

Progetto d'esame

Badi di Dati e Sistemi Informativi

**Gestione di un DB per Social Network:  
sviluppo di software Open Source**

Anno Accademico 2012/2013

Proposto da: **Alberto Arvizzigno**

Matricola n. 237953  
E-mail: [albertoa.arvizzigno@gmail.com](mailto:albertoa.arvizzigno@gmail.com)

---

## Indice

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| Indice.....                                                             | 2  |
| Formulazione e Analisi dei requisiti.....                               | 3  |
| Genesi del progetto.....                                                | 3  |
| Descrizione dei requisiti in linguaggio naturale:.....                  | 4  |
| Risoluzione delle ambiguità .....                                       | 5  |
| Nozioni basiche: Glossario dei termini e dei concetti fondamentali..... | 6  |
| Operazioni di AGGIORNAMENTO soddisfatte dal DB.....                     | 9  |
| Operazioni di INTERROGAZIONE al DB.....                                 | 9  |
| Progettazione concettuale.....                                          | 10 |
| Scelta della strategia di progetto .....                                | 16 |
| Progettazione logica.....                                               | 28 |
| Semplificazione.....                                                    | 38 |
| Generalizzazione delle gerarchie.....                                   | 38 |
| Attributi compositi.....                                                | 40 |
| Attributo multi valore.....                                             | 40 |
| Traduzione.....                                                         | 41 |
| Normalizzazione.....                                                    | 44 |
| Database MySQL.....                                                     | 50 |
| Query di aggiornamento:.....                                            | 59 |
| Query di interrogazione al DB.....                                      | 61 |
| Progettazione fisica.....                                               | 69 |
| Sviluppo di un modulo C tramite l'uso delle API di MySQL.....           | 74 |
| Interfaccia Utente.....                                                 | 75 |
| Codice Sorgente e sviluppo del programma.....                           | 78 |
| File: main_ospj.c.....                                                  | 78 |
| File: opensourcepj.c.....                                               | 79 |

---

## Formulazione e Analisi dei requisiti

### Genesi del progetto

Con il presente elaborato s'intende sviluppare un data base per Social Network tematico mirato alla progettazione di Software Open Source collaborativa tra differenti utenti.

Considerato l'utente che voglia ad esempio realizzare un'idea, per cui dispone di competenze nulle o parziali, una volta determinate le specifiche di sviluppo iniziali sarà possibile aprire a tutti lo sviluppo dell'idea tramite la registrazione ad un sito la cui gestione di dati e le correlazione tra utenti e competenze possono essere svolte da un DBMS.

Le specifiche che descrivono il progetto includeranno differenti ambiti di conoscenza poi supportati da utenti differenti e maggiormente specializzati che nel tempo decideranno di aderire al progetto stesso.

E' lecito pensare che nei processi di formazione dei team di sviluppo e dei diversi ruoli organizzativi, s'innescino dei meccanismi impliciti di formazione, di auto-selezione dei compiti in base alle capacità conoscitive di ogni singolo individuo.

Le espressioni di creatività diversificata sono derivanti dalla partecipazione di liberi programmatori che forniscono il loro supporto in un processo lento ma frequente e costante per l'ottenimento di un risultato finale.

L'utente che ha intenzione di partecipare, ad esempio per scopi di ricerca, ad un particolare progetto Open Source pubblicato sul sito, non dovrà fare altro che aderire apportando le proprie competenze settoriali al gruppo di sviluppo.

L'avanzamento del progetto evolve quindi attraverso gruppi di lavoro che costituiscono dei processi paralleli e concorrenti di elaborazione, per raggiungere il risultato finale proposto dalle specifiche tecniche iniziali.

Software di sviluppo impiegato:

-----  
L'elaborazione del progetto è stata effettuata tramite l'uso di alcuni tools di progettazione:

- *Dia 0.97.2* : per il design del modello ER
- *MySQL WorkBench 5.2.40* : per l'inserimento di dati in tabella, per il raffinamento delle Tabelle e per il test delle query di ricerca.
- *Code:Blocks* : IDE per lo sviluppo in C del programma d'implementazione del DB

La versione di MySQL impiegata è:

- *mysql Ver 14.14 Distrib 5.5.31, for debian-linux-gnu (x86\_64)*
-

---

## Descrizione dei requisiti in linguaggio naturale:

|    | <b>Descrizione dei requisiti</b>                                                                                                   |
|----|------------------------------------------------------------------------------------------------------------------------------------|
| 1  | Il database deve essere in grado di correlare <b>progetti software</b> e <b>partecipanti</b> iscritti al sito, ogni                |
| 2  | <b>partecipante</b> opererà in un settore di sviluppo software determinato in base alle competenze dichiarate.                     |
| 3  | Per inserire un nuovo progetto all'interno del DB sono necessarie alcune informazioni quali:                                       |
| 4  | il nome del progetto, la <b>data di creazione</b> , il nome dell' <b>ideatore</b> , una directory di lavoro condivisa,             |
| 5  | è inoltre richiesta una pianificazione progettuale che determini le competenze necessarie allo sviluppo.                           |
| 6  | La gestione del progetto verrà svolta da una o più <b>persone</b> , che assolvono degli incarichi in base alle                     |
| 7  | competenze che dichiarano di possedere durante la fase di registrazione.                                                           |
| 8  | Le competenze richieste dal progetto verranno poi definite in un elenco aggiornato dai manager progettisti.                        |
| 9  | A capo dell'intero progetto vi è il <b>gruppo</b> di cui fanno parte uno o più <b>utenti registrati</b> al sito, che               |
| 10 | si occuperà della suddivisione del progetto in sotto progetti di più semplice elaborazione                                         |
| 11 | e della definizione delle competenze necessarie per l'avanzamento di ogni singolo sotto progetto.                                  |
| 12 | Il progetto, una volta suddiviso in <b>sotto problemi</b> viene gestito da differenti <b>team</b> ai quali                         |
| 13 | è assegnato un <b>capo</b> che sarà anche il responsabile tecnico di sviluppo del suo sotto progetto.                              |
| 14 | Ogni utente, per potersi registrare al sito, è necessario che inserisca i propri dati personali                                    |
| 15 | quali: nome, cognome, <b>età</b> , cittadinanza, madrelingua, nonché informazioni sulle                                            |
| 16 | capacità tecniche specifiche e i dati di registrazione che verranno utilizzati per l'accesso                                       |
| 17 | al sito quali nickname e password. L'area di competenza dichiarata dall'utente permetterà anche                                    |
| 18 | di determinare l'incarico progettuale o sotto progettuale quale: project manager, team manager,                                    |
| 19 | team component o normal user. Ogni utente con un particolare incarico assegnato acquisirà anche gli                                |
| 20 | accessi in lettura o scrittura necessari per poter lavorare su una directory condivisa di sotto-progetto.                          |
| 21 | Ogni progetto, come già evidenziato, deve essere opportunamente suddiviso in sotto progetti più semplici,                          |
| 22 | tale suddivisione viene stabilita dal <b>gruppo di management</b> , in base alle necessità tecniche rilevate.                      |
| 23 | Ad ogni sotto progetto farà capo <b>un utente registrato</b> a cui viene assegnata quella posizione                                |
| 24 | solo dopo aver avuto il consenso dei manager progettisti che si occupano della gestione progettuale.                               |
| 25 | Questi sarà inoltre l'unico <b>utente</b> del suo <b>gruppo</b> che <b>potrà scrivere</b> nella <b>cartella del sotto progetto</b> |
| 26 | e dovrà coordinare le operazioni di <b>avanzamento</b> della propria <b>area di competenza</b> .                                   |
| 27 | Ogni utente registrato iscritto dovrà poter interagire attivamente sia con tutti quelli che condividono lo                         |
| 28 | stesso bagaglio di competenze specifiche del proprio team sia con chi necessita di aiuto per una particolare                       |
| 29 | problematica tecnico, pertanto ogni utente potrà definire un proprio elenco di contatti con cui                                    |
| 30 | avere uno scambio di informazioni tramite messaggi privati. Per le problematiche d'interesse generale,                             |
| 31 | ogni utente registrato al sito avrà la possibilità di scrivere o rispondere a post visibili da tutti, ponendo                      |
| 32 | domande tecniche agli altri o fornendo risposte, potrà infatti leggere post che appartengono ad un                                 |
| 33 | determinato soggetto d'interesse, proponendo delle soluzioni in base alla propria esperienza e capacità.                           |
| 34 | Gli utenti non registrati avranno solo l'accesso in lettura ai post e non potranno inserire alcun commento.                        |
| 35 | Tutte le altre tipologie di utente registrato avranno invece accesso in lettura e in scrittura                                     |
| 36 | ai post tecnici già aperti o ne potranno aprire di propri per una valutazione della collettività.                                  |

---

## Risoluzione delle ambiguità

|          |                                                                                                                                          |
|----------|------------------------------------------------------------------------------------------------------------------------------------------|
| Riga 1   | Sostituire <u>Progetti software</u> con <u>Open Source Project</u>                                                                       |
| Riga 1-2 | Sostituire <u>partecipanti</u> con il più tecnico <u>Utenti</u>                                                                          |
| Riga 4   | La <u>data di creazione</u> deve essere specifica nella forma <u>giorno/mese/anno</u>                                                    |
| Riga 4   | Sostituire <u>L'ideatore</u> con <u>l'Utente</u> , in quanto l'ideatore è anche un Utente che partecipa alla realizzazione del progetto. |
| Riga 6   | <u>Persone</u> è generico, viene sostituito con <u>Utenti Registrati</u> .                                                               |
| Riga 9   | <u>Gruppo</u> è impreciso, sostituito con <u>PMC</u> ( project management committee)                                                     |
| Riga 9   | <u>Utenti registrati</u> è generico, sostituito con Project Manager (abbreviato PM)                                                      |
| Riga 12  | <u>Sotto Problemi</u> in realtà s'intende <u>Sotto Progetti</u>                                                                          |
| Riga 12  | <u>Team</u> è generico, viene sostituito con <u>Gruppo di Lavoro</u>                                                                     |
| Riga 13  | <u>Capo</u> è generico, sostituito con <u>Team Manager</u> ( abbreviato TM)                                                              |
| Riga 15  | <u>Età</u> è impreciso, sostituito con <u>data di nascita</u>                                                                            |
| Riga 22  | <u>Gruppo di management</u> , più tecnicamente si intende la <u>PMC</u> (Project Management Committee)                                   |
| Riga 23  | Utente registrato, generico sostituire con Team Manager                                                                                  |
| Riga 25  | <u>Utente</u> è generico, sostituito con <u>Team Manager (TM)</u>                                                                        |
| Riga 25  | <u>Gruppo</u> è generico, si intende <u>Gruppo di Lavoro</u>                                                                             |
| Riga 25  | <u>Cartella del sotto-progetto</u> s'intende più tecnicamente la <u>Directory condivisa di sotto-progetto</u> .                          |
| Riga 26  | <u>Avanzamento</u> è generico, sostituire con <u>percentuale di completamento</u> .                                                      |
| Riga 26  | <u>Area di competenza</u> , s'intende la competenza specifica dell'utente                                                                |

## Nozioni basiche: Glossario dei termini e dei concetti fondamentali

| Concetto                                                          | Descrizione                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Relazionato con                                                                                                                                                                 |
|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Progetto</b>                                                   | Prodotto unico finale, nato da un'idea, derivante dallo sforzo temporaneo e sinergico di più utenti.<br>E' definito tramite specifiche generali e scomposto in sotto progetti di più facile risoluzione.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <ul style="list-style-type: none"> <li>• Utente registrato</li> <li>• PMC</li> <li>• PM</li> <li>• Sotto progetto</li> </ul>                                                    |
| <b>Sotto progetto</b>                                             | Sezione in cui viene suddiviso il progetto per la risoluzione di problematiche più specifiche e di difficoltà ridotta.<br>La scelta dei sotto progetti è effettuata dai PM che vi appartengono.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <ul style="list-style-type: none"> <li>• TM</li> <li>• TE</li> <li>• GdL</li> <li>• PCM</li> <li>• PM</li> </ul>                                                                |
| <b>Competenza</b>                                                 | Particolare abilità che può essere posseduta da un utente o può essere necessaria al determinato sotto progetto.<br>Viene utilizzata per poter definire un'assegnazione utente - sotto progetto. La definizione delle competenze necessarie al progetto viene determinata dai Project Manager secondo una tabella.<br>La tabella comprenderà sia l'elenco delle competenze richieste dai sotto progetti sia l'elenco delle competenze possedute dagli utenti.<br>Se l'utente ha delle particolari competenze che non sono necessarie ad alcun progetto attualmente aperto, questo può dichiarare la propria competenza nell'attesa che un progetto ne necessiti.<br>Esempi di possibili competenze: programmazione C, , programmatore Java, grafico, traduttore . | <ul style="list-style-type: none"> <li>• Utente registrato</li> <li>• Sotto progetto</li> <li>• Progetto</li> </ul>                                                             |
| <b>Utente registrato</b>                                          | Classe generale di persone che si occupa della realizzazione del progetto.<br>Ogni utente registrato può iscriversi ad un progetto per verificarne la struttura e la suddivisione ma la partecipazione ai vari sotto progetti di sviluppo non è comunque obbligatoria.<br>Ogni utente registrato ha la possibilità di creare un elenco di contatti definibile tra tutti gli altri utenti registrati al sito.<br>Tra di essi possiamo distinguere in linea gerarchica con differenti privilegi: <ul style="list-style-type: none"> <li>• Project Manager</li> <li>• Team Manager</li> <li>• Team Element</li> <li>• Utente Normale</li> </ul>                                                                                                                      | <ul style="list-style-type: none"> <li>• Progetto</li> <li>• Sotto progetto</li> <li>• Competenza</li> <li>• PCM</li> <li>• GdL</li> <li>• Post</li> <li>• Messaggio</li> </ul> |
| <b>Project Manager (abbr. PM)</b><br><br>(sottoclasse di Utente ) | Si occupa della buona riuscita del progetto, suddividendolo in differenti settori di competenza o sotto progetti, che verranno poi gestiti dai differenti Gruppi di Lavoro. Definisce inoltre le competenze necessarie alla sviluppo di ogni singolo sotto progetto.<br>Ogni PM può occuparsi di più progetti, il primo PM iscritto coincide anche con la figura dell'ideatore del progetto stesso.                                                                                                                                                                                                                                                                                                                                                               | <ul style="list-style-type: none"> <li>• Sotto progetto</li> <li>• Post</li> <li>• Messaggio</li> </ul>                                                                         |
| <b>Project Management Committee (abbr. PMC)</b>                   | La leadership di progetto è affidata al Project Management Committee (PMC) di cui fanno parte un gruppo di utenti che gestiscono le risorse condivise, determinano la suddivisione, l'aggiunta o l'eliminazione di altri sotto progetti e le competenze necessarie, assegnano inoltre il Team manager dei gruppi di lavoro creati per ogni sottoprogetto.<br><br>La determinazione di una PMC deve essere effettuata prima delle specifiche progettuali ed è composta inizialmente solo dall'ideatore stesso che crea il progetto,                                                                                                                                                                                                                                | <ul style="list-style-type: none"> <li>• Progetto</li> <li>• Sotto progetto</li> <li>• Competenza</li> <li>• Post</li> <li>• Messaggio</li> <li>• PM</li> </ul>                 |

|                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                      |
|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
|                                                                            | <p>il quale poi si occuperà anche dell'inserimento delle specifiche progettuali.</p> <p>Ad esempio il gruppo di Lavoro che si occupa del l'interfaccia grafica avrà bisogno di un grafico disegnatore, un programmatore C, un programmatore Qt ecc. ecc.</p>                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                      |
| <b>Team Manager (abbr. TM)</b><br><i>(sottoclasse di Utente)</i>           | <p>Si occupa dell'avanzamento del settore di competenza assegnato dal Project Manager. E' responsabile degli aspetti tecnici del sotto progetto assegnato.</p> <p>Ha diritto di scrittura sull'archivio dei sorgenti e ha diritto di voto su questioni tecniche per lo sviluppo del sotto-progetto di competenza.</p> <p>Nuovi Team Manager possono aggiungersi, in differenti gruppi di lavoro solo per invito degli appartenenti al PMC e previo consenso degli altri componenti.</p> <p>Un TM è anche un TE del suo team, e partecipa attivamente allo sviluppo del progetto.</p>                                                                                                    | <ul style="list-style-type: none"> <li>• Sotto progetto</li> <li>• Post</li> <li>• Messaggio</li> <li>• PM</li> <li>• PMC</li> </ul> |
| <b>Team Element (abbr. TE)</b><br><i>(Sottoclasse di Utente)</i>           | <p>Membro attivo del progetto che partecipa all'avanzamento con il suo apporto, devoluto in base alle proprie competenze e disponibilità.</p> <p>Ogni TE è assegnato ad un sotto progetto ed ha la possibilità di partecipare a sotto progetti differenti ma che saranno raggiungibili solo se attinenti alle proprie competenze specificate in fase di registrazione iniziale.</p> <p>Non sarà ad esempio possibile partecipare ad un sottoprogetto per cui non si hanno le competenze richieste.</p> <p>Ha un accesso in sola lettura al disco condiviso del progetto. Il TM è un particolare tipo di TE.</p>                                                                         | <ul style="list-style-type: none"> <li>• Sotto progetto</li> <li>• Post</li> <li>• Messaggio</li> <li>• Team Manager</li> </ul>      |
| <b>Utente Normale (abbr. UN)</b><br><i>(Sottoclasse di Utente)</i>         | <p>Utente che non partecipa direttamente al progetto in quanto non possiede nessuna delle competenze richieste e non è inserito in nessun team di lavoro per un sotto progetto, può esprimere pareri e suggerimenti di modifica sullo stato del progetto con commenti o post tecnici.</p>                                                                                                                                                                                                                                                                                                                                                                                               | <ul style="list-style-type: none"> <li>• Post</li> <li>• Messaggio</li> </ul>                                                        |
| <b>Utente non registrato (abbr. UNR)</b><br><i>(Sottoclasse di Utente)</i> | <p>Utente esterno e anonimo che non partecipa al progetto e ha solo capacità di lettura sui post tecnici che vengono aperti dagli utenti registrati.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <ul style="list-style-type: none"> <li>• Post ( lettura )</li> </ul>                                                                 |
| <b>Incarico</b>                                                            | <p>Ad ogni utente viene attribuito un Incarico di appartenenza all'interno del progetto dipendente dal livello gerarchico in cui questo viene inquadrato. Ogni utente può partecipare all'interno del progetto con differenti ruoli e su più sotto gruppi di lavoro. Ad esempio un utente che abbia più competenze può essere impiegato nel sottogruppo di Design dell'interfaccia grafica e di creazione del DB per la gestione degli utenti, ed essere utilizzato contemporaneamente come PM per l'intero progetto.</p> <p>Ogni funzione svolta deve essere quindi documentata, come pure tutti gli accessi alle cartelle del disco di cui fa parte a seconda delle sue funzioni.</p> | <ul style="list-style-type: none"> <li>• Progetto</li> <li>• Sotto progetto</li> <li>• Utente Registrato</li> </ul>                  |
| <b>Post</b>                                                                | <p>Discussione aperta da qualunque Utente registrato.</p> <p>Il post può essere iniziato sotto forma di domanda per l'apertura di una discussione tecnica, oppure può essere</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <ul style="list-style-type: none"> <li>• Utente Registrato</li> <li>• Utente non registrato</li> </ul>                               |

---

|                 |                                                                                                                                                                                                                                                                                                                                                  |                                                                     |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
|                 | <p>una risposta ad richiesta tecnica attinente il progetto o una sua sezione (un sotto progetto).</p> <p>Anche gli utenti non registrati sono in grado di visualizzare i Post ma non potranno rispondere. Il dato potrebbe essere impiegato per definire una statistica di interessi per quella fascia di utenti che non è iscritta al sito.</p> |                                                                     |
| <b>Messaggi</b> | <p>Sono inviati da utente ad utente e visibili solo da chi invia e da chi riceve.</p> <p>Gli utenti registrati possono inviare o ricevere i messaggi ad altri utenti registrati.</p> <p>Gli utenti non registrati non hanno accesso a questa tipologia di servizio.</p>                                                                          | <ul style="list-style-type: none"><li>• Utente registrato</li></ul> |



---

## Operazioni di AGGIORNAMENTO soddisfatte dal DB

|    |                                                  |
|----|--------------------------------------------------|
| 1  | Inserimento Utente                               |
| 2  | Inserimento Progetto                             |
| 3  | Inserimento Sotto Progetto                       |
| 4  | Inserire Post Domanda                            |
| 5  | Inserire Post Risposta                           |
| 6  | Eliminazione di un Utente                        |
| 7  | Eliminazione di un Progetto                      |
| 8  | Inserire un messaggio da un utente per un utente |
| 9  | Eliminazione di un Sotto Progetto                |
| 10 | Inserire Incarico                                |
| 11 | Inserire Competenza                              |

## Operazioni di INTERROGAZIONE al DB

|    |                                                                                                                                                                                                                                                                     |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | Visualizzare l'elenco degli utenti iscritti al sito                                                                                                                                                                                                                 |
| 2  | Visualizzare le competenze che ciascun utente ha dichiarato di possedere.                                                                                                                                                                                           |
| 3  | Trovare le competenze richieste per i sotto progetti dal progetto.                                                                                                                                                                                                  |
| 4  | Trovare quali utenti possiedono le competenze richieste dal progetto specificato.<br>La tabella deve contenere il nome_utente , la competenza posseduta dall'utente richiesta da almeno uno dei sotto progetti definiti per il progetto con id_project specificato. |
| 5  | Trovare i sotto progetti in un qualunque progetto che richiedono le competenze specifica posseduta dall'utente specificato.                                                                                                                                         |
| 6  | Visualizzare i titoli dei Post scritti dall'utente specificato. Deve essere visualizzato il titolo del post, la data in cui è stato scritto ed il nome dello username che lo ha composto.                                                                           |
| 7  | Trovare tutti i Messaggi aventi il destinatario specificato. Deve essere visualizzato il nome utente del ricevente la data d'invio ed il titolo del messaggio.                                                                                                      |
| 8  | Visualizzare nomi e cognomi dei contatti posseduti dall'utente. Deve comparire su 4 colonne l'id dell'utente che possiede il contatto e i contatti posseduti ( nome , cognome e username ).                                                                         |
| 9  | Trovare quanti utenti partecipano ad un progetto                                                                                                                                                                                                                    |
| 10 | Determinare quali utenti partecipano a più di un unico sotto-progetto ed indicare a quanti sotto-progetti.                                                                                                                                                          |
| 11 | Determinare l'elenco dei Project Manager per un determinato progetto.<br>Devono essere visualizzati nomi e cognomi e il nome del progetto di appartenenza.                                                                                                          |
| 12 | Determinare l'elenco dei sotto progetti che fanno parte di un progetto.<br>La query deve associare al nome di un progetto le specifiche del sotto progetto.                                                                                                         |
| 13 | Determinare l'avanzamento totale dei progetti come media degli avanzamenti dei sotto progetti, deve essere visualizzato il nome del progetto ed il suo stato di avanzamento.                                                                                        |
| 14 | Determinare l'elenco gli utenti che lavorano ad un determinato progetto.                                                                                                                                                                                            |
| 15 | Determinare gli utenti che NON hanno delle competenze specifiche per il sotto-progetto                                                                                                                                                                              |
| 16 | Determinare l'elenco degli utenti che hanno delle competenze comuni.                                                                                                                                                                                                |
| 17 | Determinare i titoli dei post il cui messaggio contenga una determinata stringa.                                                                                                                                                                                    |

---

## Progettazione concettuale

Vengono ora elencate le entità<sup>1</sup> definite nel progetto e le relazioni che intercorrono tra queste.

Ogni relazioni tra entità ed entità può avere differenti vincoli di cardinalità.

La partecipazione dell'entità è **opzionale** se la cardinalità minima è uguale a 0, mentre la partecipazione dell'entità è **obbligatoria** se la cardinalità minima uguale a 1.

Vengono così definite le possibili *relazioni di cardinalità*:

(0,N) indica che l'entità può partecipare alla relazione con tutti o nessun elemento in modo opzionale, ovvero con un numero qualunque di elementi.

(1,N) indica che l'entità può partecipare alla relazione con tutti gli elementi, ma almeno 1 elemento deve essere presente nella relazione.

(1,1) indica che l'entità partecipa alla relazione con 1 solo ed unico elemento

(0,1) indica che la partecipazione di un solo elemento è opzionale.

Per ogni entità viene inoltre stabilita una chiave primaria, ossia un particolare attributo in grado di identificare ogni elemento dell'entità in modo univoco nell'insieme.

**Analizzando il progetto sono state rilevate le seguenti entità:**

1. **Entità progetto:** l'entità Progetto viene suddivisa in differenti e più semplici sotto progetti di sviluppo. L'entità serve a definire delle specifiche progettuali in modo che possa essere suddiviso il lavoro in differenti sotto-progetti.

Il progetto sarà caratterizzato da attributi quali:

- **Id\_progetto:** di tipo incrementale, serve ad identificare univocamente ogni nuovo progetto inserito ed è la chiave primaria dell'entità progetto.
- **Nome\_progetto:** indica il nome assegnato al progetto dall'ideatore
- **id\_ideatore:** id dell'utente registrato che ha definito il progetto e le sue specifiche. L'ideatore diventa immediatamente anche il primo PM del proprio progetto, se non fosse specificato il suo id tra le informazioni del progetto si perderebbe traccia dell'informazione.
- **Data\_creazione:** data d'inserimento del nuovo progetto
- **Specifiche\_tecniche:** in formato testuale, le specifiche tecniche, chiare ed essenziali che guidano alla realizzazione degli obiettivi generali.

**Relazioni dell'entità progetto:**

- **Ogni progetto può possedere (0,N) sotto progetti figli :** un progetto può possedere dei sotto progetti di sviluppo a cui verranno assegnati gli utenti registrati in base alle competenze progettuali richieste dal progetto stesso e in base alle capacità che gli utenti hanno dichiarato di avere al momento della registrazione al sito.
- Ogni progetto è sempre gestito da almeno un PM, tuttavia un numero qualunque di PM possono far parte del progetto. **Il fondatore del progetto assume immediatamente il ruolo di PM.**

---

<sup>1</sup> Insieme di oggetti concreti o astratti che condividono le stesse proprietà e la propria autonomia

---

All'inserimento di un nuovo progetto vengono effettuate le seguenti azioni:

- inserimento dei dati del progetto (nome progetto , specifiche ...)
- inserimento dell'ideatore nell'elenco dei PM del progetto ( l'utente ideatore come specificato è anche il primo PM ).
- determinazione dei sotto-progetti pertinenti (questa fase può essere posticipata in attesa di successive determinazioni da parte del PM).
- determinazione delle competenze richieste da ogni sotto-progetto (questa fase che riguarda lo sviluppo sotto progettuale può essere posticipata in attesa di successive determinazioni).

2. **Entità sotto progetto:** l'entità Sotto Progetto è la naturale suddivisione in più aree di sviluppo in cui le specifiche iniziali del progetto è stato scomposto.

La scomposizione del progetto in parti più semplici è dinamica e può variare nel tempo, in quanto è possibile aggiungere successive aree di lavoro per ripartire le problematiche che sono risultate eccessivamente complesse. ( Partizionamento Top-Down )

Ad esempio l'ideatore del progetto per un sito Web potrà definire il sotto progetto che si occupa del design delle pagine, il sottoprogetto che si occupa della gestione delle pagine HTML un altro sottoprogetto che definisce il DB per la gestione di utenti al sito. Ogni sotto progetto deve essere corredato di chiare specifiche che ne permettono l'attuazione.

**L'entità sotto progetto sarà caratterizzata dai seguenti attributi:**

- **Id\_sotto progetto:** di tipo incrementale serve per attribuire un indice unico al progetto inserito ed è la chiave primaria dell'entità
- **Specifiche\_spj:** indica in formato testuale le specifiche tecniche che devono essere seguite per la realizzazione del sotto progetto.
- **Stato\_avanzamento\_spj:** indica in formato percentuale lo stato di avanzamento del sotto progetto constatato dal TM (Team Manager).
- **data\_ultima\_modifica:** indica la data in cui il TM, unico con l'accesso in scrittura alla propria directory, ha inserito gli aggiornamenti.
- **Directory\_lavoro\_sp:** indica la sotto directory in cui vengono salvati i files sorgenti per il sotto progetto. I permessi di lettura e scrittura si estendono ricorsivamente a tutte le directory annidate e sono attribuiti all'utente in base alle necessità derivanti dall'incarico. Ad esempio se l'utente appartiene al gruppo del TM ed è associato ad un determinato progetto allora avrà l'accesso in lettura e scrittura solo alla directory di lavoro del proprio Team.

**Relazioni dell'entità sotto progetto :**

- Qualunque utente registrato può occuparsi di ogni sotto progetto esistente, se possiede le competenze. Almeno 1 degli utenti che si occupano di un sotto progetto deve essere TM del suo team. Il primo utente inserito in un sotto progetto diventa automaticamente anche il TM,
- Ogni sotto progetto ha un certo numero di competenze richieste per il suo sviluppo, ad ogni competenza necessaria viene assegnato uno o più utenti che la possiedano e che siano in grado di far avanzare il progetto.
- Ogni sotto progetto può avere (0,N) persone che si occupano del suo sviluppo, tra cui solo una è anche il TM. Ogni sotto progetto non necessariamente possiede un utente che vi partecipa, quindi dopo aver determinato le specifiche in una fase posteriore vengono designati i partecipanti.

Il sotto progetto può essere in attesa di utenti che soddisfino le richieste per le competenze previste definite dai PM e pertanto la quantità dei partecipanti può essere NULLA.

- 
3. **Entità utente:** viene indicato nella forma più generale ogni interessato al sito, sia che questo partecipi attivamente sia che invece sia solo uno spettatore di passaggio.

Tale entità generalizzata, possiede due principali forme di specializzazione quali : **utente\_non\_registrato ed utente\_registrato**. La generalizzazione ha un gerarchia di tipo totale poiché per ogni istanza della generalizzazione esiste almeno un istanza della specializzazione.

- **Utente non registrato:** è un utente anonimo e di passaggio a cui viene assegnato un id e un nick name del tutto casuale, quindi non necessita d'incorporare dati personali di qualunque tipo.

**L'utente non registrato sarà caratterizzato dai seguenti attributi:**

- **id\_utente\_nr** : indica un id numerico consecutivo assegnato agli utenti che visitano il sito ma che non hanno intenzione di registrarsi. E' una chiave primaria.
- **nome\_utente:** nome utente assegnato per un'identificazione fittizia dell'user anonimo loggato ( la determinazione è del tutto casuale ).

**Relazioni dell'utente non registrato:**

- un numero qualunque di utenti non registrato può leggere un numero qualunque di post presenti sul forum. La relazione è di tipo N:N. L'utente non registrato **non può** espletare funzioni differenti da questa.
- **Utente registrato:**

Ogni utente registrato oltre a poter partecipare attivamente al progetto ha la possibilità di inviare messaggi ad altri utenti e può aprire e rispondere a post tecnici,

**Ogni utente registrato è caratterizzato dai seguenti attributi:**

- **attributi riguardanti i dati personali dell'utente:** nome, cognome, sesso, data\_di\_nascita, professione, cittadinanza, lingua. Sono dati inseriti in fase di registrazione.
- **Attributi riguardanti la gestione dell'utente all'interno del social network quali:**
- **id\_utente:** indicatore numerico consecutivo assegnato permanentemente ad ogni utente registrato al sito. E' una chiave primaria, in quanto nome e cognome utente potrebbero non essere un buon identificativo per una chiave primaria in caso di omonimia.
- **nome\_utente:** nome scelto in fase di registrazione dall'utente, è usato anche per l'accesso al sito
- **password:** la password scelta dall'utente per l'accesso al sito

**Relazioni dell'utente registrato:**

- Ogni utente può avere differenti incarichi di progetto, può ad esempio partecipare a più sotto\_progetti all'interno dello stesso progetto , oppure può svolgere una funzione di Project Manager e di Team Manager se è interessato direttamente allo sviluppo e non solo alla gestione progettuale.  
Un utente potrebbe non avere incarichi all'interno del progetto, è il caso del Normal User, che seppur iscritto al sito non assume incarichi specifici perché al momento ad esempio le sue competenze potrebbero non essere richieste. La relazione è di tipo N:N.
- Qualsiasi utente registrato può avere contatti con altri utenti registrati. La relazione è N:N viene pertanto definita un'ulteriore tabella che associ ad ogni utente un elenco dei suoi contatti.
- Qualunque utente registrato può essere iscritto ad un determinato progetto, tuttavia il progetto deve contenere almeno un utente registrato, ossia il suo ideatore.

- Qualunque utente registrato può leggere qualsiasi un POST di discussione.
- Qualunque utente registrato può rispondere a qualsiasi POST aperto.
- Qualunque utente registrato può mandare qualsiasi messaggio a qualunque altro utente registrato
- Qualunque utente registrato può ricevere qualsiasi messaggio privato da qualsiasi altro utente registrato.

L'utente registrato è una **generalizzazione** derivante dalle seguenti sottoclassi:

- **PM (Project Manager ):** figura che si occupa della gestione del progetto nonché della suddivisione in sotto problemi. E' inoltre interessato alla definizione delle competenze necessarie per il sviluppo.

**Relazioni del PM trovate:**

- Ogni PM può gestire qualunque Progetto : c'è una relazione N:N che lega PM e Progetti, in quanto ogni PM può occuparsi della gestione di più progetti, mentre ogni progetto può avere differenti PM. Come già definito in precedenza, ogni progetto deve avere almeno un utente PM, ossia il suo ideatore.
- **TC ( Team Component ) :** lavora direttamente per lo sviluppo del progetto. Ogni team component è assegnato ad un sotto progetto. Il team component è a sua volta la generalizzazione derivante dalle sotto classi: Tema manager e Team Element.
- **TM ( Team Manager ):** responsabile del Team per la risoluzione degli aspetti tecnici del sotto progetto assegnato è l'unico inoltre che ha l'accesso in scrittura nella directory del progetto.

Relazioni trovate:

- Un solo TM può gestire il sotto progetto, tuttavia differenti sotto progetti possono avere lo stesso TM.
- Un solo TM gestisce tutti i TE che fanno parte sotto progetto.
- **TE (Team Element ):** programmatore, progettista tecnico, o qualsiasi utente\_registrato che abbia le competenze richieste e si occupa del sotto progetto. Non ha accessi in scrittura ma **solo in lettura** alla directory del sotto progetto, quindi passa il suo prodotto al TM che è responsabile di copiare il codice sorgente nella directory di lavoro.

Relazioni trovate:

- Ogni TE è interessato allo sviluppo dello stesso progetto.
  - Ogni TE è gestito da un solo TM per la fase di sviluppo del sotto\_progetto assegnato
4. **Entità post:** serve per porre delle domande alla comunità e per fare in modo che ci sia una partecipazione da parte di tutti gli utenti tramite una condivisione delle conoscenze.

**Ogni post è caratterizzato dai seguenti attributi:**

- **id\_post:** chiave primaria numerica, identifica in modo univoco il post
- **titolo\_post :** il titolo assegnato all'utente
- **tes to\_post:** il testo di discussione

**Relazioni dell'entità post:**

- ogni post può essere letto da un utente qualunque sia registrato che non registrato, relazione N:N.
- ogni post deve essere scritto da 1 solo utente ( a cui appartiene ).
- ogni post può essere commentato da qualunque altro utente \_registrato, relazione N:N. Ossia qualunque utente registrato può fornire una risposta al post.

5. **Entità messaggio :** serve a mantenere in contatto differenti utenti. Il sito ha la possibilità di tenere traccia dei messaggi privati scritti da/per per ogni altri utente registrato.

**Ogni messaggio è caratterizzato dai seguenti attributi:**

- **id\_messaggio:** indica la chiave primaria che identifica univocamente il messaggio
- **titolo\_messaggio:** indica il titolo del messaggio da inviare
- **corpo\_messaggio:** è il testo del messaggio

**Relazioni dell'entità messaggio:**

- ogni messaggio può essere scritto da qualunque utente ma deve avere un solo ed unico creatore, relazione 1:N.
- ogni messaggio può essere ricevuto da qualunque utente che lo scrive ed ogni utente può aver ricevuto differenti messaggi da differenti utenti. Relazione N:N.

6. **Entità Incarico:** per ogni utente viene definito uno specifico incarico che lo differenzia dagli altri utenti fornendogli diversi accessi al progetto e differenti funzioni amministrative all'interno dello stesso. La suddivisione gerarchica e lo sviluppo delle generalizzazioni già affrontato viene risolto con l'aiuto di tale entità.

Ogni incarico specifico viene assegnato al progetto o al sotto progetto, saranno poi gli utenti specifici a ricoprire i differenti incarichi in base alle proprie competenze. L'utente che avrà un incarico assegnato sarà definito dall'elenco di incaricati.

**Ogni incarico è caratterizzato dai seguenti attributi:**

- **id\_incarico:** chiave primaria, identificatore numerico, serve a differenziare ogni tipologia di incarico.
- **nome\_incarico:** tipo ENUM, indica la tipologia di incarico che si ha all'interno del progetto. Le tipologie sono definite e statiche e possono assumere solo determinati valori quali: PM, TM, TE, NU.
- **tipo\_accesso:** di tipo ENUM, indica se l'utente per quel determinato incarico ha l'accesso in lettura alla directory definita dall'entità sotto progetto, oppure no. Può assumere solo valori definiti quali:  
 r → accesso sola lettura assegnato solo ai team element  
 rw → accesso in lettura e scrittura assegnato ai Project Manager e ai Team Manager

**Relazioni dell'entità incarico:**

- Qualunque incarico può appartenere allo stesso utente, la relazione è quindi N:N molti a molti. Infatti un utente può essere sia PM di un progetto e anche TM di uno o più sotto progetti. Inoltre ogni utente registrato può avere la stessa tipologia di incarico, sarà poi il progetto/sotto progetto a cui viene assegnato l'utente a definire le specifiche e i parametri di lavoro per quell'utente (come la directory di lavoro).

A titolo di esempio possiamo supporre che l'utente user01 a cui potranno essere assegnati differenti incarichi come TE, 1 incarico come PM e 1 incarico come TM, che possono essere impiegati, in funzione delle sue competenze all'interno di differenti progetti o sotto\_progetti.

- Qualunque incarico può essere assegnato ad un progetto oppure ad un sotto progetto. La relazione risulta pertanto essere N:N molti a molti in quanto ogni progetto o sotto progetto sfrutta 1 o più incarichi per poter essere portato a compimento.

- 
7. **Entità competenza :** tiene traccia delle capacità necessarie allo sviluppo del progetto. Le competenze richieste verranno inizialmente selezionate dall'ideatore del progetto e poi dalla commissione di progettisti. Una volta definita la tabella delle competenze i nuovi iscritti potranno correlarsi a tale tabella con le proprie competenze per poter partecipare al progetto.

Le competenze possono essere inserite in elenco anche da qualunque utente registrato che ne possieda di nuove per poi restare a disposizione di ogni sotto progetto che vorrà farne uso.

Ad esempio lo sviluppo di un progetto per un sito web può aver bisogno di un utente che si occupa degli aspetti grafici, di diversi programmatori Java, esperti di HTML, PHP ed un conoscitore di MySQL, ogni progetto a disposizione può avere necessità oppure no delle competenze disponibili.

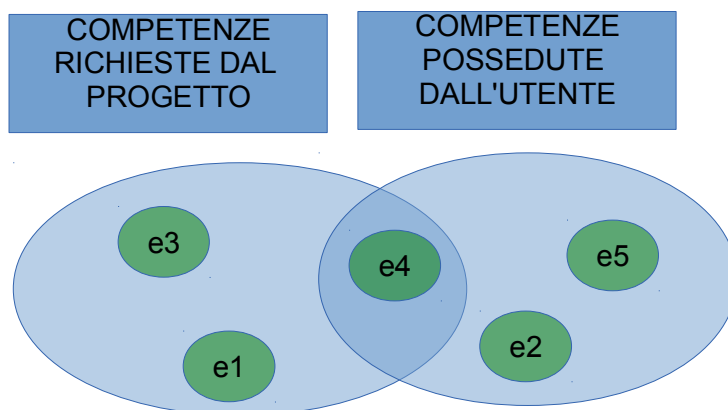
L'ideatore dovrà quindi inizialmente stilare un elenco per tali competenze, per ogni utente neo iscritto sarà poi possibile verificare se possiede una delle competenze specifiche che lo rendono in grado di partecipare attivamente allo sviluppo.

**Ogni competenza è caratterizzato dai seguenti attributi:**

- id\_competenza: chiave primaria numerica assegnata alla competenza, serve ad identificarla in modo univoco.
- nome\_competenza: di tipo alfanumerico, indica il nome assegnato alla competenza.
- data\_inserimento: indica la data d'inserimento della competenza.

**Relazioni dell'entità competenza:**

- Ogni competenza può essere posseduta da qualunque utente registrato al sito. Relazione N:N molti a molti , implica la definizione di una terza tabella che associ ad ogni utente le competenze che possiede.
- Ogni competenza può essere necessaria allo sviluppo di un sotto progetto. Relazione N:N molti a molti, implica la definizione di una terza tabella che associ ad ogni sotto progetto le competenze necessarie.










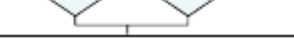

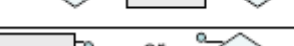











La figura mostra come SOLO le competenze che sono possedute dagli utenti E che sono anche richieste dal progetto possano essere selezionate per un apporto effettivo allo sviluppo progettuale..



## Scelta della strategia di progetto

la strategia adottata è di tipo **top down**, lo sviluppo parte da linee generali, senza scendere subito nei dettagli, dopo la creazione di uno schema iniziale che è una rappresentazione molto astratta dell'idea, viene applicato un processo di raffinamento tramite l'applicazione di primitive che permettono una definizione sempre più dettagliata del prodotto finale.

Queste le principali primitive di Top\_down<sup>2</sup> applicate al procedimento:

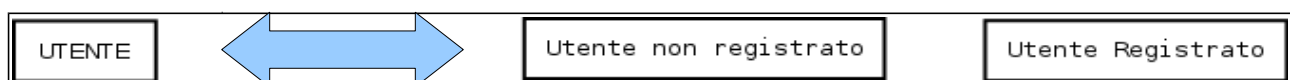
| primitive                                               | input scheme                                                                                                                                                               | output scheme                                                                                                                                                                  |
|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T <sub>1</sub> : entity → related entities              |                                                                                           |                                                                                             |
| T <sub>2</sub> : entity → generalization                |                                                                                           |                                                                                             |
| T <sub>3</sub> : entity → non-related entities          |                                                                                           |                                                                                             |
| T <sub>4</sub> : association → parallel associations    |                                                                                          |                                                                                            |
| T <sub>5</sub> : association → entity with associations |                                                                                         |                                                                                           |
| T <sub>6</sub> : attribute development                  |  or  |  or   |
| T <sub>7</sub> : composite attribute development        |  or  |  or   |
| T <sub>8</sub> : attribute refinement                   |                                                                                         |  or  |

*Illustrazione : Primitive di strategia Top\_Down*

Raccolte le informazioni necessarie per la costituzione del DB queste ora dovranno essere formalizzate tramite un diagramma E-R. Vediamo lo sviluppo secondo la strategia scelta.

**UTENTE:** all'entità generale viene applicata una trasformazione T3 in modo da scorporare gli utenti registrati dagli utenti che invece non si sono registrati al sito. La figura dell'utente non registrato è una figura fittizia dal punto di vista dello sviluppo di questo DB ed in caso reale avrebbe un ID creato momentaneamente dalla macchina a cui sarebbe possibile associare un post per la lettura.

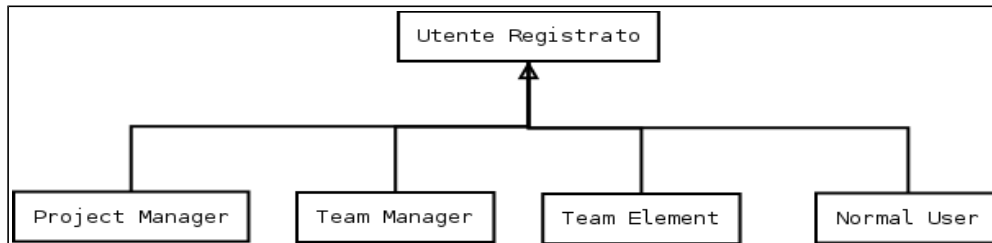
Effettuando una scelta progettuale possiamo trascurare la generalizzazione dell'entità UTENTE derivante da utente non registrato ed utente registrato in modo che siano trattate come entità separate, questo permetterà di avere un risparmio in termini di operazioni sul DB.



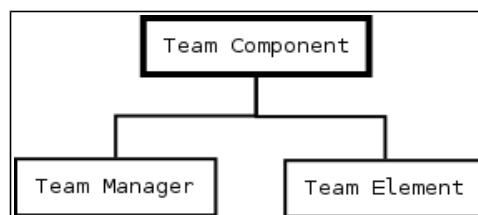


---

All'utente registrato viene poi applicata una trasformazione di generalizzazione di tipo T2 in quanto l'utente registrato può essere: PM, TM, TE, NU.



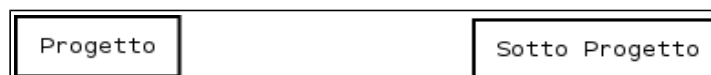
Nel processo di raffinamento possiamo anche verificare che un Team Element e un Team Manager sono dal punto di vista del DB quasi identici se non per l'unica differenza che il TM ha un accesso in scrittura nella directory di lavoro del proprio gruppo. Possiamo gerarchicamente accorpate generalizzando TM e TE sotto un'unica sottoclasse TC ( Team Component ) differenziandoli solo per i diritti di accesso in lettura e in scrittura, caratteristiche che verranno definite grazie all'assegnazione dell'incarico.



Le entità per post e messaggi:



Le entità progetto e sotto progetto:



Le entità incarico e competenze

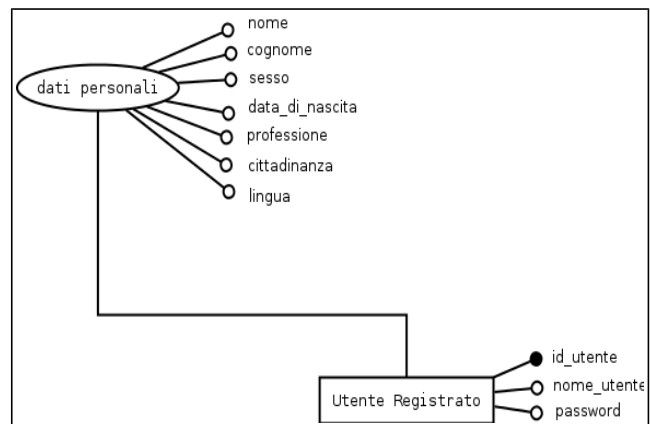


---

Ora applico le primitive T6 e T7 di sviluppo e raffinazione degli attributi alle entità generate:

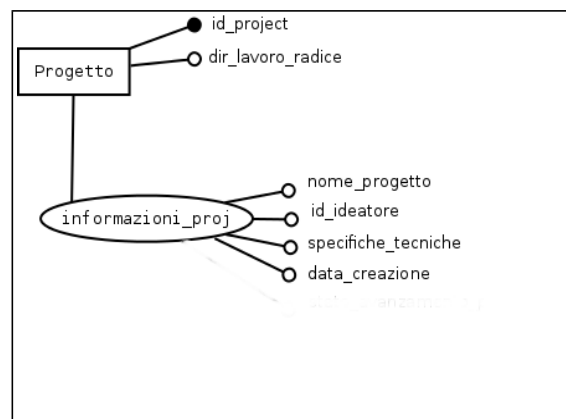
**Entità utente registrato:**

le informazioni sui dati personali  
sono attributi compositi



**Entità progetto:**

le informazioni sul  
progetto sono attributi compositi.



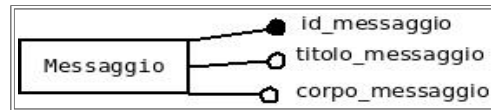
**Entità sotto progetto:**

le informazioni sul  
sotto\_progetto sono attributi compositi.



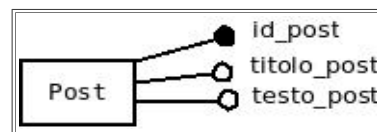
---

Entità Messaggio:



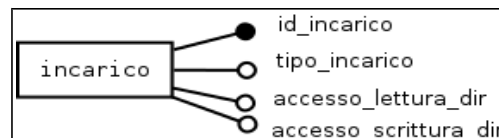
---

Entità post:



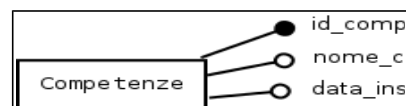
---

Entità incarico:



---

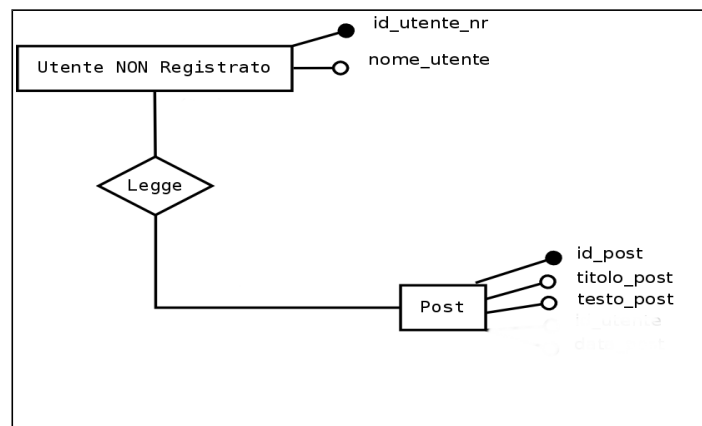
Entità competenze:



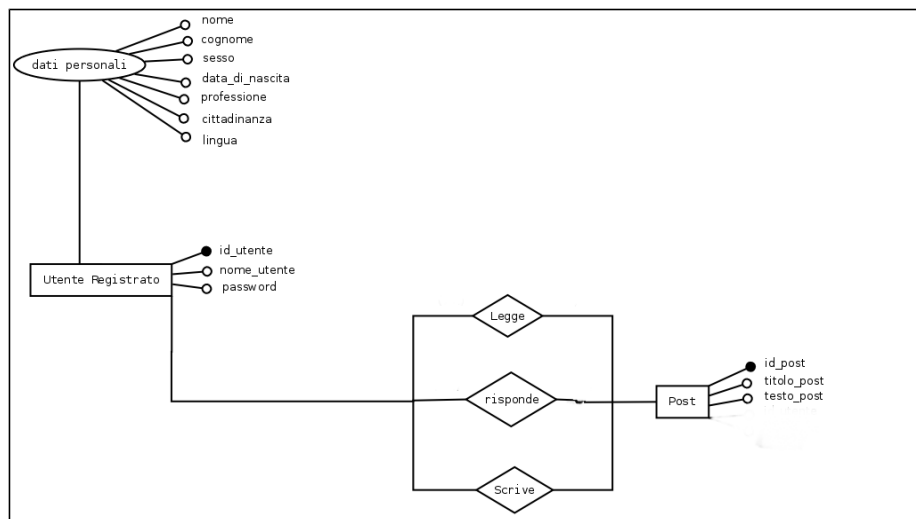
Ora si può passare all'applicazione delle trasformate T1 e T4 in modo da poter sviluppare le entità

correlate e le correlazioni parallele. La stesura delle relazioni viene diversificata in funzione dei differenti usi del DB , ogni **vista verrà poi successivamente integrata** per l'ottenimento di un DB completo e coerente.

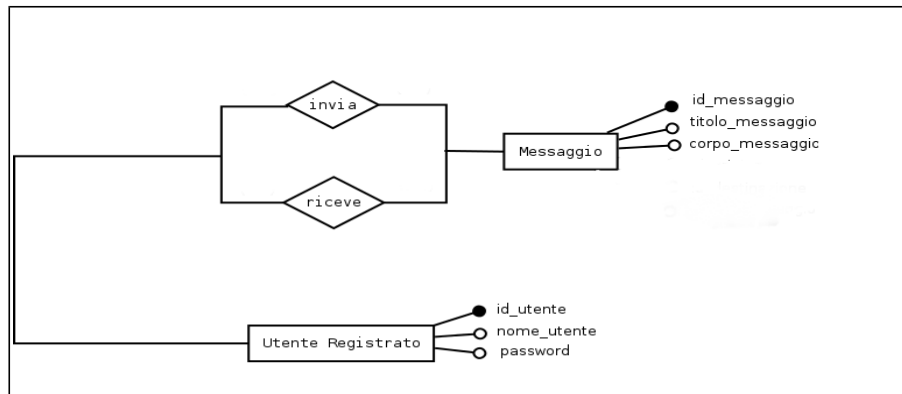
**Relazione tra utente NON registrato e post:** è una relazione N:N molti a molti in quanto ogni utente non registrato può leggere più post ed un post può avere più lettori.



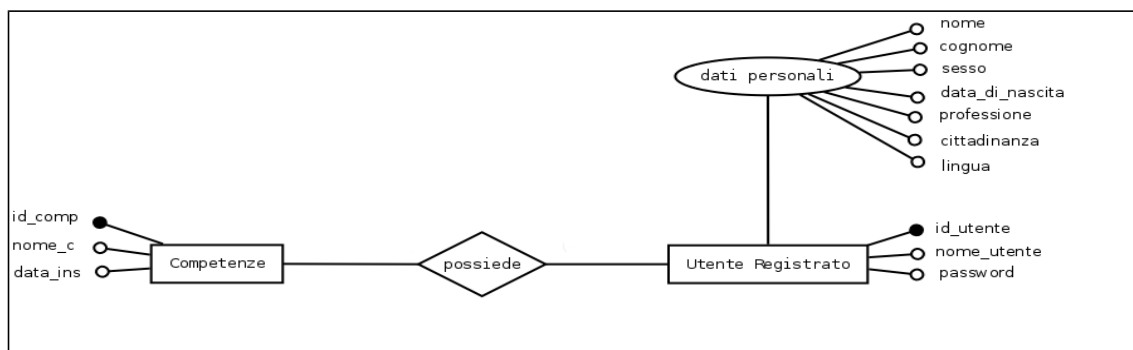
**Relazione tra utente registrato e post:** la relazione è N:N molti a molti per quanto riguarda la lettura e la risposta ad un post in quanto ogni utente registrato può leggere e rispondere a differenti post e lo stesso post può a sua volta essere letto e replicato da utenti differenti. La relazione che intercorre per la scrittura del post è invece di tipo 1:N in quanto ogni utente registrato si suppone possa scrivere differenti post tuttavia lo stesso post può essere originato da un solo utente registrato.



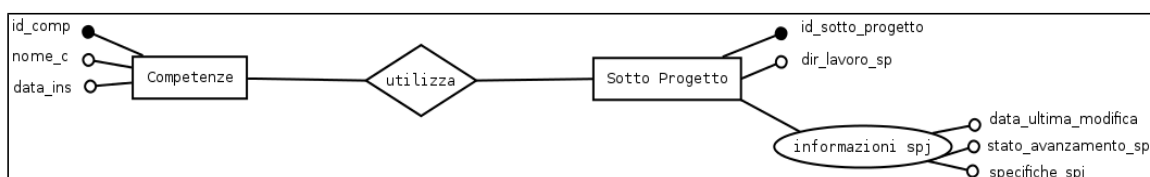
**Relazione tra utente registrato e messaggio:** ogni utente registrato può ricevere ed inviare messaggi ad altri utenti registrati. Il tipo di relazione che lega utente registrato e messaggio è 1:N per la scrittura e N:N per la ricezione. Ogni utente può ricevere differenti messaggi e lo stesso messaggio può essere destinato a differenti utenti, tuttavia ogni messaggio può avere un solo utente che lo origina.



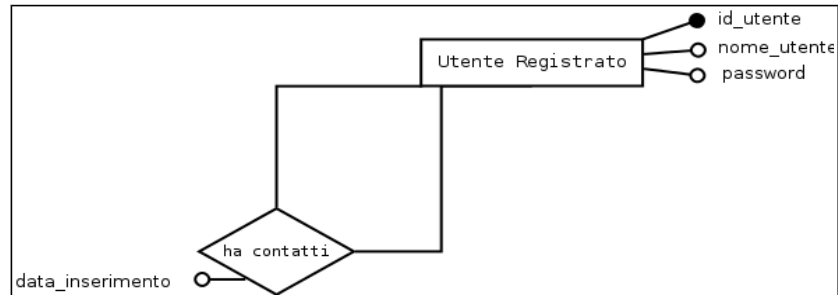
**Relazione tra utente registrato e competenze:** relazione di tipo N:N, infatti ogni utente può possedere differenti competenze e ogni competenza esistente può essere comune a più utenti. Ad esempio molti utenti possono essere programmatori C, e la competenza programmazione C può essere comune a più utenti.



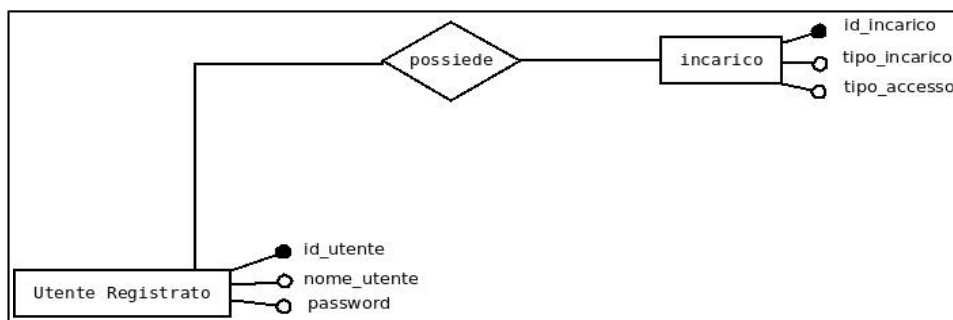
**Relazione tra Competenza e Sotto Progetto:** la relazione è di tipo N:N in quanto ogni sotto progetto utilizza una o più competenze e ogni particolare competenza può essere sfruttata da differenti sotto\_progetti.



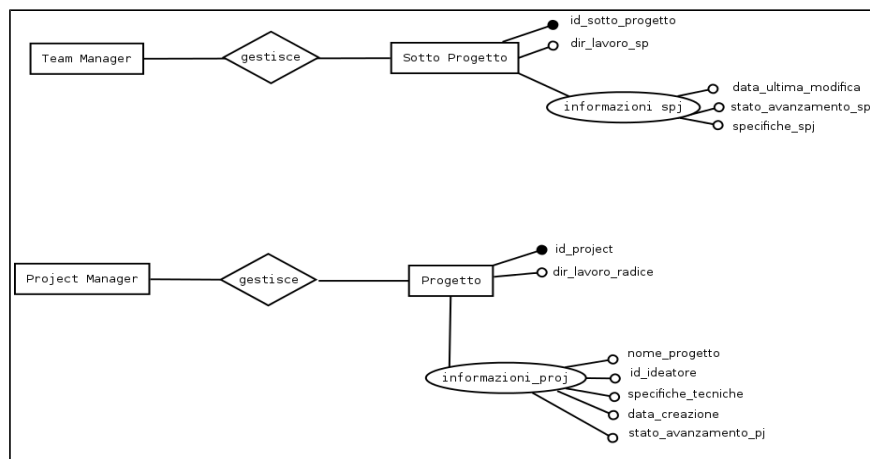
**Relazione di utente registrato con utente registrato:** relazione ad anello, ogni utente può avere dei contatti nella propria rubrica. La relazione è di tipo N:N, molti a molti, infatti ogni utente può avere differenti contatti ed appartenere a sua volta alla rubrica di differenti utenti.



**Relazione di utente registrato con incarico:** relazione di tipo 1:N infatti ogni utente registrato può avere differenti incarichi che poi verranno assegnati a più progetti / sotto progetti tuttavia solo una tipologia di incarico assegnato appartiene all'utente.

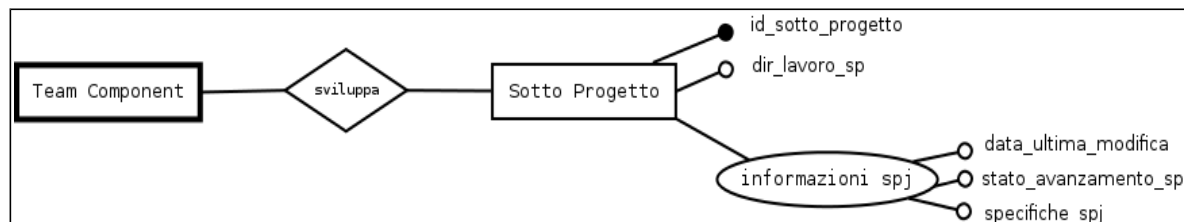


**Relazioni di gestione tra Team Manager e Sotto Progetto e tra Project Manager e Progetto:** le relazioni sono differenziabili in 1:N per quanto riguarda TM e sotto Progetto, infatti un sotto progetto può essere gestito da un solo TM che a sua volta può essere responsabile per differenti sotto progetti anche appartenenti a più progetti. La relazione tra PM e Progetto è invece di tipo N:N in quanto ogni PM può gestire più progetti e lo stesso progetto può essere gestito da differenti PM.



**Relazioni di**

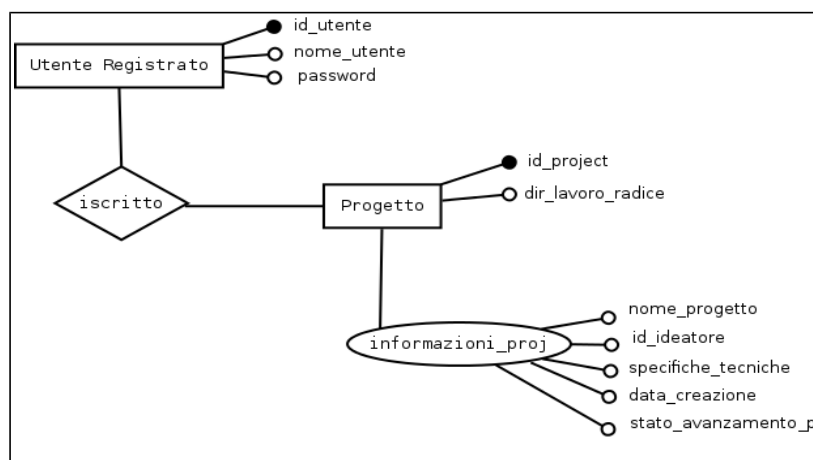
### **sviluppo tra Team Component e Sotto Progetto:** relazione di tipo N:N



Come possiamo notare dall'uso delle relazioni adottato nel diagramma, il TM si occupa della gestione del sotto progetto ma anche del suo sviluppo. Il Team Component è inoltre una generalizzazione del Team Manager che si può occupare di sviluppo e di gestione del sotto progetto e del Team Element che è invece unicamente adibito alla programmazione e allo sviluppo del sotto progetto.

Il Project Manager, ma solo in questa ed unica funzione, non si occupa mai dello sviluppo diretto del progetto ma piuttosto della sua gestione organizzando le azioni necessarie alla buona riuscita. Tuttavia l'utente può assumere differenti incarichi, infatti un utente registrato è una generalizzazione di un PM e di un TC, quindi qualora ad esempio il PM risulti essere anche buon programmatore con adeguate competenze di sviluppo, questi potrà assumere un altro incarico ed entrare a far parte di un sotto progetto per lo sviluppo diretto del software.

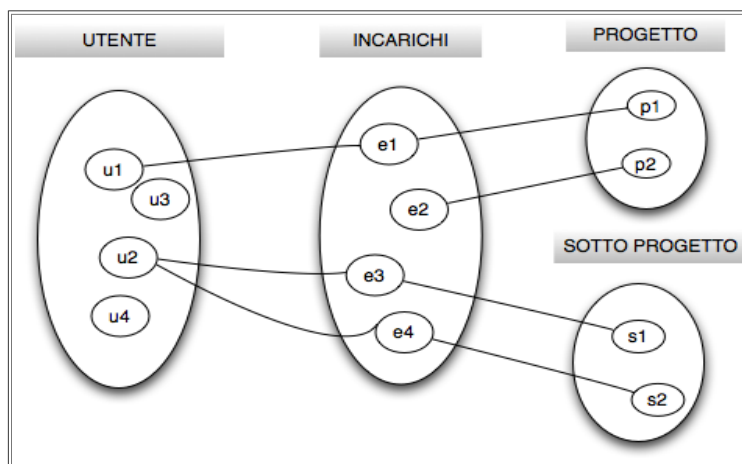
**Relazione tra utente registrato e progetto:** ogni utente registrato può essere iscritto oppure no ad uno o più progetti ( inteso come progetto manageriale unito a tutti i suoi sotto\_progetti tecnici). L'iscrizione al progetto non implica tuttavia una collaborazione attiva. Ad esempio la figura di Normal User viene implementata proprio nel caso un utente voglia seguire un progetto da vicino iscrivendosi allo stesso e non avendo per ora le competenze necessarie per poterlo sviluppare. Qualora in seguito venisse aggiunto un altro sotto progetto che necessiti di tali competenze, l'utente farebbe valere le sue competenze che potrebbero essere sfruttate al meglio. La relazione che lega utente registrato e progetto è N:N in quanto più utenti registrati possono essere iscritti allo stesso progetto e più progetti possono essere trattati dallo stesso utente.



---

### Note sulla relazione tra Utenti - Incarichi – Progetti – Sotto Progetti

Ogni in incarico, dipendente dalla tipologia, può essere assegnato sia al progetto per la sua gestione manageriale oppure al sotto progetto per lo sviluppo attivo del codice.



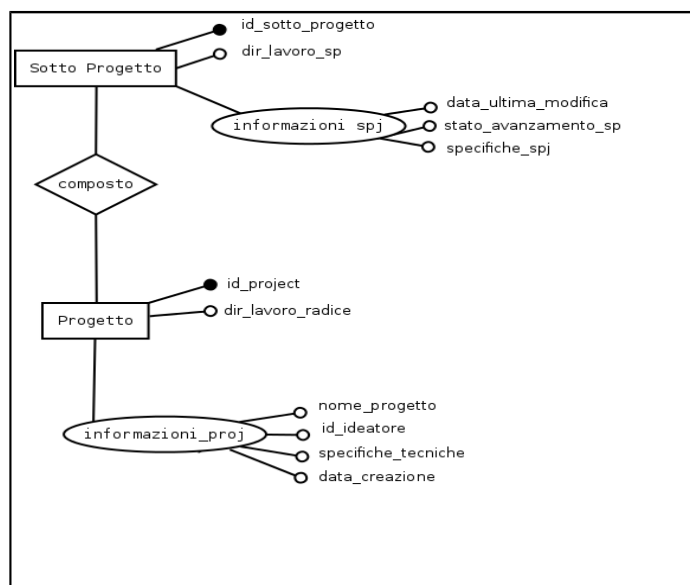
Cercando di comprendere nel dettaglio l'importante relazione che intercorre tra utenti, incarichi e progetto / sotto progetto, possiamo dire che l'incarico è definito da una chiave primaria **id\_incarico** di tipo numerico auto incrementale, ad ogni tipologia d'incarico specifico ( che è di tipo ENUM in quanto può assumere solo alcuni valori predefiniti quali PM, TM, TE, NR ) può essere assegnato un utente in modo opzionale. Ad esempio è possibile avere un sotto progetto che da specifiche necessita di alcuni incarichi particolari come 1 TM e 2 TE, oppure un progetto che necessiti di almeno 3 PM per far avanzare il lavoro in maniera efficace. Al determinato utente, scelto in base alle competenze dichiarate da lui stesso, verrà poi attribuito un incarico (che lo farà diventare utente incaricato) necessario al progetto o al sotto progetto. La relazione *utente incaricato - progetto* è di tipo 1:N, in quanto seppur un progetto / sotto progetto necessiti di più utenti incaricati, l'id\_utente\_incaricato<sup>3</sup> (ossia id\_utente con assegnato un id\_incarico) è attribuibile solo ad un unico progetto oppure solo ad un unico sotto progetto e corrisponderà poi all'utente che possiede le competenze uniche e specifiche per la risoluzione delle problematiche tecniche.

---

3 Vedi figura relazione di utente registrato – incarico pag.22



**Relazione tra progetto e sotto progetto:** ogni progetto viene suddiviso in differenti sotto progetti di più semplice risoluzione. La relazione tra progetto e sotto progetto è di tipo 1:N in quanto un progetto può essere scomposto in differenti sotto progetti che possono operare indipendentemente tuttavia ogni sotto progetto non può dipendere da più di un solo progetto alla volta.

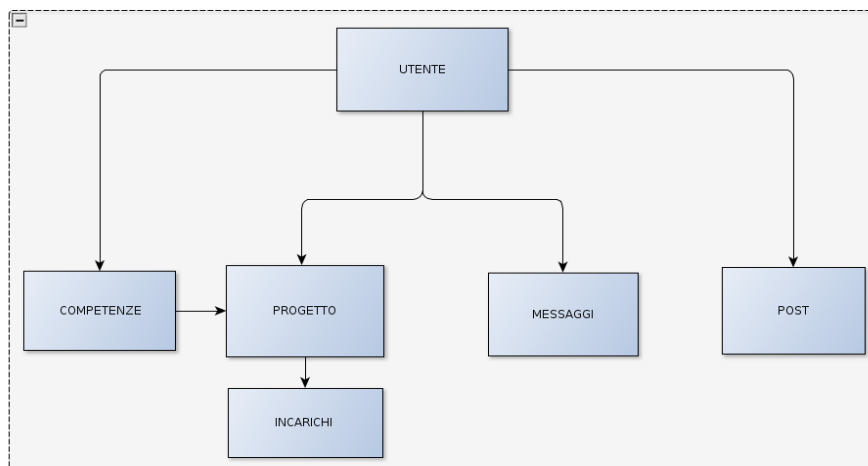


**Integrazione delle viste:** una volta sviluppate le relazioni e gli attributi è possibile effettuare un'integrazione delle viste considerando i gruppi di facile integrazione che non denotano particolari problemi di analisi o risoluzione dei conflitti. In particolare sono state identificate tre viste da unificare.

**Vista post:** con le relazioni che intercorrono tra post ed utente ( registrato – non registrato )

**Vista messaggi:** con le relazioni che intercorrono tra messaggi ed utente registrato.

**Vista progetto:** con le relazioni che riguardano lo sviluppo progettuale in sé.



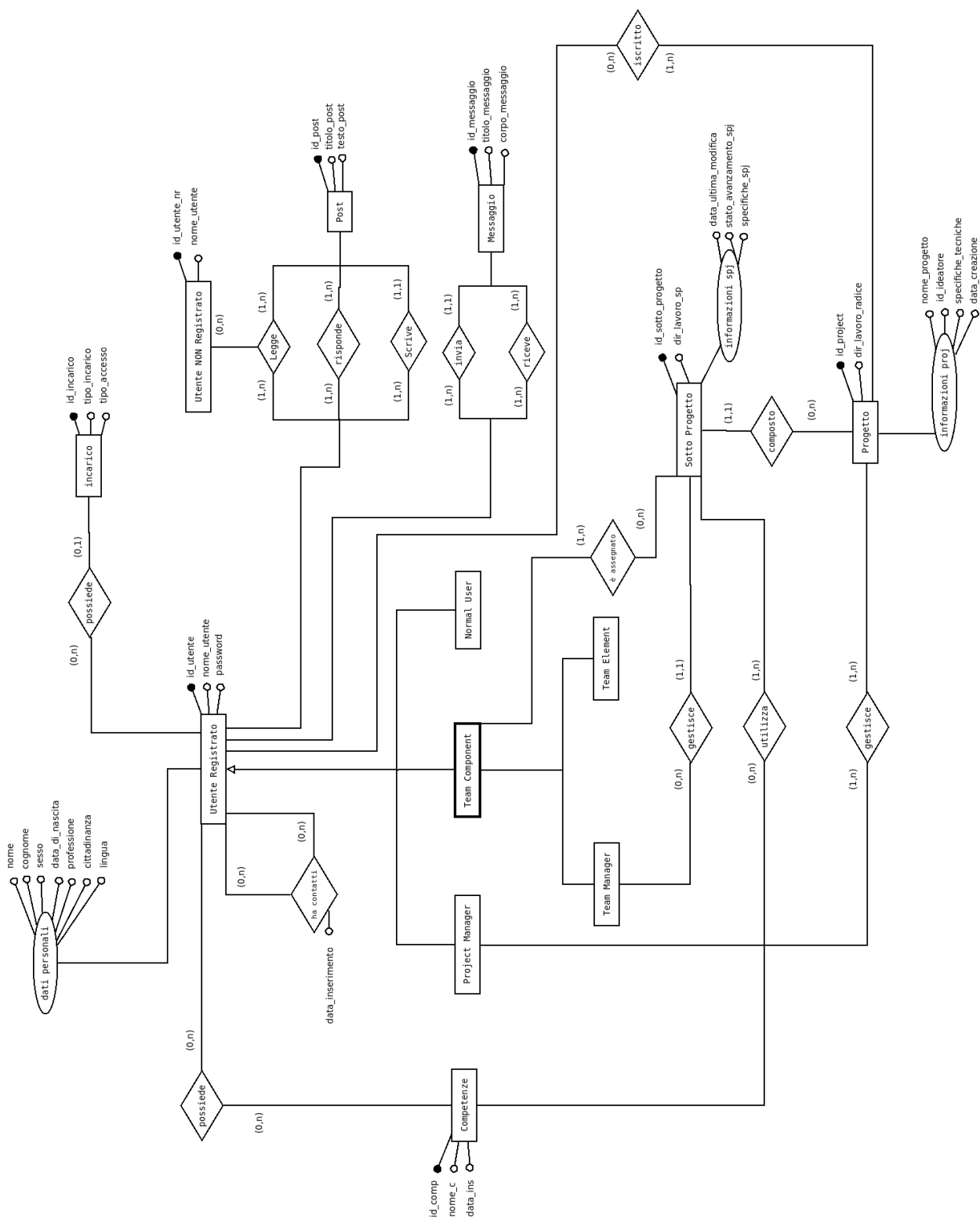
---

La progettazione del diagramma ER finale è stata sviluppata con un tool di disegno denominato Dia ( versione 0.97.1 ).

Il file in un formato leggibile dal programma Dia è disponibile nel pacchetto di file per il progetto per una più dettagliata consultazione. *Nome file: DiagrammaER.*

Nel diagramma proposto nella pagina seguente vengono sviluppati in vincoli di cardinalità determinati nelle varie relazioni con la dicitura (min,max) ossia di cardinalità minima e massima che la relazione tra due entità può esprimere.

Viene seguito lo schema di integrazione sopra riportato seguendo pariteticamente tutte le relazioni tra entità sviluppate nel presente capitolo ottenendo come risultato uno schema finale esposto nella pagina seguente.



---

## Progettazione logica

**Ottimizzazione:** *individuare almeno una scelta progettuale che necessita di analisi del carico di lavoro (esempi: attributo o associazione derivata, porzione di schema ristrutturabile tramite accorpamenti orizzontali/verticali, semplificazione di gerarchia); definire quindi tutte le informazioni (volume dei dati e specifica delle operazioni) che sono utili per calcolare i costi di accesso sulla base dei quali operare la scelta ottimale.*

Nella fase di ottimizzazione lo schema concettuale viene rivisto il progetto tenendo in considerazione il carico di lavoro, dove per carico di lavoro intendiamo l'insieme delle attività effettuate in fase di run time considerate come le più frequenti<sup>4</sup>.

Il carico di lavoro viene espresso principalmente secondo due parametri:

- quantità di dati: viene specificato il numero stimato di istanze per ogni entità ed associazione allo schema con dei valori approssimati ma indicativi
- descrizione delle operazioni che il DB dovrà supportare: sviluppate tramite schemi di navigazione, tipologie di operazioni effettuate e frequenze di attivazione.

Una delle scelte progettuali effettuate è stata quella di separare le due entità : utente non registrato ed utente registrato per trattarle separatamente. Possiamo pensare al numero di utenti NON registrati come entità dinamiche che rispecchiano il numero di operatori on-line sul sito ma che sono solo di passaggio per caso o per curiosità legata ad esempio ad un particolare interesse specifico derivante da una ricerca su un motore. Gli utenti non registrati vengono quindi creati con un ID all'apertura della pagina di un qualunque utente ed eliminati dal sistema dopo un prefissato periodo di tempo che ne permette ad esempio di trarne opportune informazioni statistiche. Questa scelta permette di evitare che memoria e capacità del sistema non siano usate in modo inefficiente separando entità più dinamiche che comunque prevedono meno attributi di gestione da entità meno dinamiche che prevedono un congruo numero di attributi.

Gli utenti registrati devono invece possedere una stabilità maggiore all'interno del sistema e del DB e pertanto intuitivamente possiamo pensare di incorporare utenti registrati ed utenti non registrati in quanto soggetti che determinano ad un carico di lavoro e a processi differenti.

A prova di quanto esposto è possibile procedere nel calcolo dei costi per le due casistiche:

- *carico non generalizzato con entità scorperate utente registrato ed utente non registrato*
- *carico generalizzato con utente che comprende utente non registrato ed utente registrato*

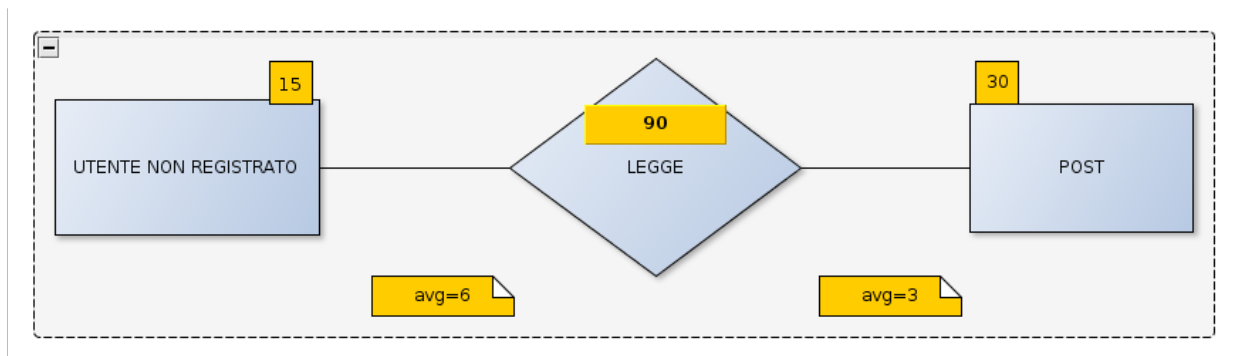
Affrontiamo nel dettaglio quanto esposto.

---

4 Come da Lezione 3.2 corso DBIS del prof Alessandro Aldini lastrina 2/16

## Analisi entità scorporate utente registrato ed utente non registrato

### 1. Utente non registrato



Analisi del carico di lavoro:

**UTENTE NON REGISTRATO – POST** : la relazione è (N:N) molti a molti.

UTENTE NON REGISTRATO : attualmente presenti nel DB esistono 15 istanze.

Da un punto di vista del sito web significa che sono presenti on line 15 persone che non hanno effettuato una registrazione ma che sono attive e stanno leggendo i post presenti al suo interno.

POST: attualmente sono presenti nel DB 30 istanze. Esistono un totale di 30 post che sono stati inseriti dagli utenti registrati per poter avere aiuto o per esprimere un giudizio.

Ogni Utente non registrato legge in media (avg=6) 6 post presenti, quindi potrebbero esistere 90 (6\*15) istanze di lettura.

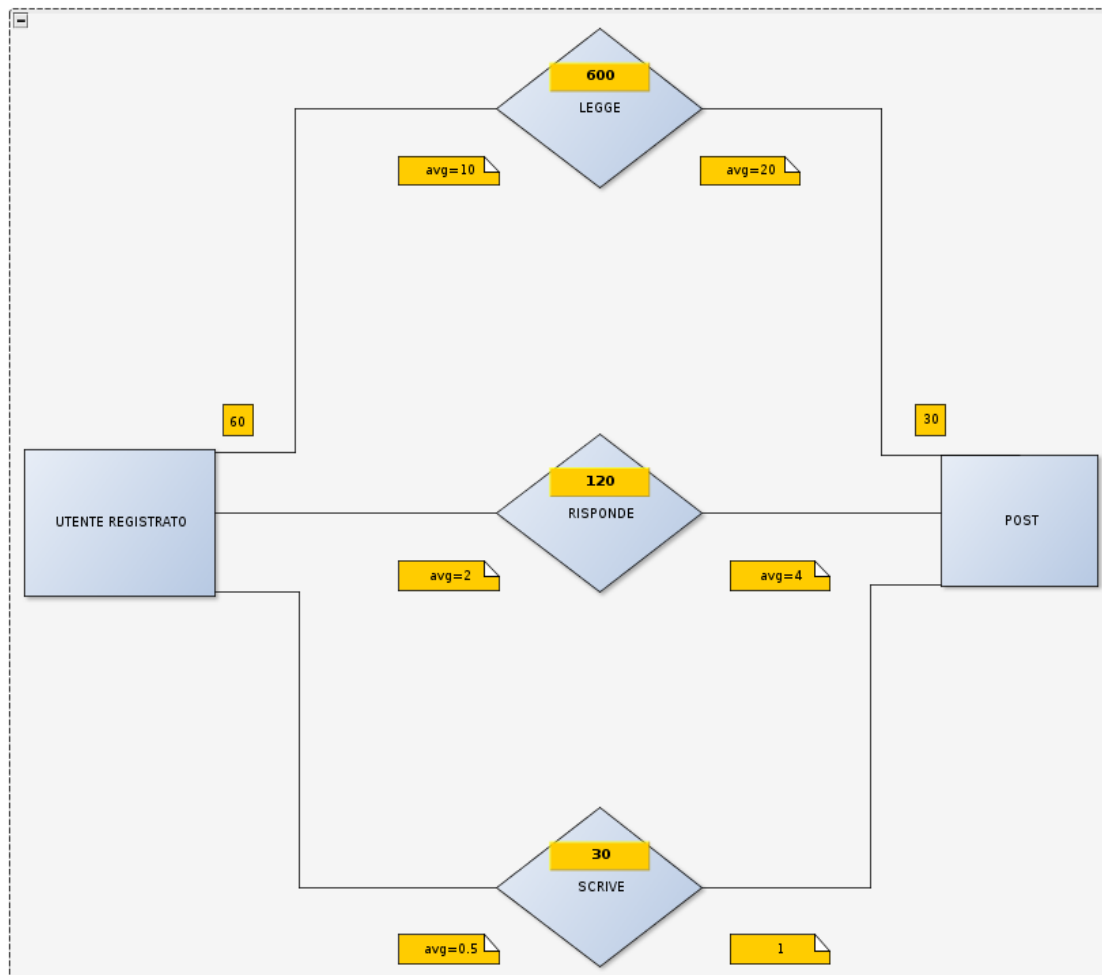
Ogni post è letto in media da (avg=3) utenti non registrati per un totale di (3\*30) 90 istanze.

Un esempio d'istanza dell'associazione LEGGE e composta dalle chiavi esterne

- id\_lettura
- data\_lettura
- **id\_utente\_non\_registrato**
- **id\_post**

In questo modo tutti i post letti dal determinato utente vengono tracciati, possono in questo modo essere effettuate delle statistiche sui post maggiormente letti determinando in questo modo la tipologia di problematica che ha maggior persistenza tra differenti utenti e che magari necessita di più supporti tecnici.

## 2. Entità utente registrato



**UTENTE REGISTRATO – POST:** possiamo notare che abbiamo 3 relazioni in questo caso, infatti l'utente registrato LEGGE, RISPONDE, SCRIVE.

**La relazione è N:N moti a moti per quanti riguarda lettura e risposta al post**

**La relazione è 1:N per quanto riguarda la scrittura in quanto ogni post ha un solo utente originante.**

Per tutte le relazioni il numero di UTENTI REGISTRATI supposto e presente nel database è stato supposto essere di 60 unità.

Ogni Utente Registrato legge una media di 10 post, risponde ad una media di 2 post (si presuppone che il numero sia inferiore a quelli letti, in quanto prima di poter rispondere un post deve necessariamente essere letto), inoltre ogni utente registrato scrive una media di 0.5 post, ossia un utente su due scrive ha necessità tecniche e quindi apre un post al riguardo.

Per quanto riguarda l'entità POST possiamo osservare che ogni post

- è in media letto 20 volte;

- ha in media 4 risposte;
- ha esattamente uno scrivente ( 1:N).

Un esempio d'istanza dell'associazione LEGGE è composta dagli attributi:

- id\_lettura (PK)
- data\_lettura
- **id\_utente**
- **id\_post**

Un esempio d'istanza dell'associazione RISPONDE è composta dagli attributi:

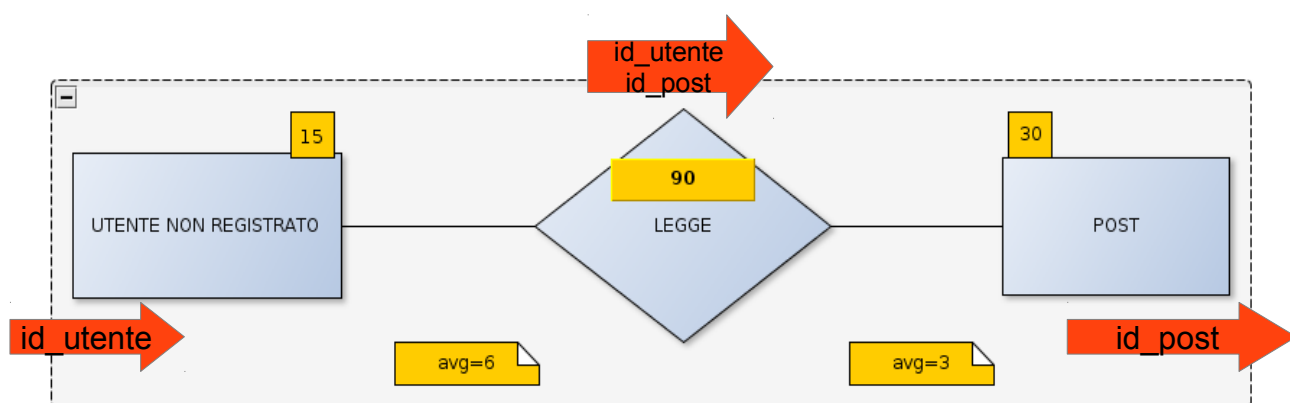
- id\_risposta (PK)
- data\_risposta
- **id\_utente**
- **id\_post\_origine**

Un esempio d'istanza dell'associazione SCRIVE è composta dagli attributi:

- id\_scrittura (PK)
- data\_post
- **id\_post**
- **id\_utente**

**Ora sviluppiamo le operazioni per poter poi effettuare un paragone:**

**1) operazione 1: definire un elenco dei messaggi letti dal particolare utente non registrato.**



Le operazioni da effettuarsi in questo caso sono :

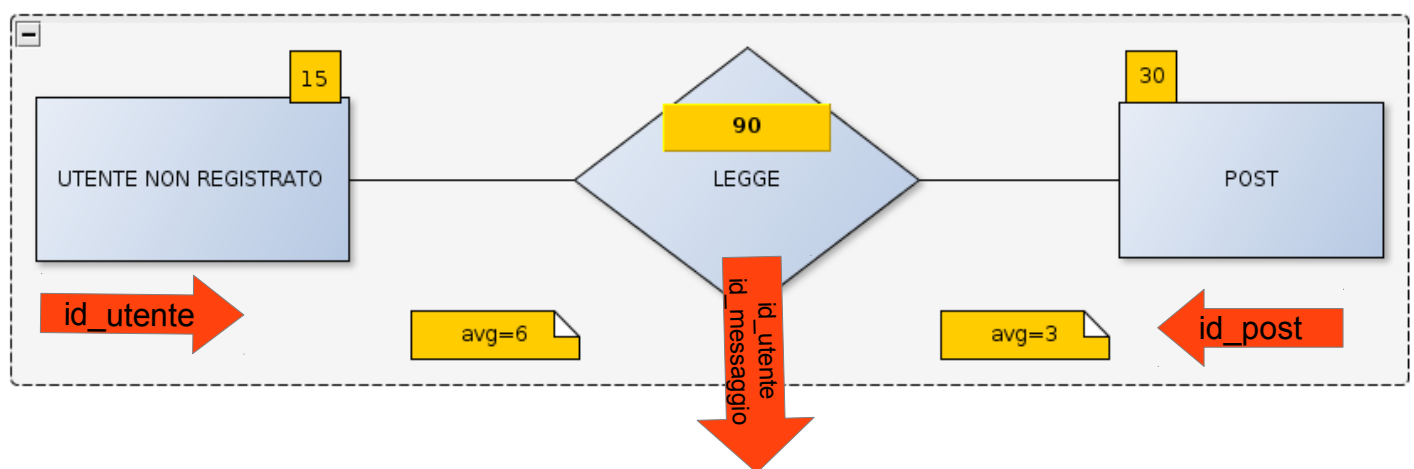
- leggere l'istanza dell'entità utente non registrato
- leggere le associazioni, una media di 6 post letti su 30 esistenti.
- Leggere i post identificati dalla precedente operazione

Analizziamo i costi con la seguente tabella

| E-R                   | READ | WRITE |
|-----------------------|------|-------|
| UTENTE NON REGISTRATO | 1    | 0     |
| LEGGE                 | 6    | 0     |
| POST                  | 6    | 0     |

Le operazioni trattate sono quindi :  $6+6+1 = 13$  operazioni di lettura

**2) Operazione 2: vogliamo aggiungere un determinato post che è stato letto dal particolare utente non registrato.**



Seguendo lo schema di navigazione le operazioni da effettuarsi nei seguenti casi sono:

- leggere l'istanza dell'entità utente non registrato
- leggere l'istanza dell'entità post
- scrivere l'istanza all'associazione LEGGE



---

Questa la tabella che riassume le informazioni di lettura e scrittura sul diagramma E-R

| E-R                   | READ | WRITE |
|-----------------------|------|-------|
| UTENTE NON REGISTRATO | 1    | 0     |
| LEGGE                 | 0    | 1     |
| POST                  | 1    | 0     |

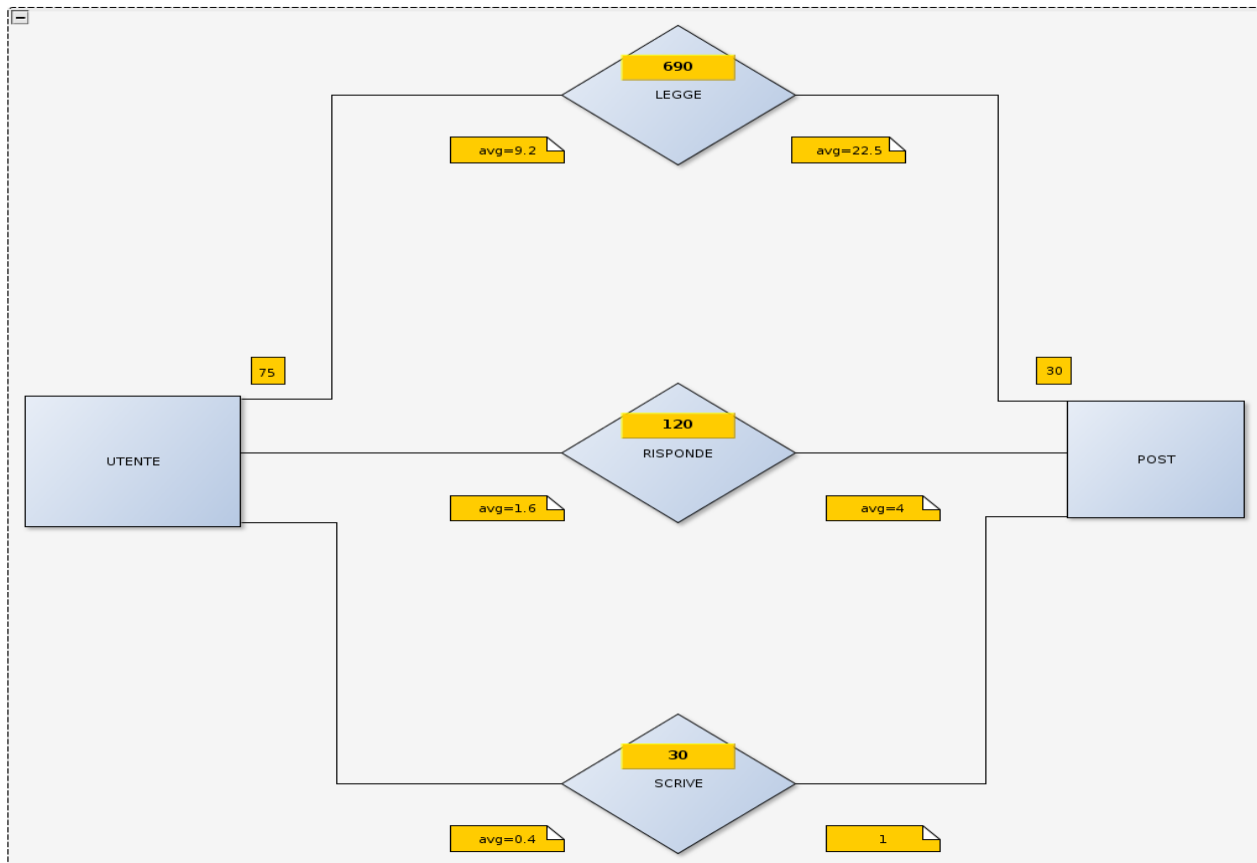
Le operazioni trattate sono quindi :  $1+1+1 = 3$  operazioni ( 2 di lettura + 1 di scrittura )

Ricapitolando le due operazioni di ricerca e di inserimento di una nuova informazioni all'interno del DB comportano un totale di  $13 + 3$  operazioni totali.

Le stesse operazioni saranno trattate in seguito per la comparazione di dati in caso di accorpamento delle due entità: utente registrato ed utente non registrato.

## Analisi del carico di lavoro delle entità NON scorporate

### Entità utente – post

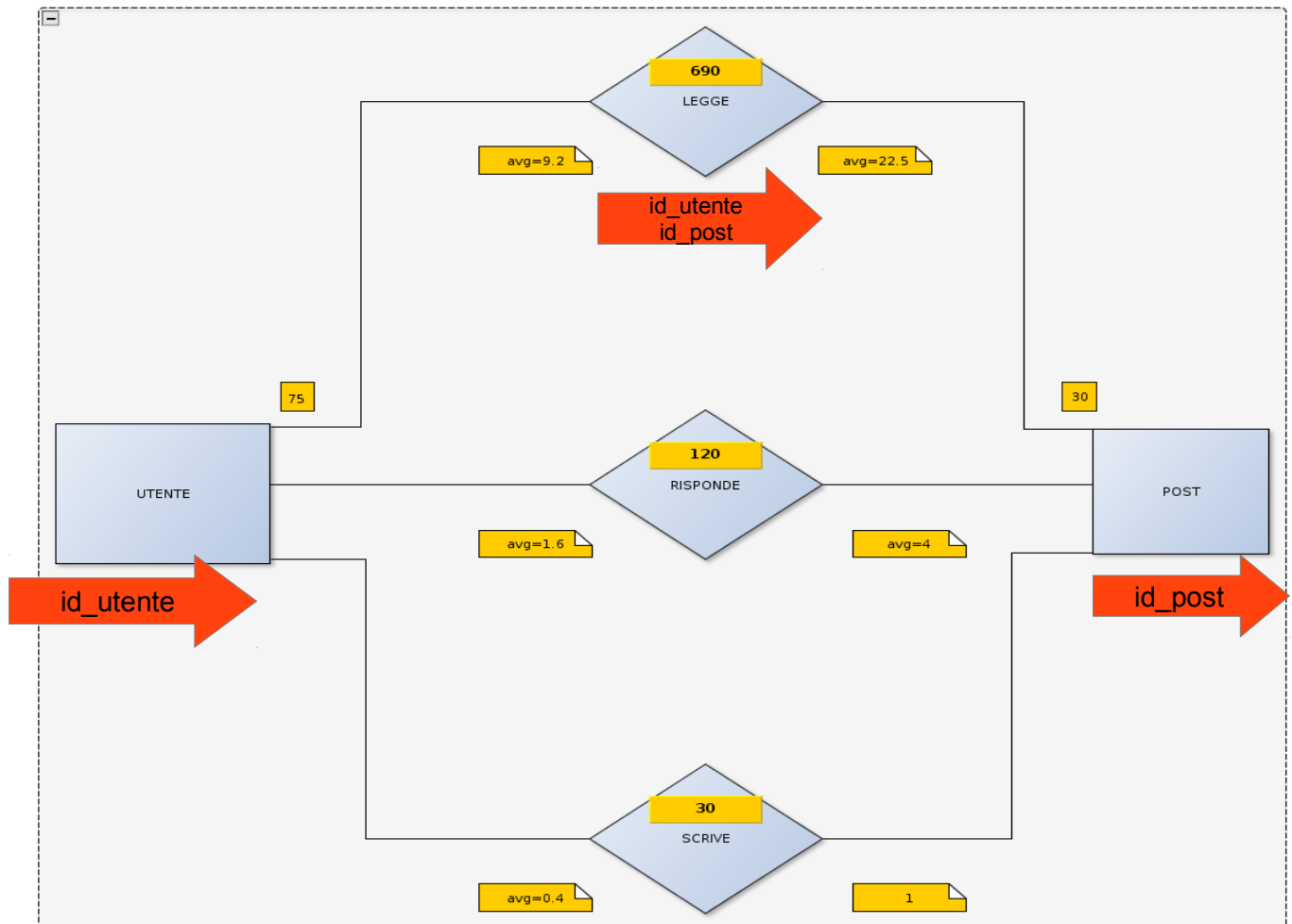


Analizzando il grafico possiamo notare che l'entità utente deve ora accorparsi sia l'utente registrato che l'utente non registrato al sito, pertanto comparando i dati con la precedente valutazione possiamo notare che :

- il numero di utenti è derivante dalla sommatoria delle precedenti entità ( 60 + 15 )
- il numero di post è rimasto invariato
- le medie per le associazioni sono variate in funzione dell'accorpamento dei dati esposti nelle assunzioni di entità scorporate.

Ora ritraiamo le operazioni trattate in precedenza prima della generalizzazione:

### Operazione 1: definire un elenco dei messaggi letti dal particolare utente indicato



Le operazioni da effettuarsi in questo caso sono :

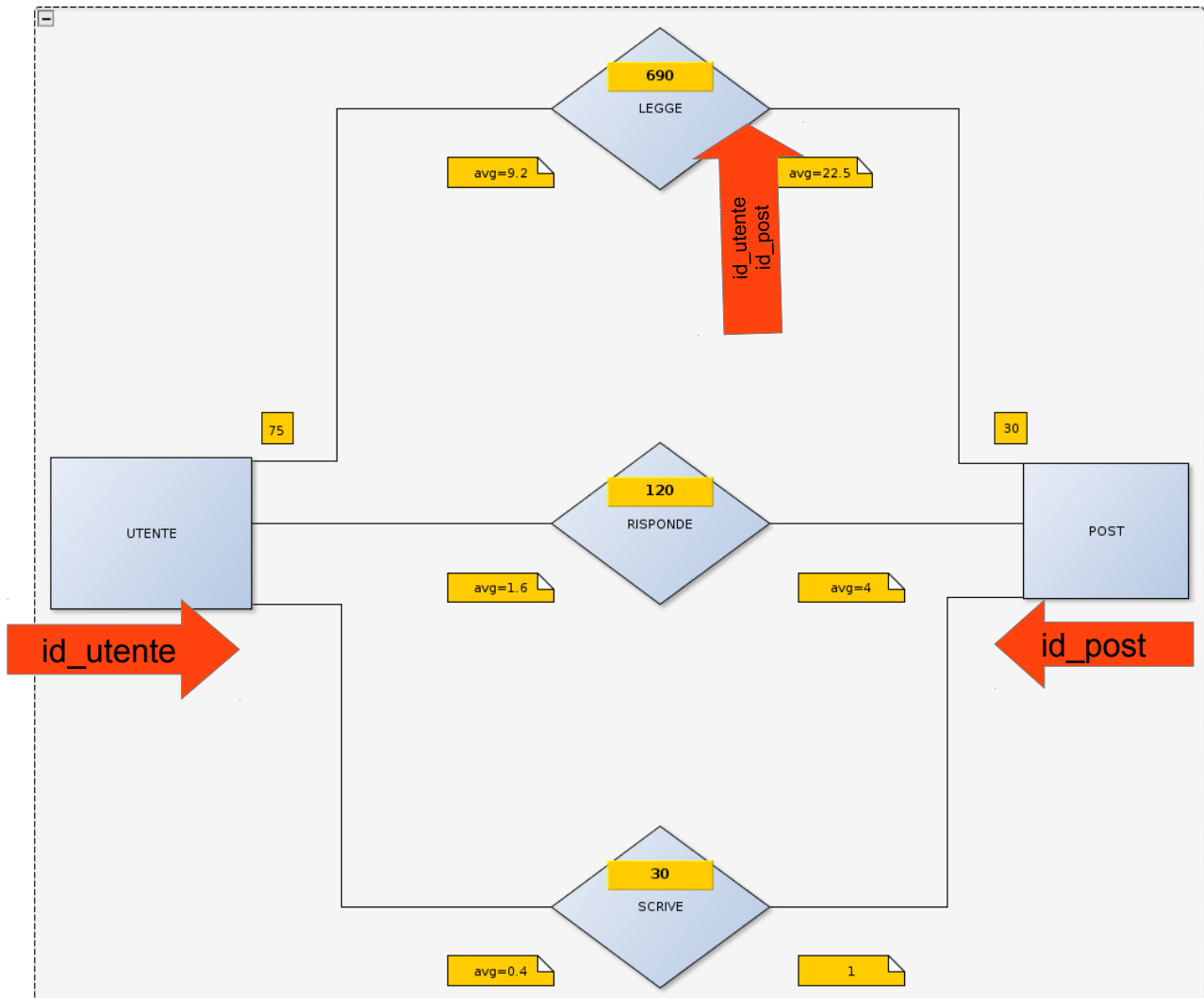
- leggere l'istanza dell'entità utente
- leggere le associazioni, una media di 9.2 post letti su 30 esistenti. ( dato arrotondato a 9 )
- Leggere i post identificati dalla precedente operazione

Analizziamo i costi con la seguente tabella

| E-R                   | READ | WRITE |
|-----------------------|------|-------|
| UTENTE NON REGISTRATO | 1    | 0     |
| LEGGE                 | 9    | 0     |
| POST                  | 9    | 0     |

Le operazioni trattate sono quindi :  $9+9+1 = 19$  operazioni di lettura

**Operazione 2: vogliamo aggiungere in elenco un determinato post che è stato letto da un particolare utente.**



Seguendo lo schema di navigazione le operazioni da effettuarsi nei seguenti casi sono:

- leggere l'istanza dell'entità utente
- leggere l'istanza dell'entità post
- scrivere l'istanza all'associazione LEGGE

Questa la tabella che riassume le informazioni di lettura e scrittura sul diagramma E-R

| E-R    | READ | WRITE |
|--------|------|-------|
| UTENTE | 1    | 0     |
| LEGGE  | 0    | 1     |
| POST   | 1    | 0     |

Le operazioni trattate sono quindi :  $1+1+1 = 3$  operazioni ( 2 di lettura + 1 di scrittura )

Ora possiamo mettere a confronto i dati nelle due casistiche in modo da poter effettuare un confronto delle operazioni eseguite.

| <i>E-R</i>   | <i>ENTITA' SCORPORATE</i> |       | <i>ENTITA' UNITE</i> |       |
|--------------|---------------------------|-------|----------------------|-------|
|              | READ                      | WRITE | READ                 | WRITE |
| OPERAZIONE 1 | 13                        | 0     | 19                   | 0     |
| OPERAZIONE 1 | 2                         | 1     | 2                    | 1     |
| TOTALE       | 15                        | 1     | 21                   | 1     |

Dalla tabella risulta evidente che il numero di operazioni separando le due entità UTENTE NON REGISTRATO e UTENTE REGISTRATO, risulta inferiore al numero di operazioni con un'unica entità generalizzata e pertanto il modello ad entità scorporate viene scelto come modello progettuale per la realizzazione del DB in forma fisica.

---

## Semplificazione

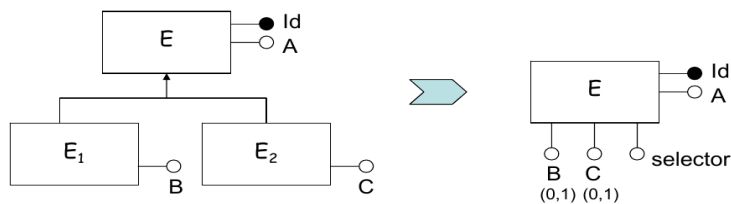
Intendiamo ristrutturare gerarchie di generalizzazione, attributi composti e valori multipli, identificatori esterni.

Procediamo alla semplificazione degli schemi ER con le seguenti modalità<sup>5</sup>:

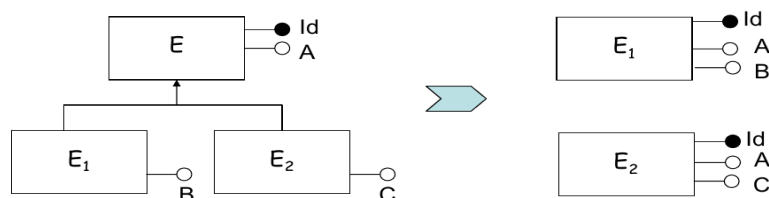
- generalizzazione delle gerarchie
- attributi composti
- attributi a valore multiplo
- identificatori esterni/misti

## Generalizzazione delle gerarchie

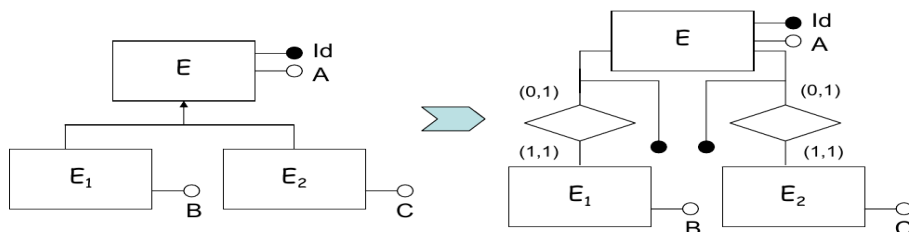
1. possiamo semplificare la gerarchia con la generalizzazione di entità ponendo un selettore atto a distinguere i membri nelle differenti specializzazioni<sup>6</sup>.



2. Le gerarchie convergono a una particolare entità



3. La generalizzazione di gerarchie è modellata tra le entità coinvolte:

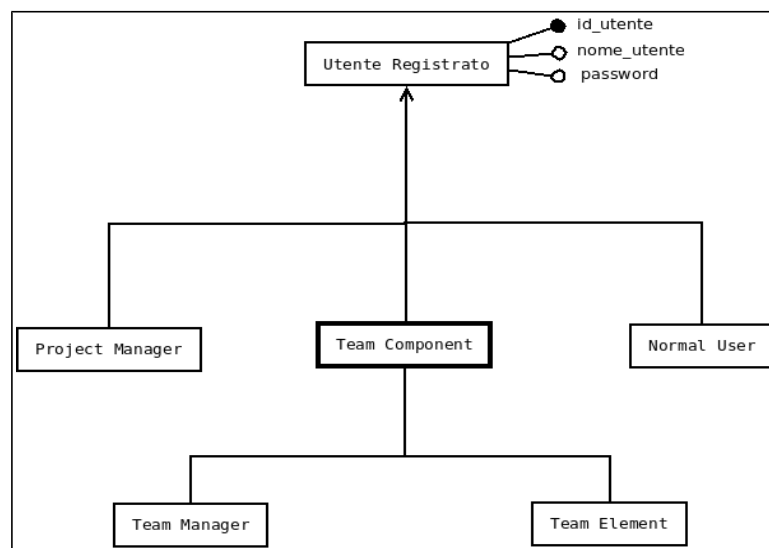


---

<sup>5</sup> Lezione 3.3 corso DBIS del prof Alessandro Aldini

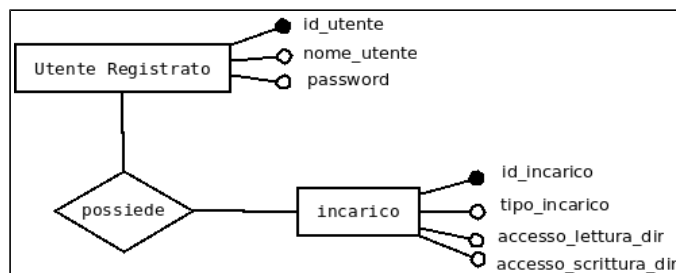
<sup>6</sup> Lezione 3.3 corso DBIS del prof Alessandro Aldini lastrina 3/13

Dal diagramma E-R sviluppato possiamo notare la seguente generalizzazione



Applicando la prima regola possiamo determinare un selettore atto a differenziare le differenti entità quali PM, TM, TE, NU.

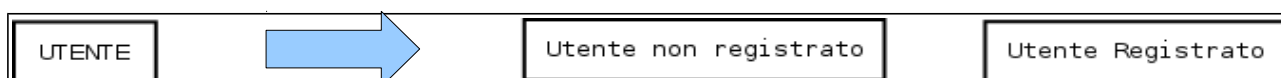
Il selettore viene determinato in questo caso da una entità differente: INCARICO che interviene solo qualora all'utente registrato sia stato assegnato effettivamente un incarico all'interno di un progetto/sotto-progetto.



L'entità INCARICO funziona in realtà da selettore per tutti i tipi di utenza ( eccetto per gli utenti non registrati ). Tale entità oltre ad attribuire un incarico specifico all'utente fornisce anche dei particolari accessi in lettura e scrittura alle directory di lavoro.

Al momento della creazione del DB verrà quindi creata un'unica entità utente registrato.

Come già descritto nel processo di ottimizzazione, ho applicato la regola 2 sopra descritta per ottenere entità separate dall'entità utente quali:



## Attributi composti

L'attributo composto viene eliminato mentre le sue sotto componenti divengono atomiche.

Come viene mostrato a titolo di esempio dalla seguente figura<sup>7</sup>:



L'attributo composto può anche venire eliminato trasformando gli attributi in un unico elemento atomico:



Questa semplificazione dello schema E-R è applicabile alle ENTITA' nel diagramma che prevedono attributi composti quali:

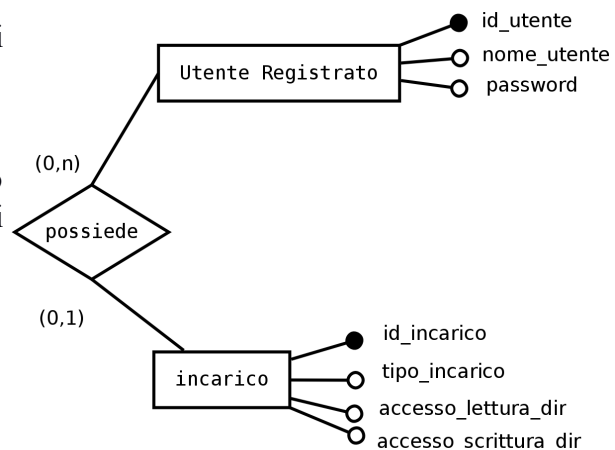
- Utente Registrato → attributo composto: Dati Personali
- Progetto → attributo composto: Informazioni Progetto
- Sotto-progetto → attributo composto: Informazioni sotto progetto

Viene applicata pertanto la trasformazione sopradescritta.

## Attributo multi valore

L'entità incarico è stata scorporata dagli attributi dell'entità di utente registrato.

Incarico sarà poi correlata alle entità Progetto e Sotto Progetto tramite la relazione di assegnazione.





---

## Traduzione

Passare da uno schema E-R ristrutturato al modello relazionale, evidenziando chiavi primarie ed importate.

Regole:

- ogni entità diventa una relazione
- ogni identificatore primario diventa una chiave primaria

Entità e relazioni definite nel diagramma E-R:

**1. utente\_registrato**

(id\_utente, nome\_utente, password, nome, cognome, sesso, data\_di\_nascita, professione, cittadinanza, lingua)

**2. contatti** (Id\_contatto, Id\_utente, Id\_utente\_contatto)

FK:

Id\_utente → utente\_registrato(id\_utente)

Id\_utente\_contatto → utente\_registrato(id\_utente)

**3. possesso\_c** (id\_possesso, id\_utente, id\_competenza)

FK:

id\_utente → utente\_registrato(id\_utente)

id\_competenza → competenze(id\_comp)

**4. lettura\_post**(id\_lettura, data\_lettura, id\_post\_letto, id\_lettore)

FK:

id\_post\_letto → post(id\_post)

id\_lettore → utente\_registrato(id\_utente)

**5. lettura\_post\_utente\_nr** (id\_lettura, data\_lettura, id\_post\_letto, id\_lettore)

FK:

id\_post\_letto → post(id\_post)

id\_lettore → utente\_non\_registrato(id\_utente\_nr)

---

**6. risposta\_post** (id\_risposta, data\_risposta, id\_post\_origine, id\_utente\_rispondi)

FK:

id\_post\_origine → post(id\_post)

id\_utente\_rispondi → utente\_registrato (id\_utente)

**7. scrittura\_post** (id\_scrittura, data\_scrittura, id\_post\_scritto, id\_utente\_scrittore)

FK:

id\_post\_scritto → post(id\_post)

id\_utente\_scrittore → utente\_registrato (id\_utente)

**8. utente non\_registrato** (id\_utente\_nr, nome\_utente)

**9. competenze** (id\_comp, nome\_c , data\_ins)

**10. uso\_competenza** (id\_uso\_c, id\_comp, id\_sotto\_pj,)

FK

id\_comp → competenze(id\_comp)

id\_sotto\_pj → sotto-progetto(id\_sotto\_progetto)

**11. progetto**

(id\_project, dir\_lavoro\_radice, nome\_progetto, id\_ideatore, specifiche\_tecniche, data\_creazione)

**12. incaricato** (id\_utente\_incaricato, id\_utente, id\_incarico)

FK:

id\_utente → utente\_registrato (id\_utente)

id\_incarico → incarico(id\_incarico)

**13. sotto-progetto**(id\_sotto\_progetto, dir\_lavoro\_sp, data\_ultima\_modifica, stato\_avanzamento\_spj, specifiche\_spj)

**14. composizione** (id\_composizione\_pj, id\_progetto, id\_sotto\_pj)

FK:

id\_progetto → progetto (id\_project)

id\_sotto\_pj → sotto-progetto (id\_sotto\_progetto)

---

**15. incarico** (id\_incarico, tipo\_accesso) // esistono solo 4 tipi di incarico con 2 tipi di accesso.

**16. incarico\_di\_progetto** ( id\_proj\_managment, id\_incarico, id\_pj)

FK:

id\_incarico → incarico(id\_incarico)

id\_pj → progetto(id\_project)

**17. incarico\_di\_sotto\_progetto** ( id\_gruppo\_di\_lavoro, id\_utente\_incaricato, id\_sotto\_progetto)

FK:

id\_utente\_incaricato → incaricato(id\_incarico)

id\_sotto\_progetto → sotto-progetto(id\_sotto\_progetto)

**18. post**(id\_post, titolo\_post, testo\_post)

**19. messaggi** (id\_messaggio, titolo\_messaggio, corpo\_messaggio)

**20. messaggi\_inv\_ric** (id\_invio, id\_messaggio\_inv, id\_utente\_inv, id\_utente\_ric, data\_invio)

FK:

id\_messaggio\_inv → messaggio (id\_messaggio)

id\_utente\_inv → utente\_registrato (id\_utente)

id\_utente\_ric → utente\_registrato (id\_utente)

---

## Normalizzazione

### Verificare le forme normali 2NF , 3NF , BCNF.

I livelli di normalizzazione permettono di eliminare la ridondanza da una relazione, scomponendola in una serie di relazioni di più piccole dimensioni e maggiormente gestibili.

Uno dei principali obiettivi imposti dalla normalizzazione è quello di evitare problematiche legate alle anomalie di aggiornamento dei dati quali: modifica, inserimento e cancellazione.

Vengono ora verificate le principali forme normali per il DB:

2NF: riguarda le tabelle in cui la chiave primaria sia composta da più attributi e stabilisce che, in questo caso, tutte le colonne corrispondenti agli altri attributi dipendano dall'intera chiave primaria<sup>8</sup>.

Nel DB sviluppato vengono costruite chiavi primarie a valore auto incrementale per ogni tipologia di tabella e non esistono chiavi composte.

Ogni attributo presente in tabella dipende solo dalla chiave intera che risolve il problema della forma normale 2NF.

3NF ( dipendenza transitiva): stabilisce che non esistono dipendenze tra le colonne di una tabella se non basate su una chiave primaria. Inoltre una relazione è 3NF se e solo se è anche 2NF.

La normalizzazione in questo caso avverrà individuando le dipendenze indirette e creando con queste nuove relazioni.

**Nel DB proposto l'analisi di tutte le relazioni sviluppate risulta essere 3NF.**

BCNF: la forma normale di Boyce-Codd è la più completa tra le forme normali conosciute.

Per poter definire la BCNF è necessario introdurre la dipendenza funzionale:

fornirò un esempio non correlato con il DB in uso:

analizziamo una tabella dei codici di avviamento postale con questi attributi

CAP( cap, nome\_città , provincia)

osserviamo che il CAP è un codice di 5 caratteri direttamente associato ad una sola città.

Per ogni CAP esiste una sola città, una sola provincia che sono legate al CAP e fanno del CAP la chiave principale.

Possiamo affermare quindi che nome\_città e provincia dipendano funzionalmente dal CAP che le determina.

$CAP \rightarrow nome\_città \mid CAP \rightarrow provincia$

**Una relazione è BCNF se non ci sono dipendenze funzionali ( $X \rightarrow A$ ) che non dipendano dalla chiave primaria<sup>9</sup>.**

---

<sup>8</sup> Daniela Dorbolò, *Guida a SQL*, McGrawHill, 2009, pag. 27

<sup>9</sup> Chiave primaria: è una particolare superchiave che il progettista del DB definisce come modo preferito per identificare le tuple di una tabella.

---

Una relazione R che possiede delle dipendenze funzionali è nella BCNF se:

**per ogni**  $X \rightarrow Y$  ( dipendenza funzionale di Y da X ) X è anche una superchiave ( ossia un attributo o un insieme di attributi che determina tutti gli altri attributi, pensando alla definizione di DF, e che non è mai duplicata ed è quindi anche una PK).

Posto che le tabelle sono tutte 3NF ed hanno tutte **una sola chiave** allora possiamo affermare che sono BCNF se le dipendenze funzionali sono rispettate rispetto a quella chiave.

Possiamo verificare quanto detto andando ad individuare le chiavi candidate nelle relazioni del DB per verificare la definizione sopra esposta di BCNF:

**1. utente\_registrato**

(id\_utente, nome\_utente, password, nome, cognome, sesso, data\_di\_nascita, professione, cittadinanza, lingua)

Chiavi candidate:

- id\_utente (PK)
- nome\_utente , password [id\_utente  $\rightarrow$  nome\_utente , password]
- nome, cognome, data\_di\_nascita [id\_utente  $\rightarrow$  nome, cognome, data\_di\_nascita]

**La relazione è in BCNF**

**2. contatti (Id\_contatto, Id\_utente, Id\_utente\_contatto)**

Chiavi candidate:

- Id\_contatto (PK)
- Id\_utente, Id\_utente\_contatto [id\_utente  $\rightarrow$  id\_utente, id\_utente\_contatto]

**La relazione è in BCNF**

**3. possesso\_c (id\_possezzo, id\_utente, id\_competenza)**

Chiavi candidate:

- id\_possezzo (PK)
- id\_utente , id\_competenza [ id\_possezzo  $\rightarrow$  id\_utente , id\_competenza ]

**La relazione è in BCNF**

**4. lettura\_post (id\_lettura, data\_lettura, id\_post\_letto, id\_lettore)**

Chiavi candidate:

- id\_lettura (PK)
- id\_post\_letto , id\_lettore [ id\_lettura  $\rightarrow$  id\_post\_letto, id\_lettore]

**La relazione è in BCNF**

---

**5. lettura\_post\_utente\_nr** (id\_lettura,data\_lettura,id\_post\_letto,id\_lettore)

Chiavi candidate:

- id\_lettura (PK)
- id\_post\_letto , id\_lettore [ id\_lettura  $\rightarrow$  id\_post\_letto, id\_lettore]

**La relazione è in BCNF**

**6. risposta\_post**

(id\_risposta,data\_risposta,id\_post\_origine,id\_post\_risposta,id\_utente\_risposta)

Chiavi candidate:

- id\_risposta (PK)
- id\_post\_origine,id\_utente\_risposta ,id\_utente\_risposta [ id\_risposta  $\rightarrow$  id\_post\_origine,id\_post\_risposta,id\_utente\_risposta]

*// esiste solo un id\_risposta che abbia un utente che risponda con un post ad un altro post.*

**La relazione è in BCNF**

**7. scrittura\_post** (id\_scrittura,data\_scrittura,id\_post\_scritto,id\_utente\_scrittore)

Chiavi candidate:

- id\_scrittura (PK)
- id\_post\_scritto,id\_utente\_scrittore [ id\_scrittura  $\rightarrow$  id\_post\_scritto,id\_utente\_scrittore]

**La relazione è in BCNF**

**8. utente non\_registrato**

(id\_utente\_nr,nome\_utente)

Chiavi candidate:

- id\_utente\_nr (PK)

**La relazione è in BCNF**

**9. competenze**

(id\_comp , nome\_c ,data\_ins)

Chiavi candidate:

- id\_comp (PK)
- nome\_c ( chiave candidata ) [ id\_comp  $\rightarrow$  nome\_c]

**La relazione è in BCNF**

---

**10. uso\_competenza** (id\_uso\_c, id\_comp, id\_sotto\_pj,) // *tabella delle competenze NECESSARIE ai sotto progetti.*

Chiavi candidate:

- id\_uso\_c (PK)
- id\_comp, id\_sotto\_pj [id\_uso\_c  $\rightarrow$  id\_comp, id\_sotto\_pj ]

**La relazione è in BCNF**

**11. progetto**

(id\_project, dir\_lavoro\_radice, nome\_progetto, id\_ideatore, specifiche\_tecniche, data\_creazione, stato\_avanzamento\_pj)

Chiavi candidate:

- id\_project (PK)

**La relazione è in BCNF**

**12. sotto-progetto**

(id\_sotto\_progetto, dir\_lavoro\_sp, data\_ultima\_modifica, stato\_avanzamento\_spj, specifiche\_spj)

Chiavi candidate:

- id\_sotto\_progetto (PK)

**La relazione è in BCNF**

**13. composizione\_pj** (id\_composizione\_pj, id\_progetto, id\_sotto\_pj)

Chiavi candidate:

- id\_composizione (PK)
- id\_progetto, id\_sotto\_pj [ id\_composizione  $\rightarrow$  id\_progetto, id\_sotto\_pj ]

**La relazione è in BCNF**

**14. incarico** (id\_incarico, tipo\_incarico, accesso\_lettura\_dir, accesso\_scrittura\_dir)

Chiavi candidate:

- id\_incarico (PK)

E' possibile notare che l'accesso in lettura o in scrittura dipende funzionalmente dall'incarico svolto:  $\text{tipo\_incarico} \rightarrow \text{accesso\_lettura\_dir}, \text{accesso\_scrittura\_dir}$ ,

---

Viene ridefinita la tabella tipologia d'incarico e tipo di accesso alla directory di lavoro.

Le tabelle vengono ridefinite e scomposte nel modo seguente:

**incarico ( id\_incarico, tipo\_accesso )** // in cui il tipo di accesso può essere :NULL, r , rw

Chiavi candidate

- id\_incarico ( PK )

**La relazione è in BCNF**

**incaricato ( id\_utente\_incaricato, id\_utente, id\_incarico )**

Chiavi candidate:

- id\_utente\_incaricato (PK)

- id\_utente, id\_incarico [ id\_utente\_incaricato → id\_utente, id\_incarico ]

**La relazione è in BCNF**

**15. incarico\_di\_progetto ( id\_proj\_managment, id\_incarico, id\_pj )**

Chiavi candidate:

- id\_proj\_managment (PK)

**La relazione è in BCNF**

**16. incarico\_di\_sotto\_progetto ( id\_gruppo\_di\_lavoro, id\_utente\_incaricato, id\_sotto\_progetto )**

Chiavi candidate:

- id\_gruppo\_di\_lavoro (PK)

- id\_utente\_incaricato, id\_sotto\_progetto [ id\_gruppo\_di\_lavoro → id\_utente\_incaricato, id\_sotto\_progetto ]

**La relazione è in BCNF**

**17. post( id\_post, titolo\_post, testo\_post )**

Chiavi candidate:

- id\_post (PK)

**La relazione è in BCNF**

**18. messaggi**

( id\_messaggio, titolo\_messaggio, corpo\_messaggio )

Chiavi candidate:

- id\_messaggio (PK)



---

**La relazione è in BCNF**

**19. messaggi\_inv\_ric**

(id\_invio, id\_messaggio\_inv, id\_utente\_inv, id\_utente\_ric, data\_invio)

Chiavi candidate:

- id\_invio (PK)

- id\_messaggio\_inv, id\_utente\_inv, id\_utente\_ric [id\_invio →  
id\_messaggio\_inv, id\_utente\_inv, id\_utente\_ric]

**La relazione è in BCNF**

**Le relazioni analizzate risultano pertanto corrette e risultano in BCNF.**

---

## Database MySQL

### Implementazione del DB Relazionale in un DB MySQL

1. **utente\_registrato**(id\_utente, nome\_utente, password, nome, cognome, sesso, data\_di\_nascita, professione, cittadinanza, lingua)

```
CREATE TABLE utente_registrato (  
  id_utente int(3) NOT NULL AUTO_INCREMENT,  
  nome_utente char(20) NOT NULL UNIQUE, // il nome utente deve essere unico  
  password char(20) NOT NULL,  
  nome char(20) NOT NULL,  
  cognome char(20) NOT NULL,  
  sesso ENUM ('F' , 'M') NOT NULL,  
  data_di_nascita DATE NULL,  
  professione char (20) NULL,  
  cittadinanza char (20) NULL,  
  lingua char (20) NOT NULL,  
  PRIMARY KEY (id_utente)  
 ) ENGINE=InnoDB;
```

2. **contatti** (id\_contatto, id\_utente, id\_utente\_contatto) // registra i contatti mantenuti da ogni utente.

```
CREATE TABLE contatti (  
  id_contatto int(3) NOT NULL AUTO_INCREMENT,  
  id_utente int(3) NOT NULL,  
  id_utente_contatto int(3) NOT NULL,  
  PRIMARY KEY (id_contatto),  
  FOREIGN KEY (id_utente) REFERENCES utente_registrato (id_utente) ON DELETE  
  CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (id_utente_contatto) REFERENCES utente_registrato (id_utente) ON  
  DELETE CASCADE ON UPDATE CASCADE,  
 ) ENGINE = InnoDB;
```

- 
3. **competenze** (id\_comp, nome\_c, data\_ins) // tabella indicante l'entità competenze, i suoi valori possono essere determinati sia dalle competenze specifiche possedute da ogni utente, sia dalle competenze necessarie richieste da uno specifico progetto.

```
CREATE TABLE competenze (  
  id_comp int(3) NOT NULL AUTO_INCREMENT,  
  nome_c char (20) NOT NULL UNIQUE, // può esserci un solo nome per ogni competenza  
  data_ins DATE NOT NULL,  
  PRIMARY KEY (id_comp)  
  ) ENGINE = InnoDB;
```

4. **possesso\_c** (id\_possesso, id\_utente, id\_competenza) // tabella delle competenze possedute dagli utenti registrati

```
CREATE TABLE possesso_c (  
  id_possesso int(3) NOT NULL AUTO_INCREMENT,  
  id_utente int(3) NOT NULL,  
  id_competenza int(3) NOT NULL,  
  PRIMARY KEY (id_possesso),  
  FOREIGN KEY (id_utente) REFERENCES utente_registrato (id_utente) ON DELETE CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (id_competenza) REFERENCES competenze (id_comp) ON DELETE CASCADE ON UPDATE CASCADE  
  ) ENGINE = InnoDB;
```

5. **post**(id\_post, titolo\_post, testo\_post) // tabella indicante l'entità post.

```
CREATE TABLE post (  
  id_post int(3) NOT NULL AUTO_INCREMENT,  
  titolo_post varchar(45) NOT NULL,  
  testo_post TEXT NOT NULL,  
  PRIMARY KEY (id_post)  
  ) ENGINE = InnoDB;
```

6. **lettura\_post**(id\_lettura, data\_lettura, id\_post\_letto, id\_lettore) // tabella indicante la lettura dei post, tiene traccia di chi ha letto il determinato post e quando.

```
CREATE TABLE lettura_post (  
  id_lettura int(3) NOT NULL AUTO_INCREMENT,
```

---

```

data_lettura DATE NOT NULL,
id_post_letto int(3) NOT NULL,
id_lettore int(3) NOT NULL,
PRIMARY KEY (id_lettura),
FOREIGN KEY (id_post_letto) REFERENCES post(id_post) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (id_creatore) REFERENCES utente_registrato (id_utente) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY (id_lettore) REFERENCES utente_registrato (id_utente) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;

```

7. **lettura\_post\_utente\_nr** (id\_lettura, data\_lettura, id\_post\_letto, id\_creatore, id\_lettore) // *tabella indicante la lettura dei post degli utenti non registrati. Questi non possono effettuare altre funzioni.*

```

CREATE TABLE lettura_post_utente_nr (
id_lettura int(3) NOT NULL AUTO_INCREMENT,
data_lettura DATE NOT NULL,
id_post_letto int(3) NOT NULL,
id_lettore int(3) NOT NULL,
PRIMARY KEY (id_lettura),
FOREIGN KEY (id_post_letto) REFERENCES post(id_post) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (id_lettore) REFERENCES utente_non_registrato (id_utente_nr) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;

```

8. **risposta\_post**(id\_risposta, data\_risposta, id\_post\_origine, id\_post\_risposta) // *tabella indicante la risposta ai post, tiene traccia di chi ha risposto al determinato post e quando.*

```

CREATE TABLE risposta_post (
id_risposta int(3) NOT NULL AUTO_INCREMENT,
data_risposta DATE NOT NULL,
id_post_origine int(3) NOT NULL,
id_post_risposta int(3) NOT NULL,
id_utente_risposta int(3) NOT NULL
PRIMARY KEY (id_risposta),

```

---

```
FOREIGN KEY (id_post_origine) REFERENCES post(id_post) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (id_post_risposta) REFERENCES post(id_post) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (id_utente_risposta) REFERENCES utente_registrato(id_utente) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;
```

**9. scrittura\_post (id\_scrittura,data\_scrittura,id\_post\_scritto,id\_utente\_scrittore)**

```
CREATE TABLE scrittura_post (
id_scrittura int(3) NOT NULL AUTO_INCREMENT,
data_scrittura DATE NOT NULL,
id_post_scritto int(3) NOT NULL,
id_utente_scrittore int(3) NULL,
PRIMARY KEY (id_scrittura),
FOREIGN KEY (id_post_scritto) REFERENCES post(id_post) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (id_utente_scrittore) REFERENCES utente_registrato (id_utente) ON
DELETE SET NULL ON UPDATE CASCADE
) ENGINE = InnoDB;
```

**10. messaggi (id\_messaggio,titolo\_messaggio,corpo\_messaggio)**

```
CREATE TABLE messaggi (
id_messaggio int(3) NOT NULL AUTO_INCREMENT,
titolo_messaggio varchar(255),
corpo_messaggio TEXT,
PRIMARY KEY (id_messaggio)
) ENGINE = InnoDB;
```

**11. messaggi\_inv\_ric (id\_invio,id\_messaggio ,id\_utente\_inv,id\_utente\_ric,data\_invio)**

```
CREATE TABLE messaggi_inv_ric (
id_invio int(3) NOT NULL AUTO_INCREMENT,
id_messaggio int(3) NOT NULL,
id_utente_inv int(3) NULL ,
```

---

```

id_utente_ric int(3) NULL,
data_invio DATE NOT NULL,
PRIMARY KEY (id_invio),
FOREIGN KEY (id_messaggio) REFERENCES messaggi (id_messaggio) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY (id_utente_inv) REFERENCES utente_registrato (id_utente) ON DELETE
SET NULL ON UPDATE CASCADE,
FOREIGN KEY (id_utente_ric) REFERENCES utente_registrato (id_utente) ON DELETE
SET NULL ON UPDATE CASCADE
) ENGINE = InnoDB;

```

- 12. utente non registrato** (id\_utente\_nr, nome\_utente) // *tabella indicante le utenze non registrate assegnate in automatico dal sistema quando si visita il sito*

```

CREATE TABLE utente_non_registrato (
id_utente_nr int(3) NOT NULL AUTO_INCREMENT,
nome_utente char(20) NOT NULL,
PRIMARY KEY (id_utente_nr)
) ENGINE = InnoDB;

```

- 13. sotto\_progetto** (id\_sotto\_progetto, dir\_lavoro\_spj, data\_ultima\_modifica, stato\_avanzamento\_spj, specifiche\_spj)

```

CREATE TABLE sotto_progetto (
id_sotto_progetto int(3) NOT NULL AUTO_INCREMENT,
dir_lavoro_spj varchar(40) NOT NULL,
data_ultima_modifica DATE NOT NULL,
stato_avanzamento_spj int NOT NULL,
specifiche_spj TEXT NOT NULL,
PRIMARY KEY (id_sotto_progetto)
) ENGINE = InnoDB;

```

- 14. progetto** (id\_project, dir\_lavoro\_radice, nome\_progetto, id\_ideatore, specifiche\_tecniche, data\_creazione) // *tabella indicante l'entità di progetto. L'ideatore eliminato dal DB non deve comportare l'eliminazione del progetto dal DB.*

```

CREATE TABLE progetto (
id_project int(3) NOT NULL AUTO_INCREMENT,
dir_lavoro_radice varchar(30) NOT NULL,

```

---

```
nome_progetto varchar(30) NOT NULL,  
id_ideatore int(3) NULL,  
specifiche_tecniche TEXT NOT NULL,  
data_creazione DATE NOT NULL,  
PRIMARY KEY (id_project),  
FOREIGN KEY (id_ideatore) REFERENCES utente_registrato (id_utente) ON DELETE  
SET NULL ON UPDATE CASCADE  
) ENGINE = InnoDB;
```

- 15. uso\_competenza** (id\_uso\_c, id\_comp, id\_sotto\_pj,) // *tabella indicante la relazione tra le competenze richieste da un sotto progetto e le competenze disponibili.*

```
CREATE TABLE uso_competenza (  
id_uso_c int(3) NOT NULL AUTO_INCREMENT,  
id_comp int(3) NOT NULL,  
id_sotto_pj int(3) NOT NULL,  
PRIMARY KEY (id_uso_c),  
FOREIGN KEY (id_comp) REFERENCES competenze (id_comp) ON DELETE  
CASCADE ON UPDATE CASCADE,  
FOREIGN KEY (id_sotto_pj) REFERENCES sotto_progetto (id_sotto_progetto) ON  
DELETE CASCADE ON UPDATE CASCADE  
)ENGINE = InnoDB;
```

- 16. incarico** (id\_incarico,nome\_incarico, tipo\_accesso) // *la tabella definisce la relazione tra la tipologia d'incarico e i permessi in lettura e scrittura, è da notare che l'esistenza di solo 5 tipologie di incarichi.*

```
CREATE TABLE incarico (  
id_incarico int(3) NOT NULL AUTO_INCREMENT,  
nome_incarico ENUM ('Project_Manager', 'Team_Manager', 'Team_Element',  
'Normal_User') NOT NULL,  
tipo_accesso ENUM ('r', 'w', 'rw'),  
PRIMARY KEY (id_incarico)  
)ENGINE = InnoDB;
```

- 17. incaricato** (id\_utente\_incaricato, id\_utente, id\_incarico) // *tabella indicante l'incarico di ogni utente. Ossia ad ogni utente viene assegnato un incarico in base alle proprie caratteristiche e competenze, questo non implica una partecipazione immediata al progetto*

---

*o al sotto progetto ma rende solo noto al sistema che l'utente ora è un utente incaricato, ossia un utente al quale è assegnabile un sotto progetto/ progetto. La partecipazione invece al sotto progetto o al sotto progetto viene determinata dalla tabella incarico\_di\_progetto e incarico\_di\_sotto\_progetto, è da notare che ogni utente può avere più incarichi assegnati, in quanto può essere project manager ma anche programmatore e quindi TE assumendo quindi più funzioni all'interno del progetto o del sotto progetto.*

```
CREATE TABLE incaricato (  
  id_utente_incaricato int(3) NOT NULL AUTO_INCREMENT,  
  id_utente int(3) NOT NULL,  
  id_incarico int(3) NOT NULL,  
  PRIMARY KEY (id_utente_incaricato),  
  FOREIGN KEY (id_utente) REFERENCES utente_registrato (id_utente) ON DELETE  
  CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (id_incarico) REFERENCES incarico (id_incarico) ON DELETE  
  CASCADE ON UPDATE CASCADE  
)ENGINE = InnoDB;
```

- 18. composizione\_pj** (id\_composizione\_pj,id\_progetto,id\_sotto\_pj) // la tabella mette in evidenza la scomposizione tra progetto e sotto progetto.

```
CREATE TABLE composizione_pj (  
  id_composizione_pj int(3) NOT NULL AUTO_INCREMENT,  
  id_progetto int(3) NOT NULL,  
  id_sotto_pj int(3) NOT NULL,  
  PRIMARY KEY (id_composizione_pj),  
  FOREIGN KEY (id_progetto) REFERENCES progetto (id_project) ON DELETE  
  CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (id_sotto_pj) REFERENCES sotto_progetto (id_sotto_progetto) ON  
  DELETE CASCADE ON UPDATE CASCADE  
)ENGINE = InnoDB;
```

- 19. incarico\_di\_progetto** (id\_proj\_management,id\_incarico,id\_pj) // la tabella mette in relazione il progetto con gli incarichi di project management assegnati agli utenti. Solo la classe project manager determinata nella tabella 'incarico' può essere assegnata ad un incarico progettuale quindi i dati contenuti nella tabella rappresentano il PCM ( la commissione dei manager ).

```
CREATE TABLE incarico_di_progetto (  
  id_proj_management int(3) NOT NULL AUTO_INCREMENT,  
  id_incarico int(3) NOT NULL,
```



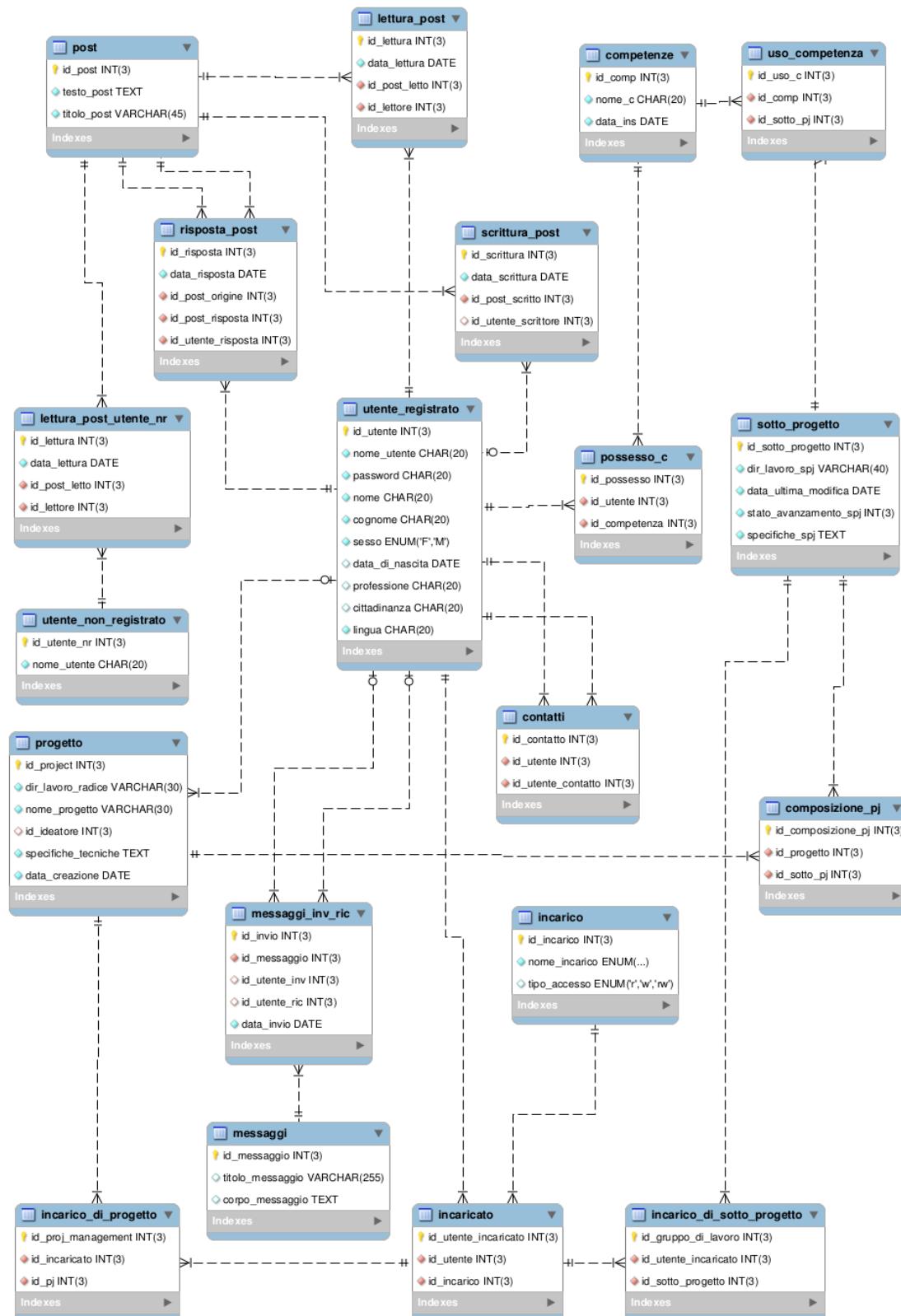
---

```
id_pj int(3) NOT NULL,  
PRIMARY KEY (id_proj_management),  
FOREIGN KEY (id_incarico) REFERENCES incaricato (id_utente_incaricato) ON  
DELETE CASCADE ON UPDATE CASCADE,  
FOREIGN KEY (id_pj) REFERENCES progetto (id_project) ON DELETE CASCADE  
ON UPDATE CASCADE  
)ENGINE = InnoDB;
```

- 20. incarico\_di\_sotto\_progetto ( id\_gruppo\_di\_lavoro, id\_incarico\_spj, id\_sotto\_progetto)//**  
*la tabella mette in relazione il sotto\_progetto con gli incarichi di project management  
assegnati agli utenti. Gli elementi della tabella sono i partecipanti al gruppo di lavoro che  
si occupa della programmazione e degli aspetti di un determinato sotto progetto, sono  
compresi pertanto tutti i Team Element ed il Team Manager.*

```
CREATE TABLE incarico_di_sotto_progetto (  
id_gruppo_di_lavoro int(3) NOT NULL AUTO_INCREMENT,  
id_incarico_spj int(3) NOT NULL,  
id_sotto_progetto int(3) NOT NULL,  
PRIMARY KEY (id_gruppo_di_lavoro),  
FOREIGN KEY (id_incarico_spj) REFERENCES incaricato (id_utente_incaricato) ON  
DELETE CASCADE ON UPDATE CASCADE,  
FOREIGN KEY (id_sotto_progetto) REFERENCES sotto_progetto (id_sotto_progetto) ON  
DELETE CASCADE ON UPDATE CASCADE  
)ENGINE = InnoDB;
```

Questo il diagramma elaborato con MySQL Work-Bench derivante dalle tabelle sopra descritte:



---

## Query di aggiornamento:

Vengono tradotte ora le principali Query di **aggiornamento** del DB.

*TUTTE le tabelle sono state popolate per scopo di **testing** con dati ed informazioni non reali.*

### 1. Inserimento Utente Registrato

```
INSERT INTO utente_registrato VALUES (' ', 'nome utente', 'password',  
'nome', 'cognome', 'sesso', 'data_di_nascita', 'professione',  
'cittadinanza', 'lingua');
```

Ogni utente inoltre può possedere dei contatti i quali vengono inseriti nel DB con:

```
INSERT INTO contatti VALUES (' ', 'id_utente', 'id_utente_contatto')
```

### 2. Inserimento Progetto

```
INSERT INTO progetto VALUES (' ', 'directory', 'nome del progetto',  
'id_ideatore', 'specifiche', 'data_creazione');
```

### 3. Inserimento Sotto\_Progetto

```
INSERT INTO sotto_progetto VALUES (' ', 'dir_lavoro_spj',  
'data_ultima_modifica', 'stato_avanzamento_spj',  
'specifiche_tecniche_sotto_progetto');
```

Per relazionare Progetto e Sotto progetto popoliamo la tabella composizione\_pj:

```
INSERT INTO composizione_pj VALUES (' ', 'id_progetto', 'id_sotto_pj')
```

### 4. Inserimento Post Domanda: inserimento post + inserimento scrittura\_post

```
INSERT INTO post VALUES (' ', 'testo_post', 'titolo_post');
```

```
INSERT INTO scrittura_post VALUES (' ', 'data_scrittura', 'id_post',  
'id_utente_scrittore')
```

### 5. Inserire Post Risposta

```
INSERT INTO post VALUES (' ', 'testo_post', 'titolo_post');  
INSERT INTO risposta_post VALUES (' ', 'data_scrittura',  
'id_post_origine', 'id_post_risposta')
```

### 6. Eliminazione di un Utente

```
DELETE FROM utente_registrato WHERE id_utente = 'id_utente'
```

### 7. Eliminazione di un Progetto

```
DELETE FROM progetto WHERE id_project = 'id_project'
```

### 8. Inserire un messaggio da un utente per un utente

```
INSERT INTO messaggi VALUES (' ', 'titolo_messaggio', 'corpo_messaggio');
```

```
INSERT INTO messaggi_inv_ric VALUES (' ', 'id_messaggio', 'id_utente_inv',  
'id_utente_ric', 'data_invio');
```

---

## 9. Eliminazione di un Sotto\_Progetto

```
DELETE FROM sotto_progetto WHERE id_sotto_progetto='id_sotto_progetto';
```

## 10. Inserire un Incarico: *// esistono 4 tipologie di incarichi basilari che un utente può assumere quali: project manager, team manager, team element, normal user.*

```
INSERT INTO incarico VALUES ('', 'nome_incarico', 'tipo_accesso');
```

Ad ogni utente, a seconda delle proprie competenze e delle proprie richieste può quindi essere assegnato ad un incarico tra quelli previsti dal progetto / sotto-progetto.

Vengono pertanto usate le seguenti tabelle:

```
INSERT INTO incaricato VALUES ('', 'id_utente', 'id_incarico') // la tabella assegna uno o più incarichi agli utenti che quindi possono svolgere più funzioni all'interno del progetto o collaborare a più di un progetto.
```

```
INSERT INTO incarico_di_sotto_progetto VALUES ('', 'id_incarico_spj', 'id_sotto_progetto') // la tabella assegna ad ogni utente con un incarico un sotto progetto, costituendo il GdL.
```

```
INSERT INTO incarico_di_progetto VALUES ('', 'id_incarico', 'id_pj') // la tabella assegna uno o più incarichi agli utenti che quindi possono svolgere più funzioni all'interno del progetto o collaborare a più di un progetto. Costituisce quindi la commissione di PM (PMC).
```

## 11. Inserire una Competenza: *// la competenza può essere inserita o in fase di registrazione al sito in quanto già posseduta dall'utente oppure in fase di definizione del progetto in quanto necessaria allo sviluppo.*

```
INSERT INTO competenze VALUES ('', 'nome_competenza', 'data_inserimento')
```

La relazione Competenze – Utenti è espressa nella tabella competenze\_c i cui dati vengono inseriti con :

```
INSERT INTO competenze_c VALUES ('', 'id_contatto', 'id_competenza')
```

Le competenze richieste dal sotto progetto vengono invece espresse nella tabella uso\_competenza e i dati vengono inseriti con:

```
INSERT INTO uso_competenza VALUES ('', 'id_comp', 'id_sotto_pj')
```

---

## Query di interrogazione al DB

### 1. Visualizzare l'elenco degli utenti iscritti al sito

```
SELECT nome , cognome FROM utente_registrato;
```

### 2. Visualizzare le competenze che ciascun utente ha dichiarato di possedere.

```
SELECT  
nome_utente , competenze.nome_c ' competenza posseduta '  
FROM  
utente_registrato , competenze , possesso_c  
WHERE  
(possesso_c.id_utente = utente_registrato.id_utente AND  
possesso_c.id_competenza=competenze.id_comp);
```

E' stato necessario correlare le seguenti tabella tramite una JOIN:

- utente\_registrato: che contiene i nomi\_utente e gli id\_utente
- competenze: che contiene un elenco generale di competenze definite da utenti e progetti
- possesso\_c: che contiene le competenze possedute da ogni utente

### 3. Trovare le competenze richieste per i sotto progetti dal progetto id\_project='1'

```
SELECT  
DISTINCT competenze.nome_c ' competenze richieste dal progetto 1 '  
FROM uso_competenza, competenze  
WHERE id_sotto_pj IN  
(SELECT id_sotto_pj FROM composizione_pj WHERE id_progetto=1) AND  
(uso_competenza.id_comp = competenze.id_comp);
```

E' stato necessario correlare le seguenti tabella tramite una JOIN:

- uso\_competenza: associa il sotto progetto alle competenze necessarie al suo sviluppo.
- competenze: che contiene un elenco generale di competenze definite da utenti e progetti

### 4. Trovare quali utenti possiedono le competenze richieste dal progetto id\_project=2 La tabella deve contenere il nome\_utente , la competenza posseduta dall'utente richiesta da almeno uno dei sotto progetti definiti per il progetto con id\_project specificato=2.

```
SELECT  
nome_utente 'Username', competenze.nome_c 'Competenza richiesta posseduta '  
FROM utente_registrato, possesso_c, competenze  
WHERE (competenze.id_comp=possesso_c.id_competenza)  
AND( utente_registrato.id_utente=possesso_c.id_utente)  
AND possesso_c.id_competenza IN  
(SELECT id_comp from uso_competenza  
WHERE  
id_sotto_pj IN  
(SELECT id_sotto_pj from composizione_pj WHERE id_progetto=2));
```

- 
- 5. Trovare i sotto progetti in un qualunque progetto che richiedono le competenze specifica posseduta dall'utente con id = 10**

```
SELECT DISTINCT
id_sotto_pj 'id sotto progetti utente 10'
FROM uso_competenza
WHERE id_comp IN
(SELECT id_competenza FROM possesso_c WHERE id_utente=10);
```

- 6. Visualizzare i titoli dei Post scritti dall'utente con id = 4 . Deve essere visualizzato il titolo del post, la data in cui è stato scritto ed il nome dello username che lo ha composto ( user4).**

```
SELECT
titolo_post, data_scrittura,nome_utente
FROM post
JOIN scrittura_post
JOIN utente_registrato
WHERE id_post=id_post_scritto
AND id_utente_scrittore=id_utente
AND id_utente= 4;
```

Per risolvere la query viene effettuata una JOIN applicata a tre tabelle quali:

- utente\_registrato : per l'id\_utente
- post: che contiene i dati sui post in particolare d'interesse il titolo e la data
- scrittura\_post : che relazione il post con l'utente che l'ha scritto

- 7. Trovare tutti i Messaggi aventi il destinatario con id= 2. Deve essere visualizzato il nome utente del ricevente la data d'invio ed il titolo del messaggio.**

```
SELECT nome_utente , data_invio , titolo_messaggio
FROM utente_registrato
JOIN messaggi_inv_ric
JOIN messaggi where id_utente_ric = 2
AND messaggi.id_messaggio = messaggi_inv_ric.id_messaggio
AND utente_registrato.id_utente = messaggi_inv_ric.id_utente_ric;
```

Parimenti a quanto effettuato con la query precedente viene applicata una JOIN alle tre tabelle che contengono i dati di interesse e vengono filtrate le righe tramite le condizioni espresse dall'AND logico.

- 
8. Visualizzare nomi e cognomi dei contatti posseduti dall'utente con id = 14. Deve comparire su 4 colonne l'id dell'utente che possiede il contatto e i contatti posseduti ( nome , cognome e username ).

```
SELECT
contatti.id_utente, nome , cognome , nome_utente
FROM
utente_registrato JOIN contatti
WHERE
id_utente_contatto = utente_registrato.id_utente
AND contatti.id_utente=14;
```

9. Trovare quanti utenti partecipano ad un progetto ( id\_project = 1 )

Per sviluppare la query è necessario contare i partecipanti ad ogni singolo sotto progetto di cui è composto il progetto e sommare a questo valore il numero di partecipanti al progetto che fanno parte della PMC ( Project Management Committè ). Inoltre visto che un utente può partecipare a più sotto progetti o essere anche PM oltre che progettista è necessario effettuare un controllo sui dati. Possiamo pensare per risolvere il problema di creare una unione di tutti i dati provenienti dalle tabelle singole e di raggrupparli per id utente in modo da essere sicuri che questi non vengano ripetuti e contare il numero dei valori trovati. L'operatore UNION usato senza parametri restituisce in un'unica istanza il risultato senza duplicazioni.

```
SELECT
COUNT(DISTINCT totale) 'Totale partecipanti al progetto'
FROM
((SELECT id_utente as totale
FROM incaricato JOIN incarico_di_sotto_progetto
JOIN sotto_progetto
JOIN composizione_pj
WHERE incaricato.id_utente_incaricato =
incarico_di_sotto_progetto.id_utente_incaricato
AND incarico_di_sotto_progetto.id_sotto_progetto =
sotto_progetto.id_sotto_progetto
AND composizione_pj.id_sotto_pj =
incarico_di_sotto_progetto.id_sotto_progetto
AND composizione_pj.id_progetto=1)
UNION
(SELECT id_utente as totale
FROM incaricato
JOIN incarico_di_progetto WHERE incaricato.id_utente_incaricato =
incarico_di_progetto.id_incaricato
AND id_pj=1)) as a;
```

La query mette in relazioni differenti tabelle:

- incaricato: che fornisce gli ID degli utenti che hanno il determinato incarico
- incarico\_di\_sotto\_progetto: che fornisce gli utenti che, possedendo il determinato incarico, fanno parte del sotto\_progetto
- incarico\_di\_progetto: fornisce gli id degli utenti che fanno parte della PMC all'interno del progetto.

---

E' da notare anche l'uso di DISTINCT all'interno della query, infatti gli utenti possono partecipare a più sotto progetti con differenti incarichi, producendo quindi una ripetizione del proprio id\_utente che in viene in questo modo filtrato per non determinare errori nel calcolo.

**10. Determinare quali utenti partecipano a più di un unico sotto-progetto ed indicare a quanti sotto-progetti.**

```
SELECT
nome_utente, totale ' N.ro progetti a cui partecipa'
FROM (SELECT utente_registrato.nome_utente, count(nome_utente) AS totale
FROM utente_registrato
JOIN incarico_di_sotto_progetto
JOIN incaricato WHERE
incaricato.id_utente_incaricato =
incarico_di_sotto_progetto.id_utente_incaricato
AND incaricato.id_utente = utente_registrato.id_utente GROUP BY
(nome_utente)) AS a WHERE totale>1;
```

Le tabelle utilizzate per l'implementazione della query sono:

- utente\_registrato: per ottenere le informazioni riguardanti lo username
- incarico\_di\_sotto\_progetto: che indica la relazione tra gli utenti e l'incarico che gli è stato assegnato all'interno del progetto.
- incaricato: la tabella assegna un possibile incarico ad ogni utente, ma non implica che l'utente faccia già parte di un sotto-progetto di lavoro.

La query fa inoltre uso di ALIAS tramite la funzione AS utile per selezionare gli utenti che partecipano a più di un progetto.

Un'altra funzione utilizzata è COUNT utile per contare il numero di partecipazioni degli utenti ad ogni progetto.

**11. Determinare l'elenco dei Project Manager per un determinato progetto (id\_project=1) Devono essere visualizzati nomi e cognomi e il nome del progetto di appartenenza.**

```
SELECT
nome , cognome,nome_progetto
FROM utente_registrato
JOIN progetto JOIN incaricato JOIN incarico_di_progetto
WHERE incarico_di_progetto.id_pj=1 AND
incarico_di_progetto.id_incaricato=incaricato.id_utente_incaricato
AND utente_registrato.id_utente=incaricato.id_utente
AND progetto.id_project=incarico_di_progetto.id_pj;
```

La query viene risolta con una JOIN usando le tabelle di interesse per estrapolare i dati richiesti.



---

**12. Determinare l'elenco dei sotto progetti che fanno parte di un progetto.**

La query deve associare al nome di un progetto le specifiche del sotto progetto.

```
SELECT progetto.nome_progetto, sotto_progetto.specifiche_spj
FROM progetto
JOIN composizione_pj
JOIN sotto_progetto
WHERE progetto.id_project=composizione_pj.id_progetto
AND sotto_progetto.id_sotto_progetto=composizione_pj.id_sotto_pj
AND progetto.id_project=composizione_pj.id_progetto;
```

**13. Determinare l'avanzamento totale dei progetti come media degli avanzamenti dei sotto progetti, deve essere visualizzato il nome del progetto ed il suo stato di avanzamento.**

```
SELECT
nome_progetto ,
AVG(sotto_progetto.stato_avanzamento_spj) 'stato % avanzamento del
progetto'
FROM progetto , composizione_pj, sotto_progetto
WHERE
(progetto.id_project = composizione_pj.id_progetto AND
sotto_progetto.id_sotto_progetto= id_sotto_pj) GROUP BY (nome_progetto);
```

Per ottenere i dati richiesti ho necessità di relazionare tre tabelle:

- progetto: che contiene il nome del progetto
- sotto\_progetto: che contiene le percentuali di avanzamento inserite dal Team Manager
- composizione\_pj: che contiene la relazione tra il progetto e i sotto progetto di appartenenza

Una volta effettuata la procedura JOIN ho necessità di raggruppare i dati di interesse ( ossia i nomi di progetto uguali derivanti dalla JOIN ) per poter effettuare un calcolo di AVG ossia di media statistica.

**14. Determinare l'elenco gli utenti che lavorano ad un determinato progetto ( progetto + sotto-progetto con project id=1)**

```
SELECT nome, cognome , utente
FROM
(SELECT id_utente as utente
FROM incaricato JOIN incarico_di_sotto_progetto
JOIN sotto_progetto
JOIN composizione_pj
WHERE      incaricato.id_utente_incaricato =
incarico_di_sotto_progetto.id_utente_incaricato
AND      incarico_di_sotto_progetto.id_sotto_progetto =
sotto_progetto.id_sotto_progetto
AND      composizione_pj.id_sotto_pj =
incarico_di_sotto_progetto.id_sotto_progetto
AND      composizione_pj.id_progetto=1
UNION
SELECT id_utente
FROM incaricato
JOIN incarico_di_progetto WHERE incaricato.id_utente_incaricato =
```

---

```
incarico_di_progetto.id_incaricato  
AND id_pj=1) AS a
```

```
JOIN utente_registrato  
WHERE utente_registrato.id_utente=a.utente;
```

La query sfrutta parzialmente i risultati della query 9 che è stata modificata per poter fornire i risultati richiesti.

#### **15. Determinare gli utenti che NON hanno delle competenze specifiche per il sotto-progetto**

Per la risoluzione della query ho necessità di confrontare le competenze possedute dagli utenti con le competenze richieste dal progetto e definire un insieme degli elementi sono posti al di fuori dell'intersezione dei due insiemi.

La funzione NOT IN esclude i valore che non sono di interesse, filtrando solo i valori che non vengono utilizzati da nessun sotto progetto ( è da notare che le competenze necessarie al progetto , come il project manager , compaiono tra le competenze non utilizzate in quanto formalmente non vengono utilizzate dal sotto progetto ma piuttosto dal progetto di appartenenza.

```
SELECT  
DISTINCT nome_utente,nome_c ' competenza non usata '  
FROM utente_registrato  
JOIN possesso_c  
JOIN competenze  
JOIN  
((SELECT DISTINCT id_competenza as competenza_f  
FROM possesso_c  
WHERE id_competenza NOT IN  
(SELECT id_comp FROM uso_competenza)) as a)  
WHERE utente_registrato.id_utente=possesso_c.id_utente  
AND possesso_c.id_competenza=a.competenza_f  
AND competenze.id_comp=possesso_c.id_competenza;
```

#### **16. Determinare l'elenco degli utenti che hanno delle competenze comuni**

```
SELECT  
utente_registrato.nome_utente as user1, nome_c as competenzad,  
uti.nome_utente as user2  
FROM utente_registrato  
JOIN competenze  
JOIN utente_registrato as uti  
JOIN  
(SELECT  
possesso_cf1.id_utente as utente, possesso_cf2.id_utente as utente2,  
possesso_cf1.id_competenza as competenza  
FROM  
possesso_c possesso_cf1 , possesso_c possesso_cf2  
WHERE possesso_cf1.id_competenza = possesso_cf2.id_competenza
```

---

```
AND possesso_cf1.id_utente > possesso_cf2.id_utente ) as a
ON utente_registrato.id_utente=a.utente
AND competenze.id_comp=a.competenza
AND uti.id_utente=a.utente2;
```

La query si sviluppa con un caso particolare di JOIN che utilizza una singola tabella che viene pertanto correlata con se stessa e in questa determinata casistica viene definita **SELF JOIN**. Supponiamo pertanto di poter lavorare su una coppia di tabelle identiche la cui seconda viene determinata dall'uso di un ALIAS, queste vengono comparate tramite una JOIN per poter ottenere i risultati richiesti. Resta da risolvere il problema dei doppi, se infatti correliamo due tabelle identiche otteniamo il doppio dei risultati richiesti, questa problematica viene risolta tramite :

```
possesso_cf1.id_utente > possesso_cf2.id_utente
```

che comporta una maggior restrizione nei parametri di paragone, se infatti avessimo usato l'operatore <> avremmo ottenuto i risultati corretti ma questi sarebbero stati espressi in forma doppia.

## 17. Determinare i titoli di post il cui messaggio contenga una determinata stringa

```
SELECT titolo_post
FROM post
WHERE testo_post LIKE '%presentazione%';
```

La query può essere risolta con una semplice SELECT che permette di ricercare all'interno del testo del post la parola o la stringa indicata.

Ora vogliamo determinare una **procedura** che possa produrre un warning su alcune parole che non vogliamo siano presenti all'interno dei nostri post.

```
CREATE PROCEDURE `cercatesto`(IN testo_ricerca TEXT)
BEGIN
IF (SELECT COUNT(titolo_post) FROM post WHERE testo_post LIKE
CONCAT('%',testo_ricerca,'%')) = 0
THEN SET @T='NORMAL';
ELSE SET @T='WARNING';
END IF;
SELECT @T;
END
```

Per richiamare la procedura possiamo usare la funzione CALL

```
CALL cercatesto ('babbeo')
```

Restituisce il valore di NORMAL o WARNING a seconda che il testo sia stato trovato all'interno di tutti i post in ricerca, anche solo in una sotto stringa.

Notiamo l'uso di alcune funzioni quali SET per l'attribuzione delle variabili in cui viene salvato lo stato del risultato e la funzione di COUNT utile qualora la SELECT restituisse

---

più di un unico risultato.

La funzione CONCAT() inoltre viene utilizzata per poter ottenere un parametro di ricerca testuale da utilizzare con la funzione LIKE.

## 18. Check constraints con uso di TRIGGERS

In MySQL non è possibile per ora implementare i CHECK CONSTRAINT in modo diretto ma è possibile determinare dei TRIGGER in modo che i dati possano essere validati<sup>10</sup>. Vengono definiti quindi dei check constrain triggers che permettono di bloccare l'inserimento di un dato nella tabella contatti qualora ad esempio si tentasse di aggiungere un utente X ai propri contatti. Ossia la tabella deve fare in modo che l'id dell'utente che inserisce il contatto sia sempre diverso dall'id dell'utente che si voglia far apparire nei propri contatti.

```
DELIMITER $$
CREATE TRIGGER controllo_contatti BEFORE INSERT ON contatti FOR EACH ROW
BEGIN
DECLARE dummy INT;
IF NEW.id_utente_contatto = NEW.id_utente THEN
SELECT 'il contatto non è inseribile per questo utente'
INTO dummy FROM information_schema.tables;
END IF;
END; $$
```

```
DELIMITERS $$
CREATE TRIGGER controllo_contatti_bu BEFORE UPDATE ON contatti FOR EACH
ROW
BEGIN
DECLARE dummy INT;
IF NEW.id_utente_contatto = NEW.id_utente THEN
SELECT 'il contatto non è inseribile per questo utente'
INTO dummy FROM information_schema.tables;
END IF;
END; $$
```

Ecco i risultati dopo l'elaborazione:

```
mysql> insert into contatti values (50,1,1);
ERROR 1172 (42000): Result consisted of more than one row
mysql> insert into contatti values (50,1,10);
Query OK, 1 row affected (0.03 sec)
```

Come è possibile verificare è effettuabile l'inserimento di id\_utete e id\_utente\_contatto solo se sono differenti tra loro, se invece i due valori sono uguali l'inserimento viene bloccato dal trigger.

---

10 Guy Harrison, MySQL Stored Procedure Programming, O'RELLY

---

## Progettazione fisica

Il livello fisico di progettazione consiste nell'implementare il livello logico e le strutture dati su supporti fisici come ad esempio hard disk e DVD. I dati godono inoltre della proprietà di indipendenza fisica al punto che questi possono essere modificati senza modificare i programmi che li gestiscono.

Le tecniche utilizzate per le strutture fisiche dei file possono essere suddivise principalmente in primarie e secondarie<sup>11</sup>.

**Le strutture primarie** dipendono dalla posizione fisica del dato che viene salvato sul supporto in un determinato modo per poter facilitare la ricerca delle informazioni, si possono basare su tecniche quali: sequenziale, strutture ad accesso calcolato (*hash*). Le strutture primarie sono quindi quelle che contengono propriamente i dati in sé.

**Le strutture secondarie** vengono realizzate tramite alberi e uso di indici. Tali strutture permettono di cambiare la posizione di un record semplicemente cambiando la posizione del puntatore e non dipendono quindi dalla posizione fisica del dato sul disco. Le strutture secondarie sono quindi tutte quelle strutture che favoriscono l'accesso ai dati senza tuttavia contenerli. Particolari strutture ad albero di accesso tramite indice sono ad esempio i B-tree e i B+tree.

Il goal per ottenere una buona ottimizzazione è dettato essenzialmente dalla riduzione del numero di accessi al disco la qualcosa è derivante dal numero di relazioni coinvolte.

**Le organizzazioni sequenziali** sono caratterizzate da una disposizione consecutiva dei record, quindi andare a ricercare l'elemento N all'interno della struttura implica che tutti i valori fino a N-1 vengano visitati fino alla lettura del dato, il costo della ricerca dell'informazione è quindi di tipo lineare, pertanto sono strutture poco efficienti e che poco si adattano ad esempio per una SELECT. Tale struttura ha tuttavia il pregio di avere una bassissima complessità nelle operazioni di inserimento in quanto l'informazione da immagazzinare viene semplicemente collegata consecutivamente all'ultima informazione utili. Qualora i valori siano invece ordinati la ricerca assumerebbe una complessità logaritmica  $\log_2(N)$ . Tuttavia i costi di ordinamento sono molto elevati da mantenere, in particolare per strutture molto dinamiche, che necessitano di un continuo aggiornamento, non sono sufficienti a compensare il basso costo di ricerca.

Per risolvere la problematica vengono **utilizzati gli indici**, con cui possiamo accedere in un modo ordinato ai record del file senza la struttura fisica del file sia organizzata in un modo ordinato. Possiamo considerare un indice di un libro che ci permette di ritrovare la pagina di interesse in mezzo ad altre informazioni.

Strutture ad albero particolari che fanno uso di indici sono ad esempio i B-Tree e i B+tree.

Possiamo effettuare alcuni test sul database utilizzando la tabella *UTENTE\_REGISTRATO*, i cui dati vengono inseriti tramite l'operazione:

```
INSERT INTO utente_registrato VALUES (' ', 'nome utente', 'password', 'nome',  
'cognome', 'sesso', 'data_di_nascita', 'professione', 'cittadinanza', 'lingua');
```

E' possibile verificare dalla figura a pag. 59 che l'entità *UTENTE\_REGISTRATO* ha innumerevoli relazioni con altre entità e viene pertanto utilizzata in un elevato numero di query per la

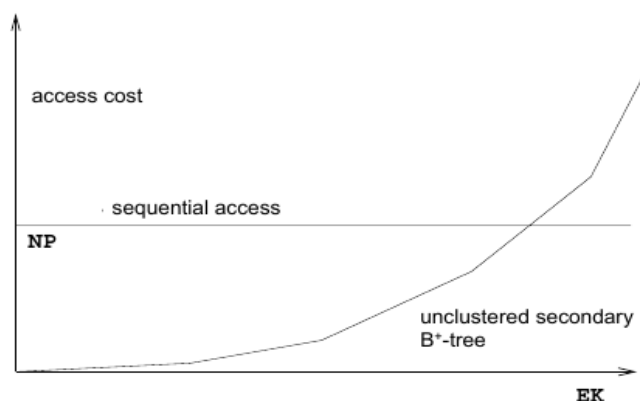
---

<sup>11</sup> Autori vari, *Basi di dati Modelli e linguaggi di interrogazione*, McGraw-Hill, 2009

determinazione delle informazioni richieste, le query richieste che impiegano tale tabella sono sia di tipo selettivo che di tipo ranged. La convenienza nell'utilizzo di un indice dipende dal numero di righe selezionate quindi a volte può essere più efficiente l'uso di uno scan sequenziale.

Come possiamo verificare dal grafico in immagine<sup>12</sup> se il numero di chiavi da ricercare aumenta in modo eccessivo allora il costo da pagare dopo un certo valore diventa superiore con l'uso di indici.

Come già detto il costo di un'operazione d'inserimento è sempre poco conveniente nel caso di indice mentre in un caso di accesso sequenziale basta inserire il dato in coda, pertanto un'organizzazione ad indici non risulta essere sempre il caso più ottimale da utilizzare.



Da quanto detto risulta necessario utilizzare o meno l'indice in base ai valori selezionati per ottenere una corretta ottimizzazione delle risorse, comprendendo le modalità di utilizzo dei dati, i tipi di query e la frequenza con cui le informazioni vengono richieste.

Analizziamo una query utilizzata per il nostro DB:

*QUERY 1 : Visualizzare l'elenco degli utenti iscritti al sito*

```
SELECT nome , cognome FROM utente_registrato;
```

L'entità `utente_registrato` richiede un determinato spazio per la registrazione pari a **154 byte** rispettando le seguenti equivalenze.

id\_utente: 3\* int = 3 Byte  
nome\_utente: 20 char = 20 byte  
password: 20 char = 20 byte  
nome: 20 char = 20 byte  
cognome: 20 char = 20 byte  
sesso: 1 char = 1 byte  
data\_di\_nascita: = 10 byte  
professione: 20 char = 20 byte  
cittadinanza: 20 char = 20 byte  
lingua: 20 char = 20 byte

<sup>12</sup> Lezione 5.5 corso DBIS del prof Alessandro Aldini lastrina 12/12

---

Applicando una dimensione di pagina pari a quella del settore del disco, dalle informazioni ricavate sulla macchina nel caso specifico, considero la pagina di una dimensione di 512 byte.

```
root@ku-64:/home/user1# fdisk -l /dev/sda
```

```
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 testine, 63 settori/tracce, 60801 cilindri, totale 976773168 settori
Unità = settori di 1 * 512 = 512 byte
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Ogni pagina può quindi contenere  $512/154 = 3.3 \sim 3$  record ( valore approssimato)  
La tabella contiene (al momento della scrittura) un valore pari a 18 utenti non registrati, quindi  $18/3 = 6$  **pagine** (NB: questo risultato verrà utilizzato per i successivi calcoli ).

La richiesta **di un solo valore** all'interno della SELECT per un accesso **di tipo sequenziale** non indicizzato avrebbe pertanto un costo di accesso di 6, ossia pari al valore delle pagine per cui viene richiesto l'accesso.

La ricerca di tutte le informazioni richiederebbe un accesso sequenziale ad ogni singola pagina per la lettura di tutte le informazioni, possiamo quindi verificare che sarebbe possibile ottenere anche tutti i record con una complessità di  $NP=6$ .

Supponiamo ora di voler **applicare un indicizzazione di tipo B+tree** in cui le informazioni sono salvate solo sulle foglie dell'albero, mentre nei nodi interni troviamo i puntatori per arrivare alle foglie.

La ricerca avviene all'interno dell'albero usando come indice **id\_utente**. Viene supposto inoltre, per semplicità di cose che le informazioni vengono salvate in modo uniformemente distribuito e che quindi ogni record abbia la stessa probabilità di poter essere trovato. Tale problematica richiede di applicare la formula di *Cardenas* che vedremo in seguito.

Per poter determinare l'altezza dell'albero, valore necessario per il calcolo del risultato finale dobbiamo calcolare il **numero di foglie dell'albero**.

Il calcolo può essere effettuato, usando un indice primario, con l'uso della formula<sup>13</sup>:

The diagram shows the formula for calculating the number of leaves (NL) in a B+tree. The formula is: 
$$NL = \frac{NR \cdot (len(k) + len(p))}{D \cdot u}$$
 Arrows point from descriptive text to the variables in the formula: 'number of records' points to NR; 'key length and pointer length (byte)' points to len(k) and len(p); 'leaf capacity' points to D; 'leaf utilization' points to u.

In cui definiamo:

$u$  ( la percentuale di utilizzazione delle foglie) = 80%

$NR = 18$  numero di record presenti

$len(k) = 154$  byte ( lunghezza della chiave precedentemente calcolata )

$len(p) = 1$  byte ( dimensione del puntatore)

$D$  = capacità della foglia ossia pari a 512 byte

procedendo con il calcolo otteniamo il valore di  $NL = 18 \cdot (154 + 1) / 512 \cdot 0.8 = 6,8 \rightarrow 7$  **foglie**

---

<sup>13</sup> Lezione 5.5 corso DBIS del prof Alessandro Aldini lastrina 7/12

Definendo il grado dell'albero pari a 3 ( $g=3$ ) è possibile calcolare l'altezza minima e massima dell'albero B+tree :

$$h \geq 1 + \log_{2g+1}(NL) \rightarrow \text{nel caso minimo } h = 1 + \log_7(7) = 2$$

$$h \leq 2 + \log_{g+1}(NL/2) \rightarrow \text{nel caso massimo } h = 2 + \log_4(3.5) = 3$$

pertanto l'albero avrà un'altezza compresa tra :  $2 < h < 3$

Il primo caso, in cui  $h=2$  ossia quello in cui l'altezza dell'albero è la più bassa possibile implica che i nodi siano pieni, il secondo caso, in cui  $h=3$  implica invece che i nodi siano semi vuoti.

Se assumiamo che il file contiene NR record in NP pagine, con ER il numero di record che vogliamo ricercare. Come abbiamo ipotizzato, supponiamo che i record siano equamente distribuiti tra le pagine e non siano sistemati sequenzialmente.

Grazie alla formula di Cardenas  $\phi(ER, NP) = NP \cdot (1 - (1 - 1/NP)^{ER})$  otteniamo la probabilità che un record compaia in una delle pagine.

Per quanto riguarda un indice che supponiamo unclustered e di indicizzazione primaria rispetto al parametro id\_utente di interesse, il costo di accesso al record singolo risulta il seguente (viene applicato l'uso della formula di Cardenas in quanto i record sono sparsi nelle pagine):

$$Ca_r = (h_A - 1) + \lceil f_F \cdot NL_A \rceil + \lceil \phi(ET_F, NP_r) \rceil$$

Dove

$h_A$  : altezza dell'albero

$f_F$  : filter factor pari a  $E_k/N_k \rightarrow$  ossia il numero di chiavi expected rispetto al numero di chiavi totali

$NL_A$  : le foglie dell'albero

$ET_F$  :  $\lceil NT_r \cdot f_F \rceil$  ( il numero delle tuple divise in NP pagine )

$NP_r$  : il numero di pagine

Qualora volessimo determinare il costo di accesso di un solo record  $Ca_r$  effettuando le opportune sostituzioni otterremmo.

$$(2-1) + \lceil 1/18 \cdot 7 \rceil + \phi(NT_r \cdot f_F, 6)$$

$$(2-1) + \lceil 1/18 \cdot 7 \rceil + \phi(18 \cdot 1/18, 6)$$

$$(1) + \lceil 7/18 \rceil + \phi(1, 6) \rightarrow \lceil \phi(ER, NP) = NP \cdot (1 - (1 - 1/NP)^{ER}) \rceil$$

$$(1) + \lceil 7/18 \rceil + \lceil 6(1 - (1 - 1/6)^1) \rceil$$

$$(1) + \lceil 7/18 \rceil + \lceil 6(1/6) \rceil$$

$$(1) + \lceil 7/18 \rceil + 1$$

$$2 + 7/18 = 43/18 = 2,4 \rightarrow \text{otteniamo un costo minimo di 3}$$

Possiamo notare la differenza rispetto alla ricerca ranged sequenziale, infatti se dovessimo ricercare tutti i risultati tramite un B+tree contenuti nella tabella utente\_registrato il costo della ricerca di un singolo record sarebbe da applicare al numero di record totali ( 18 ) ottenendo quindi un costo di accesso di:  $\lceil 1 \rceil + \lceil 7 \rceil + 6 \cdot \lceil 1 - (1 - 1/6)^{18} \rceil = 14$

Supponiamo esaminare i costi di accesso per un'operazione di JOIN, possiamo prendere ad esempio



---

con la seguente: **Visualizzare nomi e cognomi dei contatti posseduti dall'utente con id = 14.**

```
SELECT
contatti.id_utente, nome , cognome , nome_utente
FROM
utente_registrato JOIN contatti
WHERE
id_utente_contatto = utente_registrato.id_utente
AND contatti.id_utente=14;
```

Per l'analisi dei costi una JOIN esistono differenti tipologie di algoritmi utilizzabili tra cui i noti sono: Nested-loops join , Nested-loops join with local predicates, Nested-block join , Sort-merge join , Simple-hash join , Hash-partitioned join , Join index, Kd-tree.

Il più semplice da implementare, il Nested-loop Join prevede due cicli *for* annidati in cui vengono selezionati tutti i record della prima tabella per poter effettuare un paragone con i record della seconda tabella.

Nell'esempio la tabella “contatti” risulta essere la tabella interna ed “utente\_registrato” la tabella più esterna del loop, ogni volta che i dati voluti coincidono su entrambe le tabelle il record trovato viene salvato su una tabella a parte che poi sarà quindi in grado di mostrare le informazioni volute.

Nel caso dell'esempio l'elemento che fa da filtro è il valore 14 per gli id\_utente su entrambe le tabelle.

Per l'uso di una Nested-loop join analizzando i costi di scansione di entrambe le tabelle è facile verificare che in questo caso se non utilizziamo gli indici ma un accesso sequenziale i costi sono proporzionali al prodotto del numero di pagine considerate per la prima e per la seconda relazione:

$Ca_r + ETfr * Ca_s$ , ossia la lettura del numero di pagine del loop esterno sommate alla lettura delle tuple lette dalle pagine della relazione interna opportunamente filtrate. Il numero di pagine Npr della relazione esterna per gli utenti registrati ( 18 utenze ) è di 6, il numero di pagine per i contatti è facilmente calcolabile in quanto:

$id\_contatto = int * 3 = 3 \text{ byte}$

$ic\_utente = int * 3 = 3 \text{ byte}$

$id\_utente\_contatto = int * 3 = 3 \text{ byte}$

per un totale di 9 byte.

$512/9 = 57 \rightarrow 57/25 = 1 \text{ PAGINA}$  ( è sufficiente una pagina la contenimento di tutti i record )

Qualora fossimo in presenza di indici clustered per la relazione esterna ed avessimo un filtro del 10% sui dati avremmo che il costo di accesso

$Car = h-1 + NT*fr + NP*Fr = 3-1 + 18*0,1 + 7*0,1 = 2 + 2 + 1 = 5$

Il costo di accesso per il secondo indice Cas (interno ) in caso di equi join dipenderebbe dall'albero  $h-1 + 1 + 1 =$  ossia il costo dell'albero + 1 lettura di NL + 1 lettura di NP = ( supponendo  $h=3$  )  $2 + 1 + 1 = 4$ .

Ora seguendo la formula sopra espressa  $Ca_r + ETfr * Ca_s$  (in cui  $ETfr = NTr * fr = 18*0,1 = 2$ ) otteniamo un costo di accesso totale pari a  $5 + 2*4 = 13$

---

## Sviluppo di un modulo C tramite l'uso delle API di MySQL

Viene ora realizzato un modulo in C in grado di effettuare alcune operazioni all'interno del DB tra quelle già sviluppate precedentemente in MySQL.

Il programma è stato sviluppato in C e per la compilazione prevedere l'uso di un Makefile

```
#makefile
CC = gcc
PARMAIN = -ansi -Wall -o
PAROGG = -Wall -Werror -c
LIBFILEC = opensourcepj.c
LIBFILEO = opensourcepj.o
OUT = main_ospj
INCLUDES = ${shell mysql_config --cflags}
LIBS = ${shell mysql_config --libs}
tutti:
    @echo
    @echo Inizio compilazione files
    @echo
    $(CC) $(PAROGG) $(OUT).c $(INCLUDES)
    $(CC) $(PAROGG) $(LIBFILEC) $(INCLUDES)
    $(CC) $(PARMAIN) $(OUT) $(OUT).o $(LIBFILEO) $(LIBS)
rimuovi:
    @echo
    @echo Pulizia file oggetto...
    @echo
    rm -f *.o
```

La compilazione del programma fa uso di uno dei tool di mysql: `mysql_config` che viene usato per identificare parametri e directory delle librerie.

Possiamo notare che il make accetta due opzioni, `tutti` e `rimuovi`. L'opzione `rimuovi` serve per eliminare i file oggetto creati con la compilazione.

Una volta compilato il file eseguibile sarà nominato: `main_ospj`

Per la sua esecuzione basterà quindi digitare `./main_ospj`

---

## Interfaccia Utente

```
root@ku-64:/home/utente1/progettodb# ./main_ospj
```

Informazioni accesso al DB

- indirizzo\_host (localhost per server locale): localhost
- nome\_utente: root
- password: \*\*\*\*\*
- nome del DB da impiegare: opensourcepj

Connessione al DB opensourcepj avvenuta con successo

Vengono inizialmente richieste le informazioni di accesso al DB, è da tenere presente che il programma si incentra sull'utilizzo delle query e l'implementazione delle API di MySQL, quindi sono state tralasciate le problematiche attinenti la sicurezza delle informazioni ed il controllo dei dati inseriti. La password che qui vediamo asteriscata in realtà verrà digitata in chiaro e c'è solo un minimo controllo sulla correttezza di tutte le informazioni inserite.

Una volta inseriti i dati correttamente il programma sarà connesso al DB :Connessione al DB opensourcepj avvenuta con successo.

Verranno quindi mostrate le possibili scelte effettuabili all'interno del programma. Ho differenziato una scelta per l'inserimento dei dati e tre scelte per la lettura di query.

1:REGISTRARE UN NUOVO UTENTE

2:INSERISCI UN NUOVO PROGETTO

3:VISUALIZZARE L'ELENCO DEGLI UTENTI REGISTRATI

4:VISUALIZZARE GLI UTENTI CHE POSSIEDONO LE COMPETENZE RICHIESTE DAL PROGETTO

5:VISUALIZZARE LO STATO DI AVANZAMENTO DEI SOTTO PROGETTI

6:USCITA GESTIONE DATA BASE

Effettuare una scelta tra quelle indicate:

**L'opzione numero 1** prevede l'inserimento delle informazioni per la registrazione dell'utente, come specificato nell'esempio seguente:

Inserisci i dati utente:

- nome\_utente: user18
- password: pass18
- nome: Tom
- cognome: Robin
- sesso: M
- inserisci data di nascita (YYYY-MM-DD): 1991-12-21
- professione: studente
- cittadinanza: francese
- lingua: francese

---

**L'opzione 2** invece prevedere l'inserimento di un nuovo progetto.

L'operazione prevedere l'aggiornamento di 3 tabelle del DB quali:

- *tabella progetti*: con i dati del nuovo progetto aggiornati.
- *tabella incaricato*: in quanto all'utente ideatore del progetto deve essere assegnato l'incarico di PM
- *tabella incarico\_di\_progetto*: al progetto deve essere assegnato l'id dell'utente incaricato.

Non sapendo l'id dell'utente che si è registrato per poter effettuare l'inserimento del nuovo progetto, il dato viene simulato e passato alla funzione di gestione del DB.

Inserire simulazione utente id\_creatore da passare alla funzione: 13

In seguito devono essere inserite le informazioni specifiche del progetto quali : il nome e le specifiche tecniche iniziale per avviare la progettazione.

Inserisci il nome del progetto: social web

Inserisci le specifiche tecniche del progetto [termina con INVIO]: social network per bambini

Il resto delle informazioni necessarie alla tabella progetto vengono calcolate dal sistema per il successivo inserimento quali:

- directory di lavoro radice: che assume il valore del nome del progetto (/home/progetti/nome\_progetto)
- data di creazione: che assume il valore della data locale.

Il programma poi ci avvisa del corretto inserimento delle tabelle riguardanti l'incaricato e l'incarico\_di\_progetto:

L'utente ideatore assume l'incarico di PM

Assunzione dell'incarico avvenuta con successo, record salvato in tabella

L'utente incaricato PM viene inserito nell'elenco dei PM del progetto

Per quanto riguarda le query di ricerca delle informazioni viene proposta la schermata di visualizzazione degli utenti:

Selezionando **l'opzione 3** viene visualizzato l'elenco degli utenti disponibili nel DB.

ELENCO DEGLI UTENTI DISPONIBILI DEL DB:

|    |          |        |             |             |   |            |                 |            |            |
|----|----------|--------|-------------|-------------|---|------------|-----------------|------------|------------|
| 1  | utente1  | pass1  | mario       | rossi       | M | 1991-01-12 | studente        | italiana   | italiana   |
| 2  | utente2  | pass2  | luca        | bianchi     | M | 1985-02-10 | studente        | italiana   | italiana   |
| 3  | utente3  | pass3  | stefano     | verdi       | M | 1982-07-25 | ricercatore     | italiana   | italiana   |
| 4  | utente4  | pass4  | eleonora    | gianfranchi | F | 1989-01-24 | designer        | italiana   | italiana   |
| 5  | utente5  | pass5  | frank       | smith       | M | 1990-10-10 | studente        | inglese    | inglese    |
| 6  | utente6  | pass6  | sarah       | green       | F | 1991-12-24 | studente        | inglese    | inglese    |
| 7  | utente7  | pass7  | alberto     | alberti     | M | 1995-05-03 | studente        | italiano   | italiano   |
| 8  | utente8  | pass8  | marie       | martin      | F | 1993-07-10 | project manager | francese   | francese   |
| 9  | utente9  | pass9  | michel      | bernard     | M | 1993-01-10 | programmatore   | francese   | francese   |
| 10 | utente10 | pass10 | sebastien   | dubois      | M | 1993-12-12 | programmatore   | francese   | francese   |
| 11 | utente11 | pass11 | fredrich    | muller      | M | 1994-04-05 | studente        | tedesco    | tedesco    |
| 12 | utente12 | pass12 | hans        | neumann     | M | 1976-01-30 | project manager | tedesco    | tedesco    |
| 13 | utente13 | pass13 | himisci     | minamoto    | M | 1989-07-10 | web designer    | giapponese | giapponese |
| 14 | utente14 | pass14 | elena       | teodori     | F | 1988-05-25 | traduttrice     | italiana   | italiana   |
| 15 | utente15 | pass15 | peter       | kock        | M | 1994-04-27 | studente        | tedesco    | tedesco    |
| 17 | utente16 | pass16 | piergiorgio | casalegno   | M | 1997-01-13 | studente        | italiana   | italiano   |
| 18 | user18   | pass18 | Tom         | Robin       | M | 1991-12-21 | studente        | francese   | francese   |
| 19 | user19   | pass19 | tiziano     | slavi       | M | 1990-11-11 | programmatore   | italiana   | italiano   |

---

**L'opzione 4** serve invece per trovare gli utenti che hanno almeno una competenza dichiarata che è anche necessaria allo sviluppo del progetto.

Effettuare una scelta tra quelle indicate:4

SCEGLIERE id\_PROGETTO:1

ELENCO DEGLI UTENTI CHE POSSIEDONO LE COMPETENZE SPECIFICHE PER IL PROGETTO ID\_1:

|          |                    |
|----------|--------------------|
| utente1  | programmatore C    |
| utente9  | programmatore C    |
| utente10 | programmatore Java |
| utente10 | programmatore C++  |

Il DB restituisce l'elenco dei nome\_utente che possiedono le competenze richieste.

Come possiamo verificare il programma richiede l'inserimento del progetto per cui vogliamo trovare le informazioni.

**L'opzione 5** serve a trovare gli stati di avanzamento dei progetti, come media degli stati di avanzamento dei vari sotto\_progetti di cui fanno parte.

STATO DI AVANZAMENTO DEI SOTTO PROGETTI DB:

|         |         |
|---------|---------|
| disegna | 10.0000 |
| scrivi  | 25.0000 |
| suona   | 15.0000 |

Come possiamo verificare il DB restituisce il nome del progetto ed il suo stato di avanzamento espresso in percentuale.

**L'ultima opzione, la 6** , serve ad uscire dal programma di gestione ed a chiudere le connessione con il DB.

---

## Codice Sorgente e sviluppo del programma

### File: main\_ospj.c

Il file ha il compito principale di richiamare il modulo di gestione del DB.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "opensourcepj.h"
int main (int argc, char *argv[])
{
    char host_name[20]="localhost";
    char user_name[20];
    char password[20];
    unsigned int port_num = 0;
    char *socket_name = NULL; /* uso del default*/
    char db_name[20];
    printf("\nInformazioni accesso al DB\n");
    printf ("- indirizzo_host (localhost per server locale): ");/*nome host*/
    scanf ("%19s",host_name);
    printf ("- nome_utente: ");
    scanf ("%19s",user_name);
    printf ("- password: ");
    scanf ("%19s",password);
    printf("- nome del DB da impiegare: ");
    scanf ("%19s",db_name);
    /* applicazione della funzione modulo di utilizzo del DB*/
    gestione_db ( host_name, user_name, password , port_num , socket_name , db_name , 0 );
    return 0; /*ritorno main */
}
```

---

## File: opensourcepj.c

Il file contiene le funzioni necessarie alla connessione con il DB e allo sviluppo delle query.

```
/*-----*/
/*file opensourcepj.c */
/*File di implementazione del database*/
/*-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <my_global.h> /* l'ordine di inclusione è importante*/
#include <my_sys.h> /* contiene portability macros */
#include <mysql.h> /* vengono definite le strutture dati principali di mysql */
#include <time.h>

/*dichiarazione funzioni*/
void gestione_db ( char *, char *, char *, unsigned int, char *, char *, unsigned int);
void disconnessione_DB (MYSQL *);
int menu_opzioni(MYSQL *);
void crea_utente(MYSQL *);
void visualizza_elenco_utenti (MYSQL *);
void visualizza_spj(MYSQL *);
void visualizza_stato_avanzamenti(MYSQL *);
void termina_con_errore (MYSQL *);
void inserisci_progetto(MYSQL *,int);
/*TERMINE DICHIARAZIONE FUNZIONI*/

/*FUNZIONE PRINCIPALE DI UTILIZZO DEL DB, VIENE RICHIAMATO IL MENU DAL QUALE E'POSSIBILE LA SCELTA DI QUERY*/
void gestione_db ( char *host_name, char *user_name, char *password , unsigned int port_num , char *socket_name , char
*db_name , unsigned int flags )
{
    MYSQL *conn;
    conn = mysql_init (NULL); // inizializza il gestore della connessione
    int scelta=0;
    if (conn == NULL)
    { fprintf (stderr, "Errore: inizializzazione fallita, impossibile stabilire una connessione con il server\n"); }
    if (mysql_real_connect (conn, host_name, user_name, password, db_name, port_num, socket_name, flags) == NULL)
    { termina_con_errore(conn); }
    else printf (" Connessione al DB %s avvenuta con successo\n", db_name);
        while(scelta!=6) scelta=menu_opzioni(conn); /* menu a scelta fino a quando non viene premuto il tasto di
uscita*/
}

/*FUNZIONE DI SELEZIONE MENU ED ESECUZIONE QUERY*/
```

---

```
int menu_opzioni(MYSQL *conn)

{
    int id_creatore=0;
    int scelta;
    printf("\n 1:REGISTRARE UN NUOVO UTENTE\n");
    printf("\n 2:INSERISCI UN NUOVO PROGETTO\n");
    printf("\n 3:VISUALIZZARE L'ELENCO DEGLI UTENTI REGISTRATI\n");
    printf("\n 4:VISUALIZZARE GLI UTENTI CHE POSSIEDONO LE COMPETENZE RICHIESTE DAL PROGETTO\n");
    printf("\n 5:VISUALIZZARE LO STATO DI AVANZAMENTO DEI SOTTO PROGETTI\n");
    printf("\n 6:USCITA GESTIONE DATA BASE\n");
    printf("\n Effettuare una scelta tra quelle indicate:");
    scanf("%d",&scelta);

    switch(scelta)
    {
        case 1:
            crea_utente (conn);
            return 1;
        case 2:
            printf ("\nInserire simulazione utente id_creatore da passare alla funzione: ");
            scanf ("%d",&id_creatore);
            inserisci_progetto(conn, id_creatore);
            return 2;
        case 3:
            visualizza_elenco_utenti(conn);
            return 3;
        case 4:
            visualizza_spj(conn);
            return 4;
        case 5:
            visualizza_stato_avanzamenti (conn);
            return 5;
        case 6:
            printf("\nUscita dal programma e disconnessione DB\n");
            disconnessione_DB (conn);
            return 6;
    }
    return 0;
}
```

```
/*FUNZIONE DI INSERIMENTO DATI UTENTE DEL DB*/
```



---

```
void crea_utente(MYSQL *conn)
{
    /* dichiarazione variabili */
    char nome_utente[20];
    char password[20];
    char nome[20];
    char cognome[20];
    char sesso[2];
    char data_di_nascita[20];
    /*MYSQL_TIME data_ins; struttura DATA di mysql */
    char professione[20];
    char cittadinanza[20];
    char lingua[20];
    char insert_buffer [1024];/* strings supposto inserimento valori*/
    /*termine dichiarazione variabili*/

    /*INSERIMENTO VALORI UTENTE */
    printf("\nInserisci i dati utente: ");
    printf ("\n- nome_utente desiderato (nick_name): ");
    scanf ("%s",nome_utente);
    printf ("\n- password: ");
    scanf ("%s",password);
    printf ("\n- nome: " );
    scanf ("%s",nome);
    printf("\n- cognome: ");
    scanf ("%s",cognome);
    printf("\n- sesso: ");
    scanf ("%s",sesso);
    printf("\n- inserisci data di nascita (YYYY-MM-DD): \n");
    scanf ("%s",data_di_nascita);
    printf("\n- professione: ");
    scanf ("%s",professione);
    printf("\n- cittadinanza: ");
    scanf ("%s",cittadinanza);
    printf("\n- lingua: ");
    scanf ("%s",lingua);
    /* TERMINE INSERIMENTO VALORI*/

    /*parsing della stringa per l'inserimento del record in tabella*/
    sprintf (insert_buffer,"INSERT INTO utente_registrato VALUES ('
        '%s','%s','%s','%s','%s','%s','%s','%s','%s')",nome_utente,password,nome,cognome,sesso,data_di_nascita,professione,cittadinanza,lingua);

    if (mysql_query(conn, insert_buffer)) {termina_con_errore(conn);} /*inserimento dell record*/

}

/*visualizza l'elenco di tutti gli utenti*/
```

---

---

```

void visualizza_elenco_utenti (MYSQL *conn)
{
    MYSQL_RES *risultati;
    int num_campi = 0;
    MYSQL_ROW row;
    int i=0;

    printf ("\n ELENCO DEGLI UTENTI DISPONIBILI DEL DB:\n\n");
    if (mysql_query(conn, "SELECT * from utente_registrato")) {termina_con_errore(conn);}
    risultati=mysql_store_result(conn);
    num_campi= mysql_num_fields(risultati);
    while((row = mysql_fetch_row(risultati)))
    {
        for( i=0; i < num_campi; i++) /* stampa i risultati a schermo*/
        {
            printf("%16s|", row[i] ? row[i] : "NULL");
        }
        printf("\n");
    }
    mysql_free_result(risultati);/* libera i risultati trovati*/
}

/*-----*/

/* Trovare quali utenti possiedono le competenze richieste dal progetto
 * definito dall'utente. La tabella deve contenere il nome_utente ,
 * la competenza posseduta dall'utente  richiesta da almeno uno dei sotto
 * progetti definiti per il progetto richiesto*/
void visualizza_spj(MYSQL *conn)
{
    MYSQL_RES *risultati;
    int num_campi = 0;
    MYSQL_ROW row;
    int i=0;
    char insert_buffer[1024];/* stringa temporanea*/
    int numero_progetto=0;

    printf("\nSCEGLIERE id_PROGETTO:");
    scanf ("%d",&numero_progetto);
    printf ("\n ELENCO DEGLI UTENTI CHE POSSIEDONO LE COMPETENZE SPECIFICHE PER IL PROGETTO ID_%d:\n\n", numero_progetto);
    /*parsing della stringa di query*/
    sprintf (insert_buffer,"SELECT  nome_utente 'Username', competenze.nome_c 'Competenza richiesta posseduta' FROM
    utente_registrato,possezzo_c,competenze WHERE (competenze.id_comp=possezzo_c.id_competenza)
    AND( utente_registrato.id_utente=possezzo_c.id_utente) AND possezzo_c.id_competenza IN (SELECT id_comp from
    uso_competenza WHERE id_sotto_pj IN (SELECT id_sotto_pj from composizione_pj WHERE id_progetto=
    %d))",numero_progetto);

    if (mysql_query(conn, insert_buffer)) {termina_con_errore(conn);}

    /*formattazione dei risultati trovati*/

```

---

```

    risultati=mysql_store_result(conn);
    num_campi= mysql_num_fields(risultati);

    while((row = mysql_fetch_row(risultati)))
    {
        for( i=0; i < num_campi; i++) /* stampa i risultati a schermo*/
        {
            printf("%19s|", row[i] ? row[i] : "NULL");
        }
        printf("\n");
    }

    mysql_free_result(risultati);/* libera i risultati trovati*/
}
/*-----*/

/*visualizza lo stato di avanzamento dei progetti come
 * media dello stato di ogni sotto progetto che gli appartiene*/
void visualizza_stato_avanzamenti(MYSQL *conn)
{
    MYSQL_RES *risultati;
    int num_campi = 0;
    MYSQL_ROW row;
    int i=0;
    printf ("\n STATO DI AVANZAMENTO DEI SOTTO PROGETTI DB:\n");

    if (mysql_query(conn, "SELECT nome_progetto , AVG(sotto_progetto.stato_avanzamento_spj) 'stato % avanzamento del
    progetto'FROM progetto , composizione_pj, sotto_progetto WHERE (progetto.id_project = composizione_pj.id_progetto AND
    sotto_progetto.id_sotto_progetto= id_sotto_pj) GROUP BY (nome_progetto)"))

{termina_con_errore(conn);}/* TERMINA LA CONNESSIONE RIPORTANDO L'ERRORE*/
/*formattazione dei risultati trovati*/
    risultati=mysql_store_result(conn);
    num_campi= mysql_num_fields(risultati);

    while((row = mysql_fetch_row(risultati)))
    {
        for( i=0; i < num_campi; i++) /* stampa i risultati a schermo*/
        {
            printf("%19s|", row[i] ? row[i] : "NULL");
        }
        printf("\n");
    }

    mysql_free_result(risultati);/* libera i risultati trovati*/
}
void termina_con_errore (MYSQL *conn) /* stampa l'errore riportato da mysql ed esce*/

```

---

```

{
    fprintf (stderr, "Errore %u (%s): %s\n", mysql_errno (conn), mysql_sqlstate(conn), mysql_error (conn));
    mysql_close(conn);
    exit(1);
}
/*-----*/
/*funzione di disconnessione senza errore*/
void disconnessione_DB (MYSQL *conn)
{
    /* disconnessione */
    mysql_close (conn);
    printf ("\n Disconnessione avvenuta\n");
}
/*-----*/
/*funzione inserimento di un progetto:
 * per lo sviluppo di questa funzione è necessario
 * simulare l'id utente che richiede l'inserimento
 * inoltre l'utente che inserisce il progetto diviene
 * automaticamente il PM iniziale del progetto,
 * e va pertanto inserito nelle tabelle previste
 * assegnando l'incarico di PM*/

void inserisci_progetto(MYSQL *conn, int id_creatore)
{
    time_t tempo;
    struct tm time_struct;
    struct tm * time_struct_sup;
    char data_di_creazione[20];
    char dir_lavoro_radice [30];
    char specifiche_tecniche[4096];
    char nome_progetto[30];
    char insert_buffer[4200];
    char insert_buffer1[10];

    /*INSERIMENTO DEI DATI NEL DB*/
    time(&tempo); /* assegnazione del tempo attuale */

    time_struct_sup =localtime_r (&tempo,&time_struct);
    /*definizione della stringa DATA*/
    sprintf (data_di_creazione,"%d-%d-%d",(time_struct_sup->tm_year)+1900,((time_struct_sup->tm_mon)+1),
        time_struct_sup->tm_mday);

    /* inserimento del progetto con i dati */
    printf("\nInserisci il nome del progetto: ");
    while(getchar() != '\n');
    fgets (nome_progetto,29,stdin);
    nome_progetto[strlen(nome_progetto)-1]='\0';
    /*-----*/

```

---

```

/*la dir di lavoro viene creata in automatico grazie al nome del progetto*/
/* non ci possono pertanto essere due nomi progetto identici*/
sprintf (dir_lavoro_radice, "/home/progetti/%s", nome_progetto);
/*-----*/

/*inserimento specifiche tecniche*/
printf("\nInserisci le specifiche tecniche del progetto [termina con INVIO]: ");
fgets (specifiche_tecniche, 4095, stdin);
specifiche_tecniche[strlen(specifiche_tecniche)-1]='\0';
/*-----*/

/*inserisci dati in tabella progetto*/
sprintf(insert_buffer, "INSERT INTO progetto VALUES (' ', '%s',
'%s', '%d', '%s', '%s');", dir_lavoro_radice, nome_progetto, id_creatore, specifiche_tecniche, data_di_creazione);
if (mysql_query(conn, insert_buffer)) termina_con_errore(conn);
int ultimo_id_progetto = mysql_insert_id(conn); /* salvataggio dell'ultimo ID di progetto*/
/*-----*/

/*inserisci i dati nella tabella incaricato*/
printf("\nL'utente ideatore assume l'incarico di PM\n");
sprintf(insert_buffer1, "INSERT INTO incaricato VALUES (' ', '%d', '1');", id_creatore); /* l'utente prende l'incarico di
valore 1*/
printf ("%s", insert_buffer1);
if (mysql_query(conn, insert_buffer1)) termina_con_errore(conn);
else printf ("\nAssunzione dell'incarico avvenuta con successo, record salvato in tabella\n");
int ultimo_id_utente_incaricato = mysql_insert_id(conn);
/*-----*/

/*inserisci i dati nella tabella incarico_di_progetto*/
printf("\nL'utente incaricato PM viene inserito nell'elenco dei PM del progetto\n");
sprintf(insert_buffer1, "INSERT INTO incarico_di_progetto VALUES ('
', '%d', '%d')", ultimo_id_utente_incaricato, ultimo_id_progetto);
printf ("%s", insert_buffer1);
if (mysql_query(conn, insert_buffer1)) termina_con_errore(conn);
/*-----*/
}

```