IBM DB2 10.1
for Linux, UNIX, and Windows

IBM データ・サーバー用の RDF アプリケーション開発

2013 年 1 月更新版

SA88-5083-00
(英文原典：SC27-4462-00)

IBM DB2 10.1
for Linux, UNIX, and Windows

IBM データ・サーバー用の RDF アプリケーション開発

2013 年 1 月更新版

SA88-5083-00
(英文原典：SC27-4462-00)

ご注意

本書および本書で紹介する製品をご使用になる前に、 83 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、IBM Publications Center (http://www.ibm.com/shop/publications/order) をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、IBM Directory of Worldwide Contacts (http://www.ibm.com/planetwide/) をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

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

原典： SC27-4462-00
IBM DB2 10.1
for Linux, UNIX, and Windows
Developing RDF Applications for IBM
Data Servers
Updated January, 2013

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.12

目次

第 1 部 IBM データ・サーバー用の RDF アプリケーション開発

Resource Description Framework (RDF) は一連の W3 仕様であり、情報のモデリングのための標準データ交換フレームワークとして使用できます。 アプリケーションは、RDF データを DB2® Enterprise Server Edition for Linux, UNIX, and Windows (DB2 Enterprise Server Edition) データベースに保管し、照会することができます。

RDF はデータ間の関係をトリプル (例えば主語 - 述語 - 目的語という書式の式など) で作成するために Uniform Resource Identifier (URI) を使用しています。 この単純なモデルを使用すると、構造化データおよび半構造化データを、さまざまなアプリケーション同士の間でリンク、公開、共有することができます。

DB2 データベース・サーバーの RDF ストアは、RDF データ・セットを保管する、データベース・スキーマの中のユーザー表の集合です。 このような表集合には、それぞれ固有なストア名が関連付けられています。 それぞれの RDF ストアには、ストア用のメタデータを収めた表があります。 この表の名前は、ストア名と同じです。

RDF ユーティリティー・コマンドまたは Java™ API を使用して、これらのユーザー表にデータをロードすることができます。 これらの表集合に対して適切な読み取り権限と書き込み権限が必要です。サポートされる Java API は、JENA フレームワーク API です。 DB2 ソフトウェアのバージョン 9.7 以降では、DB2 RDF ユーティリティー・コマンドまたは API がサポートされています。

RDF アプリケーションは、DB2 データベースでデータを検索する際に SPARQL 照会言語を使用します。

RDF は、パーティション・データベース環境ではサポートされていません。

第 1 章 RDF 参照および関連リソース

IBM データ・サーバーにアクセスする RDF アプリケーションの開発を支援するために、多くのリソースが用意されています。

表 1. RDF 参照および関連リソース

RDF リソース	**参照リンク**
RDF Primer	http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
SPARQL Query Language	http://www.w3.org/TR/rdf-sparql-query/
JENA グラフおよびモデル API	http://jena.sourceforge.net/tutorial/RDF_API/
IBM RDF Javadoc	../javadoc/index.html
RDF application development tutorial Part 1: RDF store creation and maintenance	http://www.ibm.com/developerworks/data/tutorials/dm-1205rdfdb210/index.html

第 2 章 RDF ストア表

RDF ストアは、複数の表から構成されます。 それらの表には、RDF ストアに関するメタデータまたはユーザー・データが含まれます。

メタデータ表

次の表に、RDF ストアに関するメタデータが格納されます。

- RDF ストアと同じ名前を持つ 1 つのメタデータ表。
- システム述語メタデータ表。SPARQL 照会の結果をさらにフィルタリングするために使用できる RDF 述語についての情報が保管されます。
- 基本統計表には、RDF ストア内のデータに関する統計が保管されます。
- Top K 統計表には、ストア内で最も選択的でない RDF データに関する情報が保管されます。

データ表

以下の表には、RDF データの値が特定の文字長を超えない場合、そのデータが保管されます。

- 直接 1 次表は RDF トリプルと関連グラフを保管し、主語によって索引付けられます。 ある主語に対する述語と目的語は、この表の中でペアになっている列に保管されます。 特定の述語は、この表の 3 つの列のうちどれに入っていても構いません。 この述語に対する目的語は、述語 - 目的語のペアの中で対応している目的語の列に保管されます。
- 直接 2 次表には、RDF グラフの中で主語と述語を共有する RDF トリプルが保管されます。 このようなトリプルは、直接 1 次表にプレースホルダー ID しか持ちません。
- リバース 1 次表は RDF トリプルと関連グラフを保管し、目的語によって索引付けられます。 ある目的語に対する述語と主語は、この表の中でペアになっている列に保管されます。 特定の述語は、この表の 3 つの列のうちどれに入っていても構いません。また、その述語に対する主語は、そのペアの中で対応している主語列に入れることができます。
- リバース 2 次表には、RDF グラフの中で目的語と述語を共有する RDF トリプルが保管されます。 このようなトリプルにはプレースホルダー ID があり、その ID はリバース 1 次表に格納されます。
- データ・タイプ表には、SPARQL データ・タイプへの内部整数値のマッピング、ユーザー定義データ・タイプ、および言語タグが保管されます。

RDF の主語、述語、目的語、グラフのいずれかの値が特定の文字長を超えた場合、前述の 5 つの表には、プレースホルダー ID しか保管されません。 実際の値は、LONG ストリング表に保管されます。

第 3 章 RDF 管理用データベース・オブジェクト

RDF には、RDF ストアを管理するための関数やスケジューラー・タスク・オブジェクトがあります。

管理用データベース・オブジェクト

DB2 の RDF には、以下のような管理データベース・オブジェクトがあります。

- <*store_name*>_RDF_REGEX という名前の Java ベースの外部 UDF は、SPARQL の regex 演算子をサポートしています。 この UDF を使用するには、適切なアクセス権が付与されている必要があります。

  DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、正規表現の機能で <*store_name*>_RDF_REGEX UDF がサポートされなくなりました。正規表現では、代わりに pureXML® `fn:matches()` 関数を使用します。
- <*store_name*>_T3_STATS という名前の SQL ストアード・プロシージャーは、RDF ストアの基本統計および topK 統計を収集します。
- <*SCHEMANAME*>_<*STORENAME*>_Scheduler という名前の管理用スケジューラー・タスクは、ストアの基本統計および topK 統計の更新インターバルをスケジュールします。

第 4 章 RDF ストアのアクセス制御

DB2 RDF ストアでは、2 つのタイプのアクセス制御が使用可能です。

粒度の粗いアクセス制御

DB2 データベースの表レベルの許可を使用して、RDF ストア全体へのアクセスを制御することができます。

RDF グラフ・レベルのアクセス制御

RDF グラフ・レベルのアクセス制御は、RDF グラフのレベルでのより細かいアクセス制御を提供します。 RDF データ・セット全体ではなく、RDF ストア内でユーザーがアクセスできる RDF グラフを選択的に制御できます。

RDF グラフ・レベルのアクセス制御では、グラフ内の RDF トリプルを使用して、ユーザーに RDF グラフに対するアクセス権限があるかどうかを判別します。 RDF ストアの作成時に、ユーザーは、RDF グラフへのアクセスを制御するためにどの RDF 述語を使用するかを指定する必要があります。

ランタイムでのアクセス制御の実施 (SPARQL 照会を使用) は、DB2 エンジンに委任することができます。あるいは、DB2 RDF ストア SQL 生成プログラムによって生成される SQL の中で使用することも可能です。

DB2 エンジンによるアクセス制御の実施を選択した場合、DB2 ソフトウェアのきめ細かいアクセス制御フィーチャーを使用して、アクセス制御規則を指定する必要があります。

RDF ストア SQL 生成プログラムによるアクセス制御の実施を選択した場合は、アプリケーションはさらに `QueryExecution` コンテキストで適用される制約をパスしなければなりません。この場合には、限られた演算子とオペランドのみがサポートされます。

- グラフ・レベルのアクセス制御サポートを使用する RDF ストアの作成
- RDF ストア SQL 生成プログラムを介したグラフ・レベルのアクセス制御の実施
- DB2 エンジンを介したグラフ・レベルのアクセス制御の実施

第 5 章 デフォルト RDF ストアおよび最適化された RDF ストア

DB2 データベースでは 2 種類の RDF ストアが使用されています。 1 つはデフォルト RDF ストア、もう 1 つは最適化された RDF ストアと呼ばれます。

デフォルト RDF ストア

このベース・スキーマは、保管される RDF データについての情報がない場合や、適切なサンプルが入手できない場合に使用されます。 デフォルト RDF ストアは、直接 1 次表およびリバース 1 次表において、デフォルトの数の列を使用します。新しい RDF データ・セット (それについての情報がない) を使用して始める場合には、このデフォルト・ストアを使用します。 デフォルト・ストアでは、述語と目的語が直接 1 次表およびリバース 1 次表のどの列に入るかを、ハッシュを使用して決定します。

DB2 ソフトウェアによる述語の共存の計算の基にできる RDF データ・セットの既存のサンプル・データがない場合、デフォルト RDF ストアを作成します。

最適化された RDF ストア

RDF データ・セットの十分な量の代表データが既に使用可能であれば、直接 1 次表およびリバース 1 次表に対してより最適化されたスキーマを作成できます。この最適化されたスキーマは、複数の RDF 述語が互いに相関するということを活用することにより、実現されます。 例えば、年齢と社会保障番号は「個人」の述語として共存し、本社と収益は「企業」の述語として共存しますが、年齢と収益はどのエンティティーにおいても共に出現することはありません。

DB2 が述語の相関を計算して、述語を列にインテリジェントに割り当てるための基になる既存のデータまたはサンプル・データが RDF データ・セットにある場合は、最適化された RDF ストアを作成してください。

最適化された RDF ストアの利点

述語の相関は、デフォルト・ストアで使用されるハッシュのランダム性を大幅に減らす (多くの場合は完全になくす) ために使用されます。 そのため、デフォルト・ストアでは、述語の相関についてのナレッジの不足によって述語の衝突が発生することがあり、このために表で使用される行の数が実際に必要な数より多くなる場合があります。 余分な行により、表間の結合の効率が本来のものより低下する場合があります。

大抵、特定の述部は単一の列に限定できるため、述部の索引付けはより簡単に行えます。 また、共存しない複数の述語を単一の列に割り当てることができるので、単一の DB2 索引で複数の述語を索引付けできます。

第 6 章 RDF ストアの中心となるビュー

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、Resource Description Framework (RDF) は、特定のデータベースに存在するすべての RDF ストアを 1 つの表内にリストするようになりました。すべての RDF ストアを表示するには、SYSTOOLS.RDFSTORES 表を照会します。

SYSTOOLS.RDFSTORES 表は、**createrdfstore** コマンドまたは **createrdfsoreandloader** コマンドと API がデータベースに対して最初に発行されたときに作成されます。

表 2. SYSTOOLS.RDFSTORES 表スキーマ

列名	データ・タイプ	NULL 可能	説明
STORENAME	VARCHAR(256)	いいえ	RDF ストアの名前
SCHEMANAME	VARCHAR(256)	いいえ	RDF ストアのスキーマの名前
STORETABLE	VARCHAR(256)	いいえ	RDF ストアのメタデータ表の名前
主キー		いいえ	主キー

データベース内のすべての RDF ストアと対応するスキーマ名をリストするには、次の照会を発行します。

```
SELECT storeName, schemaName FROM SYSTOOLS.RDFSTORES
```

以下の出力例が返されます。

```
STORENAME       SCHEMANAME
--------------------------
STAFFING   DB2ADMIN
SAMPLSTORE      DB2ADMIN

2 record(s) selected.
```

第 7 章 RDF 環境のセットアップ

DB2 RDF コマンドおよび API を使用するように、ご使用の環境をセットアップします。

コマンド行ユーティリティーを使用した RDF コマンドの発行

DB2 RDF コマンド行ユーティリティーは、*<install_path>*/sqllib/rdf/bin ディレクトリーにあります。 DB2 コマンド・プロンプトを使用してこのディレクトリーからこれらのユーティリティーを起動します。

DB2 データベース・サーバーをインストールした後で、DB2 RDF コマンド行ユーティリティーを使用するには、次のタスクを実行します。

1. http://sourceforge.net/projects/jena/files/ARQ/ARQ-2.8.5/「http://sourceforge.net/projects/jena/files/ARQ/ARQ-2.8.5/」から ARQ パッケージのバージョン 2.8.5 をダウンロードします。

   ARQ パッケージの lib フォルダーから *<install_path>*/SQLLIB/rdf/lib ディレクトリーに JAR ファイルをコピーします。

   注: 「xxx-tests.jar」、「xxx-sources.jar」、「xxx-test-sources.jar」の JAR ファイルはコピーをスキップできます。

   DB2 バージョン 10.1 フィックスパック 2 以降では、http://archive.apache.org/dist/jena/binaries/「http://www.apache.org/dist/jena/binaries/」の Apache JENA バージョン 2.7.3 パッケージを使用します。

   Apache JENA パッケージの lib フォルダーから *<install_path>*/SQLLIB/rdf/lib ディレクトリーに JAR ファイルを保存します。
2. Apache Commons プロジェクトから Commons-logging-1-0-3.jar をダウンロードします。 *<install_path>*/SQLLIB/rdf/lib ディレクトリーにこの JAR を配置します。
3. コマンド・プロンプトを開き、*<install_path>*/SQLLIB/rdf/bin ディレクトリーに移動します。

   ```
   cd "<install_path>/SQLLIB/rdf/bin"
   ```
4. <install_path>/SQLLIB/java ディレクトリーの DB2 JCC ドライバーである db2jcc4.jar を、次のようにクラスパス環境変数に追加します。

   ```
   set classpath=<install_path>\SQLLIB\java\db2jcc4.jar;%classpath%
   ```

これで、DB2 RDF コマンド行ユーティリティーをこのコマンド・プロンプトで実行できるようになりました。

アプリケーション開発環境での DB2 RDF

DB2 RDF JAR ファイルと次の JAR ファイルをアプリケーション・クラスパスに追加する必要があります。

- JENA に依存する JAR ファイル

- Commons-logging-1-0-3.jar ファイル
- DB2 JCC ドライバー (db2jcc4.jar)

これらの JAR ファイルは、<*install_path*>/sqllib/rdf/lib ディレクトリーにあります。ここには、以下の JAR ファイルが含まれています。

- rdfstore.jar
- antlr-3.3-java.jar
- wala.jar

第 8 章 RDF を DB2 バージョン 9.7 と共に使用する

DB2 バージョン 10.1 クライアントをインストールし、それを DB2 バージョン 9.7 データベース・サーバーと共に使用することができます。

RDF サポートに必要な、外部 Java ライブラリーを登録してください。 DB2 バージョン 10.1 クライアントにある「rdf/bin/registerrdfudf」スクリプトを実行することで、登録できます。このスクリプトは、RDF ストアが作成される DB2 バージョン 9.7 データベースごとに実行する必要があります。例えば、以下のコマンドを発行します。

```
registerrdfudf <dbname> <username>
```

ここで、<*dbname*>は、ローカル DB2 クライアント上のカタログ・データベースです。

第 9 章 SPARQL バージョン 1.1 の HTTP 経由のグラフ・ストア・プロトコルと SPARQL の設定

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、DB2 RDF は SPARQL バージョン 1.1 グラフ・ストアの HTTP プロトコルをサポートします。このプロトコルは、Apache JENA Fuseki バージョン 0.2.4 を必要とします。 SPARQL REST API を使用するには、Fuseki 環境を設定する必要があります。

始める前に

Fuseki 環境をセットアップするには、以下のようにします。

1. jena-fuseki-0.2.4-distribution.zip ファイルを http://www.apache.org/dist/jena/binaries/「http://www.apache.org/dist/jena/binaries/」からダウンロードします。
2. ローカル・システム上にファイルを解凍します。

手順

1. コマンド・プロンプト・ウィンドウを開き、*<Fuseki install dir>*/jena-fuseki-0.2.4 ディレクトリーに移動します。
2. config.ttl ファイルを開き、接頭部として db2rdf を追加します。

```
@prefix :        <#> .
@prefix fuseki:  <http://jena.apache.org/fuseki#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tdb:     <http://jena.hpl.hp.com/2008/tdb#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .

@prefix db2rdf:  <http://rdfstore.ibm.com/IM/fuseki/configuration#>
```

3. DB2 RDF サービスを config.ttl ファイルに追加します。 このサービスを、他のすべてのサービスが登録されているファイルのセクションに追加します。

```
fuseki:services (
     <#service1>
     <#service2>
      <#serviceDB2RDF_staffing>
   ) .
```

   サービスは複数登録できます。各サービスは、それぞれ異なる DB2 RDF データ・セットを照会します。

4. 次の構成を config.ttl ファイルに追加して、RDF ネーム・スペースを初期化します。 この構成は、DB2Dataset を作成するアセンブラーを登録します。この構成は、DB2QueryEngine エンジンおよび DB2UpdateEngine エンジンも登録します。

```
# DB2
[] ja:loadClass "com.ibm.rdf.store.jena.DB2" .
db2rdf:DB2Dataset  rdfs:subClassOf  ja:RDFDataset .
```

5. DB2 RDF サービスに関するこれらすべての詳細を、config.ttl ファイルの最後に追加します。

```
# Service DB2 Staffing store
<#serviceDB2RDF_staffing>
rdf:type fuseki:Service ;
rdfs:label "SPARQL against DB2 RDF store" ;
fuseki:name "staffing" ;
fuseki:serviceQuery "sparql" ;
fuseki:serviceQuery "query" ;
fuseki:serviceUpdate "update" ;
fuseki:serviceUpload "upload" ;
fuseki:serviceReadWriteGraphStore "data" ;
fuseki:serviceReadGraphStore "get" ;
fuseki:serviceReadGraphStore "" ;
fuseki:dataset <#db2_dataset_read> ;
.

<#db2_dataset_read> rdf:type db2rdf:DB2Dataset .

# specify the RDF store/dataset and schema
db2rdf:store "staffing" ;
db2rdf:schema "db2admin" ;

# Database details. Specify either a jdbcConnectString
# with username and password or specify a jndiDataSource
db2rdf:jdbcConnectString "jdbc:db2://localhost:50000/RDFSAMPL" ;
db2rdf:user "db2admin" ;
db2rdf:password "db2admin" .

#db2rdf:jndiDataSource "jdbc/DB2RDFDS" .
```

6. 次のコマンドをコマンド行から発行します。

```
SET CLASSPATH=./fuseki-server.jar;<DB2_FOLDER>/rdf/lib/rdfstore.jar;
<DB2_FOLDER>/rdf/lib/wala.jar;<DB2_FOLDER>/rdf/lib/antlr-3.3-java.jar;
<DB2_FOLDER>/rdf/lib/commons-logging-1-0-3.jar;<DB2_FOLDER>/java/db2jcc4.jar;
%CLASSPATH%;

java org.apache.jena.fuseki.FusekiCmd --config config.ttl
```

タスクの結果

1. ブラウザーを開始して、localhost:3030 という URL をロードします。ブラウザー・ウィンドウに Fuseki ページがロードされます。
2. ブラウザー・ウィンドウの Control Panel ハイパーリンクをクリックして、ドロップダウン・リストから Dataset を選択します。ドロップダウン・リストには、config.ttl ファイルにリストされているすべてのデータ・セットが含まれます。構成したデータ・セットを選択します。
3. GUI からオプションを使用して、SPARQL の照会を発行します。最初のセクションは、SPARQL 照会言語を使用したデータ・セットの照会用です。 2 番目のセクションは、SPARQL の更新を使用したデータ・セットの変更用です。 3 番目のセクションは、データ・セットのグラフへのデータのロード用です。

第 10 章 RDF ストアの作成

アプリケーション開発要件に基づいて、デフォルトの RDF ストア、もしくは最適化された RDF ストアを作成します。 RDF ストアを最初に作成して、後からデータをロードすることもできます。

デフォルト RDF ストアの作成

既存の RDF データがない場合でも、RDF ストアを作成することができます。これはデフォルト RDF ストアとも呼ばれます。

始める前に

以下の前提条件が必要です。

- データベースのページ・サイズは最小でも 32 KB にしてください。
- **LOGFILSIZ** データベース構成パラメーターは、20000 以上に設定してください。

```
db2 UPDATE DATABASE CONFIGURATION FOR <DB_NAME>
USING LOGFILSIZ 20000
```

- SYSTOOLSPACE 表スペースが確実に存在するようにしてください。

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 4
```

- 許可 ID が以下の特権を持っていることを確認します。
  - 選択したデータベース・スキーマに対する CREATETAB 権限
  - CREATE EXTERNAL ROUTINES 権限
  - 表 SYSTOOLS.ADMINTASKSTATUS に対する更新特権

さらに、以下の機能もセットアップしてください。

- 管理用タスク・スケジューラーを YES に設定します。

```
db2set DB2_ATS_ENABLE=YES
```

- **AUTORUNSTATS** データベース構成パラメーターを ON に設定します。

```
db2 UPDATE DB CONFIG USING AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
```

- バッファー・プールを AUTOMATIC に設定し、適切な初期サイズを割り当てます。

```
db2 alter bufferpool IBMDEFAULTBP IMMEDIATE SIZE 15000 AUTOMATIC
```

手順

以下の説明に従ってデフォルト RDF ストアを作成します。

1. RDF ストアを構成するためのデータベース表および表スペースの名前をコントロールします。 `objectNames.props` プロパティー・ファイルを作成して、RDF ストアのデータ表およびその表に割り当てるための対応する名前と表スペースのリストを含めます。

   次の例に、サンプル `objectNames.props` プロパティー・ファイルの内容を示します。

```
direct_primary_hash=<user_table_name>, <tablespace>
direct_secondary=<user_table_name>, <tablespace>
reverse_primary_hash=<user_table_name>, <tablespace>
reverse_secondary=<user_table_name>, <tablespace>
long_strings=<user_table_name>, <tablespace>
basic_stats=<user_table_name>, <tablespace>
topk_stats=<user_table_name>, <tablespace>
system_predicate=<user_table_name>, <tablespace>
data_type=<user_table_name>, <tablespace>
```

注: 表名の設定に objectNames.props ファイルを使用することはオプションです。 しかし、このプロパティー・ファイルを使用しない場合、代わりにシステム生成されたファイル名が使用されます。

2. **createrdfstore** コマンドを発行します。 RDF ストアの作成に使用するデータベース・インスタンスとスキーマを決定します。 また、ストアの論理名も決定します。この名前は、そのデータベース・スキーマ内のすべての RDF ストアの中で固有でなければなりません。

   例えば、ホスト localhost のポート 60000 でデータベース DB1、スキーマ db2admin に rdfStore1 というストアを作成するには、次のコマンドを使用します。

   ```
   createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
   -user db2admin -password XXX -schema db2admin
   ```

最適化された RDF ストアの作成

API の使用による最適化された RDF ストアの作成

最適化されたストアを作成するには、代表トリプル・データの既存のセットが必要です。述語オカレンスを適切に計算するために、このデータが必要です。

トリプル・データの代表サンプル・セットがない場合は、**createrdfstore** コマンドを使用して、デフォルトのストアを作成します。述語オカレンスが計算できるだけの十分な量の代表トリプル・データが収集されるまで、デフォルト RDF ストアを使用します。

十分な量のデータを収集した後、Java クラス StoreManager の generatePredicateMappings メソッドを使用して、デフォルト RDF ストア内のトリプルの述語の相関を計算します。この API の PredicateMappings 出力を保存します。

実稼働環境のために RDF ストアを作成する場合は、Java クラス StoreManager の createStoreWithPredicateMappings メソッドを使用して、最適化されたクリーンな RDF ストアを新たに作成します。前に取得した PredicateMappings 出力を用意します。

例

次の Java プログラムは、最適化された RDF ストアを作成する方法を示しています。

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.ibm.rdf.store.StoreManager;


/**
 * This program demonstrates how to create an optimized RDF store
 * by feeding in the predicate correlation information from an existing
 * store. Creating optimized RDF stores for production
 * environments can also be done post system QA cycle on a
 * default store.
 */
public class CreateOptimizedStore {

 public static void main(String[] args) throws SQLException,
IOException {

  String currentStoreName = "sample";
  String currentStoreSchema = "db2admin";
  Connection currentStoreConn = null;

  /*
   * Connect to the "currentStore" and generate the predicate
     * correlation for the triples in it.
   */
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   currentStoreConn = DriverManager.getConnection(
    "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
   currentStoreConn.setAutoCommit(false);
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /* Specify the file on disk where the predicate
     * correlation will be stored.
     */
  String file = "/predicateMappings.nq";
  BufferedOutputStream predicateMappings = new
BufferedOutputStream(new FileOutputStream(file));

  StoreManager.generatePredicateMappings(currentStoreConn,
    currentStoreSchema, currentStoreName,
predicateMappings);

  predicateMappings.close();

  /**
   * Create an optimized RDF store using the previously
     * generated predicate correlation information.
   */
  String newOptimizedStoreName = "production";
  String newStoreSchema = "db2admin";
  Connection newOptimizedStoreConn =
DriverManager.getConnection(
    "jdbc:db2://localhost:50000/dbrdf",
"db2admin","db2admin");
  BufferedInputStream inputPredicateMappings =
 new BufferedInputStream(
     new FileInputStream(file));

 StoreManager.createStoreWithPredicateMappings(
```

```
newOptimizedStoreConn, newStoreSchema,
newOptimizedStoreName, null, inputPredicateMappings);

   }
}
```

コマンドの使用による最適化された RDF ストアの作成

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、RDF コマンドを使用して、デフォルトの RDF ストアから最適化されたストアを作成できます。

手順

コマンド・プロンプトから最適化された RDF ストアを作成するには、次のようにします。

1. **createrdfstore** コマンドを使用して、デフォルト・ストアを作成します。

   ```
   createrdfstore rdfStore1 -db RDFSAMPL
   -user db2admin -password XXX
   ```

2. SPARQL UPDATE または JENA API を使用して、データをこのストアに追加します。このデフォルト・ストアを使用して、述語オカレンスの計算に使用できるトリプル・データのセットを収集します。
3. **genpredicatemappings** コマンドを使用して、述語マッピングを生成します。

   ```
   genPredicateMappings MyStore -db RDFSAMPL -user db2admin
   -password db2admin "C:\MyStore_predicate_mappings.txt"
   ```

4. **createrdfstore** コマンドを使用して **-predicatemappings** パラメーターを渡すことで、最適化されたストアを作成します。

   前のステップで生成された述語を、**-predicatemappings** パラメーターの入力として使用します。

   ```
   createrdfstore MyOptimizedStore -db RDFSAMPL
   -user db2admin -password XXX
   -predicatemappings "C:\MyStore_predicate_mappings.txt"
   ```

タスクの結果

最適化されたストアが作成されました。

既存のデータを使用した、最適化された RDF ストアの作成

既存の RDF データを使用して、RDF ストアを作成できます。

始める前に

以下の前提条件が必要です。

- データベースのページ・サイズは最小でも 32 KB にしてください。
- **LOGFILSIZ** データベース構成パラメーターは、20000 以上に設定してください。

  ```
  db2 UPDATE DATABASE CONFIGURATION FOR <DB_NAME>
  USING LOGFILSIZ 20000
  ```

- SYSTOOLSPACE 表スペースが確実に存在するようにしてください。

  ```
  CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
  MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 4
  ```

- 許可 ID が以下の特権を持っていることを確認します。
  - 選択したデータベース・スキーマに対する CREATETAB 権限
  - CREATE EXTERNAL ROUTINES 権限
  - 表 SYSTOOLS.ADMINTASKSTATUS に対する更新特権

さらに、以下の機能もセットアップしてください。

- 管理用タスク・スケジューラーを YES に設定します。

  ```
  db2set DB2_ATS_ENABLE=YES
  ```

- **AUTORUNSTATS** データベース構成パラメーターを ON に設定します。

  ```
  db2 UPDATE DB CONFIG USING AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
  ```

- バッファー・プールを AUTOMATIC に設定し、適切な初期サイズを割り当てます。

  ```
  db2 alter bufferpool IBMDEFAULTBP IMMEDIATE SIZE 15000 AUTOMATIC
  ```

Windows プラットフォームでは、**createrdfStoreAndLoader** コマンドは CygWin アプリケーションを必要とします。このコマンドに必要な Gawk ユーティリティーは、バージョン 4.0 以降です。このコマンドに必要な Core ユーティリティーは、バージョン 8.14 以降です。 CygWin をインストールした後、*<CgyWin_install_directory>*/bin を PATH 環境変数に追加します。パス上に CygWin がないと、コマンドを実行したときに次のエラー・メッセージが表示されます。

```
'Cannot run program "sh": CreateProcess error=2, The system cannot find the
specified file.'
```

Windows プラットフォームでは、CygWin コマンド・プロンプトまたはデフォルトのコマンド・プロンプトのいずれかから **createrdfStoreAndLoader** コマンドを起動できます。 CygWin コマンド・プロンプトを使用するとき、どのファイル・パス (-rdfdata、-storeloadfile、-storeschemafile、-objectnames) にも「cygdrive」接頭部を含めることができません。代わりに、「C:¥....」などの標準の Windows のパスを使用します。

指定されたフォルダーまたはファイル名のパスにスペースが含まれている場合は、ストリング全体を二重引用符で囲む必要があります。

手順

1. 既存のデータを n-Quad ファイルにエクスポートします。
2. RDF ストアを構成するためのデータベース表および表スペースの名前をコントロールします。 objectNames.props プロパティー・ファイルを作成して、RDF ストアのデータ表およびその表に割り当てるための対応する名前と表スペースのリストを含めます。

   次の例に、サンプル objectNames.props ファイルの内容を示します。

   ```
   direct_primary_hash=<user_table_name>,<tablespace>
   direct_secondary=<user_table_name>,<tablespace>
   reverse_primary_hash=<user_table_name>,<tablespace>
   reverse_secondary=<user_table_name>,<tablespace>
   long_strings=<user_table_name>,<tablespace>
   ```

```
basic_stats=<user_table_name>,<tablespace>
topk_stats=<user_table_name>,<tablespace>
system_predicate=<user_table_name>,<tablespace>
data_type=<user_table_name>,<tablespace>
```

注: 表名の設定に objectNames.props ファイルを使用することはオプションです。 しかし、このプロパティー・ファイルを使用しない場合、代わりにシステム生成されたファイル名が使用されます。

3. **createrdfstoreandloader** コマンドを発行します。

   Windows では、このコマンドは CygWin を必要とします。 また、このコマンドに必要な Gawk ユーティリティーはバージョン 4.0 で、コア・ユーティリティーはバージョン 8.14 以降です。

   RDF ストアの作成に使用するデータベース・インスタンスとスキーマを決定します。 また、RDF ストアの論理名も決定します。 この名前は、そのデータベース・スキーマ内のすべての RDF ストアの中で固有でなければなりません。

   コマンドに、**objectnames** パラメーターを指定します。 **objectNames** パラメーターを指定しない場合、RDF ストアの表には、代わりにシステム生成されたオブジェクト名が使用されます。 ファイル・システムに必ず出力ディレクトリーができているようにしてください。

   このコマンドは、既存のデータを使用して、最適化された RDF ストアを作成し、新たに作成された RDF ストアにデータをロードするために必要な DB2 データベース・ロード・ファイルも生成します。 ロード・ファイルは、**storeloadfile** パラメーターの出力に基づいて作成されます。

   例えば、ホスト localhost のポート 60000 でデータベース DB1、スキーマ db2admin に rdfStore2 というストアを作成するには、次のコマンドを発行します。 RDF データ・ファイルに myRdfData.nq を指定して、生成されるストア・ローダー・ファイルの名前として load.sql を指定します。

   ```
   createrdfstoreandloader rdfStore2 -host localhost -port 60000 -db DB1
   -user db2admin -password XXX -schema db2admin
   -rdfdatafile ./myRdfData.nq -storeloadfile ./rdfLoader/load.sql
   ```

4. CLP 対応の DB2 コマンド・プロンプト・ウィンドウを開き、 RDF ストアが作成されたデータベース・インスタンスとスキーマに接続します。
5. ./rdfLoader/load.sql ファイルを実行します。

   このようにすると、この生成されたファイルから RDF ストアへデータがロードされます。

   注: SQL スクリプトを実行しているときは **-t** 引数を使用しないでください。コマンド区切り文字として改行を使用してこの SQL スクリプトが生成されるためです。

グラフ・レベルのアクセス制御を使用する RDF ストアの作成

RDF ストアを作成する際に、ストアがグラフ・レベルのアクセス制御を使用するようにセットアップできます。

RDF グラフへのアクセスを制御するために使用するトリプルを持つ RDF 述語を決定します。例えば、ContextId (http://myapp.net/xmlns/CONTEXTID) 述語および AppId (http://myapp.net/xmlns/APPID) 述語を使用できます。これらの述語は、フィルター述語とも呼ばれます。

これらの述語のための RDF オブジェクト値に対する DB2 データ・タイプを決定します。 現在サポートされているのは DB2 VARCHAR データ・タイプのみです。

StoreManager クラスの createStore メソッドを使用します。これは、プロパティー引数を取ります。 フィルター述語は、<RDF_PREDICATE> = <DB2_DATATYPE> というプロパティー・フォーマットで指定されます。

例

次の Java プログラムは、グラフ・レベルのアクセス制御を使用する RDF ストアを作成するために、StoreManager クラスをどのように使用できるかを示しています。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.ibm.rdf.store.StoreManager;

public class CreateStoreGraphAccessControl {

 /* These are two RDF predicates based on which
   * access to that graph is controlled.
  */
private final static String APPID =
"http://myapp.net/xmlns/APPID";
 private final static String CONTEXTID =
"http://myapp.net/xmlns/CONTEXTID";

 /**
  * The DB2 data type for these predicates is assigned.
  */
 private final static String APPID_DATATYPE = "VARCHAR(60)";
 private final static String CONTEXTID_DATATYPE = "VARCHAR(60)";



/*
  * Create a java.util.properties that list these two
   * properties and their datatypes, where
   * propertyName is the RDF predicate, and
   * propertyValue is the data type for the RDF predicate.
  */
 private static Properties filterPredicateProps = new
Properties();
 static {
  filterPredicateProps.setProperty(APPID, APPID_DATATYPE);
  filterPredicateProps.setProperty(CONTEXTID,
CONTEXTID_DATATYPE);
 }


 public static void main(String[] args) throws SQLException {

  Connection conn = null;

  // Get a connection to the DB2 database
  try {
```

```
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf",
"db2admin", "db2admin");
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /*
   *  Create the store with the access control predicates.
  */
  StoreManager.createStore(conn, "db2admin",
"SampleAccessControl", null,
   filterPredicateProps);
 }
}
```

第 11 章 RDF ストア内のデータの変更

RDF ストア内のデータは、JENA API または SPARQL Update 操作のいずれかを使用して変更することができます。

JENA API の使用による RDF ストア内のデータの変更

RDF ストアにデータを挿入したり、RDF ストア内のデータを更新したり、RDF ストアからデータを削除したりすることができます。

例

次のプログラムは、JENA API を使用して、RDF ストア内のトリプルとグラフを変更する方法を示しています。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.hp.hpl.jena.graph.Graph;
import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.graph.Triple;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.VCARD;

import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;

public class RDFStoreSampleInsert {

 public static void main(String[] args) throws SQLException {

  Connection conn = null;
  Store store = null;
  String storeName = "sample";
  String schema = "db2admin";

  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
   conn.setAutoCommit(false);
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }


  // Create a store or dataset.
  store = StoreManager.createStore(conn, schema, storeName, null);

  // If the store already exists, connect to it.
  //store = StoreManager.connectStore(conn, schema, storeName);

   // Delete store if required.
  //StoreManager.deleteStore(conn, schema, storeName);
```

```
 /*
  * Generally retain the 'Store' object. Otherwise there is always
    * an unnecessary query to know which set of tables we need to
    * work with. The 'Store' object does not keep a reference to the connection
    * passed to the StoreManager methods. Thats why in the API you need to pass
    * a connection again in RDFStoreFactory's methods. It is okay to use all
    * other objects (such as dataset, graph or model) as lightweight,
    * i.e. create a fresh object for each request.
  */

 // Add a entire named graph in the store.
 addNamedGraph(store, conn);

 // Remove an entire named graph.
 removeNamedGraph(store, conn);

 // Shows how to add a triple to the default graph.
 addTripleToDefaultGraph(store, conn);

 // Shows how to add a triple via the JENA Graph interface.
 addTripleViaGraphInterface(store, conn);


 // Delete store.
 StoreManager.deleteStore(conn, schema, storeName);

 conn.commit();
}

public static void addNamedGraph(Store store, Connection conn) {

 // Connect to a NamedModel in the store.
 Model storeModel = RdfStoreFactory.connectNamedModel(store, conn,
   "http://graph1");

 // Create a in-memory model with some data.
 Model m = getMemModelWithSomeTriples();

 // Add the whole graph to rdfstore
 storeModel.begin();
 storeModel.add(m);
 storeModel.commit();

 storeModel.close();

}

public static void removeNamedGraph(Store store, Connection conn) {

 Model storeModel = RdfStoreFactory.connectNamedModel(store, conn,
   "http://graph1");

 storeModel.begin();
 storeModel.removeAll();
 storeModel.commit();

}

public static void addTripleToDefaultGraph(Store store, Connection conn) {

 Dataset ds = RdfStoreFactory.connectDataset(store, conn);
 Model m = ds.getDefaultModel();

 // Adding via model
```

```
 m.begin();

 String personURI = "http://somewhere/JohnSmith";
 String fullName = "John Smith";
 Resource johnSmith = m.createResource(personURI);
 johnSmith.addProperty(VCARD.FN, fullName);

 m.commit();
 m.close();
}

public static void addTripleViaGraphInterface(Store store, Connection conn) {

 Graph g = RdfStoreFactory.connectNamedGraph(store, conn,
   "http://graph2");

 Node s = Node.createURI("http://sub1");
 Node p = Node.createURI("http://pred1");
 Node v = Node.createLiteral("val1");

 g.add(new Triple(s, p, v));
 g.close();
}

private static Model getMemModelWithSomeTriples() {

 Model m = ModelFactory.createDefaultModel();

 Node s = Node.createURI("somesubject");
 Node p = Node.createURI("somepredicate");
 Node v = Node.createURI("AnObject");
 Triple t = Triple.create(s, p, v);
 m.add(m.asStatement(t));

 s = Node.createURI("someothersubject");
 p = Node.createURI("someotherpredicate");
 v = Node.createURI("AnotherObject");
 t = Triple.create(s, p, v);
 m.add(m.asStatement(t));

 return m;
}
}
```

SPARQL UPDATE のサポート

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、SPARQL バージョン 1.1 UPDATE がサポートされます。 SPARQL UPDATE バージョン 1.1 では、グラフ・ストアの更新操作の 2 つのカテゴリーをサポートします。

SPARQL のグラフの更新

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、SPARQL のグラフ更新コマンドがサポートされます。これらのコマンドによって、グラフ・ストアのグラフに対するトリプルの追加と削除が容易になります。

次のグラフ更新コマンドがサポートされます。

INSERT DATA
照会に指定されたトリプルを宛先グラフに追加します。宛先グラフが存在しない場合はそれを作成します。

INSERT WHERE
照会の WHERE 条件のパターンと宛先グラフを突き合わせることでトリプルを追加します。宛先グラフが存在しない場合はそれを作成します。

DELETE DATA
照会で指定されたトリプルを削除します。 RDF ストアにもグラフにも存在しないトリプルを削除しても影響はなく、正常に完了します。

DELETE WHERE
照会の WHERE 節に指定されたパターンと突き合わせることで、トリプルを削除します。 RDF ストアにもグラフにも存在しないトリプルを削除しても影響はなく、正常に完了します。

LOAD Internationalized Resource Identifier (IRI) から RDF 文書を読み取り、そのトリプルを指定されたグラフに挿入します。宛先グラフが存在しない場合はそれを作成します。

CLEAR
指定されたグラフにあるすべてのトリプルを削除します。

SPARQL のグラフ管理

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、グラフ・ストアでグラフの作成と削除を行う SPARQL のグラフ管理コマンドがサポートされます。また、これらのコマンドによってグラフ更新操作のショートカットも提供されます。これは通常、グラフの追加、移動、およびコピーといったグラフ管理を行う際に使用されます。

次のグラフ管理コマンドがサポートされます。

CREATE
グラフ・ストアでグラフを作成します。 DB2 RDF は空のグラフを保存しないため、グラフは保持されません。

DROP 指定したグラフをグラフ・ストアから削除します。

COPY 入力グラフのすべてのデータを宛先グラフに挿入します。入力グラフのデータは影響を受けません。宛先グラフにデータがある場合、そのデータは挿入前に削除されます。

MOVE
入力グラフのすべてのデータを宛先グラフに移動します。挿入後、入力グラフは削除されます。宛先グラフにデータがある場合、そのデータは挿入前に削除されます。

ADD 入力グラフのすべてのデータを宛先グラフに挿入します。入力グラフのデータは影響を受けません。宛先グラフに初期データがある場合、そのデータはそのまま保持されます。

SPARQL UPDATE API の使用による RDF ストア内の変更

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、SPARQL バージョン 1.1 のサポート対象 UPDATE API を使用して、RDF データ・ストアのデータを更新できます。

次のプログラムは、SPARQL UPDATE API を使用して、RDF ストア内を変更する方法を示しています。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.sparql.core.Quad;
import com.hp.hpl.jena.sparql.modify.request.QuadDataAcc;
import com.hp.hpl.jena.sparql.modify.request.UpdateDataInsert;
import com.hp.hpl.jena.sparql.util.NodeFactory;
import com.hp.hpl.jena.update.UpdateAction;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;
/**
* Sample program for using SPARQL Updates
*/
public class RDFStoreUpdateSample {
 public static void main(String[] args) throws SQLException
  // Create the connection
  Connection conn = null;
  Store store = null;
  String storeName = "staffing";
  String schema = "db2admin";
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection( "jdbc:db2://localhost:50000/RDFDB",
     "db2admin", "db2admin");
   conn.setAutoCommit(false);  }
  catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }
  // Connect to the store
  store = StoreManager.connectStore(conn, schema, storeName);

  // Create the dataset
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
  // Update dataset by parsing the SPARQL UPDATE statement
  // updateByParsingSPARQLUpdates(ds);
  // Update dataset by building Update objects
  // updateByBuildingUpdateObjects(ds);
  ds.close();
  conn.commit();
}

/**
* Update by Parsing SPARQL Update
*
* @param ds
* @param graphNode
*/
private static void updateByParsingSPARQLUpdates(Dataset ds) {
 String update = "INSERT DATA
 { GRAPH <http://example/bookStore>
  { <http://example/book1> <http://example.org/ns#price>  100 } }";
 //Execute update via UpdateAction
 UpdateAction.parseExecute(update, ds);
}
/**
* Update by creating Update objects
*
* @param ds
* @param graphNode
*/
 private static void updateByBuildingUpdateObjects(Dataset ds) {
```

```
  // Graph node
  Node graphNode = NodeFactory.parseNode("http://example/book2>");
  Node p = NodeFactory.parseNode("<http://example.org/ns#price>");
  Node o = NodeFactory.parseNode("1000");
  Quad quad = new Quad(graphNode, s, p, o);
  Node s2 = NodeFactory.parseNode("<http://example/book3>");
  Node o2 = NodeFactory.parseNode("2000");
  Quad quad2 = new Quad(graphNode, s2, p, o2);
  //Create quad data to be added to the store
  QuadDataAcc acc = new QuadDataAcc();
  acc.addQuad(quad);
  acc.addQuad(quad2);
  //Create the Update object
  UpdateDataInsert insert = new UpdateDataInsert(acc);
  //Execute the Update via UpdateAction
  UpdateAction.execute(insert, ds);
 }
}
```

第 12 章 RDF ストアの照会

SPARQL を使用して、DB2 Resource Description Framework (RDF) ストアのデータを照会します。

SPARQL for RDF バージョン 1.0 がサポートされています。 また、SPARQL バージョン 1.1 の次のフィーチャーのサブセットもサポートされます。

- AVG
- COALESCE
- COUNT
- GROUP BY
- HAVING
- MAX
- MIN
- SELECT 式
- STRSTARTS
- STRENDS
- SubQueries
- SUM

DB2 バージョン 10.1 フィックスパック 2 以降、SPARQL バージョン 1.1 の次のフィーチャーもサポートされます。

- SPARQL 照会言語の UPDATE のサポート。
- SPARQL 照会言語のグラフ・ストアの HTTP プロトコルのサポート。

RDF 照会と API

SPARQL 照会言語は DB2 データベース内のデータの変更に使用されるのに対し、JENA フレームワーク API はプログラミング・インターフェースを提供します。 DB2 RDF ストアにはいくつかの制約事項があります。

SPARQL 照会のサポート

RDF データを操作する際には、考慮すべき SPARQL のさまざまな構文的または意味的な制約事項と制限があります。

URI の長さの制限

DB2 データベースには、12000 文字の長さまでの URI が保管されます。 比較演算では、最初の 2000 文字のみ使用されます。

リテラルの長さの制限

DB2 データベースには、12000 文字の長さまでのリテラルが保管されます。 比較演算および STRSTARTS や STRENDS などの他の演算では、最初の 2000 文字のみ使用されます。

FILTER 式内のデータ・タイプ演算子

ストリング定数に対するフィルターで datatype SPARQL 演算子が使用されると、式は常に false に評価されます。

```
FILTER(datatype(xsd:boolean(?v)) = xsd:boolean)
```

フィルター・ステートメントが適切に評価されるようにするには、次のような式を発行します。

```
 FILTER(datatype((?v1) = datatype(?v2)) evaluates correctly.
```

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、FILTER 式の SPARQL データ・タイプ演算子においてサポートが拡張されています。

FILTER 式内の定数

左側と右側の両方のオペランドがブール定数である filter 式は、エラー ID DB255001E および SQL エラー・コード -104 とともに RdfStoreException を戻します。

```
FILTER( (TRUE || FALSE) = ?v )
```

回避策はありません。

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、FILTER 式の定数においてサポートが拡張されています。

FILTER 式内の単項マイナス

変数に単項マイナスがある filter 式はサポートされません。

```
FILTER ( -?v = -10 )
```

この式は、RdfStoreException を返します。エラー ID は DB255001E、SQL エラー・コードは -104 です。

SELECT 式内の DISTINCT * または REDUCED * 演算子

DISTINCT * または REDUCED * 演算子を持つ select 式はサポートされません。

```
Select DISTINCT * WHERE ...
```

この式は、RdfStoreException を返します。エラー ID は DB255001E、SQL エラー・コードは -104 です。

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、SELECT 式の DISTINCT * 演算子または REDUCED * 演算子においてサポートが拡張されています。

SELECT 式内のデータ・タイプ演算子

select 式で datatype SPARQL 演算子が使用されると、この式は、ストリング定義ではなく、そのタイプの内部整数値表記が返されます。

```
select datatype(?v) WHERE { ?s dc:format ?v }
```

回避策はありません。

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、SELECT 式の SPARQL データ・タイプ演算子においてサポートが拡張されています。

正規表現でのドット・エスケープ・シーケンス

正規表現パターン・マッチングでのドット・エスケープ・シーケンスには、式がドット文字に適切に一致しないという制約があります。

```
FILTER regex(?val, "example\\.com")
```

上記のコード・サンプルは、ストリング「example.com」に予期どおりには一致しません。

二重円記号エスケープ・シーケンスの制約

ストリング内の二重円記号でのエスケープ・シーケンスは、正しく解釈されません。 回避策はありません。

Cygwin および createrdfstoreandloader コマンド (Windows)

Windows プラットフォーム上で Cygwin を使用して **createrdfstoreandloader** コマンドを実行すると、Cygwin はエラー・メッセージや警告メッセージを表示せずにハングします。そのため、**createrdfstoreandloader** コマンドは、Linux または UNIX プラットフォームでのみ実行してください。そして、生成された DB2 ロード・ファイルと SQL ファイルを、Windows プラットフォーム上の DB2 サーバーにロードするために使用します。

JENA モデル API のサポート

JENA モデル API の DB2 の実装には、`Model.read()` API および `Model.add(Model)` API の使用方法について制約があります。

`Model.read()` API 使用時の重複トリプル

入力ソースに重複トリプルが含まれている場合、`Model.read()` API の JENA ライブラリー実装では 1000 個のトリプルをバッチで一括ロードするため、重複が削除されない可能性があります。 DB2 RDF ストアは、これらのバッチ内の重複トリプルのフィルターは行いません。

回避策として、常に入力ソースをメモリー内 JENA モデルに読み込んでから、`Model.add(model)` API を使用してメモリー内モデルを DB2 ストアに追加するようにします。

`Model.add(Model)` API 使用時の重複トリプル

`Model.add(Model)` API は、追加しているグラフがデータ・セット内に存在しないことを想定しています。グラフが存在していて、重複トリプルを追加している場合は、重複トリプルは削除されません。

DB2 RDF ストアの推奨されるアプローチは以下のとおりです。

1. グラフを初めて追加するときに、`Model.add(Model)` メソッドを使用します。
2. その既存グラフにトリプルを追加または更新する場合、以下の関数を使用します。
   - `model.add(s,p,v)`
   - `model.add(statement)`

- graph.add(Triple)
- Resource.addXX()

要確認: 存在するトリプルを追加すると、エラー・メッセージが戻されます。

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、DB2 製品で JENA API を実装するときの上記の制約が両方ともなくなりました。

SPARQL 照会の発行

RDF ストアに保管されたデータは照会することができます。

例

次のプログラムは、SPARQL 照会言語を使用して RDF ストア内のデータを照会する方法を示しています。

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.exception.RdfStoreException;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;

public class RDFStoreSampleQuery   {


 public static void main(String[] args) throws SQLException, IOException {

  Connection conn = null;
  Store store = null;
  String storeName = "sample";
  String schema = "db2admin";

  // Get a connection to the DB2 database.
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }


  try {

   /* Connect to required RDF store in the specified schema. */
   store = StoreManager.connectStore(conn, schema, storeName);
```

```
/* This is going to be our SPARQL query i.e. select triples
 in the default graph where object is <ibm.com>
 */
    String query = "SELECT * WHERE { ?s ?p
   <https://www.ibm.com> }";

/* Create the Query object for the SPARQL string. */
Query q = RdfStoreQueryFactory.create(query);

/* Get the Dataset interface of the RDF store. */
Dataset ds = RdfStoreFactory.connectDataset(store, conn);

/* Create a QueryExecution object, by providing the query to execute
 and the dataset against which it to be executed. */
QueryExecution qe = RdfStoreQueryExecutionFactory.create(q, ds);

long rows = 0;
Model m = null;

/* Based on the SPARQL query type, call the proper execution
  method. */
if (q.isSelectType()) {
 ResultSet rs = qe.execSelect();
 while (rs.hasNext()) {
  QuerySolution qs = rs.next();
  System.out.println(qs);
  System.out.println();
  rows++;
 }
}
else if ( q.isDescribeType() ) {
  m = qe.execDescribe();
  m.write(System.out, "N-TRIPLE");
}
else if ( q.isAskType() ) {
 System.out.println(qe.execAsk());

}
else if (q.isConstructType()) {
 m = qe.execConstruct();
 m.write(System.out, "N-TRIPLE");
}


/* Close the QueryExecution object. This is required to ensure
 no JDBC statement leaks. */
qe.close();

// Display the # of rows returned
if ( m != null ) {
 System.out.println("Number of Rows  : " + m.size());
 m.close();
}
else {
 System.out.println("Number of Rows  : " + rows);
}

}
catch(RdfStoreException e) {

 e.printStackTrace();
}
catch(Exception e) {
 e.printStackTrace();
}
```

```
 }
}
```

すべての名前付きグラフの和の作成

SPARQL 照会に対して、デフォルトのグラフを、データ・セット内のすべての名前付きグラフの和にするように設定することができます。 この機能は、照会にのみ適用されます。 これによる保管への影響やロードの変更はありません。

以下のプログラムは、SPARQL 照会に対するデフォルトのグラフを、データ・セット内のすべての名前付きグラフの和にするように設定する方法を示しています。

例

1. 次のサンプルの Java プログラムでは、ストア・オブジェクトからのすべての照会に対して、デフォルトのグラフを設定します。

```
import java.sql.Connection;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;


public class UnionDefaultGraph {

 public static void setPerQuery() {

  Connection conn = null;
  Store store = null;

  // get a connection to the DB2 database
  //conn = DriverManager.getConnection(...);


store = StoreManager.connectStore(conn, "db2admin",
"Sample");
  String query = "SELECT * WHERE { ?s ?p <https://www.ibm.com> }";
  Query q = RdfStoreQueryFactory.create(query);
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
  QueryExecution qe = RdfStoreQueryExecutionFactory.create(q,
ds);

  /* Set the default graph as the union of all named graphs
     * in the data set just for this query.
   */
  qe.getContext().set(Symbols.unionDefaultGraph, true);

  // Proceed to execute the query.

 }

}
```

2. 次のサンプルの Java プログラムでは、照会ごとにデフォルトのグラフを設定します。

```
import java.sql.Connection;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;


public class UnionDefaultGraph {

 public static void setPerQuery() {

  Connection conn = null;
  Store store = null;

  // get a connection to the DB2 database
  //conn = DriverManager.getConnection(...);


store = StoreManager.connectStore(conn, "db2admin",
"Sample");
  String query = "SELECT * WHERE { ?s ?p <https://www.ibm.com> }";
  Query q = RdfStoreQueryFactory.create(query);
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
  QueryExecution qe = RdfStoreQueryExecutionFactory.create(q,
ds);

  /* Set the default graph as the union of all named graphs
    * in the data set just for this query.
   */
  qe.getContext().set(Symbols.unionDefaultGraph, true);

  // Proceed to run the query.

 }

}
```

カスタム DESCRIBE ハンドラーの登録

ARQ 定義のメカニズムを使用して、DESCRIBE 照会が処理される方法をカスタマイズします。

デフォルトの DESCRIBE ハンドラーは、既に DB2 RDF ストアに登録されています。 これは、選択されたリソースの 1 レベルの深さの説明を提供します。

独自の DESCRIBE ハンドラーを実装する場合は、DB2 データベース・サーバーに対して行われる呼び出しの回数を最小限にしてください。

例

以下のプログラムは、DB2 RDF ストアに対して、独自の DESCRIBE ハンドラーを登録して実装する方法を示しています。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.HashSet;
```

```
import java.util.Iterator;
import java.util.Set;

import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.graph.Triple;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.rdf.model.AnonId;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.ResourceFactory;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandler;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandlerFactory;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandlerRegistry;
import com.hp.hpl.jena.sparql.util.Context;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;

public class DescribeTest {

 /**
  * @param args
  * @throws ClassNotFoundException
  * @throws SQLException
  */

 public static void main(String[] args) throws ClassNotFoundException,
   SQLException {
  if (args.length != 5) {
   System.err.print("Invalid arguments.¥n");
   printUsage();
   System.exit(0);
  }

    /* Note: ensure that the DB2 default describe handler is also removed.
     * Use the ARQ API's to remove the default registered describe handlers.
     * If you don't do this, every resource runs through multiple describe
     * handlers, causing unnecessarily high overhead.
     */

  /*
   * Now Register a new DescribeHandler (MyDescribeHandler)
   */
  DescribeHandlerRegistry.get().add(new DescribeHandlerFactory() {
   public DescribeHandler create() {
    return new MyDescribeHandler();
   }
  });

  /*
   * Creating database connection and store object.
   */
  Store store = null;
  Connection conn = null;

  Class.forName("com.ibm.db2.jcc.DB2Driver");

  String datasetName = args[0];
  String url = args[1];
  String schema = args[2];
  String username = args[3];
```

```
 String passwd = args[4];

 conn = DriverManager.getConnection(url, username, passwd);

 if (StoreManager.checkStoreExists(conn, schema, datasetName)) {
  store = StoreManager.connectStore(conn, schema, datasetName);
 } else {
  store = StoreManager.createStore(conn, schema, datasetName, null);
 }

 /*
  * Creating dataset with test data.
  */
 Dataset ds = RdfStoreFactory.connectDataset(store, conn);

 ds.getDefaultModel().removeAll();
 ds.getDefaultModel().add(getInputData());

 /*
  * Executing a DESCRIBE SPARQL query.
  */
 String sparql = "DESCRIBE <http://example.com/x>";

 Query query = RdfStoreQueryFactory.create(sparql);

 QueryExecution qe = RdfStoreQueryExecutionFactory.create(query, ds);

 Model m = qe.execDescribe();

 m.write(System.out, "N-TRIPLES");

 qe.close();

 conn.close();
}

private static void printUsage() {
 System.out.println("Correct usage: ");
 System.out.println("java DescribeTest <DATASET_NAME>");
 System.out.println(" <URL> <SCHEMA> <USERNAME> <PASSWORD>");
}

// Creating input data.
private static Model getInputData() {
 Model input = ModelFactory.createDefaultModel();

 Resource iris[] = {
   ResourceFactory.createResource("http://example.com/w"),
   ResourceFactory.createResource("http://example.com/x"),
   ResourceFactory.createResource("http://example.com/y"),
   ResourceFactory.createResource("http://example.com/z") };

 Property p = ResourceFactory.createProperty("http://example.com/p");
 Node o = Node.createAnon(new AnonId("AnonID"));

 for (int i = 0; i < iris.length - 1; i++) {
  input.add(input.asStatement(Triple.create(iris[i].asNode(), p
    .asNode(), iris[i + 1].asNode())));
 }

 input.add(input.asStatement(Triple.create(iris[iris.length - 1]
   .asNode(), p.asNode(), o)));

 return input;
}
}
```

```
/*
 * Sample implementation of DescribeHandler.
  */
class MyDescribeHandler implements DescribeHandler {

 /*
  * Set to keep track of all unique resource which are
  * required to DESCRIBE.
  */
 private Set <Resource> resources;
      private Model accumulator;

 // Remaining field variables

 public void start(Model accumulator, Context ctx) {
  resources = new HashSet <Resource>();
  this.accumulator = accumulator;
  // Other object declaration as needed.
 }

 public void describe(Resource resource) {
  resources.add(resource);
 }

 public void finish() {
  /*
   * Implement your own describe logic.
     * Add the new triples to 'accumulator' model object.
     * It is best to avoid multiple calls to the database, hence
   * structure your logic accordingly.
     * If you need FullClosure, use the
   * com.ibm.rdf.store.internal.jena.impl.DB2Closure.closure() APIs,
   * instead of com.hp.hpl.jena.sparql.util.Closure.closure().
   */


 }
}
```

DB2 データベース・サーバーを使用したグラフ・レベルのアクセス制御の実施

DB2 エンジンにアクセス制御を実施させることにより、SPARQL 照会が特定の RDF グラフのみにアクセスできるようにすることが可能です。

手順

システム述語のメタデータ表には、RDF 述語が入っています。 このメタデータは、ストア作成時に、グラフ・レベルのアクセス制御を実施するために指定されます。 また、このメタデータには、これらの述語の値が入った直接 1 次表およびリバース 1 次表の列の名前も格納されています。

次の照会を発行します。

```
"select * from System_predicates_table>"
```

出力例:

```
ENTRY_ID   COLNAME        MAPNAME
=======================================================
1          SYSPRED_0      http://myapp.net/xmlns/APPID
1          SYSPRED_1      http://myapp.net/xmlns/CONTEXTID
```

ここで、SYSPRED_0 は述語 http://myapp.net/xmlns/APPID の列を示しており、SYSPRED_1 は述語 http://myapp.net/xmlns/CONTEXTID の列 (直接 1 次表およびリバース 1 次表の) を示しています。
これらの列を使用する直接 1 次表およびリバース 1 次表での要件に応じて、DB2 ソフトウェアの細かいアクセス制御のフィーチャーを使用して、ROW LEVEL 制約を定義できます。

RDF ストア SQL 生成プログラムを使用したグラフ・レベルのアクセス制御の実施

特定の RDF グラフへのアクセスを制御することができます。 RDF ストア SQL 生成プログラムが、生成した SQL に正しいフィルターを適用することによって、アクセスを制御できます。

トリプルを含んだ RDF グラフに基づく RDF 述語の値を決定します。この値によって、アクセス対象となるグラフが決まります。

アクセス制御の検査を、単一値に対して行うのか、値のセット内の 1 つに対して行うのかを決定します。アクセス制御の検査を単一値に対して行う場合、その値を返す QueryFilterPredicateEquals オブジェクトを作成します。アクセス制御のチェックを値のセット内の 1 つの値に対して行う場合、その値のセットを返す QueryFilterPredicateMember オブジェクトを作成します。それぞれのアクセス制御フィルター述語に対して、このプロセスを繰り返します。

発行される SPARQL 照会の QueryExecution コンテキストに、オブジェクトを設定します。

例

次の Java プログラムは、RDF ストア SQL 生成プログラムを使用したグラフのアクセス制御の方法を示します。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Literal;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.ResourceFactory;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.query.filter.QueryFilterPredicate;
import com.ibm.rdf.store.query.filter.QueryFilterPredicateEquals;
import com.ibm.rdf.store.query.filter.QueryFilterPredicateMember;
```

```
import com.ibm.rdf.store.query.filter.QueryFilterPredicateProvider;

public class QueryStoreGraphAccessControl {

 /* Property objects for the two RDF predicates based on whose triples
  * access to the graph is controlled.
  */
 private final static Property APPID =
ModelFactory.createDefaultModel()
.createProperty("http://myapp.net/xmlns/APPID");

 private final static Property CONTEXTID =
ModelFactory.createDefaultModel()
   .createProperty("http://myapp.net/xmlns/CONTEXTID");


 public static void main(String[] args) throws SQLException {

  Connection conn = null;
  Store store = null;

  // get a connection to the DB2 database
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf",
"db2admin","db2admin");
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /* Connect to the access controlled store. */
  store = StoreManager.connectStore(conn, "db2admin",
"SampleAccessControl");
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);

  // Insert some data for querying
  insertDataForQuerying(ds);

  // Query and ensure access control is enforced
  QueryWithGraphAccessControl(ds);

 }


 private static void QueryWithGraphAccessControl(Dataset ds) {

  //Create the filter value for APPID.
  final QueryFilterPredicateEquals appIdFilter =
   new QueryFilterPredicateEquals() {
    public Literal getValue() {
     return ModelFactory.createDefaultModel()
       .createLiteral("App1");
    }
   };

  //Create the filter value set for contextID.
  final QueryFilterPredicateMember ctxIdFilter = new
QueryFilterPredicateMember() {
    public List <> getValues() {
      List<Literal> a = new ArrayList<Literal>();
      a.add(ModelFactory.createDefaultModel()
       .createLiteral("Context1"));
      a.add(ModelFactory.createDefaultModel()
       .createLiteral("Context2"));
      return a;
    }
```

```
 };

  // Create the QueryExecution object.
  QueryExecution qe = RdfStoreQueryExecutionFactory.create(
   "select ?who where { ?who <http://pre/test.3> ?x }",
ds);

  // Set the access control filter values for this query.
  qe.getContext().set(Symbols. queryFilterPredicates,
new QueryFilterPredicateProvider() {
    public QueryFilterPredicate getQueryFilterPredicate(
     Property filterProperty) {
    if (filterProperty.equals(APPID) ) {
     return appIdFilter;
    }
    else if ( filterProperty.equals(CONTEXTID)) {
     return ctxIdFilter;
    }
    else
     return null;

   }

   });

  // Set the default graph as a union of all named graphs.
  qe.getContext().set(Symbols.unionDefaultGraph, true);

  /* Execute SPARQL. Note only Model1 will match and Model2
     * triples are not returned */
  ResultSet rs = qe.execSelect();
  while (rs.hasNext()) {
   QuerySolution qs = rs.next();
   System.out.println(qs.toString());
  }
  qe.close();


 }

 private static void insertDataForQuerying(Dataset ds) {

  // Adding triples to graph1.
  ds.getNamedModel("Model1").add(getMemModel());

  // Adding filter predicate triples in the existing graph.
  ds.getNamedModel("Model1").add(
ResourceFactory.createResource(
      "http://res1"), APPID, "App1");

ds.getNamedModel("Model1").add(
ResourceFactory.createResource(
     "http://res1"), CONTEXTID, "Context1");

  // Adding triples to graph2.
  ds.getNamedModel("Model2").add(getMemModel());

  // Adding filter predicate triples in the existing graph.
  ds.getNamedModel("Model2").add(
ResourceFactory.createResource(
      "http://res2"), APPID, "App2");
ds.getNamedModel("Model2").add(
ResourceFactory.createResource(
     "http://res2"), CONTEXTID, "Context2");

 }
```

```
 private static Model getMemModel() {private static Model getMemModel() {
  String sub = "http://sub/";
  String pre = "http://pre/test.";
  String obj = "http://obj/";

  Model m = ModelFactory.createDefaultModel();

  long TRIPLE_COUNT = 12;
  for (int i = 0; m.size() < TRIPLE_COUNT; i++) {
   Resource s = ResourceFactory.createResource(sub + (i % 3));
   Property p = ResourceFactory.createProperty(pre + (i % 9));
   Literal o = ResourceFactory.createPlainLiteral(obj + (i % 4));
   m.add(ResourceFactory.createStatement(s, p, o));
  }

  return m;
 }


}
```

第 13 章 RDF ストアの保守

デフォルトの DB2 RDF ストア、または最適化された DB2 RDF ストアのいずれかを作成し、そこに RDF データをロードします。 最良の照会パフォーマンスが得られるように、RDF ストアのメンテナンスを行います。

RDF ストアでの統計の更新

RDF ストアに関する統計は更新することができます。

このタスクについて

RDF ストアで最適な照会パフォーマンスを得るために、RDF ストアに対しては、DB2 データベース統計と RDF ストア固有の統計の組み合わせが維持されています。

RDF ストア作成プロセスの一部として、DB2 **RUNSTATS** コマンド・プロファイルが RDF ストアのデータ表に作成されます。 **AUTORUNSTATS** パラメーターを有効にすることで、DB2 は自動的に表の分散統計を更新します。

RDF ストアで大量のデータを更新する操作を個別に実行した場合は、自動更新を待つのではなく、手動で DB2 統計収集を実行することが推奨されます。 手動で DB2 統計収集を実行するには、RDF ストア内の各データ表に対して次のコマンドを呼び出します。

```
db2 RUNSTATS ON <table_name> USE PROFILE
```

データ表は、direct_primary、direct_secondary、reverse_primary、reverse_secondary および long_strings の各表です。

SPARQL 照会が効率的な SQL 照会を生成できるようにするためには、最も一般的な RDF トリプル (RDF 主語、目的語、述語) が、RDF ストアの Topk_Stats 表に格納されていることが重要です。 最も一般的な RDF トリプルに関する情報を自動的に収集するために、RDF ストア作成の一部として、DB2 管理用タスクが作成され、DB2 管理用タスク・スケジューラーに登録されます。

ただし、このタスクのインターバルは自動的には設定されないため、その設定をするまではタスクが実行されません。 **SETSTATSSCHEDULE** コマンドの **schedule** パラメーターを使用してください。 このパラメーターは、統計収集が実行されるインターバルを表す、標準的な CRON 形式のストリングを受け入れます。 一般的に、このタスクの頻度を 1 時間より短く設定する必要はありません。

手順

- 次のコマンドを使用して、データベース DB1 のストア rdfStore2 に対する統計収集スケジュールが毎正時になるように設定します。

```
setstatsschedule rdfStore2 -host localhost -port 60000
-db DB1 -user db2admin -password XXX
-schema db2admin -schedule "*/59 * * * *"
```

- RDF ストアで大量のデータを更新する操作を個別に実行した場合は、スケジューラーを待つのではなく、手動で統計を収集してください。 手動で実行するには、**updaterdfstorestats** コマンドを実行します。

  例えば、ホスト localhost のポート 60000 でデータベース DB1、スキーマ db2admin のストア rdfStore2 に関する統計を更新するには、次のコマンドを使用します。

```
updaterdfstorestats rdfstore2 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

最適化された RDF ストアへのデフォルト・ストアの変換

デフォルト RDF ストアから開始した場合、**reorgcheckrdfstore** コマンドを使用して、ストアの最適化が必要かどうかを検証することができます。 また、最適化されたストアに対しても、述語の相関が大幅に変更された場合になどに検証を行うことができます。 次に、**reorgrdfstore** コマンドおよび **reorgswitchrdfstore** コマンドを使用して、最適化された RDF ストアに移行します。

まず、RDF ストアの再編成が必要かどうかを検証します。 次に、RDF ストア用の再編成された表を作成します。 最後に、再編成された表に切り替えます。

RDF ストアの再編成が必要かどうかの検証

RDF データの述語相関が変更され、その変更にともなって大量のデータが挿入されると、ストア内の表の列の数と長さ、および列への述語の割り当てが、データに対して最適でなくなる可能性があります。

これらの値が最適でなくなった場合、RDF ストアに対する照会と挿入のパフォーマンスに悪影響が出ることがあります。

手順

ストアの再編成が必要かどうかを判別するには、**reorgcheckrdfstore** コマンドを発行します。 例えば、データベース DB1 のストア myRdfStore の場合、次のコマンドを発行します。

```
reorgcheckrdfstore myRdfStore -db DB1
-user db2admin -password db2admin
```

表の再編成が必要ない場合は、次のメッセージが表示されます。

```
ストア myRdfStore の再編成は不要です。
(No reorganization is required for store myRdfStore.)
```

再編成が必要な表がある場合、次の例に示すように、表がリスト出力されます。

```
TABLENAME                 REORG    ColsRequired    ColSize
-------------------------------------------------------------
direct_primary_hash        true               5        118
reverse_primary_hash       true               3        118
-------------------------------------------------------------
```

次のタスク

再編成された表を作成するには、**reorgrdfstore** コマンドを使用します。 詳しくは、RDF ストアの再編成済み表の作成に関するトピックを参照してください。

RDF ストアに対する再編成済み表の作成

reorgcheckrdfstore コマンドの結果、表の再編成が必要だとわかった場合、**reorgrdfstore** コマンドを使用して、再編成された表を作成し、そこにデータを設定します。

始める前に

必ず、RDF ストアのすべての表を読み取り専用モードに変更して、**reorgrdfstore** コマンドの実行中は RDF ストアへの更新が行われないようにしてください。

手順

RDF ストアの表を再編成するには、**reorgrdfstore** コマンドを発行します。 例えば、データベースが DB1 でストアが myRdfStore の場合、次のコマンドを発行して、表 direct_primary_hash に新規の再編成済み表を作成します。

```
reorgrdfstore myRdfStore -db DB1
-user db2admin -password db2admin
-table direct_primary_hash -newtablenamesprefix reorgd
```

新しい表の名前は reorgd_*original_table_name* になります。これは、コマンドの新しい表名の接頭部として reorgd を指定したためです。 *original_table_name* の値は、`objectNames.props` プロパティー・ファイルで direct_primary_hash 表に設定された名前を表します。

タスクの結果

表の再編成にかかる時間は、ストア内のデータの容量に応じて変わります。

table パラメーターに指定した表の再編成が不要な場合、そのことを示すメッセージが表示されます。

次のタスク

新しい再編成済みの表を使用するように、ストアを変更します。 詳しくは、RDF ストアの再編成済み表への切り替えに関するトピックを参照してください。

RDF ストアでの再編成済み表への切り替え

reorgrdfstore コマンドによって再編成済みの表は作成されますが、**reorgswitchrdfstore** コマンドを発行するまで、RDF ストアはこの新しい表を使用しません。

始める前に

RDF ストアのすべてのクライアントが切断されていることを確認してください。

手順

再編成済み表を反映して RDF ストアを更新するには、**reorgswitchrdfstore** コマンドを使用します。 例えば、データベース DB1 のストア myRdfStore の場合、次のコマンドを発行します。

```
reorgswitchrdfstore myRdfStore -db DB1
-user db2admin -password db2admin
```

タスクの結果

reorgswitchrdfstore コマンドは、元の表の名前を old_*original_table_name* に変更し、元の名前は再編成済みの表に使用します。

RDF ストアが再編成済みのストアに切り替えられたときに、その表の名前は変更されていません。ストアは新しい表を使用するようになり、古い表はそのまま残されます。

次のタスク

ストアにクライアントを再接続します。

必要に応じて、古い表をドロップします。

第 14 章 RDF コマンド

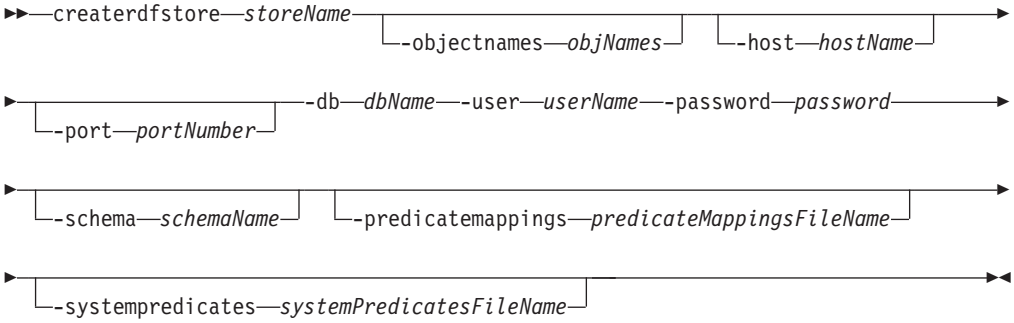
RDF コマンドは、広範なユーザー制御、カスタマイズ、および個人情報設定を行います。これらのコマンドを使用して、ストアの作成、管理、および照会に関連するさまざまなタスクを実行できます。

createrdfstore コマンド

createrdfstore コマンドは、データのない空の RDF ストアを作成します。

既存のデータを使用する最適化された RDF ストアを作成するには、代わりに **createrdfstoreandloader** コマンドを使用します。

コマンド構文



コマンド・パラメーター

-storename *storeName*
RDF ストアの名前を指定します。 この名前は、必ず DB2 データベースの表名の規則に合ったものにしてください。

-objectnames *objNames*
RDF ストア表の名前をリストする Java プロパティー・ファイルを指定します。 このファイルには任意の有効なファイル名が可能ですが、拡張子は「.properties」でなければなりません。

objectNames パラメーターを指定しない場合、システム生成された表名が使用されます。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。 最小のデータベース・ページ・サイズは 32 KB です。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアの作成に使用するデータベース・スキーマを指定します。

-predicatemappings *predicateMappingsFileName*
DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、ストアで使用する述語のマッピングを格納するファイルのパスを指定します。マッピングは、述語と割り当てられた列との間で行われます。マッピングは、述語のオカレンスに基づいて計算されます。

-systempredicates *systemPredicatesFileName*
DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、照会に適用されるフィルター述語を格納するプロパティー・ファイルを指定します。これらのシステム述語は、グラフ・レベルのアクセス制御を使用可能にできるように、RDF ストアに格納されます。

例

例 1: 次のコマンドは、ホスト localhost のデータベース DB1 に、ポート 60000 とスキーマ db2admin を使用する rdfStore1 という名前のストアを作成します。

```
createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
```

例 2: 次のコマンドは、syspreds.props ファイルのシステム述語、および predicatemappings.nq ファイルの述語のマッピングを使用して、データベース DB1 に、ポート 60000 とスキーマ db2admin を使用する rdfStore2 という名前のストアを作成します。

```
createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
-predicatemappings predicatemappings.nq -systempredicates syspreds.props
```

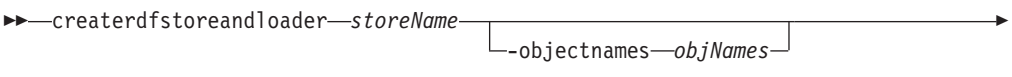
使用上の注意

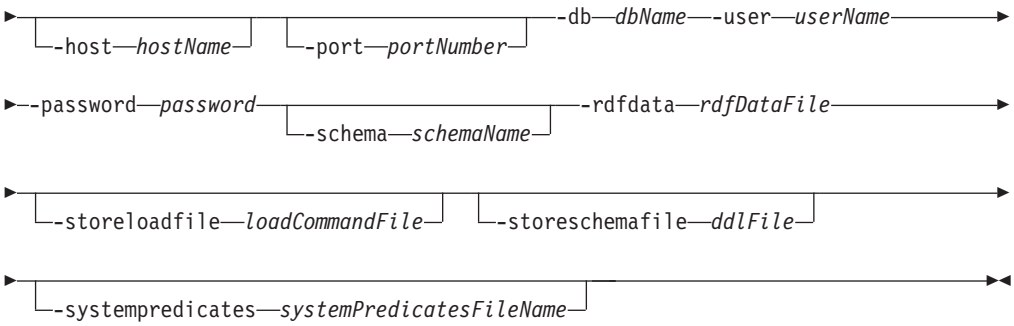
このコマンドとパラメーター名は、小文字で発行しなければなりません。

createrdfstoreandloader コマンド

createrdfstoreandloader コマンドは RDF データを分析し、既存の RDF データに対して最適化されたスキーマを持つ空の RDF ストアを作成します。 このコマンドはロード・ファイルを生成し、これらのローダー・ファイルからストアをロードするためのコマンドも生成します。

コマンド構文





コマンド・パラメーター

-storename *storeName*
RDF ストアの名前を指定します。 この名前は、必ず DB2 データベース・サーバーの表名の規則に合ったもので、データベース・スキーマの中で固有になるようにしてください。

-objectnames *objNames*
RDF ストア表の名前をリストする Java プロパティー・ファイルを指定します。 このファイルには任意の有効なファイル名が可能ですが、拡張子は「.properties」でなければなりません。

objectNames パラメーターを指定しない場合、システム生成された表名が使用されます。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。 最小のデータベース・ページ・サイズは 32 KB です。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアの作成に使用するデータベース・スキーマを指定します。

-rdfdata *rdfDataFile*
最適化された RDF ストア・スキーマの作成元になる RDF データを格納するファイルを指定します。また、このファイルの RDF データに基づいて、ロード・ファイルが作成されます。

-storeloadfile *loadCommandFile*
データを RDF ストアにロードするコマンドを格納するファイルの出力パスを指定します。このファイルは、DB2 コマンド行プロセッサー (CLP) を使用して実行できます。

-storeloadfile パラメーターを指定しない場合、loadCommands.sql という名前のファイルが現行フォルダーに作成されます。

ロード・ファイルは、コマンドを格納するファイルが作成されるフォルダーに作成されます。

-storeschemafile *ddlFile*
ストア用の DDL スクリプトが生成される出力パスとファイルを指定します。

-storeschemafile パラメーターを指定しない場合、DDL スクリプト・ファイルは作成されません。

-systempredicates *systemPredicatesFileName*
DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、照会に適用されるフィルター述語を格納するプロパティー・ファイルを指定します。これらのシステム述語は、グラフ・レベルのアクセス制御を使用可能にできるように、DB2 RDF ストアに格納されます。

例

次のコマンドは、ホスト localhost のポート 60000 でスキーマ db2admin のデータベース DB1 に、rdfStore1 という名前のストアを作成します。 入力 RDF データ・ファイルは myRdfData.nq で、生成されるストア・ローダー・ファイルの名前は load.sql です。このファイルは ./rdfLoader/ ディレクトリーに生成されます。

```
createrdfstoreandloader rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
-rdfdatafile ./myRdfData.nq -storeloadfile ./rdfLoader/load.sql
```

使用上の注意

このコマンドとパラメーター名は、小文字で発行しなければなりません。

コマンドの発行元となるディレクトリーのパスには、スペースを含めてはなりません。 **storeloadfile** パラメーターと **storeschemafile** パラメーターのパスにも、スペースを含めてはなりません。 Windows プラットフォームでは、指定されたフォルダーまたはファイル名のパスにスペースが含まれている場合は、ストリング全体を二重引用符で囲む必要があります。

Windows プラットフォームでは、**createrdfStoreAndLoader** コマンドは CygWin アプリケーションを必要とします。このコマンドに必要な Gawk ユーティリティーは、バージョン 4.0 以降です。このコマンドに必要な Core ユーティリティーは、バージョン 8.14 以降です。 CygWin をインストールした後、*<CgyWin_install_directory>*/bin を PATH 環境変数に追加します。パス上に CygWin がないと、コマンドを実行したときに次のエラー・メッセージが表示されます。

```
'Cannot run program "sh": CreateProcess error=2, The system cannot find
the specified file.'
```

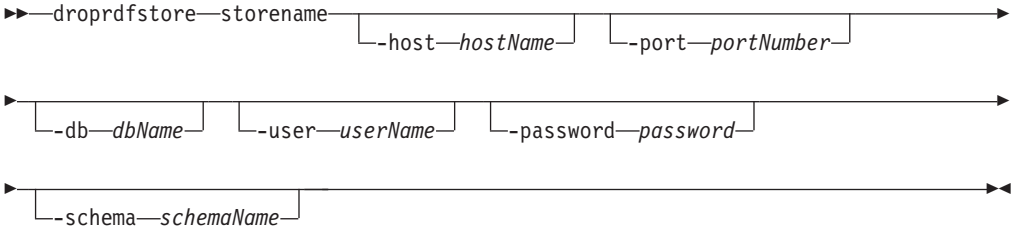
Windows プラットフォームでは、CygWin コマンド・プロンプトまたはデフォルトのコマンド・プロンプトのいずれかから **createrdfStoreAndLoader** コマンドを開始できます。 CygWin コマンド・プロンプトを使用するとき、どのファイル・パス (-rdfdata、-storeloadfile、-storeschemafile、-objectnames) にも「cygdrive」接頭部を含

めることができません。代わりに、C:¥.... などの Windows のパスを使用します。

droprdfstore コマンド

droprdfstore コマンドは、既存の RDF ストアを削除します。

コマンド構文



コマンド・パラメーター

-storename
データベースまたはスキーマの中の tripleStore を識別する名前を指定します。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF が存在するデータベース・スキーマを指定します。

例

droprdfstore コマンドを発行して、ホスト localhost のポート 60000 でデータベース DB1、スキーマ db2admin の rdfStore4 というストアを削除します。

```
droprdfstore rdfStore4 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

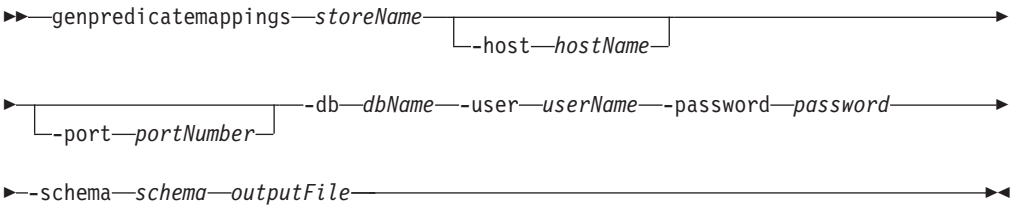
使用上の注意

- コマンドとパラメーター名は小文字で発行しなければなりません。

genpredicatemappings コマンド

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、**genpredicatemappings** コマンドは、RDF ストアの述語相関に基づいて述語マッピングを生成します。

コマンド構文



コマンド・パラメーター

storeName
RDF ストアを指定します。

-host *hostName*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアのデータベース・スキーマを指定します。

outputFile
マッピングの書き込み先のファイルのパスと名前を指定します。出力ファイルが指定されていない場合は、コンソールに出力が書き込まれます。

例

次のコマンドは、MyStore という名前の RDF ストアに述語マッピングを生成し、出力をファイルに書き込みます。

```
genpredicatemappings MyStore -db RDFSAMPL -user db2admin
-password db2admin "C:\MyStore_predicate_mappings.txt"
```

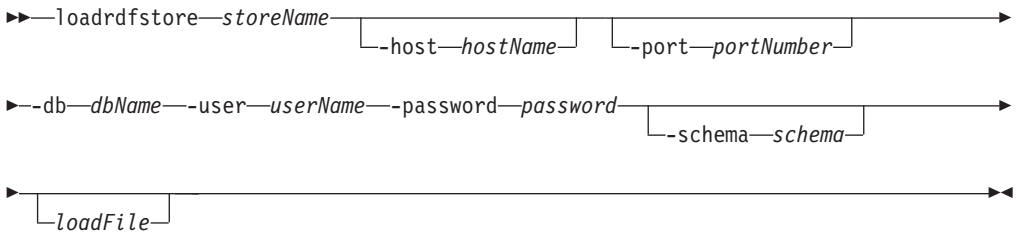
使用上の注意

このコマンドとパラメーター名は、小文字で発行しなければなりません。

loadrdfstore コマンド

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、**loadrdfstore** コマンドはトリプルを既存の RDF ストアにロードします。

コマンド構文



コマンド・パラメーター

storeName
照会する RDF ストアを指定します。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアのデータベース・スキーマを指定します。

loadFile
ロードされるトリプルを格納するファイルを指定します。ファイルのタイプは nquad、ntriple、または rdfxml のいずれかにできます。 ntriple ファイルおよび rdfxml ファイルは、デフォルトのグラフのみにロードされます。

- nquad ファイルの拡張子は .nq です。
- ntriples ファイルの拡張子は .nt です。
- rdfxml ファイルの拡張子は .rdf または .xml です。

例

次のコマンドは、RDFSAMPL データベースの myStore という名前の RDF ストアのトリプルを格納するファイルをロードします。

```
loadrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin c:\simple.nq
```

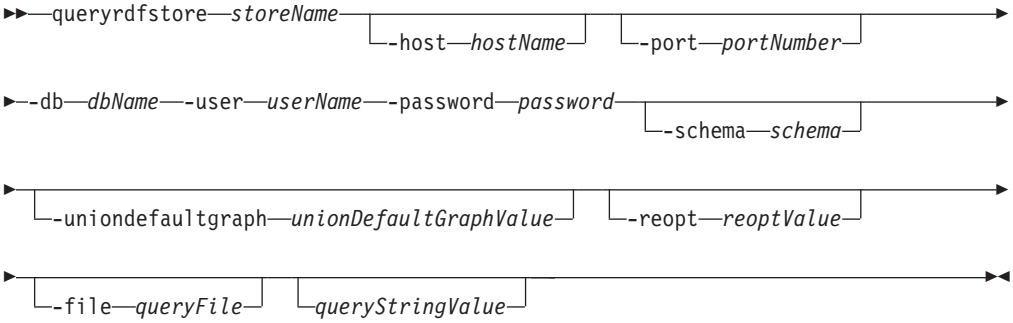
使用上の注意

このコマンドとパラメーター名は、小文字で発行しなければなりません。

queryrdfstore コマンド

DB2 バージョン 10.1 フィックスパック 2 以降のフィックスパックでは、**queryrdfstore** コマンドを使用して、コマンド行から RDF ストアを照会できます。この照会は、ファイルから、または **queryrdfstore** コマンドへの引数としてインラインで指定することで実行できます。

コマンド構文



コマンド・パラメーター

storeName
照会する RDF ストアを指定します。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアのデータベース・スキーマを指定します。

-uniondefaultgraph*unionDefaultGraphValue*
union デフォルト・グラフが使用されるかどうかを指定します。値は、`true` または `false` のいずれかです。

-reopt*reoptValue*
照会を実行するときの繰り返しのオプションを指定します。オプションは、`once`、`always`、または `none` です。デフォルト値は `none` です。

-file*queryFile*
SPARQL 照会を含むファイルを指定します。

queryStringValue
ストリングとして発行される SPARQL 照会を指定します。

例

例 1: 次のコマンドは、ローカル・システムの RDFSAMPL データベースにある myStore という名前の RDF ストアにおけるトリプルに対する照会を、**unionDefaultGraph** パラメーターを true に設定して指定します。**unionDefaultGraph** パラメーターを true に設定すると、ローカル・システム上の RDFSAMPL データベースの myStore という名前の RDF ストアのトリプルを、すべて照会できます。

```
queryrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin
-uniondefaultgraph true "select * where {?s ?p ?v}"
```

例 2: 次のコマンドは、テキスト・ファイル内の照会の使用法を指定します。

```
queryrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin
-uniondefaultgraph true -file "C:\query.txt"
```

使用上の注意

このコマンドとパラメーター名は、小文字で発行しなければなりません。

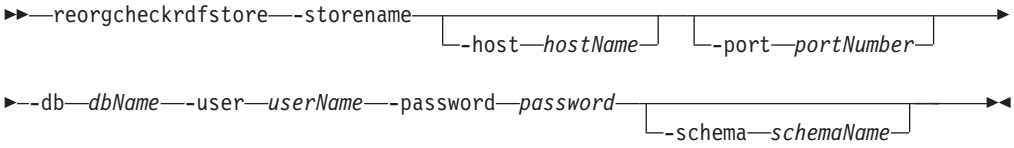
照会は、ファイル内、またはコマンドのパラメーターのいずれかで指定できますが、両方で指定することはできません。

reorgcheckrdfstore コマンド

reorgcheckrdfstore コマンドは、RDF ストアの再編成が必要かどうかを検査します。

reorgcheckrdfstore コマンドは、1 つ以上の RDF ストア表の列の数または列の長さが最適でないかどうかを識別します。 **reorgcheckrdfstore** コマンドの実行後に、**reorgrdfstore** コマンドを発行して、識別された表を再編成し、照会のパフォーマンスを向上させることができます。

コマンド構文



コマンド・パラメーター

-storename
データベースまたはスキーマの中の tripleStore の名前を指定します。

-host *hostName*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアが存在するデータベース・スキーマを指定します。

例

次のコマンドは、rdfStore3 というストアの再編成が必要かどうかを検査します。 このストアは、ホスト localhost のポート 60000 でスキーマ db2admin のデータベース DB1 にあります。

```
reorgcheckrdfstore rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

reorgcheckrdfstore コマンドの出力には、再編成を必要とする表のみがリストされます。出力には、以下のデータ列が含まれます。

tablename
RDF ストア表の論理名。

reorg 表の再編成が必要かどうかを示す true または false の値。

colsrequired
必要な列数。

colsize 最適な列の長さ。

使用上の注意

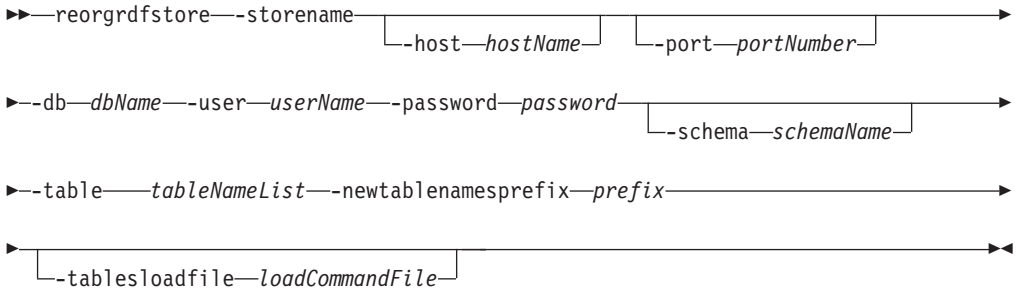
このコマンドとパラメーター名は、小文字で発行しなければなりません。

reorgrdfstore コマンド

reorgrdfstore コマンドはストア内の既存のデータに基づいて、列数と列の長さが RDF ストアに最適である、新規の再編成された表を作成します。 このコマンドはオプションで、新しい表の DDL と、再編成された表にデータをロードするロード・コマンドを含むファイルも生成します。

このコマンドは、アンロードが必要なデータの容量、ロード・ファイルの作成、および再編成された表へのデータのロードによっては、完了までに時間がかかる場合があります。 コマンドの実行中、RDF ストアのデータ表は読み取り専用にしなければなりません。

コマンド構文



コマンド・パラメーター

-storename
データベースまたはスキーマの中の tripleStore の名前を指定します。

-host *hostName*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアが存在するデータベース・スキーマを指定します。

-table *tableNameList*
再編成する表の論理名のリストを指定します。複数の表名を区切るには、パイプ文字を使用します。 リストは二重引用符内に入れる必要があります。

-newtablenamesprefix *prefix*
新しい表の接頭部を指定します。

-tablesloadfile *loadCommandFile*
新しい表の DDL および再編成された表にデータをロードするコマンドが入っているファイルの出力パスを指定します。

-tablesloadfile パラメーターを指定しない場合、現行フォルダーに loadCommands.sql という名前のファイルが作成されます。

ロード・ファイルは、ロード・コマンド・ファイルが作成されるフォルダーの中に作成されます。 これらのファイルは、CLP ウィンドウを使用して実行できます。

例

次のコマンドは、reorg という接頭部の付いた新規表名を使用して、表 direct_primary_hash および reverse_primary_hash について、ホスト localhost のポート 60000 でスキーマ db2admin のデータベース DB1 にある rdfStore3 という名前のストアのために表を再編成します。

```
reorgrdfstore rdfStore3 -host localhost -port 60000
-db DB1 -user db2admin -password XXX
-schema db2admin -table "direct_primary_hash|reverse_primary_hash"
-newtablenamesprefix reorg
```

使用上の注意

- **reorgrdfstore** コマンドは、DB2 データベース・サーバーと RDF ストアが存在するマシンで発行する必要があります。
- このコマンドとパラメーター名は、小文字で発行しなければなりません。
- このコマンドが正常に完了すると、表が自動的にロードされます。 このファイルは、CLP ウィンドウを使用して実行できます。
- このコマンドが完了しても、ストアは古い表を使い続けるため、新しい表を使用するように切り替える必要があります。 再編成された表を使用するには、**reorgswitchrdfstore** コマンドを発行します。

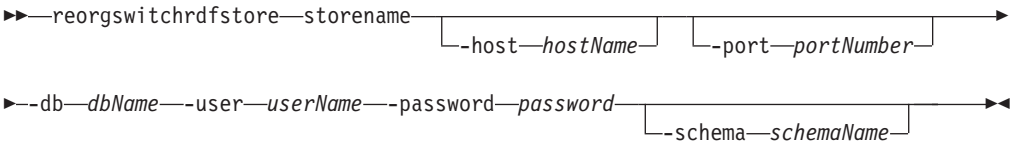
reorgswitchrdfstore コマンド

reorgswitchrdfstore コマンドは、**reorgrdfstore** コマンドを使用して作成された、新しく再編成された表を RDF ストアが使用するように切り替えます。

この操作を行う前にこの RDF ストアのクライアントをすべて切断し、コマンドの完了後に再接続する必要があります。

完了時刻は、ストア内のデータの容量には左右されません。

コマンド構文



コマンド・パラメーター

-storename
データベースまたはスキーマの中の tripleStore を識別する名前を指定します。

-host *hostNames*
ストアを作成するホストを指定します。

-port *portNumber*
ポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアが存在するデータベース・スキーマを指定します。

例

reorgswitchrdfstore コマンドを発行して、ホスト localhost のポート 60000 でデータベース DB1、スキーマ db2admin の rdfStore3 というストアの再編成済み表に切り替えます。

```
reorgswitchrdfstore rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

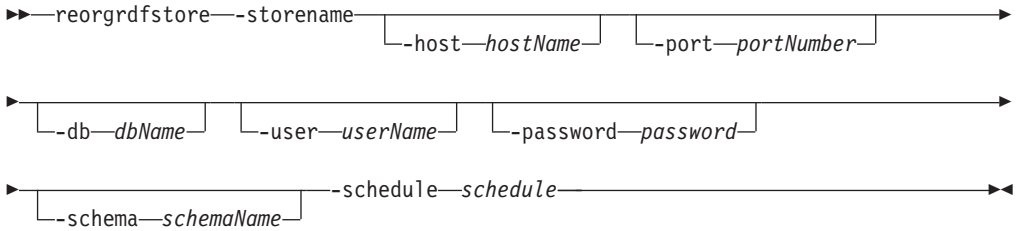
使用上の注意

- コマンドとパラメーター名は小文字で発行しなければなりません。

setstatsschedule コマンド

setstatsschedule コマンドは、RDF ストア統計の自動更新をスケジュールします。

コマンド構文



コマンド・パラメーター

-storename
データベースまたはスキーマの中の tripleStore の名前を指定します。

-host *hostName*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアが存在するデータベース・スキーマを指定します。

-schedule *schedule*
UNIX CRON 形式で、統計更新のスケジュールを指定します。 このパラメーターは二重引用符で囲んで指定する必要があります。

例

次のコマンドは、ホスト localhost のポート 60000 でスキーマ db2admin のデータベース RDFDB の RDFStore という名前のストアに対して、統計の自動更新が毎時 15 分に実行されるようにスケジュールします。

```
setStatsSchedule RDFStore -host localhost -port 60000
-db RDFDB -user db2admin -password XXX
-schema db2admin -schedule "15 * * * *"
```

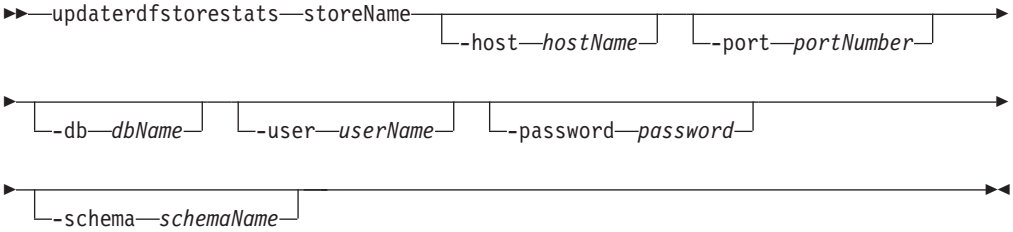
使用上の注意

- このコマンドとパラメーター名は、小文字で発行しなければなりません。

updaterdfstorestats コマンド

updaterdfstorestats コマンドは統計を更新して、RDF ストア内の現行データを反映します。

コマンド構文



コマンド・パラメーター

-storename
ストアの名前を指定します。 この名前は、データベースまたはスキーマの中で固有でなければなりません。

-host *hostNames*
データベースが存在するホストを指定します。

-port *portNumber*
データベースのポート番号を指定します。

-db *dbName*
接続を確立する先のデータベースを指定します。

-user *userName*
接続の確立に使用される許可名を指定します。

-password *password*
接続の確立に使用されるパスワードを指定します。

-schema *schemaName*
RDF ストアが存在するデータベース・スキーマを指定します。

例

次のコマンドは、ホスト localhost のポート 60000 でスキーマ db2admin のデータベース DB1 の rdfStore3 というストアに関する統計を更新します。

```
updaterdfstorestats rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

使用上の注意

- このコマンドとパラメーター名は、小文字で発行しなければなりません。

第 2 部 付録

付録 A. DB2 技術情報の概説

DB2 技術情報は、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg27009474 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 3. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-01	入手不可	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SA88-4676-01	入手可能	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SA88-4677-01	入手可能	2013 年 1 月
コマンド・リファレンス	SA88-4673-01	入手可能	2013 年 1 月
データベース: 管理の概念および構成リファレンス	SA88-4662-01	入手可能	2013 年 1 月
データ移動ユーティリティー: ガイドおよびリファレンス	SA88-4693-01	入手可能	2013 年 1 月
データベースのモニタリング ガイドおよびリファレンス	SA88-4663-01	入手可能	2013 年 1 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SA88-4694-01	入手可能	2013 年 1 月
データベース・セキュリティー・ガイド	SA88-4695-01	入手可能	2013 年 1 月

表 3. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-01	入手可能	2013 年 1 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-01	入手可能	2013 年 1 月
組み込み SQL アプリケーションの開発	SA88-4666-01	入手可能	2013 年 1 月
Java アプリケーションの開発	SA88-4669-01	入手可能	2013 年 1 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
IBM データ・サーバー用の RDF アプリケーション開発	SA88-5083-00	入手可能	2013 年 1 月
SQL および外部ルーチンの開発	SA88-4667-01	入手可能	2013 年 1 月
データベース・アプリケーション開発の基礎	GI88-4279-01	入手可能	2013 年 1 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバリゼーション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-01	入手可能	2013 年 1 月
IBM データ・サーバー・クライアント機能 インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-01	入手不可	2013 年 1 月
メッセージ・リファレンス 第 2 巻	SA88-4689-01	入手不可	2013 年 1 月
Net Search Extender 管理およびユーザーズ・ガイド	SA88-4691-01	入手不可	2013 年 1 月
パーティションおよびクラスタリングのガイド	SA88-4697-01	入手可能	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	入手不可	2013 年 1 月

表 3. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	入手不可	2013 年 1 月
pureXML ガイド	SA88-4686-01	入手可能	2013 年 1 月
Spatial Extender ユーザーズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SA88-4668-01	入手可能	2013 年 1 月
SQL リファレンス 第 1 巻	SA88-4674-01	入手可能	2013 年 1 月
SQL リファレンス 第 2 巻	SA88-4675-01	入手可能	2013 年 1 月
Text Search ガイド	SA88-4692-01	入手可能	2013 年 1 月
問題判別およびデータベース・パフォーマンスのチューニング	SA88-4664-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 の新機能	SA88-4684-01	入手可能	2013 年 1 月
XQuery リファレンス	SA88-4687-01	入手不可	2013 年 1 月

表 4. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect サーバー機能 インストールおよび構成	SA88-4682-01	入手可能	2013 年 1 月
DB2 Connect ユーザーズ・ガイド	SA88-4683-01	入手可能	2013 年 1 月

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果として生じる可能性がある状態に対応した SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v10r1 です。

DB2 バージョン 9.8 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/ です。

DB2 バージョン 9.7 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/ です。

DB2 バージョン 9.5 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 です。

DB2 バージョン 9.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9/ です。

DB2 バージョン 8 のトピックについては、*DB2 インフォメーション・センター* の URL (http://publib.boulder.ibm.com/infocenter/db2luw/v8/) を参照してください。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーション・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V10.1` ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
   c. 次のように `update-ic` スクリプトを実行します。

      ```
      update-ic
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように update-ic.bat ファイルを実行します。

```
update-ic.bat
```

タスクの結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2 インフォメーション・センター* を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2 インフォメーション・センター* を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合、インターネットに接続されていて *DB2 インフォメーション・センター* がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、「**管理者として実行**」を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、*Program_Files*¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (*Program_Files* は Program Files ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン () をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。

6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。
7. *DB2 インフォメーション・センター* を再開します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 start
     ```

タスクの結果

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド* 」の『**pureXML**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 の資料

トラブルシューティング情報は、「*問題判別およびデータベース・パフォーマンスのチューニング*」または *DB2 インフォメーション・センター* の『データベースの基本』セクションにあります。ここには、以下の情報が記載されています。

- DB2 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- DB2 データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

現在問題が発生していて、考えられる原因とソリューションを見つけるには、IBM サポート・ポータルを参照してください。 Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

適用度: これらのご利用条件は、IBM Web サイトのあらゆるご利用条件に追加で適用されるものです。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利: ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM® の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

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

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

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

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

C

D

G

L

Q

R

S

U

Printed in Japan

SA88-5083-00

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

IBM データ・サーバー用の RDF アプリケーション開発