Realizzazione di un driver JDBC per l'accesso ad un database scritto in prolog

di Stefano Tamagnini
Matricola 0000273180

Corso : Linguaggi e Modelli Computazionali 2006/07

last update : 26 giugno 2009

Indice

1 Introduzione

Come progetto per l'esame di Linguaggi e Modelli Computazionali, si è richiesto la relizzazione di un driver JDBC che permettesse di poter eseguire query SQL su un database Prolog. Della sintassi SQL che è molto vasta sono richieste le funzionalità di base che permettano basilarmente : Creation, Read, Update, Delete (CRUD), lasciando l'implementazione completa secondo lo standard SQL ad un secondo momento.

Questo driver, ho deciso di chiamarlo **PJDBC**, dove la lettera P sta appunto per Prolog ossia il linguaggio in cui è espresso il database da interrogare; le parti coinvolte in questo progetto sono :

1. il database
2. il linguaggio con cui interrogarlo (sql)
3. il driver fisico che permetta l'interazione del database con le richieste dell'utente

2 Il database

Una definizione di database può essere **il database rappresenta un archivio di dati strutturati**; la struttura più diffusa in questo momento è quella chiamata *relazionale*, che si compone di tabelle e di relazioni tra di esse.

2.1 Database Prolog

Il database che si andrà ad interrogare attraverso il driver sviluppato avrà una struttura relazionale e sarà descritto attraverso una teoria prolog; le relazioni saranno espresse mediante l'uso di opportuni predicati. La tabella all'interno di questo database sarà rappresenta da un insieme di termini *compound* prolog che rappresenteranno le righe della tabella; le righe avranno come sintassi:

```
nomeTabella(col1,col2,.....,colN).
```

I termini per appartenere ad una tabella dovranno avere come nome il nome della tabella. Esempio:

```
ta(col1,col2,col3).
ta(col11,col21,col31).
```

questi due termini rappresentano due righe di una tabella;

```
ta(col1,col2,col3).
tb(col1,col2,col3).
```

questi due termini rappresentano due righe di due tabelle rispettivamente ta e tb;

```
ta(col1,col2,col3).
ta(col1,col2).
```

questi due termini rappresentano due righe della stessa tabella, il secondo è pero mancante di un campo, si suppone quindi che i campi mancanti siano settati a *NULL*; inoltre da notare che si assume che i campi mancanti siano sempre nelle ultime colonne e non in mezzo in quanto quest'ultimo scenario non sarebbe predicibile. Per esprimere che una colonna è non settata basta usare il valore *NULL* come valore, esempio :

```
ta(col1,col2,col3).
ta(col1,NULL,col2).
```

2.2 Il metabase

Per avere la completezza di informazioni tipiche di un moderno database relazionale è necessario avere delle ulteriori informazioni su come è strutturato il database ed in particolare sulla struttura delle tabelle; queste informazioni solitamente compongono quello che è chiamato *metabase* o *metadatabase*. Un possibile insieme di informazioni utili possono essere :

1. il nome da associare ad una colonna
2. il tipo di dato che contiene una colonna
3. la descrizione del contenuto di una colonna

questi sono solo alcuni possibili elementi. Nel nostro caso il *metabase* può essere descritto attraverso l'uso di una tabella speciale dal nome **metabase**; la sintassi sarà :

```
metabase(
        "nome tabella",
        "posizione della colonna tra gli argomenti",
        "nome della colonna",
        "tipo di dato contenuto",
        "descrizione"
).
```

Questa particolare tabella non è pero necessaria, il database sarà comuque consistente, ma la sua presenza facilita l'interazione dell'utente nello scrivere query sql. Un esempio :

```
metabase("ta",1,"colonna1","string","prima colonna").
```

Permetterà all'utente di usare il nome della colonna nelle clausole sql.

2.3 Tipi di dato

2.3.1 Dati primitivi

Il database essendo scritto in prolog avrà come tipi di dati primitivi gli stessi supportati dal prolog; la mappatura tra i tipi di dato primitivi Java , Prolog e quelli usati nel database è la seguente :

•Int, Long, Double, Floa

2.3.2 Strutture Dati

Le strutture dati principali supportate saranno :

Tabella 1: Mapping Java - Prolog - Database

Java	Prolog	Database
null	_	string(null)
int	int	int or number
long	long	long or number
short	int	int or number
byte	int	int or number
double	double	double or number
float	float	float or number
boolean	NON PERVENUTO	string(true,false)

Tabella 2: Database Structure - Prolog

	Term	Struct	Var	Number	Atom	Atomic	Compound	List
0	a1	true	-	-	true	true	-	-
1	1	-	-	true	-	true	-	-
2	[a,b]	true	-	-	-	-	true	true
3	p(a,b)	true	-	-	-	-	true	-
4	_	-	true	-	-	-	-	-
5	A	-	true	-	-	-	-	-

3 Structured Query Language

Lo Structured Query Language o SQL in forma abbreviata, è un linguaggio standard utilizzato per interrogare i database relazionali; allo stato attuale i moderni database implementano solo la parte *Entry Level* di tale standard.

3.1 Il parser

Per poter interpretare le richieste sql dell'utente si è reso necessario la creazione di un parser; in particolare dovrà essere in grado di poter riconoscere un sottoinsieme dello standard SQL/92. La sua realizzazione è avvenuta attraverso lo strumento javacc che ha permesso di specificare solo la grammatica da riconoscere e di lasciare allo strumento la generazione del codice relativa all'effettiva analisi del testo. La grammatica utilizzata per generare il parser comprende :

- **Operatori del linguaggio**, ossia le parole significative, che nel nostro caso possono essere di due tipi :
  - Comandi
  - Operatori SQL
- **Tipi di dato**

3.2 Comandi

I Comandi possibili sono suddivisi in due categorie Data Definition Language o DDL e Data Manipulation Language o DML. Quelli che ho ritenuto necessari implementare sono:

- DDL :
  - Create Database
  - Drop Database
- DML :
  - Select
  - Update
  - Insert
  - Delete
  - Truncate Table

- Drop Table

Come si nota non ho inserito *Create Table* in quanto già la *Insert* permette di creare una tabella all'interno del database prolog; ovviamente in questo caso mancano i metadati ma possono essere inseriti manualmente con delle *Insert* direttamente con riferimento alla tabella metabase.

3.3 Operatori

Gli operatori, messi a disposizione dal SQL/92 si dividono in quattro categorie:

1. Operatori di confronto
2. Operatori aritmetici
3. Operatori condizionali
4. Operatori logici

Di questi sono stati presi in considerazione quelli aritmetici e quelli di confronto

4 Il driver

Il driver è stato sviluppato secondo le specifiche *JDBC 4.0* (JSR 221) del Novembre 2006; secondo queste specifiche il driver è di tipo 4, ossia interamente scritto in java con accesso diretto alle risorse che compongono il database. Per aderire a queste specifiche è necessario che vengano implementate le seguenti interfacce :

- java.sql.Driver
- java.sql.DataSource
- java.sql.DatabaseMetaData
- java.sql.ParameterMetaData
- java.sql.ResultSetMetaData
- java.sql.Wrapper

e supportare nei ResultSet la capacita di concorrenza in lettura.

4.1 Struttura

Catalog è la cartella in cui si opera Schema è il nome del file .db su cui si opera Table è il funtore che si usa all'interno della teoria prolog

Nella cartella ci sono anche dei file .log(??) che servono per tenere traccia delle transizioni

4.2 Operazion

4.3 Transazioni