データ操作

平成 13 年 6 月 19 日
峯 肇史

1　データ操作

今回は，データベース中のデータに対する操作を行う．データベースにデータを追加したり，データベース中のデータを変更，削除するには，DBMS が提供しているデータベース言語を使用しなければならない．今回，実習で使用する DBMS Jasmine では，ODQL という言語を用いる．以下のような情報をデータベースに格納するとして Jasmine での ODQL を用いたデータ操作について説明する．

得意先 (Customer)

得意先コード (customerNo)	得意先名 (customerName)	住所 (customerAddress)
1	Astore	Fukuoka
2	Bstore	Tokyo
⋮	⋮	⋮

商品 (Commodity)

商品コード (commodityNo)	商品名 (commodityName)	単価 [円](unitPrice)
1	Apple	200
2	Orange	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
⋮	⋮	⋮	⋮	⋮

以下の説明では，上記の 3 つのクラス，Customer, Commodity, Sales が構築済であることを前提とする．

2　リテラルとエンティティクラス

Jasmine にはリテラルとエンティティという 2 種類のクラス (オブジェクト) がある．データ操作の説明をする前に，これら 2 つについての簡単な説明を行う．

2.1 リテラルクラス

リテラルとは数字や文字のような値を指す．Jasmine では 2 つの種類のリテラルが提供されている．1 つはアトミッククラスであり，これは Integer や Decimal などの構造を持たないデータ型である．もう 1 つはタプルクラスであり，これは複数のメンバから構成されるような構造を持つデータ型である．リテラルオブジェクトは，データ属性やオブジェクト変数の値として取り扱われる．

以下に Jasmine での代表的なリテラルを示す．

- Boolean
  TRUE か FALSE のどちらか一方をとる．
- Date
  月，日，年を含む日付データ．
- Integer
  数量を表す数値全般 (整数)
- String
  文字列データ
- Bag<T>
  型 T のオブジェクト集合．

2.2 エンティティクラス

エンティティは，リテラル以外のオブジェクトを指す．エンティティは，オブジェクト識別子によって識別される．エンティティオブジェクトは明示的に生成された時点 (主にメソッド new() を使う) から，明示的に削除されるまで (主にメソッド delete() を使う) データベースに存在する．ユーザ定義クラスのオブジェクトはエンティティである．

3 データの追加

ユーザ定義クラスのオブジェクトをデータベースに追加するには，クラス手続き属性 new() を用いる．Customer クラスのオブジェクトとして customerNo が 1，customerName が "Astore"，customerAddress が "Fukuoka" であるようなオブジェクトを追加する場合は，インタプリタで以下のように入力する．

```
> Customer.new(customerNo:=1, customerName:="Astore", customerAddress:="Fukuoka");
```

これでデータベースにオブジェクトが追加される．データベースに追加されたのを確認するためには，問い合わせを用いる．以下のようにすると Customer クラスのすべてのオブジェクトの値を表示する．

```
1. > Bag<Customer> cus;
2. > cus = Customer from Customer;
3. > cus.print();
```

まず 1．で問い合わせの結果の Customer クラスのオブジェクト集合を格納する変数 cus を宣言している．2．の問い合わせは，Customer クラスのすべてのオブジェクトを cus 集合に格納する．where 句がないためにすべてのオブジェクトが格納される．3．で cus の内容を表示している．先ほど登録した Customer クラスのオブジェクトのみが登録されていれば，3．の実行結果は以下のようになる．

```
jasmine(mineCF) > cus.print();
Bag{
    mineCF::Customer::1 {
        customerNo = 1,
        customerName = "Astore",
        customerAddress = "Fukuoka"
    }
}
```

次に，Commodity クラスのオブジェクトを追加してみる．今回はメソッド new() の返り値 (オブジェクトへの参照) を利用して，値を追加してみる．commodityNo が 1，commodityName が "Apple"，unitPrice が 200 円であるような Commodity クラスのオブジェクトをデータベースに追加するとする．以下のようにインタプリタで入力する．

1. > Commodity com;
2. > com = Commodity.new(commodityNo:=1);
3. > com.print()

1．で新しく作成するオブジェクトのオブジェクト変数を宣言している．2．で commodityNo が 1 の Commodity クラスのオブジェクトを作成している．3. で作成したオブジェクトの情報を表示している．3．を行った結果は以下のようになる．

```
jasmine(mineCF) > com.print();
mineCF::Commodity::1 {
    commodityNo = 1,
    commodityName = NIL,
    unitPrice = NIL
}
```

上の表示結果よりわかるように，commodityNo にだけ値が入ったオブジェクトが生成されている．Jasmine での NIL というのは，値がないことを示す．データベースに新たに生成されたオブジェクトのデータ属性には，明示的に初期値が設定されない限り NIL が設定されるようになっている．ここで一つ注意点であるがクラス定義をしたときに属性値に mandatory と指定してあるところは，NIL を許さないので，new を使用してオブジェクトを生成する場合に指定するか，もしくはクラス定義のときに default(新しく生成されたオブジェクトのデータ属性に明示的に値が指定しなかった場合の値を指定する記述) を記述する必要がある．Commodity クラスでは commodityNo がこの例である．

それでは，このオブジェクトの commodityName と unitPrice に値を代入する．以下のようにインタプリタに入力すると値が代入できる．

1. > com.commodityName = "Apple";

2. > com.unitPrice = 200;

3. > com.print();

1. で オブジェクト com の commodityName を "Apple"にしている．2. で com の unitPrice を 200 にしている．3. で com の情報を表示している．表示結果は以下のようになる．

```
jasmine(mineCF) > com.print();
mineCF::Commodity::1 {
    commodityNo = 1,
    commodityName = "Apple",
    unitPrice = 200
}
```

以上のように，データの追加は クラス手続き属性 new() を用いることによりデータベースに追加される．

4 データの変更

以下のような Commodity クラスのオブジェクトが定義されているとする．

Commodity

(commodityNo)	(commodityName)	(unitPrice)
1	Apple	200
2	Orange	100

上のデータの Orange の単価 (unitPrice) を 150 に変更することを例にとって説明する．データを変更するには，変更すべきオブジェクトをデータベース中から見つけ出さなければならない．そこで以下のようにしてオブジェクトを検索する．

1. > Commodity com;

2. > com = Commodity from Commodity where Commodity.commodityName == "Orange";

3. > com.print();

1. でオブジェクト変数 com を宣言している．2. で Commodity クラスのオブジェクトの内 commodityName が"Apple"であるオブジェクトを検索してそのオブジェクトへの参照を com に代入している．3. での表示結果は以下のようになる．

```
jasmine(mineCF) > com.print();
mineCF::Commodity::2 {
    commodityNo = 2,
    commodityName = "Orange",
    unitPrice = 100
}
```

変更すべきオブジェクトが見つかったのでそのオブジェクトのデータ属性の値を変更する．今回は，単価を 150 円に変更するので，com の unitPrice を 150 にすればよい．以下のようにする．

1. > com.unitPrice = 150;
2. > com.print();

1. でオブジェクト com の unitPrice を 150 にしている．2. の表示結果は以下のようになる．unitPrice が 150 に変更されている．

```
jasmine(mineCF) > com.print();
mineCF::Commodity::2 {
    commodityNo = 2,
    commodityName = "Orange",
    unitPrice = 150
}
```

以上のようにデータの変更は，まず，変更すべきオブジェクトを見つけてそのオブジェクトに対して変更操作を行わなければならない．今回例で挙げているデータ例は，データ数が少なく，かつデータベースに格納されているデータも分かっているので，簡単な検索により見つかるが，一般的にはユーザが意図するデータを見つけ出すのは簡単ではない．今回は，これらの検索についてこれ以上の例は挙げないが，効率的な検索法についてはこの後の回で説明されるであろう．

5 データの削除

データの削除は，インスタンス手続き属性 delete() を用いる．以下のような Commodity クラスのオブジェクトがデータベース定義されているする．

Commodity

(commodityNo)	(commodityName)	(unitPrice)
1	Apple	200
2	Orange	150

上の表の Apple のデータを削除することを例にして説明する．データの変更と同様に，データの削除もまず削除すべきオブジェクトをデータベースから検索しなくてはならない．まず現在のデータを表示してみると．

1. > Bag<Commodity> comBag;
2. > comBag = Commodity from Commodity;
3. > comBag.print();
   (表示)

```
Bag{
    mineCF::Commodity::1 {
        commodityNo = 1,
        commodityName = "Apple",
```

```
        unitPrice = 200
    },
    mineCF::Commodity::2 {
        commodityNo = 2,
        commodityName = "Orange",
        unitPrice = 150
    }
}
```

それでは，commoidyNo が 1 であるデータを削除する．以下のようにして削除する．

1. > Commodity com;
2. > com = Commodity from Commodity where Commodity.commodityNo == 1;
3. > com.delete();

2．で削除すべきオブジェクトを検索して，3．で delete() を用いてオブジェクトを削除している．削除されているか確かめるには以下のようにする．確かに commodityNo が 1 であるデータが削除されている．

1. > Bag<Commodity> comBag;
2. > comBag = Commodity from Commodity;
3. > comBag.print();
   (表示)

```
Bag{
    mineCF::Commodity::2 {
        commodityNo = 2,
        commodityName = ‘‘Orange’’,
        unitPrice = 150
    }
}
```

これまで，あるクラスのオブジェクトを 1 つ削除しする例を挙げたが，あるクラスのオブジェクトすべてを削除するには以下の 2 つの方法がある．Commodity クラスの全オブジェクトを削除することを例に説明する．

方法 1 以下のようにする

1. > Bag<Commodity> comBag;
2. > Commodity com;
3. > comBag = Commodity form Commodity;
4. > scan(comBag,com){com.delete();};

1．2．は変数宣言である．3．で Commodity クラスのすべてのオブジェクトを comBag 集合に格納している．4．の scan の文は，comBag 集合の要素を 1 つずつ取りだし，com 変数に代入し, 括弧の中の処理を行うという操作である．よってすべての comBag の要素 (Commodity クラスの全オブジェクト) に対して delete() メソッドが呼び出しているので，Commodity クラスのすべてのオブジェクトが削除される．

方法 2 方法 2 では，クラス自体を削除する方法である．以下のようにする．

1. > Commodity.delete();

クラス属性メソッド delete() を呼び出して Commodity クラス自体を削除している．この時同時に Commodity クラスのすべてのオブジェクトも削除される．注意点として，この方法をとるとクラス定義自体が削除されるので，もう一度 Commodity クラスのデータを追加しようとすると，Commodity クラスの定義，構築を再び行わなければならない．

6 実習

今日の実習では，データ操作を実際に行ってもらいます．以下の手順で実習を行って下さい．

1. インタプリタを起動．
2. クラスファミリの修飾をする必要がないようにデフォルトのクラスファミリを自分のクラスファミリとする．
   > defaultCF accountCF;
3. 「問い合わせ」の回の輪講で定義したオブジェクトのデータが残っている人は，データの削除のところで説明した，クラスの全オブジェクトを削除する方法を用いてデータを削除．
4. 1 ページ目に書いてある 3 つの表のデータを，自分のクラスファミリのクラスオブジェクトとして追加．
5. Commodity クラスの commodityName が ”Apple”であるオブジェクトの unitPrice を 150 に変更．
6. Commodity クラスの commodityName が ”Orange”であるオブジェクトを削除．