

UNIVERSITÀ DEGLI STUDI DI FERRARA
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE
BASI DI DATI

Realizzazione Database per Social Network

Azzolini Damiano - Bertagnon Alessandro



UNIVERSITÀ
DEGLI STUDI
DI FERRARA
- EX LABORE FRUCTUS -

INDICE

1	Minimondo	1
1.1	Descrizione	1
1.2	Entità	1
1.2.1	Utente	1
1.2.2	Amicizia	2
1.2.3	Post	2
1.2.4	Commento	3
1.2.5	Media	3
1.2.6	Reazione	3
1.2.7	Notifica	4
1.2.8	Gruppo	4
1.3	Relazioni	4
1.4	Schema ER Completo	7
2	Da Modello ER a Modello Relazionale	8
3	Normalizzazione	9
4	Codice SQL	10
4.1	Introduzione	10
4.2	Codice	10
5	Query	12
6	Interfaccia	13

ELENCO DELLE FIGURE

1.1	Un utente riceve N notifiche ma una determinata notifica è ricevuta da un solo utente.	4
1.2	Un utente riceve N richieste di amicizia ma una determinata richiesta è ricevuta da un solo utente.	4
1.3	Un utente accetta N richieste di amicizia ma una determinata richiesta è accettata da un solo utente.	5
1.4	Una reazione scatena una notifica e una determinata notifica è scatenata da una reazione.	5
1.5	Una richiesta di amicizia genera una notifica e una determinata notifica è generata da una richiesta.	5
1.6	Un utente scrive n commenti ma un determinato commento può essere scritto solamente da un utente.	5
1.7	Un commento lancia una notifica e una determinata notifica può essere lanciata da un solo commento.	5
1.8	Un determinato commento può essere fatto solamente su un post ma un post può contenere n commenti.	5
1.9	Un utente crea n post ma un determinato post è scritto solamente da un utente.	5
1.10	Una reazione valuta un solo post ma un post può essere valutato da più reazioni.	5
1.11	Un utente può mettere n reazioni ma una reazione può essere messa da un solo utente.	6
1.12	Un post contiene n media ma un media può essere in un solo post.	6

CAPITOLO 1

MINIMONDO

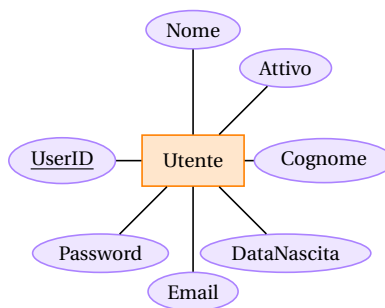
1.1 DESCRIZIONE

Il progetto di basa sullo sviluppo di un *social network*. Ciascun utente si può iscrivere al social network fornendo Nome, Cognome, e-mail e password e data di nascita. Al seguito dell'iscrizione, un utente può chiedere l'"amicizia" ad un altro utente, pubblicare un post, che può anche contenere media, che può essere commentato da altri utenti (solamente quelli tra gli amici dell'utente creatore del post). E possibile inoltre inserire, oltre a commenti, delle reazioni ai post. La scrittura di un commento o l'applicazione di una reazione ad un post viene segnalata con una notifica all'utente creatore del post, così come la richiesta di amicizia. Ciascuna entità utente, post, commento è caratterizzata da un campo attivo (flag booleano) che indica se, rispettivamente, l'utente si è cancellato dal social network (in questo caso attivo sarà fissato a 0), il post è stato cancellato o il commento è stato rimosso. I post che vengono cancellati dal social network rimangono comunque salvati nel database ma non visualizzati nella home. L'intero progetto è stato sviluppato utilizzando il framework laravel.

1.2 ENTITÀ

Di seguito vengono analizzate tutte le entità presenti nel database:

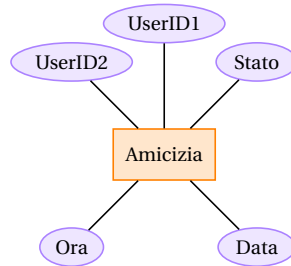
1.2.1 UTENTE



- *Nome*: nome dell'utente.
- *Cognome*: cognome dell'utente.
- *UserID*: identificativo numerico dell'utente e chiave primaria.
- *Password*: password dell'account.

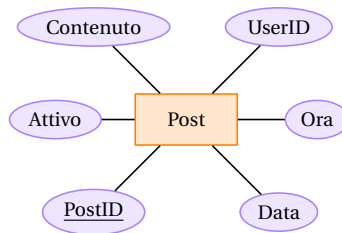
- *Email*: email dell'utente. Nel database non possono esistere due utenti diversi con la stessa mail.
- *DataNascita*: data di nascita dell'utente.
- *Attivo*: flag booleano, solitamente fissato a 0, 1 se l'utente si è cancellato dal social network.

1.2.2 AMICIZIA



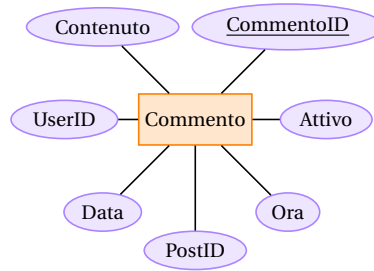
- *UserID1*: rappresenta l'identificativo dell'utente che richiede l'amicizia, chiave esterna.
- *UserID2*: rappresenta l'identificativo dell'utente che riceve la richiesta di amicizia, chiave esterna.
- *timestamp*: timestamp della richiesta dell'amicizia.
- *Stato*: variabile intera che rappresenta se l'amicizia è stata richiesta, accettata, in sospeso o rifiutata.

1.2.3 POST



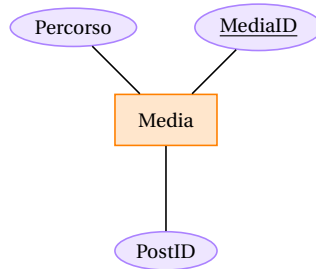
- *PostID1*: rappresenta l'identificativo del post, funge da chiave primaria.
- *Contenuto*: campo di testo che rappresenta il contenuto del post.
- *Data*: data di pubblicazione del post.
- *Ora*: ora di pubblicazione del post.
- *UserID*: chiave esterna, identificativo dell'utente autore del post.
- *Attivo*: flag booleano che indica se il post è attivo (1) o cancellato (0).

1.2.4 COMMENTO



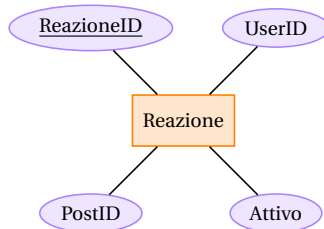
- *CommentoID*: identificativo numerico del commento e chiave primaria.
- *Contenuto*: contenuto del commento.
- *UserID*: identificativo dell'utente autore del commento, chiave esterna.
- *Data*: data del commento.
- *Ora*: ora del commento.
- *PostID*: identificativo numerico del post commentato, chiave esterna.
- *Attivo*: flag booleano che indica se il commento è attivo (1) o cancellato (0).

1.2.5 MEDIA



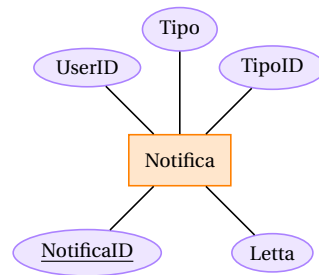
- *MediaID*: identificativo del media (foto, video, ...), chiave primaria.
- *Percorso*: link al media.
- *PostID*: identificativo del post nel quale il media viene inserito.

1.2.6 REAZIONE



- *ReazioneID*: identificativo numerico della reazione.
- *UserID*: identificativo dell'utente che mette la reazione.
- *PostID*: identificativo del post sul quale viene messa una reazione.
- *Attivo*: flag booleano che indica se la reazione è attiva (1) o è stata cancellata (0).

1.2.7 NOTIFICA



- *NotificaID*: identificativo numerico della notifica.
- *Tipo*:
- *TipoID*:
- *UserID*:
- *Letta*:

1.2.8 GRUPPO

Entità da aggiungere

1.3 RELAZIONI

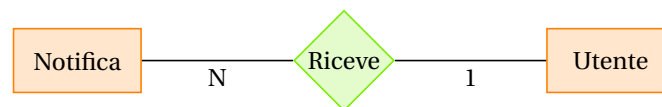


Figura 1.1: Un utente riceve N notifiche ma una determinata notifica è ricevuta da un solo utente.

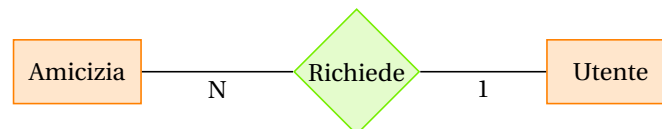


Figura 1.2: Un utente riceve N richieste di amicizia ma una determinata richiesta è ricevuta da un solo utente.

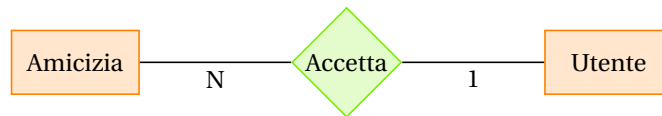


Figura 1.3: Un utente accetta N richieste di amicizia ma una determinata richiesta è accettata da un solo utente.

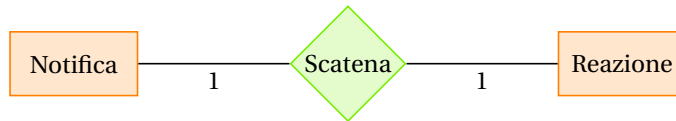


Figura 1.4: Una reazione scatena una notifica e una determinata notifica è scatenata da una reazione.

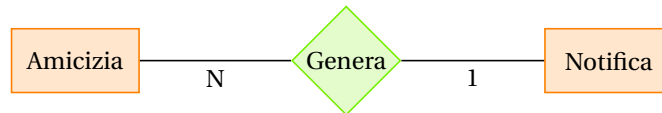


Figura 1.5: Una richiesta di amicizia genera una notifica e una determinata notifica è generata da una richiesta.



Figura 1.6: Un utente scrive n commenti ma un determinato commento può essere scritto solamente da un utente.

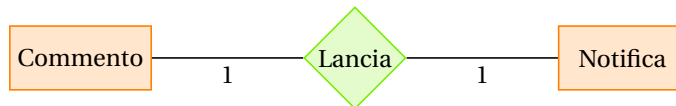


Figura 1.7: Un commento lancia una notifica e una determinata notifica può essere lanciata da un solo commento.

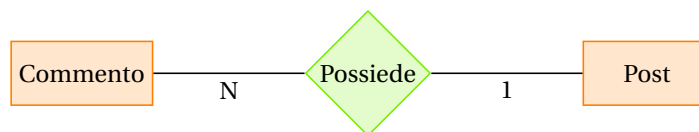


Figura 1.8: Un determinato commento può essere fatto solamente su un post ma un post può contenere n commenti.



Figura 1.9: Un utente crea n post ma un determinato post è scritto solamente da un utente.

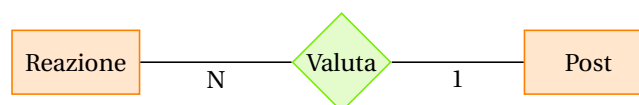


Figura 1.10: Una reazione valuta un solo post ma un post può essere valutato da più reazioni.

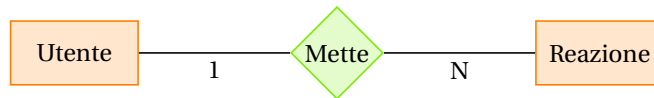


Figura 1.11: Un utente può mettere n reazioni ma una reazione può essere messa da un solo utente.

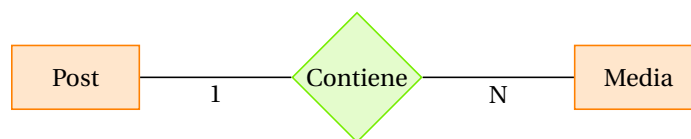


Figura 1.12: Un post contiene n media ma un media può essere in un solo post.

1.4 SCHEMA ER COMPLETO

Da inserire in una pagina nuova

CAPITOLO 2

DA MODELLO ER A MODELLO RELAZIONALE

CAPITOLO 3

NORMALIZZAZIONE

CAPITOLO 4

CODICE SQL

4.1 INTRODUZIONE

Come detto nella descrizione, l'intero progetto è stato sviluppato utilizzando il framework *laravel*. Le tabelle del database sono state create utilizzando le migration. Per ogni tabella del database è stato eseguito il comando `php artisan make:migration create_table_nomeTabella`. Questo comando genera una classe migration all'interno del file `create_table_nomeTabella.php` nella quale sono definiti i metodi `up()` e `down()`. All'interno di `up()` vengono inseriti tutti i comandi per la creazione delle tabelle. Generate le migrations per tutte le tabelle, il comando `php artisan migrate` traduce i comandi specificati nel metodo `up`, in comandi SQL per la creazione delle tabelle. Di seguito viene riportato il codice SQL generato dalle migrations.

4.2 CODICE

UTENTE

```
CREATE TABLE utente (  
    utenteID INT AUTO_INCREMENT PRIMARY_KEY,  
    nome VARCHAR(255) NOT NULL,  
    cognome VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL UNIQUE,  
    password VARCHAR(255) NOT NULL,  
    citta VARCHAR(255) NOT NULL,  
    dataNascita DATE  
);
```

NOTIFICA

```
CREATE TABLE notifica (  
    notificaID INT AUTO_INCREMENT PRIMARY_KEY,  
    utenteID VARCHAR(255) FOREIGN_KEY REFERENCES utente(utenteID),  
    tipo VARCHAR(255),  
    tipoID VARCHAR(255),  
    letta BIT DEFAULT 0  
);
```

AMICIZIA

```
CREATE TABLE amicizia (  
    utenteID1 INT FOREIGN_KEY REFERENCES utente(utenteID),  
    utenteID2 INT FOREIGN_KEY REFERENCES utente(utenteID),  
    timestamp TIMESTAMP,
```

```

        Stato VARCHAR(255) NOT_NULL DEFAULT 'sospesa'
    );

COMMENTO
CREATE TABLE commento (
    commentoid INT AUTO_INCREMENT PRIMARY_KEY,
    utenteID INT FOREIGN_KEY REFERENCES utente(utenteID),
    postID INT FOREIGN_KEY REFERENCES post(postID),
    contenuto TEXT NOT_NULL,
    timestamp TIMESTAMP
);

POST
CREATE TABLE post (
    postID INT AUTO_INCREMENT PRIMARY_KEY,
    contenuto TEXT NOT_NULL,
    timestamp TIMESTAMP,
    utenteID FOREIGN_KEY REFERENCES utente(utenteID)
);

MEDIA
CREATE TABLE media (
    mediaID INT AUTO_INCREMENT PRIMARY_KEY,
    postID INT FOREIGN_KEY REFERENCES post(PostID),
    percorso TEXT NOT_NULL
);

REAZIONE
CREATE TABLE rezione (
    reazioneID INT AUTO_INCREMENT PRIMARY_KEY,
    utenteID INT FOREIGN_KEY REFERENCES utente(utenteID),
    postID INT FOREIGN_KEY REFERENCES post(postID),
    flag BIT
);

```

CAPITOLO 5

QUERY

CAPITOLO 6

INTERFACCIA

BIBLIOGRAFIA

- [1] Marius Leibold, Gilbert J. B. Probst, Michael Gibbert, *Strategic Management in the Knowledge Economy: New Approaches and Business Applications*, John Wiley & Sons, 2007.
- [2] Tetra Pak launches FSC cartons in China, <http://beta.nepcon.org/newsroom/tetra-pak-launches-fsc-cartons-china>, 10 Giugno 2010.
- [3] Circular Economy, Sustainable Materials Management, and the Importance of KPIs: <https://sustainablepackaging.org/circular-economy-sustainable-materials-management-importance-kpis/>, 17 Maggio 2017.