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 : 12 luglio 2009

Indice

1 Introduzione

Come progetto per l'esame di Linguaggi e Modelli Computazionali, ho realizzato un driver JDBC che permetta l'esecuzione di query SQL su un database scritto in linguaggio Prolog [Prolog]. Della sintassi SQL che è molto vasta ho realizzato le funzionalità di base che permettano : 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 jdbc che permetta l'interazione del database con le richieste dell'utente

2 Il database

Una definizione di database può essere presa da Wikipedia [Wikipedia] **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 sarà una teoria prolog, dove le tabelle verranno espresse come una lista di fatti (*facts*) dove ogni atomo della lista rappresenterà una riga di tale tabella. Il nome della tabella sarà il predicato mentre gli argomenti saranno i valori delle colonne. Per rappresentare una tabella di nome nomeTabella con N colonne (col1...colN) potrò usare la seguente sintassi:

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

Un caso particolare può essere:

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

questi due termini sono ambigui, non si sa cosa si voglia rappresentare; potrebbe essere che la col3 è null e quindi è stata omessa? oppure la seconda tabella è una tabella diversa dalla prima? per semplificare ho retenuto che nel caso si presenti una situazione del genere venga generata un'eccezione e si lasci all'utente decidere come disambinguare la situazione. Un modo per risolvere potrebbe essere quello usare il termine *null* come valore, esempio :

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

Il termine *null* ovviamente dovrà essere minuscolo altrimenti dal prolog verrebbe considerato una variabile.

2.1.1 Tipi di dato e Strutture dati

Il database così costruito ha come tipi di dati primitivi gli stessi supportati dal prolog; come strutture dati supporta gli Array in quanto sono facilmente rappresentabili in prolog come delle liste.

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 di sistema dal nome **mtable**; la sintassi sarà :

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

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 :

```
mtable("ta",1,"colonna1","string").
```

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

2.3 Store Procedure e Function

E' possibile pensare a far supportare al database prolog anche le *store procedure* e le *function* in quanto in una teoria prolog oltre ai fatti si possono inserire anche delle regole (*rules*). Il supporto a tale funzionalità è stato rimandato a estensioni future.

2.4 Struttura

Dal punto di vista strutturale il database è stato cosi strutturato :

- Catalog : è il contenitore di schema
- Schema : è quello che viene chiamato impropriamente database
- Table : è la tabella canonca

Sul filesystem il catalog è visto come una cartella, al cui interno ci sono gli schema sottoforma di file con l'estensione .db; invece per i database di sistema è riservato l'uso dell'estensione .dbs (Es. metabase.dbs) la cui presenza è però facoltativa. Il nome del database è il nome del file senza l'estensione, questo per ragioni di semplicità, mentre il nome del catalog è il nome della directory. L'utilizzo del catalog si è resa necessaria per poter , in futuro, dare supporto a query sql multi-schema, cosa ormai diffusa nell'uso di un database relazionale.

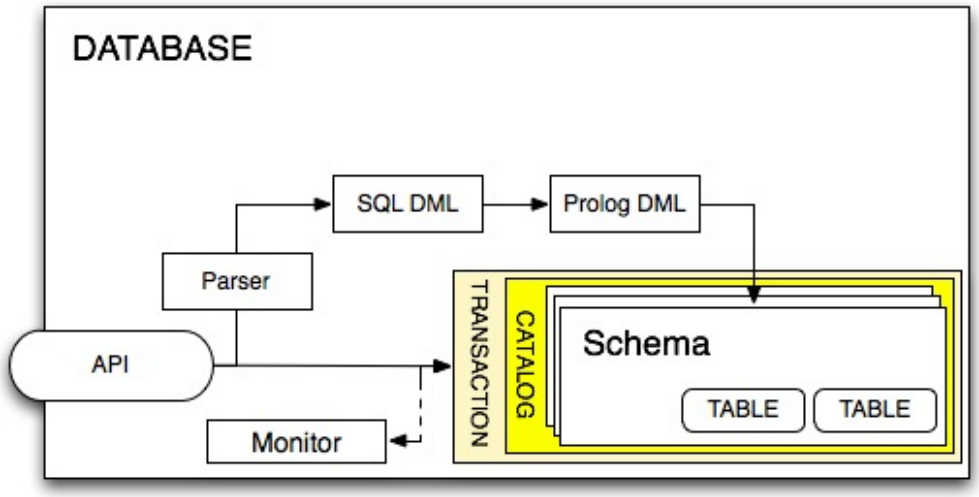

Figura 1: Database Structure

3 Structured Query Language

Lo Structured Query Language o SQL [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 [SQL92]. La sua realizzazione è avvenuta usando un tool chiamato javacc [javacc] della Sun che mi ha permesso di specificare solo la grammatica da riconoscere e di lasciare allo strumento la generazione del codice relativa all'effettiva analisi del testo. Ho trovato particolare difficoltà nella parte delle clausole **where**, quando ho dovuto risolvere la questione degli annidamenti; alla fine ho deciso di lasciarli da parte e di realizzare solo le clausole **where** dirette. La difficoltà degli annidamenti risiede non tanto nella grammatica quanto nella conversione in prolog. 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 (DDL) :
  - Create Database
  - Drop Database
- Data Manipulation Language (DML) :
  - Create Table
  - Select

- Update
- Insert
- Delete
- Drop Table

I vadi comandi sono pensati in modo da far compiere all'engine del prolog l'intera operazione sql richiesta facendo fare al java il solo ruolo di conversione sql alla sintassi prolog. Inoltre per sopperire all'eventuale mancanza di informazioni sul nome delle colonne è possibile specificare in ogni momento al posto del nome la posizione della colonna all'interno della tabella; per fare questo basta anteporre alla posizione il carattere speciale $. Questo è possibile in quanto al caricamento del database in memoria viene creato un metabase virtuale che fornisca le informazioni minime necessarie al funzionamento del sistema e che possa essere arricchito da informazioni presenti su un eventuale file contenente il metabase vero e proprio.

3.2.1 Select

Prendo come query di esempio :

```
Select $3 from nometabella where $2 = 1;
```

a questa query sql viene prima applicato un filtro per completare le parti sottointese :

```
Select [nomeschema.nometabella.$3]
from [nomeschema.nometabella]
where [nomeschema.nometabella.$2 = 1];
```

e poi viene trasformata in una query prolog :

```
nometabella(_,VAR2,VAR3),VAR2==1.
```

che verrà eseguita nella teoria prolog relativa allo schema selezionato.

3.2.2 Insert

Prendo come query di esempio :

```
insert into employee values (100,'paperino',1000);
```

verrà trasformata ,dopo gli opportuni filtri, nella seguente query prolog :

```
assert(employee(100,'paperino',1000)).
```

3.2.3 Delete

Prendo come query di esempio :

```
delete from prolog1.dept where dept.id = 1;
```

verrà trasformata ,dopo gli opportuni filtri, nella seguente query prolog :

```
dept(VAR0,VAR1),VAR1==1,retract(dept(VAR0,VAR1)).
```

3.2.4 Update

Prendo come query di esempio :

```
update prolog1.age SET age = 5 WHERE name = 'smith' ;
```

verrà trasformata ,dopo gli opportuni filtri, nella seguente query prolog :

```
age(VAR0,VAR1),
VAR0=='smith',
retract(age(VAR0,VAR1)),
assert(age(VAR0,5)).
```

3.2.5 Drop

Prendo come query di esempio :

```
drop table prolog1.employee;
```

verrà trasformata ,dopo gli opportuni filtri, nella seguente query prolog :

```
retract(employee(_,_,_)).
```

Questa particolare query oltre al normale comportamente lancia anche l'esecuzione di una query secondaria per rimuovere le informazioni di questa tabella sul 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 solo quelli di confronto, che sono solitamente i più utilizzati; questi operatori sono comunque tutti supportati dal prolog.

3.3.1 Operatori di confronto

1. ==
2. !=
3. >
4. <
5. >=
6. <=

che in prolog si trasformano nei seguenti

1. ==
2. \==
3. @>
4. @<
5. @>=
6. @<=

da non confonderli con gli operatori di confronto tra espressioni

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 ho parzialmente implementato le seguenti interfacce :

- java.sql.Driver (parziale)
- java.sql.DataSource (non implementata)
- java.sql.DatabaseMetaData (parziale)
- java.sql.ParameterMetaData (parziale)
- java.sql.ResultSetMetaData (parziale)
- java.sql.Wrapper (non implementata)

In particolare il supporto al *cursor* , alle transazioni , alle funzioni, al wrapping e alla creazione di strutture dati non è attualmente presente. Inoltre in Stantement non ho implementato la parte di PrepareStantement, quindi della figura sottostante [Fig.2] solo il ramo di sinistra è implementato e funzionante.

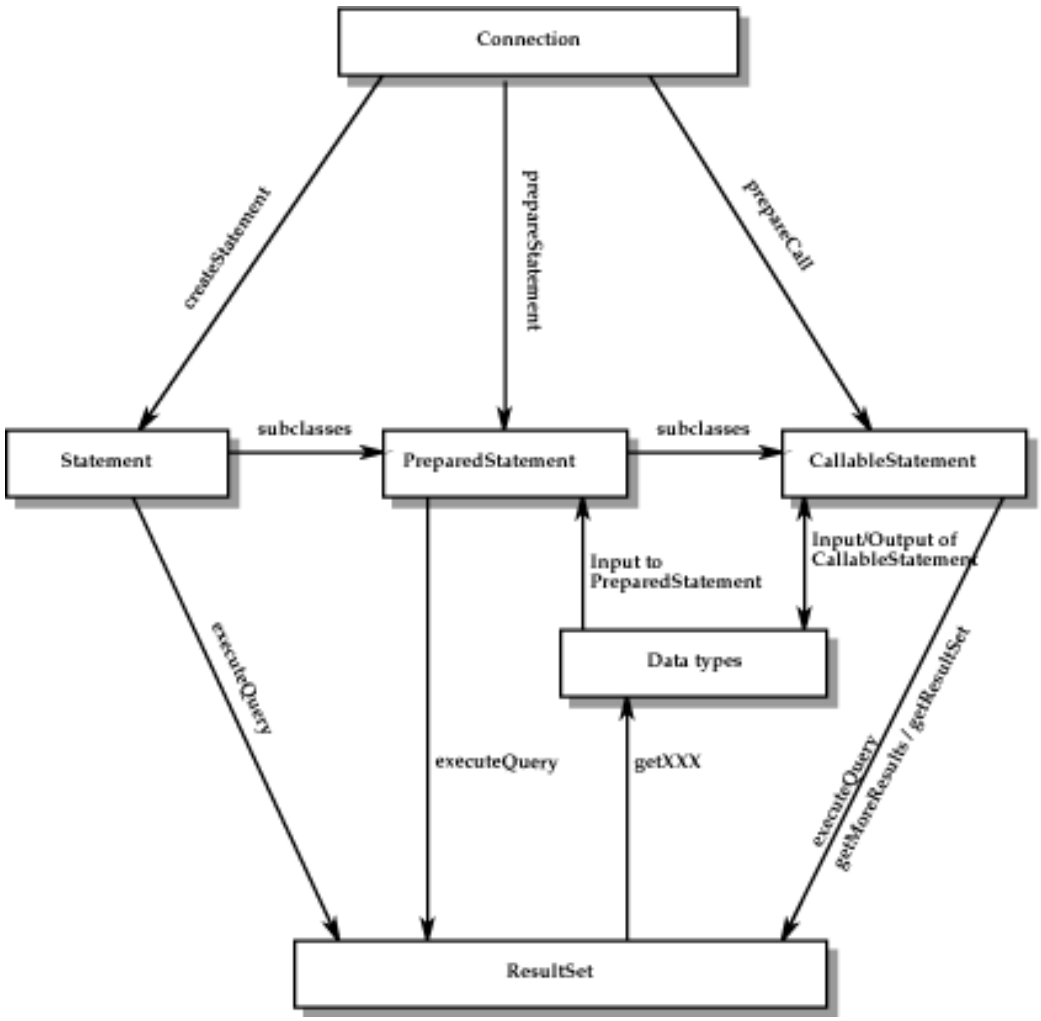

Figura 2: JDBC Class Relationship

5 Bibliografia

Prolog **Prolog** E' un linguaggio di programmazione che adotta il paradigma di programmazione logica.

Wikipedia **Wikipedia** è un sito internet il cui obiettivo è quello di creare una enciclopedia online collaborativa.

Javacc **Javacc** è uno strumento sviluppato da Sun per facilitare la creazione di parser - https://javacc.dev.java.net/

SQL92 **Sql/92** - http://www.contrib.andrew.cmu.edu/ shadow/sql/sql1992.txt