SQLStatement Class

Sql文を楽に生成するライブラリ

DatabaseにはSQLだ

SQL とは Structured Query Language(構造化問合せ言語) のことで、データベースの定義や操作などを実現するためのデータベース言語の一つです。リレーショナル型データベースに対応したデータベース言語として ISO 及び JIS において規格化されており、現在ではリレーショナル型データベースの事実上の標準として位置づけられています。

（http://www.techscore.com/tech/sql/02_02.html）

DatabaseはSQLで操作する

- SQLで書いたコマンドをデータベースに送ることで、データベースのほとんどすべての操作ができます。
- 各データベースにはsqlをおくるコマンドが用意されています。
- CやPerlやJavaなどのプログラムからもSQLを使ってデータベースの操作、問い合わせを行います。

SQLの例（テーブルの生成）

```
create table tu_data (
  id    integer  not null, -- メンバーid
  aid   integer  not null, -- 属性id
  cdate datetime not null, -- 生成日
  data  text     not null  -- データ
);
```

SQLの例（読み出し）

- データをid,aidの順に読み出し

```
select id,aid,data,cdate from tu_data
        order by id,aid;
```

- idとaidを指定してデータを読み出し

```
select  data from tu_data
        where id=13 and aid=3；
```

SQLの例（挿入、更新）

- 新規データの追加

```
insert into tu_data(id,aid,cdate,data)
    values ( 1001, 3, 'now', 'テストデータ);
```

- 既存データの更新

```
update tu_data set data='変更されたデータ,
     cdate='2004:08:21' where id=1001 and aid=3;
```

SQLの例（削除）

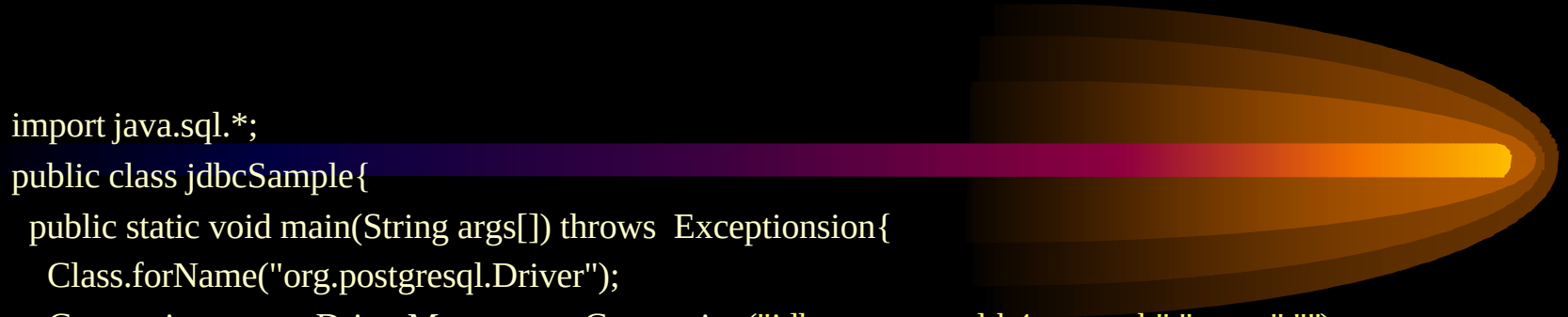
- 全データを削除

  delete tu_data;

- id=1001のデータだけ削除

  delete tu_data where id=1001;

Javaから使用する例



```
import java.sql.*;
public class jdbcSample{
  public static void main(String args[]) throws  Exceptionsion{
    Class.forName("org.postgresql.Driver");
    Connection conn= DriverManager.getConnection("jdbc:postgresql:k4mempub","www","");
    Statement st=conn.createStatement();
    String sql="select id,aid,data,cdate from tu_data order by id,aid"
    ResultSet rs = st.executeQuery(sql);
    while(rs.next()){
      int id  = rs.getInt(1);
      int aid = rs.getInt(2);
      String data = rs.getInt(3);
      Date cdate = rs.getDate(4);
      System.out.println("id="+id+" aid="+aid+ " data=" + data + " cdate" + cdate);
     }
     rs.close();
     st.close()
     conn.close();
  }
}
```

プログラム中でSQL文を書くと

- いろいろいやなことがある。
  - 引用符が入り乱れて書きにくい、読みにくい
  - 引用符が要ったり要らなかったり
  - 特殊文字の処理も面倒
  - insertではfield名と値が遠くて面倒
  - insert文とupdate文で文法違いすぎ
  - whereの条件が無い場合、1つある場合、2つ以上ある場合の処理が面倒
  - Stringに値を付け加えていくのは非効率？

引用符の入り乱れ

- idとaidを指定してデータを取得する例：

```
String sql = "select date from tu_data" +
        " where id=10 and aid=5";
```

- idとaidに変数の値を使用する例：

```
String sql = "select data from tu_data" +
      " where id=" + id + " and aid=" + aid;
```

引用符の入り乱れ（続き）

- 文字列型を使用するには引用符が必要

```
String sql = "select id,aid from tu_data" +
      " where data='" + data + "'" +
      " and cdate < '" + cdate "'";
```

だいぶ見難くなってくる。

特殊文字のエスケープ

- 文字列に特殊文字があれば前に¥を挿入する必要がある。

元の文字列　　what's michel?

=> SQLでの文字列　'what¥'s michel?'

=> Javaでの文字列　"'what¥¥'s michel?'"

特殊文字のエスケープ(続き)

- 前の例にエスケープ処理を追加

  Util.escape()というメソッド（関数）でエスケープ処理ができるとする。

```
String sql = "select id,aid from tu_data" +
    " where data='" + Util.escape(data) + "'" +
    " and cdate < '" + cdate "'";
```

さらにSQL文が書きにくいし、読みにくい。

解決策の検討

- SQLを使わない？
  - SQL部分を隠してくれるライブラリ（Torque）もあるが、それはそれで使うのが面倒。
  - 面倒な面もあるがSQLは便利
- 自前で便利なライブラリを作る
  - SQL文の各情報を構造的に保存し、文字列で出力してやれば、もっとわかりやすく設定できて、使いやすそうだ。

製作

- アイデアはあったがなかなか製作にいたらず、面倒なSQL文を我慢して書いていた。
- ある日、突然書き始める。
  - JUnitのテストコードを並行して書きながら3時間程度で完成、テスト完。
- 使い始めるととても便利
  - 既に書いたSQL文生成部もこれで書き直す。
  - 別のプロジェクトにも持ち込み使用

SqlStatementクラス

- SQL文生成ライブラリを
  - SqlStatementクラスとして作成
  - 方針
    - 簡単なSQL文のみ生成(select,insert,update,delete)
    - 使いやすいインターフェース
    - 特殊文字のエスケープ機能
    - 複数DB対応は当面考えない
    - 簡単に作る

SqlStatement メンバー変数

```
public class SqlStatement {
   private int iStatementType;  // select or insert ..
   private String sTable;           // テーブル名
   private LinkedList lField;    //  フィールドの リスト
   private LinkedList lValue;   // 値の リスト
   private LinkedList lWhere;   // where句の リスト
   private LinkedList lOrderBy; // order句の リスト
   private boolean bForUpdate;   // select for updateか
    ...
}
```

使用例（select)

使用前

```
String sql = "select id,aid from tu_data" +
    " where data='" + Util.escape(data) + "'" +
    " and cdate < '" + cdate "'";
```

使用後

```
SqlStatement sql = new SqlStatement();
sql.select("id,aid").from("tu_data");
sql.where("data=",data);
sql.where("cdate<",cdate);
ResultSet rs = conn.executeQuery(sql.toString());
```

使用例 (insert/update)

```
SqlStatement sql  = new SqlStatement();
sql.table("tu_data");
sql.set("data",data);
sql.set("cdate","now");
if(bNew){
    sql.set("id",id);
    sql.set("aid",aid);
    sql.insert();
} else {
    sql.where("id=",id);
    sql.where("aid=",aid);
    sql.update();
}
```

実現

引数の型で動作を切り替え

```
public SqlStatement set(String name, String value){
    lField.add(name);
    lValue.add(quote(value));
    return this;
}

public SqlStatement set(String name, int i){
    lField.add(name);
    lValue.add(Integer.toString(i));
    return this;
}
```

Oracle対応機能（1）

Postgresqlでは以下でcdateに現在時刻が入る

```
insert (id,aid,data,cdate) values (100,2,'foo','now');
```

Oracleでは以下のようにする。

```
insert(id,aid,data,cdate) values(100,2,'foo', SYSDATE);
```

第2引数に引用符をつけない

set_nq(String,String)メソッドを追加。

Oracle対応機能（2）

Timestamp型の値の設定

```
public SqlStatement set_ts(String name,Timestamp ts){
  lField.add(name);
  lValue.add("TO_TIMESTAMP(" + ts +
      ",'YYYY/MM/DD HH24:MI:SS'");
  return this;
}
```

感想

- javaでも必要なライブラリは自分で作っていった方が良さそうだ。
  - javaはライブラリーが豊富だが探すのが大変。
  - 自作ライブラリーの拡充は楽しい作業
- Junitは便利
  - 書いたコードの動作をすぐに確認
  - 改造時のregression test

今後

- いろいろ使っていきながら、徐々に必要な機能を追加して行きたい。
- javadocも一度ちゃんと書いてみたい。
- staticなFactoryメソッドを作れば1行減らせそう。
  - SqlStatement sql =
    SqlStatement.makeSelect().from("tu_data");