

Lezione 7: Associazioni

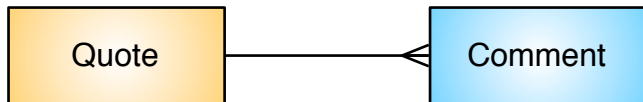
Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution – Non commercial – Share alike 2.5

Relazioni



A Quote has many Comments

A Comment belongs to one Quote

Relazioni in SQL

```
create table comments (  
  id      integer auto_increment,  
  quote_id integer,  
  body    text,  
  primary key(id)  
)
```

Oppure, usando le *migrations*

```
class CreateComments < ActiveRecord::Migration  
  def self.up  
    create_table :comments do |t|  
      t.integer :quote_id  
      t.text :body  
    end  
  end  
  ...  
end
```

Trovare i commenti di un motto

```
# select * from comments where quote_id = 123
```

```
comments = Comment.find(:all, :conditions => ['quote_id = ?', q.id])
```

```
class Quote < ActiveRecord::Base
```

```
  has_many :comments
```

```
end
```

```
comments = q.comments
```

Trovare il Quote a cui appartiene un Comment

```
quote    = Quote.find(c.quote_id)
```

```
class Comment < ActiveRecord::Base  
  belongs_to :quote  
end
```

```
quote    = c.quote
```

Riassumendo

```
create_table :quotes do |t|
  t.string :author
  t.text :body
end
```

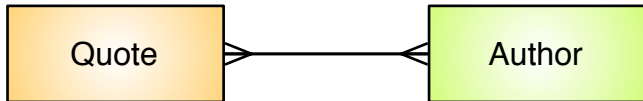
```
class Quote
  has_many :comments
end
```

```
quote = Quote.find(42)
comments = quote.comments
comments[0].quote # => the same quote
```

```
create_table :comments do |t|
  t.integer :quote_id
  t.text :body
end
```

```
class Comment
  belongs_to :quote
end
```

Relazioni multi-a-molti



Un autore ha uno o più motti

Un motto ha uno o più autori

```
create table authors (  
  id integer auto_increment,  
  name varchar(255),  
  primary key(id)  
)  
create table authors_quotes (  
  author_id integer,  
  quote_id integer,  
  primary key(author_id, quote_id)  
)
```

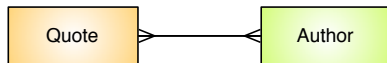
Trovare tutti i motti di un autore *a*

```
# select * from quotes where id in
#   (select quote_id from documents_quotes where author_id = 123)

class Author < ActiveRecord::Base
  has_and_belongs_to_many :quotes
end

quotes = a.quotes
```

Multi-a-molti \equiv 2 * uno-a-molti



```
create table authors (  
  id          integer auto_increment primary key,  
  name varchar(255)  
)  
create table authorships (  
  id          integer auto_increment primary key,  
  author_id integer,  
  quote_id  integer,  
  primary key(id)  
)
```

```
class Author < ActiveRecord::Base
  has_many :authorships
  has_many :quotes, :through => :authorships
end
```

```
class Authorship < ActiveRecord::Base
  belongs_to :author
  belongs_to :quote
end
```

```
class Quote < ActiveRecord::Base
  has_many :authorships
  has_many :authors, :through => :authorships
end
```

Attaccare attributi alla relazione

Un impiegato ricopre una posizione a partire da una certa data, fino a una certa data.

```
create table employees (  
  id            integer          auto_increment primary key,  
  first_name    varchar(255)    not null,  
  last_name     varchar(255)    not null  
);  
create table positions (  
  id            integer          auto_increment primary key,  
  name          varchar(255)    not null  
);  
create table position_assignments (  
  id            integer          auto_increment primary key,  
  position_id   integer          not null,  
  employee_id   integer          not null,  
  start_date    datetime         not null,  
  end_date      datetime         null  
);
```

```
class Employee < ActiveRecord::Base
  has_many :position_assignments
  has_many :positions, :through => :position_assignments
end
```

```
class Position < ActiveRecord::Base
  has_many :position_assignments
  has_many :employees, :through => :position_assignments
end
```

```
class PositionAssignment < ActiveRecord::Base
  belongs_to :position
  belongs_to :employee
end
```

```
emp.positions # => ["1o livello dal 01-03-1995 al 09-07-1998", ...]
```

Esercizio

Un'agenzia di viaggi vuole mettere online il proprio catalogo, che consiste di un insieme di **itinerari**. Gli itinerari devono essere ordinati per **Paese**, e per **continente**

Un itinerario può essere di confine, e toccare anche due o tre Paesi

Esercizio

In un'applicazione web vogliamo tenere traccia di utenti e gruppi a cui gli utenti appartengono. (Ciascun utente può appartenere a più di un gruppo.)

Disegnare il diagramma entità-relazioni, e il codice sql necessario per implementarlo. (In alternativa a sql scrivere il codice delle migrations.)

scrivere le classi ActiveRecord User e Group

Esercizio

Vogliamo realizzare un'applicazione per la gestione di una biblioteca. La biblioteca conserva una collezione di **documenti**. I documenti sono conservati in diverse **copie**, ciascuna delle quali può essere **in prestito** presso un diverso **utente**. Ciascun utente della biblioteca può avere **donato** una copia di un libro, e anche questo fatto deve essere rappresentato nel database.

Realizzare il modello dei dati per supportare tutte le funzionalità menzionate: modello entità-relazioni e codice SQL

Nell'applicazione del punto precedente: scrivere la classe Ruby per l'utente, con un metodo `find__prestiti()` che restituisce l'elenco dei libri che l'utente ha preso in prestito.

Dall'esame di stato per Ingegneri Informatici - Udine

24/6/2003 I

Progettare la base di dati per gestire un insieme di campionati di una lega sportiva di uno sport di squadra.

- ▶ esistono diversi campionati, in base alle età dei partecipanti (ad es. allievi, ragazzi, primavera, serie maggiore)
- ▶ Ogni campionato è organizzato con doppio girone all'italiana (con andata e ritorno a campi invertiti), ed il calendario delle partite è fissato all'inizio dell'anno, non modificabile
- ▶ Ciascuna squadra di ciascun campionato ha un insieme di giocatori che non cambiano durante la stagione
- ▶ Di ciascun giocatore interessano, oltre ai dati anagrafici, la carriera sportiva e l'ingaggio. Di ciascun giocatore è necessario memorizzare le eventuali squalifiche, con l'infrazione che le ha determinate e la partita in cui l'infrazione è stata commessa.

Dall'esame di stato per Ingegneri Informatici - Udine

24/6/2003 II

- ▶ Per ciascuna partita interessano i giocatori che hanno partecipato, le riserve, l'arbitro, oltre al risultato, i goal, ed eventuali note.

Le operazioni principali che si vogliono eseguire sono:

- ▶ Inserire tutti i risultati di una giornata di campionato.
- ▶ Verificare le presenze di un giocatore per verificare se ha rispettato le squalifiche
- ▶ Stilare la classifica dei 10 giocatori che hanno eseguito la maggior parte di goal nel loro campionato