

日日はOracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2021年12月15日 水曜日

RDFモデルにトリプルをロードするPL/SQLコードを書く

Oracle APEXのアプリケーションでRDFのモデルにデータをロードすることを想定して、PL/SQLのコードを書いてみます。以下のオラクルのマニュアルに記載されていることの確認になります。Oracle RDF Graph Serverを使用すると、GUIにてデータのインポートが可能です。そのため、インポートするデータがファイルとして存在する場合は、Oracle RDF Graph Serverを使って簡単にインポートできます。

RDFナレッジ・グラフ開発者ガイド

1.8 セマンティック・データのロードおよびエクスポート

バルク・ロードの実行

ステージング表を準備します。表名は**STAGE_TABLE_T1**とします。ステージング表は必ずしも毎回作成する必要はなく、再利用もできます。

```
CREATE TABLE stage_table_t1
(
  RDF$STC_sub varchar2(4000) not null,
  RDF$STC_pred varchar2(4000) not null,
  RDF$STC_obj varchar2(4000) not null,
  RDF$STC_graph varchar2(4000)
);
```

RDF\$STC_subに主語、RDF\$STC_predに述語、RDF\$STC_objに目的語を投入します。今回はRDF\$STC_graphは使用しません。

Oracle APEXから利用できることを確認するため、SQLは**SQLワークショップ**のSQLコマンドより実行します。

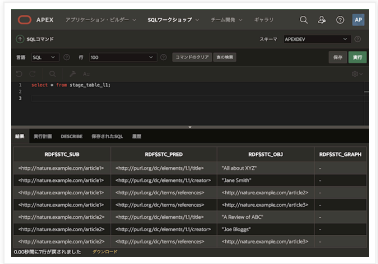


ステージング表にデータを投入します。**SQLコマンド**で実行できるのは1行のSQL、またはbegin/endで囲んだ1ブロックだけです。そのため、7行のinsert文をbegin/endで囲んでいます。

```
begin
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article1>','<http://purl.org/dc/elements/1.1/title>','"All about Nature"');
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article1>','<http://purl.org/dc/elements/1.1/creator>','"Jane Smith"');
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article1>','<http://purl.org/dc/terms/references>','<http://nature.example.com/article2>');
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article2>','<http://purl.org/dc/elements/1.1/title>','"A Review of Nature"');
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article2>','<http://purl.org/dc/elements/1.1/creator>','"Joe Bloggs"');
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj) values('<http://nature.example.com/article2>','<http://purl.org/dc/terms/references>','<http://nature.example.com/article3>');
end;
```

投入された内容を確認します。元のデータはマニュアルに記載のある[雑誌記事の情報](#)です。

```
select * from stage_table_t1;
```



モデルを作成します。モデルの名前は**T1**とします。

```
begin
  sem_apis.create_sem_model(
    model_name => 'T1',
```

```

        table_name => null,
        column_name => null,
        network_owner => 'APEXDEV',
        network_name => 'NET1'
    );
end;
/

```

ステージング表**STAGE_TABLE_T1**の内容をモデル**T1**へロードします。**flags**の指定については[マニュアルの記載](#)を参考にします。**PARSE**は必須です。

```

begin
    sem_apis.bulk_load_from_staging_table(
        model_name => 'T1'
        , table_owner => 'APEXDEV'
        , table_name => 'STAGE_TABLE_T1'
        , flags => ' PARSE '
        , network_owner => 'APEXDEV'
        , network_name => 'NET1'
    );
end;
/

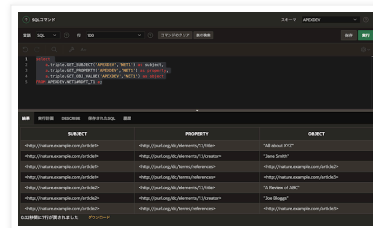
```

ロードされた内容を確認します。**ネットワーク名#RDFT_モデル名**の表を検索します。今回の例では**NET1#RDFT_T1**です。

```

select
    a.triple.GET_SUBJECT('APEXDEV','NET1') as subject,
    a.triple.GET_PROPERTY('APEXDEV','NET1') as property,
    a.triple.GET_OBJ_VALUE('APEXDEV','NET1') as object
FROM APEXDEV.NET1#RDFT_T1 a;

```



SUBJECT	PROPERTY	OBJECT
http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network
http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network
http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network
http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network
http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network	http://xmlns.com/sem/rdf/Network

以上でバルク・ロードの確認ができました。

ステージング表に投入されているデータにASCIIのBS(8)、HT(9)、LF(10)、FF(12)、CR(13)が含まれる場合、またはUnicode文字が含まれる場合は、あらかじめSEM_APIS.ESCAPE_RDF_TERM（もしくはSEM_APIS.ESCAPE_RDF_VALUE、SEM_APIS.ESCAPE_CLOB_TERM、SEM_APIS.ESCAPE_CLOB_VALUE）を使ってエンコードする必要があります。

例えば、エンコードされたデータを保持するステージング表を作成します。

```

CREATE TABLE stage_table_t1_enc
(
    RDF$STC_sub varchar2(4000) not null,
    RDF$STC_pred varchar2(4000) not null,
    RDF$STC_obj varchar2(4000) not null,
    RDF$STC_graph varchar2(4000)
);

```

その後、元のステージング表よりデータをエンコードしてコピーします。

```

insert into stage_table_t1_enc(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj)
select
    sem_apis.escape_rdf_term(RDF$STC_sub, options => ' UNI_ONLY=T '),
    sem_apis.escape_rdf_term(RDF$STC_pred, options => ' UNI_ONLY=T '),
    sem_apis.escape_rdf_term(RDF$STC_obj, options => ' UNI_ONLY=T ')
from stage_table_t1;

```

エンコードされたデータを保持しているステージング表から、ロード処理を実行します。

```

begin
    sem_apis.bulk_load_from_staging_table(
        model_name => 'T1'
        , table_owner => 'APEXDEV'
        , table_name => 'STAGE_TABLE_T1_ENC'
        , flags => ' PARSE '
        , network_owner => 'APEXDEV'
        , network_name => 'NET1'
    );
end;
/

```

エスケープされたデータを元に戻すためにSEM_APIS.UNESCAPE_RDF_TERMが使えます。

動作テストのために作業を繰り返す際に実行したコマンドを、以下に記録しておきます。作成済みのモデルからセマンティック・データをエクスポートする手順です。

```

-- ステージング表の内容を削除する。
delete from stage_table_t1;
-- モデルからデータエクスポートし、ステージング表にインポートする。
insert into stage_table_t1(RDF$STC_sub,RDF$STC_pred,RDF$STC_obj)
select

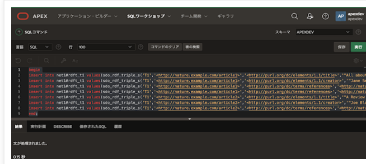
```

```
a.triple.GET_SUBJECT('APEXDEV','NET1') as subject,
a.triple.GET_PROPERTY('APEXDEV','NET1') as property,
a.triple.GET_OBJ_VALUE('APEXDEV','NET1') as object
FROM APEXDEV.NET1#RDFT_T1 a;
-- モデルをドロップする。
begin
  sem_apis.drop_sem_model(
    model_name => 'T1'
    , options => null
    , network_owner => 'APEXDEV'
    , network_name => 'NET1'
  );
end;
/
```

INSERT文の実行によるデータのロード

モデルのトリプルを保持している表は**ネットワーク名#RDFT_モデル名**です。今回の例ではNET1#RDFT_T1なので、この表に直接**SDO_RDF_TRIPLE_S**のインスタンスであるトリプルを挿入します。

```
begin
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article1>', '<http://purl.org/dc/elements/1.1/title>', '"All about XYZ"', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article1>', '<http://purl.org/dc/elements/1.1/creator>', '"Jane Smith"', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article1>', '<http://purl.org/dc/terms/references>', '<http://nature.example.com/article1>', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article1>', '<http://purl.org/dc/terms/references>', '<http://nature.example.com/article1>', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article2>', '<http://purl.org/dc/elements/1.1/title>', '"A Review of ABC"', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article2>', '<http://purl.org/dc/elements/1.1/creator>', '"Joe Bloggs"', 'APEXDEV', 'NET1')
insert into net1#rdft_t1 values(sdo_rdf_triple_s('T1','<http://nature.example.com/article2>', '<http://purl.org/dc/terms/references>', '<http://nature.example.com/article2>', 'APEXDEV', 'NET1')
end;
```



マニュアルの記載で不明だった点

ステージング表の非セマンティック・データ列の扱い

ステージング表に非セマンティック・データ用の列を追加する例が記載されています。以下のDDLです。

```
CREATE TABLE stage_table_with_extra_cols (
  source VARCHAR2(4000),
  id NUMBER,
  RDF$STC_sub varchar2(4000) not null,
  RDF$STC_pred varchar2(4000) not null,
  RDF$STC_obj varchar2(4000) not null,
  RDF$STC_graph varchar2(4000)
);
```

バルク・ロードを実行するAPIのSEM_APIS.BULK_LOAD_FROM_STAGING_TABLEに、追加列を扱う方法が記載されていません。ユーザー作成のアプリケーションの都合に応じてステージング表に列を追加しても、バルク・ロードには影響がない、ということをおっしゃいます。実際、以下の定義で作成したステージング表から問題なく、データのロードができました。

```
CREATE TABLE stage_table_t1
(
  source varchar2(4000),
  id number,
  RDF$STC_sub varchar2(4000) not null,
  RDF$STC_pred varchar2(4000) not null,
  RDF$STC_obj varchar2(4000) not null,
  RDF$STC_graph varchar2(4000)
);
```

データのエスケープについて

データのエスケープを行うために、以下のAPIが用意されています。

[SEM_APIS.ESCAPE_CLOB_TERM](#)

[SEM_APIS.ESCAPE_CLOB_VALUE](#)

TERMはURI、VALUEはリテラルを対象にしていると思われますが、マニュアルにそういった記載はありません。また、SEM_APIS.ESCAPE_RDF_VALUEについては、CLOB値が返される、といった誤った記載がされています。ファンクション定義については、マニュアルと実際とは違いがあります。APIの問題というよりはマニュアルの誤記のようです。

FUNCTION ESCAPE_RDF_VALUE RETURNS VARCHAR2				
Argument	Name	Type	In/Out	Default?

VAL		VARCHAR2	IN	
UTF_ENCODE		NUMBER	IN	DEFAULT
ALLOW_LONG		NUMBER	IN	DEFAULT
OPTIONS		VARCHAR2	IN	DEFAULT

データをエスケープする際には注意が必要でしょう。

SEM_APIS.CREATE_SEM_MODELの表指定

表を作成し、その表をSEM_APIS.CREATE_SEM_MODELの引数table_nameに与えると、表がビューに置き換えられます。

例えば表**T1_DATA**を作成します。

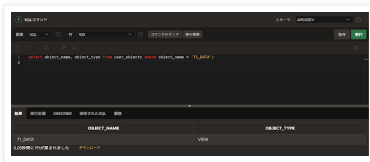
create table t1_data(triple sdo_rdf_triple_s) compress;

続いてモデル**T1**を作成します。

```
begin
  sem_apis.create_sem_model(
    model_name => 'T1',
    table_name => 'T1_DATA',
    column_name => 'TRIPLE',
    network_owner => 'APEXDEV',
    network_name => 'NET1'
  );
end;
```

以下のSQLを実行し、オブジェクト・タイプを確認すると、VIEWになっています。

select object_name, object_type from user_objects where object_name = 'T1_DATA';



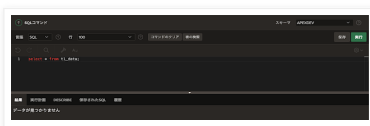
ビューとしての定義は以下です。表**NET#RDFT_T1**を参照しています。

```
CREATE OR REPLACE FORCE VIEW "T1_DATA" ("TRIPLE") AS
  SELECT triple AS "TRIPLE" FROM "APEXDEV"."NET1#RDFT_T1
/
```

モデル**T1**をドロップすると、ビューから表に戻ります。

```
begin
  sem_apis.drop_sem_model(
    model_name => 'T1'
    , options => null
    , network_owner => 'APEXDEV'
    , network_name => 'NET1'
  );
end;
```

ドロップ後の確認です。モデル**T1**をドロップしているため、表**NET1#RDFT_T1**もドロップされています。ビュー**T1_DATA**は**T1_DATA**に戻っていますが、モデルは削除されているため表**T1_DATA**の内容も空になっています。



SEM_APIS.CREATE_SEM_MODELの引数としてtable_nameを与えると、**ネットワーク名#RDFT_モデル名**より判別しやすい名前（データの挿入などに）扱えますが、データの保持のされ方が変わるといったことは無いようです。

現行のRDFグラフでは**スキーマプライベート・ネットワーク**が推奨ですし、それでネットワークを作成していれば、モデル作成時に表を指定する利点はあまりないでしょう。

[ウェブ バージョンを表示](#)

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

[詳細プロフィールを表示](#)