データベースシステム

久保正明

6/14/2001

1　データベースとは

コンピュータのエンジニアでなくとも、データベースという言葉を聞いた事はあるでしょう。また気付かないうちにほとんどの人が日常的に利用していることでしょう。例えば預金残高を間違いなく記憶してくれている銀行口座は巨大なデータベースですし、電話番号案内サービスでオペレータが即座に電話番号を調べてくれるのも、データベースを使っているからです。このようにコンピュータによって沢山のデータを管理し、簡単に目的のデータを探し出すことのできる仕組みをデータベースと言います。

今日、コンピュータが行っている業務のうち、ほとんどのものは、このようなデータ管理を持ち合わせたものになります。そのため、必然の流れとして、このようなソフトウェアの開発を容易にするため、汎用的にデータの管理を専門に行う支援システムが生まれました。それがデータベース管理システム（Database Management System: 以下 DBMS）と呼ばれるものです。

DBMS を用いることにより、ファイル構造設計、排他制御、トランザクション処理、検索、と言った極めて面倒でかつ不具合の入り込みやすい部分を肩代わりしてくれます。そのため、ソフトウェア開発者は、管理すべきデータの定義やユーザーインタフェースと言った、本来の目的の部分に集中することができます。データベース管理システムを使うことにより、少なくともすべてを独自で開発するのよりは、遥かに大きな信頼性を確保（もちろん開発工数の削減も）することができます。

2　トランザクション

データベースの果たす機能のひとつに「データの原子性」があります。これは、データが常に完全な状態で存在するか、またはまったく存在しないかのどちらかの状態であることを保証するものです。

たとえば、旅行の予約システムがあったとしましょう。ツアーに参加するには、「飛行機の予約」「バスの予約」「ホテルの予約」をすべて完了しなければなりません。あるとき、「飛行機の予約」「バスの予約」は完了したものの、「ホテルの予約」だけは満室のため失敗してしまいました。このときに、データベース上には「飛行機とバスだけが予約された不完全なツアーの予約情報」が存在していることになります。しかも、すでに「飛行機の予約」と「バスの予約」は完了してしまったため、解約手続きを実行しなければなりません。このような、不完全なデータの存在を防ぐことができる機構がトランザクション管理と呼ばれています。

トランザクション管理の考え方を、前述の旅行の予約システムに当てはめてみましょう。つまり、「飛行機の予約」「バスの予約」「ホテルの予約」のすべてが完了するまで、それらを確定しないようにするのです。この確定処理のことをコミットと呼びます。

もし、何らかの原因からトランザクションが正常に終了しなかった場合、データベースの状態をトランザクションが始まる前に戻さなければなりません。この処理のことをロールバックと呼びます。

3 Jasmine

今回の実習にはJasmineを使用してもらいます。Jasmineはオブジェクト指向データベースであり、研究室でも頻繁に利用されています。これから行う手順をふまえれば研究室のデータベースをすべて操作できます。操作には細心の注意を払ってください。今回つくってもらうデータベースは最も基本的なリレーショナルデータベースです。

3.1 使い方

ログインします。

```
% ssh jasmine
```

Jasmineで開発ができるように自分のログイン名を

```
/etc/group
```

に書き込みます。さらにシェルをbashにし、環境ファイルを読み込みます。bashを使用しているというのは特にこだわりがあるわけではなく、Jasmineのマニュアルがbashを使用しているためです。環境ファイル.profileを自分で書き換えることができればtcsh等のシェルの使用も可能です。

```
% bash
% source /jasmine/jasmine2/.profile
```

これでJasmineによる開発ができるようになりました。次回からはログイン後シェルをbashに変え、環境ファイルを読み込むだけで良いです。Jasmineでの操作は基本的にインタプリタ上で行います。インタプリタの起動は

```
% codqlie
```

終るときは

```
> end;
```

です。

3.2 ストア

ストアとはJasmineにおけるデータの入れ物のことです。ストアには定義情報(クラス、データ属性、手続き属性の定義)とユーザデータの両方が格納されます。ストアには4種類あり、実際に触ることができるのはユーザストアとシステムストアの2種類のみになります。

- システムストア
  Jasmineのシステムクラスを格納します。ユーザが利用する基本的なリテラル、手続きなどが入っています。
- ユーザストア
  ユーザが作成したものを格納するストア。
- ワークストア
  Jasmineのセション中の情報を一時的に格納するストア。
- トランザクションストア1つのトランザクション内で使われる一時的な集合を格納するストア。

3.3 クラスファミリ

クラスの集合の単位。一つのストアに対して複数のクラスファミリを作成します。クラスファミリ名が異なれば同じストア内に同名のクラスが存在することが可能です。通常、複数人で開発を行う場合一人に対して一つのクラスファミリを作ります。クラスファミリという概念のメリットはそこにあります。

3.4 クラスファミリの作成

今回は自分専用のクラスファミリを作成してもらいます。

```
% createcf login_nameCF rinkou
```

例えばログイン名が kubo だったら kuboCF といった具合いになります。これからデータベースを作成する場合は自分で作ったクラスファミリの中で作業を行ってください。

3.5 データベースの作成

今回はインタプリタの基本的な使い方のみをします。サンプルファイル (/u/kubo/jasmine/setClasses.odql) があるのでこれを自分のディレクトリの適当なところにコピーしてそのディレクトリに入ってください。そのあと

```
% codqlie
> defaultCF login_nameCF;
> execFile file_name;
```

とするとインタプリタでファイルを読み込みます。ファイルの中身をいかにかいておきます。

```
 defineClass Customer
 super:Composite
 description:"Class Customer "
{
 instance:
  Integer customerNo unique: mandatory:;
  String customerName;
  String  customerAddress;
  Void addCustomer(Integer i,String n,String a);
  Void deleteCustomer(Integer dn);
 };

 defineClass  Commodity
 super:Composite
 description:"Class Commodity"
{
 instance:
  Integer commodityNo unique: mandatory:;
  String  commodityName;
  Integer unitPrice;
```

```
  String  commodityAddress;
  Void addCommodity(Integer i,String n,Integer u,String a);
  Void deleteCommodity(Integer dn);
};


 defineClass Sales
 super:Composite
 description:"Class Sales"
 {
 instance:
  Integer salesNo unique: mandatory:;
  Integer purchaserNo    unique:;
  Integer  soldCommodityNo unique:;
  Integer quantity;
  Integer amountSold;
  Void addSales(Integer n, Integer pn, Integer sn,Integer q, Integer am);
  Void         deleteSales(Integer dn);
};
 buildClass Customer Commodity Sales;
```

ここで作成したクラスは

- Customer
- Commodity
- Sales

の 3 つです。内容を確認するには

```
> Customer.print();
```

などとすれば良いです。作ったクラスに値を入れてみましょう。

```
> Customer c;
> c=Customer.new(customerNo:=1);
> c.customerName="Sample_name";
```

これでデータベースに登録されました。とりあえず確認のため

```
> c.print();
```

としてみましょう。登録された内容が出力されます。Jasmine でデータベース中のオブジェクトを出力するには基本的に変数に代入して print() メソッドを使用します。print() メソッドは Jasmine で定義されているメソッドです。 例えば

```
> Customer d;
> d=Customer from Customer where Customer.customerNo==1;
> d.print();
```

で見ることができます。変数はインタプリタを終了するまで有効なので途中で消す場合は

```
> d.delete();
```

とします。もし変数がアトミックリテラルの場合は

```
> String s;
> undefVar s;
```

とすると消えますし、単に undefVar だけですべて消すこともできます。インタプリタを終了する場合は

```
> end;
```

です。これで一通りの使い方になります。今日は以上の作業を行ってください。

4 参考

4.1 ストアの作成方法

今回の実習ではストアを作成することはないですし、これからもないかも知れませんが作成手順について触れておきます。

```
% createStore -numberOfPages 2000 -pageSize 8 store_name /jasmine/jasmine2/jasmine/data/store_name
```

削除するときは

```
% deleteStore store_name
```

です。みなさんが使用するシステムには他に重要なデータベースもあるので、もしこれらの作業をする場合には注意してください。