データ定義

平成 13 年 6 月 15 日
峯 肇史

1　データモデル

DBMS は，データベースを各種アプリケーションから統一的に利用可能とするため，対象データをそれに対する操作を規定した共通の枠組みを提供する．この枠組みのことをデータモデルと呼ぶ．データモデルを用いることで，データの物理的な格納形態やデータ検索手順の詳細に関与することなく，論理的なレベルでのデータ記述と操作を行うことが可能になる．代表的なデータモデルの 1 つにリレーショナルデータモデルがある．リレーショナルデータモデルでは，データベースを「表」の集まりとしてとらえる見方を提供する．

2　データ定義

DBMS は，データモデルに基づくデータ記述ならびにデータ操作のためのデータベース言語 (database language) を提供する．データベース言語の内，主にデータ記述を行うための言語をデータ定義言語 (data definition language, DDL) と呼び，データ操作を行うための言語をデータ操作言語 (data mamipulation language, DML) と呼ぶ．データ定義とはデータ定義言語を用いて，データベースに作成するデータを DBMS に定義することである．

3　Jasmine でのデータ定義

今回は，以下のような 3 つの情報をデータベースに格納するとして Jasimine を用いたデータベース定義の手順を説明する．

得意先 (Customer)

得意先コード (customerNo)	得意先名 (customerName)	住所 (customerAddress)
1	峯商店	福岡市東区箱崎
2	森田菓子店	福岡市東区筥松
⋮	⋮	⋮

商品 (Commodity)

商品コード (commodityNo)	商品名 (commodityName)	単価 [円](unitPrice)
1	チョコレート	200
2	ガム	100
⋮	⋮	⋮

売り上げ (Sales)

売り上げ番号 (salesNo)	商品コード (soldCommodityNo)	得意先コード (purchaserNo)	数量 (quantity)	金額 [円] (amountSold)
1	1	2	100	20,000
2	2	1	50	5,000
⋮	⋮	⋮	⋮	⋮

上の例はすべてテーブルで表してあるので，リレーショナルデータベースを用いる場合は，このテーブルをそのまま定義すればよい．(リレーショナルデータベースのデータベース言語の国際的な標準言語は SQL である．)

しかし今回使用する DBMS Jasmine はオブジェクトデータベースであるので，それぞれのテーブルを 1 つのクラス，テーブルの各属性をそれぞれ対応するクラスの属性としてデータベースに定義する．つまり以下のようなクラスを定義する．

- Customer クラス
  属性
  - customerNo (整数)
  - customerName (文字列)
  - customerAddress (文字列)
- Commodity クラス
  属性
  - commodityNo (整数)
  - commodityName (文字列)
  - unitPrice (整数)
- Sales クラス
  属性
  - salesNo (整数)
  - soldCommodityNo (整数)
  - purchaserNo (整数)
  - quantity (整数)
  - amountSold (整数)

Jasmine では，ODQL というデータベース言語を用いてデータの定義，操作を行うことになっている．そこで，ODQL を用いてこれらのクラスをデータベースに定義する．
ODQL におけるクラスの生成は，次の 2 ステップからなる．

- defineClass コマンドを使って ODQL インタプリタでクラス定義を行う．
- buildClass コマンドを使ってクラスの構築を行う．

クラスの構築を行うときに，すべての定義が互いに矛盾しないかどうかがチェックされる．構築が終了すると，それらのクラスに対してインスタンスの生成が行えるようになる．

3.1 クラス定義

ODQL を用いた Customer クラスのクラス定義は以下のようになる．(左端の数字は行番号で実際のコードには存在しない．)

```
1. defaultCF accountCF; /*自分が使うクラスファミリ名 アカウント名 CF*/
2. defineClass    Customer          /*クラス名*/
3. super:Composite                  /*上位クラス*/
4. description:"Class Customer "    /*クラスの説明*/
5. {
6.  instance:                       /*これより下の属性はインスタンス属性*/
7.   Integer       customerNo       unique: mandatory:;
8.   String        customerName;
9.   String        customerAddress;
10. };
```

各行の説明は

1. デフォルトのクラスファミリ名を accountCF にする．
2. Customer という名前のクラスの定義を開始する．
3. このクラスの上位クラスを指定．(Jasmine では，ユーザ定義クラスは必ず Composite クラスから派生することになっている．)
4. このクラスの説明を記述．
5. これより属性の定義．
6. instace:と記述することによりこれより下に記述する属性は，インスタンス属性であることを示す．(クラス属性の場合は，class:の後に記述すればよい)
7. 整数型の customerNo という名前のインスタンス属性の定義．
8. 文字列型の customerName という名前のインスタンス属性の定義．
9. 文字列型の customerAddress という名前のインスタンス属性の定義．
10. クラス定義の終了．

6 行目の unique, mandatory というのは，制約条件のための記述である．unique は，このクラスのインスタンスすべての中で重複した値を許さないことの記述であり．mandatory は，この値に空の値 (NIL) を許さないこと記述している．(詳しい説明は，制約条件の回で説明されるであろう)

このコードをインタプリタで 1 行ずつ入力すれば，クラス定義が出来るが，もし間違えるとまたもう 1 度初めから入力しなければならないので，普通はこのコードをファイルに記述しておきそのファイルをインタプリタのコマンド execFile を用いて読み込む．例えば上の ODQL 文を sample.odql というファイルに保存したとすると以下のようにして ODQL 文を実行できる．

```
> execFile sample.odql;
```

ここで 1 つ注意点であるが，Jasmine ではクラスはクラスファミリ名の修飾が必要となっている．そこで defineClass の記述の前に以下の行を付け加えることを忘れないでもらいたい．
defaultCF mineCF;
(mineCF のところは自分が使うクラスファミリ名．アカウント名 CF)
このコマンドにより，デフォルトのクラスファミリの名前が mineCF となるので，mineCF と systemCF 以外のクラスを使うときだけクラスファミリ名の修飾をすればよい．

Commodity クラスと Sales クラスの ODQL での定義は以下のようになる．

```
/* Commodity クラス */
defineClass    Commodity
super:Composite
description:"Class Commodity"
{
 instance:
  Integer       commodityNo     unique: mandatory:;
  String        commodityName;
  Integer       unitPrice;
};

/* Sales クラス */
defineClass    Sales
super:Composite
description:"Class Sales"
{
 instance:
  Integer       salesNo         unique: mandatory:;
  Integer       soldCommodityNo;
  Integer       purchaserNo;
  Integer       quantity;
  Integer       amountSold;
};
```

3.2 クラスの構築

以上でクラス定義が出来るので，後はクラスを構築すればデータ定義は終了である．クラスの構築は，ODQL コマンド buildClass を用いる．以下にクラスファミリ mineCF に Customer クラス，Commodity クラス，Sales クラスを構築する方法を示す.

```
> buildClass mineCF::Customer;
> buildClass mineCF::Commodity;
> buildClass mineCF::Sales;
```

ここでは，クラスファミリの修飾を行っているが，以前述べたように defaultCF コマンドを用いてデフォルトのクラスファミリを指定しておけば，クラスファミリの修飾はいらない．

3.3 クラス情報の表示

以上でクラスが構築できたわけであるが，クラスが構築できているか確かめるためには，以下のようにして構築されているクラスの情報を表示することができる．

```
> className.print();
```

3.4 クラスの削除

1 度構築したクラスを削除するには以下のようにする．

```
> className.delete();
```

この操作により，クラス名が className であるクラスが削除される．

3.5 エラーの対処法

クラスを定義しているとエラーがでる場合がある．エラーがでるとエラーの場所の近くを表示するので，その周辺からコードのミスなどを探すことができる．エラーは日本語で出してくれるので分かりやすいと思う．

4 実習

今日は，自分のクラスファミリに Customer クラス，Commody クラス，Sales クラスを定義，構築して下さい．今日この文章にかいた ODQL 文を実行すればクラスの定義，構築はできます．