UninaSocialGroup è un social network che offre agli utenti la possibilità di creare partecipare ed interagire a gruppi tematici, a gestirli e amministrarli nel caso si è un creatore e a ricevere notifiche ,like e commenti, dettati dai propri vincoli. La particolarità del progetto ma, in generale del social network è anche la personalizzazione di questo e l’aggiunta di proprie conoscenze del dominio per farlo proprio.

# **IDENTIFICAZIONE E ANALISI DEI REQUISITI**

Spiegando cronologicamente le fasi del progetto, è stato ragionato partendo dal concetto iniziale di GRUPPO:

esso è cardine delle relazioni tra le classi alle quali, si susseguono tutte le principali interazioni con protagonisti i concetti di utente, post e recensione.

**UTENTE**

Partendo in primis dal concetto di **UTENTE** è stato considerato come classe avente un **nickname** e una **password** che lo distingue univocamente almeno in questa prima parte di analisi. Inoltre è specificata una **bio** opzionale da eventualmente popolare e, due counter, riguardanti il **numero di pubblicazioni** e il **numero dei gruppi** a cui un **UTENTE** è iscritto, riferenti rispettivamente alle associazioni con altre due classi: **POST** e **GRUPPO**.

Inoltre la relazione con GRUPPO, si divide ad albero in due ulteriori sotto-relazioni: la possibilità di iscriversi ad un **GRUPPO** e, se un utente è specializzato in

**UTENTE-CREATORE**, la possibilità, appunto, di creare un **GRUPPO**. Nello specifico, il compito di **CREATORE** verte sulla possibilità di accettare o rifiutare **UTENTI** all’interno di quest’ ultimo.

**GRUPPO**

GRUPPO in quanto classe cardine è stato popolato da attributi riguardanti principalmente e inizialmente il nome di un gruppo, i tag (attributo multi-valore che specificano il tema e topic relativi a quest’ultimo) , una descrizione gruppo e informazioni generali riguardanti il numero dei post (derivato dall’associazione con l’entità POST) e numero di utenti iscritti(derivato dall’associazione con l’entità UTENTE). Inoltre è descritto anche la data di creazione del suddetto GRUPPO.

Un’altra associazione fondamentale che coinvolge sia la classe gruppo che la classe utente è la classe POST. La quale coinvolge le due precedenti classi e descrive le interazioni possibili ad un UTENTE nel momento in cui è iscritto ad un GRUPPO.

**POST**

La classe post è concettualmente intesa come qualsiasi tipo di contenuto condiviso dall’utente nei suddetti gruppi che li mettono in relazione.

Nello specifico la classe **POST** è descritta fondamentalmente da due counter riguardanti il numero di like e numero di commenti (derivanti entrambi dall’associazione con l’entità **RECENSIONE.**)

La classe POST in maniera più specifica si specializza in due ulteriori sottoclassi:

**FOTO**

Contenuto multimediale che un utente ha la possibilità di condividere

con gli altri membri di un gruppo, ha come discriminante rispetto alla classe generalizzata una descrizione e una FOTO.

**CONTENUTO TESTUALE**

Semplice messaggio di testo pubblicato da un utente col fine semplice di esprimere un parere sui topic discussi nei gruppi.

Generalmente descrivono i due ruoli principali in diverse istanze che può ricoprire un post.

**RECENSIONE**

Nel dettaglio l’entità recensione specifica la volontà di un utente , attraverso le due sottoclassi nella quale si **specializza**, di lasciare un **like** o un **commento**.

Inoltre è descritta anche la data della condivisione e l’orario relativo.

La generalizzazione in un commento di una recensione ha come attributo discriminante la presenza di un contenuto testuale.

**NOTIFICA**

Inoltre al fine di realizzare un ambiente più realistico possibile al dominio nel quale ci stiamo riferendo, le classi gruppo, recensione e post hanno una relazione importante con un ultima e ulteriore classe: l’entità notifica.

Essa viene gestita in maniera diversa in altrettante diverse istanze alle quali possiamo far fronte.

Una notifica viene inviata a un utente in 3 diversi casi:

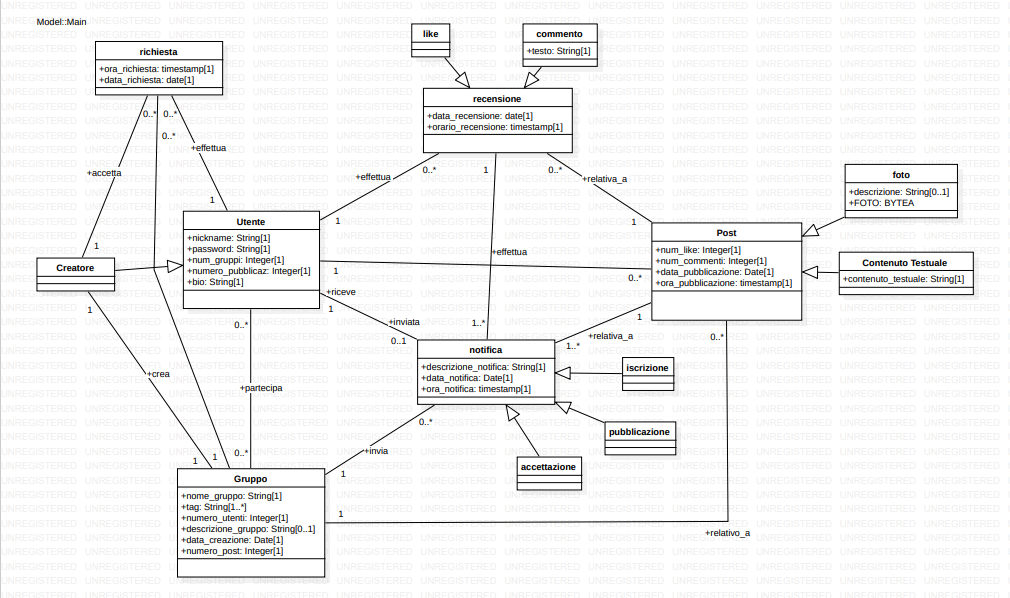
1. Nel caso di una pubblicazione di un post in un gruppo nel quale un utente è amministratore. ASSOCIAZIONE CON POST.
2. Nel caso nel quale un utente accede ad un gruppo di cui si è creatore. ASSOCIAZIONE CON GRUPPO.
3. Nel caso un utente interagisca con un post di cui si è autori. ASSOCIAZIONE CON RECENSIONE.

La gestione della discriminazione delle notifiche è prevalentemente gestita con scelte progettuali prese nella ristrutturazione del CLASS DIAGRAM.

# **PROGETTAZIONE CONCETTUALE**

Per la progettazione concettuale si parte da un CLASS DIAGRAM non ristrutturato, un diagramma EER e per poi passare alla fase di ristrutturazione con relativi successivi associazioni e vincolo.

Ci siamo serviti prevalentemente dell’applicativo STAR UML per la realizzazione dei seguenti



# **DIAGRAMMA EER**

# **RISTRUTTURAZIONE DEL CLASS DIAGRAM**

Al fine di rendere il class diagram idoneo alla creazione di uno schema relazionale saldo e coerente verranno rimosse durante questa procedura di ristrutturazione qualsiasi tipo di attributo multiplo, composto ed eventuali specializzazioni e generalizzazioni provvedendo a scelte progettuali concerne e diverse in base alla situazione che si sta andando a gestire.

Individuazione degli identificativi

La ricerca di identificativi garantisce il mantenimento dell’integrità referenziale nelle istanze diverse del nostro applicativo.

**Utente**

Per la classe utente si sceglie di applicare un idUtente univoco, successivamente gestito in maniera seriale, che distinguerà quest’ultimo, dando a questo attributo il ruolo di chiave primaria in quanto la combinazione di username e password poteva risultare scomoda nella gestione di successive scelte progettuali.

**Gruppo**

Per il gruppo si sceglie di applicare un idGruppo che come per utente è seriale e va a distinguere univocamente ogni gruppo e questa sarà la sua chiave primaria.

**Post**

Discorso analogo per la classe post al quale viene affibiato un idPost, anch’esso seriale che va a distinguerlo dagli altri post presenti nelle diverse istanze.

# **DIZIONARIO DELLE CLASSI**

|  |  |  |
| --- | --- | --- |
| **NOME** | **DESCRIZIONE** | **ATTRIBUTI** |
| **UTENTE** | Contiene tutti gli utenti standard che entrano ed interagiscono con gruppi ed altri utenti. | **Nickname**(Varchar(30)):  L’identificativo con il quale un utente si registra.  **Password** (Varchar(20))  parola chiave con la quale un utente entra nel suo profilo.  **idUtente(Int SERIAL)**  identificativo univoco associato casualmente ad ogni utente(vincolo reminder: ogni id dovrà essere diverso dall’altro)  **bio(varchar(200))**  descrizione del profilo. |
| **GRUPPO** | Contiene tutti i gruppi che sono stati creati dagli utenti e sui quali interagiscono postando contenuti. | **idGruppo(varchar(10))**  Identificativo univoco di ogni gruppo.  **Nome\_gruppo(varchar(30))**  Nome scelto dal creatore.  **Descrizione gruppo (varchar(200))**  Testo di informazioni generali sul gruppo.  **Data\_creazione(date)**  Data nella quale è stato creato il gruppo |
| **POST** | Contiene tutti i post caricati dagli utenti sul social | **idPost(varchar(10))**  identificativo univoco di ogni post  **idUtente(varchar(10))**  riferimento esterno all’utente che ha effettuato il caricamento.  **idGruppo(varchar(10))**  riferimento esterno al gruppo sul quale il post verrà caricato.  **Descrizione(varchar(30))**  Testo semplice scritto dall’utente al caricamento del post.  **Numero\_like(integer)**  Numero di like ricevuti dal post, derivato dall’associazione con recensione.  **Numero\_commenti(integer)**  Numero di commenti ricevuti dal post, derivato dall’associazione con recensione.  **Data\_pubblicazione(date)**  Data nella quale è stato caricato il post sul gruppo.  **Orario\_pubblicazione(time)**  Orario di pubblicazione del post |
| **RECENSIONE** | Contiene tutte le interazioni con un post che siano like o commenti | **idUtente(varchar(10))**  riferimento all’utente che ha lasciato l’interazione con il post.  **idPost(varchar(10))**  riferimento al post sul quale viee lasciata la recensione.  **Data\_recensione(date)**  Data nella quale è stata lasciata la recensione  **Orario\_recensione(time)**  Orario nel quale è stata lasciata la recensione  **Tipo(boolean)**  Assume 2 valori che determinano se la recensione sia un like, ciò porterà ad un incremento del numero di like del post riferito e renderà NULL l’attributo descrizione o un commento, che farà la stessa cosa con i commenti e aggiungerà un testo all’attributo descrizione |
| **Notifica** | Contiene tutte le informazioni inerenti alle notifiche | **IdUtente(varchar(10))**  Riferimento All’Utente che riceve la notifica  **IdPost(varchar(10))**  Riferimento al contenuto creato il quale di fatto scaturisce la notifica  **IdGruppo(varchar(10))**  Riferimento al Gruppo da cui proviene la notifica  **Data\_Notifica(date)**  Data Di Ricevimento Della Notifica  (ovvero la data di creazione del contenuto)  **Orario\_Notifica(time)**  Orario Di Ricevimento Notifica (ovvero l’orario di creazione del contenuto)  **Descrizione\_Notifica(varchar(200))**  Contenuto Testuale Della Notifica |
| **Tag** | Contiene tutti i Tag | **IdTag(varchar(10))**  **Testo(varchar(20))**  **IdGruppo(varchar(10))** |
| **CREA** | Classe di associazione tra l’utente che crea un gruppo e il gruppo stesso | **idUtente(varchar(10))**  id dell’utente esterno che compie l’atto di creare il gruppo  **idGruppo(varchar(10))**  id del gruppo che viene creato |
|  |  |  |

# **DIZIONARIO DELLE ASSOCIAZIONI**

|  |  |  |
| --- | --- | --- |
| **NOME** | **DESCRIZIONE** | **CLASSI COINVOLTE** |
| **PARTECIPA** | Descrive l'associazione di partecipazione con una classe Gruppo. | **Utente[0...\*]:** indica gli utenti che si registrano all'interno di un gruppo.  **Gruppo[0...\*]:** indica i gruppi del social ai quali gli utenti si iscrivono per interagire con altri utenti. |
| **INVIA** | Descrive il processo per il quale un gruppo invia una notifica relativa a un post. | **Gruppo[1]:** Indica la classe gruppo che si occuperà di inviare la notifica.  **Notifica[0...\*]:** indica la classe notifica che verrà inviata a un utente una volta che qualcuno pubblica un post in un gruppo a cui è iscritto. |
| **INVIATA\_A** | Associazione che descrive il legame tra la notifica che viene inviata all'utente iscritto ad un determinato gruppo e l'utente che riveve suddetta notifica. | **Utente[1...\*]:** partecipazione che descrive il concetto che una notifica può arrivare a nessun utente [0] nel caso non ci siano altri partecipanti in un gruppo o nel caso ce n'è siano di più viene inviata a più utenti[\*].  **Notifica[0...\*]:** la classe notifica nell'associazione ha la parte di rappresentare le notifiche inviate ad uno o più utenti che ricevono 0 o più notifiche. |
| **ACCETTA** | Associazione ricorsiva che lega due utenti e specifica il gesto di accettare o meno all'interno di un gruppo un utente che fa richiesta, associazione possibile solo nel caso che da una delle due parti vi sia un utente creatore. | **Utente[1]:** utente obbligatoriamente creatore di u gruppo che compie il gesto di accettare una richiesta di entrare in un gruppo.  **Utente[0...\*]:** nessuno o più utenti che richiedono allo stesso creatore di entrare in un gruppo. |
| **EFFETTUA** | Associazione che lega una recensione all'utente che la effettua su un post. | **Utente[1]:** L'utente univoco che scrive un commento o lascia un like ad una pubblicazione.  **Recensione[0...\*]:** zero o più recensioni pubblicate da uno stesso utente per uno o più post diversi. |
| **RELATIVA\_A** | Associazione che descrive il legame tra una o più recensioni lasciate dagli utenti su un post. | **Recensione[0...\*]:** classe recensione che indica il concetto che 0 o molte recensioni possono appartenere ad un unico post.  **Post[1]:** classe post unico al quale vengono associate più recensioni da diversi utenti. |
| **PUBBLICA** | Associazione che descrive l'azione di un utente di pubblicazione di un post in un gruppo al quale iscritto. | **Utente[1]:**classe che indica che un post è pubblicato da uno e un solo utente.  **Post[0...\*]:**classe che indica che un utente può pubblicare nessuno[0] o più post[\*] |
| **RELATIVO\_A** | Associazione che descrive il legame tra una o più pubblicazioni (post) relative ad un gruppo. | **Post[0...\*]:**classe che specifica il tipo di contenuto pubblicabile e le interazioni ottenibili su di esso.La molteplicità relativa alla classe,vuole indicare la possibilità di esserci **[0]** post in un gruppo o, molti **[\*]**.  **Gruppo[1]:** La classe gruppo specifica l'univoco**[1]** gruppo a cui puo appartenere un post. |
|  |  |  |
|  |  |  |

**DIZIONARIO DEI VINCOLI**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome vincolo | TIPO | Descrizione | Identificativo |
| Id\_utente univoco | Vincolo di univocità | L’id utente dovrà essere univoco per ogni utente e non potranno mai esserci duplicati. | UTENTE\_UK1 |
| UTENTE accetta solo se creatore | Vincolo interrelazionale | Un’entità utente può accettare un altro utente se e solo se è creatore nel suddetto gruppo della richiesta. Ergo se il suo idUtente e il gruppo in questione si trovano presenti nella tabella “creatore” | CREATORE\_CHECK |
| Data recensione maggiore data Post | Vincolo interrelazionale | La data di pubblicazione di una recensione deve essere più in avanti col tempo di quella del post a cui fa riferimento. | RECENSIONE\_DATA\_CHECK |
| Orario Recensione maggiore orario post | Vincolo interrelazionale | L’orario di una recensione effettuata nello stesso giorno di un post deve essere sempre maggiore dell’orario di pubblicazione del post. | RECENSIONE\_ORARIO\_CHECK |
|  |  |  |  |
| Id\_gruppo univoco | Vincolo di univocità | L’id di un gruppo dovrà essere univoco per ogni utente e non potranno mai esserci duplicati. | UTENTE\_UK1 |
| Tipo recensione può assumere 2 valori | Vincolo intrarelazionale | L’attributo tipo di recensione può assumere o valore ‘like’ o valore ‘commento’ | RECENSIONE\_TIPO\_CHECK1 |
| Descrizione NULL se recensione tipo ‘like’ | Vincolo intrarelazionale | Se il tipo di una recensione corrisponde a ‘like’ la sua descrizione dovra essere NULL | RECENSIONE\_TIPO\_CHECK2 |
| Numero di POST in un gruppo non negativo | Vincolo di controllo | Il numero di post pubblicati in un gruppo non può essere negativo. | GROUP\_CHECK2 |
|  |  |  |  |
| Data notifica maggiore data Post | Vincolo interrelazionale | La data di pubblicazione di una notifica deve essere più in avanti col tempo di quella del post a cui fa riferimento. | NOTIFICA\_DATA\_CHECK |
| Orario notifica maggiore orario post | Vincolo intrarelazionale | L’orario di una notifica effettuata nello stesso giorno di un post deve essere sempre maggiore dell’orario di pubblicazione del post. | NOTIFICA\_ORARIO\_CHECK |
| L’id\_utente della notifica stesso dell’utente | Vincolo di integrità referenziale | L’id utente della notifica deve referenziato a quello dell’utente al quale sarà inviato. | NOTIFICA\_CHECK\_UTENTE |
| Numero\_like non negativo | Vincolo di controllo | Il numero dei like di un post non deve essere negativo. | POST\_CHECK1 |
| Numero\_commenti non negativo | Vincolo di controllo | Il numero dei commenti di un post non deve essere negativo. | POST\_CHECK2 |
| Aumento like solo se recensione tipo like | Vincolo Interrelazionale | Il numero dei like in un post aumenta solo se la recensione è di tipo like. | RECENSIONE\_CHECK1 |
| Aumento commenti solo se recensione tipo commenti | Vincolo Interrelazionale | Il numero dei commenti in un post aumenta solo se la recensione è di tipo commento. | RECENSIONE\_CHECK2 |
| idRecensione e idUtente NULL se notifica è relativa a post | Vincolo intrarelazionale | I campi idRecensione e idUtente di una riga nella tabella recensione saranno nulli se la notifica è relativa a una pubblicazione quindi si riferiscd a un post. | **NOTIFICA\_TIPO\_CHECK1** |
| idRecensione e idPost NULL se notifica è relativa a richiesta | Vincolo intrarelazionale | I campi idRecensione e idPost saranno nulli se la notifica è di tipo relativo a una richiesta | **NOTIFICA\_TIPO\_CHECK2** |
| idRecensione e idPost NULL se notifica è relativa a richiesta | Vincolo intrarelazionale | campi idRecensione e idPost saranno nulli se la notifica è di tipo relativo a un’accettazione | **NOTIFICA\_TIPO\_CHECK3** |

# **SCHEMA LOGICO**

**SCHEMA FISICO**

CREAZIONE DELLE TABELLE

CREATE TABLE notifica (

idUtente varchar(10),

idPost varchar(10),

idGruppo varchar(10),

data\_notifica date,

orario\_notifica timestamp,

CONSTRAINT pk\_notifica PRIMARY KEY (idPost, idGruppo)

);

CREATE TABLE Utente (

nickname varchar(20) NOT NULL,

password varchar(8)NOT NULL ,

idUtente varchar(10) NOT NULL PRIMARY KEY,

numero\_gruppi integer,

bio varchar(100),

creatore BOOLEAN

);

CREATE TABLE recensione (

idRecensione SERIAL NOT NULL PRIMARY KEY,

IdUtente INTEGER,

orario TIMESTAMP,

data\_recensione DATE,

idPost INTEGER,

descrizione VARCHAR(300),

tipo VARCHAR(10),

CONSTRAINT fk\_recensione\_utente FOREIGN KEY (idUtente) REFERENCES utente(idUtente),

CONSTRAINT fk\_recensione\_post FOREIGN KEY (idPost) REFERENCES post(idPost)

);

CREATE TABLE post (

idPost SERIAL PRIMARY KEY,

idUtente VARCHAR(10),

idGruppo VARCHAR(10),

descrizione VARCHAR(300),

data\_pubblicazione DATE,

orario\_pubblicazione TIMESTAMP,

CONSTRAINT fk\_utente FOREIGN KEY (idUtente) REFERENCES Utente(idUtente),

CONSTRAINT fk\_gruppo FOREIGN KEY (idGruppo) REFERENCES Gruppo(idGruppo)

);

CREATE TABLE notifica(

idPost VARCHAR(10),

idUtente VARCHAR(10),

idGruppo VARCHAR(10),

data\_notifica DATE,

orario\_notifica TIMESTAMP,

descrizione\_notifica VARCHAR(20)

CONSTRAINT

foreign\_key\_utente FOREIGN KEY(idutente) references utente(idutente)

CONSTRAINT

foreign\_key\_post FOREIGN KEY(idpost) references post(idpost);

CONSTRAINT

foreign\_key\_gruppo FOREIGN KEY(idgruppo) references gruppo(idgruppo);

);

CREATE TABLE richiesta (

idRichiedente INTEGER,

idGruppo INTEGER,

orario\_richiesta TIMESTAMP,

data\_richiesta DATE,

CONSTRAINT fk\_richiesta\_utente FOREIGN KEY (idRichiedente) REFERENCES utente(idUtente),

CONSTRAINT fk\_richiesta\_gruppo FOREIGN KEY (idGruppo) REFERENCES gruppo(idGruppo)

);

CREATE TABLE accetta (

idutente VARCHAR(10),

idGruppo VARCHAR(10),

idCreatore VARCHAR(10),

CONSTRAINT fk\_richiesta\_utente FOREIGN KEY (idutente) REFERENCES richiesta(idRichiedente),

CONSTRAINT fk\_richiesta\_gruppo FOREIGN KEY (idGruppo) REFERENCES gruppo(idGruppo)

CONSTRAINT fk\_richiesta\_creatore FOREIGN KEY (idcreatore) REFERENCES gruppo(idutente)

);

CREATE TABLE iscrizione (

idGruppo INTEGER,

idUtente INTEGER,

CONSTRAINT fk\_gruppo FOREIGN KEY (idGruppo) REFERENCES gruppo(idGruppo),

CONSTRAINT fk\_utente FOREIGN KEY (idUtente) REFERENCES utente(idUtente)

);

## PROCEDURE

**Check\_iscritto**

create or replace function check\_iscritto() returns trigger as $$

declare

nome\_gruppo varchar(255);

begin

select g.nome\_gruppo into nome\_gruppo from gruppo as g where g.idgruppo = new.idgruppo;

if exists(select 1 from iscrizione where idutente = new.idutente

and idgruppo = new.idgruppo)

then raise exception 'Utente già iscritto al gruppo % ',nome\_gruppo;

end if;

return new;

end;

$$ language plpgsql;

create trigger before\_insert\_iscritto

before insert on iscrizione

for each row

execute function check\_iscritto();

**Check\_richiesta**

create or replace function check\_richiesta() returns trigger as $$

declare

nome\_gruppo varchar(255);

begin

select g.nome\_gruppo into nome\_gruppo from gruppo as g where g.idgruppo = new.idgruppo;

if exists(select 1 from richiesta where idutente = new.idutente

and idgruppo = new.idgruppo)

then raise exception 'hai gia richiesto di entrare al gruppo % ',nome\_gruppo;

end if;

return new;

end;

$$ language plpgsql;

create trigger before\_insert\_richiesta

before insert on richiesta

for each row

execute function check\_richiesta()

**PROCEDURA POST PER UTENTE**

CREATE OR REPLACE PROCEDURE visualizza\_post\_utente(idUtente INTEGER)

AS $$

DECLARE

username VARCHAR(10);

post\_cursor CURSOR FOR

SELECT FOTO, descrizione, data\_pubblicazione, orario\_pubblicazione

FROM post WHERE idutente = idUtente;

post\_record RECORD;

BEGIN

SELECT nickname INTO username FROM utente WHERE idutente = idUtente;

RAISE NOTICE 'Visualizzazione dei post per utente %', username;

OPEN post\_cursor;

LOOP

FETCH post\_cursor INTO post\_record;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'DESCRIZIONE: % DATA: % ORA: %', post\_record.descrizione,

post\_record.data\_pubblicazione,

post\_record.orario\_pubblicazione;

END LOOP;

CLOSE post\_cursor;

END;

$$ LANGUAGE plpgsql;

**PROCEDURA UTENTI PER GRUPPO**

CREATE OR REPLACE PROCEDURE visualizza\_utenti\_gruppo(idGruppo INTEGER)

AS $$

DECLARE

nome\_gruppo VARCHAR(20);

utenti\_cursor CURSOR FOR

SELECT u.nickname

FROM utente as u join iscrizione as isc on u.idutente = isc.idutente;

utenti\_record RECORD;

BEGIN

SELECT nome\_gruppo INTO nome\_gruppo FROM gruppo WHERE idgruppo = idGruppo;

RAISE NOTICE 'Visualizzazione utenti per gruppo %', nome\_gruppo;

OPEN utenti\_cursor;

LOOP

FETCH utenti\_cursor INTO utenti\_record;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'nome utente: %', utenti\_record.nickname;

END LOOP;

CLOSE utenti\_cursor;

END;

$$ LANGUAGE plpgsql;

**Procedura recensioni per post**

CREATE OR REPLACE PROCEDURE visualizza\_recensioni\_post(idPost INTEGER)

AS $$

DECLARE

recensione\_cursor CURSOR FOR

SELECT u.nickname as nickname, r.descrizione as descrizione, r.data\_recensione as data, r.orario as ora

FROM utente as u join recensione as r on u.idutente = r.idutente where r.tipo = 'commento';

recensione\_record RECORD;

BEGIN

OPEN recensione\_cursor;

LOOP

FETCH recensione\_cursor INTO recensione\_record;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'nome utente: % descrizione : % data: % ora: %',recensione\_record.nickname,recensione\_record.descrizione,

recensione\_record.data, recensione\_record.ora;

END LOOP;

CLOSE recensione\_cursor;

END;

$$ LANGUAGE plpgsql;

PROCEDURA GRUPPI PER UTENTE

CREATE OR REPLACE PROCEDURE public.visualizza\_gruppi\_utente(

IN idutente integer)

LANGUAGE 'plpgsql'

AS $BODY$

DECLARE

username VARCHAR(10);

group\_cursor CURSOR FOR

SELECT grup.nome\_gruppo as nome\_gruppo

FROM gruppo as grup JOIN iscritto as isc ON grup.idgruppo = isc.idgruppo

WHERE isc.idutente = idutente;

group\_record RECORD;

BEGIN

SELECT nickname INTO username FROM utente WHERE idutente = idUtente;

RAISE NOTICE 'Visualizzazione gruppi per utente %', username;

OPEN group\_cursor;

LOOP

FETCH group\_cursor INTO group\_record;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'Nome gruppo: %', group\_record.nome\_gruppo;

END LOOP;

CLOSE group\_cursor; -- Chiusura del cursore corretta

END;

$BODY$;

## VISTE

**NumeroPostPerUtente**

CREATE VIEW NumeroPostPerUtente AS

SELECT U.nickname AS Utente,

COUNT(P.idPost) AS NumeroPost

FROM UTENTE U

Left JOIN POST P ON U.idUtente = P.idUtente

GROUP BY U.idUtente, U.nickname;

**NumeroPostPerGruppo**

CREATE VIEW NumeroPostPerGruppo AS

SELECT G.nome\_gruppo AS Gruppo,

COUNT(P.idPost) AS NumeroPost

FROM GRUPPO G

JOIN POST P ON G.idGruppo = P.idGruppo

GROUP BY G.nome\_gruppo;

**NumeroUtentiPerGruppo**

CREATE VIEW NumeroUtentiPerGruppo AS

SELECT G.nome\_gruppo AS Gruppo,

COUNT(I.idUtente) AS NumeroUtenti

FROM GRUPPO G

left JOIN ISCRIZIONE I ON G.idGruppo = I.idGruppo

**NumeroCommentiPerPost**

CREATE VIEW NumeroCommentiPerPost AS

SELECT P.idPost, COUNT(R.\*) AS NumeroCommenti

FROM post P

Left JOIN recensione R ON P.idPost = R.idPost

where R.tipo = 'commento'

GROUP BY P.idPost;

GROUP BY G.idGruppo, G.nome\_gruppo;

**NumeroLikePerPost**

CREATE VIEW NumeroLikePerPost AS

SELECT P.idPost, COUNT(R.\*) AS NumeroLike

FROM post P

Left JOIN recensione R ON P.idPost = R.idPost

where R.tipo = 'like'

GROUP BY P.idPost;

## VINCOLI

**RECENSIONE\_TIPO\_CHECK1**

ALTER TABLE recensione

ADD CONSTRAINT check\_tipo

CHECK (tipo IN ('like', 'commento'));

**NOTIFICA\_TIPO\_CHECK1**

ALTER TABLE notifica

ADD CONSTRAINT check\_tipo\_pubbl\_id\_null

CHECK (

(tipo != 'pubblicazione') OR

(tipo = 'pubblicazione' AND idRecensione IS NULL AND idUtente IS NULL)

);

**NOTIFICA\_TIPO\_CHECK2**

ALTER TABLE notifica

ADD CONSTRAINT check\_tipo\_rich\_null

CHECK (

(tipo != 'richiesta') OR

(tipo = 'richiesta' AND idRecensione IS NULL and idPost is null)

);

**NOTIFICA\_TIPO\_CHECK3**

ALTER TABLE notifica

ADD CONSTRAINT check\_tipo\_pubbl\_id\_null

CHECK (

(tipo != 'accettazione') OR

(tipo = 'accettazione' AND idRecensione IS NULL and idPost)

);