UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
MATEMATICA E
INFORMATICA
DEMACS

Corso di Laurea in Informatica

A.A. 2019/2020

PROGETTO PER IL CORSO DI

BASI DI DATI RELAZIONALI

DOCENTE: PROF. P. RULLO

LABORATORIO: ING. G. LABOCCETTA,
DOTT.SSA D. ANGILICA

SISTEMA INFORMATIVO

PER LA GESTIONE DI

UN SOCIAL NETWORK.

GRUPPO 18
<200632 Matteo Perfidio>
<200705 Antonino Natale>

Sommario

1 Tematica Progettuale

La progettazione del sistema informatico in esame riguarda la gestione di un social network.

2 Raccolta e analisi dei requisiti

2.1 Raccolta dei requisiti

Il social network Quarantine vuole rivedere il proprio database. In occasione dell'introduzione di nuove funzionalità, vuole approfittarne per rendere più efficiente il sistema. Ogni utente che si registra sul sito fornisce la propria anagrafica (nome, cognome, data di nascita), una username che lo identifica, un indirizzo e-mail che può essere usato per una sola registrazione e una password lunga almeno 8 e massimo 16 caratteri. Al momento della registrazione, l'utente decide se tenere il proprio profilo pubblico o privato (scelta che può essere modificata in seguito). In quest'ultimo caso solo utenti presenti nella sua lista di amici potranno visualizzare il suo profilo e ciò che viene pubblicato. Sul profilo si possono trovare solo post testuali, ognuno con una data, un orario, eventuali tag ed eventuali commenti (di chi lo ha postato o di altri utenti) ciascuno dei quali con data e orario. Se il profilo è privato, solo gli amici possono commentare. Il sistema, ogni giorno propone all'utente i post da egli pubblicati in quel giorno negli anni precedenti. Gli utenti possono scambiare tra loro anche dei messaggi privati. Ogni messaggio ha un testo, una data e un orario e si vuole sapere se è stato letto oppure no.

2.2 Analisi dei requisti

2.2.1 Eliminazione delle ambiguità

LINEA	TERMINE	SINONIMI	MOTIVAZIONE CORREZIONE
1	Database	Sistema	Termine analogo, concettualmente, a database
1	Social Network	Sito	Termine troppo generico, usato per indicare il social network
9	Amici		Termine ambiguo che fa riferimento ad una data cerchia ristretta di utenti

2.2.2 Ristrutturazione delle specifiche e individuazione dei concetti di base

Il social network Quarantine vuole rivedere il proprio database. In occasione dell'introduzione di nuove funzionalità, vuole approfittarne per rendere più efficiente il *database*. Ogni utente che si registra sul *social network* fornisce la propria anagrafica (nome, cognome, data di nascita), una username che lo identifica, un indirizzo e-mail che può essere usato per una sola registrazione e una password lunga almeno 8 e massimo 16 caratteri. Al momento della registrazione, l'utente decide se tenere il proprio profilo pubblico o privato (scelta che può essere modificata in seguito). In quest'ultimo caso solo utenti presenti nella sua *cerchia ristretta di utenti* potranno visualizzare il suo profilo e ciò che viene pubblicato. Sul profilo si possono trovare solo post testuali, ognuno con una data, un orario, eventuali tag ed eventuali commenti (di chi lo ha postato o di altri utenti) ciascuno dei quali con data e orario. Se il profilo è privato, solo gli utenti *appartenenti ad una cerchia ristretta* possono commentare. Il *database*, ogni giorno propone all'utente i post da egli pubblicati in quel giorno negli anni precedenti. Gli utenti possono scambiare tra loro anche dei messaggi privati. Ogni messaggio ha un testo, una data e un orario e si vuole sapere se è stato letto oppure no.

	FRASI DI CARATTERE GENERALE
1-2	Il social network Quarantine vuole rivedere il proprio database. In occasione dell'introduzione di nuove funzionalità, vuole approfittarne per rendere più efficiente il *database*.

	FRASI RELATIVE AD UTENTE
2-5	Ogni utente che si registra sul *social network* fornisce la propria anagrafica (nome, cognome, data di nascita), una username che lo identifica, un indirizzo e-mail che può essere usato per una sola registrazione e una password lunga almeno 8 e massimo 16 caratteri.

	FRASI RELATIVE A PROFILO
7-9	Sul profilo si possono trovare solo post testuali, ognuno con una data, un orario, eventuali tag ed eventuali commenti (di chi lo ha postato o di altri utenti) ciascuno dei quali con data e orario.

	FRASI RELATIVE A TIPI SPECIFICI DI PROFILO
5-7	Al momento della registrazione, l'utente decide se tenere il proprio profilo pubblico o privato (scelta che può essere modificata in seguito). In quest'ultimo caso solo utenti presenti nella sua *cerchia ristretta di utenti* potranno visualizzare il suo profilo e ciò che viene pubblicato.
9-10	Se il profilo è privato, solo gli utenti *appartenenti ad una cerchia ristretta* possono commentare.

	FRASI RELATIVE A PM
11-12	Gli utenti possono scambiare tra loro anche dei messaggi privati. Ogni messaggio ha un testo, una data e un orario e si vuole sapere se `e stato letto oppure no.

2.3 Specifica delle operazioni richieste

1. Aggiornare il profilo dell'utente 'Tizio' e renderlo pubblico.
2. Trovare gli amici dell'utente 'Caio'.
3. Trovare i profili pubblici con meno di 50 amici.
4. Trovare i post dei profili pubblici con il minor numero di commenti.
5. Trovare per ogni profilo i post con il piu` alto numero di commenti.
6. Controllare che se un profilo è privato, allora solo un amico può commentare.
7. Alle 0:00 di ogni giorno si rendano disponibili, per ciascun utente, i post pubblicati in quel giorno negli anni precedenti.

3 Progettazione Concettuale

3.1 Schemi E-R

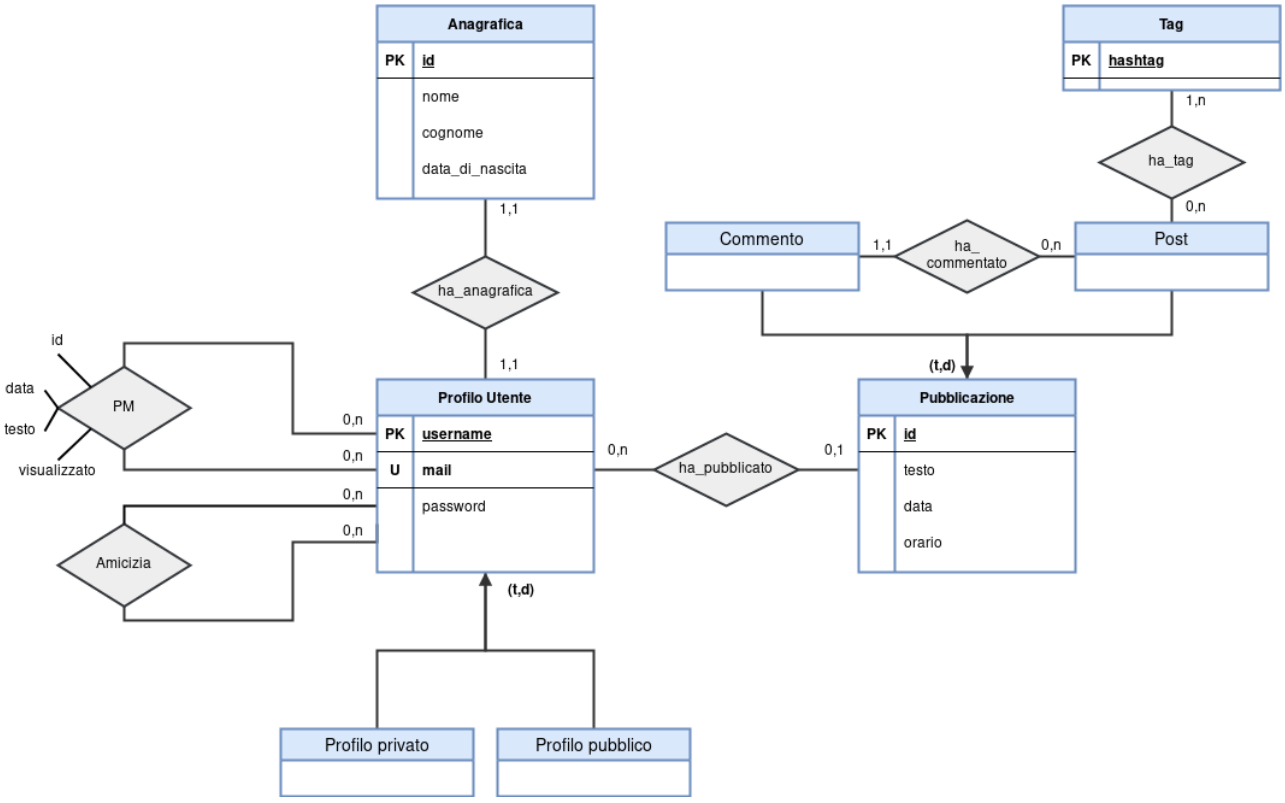

Figura 1 - Schema E/R

3.2 Documentazione dello schema E-R

3.2.1 Dizionario dei dati

ENTITÀ	IDENTIFICATORE	ATTRIBUTI	DESCRIZIONE
Profilo Utente	username	mail, password	Rappresenta l'entità dell'account utente, che descrive le informazioni univoche di ogni singolo utente registrato.
Anagrafica	Id	nome, cognome, data_di_nascita	Rappresenta l'entità che descrive le informazioni anagrafiche di un utente
Pubblicazione	id	data, orario, testo	Contenuto testuale, con funzione di commento o post.
Post			Contenuto testuale, con funzione di opinione o intervento.
Commento			Contenuto testuale, con funzione di commento.
Tag	hashtag		Un hashtag è un tipo di etichetta (tag) utilizzato come aggregatore tematico, la sua funzione è di rendere più facile per gli utenti trovare messaggi su un tema o contenuto specifico.

ENTITÀ	IDENTIFICATORE	ATTRIBUTI	DESCRIZIONE
Profilo privato			Criterio di privacy profilo utente ristretta.
Profilo pubblico			Criterio di privacy profilo utente pubblica.

3.2.2 Descrizione entità

Profilo Utente			
Descrizione	Rappresenta l'entità dell'account utente, che descrive le informazioni univoche di ogni singolo utente registrato.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Username	Alfanumerico	Obbligatorio	Nome utente che identifica univocamente un profilo.
Mail	Alfanumerico	Unico	E-Mail che identifica univocamente un profilo.
Password	Alfanumerico	Obbligatorio	Codice di accesso al profilo utente.

Anagrafica			
Descrizione	Rappresenta l'entità che descrive le informazioni anagrafiche di un utente.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo anagrafica.
Nome	Alfanumerico	Obbligatorio	Nome anagrafico dell'utente.
Cognome	Alfanumerico	Obbligatorio	Cognome anagrafico dell'utente.
Data_di_nascita	Data	Obbligatorio	Data di nascita anagrafica dell'utente.

Pubblicazione			
Descrizione	Contenuto testuale, con funzione di commento o post.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo post.
Testo	Alfanumerico	Obbligatorio	Contenuto testuale del post.
Data	Data	Obbligatorio	Data di pubblicazione post.
Orario	Temporale	Obbligatorio	Orario di pubblicazione post.

Tag			
Descrizione	Un hashtag è un tipo di etichetta (tag) utilizzato come aggregatore tematico, la sua funzione è di rendere più facile per gli utenti trovare messaggi su un tema o contenuto specifico.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Hashtag	Alfanumerico	Obbligatorio	Identificativo tag.

3.2.3 Descrizione Relazioni

Relazione “Amicizia”	
Descrizione	Informazioni che individuano un legame di amicizia tra due utenti.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Attributi	NULL

Relazione “PM”	
Descrizione	Informazioni che individuano un messaggio privato tra due utenti.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Attributi	
Id	Numerico
Data	Data
Testo	Alfanumerico
Visualizzato	Numerico

Relazione “ha_anagrafica”	
Descrizione	Informazioni che individuano per ogni utente la sua rispettiva anagrafica.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(1, 1)
Anagrafica	(1, 1)
Attributi	NULL

Relazione “ha_pubblicato”	
Descrizione	Informazioni che individuano per ogni utente i suoi rispettivi post o commenti pubblicati.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Pubblicazione	(0, 1)
Attributi	NULL

Relazione “ha_commentato”	
Descrizione	Informazioni che individuano per ogni post i rispettivi commenti.
Entità coinvolte	
Entità	**Cardinalità**
Commento	(1, 1)
Post	(0, N)
Attributi	NULL

Relazione “ha_tag”	
Descrizione	Informazioni che individuano per ogni post i rispettivi hashtag.
Entità coinvolte	
Entità	**Cardinalità**
Post	(0, N)
Tag	(1, N)
Attributi	NULL

3.2.4 Vincoli non espressi dallo schema E/R

REGOLE DI VINCOLO
L’ *e-mail* deve essere valida ed univoca per ogni utente registrato.
L’ *username* deve contenere solo caratteri alfanumerici.
La *password* deve essere lunga almeno 8, massimo 16 caratteri e, deve contenere almeno una lettera maiuscola, una minuscola ed un numero.
La *password* viene memorizzata all’interno della base di dati, dopo essere stata verificata e criptata secondo l’algoritmo di hashing SHA-256.
I *post/commenti* non devono avere contenuti offensivi.
Se il *profilo* è privato, solo gli amici possono commentare o visualizzare il profilo.
Se il *profilo* viene eliminato, i suoi post/commenti/messaggi persistono nella base di dati per un massimo di sei mesi.

4 Progettazione Logica

4.1 Ristrutturazione dello schema E/R

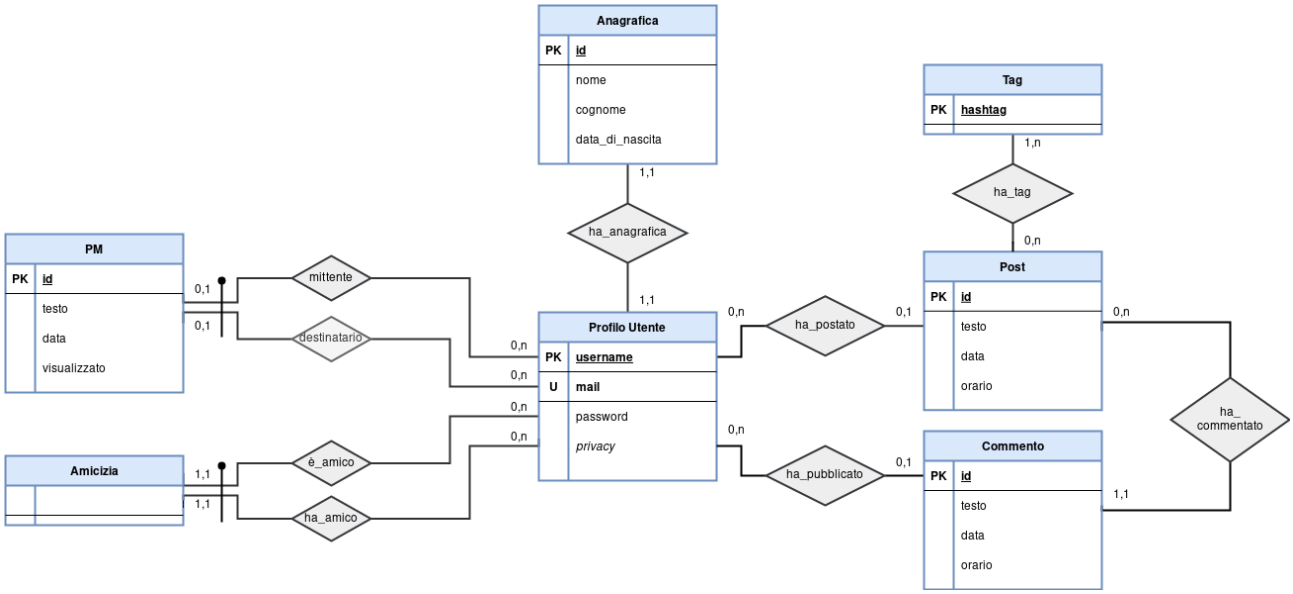

Figura 2 - Schema E/R Ristrutturato

4.1.1 Descrizione Entità

Profilo Utente			
Descrizione	Rappresenta l'entità dell'account utente, che descrive le informazioni univoche di ogni singolo utente registrato.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Username	Alfanumerico	Obbligatorio	Nome utente che identifica univocamente un profilo.
Mail	Alfanumerico	Unico	E-Mail che identifica univocamente un profilo.
Password	Alfanumerico	Obbligatorio	Codice di accesso al profilo utente.
Privacy	Booleano	Obbligatorio	Tipologia di privacy utente: privato, pubblico.

Anagrafica			
Descrizione	Rappresenta l'entità che descrive le informazioni anagrafiche di un utente.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo anagrafica.
Nome	Alfanumerico	Obbligatorio	Nome anagrafico dell'utente.
Cognome	Alfanumerico	Obbligatorio	Cognome anagrafico dell'utente.
Data_di_nascita	Data	Obbligatorio	Data di nascita anagrafica dell'utente.

Post			
Descrizione	Contenuto testuale, con funzione di commento o post.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo post.
Testo	Alfanumerico	Obbligatorio	Contenuto testuale del post.
Data	Data	Obbligatorio	Data di pubblicazione post.
Orario	Temporale	Obbligatorio	Orario di pubblicazione post.

Commento			
Descrizione	Contenuto testuale, con funzione di commento o post.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo commento.
Testo	Alfanumerico	Obbligatorio	Contenuto testuale del commento.
Data	Data	Obbligatorio	Data di pubblicazione commento.
Orario	Temporale	Obbligatorio	Orario di pubblicazione commento.

Amicizia			
Descrizione	Legame di amicizia tra due utenti.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**

PM			
Descrizione	Contenuto testuale, con funzione di messaggio privato tra due utenti.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Id	Numerico	Obbligatorio	Identificativo del messaggio.
Testo	Alfanumerico	Obbligatorio	Contenuto testuale del messaggio.
Data	Data	Obbligatorio	Data di invio messaggio.
Orario	Temporale	Obbligatorio	Orario di invio messaggio.
Visualizzato	Booleano	Obbligatorio	Stato di visualizzazione del messaggio.

Tag			
Descrizione	Un hashtag è un tipo di etichetta (tag) utilizzato come aggregatore tematico, la sua funzione è di rendere più facile per gli utenti trovare messaggi su un tema o contenuto specifico.		
Nome Attributo	**Tipo di dato**	**Vincolo**	**Descrizione**
Hashtag	Alfanumerico	Obbligatorio	Identificativo tag.

4.1.2 Descrizione Relazione

Relazione “ha_anagrafica”	
Descrizione	Informazioni che individuano, per ogni utente, la sua rispettiva anagrafica.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(1, 1)
Anagrafica	(1, 1)
Attributi	NULL

Relazione “ha_pubblicato”	
Descrizione	Informazioni che individuano, per ogni utente, i suoi rispettivi commenti pubblicati.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Commento	(0, 1)
Attributi	NULL

Relazione “ha_postato”	
Descrizione	Informazioni che individuano, per ogni utente, i suoi rispettivi post pubblicati.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Post	(0, 1)
Attributi	NULL

Relazione “ha_commentato”	
Descrizione	Informazioni che individuano, per ogni post, i rispettivi commenti.
Entità coinvolte	
Entità	**Cardinalità**
Commento	(1, 1)
Post	(0, N)
Attributi	NULL

Relazione “ha_tag”	
Descrizione	Informazioni che individuano, per ogni post, i rispettivi hashtag.
Entità coinvolte	
Entità	**Cardinalità**
Post	(0, N)
Tag	(1, N)
Attributi	NULL

Relazione “mittente”	
Descrizione	Informazioni che individuano, per ogni messaggio, privato l’utente che ha inviato il messaggio.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
PM	(0, 1)
Attributi	NULL

Relazione “destinatario”	
Descrizione	Informazioni che individuano, per ogni messaggio, privato l’utente a cui è destinato il messaggio.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
PM	(0, 1)
Attributi	NULL

Relazione “ha_amico”	
Descrizione	Informazioni che individuano un legame di amicizia tra due utenti.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Amicizia	(1, 1)
Attributi	NULL

Relazione “è_amico”	
Descrizione	Informazioni che individuano un legame di amicizia tra due utenti.
Entità coinvolte	
Entità	**Cardinalità**
Profilo Utente	(0, N)
Amicizia	(1, 1)
Attributi	NULL

4.2 Traduzione dello schema E/R

4.2.1 Traduzione entità

Anagrafica (id, nome, cognome, data_di_nascita)

Profilo_Utente (username, mail, password, privacy, anagrafica*)
Con vincolo di integrità referenziale tra l'attributo anagrafica e la relazione Anagrafica.

Amicizia (è_amico*, ha_amico*)
Con vincolo di integrità referenziale tra l'attributo è amico e la relazione Profilo Utente e tra l'attributo ha amico e la relazione Profilo Utente.

PM (id, mittente*, destinatario*, testo, data[1], visualizzato)
Con vincolo di integrità referenziale tra l'attributo mittente e la relazione Profilo Utente e tra l'attributo destinatario e la relazione Profilo Utente.

Commento (id, testo, data[1], profilo*, post*)
Con vincolo di integrità referenziale tra l'attributo profilo e la relazione Profilo Utente e tra l'attributo post e la relazione Post.

Post (id, testo, data[1], profilo*)
Con vincolo di integrità referenziale tra l'attributo profilo e la relazione Profilo Utente.

4.2.2 Traduzione Relazioni

Post_Tag (tag*, post*)
Con vincolo di integrità referenziale tra l'attributo tag e la relazione Tag e tra l'attributo post e la relazione Post.

[1] Per questioni di ottimizzazione, si è reso necessario accorpare gli attributi *data* e *orario* in un unico tipo di dato *TIMESTAMP*

5 Progettazione Fisica

5.1 Definizione dello schema della base di dati

5.1.1 Script SQL-DDL

5.1.1.1 Tabelle

```
--
-- Struttura della tabella `Amicizia`
--

CREATE TABLE `Amicizia` (
  `è_amico` varchar(64) NOT NULL,
  `ha_amico` varchar(64) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Struttura della tabella `Anagrafica`
--

CREATE TABLE `Anagrafica` (
  `id` int(11) NOT NULL,
  `nome` tinytext NOT NULL,
  `cognome` tinytext NOT NULL,
  `data_di_nascita` date NOT NULL,
  `profilo` varchar(64) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Struttura della tabella `Commento`
--

CREATE TABLE `Commento` (
  `id` int(11) NOT NULL,
  `testo` text NOT NULL,
  `data` timestamp NOT NULL DEFAULT current_timestamp(),
  `profilo` varchar(64) DEFAULT NULL,
  `post` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------
```

```sql
--
-- Struttura della tabella `PM`
--

CREATE TABLE `PM` (
  `id` int(11) NOT NULL,
  `mittente` varchar(64) DEFAULT NULL,
  `destinatario` varchar(64) DEFAULT NULL,
  `testo` text NOT NULL,
  `data` timestamp NOT NULL DEFAULT current_timestamp(),
  `visualizzato` tinyint(1) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Struttura della tabella `Post`
--

CREATE TABLE `Post` (
  `id` int(11) NOT NULL,
  `testo` text NOT NULL,
  `data` timestamp NOT NULL DEFAULT current_timestamp(),
  `profilo` varchar(64) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Struttura della tabella `Post_Tag`
--

CREATE TABLE `Post_Tag` (
  `tag` varchar(32) NOT NULL,
  `post` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------
```

```
--
-- Struttura della tabella `Profilo_Utente`
--

CREATE TABLE `Profilo_Utente` (
  `username` varchar(64) NOT NULL,
  `mail` varchar(64) NOT NULL,
  `password` varchar(64) NOT NULL,
  `privacy` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Struttura della tabella `Tag`
--

CREATE TABLE `Tag` (
  `hashtag` varchar(32) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5.1.1.2 Indici

```
--
-- Indici per le tabelle `Amicizia`
--
ALTER TABLE `Amicizia`
  ADD PRIMARY KEY (`è_amico`,`ha_amico`),
  ADD KEY `amico_v1` (`ha_amico`);

--
-- Indici per le tabelle `Anagrafica`
--
ALTER TABLE `Anagrafica`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `profilo` (`profilo`);

--
-- Indici per le tabelle `Commento`
--
ALTER TABLE `Commento`
  ADD PRIMARY KEY (`id`),
  ADD KEY `commento_post` (`post`),
  ADD KEY `commento_profilo` (`profilo`);
```

```sql
--
-- Indici per le tabelle `PM`
--
ALTER TABLE `PM`
  ADD PRIMARY KEY (`id`),
  ADD KEY `PM_ibfk_1` (`mittente`),
  ADD KEY `PM_ibfk_2` (`destinatario`);

--
-- Indici per le tabelle `Post`
--
ALTER TABLE `Post`
  ADD PRIMARY KEY (`id`),
  ADD KEY `post_profilo` (`profilo`);

--
-- Indici per le tabelle `Post_Tag`
--
ALTER TABLE `Post_Tag`
  ADD PRIMARY KEY (`tag`,`post`),
  ADD KEY `post_tag_v1` (`post`);

--
-- Indici per le tabelle `Profilo_Utente`
--
ALTER TABLE `Profilo_Utente`
  ADD PRIMARY KEY (`username`),
  ADD UNIQUE KEY `mail` (`mail`);

--
-- Indici per le tabelle `Tag`
--
ALTER TABLE `Tag`
  ADD PRIMARY KEY (`hashtag`);
```

5.1.1.3 AUTO_INCREMENT

```
--
-- AUTO_INCREMENT per la tabella `Anagrafica`
--
ALTER TABLE `Anagrafica`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT per la tabella `Commento`
--
ALTER TABLE `Commento`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT per la tabella `PM`
--
ALTER TABLE `PM`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT per la tabella `Post`
--
ALTER TABLE `Post`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

5.1.1.4 Vincoli

```
--
-- Limiti per la tabella `Amicizia`
--
ALTER TABLE `Amicizia`
  ADD CONSTRAINT `amico_v1` FOREIGN KEY (`ha_amico`) REFERENCES `Profilo_Utente` (`user
name`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `amico_v2` FOREIGN KEY (`è_amico`) REFERENCES `Profilo_Utente` (`usern
ame`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Limiti per la tabella `Anagrafica`
--
ALTER TABLE `Anagrafica`
  ADD CONSTRAINT `Anagrafica_ibfk_1` FOREIGN KEY (`profilo`) REFERENCES `Profilo_Utente
` (`username`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
--
-- Limiti per la tabella `Commento`
--
ALTER TABLE `Commento`
  ADD CONSTRAINT `commento_post` FOREIGN KEY (`post`) REFERENCES `Post` (`id`) ON DELET
E CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `commento_profilo` FOREIGN KEY (`profilo`) REFERENCES `Profilo_Utente`
 (`username`) ON DELETE SET NULL ON UPDATE CASCADE;


--
-- Limiti per la tabella `PM`
--
ALTER TABLE `PM`
  ADD CONSTRAINT `PM_ibfk_1` FOREIGN KEY (`mittente`) REFERENCES `Profilo_Utente` (`use
rname`) ON DELETE SET NULL ON UPDATE CASCADE,
  ADD CONSTRAINT `PM_ibfk_2` FOREIGN KEY (`destinatario`) REFERENCES `Profilo_Utente` (
`username`) ON DELETE SET NULL ON UPDATE CASCADE;

--
-- Limiti per la tabella `Post`
--
ALTER TABLE `Post`
  ADD CONSTRAINT `post_profilo` FOREIGN KEY (`profilo`) REFERENCES `Profilo_Utente` (`u
sername`) ON DELETE SET NULL ON UPDATE CASCADE;

--
-- Limiti per la tabella `Post_Tag`
--
ALTER TABLE `Post_Tag`
  ADD CONSTRAINT `post_tag_v1` FOREIGN KEY (`post`) REFERENCES `Post` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `post_tag_v2` FOREIGN KEY (`tag`) REFERENCES `Tag` (`hashtag`) ON DELE
TE CASCADE ON UPDATE CASCADE;
COMMIT;
```

5.1.1.5 Trigger

```
-- CheckComment.sql
-- 6. Controllare che se un profilo è privato allora solo un amico può commentare.

CREATE TRIGGER CheckComment BEFORE INSERT ON Commento
FOR EACH ROW
BEGIN

    DECLARE owner VARCHAR(64);

    SET owner = GetPostOwner(NEW.post);



    IF IsPrivate(owner) AND NOT IsFriend(owner, NEW.profilo) THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: Solo gli amici possono commentare un profilo privat
o";

    END IF;


    IF IsOffensive(NEW.testo) THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: Il commento risulta offensivo, pertanto non può ess
ere pubblicato";

    END IF;

END $$


-- CheckPost.sql
-- Controlla la validità di un post.

CREATE TRIGGER CheckPost BEFORE INSERT ON Post
FOR EACH ROW
BEGIN

    IF IsOffensive(NEW.testo) THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: Il post risulta offensivo, pertanto non può essere
pubblicato";

    END IF;

END $$
```

```sql
-- CheckSubscription.sql
-- Verifica requisiti di Password ed E-Mail

CREATE TRIGGER CheckSubscription BEFORE INSERT ON Profilo_Utente
FOR EACH ROW
BEGIN

    CALL CheckUsername(NEW.username);
    CALL CheckPassword(NEW.password);
    CALL CheckMail(NEW.mail);


    SET NEW.password = SHA2(NEW.password, 256);

END $$
```

5.1.1.6 Procedure

```sql
-- CheckMail.sql
-- Verifica la validità di una mail nel formato <xxx@xxx.xx>


CREATE PROCEDURE CheckMail(IN mail VARCHAR(64))
BEGIN

    IF LENGTH(mail) = 0 OR mail NOT LIKE '%_@_%.__%' THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: la mail inserita non è valida";

    END IF;

END $$
```

```sql
-- CheckPassword.sql
-- Verifica la validità di una password
--      # Minimo 8 caratteri, Massimo 16
--      # Almeno una lettera minuscola, una maiuscola e un numero.


CREATE PROCEDURE CheckPassword(IN password VARCHAR(64))
BEGIN

    IF LENGTH(password) < 8 OR LENGTH(password) > 16 THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: la lunghezza della password deve essere compresa tr
a 8 e 16 caratteri";

    END IF;

    IF NOT (password REGEXP BINARY "[a-z]" AND password REGEXP BINARY "[A-
Z]" AND password REGEXP "[0-9]") THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: la password deve contenere almeno una lettera maius
cola, minuscola e un numero";

    END IF;

END $$




-- DeleteOldData.sql
-- Elimina i contenuti più vecchi di 6 mesi, per ogni utente eliminato.


CREATE PROCEDURE DeleteOldData()
BEGIN

    DELETE FROM Commento
    WHERE ISNULL(Commento.profilo) AND TIMESTAMPDIFF(MONTH, Commento.data, CURRENT_TIME
STAMP()) > 6;

    DELETE FROM Post
    WHERE ISNULL(Post.profilo) AND TIMESTAMPDIFF(MONTH, Post.data, CURRENT_TIMESTAMP())
 > 6;

    DELETE FROM PM
    WHERE (ISNULL(PM.mittente) OR ISNULL(PM.destinatario)) AND TIMESTAMPDIFF(MONTH, PM.
data, CURRENT_TIMESTAMP()) > 6;

END $$
```

```sql
-- RetrieveMementoPosts.sql
- Rende disponibile all'interno della tabella "Memento" i post giornalieri degli anni p
recedenti.


CREATE PROCEDURE RetrieveMementoPosts()
BEGIN

    DECLARE owner VARCHAR(64);
    DECLARE post INT;
    DECLARE done INT DEFAULT 0;


    DECLARE cur CURSOR FOR
        SELECT Post.profilo, Post.id
        FROM Post
        WHERE TIMESTAMPDIFF(YEAR, Post.data, CURRENT_TIMESTAMP()) >= 1 AND MONTH(CURREN
T_TIMESTAMP()) = MONTH(Post.data) AND DAY(CURRENT_TIMESTAMP()) = DAY(Post.data);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;




    CREATE TABLE IF NOT EXISTS `Memento` (

        `username` VARCHAR(64) NOT NULL,
        `post` INT NOT NULL,

        PRIMARY KEY (`username`, `post`),

        FOREIGN KEY (`username`) REFERENCES Profilo_Utente(username) ON DELETE CASCADE,
        FOREIGN KEY (`post`) REFERENCES Post(id) ON DELETE CASCADE

    ) ENGINE = InnoDB;

    DELETE FROM `Memento`;




    OPEN cur;

        RetrievePosts: LOOP

            FETCH NEXT FROM cur INTO owner, post;

            IF done = 1 THEN
                LEAVE RetrievePosts;
            END IF;
```

```
            INSERT INTO Memento (username, post) VALUES (owner, post);

        END LOOP RetrievePosts;

    CLOSE cur;


END $$



-- CheckUsername.sql
-- Verifica la validità di un username.


CREATE PROCEDURE CheckUsername(IN username VARCHAR(64))
BEGIN


    IF username REGEXP "[^[:alnum:]]" THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Errore: l'username deve contenere solo caratteri alfanumeri
ci";

    END IF;

END $$
```

5.1.1.7 Funzioni

```
-- GetPostOwners.sql
-- Ottiene l'id del profilo che ha pubblicato il post dato in input.


CREATE FUNCTION GetPostOwner(post INT) RETURNS VARCHAR(64) DETERMINISTIC
BEGIN

    DECLARE owner VARCHAR(64);


    SELECT Profilo_Utente.username
    INTO owner
    FROM Profilo_Utente, Post
    WHERE Post.profilo = Profilo_Utente.username AND Post.id = post;


    RETURN owner;

END $$
```

```sql
-- IsFriend.sql
-- Verifica il legame di amicizia tra due profili utente.


CREATE FUNCTION IsFriend(u1 VARCHAR(64), u2 VARCHAR(64)) RETURNS BOOLEAN DETERMINISTIC
BEGIN


    DECLARE friend BOOLEAN;


    IF u1 = u2 THEN

        SET friend = 1;

    ELSE

        SET friend = CASE WHEN EXISTS (
            (
                SELECT
                    Amicizia.ha_amico
                FROM
                    Amicizia
                WHERE
                    Amicizia.è_amico = u1 AND Amicizia.ha_amico = u2

            UNION

                SELECT
                    Amicizia.è_amico
                FROM
                    Amicizia
                WHERE
                    Amicizia.è_amico = u2 AND Amicizia.ha_amico = u1

            )
        ) THEN 1 ELSE 0 END;

    END IF;


    RETURN friend;

END $$
```

```sql
-- IsPrivate.sql
-- Verifica che un dato profilo utente sia privato.


CREATE FUNCTION IsPrivate(profile VARCHAR(64)) RETURNS BOOLEAN DETERMINISTIC
BEGIN

    DECLARE privacy BOOLEAN;

    SELECT Profilo_Utente.privacy
    INTO privacy
    FROM Profilo_Utente
    WHERE Profilo_Utente.username = profile;

    RETURN privacy;

END $$




-- IsOffensive.sql
-- Verifica la presenza di contenuti offensivi.


CREATE FUNCTION IsOffensive(content TEXT) RETURNS BOOLEAN DETERMINISTIC
BEGIN

    DECLARE offensive BOOLEAN DEFAULT 0;


    IF INSTR(content, "parola_brutta_brutta") <> 0 THEN

        SET offensive = 1;

    END IF;


    RETURN offensive;

END $$
```

5.1.1.8 Eventi

```
-- Memento.sql
-
- Evento giornaliero che rende disponibili, per ciascun utente, i post pubblicati negli
 anni precedenti.


CREATE EVENT Memento
    ON SCHEDULE EVERY 1 DAY
    STARTS '2020-08-05 00:00:00'
    COMMENT 'Alle 0:00 di ogni giorno si rendano disponibili, per ciascun utente, i pos
t pubblicati in quel giorno negli anni precedenti.'
DO
CALL
    RetrieveMementoPosts;

$$
```

```
-- Cleaner.sql
-- Evento mensile che elimina i dati spazzatura.


CREATE EVENT Cleaner
    ON SCHEDULE EVERY 1 MONTH
    STARTS '2020-08-05 00:00:00'
    COMMENT 'Ogni mese effettua la pulizia dei dati spazzatura.'
DO
CALL
    DeleteOldData;

$$
```

5.1.1.9 Viste

```
-- Post_Commento.sql
-
- Vista che mette in releazione i post di ogni profilo e il relativo numero di commenti
.


CREATE VIEW Post_Commento (post, profilo, privacy, num_commenti) AS
SELECT
    Post.id,
    Post.profilo,
    Profilo_Utente.privacy,
    (SELECT COUNT(0) FROM Commento WHERE Commento.post = Post.id) AS num_commenti
FROM
    Post,
    Profilo_Utente
WHERE
    Post.profilo = Profilo_Utente.username;


$$
```

```sql
-- Profilo_Amici.sql
-- Vista che mette in relazione un profilo pubblico e il suo relativo numero di amici.


CREATE VIEW Profilo_Amici(profilo_pubblico, num_amici) AS
SELECT DISTINCT profilo, SUM(num_amici) FROM
    (
        SELECT
            Amicizia.è_amico AS profilo,
            COUNT(0) AS num_amici
        FROM
            Amicizia,
            Profilo_Utente
        WHERE
            Profilo_Utente.username = Amicizia.è_amico AND Profilo_Utente.privacy = 0
        GROUP BY
            Amicizia.è_amico

    UNION

        SELECT
            Amicizia.ha_amico AS profilo,
            COUNT(0) AS num_amici
        FROM
            Amicizia,
            Profilo_Utente
        WHERE
            Profilo_Utente.username = Amicizia.ha_amico AND Profilo_Utente.privacy = 0
        GROUP BY
            Amicizia.ha_amico
    ) AS F

GROUP BY
    profilo;


$$
```

5.2 Definizione delle interrogazioni per la visualizzazione dei dati

5.2.1 Script SQL-DML: Analisi Puntuali

```
-- 1. Aggiornare il profilo dell'utente 'Tizio' e renderlo pubblico.

UPDATE
    Profilo_Utente,
    Anagrafica
SET
    Profilo_Utente.privacy = 0
WHERE
    Profilo_Utente.username = Anagrafica.profilo AND Anagrafica.nome = "Tizio"
```

```
-- 2. Trovare gli amici dell'utente 'Caio'.

SELECT DISTINCT profilo FROM
    (
        SELECT
            Amicizia.ha_amico AS profilo
        FROM
            Amicizia,
            Profilo_Utente,
            Anagrafica
        WHERE
            Profilo_Utente.username = Amicizia.è_amico AND Profilo_Utente.username = An
agrafica.profilo AND Anagrafica.nome = "Caio"

    UNION

        SELECT
            Amicizia.è_amico AS profilo
        FROM
            Amicizia,
            Profilo_Utente,
            Anagrafica
        WHERE
            Profilo_Utente.username = Amicizia.ha_amico AND Profilo_Utente.username = A
nagrafica.profilo AND Anagrafica.nome = "Caio"
    ) AS F
```

```sql
-- 3. Trovare i profili pubblici con meno di 50 amici.

SELECT
    Profilo_Amici.profilo_pubblico
FROM
    Profilo_Amici
WHERE
    Profilo_Amici.num_amici < 50
```

5.2.2 Script SQL-DML: Analisi aggregate

```sql
-- 4. Trovare i post dei profili pubblici con il minor numero di commenti.

SELECT
    Post_Commento.id
FROM
    Post_Commento
WHERE
    Post_Commento.privacy = 0 AND Post_Commento.num_commenti = (
        SELECT
            MIN(Post_Commento.num_commenti)
        FROM
            Post_Commento
        WHERE
            Post_Commento.privacy = 0
    )
```

```sql
-- 5. Trovare per ogni profilo i post con il più alto numero di commenti.

SELECT
    Post_Commento.id
FROM
    Post_Commento
WHERE
    num_commenti = (
        SELECT
            MAX(PC.num_commenti)
        FROM
            Post_Commento AS PC
        WHERE
            Post_Commento.profilo = PC.profilo
    )
```