Basi di Dati

Progetto A.A. 2022/2023

CHAT MULTICANALE

0268273

Davide Leoni

Indice

1. Descrizione del Minimondo

Si vuole realizzare un sistema informativo per consentire ai lavoratori di una azienda di scambiare messaggi legati ai progetti a cui stanno attualmente lavorando.
Il sistema prevede tre livelli di utenza: gli amministratori, i capi progetto, i dipendenti.
Gli amministratori hanno la possibilità di gestire quali utenti sono capi progetto.
I capi progetto possono creare un numero arbitrario di canali di comunicazione ed invitare al loro interno tutti i dipendenti che cooperano sulle attività del progetto.
All'interno di un canale gli utenti possono inviare messaggi che possono essere letti da tutti gli altri appartenenti al canale. I messaggi sono organizzati in pagine e gli utenti possono visualizzare, una per una, le pagine della conversazione.
Un utente del sistema ha la possibilità di rispondere pubblicamente in un canale. In questa risposta, può decidere di riferire un messaggio precedentemente inviato, così che il suo messaggio appaia come risposta ad una parte specifica della comunicazione. Allo stesso modo, partendo da un qualsiasi messaggio, l'utilizzatore può decidere di rispondere in modo privato. Questa risposta privata aprirà un canale di discussione privato tra lui e il mittente del messaggio cui si sta rispondendo. I project manager possono sempre accedere (in sola lettura) a tutte le discussioni private nate nei canali dei progetti di cui sono responsabili.
Un utente può accedere in qualsiasi momento a tutti i canali di cui fa parte, e a tutte le conversazioni private (organizzate per canale) che ha creato o in cui è stato coinvolto.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
1	Lavoratori	Utenti	I lavoratori sono gli utenti del sistema
6	Dipendenti	Utenti	I dipendenti sono gli utenti del sistema
8	Gli altri	Utenti	Termine più preciso
12	Parte Specifica	Messaggio	Si riferisce a un messaggio mandato in precedenza
12	Comunicazione	Conversazione	Termine più preciso
13	Utilizzatore	Utente	Termine più preciso
14	Risposta privata	Messaggio	Termine più preciso
15	Lui	Utente	Termine più preciso
15	Mittente	Utente	Termine più preciso
15	Project Manager	Capo Progetto	È un utente con l'autorizzazione da Capo Progetto
16	Discussioni private	Conversazioni private	Termine più preciso

Specifica disambiguata

Si vuole realizzare un sistema informativo per consentire agli utenti di una azienda di scambiare messaggi legati ai progetti a cui stanno attualmente lavorando.
Il sistema prevede tre livelli di utenza: gli amministratori, i capi progetto, i dipendenti.
Gli amministratori hanno la possibilità di gestire quali utenti sono capi progetto.
I capi progetto possono creare un numero arbitrario di canali di comunicazione ed invitare al loro interno tutti gli utenti che cooperano sulle attività del progetto.
All'interno di un canale gli utenti possono inviare messaggi che possono essere letti da tutti gli altri utenti appartenenti al canale. I messaggi sono organizzati in pagine e gli utenti possono visualizzare, una per una, le pagine della conversazione.
Un utente del sistema ha la possibilità di rispondere pubblicamente in un canale. In questa risposta, può decidere di riferire un messaggio precedentemente inviato, così che il suo messaggio appaia come risposta ad un messaggio della conversazione. Allo stesso modo, partendo da un qualsiasi messaggio, l'utente può decidere di rispondere in modo privato. Questo messaggio privato aprirà un canale di discussione privato con l'utente che ha scritto il messaggio cui si sta rispondendo. I capi

progetto possono sempre accedere (in sola lettura) a tutte le conversazioni private nate nei canali dei progetti di cui sono responsabili.
Un utente può accedere in qualsiasi momento a tutti i canali di cui fa parte, e a tutte le conversazioni private (organizzate per canale) che ha creato o in cui è stato coinvolto.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Utenti	Chi utilizza il sistema		Messaggio, Progetto, Canale
Messaggi	Comunicazione tra Utenti		Utente, Canale, Progetto
Progetto	Progetto a cui partecipano Utenti		
Canale	Canale dove si scambiano messaggi tra utenti che partecipano allo stesso progetto		
Conversazione	Interna a canali, può essere privata tra due utenti o tra un gruppo di persone		

Raggruppamento dei requisiti in insiemi omogenei

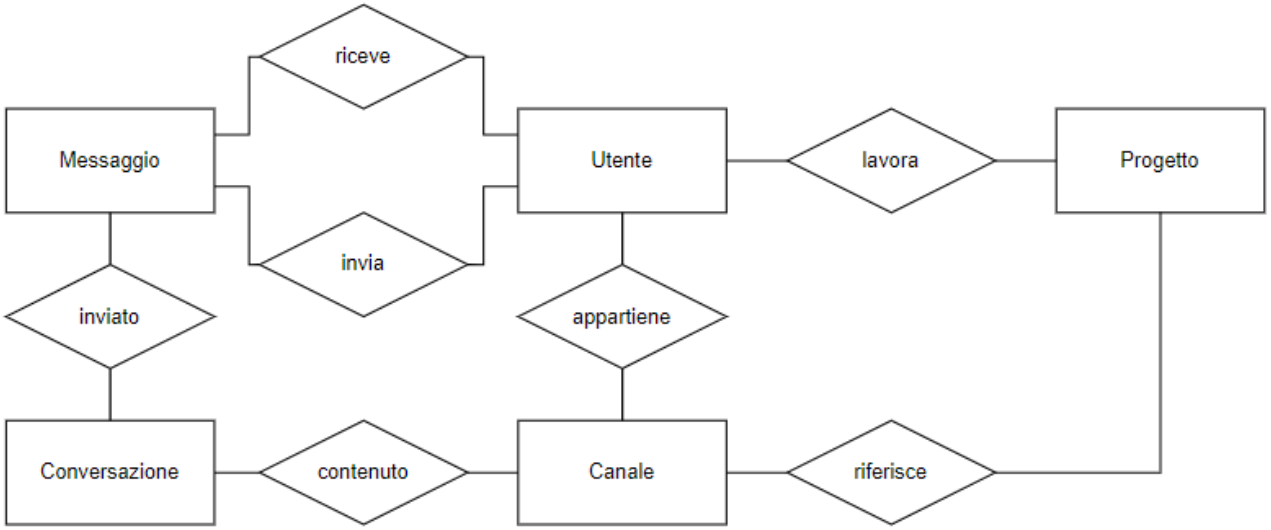
Utenti
Gli utenti possono scambiare messaggi legati ai progetti a cui lavorano. Il sistema prevede tre livelli di utenza: amministratori, capi progetto, dipendenti. Gli amministratori gestiscono quali utenti sono capi progetto. I capi progetto possono creare un numero arbitrario di canali ed invitare all'interno gli utenti che lavorano al progetto a cui i canali fanno riferimento. Tutti gli utenti appartenenti a un canale possono leggere i messaggi inviati al suo interno. Un utente può rispondere pubblicamente a uno specifico messaggio della conversazione, ma può anche rispondere a un messaggio in privato aprendo così un canale privato tra lui e il mittente del messaggio a cui ha risposto. I capi progetto

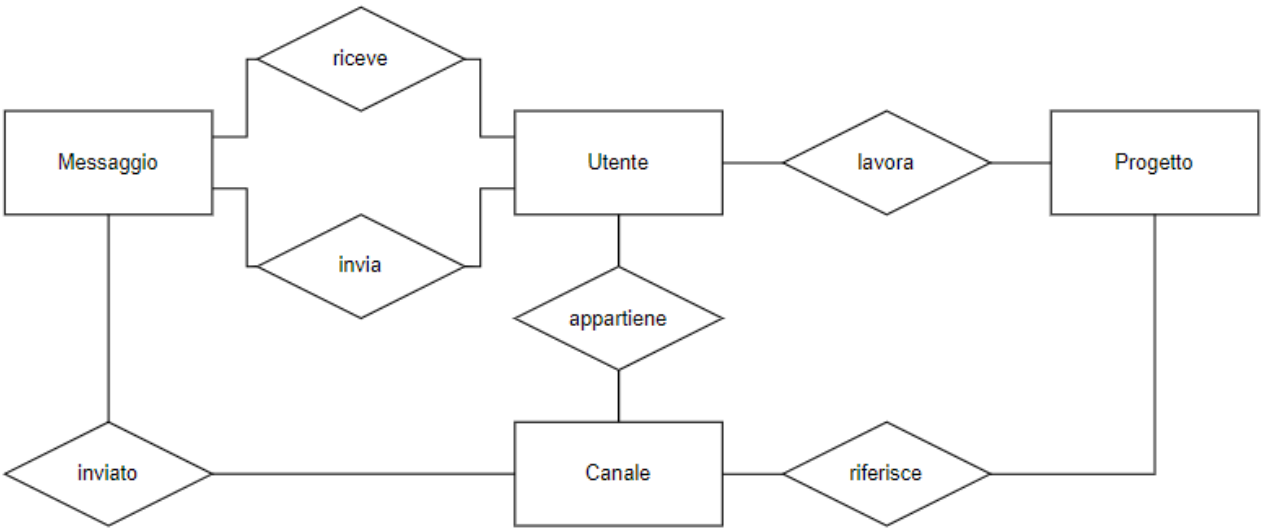
possono leggere tutte le conversazioni private nate nei canali legati a progetti che loro gestiscono. Un utente può accedere in qualsiasi momento a tutti di cui fa parte e a tutte le conversazioni che ha creato o in cui è stato coinvolto.
Messaggi
I messaggi sono organizzati in pagine della conversazione. Se un utente riferisce a un messaggio appare come risposta a quest'ultimo.
Canale
All'interno di un canale tutti gli utenti all'interno di esso possono scambiarsi messaggi. Partendo da qualsiasi messaggio un utente può rispondere in modo privato al messaggio di un altro utente avviando così un canale privato tra i due.

3. Progettazione concettuale

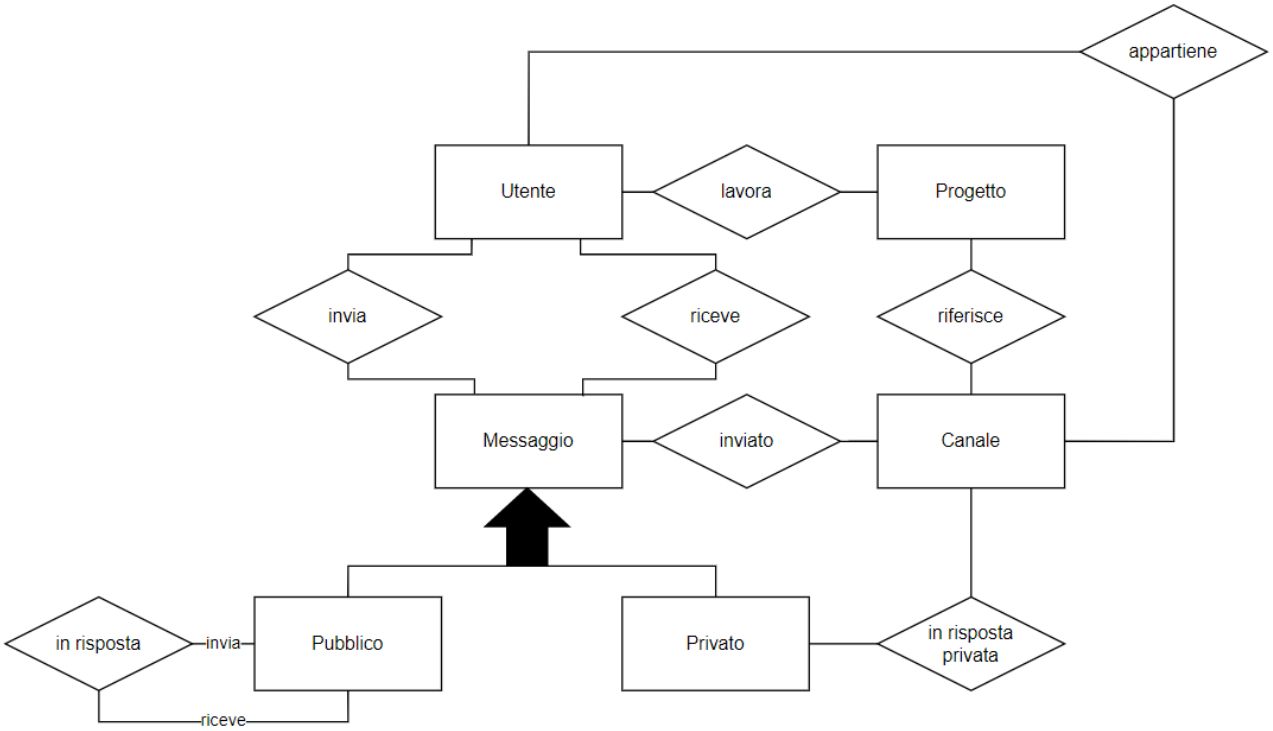
Costruzione dello schema E-R

1. Adottando una strategia mista ho inizialmente identificato i concetti chiave della specifica mettendo insieme uno scheletro iniziale di uno schema che sarà poi da raffinare.
2. Nel minimondo di riferimento le entità “Conversazione” e “Canale” sono concetti estremamente vicini quindi decido di accorpare i due concetti come entità unica.

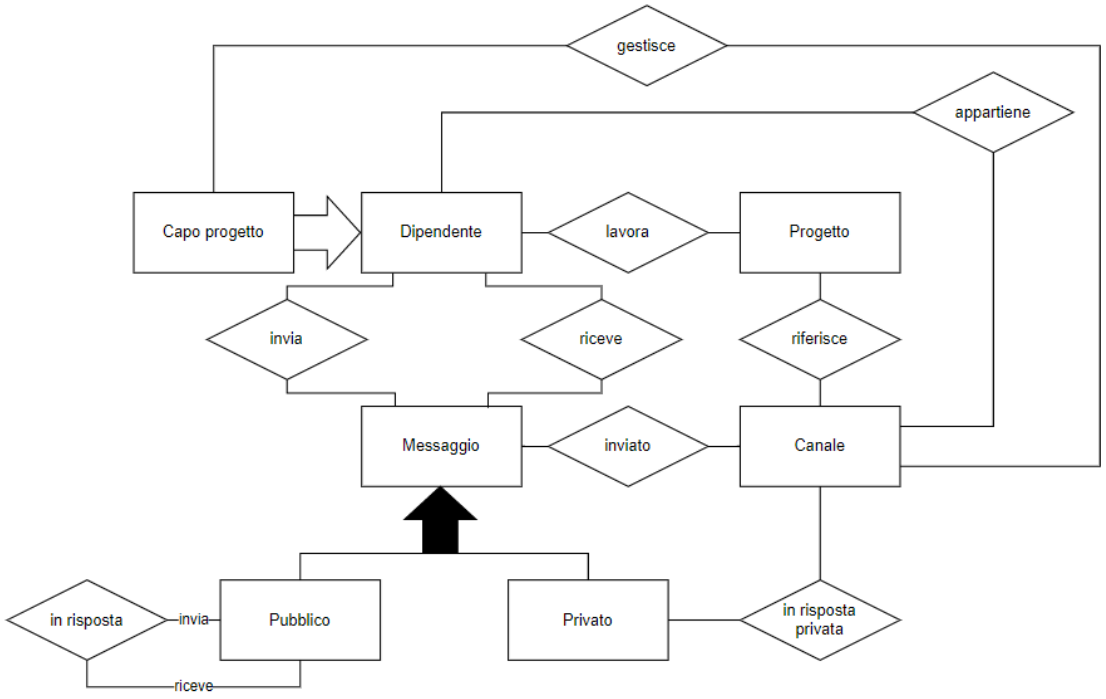




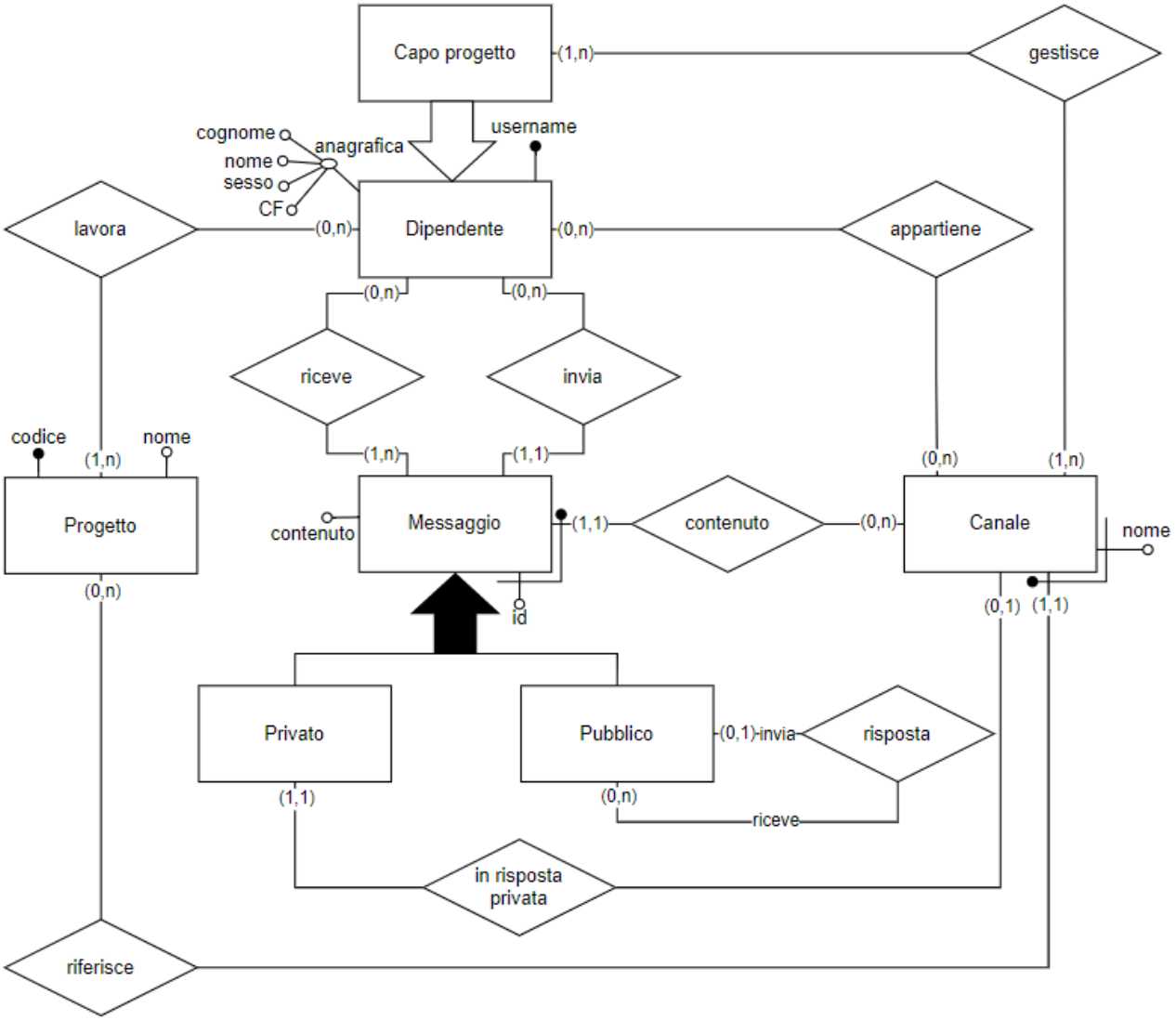
3. Mi accorgo che è necessario differenziare tra il messaggio inviato in un canale pubblico e uno in un canale privato poiché l'operazione di risposta a uno dei due scaturisce un risultato diverso.



4. È necessario dividere la tipologia di utente dato che un capo progetto può fare le stesse operazioni di un dipende più qualche aggiunta. È quindi possibile generalizzare "Capo Progetto" come "Dipendente" e aggiungere altre relazioni.



Integrazione finale



Regole aziendali

1. Se un utente tramite una risposta privata crea un canale privato verso un utente con cui ha già un canale privato aperto in precedenza il messaggio viene inviato in quel canale senza aprirne un altro.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Dipendente	Utente del sistema con privilegi da dipente	anagrafica(cognome, nome, sesso, CF)	username
Capo Progetto	Utente del sistema con privilegi da Capo progetto		username
Messaggio	Messaggio inviato all'interno di canali tra utenti	contenuto	id, contenuto
Pubblico	Messaggio inviato in un canale pubblico che può essere letto da tutti i partecipanti		id, contenuto
Privato	Messaggio inviato in un canale privato tra due utenti		id, contenuto
Canale	Canale all'interno di un progetto in cui vengono inviati i messaggi		nome, riferisce
Progetto	Progetto in cui lavorano gli utenti del sistema	nome	codice

4. Progettazione logica

Volume dei dati

Nell'analisi si ipotizza che il sistema mantenga i dati relativi ai progetti e ai messaggi per un periodo di circa 1 anno.

Si considera che a ogni progetto lavorano circa 10 persone, capo progetto compreso, e che ogni anno vengono portati a termine 10 progetti, ognuno con un proprio capo. Ogni utente può partecipare a più progetti e un capo progetto può esserlo di più progetti e può essere dipendente in altri.

Possiamo considerare quindi una media del 50% dei dipendenti del totale (10 progetti * 10 dipendenti a progetto). Ogni progetto nasce con un canale pubblico al cui interno sono presenti tutti coloro che lavorano al progetto e possono venirsi a creare fino a 10*9 canali privati tra gli utenti.

Assumiamo che in media ogni utente mandi circa 100 messaggi al giorno tra pubblici e privati (con una proporzione di circa 70/30), sono quindi 5000 messaggi inviati al giorno in totale.

Concetto nello schema	Tipo[1]	Volume atteso
Dipendente	E	50
Progetto	E	10
Capo Progetto	E	10
Canale	E	100
Messaggio	E	1.825.000
Messaggio Pubblico	E	1.277.500
Messaggio Privato	E	547.500
Appartiene	R	5.000
Gestisce	R	30
Lavora	R	500
Riferisce	R	100
Invia	R	1.825.000
Riceve	R	16.425.000
Risposta	R	912.500
In risposta apre	R	164.250

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
LD	Lista dipendenti appartenenti a un canale	50/ora

[1] Indicare con E le entità, con R le relazioni

LM	Lista messaggi in un canale	1/5min
IM	Invia messaggio in un canale	5.000/giorno
RM1	Risposta messaggio pubblico	1.000/giorno
RM2	Risposta messaggio privato	500/giorno
LC1	Lista canali in un progetto	50/ora
LC2	Lista canali a cui appartiene un utente in un progetto	50/ora
LP	Lista progetti di cui fa parte un utente	50/giorno
IP	Inserisci progetto	10/anno
ID1	Inserisci dipendente	50/anno
ID2	Inserisci dipendente in progetto	500/anno
ID3	Inserisci dipendente in canale	100/anno
IC1	Inserisci canale pubblico	30/anno
IC2	Inserisci canale privato	450/anno
LG	Login	50/giorno

Costo delle operazioni

LD - Lista dipendenti			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Canale	E	1	L
Appartiene	R	1	L
Dipendente	E	10	L
Costo totale = 13			
Eseguita 50/ora => 650/ora			

LM - Lista Messaggi contenuti in un canale			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Canale	E	1	L
Contenuto	R	1	L
Messaggi	E	1*1.000	L
Costo totale = 2.002			
Eseguita = 1/5 minuti => 2.002/5 minuti			

IM - Invia messaggio in un canale			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Invia	R	1	S
Messaggio	E	1	S
Contenuto	R	1	S
Costo totale = 6			
Eseguita 5.000/giorno => 30.000/giorno			

RM1 - Risposta messaggio pubblico			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Messaggio Pubblico	E	1	S
Risposta	R	1	S
Messaggio	E	1	S
Invia	R	1	S
Riceve	R	9	S
Contenuto	R	1	S
Costo = 28			
Eseguita 1.000/giorno => 28.000/giorno			

RM2 - Risposta messaggio privato			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Messaggio Privato	E	1	S
In risposta Apre	R	1	S
Canale	E	1	S
Messaggio	E	1	S
Contenuto	R	1	S
Invia	R	1	S
Riceve	R	1	S
Costo = 14			
Eseguita 500/giorno => 7000/giorno			

LC1 - Lista canali in un progetto			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Canale	E	10	L
Riferisce	R	1	L
Progetto	E	1	L
Costo totale = 12			
Eseguita 50/ora => 60/ora			

LC2 - Lista canali a cui appartiene un utente			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Appartiene	R	1	L
Canale	E	10	L
Riferisce	R	1	L
Costo totale = 12			
Eseguita 50/ora => 60/ora			

LP - Lista progetti di cui fa parte un utente			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Progetto	E	1	L
Lavora	R	2	L
Costo totale = 3			
Eseguita 50/giorno => 150/giorno			

IP - Inserisci progetto			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Progetto	E	1	S
Costo totale = 2			
Eseguita 10/anno => 20/anno			

ID1 - Inserisci dipendente			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Dipendente	E	1	S
Costo totale = 2			
Eseguita 50/anno => 100/anno			

ID2 - Inserisci dipendente in progetto			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Dipendente	E	1	S
Lavora	R	1	S
Costo totale = 4			
Eseguita 500/anno => 2000/anno			

ID3 - Inserisci dipendente in canale			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Dipendente	E	1	S
Appartiene	R	1	S
Costo totale = 4			
Eseguita 100/anno => 400/anno			

IC1 - Inserisci Canale Pubblico			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Canale	E	1	S
Appartiene	R	10	S
Costo totale = 22			
Eseguita 30/anno => 660/anno			

IC2 - Inserisci Canale Privato			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Canale	E	1	S
Appartiene	R	2	S
Costo totale = 6			
Eseguita 450/anno => 2700/anno			

LG - Login			
Concetto	**Costrutto**	**Accesso**	**Tipo**
Credenziali	E	1	L
Costo totale = 1			
Eseguita 50/giorno =>50/giorno			

Ristrutturazione dello schema E-R

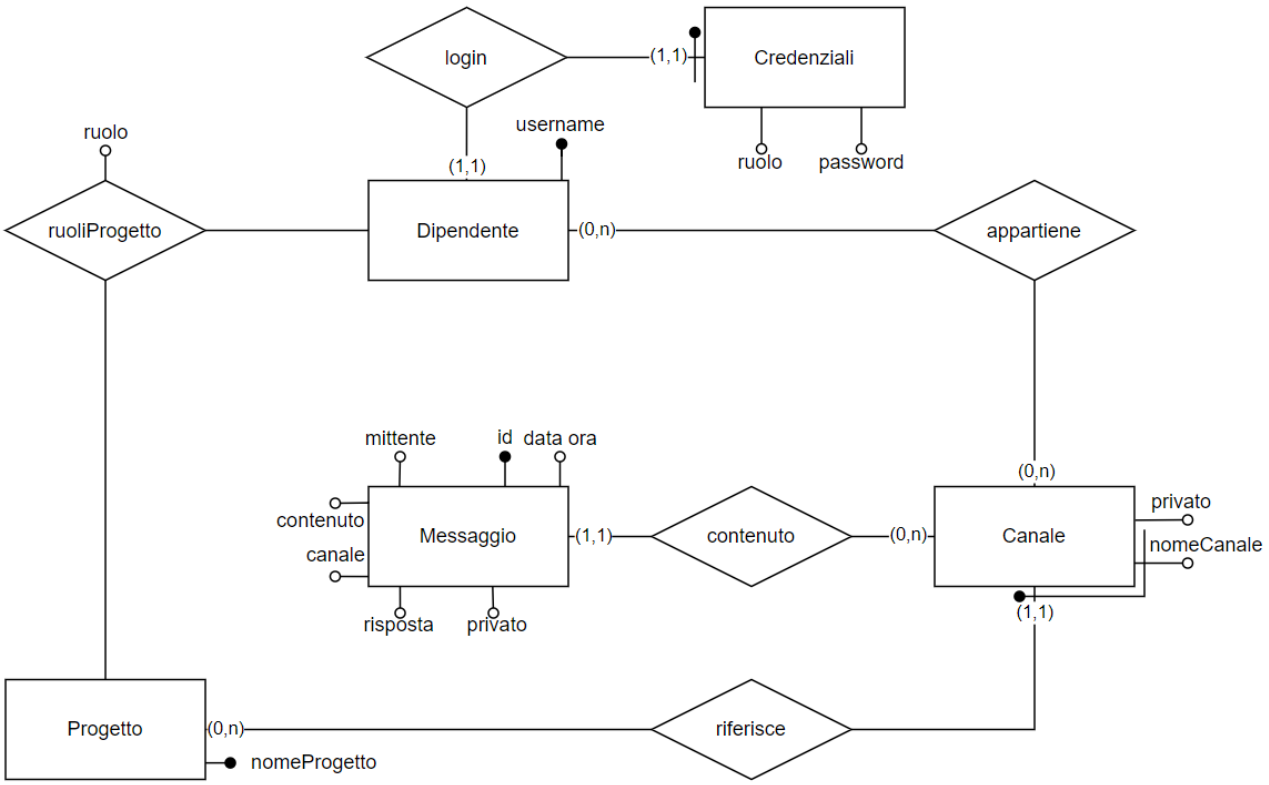
1. Relazione lavora navigabile da utenti - appartiene - canale - riferisce - progetto poiché se un utente appartiene a un canale all'interno di un progetto allora quell'utente lavorerà sicuramente a quel progetto, ma per eliminare anche la generalizzazione di Capo Progetto la relazione lavora cambia in una relazione "ruoliProgetto" tra Dipendente e Progetto che gestisce i ruoli all'interno di un progetto poiché si può verificare il caso in cui in un progetto un dipendente abbia ruolo di Capo Progetto e in un altro ruolo di dipendente.
2. La relazione creata sopra "ruoliProgetto" ha un attributo ruolo differenziato tra dipendente e capo progetto. Verrà quindi reificata in una entità debole con chiavi principali username di dipendente e progetto in progetto.
3. Gestione della generalizzazione tra messaggio, messaggio pubblico e messaggio privato. Viene assimilato tutto nell'entità padre Messaggio aggiungendo l'attributo booleano privato per identificare se un messaggio è privato o meno, e inoltre viene aggiunto l'attributo interno risposta che fa riferimento all'id di un altro messaggio per identificare di quale messaggio è risposta.
4. Le relazioni invia e riceve vengono riportate come attributo di messaggi tramite mittente poichè la relazione riceve è già identificata dal canale nel cui è inviato il messaggio.
5. Mi sono reso conto lavorando sul database che è necessaria l'aggiunta del campo nome progetto anche nella tabella del messaggio poiché possono presentarsi canali con lo stesso nome in progetti diversi, con questa struttura questo è impossibile

Trasformazione di attributi e identificatori

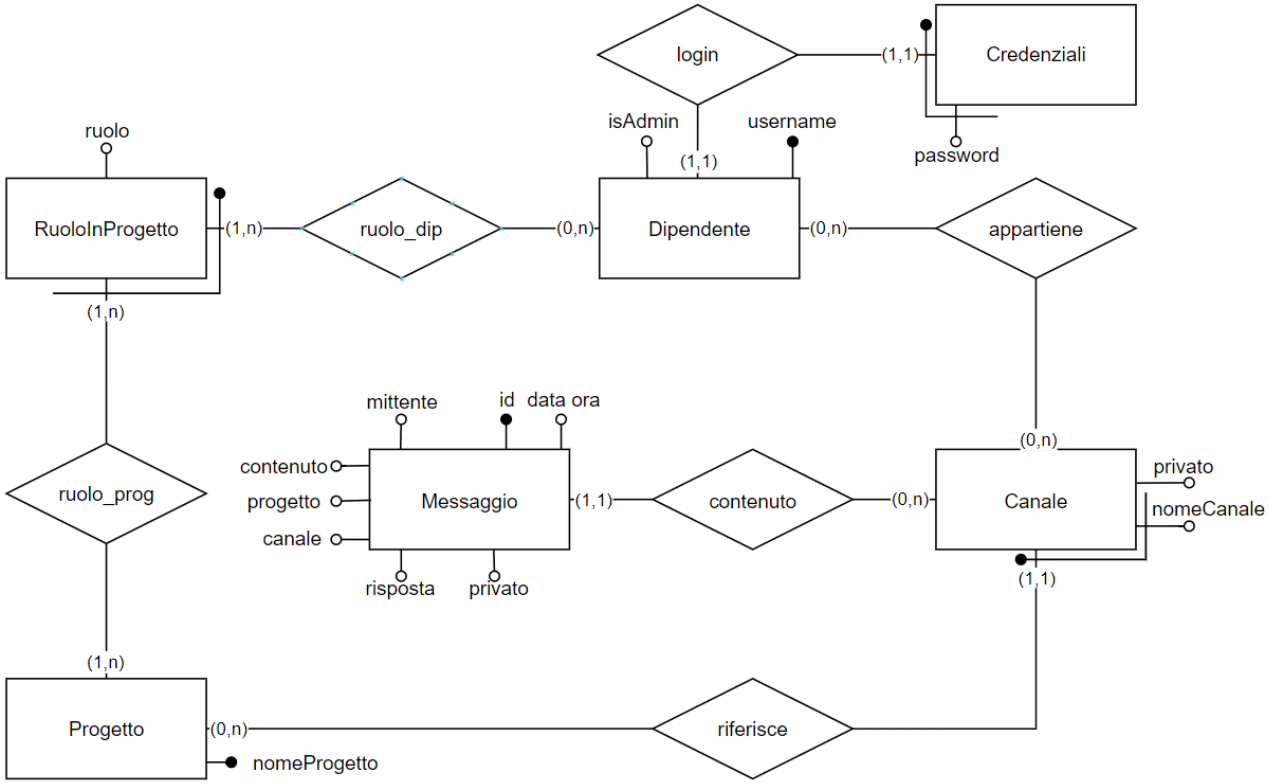
1. Identificatore debole id che fa riferimento al canale tramite la relazione contenuto viene reso identificatore forte e viene portato l'attributo canale all'interno dell'entità messaggio che fa riferimento al canale in cui è inviato.

Traduzione di entità e associazioni

Questo è lo schema con l'aggiunta della relazione ruoliProgetto



Reificando la relazione ruoloInProgetto abbiamo il seguente schema


login
(1,1)
Credenziali
isAdmin
username
password
(1,1)
ruolo
RuoloInProgetto
(1,n)
ruolo_dip
(0,n)
Dipendente
(0,n)
appartiene
(1,n)
mittente
id
data ora
contenuto
progetto
canale
Messaggio
(1,1)
contenuto
(0,n)
Canale
privato
nomeCanale
(0,n)
(1,1)
ruolo_prog
risposta
privato
(1,n)
Progetto
(0,n)
nomeProgetto
riferisce

Entità

- Dipendente (username)
- Canale (nomeCanale, nomeProgetto, privato)
- Messaggio (id, risposta, privato, canale, contenuto, mittente,progetto, data ora)
- Progetto (nomeProgetto)
- Credenziali (username, password)
- RuoloInProgetto (username, nomeProgetto, ruolo)

Relazioni

- appartiene (username, nomeCanale, nomeProgetto)
- contenuto (id, nomeCanale, nomeProgetto)
- riferisce (nomeProgetto, nomeCanale)
- ruolo_prog (nomeProgetto, username)
- ruolo_dip (nomeProgetto, username)

Normalizzazione del modello relazionale

Le tabelle precedenti sono tutte in 3FN poiché non mostrano dipendenze parziali tra attributi non chiave e chiave primaria.

5. Progettazione fisica

Utenti e privilegi

Sono previsti 4 Ruoli:

- Login:
  - GRANT sulla operazione: LG
- Dipendente
  - GRANT sulle operazioni: LD, LM, IM, RM1, RM2, LC1, LP, IC2
- Capo progetto
  - GRANT sulle operazioni: LD, LM, IM, RM1, RM2, LC1, LC2, LP, ID3, IC1, IC2
- Amministratore
  - GRANT sulle operazioni: IP, ID1, ID2

Strutture di memorizzazione

Tabella Dipendente		
Colonna	**Tipo di dato**	**Attributi**[2]
username	VARCHAR(45)	PK, NN

Tabella Canale		
Colonna	**Tipo di dato**	**Attributi**
nomeCanale	VARCHAR(45)	PK,NN
nomeProgetto	VARCHAR(45)	PK,NN
privato	BOOLEAN	NN

Tabella Progetto		
Colonna	**Tipo di dato**	**Attributi**
nomeprogetto	VARCHAR(45)	PK,NN

Tabella Credenziali		
Colonna	**Tipo di dato**	**Attributi**
username	= Utente(username)	PK,FK,NN
password	VARCHAR(45)	PK,NN

Tabella Messaggio		
Colonna	**Tipo di dato**	**Attributi**

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

id	INT	PK,NN,AI
privato	BOOLEAN	NN
risposta	= Messaggio(id)	NN,FK
canale	= Canale(nomecanale)	NN,FK
nomeProgetto	Progetto(nomeProgetto)	NN,FF
contenuto	TINYTEXT	NN
mittente	= Utente(username)	NN,FK
data	DATETIME	

Tabella della relazione Appartiene Utenti Canali		
Colonna	**Tipo di dato**	**Attributi**
username	= Utente(username)	PK,FK,NN
canale	= Canale(nomeCanale)	PK,FK,NN
progetto	= Progetto(nomeProgetto)	PK,FK,NN

Tabella RuoloInProgetto		
Colonna	**Tipo di dato**	**Attributi**
username	= Utente(username)	PK,FK,NN
nomeProgetto	= Progetto(nomeProgetto)	PK,FK,NN
ruolo	ENUM(“dipendente”,”capo progetto”,”amministratore”)	NN

Indici

Utilizzo un indice sugli attributi canale e nomeProgetto della tabella messaggi poiché l’operazione di ricerca e visualizzazione messaggi per canale (LM) è usata molto frequentemente

Tabella Messaggio	
Indice byID	**Tipo[3]:**
canale, nomeProgetto	UQ

[3] IDX = index, UQ = unique, FT = full text, PR = primary.

Trigger

Il seguente trigger verifica che all'inserimento di un messaggio privato il canale in cui si va ad inserire il messaggio sia effettivamente privato quindi con 2 partecipanti:

```
CREATE TRIGGER before_insert_messaggio
BEFORE INSERT ON messaggio
FOR EACH ROW
BEGIN
   DECLARE total_participants INT;
   DECLARE is_channel_private BOOLEAN;

   -- Controlla se il canale è privato
   SELECT privato INTO is_channel_private
   FROM canale
   WHERE nomeCanale = NEW.canale AND nomeProgetto = NEW.nomeProgetto;

   -- Se il messaggio è marcato come privato e anche il canale lo è
   IF NEW.privato = 1 AND is_channel_private = 1 THEN

      -- Controlla il numero di partecipanti nel canale
      SELECT COUNT(*) INTO total_participants
      FROM appartiene_utenti_canali
      WHERE nomeCanale = NEW.canale AND nomeProgetto = NEW.nomeProgetto;

      -- Se il canale non ha esattamente due partecipanti
      IF total_participants <> 2 THEN
          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il canale non ha esattamente due partecipanti';
      END IF;

      -- Assicurati che il mittente appartenga al canale
          IF NOT EXISTS (SELECT 1 FROM appartiene_utenti_canali WHERE nomeCanale = NEW.canale AND username = NEW.mittente) THEN
```

```
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il mittente non appartiene al canale privato';
        END IF;
    ELSEIF NEW.privato = 1 AND is_channel_private = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il messaggio è marcato come privato, ma il canale non lo è';
    ELSEIF NEW.privato = 0 AND is_channel_private = 1 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il canale è marcato come privato, ma il messaggio non lo è';
    END IF;
END !
```

Questo trigger verifica all'inserimento di un messaggio in risposta che esista effettivamente il messaggio a cui ci riferiamo rispondendo

```
DROP TRIGGER IF EXISTS verifica_esistenza_messaggio_risposta_before_insert!
CREATE TRIGGER verifica_esistenza_messaggio_risposta_before_insert
BEFORE INSERT ON messaggio
FOR EACH ROW
BEGIN
    DECLARE messaggio_esistente INT;

    -- Solo se 'risposta' è impostato, altrimenti non è una risposta
    IF NEW.risposta IS NOT NULL THEN
        SELECT COUNT(*)
        INTO messaggio_esistente
        FROM messaggio
        WHERE id = NEW.risposta;

        IF messaggio_esistente = 0 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il messaggio a cui si sta cercando di rispondere non esiste!';
        END IF;
```

```
   END IF;
END !
```

Il seguente trigger verifica che quando viene inserito un dipendente in un progetto da un amministratore il dipendente esista effettivamente nel database

```
CREATE TRIGGER verifica_esistenza_dipendente_before_insert
BEFORE INSERT ON RuoloInProgetto
FOR EACH ROW
BEGIN
   IF NOT EXISTS (SELECT 1 FROM dipendente WHERE username = NEW.username) THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Il dipendente specificato non esiste.';
   END IF;
END !
```

Il seguente trigger verifica che esista un progetto prima di assegnare un ruolo a un utente a quel progetto

```
CREATE TRIGGER verifica_esistenza_progetto_before_insert
BEFORE INSERT ON RuoloInProgetto
FOR EACH ROW
BEGIN
     IF NOT EXISTS (SELECT 1 FROM progetto WHERE nomeProgetto = NEW.nomeProgetto)
THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Il progetto specificato non esiste.';
   END IF;
END !
```

Il seguente trigger verifica l'esistenza di un utente prima di inserirlo in un canale

```
CREATE TRIGGER verifica_esistenza_utente_before_insert
BEFORE INSERT ON appartiene_utenti_canali
FOR EACH ROW
BEGIN
   IF NOT EXISTS (SELECT 1 FROM dipendente WHERE username = NEW.username) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il dipendente specificato non esiste.';
   END IF;
END !
```

Il seguente trigger verifica l'esistenza di un progetto prima di inserirci un canale all'interno

```
CREATE TRIGGER verifica_esistenza_progetto_before_insert_canale
BEFORE INSERT ON canale
FOR EACH ROW
BEGIN
   -- Verifica se il progetto associato al canale esiste
     IF NOT EXISTS (SELECT 1 FROM progetto WHERE nomeProgetto = NEW.nomeProgetto)
THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il progetto specificato non esiste.';
   END IF;
END !
```

Il seguente trigger verifica che quando un utente viene inserito in un progetto l'utente in questione non sia già presente in quel progetto con qualsiasi ruolo

```
CREATE TRIGGER before_insert_ruoloinprogetto
BEFORE INSERT ON RuoloInProgetto
FOR EACH ROW
```

```
BEGIN
   DECLARE utente_esistente INT;

   -- Controlla se l'utente è già associato al progetto
   SELECT COUNT(*) INTO utente_esistente
   FROM RuoloInProgetto
   WHERE username = NEW.username AND nomeProgetto = NEW.nomeProgetto;

   IF utente_esistente > 0 THEN
      -- Se l'utente è già associato al progetto, genera un errore
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Utente già all\'interno di questo progetto.';
   END IF;
END !
```

Eventi

Non sono stati implementati eventi.

Viste

Non sono state implementate viste.

Stored Procedures e transazioni

Procedura LD:

```
CREATE PROCEDURE ListaDipendentiInCanale(IN nome_del_canale VARCHAR(45), IN nome_del_progetto VARCHAR(45))
BEGIN
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

	SELECT a.username, r.ruolo
	FROM appartiene_utenti_canali a
	JOIN RuoloInProgetto r ON a.username = r.username AND a.nomeProgetto = r.nomeProgetto
	WHERE a.nomeCanale = nome_del_canale AND a.nomeProgetto = nome_del_progetto;
END!
```

utilizzo un livello di isolamento READ COMMITTED poiché è una procedura di sola lettura quindi previene la lettura di dati sporchi

Procedura LM

```
CREATE PROCEDURE ListaMessaggiInCanale(IN nome_del_canale VARCHAR(45), IN nome_del_progetto VARCHAR(45))
BEGIN
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

	SELECT m.*
	FROM messaggio m
	JOIN canale c ON m.canale = c.nomeCanale
	WHERE m.canale = nome_del_canale AND c.nomeProgetto = nome_del_progetto;
END !
```

utilizzo anche qui un livello di isolamento READ COMMITTED poiché è una procedura di sola lettura dei dati quindi previene la lettura di dati sporchi

Procedura IM e RM1

```
CREATE PROCEDURE InviaMessaggio(
   IN contenuto_messaggio TINYTEXT,
   IN risposta INT,
   IN is_privato BOOLEAN,
   IN nomeProgetto VARCHAR(45),
   IN mittente_username VARCHAR(45),
   IN nome_del_canale VARCHAR(45)
)
BEGIN
	DECLARE EXIT HANDLER FOR SQLEXCEPTION
   BEGIN
		ROLLBACK;
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Errore nell''invio del messaggio.';
   END;

	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
   START TRANSACTION;
```

```
   INSERT INTO messaggio(privato, canale, nomeProgetto, risposta, contenuto, mittente)
        VALUES(is_privato, nome_del_canale, nomeProgetto, risposta, contenuto_messaggio, mittente_username);

   COMMIT;
END !
```

utilizzo READ COMMITTED anche se è un inserimento così che le modifiche che apporta questa transazione siano visibili solo a lei stessa e non ad altre finché non vengono committate, SERIALIZABLE mi sembrava troppo eccessivo

Procedura RM2

```
CREATE PROCEDURE InviaRispostaPrivata(
   IN contenutoMessaggio TINYTEXT,
   IN usernameMittente VARCHAR(45),
   IN idRisposta INT,
   IN isPrivato BOOLEAN,
   IN InNomeProgetto VARCHAR(45)
)
BEGIN
   DECLARE canaleEsistente INT;
   DECLARE nomeCanaleEsistenze VARCHAR(45);
   DECLARE nomeDestinatario VARCHAR(45);

   DECLARE EXIT HANDLER FOR SQLEXCEPTION
   BEGIN
      ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Errore nell''invio della risposta privata.';
   END;
```

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;

-- Trova il nome dell'utente a cui stai rispondendo usando l'ID del messaggio di risposta
SELECT mittente INTO nomeDestinatario FROM messaggio WHERE id = idRisposta;

-- Verifica se il canale esiste in uno dei due formati
SELECT COUNT(*) INTO canaleEsistente
FROM canale
WHERE (((nomeCanale = CONCAT(usernameMittente, '_', nomeDestinatario))
      OR (nomeCanale = CONCAT(nomeDestinatario, '_', usernameMittente)))
     AND nomeProgetto = InNomeProgetto);

-- Se il canale non esiste, crealo
IF canaleEsistente = 0 THEN
   INSERT INTO canale (nomeCanale, privato, nomeProgetto)
   VALUES (CONCAT(usernameMittente, '_', nomeDestinatario), TRUE, InNomeProgetto);

   -- Aggiorna la tabella appartiene_utenti_canali
   INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
            VALUES (usernameMittente, CONCAT(usernameMittente, '_', nomeDestinatario),
InNomeProgetto);
   INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
            VALUES (nomeDestinatario, CONCAT(usernameMittente, '_', nomeDestinatario),
InNomeProgetto);
END IF;

SELECT nomeCanale INTO nomeCanaleEsistenze
FROM canale
WHERE (((nomeCanale = CONCAT(usernameMittente, '_', nomeDestinatario))
      OR (nomeCanale = CONCAT(nomeDestinatario, '_', usernameMittente)))
     AND nomeProgetto = InNomeProgetto);

-- Ora puoi inserire il messaggio
```

```
   INSERT INTO messaggio(privato, canale, nomeProgetto, contenuto, mittente, risposta)
                VALUES(isPrivato,nomeCanaleEsistenze, InNomeProgetto, contenutoMessaggio, usernameMittente, idRisposta);
   SELECT LAST_INSERT_ID();

   COMMIT;

END !
```

in questa procedura ho deciso di utilizzare un livello di isolamento maggiore come REPEATABLE READ per evitare problemi del tipo: più utenti provano a mandare messaggi privati allo stesso momento o la lettura di una canale non ancora creato da un'altra transazione. Questo livello assicura che se una transazione legge una riga nessun'altra la può modificare finché la transazione non è terminata

Procedura LC1 e LC2 implementate con la stessa procedura ma diversificate tramite il ruolo di chi la esegue

```
CREATE PROCEDURE ListaCanaliPerUtenteProgetto(
   IN input_nomeProgetto VARCHAR(45),
   IN input_username VARCHAR(45),
   IN input_ruolo ENUM('DIPENDENTE', 'CAPO_PROGETTO', 'AMMINISTRATORE')
)
BEGIN
        SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

        -- Se l'utente è un CAPO_PROGETTO
        IF input_ruolo = 'CAPO_PROGETTO' THEN
                SELECT
                        c.nomeCanale,
                        c.nomeProgetto,
                        c.Privato,
                        (auc.username IS NOT NULL) AS AppartieneUtente
```

```
        FROM canale c
        LEFT JOIN appartiene_utenti_canali auc ON c.nomeCanale = auc.nomeCanale AND
c.nomeProgetto = auc.nomeProgetto AND auc.username = input_username
        WHERE c.nomeProgetto = input_nomeProgetto;

    -- Se l'utente è un DIPENDENTE
    ELSEIF input_ruolo = 'DIPENDENTE' THEN
        SELECT
            c.nomeCanale,
            c.nomeProgetto,
            c.Privato
        FROM canale c
        JOIN appartiene_utenti_canali auc ON c.nomeCanale = auc.nomeCanale AND
c.nomeProgetto = auc.nomeProgetto
        WHERE auc.username = input_username AND auc.nomeProgetto =
input_nomeProgetto;

    -- Se l'utente è un AMMINISTRATORE (non restituisce canali in quanto esterno ai progetti)
    ELSEIF input_ruolo = 'AMMINISTRATORE' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'L\'amministratore non ha
accesso ai canali interni ai progetti.';
    END IF;
END !
```

qui viene utilizzato un livello di isolamento READ COMMITTED, poiché avviene una semplice lettura e si previene dalla lettura di dati sporchi

Procedura LP

```
BEGIN
    SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

    SELECT r.nomeProgetto, r.ruolo
```

```
        FROM RuoloInProgetto r
        WHERE r.username = input_username;
END !
```

Procedura IP

```
CREATE PROCEDURE inserisciNuovoProgetto(
  IN nome_nuovo_progetto VARCHAR(45),
  IN capo_progetto_username VARCHAR(45)
)
BEGIN
  DECLARE dipendente_exists INT;

  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    ROLLBACK;
    RESIGNAL;
  END;

  SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  -- Verifica se il capo progetto designato esiste
  SELECT COUNT(*) INTO dipendente_exists
  FROM dipendente
  WHERE username = capo_progetto_username;

  IF dipendente_exists = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Il dipendente specificato non esiste.';
  ELSE
    -- Creazione del nuovo progetto
    INSERT INTO progetto (nomeProgetto)
```

```
        VALUES (nome_nuovo_progetto);

        -- Assegna il ruolo di capo progetto all'utente designato per il progetto appena creato
        INSERT INTO RuoloInProgetto (username, nomeProgetto, ruolo)
        VALUES (capo_progetto_username, nome_nuovo_progetto, 'CAPO_PROGETTO');

        -- Creazione del canale pubblico associato al progetto appena creato
        INSERT INTO canale (privato, nomeCanale, nomeProgetto)
        VALUES (false, CONCAT(nome_nuovo_progetto, '_pubblico'), nome_nuovo_progetto);

        -- Inserisco il nuovo capo progetto nel canale appena creato
        INSERT INTO appartiene_utenti_canali (username, nomeCanale, nomeProgetto)
                VALUES (capo_progetto_username, CONCAT(nome_nuovo_progetto, '_pubblico'),
nome_nuovo_progetto);
    END IF;

    COMMIT;
END !
```

posso permettermi di usare READ COMMITTED anche se è un inserimento poiché è un inserimento che viene fatto molto raramente e difficilmente ci saranno problemi di concorrenza.

Procedura ID1

```
CREATE PROCEDURE InserisciNuovoUtente(
    IN nuovoUsername VARCHAR(45),
    IN nuovaPassword VARCHAR(45)
)
BEGIN

        DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
```

```
    RESIGNAL;
  END;

  SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  -- Inserimento in dipendente
  INSERT INTO dipendente(username) VALUES (nuovoUsername);

  -- Inserimento in credenziali
  INSERT INTO credenziali(username, password) VALUES (nuovoUsername, nuovaPassword);

  COMMIT;
END !
```

anche qui è un procedura che può fare l'amministratore e raramente verrà aggiunto un nuovo dipendente, e non serve controllare altre tabelle del database per effettuare l'inserimento quindi come livello di isolamento READ COMMITTED va più che bene

Procedura ID2

```
CREATE PROCEDURE InserisciUtenteInProgetto(
  IN inputUsername VARCHAR(45),
  IN nomeProgetto VARCHAR(45)
)
BEGIN

      DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    ROLLBACK;
    RESIGNAL;
  END;
```

```
    SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
    START TRANSACTION;

    -- Inserisci l'utente nel progetto con il ruolo specificato
    INSERT INTO RuoloInProgetto(username, nomeProgetto, ruolo)
    VALUES (inputUsername, nomeProgetto, 'DIPENDENTE');

    COMMIT;
END !
```

anche qui è una procedura che può essere utilizzata più spesso ma comunque non dovrebbe dare problemi di concorrenza potrebbe servire un REPEATABLE READ ma anche READ COMMITTED va bene

Procedura IC1

```
CREATE PROCEDURE inserisciCanalePubblico(
    IN nome_del_canale VARCHAR(45),
    IN nome_del_progetto VARCHAR(45),
    IN utenti_list VARCHAR(255)
)
BEGIN

        DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
    START TRANSACTION;

        INSERT INTO canale (nomeCanale, privato, nomeProgetto)
```

```
        VALUES (nome_del_canale, FALSE, nome_del_progetto);

    SET @s = utenti_list;
    WHILE LOCATE(',', @s) > 0 DO
        INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
        VALUES (SUBSTRING(@s, 1, LOCATE(',', @s) - 1), nome_del_canale, nome_del_progetto);
        SET @s = SUBSTRING(@s, LOCATE(',', @s) + 1);
    END WHILE;

    INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
    VALUES (@s, nome_del_canale, nome_del_progetto);

    COMMIT;
END !
```

Qui utilizzo un livello SERIALIZABLE che è il più alto. Ho optato per questo poiché anche se la procedura non effettua letture su dati esterni al di fuori dall'input quindi non dovrebbe esserci problema sulla lettura di dati sporchi, potrebbe presentarsi qualche problema se un'altra operazione può modificare i dati che sta maneggiando mentre è in corso la transazione. SERIALIZABLE fa in modo che se c'è questa procedura in corso nessun'altra verrà eseguita poiché garantisce che le transazioni verranno eseguite in modo sequenziale.

Procedura IC2

```
CREATE PROCEDURE inserisciCanalePrivato(
    IN nome_del_canale VARCHAR(45),
    IN nome_del_progetto VARCHAR(45),
    IN utenti_list VARCHAR(255)
)
BEGIN

        DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
```

```
    ROLLBACK;
    RESIGNAL;
  END;

  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  START TRANSACTION;

      INSERT INTO canale (nomeCanale, privato, nomeProgetto)
      VALUES (nome_del_canale, TRUE, nome_del_progetto);

  SET @s = utenti_list;
  WHILE LOCATE(',', @s) > 0 DO
    INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
    VALUES (SUBSTRING(@s, 1, LOCATE(',', @s) - 1), nome_del_canale, nome_del_progetto);
    SET @s = SUBSTRING(@s, LOCATE(',', @s) + 1);
  END WHILE;

  INSERT INTO appartiene_utenti_canali(username, nomeCanale, nomeProgetto)
  VALUES (@s, nome_del_canale, nome_del_progetto);

  COMMIT;
END !
```

Uguale alla precedente poiché fa la stessa cosa solo che il canale che si viene a creare è di tipo privato