muster in Nukleotid- oder Aminosäuresequenzen.

Benutzerdefinierte Funktion 'LSBarCode'

eingabe-zeichenfolge
Eine gültige Zeichenfolge zur Darstellung einer HSP-Alignments zwischen zwei Sequenzfragmenten. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Mit der benutzerdefinierten Funktion 'LSBarCode' wird anhand einer Eingabesequenz eine andere Sequenz generiert, indem jedes Zeichen (außer Leerzeichen und Pluszeichen) durch ein vertikales Balkensymbol (|) ersetzt wird.

Das Ergebnis der Funktion ist eine variable Zeichenfolge zur Darstellung einer Barcode-Sequenz.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSBarCode' - Beispiel" auf Seite 72
- „Benutzerdefinierte Funktion 'LSMultiMatch'" auf Seite 74
- „Benutzerdefinierte Funktion 'LSMultiMatch3'" auf Seite 76

Benutzerdefinierte Funktion 'LSBarCode' - Beispiel

In diesem Beispiel wird aus einer Zeichenfolgesequenz ein Barcode erstellt:

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

Das Ergebnis aus den vorstehenden Werten sieht wie folgt aus:

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

Das nächste Beispiel zeigt eine realistischere Verwendung dieser Funktion. Angenommen, ein Forscher führt eine BLAST-Suche durch und will als Rückgabewerte nur solche HSP-Alignments erhalten, die unter ihren perfekten Übereinstimmungen einen Anteil an Prolinen von weniger als 25% haben. In diesem Beispiel wird mit Hilfe der Funktion der Prozentsatz an Prolinen (Symbol 'P') unter den perfekten Übereinstimmungen in einem von BLAST zurückgegebenen Alignment berechnet. Hierbei ist zu beachten, dass in diesem Beispiel auch die benutzerdefinierte Funktion 'LSMultiMatch3' aufgerufen

wird. Die Abfrage verwendet eine Abgleichsfunktion, um perfekte Übereinstimmungen zu ermitteln. Sie wird in dieser Abfrage zusammen mit der Funktion 'LSBarCode' verwendet, da BLAST in einem Alignment nicht immer eine Folge von Balken („|") zurückgibt. Das nachstehende Beispiel zeigt Folgendes:

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

Verwenden Sie die Funktion 'LSBarCode', um zu gewährleisten, dass die Ausgabe mit der korrekten Balkenfolge ausgerichtet (aligniert) wird. Die Funktion ersetzt alle Zeichen (ausgenommen Leerzeichen und Pluszeichen) durch einen vertikalen Balken.

```
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;
```

In dieser Abfrage ist die BLAST-Ausgabe (BlastOutput) eigentlich eine Sicht eines BLAST-Kurznamens. Die Abfrage verwendet die Funktion 'LSMultiMatch3', um die perfekten Übereinstimmungen beim Alignment zurückzugeben. Bei der ersten Verwendung werden die perfekten Übereinstimmungen für Symbol „P" zurückgegeben, bei der zweiten Verwendung werden alle perfekten Übereinstimmungen zurückgegeben. Tabelle 31 zeigt eine Zeile aus der Ergebnistabelle.

Tabelle 31. Beispielergebniszeile

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

Grundlage der vorstehenden Abfrage ist folgende Publikation: 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.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSMultiMatch3' - Beispiel" auf Seite 76
- „Benutzerdefinierte Funktion 'LSBarCode'" auf Seite 72

Benutzerdefinierte Funktion 'LSMultiMatch'

eingabenukleotidsequenz oder eingabepeptidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotid- oder Peptidsequenz beschreibt. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

muster
Das Muster, das der von der PERL-Sprache angegebenen Grammatik entspricht. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Mit Hilfe der benutzerdefinierten Funktion 'LSMultiMatch' wird für jede Übereinstimmung, die sich in der Eingabesequenz nicht überschneidet, eine Tabelle zurückgegeben. Jede Tabelle besteht aus einer Anfangsposition und dem übereinstimmenden Sequenzfragment.

Das Ergebnis der Funktion wird in einer Tabelle mit zwei Spalten dargestellt. Die erste Spalte enthält eine ganze Zahl zur Darstellung der Anfangsposition einer Übereinstimmung des Musters in der Sequenz. Die zweite Spalte enthält das übereinstimmende Sequenzfragment.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSMultiMatch' - Beispiel" auf Seite 74
- „Benutzerdefinierte Funktion 'LSBarCode'" auf Seite 72
- „Benutzerdefinierte Funktion 'LSMultiMatch3'" auf Seite 76

Benutzerdefinierte Funktion 'LSMultiMatch' - Beispiel

Dieses Beispiel sucht nach der Position und den passenden Fragmenten aller sich nicht überschneidender Übereinstimmungen aus der Eingabe.

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

Die Abfrage gibt eine Tabelle zurück, die auf dieser SELECT-Anweisung basiert und die Ergebnisse der Übereinstimmungen zeigt:

Tabelle 32. Ergebnis der Funktion 'LSMultiMatch' mit mehreren zurückgegebenen Zeilen

POSITION	MATCH
7	not
15	non

Die Funktion 'LSMultiMatch' gibt die Position und die passende Zeichenfolge aller Übereinstimmungen zurück. Das folgende Beispiel durchsucht 'Entrez Nucleotide' nach Sequenzeinträge, die ein bestimmtes Motiv enthalten. Die Abfrage druckt die Sequenzkennungen und die übereinstimmenden Sequenzen aus. Die Untermuster „.{0,9}" am Anfang und am Ende müssen vor und nach der Sequenz bis neun Zeichen übereinstimmen. Die Abfrage druckt auch diese Zeichen.

```
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;
```

Die Abfrage gibt folgendes Ergebnis zurück:

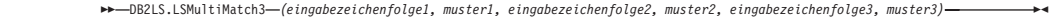
Tabelle 33. Entrez-Suchdaten

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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSMultiMatch'" auf Seite 74
- „Benutzerdefinierte Funktion 'LSBarCode'" auf Seite 72
- „Benutzerdefinierte Funktion 'LSMultiMatch3'" auf Seite 76

Benutzerdefinierte Funktion 'LSMultiMatch3'



eingabe-zeichenfolgen
Eine gültige Zeichenfolgedarstellung, die eine Nukleotid- oder Peptidsequenz, oder eine HSP_Midline-Zeichenfolge aus einem BLAST-Alignment beschreibt. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

muster
Das Muster, das der von der PERL-Sprache angegebenen Grammatik entspricht. Die Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Mit Hilfe der benutzerdefinierten Funktion 'LSMultiMatch3' werden drei Muster und drei Zeichenfolgen als Eingabe verwendet und alle Positionen zurückgegeben, an denen alle drei Zeichenfolgen mit den entsprechenden Mustern übereinstimmen. Sie können diese benutzerdefinierte Funktion verwenden, um eine Mustererkennung für ein Alignment durchzuführen.

Das Ergebnis der Funktion ist eine Tabelle mit vier Spalten. Die erste Spalte enthält eine ganze Zahl zur Darstellung der Anfangsposition einer Übereinstimmung des Musters in allen Sequenzen. Die Funktion verankert alle Zeichenfolgen zusammen an der ersten Position. Die zweite, dritte und vierte Spalte enthalten die übereinstimmenden Sequenzfragmente.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSMultiMatch3' - Beispiel" auf Seite 76
- „Benutzerdefinierte Funktion 'LSMultiMatch'" auf Seite 74
- „Benutzerdefinierte Funktion 'LSBarCode'" auf Seite 72

Benutzerdefinierte Funktion 'LSMultiMatch3' - Beispiel

Das folgende Beispiel verwendet die Funktion, um den Prozentsatz eines bestimmten Aminosäuresymbols unter den von BLAST zurückgegebenen perfekten Übereinstimmungen zu berechnen. Hierbei ist zu beachten, dass in diesem Beispiel auch die benutzerdefinierte Funktion 'LSBarCode' aufgerufen wird. Diese Funktion wird von der Abfrage benötigt, da BLAST in einem Alignment nicht immer eine Folge von Balken („|") zurückgibt.

Das nachstehende Beispiel zeigt Folgendes:

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

Verwenden Sie die Funktion 'LSBarCode' zum Umwandeln der Sequenz, um zu gewährleisten, dass die Ausgabe mit der korrekten Balkenfolge ausgerichtet (aligniert) wird. Die Funktion ersetzt alle Zeichen (außer Leerzeichen und „+"-Zeichen) durch einen vertikalen Balken.

```
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;
```

In dieser Abfrage ist die BLAST-Ausgabe (BlastOutput) eine Sicht einer BLAST-Auswahl (SELECT). Die Abfrage verwendet die Funktion 'LSMultiMatch3', um die perfekten Übereinstimmungen beim Alignment zurückzugeben. Bei der ersten Verwendung werden die perfekten Übereinstimmungen für Symbol „P" zurückgegeben, bei der zweiten Verwendung werden alle perfekten Übereinstimmungen zurückgegeben. Tabelle 34 zeigt eine Zeile aus der Ergebnistabelle.

Tabelle 34. Beispielergebniszeile

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

Grundlage der vorstehenden Abfrage ist folgende Publikation: 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.

Das folgende Beispiel sucht nach drei getrennten Mustern in drei getrennten Zeichenfolgefragmenten:

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

Die Abfrage gibt die Positionen und die passenden Zeichenfolgen aller Übereinstimmungen zurück, wie aus der folgenden Tabelle ersichtlich wird:

Tabelle 35. Ergebnis einer Mehrfachübereinstimmung mit drei Eingabemustern

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

Das nächst Beispiel sucht drei getrennte Muster in drei getrennten Zeichenfolgefragmenten:

```
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
```

Die Ergebnisse werden in nachfolgender Tabelle dargestellt:

Tabelle 36. Ergebnis einer Mehrfachübereinstimmung mit drei Eingabemustern

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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSBarCode' - Beispiel" auf Seite 72
- „Benutzerdefinierte Funktion 'LSBarCode'" auf Seite 72
- „Benutzerdefinierte Funktion 'LSMultiMatch3'" auf Seite 76

Benutzerdefiniert Umkehrfunktionen

Benutzerdefinierte Umkehrfunktionen kehren Nukleotid- oder Aminosäuresequenzen um.

Benutzerdefinierte Funktion 'LSRevComp'

eingabe-nukleotidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotidsequenz beschreibt. Die Sequenz kann IUPAC-Ambiguitätscodes enthalten. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

Der Schemaname lautet DB2LS.

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 32672 Byte. Die Zeichenfolge ist eine Darstellung des reversen (umgekehrten) Komplements der Nukleotidsequenz.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevComp' - Beispiel" auf Seite 79
- „Benutzerdefinierte Funktion 'LSRevNuc'" auf Seite 81
- „Benutzerdefinierte Funktion 'LSRevPep'" auf Seite 82

Benutzerdefinierte Funktion 'LSRevComp' - Beispiel

Sie können die Funktion 'LSRevComp' immer dann in einer SQL-Anweisung verwenden, wenn Sie eine integrierte Funktion verwenden würden, die eine Nukleotidsequenz akzeptiert. Beispiel:

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

In diesem Beispiel wird die Funktion verwendet, um das reverse (umgekehrte) Komplement der Eingabesequenz zurückzugeben, die von einer Hostvariablen übergeben wurde.

Bei Verwendung einer ungültigen Zeichenfolge oder eines ungültigen Datentyps wird die folgende Fehlernachricht angezeigt:

```
SQL0443N Die Routine "DB2LS.LSREVCOMP" (spezifischer Name "LSREVCOMP")
gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Sequenz
ist ungültig". SQLSTATE=38608
```

Ist das Alphabet der Eingabe nicht korrekt, wird eine Ausnahmebedingung ausgegeben.

Das folgende Beispiel zeigt die Funktionsweise der benutzerdefinierten Funktion 'LSRevComp' in einer Abfrage:

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

Die Abfrageergebnisse werden in nachfolgender Tabelle dargestellt:

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

AGTATTGGTCAATCTTTTGAT          ATCAAAAGATTGACCAATACT

TGGTCAATCTTTTGATA              TATCAAAAGATTGACCA

TTGGCCAATCTTTTGATATCC          GGATATCAAAAGATTGGCCAA

TCAATCTTTTGATATCC              GGATATCAAAAGATTGA

GGATATCAAAAGATTGA              TCAATCTTTTGATATCC


     5 Satz/Sätze ausgewählt.
```

Sie können die Umkehrfunktion zusammen mit anderen benutzerdefinierten Life Sciences-Funktionen verwenden, um das reverse (umgekehrte) Komplement einer Nukleotidsequenz zu übersetzen, wie aus folgendem Beispiel ersichtlich wird:

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

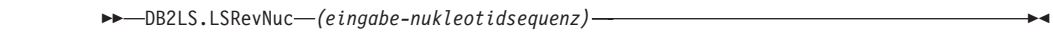
Die Abfrage gibt folgendes Ergebnis zurück:

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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevComp'" auf Seite 79

Benutzerdefinierte Funktion 'LSRevNuc'



eingabe-nukleotidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotidsequenz beschreibt. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte. Die Nukleotidsequenz muss ein Teil des DNA-Alphabets sein oder das gesamte DNA-Alphabet darstellen.

Der Schemaname lautet DB2LS.

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 32672 Byte. Die Zeichenfolge ist eine Darstellung der umgekehrten Reihenfolge der Nukleotidsequenz.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevNuc' - Beispiel" auf Seite 81
- „Benutzerdefinierte Funktion 'LSRevComp'" auf Seite 79
- „Benutzerdefinierte Funktion 'LSRevPep'" auf Seite 82

Benutzerdefinierte Funktion 'LSRevNuc' - Beispiel

Sie können die Funktion 'LSRevNuc' immer dann in einer SQL-Anweisung verwenden, wenn Sie eine integrierte Funktion verwenden würden, die eine Nukleotidsequenz akzeptiert. Beispiel:

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

In diesem Beispiel wird die Funktion verwendet, um die von einer Hostvariablen übergebenen Eingabedaten umzukehren.

Bei Verwendung einer ungültigen Zeichenfolge oder eines ungültigen Datentyps wird die folgende Fehlernachricht angezeigt:

```
SQL0443N Die Routine "DB2LS.LSREVNUC" (spezifischer Name "LSREVNUC")
gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Sequenz
ist ungültig". SQLSTATE=38608
```

Das folgende Beispiel zeigt die Verwendung der benutzerdefinierten Funktion 'LSRevNuc' in einer Abfrage.

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

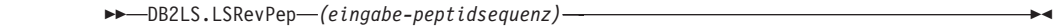
Die folgende Ergebnistabelle zeigt die von der Abfrage zurückgegebenen umgekehrten Nukleotidsequenzen:

```
HSP_H_SEQ                                          REV_HSP_H_SEQ

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

GCGCTAGCCCCCTACGTATTAC                             CATTATGCATCCCCCGATCGCG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG


    5 Satz/Sätze ausgewählt.
```

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevNuc'" auf Seite 81

Benutzerdefinierte Funktion 'LSRevPep'



eingabe-peptidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Peptidsequenz beschreibt. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte. Die Eingabesequenz muss Teil des Proteinalphabets sein.

Der Schemaname lautet DB2LS.

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 32672 Byte. Die Zeichenfolge ist eine Darstellung der umgekehrten Reihenfolge der Peptidsequenz.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevPep' - Beispiel" auf Seite 83
- „Benutzerdefinierte Funktion 'LSRevComp'" auf Seite 79
- „Benutzerdefinierte Funktion 'LSRevNuc'" auf Seite 81

Benutzerdefinierte Funktion 'LSRevPep' - Beispiel

Sie können die Funktion 'LSRevPep' immer dann in einer SQL-Anweisung verwenden, wenn Sie eine integrierte Funktion verwenden würden, die eine Peptidsequenz akzeptiert. Zum Beispiel:

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

In diesem Beispiel wird die Funktion verwendet, um die von einer Hostvariablen übergebenen Eingabedaten umzukehren.

Bei Verwendung einer ungültigen Zeichenfolge oder eines ungültigen Datentyps wird die folgende Fehlernachricht angezeigt:

```
SQL0443N Die Routine "DB2LS.LSREVPEP" (spezifischer Name "LSREVPEP")
gab einen SQLSTATE-Fehler zurück. Der Diagnosetext lautet "Die Sequenz
ist ungültig". SQLSTATE=38608
```

Das folgende Beispiel zeigt die Verwendung der benutzerdefinierten Funktion 'LSRevPep' in einer Abfrage:

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

Die folgende Ergebnistabelle zeigt die von der Abfrage zurückgegebenen umgekehrten Peptidsequenzen.

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

MLCEIECRALSTAHTRLIHDFEPRDALTYL...  HDETFINKGELYTLADRPEFDHILRTHATS...

RVVSTEHTRLVTDAYPEFSISFTATKN        NKTATFSISFEPYADTVLRTHETSVVR

STAHIRVLRDMVPGDEITCFYGSEFF         FFESGYFCTIEDGPVMDRLVRIHATS

AHTRRCPDHEPRGVITYL                 LYTIVGRPEHDPCRRTHA


  4 Satz/Sätze ausgewählt.
```

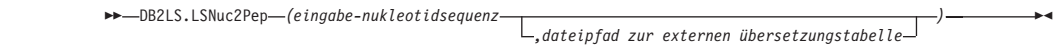
Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevPep'" auf Seite 82

Übersetzung

Die benutzerdefinierten Übersetzungsfunktionen wandeln eine Nukleotidsequenz in eine Peptidsequenz um.

Benutzerdefinierte Funktion 'LSNuc2Pep'



eingabe-nukleotidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotidsequenz beschreibt. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

dateipfad zur externen Übersetzungstabelle
Wenn Sie eine angepasste Übersetzungstabelle verwenden, müssen Sie den entsprechenden Dateipfad angeben, damit die Tabelle gefunden wird. Der Zeichenfolgewert des Pfads darf 255 Zeichen nicht überschreiten.

Der Schemaname lautet DB2LS.

Das Ergebnis der Funktion ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 10890 Byte. Die Zeichenfolge ist eine Darstellung der Peptidsequenz.

Die Eingabe ist eine Nukleotidsequenz, für die der IUB-Zeichensatz verwendet wird. Bei diesen Funktionen wird davon ausgegangen, dass das erste Codon am ersten Zeichen der Nukleotidsequenz beginnt. Beginnt das erste Codon nicht am ersten Zeichen der Nukleotidsequenz, verwenden Sie für die Eingabesequenz eine SUBSTR-Funktion.

Das Ergebnis der Funktion ist eine Peptidsequenz mit den standardmäßigen Aminosäuresymbolen.

Die Funktion

- entfernt Leerzeichen in Eingabesequenzen;
- ignoriert nicht zugehörige Nukleotide außerhalb eines Leserahmens;
- gibt eine Leerausgabe zurück, wenn eine leere Nukleotidsequenz als Eingabe verwendet wird.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSNuc2Pep' - Beispiel" auf Seite 85
- „Benutzerdefinierte Funktion 'LSTransAllFrames'" auf Seite 86

Benutzerdefinierte Funktion 'LSNuc2Pep' - Beispiel

Angenommen, Sie wollen die Daten von Nukleotidsequenzen in eine Peptidsequenz übersetzen. Bei diesem Beispiel wird davon ausgegangen, dass das erste Codon am ersten Zeichen der Nukleotidsequenz beginnt.

Sie können die Funktion mit einer values-Anweisung aufrufen. Die einzige Eingabe ist eine Nukleotidsequenz, wie in folgendem Beispiel gezeigt:

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

Das Ergebnis der vorstehenden Anweisung ist eine Peptidsequenz mit den standardmäßigen Aminosäuresymbolen:

```
FFLLSSSSYFLCC*C
```

Wenn Sie die Übersetzung im Leserahmen +2 wünschen, verwenden Sie folgendes Beispiel:

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

Die ganze Zahl in der Anweisung gibt die Anfangsposition der Suche nach dem Codon an.

Es folgt ein Beispiel zur Verwendung dieser Funktion als Vergleichselement in einer Abfrage.

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

Das Ergebnis wird in Tabelle 37 gezeigt.

Tabelle 37. Ergebnisse mit Funktion 'LSNuc2Pep' als Vergleichselement

ID	PROTEINNAME	PEPTIDESEQ
1	proteinA	FSYCLPHRISYVAD

Das folgende Beispiel übersetzt eine Nukleotidsequenz anhand einer externen Übersetzungstabelle in eine Peptidsequenz. Der erste Parameter ist die Nukleotidsequenz, und der zweite Parameter ist der Pfad zur externen Übersetzungstabelle.

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

Das Ergebnis der vorstehenden Anweisung mit der angegebenen Übersetzungstabelle ist die folgende Zeichenfolge:

```
FSYCLPHRISYVAD
```

In folgendem Beispiel werden zwei der benutzerdefinierten Funktionen kombiniert, um den zusätzlichen Nutzen der Funktionen zu demonstrieren:

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

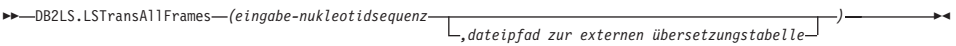
Hierbei ist zu beachten, dass das vorstehende Beispiel dieselben Ergebnisse zurückgibt wie die folgende Abfrage:

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

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSRevNuc' - Beispiel" auf Seite 81
- „Benutzerdefinierte Funktion 'LSTransAllFrames' - Beispiel" auf Seite 87
- „Benutzerdefinierte Funktion 'LSNuc2Pep'" auf Seite 84

Benutzerdefinierte Funktion 'LSTransAllFrames'



eingabe-nukleotidsequenz
Eine gültige Zeichenfolgedarstellung, die eine Nukleotidsequenz beschreibt. Die Eingabesequenz kann IUPAC-Ambiguitätscodes enthalten. Eine Zeichenfolgedarstellung muss über den Datentyp VARCHAR verfügen und darf nicht länger sein als 32672 Byte.

dateipfad zur externen Übersetzungstabelle
Wenn Sie eine angepasste Übersetzungstabelle verwenden, müssen Sie den entsprechenden Dateipfad angeben, damit die Tabelle gefunden wird. Der Zeichenfolgewert des Pfads darf 255 Zeichen nicht überschreiten.

Der Schemaname lautet DB2LS.

Verwenden Sie die benutzerdefinierte Funktion 'LSTransAllFrames', um aus einer angegebenen Nukleotidsequenz eine Gruppe von Peptidsequenzen zu generieren. Diese Peptidsequenzen stellen mögliche Übersetzungen der Eingabe-Nukleotidsequenz in jedem der 6 Rahmen dar. Diese Funktion ist nützlich, wenn die Eingabe Fehler enthält oder der Leserahmen unbekannt ist.

Das Ergebnis der Funktion wird in einer Tabelle mit zwei Spalten dargestellt. Die erste Spalte ist mit READFRAME markiert und stellt den Rahmen für die Übersetzung dar. Diese Spalte enthält eine ganze Zahl, die die Anfangsposition der Übersetzung darstellt. Eine negative ganze Zahl gibt eine Übersetzung des entgegengesetzten Strangs an. Die zweite Spalte namens PEPTID ist eine Zeichenfolge vom Datentyp VARCHAR mit einer Länge von maximal 10890 Byte. Die Zeichenfolge ist eine Darstellung der Peptidsequenz.

Die Funktion

- entfernt Leerzeichen in Eingabesequenzen;
- ignoriert nicht zugehörige Nukleotide außerhalb eines Leserahmens;
- gibt eine Leerausgabe zurück, wenn eine leere Nukleotidsequenz als Eingabe verwendet wird.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSTransAllFrames' - Beispiel" auf Seite 87
- „Benutzerdefinierte Funktion 'LSNuc2Pep'" auf Seite 84

Benutzerdefinierte Funktion 'LSTransAllFrames' - Beispiel

Angenommen, Sie wollen eine Nukleotidsequenz mit Hilfe der integrierten Übersetzungstabelle in allen sechs Leserahmen (Readframes) übersetzen. Das folgende Beispiel zeigt, wie dies funktioniert:

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

Die Abfrage gibt die Peptide in einer Tabelle zurück, wie aus folgendem Beispiel ersichtlich wird:

Tabelle 38. Ergebnis der Übersetzung einer Nukleotidsequenz

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

Das nächste Beispiel verwendet eine angepasste Übersetzungstabelle, um eine Nukleotidsequenz in allen sechs Leserahmen zu übersetzen.

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

Die Ergebnistabelle ist die gleiche wie im vorherigen Beispiel, da die Eingabesequenz sich nicht geändert hat und die Übersetzungstabelle der in die Funktion integrierten Tabelle entspricht.

In folgendem Beispiel werden zwei der benutzerdefinierten Funktionen kombiniert, um den zusätzlichen Nutzen der Funktionen zu demonstrieren:

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

Hierbei ist zu beachten, dass das vorstehende Beispiel dieselben Ergebnisse zurückgibt wie die folgende Abfrage:

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

Das folgende Beispiel wählt aus der von der Funktion 'LSTransAllFrames' generierten Ausgabe einen bestimmten Leserahmen aus.

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

Die Abfrage gibt folgendes Ergebnis zurück:

Tabelle 39. Verwendung der Leserahmenfunktion

READFRAME	PEPTIDE
–2	HQQHKKYDEEDNKK

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSNuc2Pep' - Beispiel" auf Seite 85
- „Benutzerdefinierte Funktion 'LSRevNuc' - Beispiel" auf Seite 81
- „Benutzerdefinierte Funktion 'LSTransAllFrames'" auf Seite 86

Format der Codon-Frequenztabelle

Eine Codon-Frequenztabelle zeigt die Frequenz, mit der die Aminosäuren in ein bestimmtes Codon zurück übersetzt werden. Die benutzerdefinierte Funktion 'LSPep2ProbNuc' verwendet die Codon-Frequenztabelle, um aus einer angegebenen Peptidsequenz eine Nukleotidsequenz zu ermitteln.

Die folgende Liste erläutert das Format der Datei mit der Codon-Frequenztabelle:

- Zwei benachbarte Punkte markieren den Anfang der Tabelle. Alle Textdaten, die vor diesen beiden Punkten stehen, sind Kommentare. Die beiden benachbarten Punkte sind auch dann erforderlich, wenn davor keine Kommentare stehen.
- Die Tabelle enthält die folgenden Spalten:
  1. Am-Acid: Ein aus drei Buchstaben bestehender Code für das Aminosäuresymbol.
  2. Codon: Das Codon für das betreffende Aminosäuresymbol.
  3. Number: Die Häufigkeit, mit der das betreffende Codon in den Genen vorkommt, aus denen die Tabelle aufgebaut ist.
  4. x/1000: Die erwartete Häufigkeit des Aminosäure-Codon-Paars pro 1000 Übersetzungen in Genen.
  5. Fraction: Der Bruchteil der Häufigkeit des Codons in seiner synonymen Codon-Familie.

Beispiele für Codon-Frequenztabellen stehen im Unterverzeichnis 'sqllib/samples/lifesci/ls_udfs' zur Verfügung.

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2ProbNuc'" auf Seite 53
- „Codon-Frequenztabelle - Beispiel" auf Seite 90

Codon-Frequenztabelle - Beispiel

Abb. 2 zeigt anhand eines Beispiels das Format einer Codon-Frequenztabelle.

```
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)
```

Abbildung 2. Beispiel einer Codon-Frequenztabelle

Zugehörige Referenzen:

- „Benutzerdefinierte Funktion 'LSPep2ProbNuc'" auf Seite 53
- „Format der Codon-Frequenztabelle" auf Seite 89

Format von Übersetzungstabellen

Dieser Abschnitt erläutert das Format von Übersetzungstabellen, die von den benutzerdefinierten Life Sciences-Funktionen 'LSPep2AmbNuc', 'LSTransAllFrames' und 'LSNuc2Pep' verwendet werden.

Die folgende Liste erläutert das Format der Datei mit der Codon-Frequenztabelle:

- Zwei benachbarte Punkte markieren den Anfang der Tabelle. Alle Textdaten, die vor diesen beiden Punkten stehen, sind Kommentare.
- Jede Zeile der Tabelle besteht aus einem einbuchstabigen Aminosäuresymbol, dem dreibuchstabigen Namen der Aminosäure, den eindeutigen Codons, einem Ausrufezeichen und den mehrdeutigen Codons. Die einzelnen Wörter in der Zeile werden jeweils durch ein Leerzeichen getrennt.
- Jedes Codon und jedes Aminosäuresymbol darf in der Datei jeweils nur einmal vorkommen.
- Stoppcodons werden in das Symbol '*' übersetzt.
- Codons, die aus Kleinbuchstaben bestehen, sind Startcodons.
- Alle anderen Codons bestehen aus Großbuchstaben.
- Codons, für die es keine Übersetzung in ein entsprechendes Aminosäuresymbol gibt, werden in das Symbol 'X' übersetzt.

Beispiele für Übersetzungstabellen stehen im Unterverzeichnis 'sqllib/samples/lifesci/ls_udfs' zur Verfügung.

Übersetzungstabelle - Beispiel

Abb. 3 zeigt das Format einer Beispielübersetzungstabelle.

```
                Beispielübersetzungstabelle

Symbol   3 Buchstaben 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
```

Abbildung 3. Beispielübersetzungstabelle

Eingabehilfen

Eingabehilfen unterstützen Benutzer mit körperlichen Behinderungen wie z. B. eingeschränkter Bewegungsfähigkeit oder Sehkraft beim erfolgreichen Einsatz von Softwareprodukten. Im Folgenden sind die wichtigsten Eingabehilfen aufgeführt, die unter DB2 Information Integrator Version 8 zur Verfügung stehen:

- Sie können alle Funktionen über die Tastatur anstatt mit der Maus ausführen.
- Sie können Farbe und Größe der verwendeten Schriftarten anpassen.
- Vom System wird die Ausgabe visueller und akustischer Signale unterstützt.
- DB2 unterstützt Anwendungen mit Eingabehilfen, die mit der Java™ Accessibility API arbeiten.
- Zum Lieferumfang von DB2 gehört Dokumentationsmaterial in einem behindertengerechten Format.

Tastatureingabe und Navigation

Die verfügbaren DB2-Datenbanktools (z. B. die Steuerzentrale, die Data Warehouse-Zentrale und die Replikationszentrale) können unter ausschließlicher Benutzung der Tastatur verwendet werden. Mit entsprechenden Tasten oder Tastenkombinationen können die meisten Operationen ausgeführt werden, die auch über die Maus verfügbar sind.

Auf UNIX-Systemen ist der Tastatureingabebereich hervorgehoben. Auf diese Weise wird der aktive Bereich des Fensters gekennzeichnet, in dem die Tastatureingabe erfolgt.

Eingabehilfen für Bildschirme

Die DB2-Datenbanktools verfügen über Funktionen zur Verbesserung der Benutzerschnittstelle und deren Einsatzmöglichkeiten für sehbehinderte Benutzer. Diese Eingabehilfen umfassen die Unterstützung individuell anpassbarer Schriftarteigenschaften.

Schriftarteinstellungen

Die DB2-Datenbanktools ermöglichen Ihnen die Auswahl der Farbe, Größe und der Schriftart für den Text in Menüs und Fenstern. Die entsprechenden Einstellungen können im Notizbuch 'Tools - Einstellungen' definiert werden.

Unabhängigkeit von Farben

Zur Verwendung der Funktionen des vorliegenden Produkts ist es nicht erforderlich, zwischen unterschiedlichen Farben differenzieren zu können.

Alternative Signale

Im Notizbuch 'Tools - Einstellungen' können Sie angeben, ob akustische oder visuelle Signale ausgegeben werden sollen.

Kompatibilität mit Unterstützungseinrichtungen

Die grafische Oberfläche von DB2 Information Integrator unterstützt die Java Accessibility API. Diese Anwendungsprogrammierschnittstelle ermöglicht den Einsatz von Bildschirmsprachausgabe-Einheiten und anderen Unterstützungseinrichtungen für Personen mit Behinderungen.

Dokumentation im behindertengerechten Format

Die Dokumentation für die DB2-Produktfamilie steht im HTML-Format zur Verfügung. Die Dokumentation kann mit den Anzeigeeinstellungen aufgerufen werden, die Sie in Ihrem Browser definiert haben. Darüber hinaus ist der Einsatz von Bildschirmsprachausgabe-Einheiten und anderen Unterstützungseinrichtungen möglich.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit solche Verbindungen nicht ausdrücklich von IBM bestätigt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen sind schriftlich an

IBM Europe
Director of Licensing
92066 Paris La Defense Cedex
France

zu richten. Anfragen an obige Adresse müssen auf Englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

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

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden, Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHT-LIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme geschrieben werden. Die in diesem Handbuch aufgeführten Beispiele sollen lediglich der Veranschaulichung und zu keinem anderen Zweck dienen. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation:

IBM
AIX
DB2
Domino
Informix
Lotus
Lotus Notes
QuickPlace
WebSphere

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen:

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken der Microsoft Corporation.

UNIX ist in gewissen Ländern eine eingetragene Marke von The Open Group.

Java und alle auf Java basierenden Marken und Logos sind in gewissen Ländern Marken oder eingetragene Marken der Sun Microsystems, Inc.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken oder Dienstleistungsmarken anderer Unternehmen sein.

Index

Kontaktaufnahme mit IBM

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0190 7 72243 erreichen Sie die DB2 Helpline, wo Sie Antworten zu DB2-spezifischen Problemen erhalten.

Informationen zur nächsten IBM Niederlassung in Ihrem Land oder Ihrer Region finden Sie im IBM Verzeichnis für weltweite Kontakte, das Sie im Web unter www.ibm.com/planetwide abrufen können.

Produktinformationen

Informationen zu DB2 Information Integrator erhalten Sie telefonisch oder im World Wide Web.

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180 5 5090 können Sie Handbücher telefonisch bestellen.

Rufen Sie im Web die Site www.ibm.com/software/data/integration auf. Diese Site enthält die neuesten Informationen zur technischen Bibliothek, zum Bestellen von Büchern, zu Client-Downloads, Newsgroups, FixPaks, Neuerungen und Links auf verfügbare Webressourcen.

Informationen zur nächsten IBM Niederlassung in Ihrem Land oder Ihrer Region finden Sie im IBM Verzeichnis für weltweite Kontakte, das Sie im Web unter www.ibm.com/planetwide abrufen können.

Kommentare zur Dokumentation

Ihr Feedback unterstützt IBM bei der Bereitstellung qualitativ hochwertiger Informationsmaterialen. Bitte senden Sie uns Ihre Kommentare zum vorliegenden Handbuch oder zu anderen DB2 Information Integrator-Dokumentationen. Zur Abgabe von Kommentaren können Sie folgendermaßen vorgehen:

- Verwenden Sie für Ihren Kommentar das Onlineformular für Leserkommentare, das unter www.ibm.com/software/data/rcf bereitgestellt wird.

- Senden Sie Ihre Kommentare in einer E-Mail an die Adresse comments@us.ibm.com. Bitte geben Sie unbedingt den Namen des Produkts, seine Versionsnummer sowie den Titel und die IBM Formnummer (sofern vorhanden) der Veröffentlichung an, auf die sich Ihr Kommentar bezieht. Geben Sie bei Kommentaren zu einer spezifischen Textstelle bitte auch die Position dieser Textstelle (z. B. Abschnittsüberschrift, Abbildungs- oder Seitennummer) innerhalb der Veröffentlichung an.